

Julius-Maximilians-Universität Würzburg

Chair of Computer Science VII - Robotics and Telematics



Master Thesis

Evaluation of Concepts for gNodeB Satellite Backhaul using Open-Source 5G Frameworks

Aitzol Monroy

Submission: June 30, 2022

Supervisor and Reviewer

Prof. Dr.-Ing. Markus Gardill

Second Reviewer

Prof. Dr. Guido Dietl

Abstract

There has been a tremendous increase in mobile networks in recent years. From connecting a few devices with low requirements, to the requirement of connecting many more devices with increasing technical needs. This has resulted in the urgency to increase network capacity to meet the needs of all users, increase connection speeds and decrease latency.

In many cases, the demand for 5G capacity is very difficult to achieve with conventional infrastructure, especially in remote areas where access is not easy. This is where the concept of using satellites to improve communications networks comes into play. For years, satellite communications have been separated from telecommunications networks. Today, with the new generation of satellites, built with 5G architecture in mind, they can be integrated into these telecommunications networks to extend the coverage of cars, planes or any device in remote areas.

Integrating these satellites into the mobile network, routing the backhaul of a 5G gNB over a satellite is a possible option, which will be analysed in this work.

In this master thesis, concepts for the satellite backhaul are implemented and evaluated using open-source 5G frameworks, looking into how the gNB and Core Network (CN) elements interact, and respond to changes and limitations in the network. To do so, a 5G network has been created using what Open Air Interface (OAI) provides.

Contents

1. Introduction.....	1
2. Technical Basics.....	3
2.1. 5G Architecture:.....	3
2.1.1. 5G Core.....	3
2.1.2. NG-RAN	5
2.1.3. Backhaul	6
2.1.4. Network Interfaces.....	7
2.1.5. Control and User Plane Separation	8
2.2. 5G Protocol Stack	9
2.3. 5G Deployment Scenarios	11
2.3.1. Enhanced Mobile Broadband.....	11
2.3.2. Ultra-Reliable Low Latency Communications	11
2.3.3. Massive Machine Type Communications.....	11
2.4. 5G Network Functions Deployment in Slices	12
2.5. Satellite Backhaul Technology.....	13
2.6. Multi-access Edge Computing.....	14
2.7. 5G Frequency Bands and Frame Structure	15
2.8. Open Source 5G Frameworks.....	16
2.8.1. srsRAN	16
2.8.2. OAI.....	17
3. Implementation.....	19
3.1. Retrieving the Images on Docker	19
3.2. Deploy Containers	19
3.2.1. Deploy OAI 5G Core Network.....	20
3.2.2. Deploy OAI gNB in RF Simulator Mode and in SA Mode.....	20
3.2.3. Deploy OAI NR-UE in RF Simulator Mode and in SA Mode	21
3.3. Traffic Check.....	22
3.4. Explanation on the Configuration in the Docker-compose.yaml	24
3.4.1. Content of the Blocks	24
3.4.2. Configuration of the Networks.....	28
4. Results	30
4.1. Evaluation Tools	30
4.2. Test.....	31
4.2.1. Test Without Restrictions.....	31
4.2.2. Test with Restrictions	44

5. Conclusion	54
5.1. Result Discussion	54
5.2. Future Work	54
List of Figures	56
List of Tables.....	58
Acronyms.....	59
Bibliography	62
Appendix	65
Appendix 1	65
Appendix 2	71
Appendix 3	72

1. Introduction

Mobile communications have played and continue to play a major role in all areas of society. Mobile communications are one of the main means to make the economy more dynamic and efficient.

To analyse the evolution of mobile telephony, one would have to go back to 1973, when the first mobile phone call was made. It was in 1979 that the first 1G network was launched in Japan, the first country to have 1G service. Within a few years it was extended to the whole world. This generation was launched to provide voice call which was analog signals. This generation of telephony had no security whatsoever, as the calls were not encrypted [1].

2G was born in Finland in 1992, providing many advantages compared to 1G. 2G brought smaller and cheaper mobile phones and the ability to send SMS and MMS messages. Transmission was switched from analogue to digital and calls were encrypted. Time and code multiplexing made better use of channels, with more simultaneous users supported by the network. Voice quality was clearer and noise was reduced. Circuit switching is still used in this technology.

Before moving to 3G there were other improvements such as 2.5G, which introduces packet switching for data communication, allowing for higher network performance as each packet within the network is treated individually and can follow any path it chooses.

In 2001, 3G technology appeared, with the aim of increasing transmission capacity to offer services such as mobile internet connection and file downloading. It is called UMTS standard which allows speeds up to 6Mb/s. 3G allows international roaming services to be introduced. What was really revolutionary about 3G was the ability to surf the internet.

Launched in 2009, 4G is the technological evolution that offers faster internet and higher bandwidth. 4G is an IP-based generation, which includes techniques such as Multipol Input Multiple Output (MIMO) and Orthogonal Frequency Division Multiplexing (OFDM). Today the average Down Link (DL) speed for 4G is 50 Mbps.

Due to the increase in the number of devices connected to the internet, and the need to increase speeds and reduce latencies, 5G has emerged.

5G comes with the promise of breaking the 3 Gbps barrier and reducing latency to under 1ms. 5G uses more radio spectrum than any other technology, and that allows for more users, more data and faster connections. South Korea was the first country to offer 5G in 2019.

As for the actual speed of 5G networks, it will depend on the location. In the UK, maximum speeds of between 247 Mbps and 753 Mbps have been reported by the major carriers In terms of latency, an average of 30ms has been reported [2]. 5G is essential for connecting smart cities, self-driving cars and other industries.

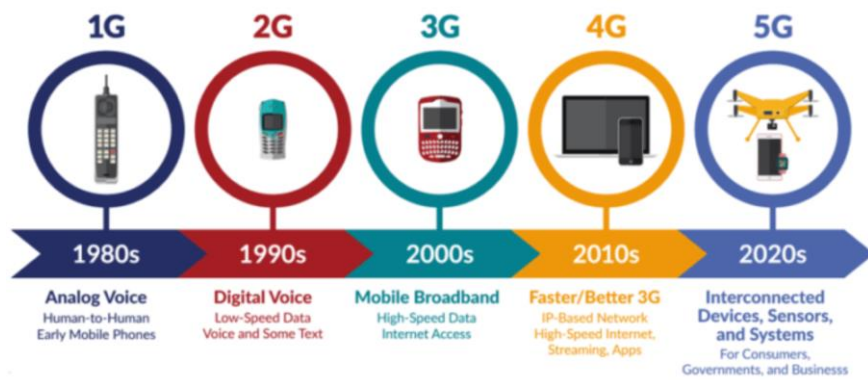


Figure 1: Evolution of the cellular network [3]

Work on 6G technology has been underway since 2019 and is expected to be commercialised in 2030, with the first real-world use cases expected from 2026. Possible uses include extended reality (XR), Artificial Intelligence (AI) and energy efficiency. Speed could be 1Tbps and 0.1ms latency.

One of the important elements of 5G is the backhaul, which is the connection of the CN to the subnetworks that connect to it. One of the challenges of mobile technologies is reaching any remote area, and this is where the term "Satellite Backhaul" comes into play. This technology makes it possible to reach places where a terrestrial infrastructure is not possible routing the communication from the Next Generation Radio Access Network (NG-RAN) to the satellite and from the satellite to the CN.

In this master thesis, concepts for the satellite backhaul are implemented and evaluated using open-source 5G frameworks. For this purpose, a 5G network will be implemented in a software framework.

2. Technical Basics

This section will introduce all the technical aspects necessary for the correct understanding of the thesis. Starting from the basis that a 5G network has been implemented and evaluated in software.

2.1. 5G Architecture:

As in previous versions of 3rd generation Partnership Project (3GPP), 3GPP defined for 5G the CN and the New Radio (NR) access technology called 5G NR. In previous generations it was necessary for both the Radio Access Network (RAN) and the CN to be deployed from the same generation. With 5G it is possible to integrate elements from different generations in different configurations [4].

On the one hand there is the Standalone (SA) configuration, which uses the 5G RAN and the 5G CN, and on the other hand there is the Non-Standalone (NSA) configuration, which combines 4G CN and 5G RAN. In the case of NSA, operators take advantage of the existing core network. Both of the cases are illustrated in Figure 2.

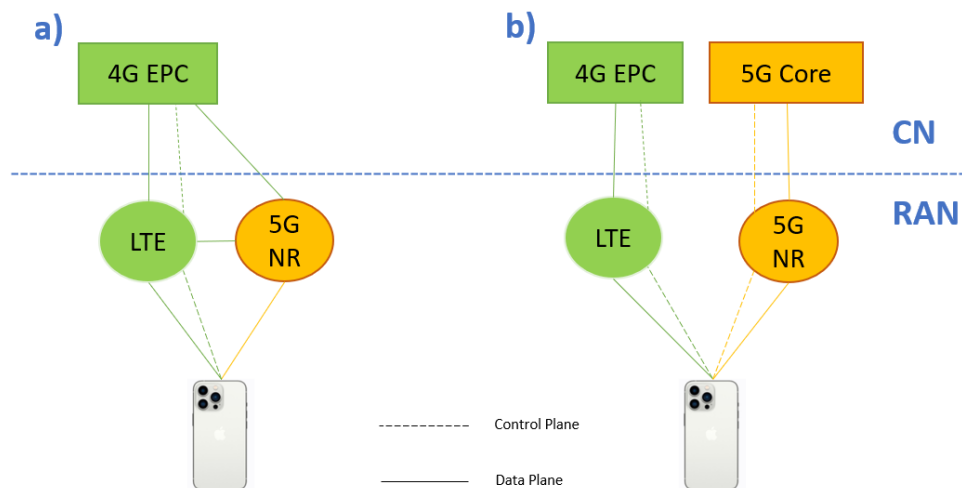


Figure 2: 5G a) non-standalone and b) standalone configuration

2.1.1. 5G Core

The 5G core network architecture is the basis of the new 5G technology specifications and enables the increased capacity demands that 5G technology must guarantee. The new 5G network core, as defined in 3GPP [5] is Service Based Architecture (SBA), which provide efficiency for the network. Compared to the previous generation of core like Evolved Packet Core (EPC) network, the elements of this new network are defined to be Network Functions (NF) which will interact with the rest of the network elements via Application Programming Interface (API) invocation. The Network Repository Functions (NRF) block allows network functions to discover the services offered by other network functions. A service is a capability offered to a 5G network, with the characteristics of

high cohesion, coupling capability and management independent of other services. This makes each NF to be handled independently without affecting other services.

With the SBA architecture the network becomes very efficient by deploying new features on it, allowing new services to be implemented easily or even bugs to be fixed easily.

This section analyses the case for 5G SA core network, which is illustrated in Figure 3.

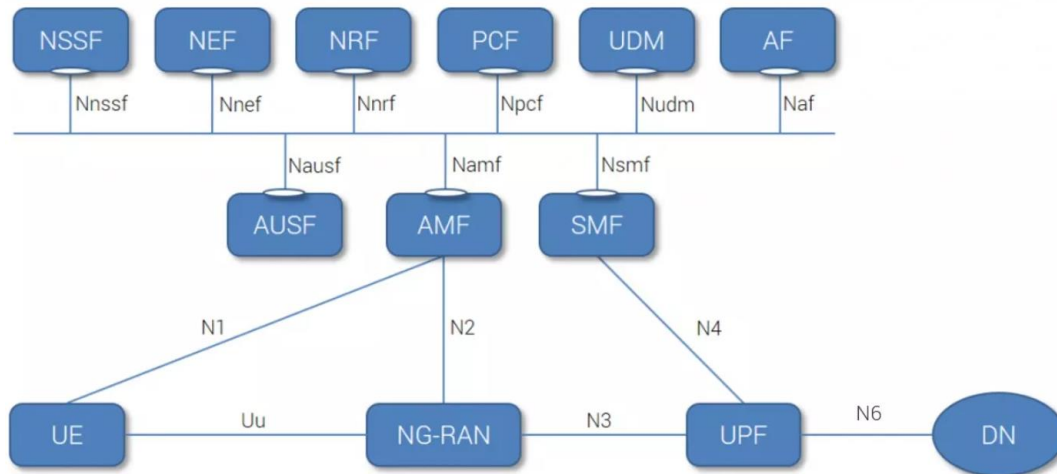


Figure 3: 5G architecture divided into several functional blocks with their interfaces [6]

Functional blocks:

- **AMF:** Access and Mobility management Function (AMF), Interacts with the RAN and the User Equipment (UE). Receives session and connection data from the UE through the N1 and N2 interfaces, but is responsible only for handling connection and mobility management tasks. It allows devices to register, authenticate and move among the network jails. It handles Non Access Stratum (NAS) protocol ciphering and integrity protection.
- **UPF:** The User Plane Function (UPF) is a very important block within the 5G CN that enables such low-latency 'edge' computing, as it can be placed close to where the service is provided, reducing the distance to the service provider. The UPF is responsible for packet routing and forwarding, packet inspection, Quality of Service (QoS) handling, and external Protocol Data Unit (PDU) session for interconnecting Data Network (DN) in the User Plane (UP) [7]. It uses N3 to connect with gNB; N6 to connect with de DN, which could be the internet connection; N9 to connect with other UPF; N4 to connect with the Session Management Function (SMF).

An important concept within the UPF is the PDU session, which provides connectivity between the UE and the DN of different QoS, depending on the service required or contracted.

- **SMF:** The SMF is primarily responsible for handling session management, such as setting up sessions, modifying them, or even releasing them. Performs DHCP protocol and ARP proxying for the PDUs and many other functionalities. It is connected to the UPF through N4 interface.

- **AUSF:** Authentication Server Function (AUSF) is responsible for the security procedure for UE authentication using the 5G-AKA authentication method, which includes mutual authentication between the device to be connected to the network and the network itself.

Security is very important for the success of any technology, with the use of this block network spoofing is tried to avoid, where a fake base station can broadcast a different tracking area code and UE's can connect to it. On the other hand, it also avoids "lack of confidentiality" concept, which could detect the presence of a specific user by intercepting certain OTA signalling messages.

- **UDM:** The Unified Data Management (UDM), which is similar to 4G's Home Subscriber Service (HSS), manages data for access authorization, user registration, roaming access and data network profiles. It selects the authentication method based on the subscriber identity and stores the long-terms credentials used in AKA authentication.
- **NRF:** NRF is a based block of the 5G SBA, as it works as a centralized repository for all the 5G NFs in the operator's network. The NRF allows 5G NFs to register, get notified about the registration of another NF and discover each other via a standards-based API. So in this case, when a new NF is added, it is only necessary to give it the IP address of the NRF so that it can communicate with it, and this NRF block will let all the other blocks know about it.

The different network entities are connected by an underlying TCP/IP transport network, which supports diff-serv QoS.

Thanks to Network Function Virtualization (NFV) technologies, the mobile network functions can be virtualised and hosted in the cloud. This makes the 5G CN flexible enough to meet the needs of different services [4].

2.1.2. NG-RAN

NG-RAN is a part of the 3GPP 5G system [8], it represents the radio access network for 5G. This node could be a gNB for the case of 5G or a NG-eNB, which provides a LTE/E-UTRAN service. The NG-RAN can be divided into access stratum, which is for dialogue between mobile equipment and the RAN, and non-access stratum, which is for dialogue between mobile equipment and the CN.

With regard to the Centralized Unit (CU)- Distributed Unit (DU) split [9] the following applies:

The NG-RAN enables splitting the gNB into a gNB-CU , gNB-DU or gNB-RadioUnit (RU). As it can be seen in the picture below, the gNB-CU and the gNB-DU are connected by the F1 interface. A gNB may consist on a gNB-CU and one or more gNB-DUs. As can be seen in Figure 4, the gNBs are connected to each other through the Xn interface. The NG interface is used to connect the gNB to the 5G CN. Being NG-U to connect to the UPF block and NG-C to connect to the AMF block. Figure 4 shows the relationships between blocks and interfaces.

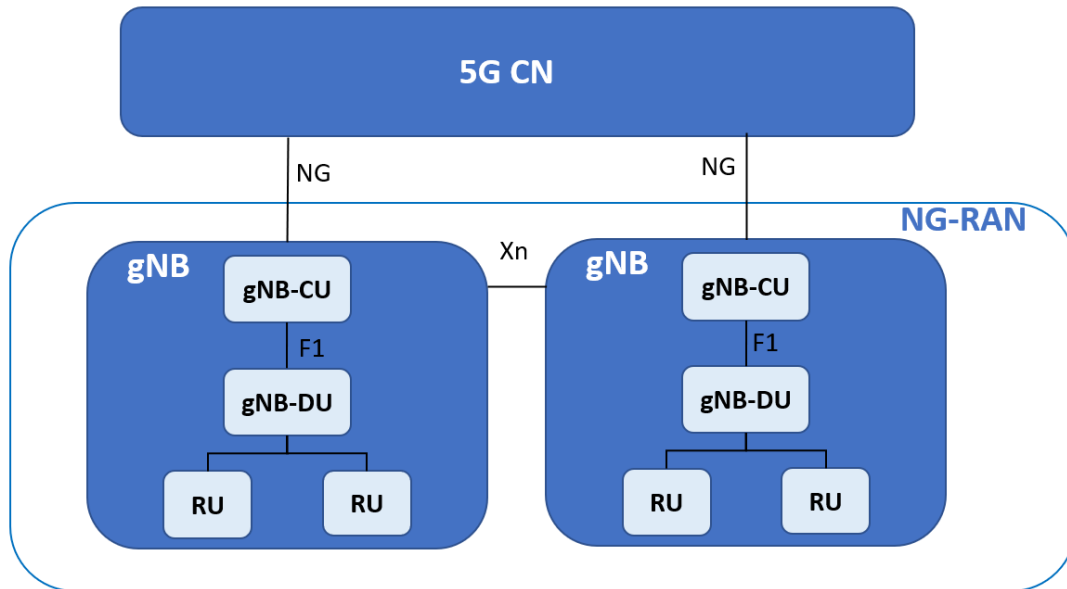


Figure 4: NG-RAN architecture

RU: This logical separation includes the hardware device that receives or transmits the radio signals from the antenna. It is responsible for converting the physical signal to digital for proper signal processing. The Layer 1 functions includes the digital front end (DFE) and the lower PHY layer. When designing the RU it is very important to calculate the size, weight and power consumption of the devices [10].

DU: This logic block can be deployed on a Commercial off-the-shelf (COTS) server, and can be placed close to the RU to reduce communication time. It runs the Radio Link Control (RLC), Medium Access Control (MAC), and parts of the PHY layer. This logical division will be responsible for a subset of the gNB's functions, depending on the functional division being made. Its operation is controlled by the CU

CU: It provides support for the Radio Resource Control (RRC), Packet Data Convergence Protocol (PDCP) and Service Data Adaptation Protocol (SDAP) layers. There is a single CU for each gNB, but there can be many DUs connected to a single CU.

By being able to split the architecture, this makes it possible to combine different distributions between the CU and DU depending on the availability of the intermediate network and the network design, required latency and backhaul bandwidth requirements. As explained above, this split architecture, which is fully virtualised using COTS or other hardware devices, can be deployed in any data centre within the RAN.

2.1.3. Backhaul

The backhaul of a network consists of connecting the CN to RANs. 5G backhaul refers to the signal between the 5G CN and remote sites or networks, such as RAN. The backhaul in the case of 5G networks has to support a large number of devices and data rates, so a correct design is important for the proper functioning of the network.

The connection between the 5G CN and the CU is called backhaul, which implements the NG interface explained before. The link between the CU and the DU is referred to as midhaul. Finally, the transport network between the DU and RU is known as fronthaul.

A distinction can be made between wired and wireless backhaul.

Wired backhaul

Wired backhaul would require the use of fibre optics for fast use, but this implies an expensive and not easy to install solution, as depending on the location or distances it may not be feasible. In this case, speeds of 10 Gbps and latencies of 0.1 ms can be achieved.

Wireless backhaul

Wireless backhaul makes it unnecessary to use cables or underground connections and makes installation easier. The use of microwave and millimetre waves makes it easy to deploy [11]. The only disadvantage of wireless backhaul is that the system can be interrupted by weather. As for the range of the backhaul, it will depend on the frequency bandwidth.

Wireless backhaul includes satellite backhaul, which facilitates the deployment of services in remote locations. A satellite backhaul is used between the CN and the RAN and is used to transport the N2 for control and N3 for data interfaces as shown in Figure 5. Among the different uses are the ability to transmit any kind of data, whether it is HDTV or video. It could also be used in an IoT device scenario.

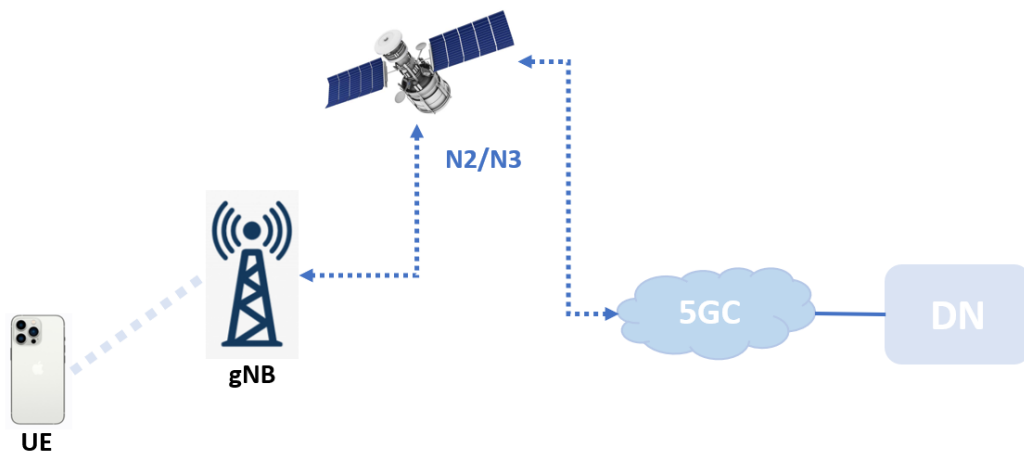


Figure 5: Satellite backhaul

2.1.4. Network Interfaces

This section describes the main network interfaces Uu, N1, N2, N3, N4 and N6, shown in Figure 3 and 4. However, it should be noted that the following are also found: F1 for communication between gNB-DU and gNB-CU, Xn for communication between gNB's. In the CN there are also N5, for the communication between PCF and AF; N7, for the communication between SMF and PCF; N8, for the communication between AMF and UDM; N9, for the communication between UPFs; N10, for the communication between

SMF and UDM; N11, for the communication between AMF and SMF; N12, for the communication between AMF and AUSF; N13, for the communication between AUSF and UDM; N14 for the communication between AMFs and finally N15 for the communication between AMF and PCF.

Uu: This interface connects the gNB and the NR-UE over the air [12].

N1: This transparent interface connects the UE and the AMF, which handles the security NAS messages.

N2: This is a very important interface, it connects gNB and the AMF. Due to Control and User Plane Separation (CUPS), before a service can be accessed, the UE must be connected to the network. This interface is used for all control plane signaling.

N3: This interface connects the NG-RAN and the UPF. In this case, this interface carries user information. It replaces S1-U interface from the 4G EPC and is key to supporting CUPS architecture. The distance of this block from the NG-RAN block depends on the latency and throughput required by the application or service to be provided. The GTP-U protocol is used. In the Figure 6 can be seen the protocol stack, the communication between them and what a user plane communication would look like.

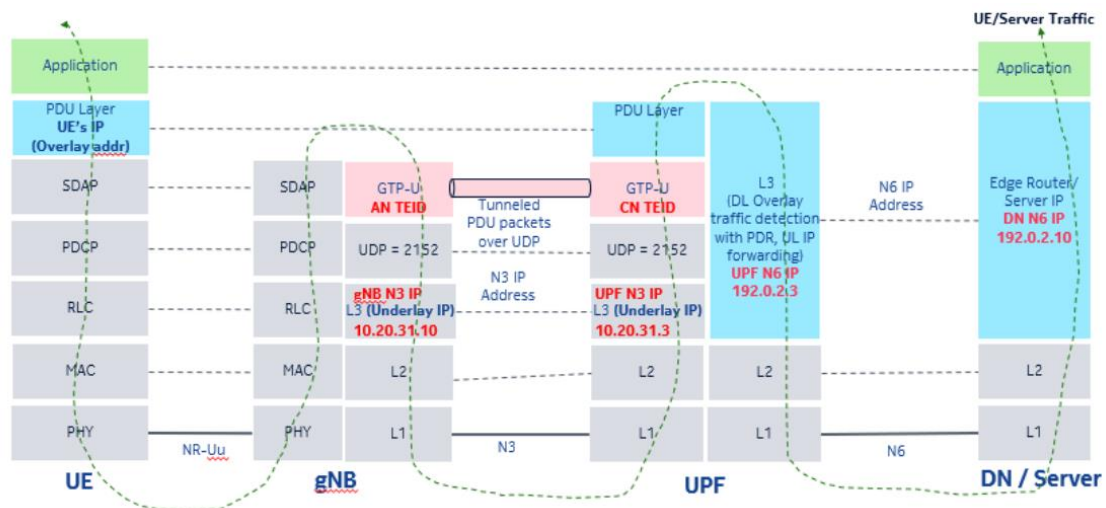


Figure 6: 5G Data Path Protocol Stack [13]

N4: This interface connects the SMF and UPF, it is the bridge between the control and user plane in the core network. This connection is responsible for the QoS and Quality of Experience (QoE) that is required. Through this interface goes the traffic of the PDU session management and traffic steering towards the UPF and PDU usage and event reporting towards the SMF. It receives the policy rules obtained from the Policy Control Function (PCF) regarding packet handling and forwarding [14].

N6: This interface provides connectivity between the UPF and DN, any internal or external network such as clouds or internet.

2.1.5. Control and User Plane Separation

Due to the increase in mobile data traffic, cellular RANs relying on CUPS have been introduced in 3GPP Release 14. In 4G EPC, Charging Data Record (CDR) can be

generated by both Serving Gateway (SGW) and Packet Data Network Gateway (PGW) where Control Plane (CP), that is related to messages of the creation of the Packet Data Network (PDN) sessions and the UP messages that represent the actual data are transferred. With CUPS architecture, the CP messages will be handled by the CP node (GW-C) and UP messages will be handled by the UP node (GW-U).

In 5G the UPF will be responsible for handling UP traffic, while the SMF is in charge of the CP. This architecture saves costs and allows the UPF to be brought closer to the user to reduce service latency. Bringing the UPF closer to the user is essential for applications such as the connected car, where low latency is required [15] [16].

Among the advantages of using the CUPS architecture are the following:

- Latency in the application service is reduced by using user plane nodes that are closer to the user or more appropriate for the type of QoE required. The clearest example would be the use of high bandwidth applications such as video. As the user plane node is close to the end user, the operator will not need to send the user's request all the way to the central hub.
- With high data traffic demand, it will be sufficient to increase the number of user plane nodes without having to increase the number of control plane nodes.
- The scaling and evolution of the resources of control and user plane are done independently.

2.2. 5G Protocol Stack

When analysing the protocol stack, it is worth mentioning that there is a difference between the control plane illustrated in Figure 7 and the user plane in Figure 8, which will be explained below. In this first part, the 5G NR-Uu protocol stack will be analysed [17] [18].

PHY

Contains digital and analogue signal processing functions, that the gNB and UE use to send and receive information. It has Frequency Division Duplex (FDD) and Time Division Duplex (TDD) configurations. Based on Orthogonal Frequency Division Multiple Access (OFDMA), Sub Carrier Spacing (SCS) could be 15,30,60,120,240 kHz and uses an adaptative modulation scheme, from Binary Phase-Shift Keying (BPSK) to 256 Quadrature Amplitude Modulation (QAM).

MAC

This layer receives both user plane and control plane data from the RLC layer and it creates logical channels to this protocol. It also provides low level control of the physical layer and data transfer and radio resource allocation. It does error correction through Hybrid Automatic Repeat Request (HARQ). It manages the priority between UEs.

RLC

RLC protocol ensures reliable delivery of data streams to upper layers with error control and correction that need to arrive intact. It also handles segmentation and duplication

detection. It has three different modes, Transparent, Unacknowledge and Acknowledge mode.

PDCP

It is a Layer 2 protocol, which carries out higher level transport functions related to integrity protection and ciphering procedures. It provides to upper layers transfer of user and control plane data. It also does in case of need reordering and in-order delivery.

SDAP

The Service Data Adaptation Protocol (SDAP) Maps the interaction between the Packet of a QoS flow and data radio bearer by marking the user data packets properly. It does the transfer of user plane data. It marks each packet with the corresponding QoS Flow ID.

RRC

Is the signalling protocol used in access stratum procedures involving the mobile and the gNB. RRC can configure PDCP, RLC and MAC. It is the responsible for the connection establishment in the N3 interface with GTP-U protocol which is used for the tunnel creation for N3 interface. It is also the interface with the NGAP protocol for the interaction with the AMF through N2 interface.

NAS

NAS transfers messages between the UE and the AMF for PDU session management, security, mobility management, authentication and registration request [19].

Figure 7 shows the protocol stack for the control plane. As can be seen, the N1 interface is used to exchange NAS messages between the UE and the AMF block. As for the N2 interface, the NG-AP protocol is used for operations such as updating configurations and managing PDU session resources.

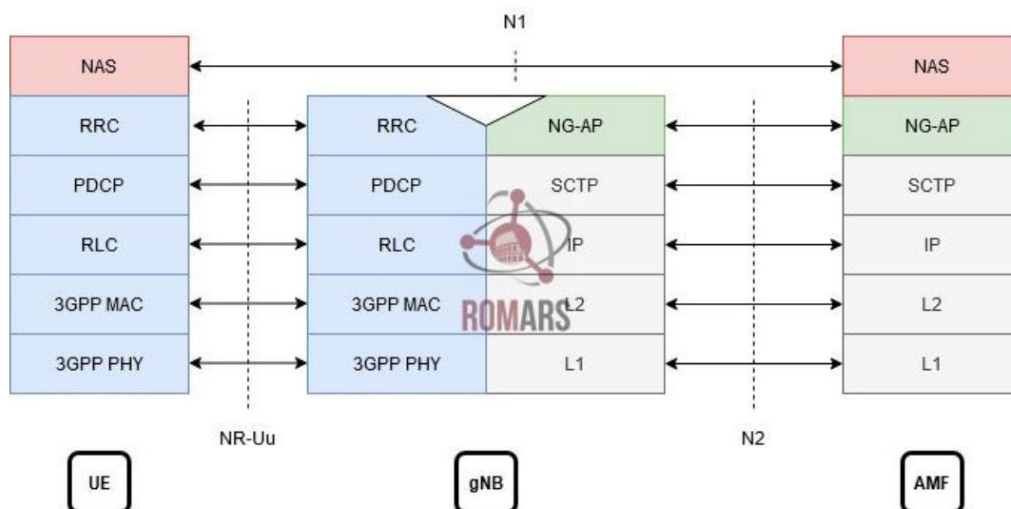


Figure 7: Control Plane protocol stack [20]

Although Figure 8 is essentially identical to Figure 6, the illustration of the user-plane protocol stack is included here again for the direct comparison to the control-plane

protocol stack. In this case, a PDU session is illustrated. The GTP-U protocol supports multiplexing of the traffic from different PDU sessions by tunnelling user data over N3 interface [21].

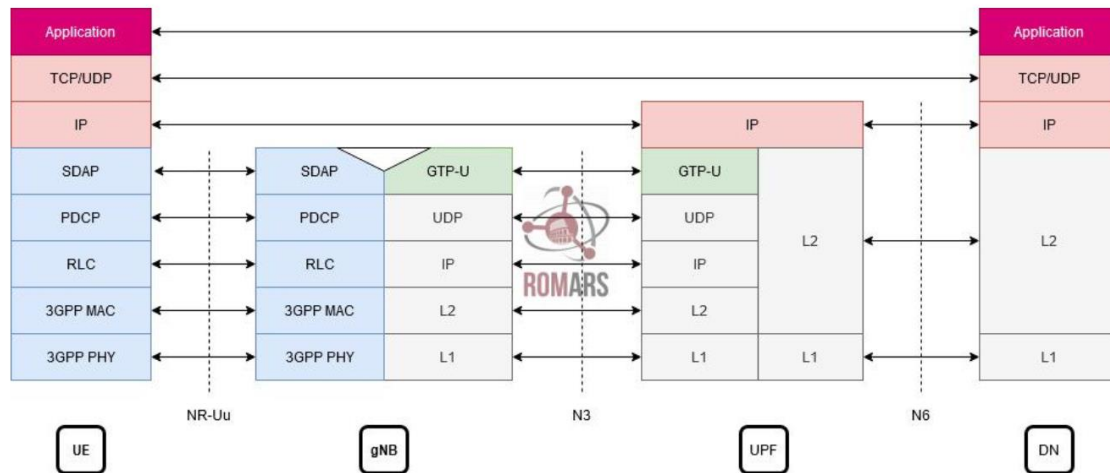


Figure 8: User Plane protocol stack [20]

2.3. 5G Deployment Scenarios

In this section, the possible 5G deployment scenarios will be explained. The three main 5G operating scenarios are Enhanced Mobile Broadband (eMBB), Massive Machine Type Communications (mMTC) and Ultra-Reliable Low Latency Communications (URLLC) [22].

2.3.1. Enhanced Mobile Broadband

eMBB is a natural evolution to existing 4G networks, it is characterised by providing broadband access over a wide coverage area that include faster data rates and better QoE than 4G mobile broadband services. To increase the capacity, broadband access should be available in densely populated areas, such as city centres or stadiums. Regarding the connectivity, it must be provided anywhere to keep a good QoE. It is not less important to improve user mobility to enable these services in moving vehicles.

2.3.2. Ultra-Reliable Low Latency Communications

URLLC supports a range of advanced services for latency-sensitive connected devices, such as self-driving vehicles, railway and transport infrastructure, 4.0 industry, security in critical missions and applications and interactive video games.

2.3.3. Massive Machine Type Communications

mMTC deployment scenarios usually envisage an assume density between 1.000 and 100.000 users per square kilometre, leading up to 1.000.000 devices per cell. The UP traffic generated is hence expected mostly on the uplink (UL), on the DL typically only device activation with data between 100 and 1.000 bits per activation are required to be transmitted. This would require networks to handle a number of connections 3-4 orders of magnitude higher than eMBB, while each connected device will lead to a data traffic

5 to 8 orders of magnitude lower than 5G eMBB. mMTC will address the needs of Low Power Wide Area (LPWA) networks, which include low-cost devices with long battery life and wide area coverage. It benefits rural activities such as agriculture as well as industrial uses and the management of cities [23].

In Figure 9 is illustrated, how depending on the type of service needed, it will move closer to or further away from each deployment scenario.

In the case of smart cities, connectivity of many devices will be needed, without the need for low latency or high speeds. In the case of Smart Home/Building, it is somewhere in between the connectivity of many devices and the need for a slightly faster data rate. For 3D video, UHD screens or for gaming and working in the cloud you will need an intermediate latency and an intermediate speed as well. For augmented reality applications, industry automation and especially self-driving cars, the reduction of the response time becomes more important. Finally, for critical mission application low latency and the connection of multiple devices is needed.

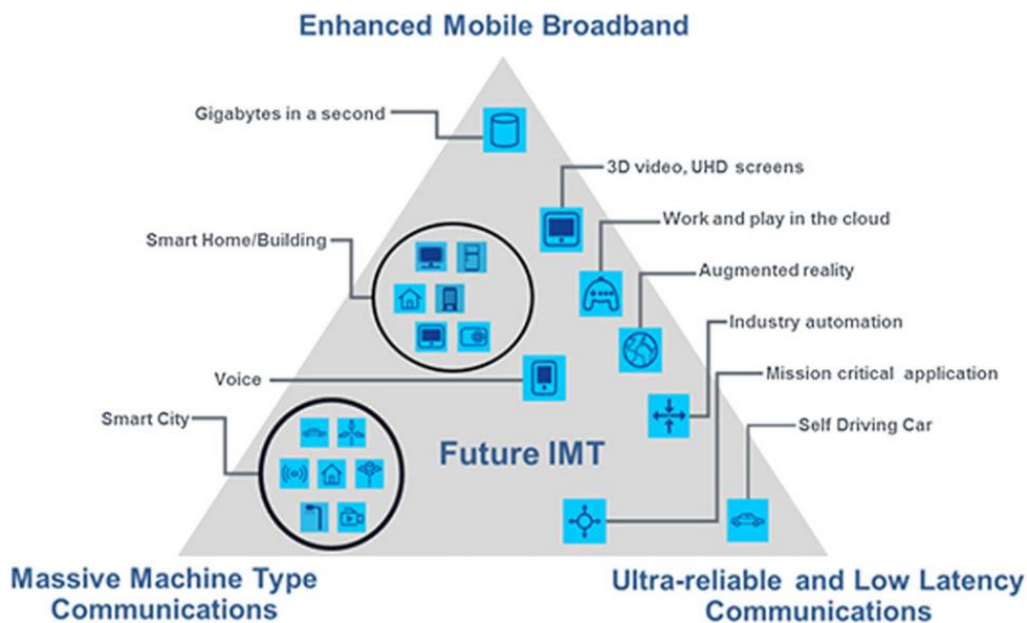


Figure 9: 5G applications [24]

2.4. 5G Network Functions Deployment in Slices

NF in different slices can be used in different configurations, or be placed further from or closer to the UE, depending on the vertical application using the slice. To reduce the latency, control functions are moved to the edge. Network functions can be implemented in virtual machines and executed using a cloud infrastructure. Figure 10 shows the different configurations depending on the type of service required.

eMBB slice: use a high capacity radio bearer and have two UPFs, one in the edge and other one in the cloud, to better support user mobility.

URLLC slice: have a radio bearer with low delay and medium capacity and many control functions moved to the edge to further reduce latency.

MIoT slice: low bit-rate radio bearer, a single UPF assuming low mobility and most control NFs in the core, assuming that latency is not important.

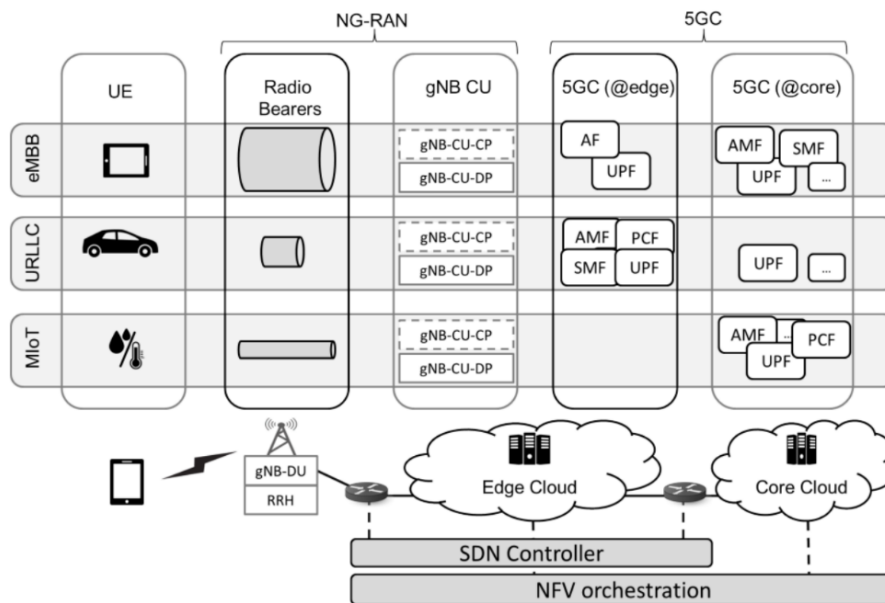


Figure 10: 5G NFs Slices [17]

2.5. Satellite Backhaul Technology

A satellite system can be used as a transport network within the 5G network in order to provide connectivity between areas. This technology started in the 2000s and is being exploited for the clear advantages it offers. The Backhaul between the RAN and the CN can therefore rely on such systems and is used to transport 5G N1, N2 and N3 interfaces. Satellite systems remain the only or the most viable system to provide connectivity in specific contexts [25].

The minimum requirements that a satellite has to have to meet the 5G Key Performance Indicator (KPI) are as follows:

- **Ubiquity:** It has to have the ability to be present everywhere at the same time offering high speed capacity using the following elements: capacity backfill within geographical gaps, overflow to satellites when terrestrial links exceed their capacity, global blanket coverage, back-up/resilience for network backhaul and, most importantly, emergency communication.
- **Mobility:** Satellite is the single most readily available technology capable of providing connectivity anywhere on the planet, whether on land, at sea or in the air for moving vehicles such as trains, ships or planes.
- **Broadcasting:** The satellite shall be capable of distributing broadcast or multicast content with information centric networks and caching at any time.

- Security: Satellite communications should offer solutions for secure, reliable and consistent deployment in difficult communication scenarios, including emergency response.

Satellite and terrestrial integration in 5G can be investigated around four main uses cases [25]:

- Use Case 1: “Edge delivery & offload for multimedia content and Multi-access Edge Computing (MEC) VNF software”. Enabling fast and efficient broadcast communication to the edge of the network for live broadcasts, broadcast streams and distribution of MEC VNF updates.
- Use Case 2: “5G fixed backhaul”: Connectivity where it is environmentally unfriendly or difficult to deploy, whether in natural areas, rural areas, maritime services or isolated areas.
- Use Case 3: “5G to premises”: Networks for use in combination with terrestrial broadband networks, either to improve coverage or improve performance.
- Use Case 4: “5G moving platform backhaul broadband connectivity to moving platforms, such as aircraft or ships.

Figure 11 would correspond to the “5G fixed backhaul”, which will be the schematic used in this work.

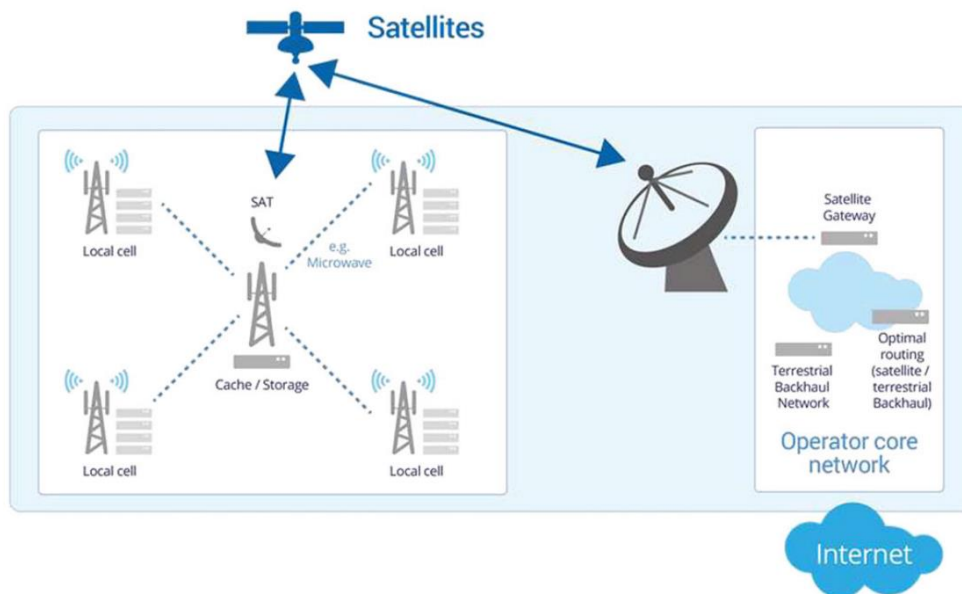


Figure 11: Satellite Backhaul [25]

2.6. Multi-access Edge Computing

MEC is a new service where network solution is designed to be implemented at servers located near mobile stations [26]. The idea behind MEC is that tasks are processed closer to the client, thereby reducing network congestion, latency and ensure efficient network operation and service delivery, and thereby improve QoE. MEC is an evolution of cloud computing using mobile and cloud technologies and edge computing, where

virtualisation technologies, including virtual machines and containers are used. Many service providers move workloads and services from the core network, mostly in data centres, to the edge of the core network.

Service providers can also benefit from MEC to collect more specific data about customers in terms of content, location and interests, in order to introduce new services or use this data for commercial reasons [27].

MEC applications [28]:

- Vehicle to everything (V2X): refers to the communication between a vehicle and anything that may affect or be affected by the car
- Video analytics: Providing video surveillance for cameras connected to the network
- Location services: Positioning solution using MEC
- IoT and Big-data: MEC can provide computational and storage resources in close proximity to data sources, reducing IoT data and signalling to ensure fast response or to enable new services
- Augmented reality
- Optimized local content distribution
- Data caching

2.7. 5G Frequency Bands and Frame Structure

5G uses a variety of frequency bands known as FR1 (Frequency Range), the ones that are below 7.125 GHz, and FR2, also known as mmWave, the ones above 24.250 GHz.

Frequency range designation	Frequency range (MHz)
FR1	410- 7125
FR2	24250-52600

Table 1: Frequency range

Lower frequencies have better range but offer lower data rates. In the case of FR2 they have higher maximum bandwidth due to the higher carrier frequencies. At mmWave spectrum, data rates are higher but waves can not get through walls, trees or even glass.

Thus, there is a trade-off between coverage and speed, depending on the deployment scenario. Low band spectrum can be used in rural areas. High band spectrum could be ideal for short range communications in dense urban areas and within buildings. They could serve thousands of users in a stadium, critical IoT applications or applications that need URLLC are more use cases.

As can be seen in Figure 12, depending on the frequency, there will be a different coverage radius and different capacity.

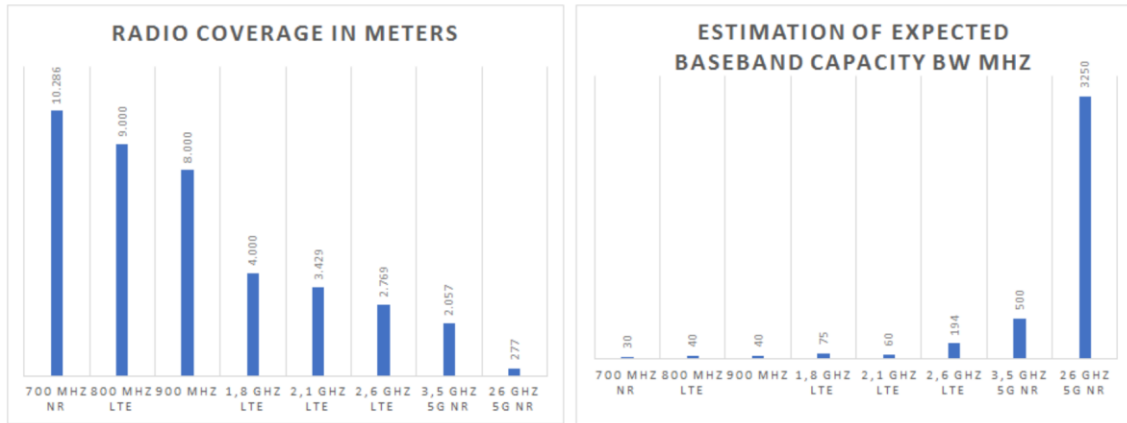


Figure 12: Characteristics of frequency range [29]

5G NR uses flexible SCS derived from basic 15 kHz SCS used in LTE, and can be implemented from 15 kHz up to 240 kHz.

The 240 kHz SCS can be used to provide millimeter wave broadcast signals. All other SCS are compatible with data and signaling, except 60 kHz, which is for physical data channels only.

Reasons for having different subcarrier spacing:

The modulation that uses 5G is OFDM. In OFDM, the number of subcarriers that can be obtained from a specific frequency range is directly related to the spectrum efficiency (bits per Hz per second). The more subcarriers are available in a frequency range, the smaller the spacing between subcarriers is, the more data can be transmitted or received.

The OFDM symbol length is related to the SCS, the narrower it is the SCS, the longer OFDM symbol length will be. With a longer OFDM symbol, there is more space for the cyclic prefix (CP). With a longer CP, you get a larger cell radius and more fading channel tolerance.

In the FR1, where frequency ranges are lower, the wide spectrum is narrow, so the SCS should be as small as possible to get as many subcarriers as possible.

In FR2, as the carrier frequency increases up to 52 GHz, the frequency drift due to transmitter or receiver motion is greater, as the doppler effect becomes more noticeable as the frequency rises. So in this FR the use of larger SCS is needed.

For low latency applications, the use of high SCS would be of interest, due to its short slot duration. The higher the SCS, the less duration of the slot.

2.8. Open Source 5G Frameworks

In this section, srsRAN and OAI open source 5G frameworks are analyzed.

2.8.1. srsRAN

srsRAN is a free and open-source 4G and 5G software radio suite, it implements 4G LTE and 5G NR UE modem entirely in software [30].

The srsRAN package of the latest version released on 22.04.2022 includes:

- srsUE: a full stack 4G and 5G NSA / SA UE application
- srsENB: a full stack 4G eNodeB and 5G NSA / SA gNB capabilities
- srsEPC: a light-weight 4G EPC implementation with Mobility Management Entity (MME), Home Subscriber Server (HSS) and S/P-GW

The SA mode was implemented in the 22.04 version, so OAI platform was chosen, due to at the time of starting with this master thesis, OAI was more suitable for analysing 5G network.

2.8.2. OAI

The OpenAirInterface Software Alliance (OSA) is French non-profit organization that gathers a community of developers from all over the world, who work together to build wireless cellular RAN and CN technologies [31].

OAI 5G RAN

5G RAN supports NSA / SA gNB and NSA / SA UE. OAI RAN sub-activities include [32]:

L1-simulation framework: In this case the RF Simulator is used, which replaced the radio board (USRP device for example) by software communications (TCP/IP) to make possible a without a RF board. The OAI gNB and the OAI UE communicate as if there were an RF interface between them, but without real-time clock constraints. The I/Q samples are transmitted over this radio channel simulator. This simulator is the ideal tool to check signal processing algorithms and protocols implementation.

L2-simulation framework: In order to simulate a large number of users, this framework allows to connect the OAI UE with the OAI gNB through the nFAPI interface. NFAPI separates the gNB into MAC entity and PHY entity and the base station connects through this interface to a channel proxy that simulates the channel and allows to connect many UE.

There is also available a CU/DU split version of the 5G gNB deployment in OAI. This split mode allows to:

- Control Plane exchanges between the CU and DU entities over F1-C interface, to enable UE registration and establish PDU session.
- User plane traffic over F1-U interface, using GTP-U protocol.

The OAI 5G-NR includes the following characteristics for the gNB and UE [33]:

- TDD and FDD, with bandwidths of 10,20,40,80 and 100 MHz
- 30 KHz of SCS with normal CP
- Intermediate DL and UL frequencies to interface with IF equipment
- Single antenna port
- 14 OFDM symbols in UL and DL
- Highly efficient 3GPP compliant LDPC and polar encoder and decoder
- Encoder and decoder for short blocks
- Support for UL transform precoding (SC-FDMA)

OAI 5G CN

5G CN is composed of the following components: AMF, SMF, NRF, AUSF, UDM and UPF [34]. This CN supports basic procedures such as registration, de-registration and PDU session management. Several UEs can be implemented at the same time and several PDU sessions can be implemented. With the NRF block running, features such as NF registration and discovery can be added.

For deployment 2 options can be used:

- 1.- The simplest is using the AMF, SMF, NRF and UPF blocks.
- 2.- A second option would be to add to the previous blocks UDM, AUSF and Unified Data Repository (UDR) to add more functions to the network and UE authentication.

3. Implementation

This section will explain the whole procedure that has been followed to set up the environment.

OAI has been chosen as the platform to create the network and run the simulations, as at the time of starting the work, it was the most developed platform to perform simulations between the gNB and the 5G CN. SA Setup with OAI NR UE Soft modem is the setup selected, where the OAI RAN (RFSIMULATOR) and CN components are build using docker images and docker-compose. The steps of [35] have been followed, adapting it to our system.

3.1. Retrieving the Images on Docker

First step would be to pull the following images from docker:

```
$ docker pull mysql:5.7
$ docker pull oaisoftwarealliance/oai-amf:latest
$ docker pull oaisoftwarealliance/oai-nrf:latest
$ docker pull oaisoftwarealliance/oai-smf:latest
$ docker pull oaisoftwarealliance/oai-spgwu-tiny:latest

$ docker pull oaisoftwarealliance/oai-gnb:develop
$ docker pull oaisoftwarealliance/oai-nr-ue:develop
```

The mysql container will serve as a database to hold the UE data, the oai-amf container will perform the functions of the AMF block, the oai-nrf the functions of the NRF block, the oai-smf the functions of the SMF block, the oai-spggw block the functions of the UPF block, the oai-gnb functions of the gNB and the oai-nr-ue will act as if it were a user.

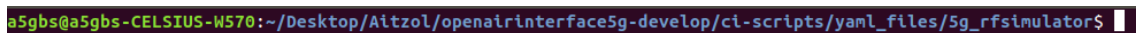
After pulling all containers, must be re-tagged for docker-compose file to work:

```
$ docker image tag oaisoftwarealliance/oai-amf:latest oai-amf:latest
$ docker image tag oaisoftwarealliance/oai-nrf:latest oai-nrf:latest
$ docker image tag oaisoftwarealliance/oai-smf:latest oai-smf:latest
$ docker image tag oaisoftwarealliance/oai-spgwu-tiny:latest oai-spgwu-tiny:latest

$ docker image tag oaisoftwarealliance/oai-gnb:develop oai-gnb:develop
$ docker image tag oaisoftwarealliance/oai-nr-ue:develop oai-nr-ue:develop
```

3.2. Deploy Containers

All the following commands need to be run from the folder “openairinterface5g-develop/ci-scripts/yaml_files/5g_rfsimulator” as seen in the Figure 13.



```
a5gbs@a5gbs-CELSIUS-W570:~/Desktop/Aitzol/openairinterface5g-develop/ci-scripts/yaml_files/5g_rfsimulator$
```

Figure 13: Folder from where the commands will be executed

3.2.1. Deploy OAI 5G Core Network

To deploy the OAI 5G CN, the command in Figure 14 must be executed:

```
$ docker-compose up -d mysql oai-nrf oai-amf oai-smf oai-spgwu oai-ext-dn
Creating network "rfsim5g-oai-public-net" with driver "bridge"
Creating network "rfsim5g-oai-traffic-net" with driver "bridge"
Creating rfsim5g-oai-nrf ... done
Creating rfsim5g-mysql ... done
Creating rfsim5g-oai-amf ... done
Creating rfsim5g-oai-smf ... done
Creating rfsim5g-oai-spgwu ... done
Creating rfsim5g-oai-ext-dn ... done
```

Figure 14: CN execution

As can be seen in the Figure 15, after executing the command in Figure 14, 2 docker-bridges are created: "rfsim5g-oai-public-net" and "rfsim5g-oai-traffic_net-net", together with the CN blocks.

```
rfsim5g-public: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.71.129 netmask 255.255.255.192 broadcast 192.168.71.191
inet6 fe80::42:abff:fe93:62f3 prefixlen 64 scopeid 0x20<link>
ether 02:42:ab:93:62:f3 txqueuelen 0 (Ethernet)
RX packets 4 bytes 112 (112.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 32 bytes 5584 (5.5 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

rfsim5g-traffic: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.72.129 netmask 255.255.255.192 broadcast 192.168.72.191
inet6 fe80::42:fff:feed:2680 prefixlen 64 scopeid 0x20<link>
ether 02:42:0f:ed:26:80 txqueuelen 0 (Ethernet)
RX packets 9225 bytes 489256 (489.2 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 18540 bytes 27992550 (27.9 MB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figure 15: Created bridges after executing CN

3.2.2. Deploy OAI gNB in RF Simulator Mode and in SA Mode

It must be taken into account that for the correct operation of the environment, before executing this point, the CN must be in correct operation, for that the following command in Figure 16 is used:

```
$ docker-compose ps -a
-----
Name                                Command                                State                                Ports
-----
rfsim5g-mysql                        docker-entrypoint.sh                  Up (healthy)                        3306/tcp, 33060/tcp
rfsim5g-oai-amf                      /bin/bash /openair-amf/bin ...      Up (healthy)                        38412/sctp, 80/tcp, 9090/tcp
rfsim5g-oai-ext-dn                   /bin/bash -c apt update; ...      Up (healthy)
rfsim5g-oai-nrf                      /bin/bash /openair-nrf/bin ...      Up (healthy)                        80/tcp, 9090/tcp
rfsim5g-oai-smf                      /bin/bash /openair-smf/bin ...      Up (healthy)                        80/tcp, 8805/udp, 9090/tcp
rfsim5g-oai-spgwu                    /openair-spgwu-tiny/bin/en ...      Up (healthy)                        2152/udp, 8805/udp
```

Figure 16: CN blocks in healthy state

Once the correct functioning of the CN is checked, the gNB block is deployed as it follows in Figure 17.

```
$ docker-compose up -d oai-gnb
rfsim5g-mysql is up-to-date
rfsim5g-oai-nrf is up-to-date
rfsim5g-oai-amf is up-to-date
rfsim5g-oai-smf is up-to-date
rfsim5g-oai-spgwu is up-to-date
rfsim5g-oai-ext-dn is up-to-date
Creating rfsim5g-oai-gnb ... done
```

Figure 17: gNB correct execution

It can be verified that the gNB is connected to the AMF executing “docker logs rfsim5g-oai-amf” command and getting the result in Figure 18.

```
Successfully sent 512 bytes on stream 0
Sending NG_SETUP_RESPONSE OK
gNB with gNB_id 0x0, assoc_id 4 has been attached to AMF
```

-gNBs' information-					
Index	Status	Global ID	gNB Name	PLMN	
1	Connected	0x0	gnb-rfsim	208, 99	

Figure 18: gNB connection in the AMF

3.2.3. Deploy OAI NR-UE in RF Simulator Mode and in SA Mode

Now, the UE is implemented executing the command in Figure 19:

```
$ docker-compose up -d oai-nr-ue
rfsim5g-mysql is up-to-date
rfsim5g-oai-nrf is up-to-date
rfsim5g-oai-amf is up-to-date
rfsim5g-oai-smf is up-to-date
rfsim5g-oai-spgwu is up-to-date
rfsim5g-oai-ext-dn is up-to-date
rfsim5g-oai-gnb is up-to-date
Creating rfsim5g-oai-nr-ue ... done
```

Figure 19: OAI NR-UE correct deployment

You can have as many UE as you want, to deploy a second user the following has to be added in the oai_db.sql file as shown in Figure 20 and in the docker-compose.yaml file as shown in Figure 21.

```
INSERT INTO `users` VALUES ('20899010001101','1','55000000000000',NULL,'PURGED',50,40000000,100000000,47,000000000,1,0xfec86ba6eb707ed08905757b1bb44b8f,0,0,0x40,'ebd07771ace8677a',0xc42449363bbad02b66d16bc975d77cc1);
```

Figure 20: Adding a user in the SQL configuration

```

oai-nr-ue2:
  image: oai-nr-ue:develop
  privileged: true
  container_name: rfsim5g-oai-nr-ue2
  environment:
    RFSIMULATOR: 192.168.71.140
    FULL_IMSI: '208990100001101'
    FULL_KEY: 'fec86ba6eb707ed08905757b1bb44b8f'
    OPC: 'C42449363BBAD02B66D16BC975D77CC1'
    DNN: oai
    NSSAI_SST: 1
    NSSAI_SD: 1
    USE_ADDITIONAL_OPTIONS: -E --sa --rfsim -r 106 --numerology 1 -C 3619200000 --
  nokrnmod --log_config.global_log_options level,nocolor,time
  depends_on:
    - oai-gnb
  networks:
    public_net:
      ipv4_address: 192.168.71.151
  healthcheck:
    test: /bin/bash -c "pgrep nr-uesoftmodem"
    interval: 10s
    timeout: 5s
    retries: 5

```

Figure 21: Adding a user in docker-compose configuration file

In the Figure 22 can be seen how the environment will look like after deploying all the containers in docker, with the corresponding IP addresses. The interfaces to be analyzed later on are the Uu, N2, N3 and N6. The NG-AP and GTP-U protocol, which are marked in red, are important as they are the protocols used by the AMF to communicate with the gNB and by the UPF to communicate with the UPF.

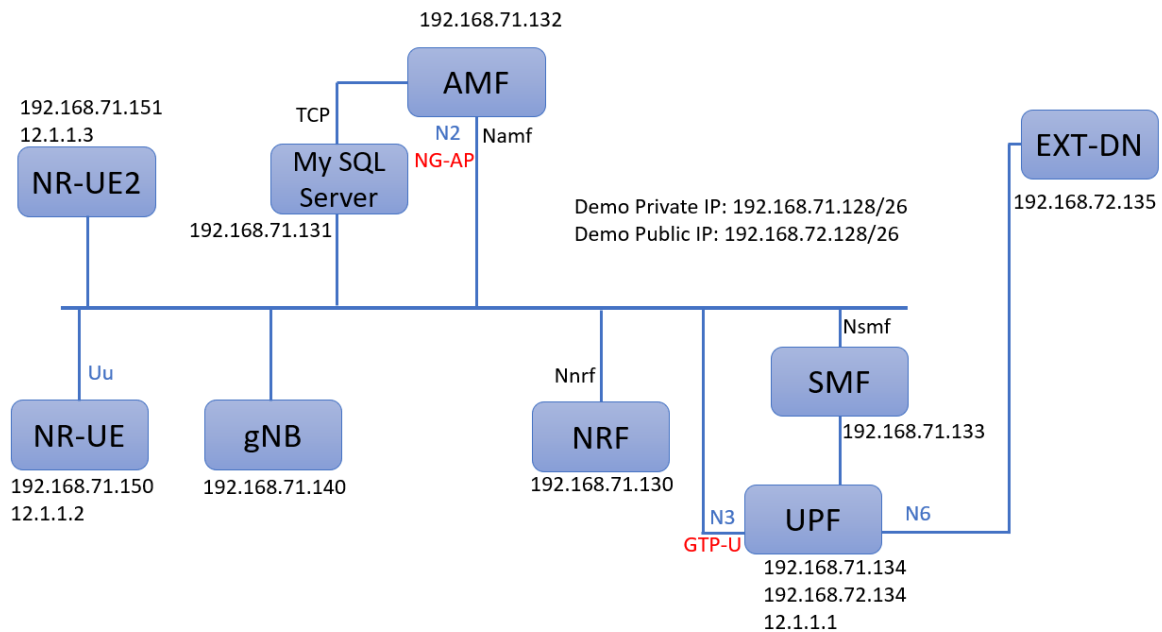


Figure 22: Final setup with each block information

3.3. Traffic Check

To check internet connectivity, ping from the UE block to www.lemonde.fr through the oaitun_ue1 interface is done. As it can be seen in the Figure 23, the ping arrives without any problem.

```

root@4baedba75c19:/opt/oai-nr-ue# ping -I oaitun_ue1 -c 10 www.lemonde.fr
PING s2.shared.global.fastly.net (151.101.114.217) from 12.1.1.2 oaitun_ue1: 56(84) bytes of data.
64 bytes from 151.101.114.217 (151.101.114.217): icmp_seq=1 ttl=56 time=11.7 ms
64 bytes from 151.101.114.217 (151.101.114.217): icmp_seq=2 ttl=56 time=8.99 ms
64 bytes from 151.101.114.217 (151.101.114.217): icmp_seq=3 ttl=56 time=13.9 ms
64 bytes from 151.101.114.217 (151.101.114.217): icmp_seq=4 ttl=56 time=20.9 ms
64 bytes from 151.101.114.217 (151.101.114.217): icmp_seq=5 ttl=56 time=10.8 ms
64 bytes from 151.101.114.217 (151.101.114.217): icmp_seq=6 ttl=56 time=12.6 ms
64 bytes from 151.101.114.217 (151.101.114.217): icmp_seq=7 ttl=56 time=13.8 ms
64 bytes from 151.101.114.217 (151.101.114.217): icmp_seq=8 ttl=56 time=9.36 ms
64 bytes from 151.101.114.217 (151.101.114.217): icmp_seq=9 ttl=56 time=28.6 ms
64 bytes from 151.101.114.217 (151.101.114.217): icmp_seq=10 ttl=56 time=8.36 ms

--- s2.shared.global.fastly.net ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9012ms
rtt min/avg/max/mdev = 8.369/13.930/28.693/5.992 ms
root@4baedba75c19:/opt/oai-nr-ue#

```

Figure 23: Checking internet connectivity

Now, UDP traffic in DL is going to be tested, for this, iperf server is started inside the NR-UE container, and iperf client is started inside the ext-dn container. As can be seen in the Figure 24 and Figure 25, the traffic sent from the server reaches the client in its entirety.

```

root@4baedba75c19:/opt/oai-nr-ue# iperf -B 12.1.1.2 -u -i 1 -s
-----
Server listening on UDP port 5001
Binding to local address 12.1.1.2
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)
-----
[ 3] local 12.1.1.2 port 5001 connected with 192.168.72.135 port 43088
[ ID] Interval      Transfer      Bandwidth      Jitter      Lost/Total Datagrams
[ 3] 0.0- 1.0 sec  63.2 KBytes   517 Kbits/sec  0.433 ms    0/ 44 (0%)
[ 3] 1.0- 2.0 sec  63.2 KBytes   517 Kbits/sec  0.368 ms    0/ 44 (0%)
[ 3] 2.0- 3.0 sec  61.7 KBytes   506 Kbits/sec  0.412 ms    0/ 43 (0%)
[ 3] 3.0- 4.0 sec  63.2 KBytes   517 Kbits/sec  0.395 ms    0/ 44 (0%)
[ 3] 4.0- 5.0 sec  61.7 KBytes   506 Kbits/sec  0.383 ms    0/ 43 (0%)
[ 3] 5.0- 6.0 sec  63.2 KBytes   517 Kbits/sec  0.497 ms    0/ 44 (0%)

```

Figure 24: Iperf server check

```

a5gbs@a5gbs-CELSIUS-W570:~/Desktop/Aitzol/openairinterface5g-develop/ci-
ator$ docker exec -it rfsim5g-oai-ext-dn /bin/bash
root@53554702db2f:/# iperf -c 12.1.1.2 -u -i 1 -t 20 -b 500K
-----
Client connecting to 12.1.1.2, UDP port 5001
Sending 1470 byte datagrams, IPG target: 22968.75 us (kalman adjust)
UDP buffer size: 208 KByte (default)
-----
[ 3] local 192.168.72.135 port 43088 connected with 12.1.1.2 port 5001
[ ID] Interval      Transfer      Bandwidth
[ 3] 0.0- 1.0 sec  64.6 KBytes   529 Kbits/sec
[ 3] 1.0- 2.0 sec  63.2 KBytes   517 Kbits/sec
[ 3] 2.0- 3.0 sec  61.7 KBytes   506 Kbits/sec
[ 3] 3.0- 4.0 sec  63.2 KBytes   517 Kbits/sec
[ 3] 4.0- 5.0 sec  61.7 KBytes   506 Kbits/sec

```

Figure 25: Iperf client check

3.4. Explanation on the Configuration in the Docker-compose.yaml

In Appendix 1 can be seen all the code of the docker-compose.yaml file, in this section will be explained what has been thought to be the most important. Two parts can be distinguished, on the one hand the individual programming of each block, and on the other hand the programming of the networks.

3.4.1. Content of the Blocks

In the table 2 is a summary of the most relevant information for each block. This information has been obtained from the file docker-compose.yaml, which can be found in appendix 1.

Container name	IP	Info
oai-nrf	192.168.71.130	-
mysql	192.168.71.131	-
oai-amf	192.168.71.132	depends on: - oai-nrf , - mysql connected with: - mysql - NRF
oai-smf	192.168.71.133	depends on: - oai-nrf , - oai-amf connected with: - UPF - NRF
oai-spgw	192.168.71.134 192.168.72.134	depends on: - oai-nrf , - oai-smf connected with: - SMF - NRF
oai-ext-dn	192.168.72.135	depends on: - oai-spgw "ip route add 12.1.1.0/24 via 192.168.72.134 dev eth0"
oai-gnb	192.168.71.140	depends on: - oai-ext-dn connected with: - AMF
oai-nr-ue	192.168.71.150	depends on: - oai-gnb connected with: - gNB

Table 2: Information of each block

oai-nrf

The first block to be deployed is the oai-nrf as this block does not depend on any other block. The function of this block is to allow 5G NFs to register and discover each other via a standards-based API. The IPv4 that has been assigned is 192.168.71.130, as shown in Figure 26, in which the ifconfig command has been executed from inside the block. This procedure will be done in the other blocks as well.

```

a5gbs@a5gbs-CELSIUS-W570:~/Desktop/Aitzol/openairinterface5g-develop/ci-scripts/yaml_files/5g_rfsimul
ator$ docker exec -it rfsim5g-oai-nrf /bin/bash
root@2b9f70a2a097:/openair-nrf# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.71.130 netmask 255.255.255.192 broadcast 192.168.71.191
    ether 02:42:c0:a8:47:82 txqueuelen 0 (Ethernet)
    RX packets 11636 bytes 1301720 (1.3 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 7701 bytes 713773 (713.7 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Figure 26: OAI-NRF block interfaces

mysql

The next block to be implemented is mysql and the IP assigned to it is 192.68.71.131. In this case there is no interface figure, as no commands could be executed from the block.

oai-amf

The next container to be activated is oai-amf, which it depends on the oai-nrf container and mysql, so the correct order of deployment is important. This block receives all connection and session related information from the UE through the N1 and N2 interfaces. IPv4 192.168.71.132 has been assigned, as can be observed in Figure 27. The oai-amf is connected to the mysql server and the NRF.

```

a5gbs@a5gbs-CELSIUS-W570:~/Desktop/Aitzol/openairinterface5g-develop/ci-scripts/yaml_files/5g_rfsimul
ator$ docker exec -it rfsim5g-oai-amf /bin/bash
root@5593d2d5b76f:/openair-amf# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.71.132 netmask 255.255.255.192 broadcast 192.168.71.191
    ether 02:42:c0:a8:47:84 txqueuelen 0 (Ethernet)
    RX packets 928 bytes 87457 (87.4 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 866 bytes 75891 (75.8 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 4 bytes 257 (257.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 4 bytes 257 (257.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Figure 27: OAI-AMF block interfaces

oai-smf

Then it comes oai-smf, the IP assigned to this block is 192.168.71.133, as is shown in Figure 28. It depends on the oai-nrf and oai-amf blocks and it is connected with the UPF and NRF. Its responsibility is to create, update and remove PDU sessions and manage session context with the UPF.


```

extl
a5gbs@a5gbs-CELSIUS-W570:~/Desktop/Aitzol/openairinterface5g-develop/ci-scripts/yaml_files/5g_rfsimul
ator$ docker exec -it rfsim5g-oai-smf /bin/bash
root@7e4fab1005c9:/openair-smf# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.71.133 netmask 255.255.255.192 broadcast 192.168.71.191
    ether 02:42:c0:a8:47:85 txqueuelen 0 (Ethernet)
    RX packets 5397 bytes 446685 (446.6 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 7253 bytes 762137 (762.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 18 bytes 1194 (1.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 18 bytes 1194 (1.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Figure 28: OAI-SMF block interfaces

oai-spgwu

In the case of oai-spgwu, it depends on the oai-nrf and oai-smf and connects to the SMF and NRF. In Figure 29 it is visible how it uses the eth0 interface, with the 192.168.71.134 IP to connect to the "rfsim5g-oai-public-net" network, the one where almost all blocks are hosted; and the eth1 interface, with 192.168.72.134 IP address to connect to the "rfsim5g-oai-traffic-net", which it uses to connect to the External Data Network (EXT-DN). This 2 network will be seen later. It also has the tun0 interface, with 12.1.1.1 IP which is used to connect to the UE.

```

extl
a5gbs@a5gbs-CELSIUS-W570:~/Desktop/Aitzol/openairinterface5g-develop/ci-scripts/yaml_files/5g_rfsimul
ator$ docker exec -it rfsim5g-oai-spgwu /bin/bash
root@22d94de8bfa8:/openair-spgwu-tiny# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.71.134 netmask 255.255.255.192 broadcast 192.168.71.191
    ether 02:42:c0:a8:47:86 txqueuelen 0 (Ethernet)
    RX packets 5443 bytes 459351 (459.3 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 12973 bytes 5203583 (5.2 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.72.134 netmask 255.255.255.192 broadcast 192.168.72.191
    ether 02:42:c0:a8:48:86 txqueuelen 0 (Ethernet)
    RX packets 5138 bytes 4495607 (4.4 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2227 bytes 201152 (201.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 4 bytes 257 (257.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 4 bytes 257 (257.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
    inet 12.1.1.1 netmask 255.255.255.0 destination 12.1.1.1
    unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500 (UNSPEC)
    RX packets 24 bytes 12530 (12.5 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2851 bytes 4253830 (4.2 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Figure 29: OAI-SPGWU block interfaces

oai-ext-dn

Then it comes oai-ext-dn, this block simulates the external network, which could be the internet or any other network. This is the block that will be used later to generate the traffic, simulating an external network. It is assigned the IP 192.168.72.135 and in order

to have connectivity with the UE and UPF block the following command is added in the docker-compose.yaml file when executing it: "ip route add 12.1.1.0/24 via 192.168.72.134 dev eth0".

oai-gnb

The oai-gnb block depends on the oai-ext-dn and connects to the AMF. The assigned IP is 192.168.71.140, as shown in Figure 30. This block is important because this block connects to the CN, and from here the N2 and N3 interfaces come out, which are the ones of interest.

```
a5gbs@a5gbs-CELSIUS-W570:~/Desktop/Aitzol/openairinterface5g-develop/ci-scripts/yaml_files/5g_rfsimul
ator$ docker exec -it rfsim5g-oai-gnb /bin/bash
root@d86461c1dc76:/opt/oai-gnb# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.71.140 netmask 255.255.255.192 broadcast 192.168.71.191
    ether 02:42:c0:a8:47:8c txqueuelen 0 (Ethernet)
    RX packets 160156852 bytes 3979729783234 (3.9 TB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 157183613 bytes 2959097923004 (2.9 TB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figure 30: OAI-gNB block interfaces

oai-nr-ue

Then there is the oai-nr-ue block, which it depends on the oai-ext-dn block and it connects with the gNB. The 192.168.71.150 IP address is assigned to the interface eth0 of the network the "rfsim5g-oai-public-net", and the interface oaitun_ue1 will receive the 12.1.1.2 IP address, as can be observed in Figure 31. The oaitun_ue1 is an interface that is generated with the RF simulator.

```
a5gbs@a5gbs-CELSIUS-W570:~/Desktop/Aitzol/openairinterface5g-develop/ci-scripts/yaml_files/5g_rfsimul
ator$ docker exec -it rfsim5g-oai-nr-ue /bin/bash
root@4baedba75c19:/opt/oai-nr-ue# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.71.150 netmask 255.255.255.192 broadcast 192.168.71.191
    ether 02:42:c0:a8:47:96 txqueuelen 0 (Ethernet)
    RX packets 99483753 bytes 1829332576370 (1.8 TB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 98161670 bytes 2460039115948 (2.4 TB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 6 bytes 552 (552.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 6 bytes 552 (552.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

oaitun_ue1: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
    inet 12.1.1.2 netmask 255.255.255.0 destination 12.1.1.2
    unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500 (UNSPEC)
    RX packets 2849 bytes 4250834 (4.2 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 24 bytes 12530 (12.5 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figure 31: OAI-NR-UE block interfaces

oai-nr-ue2

Then follows oai-nr-ue2. In this case another user is simulated, for this purpose the IP 192.168.71.151 and 12.1.1.3 are assigned, as found in Figure 32. Everything else will be the same as oai-nr-ue1.

```

a5gbs@a5gbs-CELSIUS-W570:~/Desktop/Aitzol/openairinterface5g-develop/ci-scripts/yaml_files/5g_rfsimul
ator$ docker exec -it rfsim5g-oai-nr-ue2 /bin/bash
root@291c4c68f683:/opt/oai-nr-ue# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.71.151 netmask 255.255.255.192 broadcast 192.168.71.191
    ether 02:42:c0:a8:47:97 txqueuelen 0 (Ethernet)
    RX packets 216112 bytes 4066462969 (4.0 GB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 230631 bytes 5469571874 (5.4 GB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

oaitun_ue1: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
    inet 12.1.1.3 netmask 255.255.255.0 destination 12.1.1.3
    unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500 (UNSPEC)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Figure 32: OAI-NR-UE2 block interfaces

3.4.2. Configuration of the Networks

In this case there are 2 networks, "rfsim5g-oai-public-net" and "rfsim5g-oai-traffic-net", where both networks are deployed in bridge way, as shown in Figure 33.

A bridge network is a Link Layer device which forwards traffic between network segments, in Docker terms, a bridged network allows containers connected to the same bridged network to communicate, while providing isolation from containers that are not connected to that bridged network [36].

```

networks:
  public_net:
    driver: bridge
    name: rfsim5g-oai-public-net
    ipam:
      config:
        - subnet: 192.168.71.128/26
    driver_opts:
      com.docker.network.bridge.name: "rfsim5g-public"
  traffic_net:
    driver: bridge
    name: rfsim5g-oai-traffic-net
    ipam:
      config:
        - subnet: 192.168.72.128/26
    driver_opts:
      com.docker.network.bridge.name: "rfsim5g-traffic"

```

Figure 33: OAI created networks

rfsim5g-oai-public-net

This is a network with IP range 192.168.71.128/26. The block contains the following containers: OAI-SMF, OAI-AMF, OAI-SPGWU, OAI-gNB, OAI-NRF, mysql, and the UEs.

rfsim5g-oai-traffic-net

In this case it is a network with IP range 192.168.72.128/26. The block contains the following containers: OAI-SPGWU and OAI-EXT-DN.

4. Results

4.1. Evaluation Tools

To measure latency and communication status, the ping command has been used. Packet generation has been done using the iperf tool. Iperf requires 2 systems, one to act as a server while the other acts as a client. The client is the one that connects to the server. In this case the NR-UE container has been used as the server and the EXT-DN container as the client.

For the simulations that have been carried out in this project, UDP traffic has been created as it allows to define the transmission speed. It gives information about bandwidth, delay, jitter and datagram loss on the connection.

These are the different command formats that have been used:

Inside the server: `iperf -B 12.1.1.2 -u -i 1 -s`

Inside the client: `iperf -c 12.1.1.2 -u -i 1 -t 20 -b 500K`

-B: To tell the server that the traffic will come through this interface.

-u: To run iperf in client mode.

-i: To establish the reporting intervals.

-s: server command.

-t: length of the test in seconds.

-b: set bandwidth in bit/s.

Wireshark software can be used to analyse the traffic between the interfaces. This tool allows to intercept the traffic within the network, how often each packet is sent or the latency of the packets. One of the great advantages of Wireshark is that it allows to use filters to capture what you are interested in, either filtering by IP address or protocol. Not only that, it also allows to view and create graphs of the size or number of packets received.

In the case of this thesis, Wireshark has been used to analyse the traffic on each interface of interest. Once the data was available, graphs and tables with values were created using the tools provided by this software, such as the size of the packets exchanged, the amount of packets sent or the origin and destination of these.

The Traffic control (tc) tool is used to create block limitations. It is a Linux utility that allows to add network delay, packet loss or bandwidth limits.

In the case of this project it will basically be used to limit the bandwidth, since the datasheet of a SXC1280 device has been used to try to adapt to these limitations.

4.2. Test

In the case of the tests, it has been performed first without restrictions and then with restrictions.

4.2.1. Test Without Restrictions

In this first section, the results obtained will be analysed without adding delays or extra constraints.

4.2.1.1. Uu Air Interface

This is the traffic on the Uu interface between the gNB and UE containers, without creating any traffic from the outside. The traffic that is already between these two blocks belongs to the RF Simulator, which sends I/Q samples.

In the configuration of the docker-compose.yaml file the lines in Figure 34 and Figure 35 can be seen, confirming that the traffic on this interface belongs to the RF Simulator.

```
USE_ADDITIONAL_OPTIONS: --sa -E --rfsim
```

Figure 34: Configuration of the RF Simulator in the gNB block

```
USE_ADDITIONAL_OPTIONS: -E --sa --rfsim -r 106 --numerology 1 -C 3619200000 --nokrnmod
```

Figure 35: Configuration of the RF Simulator in the UE block

--rfsim means that the RF Simulator is used.

--sa means that is a SA mode.

-r 106 the bandwidth in terms of number of resource blocks, 106 resource blocks.

-C 3619200000, means the DL carrier frequency used in Hz.

Figures 36 and 37 show the traffic exchanged between the UE and the gNB, with Figure 36 corresponding to the gNB and Figure 37 to the UE.

```
[RLC] [mac_rlc_status_ind] Radio Bearer (channel ID 4) is NULL for UE with rntiP 1234
[NR_PHY] Number of bad PUCCH received: 369
[NR_MAC] Frame.Slot 768.0
UE ID 0 RNTI 1234 (1/1) PH 0 dB PCMAX 0 dBm, average RSRP -44 (8 meas)
UE 0: dlsch_rounds 15848/93/93/93, dlsch_errors 93, pucch0_DTX 369, BLER 0.00000 MCS 0
UE 0: dlsch_total_bytes 19794152
UE 0: ulsch_rounds 15850/93/92/92, ulsch_DTX 369, ulsch_errors 92
UE 0: ulsch_total_bytes_scheduled 19796650, ulsch_total_bytes_received 19681742

[RLC] [mac_rlc_status_ind] Radio Bearer (channel ID 4) is NULL for UE with rntiP 1234
[NR_PHY] Number of bad PUCCH received: 369
[NR_MAC] Frame.Slot 896.0
UE ID 0 RNTI 1234 (1/1) PH 0 dB PCMAX 0 dBm, average RSRP -44 (8 meas)
UE 0: dlsch_rounds 15976/93/93/93, dlsch_errors 93, pucch0_DTX 369, BLER 0.00000 MCS 0
UE 0: dlsch_total_bytes 19954024
UE 0: ulsch_rounds 15978/93/92/92, ulsch_DTX 369, ulsch_errors 92
UE 0: ulsch_total_bytes_scheduled 19956522, ulsch_total_bytes_received 19841614
```

Figure 36: gNB rfsim running

```

[NR_PHY] =====
[NR_MAC] [773.1] Received TA_COMMAND 31 TAGID 0 CC_id 0
[NR_MAC] [783.1] Received TA_COMMAND 31 TAGID 0 CC_id 0
[NR_MAC] [793.1] Received TA_COMMAND 31 TAGID 0 CC_id 0
[NR_MAC] [803.1] Received TA_COMMAND 31 TAGID 0 CC_id 0
[NR_MAC] [813.1] Received TA_COMMAND 31 TAGID 0 CC_id 0
[NR_MAC] [823.1] Received TA_COMMAND 31 TAGID 0 CC_id 0
[NR_PHY] =====
[NR_PHY] Harq round stats for Downlink: 15819/1/1/1 DLSCHE errors: 0
[NR_PHY] =====
[NR_MAC] [833.1] Received TA_COMMAND 31 TAGID 0 CC_id 0
[NR_MAC] [843.1] Received TA_COMMAND 31 TAGID 0 CC_id 0
[NR_MAC] [853.1] Received TA_COMMAND 31 TAGID 0 CC_id 0
[NR_MAC] [863.1] Received TA_COMMAND 31 TAGID 0 CC_id 0
[NR_MAC] [873.1] Received TA_COMMAND 31 TAGID 0 CC_id 0
[NR_MAC] [883.1] Received TA_COMMAND 31 TAGID 0 CC_id 0
[NR_MAC] [893.1] Received TA_COMMAND 31 TAGID 0 CC_id 0
[NR_PHY] =====
[NR_PHY] Harq round stats for Downlink: 15883/1/1/1 DLSCHE errors: 0
[NR_PHY] =====

```

Figure 37: UE rfsim running

In the case of this interface there is a lot of traffic, because when launching the RF Simulator, this simulator will use most of the available CPU resources. It will be seen later that when Wireshark is started and the computer requires more CPU resources, the transmission speed on this interface will decrease.

Table 3 shows the average traffic in each direction, obtained from Figure 38.

Direction	Average
gNB - UE	686 Mb/s
UE - gNB	932 Mb/s

Table 3: Data rate at the Uu interface

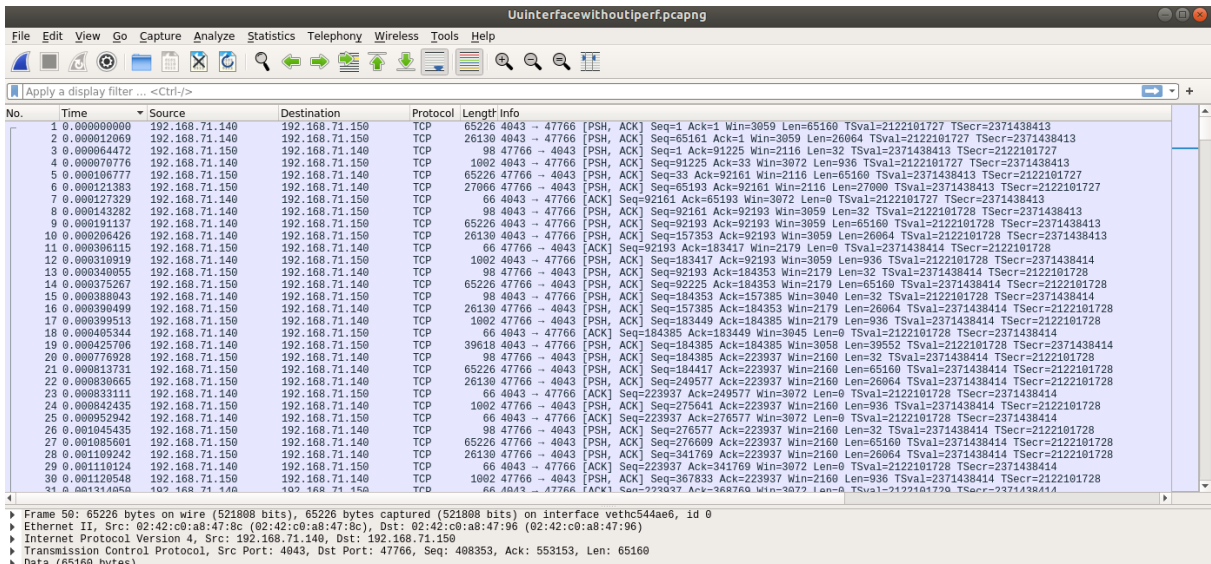
Figure 38 shows the traffic that has moved from 192.168.71.140, the gNB, to 192.168.71.150, the UE, and vice versa.

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
192.168.71.132	192.168.71.140	2	196	1	98	1	98	0.602021	0.0000	—	—
192.168.71.140	192.168.71.150	364,208	7,512M	188,011	3,185M	176,197	4,327M	0.000000	37.1344	686M	932M

Figure 38: Uu traffic flow

In the Figure 39 the traffic exchange between the gNB and the UE can be seen. As said before, I/Q samples are exchanged between the gNB and UE.

This traffic is automatically generated by simulating the communication between users and the gNB.

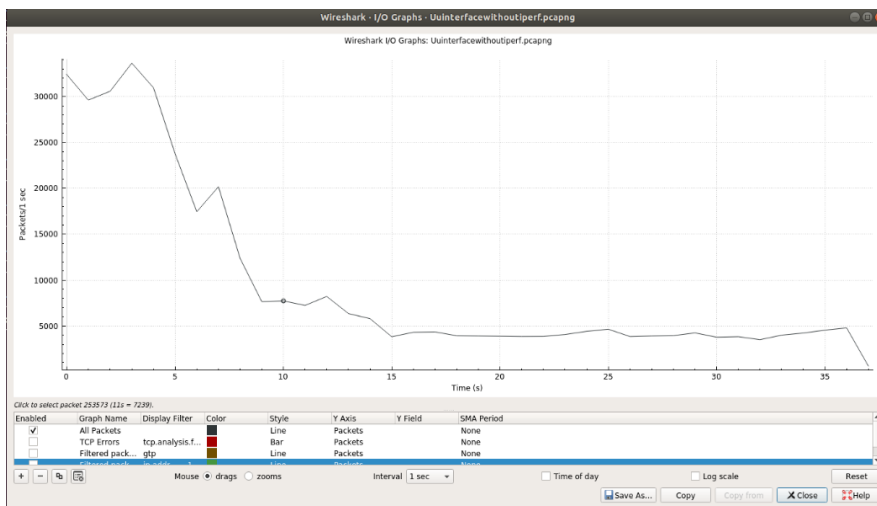


As for the size of the packets sent, as shown in Figure 40, it can be seen that half of the packets have an average size of 45637 bytes, being large packets. It should be noted that the MTU for the loopback interface is 65536 bytes. For the rest of the interfaces it is 1500 bytes.

Wireshark · Packet Lengths · Uuinterface/withoutperf.pcapng									
Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start	
▼ Packet Lengths	364210	20626.16	66	65226	9.8079	100%	38.2900	0.785	
0-19	0	-	-	-	0.0000	0.00%	-	-	
20-39	0	-	-	-	0.0000	0.00%	-	-	
40-79	70722	66.04	66	78	1.9045	19.42%	8.0700	2.815	
80-159	73627	98.00	82	110	1.9827	20.22%	8.0000	0.785	
160-319	0	-	-	-	0.0000	0.00%	-	-	
320-639	5789	524.34	322	630	0.1559	1.59%	0.7300	4.801	
640-1279	50163	1005.58	642	1186	1.3508	13.77%	5.6200	1.960	
1280-2559	471	1545.97	1290	2482	0.0127	0.13%	0.7100	2.809	
2560-5119	302	3686.87	2962	4874	0.0081	0.08%	0.2400	7.452	
5120 and greater	163136	45637.05	5346	65226	4.3931	44.79%	16.7800	0.840	

Figure 40: Uu interface packet lengths

In the Figure 41, the Packets/sec values can be observed. It starts being 30.000 Packets/sec, then the value goes decreasing until the second 15 where it established to an average of 5.000 Packets/sec. This is because the more CPU available is, the faster it will transmit. So, when launching Wireshark as it needs CPU resources, it decreases the Packets/sec sent.



Then, the following iperf traffic is created: `iperf -c 12.1.1.2 -u -i 1 -t 10 -b 3M`

As can be seen in the Figure 42, only 40 to 50 packets per second are detected, when 250 packets should be detected. It can also be seen that the graph is not uniform and has many packet losses.

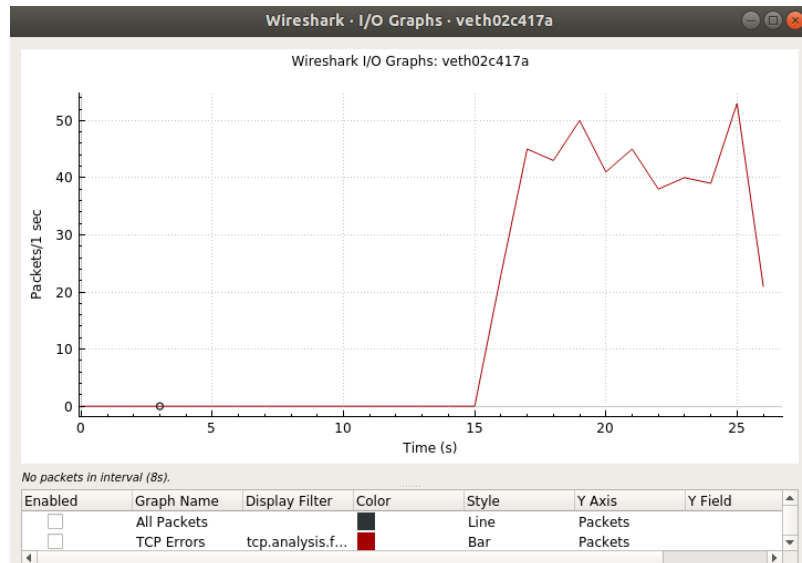


Figure 42: Iperf traffic on the Uu interface

After creating 3Mbits/sec during 10 seconds of traffic from DN to the UE, the following can be seen: only 557 kbits/sec and 26kb/s arrived, as illustrated in Figure 43. After analyzing every interface, a comparison with each one will be done. However, this is because there is so much traffic between the gNB and UE, that the traffic that is created through the interface does not reach the UE.

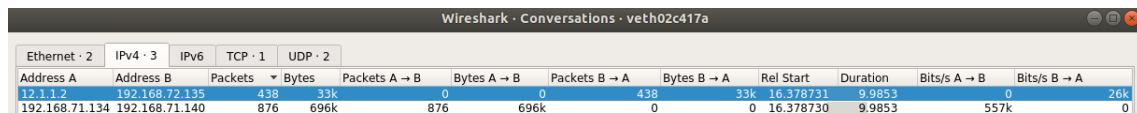


Figure 43: Uu iperf traffic flow

In the Figure 44 can be seen the UDP traffic that goes from the EXT-DN to the UE, the protocol used is GTP-U. This protocol is used for the transmission of user information between the CN and the CN and the RAN. The GTP-U protocol has been used from 3G/4G networks to the current 5G networks. When sending information, a GTP tunnel is created between the receiver and the transmitter to send encapsulated PDUs.

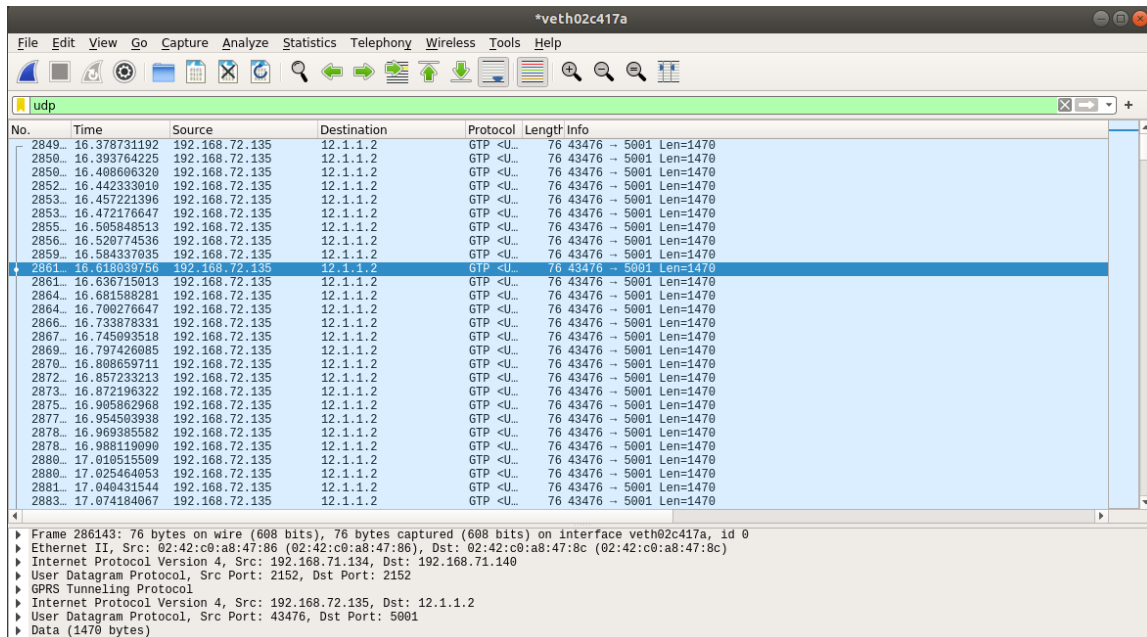


Figure 44: UU iperf traffic created

4.2.1.2. N2 interface

As explained above, only control information will go through this N2 interface. At the start of the gNB and UE session, the NGAP messages described below are exchanged, then SCTP messages, which are explained later. Below is an explanation of the control messages exchanged in the beginning [37]:

NGSetupRequest and NGSetupResponse: To exchange application level data between the gNB and AMF. It deletes any existing application level configuration data in the two nodes and replaces it by the one received at that moment.

InitialUEMessage, registration request: This initial message procedure is used when the NG-RAN has received from the gNB the first UL NAS message transmitted on an RRC connection to be forwarded to an AMF. It contains a UE-AMF message that is transferred without interpretation in the NG-RAN node.

DownlinkNASTransport, Authentication request: After receiving the InitialUEMessage this message is sent from the AMF to the NG-RAN node, the AMF allocates a unique AMF UE NGAP ID to be used for the UE. It sends the key selector, RAND and AUTN to the UE.

UplinkNASTransport, Authentication response: The UE responds to the authentication request.

DownlinkNASTransport, Security mode command: The purpose of this message is to initialize and start NAS signalling security between the UE and the AMF. It includes the ngKSI (key set identifier).

UplinkNASTransport: This message is used when the NG-RAN has received from the radio interface a NAS message to be forwarded to the AMF to which a UE-associated logical NG-connection for the UE exists.

InitialContextSetupRequest: This message goes from the AMF to request the setup of a UE context.

UERadioCapabilityInfoIndication: This is a message from the gNB informing all the details of UE radio capabilities.

SACK: Selective Acknowledgment mechanism, to inform that sender data has been received. It allows the receiver to modify the ACK field to describe noncontinuous blocks of received data, that way the sender could retransmit only is missing at the receiver's end.

InitialContextSetupResponse: This message goes from the NG-RAN node to confirm the setup of a UE context.

PDUSessionResourceSetupRequest: It contains the information required by the NG-RAN node to setup the PDU session. To assign resources on Uu and NG-U for one or several PDU sessions and QoS flows and to setup corresponding DRBs for a given UE.

PDUSessionResourceSetupResponse: It responds to the previous request, in case the PDU sessions resources failed, it tells to the AMF.

Figure 45 shows the NGSetupRequest messages exchanged at the time of connecting the base station to the CN.

Source	Destination	Protocol	Length	Info
192.168.71.140	192.168.71.132	NGAP	132	NGSetupRequest
192.168.71.132	192.168.71.140	NGAP	576	NGSetupResponse

Figure 45: Moment when the gNB is connected to the CN

On the other hand, the Figure 46 represents the messages exchanged when connecting the UE to the gNB.

580.487150435	192.168.71.140	192.168.71.132	NGAP/N...	146	InitialUEMessage, Registration request
580.614179149	192.168.71.132	192.168.71.140	NGAP/N...	630	DownlinkNASTransport, Authentication request
580.618831177	192.168.71.140	192.168.71.132	NGAP/N...	146	UplinkNASTransport, Authentication response
580.621287601	192.168.71.132	192.168.71.140	NGAP/N...	462	DownlinkNASTransport, Security mode command
580.627447483	192.168.71.140	192.168.71.132	NGAP/N...	174	UplinkNASTransport
580.630300059	192.168.71.132	192.168.71.140	NGAP/N...	1390	InitialContextSetupRequest
580.640224480	192.168.71.140	192.168.71.132	NGAP	122	UERadioCapabilityInfoIndication
580.843921253	192.168.71.132	192.168.71.140	SCTP	62	SACK
580.843951339	192.168.71.140	192.168.71.132	NGAP	86	InitialContextSetupResponse
581.047922807	192.168.71.132	192.168.71.140	SCTP	62	SACK
581.646460391	192.168.71.140	192.168.71.132	NGAP/N...	118	UplinkNASTransport
581.847925392	192.168.71.132	192.168.71.140	SCTP	62	SACK
581.847954768	192.168.71.140	192.168.71.132	NGAP/N...	146	UplinkNASTransport
581.878228125	192.168.71.132	192.168.71.140	NGAP/N...	266	PDUSessionResourceSetupRequest
581.882424381	192.168.71.140	192.168.71.132	NGAP	122	PDUSessionResourceSetupResponse

Figure 46: Moment when the UE is connected to the CN

In the Figure 47 it can be seen that until second 369 there is no traffic at all, which is normal since the gNB connects at that moment. On the other hand, the UE connects at second 580 and a peak can be seen in the graph that appears because at that moment most of the messages are exchanged between the gNB and AMF for the correct connection of the UE.

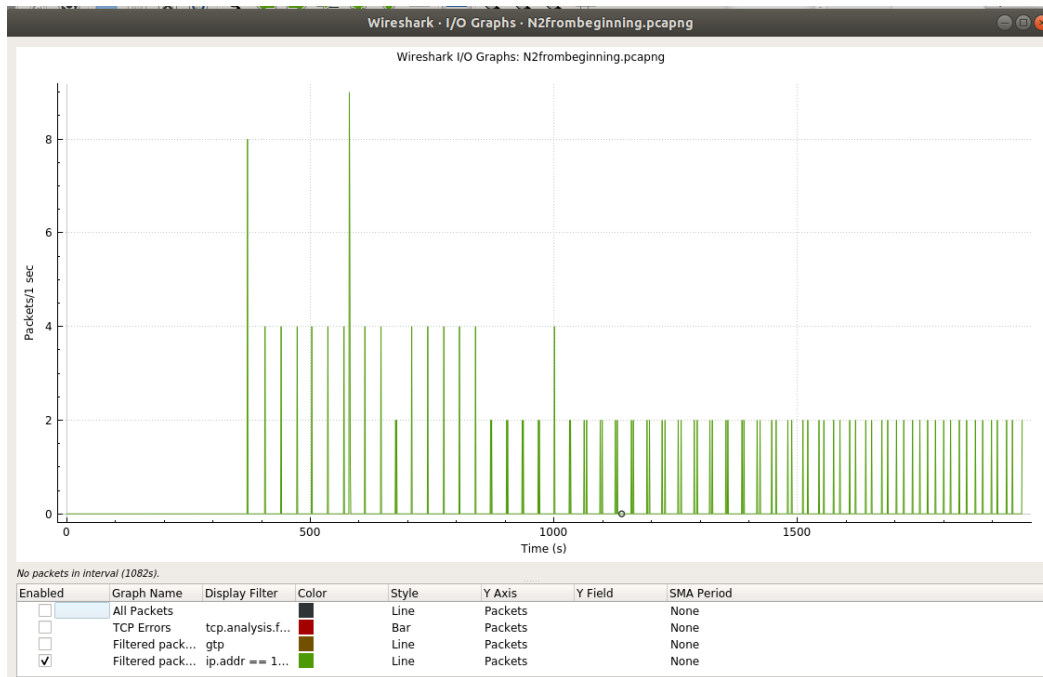


Figure 47: Packets/sec on the N2 interface

From Figure 48 it can be seen that most of the packets are between 40 and 159 bytes in size. This means that the packets exchanged are small in size, which means that no large resources would be needed for proper transmission on this link.

Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
Packet Lengths	419	112,36	42	2088	0,0002	100%	0,4600	581,848
0-19	0	-	-	-	0,0000	0,00%	-	-
20-39	0	-	-	-	0,0000	0,00%	-	-
40-79	142	52,31	42	74	0,0001	33,89%	0,3500	581,848
80-159	222	99,68	82	147	0,0001	52,98%	0,0900	580,581
160-319	45	190,80	160	308	0,0000	10,74%	0,0800	580,594
320-639	5	503,00	381	630	0,0000	1,19%	0,0300	580,605
640-1279	3	981,33	844	1104	0,0000	0,72%	0,0300	581,850
1280-2559	2	1739,00	1390	2088	0,0000	0,48%	0,0200	580,595
2560-5119	0	-	-	-	0,0000	0,00%	-	-
5120 and greater	0	-	-	-	0,0000	0,00%	-	-

Figure 48: Packet size on the N2 interface

As can be seen in the Figure 49, the messages exchanged between the AMF and the gNB are very few and of small size. With an average of 60 bits per second.

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
192.168.71.129	224.0.0.251	12	1.774	12	1.774	0	0	4.309778	1505,4438	9	0
192.168.71.129	192.168.71.191	6	552	6	552	0	0	992.824713	5.0122	881	0
192.168.71.130	192.168.71.132	10	1.757	4	1.202	6	555	581.849229	0.0005	-	-
192.168.71.131	192.168.71.132	34	6.122	16	3.974	18	2.148	580.581074	0.0326	974k	526k
192.168.71.132	192.168.71.140	218	24k	109	13k	109	11k	371.789556	1590.1944	57	55
192.168.71.132	192.168.71.133	30	4.609	16	2.354	14	2.255	581.850463	0.0376	500k	479k

Figure 49: Traffic flow in N2 interface

Once the UE is successfully connected, only SCTP messages with the characteristics of Figure 50 are sent.

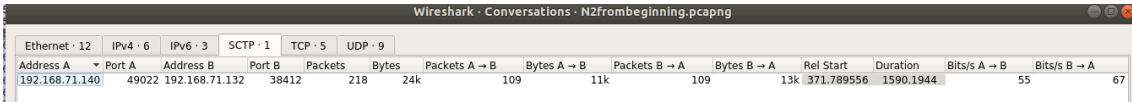


Figure 50: SCTP protocol traffic flow in N2 interface

SCTP messages are the only traffic after the NGAP messages have been sent. The SCTP protocol sends a HEARTBEAT message to the destination to monitor the accessibility of the destination address. In this case it is the AMF block that sends it to the gNB, which replies with a HEARTBEAT_ACK.

As can be seen in the Figure 51, 2 HEARTBEAT and 2 HEARTBEAT_ACK messages are sent every 30 seconds.

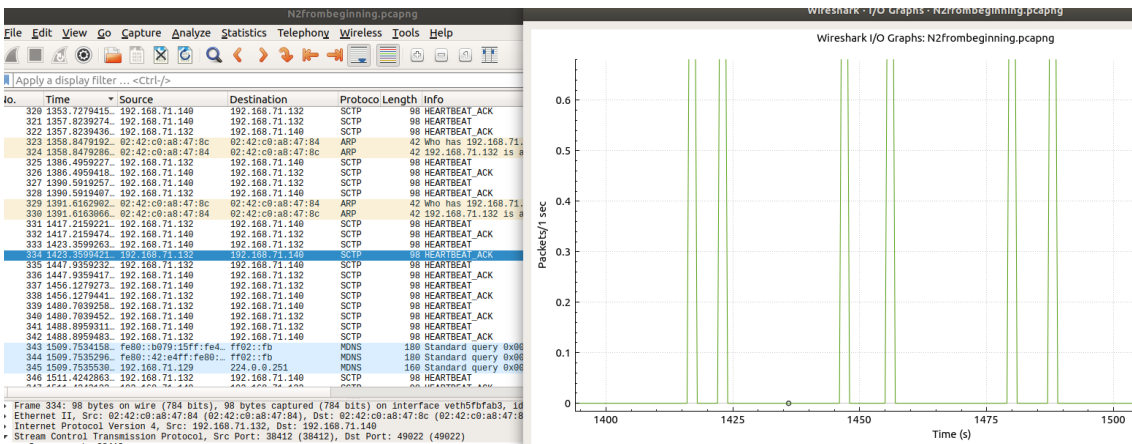


Figure 51: N2 interface HEARTBEAT messages

4.2.1.3. N3 interface

In this interface, without iperf traffic creation, there is no traffic at all. So the following traffic shown in Figure 52 has been created using the next command: `iperf -c 12.1.1.2 -u -i 1 -t 10 -b 1M`

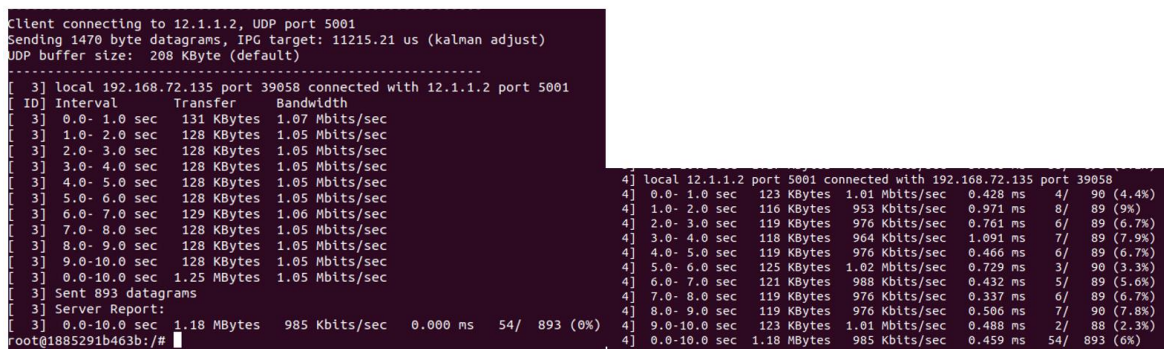


Figure 52: Iperf traffic created on the N3 interface

As can be seen in the Figure 52, 1Mb/s is sent for 10 seconds. The server figure on the right shows that 6% of the total datagrams were lost, which may be due to the high traffic on the last Uu interface.

In Figure 53 it is possible to see the moment when the iperf traffic is created. The packets in green correspond to the filtered traffic for the created traffic, and in black considering all the traffic, as there is some control traffic between the NRF, SMF and UPF.

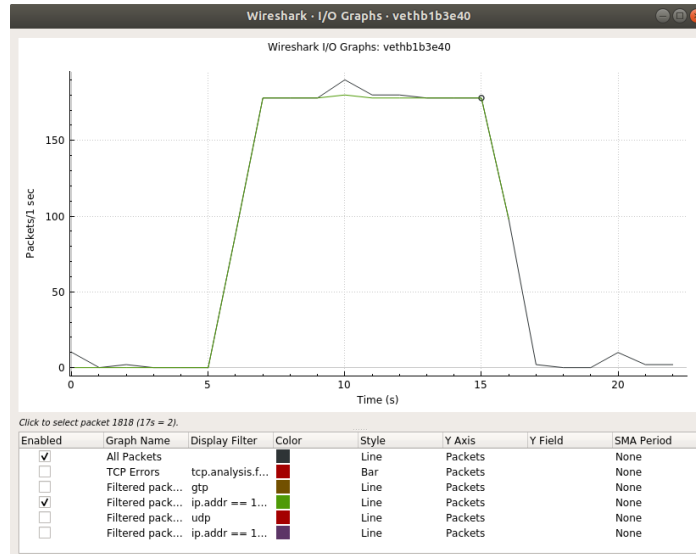


Figure 53: Packets/sec on N3 interface

In Figure 54 it can be seen that during the 10 seconds that the Iperf transmission lasts, around 1.132 kb/s are transmitted from 192.168.71.134 to 192.168.71.140, and 54 kb/s from 192.168.72.135 to 12.1.1.2, which is all the traffic created.

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
12.1.1.2	192.168.72.135	895	68k	1	68	894	67k	6.512715	10.0373	54	54k
192.168.71.130	192.168.71.134	30	2.997	12	1.107	18	1.890	0.000000	20.0005	442	755
192.168.71.133	192.168.71.134	6	348	3	174	3	174	2.184481	20.0008	69	69
192.168.71.134	192.168.71.140	1.790	1.423k	1.788	1.421k	2	1.582	6.512704	10.0373	1.132k	1.260

Figure 54: N3 interface traffic flow

As for the size of the packets shown in Figure 55, it can be noticed that most of them are 75 or 1514 bytes length, which correspond to the Iperf traffic created.

Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
Packet Lengths	1832	778,73	42	1514	0,0826	100%	0,2800	9,911
0-19	0	-	-	-	0,0000	0,00%	-	-
20-39	0	-	-	-	0,0000	0,00%	-	-
40-79	931	75,45	42	76	0,0420	50,82%	0,1700	9,911
80-159	0	-	-	-	0,0000	0,00%	-	-
160-319	6	227,50	163	292	0,0003	0,33%	0,0200	0,000
320-639	0	-	-	-	0,0000	0,00%	-	-
640-1279	0	-	-	-	0,0000	0,00%	-	-
1280-2559	895	1514,00	1514	1514	0,0403	48,85%	0,1000	13,535
2560-5119	0	-	-	-	0,0000	0,00%	-	-
5120 and greater	0	-	-	-	0,0000	0,00%	-	-

Figure 55: Packet length on the N3 interface

The packets are fragmented and reassembled as the MTU is by default 1500 in this case, so if it exceeds this limit, it will be fragmented. Each network element will reassemble the IP packet to get the full data and then establish a GTP tunnel to send the data to the next network element. IP header and data is tunnelled by GTP. When there are a lot of packets, these are fragmented and reassembled, consuming a lot of the kernel's IP resources and harming the network throughput. In the case of Figure 56, the traffic that has been captured in Wireshark has had to be fragmented.

No.	Time	Source	Destination	Protocol	Length	Info
15	6.031054113	192.168.71.134	192.168.71.140	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=49f9) [Reassembled in #16]
16	6.031061483	192.168.72.135	12.1.1.2	GTP <U..	76	39058 - 5001 Len=1470
17	6.042361349	192.168.71.134	192.168.71.140	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=49fa) [Reassembled in #18]
18	6.042367254	192.168.72.135	12.1.1.2	GTP <U..	76	39058 - 5001 Len=1470
19	6.053501995	192.168.71.134	192.168.71.140	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=49fc) [Reassembled in #20]
20	6.053509161	192.168.72.135	12.1.1.2	GTP <U..	76	39058 - 5001 Len=1470
21	6.064740336	192.168.71.134	192.168.71.140	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=49fd) [Reassembled in #22]
22	6.064748049	192.168.72.135	12.1.1.2	GTP <U..	76	39058 - 5001 Len=1470
23	6.075960409	192.168.71.134	192.168.71.140	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=49fe) [Reassembled in #24]
24	6.075967205	192.168.72.135	12.1.1.2	GTP <U..	76	39058 - 5001 Len=1470
25	6.087161509	192.168.71.134	192.168.71.140	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=49ff) [Reassembled in #26]
26	6.087168677	192.168.72.135	12.1.1.2	GTP <U..	76	39058 - 5001 Len=1470
27	6.098381329	192.168.71.134	192.168.71.140	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=4a01) [Reassembled in #28]
28	6.098387957	192.168.72.135	12.1.1.2	GTP <U..	76	39058 - 5001 Len=1470
29	6.109597546	192.168.71.134	192.168.71.140	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=4a04) [Reassembled in #30]
30	6.109604601	192.168.72.135	12.1.1.2	GTP <U..	76	39058 - 5001 Len=1470
31	6.120823190	192.168.71.134	192.168.71.140	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=4a07) [Reassembled in #32]
32	6.120829388	192.168.72.135	12.1.1.2	GTP <U..	76	39058 - 5001 Len=1470
33	6.132037720	192.168.71.134	192.168.71.140	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=4a08) [Reassembled in #34]
34	6.132044624	192.168.72.135	12.1.1.2	GTP <U..	76	39058 - 5001 Len=1470

Figure 56: Packets detected on the N3 interface

4.2.1.4. N6 interface

On this interface, without creating any external traffic, the only thing that has been found are some pings between the UPF and the EXT-DN, probably to check the correct connection. As shown in the Figure 57 the 42 bytes packets are ARP messages requesting the IP, the 98 bytes packets correspond to ping messages.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.72.135	192.168.72.134	ICMP	98	Echo (ping) request id=0x021a, seq=1/256, ttl=64 (reply in 2)
2	0.000024297	192.168.72.134	192.168.72.135	ICMP	98	Echo (ping) reply id=0x021a, seq=1/256, ttl=64 (request in 1)
3	1.005361853	192.168.72.135	192.168.72.134	ICMP	98	Echo (ping) request id=0x021a, seq=2/512, ttl=64 (reply in 4)
4	1.005387070	192.168.72.134	192.168.72.135	ICMP	98	Echo (ping) reply id=0x021a, seq=2/512, ttl=64 (request in 3)
5	11.161487754	192.168.72.135	192.168.72.134	ICMP	98	Echo (ping) request id=0x021b, seq=1/256, ttl=64 (reply in 6)
6	11.161520929	192.168.72.134	192.168.72.135	ICMP	98	Echo (ping) reply id=0x021b, seq=1/256, ttl=64 (request in 5)
7	12.173364366	192.168.72.135	192.168.72.134	ICMP	98	Echo (ping) request id=0x021b, seq=2/512, ttl=64 (reply in 8)
8	12.173387608	192.168.72.134	192.168.72.135	ICMP	98	Echo (ping) reply id=0x021b, seq=2/512, ttl=64 (request in 7)
9	22.380643034	192.168.72.135	192.168.72.134	ICMP	98	Echo (ping) request id=0x021c, seq=1/256, ttl=64 (reply in 10)
10	22.380666011	192.168.72.134	192.168.72.135	ICMP	98	Echo (ping) reply id=0x021c, seq=1/256, ttl=64 (request in 9)
11	23.405361234	192.168.72.135	192.168.72.134	ICMP	98	Echo (ping) request id=0x021c, seq=2/512, ttl=64 (reply in 12)
12	23.405380271	192.168.72.134	192.168.72.135	ICMP	98	Echo (ping) reply id=0x021c, seq=2/512, ttl=64 (request in 11)
13	28.589340203	02:42:c0:a8:48:87	02:42:c0:a8:48:86	ARP	42	Who has 192.168.72.134? Tell 192.168.72.135
14	28.589361229	02:42:c0:a8:48:86	02:42:c0:a8:48:87	ARP	42	192.168.72.134 is at 02:42:c0:a8:48:86
15	33.589321381	192.168.72.135	192.168.72.134	ICMP	98	Echo (ping) request id=0x021d, seq=1/256, ttl=64 (reply in 16)
16	33.589355474	192.168.72.134	192.168.72.135	ICMP	98	Echo (ping) reply id=0x021d, seq=1/256, ttl=64 (request in 15)
17	34.605360855	192.168.72.135	192.168.72.134	ICMP	98	Echo (ping) request id=0x021e, seq=2/512, ttl=64 (reply in 18)
18	34.605385736	192.168.72.134	192.168.72.135	ICMP	98	Echo (ping) reply id=0x021e, seq=2/512, ttl=64 (request in 17)
19	38.829353071	02:42:c0:a8:48:86	02:42:c0:a8:48:87	ARP	42	Who has 192.168.72.135? Tell 192.168.72.134
20	38.829360485	02:42:c0:a8:48:87	02:42:c0:a8:48:86	ARP	42	192.168.72.135 is at 02:42:c0:a8:48:86
21	44.729309403	192.168.72.135	192.168.72.134	ICMP	98	Echo (ping) request id=0x021e, seq=1/256, ttl=64 (reply in 22)
22	44.729340700	192.168.72.134	192.168.72.135	ICMP	98	Echo (ping) reply id=0x021e, seq=1/256, ttl=64 (request in 21)
23	45.741363907	192.168.72.135	192.168.72.134	ICMP	98	Echo (ping) request id=0x021e, seq=2/512, ttl=64 (reply in 24)
24	45.741387754	192.168.72.134	192.168.72.135	ICMP	98	Echo (ping) reply id=0x021e, seq=2/512, ttl=64 (request in 23)
25	55.921190094	192.168.72.135	192.168.72.134	ICMP	98	Echo (ping) request id=0x021f, seq=1/256, ttl=64 (reply in 26)

Figure 57: Traffic captured at the N6 interface

As can be seen in the Figure 58, the traffic on this interface is very small. Only 179 bits/s in each direction, without having created any traffic.

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
02:42:c0:a8:48:86	02:42:c0:a8:48:87	20	1.736	10	868	10	868	0.000000	38.7729	179	179

Figure 58: N6 interface traffic flow

As for the number and size of packets, the Figure 59 shows that 4 packets correspond to ARP messages and 16 packets to ping messages.

Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
Packet Lengths	20	86,80	42	98	0,0005	100%	0,0200	0,000
0-19	0	-	-	-	0,0000	0,00%	-	-
20-39	0	-	-	-	0,0000	0,00%	-	-
40-79	4	42,00	42	42	0,0001	20,00%	0,0200	16,245
80-159	16	98,00	98	98	0,0004	80,00%	0,0200	0,000
160-319	0	-	-	-	0,0000	0,00%	-	-
320-639	0	-	-	-	0,0000	0,00%	-	-
640-1279	0	-	-	-	0,0000	0,00%	-	-
1280-2559	0	-	-	-	0,0000	0,00%	-	-
2560-5119	0	-	-	-	0,0000	0,00%	-	-
5120 and greater	0	-	-	-	0,0000	0,00%	-	-

Figure 59: N6 interface packet length

Then, 3Mb/s of traffic UDP will be created for 10 seconds. The graph of packets through the interface can be seen in the Figure 60. It can be seen to change from almost zero traffic, to just over 250 packets/second.

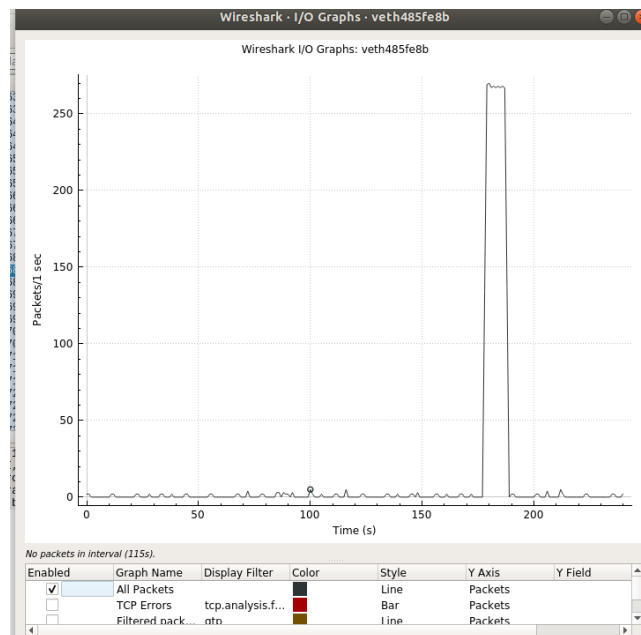


Figure 60: Packets/sec when creating iperf traffic on the N6 interface

In this case, out of 2676 datagrams, only 17 were lost, the 0.64%.

```
3] 0.0-10.0 sec 3.73 MBytes 3.13 Mbits/sec 0.329 ms 17/ 2676 (0.64%)
```

Figure 61: Result of iperf traffic created

Regarding the traffic passing through this N6 interface, a traffic of 3229kb/s has been detected for 10 seconds, with a total of 4049 kB sent, as shown in Figure 62. Which is coherent with the traffic generated.

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
12.1.1.2	192.168.72.135	2,678	4,049k	1	1,512	2,677	4,047k	3,590539	10,0257	1,206	3,229k

Figure 62: N6 iperf traffic flow

Calculating 2678 packets as shown in Figure 63 multiplied by 1512 bytes per packet, gives a total of 4049 kB, giving the same result as above.

Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
Packet Lengths	2690	1505,61	42	1512	0,1976	100%	0,2900	5,086
0-19	0	-	-	-	0,0000	0,00%	-	-
20-39	0	-	-	-	0,0000	0,00%	-	-
40-79	4	42,00	42	42	0,0003	0,15%	0,0200	5,180
80-159	8	98,00	98	98	0,0006	0,30%	0,0200	0,000
160-319	0	-	-	-	0,0000	0,00%	-	-
320-639	0	-	-	-	0,0000	0,00%	-	-
640-1279	0	-	-	-	0,0000	0,00%	-	-
1280-2559	2678	1512,00	1512	1512	0,1967	99,55%	0,2800	11,928
2560-5119	0	-	-	-	0,0000	0,00%	-	-
5120 and greater	0	-	-	-	0,0000	0,00%	-	-

Figure 63: N6 packet lengths

4.2.1.5. Other results

SQL-AMF

To see the communication between the SQL and the AMF go to appendix 2. This appendix shows the communication between the SQL server and the AMF. In order for the NR-UE to connect to the core network, the entry in the SQL database must be created. In the appendix can be seen the user login request and the handshake for the exchange of keys and certificates.

Comparison of Uu, N3 and N6 interfaces

Next, 3Mb/s iperf traffic was created for 10 seconds, and analysed on the N6, N3 and Uu interfaces at the same time. For the graphs shown in Figures 64, 65 and 66, it should be noted that traffic from 192.168.72.135 to 12.1.1.2 has been captured.

In the case of the N6 interface in the Figure 64, it is seen that just over 250 packets/s are passing through at a steady rate.

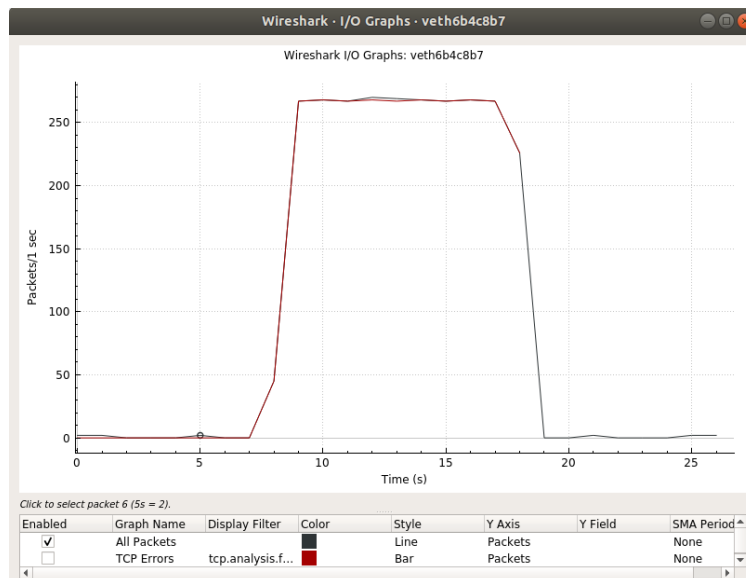


Figure 64: N6 iperf analysis

For the N3 interface, the same packets are observed as on N6, just over 250 packets/s at a steady rate as well, as illustrated in Figure 65.

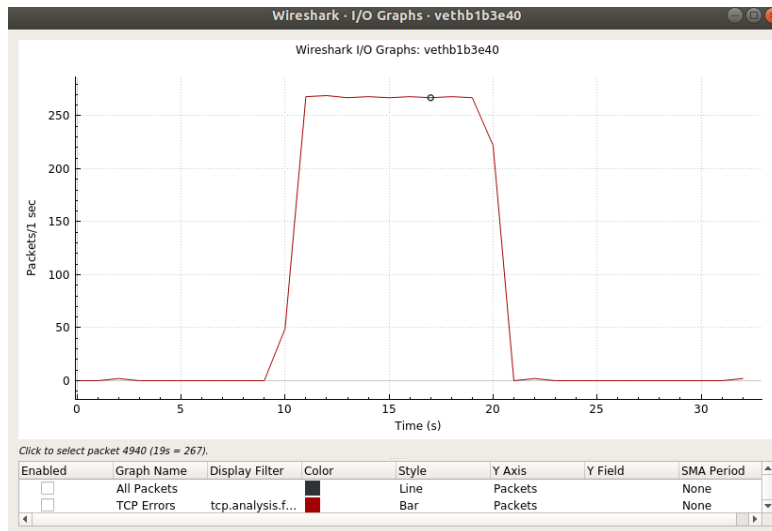


Figure 65: N3 iperf analysis

In the case of the Uu interface, only a maximum of 50 packets/s is seen to pass through, with irregular ups and downs, so more than 200 are lost. This is due to the fact that, as mentioned above, between the gNB and the UE, the RF Simulator is running, and this generates about 900Mb/s both for DL and UL. Figure 66 shows the above described.

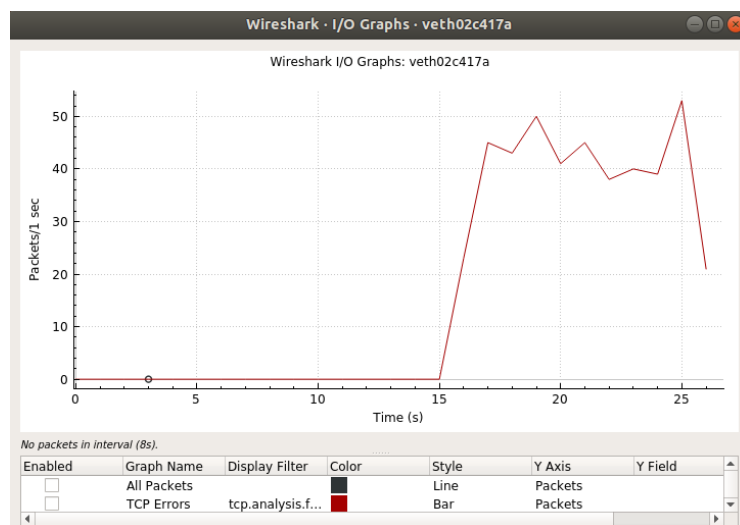


Figure 66: Uu iperf analysis

Analysing the results obtained in Figure 67, it can be said that on the N6 interface, a data rate of 3,232 kb/s has been obtained from 192.168.72.135 to 12.1.1.2. To analyse the data of the Uu and N3 interfaces, it is necessary to take into account Figure 56, in which it can be seen that the packets have been fragmented on the N3 interface. Thus, on the N3 interface, a speed of 162kb/s was obtained for GTP-U protocol traffic and 3,399 kb/s for IPv4 traffic. Finally, on the Uu interface, 26 kb/s was obtained for GTP-U traffic and 557 kb/s for IPv4 traffic. The packet loss on the Uu interface is proportional for packets going from 192.168.72.135 to 12.1.1.2, than for those going from 192.168.71.134 to

192.168.71.140. Since dividing 3.399k by 557k equals 6.1 and dividing 162k by 26k equals 6.2.

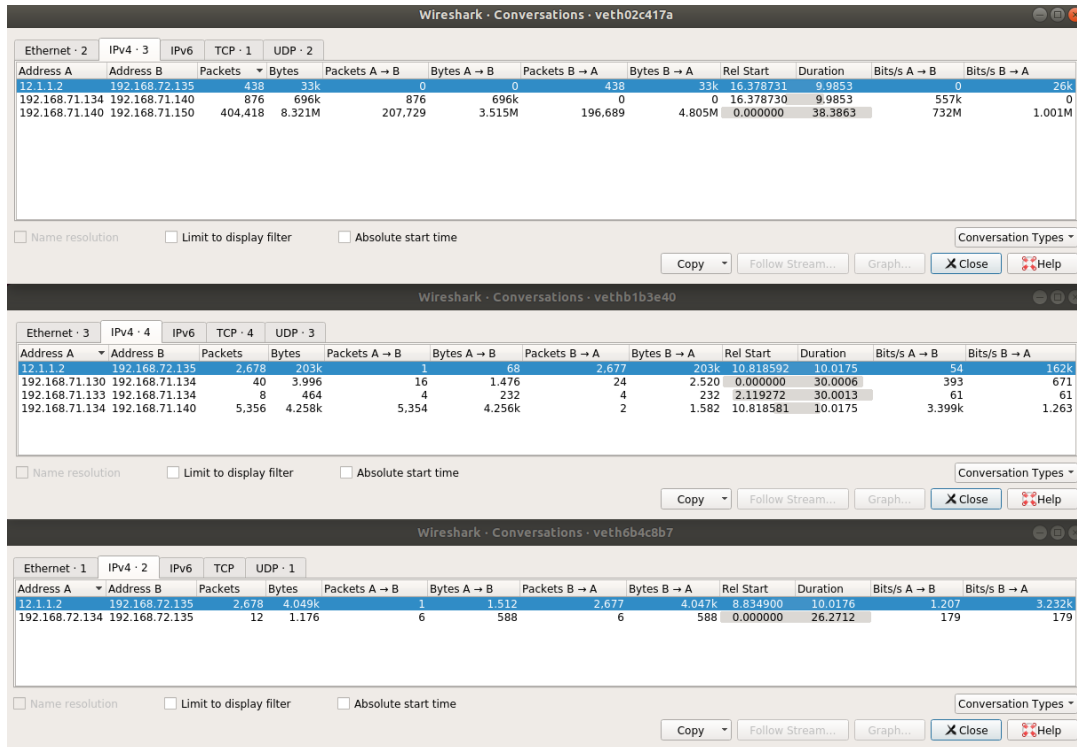


Figure 67: Comparison of iperf traffic from top to bottom: Uu, N3 and N6

4.2.2. Test with Restrictions

In this next section, some limitations will be added to see how the network will react. For this purpose, an SXC1280 device has been taken as a reference. This transceiver provides ultra long range communication in the 2.4 GHz band. Assuming that 2 chips are used for Tx and 2 for Rx, the maximum data rate in each mode can be 4Mbit/s in FSK Mode. So tests will be done limiting to this speed, and even going down to 125 kb/s. The Figure 68 shows the possible data rate for each chip.

Symbol	Raw Bitrate Rb [Mb/s]	Bandwidth BW [MHz DSB]	Sensitivity [dBm]
FSK_BR_2_000_BW_2_4	2.0	2.4	-83
FSK_BR_1_600_BW_2_4	1.6	2.4	-84
FSK_BR_1_000_BW_2_4	1.0	2.4	-87
FSK_BR_1_000_BW_1_2	1.0	1.2	-88
FSK_BR_0_800_BW_2_4	0.8	2.4	-87
FSK_BR_0_800_BW_1_2	0.8	1.2	-89
FSK_BR_0_500_BW_1_2	0.5	1.2	-90
FSK_BR_0_500_BW_0_6	0.5	0.6	-89
FSK_BR_0_400_BW_1_2	0.4	1.2	-91
FSK_BR_0_400_BW_0_6	0.4	0.6	-90
FSK_BR_0_250_BW_0_6	0.25	0.6	-92
FSK_BR_0_250_BW_0_3	0.25	0.3	-93
FSK_BR_0_125_BW_0_3	0.125	0.3	-95

Figure 68: Possible SXC1280 data rate values

The following command format shall be used to set the limitations:

```
tc qdisc add dev eth0 root tbf rate "X"mbit latency "Z"ms burst "Y"kb
```

Data rate is limited to "X" Mbps in the eth0 interface, burst up to "Y" kbit to be sent at max rate. Packets over "Z" ms due to rate limitation are dropped.

Appendix 3 shows the ping times obtained before the limitations, which are very low.

4.2.2.1. N2 interface with limitations

To start with, a limitation of 4Mb/s is set using the next command:

```
tc qdisc add dev eth0 root tbf rate 4.0mbit latency 500ms burst 256kb.
```

Without any limitation the ping below would be less than 1 ms as it can be seen in the appendix 3, as the gNB and AMF are being simulated on the same network. But in this case by limiting the data rate to 4Mb/s, the ping goes to 18ms as shown in Figure 69, but it still works.

```
root@bbde1bde4f5d:/opt/oai-gnb# tc qdisc add dev eth0 root tbf rate 4.0mbit latency 500ms burst 256kb
root@bbde1bde4f5d:/opt/oai-gnb# ping 192.168.71.132 -c 5
PING 192.168.71.132 (192.168.71.132) 56(84) bytes of data:
64 bytes from 192.168.71.132: icmp_seq=1 ttl=64 time=12.4 ms
64 bytes from 192.168.71.132: icmp_seq=2 ttl=64 time=23.8 ms
64 bytes from 192.168.71.132: icmp_seq=3 ttl=64 time=19.9 ms
64 bytes from 192.168.71.132: icmp_seq=4 ttl=64 time=17.9 ms
64 bytes from 192.168.71.132: icmp_seq=5 ttl=64 time=16.4 ms

--- 192.168.71.132 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 12.422/18.133/23.842/3.780 ms
```

Figure 69: Ping on N2 with 4 Mb/s limitation

The following Figure 70 displays how in the second 84 the ping on N2 is detected, while the traffic on the N2 link continues to work without any problem. It is worth noting that the ping messages contain only 98 bytes per packet, so it is not too much of a problem for the interface.

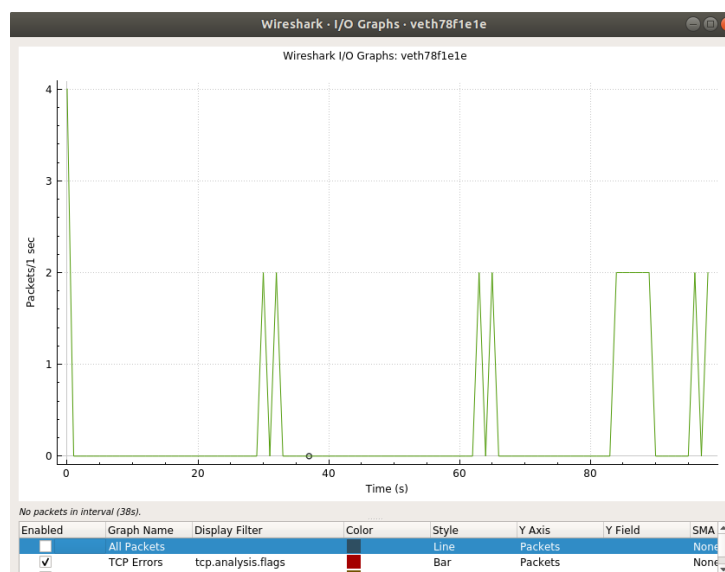


Figure 70: N2 ping Packets/sec graph

In the Figure 71 it can be seen how the AMF correctly receives the ping and the ping response is sent. This assumes that the system is still functioning properly.

No.	Time	Source	Destination	Protocol	Length	Info
3	0.063415657	192.168.71.132	192.168.71.140	SCTP	98	HEARTBEAT_ACK
4	0.063600794	192.168.71.140	192.168.71.132	SCTP	98	HEARTBEAT_ACK
5	5.142209957	02:42:c0:a8:47:8c	02:42:c0:a8:47:84	ARP	42	Who has 192.168.71.132? Tell 192.168.71.140
6	5.142214851	02:42:c0:a8:47:84	02:42:c0:a8:47:8c	ARP	42	192.168.71.132 is at 02:42:c0:a8:47:84
7	30.715093274	192.168.71.132	192.168.71.140	SCTP	98	HEARTBEAT
8	30.785310899	192.168.71.140	192.168.71.132	SCTP	98	HEARTBEAT_ACK
9	32.763991341	fe80::eac:40ff:fe5...	ff02::2	ICMPv6	70	Router Solicitation from ee:ac:40:53:88:fc
10	32.834627587	192.168.71.140	192.168.71.132	SCTP	98	HEARTBEAT
11	32.834639355	192.168.71.132	192.168.71.140	SCTP	98	HEARTBEAT_ACK
12	63.483989315	192.168.71.132	192.168.71.140	SCTP	98	HEARTBEAT
13	63.517210013	192.168.71.140	192.168.71.132	SCTP	98	HEARTBEAT_ACK
14	65.568171369	192.168.71.140	192.168.71.132	SCTP	98	HEARTBEAT
15	65.568186563	192.168.71.132	192.168.71.140	SCTP	98	HEARTBEAT_ACK
16	68.648941488	02:42:c0:a8:47:8c	02:42:c0:a8:47:84	ARP	42	Who has 192.168.71.132? Tell 192.168.71.140
17	68.648948924	02:42:c0:a8:47:84	02:42:c0:a8:47:8c	ARP	42	192.168.71.132 is at 02:42:c0:a8:47:84
18	70.651987131	02:42:c0:a8:47:84	02:42:c0:a8:47:8c	ARP	42	Who has 192.168.71.140? Tell 192.168.71.132
19	70.733888899	02:42:c0:a8:47:8c	02:42:c0:a8:47:84	ARP	42	192.168.71.140 is at 02:42:c0:a8:47:8c
20	84.262525961	192.168.71.140	192.168.71.132	ICMP	98	Echo (ping) request id=0x00c6, seq=1/256
21	84.262538566	192.168.71.132	192.168.71.140	ICMP	98	Echo (ping) reply id=0x00c6, seq=1/256
22	85.283950664	192.168.71.140	192.168.71.132	ICMP	98	Echo (ping) request id=0x00c6, seq=2/512
23	85.283961054	192.168.71.132	192.168.71.140	ICMP	98	Echo (ping) reply id=0x00c6, seq=2/512
24	86.288922740	192.168.71.140	192.168.71.132	ICMP	98	Echo (ping) request id=0x00c6, seq=3/768
25	86.288933935	192.168.71.132	192.168.71.140	ICMP	98	Echo (ping) reply id=0x00c6, seq=3/768
26	87.279463180	192.168.71.140	192.168.71.132	ICMP	98	Echo (ping) request id=0x00c6, seq=4/1024
27	87.279473871	192.168.71.132	192.168.71.140	ICMP	98	Echo (ping) reply id=0x00c6, seq=4/1024
28	88.282679349	192.168.71.140	192.168.71.132	ICMP	98	Echo (ping) request id=0x00c6, seq=5/1280
29	88.282689910	192.168.71.132	192.168.71.140	ICMP	98	Echo (ping) reply id=0x00c6, seq=5/1280
30	89.298450613	192.168.71.140	192.168.71.132	ICMP	98	Echo (ping) request id=0x00c6, seq=6/1536
31	89.298460584	192.168.71.132	192.168.71.140	ICMP	98	Echo (ping) reply id=0x00c6, seq=6/1536
32	90.251996028	192.168.71.132	192.168.71.140	SCTP	98	HEARTBEAT
33	90.300363286	192.168.71.140	192.168.71.132	SCTP	98	HEARTBEAT_ACK
34	90.345069388	192.168.71.140	192.168.71.132	SCTP	98	HEARTBEAT
35	90.345024129	192.168.71.132	192.168.71.140	SCTP	98	HEARTBEAT_ACK
36	101.406945680	02:42:c0:a8:47:8c	02:42:c0:a8:47:84	ARP	42	Who has 192.168.71.132? Tell 192.168.71.140
37	101.406952944	02:42:c0:a8:47:84	02:42:c0:a8:47:8c	ARP	42	192.168.71.132 is at 02:42:c0:a8:47:84

Figure 71: Ping detected in N2 interface

Then a limitation of 2 Mb/s is set, giving the following results: In this case the ping goes up to 32 seconds, but is still acceptable. So, depending on the type of service required, it will be valid or not. Figure 72 shows the ping obtained.

```
root@bbde1bde4f5d:/opt/oai-gnb# tc qdisc add dev eth0 root tbf rate 2.0mbit latency 500ms burst 256kb
root@bbde1bde4f5d:/opt/oai-gnb# ping 192.168.71.132 -c 5
PING 192.168.71.132 (192.168.71.132) 56(84) bytes of data.
64 bytes from 192.168.71.132: icmp_seq=1 ttl=64 time=43.3 ms
64 bytes from 192.168.71.132: icmp_seq=2 ttl=64 time=45.3 ms
64 bytes from 192.168.71.132: icmp_seq=3 ttl=64 time=36.5 ms
64 bytes from 192.168.71.132: icmp_seq=4 ttl=64 time=32.9 ms
64 bytes from 192.168.71.132: icmp_seq=5 ttl=64 time=4.72 ms

--- 192.168.71.132 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 4.724/32.581/45.323/14.635 ms
```

Figure 72: Ping on N2 with 2 Mb/s limitation

The data rate in this next case will be limited to 1 Mb/s, resulting in 40 ms the average ping, as illustrated in Figure 73.

```
root@bbde1bde4f5d:/opt/oai-gnb# tc qdisc add dev eth0 root tbf rate 1.0mbit latency 100ms burst 64kb
root@bbde1bde4f5d:/opt/oai-gnb# ping 192.168.71.132 -c 5
PING 192.168.71.132 (192.168.71.132) 56(84) bytes of data.
64 bytes from 192.168.71.132: icmp_seq=1 ttl=64 time=51.4 ms
64 bytes from 192.168.71.132: icmp_seq=2 ttl=64 time=34.6 ms
64 bytes from 192.168.71.132: icmp_seq=3 ttl=64 time=34.8 ms
64 bytes from 192.168.71.132: icmp_seq=4 ttl=64 time=35.8 ms
64 bytes from 192.168.71.132: icmp_seq=5 ttl=64 time=43.7 ms

--- 192.168.71.132 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4002ms
rtt min/avg/max/mdev = 34.605/40.110/51.467/6.607 ms
```

Figure 73: Ping on N2 with 1 Mb/s limitation

The data rate limitation is then lowered to 500 kb/s, with an average result of 86 ms. This result can be seen in Figure 74.

```
$ docker exec -it rfsim5g-oai-gnb /bin/bash
root@4ea38f441804:/opt/oai-gnb# tc qdisc add dev eth0 root tbf rate 500kbit latency 100ms burst 64kb
root@4ea38f441804:/opt/oai-gnb# ping 192.168.71.132 -c 5
PING 192.168.71.132 (192.168.71.132) 56(84) bytes of data.
64 bytes from 192.168.71.132: icmp_seq=1 ttl=64 time=93.7 ms
64 bytes from 192.168.71.132: icmp_seq=2 ttl=64 time=62.5 ms
64 bytes from 192.168.71.132: icmp_seq=3 ttl=64 time=81.1 ms
64 bytes from 192.168.71.132: icmp_seq=4 ttl=64 time=100 ms
64 bytes from 192.168.71.132: icmp_seq=5 ttl=64 time=93.8 ms

--- 192.168.71.132 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 62.571/86.295/100.143/13.368 ms
```

Figure 74: Ping on N2 with 500 kb/s limitation

In the case of limiting the data rate to 250 kb/s ping starts to increase too much, as can be seen in Figure 75 reaching 150 ms on average.

```
root@4ea38f441804:/opt/oai-gnb# tc qdisc add dev eth0 root tbf rate 250kbit latency 500ms burst 64kb
root@4ea38f441804:/opt/oai-gnb# ping 192.168.71.132 -c 5
PING 192.168.71.132 (192.168.71.132) 56(84) bytes of data.
64 bytes from 192.168.71.132: icmp_seq=1 ttl=64 time=173 ms
64 bytes from 192.168.71.132: icmp_seq=2 ttl=64 time=150 ms
64 bytes from 192.168.71.132: icmp_seq=3 ttl=64 time=121 ms
64 bytes from 192.168.71.132: icmp_seq=4 ttl=64 time=140 ms
64 bytes from 192.168.71.132: icmp_seq=5 ttl=64 time=164 ms

--- 192.168.71.132 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 121.216/150.029/173.479/18.221 ms
```

Figure 75: Ping on N2 with 250 kb/s limitation

Now the limitation will be the lowest that the SXC1280 device can have, setting it to 125 kb/s, resulting in an average ping of 316 ms as can be found in Figure 76.

```
root@bbde1bde4f5d:/opt/oai-gnb# ping 192.168.71.132 -c 5
PING 192.168.71.132 (192.168.71.132) 56(84) bytes of data.
64 bytes from 192.168.71.132: icmp_seq=1 ttl=64 time=212 ms
64 bytes from 192.168.71.132: icmp_seq=2 ttl=64 time=380 ms
64 bytes from 192.168.71.132: icmp_seq=3 ttl=64 time=354 ms
64 bytes from 192.168.71.132: icmp_seq=4 ttl=64 time=426 ms
64 bytes from 192.168.71.132: icmp_seq=5 ttl=64 time=206 ms

--- 192.168.71.132 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4003ms
rtt min/avg/max/mdev = 206.839/316.024/426.068/89.848 ms
```

Figure 76: Ping on N2 with 125 kb/s limitation

In the graph below on the Figure 77 can be seen how the ping coincides with the normal messages on the N2, without having any negative impact on them.

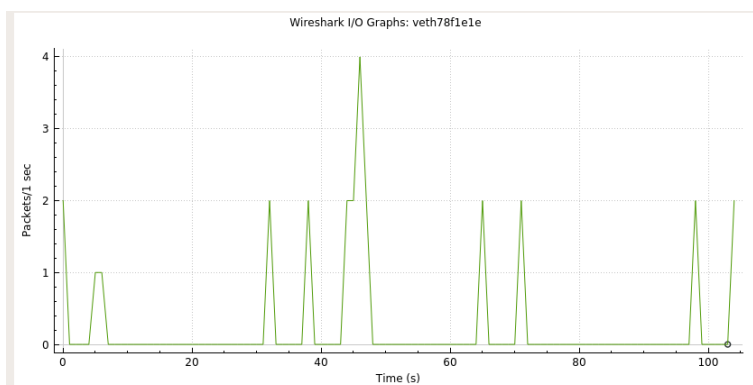


Figure 77: Packets/sec graph in N2 interface with limitation

Then a 50 kb/s limitation was set, which caused the ping to go up to 8388 ms, which is an outrageous amount for a normal application. Figure 78 shows the value obtained.

```
root@4ea38f441804:/opt/oai-gnb# tc qdisc add dev eth0 root tbf rate 50kbit latency 500ms burst 64kb
root@4ea38f441804:/opt/oai-gnb# ping 192.168.71.132 -c 5
PING 192.168.71.132 (192.168.71.132) 56(84) bytes of data.
64 bytes from 192.168.71.132: icmp_seq=1 ttl=64 time=9988 ms
64 bytes from 192.168.71.132: icmp_seq=2 ttl=64 time=9494 ms
64 bytes from 192.168.71.132: icmp_seq=3 ttl=64 time=8486 ms
64 bytes from 192.168.71.132: icmp_seq=4 ttl=64 time=7474 ms
64 bytes from 192.168.71.132: icmp_seq=5 ttl=64 time=6497 ms

--- 192.168.71.132 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4078ms
rtt min/avg/max/mdev = 6497.853/8388.335/9988.816/1281.011 ms, pipe 5
```

Figure 78: Ping on N2 with 50 kb/s limitation

The limit has been lowered to find a value that the ping did not reach, to look for the failure. To do this, the limit has been lowered to 25 kb/s to obtain a ping with no response. As can be clearly seen in Figure 79, if this data rate were used, communication would not be possible.

```
root@4ea38f441804:/opt/oai-gnb# tc qdisc add dev eth0 root tbf rate 25kbit latency 500ms burst 64kb
root@4ea38f441804:/opt/oai-gnb# ping 192.168.71.132 -c 5
PING 192.168.71.132 (192.168.71.132) 56(84) bytes of data.

--- 192.168.71.132 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 7099ms
```

Figure 79: Ping on N2 with 25 kb/s limitation

4.2.2.2. N3 interface with limitations

For this interface the steps of the N2 will be followed, starting with 4 Mb/s and reducing it to the minimum possible.

For this first step, a limit of 4 Mb/s has been set, giving an average ping of 32 seconds, as shown in Figure 80.

```
root@bbde1bde4f5d:/opt/oai-gnb# tc qdisc add dev eth0 root tbf rate 4.0mbit latency 500ms burst 256kb
root@bbde1bde4f5d:/opt/oai-gnb# ping 192.168.71.134 -c 5
PING 192.168.71.134 (192.168.71.134) 56(84) bytes of data.
64 bytes from 192.168.71.134: icmp_seq=1 ttl=64 time=20.5 ms
64 bytes from 192.168.71.134: icmp_seq=2 ttl=64 time=29.3 ms
64 bytes from 192.168.71.134: icmp_seq=3 ttl=64 time=33.1 ms
64 bytes from 192.168.71.134: icmp_seq=4 ttl=64 time=31.1 ms
64 bytes from 192.168.71.134: icmp_seq=5 ttl=64 time=48.9 ms

--- 192.168.71.134 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 20.502/32.617/48.915/9.223 ms
```

Figure 80: Ping on N3 with 4 Mb/s limitation

Then, iperf traffic of 500 kb/s during 20 seconds is created.

The Figure 81 illustrates the ping at second 35, and the iperf traffic from second 75 onwards. The green traffic belongs to the UPF-gNB communication, while the black one is in general, taking into account some control traffic between the NRF, SMF and UPF. So in this N3 interface with this limitation it still works.

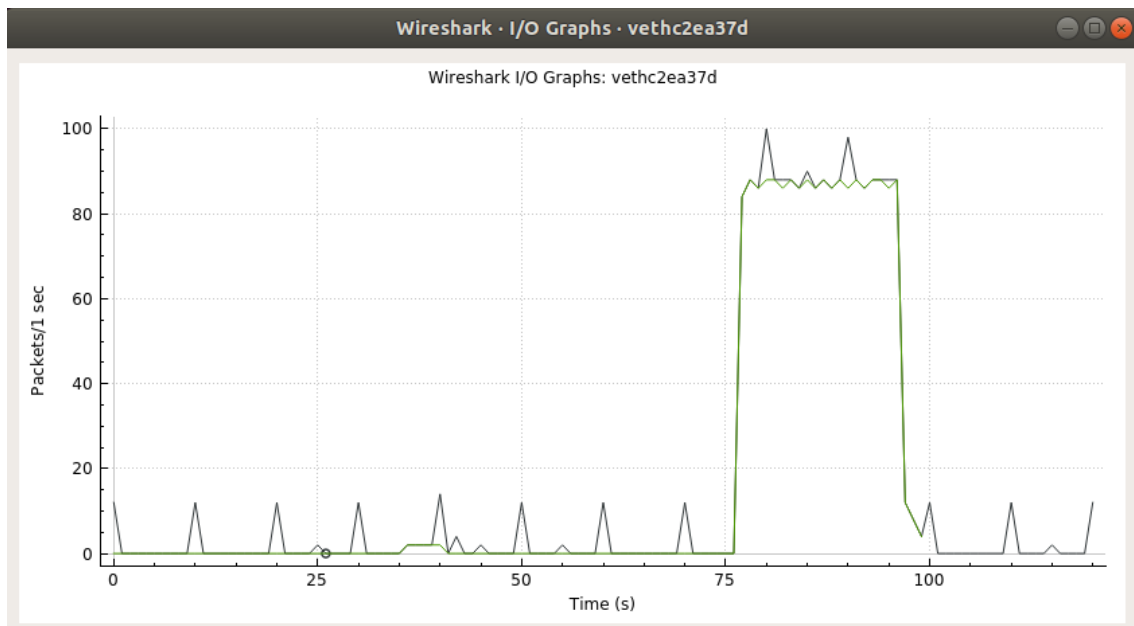


Figure 81: Packets/sec graph in N3 interface with limitation

Next, a limitation of 2 Mb/s is set, resulting in an average ping of 34 seconds, still a very acceptable ping for most applications. The Figure 82 indicates the ping obtained.

```

root@bbde1bde4f5d:/opt/oai-gnb# tc qdisc add dev eth0 root tbf rate 2.0mbit latency 50ms burst 256kb
root@bbde1bde4f5d:/opt/oai-gnb# ping 192.168.71.134 -c 5
PING 192.168.71.134 (192.168.71.134) 56(84) bytes of data.
64 bytes from 192.168.71.134: icmp_seq=1 ttl=64 time=60.3 ms
64 bytes from 192.168.71.134: icmp_seq=2 ttl=64 time=29.1 ms
64 bytes from 192.168.71.134: icmp_seq=3 ttl=64 time=30.5 ms
64 bytes from 192.168.71.134: icmp_seq=4 ttl=64 time=27.2 ms
64 bytes from 192.168.71.134: icmp_seq=5 ttl=64 time=27.3 ms

--- 192.168.71.134 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 27.211/34.927/60.322/12.758 ms

```

Figure 82: Ping on N3 with 2 Mb/s limitation

In this next case it has been limited to 1 Mb/s, obtaining an average ping of 36 ms, as shown in Figure 83, somewhat higher than in the previous case but without much difference.

```

root@4ea38f441804:/opt/oai-gnb# tc qdisc add dev eth0 root tbf rate 1mbit latency 500ms burst 64kb
root@4ea38f441804:/opt/oai-gnb# ping 192.168.71.134 -c 5
PING 192.168.71.134 (192.168.71.134) 56(84) bytes of data.
64 bytes from 192.168.71.134: icmp_seq=1 ttl=64 time=45.0 ms
64 bytes from 192.168.71.134: icmp_seq=2 ttl=64 time=29.0 ms
64 bytes from 192.168.71.134: icmp_seq=3 ttl=64 time=53.5 ms
64 bytes from 192.168.71.134: icmp_seq=4 ttl=64 time=28.6 ms
64 bytes from 192.168.71.134: icmp_seq=5 ttl=64 time=27.3 ms

--- 192.168.71.134 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 27.302/36.721/53.580/10.637 ms

```

Figure 83: Ping on N3 with 1 Mb/s limitation

Now, by limiting it to 500 kb/s the ping has gone up to 82 ms, so depending on the service you want to provide, it could be a very high latency. Figure 84 illustrates the ping performed.


```

root@4ea38f441804:/opt/oai-gnb# tc qdisc add dev eth0 root tbf rate 500kbit latency 500ms burst 64kb
root@4ea38f441804:/opt/oai-gnb# ping 192.168.71.134 -c 5
PING 192.168.71.134 (192.168.71.134) 56(84) bytes of data.
64 bytes from 192.168.71.134: icmp_seq=1 ttl=64 time=70.9 ms
64 bytes from 192.168.71.134: icmp_seq=2 ttl=64 time=96.6 ms
64 bytes from 192.168.71.134: icmp_seq=3 ttl=64 time=65.5 ms
64 bytes from 192.168.71.134: icmp_seq=4 ttl=64 time=84.9 ms
64 bytes from 192.168.71.134: icmp_seq=5 ttl=64 time=95.2 ms

--- 192.168.71.134 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 65.591/82.683/96.687/12.552 ms

```

Figure 84: Ping on N3 with 500 kb/s limitation

In the case of limiting it to 250 kb/s, the previous ping has doubled to 173 ms, as shown in Figure 85.

```

root@4ea38f441804:/opt/oai-gnb# tc qdisc add dev eth0 root tbf rate 250kbit latency 500ms burst 64kb
root@4ea38f441804:/opt/oai-gnb# ping 192.168.71.134 -c 5
PING 192.168.71.134 (192.168.71.134) 56(84) bytes of data.
64 bytes from 192.168.71.134: icmp_seq=1 ttl=64 time=151 ms
64 bytes from 192.168.71.134: icmp_seq=2 ttl=64 time=172 ms
64 bytes from 192.168.71.134: icmp_seq=3 ttl=64 time=143 ms
64 bytes from 192.168.71.134: icmp_seq=4 ttl=64 time=214 ms
64 bytes from 192.168.71.134: icmp_seq=5 ttl=64 time=185 ms

--- 192.168.71.134 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4003ms
rtt min/avg/max/mdev = 143.237/173.421/214.079/25.198 ms

```

Figure 85: Ping on N3 with 250 kb/s limitation

In the following case it is limited to 125 kb/s, giving an average ping of 301 ms. Figure 86 shows the result obtained.

```

root@bbde1bde4f5d:/opt/oai-gnb# ping 192.168.71.134 -c 5
PING 192.168.71.134 (192.168.71.134) 56(84) bytes of data.
64 bytes from 192.168.71.134: icmp_seq=1 ttl=64 time=353 ms
64 bytes from 192.168.71.134: icmp_seq=2 ttl=64 time=230 ms
64 bytes from 192.168.71.134: icmp_seq=3 ttl=64 time=204 ms
64 bytes from 192.168.71.134: icmp_seq=4 ttl=64 time=372 ms
64 bytes from 192.168.71.134: icmp_seq=5 ttl=64 time=346 ms

--- 192.168.71.134 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 204.661/301.517/372.375/69.485 ms

```

Figure 86: Ping on N3 with 125 kb/s limitation

The data rate is then limited to 50 kb/s, resulting in a fairly high average ping of 781 ms. So the system would presumably continue to work, but with a very high latency. Figure 87 illustrates the ping result obtained.

```

root@4ea38f441804:/opt/oai-gnb# tc qdisc add dev eth0 root tbf rate 50kbit latency 500ms burst 64kb
root@4ea38f441804:/opt/oai-gnb# ping 192.168.71.134 -c 5
PING 192.168.71.134 (192.168.71.134) 56(84) bytes of data.
64 bytes from 192.168.71.134: icmp_seq=1 ttl=64 time=811 ms
64 bytes from 192.168.71.134: icmp_seq=2 ttl=64 time=796 ms
64 bytes from 192.168.71.134: icmp_seq=3 ttl=64 time=781 ms
64 bytes from 192.168.71.134: icmp_seq=4 ttl=64 time=766 ms
64 bytes from 192.168.71.134: icmp_seq=5 ttl=64 time=751 ms

--- 192.168.71.134 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3998ms
rtt min/avg/max/mdev = 751.375/781.468/811.767/21.355 ms

```

Figure 87: Ping on N3 with 50 kb/s limitation

For the latter case the data rate has been limited to 25 kb/s and as in the previous interface the ping has not been received back. So, this limit would be the limit. Figure 88 reveals how the ping has not received a response.

```

root@4ea38f441804:/opt/oai-gnb# tc qdisc add dev eth0 root tbf rate 25kbit latency 500ms burst 64kb
root@4ea38f441804:/opt/oai-gnb# ping 192.168.71.134 -c 5
PING 192.168.71.134 (192.168.71.134) 56(84) bytes of data.

--- 192.168.71.134 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4080ms

```

Figure 88: Ping on N3 with 25 kb/s limitation

The table 4 shows the comparison of latency times between the N2 and N3 interface under the same constraint. As can be seen, the latencies are quite similar between the 2 interfaces.

Data Rate limitation	N2 latency	N3 latency
4 Mb/s	18 ms	32 ms
2 Mb/s	32 ms	34 ms
1 Mb/s	40 ms	36 ms
500 kb/s	86 ms	82 ms
250 kb/s	150 ms	173 ms
125 kb/s	316 ms	301 ms
50 kb/s	8388 ms	780 ms
25 kb/s	no reply	no reply

Table 4: N2 and N3 interface latency

Figure 89 illustrates the values in the table 4. As can be seen, latency increases at the same rate. The data with 50 kb/s and 25 kb/s have not been added to the graph as they were very high values.

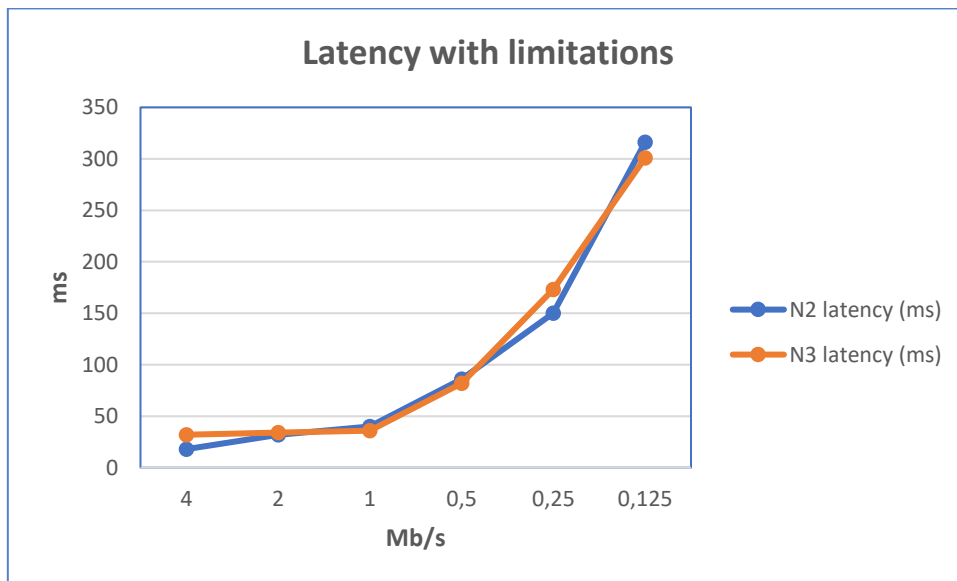


Figure 89: N2 N3 latency comparison

4.2.2.3 Uu interface

Next, a limitation of 4 Mb/s has been established in the gNB and the behaviour of the Uu interface has been analysed, taking into account that there was a lot of traffic in this interface since the RFSimulator is running, obtaining the following results:

The Figure 90 and 91 show how a 500 kb/s traffic is created for 5 seconds from the EXT-DN to the UE. In total 314 kB are sent, of which only 274 kB arrive on the receiver side

when analysed. This is due to the fact that the limitation given to the gNB is applied to the communication with the UE and there is congestion and errors. As can be seen in the Figure 91, the jitter was 88 ms, which is quite high.

```

root@48bbde5099cc:/# iperf -c 12.1.1.2 -u -i 1 -t 5 -b 500K
-----
Client connecting to 12.1.1.2, UDP port 5001
Sending 1470 byte datagrams, IPG target: 22968.75 us (kalman adjust)
UDP buffer size: 208 KByte (default)
-----
[ 3] local 192.168.72.135 port 48065 connected with 12.1.1.2 port 5001
[ ID] Interval      Transfer      Bandwidth
[ 3] 0.0- 1.0 sec  64.6 KBytes   529 Kbits/sec
[ 3] 1.0- 2.0 sec  63.2 KBytes   517 Kbits/sec
[ 3] 2.0- 3.0 sec  61.7 KBytes   506 Kbits/sec
[ 3] 3.0- 4.0 sec  63.2 KBytes   517 Kbits/sec
[ 3] 4.0- 5.0 sec  61.7 KBytes   506 Kbits/sec
[ 3] 0.0- 5.0 sec   314 KBytes    512 Kbits/sec
[ 3] Sent 219 datagrams
[ 3] WARNING: did not receive ack of last datagram after 10 tries.

```

Figure 90: Iperf TX in Uu interface with 4 Mb/s restrictions

```

Croot@d95de34024a9:/opt/oai-nr-ue# iperf -B 12.1.1.2 -u -i 1 -s
-----
Server listening on UDP port 5001
Binding to local address 12.1.1.2
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)
-----
[ 3] local 12.1.1.2 port 5001 connected with 192.168.72.135 port 48065
[ ID] Interval      Transfer      Bandwidth      Jitter      Lost/Total Datagrams
[ 3] 0.0- 1.0 sec  30.1 KBytes   247 Kbits/sec   37.627 ms    5/ 26 (19%)
[ 3] 1.0- 2.0 sec  15.8 KBytes   129 Kbits/sec   49.183 ms    0/ 11 (0%)
[ 3] 2.0- 3.0 sec  23.0 KBytes   188 Kbits/sec   69.105 ms    0/ 16 (0%)
[ 3] 3.0- 4.0 sec  24.4 KBytes   200 Kbits/sec   79.266 ms    6/ 23 (26%)
[ 3] 4.0- 5.0 sec  12.9 KBytes   106 Kbits/sec   61.321 ms    0/ 9 (0%)
[ 3] 5.0- 6.0 sec  24.4 KBytes   200 Kbits/sec   70.154 ms    0/ 17 (0%)
[ 3] 6.0- 7.0 sec  24.4 KBytes   200 Kbits/sec   79.981 ms    5/ 22 (23%)
[ 3] 7.0- 8.0 sec  28.7 KBytes   235 Kbits/sec   69.194 ms    0/ 20 (0%)
[ 3] 8.0- 9.0 sec  14.4 KBytes   118 Kbits/sec   66.820 ms    0/ 10 (0%)
[ 3] 9.0-10.0 sec  24.4 KBytes   200 Kbits/sec   78.807 ms    6/ 23 (26%)
[ 3] 10.0-11.0 sec 28.7 KBytes   235 Kbits/sec   68.839 ms    0/ 20 (0%)
[ 3] 11.0-12.0 sec 14.4 KBytes   118 Kbits/sec   66.417 ms    0/ 10 (0%)
[ 3] 0.0-12.5 sec  274 KBytes   180 Kbits/sec   87.868 ms   28/ 219 (13%)

```

Figure 91: Iperf RX in Uu interface with 4 Mb/s restrictions

For the following test a limit of 125 kb/s has been set and a traffic of 500 kb/s has been created for 5 seconds. As expected, traffic did not reach the destination and the following results were obtained as shown in Figure 92 and 93.

Although 314 kB of traffic was sent, nothing was received, as it was limited to 125 kb/s and 500 kb/s were sent.

```

Client connecting to 12.1.1.2, UDP port 5001
Sending 1470 byte datagrams, IPG target: 22968.75 us (kalman adjust)
UDP buffer size: 208 KByte (default)
-----
[ 3] local 192.168.72.135 port 57588 connected with 12.1.1.2 port 5001
[ ID] Interval      Transfer      Bandwidth
[ 3] 0.0- 1.0 sec  64.6 KBytes   529 Kbits/sec
[ 3] 1.0- 2.0 sec  63.2 KBytes   517 Kbits/sec
[ 3] 2.0- 3.0 sec  61.7 KBytes   506 Kbits/sec
[ 3] 3.0- 4.0 sec  63.2 KBytes   517 Kbits/sec
[ 3] 4.0- 5.0 sec  61.7 KBytes   506 Kbits/sec
[ 3] 0.0- 5.0 sec   314 KBytes    512 Kbits/sec
[ 3] Sent 219 datagrams
[ 3] WARNING: did not receive ack of last datagram after 10 tries.
root@48bbde5099cc:/#

```

Figure 92: Iperf TX in Uu interface with 125 kb/s restrictions


```

Server listening on UDP port 5001
Binding to local address 12.1.1.2
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)
-----
[ 3] local 12.1.1.2 port 5001 connected with 192.168.72.135 port 57588
[ ID] Interval      Transfer      Bandwidth      Jitter      Lost/Total Datagrams
[ 3]  0.0- 1.0 sec  5.74 KBytes   47.0 Kbits/sec  4.038 ms    0/    4 (0%)
[ 3]  1.0- 2.0 sec  0.00 Bytes    0.00 bits/sec   0.000 ms    0/    0 (0%)
[ 3]  2.0- 3.0 sec  0.00 Bytes    0.00 bits/sec   0.000 ms    0/    0 (0%)
[ 3]  3.0- 4.0 sec  0.00 Bytes    0.00 bits/sec   0.000 ms    0/    0 (0%)
[ 3]  4.0- 5.0 sec  0.00 Bytes    0.00 bits/sec   0.000 ms    0/    0 (0%)
[ 3]  5.0- 6.0 sec  0.00 Bytes    0.00 bits/sec   0.000 ms    0/    0 (0%)
[ 3]  6.0- 7.0 sec  8.61 KBytes   70.6 Kbits/sec 293.591 ms   0/    6 (0%)
[ 3]  7.0- 8.0 sec  0.00 Bytes    0.00 bits/sec   0.000 ms    0/    0 (0%)
[ 3]  8.0- 9.0 sec  0.00 Bytes    0.00 bits/sec   0.000 ms    0/    0 (0%)
[ 3]  9.0-10.0 sec  0.00 Bytes    0.00 bits/sec   0.000 ms    0/    0 (0%)
[ 3] 10.0-11.0 sec  0.00 Bytes    0.00 bits/sec   0.000 ms    0/    0 (0%)
[ 3] 11.0-12.0 sec  0.00 Bytes    0.00 bits/sec   0.000 ms    0/    0 (0%)
[ 3] 12.0-13.0 sec  8.61 KBytes   70.6 Kbits/sec 485.632 ms   0/    6 (0%)
[ 3] 13.0-14.0 sec  0.00 Bytes    0.00 bits/sec   0.000 ms    0/    0 (0%)
[ 3] 14.0-15.0 sec  0.00 Bytes    0.00 bits/sec   0.000 ms    0/    0 (0%)
[ 3] 15.0-16.0 sec  0.00 Bytes    0.00 bits/sec   0.000 ms    0/    0 (0%)
[ 3] 16.0-17.0 sec  0.00 Bytes    0.00 bits/sec   0.000 ms    0/    0 (0%)
[ 3] 17.0-18.0 sec  0.00 Bytes    0.00 bits/sec   0.000 ms    0/    0 (0%)
[ 3] 18.0-19.0 sec  8.61 KBytes   70.6 Kbits/sec 620.269 ms   0/    6 (0%)
[ 3] 19.0-20.0 sec  0.00 Bytes    0.00 bits/sec   0.000 ms    0/    0 (0%)
[ 3] 20.0-21.0 sec  0.00 Bytes    0.00 bits/sec   0.000 ms    0/    0 (0%)
[ 3] 21.0-22.0 sec  0.00 Bytes    0.00 bits/sec   0.000 ms    0/    0 (0%)
[ 3] 22.0-23.0 sec  0.00 Bytes    0.00 bits/sec   0.000 ms    0/    0 (0%)
[ 3] 23.0-24.0 sec  0.00 Bytes    0.00 bits/sec   0.000 ms    0/    0 (0%)
[ 3] 24.0-25.0 sec  1.44 KBytes   11.8 Kbits/sec 967.931 ms   0/    1 (0%)
[ 3] 25.0-26.0 sec  5.74 KBytes   47.0 Kbits/sec 752.926 ms   0/    4 (0%)
[ 3] 26.0-27.0 sec  0.00 Bytes    0.00 bits/sec   0.000 ms    0/    0 (0%)
[ 3] 27.0-28.0 sec  0.00 Bytes    0.00 bits/sec   0.000 ms    0/    0 (0%)
[ 3] 28.0-29.0 sec  0.00 Bytes    0.00 bits/sec   0.000 ms    0/    0 (0%)
[ 3] 29.0-30.0 sec  0.00 Bytes    0.00 bits/sec   0.000 ms    0/    0 (0%)
[ 3] 30.0-31.0 sec  0.00 Bytes    0.00 bits/sec   0.000 ms    0/    0 (0%)
^CWaiting for server threads to complete. Interrupt again to force quit.
[ 3] 31.0-32.0 sec  8.61 KBytes   70.6 Kbits/sec 802.034 ms   0/    6 (0%)
[ 3] 32.0-33.0 sec  0.00 Bytes    0.00 bits/sec   0.000 ms    0/    0 (0%)
[ 3] 33.0-34.0 sec  2.87 KBytes   23.5 Kbits/sec 847.168 ms   0/    2 (0%)
root@d95de34024a9: /opt/oai-nr-ue#

```

Figure 93: Iperf RX in Uu interface with 125 kb/s restrictions

5. Conclusion

The main objective of this thesis was to evaluate the concepts for gNB satellite backhaul using open-source 5G frameworks. This study has consisted, firstly, in an investigation of the technical 5G foundations as well as the analyse of the use of the srsRAN and OAI platform, as they are open-source frameworks.

Once it was decided that OAI would be used, the different configurations offered were studied. Since for this project it was decided that simulations would be done on a single computer, without the use of any hardware device, and taking into account that the main objective was to analyse the N2 and N3 interfaces, which connect the control and data plane between the gNB and the CN, it was decided to use the “OAI Full Stack 5G-NR RF simulation with containers” [35]. Since only one computer was used for this thesis, this made it very cost-effective.

The reason for analysing the N2 and N3 interfaces was that these 2 interfaces can be routed through the satellite, to enable access to the telecommunications network to any base station in remote locations.

In the case of the CN, only the UPF, AMF, SMF and NRF blocks were deployed, as they were sufficient for the purpose of analysis.

5.1. Result Discussion

To analyse the results, the traffic from each block of interest to the block next to it was studied. In this way, the messages exchanged on the interfaces Uu, from UE to gNB; N6, from UPF to EXT-DN; N2, from gNB to AMF; N3, from gNB to UPF were captured, the last two interfaces being the most important ones to consider.

In the case of N2, the control messages exchanged for the correct registration of each UE have been studied. In the case of N3, being a data interface, it has been analysed how the traffic is sent through this interface.

Once these interfaces were analysed, in which the transmission was clean, without any type of limitation, other than that which the capacity of the computer used could offer. Some limitations have been introduced in the gNB to see how the traffic and response times varied and make it more realistic, as the aim of this thesis was to discuss the possible use of routing the N2 and N3 interfaces via satellites, and the limitations this offers have to be taken into account.

By applying the limitations and limiting the data rate, it has been seen that the ping has been rising, so it will depend on the type of service or application to be used to know if it is acceptable or not, it has also been seen that with a limitation of 25kb/s, the ping did not reach, so that no service could be offered with that data rate.

5.2. Future Work

As mentioned above, the simulations have been done fully in software, without

introducing any kind of device, be it USRP device or mobile phone. So a possible future project would be to introduce these hardware to make the simulations more realistic.

Another possible project would be to use devices that each simulate a 5G CN block and thus have to be interconnected.

It is also possible to simulate using more blocks in the CN, since in this case only the essential ones have been used to analyse the interfaces that were of interest.

Other types of channel limitations could also be introduced for signal propagation.

List of Figures

Figure 1: Evolution of the cellular network [3].....	2
Figure 2: 5G a) non-standalone and b) standalone configuration	3
Figure 3: 5G architecture divided into several functional blocks with their interfaces [6]	4
Figure 4: NG-RAN architecture	6
Figure 5: Satellite backhaul.....	7
Figure 6: 5G Data Path Protocol Stack [13]	8
Figure 7: Control Plane protocol stack [20]	10
Figure 8: User Plane protocol stack [20].....	11
Figure 9: 5G applications [24].....	12
Figure 10: 5G NFs Slices [17]	13
Figure 11: Satellite Backhaul [25]	14
Figure 12: Characteristics of frequency range [29]	16
Figure 13: Folder from where the commands will be executed.....	19
Figure 14: CN execution.....	20
Figure 15: Created bridges after executing CN	20
Figure 16: CN blocks in healthy state.....	20
Figure 17: gNB correct execution	21
Figure 18: gNB connection in the AMF	21
Figure 19: OAI NR-UE correct deployment.....	21
Figure 20: Adding a user in the SQL configuration	21
Figure 21: Adding a user in docker-compose configuration file	22
Figure 22: Final setup with each block information.....	22
Figure 23: Checking internet connectivity.....	23
Figure 24: Iperf server check.....	23
Figure 25: Iperf client check.....	23
Figure 26: OAI-NRF block interfaces	25
Figure 27: OAI-AMF block interfaces	25
Figure 28: OAI-SMF block interfaces	26
Figure 29: OAI-SPGWU block interfaces.....	26
Figure 30: OAI-gNB block interfaces.....	27
Figure 31: OAI-NR-UE block interfaces	27
Figure 32: OAI-NR-UE2 block interfaces.....	28
Figure 33: OAI created networks	28
Figure 34: Configuration of the RF Simulator in the gNB block	31
Figure 35: Configuration of the RF Simulator in the UE block.....	31
Figure 36: gNB rfsim running.....	31
Figure 37: UE rfsim running.....	32
Figure 38: Uu traffic flow	32
Figure 39: Uu interface Wireshark traffic	33
Figure 40: Uu interface packet lengths	33
Figure 41: Packets/sec going over the Uu interface	33
Figure 42: Iperf traffic on the Uu interface	34
Figure 43: Uu iperf traffic flow.....	34
Figure 44: UU iperf traffic created.....	35
Figure 45: Moment when the gNB is connected to the CN.....	36
Figure 46: Moment when the UE is connected to the CN	36
Figure 47: Packets/sec on the N2 interface	37

Figure 48: Packet size on the N2 interface	37
Figure 49: Traffic flow in N2 interface.....	37
Figure 50: SCTP protocol traffic flow in N2 interface	38
Figure 51: N2 interface HEARTBEAT messages	38
Figure 52: Iperf traffic created on the N3 interface.....	38
Figure 53: Packets/sec on N3 interface	39
Figure 54: N3 interface traffic flow	39
Figure 55: Packet length on the N3 interface.....	39
Figure 56: Packets detected on the N3 interface	40
Figure 57: Traffic captured at the N6 interface	40
Figure 58: N6 interface traffic flow	40
Figure 59: N6 interface packet length	41
Figure 60: Packets/sec when creating iperf traffic on the N6 interface.....	41
Figure 61: Result of iperf traffic created.....	41
Figure 62: N6 iperf traffic flow.....	41
Figure 63: N6 packet lengths.....	42
Figure 64: N6 iperf analysis	42
Figure 65: N3 iperf analysis	43
Figure 66: Uu iperf analysis	43
Figure 67: Comparison of iperf traffic from top to bottom: Uu, N3 and N6	44
Figure 68: Possible SXC1280 data rate values	44
Figure 69: Ping on N2 with 4 Mb/s limitation.....	45
Figure 70: N2 ping Packets/sec graph	45
Figure 71: Ping detected in N2 interface	46
Figure 72: Ping on N2 with 2 Mb/s limitation.....	46
Figure 73: Ping on N2 with 1 Mb/s limitation.....	46
Figure 74: Ping on N2 with 500 kb/s limitation.....	47
Figure 75: Ping on N2 with 250 kb/s limitation.....	47
Figure 76: Ping on N2 with 125 kb/s limitation.....	47
Figure 77: Packets/sec graph in N2 interface with limitation	47
Figure 78: Ping on N2 with 50 kb/s limitation.....	48
Figure 79: Ping on N2 with 25 kb/s limitation.....	48
Figure 80: Ping on N3 with 2 Mb/s limitation.....	48
Figure 81: Packets/sec graph in N3 interface with limitation	49
Figure 82: Ping on N3 with 2 Mb/s limitation.....	49
Figure 83: Ping on N3 with 1 Mb/s limitation.....	49
Figure 84: Ping on N3 with 500 kb/s limitation.....	50
Figure 85: Ping on N3 with 250 kb/s limitation.....	50
Figure 86: Ping on N3 with 125 kb/s limitation.....	50
Figure 87: Ping on N3 with 50 kb/s limitation.....	50
Figure 88: Ping on N3 with 25 kb/s limitation.....	51
Figure 89: N2 N3 latency comparison.....	51
Figure 90: Iperf TX in Uu interface with 4 Mb/s restrictions	52
Figure 91: Iperf RX in Uu interface with 4 Mb/s restrictions.....	52
Figure 92: Iperf TX in Uu interface with 125 kb/s restrictions.....	52
Figure 93: Iperf RX in Uu interface with 125 kb/s restrictions	53

List of Tables

Table 1: Frequency range	15
Table 2: Information of each block.....	24
Table 3: Data rate at the Uu interface	32
Table 4: N2 and N3 interface latency	51

Acronyms

gNB gNodeB

CN Core Network

OAI Open Air Interface

UMTS Universal Mobile Telecommunications System

MIMO Multiple Input Multiple Output

OFDM Orthogonal Frequency Division Multiplexing

DL Down Link

AI Artificial Intelligence

NG-RAN Next Generation Radio Access Network

3GPP 3rd generation Partnership Project

NR New Radio

RAN Radio Access Network

SA Stand Alone

NSA Non-Stand Alone

NFV Network Functions Virtualization

5GC 5G Core

SBA Service Base Architecture

EPC Evolved Packet Core

NF Network Functions

API Application Programming Interface

NRF Network repository Functions

AMF Access and Mobility management Function

UE User Equipment

NAS Non Access Stratum

UPF User Plane Function

QoS Quality of Service

PDU Protocol Data Unit

DN Data Network

UP User Plane

SMF Session Management Function
AUSF Authentication Server Function
UDM Unified Data Management
HSS Home Subscriber Service
CU Centralized Unit
DU Distributed Unit
TCP Transmission Control protocol
DFE Digital Front End
PHY Physical
COTS Commercial off-the-shelf
RLC Radio Link Control
MAC Medium Access Control
RRC Radio Resource Control
PDCP Packet Data Convergence Protocol
SDAP Service Data Adaptation Protocol
CUPS Control and User Plane Separation
GTP-U GPRS Tunneling Protocol
QoE Quality of Experience
PCF Policy Control Function
CDR Charging Data Record
SGW Serving Gateway
PGW Packet Data Network Gateway
CP Control Plane
PDN Packet Data Network
FDD Frequency Division Duplex
TDD Time Division Duplex
OFDMA Orthogonal Frequency Division Multiple Access
SCS Sub-Carrier Spacing
BPSK Binary Phase-Shift Keying
QAM Quadrature Amplitude Modulation
HARQ Hybrid Automatic Repeat Request
SDAP Service Data Adaptation Protocol

NGAP NG Application Protocol
eMBB Enhanced Mobile Broadband
URLLC Ultra Reliable Low Latency Communications
MIoT Massive IoT
UL Up Link
LPWA Low Power Wide Area
KPI Key Performance Indicator
MEC Multi-access Edge Computing
V2X Vehicle to Everything
IoT Internet of Things
FR Frequency Range
CP Cyclic Prefix
MME Mobility Management Entity
HSS Home Subscriber Server
OSA OpenAirInterface Software Alliance
SC-FDMA Single Carrier frequency-division multiple acces
UDR Unified Data Repository
EXT-DN External Data Network
UDP User Datagram Protocol
SCTP Stream Control Transmission Protocol
MTU Maximum Transmission Unit
USRP Universal Software Radio Peripheral

Bibliography

- [1] A. F. M. Shahen Shah, "A Survey From 1G to 5G Including the Advent of 6G: Architectures, Multiple Access Techniques, and Emerging Technologies," 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), 2022, pp. 1117-1123, doi: 10.1109/CCWC54503.2022.9720781.
- [2] S. K. James Rogerson, 15 02 2022. [Online]. Available: https://5g.co.uk/guides/how-fast-is-5g/#5G_latency_explained.
- [3] "Reorganización del espectro," 07 03 2022. [Online]. Available: <https://forum.huawei.com/enterprise/es/reorganizaci%C3%B3n-del-espectro/thread/834875-100267>.
- [4] "Road to 5G: Introduction and Migration," 04 2018.
- [5] ETSI TS 123.501 V16.6.0", 10-2020, [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/16.06.00_60/ts_123501v160600p.pdf.
- [6] M. Dryjanski, "GRANDMETRIC, 5G Core Network Functions," [Online]. Available: <https://www.grandmetric.com/2018/03/02/5g-core-network-functions/>.
- [7] Cisco, Ultra Cloud Core 5G User Plane Function, Release 2021.01. https://www.cisco.com/c/en/us/td/docs/wireless/ucc/upf/2020-03/b_ucc-5g-upf-config-and-admin-guide_2020-03/b_UPF_chapter_011011.pdf.
- [8] "3GPP, NG-RAN Architecture," 2021. https://www.3gpp.org/news-events/2160-ng_ran_architecture.
- [9] Masini, G. (2021). A Guide to NG-RAN Architecture. In: Lin, X., Lee, N. (eds) 5G and Beyond. Springer, Cham. https://doi.org/10.1007/978-3-030-58197-8_8
- [10] E. Jordan, "5G Technology world," 24 02 2021. [Online]. Available: <https://www.5gtechnologyworld.com/open-ran-functional-splits-explained/>.
- [11] "Principles of 5G Backhaul," 28 04 2021. [Online]. Available: <https://www.aciist.com/principles-of-5g-backhaul/>.
- [12] "ETSI TS 138 305 V15.1.0" 10 2018. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/138300_138399/138305/15.01.00_60/ts_138305v150100p.pdf.
- [13] D. Cheung, "5G Core GTP-U," 07 2020. [Online]. Available: <https://clcnetwork.wordpress.com/2020/07/05/5g-core-part-3-user-plane-and-gtp-u-tunnel/>.
- [14] "Developing Solutions," [Online]. Available: <https://www.developingsolutions.com/products/dstest-5g-core-network-testing/n4-interface/>.
- [15] K. Liang, G. Liu, L. Zhao, X. Chu, S. Wang and L. Hanzo, "Performance Analysis of Cellular Radio Access Networks Relying on Control- and User-Plane Separation," in *IEEE Transactions on Vehicular Technology*, vol. 68, no. 7, pp. 7241-7245, July 2019, doi: 10.1109/TVT.2019.2918546.
- [16] K. R. Kumar, K. S. Kumar Kavuluri and D. Das, "Novel Algorithm to Recover the Lost CDR Information by Control and User Planes Separation in 4G and 5G," 2021 *IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, 2021, pp. 1-6, doi: 10.1109/CONECCT52877.2021.9622598.
- [17] A. Detti, 5G Functional Architecture. <https://www.5gitaly.eu/2018/wp-content/uploads/2019/01/5G-Italy-White-eBook-Functional-architecture.pdf>.

- [18] "ETSI TS 138 211," [Online]. Available: https://www.etsi.org/deliver/etsi_ts/138200_138299/138211/15.02.00_60/ts_138211v150200p.pdf.
- [19] F. Firmin, "3GPP, NAS," Available: <https://www.3gpp.org/technologies/keywords-acronyms/96-nas>.
- [20] F. Zampognaro, "ROMARS," [Online]. Available: <https://romars.tech/publicazioni/n3iwf/>.
- [21] S. Ahmadi, "ScienceDirect, "5G Network Architecture"," 2019. <https://www.sciencedirect.com/topics/engineering/protocol-stack>.
- [22] Zhengyu Zhu, Xingwang Li, Zheng Chu, Chapter 2 - Three major operating scenarios of 5G: eMBB, mMTC, URLLC, Editor(s): Zhengyu Zhu, Zheng Chu, Xingwang Li, Intelligent Sensing and Communications for Internet of Everything, Academic Press, 2022, Pages 15-76, ISBN 9780323856553, <https://doi.org/10.1016/B978-0-32-385655-3.00006-0>
- [23] M. C. X. A. T. M. Riccardo Trivisonno, "MDPI, mIoT Slice for 5G Systems: Design and Performance Evaluation," 21 02 2018. <https://doi.org/10.3390/s18020635>.
- [24] "D2.1 5G VICTORI Use case and requirements definition and reference architecture for vertical services," 2020. https://www.5g-victori-project.eu/wp-content/uploads/2020/06/2020-03-31-5G-VICTORI_D2.1_v1.0.pdf.
- [25] Liolis, Geurtz, A., Sperber, R., Schulz, D., Watts, S., Poziopoulou, G., Evans, B., Wang, N., Vidal, O., Tiomela Jou, B., Fitch, M., Diaz Sendra, S., Sayyad Khodashenas, P., & Chuberre, N. (2019). "Use cases and scenarios of 5G integrated satellite-terrestrial networks for enhanced mobile broadband: The SaT5G approach. International Journal of Satellite Communications and Networking, 37(2), 91–112. <https://doi.org/10.1002/sat.1245>"
- [26] K. Ray and A. Banerjee, "Prioritized Fault Recovery Strategies for Multi-Access Edge Computing Using Probabilistic Model Checking," in *IEEE Transactions on Dependable and Secure Computing*, doi: 10.1109/TDSC.2022.3143877.
- [27] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta and D. Sabella, "On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration," in *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657-1681, thirdquarter 2017, doi: 10.1109/COMST.2017.2705720.
- [28] "ETSI, Multi-access Edge Computing," [Online]. Available: <https://www.etsi.org/technologies/multi-access-edge-computing>.
- [29] 5. P. A. W. Group, "View on 5G Architecture," 2019. https://5g-ppp.eu/wp-content/uploads/2019/07/5G-PPP-5G-Architecture-White-Paper_v3.0_PublicConsultation.pdf.
- [30] "srsRAN 22.04 Documentation," [Online]. Available: <https://docs.srsran.com/en/latest/>.
- [31] "OAI," [Online]. Available: <https://openairinterface.org/>.
- [32] "OAI-RAN-PROJECT," [Online]. Available: <https://openairinterface.org/oai-5g-ran-project/>.
- [33] "OAI-FeatureSet," [Online]. Available: https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/develop/doc/FEATURE_SET.md#functional-split-architecture.
- [34] "OAI-CN," [Online]. Available: <https://openairinterface.org/oai-5g-core-network-project/>.
- [35] "OAI Full Stack 5G-NR RF simulation with containers," [Online]. Available: https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/develop/ci-scripts/yaml_files/5g_rfsimulator/README.md.

- [36] "Docker, Use bridge networks," [Online]. Available: <https://docs.docker.com/network/bridge/>.
- [37] "ETSI TS 138 413 V15.0.0," 07 2018. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/138400_138499/138413/15.00.00_60/ts_138413v150000p.pdf.

Appendix

Appendix 1

```
version: '3.8'
services:
  oai-nrf:
    container_name: "rfsim5g-oai-nrf"
    image: oai-nrf:latest
    environment:
      - NRF_INTERFACE_NAME_FOR_SBI=eth0
      - NRF_INTERFACE_PORT_FOR_SBI=80
      - NRF_INTERFACE_HTTP2_PORT_FOR_SBI=9090
      - NRF_API_VERSION=v1
      - INSTANCE=0
      - PID_DIRECTORY=/var/run
    networks:
      public_net:
        ipv4_address: 192.168.71.130
    volumes:
      - ./nrf-healthcheck.sh:/openair-nrf/bin/nrf-healthcheck.sh
    healthcheck:
      test: /bin/bash -c "/openair-nrf/bin/nrf-healthcheck.sh"
      interval: 10s
      timeout: 5s
      retries: 5
  mysql:
    container_name: "rfsim5g-mysql"
    image: mysql:5.7
    volumes:
      - ./oai_db.sql:/docker-entrypoint-initdb.d/oai_db.sql
      - ./mysql-healthcheck.sh:/tmp/mysql-healthcheck.sh
    environment:
      - TZ=Europe/Paris
      - MYSQL_DATABASE=oai_db
      - MYSQL_USER=test
      - MYSQL_PASSWORD=test
      - MYSQL_ROOT_PASSWORD=linux
    healthcheck:
      test: /bin/bash -c "/tmp/mysql-healthcheck.sh"
      interval: 10s
      timeout: 5s
      retries: 5
    networks:
      public_net:
        ipv4_address: 192.168.71.131
  oai-amf:
    container_name: "rfsim5g-oai-amf"
    image: oai-amf:latest
    environment:
      - TZ=Europe/paris
      - INSTANCE=0
      - PID_DIRECTORY=/var/run
      - MCC=208
      - MNC=99
```

```

- REGION_ID=128
- AMF_SET_ID=1
- SERVED_GUAMI_MCC_0=208
- SERVED_GUAMI_MNC_0=99
- SERVED_GUAMI_REGION_ID_0=128
- SERVED_GUAMI_AMF_SET_ID_0=1
- SERVED_GUAMI_MCC_1=460
- SERVED_GUAMI_MNC_1=11
- SERVED_GUAMI_REGION_ID_1=10
- SERVED_GUAMI_AMF_SET_ID_1=1
- PLMN_SUPPORT_MCC=208
- PLMN_SUPPORT_MNC=99
- PLMN_SUPPORT_TAC=0x0001
- SST_0=1
- SD_0=1
- SST_1=1
- SD_1=12
- AMF_INTERFACE_NAME_FOR_NGAP=eth0
- AMF_INTERFACE_NAME_FOR_N11=eth0
- SMF_INSTANCE_ID_0=1
- SMF_FQDN_0=oai-smf
- SMF_IPV4_ADDR_0=0.0.0.0
- SMF_HTTP_VERSION_0=v1
- SELECTED_0=true
- SMF_INSTANCE_ID_1=2
- SMF_FQDN_1=oai-smf
- SMF_IPV4_ADDR_1=0.0.0.0
- SMF_HTTP_VERSION_1=v1
- SELECTED_1=false
- MYSQL_SERVER=192.168.71.131
- MYSQL_USER=root
- MYSQL_PASS=linux
- MYSQL_DB=oai_db
- OPERATOR_KEY=c42449363bbad02b66d16bc975d77cc1
- NRF_IPV4_ADDRESS=192.168.71.130
- NRF_PORT=80
- NF_REGISTRATION=yes
- SMF_SELECTION=yes
- USE_FQDN_DNS=yes
- NRF_API_VERSION=v1
- NRF_FQDN=oai-nrf
- EXTERNAL_AUSF=no
- AUSF_IPV4_ADDRESS=0.0.0.0
- AUSF_PORT=80
- AUSF_API_VERSION=v1
- AUSF_FQDN=localhost
depends_on:
- oai-nrf
- mysql
volumes:
- ./amf-healthcheck.sh:/openair-amf/bin/amf-healthcheck.sh
healthcheck:
test: /bin/bash -c "/openair-amf/bin/amf-healthcheck.sh"
interval: 10s
timeout: 15s
retries: 5
networks:
public_net:
ipv4_address: 192.168.71.132
oai-smf:
container_name: "rfsim5g-oai-smf"

```

```

image: oai-smf:latest
environment:
  - TZ=Europe/Paris
  - INSTANCE=0
  - PID_DIRECTORY=/var/run
  - SMF_INTERFACE_NAME_FOR_N4=eth0
  - SMF_INTERFACE_NAME_FOR_SBI=eth0
  - SMF_INTERFACE_PORT_FOR_SBI=80
  - SMF_INTERFACE_HTTP2_PORT_FOR_SBI=9090
  - SMF_API_VERSION=v1
  - DEFAULT_DNS_IPV4_ADDRESS=172.21.3.100
  - DEFAULT_DNS_SEC_IPV4_ADDRESS=4.4.4.4
  - AMF_IPV4_ADDRESS=0.0.0.0
  - AMF_PORT=80
  - AMF_API_VERSION=v1
  - AMF_FQDN=oai-amf
  - UDM_IPV4_ADDRESS=127.0.0.1
  - UDM_PORT=80
  - UDM_API_VERSION=v1
  - UDM_FQDN=localhost
  - UPF_IPV4_ADDRESS=192.168.71.134
  - UPF_FQDN_0=oai-spgwu
  - NRF_IPV4_ADDRESS=192.168.71.130
  - NRF_PORT=80
  - NRF_API_VERSION=v1
  - NRF_FQDN=oai-nrf
  - REGISTER_NRF=yes
  - DISCOVER_UPF=yes
  - USE_FQDN_DNS=yes
  - DNN_NI0=oai
  - DNN_NI2=oai.ipv4
depends_on:
  - oai-nrf
  - oai-amf
volumes:
  - ./smf-healthcheck.sh:/openair-smf/bin/smf-healthcheck.sh
healthcheck:
  test: /bin/bash -c "/openair-smf/bin/smf-healthcheck.sh"
  interval: 10s
  timeout: 5s
  retries: 5
networks:
  public_net:
    ipv4_address: 192.168.71.133
oai-spgwu:
  container_name: "rfsim5g-oai-spgwu"
  image: oai-spgwu-tiny:latest
  environment:
    - TZ=Europe/Paris
    - PID_DIRECTORY=/var/run
    - SGW_INTERFACE_NAME_FOR_S1U_S12_S4_UP=eth0
    - SGW_INTERFACE_NAME_FOR_SX=eth0
    - PGW_INTERFACE_NAME_FOR_SGI=eth0
    - NETWORK_UE_NAT_OPTION=yes
    - NETWORK_UE_IP=12.1.1.0/24
    - SPGWC0_IP_ADDRESS=192.168.71.133
    - BYPASS_UL_PFCP_RULES=no
    - MCC=208
    - MNC=99
    - MNC03=099
    - TAC=1

```



```

- GTP_EXTENSION_HEADER_PRESENT=yes
- GW_ID=1
- REALM=openairinterface.org
- ENABLE_5G_FEATURES=yes
- REGISTER_NRF=yes
- USE_FQDN_NRF=yes
- UPF_FQDN_5G=oai-spgwu
- NRF_IPV4_ADDRESS=192.168.71.130
- NRF_PORT=80
- NRF_API_VERSION=v1
- NRF_FQDN=oai-nrf
- NSSAI_SST_0=1
- NSSAI_SD_0=1
- DNN_0=oai
depends_on:
- oai-nrf
- oai-smf
cap_add:
- NET_ADMIN
- SYS_ADMIN
cap_drop:
- ALL
privileged: true
volumes:
- ./spgwu-healthcheck.sh:/openair-spgwu-tiny/bin/spgwu-
healthcheck.sh
healthcheck:
test: /bin/bash -c "/openair-spgwu-tiny/bin/spgwu-
healthcheck.sh"
interval: 10s
timeout: 5s
retries: 5
networks:
public_net:
ipv4_address: 192.168.71.134
traffic_net:
ipv4_address: 192.168.72.134
oai-ext-dn:
image: ubuntu:bionic
privileged: true
container_name: rfsim5g-oai-ext-dn
entrypoint: /bin/bash -c \
"apt update; apt install -y procps iptables iproute2
iperf iputils-ping;"\
"iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE;"\
"ip route add 12.1.1.0/24 via 192.168.72.134 dev eth0;
sleep infinity"
depends_on:
- oai-spgwu
networks:
traffic_net:
ipv4_address: 192.168.72.135
healthcheck:
test: /bin/bash -c "ping -c 2 192.168.72.134"
interval: 10s
timeout: 5s
retries: 5
oai-gnb:
image: oai-gnb:develop
privileged: true
container_name: rfsim5g-oai-gnb

```

```

environment:
  RFSIMULATOR: server
  USE_SA_TDD_MONO: 'yes'
  GNB_NAME: gnb-rfsim
  TAC: 1
  MCC: '208'
  MNC: '99'
  MNC_LENGTH: 2
  NSSAI_SST: 1
  NSSAI_SD0: 1
  NSSAI_SD1: 112233
  AMF_IP_ADDRESS: 192.168.71.132
  GNB_NGA_IF_NAME: eth0
  GNB_NGA_IP_ADDRESS: 192.168.71.140
  GNB_NGU_IF_NAME: eth0
  GNB_NGU_IP_ADDRESS: 192.168.71.140
  USE_ADDITIONAL_OPTIONS: --sa -E --rfsim --
log_config.global_log_options level,nocolor,time
depends_on:
  - oai-ext-dn
networks:
  public_net:
    ipv4_address: 192.168.71.140
healthcheck:
  test: /bin/bash -c "pgrep nr-softmodem"
  interval: 10s
  timeout: 5s
  retries: 5
oai-nr-ue:
  image: oai-nr-ue:develop
  privileged: true
  container_name: rfsim5g-oai-nr-ue
  environment:
    RFSIMULATOR: 192.168.71.140
    FULL_IMSI: '208990100001100'
    FULL_KEY: 'fec86ba6eb707ed08905757b1bb44b8f'
    OPC: 'C42449363BBAD02B66D16BC975D77CC1'
    DNN: oai
    NSSAI_SST: 1
    NSSAI_SD: 1
    USE_ADDITIONAL_OPTIONS: -E --sa --rfsim -r 106 --
numerology 1 -C 3619200000 --nokrnmod --log_config.global_log_options
level,nocolor,time
depends_on:
  - oai-gnb
networks:
  public_net:
    ipv4_address: 192.168.71.150
healthcheck:
  test: /bin/bash -c "pgrep nr-uesoftmodem"
  interval: 10s
  timeout: 5s
  retries: 5
oai-nr-ue2:
  image: oai-nr-ue:develop
  privileged: true
  container_name: rfsim5g-oai-nr-ue2
  environment:
    RFSIMULATOR: 192.168.71.140
    FULL_IMSI: '208990100001101'

```

```

    FULL_KEY: 'fec86ba6eb707ed08905757b1bb44b8f'
    OPC: 'C42449363BBAD02B66D16BC975D77CC1'
    DNN: oai
    NSSAI_SST: 1
    NSSAI_SD: 1
    USE_ADDITIONAL_OPTIONS: -E --sa --rfsim -r 106 --
numerology 1 -C 3619200000 --nokrnmod --log_config.global_log_options
level,nocolor,time
    depends_on:
      - oai-gnb
    networks:
      public_net:
        ipv4_address: 192.168.71.151
    healthcheck:
      test: /bin/bash -c "pgrep nr-uesoftmodem"
      interval: 10s
      timeout: 5s
      retries: 5

networks:
  public_net:
    driver: bridge
    name: rfsim5g-oai-public-net
    ipam:
      config:
        - subnet: 192.168.71.128/26
    driver_opts:
      com.docker.network.bridge.name: "rfsim5g-public"
  traffic_net:
    driver: bridge
    name: rfsim5g-oai-traffic-net
    ipam:
      config:
        - subnet: 192.168.72.128/26
    driver_opts:
      com.docker.network.bridge.name: "rfsim5g-traffic"

```

Appendix 2

No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	fe80::2cdc:9aff:fe7...	ff02::2	ICMPv6	70	Router Solicitation from 2e:dc:9a:72:b2:84
2	102.613771618	192.168.71.129	224.0.0.251	MDNS	87	Standard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR _ipp._tcp.local, "QM" quest
3	107.762882684	fe80::42:e4ff:fe80:	ff02::fb	MDNS	107	Standard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR _ipp._tcp.local, "QM" quest
4	107.942176389	fe80::2cdc:9aff:fe7...	ff02::fb	MDNS	107	Standard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR _ipp._tcp.local, "QM" quest
5	470.093563579	02:42:c0:a8:47:8c	Broadcast	ARP	42	Who has 192.168.71.132? Tell 192.168.71.140
6	676.682174632	02:42:c0:a8:47:96	Broadcast	ARP	42	Who has 192.168.71.149? Tell 192.168.71.150
7	678.885025952	02:42:c0:a8:47:84	Broadcast	ARP	42	Who has 192.168.71.131? Tell 192.168.71.132
8	678.885059016	02:42:c0:a8:47:83	02:42:c0:a8:47:84	ARP	42	192.168.71.131 is at 02:42:c0:a8:47:83
9	678.885071732	192.168.71.132	192.168.71.131	TCP	74	35934 - 3306 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3864300981 TSecr=0 WS
10	678.885084151	192.168.71.131	192.168.71.132	TCP	74	3306 - 35934 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2870286653
11	678.885097186	192.168.71.132	192.168.71.131	TCP	66	35934 - 3306 [ACK] Seq=1 Ack=1 Win=64512 Len=0 TSval=3864300981 TSecr=2870286653
12	678.885257558	192.168.71.131	192.168.71.132	MySQL	144	Server Greeting proto=10 version=5.7.38
13	678.885275638	192.168.71.132	192.168.71.131	TCP	66	35934 - 3306 [ACK] Seq=1 Ack=79 Win=64512 Len=0 TSval=3864300981 TSecr=2870286653
14	678.894929931	192.168.71.132	192.168.71.131	MySQL	102	Login Request user=
15	678.894938261	192.168.71.131	192.168.71.132	TCP	66	3306 - 35934 [ACK] Seq=79 Ack=37 Win=65536 Len=0 TSval=2870286663 TSecr=3864300991
16	678.897513393	192.168.71.132	192.168.71.131	TLSv1.2	264	Client Hello
17	678.897523658	192.168.71.131	192.168.71.132	TCP	66	3306 - 35934 [ACK] Seq=79 Ack=235 Win=65536 Len=0 TSval=2870286666 TSecr=3864300994
18	678.898867954	192.168.71.131	192.168.71.132	TLSv1.2	2088	Server Hello, Certificate, Server Key Exchange, Certificate Request, Server Hello Done
19	678.898894750	192.168.71.132	192.168.71.131	TCP	66	35934 - 3306 [ACK] Seq=235 Ack=2101 Win=64512 Len=0 TSval=3864300995 TSecr=2870286667
20	678.907646656	192.168.71.132	192.168.71.131	TLSv1.2	171	Certificate, Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
21	678.907654773	192.168.71.131	192.168.71.132	TCP	66	3306 - 35934 [ACK] Seq=2101 Ack=340 Win=65536 Len=0 TSval=2870286676 TSecr=3864301004
22	678.907910116	192.168.71.131	192.168.71.132	TLSv1.2	308	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
23	678.907928288	192.168.71.132	192.168.71.131	TCP	66	35934 - 3306 [ACK] Seq=340 Ack=2343 Win=64512 Len=0 TSval=3864301004 TSecr=2870286676
24	678.908128463	192.168.71.132	192.168.71.131	TLSv1.2	206	Application Data
25	678.908132936	192.168.71.131	192.168.71.132	TCP	66	3306 - 35934 [ACK] Seq=2343 Ack=540 Win=65536 Len=0 TSval=2870286676 TSecr=3864301004
26	678.908212105	192.168.71.131	192.168.71.132	TLSv1.2	117	Application Data
27	678.908226795	192.168.71.132	192.168.71.131	TCP	66	35934 - 3306 [ACK] Seq=540 Ack=2394 Win=64512 Len=0 TSval=3864301004 TSecr=2870286676
28	678.908246435	192.168.71.132	192.168.71.131	TLSv1.2	102	Application Data
29	678.908249866	192.168.71.131	192.168.71.132	TCP	66	3306 - 35934 [ACK] Seq=2394 Ack=576 Win=65536 Len=0 TSval=2870286676 TSecr=3864301004
30	678.908277962	192.168.71.131	192.168.71.132	TLSv1.2	106	Application Data
31	678.908290020	192.168.71.132	192.168.71.131	TCP	66	35934 - 3306 [ACK] Seq=576 Ack=2434 Win=64512 Len=0 TSval=3864301004 TSecr=2870286676
32	678.908309527	192.168.71.132	192.168.71.131	TLSv1.2	184	Application Data
33	678.908309329	192.168.71.131	192.168.71.132	TCP	66	3306 - 35934 [ACK] Seq=2434 Ack=694 Win=65536 Len=0 TSval=2870286676 TSecr=3864301004
34	678.908746450	192.168.71.131	192.168.71.132	TLSv1.2	381	Application Data
35	678.908761696	192.168.71.132	192.168.71.131	TCP	66	35934 - 3306 [ACK] Seq=694 Ack=2749 Win=64512 Len=0 TSval=3864301005 TSecr=2870286677
36	678.916860398	192.168.71.132	192.168.71.131	TLSv1.2	215	Application Data
37	678.916868140	192.168.71.131	192.168.71.132	TCP	66	3306 - 35934 [ACK] Seq=2749 Ack=843 Win=65536 Len=0 TSval=2870286685 TSecr=3864301013
38	678.917315079	192.168.71.131	192.168.71.132	TLSv1.2	147	Application Data
39	678.917333641	192.168.71.132	192.168.71.131	TCP	66	35934 - 3306 [ACK] Seq=843 Ack=2830 Win=64512 Len=0 TSval=3864301013 TSecr=2870286685
40	678.917388729	192.168.71.132	192.168.71.131	TLSv1.2	176	Application Data
41	678.917662881	192.168.71.131	192.168.71.132	TLSv1.2	147	Application Data
42	678.917678543	192.168.71.132	192.168.71.131	TCP	66	35934 - 3306 [ACK] Seq=953 Ack=2911 Win=64512 Len=0 TSval=3864301014 TSecr=2870286686
43	680.154417656	02:42:c0:a8:47:84	Broadcast	ARP	42	Who has 192.168.71.133? Tell 192.168.71.132
44	684.031909997	02:42:c0:a8:47:83	02:42:c0:a8:47:84	ARP	42	Who has 192.168.71.132? Tell 192.168.71.131
45	684.031934374	02:42:c0:a8:47:84	02:42:c0:a8:47:83	ARP	42	192.168.71.132 is at 02:42:c0:a8:47:84
46	884.735965652	fe80::2cdc:9aff:fe7...	ff02::2	ICMPv6	70	Router Solicitation from 2e:dc:9a:72:b2:84

- ▶ Frame 1: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface vethd9b78c9, id 0
- ▶ Ethernet II, Src: 2e:dc:9a:72:b2:84 (2e:dc:9a:72:b2:84), Dst: IPv6mcast_02 (33:33:00:00:00:02)
- ▶ Internet Protocol Version 6, Src: fe80::2cdc:9aff:fe72:b284, Dst: ff02::2
- ▶ Internet Control Message Protocol v6

Appendix 3

Ping from the UE to blocks that are on the same network rfsim5g-oai-public-net (gNB, AMF, NRF, SMF, UPF, MySQL) : 0.04 ms

```
root@e6e4f504f01c:/opt/oai-nr-ue# ping 127.0.0.1 -c 2
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.025 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.021 ms

--- 127.0.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1004ms
rtt min/avg/max/mdev = 0.021/0.023/0.025/0.002 ms
root@e6e4f504f01c:/opt/oai-nr-ue# ping 192.168.71.134 -c 2
PING 192.168.71.134 (192.168.71.134) 56(84) bytes of data.
64 bytes from 192.168.71.134: icmp_seq=1 ttl=64 time=0.048 ms
64 bytes from 192.168.71.134: icmp_seq=2 ttl=64 time=0.102 ms

--- 192.168.71.134 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1010ms
rtt min/avg/max/mdev = 0.048/0.075/0.102/0.027 ms
root@e6e4f504f01c:/opt/oai-nr-ue# ping 192.168.71.133 -c 2
PING 192.168.71.133 (192.168.71.133) 56(84) bytes of data.
64 bytes from 192.168.71.133: icmp_seq=1 ttl=64 time=0.043 ms
64 bytes from 192.168.71.133: icmp_seq=2 ttl=64 time=0.030 ms

--- 192.168.71.133 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1006ms
rtt min/avg/max/mdev = 0.030/0.036/0.043/0.008 ms
root@e6e4f504f01c:/opt/oai-nr-ue# ping 192.168.71.130 -c 2
PING 192.168.71.130 (192.168.71.130) 56(84) bytes of data.
64 bytes from 192.168.71.130: icmp_seq=1 ttl=64 time=0.049 ms
64 bytes from 192.168.71.130: icmp_seq=2 ttl=64 time=0.037 ms

--- 192.168.71.130 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1011ms
rtt min/avg/max/mdev = 0.037/0.043/0.049/0.006 ms
root@e6e4f504f01c:/opt/oai-nr-ue# ping 192.168.71.132 -c 2
PING 192.168.71.132 (192.168.71.132) 56(84) bytes of data.
64 bytes from 192.168.71.132: icmp_seq=1 ttl=64 time=0.047 ms
64 bytes from 192.168.71.132: icmp_seq=2 ttl=64 time=0.031 ms

--- 192.168.71.132 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1019ms
rtt min/avg/max/mdev = 0.031/0.039/0.047/0.008 ms
root@e6e4f504f01c:/opt/oai-nr-ue# ping 192.168.71.140 -c 2
PING 192.168.71.140 (192.168.71.140) 56(84) bytes of data.
64 bytes from 192.168.71.140: icmp_seq=1 ttl=64 time=0.038 ms
64 bytes from 192.168.71.140: icmp_seq=2 ttl=64 time=0.035 ms
```

```

--- 192.168.71.140 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1023ms
rtt min/avg/max/mdev = 0.035/0.036/0.038/0.006 ms
root@e6e4f504f01c:/opt/oai-nr-ue# ping 192.168.71.150 -c 2
PING 192.168.71.150 (192.168.71.150) 56(84) bytes of data.
64 bytes from 192.168.71.150: icmp_seq=1 ttl=64 time=0.024 ms
64 bytes from 192.168.71.150: icmp_seq=2 ttl=64 time=0.020 ms

--- 192.168.71.150 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1027ms
rtt min/avg/max/mdev = 0.020/0.022/0.024/0.002 ms
root@e6e4f504f01c:/opt/oai-nr-ue# ping 12.1.1.2 -c 2
PING 12.1.1.2 (12.1.1.2) 56(84) bytes of data.
64 bytes from 12.1.1.2: icmp_seq=1 ttl=64 time=0.025 ms
64 bytes from 12.1.1.2: icmp_seq=2 ttl=64 time=0.022 ms

--- 12.1.1.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1032ms
rtt min/avg/max/mdev = 0.022/0.023/0.025/0.005 ms
root@e6e4f504f01c:/opt/oai-nr-ue# ping 192.168.71.131 -c 2
PING 192.168.71.131 (192.168.71.131) 56(84) bytes of data.
64 bytes from 192.168.71.131: icmp_seq=1 ttl=64 time=0.084 ms
64 bytes from 192.168.71.131: icmp_seq=2 ttl=64 time=0.035 ms

--- 192.168.71.131 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1019ms
rtt min/avg/max/mdev = 0.035/0.059/0.084/0.025 ms
root@e6e4f504f01c:/opt/oai-nr-ue#

```

Ping from EXT-DN to any container on the same network rfsim5g-oai-traffic-net (UPF, EXT-DN): 0.03 ms

```

root@c31e44371ca9:/# ping 192.168.72.134 -c 2
PING 192.168.72.134 (192.168.72.134) 56(84) bytes of data.
64 bytes from 192.168.72.134: icmp_seq=1 ttl=64 time=0.044 ms
64 bytes from 192.168.72.134: icmp_seq=2 ttl=64 time=0.031 ms

--- 192.168.72.134 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1024ms
rtt min/avg/max/mdev = 0.031/0.037/0.044/0.008 ms
root@c31e44371ca9:/# ping 192.168.72.135 -c 2
PING 192.168.72.135 (192.168.72.135) 56(84) bytes of data.
64 bytes from 192.168.72.135: icmp_seq=1 ttl=64 time=0.026 ms
64 bytes from 192.168.72.135: icmp_seq=2 ttl=64 time=0.020 ms

--- 192.168.72.135 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1021ms
rtt min/avg/max/mdev = 0.020/0.023/0.026/0.003 ms
root@c31e44371ca9:/# ping 127.0.0.1 -c 2
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.026 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.019 ms

```

Ping from the UE to a container on the other network: 3 ms

```

root@c31e44371ca9:/# ping 12.1.1.2 -c 2
PING 12.1.1.2 (12.1.1.2) 56(84) bytes of data.
64 bytes from 12.1.1.2: icmp_seq=1 ttl=63 time=3.68 ms
64 bytes from 12.1.1.2: icmp_seq=2 ttl=63 time=2.79 ms

--- 12.1.1.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 2.795/3.237/3.680/0.446 ms

```