UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Design and Implementation of a Dynamic Autoscaling Mechanism for the User-plane Services of the 5G Core Network

Diploma Thesis

Sokratis Christakis

Supervisor: Athanasios Korakis

September 2022

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Design and Implementation of a Dynamic Autoscaling Mechanism for the User-plane Services of the 5G Core Network

Diploma Thesis

Sokratis Christakis

Supervisor: Athanasios Korakis

September 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Σχεδιασμός και Υλοποίηση Μηχανισμού Δυναμικής Αναπροσαρμογής αντιγράφων Υπηρεσιών δεδομένων του 5G Core Δικτύου

Διπλωματική Εργασία

Σωκράτης Χριστάκης

Επιβλέπων: Αθανάσιος Κοράκης

Σεπτέμβριος 2022

Approved by the Examination Committee:

Supervisor **Athanasios Korakis**
Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Dimitrios Bargiotas**
Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Antonios Argyriou**
Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Acknowledgements

Upon completion of my studies and my thesis, I would like to express my deepest gratitude to all those who helped me throughout this 5-year journey.

First and foremost, I would like to thank my thesis supervisor Professor Athanasios Korakis, for his guidance and support throughout the duration of my studies. Most importantly, I would like to thank him for the opportunity he gave me to join NITlab and cooperate with such an inspiring and advanced research group. Additionally, I would like to deeply thank Professor Dimitrios Brgiotas and Associate Professor Antonios Argyriou, who were members of the Examination Committee and approved this Thesis.

I would also like to thank the Senior Researcher and Postdoctoral Nikolaos Makris, for always being available for me and always steering me to the right direction with his scientific guidance. His help was vital for the outcome of this thesis.

Last but not least, I would like to thank my family, from the bottom of my heart, for the continuous emotional and financial support they gave me throughout these years.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Sokratis Christakis

Diploma Thesis

Design and Implementation of a Dynamic Autoscaling Mechanism for the User-plane Services of the 5G Core Network

Sokratis Christakis

Abstract

The advent of new technologies such as the interconnection of systems and devices (Internet of Things), ”smart” cities and homes, autonomous cars, Virtual (VR) and Augmented (AR) reality applications, high-resolution video (4K/8K) and many others, have forced the scientific community to find new ways to respond to the extreme amount of data that has to be transferred in wireless networks. The Fifth Generation (5G) of cellular networks is considered to give answers to these problems by using innovative technologies and ideas.

5G networks form the next generation of wireless communications, offering faster speeds and more reliable connections to smartphones and many other devices. This new generation of networks promises response times of 1 millisecond, with simultaneous communication reliability of 99.9% , making any delay impossible to perceive. According to the latest research, the 5G network will be able to offer much faster connections than existing connections, with average download speeds reaching 1GBps. 5G is intended for conditions in which tens of billions of devices depend on their continuous internet connection. 5G, through advanced technologies, will support new transmission frequencies that have not been used in the past, as well as support even more simultaneous users and devices, providing them with speeds much higher than those of 4G.

In this thesis we use the NITOS Testbed experimental facilities, as well as the OpenAir-Interface open-source platform, which implements 5G networks using software. The purpose is to deploy the OpenAirInterface platform in a microservices environment, and to dynamically scale specific network functions in order to meet the network demand dynamically. To achieve this, Docker and Kubernetes technologies are used. Docker enables us to create containers, which in our case perform the functions of the core network. Then, we utilize Kubernetes to manage these containers (and therefore the network) in the best way possible. Finally, we use this distributed network to implement new capabilities that allow scaling of the functions of the core network. These extensions regard the operation of specific components

of the Kubernetes framework (Horizontal Pod Autoscaler) as well as the communication substrate between the core and the radio access network in the OpenAirInterface platform. This scaling creates multiple copies of some services and does load balancing between them, based on some policy-metrics.

Διπλωματική Εργασία

Σχεδιασμός και Υλοποίηση Μηχανισμού Δυναμικής Αναπροσαρμογής αντιγράφων Υπηρεσιών δεδομένων του 5G Core Δικτύου

Σωκράτης Χριστάκης

Περίληψη

Η έλευση νέων τεχνολογιών όπως είναι η διασύνδεση συστημάτων και συσκευών (Internet of Things), ”έξυπνες” πόλεις και σπίτια, αυτόνομα αυτοκίνητα, εφαρμογές εικονικής πραγματικότητας, βίντεο υψηλής ανάλυσης και πολλά άλλα, ανάγκασαν την επιστημονική κοινότητα να βρεί νέους τρόπους, ώστε να ανταποκριθεί στον υπερπολλαπλάσιο μεταφερόμενο όγκο δεδομένων στα ασύρματα δίκτυα. Τη λύση στο πρόβλημα αυτο καλούνται να δώσουν τα δίκτυα 5ης Γενιάς (5G).

Τα δίκτυα 5G αποτελούν την επόμενη γενιά σύνδεσης ασύρματων συσκευών στο διαδίκτυο, προσφέροντας μεγαλύτερες ταχύτητες αλλά και πιο αξιόπιστες συνδέσεις σε smartphones και άλλες συσκευές. Η νεα αυτή γενια δικτύων υπόσχεται χρόνους απόκρισης στο 1 χιλιοστό του δευτερολέπτου, με ταυτόχρονη αξιοπιστία επικοινωνίας στο 99,9% καθιστώντας την όποια καθυστέρηση, αδύνατο να αντιληφθεί. Σύμφωνα με τις τελευταίες έρευνες, το 5G δίκτυο θα μπορεί να προσφέρει συνδέσεις πολύ πιο γρήγορες από τις ηδη υπάρχουσες συνδέσεις, με τις μέσες ταχύτητες λήψης να αγγίζουν το 1GBps. Το 5G προορίζεται για συνθήκες στις οποίες δεκάδες δισεκατομμύρια συσκευές εξαρτώνται από τη συνεχή σύνδεση τους στο διαδίκτυο. Το 5G μέσω ανεπτυγμένων τεχνολογιών, θα υποστηρίζει νέες συχνότητες μετάδοσης που δεν έχουν χρισημοποιηθεί στο παρελθόν, καθώς επίσης θα υποστηρίζει ακόμα περισσότερους ταυτόχρονους χρήστες και συσκευές, παρέχοντας τους τους ταχύτητες πολύ υψηλότερες από αυτές του 4G.

Σε αυτή τη διατριβή χρησιμοποιούμε τις πειραματικές εγκαταστάσεις του Nitos Testbed, καθώς και την πλατφόρμα ανοιχτού κώδικα OpenAirInterface, η οποία υλοποιεί 5G δίκτυα με την χρήση λογισμικού. Σκοπός είναι να γίνει deploy η πλατφόρμα OpenAirInterface σε περιβάλλον microservices και να κλιμακώσει δυναμικά συγκεκριμένες λειτουργίες του κεντικού δικτύου προκειμένου να διαχειριστεί δυναμικά τις απαιτήσεις του δικτύου. Για να επιτευχθεί αυτό, χρησιμοποείται το λογισμικό Docker, το οποίο δίνει την δυνατότητα να δημιουργήσουμε περιέκτες (containers), οι οποίοι στην περίπτωση μας εκτελούν τις λειτουργίες του

κεντρικού δικτύου. Στη συνέχεια εισάγουμε το λογισμικό Kubernetes για να διαχειρίζεται τους περιέκτες (και συνεπώς το δίκτυο) με τον καλύτερο δυνατό τρόπο. Τέλος, υλοποιούνται πάνω σε αυτό δυνατότητες που χρειάζονται προκειμένου να επιτευχθεί scaling των υπηρεσίων του κεντρικου δικτύου (Core Network). Αυτές οι επεκτάσεις αφορούν τη λειτουργία συγκεκριμένων στοιχείων του λογισμικού Kubernetes (Horizontal Pod Autoscaler) καθώς και τη διεπαφή επικοινωνίας μεταξύ του κεντρικού δικτύου και του δικτύου ασύρματης πρόσβασης, στην πλατφόρμα OpenAirInterface. Το scaling αυτό θα δημιουργεί πολλαπλά αντίγραφα απο κάποια services και θα κάνει load balancing ανάμεσα σε αυτά με βάση κάποιες πολιτικές-μετρικές.

Table of contents

List of figures

Abbreviations

3GPP	3rd Generation Partnership Project
3G	3rd Generation
4G	4th Generation
5G NR	5G New Radio
5G	5th Generation
AF	Application Function
AKA	Authentication and Key Agreement
AMF	Access and Mobility Management Function
API	Application Program Interface
APN	Access Point Name
AUSF	Authentication Server Function
BSF	Binding Support Function
CLI	Command Line
CNAME	Canonical Name
CNI	Container Network Interface
CUPS	Control and User Plane Separation
DN	Data Networks
EDGE	Enhanced Data rates for GSM Evolution
EPC	Evolved Packet Core
EPS	Evolved Packet System
EUTRAN	Evolved Universal Terrestrial Radio Access Network
FDMA	Frequency-Division Multiple Access
GSM	Global System for Mobile communication
HSDPA	High-Speed Downlink Packet Access
HSuPA	High-Speed Uplink Packet Access

HSPA	High Speed Packet Access
HSS	Home Subscriber Server
HSS	Radio Access Network
HTTP	Hypertext Transfer Protocol
IMSI	International Mobile Subscriber Identity
IP	Internet Protocol
ITU-R	International Telecommunication Union
LAN	Local Area Network
LTE-A	Long Term Evolution Advanced
LTE	Long Term Evolution
MAC	Media Access Control
MIMO	Multiple Input Multiple Output
MME	Mobility Management Entity
MSISDN	Mobile Station International Subscriber Directory Number
Massive MIMO	Massive Multiple Input Multiple Output
NAS	Non-access stratum
NAT	Network Address Translation
NEF	Network Exposure Function
NITOS	Network Implementation using Open-Source software
NRF	Network Repository Function
NSSF	Network Slice Selection Function
OAI	Open Air Interface
OFDMA	Orthogonal Frequency-Division Multiple Access
OMF	Control and Management Framework
OS	Operating System
P-GW	Packet Data Network Gateway
PCF	Policy Control Function
PDCP	Packet Data Convergence Protocol
PDN	Packet Data Network
PHY	Physical Layer
QoS	Quality of Service
RAN	Radio Access Network

RF	Radio Frequency
RLC	Radio Link Contro
RRC	Radio Resource Control
S-GW	Serving Gateway
SBA	Service Based Architecture
SC-FDMA	Single-carrier Frequency-Division Multiple Access
SDR	Software Defined Radio
SMF	Session Management Function
SPGW-C	Control Plane of the Packet Data Network Gateway
SPGW-U	User Plane of the Packet Data Network Gateway
SSH	Secure Shell
TDMA	Time-Division Multiple Access
UDM	Unified Data Management
UE	User Equipment
UMTS	Universal Mobile Telecommunication System
UPF	User Plane Function
USRP	Universal Software Defined Radio Peripherals
VM	Virtual Machine
VR	Virtual Reality
VoIP	Voice over Internet Protocol
WLAN	Wireless Local Area Network
Wi-Fi	Wireless Fidelity
WiMAX	Worldwide Interoperability for Microwave Access
eNB	Evolved Node B
gNB	Next Generation Node B
k8s	Kubernetes
mmWaves	Millimeter Waves

Chapter 1

Introduction

5G is the new generation of mobile technology that greatly improves network performance and capabilities compared to previous technologies (2G, 3G, 4G, 4.5G). It offers nearly zero latency and reliable connectivity for billions of devices. It also gives the ability to support multiple and separate ”slices” of the network at the same time, instead of exclusively support a large network. In other words, it supports multiple, specialized virtual networks at the same time, meaning users have the best possible online experience. In practical terms, this means that the 5G network is able to handle each need that arises individually, whether it is the car’s response to an accident (that requires zero response time), or a live streaming 8K video etc.

5G networks is one of the major reasons why containers (especially Docker-based containers) have exploded in popularity. Practically, the advent of 5G allowed demanding and complicated applications and services to be used widely. This has led to the replacement of traditional VM’s with containers, for the simple reason that virtual machines need to launch an operating system during start up, while containers only need to start a process. In reality, a virtual machine often takes a minute (or more) to start, in contrast with a container that takes milliseconds to start.

However, the use of such an enormous amount of containers, required software for automated management and control. This is where Kubernetes was introduced as a container orchestration tool, to manage complex workloads running on a large number of containers distributed across physical or virtual hosts. It supports the use of containers to organize software into small interconnected services (microservices) and is automated using declarative methods.

1.1 The subject of Thesis

In this diploma, we combine the technologies described in the previous section to deploy a 4G/LTE network on actual machines, using the OAI platform. OAI is an open-source platform, which implements 4G/LTE and 5G networks using software.

To start with, we take advantage of the NITOS Testbed (located in University of Thessaly) and use its advanced infrastructure to form an appropriate environment for the experiments. After that, we obtain software from OAI and deploy a fully fucntional and distributed 4G/LTE network, by using Docker and Kubernetes.

The first challenge is to deploy this 4G network in a microservices environment, using Kubernetes. For this purpose, a Kubernetes cluster needs to be created to manage the resources on NITOS' actual machines. The functions and the components of the core network as well as the eNB are containirized with the use of Docker. Then, they are deployed as pods and assigned to the worker nodes of the cluster by Kubernetes.

Next, we will use this distributed network to implement capabilities that allow scaling of the functions of the core network. Practically, the default setting of OAI's 4G model is one interface between the components of the core network. The goal of this thesis is to create multiple copies of the core network's function SPGW-U and connect each one of them with the base station (eNB) through a new interface. The number of the the new replicas and interfaces created are determined by the amount of traffic that takes place in the network. This implementation serves load-balancing purposes and intends to scale the traffic through the new interfaces, when the network is overloaded.

Finally, we discuss the perspective of extending this implementation to OAI's 5G model, due to the architecture similarities towards the OAI's 4G model.

1.2 Thesis Structure

In Chapter 2 we introduce the concept of wireless networks. In detail, we describe the 4th generation of wireless networks (4G), LTE networks as well as the key technologies and their architecture. Then we analyze the fifth generation of wireless networks (5G), listing the innovative technologies developed for their implementation. In Chapter 3 we present Kubernetes and Docker technologies, which were used for the distribution of the network and were responsible for the management of the resources and functions of the network.

In Chapter 4, we discuss the experimental infrastructure, the experimental setup and the experimental results. Finally, in Chapter 5, we summarize the thesis, discuss the results that took place and suggest some possible future work.

Chapter 2

4G/LTE & 5G Cellular Networks

2.1 Introduction to Cellular Networks

A cellular network [1], also known as a mobile network, is a radio network that is distributed over geographical areas known as cells. Each cell is served by at least one fixed-location transceiver, which represents a base station. To avoid interference and provide assured bandwidth within a cell, each cell in a cellular network uses a separate range of frequencies from nearby cells. Cellular Networks made their first appearance in Japan in 1979 and were supporting only voice calls, while suffering from reliability, poor protection and signal interference issues. Although, the evolution of technology, made it possible for mobile networks to be upgraded, used widely (Figure 2.1) and support high speed social connectivity.

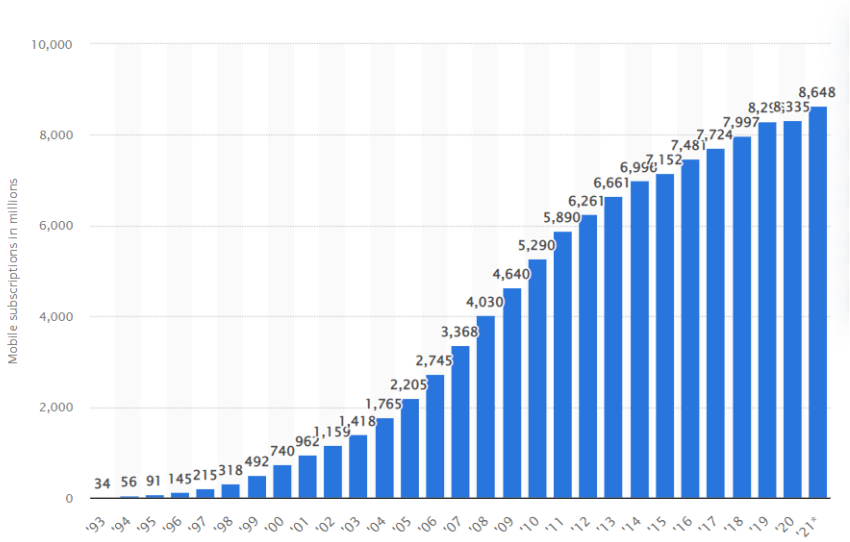

Figure 2.1: Subscribers in millions from 1993 until 2021 [2]

2.2 4G Cellular Networks

2.2.1 Introduction to 4G

4G [3] is the fourth generation of cellular broadband technology and the successor to 3G. In essence, it is an International Telecommunication Union (ITU-R) standard that establishes a type of wireless broadband network that offers high-speed data transfer. It was designed with the purpose of evolving radio access technology so that all services are based on packet switching and not circuit switching. Unlike previous generations, 4G fully supports IP telecommunications protocols such as VoIP. With 4G, speeds are even greater, thanks to the new technologies it uses. 4G technology is based on WiMAX and LTE Advanced technology. WiMAX technology works similarly to Wi-Fi, however, it ensures a communication range of 35 (or more) kilometers as opposed to the 100 meters or so that Wi-Fi ensures. The standards for the 4G network are still evolving, and the maximum speed of 4G can reach 1 Gbps with the LTE-Advanced (LTE-A) standard. However, the actual speed is ranged between 20-100 Mbps for mobile phones. In addition, it gives ability to use the internet connection anywhere within a city or country even if the user is on the move. 4G networks support higher data rates and adequate quality of service (QoS) in comparison with 3G networks. The spectrum allocation is such, that these high rates are supported over medium-sized cells. The networks consist entirely of packet switching circuits, while all the elements of the network are digital. Finally, an important feature of 4G networks is low hierarchy in architecture, with access points and terminals supporting multiple access modes.
To achieve higher performance, 4G networks have the following characteristics:

- **Ubiquitous service coverage**

  The user is able to use any service, anywhere and anytime, without being hindered by the limitations inherent in a wireless network.

- **Improved connectivity**

  The user is always connected to the most adequate access networks to ensure the requested quality of service (QoS) and mobility requirements every time.

- **Continuous connection**

  The user is always connected to the heterogeneous network and experiences very little access delay.

2.2.2 4G Key Technologies

- **OFDMA**

  OFDMA [4] is the multi-user OFDM technology where users can be assigned on both a TDMA and FDMA basis where a single user does not necessarily need to occupy all sub-carriers at all times. OFDMA technology applies time and frequency separation into OFDM sub-carriers. These sub-carriers can be assigned to different users. This assignment gives the ability of exploiting the diversity of many users (Multiuser Diversity). This differentiation can be done either in time or in frequency or in both, providing greater freedom to the partitioning algorithm that manages available network resources. The main feature of OFDMA is that a user can be assigned one or more sub-carriers, depending on the conditions and the movement the user holds in the network. This allows simultaneous low data rate transmission by multiple users, as it can be dynamically assigned to the best non-fading low-interference channels for a given user and avoid assigning bad sub-carriers (Figure 2.2). Fixed and mobile Point-to-Multipoint systems use OFDMA, including Mobile WiMAX and LTE.

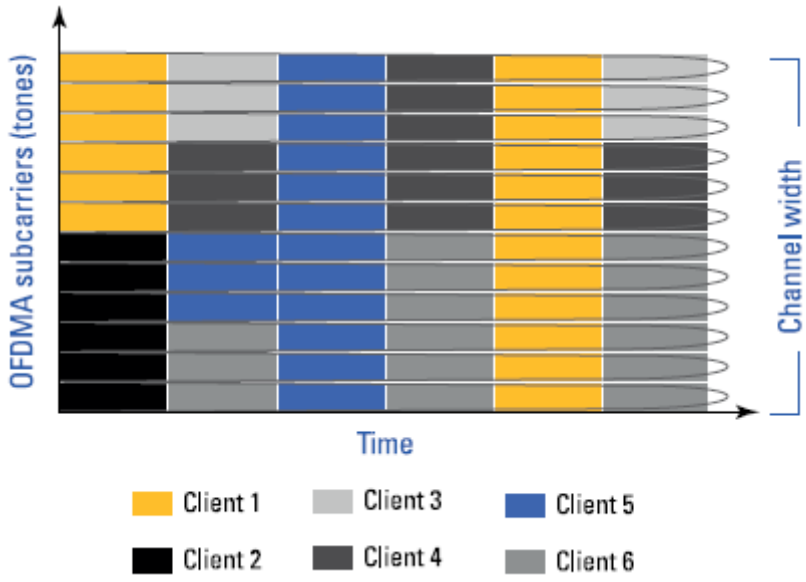

Figure 2.2: OFDMA Client Management [5]

- **MIMO (Multiple Input Multiple Output)**

  As its name suggests, MIMO [6] refers to a practical technique for sending and receiving more than one data signal simultaneously through the same radio channel, exploiting the multiple transmission. MIMO has become an essential element of wireless communication standards, including IEEE 802.11n (Wi-Fi), HSPA+ (3G), WiMAX (4G) and Long Term Evolution (4G LTE). Transmit/receive antennas serve the cell as well as the neighboring cells. This means that they can improve the quality of the received signal and reduce the cross-channel interference from the other neighboring cells. A MIMO technology example is shown in Figure 2.3.

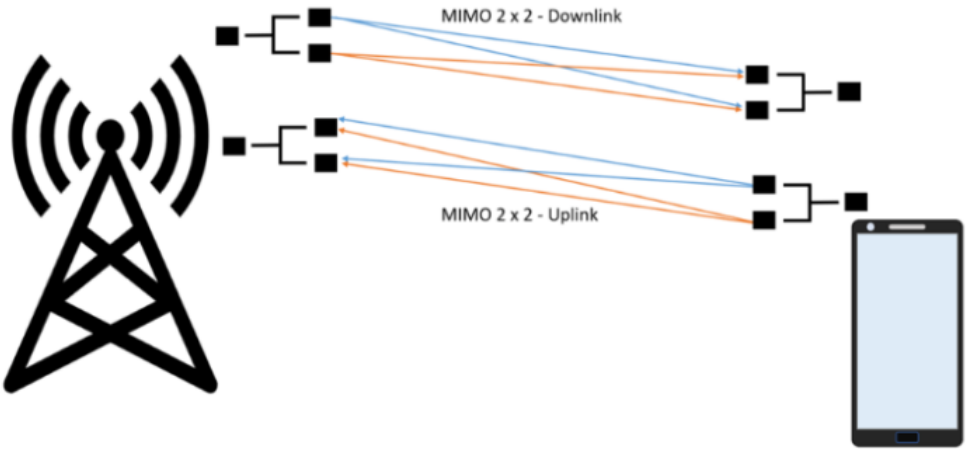

Figure 2.3: MIMO in LTE [7]

- **QoS (Quality of Service)**

  The purpose of QoS is handling the priorities in the network from some other services. It is necessary for a network prioritize traffic when there's a heavy workload and give priority to customers using demanding applications. In 4G, Qos implementations are done not only between the UE, but also on the PDN between the gateways where it is applied to all carriers. It is a set of virtual concepts and settings in the network to provide priority to some packets. For example, VoIP services have priority over those of the web Browser.

2.3 LTE Cellular Networks

2.3.1 Introduction to LTE

LTE [8], also known as 3GPP Long Term Evolution, is a cutting-edge technology that enables fast wireless networking and communication between mobile devices. By utilizing innovative setup methods, it expanded upon the GSM/EDGE and UMTS/HSPA networks that already existed. LTE was supposed to be an upgrade to the 3G standards. The cellular industry recognized its major benefits, and was accepted as the next generation. It was finally created by the 3GPP organization and despite being referred to as a fourth generation mobile communication technology, it does not match the standards established by the official organization ITU-R. Because of this, LTE needed to be upgraded to LTE Advanced in order to be recognized as a 4G system. Regarding the network architecture, the term LTE represents the evolution of radio access and focuses exclusively on optimizing the support of packet-switched applications, such as multimedia applications. It also sets very high and ambitious goals of exceeding the limits of 14.4 Mbps and 5.8 Mbps achieved in HSDPA and HSUPA respectively. It covers the entire range between 3G's to 4G's theoritical speed, giving it a massive range of potential speeds. On average, however, download speeds range from 12-30 Mbps, with faster speeds available in major cities. Theoretical and actual speeds by generation, are shown in Figure 2.4.

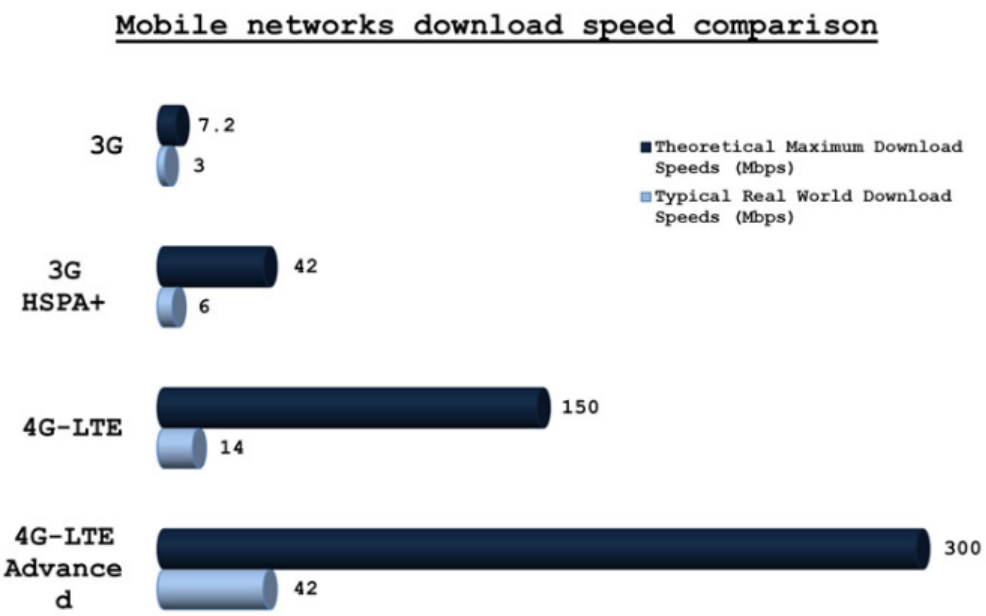

Figure 2.4: Speed Comparison Between 3G-LTE-4G [9]

2.3.2 LTE Architecture

2.3.2.1 Evolved Packet System (EPS)

The Evolved Packet System [10] is a logical representation of the LTE network architecture (shown in Figure 2.5), which identifies the functional entities of the architecture and the reference points between the functional entities with which interoperability is achieved. The overall architecture has two distinct components: the access network and the core network. The access network is the Evolved Universal Terrestrial Radio Access Network (E-UTRAN) and the core network is called the Evolved Packet Core (EPC). E-UTRAN and EPC together constitute the Evolved Packet System (EPS).
Both, the access network and the core network can achieve multiple functions, including:

- Network Access Control Functions
- Packet Routing and TransferFunctions
- Mobility Management Functions
- Security Functions
- Radio Resource Management Functions
- Network Management Functions

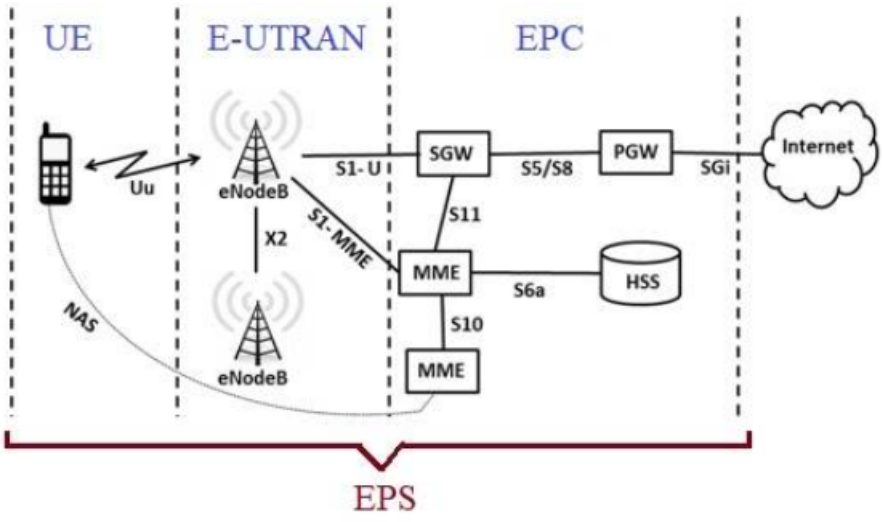

Figure 2.5: Evolved Packet System Architecture [11]

2.3.2.2 Evolved Universal Terrestrial Radio Access Network (E-UTRAN)

E-UTRAN [12] is the air interface for mobile networks. It is a wireless access network standard that replaces UMTS, HSDPA and HSUPA technologies defined in 3GPP standard and beyond. E-UTRAN is one new air interface system for data packet transfer. It is responsible for all functions related to the wireless part of the network, such as wireless resource management and header compression, to ensure efficient use of the wireless interface, security, and EPC connectivity. It uses OFDMA radio access for the downlink and SC-FDMA for the uplink. The E-UTRAN in the LTE architecture consists of a single node, the eNodeB that interfaces with the user equipment. The goal of this simplification is to reduce the latency of all radio interface tasks. The eNodeBs are connected to each other through the X2 interface, and then connected to the core network via the S1 interface. Each eNB is a base station that controls mobiles in one or more cells. A mobile communicates with only one base station and one cell at a time. The base station communicating with a mobile at a given time is called the serving eNB. The eNB has two main functions. First, the eNB sends radio broadcasts to all mobiles in the downlink and receives transmissions from them in the uplink, using the analog and digital signal processing functions of the LTE radio interface. Second, the eNB controls the low-level operation of all of its mobile phones and is connected to nearby base stations by the X2 interface, which is mainly used for signaling and packet transmission during transfer. The architecture of EUTRAN is shown in Figure 2.6.

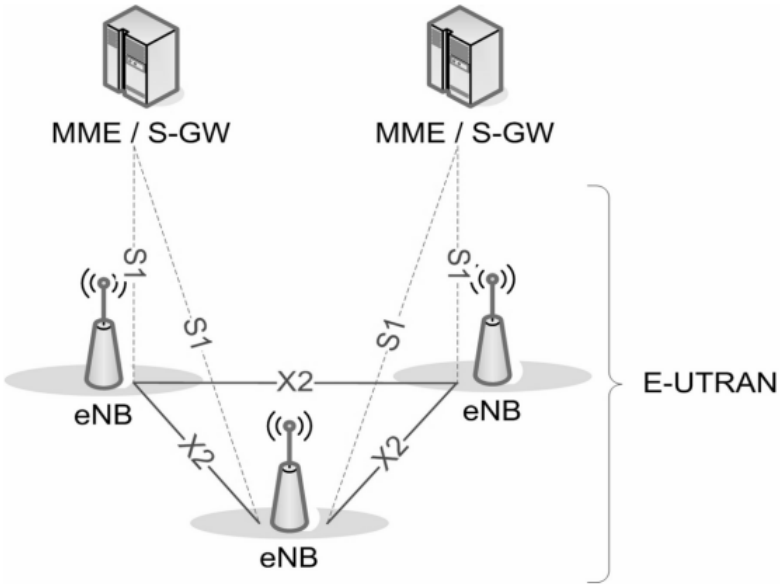

Figure 2.6: Evolved Universal Terrestrial Radio Access Network Architecture [13]

E-UTRAN Architecture Protocols

The general architecture protocol of E-UTRAN, presented in Figure 2.7 divides the radio interface into three layers: the physical layer (Layer 1), the data link layer (Layer 2), and the network layer (Layer 3). The purpose of the protocol stack is to configure services to organize the information they transmit through logical channels and to map these logical channels to transport channels so that it can control how and with what characteristics the information within each logical channel transmitted over the wireless interface. This means that for each transport channel there is one or more transport formats associated, each of which is defined by the encoding and mapping on the physical channel. Each layer is characterized by the services it provides to higher layers or the entities and functions they support as follows:

- **Physical Layer**

  Carries all information from the MAC transport channels over the air interface. Takes care of link adaptation, power control, cell search and other measurements for the RRC layer.

- **MAC (Medium Access Control)**

  The MAC layer offers a set of logical channels to the RLC layer that multiplexes within the physical layer transport channels. It also handles Hybrid Automatic Repeat Request error correction, prioritization of logical channels for the same UE and dynamic scheduling between UEs.

- **RLC (Radio Link Control):**

  Carries the PDCP Protocol Data Unit. It can work in three different ways, depending on the reliability provided. Depending on the way it works, it can provide ARQ (Automatic Repeat Request), error correction and segmentation / concatenation of PDUs.

- **PDCP (Packet Data Convergence Protocol)**

  It offers data transport with encryption and protection for the RRC layer as well as data transport with ROHC (Robust Header Compression) header compression and retransmission of its own SDUs (Service Data Units) for the IP layer.

- **RRC (Radio Resource Control)**

  It takes care of the transmitted system information related to the access and transport layer of Non-Access Stratum (NAS) messages, paging and many more.

- **NAS (Non-access stratum)**

  It is the protocol between the UE and the MME. Among other things it performs the mobility and session management of the UE and is responsisble for security control.

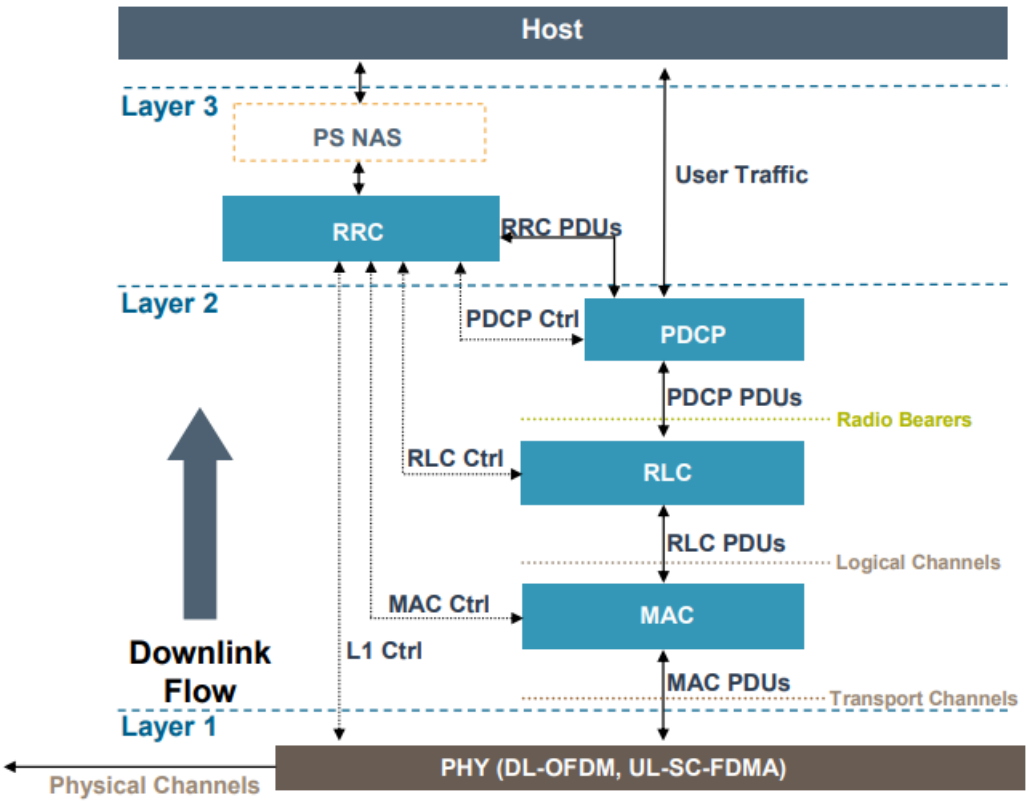

Figure 2.7: Evolved Universal Terrestrial Radio Access Network Protocol Stack [14]

2.3.2.3 User Equipment (UE)

In LTE and 4G wireless communications, a mobile device used to access the network, such as a smartphone, tablet, laptop, or other sort of wireless device, is referred to as User Equipment.

2.3.2.4 Evolved Packet Core (EPC)

The core of an LTE network is represented by the EPC [15]. The primary nodes that make up this network are MME, SGW, PGW, and HSS. These nodes provide a wide range of capabilities, including bearer configuration, authentication, mobility management, session management, and the use of various Quality of Services. It improves network performance by separating the control and data layers and provides packet switched services. Network complexity is reduced, as there are less hops in both signaling and data layers. The EPC is designed to support 3GPP and non-3GPP mobile IP access. It is a high performance and high capacity network that uses a IP-based structure to enabling simultaneous connection to more than one packet network and also helps service operators to provide more services such as voice over IP calls. Its goal is to manage the data effectively in terms of costs and performance. The handling of the traffic involves a small number of network nodes and protocol conversion is avoided. It is very important to mention that it is constructed in such way to make the scaling independent. For this purpose, it was implemented to split the user plane (user data), from the control plane (signaling).

EPC Architecture

The Evolved Packet Core (EPC) is responsible for the overall control of the UE and for the establishment of the bearers. Its representation is shown in Figure 2.8. It consists of the following functional elements:

- **Home Subscription Server (HSS)**

  Home Subscription Server is a central database that contains information about all network administrator subscribers. It also records the location of the user at the level of the visited node.

- **Packet data network gateway (P-GW)**

  Packet data network gateway is the EPC's point of contact with the packet data network. Through the SGi interface, each PDN gateway exchanges data with one or more external devices or packet data networks, such as the network manager's servers, the Internet or the IP multimedia subsystem. Each packet data network is identified by an Access Point Name (APN) and each network administrator typically uses a number of different APNs, Each mobile is assigned a default PDN gateway when it is first

activated to give continuous connectivity to a default package data network.

- **Serving gateway (S-GW)**

  Serving gateway acts as a high-level router, where data is passed between the base station and the PDN gateway. A typical network may contain a number of serving gateways and each mobile is assigned to a serving gateway. The serving gateway can be changed if the mobile moves far.

- **Mobility management entity (MME)**

  The Mobility management entity controls the high-level operation of the mobile, sending signaling messages for issues such as security and data flow management related to radio communications. Each mobile is assigned to a single MME, known as the serving MME, but this can change if the mobile moves far enough. The MME also controls the other network elements, through signaling messages which are related to the EPC.

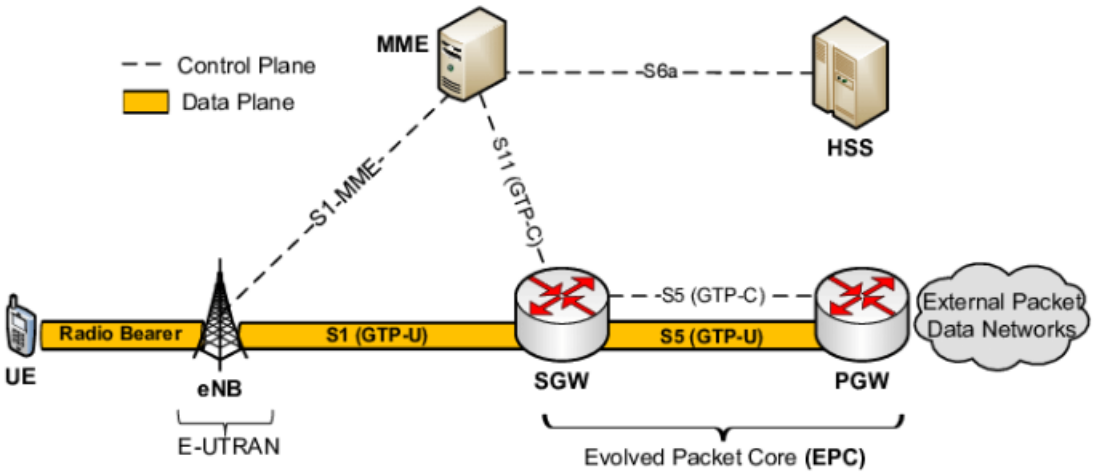

Figure 2.8: Evolved Packet Core Architecture [16]

2.4 5G Cellular Networks

2.4.1 Introduction to 5G

By 2015 4G networks had been installed across the globe, but the demand for higher data rates is increasing due to the growth of video traffic, VR applications, high-definition video streaming etc.. This results in the introduction of the 5th generation (5G) networks [17]. The development of 5th generation telecommunication systems was expected to be implemented in early 2020, a milestone year for wireless communication as it brings new ”smart” antennas, high-tech devices and new applications for wireless communication. In 5G networks the service area provided by carriers is divided into a mosaic of tiny geographic regions called cells, like in earlier generations of 2G, 3G, and 4G mobile networks.Local antennas are linked together via wireless backhaul or high-bandwidth optical fibers. Similarily to previous generations, when a user changes cells, his mobile device seamlessly shifts to the antenna of the new cell. The fundamental benefit of 5G networks is that they attain far greater data rates than earlier cellular networks, up to 10 Gbit/s, which is 100 times faster than the previous cellular technology, 4G/LTE, and faster than existing cable internet. Its network’s decreased latency is another major benefit. The amount of time it takes for a message to travel between a sender and a receiver is known as network latency. Compared to earlier cellular networks, 5G will have latency below 1ms as opposed to 30–70 ms in 4G. 5G networks achieve higher data rates with higher frequency radio waves, using the millimeter wave band, which ranges from 30 to 300 GHz. Previous cellular networks use frequencies in the microwave band, between 700 MHz and 3 GHz. Due to the greater bandwidth at millimeter wave frequencies, 5G networks use wider frequency channels, up to 400 MHz, to communicate with the wireless device, compared to 20 MHz in 4G/LTE. This means that they can transmit more data (bits) per second and achieve greater throughputs. So in general with 5G technology:

- Society’s demands for a greater amount of information and applications will be met.
- The capacity problem that already exists in 4G/LTE systems will be eliminated.
- Massive machine connectivity.
- New spectrum bands and
- New Network functions for mobility and security.

2.4.2 5G Key Technologies

5G technology must handle much more traffic in much higher speeds than the base stations that make up today's cellular networks. To achieve this, engineers are designing a number of brand new technologies, in order to deliver data with a latency of less than 1 ms and provide users with maximum download speeds.

These technologies include:

- **Millimeter Waves (mmWaves)**

  It is obvious that the mount of data exchanged around the world has grown tremendously. However the use of the same bands of the radio frequency spectrum, has led to less bandwidth, causing low speeds and more connection drops. This is why scientists considered of using mmWaves [18], which use higher frequencies than the radio waves used for mobile phones. mmWaves finally gave the ability to signal in a completely new frequency band that has not been used for mobile service before. mmWaves transmit at frequencies between 30 and 300 GHz (practically 30 - 100 GHZ) and vary in length from 1 to 10 mm, compared to the radio waves that serve today's smartphones, which are tens of centimeters long (Figure 2.9). Up to this point, the only practical uses of mmWaves were in satellites and radar systems. Recently, some mobile providers have started using them to send data between two base stations. However, a significant disadvantage of mmWaves is that they are easily blocked by objects like buildings and can be absorbed by rain.

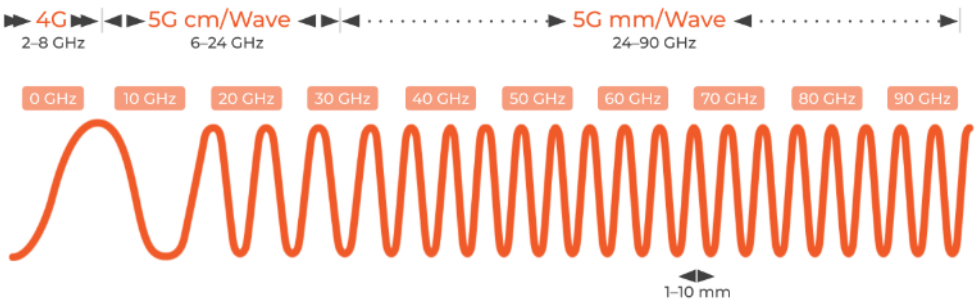

Figure 2.9: mmWaves and Radio waves frequency spectrum analysis [19]

- **Massive MIMO**

  Today's 4G base stations have twelve antenna ports that handle all cellular traffic, eight for transmitters and four for receivers. It all starts with MIMO technology, which, as mentioned earlier, describes wireless systems that use two or more transmitters and receivers to send and receive more data simultaneously. In order to send and receive signals from more users concurrently, 5G base stations host approximately 100 ports, which increases the capacity of mobile networks by a factor of 22 or more.

- **Small Cells**

  Small cells [20] are moveable and small base stations that require less electricity to operate. Thousands of these stations may be deployed across a city to create a dense network that receives signals from other base stations and sends data to users wherever they are, preventing signal dropouts. Fortunately, small cells will utilise mmWaves for transmission, hence their antennas may be considerably smaller than conventional antennas. Their small size gives the advantage of installing Small Cells everywhere, like light poles, trees, buildings etc. This fundamentally redesigned network architecture would need to offer more targeted and efficient spectrum usage. This means that in order to serve more clients, a station in one location may reuse frequencies used by another station in another area.

- **Full Duplex**

  With full duplex [21], a transceiver will be able to send and receive data simultaneously over the same frequency. Full duplex is a method that has the potential to double the physical layer capacity of wireless networks. Modern full-duplex channels generally utilize two distinct channels for communication between systems. Already, advanced telecommunications networks and full-duplex Ethernet both rely on channels that may be utilized for simultaneous transmission.

- **Beamforming**

  Cellular base stations employ beamforming, a traffic-manager mechanism, to determine the most effective data distribution path to a specific user while minimizing interference. Beamforming is used along with MIMO technology to direct the wireless signal in a given direction towards a particular receiving device. As a consequence, the signals are enhanced, the number of connections is increased and interference is

minimized as much as possible.

2.4.3 5G Architecture

5G network was designed as a ground up model and its functions are split up by service. That is why this architecture is also called 5G core Service Based Architecture [22]. The diagram in Figure 2.10 shows the key components of a 5G core network:

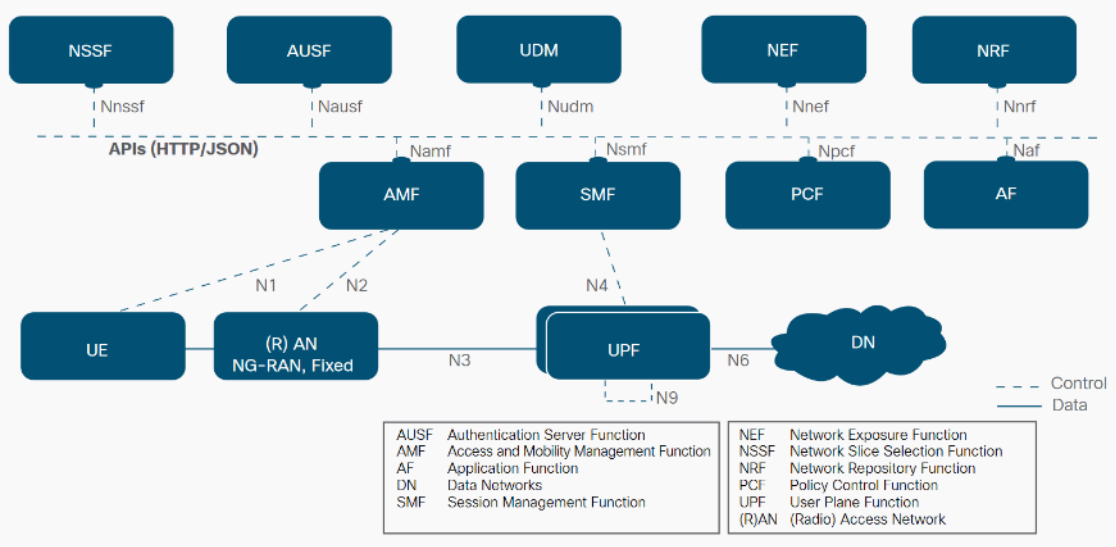

Figure 2.10: 5G Core Network Architecture [23]

- **AMF**: Registration, access control and mobility management.
- **SMF**: Creates, updates and removes PDU sessions. Manages session context with UPF. UE IP address allocation and DHCP role.
- **UPF**: User plane packet forwarding and routing. .
- **NRF**: Maintains updated records of services provided by other NFs.
- **NEF**: Securely opens up the network to third-party applications.
- **AUSF**: Authentication for 3GPP access and untrusted non-3GPP access.
- **NSSF**: Selects network slice instances for the UE. Determines AMF set to serve the UE.
- **UDM**: Generates AKA authentication credentials and uthorizes access based on subscription data.

- **AF**: Interfaces with 3GPP core network for traffic routing preferences, NEF access, policy framework interactions and IMS interactions.
- **BSF**: Binds an AF request to the relevant PCF.

2.5 Contribution

In this thesis we use the previously analyzed 4G-LTE architecture, which we deploy on real machines in the NITOS testbed, located in the University of Thessaly. More specificalLy, we use the OpenAirInterface's 4G architecture, which is based on the actual 4G-LTE architecture. By having a real network at our disposal, we then use its structure to carry out the appropriate experiments in realistic conditions.

The OpenAirInterface gives the ability of deploying a 5G network on real machines as well. This could be done by changing the major functions of 4G/LTE with those of 5G as shown in Figure 2.11:

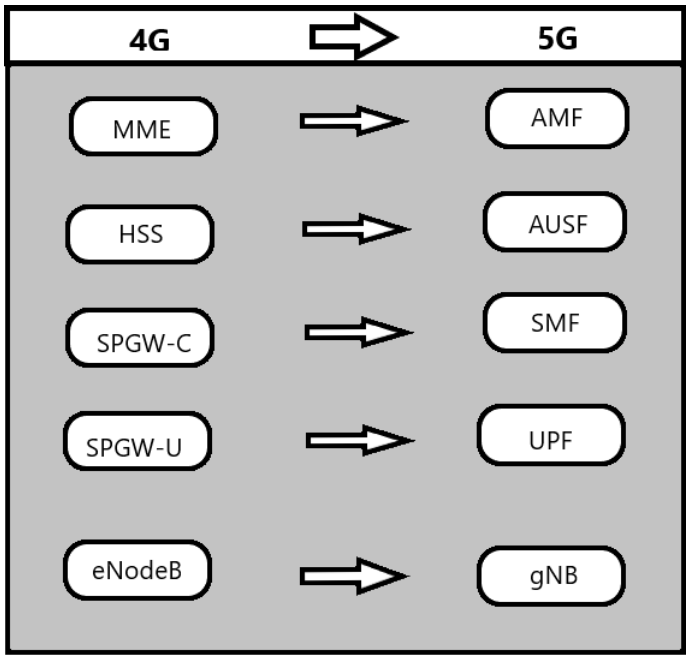

Figure 2.11: Functions Transition from 4G to 5G

Chapter 3

Software

3.1 Docker

3.1.1 Overview

To begin with, there are three main ways to run an application:

1. Run the application directly from the operating system
2. Run the application on a virtual machine (VM).
3. Run the application in a container (Docker).

In this analysis, we present Docker [24], along with its advantages over the other two options. Docker is a container host that offers container management services. As their name suggests, ”containers - boxes” have specific properties and constitute the space in which the applications are executed. Containerization gives us the ability to deploy multiple applications using the same operating system on a virtual machine or host machine. Through Docker, it is possible to package application components in container images, store container images in local or remote registries as well as create containers through their images. A Docker container does not have its own core. It uses the host operating system and the host kernel. As a result, it is very light and uses only the necessary resources to run the services. Each container is completely isolated from the host operating system and other containers. The concept of Docker is to start a container for every service we want. For example, if we need Apache or mySQL, we start a container with Apache or mySQL respectively.

3.1.2 Advantages of Using Docker

1. **Quick start of the application.**

   In order to run an application in a VM, a complete Operating System must be installed and running, which obviously requires a lot of time. In Containers the application starts immediately because there is no complete Operating System (Figure 3.2).

2. **Low resource consumption**

   As mentioned, containers do not contain a complete operating system and therefore they appear in computer systems as services, consuming only resources that are required to execute the application.

3. **Execution of multiple replicas of the same application on the same computer unit.**

   Each container is isolated from the Operating System having its own file system, its own network and all necessary libraries to run the application. Each container can access its specific resources system if desired. This resource isolation allows us to run the same application on same or different version many times (Figure 3.1).

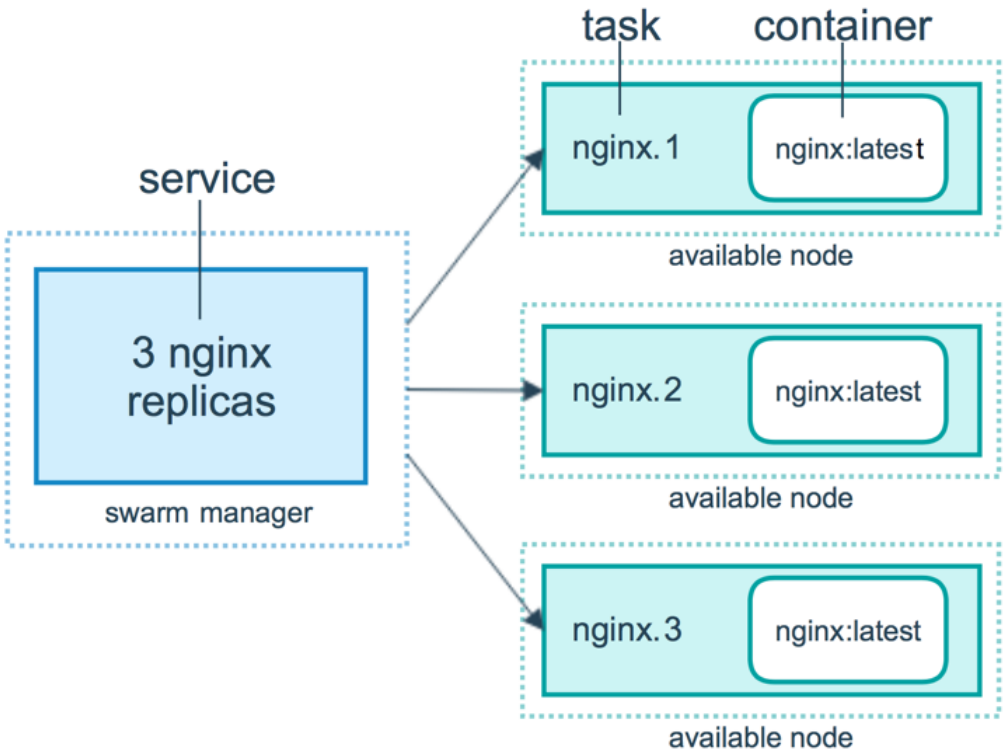

Figure 3.1: Application replicas [25]

4. **Ease of moving the application to another computer unit.**

   The ability of containers to be executed regardless of the Operating System and the computer structure, gives them autonomy and flexibility to move the application from one computer unit to another.

5. **Security**

   In addition to the great flexibility and ease of creating container images, Docker offers a high level of isolation and security.

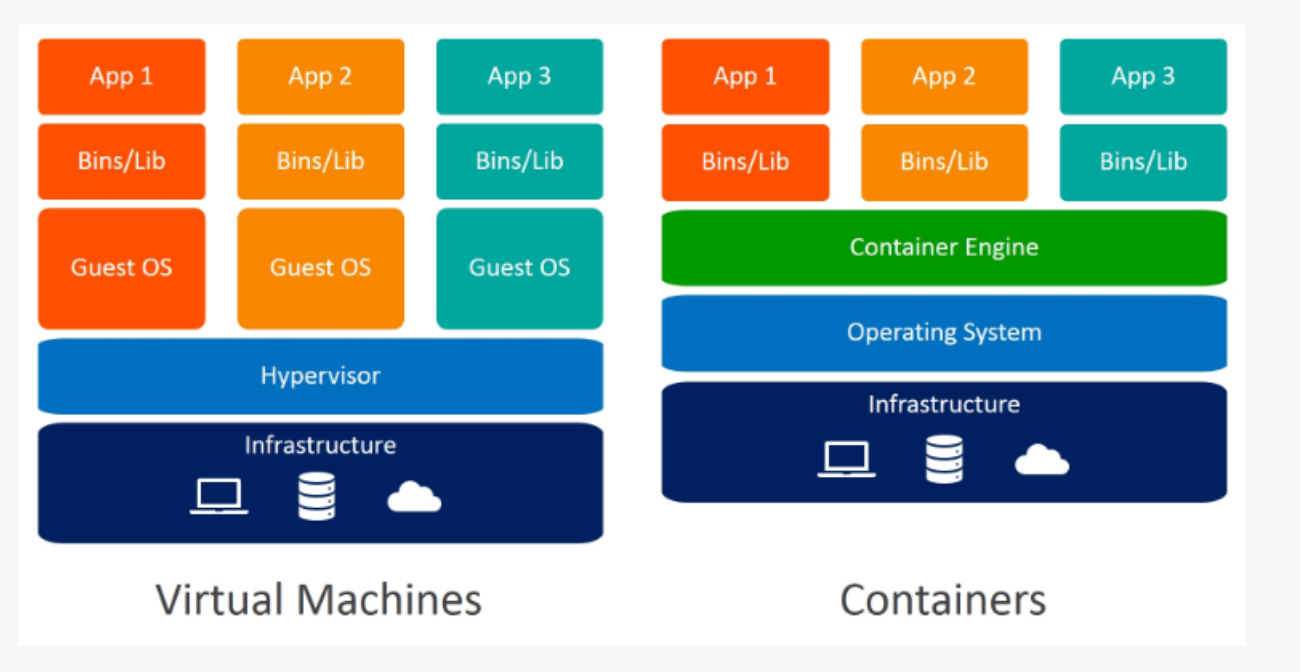

Figure 3.2: Virtual Machines vs Containers [26]

Although containers consume minimal resources compared to a virtual machine, they are not so stable and often fail. However, the short start time allows us to replace them in a long time short time.

3.1.3 Docker Architecture

The Docker mechanism is based on the client-server model [27], which is called Docker Engine and consists of the following main elements :

- A **server** running on the operating system as a service called Docker daemon.
- A **REST API** which defines interfaces with which the programs can communicate with the server.
- A command line interface **(CLI Client)**.

The execution of commands within the command line (CLI Client), will communicate through the REST API with the Docker Daemon. Then, the daemon will perform any work of creation, execution or Docker Container distribution. Figure 3.3 represents Docker's architecture as described above.

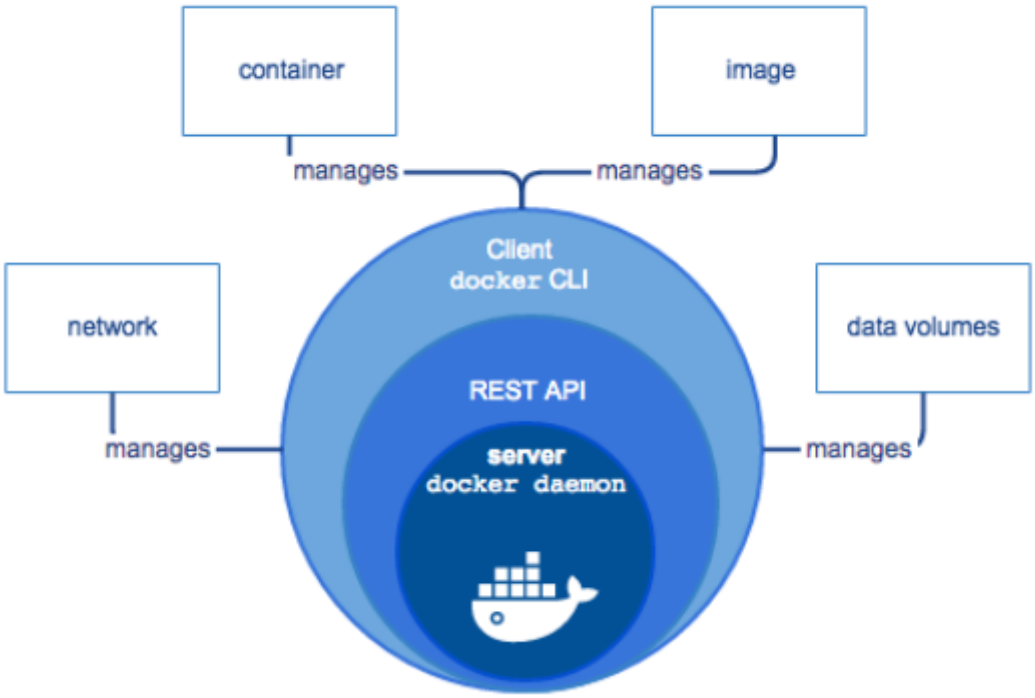

Figure 3.3: Docker Architecture [28]

A Docker image is a file that includes all the components such as libraries, configuration files, and system tools, that are required for an application to run in a container. Usually, an image is based on one another image with some additional changes. Docker uses a file named Dockerfile which contains a description of how to create the images and how they are going to be executed. We can either create our own images (with the use of a Dockerfile) or use images that have been by created other users (uploaded to registries). A layer refers to each individual file that makes up a Docker image. These layers provide a series of intermediary images that are constructed one on top of the other incrementally. Each layer is reliant to the layer directly behind it (Figure 3.4). The layer hierarchy is essential for effective lifecycle management of the Docker images. Layers that change more frequently should be placed on the stack as high as possible. When a layer in an image is modified, Docker rebuilds all layers based on that layer as well. Therefore, if a layer at the top of a stack is changed, it will take less computational work to recreate the complete image.

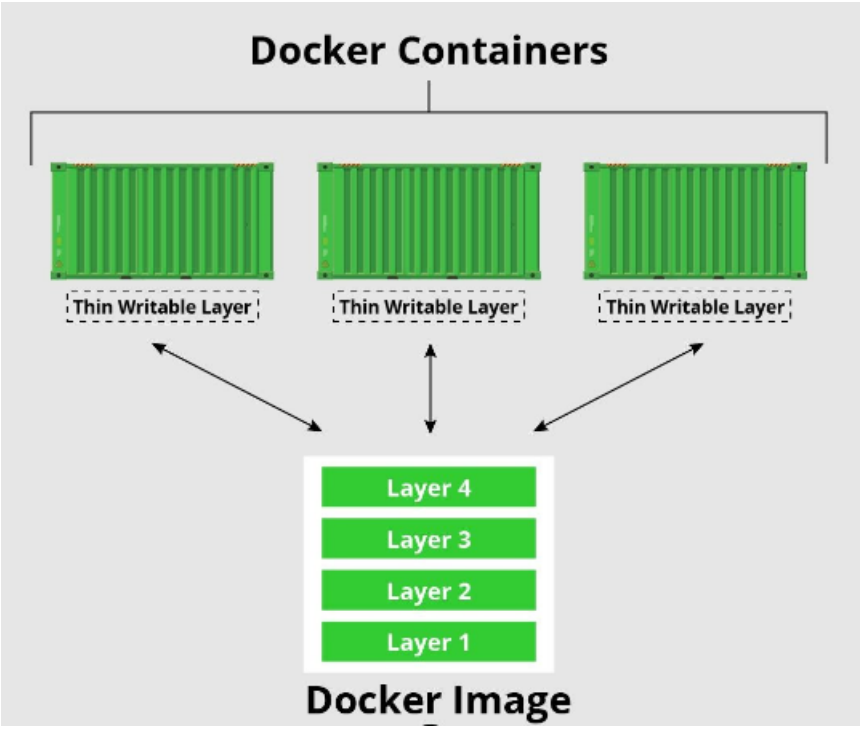

Figure 3.4: Docker Layers [29]

3.2 Kubernetes

3.2.1 Introduction to Kubernetes

3.2.1.1 Overview

The development, management, and scaling of container applications are organized and automated via the open source framework Kubernetes (K8s) [30]. It is developed to orchestrate containers over a large number of computers, including cloud-based and native systems. The application's state is continuously monitored and maintained by Kubernetes, according to detailed specifications. Kubernetes uses a master-slave model, in which the master component manages the worker nodes, where the containers are running. Some of the main features that K8s is capable of are presented below:

1. **Load balancing**

   Load balancing is the process of distributing network resources efficiently among multiple services. It represents a strategy that is used for maximizing availability and scalability. Kubernetes uses a wide variety of load balancing strategies and algorithms to achieve the most ideal resource management.

2. **Storage Management**

   Users may choose to automatically mount their preferred storage system (local storage, public cloud providers etc.) using Kubernetes. Nearly every application requires some form of external storage and as a result, Kubernetes storage mechanisms were developed to provide persistent storage to those containers. Kubernetes offers an API that separates the specifics of storage provisioning from storage consumption. However, the underlying storage system must still be provided.

3. **Continuous Control of the Desired State of Deployment**

   YAML or JSON files may be used to describe the state of deployed containers. Then, Kubernetes is in charge of gradually transforming the present state into the intended state.

4. **Resource Management**

   Kubernetes provides a variety of Resource Managers to handle applications with high throughput and latency-critical requirements.

5. **Health Checking**

   If an application component or containerized app crashes, Kubernetes will immediately restart it or replace it. Kubernetes owns mechanisms capable of fixing pods as well. The replication controller provides the fault tolerance of apps during the self-healing process. Kubernetes is able to find and restart a container that has failed or even terminate the container if it doesn't responding to the client. Finally, it may reschedule the containers on other nodes, in case a node goes down.

6. **Secret and configuration management**

   Kubernetes handles and stores sensitive data like passwords, SSH keys, etc. Secrets and application configuration may be updated and deployed without requiring the user to rebuild the container images or expose secrets in the stack configuration.

3.2.1.2 Kubernetes Cluster

Generally a cluster [31] is the interconnection of many computers with similar technical characteristics (although not necessary) in a network. A cluster can exchange data through its network and can be considered as a single computer. As a result, clusters can carry out complicated tasks, but need a lot of processing power because it is unknown in which machine each activity is executing.

The machines that make up a cluster are referred to as ”nodes,” and specifically in K8s, they are referred to as ”worker nodes”. Depending on the cluster, the K8s’ worker nodes may be virtual machines (VMs), real computers, or a combination of both. Additionally, there are no restrictions on the nodes’ locations, allowing their installation in hybrid clusters. Task scheduling and cluster state monitoring are the responsibilities of the master node, which makes up the control plane of a cluster. A master node controls and manages a set of worker nodes. A typical cluster layout is shown in Figure 3.5.

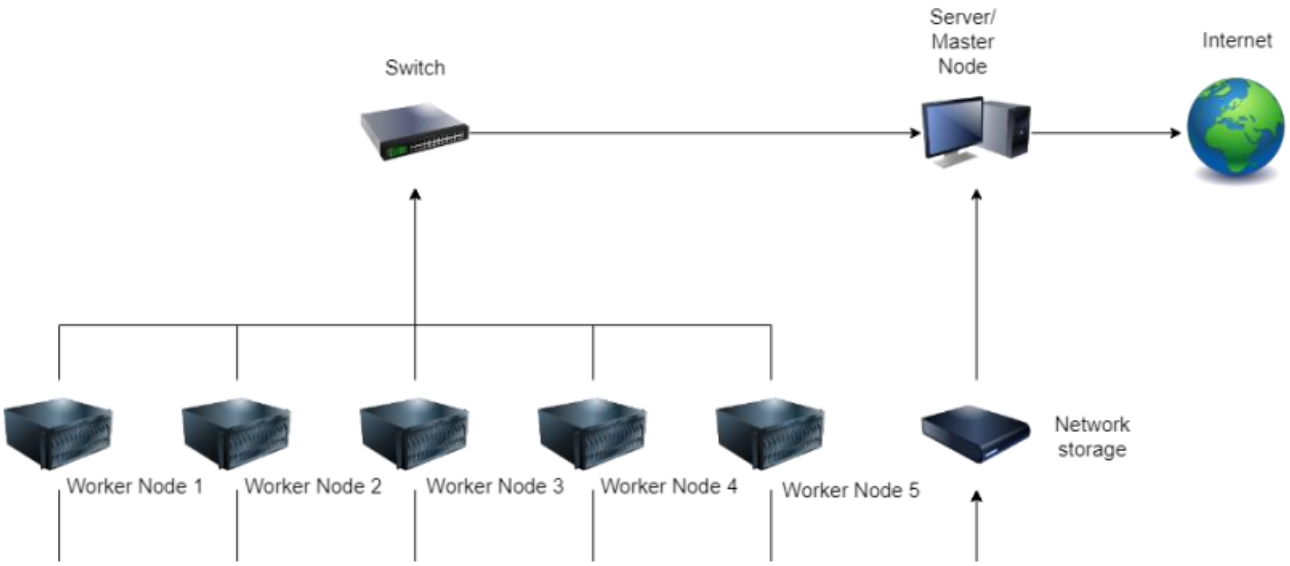

Figure 3.5: Cluster Architecture

A cluster’s active processes are represented by pods. Pods may have one or several containers, that collaborate closely to carry out a specific task. Multiple containers in a pod make it easier to communicate and share data. Containers in a pod, can communicate over the localhost since they all share the same network namespace. Pods can connect between them by referencing a resource that is located in another pod or by utilizing the IP address of another pod. Pods also make scaling easier by allowing replica pods to be automatically initiated and stopped in response to variations.

Pods have unique IP addresses, persistent storage volumes and information about the execution of a container.

Practically, the set of services running on a master node is (Figure 3.6):

- **Kube-apiserver**

  It serves as the gateway to the K8s cluster and is used to communicate and interact with users. The kube-apiserver grants access to Kubernetes and manages all cluster-related tasks.

- **etcd**

  All objects createed and developed in the Kubernetes Cluster as well as their condition are stored on a permanent base called etcd. The API Server is the only component capable of communicating with the etcd database.

- **Kube-scheduler**

  The Pods assignment to the nodes of the cluster is done by the kube-scheduler. The Kube-apiserver informs the scheduler that a new Pod has been requested and as a result the scheduler assigns the pod, without executing it.

- **Kube-controller-manager**

  Kube-controller-manager operates on master nodes and manages the various controller processes. When the actual and expected statuses of the various services provided by the API don't keep up, these controllers take remedial action. Except from managing nodes, workloads, namespaces, and service accounts, Kube-Controller-Manager also manages nodes and replication controllers for workloads.

Respectively, the set of services running on a worker node is:

- **Kubelet**

  Worker Nodes execute the Kubernetes' component known as Kubelet, which is in charge of managing all operations on the Node. It initiates the establishment of the node-entry to the cluster and is responsible for running the appropriate pods. Additionally it keeps track of the new pods assigned, notifies the state of the running containers and continuously checks their health. Finally, it tracks the usage of Node resources in the Kube apiserver.

- **Kube-proxy**

  Kube-proxy guarantees that K8s users access the necessary services provided by the cluster administrator via the Kube-apiserver. Therefore, Kube-proxy makes sure that clients connect to the appropriate IP address (and ports) of the Pods providing the service. Kube-proxy also implements load balancing between these Pods.

- **Container Runtime**

  It is simply the mechanism that Pods use to run the containers.

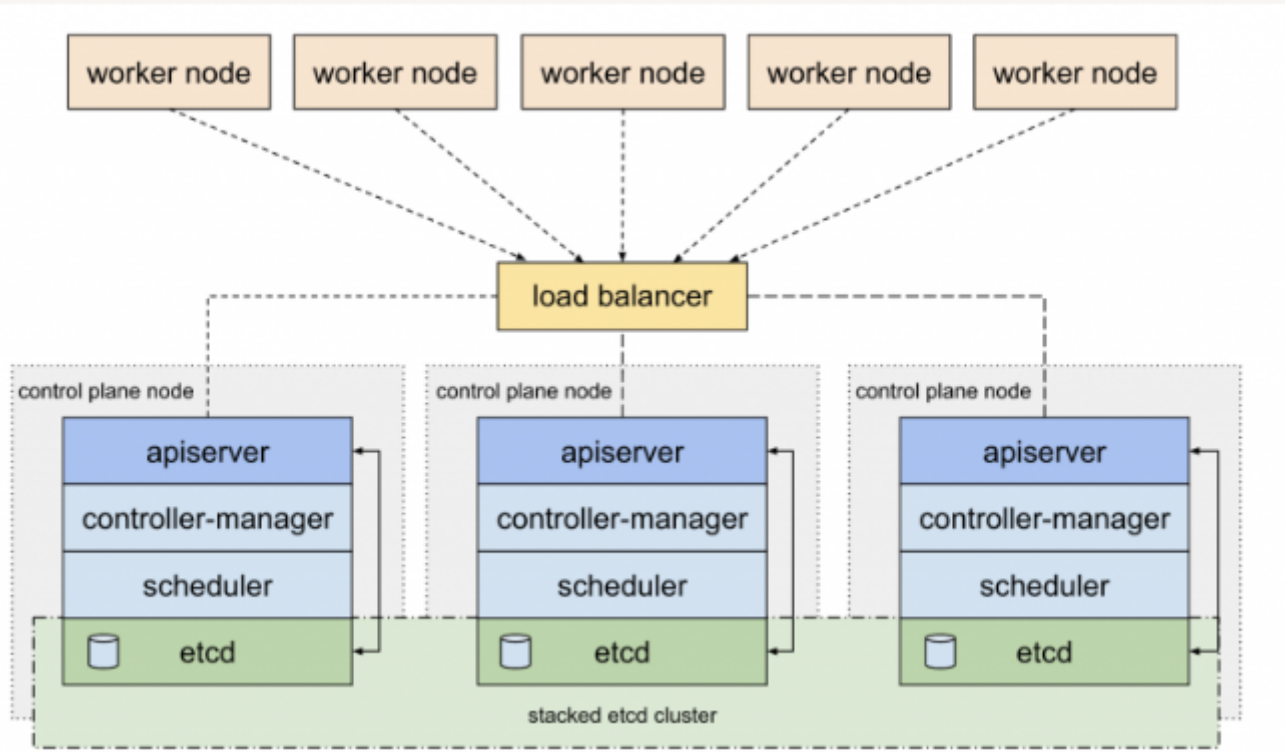

Figure 3.6: Services on a Cluster [32]

3.2.1.3 Kubernetes Networking

Kubernetes uses an ”overlay” technology network for communication between the Pods. Kubernetes does not have any built-in networking mechanism with the installation of the cluster, but offers the description of the mechanism and the data prerequisites, leaving its implementation to external mechanisms. Kubernetes supports 23 external mechanisms for networking. Each mechanism implements the communication between pods and containers and ensures that the following requirements are met:

- The IP addresses of each pod should be distinct.

- Each Pod on a worker node should be able to connect directly with Pods on the same and other nodes, without the use of NAT.

- Node’s pods should be able to directly connect with all the services that are running on the node, without the need of NAT.

In a kubernetes network, there are three main ways for communication between containers and pods:

1. **Container to Container Communication**
   The same IP address is shared by all containers in the same Pod and can communicate by using the localhost address. For instance, a container in a pod can use the address localhost:8080 to communicate with another container on port 8080.

2. **Intra-Node Pod Communication**
   Intra node communication describes the communication between pods in the same node. The pod establishes a connection via the virtual ethernet device and tunnels the traffic to the node’s ethernet device. A network bridge called cbr0 makes it possible for pods to communicate with one another. This network bridge connects every pod and routes traffic when a network request is made, by looking up the proper destination.

3. **Inter-Node Pod Communication**

   Inter node communication describes the communication between pods in different nodes. The network bridge sends traffic to the default gateway when the required IP address cannot be located in the pod. The IP will then be searched at the cluster level. All IPs connected to each node are recorded by Kubernetes. When a request is made to an IP address in the cluster, it first searches for the desired IP and then directs the request to the right node and pod.

3.2.1.4 Kubernetes Services

A service is in charge of providing pods with an interface that allows network connection within the cluster and between external processes.

- The default service type, **ClusterIP**, only allows connections from within the cluster and exposes the service on a cluster-internal IP.

- **NodePort** exposes the service on each Node's IP at a static port. A NodePort service can be accessed outside the cluster.

- **LoadBalancer** makes the service accessible to other parties through a load balancer of a cloud provider. Thus, the NodePort and ClusterIP services, that the external load balancer is using, are generated automatically.

- **ExternalName** returns a CNAME record with the value of the externalName field, mapping the Service to its contents.

3.2.2 Multus CNI

CNI stands for Container Networking Interface and its goal is to create a generic plugin-based networking solution for containers. Multus CNI [33] is a Kubernetes container network interface plugin that enables pods to be connected to numerous network interfaces. With the exception of a loopback, each Kubernetes pod has typically only one network interface. However, Multus allows you to construct multi-homed pods with many interfaces.

3.2.3 Flannel Networking

Developed for Kubernetes, Flannel [34] is an open-source virtual network project representing a basic overlay network that works by assigning a range of subnet addresses. In a flannel cluster, each server executes an agent named flanneld. Each host is given a subnet, which serves as a pool of IP addresses for the containers on the host. Using their IP address, containers may then communicate with one another directly. For packet encapsulation, Flannel offers many backends. The concept of Flannel is represented in Figure 3.7.

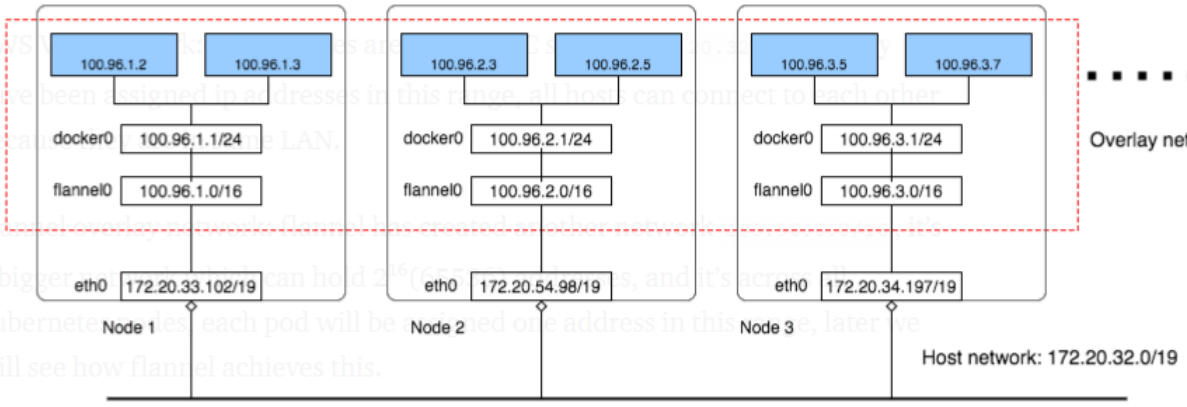

Figure 3.7: Flannel Networking [35]

3.2.4 Prometheus

Kubernetes and other cloud-native systems may be monitored and alerted by using Prometheus [36], an open-source platform that is capable of recording and gathering measurements as time-series. Prometheus obtains metrics by using an accessible HTTP endpoint . As soon as an endpoint is accessible, it starts gathering numerical data ,record it as a time series, and store it in a local database designed to hold time-series data.
The client libraries of Prometheus offer four types of metrics:

- **Counter**
  This measurement stands for a single growing counter, whose value can either rise or reset to zero upon restart.

- **Gauge**
  This measure indicates a single numerical value that may be freely increased or decreased. Values like current memory utilization or temperatures are frequently measured using a gauge.

- **Histogram**
  A histogram samples observations, such as request durations or response sizes.The observations are then added together in a customizable bucket. A histogram can also show the sum of all the values that were sampled.

- **Summary**
  Observations like request durations and response sizes can be sampled in a summary. A total count of the observations as well as the sum of all observed values can also be provided.

Once Prometheus gathers the data, the user can use the PromQL query language to retrieve it and then export it to graphical interfaces like Grafana or Alertmanager. The Prometheus Architecture is shown in Figure 3.8

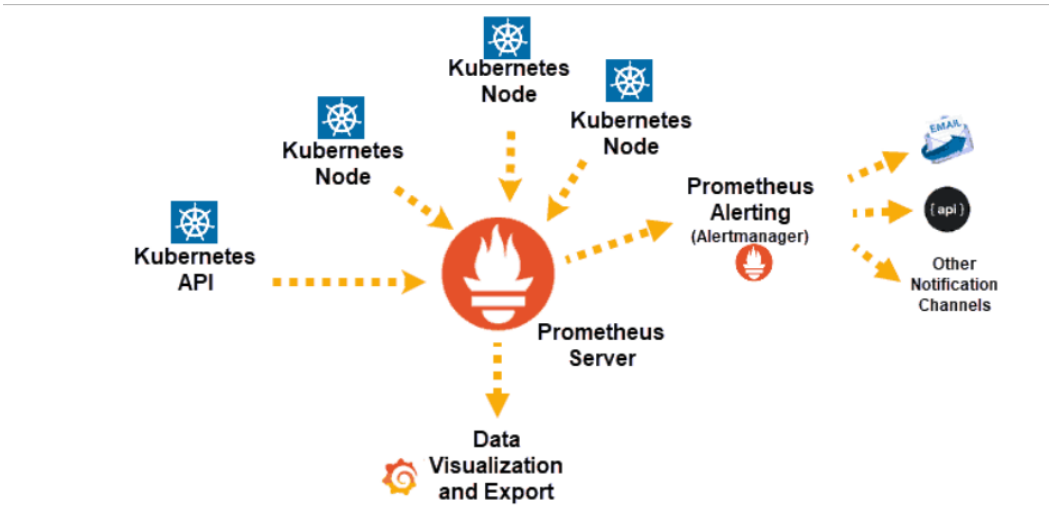

Figure 3.8: Prometheus Architecture [37]

3.2.5 Grafana

Grafana [38] is an open-source analytics and interactive visualization web application. It can access many different types of data sources, including Prometheus, InfluxDB, Elastic-Search, and conventional relational database engines. Using these sources, a user may choose important features from his data and create complicated dashboards (Figure 3.9).

Figure 3.9: Grafana Dashboards Examples [39]

3.2.6 Horizontal Pod Autoscaler

The Horizontal Pod Autoscaler adjusts the shape of the Kubernetes workload by adjusting the number of Pods automatically in response to any custom metrics that are reported from within Kubernetes or external metrics from sources outside of your cluster.

3.3 Software Contribution

As mentioned previously in this thesis, the experiments are based on the OpenAirInterface 4G-LTE concept. However, it is crucial that the functions of the network are executed in a microservices environment. This is where the Kubernetes software is introduced to deploy and manage the resources of this network. However, since the K8s is a platform that manages containers, the functions of the OAI should be containerized. For this purpose the Docker platform is used. This helps to perform these functions together through the pods offered by kubernetes. More specifically, the docker images that were containirized to run as pods are cassandra, HSS, MME, SPGW-C, SPGW-U and the eNodeB.

A Kubernetes Cluster was then created for deploying and managing these pods. The cluster consists of one master node and 3 worker nodes that share these pods. The cluster operates with Flannel networking technology to allow the containers receive a valid IP address and communicate with each other. It also uses Multus CNI to enable pods to be connected to numerous network interfaces, instead of one.

In order to scale and manage the number of pods that should be deployed, we adjusted the Horizontal Pod Autoscaler to get the desired metric from Kube-prometheus.

Finally, Prometheus and Grafana were used extensively to monitor, analyse and virtualize the measurements taken by the experiments.

Chapter 4

Experiments and Experimental infrastructure

4.1 NITOS

NITOS [40] is a heterogeneous testbed-integrated facility with an emphasis on promoting experimentation-based research in the field of wired and wireless networks. The scientific community has 24/7 access to NITOS over the internet and is able to control and manage the testbed through a very useful open-source software called Control and Management Framework (OMF). Through NITOS scheduler and OMF management framework, users may easily conduct their experiments by reserving slices (nodes, frequency spectrum) of the testbed, which allow experimentation and code development. It consists of three distinct testbeds, including the Outdoor Testbed, the Office Testbed and the Indoor RF Isolated Testbed.

- **Outdoor Testbed**

  The NITOS Outdoor deployment is composed of 50 powerful nodes that come with several wireless interfaces and allow experimentation of heterogeneous (Wi-Fi, WiMAX, LTE) wireless technologies. It is built on top of the University of Thessaly's campus building (Figures 4.1 & 4.2).

Figure 4.1: Outdoor Testbed [40]

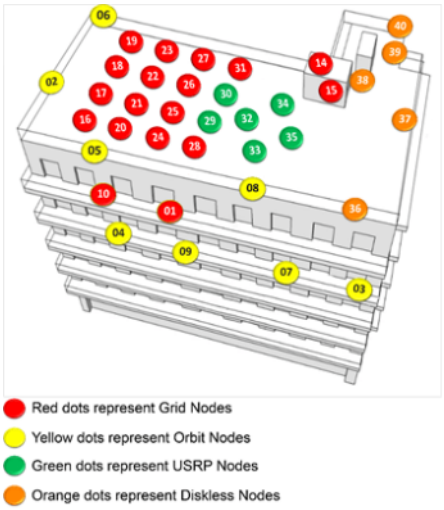

Figure 4.2: Outdoor Testbed Topology [40]

- **Office Testbed**

  The Office Indoor Testbed consists of ten second generation Icarus nodes, like the outdoor testbed and give the ability to create and carry out real-world scenarios in a realistic office setting (Figures 4.3).

Figure 4.3: Office Testbed [40]

- **Indoor Testbed**

  The NITOS Indoor deployment (Figure 4.4) consists of 50 Icarus nodes that feature multiple wireless interfaces (Wi-Fi, WiMAX, LTE) and is deployed in an isolated basement in University of Thessaly's campus building. It is also equipped with directional antennas prototypes.

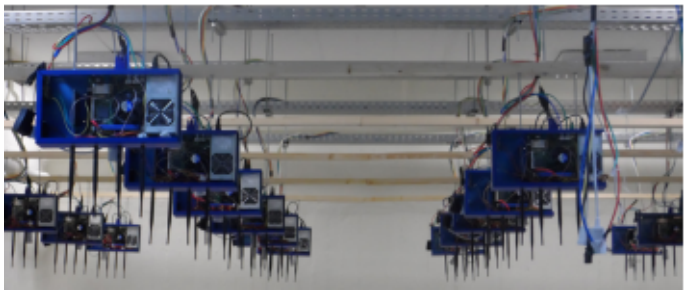

Figure 4.4: Indoor RF Isolated Testbed [40]

In order to build a flexible experimental environment, resources from each testbed may be combined. The overall architecture of the NITOS facility is demonstrated in Figure 4.5:

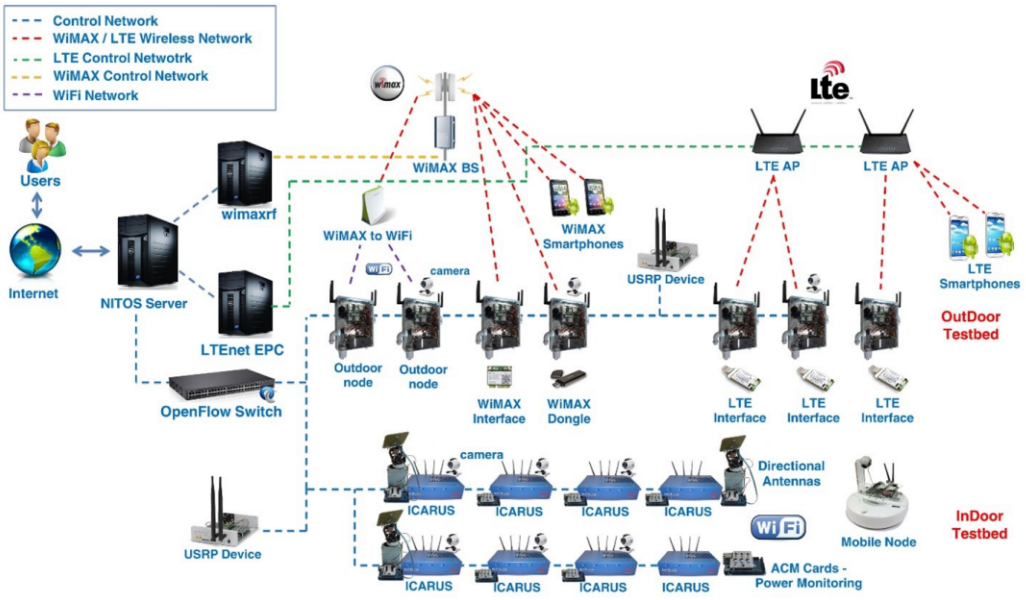

Figure 4.5: NITOS Overall Testbed Topology [40]

Icarus Node

For the experiments that took place in this thesis, the testbed that we used was the Indoor RF Isolated Testbed. As mentioned before, this testbed uses Icarus nodes (Figure 4.6), which fully support experimentation with USRP and LTE technologies used for this thesis. Some additional features of the Icarus Nodes are presented in Figure 4.7:

Figure 4.6: Icarus Node [40]

Motherboard	Features 2 Gigabit network interfaces and supports 2 wireless interfaces
CPU	Intel Core i7-2600 Processor, 8M Cache at 3.40Ghz
RAM	4G HYPERX BLU DDR3
Wireless interfaces	Atheros 802/11 a/b/g & Atheros 802.11 a/b/g/n (MIMO)
Chassis Manager Card	UTH's CM card
Storage	Solid state drive
Power supply	350 Watt mini-ATX
Antennas	Multi-band 5dbi, operates both on 2,4Ghz & 5Ghz
Pigtails	High quality pigtails (UFL to RP-SMA)

Figure 4.7: Icarus Node Specifications [40]

4.2 Open Air interface

4.2.1 Overview

OpenAirInterface [41] is a platform created by EURECOM to support mobile telecommunication systems like 4G and 5G. Its objective is to create access solutions for the Radio Access Network and the Core Network. It is an open-source platform, meaning that the code may be modified for various deployment and use cases as well as the addition of new functionality. Using general purpose x86 computer hardware and commercial SDR cards like the USRP, OAI implements 3rd Generation Partnership Project technology. It supports deployment and operation of both 5G New Radio (NR) networks and 4G Long-Term Evolution (LTE) networks.

4.2.2 OAI's 4G-based Architecture

As mentioned before, OAI has implemented open source hardware and software realising the EPC, access network, and user equipment of cellular networks. The concept [42] of the software components used for the experiments is briefly analyzed in Figure 4.8:

- **Home Subscriber Server (HSS)**

  The Home Subscriber Server is the central database containing the information about users and their subscriptions. In detail, information about user identity and addressing (IMSI and MSISDN) is kept and updated by HSS as needed, as well as information about the user's profile, such as the status of their service subscriptions and Quality of Service issues. Additional HSS functionalities include mobility management, session establishment, user authentication and access authorisation. It provides its service to the MME via the S6a interface.

- **Mobility Management Entity (MME)**

  The User Equipment and eNodeB service requests, as well as the attaching and detaching processes, are handled by the Mobility Management Entity. It connects with eNodeBs over the S1-C interface, with SPGW-C through the S11 interface, and with HSS over the S6a interface.

- **The Control Plane of the Packet Data Network Gateway (SPGW-C)**

  The controlling element of the combined Serving Gateway and Packet Data Network

Gateway is provided by the Packet Data Network Gateway's Control Plane. In other words, OAI incorporates SGW and PGW, but it also employs control and user plane separation (CUPS). The SPGW-C responds to control requests from the MME via the S11 interface and exchanges information with the SPGW-U via the SXab interface.

- **The User Plane of the Packet Data Network Gateway (SPGW-U)**

  The User Plane of the Packet Data Network Gateway manages the forwarding of user traffic between the eNodeB over the S1-U interface and the Public Data Network (PDN) at the SGi interface, which is primarily the public Internet. SPGW-C utilizes the SXab interface to manage the configuration of user traffic tunnels and through the use of the GPRS Tunnelling Protocol, it establishes communication with the eNodeB.

- **Evolved Node B (eNodeB)**

  The Evolved Node B (eNodeB) implements a base station and as a result OAI implemented software that is supported by a few Software-Defined Radio (SDR) hardwares, like the ETTUS USRP B210. The software component, which runs on a typical Linux PC, requires a low-latency real-time kernel as it is timing-critical. Therefore, running the eNodeB on a dedicated PC is highly advised.

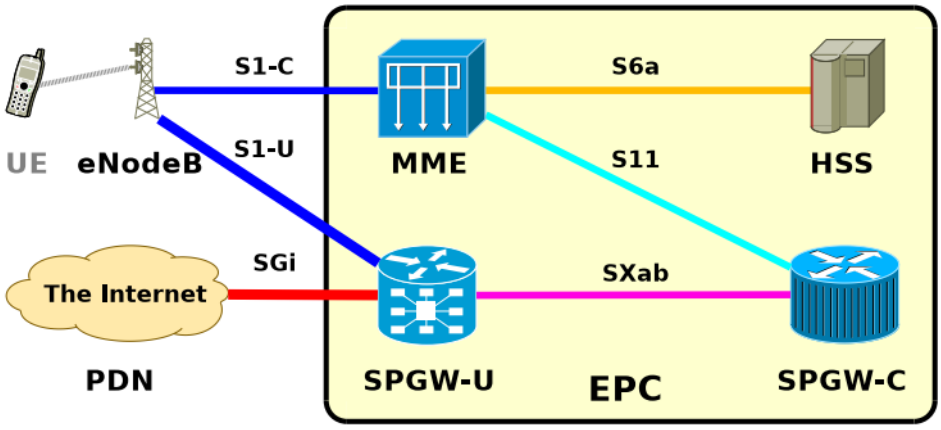

Figure 4.8: The OpenAirInterface Components [42]

4.3 Experiments

4.3.1 Experiments Overview

In this thesis, we make advantage of the Nitos Testbed's experimental resources and the OpenAirInterface open-source platform. The goal is to set up the OpenAirInterface's 4G/LTE model in a microservices environment. The Docker program is utilized to do this, allowing us to construct containers that, in our example, execute the core network's functions. The Kubernetes software is then introduced to provide the ideal network management for these containers. Finally, we provide new features that enable scalability of the SPGW-U function of this distributed network. Based on specified policy-metrics, this scaling multiplies the amount of SPGW-Us and does load balancing between them.

4.3.2 Experiment Setup

For the completion of the experiments, 5 Icarus nodes from the Indoor RF Isolated Testbed were used. As mentioned in the beginning, we are deploying the OAI 4G Long-Term Evolution platform in a microservices environment, meaning that a kubernetes cluster deployment needs to be created to distribute the network and manage the OpenAirInterface's Docker-containirized functions. 4 of the nodes mentioned above were used in the kubernetes cluster (1 master node and 3 woker nodes) deployment and 1 was used as the User Equipment indepedently.

The Kubernetes cluster operates with Flannel networking technology to allow the containers to receive a valid IP address and communicate with each other.

However, each Kubernetes pod has typically only one network interface. To overcome this problem, Multus CNI was added to the cluster, to enable pods to be connected to numerous network interfaces, instead of one. For the EPC functions and the eNodeB, Docker-containirized images were used and managed by Kubernetes. More specifically, the docker images included cassandra, HSS, MME, SPGW-C, SPGW-U, and eNodeB. These images were deployed (in that order) as Kubernetes pods and each pod was assigned by the master node to one of the worker nodes.

The management and the number of the deployed SPGW-Us, were provided by the modified Horizontal Pod Autoscaler. The concept is to use the packet reception metric (which describes the network traffic) from Kube-Prometheus and increase or decrease the number

of SPGW-Us respectively.

Kube-Prometheus was also used to monitor and keep track of the experiments. It was extensively used to store the metrics about the received packets by the SPGWUs in a time-series database. After that, these metrics were passed to Grafana for visualization and detailed analysis.

In detail, node 68 was used as the master node of the Kubernetes cluster and nodes 64, 66, 67 were used as worker nodes. Node 67 has a B210 USRP device and was used as the eNodeB of the setup. Node 86 has a Huawei E3372 dongle and acted as the UE that was used to connect to the USRP of node 67. Nodes 64 and 66 were assigned the EPC functions (cassandra, HSS, MME, SPGWC, SPGWU) by Kubernetes. The topology of the nodes used for the experiments, are presented in Figure 4.9.

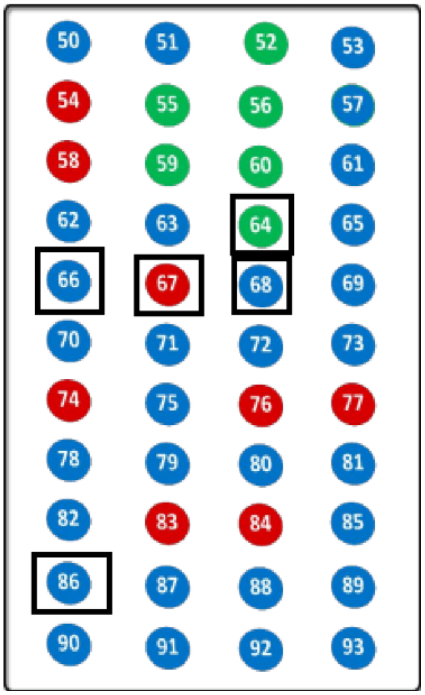

Figure 4.9: Nodes Used & Topology [40]

To complete this research, 5 different experiments were carried out. The first experiment uses the classic architecture of the OpenAirInterface's 4G/LTE model. In the remaining 4, an additional SPGW-U (and as result an extra interface) is added between the eNodeB and the SPGW-U.

Each experiment used a file containing a series of throughput speeds ranging from 1.4 - 7 Mbps. More specifically, 168 different speed values (in that range) were used, for one minute each. It is worth mentioning that the speeds increased to a certain point and decreased

respectively several times and did not have any specific pattern. The experiments started with the UE sending traffic to the PDN through the eNB's interface. The amount of traffic that was send by the UE was exclusively depending on the file with the throughput speeds that was described above. This means that each throughput speed was used by the UE for one minute until all 168 speeds were used equally. Therefore it can easily understood that each experiment lasted approximately 2 hours and 48 minutes and tested all possible scenarios in that range. The purpose of this method is to observe whether the increasing number of SPGW-Us (and interfaces) share the traffic equally and load-balancing is achieved.

More information about the exact conditions of each experiment are given in the Experimental Results section below.

4.3.3 Experimental Results

As mentioned before, the addition of new SPGW-Us, requires extra interfaces between the core network and the eNodeB. To form additional interfaces it was necessary to modify the eNodeB's source code. The default source code implements only one UDP socket, where the eNodeB acts as the server and the SPGW-U as client. For each extra interface that was added, a new socket was created to establish the new connection. Then, the source code was further modified to route traffic equally on each socket.

4.3.3.1 1 Interface

The first experiment uses the default OAI architecture described in Figures 4.8 and 4.10, in which only one interface connects the eNodeB and the SPGW-U.

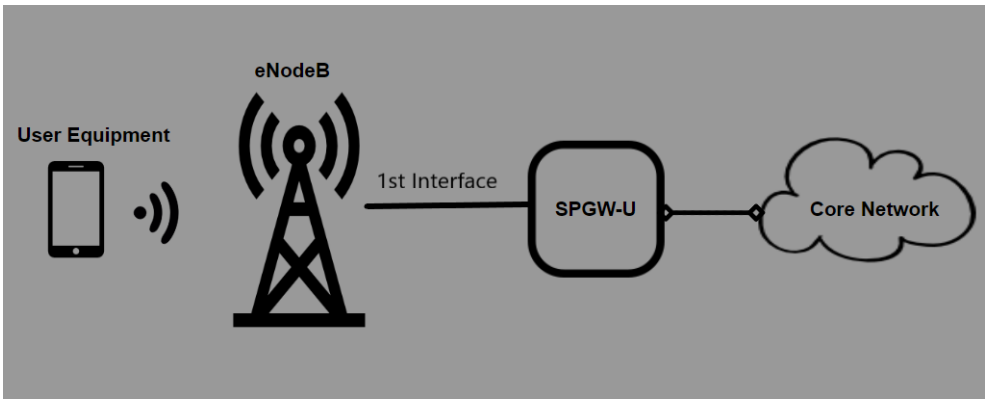

Figure 4.10: 1 Interface Diagram

Then we initiate traffic from the UE to the eNB, according to the data presented in the previous section and get a graph (Figure 4.11) that shows the packet reception rate for the 168 different speed values.

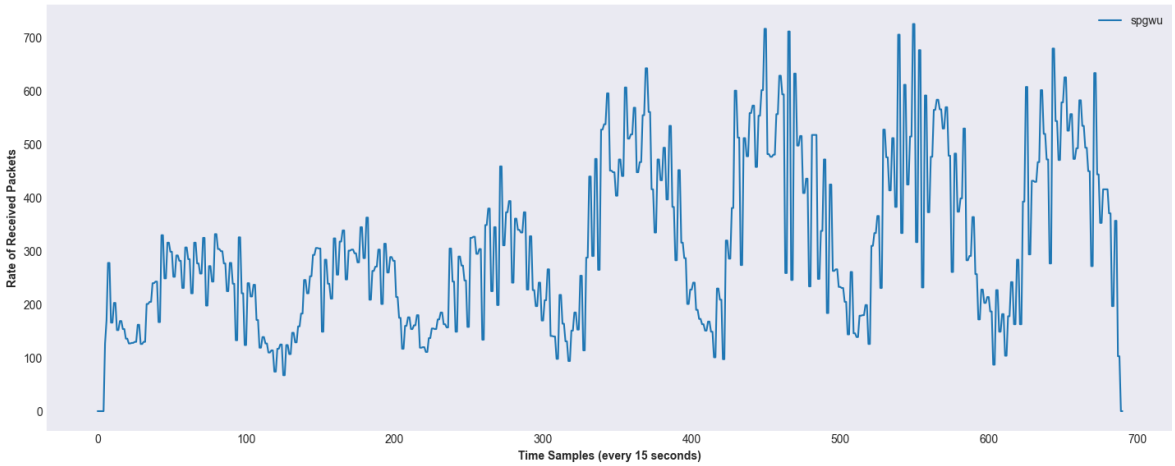

Figure 4.11: 1 Interface Measurements

It is important to mention that the blue line represents the rate of the received packets by the SPGW-U and there are 7 major peaks that reach up to approximately 700 packets. The x-axis represents the time samples, which are taken every 15 seconds.

4.3.3.2 2 Interfaces

In the second experiment a second SPGW-U is added to the network and as a result a second interface needs to be created to connect the new SPGW-U with the same eNodeB (Figure 4.12).

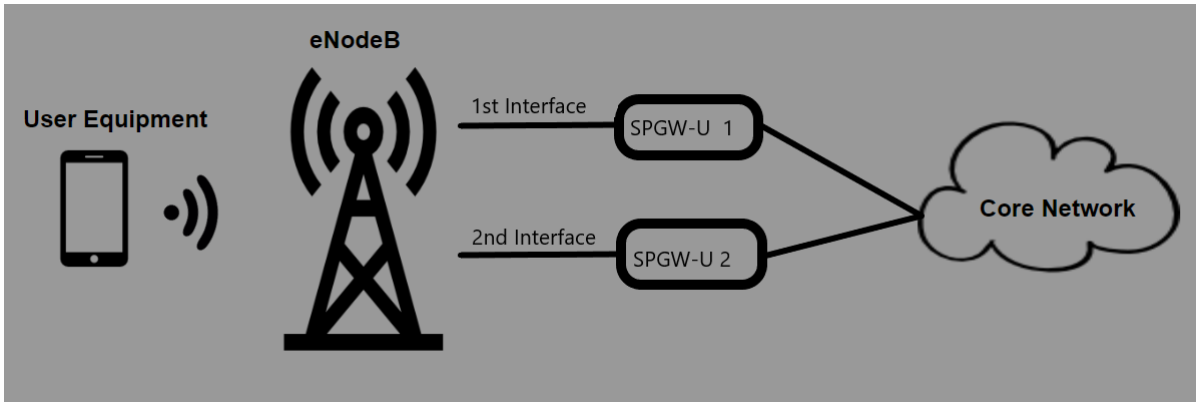

Figure 4.12: 2 Interfaces Diagram

As in the previous experiment, we initiate traffic again for these 168 throughput values, with the major difference that now we have 2 interfaces available. The generated graph is presented in Figure 4.13:

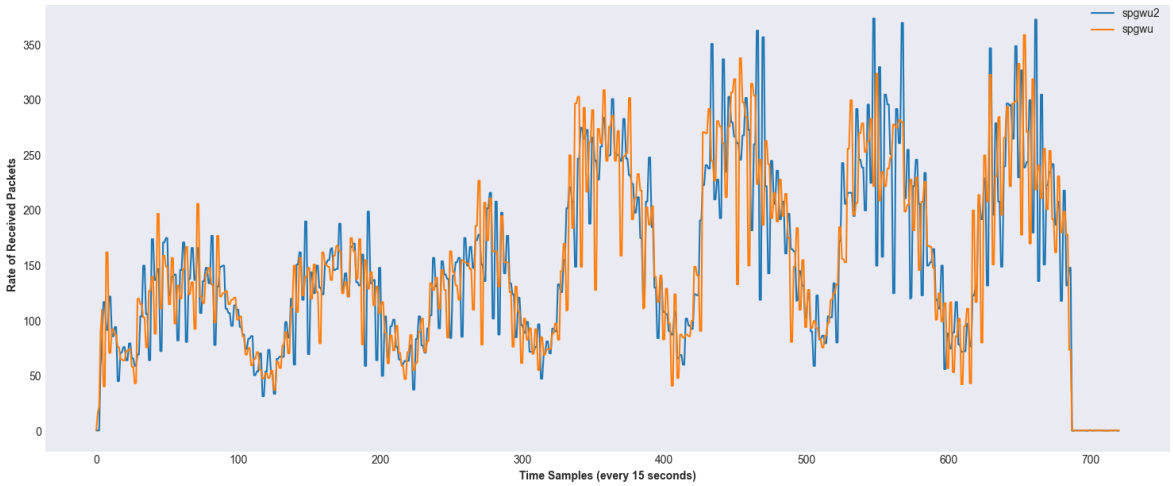

Figure 4.13: 2 Interfaces Measurements

The orange line and the blue line represent the rate of received packets by SPGW-U and SPGW-U-2 respectively. It is obvious that this graph has 7 peaks and same form as the previous one. However, the average maximum peak that is achieved now by each SPGW-U is approximately 350. This makes sense, since the packets are now forwarded equally to the 2 available interfaces. As we expected the sum of each measurement of the two SPGW-Us is equal to the corresponding value of the 1st experiment that used the default OAI setup.

4.3.3.3 3 Interfaces

In the third experiment a third SPGW-U is added to the network and as a result a third interface needs to be created to connect the new SPGW-U with the same eNodeB (Figure 4.14).

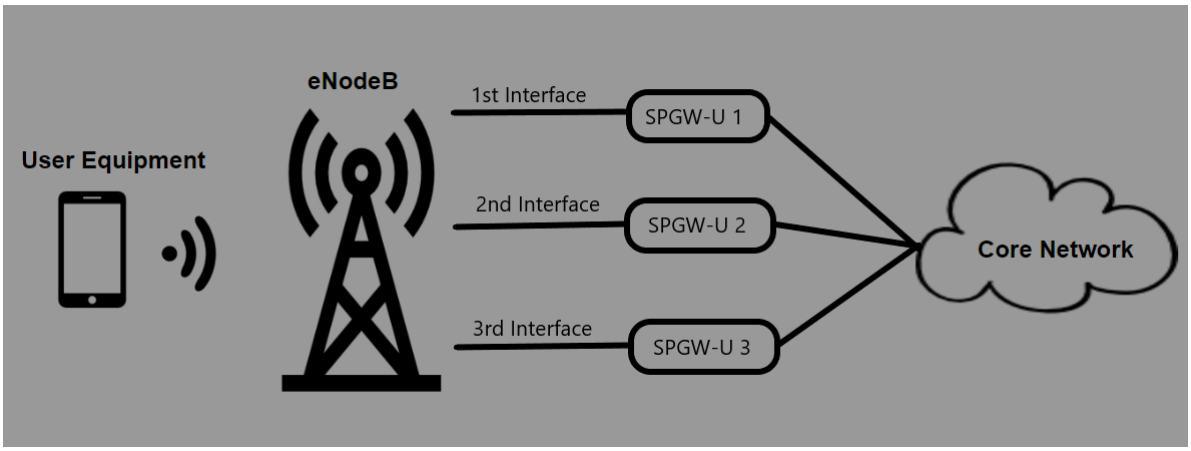

Figure 4.14: 3 Interfaces Diagram

As in the previous experiments, we send traffic again for these 168 throughput values, with the major difference that now we have 3 interfaces available. The generated graph is presented in Figure 4.15:

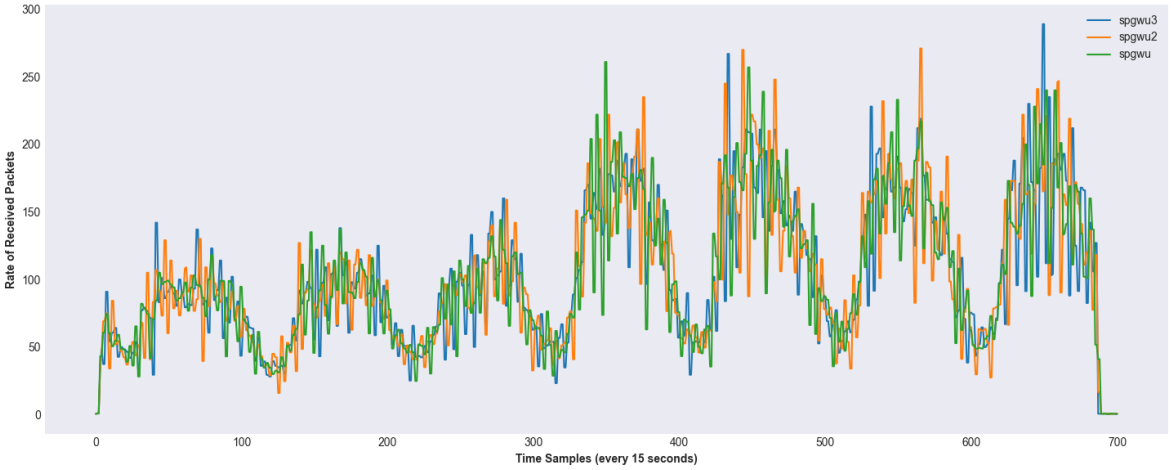

Figure 4.15: 3 Interfaces Measurements

The green,orange and the blue lines represent the rate of received packets by SPGW-U, SPGW-U-2 and SPGW-U-3 respectively. This graph has 7 peaks and same form with the previous ones as well. However, the average maximum peak that is achieved now by each SPGW-U is approximately 240. This makes sense, since the packets are now forwarded equally to the 3 available interfaces. As we expected the sum of each measurement of the three SPGW-Us is equal to the corresponding value of the 1st experiment that used the default OAI setup.

4.3.3.4 4 Interfaces

In the fourth experiment a fourth SPGW-U is added to the network and as a result a fourth interface needs to be created to connect the new SPGW-U with the same eNodeB (Figure 4.16).

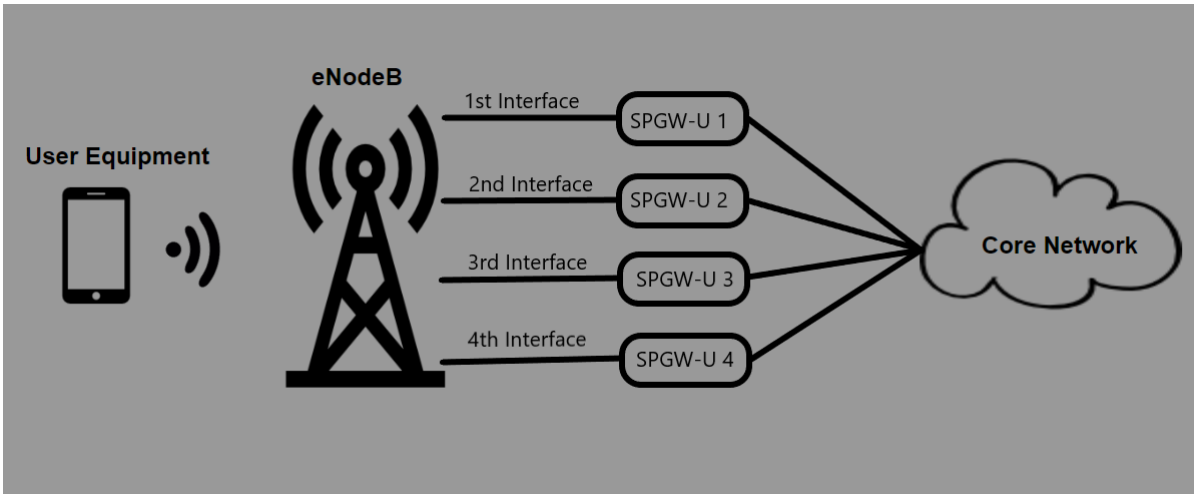

Figure 4.16: 4 Interfaces Diagram

As in the previous experiments, we send traffic again for these 168 throughput values, with the major difference that now we have 4 interfaces available. The generated graph is presented in Figure 4.17:

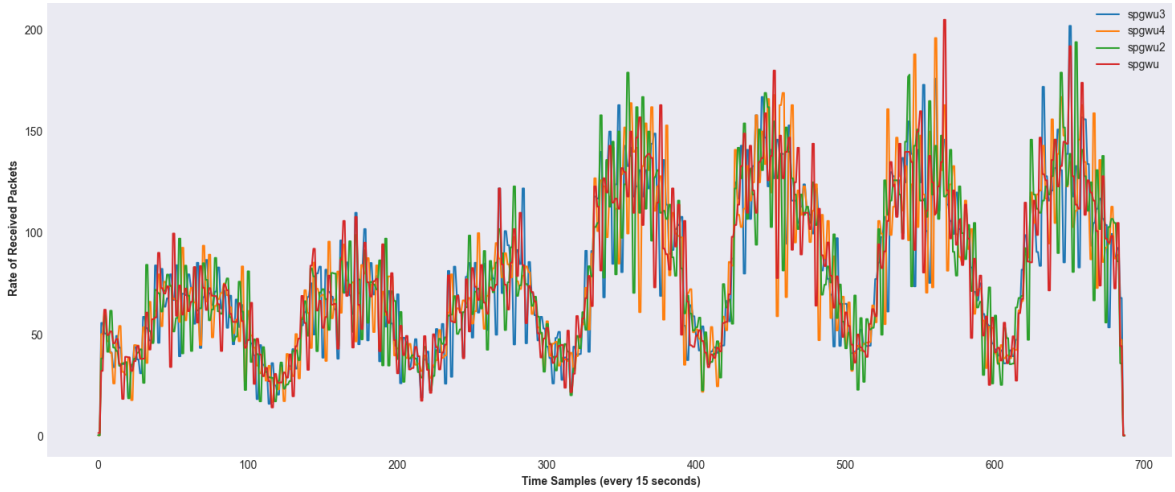

Figure 4.17: 4 Interfaces Measurements

The red, green, orange, blue and the orange lines represent the rate of received packets by SPGW-U, SPGW-U-2, SPGW-U-3 and SPGW-U-4 respectively. This graph has 7 peaks and same form with the previous ones as well. However, the average maximum peak that is achieved now by each SPGW-U is approximately 180. This makes sense, since the packets are now forwarded equally to the 4 available interfaces. As we expected the sum of each measurement of the 4 SPGW-Us is equal to the corresponding value of the 1st experiment that used the default OAI setup.

4.3.3.5 5 Interfaces

In the fifth experiment a fifth SPGW-U is added to the network and as a result a fifth interface needs to be created to connect the new SPGW-U with the same eNodeB (Figure 4.18).

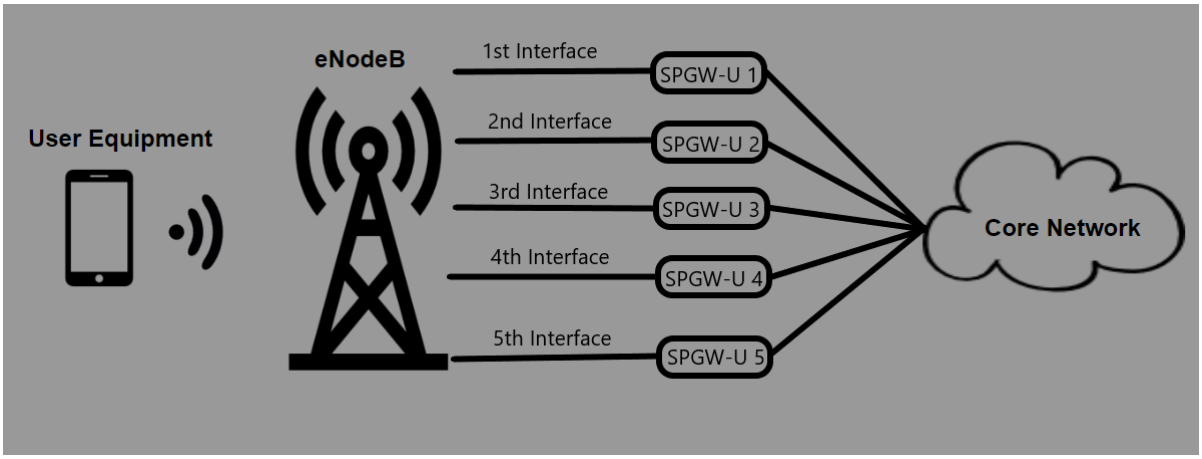

Figure 4.18: 5 Interfaces Diagram

As in the previous experiments, we send traffic again for these 168 throughput values, with the major difference that now we have 5 interfaces available. The generated graph is presented in Figure 4.19:

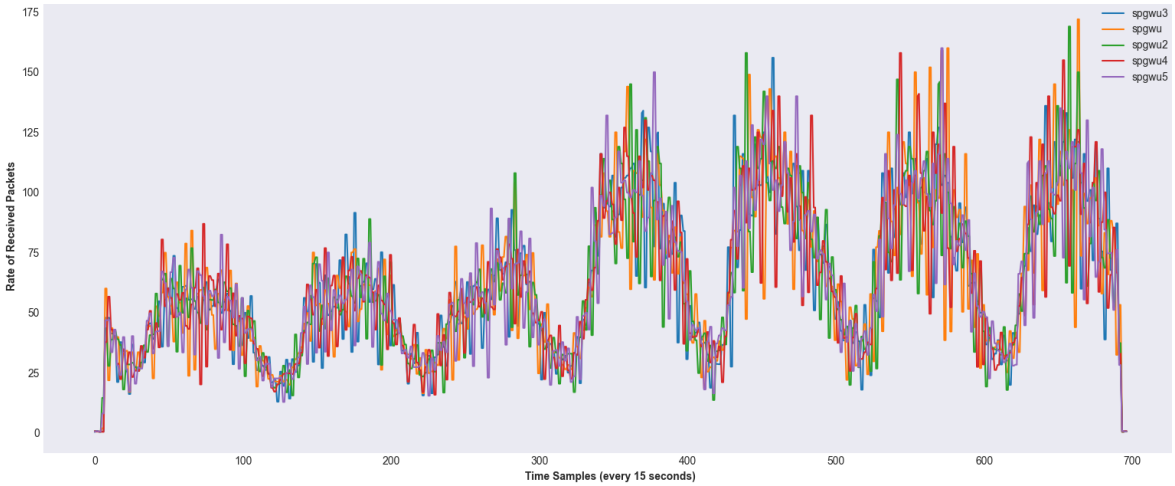

Figure 4.19: 5 Interfaces Measurements

The orange, green, blue, red and the purple lines represent the rate of received packets by SPGW-U, SPGW-U-2, SPGW-U-3, SPGW-U-4 and SPGW-U-5 respectively. This graph has 7 peaks and same form with the previous ones as well. However, the average maximum peak that is achieved now by each SPGW-U is approximately 140. This makes sense, since

the packets are now forwarded equally to the 5 available interfaces. As we expected the sum of each measurement of the 5 SPGW-Us is equal to the corresponding value of the 1st experiment that used the default OAI setup.

Chapter 5

Conclusions

5.1 Summary and Conclusions

The aim of the thesis was to deploy a fully functional 4G/LTE distributed network on an actual infrastructure, by using Kubernetes and Docker. Then, we managed to create mechanisms that allowed scaling between the base station (eNodeB) and the Core Network Function SPGW-U. This scaling was based on the rate of packet reception, a metric that was provided by kube-Prometheus. After analyzing the experimental results, we observed that every time we added an extra SPGW-U (and an extra interface), the load-balancing was successful and each new SPGW-U took an equal share of the total amount of packets received by the eNodeb.

5.2 Future Work

As future work, we examine the perspective of extending this implementation to OpenAirInterface's 5G model, due to the architecture similarities towards the OAI's 4G model. As metioned earlier, OpenAirInterface gives also the ability of deploying a 5G network on real machines. In chapter 2 we discussed that many of the CN's components of the 4G model can be easily changed to 5G's as they share many implementation similarities.

In addition, various enhancements can be considered in order to upgrade the dynamic algorithm and achieve better scaling and higher performances.

Bibliography

[1] Q. Arshad, A. Kashif, and I. Quershi, "A review on the evolution of cellular technologies," Jan. 2019, pp. 989–993. DOI: 10.1109/IBCAST.2019.8667173.

[2] https://www.statista.com/statistics/262950/global-mobile-subscriptions-since-1993/.

[3] Y. Zanjireh, S. Hoseini, and H. Niknam, *Introduction to 4G Mobile Systems*. Feb. 2016.

[4] K. Vaigandla and N. Venu, "Ber, snr and papr analysis of ofdma and sc-fdma," Jul. 2022.

[5] https://www.rfwireless-world.com/Terminology/What-is-MU-OFDMA-in-WLAN-802-11ax.html.

[6] *Mimo technology*, https://www.electronics-notes.com/articles/connectivity/4g-lte-long-term-evolution/mimo.php.

[7] https://commsbrief.com/mimo-in-lte-what-is-multiple-input-multiple-output-in-4g/.

[8] S. Todorova, "Lte terms and technology," Jun. 2022. DOI: 10.13140/RG.2.2.22167.24480.

[9] M. Islam, "Future impact of 4g on business in bangladesh," Jan. 2019.

[10] Y. Chen, T. Bretagne, and X. Lagrange, "Architecture and protocols of epc-lte with relay," Jan. 2014.

[11] https://www.rfwireless-world.com/Terminology/Difference-between-EPS-and-EPC-in-LTE.html.

[12] D. Kakadia, J. Yang, and A. Gilgur, "Evolved universal terrestrial radio access network (eutran)," in Sep. 2017, ISBN: 978-81-322-3719-8. DOI: 10.1007/978-81-322-3721-1_3.

[13] https://jwcn-eurasipjournals.springeropen.com/articles/10.1155/2010/750173/figures/1.

[14] https://www.nxp.com/docs/en/white-paper/LTEPTCLOVWWP.pdf.

[15] M. Abu-Lebdeh, F. Belqasmi, and R. Glitho, "An architecture for qos-enabled mobile video surveillance applications in a 4g epc and m2m environment," in Jan. 2016, vol. 4. DOI: 10.1109/ACCESS.2016.2592919.

[16] M. R. Sama, S. Ben Hadj Said, K. Guillouard, and L. Suciu, "Enabling network programmability in lte/epc architecture using openflow," May 2014, pp. 389–396, ISBN: 978-3-901882-63-0. DOI: 10.1109/WIOPT.2014.6850324.

[17] M. Alam, S. Samin, R. Islam, and Z. Sumiya, "5g technology," Jan. 2022. DOI: 10.13140/RG.2.2.11596.69766.

[18] N. Ana-Maria, A. Martian, and E. Popovici, "Study of millimeter waves in 5g," May 2021, pp. 1–4. DOI: 10.1109/BlackSeaCom52164.2021.9527846.

[19] https://www.pluralsight.com/guides/5g-and-what-you-should-know.

[20] S. Vahid, R. Tafazolli, and M. Filo, "Small cells for 5g mobile networks," in May 2015, ISBN: 9781118867525. DOI: 10.1002/9781118867464.ch3.

[21] K. Rikkinen, M. Juntti, V. Tapio, *et al.*, "Full duplexing," in Jun. 2017, ISBN: 9781785610615. DOI: 10.1049/PBTE069E_ch8.

[22] R. Taid, "5g core network architecture," in Apr. 2021, pp. 447–472. DOI: 10.1002/9781119778714.ch20.

[23] https://moniem-tech.com/questions/why-virtualization-is-important-for-5g-core/.

[24] *Docker*, https://docs.docker.com/get-started/overview/.

[25] https://docs.docker.com/engine/swarm/how-swarm-mode-works/services/.

[26] https://www.weave.works/blog/a-practical-guide-to-choosing-between-docker-containers-and-vms.

[27] C.-C. Chen, M.-H. Hung, K.-C. Lai, and D. Lin, "Docker and kubernetes," in Oct. 2021, pp. 169–213. DOI: 10.1002/9781119739920.ch5.

[28] https://www.edureka.co/blog/docker-architecture/.

[29] https://jfrog.com/knowledge-base/a-beginners-guide-to-understanding-and-building-docker-images/.

[30] *Kubernetes*, https://kubernetes.io/docs/concepts/.

[31] *Cluster definition*, https://www.suse.com/suse-defines/definition/computer-cluster/.

[32] https://superuser.openstack.org/articles/a-guide-to-kubernetes-etcd-all-you-need-to-know-to-set-up-etcd-clusters/.

[33] *Multus cni*, https://github.com/k8snetworkplumbingwg/multus-cni.

[34] *Flannel networking*, https://github.com/flannel-io/flannel.

[35] https://www.devopsschool.com/tutorial/kubernetes/kubernetes-cni-flannel-overlay-networking.html.

[36] *Kube-prometheus*, https://github.com/prometheus-operator/kube-prometheus.

[37] https://phoenixnap.com/kb/kubernetes-monitoring-prometheus.

[38] *Grafana*, https://grafana.com/grafana/.

[39] https://techblog.commercetools.com/adding-consistency-and-automation-to-grafana-e99eb374fe40.

[40] *Nitos testbed*, http://nitos.inf.uth.gr.

[41] *Openairinterface*, https://openairinterface.org/.

[42] T. Dreibholz, "Flexible 4g/5g testbed setup for mobile edge computing using openair-interface and open source mano," in Mar. 2020, pp. 1143–1153, ISBN: 978-3-030-44037-4. DOI: 10.1007/978-3-030-44038-1_105.