TALLINN UNIVERSITY OF TECHNOLOGY

School of Information Technologies

Taivo Miller 182566IVEM

Open Source Open5GCore (O5GC) Platform-based Training Ground

Master's thesis

Supervisor: Margus Rohtla
MSc

Co-supervisor: Yannick Le Moullec
PhD

Tallinn 2024

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Taivo Miller 182566IVEM

Avatud lähtekoodiga Open5GCore (O5GC) platvormil põhinev harjutusväljak

Magistritöö

Juhendaja:	Margus Rohtla MSc
Kaasjuhendaja:	Yannick Le Moullec PhD

Tallinn 2024

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Taivo Miller

06.05.2024

Abstract

5G is the fifth-generation mobile communication network, which represents the latest deployment in mobile communication technology. It offers faster data transfer, provides lower latency, have higher network capacity, and better network connectivity than previous cellular network generation such as 3G and 4G. Thanks to all these improvements, the quality of service and quality of experience have raised for the user, and it can expand services in an unprecedented direction, which was not supported by the older generation mobile communication networks. 5G is such a recent technological development that most mobile network operators have just started opening 5G stand-alone networks. This master thesis deals with several problems. The first one is to get an understanding of 5G ecosystem and an introduction to its architecture. A software application called Open5GCore (O5GC) by Fraunhoffer has been used to explore and learn about the 5G network. O5GC is designed for the 5G ecosystem based on standard 3GPP components and protocols that are easy to adapt and modify as needed. The second one is to analyse the suitability of O5GC towards developing a new course and to carry out a practical lab guide for it. The thesis provides an overview of the 5G ecosystem and its capabilities. It describes the standard developed by 3GPP and O5GC's compliance with that standard. Procedures have been prepared for starting and using O5GC. For that, preliminary work has been performed and guidelines and standards have been studied in order to theoretically understand how 5G works; tests have been carried out with O5GC simulator to connect user equipment (UE) to the network. The procedure helps to carry out O5GC getting started and teaches how to connect the user's devices to the network and analyse the necessity of various functions in the network system. With the master thesis, a guide for doing laboratory work has been made, separated in 3 labs. The thesis paves the way for further developing training material with the O5GC simulator.

This thesis is written in English and is 87 pages long, including 6 chapters, 48 figures and 2 tables.

Annotatsioon

Avatud lähtekoodiga Open5GCore (O5GC) platvormil põhinev harjutusväljak

5G on viienda põlvkonna mobiilsidevõrk, mis esindab uudsemat arengut mobiilside tehnoloogias. See pakub kiiremat andmeedastust, väiksemat viivitust, suuremat võrgumahtu ja paremat võrguühendust kui eelnevad mobiilsidevõrgu põlvkonnad nagu 3G ja 4G. Kõigi nende parendustega paraneb kasutajale vajalik andmeedastuse kvaliteet, mis annab võimaluse laiendada teenuseid ennenägematus suunas, mida seni vanema põlvkonna mobiilsidevõrgud ei toetanud. 5G on nõnda uus tehnoloogiline areng ja selletõttu ei ole mobiilsidevõrgu operaatorid veel jõudnud iseseisevat (*stand alone*) võrke avada. Lõputöö käsitleb korraga mitut probleemi. Need on 5G ökosüsteemist arusaamine ja tutvustus selle ülesehitusest. 5G võrguga tutvumiseks ja õppimiseks on kasutatud Fraunhofferi poolset tarkvararakendust nimega *Open5GCore (O5GC). O5GC* on loodud 5G ökosüsteemi jaoks, mis põhineb standardsetel 3GPP komponentidel ja protokollidel, mida on lihtne kohandada ja muuta vastavalt vajadusele. Õppida *O5GC* kasutama, analüüsida *O5GC* sobivust uue kursuse läbiviimiseks ja selle jaoks teostada praktiline labori juhend. Lõputöö annab ülevaate 5G ökosüsteemist ja selle võimekusest. Lõputöö kirjeldab lahti 3GPP poolt välja töötatud standardi ja *O5GC* näitab selle vastavust standardiga. *O5GC* jaoks on koostatud protseduurid, kuidas *O5GC* käivitada ja kasutada. Selleks on tehtud eeltöö ja uuritud juhendeid ja standardeid, et aru saada teoreetiliselt kuidas 5G töötab ja *O5GC* simulaatoriga on viidud läbi katsed ühendamaks kasutaja seade (*UE*) võrku. Protseduur aitab viia läbi *O5GC* käivitamise ja õpetab ühendama kasutaja seadme võrku ning analüüsima erinevate funktsioonide vajalikkust võrgu süsteemis. Lõputööga valmis juhend laboritöö tegemiseks ja sellega seoses ka 3 laboritöö ülesannet. Autor on välja toonud võimalikud lahendused, mida *O5GC* simulaatoriga veel teha saaks.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 87 leheküljel, 6 peatükki, 48 joonist, 2 tabelit.

List of abbreviations and terms

5G AKA	5G Authentication and Key Agreement
5GC	5G Core Network
5GS	5G System
5G-AN	5G Access Network
5G-GUTI	5G Globally Unique Temporary Identifier
5G-S-TMSI	5G S-Temporary Mobile Subscription Identifier
5G-OCN	5G Open 5G Core Network project
AF	Application Function
AMF	Access and Mobility Management Function
AoI	Area of Interest
AS	Access Stratum
ARIB	Agricultural Registers and Information Board
ATIS	Automatic terminal information service
AUSF	Authentication Server Function
CCSA	Carbon capture and storage association
CM	Connection Management
CN	Core Network

DL	Downlink
ETSI	European Telecommunications Standards Institute
ETTUS	Ettus Research, a National Instrument Brand
GTP	GPRS Tunnelling Protocol
GUAMI	Globally Unique AMF Identifier
HR	Home Routed (roaming)
IMEI/TAC	IMEI Type Allocation Code
LADN	Local Area Data Network
LMF	Location Management Function
Li-Fi	Light fidelity - wireless access network
MM	Mobility Management
N3IWF	Non-3GPP InterWorking Function
NAI	Network Access Identifier
NAS	Non-Access Stratum
NEF	Network Exposure Function
NF	Network Function
NGAP	Next Generation Application Protocol
NID	Network identifier
NR	New Radio
NRF	Network Repository Function
NSSF	Network Slice Selection Function

NSSP	Network Slice Selection Policy
PTP	Precision Time Protocol
(R)AN	(Radio) Access Network
RM	Registration Management
RG	Residential Gateway
RTT	Round Trip Time
SA NR	Standalone New Radio
SBA	Service Based Architecture
SBI	Service Based Interface
SCP	Service Communication Proxy
SDR	Software Define Radio
SEPP	Security Edge Protection Proxy
SF	Service Function
SFC	Service Function Chain
SMF	Session Management Function
SN	Sequence Number
S-NSSAI	Single Network Slice Selection Assistance Information
SUCI	Subscription Concealed Identifier
SUPI	Subscription Permanent Identifier
SV	Software Version
TA	Tracking Area

TAI	Tracking Area Identity
TCP	Transmission Control Protocol
TNLA	Transport Network Layer Association
TSC	Time Sensitive Communication
TSCAI	TSC Assistance Information
TSDSI	Telecommunications Standards Development Society
TSN	Time Sensitive Networking
TSS	Timing Synchronization Status
TTA	Telecommunications Technology Association
TTC	Telecommunications Technology Centre
TWIF	Trusted WLAN Interworking Function
UDM	Unified Data Management
UDR	Unified Data Repository
UL	Uplink
UPF	User Plane Function
URLLC	Ultra Reliable Low Latency Communication
USRP	Universal Software Radio Peripheral
Wi-Fi	Wireless access network

List of Network Functions and Entities (based on [1])

In addition to the abbreviations listed on the previous pages, what follows provides a list of the essential terms related to network functions and entities and service-based interfaces.

The 5G System architecture consists of the following network functions (NF):

- Authentication Server Function (AUSF).
- Access and Mobility Management Function (AMF).
- Data Network (DN), e.g. operator services, Internet access or 3rd party services.
- Unstructured Data Storage Function (UDSF).
- Network Exposure Function (NEF).
- Network Repository Function (NRF).
- Network Slice Admission Control Function (NSACF).
- Network Slice-specific and SNPN Authentication and Authorization Function (NSSAAF).
- Network Slice Selection Function (NSSF).
- Policy Control Function (PCF).
- Session Management Function (SMF).
- Unified Data Management (UDM).
- Unified Data Repository (UDR).
- User Plane Function (UPF).
- UE radio Capability Management Function (UCMF).
- Application Function (AF).

- User Equipment (UE).

- (Radio) Access Network ((R)AN).

- Network Data Analytics Function (NWDAF).

- Time Sensitive Networking AF (TSN AF).

The 5G System architecture also comprises the following network entities:

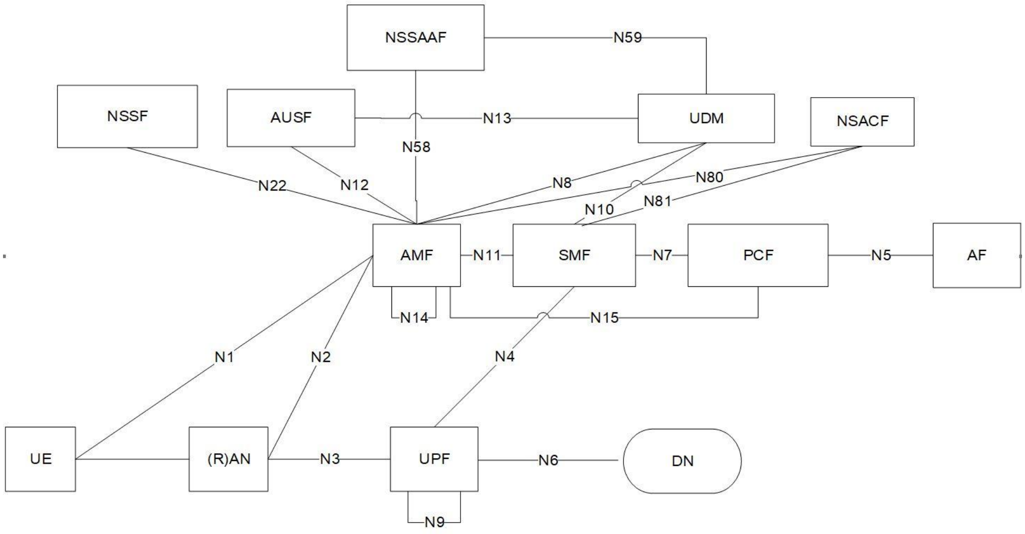
- Service Communication Proxy (SCP).

- Security Edge Protection Proxy (SEPP).

"The functional descriptions of these Network Functions and entities are specified in clause 6 under standard."

- Non-3GPP InterWorking Function (N3IWF).

- Trusted Non-3GPP Gateway Function (TNGF).



Additional Figure 1. Non-Roaming 5G System Architecture in reference point representation. From standard 23.501 [2].

Service-based interfaces (based on [2])

The 5G System Architecture contains the following service-based interfaces.

Pointed out only covered interfaces:

Namf: Service-based interface exhibited by AMF.

Nsmf: Service-based interface exhibited by SMF.

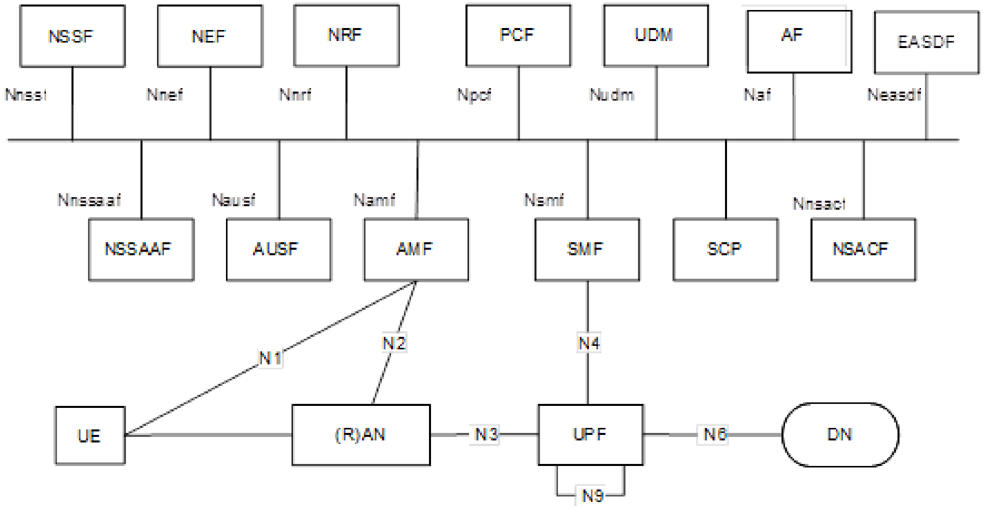
Npcf: Service-based interface exhibited by PCF.

Nudm: Service-based interface exhibited by UDM.

Nnrf: Service-based interface exhibited by NRF.

Nnssaaf: Service-based interface exhibited by NSSAAF.

Nausf: Service-based interface exhibited by AUSF.



Additional Figure 2. Non-Roaming 5G System Architecture. From [2].

Reference points [2]

The 5G System Architecture contains the following reference point, see Figure 1 [2]. Pointed out only covered reference point:

N1: Reference point between the UE and the AMF.

N2: Reference point between the (R)AN and the AMF.

N3: Reference point between the (R)AN and the UPF.

N4: Reference point between the SMF and the UPF.

N6: Reference point between the UPF and a Data Network.

N5: Reference point between the PCF and an AF or TSN AF.

N7: Reference point between the SMF and the PCF.

N8: Reference point between the UDM and the AMF.

N9: Reference point between two UPFs.

“The following reference points show the interactions that exist between the NF services in the NFs. These reference points are realized by corresponding NF service-based interfaces and by specifying the identified consumer and producer NF service as well as their interaction in order to realize a particular system procedure. [2]”

N10: Reference point between the UDM and the SMF.

N11: Reference point between the AMF and the SMF.

N12: Reference point between AMF and AUSF.

N13: Reference point between the UDM and Authentication Server function the AUSF.

N14: Reference point between two AMFs.

N15: Reference point between the PCF and the AMF in the case of non-roaming scenario, PCF in the visited network and AMF in the case of roaming scenario.

N22: Reference point between AMF and NSSF.

N58: Reference point between AMF and the NSSAAF.

N59: Reference point between UDM and the NSSAAF.

N60: Reference point between AUSF and NSWOF.

NOTE 8: The functionality of N60 reference point is defined in TS 33.501 [1].

N80: Reference point between AMF and NSACF.

N81: Reference point between SMF and NSACF.

N82: Reference point between NSACF and NEF.

N83: Reference point between AUSF and NSSAAF.

Table of contents

List of figures

List of tables

1. Introduction

5G is the fifth generation of mobile wireless technology. It is the successor of 4G. 5G is a key enabling technology that support’s today’s needs, such as every smart system including mobile phones, drones or vehicles that need a cellular network connection for example to provide autonomy, better positioning resolution, or to transfer more information faster. Looking back in time (see Figure 1), the evolution of cellular network started in the 80s where 1G was developed. It took decades to evolve to 2G, then 3G, and for the last 10 years 4G, which is nowadays widely deployed and very reliable. The main differences come not only about speed and information transfer capacity, where 4G could reach up to 100 Mbps and now 5G could reach up to 20 Gbps, but also in better service, more reliable, higher capacity, and lower latency [3].

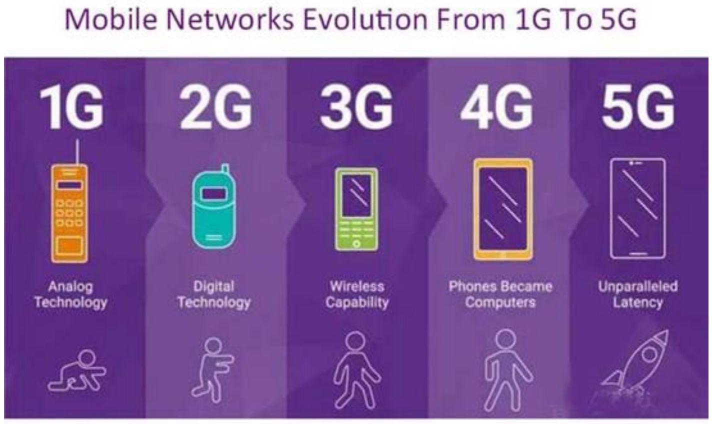


Figure 1. Evolution of mobile networks from 1G to 5G [4].

Wireless internet first became available from 3G, then widely used with 4G and now moving on to new performance levels and usage with 5G.

Internet access providers and mobile network operators transmit data over optical networks for the so-called core network and over radio waves at the edge of the network (referred to as the radio access network (RAN)). User’s phones or other devices known as user equipment (UE) pick up the waves and get connected to the RAN and subsequently to the 5G core network (5GC). Users can collect information from the internet; the faster over the cellular network, the better for the user.

The above advancements are grouped under three pillars that are referred to as the 5G triangle, as seen in Figure 2 [3]:

1) Higher bandwidth, referred to as enhanced Mobile Broadband (eMBB);

2) Lower latency, referred to as Ultra-Reliable Low-Latency Communication (uRLLC);

3) Types and number of connections, referred to as Massive Machine-Type Communication (mMTC).

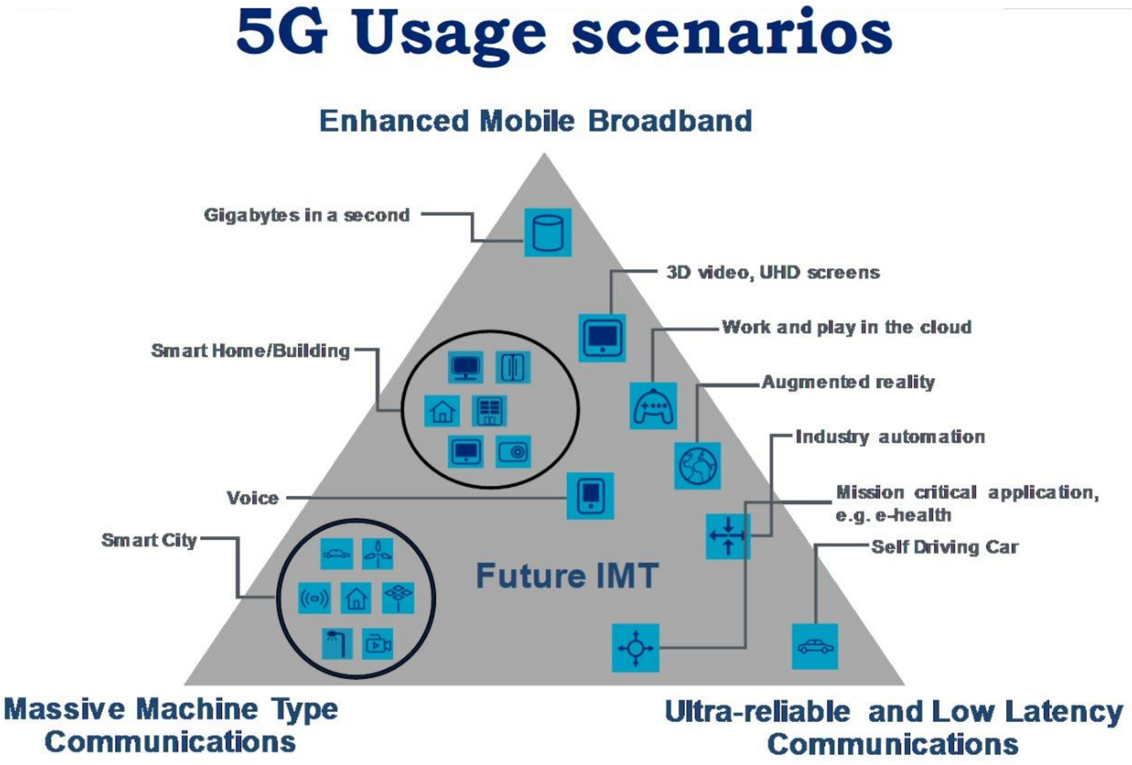

5G Usage scenarios
Enhanced Mobile Broadband
Gigabytes in a second
3D video, UHD screens
Work and play in the cloud
Smart Home/Building
Augmented reality
Industry automation
Voice
Mission critical application, e.g. e-health
Smart City
Self Driving Car
Future IMT
Massive Machine Type Communications
Ultra-reliable and Low Latency Communications

Figure 2. Illustration of the 5G triangle with the three pillars: enhanced Mobile Broadband (eMBB), Ultra-Reliable Low-Latency Communication (uRLLC), and Massive Machine-Type Communication (mMTC) [3].

The advantages of this “more” are to provide users with information in higher precision or resolution faster, for example to make live view streaming over the cellular network, or to give sensors more reliable information and repair accuracy. This accuracy provides the possibility to move or work on live view, for example telemedicine and telesurgery.

At the moment, many 5G networks are operated in non-standalone (NSA) mode (i.e. 5G RAN but 4G based core network). But more and more 5G networks are being deployed in Stand Alone (SA) mode (i.e. combining 5G RAN and 5GC) to deploy “real” 5G systems and deliver its promises. Although 5G commercial deployments have been ongoing for a few years already, the technology is still relatively new and requires teaching it at universities and other educational organizations. In particular, TalTech identified the need for setting up a practical training environment for 5G and supports this via its 5G Open Core Network (5G-OCN) project[1] based on Fraunhofer O5GC. In this context, open-source software and hardware are essential tools for training purposes because they allow studying, understanding and experimenting beyond what is typically possible with commercial closed-source tools.

One of the goals of this thesis is to provide an overview of a 5G system, i.e. its main elements, their interconnections, models, and functions. For practical purposes, it was decided to focus on how to register a UE (e.g. mobile phone or modem) to the network. The output of the work shall be practical procedures that help students or other people to understand easily how (UEs) can register and connect to the network.

To implement the above, different open 5G core network testbeds have been considered. In the thesis topic, Fraunhofer O5GC testbed (see its architecture in Figure 3) had already been selected, as part of the 5G-OCN project, in view of showing a practical working solution. O5GC is licensed and available at TalTech, and it has many different features suitable for the practical work described above. That said, alternatives have also been studied and a short comparison is provided later in Table 1 in Section 2.2. It should be mentioned that the information presented in the table is as of 01.12.2023; different open testbeds are still being released, meaning that in the near future there might be more options to compare and consider.

[1] 5G-OCN has received funding from TalTech Educational Development Fund / Õppearendusfond (project number SÕF23031) and from Thomas Johann Seebeck Department of Electronics.

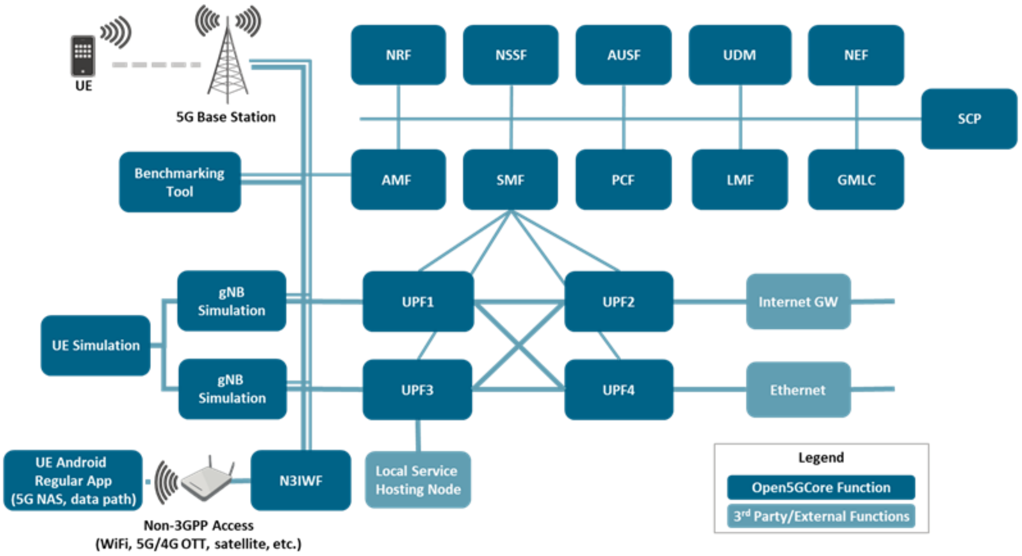


Figure 3. Fraunhofer O5GC architecture diagram [5].

For illustration purposes in this introduction, Figure 3 shows the O5GC architecture diagram. In short, the O5GC architecture can be divided in three groups: UE, 5G Base Station (known as gNode B (gNB)), and 5GC. The 5GC consists of all the elements that are visible in Figure 3, except for the UE, UE simulation, 5G base station and gNB simulation[2]. This thesis is focused on the 5GC functions and UE connections to the network.

1.1 Research statement

The challenge addressed in this thesis is two-fold, as described in what follows.

1) First there is a need for understanding and documenting the 5G ecosystem, its architecture and its main operating principles. In particular, the O5GCore functions are numerous and hard to follow. Although the O5GCore simulator operates as the 3GPP release 17 TS 23.502 standard describes [6], even if it supports only release 15 and 16, it also includes more detailed information related to its implementation that are not included in the standard, which may be confusing for students and other learners. A detailed analysis and comparison of the relevant parts of the 3GPP standard and live execution of the O5GCore

[2] O5GC also supports non-3GPP access (e.g. WiFi, satellite, etc.); this feature is outside the scope of this thesis.

simulator is therefore needed. This thesis aims to pave the way for developing and documenting this understanding.

2) Secondly, there is also a need to develop new practical laboratory exercise for guiding learners to gain an understanding of the above and to develop their practical skills in getting started and operate the O5GCore simulator, in particular how to connect the UE to the network and understand and experiment with the different functions of the 5G core. This thesis aims to develop and document an example of laboratory instructions.

In order to achieve the above, the following tasks and steps have been taken.

1. Gain an understanding of the 5G system in general, its architecture and its theoretical working principles.
2. Get acquainted with the O5GC environment, especially the 5GC functions and protocols (Service Based Architecture (SBA), Non-access Stratum (NAS), NGAP, GTP, etc.) and how they match and follow the 3GPP Release 15, TS 23.501 standard.
3. Analyse O5GC suitability for developing a new training and carry out experiments with O5GC simulator to learn how to connect the UE to the network.
4. Prepare one practical lab that could be carried out with the O5GC simulator, including preparatory work, guidelines, and possible solutions.

1.2 Thesis organization

The rest of this thesis is organized as follows:

- Chapter 2 provides an overview of related works;
- Chapter 3 provides an overview of the gained knowledge of the relevant parts of the 3GPP standard, the O5GC simulator, its “Call flow”, and the gained know-how for connecting a UE to the network;
- Chapter 4 describes the example laboratory work developed in this thesis;
- Chapter 5 describes O5GC typical problems and errors encountered in this work;
- Chapter 6 is concludes the thesis with a summary and a few perspectives;
- Appendix 1 presents additional information related to O5GC procedures;

- Appendix 2 presents additional information related to errors figures. Appendix 2 presents UE1 info;
- Appendix 3 – presents gNB information;
- Appendix 4 – AMF information;
- Appendix 5 – Shows terminal command and VirtualBox view;
- Appendix 6 – UE1.json original configuration fileAppendix 7 – UE2 configuration file in terminal command;
- Appendix 8 – UE2 changed configuration file by the author;
- Appendix 9 – Wireshark preview info: initial registration, etc.

2. Overview of Related Works and Background Information

This chapter provides an overview of works related to the deployment and evaluation of open source 5GCs. The chapter begins with a summary of selected research papers that have deployed and/or evaluated testbeds built upon open source 5GCs; the contributions of the cited references are presented individually, while their pros and cons are collectively summarized at the end of Section 2.1. Then, a comparison of four selected open source 5GCs (O5GCore, Open5GS, OAI5G Core Network, and Free5GC) is provided in Section 2.2, highlighting their respective features, advantages, and drawbacks. Some background information about O5GCore and 3GPP 5G standards are provided in Section 2.3 and Section 2.4, respectively.

2.1 Related works overview

In [7] two distinct testbeds are built upon open source-based solutions for 5G RAN and 5GC operations. The first testbed is referred to as a portable network that uses a laptop (Intel i7 10750H (6 cores), 16 GB RAM) running i) OAI 5GC for the 5G core part and ii) OAI 5G RAN in conjunction with an entry-level USRP B210 SDR board for the 5G RAN radio part. The second testbed is referred to as a full-scale network, i.e. a campus-based 5G network with a server (Intel Xeon Gold 5315Y (8 cores), 64 GB RAM) running i) Open5GS for the 5G core part and ii) srsRAN in conjunction with a higher-end USRP X300 SDR board for the 5G RAN radio part. The performance of the two testbeds is evaluated in terms of i) network connectivity by ensuring a UE can connect to the mobile network (obtaining an IP address and a globally unique temporary identifier (GUTI)), ii) computing capacity utilization (CPU and memory) under background software suites and active network connection, and iii) numerical performance (link capacity, latency, jitter, and packet loss). Results from a small set of experiments with both testbeds report that the RAN components use significantly more CPU and memory resource than the core network functions, indicating its higher resource-intensive nature. Averaged performance numbers are: DL of 94-98 Mbps, UL of 4.79 Mbps, median delay of 25-30 ms, stable jitter (1 ms), and null packet loss ratio.

The work presented in [8] analyses the performance of different open source implementations of the 5G network core (free5GC v3.0.6 and Open5GS v2.3.6) for large-scale setups, focusing on i) user equipment registration, and ii) session establishment and streaming user data over parallel data plane connections. For session establishment and registration time tests, a virtual machine (VM) configuration (Intel Core i7 8700T @ 2.4 GHz with 12 virtual cores and 8 GB RAM) was used. The tests revealed limitations, i.e. the failure of free5GC v3.0.6 to connect more than 10 UEs, the failure of OAI v1.4.0 after connecting 15 UEs, and the failure of Open5GS v2.3.6 which could connect 1024 devices simultaneously before crashing the Access and Mobility Management Function (AMF) network function. Their next experiments are limited to 11 gNBs, resulting in a total of 1100 connected UEs, reaching the Open5GS limitation.

For connection intervals of 500 ms, free5GC had average connection time of 1089 ms (median and maximum of 853 ms and 5.57 s, respectively). For connection interval of 100 ms, free5GC' average connection time was 7731 ms (median and maximum of 3027 ms and 54 s, respectively). Correspondingly, for connection interval of 500 ms, Open5GS had an average connection time of 306 ms (median and maximum of 305 ms and 345, respectively). For 100 ms connection interval, Open5GS' average connection time was 364 ms (median and maximum of 358 ms and 490 ms, respectively).

For data plan experiments, other VM configurations were used (Intel Core i7 8700T @ 2.4 GHz with 6 cores and 4 GB RAM, as well as 8 and 12 cores (8 GB RAM each). With 12 virtual cores and 8 GB RAM, the results showed that free5GC had a higher average bit rate for each execution, i.e. 338.4 Mbps with 1 UE, 777 Mbps with 4 UEs, and 1001.1 Mbps with 8 UEs. On the other hand, Open5GS had a lower average bit rate for one connected UE, i.e. 314.5 Mbps, and reducing to 262.7 Mbps with 4 UEs and 241.1 Mbps with 8 UEs. CPU and RAM resource utilization was also analysed, but only for the smallest VM version (4 virtual cores and 4 GB RAM). With free5GS, CPU usage was close to 100% (91.87% for testers and UEs, 0.17% for free5GC, and remaining percentage for other containers), indicating that the free5GS uses minimal CPU resources. With Open5GS, the total average CPU usage was 93.75% (60.24% and 22.1% for the tester andOpen5GS, respectively). These difference are attributed to Open5GS's lower bit rates. Moreover, the results indicate that free5GC likely uses parallelization to manage data plane processing of multiple UEs, contrary to Open5GS.

To sum-up, their results show that free5GC offers better performance in data plane bit rates, while Open5GS shows better stability during device registration.
In [9], several 4G and 5G network testbeds based on open source software stacks were implemented and evaluated, including 5G NSA and 5G SA. Two versions of the 5G NSA were i) X300 SDR board, srsRAN, and srsEPC CN, and ii) B210 SDR board and OAI RAN and CN stack. The 5G SA was based on an X300 SDR board, srsRAN gNB, and Open5GS. Regarding the 5G NSA performance, as srsEPC supports only 50 prbs and 10 MHz bandwidth, it had an average of approximately 39 Mbps DL and 10 Mbps UL. On the other hand, the SDR-OAI based implementation achieved average of approximately 85 Mbps DL and 35 Mbps UL thanks to its support for 106 prbs. Moreover, e2e latency with srsEPC was 11.8 ms, whereas OAI achieved approximately 5 ms (i.e. ultra-low latency below 10 ms). Next, regarding the 5G SA (including Open5GS), it achieved an average of approximately 30 Mbps and 5 Mbps DL and UL, respectively. This is attributed to the limitation of the used srsRAN gNB implementation (support for only 10 MHz bandwidth in 5G SA mode), and the possibly non-optimal manual tuning and placement of the base station antennas. The 5G SA e2e latency was 12 ms.

In [10], the authors of [9] extended their evaluation of open source based 5G testbeds to a campus network. They experimented with four combinations, i.e. 1) srsRAN and Open5GS CN, 2) OAI-RAN and Open5GS CN, 3) srsRAN and Free5GC CN, and 4) OAI-RAN and Free5GC CN. The RAN radio hardware part was an entry-level B210 SDR board. Commercial UEs included Google Pixel5, Asus Zenfone9, and Nokia X30 phones. The srsRAN-Open5GS achieved average DL and UL of 26.03 Mbps and 16.73 Mbps, respectively. srsRAN-Free5GC achieved average DL and UL of 26.64 Mbps and 13.85 Mbps, respectively. The average e2e latency for srsRAN-Open5GS was 58.85 ms, and that of srsRAN-Free5GC was 63.47 ms. The OAI-Open5GS achieved average DL and UL of 45.05 Mbps and 4.45 Mbps, respectively. OAI-Free5GC achieved average DL and UL of 49.70 Mbps and 5.02 Mbps, respectively. OAI-Open5GS' average e2e latency was 32.65 ms; that of OAI-Free5GC was 32.30 ms.

In [11], a 5G NSA network was deployed on the basis of the OAI stack and USRP B210 boards. Linux containers were used for both the core and RAN parts of the network, implemented on a PC with an Intel Core i7-12650H processor (10 cores, 16 GB RAM).

2 B210 SDR boards (for the eNodeB and gNodeB) were used for the RAN radio hardware. Indoor measurements with a Samsung Galaxy A52s 5G phone shown an average latency of approximately 12 ms (minimum 7.33 ms), UL of 7 Mbps and DL of 125 Mbps.

The paper [12] presents the deployment of a Software Defined Network (SDN) SA 5G testbed based on open-source projects, namely Open5GS for the CN, UERANSIM and srsRAN for the RAN, VMware ESXi for the virtual infrastructure manager (VIM) and hypervisor, and Open Daylight for the SDN controller. A first scenario used UERANSIM as the UE and RAN simulator with Open5GS as the 5GC. Two network slices were created and declared into AMF, network slice selection function (NSSF), session management function (SMF), user plane function (UPF), and gNB configuration files. A second scenario used srsRAN for the UE and RAN simulator with Open5GS as the 5GC; only a single gNB and UE are supported, leading to testing the setup with one UE. Scenario 1 (UERANSIM) resulted in DL ranging in 98.27-112.82 Mbps and UL in 111-131.68 Mbps. Network slicing was verified by configuring the first UE to 20 Mbps for DL and UL and the second one to 10 Mbps for DL and UL, effectively achieving 21.72 Mbps DL and 22.14 Mbps UL, and 10.79 Mbps DL and 11.24 Mbps UL, respectively. Scenario 2 (srsRAN) resulted in significantly higher average DL of 582.67 Mbps and average UL of 281.07 Mbps owing to srsRAN use of ZeroMQ messaging library[3] that provides asynchronous communication, multithreading, high-performance transport mechanisms, and high-throughput communication design.

Another deployment of a SA 5G network based on open source software is presented in [13]. Their setup consists of a server with a Ryzen Threadripper 3970X with 64 CPUs and 256 GB RAM, open5GS (v2.6.4) core network and srsRAN (v23.10) RAN via ZeroMQ (the RAN radio is a B210 SDR board), as well as three different UEs, i.e. a 4G UE from srsRAN, a laptop with a Simcom SIM8262E-M2 5G modem, and an Oneplus Nord CE 2 Lite 5G phone. Indoor measurements shown a range of approximately 20 meters (connection got lost at approximately -110 dBm to -120 dBm), DL ranging between 65-144 Mbsp and UL between 27-56 Mbps depending on the frame structure,

[3] https://zeromq.org/

and latency (RTT) ranging between 15.7-17.3 ms. The paper notes that the open5GS CN is well-developed for deploying private 5G SA networks, supporting multi-slice networks, but suggests that adding network exposure functions (NEF) would enable third-party applications to access network parameters or configure networks using AI/ML techniques. Additionally, incorporating a location management function (LMF) would allow exploring different UE-based and UE-assisted 5G positioning techniques.

In [14], two 5GC testbeds are presented, and some learned lessons are shared. Both are based on Aether [15] for the 5GC software and UERANSIM for the RAN simulator. A first PC runs UERANSIM in a Docker Compose environment on an Intel Core i3 processor with 8 GB RAM. A second PC runs the Aether in the Kubernetes environment on an Intel Xeon processor and 12 GB RAM. Each PC has two separate network interfaces, one for internet access and control plane communication between RAN and 5GC, and the other for data plane communication. This testbed is moderately powerful as it was initially designed for educational purposes, but its computing capacity can be increased by adding more RAN nodes or deploying 5G control plane and user plane functions in different machines. The second testbed enhances the first one with five VMs running on a server with two Intel Xeon processors (40 Core CPU) and 128 GB RAM; the RAN, 5GC, and Multi-access Edge Computing (MEC) servers are deployed in separate networks that are connected via two virtual routers. While the paper does not present performance evaluation results, it does share some lessons learned. Firstly, the authors highlight that as 5G transitions to cloud-native systems, deploying a 5G testbed in a container-based environment is deemed paramount. Docker Compose and Kubernetes are popular container orchestration frameworks for this purpose. Docker Compose is easy to use, allowing for quick deployment of components. However, it is designed for single-host deployment, limiting system scalability. Kubernetes, on the other hand, can deploy and manage containers across multiple nodes, offering advanced capabilities like automatic scaling and resource management. They can be combined in certain environments, e.g. Docker Compose for simplifying RAN deployment and Kubernetes for enabling scalability in 5GC deployment. Secondly, the authors highlight that understanding the network of a 5G system and its alignment with container platforms is essential for building a 5G testbed. Using different network interfaces or physical networks for the control plane and the data plane is recommended to avoid conflicting configurations. The communication between RAN and 5GC, or between Internet and UPF, takes place between the outside of the Kubernetes cluster and components inside, unlike the communication among network functions in the 5GC that take place inside Kurbernetes.

Finally, the work presented in [16] provides a qualitative and quantitative analysis of three open source 5GCs, i.e. free5GC, OAI 5G CN, and Open5GS. First, the paper highlights the importance of the licensing schemes, as they bring different restrictions for the use and distribution of open-source software. The free5GC project uses an Apache 2.0 license, while Open5GS uses AGPL-3.0. Copyleft licenses are essential for open-source research, as modified software must be publicly available. However, commercial distribution of modified software is a constraint. OAI 5G CN is based on a heterogeneous license called OAI Public License V1.1, which is based on the Apache 2.0 license for research and non-profit usage and extended for commercial use. This makes it feasible for large companies to contribute to these projects without conflicts with their own patents. Second, regarding the programming languages, the papers reports that OAI 5G CN and Open5GS are hardware-close (C/C++) and free5GC is based on Golang. All infrastructures are deployed as VMs or container-based services, with free5GC and Open5GS offering adaptability for VMs and container deployment. OAI 5G CN supports only the container environment. For free5GC, the entire stack is restarted in case of failure, while container environments allow individual network functions to be restarted while maintaining other services. Third, regarding what the paper calls "relevance" (based on the number of scientific publications), OAI 5G CN appears as the most relevant, followed by Open5GS and free5GC. Open5GS and free5GC have gained attention in recent years due to their feasibility for deploying CN as a virtual machine and container service. However, the authors deem that OAI 5G CN, Open5GS, and Free5GC will remain relevant to acknowledge growing research works. Fourth, regarding resource utilization (CPU average load with 4 cores vs. 2, 4, and 8 UEs), the average CPU load ranges approximately between 2-4%, 3.5-5%, and 1.4-2.5%, for free5GC, OAI 5G CN, and Open5GS respectively. For two UEs, Open5GS is less resources hungry than OAI 5G CN and free5GC. Resource usage grows higher for OAI 5G CN with the number of UEs. Open5GS resource load remains stable. It was also found that the behaviour of free5GC varies with six UEs, starting with a high load caused by the registration of the end-user before gradually decreasing and remaining stable. Finally, regarding the round-trip time (RTT), the average values for free5GC, OAI 5G CN, and Open5GS range between 23.7-25.6 ms, 15.6-35 ms, and 23.23-24.9 ms, respectively. The results indicate a trade-off between CPU load and latency (here RTT). OAI 5G CN's CPU load gets larger when providing better performance, while Open5GS is consistent with UE scaling, and free5GC is similar to Open5GS but with small CPU load fluctuations.

2.1 Conclusion on the selected related works

The selected works introduced above clearly show the feasibility of using open source software for deploying 5G networks in academic environments. It should be noted that [13] provides a more detailed explanations of the principles of operations of the different components of the 5G network; this provides a quick reference useful for training/teaching purposes. Moreover, the majority of the papers presented performance numbers (e.g. DL and UL rates, latency, and/or CPU load), but they did not describe the installation and configuration procedures in details. An exception is [14] that provides detailed installation guidelines for their two testbeds as they are publicly available in a GitHub repository [17]. That said, it is clear that further work is needed for preparing material for teaching open source based 5G networks deployment and assessment, in line with the theme of this thesis.

Moreover, the papers illustrate that a certain degree of diversity and options exist in the selection of the open source components, including for the 5GC. However, it was also noted that Open5GCore (pre-assigned for this thesis) is not used in any of the discussed papers, possibly due to the fact that, although open source, it is not available for free (a paid license is needed for obtaining the binaries, and an additional paid licence is needed to get the source code). However, Open5GCore has several advantages (including its simulator) over the free open 5GCs, as discussed in the following section.

2.2 Comparison of four selected open source 5G cores

5G is the fifth-generation standard for cellular communication networks, of which the commercial deployment started in 2019. However, as of today, this technology is still being deployed and each new release from the 3rd Generation Partnership Project (3GPP) standard brings its share of new features. The complexity of 5G, its cost, and its continuous development also means that as of today, there is need to develop and strengthen the know-how and practical experience with 5G technologies in the Estonian universities. TalTech is currently planning a new course and trainings in 5G that will exploit the Open5GCore based testbed. This new course's plan is to work out a new theoretical and practical setup where the network core (i.e. an IT solution) and the RAN part are connected together. The goal is to provide both communication specialists and radio specialists a common theoretical overview of 5G and a practical part where they come together and help each other, i.e. how network core (IT) specialists support radio specialists, and vice-versa.

As a first step to address the above, TalTech has deployed its 5G Academic Network that provides industry-grade performance for R&D projects. However, vendors' technical restrictions and commercial priorities mean that this network does not provide all the latest 3GPP standards' features nor access to specific interfaces and source-code. For example, Ericsson and Nokia have commercial solutions, but their detailed documentations and related knowledge are not publicly available. Their source code is not open because every company protect their knowledge and intellectual properties.

To overcome these limitations, an open 5G testbed is required as a complementary resource to TalTech's 5G Academic Network, especially for teaching and training purposes. TalTech already has 5G open source components such as OAI open source software stack, ETTUS USRP devices (of which the hardware and firmware are also open source), several RAN open source software stacks, and an open 5GC, namely Open5GCore (O5GC).

Although using O5GCore is part of this thesis assignment and must be used, the author still considered today's needs and that of the foreseeable future, and four testbeds (the ones that are assessed as the most suitable for this work) have been researched and compared to each other, as can be seen in Table 1 and Table 2.

Table 1 and Table 2 were generated in December 2023.

Table 1. Comparison of four selected open 5GC testbeds.

They are arranged in decreasing order of features from left to right.

Features	O5GCore [5]	Open5GS [18]	OAI5G Core network [19]	Free5GC [20]
Handover	Yes	Yes	Yes	Yes
VoNR	Yes	Yes	No	No
SMSF	Yes	Yes	No	No
Roaming (SCP, SEPP)	Yes*	Yes	No	No
Positioning (LMF, GMLC)	Yes	No	Yes**	No
Non 3GPP access (N3IWF, TNGW)	Yes	No	No	Yes
Release 15 and Release 16	Yes	Yes	Yes	Yes
Prototypes and COTS phones	Yes	No	Yes	No
Android app possibilities	Yes	No	No	No
Multiple slice possibilities (URSP´s)	Yes	Yes	Yes***	No
Simulation (UE, gNB)	Yes	Yes****	Yes	Yes
Open RAN	Yes	No	Yes	No
Monitoring capabilities, benchmark	Yes	No	No	No
Time sensitive network (TSN)	Yes	No	Yes	No
6G ready	Yes	No	No	No

* Security Edge Protection Proxy (SEPP) – is replaced with HTTPS

** Supported NRPPa interface for LMF

*** Network slicing partially supported

**** Can use 3rd party simulator UE and gNB.

The compared information was obtained from the 5GCs providers' homepages and additionally sought from Internet specialized forums or technical documents because some of those homepages do not focus on the 5GCs features and do not provide their detailed capabilities.

The main advantages and disadvantages of the compared open 5GCs are summarized in Table 2.

Table 2. Main advantages and disadvantages of the four compared Open 5GCs.

	O5GCore	Open5GS	OAI5G Core network	Free5GC
Advantages	Many features not available in the other testbeds *(in particular location functionality (LMF, GMLC))*, and e.g. TNGW, URSP, TSN, benchmark, Android app, 6G ready). Also, its developer has over 20 years' experience in providing testbed platforms for the R&D activities of the telecommunication industry. Can be used to clone and customize own testbed for specific requirements. Paid licenses are more reliably developed on the long term than free ones.	Free (paid license no needed). Easier to handle than O5GCore, have good documentation and manual.	Free (paid license no needed). It is possible to connect Free5GC here. Has its own labkit (hardware and software package) named OAIBOX [21].	Free (paid license no needed)
Dis-advantages	Paid license is needed. Physical labkit is not available.	Does not include all the features in Table 1. Physical labkit is not available	Does not include all the features in Table 1. Labkit is not free.	Does not include all the features in Table 1. No labkit.

2.3 Background information about O5GC

Given that O5GC must be used in this master work, what follows provides some background information about it (more details are presented later in Section 3.2). Although O5GC is a 5GC, it can simulate UE connection to the network and provide an overview of the processes performed. O5GC can provide feedback, and the user can collect and analyse the data to understand the different states that have occurred. The provided overview shows what kind of functions in the 5GC have been implemented in O5GC. During this master work, there were two options to analyse and test the simulator provided with O5GC. The first option is to use its UE simulator and gNB simulator and perform the tests through the "Call flow", which should be configured during the simulation process. Although it is possible to use Wireshark and analyse the collected data, the author does not cover Wireshark analysis in this thesis as it is beyond the scope. It is only briefly analysed how to find related information for O5GC using Wireshark.

The second option is to use a real UE such as a 5G mobile phone and SIM card, or a SDR board and SIM card, to test through the "Call flow". Although a commercial RAN was already located at TalTech, it would have needed some configuration from its vendor to make it work correctly; the author does not cover this because this would have required time, expertise, and cooperation with the vendor, which were beyond the scope of this thesis. In Figure 4, the author brings out with red circles the O5GC' functions that are used to establish connections with the UEs. The yellow circles represent optional elements that were initially considered if there would have been enough time and knowledge to configure them.

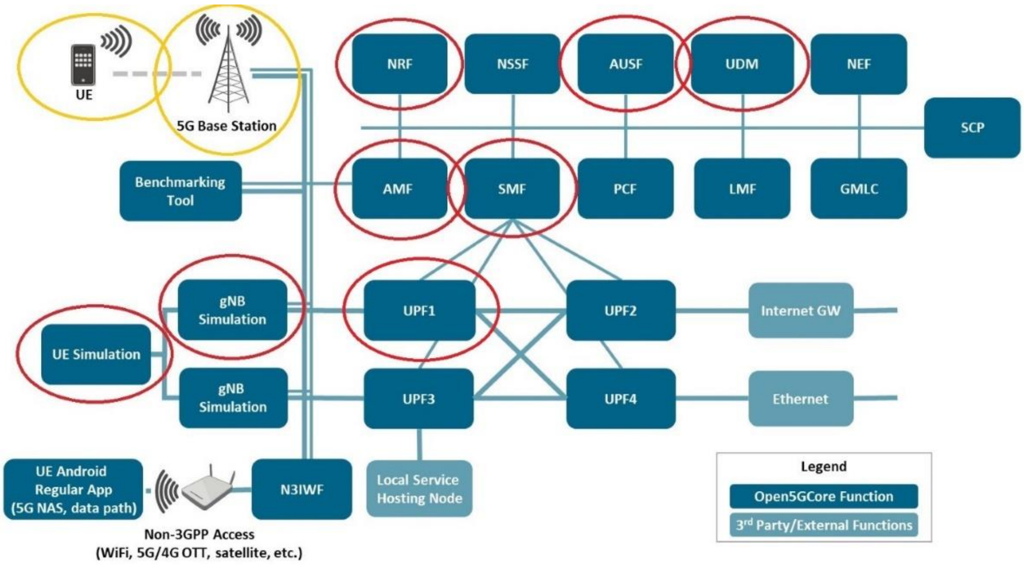


Figure 4. O5GC architecture. The functions used by the author are marked in red circles. Elements in yellow circles are optional, and eventually were not covered in this thesis.

An advantage of O5GC is that the messages are opened and easily readable, where as in real life most information is encrypted and not understandable. Moreover, the simulator operates under ideal conditions with no interferences involved, which as a first step allows verifying the main principles of operations without having to troubleshoot.

2.4 Some background information about 3GPP 5G standards

In general, standards describe telecommunication technologies specifications, including radio access to network and service capabilities, and provides complete system description of how and what is allowed to perform in the 5G core system. When it comes to 5G, ITU provided the requirements [22] and 3GPP provides many SA and RAN study items that have been successfully completed in 3GPP. O5GC already implements 3GPP Release-15 and Release-16, and new relevant work items in the normative phase have been selected in both TSG SA and RAN for 3GPP Release-17 [23] and upcoming 3GPP Release-18.

In particular, the system architecture for 5G system (5GS) is described in 3GPP standard TS 23.501[2], providing a diagram overview. It is a Non-Roaming 5GS in reference point representation. Procedures for the 5G system are described in 3GPP standard TS 23.502

[23] that gives step-by-step procedures for how to register a UE device to the network. For example release 18 document contains 899 pages, covering many aspects in detail, meaning that it is not straightforward to understand the whole procedure and it is needed to focus on the relevant parts. Those parts relevant to this thesis are mentioned in the next chapter.

This chapter has presented a summary of selected related research papers that built upon open source 5G cores, concluding that open source 5G deployments are possible but that teaching and learning material remains to be developed. The chapter also provided a comparison of O5GCore, Open5GS, OAI5G Core Network, and Free5GC, highlighting their respective features, advantages, and drawbacks. The chapter also provided some background information about O5GCore and the related 3GPP standards.

The next chapter provides a more detailed overview of the relevant 3GPP 5G standard and O5GC system.

3. Overview of 3GPP 5G Standard and O5GC System

In this chapter, the author describes the relevant parts of the 3GPP 5G standard and the O5GCore itself. For both parts, there is a focus on the UE registration “Call flow”. Moreover, for the O5GCore part, the author provides practical information for experimenting with this call flow with the simulator.

3.1 5G standard overview

“3GPP, the 3rd Generation Partnership Project, was established in 1998 (during the 2G times) to start developing the specifications of 3G cellular network in a standard. This includes radio access, core network and service capabilities. It provides a complete system description for mobile telecommunications. 3GPP unites seven national or regional telecommunications standard development organizations (ARIB, ATIS, CCSA, ETSI, TSDSI, TTA, TTC). They are also known as “Organizational Partners””. They provide their members, for example in Estonia Telia, Tele2 and Elisa, with a stable environment to access and/or contribute to the Reports and Specifications that define 3GPP technologies. Under different releases is possible to find specific standard descriptions (see Figure 5). At the time of writing, the ongoing one is Release 19 which is not completed yet. Release 19 is starting to develop 6G from 5G, to advanced it further. Back in 2018 was released Release 15 which started the first phase of 5G specification and today Release 18 has defined 5G-Advanced, as the mid-point of 5G standardization. Release 18 is not deployed yet [24]. 5G Advanced (5.5G) is faster than 5G, as was 4G versus 4.5G in its time. Thus, 5G Advanced is a bridge to 6G. It should also be noted that there is often several months between a release is frozen and when it is implemented in commercial devices. It should also be noted that not all features are necessarily implemented.

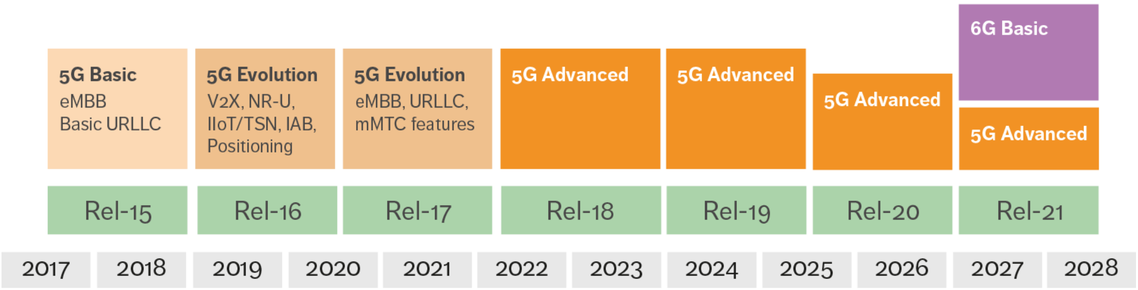


Figure 5. Timeline of 3GPP 5G standardisation releases [25].

This thesis has used two main releases, i.e. Release 17 (3GPP TS 23.501) and Release 18 (3GPP 23.502 v18.3.0 (2023-09)). TS 23.501 describes the system architecture for the 5G system and TS 23.502 describes the procedures for the 5G system. These two standards releases also give information for checking additional standards where to find more detailed information. Different releases from 15 to 18 have been checked and release 17 and 18 gave a lot of information more than 15, but build up is the same.

The architecture standard gives an overview how cellular network works and the 5G system standard describes specifically 5G functions and their meanings. For example, a UE is user equipment such as a mobile phone and other IoT devices and AMF is access and mobility functions that gives access to UE to network and makes needed request for that. All the functions are described additionally under network functions and entities. These two functions are related to other parts of the 3GPP standards, and they are also referred to by the other standards.

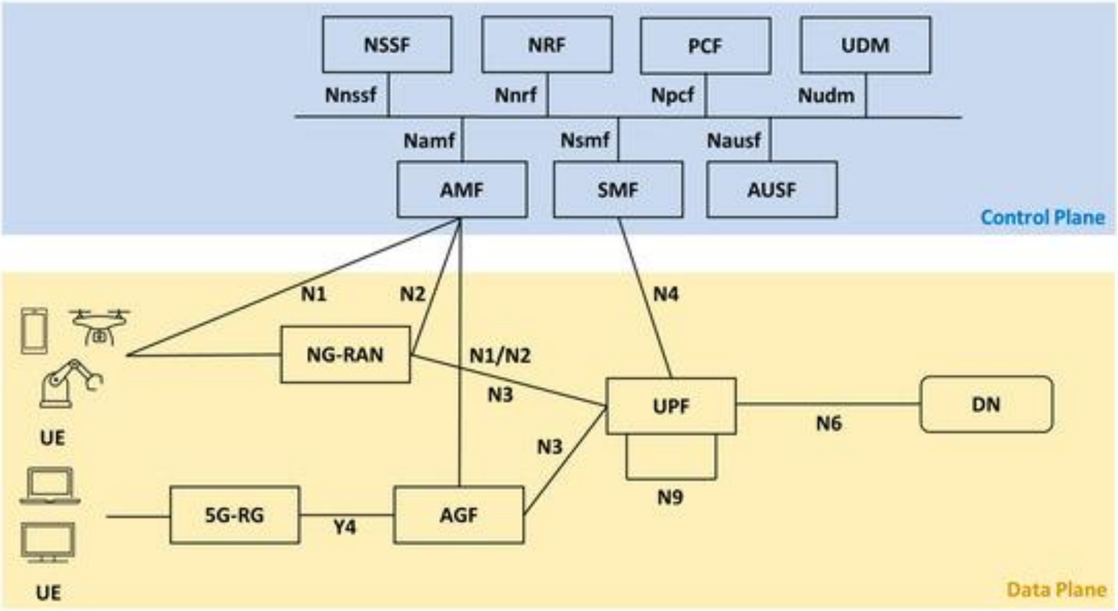


Figure 6. Network functions and entities [26].

The 5G architecture network functions (NF) are divided into two different planes. The main core functions are under the control plane and other parts UE, RAN, UPF and internet are in the data plane. See Figure 6 where the blue part details the control plane and the yellow part details the data plane. In the figure, it is visible that under control plane is described service functions, such as Namf. Service functions are used to make connection between different functions. For example, if AMF wants to access UDM, then it uses Nudm service function. Under data plane are described connections between UE to data network (DN). It means that the main core in the control plane is not involved anymore after registration and PDU session establishment. The UE can access the internet via UPF. Under data plane is also described point to point interfaces, for example named N1 which is the connection point for UE to AMF. These N points use dedicated protocols or API-based interfaces to make the connection. For example, if AMF wants to access to UDM, then it uses N8 connection points, See additional figure 1. It also needs to remind that HTTP/2 is used for signalling between functions. This is used to provide multiple requests and faster communication and reducing latency on a single TCP connection, etc.

What follows defined the core functions used under the simulator:

UE can be any user device or IoT device, such as mobile phone, drone, self-driving car, computer, sensors, etc.

NG-RAN is the 5G new radio access network. The base station also known as gNB, consists of the antenna and electronics together. It is divided into central unit (CU) and distribution unit (DU). gNB is cell tower between UE and core system. It delivers NAS (N1) messages to AMF.

NGAP is used between gNB and AMF to configure gNB and also to send and receive NAS messages. NAS messages can be MM and CM status information, registration management, authentication and security algorithms.

AMF is in charge of connection and mobility management duties. Also, it gives integrity protection, subscribes authentication, supports location services, etc. All information for connection and session related is involved in the AMF.

NRF is a central repositor for all network functions. It allows NFs to be registered and recognized.

AUSF is managing UE authentication. It decides if a UE can access the network or not.

UDM stores all listed UEs profiles and indicates to AUSF that a UE is on the list or not. For example, O5GC provides SUPI code. SUPI is stored in UDM.

SMF is responsible for session management, for example PDU session. It can create, update and terminate PDU sessions. It can allocate UE IP addresses, select UPF, and makes connection with the PCF.

UPF handles UE data traffic by forwarding data packets. It is also a connection tunnel from UE via gNB to internet.

Other functions are not used in this thesis, because of initial registration and under simulation it is not needed to use them; however, they are described in what follows for information purposes.

NSSF selects network slices that are provided to UE.

NEF provides securely expose services between UE and 3rd party or internal devices.

PCF establishes policy rules by controlling NFs, for example roaming and slicing.

LMF is location function to manage UE location to access UE location information.

GMLC is also location service function, it handles location-based services, it is interfaced between AMF and SMF.

SCP is a new HTTP/2-based network function enabling dynamic scaling and management of communication and services in the 5G network. It is used to act as an intermediary between 5G NF and external entities. It is providing a standardized interface for service functions.

N3IWF is non-3GPP access stratum such as Wi-Fi, Li-Fi or satellite connection.

Android app – O5GC has specific applications to make registration to the network using for example Wi-Fi.

TS 23.502 Procedure standard gives specific overview about the "Call flow", i.e. how UE or IoT devices should connect to network and what procedures must be completed to make it happen. For example, UE starting up and emits certain information like SUPI or SUCI. Before SUPI there are more procedures done, but not covered in this thesis. The next section describes shortly this call flow, focused mainly on "Initial Registration".

3.1.1 UE registration "Call Flow"

Call flow means the procedure for how a UE is going to register to the network and what features are used. 5G registration management (RM) is used for that. General registration figure (Figure 7) shows the connection management, how it is used to establish a connection between UE and AMF. The connection management is used to register and deregister a UE from the network. The mobility management functions are used to keep track of the current UE location. When UE is performing Initial Registration, its current status is RM-DEREGISTERED. UE should indicate GUAMI information as its 5 parameters. However, when the UE is performing Initial Registration with SUCI or SUPI, then GUAMI parameters is not indicated. After that, the registered UE indicates RM-REGISTERED. RM-REGISTERED is indicated by e.g. MM state and CM state. It means that mobility management and connection management are registered and connected. If the UE performs connection management update, the UE includes the list of PDU session to be active. Normally, the UE should be always-on PDU session which are accepted by the network. It is indicated by CM CONNECTED if PDU is established.

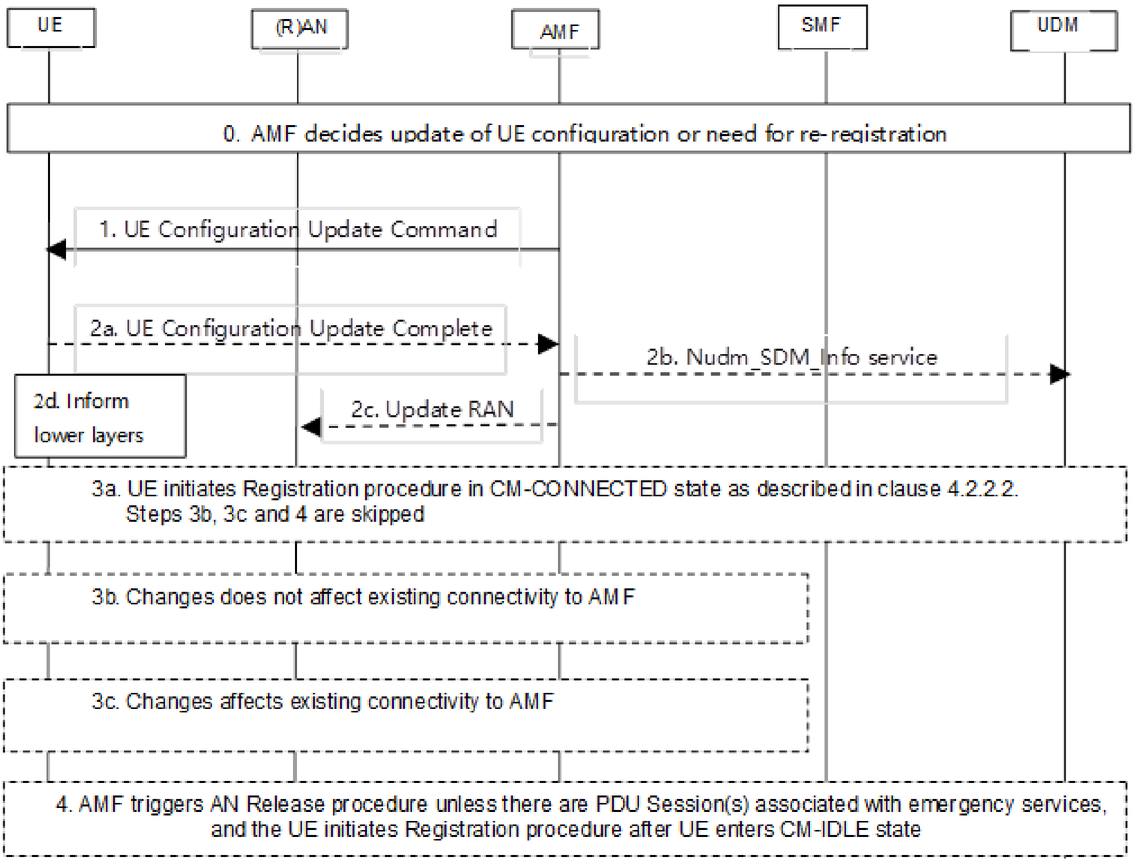

UE
(R)AN
AMF
SMF
UDM
0. AMF decides update of UE configuration or need for re-registration
1. UE Configuration Update Command
2a. UE Configuration Update Complete
2b. Nudm_SDM_Info service
2d. Inform lower layers
2c. Update RAN
3a. UE initiates Registration procedure in CM-CONNECTED state as described in clause 4.2.2.2. Steps 3b, 3c and 4 are skipped
3b. Changes does not affect existing connectivity to AMF
3c. Changes affects existing connectivity to AMF
4. AMF triggers AN Release procedure unless there are PDU Session(s) associated with emergency services, and the UE initiates Registration procedure after UE enters CM-IDLE state

Figure 7. UE configuration updating procedure architecture diagram [6].

The text what follows is based on the standard documentation.

UE PDU can be CM – IDLE or CM - CONNECTED. This procedure is initiated by AMF when the UE wants to update the configuration. If UE is in CM - CONNECTED state, then it is possible to have PDU session establishment and the UE can access the internet. If in CM – IDLE state, then the UE cannot access the internet, but the UE is still registered and can access to perform PDU update. The “List OF PDU Sessions To Be Activated” is provided by the UE. SMF indicates that update and active N3 tunnel is provided by SMF. Then N3 bearers starts to be used for data transfer between core network UPF and UE. N6 provides internet access for UE to DN. Going back to Initial registration, if initial registration is started then the UE needs to register with the network to get authorized to receive services, to enable mobility tracking, and to enable reachability.

The UE initiates the Registration procedure then UE is using one of the following Registration types:[6]

“

- Initial Registration to the 5GS;
- Mobility Registration Update upon changing to a new Tracking Area (TA) outside the UE's Registration Area in both CM-CONNECTED and CM-IDLE state;
- Periodic Registration Update (due to a predefined time period of inactivity); or
- Emergency Registration; or it can be also
- Disaster Roaming Initial Registration, as specified in clause 5.40 of TS 23.501 [2]; or
- Disaster Roaming Mobility Registration Update.

The following are the clear text information elements (IEs), as defined in TS 24.501 [25] that can be sent by the UE in the Registration Request message if the UE has no NAS security context:

- Registration type;
- SUCI or 5G-GUTI or PEI;
- Security parameters;
- Additional GUTI;
- 4G Tracking Area Update;
- the indication that the UE is moving from EPS;
- PLMN with Disaster Condition;
- if the UE is registering with an SNPN, the NID of the SNPN that assigned the 5G-GUTI.

”

This thesis focuses on the Initial Registration to the 5GS.

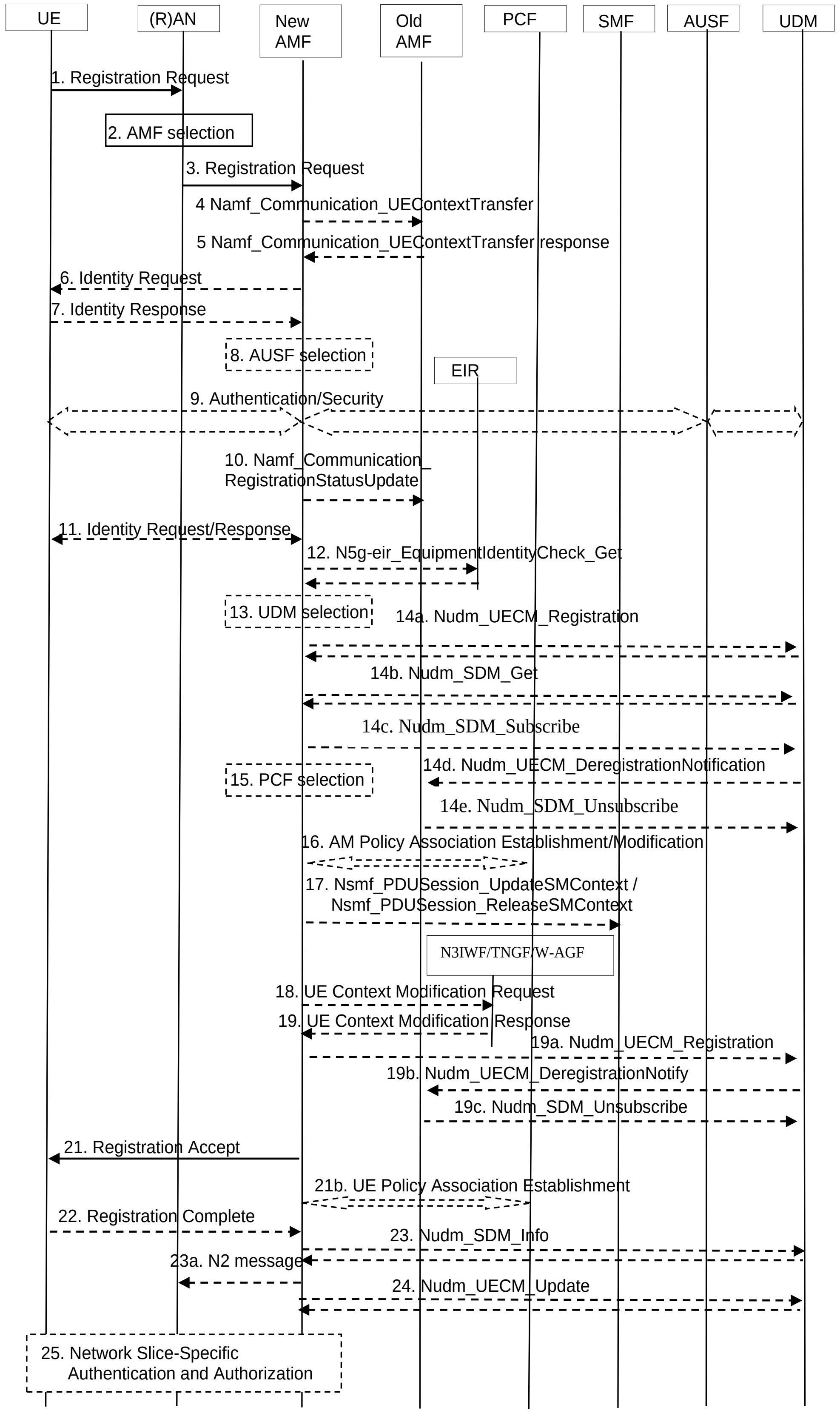

UE
(R)AN
New AMF
Old AMF
PCF
SMF
AUSF
UDM
1. Registration Request
2. AMF selection
3. Registration Request
4 Namf_Communication_UEContextTransfer
5 Namf_Communication_UEContextTransfer response
6. Identity Request
7. Identity Response
8. AUSF selection
EIR
9. Authentication/Security
10. Namf_Communication_ RegistrationStatusUpdate
11. Identity Request/Response
12. N5g-eir_EquipmentIdentityCheck_Get
13. UDM selection
14a. Nudm_UECM_Registration
14b. Nudm_SDM_Get
14c. Nudm_SDM_Subscribe
14d. Nudm_UECM_DeregistrationNotification
15. PCF selection
14e. Nudm_SDM_Unsubscribe
16. AM Policy Association Establishment/Modification
17. Nsmf_PDUSession_UpdateSMContext / Nsmf_PDUSession_ReleaseSMContext
N3IWF/TNGF/W-AGF
18. UE Context Modification Request
19. UE Context Modification Response
19a. Nudm_UECM_Registration
19b. Nudm_UECM_DeregistrationNotify
19c. Nudm_SDM_Unsubscribe
21. Registration Accept
21b. UE Policy Association Establishment
22. Registration Complete
23. Nudm_SDM_Info
23a. N2 message
24. Nudm_UECM_Update
25. Network Slice-Specific Authentication and Authorization

Figure 8. General registration call flow, from standard TS 23.502 [23].

3.1.2 Call flow procedure description

Under general registration (see Figure 8) it is shown that the UE is going to registered to the network.
The UE sends a registration request to the R(AN).

Initial registration request includes: [6].
“

- Registration type
- UE ID like SUPI or SUCI
- NSSAI info

”

R(AN) selects AMF, old or new AMF. R(AN) sends registration request to the AMF, typically new AMF, it depends on the registration request type. Under initial registration, it is always selected new AMF and old AMF is not involved and thus Namf is not used.
AMF collects UE identity.
AMF selects AUSF to authorise and secure the connection using Nausf service.
UDM is selected for the registration. Nudm service is used to get whether UE data is on the list. If it is, then Nudm subscription is given and depends on reguest, the old AMF will be UE deregistered and unsubscribed.
PCF is selected to define policies and gives accepts for AMF.
AMF accepts registration if all policies are good, and UE is registered to core.

If registration accepted from AMF, then AMF sends back: [6].
“

- 5G – GUTI like MNC and MCC info
- Registration area
- Allowed NSSAI
- Also writing registration accepted

”

The UE confirms to AMF that registration is complete.

UE stays registered until deregistration is performed, for example when the UE is shutting down.

3.1.3 Overview of SUPI/SUCI/5G-GUTI

SUPI includes mobile country code, mobile network code and ID, see Figure 9.

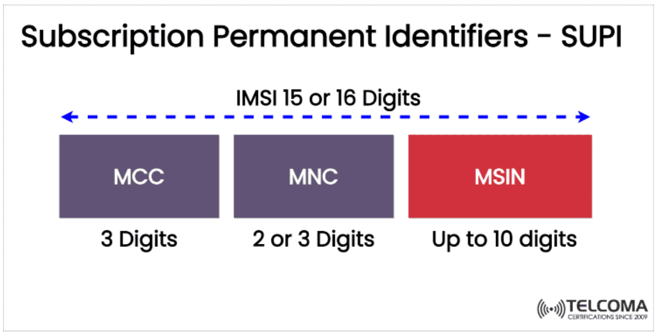


Figure 9. SUPI code overview [27].

The UE can generate SUCI code. SUCI includes SUCI type, home network ID. Routing indictor, protection scheme, home network public key ID and protection scheme output. See Figure 10.

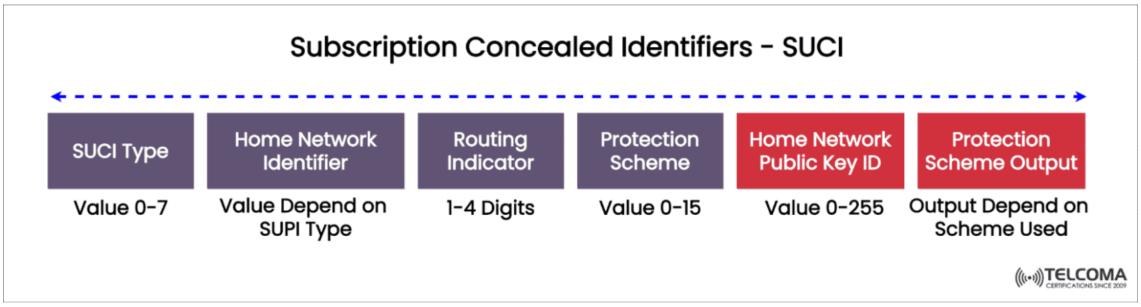

Subscription Concealed Identifiers - SUCI
SUCI Type
Home Network Identifier
Routing Indicator
Protection Scheme
Home Network Public Key ID
Protection Scheme Output
Value 0-7
Value Depend on SUPI Type
1-4 Digits
Value 0-15
Value 0-255
Output Depend on Scheme Used
TELCOMA
CERTIFICATIONS SINCE 2009

Figure 10. SUCI information [27].

5G GUTI (Figure 11) includes different ID numbers, because the SUPI should not be transferred in clear text over 5G-RAN. 5G GUTI is used by AMF and is needed to provide when the UE is moving one location to another.

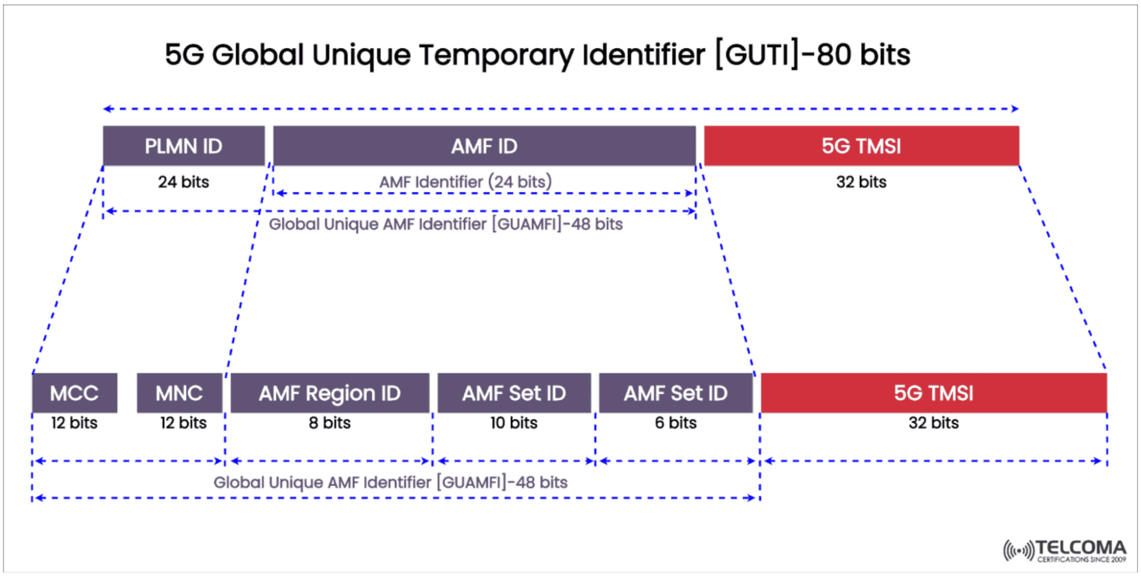

5G Global Unique Temporary Identifier [GUTI]-80 bits
PLMN ID
24 bits
AMF ID
AMF Identifier (24 bits)
5G TMSI
32 bits
Global Unique AMF Identifier [GUAMFI]-48 bits
MCC
12 bits
MNC
12 bits
AMF Region ID
8 bits
AMF Set ID
10 bits
AMF Set ID
6 bits
5G TMSI
32 bits
Global Unique AMF Identifier [GUAMFI]-48 bits
TELCOMA
CERTIFICATIONS SINCE 2009

Figure 11. 5G GUTI information [27].

Detailed explanation of the above so-called 5G AKA mechanism is not in the scope of this thesis. The interested reader can refer to section 5 in standard TS33.501 [28].

3.2 O5GC overview [5]

Open5GCore, also known as O5GC [5], is developed and provided by Fraunhofer Company for testing 5G network. O5GC testbed toolkit developments are supported by Fraunhofer FOKUS and the Technische Universität Berlin (TUB) AV research group. O5GC prototypes 3GPP Release 15 and 16 core network functionalities and is compatible with 5G NR base stations and user equipment. O5GC is ready for release 17 also. O5GC provides a basis for building 5G testbeds and is also able to support the development from 5G to 6G. It is 3GPP standard-oriented and implements the new 5G components. It is SA 5G, independent from the previous 4G functionality. Through this, O5GC supports fast user hands-on and implementation, realistic evaluation and demonstration of new concepts and use cases. Open5GCore development is well supported and will be updated on the long term.

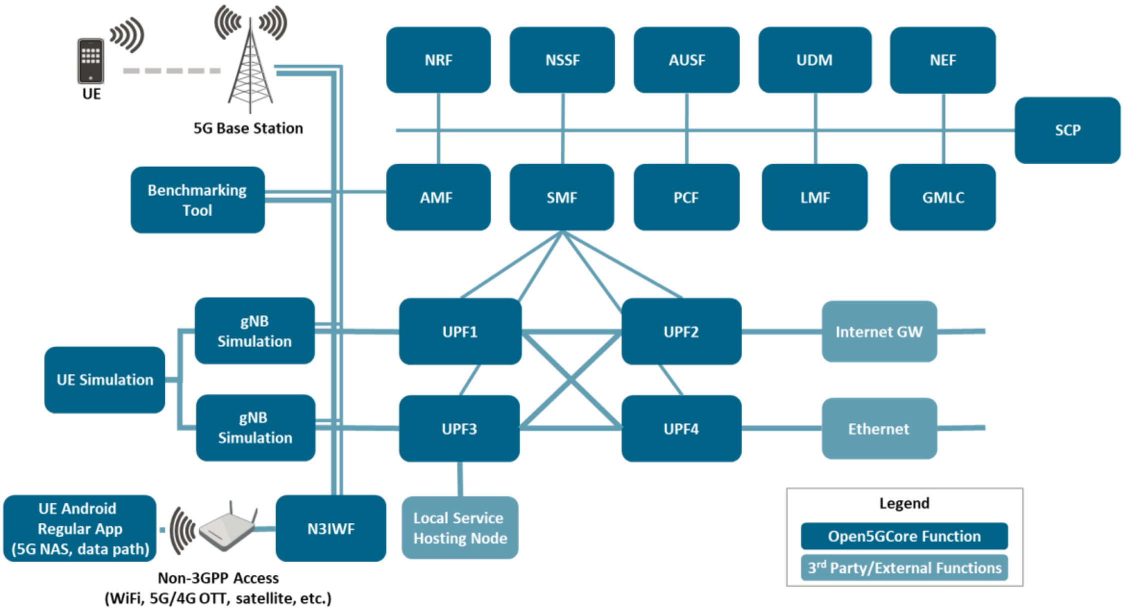


Figure 12. Open5GCore basic architecture diagram [5].

For ease of read, Figure 12 shows again O5GCore basic architecture diagram, and it is possible to compare it with 5G standard architecture diagram, see Additional Figure 1 and Additional Figure 2.

O5GC simulator includes 6 different setups:

1. 5G basic – for initial registration and PDU session testing
2. 5G AF – for testing data connection management
3. 5G LMF – for location testing
4. 5G N3IWF – for 3rd party like Wi-Fi and satellite connection testing.
5. 5G NSSF – for slicing, useful Voice over NR testing
6. 5G SCP – for roaming tests

The author had used "1. 5G basic" configuration and this architecture diagram is a little bit different from the other configurations. For example, PCF function is not used, and it is not needed for learning purposes.

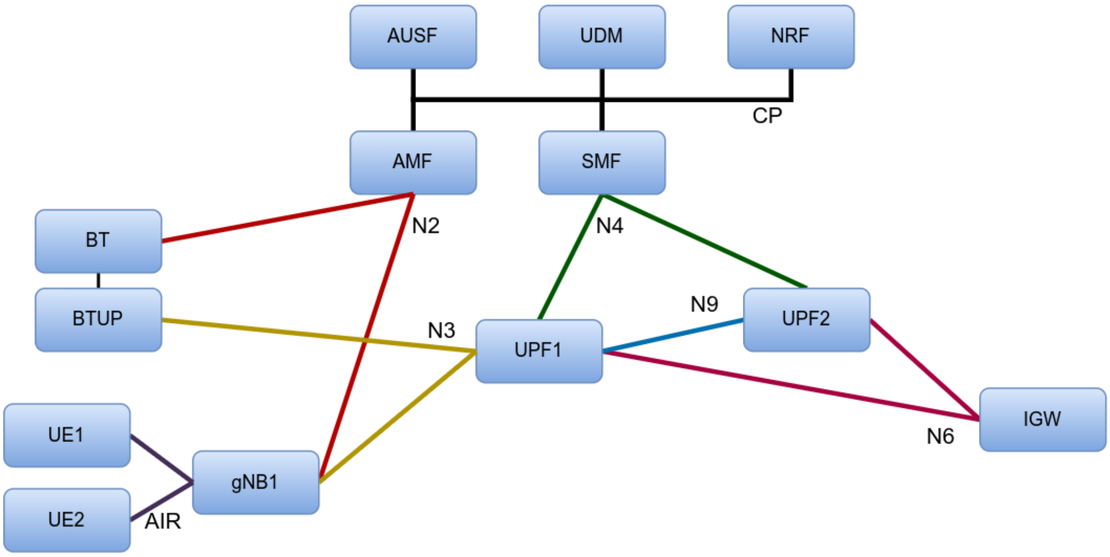


Figure 13. 5G basic configuration diagram. [from open5GC wiki, provided from simulator manual]

Under registration “Call Flow” there is described what functions are in use during the laboratory work (Figure 13, Figure 14).

3.2.1 O5GC Call flow for UE registration to the network

Call flow means procedure how UE is going to register to the network and what functions are used for that. This section describes registration Call Flow from simulator how UE is going to be registered to the network. The Initial registration procedure is used to register the UE to the network.

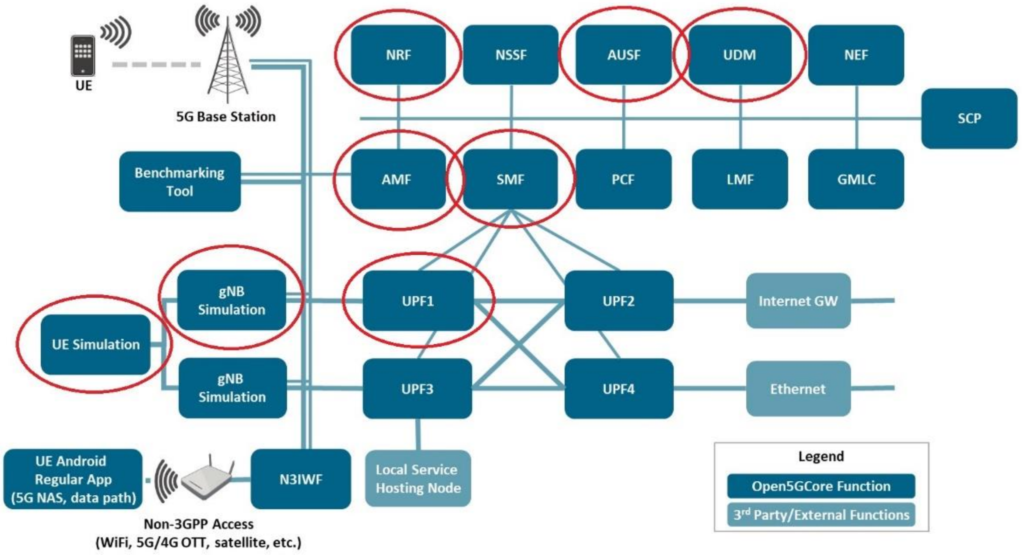


Figure 14. Open5GCore basic architecture diagram. The added red marked circles indicate the elements and functions needed for UE registration components using the simulator.

3.2.2 Registration Call flow procedure description

Using O5GC simulator, the call flow looks as follows. It is possible to zoom in on the following figures (Figure 15, Figure 16, Figure 17) to see more details or separated view under Appendix 2, 3, 4.

Under initial registration it is shown that UE is going to registered to the network; only 5 functions are used. Here, it is analysed three main functions: UE1, GNB(RAN) and AMF. Additionally, AUSF and UDM are also analysed.

Registration procedure

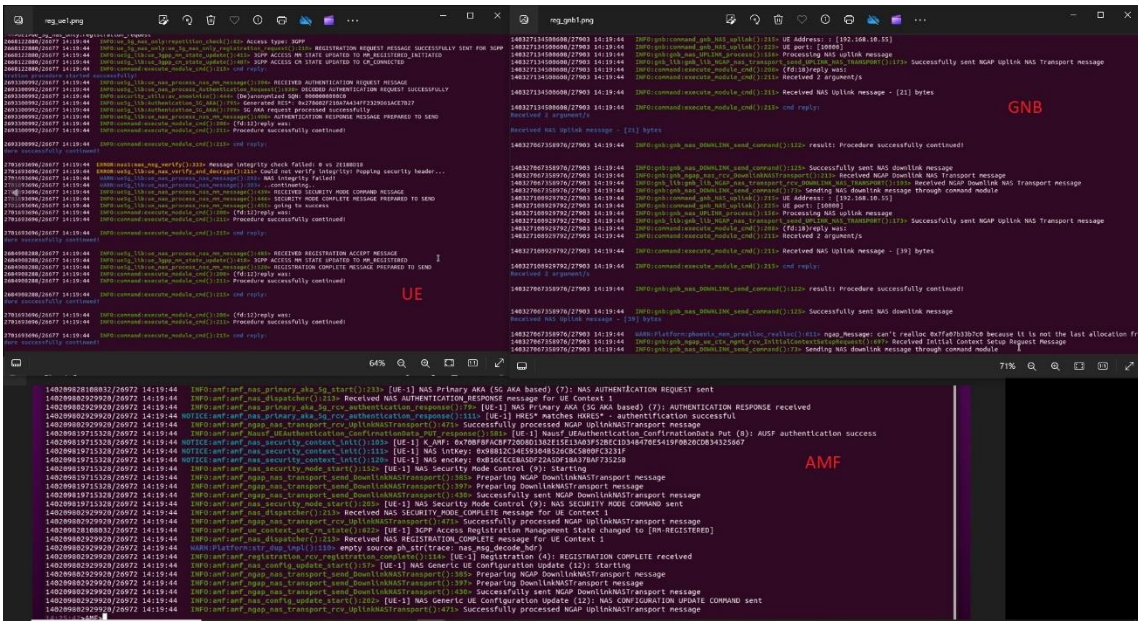
UE sends registration request to the R(AN) – sending initial registration request.
Initial registration request includes:

- Registration type
- SUPI information

R(AN) selects new AMF, old AMF is not present on initial request.
R(AN) sends registration request to the AMF.
AMF collects UE identity and accepts registration request.
If accepted, then AMF sends back to UE:

- 5G – GUTI info this includes GUAMI info (MNC and MCC, AMF region, etc.)

AMF selects AUSF to authorise and secure connection.
UDM is selected for the registration to cache UE on the list.
AMF accepts registration if all policies are good, and UE is on the list.
AMF sends information to UE that registration is complete.
UE confirms to AMF that registration is complete.
UE is registered until UE deregistration is done, for example UE is shutting down.



```
2668122880/26677 14:19:44   INFO:ue_5g_nas_only:repetition_check():62> Access type: 3GPP
2668122880/26677 14:19:44   INFO:ue_5g_nas_only:ue_5g_nas_only_registration_request():216> REGISTRATION REQUEST MESSAGE SUCCESSFULLY SENT FOR 3GPP
2668122880/26677 14:19:44   INFO:ue5g_lib:ue_3gpp_mm_state_update():415> 3GPP ACCESS MM STATE UPDATED TO MM_REGISTERED_INITIATED
2668122880/26677 14:19:44   INFO:ue5g_lib:ue_3gpp_cm_state_update():407> 3GPP ACCESS CM STATE UPDATED TO CM_CONNECTED
2668122880/26677 14:19:44   INFO:command:execute_module_cmd():215> cmd reply:
tration procedure started successfully!
2693300992/26677 14:19:44   INFO:ue5g_lib:ue_nas_process_nas_mm_message():394> RECEIVED AUTHENTICATION REQUEST MESSAGE
2693300992/26677 14:19:44   INFO:ue5g_lib:ue_nas_process_Authentication_Request():838> DECODED AUTHENTICATION REQUEST SUCCESSFULLY
2693300992/26677 14:19:44   INFO:security_utils:av_anonimize():444> (De)anonymized SQN: 0000000000C0
2693300992/26677 14:19:44   INFO:ue5g_lib:Authentication_5G_AKA():795> Generated RES*: 0x278602F210A7A434FF2329D61ACE7B27
2693300992/26677 14:19:44   INFO:ue5g_lib:Authentication_5G_AKA():799> 5G AKA request processed successfully
2693300992/26677 14:19:44   INFO:ue5g_lib:ue_nas_process_nas_mm_message():406> AUTHENTICATION RESPONSE MESSAGE PREPARED TO SEND
2693300992/26677 14:19:44   INFO:command:execute_module_cmd():208> (fd:12)reply was:
2693300992/26677 14:19:44   INFO:command:execute_module_cmd():211> Procedure successfully continued!

2693300992/26677 14:19:44   INFO:command:execute_module_cmd():215> cmd reply:
dure successfully continued!

2701693696/26677 14:19:44  ERROR:nas5:nas_msg_verify():333> Message integrity check failed: 0 vs 2E188D18
2701693696/26677 14:19:44  ERROR:ue5g_lib:ue_nas_verify_and_decrypt():211> Could not verify integrity! Popping security header...
2701693696/26677 14:19:44   WARN:ue5g_lib:ue_nas_process_nas_message():292> NAS integrity failed!
2701693696/26677 14:19:44   WARN:ue5g_lib:ue_nas_process_nas_message():303> ..continueing..
2701693696/26677 14:19:44   INFO:ue5g_lib:ue_nas_process_nas_mm_message():439> RECEIVED SECURITY MODE COMMAND MESSAGE
2701693696/26677 14:19:44   INFO:ue5g_lib:ue_nas_process_nas_mm_message():446> SECURITY MODE COMPLETE MESSAGE PREPARED TO SEND
2701693696/26677 14:19:44   INFO:ue5g_lib:ue_nas_process_nas_mm_message():451> going to success
2701693696/26677 14:19:44   INFO:command:execute_module_cmd():208> (fd:12)reply was:
2701693696/26677 14:19:44   INFO:command:execute_module_cmd():211> Procedure successfully continued!

2701693696/26677 14:19:44   INFO:command:execute_module_cmd():215> cmd reply:
dure successfully continued!

2684908288/26677 14:19:44   INFO:ue5g_lib:ue_nas_process_nas_mm_message():485> RECEIVED REGISTRATION ACCEPT MESSAGE
2684908288/26677 14:19:44   INFO:ue5g_lib:ue_3gpp_mm_state_update():418> 3GPP ACCESS MM STATE UPDATED TO MM_REGISTERED
2684908288/26677 14:19:44   INFO:ue5g_lib:ue_nas_process_nas_mm_message():520> REGISTRATION COMPLETE MESSAGE PREPARED TO SEND
2684908288/26677 14:19:44   INFO:command:execute_module_cmd():208> (fd:12)reply was:
2684908288/26677 14:19:44   INFO:command:execute_module_cmd():211> Procedure successfully continued!

2684908288/26677 14:19:44   INFO:command:execute_module_cmd():215> cmd reply:
dure successfully continued!

2701693696/26677 14:19:44   INFO:command:execute_module_cmd():208> (fd:12)reply was:
2701693696/26677 14:19:44   INFO:command:execute_module_cmd():211> Procedure successfully continued!

2701693696/26677 14:19:44   INFO:command:execute_module_cmd():215> cmd reply:
dure successfully continued!
```

```
140327134508608/27903 14:19:44   INFO:gnb:command_gnb_NAS_uplink():215> UE Address: : [192.168.10.55]
140327134508608/27903 14:19:44   INFO:gnb:command_gnb_NAS_uplink():225> UE port: [10000]
140327134508608/27903 14:19:44   INFO:gnb:gnb_nas_UPLINK_process():156> Processing NAS uplink message
140327134508608/27903 14:19:44   INFO:gnb_lib:gnb_lib_NGAP_nas_transport_send_UPLINK_NAS_TRANSPORT():173> Successfully sent NGAP Uplink NAS Transport message
140327134508608/27903 14:19:44   INFO:command:execute_module_cmd():208> (fd:18)reply was:
140327134508608/27903 14:19:44   INFO:command:execute_module_cmd():211> Received 2 argument/s

140327134508608/27903 14:19:44   INFO:command:execute_module_cmd():211> Received NAS Uplink message - [21] bytes

140327134508608/27903 14:19:44   INFO:command:execute_module_cmd():215> cmd reply:
Received 2 argument/s

Received NAS Uplink message - [21] bytes

140327067358976/27903 14:19:44   INFO:gnb:gnb_nas_DOWNLINK_send_command():122> result: Procedure successfully continued!


140327067358976/27903 14:19:44   INFO:gnb:gnb_nas_DOWNLINK_send_command():125> Successfully sent NAS downlink message
140327067358976/27903 14:19:44   INFO:gnb:gnb_ngap_nas_rcv_DownlinkNASTransport():213> Received NGAP Downlink NAS Transport message
140327067358976/27903 14:19:44   INFO:gnb_lib:gnb_lib_NGAP_nas_transport_rcv_DOWNLINK_NAS_TRANSPORT():193> Received NGAP Downlink NAS Transport message
140327067358976/27903 14:19:44   INFO:gnb:gnb_nas_DOWNLINK_send_command():73> Sending NAS downlink message through command module
140327108929792/27903 14:19:44   INFO:gnb:command_gnb_NAS_uplink():215> UE Address: : [192.168.10.55]
140327108929792/27903 14:19:44   INFO:gnb:command_gnb_NAS_uplink():225> UE port: [10000]
140327108929792/27903 14:19:44   INFO:gnb:gnb_nas_UPLINK_process():156> Processing NAS uplink message
140327108929792/27903 14:19:44   INFO:gnb_lib:gnb_lib_NGAP_nas_transport_send_UPLINK_NAS_TRANSPORT():173> Successfully sent NGAP Uplink NAS Transport message
140327108929792/27903 14:19:44   INFO:command:execute_module_cmd():208> (fd:18)reply was:
140327108929792/27903 14:19:44   INFO:command:execute_module_cmd():211> Received 2 argument/s

140327108929792/27903 14:19:44   INFO:command:execute_module_cmd():211> Received NAS Uplink message - [39] bytes

140327108929792/27903 14:19:44   INFO:command:execute_module_cmd():215> cmd reply:
Received 2 argument/s

140327067358976/27903 14:19:44   INFO:gnb:gnb_nas_DOWNLINK_send_command():122> result: Procedure successfully continued!


140327067358976/27903 14:19:44   INFO:gnb:gnb_nas_DOWNLINK_send_command():125> Successfully sent NAS downlink message
Received NAS Uplink message - [39] bytes

140327067358976/27903 14:19:44   WARN:Platform:phoenix_mem_prealloc_realloc():611> ngap_Message: can't realloc 0x7fa07b33b7c0 because it is not the last allocation fr
140327067358976/27903 14:19:44   INFO:gnb:gnb_ngap_ue_ctx_mgmt_rcv_InitialContextSetupRequest():697> Received Initial Context Setup Request Message
140327067358976/27903 14:19:44   INFO:gnb:gnb_nas_DOWNLINK_send_command():73> Sending NAS downlink message through command module
```

```
140209828108032/26972 14:19:44   INFO:amf:amf_nas_primary_aka_5g_start():233> [UE-1] NAS Primary AKA (5G AKA based) (7): NAS AUTHENTICATION REQUEST sent
140209802929920/26972 14:19:44   INFO:amf:amf_nas_dispatcher():213> Received NAS AUTHENTICATION_RESPONSE message for UE Context 1
140209802929920/26972 14:19:44   INFO:amf:amf_nas_primary_aka_5g_rcv_authentication_response():79> [UE-1] NAS Primary AKA (5G AKA based) (7): AUTHENTICATION RESPONSE received
140209802929920/26972 14:19:44 NOTICE:amf:amf_nas_primary_aka_5g_rcv_authentication_response():111> [UE-1] HRES* matches HXRES* - authentification successful
140209802929920/26972 14:19:44   INFO:amf:amf_ngap_nas_transport_rcv_UplinkNASTransport():471> Successfully processed NGAP UplinkNASTransport message
140209819715328/26972 14:19:44   INFO:amf:amf_Nausf_UEAuthentication_ConfirmationData_PUT_response():581> [UE-1] Nausf_UEAuthentication_ConfirmationData Put (8): AUSF authentication success
140209819715328/26972 14:19:44 NOTICE:amf:amf_nas_security_context_init():103> [UE-1] K_AMF: 0x70BF8FACBF720D8D1382E15E13A03F52BEC1D348470E5419F0B20CDB34325667
140209819715328/26972 14:19:44 NOTICE:amf:amf_nas_security_context_init():111> [UE-1] NAS intKey: 0x98812C34E59304B526CBC5800FC3231F
140209819715328/26972 14:19:44 NOTICE:amf:amf_nas_security_context_init():120> [UE-1] NAS encKey: 0xB16CECE8A5DF22A5DF18A37BAF73525B
140209819715328/26972 14:19:44   INFO:amf:amf_nas_security_mode_start():152> [UE-1] NAS Security Mode Control (9): Starting
140209819715328/26972 14:19:44   INFO:amf:amf_ngap_nas_transport_send_DownlinkNASTransport():385> Preparing NGAP DownlinkNASTransport message
140209819715328/26972 14:19:44   INFO:amf:amf_ngap_nas_transport_send_DownlinkNASTransport():397> Preparing DownlinkNASTransport message
140209819715328/26972 14:19:44   INFO:amf:amf_ngap_nas_transport_send_DownlinkNASTransport():430> Successfully sent NGAP DownlinkNASTransport message
140209819715328/26972 14:19:44   INFO:amf:amf_nas_security_mode_start():205> [UE-1] NAS Security Mode Control (9): NAS SECURITY MODE COMMAND sent
140209802929920/26972 14:19:44   INFO:amf:amf_nas_dispatcher():213> Received NAS SECURITY_MODE_COMPLETE message for UE Context 1
140209802929920/26972 14:19:44   INFO:amf:amf_ngap_nas_transport_rcv_UplinkNASTransport():471> Successfully processed NGAP UplinkNASTransport message
140209828108032/26972 14:19:44   INFO:amf:amf_ue_context_set_rm_state():622> [UE-1] 3GPP Access Registration Management State changed to [RM-REGISTERED]
140209802929920/26972 14:19:44   INFO:amf:amf_nas_dispatcher():213> Received NAS REGISTRATION_COMPLETE message for UE Context 1
140209802929920/26972 14:19:44   WARN:Platform:str_dup_impl():110> empty source ph_str(trace: nas_msg_decode_hdr)
140209802929920/26972 14:19:44   INFO:amf:amf_registration_rcv_registration_complete():114> [UE-1] Registration (4): REGISTRATION COMPLETE received
140209802929920/26972 14:19:44   INFO:amf:amf_nas_config_update_start():57> [UE-1] NAS Generic UE Configuration Update (12): Starting
140209802929920/26972 14:19:44   INFO:amf:amf_ngap_nas_transport_send_DownlinkNASTransport():385> Preparing NGAP DownlinkNASTransport message
140209802929920/26972 14:19:44   INFO:amf:amf_ngap_nas_transport_send_DownlinkNASTransport():397> Preparing DownlinkNASTransport message
140209802929920/26972 14:19:44   INFO:amf:amf_ngap_nas_transport_send_DownlinkNASTransport():430> Successfully sent NGAP DownlinkNASTransport message
140209802929920/26972 14:19:44   INFO:amf:amf_nas_config_update_start():202> [UE-1] NAS Generic UE Configuration Update (12): NAS CONFIGURATION UPDATE COMMAND sent
140209802929920/26972 14:19:44   INFO:amf:amf_ngap_nas_transport_rcv_UplinkNASTransport():471> Successfully processed NGAP UplinkNASTransport message
14:25:42>AMF>
```

Figure 15. O5GC simulator functions: UE1, GNB and AMF. Separated pictures are provide under Appendixes 2, 3, and 4.

From UE side, see Figure 15:

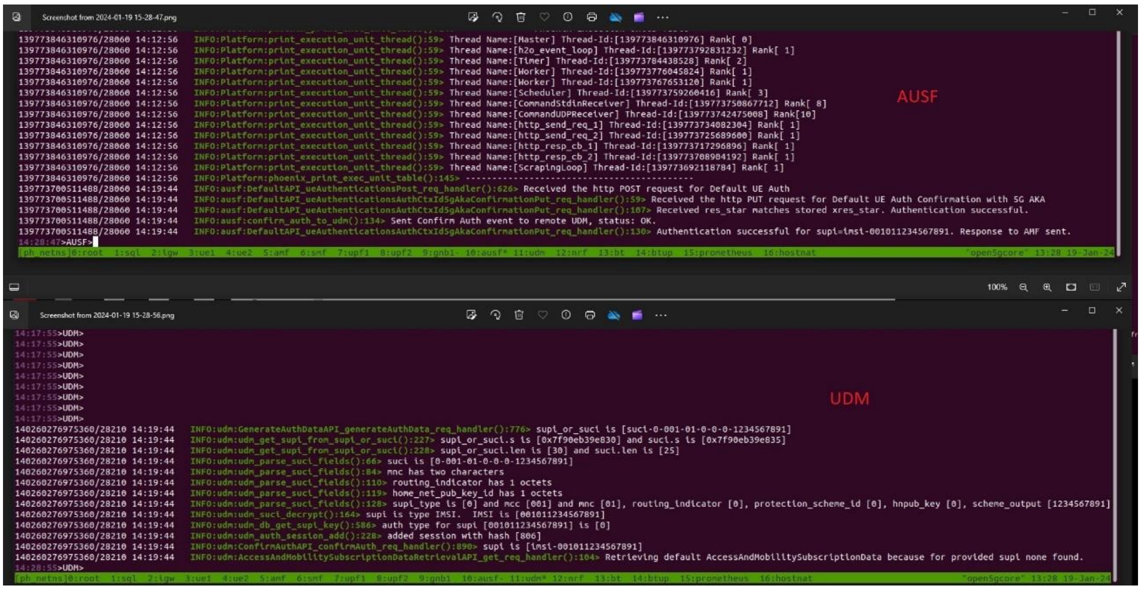
- UE1 starts Initial registration to gNB, delivers NAS information and is security protected. Messages is registration request only.
- MM registered and CM connected initiated.
- Starts 5G AKA security procedure. This is securely forwarding SUPI code using SUCI code. UE1 has SUPI code, K code and SQN code. Combining together SUPI code is delivered securely to AMF and from there to UDM. This is called Security mode command message. More information from standard 5G AKA.[28]
- Authentication request received and decoded. Integrity failed because it is initial registration only without PDU session. Integrity failed on simulator and the standard asks to show that NAS integrity check failed. Integrity will not fail if PDU is also included. It does not affect registration and the simulator continues. It is not a secured connection. Integrity is optional between UE and gNB. It should be noted to TalTech researchers to check this with Fraunhoffer company. It should not fail with registration only also because under 5G registration request and PDU session have been separated, whereas in 4G they were registered together.
- UE receives registration acceptance message and change state to MM REGISTERED.
- EU sends back message to AMF registration complete.

gNB, in parallel to the above, see Figure 15:

- gNB forwards registration request to AMF.
- gNB gives UE IP address and port to AMF through command module and forwards NAS information back and forward until process are successful.

From AMF side, see Figure 15:

- AMF starts authentication and gets received info that authentication complete.
- AMF gets same information like UDM: HRES matches HXRES database info, generated by 5G AKA rules [28].
- Through Nausf service, it is given that authentication successfully complete.
- After integrity check, security mode check is complete.
- AMF gets RM-REGISTERED and it means UE1 MM is REGISTERED status.
- AMF sends registration complete message to UE1.
- UE1 sends back registration complete message.
- After that, UE1 gets NAS configuration updated message.


Screenshot from 2024-01-19 15-28-47.png
AUSF
100%
Screenshot from 2024-01-19 15-28-56.png
UDM

```
139773846310976/28060 14:12:56   INFO:Platform:print_execution_unit_thread():59> Thread Name:[Master] Thread-Id:[139773846310976] Rank[ 0]
139773846310976/28060 14:12:56   INFO:Platform:print_execution_unit_thread():59> Thread Name:[h2o_event_loop] Thread-Id:[139773792831232] Rank[ 1]
139773846310976/28060 14:12:56   INFO:Platform:print_execution_unit_thread():59> Thread Name:[Timer] Thread-Id:[139773784438528] Rank[ 2]
139773846310976/28060 14:12:56   INFO:Platform:print_execution_unit_thread():59> Thread Name:[Worker] Thread-Id:[139773776045824] Rank[ 1]
139773846310976/28060 14:12:56   INFO:Platform:print_execution_unit_thread():59> Thread Name:[Worker] Thread-Id:[139773767653120] Rank[ 1]
139773846310976/28060 14:12:56   INFO:Platform:print_execution_unit_thread():59> Thread Name:[Scheduler] Thread-Id:[139773759260416] Rank[ 3]
139773846310976/28060 14:12:56   INFO:Platform:print_execution_unit_thread():59> Thread Name:[CommandStdinReceiver] Thread-Id:[139773750867712] Rank[ 8]
139773846310976/28060 14:12:56   INFO:Platform:print_execution_unit_thread():59> Thread Name:[CommandUDPReceiver] Thread-Id:[139773742475008] Rank[10]
139773846310976/28060 14:12:56   INFO:Platform:print_execution_unit_thread():59> Thread Name:[http_send_req_1] Thread-Id:[139773734082304] Rank[ 1]
139773846310976/28060 14:12:56   INFO:Platform:print_execution_unit_thread():59> Thread Name:[http_send_req_2] Thread-Id:[139773725689600] Rank[ 1]
139773846310976/28060 14:12:56   INFO:Platform:print_execution_unit_thread():59> Thread Name:[http_resp_cb_1] Thread-Id:[139773717296896] Rank[ 1]
139773846310976/28060 14:12:56   INFO:Platform:print_execution_unit_thread():59> Thread Name:[http_resp_cb_2] Thread-Id:[139773708904192] Rank[ 1]
139773846310976/28060 14:12:56   INFO:Platform:print_execution_unit_thread():59> Thread Name:[ScrapingLoop] Thread-Id:[139773692118784] Rank[ 1]
139773846310976/28060 14:12:56   INFO:Platform:phoenix_print_exec_unit_table():145> ----------------------------------------
139773700511488/28060 14:19:44   INFO:ausf:DefaultAPI_ueAuthenticationsPost_req_handler():626> Received the http POST request for Default UE Auth
139773700511488/28060 14:19:44   INFO:ausf:DefaultAPI_ueAuthenticationsAuthCtxId5gAkaConfirmationPut_req_handler():59> Received the http PUT request for Default UE Auth Confirmation with 5G AKA
139773700511488/28060 14:19:44   INFO:ausf:DefaultAPI_ueAuthenticationsAuthCtxId5gAkaConfirmationPut_req_handler():107> Received res_star matches stored xres_star. Authentication successful.
139773700511488/28060 14:19:44   INFO:ausf:confirm_auth_to_udm():134> Sent Confirm Auth event to remote UDM, status: OK.
139773700511488/28060 14:19:44   INFO:ausf:DefaultAPI_ueAuthenticationsAuthCtxId5gAkaConfirmationPut_req_handler():130> Authentication successful for supi=imsi-001011234567891. Response to AMF sent.
14:28:47>AUSF>
[ph_netns]0:root  1:sql  2:igw  3:ue1  4:ue2  5:amf  6:smf  7:upf1  8:upf2  9:gnb1- 10:ausf* 11:udm  12:nrf  13:bt  14:btup  15:prometheus  16:hostnat            "open5gcore" 13:28 19-Jan-24
```

```
14:17:55>UDM>
14:17:55>UDM>
14:17:55>UDM>
14:17:55>UDM>
14:17:55>UDM>
14:17:55>UDM>
14:17:55>UDM>
14:17:55>UDM>
14:17:55>UDM>
140260276975360/28210 14:19:44   INFO:udm:GenerateAuthDataAPI_generateAuthData_req_handler():776> supi_or_suci is [suci-0-001-01-0-0-0-1234567891]
140260276975360/28210 14:19:44   INFO:udm:udm_get_supi_from_supi_or_suci():227> supi_or_suci.s is [0x7f90eb39e830] and suci.s is [0x7f90eb39e835]
140260276975360/28210 14:19:44   INFO:udm:udm_get_supi_from_supi_or_suci():228> supi_or_suci.len is [30] and suci.len is [25]
140260276975360/28210 14:19:44   INFO:udm:udm_parse_suci_fields():66> suci is [0-001-01-0-0-0-1234567891]
140260276975360/28210 14:19:44   INFO:udm:udm_parse_suci_fields():84> mnc has two characters
140260276975360/28210 14:19:44   INFO:udm:udm_parse_suci_fields():110> routing_indicator has 1 octets
140260276975360/28210 14:19:44   INFO:udm:udm_parse_suci_fields():119> home_net_pub_key_id has 1 octets
140260276975360/28210 14:19:44   INFO:udm:udm_parse_suci_fields():128> supi_type is [0] and mcc [001] and mnc [01], routing_indicator [0], protection_scheme_id [0], hnpub_key [0], scheme_output [1234567891]
140260276975360/28210 14:19:44   INFO:udm:udm_suci_decrypt():164> supi is type IMSI.  IMSI is [001011234567891]
140260276975360/28210 14:19:44   INFO:udm:udm_db_get_supi_key():586> auth type for supi [001011234567891] is [0]
140260276975360/28210 14:19:44   INFO:udm:udm_auth_session_add():228> added session with hash [806]
140260276975360/28210 14:19:44   INFO:udm:ConfirmAuthAPI_confirmAuth_req_handler():890> supi is [imsi-001011234567891]
140260276975360/28210 14:19:44   INFO:udm:AccessAndMobilitySubscriptionDataRetrievalAPI_get_req_handler():104> Retrieving default AccessAndMobilitySubscriptionData because for provided supi none found.
14:28:55>UDM>
[ph_netns]0:root  1:sql  2:igw  3:ue1  4:ue2  5:amf  6:smf  7:upf1  8:upf2  9:gnb1  10:ausf- 11:udm* 12:nrf  13:bt  14:btup  15:prometheus  16:hostnat            "open5gcore" 13:28 19-Jan-24
```

Figure 16. O5GC simulator functions: AUSF and UDM

From AUSF side, see Figure 16:

- AMF asks from AUSF whether UE1 registered to start registration acceptance and is UE on the UDM list. Receives HTTP POST request for default UE Auth.
- AUSF gets SUPI code from AMF and asks from UDM, is UE on the list. AUSF gets confirmation, putting UE Auth confirmation on it.

- AUSF gets res_star matches stored xres_star and tell authentication successful. res_star is UE side and xres_star is UDM side.
- AUSF gets back SUPI code and presenting it with the code. And telling to UDM status is OK.

From UDM side: See Figure 16

- SUCI code is generated regarding to 5G AKA rules [28].
- UDM gets SUPI code and comparing in on his database.
- UDM confirms that SUPI belong to type IMSI code. Here is initial registration and SUPI code is only provided and presented. UDM provides acceptance response to AUSF and AUSF tells to AMF that the UE1 is allowed to register. AMF sends back to UE1 that registration is completed.
- UE1 response that registration complete.

3.2.3 Some practical notes

The computer software and hardware specification used for performing the above experiments and preparing the lab work presented in the next chapter is a follows:

General

- Ubuntu (64bit) version 20.04

Virtual machine information

- Processor: 1 CPU
- Memory: 1024 MB
- Graphic card: integrated 16 MB
- Storage: 20 GB
- Needed storage size: 10,3 GB
- Network adapter: Intel PRO/1000 MT Desktop

Available computer hardware build:

- Processor: 16 CPU cores
- Memory: 64 GB
- Video memory: 128 MB
- Hard disk: 512 GB

Monitoring with Wireshark

It is possible to monitor simulator commands with Wireshark. Wireshark can give different overview of the network traffic with given IP addresses, names, sources and many more what has changed or intervened, see Appendix 9.

Search NGAP and/or HTTP/2:

- NGAP is user profile information.
- HTTP/2 is Core side information.

O5GC manual

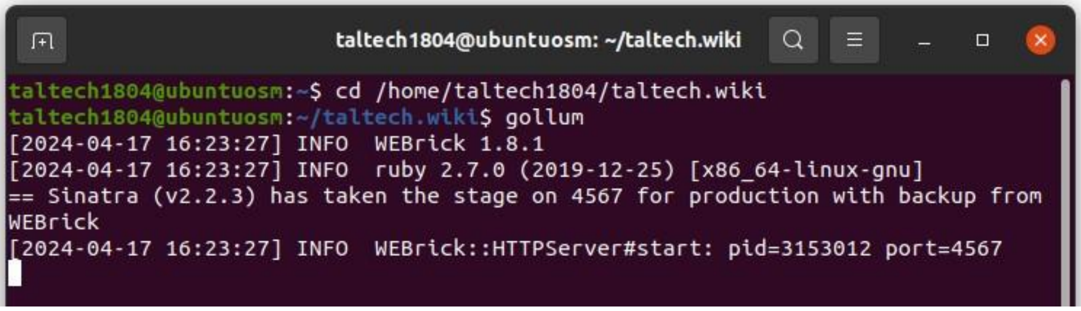
It is possible to use the simulator manual over browser menu named O5GC Wiki.

Procedures for O5GC manual activation:

1) Open terminal

2) Type: cd /home/taltech1804/taltech.wiki
3) Type: gollum

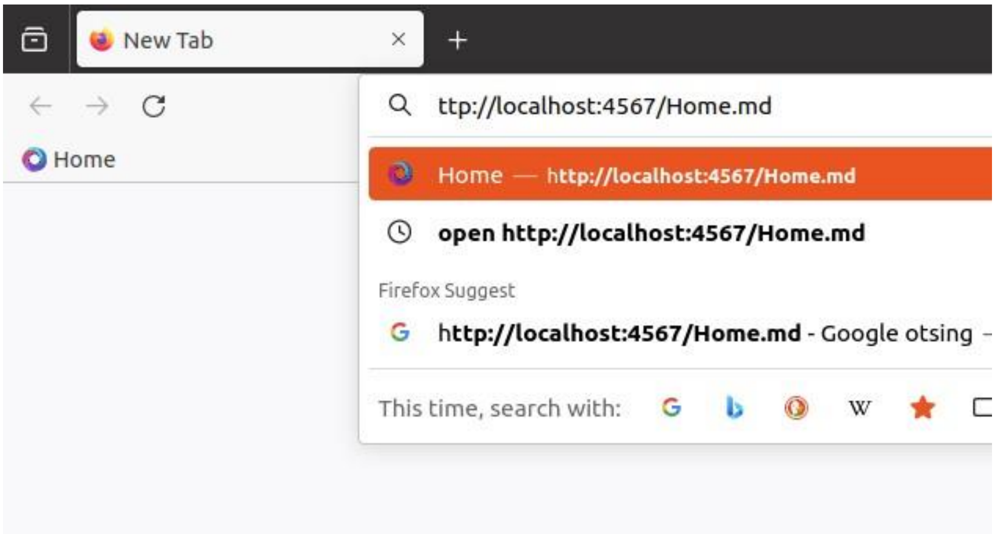
   See Figure 17 for the expected status and proceed with Step 4 below.



```
taltech1804@ubuntuosm:~$ cd /home/taltech1804/taltech.wiki
taltech1804@ubuntuosm:~/taltech.wiki$ gollum
[2024-04-17 16:23:27] INFO  WEBrick 1.8.1
[2024-04-17 16:23:27] INFO  ruby 2.7.0 (2019-12-25) [x86_64-linux-gnu]
== Sinatra (v2.2.3) has taken the stage on 4567 for production with backup from
WEBrick
[2024-04-17 16:23:27] INFO  WEBrick::HTTPServer#start: pid=3153012 port=4567
```

Figure 17. O5GC manual Wiki activation

4) As in Figure 18, open a browser and type: **http://localhost:4567/Home.md**



New Tab
Home
ttp://localhost:4567/Home.md
Home — http://localhost:4567/Home.md
open http://localhost:4567/Home.md
Firefox Suggest
http://localhost:4567/Home.md - Google otsing -
This time, search with:

Figure 18. Starting O5GC manual Wiki in a browser.

This chapter has provided an overview of the relevant parts of the 3GPP 5G standards and of the O5GC system, both with a focus on the registration of a UE. The chapter has also provided practical information for using the O5GC simulator to implement the registration procedure. The next chapter provides an example of a laboratory work that builds upon this.

4. Laboratory work example “Getting started”

This chapter builds upon the knowledge and experience documented in the previous chapter in terms of UE registration to the network. The chapter provides a laboratory work example that can serve as a baseline for the future course and training holders at TalTech.

------------- *Start of laboratory work instructions* -------------

Lab 1 – Getting started

By: Taivo Miller

06.05.2024

Objective of the laboratory work

This laboratory work consist of three tasks:

1) Start the simulator and get to know how the O5GC simulator works.
2) The second part is designed to perform the initial registration, initial registration with PDU and deregistration, analyse the results, and compare them with the relevant 3GPP standard elements.
3) The third part is designed to run an initial registration which will purposefully cause an error and needs to be analysed.

Laboratory specific work goals

1. To get to know how the O5GC simulator works.
2. To perform UE registration to the network and compare the commands with the relevant 3GPP standard. 3GPP standard registration flow is shown under general registration found in TS 23.501, Figure 19, or is provided by the lab instructor with the needed commands to use with the simulator.
3. To observe possible ways to understand whether the UE is registered and connected to the internet.

Requirements for documenting the lab work

- Cover page: Course description and title, lab description and title, student name and code, conduct date.

- General requirements: numbered pages, captioned tables and figures, proper referencing to figures/tables.
- Main content: lab introduction, task/experiment description, results explanation, conclusion.

Notes

5G in the O5GC simulator is a SA technology. It works in an ideal environment and computer hardware does not need to be highly powerful. The systems need Linux operating system and VirtualBox to run it. O5GC is already installed on the laboratory computers and student can start it and run it. Terminal command center is used additionally to bring out comfortable view window size. To use the simulator, it is needed to have knowledge's about TMUX commands. TMUX commands help to navigate under the simulator and run it. Be aware of copy paste commands. They will not work as with Windows or other operation systems.

The proper commands are described in what follows.

Using O5GCore simulator instructions

Some commands are provided for how to use Linux and TMUX under terminal command and under the simulator.

Selection of TMUX commands to navigate under simulator

1. **Ctrl + B** and **1 to 9** – selects feature
2. **Ctrl + B** and **N** – selects next feature
3. **Ctrl + B** and **P** – selects previous feature
4. **Ctrl + B** and **arrow down or up** – selects under Ue1 or Ue2 cursor location on the window
5. **TAB** – list of commands under feature.
6. **Ctrl + B** and **D** – detach. Exit simulator

Used simulator commands under UE and description:

1. **ue_5g_nas_only.registration_request** – sends registration request message to the network for 3GPP and non-3GPP access.
2. **ue_5g_nas_only.pdu_session_establishment_request** – sends PDU session establishment request to the network. UE can access the internet.
3. **ue_5g_nas_only.pdu_session_release_request –** sends PDU session release request to the network.
4. **ue_5g_nas_only.registration_request_and_pdu_connesction** – sends registration request message to the network and right after PDU session establishment request.
5. **ue_5g_nas_only.an_conn_release_command –** Trigger AN connection release to the connected gNB.
6. **ue_5g_nas_only.deregistration_request -** sends deregistration request message to the network.
7. **ue_5g_nas_only.print.ue_context –** Prints UE current status.

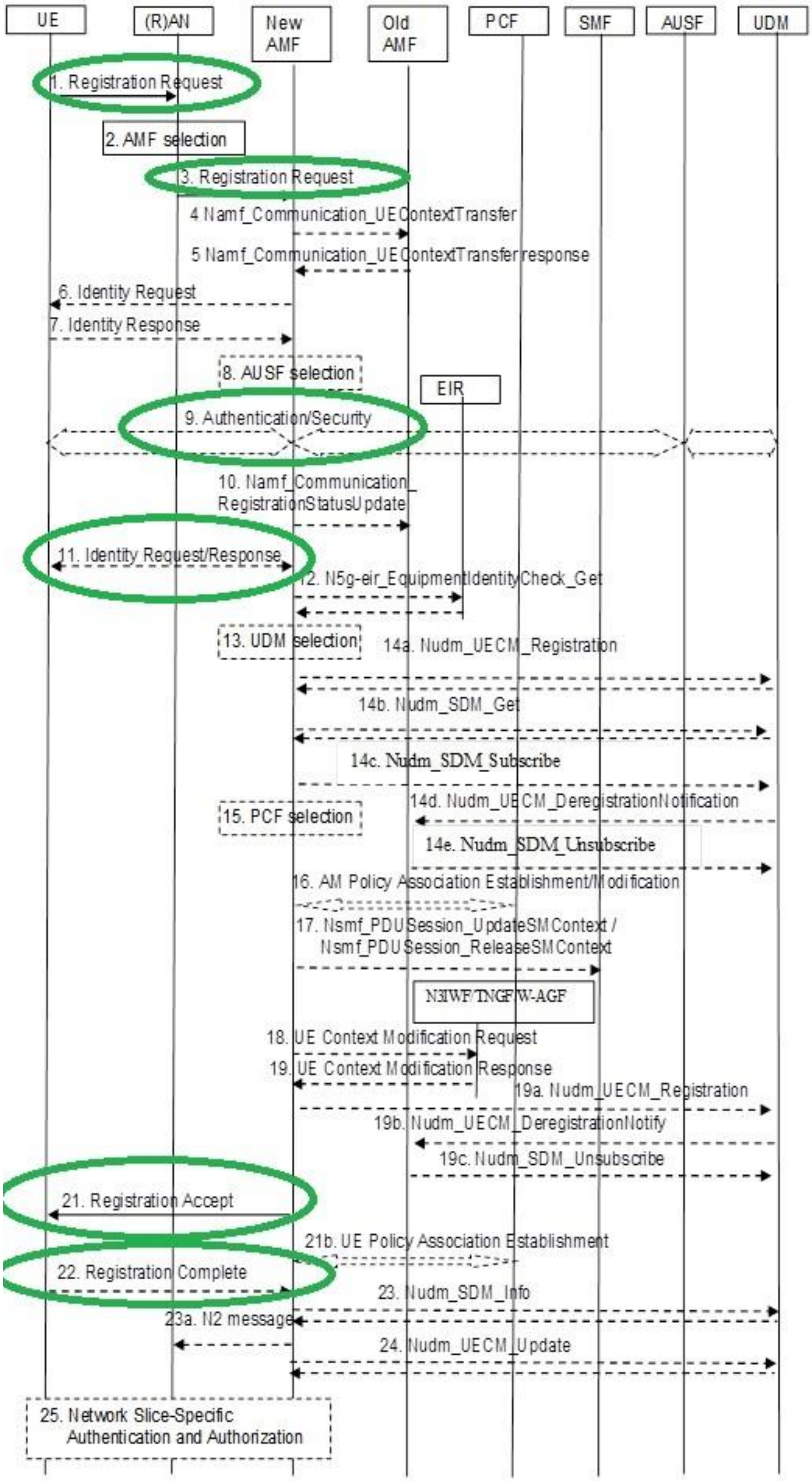

UE
(R)AN
New AMF
Old AMF
PCF
SMF
AUSF
UDM
1. Registration Request
2. AMF selection
3. Registration Request
4 Namf_Communication_UEContextTransfer
5 Namf_Communication_UEContextTransfer response
6. Identity Request
7. Identity Response
8. AUSF selection
EIR
9. Authentication/Security
10. Namf_Communication_RegistrationStatusUpdate
11. Identity Request/Response
12. N5g-eir_EquipmentIdentityCheck_Get
13. UDM selection
14a. Nudm_UECM_Registration
14b. Nudm_SDM_Get
14c. Nudm_SDM_Subscribe
14d. Nudm_UECM_DeregistrationNotification
15. PCF selection
14e. Nudm_SDM_Unsubscribe
16. AM Policy Association Establishment/Modification
17. Nsmf_PDUSession_UpdateSMContext / Nsmf_PDUSession_ReleaseSMContext
N3IWF/TNGF/W-AGF
18. UE Context Modification Request
19. UE Context Modification Response
19a. Nudm_UECM_Registration
19b. Nudm_UECM_DeregistrationNotify
19c. Nudm_SDM_Unsubscribe
21. Registration Accept
21b. UE Policy Association Establishment
22. Registration Complete
23. Nudm_SDM_Info
23a. N2 message
24. Nudm_UECM_Update
25. Network Slice-Specific Authentication and Authorization

Figure 19. General registration [23]. Green circles mark is the needed content under simulator from UE side.

When Initial registration is requested, the green circles in Figure 19 highlight the commands that the student should find.

4.1 Starting VirtualBox and O5GC simulator

Please follow the procedure to start the simulator program.

Step 1: Start and log in to the computer. Login password is provided by the instructor.

Step 2: Start VirtualBox. See picture 1 named Figure 20.

Type in the search box “VirtualBox” and press ENTER.

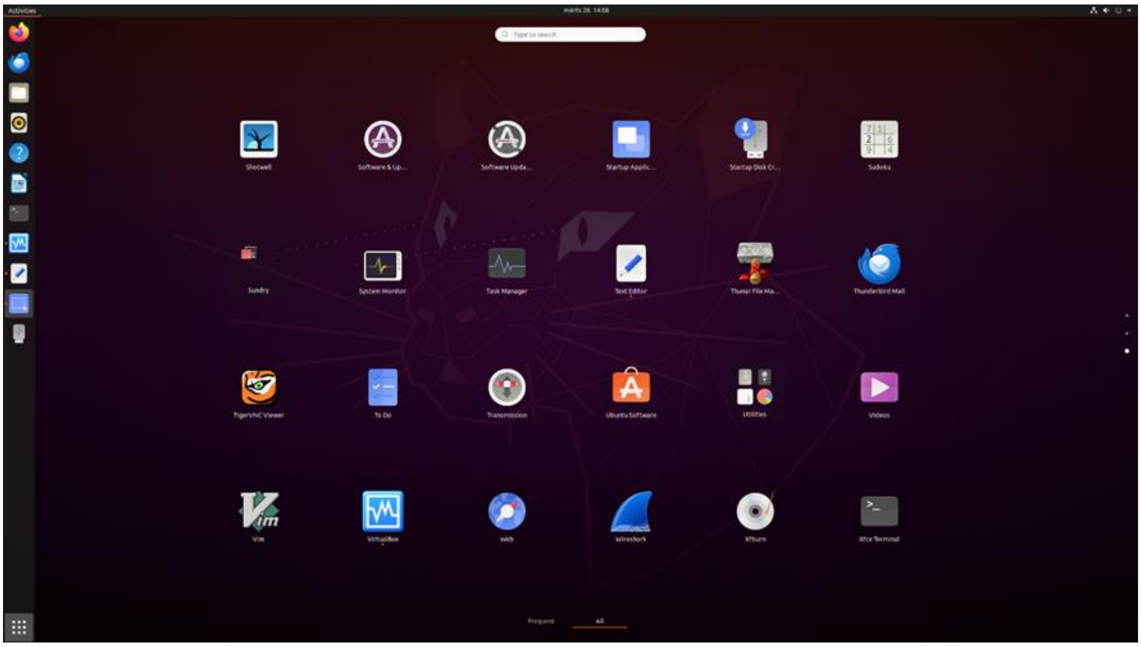
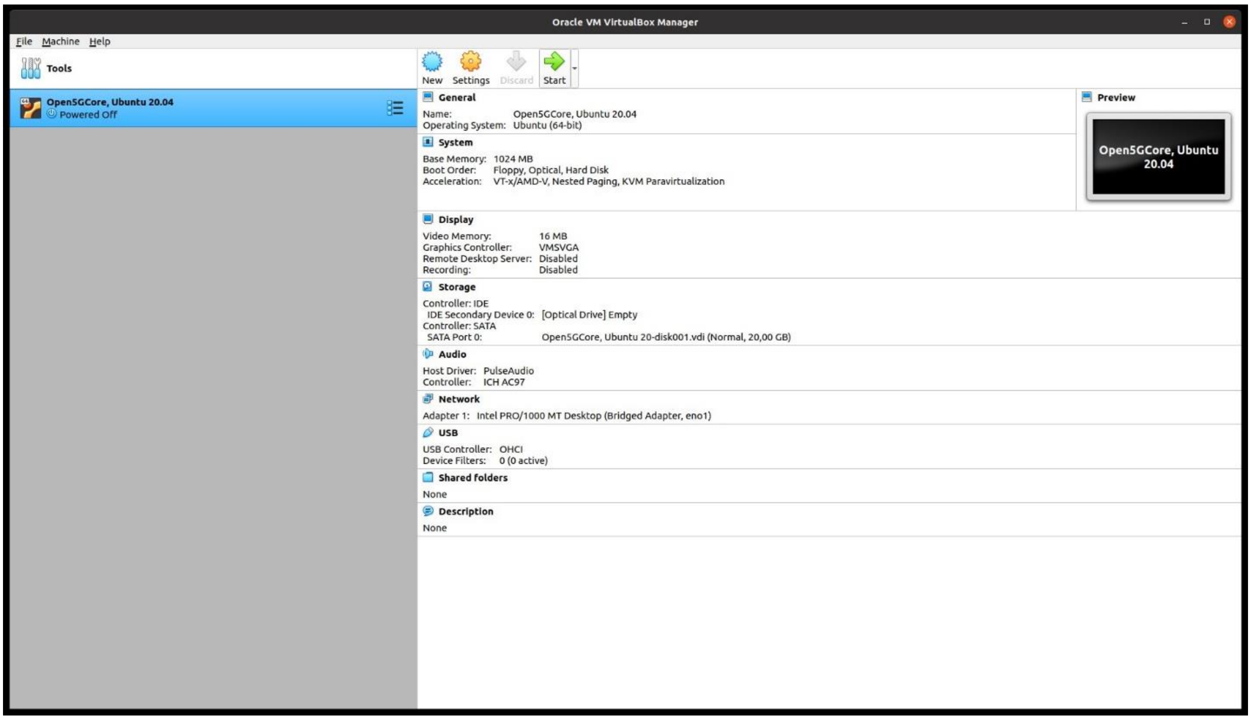


Figure 20. Linux start menu

Step 3: Select Open5GCore and press the START green arrow up in VirtualBox (Figure 21).



Open5GCore, Ubuntu 20.04
Powered Off

General

```
Name:               Open5GCore, Ubuntu 20.04
Operating System:   Ubuntu (64-bit)
```

System

```
Base Memory:   1024 MB
Boot Order:    Floppy, Optical, Hard Disk
Acceleration:  VT-x/AMD-V, Nested Paging, KVM Paravirtualization
```

Display

```
Video Memory:            16 MB
Graphics Controller:     VMSVGA
Remote Desktop Server:   Disabled
Recording:               Disabled
```

Storage

```
Controller: IDE
 IDE Secondary Device 0:  [Optical Drive] Empty
Controller: SATA
 SATA Port 0:             Open5GCore, Ubuntu 20-disk001.vdi (Normal, 20,00 GB)
```

Audio

```
Host Driver:  PulseAudio
Controller:   ICH AC97
```

Network

```
Adapter 1:  Intel PRO/1000 MT Desktop (Bridged Adapter, eno1)
```

USB

```
USB Controller:  OHCI
Device Filters:  0 (0 active)
```

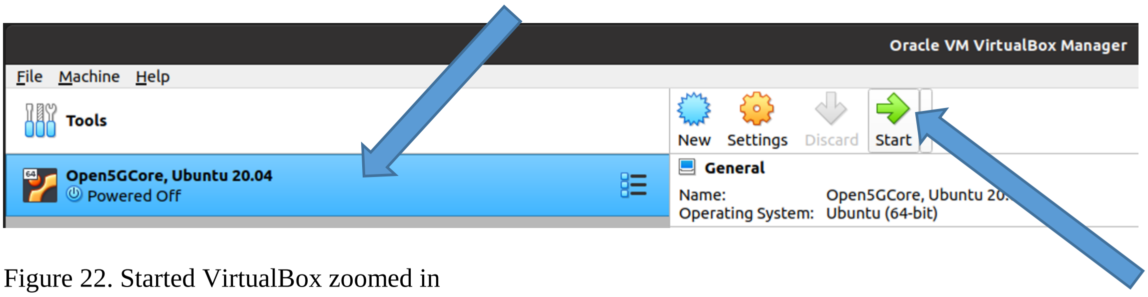
Shared folders

None

Description

None

Figure 21. Started VirtualBox running view



Open5GCore, Ubuntu 20.04
Powered Off

```
Name:               Open5GCore, Ubuntu 20.
Operating System:   Ubuntu (64-bit)
```

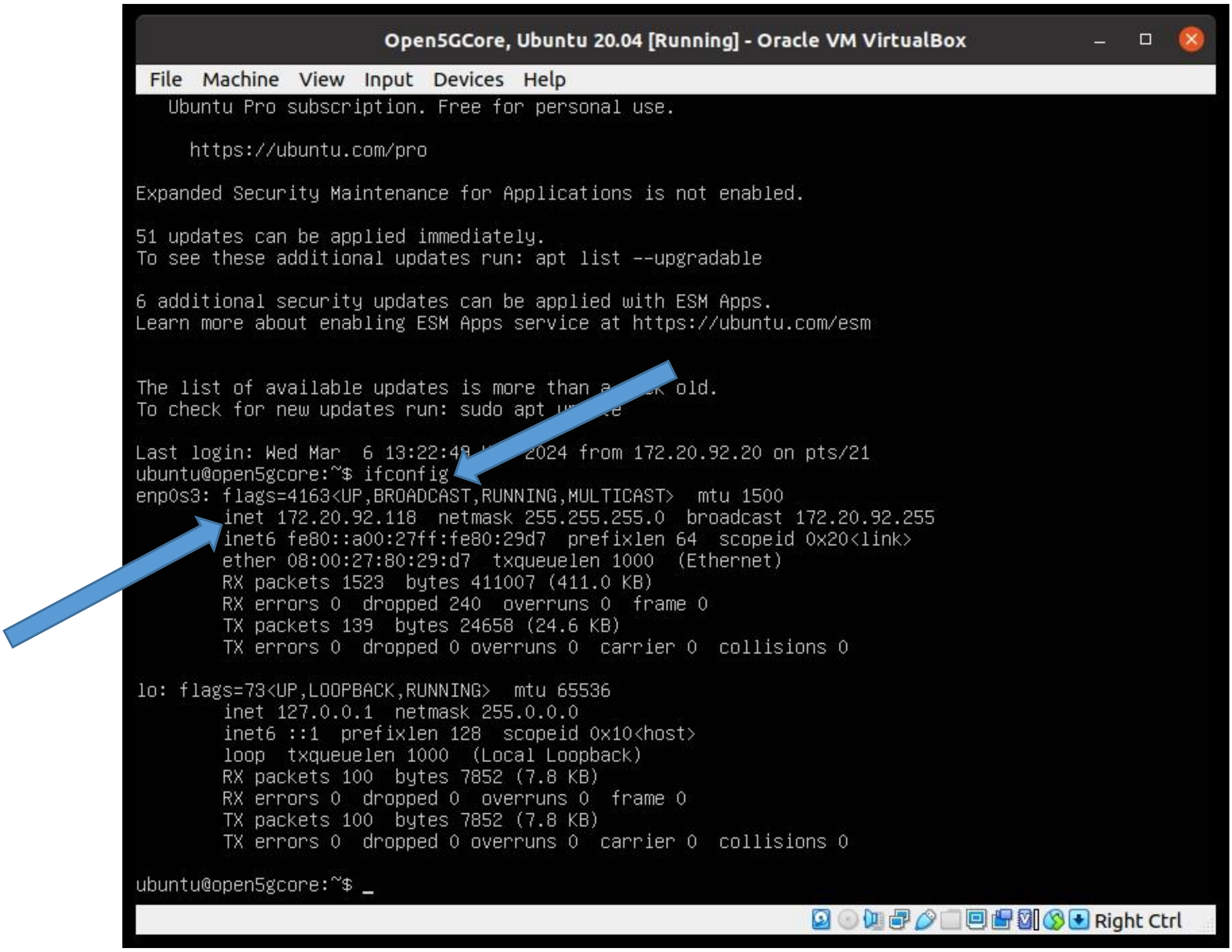
Figure 22. Started VirtualBox zoomed in

As per Figure 22, select Open5GCore, Ubuntu 20.04 and push START

Wait till VirtualBox has started.

Next it is recommended to use Linux terminal window because Virtual box’s problem is its small window size and it is not possible to resize it. That is why the next commands are typed under Linux terminal window. Before that, it is needed to find out the simulator IP address and to log in under a terminal window.

Step 4: Find out virtual machine's IP address typing “ifconfig” (Figure 23).



```
  Ubuntu Pro subscription. Free for personal use.

    https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

51 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

6 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Wed Mar  6 13:22:48 UTC 2024 from 172.20.92.20 on pts/21
ubuntu@open5gcore:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.20.92.118  netmask 255.255.255.0  broadcast 172.20.92.255
        inet6 fe80::a00:27ff:fe80:29d7  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:80:29:d7  txqueuelen 1000  (Ethernet)
        RX packets 1523  bytes 411007 (411.0 KB)
        RX errors 0  dropped 240  overruns 0  frame 0
        TX packets 139  bytes 24658 (24.6 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
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 100  bytes 7852 (7.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 100  bytes 7852 (7.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ubuntu@open5gcore:~$ _
```

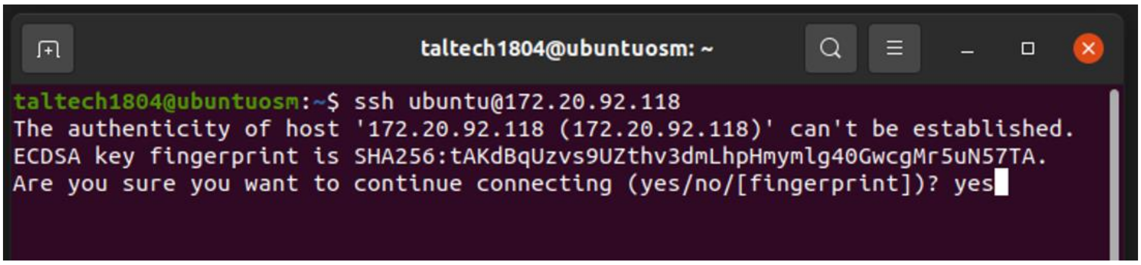
Figure 23. IP address view

To find the IP address from VirtualBox window, look under enp0s3. This is Virtual Box adapter name. But things can be a little bit different than what the student can see under the graphic view, see Figure 21. Under VirtualBox, there is eno1, see Figure 21

All other steps are now under terminal commands, see Appendix 5.

Step 5: Open the terminal command centre. Log in into virtual machine via SSH, username: ubuntu password: Ubuntu

As per Figure 24, type: ssh ubuntu@*(collected ip address)*



```
taltech1804@ubuntuosm:~$ ssh ubuntu@172.20.92.118
The authenticity of host '172.20.92.118 (172.20.92.118)' can't be established.
ECDSA key fingerprint is SHA256:tAKdBqUzvs9UZthv3dmLhpHmymlg40GwcgMr5uN57TA.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

Figure 24. Terminal window command view

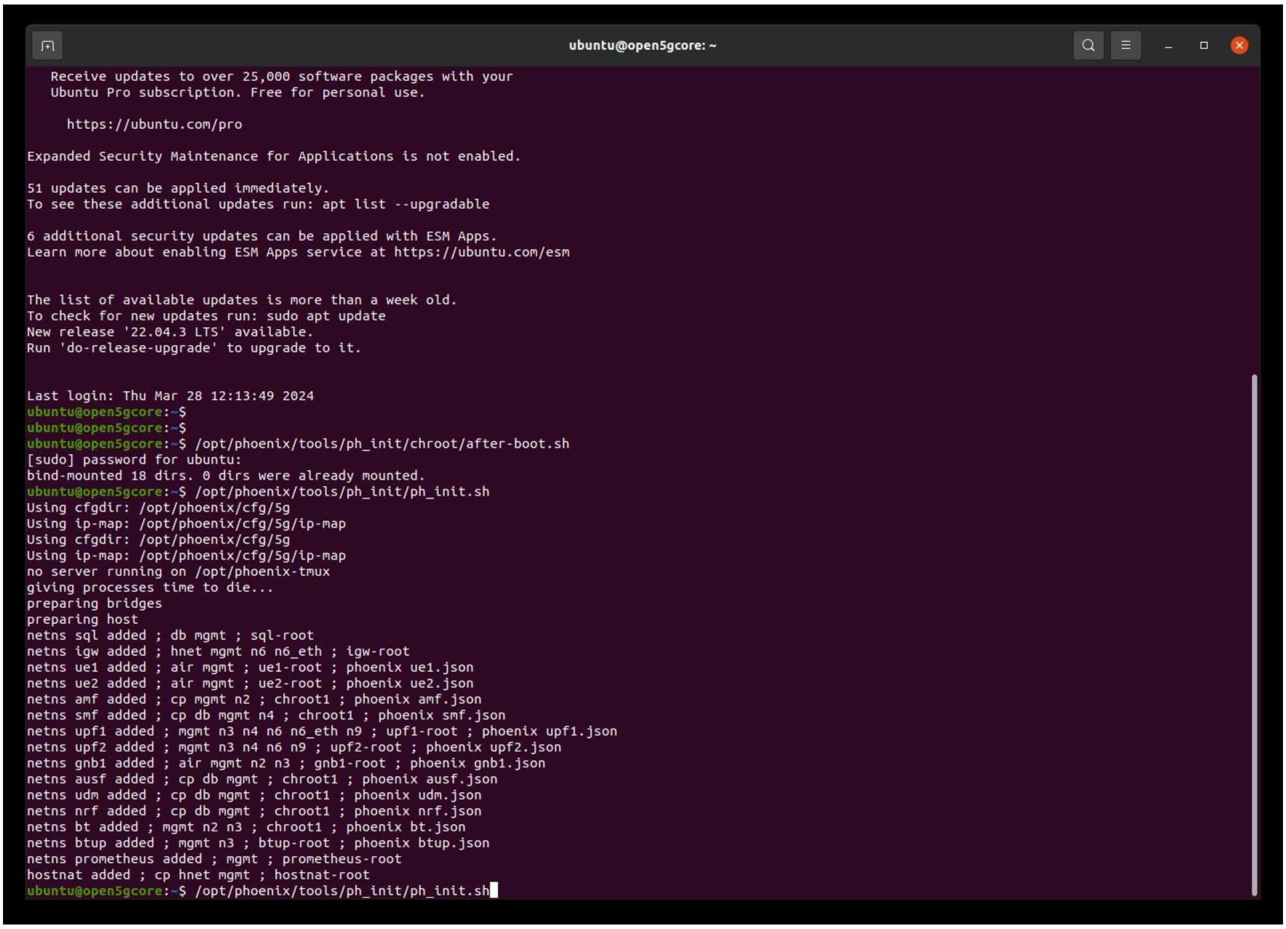
Step 6: If needed authenticity of host then type “yes”

Step 7: Type /opt/phoenix/tools/ph_init/chroot/after-boot.sh

PS! If you do not press Enter within a certain time, the command gives you “Connection closed”; if so, repeat the command again.

Step 8: As per Figure 25, type: /opt/phoenix/tools/ph_init/ph_init.sh

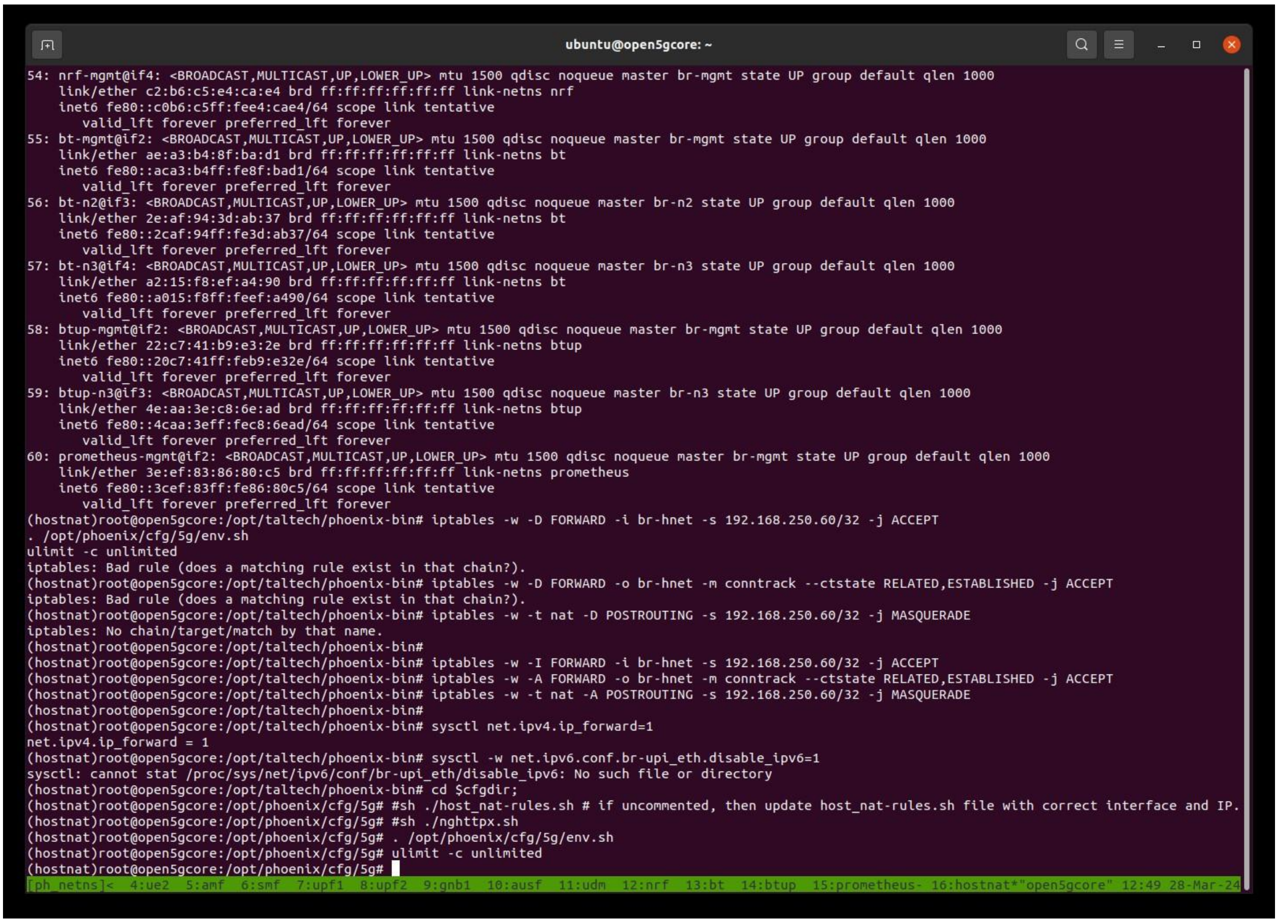
Step 9: Type again like in Step 8: /opt/phoenix/tools/ph_init/ph_init.sh push ENTER



```
   Receive updates to over 25,000 software packages with your
   Ubuntu Pro subscription. Free for personal use.

     https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

51 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

6 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Thu Mar 28 12:13:49 2024
ubuntu@open5gcore:~$
ubuntu@open5gcore:~$
ubuntu@open5gcore:~$ /opt/phoenix/tools/ph_init/chroot/after-boot.sh
[sudo] password for ubuntu:
bind-mounted 18 dirs. 0 dirs were already mounted.
ubuntu@open5gcore:~$ /opt/phoenix/tools/ph_init/ph_init.sh
Using cfgdir: /opt/phoenix/cfg/5g
Using ip-map: /opt/phoenix/cfg/5g/ip-map
Using cfgdir: /opt/phoenix/cfg/5g
Using ip-map: /opt/phoenix/cfg/5g/ip-map
no server running on /opt/phoenix-tmux
giving processes time to die...
preparing bridges
preparing host
netns sql added ; db mgmt ; sql-root
netns igw added ; hnet mgmt n6 n6_eth ; igw-root
netns ue1 added ; air mgmt ; ue1-root ; phoenix ue1.json
netns ue2 added ; air mgmt ; ue2-root ; phoenix ue2.json
netns amf added ; cp mgmt n2 ; chroot1 ; phoenix amf.json
netns smf added ; cp db mgmt n4 ; chroot1 ; phoenix smf.json
netns upf1 added ; mgmt n3 n4 n6 n6_eth n9 ; upf1-root ; phoenix upf1.json
netns upf2 added ; mgmt n3 n4 n6 n9 ; upf2-root ; phoenix upf2.json
netns gnb1 added ; air mgmt n2 n3 ; gnb1-root ; phoenix gnb1.json
netns ausf added ; cp db mgmt ; chroot1 ; phoenix ausf.json
netns udm added ; cp db mgmt ; chroot1 ; phoenix udm.json
netns nrf added ; cp db mgmt ; chroot1 ; phoenix nrf.json
netns bt added ; mgmt n2 n3 ; chroot1 ; phoenix bt.json
netns btup added ; mgmt n3 ; btup-root ; phoenix btup.json
netns prometheus added ; mgmt ; prometheus-root
hostnat added ; cp hnet mgmt ; hostnat-root
ubuntu@open5gcore:~$ /opt/phoenix/tools/ph_init/ph_init.sh
```

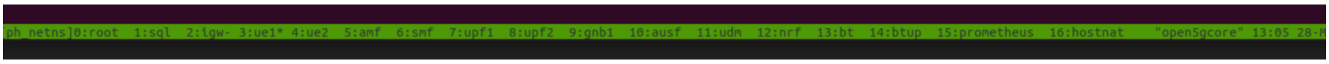
Figure 25. Last command before running simulator

Now O5GC simulator is started and running (Figure 26).



```
54: nrf-mgmt@if4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-mgmt state UP group default qlen 1000
    link/ether c2:b6:c5:e4:ca:e4 brd ff:ff:ff:ff:ff:ff link-netns nrf
    inet6 fe80::c0b6:c5ff:fee4:cae4/64 scope link tentative
       valid_lft forever preferred_lft forever
55: bt-mgmt@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-mgmt state UP group default qlen 1000
    link/ether ae:a3:b4:8f:ba:d1 brd ff:ff:ff:ff:ff:ff link-netns bt
    inet6 fe80::aca3:b4ff:fe8f:bad1/64 scope link tentative
       valid_lft forever preferred_lft forever
56: bt-n2@if3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-n2 state UP group default qlen 1000
    link/ether 2e:af:94:3d:ab:37 brd ff:ff:ff:ff:ff:ff link-netns bt
    inet6 fe80::2caf:94ff:fe3d:ab37/64 scope link tentative
       valid_lft forever preferred_lft forever
57: bt-n3@if4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-n3 state UP group default qlen 1000
    link/ether a2:15:f8:ef:a4:90 brd ff:ff:ff:ff:ff:ff link-netns bt
    inet6 fe80::a015:f8ff:feef:a490/64 scope link tentative
       valid_lft forever preferred_lft forever
58: btup-mgmt@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-mgmt state UP group default qlen 1000
    link/ether 22:c7:41:b9:e3:2e brd ff:ff:ff:ff:ff:ff link-netns btup
    inet6 fe80::20c7:41ff:feb9:e32e/64 scope link tentative
       valid_lft forever preferred_lft forever
59: btup-n3@if3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-n3 state UP group default qlen 1000
    link/ether 4e:aa:3e:c8:6e:ad brd ff:ff:ff:ff:ff:ff link-netns btup
    inet6 fe80::4caa:3eff:fec8:6ead/64 scope link tentative
       valid_lft forever preferred_lft forever
60: prometheus-mgmt@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-mgmt state UP group default qlen 1000
    link/ether 3e:ef:83:86:80:c5 brd ff:ff:ff:ff:ff:ff link-netns prometheus
    inet6 fe80::3cef:83ff:fe86:80c5/64 scope link tentative
       valid_lft forever preferred_lft forever
(hostnat)root@open5gcore:/opt/taltech/phoenix-bin# iptables -w -D FORWARD -i br-hnet -s 192.168.250.60/32 -j ACCEPT
. /opt/phoenix/cfg/5g/env.sh
ulimit -c unlimited
iptables: Bad rule (does a matching rule exist in that chain?).
(hostnat)root@open5gcore:/opt/taltech/phoenix-bin# iptables -w -D FORWARD -o br-hnet -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
iptables: Bad rule (does a matching rule exist in that chain?).
(hostnat)root@open5gcore:/opt/taltech/phoenix-bin# iptables -w -t nat -D POSTROUTING -s 192.168.250.60/32 -j MASQUERADE
iptables: No chain/target/match by that name.
(hostnat)root@open5gcore:/opt/taltech/phoenix-bin#
(hostnat)root@open5gcore:/opt/taltech/phoenix-bin# iptables -w -I FORWARD -i br-hnet -s 192.168.250.60/32 -j ACCEPT
(hostnat)root@open5gcore:/opt/taltech/phoenix-bin# iptables -w -A FORWARD -o br-hnet -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
(hostnat)root@open5gcore:/opt/taltech/phoenix-bin# iptables -w -t nat -A POSTROUTING -s 192.168.250.60/32 -j MASQUERADE
(hostnat)root@open5gcore:/opt/taltech/phoenix-bin#
(hostnat)root@open5gcore:/opt/taltech/phoenix-bin# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
(hostnat)root@open5gcore:/opt/taltech/phoenix-bin# sysctl -w net.ipv6.conf.br-upi_eth.disable_ipv6=1
sysctl: cannot stat /proc/sys/net/ipv6/conf/br-upi_eth/disable_ipv6: No such file or directory
(hostnat)root@open5gcore:/opt/taltech/phoenix-bin# cd $cfgdir;
(hostnat)root@open5gcore:/opt/phoenix/cfg/5g# #sh ./host_nat-rules.sh # if uncommented, then update host_nat-rules.sh file with correct interface and IP.
(hostnat)root@open5gcore:/opt/phoenix/cfg/5g# #sh ./nghttpx.sh
(hostnat)root@open5gcore:/opt/phoenix/cfg/5g# . /opt/phoenix/cfg/5g/env.sh
(hostnat)root@open5gcore:/opt/phoenix/cfg/5g# ulimit -c unlimited
(hostnat)root@open5gcore:/opt/phoenix/cfg/5g#
[ph_netns]<  4:ue2  5:amf  6:smf  7:upf1  8:upf2  9:gnb1  10:ausf  11:udm  12:nrf  13:bt  14:btup  15:prometheus- 16:hostnat*"open5gcore" 12:49 28-Mar-24
```

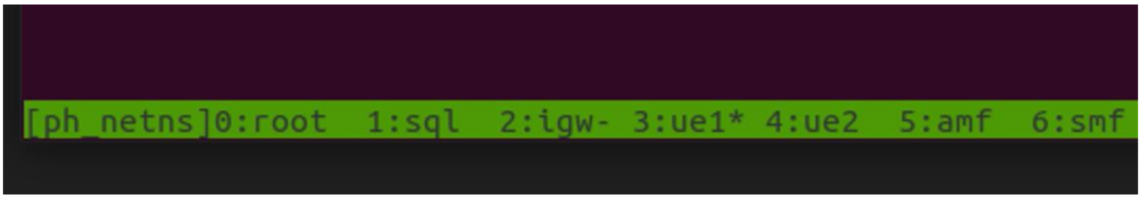
Figure 26. O5GCore simulator view



```
[ph_netns]0:root  1:sql  2:igw- 3:ue1* 4:ue2  5:amf  6:smf  7:upf1  8:upf2  9:gnb1  10:ausf  11:udm  12:nrf  13:bt  14:btup  15:prometheus  16:hostnat    "open5gcore" 13:05 28-M
```

Figure 27. O5GC features view

Figure 27 and Figure 28: this window is provided to show the current function view, which function is active, for example “ue1*”. Change the view using shortcut keys, which are described above in the heading “Selection of TMUX commands to navigate under simulator.”



```
[ph_netns]0:root  1:sql  2:igw- 3:ue1* 4:ue2  5:amf  6:smf
```

Figure 28. Zoomed marked function view

Active window is ue1. Highlighted with “*”.

Step 10: If you want to stop the simulator, enter **Ctrl + B** and **D** – detach and then type: /opt/phoenix/tools/ph_init/ph_init.sh down

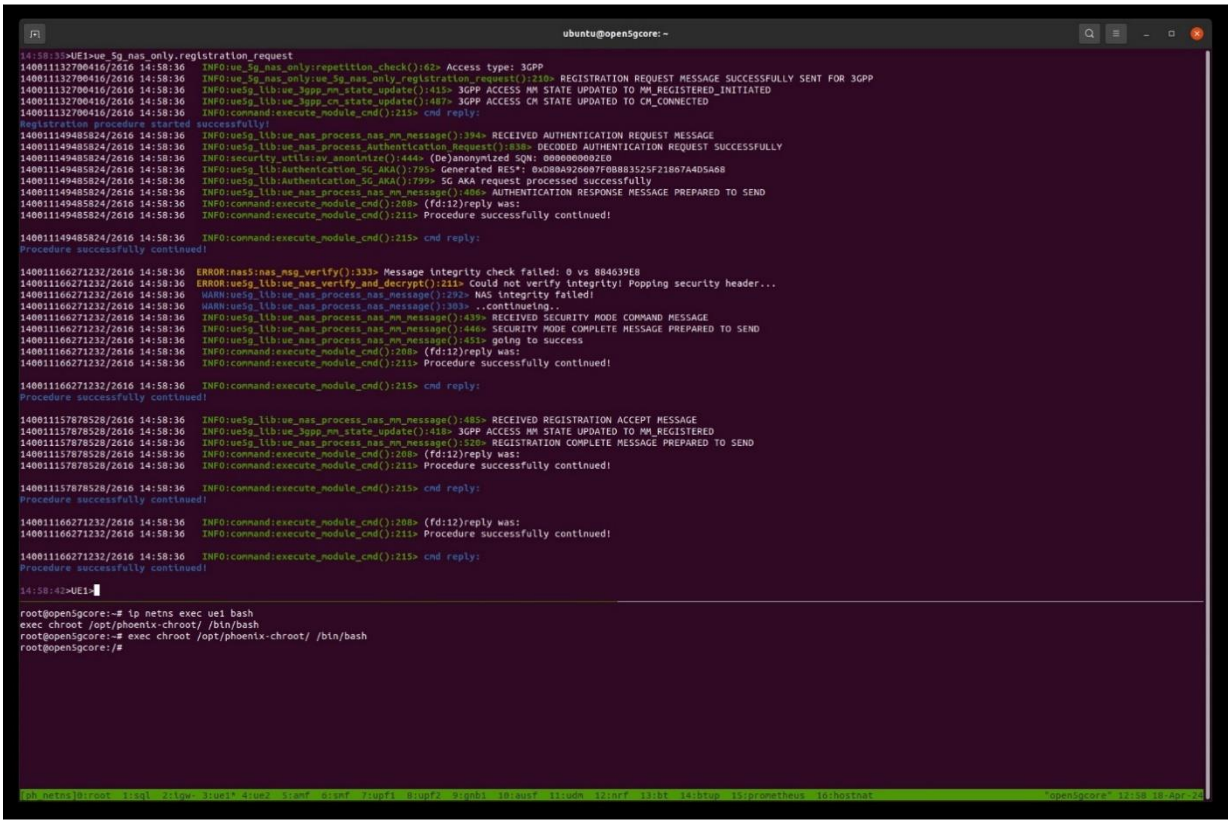
This will shut down the simulator and you can close the terminal window typing exit. Password: ubuntu

Note: If you have not exited and want to restart simulator again, then type: /opt/phoenix/tools/ph_init/ph_init.sh

4.2 Task 1: UE Initial registration.

1. Start the simulator.
2. Run under “ue1” the command needed to perform “Initial registration only”.
3. Write down what command is used for registration.
4. Write down what happened and whether it succeeded. This should be identified in the command line.
5. Write down what Network Functions (NF) were in use to get registered.
6. Compare it with the 3GPP standard TS 23.501 General registration call flow with figure 19.
7. Explain why UE1 cannot access the internet?
8. Add and compare 3GPP standard figures and simulator figures.

Step 1: Go to under “ue1” using TMUX command and make initial registration only. See Figure 29.



```
14:58:35>UE1>ue_5g_nas_only.registration_request
140011132700416/2616 14:58:36   INFO:ue_5g_nas_only:repetition_check():62> Access type: 3GPP
140011132700416/2616 14:58:36   INFO:ue_5g_nas_only:ue_5g_nas_only_registration_request():210> REGISTRATION REQUEST MESSAGE SUCCESSFULLY SENT FOR 3GPP
140011132700416/2616 14:58:36   INFO:ue5g_lib:ue_3gpp_mm_state_update():415> 3GPP ACCESS MM STATE UPDATED TO MM_REGISTERED_INITIATED
140011132700416/2616 14:58:36   INFO:ue5g_lib:ue_3gpp_cm_state_update():487> 3GPP ACCESS CM STATE UPDATED TO CM_CONNECTED
140011132700416/2616 14:58:36   INFO:command:execute_module_cmd():215> cmd reply:
Registration procedure started successfully!
140011149485824/2616 14:58:36   INFO:ue5g_lib:ue_nas_process_nas_mm_message():394> RECEIVED AUTHENTICATION REQUEST MESSAGE
140011149485824/2616 14:58:36   INFO:ue5g_lib:ue_nas_process_Authentication_Request():838> DECODED AUTHENTICATION REQUEST SUCCESSFULLY
140011149485824/2616 14:58:36   INFO:security_utils:av_anonimize():444> (De)anonymized SQN: 0000000002E0
140011149485824/2616 14:58:36   INFO:ue5g_lib:Authentication_5G_AKA():795> Generated RES*: 0xD80A926007F0B883525F21867A4D5A68
140011149485824/2616 14:58:36   INFO:ue5g_lib:Authentication_5G_AKA():799> 5G AKA request processed successfully
140011149485824/2616 14:58:36   INFO:ue5g_lib:ue_nas_process_nas_mm_message():406> AUTHENTICATION RESPONSE MESSAGE PREPARED TO SEND
140011149485824/2616 14:58:36   INFO:command:execute_module_cmd():208> (fd:12)reply was:
140011149485824/2616 14:58:36   INFO:command:execute_module_cmd():211> Procedure successfully continued!

140011149485824/2616 14:58:36   INFO:command:execute_module_cmd():215> cmd reply:
Procedure successfully continued!

140011166271232/2616 14:58:36  ERROR:nas5:nas_msg_verify():333> Message integrity check failed: 0 vs 884639E8
140011166271232/2616 14:58:36  ERROR:ue5g_lib:ue_nas_verify_and_decrypt():211> Could not verify integrity! Popping security header...
140011166271232/2616 14:58:36   WARN:ue5g_lib:ue_nas_process_nas_message():292> NAS integrity failed!
140011166271232/2616 14:58:36   WARN:ue5g_lib:ue_nas_process_nas_message():303> ..continueing..
140011166271232/2616 14:58:36   INFO:ue5g_lib:ue_nas_process_nas_mm_message():439> RECEIVED SECURITY MODE COMMAND MESSAGE
140011166271232/2616 14:58:36   INFO:ue5g_lib:ue_nas_process_nas_mm_message():446> SECURITY MODE COMPLETE MESSAGE PREPARED TO SEND
140011166271232/2616 14:58:36   INFO:ue5g_lib:ue_nas_process_nas_mm_message():451> going to success
140011166271232/2616 14:58:36   INFO:command:execute_module_cmd():208> (fd:12)reply was:
140011166271232/2616 14:58:36   INFO:command:execute_module_cmd():211> Procedure successfully continued!

140011166271232/2616 14:58:36   INFO:command:execute_module_cmd():215> cmd reply:
Procedure successfully continued!

140011157878528/2616 14:58:36   INFO:ue5g_lib:ue_nas_process_nas_mm_message():485> RECEIVED REGISTRATION ACCEPT MESSAGE
140011157878528/2616 14:58:36   INFO:ue5g_lib:ue_3gpp_mm_state_update():418> 3GPP ACCESS MM STATE UPDATED TO MM_REGISTERED
140011157878528/2616 14:58:36   INFO:ue5g_lib:ue_nas_process_nas_mm_message():520> REGISTRATION COMPLETE MESSAGE PREPARED TO SEND
140011157878528/2616 14:58:36   INFO:command:execute_module_cmd():208> (fd:12)reply was:
140011157878528/2616 14:58:36   INFO:command:execute_module_cmd():211> Procedure successfully continued!

140011157878528/2616 14:58:36   INFO:command:execute_module_cmd():215> cmd reply:
Procedure successfully continued!

140011166271232/2616 14:58:36   INFO:command:execute_module_cmd():208> (fd:12)reply was:
140011166271232/2616 14:58:36   INFO:command:execute_module_cmd():211> Procedure successfully continued!

140011166271232/2616 14:58:36   INFO:command:execute_module_cmd():215> cmd reply:
Procedure successfully continued!

14:58:42>UE1>
```

```
root@open5gcore:~# ip netns exec ue1 bash
exec chroot /opt/phoenix-chroot/ /bin/bash
root@open5gcore:~# exec chroot /opt/phoenix-chroot/ /bin/bash
root@open5gcore:/#
```

```
[ph_netns]0:root  1:sql  2:igw-  3:ue1* 4:ue2  5:amf  6:smf  7:upf1  8:upf2  9:gnb1  10:ausf  11:udm  12:nrf  13:bt  14:btup  15:prometheus  16:hostnat        "open5gcore" 12:58 18-Apr-24
```

Figure 29. UE1 registration request command

Compare results with standard TS 23.501 General registration call flow with Figure 19.

It is possible to check UE status by printing context.

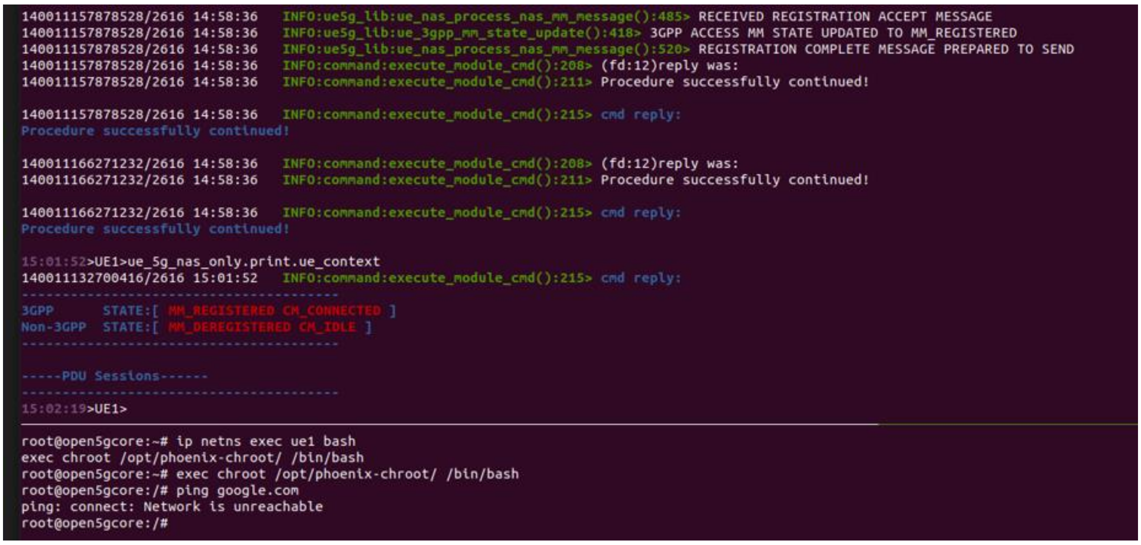
Step 2: Make a status check running command: “ue_5g_nas_only.print.ue_context”

It is possible to check whether the UE has access to the internet.

Step 3: Go down using TMUX command **Ctrl + B** and **arrow down**

Step 4: Run there ping test, for example: ping google.com

It should not be reachable to the internet. See Figure 30.



```
140011157878528/2616 14:58:36   INFO:ue5g_lib:ue_nas_process_nas_mm_message():485> RECEIVED REGISTRATION ACCEPT MESSAGE
140011157878528/2616 14:58:36   INFO:ue5g_lib:ue_3gpp_mm_state_update():418> 3GPP ACCESS MM STATE UPDATED TO MM_REGISTERED
140011157878528/2616 14:58:36   INFO:ue5g_lib:ue_nas_process_nas_mm_message():520> REGISTRATION COMPLETE MESSAGE PREPARED TO SEND
140011157878528/2616 14:58:36   INFO:command:execute_module_cmd():208> (fd:12)reply was:
140011157878528/2616 14:58:36   INFO:command:execute_module_cmd():211> Procedure successfully continued!

140011157878528/2616 14:58:36   INFO:command:execute_module_cmd():215> cmd reply:
Procedure successfully continued!

140011166271232/2616 14:58:36   INFO:command:execute_module_cmd():208> (fd:12)reply was:
140011166271232/2616 14:58:36   INFO:command:execute_module_cmd():211> Procedure successfully continued!

140011166271232/2616 14:58:36   INFO:command:execute_module_cmd():215> cmd reply:
Procedure successfully continued!

15:01:52>UE1>ue_5g_nas_only.print.ue_context
140011132700416/2616 15:01:52   INFO:command:execute_module_cmd():215> cmd reply:
---------------------------------------
3GPP      STATE:[ MM_REGISTERED CM_CONNECTED ]
Non-3GPP  STATE:[ MM_DEREGISTERED CM_IDLE ]
---------------------------------------

-----PDU Sessions------
---------------------------------------
15:02:19>UE1>

root@open5gcore:~# ip netns exec ue1 bash
exec chroot /opt/phoenix-chroot/ /bin/bash
root@open5gcore:~# exec chroot /opt/phoenix-chroot/ /bin/bash
root@open5gcore:/# ping google.com
ping: connect: Network is unreachable
root@open5gcore:/#
```

Figure 30. UE1 is registered to the network.

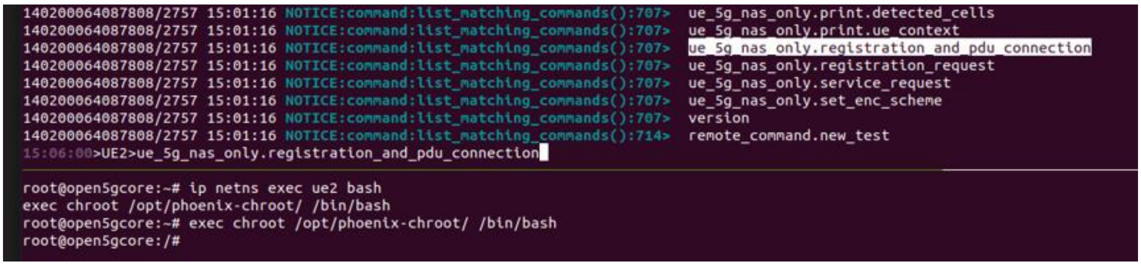
4.3 Task 2: UE2 Initial registration with PDU.

Please follow the procedure to test UE1 and UE2.

1. Run under “UE2” the command “Initial registration with PDU”.
2. Write down UE2 IP address: ………………………
3. Declare that MM is registered, CM is connected, and PDU is active using print.ue command.
4. Make a ping test. Does the UE have access to the internet?
5. Make a ping test, can UE1 see UE2 and vice versa?
6. Run under UE1 the command “ue_5g_nas_only.pdu_session_establishement_request”.
7. Write down UE1 IP address: ………………………
8. Make a ping test, can UE1 see UE2 and vice versa?
9. Run UE2 service de-registration request, using for that the “an_conn_release_command”.
10. Declare that MM is registered and CM is idle.
11. Make a ping test. Does the UE have access to the internet?
12. Run UE2 de-registration using for that “deregistration_request”.
13. Declare that MM is deregistered and CM is idle.

Step 1: Go to under “ue2” using TMUX command.

Step 2: Run command “ue_5g_nas_only.registration_request_and_pdu_connection. See Figure 31.

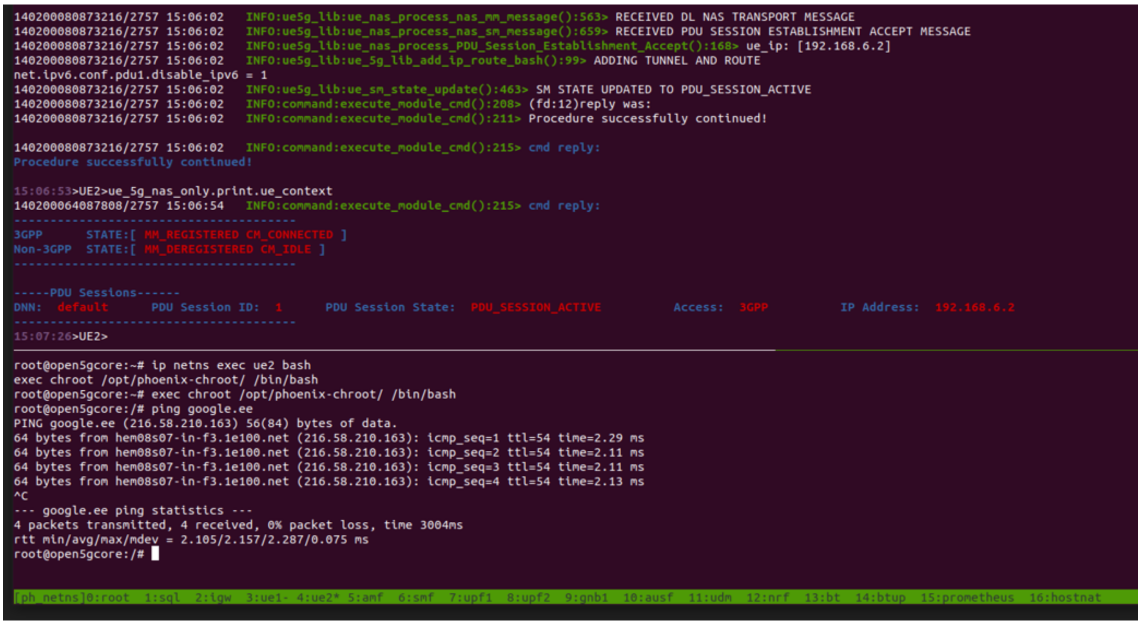


```
140200064087808/2757 15:01:16 NOTICE:command:list_matching_commands():707>  ue_5g_nas_only.print.detected_cells
140200064087808/2757 15:01:16 NOTICE:command:list_matching_commands():707>  ue_5g_nas_only.print.ue_context
140200064087808/2757 15:01:16 NOTICE:command:list_matching_commands():707>  ue_5g_nas_only.registration_and_pdu_connection
140200064087808/2757 15:01:16 NOTICE:command:list_matching_commands():707>  ue_5g_nas_only.registration_request
140200064087808/2757 15:01:16 NOTICE:command:list_matching_commands():707>  ue_5g_nas_only.service_request
140200064087808/2757 15:01:16 NOTICE:command:list_matching_commands():707>  ue_5g_nas_only.set_enc_scheme
140200064087808/2757 15:01:16 NOTICE:command:list_matching_commands():707>  version
140200064087808/2757 15:01:16 NOTICE:command:list_matching_commands():714>  remote_command.new_test
15:06:00>UE2>ue_5g_nas_only.registration_and_pdu_connection

root@open5gcore:~# ip netns exec ue2 bash
exec chroot /opt/phoenix-chroot/ /bin/bash
root@open5gcore:~# exec chroot /opt/phoenix-chroot/ /bin/bash
root@open5gcore:/#
```

Figure 31. UE2 5G registration request and PDU session request

Step 4: Make a status check running the command: “ue_5g_nas_only.print.ue_context”.

Step 5: Make a ping test: for example ping google.ee See Figure 32



```
140200080873216/2757 15:06:02   INFO:ue5g_lib:ue_nas_process_nas_mm_message():563> RECEIVED DL NAS TRANSPORT MESSAGE
140200080873216/2757 15:06:02   INFO:ue5g_lib:ue_nas_process_nas_sm_message():659> RECEIVED PDU SESSION ESTABLISHMENT ACCEPT MESSAGE
140200080873216/2757 15:06:02   INFO:ue5g_lib:ue_nas_process_PDU_Session_Establishment_Accept():168> ue_ip: [192.168.6.2]
140200080873216/2757 15:06:02   INFO:ue5g_lib:ue_5g_lib_add_ip_route_bash():99> ADDING TUNNEL AND ROUTE
net.ipv6.conf.pdu1.disable_ipv6 = 1
140200080873216/2757 15:06:02   INFO:ue5g_lib:ue_sm_state_update():463> SM STATE UPDATED TO PDU_SESSION_ACTIVE
140200080873216/2757 15:06:02   INFO:command:execute_module_cmd():208> (fd:12)reply was:
140200080873216/2757 15:06:02   INFO:command:execute_module_cmd():211> Procedure successfully continued!

140200080873216/2757 15:06:02   INFO:command:execute_module_cmd():215> cmd reply:
Procedure successfully continued!

15:06:53>UE2>ue_5g_nas_only.print.ue_context
140200064087808/2757 15:06:54   INFO:command:execute_module_cmd():215> cmd reply:
---------------------------------------
3GPP      STATE:[ MM_REGISTERED CM_CONNECTED ]
Non-3GPP  STATE:[ MM_DEREGISTERED CM_IDLE ]
---------------------------------------

-----PDU Sessions------
DNN:  default      PDU Session ID:  1      PDU Session State:  PDU_SESSION_ACTIVE         Access:  3GPP          IP Address:  192.168.6.2
---------------------------------------
15:07:26>UE2>

root@open5gcore:~# ip netns exec ue2 bash
exec chroot /opt/phoenix-chroot/ /bin/bash
root@open5gcore:~# exec chroot /opt/phoenix-chroot/ /bin/bash
root@open5gcore:/# ping google.ee
PING google.ee (216.58.210.163) 56(84) bytes of data.
64 bytes from hem08s07-in-f3.1e100.net (216.58.210.163): icmp_seq=1 ttl=54 time=2.29 ms
64 bytes from hem08s07-in-f3.1e100.net (216.58.210.163): icmp_seq=2 ttl=54 time=2.11 ms
64 bytes from hem08s07-in-f3.1e100.net (216.58.210.163): icmp_seq=3 ttl=54 time=2.11 ms
64 bytes from hem08s07-in-f3.1e100.net (216.58.210.163): icmp_seq=4 ttl=54 time=2.13 ms
^C
--- google.ee ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 2.105/2.157/2.287/0.075 ms
root@open5gcore:/#

[ph_netns]0:root  1:sql  2:igw  3:ue1- 4:ue2* 5:amf  6:smf  7:upf1  8:upf2  9:gnb1  10:ausf  11:udm  12:nrf  13:bt  14:btup  15:prometheus  16:hostnat
```

Figure 32. UE2 registration status check

PDU is active.

Step 6: Go down using TMUX command **Ctrl + B** and **arrow down**.

Step 7: Run ping test: ping 192.168.6.1

It should not be possible to ping UE1

Step 8: Run under UE1 the command “ue_5g_nas_only.pdu_session_establishement_request”.

Step 9: Run ping test: ping 192.168.6.1

Step 10: Run command “ue_5g_nas_only.deregistration_request”

Step 11: Make a status check running command: “ue_5g_nas_only.print.ue_context”

4.4 Task 3 (optional)

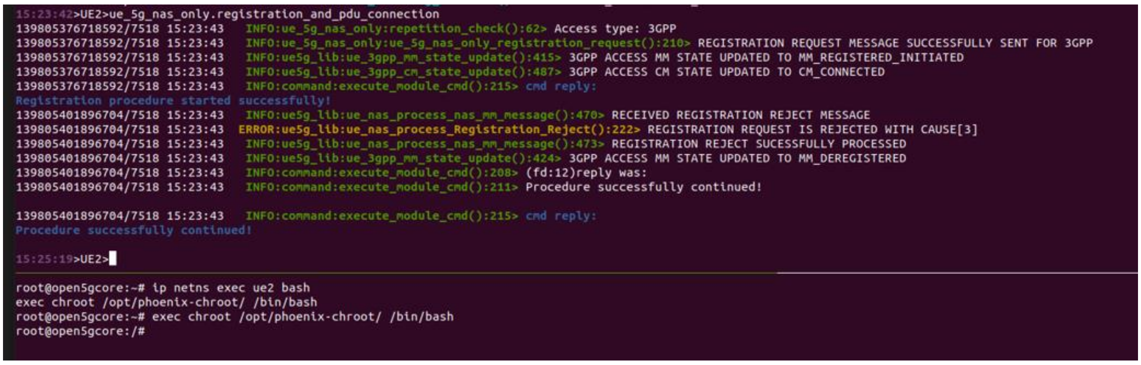
The setup needs to be reconfigured by the lab instructor before continuing or create an additional UE3. UE2 SUPI has changed, thus initial registration fails and causes error. See Appendix 7 and 8

Task 3: ue2 has changed, find out why.

1. Run under ue2 command initial registration with PDU.
2. Write down what happened, and does it succeed?
3. Declare that MM and CM state?
4. Find SUPI info and write it down. SUPI …………………….
5. What causes the error?
6. Why are only a few NFs are intervened?
7. Why is NF preholding SUPI info?

Step 1: Go to under "ue1" using TMUX command.

Step 2: Run command "ue_5g_nas_only.registration_and_pdu_connection". See Figure 33.

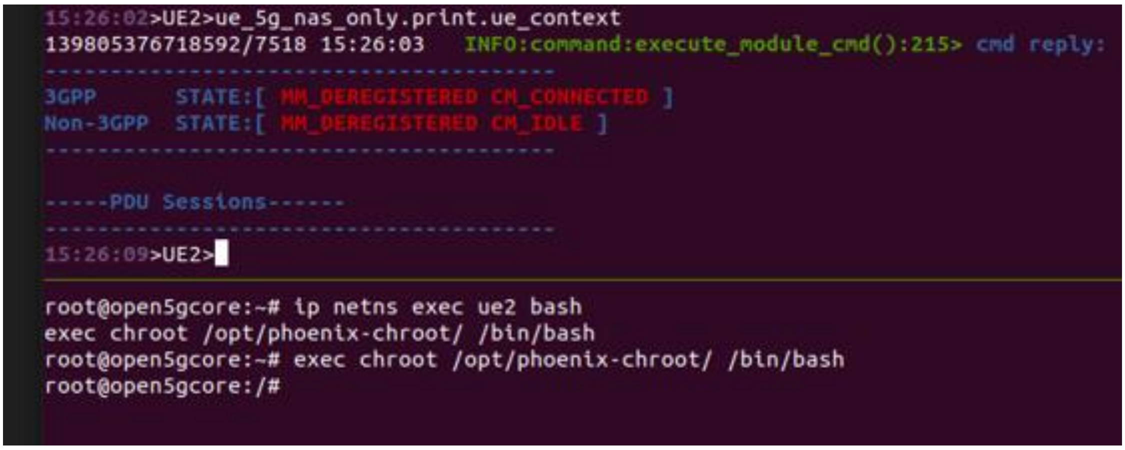


```
15:23:42>UE2>ue_5g_nas_only.registration_and_pdu_connection
139805376718592/7518 15:23:43   INFO:ue_5g_nas_only:repetition_check():62> Access type: 3GPP
139805376718592/7518 15:23:43   INFO:ue_5g_nas_only:ue_5g_nas_only_registration_request():210> REGISTRATION REQUEST MESSAGE SUCCESSFULLY SENT FOR 3GPP
139805376718592/7518 15:23:43   INFO:ue5g_lib:ue_3gpp_mm_state_update():415> 3GPP ACCESS MM STATE UPDATED TO MM_REGISTERED_INITIATED
139805376718592/7518 15:23:43   INFO:ue5g_lib:ue_3gpp_cm_state_update():487> 3GPP ACCESS CM STATE UPDATED TO CM_CONNECTED
139805376718592/7518 15:23:43   INFO:command:execute_module_cmd():215> cmd reply:
Registration procedure started successfully!
139805401896704/7518 15:23:43   INFO:ue5g_lib:ue_nas_process_nas_mm_message():470> RECEIVED REGISTRATION REJECT MESSAGE
139805401896704/7518 15:23:43  ERROR:ue5g_lib:ue_nas_process_Registration_Reject():222> REGISTRATION REQUEST IS REJECTED WITH CAUSE[3]
139805401896704/7518 15:23:43   INFO:ue5g_lib:ue_nas_process_nas_mm_message():473> REGISTRATION REJECT SUCESSFULLY PROCESSED
139805401896704/7518 15:23:43   INFO:ue5g_lib:ue_3gpp_mm_state_update():424> 3GPP ACCESS MM STATE UPDATED TO MM_DEREGISTERED
139805401896704/7518 15:23:43   INFO:command:execute_module_cmd():208> (fd:12)reply was:
139805401896704/7518 15:23:43   INFO:command:execute_module_cmd():211> Procedure successfully continued!

139805401896704/7518 15:23:43   INFO:command:execute_module_cmd():215> cmd reply:
Procedure successfully continued!

15:25:19>UE2>
```

```
root@open5gcore:~# ip netns exec ue2 bash
exec chroot /opt/phoenix-chroot/ /bin/bash
root@open5gcore:~# exec chroot /opt/phoenix-chroot/ /bin/bash
root@open5gcore:/#
```

Figure 33. UE2 registration request with PDU session.

Step 3: Make a status check running command: "ue_5g_nas_only.print.ue_context". See Figure 34



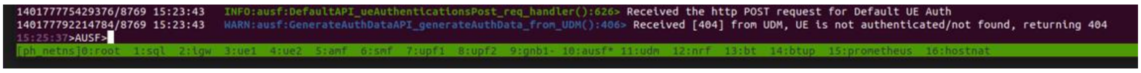
```
15:26:02>UE2>ue_5g_nas_only.print.ue_context
139805376718592/7518 15:26:03   INFO:command:execute_module_cmd():215> cmd reply:
---------------------------------------
3GPP      STATE:[ MM_DEREGISTERED CM_CONNECTED ]
Non-3GPP  STATE:[ MM_DEREGISTERED CM_IDLE ]
---------------------------------------

-----PDU Sessions------
---------------------------------------
15:26:09>UE2>

root@open5gcore:~# ip netns exec ue2 bash
exec chroot /opt/phoenix-chroot/ /bin/bash
root@open5gcore:~# exec chroot /opt/phoenix-chroot/ /bin/bash
root@open5gcore:/#
```

Figure 34. UE2 status check

Step 4: Use TMUX command **Ctrl + B** and **9** to see gNB commands.

Step 5: Use TMUX command **Ctrl + B** and **5** to see AMF commands.

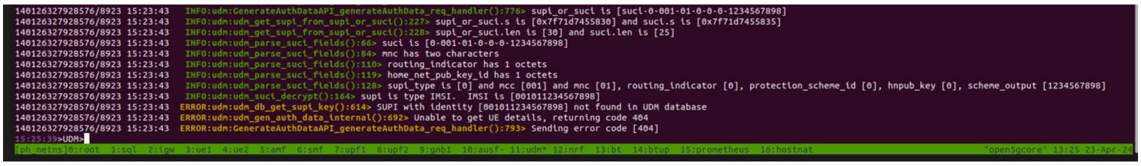
Step 6: Use TMUX command to go under section 10 AUSF**,** use command **Ctrl + B** and 9, then **Ctrl + B** and N to see AUSF command. See Figure 35.



```
140177775429376/8769 15:23:43   INFO:ausf:DefaultAPI_ueAuthenticationsPost_req_handler():626> Received the http POST request for Default UE Auth
140177792214784/8769 15:23:43   WARN:ausf:GenerateAuthDataAPI_generateAuthData_from_UDM():406> Received [404] from UDM, UE is not authenticated/not found, returning 404
15:25:37>AUSF>
[ph_netns]0:root  1:sql  2:igw  3:ue1  4:ue2  5:amf  6:smf  7:upf1  8:upf2  9:gnb1- 10:ausf* 11:udm  12:nrf  13:bt  14:btup  15:prometheus  16:hostnat
```

Figure 35. AUSF command view.

Step 7: Use TMUX command **Ctrl + B** and **N** to see UDM. See Figure 36.



```
140126327928576/8923 15:23:43   INFO:udm:GenerateAuthDataAPI_generateAuthData_req_handler():776> supi_or_suci is [suci-0-001-01-0-0-0-1234567898]
140126327928576/8923 15:23:43   INFO:udm:udm_get_supi_from_supi_or_suci():227> supi_or_suci.s is [0x7f71d7455830] and suci.s is [0x7f71d7455835]
140126327928576/8923 15:23:43   INFO:udm:udm_get_supi_from_supi_or_suci():228> supi_or_suci.len is [30] and suci.len is [25]
140126327928576/8923 15:23:43   INFO:udm:udm_parse_suci_fields():66> suci is [0-001-01-0-0-0-1234567898]
140126327928576/8923 15:23:43   INFO:udm:udm_parse_suci_fields():84> mnc has two characters
140126327928576/8923 15:23:43   INFO:udm:udm_parse_suci_fields():110> routing_indicator has 1 octets
140126327928576/8923 15:23:43   INFO:udm:udm_parse_suci_fields():119> home_net_pub_key_id has 1 octets
140126327928576/8923 15:23:43   INFO:udm:udm_parse_suci_fields():128> supi_type is [0] and mcc [001] and mnc [01], routing_indicator [0], protection_scheme_id [0], hnpub_key [0], scheme_output [1234567898]
140126327928576/8923 15:23:43   INFO:udm:udm_suci_decrypt():164> supi is type IMSI.  IMSI is [001011234567898]
140126327928576/8923 15:23:43  ERROR:udm:udm_db_get_supi_key():614> SUPI with identity [001011234567898] not found in UDM database
140126327928576/8923 15:23:43  ERROR:udm:udm_gen_auth_data_internal():692> Unable to get UE details, returning code 404
140126327928576/8923 15:23:43  ERROR:udm:GenerateAuthDataAPI_generateAuthData_req_handler():793> Sending error code [404]
15:25:39>UDM>
[ph_netns]0:root  1:sql  2:igw  3:ue1  4:ue2  5:amf  6:smf  7:upf1  8:upf2  9:gnb1  10:ausf- 11:udm* 12:nrf  13:bt  14:btup  15:prometheus  16:hostnat                    "open5gcore" 13:25 23-Apr-24
```

Figure 36. UDM command view.

Every function gives feedback error about what is responsible under initial registration.

-------------- *End of laboratory work instructions* --------------

This chapter has provided a laboratory work example related to UE registration to the 5G network with the O5GC simulator. The next chapter provides a short overview of problems encountered with the simulator.

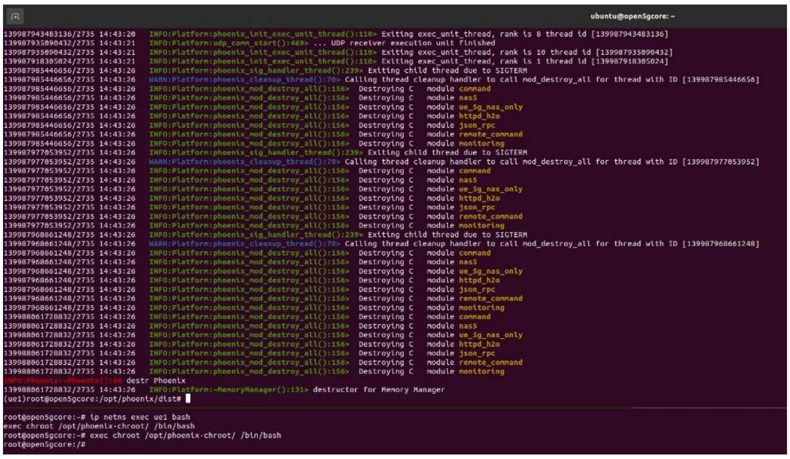
5. Problems Encountered with the Simulator During the Practical Part

This chapter provides a short overview of problems encountered with the simulator during the practical part. It can be called "lessons learned".

1. It must be mentioned that in Figure 14, PCF should be also red circled, but it is not used under the simulation because the configuration file doesn't include PCF and it is not needed when UE and gNB simulator is used.

2. Integrity error occurs and is shown in the terminal. But this error is not a problem to connect to network, and all procedures can be made. If PDU session is added, then integrity does not fail. The 3GPP standard requires to show this error. Normally, a UE always asks registration with PDU. But this does not affect registration itself and the registration can be completed. It is noted to TalTech supervisor.

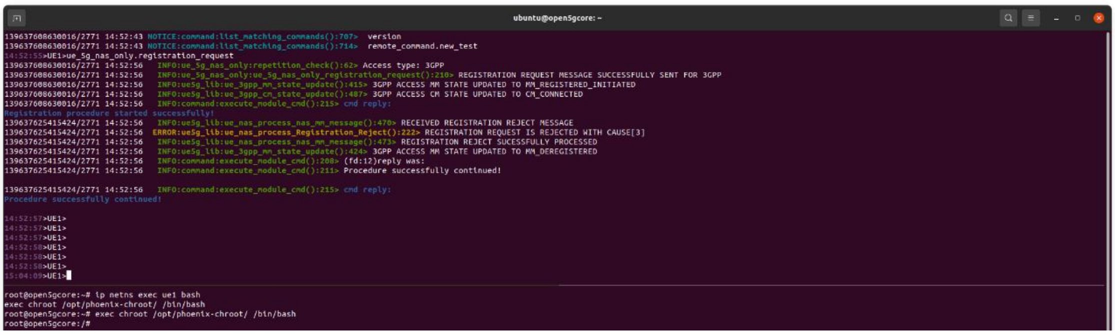
3. Wrong command is used, because TMUX works differently than typical Windows or other operating systems. For example, Ctrl + C (Figure 37) closes the simulator active function (contrary to many other systems where this is used to copy some text). If this command is typed, it is needed to restart the whole simulator to run the simulation correctly.



```
139987943483136/2735 14:43:20   INFO:Platform:phoenix_init_exec_unit_thread():110> Exiting exec_unit_thread, rank is 8 thread id [139987943483136]
139987935090432/2735 14:43:21   INFO:Platform:udp_comm_start():469> ... UDP receiver execution unit finished
139987935090432/2735 14:43:21   INFO:Platform:phoenix_init_exec_unit_thread():110> Exiting exec_unit_thread, rank is 10 thread id [139987935090432]
139987918305024/2735 14:43:21   INFO:Platform:phoenix_init_exec_unit_thread():110> Exiting exec_unit_thread, rank is 1 thread id [139987918305024]
139987985446656/2735 14:43:26   INFO:Platform:phoenix_sig_handler_thread():239> Exiting child thread due to SIGTERM
139987985446656/2735 14:43:26   WARN:Platform:phoenix_cleanup_thread():70> Calling thread cleanup handler to call mod_destroy_all for thread with ID [139987985446656]
139987985446656/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module command
139987985446656/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module nas5
139987985446656/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module ue_5g_nas_only
139987985446656/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module httpd_h2o
139987985446656/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module json_rpc
139987985446656/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module remote_command
139987985446656/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module monitoring
139987977053952/2735 14:43:26   INFO:Platform:phoenix_sig_handler_thread():239> Exiting child thread due to SIGTERM
139987977053952/2735 14:43:26   WARN:Platform:phoenix_cleanup_thread():70> Calling thread cleanup handler to call mod_destroy_all for thread with ID [139987977053952]
139987977053952/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module command
139987977053952/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module nas5
139987977053952/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module ue_5g_nas_only
139987977053952/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module httpd_h2o
139987977053952/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module json_rpc
139987977053952/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module remote_command
139987977053952/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module monitoring
139987968661248/2735 14:43:26   INFO:Platform:phoenix_sig_handler_thread():239> Exiting child thread due to SIGTERM
139987968661248/2735 14:43:26   WARN:Platform:phoenix_cleanup_thread():70> Calling thread cleanup handler to call mod_destroy_all for thread with ID [139987968661248]
139987968661248/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module command
139987968661248/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module nas5
139987968661248/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module ue_5g_nas_only
139987968661248/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module httpd_h2o
139987968661248/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module json_rpc
139987968661248/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module remote_command
139987968661248/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module monitoring
139988061728832/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module command
139988061728832/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module nas5
139988061728832/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module ue_5g_nas_only
139988061728832/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module httpd_h2o
139988061728832/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module json_rpc
139988061728832/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module remote_command
139988061728832/2735 14:43:26   INFO:Platform:phoenix_mod_destroy_all():156>  Destroying C   module monitoring
INFO:Phoenix:~Phoenix():40 destr Phoenix
139988061728832/2735 14:43:26   INFO:Platform:~MemoryManager():131> destructor for Memory Manager
(ue1)root@open5gcore:/opt/phoenix/dist# 

root@open5gcore:~# ip netns exec ue1 bash
exec chroot /opt/phoenix-chroot/ /bin/bash
root@open5gcore:~# exec chroot /opt/phoenix-chroot/ /bin/bash
root@open5gcore:/#
```

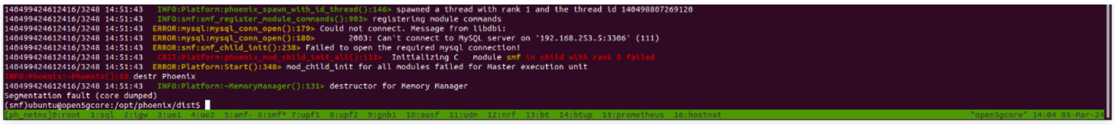
Figure 37. Ctrl+C shortcut key pressed, it closed the simulator active function.

4. Installing Wireshark under the virtual machine generates an error to the simulator (Figure 38), and it does not work anymore. Two different programs can crash the simulator, because the simulator is actively using files that are in the folder. Wrong files can crash the simulator (Figure 39). Therefore, do not install any other programs under the virtual machine.



```
139637608630016/2771 14:52:43 NOTICE:command:list_matching_commands():707>  version
139637608630016/2771 14:52:43 NOTICE:command:list_matching_commands():714>  remote_command.new_test
14:52:55>UE1>ue_5g_nas_only.registration_request
139637608630016/2771 14:52:56   INFO:ue_5g_nas_only:repetition_check():62> Access type: 3GPP
139637608630016/2771 14:52:56   INFO:ue_5g_nas_only:ue_5g_nas_only_registration_request():210> REGISTRATION REQUEST MESSAGE SUCCESSFULLY SENT FOR 3GPP
139637608630016/2771 14:52:56   INFO:ue5g_lib:ue_3gpp_mm_state_update():415> 3GPP ACCESS MM STATE UPDATED TO MM_REGISTERED_INITIATED
139637608630016/2771 14:52:56   INFO:ue5g_lib:ue_3gpp_cm_state_update():487> 3GPP ACCESS CM STATE UPDATED TO CM_CONNECTED
139637608630016/2771 14:52:56   INFO:command:execute_module_cmd():215> cmd reply:
Registration procedure started successfully!
139637625415424/2771 14:52:56   INFO:ue5g_lib:ue_nas_process_nas_mm_message():470> RECEIVED REGISTRATION REJECT MESSAGE
139637625415424/2771 14:52:56  ERROR:ue5g_lib:ue_nas_process_Registration_Reject():222> REGISTRATION REQUEST IS REJECTED WITH CAUSE[3]
139637625415424/2771 14:52:56   INFO:ue5g_lib:ue_nas_process_nas_mm_message():473> REGISTRATION REJECT SUCESSFULLY PROCESSED
139637625415424/2771 14:52:56   INFO:ue5g_lib:ue_3gpp_mm_state_update():424> 3GPP ACCESS MM STATE UPDATED TO MM_DEREGISTERED
139637625415424/2771 14:52:56   INFO:command:execute_module_cmd():208> (fd:12)reply was:
139637625415424/2771 14:52:56   INFO:command:execute_module_cmd():211> Procedure successfully continued!

139637625415424/2771 14:52:56   INFO:command:execute_module_cmd():215> cmd reply:
Procedure successfully continued!

14:52:57>UE1>
14:52:57>UE1>
14:52:57>UE1>
14:52:58>UE1>
14:52:58>UE1>
14:52:58>UE1>
15:04:09>UE1>

root@open5gcore:~# ip netns exec ue1 bash
exec chroot /opt/phoenix-chroot/ /bin/bash
root@open5gcore:~# exec chroot /opt/phoenix-chroot/ /bin/bash
root@open5gcore:/#
```

Figure 38. Error caused installing Wireshark under VirtualBox

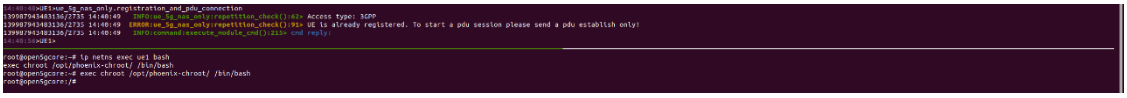


```
140499424612416/3248 14:51:43   INFO:Platform:phoenix_spawn_with_id_thread():146> spawned a thread with rank 1 and the thread id 140498807269120
140499424612416/3248 14:51:43   INFO:smf:smf_register_module_commands():903> registering module commands
140499424612416/3248 14:51:43  ERROR:mysql:mysql_conn_open():179> Could not connect. Message from libdbi:
140499424612416/3248 14:51:43  ERROR:mysql:mysql_conn_open():180>       2003: Can't connect to MySQL server on '192.168.253.5:3306' (111)
140499424612416/3248 14:51:43  ERROR:smf:smf_child_init():238> Failed to open the required mysql connection!
140499424612416/3248 14:51:43   CRIT:Platform:phoenix_mod_child_init_all():133>  Initializing C   module smf in child with rank 0 failed
140499424612416/3248 14:51:43  ERROR:Platform:Start():348> mod_child_init for all modules failed for Master execution unit
INFO:Phoenix:~Phoenix():60 destr Phoenix
140499424612416/3248 14:51:43   INFO:Platform:~MemoryManager():131> destructor for Memory Manager
Segmentation fault (core dumped)
(smf)ubuntu@open5gcore:/opt/phoenix/dist$
[ph_netns]0:root  1:sql  2:igw  3:ue1  4:ue2  5:amf- 6:smf* 7:upf1  8:upf2  9:gnb1  10:ausf  11:udm  12:nrf  13:bt  14:btup  15:prometheus  16:hostnat          "open5gcore" 14:04 05-Mar-24
```

Figure 39. Error shown under SMF, mysql database lost.

5. Can not registered twice. UE1 was initially registered only to the network.

Author tried to make PDU establishment but used wrong command. UE1 is already registered (Figure 40).



```
14:40:48>UE1>ue_5g_nas_only.registration_and_pdu_connection
139987943483136/2735 14:40:49   INFO:ue_5g_nas_only:repetition_check():62> Access type: 3GPP
139987943483136/2735 14:40:49  ERROR:ue_5g_nas_only:repetition_check():91> UE is already registered. To start a pdu session please send a pdu establish only!
139987943483136/2735 14:40:49   INFO:command:execute_module_cmd():215> cmd reply:
14:40:56>UE1>

root@open5gcore:~# ip netns exec ue1 bash
exec chroot /opt/phoenix-chroot/ /bin/bash
root@open5gcore:~# exec chroot /opt/phoenix-chroot/ /bin/bash
root@open5gcore:/#
```

Figure 40. Wrong command used

6. Conclusion

6.1 Summary

This thesis aimed to alleviate the challenges of 1) filling the need for understanding and documenting the 5G system, and 2) develop new practical laboratory exercise for building the knowledge of learners and develop their practical skills.

For doing so, the thesis has investigated open source 5GC software solutions, the relevant parts of the 3GPP 5G standards, and developed a practical laboratory work example. The thesis started by providing an overview of related works and continued with an overview of the gained knowledge of the relevant parts of the 3GPP standard, of the O5GC simulator and in particular the “Call flow” and the gained know-how for connecting a UE to the network. Then the laboratory work has been developed and documented in detail. In addition, typical problems and errors encountered during the practical part of the work have been provided.

As a result, the O5GC simulator was installed, configured, got to run, and was able to execute commands; the UE got registered to the 5G network. The work also revelated that the O5GC simulator shows in an understandable way how different features actually execute in the network core and that the results were comparable with the 3GPP 5G standard. Even if all the work was conducted under terminal commands, it was still possible to run through the procedures and the result was positive, i.e. the O5GC simulator works as expected as per the 5G standards. Students and learners can analyse the processes and results, which can give a much more concrete understanding of how 5G works in practice and makes it understandable how the 5G standard is built up. Without such a simulator, it is quite hard for students and learners to understand what is described in the standard [29]. Based on the achieved understanding of the 5G system and the developed laboratory work, it is concluded that the main goal of this thesis has been achieved.

6.2 A few perspectives

The successful investigation achieved in this thesis paves the way for many more additional learning opportunities and the development of a full course. Two main possible topics to be further explored and developed include, but are not limited to:

1. Experimenting and validating other procedures than the registration. For example, network slice selection, session establishment, bearer establishment, data transmission, etc. These steps are part of the overall process to ensure that the UE can securely and efficiently use the 5G network services after registration. The exact sequence and specifics vary depending on the network configuration and the UE's capabilities, meaning that many different laboratory work variations could be developed.

2. Integrating and using real devices. O5GC could be used in real mode rather than simulation mode, together with real RAN and UE devices, possibly implemented in conjunction with SDR boards. This would require extensive preparatory work but would also provide unvaluable knowledge and practical experience to the students and learners.

3. It is possible to install O5GC.apk under UE for example mobile phone to make registration to network over Wi-Fi. It is possible to use non-3GPP access point for connection. It needs to Wi-Fi configuration and application configuration.

References

[1] TSGS, “TS 133 501 - V17.13.0 - 5G; Security architecture and procedures for 5G System (3GPP TS 33.501 version 17.13.0 Release 17),” 2024, Accessed: Apr. 30, 2024. [Online]. Available: https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

[2] TSGS, “TS 123 501 - V17.12.0 - 5G; System architecture for the 5G System (5GS) (3GPP TS 23.501 version 17.12.0 Release 17),” 2024, Accessed: Apr. 29, 2024. [Online]. Available: https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

[3] “IoT: How 5G differs from LTE - 5G Technology World.” Accessed: May 06, 2024. [Online]. Available: https://www.5gtechnologyworld.com/iot-how-5g-differs-from-lte/

[4] “Mobile networks’ evolution from 1G to 5G.” Accessed: May 06, 2024. [Online]. Available: https://www.linkedin.com/pulse/mobile-networks-evolution-from-1g-5g-ct-rf-antennas-inc

[5] “Open5GCore | Open5GCore.” Accessed: Apr. 30, 2024. [Online]. Available: https://www.open5gcore.org/

[6] TSGS, “TS 123 502 - V17.12.0 - 5G; Procedures for the 5G System (5GS) (3GPP TS 23.502 version 17.12.0 Release 17),” 2024, Accessed: Apr. 29, 2024. [Online]. Available: https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

[7] K. Mubasier, F. Y. Li, J. A. S. Ogaard, and M. C. Vochin, “Campus-Based Full-Scale and Portable Open-Source 5G SA Networks: Prototyping and Experiments,” *International Symposium on Wireless Personal Multimedia Communications, WPMC*, pp. 70–75, 2023, doi: 10.1109/WPMC59531.2023.10338948.

[8] G. Lando, L. A. F. Schierholt, M. P. Milesi, and J. A. Wickboldt, “Evaluating the performance of open source software implementations of the 5G network core,” *Proceedings of IEEE/IFIP Network Operations and Management Symposium 2023, NOMS 2023*, 2023, doi: 10.1109/NOMS56928.2023.10154399.

[9] M. Chepkoech, N. Mombeshora, B. Malila, and J. Mwangama, “Evaluation of Open-Source Mobile Network Software Stacks: A Guide to Low-cost Deployment of 5G Testbeds,” *2023 18th Wireless On-Demand Network Systems and Services Conference, WONS 2023*, pp. 56–63, 2023, doi: 10.23919/WONS57325.2023.10061896.

[10] M. Chepkoech, E. R. Modroiu, J. Mwangama, M. Corici, and T. Magedanz, "Evaluation of OSS-Enabled OpenRAN Compliant 5G StandAlone Campus Networks," *International Conference on Electrical, Computer and Energy Technologies, ICECET 2023*, 2023, doi: 10.1109/ICECET58911.2023.10389574.

[11] L. P. Sanchez, J. R. Ramírez, F. S. Sulca, and D. M. Avilés, "Exploring the Potential of a 5G NSA Network Deployed with Open Source and USRP B210 Equipment," *ECTM 2023 - 2023 IEEE 7th Ecuador Technical Chapters Meeting*, 2023, doi: 10.1109/ETCM58927.2023.10309101.

[12] D. Pineda, R. Harrilal-Parchment, K. Akkaya, A. Ibrahim, and A. Perez-Pons, "Design and Analysis of an Open-Source SDN-based 5G Standalone Testbed," *IEEE INFOCOM 2023 - Conference on Computer Communications Workshops, INFOCOM WKSHPS 2023*, 2023, doi: 10.1109/INFOCOMWKSHPS57453.2023.10225862.

[13] J. E. Hakegard, H. Lundkvist, A. Rauniyar, and P. Morris, "Performance Evaluation of an Open Source Implementation of a 5G Standalone Platform," *IEEE Access*, vol. 12, pp. 25809–25819, 2024, doi: 10.1109/ACCESS.2024.3367120.

[14] L. A. Phan, D. Pesch, U. Roedig, and C. J. Sreenan, "Building a 5G Core Network Testbed: Open-Source Solutions, Lessons Learned, and Research Directions," in *Proc of the 38th International Conference on Information Networking (ICOIN 2024), 17-19 January 2024, Ho Chi Minh City, Vietnam*, 2024.

[15] "Open Networking Foundation," "Aether: Private 5G Platform."

[16] R. Reddy, M. Gundall, C. Lipps, and H. D. Schotten, "Open Source 5G Core Network Implementations: A Qualitative and Quantitative Analysis," *2023 IEEE International Black Sea Conference on Communications and Networking, BlackSeaCom 2023*, pp. 253–258, 2023, doi: 10.1109/BLACKSEACOM58138.2023.10299755.

[17] Linh-An Phan, "'5G LAN Testbed GitHub Repository,'" https://github.com/linhanphan/5glan (accessed April 28, 2024).

[18] Sukchan Lee, "Open5GS testbed homepage." Accessed: May 01, 2024. [Online]. Available: https://open5gs.org/

[19] "5G CORE NETWORK – OpenAirInterface." Accessed: May 01, 2024. [Online]. Available: https://openairinterface.org/oai-5g-core-network-project/

[20] "free5GC." Accessed: May 01, 2024. [Online]. Available: https://free5gc.org/

[21] “OAIBOX: The Ultimate Open Source 5G Platform for Academic and Industrial Research.” Accessed: Apr. 30, 2024. [Online]. Available: https://www.oaibox.com/

[22] “About ITU.” Accessed: Apr. 30, 2024. [Online]. Available: https://www.itu.int/en/about/Pages/default.aspx

[23] TSGS, “TS 123 502 - V15.17.0 - 5G; Procedures for the 5G System (5GS) (3GPP TS 23.502 version 15.17.0 Release 15),” 2023, Accessed: Apr. 29, 2024. [Online]. Available: https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

[24] “Introducing 3GPP.” Accessed: May 01, 2024. [Online]. Available: https://www.3gpp.org/about-us/introducing-3gpp

[25] “3GPP Release.” Accessed: May 06, 2024. [Online]. Available: https://devopedia.org/3gpp-release

[26] J. A. Brito, J. I. Moreno, L. M. Contreras, M. Alvarez-Campana, and M. Blanco Caamaño, “Programmable Data Plane Applications in 5G and Beyond Architectures: A Systematic Review,” *Sensors 2023, Vol. 23, Page 6955*, vol. 23, no. 15, p. 6955, Aug. 2023, doi: 10.3390/S23156955.

[27] “5G Identifiers SUPI, SUCI, GUTI, GPSI, PEI , AMF, DNN - TELCOMA.” Accessed: May 03, 2024. [Online]. Available: https://telcomaglobal.com/p/5g-identifiers

[28] TSGS, “TS 133 501 - V15.2.0 - 5G; Security architecture and procedures for 5G System (3GPP TS 33.501 version 15.2.0 Release 15),” 2018, Accessed: May 03, 2024. [Online]. Available: https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

[29] K. Du, X. Wen, L. Wang, and T. T. Nguyen, “A Cloud-Native Based Access and Mobility Management Function Implementation in 5G Core,” *2020 IEEE 6th International Conference on Computer and Communications, ICCC 2020*, pp. 1251–1256, Dec. 2020, doi: 10.1109/ICCC51575.2020.9345262.

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis[4]

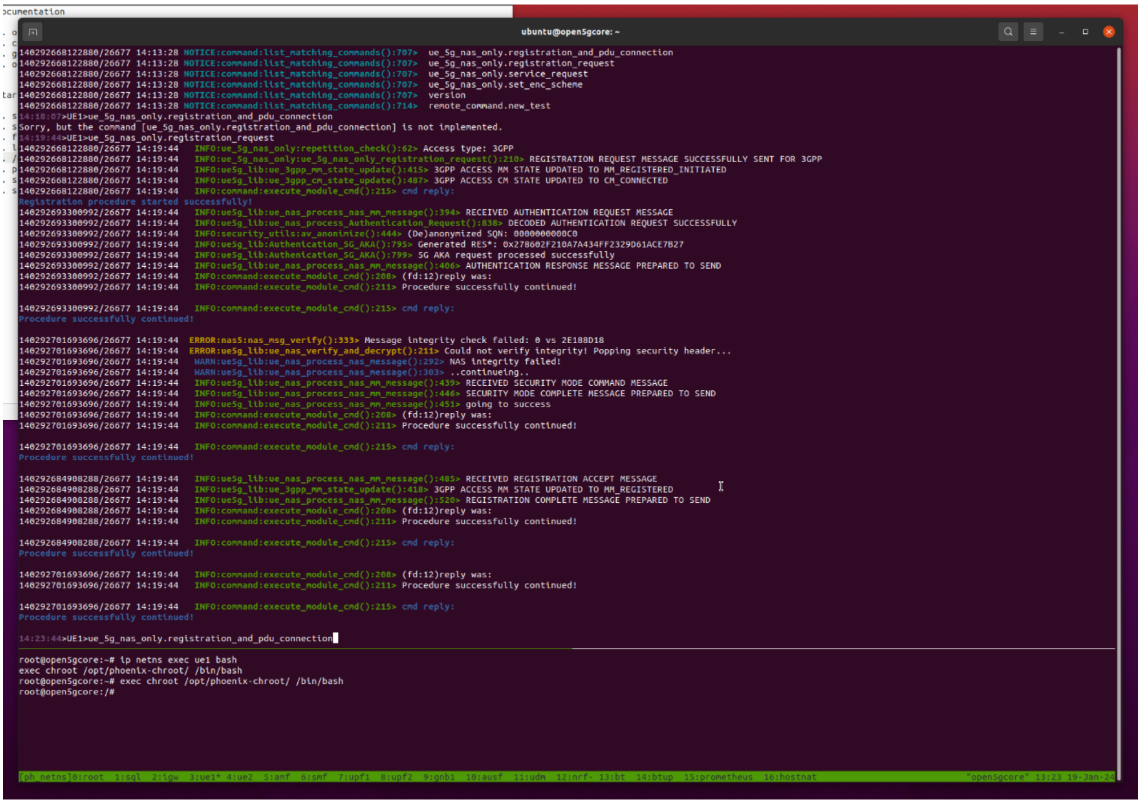
I Taivo Miller

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis “OPEN SOURCE OPEN5GCORE (O5GC) PLATFORM BASED TRAINING GROUND”, supervised by Margus Rohtla and Yannick Le Moullec
    1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
    1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

06.05.2024

[4] The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

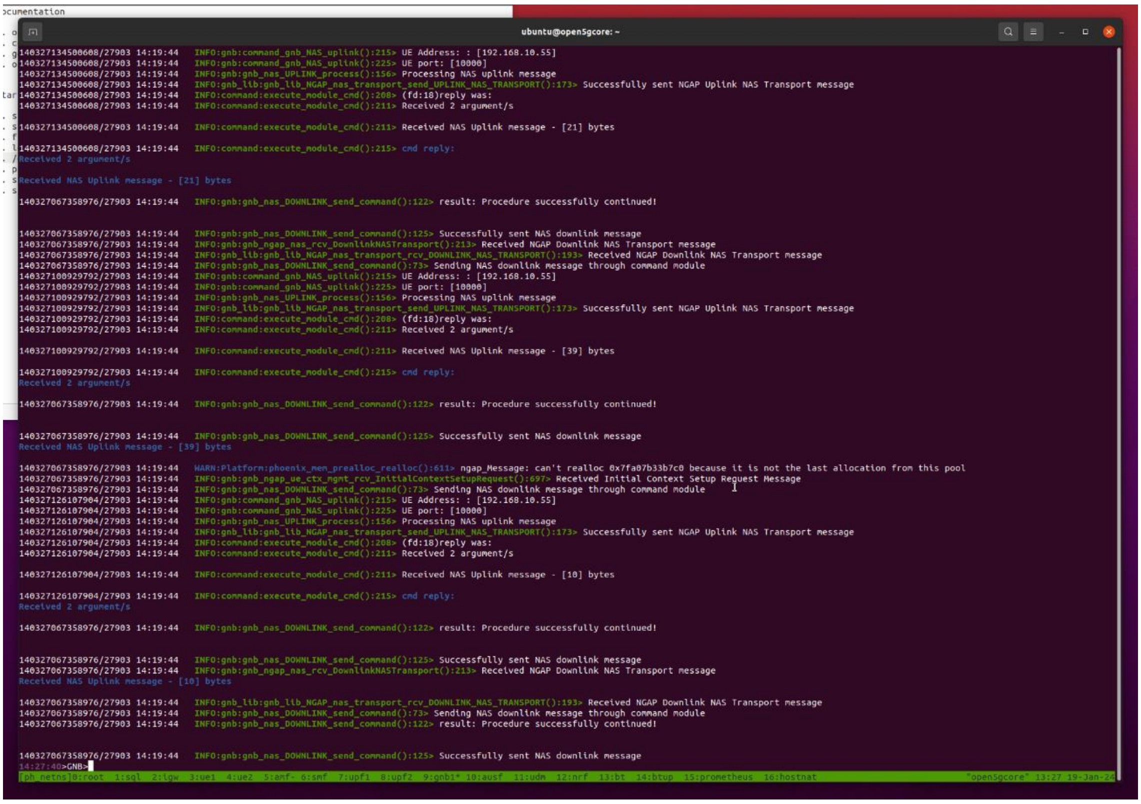
Appendix 2 – UE1 info



```
140292668122880/26677 14:13:28 NOTICE:command:list_matching_commands():707>  ue_5g_nas_only.registration_and_pdu_connection
140292668122880/26677 14:13:28 NOTICE:command:list_matching_commands():707>  ue_5g_nas_only.registration_request
140292668122880/26677 14:13:28 NOTICE:command:list_matching_commands():707>  ue_5g_nas_only.service_request
140292668122880/26677 14:13:28 NOTICE:command:list_matching_commands():707>  ue_5g_nas_only.set_enc_scheme
140292668122880/26677 14:13:28 NOTICE:command:list_matching_commands():707>  version
140292668122880/26677 14:13:28 NOTICE:command:list_matching_commands():714>  remote_command.new_test
14:18:07>UE1>ue_5g_nas_only.registration_and_pdu_connection
Sorry, but the command [ue_5g_nas_only.registration_and_pdu_connection] is not implemented.
14:19:44>UE1>ue_5g_nas_only.registration_request
140292668122880/26677 14:19:44   INFO:ue_5g_nas_only:repetition_check():62> Access type: 3GPP
140292668122880/26677 14:19:44   INFO:ue_5g_nas_only:ue_5g_nas_only_registration_request():210> REGISTRATION REQUEST MESSAGE SUCCESSFULLY SENT FOR 3GPP
140292668122880/26677 14:19:44   INFO:ue5g_lib:ue_3gpp_mm_state_update():415> 3GPP ACCESS MM STATE UPDATED TO MM_REGISTERED_INITIATED
140292668122880/26677 14:19:44   INFO:ue5g_lib:ue_3gpp_cm_state_update():487> 3GPP ACCESS CM STATE UPDATED TO CM_CONNECTED
140292668122880/26677 14:19:44   INFO:command:execute_module_cmd():215> cmd reply:
Registration procedure started successfully!
140292693300992/26677 14:19:44   INFO:ue5g_lib:ue_nas_process_nas_mm_message():394> RECEIVED AUTHENTICATION REQUEST MESSAGE
140292693300992/26677 14:19:44   INFO:ue5g_lib:ue_nas_process_Authentication_Request():838> DECODED AUTHENTICATION REQUEST SUCCESSFULLY
140292693300992/26677 14:19:44   INFO:security_utils:av_anonimize():444> (De)anonymized SQN: 0000000000C0
140292693300992/26677 14:19:44   INFO:ue5g_lib:Authenication_5G_AKA():795> Generated RES*: 0x278602F210A7A434FF2329D61ACE7B27
140292693300992/26677 14:19:44   INFO:ue5g_lib:Authenication_5G_AKA():799> 5G AKA request processed successfully
140292693300992/26677 14:19:44   INFO:ue5g_lib:ue_nas_process_nas_mm_message():406> AUTHENTICATION RESPONSE MESSAGE PREPARED TO SEND
140292693300992/26677 14:19:44   INFO:command:execute_module_cmd():208> (fd:12)reply was:
140292693300992/26677 14:19:44   INFO:command:execute_module_cmd():211> Procedure successfully continued!

140292693300992/26677 14:19:44   INFO:command:execute_module_cmd():215> cmd reply:
Procedure successfully continued!

140292701693696/26677 14:19:44  ERROR:nas5:nas_msg_verify():333> Message integrity check failed: 0 vs 2E188D18
140292701693696/26677 14:19:44  ERROR:ue5g_lib:ue_nas_verify_and_decrypt():211> Could not verify integrity! Popping security header...
140292701693696/26677 14:19:44   WARN:ue5g_lib:ue_nas_process_nas_message():292> NAS integrity failed!
140292701693696/26677 14:19:44   WARN:ue5g_lib:ue_nas_process_nas_message():303> ..continueing..
140292701693696/26677 14:19:44   INFO:ue5g_lib:ue_nas_process_nas_mm_message():439> RECEIVED SECURITY MODE COMMAND MESSAGE
140292701693696/26677 14:19:44   INFO:ue5g_lib:ue_nas_process_nas_mm_message():446> SECURITY MODE COMPLETE MESSAGE PREPARED TO SEND
140292701693696/26677 14:19:44   INFO:ue5g_lib:ue_nas_process_nas_mm_message():451> going to success
140292701693696/26677 14:19:44   INFO:command:execute_module_cmd():208> (fd:12)reply was:
140292701693696/26677 14:19:44   INFO:command:execute_module_cmd():211> Procedure successfully continued!

140292701693696/26677 14:19:44   INFO:command:execute_module_cmd():215> cmd reply:
Procedure successfully continued!

140292684908288/26677 14:19:44   INFO:ue5g_lib:ue_nas_process_nas_mm_message():485> RECEIVED REGISTRATION ACCEPT MESSAGE
140292684908288/26677 14:19:44   INFO:ue5g_lib:ue_3gpp_mm_state_update():418> 3GPP ACCESS MM STATE UPDATED TO MM_REGISTERED
140292684908288/26677 14:19:44   INFO:ue5g_lib:ue_nas_process_nas_mm_message():520> REGISTRATION COMPLETE MESSAGE PREPARED TO SEND
140292684908288/26677 14:19:44   INFO:command:execute_module_cmd():208> (fd:12)reply was:
140292684908288/26677 14:19:44   INFO:command:execute_module_cmd():211> Procedure successfully continued!

140292684908288/26677 14:19:44   INFO:command:execute_module_cmd():215> cmd reply:
Procedure successfully continued!

140292701693696/26677 14:19:44   INFO:command:execute_module_cmd():208> (fd:12)reply was:
140292701693696/26677 14:19:44   INFO:command:execute_module_cmd():211> Procedure successfully continued!

140292701693696/26677 14:19:44   INFO:command:execute_module_cmd():215> cmd reply:
Procedure successfully continued!

14:23:44>UE1>ue_5g_nas_only.registration_and_pdu_connection

root@open5gcore:~# ip netns exec ue1 bash
exec chroot /opt/phoenix-chroot/ /bin/bash
root@open5gcore:~# exec chroot /opt/phoenix-chroot/ /bin/bash
root@open5gcore:/#

[ph_netns]0:root  1:sql  2:igw  3:ue1* 4:ue2  5:amf  6:smf  7:upf1  8:upf2  9:gnb1  10:ausf  11:udm  12:nrf- 13:bt  14:btup  15:prometheus  16:hostnat        "open5gcore" 13:23 19-Jan-24
```

Figure 41. UE1 initial registration overview

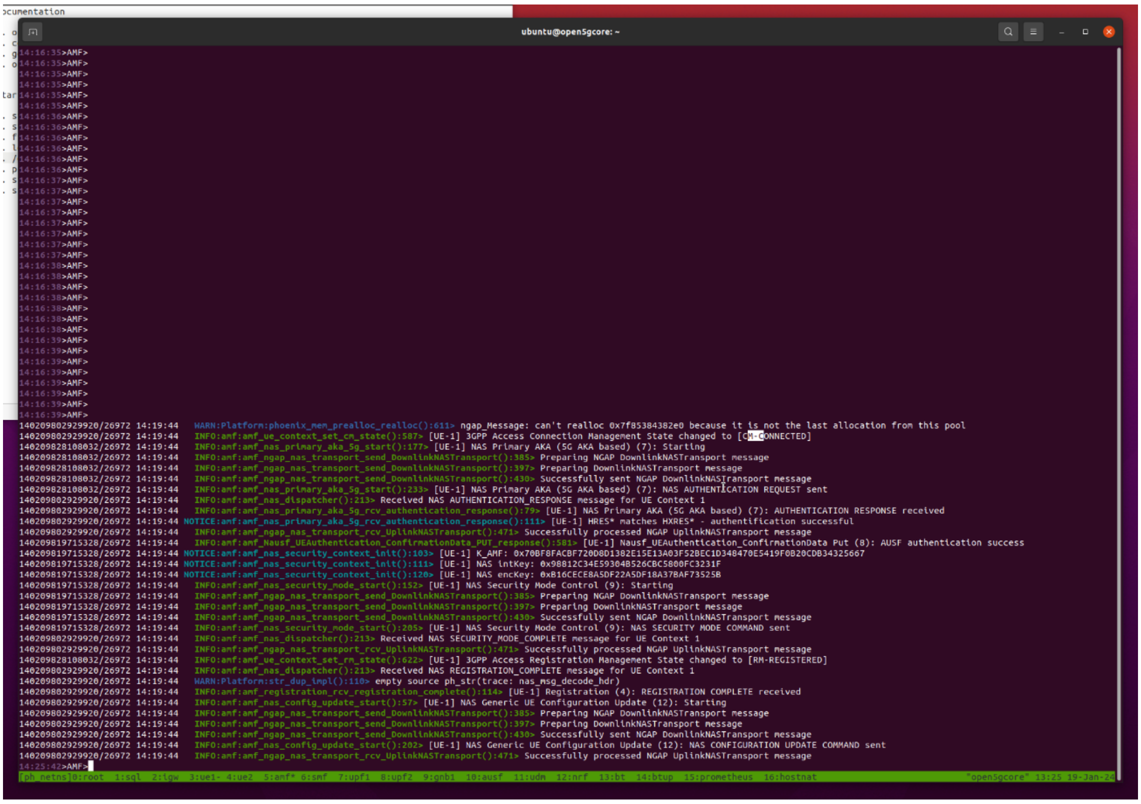
Appendix 3 – gNB info



```
140327134500608/27903 14:19:44   INFO:gnb:command_gnb_NAS_uplink():215> UE Address: : [192.168.10.55]
140327134500608/27903 14:19:44   INFO:gnb:command_gnb_NAS_uplink():225> UE port: [10000]
140327134500608/27903 14:19:44   INFO:gnb:gnb_nas_UPLINK_process():156> Processing NAS uplink message
140327134500608/27903 14:19:44   INFO:gnb_lib:gnb_lib_NGAP_nas_transport_send_UPLINK_NAS_TRANSPORT():173> Successfully sent NGAP Uplink NAS Transport message
140327134500608/27903 14:19:44   INFO:command:execute_module_cmd():208> (fd:18)reply was:
140327134500608/27903 14:19:44   INFO:command:execute_module_cmd():211> Received 2 argument/s

140327134500608/27903 14:19:44   INFO:command:execute_module_cmd():211> Received NAS Uplink message - [21] bytes

140327134500608/27903 14:19:44   INFO:command:execute_module_cmd():215> cmd reply:
Received 2 argument/s

Received NAS Uplink message - [21] bytes

140327067358976/27903 14:19:44   INFO:gnb:gnb_nas_DOWNLINK_send_command():122> result: Procedure successfully continued!


140327067358976/27903 14:19:44   INFO:gnb:gnb_nas_DOWNLINK_send_command():125> Successfully sent NAS downlink message
140327067358976/27903 14:19:44   INFO:gnb:gnb_ngap_nas_rcv_DownlinkNASTransport():213> Received NGAP Downlink NAS Transport message
140327067358976/27903 14:19:44   INFO:gnb_lib:gnb_lib_NGAP_nas_transport_rcv_DOWNLINK_NAS_TRANSPORT():193> Received NGAP Downlink NAS Transport message
140327067358976/27903 14:19:44   INFO:gnb:gnb_nas_DOWNLINK_send_command():73> Sending NAS downlink message through command module
140327100929792/27903 14:19:44   INFO:gnb:command_gnb_NAS_uplink():215> UE Address: : [192.168.10.55]
140327100929792/27903 14:19:44   INFO:gnb:command_gnb_NAS_uplink():225> UE port: [10000]
140327100929792/27903 14:19:44   INFO:gnb:gnb_nas_UPLINK_process():156> Processing NAS uplink message
140327100929792/27903 14:19:44   INFO:gnb_lib:gnb_lib_NGAP_nas_transport_send_UPLINK_NAS_TRANSPORT():173> Successfully sent NGAP Uplink NAS Transport message
140327100929792/27903 14:19:44   INFO:command:execute_module_cmd():208> (fd:18)reply was:
140327100929792/27903 14:19:44   INFO:command:execute_module_cmd():211> Received 2 argument/s

140327100929792/27903 14:19:44   INFO:command:execute_module_cmd():211> Received NAS Uplink message - [39] bytes

140327100929792/27903 14:19:44   INFO:command:execute_module_cmd():215> cmd reply:
Received 2 argument/s

140327067358976/27903 14:19:44   INFO:gnb:gnb_nas_DOWNLINK_send_command():122> result: Procedure successfully continued!


140327067358976/27903 14:19:44   INFO:gnb:gnb_nas_DOWNLINK_send_command():125> Successfully sent NAS downlink message
Received NAS Uplink message - [39] bytes

140327067358976/27903 14:19:44   WARN:Platform:phoenix_mem_prealloc_realloc():611> ngap_Message: can't realloc 0x7fa07b33b7c0 because it is not the last allocation from this pool
140327067358976/27903 14:19:44   INFO:gnb:gnb_ngap_ue_ctx_mgmt_rcv_InitialContextSetupRequest():697> Received Initial Context Setup Request Message
140327067358976/27903 14:19:44   INFO:gnb:gnb_nas_DOWNLINK_send_command():73> Sending NAS downlink message through command module
140327126107904/27903 14:19:44   INFO:gnb:command_gnb_NAS_uplink():215> UE Address: : [192.168.10.55]
140327126107904/27903 14:19:44   INFO:gnb:command_gnb_NAS_uplink():225> UE port: [10000]
140327126107904/27903 14:19:44   INFO:gnb:gnb_nas_UPLINK_process():156> Processing NAS uplink message
140327126107904/27903 14:19:44   INFO:gnb_lib:gnb_lib_NGAP_nas_transport_send_UPLINK_NAS_TRANSPORT():173> Successfully sent NGAP Uplink NAS Transport message
140327126107904/27903 14:19:44   INFO:command:execute_module_cmd():208> (fd:18)reply was:
140327126107904/27903 14:19:44   INFO:command:execute_module_cmd():211> Received 2 argument/s

140327126107904/27903 14:19:44   INFO:command:execute_module_cmd():211> Received NAS Uplink message - [10] bytes

140327126107904/27903 14:19:44   INFO:command:execute_module_cmd():215> cmd reply:
Received 2 argument/s

140327067358976/27903 14:19:44   INFO:gnb:gnb_nas_DOWNLINK_send_command():122> result: Procedure successfully continued!


140327067358976/27903 14:19:44   INFO:gnb:gnb_nas_DOWNLINK_send_command():125> Successfully sent NAS downlink message
140327067358976/27903 14:19:44   INFO:gnb:gnb_ngap_nas_rcv_DownlinkNASTransport():213> Received NGAP Downlink NAS Transport message
Received NAS Uplink message - [10] bytes

140327067358976/27903 14:19:44   INFO:gnb_lib:gnb_lib_NGAP_nas_transport_rcv_DOWNLINK_NAS_TRANSPORT():193> Received NGAP Downlink NAS Transport message
140327067358976/27903 14:19:44   INFO:gnb:gnb_nas_DOWNLINK_send_command():73> Sending NAS downlink message through command module
140327067358976/27903 14:19:44   INFO:gnb:gnb_nas_DOWNLINK_send_command():122> result: Procedure successfully continued!


140327067358976/27903 14:19:44   INFO:gnb:gnb_nas_DOWNLINK_send_command():125> Successfully sent NAS downlink message
14:27:40>GNB>
[ph_netns]0:root  1:sql  2:igw  3:ue1  4:ue2  5:amf- 6:smf  7:upf1  8:upf2  9:gnb1* 10:ausf  11:udm  12:nrf  13:bt  14:btup  15:prometheus  16:hostnat          "open5gcore" 13:27 19-Jan-24
```

Figure 42. GNB initial registration overview

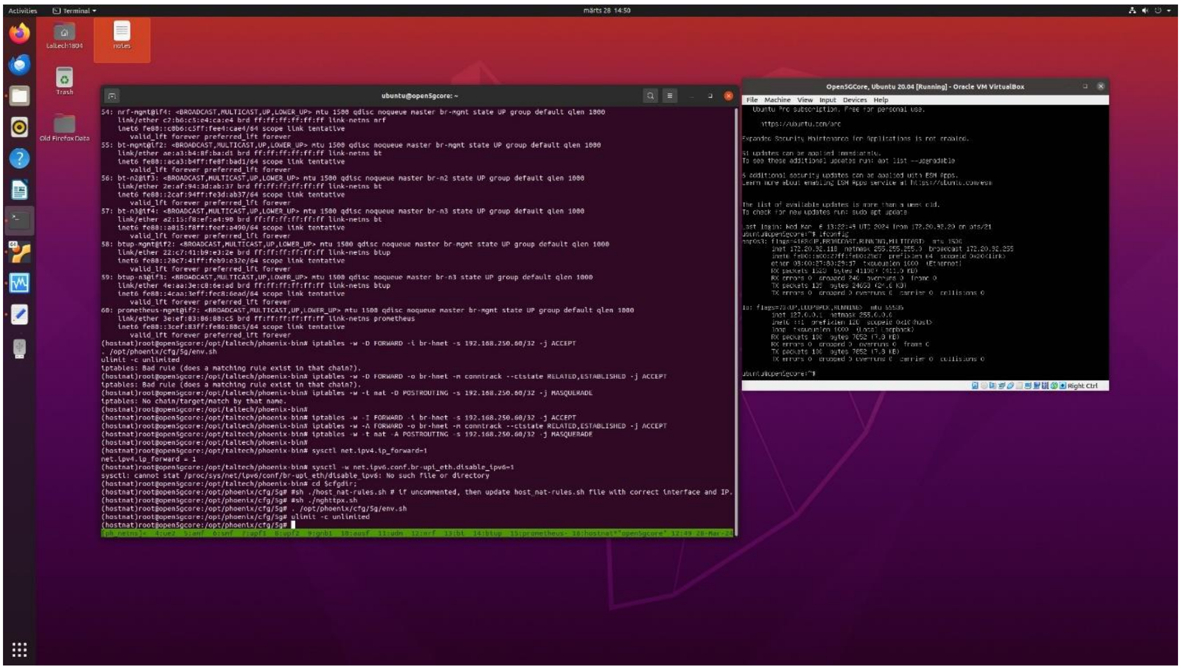
Appendix 4 – AMF info



```
14:16:35>AMF>
14:16:35>AMF>
14:16:35>AMF>
14:16:35>AMF>
14:16:35>AMF>
14:16:35>AMF>
14:16:36>AMF>
14:16:36>AMF>
14:16:36>AMF>
14:16:36>AMF>
14:16:36>AMF>
14:16:36>AMF>
14:16:37>AMF>
14:16:37>AMF>
14:16:37>AMF>
14:16:37>AMF>
14:16:37>AMF>
14:16:37>AMF>
14:16:37>AMF>
14:16:37>AMF>
14:16:38>AMF>
14:16:38>AMF>
14:16:38>AMF>
14:16:38>AMF>
14:16:38>AMF>
14:16:38>AMF>
14:16:38>AMF>
14:16:39>AMF>
14:16:39>AMF>
14:16:39>AMF>
14:16:39>AMF>
14:16:39>AMF>
14:16:39>AMF>
14:16:39>AMF>
14:16:39>AMF>
140209802929920/26972 14:19:44   WARN:Platform:phoenix_mem_prealloc_realloc():611> ngap_Message: can't realloc 0x7f85384382e0 because it is not the last allocation from this pool
140209802929920/26972 14:19:44   INFO:amf:amf_ue_context_set_cm_state():587> [UE-1] 3GPP Access Connection Management State changed to [CM-CONNECTED]
140209828108032/26972 14:19:44   INFO:amf:amf_nas_primary_aka_5g_start():177> [UE-1] NAS Primary AKA (5G AKA based) (7): Starting
140209828108032/26972 14:19:44   INFO:amf:amf_ngap_nas_transport_send_DownlinkNASTransport():385> Preparing NGAP DownlinkNASTransport message
140209828108032/26972 14:19:44   INFO:amf:amf_ngap_nas_transport_send_DownlinkNASTransport():397> Preparing DownlinkNASTransport message
140209828108032/26972 14:19:44   INFO:amf:amf_ngap_nas_transport_send_DownlinkNASTransport():430> Successfully sent NGAP DownlinkNASTransport message
140209828108032/26972 14:19:44   INFO:amf:amf_nas_primary_aka_5g_start():233> [UE-1] NAS Primary AKA (5G AKA based) (7): NAS AUTHENTICATION REQUEST sent
140209802929920/26972 14:19:44   INFO:amf:amf_nas_dispatcher():213> Received NAS AUTHENTICATION_RESPONSE message for UE Context 1
140209802929920/26972 14:19:44   INFO:amf:amf_nas_primary_aka_5g_rcv_authentication_response():79> [UE-1] NAS Primary AKA (5G AKA based) (7): AUTHENTICATION RESPONSE received
140209802929920/26972 14:19:44 NOTICE:amf:amf_nas_primary_aka_5g_rcv_authentication_response():111> [UE-1] HRES* matches HXRES* - authentification successful
140209802929920/26972 14:19:44   INFO:amf:amf_ngap_nas_transport_rcv_UplinkNASTransport():471> Successfully processed NGAP UplinkNASTransport message
140209819715328/26972 14:19:44   INFO:amf:amf_Nausf_UEAuthentication_ConfirmationData_PUT_response():581> [UE-1] Nausf_UEAuthentication_ConfirmationData Put (8): AUSF authentication success
140209819715328/26972 14:19:44 NOTICE:amf:amf_nas_security_context_init():103> [UE-1] K_AMF: 0x70BF8FACBF720D8D1382E15E13A03F52BEC1D348470E5419F0B20CDB34325667
140209819715328/26972 14:19:44 NOTICE:amf:amf_nas_security_context_init():111> [UE-1] NAS intKey: 0x98812C34E59304B526CBC5800FC3231F
140209819715328/26972 14:19:44 NOTICE:amf:amf_nas_security_context_init():120> [UE-1] NAS encKey: 0xB16CECE8A5DF22A5DF18A37BAF73525B
140209819715328/26972 14:19:44   INFO:amf:amf_nas_security_mode_start():152> [UE-1] NAS Security Mode Control (9): Starting
140209819715328/26972 14:19:44   INFO:amf:amf_ngap_nas_transport_send_DownlinkNASTransport():385> Preparing NGAP DownlinkNASTransport message
140209819715328/26972 14:19:44   INFO:amf:amf_ngap_nas_transport_send_DownlinkNASTransport():397> Preparing DownlinkNASTransport message
140209819715328/26972 14:19:44   INFO:amf:amf_ngap_nas_transport_send_DownlinkNASTransport():430> Successfully sent NGAP DownlinkNASTransport message
140209819715328/26972 14:19:44   INFO:amf:amf_nas_security_mode_start():205> [UE-1] NAS Security Mode Control (9): NAS SECURITY MODE COMMAND sent
140209802929920/26972 14:19:44   INFO:amf:amf_nas_dispatcher():213> Received NAS SECURITY_MODE_COMPLETE message for UE Context 1
140209802929920/26972 14:19:44   INFO:amf:amf_ngap_nas_transport_rcv_UplinkNASTransport():471> Successfully processed NGAP UplinkNASTransport message
140209828108032/26972 14:19:44   INFO:amf:amf_ue_context_set_rm_state():622> [UE-1] 3GPP Access Registration Management State changed to [RM-REGISTERED]
140209802929920/26972 14:19:44   INFO:amf:amf_nas_dispatcher():213> Received NAS REGISTRATION_COMPLETE message for UE Context 1
140209802929920/26972 14:19:44   WARN:Platform:str_dup_impl():110> empty source ph_str(trace: nas_msg_decode_hdr)
140209802929920/26972 14:19:44   INFO:amf:amf_registration_rcv_registration_complete():114> [UE-1] Registration (4): REGISTRATION COMPLETE received
140209802929920/26972 14:19:44   INFO:amf:amf_nas_config_update_start():57> [UE-1] NAS Generic UE Configuration Update (12): Starting
140209802929920/26972 14:19:44   INFO:amf:amf_ngap_nas_transport_send_DownlinkNASTransport():385> Preparing NGAP DownlinkNASTransport message
140209802929920/26972 14:19:44   INFO:amf:amf_ngap_nas_transport_send_DownlinkNASTransport():397> Preparing DownlinkNASTransport message
140209802929920/26972 14:19:44   INFO:amf:amf_ngap_nas_transport_send_DownlinkNASTransport():430> Successfully sent NGAP DownlinkNASTransport message
140209802929920/26972 14:19:44   INFO:amf:amf_nas_config_update_start():202> [UE-1] NAS Generic UE Configuration Update (12): NAS CONFIGURATION UPDATE COMMAND sent
140209802929920/26972 14:19:44   INFO:amf:amf_ngap_nas_transport_rcv_UplinkNASTransport():471> Successfully processed NGAP UplinkNASTransport message
14:25:42>AMF>
[ph_netns]0:root  1:sql  2:igw  3:ue1- 4:ue2  5:amf* 6:smf  7:upf1  8:upf2  9:gnb1  10:ausf  11:udm  12:nrf  13:bt  14:btup  15:prometheus  16:hostnat        "open5gcore" 13:25 19-Jan-24
```

Figure 43. AMF initial registration overview

Appendix 5 – Terminal command and started VirtualBox



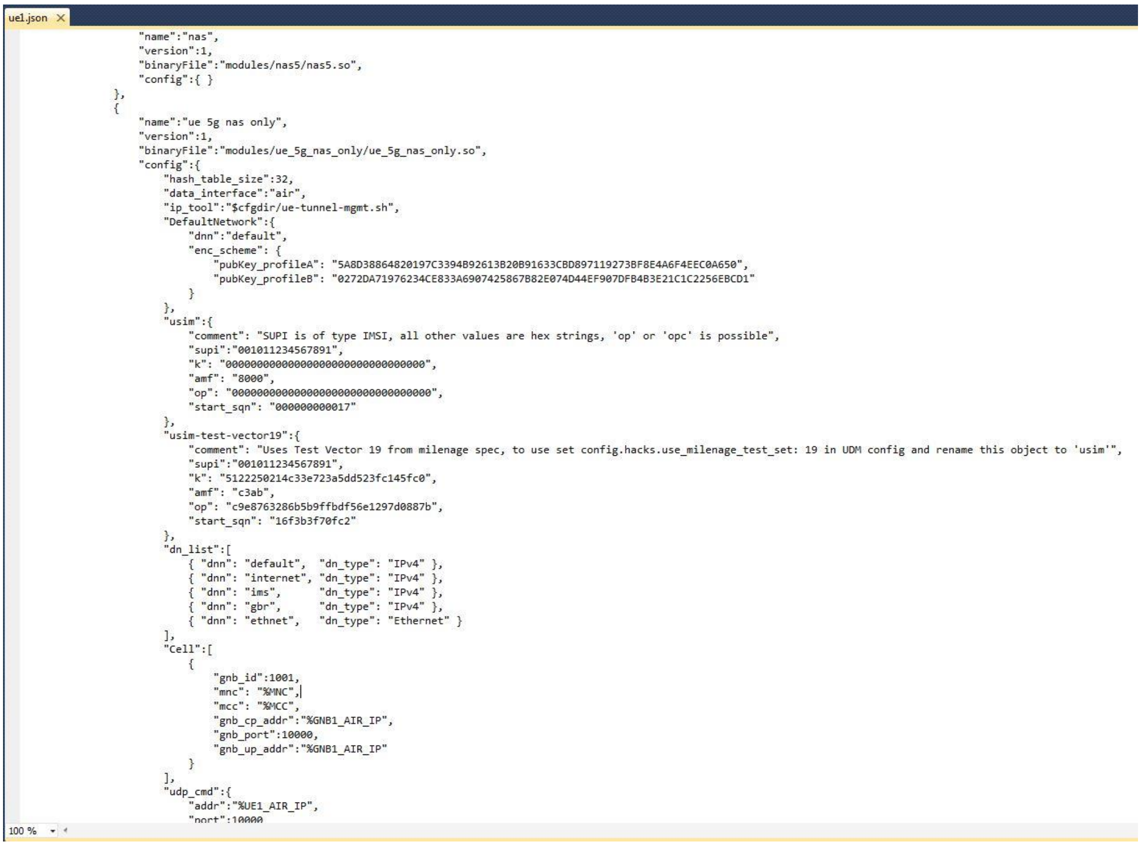
```
54: nrf-mgmt@if4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-mgmt state UP group default qlen 1000
    link/ether c2:b6:c5:e4:ca:e4 brd ff:ff:ff:ff:ff:ff link-netns nrf
    inet6 fe80::c0b6:c5ff:fee4:cae4/64 scope link tentative
       valid_lft forever preferred_lft forever
55: bt-mgmt@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-mgmt state UP group default qlen 1000
    link/ether ae:a3:b4:8f:ba:d1 brd ff:ff:ff:ff:ff:ff link-netns bt
    inet6 fe80::aca3:b4ff:fe8f:bad1/64 scope link tentative
       valid_lft forever preferred_lft forever
56: bt-n2@if3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-n2 state UP group default qlen 1000
    link/ether 2e:af:94:3d:ab:37 brd ff:ff:ff:ff:ff:ff link-netns bt
    inet6 fe80::2caf:94ff:fe3d:ab37/64 scope link tentative
       valid_lft forever preferred_lft forever
57: bt-n3@if4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-n3 state UP group default qlen 1000
    link/ether a2:15:f8:ef:a4:90 brd ff:ff:ff:ff:ff:ff link-netns bt
    inet6 fe80::a015:f8ff:feef:a490/64 scope link tentative
       valid_lft forever preferred_lft forever
58: btup-mgmt@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-mgmt state UP group default qlen 1000
    link/ether 22:c7:41:b9:e3:2e brd ff:ff:ff:ff:ff:ff link-netns btup
    inet6 fe80::20c7:41ff:feb9:e32e/64 scope link tentative
       valid_lft forever preferred_lft forever
59: btup-n3@if3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-n3 state UP group default qlen 1000
    link/ether 4e:aa:3e:c8:6e:ad brd ff:ff:ff:ff:ff:ff link-netns btup
    inet6 fe80::4caa:3eff:fec8:6ead/64 scope link tentative
       valid_lft forever preferred_lft forever
60: prometheus-mgmt@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-mgmt state UP group default qlen 1000
    link/ether 3e:ef:83:86:88:c5 brd ff:ff:ff:ff:ff:ff link-netns prometheus
    inet6 fe80::3cef:83ff:fe86:88c5/64 scope link tentative
       valid_lft forever preferred_lft forever
(hostnat)root@open5gcore:/opt/taltech/phoenix-bin# iptables -w -D FORWARD -i br-hnet -s 192.168.250.60/32 -j ACCEPT
. /opt/phoenix/cfg/5g/env.sh
ulimit -c unlimited
iptables: Bad rule (does a matching rule exist in that chain?).
(hostnat)root@open5gcore:/opt/taltech/phoenix-bin# iptables -w -D FORWARD -o br-hnet -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
iptables: Bad rule (does a matching rule exist in that chain?).
(hostnat)root@open5gcore:/opt/taltech/phoenix-bin# iptables -w -t nat -D POSTROUTING -s 192.168.250.60/32 -j MASQUERADE
iptables: No chain/target/match by that name.
(hostnat)root@open5gcore:/opt/taltech/phoenix-bin#
(hostnat)root@open5gcore:/opt/taltech/phoenix-bin# iptables -w -I FORWARD -i br-hnet -s 192.168.250.60/32 -j ACCEPT
(hostnat)root@open5gcore:/opt/taltech/phoenix-bin# iptables -w -A FORWARD -o br-hnet -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
(hostnat)root@open5gcore:/opt/taltech/phoenix-bin# iptables -w -t nat -A POSTROUTING -s 192.168.250.60/32 -j MASQUERADE
(hostnat)root@open5gcore:/opt/taltech/phoenix-bin#
(hostnat)root@open5gcore:/opt/taltech/phoenix-bin# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
(hostnat)root@open5gcore:/opt/taltech/phoenix-bin# sysctl -w net.ipv6.conf.br-upi_eth.disable_ipv6=1
sysctl: cannot stat /proc/sys/net/ipv6/conf/br-upi_eth/disable_ipv6: No such file or directory
(hostnat)root@open5gcore:/opt/taltech/phoenix-bin# cd $cfgdir;
(hostnat)root@open5gcore:/opt/phoenix/cfg/5g# #sh ./host_nat-rules.sh # if uncommented, then update host_nat-rules.sh file with correct interface and IP.
(hostnat)root@open5gcore:/opt/phoenix/cfg/5g# #sh ./nghttpx.sh
(hostnat)root@open5gcore:/opt/phoenix/cfg/5g# . /opt/phoenix/cfg/5g/env.sh
(hostnat)root@open5gcore:/opt/phoenix/cfg/5g# ulimit -c unlimited
(hostnat)root@open5gcore:/opt/phoenix/cfg/5g# 
[ph_netns]< 4:ue2 5:amf 6:smf 7:upf1 8:upf2 9:gnb1 10:ausf 11:udm 12:nrf 13:bt 14:btup 15:prometheus- 16:hostnat*"open5gcore" 12:49 28-Mar-24
```

```
 Ubuntu Pro subscription. Free for personal use.

   https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

51 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

5 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Wed Mar  6 13:22:49 UTC 2024 from 172.20.92.20 on pts/21
ubuntu@open5gcore:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.20.92.118  netmask 255.255.255.0  broadcast 172.20.92.255
        inet6 fe80::a00:27ff:fe80:29d7  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:80:29:d7  txqueuelen 1000  (Ethernet)
        RX packets 1520  bytes 411007 (411.0 KB)
        RX errors 0  dropped 240  overruns 0  frame 0
        TX packets 135  bytes 24653 (24.6 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
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 100  bytes 7652 (7.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 100  bytes 7652 (7.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ubuntu@open5gcore:~$
```

Figure 44. Preview for VirtualBox to going over terminal command

Appendix 6 – 1.json original configuration file

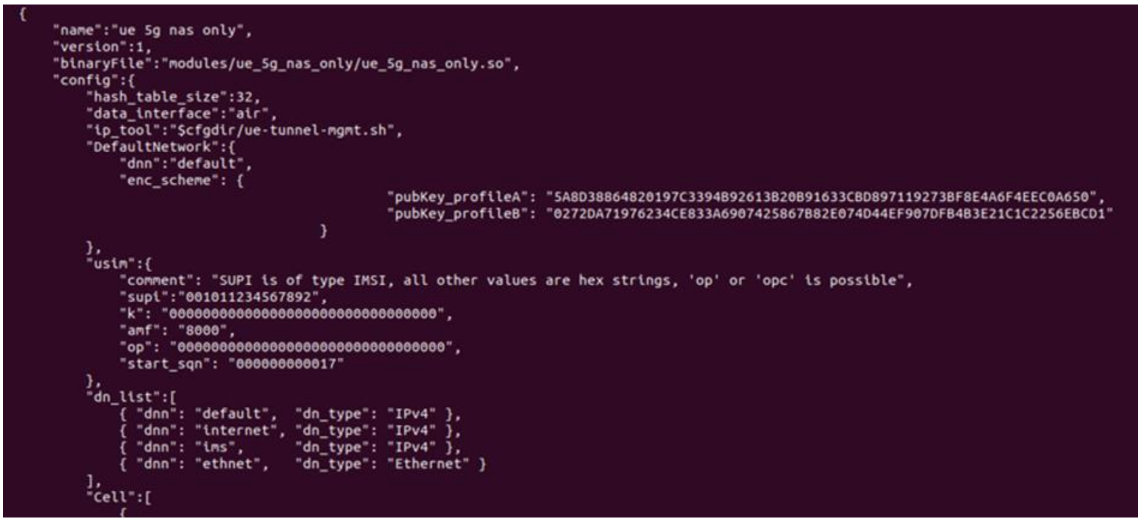
Visible SUPI code, K code and SQN code. Also, MNC and MCC code. Needed to generate SUCI code.



```
            "name":"nas",
            "version":1,
            "binaryFile":"modules/nas5/nas5.so",
            "config":{ }
        },
        {
            "name":"ue 5g nas only",
            "version":1,
            "binaryFile":"modules/ue_5g_nas_only/ue_5g_nas_only.so",
            "config":{
                "hash_table_size":32,
                "data_interface":"air",
                "ip_tool":"$cfgdir/ue-tunnel-mgmt.sh",
                "DefaultNetwork":{
                    "dnn":"default",
                    "enc_scheme": {
                        "pubKey_profileA": "5A8D38864820197C3394B92613B20B91633CBD897119273BF8E4A6F4EEC0A650",
                        "pubKey_profileB": "0272DA71976234CE833A6907425867B82E074D44EF907DFB4B3E21C1C2256EBCD1"
                    }
                },
                "usim":{
                    "comment": "SUPI is of type IMSI, all other values are hex strings, 'op' or 'opc' is possible",
                    "supi":"001011234567891",
                    "k": "00000000000000000000000000000000",
                    "amf": "8000",
                    "op": "00000000000000000000000000000000",
                    "start_sqn": "000000000017"
                },
                "usim-test-vector19":{
                    "comment": "Uses Test Vector 19 from milenage spec, to use set config.hacks.use_milenage_test_set: 19 in UDM config and rename this object to 'usim'",
                    "supi":"001011234567891",
                    "k": "5122250214c33e723a5dd523fc145fc0",
                    "amf": "c3ab",
                    "op": "c9e8763286b5b9ffbdf56e1297d0887b",
                    "start_sqn": "16f3b3f70fc2"
                },
                "dn_list":[
                    { "dnn": "default",  "dn_type": "IPv4" },
                    { "dnn": "internet", "dn_type": "IPv4" },
                    { "dnn": "ims",      "dn_type": "IPv4" },
                    { "dnn": "gbr",      "dn_type": "IPv4" },
                    { "dnn": "ethnet",   "dn_type": "Ethernet" }
                ],
                "Cell":[
                    {
                        "gnb_id":1001,
                        "mnc": "%MNC",
                        "mcc": "%MCC",
                        "gnb_cp_addr":"%GNB1_AIR_IP",
                        "gnb_port":10000,
                        "gnb_up_addr":"%GNB1_AIR_IP"
                    }
                ],
                "udp_cmd":{
                    "addr":"%UE1_AIR_IP",
                    "port":10000
```

Figure 45. UE1.json original configuration file

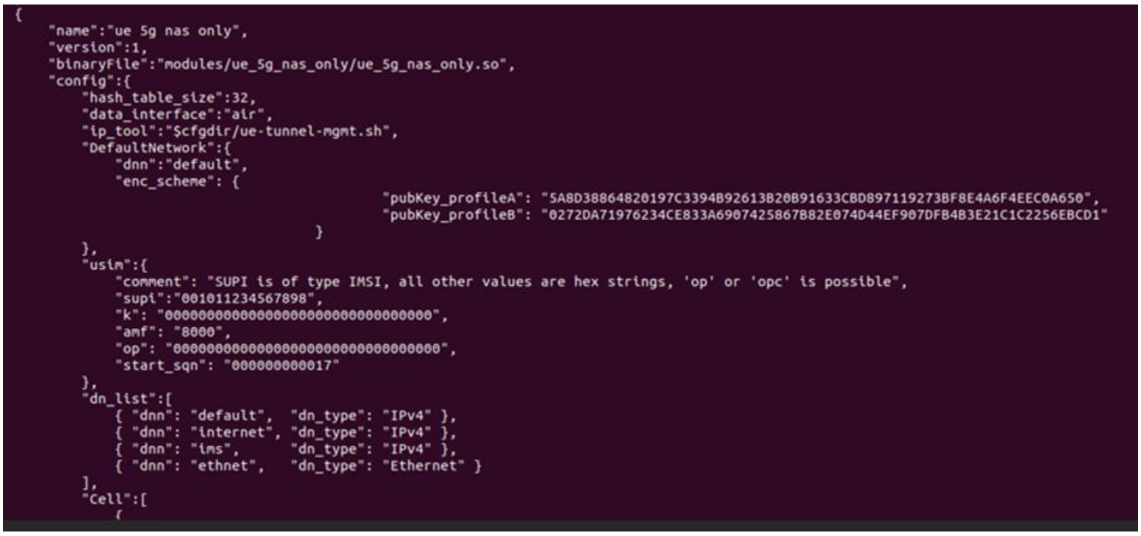
Appendix 7 – UE2 configuration file in terminal command



```
{
    "name":"ue 5g nas only",
    "version":1,
    "binaryFile":"modules/ue_5g_nas_only/ue_5g_nas_only.so",
    "config":{
        "hash_table_size":32,
        "data_interface":"air",
        "ip_tool":"$cfgdir/ue-tunnel-mgmt.sh",
        "DefaultNetwork":{
            "dnn":"default",
            "enc_scheme": {
                                            "pubKey_profileA": "5A8D38864820197C3394B92613B20B91633CBD897119273BF8E4A6F4EEC0A650",
                                            "pubKey_profileB": "0272DA71976234CE833A6907425867B82E074D44EF907DFB4B3E21C1C2256EBCD1"
                                }
        },
        "usim":{
            "comment": "SUPI is of type IMSI, all other values are hex strings, 'op' or 'opc' is possible",
            "supi":"001011234567892",
            "k": "00000000000000000000000000000000",
            "amf": "8000",
            "op": "00000000000000000000000000000000",
            "start_sqn": "000000000017"
        },
        "dn_list":[
            { "dnn": "default",  "dn_type": "IPv4" },
            { "dnn": "internet", "dn_type": "IPv4" },
            { "dnn": "ims",      "dn_type": "IPv4" },
            { "dnn": "ethnet",   "dn_type": "Ethernet" }
        ],
        "Cell":[
            {
```

Figure 46. UE 2 SUPI original code

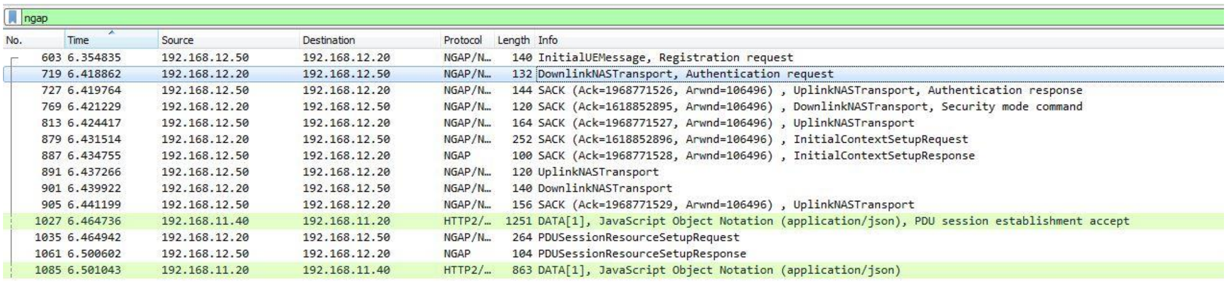
Appendix 8 – UE2 changed configuration file by author



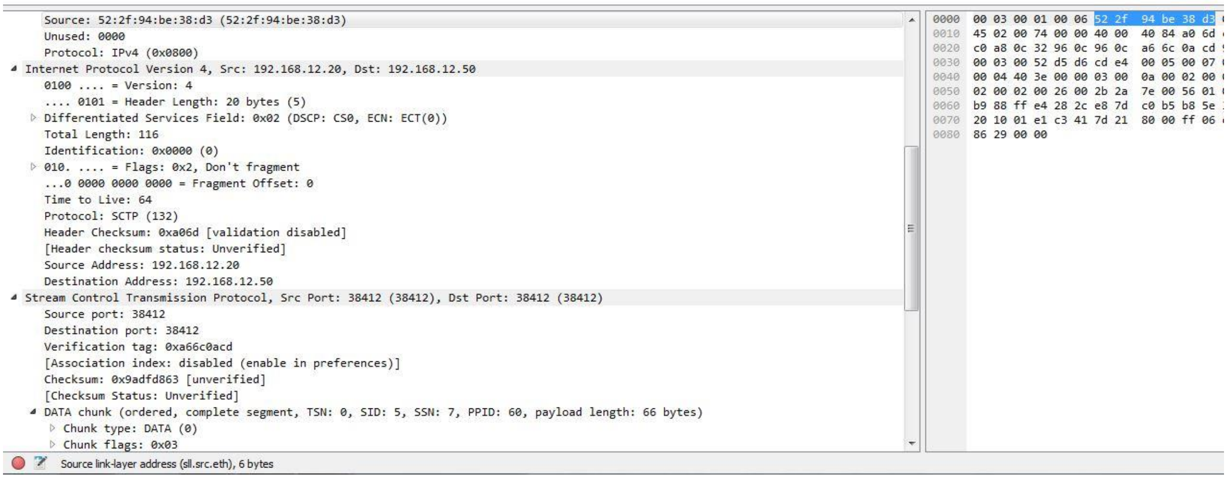
```
{
    "name":"ue 5g nas only",
    "version":1,
    "binaryFile":"modules/ue_5g_nas_only/ue_5g_nas_only.so",
    "config":{
        "hash_table_size":32,
        "data_interface":"air",
        "ip_tool":"$cfgdir/ue-tunnel-mgmt.sh",
        "DefaultNetwork":{
            "dnn":"default",
            "enc_scheme": {
                                            "pubKey_profileA": "5A8D38864820197C3394B92613B20B91633CBD897119273BF8E4A6F4EEC0A650",
                                            "pubKey_profileB": "0272DA71976234CE833A6907425867B82E074D44EF907DFB4B3E21C1C2256EBCD1"
                                }
        },
        "usim":{
            "comment": "SUPI is of type IMSI, all other values are hex strings, 'op' or 'opc' is possible",
            "supi":"001011234567898",
            "k": "00000000000000000000000000000000",
            "amf": "8000",
            "op": "00000000000000000000000000000000",
            "start_sqn": "000000000017"
        },
        "dn_list":[
            { "dnn": "default",  "dn_type": "IPv4" },
            { "dnn": "internet", "dn_type": "IPv4" },
            { "dnn": "ims",      "dn_type": "IPv4" },
            { "dnn": "ethnet",   "dn_type": "Ethernet" }
        ],
        "Cell":[
            {
```

Figure 47. UE2 changed SUPI code.

Appendix 9 – Wireshark preview info: initial registration, etc.



```
ngap
No.     Time        Source          Destination     Protocol  Length Info
 603 6.354835   192.168.12.50   192.168.12.20   NGAP/N…    140 InitialUEMessage, Registration request
 719 6.418862   192.168.12.20   192.168.12.50   NGAP/N…    132 DownlinkNASTransport, Authentication request
 727 6.419764   192.168.12.50   192.168.12.20   NGAP/N…    144 SACK (Ack=1968771526, Arwnd=106496) , UplinkNASTransport, Authentication response
 769 6.421229   192.168.12.20   192.168.12.50   NGAP/N…    120 SACK (Ack=1618852895, Arwnd=106496) , DownlinkNASTransport, Security mode command
 813 6.424417   192.168.12.50   192.168.12.20   NGAP/N…    164 SACK (Ack=1968771527, Arwnd=106496) , UplinkNASTransport
 879 6.431514   192.168.12.20   192.168.12.50   NGAP/N…    252 SACK (Ack=1618852896, Arwnd=106496) , InitialContextSetupRequest
 887 6.434755   192.168.12.50   192.168.12.20   NGAP       100 SACK (Ack=1968771528, Arwnd=106496) , InitialContextSetupResponse
 891 6.437266   192.168.12.50   192.168.12.20   NGAP/N…    120 UplinkNASTransport
 901 6.439922   192.168.12.20   192.168.12.50   NGAP/N…    140 DownlinkNASTransport
 905 6.441199   192.168.12.50   192.168.12.20   NGAP/N…    156 SACK (Ack=1968771529, Arwnd=106496) , UplinkNASTransport
1027 6.464736   192.168.11.40   192.168.11.20   HTTP2/…   1251 DATA[1], JavaScript Object Notation (application/json), PDU session establishment accept
1035 6.464942   192.168.12.20   192.168.12.50   NGAP/N…    264 PDUSessionResourceSetupRequest
1061 6.500602   192.168.12.50   192.168.12.20   NGAP       104 PDUSessionResourceSetupResponse
1085 6.501043   192.168.11.20   192.168.11.40   HTTP2/…    863 DATA[1], JavaScript Object Notation (application/json)
```



```
      Source: 52:2f:94:be:38:d3 (52:2f:94:be:38:d3)
      Unused: 0000
      Protocol: IPv4 (0x0800)
◢ Internet Protocol Version 4, Src: 192.168.12.20, Dst: 192.168.12.50
      0100 .... = Version: 4
      .... 0101 = Header Length: 20 bytes (5)
   ▷ Differentiated Services Field: 0x02 (DSCP: CS0, ECN: ECT(0))
      Total Length: 116
      Identification: 0x0000 (0)
   ▷ 010. .... = Flags: 0x2, Don't fragment
      ...0 0000 0000 0000 = Fragment Offset: 0
      Time to Live: 64
      Protocol: SCTP (132)
      Header Checksum: 0xa06d [validation disabled]
      [Header checksum status: Unverified]
      Source Address: 192.168.12.20
      Destination Address: 192.168.12.50
◢ Stream Control Transmission Protocol, Src Port: 38412 (38412), Dst Port: 38412 (38412)
      Source port: 38412
      Destination port: 38412
      Verification tag: 0xa66c0acd
      [Association index: disabled (enable in preferences)]
      Checksum: 0x9adfd863 [unverified]
      [Checksum Status: Unverified]
   ◢ DATA chunk (ordered, complete segment, TSN: 0, SID: 5, SSN: 7, PPID: 60, payload length: 66 bytes)
      ▷ Chunk type: DATA (0)
      ▷ Chunk flags: 0x03
```

```
0000   00 03 00 01 00 06 52 2f  94 be 38 d3
0010   45 02 00 74 00 00 40 00  40 84 a0 6d
0020   c0 a8 0c 32 96 0c 96 0c  a6 6c 0a cd
0030   00 03 00 52 d5 d6 cd e4  00 05 00 07
0040   00 04 40 3e 00 00 03 00  0a 00 02 00
0050   02 00 02 00 26 00 2b 2a  7e 00 56 01
0060   b9 88 ff e4 28 2c e8 7d  c0 b5 b8 5e
0070   20 10 01 e1 c3 41 7d 21  80 00 ff 06
0080   86 29 00 00
```

Source link-layer address (sll.src.eth), 6 bytes

Figure 48. Wireshark shows initial registration and authentication request info, etc.