Mobile Computing Project Report

(WS 2023/2024)

TeamSlicex

Members:

Member#1: Mashnunul Huq

ID: 1384042

Member#2: Anurag De
ID: 1400450

Member#3: Sourav Paul Sumit

ID: 1344118

Member#4: Erdi Tras

ID: 1427187

Guidance

Prof. Dr. Armin Lehmann

ACKNOWLEDGEMENT

With the initial guidance of Professor Dr. Armin Lehmann (Frankfurt University of Applied Sciences), this project started the path of completion. Shigeru Ishida's Open5GS helped us out tremendously during the course of our project execution, and we owe them a debt of gratitude. They serve as our project mentors, giving us the knowledge and direction we needed to finish the project and meet our objectives.

ABSTRACT

5G networks offer high-speed data rates and support a wide range of applications due to their advanced capabilities. The dynamic and adaptable features of network slicing can greatly enhance the architecture of a 5G network. The primary aim of implementing network slicing in a 5G network is to enhance flexibility and adaptability to cater to modern network applications. In our project, we have deployed a 5G Stand Alone (SA) network utilizing Open5GS as the core and UERANSIM for the 5G-RAN. Our main objective is to enable network slicing to facilitate File Sharing, Streaming, and VOIP calling services for two distinct tenant groups. Initially, we employed Open5GS and SRSRAN for the implementation; however, due to software limitations, SRSRAN was found incompatible with network slicing. Consequently, UERANSIM was utilized in place of SRSRAN to support network slicing. Additionally, we configured Kamailio, Next Cloud, JellyFin servers in separate VMs to provide VOIP Calling, File Sharing, and Streaming services, respectively. This setup allowed the tenant groups to access the services through dedicated slices tailored to their specific requirements.

INTRODUCTION

The fundamental principles and technological advancements of 5G distinguish it from earlier generations of mobile communication networks. Central to the concept of separating control and user planes are two crucial elements: network slicing and CUPS. Additionally, Network Slicing is anticipated to establish multiple logical (virtual) networks, known as "Network slices," on the same underlying physical network infrastructure. Each network slice can be tailored to serve diverse consumers or services. To implement the principles of CUPS and Network Slicing in the 5G network, it is imperative to integrate Network Function Virtualization, Software Defined Networking, and Cloud-Native technologies into the core network design. These technologies are aimed at enhancing the adaptability, scalability, and flexibility of the 5G core network.

SCOPE

The aim of this project is to construct a 5G Stand-alone (SA) system utilizing Open5GS, capable of delivering various services between hosts and effectively segmenting the data network. This project involves the utilization of both UERANSIM and Open5GS. The envisioned concept holds the establishment of SIP sessions, facilitates of file sharing, and enables streaming between designated hosts within each network slice, while concurrently preventing communication between servers located in distinct network slices. The concept of network slicing is central key to this project. Network slicing allows for the creation of multiple virtual networks (slices) on top of a shared physical infrastructure. Each slice can be customized to meet the specific requirements of different applications, users or services. By employing network slicing, the project aims to ensure that communication between hosts within the same network slice is seamless and efficient while maintaining isolation and security between slices through SST and SD tags.

DEFINITIONS AND ABBREVIATIONS

• VirtualBox: VirtualBox is a powerful, open-source virtualization software package developed by Oracle. It allows users to run multiple operating systems (OS) simultaneously on a single physical machine. This is achieved by creating virtual machines (VMs) within the VirtualBox software, each of which can run its own guest OS independently of the host OS. VirtualBox is available for multiple host operating systems including Windows, macOS, Linux, and Solaris. It also supports a wide range of guest OSes, including various versions of Windows, Linux distributions, macOS, and others. We used Microsoft Windows 11 as our base OS and Ubuntu 22.04 as OS of all the VirtualBox

• Open5GS: Open5GS is an open-source project that focuses on the implementation of a 5G mobile core network, implementing 5GC and EPC using C-language. Its main goal is to implement the 5G Core network and support it by enabling 5G Core network functions. Open5GS is designed to be interoperable with existing network infrastructure and standards compliant. It provides interfaces for integration with legacy 4G/LTE networks as well as interfaces for interconnecting with external networks and services.

• UERANSIM: It is an open-source implementation of the User Equipment (UE) side of the 5G Radio Access Network (RAN). Developed as part of the OpenAirInterface project, UERANSIM simulates the behavior of a 5G UE, allowing developers, researchers, and network operators to test and experiment with various aspects of 5G networks in a controlled environment. It is free and open-source state of the art featuring both UE and eNodeB/gNodeB applications. It can be integrated with third-party core network solutions (like Open5GS) to build complete end-to-end mobile wireless networks.

• Interface: Interface is the logical link established between two instances of a logical network thus providing connection.

• IP Forwarding: IP forwarding refers to an operating system's capacity to accept incoming network packets on one interface, identify that they are not intended for the system itself, but rather for transmission to another network, and then forward them appropriately.

• NAT Translation: Network address translation involves changing the network address information in packets' IP headers as they travel via a traffic routing device to translate one IP address space into another. VirtualBox's NAT (Network Address Translation) networking is a type of network configuration that allows virtual machines (VMs) running within VirtualBox to communicate with the external network and the internet through the host system's (Windows 11) network connection. VirtualBox includes a built-in DHCP server for NAT networking, which automatically assigns IP addresses to VMs configured to use NAT. This simplifies the setup process and allows VMs to obtain network configurations dynamically.

5G SA	5G Stand Alone
AMF	Access and Mobility Management Function
5QI	5G QoS Identifier
NRF	Network Repository Function
UE	User Equipment
NSSF	Network Slice Selection Function
PDU	Protocol Data Unit
NGAP	Next Generation Application Protocol
SMF	Session Management Function
SD	Slice Differentiator
SST	Slice/Service Type
SDN	Software Defined Network
UPF	User Plane Function
RAN	Radio Access Network
PCF	Policy Control Function

THEORETICAL FOUNDATIONS

In this project, the approach to achieving the project's goals is thoroughly examined and explained. The project relies on the utilization of UERANSIM to realize essential functionalities such as VoIP communication, file sharing, and streaming services across different user groups or "tenant

groups". A crucial concept employed in enabling these services is "Network slicing", which involves partitioning the network into multiple virtual networks tailored to specific use cases or user groups. However, it's emphasized that the implementation of network slicing is feasible only with UERANSIM, as srsRAN lacks compatibility for this functionality in some cases with Open5GS.

Furthermore, this section provides a brief but informative overview of how all services are implemented and the underlying architectural mechanisms governing their operation. It sheds light on the technical intricacies involved in realizing VoIP, file sharing, and Video streaming services within the context of the project's objectives. Overall, this section serves as a comprehensive guide to understanding the strategies and methodologies employed in achieving the desired outcomes of the project.

5G CORE

The 5G Core (5GC) serves as the central intelligence of a 5G network, overseeing both data and control plane operations. Among its diverse functions, the 5G core manages data traffic, establishes connections with User Equipment (UE), delivers essential network services, and enhances security measures. Its fundamental architecture comprises two distinct functional planes: the Control Plane and the User Plane. These planes operate independently to facilitate flexible scalability and dynamic deployments, a concept known as Software-Defined Networking (SDN). The User Plane consists of three primary components: UE, Radio Access Network, and User Plane Function (UPF). The UE connects to the UPF via the RAN, effectively acting as a gateway to the Internet. The UPF serves as a gateway to other networks and is responsible for Quality of Service (QoS) management. Conversely, the Control Plane encompasses functions such as the Access and Mobility Management Function (AMF), which manages access control, and the Session Management Function (SMF), which oversees session management between the UE and UPF. Various open-source projects are available for implementing 5G Stand-Alone (SA) networks, one of which is Open5gs, utilized in this project's implementation. Open5gs implements 16 network functions including AMF, SMF, PCF, UDM, AUSF, NRF, NSSF, UDR, and UPF.

RADIO ACCESS NETWORKS (RAN)

The Radio Access Network (RAN) plays a vital role in wireless communication systems, establishing radio connections between individual devices and different parts of the network. It facilitates connections via devices like cellphones, computers, or remotely controlled machines, utilizing either fiber or wireless connections. In this project, connection experimentation involves only UERANSIM. UERANSIM is an open-source project consisting of two essential components: gNodeB and UE. The gNodeB connects to the Access and Mobility Management Function (AMF) and manages user traffic on simulated radio links, while the UE acts as a subscriber. Like srsRAN

it doesn’t have ENB nodes. But UERANSIM works flawlessly with Open5GS but srsRAN has some compatibility issues with Open5GS Core.

NETWORK SLICING

The primary enabling technology in the era of 5G, 'Network Slicing', facilitates the sharing of a single telecommunication network infrastructure by multiple logical networks, known as network slices. It enhances the flexibility, efficiency of resource allocation, and security of 5G networks, allowing 5G to cater to a wide range of service requirements. Network slicing enables the support of various mobile broadband applications within one slice while dedicating another slice to specific non-mobile applications, all utilizing shared physical resources. Despite utilizing a common physical Core and radio network, network slices appear as distinct networks from the user's perspective, thanks to the virtualization of network functions. In 5G, network slicing ensures isolation and caters to diverse Quality of Service (QoS) needs for each end-to-end network sharing the same physical resources. Through the creation of multiple virtual end-to-end networks, or network slices, virtualization technologies like Software-Defined Networking (SDN) and Network Function Virtualization (NFV) empower 5G network operators to offer customized networks tailored to specific functional, QoS, and user-specific requirements. NFV allows the isolation of each network slice by implementing network functions (NF) within it, ensuring separation from other network slices. SDN, on the other hand, manages the QoS flow for each network slice once it is established, enabling the monitoring and enforcement of QoS standards. Essentially, network slicing enables the creation of numerous virtual networks on a shared physical infrastructure. The allocation of capacity in this virtualized network environment is dynamically adjusted based on demand, allowing for efficient and interactive utilization of resources.

NEXT CLOUD

Nextcloud is a cloud storage platform renowned for its capability to establish and utilize file hosting services. Despite being freely available and open source, Nextcloud enables installation and usage on private servers by anyone. It operates akin to Dropbox but does not offer on-premises file storage hosting as a service. Nextcloud encompasses a plethora of features including file sharing, messaging, shared document editing, and a private information manager. It introduces a broad spectrum of enhancements, including improved performance in file backend and extensive collaboration functionalities in text, chat, groupware, and files. Nextcloud can be deployed to cater to various demands, ranging from single-board computers to vast data centers servicing millions of users. Its sophisticated authorization mechanism and federation option render it highly suitable for corporate environments.

Nextcloud Hub introduces several key enhancements: Firstly, it implements a high-performance backend for Nextcloud Files, significantly reducing server demand from desktop clients and web interface polling while delivering fast notifications to users. Secondly, it incorporates various performance upgrades aimed at accelerating page loads and alleviating the server's workload. Thirdly, it introduces collaboration features to enhance team productivity, including a new Whiteboard and author colors in Text and Document Templates. Fourthly, Nextcloud Talk introduces message status indications, hand-raising feature, group conversation descriptions, and more. Lastly, numerous Groupware updates are introduced, such as drag-and-drop functionality, improved threading in Mail, and synchronization of social media avatars with contacts.

JellyFin

Jellyfin is an open-source media server software that allows users to organize, manage, and stream their personal collection of movies, TV shows, music, and photos. It provides a platform for users to access their media files from various devices, including smart TVs, mobile phones, tablets, and computers, regardless of their location. With Jellyfin, users can create customizable libraries, organize their media content with metadata, and enjoy features like transcoding for smooth streaming across different devices and bandwidths. Additionally, Jellyfin offers support for user authentication and parental controls, ensuring a secure and family-friendly streaming experience. As a free and community-driven alternative to commercial media servers, Jellyfin empowers users to take control of their media consumption and create their own personalized streaming ecosystem.

DESIGN AND IMPLEMENTATION

DESIGN

General Environment setup:

The following packages and tools are needed to complete this project and construct a large network.

1. Linux Based Virtual Machine

The fundamental configuration involves the installation of Oracle VM Virtual Box (7.0), which offers cross-platform virtualization and enables the simultaneous use of various operating systems. The installation process uses the Ubuntu 22.04 iso file. We did not use any containerization so docker or LXC is not required. Here two set up were made because group mates worked parallelly with two set ups. Two group mates (#1 and #2) worked with VoIP SIP calling and Video Streaming and two group mates (#3 and #4) worked with Nextcloud file sharing.

2. Kamailio

For SIP VoIP Calling Kamailio v5.5.1 is installed in the system. This feature will provide a SIP calling service for desired network slice.

3. Linphone

To support VoIP calls, the Linphone software package v4.4.21 is installed in the end user UERANSIM VM's were GNB and UE's are situated.

4. Wireshark

For traffic collecting and analysis, we required all VM's to have Wireshark v3.2.3 installed as a packet capture tool.

5. JellyFin

To achieve video streaming services, JellyFin v10.8.13 is installed in first UE Container of the kamailio server setup.

6. Next cloud

To simulate a file sharing platform, Next Cloud of v28 is installed.

7. Open5GS

To establish the 5G core part in this project, Open5GS v2.7.0 is installed which will act as a control plane of the 5G core network.

8. UERANSIM (v3.2.6)

UERANSIM v3.2.6 is downloaded from github which comprises of multiple UEs and gNodeB.

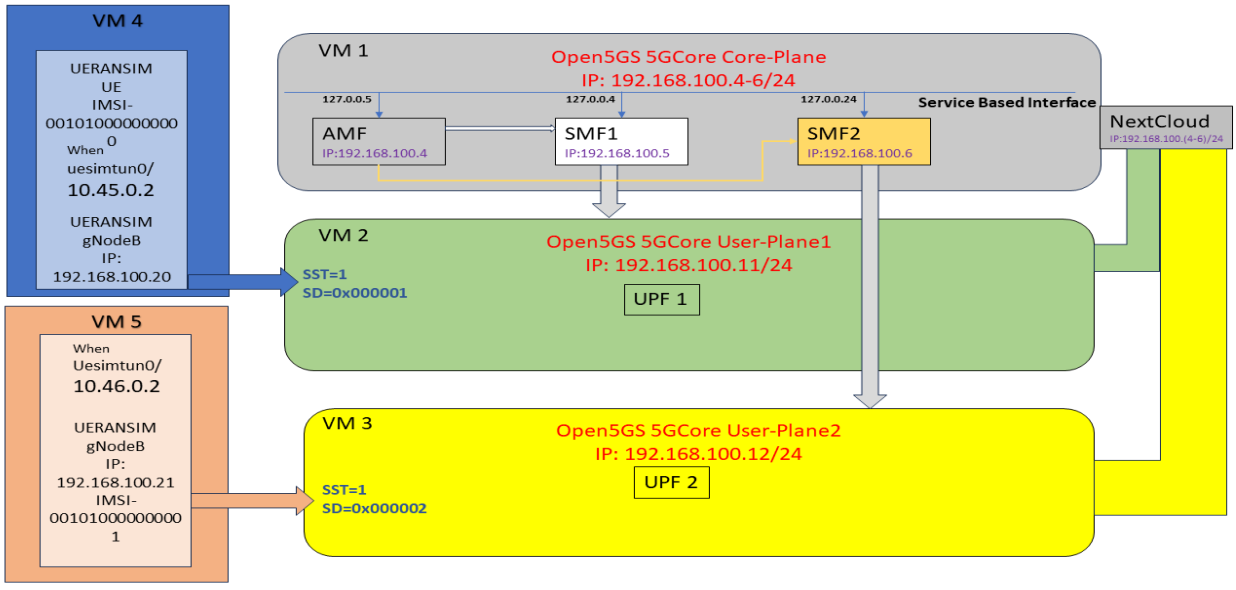

Figure 1: Architectural design of 5G network for File Sharing and internet pinging using Open5gs & UERANSIM

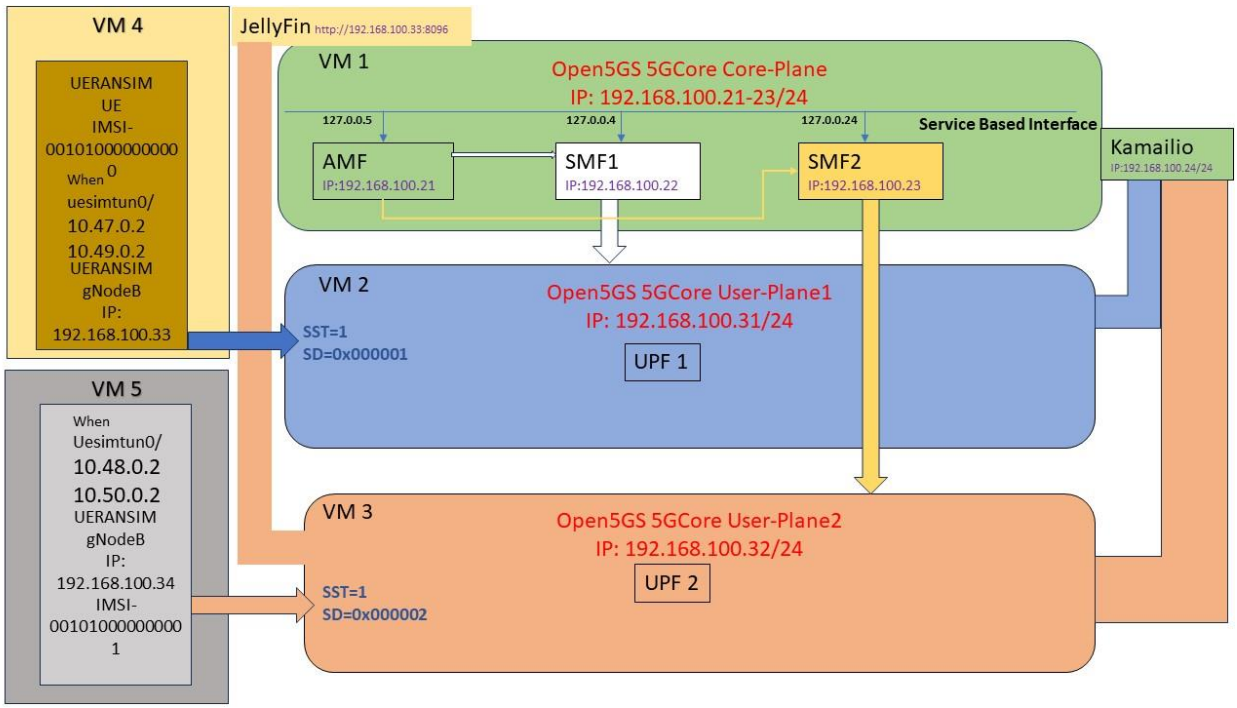

Figure 2: Architectural design of 5G network for VoIP Calling and Video Streaming using Open5GS & UERANSIM

The above figure illustrates the 5G network topology implemented in the project to provide the File sharing, Video streaming and VoIP services to the UEs using network slicing.

Architecture Explanation:

1. All network components are configured in different VMs' with details mentioned in Table 1.

2. Four UEs were created in separate VMs with details mentioned in table. We have provided separate slicing to individual User Equipment to access the File sharing, streaming and VoIP calling services. The corresponding details of services available to a particular UE.

3. As per the project requirement UE1 and UE2 (of MultiVMApproach Folder Configuration in github) have access to File Sharing, UEConfigs and UEConfigs2 (of MultiVMForKamailioServer Folder Configuration in github) have VoIP calling service, UE3 and UE4 (of MultiVMForKamailioServer Folder Configuration in github) Streaming services and associated with UPF-1 and UPF-2. UE3 and UE4 have access to VoIP calling service and associated with UPF-3 and UPF-4. Then UE5 and UE6 have access to VoIP calling service and associated with UPF-3 and UPF-4

4. Each of the four SMF files are used for network slice selection as per the slicing parameters provided by the UEs. The details of association of each SMF and their corresponding slicing information is provided in the below Table 2.

5. The data network consists of NextCloud server for providing File sharing service to UE1 and UE2 installed in MainCore VMs corresponding to MultiVMApproach Folder Configuration in

github. JellyFIn server for providing live streaming service to UE5 and UE6 installed in MainCore2 VMs corresponding to MultiVMForKamailioServer Folder Configuration in github. Kamailio server for providing VoIP calling server to UE3 and UE4 installed in MainCore2 VMs corresponding to MultiVMForKamailioServer Folder Configuration in github, details mentioned in Table 4.

6. Each slice is associated with the different Data Network subnet to access the services as mentioned in Table 3.

7. We have used standard 5QI values for providing the File sharing, streaming and VoIP

Below table shows the role and configurations of each VM's installed for this project:

Table 1: VM's role and configuration setup

VM For	SW and Role	IP Address/url
Nextcloud	NextCloud v28 for File sharing	192.168.100.4/nextcloud
Open5G Core (Machine 1)	Open5GS v2.7.0 5GC C-Plane	192.168.100.4-6
UERANSIM (Machine 1)	UERANSIM v3.2.6 UE	192.168.100.20
UERANSIM (Machine 2)	UERANSIM v3.2.6 UE	192.168.100.21
UPF 1	Open5GS v2.7.0 5GC U-Plane1	192.168.100.11
UPF 2	Open5GS v2.7.0 5GC U-Plane2	192.168.100.12
Open5G Core (Machine 2)	Open5GS v2.7.0 5GC U-Plane1	192.168.100.21-23
UERANSIM (Machine 3)	UERANSIM v3.2.6 UE	192.168.100.33
UERANSIM (Machine 4)	UERANSIM v3.2.6 UE	192.168.100.34
UPF 1	Open5GS v2.5.8 5GC U-Plane3	192.168.100.31
UPF 2	Open5GS v2.5.8 5GC U-Plane4	192.168.100.32
Kamailio	Kamailio v5.5.1 for VoIP	192.168.100.24
JellyFin	JellyFin v23.0.0 for Video Streaming	192.168.100.33:8096

AMF and NSSF configurations are shown in below table:

Table 2: AMF and SMF configuration

Network Function	IP Address	IP Address on SBI	Supported S-NSSAI
AMF	192.168.100.4	127.0.0.5	SST: 1 SD: 000001, SST: 1 SD: 000002
SMF 1	192.168.100.5	127.0.0.4	SST: 1 SD: 000001, SST: 1 SD: 000002
SMF 2	192.168.100.6	127.0.0.24	SST: 1 SD: 000001, SST: 1 SD: 000002
AMF	192.168.100.21	127.0.0.5	SST: 1 SD: 000001, SST: 1 SD: 000002, SST: 2 SD: 000003, SST: 2 SD: 000004

SMF 3	192.168.100.22	127.0.0.4	SST: 1 SD: 000001, SST: 2 SD: 000003
SMF 4	192.168.100.23	127.0.0.24	SST: 1 SD: 000002, SST: 2 SD: 000004

Each Data Networks is shown in below table:

Table 3: Data networks

Data Network	S-NSSAI	TUNnel interface of DN	DNN	TUNnel interface of UE	U-Plane #
10.45.0.0/16	SST:1 SD:0x000001	ogstun/10.45.0.1	internet	ogstun	U-Plane 1
10.46.0.0/16	SST:1 SD:0x000002	ogstun/10.46.0.1	internet	ogstun	U-Plane 2
10.47.0.0/16	SST:1 SD:0x000001	ogstun/10.47.0.1	voip	uesimtun0	U-Plane 3
10.48.0.0/16	SST:1 SD:0x000002	ogstun2/10.48.0.1	voip	uesimtun0	U-Plane 4
10.49.0.0/16	SST:2 SD:0x000003	ogstun/10.49.0.1	internet	uesimtun0	U-Plane 3
10.50.0.0/16	SST:2 SD:0x000004	ogstun2/10.50.0.1	internet	uesimtun0	U-Plane 4

Subscriber Information (other information is default) is also shown in below table:

Table 4: Subscriber Information

UE Name	IMSI	DNN	Supported S-NSSAI	5QI Values (as per TS 23.501)
ue	001010000000000	internet	SST: 1 SD: 0x000001	9
ue	001010000000001	internet	SST: 1 SD: 0x000002	9
ue	001010000000000	voip	SST: 1 SD: 0x000001	9
ue	001010000000001	Voip	SST: 1 SD: 0x000002	9
ue 1	001010000000003	internet	SST: 2 SD: 0x000003	9
ue 1	001010000000004	internet 2	SST: 2 SD: 0x000004	9

IMPLEMENTATION

This section will give the brief description of all the required steps, configuration, and setups to achieve the network slicing, VoIP, File Sharing and Video Streaming services through UERANSIM.

INSTALLING DIFFERENT SOFTWARES ON VIRTUAL MACHINES

Five plus Five in total 10 Virtual Machines are set up on ubuntu 22.04 VM for installing different software's like Open5gs and UERANSIM:

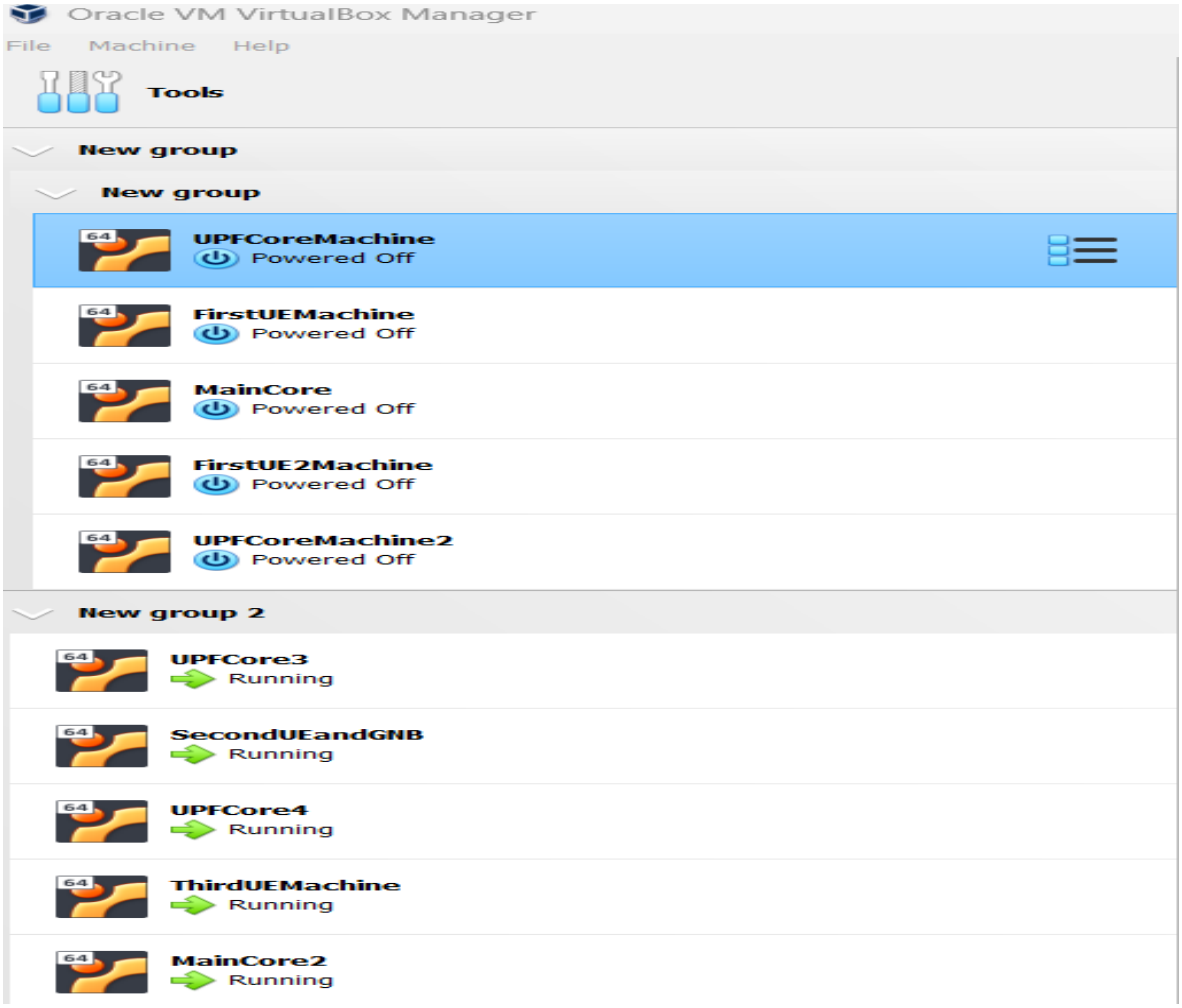

Figure 3: Virtual Machines used for this project.

CONFIGURATION FILES SETUP AND STATUS

The AMF and NSSSF configuration files are edited and two SMF configuration file are created for each architecture. SST and SD combination are given as per our project requirement.

Below is the screenshot of one of the SMF file where the changes have been made in the IP address, SST and SD values.

```
GNU nano 6.2                    /etc/open5gs/smf1.yaml
    - address: 192.168.100.22
  metrics:
    server:
      - address: 127.0.0.4
        port: 9090
  session:
    - subnet: 10.47.0.1/16
      dnn: voip
    - subnet: 10.49.0.1/16
      dnn: internet
  dns:
    - 8.8.8.8
    - 8.8.4.4
  mtu: 1400
#  p-cscf:
#    - 127.0.0.1
#    - ::1
#  ctf:
#    enabled: auto    # auto(default)|yes|no
  freeDiameter: /etc/freeDiameter/smf1.conf
  info:
    - s_nssai:
        - sst: 1
          sd: 000001
          dnn:
            - voip
        - sst: 2
          sd: 000003
          dnn:
            - internet
      tai:
        - plmn_id:
            mcc: 001
            mnc: 01
          tac: 1
  pfcp:
    client:
      upf:
        - address: 192.168.100.31
          dnn: [voip, internet]
```

Similarly, other SMF files are configured.

After the installation of Open5gs is completed, we checked the status of AMF and SMF file which should be in the running stage to run the further configurations correctly.

Below command is used to check the status of the AMF and SMF files.

sudo service open5gs-amfd status

sudo service open5gs-smfd status

REGISTER UE DEVICE

Open5GS provides a WebUI application. We can register the UE devices via connecting to the WebUI through a browser within container session so, we used a browser ‘firefox’ in the VM ‘MainCore’ and ‘MainCore2’.

The web interface will start on the firefox window with the below url: http://localhost:9999

After connecting to WebUI we added new subscribers for different slices as shown in the below screenshots.

Below figure shows the configuration of one UE. In a similar pattern, all other UEs for each SST/SD combinations are configured.

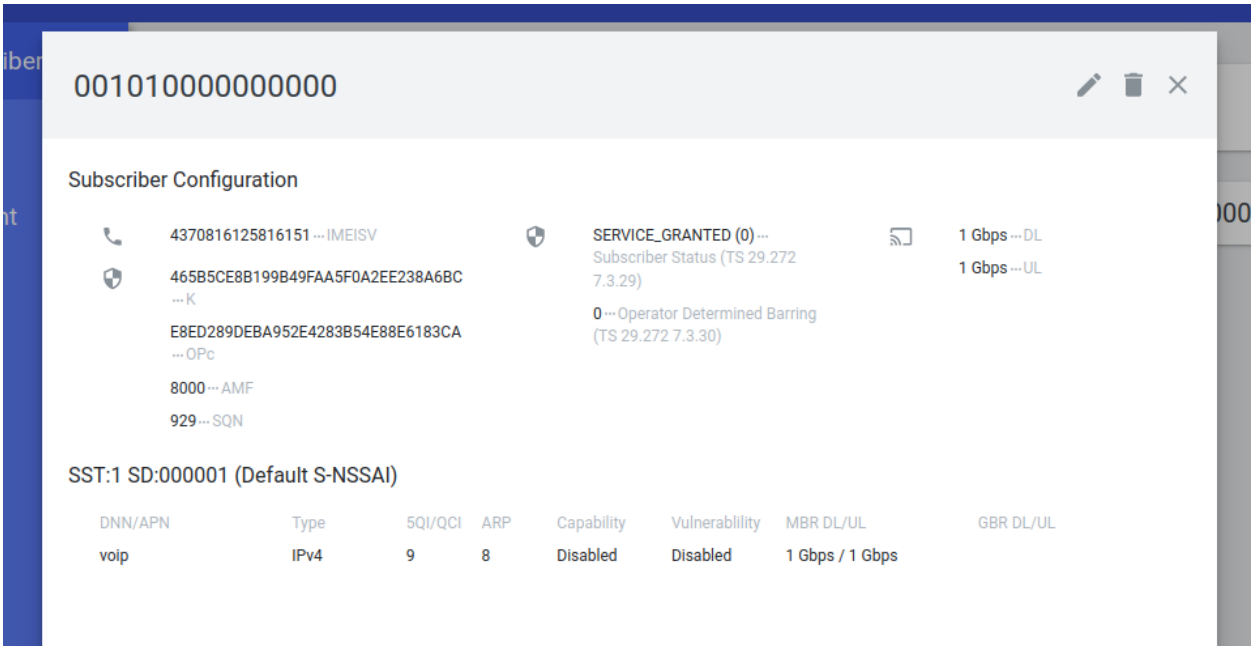

Figure 4: WEBUI with one subscriber detail

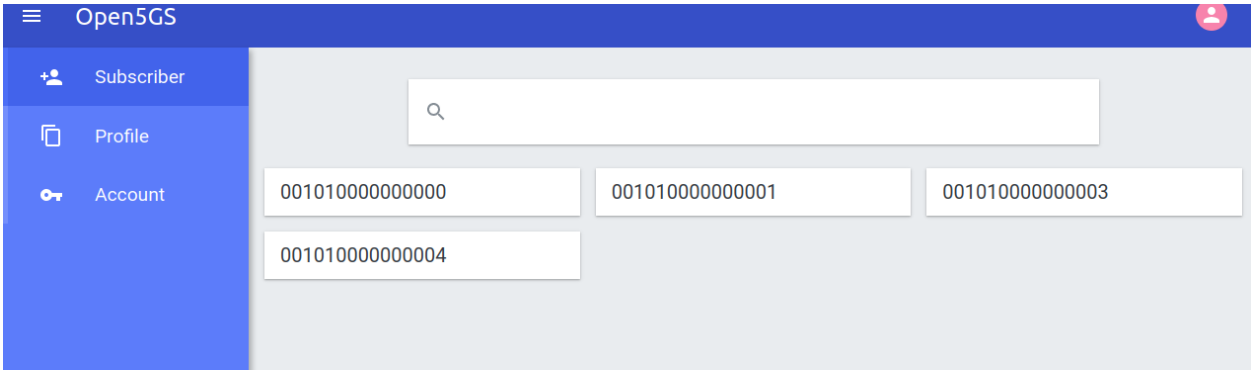

Figure 5: WEBUI with four subscriber (UE)

CREATION OF USER PLANES FOR SST1 and SST2

In our 5G architecture we are separating the control plane (Open5gs) and user plane (UPFs, UEs and RANs). So, for implementing this approach we created two separate UPFs in two different Virtual Machines (namely UPFCoreMachine & UPFCoreMachine2 for File Sharing and UPFCore3 & UPFCore4 for Video Streaming and SIP Calling) for each of the SST 1 SDs and SST 2 SDs.

The upf.yaml configuration files are edited with the change in the values of pfcp and gtpu bind address to match it with the IP addresses of the esp0 interface of the VMs.

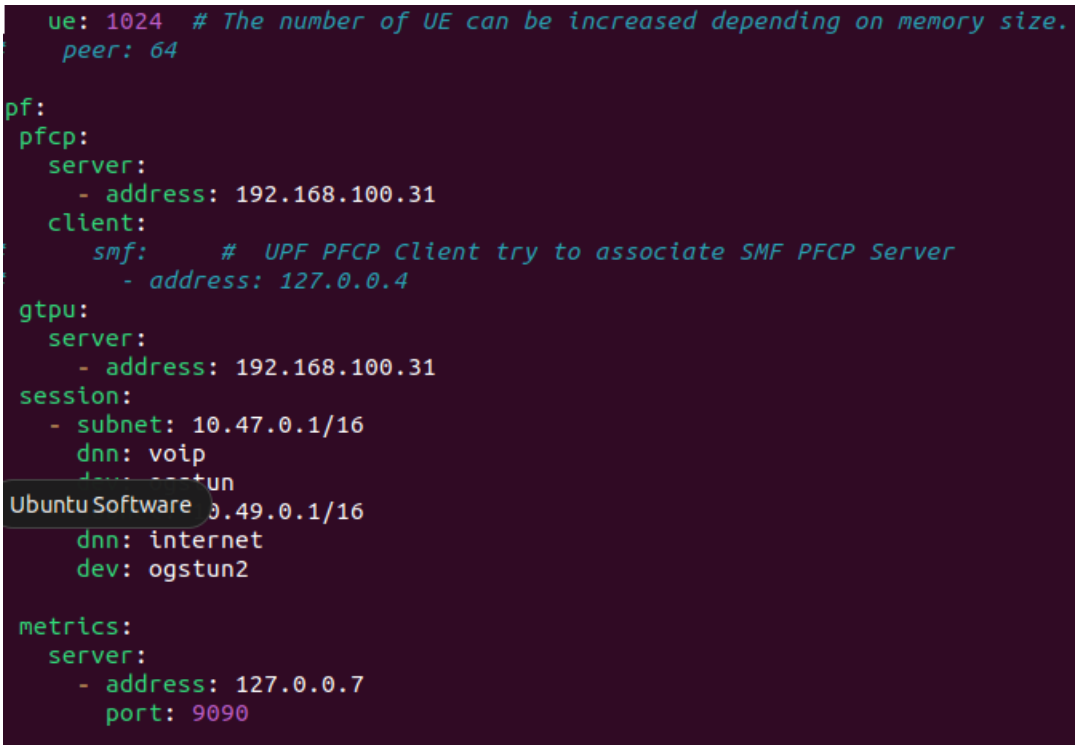

Figure 6: Modified UPF file (upf.yaml file)

Above is the screenshot of one of the modified UPF file. Similarly, another UPF file is also modified to execute the process. Thereafter we enabled IP Forwarding and NAT Port Translation for tunnel interface 'ogstun' and 'ogstun' with IP address 10.45.0.1/16 and 10.46.0.1/16 and for tunnel interface 'ogstun' and 'ogstun2' with IP address 10.47.0.1/16 & 10.48.0.1/16 and 10.49.0.1/16 10.50.0.1/16 respectively.

CONFIGURATION OF UERANSIM (gNodeB and UEs)

Virtual Machines namely 'FirstUEMachine'/'SecondUEMachine' is created for installing the UERANSIM and then configuring gNodeB.

UERANSIM contains two parts gNodeB and UE. The gNodeB config files which are related to Open5GS are in UERANSIM/config/open5gs-gnb.yaml. We configured the linkIp, ngapIp, gtpIp and amfConfigs: address in the config file. linkIp, ngapIp, gtpIp are setup with (IP address of UERANSIM running VM's own IP). The amfConfigs: address (IP address of the Open5GC running VM's first IP among the 3 IPs). The gNodeB can be started with the UERANSIM/build/nr-gnb script by using the UERANSIM/config/open5gs-gnb.yaml config file.

Figure 7: gnodeB configuration of one of the UERANSIM VM's

In the next process 4 UEs are created and connected to a single gNodeB. Each UE is configured to be used with a single slice making a total of 4 slices (i.e., for each SST-SD combination) as described below:

UE 1 (IMSI-001010000000000) SST-1, SD-1 for internet connected via upf1 for NextCloud.

UE 2 (IMSI-001010000000001) SST-1, SD-2 for internet connected via upf2 for NextCloud.

UE 3 (IMSI-001010000000000) SST-1, SD-1 for voip connected via upf1 for Kamailio.

UE 4 (IMSI-001010000000001) SST-1, SD-2 for voip2 connected via upf2 for Kamailio.

UE 5 (IMSI-001010000000003) SST-2, SD-3 for internet connected via upf1 for JellyFin.

UE 6 (IMSI-001010000000004) SST-2, SD-4 for internet2 connected via upf2 for JellyFin.

```
# IMSI number of the UE. IMSI = [MCC|MNC|MSISDN] (In total 15 digits)
supi: 'imsi-001010000000001'
# Mobile Country Code value of HPLMN
mcc: '001'
# Mobile Network Code value of HPLMN (2 or 3 digits)
mnc: '01'
# SUCI Protection Scheme : 0 for Null-scheme, 1 for Profile A and 2 for Profile>
protectionScheme: 0
# Home Network Public Key for protecting with SUCI Profile A
homeNetworkPublicKey: '5a8d38864820197c3394b92613b20b91633cbd897119273bf8e4a6f4>
# Home Network Public Key ID for protecting with SUCI Profile A
homeNetworkPublicKeyId: 1
# Routing Indicator
routingIndicator: '0000'

# Permanent subscription key
key: '465B5CE8B199B49FAA5F0A2EE238A6BC'
# Operator code (OP or OPC) of the UE
op: 'E8ED289DEBA952E4283B54E88E6183CA'
# This value specifies the OP type and it can be either 'OP' or 'OPC'
opType: 'OPC'
# Authentication Management Field (AMF) value
amf: '8000'
# IMEI number of the device. It is used if no SUPI is provided
imei: '356938035643803'
# IMEISV number of the device. It is used if no SUPI and IMEI is provided
imeiSv: '4370816125816151'

# List of gNB IP addresses for Radio Link Simulation
gnbSearchList:
  - 192.168.100.34

# UAC Access Identities Configuration
```

Figure 8: UE configuration for mnc and mcc in imsi

In the UE configuration we changed the mcc and mnc values to 001 and 01 respectively. So the Supi or the IMSI number must be changed accordingly where the first 3 values represents mcc and later 2 values the mnc.

In the following picture slice information given in the UE configuration is mentioned. It has to be under the session tag followed by a configured and a default slice configuration. In practical life these configured and default sessions can be different to make a back up connection for each UE but in our configuration all three holds the same SST and SD values. Currently for one set up two types of SST is not getting supported in Open5GS and giving error. So current version of Open5GS only different SD values can make different slices. It's been informed in the github issues of Open5GS and their team is working on it.

```
sessions:
  - type: 'IPv4'
    apn: 'voip'
    slice:
      sst: 1
      sd: 0x000002

# Configured NSSAI for this UE by HPLMN
configured-nssai:
  - sst: 1
    sd: 0x000002

# Default Configured NSSAI for this UE
default-nssai:
  - sst: 1
    sd: 0x000002

# Supported integrity algorithms by this UE
integrity:
  IA1: true
  IA2: true
  IA3: true

# Supported encryption algorithms by this UE
Terminal
  EA2: true
  EA3: true

# Integrity protection maximum data rate for user plane
integrityMaxRate:
  uplink: 'full'
  downlink: 'full'
```

Figure 9: UE configuration with correct slice information.

All the four UE config files are then modified according to the details of the subscriber registered in Open5gs WebUI, gNodeB address and slicing information.

Next step in setting up the whole process is to run below steups:

• Open5GS 5GC C-Plane

• Open5GS 5GC U-Plane1(UPF1) & U-Plane2(UPF2)

• UERANSIM (gNodeB)

• UERANSIM (UE set to SST:1 and SD:0x000001, DNN/APN: internet)

• UERANSIM (UE set to SST:2 and SD:0x000003, DNN/APN: voip)

KAMAILIO SERVER CONFIGURATION

To use VoIP service, a call server is set up with the help of "Kamailio" package. A "Call Server" is created using MariaDB database Server for authentication. This provision is available in Kamailio. The protocol used for VoIP (Voice over IP) is SIP (Session Initiation Protocol. The user registration is possible through 'Linphone'. Since each host (MainCore2-192.168.100.24) with slice SST1, SD1 and SST1, SD2 with VoIP is installed with a Lin0phone package, the users created in database are used to be registered in each host to place a call after PDU Session establishment using the uesimtun0 interface using the nr-binder tool of UERANSIM.

Linux Command:

kamctl start #To start Kamailio server.

sudo systemctl restart kamailio #To restart Kamailio server.

kamctl db show subscriber #It shows all the registered subscribers in the database of MySQL.

Two UEs added at VMs 'ue' and 'ue1' for SIP domain 192.168.100.24:

```
root@teamslicexcore2-VirtualBox:/home/teamslicexcore2# kamctl db show subscriber
mysql: [Warning] Using a password on the command line interface can be insecure.
+----+-------------+---------------+-----------+----------------------------------+----------------------------------+
| id | username    | domain        | password  | ha1                              | ha1b                             |
+----+-------------+---------------+-----------+----------------------------------+----------------------------------+
|  1 | ue_sst1_sd1 | 192.168.100.24 | 73@mslice | 0320368925f3ca9e9f852e3444141101 | 216a9ac23d17510fe319bf41df2d58f8 |
|  2 | ue_sst1_sd2 | 192.168.100.24 | 73@mslice | 024210ec05e416aa079387c6c59c93f4 | ca5ba68b953ee70164e5cb0501b800ea |
+----+-------------+---------------+-----------+----------------------------------+----------------------------------+
root@teamslicexcore2-VirtualBox:/home/teamslicexcore2#
```

Figure 10: Two UEs added for SIP domain 192.168.100.24

Each time system starts. Kamailio must be started manually.

The softphone 'Linphone' was installed in the User Equipments by downloading the software package from its website and then extracting the zipped package to run it using the commands:

apt-get install -y linphone

sh nr-binder 10.48.0.2 linphone

FILE SHARING PLATFORM NEXTCLOUD

Next cloud is installed on the MainCore VM Machine system to enable file sharing between users. The Firefox browser in the VM is also installed by default to configure the admin and user profile in next cloud server. The Next Cloud server admin can be accessed from the url: http://192.168.100.4/nextcloud/ where the IP address 192.168.100.4 is the IP address of the VM of MainCore in which next cloud server is installed. As MainCore Machine hase 3 IP Addresses,

nextcloud can also be accessed by 192.168.100.5 and 192.168.100.6. After registering as an admin in Next Cloud, one user profile were created to be used by the two User Equipments for accessing the file sharing services having slice SST1, SD1 and SST1, SD2.

To install Next Cloud:

sudo wget https://download.nextcloud.com/server/releases/nextcloud-24.0.1.zip

sudo unzip nextcloud-24.0.1.zip

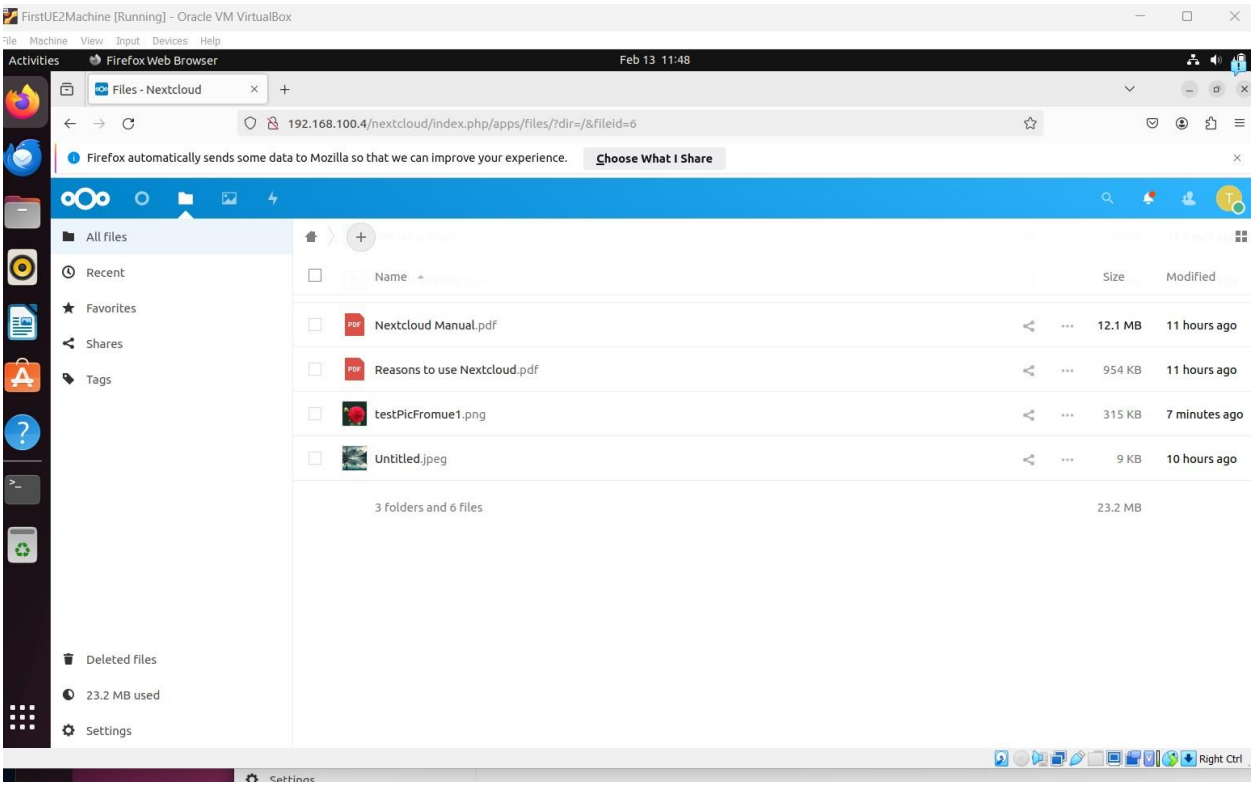

Figure 11: Next Cloud with UEs registered.

VIDEO STREAMING VIA JellyFin

To establish the video streaming service, JellyFin is installed in the UE1(192.168.100.33) VM. Next, Firefox browser is also installed in the VM by default to configure the admin in JellyFin server. To verify the installation by visiting the default page on our server IP at port 8096: http://192.168.100.33:8096. To configure JellyFin, we logged in to the /admin backend dashboard via port 8096. http://192.168.100.33:8096/admin.

Connect Firefox on User Equipment to JellyFin Server as JellyFin has integrated video streaming and player through web browser. For achieving this we have connected ‘ue1’ where the UE for slice with SST 2, SD 4 (192.168.100.34) is available which will be broadcasting a video on a TUN interface after PDU session established via 5G Core ogstun2 from the JellyFin server. UE

from 192.168.100.33 can start streaming using the ogstun interface using the nr-binder tool of UERANSIM as below:

sh nr-binder <IP address of PDU Session> firefox

Linux Command:

curl https://repo.jellyfin.org/install-debuntu.sh | sudo bash

ps -ef | grep -i jellyfin

EXECUTING THE 5G NETWORK

To achieve the network slicing, which is discussed further below in the next section, we ran all the software components involved in the 5G architecture as discussed below.

a) Run Open5GS 5GC

We first we stop the 5G core components and then restart.

Linux command to run the Open5GS 5GC C-Plane:

systemctl stop open5gs-smfd &

systemctl stop open5gs-nrfd &

systemctl stop open5gs-scpd &

systemctl stop open5gs-amfd &

systemctl stop open5gs-ausfd &

systemctl stop open5gs-udmd &

systemctl stop open5gs-udrd &

systemctl stop open5gs-pcfd &

systemctl stop open5gs-nssfd &

systemctl stop open5gs-bsfd &

systemctl stop open5gs-webui

Linux command to run the Open5GS 5GC C-Plane:

systemctl start open5gs-nssfd &

systemctl start open5gs-nrfd &

systemctl start open5gs-scpd &

```
systemctl start open5gs-amfd &

systemctl start open5gs-ausfd &

systemctl start open5gs-udmd &

systemctl start open5gs-udrd &

systemctl start open5gs-pcfd &

systemctl start open5gs-bsfd &

systemctl start open5gs-webui
```

After running the above commands, we checked the logs of relevant network functions like amf log, smf log, nssf log, etc. Next is to restart the newly created four SMF config files for each SST/SD combination and stopping the default file smf.yaml and checked their status.

```
/bin/open5gs-smfd -c /etc/open5gs/smf1.yaml

/bin/open5gs-smfd -c /etc/open5gs/smf2.yaml
```

Similar command is used for each respective SST and SD combinations.

b) Run Open5GS 5GC U-Plane1(UPF1) & U-Plane2(UPF2)

Next, we executed the Open5GS 5GC U-Plane1(UPF1) & U-Plane2(UPF2).

Linux command to execute:

```
ip tuntap add name ogstun mode tun

ip addr add 10.45.0.1/16 dev ogstun

ip link set ogstun up | iptables -t nat -A POSTROUTING -s 10.45.0.0/16 ! -o ogstun -j MASQUERADE
```

The above commands will be executed in both VMs according to ip address (upf1 and upf2).

c) Run UERANSIM (gNodeB)

We executed the UERANSIM (gNodeB) as a next step. We first, did an NG Setup between gNodeB and 5GC, then register the UEs with 5GC and establish a PDU session. The RAN (Ueransim gNodeB) was connected with the following set of commands:

```
cd Downloads/UERANSIM/build

./nr-gnb -c ../config/open5gs-gnb.yaml

./nr-ue -c ../config/open5gs-ue.yaml
```

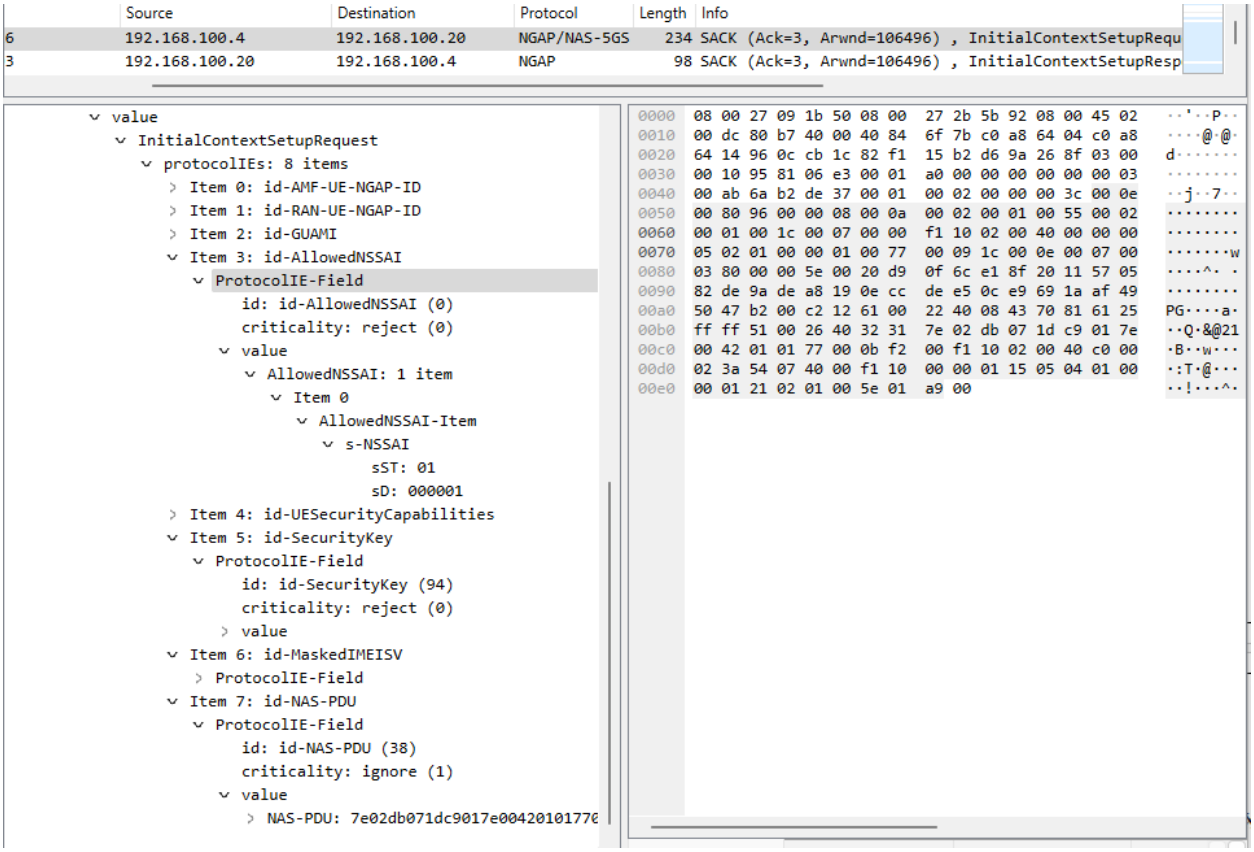

Figure 12: Interface capture of NG Setup Request and Response between gNodeB and AMF

d) Attaching the UEs

After successful NG setup, we ran the four UERANSIM (UEs) configured in the four VMs for each of the SST/SD combinations.

For the UE in VM - ue1 with slice parameters SST:1 and SD:0x000001, DNN/APN: internet

We ran the below command to register the UE with 5GC and establish a PDU session between the UE and the UPF-1 (U Plane-1). We used the below command:

```
sudo ./build/nr-ue -c open5gs-ue.yaml
```

After PDU session was successfully established then the IP address 10.45.0.5 was assigned to the TUN interface ‘uesimtun0’ interface of the UE. This can also be verified from the UPF (U- Plane 1) logs as shown in the figure.

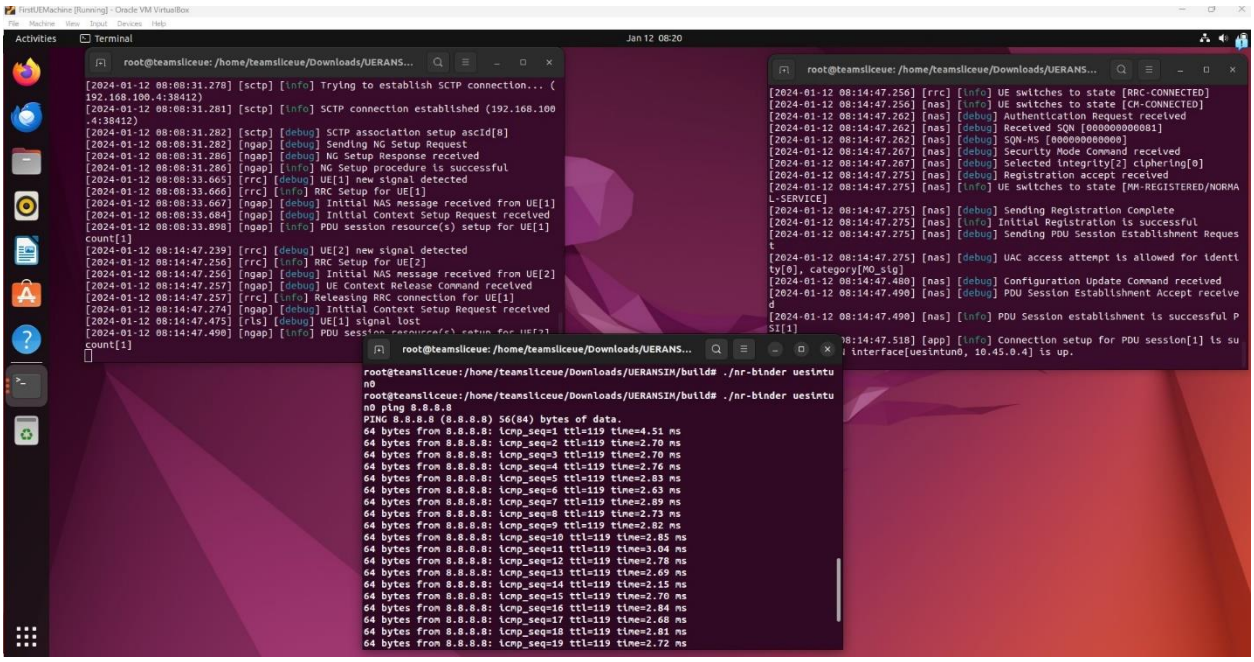

Figure 13: PDU session successfully established for UE in VM 'ue1' with SST-1 and SD-1

PDU Session Request and Response for UE_SST1_SD1 captured between AMF and gNB.

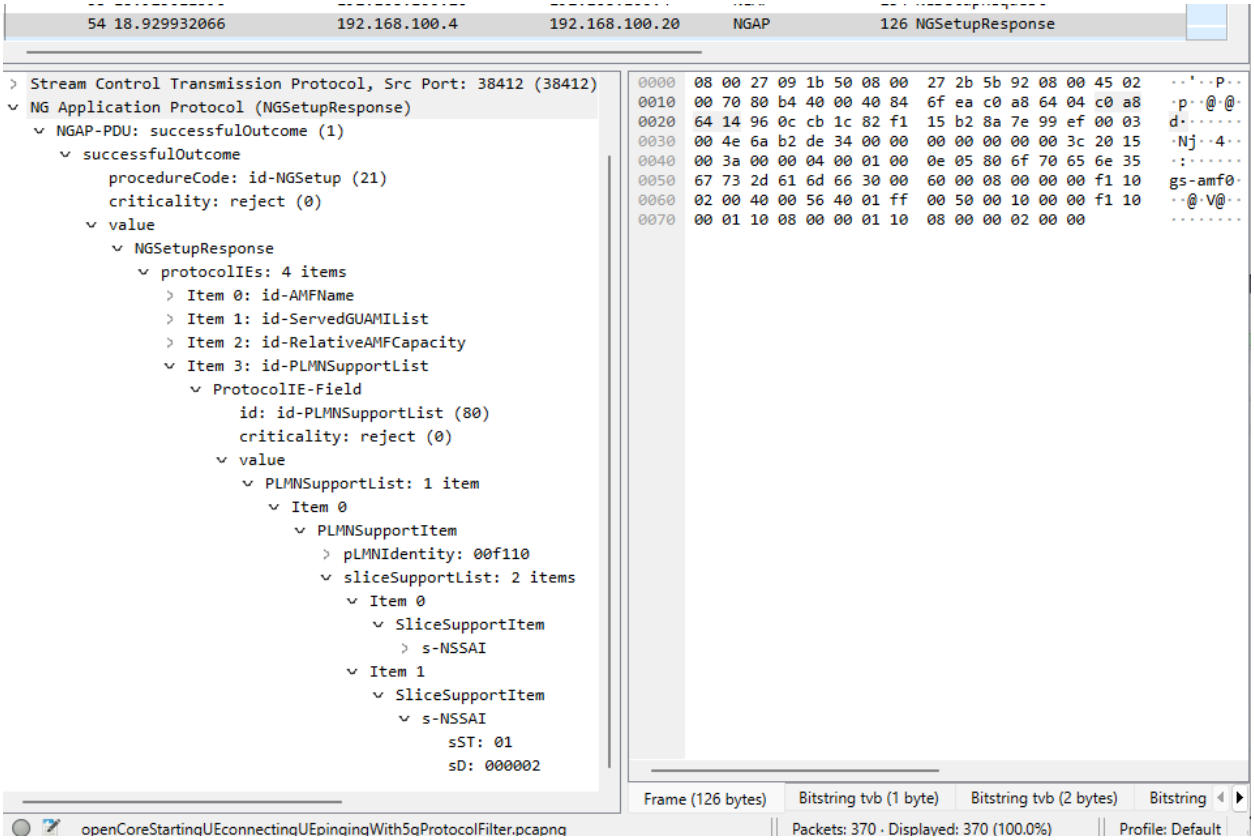

Figure 14: PDU Session Response for UE SST1 SD1 and SD2 captured between AMF and gNB

Similarly, the other three UEs were also successfully attached to their respective U-Planes based on their slicing configurations mentioned in previous sections.

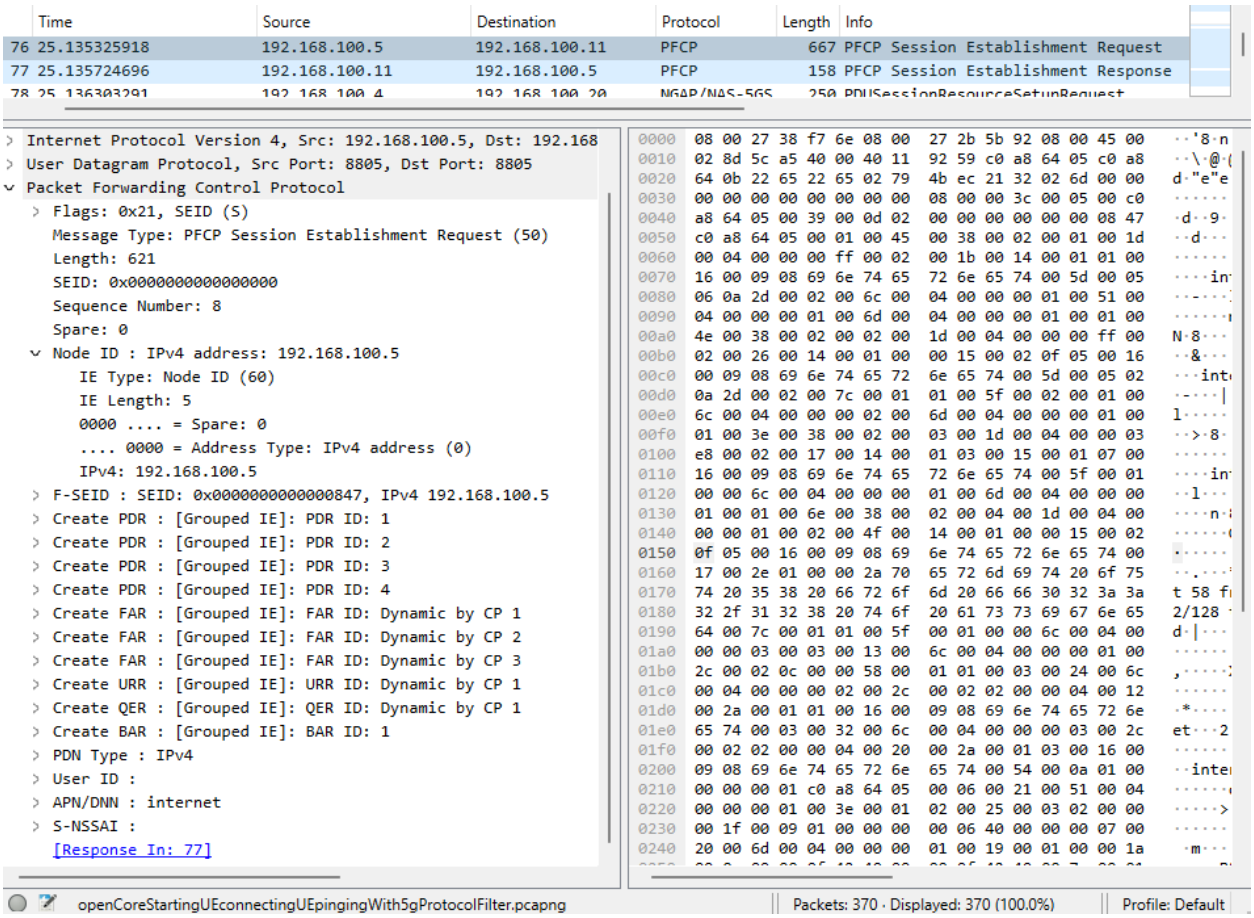

Figure 15: PFCP Session Establishment Request captured between SMF and UPF-1

Accessing the Services using Network Slices

After successfully establishing the PDU sessions for all four UEs, the final configuration of the complete UE network can be observed from the below table:

VM and UE Name	S-NSSAI	Tun Interface IP	DNN	Services /Software and Server IP	U-Plane
ue1 open5gs-ue.yaml	SST:1 SD:0x000001	10.45.0.2	internet	File Sharing – NextCloud (192.168.100.4)	U-Plane 1
ue2 open5gs-ue.yaml	SST:1 SD:0x000002	10.46.0.2			U-Plane 2
ue3 open5gs-ue.yaml	SST:1 SD:0x000001	10.47.0.2	voip	SIP Voip Calling with Linphone –	U-Plane 1

ue4 open5gs-ue.yaml	SST:1 SD:0x000002	10.48.0.2	voip	Kamailio (192.168.100.24)	U-Plane 2
ue5 open5gs-ue.yaml	SST:2 SD:0x000003	10.49.0.2	internet	Streaming - JellyFin (192.168.100.33)	U-Plane 1
ue6 open5gs-ue.yaml	SST:2 SD:0x000004	10.50.0.2	internet		U-Plane 2

NETWORK SLICING

Here we will discuss the three platforms for file sharing, video streaming and SIP call between the users using different slicing to achieve these services.

File sharing using NextCloud between ue1(sst1, sd1) and ue2(sst1, sd2)

• For accessing the file sharing service, the UEs in VMs 'ue1' and 'ue2' will be accessing the file sharing server NextCloud available in VM 'MainCore' with IP address 192.168.100.4.

• These two UEs are already configured with the necessary slice to support the file sharing service functionality as already described in previous sections.

Below series of steps are involved in accessing the file sharing service:

1. NG Connection establishment between gNB and 5GC-Control Plane.

2. PDU session establishment between U-Plane 1 and ue1.

3. PDU session establishment between U-Plane 1 and ue2.

4. The IP address for the 'uesimtun0' interfaces of both the UEs are shown in below table:

VM and UE File Name	IP of interface uesimtun0	Slice used
ue1 open5gs-ue.yaml	10.45.0.6	SST1-SD1
ue2 open5gs-ue.yaml	10.46.0.5	SST1-SD2

5. Once the UEs are assigned the IP addresses on the 'ogstun' interface then 'nr-binder' tool available as a part of UERANSIM is used to access the NextCloud server via web browser already installed in both UE1 and UE2 using the below command:

Linux command for ue1 and ue2:

```
cd Downloads/UERANSIM/build

sh nr-binder 10.45.0.2 firefox (for ue1)

sh nr-binder 10.46.0.2 firefox (for ue2)
```

6. In the browser windows the NextCloud server can be accessed from the url https://192.168.100.4/nextcloud/ by logging in to users ‘UE_SST1_SD1’ and ‘UE_SST1_SD2’ already registered in the NextCloud server in previous sections in VMs ‘ue1’ and ‘ue2’ respectively.

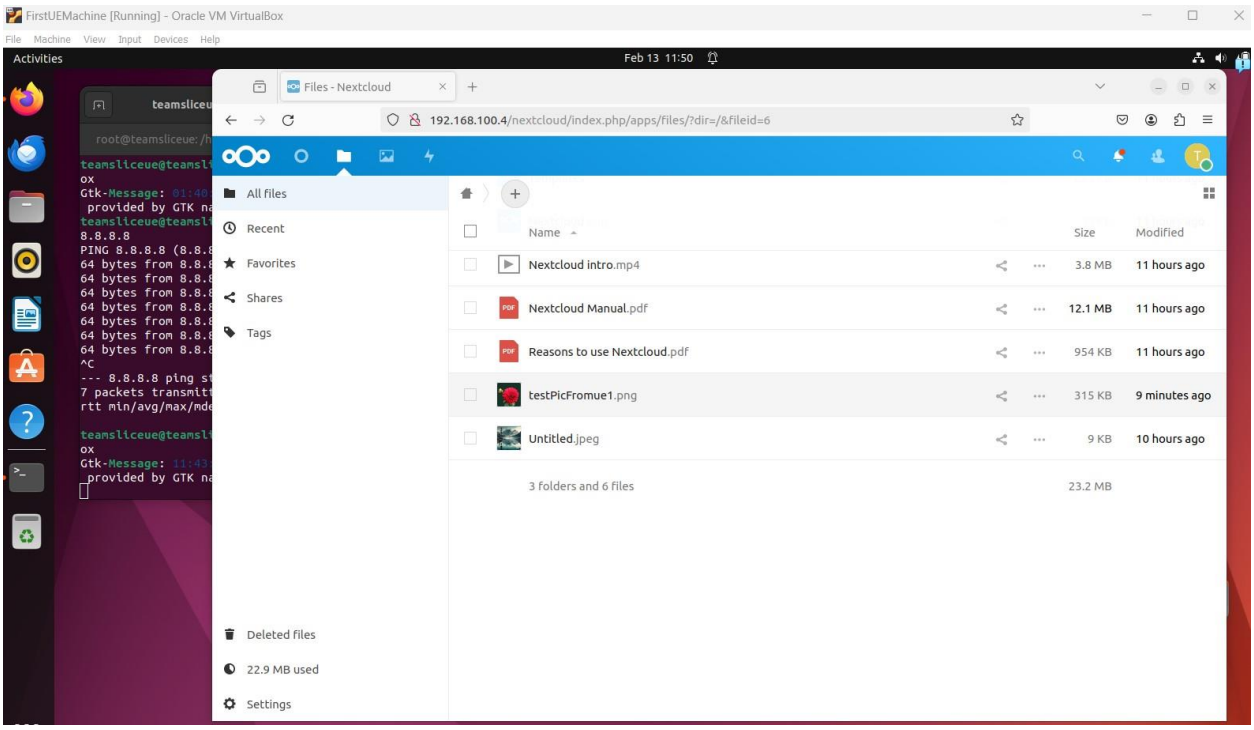

Figure 16: Shared file accessed by the user ‘UE SST1 SD1’ in its dashboard.

Video Live Streaming using JellyFin and Firefox between Ue5(SST2, SD3) and Ue6(SST2, SD4)

1) For accessing the live streaming service, the UEs in VM ‘ue5’ and ‘ue6’ will be accessing the live streaming server JellyFin available in VM ‘SecondUEandGNB’ with IP address 192.168.100.33.

2) These two UEs are already configured with the necessary slice to support the streaming service functionality as already described in previous sections.

3) We used the UE ‘open5gs-ue1.yaml’ which is configured at VM ‘SecondUEandGNB’ to broadcast a video on a TUN interface after PDU session established via 5G Core using Firefox Browser already installed in the VM and the UE ‘open5gs-ue1.yaml’ which is configured in VM ’ThirdUEMachine’ to watch the live stream video using the browser interface of JellyFin.

4) Below series of steps are involved in accessing the streaming service:

a) NG Connection establishment between gNB and 5GC-Control Plane.

b) PDU session establishment between U-Plane 1 and ue5.

c) PDU session establishment between U-Plane 1 and ue6.

d) The IP address for the ‘uesimtun0’ interfaces of both the UEs are shown in below table:

VM and UE Name	IP of interface uesimtun0	Slice used
ue5 open5gs-ue1.yaml	10.49.0.6	SST2-SD3
ue6 open5gs-ue1.yaml	10.50.0.5	SST2-SD4

e) Once the UEs are assigned the IP addresses on the ‘ogstun2’ interface then ‘nr-binder’ tool available as a part of UERANSIM is used to access the JellyFin server via Firefox in web browser in ‘ue2’ using the below commands:

Linux command for ue1 and ue2:

cd UERANSIM/build

sh nr-binder 10.45.0.6 firefox (for ue5)

sh nr-binder 10.46.0.5 firefox (for ue6)

After running the above command in ue1 and an already downloaded video was selected for live streaming on ue6.

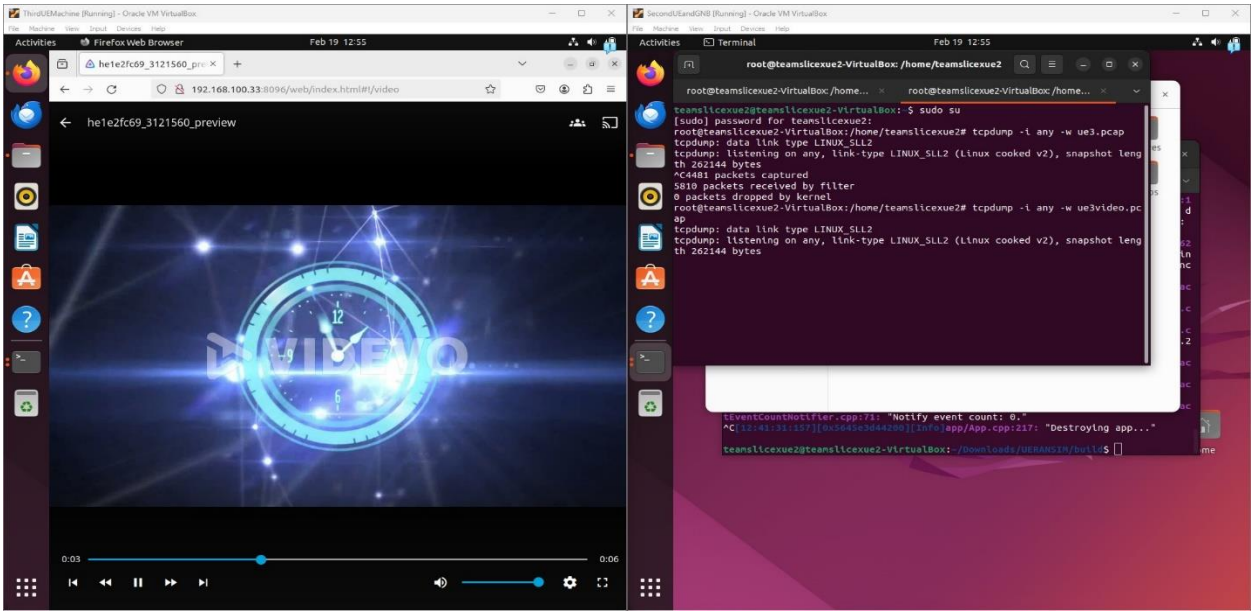

Figure 17: UE with SST-2, SD-3 broadcasting a video from firefox JellyFin dashboard.

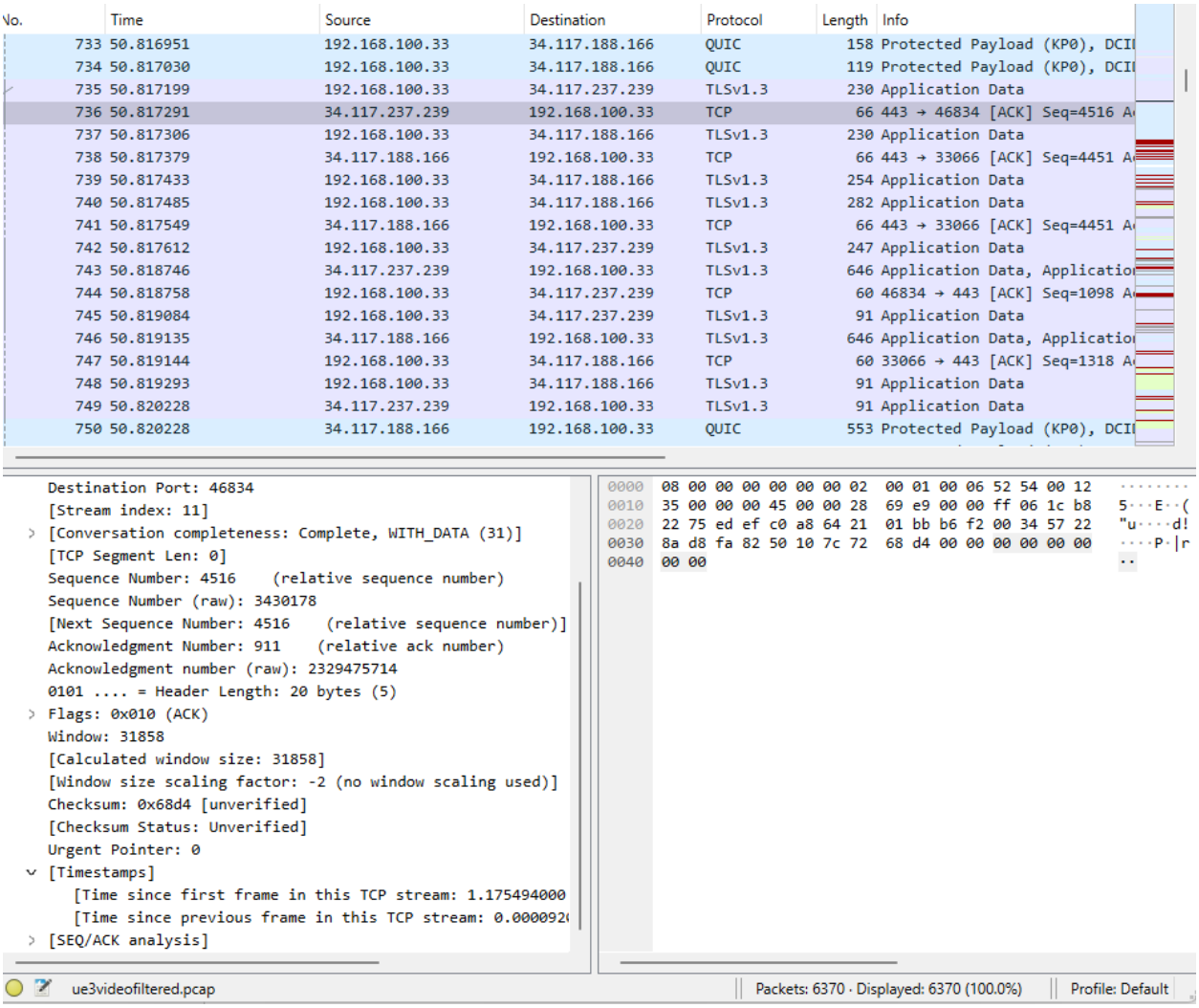

Figure 18: Video streaming service interface capture (esp0) of VM 'SecondUEandGNB' between ue5 and ue6.

VOIP SIP Calling using Kamailio Server between ue3(SST1, SD1) and ue4(SST1, SD2)

• For VOIP calling the UEs in VM 'ue3' and 'ue4' have used their respective slices dedicated for voip calling using a softphone 'Linphone' to access kamailio server already installed at VM 'MainCore2' with IP address 192.168.100.24 as mentioned below:

ue3 open5gs-ue.yaml	SST:1 SD:0x000001	DNN: voip	SIP Voip Calling – Linphone	U-Plane 1
ue4 open5gs-ue.yaml	SST:1 SD:0x000002	DNN: voip	SIP Voip Calling – Linphone	U-Plane 2

• These two UEs are already configured with the necessary slices to support the VoIP calling service functionality as already described in previous sections.

• Below series of steps are involved in accessing the VoIP calling service:

i. NG Connection establishment between gNodeB and 5GC-Control Plane.

i. PDU session establishment between U-Plane 1 and ue3.

ii. PDU session establishment between U-Plane 2 and ue4.

iii. The IP address for the 'uesimtun0' interfaces of both the UEs are shown in below table:

VM and UE Name	IP of interface 'ogstun'
ue3 open5gs-ue.yaml	10.47.0.2
ue4 open5gs-ue.yaml	10.48.0.2

Once the UEs are assigned the IP addresses on the 'ogstun' interface then 'nr-binder' tool available as a part of UERANSIM was used to access the VoIP calling service via Linphone in both 'ue3' and 'ue4' using the below commands:

For ue3:

cd Downloads/UERANSIM/build

sh nr-binder 10.47.0.2 linphone

For ue4:

cd Downloads/UERANSIM/build

sh nr-binder 10.48.0.2 linphone

In the Linphone application interface, the subscriber account already registered in the Kamailio MYSQL database on both ue3 and ue4 was logged in to make a VoIP call. Ex- Dial 'ue_sst1_sd2' from 'ue3' side to make a call to 'ue4' which is registered in Kamailio by this username.

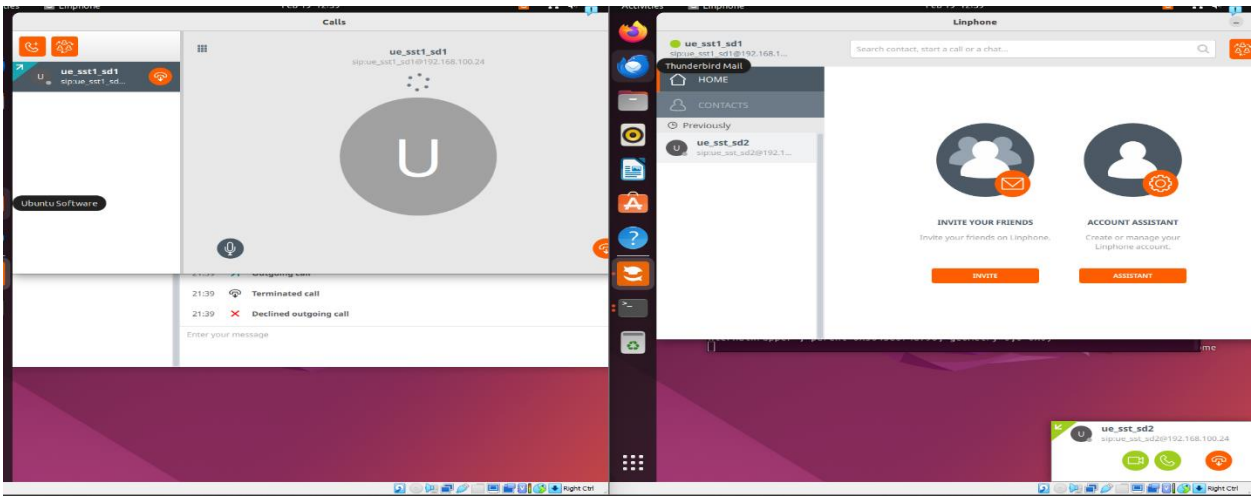

Figure 19: (Left) UE 'ue3' calling 'ue4' using Linphone application accessed on its 'ogstun' interface. (Right) UE 'ue4' is receiving the incoming call from 'ue3' on it 'ogstun' interface.

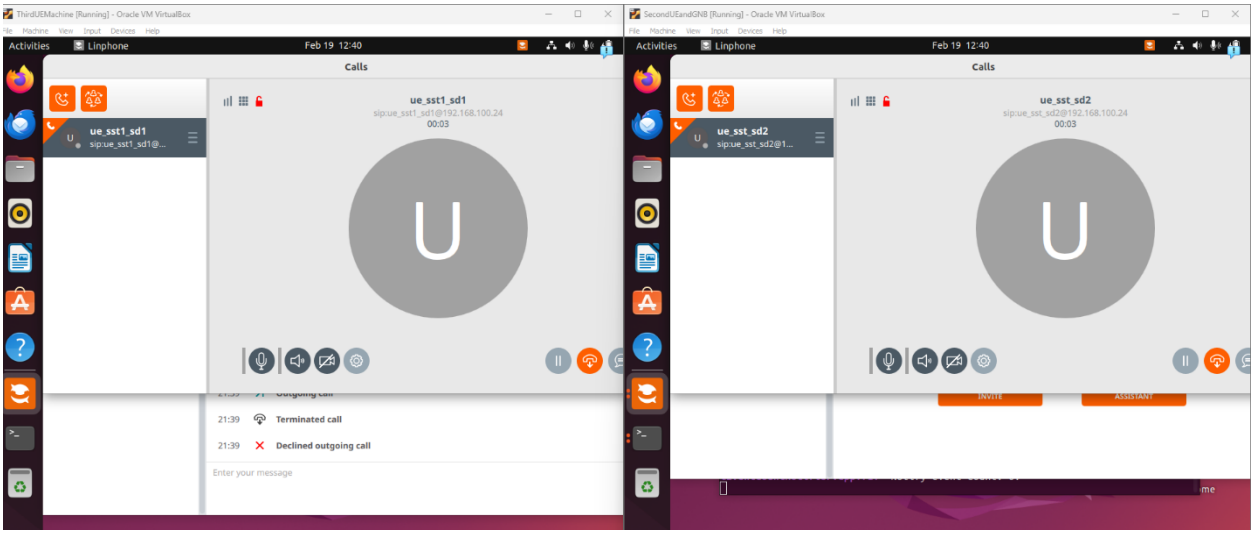

Figure 20: Stablished Call Session between 'ue3' and 'ue4'

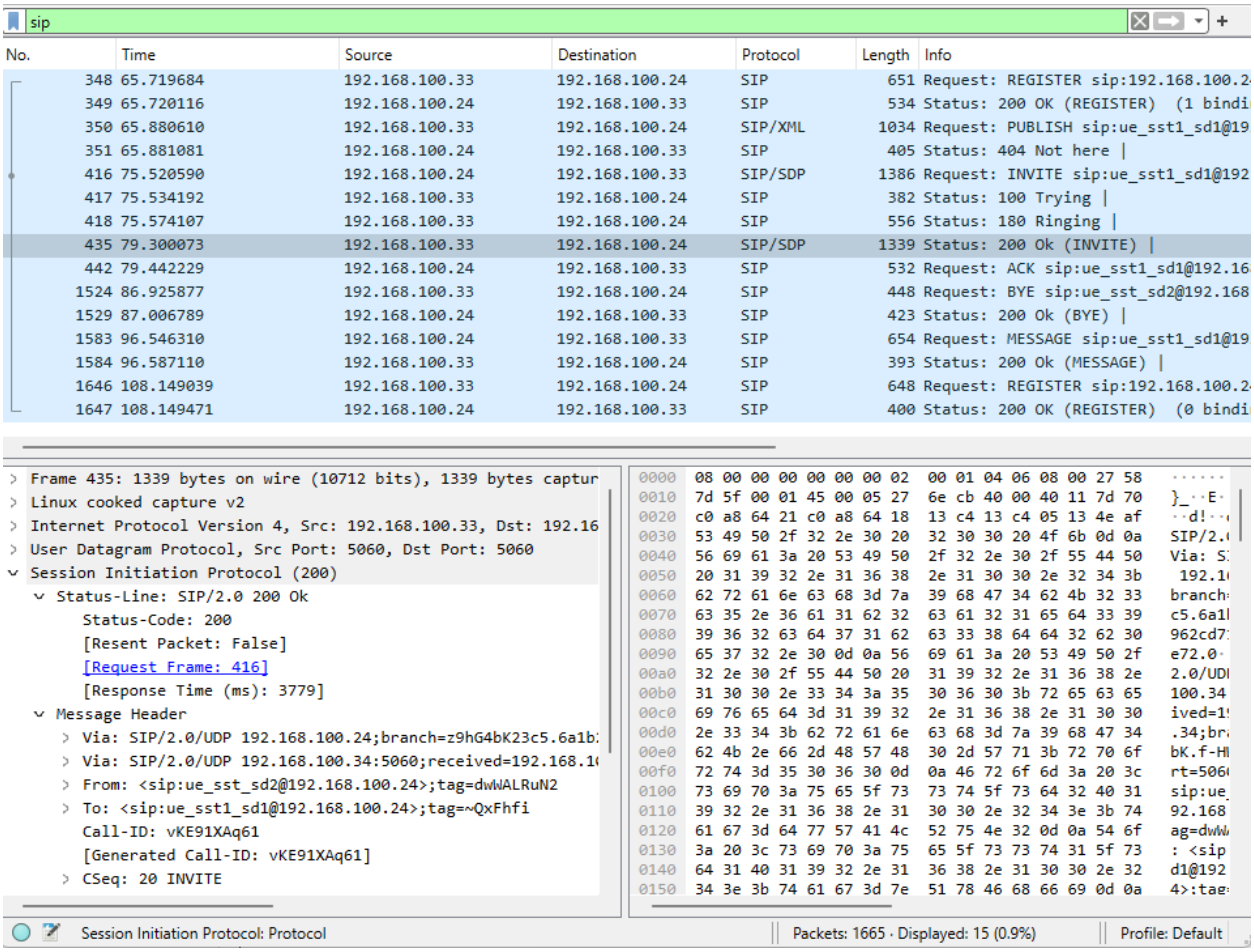

Figure 21: VoIP calling service interface capture (enp0s3) of VM 'MainCore2' between ue3 and ue4.

LIMITATIONS

1. It is found in the implementation process of srsRAN that ZMQ (virtual radio in srsRAN) only support a single eNB and a single UE.

2. To run more than one user equipment, multiple UE/gnb pairs are required to setup in different VMs.

3. In the configuration of UERANSIM and Open5GS more than one SST value was not supported at the beginning of the project. SD values were the only probable separation technique for then. Later Open5GS solved the problem and we configured accordingly. But because of lot of files, in some cases previous pcap files or configuration information may exist in the project.