FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study programme/specialisation: Computer Science	Spring / ~~Autumn~~ semester, 2019 Open/~~Confidential~~
Author: Mari Håland	Mari Håland ..... (signature of author)
Programme coordinator: Gianfranco Nencioni Supervisor(s): Gianfranco Nencioni and Rosario G. Garroppo	
Title of master's thesis: Evaluation of low-cost software-defined LTE/5G frameworks: cloudification	
Credits: 30	
Keywords: Mobile communication, Cloudifcation, OpenAirInterface, LTE/5G, Emulation/Simulation	Number of pages: ..42..... + supplemental material/other: Appendix A Stavanger,..13.06.2019..... date/year

University of Stavanger

Evaluation of low-cost software-defined LTE/5G frameworks: cloudification

Author:
Mari Håland

Supervisors:
Gianfranco Nencioni
Rosario G. Garroppo

June 13, 2019

Abstract

There has been a tremendous rise in mobile communication units over the past decade. Where cellphones once dominated the user market for fast wireless communication solutions, we now see an abundance of devices and appliances being enabled with cellular communication technology to enable high-speed connectivity.

Machine to Machine (M2M) communication services entered the market during the late 2000s with the rise of 3G mobile communication, paving the way for a more generalized approach to mobile communication services that was targeted at industrial instrumentation and automation, without the need for proprietary intermediate interfaces and hub-and-spoke architecture.

Internet of Things (IoT) describes the most recent trend in mobile connectivity, where anything from household appliances to micro-controllers for street lights are enabled with cellular communication technology to enable management, reporting, remote controllability, surveillance and other features. The recent mobile technologies like 4G/LTE enables high-speed, low-latency wireless communication over vast distances, in harsh RF environments, and eliminates the need for cabling and wires between endpoints.

The next evolution in mobile networks comes with 5G, where the key successes from 4G/LTE is evolved to further improve and strengthen the mobility services for both consumer, business and industry segments. 'More speed', 'higher availability' and 'enhanced subscriber management' are terms we know from the vendor sales pitch. The technical side is far more complex, and reveals a subset of technologies that goes deeper into the IoT and M2M branches of mobile communication.

This thesis will focus on specific aspects of both 4G/LTE and 5G technologies, looking into how the base stations and radio core elements interact, and respond to changes in the network. With server virtualization and software enablement of the mobile core, components that were previously hosted in-house can now be moved into the cloud. How does this affect performance? What are the consequences of these changes from a Quality of Experience (QoE) perspective? These are questions I will look into in further detail in this thesis.

Acknowledgements

I would like to thank my supervisors, Professor Gianfranco Nencioni and Rosario G. Garroppo, for their guidance and feedback throughout my work on this thesis.

I would also like to express my gratitude toward my coworker Tord Førland at Tampnet for all the help and advice throughout my work on this thesis.

Contents

List of Abbreviations

Mbps	Megabits per second
Kbps	Kilobits per second
SDN	Software Defined Networking
NFV	Network Function Virtualization
uRLLC	Ultra-Reliable Low-Latency Communication
eMBB	Enhanced Mobile Broadband
mMTC	Massive Machine-Type Communication
MEC	Multi-access Edge Computing
MCN	Mobile Core Network
OAI	OpenAirInterface
VPN	Virtual Private Network
UE	User Equipment
LTE	Long Term Evolution
CN	Core Network
EPC	Evolved Packet Core
eNB	Evolved NodeB
MME	Mobility Management Entity
HSS	Home Subscriber Server
S-GW	Serving Gateway
PDN	Packet Data Network

P-GW	PDN Gateway
PCRF	Policy Control and Charging Rules Functions
IMSI	International Mobile Subscriber Identity
APN	Access Point Name
IMEI	International Mobile Equipment Identity
S1-C	S1-Control interface
S1-U	S1-User interface
UMTS	Universal Mobile Telecommunications System
E-UTRAN	Evolved UMTS Terrestrial Radio Access Network
AMF	Access and Mobility Management Function
SMF	Session Management Function
AUSF	Authentication Server Function
NAS	Non-Access Stratum
UDM	Unified Data Management
UPF	User Plane Function
QoS	Quality of Service
UDSF	Unstructured Data Storage Function
NF	Network Function
PCF	Policy Control Function
AF	Application Function
HTTP	Hypertext Transfer Protocol
XML	Extensible Markup Language
NSSF	Network Slice Selection Function
NR	New Radio
SDR	Software-Defined Radio
VM	Virtual Machine
SLA	Service Level Agreement

ETSI	European Telecommunications Standards Institute
AR	Augmented Reality
VR	Virtual Reality
RAN	Radio Access Network
SRS	Software Radio Systems
RRC	Radio Resource Control
PDCP	Packet Data Convergence Protocol
RLC	Radio Link Control
MAC	Medium Access Control
PHY	Physical
IP	Internet Protocol
PLMN	Public Land Mobile Network
MCC	Mobile Country Code
MNC	Mobile Network Code
API	Application programming interface
nFAPI	Network Functional API
PNF	Physical Network Function
VNF	Virtual Network Function
MTU	Maximum Transmission Unit
DHCP	Dynamic Host Configuration Protocol
NAT	Network Adders Translation
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
SNR	Signal-to-Noise Ratio
RSRP	Reference Signal Received Power
SCTP	Stream Control Transmission Protocol
S1AP	S1 Application Protocol

EPS	Evolved Packet System
UL	Uplink
DL	Downlink
GUI	Graphical User Interface
NTP	Network Time Protocol
RTP	Real-time Transport Protocol
RTPS	Real-Time Publish-Subscribe
SIM	Subscriber Identity Module3GPP
QoE	Quality of Experience
3GPP	3rd Generation Partnership Program

Chapter 1

Introduction

This thesis will focus on the backend systems and solutions for 4G/LTE and 5G networks, investigating the latest generations of Software Defined Radio (SDR) and radio core virtualization and simulation software. The thesis will also look into functions for network virtualization and slicing, for the purpose of segmentation and network segregation.

Traditional mobile networks has been composed of various appliances, vendor-specific hardware units, and server components that interact to form the mobile core network. As the digital era has moved towards virtualization and software enablement of features and functions that was previously only available in appliances – the mobile core network and its components have been modernized to fit into this stage of the software evolution. Different tools for mobile core network virtualization and emulation has been tested as part of this thesis, giving the ability to create a virtual mobile core- and access network to emulate a virtual operator scenario.

As the thesis focuses on software defined services and virtualization, a natural aspect of these elements would be to investigate how a radio access network (RAN) and mobile core network (MCN) responds to a change in topology, where the core elements are moved from in-house to external hosting (i.e. the 'cloud').

What are the impacts of increased network distance between base station and mobile core? How will increased latency and network distance metric impact the experienced quality of services? Which methods can be utilized to measure these attributes? These are questions the thesis will try to answer, both from a quantitative approach through measurements, and a QoE approach through interpretation of the results. The thesis will summarize all findings and present conclusions for the lab trials, as well as presenting some own thoughts and views from the author.

1.1 Contributions and Outline

The following contributions are made in this thesis:

- Experiment with OpenAirInterface's (OAI) open-source emulation platform for LTE/5G network.
- Implemented an OAI core network in a cloud environment and set up a VPN tunnel to the cloud instance.
- Implemented an OAI core network in an Intel NUC machine and in a virtual machine, simulating a cloud environment.
- Implemented an OAI simulated base station and user equipment (UE).
- Connected up to 10 simulated UEs to the network and verified Internet connectivity for all of them.
- Connected the base station to two core network simultaneously.
- Evaluated the performance of the network when using the simulated cloud and MEC core networks, in terms of latency, jitter, and packet loss. Compared the performance of having only one UE connected with having 10 UEs connected.
- Evaluated the experience of the user equipment using video and audio streaming.
- Compare the performance of the test-bed with a production LTE network.

The remainder of this thesis is outlined as follows:

Chapter 2: Gives an overview of the 4G and 5G system, looking at how the 5G infrastructure will differ from 4G and key elements of the 5G are presented; including virtualization, softwarization, and network slicing.

Chapter 3: Elaborates on the test procedure of test setup in this thesis, mostly at a lab environment at the University of Stavanger but also including setup of an industrial 4G system at Tampnet's office in Stavanger.

Chapter 4: Presents the evaluation and tests conducted on both test-beds.

Chapter 5: Concludes the work of this thesis and present ideas for future work.

Chapter 2

Background

The road to 4G/LTE started in the 80s with the launch of GSM, and later EDGE and GPRS. Finally, the industry had a standardized way of enabling packet transport over the mobile carriers. This evolution, driven by the 3rd Generation Partnership Program (3GGP), later led to what would become the Third-Generation mobile network technology, 3G. With 3Gs many advancements during its development lifecycle, what started out as 3G ended up in 3.9G and HSPA+ for the 3GGP release 7. Release 7 marked the end of the 3G development cycle, and the first LTE specifications were announced with the release of 3GGP Release 8 [1].

This chapter will focus on the 4G/LTE and 5G network architecture and mobile core components. A presentation of the core network and its component will be followed by the access network architecture. The similarities and differences between the architectures and their respective components will also be illustrated, in order to easier map function and placement in the network.

After covering 4G/LTE and 5G, attention will shift to the virtualization and software enablement. Concepts like virtualized mobile core, software-enabled network functions and Multi-access Edge Computing (MEC) will be presented and discussed, referencing related works from other parties.

2.1 LTE architecture

Long Term Evolution (LTE) is the 4th evolutionary step in mobile communication networks, often referred to as 4G. The technology consist of several central- and decentralized components, covering both radio, packet transport and service awareness. This chapter will detail some of LTEs most common components and their functions.

2.1.1 Evolved Packet Core Architecture

The Evolved Packet Core (EPC) consists of 5 main components; the Mobility Management Entity (MME), Home Subscriber Server (HSS), Serving Gateway (S-GW), Packet Data Network (PDN) Gateway (P-GW) and Policy Control and Charging Rules Function (PCRF).

The HSS is a database containing subscriber information; like information like International Mobile Subscriber Identity (IMSI) number of the SIM cards, Access Point Name (APN) information, and International Mobile Equipment Identity (IMEI) number. The Home Subscriber Server (HSS) is used by the Mobility Management Entity (MME) to verify that the users connecting are allowed on the network; this is not new to LTE; it is functionality continued from UMTS and GSM. The MME controls high-level operations of the EPC; meaning that it is responsible for mobility and session management. As figure 2.1 depicts, the MME communicates with both the E-UTRAN and the HSS for this. The MME communicates with the E-UTRAN via the S1-Control (S1-C) interface, called S1-MME in Figure 2.1, the eNB uses this interface when communicating with the MME, for instance, to verify that devices connecting are allowed on the network, as mentioned above [2].

The Packet Data Network (PDN) Gateway (P-GW) allows communication with the outside world via the SGi interface as depicted in figure 2.1. While the Serving Gateway (S-GW) acts just like a router, it forwards data packets between the base stations and the PDN. The S1-User (S1-U) interface is, like the S1-MME/C interface used for communication with the E-UTRAN, the S1-U interface is used to transfer traffic sent to and from the connected UEs.

The last part of the EPC is the Policy Control and Charging Rules Function (PCRF); this resides in the P-GW and is responsible for functionalities for flow-based charging and policy control decision-making [2].

2.1.2 E-UTRAN

The Evolved UMTS Terrestrial Radio Access Network (E-UTRAN) consists of one component, the Evolved nodeB (eNodeB/eNB). It is responsible for the radio communication between the user equipment (UE) and the EPC. UEs can be any device connected to the mobile network, like a phone, a tablet, or a laptop with a cellular interface. The eNB can contain one or more cells, the cells are either at different frequencies or cover different sectors, i.e., by having multiple antennas pointing in different directions. The eNB is responsible for the communication to and from the connected UEs. It sends and receives radio transmission to the UEs via the LTE radio interface. The eNB also controls

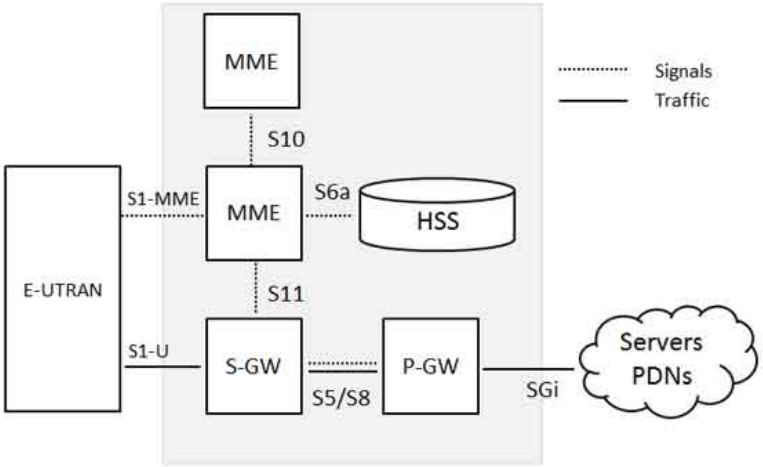


Figure 2.1: EPC [2]

low-level operations of all connected UEs via signaling traffic. There are usually more than one eNB in a network, and they communicate with each other via the X2 interface, depicted in Figure 2.2. This interface is mainly used for signaling and packet forwarding during handover, i.e., when a UE moves from one eNB to another, because the UE changes its position and are thereby getting better signal levels from another eNB.

Figure 2.2 illustrates the S1 interface from the eNB to the EPC. As explained in the previous section is the S1 interface divided into two interfaces, the S1-C and S1-U, shown with different lines in the figure. As both of these interfaces connects to one module in the E-UTRAN, and not different modules as in the EPC are they usually described as just the S1 interface [2].

2.2 5G architecture

5G is the fifth evolution of mobile networks, released by 3GGP in Release 15. While the technology and its components are still mostly unreleased to the mass market, and no large-scale deployment has been done outside of controlled lab environments; 5G has been big news around the world both because of its technology, and as key-component and leverage technology in world business and trade politics. 5G builds on the most successful components of 4G/LTE, and presents further enhancements towards higher capacity and lower latency networking [1, 3, 4].

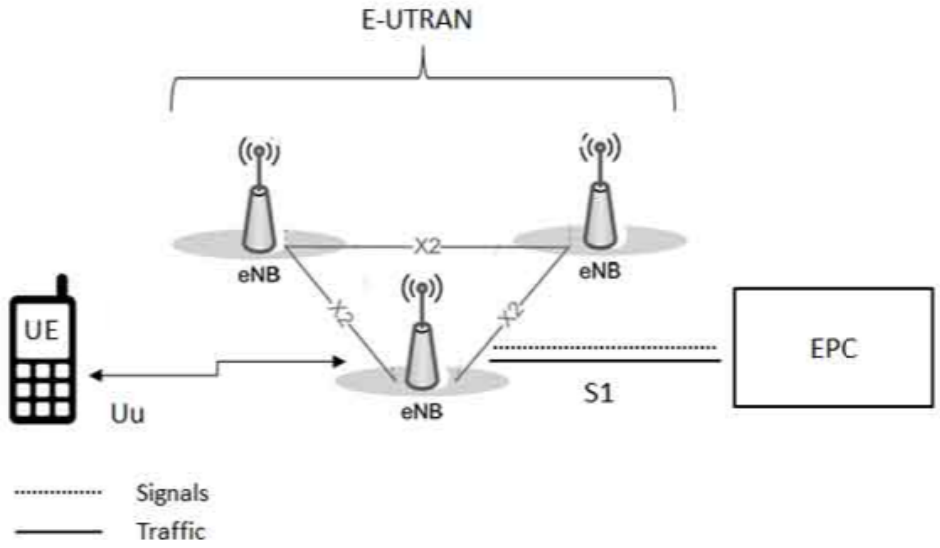


Figure 2.2: eNodeB [2]

2.2.1 5G Core

Figures 2.3, 2.4, and 2.5 depict a comparison of which components of the new 5G Core (5GC) that will replace the different components in the 4G EPC. Figure 2.3 displays the 4G MME, and show how the functionality of the MME will be split into three parts in the 5GC, Access and Mobility Management Function (AMF), Session Management Function (SMF) and Authentication Server Function (AUSF). The AMF is responsible for paging, mobility management, and the connection with the UEs, including maintaining NAS signaling and the registration procedure of the UEs.

The AUSF manages authentication of the connecting devices, and similar to the MME it checks with the database when a UEs is connecting. In 4G the MME checks with the HSS while in 5G the AUSF checks with the Unified Data Management (UDM).

The SMF is taking over some functionalities for both the MME and the S-GW and P-GW as illustrated in figure 2.4. It provides session management, which earlier was a function in the MME, as well as taking over the control plane functionalities for the S-GW and P-GW by allocating IP addresses for the UEs. The user plane functionalities of the S-GW and P-GW is in 5GC performed by the User Plane Function (UPF); this includes anchoring the UE IP addresses, traffic transportation, both uplink, and downlink as well as Quality of Service (QoS) enforcement [5].

As previously mentioned is the EPC HSS replaced with the UDM. The UDM stores the same kind of information as the HSS like subscriber information and as depicted in

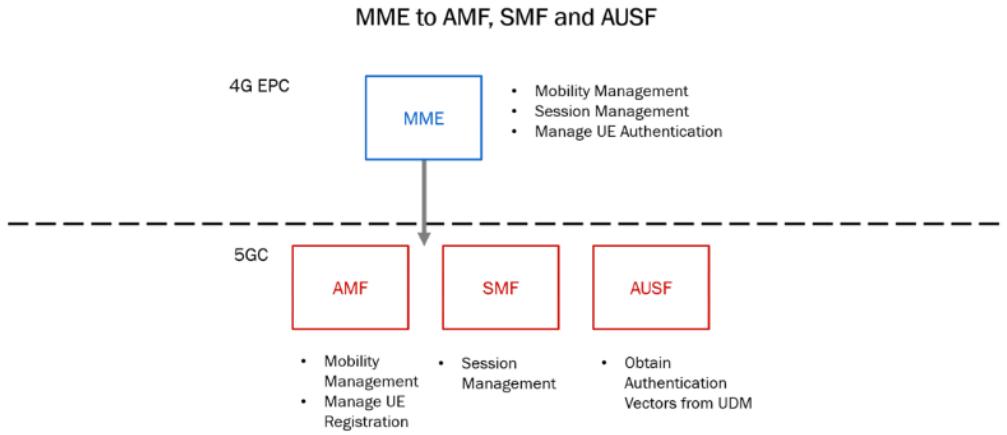


Figure 2.3: 4G MME vs. 5GC [5]

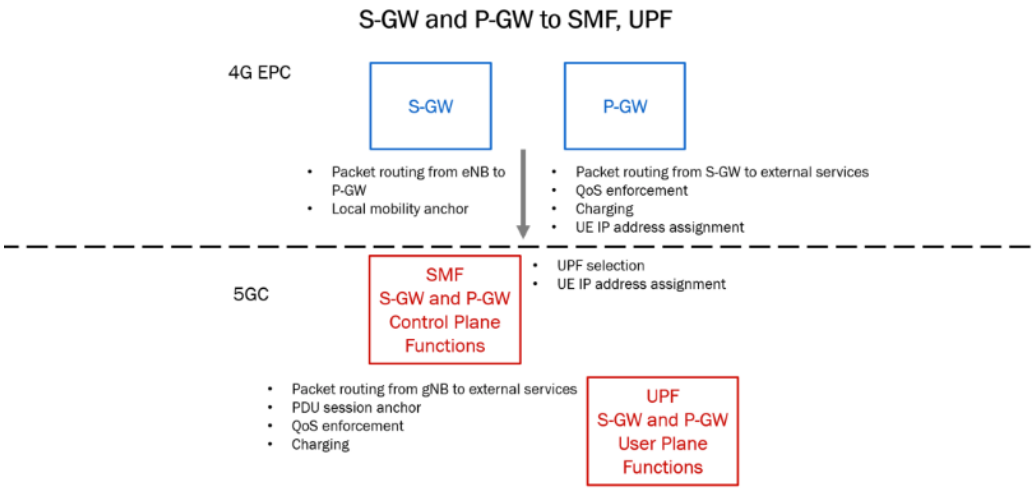


Figure 2.4: 4G SPGW vs. 5GC [5]

Figure 2.5, it communicates and makes this information available for both the AMF and the AUSF. The Unstructured Data Storage Function (UDSF) is part of the UDM, and it is used by the Network Functions (NFs) to store and retrieve unstructured data, and as Figure 2.5 depicts, it communicates with the Policy Control Function (PCF) [6, 5].

The PCF in 5G replaces the 4G PCRF. The PCF provides, in addition to the functionality of the PCRF, even more, capabilities. It allows for resource request reservation from the Application Function (AF) using an HTTP/XML-based Application Programming Interface (API) from other services which allow other, non-operators, services to provide QoS control to their subscribers [5].

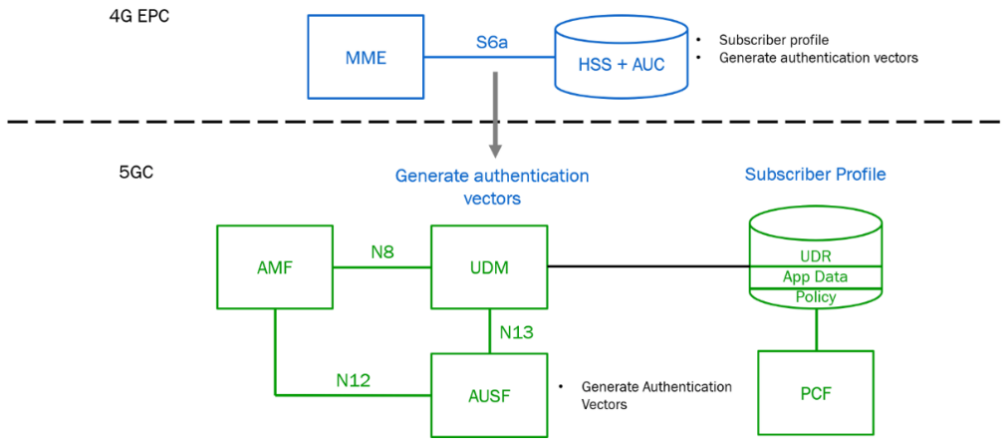


Figure 2.5: 4G EPC vs. 5GC [5]

Network slicing is one of the leading new capabilities of 5G, and the Network Slice Selection Function (NSSF) maintains the list of slices; this includes information like their requirements and definitions. UEs can be configured with a list of subscribed networks. When a UE then connects to the network it can request these slices, and the AMF will check with the UDM or NSSF for authorization [5].

2.2.2 Next Generation Radio Access Network

The Next Generation Radio Access Network (NG-RAN) consists of the New Radio (NR) and LTE radio access. The NG-RAN nodes, i.e., the base stations, called eNB in 4G, can either be gNBs or ng-eNBs. gNBs is the new 5G base station, and they provide the user- and control plane services in NR towards the UEs, while ng-eNBs provide the LTE/E-UTRAN services. Figure 2.6 from [7] depicts the NG-RAN in relation with the 5G system. This figure also depicts the connection between the base stations and the core network. The ng-eNBs and gNBs connect by using the Xn interface, and the NG interface is used to connect with the 5GC, similarly to the X2 and S1 interfaces of 4G. The interface towards the core network is split into two different interfaces, similar to the 4G EPC; the NG-C interface is used for the control plane communication and the NG-U for the user plane traffic. The NG-C connects with the AMF in the 5GC, while the NG-U connects with the UPF [7].

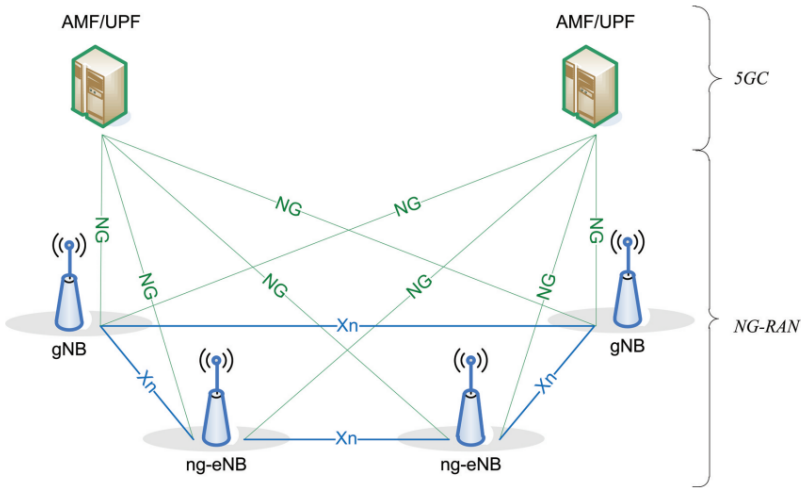


Figure 2.6: NG-RAN in relation to the 5G system [7]

2.3 Software enablement and Virtualisation

Network virtualization and Software Enablement are essential enablers for the fifth generation of mobile networks. Software Enablement means to implement software to perform tasks and functions that would previously be implemented directly in the devices hardware, this makes the hardware itself a lot more flexible and reusable. Proprietary appliance could be replaced with generic hardware able to perform the same tasks, as well as being able to be reprogrammed for new tasks, without needing to replace the entire device. One such example is Software-Define Radio (SDR), where functionality that traditionally has been implemented in the hardware is moved to software, for instance amplifiers and filters.

The term virtualization means to virtualize hardware functionalities, creating one or more virtual instances of some functionality on top of the actual appliances. One of the most popular types of virtualization is virtual machines (VMs), this is virtual operating systems running on top of the native operating system of the computer. The VMs uses the physical resources of the host machine, but all functionality is logically separate from both the host machine and other VMs running on the same host.

Software Defined Networking (SDN) and Network Function Virtualisation (NFV) are critical enablers for the fifth generation of mobile networks. SDN is a network architecture that separates the control plane from the data plane, making it more agile and flexible. Centralization of the control plane also means that a network administrator can control the network and make it easier to adapt to changes in the network without having to dive down to the infrastructure level [8, 9].

Network Function Virtualisation (NFV) means to virtualize hardware functions by instead of having them run on proprietary hardware which is traditionally used, services like routing, firewall, and load balancing can run on generic hardware in virtual instances [10].

5G is divided into three categories, ultra-Reliable Low-Latency Communication (uRLLC), enhanced Mobile Broadband (eMBB), and massive Machine-Type Communication (mMTC) [11]. uRLLC defines the latency-sensitive services, like the ones mentioned in the previous chapter, autonomous driving, and remote surgery. eMBB is enhanced mobile broadband, meaning high bandwidth internet access, and will be used by smartphones, tablets and other smart devices for things like surfing the web and video streaming. mMTC is for sensing- and monitoring devices, i.e., IoT devices, providing them with narrowband internet access [11].

Network slicing is one of the most innovative parts of the next generation of the mobile network. The concept of network slicing means to divide the network into multiple logical networks, logically isolated from each other; this is an essential part of the realization of 5G. Network slicing makes network providers able to create a customized virtual network for each use case. Each slice can be tailored to each use case, and since this is a virtualized network and not as hardware dependent as previous generations, the network can better be created and changed according to specified requirement like Service Level Agreement (SLA) and other business policies [12].

2.4 Multi-access Edge Computing

Multi-access Edge Computing (MEC) is a computing architecture standard created by the European Telecommunications Standards Institute's (ETSI's) MEC Industry Specification Group (ISG). Edge computing is a general term for having cloud services located at the *edge* of the network to bring low-latency, high-bandwidth access to appliances and applications. This can let cloud-native applications utilize the increased capabilities of lower latency and higher bandwidth.

There are many use cases for edge computing, some of which have been mentioned earlier, like autonomous cars and telesurgery, other use cases which are already being realized is augmented reality (AR) and virtual reality (VR) where low-latency is essential for the user experience.

2.5 Related Works

In "Slicing architecture from OAI-based End-to-End Network Slicing" [13], the authors set up and simulate two different slices, one created for eMBB, and the other one for uRLLC. In the latter one, they show how they change the architecture of the network to accommodate for the requirements in latency for this type of slice. By moving the management of the uRLLC slice to the edge, in this experiment, this is done by moving the core network close to the base station. Figure 2.7 depicts the architecture used to create the two slices.

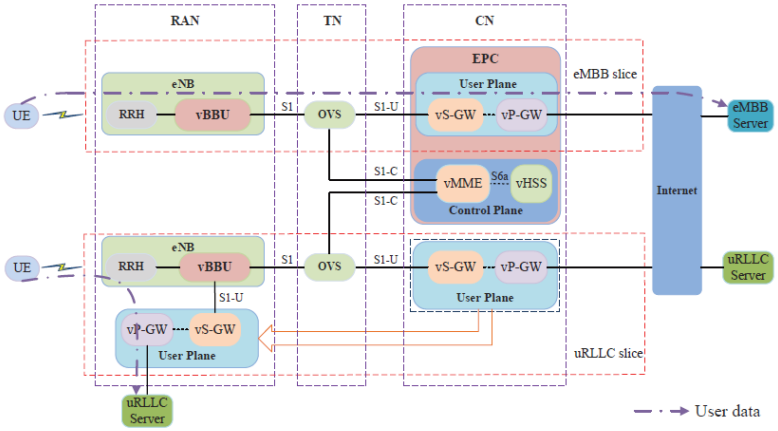


Figure 2.7: Slicing architecture from OAI-based End-to-End Network Slicing [13]

By creating these slices, they show that they successfully can accommodate the requirements for uRLLC and eMBB. Similar setups will most likely be necessary to be able to fulfill the SLA requirements for uRLLC networks.

In [14], the authors design and prototype a solution for on-demand creation of network slices using Cloud RAN. For this prototype, they use OAI [15, 16] and FlexRAN [17] SDN controller and they show that their emulated prototype reacts well to inputs coming from the SDN controller, changing the resource allocation for each slice on-demand.

In [18], the authors developing an end-to-end network slicing protocol stack called POSENS. POSENS relies on a slice-aware shared RAN, and the authors validate that the effectiveness of the solution by achieving tenant isolation and network slice customization without this affecting the performance. POSENS is built as an extension of srsLTE's UE and eNB and using OAI's core network as the software solution for the CN.

Chapter 3

Implementation

Before starting the implementation and testing for this thesis research - testing of different open-source platforms were conducted.

Software Radio Systems's (SRS) srsLTE [19] and OpenAirInterface's (OAI) [15, 16] experimentation platforms were tested, and OAI's platform was chosen, as OAI's includes a full implementation of both the EPC, eNB, and UE while the SRS platform only includes a light-weight LTE core network implementation. These two platforms can be combined, for instance by using the core network from OAI and eNB and UE from srsLTE, however, as OAI provides compatible platforms for all of these modules, it was decided to use OAI for both the core network, eNB, and UE.

3.1 OpenAirInterface

OAI is an open source experimentation platform with an active community that continuously updates and improvements to the solution. The platform is created by EURECOM and is written in standard C, supporting several real-time Linux variants. OAI provide software to simulate both the EPC, including MME, HSS, and SPGW, the E-UTRAN (the access network), i.e., the eNB as well as the UE; this can be set up either by using physical hardware, or as simulations, and it is compliant with 3GPP LTE standards [20].

OAI's hardware RF platform is designed to be able to support different types of SDR RF platforms, but officially they only support two hardware platforms, EURECOM^{TM} EXMIMO2, and USRP X-series/B-Series [20].

Figure 3.1 from [20] depicts the OpenAirInterface LTE software stack. In addition to

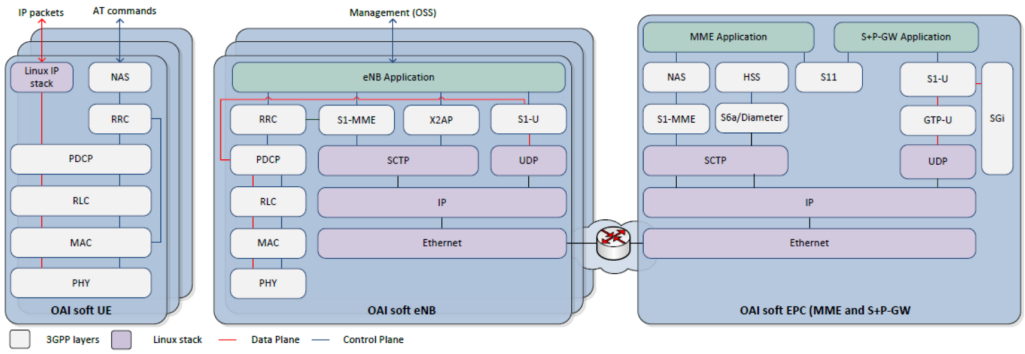


Figure 3.1: OpenAirInterface LTE software stack [20]

supporting the hardware platforms, there is a built-in emulation platform containing the full protocol stack. The emulation platform can be used to conduct realistic experimentation in a controlled environment. The platform represents the behavior of the wireless access network while obeying the parameters of the air-interface. The different layers of the platform are illustrated in 3.1, it implements the layers from the Radio Resource Control(RRC) layer to the Physical (PHY) layer. Additionally, it contains the layer-3 stack, including Internet Protocol (IP).

3.2 Lab Setup and Testing

The lab environment used in this thesis consisted of two Intel NUC computers running Ubuntu 16.04 with low latency kernel (4.13.0-36-lowlatency), and two Lime Software Defined Radios (SDRs). The two computers were connected with Ethernet/IP connective, with one of them also being connected to the Internet. The computer connected to the Internet was set up to run the core network, i.e., the HSS, MME, and SPGW while the other computer was set up to run the eNB and UEs. Figure 3.2 depicts an overview of this setup. The aim of this thesis was to experiment with software-defined simulation/emulation platforms and to create a test-bed consisting of a MEC core network, a cloud core network, an eNB and user equipment. To achieve this, multiple different implementations of the OAI platform were tested until a suitable test-bed was established, and the user equipment could reach the Internet.

In the following sections, the different implementation approaches are presented along with the challenges and results achieved from each of them. In section 3.2.1 the first approaches are explained, these did not include the core network, but included testing with the LimeSDRs. LimeSDR is a low cost, open source, apps-enabled software defined radio (SDR) platform that can be used to send and receive many different types of wireless communication standard; including UMTS, LTE, GSM, Bluetooth, and Digital Broad-

casting [21]. For this thesis, the LimeSDRs was used in the experimentation phase, both as eNB and UE.

In section 3.2.2 the core network is implemented, both in one of the NUCs, at an instance in an OpenStack Cloud and in a virtual machine instance on the same NUC as the one running the other core network. The proposed solution for the test-bed is presented at the end of this chapter.

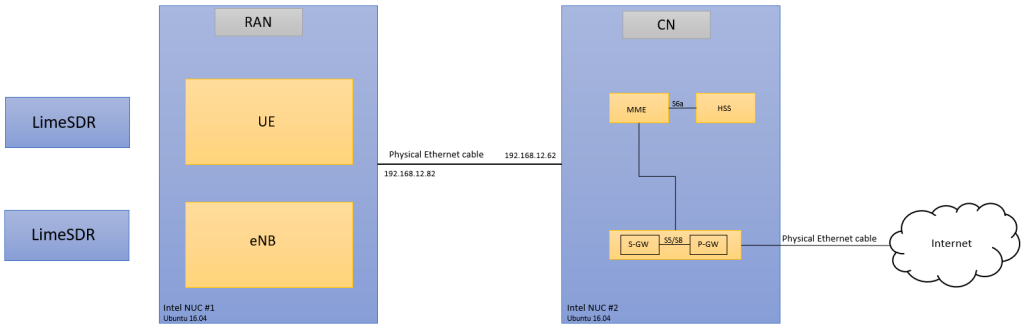


Figure 3.2: Lab setup

3.2.1 OAI without S1 Interface

The initial testing of the OAI solution was using the *oaisim* without the S1 interface, i.e., without the core network. This simulation contains both the eNB and the UE in one build, and it was set up in a virtual environment. Following the instructions on OAI's Gitlab wiki page [22], the simulation was build and run successfully, and IP connectivity was confirmed between the eNB and the UE.

To verify the functionality of the LimeSDRs, LimeSuite, and SoapySDR [23] was installed. LimeSuite and SoapySDR is also needed for the LimeSDRs to work correctly together with OAI. The SDRs were tested using the instructions from Myriad-RF [24]; these tests verify that the SDR works as they should and include both a "self-init" test and an example configuration for testing. Both SDRs were successfully set up and verified to be working correctly. The eNB from OAI was then properly configured and compiled to use the LimeSDR, using one of the Intel NUC machines.

A new version of the OpenairInterface5G was released in late 2018, in this version, the old *oaisim* build, i.e., the simulated UE and eNB was replaced by softmodems, separate buildable files for the UE and the eNB. These can be configured to use either real hardware, like a LimeSDR, or simulations.

This new version of the simulated OpenAirInterface5G softmodems was therefore used [25]. First, a simulation was set up on two different virtual machines without the S1

interface, one running the UE and one the eNB. After configuring the softmodems to use the local connections between the two virtual machines, they were successfully built and connected. The simulated OAI interfaces of the UE and eNB were brought up, and IP connection was verified by sending IP packets, i.e., pinging, between the UE and the eNB via the simulated interfaces. The same version of OAI was then set up in the lab environment on the NUC computers, on one NUC each, similarly to the setup in the virtual instances, the connection process went successfully, and IP connectivity was verified.

Next, the UE and the eNB were configured to use the LimeSDRs, at a low transmission power, to not interfere with other networks. Both were successfully built and able to find its connected Lime SDR when set to run, Figure 3.3 depicts the lab setup with the LimeSDRs set up as eNB and UE. However, the UE was not able to connect with the eNB wirelessly. Different configurations were tested, but it did not yield the desired results, and as coaxial cables were not available to connect them, it was decided not to use the two LimeSDRs for this experimentation and rather use simulated/emulated hardware instead.

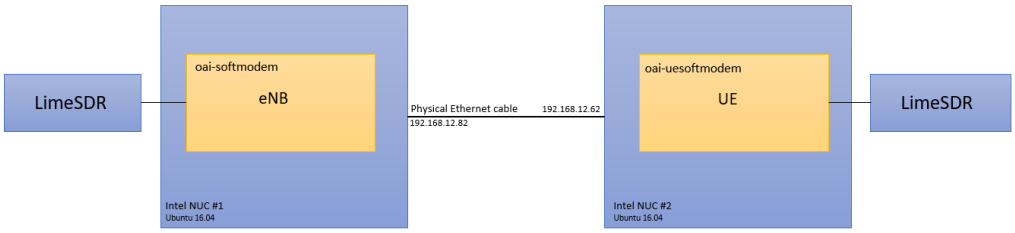


Figure 3.3: Lab setup without core network

3.2.2 EPC Implementation and S1 Interface

For the next part of the experimentation, the EPC was implemented. OAI's guide [26] on how to configure and build the core network was used in this phase. The EPC, including the MME, HSS, and SPGW, was configured and build successfully on one of the NUC machines. For the MME to be able to connect with the HSS, its hostname has to be added to the database, as the HSS makes a query to its database to check if the MME's hostname is valid before allowing it to connect.

The simulated UE and eNB were configured and build on one machine, connected via Ethernet to the other machine running the EPC. The connection process went successfully, the eNB connected to the MME and the UE attached to the eNB. The UE also received an IP address form the EPC. However, IP connectivity between the UE and the EPC was not successfully verified over the S1 interface. Even though the UE received an IP address for

its simulated interface, the EPC could not reach the UE. When troubleshooting this issue, it was found that the traffic did go through in one direction when the UE was pinging the gtp0 interface of the SPGW, the traffic was observed incoming on this interface using *tcpdump*. However, the replies did not find its way back to the UE, even though routing had been set up to send the packets over the S1 interface.

L2 nFAPI Simulator

An updated release of OpenAirInterface5g, version v1.0.0, was released in January 2019, this include a new simulation setup that uses network functional API (nFAPI) simulator for the UE and the eNB, i.e. new versions of the *lte-softmodem* for the eNB and *lte-uesoftmodem* for the UE, and it allows for the creations of multiple UE running in the same *lte-uesoftmodem* [27]. This OAI solution builds on Cisco's Open-nFAPI implementation which defines a network protocol that can connect a Layer 1 LTE running on Physical Network Function (PNF) to a Virtual Network Function (VNF) running LTE layer 2 and higher. They aim to provide an open interface between layer 1 and 2, this to allow for sharing of PNF's between VNF's [28].

OAI's L2 nFAPI simulator allows for testing of layer 2 of the OSI Model, and higher layers, i.e., it does not include the physical layer. Figure 3.4 from [29] displays the which layers of the OSI model the different simulators implement, the red is the *L1 simulator*, the blue is the *ulsimdlsim* simulator, yellow is the *basic simulator* while green is the simulator used in this thesis, the *L2 FAPI simulator*. As depicted here, the L2 FAPI simulator includes the MAC, RLC and PDCP layers, and do not simulate the PHY nor the channel of the eNB and the UE.

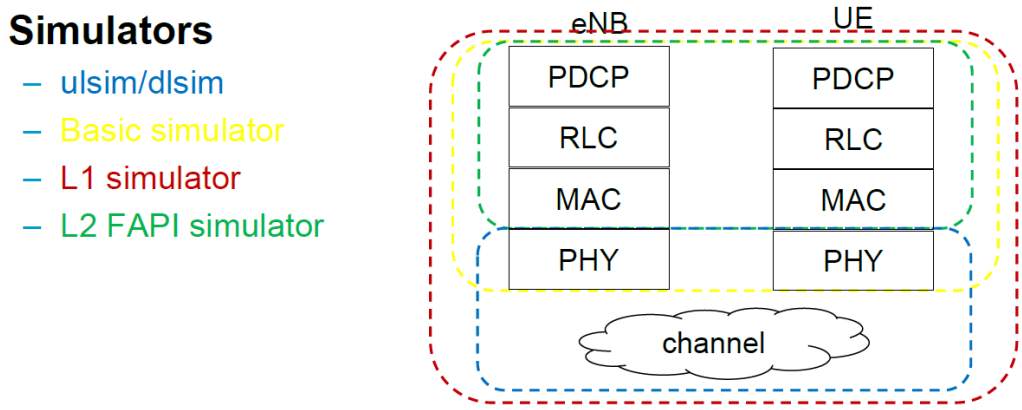


Figure 3.4: OAI simulators [29]

The different simulators are also depicted in Figure 3.5, showing the functional spit of the eNB. In this figure the connection between the Radio-Cloud Center, Radio-Access

Unit, and Remote Radio-Unit is depicted, where the lower part of the figure displays the nFAPI is the one used for the implementation in this thesis.

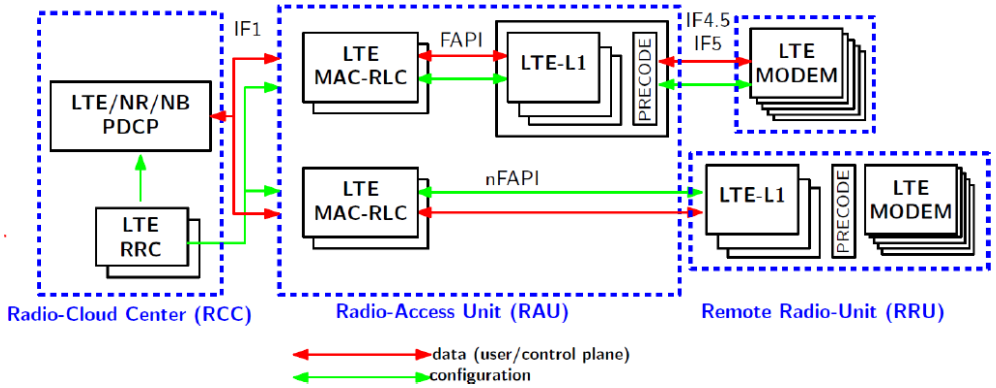


Figure 3.5: OAI simulators [29]

Before building the *lte-uesoftmodem*, the specifications of each desired UE needs to be configured, including IMEI, IMSI, and keys for each of the UEs to be simulated. For the UEs to be able to attach to the network, their SIM information has to be in the HSS database; this is because the MME checks the HSS database whenever a device wants to connect to the network to verify if it is allowed to connect. The SIM configuration also needs to be set to the correct Public Land Mobile Network (PLMN) code, consisting of the Mobile Country Code (MCC) and Mobile Network Code (MNC), indicating which network it can connect to. For the SIM configuration using OAI's nFAPI, the file $ue\_eurecom\_test\_sfr.conf$ was used, here all the network PLMNs the UEs can connect to are configured, and for each UE the PLMNs are defined, as well as the desired one. The desired PLMN is used as the first five numbers of the IMSI numbers when the *lte-uesoftmodem* is compiled, and then each UE gets a unique number to define its IMSI number. The PLMN configured for the UEs needs to be the same as the one configured at the eNB and the MME, otherwise, the UE will not be able to connect to the network.

With this new approach, IP connectivity was verified to work as expected. In this trial the simulation setup contained 3 UEs and one eNB, all thre UEs attached successfully to the EPC and received IP addresses from the EPC. Ping was successful in both directions to all three UEs. The UEs was also able to ping the Internet.

As of May 2019, this is the only validated deployment setup from OAI using nFAPI, i.e., having the UE and eNB on the same machine, using S1 interface to connect to the EPC, deployed on a separate machine. This setup is depicted in figure 3.6 from [27], updated with the IP addresses used in this experimentation.

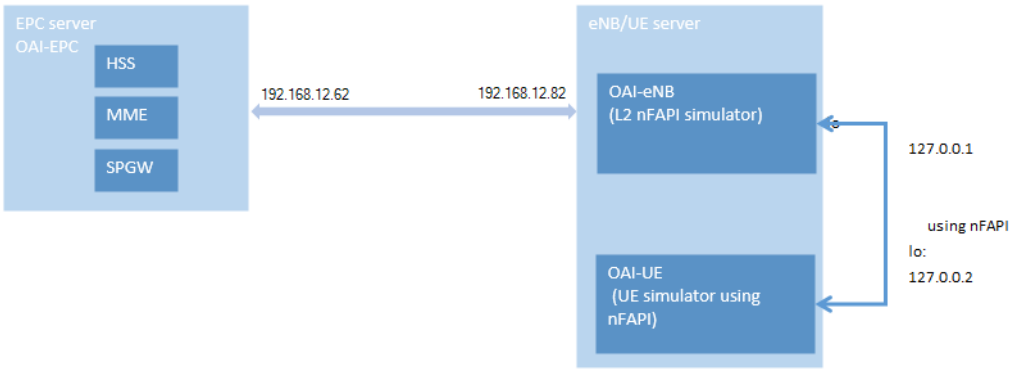


Figure 3.6: Lab setup using nFAPI [27]

3.2.3 Cloud Core

One of the objectives of this thesis is to experiment with cloudification. The next testing scenario included an OpenStack Cloud environment. Before we move into the specifics of the lab environment, it is useful to understand the definition of distance in a network. While distance is usually defined in length of geography, we are specifically referring to network distance in terms of network metric and network path distance. Having the core network located in the cloud, usually meaning further away from the base station versus having it closer, means that the Quality of Experience (QoE) for the end user can be degraded due to longer delays and higher latency.

In the cases where the network is used for activities like web surfing or chatting, this extra will delay be negligible. However, one of the expected attributes of 5G is the ability to support all kinds of communication demands, like self-driving cars, and remote surgery, where ultra-low latency is crucial.

OpenStack Cloud and VPN Tunnel

An Ubuntu 16.04 instance was created on the cloud, with the same kernel as the one used on the local NUC machine. The same version of the OAI core network (v0.5.0) as used on the NUC machine was successfully configured and built on the cloud instance. Then an OpenVPN server was deployed on the cloud instance to create a VPN tunnel between the cloud instance and the NUC running the eNB. The NUC running the eNB was set up as a VPN client, and the VPN tunnel allowed the NUC to reach the internal network of the cloud instance and vice versa [30].

The eNB was reconfigured to use the VPN tunnel and connect to the EPC deployed in the cloud. However, the eNB was not able to connect to the cloud EPC, even though the

connection was verified, and the Maximum Transmission Unit (MTU) was set to be the same as on the local connection between the NUCs. It was also verified that this MTU size was valid, and that it was possible to send packets of the specified size over the VPN connection. I used packet capture tools (tcpdump) to troubleshoot the issue, but was unable to find any clear cause for the issue.

Additional EPC in VirtualBox

As the implementation of the cloud core did not yield the desired results, and connectivity between the eNB and core network was unsuccessful, another core network was implemented in a virtual instance using VirtualBox, on the NUC already running the core network. The virtual machine was set up with the same specifications as the host machine, i.e., Ubuntu 16.04 and low-latency kernel. The same version of the OAI core network as used on the cloud instance and directly installed on the NUC was used here as well(v0.5.0). The VM was set up with two network interfaces using bridge adapters for both of them. When using a bridge adapter, the VM gets a regular IP address, similarly to the host machine. When this is set up, the desired network interface of the host machine is chosen to be the one used by the VM, and the host and the VM can reach each other using these IP addresses. If the host machine is connected to a network with a Dynamic Host Configuration Protocol (DHCP) server, i.e., it gets its IP address dynamically, the VM interface will as well. In the lab environment used in this thesis, the IP address for interface towards the other NUC was configured manually, while the interface connected to the Internet received an IP address from a DHCP server at the University; therefore, the VM was set up similarly, with automatic and manual IP addresses for the corresponding interfaces. To make the VM simulate a cloud environment, simulated delay was added to the link towards the eNB.

The VM instances created in VirtualBox is automatically set up a with Network Address Translation (NAT) interface. This means that the VM gets an internal IP address which is translated into an external address when connecting with the outside. The internal IP address is only reachable from within that VM, but the VM can reach other IP addresses using this interface. This means that the VM could reach the Internet via the host machine, as well as every other address the host machine was able to reach. However, if a NAT interface had been used, the host machine and the other NUC machine could not reach the VM, as it was not assigned an IP address outwards.

To make the eNB connect to both core networks simultaneously, i.e., create a S1 channel to both of them, the eNB configuration had to be modified. With the new release of OpenAirInterface5G, S1-flex was introduced, making the eNB able to connect with multiple core networks simultaneously. The configuration approach for this is described in [31], here the changes that have to be made in the configuration of the eNB is described. This

includes adding the additional PLMNs to the PLMN list, adding the IP address for the additional MMEs as well as including an additional field for each MME, stating which PLMN setting is used for each of the MMEs. This information needs to be reflected in the MME configuration of the core network, where they have to be configured with the same PLMN information as stated in the eNB configuration (or the other way around).

After the S1-Flex had been configured correctly, the eNB successfully connected to both core networks. Figure 3.7 depicts the setup at the lab, including the S1 interfaces from each of the core networks to the eNB, using S1-Flex configuration.

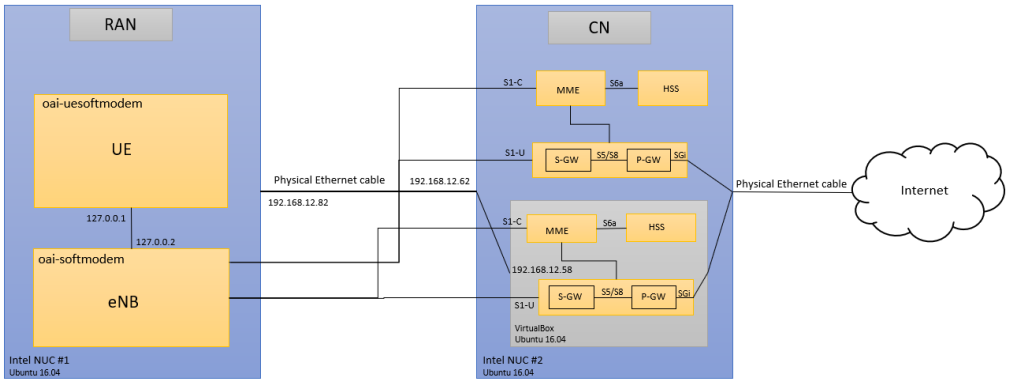


Figure 3.7: Lab setup

The configuration for the UEs were also modified to reflect this new setup, three UEs were configure, two of them were kept on the network for the "old" core network, i.e., the core network implemented directly on the NUC machine, while the last one was configured to attach to the new core network running in the VM. The MME in the VM was configured with a different PLMN than the one already configured on the MME at the host machine.

The plan was to have these UEs connect to the two core networks simultaneously, two to the core network on the host machine and one to the core network in the VM, with the VM set up to act as a cloud environment. However, even though the eNB was able to connect with both core networks simultaneously, it was not able to attach the UEs to different core networks. All the attach messages sent from the eNB to the core network regarding the UEs were sent to the MME defined first in the MME list at the eNB.

Because of this, the UEs with different PLMN was not able to connect simultaneously. If the core network running directly on the host machine was defined first, all three UEs tried to connect here, and as the third UE was configured to connect to the core network in the VM, the MME was not able to verify it, and it was not able to connect. This scenario was also tested the other way around, with having the core network in the VM defined first both in the MME list and PLMN list in the eNB configuration file and the

result was then the same, just reverted. The third UE was now able to connect but not the two others.

It was decided to have the second core network (in the VM) set up with the same PLMN configuration as the one running on the host machine and use this for testing how the UEs experienced the difference when connected to the two different cores, as the UEs were not able to connect to the two different core networks simultaneously. Figure 3.8 displays the proposed solution used in the evaluation and testing phase, explained in the next chapter.

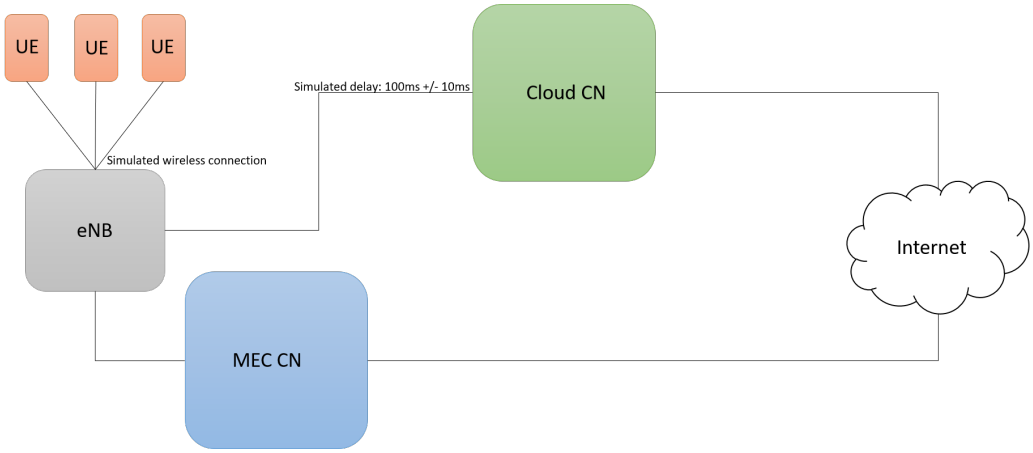


Figure 3.8: Proposed solution

3.3 Production LTE Network

An additional lab environment was set up at the offices of Tampnet AS in Stavanger. Tampnet provides LTE service in the Noth Sea. This setup included a production eNB and a Cisco 4G router connected to a production core network. The configuration and setup needed to do tests on this setup was minimal as both the core network and the eNB was already fully configured and up and running.

A Tampnet SIM card was provisioned to one of Tampnets APNs using a private IP address, and the UE connected to the eNB at the lab. The cellular interface on the router was directly connected to the eNB via a coaxial cable. A static IP address was configured on the HSS for the SIM card and once it was connected to the eNB and all configuration was correctly set up at the UE, like having the APN configuration on the UE match the configuration on the HSS, the UE attached successfully to the eNB and was verified connected to the core network. The assigned static IP address for the cellular interface of UE was, as mentioned, an internal IP address which means that the UE

does not automatically have access to the Internet, however, this can be configured. The traffic was routed from the UE, via the eNB thorough Tampnet's infrastructure to the core network. From here there traffic is routed to the users site, providing connectivity between the UE and the user side.

Tampnet's LTE network contains two core network sites; one located at the Norwegian edge of the network, and the other located at the edge of the netowrk in the Netherlands. Tampnet owns and operates the network between the mobile core sites. Tests and comparison of the experienced latency when connected to either of them were conducted.

Chapter 4

Evaluation of Lab Trials

4.1 Test Setup

After multiple different lab scenarios were tested and presented in the previous chapter, the functional setup was found to be using two Intel NUC machines connected via Ethernet/IP, where one was running two core networks and also connected to the Internet and the other one running the eNB and UEs using the nFAPI simulator. As depicted in the previous chapter in figure 3.7 and 3.8, this setup was used as the basis in the next phase of this thesis, which included evaluation and testing.

A couple of different scenarios were set up to evaluate the network. As two cores were implemented, they were configured at the same PLMN, meaning that the eNB connected to both of them, but only one PLMN setting were distributed to the UEs, and they connected to the core network defined first at the eNB.

4.1.1 Evaluation Tools

The packet generation tool iPerf3 was used to generate test traffic over the S1 tunnel. iPerf can measure maximum bandwidth using TCP, as well as the tuning of various parameters using UDP, like desired speed and packet size, reporting bandwidth, delay jitter, datagram loss over the connection[32].

Both Wireshark and vnStat was used to capture the traffic sent over the S1 link to and from the UE. vnStat is a network monitoring tool that keeps logs of network traffic for the selected interface or interfaces, which in this scenario was set to be the UE interface oip1. It is an open source program build for Linux and BSD, and it uses the network interface statistics provided by the kernel as the information source, meaning that it does

not sniff the traffic, which makes it lightweight for the system resources regardless of the amount of traffic sent over the network. Wireshark is one of the most widely-used network protocol analyzers; one can look at the all the traffic that is being sent over the desired interface including information about things like the protocol, check-sums and response time [33].

One of the objectives of this thesis is to have the eNB connect to both a Multi-access Edge Core (MEC) core and a cloud core. Netem's Network emulator was used to simulate that the EPC in the VM was located in a cloud environment. The Linux Foundation offers a guide on how to implement this, which is an already enabled tool in the kernel of multiple newer Linux distributions. It contains functionalities for testing protocols by emulating properties of wide-area networks and contain emulations for delay, loss, duplication and re-ordering [34].

By using Netem's network emulator, the desired delay of 100ms +/- 10ms, could be added to the bridge adopter interface of the VM, making it appear to be located further away but still have the same latency and access to the Internet. Adding the delay to only the interface of the VM also makes sure that the core network running on the host machine does not get affected by this, and that this core network can therefore still act as an edge core.

4.1.2 Test Setup with Production LTE

To test the setup of the production LTE network, an EXFO hardware traffic generator was used to generate the desired amount of traffic on the link. This traffic tester can also record the latency and jitter on the link, which is what the users of the link would experience. To perform these tests, another router was connected to the Cisco modem using an Ethernet cable. IP addresses were configured on the link between them, and the router was brought "online," on the internal network. A loopback was configured on another port on the connected router, making it send all incoming traffic right back again, simulating a reply being sent back. The traffic generator is also configurable as a tester, meaning that thresholds can be set for acceptable latency and jitter. This makes it easy to test any setup and find out what is the maximum throughput should be set on the link given the signal levels of the service.

The traffic generator is also used to test the stability of the link, to determine how much traffic the link is able to handle before it affects the QoE for the user. To evaluate the performance of an LTE link parameters like Signal-to-noise ratio (SNR), Signals Received Power (RSRP), Received Signal Strength Indicator (RSSI) and Reference Signal Received Quality (RSRQ) are used. These parameters reports information about how strong the signal is and how much interference there is. If the signal levels of the UE is not optimal,

the increased traffic load will make the Quality of Experience (QoE) for the user worse because there are fewer resource blocks on the link. The link might be able to send traffic, but if the amount of traffic is increased above what the channel can handle, packets will be dropped and the QoE for the user will be affected. These parameters have, however, not been considered in this thesis as the production LTE setup uses a coaxial cable between the UE and the base station, and the lab set up at the University uses emulated hardware.

The vendor of the core network's network manager system was used to capture the traffic. With this tool, both the eNB and the UEs can be monitored, capturing both the signal levels and the throughput of the UEs at the eNB side. This means looking at the signal the eNB are receiving from the connected UEs.

4.2 Test Results

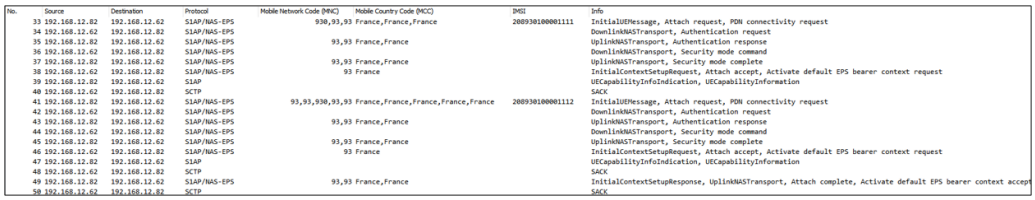
To verify that the emulated UE and eNB connected correctly to the core network, Wireshark was used to capture the packets sent between the two NUCs. The first screenshot, Figure 4.1, displays the messages sent when the eNB connects to both of core networks. In the figure, one can see that the eNB first send a Stream Control Transmission Protocol (SCTP) message to both core networks to initialize the connection. Once the core networks have acknowledged the connection, the eNB sends S1 Application Protocol (S1AP) messages to both of them, including the PLMN (MNC, MCC) information, requesting to connect to the networks. Both core networks accept the connection by sending S1AP messages back to the eNB, bringing up the S1-C channels (S1-Flex).

No.	Source	Destination	Protocol	Mobile Network Code (MNC)	Mobile Country Code (MCC)	Info
1	192.168.12.82	192.168.12.62	SCTP			INIT
2	192.168.12.62	192.168.12.82	SCTP			INIT_ACK
3	192.168.12.82	192.168.12.62	SCTP			COOKIE_ECHO
4	192.168.12.82	192.168.12.58	SCTP			INIT
5	192.168.12.62	192.168.12.82	SCTP			COOKIE_ACK
6	192.168.12.58	192.168.12.82	SCTP			INIT_ACK
7	192.168.12.82	192.168.12.62	S1AP	93,93	France,France	S1SetupRequest
8	192.168.12.82	192.168.12.58	SCTP			COOKIE_ECHO
9	192.168.12.62	192.168.12.82	SCTP			SACK
10	192.168.12.58	192.168.12.82	SCTP			COOKIE_ACK
11	192.168.12.82	192.168.12.58	S1AP	93,93	France,France	S1SetupRequest
12	192.168.12.62	192.168.12.82	S1AP	93	France	S1SetupResponse
13	192.168.12.82	192.168.12.62	SCTP			SACK
14	192.168.12.58	192.168.12.82	SCTP			SACK
15	192.168.12.58	192.168.12.82	S1AP	93	France	S1SetupResponse
16	192.168.12.82	192.168.12.58	SCTP			SACK

Figure 4.1: eNB connecting to both core networks

Figure 4.2 displays the messages sent between the eNB and the MEC core network to connect the UEs, the eNB first sends an S1AP/NAS-EPS messages to the core network. These messages contain requests to have the UE connected to them, and they include information like IMSI and IMEI number, key, and PLMNs of the connecting UE which the MME checks against the HSS database to verify the connection. Once this has been

verified, the core network responds with an acknowledgment message, attaching the UE to the network. The SPGW then assigns an IP address to the UE, and the S1-U interface is up. As long as the P-GW has access to the Internet, the UE will too. In the figure two UEs are connecting to the network, displaying the same messages sent between the eNB and the core network for both of them.



No.	Source	Destination	Protocol	Mobile Network Code (MNC)	Mobile Country Code (MCC)	IMSI	Info
33	192.168.12.82	192.168.12.62	S1AP/NAS-EPS	930,93,93	France,France,France	208930100001111	InitialUEMessage, Attach request, PDN connectivity request
34	192.168.12.62	192.168.12.82	S1AP/NAS-EPS				DownlinkNASTransport, Authentication request
35	192.168.12.82	192.168.12.62	S1AP/NAS-EPS	93,93	France,France		UplinkNASTransport, Authentication response
36	192.168.12.62	192.168.12.82	S1AP/NAS-EPS				DownlinkNASTransport, Security mode command
37	192.168.12.82	192.168.12.62	S1AP/NAS-EPS	93,93	France,France		UplinkNASTransport, Security mode complete
38	192.168.12.62	192.168.12.82	S1AP/NAS-EPS	93	France		InitialContextSetupRequest, Attach accept, Activate default EPS bearer context request
39	192.168.12.82	192.168.12.62	S1AP				UECapabilityInfoIndication, UECapabilityInformation
40	192.168.12.62	192.168.12.82	SCTP				SACK
41	192.168.12.82	192.168.12.62	S1AP/NAS-EPS	93,93,930,93,93	France,France,France,France,France	208930100001112	InitialUEMessage, Attach request, PDN connectivity request
42	192.168.12.62	192.168.12.82	S1AP/NAS-EPS				DownlinkNASTransport, Authentication request
43	192.168.12.82	192.168.12.62	S1AP/NAS-EPS	93,93	France,France		UplinkNASTransport, Authentication response
44	192.168.12.62	192.168.12.82	S1AP/NAS-EPS				DownlinkNASTransport, Security mode command
45	192.168.12.82	192.168.12.62	S1AP/NAS-EPS	93,93	France,France		UplinkNASTransport, Security mode complete
46	192.168.12.62	192.168.12.82	S1AP/NAS-EPS	93	France		InitialContextSetupRequest, Attach accept, Activate default EPS bearer context request
47	192.168.12.82	192.168.12.62	S1AP				UECapabilityInfoIndication, UECapabilityInformation
48	192.168.12.62	192.168.12.82	SCTP				SACK
49	192.168.12.82	192.168.12.62	S1AP/NAS-EPS	93,93	France,France		InitialContextSetupResponse, UplinkNASTransport, Attach complete, Activate default EPS bearer context accept
50	192.168.12.62	192.168.12.82	SCTP				SACK

Figure 4.2: Two UEs connecting to the network

iPerf was used to generated traffic over the S1 channel between the gtp0 interface of the SPGW and the UE interface, and by sending TCP traffic, iPerf found the maximum bandwidth of the link. As depicted in 4.3 captured at the UE, the maximum speed captured on the downlink(DL) was 7.78 Mb/s and up to 19.08 Mb/s on the uplink(UL). The upper part of the figure is the UL/DL speeds captured using vnStat, here shown as Rx/Tx, i.e., received and transmitted data, measured in bits per second, while the lower part of the figure is captured simultaneously using OAI's Graphical User Interface (GUI). The traffic captured by vnStat is not entirely the same as the once captured by OAI's GUI but very close, verifying that the throughput speeds of both the uplink and downlink are correct. These tests were conducted both having the UE connected to the MEC core and the cloud core, and the maximum acquired bandwidth was captured to be the same.

Multiple tests were conducted in order to measure and compare the latency when connected to the two core networks. Using iPerf and UDP, it is possible to configure iPerf to send traffic at a desired speed and time. Even though UDP does not send any reply does iPerf report lost packages and jitter, this is achieved by setting the receiver up a an iPerf server and the sender as an iPerf client. iPerf with TCP was first used to measure the latency experienced at the UE but because of TCP's properties like Slow-Start and congestion control does this not give a full picture of the latency, UDP was therefore used instead.

A UDP stream between the UE and an external computer, was set up. Since UDP is a best-effort protocol with no acknowledgment of received packets, the latency were calculated by capturing the traffic in both ends using Wireshark, i.e., capturing the timestamp of each packet being sent from the UE and arrival time of each packet at the external computer. The clocks of both computers were synchronized using Network Time Protocol (NTP). A C# program was created to calculate the delay between the UEs and

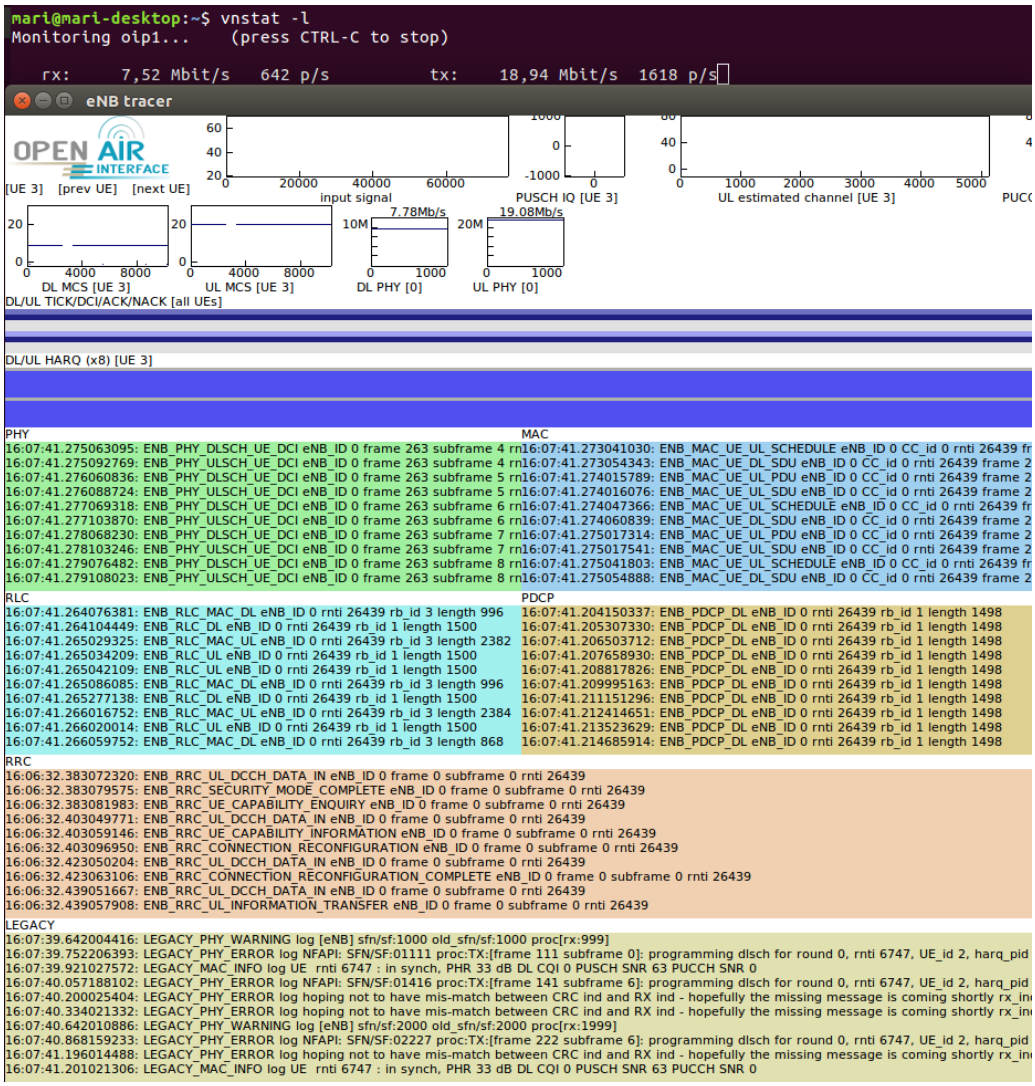


Figure 4.3: OAI's graphical interface and vnStat live

the receiving computer. The Wireshark files were saved as *.csv* files, containing only the information needed to calculate the delay for each UE.

Tests were run both with one UE connected, and later with up to 10 UEs connected - in order to evaluate the network, having both the cloud and MEC core network serving the UEs. When multiple UEs were connected, UDP traffic was generated at the UEs and sourced from the UE interface. With iPerf3, the server can only receive one incoming stream of traffic at a time, and as such, to have all connected UE send traffic to the same destination, one iPerf3 server was set up to listen to each of the UEs. The destination port was also needed when calculating the delay of each UE, as the receiving computer

only saw the IP address of the SGi interface of the P-GW and could therefore not separate the traffic for each UE without this.

Figure 4.4 and 4.5 display the latency of a UDP stream from the UE to the external computer when sending test traffic at 3Mbps over the link using iPerf. The speed of the generated test traffic was set to 3Mbps because if it was set higher the amount of lost packets were almost 50% when connected to the cloud core. To make the tests more representative, and be able to compare the experienced latency when connected to both core networks, the speed needed to be set to the same in both scenarios. The graph compares the latency experienced at the UE when connected to the MEC core network and the cloud core network when only one UE is connected. Figure 4.4 depicts the median latency per second, where an average of 255 datagrams are sent per second where each datagram has a size of 1472, while Figure 4.5 depicts the latency of all packets sent. As can be seen here, the latency when only one UE is connected is stable in both scenarios, but the latency is, as expected, higher when connected to the cloud core network. When connected to the MEC core network, the average latency is 66 milliseconds while it is 246 milliseconds when connected to the cloud core network.

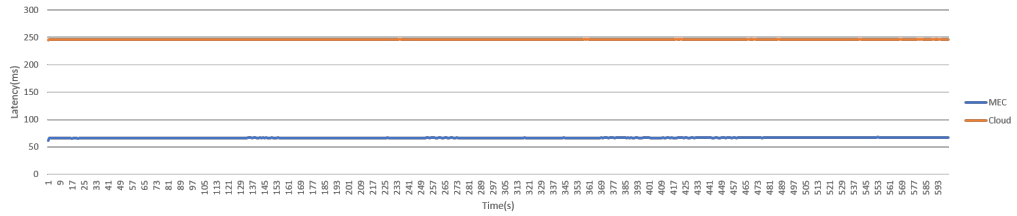


Figure 4.4: Median latency per second, 1 UE connected sending 3Mbps

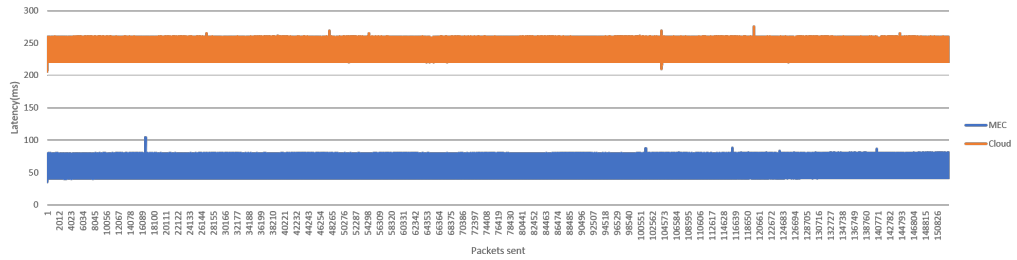


Figure 4.5: Latency of all packets, 1 UE connected sending 3Mbps

Jitter and lost packets in the UDP stream were calculated at the server side per second by iPerf3. Figure 4.7 and 4.6 depicts the jitter and lost packets for the UE, comparing the cloud core network and the MEC core network. As can be seen in 4.6, the jitter is very stable in both scenarios, though a bit higher when connected to the cloud core network.

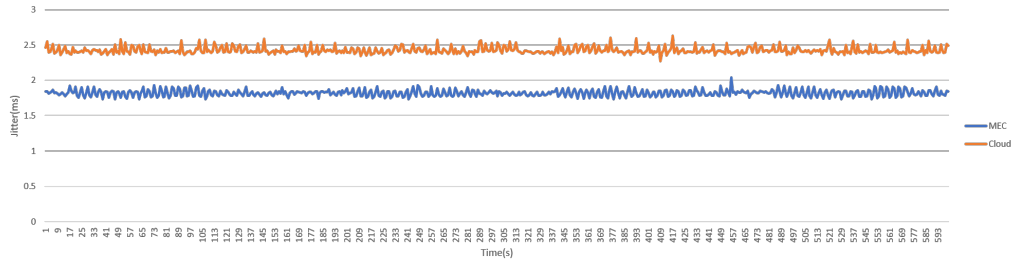


Figure 4.6: Jitter in milliseconds, 1 UE connected sending 3Mbps

Figure 4.7 shows that there was not much packet loss in either of the two scenarios, for large parts of the tests are there no lost datagrams. There are some spikes in the graph for both scenarios, but these are at a max of just beneath 0.8%, meaning that 2 of 255 packets were lost (0.78%).

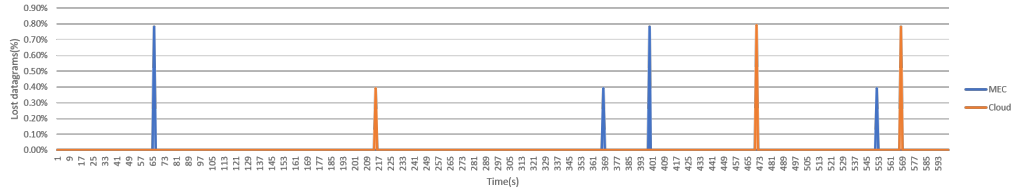


Figure 4.7: Packet loss in %, 1 UE connected sending 3Mbps

For the next test scenario, 10 UEs were configured and attached to the eNB. Tests were conducted similarly to the ones described above, but for these tests, only 100kbps were sent from each UE to not overload the system. Each UE was set to send UDP traffic from their assigned interface to different ports on the receiver PC.

Figure 4.8 and 4.10 displays the median latency per second of each UE and Figure 4.9 shows the latency of all the packets when connected to the MEC core while Figure 4.11 shows the latency of all packets when connected to the cloud core. The immediate difference is first that, as expected, the latency for each of the UEs is lower when connected to the MEC core versus when connected to the cloud core. Overall, the average latency of the UEs is 240ms when connected to the MEC core and 300ms when connected to the cloud core. Comparing this to the latencies found when only one UE was connected have the latency when connected to the MEC core increase significantly, while it has not increased that much for the scenario with the cloud core.

Figure 4.12 and 4.13 shows the jitter in milliseconds from the same test runs. Here one can see that when connected to the MEC core, the overall jitter of the UEs is a bit lower than when connected to the cloud core, but the difference is minor. However, several UEs have better jitter values when connected to the MEC core than to the cloud core,

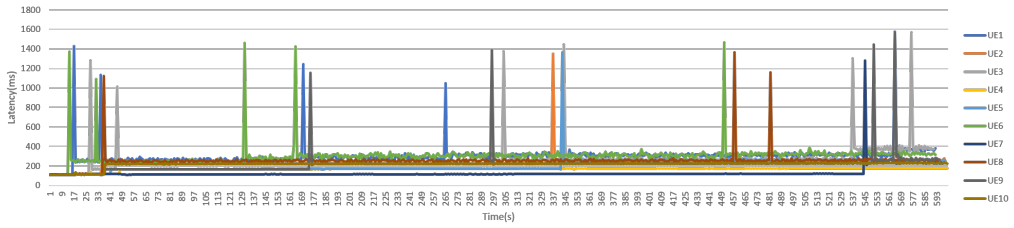


Figure 4.8: Median latency per second, 10 UEs connected to the MEC core sending 100kbps

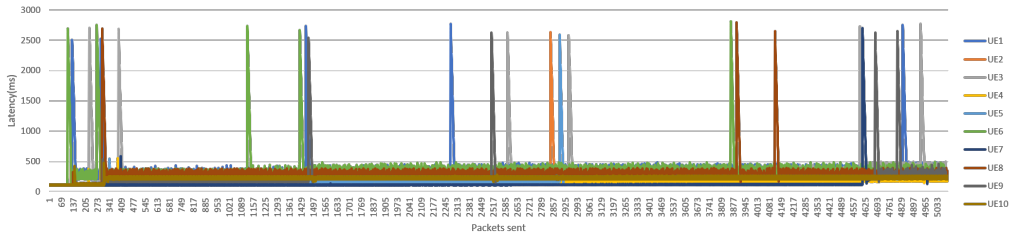


Figure 4.9: Latency of all packets, 10 UEs connected to the MEC core sending 100kbps

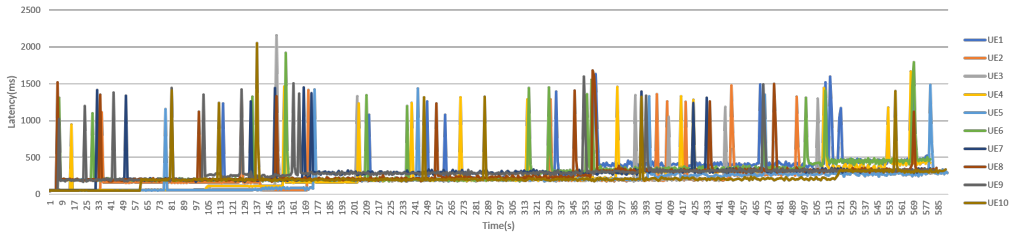


Figure 4.10: Median latency per second, 10 UEs connected to the cloud core sending 100kbps

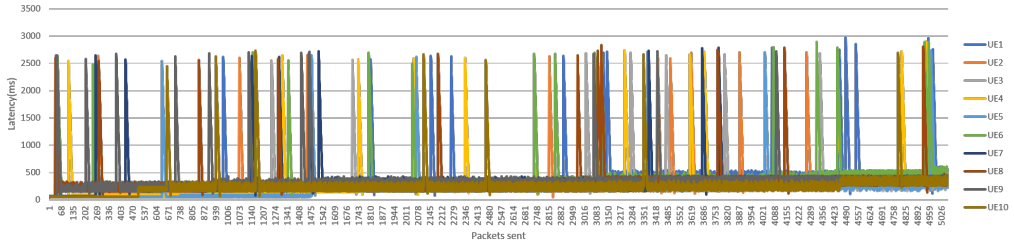


Figure 4.11: Latency of all packets, 10 UEs connected to the cloud core sending 100kbps

and there are also here more spikes, similar to what could be observed in the latency graphs.

Figure 4.14 and 4.15 shows the number of lost datagrams in the UDP stream in %. Here

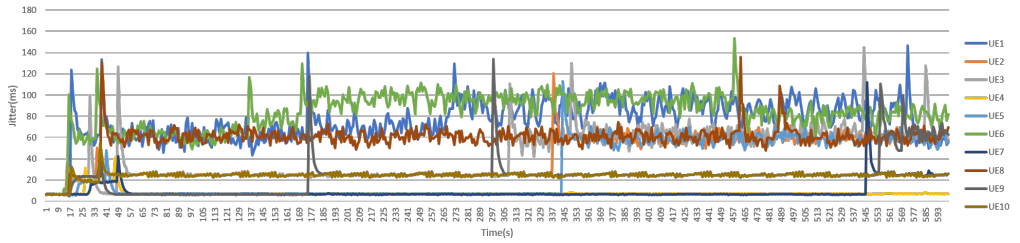


Figure 4.12: Jitter in milliseconds, 10 UEs connected to the MEC core network sending 100kbps

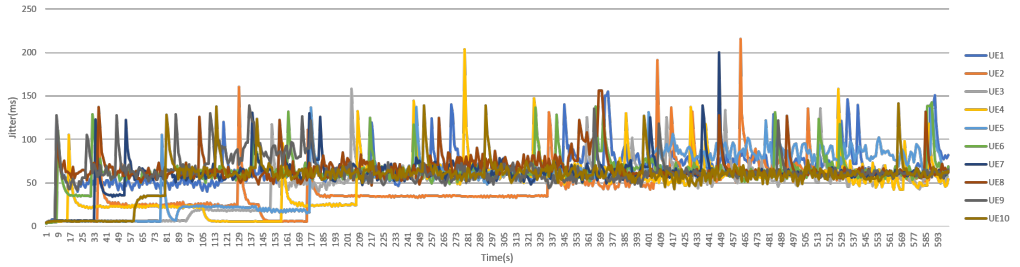


Figure 4.13: Jitter in milliseconds, 10 UEs connected to the cloud core network sending 100kbps

the same pattern can be observed; there are a lot more lost datagrams when 10 UEs are connected compared to when only one UE is connected, and there is also quite a lot more lost packets when connected to the cloud core. When connected to the MEC core the number of lost datagrams is around 5%, except for UE 6 which loses 33% of the datagrams sent in the packet in the beginning. When connected to the cloud core network, however, there is a lot more frequent and higher numbers of lost datagrams, with two peaks at 50% and two at 33%.

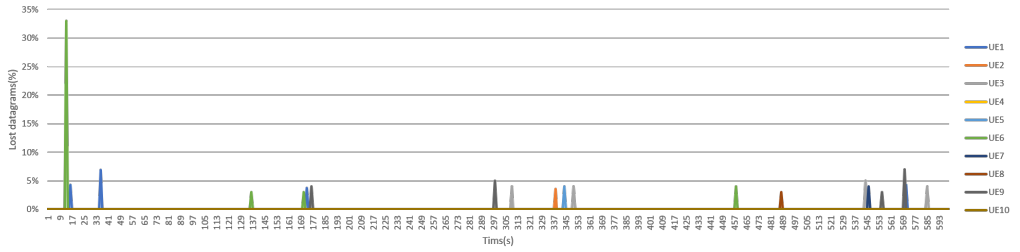


Figure 4.14: Lost datagrams in %, 10 UEs connected to the MEC core network sending 100kbps

In both scenarios with 10 UEs connected, there are latency spikes which do not occur when only one UE is connected. These are more significant in the case where the UEs

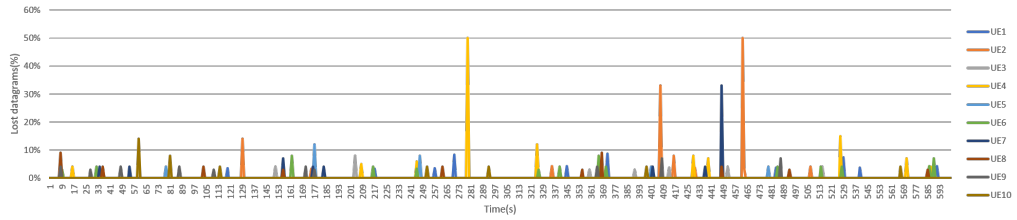


Figure 4.15: Lost datagrams in %, 10 UEs connected to the cloud core network sending 100kbps

are connected to the cloud core, here there are a lot more latency spikes, mostly up to between 1000 and 1500ms but also some that are up to 2000ms, i.e., up to 2 seconds. When connected to the MEC core, the latency spikes are not that frequent, and they have a maximum of 1600ms. When correlating the latency spikes of each UE with the lost datagrams, one can see that the latency spikes occur when there is packet loss. This correlation can clearly be seen in Figure 4.16 where only the lost datagrams and latency from UE 1 connected to the cloud is displayed.

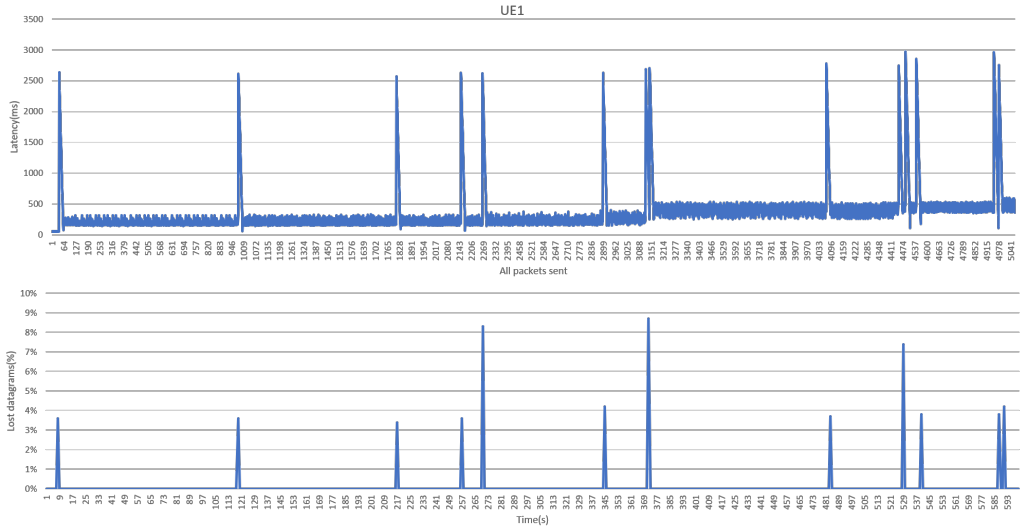


Figure 4.16: UE1 connected to the cloud core network

The source of the latency spikes was found to be in the uplink of the UEs. UDP test traffic was set up between the 10 UEs and the NUC running the core network, and as can be seen in Figure 4.17 and 4.18 are the latency spikes not present in the downlink stream. Test traffic was also sent the other way around, from the NUC to the UEs, these also contained latency spikes, verifying that the spikes did not occur between the core and the external machine. Downlink tests from the external computer to the UEs was not conducted because the external computer was not able to reach the private IP address of

the UEs and port forwarding was not set up.

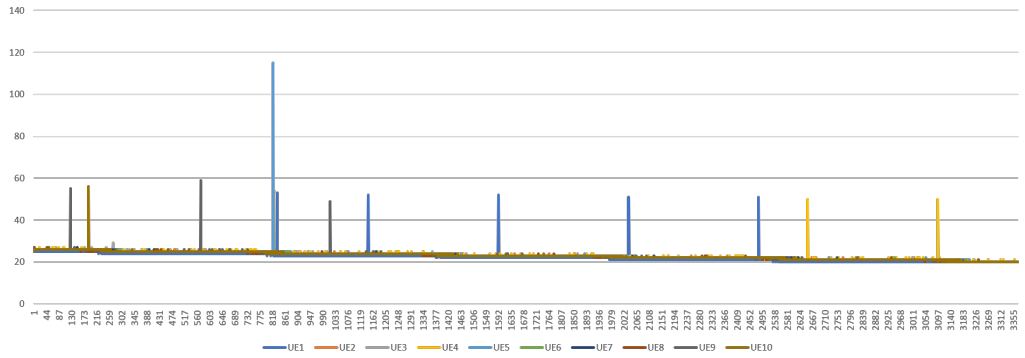


Figure 4.17: Downlink traffic to 10 UEs connected to the MEC core

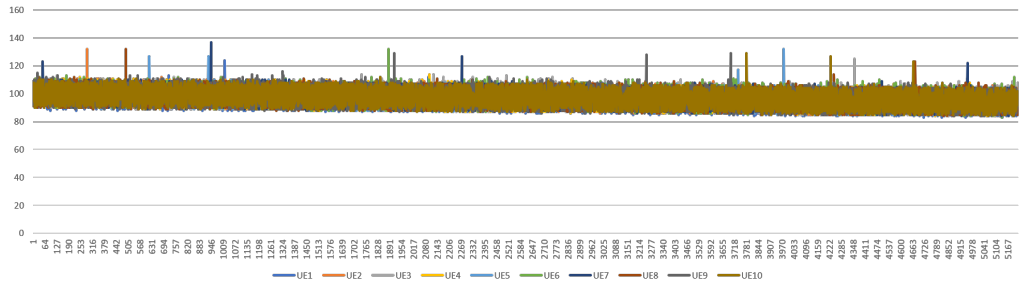


Figure 4.18: Downlink traffic to 10 UEs connected to the cloud core

To confirm that the latency spikes only affect a small part of the transmitted traffic, histograms were made of the latency for each of the UEs, both for the tests when connected to the MEC and the cloud core network. Figure 4.19 is an example of one of these histogram, the figure show the histogram for UE 4 when connected to the cloud. This verifies that the majority of the latency is between 43 and 526ms.

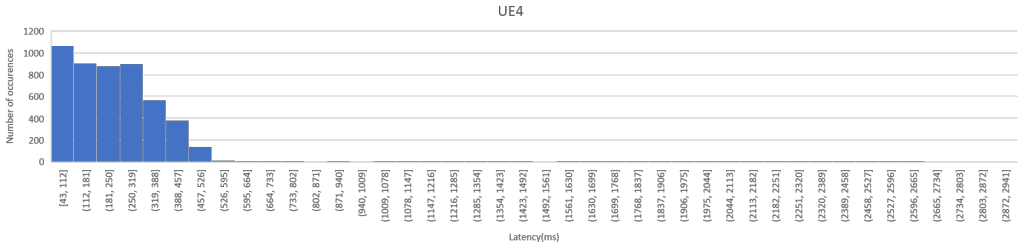


Figure 4.19: Histogram of latency for UE4 when connected to the cloud core network

The next test scenario consisted of near real-time traffic; this was conducted in two different ways, one where an external computer was set up to stream a MP3 audio file and one

where it was streaming a video, both using VLC Media Player [35]. Both streams were set up from the external computer sending the audio and video via Real-time Transport Protocol (RTP) over Real-Time Publish-Subscribe (RTPS). RTPS is a protocol designed to run over an unreliable IP/UDP connection, and it makes it possible for multiple subscribers to stream the audio or video simultaneously using the IP address, port and path set up by the publisher and it uses RTP to transfer the data stream [36]. This test was conducted when having the UE connected to both core networks.

For these test the QoE is used to measure and compare the quality of the channel, the streams were captured in Wireshark at both ends, but as VLC buffers the audio stream, the delay does not play a crucial part and does not affect the experience of the user. Therefore no technical evaluation was made from these tests.

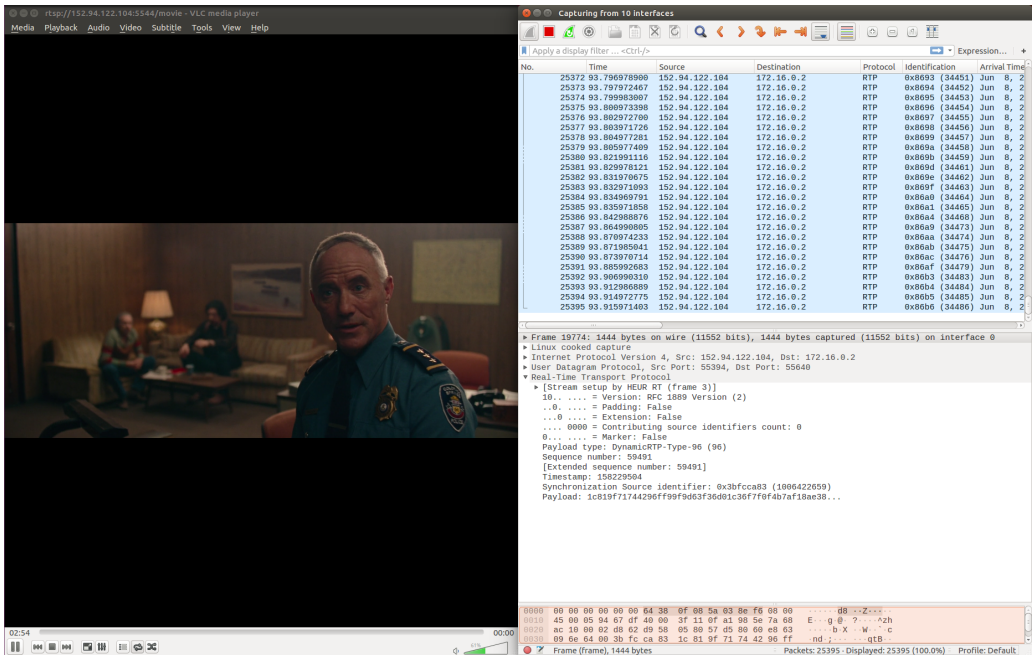


Figure 4.20: Video stream with UE connected to the MEC core network

The experience of the audio quality was not noticeably different when connected to either of the core networks, however, when streaming video there was. Figure 4.20 shows a screenshot of the video stream at the UE when connected to the MEC core. The video quality was good, and there were only short disruptions of the stream when the application was buffering, however, throughout the testing, this did not happen often, and there was minimal disruption of the stream.

Figure 4.21 shows a screenshot from the UE when it is connected to the cloud core. Here one can see that the picture is quite blurred; this was the general experience at the UE when connected to the cloud core. The pictures kept blurring into each other from one

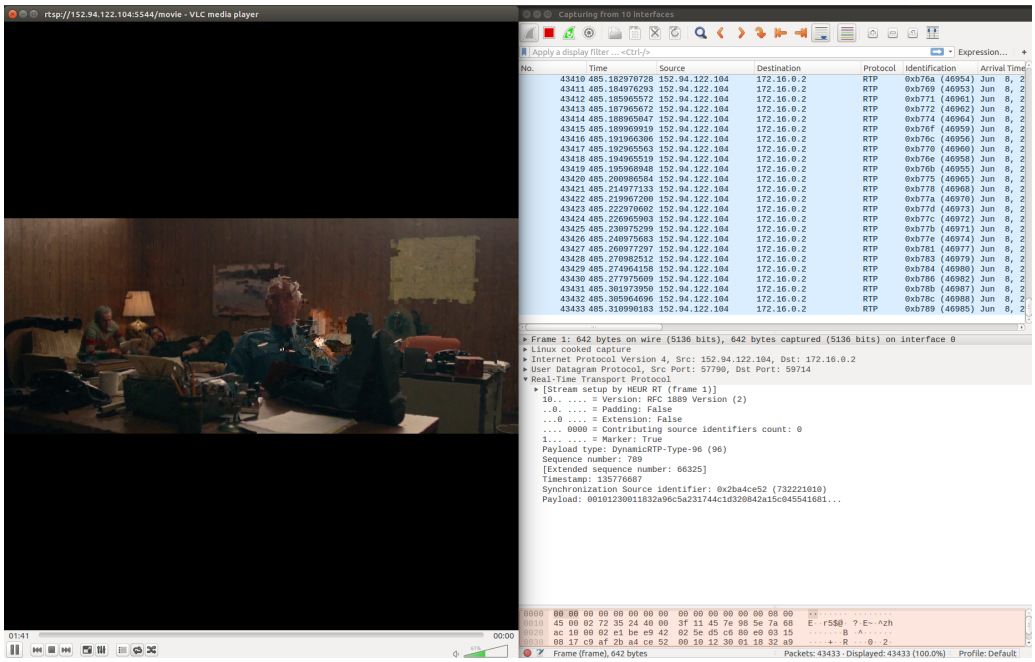


Figure 4.21: Video stream with UE connected to the cloud core network

scene to the next, and some times not loading at all, showing only a black screen and playing the audio.

The audio quality of the video was good both when connected to the MEC and the cloud core, however, this made the experience at the UE almost worse, hearing the video stream continuing without the picture moving accordingly.

These simulations does not only illustrate the scenario of having the core network further away in a cloud environment, but it could also illustrates a scenario where there is one core network in the system and multiple base stations, or multiple cores, serving a set of base stations in a large geographical area. This would cause the delay from the core network to the different base stations to vary depending on their relative distance to the core network. Making the experience for the end user different depending on which base station it is connected to.

With this setup, it was possible to achieve the same maximum bandwidth even when connected to the cloud core network, showing that there are enough resources to keep a service of this size running even if the link to the core is further away. However, the experienced latency for the users is a lot higher when the delay between the base station and the core network is increased. This means that to meet the requirement for uRLLC the delay between the eNB and the core network plays a crucial role and in the most time-critical operations will the core network have to be located relatively close to the

base station, i.e. by the use of Multi-access Edge Computing (MEC),

4.2.1 Test Results with a Production LTE network

As mentioned in the previous chapter, Tampnet has two mobile core sites in its network. The first tests were conducted while having the eNB connected to the core network located relatively close to the eNB. Afterward, the SIM was moved to the core network located further away, and the same tests were conducted there. The tests conducted on the production LTE network consisted of using the traffic generator to generate different amounts of traffic on the link, starting at 2Mbps and increasing with 2Mbps and going up to 14Mbps.

Figure 4.22 depicts the latency in correlation with the throughput, with the throughput displayed as increasing on the X-axis and the average latency from each test at the Y-axis. Here one can see that the latency is not affected by the increase in throughput and stays almost constant with only small variations as the throughput is increased.

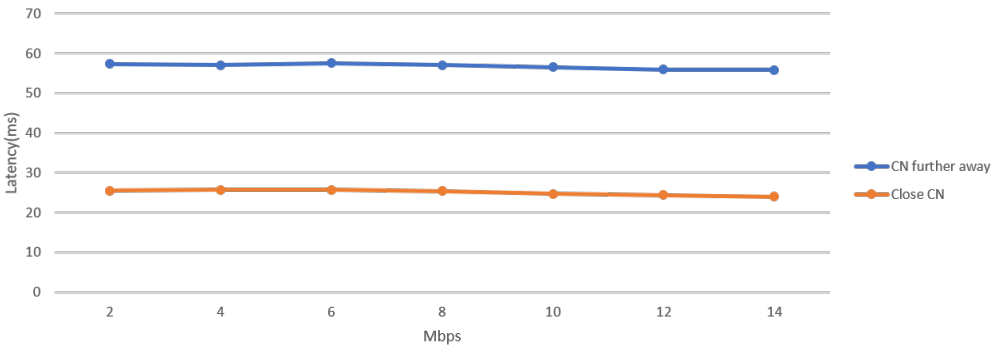


Figure 4.22: Latency in milliseconds at Tampnet's lab

4.3 Discussion

Looking at the results presented in this chapter, findings show how the users would experience the delay when having the base station at different distances, i.e., with different varying delay to the core network. The QoE for users would be having applications loading slower and lagging when streaming videos or playing online games. For the user in the category of eMBB, the delay and jitter might not affect them that much in day-to-day use and only be noticeable in some very network consuming applications. However, 5G is aiming to support other types of industries as well, like the health industry, the oil- and gas industry and autonomous cars. In these types of applications low-latency and

ultra-reliable connections are crucial, and the extra latency added because of the distance between the base station and the core network is not acceptable.

The latency experienced at the UE when testing in the lab environment at Tampnet's offices show a much lower latency than the one obtained at the simulated test-bed at the University. Even in the scenario where only one UE was connected are the experienced latency higher, having the lowest average of 66ms in the simulated test-bed versus 25ms in the tests at Tampnet's lab. This shows that even though the simulated test-bed is functional and can be used for experimental testing is there still a major difference when compared to a production LTE network.

Chapter 5

Conclusion and Future Work

The main objective of this thesis was to study readily available LTE/5G software-defined frameworks with particular focus on simulation/emulation. This has been completed, and an operational test-bed for emulating low-cost LTE/5G has been implemented. This has been conducted using only two Intel NUC machines connected two each other via Ethernet and one being connected to the Internet via Ethernet. That makes the proposed solution in this thesis very cost-effective, as not a lot of expensive equipment is needed to deploy and experiment with it.

Using software enablement and virtualization to experiment and eventually deploy the next generation of the mobile network could in the long term be cost efficient for the service providers but this will first require restructuring of the network. There are many business concerns that needs to be evaluated once the roll-out of 5G begins. Having a network with one or many edge cores, compared to having it in a cloud environment is more expense but gives the service provider more control as well as providing a more stable and reliable connection to the base station.

From an operational point of view is there multiple concerns regarding having the core network in a cloud environment. Since the connection to the core network goes via more nodes to reach the base stations there is a higher risk of unforeseen events, out of the control of the service provider. On the other hand is there a lot of cost involved with having multiple edge cores placed out on large geographical areas.

For future work would it be interesting to test more with the LimeSDRs or similar SDRs, either by using one SDRs as the base station and have a phone connect to the network or by using coaxial cables between the two SDRs and have one of them run as the UE. The test-bed experimented with in this thesis should make it possible to extend the solution to this. The next thing that could be experimented further with is the connection to

the cloud, try a different type of setup, where easier access/connection between the node running the base station and the cloud instance running the core network would make the base station able to connect and bring up the S1 tunnel.

List of Figures

Bibliography

[1] electronics notes. 3gpp Releases | Specification Release Numbers | Electronics Notes.

[2] tutorialspoint.com. LTE Network Architecture. https://www.tutorialspoint.com/lte/lte_network_architecture.htm.

[3] Elizabeth Woyke. China is racing ahead in 5g. Here's what that means., December 2018.

[4] Åshild Breian. Huawei tilbyr «No-spy-agreement» dersom de får bygge ut 5g-nettet i Norge, May 2019.

[5] Paul Shepherd. Discover 5g Core Network Functions compared to 4g LTE | LinkedIn. https://www.linkedin.com/pulse/discover-5g-core-network-functions-compared-4g-lte-paul-shepherd/, September 2018.

[6] Phanidra Palagummi, Vedant Somani, Krishna M. Sivalingam-IIT Madras; Balaji Venkat-Independent Consultant, and Chennai. An overview of the 5g mobile network architecture – Page 3 – A Quarterly Publication of ACCS. https://acc.digital/an-overview-of-the-5g-mobile-network-architecture/3/, August 2018.

[7] Balazs Bertenyi, Chairman of 3GPP RAN, Hungary, Richard Burbidge, Senior Wireless Systems Architect at Intel Corporation, Shrivenham, Oxfordshire, UK, Gino Masini, Systems Manager, Concepts and Standards, at LM Ericsson AB, Stockholm, Sweden, Sasha Sirotkin, Vice-Chair of 3GPP RAN3 at Intel Corporation, Israel, Yin Gao, and Vice Chairman 3GPP, ZTE Corporation, China. NG Radio Access Network (NG-RAN). *Journal of ICT Standardization*, 6(1):59–76, 2018.

[8] Citrix Systems, Inc. What is Software-defined Networking? https://www.citrix.com/products/citrix-adc/resources/sdn-101.html.

[9] Margaret Rouse. What is software-defined networking (SDN)? - Definition from WhatIs.com. https://searchnetworking.techtarget.com/definition/software-defined-networking-SDN.

[10] Margaret Rouse. What is network functions virtualization (NFV)? - Definition from WhatIs.com. https://searchnetworking.techtarget.com/definition/network-functions-virtualization-NFV.

[11] EventHelix. Ultra-Reliable Low-Latency Communication (URLLC). https://medium.com/5g-nr/ultra-reliable-low-latency-communication-urllc-9b2505e81579, May 2018.

[12] sdx central. What is 5g Network Slicing? A Definition — SDxCentral.com. https://www.sdxcentral.com/5g/definitions/5g-network-slicing/.

[13] T. Li, L. Zhao, F. Song, and C. Pan. OAI-based End-to-End Network Slicing. In *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*, pages 1–4, November 2018.

[14] S. Costanzo, I. Fajjari, N. Aitsaadi, and R. Langar. A network slicing prototype for a flexible cloud radio access network. In *2018 15th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 1–4, January 2018.

[15] Openairinterface 5G. oai / openairinterface5g. https://gitlab.eurecom.fr/oai/openairinterface5g.

[16] Contributeto{OPENAIRINTERFACE}/openair-cndevelopmentbycreatinganaccounton{GitHub}, April 2019. original-date: 2017-09-08T14:46:56Z.

[17] University of Edinburgh, Eurecom, and NCSR "Demokritos". FlexRAN — A Flexible, Programmable and Open-Source SD-RAN Platform. https://networks.inf.ed.ac.uk/flexran/.

[18] G. Garcia-Aviles, M. Gramaglia, P. Serrano, and A. Banchs. POSENS: A Practical Open Source Solution for End-to-End Network Slicing. *IEEE Wireless Communications*, 25(5):30–37, October 2018.

[19] Software Radio Systems. Open source SDR LTE software suite from Software Radio Systems (SRS): srsLTE/srsLTE. https://github.com/srsLTE/srsLTE, May 2019. original-date: 2013-12-06T13:53:04Z.

[20] OpenAirInterface. Towards open cellular ecosystem. https://www.openairinterface.org/?page_id=864.

[21] Lime Microsystems Ltd. LimeSDR. https://limemicro.com/products/boards/limesdr/.

[22] OpenAirInterface. Openairinterface system emulation (under further updates). https://gitlab.eurecom.fr/oai/openairinterface5g/wikis/OpenAirLTEEmulation.

[23] MYRIAD RF. Lime Suite - Myriad-RF Wiki. https://wiki.myriadrf.org/Lime_Suite.

[24] MYRIAD RF. LimeSDR-USB Quick Test - Myriad-RF Wiki. https://wiki.myriadrf.org/LimeSDR-USB_Quick_Test.

[25] Openairinterface 5G. OpenAirLTEEmulationNEW · Wiki · oai / openairinterface5g. https://gitlab.eurecom.fr/oai/openairinterface5g/wikis/OpenAirLTEEmulationNEW.

[26] Openairinterface 5G. HowToConnectCOTSUEwithOAIeNBNew · Wiki · oai / openairinterface5g. https://gitlab.eurecom.fr/oai/openairinterface5g/wikis/HowToConnectCOTSUEwithOAIeNBNew.

[27] Openairinterface 5G. l2 nfapi simulator w S1 same machine · Wiki · oai / openairinterface5g. https://gitlab.eurecom.fr/oai/openairinterface5g/wikis/l2-nfapi-simulator/l2-nfapi-simulator-w-S1-same-machine.

[28] An open source implementation of the Small Cell Forum's Network Functional API (nFAPI): cisco/open-nFAPI. https://github.com/cisco/open-nFAPI, March 2019. original-date: 2017-02-10T20:25:14Z.

[29] Florian Kaltenberger. OSA EURECOM KALTENBERGER. https://docbox.etsi.org/Workshop/2018/201812_ETSI_OAI/TRAINING/OSA_EURECOM_KALTENBERGER.pdf, December 2018.

[30] Justin Ellingwood. How To Set Up an OpenVPN Server on Ubuntu 16.04. https://www.digitalocean.com/community/tutorials/how-to-set-up-an-openvpn-server-on-ubuntu-16-04.

[31] OpenAirInterface. s1 flex conf nnsf · Wiki · oai / openairinterface5g. https://gitlab.eurecom.fr/oai/openairinterface5g/wikis/s1-flex-conf-nnsf.

[32] NLANR/DAST : Iperf - The TCP/UDP Bandwidth Measurement Tool. https://web.archive.org/web/20081012013349/http://dast.nlanr.net/Projects/Iperf/, October 2008.

[33] Wireshark Foundation. Wireshark · Go Deep. https://www.wireshark.org/.

[34] mlouielu. networking:netem [Wiki]. https://wiki.linuxfoundation.org/networking/netem, September 2018.

[35] VideoLAN. Official download of VLC media player, the best Open Source player - VideoLAN. https://www.videolan.org/vlc/index.nb.html.

[36] Margaret Rouse. What is Real-Time Transport Protocol (RTP)? - Definition from WhatIs.com. https://searchnetworking.techtarget.com/definition/Real-Time-Transport-Protocol, April 2007.

Appendices

Appendix A

Configuration guide and test data

Configuration files, test data and C# program used in this thesis can be found at https://github.com/mari-h/ConfigMasterThesis.git.

The functional test-bed set up and used for evaluation in this thesis was deployed on two Intel NUC computers running Ubuntu 16.04 with low-latency kernel (4.13.0-36-lowlatency).

Steps needed before stating the implementation of the OAI EPC, eNB and UE:

```
$ sudo apt-get update
$ sudo apt-get dist upgrade
$ sudo apt-get install linux-image-4.13.0-36-lowlatency
linux-headers-4.13.0-36-lowlatency
$ sudo apt-get install git
```

After updating the kernel the computer needs to be restarted. Upon restart press *Esc* to enter the *GNU GRUB* menu, choose *Advanced options for Ubuntu* and choose the correct kernel distribution. This can be set as default in the */etc/default/grub* file, otherwise it needs to be done for every restart of the computer.

A.1 Configuration of the OAI EPC

The steps below below are based on [26] but modified according to functional setup in this thesis.

Make sure the hostname of the computer is correctly set, *HOSTNAME* in the below commands should be replaced with the hostname of the computer. *openair4G.eur* can be replace by something else but this needs to be reflected in the configuration of the EPC:

```
$ cat /etc/hostname
HOSTNAME
$ cat /etc/hosts
127.0.0.1     localhost
127.0.1.1     HOSTNAME.openair4G.eur    HOSTNAME
127.0.1.1     hss.openair4G.eur     hss
$ hostname
HOSTNAME
$ hostname -f
HOSTNAME.openair4G.eur
```

Make sure the above is correct before proceeding.

```
$ git clone https://github.com/OPENAIRINTERFACE/openair-cn.git
$ cd openair-cn
$ git checkout v0.5.0
$ git branch
* (HEAD detached at v0.5.0)
$ git pull
$ source oaienv
$ cd scripts
#Need to run these only once to install missing packages
$ ./build_mme -i
$ ./build_hss -i
$ ./build_spgw -i
```

Copy the EPC config files in /usr/local/etc/oai

```
$ sudo mkdir -p /usr/local/etc/oai/freeDiameter
$ sudo cp ~/openair-cn/etc/mme.conf /usr/local/etc/oai
$ sudo cp ~/openair-cn/etc/hss.conf /usr/local/etc/oai
$ sudo cp ~/openair-cn/etc/spgw.conf /usr/local/etc/oai
$ sudo cp ~/openair-cn/etc/acl.conf /usr/local/etc/oai/freeDiameter
$ sudo cp ~/openair-cn/etc/mme_fd.conf /usr/local/etc/oai/freeDiameter
$ sudo cp ~/openair-cn/etc/hss_fd.conf /usr/local/etc/oai/freeDiameter
```

The configurations files for the MME, HSS and SPGW needs to be modified, according to the network setup used. the configuration files used in this thesis are attached to the pdf version of this document.

The following files should be modified:

```
/usr/local/etc/oai/mme.conf
/usr/local/etc/oai/spgw.conf
/usr/local/etc/oai/freeDiameter/hss_fd.conf
/usr/local/etc/oai/freeDiameter/mme_fd.conf
/usr/local/etc/oai/hss.conf
```

Once these files have been modified correctly, proceed to install certificates:

```
cd ~/openair-cn/scripts
./check_hss_s6a_certificate /usr/local/etc/oai/freeDiameter/ hss.openair4G.eur
#Note that HOSTNAME should be replaced with the hostname of the computer
./check_mme_s6a_certificate /usr/local/etc/oai/freeDiameter/ HOSTNAME.openair4G.eur
```

Compile and Run HSS (ALWAYS RUN HSS FIRST):

```
cd ~/openair-cn
cd scripts
./build_hss -c
#After installing the databse the hostname of the computer need to be added
#to the HSS database for it to accept the connection from the MME
./run_hss -i ~/openair-cn/src/oai_hss/db/oai_db.sql #Run only once to install database
./run_hss  #Run this for all subsequent runs
```

Compile and Run MME:

```
cd ~/openair-cn/scrpis
./build_mme -c
./run_mme
```

Compile and Run SP-GW:

```
cd ~/openair-cn/scripts
./build_spgw -c
./run_spgw
```

Once MEE is connected to HSS you should see STATE_OPEN in the terminal for both the MME and HSS.

A.2 Configuration of eNB and UE using nFAPI L2 simulator

The steps below are from [27]. This configuration is verified by having the OAI EPC run on one computer and the eNB and UE on a second computer (UE and eNB on same computer).

Retrieve the OAI eNB and UE source code:

```
$ git clone https://gitlab.eurecom.fr/oai/openairinterface5g/ enb_folder
$ cd enb_folder
$ git checkout -f v1.0.0
$ cd ..
$ cp -Rf enb_folder ue_folder
```

Setup of the USIM information in UE folder:

```
$ cd ue_folder
# Edit openair3/NAS/TOOLS/ue_eurecom_test_sfr.conf with your preferred editor
```

Edit the USIM information within this file in order to match the HSS database. They HAVE TO match:

- PLMN+MSIN and IMSI of users table of HSS database SHALL be the same.
- OPC of this file and OPC of users table of HSS database SHALL be the same.
- USIM_API_K of this file and the key of users table of HSS database SHALL be the same.

When testing multiple UEs, it is necessary to add the information for all UEs. Only UE0 (first UE) information is written in the original file. The SIM information for all UEs need to be added to the HSS database.

The eNB Configuration file:

```
$ cd enb_folder
# Edit ci-scripts/conf_files/rcc.band7.tm1.nfapi.conf with your preferred editor
```

Edit the *mme_ip_address*, *NETWORK_INTERFACES* and *MACRLCs* to match your network setup. The *MACRLCs* should be the IP address towards the UE, in this thesis (and in the OAI guide) the loopback interface is used, lo: (127.0.0.2) for the eNB and lo (127.0.0.1) for the UE.

The UE Configuration file:

```
$ cd ue_folder
# Edit ci-scripts/conf_files/ue.nfapi.conf with your preferred editor
```

If lo: is used by the eNB; bring up a second loopback interface:

```
$ sudo ifconfig lo: 127.0.0.2 netmask 255.0.0.0 up
```

Build the eNB:

```
$ cd enb_folder
$ source oaienv
$ cd cmake_targets
$ ./build_oai --eNB -t ETHERNET -c
```

Build the UE:

```
$ cd ue_folder
$ source oaienv
$ cd cmake_targets
$ ./build_oai --UE -t ETHERNET -c
```

After finishing building UE(s), some files are generated in ue_folder/targets/bin/ and these files are necessary in cmake_targets:

```
$ cd ue_folder/targets/bin/
$ cp .u* ../../cmake_targets/
$ cp usim ../../cmake_targets/
$ cp nvram ../../cmake_targets/
```

Initialize the NAS UE Layer:

```
$ cd ue_folder/cmake_targets/tools
$ source init_nas_s1 UE
```

Initialize the NAS UE Layer:

```
$ cd ue_folder/cmake_targets/tools
$ source init_nas_s1 UE
```

After completing the above steps the eNB and UE are ready to be run. To test the setup, now start the EPC on the other computer. Once the EPC is up and running, and the HSS and MME is connected the eNB and UE can be started.

Start the eNB:

```
$ cd enb_folder/cmake_targets
$ sudo -E ./lte_build_oai/build/lte-softmodem
-O ../ci-scripts/conf_files/rcc.band7.tm1.nfapi.conf > enb.log 2>&1
```

Start the UE:

```
# Test 10 UEs, 10 threads in FDD mode
$ sudo -E ./lte_build_oai/build/lte-uesoftmodem
-O ../ci-scripts/conf_files/ue.nfapi.conf
--L2-emul 3 --num-ues 10 --nums_ue_thread 10 > ue.log 2>&1
# Note: found that when running with more than 3 UEs
# it worked best without the --num_ue_thread attribute
```

A.3 Configuration of S1-flex

To make the eNB connect to more than one EPC S1-Flex was used in this thesis. The steps below are from the OAI's guide [31]. With this the eNB successfully connected to two CNs, however, in this thesis were we not able to make UEs connect to the different CNs simultaneously. The eNB only sent S1AP/NAS-EPS messages (requesting the UE to connect to the EPC) to one CN, the one defined first in the configuration file, even if the PLMNs were different on the two CNs and one(or more) UE was configured to connect to each of them. The eNB sends these messages to the first one defined in the configuration file if it is operational, if only the second CN is running it will send the attache request to the second defined CN.

Add PLMN and MME IP address of all CNs you want the eNB to connect to:

```
# Open the eNB configuration file:
# ci-scripts/conf_files/rcc.band7.tm1.nfapi.conf

tracking_area_code = 1;
plmn_list = (
  { mcc = 208; mnc = 93; mnc_length = 2; },
  { mcc = 208; mnc = 95; mnc_length = 2; }
)
```

```
mme_ip_address = (
  {
    ipv4       = "192.168.12.240";
    ipv6       = "192:168:30::17";
    active     = "yes";
    preference = "ipv4";
  },
  {
    // ... second MME ...
  }
);
```

An additional broadcast_plmn_index parameter has been added to the MME config in the eNB. It permits to define the Global eNodeB ID as exposed to the MME in S1SetupRequest as well as defining which broadcasted PLMNs are communicated. Assuming the above given plmn_list with two Broadcast PLMNs, the following example defines that the eNodeB communicates the first PLMN both as ID and its only PLMN tracking area to the first MME ([0]), whereas for the second MME, the second PLMN is used as ID and both the first and second PLMN ([0,1]) are announced in its tracking area.

Add broadcast_plmn_index parameter to the MME config in the eNB configuration file:

```
# Still in the eNB configuration file:
# ci-scripts/conf_files/rcc.band7.tm1.nfapi.conf

mme_ip_address = (
  {
    // IP address etc here
    broadcast_plmn_index = [0];
  },
  {
    // IP address etc here
    broadcast_plmn_index = [0,1];
  }
);
```

If broadcast_plmn_index = [1] was given for the second MME, the second PLMN ID would be used for both Global eNodeB ID and PLMN tracking area. Thus, no MME would know that this eNodeB operates for different CNs. Leaving out this parameter assumes a default of [0,...,N-1] for all MMEs with N being the number of PLMN IDs in

the plmn_list.