Departamento de Eletrónica Telecomunicações e Informática

8240 - Mestrado Integrado em Engenharia de Computadores e Telemática

5G System

Core & Access

Authors:
Raquel Pinto 92948
Rodrigo Martins 93264

january, 2022

Contents

1 Acronyms

DNN Data Network Name

AMF Access and Mobility Management Function

SMF Session Management Function

UPF User Plane Function

UE User Equipment

MCC Mobile Country Code

MNC Mobile Network Code

QoS Quality of Service

TAC Tracking Area Code

GNB Next Generation NodeB

VM Virtual Machine

RAN Radio Access Network

AUSF Authentication Server Function

5QI 5G QoS Identifier

NGAP NG Application Protocol

2 Introduction

In this report will be presented our approach to implement a complete 5G System (core and access) using the recommended opensource software (Open5GS (URL1) and UERANSIM (URL2)). To implement this system there was designed an architecture that consists in 3 main components, the core and 2 access components, each represented by a virtual machine running ubuntu 20.04. The core component provides access to 5 Data Network Names (DNNs), each with different ip pools, Quality of Service (QoS) parameters and output interfaces. The access components have different Tracking Area Codes (TACs) and User Equipments (UEs). With this architecture all of the different UEs can access the internet through the 5G Core with different bandwidth and ips as specified in the architecture.

3 Architecture

As previously mentioned, to implement a complete 5G System that includes core and access was used the architecture shown in Figure 1. This architecture is composed of three components, the core and two access areas. Each of these components is a Virtual Machine (VM), the core component only uses the Open5gs software and the access components use the UERANSIM software. For the access areas component it was used 2 VMs, where one of them represents an access area that has 1 Next Generation NodeB (GNB), 4 UEs that access 3 different DNNs named internet, iot and internet2, this access area has a TAC of 1 to differentiate from the other access area that has a TAC of 2, 1 GNB, 3 UEs that access the edge and myslice DNNs. In both access areas the Mobile Country Code (MCC) and Mobile Network Code (MNC) values were left at default values, which are 907 and 70, respectively.

As for the core, 1 VM was used where of all the Open5gs core components only the Access and Mobility Management Function (AMF), Session Management Function (SMF) and User Plane Function (UPF) components were changed. The AMF component was edited to allow all of the TACs (1 and 2) used and to allow the access components to connect to the AMF, by changing the ip address of the NG Application Protocol (NGAP) server. The SMF and UPF components were changed to assign different polls of different ips addresses to each DNN. In Figure 1 these DNNs are also accessed by different interfaces ranging from ogstun to ogstun4. The configuration of the components and some extra configurations needed are available in the Appendix along with some useful links used during the implementation of this project.

The Ip pools and the output interfaces choosen to the different DNNs were:

- internet: 10.45.0.1/16 - ogstun
- edge: 10.46.0.1/16 - ogstun2
- iot: 10.47.0.1/16 - ogstun4
- mySlice: 10.48.0.1/16 - ogstun3
- internet2: 10.49.0.1/16 - ogstun1

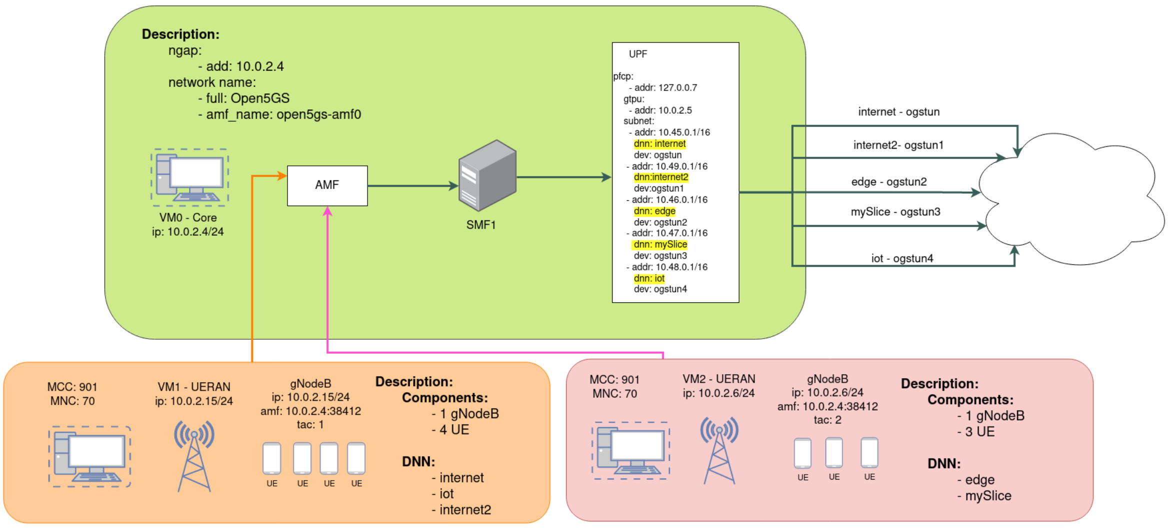


Figure 1: Used architecture.

4 Results

To test the 5G System various tests were performed in the key components of the project. To perform this tests wireshark was extensively used to capture the packet communication between each component and iPerf3 was also used to test the bandwidth of the network.
The obtained results are presented in the following sections.

4.1 gNB Connectivity

To test the connection between GNB and the core, a packet capture was done using wireshark on interface (any) on the core while GNB from tac1 was connecting. The results of this capture can be seen in Figure 2 and 3.

sctp

No.	Time	Source	Destination	Protocol	Length	Info
36	2.435256	10.0.2.15	10.0.2.5	SCTP	84	INIT
37	2.435370	10.0.2.5	10.0.2.15	SCTP	308	INIT_ACK
38	2.435825	10.0.2.15	10.0.2.5	SCTP	280	COOKIE_ECHO
39	2.435913	10.0.2.5	10.0.2.15	SCTP	52	COOKIE_ACK
40	2.437299	10.0.2.15	10.0.2.5	NGAP	136	NGSetupRequest
41	2.437342	10.0.2.5	10.0.2.15	SCTP	64	SACK
42	2.438342	10.0.2.5	10.0.2.15	NGAP	120	NGSetupResponse

Figure 2: Packages obtained while the GNB was registering.

sctp

No.	Time	Source	Destination	Protocol	Length	Info
22	4.389917	10.0.2.15	10.0.2.5	SCTP	84	INIT
23	4.389984	10.0.2.5	10.0.2.15	SCTP	308	INIT_ACK
24	4.390518	10.0.2.15	10.0.2.5	SCTP	280	COOKIE_ECHO
25	4.390579	10.0.2.5	10.0.2.15	SCTP	52	COOKIE_ACK
26	4.391628	10.0.2.15	10.0.2.5	NGAP	136	NGSetupRequest
27	4.391654	10.0.2.5	10.0.2.15	SCTP	64	SACK
28	4.392166	10.0.2.5	10.0.2.15	NGAP	120	NGSetupResponse
29	4.392606	10.0.2.15	10.0.2.5	SCTP	64	SACK
49	8.851808	10.0.2.15	10.0.2.5	SCTP	62	SHUTDOWN
50	8.851885	10.0.2.5	10.0.2.15	SCTP	52	SHUTDOWN_ACK
51	8.852261	10.0.2.15	10.0.2.5	SCTP	62	SHUTDOWN COMPLETE

```
> Frame 22: 84 bytes on wire (672 bits), 84 bytes captured (672 bits)
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 10.0.2.15, Dst: 10.0.2.5
> Stream Control Transmission Protocol, Src Port: 58742 (58742), Dst Port: 38412 (38412)
```

Figure 3: Packages obtained while the GNB disconnecting.

It is possible to see in the results the messages exchanged between GNB and the core and that the establishment of the NG Setup is successful, i.e. the GNB was able to connect successfully to the core and receive UEs compatible with its parameters, it is also possible to see that the GNB is able to disconnect sucessfully from the core.

4.2 UE Connectivity

4.2.1 UE Registration

When a new UE starts the registration procedure, it was expected that the UE requests access to the Radio Access Network (RAN) that recognizes a new UE and sends a message to initialize the UE to the AMF that verifies the request and sends an authentication request to the Authentication Server Function (AUSF), after that the UE and the network authenticate each other. Then the security settings and policies are initiated. Only at the end of this, the UE registration is completed. This procedure is represented in the diagram shown below.

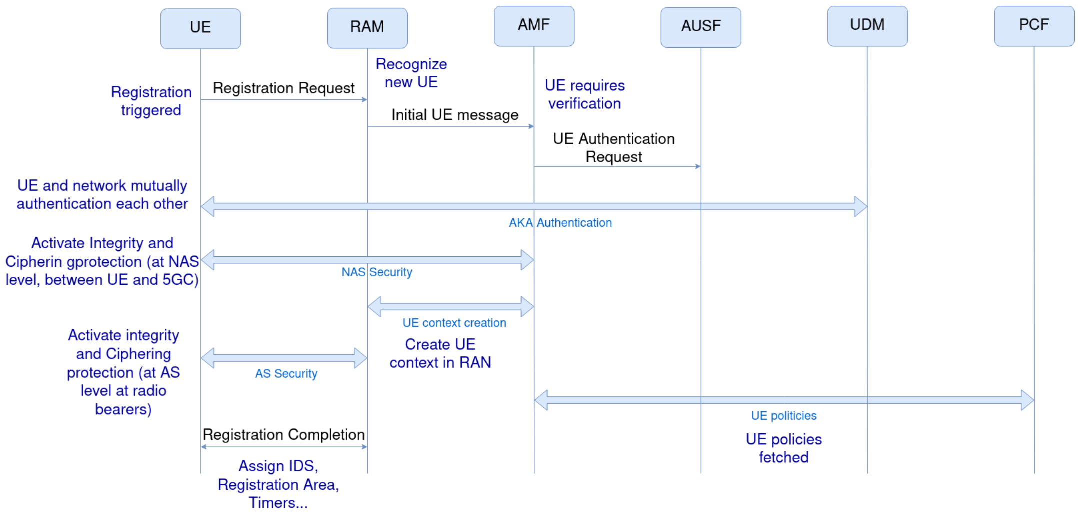
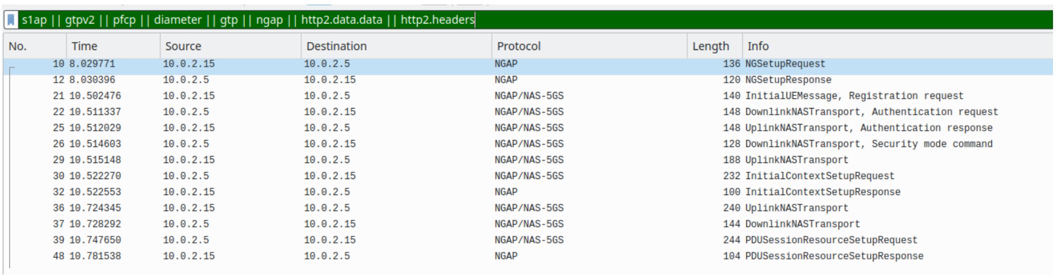


Figure 4: UE register package diagram.

To verify this packet communication a packet capture using wireshark between the core and UE was performed(Figure 5), using as filter s1ap || gtpv2 || pfcp || diameter || gtp || ngap || http2.data.data || http2.headers, the following packets can be seen: UplinkNASTransport, Authentication response, PDUSessionResourceSetupResponse, PDUSessionResourceSetupRequest, NGSetupResponse, NGSetupRequest, InitialUEMessage, Registration request, InitialContextSetupResponse, InitialContextSetupRequest, DownlinkNASTransport, Security mode comman and Authentication request as expected. We also expected to capture some http packets, this didn't happen because we are working in VMs with limited resources so wireshark didn't catch all the packets.


s1ap || gtpv2 || pfcp || diameter || gtp || ngap || http2.data.data || http2.headers

No.	Time	Source	Destination	Protocol	Length	Info
10	8.029771	10.0.2.15	10.0.2.5	NGAP	136	NGSetupRequest
12	8.030396	10.0.2.5	10.0.2.15	NGAP	120	NGSetupResponse
21	10.502476	10.0.2.15	10.0.2.5	NGAP/NAS-5GS	140	InitialUEMessage, Registration request
22	10.511337	10.0.2.5	10.0.2.15	NGAP/NAS-5GS	148	DownlinkNASTransport, Authentication request
25	10.512029	10.0.2.15	10.0.2.5	NGAP/NAS-5GS	148	UplinkNASTransport, Authentication response
26	10.514603	10.0.2.5	10.0.2.15	NGAP/NAS-5GS	128	DownlinkNASTransport, Security mode command
29	10.515148	10.0.2.15	10.0.2.5	NGAP/NAS-5GS	188	UplinkNASTransport
30	10.522270	10.0.2.5	10.0.2.15	NGAP/NAS-5GS	232	InitialContextSetupRequest
32	10.522553	10.0.2.15	10.0.2.5	NGAP	100	InitialContextSetupResponse
36	10.724345	10.0.2.15	10.0.2.5	NGAP/NAS-5GS	240	UplinkNASTransport
37	10.728292	10.0.2.5	10.0.2.15	NGAP/NAS-5GS	144	DownlinkNASTransport
39	10.747650	10.0.2.5	10.0.2.15	NGAP/NAS-5GS	244	PDUSessionResourceSetupRequest
48	10.781538	10.0.2.15	10.0.2.5	NGAP	104	PDUSessionResourceSetupResponse

Figure 5: Packages obtained from the UE register.

4.2.2 UE Internet Connectivity

To test the internet connectivity some pings were done between different UEs and google.com.

The obtained results are shown below:

```
ueran@ueran:~/UERANSIM$ ping google.com -I uesimtun0 -c 10
PING google.com (172.217.17.14) from 10.45.0.3 uesimtun0: 56(84) bytes of data.
64 bytes from google.com (172.217.17.14): icmp_seq=1 ttl=56 time=25.7 ms
64 bytes from google.com (172.217.17.14): icmp_seq=2 ttl=56 time=23.0 ms
64 bytes from google.com (172.217.17.14): icmp_seq=3 ttl=56 time=23.3 ms
64 bytes from google.com (172.217.17.14): icmp_seq=4 ttl=56 time=23.8 ms
64 bytes from google.com (172.217.17.14): icmp_seq=5 ttl=56 time=25.7 ms
64 bytes from google.com (172.217.17.14): icmp_seq=6 ttl=56 time=23.7 ms
64 bytes from google.com (172.217.17.14): icmp_seq=7 ttl=56 time=22.5 ms
64 bytes from google.com (172.217.17.14): icmp_seq=8 ttl=56 time=23.8 ms
64 bytes from google.com (172.217.17.14): icmp_seq=9 ttl=56 time=25.5 ms
64 bytes from google.com (172.217.17.14): icmp_seq=10 ttl=56 time=25.2 ms

--- google.com ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9014ms
rtt min/avg/max/mdev = 22.545/24.224/25.726/1.126 ms
```

Figure 6: Ping from UE to google.com.

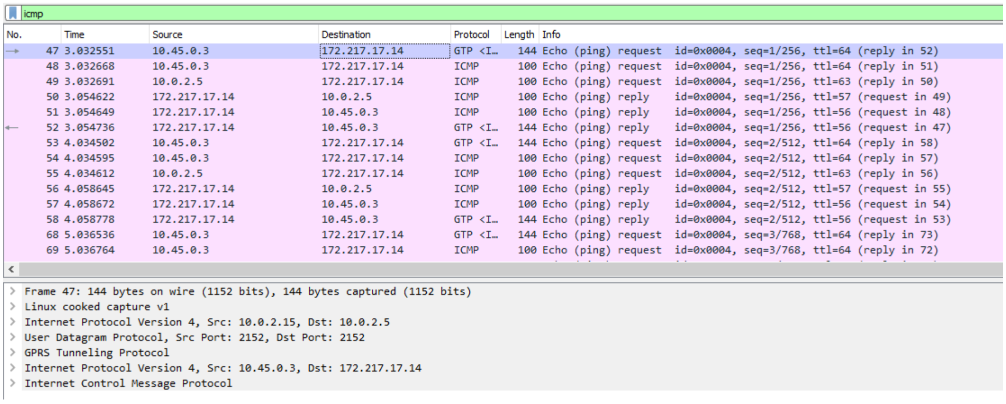


Figure 7: Capture done in Core.

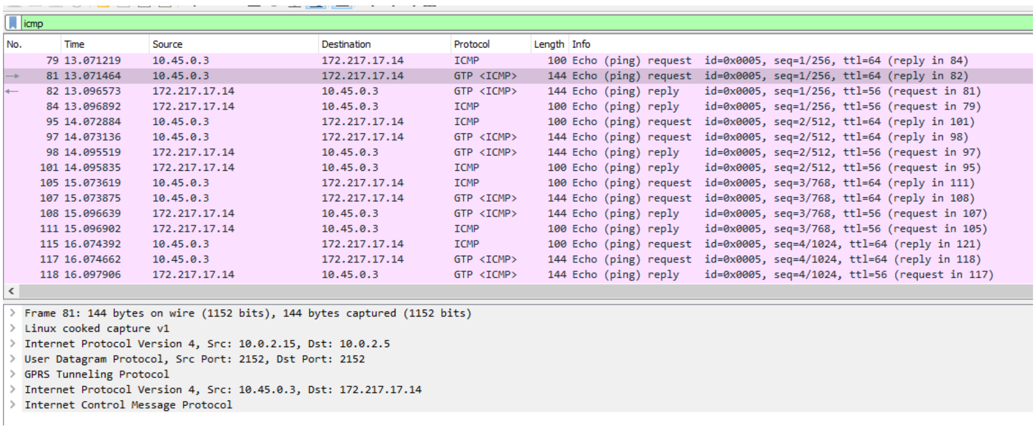


Figure 8: Capture done in TAC1.

The results match the expectation, that is we expected that the UE should be able to ping google.com and and the packets would travel from UE to the GNB then enter the GTP tunnel, go to through the core and to return do the reverse path.

To test if the UEs had connectivity between each other some pings between them were done.

The following images show the results:

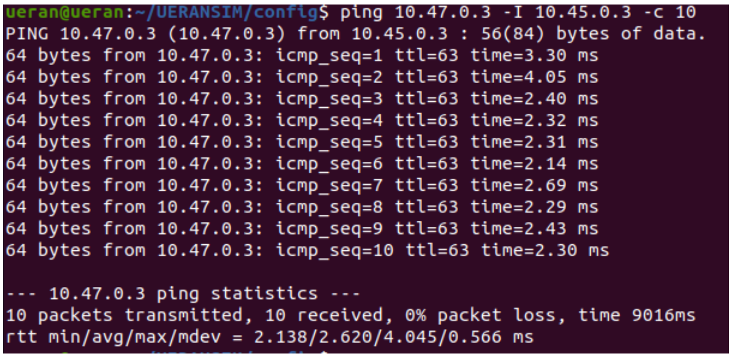


Figure 9: Ping between 2 UEs.

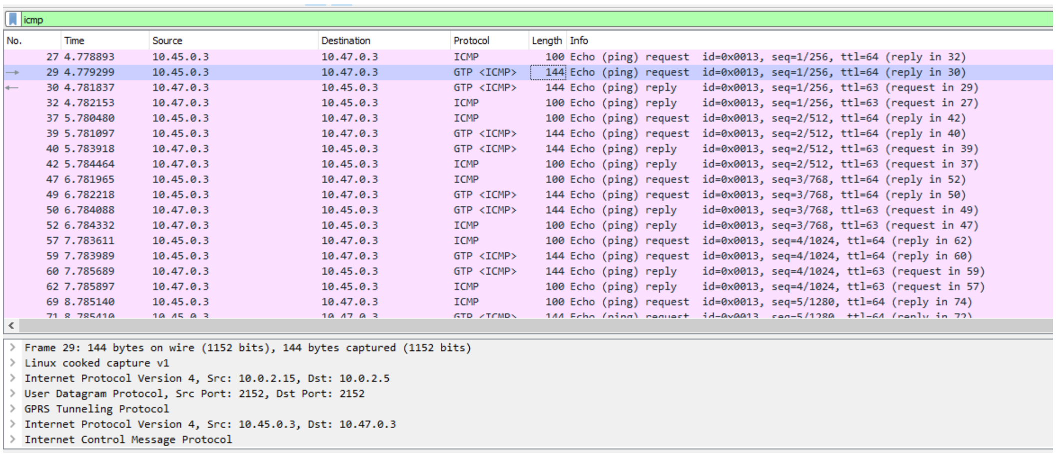


Figure 10: Ping between 2 UEs.

4.2.3 UE QoS Tests

To test the QoS parameters of the network various configurations were set in the Open5gs WebUI. To take conclusions about the effects of the various configurations the iPerf3 tool was used to measure the bandwidth of the different UEs, the iPerf3 server was attached to the output interface and the client was attached to the UE. The UEs choosen to this tests were the ones that connect to the mySlice DNNs and the UEs 42 was the one selected to receive each new configuration.

The list of different configurations is shown below:

- Default config. :
  - Session-AMBR Downlink and Session-AMBR Uplink = 1 Gbps
- Reduced Downlink config. :
  - Session-AMBR Downlink = 20 Mbps
  - Session-AMBR Uplink = 1 Gbps
- Reduced Uplink config. :
  - Session-AMBR Downlink = 1 Gbps
  - Session-AMBR Uplink = 20 Mbps

The results obtained are shown below:

Uplink - bandwidth (Mbits/seg)		
	UE 41	UE 42
	63,6	58,6
	56,8	59
	59,5	61,6
average	59,5	59

Table 1: Default Uplink configuration results.

Downlink - bandwidth (Mbits/seg)		
	UE 41	UE 42
	51,9	53,1
	51,8	53,3
	50	51,3
average	51,8	53,1

Table 2: Default Downlink configuration results.

To test if the 5G QoS Identifier (5QI) and ARP level priority had any impact on the bandwidth of the network 2 UEs were configured with the same 5QI but different ARP level priority, this UEs were then used by iPerf3 to measure their bandwidth. It was expected that the UEs with the smallest ARP priority level would have fewer bandwidth than the other.

The configurations used were:

- UE 41 config. :
  - 5QI: 65
  - ARP priority level: 15
- UE 42 config. :
  - 5QI: 65
  - ARP priority level: 1

The results obtained can be seen in the following tables:

Uplink - bandwidth (Mbits/seg)		
	UE 41	UE 42
	63,3	22,6
	56,9	23,4
	59,5	23,1
average	59,5	23,1

Table 3: Reduced Uplink configuration results.

Uplink - bandwidth (Mbits/seg)		
	UE 41	UE 42
	52,2	22,5
	51,8	22,3
	50	22,9
average	51,8	22,5

Table 4: Reduced Downlink configuration results.

bandwidth (Mbits/seg)		
	Uplink	Downlink
	60,2	50,4
	61,2	51,2
	61,1	51,8
average	61,1	51,2

Table 5: UE 41 test results.

bandwidth (Mbits/seg)		
	Uplink	Downlink
	56	50
	57,1	50
	59,6	50,7
average	57,1	50

Table 6: UE 42 test results.

After analysing the results obtained we can conclude that the ARP priority level doens't have almost any impact in the UEs bandwidth. These results were not satisfactory and we decided to test the network with an increased load, this was achieved by connection all of the available UEs to the network and testing their bandwidth at the same time, this produced different results every time the test was performed. This situation could only be explained after doing some research through the documention of Open5gs, with this research it was found that some QoS parameters were not implemented in the software, this means that although the values of 5QI can be changed they do not produce different results because the core treats them all the same way.

4.3 Slicing

As mentioned in the Architecture section the different UEs are in different slices depending on which DNN they connect to.

The next figure represents the situation mentioned above:

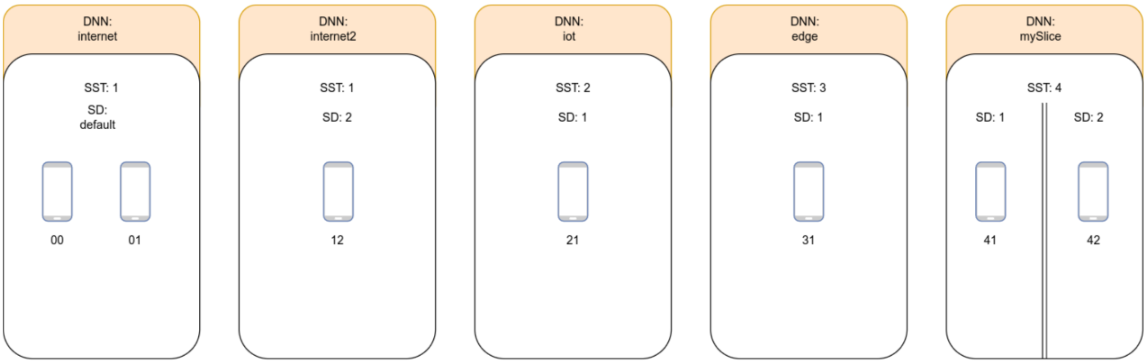


Figure 11: Slicing.

To test if the core was recognizing the different slices the log files of the AMF were checked.

The output of this log file can be seen in the next figure:

```
02/03 17:32:37.198: [amf] INFO: InitialUEMessage (../src/amf/ngap-handler.c:361)
02/03 17:32:37.201: [amf] INFO: [Added] Number of gNB-UEs is now 1 (../src/amf/context.c:2036)
02/03 17:32:37.201: [amf] INFO:     RAN_UE_NGAP_ID[1] AMF_UE_NGAP_ID[5] TAC[1] CellID[0x10] (../src/amf/ngap-handler.c:497)
02/03 17:32:37.230: [amf] INFO: [suci-0-901-70-0000-0-0-0000000000] known UE by SUCI (../src/amf/context.c:1386)
02/03 17:32:37.230: [gmm] INFO: Registration request (../src/amf/gmm-sm.c:134)
02/03 17:32:37.230: [gmm] INFO: [suci-0-901-70-0000-0-0-0000000000]    SUCI (../src/amf/gmm-handler.c:156)
02/03 17:32:37.448: [amf] INFO: [imsi-901700000000000:1] Release SM context [204] (../src/amf/amf-sm.c:451)
02/03 17:32:37.448: [amf] INFO: [Removed] Number of AMF-Sessions is now 2 (../src/amf/context.c:2054)
02/03 17:32:37.809: [gmm] INFO: [imsi-901700000000000] Registration complete (../src/amf/gmm-sm.c:1063)
02/03 17:32:37.810: [amf] INFO: [imsi-901700000000000] Configuration update command (../src/amf/nas-path.c:389)
02/03 17:32:37.810: [gmm] INFO:     UTC [2022-02-03T17:32:37] Timezone[0]/DST[0] (../src/amf/gmm-build.c:502)
02/03 17:32:37.810: [gmm] INFO:     LOCAL [2022-02-03T17:32:37] Timezone[0]/DST[0] (../src/amf/gmm-build.c:507)
02/03 17:32:37.811: [amf] INFO: [Added] Number of AMF-Sessions is now 3 (../src/amf/context.c:2048)
02/03 17:32:37.811: [gmm] INFO: UE SUPI[imsi-901700000000000] DNN[internet] S_NSSAI[SST:1 SD:0xffffff] (../src/amf/gmm-handler.c:1042)
02/03 17:33:47.401: [amf] INFO: InitialUEMessage (../src/amf/ngap-handler.c:361)
02/03 17:33:47.401: [amf] INFO: [Added] Number of gNB-UEs is now 2 (../src/amf/context.c:2036)
02/03 17:33:47.401: [amf] INFO:     RAN_UE_NGAP_ID[2] AMF_UE_NGAP_ID[6] TAC[1] CellID[0x10] (../src/amf/ngap-handler.c:497)
02/03 17:33:47.401: [amf] INFO: [suci-0-901-70-0000-0-0-0000000012] known UE by SUCI (../src/amf/context.c:1386)
02/03 17:33:47.401: [gmm] INFO: Registration request (../src/amf/gmm-sm.c:134)
02/03 17:33:47.401: [gmm] INFO: [suci-0-901-70-0000-0-0-0000000012]    SUCI (../src/amf/gmm-handler.c:156)
02/03 17:33:47.405: [amf] INFO: [imsi-901700000000012:1] Release SM context [204] (../src/amf/amf-sm.c:451)
02/03 17:33:47.405: [amf] INFO: [Removed] Number of AMF-Sessions is now 2 (../src/amf/context.c:2054)
02/03 17:33:47.641: [gmm] INFO: [imsi-901700000000012] Registration complete (../src/amf/gmm-sm.c:1063)
02/03 17:33:47.641: [amf] INFO: [imsi-901700000000012] Configuration update command (../src/amf/nas-path.c:389)
02/03 17:33:47.641: [gmm] INFO:     UTC [2022-02-03T17:33:47] Timezone[0]/DST[0] (../src/amf/gmm-build.c:502)
02/03 17:33:47.641: [gmm] INFO:     LOCAL [2022-02-03T17:33:47] Timezone[0]/DST[0] (../src/amf/gmm-build.c:507)
02/03 17:33:47.641: [amf] INFO: [Added] Number of AMF-Sessions is now 3 (../src/amf/context.c:2048)
02/03 17:33:47.641: [gmm] INFO: UE SUPI[imsi-901700000000012] DNN[internet2] S_NSSAI[SST:1 SD:0x2] (../src/amf/gmm-handler.c:1042)
```

Figure 12: Log file output.

With this log file and the fact that each UE behaves like the specification is possible to comprove that the network slicing if working correctly (in the last line of the log file is possible to see the recognition of the slice and sd).

5 Conclusion

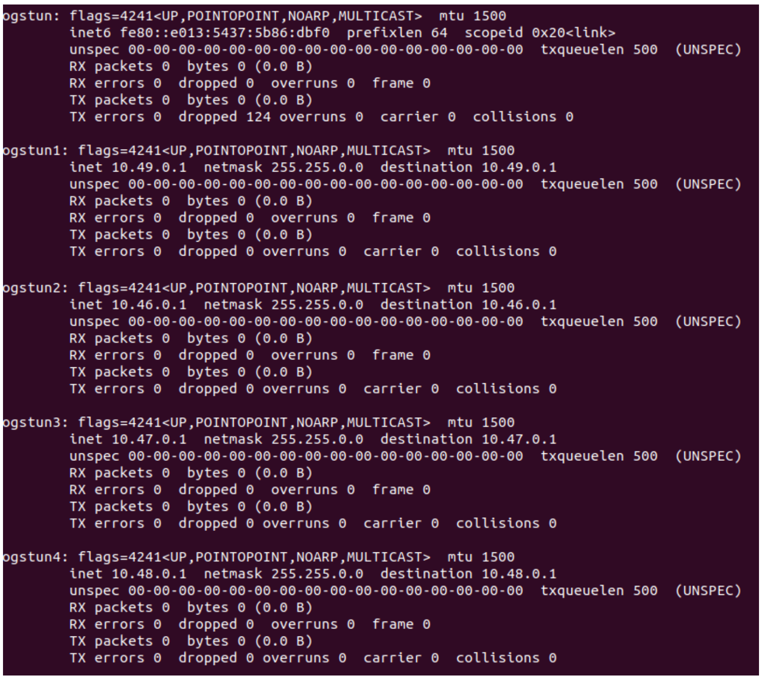
In this project we were able to implement a complete 5G System (core and access) as requested. In this system we implemented some slicing situations but other more realistic variants could be implemented, this was not possible because some features of QoS are not yet implemented in the software we use (open5gs). We can also conclude that this hands-on approach to implementing a 5G System helped us became more educated in 5G topic.

6 Appendix

To configure all of the ouput interfaces the following commands were done:

```
$ sudo ip tuntap add name ogstun1 mode tun
$ sudo ip tuntap add name ogstun2 mode tun
$ sudo ip tuntap add name ogstun3 mode tun
$ sudo ip tuntap add name ogstun4 mode tun

$ sudo ip addr add 10.49.0.1/16 dev ogstun1
$ sudo ip addr add 10.46.0.1/16 dev ogstun2
$ sudo ip addr add 10.47.0.1/16 dev ogstun3
$ sudo ip addr add 10.48.0.1/16 dev ogstun4

$ sudo ip link set ogstun1 up
$ sudo ip link set ogstun2 up
$ sudo ip link set ogstun3 up
$ sudo ip link set ogstun4 up
```

This are the results after the commands mentioned above were introduced:



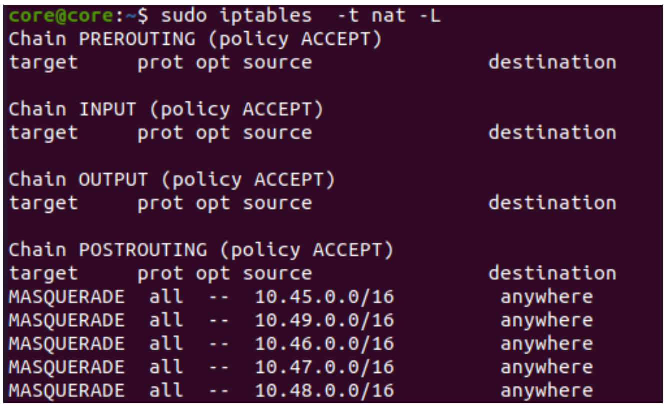
```
ogstun: flags=4241<UP,POINTOPOINT,NOARP,MULTICAST>  mtu 1500
        inet6 fe80::e013:5437:5b86:dbf0  prefixlen 64  scopeid 0x20<link>
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 500  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 124 overruns 0  carrier 0  collisions 0

ogstun1: flags=4241<UP,POINTOPOINT,NOARP,MULTICAST>  mtu 1500
        inet 10.49.0.1  netmask 255.255.0.0  destination 10.49.0.1
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 500  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ogstun2: flags=4241<UP,POINTOPOINT,NOARP,MULTICAST>  mtu 1500
        inet 10.46.0.1  netmask 255.255.0.0  destination 10.46.0.1
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 500  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ogstun3: flags=4241<UP,POINTOPOINT,NOARP,MULTICAST>  mtu 1500
        inet 10.47.0.1  netmask 255.255.0.0  destination 10.47.0.1
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 500  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ogstun4: flags=4241<UP,POINTOPOINT,NOARP,MULTICAST>  mtu 1500
        inet 10.48.0.1  netmask 255.255.0.0  destination 10.48.0.1
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 500  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 13: Interfaces of the Core VM

To allow the different UEs to have internet connectivity this rules needed to be added to the iptables of the core VM:

```
$ sudo iptables -t nat -A POSTROUTING -s 10.45.0.0/16 ! -o ogstun  -j MASQUERADE
$ sudo iptables -t nat -A POSTROUTING -s 10.49.0.0/16 ! -o ogstun1 -j MASQUERADE
$ sudo iptables -t nat -A POSTROUTING -s 10.46.0.0/16 ! -o ogstun2 -j MASQUERADE
$ sudo iptables -t nat -A POSTROUTING -s 10.47.0.0/16 ! -o ogstun3 -j MASQUERADE
$ sudo iptables -t nat -A POSTROUTING -s 10.48.0.0/16 ! -o ogstun4 -j MASQUERADE
```

This are the results after the commands mentioned above were introduced:



```
core@core:~$ sudo iptables  -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  10.45.0.0/16         anywhere
MASQUERADE  all  --  10.49.0.0/16         anywhere
MASQUERADE  all  --  10.46.0.0/16         anywhere
MASQUERADE  all  --  10.47.0.0/16         anywhere
MASQUERADE  all  --  10.48.0.0/16         anywhere
```

Figure 14: Iptables output

URLS:

1. https://open5gs.org/open5gs/docs/guide/01-quickstart/
2. https://github.com/aligungr/UERANSIM/wiki
3. https://github.com/s5uishida/open5gs_5gc_ueransim_sample_config
4. https://github.com/s5uishida/open5gs_5gc_ueransim_nearby_upf_sample_config