Universidade de Aveiro
2022

Jorge Ricardo Pinto de Figueiredo Catarino

Firmware de Seleção de Célula para Redes 5G Não-Terrestres

Cell Selection Firmware for Non-Terrestrial 5G Networks

Universidade de Aveiro
2022

Jorge Ricardo Pinto de Figueiredo Catarino

Firmware de Seleção de Célula para Redes 5G Não-Terrestres

Cell Selection Firmware for Non-Terrestrial 5G Networks

"*The bounties of space, of infinite outwardness, were three: empty heroics, low comedy, and pointless death.*"

— Kurt Vonnegut

Universidade de Aveiro
2022

Jorge Ricardo Pinto de Figueiredo Catarino

Firmware de Seleção de Célula para Redes 5G Não-Terrestres

Cell Selection Firmware for Non-Terrestrial 5G Networks

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor Arnaldo Silva Rodrigues de Oliveira, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Dedicado aos meus colegas e às noites passadas em branco para entregar *aquele* projecto. Vocês sabem qual.

o júri / the jury

presidente / president	Prof. Doutor Daniel Nunes Corujo Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro
vogais / examiners committee	Prof. Doutora Marília Pascoal Curado Professora Catedrática da Universidade de Coimbra
	Prof. Doutor Arnaldo Silva Rodrigues de Oliveira Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

agradecimentos / acknowledgements

Queria agradecer em primeiro lugar à Universidade de Aveiro e a todos os docentes com que me cruzei durante o meu percurso, por todas as contribuições para o meu ensino.
Agradeço também à Ubiwhere e ao Instituto de Telecomunicações por me terem acolhido e proporcionado o ambiente e as condições ideais para a realização deste trabalho.
Ao meu orientador, o Professor Arnaldo Oliveira, e aos colaboradores Mestre André Duarte e Mestre Hélio Simeão, por todo o acompanhamento e também pelas inúmeras ideias e críticas, sem as quais este projeto não teria avançado da mesma forma.
Deixo um agradecimento aos meus colegas Daniel e Rui pelos momentos de entreajuda, especialmente quando as instalações partilhadas não queriam cooperar.
Do lado mais pessoal, não há espaço aqui para exprimir o quão endividado estou para com a minha mãe, por todos o esforços e pelo apoio incondicional.
Aos meus irmãos, Isabel, Rosa, Maria, Joaquim e ao meu sobrinho Ricardo, presenças constantes mesmo quando dá tudo para o torto.
Não posso deixar de agradecer aos amigos que fui conhecendo e que me acompanharam ao longo dos anos.
Aos que foram colegas de curso, que enfrentaram comigo todo o tipo de adversidades: Paulo, Pedro, Óscar, Ricardo, Daniela, Gustavo, Francisco, Daniel, Daniela, Inês, Filipe, Mariana, Miguel Lopes, Miguel Neves, Raquel, Tomás, André, Rafael, Carlos. Aos amigos que não me deixam de inspirar: Joana, Tiago, Alexandre, Filipe, Mafalda, Ricardo, Rodrigo, José, Miguel, Xavier, Bárbara, Daniel.
Por fim, agradeço ao Elvis, o meu gato, pelas várias tentativas sem sucesso de contribuir para este trabalho.

Palavras Chave 5G, Redes de Acesso, Satélite, Firmware, Open Source

Resumo

A integração de tecnologias satélite nas redes 5G vai permitir que estas se tornem mais seguras e omnipresentes, estendendo a cobertura de forma a abranger áreas remotas e tornando estas redes mais resilientes contra catástrofes naturais. É expectado que as redes não-terrestres venham a coexistir com as atuais redes terrestres, partilhando os mesmos requisitos. Por sua vez isto vai permitir que os terminais se conectem a ambos, abrindo assim novas possibilidades e casos de uso. Com esta dissertação pretende-se projetar um *firmware* que prepare estes dispositivos para tomar partido deste novo paradigma. Este *firmware* funciona como uma versão estendida, ciente do *backhaul*, do esquema de *cell selection*, de forma a que este possa decidir entre conectar *cells* terrestres ou não terrestres. Esta decisão é informada por métricas como a latência e a perda de pacotes da ligação, além dos indicadores de força de sinal tradicionais. Para a validação desta solução foi necessário a instalação de uma rede 5G *end-to-end* que incluísse tanto um *gNodeB* (gNB) capaz de simular atraso de propagação induzido pelas longas distâncias, tal como um nó terrestre. Esta instalação usa o *OpenAirInterface* (OAI), uma implementação da *stack* 5G. Usando esta *testbench*, a implementação do *firmware* projetado foi testada face a cenários de degradação da rede. Estes incluem, por exemplo, a falha total do gNB terrestre ou um aumento crescente da latência. Os resultados obtidos mostram que o uso deste *firmware* poderá ajudar a manter a qualidade de serviço de um terminal que o utilize.

Keywords 5G, Access Networks, Satellite, Firmware, Open Source

Abstract

The integration of satellite technology in 5G will enable networks to become more ubiquitous and reliable, extending coverage to previously underserved areas and making the network more resilient to natural catastrophes. The non-terrestrial networks (NTN) are expected to co-exist with the current terrestrial infrastructures, sharing much of the same requirements. This in turn will allow the User Equipment to connect to both, opening up new use cases and possibilities. The intent of this dissertation is to design a firmware that prepares these devices to take advantage of this new paradigm. This firmware implements an extended, radio access and backhaul-aware cell selection scheme, that chooses either to connect to terrestrial or non-terrestrial cells. The selection is based on metrics, such as, the latency and packet loss of the link, in addition to the traditional signal strength indicators. Testing the solution required deploying an end-to-end 5G network which includes not only a gNodeB (gNB) capable of simulating the propagation delay induced by long distances but also a terrestrial node. This deployment uses the OpenAirInterface (OAI) 5G software stack. With the use of this testbench, the implemented firmware was tested against key network degradation scenarios. These scenarios include, for example, the total failure of the terrestrial gNB and the steady increase of latency. The results show that this use of the firmware might help upkeep the quality of service for the User Equipment using it.

Contents

List of Figures

List of Tables

Glossary

2G	2nd Generation
3G	3rd Generation
3GPP	3rd Generation Partnership Project
4G	4th Generation
5G	5th Generation
AMF	Access and Mobility Management Function
API	Application Programming Interface
APN	Access Point Name
AVX2	Advanced Vector Extensions
ARFCN	Absolute Radio-Frequency Channel Number
AT	Attention
AUSF	Authentication Server Function
CN	Core Network
COTS	Commercial Off-The-Shelf
CP	Cyclic Prefix
CPU	Central Processing Unit
CU	Central Unit
DM-RS	Demodulation Reference Signal
DU	Distributed Unit
eMBB	Enhanced mobile broadband
EPC	Evolved Packet Core
FFT	Fast Fourier Transform
FPGA	Field-Programmable Gate Array
GEO	Geostationary Orbit
gNB	gNodeB
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GTP	GPRS Tunneling Protocol
HARQ	Hybrid Automatic Repeat Request
HO	Handover
IMT	International Mobile Telecommunications
IoT	Internet of Things
IP	Internet Protocol
IPTV	Internet Protocol Television
ISL	Inter-Satellite Link
ITU	International Telecommunication Union
LAN	Local Area Network
LDPC	Low-Density Parity-Check Code
LEO	Low Earth Orbit
LP	Linear Programming
LTE	Long Term Evolution
MAC	Medium Access Control
MEO	Medium Earth Orbit
MIB	Master Information Block
mMTC	Massive machine type communication
mmWave	Millimeter-Wave
N3IWF	Non-3GPP Interworking Function
NAS	Non-Access Stratum
NF	Network Function
NGAP	NG Application Protocol
NR	New Radio
NRF	NF Repository Function
NSA	Non-Standalone
NSSF	Network Slice Selection Function
NTN	Non-Terrestrial Network
OAI	Open Air Interface
OBP	Onboard Processor
ONF	Open Networking Foundation
O-RAN	Open Radio Access Network
OS	Operating System
PDCP	Packet Data Convergence Protocol
PDP	Packet Data Protocol
PDU	Protocol Data Unit
PHY	Physical
PLMN	Public Land Mobile Network
PRACH	Physical Random Access Channel
PS	Packet Switching
PTRS	Phase Tracking Reference Signal
QAM	Quadrature Amplitude Modulation
QoE	Quality of Experience
QoS	Quality of Service
QPSK	Quadrature Phase-Shift Keying
RACH	Random Access Channel
RAM	Random Access Memory
RAN	Radio Access Network
RAR	Random Access Response
RAT	Radio Access Technology
RF	Radio Frequency
RIC	RAN Intelligent Controller

RLC	Radio Link Control
RRC	Radio Resource Control
RRM	Radio Resource Management
RSRP	Reference Signal Received Power
RSRQ	Reference Signal Received Quality
RSSI	Received Signal Strength Indicator
RT	Real Time
RU	Radio Unit
RX	Reception
SA	Standalone
SatCom	Satellite Communications
SBA	Service-Based Architecture
SDAP	Service Data Adaptation Protocol
SDK	Software Development Kit
SDN	Sofware Defined Network
SD-RAN	Software Defined Radio Access Network
SIB	System Information Block
SIM	Subscriber Identity Module
SIMD	Single Instruction, Multiple Data
SMF	Session Management Function
SMS	Short Messaging Service
SRI	Satellite Radio Interface
SSE	Streaming SIMD Extensions
SSH	Secure Shell
TCP	Transmission Control Protocol
TSF	Télécoms Sans Frontières
UAS	Unmanned Aircraft System
UDM	Unified Data Management
UDN	Ultra Dense Networks
UDP	User Datagram Protocol
UDR	Unified Data Repository
UE	User Equipment
UPF	User Plane Function
URLLC	Ultra-Reliable and Low Latency Communication
USRP	Universal Software Radio Peripheral
VoIP	Voice over Internet Protocol

CHAPTER 1

Introduction

1.1 Context

Ever since the introduction of the 2nd Generation (2G), the first digital mobile network, a new generation has arrived roughly every 10 years [1]. The 5th Generation (5G) is the current pinnacle of this evolution, promising higher speeds and bandwidth, lower latency, and increased connection capacity. This technology will have a major impact across various industry sectors, such as transportation and healthcare, and it is estimated that it will have a global economic value of $13.3 trillion by 2035 [2].

To fulfill these demanding requirements, the 3rd Generation Partnership Project (3GPP) developed a new Radio Access Technology (RAT), aptly named 5G New Radio (NR). This air interface can use bands both sub-6GHz, and above 24 GHz, in the Millimeter-Wave (mmWave) frequency range [3] and is implemented by the gNB base stations, the centerpiece of 5G Radio Access Network (RAN). The architecture, proposed by 3GPP in Release 15 for these base stations, promotes flexibility by splitting it into three logical nodes, the Central Unit (CU), the Distributed Unit (DU) and the Radio Unit (RU) [4], allowing operators to adapt the architecture to suit the needs of their specific deployment scenarios in order to optimize performance [5] and to meet 5G demanding requirements.

Additionally, there have been efforts to produce new standards that will allow for more heterogeneous 5G networks and diverse RAT deployments. With the use of satellite communications, 5G mobile networks will be brought to space.

Satellite communications are not new. There have been attempts to build satellite constellations for communication purposes alone all the way back to the 90s, but those often ended in bankruptcy [6]. Recently there has been a renewed interest in these technologies, partly due to the efforts of tech companies such as Starlink, Amazon, OneWeb, and Telesat [6], but mostly because of what is arguably its greatest strength, coverage, and availability.

Internet access is increasingly becoming a necessity since the COVID-19 pandemic forced people to rely on it to access essential services such as healthcare and education or to go to work. In fact, 93% of adults in the U.S believe an interruption in the phone or Internet service during the outbreak could have an impact on daily life [7]. Despite this, the International Telecommunication Union (ITU) estimates that 2.9 billion people, roughly 37% of the world population, still have no Internet connection, and at least 390 million people are not covered by mobile broadband signal [8]. Even in developed countries, rural areas tend to have little to no broadband coverage due to geography making it more expensive to install the necessary infrastructure or simply because the low population density means less commercial return [9]. Satellites can serve as a solution to both these problems: by being in space, they become impervious to network cascading failures, becoming an alternative in cases of outages caused, for example, due to natural disasters; by providing wide coverage they can become a low-cost way of serving rural areas or even the open seas.

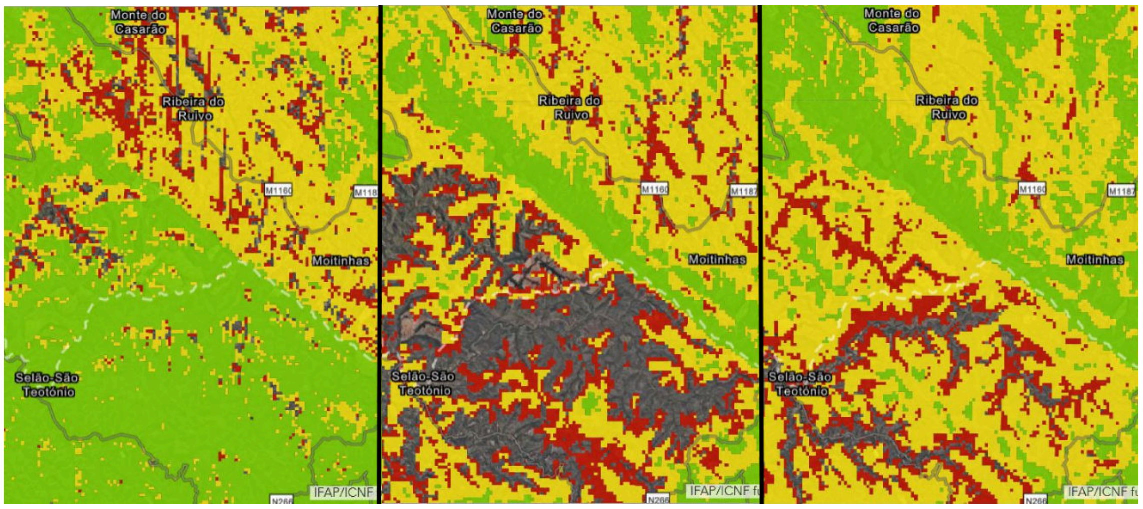


Figure 1.1: 4G coverage of a Portuguese rural area from three operators (Left to Right: NOS, MEO, Vodafone)[10].

For these reasons, 3GPP is already working for the inclusion of satellites in the 5G ecosystem [11] [12] [13], in order to tackle use cases where ubiquity, continuity, or scalability are required. Additionally, Lockheed Martin and Omnispace have an agreement to create the first hybrid 5G platform for both governmental and commercial use [14]. It is then relevant to start researching the possibility of a User Equipment (UE), that can take advantage of these architectures, having the ability to communicate both with terrestrial and NTNs.

Considering that these two networks have different characteristics and to avoid frequent re-selections, it is important to make this device capable of understanding when it is more beneficial to connect to a terrestrial base station or to a satellite, prioritizing speed and Quality of Service (QoS) while also ensuring smooth handovers.

1.2 Objectives

The main objective of this dissertation is to research, design, develop and validate a firmware capable of deciding when to switch between terrestrial and NTNs, based on the current status of the network. This switch should be quick so that it does not critically affect the service. This work has no prior basis, so it has to be built from scratch.

In order to propose a solid solution to the problem at hand, the first step will be to build a solid theoretical foundation on the concepts of 5G networks, including non-terrestrial networks, as well as satellite networks and elaborate a literature review that encapsulates current research on cell selection and handover algorithms as well as the latest developments on non-terrestrial 5G networks. Moreover, an assessment of current open-source 5G-related software has to be made. This work will allow for a practical understanding of the field and inform choices made later on.

The conditions when terrestrial to satellite handover should occur and how the firmware will behave both have to be determined. With this, the next step is to determine what devices support this type of firmware. Afterward, the solution can be developed. The developed software needs to be validated to see if it fits its requirements. To do so, a 5G testbench, that simulates an end-to-end network and includes an emulated satellite gNB needs to be developed. In addition, this testbench needs to be capable of simulating stressed network scenarios.

By extracting results of how the firmware behaves in these test scenarios, the main goals can be validated.

1.3 Contributions

This dissertation digs into the prospects brought by the insertion of non-terrestrial devices on the 5G mobile networks, enabling a UE to reap benefits from the new access type, achieving a more reliable, continuous service.

Academically, the research on the UE side of 5G NTN is still frankly underdeveloped, so there is still a lot of work to be done. The results of this work will provide an extended cell selection scheme that, based on signal-to-noise ratio and other metrics, can decide which gNB type to attach to at any given time, shedding light on how these processes may be adapted to a non-terrestrial reality. The dissertation will also provide insights on how NTN may be simulated using only open-source software.

Despite being mostly focused on practical results, this work provides an overview of the current research on this area of study.

1.4 Structure

The remain of this dissertation is organized as follows:

Chapter 2 features a Literature Review on 5G and NTN concepts, also including a state-of-the-art on 5G open-source software, mobility algorithms, and NTN research.

Chapter 3 details the design of both the eXtend5G firmware and testbench. It announces the methodology of this research and the scenario considered.

Chapter 4 presents the hardware and software choices that were made in the development of the solutions proposed in Chapter 3.

Chapter 5 describes the three tests designed to validate the solution and how they were executed, and showcases the results of said tests.

Chapter 6 provides a summary of the work done and suggests future work.

CHAPTER 2

Literature Review

This chapter starts with an introduction to 5G technology, with a focus on mobility. It presents the current open source implementations of 5G functions and their current state. Satellite concepts are introduced, finally leading to 5G Non-Terrestrial Networks (NTNs).

2.1 5G Networks

It is agreed that 5G is the fifth generation of cellular networks, but a more informative, universally accepted definition on what it actually represents does not exist [5]. To figure out what 5G is, a good place to start may be International Mobile Telecommunications (IMT)-2020 standards published by the ITU. These standards detail the minimum performance requirements a radio interface has to meet to qualify as an IMT-2020/5G technology. This hypothetical application should also fall in one of three scenarios, as defined in [15]. The categories are:

- Enhanced mobile broadband (eMBB) - this scenario concerns human-centric applications, consisting of improving the existing mobile broadband (e.g. 4th Generation (4G)) to achieve better performance, increasing speed, and reducing latency;
- Ultra-Reliable and Low Latency Communication (URLLC) - for applications with constricted requirements for availability, latency, or throughput;
- Massive machine type communication (mMTC) - represents use cases that require a large number of connected devices transmitting low data volumes (e.g. Internet of Things (IoT)).

Having these usage models in mind is helpful to begin comprehending the scope of 5G technologies and how they will be present in our lives. Some example applications are shown in Fig. 2.1.

While these usage scenarios may have their individual technical performance requirements, [15] offers a summary of IMT-2020 key requirements. These include:

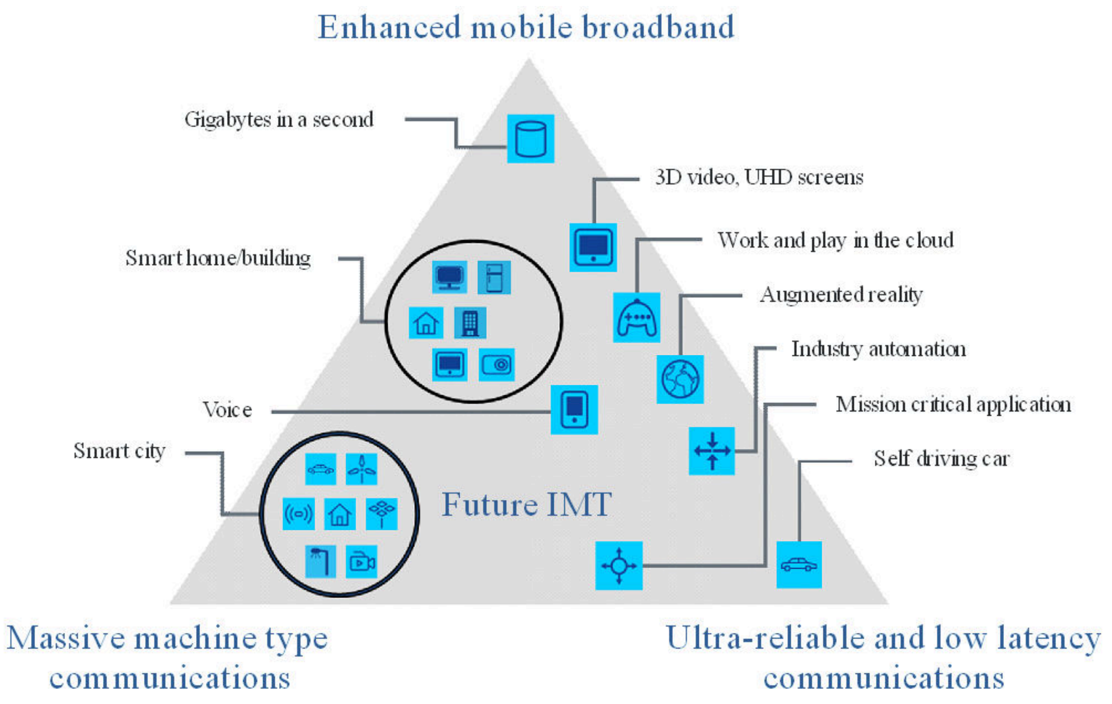

Enhanced mobile broadband
Gigabytes in a second
3D video, UHD screens
Smart home/building
Work and play in the cloud
Augmented reality
Industry automation
Voice
Mission critical application
Smart city
Self driving car
Future IMT
Massive machine type communications
Ultra-reliable and low latency communications

Figure 2.1: IMT-2020 usage scenarios (Source: [15]).

- User-experienced data rate: 100 Mbps - 1 Gbps
- Peak data rate: 20 Gbps
- Latency: <1 ms
- Mobility: 500 km/h

3GPP is a consortium of standard organizations, currently working on a set of technical standards to fulfill ITU requirements for 5G technologies. The organization uses a system of timed "Releases" designed to give developers a stable body of standards to facilitate the implementation of features [16]. While their standards do not represent 5G landscape as a whole, they are an important piece of it. It makes sense to develop solutions with them in mind in order to increase compatibility with other solutions deployed around the globe. There are already open source solutions that implement these standards, detailed in Section 2.2. Other standards organizations exist but usually tend to complement 3GPP standards other than to compete with them. One such organization, Open Radio Access Network (O-RAN) Alliance, is alluded to later in this section.

The rest of this section will detail different 5G access and mobility standards and aspects, as well some current research on handover algorithms.

2.1.1 Deployments

Currently, there are two main 5G deployments, Standalone (SA) and Non-Standalone (NSA).

In NSA, the 5G network shares the infrastructure with Long Term Evolution (LTE): the 5G RAN connects to an Evolved Packet Core (EPC). The purpose of this deployment is to leverage the existing infrastructure while the industry transits from 4G to 5G.

The SA deployment is when a 5G RAN connects to a 5G CN.

Using a NSA deployment will enable operators to start providing 5G connectivity for a lesser cost. Nonetheless, to take advantage of the main features of 5G networks, such as ultra-low latency and virtualization, a SA deployment is necessary [17].

2.1.2 5G Core Network (CN)

The 5G CN is the evolution of the 4G EPC. Its functions are similar to its predecessor: it is responsible for authentication, session management and traffic routing, among other features.

One of the main innovations is that 3GPP designed the 5G core using a Service-Based Architecture (SBA), where the various functions of the core are decoupled into smaller, independent services, called Network Functions (NFs), that are able to communicate with each other [18]. This architecture is more flexible and facilitates the introduction of new elements in the network. This architecture, as defined in [19], along with the essential NFs, can be seen in Fig. 2.2.

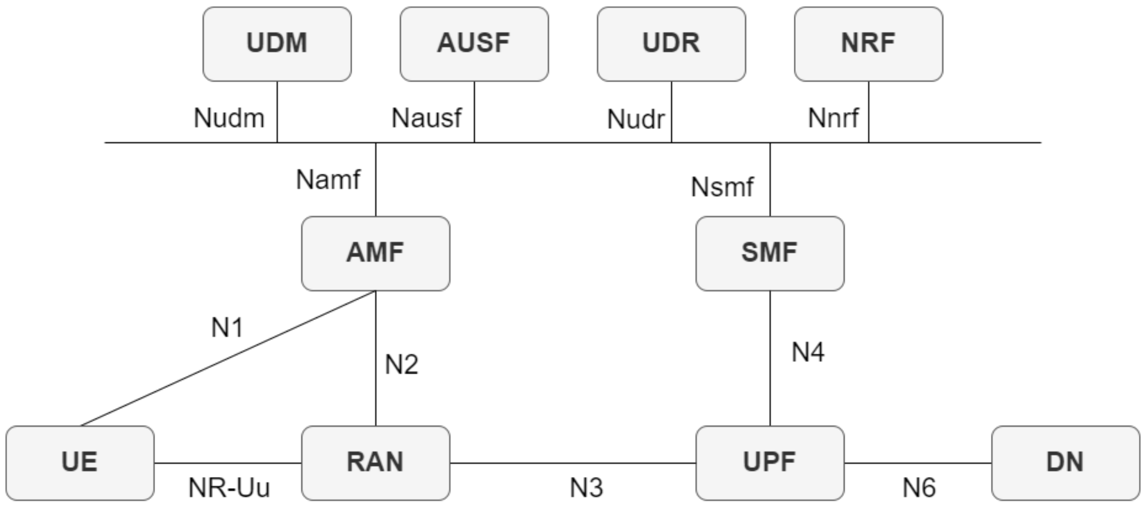


Figure 2.2: 5G CN Architecture ([19]).

The main NFs, and their roles, are:

- **Access and Mobility Management Function (AMF)** - The access point of the core network. It is responsible for authenticating and authorizing accesses. It also manages UE registration, connection and mobility.
- **Session Management Function (SMF)** - As the name may indicate, this module is responsible for the establishment and release of new sessions, when requested by an AMF.
- **Unified Data Management (UDM) and Unified Data Repository (UDR)** - This pair handles the storage and management of the subscriber profiles and other data related to authorization, which is provided to the AMF when requested.
- **Authentication Server Function (AUSF)** - An authentication service to enable the AMF to authenticate subscribers.
- **NF Repository Function (NRF)** - The purpose of this module is to discover, register and mantain the data related to the NFs that are currently part of the network. It communicates to each NF which services are currently available to be consumed.

- **User Plane Function (UPF)** - It is the bridge between the gNB and the UE it serves and the data networks, handling packet forwarding and routing.

2.1.3 NG-RAN

NG-RAN is a set of gNBs connected to the AMF and UPF through the NG interface and interconnected by the Xn interface [20]. It was defined by 3GPP to serve as the intermediary between the UE and the 5G-CN. The connection between a gNB and a UE is done via the Uu interface. As mentioned in Chapter 1, the gNB, as defined by 3GPP, consists of a gNB-CU and one or more gNB-DUs, connected via the F1 interface [21]. A NG-RAN node, as specified in [22], should implement these functionalities, among others:

- Radio Resource Management (RRM) functions, such as Radio Bearer Control and scheduling, both uplink and downlink, for the UE;
- Setup and release of a UE radio interface connection;
- Mobility control mechanisms for UEs both in CONNECTED and INACTIVE states;
- Support for UE measurement and measurement reports for mobility and scheduling;
- Routing of User Plane and Control Plane packets towards the AMF and UPF;
- Network Slicing;
- QoS Flow management.

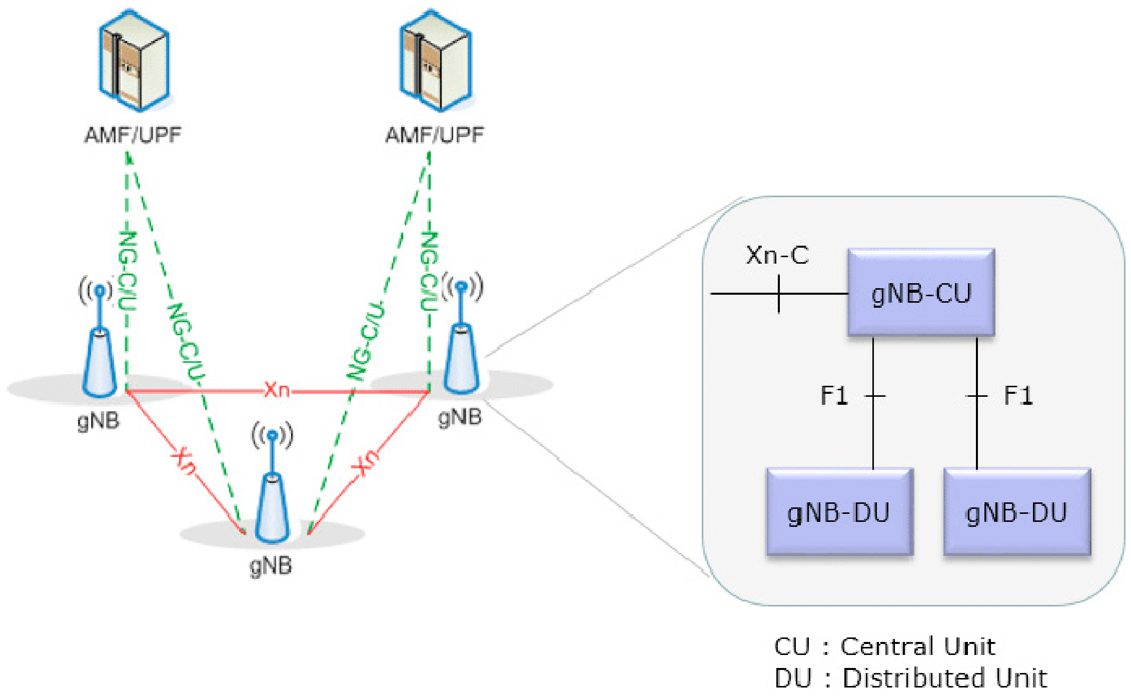


Figure 2.3: NG-RAN Logical Architecture (Adapted from [23], under CC BY-NC-ND).

In the standardized CU/DU split architecture, tipically the CU is responsible for the higher level protocol layers, Radio Resource Control (RRC), Packet Data Convergence Protocol (PDCP) and Service Data Adaptation Protocol (SDAP), while the DU covers lower level protocol layers, Physical (PHY), Medium Access Control (MAC) and Radio Link

Control (RLC) [24]. This architecture allows for centralized RRM: when one CU is able to control more coverage cells, it optimizes mobility mechanisms by reducing the number of inter-gNB handovers, which are significantly more costly, instead making more intra-gNB handover [25]. Another advantage of this architecture is having centralized RRM for improved performance. Despite this, the layer separation is not perfect as some higher layer functionalities with tighter latency requirements may need to be implemented in the DU, and it does not centralize any user plane functions [5].

Although this split was standardized by 3GPP, other options were considered as shown in Fig. 2.4. As a rule of thumb, the lower the gNB is split, the more layers are centralized, which in turn increases performance. As a consequence, there is also an increase in implementation complexity and bandwidth requirements overall.

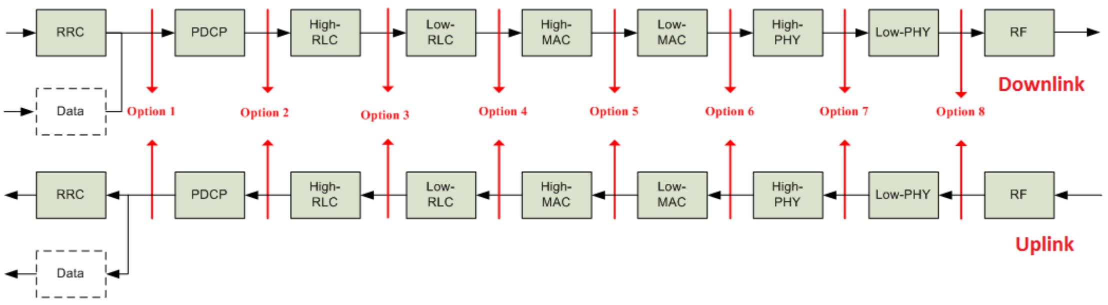


Figure 2.4: CU/DU functional split options (Source: [26]).

The O-RAN Alliance recognized the benefits of having a standardized low-level split, which led to the standardization of another split option [27]. The 7.2x split is an intra-PHY, which means the functions of the PHY layer are divided between two units. In the downlink, functions such as inverse Fast Fourier Transform (FFT), digital beamforming and Cyclic Prefix (CP) addition belong in the O-RU, while the rest is implemented in the O-DU, while in the uplink, FFT, CP removal, and beamforming is part of the O-RU and the rest implemented in the O-DU [27]. Amidst other advantages, this division of functions also guarantees that the 3GPP specific functions are all implemented in the DU, which makes the RU implementable in hardware and future-proofs it against eventual specification changes [27].

Other than the 7.2x split, the O-RAN specifications also expand on the 3GPP RAN architecture in other ways, adding new interfaces and logical nodes. One of these additions is the RAN Intelligent Controller (RIC). Split into two nodes, the RIC allows for the control and managements of nodes (e.g. gNB, DU, CU) connected via the E2 interface, bringing Sofware Defined Network (SDN) concepts to a mobile infrastructure. It does so by hosting xApps, essentially apps that collect information via the E2 interface in order to provide optimized services [28]. Use-cases that may be accomplished with the RIC and the use of xApps include traffic sterring and QoS/Quality of Experience (QoE) optimization [29].

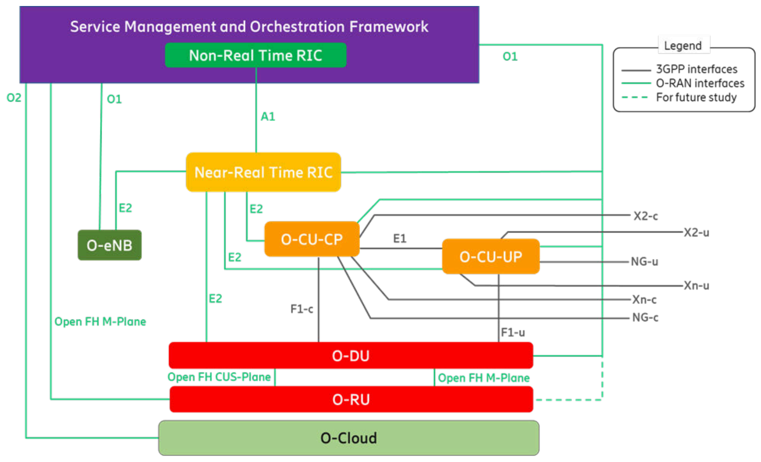


Figure 2.5: O-RAN Logical Architecture (Source: [28]).

2.1.4 Mobility in 5G

Mobility features exist to keep devices connected to the mobile networks while they move. This subsection will present some aspects of mobility, pertaining to 5G networks, that were deemed relevant to the research objectives.

RRC

The RRC is a layer 3 protocol for communication between UE and gNB. Some of the functions of this protocol are the broadcast of system information, the establishment of sessions between UE and the RAN and detection and recovery of radio link failures. It also implements some important mobility functions such as handover, cell selection and reselection and mobility between different RAT [30].

The protocol follows a state machine design, so the UE can be in one of the following states:

- RRC_IDLE
- RRC_INACTIVE
- RRC_CONNECTED

When a UE is turned on, and it is not attached to any cell, its state is RRC_IDLE. When in this state, the device is responsible for mobility through cell selection or re-selection, meaning that the UE itself will have to perform the required measurements necessary for both procedures. To be able to transmit uplink, the equipment has to establish an RRC session with the gNB by first sending the Random Access Channel (RACH) preamble.

In RRC_CONNECTED, both UE and gNB have the context necessary to communicate, meaning the UE may send data uplink and receive data downlink. The mobility in this state is controlled by the network, and while the UE is still responsible for taking measurements, these are reported back to the gNB, so it can decide if it is necessary to perform handover.

RRC_INACTIVE is the only one of these states that has not been inherited from LTE. This state has been created to cater to devices that would periodically send small amounts of data, switching to the IDLE state as means to save power, causing both the UE and gNB to lose the RRC context. This means that this session would then have to be established again each time the equipment needed to communicate, requiring extra signaling and energy expense. To solve this, RRC_INACTIVE is introduced as an intermediate state between RRC_IDLE and RRC_CONNECTED, where the UE is responsible for its mobility and is not able to communicate uplink, but both it and the gNB keep the RRC context established, thus allowing it to quickly switch back to RRC_CONNECTED without the need to signal the CN.

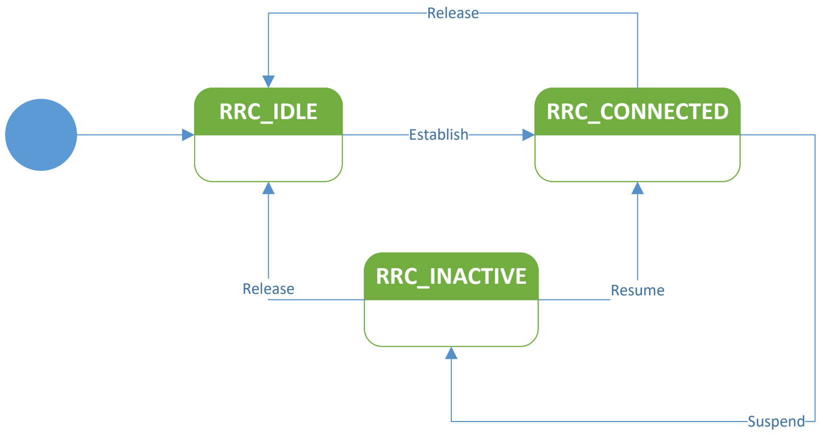


Figure 2.6: UE RRC state machine (Adapted from: [30]).

Cell Selection and Handover

Cell Selection is the mechanism that determines which cell an idle UE will camp on when joining a mobile network [31]. 3GPP defines in [32] four cell categories: suitable, acceptable, barred, and reserved. A cell is deemed suitable when it is part of the UE current Public Land Mobile Network (PLMN), and it meets the cell selection criteria. A cell that offers limited services, such as initiating emergency calls, is considered an acceptable cell. To restrict access, an operator may set the cell status to barred in the Master Information Block (MIB) or System Information Block (SIB)1 to disallow UE from camping on it. In case the operator wants to set some reservations to control accesses, it may do so by setting the cell status to reserved.

To find a suitable cell to camp on, the device must base its decision on its measurements and on the cell selection criteria [32]. To achieve that, a UE needs to scan all Radio Frequency (RF) channels across NR bands, selecting the strongest performing cell.

The cell selection criterion is defined by 3GPP as:

$$Srxlev > 0 \wedge Squal > 0 \tag{2.1}$$

$$Srxlev = Q_{rxlevmeas} - (Q_{rxlevmin} + Q_{rxlevminoffset}) - P_{compensation} - Qoffset_{temp} \tag{2.2}$$

$$Squal = Q_{qualmeas} - (Q_{qualmin} + Q_{qualminoffset}) - Qoffset_{temp} \tag{2.3}$$

where:

$Srxlev$ = Cell selection Reception (RX) level value;

$Squal$ = Cell selection quality value;

$Q_{rxlevmeas}$ = Cell RX value Reference Signal Received Power (RSRP);

$Q_{rxlevmin}$ = Minimum required RX value in the cell;

$Q_{rxlevminoffset}$ = Offset to $Q_{rxlevmin}$, only considered when searching for a higher priority PLMN when roaming;

$Q_{qualmeas}$ = Cell quality level Reference Signal Received Quality (RSRQ);

$Q_{qualmin}$ = Minimum required cell quality value;

$Q_{qualminoffset}$ = Offset to $Q_{qualmin}$, only considered when searching for a higher priority PLMN when roaming;

$Qoffset_{temp}$ = Offset temporarily applied to a cell, as described in [30];

$P_{compensation}$ = The maximum value between the uplink transmission capacity of a UE and the RF output power according to its power class.

While some of these values are sent to the UE via SIB1, the others are measured by the UE, in particular $Q_{rxlevmeas}$ and $Q_{qualmeas}$, that represent the reference signals RSRP and RSRQ respectively. RSRP is the measured power of 5G signals across broadband and narrowband, while RSRQ represents the signal-to-noise ratio [33].

Handover is the process of transferring an ongoing UE connection from the current cell to a more suitable target cell. This process can be triggered for reasons ranging from load balancing to device movement [34]. While connected, the UE sends periodic MeasurementReport messages to the gNB, with the measured reference signals, so that it may make a decision using these values.

Handovers may be, not exclusively, intra-gNB, that is between two gNBs and inter-RAT, between two different RATs. In 5G, due to the disagregated CU and DU, new handover types were defined. They are the intra gNB-DU handover, the inter gNB-DU and gNB-CU handover, and inter gNB-CU handover.

While the use of signal strength indicators has been the standard on traditional mobile networks, 5G requirements introduced new scenarios where this method may not be the most suitable. One of those scenarios is, for example, the 5G Ultra Dense Networks (UDN), designed to meet 5Gs low latency and data traffic requirements, where the huge number of deployed cells may cause load balancing and interference problems [35]. It may also happen that the cell with the strongest signal indicator may not be reliable at all.

To tackle these problems, researchers have been suggesting alternative algorithms or methods for cell selection/handover. In [36], the researchers propose a SDN-based approach that uses UE mobility data and cell load in addition to the signal strength. This method aims to satisfy the 5G latency requirement of less than 1ms. It uses a Linear Programming (LP) strategy to effectively optimize the process of choosing the next cell. To further reduce handover time, this method suggests the pre-allocation of a channel for the UE. The insertion of the SDN controller also decreases the number of direct messages between UE and gNB therefore making the handover process faster.

The authors of [37] present a cell selection scheme capable of adaptation in regard to the characteristics and needs of the UE, in particular high-speed vehicles. For handover, this scheme considers the current speed and direction of the terminal. The algorithm is composed of six phases: Configuration, Decision, Filtering, Narrowing, Selection, and finally Handover (HO) Trigger. The results show that the scheme is capable of reducing the number of handovers and increasing higher downlink data rates, especially when compared to the traditional Handover algorithm.

The algorithm proposed in [38] uses a fuzzy logic scheme to optimize cell selection based on the needs of the application. In [39], the authors designed a handover algorithm that uses a jump Markov linear system and deep reinforcement learning platform. This platform predicts the deterioration pattern on the target link to avoid any unnecessary handovers.

Soon it will become apparent that the 3GPP defined cell selection algorithm falls short of NTN requirements. Some of the algorithms described before use speed as a discriminating variable. However, since satellites move at much higher speeds than the typical UE, this characteristic is not very helpful in NTN scenarios. Even so, these algorithms give precious insight on how to better suit these processes to the application or, in the case of satellites, to the platform where the RAN is mounted in.

2.2 5G Open Source Software

Software is considered Open Source when the source code is available for anyone to inspect, modify or contribute. This code may be under a license that restricts its usage.

Open Source implementations of CN and RAN functions allow for fast, costless deployments of 5G infrastructures. These may be used in research, for example, to test new equipment in a controlled environment. If these projects could reach a stable state, they could theoretically be used by network operators in real-world implementations.

This section consists of a summary of the existing 5G open source projects, with a review of their current state.

2.2.1 Open Air Interface (OAI)

OAI is a collection of open-source projects conducted by the OpenAirInterface Software Alliance, that include a 3GPP-compliant implementation of a 5G CN [40] and RAN. These implementations are developed for Linux systems using the C language and publicly available under the OAI Public License. Using only OAI software, it is possible to have a functional end-to-end network, and for this reason, it has been widely used by researchers to set up virtualized 5G testbeds [41] [42]. At the time of writing, the current stable version of OAI RAN only features LTE functions, but an implementation of the 5G gNB and a 5G UE are made available on the development branch, that is updated on a weekly basis [43].

Currently, the OAI gNB implementation is able to emulate a full 5G RAN protocol stack, including layer 1 (PHY), layer 2 (MAC, RLC and PDCP) and layer 3 (RRC). It allows for both SA and NSA deployments and supports the CU/DU split detailed in Section 2.1. The modem supports the use of Universal Software Radio Peripherals (USRPs) as radio heads, which makes it possible to test the network using Commercial Off-The-Shelf (COTS) devices. It is also possible to replace the radio heads with an RF simulator, allowing the OAI UE to connect to a OAI gNB without the hassle of over the air perturbations. It is important to note that this simulator currently only supports one UE connection.

OAI software can be deployed directly on the Operating System (OS) or using Docker containers. There is a Kubernetes-based platform, Kube5G, that can automatically scale the network according to user traffic.

There have been some efforts to prepare OAI software for NTN. Fraunhofer IIS is currently working to implement Phase Tracking Reference Signal (PTRS) and additional Demodulation Reference Signal (DM-RS) positions both for uplink and downlink and improve time synchronization and frequency error estimation [44]. Some of these changes are already integrated on the *develop* branch. The 5G-SpaceLab has published in the *develop_SnT* branch a version of the OAI-RAN capable of dealing with a fixed delay and doppler support. This version may later support dynamic delay and doppler [45]. The authors of [46] released on the *SnT-enb-ntn-rel-1.0* branch, a LTE stack with PHY and MAC adaptations to account for NTN delays and an implementation of a NTN channel simulator.

To achieve a good performance, OAI software, in particular, the 5G modems, require a considerable amount of processing power and memory. In order to make these requirements less stringent, there have been efforts to use hardware accelerators to offload some of the most demanding processes. One such case is the Low-Density Parity-Check Code (LDPC) decoder which can be run on a Field-Programmable Gate Array (FPGA) [47].

OAI boasts of a current performance of 80Mbps downlink throughput with 64-Quadrature Amplitude Modulation (QAM) and a 7Mbps uplink throughput using Quadrature Phase-Shift Keying (QPSK) on NSA deployment, when up to 2 users are using the network. In SA deployments, OAI software achieves a maximum of 80Mbps downlink throughput, and 7Mbps uplink throughput, with a latency mean of 9ms. Both RAN deployments were tested using COTS UEs, and diverse core networks, both open and closed source [48].

Another project from the OpenAirInterface Software Alliance is the FlexRIC, an O-RAN compatible SDK to ease the development of RAT-agnostic and vendor-independent Software Defined Radio Access Network (SD-RAN) controllers for specific use-cases [49]. Contrary to the O-RAN reference architecture, this implementation did not use a microservice-based architecture, dependent on Docker and Kubernetes. As a result, it has less memory and Central Processing Unit (CPU) usage than the reference implementation [49]. It does not implement a full non-Real Time (RT) controller.

2.2.2 Open Networking Foundation (ONF) SD-RAN

The SD-RAN project aims to develop RAN solutions that are both 3GPP-compliant and compatible with the O-RAN architecture. The current release includes a nRT-RIC based on ONOS that is able to communicate with the RAN hardware via the E2AP interface, as defined in the O-RAN architecture. The RAN modules, mainly the CU, DU and RU, on this release are based on OAI software, modified to be more consistent with the O-RAN architecture [50]. These changes include support for the E2AP interface and support for RAN slicing [50]. The RIC features a cloud-native microservice-based architecture implemented using Docker and Kubernetes. It can be used with RAN hardware from different vendors. However, at the time of writing, the software network solutions provided by SD-RAN only support LTE.

Deutsche Telekom used ONF SD-RAN in a outdoor test trial [51].

2.2.3 Other implementations

Alternative implementations of the 5G core include free5gc and open5gs.

Free5gc is an open-source implementation of the 5G-CN, hosted by the National Chiao Tung University. The most recent release implements most functions and interfaces necessary for a functional 5G core, including the AMF, the SMF, the NRF, the Network Slice Selection Function (NSSF), the UDM, the UDR and the AUSF. It features a 5G core orchestrator and supports Internet Protocol Television (IPTV) services and non-3GPP access via Non-3GPP Interworking Function (N3IWF) [52]. Open5GS is an open-source implementation of both the 5G core and the EPC, supporting both 4G and 5G non-standalone and standalone deployments. It has been tested with multiple commercial gNB, and it can be used with multiple RAN software, such as OAI and UERANSIM [53].

There are other open-source RAN projects, such as Free5GRAN and UERANSIM. Free5GRAN is an open-source implementation of the 5G RAN stack. The most recent version can decode MIB and SIB1 data. It can also be used as a cell scanner. It only supports Standalone mode, and its PHY layer is incomplete. It has no support for other stack layers. At the time of writing, the project seems to be inactive [54]. UERANSIM is an implementation of both 5G UE and RAN. The software can be used to simulate both nodes in order to test a 5G core network, in standalone mode. It implements both the Non-Access Stratum (NAS) and the NG Application Protocol (NGAP) interfaces of the Control Plane and the GPRS Tunneling Protocol (GTP) in the User Plane. The 5G-NR interface is simulated over User Datagram Protocol (UDP), which means that the PHY, MAC, RLC and PDCP layers are not implemented [55].

2.3 Satellite Communications

A satellite is a flying device that, with the use of a transponder, can transmit a signal from one spot on Earth to another [56]. In satellite systems, one or more of these objects typically make up the space segment, being responsible for the exchange of information between the various terminals and gateways that comprise the ground segment [57].

Satellite systems today provide a vast range of services, that generally fall in one of these categories: Observation, Navigation, and Communication. Observation satellites typically gather data that can be used to evaluate the current status of planet Earth. With their privileged vantage point, these satellites may be used to collect meteorological and geographical data that can be used in order to help identify potentially hazardous situations in the near

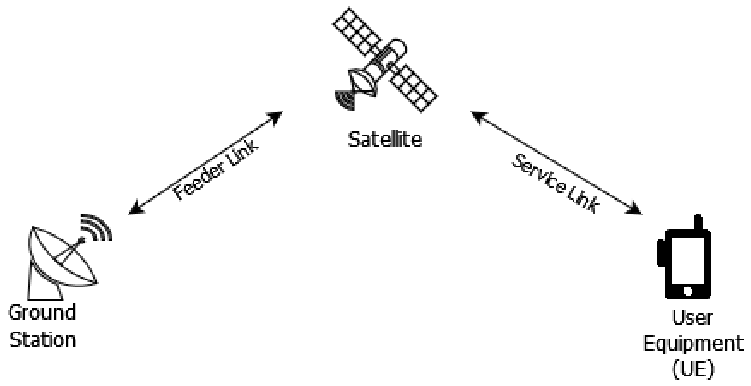


Figure 2.7: Typical end-to-end satellite communications system (Adapted from: [58]).

future, such as natural catastrophes, as is the case in projects like EUMETSAT [59]. Due to generalized usage of Global Navigation Satellite Systems (GNSSs) like GPS, navigation satellites have become ubiquitous in everyday life, as they allow for real-time, fast localization. The service provided by these satellites enables diverse applications such as sports monitoring or simple navigation.

Communication satellites are used to propagate communication links. Service providers like Iridium [60], and Inmarsat [61] provide data, voice, and text services uninterrupted across the globe, with solutions to diverse markets. As mentioned in Chapter 1, these satellites can be used to reach underserved areas but also to connect ships and planes. The download and upload speed of these networks still has not reached fixed broadband, but it is getting there. In a study by Ookla [62], the average download speed for the Starlink constellation was around 97.23 Mbps, while the upload speed clocked by 13.89 Mbps, with a latency of 45 ms. Other providers like HughesNet and Viasat performed at about 19 Mbps with an upload speed of around 3 Mbps, with much higher latencies. Eutelsat offers plans that range from 30 Mbps to 100 Mbps download speed and Vivanet from 16 Mbps to 50 Mbps download [63], but no data was found on their average performance.

Due to their easiness of setup, global coverage, and resistance, communication satellites have been widely used in disaster relief and emergency situations, where the communication infrastructure has been damaged or in some cases, non-existent. The rapid response becomes necessary in these situations, as failure to communicate may lead to loss of life. Télécoms Sans Frontières (TSF) is an non-governmental organization whose main activity is the setup of satellite communications for phone and internet access in areas affected by natural disasters or other crises [64]. The solutions to achieve the desired connectivity include Oxbird, a service that uses traffic shaping techniques to guarantee optimal bandwidth use and QoS during emergencies, and the use of fixed and portable satellite terminals to provide permanent or temporary access to communication [65]. In the 2021 Southwest Haiti earthquake, TSF efforts brought connection to a group of humanitarian efforts with the use of satellite broadband, which allowed, for example, doctors on-field temporary camps to contact the operation center to communicate the needs of their patients [66].

During the Russian invasion of Ukraine, seeing the communication infrastructure being attacked, the vice prime minister requested help from Starlink in order to guarantee access to the internet to the citizens and military. Starlink activated the network and sent some user terminals to the country [67].

One thing in common with these examples is the necessity of setting up the infrastructure to be able to communicate with the satellites. If these somehow were integrated on existing mobile networks, the relief efforts could potentially save some time which in these situations is critical. The effort to integrate these technologies in 5G networks is detailed in Section 2.4. The rest of this section is dedicated to essential satellite concepts.

2.3.1 Orbit Configurations

When choosing a satellite system, it is important to be aware of the different orbit configurations and their drawbacks, in order to select the most suitable for the use case in question. Depending on the altitude, communication satellites orbits tipically fall in one of three categories:

- Low Earth Orbit (LEO): from 500 km to 2000 km;
- Medium Earth Orbit (MEO): from 2000 km to 35786 km;
- Geostationary Orbit (GEO): around 35786km.

The main advantage of LEO satellites is the lower distance the signal has to travel, resulting in lower propagation delays and, therefore, lower latencies. However, being at a lesser distance from Earth also means it has less area coverage. So, in order to maximize coverage and assure an arbitrary terminal has continuous service, it is necessary to deploy a larger number of satellites. Due to the high orbital speed of LEO satellites, frequent handovers are necessary to maintain a stable connection. In opposition, GEO satellites rotate at the same rate as Earth, meaning that they appear to stand still at a point, meaning that a single satellite can maintain continuous coverage of a given area, meaning the antenna can be stationary. Antennas connected to LEO and MEO need to be able to track the satellites with, for example, the use of phased arrays. As a result of the higher altitude, GEO satellites can cover a wider area, but the signal suffers a longer propagation delay.

	GEO	MEO	LEO
Latency	High	Low	Very Low
Coverage	Very Large	Large	Small
Satellites Required	Three	Six	Hundreds
Antenna Speed	Stationary	1-hour slow tracking	10-minute fast tracking

Table 2.1: Orbit configuration comparison (Adapted from: [68]).

2.3.2 Payloads

The satellite payload is the group of devices (antennas, receivers, and transmitters) that make it possible for the device to fulfill its purpose. These payloads typically fall into two categories:

- Bent-pipe: Bent-pipe payloads, also known as reflective, receive a signal, filter it, frequency-converts, switches, amplifies, and transmits it back to Earth.
- Regenerative: Regenerative payloads differentiate themselves by having a Onboard Processor (OBP), allowing the implementation of more complex functions.

2.3.3 Challenges

The use of satellites for communication is not without its problems. For example, the distance between the terminals and the satellites makes larger latency values unavoidable. However, while the signal propagation cannot be sped up, there are other techniques that can reduce latency in satellite links, like Transmission Control Protocol (TCP) acceleration [69]. The use of LEO constellations also results in lower propagation delays, as stated previously. Doppler Effect is a prominent problem in LEO constellations, caused by the speed difference between base stations and the satellites they connect to. This effect, if not accounted for, will lead to failure and packet loss in satellite links [70]. Nonetheless, there are Doppler compensation techniques that can be used to reduce the impact on the communication link [71]. When integrating satellites in the existing mobile infrastructure, these problems will have an impact and therefore need to be considered, as will be discussed in the next section.

2.4 5G Non-Terrestrial Networks (NTNs)

NTN is an umbrella term used by 3GPP to refer to networks that have RF resources mounted on a satellite or other Unmanned Aircraft System (UAS). In this dissertation, the focus will be specifically on satellites. As mentioned in Chapter 1, 3GPP has been working to expand the 5G ecosystem with the use of NTN, and specification is scheduled to start on Release 17 [72]. This integration was motivated by the fact that satellites can empower diverse use cases in eMBB and mMTC scenarios. These use cases include connecting low-density areas and providing service to moving platforms. They will also assist on mission-critical services, for example, by allowing emergency broadcasts to reach people, not in range of a cell tower. This section covers fundamental concepts of 5G NTN and the current state of research.

2.4.1 Architectures

3GPP defined in [11] a set of architectures for NTN systems. These architectures are similar to the one shown in Figure 2.7, and they can be categorized based on their payload, which can be either bent-pipe or regenerative. Each of these has a different impact on existing NR infrastructure, that not only depend on the new spaceborne elements themselves but also how these communicate with existing NR elements [73].

As mentioned before, bent-pipe payloads receive data and send it back down to Earth. With the bent-pipe payload architecture (Fig. 2.8), the satellite is then only responsible for maintaining the NR-Uu interface between the UE and the gNB, both on the service link and on the feeder link. No processing is done on the satellite since it is mainly acting as a repeater. This means that the Control Plane and User Plane protocols are terminated on the ground.

The use of this architecture does not presuppose any changes to the existing 5G RAN infrastructure, except for the timers of the NR-Uu interface, which have to be adjusted to allow for long delays caused by the satellite links. Consequentially, this architecture is less complex to implement, and therefore can be deployed quickly to the market.

Regenerative architectures (Fig. 2.9 and 2.10) have the satellite act as a gNB. This change significantly reduces delays as the NR-Uu interface terminates on the service link. The payload should implement Inter-Satellite Links (ISLs), that are established between satellites, permitting processes like handover over Xn interface. On the feeder side, the link can be implemented with any Satellite Radio Interface (SRI) if it can assure all signalling operations, meaning state-of-the-art Satellite Communications (SatCom) air interfaces can be used [73].

In the gNB based payload architecture, the SRI transports the NG protocol while in gNB-split based architecture, it transports the F1 interface, used for communication between CU and DU. Latency is still a problem, and both Control Plane and User Plane processes may be affected. This is more pronounced in the split-based architecture since protocols like RRC are terminated on the CU on the ground. The F1 interface will also be impacted by the long delays.

NTN architectures keep the same interfaces as the terrestrial networks. This is so that the non terrestrial devices may coexist with terrestrial ones. Moreover, it will allow that the same UE may connect to both terrestrial and non-terrestrial networks, as long it has the power to do so.

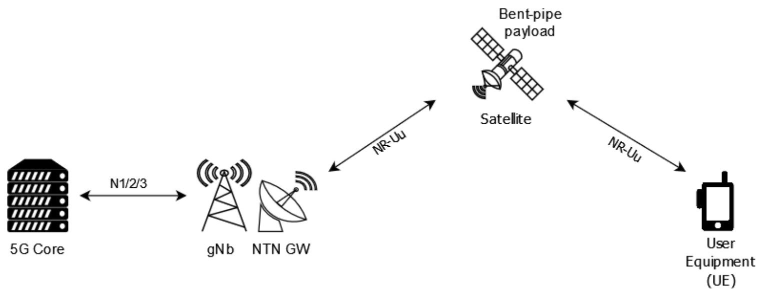


Figure 2.8: NTN RAN architecture with bent-pipe payload (Adapted from: [11]).

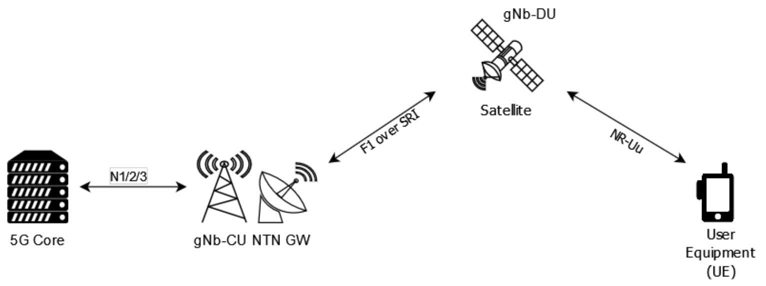


Figure 2.9: NTN RAN architecture with a gNB-DU based regenerative payload (Adapted from: [11]).

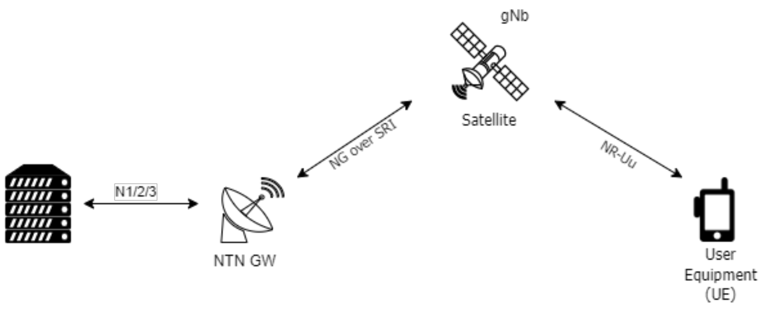


Figure 2.10: NTN RAN architecture with gNB based regenerative payload (Adapted from: [11]).

2.4.2 NTN Challenges

The ongoing integration of satellites in 5G networks has brought to light a couple of challenges and issues that need to be tackled in order to ensure that this integration is seamless [74]. The focus here will be specifically on challenges related to mobility and access. These also should be passible of being solved with software. One of such challenges is the possibility of frequent handovers. Due to the rapid motion of LEO satellites, a node may only be visible to a single UE for a short time. When a satellite gets out of view, a handover will undoubtedly be triggered. Since this process happens when the UE is RRC_CONNECTED, this process is time-critical. Failure to execute a handover will result in loss of data, which can be aggravated by the frequency of these events, resulting in a cascading failure [12]. This issue is not found in the case of GEO satellites. Another problem happens when both terrestrial and non-terrestrial cells overlap. In fixed, terrestrial cells, the RSRP value notably decreases as the UE starts gaining distance from it. On non-terrestrial cells, this is not the case. The RSRP may not variate too much while the UE is in the beam of the satellite, making this value less useful on the decision process to select the best cell. This problem may lead the UE to go back and forth between cells and may also adulterate the process of cell selection for idle devices.

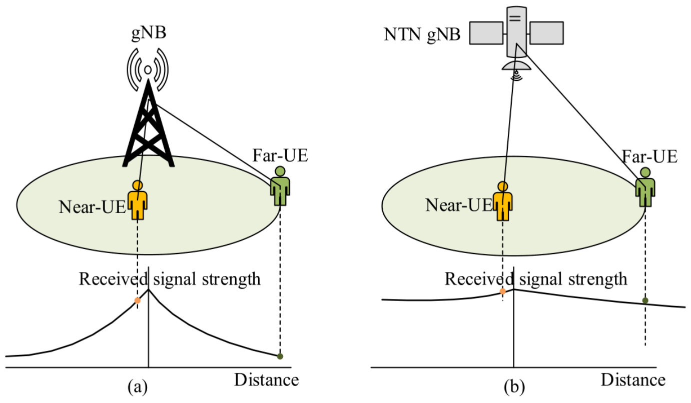


Figure 2.11: Near-Far Effect: (a) Terrestrial Cell (b) Non-Terrestrial Cell (Source: [11]).

A third problem is caused by the long latencies associated with satellites. These delays may cause situations where the network or the UE can receive outdated information. If that information is used for decision-making, it could cause unwanted behavior [11].

2.4.3 Current Research

Despite the relative recency of 5G NTNs, there already is a considerable body of work striving to make its realization possible.

In [71], the authors give an overview of NTN opportunities and challenges and propose a mobility scheme to tackle frequent handover issues. This scheme suggests that the UE starts looking for neighbor satellites if the signal level of the cell it is currently camped on falls below a given threshold. The information necessary for the decision would be broadcasted by the neighbor cells. The satellites should adjust their beams to a fixed spot on Earth in order to maintain the same Tracking Area Identity. However, this scheme does not consider problems like cell overlapping or measurement validation. It assumes that the UE only connects to satellite if terrestrial coverage does not exist, failing to consider other situations where switching to non-terrestrial cells may be beneficial.

The Open5GENESIS demo in [75] stands as an example of how 5G NTN might serve rural areas. Their network architecture uses a satellite as backhaul, connecting the gNB to the CN. To deal with the latency, they deploy some data plane functions on the satellite edge. However, this setup still requires a terrestrial gNB to be deployed at the site. In case that the gNB is damaged (e.g. natural disasters), the application will lose connection to the CN.

A SDN approach to the integration of non-terrestrial segments in 5G networks is discussed on [76], in the context of the Sat5G project. This paper proposes an architecture based on MANO and 3GPP standards for the end-to-end management and orchestration of these heterogenous networks. It recognizes that the implementation of such architecture may be challenging due to the different requirements of each segment. Similarly, [77] also explores how SDN concepts are necessary to fulfill non-terrestrial integration and how an end-to-end model makes it possible to program the network consistently, indifferent to the different segments. The researchers in [78] put these concepts into practice in a 5G testbed that integrates satellites and is fully orchestrated using OpenMano.

In conclusion, the purpose of this review was to give an overview of the current status of the 5G ecosystem, with a focus on NTN and mobility. A discussion on the current open source solutions was also provided.

As mentioned, the research on cell selection schemes seems to be focused on high density environments and vehicular use cases. With the arrival of NTN, more work should be conducted on how to adapt these processes to these new scenarios. It would also be beneficial if the current open source software packages would allow to simulate satellite channel conditions, to enable developers and researchers to start testing their 5G NTN projects on the coming years.

These ideas will motivate the rest of this dissertation, as will be made evident in the following chapters.

CHAPTER 3

Methodology and Design

The purpose of this chapter is to present the methodology followed during this work and to elaborate on the architecture of the proposed solution found to the problem described in Chapter 1.

The first section restates the motivation of the work and explains how the work to develop and test the solution was organized.

Afterward, there is a transition to the specification and architectural details of the solutions, where the requirements for each part of the work are defined for later implementation.

3.1 Methodology

As mentioned in section 1.2, the main objective of this dissertation is to create a firmware that may be used by 5G UE to assist with the choice of whether to connect to a satellite or terrestrial gNB. This firmware will implement a cell selection scheme that, using network performance metrics collected by the device, may decide which RAT is better at a given moment. It also strives to make this switch as fast as possible, so that it does not affect the performance of the modem critically. As Chapter 2 indicates, there is still a lack of literature on how these processes can be adapted to NTN, which is the focus of this research.

In order to evaluate the efficacy of the firmware, it was necessary to gather data on how the UE behaves running it, in different network conditions and scenarios. To have a base to compare the collected values, information on how the UE behaves when not using the developed firmware solution will also be required.

Enabling these tests required the deployment of a 5G network. Designing and finding a solution that would satisfy the needs of this proof of concept proved to be more difficult than initially expected. Thus the development of this testbench ended up becoming a significant part of this dissertation.

To this end, the work on this dissertation was split in four main phases, as evidenced in Fig. 3.1.

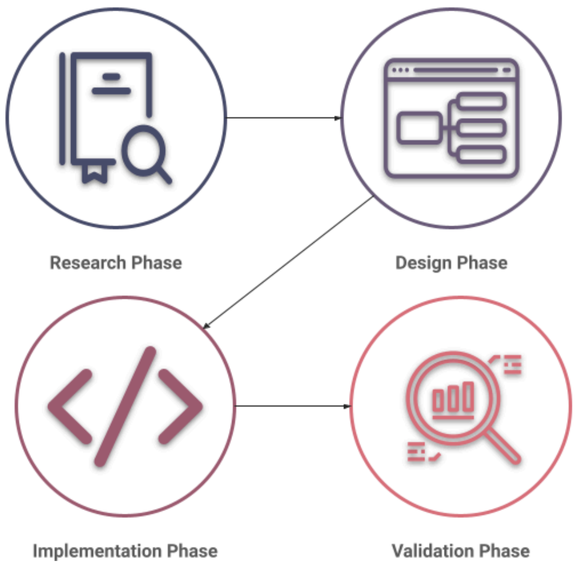


Figure 3.1: Methodology Diagram.

The first phase was to research 5G and NTN topics, to gain knowledge about the current landscape. The end product of the phase was the Literature Review, Chapter 2. The second phase was Design, where the knowledge gained in the previous phase is put to use to elicit the requirements of a possible solution. This was followed by an Implementation phase. The goal of this phase was to produce a working proof-of-concept solution based on the design done on the previous phase. Lastly, a Validation phase, where the software developed was tested. The tests provided quantitative data that allowed the appraisal of the solution.

The rest of this chapter will focus only on the conceptualization and architecture of the solution. This may contain elements that were considered but may have not made it to the proof of concept developed. For these and other implementation details, refer to the next chapter.

3.2 Specification and Architecture

When working toward a solution, writing its specification is an important step to ensure that the implementation will meet the requirements.

This section starts with a description of the scenario considered during the design of the solution, that illustrates the goals and constraints of the project. This is followed by the definition and modeling of each part of the solution developed, starting with the firmware and ending with the network degradation module.

The project was given the codename eXtend5G. This term will be used in reference to the designed solutions.

3.2.1 Scenario

The 5G ecosystem includes a wide array of use cases, across multiple sectors. Each use case may include multiple device types and take advantage of different functionalities of the new generation mobile networks. Additionally, as hinted in Chapter 2, integrating non-terrestrial RATs in these networks has introduced many new challenges of different complexities.

It would be difficult to come up with a one size fits all solution or to tackle the various complications of NTNs. Hence, for a solution to be attainable, the scope of the work had to be reduced. Furthermore, it is important to define what requirements the solution being developed should fulfill. The result of that process is the scenario that follows. While this is a hypothetical scenario, it is based on real-world cases of IoT commercial solutions. It aims to contextualize the work being done, hint at architectural elements and highlight the scope constraints.

The UE may be thought of as a generic, nondescript IoT device, that needs to report and access data in a remote server. This device does not have particularly strict low latency requirements, unlike an autonomous vehicle for example, but failure to respond or to fetch data in time may have consequences service-wise. Roaming is not an issue, as this machine will not cross geographical borders.

To communicate with the remote server, the device makes use of a 5G network. It will have to establish a data session with the network so that it can receive and send data packets. It will not make voice calls or send Short Messaging Service (SMS) messages.

Since this network serves an area typically affected by seismic activity that causes partial or total failure of the terrestrial base stations, it implements a non-terrestrial RAT, in the form of a satellite, that devices can attach to in order to keep connected to the network when the terrestrial gNB underperforms or become unavailable.

However, the UE still uses traditional Received Signal Strength Indicator (RSSI) cell selection algorithms and therefore is not able to take proper advantage of this network deployment. A firmware, runnable on existing hardware, that would allow this device to maximize its network usage in disaster situations would then become desirable.

3.2.2 eXtend5G Firmware

As mentioned before, the main objective of this firmware is to seamlessly switch to the best option between terrestrial or non-terrestrial gNB, at a given moment. Which of them is the best option is determined by analyzing a few network metrics collected by the device.

To achieve this objective, it was necessary to define exactly how this firmware should behave, what metrics factor in the choice to switch and what hardware it should be run to as well as define some performance requirements.

Behavior

Initially, the behavior of the firmware was modeled with a state machine, a common design pattern in embedded software, where the behavior of the software changes in tandem with its internal state. The high-level architecture of this state machine can be found in Fig. 3.2.

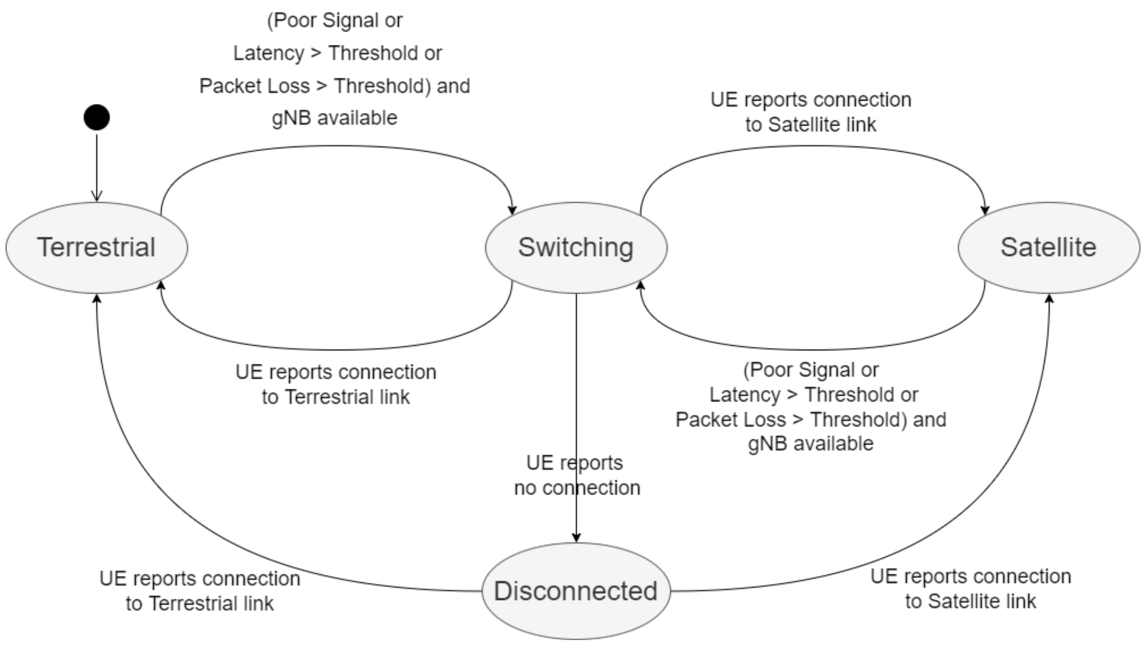


Figure 3.2: eXtend5G Firmware High-Level State Machine.

The designed state machine includes five states:

- **Init State**: The starting point, where the firmware collects information related to the initial state of the 5G module. The main decision taken is to choose either to transit to the Satellite, Terrestrial or Disconnected states, based on the current 5G module connection.
- **Terrestrial State**: While the modem is maintaining a terrestrial link, the firmware will be collecting data and analyzing it. In case the network conditions justify it, the firmware signals that a switch is necessary, transiting to the Switching State. Otherwise, it stays in the same state.
- **Satellite State**: Similar to the terrestrial state, but used when the modem is using a satellite link.
- **Switching State**: The firmware starts the switching process from terrestrial to satellite or vice versa by signaling the modem. It should stay in this state while the switching is not complete. It periodically monitors the modem status, transitioning to either Terrestrial, Satellite, or Disconnected based on the responses given.
- **Disconnected State**: When the modem is not able to establish a link to a cell of any type, the firmware periodically searches for new cells until it connects to the first available one. Afterward, the next state is decided according to the type of cell the modem managed to connect to.

Metrics

Having nailed down the behavior of the firmware, the following step was to define what metrics the software would gather would base its decisions on. It is important that these metrics accurately reflect the current state of the connection established by the UE. They should be enough to determine if the network is under stress or beginning to fail so that the switch is made before it further deteriorates.

One of such metrics that the firmware will gather are the RSSI measurements, for example, the RSRP and RSRQ, already commonly used in cell selection algorithms. Despite the limitations of these measurements on satellite scenarios, as exposed in Chapter 2, they are still the prime way to measure the signal strength and quality in relation to a cell tower, and should not be disregarded. The other metrics shall be chosen to compensate where these indicators falter. These metrics are usually calculated by the UE and are therefore easily available.

Latency is another metric that will be considered. Latency is the amount of time it takes for a network request to go from source to destination. If the time for the answer to the request factors in, it is called the round-trip time. Long latencies in networks can be caused by numerous factors, for instance, distance, as is the case for satellites. These metrics are also affected by the application being used, if it takes a long time to processing requests. That is to say that the cause of the network latency may be hard to pin down. Nevertheless, it is still a useful metric for the algorithm, given that it is carefully calculated to only include delays within the 5G network. This metric should be configurable, so that devices with low latency requirements can stick to the lower bound for latency, switching as soon as the packet latency is above the usual satellite performance, while others may stick to the higher bound, only performing the switch if it crosses well above usual network expectations. To gauge the satellite performance, the UE will have to be made aware of the propagation delay of the satellite link somehow, either informed by the network or by user input.

Packet loss, one of the other metrics considered, is the percentage of network requests lost during a communication between two hosts. It can cause certain data to be transferred incompletely. Packet loss is a constant in satellite networks, for example, Starlink reports a loss of up to 1.96% when uploading HTTP/3 data [79]. If packet loss crosses a certain threshold while connected to a terrestrial cell, it may be worth performing a switch. This threshold should be configurable, as some applications like VoIP or streaming are more sensitive to this loss compared to others.

Both latency and packet loss give the algorithm an awareness not only of the radio fronthaul but also of the network backhaul. These metrics can be calculated using network performance measure tools like *ping* or *nperf*.

Finally, the last metric to be considered was the location of the satellites. As will be seen in the next chapter, this was not implemented. The location of the satellite would work as a complement to the other metrics analyzed. Having this information, the algorithm would be able to estimate when and for how long a satellite would be within the UE range, consequently being able to determine when would be the best time to start switching if necessary, or even if this switch would be possible at the time it was requested. However, to get this metric the UE would need to be able to estimate its position, with a Global Positioning System (GPS) for example, and it would need to receive information regarding the satellite positions.

Hardware and Performance

The foremost problem with the hardware choice is the level of control. It is clear that, to connect to a 5G network, the UE will have to possess a 5G module or modem. Ideally, it would be possible to extend or change the cell selection methods of the modem firmware, but since these tend to be proprietary and closed source this is not commonly possible. Consequently, the other option is to use the low-level interfaces these modules usually expose for applications, allowing them to control it to some degree.

Therefore, using a COTS UE like an Android smartphone is out of the question, as the Android Software Development Kit (SDK) does not expose those lower-level interfaces. So, to get a base station to develop the firmware to, it is needed to select both the 5G module and the device where the application that interfaces with this module, the firmware, will be executing.

The selection of the 5G module involved researching the available options on the market, with a preference for development kits, so that it came ready to start being used. Other factors that were considered were if it contained a GPS module and if it was designed for IoT usage.

As for the device, it should be a single-board computer running a Linux distribution. Single board computers are widely used in IoT projects. Having an operating system opens up the possibilities in language choice for the firmware, while still requiring it to be optimized for performance because of the usual low resources these computers have. Having a Linux distribution also makes it easier to interface with the modem, as suppliers usually provide drivers for these platforms. Using other platforms could result in having to develop drivers, resulting in additional work for this dissertation. It is also important to make sure the device chosen is able to power the 5G module chosen.

To develop the firmware, preferably a low-level, typed language like C or Rust should be chosen. The software developed should be reliable and fast, and these compiled languages can be optimized for speed. However, for this proof of concept, a higher-level language may become an option due to the ease of development and quick prototyping.

The firmware should use as few resources as possible to maximize the resources available to the other applications running on the device.

Running this firmware externally to the 5G module also has the advantage that it can be easily containerized and deployed to different IoT devices.

What hardware was chosen and how this firmware was developed is answered in the next chapter.

3.2.3 eXtend5G Testbench

Testing and validating our firmware required an end-to-end 5G network, that included both a terrestrial and non-terrestrial gNB. In addition, there had to be a way to programmatically degrade these network conditions, to englobe all switch conditions of the firmware. The tools to deploy and degrade the network are known as the eXtend5G testbench. The next sections will highlight the choices of architecture and the specification of both the deployment and degradation tools.

Network Architecture and Deployment

The network architectures that will be considered are based on the architectures depicted in 2.4, specifically on the regenerative ones, that is, architectures where the gNB is deployed in part or fully on the satellite. In addition to the satellite gNB, a terrestrial gNB will also factor in the network architecture.

If possible, the plan was to deploy two different network architectures, one monolithic and one using a CU/DU split, as depicted in Fig. 3.3 and Fig. 3.4. With this, it would be possible to get results that go beyond the firmware, testing how the different network dispositions would be affected by latency and other satellite features. Only 5G standalone will be considered.

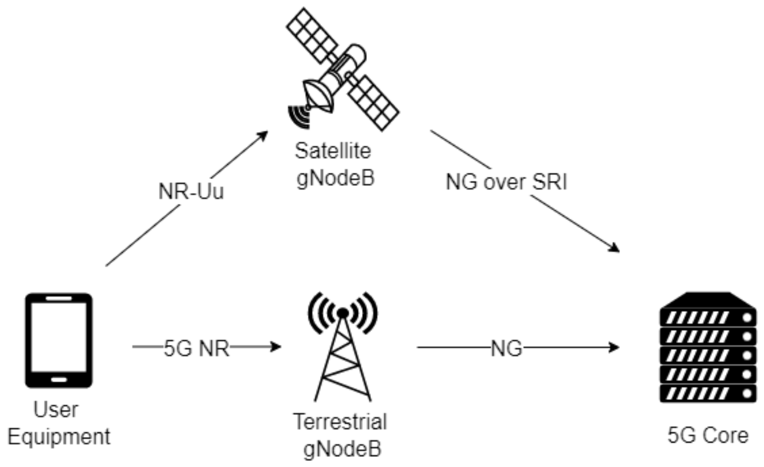


Figure 3.3: Suggested monolithic network architecture.

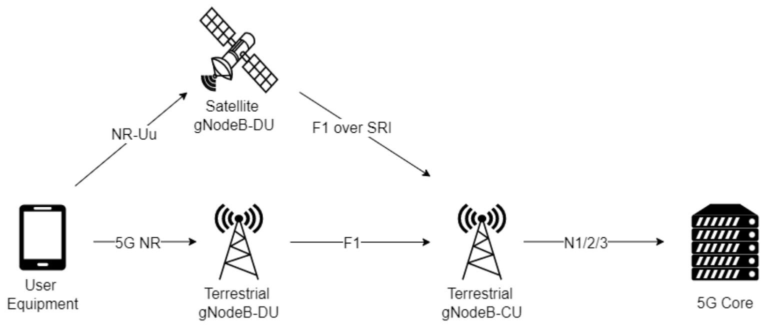


Figure 3.4: Suggested split network architecture.

Putting this plan into practice called for the identification of the hardware and software infrastructure that would effectively allow for the deployment of this network. The main ways to achieve this involve either using a fully open-source software deployment or using a mixture of both software and COTS hardware, as long as the choices fit the proof of concept requirements.

The proof of concept being developed had three main requirements for the network deployment:

1. Allow UE connect via RF;
2. At least one gNB has to have satellite-like features;
3. the satellite gNB needs to be distinguishable for the UE;

Both these requirements mainly limit the choice of software or hardware for the gNB, while still leaving the choice for the core network open-ended.

These choices required the study of most of the software listed in Section 2.2 and other solutions on the market.

Requirement (2) was found to be the most limiting and complex. As of the time of writing, there are no known instances of a satellite deployed with a full 5G gNB. Aside from the complexities at the hardware level to create such a solution, even the act of adapting a gNB to satellite conditions is not trivial [12].

For this project, the main satellite feature that was chosen to be implemented was the propagation delay caused by the bigger distances. As mentioned in Chapter 2, this delay, depending on the satellite orbit, can be several times higher than propagation delays to terrestrial cells. This introduction of this latency on the connections will require several changes to the 5G gNB, that include but are not limited to, disabling or extending Hybrid Automatic Repeat Request (HARQ) processes, creating a Random Access Response (RAR) window for NTN purposes for the Physical Random Access Channel (PRACH) procedure [12].

Making these changes and validating them was considered out of the scope of this dissertation. So, for this research, a compromise had to be made. The focus then was to find the maximum latency supported by the fronthaul without having to develop these modifications. Furthermore, a way to simulate this latency had to be found, either by physical means or software.

As discussed in section 3.2.2, the firmware will be running externally to the modem. This means it technically cannot choose which cell it connects to directly, since this process is done internally in the 5G module. Hence, as the Requirement (3) states, the satellite gNB needs to be deployed in a way that distinguishes itself from the terrestrial gNB so that it can be chosen deterministically using only the commands the 5G module exposes to the application.

How all this was achieved and implemented is discussed in the next chapter.

Network Launcher

The testbench network for this dissertation will be composed of various parts and hosts as evidenced by the network architecture in Fig. 3.3. Launching and configuring each machine manually every time a test had to be run would not be time efficient.

The network launcher, also known as the deployment tool or the manager, was designed as a solution to this problem. The purpose of this software is to automate and facilitate the 5G network deployment.

This program should allow the user to configure and deploy a network architecture based on a configuration file, including the core and the needed gNB, be their satellite or terrestrial, monolithic, or split. These network modules should be handled independently; if one of them fails to execute it should not affect the others. Moreover, each module should have its log, to facilitate debugging and troubleshooting.

This software should be able to set up the 5G network remotely. How this can be achieved, and what protocols will be used is intrinsically linked to the hardware and software choices made for the network infrastructure deployment.

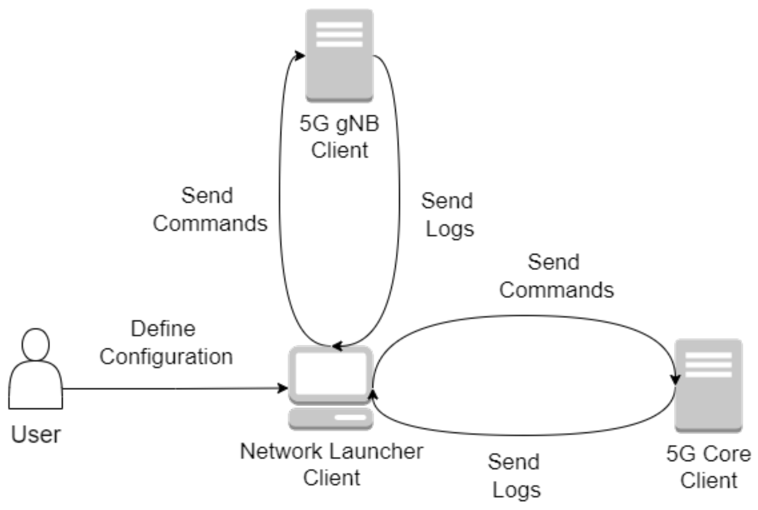


Figure 3.5: Network Launcher Tool Diagram.

Network Degradation

Initially, the dissertation was going to focus on UE behavior in catastrophic scenarios where network access would be completely cut off terrestrially. However, such cases would not evaluate the firmware developed properly, as the only access method available would be through satellite, therefore trivializing the decision-making. So, instead of just focusing on such scenarios, the idea of emulating more realistic, everyday network problems, essentially tampering with the QoS, became more pertinent, enabling the firmware can be tested under conditions similar to those it would be operating on a day-to-day basis.

The main metrics chosen to be tampered with were the latency and the packet loss, as these are the ones being monitored by the firmware developed. If this could be achieved both on the fronthaul and the backhaul of the network, it would be ideal. However, since the fronthaul will be mainly composed of RF links, this would require a controlled environment and the use of additional hardware. The backhaul connections are wired and may be susceptible to tampering using only software.

As with the network launcher, the extent that the network may be tempered with and how is linked to the hardware and software choices made for the network infrastructure.

This process resulted in another software tool, the Network Degradation Simulator, whose implementation is described next chapter.

The main requirement was to allow the user to automatically or manually tamper with each connection of the network infrastructure. To each link between two devices (for example, between gNB and the core network), the user may apply a scenario, a series of instructions that describe how the link will be degraded over time or just tamper with the metrics manually. This program will then communicate with each host and configure it based on the user inputs.

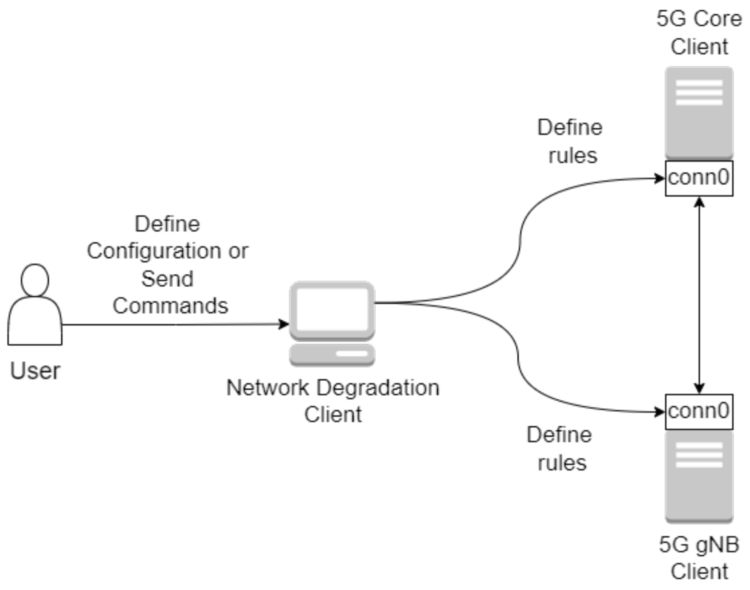


Figure 3.6: Network Disruption Tool Diagram.

During this chapter, the various modules that will make up our solution were defined. Having these in mind, it becomes possible to develop and test a proof-of-concept, satellite-ready UE.

CHAPTER 4

Implementation

This chapter is focused on the development of the two main modules of this dissertation, highlighting the choices and compromises made to achieve the proof of concept. It starts with a detailed explanation of the testbench, which includes the network per se and the network degradation simulator. To finish, there is a discussion of the implementation of the eXtend5G UE firmware.

The network launcher and the disruption tool will be the focus of section 4.1, with the firmware coming into the spotlight in section 4.2.

4.1 Testbench Implementation

The eXtend5G Testbench comprises all tools necessary to deploy the test network used during this project. This includes the scripts to launch the end-to-end network, the network degradation tools, the gNB configurations, etc. In this section, all these steps will be described.

4.1.1 Network Deployment

Getting the network up and running involved the choice of the RAN software, the evaluation of the hardware requirements and the set-up of each component.

After an assessment of the list of software detailed in section 2.2, the one that was picked was OAI. Compared to the alternatives, OAI is the one that stands out because it is currently the only one that implements all the elements necessary to set up an end-to-end 5G network, using actual radio heads, without the use of commercial gNBs or cores. As referenced before it is one of the most active projects and the one with more functionalities implemented, including the CU/DU split, factors that also weighed in on the choice.

There was an interest in deploying two different architectures, one with monolithic gNBs and another using split CU/DU. However, for the proof of concept only one architecture was deployed, the one using monolithic gNBs. This choice was made because this architecture was considered the most reliable and consistent to deploy with the software chosen. Split deployments, while initially considered, had to be discarded due to an issue with OAI, where COTS UEs were unable to establish a Protocol Data Unit (PDU) session using split gNB. This problem was reported to the development team.

A quick look at Fig. 3.3 makes clear that at least three machines are necessary for this deployment, one for the core and two for each gNB. Technically, it would be possible for one machine to both host a core and a gNB, but after reading about how the network disruption tool works later, it will be easy to deduce why this becomes undesirable in this case. Two USRPs B210 are also necessary to serve as the radio heads for the gNBs. The USRP B210 is a software-defined radio developed by Ettus Research thus it can be reprogrammed according to the necessities of the users. All devices are connected to the same wired Local Area Network (LAN), each configured with a route to the core network.

Since the OAI gNB has demanding requirements resource-wise, the specification of the hosts needed to have powerful enough hardware to run the software without encountering unexpected problems. On top of that, OAI uses integer Single Instruction, Multiple Data (SIMD) instruction sets like the Streaming SIMD Extensions (SSE) and the Advanced Vector Extensions (AVX2), which means it requires an x86/x64-based CPU to be executed. For these reasons, machines used to run the gNB sport a 12th Gen Intel(R) Core(TM) i9-12900K CPU, with 12 cores, running up to 5.20 GHz, with a total of 64 GB of Random Access Memory (RAM). As for the core, since its resource usage is observed to be less intensive, even when two gNBs are connected, it may run in a less powerful machine. During this experiment, the core was run in a machine with Intel(R) Core(TM) i5-4440 CPU, running at 3.10GHz, with a total of 16GB of memory. All the machines used were using Ubuntu 18.04 as the OS with the 4.15.0-192-lowlatency kernel, as per OAI recommendations.

On both gNB machines, the OAI version was frozen on the commit tagged 2022.w30, for stability reasons. Further modifications were made to this version of OAI due to the requirements of this dissertation, which will be described later on. The gNBs are compiled on the machine and run on bare-metal, as opposed to the core which is deployed using Docker. The core is deployed using a basic configuration that includes a UDR, a AUSF, a AMF, a SMF, a UDM, a NRF, a UPF and a MySQL server for subscriber data. The docker-compose file and script used for the deployment are available on the oai-cn5g-fed git repository. Only slight changes were made to these files, to add more subscribers to the database and to adapt them to the available network configuration.

As mentioned in Section 3.2.3, there was a need to differentiate the terrestrial from the non-terrestrial gNB and to simulate the regular conditions of a satellite link as well as possible, with the existing tools and hardware.

One of the most important parameters to simulate on the satellite link was the insertion of fronthaul latency, caused by the signal propagation delay. The approach chosen to achieve this was via software and involved making changes to the OAI USRP driver. Inserting a waiting interval after the radio samples were captured, but before they were sent for processing allowed them to achieve the desired fronthaul delay. The modifications done let the user configure the gNB RU to have a delay both on reception and transmission.

However, it needs to be pointed out that this approach has its limitations. Since the base OAI has not been adapted to support NTNs, delays close to 1ms cause the gNB to critically malfunction, often resulting in software crashes. As mentioned earlier, this could change if changes to the timers, synchronization and HARQ procedures were made, but that is out of the scope of this dissertation.

As a result, it became necessary to estimate reception and transmission delay values that would allow the software to function correctly. Using empirical methods it was estimated that these values would be around 500μs for both variables.

$$Distance = RF_{propagationtime} \times c \tag{4.1}$$

where:

c = Speed of light (299 792 458 m/s)

With Equation 4.1 it is possible to calculate the distance simulated by inserting the previously calculated delays. Substituting $RF_{propagationtime}$ by 500μs, the distance calculated is 150Km. Even if this distance falls short of a LEO satellite, the one with the lowest altitude, it is still above the average altitude of a UAS. Since these devices are also being considered NTN platforms by 3GPP and due to the other constraints mentioned, this delay was deemed satisfactory for the purposes of this research.

In addition to the fronthaul delay, a way to insert a backhaul delay was also necessary. Since backhaul connections are not done through radio but rather through ethernet, the way this was achieved was by using the tools developed for network disruption described later in this chapter.

Finally, there was a need to find a way to distinguish both terrestrial and NTN gNBs. As explained in Section 3.2.3, if there was no distinction between them, it would not be possible for the firmware to select exactly which cell to connect to. In any case, the approach taken to solve this problem was to configure each gNB in a different band.

The terrestrial cell is configured to use band n41, with an Absolute Radio-Frequency Channel Number (ARFCN) of 514854, which translates to a frequency of 2574.270MHz. This band was picked because it is widely common and used by 3rd Generation (3G), 4G, and 5G networks. As for the non-terrestrial cell, the band n78 was picked. The gNB is configured to use the 3600 MHz frequency, which means an ARFCN is 640008. This band is also known as C-Band and is commonly used in satellite networks [80].

As discussed before, to test the firmware, the network will need to be launched and taken down multiple times. Since it is split by many machines, doing this task by hand would quickly become inefficient. For this purpose, a program to consistently and quickly launch the network was developed. This program, the launcher, was developed using the Python programming language.

The launcher executes OAI software on a given number of machines, based on the architecture described on a configuration file that is passed by argument. In this configuration

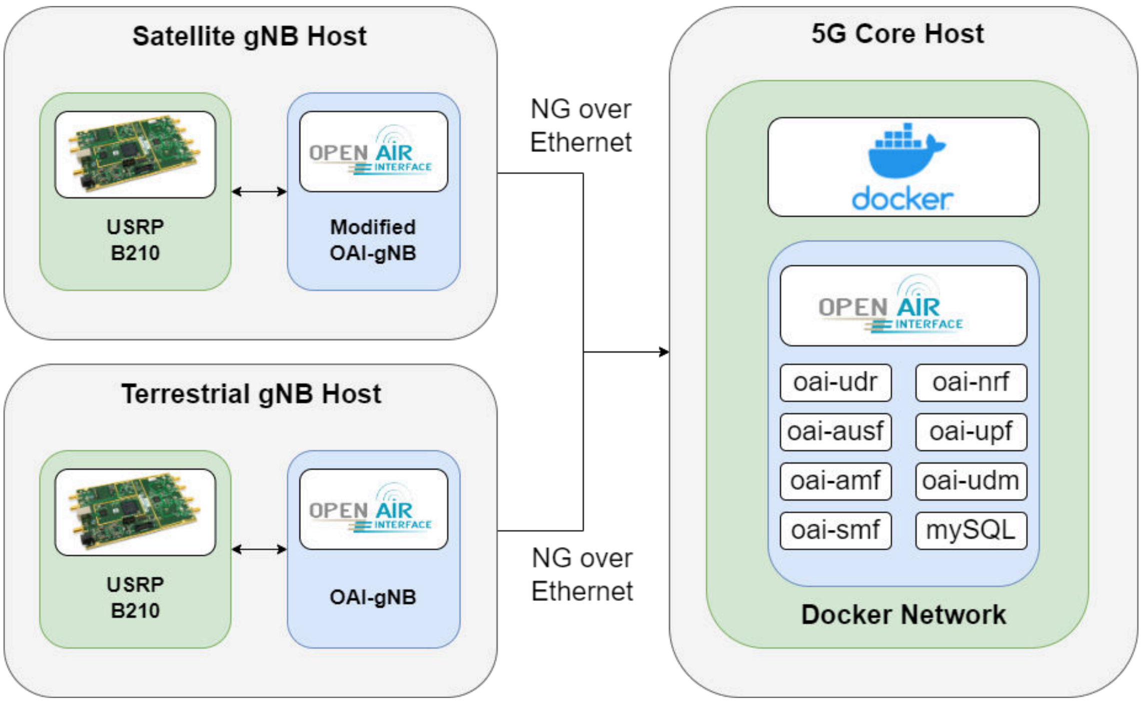


Figure 4.1: 5G network architecture.

file, the user needs to specify what type of module (UE, gNB, core) they want to run for each given machine. Depending on the type of module, the user may pass further configuration parameters. For example, if the machine is to run gNB software, it is possible to detail if the cell is terrestrial or non-terrestrial and define the fronthaul and backhaul delays in case it is, as shown in Fig. 4.2. It is also possible to detail if the gNB should run in a monolithic or CU/DU split architecture.

```
{"ip":"192.168.81.214",
"username":"###########",
"password":"###########",
"type":"gnb",
"conf_file":"gnb2.sa.band78.fr1.106PRB.usrpb210.conf",
"satellite":true,
"delay_rx": 500,
"amf_to_gnb_ip":"192.168.1.15",
"amf_to_gnb_interface":"enp3s0",
"gnb_to_amf_interface":"enxf8e43bbb57d5",
"delay_bh": 10}
```

Figure 4.2: JSON example of a configuration for a satellite gNB.

The launcher sends the commands to other hosts via Secure Shell (SSH). It is expected that these host machines already have an executable build of OAI installed and that the configurations files are already present too.

The launcher uses a threaded approach, spawning each module on an individual thread, allowing each module to be handled independently. If one of the modules fails to launch, it will not affect the others. The logs can also be extracted by module.

When the launcher is closed, all the gNBs and UEs are brought down with it. Because the core network is the module that takes the most time to set up, the launcher does not take it down between runs unless stated, to save time between tests. Every time a network is deployed the software checks the machine set to run the core to see if the core is already running.

4.1.2 Network Disruption Simulator

It had been determined, in Section 3.2.3, that to test the firmware, a catastrophic scenario would not be enough. Using more common, everyday network disruptors tests the conditions the firmware will be operating in more frequently. This can be done by tampering with the network QoS, in order to increase the values of latency and packet loss. Ideally, it would be preferable to be able to induce this both on the fronthaul and backhaul of the network. Indeed, with the changes done to the OAI USRP driver, it is possible to insert latency, but not enough that would cause significant disturbances to any service, because of the restraints detailed in an earlier section. Packet loss technically could be achieved with more changes to OAI software, but it was considered out of scope. Therefore, the focus went to disrupt the backhaul of the network.

The backhaul connections are made using Ethernet network interfaces and the hosts are all running Linux, making the Traffic Control (tc) tool a good choice to tamper the metrics needed [1]. The tc tool implements the necessary structures to make changes to the kernel packet scheduler. What this means in practice is that a user may use it to shape the traffic of a network interface in different ways, such as limiting bandwidth, increasing packet delay, dropping a percentage of packets, or corrupting bits. Tc also allows the creation of filters, which can be used to differentiate traffic, giving different rules to a certain type of packets or traffic from a given source.

With the tool chosen, the next step was to create a Python wrapper for it, to use on other software developed. This wrapper exposes functions to:

- Add latency, with or without jitter;
- Add packet loss;
- Add packet corruption;
- Set-up filters;
- Clear network rules;
- Clear filters.

This wrapper is not only used by the disruption tool that will be described shortly, but also by the launcher tool while setting up a gNB as a satellite.

Employing these functions, the development of the disruption tool was made possible. This tool, also called the Network Disruption Simulator, can be used to run certain behavioral scenarios on given links of a network. These links need to be described in a configuration file,

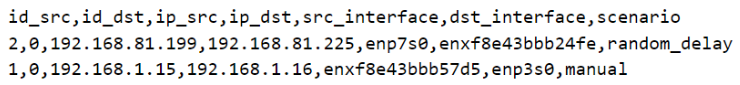
[1]Despite the focus on the backhaul, it is technically possible to use these tools on fronthaul connections if using the OAI UE modem. This modem creates a tunnel interface whose traffic might be shaped using tc. The same does not happen when using radio heads, as done during this research.

that includes source and destination addresses and interfaces, and the scenario to be run on that connection.

The scenario, in this case, is a function that describes how a given link will be degraded during the execution of the program, automating this process. One example of one such scenario is, for example, incrementing the latency by 10ms every 10 seconds. Moreover, it is also possible to change the metrics of the link manually for testing purposes.

It should be noted that, for each link a filter is created, so that the rules inserted only affect traffic from and to the source and destination of that link. What this means is that inserting a delay between the terrestrial gNB and core should not affect the connection between the core and other active gNBs. When finishing the execution of the disruption tool, it cleans all rules and filters created, as so to leave a clean set-up for future uses.

The software may be extended by the addition of new scenarios.



```
id_src,id_dst,ip_src,ip_dst,src_interface,dst_interface,scenario
2,0,192.168.81.199,192.168.81.225,enp7s0,enxf8e43bbb24fe,random_delay
1,0,192.168.1.15,192.168.1.16,enxf8e43bbb57d5,enp3s0,manual
```

Figure 4.3: Example configuration of the Network Disruption tool.

4.2 Firmware Implementation

With the network deployed and the scenario simulator, it is finally possible to move on to the eXtend5G firmware. As it was mentioned before, the main objective of this firmware is to seamlessly switch to the best option between terrestrial or non-terrestrial gNB, at a given moment. Which of them is the best option is determined by analyzing a few network metrics collected by the device. This section is based on the implementation of said firmware and the discussion of the various choices made during the work and how they relate to the specification in the previous chapter.

4.2.1 Hardware

Choosing the hardware to run the firmware was of foremost importance, because of the impact this choice has on the design of the firmware. There was a need for a module that could connect to SA 5G networks, was compatible with OAI and allowed, at least, some degree of control. Ideally, the module would also allow flashing custom firmware, but this is seldom the case with commercial devices.

After some research, the one that was picked was the RMU500EK Development Kit, which comes loaded with a Quectel RM500Q-GL module, for fitting the requisites and being cost-effective. The RM500Q-GL is a module honed for IoT application [81] and is 5G ready. It implements an Attention (AT) command set, which can be used to configure and control it programmatically. Additionally, it is part of the family of modules already tested by the OAI community, ensuring compatibility [82]. Both the Subscriber Identity Module (SIM) card and the module were configured to connect to the OAI network.

Since the module does not allow for custom firmware, it becomes clear that the firmware will have to be run externally. For this external device, a Raspberry Pi 4 Model B was chosen. The Raspberry Pi is a single-board computer that usually runs a Linux Operating System, commonly used in IoT projects, therefore fitting the scenario. It has two USB3 ports, making it compatible with the Quectel Development Kit used. Since it runs Linux, it also allows for flexibility in the implementation, for example regarding the programming language chosen.

This duo of the 5G module and the IoT device corresponds to the UE in the network architecture. With this, two implementation details became clear:

1. the data gathering and analysis would be performed by the Raspberry Pi;
2. the decision must be turned into AT commands for the 5G module.

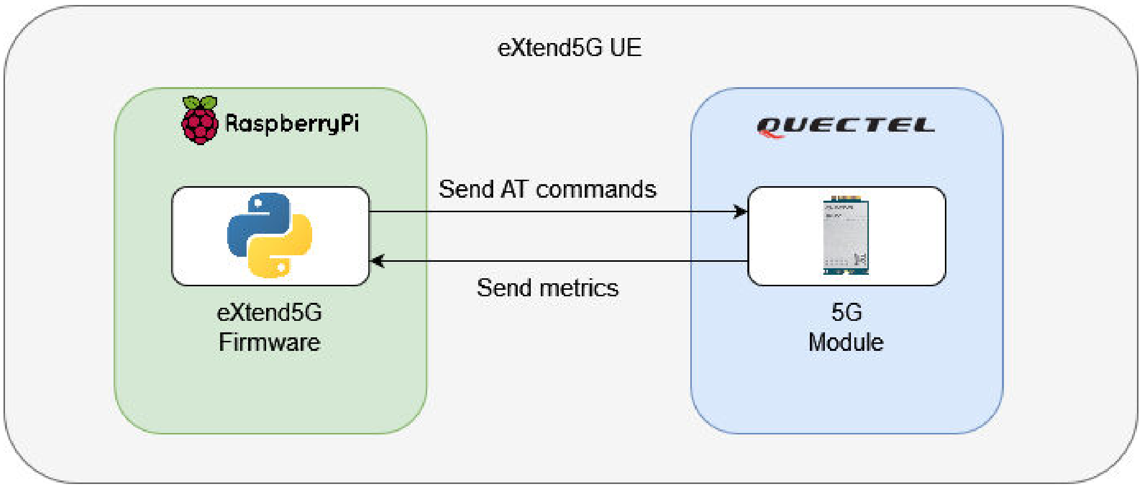


Figure 4.4: eXtend5G UE System Architecture.

The RaspberryPi is running the Raspberry Pi OS, with kernel version 5.15.64-v7l+. This kernel version, at the time of writing, is not stable, however, it supports Quectel devices out of the box without the need for further modification or extensions.

The physical UE may be seen in Fig. 4.5.

4.2.2 Implementation

In concordance with the rest of the research, the language chosen for the implementation of the firmware was Python. Python software usually underperforms when compared to lower-level, compiled languages like C or Rust, in terms of speed and efficiency. However, the ease of development and the speed of prototyping were the heaviest factors in the decision to use it for the firmware.

The first module to be developed was a Python Application Programming Interface (API) for the executions of AT commands on the Quectel 5G module.

Modem API

ATtention command sets are typically used to control modems, such as the one used in this dissertation. Typically, these commands start with 'AT+' and end with a carriage return. These commands can be used to read or set a parameter, or to fetch information from the modem. Each command includes a maximum response time that can be as low as 300ms or as high as 180s. The Quectel Development Kit exposes a serial port to send these commands and fetch their responses. Every AT command produces a result code, for example, "OK" or "ERROR". Read commands also include one or more event/information responses, which are the responses that include the data fetched. These responses start with the character '+'.

From this short description, it is already possible to ascertain some implementation details. For starters, it is necessary to set up a connection with the 5G module serial port. This connection is used to send the crafted AT commands and must be monitored for the responses and other events. Since the native Python library does not include a serial module, the library pySerial was used for these purposes. pySerial is a wrapper for the serial port, providing a simple interface for interaction. This library is used by the modem module to establish a connection with the serial port and to enable the sending of AT commands and receiving the responses as well.

Secondly, there was a need to differentiate the two types of return values of the commands, the result codes, and the information responses. For that two queues were created, for the codes and data returned. Additionally, two send functions were created: one for commands that only have a return code, and the other for commands that require waiting for the return of the information responses. This allows commands that only return a code to execute more quickly. As for commands that may have information responses, the program reads the information queue until a result code is received. Some commands return more than one information result at different times, so it is important to lock the information queue until every response comes through.

With this, the module was ready to handle any AT command. Next was the choice of which AT commands to implement in the API for the firmware. A look at the RG50xQ&RM5xxQ Series AT Commands manual [83] shows how extensive the command set is, so it was not practicable to implement a function and a parser for each of them. Therefore, it was decided to only implement those that would be used by the eXtend5G firmware.

Hence, what follows is a description of the 5G module interface developed. Its functions can be loosely divided into two types, the setters, and the getters.

The setters are commands used solely for configuration purposes. For example, to allow the module to connect to a OAI network, it is necessary to set the Access Point Name (APN) to 'oai' and set the Packet Data Protocol (PDP) type to 'IP'. The module includes a setter for PDP contexts made for this exact purpose. It may also be desirable to limit the network searching capabilities to 5G only, therefore, the interface includes a network mode setter. The commands described until now are indeed the ones used to set the default modem configuration for the project.

An especially relevant setter is the one for the NR bands the 5G module is allowed to search in. By default, the modem does not have a AT command that allows the user to select the exact cell for attachment. This selection is done by the proprietary firmware of the module and, as established before, it cannot be changed. Since the eXtend5G firmware required the module to switch between satellite and terrestrial gNB, a way to connect to specific gNBs needed to be found. So it was decided to deploy the gNB in different bands and limit the 5G module band search to only the band of the mode it is supposed to be currently on according to the state machine, therefore achieving some control over the gNB it connects to.

The last setter implemented is related to the attach and detach from Packet Switching (PS) sessions.

The purpose of the getter commands is to fetch data from the 5G module. The interface includes getters to fetch the RSRQ and the RSRQ so it is possible to measure the current signal status. It also has a getter to get the current serving cell info and another to get every cell that the UE can find at that given moment.

Whenever possible, these functions were implemented using 3GPP-defined AT commands, which ensure compatibility with other 5G modules. However, that was not possible for some of the functionalities required. Even so, to make the firmware work with non-Quectel modems, it is possible to extend the API and change the functions where proprietary commands are used, as long as they produce the same results.

How the functions of this interface are used will be made clear now, when discussing the firmware module.

Firmware Module

The implementation of the eXtend5G firmware closely follows the state machine design specified beforehand. On the Init state, the firmware sets up its initial state and the class variables. It queries the 5G module for its state, mostly what state the modem is now and what link type it is connected to.

There are several link types. The link is classified by Satellite if the module is connected to a gNB on the satellite band, or by Terrestrial. Another link type is Unknown link, in case the module is connected to another type of RAT. If the modem is not connected at all, the type of the link will be "NoLink".

The modem can be either in SEARCH state if it is disconnected and looking for a cell, LIMSRV if it is camping on a cell with no service, NOCONN, if it is connected and registered in a cell without an ongoing call, and CONNECTED.

Initially, the firmware would also search for all the nearby cells. This information would be used to check if there are other terrestrial gNBs available before switching to satellite. However, running the AT command that would return this data would instantly crash all the OAI gNBs running. For this reason, the idea had to be dropped.

Based on the band the modem is connected to, the firmware decides if the next state is either going to be Satellite or Terrestrial. Both these state function in a similar matter. They monitor the RSRQ, the latency, and the packet loss of the current service. The signal strength values can be fetched from the modem using the API created.

The rest of the metrics required the creation of a new function, the ping. As the name might indicate, this function runs a ping to a given server and parses the output to extract both latency and packet loss. The parsing is done by a regular expression. Consequently, since the ping output is different in Windows platforms, the firmware may only be run in

Linux systems. Also, while the ping tool does not give the most accurate results, nor should it be used on an automated program, it was chosen since it was the tool that required the least setup. Other options like nmap or iperf were considered, but they either required the setup of a server or they did not give the information necessary for the implementation of the firmware.

The location of the UE was initially considered, as mentioned in the previous chapter, but discarded since there the Development Kit does not include GPS capabilities and, even if it did, it would require additional work to simulate the satellite orbit.

After fetching all the metrics, the firmware is able to make a decision. If the signal strength is deemed poor or worse, and there are no other gNBs available, it triggers a switch. On the contrary, if the signal is good, it checks if the latency or packet loss is below the thresholds. The thresholds are not fixed and can be set by argument when running the firmware.

Whenever a switch is signaled, the firmware transits to the Switch state. To switch, the modem is first signaled to close the current PS session. If the PS session is not closed before attempting a switch, it will cause the gNB to crash. After the session is closed, the firmware sets the module band to the one it is switching to. Afterward, it tries to attach to a new PS session. If this process is successful, the modem should be in NOCONN state, connected to the new gNB. If the firmware detects this, it changes state to either Terrestrial or Satellite. However, if after three queries, the modem has not established a new session, the firmware switches to Disconnected state.

In this state, the firmware sets the 5G module to search on both satellite and terrestrial bands until it finds an gNB to attach to. The firmware will periodically monitor the modem until it connects to one gNB. After detecting a connection, it will switch to the appropriate state so that normal behavior can resume.

It is important to note that the RAN software used, OAI, does not implement any type of handover at the moment, so no mobility functions are triggered by the gNB. The UE is responsible for all selection processes.

This firmware can be run either on bare metal or it can be containerized. The serial port and the Internet Protocol (IP) address attributed to the 5G module have to be passed as arguments. The 5G module must be connected to the machine.

In this chapter, the implementation choices that were made were described. While it was not possible to concretize every detail of the specification defined, a working proof of concept was made. Following the methodology designed, the next step is to validate this solution. This validation and its results are presented on Chapter 5.

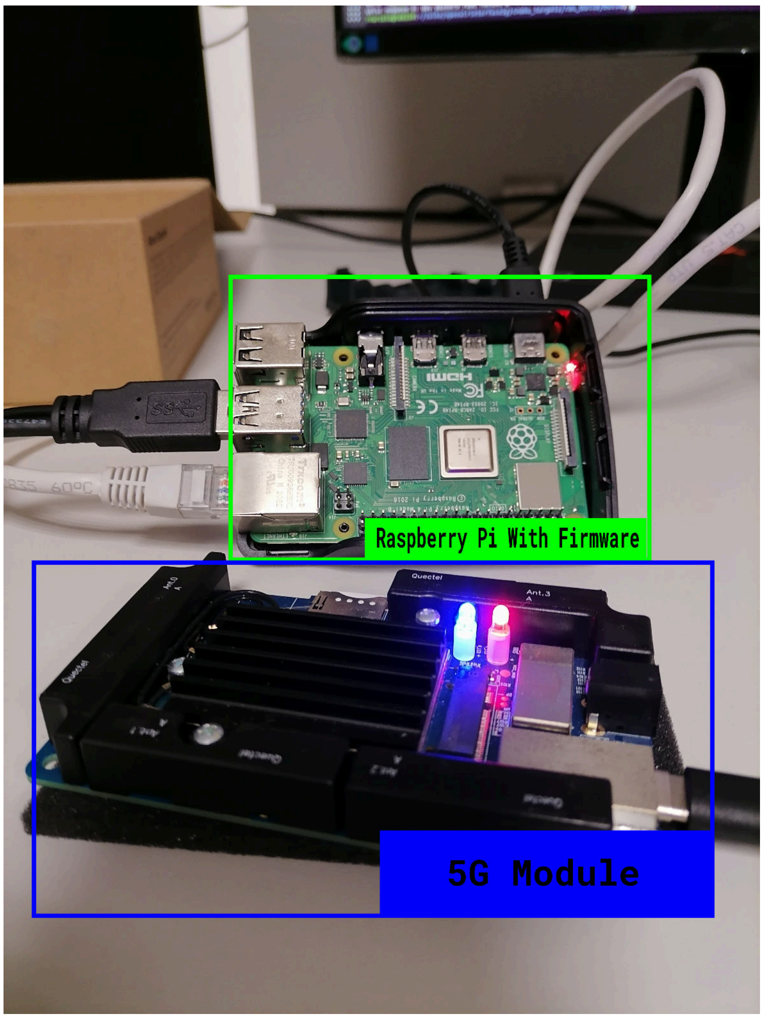


Figure 4.5: eXtend5G Hardware.

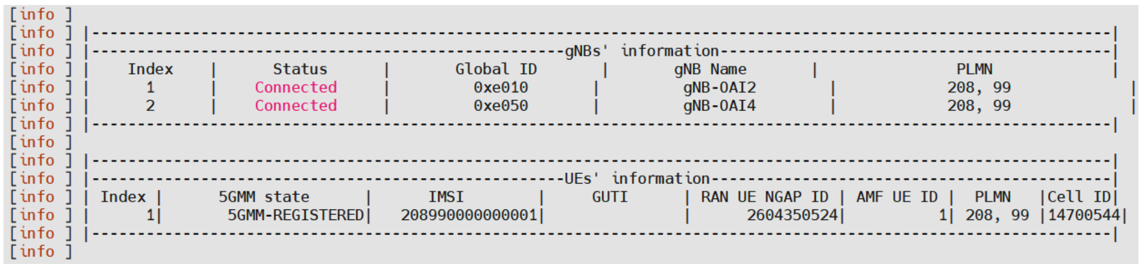
CHAPTER 5

Results

This chapter presents the tests run to validate the developed software. It focuses on how the tests were designed, what configurations were used and how they evaluate the solutions developed.

5.1 Testbench Validation

To test our firmware, it was necessary to assure that the testbench tools were producing the intended results. As aforementioned, the network deployment should include two gNBs and a core. So it is expected that after executing the launcher tool both gNBs are registered on the core AMF. This can be confirmed by looking at the AMF logs as represented in Fig. 5.1.

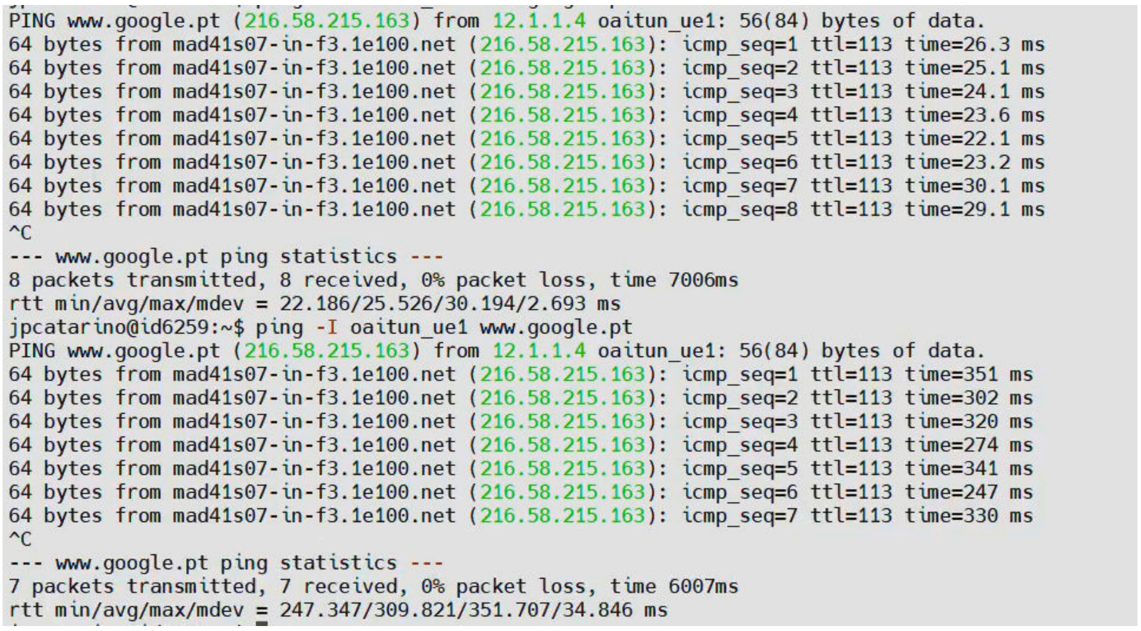


```
[info ]
[info ] |----------------------------------------------------------------------------------------------------|
[info ] |----------------------------------------------gNBs' information-------------------------------------|
[info ] |    Index    |      Status      |       Global ID       |       gNB Name       |              PLMN             |
[info ] |      1      |    Connected     |        0xe010        |       gNB-OAI2        |             208, 99             |
[info ] |      2      |    Connected     |        0xe050        |       gNB-OAI4        |             208, 99             |
[info ] |----------------------------------------------------------------------------------------------------|
[info ]
[info ] |----------------------------------------------------------------------------------------------------|
[info ] |-----------------------------------------------UEs' information-------------------------------------|
[info ] | Index |      5GMM state      |      IMSI        |     GUTI      | RAN UE NGAP ID | AMF UE ID |  PLMN   |Cell ID|
[info ] |      1|       5GMM-REGISTERED|   208990000000001|               |      2604350524|          1| 208, 99 |14700544|
[info ] |----------------------------------------------------------------------------------------------------|
[info ]
```

Figure 5.1: OAI AMF log depicting two deployed gNBs.

The satellite gNB should have a fronthaul delay configured. A glimpse of how this delay affects the UE processes may be seen in Fig. 5.3.

As for the network degradation tool, it was validated empirically while running tests for the firmware. In Fig. 5.2, there are the results of two ping commands. The first was executed before using the network degradation tool. The second was executed after using the network degradation tool to add a random delay to the link between gNB the UE was connected to and the core.



```
PING www.google.pt (216.58.215.163) from 12.1.1.4 oaitun_ue1: 56(84) bytes of data.
64 bytes from mad41s07-in-f3.1e100.net (216.58.215.163): icmp_seq=1 ttl=113 time=26.3 ms
64 bytes from mad41s07-in-f3.1e100.net (216.58.215.163): icmp_seq=2 ttl=113 time=25.1 ms
64 bytes from mad41s07-in-f3.1e100.net (216.58.215.163): icmp_seq=3 ttl=113 time=24.1 ms
64 bytes from mad41s07-in-f3.1e100.net (216.58.215.163): icmp_seq=4 ttl=113 time=23.6 ms
64 bytes from mad41s07-in-f3.1e100.net (216.58.215.163): icmp_seq=5 ttl=113 time=22.1 ms
64 bytes from mad41s07-in-f3.1e100.net (216.58.215.163): icmp_seq=6 ttl=113 time=23.2 ms
64 bytes from mad41s07-in-f3.1e100.net (216.58.215.163): icmp_seq=7 ttl=113 time=30.1 ms
64 bytes from mad41s07-in-f3.1e100.net (216.58.215.163): icmp_seq=8 ttl=113 time=29.1 ms
^C
--- www.google.pt ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7006ms
rtt min/avg/max/mdev = 22.186/25.526/30.194/2.693 ms
jpcatarino@id6259:~$ ping -I oaitun_ue1 www.google.pt
PING www.google.pt (216.58.215.163) from 12.1.1.4 oaitun_ue1: 56(84) bytes of data.
64 bytes from mad41s07-in-f3.1e100.net (216.58.215.163): icmp_seq=1 ttl=113 time=351 ms
64 bytes from mad41s07-in-f3.1e100.net (216.58.215.163): icmp_seq=2 ttl=113 time=302 ms
64 bytes from mad41s07-in-f3.1e100.net (216.58.215.163): icmp_seq=3 ttl=113 time=320 ms
64 bytes from mad41s07-in-f3.1e100.net (216.58.215.163): icmp_seq=4 ttl=113 time=274 ms
64 bytes from mad41s07-in-f3.1e100.net (216.58.215.163): icmp_seq=5 ttl=113 time=341 ms
64 bytes from mad41s07-in-f3.1e100.net (216.58.215.163): icmp_seq=6 ttl=113 time=247 ms
64 bytes from mad41s07-in-f3.1e100.net (216.58.215.163): icmp_seq=7 ttl=113 time=330 ms
^C
--- www.google.pt ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6007ms
rtt min/avg/max/mdev = 247.347/309.821/351.707/34.846 ms
```

Figure 5.2: Ping command results.

5.2 Firmware Validation

5.2.1 Test Description

To validate the eXtend5G firmware and to gauge its performance, three main tests were devised.

The first test pretends to emulate a scenario where the gNB that the UE is attached to goes out of service. The purpose of this test will be to estimate the time that the firmware takes to perform a switch.

The objective of the second test is to simulate a scenario of steadily increasing latency, in order to validate the logic of the solution.

The third test is similar to the latter, but instead of the latency, the metric tested will be packet loss.

Other tests were considered, but ultimately not executed. One of them was to meddle with the RSRP and RSRQ. This test has to be discarded because the environment where the tests were executed was not controlled, resulting in an unmanageable degradation of those indicators, which made it hard to reliable execute the tests.

All the tests will be executed using the UE with and without firmware so that it is possible to compare both behaviors and evaluate whether using the firmware makes a difference or not.

5.2.2 Set-up and Configuration

Every test performed was executed using the same set-up, the one described in 4.1.1.

This set-up includes two OAI gNBs, one terrestrial and another configured as satellite, running in Ubuntu machines with a 12th Gen Intel(R) Core(TM) i9-12900K CPU and a total of 64 GB of RAM, and a OAI 5G core network that is executed in a Ubuntu machine with an Intel(R) Core(TM) i5-4440 CPU with a total of 16GB of memory.

The terrestrial gNB is configured to use the n41 band. The satellite gNB uses the n78 band and is configured to have a fronthaul delay of 500μs and a backhaul delay of 10ms. The purpose of the increased backhaul delay is to further differentiate the latency of the terrestrial gNB from the satellite gNB.

The only exception to this is that a network without any configured delay was deployed for the first test, for reasons explained further on.

The tests will start with the UE connected to the terrestrial gNB. When using the UE with the firmware, the device will be configured to only scan terrestrial bands at the beginning. When running without the firmware, the 5G module will be configured to scan both terrestrial and satellite bands, and no configurative AT commands will be run while it is executing.

The thresholds configured for the firmware are 50ms plus the satellite link delay in case of the latency and 3% of packet loss.

5.2.3 Switching Test

As mentioned in Section 5.2.1, the objective of this test is to replicate a scenario where the terrestrial gNB crashes, making the gNB have to switch. This process is timed for comparison.

To trigger this behaviour, the OAI gNB that the UE is connected to is simply shut down. Measuring the time the UE without firmware required the creation of a script that would monitor its current status. The timer starts when the UE reports that's disconnected and stops when it reports that it is attached to a new gNB. This means that the time to detect the gNB was down is not accounted for in the measurements.

The UE without firmware was tested in two different network configurations, one with the satellite delays and another without any type of delay, effectively with two gNBs acting as terrestrial. The motivation for this was to check whether the fronthaul delay had an effect on the usual RRC registration procedures.

To measure the time to switch of a UE a similar process was followed.

It is acknowledged that some non-deterministic factors, with origins in various sources, from the environment to the OS used, may influence the results taken.

In Fig. 5.3, the results of this test are present. The time to switch calculated is the average of eight different readings for each value. Each reading was done under similar conditions.

By comparing the switching times of both readings without firmware, it's possible to conclude that the satellite delays introduce a 10% increase in the switching time overall. Using the firmware developed grants a decrease of 28% on the switching time, which will ultimately lead to less downtime every time the device has to switch from terrestrial to satellite cells.

However, these values represent best-case scenarios for the firmware. Some of the AT commands used by the firmware report a Maximum Report Time of up to 140 seconds, including those used when triggering a switch. This means that these values may not always be observed under every condition.

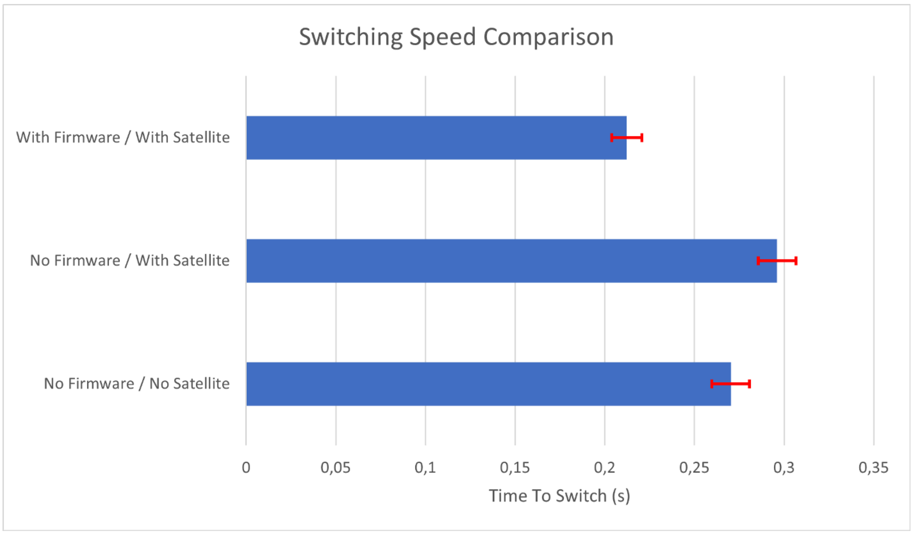


Figure 5.3: Switching test results.

5.2.4 Latency and Packet Loss Test

The focus of these tests was to put to test the developed solution with increasing delays and packet loss.

For this purpose, two scenarios were developed using the network disruption tool, to generate a similar pattern of degradation for the tests run. No two runs are exactly the same, due to the natural and simulated network jitter that may happen but the values were deemed close enough.

Each scenario will run for 90 seconds. During this time, information on latency and packet loss will be collected by the device, using the ping function, configured to send ten packets. Similarly to what was done in the first test, the connection status of the UE will also be monitored.

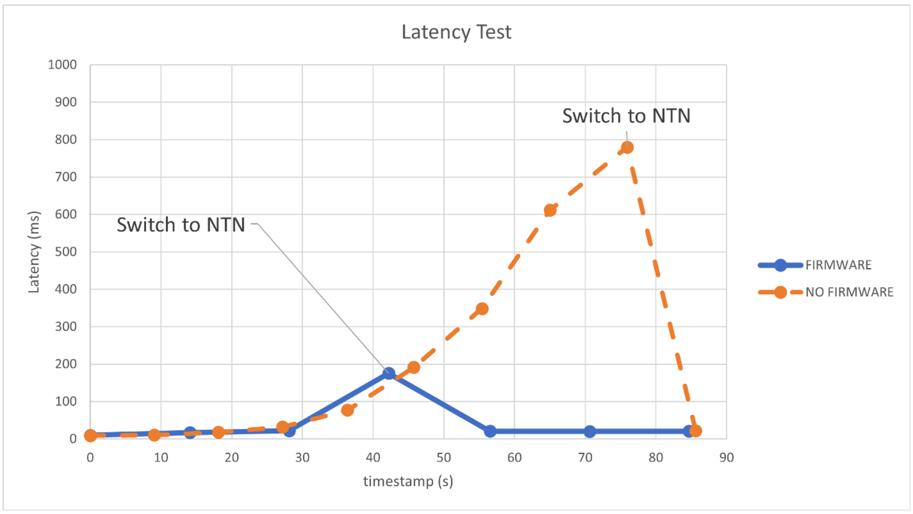


Figure 5.4: Latency Test Results.

Fig.5.4 and Fig.5.5 are the results of a run of each of the tests described. As expected, the

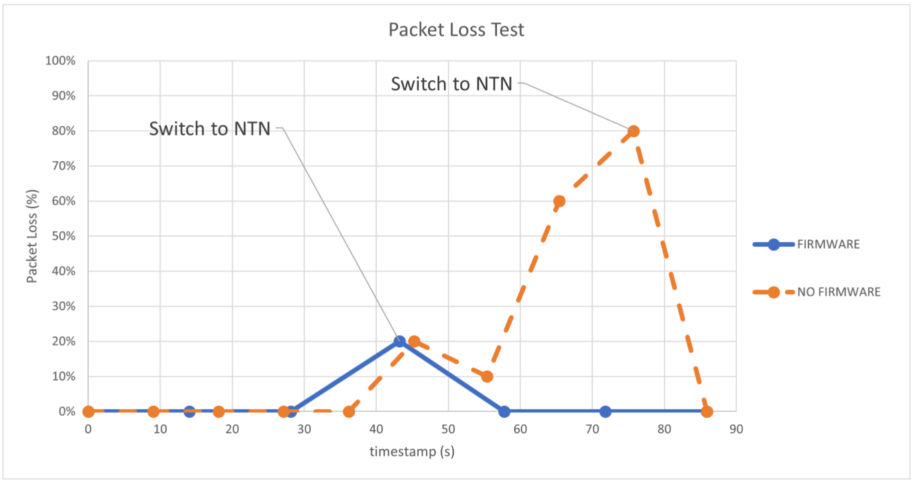
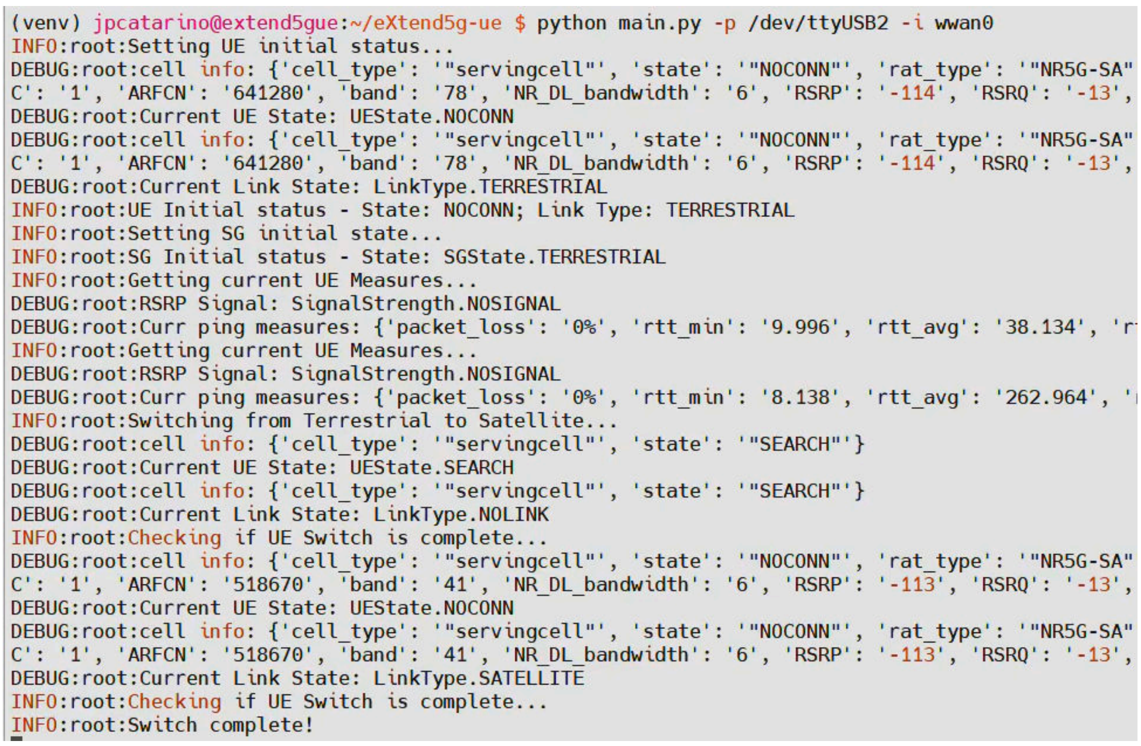


Figure 5.5: Packet Loss Test Results.

UE when using the firmware signals a switch when the values monitored cross the thresholds defined. After registering 200ms of latency in one case and around 20% of packet loss in the other, the firmware decides to switch to the alternative so that it can keep up with the requirements set by the user.

On the other hand, the UE when not using the firmware usually only triggers a cell selection when degradation levels are at values up to delays of 800ms and 80% packet loss, when most services would already be severely impacted.

However, the time between each cycle of the firmware could potentially be improved. As can be observed in Fig. 5.4, each reading approximately occurs every 14 seconds, the sum of the wait time between cycles and the duration of the ping command. In this case, this caused the firmware to act later, letting the network condition degrade for longer. It could be improved by reducing the wait time and the number of packets sent to calculate the latency.



```
(venv) jpcatarino@extend5gue:~/eXtend5g-ue $ python main.py -p /dev/ttyUSB2 -i wwan0
INFO:root:Setting UE initial status...
DEBUG:root:cell info: {'cell_type': '"servingcell"', 'state': '"NOCONN"', 'rat_type': '"NR5G-SA"
C': '1', 'ARFCN': '641280', 'band': '78', 'NR_DL_bandwidth': '6', 'RSRP': '-114', 'RSRQ': '-13',
DEBUG:root:Current UE State: UEState.NOCONN
DEBUG:root:cell info: {'cell_type': '"servingcell"', 'state': '"NOCONN"', 'rat_type': '"NR5G-SA"
C': '1', 'ARFCN': '641280', 'band': '78', 'NR_DL_bandwidth': '6', 'RSRP': '-114', 'RSRQ': '-13',
DEBUG:root:Current Link State: LinkType.TERRESTRIAL
INFO:root:UE Initial status - State: NOCONN; Link Type: TERRESTRIAL
INFO:root:Setting SG initial state...
INFO:root:SG Initial status - State: SGState.TERRESTRIAL
INFO:root:Getting current UE Measures...
DEBUG:root:RSRP Signal: SignalStrength.NOSIGNAL
DEBUG:root:Curr ping measures: {'packet_loss': '0%', 'rtt_min': '9.996', 'rtt_avg': '38.134', 'r
INFO:root:Getting current UE Measures...
DEBUG:root:RSRP Signal: SignalStrength.NOSIGNAL
DEBUG:root:Curr ping measures: {'packet_loss': '0%', 'rtt_min': '8.138', 'rtt_avg': '262.964', '
INFO:root:Switching from Terrestrial to Satellite...
DEBUG:root:cell info: {'cell_type': '"servingcell"', 'state': '"SEARCH"'}
DEBUG:root:Current UE State: UEState.SEARCH
DEBUG:root:cell info: {'cell_type': '"servingcell"', 'state': '"SEARCH"'}
DEBUG:root:Current Link State: LinkType.NOLINK
INFO:root:Checking if UE Switch is complete...
DEBUG:root:cell info: {'cell_type': '"servingcell"', 'state': '"NOCONN"', 'rat_type': '"NR5G-SA"
C': '1', 'ARFCN': '518670', 'band': '41', 'NR_DL_bandwidth': '6', 'RSRP': '-113', 'RSRQ': '-13',
DEBUG:root:Current UE State: UEState.NOCONN
DEBUG:root:cell info: {'cell_type': '"servingcell"', 'state': '"NOCONN"', 'rat_type': '"NR5G-SA"
C': '1', 'ARFCN': '518670', 'band': '41', 'NR_DL_bandwidth': '6', 'RSRP': '-113', 'RSRQ': '-13',
DEBUG:root:Current Link State: LinkType.SATELLITE
INFO:root:Checking if UE Switch is complete...
INFO:root:Switch complete!
```

Figure 5.6: Example of a firmware output (cropped for visibility).

CHAPTER 6 Conclusion

In the next few years, satellites may be co-existing with terrestrial cell towers in 5G networks to bring mobile coverage to places and people that couldn't be reached before, for geographical or economical reasons.

In this dissertation, a new firmware solution that included a backhaul-aware, cell selection scheme, was designed for use in the emergent hybrid, terrestrial and non-terrestrial, 5G networks. The aspiration behind this firmware was to expand upon the commonly used signal strength selection methods, adding new metrics so that the UE may have a clear picture of the network status before making the decision to switch to satellite and vice versa. Using this firmware should help devices keep a connection when in underserved areas or during natural catastrophes.

A proof of concept 5G UE running a partial implementation of said firmware was created and validated on an end-to-end deployment of a 5G network with terrestrial and non-terrestrial links. With the use of a disruption tool, the conditions of the network were tampered with in order to test the behavior of the firmware.

The results suggest that the solution developed may help the devices that use it to keep an acceptable level of service quality in situations where the terrestrial links are under stress or malfunctioning. The firmware also offers a potential 28% improvement in switching speed which makes up for a lower downtime.

This research was limited by the equipment and software solutions available at the time. For example, the version of OAI used crashed when the modem executed certain AT commands, which ultimately led to a feature not being implemented. Also, no current platform was ready to emulate satellite conditions on a gNB, including the high propagation delays and orbits. It would have been interesting to test this firmware under a LEO constellation or with more than one terrestrial gNB, but this would require acquiring more hardware.

The tests also fail to consider situations where the satellite link may be more degraded than the terrestrial one, and how the firmware would adapt to such scenarios.

Despite the limitations, this work may still offer some insight into how to adapt the UEs to take advantage of the new paradigm, in some scenarios. It also gives some pointers on how a hybrid 5G testbench may be deployed to test devices over the air.

6.1 Future Work

When wrapping this research up, it was made clear that there are still further improvements that may be done. Since this is an emerging area, there are also a lot of new opportunities to explore.

The most obvious next steps are to implement the features that had to be discarded, such as the use of the satellite location to check if one is available to complete or anticipate a switch.

Adapting OAI to simulate higher propagation delays might be a worthwhile pursuit to make these tools closer to the satellite reality.

In this research, we only considered one device, but a network may have hundreds of devices connected. It would be worthwhile exploring a scenario with more than one UE in hybrid networks, monitoring how the network reacts to this number of UE switching from terrestrial to non-terrestrial. Ultimately this may lead to adapting the firmware to have some attention to the question of load balancing, as to avoid satellite congestion.

References

[1] J. Cadete de Matos, «ANACOM and 5G Regulation», *INESC TEC Science&Society*, vol. 1, no. 3, Dec. 2021. [Online]. Available: `https://science-society.inesctec.pt/index.php/inesctecesociedade/article/view/anacom_and_5g_regulation`.

[2] World Economic Forum, «The Impact of 5G: Creating New Value across Industries and Society», *World Economic Forum*, no. January, p. 24, 2020. [Online]. Available: `http://www3.weforum.org/docs/WEF_The_Impact_of_5G_Report.pdf`.

[3] F. W. Vook, A. Ghosh, E. Diarte, and M. Murphy, «5G New Radio: Overview and Performance», in *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, 2018, pp. 1247–1251. DOI: `10.1109/ACSSC.2018.8645228`.

[4] 3GPP, «Release 15 Description; Summary of Rel-15 Work Items», 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 21.915, Oct. 2019, Version 15.0.0. [Online]. Available: `https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3389`.

[5] S. Sirotkin, *5G Radio Access Network Architecture*. 2021, pp. 68–70, ISBN: 9781119550884.

[6] N. Pachler, I. del Portillo Barrios, E. Crawley, and B. Cameron, «An Updated Comparison of Four Low Earth Orbit Satellite Constellation Systems to Provide Global Broadband», 2021, pp. 1–7. DOI: `10.1109/ICCWorkshops50388.2021.9473799`.

[7] M. Anderson and E. A. Vogels, *Americans turn to technology during covid-19 outbreak, say an outage would be a problem*, Jul. 2020. [Online]. Available: `https://www.pewresearch.org/fact-tank/2020/03/31/americans-turn-to-technology-during-covid-19-outbreak-say-an-outage-would-be-a-problem/`.

[8] ITU, *Measuring digital development. Facts and figures 2021*. 2021, pp. 1–15, ISBN: 9789261295110. [Online]. Available: `https://www.itu.int/en/mediacentre/Documents/MediaRelations/ITU%20Facts%20and%20Figures%202019%20-%20Embargoed%205%20November%201200%20CET.pdf`.

[9] B. Whitacre, *Technology is improving – why is rural broadband access still a problem?*, 2021. [Online]. Available: `https://theconversation.com/technology-is-improving-why-is-rural-broadband-access-still-a-problem-60423`.

[10] ANACOM, *tem.REDE*. [Online]. Available: `https://anacom.maps.arcgis.com/apps/Cascade/index.html?appid=ad3f71dbb09541518f436aa828feb28e`.

[11] 3GPP, «Solutions for NR to support non-terrestrial networks (NTN)», 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.821, Jun. 2021, Version 16.1.0. [Online]. Available: `https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3525`.

[12] ——, «Study on New Radio (NR) to support non-terrestrial networks», 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.811, Oct. 2020, Version 15.4.0. [Online]. Available: `https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3234`.

[13] ——, «Study on architecture aspects for using satellite access in 5G», 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.737, Mar. 2021, Version 17.2.0. [Online]. Avail-

able: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3485.

[14] *Lockheed Martin and Omnispace explore space-based 5G Global Network*, Mar. 2021. [Online]. Available: https://news.lockheedmartin.com/5g-omnispace-agreement.

[15] R12-SG05, «IMT Vision - Framework and overall objectives of the future development of IMT for 2020 and beyond», International Telecommunication Union (ITU), Recommendation M.2083, Oct. 2015. [Online]. Available: https://www.itu.int/rec/R-REC-M.2083.

[16] 3. G. P. P. (3GPP), *Releases*, 2022. [Online]. Available: https://www.3gpp.org/specifications/67-releases.

[17] Cisco, «5G Non Standalone Solution Overview», Cisco Systems, Tech. Rep. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/wireless/asr_5000/21-9_6-3/5G-NSA-Solution/21-9-5G-NSA-Solution-Guide/21-9-5G-NSA-Solution-Guide_chapter_01.pdf.

[18] F. Mademann, *System architecture milestone of 5G Phase 1 is achieved*, 2017. [Online]. Available: https://www.3gpp.org/news-events/3gpp-news/sys-architecture (visited on 11/30/2022).

[19] 3GPP, «System architecture for the 5G System (5GS)», 3rd Generation Partnership Project (3GPP), Technical specification (TS) 23.501, Dec. 2021, Version 16.11.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144.

[20] B. Bertenyi, R. Burbidge, G. Masini, S. Sirotkin, and Y. Gao, «NG radio access network (NG-RAN)», *Journal of ICT Standardization*, vol. 6, no. 1, pp. 59–76, 2018.

[21] 3GPP, «NG-RAN; Architecture description», 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.401, Dec. 2021, Version 16.8.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3219.

[22] ——, «NR; NR and NG-RAN Overall description; Stage-2», 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.300, Dec. 2021, Version 16.8.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3191.

[23] J. Kim, G. Jo, and J. Jeong, «A Novel CPPS Architecture Integrated with Centralized OPC UA server for 5G-based Smart Manufacturing», *Procedia Computer Science*, vol. 155, pp. 113–120, 2019. DOI: 10.1016/j.procs.2019.08.019.

[24] C. Johnson, *5G New Radio in Bullets*. Independently Published, 2019, ISBN: 9781077484351. [Online]. Available: https://books.google.pt/books?id=NoRjyAEACAAJ.

[25] Samsung, «Virtualized Radio Access Network: Architecture, Key technologies and Benefits», 2019. [Online]. Available: https://images.samsung.com/is/content/samsung/assets/global/business/networks/insights/white-papers/virtualized-radio-access-network/white-paper_virtualized-radio-access-network_1.pdf.

[26] 3GPP, «Study on new radio access technology: Radio access architecture and interfaces», 3rd Generation Partnership Project (3GPP), Technical Report (TR) 38.801, Apr. 2017, Version 14.0.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3056.

[27] O.-R. A. W. G. 4, «O-RAN Fronthaul Control, User and Synchronization Plane Specification», O-RAN ALLIANCE, Technical Specification (TS) O-RAN.WG4.CUS.0, Jul. 2021, Version 07.00. [Online]. Available: https://www.o-ran.org/specification-access.

[28] O.-R. A. W. G. 1, «O-RAN Architecture Description», O-RAN ALLIANCE, Technical Specification (TS) O-RAN.WG1.O-RAN-Architecture-Description-v05.00, Jul. 2021, Version 5.0. [Online]. Available: https://www.o-ran.org/specification-access.

[29] O.-R. ALLIANCE, «O-RAN Use Cases and Deployment Scenarios», Feb. 2020. [Online]. Available: https://static1.squarespace.com/static/5ad774cce74940d7115044b0/t/5e95a0a306c6ab2d1cbca4d3/1586864301196/O-RAN+Use+Cases+and+Deployment+Scenarios+Whitepaper+February+2020.pdf.

[30] 3GPP, «NR; Radio Resource Control (RRC); Protocol specification», 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.331, Sep. 2021, Version 16.6.0. [Online]. Avail-

able: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3197.

[31] D. Amzallag, R. Bar-Yehuda, D. Raz, and G. Scalosub, «Cell Selection in 4G Cellular Networks», in *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*, 2008, pp. 700–708. DOI: 10.1109/INFOCOM.2008.120.

[32] 3GPP, «NR; User Equipment (UE) procedures in idle mode and in RRC Inactive state», 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.304, Dec. 2021, Version 16.7.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3192.

[33] Twilio, *How to determine good cellular signal strength.* [Online]. Available: https://www.twilio.com/docs/iot/supersim/how-determine-good-cellular-signal-strength.

[34] 3GPP, «Procedures for the 5G System (5GS)», 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.502, Dec. 2021, Version 17.3.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3145.

[35] I. Dd, J. Xu, G. Shi, and C.-X. Wang, *5G Wireless Systems: Simulation and Evaluation Techniques.* 2018, ISBN: 978-3-319-61868-5. DOI: 10.1007/978-3-319-61869-2.

[36] J. Lee and Y. Yoo, «Handover cell selection using user mobility information in a 5G SDN-based network», in *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, 2017, pp. 697–702. DOI: 10.1109/ICUFN.2017.7993880.

[37] I. A. Alablani and M. A. Arafah, «An Adaptive Cell Selection Scheme for 5G Heterogeneous Ultra-Dense Networks», *IEEE Access*, vol. 9, pp. 64 224–64 240, 2021, ISSN: 21693536. DOI: 10.1109/ACCESS.2021.3075324.

[38] S. E. A. Abdalla, A. A. Abdalla, and M. H. Khalil, «Application-based selection for 5G networks», in *Proceedings of: 2020 International Conference on Computer, Control, Electrical, and Electronics Engineering, ICCCEEE 2020*, 2021, ISBN: 9781728191119. DOI: 10.1109/ICCCEEE49695.2021.9429556.

[39] M. Chiputa, M. Zhang, G. G. M. N. Ali, P. H. J. Chong, H. Sabit, A. Kumar, and H. Li, «Enhancing Handover for 5G mmWave Mobile Networks Using Jump Markov Linear System and Deep Reinforcement Learning», *Sensors*, vol. 22, no. 3, 2022, ISSN: 1424-8220. DOI: 10.3390/s22030746. [Online]. Available: https://www.mdpi.com/1424-8220/22/3/746.

[40] OSA, *5G Core Network.* [Online]. Available: https://openairinterface.org/oai-5g-core-network-project/.

[41] A. Younis, B. Qiu, and D. Pompili, «Latency and quality-aware task offloading in multi-node next generation RANs», *Computer Communications*, vol. 184, pp. 107–117, 2022, ISSN: 0140-3664. DOI: https://doi.org/10.1016/j.comcom.2021.11.026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366421004655.

[42] N. Salhab, R. Langar, and R. Rahim, «5G network slices resource orchestration using Machine Learning techniques», *Computer Networks*, vol. 188, p. 107 829, 2021, ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2021.107829. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128621000165.

[43] O. Contributors, *Feature Set - master.* [Online]. Available: https://gitlab.eurecom.fr/oai/openairinterface5g/blob/master/doc/FEATURE_SET.md.

[44] T. Schlichter, *Adaptations of OAI for Satellite.* 2021. [Online]. Available: https://www.openairinterface.org/docs/workshop/2020-11-VIRTUAL-EVENT/13-B-SCHLICHTER-FRAUNHOFER-IIS-OAI-5G-NR-NTN.mp4.

[45] O. S. Alliance, *5G SpaceLab: Unleash the Potential of Future Space Communication.* 2021. [Online]. Available: https://www.youtube.com/watch?v=ZvbA-O9TDkQ.

[46] O. Kodheli, A. Astro, J. Querol, M. Gholamian, S. Kumar, N. Maturo, and S. Chatzinotas, «Random Access Procedure Over Non-Terrestrial Networks: From Theory to Practice», *IEEE Access*, vol. 9, pp. 109 130–109 143, 2021. DOI: 10.1109/ACCESS.2021.3101291.

[47] E.-A. Papatheofanous, D. Reisis, and K. Nikitopoulos, «LDPC Hardware Acceleration in 5G Open Radio Access Network Platforms», *IEEE Access*, vol. PP, p. 1, 2021. DOI: 10.1109/ACCESS.2021.3127039.

[48] O. S. Alliance, *Fall 2021 OpenAirInterface Workshop: Plenary track*. 2021. [Online]. Available: https://www.youtube.com/watch?v=S6QxA1sYUiM.

[49] R. Schmidt, M. Irazabal, and N. Nikaein, «FlexRIC: An SDK for next-Generation SD-RANs», in *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '21, New York, NY, USA: Association for Computing Machinery, 2021, pp. 411–425, ISBN: 9781450390989. DOI: 10.1145/3485983.3494870. [Online]. Available: https://doi.org/10.1145/3485983.3494870.

[50] O. N. Foundation, *ONF announces first release of SD-RAN v1.0 software platform for open ran*, Mar. 2021. [Online]. Available: https://opennetworking.org/news-and-events/press-releases/onf-announces-first-release-of-sd-ran-v1-0-software-platform-for-open-ran/.

[51] Open Networking Foundation, *ONF and Deutsche Telekom Demonstrate Fully Disaggregated Open RAN with Open RIC Platform*, 2021. [Online]. Available: https://opennetworking.org/news-and-events/press-releases/onf-and-deutsche-telekom-demonstrate-fully-disaggregated-open-ran-with-open-ric-platform/ (visited on 11/26/2022).

[52] Free5gc, «Roadmap of free5GC project», 2022. [Online]. Available: https://www.free5gc.org/roadmap/.

[53] Open5GS, «Open5GS Documentation», 2022. [Online]. Available: https://open5gs.org/open5gs/docs/.

[54] Free5GRAN, «Free5GRAN Documentation», *https://free5g.github.io/free5GRAN-documentation/*, 2022.

[55] UERANSIM, «Feature Set», *https://github.com/aligungr/UERANSIM/wiki/Feature-Set*, 2022.

[56] T. E. of Encyclopaedia, *Satellite applications*. [Online]. Available: https://www.britannica.com/technology/satellite-communication/Satellite-applications.

[57] B. R. Elbert, *The satellite communication ground segment and earth station handbook*. 2001, p. 370, ISBN: 158053046X.

[58] T. M. Braun and W. R. Braun, *Satellite Communications Payload and System*, ser. IEEE Press. Wiley, 2021, ISBN: 9781119384311. [Online]. Available: https://books.google.pt/books?id=Po02EAAAQBAJ.

[59] EUMETSAT, «What we do», 2022. [Online]. Available: https://www.eumetsat.int/about-us/what-we-do.

[60] Iridium, «Iridium Website», 2022. [Online]. Available: https://www.iridium.com/.

[61] Inmarsat, «Inmarsat Website», 2022. [Online]. Available: https://www.inmarsat.com/en/index.html.

[62] I. McKetta, *How Starlink's satellite internet stacks up against hughesnet and Viasat around the globe*, 2021. [Online]. Available: https://www.speedtest.net/insights/blog/starlink-hughesnet-viasat-performance-q2-2021/.

[63] Anacom, «Access to the internet via satellite is available and grew by more than 78.5% in Portugal», 2022. [Online]. Available: https://anacom.pt/render.jsp?contentId=1618921.

[64] T. S. Frontières, «Télécoms Sans Frontières Website», 2022. [Online]. Available: https://www.tsfi.org/en/.

[65] ——, «Our Solution», 2022. [Online]. Available: https://www.tsfi.org/en/our-solutions.

[66] ——, «Disaster Response - South-West Haiti earthquake», 2022. [Online]. Available: https://www.tsfi.org/en/our-missions/disaster-response/south-west-haiti-earthquake.

[67] J. Wattles, *SpaceX sent Starlink Internet Terminals to Ukraine. they could paint a 'giant target' on users' backs, experts say*, Mar. 2022. [Online]. Available: https://edition.cnn.com/2022/03/03/tech/spacex-starlink-ukraine-internet-security-risks-scn/index.html.

[68] SES, *Geo, Meo, and leo*, May 2021. [Online]. Available: https://www.satellitetoday.com/content-collection/ses-hub-geo-meo-and-leo/.

[69] E. Yaacoub and M.-S. Alouini, «A Key 6G Challenge and Opportunity—Connecting the Base of the Pyramid: A Survey on Rural Connectivity», *Proceedings of the IEEE*, vol. 108, no. 4, pp. 533–582, 2020. DOI: 10.1109/JPROC.2020.2976703.

[70] C. Cao and S. Zhai, «The influence of LEO satellite Doppler effect on LoRa modulation and its solution», in *Journal of Physics Conference Series*, ser. Journal of Physics Conference Series, vol. 1883, 2021, p. 12 071. DOI: 10.1088/1742-6596/1883/1/012071.

[71] A. Gaber, M. A. Elbahaay, A. Maher Mohamed, M. M. Zaki, A. Samir Abdo, and N. Abdelbaki, «5G and Satellite Network Convergence: Survey for Opportunities, Challenges and Enabler Technologies», *2nd Novel Intelligent and Leading Emerging Sciences Conference, NILES 2020*, pp. 366–373, 2020. DOI: 10.1109/NILES50944.2020.9257914.

[72] R. Ratasuk, *Non-Terrestrial Networks for 5G New Radio*, 2019. [Online]. Available: https://futurenetworks.ieee.org/images/files/pdf/FirstResponder/2019/Rapeepat-Ratasuk.pdf.

[73] A. Guidotti, S. Cioni, G. Colavolpe, M. Conti, T. Foggi, A. Mengali, G. Montorsi, A. Piemontese, and A. Vanelli-Coralli, «Architectures, standardisation, and procedures for 5G Satellite Communications: A survey», *Computer Networks*, vol. 183, p. 107 588, 2020, ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2020.107588. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S138912862031224X.

[74] M. Bacco, F. Davoli, G. Giambene, A. Gotta, M. Luglio, M. Marchese, F. Patrone, and C. Roseti, «Networking Challenges for Non-Terrestrial Networks Exploitation in 5G», in *2019 IEEE 2nd 5G World Forum (5GWF)*, 2019, pp. 623–628. DOI: 10.1109/5GWF.2019.8911669.

[75] A. Fornes-Leal, R. Gonzalez-Usach, C. E. Palau, M. Esteve, D. Lioprasitis, A. Priovolos, G. Gardikis, S. Pantazis, S. Costicoglou, A. Perentos, E. Hadjioannou, M. Georgiades, and A. Phinikarides, «Deployment of 5G Experiments on Underserved Areas using the Open5GENESIS Suite», in *2021 International Conference on Smart Applications, Communications and Networking (SmartNets)*, 2021, pp. 1–4. DOI: 10.1109/SmartNets50376.2021.9555428.

[76] H. Khalili, P. Sayyad Khodashenas, C. Fernandez, D. Guija, K. Liolis, C. Politis, G. Atkinson, J. Cahill, R. King, M. Kavanagh, B. T. Jou, and O. Vidal, «Benefits and Challenges of Software Defined Satellite-5G Communication», in *2019 15th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, 2019, pp. 1–4. DOI: 10.23919/WONS.2019.8795462.

[77] S. K. Sharma, S. Chatzinotas, and P. D. Arapoglou, *Satellite communications in the 5G Era.* 2018, pp. 1–570, ISBN: 9781785614279. DOI: 10.1049/PBTE079E.

[78] B. Evans, N. Wang, Y. Rahulan, S. Kumar, J. Cahill, M. Kavanagh, S. Watts, D.-K. Chau, Y. Begassat, A.-P. Brunel, T. Masson, and M. Diarra, «An integrated satellite–terrestrial 5G network and its use to demonstrate 5G use cases», *International Journal of Satellite Communications and Networking*, vol. 39, no. 4, pp. 358–379, 2021. DOI: https://doi.org/10.1002/sat.1393. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sat.1393.

[79] F. Michel, M. Trevisan, D. Giordano, and O. Bonaventure, «A First Look at Starlink Performance», in *Proceedings of the 22nd ACM Internet Measurement Conference*, ser. IMC '22, New York, NY, USA: Association for Computing Machinery, 2022, pp. 130–136, ISBN: 9781450392594. DOI: 10.1145/3517745.3561416. [Online]. Available: https://doi.org/10.1145/3517745.3561416.

[80] E. Lagunas, C. G. Tsinos, S. K. Sharma, and S. Chatzinotas, «5G Cellular and Fixed Satellite Service Spectrum Coexistence in C-Band», *IEEE Access*, vol. 8, pp. 72 078–72 094, 2020. DOI: 10.1109/ACCESS.2020.2985012.

[81] Quectel Wireless Solutions, *Quectel RM500Q-GL Specifications*, 2021.

[82] OSA, *NR SA CN5G GNB B210 COTS UE Tutorial.* [Online]. Available: https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/develop/doc/NR_SA_CN5G_gNB_B210_COTS_UE_Tutorial.md.

[83] Quectel Wireless Solutions, *RG50xQ&RM5xxQ Series - AT Commands Manual*, 2021.

Appendix A

This appendix includes the details of the OAI configuration used during this project, as well as the basic configuration.

Initial Setup

Both gNBs are executed in Ubuntu machines with a 12th Gen Intel(R) Core(TM) i9-12900K CPU and 64 GB of RAM. The core is run in a Ubuntu machine with an Intel(R) Core(TM) i5-4440 CPU with a total of 16GB of memory. All machines are using a low-latency kernel (4.15.0-192-lowlatency), which is required for OAI. Each gNB had a USRPs B210 connected through USB.

In the deployment, the machines used were all connected to the same LAN network. The only exception to this rule was that one of the machines used for the gNBs had a direct wired connection to the core. This was to allow experimenting with the tc commands without the risk of losing SSH access to the machines.

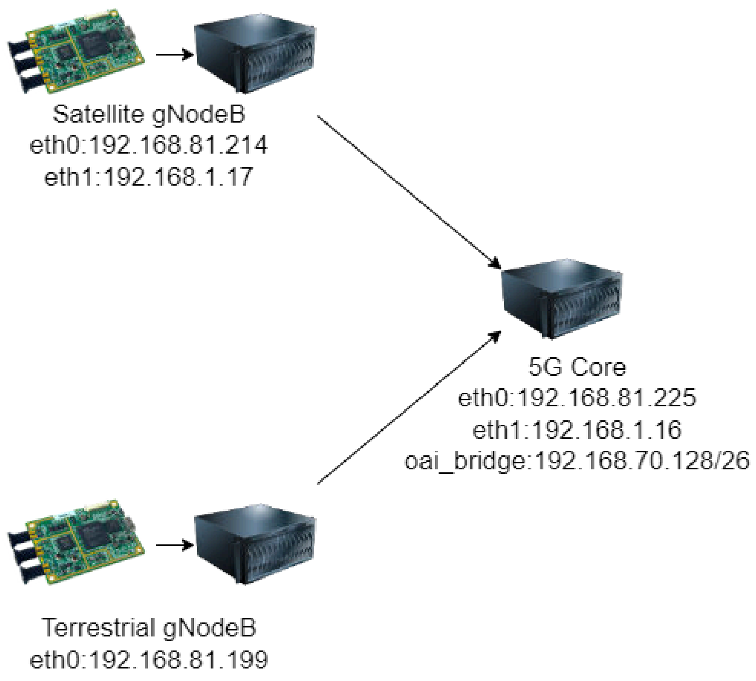


Figure A.1: Example network configuration diagram.

An example host network configuration may be seen in Fig. A.1. The arrows are meant to represent the routes defined in each gNB to direct traffic to the core, accessible through the oai_bridge.

Core Setup

The first step to setting up the core host is to download the Docker Engine. When using Linux hosts, this can be achieved with Code A.1.

```
curl -fsSL get.docker.com -o get-docker.sh
sudo sh get-docker.sh
```

Code A.1: Installing Docker

Afterward, the docker images may be retrieved [1].

With this, it is already possible to execute the docker-compose files made available in the OAI repository *oai-cn5g-fed*. However, to guarantee that end-to-end connections will be possible, it is necessary to enable port forwarding [2].

It is also necessary to configure Mobile Country Code (MCC) and Mobile Network Code (MNC) to correspond to the ones later configured on the gNBs and on the SIM card. Additionally, the registration data of the UE also needs to be added to the database of the core. One way to do this is represented in Code A.2.

```
# Pull git repository
git pull https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed.git

cd oai-cn5g-fed/docker-compose/

# Configure PLMN
sed -i '/^[[:space:]]*- .*MNC.*=/ s/=.*/=99/' docker-compose-basic-nrf.yaml
sed -i '/^[[:space:]]*- .*MCC.*=/ s/=.*/=208/' docker-compose-basic-nrf.yaml

cd database/

# Add SIM info to database
sed -i "250i \(\'208990000000001\', \'5G_AKA\', \'
    fec86ba6eb707ed08905757b1bb44b8f\', \'fec86ba6eb707ed08905757b1bb44b8f\',
    \'\{\\\\\"sqn\\\\\": \\\\\"000000000020\\\\\", \\\\\"sqnScheme\\\\\":
    \\\\\"NON_TIME_BASED\\\\\", \\\\\"lastIndexes\\\\\": \{\\\\\"ausf\\\\\":
    0\}\}\', \'8000\', \'milenage\', \'C42449363BBAD02B66D16BC975D77CC1\',
    NULL, NULL, NULL, NULL, \'208990000000001\'\)\," oai_db2.sql
```

Code A.2: Core downloading and configuration

The first sed commands edit all occurrences of MCC and MNC to correspond to the value given, in this case, 208 and 99. The last sed command adds the registration data to the core database, allowing the UE to connect later.

[1] See: `https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed/-/blob/master/docs/RETRIEVE_OFFICIAL_IMAGES.md`

[2] See: `https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed/-/blob/master/docs/DEPLOY_SA5G_BASIC_DEPLOYMENT.md`

If necessary, it is possible to change the Access Point Name of the network by changing DNN_NI0 and DNN_NI1 on the configuration of the SMF.

After following these steps, the core host is ready to be used by the network launcher tool developed. Alternatively, it can be launched manually by executing the command in Code A.3.

```
cd ~/oai-cn5g-fed/docker-compose/ && python3 core-network.py --type start-
    basic --scenario 1
```

Code A.3: Launch core network

gNB Setup

To configure the gNB hosts it is necessary to pull both the OAI main repository, which includes the gNB source and also the *eXtend5G-testbench* repository [3] that includes the configuration files and the propagation delay patch.

In the terrestrial host, OAI can be compiled as is, while in the satellite host it is necessary to patch the code before compiling, as represented in Code A.4.

```
# fetch repositories
git pull https://gitlab.eurecom.fr/oai/openairinterface5g.git
git pull https://github.com/JPCatarino/eXtend5g-testbench.git

cd openairinterface5g/

git checkout 2022.w30

# Apply patch
cp ~/extend5g-testbench/oai/gnb/files_with_delay/usrp_lib.cpp /targets/ARCH/
    USRP/USERSPACE/LIB/usrp_lib.cpp
cp ~/extend5g-testbench/oai/gnb/files_with_delay/softmodem-common.h /
    executables/softmodem-common.h
cp ~/extend5g-testbench/oai/gnb/files_with_delay/main-ocp.c /executables/main
    -ocp.c
cp ~/extend5g-testbench/oai/gnb/files_with_delay/common_lib.h targets/ARCH/
    COMMON/common_lib.h

# Install dependencies and compile gNB to be used with USRP
source oaienv
cd cmake_targets/
./build_oai -I -w USRP --gNB
```

Code A.4: Satellite host OAI installation

Before running the compiled gNBs, to have a connection to the core network it is necessary to set up a route. If using the addresses and interfaces of Fig. A.1, the command in Code A.5 could be executed to achieve this.

[3]Private, at the time of writing.

```
sudo ip route add 192.168.70.128/26 via 192.168.1.16 dev eth1
```

Code A.5: Satellite host IP route to core

Afterward, the hosts are ready to use the launcher program developed.

Alternatively, they may be executed manually using the command in Code A.6. To activate the propagation delay on the fronthaul, in case of simulating satellite, the flags *delay_rx* and *delay_tx* must be set.

```
cd ~/openairinterface5g/cmake_targets/ran_build/build/ && sudo -S ./nr-
    softmodem -O CONF_FILE --gNBs.[0].min_rxtxtime 6 -E --sa --continuous-tx
```

Code A.6: Run OAI gNB

In the configuration files, the addresses to AMF must be changed to correspond to the ones of the hosts used. Additionally, the MCC and MNC must be the same used on the core network. For the deployment used, both gNB were deployed in different bands (n41 and n78). When deploying two gNB it is important to check if their frequencies don't overlap, as this will cause them to malfunction.

Appendix B

This appendix serves as a quickstart guide to prepare the UE to use the firmware developed. The steps presented were made to run in a Raspberry Pi 4 Model B, connected to a Quectel RMU500EK Development Kit, but some may apply to other configurations.

Prerequisites

SIM Configuration

The UE is only able to register in the core network if the SIM card details correspond to a subscription data entry in the UDR. In this project, a OpenCells [1] SIM card was used, configured with the software they provide.

Quectel Drivers

To make the Raspberry Pi able to interface with the Quectel modem, it is necessary to install the device drivers. The driver documents provided by Quectel suggest patching the kernel USB drivers to support the devices of the supplier.

However, by examining kernel lore [2], it was found that the patches required were already merged in more recent versions of the Linux kernel. At the time of writing, these changes are not present in the latest stable release of the Raspberry Pi OS kernel, but, by downloading the bleeding edge kernel, the Raspberry will support Quectel devices without the need for further configuration.

This can be achieved by executing the *rpi-update* command, which will both update the firmware and the kernel of the Raspberry Pi.

5G Modem Configuration

Before using the solution developed, some of the 5G modem parameters must be configured. These parameters include the Access Point Name (APN), which needs to be changed to the one configured on the SMF. Moreover, the modem should be set to only scan for 5G cells. If using the solution developed, it should be set to only connect to satellite or terrestrial bands at the beginning.

[1] See `https://open-cells.com/index.php/sim-cards/`

[2] See: `https://lore.kernel.org/linux-arm-kernel/YrRVzHktW3oelFaA@hovoldconsulting.com/t/`

Additionally, in this research, it was observed that setting the RRC release version to R99 caused fewer issues when connecting to the OAI network, so that also should be configured.

An example of how to configure the modem can be seen in Code B.1

```
# Open connection to modem
sudo socat - /dev/ttyUSB2,crnl,b115200

# Disable automatic MBN files
AT+QMBNCFG="Select","ROW_Commercial"
AT+QMBNCFG="AutoSel",0
# Reset Modem
AT+CFUN=1,1
# Set APN
AT+CGDCONT=1,"IP","oai"
# Set modem to 5G only and to only scan band 41
AT+QNWPREFCFG="mode_pref",NR5G
AT+QNWPREFCFG="nr5g_band",41
# Set RRC version to R99
AT+QCFG="rrc",0
```

Code B.1: Modem Initial Configuration

As an alternative, the latter part of this configuration can be executed using the Python wrapper developed, as shown in Code B.2.

```python
import serial
import serial.threaded
import RMU500EK.RMU500EK as modem

ser = serial.Serial('/dev/ttyUSB2', baudrate=115200, timeout=180)
with serial.threaded.ReaderThread(ser, modem.RMU500EK) as m5g_module:
    # Set APN
    m5g_module.set_pdp_contexts(1, "IP", "oai")
    # Set modem to 5G only and to only scan band 41
    m5g_module.set_network_search_mode(["NR5G"])
    m5g_module.set_5g_nrsa_band(["41"])
    # Set RRC version to R99
    m5g_module.set_rrc_version("0")
```

Code B.2: Modem Initial Configuration in Python

Quectel Connection Manager Installation

The Quectel Connection Manager (*quectel-cm*) is a software tool[3] that helps establish data connections. It also creates a USB network interface using one of the drivers available so that the device may send data packets through the modem.

Since the firmware developed makes use of this interface, *quectel-cm* must be installed on the host device.

[3]This tool is exclusively for Quectel clients and has to be requested through their contacts.

Before installing this tool, the ModemManager that comes loaded on RaspberryPi OS must be uninstalled, as both tools conflict with each other.

Afterward, *quectel-cm* can be compiled and executed, as in Code B.3.

```
# Remove confliting software
sudo apt remove modemmanager

cd quectel-CM/
# Compile and execute
make
sudo ./quectel-CM
```

Code B.3: Install and execute *quectel-cm*

Firmware Installation

After completing all previous steps, the solution developed is ready to be installed and used. To do so, it is only necessary to pull the *extend5g-ue* repository [4], install the python requirements, and run the main file, as depicted in Code B.4.

```
# Pull repository
git pull https://github.com/JPCatarino/eXtend5g-ue.git

cd eXtend5g-ue/
# Install application requirements
pip install -r requirements.txt

# Run firmware with default parameters and debug mode active
python main.py -p /dev/ttyUSB2 -i wwan0 -d
```

Code B.4: Install and execute the firmware developed

The -p parameter sets the serial port that is used to send AT commands to the firmware, while the -i parameter defines the USB network interface that the device uses to send packets.

[4]Private, at the time of writing