Optimal SDN managed non-IP transport for 5G networks

Aaron Montilla Vicent

School of Electrical Engineering

Bachelor's thesis
Espoo 13.6.2019

Supervisor

Prof Raimo Kantola

Advisor

Dr Jose Costa Requena

Aalto University
School of Electrical
Engineering

Aalto University
School of Electrical
Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the bachelor's thesis

Author Aaron Montilla Vicent

Title Optimal SDN managed non-IP transport for 5G networks

Degree programme Electronics and electrical engineering

Major Telecommunication Engineering **Code of major** ELEC0007

Teacher in charge Prof Raimo Kantola

Advisor Dr Jose Costa Requena

Date 13.6.2019 **Number of pages** 46 **Language** English

Abstract

The wide spread of smartphones and tablets leads to a concern about the bandwidth and resources consumption. For this reason, a new and better transport solution is needed to achieve a more efficient use of the network capacity.

In this project, the aim will be to achieve a good method to optimize the transport in the network. The implementation will be done with a modification of the GTP tunnel managed by a centralized SDN controller. This new transport will be added to an SDR network to deliver optimized transport in 5G networks.

Keywords srsLTE, USRP, connection, eNode B, EPC, SDN , bandwidth

Contents

Abbreviations

EPC	Evolved Packet Core
eNB	Evolved Node B
USRP	Universal Software Radio Peripheral
LTE	Long Term Evolution
SDN	Software Defined Networking
GTP	GPRS Transport Protocol
GPRS	General Packet Radio Service
TEID	Tunnel Endpoint Identifier
MCC	Mobile Country Code
MNC	Mobile Network Code
HSS	Home Subscriber Server
AMF	Access and Mobility Management Function
SMF	Session Management Function
MME	Mobility Management Entity
UPF	User Plane Function
APU	Accelerated Processing Unit
UE	User Equipment
EARFCN	E-UTRA Absolute Radio Frequency Channel Number
UHD	USRP Hardware Driver
VLAN	Virtual Local Area Network
SDR	Software Defined Radio
OVS	Open Virtual Switch
ARP	Address Request Protocol
UDP	User Datagram Protocol
NOMA	Non-Orthogonal Multiple Access
OMA	Orthogonal Multiple Access
MAC	Media Access Control
IP	Internet Protocol
TCP	Transmission Control Protocol
OPEX	Operative Expensiveness
SGW	Service Gateway
RACH	Radio Access Channel
PRACH	RACH Preamble
SQL	Structured Query Language
IMSI	International Mobile Subscriber Identity
ID	Identifier
GSM	Global System for Mobile Communication
UMTS	Universal Mobile Telecommunications System
PDCP	Packet Data Converge Protocol

1 Introduction

The huge increase in the use of bandwidth and resources consumption due to the wide spread of smartphones and tablets is leading to a concern about these.

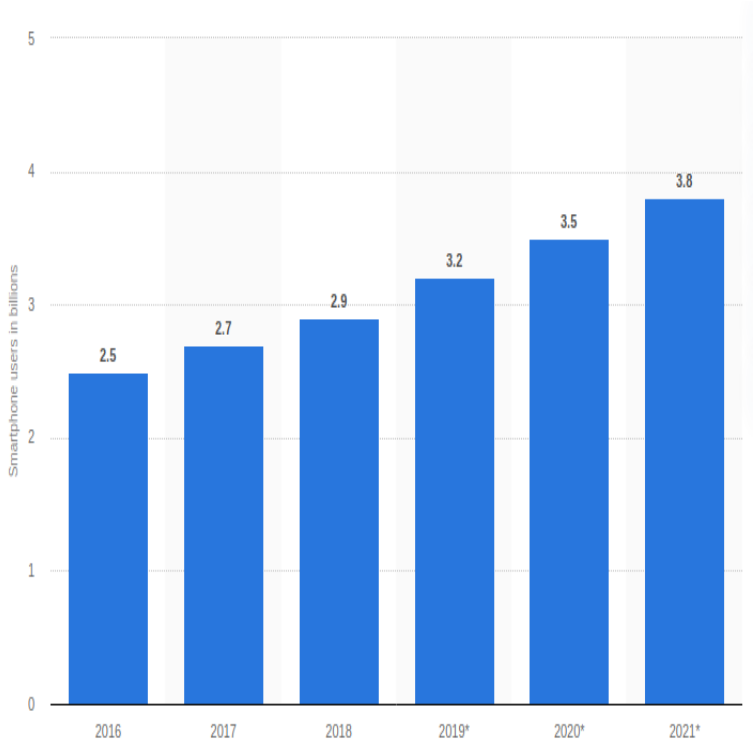

Figure 1: Number of smartphones in billions per year between 2016-2021 [1].

As the graph shows, the increase in the number of smartphones is unstoppable, reaching 3.2 billions in 2019 and with a prevision of 3.8 billions in 2021. This problem will increase with the incoming 5G networks and the amount of connected devices that these will make possible. According to the specifications of the incoming system it will suppose an increment of 100 times over the 4G in terms of devices connected per unit of area. Moreover, 5G systems are suppose to provide an increment of the data rate between 10 and 100 times over 4G networks.

This situation will be unsustainable with the methods that the networks are using currently, this is the reason why new methods and implementations are needed to solve this problem which will get worse with time.

The future methods must provide the existing system with implementations for providing an increment in the efficiency of the use of the available bandwidth and resources. SDN (Software Defined Networking) is really important to achieve the efficiency that these new networks need. With SDN it is possible to achieve dynamic and scalable networks.

SDN architectures, by definition, decouple network control and forwarding functions, enabling the network control to become directly programmable and the underlying infrastructure to be abstracted from applications and network services [2].

In other words, it implies the separation of the control plane and the data plane of the network. The addition of SDN to the network allows centralized and flexible networks and reducing time and complexity in troubleshooting. That is why most of the previous proposed solutions and the proposed solution of this project use SDN.

The arrival of SDN came from the necessity of a new network architecture that allow the massive use of the systems. Some of the key computing trends driving the need for a new network paradigm include:

- Changing traffic patterns: Today's application access different databases and servers. The traditional client-server architecture was not longer suitable for these applications. This attribute has been the main reason of the addition of SDN in this the system and will be explained with more detail in the following sections.

- The rise of cloud services: Lots of companies have embraced both public and private cloud services. The enterprises want to access to this resources in an agile way and on demand. Moreover, they require to perform this actions in a highly secured environment.

- The "consumerization of IT": The previously explained increment of the number of personal devices used by the users leads to a difficulty to accommodate all these users within the system.

2 Previous solutions for this project

Previously, several solutions have been proposed to face the problem of bandwidth and resources consumption.

The approach of many solutions consisted in NS (Network Slicing) where both NFV (Network Function Virtualization) and SDN (Software Defined Network) are used. Network functions virtualization is a network architecture concept that uses the technologies of IT virtualization to virtualize entire classes of network node functions into building blocks that may connect, or chain together, to create communication services. NFV allows the widest possible flexibility as to the physical location of the virtualized functions[3]. Therefore, in the ideal situation, virtualized functions are placed where they are the most efficient but the least expensive at the same time.

This will allow a flexible Network Functions deployment and thus increasing the efficiency of the network. Flexible deployment of Network Functions is about creating and removing these functions when they are needed and locate them where we want in the network for an optimal functioning. The creation and removing of the NF as well as the location are managed by an SDN Switch making possible the reduction of the OPEX of the network.

So the Network Slicing architecture can be summarize into two main blocks. The first one is responsible of the slice implementation and the other one to the management of the slices.

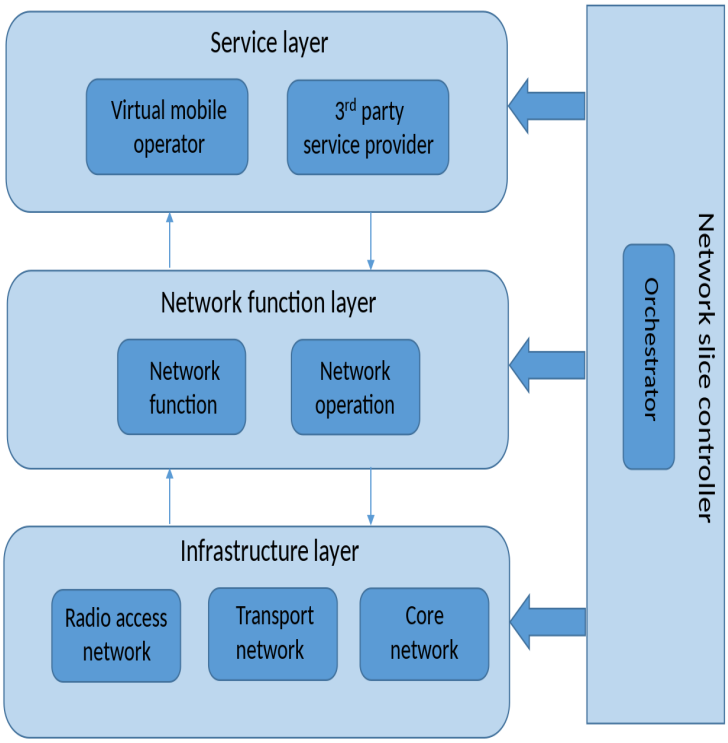

Figure 2: Network Slicing general architecture[4].

Other proposed solutions are about the multiple access protocol called NOMA

(Non-Orthogonal Multiple Access) which is a new proposal for encoding technology. In NOMA, several users of the network can use the bandwidth at the same time due to the difference in the power used. In the OMA (Orthogonal Multiple Access) methods, the problem was that the spectral efficiency was low when the resources were used by users with poor CSI (Channel State Information). Now with NOMA, the bandwidth used by the users with poor CSI can still be accessed by the users high CSI, increasing the spectral efficiency significantly [5].

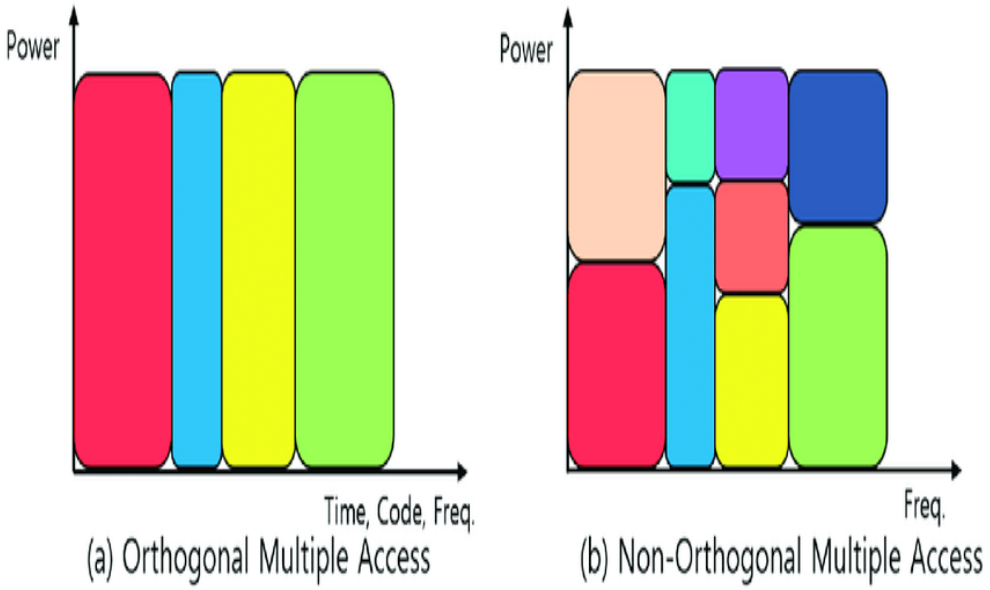

Figure 3: OMA and NOMA schemes. [6]

With the addition of this protocol the network will achieve massive connectivity with lower latency. These solutions are about the combination of this protocol with the MIMO technologies, cooperative NOMA and also the interplay between cognitive radio and NOMA.

3 Proposed solution

The solution proposed in this project focuses in the implementation of a new transport method. The objective is to integrate a Software based radio node with this new transport method to deliver optimized transport for 5G networks. This new method will be part of the network backhaul that will allow to deliver optimized and low latency transport managed from a centralized SDN controller.

The proposed solution will have the following steps are the main ones:

- Building a SDR based eNB using srsLTE.
- Integrate SDN functionality into the new eNB.
- Add L2 transport functionality to SDR eNB.
- Management of the network transport from eNB using SDN.

The project was faced with no previous knowledge in neither Linux environment nor the use of a USRP.

The hardware used in this project is:

- Laptop model HP ZBook 15 G3 with Ubuntu 18.04.
- ASUS laptop with Windows 10.
- USRP model x310.
- APU Box.
- Cell phone model Samsung J6
- SIM card reader
- USB-Ethernet cables adapter

The software used to implement the network utilities is:

- srsLTE (srsenb part).
- GSim Writer.
- EPC Cumucore software.
- UHD Repository
- GNU Radio

4 Installation

4.1 UHD and GNU installation

Firstly, the uhd-host and UHD repository must be installed in the Ubuntu laptop. The UHD repository is a free and open-source software driver and API for the Universal Software Radio Peripheral (USRP). This software is created and distributed by the Ettus Research company [7]. The installation of this repository will include all the libraries needed for the management of the USRP in the laptop. Depending on the USRP used, the installation will proceed differently. This project is using the model x310 which belongs to the third generation of these devices. To proceed with the installation run the following commands:

```
$ sudo apt-get update
```

After this command it is necessary to install the proper libraries. The command for the libraries is in the appendix A

At this point, git is needed. Git is a distributed version-control system for tracking changes in source code during the development of a project. This software is really used and useful, because it does not only permits an easy coordination among programmers, it allows the access and download of many different versions that have been uploaded in the different versions of the software. In addition, Git is also a free and open-source software. If this software had not been already used, it can be installed with the following command [8]:

```
$ sudo apt install git
```

Now for downloading the UHD repository and install it:[9]

```
$ git clone --recursive git://github.com/EttusResearch/uhd.git
$ cd host
$ mkdir build
$ cd build
$ cmake ../
$ make
$ make test
$ sudo make install
$ sudo ldconfig
```

And finally GNU Radio. GNU Radio is a free and open-source software development toolkit that provides signal processing blocks to implement software radios. It

is commonly used with an external hardware but it also can be used without any other hardware thanks to the software environment that is already included in the software of GNU [10]. Again, the following commands will include the software in the computer:[9]

```
$ cd ~/offline/src/gnuradio
$ git checkout v3.7.10.2
$ mkdir build
$ cd build
$ cmake ..
$ make -j4
$ sudo make install
$ sudo ldconfig
```

After completing the installation the command "uhd_find_devices" and "uhd_usrp_probe" have to be able to find the USRP but only if IP address are configured correctly.

4.2 srsLTE installation

The next software that must be installed is srsLTE which will simulate the eNode B part of the network.

srsLTE is an open-source LTE software suite developed by SRS. It includes:

- srsUE: a complete SDR LTE UE application featuring all layers from PHY: to IP.
- srsENB: a complete SDR LTE eNodeB application.
- srsEPC: a light-weight LTE core network implementation with MME, HSS and S/P-GW.
- A highly modular set of common libraries for PHY, MAC, RLC,PDCP, RRC, NAS, S1AP and GW layers.

The following commands download and install this software[11]:

```
$ git clone https://github.com/srsLTE/srsLTE.git
$ cd srsLTE
$ mkdir build
$ cd build
$ cmake ../
$ make
$ make test
$ sudo make install
$ srslte_install_configs.sh user
```

srsLTE was not the first option for this part of the software. In fact, the first try was the installation of OAI (Open Air Interface) but after two weeks trying to install the software, it was impossible to achieve. So a change in the software was preferable and srsLTE was the chosen software. Once it is installed, enb.conf file of srsLTE must be configured properly , specially MCC, MNC, and the IP addresses mme_addr, gtp_bind_addr and s1c_bind_addr. Also the dl_earfcn parameter is really important for the connection with the cell phone. Every configuration file of the srsLTE program as used in this project are included in appendix B. In the installation of the program, the enb.conf file includes more parameters which were not used in this project and have been removed.

At this point, all the installations needed in the Ubuntu laptop are done. Now the APU box has to be configured following the instructions of the Cumucore manual for EPC installation.

5 SIM configuration

In the configuration of the SIM card the GSIM Writer software is needed. This program only runs on Windows, so this part of the project was executed in the other laptop. There are few parameters that are important in the configuration:

1. IMSI number
2. K parameter
3. Op parameter

The other parameters can be assigned randomly with the proper buttons of the program. But the parameters of the previous list must be assigned properly to connect with the EPC and the eNode B. The IMSI (International Mobile Subscriber Identifier) is a unique code for a mobile that enables the identification with GSM and UMTS. The K and Op parameters are used to get all the codes needed for the authentication. So it is necessary to ensure that the parameters fits the authentication of the system and the MCC and MNC numbers.

After initializing the parameters in the SIM card,the parameters in the EPC part must be checked. The IP addresses of the AMF and SMF part must agree and the data of the user (cell phone) has to be included in the HSS (Home Subscriber Server) part that has a MySQL structure, so it has to be included using MySQL parameters.

The HSS is the master user database that supports the IMS (IP Multimedia Subsystem) network for handling the calls/sessions. Inside the server remains all the information needed of the users. Although the tables of the HSS include several parameters, the only important are the ones mentioned in the list above.

With all the parts installed, the first tests of the network can start. Despite following all the previous steps, several errors can appear in the network depending on the environment used. So the problems found in the implementation of this project will be explained in the next section.

6 Common errors in the implementation

At the beginning of this project the ASUS laptop used for the Windows part was also used for the Ubuntu part, producing some errors that will be explained later on in this section. But the mistakes are going to be explained in the order that they appeared in this implementation.

6.1 IP addresses

Several problems with concordance of the IP addresses of the laptop used, the EPC part and the authentication server were faced in this implementation, but these can be avoided following the tips of the installation of the EPC part. However, these errors can make that the laptop cannot find the USRP, wrong functioning of the EPC part and also that the eNode B part cannot find the EPC for the connection.

6.2 EPC and eNB connection

Once IP addresses errors were solved, EPC and eNode B could connect. But the connectivity was really bad. When the eNode B part of the srsLTE command was run, it appeared an error message "Error, timed out while receiving samples from UHD". The reason of this error was that the adapter of the Ethernet cables was connected to the regular USB port. Notice that the Ethernet cables must be connected to the Ethernet port or to the USB 3.0 port if the adapter is being used. Otherwise, the connection would not have the proper speed causing this error. After solving this error, the connection between EPC and eNode B worked properly. If this connection is working, eNode B and its data will appear in EPC logs.

6.3 Frequency band allocation

After the previous connection, the only one missing is the UE connection. So the customized SIM card must be inserted into the cell phone. The best part to achieve this connection is setting the cell phone to plane mode and disconnecting plane mode when EPC and eNode B are connected. At this point the next error appeared. When plane mode was disconnected the phone signals did not appear in the logs of the network. In fact, the cell phone was not able to find the network (the network name is the junction of the MNC and MCC). This error was caused because the network was not working in the same frequency bands as the cell phone. Notice that some cell phones are working in several frequency bands while others only work in few frequencies. Frequency bands where your phone works can be found in Google.

To change the frequency which the network is working in the dl_earfcn parameter of the enb.conf file is needed. It is really important to change this parameter instead of the downlink and uplink ones. An earfcn calculator can be used to check the proper earfcn for the band needed.

After solving this problem, the signals from the cell phone appeared in the logs of both the EPC and the eNode B but the link with the did not work properly.

6.4 Cell phone connection

At this point, the PRACH messages appeared in the logs of the eNode B. In fact, it even achieved the connection in some tests but it lasted only few seconds and the strength of the connection was really low. The cause of this problem was that the ASUS laptop's CPU was not good enough to process the signals from the UE resulting in a bad connectivity.

To get this problem it was necessary to ask in the srsLTE mailing list. It is really useful because errors can be checked by the creators of the software and also previous emails with questions can be found in the web page of the mailing list. UHD also has its own mailing list for problems with the USRP and its connection.

That was the reason of the change of laptop in this project. The new laptop had enough CPU to process all the petitions. This made all the network work properly and the connection of the cell phone is excellent for an SDR based network.

7 Working instructions

7.1 EPC instructions

In this project three command windows are used to run the network. One window is used to run the AMF and SMF part of the EPC, another one is used for running the UPF part of the EPC and the third one is used to run the srsLTE software. Firstly, the UPF part is going to be run. It is important to run this first for the proper functioning of the data plane. After this, the AMF and SMF part need to be run to complete the EPC part. At this point something similar to this will be shown in the logs of each part

```
-- Logs begin at Thu 2017-03-02 03:00:57 EET. --
Dec 04 15:49:44 ubuntu amf_ciot[1059]: 15:49:44.824206  INFO - PLMN                          = 42f425
Dec 04 15:49:44 ubuntu amf_ciot[1059]: 15:49:44.824258  INFO - AMF S10 ADDR  = 127.0.0.12
Dec 04 15:49:44 ubuntu amf_ciot[1059]: 15:49:44.824312  INFO - MYSQL HOST         = localhost
Dec 04 15:49:44 ubuntu amf_ciot[1059]: 15:49:44.824366  INFO - MYSQL DATABSE NAME = hss_lte_db
Dec 04 15:49:44 ubuntu amf_ciot[1059]: 15:49:44.824419  INFO - MYSQL USERNAME = hss
Dec 04 15:49:44 ubuntu amf_ciot[1059]: 15:49:44.824445  INFO - AMF S11U ADDR(NB-IOT) = 127.0.0.10
Dec 04 15:49:44 ubuntu amf_ciot[1059]: 15:49:44.824471  INFO - TAC = 12594
Dec 04 15:49:44 ubuntu amf_ciot[1059]: 15:49:44.824527  INFO - TAC = 0
Dec 04 15:49:52 ubuntu amf_ciot[1059]: 15:49:52.828511  INFO - Trying to connect to Auth Server ...
Dec 04 15:49:52 ubuntu amf_ciot[1059]: 15:49:52.829201  INFO - Connected to Auth server.
Dec 04 15:49:52 ubuntu amf_ciot[1059]: 15:49:52.829263  INFO - Trying to send auth request ...
Dec 04 15:49:52 ubuntu amf_ciot[1059]: 15:49:52.829366  INFO - Auth request sent.
Dec 04 15:49:52 ubuntu amf_ciot[1059]: 15:49:52.829421  INFO - Waiting for auth response
Dec 04 15:49:52 ubuntu amf_ciot[1059]: 15:49:52.861955  INFO - Auth Successful.
Dec 04 15:49:52 ubuntu amf_ciot[1059]: 15:49:52.877881  INFO - HSS Connection Initialized
Dec 04 15:49:52 ubuntu amf_ciot[1059]: 15:49:52.878663  INFO - Initializing S11-U interface ....
Dec 04 15:49:52 ubuntu amf_ciot[1059]: 15:49:52.878826  INFO - S11-U Interface Initialized
Dec 04 15:49:52 ubuntu amf_ciot[1059]: 15:49:52.879602  INFO -   "name":        "demovepc"
Dec 04 15:49:52 ubuntu amf_ciot[1059]: 15:49:52.879669  INFO -   "port":        null
Dec 04 15:49:52 ubuntu amf_ciot[1059]: 15:49:52.879731  INFO -   "s1_adr":         "172.16.0.1"
Dec 04 15:49:52 ubuntu amf_ciot[1059]: 15:49:52.879759  INFO -   "smf_s11_adr":        "127.0.0.1"
Dec 04 15:49:52 ubuntu amf_ciot[1059]: 15:49:52.879816  INFO -   "m3_adr":        "127.0.0.1"
Dec 04 15:49:52 ubuntu amf_ciot[1059]: 15:49:52.879872  INFO -   "mbms_adr":         "127.0.0.5"
Dec 04 15:49:52 ubuntu amf_ciot[1059]: 15:49:52.879935  INFO -   "disco_adr":        "127.0.0.1"
Dec 04 15:49:52 ubuntu amf_ciot[1059]: 15:49:52.881406  INFO - M3 connection on 127.0.0.1 : 36444
Dec 04 15:49:52 ubuntu amf_ciot[1059]: 15:49:52.881518  INFO - Server on 172.16.0.1 : 36412
Dec 04 15:49:52 ubuntu amf_ciot[1059]: 15:49:52.881551  INFO - S11 connection on 127.0.0.1 : 2123
Dec 04 15:49:52 ubuntu amf_ciot[1059]: 15:49:52.881591  INFO - MBMS S11 connection on 127.0.0.5 : 2123
Dec 04 15:49:52 ubuntu amf_ciot[1059]: 15:49:52.881659  INFO - S10 Connection opened on Addr = 127.0.0.12  PORT = 2123
Dec 04 15:49:52 ubuntu amf_ciot[1059]: 15:49:52.881723  INFO - S10 Sending connection opened on Addr = 10.128.2.240  PORT = 2123
```

Figure 4: AMF logs without UE connected.

```
(venv) root@ubuntu:/home/ubuntu/upfgw# ./bin/upfgw -c etc/upfgw.conf
INFO     - [__main__                          <module>()] Loaded config
INFO     - [__main__                          __init__()] Init upfgw
INFO     - [upfgw.gtpdev                open_echo_sock()] Listening GTP1U-ECHO on 172.16.0.1:2152
INFO     - [upfgw.quehandler                  __init__()] Init queue handler
INFO     - [upfgw.quehandler           open_reply_sock()] Opening queue-control socket
INFO     - [__main__                    open_udp_sock()] Starting UDP listener
INFO     - [__main__                          __init__()] Run main loop
```

Figure 5: UPF logs without UE connected.

7.2 eNB instructions

Next, eNode B part must be run. So, in the third and last window in the srsenb folder the command that has to be run is "sudo srsenb enb.conf". This command is recommended otherwise the parameters of the enb.conf are not used in the configuration and the values have to be included in the command adding complexity.

If all problems explained before are solved the eNode B will connect to the EPC and these will be the logs of both parts:

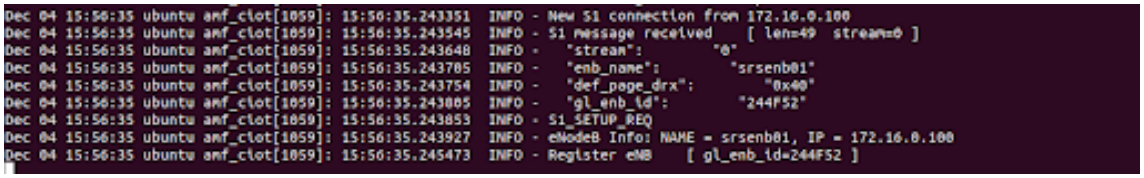

Figure 6: eNode B data in AMF logs.

So the last part is the UE connection as explained before. So after disconnecting plane mode in the phone the logs of the EPC and eNB must be like this:

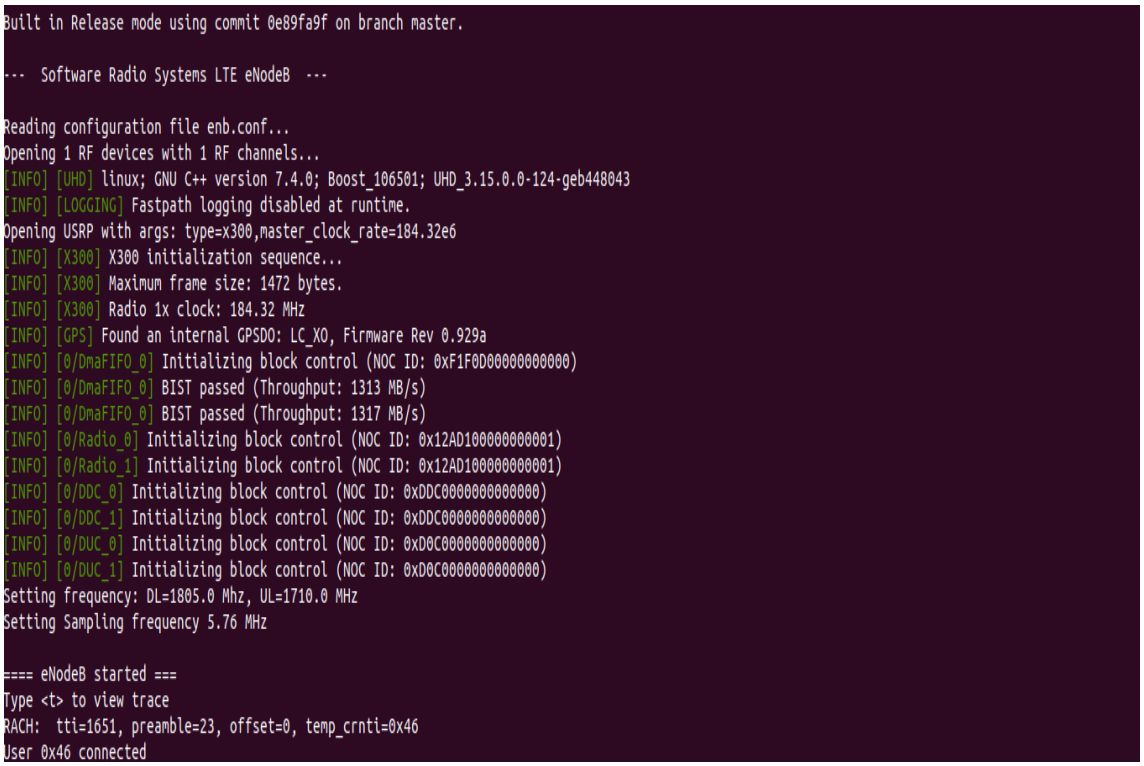

Figure 7: eNode B logs with UE connected.

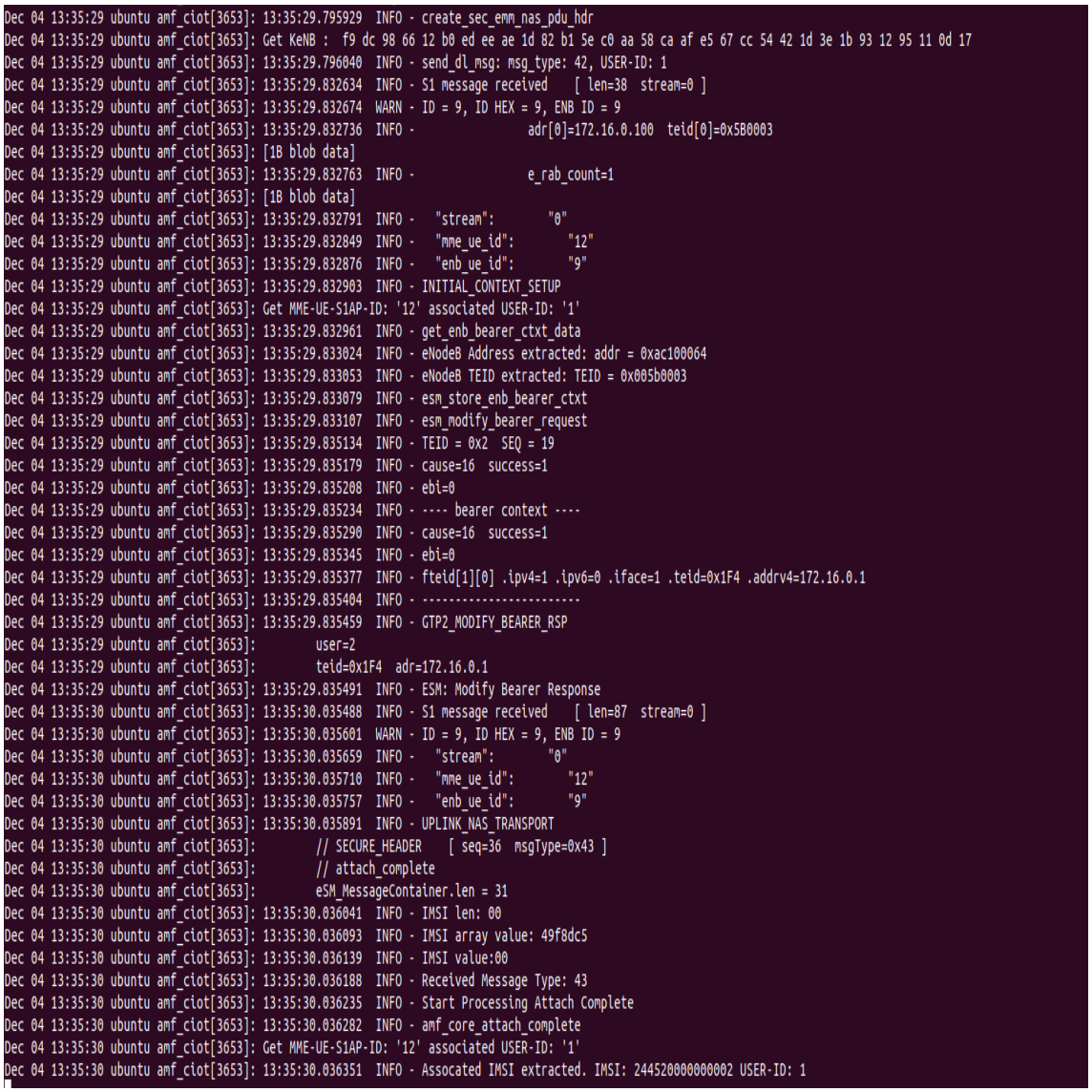

Figure 8: AMF logs with UE connected.

At this point the cell phone has Internet connection that can be use for whatever we need.

8 Results

With all the parts of the network connected it was time to check the results of the tests. One important improvement was the time that the network lasted to connect. Before changing the laptop, it took some unsuccessful connection attempts before the correct connection but after the change it only took one try. Additional tests like disconnecting eNode B with the UE connected and then binding it again were made and all of them succeeded.

After that, the speed of the connection was checked but that could not be tested without Internet connection. So the ping application was used to check whether the Internet connection was truly active. This application works as the ping command in the command line, so the IP direction 8.8.8.8 was used to check it and the response verified the connection . After making sure that the connection worked, the next and more advanced test used the "Speed test" application. Both application were downloaded from Google Play for free.

As a first step, the app needs to find the server and this can lasts few seconds even with a good connection. After this, the environment is ready the app can make the test. In this project several test have been made in the network. Taking into account the test it is visible that the network works better after some time of connection. However, the difference is not really significant. Here there are two examples of the tests:

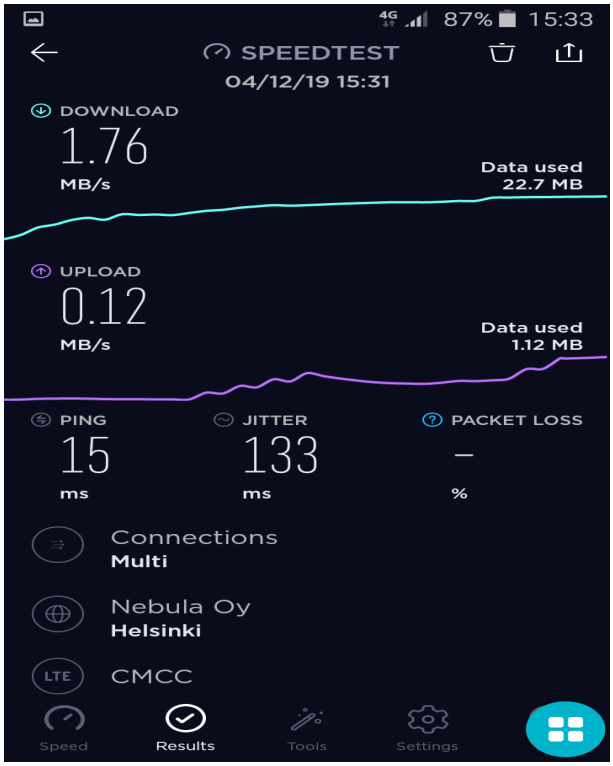

Figure 9: Test when the phone just connected.

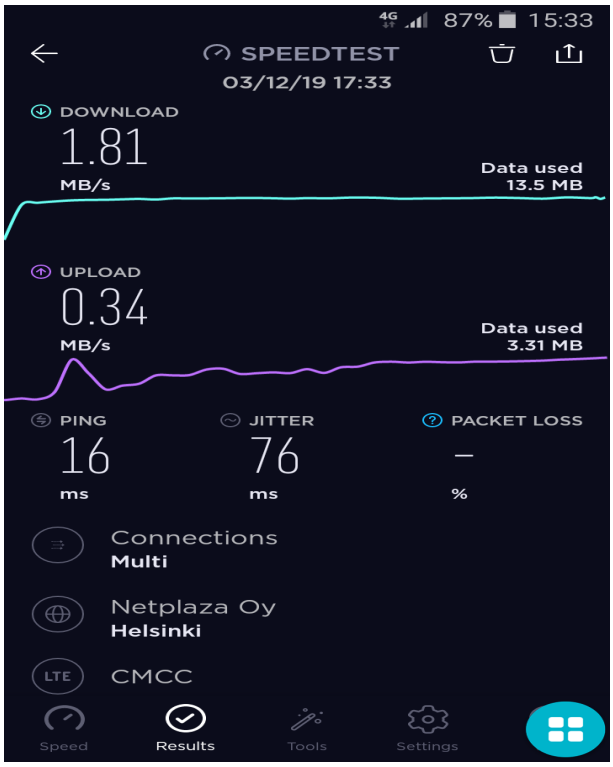

Figure 10: Test after some minutes of connection.

In addition, in this project the network has been tested with several common applications. Some of the most important test have been web searching, use of email sending messages as well as receiving emails and watching YouTube videos with good quality and good speed charging the video.

9 GTP tunnel modification

9.1 Modification Statement

Once the 4G network is working, the next steps focus in the modification of the GTP tunnel, in order to be able to manage it by the use of SDN. The main idea of the modification is the implementation of a module to switch between GTP and VLAN encapsulation. This new transport method will create different VLANs with different priorities. The packets will use the proper VLAN according to their QoS requirements.

The new module will have to perform all the authentication and routing processes that were performed by the old module. The TEID values coming from the GTP headers from the packets are now replaced by other authentication methods.

Once a new UE is registered in the system, the AMF from the EPC will send a UDP message to the MBO controller, which will forward the information to the proper module of the srsLTE software to process it. This UDP message will include the information of the user. This parameters are: IP address of the UE, TEID assign by the SMF and the VLAN ID that the UE will use.

This information is stored by the srsLTE. Then the program filters the incoming packets regarding the IP destination address. Using the previously received information, the software can map the value of the destination IP address to the proper TEID value of the connection. Therefore, the TEID value is still used in the software, but is only used internally for the authentication methods.

Separately, srsLTE will perform the creation of the VLANs itself. This is the reason why it needs to have the information about the VLAN ID, this parameters are key for the creation of the Ethernet header.

The following sections will explain in a more detailed form this modification.

9.2 Programming of the adjustments

The modification of the program code is a crucial part in this project. The completely understanding of the functioning of the code has huge importance. In this case, the code of the srsLTE software is programmed with C ++ but there are also some C files. For the purpose of the modification there are some key files in the software code:

```
srsenb/hdr/stack/upper/gtpu.h
srsenb/src/stack/upper/gtpu.cc
```

The most difficult part of this implementation will be the combination of these modules with the new adjustments for the perfect functioning and coordination between them.

9.2.1 Sockets

A socket is one endpoint of a communication with two ways between two programs running in the network [12]. There are three types of sockets, one is for the TCP protocol. In this sockets, it is not possible to send data until the connection is established successfully. These are used when the information send is important. This way the correct reception of the data is guaranteed. And the second one is for the UDP protocol. Any socket can send data anytime. This protocol does not guarantee that the data arrives or the correct order of the packets, but it guarantees that the received data is correct. So due to the requisites of this project, the second type of socket is the one used. The last ones are the raw sockets that allows sending and receiving of IP packets without any protocol-specific transport layer formatting. The huge inconvenient of these sockets is the need to create every header. The user will need to implement every header with the proper parameters and formats. This project use mainly raw socket. Raw sockets give freedom to customize the Ethernet headers in uplink, providing the possibility to create the VLAN header before sending the packets. In addition, it also allows the reception of the Ethernet header in downlink, making possible the parse of the destination IP header with the users in the system. One UDP socket is also used for receiving the JSON messages from the MBO. Of course, the program uses additional sockets for its payload but they are out of the scope of this project.

9.2.2 Management of the incoming data

As the previous reception focused in the GTP header, it was necessary a whole new reception module for this project. Previously, the software used a UDP socket for the GTP messages. From the packets, it parsed the value of the TEID and checked the existence of the user in the system. After that, the PDU was forwarded to the proper functions so the information could be sent to the UE.

As it has been explained before, the software is no longer receiving the GTP information in the packages. Therefore, it was necessary to think about a different approach for recognizing which of the UEs in the system was the target of each incoming message. For that reason the UDP socket has been substituted by a raw socket in reception. This change makes more difficult the packet filtering but allows the reception of packets destined to multiple IP addresses and the mapping of the proper TEID so the packets are forwarded to the proper UE.

9.2.3 Forwarding of the packets to the Gateway

The forwarding of the packets through the proper VLAN is a key part for achieving the desired QoS in each flux. The creation of every VLAN and the forwarding of the messages in the proper one will be performed by the srsLTE itself. It is crucial to use a raw socket in this function due to the need of creating customize header for each transmitted message. The sender socket receives the proper PDU to be forwarded, which includes the proper IP header. But the Ethernet header of the packet needs to be completely created by the socket.

The main parameters needed in the Ethernet header are the source MAC address, the destination MAC address and the protocol.

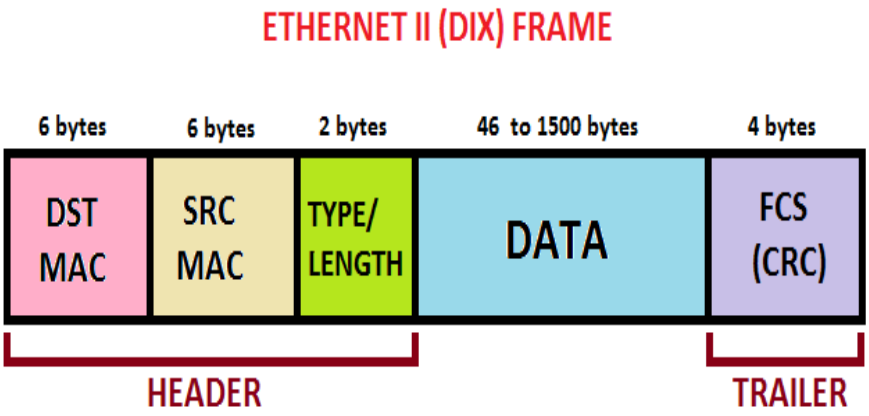

Figure 11: Ethernet header [13].

As the software does not use regular transmissions, it is necessary to create the VLAN version of the Ethernet header. In this version, the Ethernet header is 4 bytes longer. This 4 bytes, which store the information about the VLAN tag, will be placed between the source MAC address and the Ethernet type.

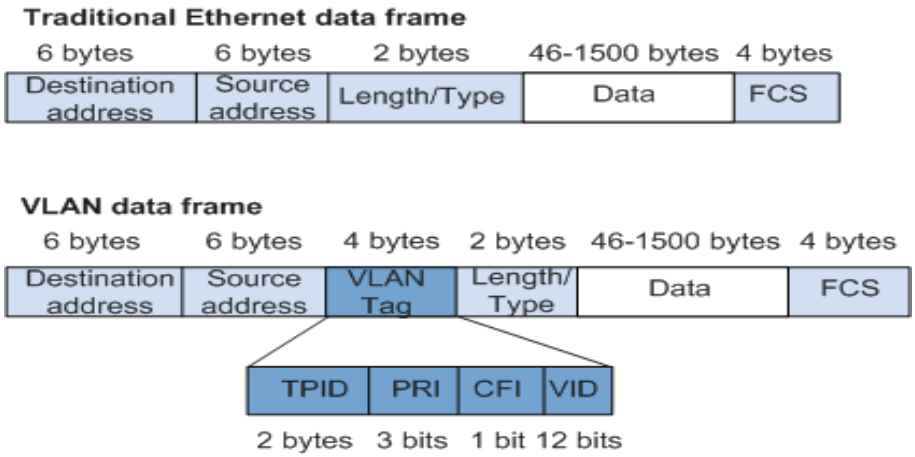

Figure 12: Ethernet header with VLAN Tag [14].

Nevertheless, the implementation of the Ethernet header is not an easy task. The obtaining of the destination MAC address requires an ARP request (Address Resolution Protocol). It is a protocol used to discover the link address, such as MAC address in this situation, associated to a given internet layer address (IP address) [15].

This project uses an ARP program which has as input parameter the interface used to send the ARP request and the IP address from which the MAC address is needed. As the destination IP address will be fixed, being the address of the Serving

Gateway, it does not imply any bigger difficulty than the integration of the ARP request program in the code for the Ethernet header creation. Furthermore, the code uses another program for extracting the Source MAC address of the own laptop and will use a fixed value of 0x0800 for the Ethernet protocol.

Finally, the only information left in the header is the VLAN tag. The value for the VLAN type will remain constant and equal to 0x8100 as only one tag is needed. The remaining information will have proper functions which, using the users database, will extract the values for the VLAN ID and thus priority through which each packet needs to be forwarded.

9.2.4 Users database

The user database provides the system with the tools to perform the end-to-end connections without the GTP tunnel. The database is an array with the information of the users in the system. Each user in the array have 3 fields of information. IP address of the UE, TEID assigned by the eNB and VLAN ID to forward the packet.

The information about the users is sent to the MBO controller, which after getting the proper information for the routing rules and the creation of the VLAN in downlink, will forward the message to the srsLTE module. The software will use a packet handler to process the messages once they are received. The module uses a mutex to block the use of the database while the new data is being included. Once the information is successfully added in the database the system will unblock the database to be used by other functions in the program and resume the proper functioning of the system.

In the reception the software will use the database to extract the TEID of the matching user regarding the destination IP of the packet. If the IP is not in the array the system will drop the packet. This is usual in the process due to the use of a raw socket for reception which can lead to the reception of undesired packets that were not intended to be received by srsLTE. For this reason, the connections in the reception interface have been limited as much as possible.

The MBO send a JSON message every time that the program is going to receive a message. Because of that reason, it was necessary to include a module to check if the information of the JSON message was already in the array. Notice the possibility that the TEID was already in the array but the information about the destination IP address or the VLAN ID has changed. In that case, the array must be updated. This possibility is very feasible since the QoS of the connections will be dynamically managed, changed then the VLAN ID of a connection when the quality is changed.

9.2.5 Open vSwitch

Open vSwitch is a switch platform that supports standard management interfaces and opens the forwarding functions to programmatic extension and control. The code is written in platform-independent C and is easily ported to other environments[16]. In combination with the OpenFlow orders or commands it allows to create an interface environment which can be dynamically managed according to the content of each packet using SDN.

OpenFlow is a communications protocol that gives access to the forwarding plane of a network switch or router over the network. OpenFlow enables to manage the traffic that is passing through the network. The controllers are different from the switches. This separation of the control from the user plane allows for more sophisticated traffic management than is feasible using access control lists (ACLs) and routing protocols [17]. Along the functions that this combination can perform there are:

- Creation of customized VLAN tags.
- Untag VLAN packets.
- Dynamic forwarding rules according to the destination IP.
- Dynamic forwarding rules according to the VLAN ID.

All these actions have a huge importance for the goal of this project.

In addition, in this project the Ryu controller will be used. Ryu Controller is an open, SDN Controller designed to increase the agility of the network by making it adapt how traffic is handled [18]. The code of this controller is programmed in Python and has been tested with OVS. The use of Ryu controller will allow the management of the OVS from the AMF.

The new transport method will use two different modules of OVS. One is placed after the srsLTE and the second one is placed with the Service Gateway. The OVS node will performed the actions that cannot be managed by the srsLTE software. Of course the nodes are continuously under the MBO management.

In uplink, the OVS node of the srsLTE part will only forward the packets that are passing through, as the sender raw socket has created the proper VLAN type Ethernet header. However, the OVS node of the SGW part will have a huge importance in the uplink process. This module will be in charge of popping the VLAN header, in other words, it will replace the VLAN type Ethernet header with a regular Ethernet header. Otherwise, after the forwarding of the packets performed by the gateway, the destination host will drop the packets.

In parallel the downlink process will be also running. Once the Internet host has sent the response to the UE it will first arrive to the gateway. The gateway will forward it to its OVS node, which taking into account the value of the destination IP will forward it through the proper VLAN. Once the packets are received in the srsLTE part, it is the MBO agent who pops the VLAN and forward it to the srsLTE program for the proper reception and following processes.

10 Final results

Once the new transport has been completely integrated with the old GTP transport, being able to the customer to choose between each of them, it is time to perform the final tests and get the final results.

The final test will consist in the comparison between the performance of the transports. In other words, the test will measure what is the difference between the throughput and the quality of the signal for each transport. For this reason, it is really important to know what is the basis of this test. This new transport will remove the GTP header, allowing a smaller packet size and then reducing the message overhead. But furthermore, the tagging of the different tunnels in the transport allows different routing and quality managing not only between VLANs and GTP tunnels but also between different VLANs. This last possibility is going to be tested.

The set up will include two cell phones, acting as User Equipment here. Each of the phones will use one of the transports used in this project. Thus one cell phone will be using a regular GTP connection, implemented already in srsLTE. And the other phone will be using the new method, then, the information of this transport will use VLANs. As the Software will only receive the JSON message for the second phone, this information will be the only in the user array previously explained. Thus, when the program starts running both of the phones will face the same criteria. Firstly, both phones will use the GTP transport method but after receiving the proper JSON message, the destination IP of the phones will be compared with the newly added information in the array. As the first phone does not coincide with the IP in the array, it will continue using the old GTP transport, but when the second phone IP is checked, it will coincide with the information in the array. At this point, the transport method for this UE will change to the new method.

The second phone will follow a different reception process. As the GTP reception process will fail it will use the new process. The raw socket extracts the destination IP of the messages, with this information the software can get the TEID from the users array. Dealing with the problem of not being able to extract the TEID from the GTP header. After this, the Ethernet header will be deleted from the message and the IP packet will be forwarded to the PDCP interface for its following transmission to the proper UE.

However, the most important part of this process and the main objective of the main objective of the project comes with the transmission of the packets received from the UE. After the proper VLAN encapsulation, it is possible to route the different transports according to different routing rules. Notice that this was not possible when it only existed one unique GTP tunnel. With different routing rules, different QoS can be achieved in each connection. Those are the causes of the results that are going to be shown in this section.

10.1 Connection of two UE with different transports

Firstly it is necessary to make sure that the connection is working with both of the cell phones. Once the srsenb is running, the phone using GTP was the one attached

and tested that it had a good connection quality. This test is performed with a simple Ping with the "Ping" app for cell phones, available in "Play Store" and also with the "Speed Test" app.

Once the connection is established, we disconnect the plane mode of the second device. This is the best practice for establishing the connection, as the phone will immediately try to connect to a network. When the connection is achieved the phone will start transmitting and receiving information and when the system receive the proper JSON message the transport will be switch to the new one.

This process is difficult to understand with images. However, since it is impossible to show a video of the process, some captures will be shown bellow.

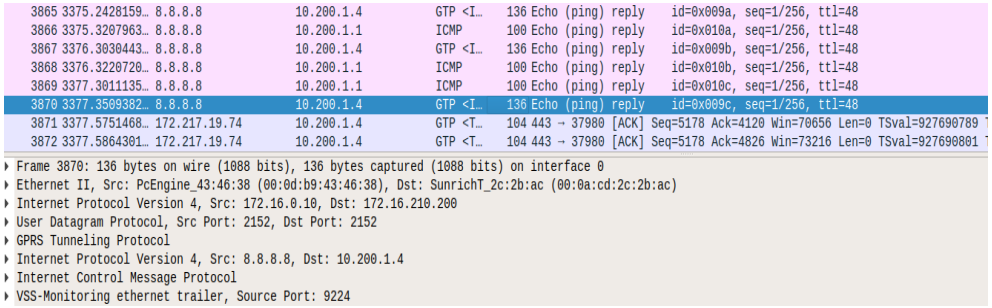

Figure 13: Wireshark capture with flows for both phones .

In order to extract the data about the performance of both of the methods, only one phone has been used to perform the final speed tests. This way the environment in which both transport is the same and away from tiny differences that the routes used might have, one can conclude that the differences seen are entirely cause of the new transport method.

In this test the phone will be connected normally to the network and once the connection is achieved the speed has been measured to know the base state of the system. Bellow, a screenshot of this situation is shown.

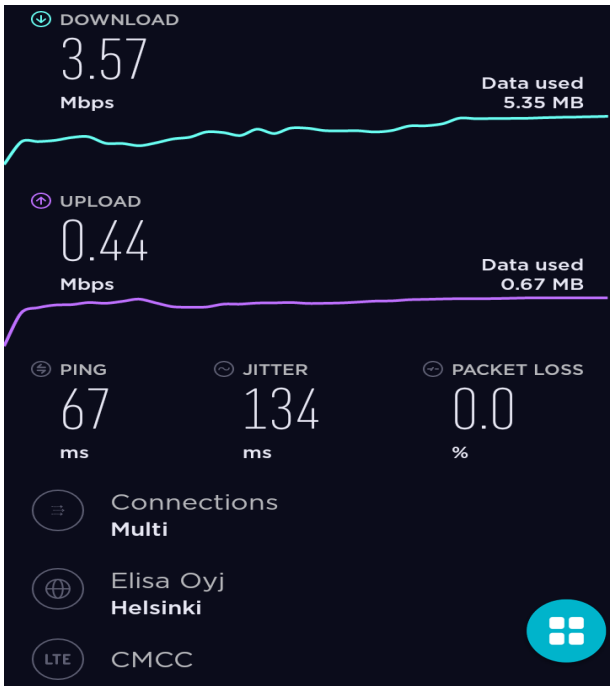

Figure 14: Speed test of the phone with GTP transport with no traffic.

After making sure what is the initial point of this test, some additional traffic was injected in the same interface or route. This simulates the situation when some congestion is found in the network. Taking into account the following capture, it can be clearly seen how the traffic has affected the speed of the connection.

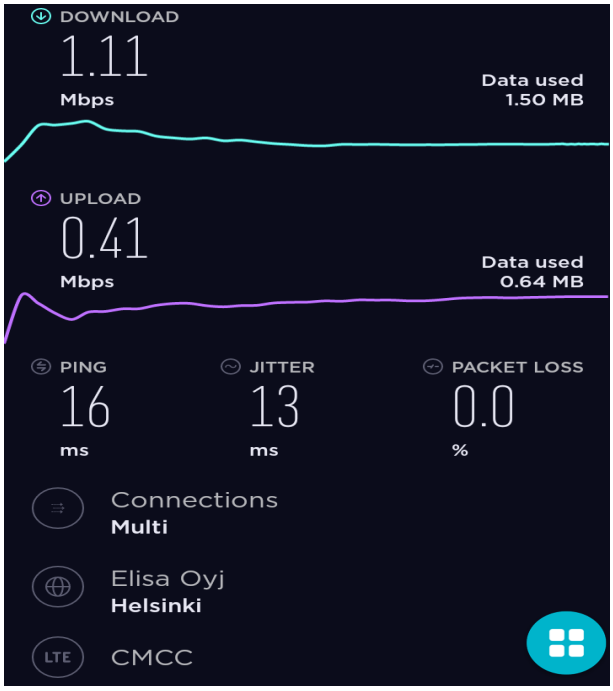

Figure 15: Speed test of the phone with GTP transport with added traffic.

This reduction in the throughput is the main reason why this project is necessary. After finishing this test the switch transport protocol process started. Once the traffic for this user was encapsulated in a VLAN, with its respective VLAN ID, the OVS was able to perform the differentiated routing. Thus, the traffic of the VLAN with VLAN ID equal to 2, is redirected to another route with less traffic. In this case, with no additional traffic at all. Then, the connection can use a higher bit rate. This is the final speed test of the phone using the alternative route.

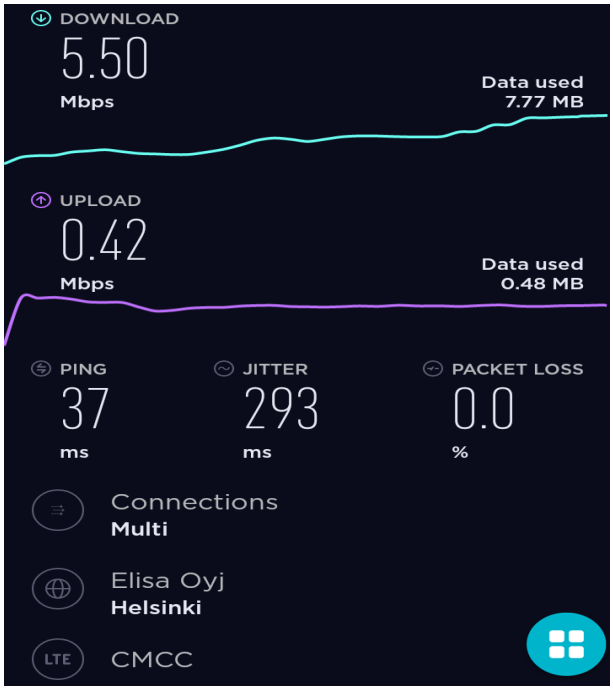

Figure 16: Speed test of the phone with VLAN transport in an additional route.

The change of route not only has recovered the quality of the connection, moreover it has improve it. This improvement is caused due to the reduction of the header size of each packet as it has been explained. The reduction of the overhead allows this increase in the downlink bit rate comparing with the initial test. Although, as the link condition are not fixed when connecting to the Internet, it is possible that this test shows a too optimistic result, the benefits of this system are clear.

11 Summary

In conclusion, building an SDR 4G network is a long and hard task that might have a lot of problems in the process, but taking into account the test done before the addition of the new code and the speed of the connection with the initial software, one can say that it was a success.

The software modification was also a big difficulty in this project, as the project was faced with no knowledge in neither C++ or C programming. The understanding of the existing code and the creation and combination of the new code with the existing one was the biggest challenge of this project.

Since the initial idea of this project was the completely substitution of the GTP tunnel for the new one. At some point the VLAN tunnel was the only transport method of the system. The speed test was also performed with that code and the system achieved a good link quality. However, before the reception of the JSON message in the system, the user is connected to the network but the exchange of messages or information is not available. This problem have been solved with the initial and default GTP connection in the code.

Regarding the final testing, with the data extracted from the final results and the performance of both phones connected at the same time, one can say that all the tests have been passed. This project has shown what can be the consequences derived from the addition of this new method to the existing mobile backhaul of the incoming 5G systems.

Apart from the obvious benefits in the bit rate of the connection, the main benefit of this new transport resides in the addition of an SDN manager. This attribute provides to the system enough tools and capabilities to perform a centralized management of the system leading to an optimized transport in the 5G networks where, as it has been explained in the introduction, the resource use and managements of each user is really important. The addition of the benefits will reduce the impact of the massive increase of users in the networks and the resource allocation for each user, which is one of the biggest concerns of the industry about the incoming system. The new transport, as it has been proved in this project can be first implemented is 4G eNB, thus the implementation will be added before all the changes that this new era of the technology will bring.

12 Future research

This project has only been a first contact with the new transport implementation and benefits. As it happens in all the parts of the industry, the implementation can be improved and thus achieve a better performance with the addition of systems and modules that increase the performance.

One of the big benefits that are not tested in this project is the influence of the assignment of different priorities in the "VLAN priority tag". This attribute can give additional tools for the management of the system and consequently to achieve the respective quality of service that must be ensured for each user. The user can have a bigger distinction from the other user regarding the quality of their link. Therefore, apart from the redirection of the users through a different route, some users might have priority over other users with a higher priority value in the VLAN tag. Regarding, the code that is already implemented, it would not be too difficult to include this function in the existing code. Since the creation of the packets use raw sockets, it will be enough with passing the desired value of the VLAN priority to the existing module. Previously, the AMF would have sent the priority value included in the JSON message and will be included in the user array, for the following extraction of information depending on the IP address.

Finally, as it has been explained many times in this document, the hardware used in this project can be improved if bigger funding is available for the project. The enhancements can come from the use of a laptop with higher capabilities, which has been without doubts, the limiting part of the implementation and testing of the system of the project. The increasing performance that would be achieve with the use of a computer with bigger CPU will bring a huge increment in the stability of the system and allowing the use of high performance cell phones and thus a huge increment in the bit rate of the connection in both downlink and uplink.

References

[1] Number of smartphone users worldwide from 2016 to 2021 *Statista* https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

[2] Software Defined Networking *Wikipedia* https://en.wikipedia.org/wiki/Software-defined_networking

[3] Network Function Virtualization *Wikipedia* https://en.wikipedia.org/wiki/Network_function_virtualization

[4] Network Slicing *Wikipedia* https://en.wikipedia.org/wiki/5G_network_slicing#/media/File:Generic_5G_network_slicing_framework.svg

[5] What is Non Orthogonal Multiple Access *Research gate* https://www.researchgate.net/post/What_is_Non_Orthogonal_Multiple_Access_NOMA_for_wireless_communications

[6] The difference between OMA and NOMA https://www.researchgate.net/figure/The-difference-between-orthogonal-multiple-access-OMA-and-non-orthogonal_fig2_320680888

[7] UHD Documentation *Ettus Research* https://github.com/EttusResearch/uhd

[8] Git *Wikipedia* https://en.wikipedia.org/wiki/Git

[9] Building and Installing the USRP Open-Source Toolchain (UHD and GNU Radio) on Linux *Ettus Research* https://kb.ettus.com/Building_and_Installing_the_USRP_Open-Source_Toolchain_(UHD_and_GNU_Radio)_on_Linux

[10] About GNU Radio *GNU Radio* https://www.gnuradio.org/about/

[11] srsLTE Documentation, 2019 https://docs.srslte.com/en/latest/

[12] Network Sockets *Wikipedia* https://en.wikipedia.org/wiki/Network_socket

[13] Ethernet header picture *Bit forest info* https://www.bitforestinfo.com/2018/01/code-ethernet-ii-raw-packet-in-python.html

[14] VLAN header picture *Huawei* https://support.huawei.com/enterprise/en/doc/EDOC1100088104

[15] Address Resolution Protocol *Wikipedia* https://en.wikipedia.org/wiki/Address_Resolution_Protocol

[16] What Is Open vSwitch? *Open vSwitch* http://docs.openvswitch.org/en/latest/intro/what-is-ovs/

[17] OpenFlow *Wikipedia* https://en.wikipedia.org/wiki/OpenFlow

[18] What Is Ryu Controller? *SDX central* https://www.sdxcentral.com/networking/sdn/definitions/what-is-ryu-controller/

A Appendix: Additional commands

```
$ sudo apt-get --download-only install git swig cmake
doxygen build- essential libboost-all-dev libtool
libusb-1.0-0  libusb-1.0-0-dev libudev- dev libncurses5-dev
libfftw3-bin libfftw3-dev libfftw3-doc libcppunit- 1.14-0
libcppunit-dev libcppunit-doc ncurses-bin cpufrequtils
python-numpy python-numpy-doc python-numpy-dbg python-scipy
python-docutils qt4-bin-dbg qt4-default qt4-doc libqt4-dev
libqt4-dev-bin python-qt4 python-qt4-dbg python-qt4-dev
python-qt4-doc python-qt4-doc libqwt6abi1 libfftw3-bin
libfftw3-dev libfftw3-doc ncurses-bin libncurses5
libncurses5-dev libncurses5-dbg libfontconfig1-dev
libxrender-dev libpulse-dev swig g++ automake autoconf
libtool python-dev libfftw3-dev libcppunit-dev
libboost-all-dev libusb-dev libusb-1.0-0-dev fort77
libsdl1.2-dev python-wxgtk3.0 git libqt4-dev python-numpy
ccache python-opengl libgsl-dev python-cheetah python-mako
python-lxml doxygen qt4-default qt4-dev-tools
libusb-1.0-0-dev libqwtplot3d-qt5-dev pyqt4-dev-tools
python-qwt5-qt4 cmake git wget libxi-dev
gtk2-engines-pixbuf r-base-dev python-tk liborc-0.4-0
liborc-0.4-dev libasound2-dev python-gtk2 libzmq3-dev
libzmq5 python-requests python-sphinx libcomedi-dev
python-zmq libqwt-dev libqwt6abi1 python-six libgps-dev
libgps23 gpsd gpsd-clients python-gps python-setuptools
screen sshfs
```

B Appendix: Configuration files

B.1 enb.conf file

```
                    srsENB configuration file


eNB configuration

enb_id:          20-bit eNB identifier.
cell_id:         8-bit cell identifier.
tac:             16-bit Tracking Area Code.
mcc:             Mobile Country Code
mnc:             Mobile Network Code
mme_addr:        IP address of MME for S1
connection
gtp_bind_addr:   Local IP address to bind for GTP
connection
s1c_bind_addr:   Local IP address to bind for S1AP
connection
n_prb:           Number of Physical Resource Blocks
(6,15,25,50,75,100)
tm:              Transmission mode 1-4 (TM1 default)
nof_ports:       Number of Tx ports
(1 port default, set to 2 for TM2/3/4)


[enb]
enb_id = 0x19B
cell_id = 0x01
phy_cell_id = 1
tac = 12594
mcc = 244
mnc = 52
mme_addr = 172.16.0.10
gtp_bind_addr = 172.16.210.200
s1c_bind_addr = 172.16.210.200
n_prb = 25
-tm = 4
-nof_ports = 2


eNB configuration files
```

```
-
- sib_config:  SIB1, SIB2 and SIB3 configuration file
- note: when enabling mbms, use the sib.conf.mbsfn
configuration file which includes SIB13
- rr_config:   Radio Resources configuration file
- drb_config:  DRB configuration file
```

```
[enb_files]
sib_config = sib.conf
rr_config  = rr.conf
drb_config = drb.conf
```

```
[rf]
dl_earfcn = 1200
tx_gain = 80
rx_gain = 40

-device_name = auto


-device_args = auto
-time_adv_nsamples = auto
-burst_preamble_us = auto
```

```
[pcap]
enable = false
filename = /tmp/enb.pcap

[log]
all_level = warning
all_hex_limit = 32
filename = /tmp/enb.log
file_max_size = -1

[gui]
enable = false
```

```
- Scheduler configuration options
-
- pdsch_mcs:          Optional fixed PDSCH MCS
(ignores reported CQIs if specified)
```

```
- pdsch_max_mcs:      Optional PDSCH MCS limit
- pusch_mcs:          Optional fixed PUSCH MCS
(ignores reported CQIs if specified)
- pusch_max_mcs:      Optional PUSCH MCS limit
- -nof_ctrl_symbols: Number of control symbols
-
```

```
[scheduler]
-pdsch_mcs          = -1
-pdsch_max_mcs      = -1
-pusch_mcs          = -1
pusch_max_mcs      = 16
nof_ctrl_symbols = 3
```

B.2 rr.conf file

```
mac_cnfg =
{
  phr_cnfg =
  {
    dl_pathloss_change = "dB3"; // Valid:
    1, 3, 6 or INFINITY
    periodic_phr_timer = 50;
    prohibit_phr_timer = 0;
  };
  ulsch_cnfg =
  {
    max_harq_tx = 4;
    periodic_bsr_timer = 20; // in ms
    retx_bsr_timer = 320;    // in ms
  };

  time_alignment_timer = -1; // -1 is infinity
};

phy_cnfg =
{
  phich_cnfg =
  {
    duration  = "Normal";
    resources = "1/6";
  };

  pusch_cnfg_ded =
  {
    beta_offset_ack_idx = 6;
    beta_offset_ri_idx  = 6;
    beta_offset_cqi_idx = 6;
  };

  // PUCCH-SR resources are scheduled on time-frequeny
  domain first, then multiplexed in the same resource.
  sched_request_cnfg =
  {
    dsr_trans_max = 64;
    period = 20;        // in ms
    subframe = [1];     // vector of subframe indices
    allowed for SR transmissions
    nof_prb = 2;        // number of PRBs on each extreme
```

```
    used for SR (total prb is twice this number)
  };
  cqi_report_cnfg =
  {
    mode = "periodic";
    simultaneousAckCQI = true;
    period = 40; // in ms
    subframe = [0];
    nof_prb = 2;
    m_ri = 8; // RI period in CQI period
  };
};
```

B.3 sib.conf file

```
sib1 =
{
    intra_freq_reselection = "Allowed";
    q_rx_lev_min = -65;
    //p_max = 3;
    cell_barred = "NotBarred"
    si_window_length = 20;
    sched_info =
    (
        {
            si_periodicity = 16;
            si_mapping_info = []; // comma-separated
            array of SIB-indexes (from 3 to 13).
            // Leave empty or commented to just
            scheduler sib2
        }
    );
    system_info_value_tag = 0;
};

sib2 =
{
    rr_config_common_sib =
    {
        rach_cnfg =
        {
            num_ra_preambles = 52;
            preamble_init_rx_target_pwr = -104;
            pwr_ramping_step = 6;  // in dB
            preamble_trans_max = 10;
            ra_resp_win_size = 10;  // in ms
            mac_con_res_timer = 64; // in ms
            max_harq_msg3_tx = 4;
        };
        bcch_cnfg =
        {
            modification_period_coeff = 16; // in ms
        };
        pcch_cnfg =
        {
            default_paging_cycle = 32; // in rf
            nB = "1";
        };
```

```
prach_cnfg =
{
    root_sequence_index = 128;
    prach_cnfg_info =
    {
        high_speed_flag = false;
        prach_config_index = 3;
        prach_freq_offset = 2;
        zero_correlation_zone_config = 5;
    };
};
pdsch_cnfg =
{
    /* Warning: Currently disabled and forced to
    p_b=1 for TM2/3/4 and p_b=0 for TM1
     */
    p_b = 1;
    rs_power = 0;
};

pusch_cnfg =
{
    n_sb = 1;
    hopping_mode = "inter-subframe";
    pusch_hopping_offset = 2;
    enable_64_qam = false; // 64QAM PUSCH is not
    currently enabled
    ul_rs =
    {
        cyclic_shift = 0;
        group_assignment_pusch = 0;
        group_hopping_enabled = false;
        sequence_hopping_enabled = false;
    };
};

pucch_cnfg =
{
    delta_pucch_shift = 2;
    n_rb_cqi = 2;
    n_cs_an = 0;
    n1_pucch_an = 12;
};
ul_pwr_ctrl =
{
```

```
            p0_nominal_pusch = -85;
            alpha = 0.7;
            p0_nominal_pucch = -107;
            delta_flist_pucch =
            {
                format_1  = 0;
                format_1b = 3;
                format_2  = 1;
                format_2a = 2;
                format_2b = 2;
            };
            delta_preamble_msg3 = 6;
        };
        ul_cp_length = "len1";
    };

    ue_timers_and_constants =
    {
        t300 = 2000; // in ms
        t301 = 100;  // in ms
        t310 = 1000; // in ms
        n310 = 1;
        t311 = 1000; // in ms
        n311 = 1;
    };

    freqInfo =
    {
        ul_carrier_freq_present = true;
        ul_bw_present = true;
        additional_spectrum_emission = 1;
    };

    time_alignment_timer = "INFINITY"; // use "sf500",
    "sf750", etc.
};
```

B.4 drb.conf file

```
// All times are in ms. Use -1 for infinity, where available

qci_config = (

{
  qci=7;
  pdcp_config = {
    discard_timer = 100;
    pdcp_sn_size = 12;
  }
  rlc_config = {
    ul_um = {
      sn_field_length = 10;
    };
    dl_um = {
      sn_field_length = 10;
      t_reordering    = 45;
    };
  };
  logical_channel_config = {
    priority = 13;
    prioritized_bit_rate   = -1;
    bucket_size_duration  = 100;
    log_chan_group = 2;
  };
},
{
  qci=9;
  pdcp_config = {
    discard_timer = -1;
    status_report_required = true;
  }
  rlc_config = {
    ul_am = {
      t_poll_retx = 120;
      poll_pdu = 64;
      poll_byte = 750;
      max_retx_thresh = 16;
    };
    dl_am = {
      t_reordering = 50;
      t_status_prohibit = 50;
```

```
      };
    };
    logical_channel_config = {
      priority = 11;
      prioritized_bit_rate    = -1;
      bucket_size_duration  = 100;
      log_chan_group = 3;
    };
}

);
```