UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
FACULDADE DE ENGENHARIA ELÉTRICA E BIOMÉDICA
CURSO DE ENGENHARIA ELÉTRICA

VIRGINIA BRIOSO TAVARES

CONTAINER-BASED NETWORK FUNCTION VIRTUALIZATION OF A 5G/4G NETWORK

BELÉM-PA
2021

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
FACULDADE DE ENGENHARIA ELÉTRICA E BIOMÉDICA
CURSO DE ENGENHARIA ELÉTRICA

VIRGINIA BRIOSO TAVARES

CONTAINER-BASED NETWORK FUNCTION VIRTUALIZATION OF A 5G/4G NETWORK

Trabalho de Conclusão de Curso apresentado à Faculdade de Engenharia Elétrica e Biomédica da Universidade Federal do Pará, como requisito para a obtenção do Grau de Bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Aldebaro Barreto da Rocha Klautau Junior
Coorientador: Prof. Dra. Ana Carolina Quintao Siravenha Müller

BELÉM-PA
2021

VIRGINIA BRIOSO TAVARES

CONTAINER-BASED NETWORK FUNCTION VIRTUALIZATION OF A 5G/4G NETWORK

Este trabalho foi julgado adequado em ____/____/____ para a obtenção do Grau de Bacharel em Engenharia Elétrica, aprovado em sua forma pela banca examinadora que atribuiu o conceito _____.

_____

Prof. Dr. Aldebaro Barreto da Rocha Klautau Junior

ORIENTADOR

_____

Prof. Dra. Ana Carolina Quintao Siravenha Müller

COORIENTADOR

_____

Prof. Dr. Ilan Sousa Correa

MEMBRO DA BANCA EXAMINADORA

_____

Prof. Dr. Miércio Cardoso de Alcântara Neto

MEMBRO DA BANCA EXAMINADORA

ACKNOWLEDGEMENTS

I would like to thank the following people (without whom I would not have reached the end of this graduation):

To my parents, Antônia Brioso and Guaracy Tavares, for all the support/love they offered and for taking care of me. Paying attention to my physical and mental condition. My three sisters (Victória, Valéria and Ana Luísa), without them, this journey would have been more difficult. All the other family members who always encouraged and supported me.

To my best friend and boyfriend Gabriel Santos, thank you for all your support and for believing in me more than I believe in myself.

To my dear life friends: Jade Lima, Ivy Nunes and Camila Ferreira. For always helping me stay strong (and sane).

To my friends from the 2017.2 Electrical Engineering UFPA class, you made all my mornings happier and more fun.

To the LASSE Team (LASSE is a family) for all the knowledge and help during the nearly three years I participated. And also for the great friendships. Special thanks to Cleverson Nahum and Lucas Nóvoa who guided me and actively participated in the research that resulted in this work. And to Professor Aldebaro Klautau for believing and encouraging every member of LASSE.

All EAUFPA (eternal NPI) and UFPA professors who have been dedicated to providing me a quality public education since 2004 (13 years of NPI and 5 years of UFPA).

"Last but not least, I wanna thank me

I wanna thank me for believing in me

I wanna thank me for doing all this hard work

I wanna thank me for having no days off

I wanna thank me for, for never quitting"

“There is no limit to what we, as women, can accomplish.”
(Michelle Obama)

ABSTRACT

The 5G aims to use technologies to generate more evolved and scalable networks. Network function virtualization is one of the keys to 5G as it allows the 5G network infrastructure to be changed, managed, monitored and optimized more effectively and efficiently. 5G Core requires the Cloud-Native approach to decompose functions into microservices. These microservices are network functions in containers, making the Core Network more dynamic, configurable and agile. This work presents creation and container-based virtualization of a 4G/5G testbed with SD-RAN and its validation through performance analysis of network scenarios. The contribution of this work is the creation of a low-cost and container virtualized 4G/5G environment that can be used for research purposes.

Keywords: 5G, 4G, Container, Virtualization, Orchestration

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

1G	1st Generation
2G	2nd Generation
3G	3rd Generation
3GPP	3rd Generation Partnership Project
4G	4th Generation
5G	5th Generation
5GC	5G Core Network
AMF	Access And Mobility Management Function
AMPS	Advanced Mobile Phone System
API	Application Programming Interface
AUSF	Authentication Server Function
BBU	Baseband Unit
C-RAN	Centralized Radio Access Network
CDMA	Code Division Multiple Access
CNF	Cloud-Native Network Functions
DN	Data Network
E-UTRAN	Evolved Universal Terrestrial Radio Access Network
EDGE	Enhanced Data Rate For GSM Evolution
eNodeB	Evolved Node B
EPC	Evolved Packet Core
ETSI	European Telecommunications Standards Institute
FDMA	Frequency Division Multiple Access
G	Generation
gNB	5G Base Station
GSM	Global System For Mobile Communication
HSS	Home Subscriber Server
HTTP	Hypertext Transfer Protocol
IMSI	International Mobile Subscriber Identity

ISG	Industry Specification Group
K8s	Kubernetes
LTE	Long Term Evolution
LXC	Linux Containers
MAC	Media Access Control
MANO	Management And Orchestration
MCC	Mobile Country Codes
MME	Mobile Mobility Entity
MMS	Multimedia Message Service
MNC	Mobile Network Code
NAS	Non-Access Stratum
NF	Network Function
nFAPI	Network Functional API
NFV	Network Function Virtualization
NFVI	NFV Infrastructure
ng-eNB	Next Generation ENodeB
NG-RAN	Next Generation Radio Access Network
NGFI	Next Generation Fronthaul Interface
NR	New Radio
NRF	Network Function Repository Function
NSSF	Network Slice Selection Function
OAI	OpenAirInterface
OFDMA	Orthogonal FDMA
OP	Operator Code
OS	Operational System
P-GW	Packet Data Network Gateway
PCF	Policy Control Function
PCRF	Policy And Charging Rules Function
PDN	Packet Data Networks
PDU	Protocol Data Unit
PHY	Physical Layer
PLMN	Public Land Mobile Network

RAN	Radio Access Network
RCC	Radio Cloud Center
REST	Representational State Transfer
RRH	Remote Radio Head
RRU	Remote Radio Unit
S-GW	Serving Gateway
SBA	Service-Based Architecture
SC-FDMA	Single Carrier - FDMA
SD-RAN	Software-Defined Radio Access Network
SDN	Software-Defined Networking
SMF	Session Management Function
SMS	Short Message Service
TDMA	Time Division Multiple Access
TLS	Transport Layer Security
UDM	Unified Data Management
UE	User Equipment
UI	User Interface
UPF	User Plane Function
VM	Virtual Machine
VNF	Virtualized Network Function
WWW	World Wide Web

CONTENTS

1 INTRODUCTION

The 5G networks are the future of telecommunications system, allowing new features for different use cases. A major challenge for the successful launch of 5G and the delivery of the quality of service expected for various use-cases of 5G — voice, data, commerce, health, IoT, surveillance — is the necessity for a robust virtualization technology, with high programmability, to attend different scenarios. The Cloud-Native Network Functions approach appears to accelerate the implementation of specifications to fulfill the 5G promise. Its main objective is to virtualize network services and abstract them from dedicated hardware. Thus, enabling the use of network resources without worrying about their physical located, cost and organization.

1.1 Objective

1.1.1 Geral Objective

The study goal is to develop a virtualized mobile network with 5G Core associated with 4G Radio Acess Network, as well as additional features such as Software-Defined Radio Access Network (SD-RAN) and User Equipment (UE) emulation. Virtualization occurs through containers that perform network functions, these containers are orchestrated and connected according to the network architecture. This solution is aimed at research environments that want the use of multiple 4G/5G network scenarios, that is, it allows the provision of a virtualized and private 5G/4G network. Thus providing an accessible environment that, in an intuitive and automated way, will allow the user easy access.

1.1.2 Specific Objectives

- Characterize the generations of mobile networks (focusing on 4G and 5G).
- Discuss container creation and orchestration.
- Propose a 4G/5G network (softwares and architecture).
- Optimally mount the proposed mobile network using containers.
- Analyze possible scenarios of the mounted network and validate them.

1.2 Related Works

The theme developed is result of the initial work [TAVARES et al., 2019], on this paper a 4G mobile network was created using Open-Source software. From that initial contact, there was a deepening in relation to the way the hardware is used, with a focus on the virtualization of

mobile networks through containers, and also on 5G technology. The scope of this work details this process of virtualization and use of Open Source software related to 4G and 5G technologies. The 4G/5G testbed generated in this work has already enabled the publication of five articles in scientific journals. [NAHUM et al., 2020] presents how the 4G/5G mobile network testbed can be integrate to artificial intelligence applications and its use cases. [CASTRO et al., 2020] introduces the testbed implementation that is licensed to operate for scientific and experimental purposes in the 700 MHz band. [NOVOA et al., 2020] focus RAN slicing in C-RAN scenario, the testbed was used for network implementation. [TAVARES et al., 2020] presents presents a simple machine learning workload using Kubeflow tool, which chooses the best scenario for the Network Function (NF) placement on the testbed. [GUERREIRO et al., 2020] describes an Elastic Stack implementation in 4G/5G network, the testbed was used for network creation. None of these papers sought to show how the containerization of the network was effectively carried out, which is the proposal of this thesis.

1.3 Outline

The remainder of this work is outlined as follows:

- **Chapter 2:** Provides an overview of the 4G and 5G system, describing the components that make up the architecture of each; presents ideas that have been highlighted in 5G such as virtualization, containerization and orchestration.
- **Chapter 3:** Elaborates on the procedure for creating an automated 4G/5G testbed in a laboratory environment at the Federal University of Pará; detailing the open-souce software used and how they were used, as well as how this network was automated to allow the creation of multiple scenarios.
- **Chapter 4:** Presents three created scenarios and the evaluation of each one, aiming to validate the use of the created network.
- **Chapter 5:** Synthesizes the work and presents final contributions.

2 BASIC CONCEPTS

2.1 Evolution of Mobile Wireless Technology

From the early 1970s, the wireless mobile industry began creating its communication network technology. In the 1990s, mobile communication experienced an explosive growth, and since then it has become a part of peoples' daily routines. Mobile wireless communication network Generation (G) refers to a change in the nature of the mobile wireless communication, such as the nature of the system, speed, technology, frequency, data capacity, latency, etc. Also, each Generation has certain standards, different capabilities, new techniques and new features that distinguish it from the previous one. Mobile wireless communication networks have undergone remarkable changes, this section is intended to provide a brief overview of each generation (1G to 5G).

2.1.1 First Generation

The origin of mobile networks is known as 1st Generation (1G), it was utilized only for voice calls. The first handheld mobile cell phone was demonstrated by Motorola in 1973, and the first commercial automated cellular network was launched by NTT in Japan in 1979. Handheld cell phones became the technology of the moment in the early 1990s [GAWAS, 2015]. 1G was based on the Advanced Mobile Phone System (AMPS), it was frequency modulated and used Frequency Division Multiple Access (FDMA) with a channel capacity of 30KHz, speed up to 2.4kbps, and frequency band of 824 to 894MHz [KHARBULI; SULTANA, 2018]. 1G was characterized by low capacity, unreliable transmission, weak voice calls, and no security. This lack of security was due to the fact that voice calls were played over radio towers, making these calls susceptible to unwanted eavesdropping by third parties [VORA, 2015].

2.1.2 Second Generation

In the 1990s, 2nd Generation (2G) came up with voice transmission using digital signals along with data, this was possible due to the fact that 2G introduced the packet-switching. Global System for Mobile Communication (GSM) is the most widely used 2G mobile standard. Voice calls were encrypted and computerized, calls had become completely clearer. Significant features were Short Message Service (SMS), Multimedia Message Service (MMS), Email, as well as World Wide Web (WWW) access. 2G speed is 64kbps and the bandwidth is 30-200KHz. To communicate more and more users multiple access techniques are used, i.e. FDMA, Code Division Multiple Access (CDMA) and Time Division Multiple Access (TDMA). This generation implemented the concept of CDMA [SALIH et al., 2020]. To provide better service, the wireless mobile industry developed the advanced system called 2.5 Generation (2.5G), it includes voice with data and this process progresses enhanced data rate for Enhanced Data Rate

for GSM Evolution (EDGE). It implements a packet switched domain, and provides a data rate of 144kbps [VORA, 2015].

2.1.3 Third Generation

3rd Generation (3G) was launched in 2000 with the purpose of offering high speed data. A 3G mobile system was defined by an organization called 3rd Generation Partnership Project (3GPP). 3G's expanded information transmission abilities are multiple times quicker than 2G [SALIH et al., 2020]. The features increased are bandwidth and data transfer rate to fit a web-based application, audio, video file and ended IP. It has a bandwidth of 15-20MHz used for high speed internet, video chatting, etc. This generation offered a peak rate of 100-300 Mbps. 3G offers more advanced services to the users compared to the previous generations. Along with voice communication it includes data services, access to TV/videos, Web browsing, e-mail, video conferencing, paging, fax and navigational maps [GAWAS, 2015].

2.1.4 Fourth Generation

4th Generation (4G) is an IP based network system introduced in the late 2000s. The reason for the transition to all IP is to have a common platform to all the technologies developed so far. The first successful field trial for 4G was conducted in 2005 [GAWAS, 2015]. 4G purpose was to provide high speed, high quality, high capacity, security and low cost services for voice and data services, multimedia and internet over IP. 4G enables 10Mbps-1Gbps speed, it provides the same features as 3G, and some additional services with more clarity. It sends data much faster than previous generations [VORA, 2015]. Long Term Evolution (LTE) is considered the 4G technology, it uses a different form of radio interface (Orthogonal FDMA (OFDMA)/Single Carrier - FDMA (SC-FDMA) instead of CDMA), but there are many similarities with the earlier forms of 3G architecture and opportunities for re-use of some elements of 3G network architecture [HOLMA; TOSKALA, 2009].

2.1.5 Fifth Generation

5th Generation (5G) is the next generation of wireless mobile broadband technology. 5G enables a new kind of network that is designed to connect virtually everything and everyone together, including machines, objects, and devices. 5G wireless technology is intended to deliver higher multi-Gbps peak data speeds, ultra-low latency, greater reliability, massive network capacity, greater availability, and a more seamless user experience for more users. Higher performance and greater efficiency empower new user experiences and connect new industries [GAWAS, 2015]. 5G radio frequency ranges have been extended to meet the need for improved data and performance. They now include all those previously held by 4G, as well as more frequencies up to 6GHz (Sub-6) and the high band (mmWave) spectrums. Sub-6 5G capacity could theoretically run out in mature markets by 2023, as a direct result of increased data

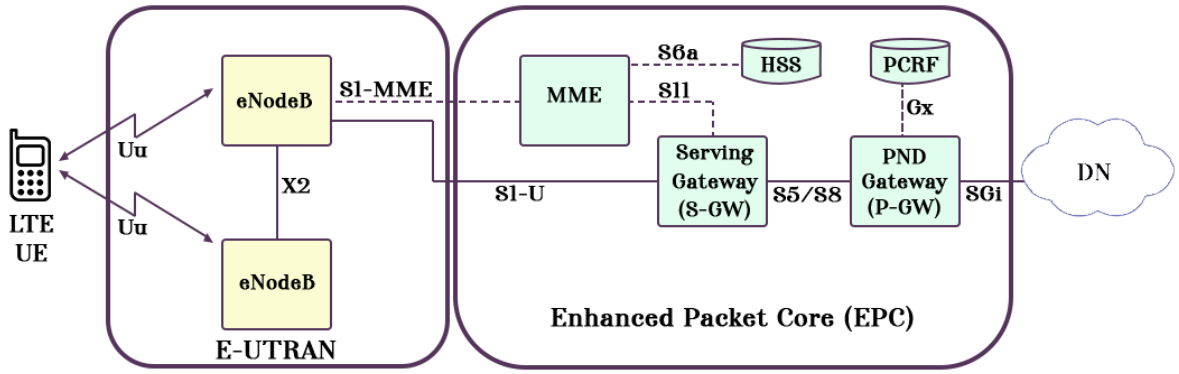
consumption. This would effectively make 5G mmWave a valuable resource for the continued provision of enhanced mobile broadband services [ERICSSON, 2021].

2.2 4G and 5G Network

2.2.1 Network Architecture of 4G

A 4G network is composed of several functional entities, whose functions and interfaces are defined. A LTE network can be divided into two parts: Evolved Packet Core (EPC) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN). Each component is described in more detail next. Fig 1 shows that UE (e.g. smartphones) connect through E-UTRAN to EPC and later to Data Network (DN), such as the Internet.

Figure 1 – Illustration of 4G architecture, highlighting its main components.



Source: Author

2.2.1.1 Evolved Packet Core

EPC was first introduced by 3GPP in Release 8 of LTE standard, it is the LTE Core Network. EPC offer multiple functionality such as authentication, mobility management, session management, and Quality of Services [3GPP, 2008c]. EPC functions are green highlighted in Fig 1. According to [3GPP, 2008b], the EPC network functions are characterized as:

- **Home Subscriber Server (HSS):** This is a database that contains user-related and subscriber-related information. It also provides support functions in mobility management, call and session setup, user authentication and access authorization.
- **Mobile Mobility Entity (MME):** This functional entity serves the UE providing mobility and session management. The MME is connected to the HSS over the S6a interface. The

Non-Access Stratum (NAS) is used to convey non-radio signalling between the UE and the MME. The S1-MME and S11 interfaces are used for the Control Plane between the MME - E-UTRAN and MME - S-GW, respectively.

- **Policy and Charging Rules Function (PCRF):** The PCRF supports service data flow detection, policy enforcement and flow-based charging. The Gx interface is responsible for the connection between PCRF and P-GW.
- **Packet Data Network Gateway (P-GW):** This entity is communicates with the outside world through, i.e. Packet Data Networks (PDN). This communication occurs using the SGi interface. The P-GW includes functionality for IP address allocation, charging, packet filtering, and policy-based control of user-specific IP flows.
- **Serving Gateway (S-GW):** The S-GW forwards data between the base station and the P-GW, also it terminates the S1-U User Plane interface towards the base stations. The interface between the S-GW and P-GW is called S5/S8, in principle S5 and S8 is the same interface, the difference is that S8 is used when roaming between different operators while S5 is network internal. The S1-U Interface is used for the User Plane between the S-GW and E-UTRAN.

2.2.1.2 Evolved UMTS Terrestrial Radio Access Network

As stated by [3GPP, 2008a], E-UTRAN is composed by a set of Evolved Node B (eNodeB) and UEs. ENodeBs are base stations that serve as access infrastructure for end users. An eNodeB has several functions, the main ones are: radio bearer control, radio admission control, connection mobility control, and dynamic allocation of resources to UEs in both uplink and downlink. X2 interface allows interconnection between eNodeBs, thus providing the exchange of information necessary in carrying out operations such as handoff and spectrum management. UE and eNodeB are associated by Uu wireless interface. The eNodeBs, UE and interfaces are showed in Fig 1.

2.2.2 Radio Access Network

According to [ERICSSON, 2020], Radio Access Network (RAN) is a mobile network part that connects individual devices to other network components through radio connections. A traditional RAN consists of three main components: baseband, radio and antennas. Baseband is composed of signal processing functions that make wireless communication possible. Radio is the part responsible for converting digital information into wireless signals, as well as making sure that the transmitted signals are in the correct frequency bands and have the correct power levels. Antenna is responsible for radiate electrical signals in radio waves. Typically, RAN is used to refer to a group of base stations.

2.2.2.1 Centralized Radio Access Network

Centralized Radio Access Network (C-RAN) was introduced by China Mobile Research Institute in 2010, it is a new cloud computing architecture for 5G mobile network. C-RAN has the goal to be very flexible, so that they can be adopted to cater to a wide range of use case scenarios [HSU, 2020]. It allows the use and sharing of a lot of resources for an efficient radio signal processing. In addition to centralized processing, it also allows collaborative radio and real-time radio networking [BOULOS, 2019]. This architecture is made of a Baseband Unit (BBU) and a Remote Radio Head (RRH), these are two functions that work individually. When base station receives the signal, RRH retrieves the signal from antenna and creates an analog transmitted radio frequency for BBU. After that, BBU analyzes the original frequency range of transmission signal and then modulates, generates and processes the digitized signal. Overall, BBU handles most of the digital processing and RRH is responsible for the analog part [ERICSSON, 2020].

The Next Generation Fronthaul Interface (NGFI) has been formed to standardize the fronthaul interface for future 5G cellular networks. According to NGFI, BBU is redefined as Radio Cloud Center (RCC) and RRH is known as Remote Radio Unit (RRU) [ZHILING, 2015]. This work will use the NGFI terminology. In broad terms, C-RAN redesign the RAN to achive its goal, for that it separates the RRU from the RCC into two different entities. A RCC module can connect to multiples RRUs. All received signals by RRUs are sent through a transport network, called Fronthaul, to a RCC module in order to be processed.

2.2.2.2 Software-Defined Radio Access Network

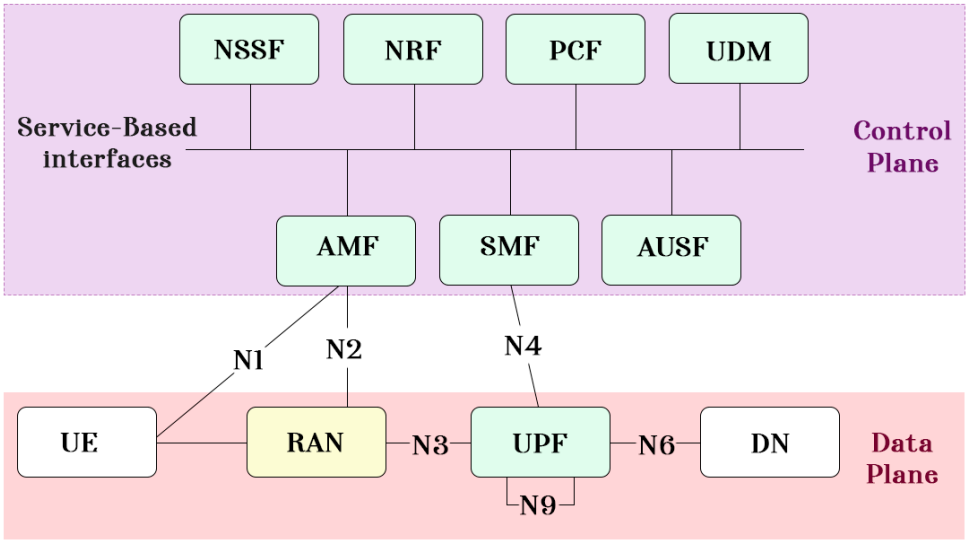
SD-RAN [LU et al., 2015] is receiving a lot of attention in 5G networks, since it offers a more flexible and programmable RAN. Software-Defined Networking (SDN) [RISCHKE; SALAH, 2020] is an emerging architecture that aims to make the network more flexible and easier to manage. A fundamental characteristic of SDN is the separation of the data forwarding function and Control Planes, this decoupling make the network control become directly programmable [GöRANSSON; BLACK; CULVER, 2017]. An SDN architecture consists of a Controller to manage flow control, Southbound APIs to relay information between Controller and individual network devices (such as switches, access points, routers, and firewalls), Northbound APIs to relay information between Controller and applications [CISCO, 2020]. SD-RAN refers to RAN making use of SDN ideas, thus allowing actions such as Control Plane and Data Plane separation, centralized and real-time control, etc.

2.2.3 Network Architecture of 5G

Compared to previous generations, 5G architecture is defined as a Service-Based Architecture (SBA) [3GPP, 2017b]. A SBA is an architecture whereby the Control Plane functionality and common data repositories of a 5G network are delivered by interconnected network functions that have authorization to access each other's services. For the interconnection between network

functions, 3GPP has defined the usage of Application Programming Interface (API) over the Hypertext Transfer Protocol (HTTP) protocol [MAYER, 2018]. These APIs follows the Representational State Transfer (REST) model, it is a model style for providing standards between computer systems on the web, making it easier for systems to communicate with each other. SBA allows the use of cloud resources, orchestration and Edge Computing to achieve better efficiency and stability [3GPP, 2017a]. 5G is based on the most successful 4G architecture entities and adds features for higher capacity and low latency network. The 5G idea was to separate the user data (Data Plane) and the signaling (Control Plane) to enable network independent scaling. Fig 2 indicates an overview of 5G architecture, it point out 5G network functions and interfaces, also it group network functions according to Data Plane and Control Plane separation.

Figure 2 – Overview of a 5G architecture.



Source: Author

2.2.3.1 5G Core

The 5G Core Network (5GC) entities are responsible for tasks such as managing user authentication, session establishment, routing, radio network interface, UE interface, mobility, tunneling with data network, among other products. These entities are highlighted in green in Fig 2. The most significant differences between the EPC and the 5GC is the separation of Control Plane and User Plane that is accomplished in the 5GC, specific functions are created to handle these [OLIVEIRA; ALENCAR; LOPES, 2018]. Below is a list of the new main 5GC functions and their descriptions, according to [MAILER et al., 2020].

- **User Plane Function (UPF):** The UPF receives User Plane tasks from EPC entities S-GW and P-GW, such as RAN packet routing to external services, Protocol Data Unit (PDU) session anchor (i.e. handle connections between UE and DN), QoS enforcement and

charging. It is a network function that manages user traffic, allowing data forwarding to be deployed and scaled independently so that packet processing and traffic aggregation can be distributed. UPF has four main interfaces: N3, N4, N6 and N9. N3 is the User Plane interface between RAN and 5GC. N4 connects UPF to SMF (allowing session management). N6 transports UPF data to the Internet. And N9 connects two Core UPF (if any).

- **Access and Mobility Management Function (AMF):** This function inherits tasks from EPC MME, receives all connection and session information from UE or RAN, but it handles only UE registration and mobility management tasks. AMF also terminates NAS signaling. AMF has two interfaces: N1 and N2. The first transfers information from the UE to the AMF. While N2 is the Control Plane interface between RAN and 5GC.
- **Session Management Function (SMF):** SMF receives S-GW and P-GW Control Plane tasks, such as UPF selection and UE IP address assignment. In general terms, it is responsible for interacting with the decoupled data plane by creating, updating and removing PDU sessions and managing the session context within the UPF.
- **Unified Data Management (UDM):** Centralized element that manages network user data. It supersedes EPC HSS.

However, still additional network functions that extend as features of the 5GC, such as:

- **Authentication Server Function (AUSF):** It is responsible for executing the security procedure for the correct authentication of the end users' SIM card.
- **Policy Control Function (PCF):** Provides policy rules to Control Plane functions. It contains all the session-related functions of the EPC PCRF, and add new control mechanisms for UE activities.
- **Network Function Repository Function (NRF):** It maintains the list of available network functions instances and their profiles.
- **Network Slice Selection Function (NSSF):** Select of the network slice instances to serve the end users.

2.2.3.2 Next Generation Radio Access Network

The 3GPP 5G Next Generation Base Station is named 5G Base Station (gNB), it uses 5G Radio Access technology called New Radio (NR). 5G NR was introduced in 3GPP release 15, this report covers most physical layer definitions. 3GPP Release 16 introduces new layers, the use of millimeter-wave from high GHz channels, and unlicensed spectrum [3GPP, 2019]. In relation to LTE, the NR showed an increase in values related to technical aspects. Conforming

to [HÅLAND, 2019], Next Generation Radio Access Network (NG-RAN) provides both NR and LTE radio access. NG-RAN is composed by nodes, each node is a Base Station, it can be gNB or Next Generation eNodeB (ng-eNB) type. The ng-eNBs refer to a version of eNodeB capable of being connected to 5GC [BERTENYI et al., 2018].

2.3 Network Functions Virtualization

According to [NASCIMENTO, 2013], virtualization is defined as logical abstraction of the underlying hardware devices within a network, through software implementation. The abstraction is limited to hardware, software embedded into hardware can also been virtualized as independent virtual function elements. Virtualization allows to decouple or disassociate logical resources from the underlying physical resources. Heterogeneity, scalability, and interoperability are consequences of this technique [ALAM et al., 2020]. The topics below provides details about the most common approaches for isolation for running services, and how do these virtualization techniques provide isolation during runtime.

a) Virtual Machines

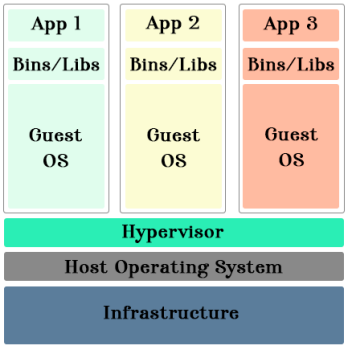
The Virtual Machine (VM) application realizes a software application implementation of a machine architecture, computing platform, or execution environment. For the purpose of provide machine level isolation, the VM requests physical machine virtualization hardware on which it runs (usually called "host machine") [MAHESHWARI et al., 2018]. A VM uses specialized software, called a "hypervisor", capable of emulating hardware resources. This specialized software allows host machine to support multiple guest VMs by virtually sharing its resources, such as memory and processing. By using the resources of a single machine efficiently it is possible to reduce limits cost of additional hardware, consequently, reducing power, cooling requirements, and human resource management efforts [VMWARE, 2019]. Fig 3a illustrates three applications, each contained within a VM. Each VM has its own operating system and is completely isolated from the host operating system (Guest OS). Each application has its own binaries, libraries and applications. In the figure is possible to noticed the hypervisor layer acting as an intermediary between the infrastructure and the VMs.

b) Container

Containers are virtual environments for developing and deploying applications oblivious to the infrastructure. The virtual environments are packed through a specific code that describes the dependencies to run applications and services in a virtualized way. Each container are isolated from each other, and they share access to the Operational System (OS) and kernel of the host machine [BONATI et al., 2020]. Fig 3ab demonstrate three apps running inside containers. To run the apps each container requires

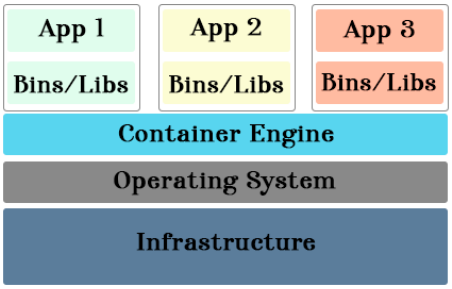
are apps related bins, libraries and other runtime components. The Container Engine (the runtime that enables container applications) run on top of Operating System.

(a) Virtual Machine.



Source: Author

(b) Container.



Source: Author

Figure 3 – Main virtualization approaches.

2.3.1 Container Virtualization

The act of containerizing an application refers to bundling that application with the libraries, dependencies, and configuration files needed to create it. In other words, packaging an application to run efficiently in multiple computing environments. This work uses container as the approach to virtualization. The reasons for choosing this approach were that the use of containers generates effective resource sharing, platform independence, smooth scaling and predictable environment [RODRIGUEZ; BUYYA, 2020]. Thus, this subsection describes three Open-Source software/platforms related to container virtualization and orchestration.

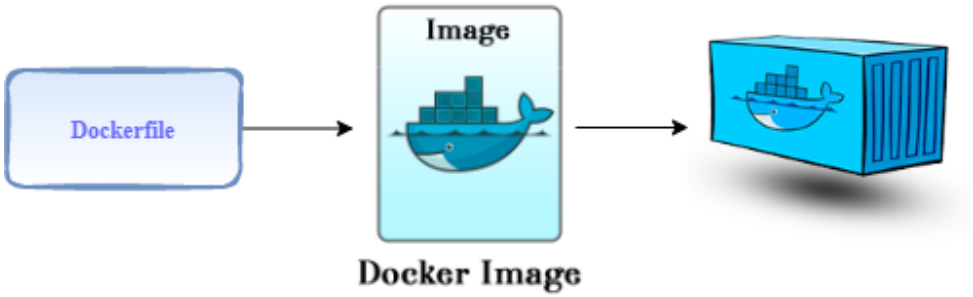
2.3.1.1 Docker

Docker [DOCKER, 2021a] is an open platform that relies on free existing kernel resources to isolate a running process. It all started in early 2013 with an Open-Source project at dotCloud, a cloud-centric platform-as-a-service company. Docker is a lightweight virtualization based on Linux Containers (LXC), this is an operating system level virtualization technology that creates a virtual sandbox environment in Linux [NEDU; PUTTARAJU, 2016]. The size of a Docker container, in relation to storage and CPU usage, is much smaller than that of a VM because it only replicates libraries and binaries of the application that is being virtualized. In general terms, Docker extends the LXC technology, providing a simple toolset and an unified API to manage some technologies at kernel level. Docker container defines a standardized way to deliver software [MERKEL et al., 2014].

Docker images become containers when they run on the Docker Engine. A Docker image is a read-only template that allows you to package an application so that it can run in

a development or production environment (on a server) [DOCKER, 2021b]. This image is represented by one or more files, where the main file is the Dockerfile and can pass other files to the container. Dockerfile is a text document that contains all the commands an user can invoke from the command line to mount an image [DOCKER, 2021c]. Figure 4 illustrates the construction of a Docker Container. These images can be located locally or can be uploaded to repositories like the Docker Hub.

Figure 4 – Example of a Docker usage.

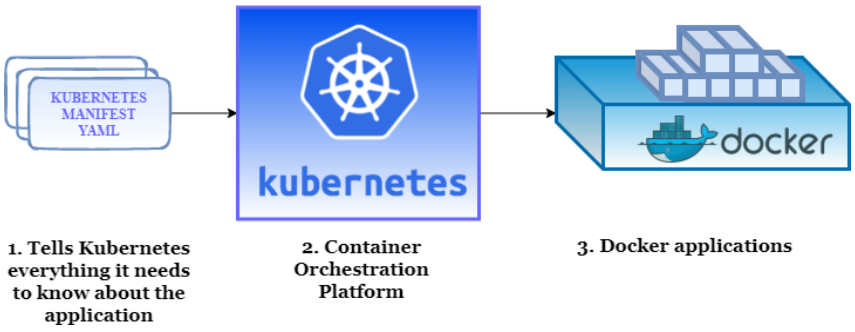


Source: Author

2.3.1.2 Kubernetes

In June 2014, Google Lab announced Kubernetes (K8s) [KUBERNETES, 2021a], a source cluster manager for Docker containers. Kubernetes premise is to be an Open-Source system capable of decoupling containers from the systems they run on. This decoupling simplifies application development and also simplifies data center operations. Kubernetes enables container orchestration, which automates the deployment, management, sizing and networking of these containers. It's perfect for companies that need to deploy and generate containers [BERNSTEIN, 2014]. Fig 5 illustrate a Kubernetes application creation.

Figure 5 – Kubernetes usage.



Source: Author

K8s cluster is a group of machines, where each machine is a "node", which will be used to run applications in containers. Clustering provides the main advantage of Kubernetes: ability to schedule and run containers on a set of physical, virtual, on-premises or cloud machines.

Kubernetes containers are not tied to individual machines; in fact, they are abstracted across the entire cluster [IBM, 2021]. A cluster must contain at least one node and Control Plane — the single-cluster mode is when these two are on the same machine. Control plane is responsible for the maintenance and organization of the cluster, the node that hosts the Control Plane is called "master node", while the other node are the "workers". Kubernetes does not run containers directly, instead it wraps one or more containers in a higher-level structure called a pod. Any container in the same pod will share the same resources and local network [KUBERNETES, 2021b]. Kubernetes is configured to automatically control the cluster to match its settings. These settings are passed through manifest-formed configuration files, usually JSON or YAML files, which declare the type of application to be run and details about this application. The command line (kubectl) or the Kubernetes API allows interaction with Kubernetes to set or modify the cluster state. The main types of Kubernetes application are deployment and service. The deployment can create and destroy pods dynamically, while service connect a set of pods to an abstracted service name and IP address [KUBERNETES, 2021c].

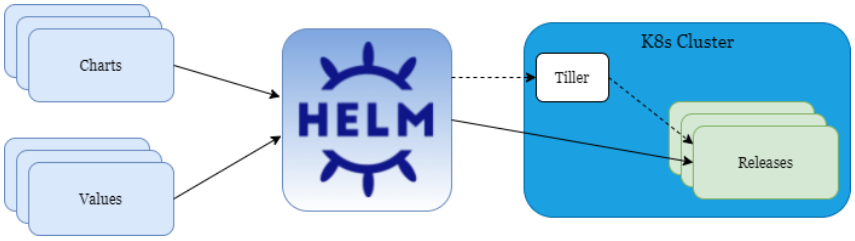
2.3.1.3 Helm

Helm is an application package manager that runs on Kubernetes [HELM, 2021]. Helm is maintained by the community and giants like Google and Microsoft. Writing and maintaining Kubernetes YAML manifests for all the necessary Kubernetes objects can be time-consuming, so creating templates is useful. A Helm-Chart is a collection of files organized in a specific directory structure, during the installation of each Helm-chart it is possible to customize parameters created through variables contained in the YAML manifests [BUCHANAN; RANGAMA; BELLAVANCE, 2020]. Helm has an official repository called Artifact Hub [ARTIFACTHUB, 2021], that has numerous charts that are made available by the community. They can be used to install various applications on Kubernetes. Helm helps to define, install, and upgrade even the most complex Kubernetes application. Fig 6 shows how Helm creates a Kubernetes application. First, Helm reads a Helm-Chart (application package) and replaces the Helm-Chart variables with user-defined values. This definition can occur through the command line or by passing value files that describe the variables. After that, Helm creates a resource, that is, an instance of a Helm-Chart that runs on a K8s cluster. The white block inside the K8s cluster represents the Tiller, it is the Helm server-side templating engine.

2.3.2 NFV Overview

Network Function Virtualization (NFV) is defined as a technique where decoupling of network functions from their hosting hardware platform occurs [HAN et al., 2015]. This technique then enables effective network management that is appropriate for all network scenarios. In 2012, the European Telecommunications Standards Institute (ETSI) developed NFV specifications to provide information to ensure implementation, interoperability and industry security [ETSI,

Figure 6 – Helm usage.



Source: Author

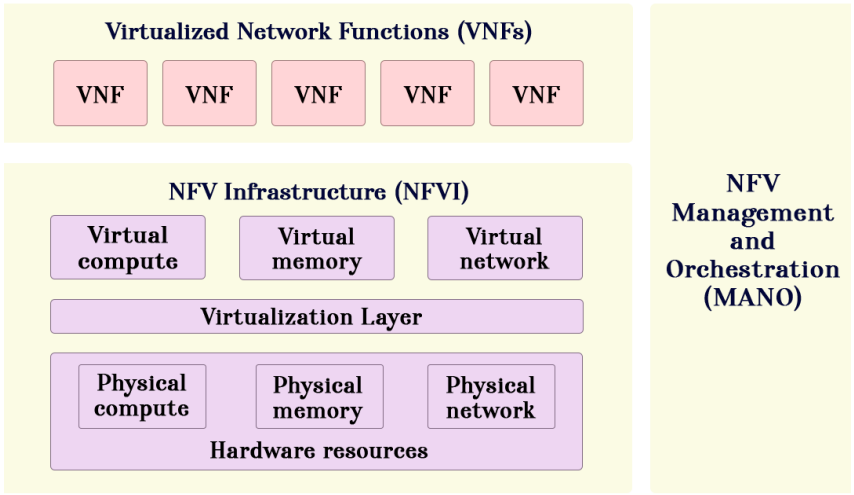
2014]. Also in 2012, ETSI created the Industry Specification Group (ISG) for the NFV, this groups focus into research on virtualized resource management and orchestration. The ETSI ISG NFV contains seven of the world's leading telecom network operators, and has become the home of NFV [REHMAN; AGUIAR; BARRACA, 2019].

From an architectural point of view, ETSI ISG specifications describe and specify virtualization requirements, NFV architecture framework, functional components and their interfaces, as well as the protocols and the APIs for these interfaces. The standart architecture has a major components: one or more Virtualized Network Function (VNF), NFV Infrastructure (NFVI) and NFV Management and Orchestration (MANO). The VNF is responsible for handle specific network functions like firewalls or load balancing. The NFVI component describes the hardware and software components on which virtual networks are built. MANO is an architectural framework that coordinates network resources for cloud-based applications and the lifecycle management of VNFs and network services [TEIXEIRA, 2018]. An overview of the elements of the NFV architecture is shown in Fig 7. NFVI is illustrated in purple blocks, it contains the infrastructure (virtual and physical) needed to run VNFs. Such as compute, memory and network. Also, there is a virtualization layer to communicate the hardware resources and the virtual resources. In this figure red blocks illustrated the VNFs. The MANO component is shown on the right side of figure.

2.3.3 Cloud-Native Network Virtualization

NF are physical devices that process packets supporting a network and/or application services [HAN et al., 2015]. Routers, switches and firewalls are examples of physical devices. VNF is an NF designed to run in a virtual machine, typically deployed on a common hardware infrastructure. Conforming to [PAVAN, 2020], VNFs on VMs require far more powerful servers than standard hardware networking functions. A solution to the performance problem is provided through use of containers, which bring the advantage of a lighter virtualization paradigm and don't have all the overhead introduced by VMs. Cloud-Native Network Functions (CNF) is a container-based approach to building network solutions, it uses containers for network function virtualization instead of VMs. This approach is progressively becoming a solution adopted by telecom operators to provide network services. CNF deployment architecture can support

Figure 7 – NFV architecture illustration.



Source: Author

massive scale with low operational overhead compared to VNF. These bring many benefits, such as: no need to run on specialized hardware anymore, agility, scalability, fault-tolerance and resilience [HUANG; CHOU, 2020]. This work adopts CNFs, the next chapter will cover the details of how the network virtualization was accomplished.

2.4 Used Softwares/Platforms

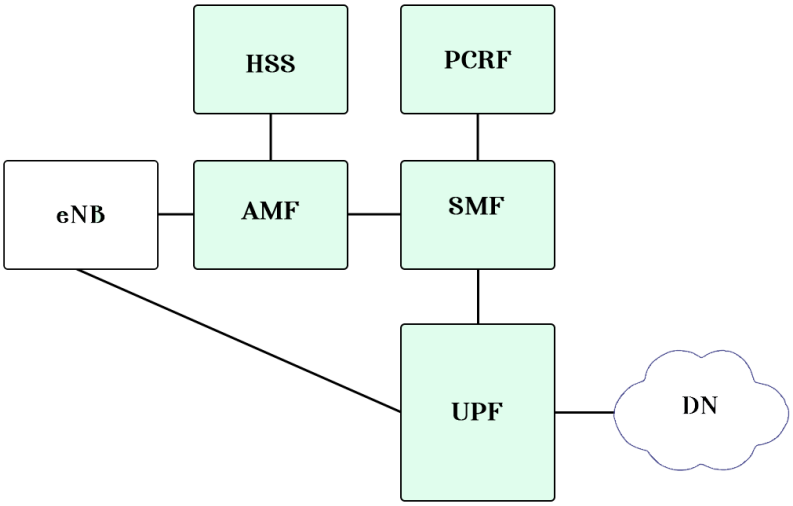
2.4.1 Free5GC

Free5GC is an Open-Source project for 5GC [FREE5G, 2021], it was initially based on NextEPC [NEXTEPC, 2018]. The objective of this project is to implement the 5GC defined in 3GPP Release 15 [3GPP, 2019] and beyond. Free5GC is an initiative of the National Chiao Tung University (Taiwan) and started in 2019. It is actively wielded Open-Source option with support for multiple NFs. The first version (launched on January 2019) is named Stage 1, the objective was to migrate 4G EPC to 5GC SBA. To do this, Free5GC migrated the 4G EPC entities, including MME, S-GW and P-GW to the 5GC functions of AMF, SMF and UPF. Free5GC uses 4G protocols to communicate with the 4G UE and eNodeB. Thus, the authentication protocol is still based on 4G. It is built and tested with Go Language (Golang) [FREE5G, 2019b].

The network functions implemented by Free5GC Stage 1 are as follows: AMF, SMF, HSS, PCRF and UPF. In addition to these functions, there is also a Web Interface App that allows register user devices in the Core using parameters such as Public Land Mobile Network (PLMN), International Mobile Subscriber Identity (IMSI), Operator Code (OP), Subscriber Authentication Key, Mobile Country Codes (MCC), Mobile Network Code (MNC) and others. And it is also possible to view connection information via User Interface (UI). The NFs and Webapp are

connected to a MongoDB database. Free5GC uses the HTTP2 protocol to implement a REST architecture and has Transport Layer Security (TLS) v1.2 encryption to ensure the privacy of data exchanged between NFs [MAILER et al., 2020]. Fig 8 shows on color greeen the 4G functions supported by Free5GC. It also shows how these functions communicate with each other and external blocks (e.g. eNodeB and DN).

Figure 8 – Free5GC architecture.



Source: Author

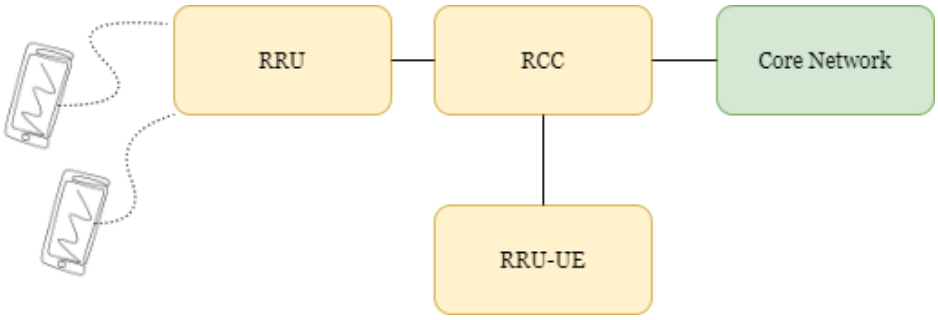
2.4.2 Open Air Interface

OpenAirInterface (OAI) is a platform created by EURECOM to implement the 3GPP stack: RAN (eNB, gNB, UE 4G and UE 5G) as well as the Core Network (EPC and 5GC) [BONATI et al., 2020]. OAI platform is engaged in developing access solutions for networks, radios and core networks [EURECOM, 2020]. OAI is an umbrella project, meaning that it has several "subprojects". On this work it is used the OAI-RAN subproject to provide a software-based implementations of eNodeB and 4G UEs. The OAI official repository provides codes for eNB physical layer implementation, eNB monolith implementation and UE emulation [PATANE et al., 2019]. OAI enables C-RAN and monolith architectures for RAN emulation. As this work is focused on CNF virtualization, the most interesting architecture to work with is the C-RAN, due to its scalability and flexibility.

According to [HÅLAND, 2019], in January 2019 the OAI-RAN Project launched the version v1.0.0, this version included a new simulation setup that uses a Network Functional API (nFAPI) simulator for UE and eNodeB. It allows creation of multiple UEs in the same host. The aim of nFAPI mode is to create an open interface between LTE layer 1 and layer 2 [OAI, 2018]. This research uses OAI-RAN in nFAPI mode, it will refer to RCC as the part that performs access to the Media Access Control (MAC) and RRU as the part that performs the Physical

Layer (PHY). When using only the UE emulator, i.e. no real UE, it is not necessary to perform PHY layer, this work will refer to RRU-UE as the part responsible for UE simulation. Fig 9 demonstrate an environment of RAN builded using the OAI-RAN subproject where the C-RAN architecture is used. The C-RAN environment creates a virtualization of a base station using a two-tiers base station. These tiers are defined as: baseband processing unit and radio frequency. They are, respectively, specified as RCC and RRU. Also, this figure illustrate the RRU-UE usage.

Figure 9 – Example of OAI RAN using C-RAN architecture.



Source: Author

2.4.3 FlexRAN

The non-profit initiative Mosaic5G launched FlexRAN as a controller to RAN based on the OAI RAN 4G/5G open-source platform, it is the first Open-source implementation of a flexible and programmable platform for SD-RAN [BONATI et al., 2020]. It is made up of two main components: FlexRAN Control Plane and FlexRAN Agent API. The FlexRAN Control Plane has a hierarchical design, a Master Controller is connected to a number of FlexRAN Agents (one Agent for each eNodeB). The FlexRAN Agent API provides control and data plane separation. It acts as the southbound API with FlexRAN Control Plane on one side and eNodeB data plane on the other side. FlexRAN protocol facilitates the communication between Master Controller and Agents. FlexRAN is designed with flexibility, programmability, and simplicity of implementation. It offers a level of flexibility to dynamically perform mobile operations within RANs [FOUKAS et al., 2016].

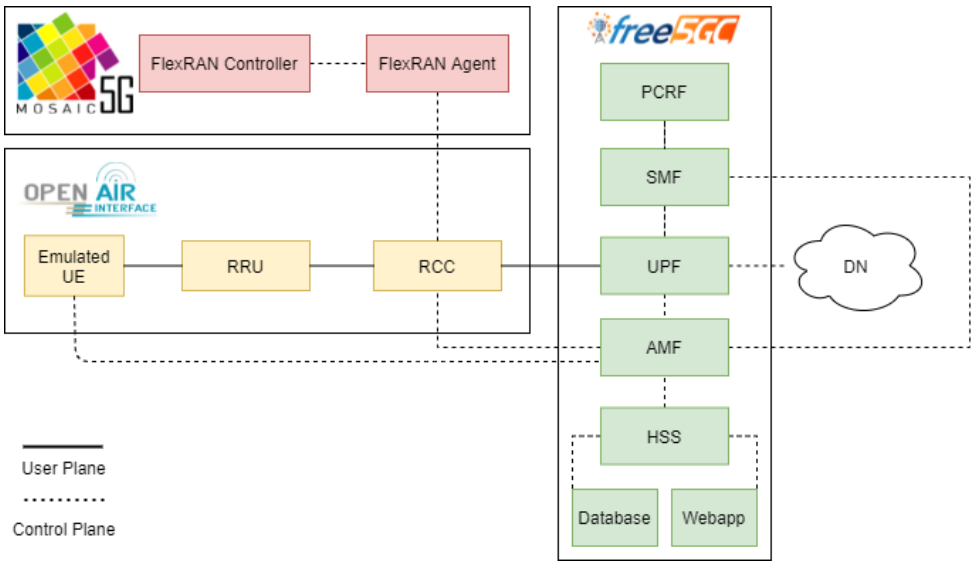
3 PROPOSED WORK

The purpose of this work is implement a CNF virtualized, functional and automated multi-scenario 4G/5G network. For the implementation of this network enviroment, previously, it is necessary to virtualizate and integrate the network functions. This integration consists on create correct communication between the Free5GC, Open Air Interface and FlexRAN softwares. This chapter is intended to present the used softwares, describe network enviroment architecture, explain network conteinerization process using Docker, explain the automatization of K8s cluster creation using Helm tool, and explain the network montage automation utilizing a code written in Python. Through that is expected that the reader can understand this work contribuition to CNF virtualization on a 4G/5G network.

3.1 Architecture

Fig 10 shows a diagram representing the Mobile Network created. This network is capable of serving real UEs and emulated UEs. This is characterized as a 4G/5G network because a 5GC is associated with a 4G RAN. This 4G RAN has C-RAN architecture and a SD-RAN associated with it. On Fig 10 the network functions are grouped according to the software that enables them. The green blocks represent Core Network functions created by Free5GC, the red modules refer FlexRAN components, and the yellow blocks represent network functions created by Open Air Interface. Black lines represent User Plane, while dotted lines refer to Control Plane.

Figure 10 – 4G/5G Mobile Network Diagram.



Source: Author

3.2 Containerization

The first step used for CNF development is the creation of Docker images for all network functions. The methodology used for development of CNFs consists of studying the official documentation of the used softwares and adapting them to container environment. When running a container, it uses an isolated custom file system provided by a Docker image. A Docker image can contain other settings for the container, such as environment variables, run commands, and other metadata. Each Docker image should contain everything needed to run the refer network function — all dependencies, settings, programs, binaries, etc.

3.2.1 Core Network

Most of Free5GC modules have same base environment, these modules are AMF, SMF, PCRF, UPF, HSS and Webapp. So to facilitate the process of building each environment, a parent Docker image is created for base the container environments. The parent image of the Free5GC containers imports the official Go Docker image to use as a base, this Go image already has all the tools and packages to run a Go application. At the top of the Go image, the necessary packages and official Free5GC files are downloaded, as indicated in [FREE5G, 2019a]. Then the binaries contained in the official Free5GC repository are installed and compiled. The Docker images of the modules AMF, SMF, UPF, PCRF, HSS and Webapp use "free5gc-base" as Docker parent image. MongoDB database is based on Ubuntu 18.04 Official Docker image.

For the creation of the individual Docker images of each module, at the top of the Docker parent image the related configuration files and codes are passed. For 5G network functions including the HSS service, the main configuration files are "free5gc.conf" and "[module-name].conf", these must be filled according to the scenario IP adresses. For automate the functions initiation, to each Free5GC module Docker image is passed a Bash script file named "setup-lasse.sh". For 5G network functions (and HSS), it is responsible for fill "free5gc.conf" and "[module-name].conf" properly. For MongoDB database Image, it is responsible for initializing the Free5GC database and registering specific UEs. UPF and MongoDB Docker Images need additional configuration files for correct operation. According to [FREE5G, 2019a], UPF must have two network interfaces, a bridge for Free5GC modules communication and a TUN device for user data tunneling. The Bash script named "setup.sh" is passed to UPF container to create theses intefaces, it allows the creation of these interface by adding routes and creating interface tunnel. MongoDB has the additional JSON file "UE.json", it stores information about the UEs that should be automatically registered when the database image is created. This is useful to avoid having to add UEs manually when the network is started from scratch. Table 2 shows and details all files used to create Free5GC module Docker Images.

Table 2 – Used files for Core Network Docker Images creation.

	File	Description
AMF	Dockerfile	Commands, in order, needed to build the Docker image
	Free5gc.conf	Free5GC AMF configuration file configurated with IP_AMF, IP_MONGO and IP_SMF variables
	AMF.conf	freeDiameter deamon configurated file for AMF configurated with IP_AMF and IP_HSS variables
	setup-lasse.sh	Bash script that replaces the variables with the real IP addresses and starts the function
HSS	Dockerfile	Commands, in order, needed to build the Docker image
	Free5gc.conf	Free5GC HSS configuration file configurated with IP_MONGO variable
	hss.conf	freeDiameter deamon configurated file for HSS configurated with IP_AMF and IP_HSS variables
	setup-lasse.sh	Bash script that replaces the variables with the real IP addresses and starts the function
SMF	Dockerfile	Commands, in order, needed to build the Docker image
	Free5gc.conf	Free5GC SMF configuration file configurated with IP_SMF, IP_UPF and IP_MONGO variables
	SMF.conf	freeDiameter deamon configurated file for SMF configurated with IP_SMF and IP_PCRF variables
	setup-lasse.sh	Bash script that replaces the variables with the real IP addresses and starts the function
PCRF	Dockerfile	Commands, in order, needed to build the Docker image
	Free5gc.conf	Free5GC PCRF configuration file configurated with IP_MONGO variable
	PCRF.conf	Free5GC configuration file configurated with IP_MONGO variable
	setup-lasse.sh	Bash script that replaces the variables with the real IP addresses and starts the function
UPF	Dockerfile	Commands, in order, needed to build the Docker image
	Free5gc.conf	Free5GC UPF configuration file configurated with IP_UPF and IP_MONGO variables
	setup.sh	Bash script that create a tun interface that enables access the IPv6 network via a tunnel
	setup-lasse.sh	Bash script that replaces the variables with the real IP addresses and starts the function
Webapp	Dockerfile	Commands, in order, needed to build the Docker image
	setup-lasse.sh	Bash script that start the User Interface
MongoDB	Dockerfile	Commands, in order, needed to build the Docker image
	UE.json	Text file that contains the configurated UEs
	setup-lasse.sh	Bash script that insert the configurated UEs in the database

Source: Author

3.2.2 RAN

Following [EURECOM, 2018], two Docker images are built to virtualize OAI RAN on nFAPI mode, one image is for RCC and the other one is for RRU-UE. The RAN containers have "base-nfapi" as Docker parent image. This Docker parent image has a first layer the Ubuntu 18.04 Official Docker image. Over the first layer the recommended dependencies packages are installed and enviroment configurations are made (such as kernel and CPU speed modifications). On RCC and RRU-UE individual Dockerfiles, a pre-configured file and an init script are passed. The pre-configured file is based on templates provided by OAI. The init script is called "ran.sh" and it is responsible for starting the network function that the image refers to. The files used to create the RAN related Docker Images are listed and explained in Table 3.

Table 3 – Used files for RAN Docker Images creation.

	File	Description
RCC	Dockerfile	Commands, in order, needed to build the Docker image
	RCC.band7.tm1.nfapi.conf	OAI RCC configuration file
	ran.sh	Bash script that configures MCC, MNC, RRU_IP and AMF_IP, and start RCC function
RRU-UE	Dockerfile	Commands, in order, needed to build the Docker image
	UE_eurecom_test_sfr.conf	OAI RRU and UE configuration file
	ran.sh	Bash script that start RRU function and UE emulated

Source: Author

3.2.3 SD-RAN Controller

For the SD-RAN controller an isolated FlexRAN Docker Image is created, the set up and run tutorial is provided on [EURECOM, 2021]. The only file used was a Dockerfile containing the indicated settings. For FlexRAN Docker image the Ubuntu 18.04 Official Docker image is used as parent Docker Image. On top of this first layer, all necessary dependencies are installed and the official FlexRAN repository is downloaded. Then the binaries of the official repository are installed and compiled. Table 4 described the FlexRAN related Dockerfile.

Table 4 – Used file for SD-RAN Controller Docker Image creation.

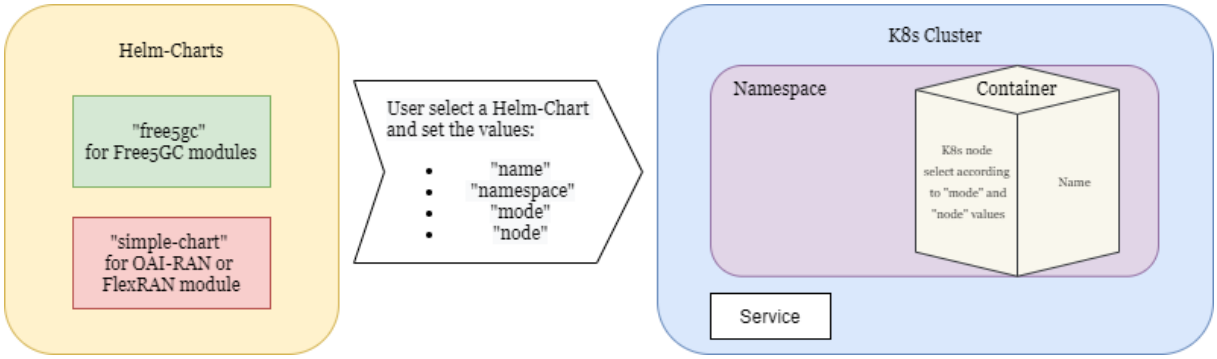
	File	Description
RCC	Dockerfile	Commands, in order, needed to build the Docker image

Source: Author

3.3 Orchestration

The next important step is to create K8s cluster orchestration. Kubernetes manages pods rather than directly managing containers, so each container must be into a pod. From Kubernetes hands-on enviroment, nine manifest-formed deployment configuration files and one manifest-formed service configuration file are required for Core Network creation. It is necessary a manifest-formed deployment configuration file for RCC, RRU and FlexRAN. FlexRAN also needs a manifest-formed service configuration file. With the growth of the network, it would be necessary to modify several files. The Free5GC module containers have the same basis for K8s deployment file. Whereas RAN modules and FlexRAN have same base for K8s deployment file. Templating is a good idea to avoid the problem of creating multiple files, with this solution a manifest base file can be reused and adapted as needed. Helm is a very powerful resource for this solution. Fig 11 illustrates the Helm usage for the creation of deployments and services.

Figure 11 – Helm usage.



Source: Author

A Helm-Chart called "free5gc" was created to create Free5GC modules on K8s. It contains a collection of files describing a set of Kubernetes features needed by Free5GC. When Helm-Chart "free5gc" is used, it creates a K8s deployment in a specified Kubernetes namespace. Namespace and deployment names are passed by the user. Deployment name must be the same as the name of the Docker Image the user wants to use in the container. Therefore, the deployment name must be one of the names of the Free5GC Docker images. The user must also define a mode value, this is a value related to how Kubernetes should recognize correct node where the reference deployment should be placed. Two modes are allowed: local or nodename. If local mode is set, the container will be placed in a node K8s which is the environment labeled with a user-defined expression (e.g. antenna, cloud and edge). If nodename mode is used, the container will be allocated to a node named exactly by a user-defined expression (e.g. this-is-my-pc-name). When "free5gc" Helm-chart is used to deploy Free5GC UPF, it automatically enables NET_ADMIN capability in K8s deployment config file. NET_ADMIN capability enables the user control the network routing table inside UPF container. When this Helm-Chart is used to deploy the Free5GC Webapp, in addition to the deployment, it also created a service to enable UI acess.

For OAI and FlexRAN Kubernetes Deployment, it is created Helm-Chart named "simple-chart". This name is because it can be used in many uses cases. The "simple-chart" receive as variables: "name", "namespace", "mode" and "node". The values usage recommendation is the same as explained for "free5gc" Helm-Chart. If the "name" value on "simple-chart" is "flexran", the Helm-chart creates a Service Kubernetes for FlexRAN be exposed to the Kubernetes cluster. Table 5 described the values that the user should inform to "free5gc" and "simple-chart" Helm-Charts.

Table 5 – Helm-chart required values.

Value	**Description**
Name	Deployment Kubernetes name (refer to the function that should be virtualizated).
Namespace	Minicluster where the deployment should be associated with.
Mode	How the node that will host the container must be identified. There are two modes: "local" and "nodename".
Node	This field must be filled in according to the mode selection. If the mode is "local it must be a Kubernetes node label associated with the requested K8s node. If the mode is "nodename it must be exactly the requested K8s node name.

Source: Author

3.4 Automatization

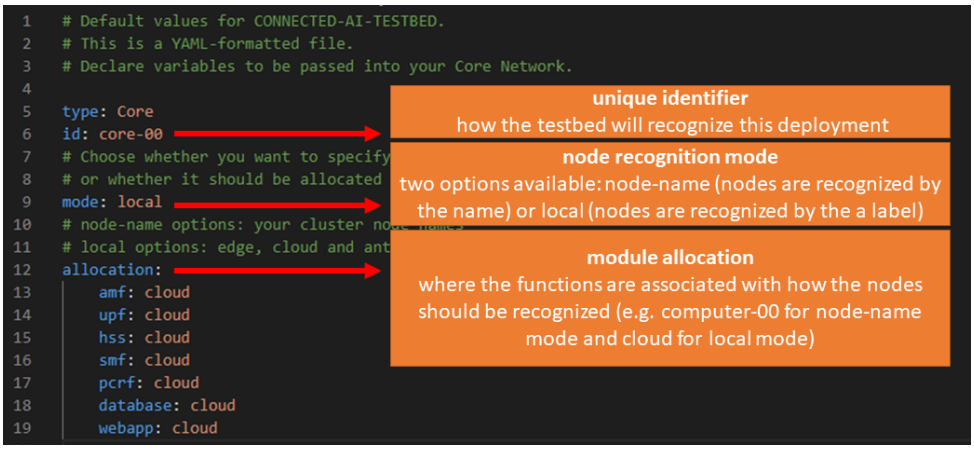
Based on what has been described so far, it is already possible to create a Kubernetes cluster with containers that emulate the 4G/5G network functions. However, a lot of manual work is required to set up the network. It is already knowed that Helm automates development in Kubernetes, but the task of creating and configuring all containers via the command line still remains. There are many programming languages that allow to perform repetitive tasks

efficiently, such as Python, Ruby, Java and others. Thus, to make the network automated, a program in Python language was created to perform the tasks of configuration and installation of the mobile network. This language choice was due to the fact that Python has been used on a large scale in recent years [SRINATH, 2017]. It has several advantages, the most outstanding ones are: the amount of libraries available for use, and it is easy to learn/use.

YAML [YAML, 2011] is serialization language often utilized to create configuration files. For example, YAML is used for Kubernetes and Helm tools. Therefore, YAML format was chosen to store data (instructions) related to network configuration. Individual YAML configuration files were created for Core, RAN and FlexRAN. Thus, it is possible to separate configuration files into three categories: Core, RAN or FlexRAN. Each category file has parameters that will be interpreted by the program, it will configure mini-clusters according to these parameters. Mini-clusters are namespaces within the K8s cluster. Various combinations between mini-clusters can be created, so the program allows multiple scenarios to be created.

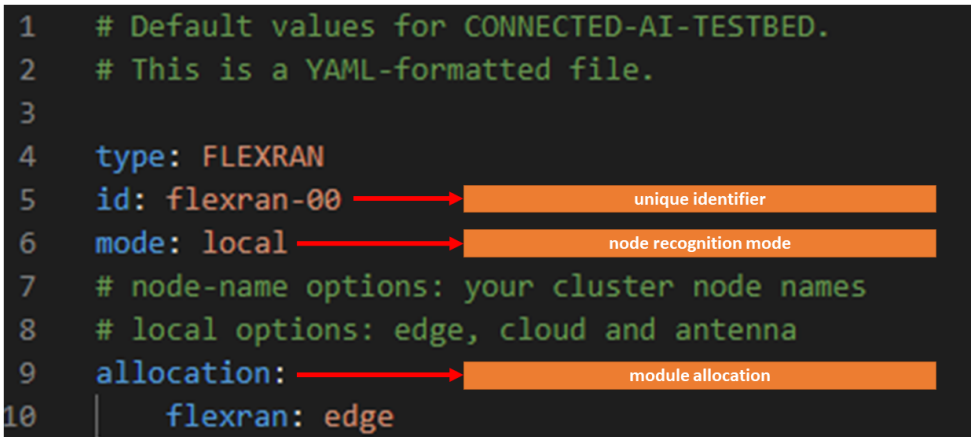
The config files categories have as parameters: "type", "id", "mode" and "allocation". The "type" parameter is intended to identify the category of the YAML config filetype. The "id" is a unique identifier that will be used to refer to the minicluster, as in Kubernetes each Namespace must have a unique name. The "mode" parameter refers to which type of configuration will be used for the assignment between the nodes of the Kubernetes cluster, this parameter can be "mode: local" or "mode: node name". For the "allocation" parameter, the node labels or nodenames are associated with each container. For example, RCC should be in the node labeled "edge", then it is seted "RCC: edge". The node specification depends on how the "mode" parameter was previously configured, if it is "mode: local" it must be described according to a label that characterizes only that node, on the other hand if it is "mode: nodename" the user must write exactly nodename. Examples of each YAML file from these mini cluster categories are seen in Figures 12, 13 and 14.

Figure 12 – Core YAML file.



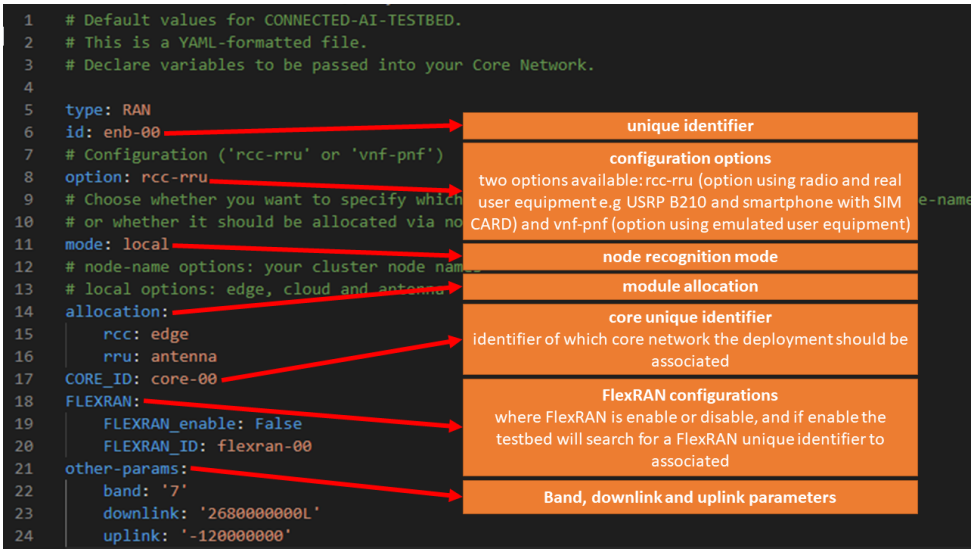
Source: Author

Figure 13 – FlexRAN YAML file.



Source: Author

Figure 14 – RAN YAML file.



Source: Author

The RAN config file also contain the parameters: "Option", "Core ID", "FLEXRAN" and "Other params". The "option" parameter could be "option: RCC-RRU" or "option: vnf-pnf". The first one refers to a real RAN, where the RCC and RRU containers will be created, the node that holds the RRU must be connected to a 4G radio equipment and a real User Equipment should be attached to the network. The radio equipment recommended by the OAI is a USRP B210. On "option: vnf-pnf" create a RAN and a emulated UE. The parameter "Core ID" is associated with the uniqUE identifier of a Core type mini-cluster, this Core will be the one that the RAN will be attached to. On script is set that is mandatory for the RAN creation to have an pre-existing Core mini-cluster to the RAN be connect to. The "FLEXRAN" parameter is formed by two sub parameters "flexran_enable" and "flexran_ID", the first subparameter can be "true" or "false", it defines if a RAN controller can access the RAN mini-cluster. The second subparameter must be filled only if "flexran_enable: true", it refers to the unique identifier of the

FlexRAN mini-cluster that should be linked to the RAN. The "other-parameters" is related to the RAN radio settings offered by the OAI, which are "eNB_ID", "MCC", "MNC", "N_RB_DL", "tx_gain", "rx_gain", "band", "downlink" and "uplink". Table 6 explains each sub parameter related to "other-parameters".

Table 6 – RAN configuration file "Other parameters".

	Description
eNB_ID	20-bit eNB identifier. By default: "0xe00".
MCC	Mobile Country Code. By default: "208".
MNC	Mobile Network Code. By default: "93".
N_RB_DL	Bandwidth in terms of number of available PRBS: 25 (5MHz), 50 (10MHz), and 100 (20MHz). By default: "25".
tx_gain	Transmit gain (dB). By default: "90".
rx_gain	Receive gain (dB). By default: "125"
band	The LTE EUTRA frequency band. By default: "7"
downlink	Downlink frequency band. By default: "2680000000L"
uplink	Uplink frequency offset. By default: "-120000000"

Source: Open Air Interface (OAI)

Once the configuration files have all been created, the Python program development comes next. First program stage perform the job of create and configure mini-clusters according to the passed YAML files. For this, the Python package Pyyaml [PYYAML, 2019] was utilized, it is a YAML parser and emitter for Python.

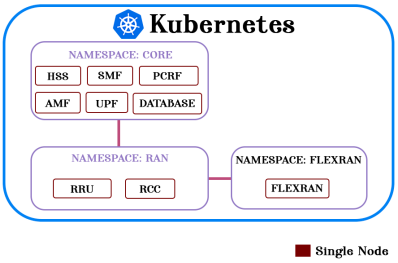
The use of flags was added to the program, these serve to indicate which action must be performed using the indicated file. The creation of flags was done through the Argparse library [FOUNDATION, 2021]. For example, the command "run.py -c core.yaml -r ran.yaml -f flexran.yaml" will run the program and the flags "-c", "-r" and "-f" indicate the functions that should be executed and the configuration file that each function will receive as input. This makes configuring the network more intuitive for the user. It is possible to create only network components. For example, using the flag and input "-c core.yaml" the program will only create the Core Network.

The second program stage is creating the proper functions for each type of YAML file. For the initiation of a Core several tasks must be performed, such as the creation of deployments of Free5GC modules, IP adresses replacement on Free5GC pre-configured files, database and UI generation, tunneling interfaces creation, and 5G Core network functions startup. To accomplish tasks inside containers the command "kubectl exec" is used to access the container's command line. The Core function creates the all Free5GC containers, stores the containers IP addresses and Pod names, runs "setup.sh" on the UPF and Webapp containers (to create the tunneling network interface and UI, respectively), and runs "script-lasse.sh" for start the modules containers. The RAN creation function receives the IP addresses and pod names of the OAI modules. This function accesses the command line of the RAN containers configures according to the parameters passed, thus replacing IP addresses and other values where necessary. For example, filling the RCC configuration file with the IP address of the mini-cluster's AMF indicated in the "CORE_ID" parameter. This function also initializes the RAN modules created. The FlexRAN

YAML file interpreter function creates the FlexRAN container/namespace on indicated node and initializes it.

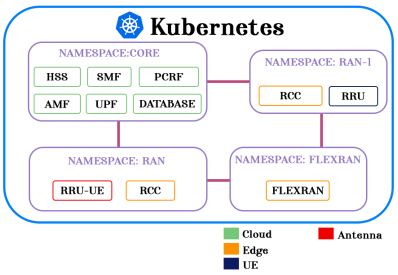
Custom configuration files and an automation script make possible to create several possible scenarios for a network according to the environment provided. Fig 15a and Fig 15b shows examples of diferent scenarios that can be created using the proposed method.

(a) Single node cluster example.



Source: Author

(b) Four node cluster example.



Source: Author

Figure 15 – Scenarios examples.

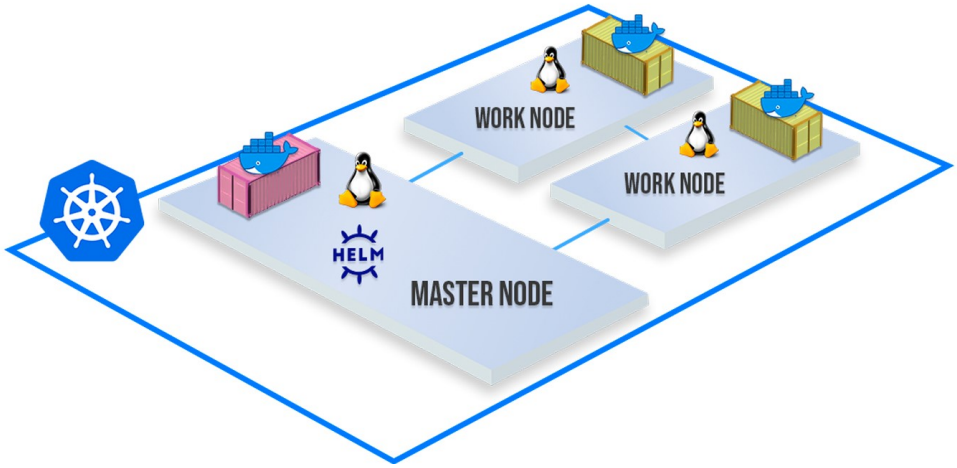
4 RESULTS

4.1 Simulation/Emulation Environment

Three machines composes the development environment, each machine has Intel Core i5- 7500 CPU@3.4 GHz, uses Ubuntu 18.04 as OS, and has low-latency kernel. Two of the machines have 8 GB of RAM DDR4 memory, and the other machine has 16 GB of RAM DDR4 memory. For a better understanding the machines with 8 GB RAM machines will be mentioned as "Edge" and "Antenna" each, and "Cloud" will refer to the 16 GB RAM machine. In this way, the environment will be following the label model proposed for using the Python code that performs the CNF-placement.

All nodes get the Docker and Kubernetes installed on. After that, all the Docker images developed in the previous chapter are built in the nodes. Also, the CNI plugin Calico [CALICO, 2021] is installed on nodes to offer container networking to the Kubernetes cluster. The "Cloud" machine is choosen as Kubernetes cluster master node, therefore the "Antenna" and "Edge" machines are the work nodes. The Helm software is installed only on the master node, because this node is the one responsible for the command line tool (Kubectl) used to run commands against the Kubernetes cluster. Also, in this node the Python 3.9 compiler (interpreter) was installed, so it be able to run Python scripts. Fig. 16 illustrate the development environment.

Figure 16 – Experiment development environment.



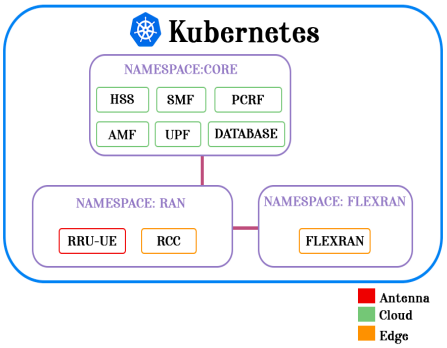
Source: Author

4.2 Emulated Scenarios

For the analysis of the proposed network, this research addresses three main scenarios to compare and validate them. The first scenario is presented in Fig. 17a. This is a generic scenario commonly used in the C-RAN architecture. It focuses on a low cost deployment as the Core

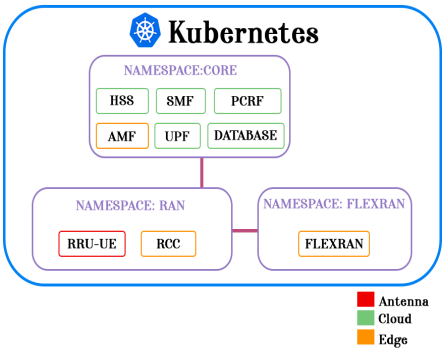
Network is allocated only to the "Cloud" node. As only one machine is used, this scenario requires less computer to provide Core Network functions. The RCC function is allocated to the "Edge" node (machine that acts as an end-user portal for communication with other compute cluster nodes), while the RRU-UE pod is allocated to the "Antenna" node (the computer that represents the end user). The objective of second and third scenarios is to physically move computing from the data centers to the edge of the network, thereby reducing pressure on the data centers. This data is often overloaded from devices and therefore may not always provide acceptable response times. Fig. 17b represents the second scenario, it is possible to notice that the function AMF moves to the node "Edge". This function was chosen to be moved because it is responsible for managing the connection and tasks of mobility, so moving it shortens the path between the eNB and the AMF. Fig. 17c shows the third scenario, in this the UPF and AMF have been relocated to the "Edge" node, because this new arrangement makes the route RCC to AMF shortened. It also shortens the path between the RRU-UE and UPF, where data flows from the UE through the eNB to UPF.

(a) C-RAN Default Scenario.



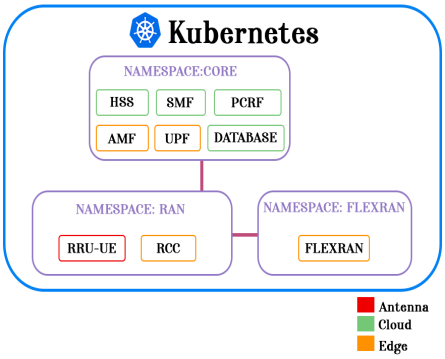
Source: Author

(b) AMF function on "Edge" node.



Source: Author

(c) AMF and UPF functions on "Edge" node.



Source: Author

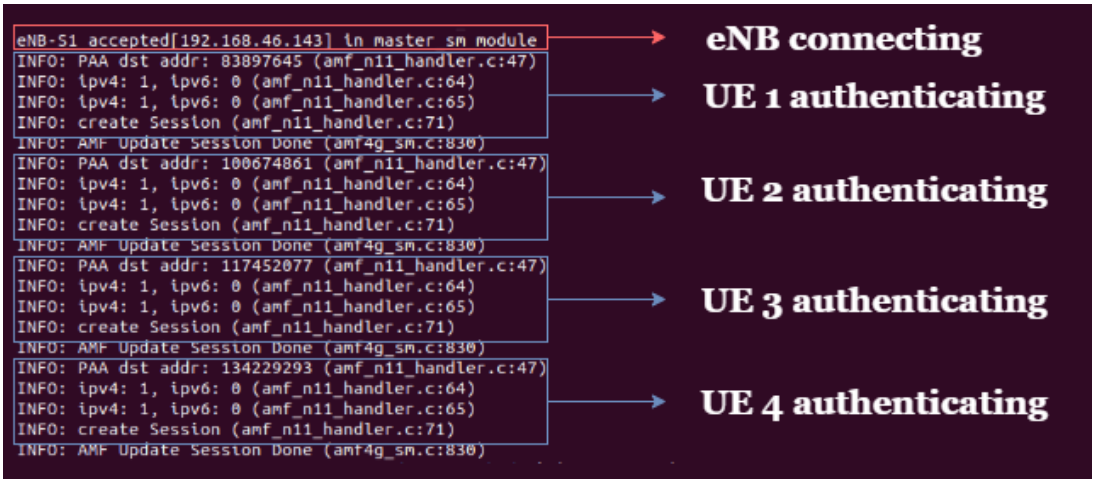
Figure 17 – Three main scenarios.

4.3 Scenarios Validation

This work purpose also involves building a functional 4G/5G mobile network, therefore, it is necessary to visualize the service quality that the network can offer. The AMF contain all registers and sessions that happen on the network since AMF function was started. From the UPF the traffic usage report is obtained. In general for the Core Network it is important to scout two specific functions to validate if the network is really working, these functions are AMF and UPF. The network builded, in any scenario choosed, can be monitored by Kubernetes pod logs. These logs are obtained using the K8s command "kubectl logs" in the specific pod required.

The AMF register all attempts to access the Core Network, such as interactions between the network and UEs or eNB. Fig 18 show some of these interactions, it is possible to notice the connection with the eNB being established and the authentication of authorized users on the network.

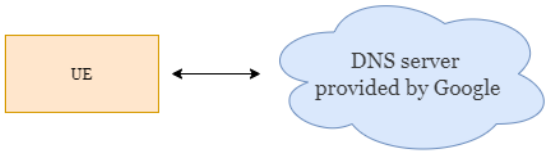
Figure 18 – AMF Kubernetes pod log.



Source: Author

Once the eNB, emulated UEs and Core Network are up and running, it is possible to add traffic to each UE by sending packets to a specific host, for example by accessing the DNS server provided by Google. Fig 19a illustrate the communication between the UE and Google Server. Fig 19b shows a UE created by the OAI software inside the RRU-UE pod that can access the Internet. In the first block of the figure is shown the output of the command "ping", this command shows that the interface can access the Google Server. In 5G it is also important to test the N3 interface, which is responsible for transmitting user data from the RAN to the UPF function, enabling the creation of low and high latency services. In Fig 19c is shown the output of the command "iftop" on the TUN interface inside the UPF pod, showing the operation on the N3 interface in UPF when a UE accesses the given DNS server for Google. Analysis of the AMF pod log, N3 interface, and UE Internet access were performed in all three main scenarios. These analyzes allowed validation of the correct functioning of all three scenarios.

(a) Illustration of the communication between the UE and Google Server.



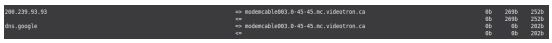
Source: Author

(b) Interface created for the UE connecting to the internet.

```
root@pnf-6bc56bdcb-m9jqt:/openairinterface5g# ping -I oaitun_ue2 8.8.8.8
PING 8.8.8.8 (8.8.8.8) from 45.45.0.9 oaitun_ue2: 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=51 time=55.8 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=51 time=54.0 ms
```

Source: Author

(c) Communication between the UE and the Google Server via TUN interface.



Source: Author

Figure 19 – UE accessing Google server.

4.4 Scenarios Performance Evaluation

Performance assessment can be classified into hardware measurement and service measurement. The hardware performance is possible using CPU and memory parameters, while service measurement uses the amount of delay on the UE Internet connection as a parameter.

4.4.1 Hardware Measurement

4.4.1.1 Scenario 1

This scenario is a default C-RAN use case, the processing power is divided between the machines, its resources can be utilized more effectively with less consumption and as well as allowing access to larger amounts of resources when needed. Fig 20a shows that the "Cloud" node is using more resources than the others, this is expected since the Cloud Network Functions demand a lot of CPU because it process interactions such as authentication, security, session management and aggregation of traffic from end devices. Its noted that the RCC pod requires more CPU and memory than the RCC-UE pod, but both pods are below 10%.

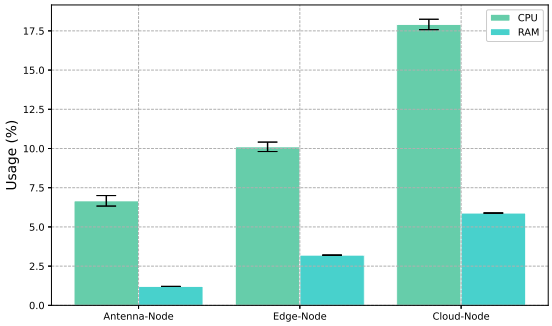
4.4.1.2 Scenario 2

Fig 20b present the increase in CPU usage at the "Edge" node when the AMF function is reallocated. Its possible to confirm that the new CNF placement only affected the nodes that have CNF moved. The "Cloud" node the resourcement usage still above 10%, but is less than the showed in the Fig. 20a. Also, it is possible to see that the RAM usage bars aren't proportional to CPU usage bars anymore.

4.4.1.3 Scenario 3

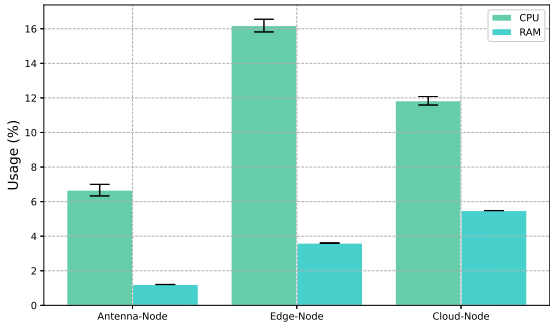
This scenario is where the CNFs are located closest to the users. Fig 20c demonstrates that moving computing away from the data centers towards the edge of the network requires more resources at the node representing the edge. The edge computing network increases the cost because if the number of eNBs increases (which usually happens in telecommunications networks) more high-perfomance equipment will be needed. The CPU usage of the "Edge" node is almost four times the usage of the other nodes. Also, it is observed that the RAM usage of the "Edge" node and the "Cloud" node are very similar, both around 5%, in the previous scenarios the RAM usage of the "Cloud" node was above the "Edge" node RAM , this demonstrates that memory usage is better distributed in this scenario.

(a) Scenario 1.



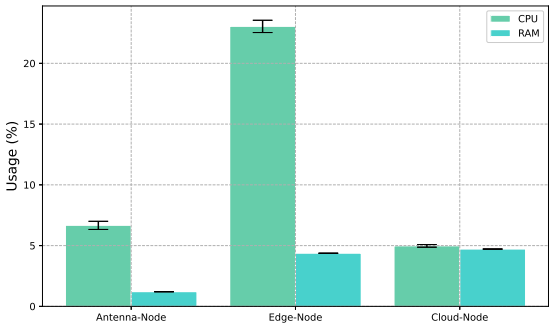
Source: Author

(b) Scenario 2.



Source: Author

(c) Scenario 3.



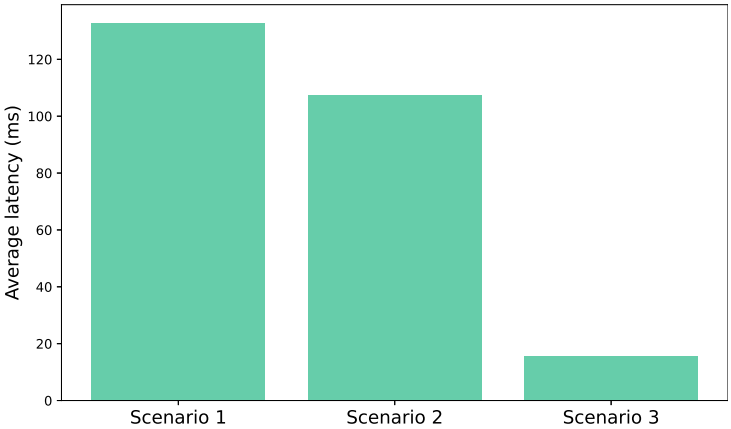
Source: Author

Figure 20 – Resource usage.

4.4.2 Service Measurement

Latency refers to the delay users experience as data makes a round trip through the network. If the latency of a network is high, this means that network will feel less responsive and be slower to react to user requests. So, to the service measurement (or user experience) analysis, it is used the avarage latency parameter (measured in milliseconds). It represents the typical delay that an user experiences when connecting across an operator's networks. The comparison of average latency of the scenarios is shown in Fig 21.

Figure 21 – Average latency of the scenarios.



Source: Author

The average latency comparison showed that if the AMF and UPF functions are closer to the UE (same node) the user experience increases, that is, the latency decreases. However, the default C-RAN architecture (scenario 1) is the weakest network scenario, i.e. it is not suitable for use where response time can have an impact on the result. Scenario 2 has a latency more similar to scenario 1 than scenario 3, it is considered a good scenario. Scenario 3 is the most efficient with regard to average latency, therefore, suitable for situations that need a direct service advantage for highly immersive and interactive application environments, such as multiplayer gaming and multimedia-rich communications.

5 CONCLUSION

Given the results obtained, it is concluded that the objectives were successfully achieved, culminating in a simple, flexible, scalable and effective solution for the implementation of a 4G/5G mobile network testbed (with SD-RAN controller) virtualized through containers. This work detailed the mobile network technologies and Open Source platforms. It also detail about main containers related softwares, and about the process of the network functions containers creation and orchestration .

The containerization process was carried out by adapting the software used to the Docker platform. Orchestrating network functions containers using the Kubernetes tool has demonstrated that it is possible to integrate network functions and create various scenarios in a practical way. Thus, proving the flexibility and scalability of a container-based network architecture. The creation of a program in Python language allowed for the ease of use of the testbed, the user started having less work in creating the network.

The validation of the proposed testbed occurred through the assembly of three different scenarios using the created Docker images and Python program. All scenarios were considerable acceptable as they all had latency less than 150 ms.

Finally, it is concluded that the use of CNF allows the user to build their mobile network according to their needs and also allows portability and scalability for the network. When compared to other testbeds, the presented testbed has the differences: being fully virtualized, ability to create different scenarios in an automated way, containerized RAN controller feature, and possibility to use emulated and/or real UE in a scenario. The work provides an environment for research in 4G/5G with emulation capability, low complexity, low computational requirements. Thus, it helps other researchers to conduct experiments related to 4G/5G mobile networks without having to invest much time in the network creation practical steps.

BIBLIOGRAPHY

3GPP. *LTE*. 2008. [Online]. Available: <https://www.3gpp.org/technologies/keywords-acronyms/98-lte> Accessed on 2021-08-04.

3GPP. *Overview of 3GPP Release 8*. 2008. [Online]. Available: <https://www.3gpp.org/specifications/releases/72-release-8> Accessed on 2021-08-04.

3GPP. *The Evolved Packet Core*. 2008. [Online]. Available: <https://www.3gpp.org/technologies/keywords-acronyms/100-the-evolved-packet-core> Accessed on 2021-08-04.

3GPP. *System architecture for the 5G system - TS 23.501*. 2017.

3GPP. *System architecture milestone of 5G Phase 1 is achieved*. 2017. [Online]. Available: <https://www.3gpp.org/news-events/1930-sys_architecture> Accessed on 2021-08-04.

3GPP. *Release 15*. 2019.

ALAM, I. et al. A Survey of Network Virtualization Techniques for Internet of Things Using SDN and NFV. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 53, n. 2, abr. 2020. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3379444>.

ARTIFACTHUB. *The Artifact Hub*. 2021. [Online]. Available: <https://artifacthub.io/> Accessed on 2021-08-04.

BERNSTEIN, D. Containers and Cloud: From LXC to Docker to Kubernetes. *IEEE Cloud Computing*, v. 1, n. 3, p. 81–84, 2014.

BERTENYI, B. et al. Ng radio access network (ng-ran). *Journal of ICT Standardization*, River Publishers, v. 6, n. 1, p. 59–76, 2018.

BONATI, L. et al. Open, Programmable, and Virtualized 5G Networks: State-of-the-Art and the Road Ahead. *Computer Networks*, v. 182, p. 107516, 2020. ISSN 1389-1286. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1389128620311786>.

BOULOS, K. *BBU-RRH Association Optimization in Cloud-Radio Access Networks*. Tese (Doutorado) — Université Paris-Saclay (ComUE), 2019.

BUCHANAN, S.; RANGAMA, J.; BELLAVANCE, N. Helm Charts for Azure Kubernetes Service. In: *Introducing Azure Kubernetes Service*. [S.l.]: Springer, 2020. p. 151–189.

CALICO. *Calico*. 2021. [Online]. Available: <https://docs.projectcalico.org/> Accessed on 2021-08-04.

CASTRO, L. et al. PA5Ge - Implementação de uma Rede Celular Privada com Funcionalidades 5G no Campus da UFPA. In: . [S.l.: s.n.], 2020.

CISCO. *Software-Defined Networking*. 2020. [Online]. Available: <https://www.cisco.com/c/en/us/solutions/software-defined-networking/overview.html> Accessed on 2021-08-04.

DOCKER. *Docker*. 2021. [Online]. Available: <https://www.docker.com/> Accessed on 2021-08-04.

DOCKER. *Dockerfile image*. 2021. [Online]. Available: <https://docs.docker.com/engine/reference/commandline/image/> Accessed on 2021-08-04.

DOCKER. *Dockerfile reference*. 2021. [Online]. Available: <https://docs.docker.com/engine/reference/builder/> Accessed on 2021-08-04.

ERICSSON. *5G radio access networks - Mobile radio access networks and 5G evolution*. 2020. [Online]. Available: <https://www.ericsson.com/495922/assets/local/policy-makers-and-regulators/5-key-facts-about-5g-radio-access-networks.pdf> Accessed on 2021-08-04.

ERICSSON. *Leveraging the potential of 5G millimeter wave*. 2021. [Online]. Available: <https://www.ericsson.com/490025/assets/local/reports-papers/further-insights/doc/leveraging-the-potential-of-5g-millimeter-wave.pdf/> Accessed on 2021-08-04.

ETSI, N. F. V. Network Functions Virtualisation (NFV). *Management and Orchestration*, v. 1, p. V1, 2014.

EURECOM. *nFAPI How to*. 2018. [Online]. Available: <https://gitlab.eurecom.fr/oai/openairinterface5g/-/wikis/nFAPI-howto> Accessed on 2021-07-27.

EURECOM. *Overview - Open Air Interface*. 2020. [Online]. Available: <https://openairinterface.org/overview/> Accessed on 2021-08-04.

EURECOM. *Set up and run the FlexRAN Realtime Controller*. 2021. [Online]. Available: <https://gitlab.eurecom.fr/mosaic5g/mosaic5g/-/wikis/tutorials/flexran#set-up-and-run-the-flexran-realtime-controller> Accessed on 2021-07-27.

FOUKAS, X. et al. FlexRAN: A flexible and programmable platform for software-defined radio access networks. In: *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*. [S.l.: s.n.], 2016. p. 427–441.

FOUNDATION, P. S. *argparse*. 2021. [Online]. Available: <https://docs.python.org/3/library/argparse.html> Accessed on 2021-08-04.

FREE5G. *free5GC Installation on KVM*. 2019. [Online]. Available: <https://www.free5gc.org/installations/stage-1-all-in-one/> Accessed on 2021-07-27.

FREE5G. *Roadmap of Free5GC project*. 2019. [Online]. Available: <https://www.free5gc.org/> Accessed on 2021-08-04.

FREE5G. *Free5GC: open-source 5GC*. 2021. [Online]. Available: <https://www.free5gc.org/roadmap/> Accessed on 2021-08-04.

GAWAS, A. U. An overview on evolution of mobile wireless communication networks: 1G-6G. *International Journal on Recent and Innovation Trends in Computing and Communication*, v. 3, n. 5, p. 3130–3133, 2015.

GUERREIRO, T. S. et al. 4G/5G Network Data Collection and Processing using Elastic Stack. *X Conferência Nacional em Comunicações, Redes e Segurança da Informação – ENCOM 2020*, p. 1–2, 2020.

GöRANSSON, P.; BLACK, C.; CULVER, T. Chapter 4 - how sdn works. In: GöRANSSON, P.; BLACK, C.; CULVER, T. (Ed.). *Software Defined Networks (Second Edition)*. Second edition. Boston: Morgan Kaufmann, 2017. p. 61–88. ISBN 978-0-12-804555-8. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780128045558000041>.

HÅLAND, M. *Evaluation of low-cost software-defined LTE/5G frameworks: cloudification*. Dissertação (Mestrado) — University of Stavanger, Norway, 2019.

HAN, B. et al. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, IEEE, v. 53, n. 2, p. 90–97, 2015.

HELM. *The package manager for Kubernetes*. 2021. [Online]. Available: <https://helm.sh/> Accessed on 2021-08-04.

HOLMA, H.; TOSKALA, A. *LTE for UMTS: OFDMA and SC-FDMA based radio access*. [S.l.]: John Wiley & Sons, 2009.

HSU, S. *What is C-RAN? The Evolution From D-RAN to C-RAN*. 2020. [Online]. Available: <https://www.ufispace.com/company/blog/what-is-cran-the-evolution-from-dran-to-cran> Accessed on 2021-08-04.

HUANG, Y.-X.; CHOU, J. Evaluations of Network Performance Enhancement on Cloud-Native Network Function. In: *Proceedings of the 2021 on Systems and Network Telemetry and Analytics*. New York, NY, USA: Association for Computing Machinery, 2020. (SNTA '21), p. 3–8. ISBN 9781450383868. Disponível em: <https://doi.org/10.1145/3452411.3464442>.

IBM. *What is Kubernetes?* 2021. [Online]. Available: <https://www.ibm.com/cloud/learn/kubernetes> Accessed on 2021-08-04.

KHARBULI, P.; SULTANA, A. A comparative study on the generations of mobile wireless telephony: 1g -5g. 09 2018.

KUBERNETES. *Kubernetes*. 2021. [Online]. Available: <https://kubernetes.io/pt-br/> Accessed on 2021-08-04.

KUBERNETES. *Kubernetes Components*. 2021. [Online]. Available: <https://kubernetes.io/docs/concepts/overview/components/> Accessed on 2021-08-04.

KUBERNETES. *Service | Kubernetes*. 2021. [Online]. Available: <https://kubernetes.io/docs/concepts/services-networking/service/> Accessed on 2021-08-04.

LU, Y. et al. Software Defined Radio Access Network in 5G Mobile Network. In: *2015 10th International Conference on Communications and Networking in China (ChinaCom)*. [S.l.: s.n.], 2015. p. 132–136.

MAHESHWARI, S. et al. Comparative Study of Virtual Machines and Containers for DevOps Developers. 08 2018.

MAILER, C. et al. Plataforma de CORE 5G em nuvem para disponibilização de funções de rede como serviço. Blumenau, SC, 2020.

MAYER, G. RESTful APIs for the 5G service based architecture. *Journal of ICT Standardization*, p. 101–116, 2018.

MERKEL, D. et al. Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, v. 2014, n. 239, p. 2, 2014.

NAHUM, C. et al. Testbed for 5G Connected Artificial Intelligence on Virtualized Networks. *IEEE Access*, v. 8, p. 223202–223213, 2020.

NASCIMENTO, R. D. F. *TÉCNICAS DE VIRTUALIZAÇÃO EM CLOUD COMPUTING*. 2013. Monografia (Bacharel em Análise e Desenvolvimento de Sistemas), FEMA (Fundação Educacional do Município de Assis), Assis, Brazil.

NEDU, V.; PUTTARAJU, A. R. M. A survey on Docker and its significance in cloud. In: . [S.l.: s.n.], 2016.

NEXTEPC. *What's NextEPC*. 2018. [Online]. Available: <https://github.com/nextepc/nextepc> Accessed on 2021-08-04.

NOVOA, L. et al. RAN Slicing using OpenAirInterface and FlexRAN in a Virtualized Scenario. *XXXVIII SIMPÓSIO BRASILEIRO DE TELECOMUNICAÇÕES E PROCESSAMENTO DE SINAIS – SBrT 2020,* , p. 1–2, 2020.

OAI. *open-nFAPI*. 2018. [Online]. Available: <https://gitlab.flux.utah.edu/powder-mirror/openairinterface5g/-/tree/2021.w09/nfapi/open-nFAPI> Accessed on 2021-08-04.

OLIVEIRA, L. A.; ALENCAR, M. S.; LOPES, W. T. A. Evolução da arquitetura de redes móveis rumo ao 5g. *Revista de Tecnologia da Informação e Comunicação*, v. 8, n. 2, p. 43–50, 2018.

PATANE, G. M. M. et al. Flexible SDN/NFV-based SON testbed for 5G mobile networks. In: *2019 IEEE/ACM 23rd International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*. [S.l.: s.n.], 2019. p. 1–8.

PAVAN, F. *Offering Cloud Native Network Services to Residential Users*. Tese (Doutorado) — Politecnico di Torino, 2020.

PYYAML. *PyYAML Documentation*. 2019. [Online]. Available: <https://github.com/yaml/pyyaml> Accessed on 2021-07-27.

REHMAN, A. U.; AGUIAR, R. L.; BARRACA, J. P. Network functions virtualization: The long road to commercial deployments. *IEEE Access*, v. 7, p. 60439–60464, 2019.

RISCHKE, J.; SALAH, H. Chapter 6 - software-defined networks. In: FITZEK, F. H.; GRANELLI, F.; SEELING, P. (Ed.). *Computing in Communication Networks*. Academic Press, 2020. p. 107–118. ISBN 978-0-12-820488-7. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780128204887000189>.

RODRIGUEZ, M.; BUYYA, R. Container orchestration with cost-efficient autoscaling in cloud computing environments. In: *Handbook of research on multimedia cyber security*. [S.l.]: IGI Global, 2020. p. 190–213.

SALIH, A. A. et al. Evolution of mobile wireless communication to 5g revolution. *Technology Reports of Kansai University*, v. 62, n. 5, p. 2139–2151, 2020.

SRINATH, K. Python - the fastest growing programming language. *International Research Journal of Engineering and Technology (IRJET)*, v. 4, n. 12, p. 354–357, 2017.

TAVARES, V. et al. Emulação de Rede LTE Usando OpenAirInterface e Análise de Tráfego Via Protocolo de Rede. *XXXVII SIMPÓSIO BRASILEIRO DE TELECOMUNICAÇÕES E PROCESSAMENTO DE SINAIS – SBrT2019*, p. 1–1, 2019.

TAVARES, V. et al. Machine Learning Workload in a 4G/5G Testbed using Kubeflow. *X Conferência Nacional em Comunicações, Redes e Segurança da Informação – ENCOM 2020*, p. 1–2, 2020.

TEIXEIRA, E. V. Virtualização (SDN e NFV) com ênfase em desempenho de rede. Universidade Tecnológica Federal do Paraná, 2018.

VMWARE. *What is a hypervisor?* 2019. [Online]. Available: <https://www.vmware.com/topics/glossary/content/hypervisor> Accessed on 2021-07-28.

VORA, L. J. Evolution of mobile generation technology: 1g to 5g and review of upcoming wireless technology 5g. *International journal of modern trends in engineering and research*, v. 2, n. 10, p. 281–290, 2015.

YAML. *YAML*. 2011. [Online]. Available: <https://yaml.org/> Accessed on 2021-08-04.

ZHILING, e. a. Y. White paper of next generation fronthaul interface v1.0. In: CHINA MOBILE RESEARCH INSTITUTE. *Tech. Rep.* [S.l.], 2015.