UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Design and Implementation of Virtual Network Functions for 5G Base Stations

Diploma Thesis

Zafeiris Vasileios

Supervisor: Korakis Athanasios

Volos 2022

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Design and Implementation of Virtual Network Functions for 5G Base Stations

Diploma Thesis

Zafeiris Vasileios

Supervisor: Korakis Athanasios

Volos 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Σχεδιασμός και υλοποίηση εικονικών συναρτήσεων δικτύου σταθμών βάσης 5ης Γενιάς

Διπλωματική Εργασία

Ζαφείρης Βασίλειος

Επιβλέπων: Κοράκης Αθανάσιος

Βόλος 2022

Approved by the Examination Committee:

Supervisor	**Korakis Athanasios** Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly
Member	**Katsaros Dimitrios** Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly
Member	**Argyriou Antonios** Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Date of approval: Tuesday 25th January, 2022

Acknowledgements

I would like to thank all the people that helped me in the whole journey of my studies. First of all im gratefull to my family and friends for always supporting me to go forward. I would like to thank Prof. Athanasios Korakis for the confidence he showed in me and his belief in my abilities, offering me the opportunity to work on this subject. Also I feel blessed to be working with all the NITLab team and especially with Nikos Makris, who helped me and guided me.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Abstract

The developments in cellular mobile technologies and the rise of multi-tenant 5G networks accelerated the transition to a virtualized and finally cloud-native-based context. These new approaches mark the evolution of the technology of computer networks, called Network Function Virtualization (NFV). Network Functions Virtualization infrastructure aims to provide a common interface for technology developers and service operators for the instantiation of virtual network functions over generic equipment. It can help to save costs since it allows the use of virtualized base stations instead of physical deployment options, supporting their deployment in different areas more easily.

To achieve a high level of virtualization and to home multiple components in high processing-powered hosts, Kubernetes enters the picture. This framework tool manages the former, in a well-defined and scalable manner, permitting a fellow developer/researcher to cover from a wide variety of state-of-the-art use cases, met in real-life scenarios (e.g. V2X/Autonomous Driving, VR/AR, Smart Grid applications, etc.).

Considering the above, this thesis covers many details that relate to the setup and deployment of a Kubernetes-based virtualized 5G Network, while to apply these concepts in a practical & usable manner, we utilize the NITOS Testbed.

Contents

List of Figures

Abbreviations

5G	5th Generation Network
LTE	Long Term Evolution
VR	Virtual Reality
AR	Augmented Reality
V2X	Vehicle to everything
ETSI	European Telecommunications Standards Institute
NFV	Network Function Virtualization
VNF	Virtual Network Function
UE	User equipment
RAN	Radio Access Network
OAI	Open-Air Interface
3GPP	3rd Generation Partnership Project
GSM	Global System for Mobile communication
UMTS	Universal Mobile Telecommunications System
HSPA	High Speed Packet Access
OFDMA	Orthogonal frequency-division multiple access
FDE	Frequency balancing system
WLAN	Wireless Local Area Network
MIMO	Multiple In Multiple Out
EUTRAN	Evolved Universal Terrestrial Radio Access Network
EPC	Evolved Packet Core
MME	Mobile Managment Entity
SGW	S-Gateway
PGW	P-Gateway
HSS	Home SubscriberServer

PCRF	Policy Control and Charging Rules Function
PDN	Packet Data Network
PCEF	Policy Control Enforcement Function
UPF	User plane Function
DN	Data network
AMF	Access and Mobility Management Function
AUSF	Authentication Server Function
SMF	Session Management Function
NSSF	Network Slice Selection Function
NEF	Network Exposure Function
NRF	Network Repository Function
PCF	Policy Control function
UDM	Unified Data Management
AF	Application Function
SDN	Self Defined Network
eMBB	Enhanced Mobile Broadband
mMTC	Massive Machine Type Communications
uRRLC	UltraReliable Low Latency Communication
5G NR	5G New Radio
MANO	Managment and Orcherstration
NFVO	Network Function Virtualization Orchestrator
VNFM	Network Function Virtualization Manager
VIM	Virtualized Infrastructure Manage
CNF	Cloud Network Function
NIC	Network interface controller
NSA	Universal Serial Bus
USRP	Universal Software Radio Peripheral
API	Application Programming Interface
K8s	Kubernetes
SSH	Universal Software Radio Peripheral
CNI	Container Network Interface

Chapter 1

Introduction

The demand for Network Services market kickstarted the telecommunications revolution that is taking place to this day, as it leads to the constant finding and introduction of innovative ideas. By achieving high system capacity, leveraged peak data rates and low latency, 4th generation network comes to reduce operating costs, with flexible bandwidth operations. The configuration of LTE is one of the most important milestones for wireless communications.

Subsequent increase in user demand for mobile data and high speed services made clear that new ideas had to surface. Some of the use cases that are introduced through these demands, increasingly used in our everyday lives are VR, AR and V2X. Those demands that are transforming to new technologies, led to 5th generation networks. Network scaling in LTE is a time-taking and complex task which increases the operating expenses of providers. 5G networks address this problem with auto-scaling of the infrastructure meaning that network, software and resources can be scaled-up and down according to current needs and situations. At the same time cloud-based functionality which is already accommodated since the previous generation networks has shown that virtualization and containerization will be inevitably be used.

Following ETSI NFV, network functions are now sliced and distributed in a well-defined and scalable manner to multiple users of different use case scenarios and demand. NFV architecture allows hardware resources to be utilized and shared virtually with components called VNFs Virtual Network Functions. In parallel with that, containerized technology, that Docker introduced, can be used to add these VNFs to containers. To tackle with problems that relate to the management and orchestration of multiple Docker containers that home our virtualized NFs, tools like Kubernetes will be deployed on a real-life testbed environment

(NITOS testbed).

In this work, we present a study and realization of a modern virtual 5G topology that relates to typical use cases and concerning the connectivity and maintenance of a user equipment (UE). This scheme is deployed under Kubernetes-based orchestration with pods acting as network components that form a fully-fledged Core & Radio Access Network (RAN), utilizing the srsRAN implementation that has open-source availability.

1.1 Thesis Structure

This thesis is structured in five chapters. The first chapter is this introduction itself. The second chapter sets the technological background and theoretical concepts of Mobile Technologies and Network Virtualization. In the third chapter the architecture of the thesis's approach is presented, explaining the components used and in the fourth are provided details about the implementation and design of the whole system. Lastly, the fifth chapter draws conclusions and mentions future directions.

1.2 Related Work

In the areas of virtualized mobile telecommunications, we may find relevant implementations like Kube5G [1]. The latter was first presented in 2021 with a relevant demo that included setting up the Core and Access networks with the help of OpenAirInterface (OAI) and connecting a single 5G UE to the architecture. Other examples include Nervion [2], a scalable and flexible RAN emulator for mobile core system evaluation that features a novel cloud-native approach.

The examples presented above, show that a virtualized Cloud-RAN approach is desired for current and future 5G-based deployments, and based on these efforts, we will present our own setup and implementation concerning a virtualized and Kubernetes-orchestrated 5G network.

Chapter 2

Background

2.1 LTE Networks

LTE or 3GPP Long Term Evolution is the state-of-the-art technology used for high-speed wireless communication and networking of mobile devices. It relies on pre-existing GSM/EDGE(2G) and UMTS/HSPA(3G/3G+) networks, increasing network capacity and speed using new configuration techniques [3]. Unlike previous networks generations, 4th generation, otherwise known as LTE, does not support the traditional switching telephony service, but it supports Internet-based IP protocol communication, such as IP telephony. The transmission of multiple OFDMA carriers and other frequency balancing (FDE) systems enabled the transmission of higher data rates across multiple paths compared to the radio spectrum propagation technology used in 3G systems which was widely used [4]. Maximum upload data rates are in the range of 75 Mbps and maximum download data rates are in the 300 Mbps range.Moreover, connectivity to WLAN, satellite and older systems(GSM) is supported. Additionally, 4G networks uses techniques such as MIMO, a technology that uses multiple antennas at both the transmitter and the receiver, to improve signal quality and bit rates for mobile phone users. Furthermore, MIMO increases the signal quality and bit rates by employing spatial multiplexing, diversity and beamforming techniques.

2.1.1 LTE Architecture

LTE architrecture and the interfaces between the different parts of the system are illustrated in Figure 2.1 below:

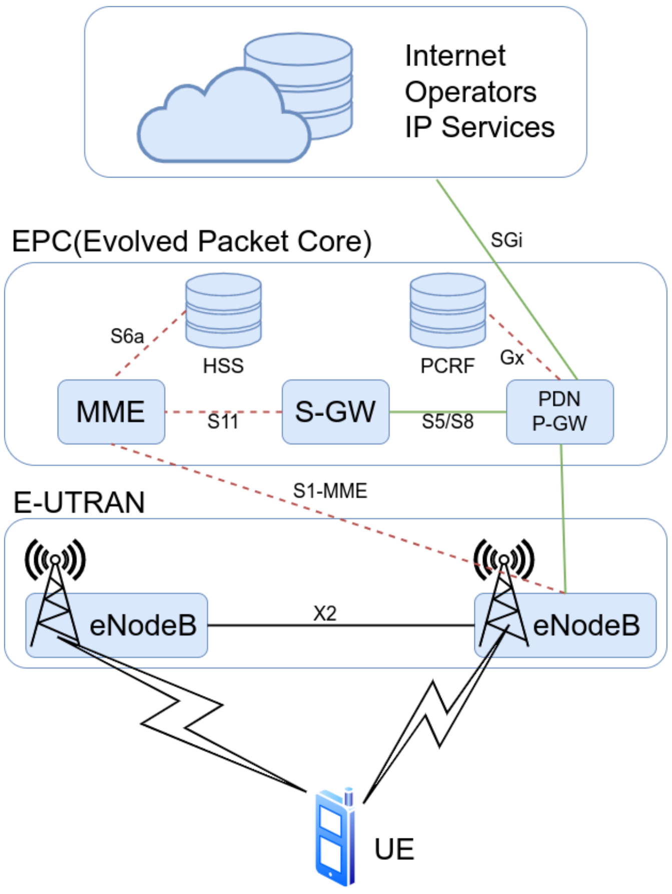

Figure 2.1: LTE Architecture

The high-level network architecture of LTE is comprised of:

- the User Equipment(UE)
- the Evolved UMTS Terrestrial Radio Access Network (E-UTRAN)
- the Evolved Packet Core (EPC)

The internal architecture of the UE for LTE is identical to the one used by UMTS and GSM which is a mobile equipment. The E-UTRAN handles the radio communications between the mobile and the evolved packet core and has just one component, the evolved base station, also known as eNodeB or eNB. Each eNB connects with the EPC by the means of the S1 interface and it can also be connected to nearby base stations by the X2 interface, which is mainly used for signalling and packet forwarding during handover. The LTE EPC architecture consists of MME, SGW, PGW, HSS and PCRF. The mobility management entity (MME) controls the high-level operation of the mobile by means of signalling messages and Home Subscriber Server (HSS). The Packet Data Network (PDN) Gateway (P-GW) communicates with the outside world. The serving gateway (S-GW) acts as a router, and forwards data between the base station and the PDN gateway. The Home Subscriber Server (HSS) component is a central database that contains information about all the network operator's subscribers. The Policy Control and Charging Rules Function (PCRF) is a component responsible for policy control decision-making, as well as for controlling the flow-based charging functionalities in the Policy Control Enforcement Function (PCEF), which resides in the P-GW.The interface between the serving and PDN gateways is known as S5/S8. This has two slightly different implementations, namely S5 if the two devices are in the same network, and S8 if they are in different networks.

2.2 5G Networks

The main advantage of the new networks is that they will have greater bandwidth, giving higher download speeds, eventually up to 10 gigabits per second (Gbit/s). The latter is noted to happen in combination of ultra-density and ”zero-downtime” availability.

By focusing on virtualization and disaggregation of its critical parts, 5G access networks address scalability and availability restrictions that applies to the now legacy implementation of the LTE standard.As a consequence, various abstracted network utilities can be packaged inside a network function that can be orchestrated efficiently.

2.2.1 5G Architecture

5G architecture and the interfaces between the different parts of the system are illustrated in Figure 2.2 below:

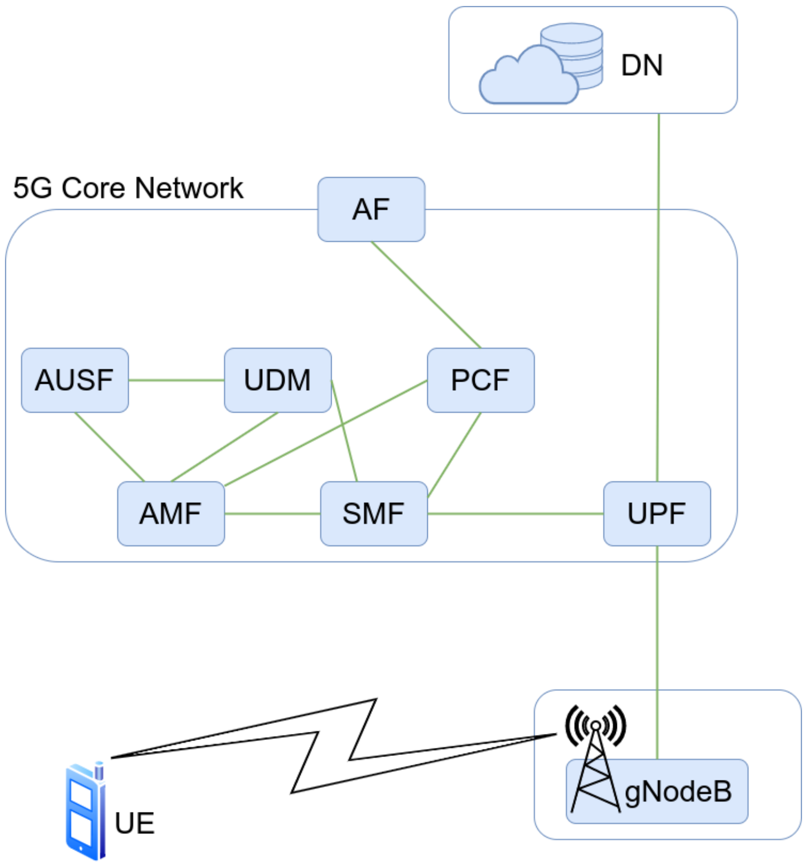

Figure 2.2: 5G Architecture

5G network architecture is much more service oriented than previous generations. Services are provided via a common framework to network functions that are permitted to make use of these services. Modularity, reusability and self-containment of network functions are additional design considerations for a 5G network architecture. The components of the 5G core architecture include:

- User plane Function (UPF)
- Data network (DN) e.g. operator services, Internet access or 3rd party services
- Core Access and Mobility Management Function (AMF)
- Authentication Server Function (AUSF)
- Session Management Function (SMF)
- Network Slice Selection Function (NSSF)
- Network Exposure Function (NEF)

- NF Repository Function (NRF)
- Policy Control function (PCF)
- Unified Data Management (UDM)
- Application Function (AF)

User Equipment (UE) like 5G smartphones or 5G cellular devices connect over the 5G New Radio Access Network to the 5G Core and further to Data Networks (DN), like the Internet. The Access and Mobility Management Function (AMF) acts as a single-entry point for the UE connection. Based on the service requested by the UE, the AMF selects the respective Session Management Function (SMF) for managing the user session.The User Plane Function (UPF) transports the IP data traffic between the User Equipment (UE) and the external networks.The Authentication Server Function (AUSF) allows the AMF to authenticate the UE and access services of the 5G Core.Other functions like the Session Management Function (SMF), the Policy Control Function (PCF), the Application Function (AF) and the Unified Data Management (UDM) function provide the policy control framework, applying policy decisions and accessing subscription information, to govern the network behavior [5].

2.2.2 5G Network Slicing

Network slicing is a type of virtual networking architecture that are moving modern networks toward software-based automation. Software-defined networking (SDN) and network functions virtualization (NFV) are network technologies that assist in network slicing. This allows for the creation of multiple virtual networks atop a shared physical infrastructure. In a virtualized network scenario, physical components are dynamically partitioned according to current needs. As these needs change, so can the devoted resources. Using common resources such as storage and processors, network slicing permits the creation of slices devoted to logical, self-contained, and partitioned network functions. [6]

Network slicing has a plethora of use cases and for that reason 5G characterized as a use case driven technology. Most of these cases fit into one of three service categories:

- Enhanced Mobile Broadband (eMBB)

  This service type, seen as an evolution of currently available LTE services, covers Use Cases with different demands by means of user density, traffic capacity and user mobility but has a common need for seamless radio coverage that is irrelevant to location and timing.

- Massive Machine Type Communications (mMTC)

  This use case category related by the mass deployment of billions of low-cost, low-powered devices. Can support use cases like home automation including sensors and actuators, and machine monitoring systems that require low data rates. Required as a foundation for the digitisation of many industrial applications, like agricaltural or logistics applications through using cellular IoT technologies.

- Ultra-Reliable Low Latency Communication (uRRLC)

  Ultra-reliable low latency communication is when highly reliable connectivity is required to offer extremely low latencies of one millisecond or below.5G NR can enable latencies of 1 millisecond or below to support use cases like self-driving cars, mission-critical applications, industrial automation and many others.

2.3 Network Function Virtualization

State-of-the-art data centers and telecoms enterprises increasingly build upon network functions virtualization (NFV), to scalably offer what is needed over their deployed network system infrastructure. NFV is thus considered a network architecture concept that leverages virtualization techniques to flexibly and scalably share entire classes of network node functions into building blocks that can also be chained together, creating and delivering communication services and utilities. NFV decouples software from hardware by replacing various network functions such as firewalls, load balancers and routers with running virtualized software instances. The NFV architecture defined by ETSI, in 2013. The European Telecommunications Standards Institute (ETSI) is a European standards organisation in the telecoms industry.

2.3.1 NFV Architecture

The high level architecture is composed of the Management and Orchestration (MANO), which is in turn composed of the NFV Orchestrator (NFVO), the VNF Manager (VNFM), and the Virtualized Infrastructure Manager (VIM). NFVO manages the life-cycle of the instantiated networking services. VNFM manages the life-cycle of its instantiated VNFs. VIM allocates the virtual resources required by the NFVO and/or the VNFM on the physical infrastructure. [7]

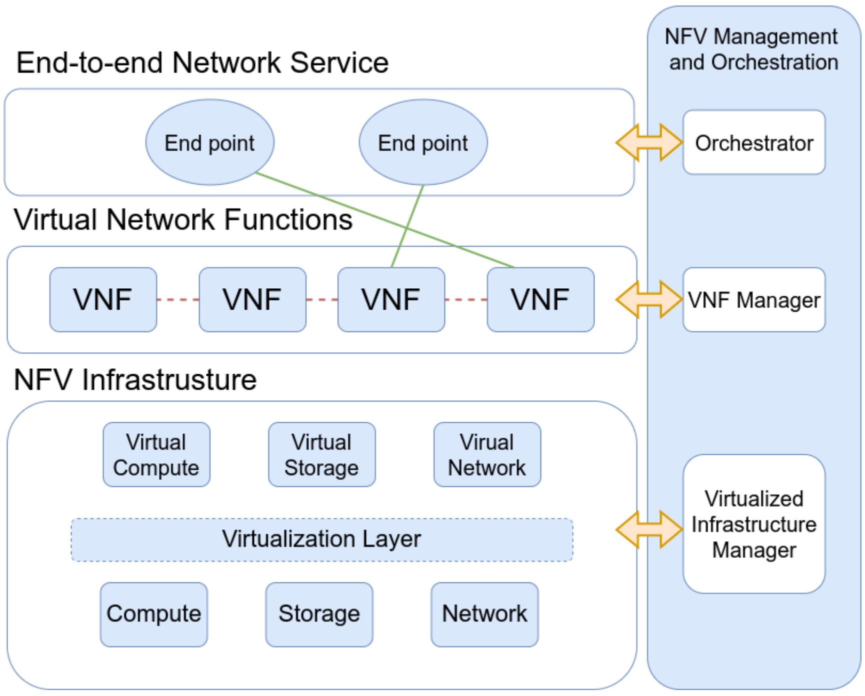

Figure 2.3: 5G High-level Architecture

2.3.2 NFV MANO Standard

NFV MANO is the key framework, that ETSI defined, for the management and orchestration of the resources in a virtualized data center such us compute, networking, storage, and virtual machine resources. NFV MANO has the role to manage the NFVI and orchestrate the allocation of resources needed by the NSs and VNFs, in addition with allowing the flexible on-boarding of network components. The functional blocks of NFV MANO are the NFV Orchestrator (NFVO), the VNF Manager (VNFM) and the Virtualised infrastructure manager (VIM). Each of these blocks has a number of responsibilities and works on specific entities.

2.3.2.1 NFV Infrastructure

The environment providing computing, storage and network capabilities on which Virtual Network Functions are instantiated, includes both physical resources such as servers, network-attached storage and switches, and software, hypervisors, operating systems, and virtual infrastructure managers. NFV Infrastructure includes the virtualization layer that allows partitioning and access to physical resources, allowing an hardware-independent lifecycle for VNFs [8].

2.3.2.2 Virtual Network Functions

Virtual Network Functions (VNF) are entities responsible of the task of handling specific Network Functions. It generally runs on virtual machines on top of the physical network infrastructure. A VNF is meant to perform a certain network function e.g. router, switch, firewall, load-balancer, etc. and a combination of these VNFs may be required to implement the complete network segment that is being virtualized.

2.3.2.3 VNF Manager

The VNF Manager is an NFV-MANO block that controls, manages and monitors the VNFs lifecycle under the direction of the NFVO. Especially, it is responsible for instantiation- and configuration- related procedures, by using a VNF Template that is able to control the state of VNF instances and thus manage the related software. It can also handle procedures like scaling in/out or other procedures related to the administrative termination of VNF instances. As mentioned above, each VNF is defined in a template called Virtualised Network Function Descriptor (VNFD) and is stored in a VNF package repository, called the VNF catalog. The VNFD defines the operational behaviour of a VNF, specifying how it should be deployed by providing a full description of its attributes and requirements.

2.3.2.4 NFV Orchestrator

NFVO is the component that manages the lifecycle of virtualized network services, by communicating with both the VNF Manager and the VIM. Key features observed in NFV Orchestrator are:

- Resource Orchestration is the function that manages resources to services that already provided by NFVI, bypassing any VIM.Some of the features provided by this function are:
  - Validation and authorization of NFVI resource requests from the VNF Managers, to control how the allocation of the requested resources interacts within one NFVI

or across multiple NFVI.

  - NFVI resource management, including the distribution, reservation and allocation of NFVI resources to NS and VNF instances. Also resolves the location of VIMs, providing it to the VNFMs if required.
  - Validation and authorization of any NFVI resource request that may come from a VNF Manager, to control its impact on the current Network Services
  - Verification of the integrity, authenticity and consistency of deployment templates, including the on-boarding of new Network Services and VNF Packages that are available in any NFVI, with support of VIM.

- Network Service Orchestration, meaning the function of the NFVO that uses the services exposed by the VNF Manager function and by the Resource Orchestration function to provide important capabilities such us:
  - Management of the instantiation of VNFs, in coordination with VNF Managers
  - Network Service instantiation and Network Service instance lifecycle management, through operations like updating, querying, scaling and terminating a Network Service. This also includes collecting performance measurement results and recording events
  - Management of the VNF Forwarding Graphs that define the topology of a Network Service instance
  - Validation and authorization of any NFVI resource request that may come from a VNF Manager, to control its impact on the current Network Services
  - Verification of the integrity, authenticity and consistency of deployment templates, including the on-boarding of new Network Services and VNF Packages that are available in any NFVI, with support of VIM.
- Management of the VNF Forwarding Graphs that define the topology of a Network Service instance

2.3.2.5 Virtualized Infrastructure Manager

The MANO block that manages hardware resources and performance (availability, status, power and utilization) is contained in an NFV infrastructure is VIM. In parallel, VIM can define the amount of resources needed for Network Functions both on the Physical and Virualization Layer or across the End-to-end Network Service, but also into multiple NFVI.

The role of a VIM is to configure the compute, hypervisor and infrastructure network domains. Considering the correlation between NFVI and VIM, we note that NFVIs through the Virtualization Layer join the hardware physical resources to Virtual Hardware. Such as Computing, Storage and Network resources are perceived as Virtual Computing, Storage and Network, as per ETSI framework. Therefore Computing (CPU plus Memory) can be converted to pool and be used by different host through a cluster formatted resource. Virtual Storage can be formatted into NAS topology and be assigned or de-assigned to devices when needed. Respectively Virtual Networking can combine NICs via available ports from routers, switches Optical transponders and Wi-Fi to provide capacity when needed. It must be mentioned that these functional blocks are independent from each other and not mandatory to run on a single device. The whole process and management of this function is managed from VIM [9].

In particular, VIM deals with: • Resource management. VIM is in charge of allocating, and releasing resources from the NFVI when requested by the NFVO; keeps track of used resources and their physical allocation, optimizing hardware usage. Manages the pooling of hardware resources, and the list of available virtual and physical resources available. • Networking. Provides the networking infrastructure supporting the VNF Forwarding Graphs in the form of virtual links, networks, and subnets. • Image management. Manages the catalog of software images available to the NFVO, and allows creation, update, and deletion of images. • Reporting. Collects metrics about performance and faults, making information available when requested from the NFVO.

2.3.3 Cloud-native Network Functions

In recent years, the market has adopted a cloud-native approach, which is based on microservices and container orchestration systems, for problems like weaknesses of a network function performance as it trying to foresee and be ready for the future evolution needs. These also fall under the umbrella of NFV but are implemented with containers instead of virtual machines. Containers share the underlying operating system, which means they make more efficient use of resources than VMs.

As CNF is a network function that designed and implemented to run inside a container instead of a virtual machine. They inherit all the principles consisted in a cloud-native scenario, introducing a number of advantages[10]:

- Performance. CNF processing in user space, NIC hardware control, multi-core tuning and kernel bypass all contribute to maximum throughput and resource efficiency. Characterizing CNF performance is underway in the CNF Testbed project.
- Agnostic to host environments.
- Stateless Configuration. CNFs operate in pods. Pods are durable, but if something breaks, or new functions are needed, then you can tear down pods and start a new instance. The dynamic nature of cloud native environments precludes centralized or stateful CNF configuration management.
- Lifecycle parity with application containers. CNFs receive, at no cost, all existing best practices bestowed to application pods. These include: development environments, toolchains, CI/CD, K8s orchestration, 12-factor app, distributed management, logging, and telemetry streaming.

Chapter 3

System Architecture

3.1 NITOS Testbed

NITOS (Network Implementation Testbed using Open Source code) is a heterogeneous testbed located in the campus of University of Thessaly, in Greece. NITOS managed by the Network Implementation Testbed Laboratory (NITlab) of the Electrical and Computer Engineering Department in collaboration with the Center of Research and Technology Hellas (CERTH). The NITOS testbed gives access to researchers to perform experiments and test their implementations in real-time environments.

Figure 3.1: NITOS Logo

3.1.1 Testbed Architecture

The testbed consists of wireless nodes based on open-source software and are partitioned across three different testbed locations (two indoor, one outdoor), running any of the major UNIX based distributions. The testbed provides the necessary extensions to our framework in order to deploy services over virtualized wireless network interfaces, hosted on the generic networking nodes. The NITOS testbed architecture illustrated in Figure3.2 .

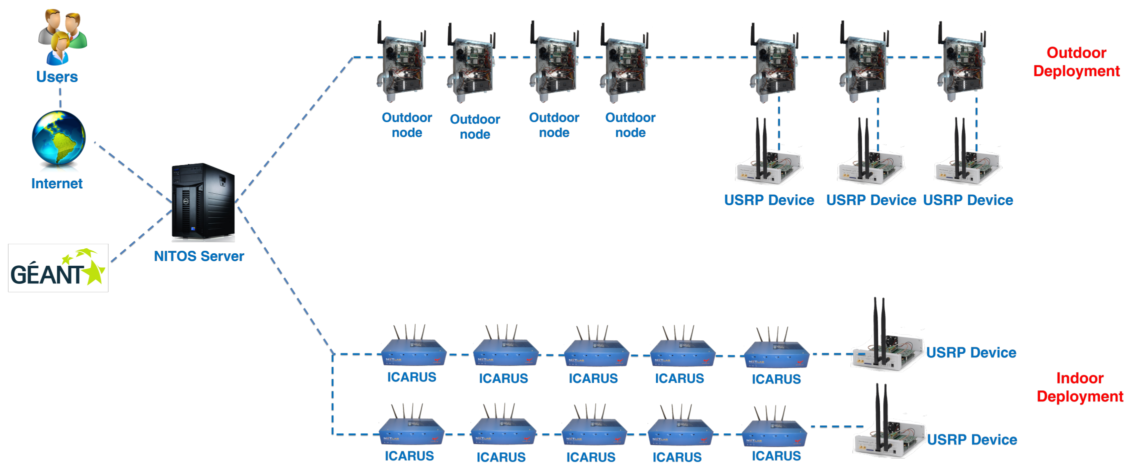

Figure 3.2: NITOS Testbed Architecture

The indoor testbeds comprise of multiple Icarus nodes, that have been developed by the NITlab team. They are able to execute many realistic scenarios through the multiple heterogeneous interfaces they dispose. The basic manufacture characteristics of ICARUS nodes are that they are equipped with 802.11a/b/g and 802.11a/b/g/n wireless interfaces and feature new generation intel 4-core CPU's and new generation solid-state drives.

Figure 3.3: ICARUS Node

3.1.2 Orchestration Infrastructure

The orchestration software that is running in the testbed is acting as the "glue" between the developed software frameworks and the physical infrastructure of the testbed. Network Functions Virtualization Management and Orchestration (NFV-MANO) provides a standardized approach on the management and effortless deployment of (virtual) services. Although NFV-MANO initially focused on the deployment of services over datacenters, the introduction of fully softwarized network architectures even for the wireless part creates fertile ground for the re-conception of the manner through which the underlying hardware is managed. As all of our prior contributions in the testbed are software based, the NFV-MANO architecture can be employed for the efficient orchestration of the frameworks in the testbed. NITOS has adopted the Open Source MANO framework for provisioning virtual services on top of the virtualized wireless equipment. Through extensions to the Virtual Infrastructure Manager service for the testbed, we are able to establish and manage virtualized wireless network interfaces, hosted on the generic networking nodes of the testbed [6]. The extensions are introduced transparently and as an optional feature to the existing operation of the orchestrator, in order to allow the portability of network services and network functions to instances that do not implement our extensions. With our contributions, we manage to deploy virtual functions inter-networked over wireless links, as well as maintain the traditional NFV MANO deployment process [11].

3.2 Architecture Components

3.2.1 Docker

Docker is an open source project providing a systematic way to automate the faster deployment of Linux applications inside containers. Docker containers are created using base application-level images. A Docker image can include just the OS basic configurations, or it can consist of an application stack, ready to deploy.

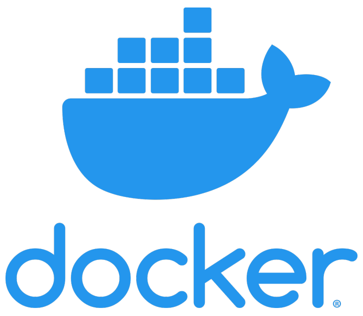

Figure 3.4: Docker Logo

3.2.1.1 Virtual machines and containers

Virtual machines is the predecessor virtualization technology of containerization technologies. Virtual machines are built on top of a hypervisor, which allows several virtual machines to run on one machine. Each virtual machine instance contains a full copy of an operating system and all the dependencies needed to run an application, which take up several gigabytes of storage space. On the contrary containers are built on top of the Host operating system with the possibility to run simultaneously and share the same Operating System kernel. Each container runs as an isolated process in user space, which means that there is not necessarily communication between them. Container sizes vary but are generally in a range that is usually less than 1GB, while being able to boot up almost instantly, owing to their lightweight nature.

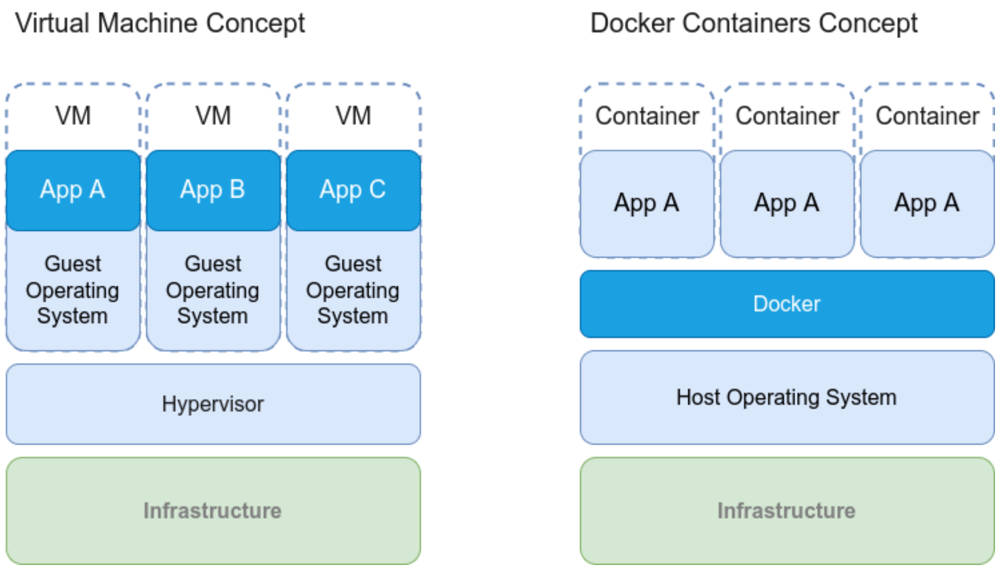

Figure 3.5: Virtual machine and Docker container high-level architecture

3.2.2 srsRAN

srsRAN is an open-source 4G and 5G software radio suite that provides UE and RAN solutions. It can leverage off-the-shelf equipment, RF hardware, for both the UE and eNodeB/gNodeB applications. The srsRAN includes features like srsEPC, srsENB and srsUE.srsRAN uses srsEPC as a lightweight implementation of a complete LTE core network (EPC). However, srsEPC can run as a single binary and still provide the key EPC components such as Home Subscriber Service (HSS), Mobility Management Entity (MME), Service Gateway (S-GW), and Packet Data Network Gateway (P-GW). srsRAN uses srsENB in conjunction with srsEPC to implement functionalities of a complete 4G/5G BS.

srsUE is an application running on Linux-based systems. Its major advantage over other candidate software implementations like OpenAirInterface (OAI), is its ability to be deployed either on 4G LTE or 5G NR NSA modes. Using this software, a client may request, connect and receive service over standardized interfaces and methods and carry out different tasks that may include Downlink (DL) and Uplink (UL) traffic, as made possible by the containerized network components.

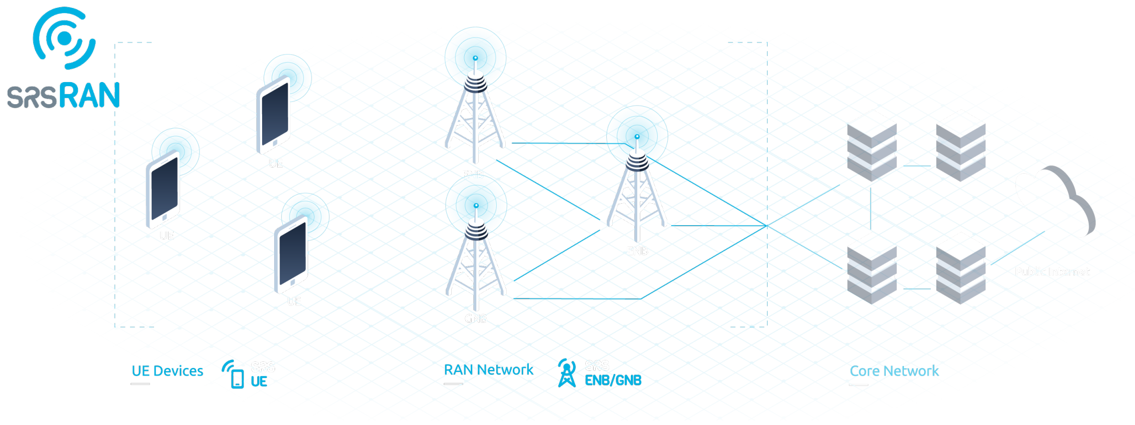

Figure 3.6: srsRAN Architecture

srsRAN library deals with buffers of samples in system memory, thus it is able to work with any RF front-end. It currently provides interfaces to the Universal Hardware Driver (UHD), giving support to the Ettus USRP family of devices. Our requirements thus are a USRP B210 RF-frontend, the number of antennas it requires and also a CPU set (Intel x86-64) running the software application of the E-UTRAN (in this case srsLTE eNB). These tools constitute the necessary hardware & software equipment that our implementation requires [12].

3.2.3 Kubernetes

At June 2014 Google Developer Forum, Kubernetes was announced, an open source cluster manager for Docker containers. Kubernetes plays the role of a container orchestration tool which automates container deployment, scaling and load balancing, grouping the former into logical units.

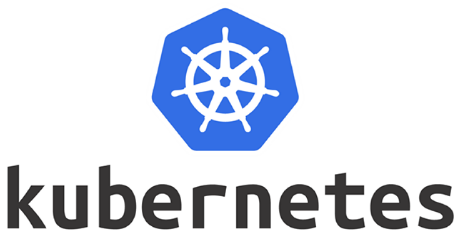

Figure 3.7: Kubernetes Logo

3.2.3.1 Kubernetes Architecture

The following figure 3.8 illustrates a general Kubernetes cluster.

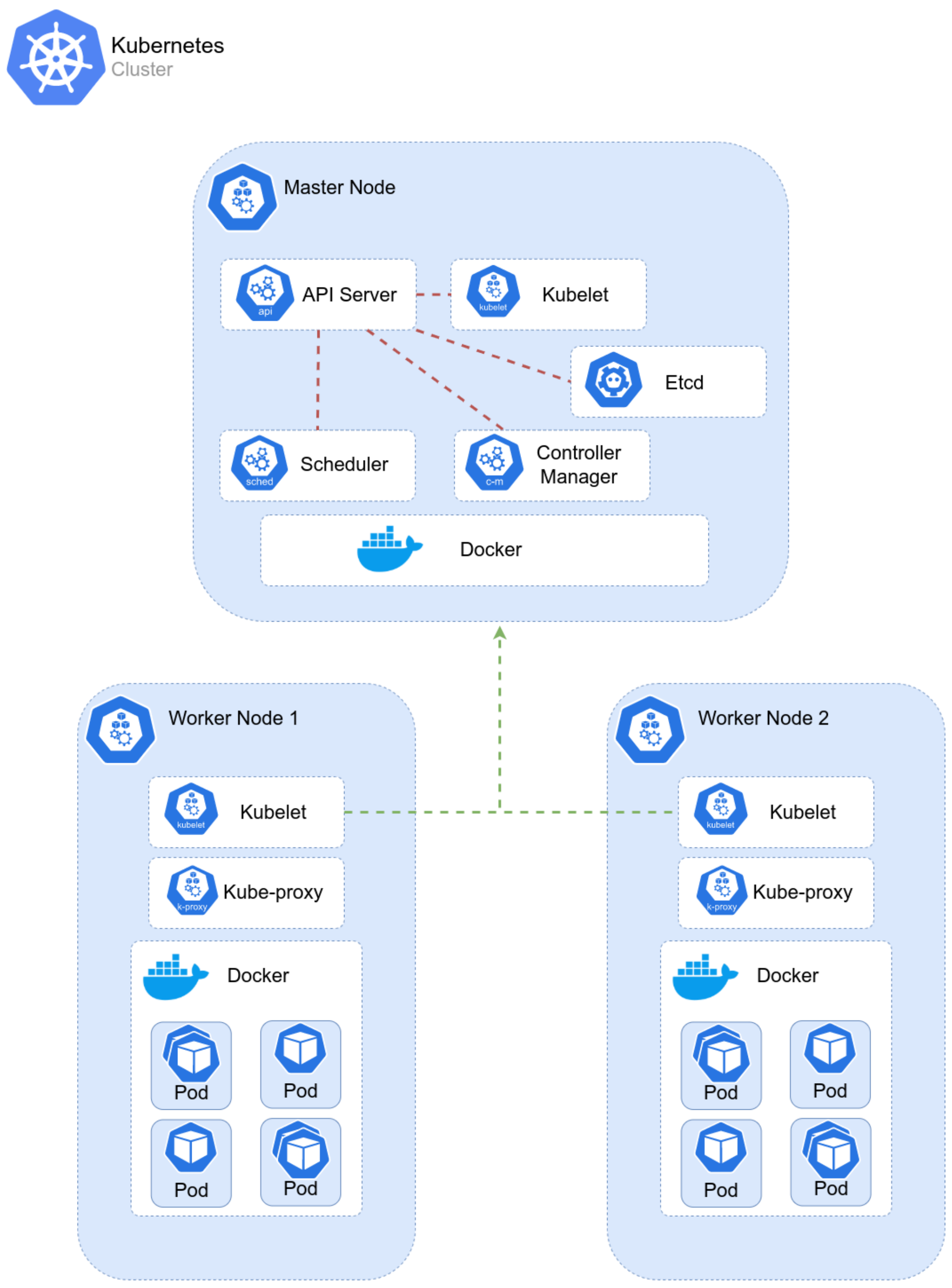

Figure 3.8: Kubernetes Architecture

The central component of Kubernetes is the cluster. A cluster is made up of many virtual or physical machines that each serve a specialized function either as a master or as a worker node. Each node hosts groups of one or more containers, which contains applications, and the master communicates with nodes about when to create or destroy containers. At the same time, it tells nodes how to re-route traffic based on new container alignments.[13]

3.2.3.2 Master Node

The Kubernetes master is the control plane, from which administrators and other users interact with the cluster to manage the scheduling and deployment of containers. The master stores the state and configuration data for the entire cluster in etcd, a persistent and distributed key-value data store. Each node has access to ectd, and through it, nodes learn how to maintain the configurations of the containers they're running. Master nodes communicate with the rest of the cluster through the kube-apiserver, the main access point to the control plane.

3.2.3.3 Worker Node

Each Worker Node must be configured with a container runtime, like Docker.The container runtime starts and manages the containers as Kubernetes deploys them to nodes in the cluster. Kubelet is an agent process that runs in alla nodes and is responsible for managing the state of the node: starting, stopping, and maintaining application containers based on instructions from the control plane. The kubelet collects performance and health information from the node, pods, and containers that it runs. It shares that information with the control plane to help it make scheduling decisions. The kube-proxy is a network proxy that runs on worker nodes and works as a load balancer for services running on each node.

3.2.3.4 Pod

The basic scheduling unit is a pod, which consists of one or more containers sharing resources among each other. Each pod is assigned a unique IP address within the cluster, allowing the application to use ports without conflict. Thei main purpose of a pod is to support co-located and managed helper programs, like proxies, file and data loaders, log and checkpoint backups. In practice, microservices are containerized and deployed on a Kubernetes cluster as pods. Pods can be created manually as well as by controllers. A pod's template along with its desired number of replicas and other information such as upgrade strategy and labels are included in a controller specification.Once the controller is deployed to the cluster, it creates the desired number of pods based on the provided template and continuously maintains their number equal to the desired number. Controllers are watch loops that continuously work to bring the current state of the application to its desired state.[14]

3.2.3.5 Similar to Kubernetes

Kubernetes is not the only option available for container orchestrator engine. There is a plethora of options but the majority of them using different forms of Kubernetes. Two of the most common used alternatives to Kubernetes are:

- Docker Swarm

  Open source solution coming from Docker's team. By default, any software, services, or tools that run with Docker containers, has full support in Swarm. It uses the same command line from Docker. Simpler to use and to install than Kubernetes, but not recommended in production environments.

- Mesos

  Open source software that provides efficient resource isolation and sharing across distributed applications or frameworks. Mesos architecture is based on a master-slave approach, where daemons run task on the slaves. Mostly recommended for multi-cloud and multi-region clusters. Mesos has more complexity than Kubernetes, thus is currently being adapted to add a lot of the Kubernetes concepts and to support the K8s API.

3.2.4 Ansible

Ansible is an open-source software provisioning, configuration management, and application-deployment tool enabling infrastructure as code. It runs on many Unix-like systems and uses agent-less architecture. Communication between workstation and managed nodes is over SSH. A typical scenario using Ansible is the following:

- Ansible connects over SSH to the host node. The user can be authenticated with password or key.
- Pushes programs to the host node, called ”Ansible modules”. These programs define how the system should be configured.
- Ansible executes those programs and transmits their output to workstation's terminal.
- Ansible removes those programs from managed node.

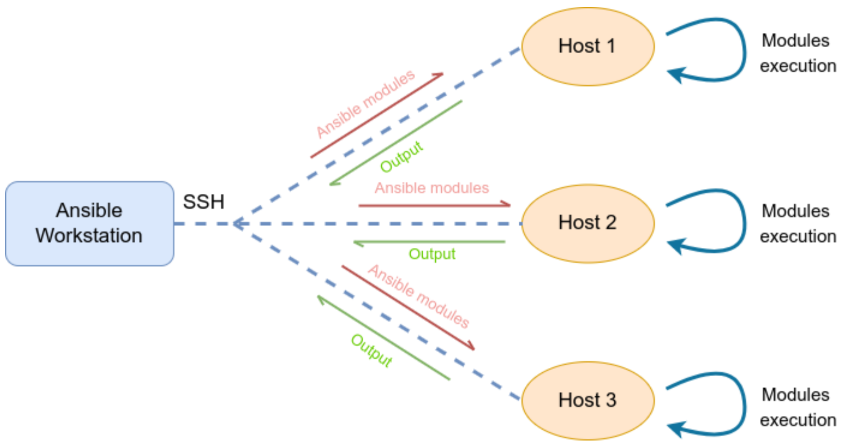

Figure 3.9: Ansible Architecture

Ansible starts up the environments according to a playbook. Playbooks are Ansible's configuration, deployment and orchestration language, which executes selected commands in wanted order. These files are expressed in YAML format and have a minimum of syntax. One playbook can contain a single or multiple play which are run in the order specified in the playbook. Also playbooks use the hosts-file that contains all the relevant information for the needed environment in order to initialize the connections between the nodes of the environment.

Chapter 4

Design

The purpose of this section is to describe the integration of the system architecture components that were mentioned previously. The 5G NFV network model, following the architecture by ETSI is combined with the deployment of VNFs in Docker containers. The latter are orchestrated and managed by Kubernetes, with an overview of its integration on top of the relative network context presented below.

4.1 Topology Initialization

The deployment flow of our network topology begins with serving and deploying four nodes on the NITOS Testbed with Linux operating system, Ubuntu 18.06 as shown below on the Figure 4.1.

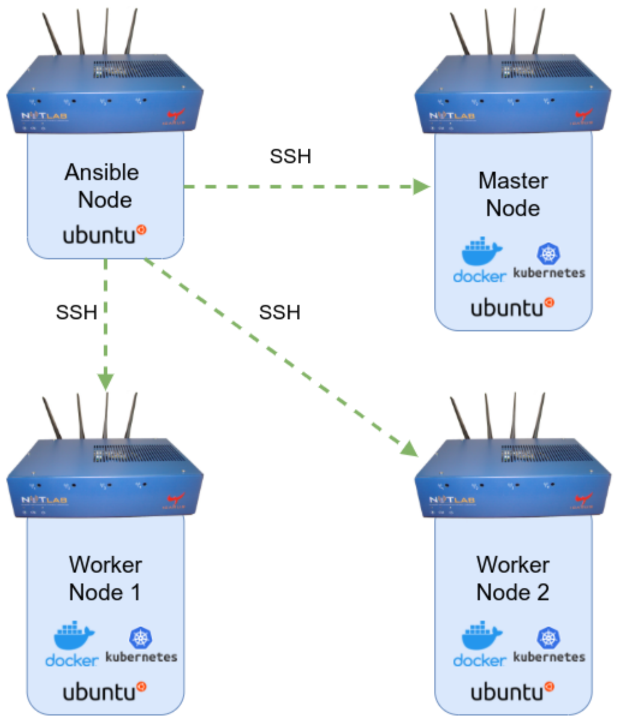

Figure 4.1: Initialization Topology in NITOS Testbed

The first one, the Ansible Node, connects with the other three nodes over SSH, labeling them as Worker or Master Node. It is responsible for installing and deploying Docker, Kubernetes and any dependencies needed in the other nodes and eventually for the initialization of the Kubernetes Cluster. The Workers node require to have USRP connected, as there will be taking place the experimentation scenario of 5G NFV model.

The Ansible Node contains the hosts file, where there is all the information needed to initial the connection between topology nodes, such as IP-addresses and node roles. All the playbooks that are used in this design operate with the host file to target the desired node and execute the specific tasks. As we mentioned before, playbooks are models of configuration processes written in YAML format. In this design the yaml files are the following:

- initial.yml

  Is responsible for the establishing the connection between the Ansible Node and the other three nodes over SSH.

- docker-k8s-dependencies.yml

  Installing Docker, Kubernetes to the Master and Workers Nodes

- master.yml

  Initializing and deploying the Master Node as Kubernetes control plane

- workers.yml
  Connecting the Worker Nodes in the Kubernetes Cluster

4.2 Kubernetes Integration

As the Kubernetes framework has been deployed and the control plane node is running in the testbed infrastructure, which has direct network access to the to the wireless nodes of the testbed, the Master Node contains the K8s control plane. The two workers shown represent 5G Network Components, so we assign the UE role to the first worker and the eNodeB, EPC roles are collocated in the second worker node. Regarding the selection of Kubernetes Network, we deploy the Flannel CNI (Container Network Interface) plugin to the cluster. Flannel is a simple overlay network that acts as a network of containers and consequently as a pod network. This feature is possible, while also we use the Multus plugin to add more than one interface to the pods.

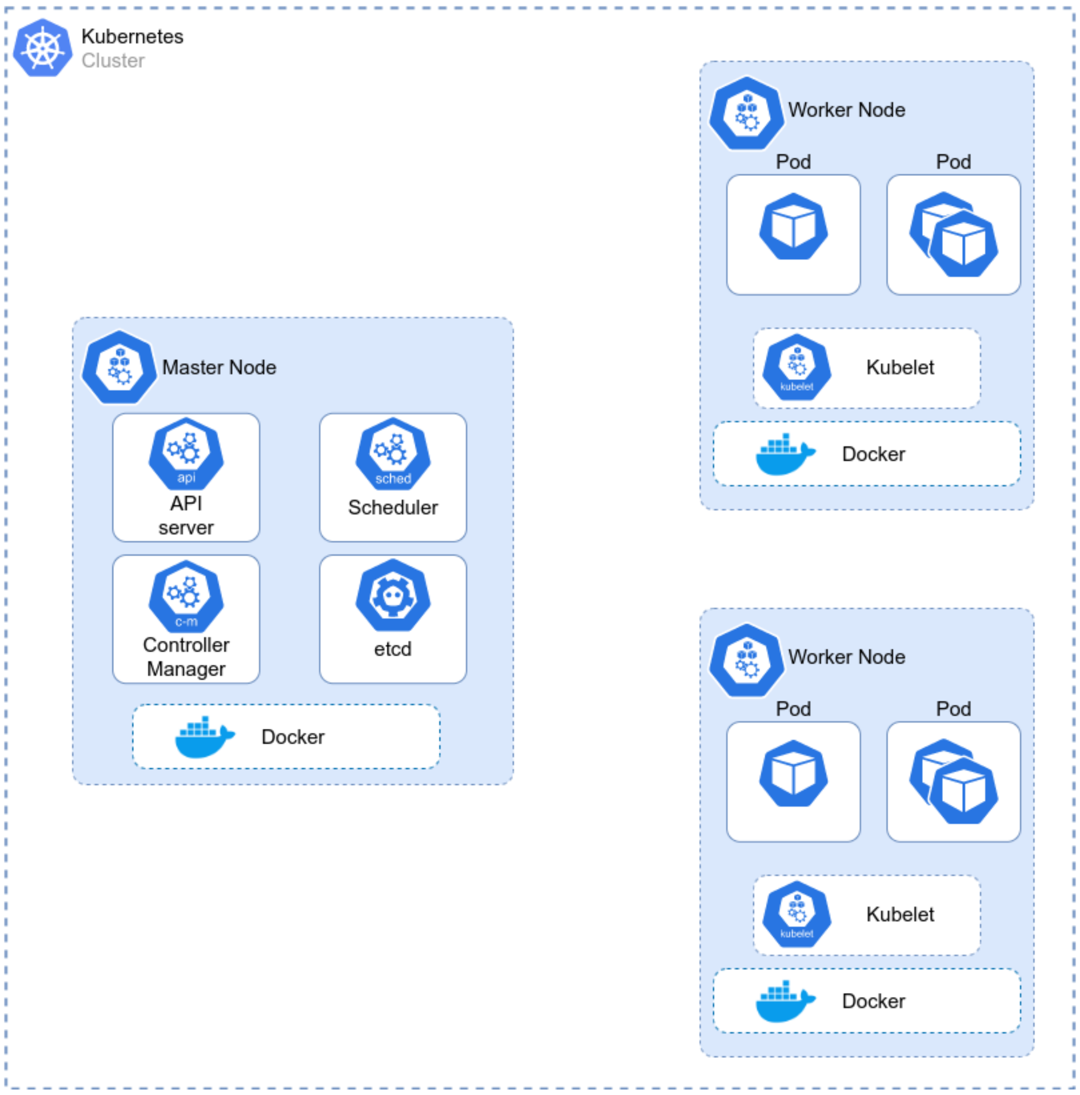

Figure 4.2: Topology after Kubernetes Intergation

4.2.1 Pods Setup

With the help of Kubernetes command-line tool, kubectl, we apply the configuration that is described in the respective YAML files in order to deploy the desired pod network. This network design consists by two pods deployed in Worker Node 1 where the roles assigned were described previously. All following YAML files utilize srsRAN Docker container images.

- srsepc-pod.yml

  Deploying a pod as EPC service to Worker Node 1, using the srsepc image.

- srsenb-pod.yml

  Deploying a pod as eNodeB service to Worker Node 1, using the srsenb image.

- srsue-pod.yml

  Deploying a pod as the UE component on the network to Worker Node 2, using the srsue image.

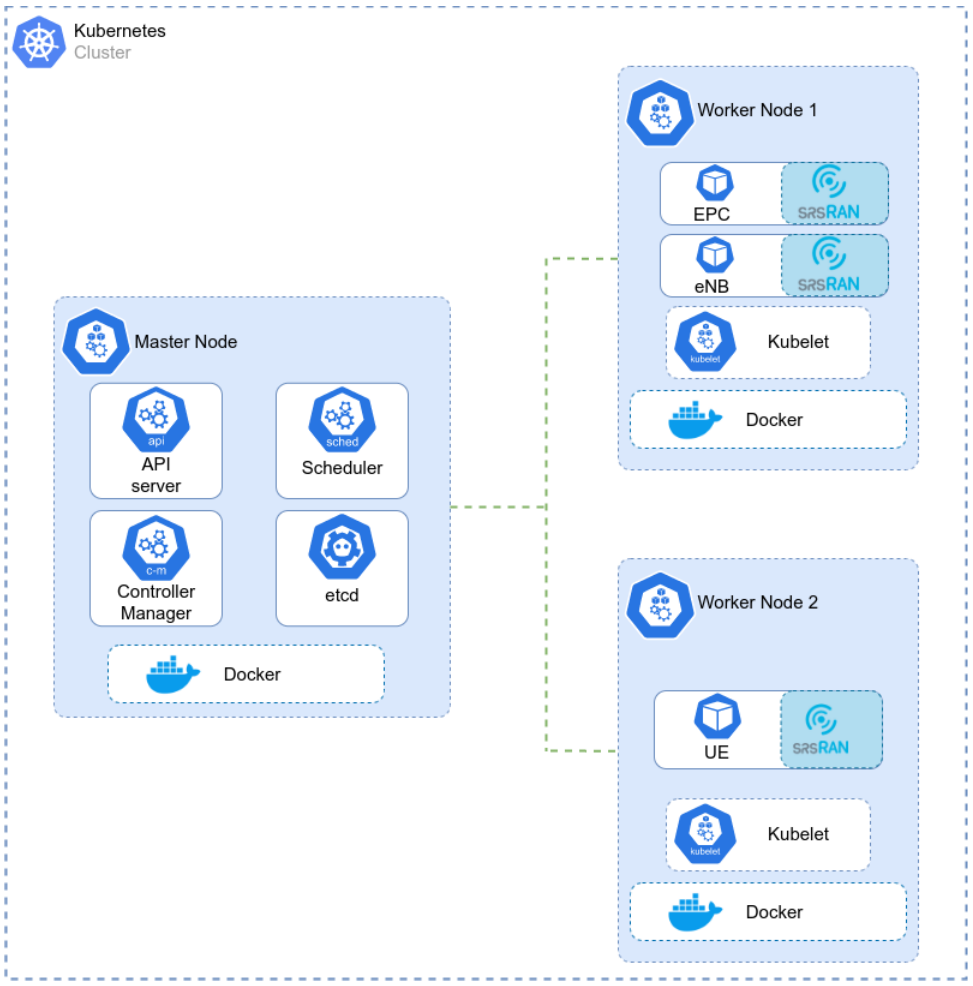

Figure 4.3: Thesis's Kubernetes Cluster

4.3 Testbed Integration

NITOS testbed uses a scheduler interface in order to give the opportunity to researchers to select the number of nodes they want to use within a timeslot. Timeslots are a unit of time where nodes can be scheduled. During these time-slots we can operate our virtualized 5G solution we the tools described before. After all network components are initialized as containers in the pods network, the firing order is EPC, eNB and finally UE. In this setup the eNB component is connected to the core network (EPC) while the UE, after registering with the core receives connectivity from the eNB. For connectivity troubleshooting we test with the ping utility or common network traffic tools, like iperf. Both traffic directions must be tested to make sure that UE reaches core network and vice versa.

Chapter 5

Conclusion & Future Work

In this thesis, we first provided an outline of the Radio Access Network evolution across the last two major generations. We then provided an overview of our system and network components, with an emphasis on extensive network disaggregation and virtualization of the logic components that make up typical procedures met in current 5G deployments. These containerized software instances are adapted according to the needs of the network that includes either the incorporation of more nodes or simply just more clients that need to be connected and maintained in real-life workloads that our network deployments comes across. By a proof-of-concept implementation, we covered how can a developer/researcher of the relevant fields we have worked on, can setup, deploy and manage a virtualized 5G network in real-life testbed conditions.

Concerning future efforts and directions, visualization is key concerning the real-time monitoring of consumption and usage by our deployment. Web-based dashboards can serve this purpose, easily integrated with the main application (see e.g. Kubernetes dashboard, Prometheus, etc.). Further, we can investigate and integrate various cutting-edge use cases, like Machine Learning-based provisioning services that are known to require intensive CPU usage and/or GPU devices, also viewing them on a web-based visualization service (e.g. Tensorflow's Tensorboard, etc.).

Bibliography

[1] Navid Nikaein Osama Arouk. Kube5g: A cloud-native 5g service platform. Communications Systems Department - EURECOM, Biot, France, 2020.

[2] Jacobus Van der Merwe Jon Larrea, Mahesh K. Marina. Nervion: A cloud native ran emulator for scalable and flexible mobile core evaluation. The University of Edinburgh, 2022.

[3] `https://el.wikipedia.org/wiki/LTE`.

[4] `https://el.wikipedia.org/wiki/4G`.

[5] `https://www.digi.com/blog/post/5g-network-architecture`,.

[6] `https://www.sdxcentral.com/5g/definitions/5g-network-slicing/`,.

[7] Jean-Philippe Wary Ghada Arfaoui, Jos´e Manuel Sanchez Vilchez. Security and resilience in 5g: Current challengesand future directions. In *26th Conference on Design of Circuits and Integrated Systems (DCIS)*, Orange Labs, Chˆatillon, France, Aug. 2017.

[8] Daniele Decaro. Integration of smart orchestration in an open source nfv framework. POLITECNICO DI TORINO, Mar. 2019.

[9] Kapridakis Iosif. Introduction, overview experimentation analysis of etsi open source management orchestration (osm mano) architecture. UNIVERSITY OF PIRAEUS DEPARTMENT OF DIGITAL SYSTEMS, Sep. 2018.

[10] Cloud-native network functions. `https://ligato.io/cnf/cnf-def/`.

[11] Zarafetas C. Valantasis A. Korakis-T. Tassiulas L Makris, N. Integrating nfv-mano with wireless services: Experiences and testbed development. In *O4SDI - 4th Workshop on Orchestration for Software-Defined Infrastructures (at) 2018 IEEE NFV-SDN (NFV-SDN'18 - O4SDI);*, University of Thessaly, Volos, Greece, Sep. 2018.

[12] SADIQ IQBAL and JEHAD M HAMAMREH. A comprehensive tutorial on how to practically build and deploy 5g networks using open-source software and general-purpose, off-the-shelf hardware.

[13] `https://newrelic.com/blog/how-to-relic/what-is-kubernetes`,.

[14] Leila Abdollahi Vayghan. Kubernetes as an availability manager for microservice based applications. Concordia University, Montreal, Quebec, Canada, Aug. 2019.