Master Project

Integration of an Amarisoft 5G Base Station and the 5G Core Implementation Free5GC

Submitted by: Kumar Satyam
First examiner: Prof. Dr.- Ing. Ulrich Trick
Date of start: August 14, 2023
Date of submission: February 15, 2024

Statement

I confirm that I have written this thesis on my own. No other sources were used except those referenced. Content which is taken literally or analogously from published or unpublished sources is identified as such. The drawings or figures of this work have been created by myself or are provided with an appropriate reference. This work has not been submitted in the same or similar form or to any other examination board.

Kumar Satyam

Feb-11-2024

Date, signature of the student.

Content

1 Introduction

Over the past several decades, the field of telecommunication has experienced a significant evolution. This began as a basic framework for voice communications and has transformed into a vast array of technologies that allow for easy and seamless global connectivity. The telecommunications industry began in the early 19th century with the invention of the telegraph, which allowed for the first-time long-distance transmission of messages using electrical signals. This was invented by Samuel Morse in the 1830s. The advent of 5G technology has ushered in a new era for mobile network development. Previously, these networks primarily aimed to deliver communication services to individuals. In the case of 4G, multimedia data services such as video streaming with a smartphone, tablet, or generally a computer as the end device are the most important. With previous versions, the further back, the more the focus was on telephony. Now, with 5G, the multimedia applications consumed by mobile users fall under traditional services, although supported by very high bit rates. Compared to previous versions, at least before 4G, the support of M2M (Machine to Machine communications) and IoT (Internet of Things) comes more into focus, but still with the corresponding 4G air interface, now with a high connection density compared to the beginnings with 4G. A completely new feature of 5G is the support of services in system and safety-critical application areas such as Smart Grid for intelligent energy supply networks and autonomous driving with very high demands on latency, response times, and system and service availability. [Ref. (Ulrich Trick ,5G, 2024) (LinkedIn news, Published by Sound of Data)]

1G was the first generation of cellular networks that allowed for analog voice communication. 2G marked the transition from analog to digital telecommunications. This generation introduced digital voice services (such as GSM) and the first form of data communication via SMS (Short Message Service). 3G, also known as UMTS brought significant improvements in data transmission, enabling not just voice and SMS but also mobile internet and video calls. 4G, also known as LTE (Long-Term Evolution), provided even faster data speeds and more reliable internet connections. This generation supported HD mobile TV, video conferencing, and cloud computing, significantly enhancing the mobile internet experience. 5G is the latest generation of mobile networks. It promises ultra-fast data speeds, almost negligible latency, massive network capacity, and more reliability. 5G is set to enable a host of new technologies and services, including IoT (Internet of Things), autonomous vehicles, and advanced smart city applications.

2 Theoretical Background

This chapter discusses the evolution of mobile network technologies from 1G (first generation) to 5G (fifth generation). It will explain the architecture of 5G, including definitions of various Network Functions (NFs), the Non-Standalone (NSA) architecture, the 5G Standalone (SA) architecture, and the Next Generation Radio Access Network (NG-RAN) architecture. The next chapter will cover software driven Free5GC frameworks including Amarisoft's Amari Callbox Classic.

2.1 Evolution of Mobile Network in Telecommunication

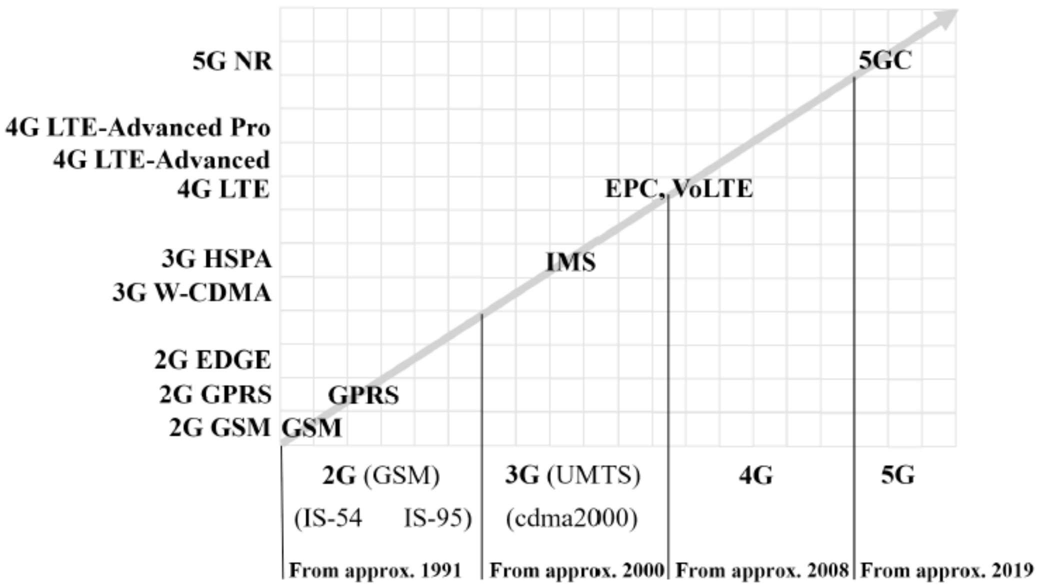


Fig 2.1 Evolution of Mobile Network in Telecommunication [Ref. Ulrich Trick, 5G, 2024]

The evolution of mobile network in telecommunication is a fascinating journey through technology advancements across different generations. Each generation brings significant improvements in speed, capacity, and their functionality. The figure 2.1 illustrates the chronological progression of mobile network technologies from the 2nd generation (2G) in approximately 1991 to the anticipated 5th generation core (5GC) in approximately 2019. It begins with 2G technologies such as GSM, progressing through GPRS and EDGE, which enhanced data capabilities within the 2G spectrum. The transition to 3G introduced UMTS and HSPA, marking significant improvements in data transmission speeds and initiating the deployment of the IP Multimedia Subsystem (IMS), which facilitated the convergence of voice, video, and messaging services over IP networks. The advancement to 4G is depicted with the introduction of LTE, followed by iterations like LTE-Advanced and LTE-Advanced Pro, which brought substantial enhancements in speed, capacity, and efficiency. This phase also introduces Voice over LTE (VoLTE) under the Evolved Packet Core (EPC) framework, optimizing the delivery of voice services over the LTE network. The image culminates with the 5G era, highlighted by the New Radio (NR) standard and the transition to 5G Core (5GC) networks, representing the latest advancements in network speed, capacity, latency, and flexibility, designed to support a vast array of devices and next-generation services.

2.2 5G Networks

Design Principle: The design includes modules for Access Network (AN) and Core Network (CN) functions for the Control Plane (CP) with the signalling and control protocols and the User Plane (UP) for the user data. These functional modules are put together and combined according to requirements. These are relatively fine-grained network functions (NF) that are provided in a repository and can be called via APIs. This concept is called Service Based Architecture (SBA).

The following design principles and key technologies are the basis of a 5G system:

- All-IP network
- Modularization with SBA
- Network softwarization with NFV, SDN, and network slicing.
- Multi-tenant capability
- Cloudification, including C-RAN and MEC
- Openness for 3rd party providers
- Heterogeneous RAN technology
- Various radio and wired access network technologies.
- Core network decoupled from access network technology.
- Flexible terminal device connectivity
- Downward and upward compatibility. [Ref. Ulrich Trick, 5G, 2024]

Overview of 3GPP Release 15:

5G system – phase 1

NSA architecture (non-standalone) – early drop

SA architecture (Standalone) – main drop

5G access network

- NR, for FR- and/or FR3
- gNB can be divided into gNB-CU (Control Unit) and gNB-DU (Distributed Unit)
- Split CU into CU-UP and CU-CP
- Dual Connectivity
- Coexistence with LTE

5G core network

- Service Based Architecture (SBA)
- Network Slicing
- Local hosting of services and edge computing
- Uniform access control
- Support of 3GPP- und non-3GPP access networks
- Framework for policy control and QoS support
- Make network functions available to 3rd party providers.
- IMS optional

Security model for NSA and SA

Enhancements for Mission Critical communication (MC; low latency, high availability) with 5G- or BG

technology (EPC, LTE)

Enhanced performance for MTC and IoT applications

Vehicle-to-Everything (V3X) – phase 3

- Platooning
- Integration of information from remote sensors (e.g., in a vehicle) into the own view of a pedestrian or other vehicle
- Autonomous driving
- Driving with remote control

WLAN for

- Proximity-based Services (ProSe) with device-to-device communication for UEs in the neighborhood
- VoWLAN (Voice over WLAN)

APIs for 3rd party access to 5G services

Requirements for Mobile communication system for railways (Future Railway Mobile Communication System, FRMCS) [Ref. Ulrich Trick, 5G, 2024]

2.2.1 5G-NSA and SA architecture:

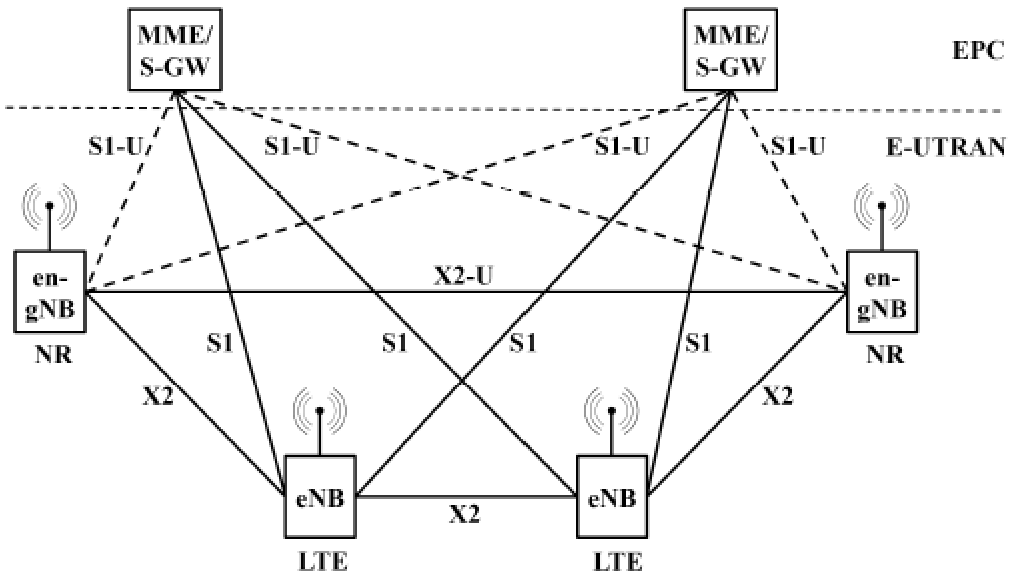


Fig 2.1 5G-NSA architecture [Ref. Ulrich Trick, 5G, 2024]

The "Non-Stand Alone" (NSA) setup is a hybrid approach where the 5G Radio Access Network (RAN) and its New Radio (NR) interface operate alongside the existing 4G infrastructure, including both the LTE radio and the Evolved Packet Core (EPC). This design allows for the deployment of NR technology without the need for a complete network overhaul. Within this arrangement, while only 4G services are directly supported, they benefit from the enhancements of the 5G NR such as reduced latency. The NSA mode is also referred to as "E-UTRA-NR Dual Connectivity (EN-DC). [Ref. 3GPP A GLOBAL INITIATIVE- 5G system overview]

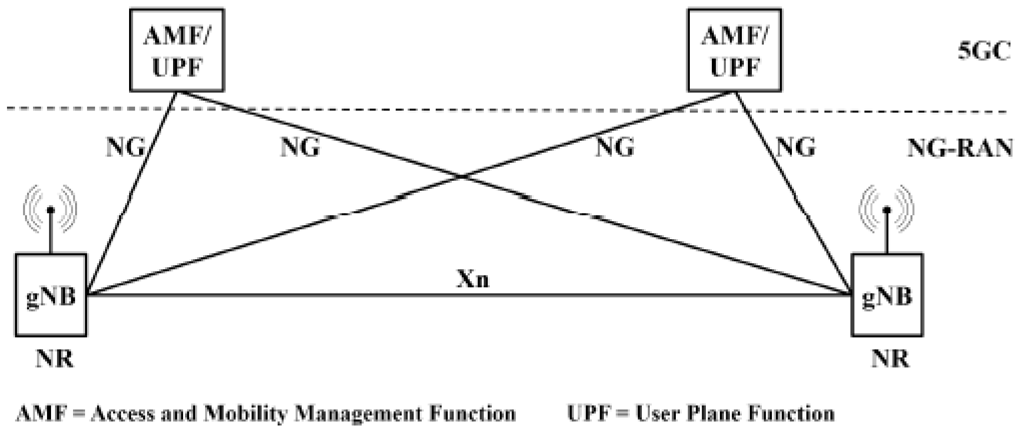


Fig 2.2 5G-SA architecture [Ref. Ulrich Trick, 5G, 2024]

The primary release within the Release 15 introduces the standardization of the SA architecture (standalone), incorporating the utilization of a new core network known as the 5G Core (5GC). As depicted in Figure 2.7, the new 5G base stations (gNB) can establish a direct connection to the 5GC. This signifies that 5G operations can now function independently without the need for 4G infrastructure. [Ref. Ulrich Trick, 5G, 2021]

2.2.2 RAN Architecture:

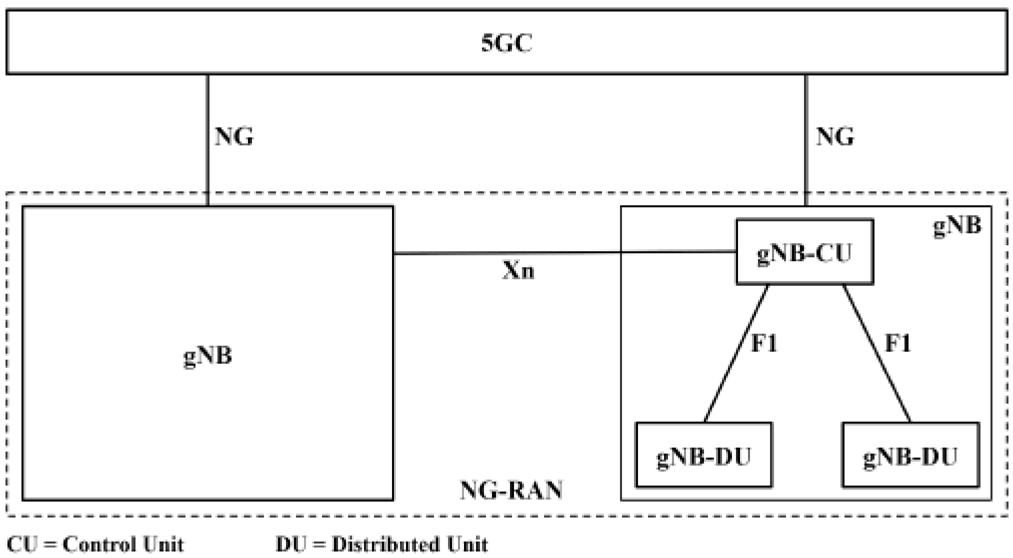


Fig 2.2.2 NG-RAN architecture [Ref. Ulrich Trick, 5G, 2024]

The NG-RAN consists of a set of gNBs connected to the 5GC through the NG interface. An gNB can support FDD mode, TDD mode or dual mode operation. gNBs can be interconnected through the Xn interface. A gNB may consist of a gNB-CU and one or more gNB-DU(s). A gNB-CU and a gNB-DU is connected via F1 interface. One gNB-DU is connected to only one gNB-CU. [3GPP TS 38.401 v 15.2.0 Release 15]

2.3 5G Architecture and Different Network Functions:

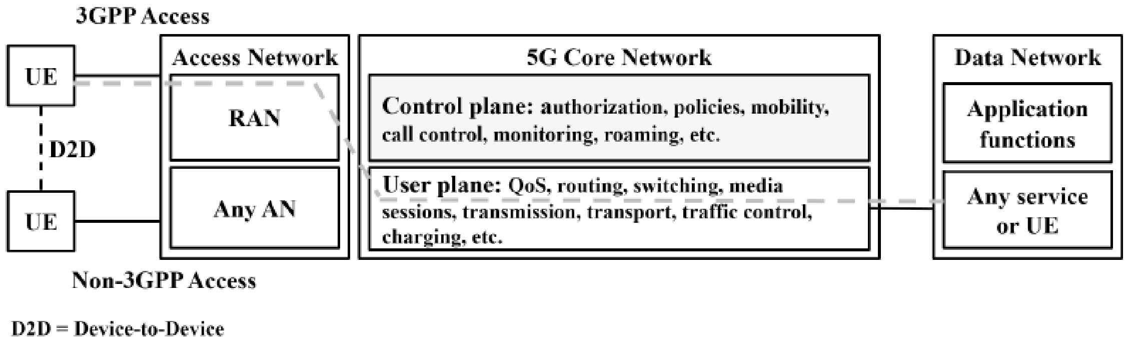


Fig 2.3.1 5G system Structuring [Ref. Ulrich Trick, 5G, 2024]

Fig 2.3.1 illustrates the structure of 5G system in four area:

- Terminal equipment (UEs)
- Access Network (AN) with various 3GPP and non-3GPP RATs and wired subscriber interface.
- Core Network (CN)
- Data Network with the actual applications or communication services for the users.

This image shows the division of the network functions into the Control Plane (CP) with the signaling and control protocols and the User Plane (UP) for the user data., that is helpful for modularization. The UP includes functions such as user data transport, routing, and forwarding of data packets, traffic control, provisioning of the required QoS, ensuring service continuity for mobile use, and recording of billing data. The CP is responsible for authentication and authorization, compliance with the specified or agreed policies for users and the network, mobility management including roaming, and the connection of 3rd party providers to the 5G core network concerning signaling, control, and monitoring. [Ref. Ulrich Trick, 5G, 2024]

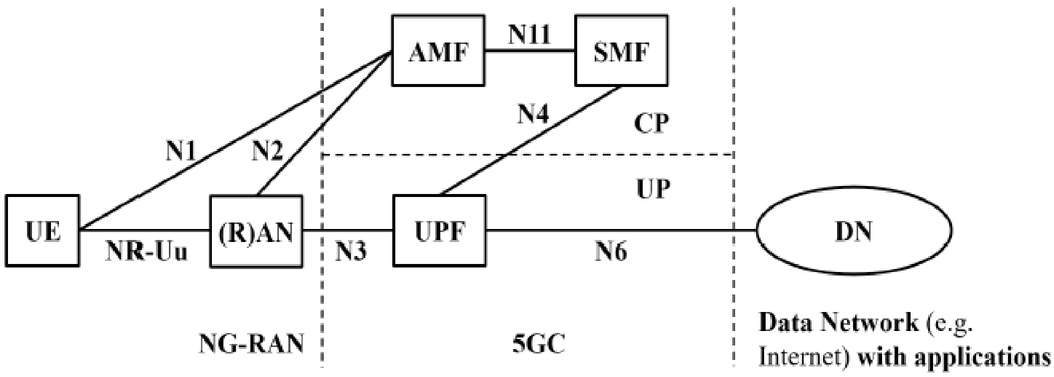


Fig 2.3.2 5G Basic Architecture [Ref. Ulrich Trick, 5G, 2024]

Fig 2.3.2 shows the 5G basic system architecture, and it also showing the basic 5GC network function UPF (User Plane Function) in the User Plane (UP) and AMF (Access and Mobility Management Function), and SMF (Session Management Function) in the Control Plane (CP). Three 5GC NFs are always required when user data is exchanged between a base station in the RAN and the Data Network (DN), e.g., the Internet, via the 5G core.

Also, this figure communicates peer-to-peer with each other: UE and gNB in the RAN via the NR-Uu interface, UE and AMF via N1, gNB and AMF via N2, gNB and UMF via N3, UPF and DN via N6, where the SMF controls the UPF via the N4 interface. N11 between AMF and SMF takes a special role since it is a service-based interface as part of the Service Based Architecture (SBA). The 5GC network functions AMF, SMF, and UPF, the (R)AN with the UE terminal connected, and the external DN with the application functions used by the UE (e.g., web server for WWW services or IMS for telephony), standardized reference points Nx for their interconnection.

[Ref. Ulrich Trick, 5G, 2024]

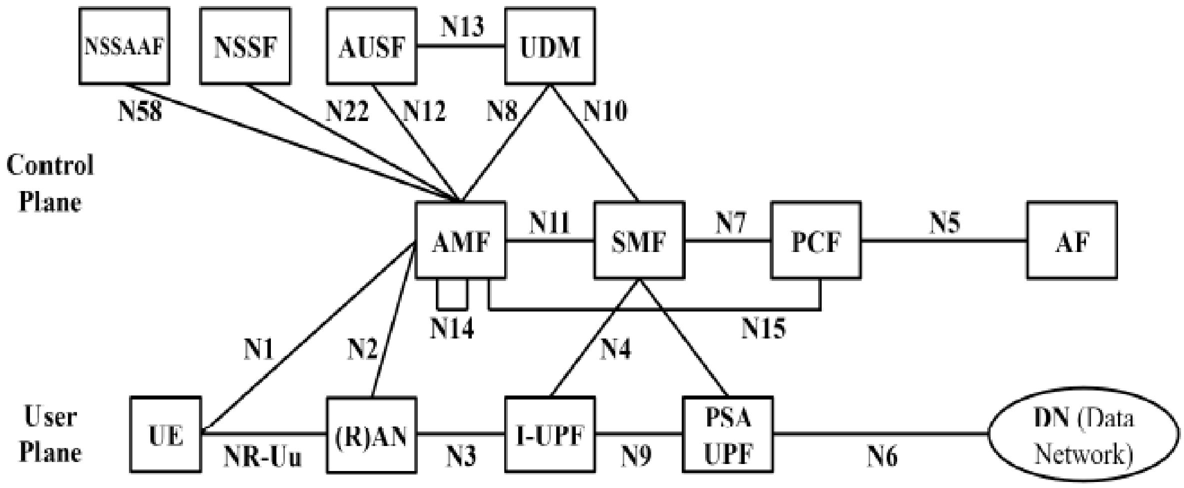


Fig 2.3.3 NFs in 5G system [Ref. Ulrich Trick, 5G, 2024]

Access and Mobility Management Function (AMF)

The Access and Mobility Management Function (AMF) mimics half of the functionalities of the MME of 4G EPC. The AMF obtains the connection and session related information from UE and RAN via point-to-point reference interfaces N1 and N2 respectively. At the interface N1, the AMF, serving as an access point in CP, applies the NAS ciphering and integrity protection techniques on the user traffic. At the N2 interface, the AMF executes the 5GC control part signaling with RAN. AMF is responsible for registration, connection, accessibility, and mobility management. It only deals with connection and mobility management, while all the session management-based operations are redirected to SMF. Furthermore, user access authentication and authorization, lawful inter-ception and other security management tasks are also conducted by the AMF (3GPP TS 23.501 V15.10.0, 2020).

Session Management Function (SMF)

The Session Management Function (SMF) is a crucial NF of 5G SBA, since it interacts with the decoupled User Plane Function (UPF) by instigating, updating, and releasing the PDU session and handling session framework within the UPF. The SMF also serves as a Dynamic Host Configuration Protocol (DHCP) server and performs IP address allocation and authorization for UE, while segregating CP functions from UP. Furthermore, it performs policy enforcement, QoS management, UPF selection, traffic steering configuration for UPF, etc. (3GPP TS 23.501 V15.10.0, 2020).

Authentication Server Function (AUSF)

The Authentication Server Function (AUSF) performs the UE authentication for both 3GPP and non-3GPP access. The AUSF mainly exhibits Extensible Authentication Protocol (EAP) server's functionalities and stores authenti-cation keys for UEs. It supports AMF in obtaining the legitimacy of each UE. The AUSF receives the UE's subscription detail for an authentication method (EAP or 5G-AKA) from Unified Data Management (UDM), based on which it performs further authentication tasks (3GPP TS 29.509 V16.7.0, 2021).

Policy Control Function (PCF)

The Policy Control Function (PCF) operates on the unified policy control framework in order to provide policies for regulating network activities such as traffic steering, QoS handling, etc. It retrieves user subscription details from Unified Data Repository (UDR) to make policy-based decisions and forward these policy rules to other CP NFs (3GPP TS 29.509 V16.7.0, 2021).

Unified Data Management (UDM)

The Unified Data Management (UDM) in 5GC imitates the functionalities of 4G EPC's HSS. The UDM manages user data and performs functionalities based on subscriber's profile such as user identification and subscription management, user authentication by providing 3GPP AKA certificates, access authorization, storage of serving NFs for UE, etc. (3GPP TS 29.509 V16.7.0, 2021).

Network Slice Selection Function (NSSF)

The Network Slice Selection Function (NSSF) contains the set of operator-defined network slice instances. It determines the network slice instances for a particular UE based on the subscription details and decides the most suitable AMF instances to serve the UE. NSSF also defines the permitted and configured Network Slice Selection Assistance Information (NSSAI) (3GPP TS 29.509 V16.7.0, 2021).

Application Function (AF)

The Application Function (AF) is not a part of 5G CN; however, it interconnects with 5G CN for additional CP services provision. The AF can directly interact with relevant 5GC NFs if authorized by the operator. The unauthorized AF can communicate with relevant 5GC NFs via NEF. The AF is responsible to access NEF for resources retrieval, communicating with PCF for policy control, support application effect on packet routing (3GPP TS 29.509 V16.7.0, 2021).

User Plane Function (UPF)

In 5G CN, the User Plane Function (UPF) mimics the operations of 4G EPC's SGW-U and PGW-U since it supports the control and user plane separation mechanism. The UPF is responsible for managing user data and acts as the coupling point for inter-RAT or intra-RAT traffic mobility including user traffic steering via the N3 interface and towards DN. The 5GC UPF performs some critical functionalities such as routing and forwarding of user data packets, policy enforcement under UP, QoS management, packet inspection, lawful interception, etc. The reference point interface N4 allows the configuration of UPF functionality by the SMF (3GPP TS 29.509 V16.7.0, 2021).

Regarding the UPF, two UPFs connected via reference point N9 are shown in Figure 2.3.3, an I-UPF (Intermediate UPF) and a PSA UPF (PDU Session Anchor). The PSA UPF must always exist in a 5GC. It implements the IP anchor, i.e., the GTP-U tunnel endpoint at the transition to the DN and is thus essential for mobility. Even if the UE moves and, as a result, the gNB and, thus, its IP address changes, the tunnel through the 5GC remains unchanged for the application data. [Ref. Ulrich Trick, 5G, 2024]

Network Slice Specific Authentication and Authorization Function (NSSAAF)

The NSSAAF (Network Slice Specific Authentication and Authorization Function) processes slice-specific authentication and authorization requests from the AMF. Communicates with the corresponding AAA server and provides protocol conversion. [Ref. Ulrich Trick, 5G, 2024]

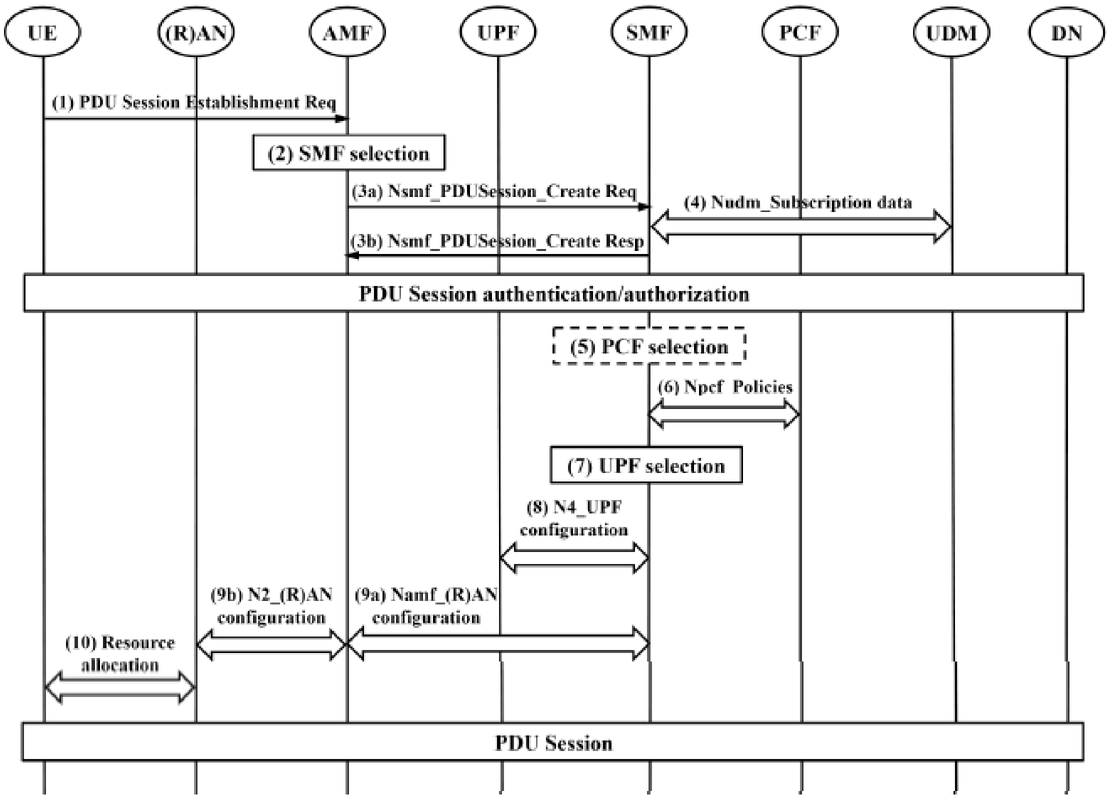


Fig.2.3.4 PDU session setup [Ref. Ulrich Trick, 5G, 2024]

According to fig 2.3.4, the UE sends a (1) PDU Session Establishment Request to the AMF. This selects an SMF in (2) and signals the desired PDU session establishment in (3) via the SBI Nsmf. The SMF in (4) then contacts the UDM (Unified Data Management) via Nudm to retrieve the subscription data. If necessary, the SMF selects a PCF (Policy Control Function) in (5). The SBI Npcf is then used to call up the policies for the desired PDU session in (6). Then the SMF in (7) selects the UPF for this PDU session and allocates an IPv6 address or IPv6 prefix. Next, the UPF is configured via N4 in (8). In (9), the (R)AN is then configured accordingly via the Namf of the AMF and N2. And in (10), the UE is equipped with IPv6 address and prefix and provided with the appropriate QoS rules. The PDU session is established, and user data packets can be transmitted from the UE to the DN (Data Network) and vice versa. [Ref. Ulrich Trick, 5G, 2024]

2.4 Free5GC

Free5GC is an open-source project focused on developing a 5G core network (5GC) that is based on the 3GPP Release 16 standards and beyond. It involved the support for 5G NGAP protocol, 5G NAS encryption and integrity protection algorithms and 5G authentication features. A fully functional 5G Core Network together with application services and network slicing features is introduced. The Free5GC software is implemented with the combination of C and Golang. Free5GC includes different NFs such as Access and Mobility Management Function (AMF), Session Management Function (SMF), User Plane Function (UPF) and others. The Free5GC software is available under the Apache 2.0 license. For the software installation, the source code can be freely downloaded from the website or GitHub repository. [Free5GC Github]

2.5 Amarisoft Amari Callbox Classic

Amarisoft delivers the state-of-the-art 3GPP compliant fully functional closed source 5G soft-ware suite that runs on COTS hardware and provides the functional and performance testing of 4G LTE and 5G NR services with all types of UEs having advanced configurations. One of the most remarkable and affordable 5G solutions is offered by Amarisoft, named the Amari Callbox Series fig 2.5. [Amarisoft, 2021]

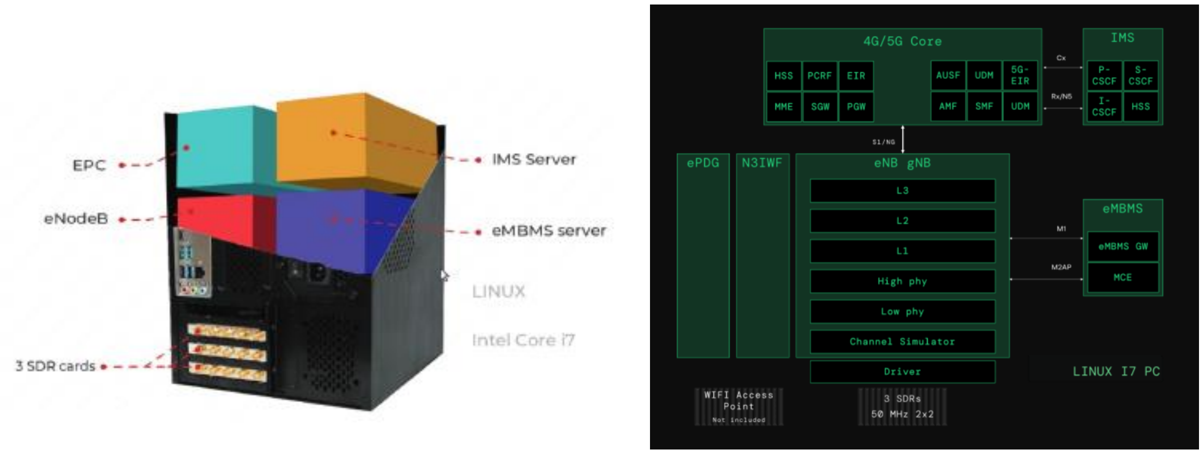


Fig 2.5 Amari Callbox and its architecture [Amarisoft]

The Amari Callbox Classic not only contains the software components of 5G system such as gNB and 5GC, but also eNB and EPC software components for 4G LTE system. Furthermore, it integrates the IP Multimedia Sub-system (IMS) server and evolved Multimedia Broadcast and Multicast Services (eMBMS) gateway to test Voice over LTE and other advanced multimedia services.

3 Requirements Analysis

The requirements analysis will highlight the objective of this project, what are the specific requirements and the goals, and the potential uses of the application like Amarisoft and Free5GC. This part elaborates on the technical constraints, including hardware and software considerations, that need to be considered while establishing a standalone 5G network test environment.

3.1 General Objectives

The primary objective of this project is to achieve the Integration of an Amarisoft 5G Base Station and the 5G Core Implementation Free5GC. Free5GC is an open-source project, which is available online freely and the main objective or goal is to implement the 5G core network (5GC) in 3GPP Rel-15. These functions include the Access and Mobility Management Function (AMF), User Plan Function (UPF), Session Management Function (SMF), Authentication Server Function (AUSF), Unified Data Management (UDM).

The initial phase involves setting up and configuring the Free5GC software, followed by a series of tests on the Free5GC core network. Testing of Free5GC core network that includes Testing of Registration, GUTI-Registration, Service Request, Xn-Handover, Deregistration, PDU session Release Request, Paging, N2Handover, Non3GPP, ReSynchronization, und ULCL. Free5GC provides a simple web tool Web Console to help create and manage UE registrations to be used by various 5G Network Function (NF). [Free5gc user guide]

Amari Callbox Classic physical 5G testbed or base station. It is a 3GPP compliant gNodeB and 5GC allowing functional and ideal solution or performance for NR testing of all types of user equipment with advanced configuration. The Callbox is powered by a deployment quality NR software suite offering the same level of baseband functionality as an indoor/outdoor network.

3.2 Clarifying the Requirements

The hardware and software requirements for the experimental setup that is used to emulate an operational 5G core network environment is listed in the following sections.

3.2.1 Hardware Requirement for Amari Callbox Classic and Free5GC Network

The Free5gc v3.3.0. Hardware Requirements:

Component	Minimum Requirement	Recommended Requirement
CPU	Intel i5 processor	Intel i7 processor
RAM	4GB	8GB
Hard Drive	160GB	160GB
NIC	1Gbps Ethernet card (Linux kernel supported)	10Gbps Ethernet card (Linux kernel supported)

[Ref. https://free5gc.org/guide/Environment/ (Free5gc Recommended Environment)]

Amarisoft: Amari Callbox Classic Hardware Requirements:

A PC (Host) used to take the remote session of callbox software components and to connect with the Amarisoft Web GUI.

Appropriate antennas for the intended LTE frequencies or cables and attenuators to connect to a UE. Any commercial UE compatible with LTE FDD or TDD. All LTE FDD and TDD frequency bands are supported. If you use the Amarisoft Core Network, the device must accept test USIM cards. A test USIM card which is provided by Amarisoft. Power adapter specifications: Input: 100~240 V AC, 50~60 Hz. A fast PC:

For best performances, a quad core Intel Core i5 or i7 CPU with AVX2 support (Haswell

architecture or later) is recommended. Support of the SSE4.1 instruction set extension is required to run the software.

 At least 2x 1 Gigabit Ethernet ports.

 At least 2 GB of RAM.

 At least 1 GB of hard disk space.

gNodeB technical Specification:

Specification	Details
3GPP release	Release 16
Frequency bands	FDD/TDD FR1 (< 6 GHz)
Bandwidth	Up to 50 MHz
MIMO	Up to MIMO 4x4 in DL
Subcarrier spacing	Data subcarrier spacing: 15, 30, 60 or 120 KHz
	SSB subcarrier spacing: 15, 30, 120 or 240 KHz
Modulation schemes	Up to 256QAM in DL and 256QAM in UL
Supported modes	NSA, SA
NR Split Bearer	3, 3a and 3x
Use case	eMBB
Network interfaces	NG interface (NGAP and GTP-U) to 5GC
	XnAP between gNodeBs
Carrier Aggregation	Up to 3 DL carriers in SA and NSA
Handover	Intra gNodeB, NG, Xn and 5GS to EPS handover support

5G Core Technical Specifications:

Specification	Details
Network elements	Access and Mobility Management Function (AMF), Authentication Server Function (AUSF), Session Management Function (SMF), User plane Function (UPF), UDM (Unified Data Management), and 5G-EIR (5G Equipment Identity Register) all integrated within the same software component
3GPP release	Release 16
NAS encryption and integrity protection	AES, SNOW3G, ZUC
USIM authentication	XOR, Milenage, TUAK 5G-AKA
IP version	IPv4, IPv4v6, IPv6 and unstructured PDUs support
QoS	Configurable QoS flows
PDU	Multi PDU sessions support
Network interfaces	NG interface (NGAP and GTP-U protocols) to several gNodeBs, ng-eNodeBs or N3IWFs Rx to external IMS server, N12 to external AUSF N8 to external 5G-EIR, N50 to external CBC
RAT	NR, LTE, NB-IoT and non-3GPP RAT
Handover	intra-AMF and 5GS EPS IRAT support

[Ref. https://www.amarisoft.com/app/uploads/2021/10/AMARI-Callbox-Classic.pdf (Amarisoft Academy)]

3.2.2 Software Requirements for Free5gc and Amari Callbox Classisc:

Free5GC v3.3.0 Software Requirements:

Requirement	**Specification**
Operating System	Ubuntu 20.04.1
Compiler	gcc 7.3.0
Programming Language	Go 1.14.4 linux/amd64 or Go 1.17.8 linux/amd64
Kernel Version	5.4.0-169-generic (5.4.x -generic)

[Ref. https://free5gc.org/guide/Environment/ (Free5gc Recommended Environment)]

Wireshark is also used for monitoring and troubleshooting purposes. So, in Free5gc, Wireshark can capture packets on any core-connected interface to check with various testing, like registration, service requests, and handovers. It supports a variety of protocols such as ‘ngap||nas=5g||rrc||gtp-u||stcp’ which are used in the communication between the core network and the gNB of amarisoft.

Amari Callbox Classic software requirements:

A 64-bit Linux distribution. Fedora 36 is the officially supported distribution.

The following distributions are known as compatible:

- Fedora 22 to 36
- Cent OS 7
- Ubuntu 14 to 20

Your system requires at least GLIBC 2.17.

[Ref. lteenb-2023-06-10]

Sim Card:

Sim Card Specification:

- IMSI- 001010123456789
- K: 00112233445566778899aabbccddeeff
- sim algo: XOR (Supported by Amarisoft)
- Nonprogrammable USIM card
- Support both 1.8V and 3V voltage.

SIM card reader to change the configuration which parameter is required during project or which parameter need to be added.

3.3 Time frames

In this section, the timetable including the start and end date of the project is depicted. Moreover, the milestones of the work are presented as well.

Start date: 14.08.2023.

End date: 15.02.2024.

Milestone:

14.08.2023 - 09.10.2023:	Analysis of the requirements
09.10.2023	Submission of the requirement analysis
02.09.2023 - 10.11.2023:	Literature review and learning theoretical background.
25.08.2023 - 10.10.2023:	Free5GC setup
11.10.2023 - 20.10.2023:	Amarisoft setup
15.10.2023 - 30.10.2023:	Planning of feasibility of different test scenarios
01.11.2023 – 18.11.2023:	Compatibility Test Execution phase
19.11.2023 – 05.12.2023:	Documentation update based on test result / re-execution.
07.12.2023:	Submitting the project draft to the supervisor.
09.12.2023-17.12.2023:	Revise the project based on review/proposals from the supervisor.
24.12.2023:	Submitting the project to the examination office.

3.4 Target State

The free5GC is an open-source project for 5th generation (5G) mobile core networks. The goal of this project is to implement the 5G core network (5GC) defined in 3GPP Release 15 (R15) and beyond. It contains various components within itself like AMF (Access Management Function), SMF (Session Management Function), UPF (User Plane Function) etc. The components are the most important ones and are of paramount importance when building the communication system. Each of the above components are connected to each other to enable smooth functioning of the system.

All the network elements of 5GC such as AMF (Access and Mobility Management Function), SMF (Session Management Function), UPF (User Plane Function), are software-defined and hosted inside the Callbox PC. The network functionality of gNB is also available in the same Callbox PC. The PCIe SDR cards are attached to the Callbox and serve radio frequency frontend can relate to the antennas and controlled by the access network. The core network maintains a persistent database storing all the parameters of the configured UEs. The utilized commercial UEs, i.e., 5G-powered mobile phones shall be compatible with the employed n77 or n78 NR frequency band, specifically for the range of 3.7 to 3.8 GHz. The Amarisoft Core Network, the device must accept test USIM cards. Insert the test USIM card in the device and configure accordingly.

When the 5G environment runs, the connection between the AMF of 5GC and the gNB should be established using the NGAP signaling. Afterward, the SCTP connection shall be established between the 5GC and gNB. When UEs are turned on and kept within a few meters range (not more than 10 meters) from the gNB's antennas, they shall start scanning the compatible NR bandwidth. Once the UEs detect the gNB signals, the initial PRACH signals alongside the registration request from the UEs shall be transmitted to the base station or gNB. Afterward, the UEs shall be able to connect with the simulated 5G core network. The 5G core network contains the persistent UE database, which stores all the data and parameters of the configured UEs. Once the initial connection is established successfully, then the UEs shall connect to the 5G and get the IP addresses allocated, after which each UE shall display the 5G signal icon and "connected" status.

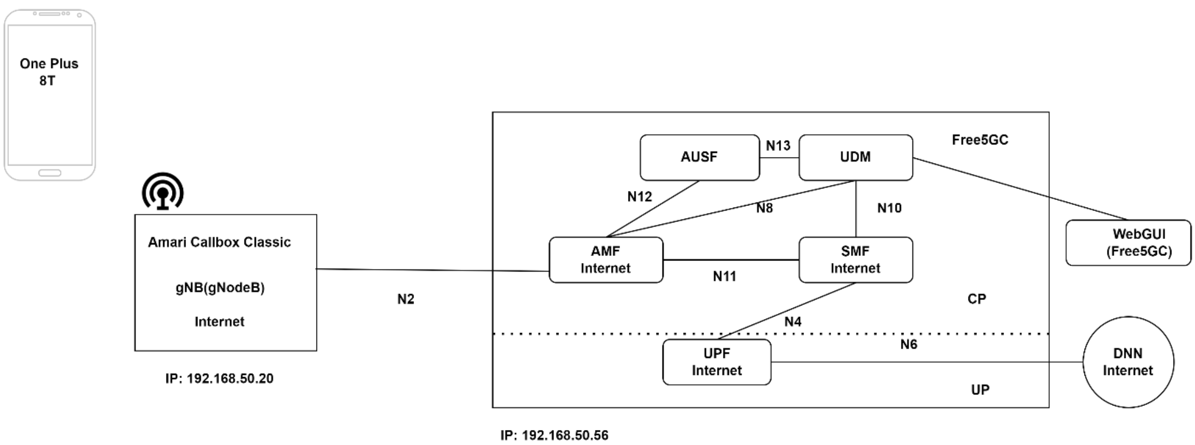


Figure 3.2: Target system in an overview

3.5 Use Cases for the Prototype

The prototype developed in this project utilizes the Amari Callbox Classic to establish a 5G network environment. The use cases considered are as follows:

Implementation and testing Free5GC and Amari Callbox Classic gNB:

- The purpose of this prototype is to integrate and meticulously evaluate the performance of the Free5gc and the gNB to ensure that they work efficiently, with the project goal.
- The software elements of Free5gc and Amari callbox classic “gNB” 5G base station are setup and examined in a standalone mode (SA).
- Testing of software-defined network elements of 5GC i.e., AMF, SMF, UPF, AUSF, UDM, with the gNB to check whether they are communicating or not.

Functional testing of 5G NR

- The prototype is tested on the 5G NR FR1 bands such as n77 and n78 in TDD mode.
- The emulated 5G system is capable of transmitting and receiving on the above-stated NR frequency range.
- The permitted frequency range lies between 3700 MHz to 3800 MHz (3.7 GHz to 3.8 GHz).

Evaluation of 5G capable smartphone:

- The smartphone model “OnePlus 8T” or UE (User Equipment) is used for experimental purposes within the crafted 5G network.
- The simulated 5G network is suitable for conducting tests with commercially available 5G enabled mobile phones.

Testing of SIM Card:

- The sim card performs user access authentication and authorization by using sim algorithms XOR, TUAK 5G-AKA, MILENAGE.
- For this use case sim card reader is in use. To check whether PLMN, K, OPC, IMSI, these numbers are correctly configured or not.

4 Realisation

This section of the document showcases the configuration and detailed process of establishing an advanced 5G standalone network using the capabilities of the Amari Callbox Classic with Free5GC.

4.1 Setup of 5G Standalone environment

The aim of this project is to establish an open source 5G Core network using Free5GC on a local virtual machine, to integrate with the Amari Callbox Classic (gNB), and to facilitate remote access to the Amarisoft Web GUI along with its various components.

Figure 4.1 presents a detailed view of the entire system configuration, showcasing the integration of the Amari Callbox Classic with the Free5GC for the development of a 5G network setup.

- To fulfill the objectives, another PC “Host PC” is connected to the Amari callbox PC. The host PC is used to install and configure Free5GC software under the research study.
- The host PC will enable remote connection to the Callbox component through the Secure Shell Protocol (SSH) for remote operations. In addition, the Amarisoft Web GUI is accessible via this same computer.

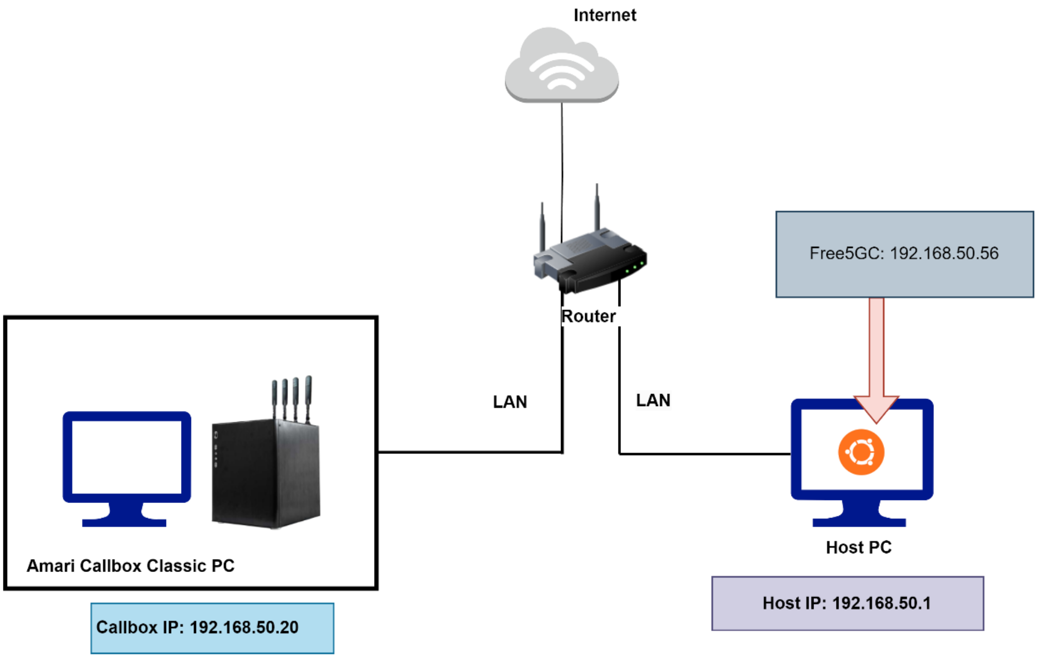


Fig 4.1: 5G Network Setup with Free5GC and Amari Callbox Classic

Network Setup

The router comes equipped with built-in DHCP capabilities. As soon as the router and the PCs are activated, the router will seamlessly distribute the dynamic address of IPs to each PC. Fig. 4.2, Fig 4.3, and Fig 4.4 shows the IP allocated to Amari Callbox Classic (192.168.50.20), Free5GC (192.168.50.56) and Host PC (192.168.50.1).

After the successful IP assignment, Both PCs should be able to ping or able to take tcpdump from each other.

```
[root@CBC-2021102801 ~]# ifconfig
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.50.20  netmask 255.255.0.0  broadcast 192.168.255.255
        inet6 fe80::3a41:95f0:d615:489c  prefixlen 64  scopeid 0x20<link>
        ether 7c:10:c9:9c:6d:be  txqueuelen 1000  (Ethernet)
        RX packets 458942  bytes 85867402 (81.8 MiB)
        RX errors 0  dropped 250733  overruns 0  frame 0
        TX packets 43985  bytes 10041111 (9.5 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 16  memory 0xa0500000-a0520000

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 563102  bytes 47360457 (45.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 563102  bytes 47360457 (45.1 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Fig 4.2: IP Address of Amari Callbox Classic

```
ubuntu@free5gc:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.50.56  netmask 255.255.0.0  broadcast 192.168.255.255
        inet6 fe80::a00:27ff:fe5e:6e3b  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:5e:6e:3b  txqueuelen 1000  (Ethernet)
        RX packets 58  bytes 11729 (11.7 KB)
        RX errors 0  dropped 26  overruns 0  frame 0
        TX packets 13  bytes 1554 (1.5 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 80  bytes 5920 (5.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 80  bytes 5920 (5.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Fig 4.3: IP Address of Free5GC

```
PS C:\Users\Student> ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : fb2.fh-frankfurt.de
   Link-local IPv6 Address . . . . . : fe80::18d8:b3bd:3147:bf1e%15
   IPv4 Address. . . . . . . . . . . : 192.168.50.17
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . : 192.168.0.1
```

Fig 4.3: IP Address of Host PC

Access Remote of Amari Callbox PC:

Amari Callbox Classic can be remotely accessed via the Host PC by executing the following command in the Host PC's terminal as show in Listing4.1

```
ssh root@192.168.50.20   // 192.168.50.20 is the IP of Amari Callbox Classic
username: root
password: toor
```

Listing 4.1: CLI command to access Amari Callbox Classic

Access Remote of Amarisoft WebGUI:

The WebGUI can be remotely accesses from any Web Browser on the Host PC by using the following URL listing 4.2:

```
http://192.168.50.20/lte/   // Amarisoft WebGUI
username: root
password: toor
```

Listing 4.2 WebGUI of Amarisoft

Free5gc WebGUI:

The WebGUI can be accesses from Web Browser on the Host PC by using the following URL listing 4.3:

```
http://192.168.50.56:5000/#/   // Free5GC WebGUI
username: admin
password: free5gc
```

Listing 4.3: Free5GC WebGUI

Remote Access for Free5GC:

The Host PC can access free5gc via powershell or cmd by executing the following command listing 4.4:

```
ssh 192.168.50.56 -l ubuntu    // IP address of Free5GC
username: ubuntu
password: ubuntu@123
```

Listing 4.4: CLI command to access Free5GC.

4.2 Installation

1) Installation of Ubuntu VM and Ubuntu Server:

To install VirtualBox version 6.1.18, it needs to be downloaded from the official VirtualBox website at virtualbox.org as shown in Fig 4.5.

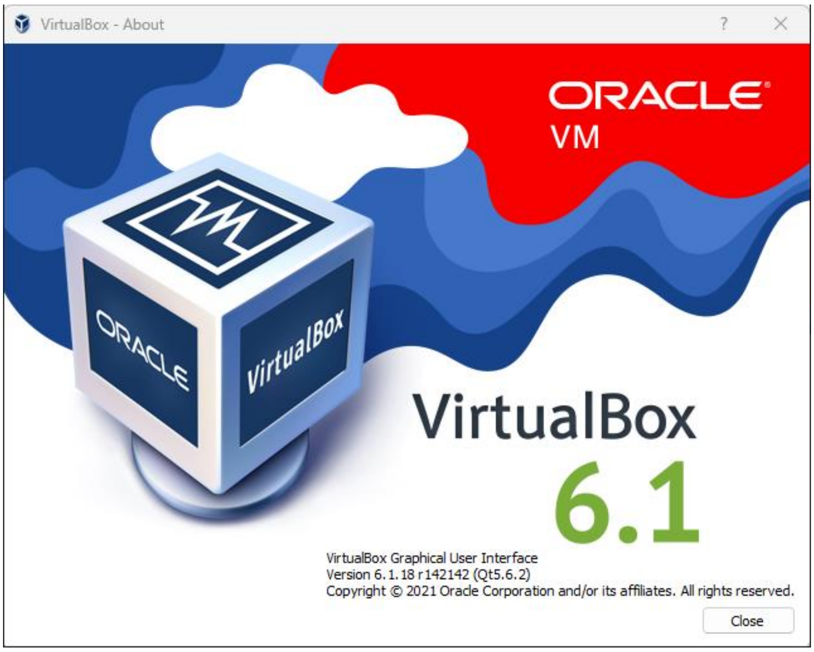


Fig 4.5: VirtualBox

After installing VirtualBox, the next step is to download the Ubuntu Server. For this project an .iso image file with a name like ubuntu-20.04.1-live-server-amd64.iso is used show in fig 4.6.

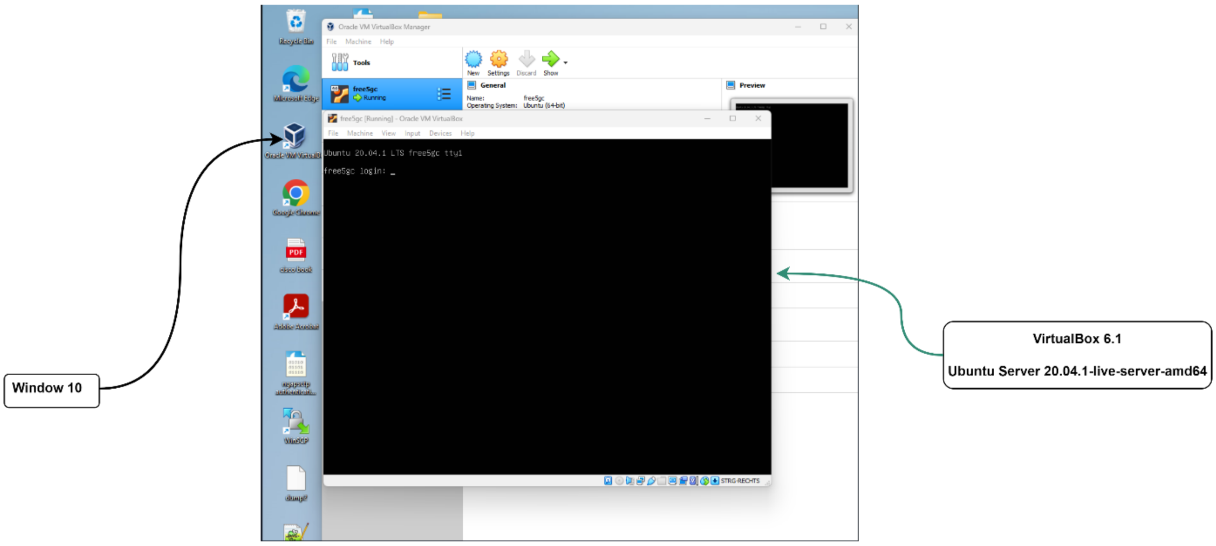


Fig 4.6: Ubuntu Server 20.04.1 LTS

In this project, Bridge Adapter is utilized as shown in Fig 4.7. This is because VM can interact with another device on the network, including the Host PC and it can also verify whether two PCs are communicating or not by using ping and tcpdump.

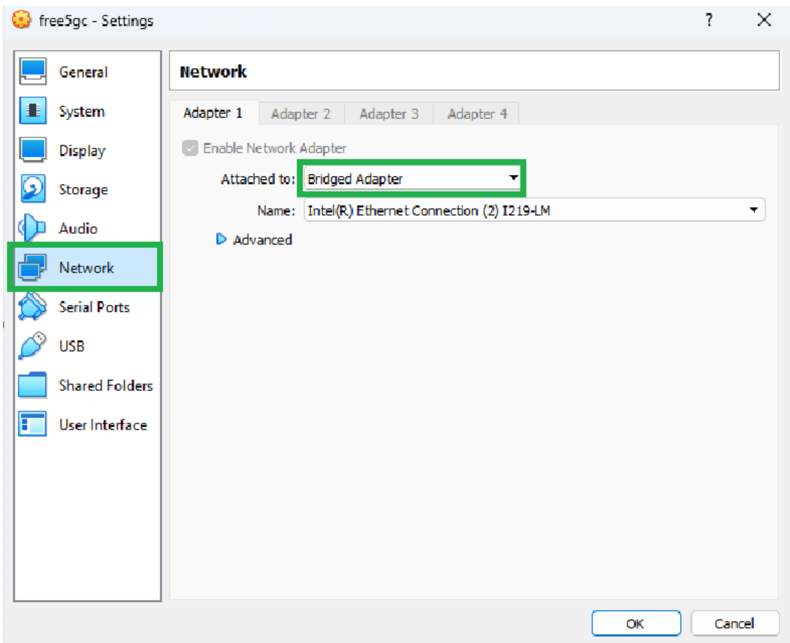


Fig 4.7 Bridge Adapter

After starting VM, firstly try to run the command "ifconfig" to check the network configuration. If "ifconfig" is not installed, it can show an error which stating at 'Command 'ifconfig' not found, but can be installed with: "sudo apt install net-tools"'. This command is used to install the net-tools package on ubuntu based Linux distributions.

This package includes some important networking tools which are used for network troubleshooting and configuration tasks. These net-tools package includes tools such as: ifconfig, netstat, arp, route. Running "ifconfig" again and it will now display the network interface information as shown in Fig. 4.3 of this report. At the end check if the VM has internet access or not by using this command "ping google.com"

Use the command i.e., "sudo apt update" for update the list of available packages and their versions. Another command is "sudo apt upgrade" which will upgrade all the installed packages to the newest available versions according to the updated package lists. [But for my recommendation don't upgrade every time.]

2) <u>Changing the hostname and hosts</u>:

After setting up the Ubuntu VM, the default hostname will be "ubuntu". To rename the hostname to "free5gc". For that nano or vi text editor is used to executing the command "sudo nano /etc/hostname". Once the file is open in nano or vi, replace the default hostname with 'free5gc', save the file (Ctrl-s) and exit (Ctrl-x) for the nano editor. After this don't forget to change in hosts also the command is "sudo name /etc/hosts", over there remove the default hosts "ubuntu to free5gc" shown in fig 4.8 and fig 4.8 (a).

```
ubuntu@free5gc:/etc$ sudo nano hostname
[sudo] password for ubuntu:
ubuntu@free5gc:/etc$ more hostname
free5gc
```

Fig 4.8 Change in hostname

```
ubuntu@free5gc:/etc$ sudo nano hosts
ubuntu@free5gc:/etc$ more hosts
127.0.0.1 localhost
127.0.1.1 free5gc

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Fig 4.8(a) change in hosts

3) <u>Building, Installation and Testing of Free5gc</u>:

In Fig 4.9 shows different commands:

- "lsb_release -a" command used to display the LSB (Linux Standard Base) information about the Ubuntu distribution.
- The command "uname -a" used in OS to display all available system information. It provides some details such as Kernel name, Kernel release, Kernel version, and Operating system.
- The command "uname -r" just reiterates the kernel release version.

```
ubuntu@free5gc:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 20.04.1 LTS
Release:        20.04
Codename:       focal
ubuntu@free5gc:~$ uname -a
Linux free5gc 5.4.0-170-generic #188-Ubuntu SMP Wed Jan 10 09:51:01 UTC 2024 x86_64 x86_64 x86_64 GNU/Linux
ubuntu@free5gc:~$ uname -r
5.4.0-170-generic
```

Fig 4.9 Linux Operating System

Free5gc is built and tested with Golang, so to check whether Go is in system, use the command “go version” in a command prompt. If another version of Go is installed, then remove the existing version and install Go lang again.

Commands for removing the current version of Go: “sudo rm -rf /usr/local/go”

If Go is not installed on system, just follow the commands:

```
wget https://dl.google.com/go/go1.18.10.linux-amd64.tar.gz
sudo tar -C /usr/local -zxvf go1.18.10.linux-amd64.tar.gz
mkdir -p ~/go/{bin,pkg,src}
# The following assume that your shell is bash:
echo 'export GOPATH=$HOME/go' >> ~/.bashrc
echo 'export GOROOT=/usr/local/go' >> ~/.bashrc
echo 'export PATH=$PATH:$GOPATH/bin:$GOROOT/bin' >> ~/.bashrc
echo 'export GO111MODULE=auto' >> ~/.bashrc
source ~/.bashrc
```

Now, again run the command “go version” shown in Fig 4.10

```
ubuntu@free5gc:/$ go version
go version go1.18.10 linux/amd64
```

Fig 4.10 Go Version

```
sudo apt -y update
sudo apt -y install mongodb wget git
sudo systemctl start mongodb
```

Fig 4.11 Installation of MongoDB

Fig 4.11 shows the commands that are supporting packages for the control plane. Where the commands are listed for updating the packages, installing MongoDB, and then starting the MongoDB.

```
sudo apt -y update
sudo apt -y install git gcc g++ cmake autoconf libtool pkg-config libmnl-dev libyaml-dev
```

Fig 4.12 Packages supporting by User-Plane

```
sudo sysctl -w net.ipv4.ip_forward=1
sudo iptables -t nat -A POSTROUTING -o <dn_interface> -j MASQUERADE
sudo iptables -A FORWARD -p tcp -m tcp --tcp-flags SYN,RST SYN -j TCPMSS --set-mss 1400
sudo systemctl stop ufw
sudo systemctl disable ufw # prevents the firewall to wake up after a OS reboot
```

Fig 4.13 Configuration of Linux Host Network Settings

Fig 4.13 shows the command list which are used for configuring network settings on a Linux system. The first command enables IP forwarding in the kernel, which allows the Linux system to forward packets. The second command iptables appends a rule to the POSTROUTING chain of the nat table to masquerade IP addresses. Replace <dn_interface> with the name of the outbound network interface, for example (enp0s3, enp0s8).

In third command, the TCP Maximum Segment Size (MSS) for connection to prevent issues with packet fragmentation. Fourth and Fifth command is to stop the uncomplicated Firewall(ufw) and disables ufw because it won't start automatically on system boot.

<u>Control Plane Elements Installation</u>:

The latest stable build version for free5gc is v 3.3.0

To install the latest stable version, need to follow commands mentioned in fig 4.14.

```
cd ~
git clone --recursive -b v3.3.0 -j `nproc` https://github.com/free5gc/free5gc.git
cd free5gc
```

Fig 4.14: Free5gc latest stable version

The below command is used to compile the AMF (Access and Mobility Management Function) and it is also used to build process for the AMF, command is shown below fig 4.15

```
cd ~/free5gc
make amf
```

Fig 4.15 Make AMF

The command to build all the network functions is given below fig 4.16.

```
cd ~/free5gc
make
```

Fig 4.16 make to build all the network functions.

User Plane Function (UPF) Installation:

Again, run the command "uname -r" to verify the version. By using "git" to download the 5G GTP-U kernel module. Fig 4.17

```
git clone -b v0.8.3 https://github.com/free5gc/gtp5g.git
cd gtp5g
make
sudo make install
```

Fig 4.17 GTP installation

After this command, build the UPF to build using "make upf" shown in fig 4.18.

```
cd ~/free5gc
make upf
```

Fig. 4.18 make upf

To run all the tests, first execute the commands which need to be run which are shown in fig 4.19. Simultaneously, start the Wireshark and apply the filter 'pfcp||icmp||gtp'.

```
cd ~/free5gc
make upf
chmod +x ./test.sh
```

Fig 4.19 Test Run

All test commands are to be executed within “~/free5gc”. Following are the commands:

- TestRegistration : ./test.sh TestRegistration
- TestGUTIRegistration: ./test.sh TestGUTIRegistration
- TestServiceRequest: ./test.sh TestServiceRequest
- TestXnHandover: ./test.sh TestXnHandover
- TestDeregistration: ./test.sh TestDeregistration
- TestPDUSessionReleaseRequest: ./test.sh TestPDUSessionReleaseRequest
- TestPaging: ./test.sh TestPaging
- TestN2Handover: ./test.sh TestN2Handover
- TestNon3GPP: ./test.sh TestNon3GPP
- TestReSynchronization: ./test.sh TestReSynchronization
- TestULCL: ./test_ulcl.sh TestRequestTwoPDUSessions

Installation of Free5GC WebConsole:

Free5GC WebConsole is to help create and manage UE registrations to be used by various 5G NFs. So, to build WebConsole first need Node.js and Yarn.

Before building WebConsole, install nodejs and yarn packages by following commands fig 4.20.

```
sudo apt remove cmdtest
sudo apt remove yarn
curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -
echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee /etc/apt/sources.list.d/yarn.list
sudo apt update
sudo apt install -y nodejs yarn
```

Fig 4.20 installation of nodejs and yarn packages

To build WebConsole, fig 4.21:

```
cd ~/free5gc
make webconsole
```

Fig 4.21: Build WebConsole

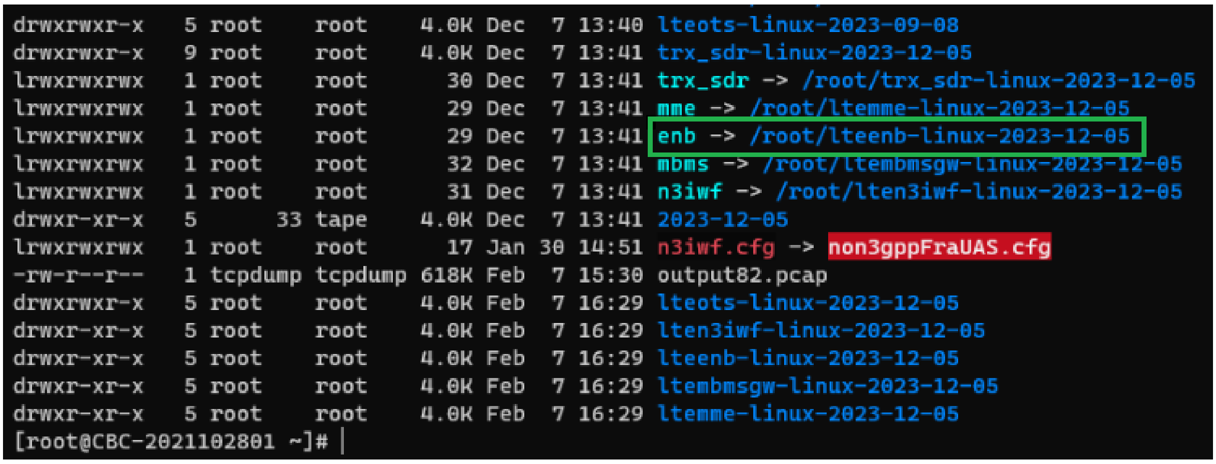
To use WebConsole and to add a UE, first this command needs to be run fig 4.22. After this command, port number will generate ":5000" at the end. Kindly ref to listing 4.3.

```
cd ~/free5gc/webconsole
go run server.go
```

Fig 4.22 Start up the WebConsole server.

[Ref. https://free5gc.org/guide/]

In Amari Callbox Classic, amarisoft was pre-installed and Fig 4.23 shows the version of LTEENB:



```
drwxrwxr-x   5 root    root    4.0K Dec  7 13:40 lteots-linux-2023-09-08
drwxrwxr-x   9 root    root    4.0K Dec  7 13:41 trx_sdr-linux-2023-12-05
lrwxrwxrwx   1 root    root      30 Dec  7 13:41 trx_sdr -> /root/trx_sdr-linux-2023-12-05
lrwxrwxrwx   1 root    root      29 Dec  7 13:41 mme -> /root/ltemme-linux-2023-12-05
lrwxrwxrwx   1 root    root      29 Dec  7 13:41 enb -> /root/lteenb-linux-2023-12-05
lrwxrwxrwx   1 root    root      32 Dec  7 13:41 mbms -> /root/ltembmsgw-linux-2023-12-05
lrwxrwxrwx   1 root    root      31 Dec  7 13:41 n3iwf -> /root/lten3iwf-linux-2023-12-05
drwxr-xr-x   5      33 tape    4.0K Dec  7 13:41 2023-12-05
lrwxrwxrwx   1 root    root      17 Jan 30 14:51 n3iwf.cfg -> non3gppFraUAS.cfg
-rw-r--r--   1 tcpdump tcpdump 618K Feb  7 15:30 output82.pcap
drwxrwxr-x   5 root    root    4.0K Feb  7 16:29 lteots-linux-2023-12-05
drwxr-xr-x   5 root    root    4.0K Feb  7 16:29 lten3iwf-linux-2023-12-05
drwxr-xr-x   5 root    root    4.0K Feb  7 16:29 lteenb-linux-2023-12-05
drwxr-xr-x   5 root    root    4.0K Feb  7 16:29 ltembmsgw-linux-2023-12-05
drwxr-xr-x   5 root    root    4.0K Feb  7 16:29 ltemme-linux-2023-12-05
[root@CBC-2021102801 ~]#
```

Fig 4.23: LTEENB Version

4.3 Configuration

Configuration of Free5GC:

After launching the Free5GC VM, it needs to be configured to communicate with the Amari Callbox Classic (gNB). To achieve this, follow the path outlined in Figure 4.24. Upon reaching the configuration location, there are various '.yaml' files.

```
ubuntu@free5gc:~/free5gc/config$ ls -lrth
total 124K
-rw-rw-r-- 1 ubuntu ubuntu  473 Dec  7 14:26 webuicfg.yaml
-rw-rw-r-- 1 ubuntu ubuntu 1.1K Dec  7 14:26 upfcfg.test.yaml
-rw-rw-r-- 1 ubuntu ubuntu  872 Dec  7 14:26 udrcfg.yaml
-rw-rw-r-- 1 ubuntu ubuntu 1.7K Dec  7 14:26 udmcfg.yaml
-rw-rw-r-- 1 ubuntu ubuntu 2.1K Dec  7 14:26 pcfcfg.yaml
-rw-rw-r-- 1 ubuntu ubuntu 2.9K Dec  7 14:26 n3iwfcfg.yaml
-rw-rw-r-- 1 ubuntu ubuntu 2.9K Dec  7 14:26 n3iwfcfg.test.yaml
drwxrwxr-x 2 ubuntu ubuntu 4.0K Dec  7 14:26 multiUPF
-rw-rw-r-- 1 ubuntu ubuntu 1.1K Dec  7 15:03 upfcfg.testulcl.yaml
-rw-rw-r-- 1 ubuntu ubuntu 1.1K Dec  7 15:28 upfcfg.yaml
-rw-rw-r-- 1 ubuntu ubuntu 1.2K Dec 18 16:11 nrfcfg.yaml
-rw-rw-r-- 1 ubuntu ubuntu 2.0K Dec 18 16:15 uerouting.yaml
-rw-rw-r-- 1 ubuntu ubuntu  23K Dec 20 18:07 nssfcfg.yaml.save
-rw-rw-r-- 1 ubuntu ubuntu 1.5K Jan 19 11:48 ausfcfg.yaml
-rw-rw-r-- 1 ubuntu ubuntu 5.0K Jan 22 14:51 smfcfg.yaml
-rw-rw-r-- 1 ubuntu ubuntu 6.4K Jan 24 17:06 amfcfg.yaml
-rwxrwxr-x 1 ubuntu ubuntu 7.4K Jan 25 13:36 ue_3GPP.sh
-rw-rw-r-- 1 ubuntu ubuntu  23K Jan 25 14:58 nssfcfg.yaml
ubuntu@free5gc:~/free5gc/config$
```

Fig: 4.24 Free5GC.yaml list

In this project, the "amfcfg.yaml" file is initially configured because it communicate with gNB of Amari Callbox Classic via N2 interface. AMF (Access Mobility Management Function) which is also responsible for the management of mobility, authentication, and security. Fig 4.25 below illustrates the modification to the "amfcfg.yaml" file.

In "amfcfg.yaml" the ngapIpList is set to 192.168.50.56 (which is the IP address of Free5GC) and ngapPort: 38412. The PLMN as 00101 given by Amarisoft test sim card and the IMSI number was 001010123456789. The TAC is entered via the Amarisoft WebGUI for NGAP request and response i.e., 000064. The report focuses on a single network slicing.

```
GNU nano 4.8                                                                                        amfcfg.yaml
info:
  version: 1.0.9
  description: AMF initial local configuration

configuration:
  amfName: AMF # the name of this AMF
  ngapIpList:  # the IP list of N2 interfaces on this AMF
    - 192.168.50.56 # 127.0.0.18
  ngapPort: 38412 # the SCTP port listened by NGAP
  sbi: # Service-based interface information
    scheme: http # the protocol for sbi (http or https)
    registerIPv4: 127.0.0.18 # IP used to register to NRF
    bindingIPv4: 127.0.0.18  # IP used to bind the service
    port: 8000 # port used to bind the service
    tls: # the local path of TLS key
      pem: cert/amf.pem # AMF TLS Certificate
      key: cert/amf.key # AMF TLS Private key
  serviceNameList: # the SBI services provided by this AMF, refer to TS 29.518
    - namf-comm # Namf_Communication service
    - namf-evts # Namf_EventExposure service
    - namf-mt   # Namf_MT service
    - namf-loc  # Namf_Location service
    - namf-oam  # OAM service
  servedGuamiList: # Guami (Globally Unique AMF ID) list supported by this AMF
    # <GUAMI> = <MCC><MNC><AMF ID>
    - plmnId: # Public Land Mobile Network ID, <PLMN ID> = <MCC><MNC>
        mcc: 001 # 208 # Mobile Country Code (3 digits string, digit: 0~9)
        mnc: 01 # 93 # Mobile Network Code (2 or 3 digits string, digit: 0~9)
      amfId: 800101 # 03c410 # 00f110 # 800101 # cafe00 # AMF identifier (3 bytes hex string, range: 000000~FFFFFF)
  supportTaiList:  # the TAI (Tracking Area Identifier) list supported by this AMF
    - plmnId: # Public Land Mobile Network ID, <PLMN ID> = <MCC><MNC>
        mcc: 001 # 208 # Mobile Country Code (3 digits string, digit: 0~9)
        mnc: 01 # 93 # Mobile Network Code (2 or 3 digits string, digit: 0~9)
      tac: 000064 # 100 # 000001 # Tracking Area Code (3 bytes hex string, range: 000000~FFFFFF)
  plmnSupportList: # the PLMNs (Public land mobile network) list supported by this AMF
    - plmnId: # Public Land Mobile Network ID, <PLMN ID> = <MCC><MNC>
        mcc: 001 # 208 # Mobile Country Code (3 digits string, digit: 0~9)
        mnc: 01 # 93 # Mobile Network Code (2 or 3 digits string, digit: 0~9)
      snssaiList: # the S-NSSAI (Single Network Slice Selection Assistance Information) list supported by this AMF
        - sst: 1 # Slice/Service Type (uinteger, range: 0~255)
          sd: 000000 # Slice Differentiator (3 bytes hex string, range: 000000~FFFFFF)
#        - sst: 1 # Slice/Service Type (uinteger, range: 0~255)
#          sd: 112233 # Slice Differentiator (3 bytes hex string, range: 000000~FFFFFF)
  supportDnnList:  # the DNN (Data Network Name) list supported by this AMF
    - internet
  nrfUri: http://127.0.0.10:8000 # a valid URI of NRF
  security:  # NAS security parameters
    integrityOrder: # the priority of integrity algorithms
      - NIA2
      # - NIA0
    cipheringOrder: # the priority of ciphering algorithms
      - NEA0
      # - NEA2
```

Fig 4.25 amfcfg.yaml

Some configuration happened in smfcfg.yaml as well as upfcfg.yaml files. In “smfcfg.yaml” changes made in PLMN were modified to 00101 (mcc: 001 and mnc: 01) and also endpoint IP is also changed to IP address of Free5GC at N3 which works between UPF and gNB, which is shown in Fig 4.26.

For the “upfcfg.yaml” file i.e., Fig. 4.27, the IP address of Free5GC which is already mentioned. Gtpu protocol facilitates communication between the gNB of Amari and the UPF of 5G core network.

```yaml
  GNU nano 4.8                                                                          smfcfg.yaml
#       sst: 1 # Slice/Service Type (uinteger, range: 0~255)
#        sd: 112233 # Slice Differentiator (3 bytes hex string, range: 000000~FFFFFF)
#      dnnInfos: # DNN information list
#        - dnn: internet # Data Network Name
#          dns: # the IP address of DNS
#            ipv4: 8.8.8.8
#            ipv6: 2001:4860:4860::8888
  plmnList: # the list of PLMN IDs that this SMF belongs to (optional, remove this key when unnecessary)
    - mcc: 001 # 208 # Mobile Country Code (3 digits string, digit: 0~9)
      mnc: 01 # 93 # Mobile Network Code (2 or 3 digits string, digit: 0~9)
  locality: area1 # Name of the location where a set of AMF, SMF, PCF and UPFs are located
  pfcp: # the IP address of N4 interface on this SMF (PFCP)
    # addr config is deprecated in smf config v1.0.3, please use the following config
    nodeID: 127.0.0.1 # the Node ID of this SMF
    listenAddr: 127.0.0.1 # the IP/FQDN of N4 interface on this SMF (PFCP)
    externalAddr: 127.0.0.1 # the IP/FQDN of N4 interface on this SMF (PFCP)
  userplaneInformation: # list of userplane information
    upNodes: # information of userplane node (AN or UPF)
      gNB1: # the name of the node
        type: AN # the type of the node (AN or UPF)
      UPF: # the name of the node
        type: UPF # the type of the node (AN or UPF)
        nodeID: 127.0.0.8 # the Node ID of this UPF
        addr: 127.0.0.8 # the IP/FQDN of N4 interface on this UPF (PFCP)
        sNssaiUpfInfos: # S-NSSAI information list for this UPF
          - sNssai: # S-NSSAI (Single Network Slice Selection Assistance Information)
              sst: 1 # Slice/Service Type (uinteger, range: 0~255)
              sd: 000000 # 010203 # Slice Differentiator (3 bytes hex string, range: 000000~FFFFFF)
            dnnUpfInfoList: # DNN information list for this S-NSSAI
              - dnn: internet
                pools:
                  - cidr: 10.60.0.0/16
                staticPools:
                  - cidr: 10.60.100.0/24
#          - sNssai: # S-NSSAI (Single Network Slice Selection Assistance Information)
#              sst: 1 # Slice/Service Type (uinteger, range: 0~255)
#              sd: 112233 # Slice Differentiator (3 bytes hex string, range: 000000~FFFFFF)
#            dnnUpfInfoList: # DNN information list for this S-NSSAI
#              - dnn: internet
#                pools:
#                  - cidr: 10.61.0.0/16
#                staticPools:
#                  - cidr: 10.61.100.0/24
        interfaces: # Interface list for this UPF
          - interfaceType: N3 # the type of the interface (N3 or N9)
            endpoints: # the IP address of this N3/N9 interface on this UPF
              - 192.168.50.56 # 127.0.0.8
            networkInstances:  # Data Network Name (DNN)
              - internet
    links: # the topology graph of userplane, A and B represent the two nodes of each link
      - A: gNB1
        B: UPF
```

Fig. 4.26 smfcfg.yaml

```yaml
  GNU nano 4.8                                                                          upfcfg.yaml
version: 1.0.3
description: UPF initial local configuration

# The listen IP and nodeID of the N4 interface on this UPF (Can't set to 0.0.0.0)
pfcp:
  addr: 127.0.0.8   # IP addr for listening
  nodeID: 127.0.0.8 # External IP or FQDN can be reached
  retransTimeout: 1s # retransmission timeout
  maxRetrans: 3 # the max number of retransmission

gtpu:
  forwarder: gtp5g
  # The IP list of the N3/N9 interfaces on this UPF
  # If there are multiple connection, set addr to 0.0.0.0 or list all the addresses
  ifList:
    - addr: 192.168.50.56 # 127.0.0.8
      type: N3
      # name: upf.5gc.nctu.me
      # ifname: gtpif
      # mtu: 1400

# The DNN list supported by UPF
dnnList:
  - dnn: internet # Data Network Name
    cidr: 10.60.0.0/24 # Classless Inter-Domain Routing for assigned IPv4 pool of UE
    # natifname: eth0

logger: # log output setting
  enable: true # true or false
  level: info # how detailed to output, value: trace, debug, info, warn, error, fatal, panic
  reportCaller: false # enable the caller report or not, value: true or false
```

Fig 4.27 upfcfg.yaml

In the file of “ausfcfg.yaml”, just need to change the PLMN to 00101. The Fig 4.28 is given below:

```
  GNU nano 4.8                                                                                      ausfcfg.yaml
info:
  version: 1.0.3
  description: AUSF initial local configuration

configuration:
  sbi: # Service-based interface information
    scheme: http # the protocol for sbi (http or https)
    registerIPv4: 127.0.0.9 # IP used to register to NRF
    bindingIPv4: 127.0.0.9  # IP used to bind the service
    port: 8000 # Port used to bind the service
    tls: # the local path of TLS key
      pem: cert/ausf.pem # AUSF TLS Certificate
      key: cert/ausf.key # AUSF TLS Private key
  serviceNameList: # the SBI services provided by this AUSF, refer to TS 29.509
    - nausf-auth # Nausf_UEAuthentication service
  nrfUri: http://127.0.0.10:8000 # a valid URI of NRF
  plmnSupportList: # the PLMNs (Public Land Mobile Network) list supported by this AUSF
    - mcc: 001 # 208 # Mobile Country Code (3 digits string, digit: 0~9)
      mnc: 01 # 93  # Mobile Network Code (2 or 3 digits string, digit: 0~9)
    - mcc: 001 # 123 # Mobile Country Code (3 digits string, digit: 0~9)
      mnc: 01 # 45 # Mobile Network Code (2 or 3 digits string, digit: 0~9)
  groupId: ausfGroup001 # ID for the group of the AUSF
  eapAkaSupiImsiPrefix: true # false # including "imsi-" prefix or not when using the SUPI to do EAP-AKA' authentication

logger: # log output setting
  enable: true # true or false
  level: info # how detailed to output, value: trace, debug, info, warn, error, fatal, panic
  reportCaller: false # enable the caller report or not, value: true or false
```

Fig 4.28 ausfcfg.yaml

<u>The configuration file of gNB Amari Callbox Classic</u>:

In the Amari Callbox Classic, the default file used by the LTE automatic service for configuring the eNB and gNB is named enb.cfg. Both eNB and gNB components are integrated within the same module, referred to as enb. Screenshots are provided below as Figure 4.29, which shows the path, and Figure 4.30, illustrating the use of the command ln -sfn <config file> enb.cfg to create a soft link. This is necessary for making any changes or updates to the soft link file.

```
[root@CBC-2021102801 ~]# cd enb/config/
[root@CBC-2021102801 config]# ls
drb.cfg              enb.cfg              enb-nbiot-standalone.cfg  gnb-pscell-change.cfg       gnbTEST.cfg                    rf_select.sh       sib2_3_nosrs.asn  sib4_nb.asn                 test2x2_20mhz.cfg
drb_nb.cfg           enb.cfg.bak          enb-tm-pdsch.cfg          gnb-sa.cfg                  gnb-tm-pdsch.cfg               sam.enb.cfg        sib24.asn         sib4_nr.asn
drb_nr.cfg           enb.default.cfg      gnb-2cc-nsa.cfg           gnb-sa-ho.backup.cfg        meas_config_a3_3299.asn        sdr                sib2_nb.asn       sib5.asn
enb-2cc.cfg          enb.default.cfg.bak  gnb-2cc-sa.cfg            gnb-sa-ho.cfg               meas_config_a3_3350.asn        sib15.asn          sib2_nb_ce.asn    sib5_nb.asn
enb-2cell-ho.cfg     enbMarius.cfg        gnb-dss.cfg               gnb-sa-lte-ho.cfg           meas_config_periodic.asn       sib1.asn           sib2_nr.asn       sib5_nr.asn
enb-3cc.cfg          enb-mbms.cfg         gnb-nr-dc.cfg             gnb-sa-ntn.cfg              meas_config_periodic_nr.asn    sib2_3.asn         sib3_nb.asn       sib6.asn
enb-catm1.cfg        enb-nbiot.cfg        gnb-nsa.cfg               gnb-sa-redcap.cfg           Nukovic_Test.cfg               sib2_3_br.asn      sib3_nr.asn       sib7.asn
enb-catm1-nbiot.cfg  enb-nbiot-ntn.cfg    gnb-nsa-ho-lte.cfg        gnb-sa-tdd-low-latency.cfg  rf_driver                      sib2_3_br_ce.asn   sib4.asn          test2x2_20mhz_2Cells.cfg
[root@CBC-2021102801 config]#
```

Fig 4.29 Path for config in gNB

```
-rw-r--r-- 1 root root 9.3K Dec 21 14:49 enbMarius.cfg
-rw-r--r-- 1 root root 9.4K Jan 18 16:51 sam.enb.cfg
lrwxrwxrwx 1 root root   11 Feb  6 17:25 enb.cfg -> sam.enb.cfg
[root@CBC-2021102801 config]#
```

Fig 4.30 soft link (ln -sfn <sam.enb.cfg>enb.cfg)

The configuration is made in gNB “sam.enb.cfg” in amf list and gtp_addr which is shown in fig 4.31. Here, the amf_addr: 192.168.50.56 is Free5gc IP address and gtp_addr: 192.168.50.20 is the IP address of Amaris Callbox Classic.

```
amf_list: [
  {
    /* address of AMF for NGAP connection. Must be modified if the AMF runs on a different host. */
      // amf_addr: "127.0.1.100",
     amf_addr: "192.168.50.56",
  },
],
/* GTP bind address (=address of the ethernet interface connected to
   the AMF). Must be modified if the AMF runs on a different host. */
        // gtp_addr: "127.0.1.1",
      gtp_addr: "192.168.50.20",

gnb_id_bits: 28,
gnb_id: 0x12345,
```

Fig 4.31 sam.enb.cfg

In this project some configurations are also made in mme file also: it has the same path “cd /mme/config/”, shown in fig 4.32.

```
lrwxrwxrwx 1 root root   15 Jan 25 12:22 ims.cfg -> ims.default.cfg
-rw-r--r-- 1   33 tape 3.9K Jan 31 13:09 ue_db-ims.cfg
-rw-r--r-- 1 root root    4 Feb  7 16:29 lte_ue_ims.db
lrwxrwxrwx 1 root root   11 Feb  8 20:11 mme.cfg -> sam-mme.cfg
[root@CBC-2021102801 config]#
```

Fig 4.32 sam-mme.cfg

The configuration is made in mme “sam-mme.cfg” in gtp_addr using N12 interface of Free5gc ausfcfg.yaml which is shown in fig 4.33. [Ref. https://tech-academy.amarisoft.com/MME_Open5GS.html (Amarisoft tech academy)

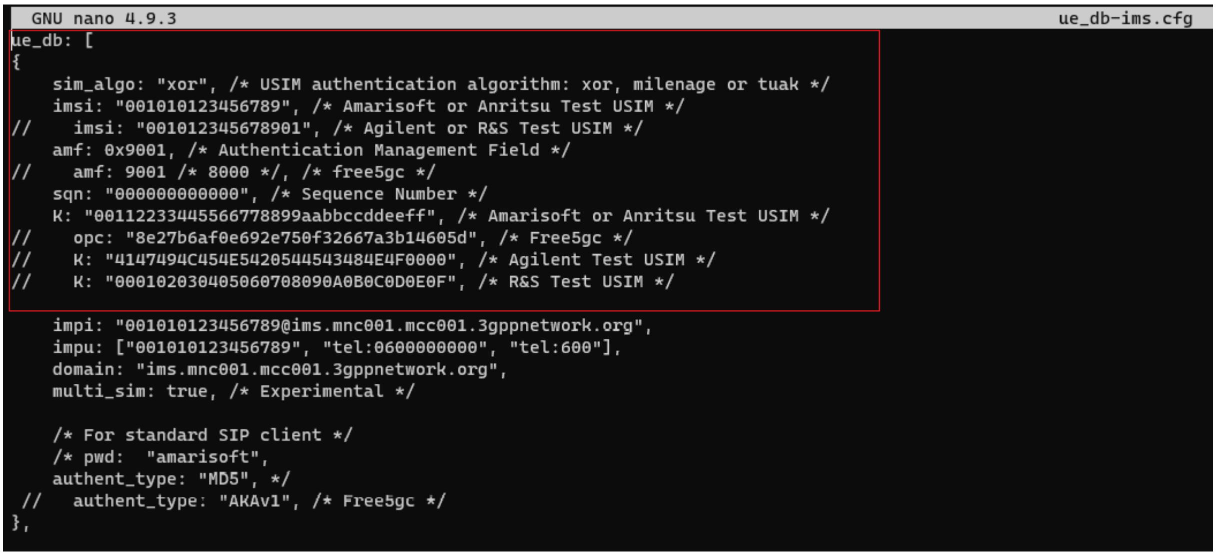
```
/* bind address for GTP-U. Normally = address of the PC, here bound
   on local interface to be able to run ltemme on the same PC as
   lteenb. By default, the S1AP SCTP connection is bound on the same
   address. */
 gtp_addr: "127.0.1.100",
 n12: {
       api_root:"http://192.168.50.56:8000",
       bind_addr: "192.168.50.20",
      },
```

Fig 4.33 sam-mme.cfg

Within the system, there is a UE database the file name is “ue_db-ims.cfg”. If the test sim card provided with the Amari callbox is used, there is no need for any configuration (no customization is required).

However, if other test SIM cards with different IMSI or secret key values are used, they must be declared in the EPC database (HSS).

In this project, Amarisoft test sim card was used, shown in Fig 4.34.



```
GNU nano 4.9.3                                                        ue_db-ims.cfg
ue_db: [
{
    sim_algo: "xor", /* USIM authentication algorithm: xor, milenage or tuak */
    imsi: "001010123456789", /* Amarisoft or Anritsu Test USIM */
//    imsi: "001012345678901", /* Agilent or R&S Test USIM */
    amf: 0x9001, /* Authentication Management Field */
//    amf: 9001 /* 8000 */, /* free5gc */
    sqn: "000000000000", /* Sequence Number */
    K: "00112233445566778899aabbccddeeff", /* Amarisoft or Anritsu Test USIM */
//    opc: "8e27b6af0e692e750f32667a3b14605d", /* Free5gc */
//    K: "4147494C454E5420544543484E4F0000", /* Agilent Test USIM */
//    K: "000102030405060708090A0B0C0D0E0F", /* R&S Test USIM */

    impi: "001010123456789@ims.mnc001.mcc001.3gppnetwork.org",
    impu: ["001010123456789", "tel:0600000000", "tel:600"],
    domain: "ims.mnc001.mcc001.3gppnetwork.org",
    multi_sim: true, /* Experimental */

    /* For standard SIP client */
    /* pwd:  "amarisoft",
    authent_type: "MD5", */
 //   authent_type: "AKAv1", /* Free5gc */
},
```

Fig 4.34 UE database

UE Configuration:

In internet setting for connecting device to internet, APN setting must be configured at UE side to match the callbox setting:

So, in this project, the smartphone which is used: One plus 8T. The following parameters need to be followed:

- Go to settings/ Mobile network/select SIM 1or SIM 2 if sim is available in that slot.
- Turn on Data roaming.
- Go back to mobile network.
- Select APN (Access point names), add the first APN with following parameter:
  - I. Name = Internet
  - II. APN = internet
  - III. MCC = 001
  - IV. MNC = 01
  - V. APN type = internet, default
- Save it and select it.
- Reboot smartphone.

[Ref: https://www.amarisoft.com/app/uploads/2022/03/userguide_callbox_classic.pdf]

Configuration of Free5gc WebGUI:

After accessing the Free5GC WebGUI, subscribers need to be edited. Figure 4.35 below illustrates how to edit a subscriber. The subscriber details should match exactly those in the ue_db-ims.cfg configuration file. Any updates should be made in the configuration file and the WebGUI.

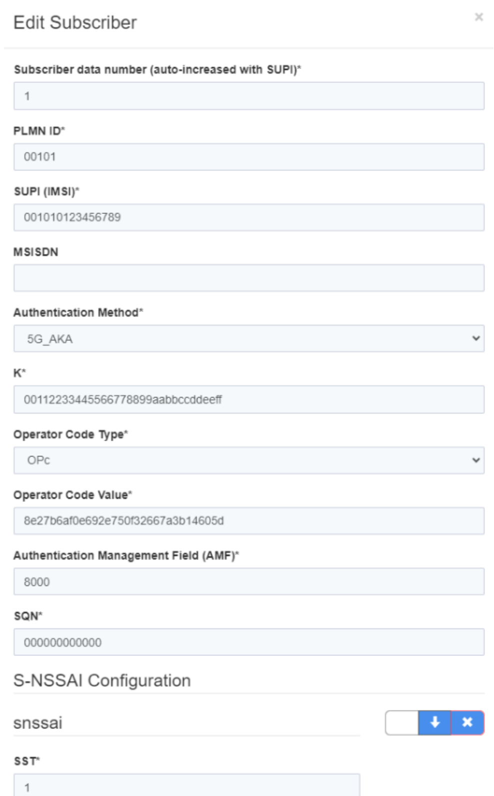


Fig 4.35 Free5GC WebGUI

Configuration of SIM card:

In this project, the Amari callbox utilizes the SIM algorithm "XOR," while Free5GC employs "Milenage." With the aid of a SIM card reader, the SIM can be configured or modified in terms of parameters such as OPC, K, IMSI, and SIM algorithm, etc., to ensure seamless operation.

During this project, an additional Sysmocom test SIM card was provided. It was necessary to configure all the parameters to facilitate authentication between the UE (User Equipment) and the 5GC (5G Core).

To configure the SIM card, an Ubuntu desktop with at least 2GB of RAM is required. For installation instructions, follow this GitHub link: [https://github.com/osmocom/pysim] and additional information can be found on the Tech Academy Amarisoft page [https://tech-academy.amarisoft.com/update.wiki]

[**Note**: Currently, this configuration process is an ongoing requirement in the project. Figure 4.36 depicts the SIM card reader and the test SIM card.]

Fig 4.36 SIM card reader and sysmocom test sim card

SIM Card specification:

- IMSI: 999700000082762
- ICCIS: 8988211000000827629
- ACC: 0004
- PIN1:6172
- PUK 1: 97072069
- PIN2: 7711
- PUK2: 94286976
- Ki= C1EC7A2DA67906928B5F829085D12639
- OPC: 69A4DB5174BC0A2AB538B9F123BDDEAA
- ADM1: 99098579

4.4 Observation

After configuring all the parameters in Free5GC and Amari callbox classic, restart both VM's i.e., Free5GC and Amari callbox.

After opening Free5GC VM via SSH:

- In Free5GC just need to follow the commands which are mentioned in fig 4.37. These commands have already been explained.

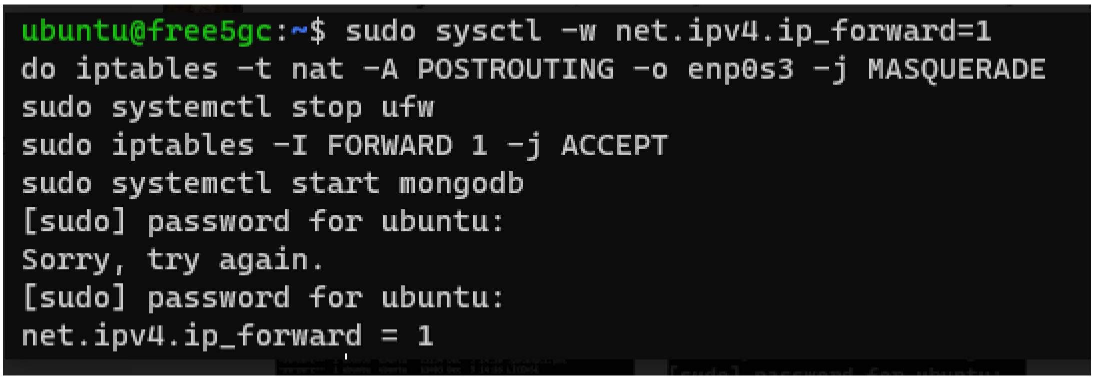
```
sudo sysctl -w net.ipv4.ip_forward=1
sudo iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE
sudo systemctl stop ufw
sudo iptables -I FORWARD 1 -j ACCEPT
sudo systemctl start mongodb
```

Fig 4.37 test command to run Free5GC.

- After this command, build the UPF to build using “make upf” shown in fig 4.38.

```
cd ~/free5gc
make upf
```

Fig4.38 make upf



```
ubuntu@free5gc:~$ sudo sysctl -w net.ipv4.ip_forward=1
do iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE
sudo systemctl stop ufw
sudo iptables -I FORWARD 1 -j ACCEPT
sudo systemctl start mongodb
[sudo] password for ubuntu:
Sorry, try again.
[sudo] password for ubuntu:
net.ipv4.ip_forward = 1
```

Fig 4.39 [combination of fig 4.37 and 4.38]

- Execute the run script: This involves running the free5gc by using the command "**./run.sh**", as depicted in Figure 4.41. The "run.sh" command is located within the directory illustrated in Figure 4.40.

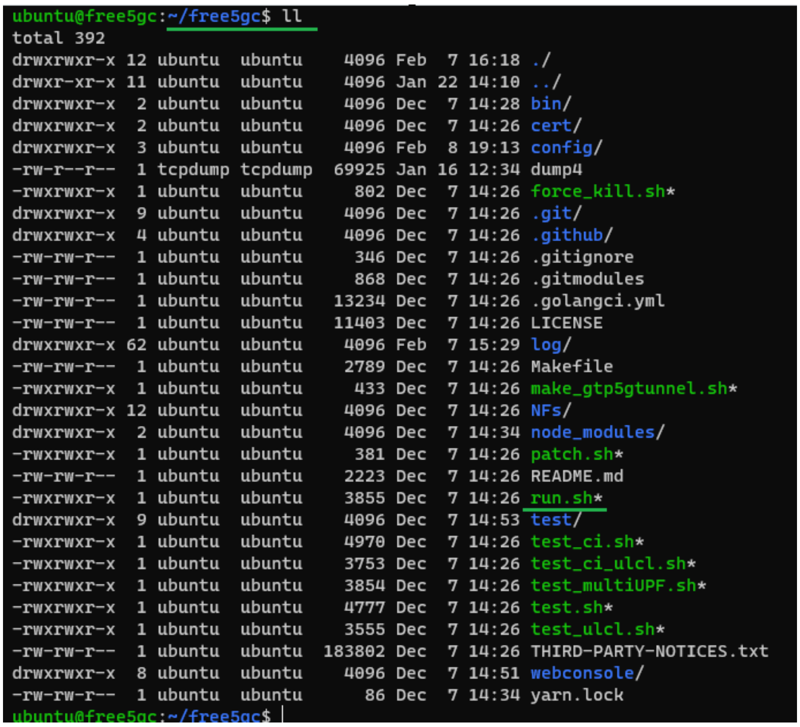


Fig 4.40 run.sh inside Free5GC

Fig 4.41./run.sh

- After running this command, the logs will appear that provide details record of the operations and status message from various network function within the free5gc framework.

The logs are given in screenshot in fig 4.42.

```
ubuntu@free5gc:~/free5gc$ ./run.sh
log path: ./log/20240209_123804/
2024-02-09T12:38:04.163501497Z [INFO][UPF][Main] UPF version:
        free5GC version: v3.3.0
        build time:      2023-12-07T14:28:35Z
        commit hash:     4474dc86
        commit time:     2023-06-08T03:37:39Z
        go version:      go1.18.10 linux/amd64
2024-02-09T12:38:04.164864008Z [INFO][UPF][CFG] Read config from [./config/upfcfg.yaml]
2024-02-09T12:38:04.165293562Z [INFO][UPF][CFG] ==================================================
2024-02-09T12:38:04.165370267Z [INFO][UPF][CFG] (*factory.Config)(0xc00048e190)({
        Version: (string) (len=5) "1.0.3",
        Description: (string) (len=31) "UPF initial local configuration",
        Pfcp: (*factory.Pfcp)(0xc0003da780)({
                Addr: (string) (len=9) "127.0.0.8",
                NodeID: (string) (len=9) "127.0.0.8",
                RetransTimeout: (time.Duration) 1s,
                MaxRetrans: (uint8) 3
        }),
        Gtpu: (*factory.Gtpu)(0xc0003dba10)({
                Forwarder: (string) (len=5) "gtp5g",
                IfList: ([]factory.IfInfo) (len=1 cap=1) {
                        (factory.IfInfo) {
                                Addr: (string) (len=13) "192.168.50.56",
                                Type: (string) (len=2) "N3",
                                Name: (string) "",
                                IfName: (string) "",
                                MTU: (uint32) 0
                        }
                }
        }),
        DnnList: ([]factory.DnnList) (len=1 cap=1) {
                (factory.DnnList) {
                        Dnn: (string) (len=8) "internet",
                        Cidr: (string) (len=12) "10.60.0.0/24",
                        NatIfName: (string) ""
                }
        },
        Logger: (*factory.Logger)(0xc000480640)({
                Enable: (bool) true,
                Level: (string) (len=4) "info",
                ReportCaller: (bool) false
        })
})
2024-02-09T12:38:04.170133026Z [INFO][UPF][CFG] ==================================================
2024-02-09T12:38:04.170186193Z [INFO][UPF][Main] Log level is set to [info]
2024-02-09T12:38:04.170277974Z [INFO][UPF][Main] Report Caller is set to [false]
2024-02-09T12:38:04.170811809Z [INFO][UPF][Main] starting Gtpu Forwarder [gtp5g]
2024-02-09T12:38:04.170888262Z [INFO][UPF][Main] GTP Address: "192.168.50.56:2152"
2024-02-09T12:38:04.189346468Z [INFO][UPF][BUFF] buff netlink server started
2024-02-09T12:38:04.189382826Z [INFO][UPF][Perio] perio server started
2024-02-09T12:38:04.189398709Z [INFO][UPF][Gtp5g] Forwarder started
2024-02-09T12:38:04.190276160Z [INFO][UPF][PFCP][LAddr:127.0.0.8:8805] starting pfcp server
2024-02-09T12:38:04.190304002Z [INFO][UPF][PFCP][LAddr:127.0.0.8:8805] pfcp server started
2024-02-09T12:38:04.190320517Z [INFO][UPF][Main] UPF started
MongoDB shell version v3.6.8
connecting to: mongodb://127.0.0.1:27017/free5gc
```

```
Implicit session: session { "id" : UUID("5f18bb5a-91df-401a-a28b-8d566004271a") }
MongoDB server version: 3.6.8
false
./run.sh: line 112: mongosh: command not found
2024-02-09T12:38:04.393704052Z [INFO][NRF][Main] NRF version:
        free5GC version: v3.3.0
        build time:      2023-12-07T14:27:45Z
        commit hash:     db4c0f90
        commit time:     2023-05-11T08:14:48Z
        go version:      go1.18.10 linux/amd64
2024-02-09T12:38:04.395273152Z [INFO][NRF][CFG] Read config from [./config/nrfcfg.yaml]
2024-02-09T12:38:04.396072448Z [INFO][NRF][Main] Log enable is set to [true]
2024-02-09T12:38:04.396159767Z [INFO][NRF][Main] Log level is set to [info]
2024-02-09T12:38:04.396271905Z [INFO][NRF][Main] Report Caller is set to [false]
2024-02-09T12:38:04.396514128Z [INFO][NRF][Init] nrfconfig Info: Version[1.0.2] Description[NRF initial local configuration]
2024-02-09T12:38:04.396771940Z [INFO][NRF][Init] Server starting
2024-02-09T12:38:04.397396073Z [INFO][NRF][Init] Binding addr: [127.0.0.10:8000]
2024-02-09T12:38:04.480268724Z [INFO][AMF][Main] AMF version:
        free5GC version: v3.3.0
        build time:      2023-12-07T14:26:56Z
        commit hash:     7907d3c0
        commit time:     2023-05-20T15:04:00Z
        go version:      go1.18.10 linux/amd64
2024-02-09T12:38:04.483508131Z [INFO][AMF][CFG] Read config from [./config/amfcfg.yaml]
2024-02-09T12:38:04.485107275Z [INFO][AMF][Main] Log enable is set to [true]
2024-02-09T12:38:04.485177413Z [INFO][AMF][Main] Log level is set to [info]
2024-02-09T12:38:04.485438712Z [INFO][AMF][Main] Report Caller is set to [false]
2024-02-09T12:38:04.485580257Z [INFO][AMF][Util] amfconfig Info: Version[1.0.9]
2024-02-09T12:38:04.485860543Z [INFO][AMF][Init] Server started
2024-02-09T12:38:04.486170292Z [INFO][AMF][Util] amfconfig Info: Version[1.0.9]
2024-02-09T12:38:04.487228289Z [INFO][AMF][Ngap] Listen on 192.168.50.56:38412
2024-02-09T12:38:04.488462100Z [INFO][NRF][NFM] Handle NFRegisterRequest
2024-02-09T12:38:04.490061686Z [INFO][NRF][NFM] urilist update
2024-02-09T12:38:04.501828772Z [INFO][NRF][NFM] Create NF Profile
2024-02-09T12:38:04.503073833Z [INFO][NRF][NFM] Location header:  http://127.0.0.10:8000/nnrf-nfm/v1/nf-instances/f8a55f4f-568b-4d85-b041-e77688a23f59
2024-02-09T12:38:04.503455630Z [INFO][NRF][GIN] | 201 |       127.0.0.1 | PUT     | /nnrf-nfm/v1/nf-instances/f8a55f4f-568b-4d85-b041-e77688a23f59 |
2024-02-09T12:38:04.585451409Z [INFO][SMF][Main] SMF version:
        free5GC version: v3.3.0
        build time:      2023-12-07T14:28:08Z
        commit hash:     8eb6843b
        commit time:     2023-05-31T04:43:19Z
        go version:      go1.18.10 linux/amd64
2024-02-09T12:38:04.587152881Z [INFO][SMF][CFG] Read config from [./config/smfcfg.yaml]
2024-02-09T12:38:04.589204131Z [INFO][SMF][CFG] Read config from [./config/uerouting.yaml]
2024-02-09T12:38:04.589809703Z [INFO][SMF][Main] Log enable is set to [true]
2024-02-09T12:38:04.589941171Z [INFO][SMF][Main] Log level is set to [info]
2024-02-09T12:38:04.590066046Z [INFO][SMF][Main] Report Caller is set to [false]
2024-02-09T12:38:04.590231277Z [INFO][SMF][CTX] smfconfig Info: Version[1.0.7] Description[SMF initial local configuration]
2024-02-09T12:38:04.590473948Z [INFO][SMF][CTX] Endpoints: [192.168.50.56]
2024-02-09T12:38:04.590640140Z [INFO][SMF][Init] Server started
2024-02-09T12:38:04.591950028Z [INFO][NRF][NFM] Handle NFRegisterRequest
2024-02-09T12:38:04.593245484Z [INFO][NRF][NFM] urilist update
2024-02-09T12:38:04.593782786Z [INFO][NRF][NFM] Create NF Profile
2024-02-09T12:38:04.594367147Z [INFO][NRF][NFM] Location header:  http://127.0.0.10:8000/nnrf-nfm/v1/nf-instances/3210ba3f-1552-4780-8b0f-084612cc3d40
2024-02-09T12:38:04.594599886Z [INFO][NRF][GIN] | 201 |       127.0.0.1 | PUT     | /nnrf-nfm/v1/nf-instances/3210ba3f-1552-4780-8b0f-084612cc3d40 |
2024-02-09T12:38:04.595210573Z [INFO][SMF][Init] SMF Registration to NRF {3210ba3f-1552-4780-8b0f-084612cc3d40 SMF REGISTERED 0 0xc00009e858 0xc00009e888 [] []   [127.0.0.2] [] <nil> [] [] <nil> 0 0 0 area1 <n
il> <nil> <nil> <nil> 0xc000082740 <nil> <nil> <nil> <nil> <nil> map[] <nil> false 0xc00009e750 false false []}
```

```
2024-02-09T12:38:04.596550964Z [INFO][UPF][PFCP][LAddr:127.0.0.8:8805] handleAssociationSetupRequest
2024-02-09T12:38:04.597870105Z [INFO][UPF][PFCP][LAddr:127.0.0.8:8805][CPNodeID:127.0.0.1] New node
2024-02-09T12:38:04.598297301Z [INFO][SMF][Main] Received PFCP Association Setup Accepted Response from UPF[127.0.0.8]
2024-02-09T12:38:04.687677151Z [INFO][UDR][Main] UDR version:
        free5GC version: v3.3.0
        build time:      2023-12-07T14:28:23Z
        commit hash:     a8ef9d9f
        commit time:     2023-05-11T08:26:37Z
        go version:      go1.18.10 linux/amd64
2024-02-09T12:38:04.688968594Z [INFO][UDR][CFG] Read config from [./config/udrcfg.yaml]
2024-02-09T12:38:04.689854427Z [INFO][UDR][Main] Log enable is set to [true]
2024-02-09T12:38:04.689926307Z [INFO][UDR][Main] Log level is set to [info]
2024-02-09T12:38:04.690051768Z [INFO][UDR][Main] Report Caller is set to [false]
2024-02-09T12:38:04.690260647Z [INFO][UDR][Init] UDR Config Info: Version[1.0.2] Description[UDR initial local configuration]
2024-02-09T12:38:04.690425031Z [INFO][UDR][Init] Server started
2024-02-09T12:38:04.690737254Z [INFO][UDR][Util] udrconfig Info: Version[1.0.2] Description[UDR initial local configuration]
2024-02-09T12:38:04.692831772Z [INFO][NRF][NFM] Handle NFRegisterRequest
2024-02-09T12:38:04.693998945Z [INFO][NRF][NFM] urilist update
2024-02-09T12:38:04.694502912Z [INFO][NRF][NFM] Create NF Profile
2024-02-09T12:38:04.695048600Z [INFO][NRF][NFM] Location header:  http://127.0.0.10:8000/nnrf-nfm/v1/nf-instances/a79812a1-2aef-4939-9681-5cdeafccd3c6
2024-02-09T12:38:04.695255050Z [INFO][NRF][GIN] | 201 |       127.0.0.1 | PUT     | /nnrf-nfm/v1/nf-instances/a79812a1-2aef-4939-9681-5cdeafccd3c6 |
2024-02-09T12:38:04.800932342Z [INFO][PCF][Main] pcf
2024-02-09T12:38:04.801132296Z [INFO][PCF][Main] PCF version:
        free5GC version: v3.3.0
        build time:      2023-12-07T14:28:03Z
        commit hash:     17f2a8fc
        commit time:     2023-05-11T08:21:46Z
        go version:      go1.18.10 linux/amd64
2024-02-09T12:38:04.803112521Z [INFO][PCF][CFG] Read config from [./config/pcfcfg.yaml]
2024-02-09T12:38:04.805070060Z [INFO][PCF][Main] Log enable is set to [true]
2024-02-09T12:38:04.805143640Z [INFO][PCF][Main] Log level is set to [info]
2024-02-09T12:38:04.805386603Z [INFO][PCF][Main] Report Caller is set to [false]
2024-02-09T12:38:04.805561903Z [INFO][PCF][Util] pcfconfig Info: Version[1.0.2] Description[PCF initial local configuration]
2024-02-09T12:38:04.805875219Z [INFO][PCF][Init] Server started
2024-02-09T12:38:04.806109922Z [INFO][PCF][Util] pcfconfig Info: Version[1.0.2] Description[PCF initial local configuration]
2024-02-09T12:38:04.808254731Z [INFO][NRF][NFM] Handle NFRegisterRequest
2024-02-09T12:38:04.809554307Z [INFO][NRF][NFM] urilist update
2024-02-09T12:38:04.810497145Z [INFO][NRF][NFM] Create NF Profile
2024-02-09T12:38:04.811780963Z [INFO][NRF][NFM] Location header:  http://127.0.0.10:8000/nnrf-nfm/v1/nf-instances/5ec5953c-c9a8-42f1-9824-1111667229c8
2024-02-09T12:38:04.812130636Z [INFO][NRF][GIN] | 201 |       127.0.0.1 | PUT     | /nnrf-nfm/v1/nf-instances/5ec5953c-c9a8-42f1-9824-1111667229c8 |
2024-02-09T12:38:04.813715741Z [INFO][NRF][DISC] Handle NFDiscoveryRequest
2024-02-09T12:38:04.815687010Z [INFO][NRF][GIN] | 200 |       127.0.0.1 | GET     | /nnrf-disc/v1/nf-instances?requester-nf-type=PCF&service-names=nudr-dr&target-nf-type=UDR |
2024-02-09T12:38:04.887476071Z [INFO][UDM][Main] UDM version:
        free5GC version: v3.3.0
        build time:      2023-12-07T14:28:17Z
        commit hash:     f9aad0ef
        commit time:     2023-05-11T08:24:25Z
        go version:      go1.18.10 linux/amd64
2024-02-09T12:38:04.889356207Z [INFO][UDM][CFG] Read config from [./config/udmcfg.yaml]
2024-02-09T12:38:04.890877779Z [INFO][UDM][Main] Log enable is set to [true]
2024-02-09T12:38:04.890952752Z [INFO][UDM][Main] Log level is set to [info]
2024-02-09T12:38:04.891181799Z [INFO][UDM][Main] Report Caller is set to [false]
2024-02-09T12:38:04.891314481Z [INFO][UDM][Init] UDM Config Info: Version[1.0.3] Description[UDM initial local configuration]
2024-02-09T12:38:04.891565635Z [INFO][UDM][Init] Server started
2024-02-09T12:38:04.891842476Z [INFO][UDM][Util] udmconfig Info: Version[1.0.3] Description[UDM initial local configuration]
2024-02-09T12:38:04.893189454Z [INFO][NRF][NFM] Handle NFRegisterRequest
2024-02-09T12:38:04.894502226Z [INFO][NRF][NFM] urilist update
2024-02-09T12:38:04.895053333Z [INFO][NRF][NFM] Create NF Profile
```

```
2024-02-09T12:38:05.012827103Z [INFO][NRF][NFM] Location header:  http://127.0.0.10:8000/nnrf-nfm/v1/nf-instances/aeb01fe5-f1e4-440e-9b5b-c0b82aa71a27
2024-02-09T12:38:05.013076882Z [INFO][NRF][GIN] | 201 |       127.0.0.1 | PUT     | /nnrf-nfm/v1/nf-instances/aeb01fe5-f1e4-440e-9b5b-c0b82aa71a27 |
2024-02-09T12:38:05.091377386Z [INFO][AUSF][Main] AUSF version:
        free5GC version: v3.3.0
        build time:      2023-12-07T14:27:40Z
        commit hash:     64f47ebe
        commit time:     2023-05-11T08:11:05Z
        go version:      go1.18.10 linux/amd64
2024-02-09T12:38:05.092284180Z [INFO][AUSF][CFG] Read config from [./config/ausfcfg.yaml]
2024-02-09T12:38:05.093539448Z [INFO][AUSF][Main] Log enable is set to [true]
2024-02-09T12:38:05.093628639Z [INFO][AUSF][Main] Log level is set to [info]
2024-02-09T12:38:05.093738357Z [INFO][AUSF][Main] Report Caller is set to [false]
2024-02-09T12:38:05.093937335Z [INFO][AUSF][Init] ausfconfig Info: Version[1.0.3] Description[AUSF initial local configuration]
ausf context =  &{{{0 0} {<nil>} map[] 0} {{0 0} {<nil>} map[] 0} f2a5bbb7-db62-4c29-ade4-64f9be89983a ausfGroup001 8000 127.0.0.9 127.0.0.9 http://127.0.0.9:8000 http http://127.0.0.10:8000 map[nausf-auth:{f2
a5bbb7-db62-4c29-ade4-64f9be89983a nausf-auth 0xc00000d620 http REGISTERED   0xc00000d608  [] <nil> [] [] <nil> 0 0 0 <nil> <nil>  }] [{001 01} {001 01}]  0xc0004f40a0 false}
2024-02-09T12:38:05.094307553Z [INFO][AUSF][Init] Server started
2024-02-09T12:38:05.095636094Z [INFO][NRF][NFM] Handle NFRegisterRequest
2024-02-09T12:38:05.096769668Z [INFO][NRF][NFM] urilist update
2024-02-09T12:38:05.097280143Z [INFO][NRF][NFM] Create NF Profile
2024-02-09T12:38:05.097835821Z [INFO][NRF][NFM] Location header:  http://127.0.0.10:8000/nnrf-nfm/v1/nf-instances/f2a5bbb7-db62-4c29-ade4-64f9be89983a
2024-02-09T12:38:05.098089566Z [INFO][NRF][GIN] | 201 |       127.0.0.1 | PUT     | /nnrf-nfm/v1/nf-instances/f2a5bbb7-db62-4c29-ade4-64f9be89983a |
2024-02-09T12:38:26.169979994Z [INFO][AMF][Ngap] [AMF] SCTP Accept from: 192.168.50.20:39367
2024-02-09T12:38:26.171070017Z [INFO][AMF][Ngap] Create a new NG connection for: 192.168.50.20:39367
2024-02-09T12:38:26.173862334Z [INFO][AMF][Ngap][ran_addr:192.168.50.20:39367] Handle NGSetupRequest
2024-02-09T12:38:26.173950419Z [INFO][AMF][Ngap][ran_addr:192.168.50.20:39367] Send NG-Setup response
2024-02-09T12:38:39.019573916Z [INFO][AMF][Ngap] Handle SCTP Notification[addr: <nil>]
2024-02-09T12:38:39.019992303Z [INFO][AMF][Ngap][ran_addr:192.168.50.20:39367] SCTP_SHUTDOWN_EVENT notification, close the connection
2024-02-09T12:38:39.021058214Z [INFO][AMF][Ngap][ran_addr:192.168.50.20:39367] Remove RAN Context[ID: <PlmnID: {Mcc:001 Mnc:01}, GNbID: 0012345>]
2024-02-09T12:38:39.021755963Z [INFO][AMF][Ngap] Handle SCTP Notification[addr: <nil>]
2024-02-09T12:38:39.022398085Z [WARN][AMF][Ngap] RAN context has been removed[addr: <nil>]
2024-02-09T12:38:39.022503525Z [INFO][AMF][Ngap] Handle SCTP Connection Error[addr: <nil>] - remove RAN
2024-02-09T12:38:39.022687456Z [WARN][AMF][Ngap] RAN context has been removed[addr: <nil>]
2024-02-09T12:38:48.175751394Z [INFO][AMF][Ngap] [AMF] SCTP Accept from: 192.168.50.20:41920
2024-02-09T12:38:48.177723066Z [INFO][AMF][Ngap] Create a new NG connection for: 192.168.50.20:41920
2024-02-09T12:38:48.180800556Z [INFO][AMF][Ngap][ran_addr:192.168.50.20:41920] Handle NGSetupRequest
2024-02-09T12:38:48.181156653Z [INFO][AMF][Ngap][ran_addr:192.168.50.20:41920] Send NG-Setup response
```

Fig 4.42 logs of various NFs

After running the command "./run.sh", in the log it is noticeable that NGAP request and response is happening between AMF and gNB, in fig 4.43.

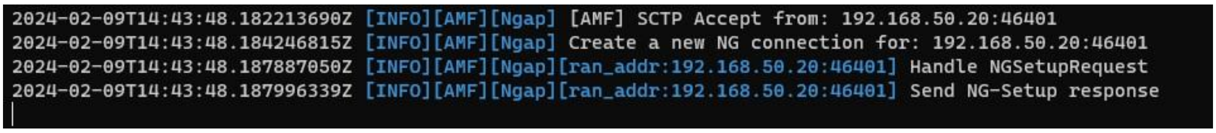


Fig 4.43 NGAP Request and Response (in Free5GC)

Open the Amari Callbox Classic in another tab via SSH.:

- Callbox is configured to provide an automatic LTE service. At each reboot of the PC, the LTE network is turned on automatically.
- There are few commands to manage LTE automatic service.

I. Status: "service lte status" //this command will return "active(running)" status if service is running.
II. Stop: "service lte stop" // this command will stop the services.
III. Start: "service lte start" // this command will start the LTE service.
IV. Disable: "sysytemctl disable lte" // LTE service remains enable until next reboot.
V. Enable: "sysytemctl enable lte". // LTE service remains disable until next reboot.

Fig 4.44 shows one command is using below. Executing any of these commands will affect the Amarisoft WebGUI, which will refresh automatically. [Ref. https://www.amarisoft.com/app/uploads/2022/03/userguide_callbox_classic.pdf]

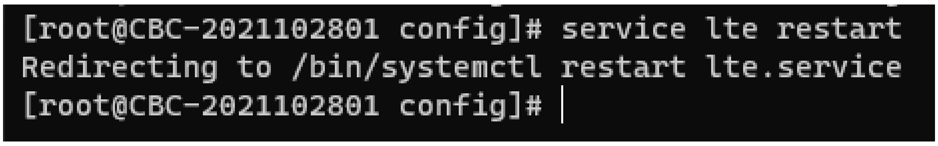


Fig 4.44 Command to manage LTE service.

- Amarisoft WebGUI:

Fig 4.45 will display a screenshot of the Amarisoft WebGUI, illustrating the NGAP request and NGAP response between AMF and gNB.

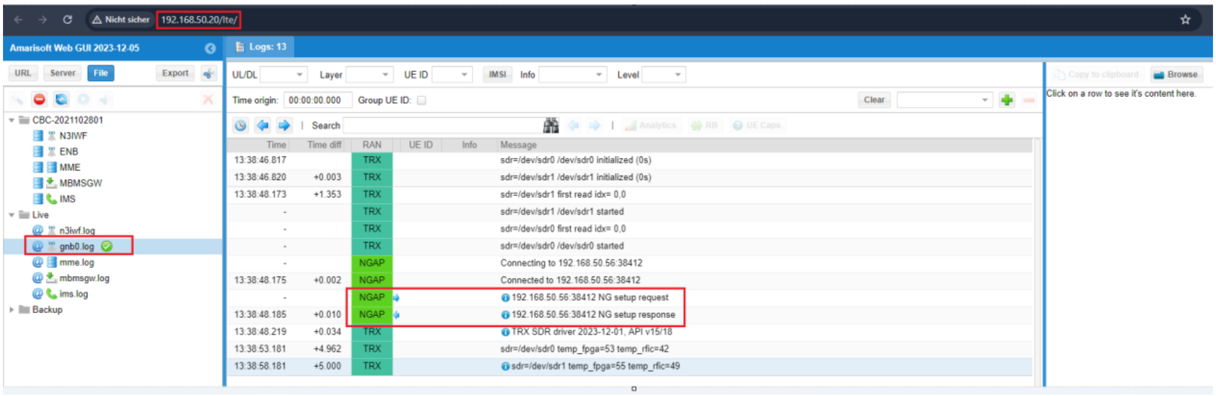
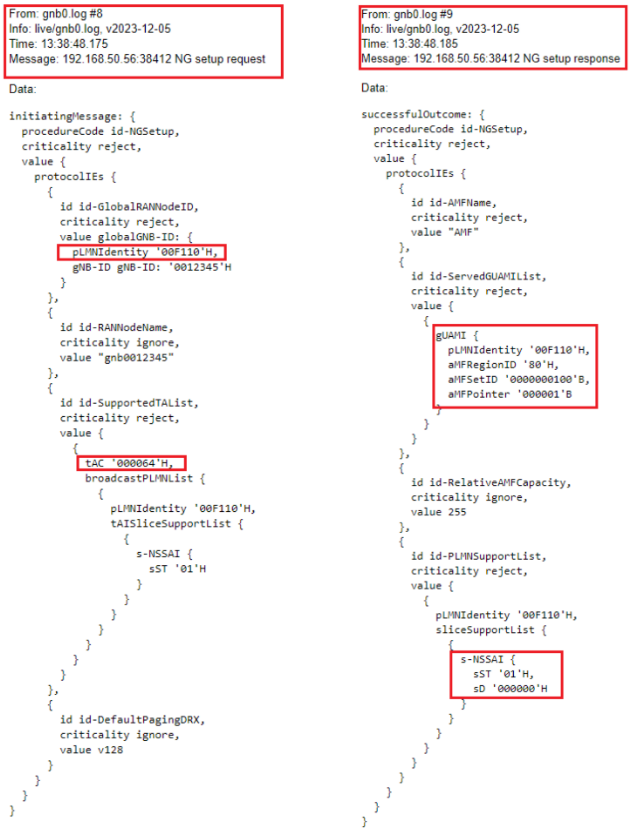


Fig 4.45 NGAP Request and NGAP Response

- <u>Logs of NGAP Request and NGAP Response log in Amarisoft WbGUI:</u>



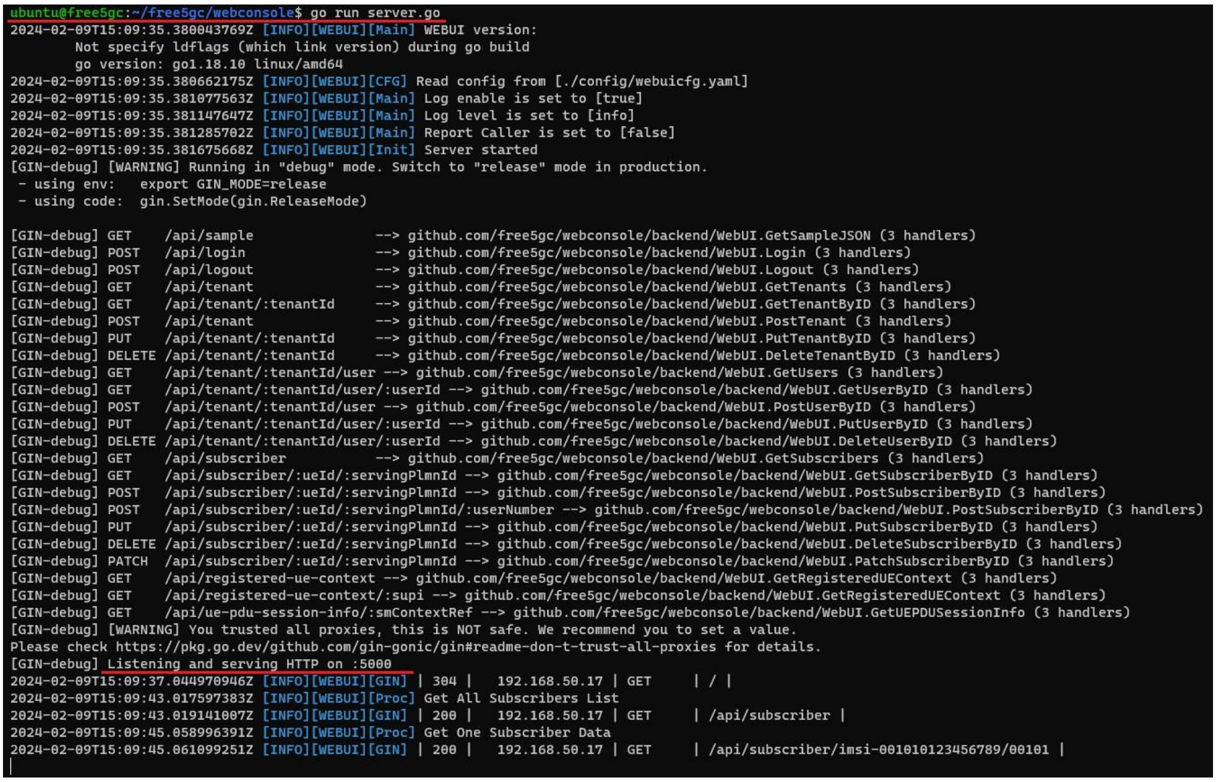
```
From: gnb0.log #8
Info: live/gnb0.log, v2023-12-05
Time: 13:38:48.175
Message: 192.168.50.56:38412 NG setup request

Data:

initiatingMessage: {
  procedureCode id-NGSetup,
  criticality reject,
  value {
    protocolIEs {
      {
        id id-GlobalRANNodeID,
        criticality reject,
        value globalGNB-ID: {
          pLMNIdentity '00F110'H,
          gNB-ID gNB-ID: '0012345'H
        }
      },
      {
        id id-RANNodeName,
        criticality ignore,
        value "gnb0012345"
      },
      {
        id id-SupportedTAList,
        criticality reject,
        value {
          {
            tAC '000064'H,
            broadcastPLMNList {
              {
                pLMNIdentity '00F110'H,
                tAISliceSupportList {
                  {
                    s-NSSAI {
                      sST '01'H
                    }
                  }
                }
              }
            }
          }
        }
      },
      {
        id id-DefaultPagingDRX,
        criticality ignore,
        value v128
      }
    }
  }
}
```

```
From: gnb0.log #9
Info: live/gnb0.log, v2023-12-05
Time: 13:38:48.185
Message: 192.168.50.56:38412 NG setup response

Data:

successfulOutcome: {
  procedureCode id-NGSetup,
  criticality reject,
  value {
    protocolIEs {
      {
        id id-AMFName,
        criticality reject,
        value "AMF"
      },
      {
        id id-ServedGUAMIList,
        criticality reject,
        value {
          {
            gUAMI {
              pLMNIdentity '00F110'H,
              aMFRegionID '80'H,
              aMFSetID '0000000100'B,
              aMFPointer '000001'B
            }
          }
        }
      },
      {
        id id-RelativeAMFCapacity,
        criticality ignore,
        value 255
      },
      {
        id id-PLMNSupportList,
        criticality reject,
        value {
          {
            pLMNIdentity '00F110'H,
            sliceSupportList {
              {
                s-NSSAI {
                  sST '01'H,
                  sD '000000'H
                }
              }
            }
          }
        }
      }
    }
  }
}
```

Fig 4.46 Amarisoft WebGUI log of NGAP Response and NGAP Response

<u>Again, open the Free5GC in another tab via SSH</u>.

- This tab is used for running Free5gc WebGUI which is illustrated in fig 4.47.



```
ubuntu@free5gc:~/free5gc/webconsole$ go run server.go
2024-02-09T15:09:35.380043769Z [INFO][WEBUI][Main] WEBUI version:
        Not specify ldflags (which link version) during go build
        go version: go1.18.10 linux/amd64
2024-02-09T15:09:35.380662175Z [INFO][WEBUI][CFG] Read config from [./config/webuicfg.yaml]
2024-02-09T15:09:35.381077563Z [INFO][WEBUI][Main] Log enable is set to [true]
2024-02-09T15:09:35.381147647Z [INFO][WEBUI][Main] Log level is set to [info]
2024-02-09T15:09:35.381285702Z [INFO][WEBUI][Main] Report Caller is set to [false]
2024-02-09T15:09:35.381675668Z [INFO][WEBUI][Init] Server started
[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
 - using env:   export GIN_MODE=release
 - using code:  gin.SetMode(gin.ReleaseMode)

[GIN-debug] GET    /api/sample               --> github.com/free5gc/webconsole/backend/WebUI.GetSampleJSON (3 handlers)
[GIN-debug] POST   /api/login                --> github.com/free5gc/webconsole/backend/WebUI.Login (3 handlers)
[GIN-debug] POST   /api/logout               --> github.com/free5gc/webconsole/backend/WebUI.Logout (3 handlers)
[GIN-debug] GET    /api/tenant               --> github.com/free5gc/webconsole/backend/WebUI.GetTenants (3 handlers)
[GIN-debug] GET    /api/tenant/:tenantId     --> github.com/free5gc/webconsole/backend/WebUI.GetTenantByID (3 handlers)
[GIN-debug] POST   /api/tenant               --> github.com/free5gc/webconsole/backend/WebUI.PostTenant (3 handlers)
[GIN-debug] PUT    /api/tenant/:tenantId     --> github.com/free5gc/webconsole/backend/WebUI.PutTenantByID (3 handlers)
[GIN-debug] DELETE /api/tenant/:tenantId     --> github.com/free5gc/webconsole/backend/WebUI.DeleteTenantByID (3 handlers)
[GIN-debug] GET    /api/tenant/:tenantId/user --> github.com/free5gc/webconsole/backend/WebUI.GetUsers (3 handlers)
[GIN-debug] GET    /api/tenant/:tenantId/user/:userId --> github.com/free5gc/webconsole/backend/WebUI.GetUserByID (3 handlers)
[GIN-debug] POST   /api/tenant/:tenantId/user --> github.com/free5gc/webconsole/backend/WebUI.PostUserByID (3 handlers)
[GIN-debug] PUT    /api/tenant/:tenantId/user/:userId --> github.com/free5gc/webconsole/backend/WebUI.PutUserByID (3 handlers)
[GIN-debug] DELETE /api/tenant/:tenantId/user/:userId --> github.com/free5gc/webconsole/backend/WebUI.DeleteUserByID (3 handlers)
[GIN-debug] GET    /api/subscriber           --> github.com/free5gc/webconsole/backend/WebUI.GetSubscribers (3 handlers)
[GIN-debug] GET    /api/subscriber/:ueId/:servingPlmnId --> github.com/free5gc/webconsole/backend/WebUI.GetSubscriberByID (3 handlers)
[GIN-debug] POST   /api/subscriber/:ueId/:servingPlmnId --> github.com/free5gc/webconsole/backend/WebUI.PostSubscriberByID (3 handlers)
[GIN-debug] POST   /api/subscriber/:ueId/:servingPlmnId/:userNumber --> github.com/free5gc/webconsole/backend/WebUI.PostSubscriberByID (3 handlers)
[GIN-debug] PUT    /api/subscriber/:ueId/:servingPlmnId --> github.com/free5gc/webconsole/backend/WebUI.PutSubscriberByID (3 handlers)
[GIN-debug] DELETE /api/subscriber/:ueId/:servingPlmnId --> github.com/free5gc/webconsole/backend/WebUI.DeleteSubscriberByID (3 handlers)
[GIN-debug] PATCH  /api/subscriber/:ueId/:servingPlmnId --> github.com/free5gc/webconsole/backend/WebUI.PatchSubscriberByID (3 handlers)
[GIN-debug] GET    /api/registered-ue-context --> github.com/free5gc/webconsole/backend/WebUI.GetRegisteredUEContext (3 handlers)
[GIN-debug] GET    /api/registered-ue-context/:supi --> github.com/free5gc/webconsole/backend/WebUI.GetRegisteredUEContext (3 handlers)
[GIN-debug] GET    /api/ue-pdu-session-info/:smContextRef --> github.com/free5gc/webconsole/backend/WebUI.GetUEPDUSessionInfo (3 handlers)
[GIN-debug] [WARNING] You trusted all proxies, this is NOT safe. We recommend you to set a value.
Please check https://pkg.go.dev/github.com/gin-gonic/gin#readme-don-t-trust-all-proxies for details.
[GIN-debug] Listening and serving HTTP on :5000
2024-02-09T15:09:37.044970946Z [INFO][WEBUI][GIN] | 304 |   192.168.50.17 | GET     | / |
2024-02-09T15:09:43.017597383Z [INFO][WEBUI][Proc] Get All Subscribers List
2024-02-09T15:09:43.019141007Z [INFO][WEBUI][GIN] | 200 |   192.168.50.17 | GET     | /api/subscriber |
2024-02-09T15:09:45.058996391Z [INFO][WEBUI][Proc] Get One Subscriber Data
2024-02-09T15:09:45.061099251Z [INFO][WEBUI][GIN] | 200 |   192.168.50.17 | GET     | /api/subscriber/imsi-001010123456789/00101 |
```

Fig 4.47 Free5GC WebConsole command and port

After running the above command, simply open any web browser and enter the IP address followed by the given port number (e.g., 192.168.50.56:5000). The login details for the WebGUI are mentioned above. Once logged in, navigate to 'SUBSCRIBERS' and add a new subscriber (as illustrated in Figure 4.35). After saving and exiting, the subscriber details will be displayed as shown in Figure 4.47(a).

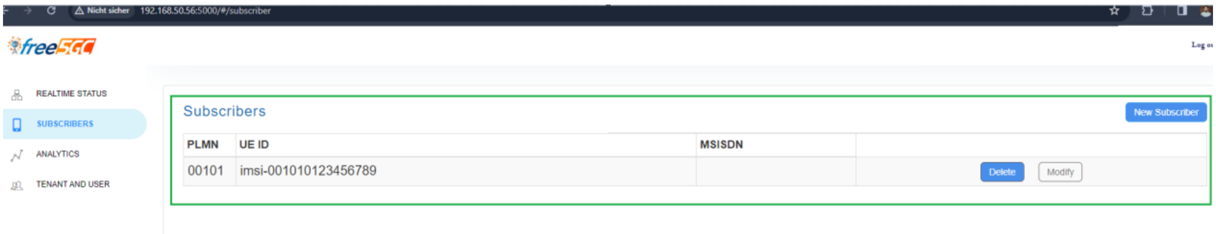


Fig 4.47(a) Subscribers

- Wireshark:

After executing all the commands together (./run.sh, go run server.go, and service lte restart on the Amari callbox), open Wireshark and select 'Ethernet'. After selecting 'Ethernet', apply the filter (ngap||http||sctp) as shown in Fig. 4.49. Fig. 4.48 illustrates which filter needs to be used for capturing.

Fig 4.48 Ethernet filter

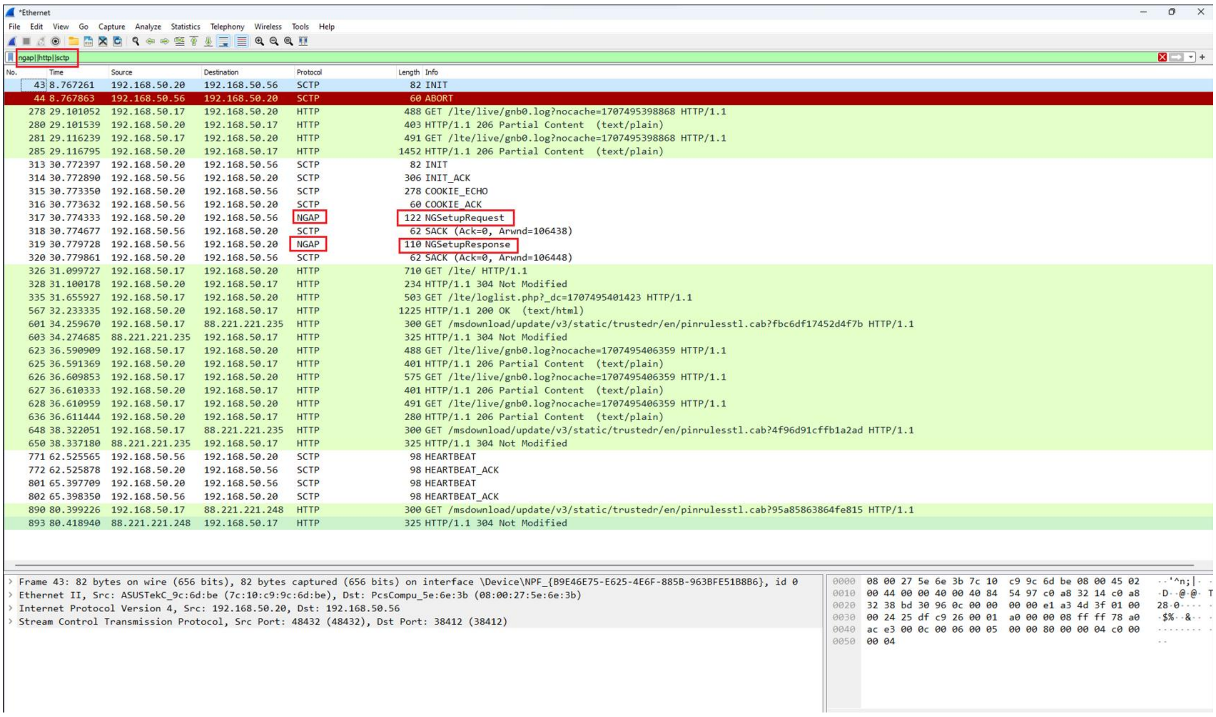


Fig 4.49 ngap||http||sctp

- Wireshark of NGSetupRequest details fig 4.50:

```
Frame 317: 122 bytes on wire (976 bits), 122 bytes captured (976 bits) on interface \Device\NPF_{B9E46E75-E625-4E6F-885B-963BFE51B8B6}, id 0
Ethernet II, Src: ASUSTekC_9c:6d:be (7c:10:c9:9c:6d:be), Dst: PcsCompu_5e:6e:3b (08:00:27:5e:6e:3b)
Internet Protocol Version 4, Src: 192.168.50.20, Dst: 192.168.50.56
Stream Control Transmission Protocol, Src Port: 41158 (41158), Dst Port: 38412 (38412)
NG Application Protocol (NGSetupRequest)
  NGAP-PDU: initiatingMessage (0)
    initiatingMessage
      procedureCode: id-NGSetup (21)
      criticality: reject (0)
      value
        NGSetupRequest
          protocolIEs: 4 items
            Item 0: id-GlobalRANNodeID
              ProtocolIE-Field
                id: id-GlobalRANNodeID (27)
                criticality: reject (0)
                value
                  GlobalRANNodeID: globalGNB-ID (0)
                    globalGNB-ID
                      pLMNIdentity: 00f110
                      gNB-ID: gNB-ID (0)
                        gNB-ID: 00123450 [bit length 28, 4 LSB pad bits, 0000 0000  0001 0010  0011 0100  0101 .... decimal value 74565]
            Item 1: id-RANNodeName
              ProtocolIE-Field
                id: id-RANNodeName (82)
                criticality: ignore (1)
                value
                  RANNodeName: gnb0012345
            Item 2: id-SupportedTAList
              ProtocolIE-Field
                id: id-SupportedTAList (102)
                criticality: reject (0)
                value
                  SupportedTAList: 1 item
                    Item 0
                      SupportedTAItem
                        tAC: 100 (0x000064)
                        broadcastPLMNList: 1 item
            Item 3: id-DefaultPagingDRX
              ProtocolIE-Field
                id: id-DefaultPagingDRX (21)
                criticality: ignore (1)
                value
                  PagingDRX: v128 (2)
```

Fig 4.50 NGSetupRequest

- Wireshark of NGSetupResponse details fig 4.51

```
Frame 319: 110 bytes on wire (880 bits), 110 bytes captured (880 bits) on interface \Device\NPF_{B9E46E75-E625-4E6F-885B-963BFE51B8B6}, id 0
Ethernet II, Src: PcsCompu_5e:6e:3b (08:00:27:5e:6e:3b), Dst: ASUSTekC_9c:6d:be (7c:10:c9:9c:6d:be)
Internet Protocol Version 4, Src: 192.168.50.56, Dst: 192.168.50.20
Stream Control Transmission Protocol, Src Port: 38412 (38412), Dst Port: 41158 (41158)
NG Application Protocol (NGSetupResponse)
  NGAP-PDU: successfulOutcome (1)
    successfulOutcome
      procedureCode: id-NGSetup (21)
      criticality: reject (0)
      value
        NGSetupResponse
          protocolIEs: 4 items
            Item 0: id-AMFName
              ProtocolIE-Field
                id: id-AMFName (1)
                criticality: reject (0)
                value
                  AMFName: AMF
            Item 1: id-ServedGUAMIList
              ProtocolIE-Field
                id: id-ServedGUAMIList (96)
                criticality: reject (0)
                value
                  ServedGUAMIList: 1 item
                    Item 0
                      ServedGUAMIItem
                        gUAMI
                          pLMNIdentity: 00f110
                          aMFRegionID: 80 [bit length 8, 1000 0000 decimal value 128]
                          aMFSetID: 0100 [bit length 10. 6 LSB pad bits, 0000 0001  00.. .... decimal value 4]
                          aMFPointer: 04 [bit length 6, 2 LSB pad bits, 0000 01.. decimal value 1]
            Item 2: id-RelativeAMFCapacity
              ProtocolIE-Field
                id: id-RelativeAMFCapacity (86)
                criticality: ignore (1)
                value
                  RelativeAMFCapacity: 255
            Item 3: id-PLMNSupportList
              ProtocolIE-Field
                id: id-PLMNSupportList (80)
                criticality: reject (0)
                value
                  PLMNSupportList: 1 item
                    Item 0
                      PLMNSupportItem
                        pLMNIdentity: 00f110
                          Mobile Country Code (MCC): Unknown (1)
                          Mobile Network Code (MNC): Unknown (01)
                        sliceSupportList: 1 item
                          Item 0
                            SliceSupportItem
                              s-NSSAI
                                sST: 01
                                sD: 000000
```

Fig 4.51 NGSetupResponse

4.5 Issues During Project

In this issue, there are two types of authentication methods in Free5GC: 1) 5G_AKA, and 2) EAP_AKA_PRIME. During this project, we tested these authentication methods to evaluate their performance.

- 5G_AKA: In the Free5GC WebGUI, when editing a subscriber, if the authentication method is set to 5G_AKA and the configuration is saved, this issue occurs after re-running all the commands in Free5GC and Amarisoft. The issue arises when the smartphone is either restarted or put into airplane mode. Fig 4.52 and Fig 4.53 show those issues.

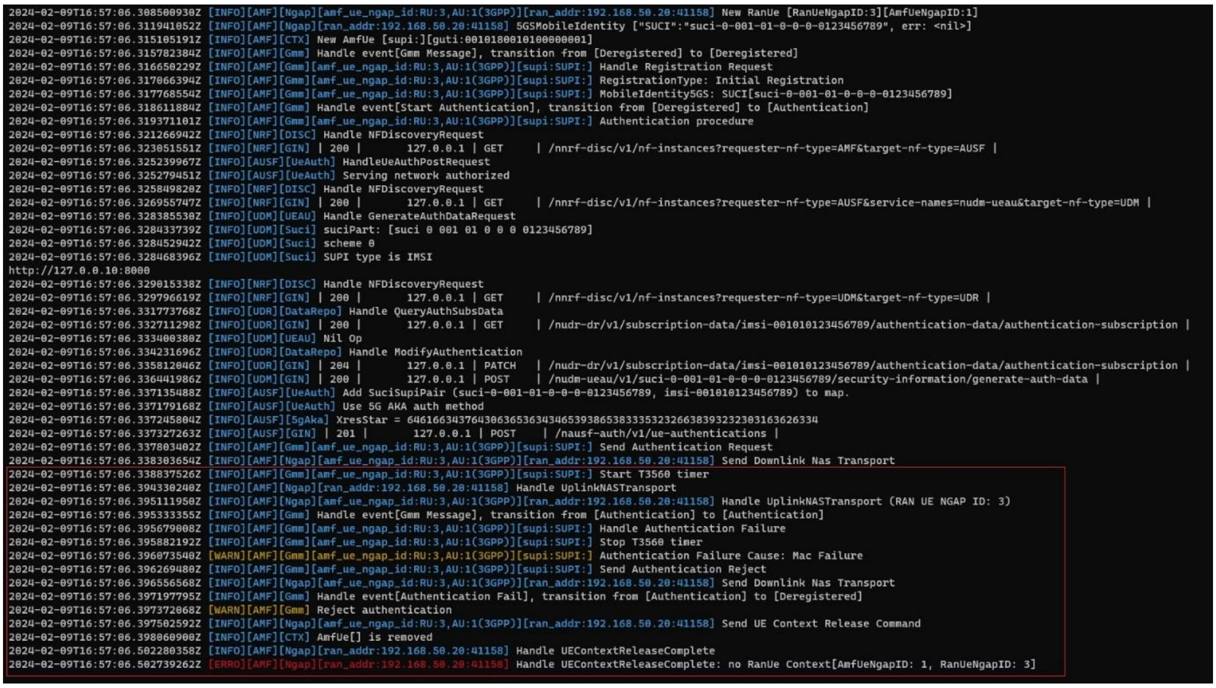


Fig 4.52 Authentication failure (MAC failure) at Free5GC

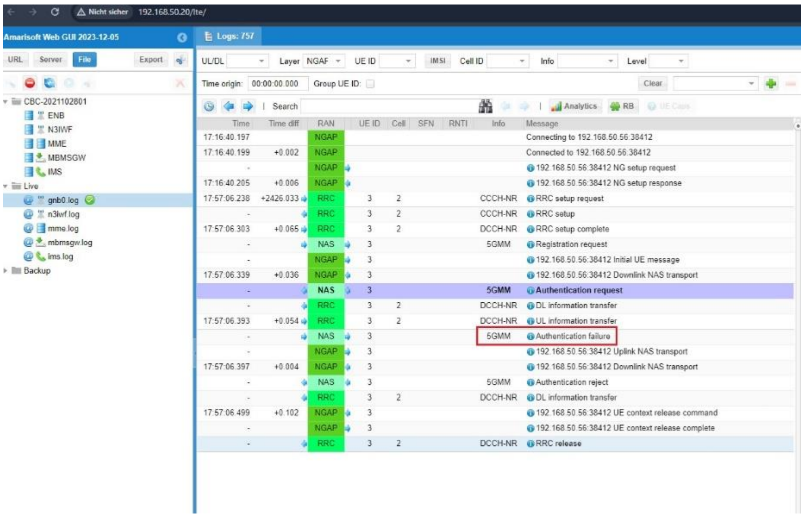
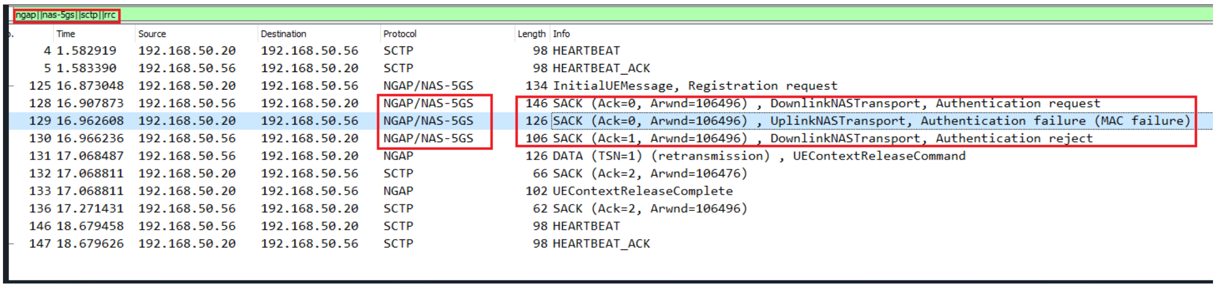


Fig 4.53 Authentication failure logs in Amarisoft WebGUI

Wireshark of MAC failure:



ngap||nas-5gs||sctp||rrc

No.	Time	Source	Destination	Protocol	Length	Info
4	1.582919	192.168.50.20	192.168.50.56	SCTP	98	HEARTBEAT
5	1.583390	192.168.50.56	192.168.50.20	SCTP	98	HEARTBEAT_ACK
125	16.873048	192.168.50.20	192.168.50.56	NGAP/NAS-5GS	134	InitialUEMessage, Registration request
128	16.907873	192.168.50.56	192.168.50.20	NGAP/NAS-5GS	146	SACK (Ack=0, Arwnd=106496) , DownlinkNASTransport, Authentication request
129	16.962608	192.168.50.20	192.168.50.56	NGAP/NAS-5GS	126	SACK (Ack=0, Arwnd=106496) , UplinkNASTransport, Authentication failure (MAC failure)
130	16.966236	192.168.50.56	192.168.50.20	NGAP/NAS-5GS	106	SACK (Ack=1, Arwnd=106496) , DownlinkNASTransport, Authentication reject
131	17.068487	192.168.50.56	192.168.50.20	NGAP	126	DATA (TSN=1) (retransmission) , UEContextReleaseCommand
132	17.068811	192.168.50.20	192.168.50.56	SCTP	66	SACK (Ack=2, Arwnd=106476)
133	17.068811	192.168.50.20	192.168.50.56	NGAP	102	UEContextReleaseComplete
136	17.271431	192.168.50.56	192.168.50.20	SCTP	62	SACK (Ack=2, Arwnd=106496)
146	18.679458	192.168.50.56	192.168.50.20	SCTP	98	HEARTBEAT
147	18.679626	192.168.50.20	192.168.50.56	SCTP	98	HEARTBEAT_ACK

Fig 4.54 MAC failure (5G_AKA)

Log of Authentication request (from 5G core to gNB) fig 4.55, Authentication failure (gNB to AMF) fig 4.56, Authentication reject (AMF to gNB) 4.57:

```
Frame 128: 146 bytes on wire (1168 bits), 146 bytes captured (1168 bits) on interface \Device\NPF_{B9E46E75-E625-4E6F-885B-963BFE51B8B6}, id 0
Ethernet II, Src: PcsCompu_5e:6e:3b (08:00:27:5e:6e:3b), Dst: ASUSTekC_9c:6d:be (7c:10:c9:9c:6d:be)
Internet Protocol Version 4, Src: 192.168.50.56, Dst: 192.168.50.20
Stream Control Transmission Protocol, Src Port: 38412 (38412), Dst Port: 41158 (41158)
NG Application Protocol (DownlinkNASTransport)
    NGAP-PDU: initiatingMessage (0)
        initiatingMessage
            procedureCode: id-DownlinkNASTransport (4)
            criticality: ignore (1)
            value
                DownlinkNASTransport
                    protocolIEs: 3 items
                        Item 0: id-AMF-UE-NGAP-ID
                            ProtocolIE-Field
                                id: id-AMF-UE-NGAP-ID (10)
                                criticality: reject (0)
                                value
                                    AMF-UE-NGAP-ID: 1
                        Item 1: id-RAN-UE-NGAP-ID
                            ProtocolIE-Field
                                id: id-RAN-UE-NGAP-ID (85)
                                criticality: reject (0)
                                value
                                    RAN-UE-NGAP-ID: 3
                        Item 2: id-NAS-PDU
                            ProtocolIE-Field
                                id: id-NAS-PDU (38)
                                criticality: reject (0)
                                value
                                    NAS-PDU: 7e005600020000210046a306442990fcca42068ab1e0ea5220103dc36fa8876c8000ed6f…
                                        Non-Access-Stratum 5GS (NAS)PDU
                                            Plain NAS 5GS Message
                                                Extended protocol discriminator: 5G mobility management messages (126)
                                                0000 .... = Spare Half Octet: 0
                                                .... 0000 = Security header type: Plain NAS message, not security protected (0)
                                                Message type: Authentication request (0x56)
                                                0000 .... = Spare Half Octet: 0
                                                NAS key set identifier - ngKSI
                                                    .... 0... = Type of security context flag (TSC): Native security context (for KSIAMF)
                                                    .... .000 = NAS key set identifier: 0
                                                ABBA
                                                    Length: 2
                                                    ABBA Contents: 0000
                                                Authentication Parameter RAND - 5G authentication challenge
                                                    Element ID: 0x21
                                                    RAND value: 0046a306442990fcca42068ab1e0ea52
                                                Authentication Parameter AUTN (UMTS and EPS authentication challenge) - 5G authentication challenge
                                                    Element ID: 0x20
                                                    Length: 16
                                                    AUTN value: 3dc36fa8876c8000ed6f031096971ad3
                                                        SQN xor AK: 3dc36fa8876c
                                                        AMF: 8000
                                                        MAC: ed6f031096971ad3
```

Fig 4.55 Authentication request (DownlinkNASTransport)

```
Frame 129: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits) on interface \Device\NPF_{B9E46E75-E625-4E6F-885B-963BFE51B8B6}, id 0
Ethernet II, Src: ASUSTekC_9c:6d:be (7c:10:c9:9c:6d:be), Dst: PcsCompu_5e:6e:3b (08:00:27:5e:6e:3b)
Internet Protocol Version 4, Src: 192.168.50.20, Dst: 192.168.50.56
Stream Control Transmission Protocol, Src Port: 41158 (41158), Dst Port: 38412 (38412)
NG Application Protocol (UplinkNASTransport)
  NGAP-PDU: initiatingMessage (0)
    initiatingMessage
      procedureCode: id-UplinkNASTransport (46)
      criticality: ignore (1)
      value
        UplinkNASTransport
          protocolIEs: 4 items
            Item 0: id-AMF-UE-NGAP-ID
              ProtocolIE-Field
                id: id-AMF-UE-NGAP-ID (10)
                criticality: reject (0)
                value
                  AMF-UE-NGAP-ID: 1
            Item 1: id-RAN-UE-NGAP-ID
              ProtocolIE-Field
                id: id-RAN-UE-NGAP-ID (85)
                criticality: reject (0)
                value
                  RAN-UE-NGAP-ID: 3
            Item 2: id-NAS-PDU
              ProtocolIE-Field
                id: id-NAS-PDU (38)
                criticality: reject (0)
                value
                  NAS-PDU: 7e005914
                    Non-Access-Stratum 5GS (NAS)PDU
                      Plain NAS 5GS Message
                        Extended protocol discriminator: 5G mobility management messages (126)
                        0000 .... = Spare Half Octet: 0
                        .... 0000 = Security header type: Plain NAS message, not security protected (0)
                        Message type: Authentication failure (0x59)
                        5GMM cause
                          5GMM cause: MAC failure (20)
            Item 3: id-UserLocationInformation
              ProtocolIE-Field
                id: id-UserLocationInformation (121)
                criticality: ignore (1)
                value
                  UserLocationInformation: userLocationInformationNR (1)
                    userLocationInformationNR
                      nR-CGI
                        pLMNIdentity: 00f110
                        nRCellIdentity: 0x0001234502
                      tAI
                        pLMNIdentity: 00f110
                        tAC: 100 (0x000064)
```

Fig 4.56 Authentication failure (UplinkNASTransport)

```
Ethernet II, Src: PcsCompu_5e:6e:3b (08:00:27:5e:6e:3b), Dst: ASUSTekC_9c:6d:be (7c:10:c9:9c:6d:be)
Internet Protocol Version 4, Src: 192.168.50.56, Dst: 192.168.50.20
Stream Control Transmission Protocol, Src Port: 38412 (38412), Dst Port: 41158 (41158)
NG Application Protocol (DownlinkNASTransport)
  NGAP-PDU: initiatingMessage (0)
    initiatingMessage
      procedureCode: id-DownlinkNASTransport (4)
      criticality: ignore (1)
      value
        DownlinkNASTransport
          protocolIEs: 3 items
            Item 0: id-AMF-UE-NGAP-ID
              ProtocolIE-Field
                id: id-AMF-UE-NGAP-ID (10)
                criticality: reject (0)
                value
                  AMF-UE-NGAP-ID: 1
            Item 1: id-RAN-UE-NGAP-ID
              ProtocolIE-Field
                id: id-RAN-UE-NGAP-ID (85)
                criticality: reject (0)
                value
                  RAN-UE-NGAP-ID: 3
            Item 2: id-NAS-PDU
              ProtocolIE-Field
                id: id-NAS-PDU (38)
                criticality: reject (0)
                value
                 NAS-PDU: 7e0058
                    Non-Access-Stratum 5GS (NAS)PDU
                      Plain NAS 5GS Message
                        Extended protocol discriminator: 5G mobility management messages (126)
                        0000 .... = Spare Half Octet: 0
                        .... 0000 = Security header type: Plain NAS message, not security protected (0)
                        Message type: Authentication reject (0x58)
```

Fig 4.57 Authentication Reject

- EAP_AKA_PRIME: The same approach as used in 5G_AKA should be applied, with the only modification being the authentication method, which should be changed to EAP_AKA_PRIME instead of 5G_AKA in fig 4.58. The entire process will remain the same.

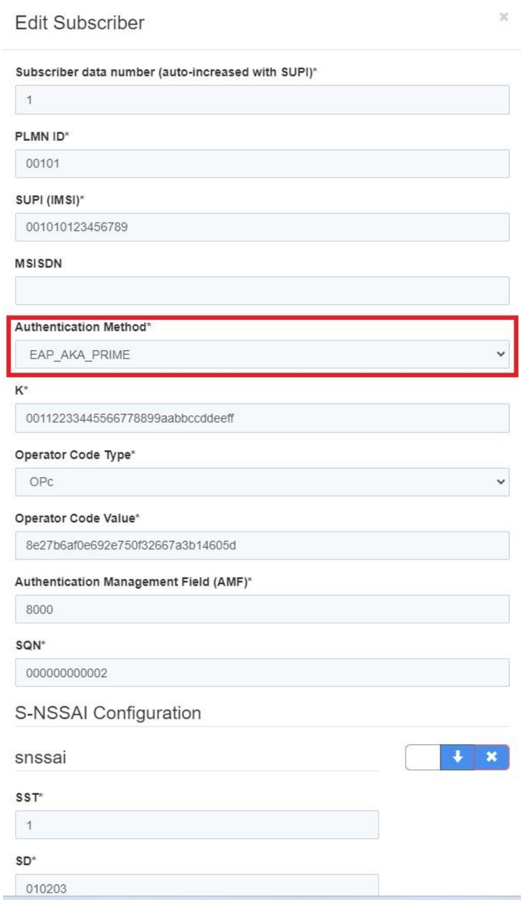


Fig 4.58 EAP_AKA_PRIME

Issues occurred in Free5GC after changing Authentication method Fig 4.59. And Fig 4.60 shows the Amarisoft WebGUI getting Authentication request, Authentication response and the rejection of Authentication.

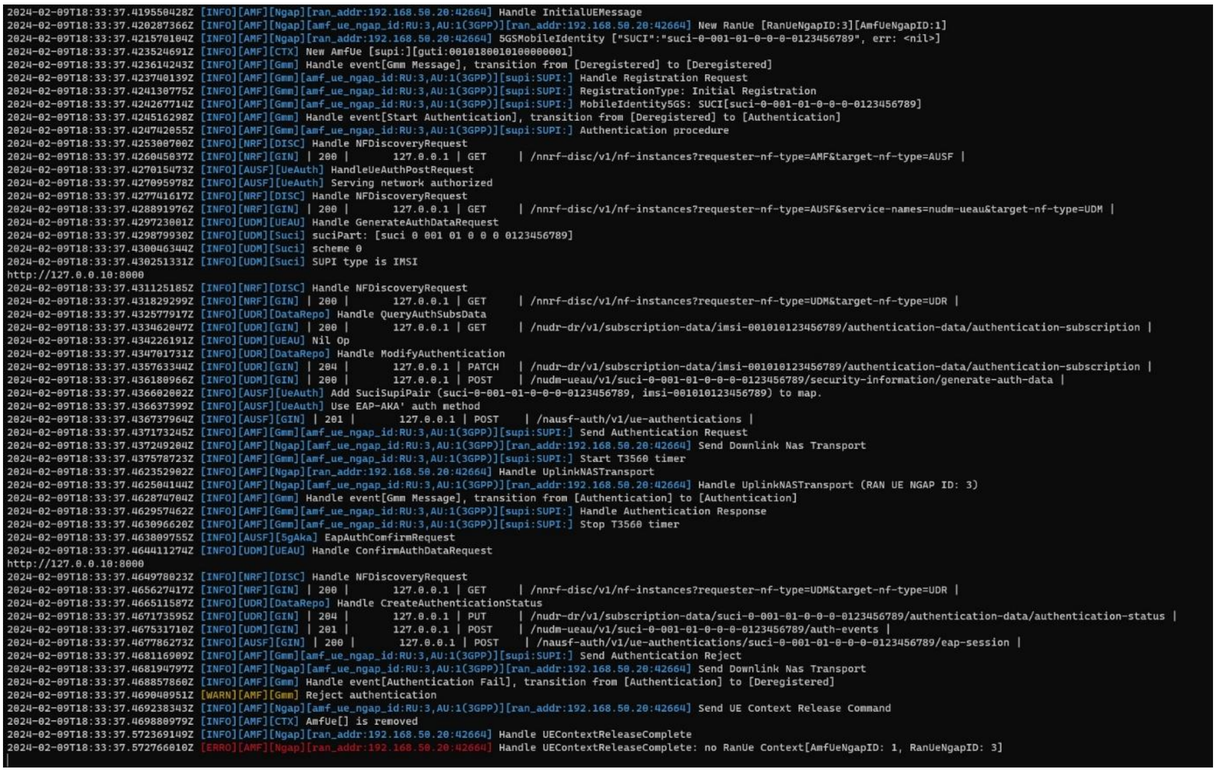


Fig 4.59 Free5GC changing Authentication Method

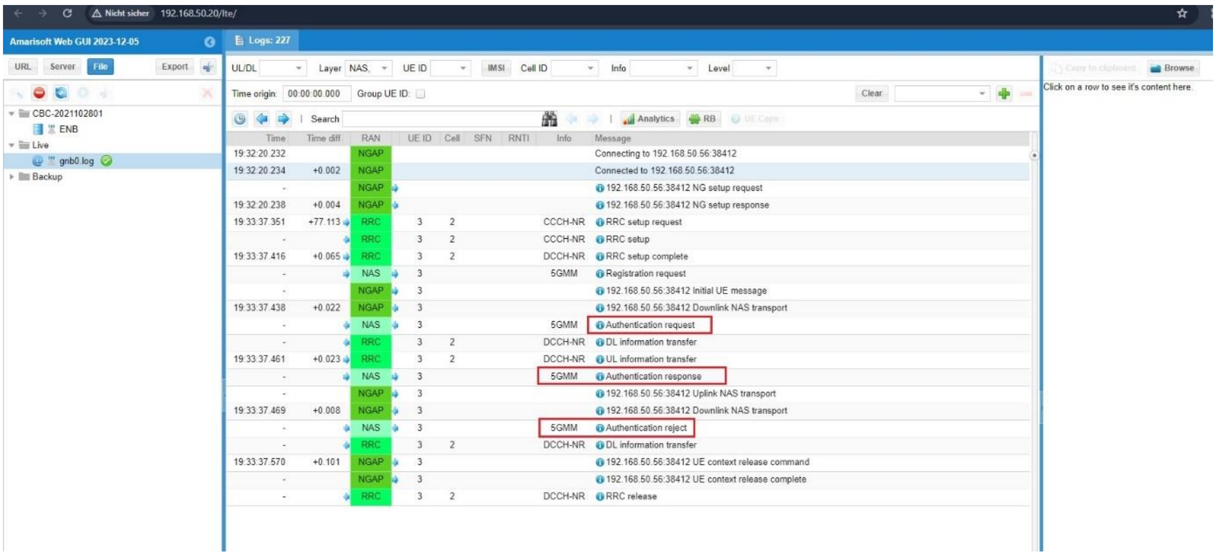


Fig 4.60 Amarisoft WebGUI Authentication request, response, reject

Wireshark at EAP_AKA_PRIME:

ngap||nas-5gs||sctp||rrc||http

.	Time	Source	Destination	Protocol	Length	Info
99	11.674711	192.168.50.20	192.168.50.56	SCTP	82	INIT
100	11.675358	192.168.50.56	192.168.50.20	SCTP	306	INIT_ACK
101	11.675525	192.168.50.20	192.168.50.56	SCTP	278	COOKIE_ECHO
102	11.676121	192.168.50.56	192.168.50.20	SCTP	60	COOKIE_ACK
103	11.676516	192.168.50.20	192.168.50.56	NGAP	122	NGSetupRequest
104	11.677043	192.168.50.56	192.168.50.20	SCTP	62	SACK (Ack=0, Arwnd=106438)
106	11.681713	192.168.50.56	192.168.50.20	NGAP	110	NGSetupResponse
107	11.681900	192.168.50.20	192.168.50.56	SCTP	62	SACK (Ack=0, Arwnd=106448)
130	13.946006	192.168.50.20	192.168.50.56	SCTP	60	SHUTDOWN
131	13.946625	192.168.50.56	192.168.50.20	SCTP	60	SHUTDOWN_ACK
132	13.946781	192.168.50.20	192.168.50.56	SCTP	60	SHUTDOWN_COMPLETE
173	23.680020	192.168.50.20	192.168.50.56	SCTP	82	INIT
174	23.680674	192.168.50.56	192.168.50.20	SCTP	306	INIT_ACK
175	23.680915	192.168.50.20	192.168.50.56	SCTP	278	COOKIE_ECHO
176	23.681424	192.168.50.56	192.168.50.20	SCTP	60	COOKIE_ACK
177	23.682041	192.168.50.20	192.168.50.56	NGAP	122	NGSetupRequest
178	23.682447	192.168.50.56	192.168.50.20	SCTP	62	SACK (Ack=0, Arwnd=106438)
179	23.686114	192.168.50.56	192.168.50.20	NGAP	110	NGSetupResponse
180	23.686322	192.168.50.20	192.168.50.56	SCTP	62	SACK (Ack=0, Arwnd=106448)
198	27.626639	192.168.50.17	192.168.50.20	HTTP	488	GET /lte/live/gnb0.log?nocache=1707503544476 HTTP/1.1
200	27.627164	192.168.50.20	192.168.50.17	HTTP	404	HTTP/1.1 206 Partial Content (text/plain)
201	27.640830	192.168.50.17	192.168.50.20	HTTP	491	GET /lte/live/gnb0.log?nocache=1707503544476 HTTP/1.1
210	27.641327	192.168.50.20	192.168.50.17	HTTP	282	HTTP/1.1 206 Partial Content (text/plain)
381	55.241116	192.168.50.56	192.168.50.20	SCTP	98	HEARTBEAT
382	55.241293	192.168.50.20	192.168.50.56	SCTP	98	HEARTBEAT_ACK
397	61.118328	192.168.50.20	192.168.50.56	SCTP	98	HEARTBEAT
398	61.118925	192.168.50.56	192.168.50.20	SCTP	98	HEARTBEAT_ACK
599	85.960860	192.168.50.56	192.168.50.20	SCTP	98	HEARTBEAT
600	85.961034	192.168.50.20	192.168.50.56	SCTP	98	HEARTBEAT_ACK
648	92.862149	192.168.50.20	192.168.50.56	SCTP	98	HEARTBEAT
649	92.862723	192.168.50.56	192.168.50.20	SCTP	98	HEARTBEAT_ACK
665	100.864396	192.168.50.20	192.168.50.56	NGAP/NAS-5GS	134	InitialUEMessage, Registration request
666	100.885254	192.168.50.56	192.168.50.20	NGAP/NAS-5GS/EAP	222	SACK (Ack=1, Arwnd=106496) , DownlinkNASTransport, Authentication request, Request, UMTS Authentication and Key Agreement' EAP (EAP-AKA')
668	100.909374	192.168.50.20	192.168.50.56	NGAP/NAS-5GS/EAP	138	SACK (Ack=1, Arwnd=106496) , UplinkNASTransport, Authentication response, Response, UMTS Authentication and Key Agreement' EAP (EAP-AKA')
669	100.916527	192.168.50.56	192.168.50.20	NGAP/NAS-5GS/EAP	114	SACK (Ack=2, Arwnd=106496) , DownlinkNASTransport, Authentication reject, Failure
670	101.017353	192.168.50.56	192.168.50.20	NGAP	134	DATA (TSN=2) (retransmission) , UEContextReleaseCommand
671	101.017574	192.168.50.20	192.168.50.56	SCTP	66	SACK (Ack=3, Arwnd=106476)
672	101.017696	192.168.50.20	192.168.50.56	NGAP	102	UEContextReleaseComplete
673	101.221676	192.168.50.56	192.168.50.20	SCTP	62	SACK (Ack=3, Arwnd=106496)

Fig 4.61 Wireshark after changing Authentication Method

Logs of Wireshark: Fig 4.62 gives the brief description of Authentication request from 5G core to gNB:

```
Frame 666: 222 bytes on wire (1776 bits), 222 bytes captured (1776 bits) on interface \Device\NPF_{B9E46E75-E625-4E6F-885B-963BFE51B8B6}, id 0
Ethernet II, Src: PcsCompu_5e:6e:3b (08:00:27:5e:6e:3b), Dst: ASUSTekC_9c:6d:be (7c:10:c9:9c:6d:be)
Internet Protocol Version 4, Src: 192.168.50.56, Dst: 192.168.50.20
Stream Control Transmission Protocol, Src Port: 38412 (38412), Dst Port: 42664 (42664)

NG Application Protocol (DownlinkNASTransport)
  NGAP-PDU: initiatingMessage (0)
    initiatingMessage
      procedureCode: id-DownlinkNASTransport (4)
      criticality: ignore (1)
      value
        DownlinkNASTransport
          protocolIEs: 3 items
            Item 0: id-AMF-UE-NGAP-ID
              ProtocolIE-Field
                id: id-AMF-UE-NGAP-ID (10)
                criticality: reject (0)
                value
                  AMF-UE-NGAP-ID: 1
            Item 1: id-RAN-UE-NGAP-ID
              ProtocolIE-Field
                id: id-RAN-UE-NGAP-ID (85)
                criticality: reject (0)
                value
                  RAN-UE-NGAP-ID: 3
            Item 2: id-NAS-PDU
              ProtocolIE-Field
                id: id-NAS-PDU (38)
                criticality: reject (0)
                value
                  NAS-PDU: 7e00560002000078006c015c006c3201000001050000c122b9e525ec57d4fc26d261b186…
                    Non-Access-Stratum 5GS (NAS)PDU
                      Plain NAS 5GS Message
                        Extended protocol discriminator: 5G mobility management messages (126)
                        0000 .... = Spare Half Octet: 0
                        .... 0000 = Security header type: Plain NAS message, not security protected (0)
                        Message type: Authentication request (0x56)
                        0000 .... = Spare Half Octet: 0
                        NAS key set identifier - ngKSI
                          .... 0... = Type of security context flag (TSC): Native security context (for KSIAMF)
                          .... .000 = NAS key set identifier: 0
                        ABBA
                          Length: 2
                          ABBA Contents: 0000
                        EAP message
                          Element ID: 0x78
                          Length: 108
                          Extensible Authentication Protocol
```

Fig 4.62 Authentication Request EAP_AKA

Logs of Wireshark fig 4.63 Authentication response gNB to 5G core

```
Frame 668: 138 bytes on wire (1104 bits), 138 bytes captured (1104 bits) on interface \Device\NPF_{B9E46E75-E625-4E6F-885B-963BFE51B8B6}, id 0
Ethernet II, Src: ASUSTekC_9c:6d:be (7c:10:c9:9c:6d:be), Dst: PcsCompu_5e:6e:3b (08:00:27:5e:6e:3b)
Internet Protocol Version 4, Src: 192.168.50.20, Dst: 192.168.50.56
Stream Control Transmission Protocol, Src Port: 42664 (42664), Dst Port: 38412 (38412)
NG Application Protocol (UplinkNASTransport)
  NGAP-PDU: initiatingMessage (0)
    initiatingMessage
      procedureCode: id-UplinkNASTransport (46)
      criticality: ignore (1)
      value
        UplinkNASTransport
          protocolIEs: 4 items
            Item 0: id-AMF-UE-NGAP-ID
              ProtocolIE-Field
                id: id-AMF-UE-NGAP-ID (10)
                criticality: reject (0)
                value
                  AMF-UE-NGAP-ID: 1
            Item 1: id-RAN-UE-NGAP-ID
              ProtocolIE-Field
                id: id-RAN-UE-NGAP-ID (85)
                criticality: reject (0)
                value
                  RAN-UE-NGAP-ID: 3
            Item 2: id-NAS-PDU
              ProtocolIE-Field
                id: id-NAS-PDU (38)
                criticality: reject (0)
                value
                  NAS-PDU: 7e0057780008025c000832020000
                    Non-Access-Stratum 5GS (NAS)PDU
                      Plain NAS 5GS Message
                        Extended protocol discriminator: 5G mobility management messages (126)
                        0000 .... = Spare Half Octet: 0
                        .... 0000 = Security header type: Plain NAS message, not security protected (0)
                        Message type: Authentication response (0x57)
                        EAP message
                          Element ID: 0x78
                          Length: 8
                          Extensible Authentication Protocol
            Item 3: id-UserLocationInformation
              ProtocolIE-Field
                id: id-UserLocationInformation (121)
                criticality: ignore (1)
                value
                  UserLocationInformation: userLocationInformationNR (1)
                    userLocationInformationNR
                      nR-CGI
                        pLMNIdentity: 00f110
                        nRCellIdentity: 0x0001234502
                      tAI
                        pLMNIdentity: 00f110
                        tAC: 100 (0x000064)
```

Fig 4.63 Authentication response EAP_AKA

Log of Wireshark fig 4.64 Authentication reject 5GC to gNB

```
Frame 669: 114 bytes on wire (912 bits), 114 bytes captured (912 bits) on interface \Device\NPF_{B9E46E75-E625-4E6F-885B-963BFE51B8B6}, id 0
Ethernet II, Src: PcsCompu_5e:6e:3b (08:00:27:5e:6e:3b), Dst: ASUSTekC_9c:6d:be (7c:10:c9:9c:6d:be)
Internet Protocol Version 4, Src: 192.168.50.56, Dst: 192.168.50.20
Stream Control Transmission Protocol, Src Port: 38412 (38412), Dst Port: 42664 (42664)
NG Application Protocol (DownlinkNASTransport)
    NGAP-PDU: initiatingMessage (0)
        initiatingMessage
            procedureCode: id-DownlinkNASTransport (4)
            criticality: ignore (1)
            value
                DownlinkNASTransport
                    protocolIEs: 3 items
                        Item 0: id-AMF-UE-NGAP-ID
                            ProtocolIE-Field
                                id: id-AMF-UE-NGAP-ID (10)
                                criticality: reject (0)
                                value
                                    AMF-UE-NGAP-ID: 1
                        Item 1: id-RAN-UE-NGAP-ID
                            ProtocolIE-Field
                                id: id-RAN-UE-NGAP-ID (85)
                                criticality: reject (0)
                                value
                                    RAN-UE-NGAP-ID: 3
                        Item 2: id-NAS-PDU
                            ProtocolIE-Field
                                id: id-NAS-PDU (38)
                                criticality: reject (0)
                                value
                                    NAS-PDU: 7e0058780004045c0004
                                        Non-Access-Stratum 5GS (NAS)PDU
                                            Plain NAS 5GS Message
                                                Extended protocol discriminator: 5G mobility management messages (126)
                                                0000 .... = Spare Half Octet: 0
                                                .... 0000 = Security header type: Plain NAS message, not security protected (0)
                                                Message type: Authentication reject (0x58)
                                                EAP message
                                                    Element ID: 0x78
                                                    Length: 4
                                                    Extensible Authentication Protocol
```

Fig 4.64 Authentication reject EAP_AKA

This issue occurred because the AUSF and UDM were unable to authenticate the UE. This was due to the UDM checking the data from the Free5GC WebGUI, where there was a mismatch in the values of OPC, SQN, and the Authentication Management Field (AMF). Free5GC does not support the AMF value '9001' and the SQN value '000000000000'. The most significant problem was that Free5GC does not support the SIM algorithm 'Milenage', whereas the Amarisoft UE supports the SIM algorithm 'XOR'. As a result of this issue, NAS authentication failed; in the NAS Authentication Request, half of the AUTN values were correct, while the other half were incorrect. Due to this issue, needed to configure a new SIM card on Ubuntu desktop. However, after installing Ubuntu, the system became corrupt, and encountered an 'ADM key error' message.

5 Summary

The project aims to integrate Amarisoft's Callbox with the Free5GC implementation to establish a fully functional 5G Standalone (SA) network. Several objectives have been successfully achieved during the project timeline.

At the outset of the project, the first step involved gaining a comprehensive understanding of 5G technology and its core architecture. Concurrently, efforts were made to implement the software-defined Free5GC platform using resources from its GitHub repository. Initially, the project's scope was somewhat unclear, leading to the implementation of some components that were later deemed unnecessary. However, as the project progressed, its objectives became clearer.

One key task was configuring the Access and Mobility Management Function (AMF) within Free5GC and the gNodeB (gNB) within the Amarisoft Callbox to verify the establishment of an NGAP/SCTP connection. Following the successful establishment of this connection, the next step involved testing the Non-Access Stratum (NAS) functionality for User Equipment (UE). A challenge arose due to a compatibility issue between the SIM card algorithms used by Amarisoft Callbox, which accepts the 'XOR' algorithm, and Free5GC, which requires the 'Milenage' algorithm. This discrepancy prevented UEs from authenticating with the 5G core components, including the Authentication Server Function (AUSF), Unified Data Management (UDM), and Free5GC's WebGUI.

During this project some difficulties faced and some criticism for own work such as not thinking about the SIM algo’s. Everytime working in MME (that was not relevant), waste time to find the solution such as what is MAC failure, what is T3560 timer, how its works between AMF and UEs. What is the difference between Authentication Method like (5G_AKA and EAP_AKA_PRIME). This took 2-3 months but at the end the problem was regarding the SIM algo.

At the end, the project objective was not achieved, some results are successful establishment.

6 Abbreviations

0

…

3

3G Third Generation

3GPP Third Generation Partnership Project

…

4

4G Fourth Generation

…

5

5G Fifth Generation

A

AF Application Function

AKA Authentication and Key Agreement

AMF Access and Mobility Management Function

API Application Programming Interface

APN Access Point Name

ARP Address Resolution Protocol

AUSF Authentication Server Function

…

C

CLI Command Line Interface

CN Core Network

CP Control Plane

…

D

DB Database

DC Data Center

DHCP Dynamic Host Configuration Protocol

DL Downlink

…

E

EIA Encryption Integrity Algorithm

ENB Evolved Node B

EAP_AKA Extensible Authentication Protocol Authentication and Key Agreement

EPC Evolved Packet Core

EPS Evolved Packet System

ETSI European Telecommunications Standards Institute

…

F

FDD Frequency Division Duplex

FR Frequency Range

FTP File Transfer Protocol

…

G

gNB gNodeB

GUAMI Globally Unique AMF Identifier

GSM Global System for Mobile communications

GUI Graphical User Interface

GW Gateway

…

H

HSS Home Subscriber Server

HTML Hypertext Mark-up Language

HTTP Hypertext Transfer Protocol

HW Hardware

…

I

IMEI International Mobile Equipment Identity

IMEISV International Mobile Equipment Identity Software Version

IMPI IP Multimedia Private Identity

IMPU IP Multimedia Public Identity

IMS IP Multimedia Subsystem

IMSI	International Mobile Subscriber Identity
IMT	International Mobile Telecommunications
IN	Intelligent Network
IP	Internet Protocol
IT	Information Technology
ITU	International Telecommunication Union

...

L

LAN	Local Area Network
LTE	Long-Term Evolution
LTEENB	Long-Term Evolution Evolved Node B
LTEIMS	Long-Term Evolution IMS
LTEMME	Long-Term Evolution Mobility Management Entity
LTENB	Long-Term Evolution Node B
LTEUE	Long-Term Evolution User Equipment
LTS	Long-Term Support

…

M

MAC	Media Access Control
MAC	Message Authentication Code
MBMSGW	Multimedia Broadcast Multicast Service Gateway
MEC	Multi-Access Edge Computing
MIMO	Multiple-Input Multiple-Output
MME	Mobility Management Entity
MNC	Mobile Network Code
MNO	Mobile Network Operator
MTBF	Mean Time Between Failures
MTC	Machine-Type Communication

…

N

NAS	Non-Access Stratum
NEF	Network Exposure Function
NF	Network Function
NFV	Network Function Virtualization

NFVI Network Function Virtualization Infrastructure

NFVO Network Function Virtualization Orchestrator

NG Next Generation

NGAP Next Generation Application Protocol

NGC Next Generation Core

NR New Radio

NRF Network Repository Function

NS Network Slice

NSA Non-Standalone

NSSAI Network Slice Selection Assistance Information

NSSF Network Slice Selection Function

…

O

OS Operating System

OSS Operations Support Systems

…

R

RAM Random Access Memory

RAN Radio Access Network

RAND Random Number

RAT Radio Access Technology

RF Radio Frequency

RFC Request for Comments

RJ45 Registered Jack 45

RRC Radio Resource Control

RRH Remote Radio Head

…

S

SA Standalone
SBA Service-Based Architecture
SBI Service-Based Interface
SCP Service Control Point
SCTP Stream Control Transmission Protocol
SDF Service Data Flow
SDN Software-Defined Networking
SDR Software-Defined Radio
SFC Service Function Chaining

SFTP Secure File Transfer Protocol
SIM Subscriber Identity Module
SIP Session Initiation Protocol
SM Session Management
SMF Session Management Function
SMS Short Message Service
SMSF Short Message Service Function
SOS Satellite Phone Emergency Service
SQL Structured Query Language
SUCI Subscription Concealed Identifier
SUPI Subscription Permanent Identifier
SW Software
SYNC Synchronize
…
T
TAC Tracking Area Code
TCP Transmission Control Protocol
TDD Time Division Duplex
…
U
UDM Unified Data Management
UDP User Datagram Protocol
UDR Unified Data Repository
UDSF Unified Data Storage Function
UE User Equipment
UL Uplink
UMTS Universal Mobile Telecommunications System
UP User Plane
UPF User Plane Function
URI Uniform Resource Identifier
URL Uniform Resource Locator
URLLC Ultra-Reliable Low Latency Communication
USIM Universal Subscriber Identity Module
…
V
VM Virtual Machine
VNF Virtual Network Function
VPLMN Visited Public Land Mobile Network
…
W
WA Wide Area
WLAN Wireless Local Area Network
…
X
XML Extensible Markup Language
XOR Exclusive OR

7 References

1. 5G; NG-RAN; Architecture description (3GPP TS 38.401 version 15.2.0 Release 15) (2024) [online] https://www.etsi.org/deliver/etsi_ts/138400_138499/138401/15.02.00_60/ts_138401v150200p.pdf [accessed 12 January 2024]
2. 5G; Non-Access-Stratum (NAS) protocol for 5G System (5GS); Stage 3 (3GPP TS 24.501 version 15.0.0 Release 15) (2023) [online]; https://www.etsi.org/deliver/etsi_ts/124500_124599/124501/15.00.00_60/ts_124501v150000p.pdf [accessed 10 October 2023- 23 December 2023]
3. 5G; Procedures for the 5G System; (3GPP TS 23.502 version 15.3.0 Release 15) (2023/2024) [online]; https://www.etsi.org/deliver/etsi_ts/123500_123599/123502/15.03.00_60/ts_123502v150300p.pdf [accessed 02 October 2023 – 10 February 2024]
4. 5G; System architecture for the 5G System (5GS) (3GPP TS 23.501 version 16.6.0 Release 16) (2023/2024) [online] https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/16.06.00_60/ts_123501v160600p.pdf [accessed 16 September 2023 – 10 February 2024]
5. Amarisoft Tech Academy (2023): Amari Callbox Classic User guide [offline] [accessed 10 September 2023- 20 Nov ember 2023]
6. Amarisoft Tech Academy (2023/2024): AMARI Callbox Classic [online] https://www.amarisoft.com/test-and-measurement/device-testing/device-products/amari-callbox-classic [accessed 10 September 2023 - 10 February 2024]
7. Amarisoft Tech Academy (2023/2024): Amarisoft MME + Open5GS [online] https://tech-academy.amarisoft.com/MME_Open5GS.html [accessed 10 September 2024 – 10 February 2024]
8. Amarisoft Tech Academy (2023/2024): LTE and NR Core Network version [online] https://tech-academy.amarisoft.com/ltemme.doc#prop.nas_integ_algo_pref [accessed 10 September 2023 - 10February 2024]
9. Amarisoft Tech Academy (2024): Amari Callbox Classic User guide [online] https://tech-academy.amarisoft.com/userguide_callbox_classic.doc [accessed 10 February 2024]
10. Amarisoft Tech Academy (2024): How to program Amarisoft test sim card, [online] https://tech-academy.amarisoft.com/update.wiki [accessed 25 January 2024]
11. Amarisoft Tech Academy: LTE Software eNodeB and NR Software gNB (2023) [offline] [accessed 10 September 2023 – 29 November 2023]
12. Amarisoft Tech Academy: LTE Software eNodeB and NR Software gNB (2024) [online] https://tech-academy.amarisoft.com/lteenb.doc [accessed 10 February 2024]
13. Cisco Session Timer (2023) [offline] https://www.cisco.com/c/en/us/td/docs/wireless/ucc/amf/2021-04/config-and-admin/b_ucc-5g-amf-config-and-admin-guide_2021-04/m_timers.pdf [accessed 15 December 2023]
14. Free5GC Forum: Running free5gc stage3 with Amarisoft gNodeB + UE (2023/2024) [online] https://forum.free5gc.org/t/running-free5gc-stage3-with-amarisoft-gnodeb-ue/532 [accessed 08 November2023- 31 January 20240]
15. Free5GC Github (2023/2024) [online]: https://github.com/free5gc/free5gc [accessed 20 August 2023 – 10 February2024]
16. Github: UE-non3GPP (2024) [online] https://github.com/LABORA-INF-UFG/UE-non3GPP [accessed 16 January 2024]
17. GSMTS; ETSI; Digital cellular telecommunications system (Phase 2+); Specification of the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface; (GSM 11.11) (2024) [online] https://www.etsi.org/deliver/etsi_gts/11/1111/05.03.00_60/gsmts_1111v050300p.pdf [accessed 22 January 2024]
18. Maria Majid; Master Thesis; Analyzation and Implementation of 5G Environment based on Amarisoft Callbox (2023/2024) [offline] [accessed 20 August 2023 – 10 February 2024]
19. Pysim (2024) [online] https://github.com/osmocom/pysim [20 January 2024 - 10February 2024]
20. Pysim (2024): [online] https://github.com/aldebjer/pysim/blob/master/doc/building_on_linux.rst [25 Januaery 2024]
21. Sharenote (2023/2024): 5G/NR Network - NGAP/N2 Interface [online] https://www.sharetechnote.com/html/5G/5G_NGAP.html [accessed 10 October 2023 – 10 February 2024]

22. Sharenote(2023/2024): Free5GC and 5G/NR [online] https://www.sharetechnote.com/html/5G/Handbook_5G_Index.html , https://www.sharetechnote.com/html/OpenRAN/OR_free5GC_Run.html#PDU_Session_Establishment [accessed 20 August 2023 – 10 Febuary 2024]
23. Sharenote: 4G/LTE (2023) [online]; https://www.sharetechnote.com/html/Handbook_LTE_Authentication-Failure.html [accessed 05 November 2023 – 24 December2023]
24. Sharenote: 5G/NR-NAS (2023) [online]; https://www.sharetechnote.com/html/5G/5G_Authentication.html [accessed 05 November 2023 – 24 December 2023]
25. Tech-invite 3GPP space IETF space (203/2024) [online]; https://www.tech-invite.com/3m24/toc/tinv-3gpp-24-501_s.html [accessed 10 November 2023 – 20 January 2024]
26. Thomas Zielasny; Implementieren und Testen eines 5G-Core-Netzes mit Free5GC; (2023) [offline] [accessed 20 August 2023 – 10 October 2023]
27. Ulrich Trick; 5G An Introduction to the 5th Generation Mobile Networks; 2021; (2023) [offline] [accessed 19 August 2023 – 30 October 2023]
28. Ulrich Trick; 5G The 5th Generation Mobile Networks; 2nd Edition 2024; (2024) [offline] [accessed 01 February2024]
29. Universal Mobile Telecommunications System (UMTS); LTE; Common test environments for User Equipment (UE); Conformance testing,(3GPP TS 34.108 version 15.0.0 Release 15) (2023/2024) [online] https://www.etsi.org/deliver/etsi_ts/134100_134199/134108/15.00.00_60/ts_134108v150000p.pdf [accessed 16 December 2023 – 10 February 2024]