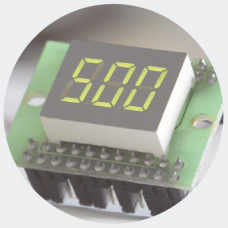
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






Orchestration and Lifecycle Management over Virtualized Network Functions

PEDRO MIGUEL MENDONCA BARBOSA
novembro de 2019

POLITÉCNICO
DO PORTO

Orchestration and Lifecycle Management over Virtualized Network Functions

Pedro Miguel Mendonça Barbosa

Mestrado em Engenharia Eletrotécnica e de Computadores
Área de Especialização de Telecomunicações
Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
2019

Cofinanciado por:

UNIÃO EUROPEIA
Fundo Europeu
de Desenvolvimento Regional

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha de Unidade Curricular de Tese/Dissertação, do 2.º ano, do Mestrado em Engenharia Eletrotécnica e de Computadores

Candidato:Pedro Miguel Mendonça Barbosa, Nº 1131203, 1131203@isep.ipp.pt
Orientação científica: Professor Doutor Jorge Mamede, jbm@isep.ipp.pt
Empresa: Altran Portugal
Co-orientador: Professor Doutor Sérgio Figueiredo, sergio.figueiredo@altran.com

Mestrado em Engenharia Eletrotécnica e de Computadores
Área de Especialização de Telecomunicações
Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
2019

Cofinanciado por:

UNIÃO EUROPEIA
Fundo Europeu
de Desenvolvimento Regional

"It always seems impossible until it's done"

Nelson Mandela

Agradecimentos

A elaboração deste relatório não seria possível sem a orientação, acompanhamento e força de várias pessoas. Gostaria de agradecer a todas as pessoas que direta ou indiretamente contribuíram para que a realização deste projeto fosse possível. Quero manifestar gratidão a todas essas pessoas.

Primeiramente quero agradecer ao Professor Doutor Jorge Mamede, orientador do ISEP para este projeto, pela disponibilidade e tempo despendido na orientação e acompanhamento.

De seguida ao Professor Doutor Sérgio Figueiredo e ao Engenheiro Bruno Parreira, co-orientadores e líderes do projeto M5G na Altran Portugal, pelo projeto que desenvolvi, pelo acompanhamento, orientação e transmissão de conhecimentos, bem como a amabilidade e a fácil integração que me proporcionou na empresa.

Aos colegas de trabalho que desde o início sempre mostraram amizade, companheirismo, paciência, apoio e entreajuda.

A todos os meus amigos, com consideração especial à "Matilha", quer sejam da faculdade ou não, que, de alguma forma, me ajudaram neste percurso pela amizade, camaradagem e entreajuda.

Aos meus pais, irmã e todos os meus familiares que me apoiaram desde esta etapa, pelo carinho, compreensão e amabilidade.

Por fim, à minha namorada, Cristina Moreira, pelo apoio incondicional, pela ajuda que prestou, pelo carinho, pela amabilidade, pela paciência e pelo tempo disponibilizado.

A todos estes um muito obrigado.

Resumo

A crescente evolução da nova geração de redes móveis aproxima a comercialização de redes de quinta geração da realidade. Os operadores móveis já dizem possuir dispositivos preparados para o 5G e já demonstram protótipos de serviços criados e sustentados para estas redes. Na realidade, as Redes de Acesso Radio (Radio Access Network - RAN) planeadas para o 5G já estão desenvolvidas e testadas, existindo já algumas redes de acesso radio com novas antenas preparadas para 5G em várias partes do mundo.

Porém, as redes 5G vão mais além que a evolução das RAN, sendo a principal novidade desta geração de redes móvel a virtualização das redes centrais, incluindo nestas serviços e funções de rede. A Virtualização de Funções de Rede (Network Function Virtualization - NFV) está a ser normalizada pela European Telecommunications Standards Institute (ETSI). Esta tecnologia está a ser usada pela 3rd Generation Partnership Project (3GPP) para normalizar as redes 5G, nomeadamente a virtualização das suas redes centrais.

A virtualização não só irá permitir o surgimento de novos serviços e funções de rede como também permitirá virtualizar serviços e funções de rede antigas. A virtualização não só irá melhorar significativamente as redes móveis, como também será mais fácil a gestão e a orquestração destas.

Em Portugal, o consórcio 5GO, que envolve várias empresas com uma vasta experiência de Pesquisa e Desenvolvimento (Research & Development - R&D) na área de Telecomunicações, desenvolve o projeto Mobilizador 5G. Esta iniciativa portuguesa visa desenvolver soluções inovadoras para a rede 5G.

Esta dissertação aborda inicialmente uma análise às tecnologias que permitem as redes 5G. De seguida foca-se nas plataformas de gestão e orquestração de redes existentes e por fim, nas operações de gestão de ciclo de vida (Lifecycle Management - LCM) que estas plataformas poderão realizar nas funções de redes. O objetivo final será testar e analisar essas operações em funções de rede através de uma das plataformas existentes.

Palavras-Chave: Orquestração, NFV, MANO, ONAP, 5G, LCM, Virtualização.

Abstract

The growing evolution of the new mobile generation networks brings the commercialization of 5th generation networks closer to reality. Mobile operators already claim to have 5G-ready devices and already demonstrate prototypes of services created and sustained for these networks. In fact, the Radio Access Network (RAN) planned for 5G are already developed and tested, and there are already some radio access networks with new 5G-ready antennas in diverse parts of the world.

However, 5G networks go beyond the evolution of RANs, with the main novelty of this mobile network generation being the virtualisation of central networks, including in these services and network functions. Network Function Virtualization (NFV) is being standardized by the European Telecommunications Standards Institute (ETSI). This technology is being used by the 3rd Generation Partnership Project (3GPP) to standardise 5G networks, particularly the virtualisation of core networks.

Virtualization will not only allow the development of new network services and functions, it will also allow the virtualization of legacy network services and functions. Virtualization will not only significantly improve mobile networks, but also make it easier to manage and orchestrate them.

In Portugal, the 5GO consortium, which involves several companies with a vast experience in Research & Development (R&D) in the area of Telecommunications, develops the project Mobilizador 5G. This Portuguese initiative aims to develop innovative solutions for the 5G network.

This dissertation initially addresses an analysis of the technologies that enable 5G networks. It then focuses on the existing network management and orchestration platforms and, finally, on the Lifecycle Management (LCM) operations that these platforms can perform in network functions. The final objective will be to test and analyse these operations in network functions through one of the existing platforms.

Keywords: Orchestration, NFV, MANO, ONAP, 5G, LCM, Virtualization.

Content

List of Figures

List of Tables

Acronyms

3GPP 3rd Generation Partnership Project

4G 4th Generation Network

5G 5th Generation Network

5GPPP 5G Infrastructure Public Private Partnership

A-CPI Application-controller Plane Interface

AAF Application Authorization Framework

AAI Active and Available Inventory

API Application Programming Interface

APP-C Application Controller

ATIS Alliance for Telecommunication Union Radiocommunications

BSS Business Support System

CDT Controller Design Tool

CPD Connection Point Descriptor

CPE Customer Premises Equipment

CPU Central Processing Unit

CRUD Create, Read, Update, Delete

COTS Commercial off-the-shelf

D-CPI Data-controller Plane Interface

DCAE Data Collection, Analytics and Events

DG Directed Graph

DMaaP Data Management as a Platform Data Bus Controller

EM Element Management

eMBB Enhanced Mobile Broadband

ESR External System Register

ETSI European Telecommunications Standards Institute

FCAPS Fault-management, Configuration, Accounting, Performance and Security

GVNFM Generic Virtualised Network Function Manager

HTTP Hypertext Transference Protocol

IaaS Infrasctructure as a Service

IETF Internet Engineering Task Force

IM-NBI Information Model and Northbound API

IMS IP Multimedia Subsystem

IMT International Mobile Telecommunications

IoT Internet of Things

ISG Industry Specification Group

IT Information Technology

ITU-R International Telecommunications Union Radiocommunication

JSON JavaScript Object Notation

LCM Lifecycle Management

M5G Mobilizador 5G

MANO Management and Orchestration

mMTC Massive Machine Type Communications

MON OSM Monitoring Module

MSB Microservices Bus

N2VC Network To Virtualised Network Function Configuration

NaaS Network as a Service

NETCONF Network Configuration Protocol

NF Network Function

NFFG Network Function Forwarding Graph

NFV Network Function Virtualisation

NFVI Network Function Virtualisation Infrastructure

NFVIaaS Network Function Virtualisation Infrastructure as a Service

NFVO Network Function Virtualisation Orchestrator

NGMN Next Generation Mobile Networks Alliance

NS Network Service

NSD Network Service Descriptor

NV Network Virtualisation

ONAP Open Network Automation Platform

OOM ONAP Operations Manager

OS Operating System

OSI Open System Interconnection

OSM Open Source MANO

OSS Operations Support System

PNF Physical Network Function

PNFD Physical Network Function Descriptor

PPS Products, Processes and Services

QoS Quality of Service

R&D Research & Development

RAM Random Access Memory

RAN Radio Access Network

RFC Request for Comments

RO Resource Orchestrator

RPC Remote Procedure Call

SDC Service Design & Creation

SDN Software Defined Network

SDN-C Software Defined Network Controller

SO Service Orchestrator

SSH Secure Shell

SVNFM Specified Virtualised Network Function Manager

TOSCA Topology and Orchestration Specification for Cloud Applications

UFW Uncomplicated Firewall

UI User Interface

UI-Apps User Interface Applications

URLLC Ultra-Reliable and Low Latency Communications

VCA VNF Configuration and Abstraction

vCPU Virtual Computing Processor Unit

VDU Virtualisation Deployment Unit

VF Virtual Function

VF-C Virtual Function Controller

vFW Virtual Firewall

VID Virtual Infrastructure Deployment

VIM Virtualised Infrastructure Manager

VLD Virtual Link Descriptor

VSP Vendor Software Products

VM Virtual Machine

VNF Virtualised Network Function

VNFC Virtualised Network Function Component

VNFD Virtualised Network Function Descriptor

VNFFG Virtualised Network Function Forwarding Graph

VNFFGD Virtualised Network Function Forwarding Graph Descriptor

VNFM Virtualised Network Function Manager

WinRM Windows Remote Manager

XML Extensible Markup Language

Chapter 1

Introduction

1.1 Thesis Context

Mobile networks are evolving at a huge pace and new mobile network generation is close to being a reality. However, 5th Generation Network (5G) networks are more than Radio Access Network (RAN) and a significant amount of changes reside in the Network Core, and in the way network services are deployed and defined.

Network Function Virtualisation (NFV) and Software Defined Network (SDN) are important enablers to 5G because they will allow legacy mobile network components to be realized by virtualised functions managed efficiently in datacenters. This will allow an easier deployment of mobile networks than with current physical networks. Because of physical limitations of legacy Network Functions, Virtualisation will bring more services, however it will require more complex service orchestration and provisioning. It will enhance network services and creation of new ones, resulting in significant increase regarding mobile networks' uses.

All these technologies are being defined by organizations like European Telecommunications Standards Institute (ETSI), that standardized NFV technology, or 3rd Generation Partnership Project (3GPP) which defines all the standards pertaining to 5G networks.

1.2 Thesis Motivation

In Portugal, the 5GO consortium involves some of the companies with larger Research & Development (R&D) experience in the Telecommunications sector, like Altran, Altice Labs and Instituto de Telecomunicações. The main scope of Mobilizador 5G (M5G) project is to be an instrument for development and innovation of 5G technology addressing the integrated design and validation of

a set of products capable of being part of and providing services within the ecosystem of the future 5G networks [17].

This project is organized in six Products, Processes and Services (PPS). This thesis is included in PPS2, that focuses on products and services for the Network Core, led by Altran. Within PPS5 scope, solutions resulting from PPS1 (products and services for network edge), PPS2, PPS3 (products and services for Machine-to-Machine communication) and PPS4 (products and services for human communication) will be deployed and integrated into an End-to-End 5G testbed. [16]

1.3 Thesis Objectives

This thesis has the following objectives, which are addressed throughout its various phases:

- **Objective 1 - Analysis on ETSI NFV standardization:** This thesis aims to analize the ETSI NFVs specification, more specifically NFV's architecture, function and service descriptor's models, like Topology and Orchestration Specification for Cloud Applications (TOSCA), of key concepts like network services, network functions, connection points, etc., and NFVs relevance for the evolution of 5G network technology. This analysis is important to understand how NFV affects 5G networks and to understand how is planned by ETSI NFV to orchestrate 5G core networks;
- **Objective 2 – Analysis on open-source orchestration software:** To analyse open-source software currently available for orchestration of virtualised resources in 5G networks context, e.g. Open Source MANO (OSM) or Open Network Automation Platform (ONAP), as well as acquisition of hands-on experience in at least one of the analysed platforms by means of a short study on them;
- **Objective 3 – Orchestration platform integration on a cloud management platform:** To integrate the chosen orchestration platform with a datacenter management software to offer Cloud services;
- **Objective 4 – Evaluation of Lifecycle Management operations in network services:** To execute, test and evaluate the Lifecycle Management (LCM) operation on a network service.

1.4 Contributions

The development of this dissertation contributed to a publication entitled "Addressing end-to-end Orchestration of Virtualized Telco Services using ONAP

in a R&D environment" for the InForum 2019 conference held at the University of Minho. [21]

It also contributed to the Mobilizador 5G project within the 5GO consortium [17].

1.5 Document Structure

This chapter explained the context, motivation and objectives of this thesis. The remainder of the document includes:

- **Chapter 2:** describe the current state of the art of 5G networks and its key enablers, like NFV and SDN;
- **Chapter 3:** describes the reference platforms for Service Lifecycle Management and Orchestration, more precisely OSM and ONAP;
- **Chapter 4:** describes the implemented solution for the LCM operations, explaining the platform setup experience and describing the LCM operations and the protocols used to execute them;
- **Chapter 5:** Presents the evaluation targets for the infrastructure tests and validation for LCM operations, the target Virtualised Network Function (VNF) description. Also, the results and analysis of the ran tests are presented. In the end, the identified challenges and difficulties encountered through the development phase are also specified;
- **Chapter 6:** presents the conclusions that relate to the objectives, final conclusions and future work of this dissertation.

1.6 Acknowledgments

This work is supported by the European Regional Development Fund (FEDER), through the Regional Operational Programme of Lisbon (POR LISBOA 2020) and the Competitiveness and Internationalization Operational Programme (COMPETE 2020) of the Portugal 2020 framework [Project 5G with Nr. 024539 (POCI-01-0247-FEDER-024539)].

Chapter 2

5G and key enablers

2.1 5th Generation Networks

Current section describes mobile networks' fifth generation (5G), its objectives and motivations, highlighted features and use cases. Subsequently, Network Virtualisation (NV) and its role in 5G are described, as well as technologies that support the virtualisation.

2.1.1 5G and standardization

5G network's principles and architecture are defined by several entities whose objective is to develop a standard vision of upcoming generation of mobile networks. 3GPP is an entity that unites some telecommunications standard development organizations like ETSI, representing Europe, and Alliance for Telecommunication Union Radiocommunications (ATIS) representing United States. [14] Besides concrete proposals from companies and individual contributors, 3GPP receives inputs from entities such as 5G Infrastructure Public Private Partnership (5GPPP) [19] and Next Generation Mobile Networks Alliance (NGMN). [42]

2.1.2 Performance Targets

Differently from conventional networks, 5G networks adopt a service-based architecture. The abstraction given by virtualisation gives support to a large variety of services, different traffic loads requirements and diverse user communities.[15] The envisaged role of 5G is to provide a universal communication environment capable of responding to all kinds of needs across a wide spectrum of industry, service and security domains. To achieve those requirements, 5G needs to be programmable, secure, ubiquitous, flexible, dependable and needs to preserve privacy. The technology must also be power-efficient to reduce power consumption decreasing energy cost per bit. [39]

"IMT for 2020 and beyond" (International Mobile Telecommunications (IMT)) defines International Telecommunications Union Radiocommunication (ITU-R) vision and target requirements that radio communications from 2020 and beyond - corresponding to 5G networks – should fulfil. [54] Figure 2.1 shows the main objectives for 5G to start deploying in 2020.

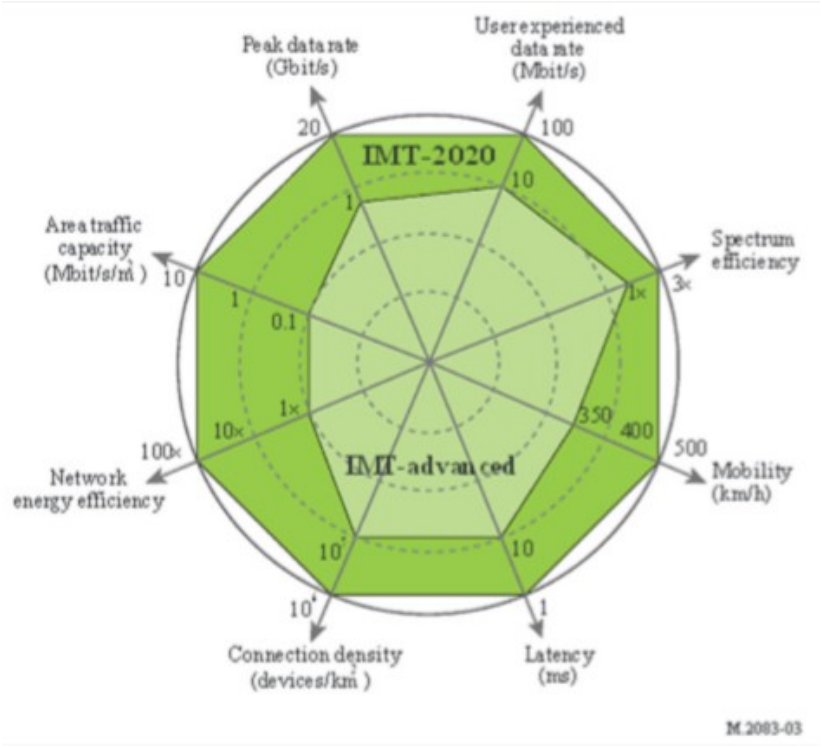


Figure 2.1: Evolution between IMT-Advanced (4G) and IMT-2020 (5G) [54]

According to the figure above, the targets for 5G networks are [37]:

- **Peak data rate:** more than 20 times the IMT-advanced, reaching 20 Gbit/s under ideal conditions;
- **User experienced data rate:** 10 times more than IMT-advanced, that is 100 Mbit/s available ubiquitously across the coverage area;
- **Area traffic capacity:** 100 times more capacity giving 10 Mbit/s/m2;
- **Spectrum efficiency:** expected to be 3 times greater than IMT-advanced or more with further developments;
- **Network energy efficiency:** energy efficiency is expected to be 100 times higher;

- **Mobility:** 5G is expected to cover, with adequate Quality of Service (QoS) users that travel at speeds around 500 km/h. This is being planned for supporting, for instance, use cases within high-speed trains;

- **Connection density:** that parameter is expected to be 100 times higher than the current technology, that would be able to reach 106 devices per km2;

- **Latency:** another one of the biggest innovations and challenges for 5G is reducing 10 times the latency, trying to go, in some cases, under 1ms, for example, autonomous driving.

2.1.3 Highlighted Features

With realization of a highly mobile and fully connected society, supported by 5G, a wide range of new services is expected to emerge in various areas, such as industrial automation and vertical marketing. [20] Besides, it demands machine-to-machine and machine-to-human type applications for making life safer and convenient. All combined, such demands infer increasing mobile traffic and necessity of better performance and QoS. [20] So, 5G network must respond to those necessities, and to meet them, 5G has some enablers like Network Slicing and Service-based Architecture. These two enablers will be explained in next subchapters.

2.1.3.1 Network Slicing

Network Slicing is defined by 5GPPP like "**an end-to-end concept covering all network segments including radio networks, wire access, Network Core, transport and edge networks**". [20] This means that network slicing combined with virtualisation can provide multiple custom networks isolated on the same infrastructure to serve a huge variety of users in all End-to-End extension. [15]

5G network infrastructure will support deployments of one or more slices. Also according to 5GPPP, "**network slice is a composition of adequately configured network functions, network applications, and underlying cloud infrastructure (physical, virtual or even emulated resources, RAN resources etc.), that are bundled together to meet the requirements of a specific use case (...), coupled with a business purpose.**" [20] This implies that slices are programmable and defined in order to meet one or more client's requirements. Figure 2.2 shows an example of network infrastructure with a stack of slices running in it.

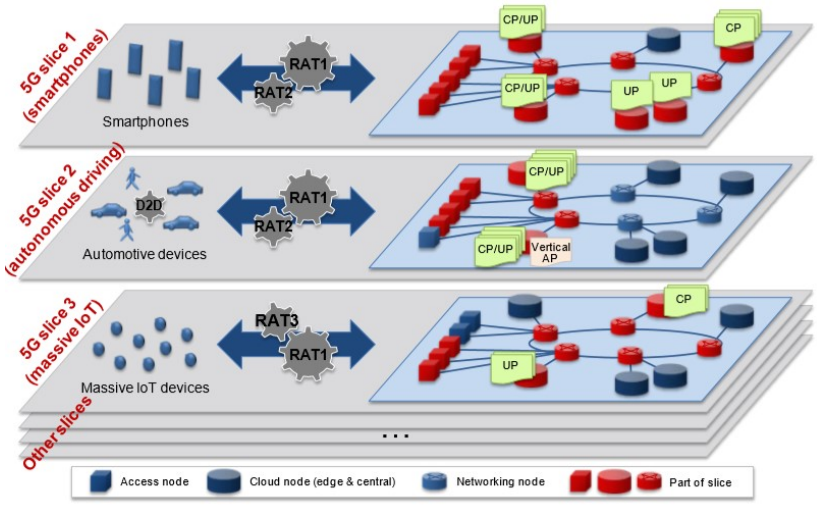


Figure 2.2: Stack of network slices in network infrastructure [40]

2.1.3.2 Service-based Architecture

Mobile core network is responsible for functions such as session management, mobility, authentication and security, critical to providing services. 5G networks, by its service-based architecture, enables more communicational flexibility between its network function. In 5G networks, the service-based architecture consists in a control plane network function that allows other authorized network functions from a communication ”bus”, to access to the services. In this architecture, all the network function only use service-based interfaces to interact. Also, service-based architecture decouples the end-user service from the underlying network and infrastructure, enabling functional and service agility. Allied with virtualisation, the operator can easily add, remove or modify VNF from a network processing path (functional agility) and create new specific services (service agility). [24]

According to NGMN [41], service-based architecture shall bring the following benefits to 5G:

- **Updating production network:** individual services can be upgraded with minimal impact to other services);
- **Extensibility:** each service can directly interact with other services. The Service-based interface can be easily extended;
- **Modularity and reusability:** network composed by modular services, supporting 5G features such as network slicing. With proper authorization, services can invoke other services, enabling the reusability of services;

- **Openness:** information, management and control functions of a 5G network can be exposed through specific services.

For more information about service-based architecture, see [24] and [41].

2.1.4 Use Cases

According to ITU-R, 5G networks will be deployed in three major use cases, expressed in Figure 2.3.

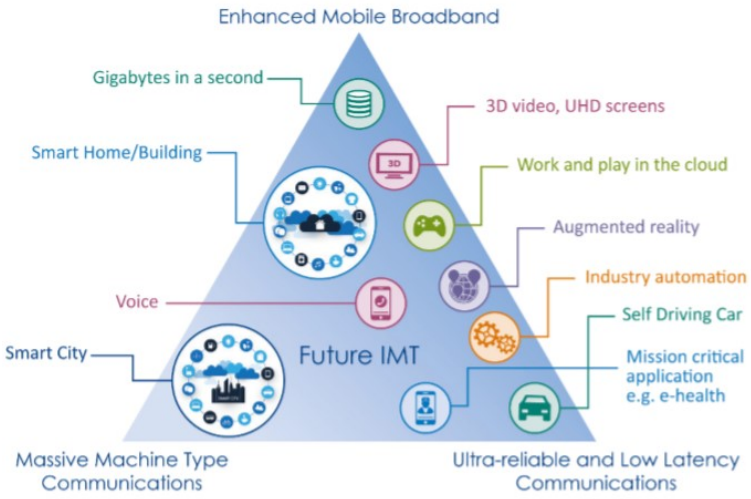


Figure 2.3: 5G major use cases [22]

Enhanced Mobile Broadband (eMBB) is the primary use case to be supported by 5G networks because it is essentially achievable through an increase in throughput, enabling high-speed mobile connections in highly crowded areas, and novel demanding mobile multimedia experiences with the highest quality. [22]

Ultra-Reliable and Low Latency Communications (URLLC) is a general use case demanding very low latency connections, high throughput and high availability like autonomous vehicles or industrial automation that need an almost instant response to commands. [54]

Massive Machine Type Communications (mMTC))is a use case for a demand for a large number of connected devices in the network transmitting small-sized data. It will be used for supporting use cases, such as smart cities and energy monitoring. [54]

2.1.5 Virtualisation

Networks are evolving according to the availability of virtualisation technologies, becoming more dynamic and flexible. [39] In the case of platform virtualistion, this concept breaks some network's limitations imposed by physical resources, virtualising them and allocating them in Virtual Machine (VM). [18] This technology enables to deploy several network functions on a single machine, sharing those functions to multiple users. Also, it enables the creation of virtual networks, giving the opportunity to implement independent virtual infrastructures for each tenant easing the development of new business models, and allowing to differentiate some specialized communication infrastructures. The first step towards virtualisation of the network, its functions and services, as well as their interconnection, has been made possible with advances spanning from SDN and NFV technologies. [39]

2.1.5.1 Software Defined Networking (SDN)

SDN is a network architecture where, as the name implies, networks are defined by software. In that feature, the architecture decouples network data plane from the control plane, where the control plane controls numerous network devices. The relocation of control into manageable computing devices allows underlying infrastructure to be abstracted from applications and network services. Network Intelligence is centralized in Software Defined Network Controller (SDN-C) gaining a global view of the entire network. To applications and policies, the network appears to be a unique and logical switch. [49]

Base Principles

According to [49], SDN is based on three principles:

- **Decoupling of traffic forwarding and processing from control:** allows independent deployment of control and traffic forwarding and processing entities. Also, decoupling can separate optimizations of platform technology and software life cycles. The decoupling principle is reflected by SDN-C, which is responsible for management and control of resources;
- **Logically centralized control:** Network control is centralized in a SDN controller that have a entire view of the network, controlling the network from one single point.
- **Programmability of network services:** permits a client to exchange service information with an SDN-C. This principle is based on the idea that service applications and resources benefit from the exchange of information.

High level SDN architecture

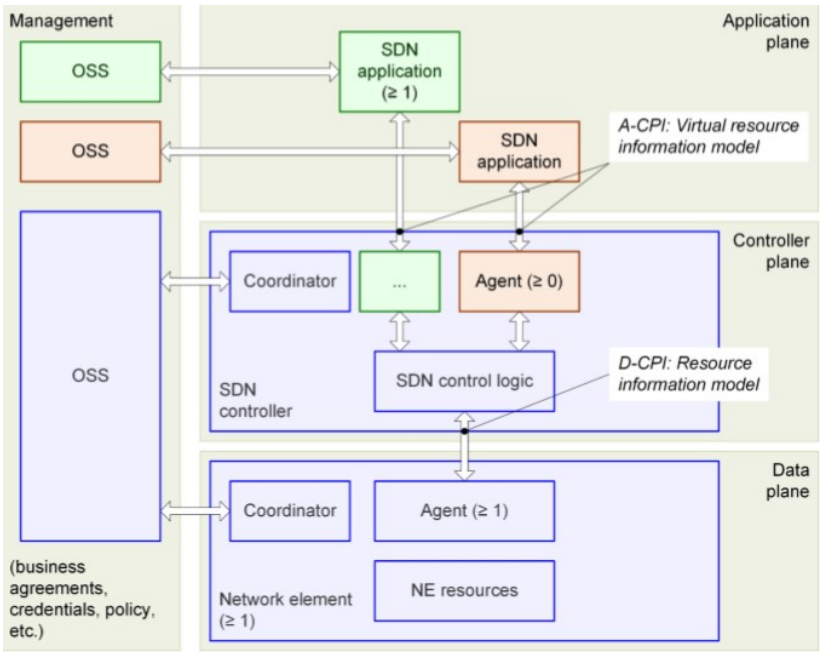


Figure 2.4: SDN architecture overview [47]

According to [47], SDN architecture is arranged in three layers:

- **Data Plane:** contains network elements and expose their capabilities to Controller Plane through Data-controller Plane Interface (D-CPI);
- **Controller Plane:** consist of SDN-C tasks, translating the applications' requirements, controlling network elements and sending crucial information to SDN applications. Services are served to applications through the Application-controller Plane Interface (A-CPI). SDN-C orchestrate competing application requests for limited network resources;
- **Application Plane:** it's where SDN applications reside and transmit their network requirements to the Controller Plane through A-CPI.

For further information about SDN, see [49] and [47].

2.1.5.2 Network Function Virtualisation (NFV)

NFV is a technology with the objective to transform actual physical networks into virtualised ones. This allows to consolidate network resources into high volume servers, switches and storage located in datacentres, network nodes and end

user locations. NFV consists in implement network functions through software that can run in Commercial off-the-shelf (COTS) resources and can be migrated to various places in the network. [26] Next chapter describes this technology with more detail.

2.2 Network Function Virtualisation Standardization

In order to define a standard for NFV and how industry should use it, the ETSI created, in 2012, the Industry Specification Group (Industry Specification Group (ISG)) for NFV (ETSI ISG NFV). [26] At the beginning of 2019, ETSI ISG already has over 300 partnerships and over 100 publications on NFV standardisation studies leading the standardisation for that technology. [36]

In the next subchapters, ETSI ISG NFV and its definition are presented. Subsequently, is described ETSI's NFV architecture and his components. Then, the relationship between NFV and SDN technologies is mentioned, as well as the impact of NFV in 5G networks. Following, some modelling languages used in NFV and associated use cases are presented. Finally, some tools for mobile network orchestration are presented.

2.2.1 Definition and main objectives

The main change of NFV in relation to the legacy Network Function (NF) is to enable additional dynamic methods in construction and management of network function graphs or sets combining between them. So, the major focus is to exploit that dynamic construction and management and the relationship between them like associated data, control, management and dependencies. [33]

The base principles of NFV will be described earlier in Section 2.2.2.

According to [29], in a high-level view, NFV objectives are:

- Rapid service innovation through software-based deployment;
- Improved operational efficiencies resulting from common automation and operating procedures;
- Reduced power usage by powering down unused hardware;
- Standardised and open interfaces between NFs so that different network elements can be distributed by different players;
- Flexibility in assigning VNFs to hardware;
- Improved capital efficiencies compared with dedicated hardware implementations.

2.2.2 Base principles

NFV follows three base principles that differ from actual standard networks [33]:

- **Decoupling software from hardware:** network elements are no longer a collection of hardware and software. Then, each hardware and software can evolve separately from each other;
- **Flexible network function deployment:** with NFV it's possible to share resources of infrastructures and make diverse functions at various times. This makes easier to implement new network services faster on a physical platform;
- **Dynamic operation:** NFV turns networks more flexible to scale VNF performance.

2.2.3 Architecture

In a high-level perspective, NFV foresees NF to be defined by software that runs on an Network Function Virtualisation Infrastructure (NFVI). There are three main domains in NFV [33]:

- Virtualised Network Functions (VNF) (Middle left on Figure 2.5);
- NFV Infrastructure (NFVI) (Bottom left corner on Figure 2.5);
- NFV Management and Orchestration (NFV Management and Orchestration (MANO) (Right side on Figure 2.5);

NFV architecture, as is showed in Figure 2.5, identifies functional blocks and the references between them. [33]

In the following subchapters, each function is briefly described for each block in the NFV architecture.

2.2.3.1 Virtualised Network Function (VNF)

As the name implies, a VNF is a Virtualised instance of a non-Virtualised legacy network function. [33] VNF is a network function that can run on NFVIs and be orchestrated by Network Function Virtualisation Orchestrator (NFVO)s and Virtualised Network Function Manager (VNFM). A VNF may implement a single network entity or a group of them. When VNF implements a group of network entities, the internal interfaces and connections do not need to be exposed. [31] Function VNF behaviour and state should be the same as NF

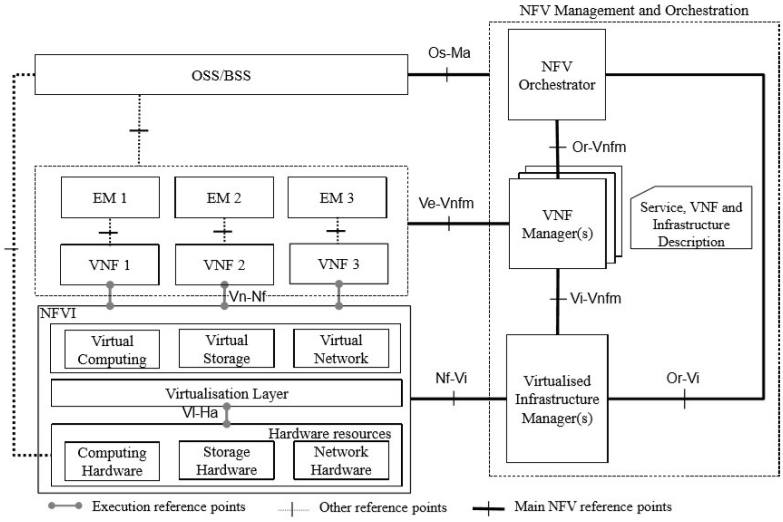


Figure 2.5: NFV architecture [33]

non-virtualised because the behaviour and the external operational interfaces of Physical Network Function (PNF) and their homologous VNFs are expected to seem the same for interfaces that will be dealing with. [33]

2.2.3.2 Network Function Virtualised Infrastructure (NFVI)

In a quick view, NFVI contains all hardware and software that together form the environment where VNFs are deployed, managed and executed. The infrastructure can be deployed in diverse locations connected between all of them. [33]

2.2.3.3 NFV Management and Orchestration (NFV-MANO)

NFV Orchestrator (NFVO)

NFVO plays a central role in this architecture. It is in charge of orchestration and management of entire NFV Infrastructure and software resource and realizes network services on the infrastructure. [33]

According to [30], NFVO has two main responsibilities:

- Orchestration of NFVI resources across multiples Virtualised Infrastructure Manager (VIM);
- Lifecycle management of network services.

Virtualised Infrastructure Manager (VIM)

VIM is responsible for controlling and managing of NFVI compute, storage and network resources. They may be specialized in handling a type of NFVI resource, like compute-only and storage-only or multiple types of NFVI resources. [30] VIM can be instantiated multiple times. [33]

According to [33], VIM functions are:

- Resource management, in charge of:
  - Inventory of software, computing, storage and network resources dedicated to NFV infrastructure;
  - Allocation of virtualisation enablers;
  - Management of infrastructure resource and allocation.
- Operations, for:
  - Visibility into and management of NFV infrastructure;
  - Root cause analysis of performance issues;
  - Collection of infrastructure error information;
  - Collection of information for capacity planning, monitoring and optimization.

Virtualised Network Function Manager (VNFM)

A VNFM is an entity responsible to manage a VNF lifecycle, since its instantiation to its termination. VNFM can be deployed to serve a single VNF or serve multiple VNFs. [33]

Generally, VNFM is deployed to assume generic common functions (Generic Virtualised Network Function Manager (GVNFM)) to manage any type of VNF . However, in NFV MANO some VNF instances support specific cases (Specified Virtualised Network Function Manager (SVNFM)) for their lifecycle management and for those cases, the functionality may be specified in a VNF Package. [30]

All information that VNFM captures of a VNF is written in a template called Virtualised Network Function Descriptor (VNFD) and stored in VNF catalogue. [30] Those two resources are briefly explained in Section 2.2.3.6.

2.2.3.4 Operations Support System/Business Support System (OSS/BSS)

Operations Support System (OSS) and Business Support System (BSS) are the combination of all operations and business support functions of operators and are not represented in NFV architecture. However, it is expected that they

exchange information with other function blocks of the architecture. They provide management and orchestration in legacy systems and have full visibility of End-to-End services provided by legacy network functions in operator's networks. [30]

2.2.3.5 Element Management (EM)

Element Management (EM) element is responsible for the normal management of one or more VNFs. [33] According to [30] this includes:

- Configuration for network functions provided by VNF;
- Error management for network functions provided by VNF;
- Accounting for the usage of VNF functions;
- Collecting performance measurement results for functions provided by VNF;
- Security management for VNF functions.

EM may cooperate with VNFM to execute management functions that need interactions with information about NFVI resources related to VNF. [30]

2.2.3.6 Templates

Services delivered by NFV technology are defined in templates that describe the attributes and requirements needed to instantiate Network Service (NS) and VNFs and manage their lifecycle. [35]

There are two main NFV templates [35]:

- **Virtualised Network Function Descriptor (VNFD):** describes one or more VNFs;
- **Network Service Descriptor (NSD):** describes one or more Network Services.

VNFD is a template included in VNF packages which describe VNFs in relation to deployment and operational behaviour. [34] A VNFD contains three main pieces of information (for more information, see [35]):

- **Virtualisation Deployment Unit (VDU):** describes the capabilities of the Virtualised containers, such as virtual Central Processing Unit (CPU), Random Access Memory (RAM) and disks;
- **Connection Point Descriptor (CPD):** describes Connection Points;

- **Virtual Link Descriptor (VLD):** describes Virtual Links.

This descriptor is used by NFVO and VNFM. [35]

NSD is a template which consists of information used by NFVO for lifecycle management of NSs. [26] NSD includes or references the following descriptors [35]:

- **Virtualised Network Function Descriptors (VNFD);**
- **Physical Network Function Descriptor (PNFD):** describes PNFs;
- **Nested NSDs;**
- **Virtual Link Descriptors (VLD);**
- **Virtualised Network Function Forwarding Graph Descriptor (VNFFGD):** describes the interconnection between VNFs.

NSD doesn't need to include or reference all of the items listed above. NSD can include more than one descriptor of each item above.

2.3 NFV and SDN relationship

NFV and SDN are two technologies that complement each other. However, both are not dependent on each other. NFV objectives can be achieved without using SDN, but with SDN principles, those objectives can be greatly enhanced and simplified. NFV supports SDN by providing the infrastructure where SDN can run. [26]

According to [47], the relationship can be resumed in the following:

1. NFV focuses on creation and management of VNFs,
2. Then, SDN helps organize VNFs into NFV network services;
3. with that, VNFs and network services become resources for construction and optimization of services for the client.

For more detailed information, see [48] and [32].

2.4 NFV Modelling Languages

For MANO operations in VNFs and network services, NFV works with VNFDs and other templates that are defined using modelling languages.

TOSCA is a declarative modelling language for describing components of a cloud application and their relationship. It uses the concept of service templates to describe cloud workloads. [35] TOSCA already have a specification made by OASIS for NFV that are used to commercial solutions. [20]

YANG is another present language in NFV-MANO's objects, more properly in NSDs. YANG is a data modelling language used to design configuration and state data manipulated by Network Configuration Protocol (NETCONF) Protocol. [50]

Another language that provides models to VNFDs is OpenStack Heat. Used more in OpenStack environment, Heat is a declarative model for orchestration OpenStack resources and managing their lifecycle. [38]

2.5 Reference Use Cases

The next subchapters reveal some use cases realized using NFV. For more use cases and more detailed information on the next described use cases, see [29] and [33].

2.5.1 Virtualisation of the Home Environment

Current operators provide home services using network-located systems or/and Customer Premises Equipment (CPE) devices. This CPE devices sometimes demands on-site services to install and support the client services [33].

With the NFV technology, this services and devices can be virtualised and can be removed from the home environment, reducing the need for hardware-specific devices, reducing the costs and enabling the urge of new services, enhancement of QoS And ease of service updates [33].

2.5.2 Virtualisation of Mobile Core Network and IMS

NFV targets to reduce network complexity and related operational issues aggravated by an exponential growth of devices connected to mobile networks. The virtualisation of the Network Core enables to increase the network efficiency and accommodate increased demand for particular services without relying on services restrictions, for example, a sudden increase of voice communication on a network. [33]

According to [33], possible advantages of virtualisation of the mobile core network and IP Multimedia Subsystem (IMS) include:

- Reduced Total Cost of Ownership;
- Improved network usage efficiency due to the flexible allocation of NFs;
- Higher service availability and resiliency provided to end users by dynamic network reconfiguration;
- Elasticity, in other words, the capacity to each NF can be dynamically modified according to actual network's load;
- Topology reconfiguration to optimise network performance and to support the introduction of new services.

2.6 Chapter Summary

In this chapter the 5G networks were presented. At the beginning of the chapter, it was presented some of the organizations that have a main role on the 5G standardization, such as ETSI and ATIS. Then, it was described the major performance targets that 5G networks must fulfil, like data rates, traffic capacity, spectrum efficiency, energy efficiency, mobility, connection density and latency.

After that, Network Slicing and Service-Based Architecture were defined as highlighted features for 5G networks. Network slicing is a feature that allows several network services to run simultaneously on the same infrastructure. The SBA, on the other hand, will bring some benefits to 5G, such as minimal impact of updating network services, direct connection between services, reuse and modulation of services, and opening of information management and control exposed by specific services.

After that, the three main use cases of 5G were listed, namely eMBB (high throughput), URLLC (low latency) and mMTC (large number of connected devices).

Subsequently, the important role of virtualisation in 5G networks was shown, which will allow the creation of independent virtual infrastructures for each tenant, as well as the creation of new services. virtualisation of 5G networks has been driven by technologies such as SDN and NFV. These two technologies have been described, presenting their base principles and high level architecture.

After that, the NFV, which is a technology standardised by ETSI, was explained in detail. The main objectives of this technology are service innovation through software-based deployment, greater operational efficiency, reduction of

energy consumption, opening and standardisation of network functions, flexibility to associate VNFs with hardware and greater capital efficiency. The basic principles of NFV are decoupling software from hardware, flexibility in the implementation of network functions and dynamic operation. The architecture of the NFV is divided into three major parts: the VNF, the NFVI and the NFV MANO.

Subsequently, the relationship between NFV and SDN for 5G was described, which consists in three steps: NFV focuses on creation and management of VNFs, then SDN helps organize VNFs into NFV network services and, with that, VNFs and network services become resources for construction and optimization of services for the client.

Next, some of the NFV modeling languages were described, such as TOSCA, YANG and HEAT.

Finally, some of the cases of use of NFV for 5G networks were presented, in particular Virtualisation of the Home Environment and Virtualisation of Mobile Core Network and IMS.

Chapter 3

Reference platforms for Service Lifecycle Management & Orchestration

This chapter presents relevant reference platforms for Service Lifecycle Management & Orchestration based on NFV MANO architecture. The first one is OSM where is described what is OSM and how is his architecture. In the second chapter, ONAP is described, where his architecture is focused and, with more detail, the Application Controller (APP-C) and his LCM capacity on VNFs.

3.1 Open Source MANO (OSM)

OSM is an open source platform developed by ETSI community aimed at developing a solution aligned with NFV MANO requirements [50]. It targets rapid installation in VNF vendor, system integrator and operator environments worldwide. OSM was proposed as the vision to make NFV easy to use and give choices to operators [53].

This platform has a modular and model-driven architecture (Figure 3.1) and adopted cloud-native design principals based on NFV MANO architecture. Service Orchestrator (SO) manages network services, working with Resource Orchestrator (RO) and Network To Virtualised Network Function Configuration (N2VC) module. This last module supports rapid progress on VNF configuration functionality. All those modules are managed by the monitoring module. Information Model and Northbound API (IM-NBI) support the growing demand for user differentiation options. This last module interacts with User Interface Applications (UI-Apps). [53]

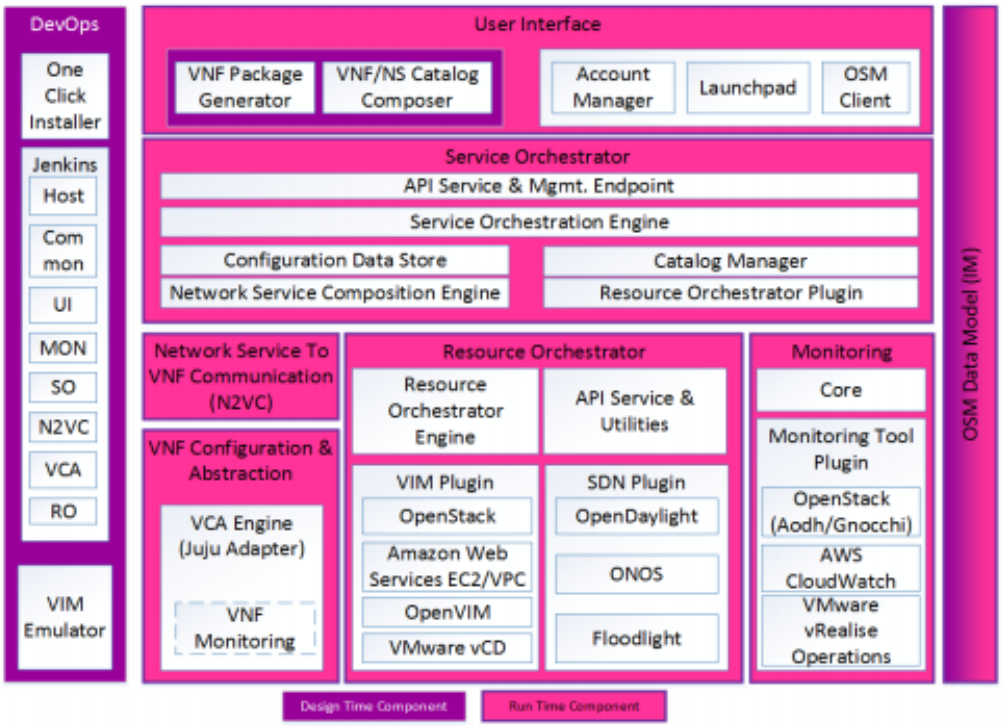


Figure 3.1: OSM architecture in modules [53]

For more detailed information about OSM see [50], [52], [51] and [53].

The modules of the OSM architecture are briefly described in the next sections.

3.1.1 DevOps

DevOps module is a Design-time module that enables the development and installation of OSM platform. Built over Jenkins, this components have the descriptors to configure all components to be installed in the platform [51].

3.1.2 User Interface

The User Interface is a module that have design and run-time tasks. Belonging to the design-time, this component has a VNF Package Generator and a VNF/NS Catalog Composer. These parts enable the service design to the platform user [51].

In the Run-time part, the User Interface has a Account Manager to manage the access to VIM environments, a graphic interface to use LCM operations and to retrieve detailed statistics and monitoring on VNFs and services [51].

3.1.3 Service Orchestrator

This Run-time SO component is responsible for all the aspects of service orchestration including LCM operations and service execution. It also acts as a OSM inventory and catalogue, holding all the services and VNF related descriptors defined in YANG modelling language [51].

3.1.4 Network Service to VNF Communication

The N2VC module is a Run-time module responsible for the plugin framework that connects the SO module to the VNF Configuration and Abstraction (VCA) layer [51].

3.1.5 VNF Configuration and Abstraction

This OSM layer that belongs to the Run-time module is responsible for enabling configuration, actions and notifications to VNFs and EMs. This module provides resources for GVNFM and SVNFM [51].

3.1.6 Resource Orchestrator

This Run-time module is responsible to interact with SO module for Management and Orchestration of NSs and VNFs. It is responsible for managing and orchestrating resource allocations across multiple geolocated VIMs and SDN controllers [51].

3.1.7 Monitoring

The OSM Monitoring Module (MON) is a Run-time module intended to be a tool for driving monitoring configuration update to external monitoring tools. MON will trigger events into SO. However, events will not be triggered directly by MON, but by external monitoring tools such as OpenStack Aodh, Amazon CloudWatch and other similar monitoring tools [51].

3.1.8 OSM Information Model

This Run-time module was created to be a point of authority on OSM data model. It shares VNFDs and NSDs between components in their original formats [51].

3.2 Open Network Automation Platform (ONAP)

ONAP is an open source Linux Foundation platform for NFV MANO. It provides a platform for real-time, policy-driven orchestration and automation of

physical and virtual network functions that enable software, network, IT and cloud providers and developers to automate new services and support complete LCM. ONAP has been considered a MANO++ platform because it covers the entire scope of ETSI MANO and beyond it [38].

ONAP architecture, shown in Figure 3.2, has two major systems: design time and run time, splitting design from operational roles. According to [38] these systems have the following responsibilities:

- Design time environment:
  - VNF onboarding/validation;
  - Network service/SDN service design;
  - Policy creation;
  - Workflow design;
  - Analytics application onboarding;
  - Data Collection, Analytics and Events (DCAE) workflow design;
  - Event monitoring.
- Run time environment:
  - Service orchestration
    - Service orchestration & LCM;
    - VNF controller (VNF orchestration & LCM);
    - Infrastructure controller (interface to VIM, SDN-C);
  - Monitoring and service assurance
    - Data collection, analytics and events;
    - Storage of all active and available inventory.

Additionally, ONAP includes an User Interface (UI), an Application Programming Interface (API) interface and ONAP Operations Manager (OOM). OOM is a project that is responsible for the deployment, instantiation and LCM operations of ONAP platform himself [45]. On northbound interface, ONAP talks to OSS, BSS, big data analytics and E-services applications. On southbound interface, ONAP interacts with VIM, NFVI and NFV cloud [38].

3.2.1 ONAP Components

ONAP platform is composed by several components, each one with a well-defined role on the ONAP architecture. Some of those components are related to Design Time environment, others are related to Run Time environment and even other components can be part of two environments. These components are briefly described in the following sections.

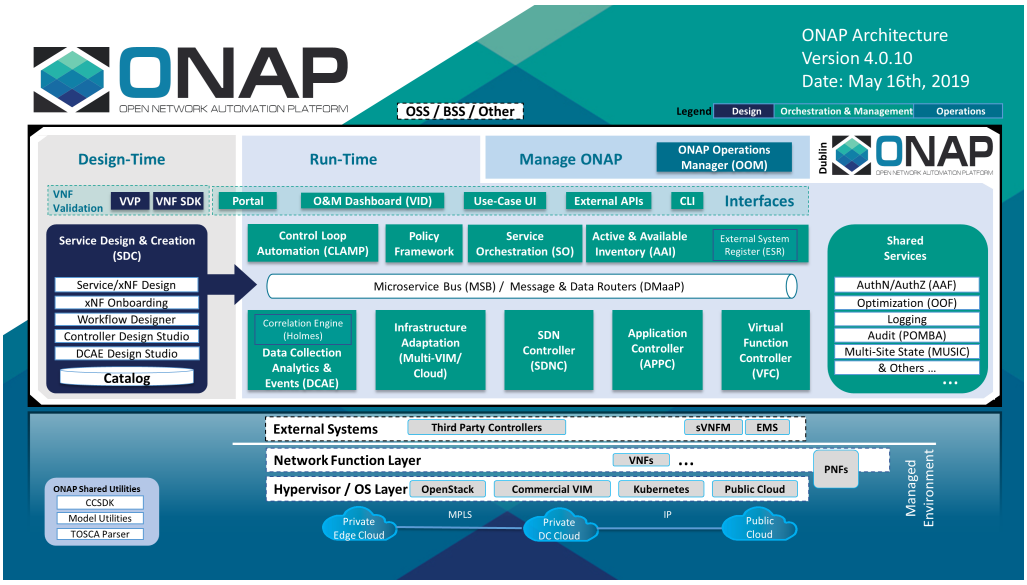


Figure 3.2: ONAP architecture [38]

3.2.1.1 Active and Available Inventory

Active and Available Inventory (AAI) component is the central inventory system of ONAP. It is where all data in the platform converge, including network, VNF, service and model information. This component takes updates in real-time from all the others components and works as a datacenter for those components to access to all the data needed to perform decisions. AAI makes part of Run Time environment and is accessed via REST APIs. [3]

3.2.1.2 Application Controller

APP-C is a Run-Time environment component, where his main function is to perform LCM operations on VNF as well as the components that execute these functions. This component makes the cloud infrastructure to be abstracted from Virtual Functions to make repeatable actions and enabling automation on LCM and configuration operations on VNFs. [43]

An in-depth description of APP-C component can be found in Section 4.3.

3.2.1.3 Application Authorization Framework

Application Authorization Framework (AAF) component's function is to support implementations for authentication and authorization in ONAP platform. This component consists in a set of Client Libraries and RESTful Services that support multiple authentication as well as authorization protocols. It provides

authentication, authorization and security to ONAP components, freeing other components and services from those tasks. [1]

3.2.1.4 Data Collection, Analytics and Events

DCAE is the data monitoring and analysis component of ONAP. It is a system that collects the data from ONAP platform and network entities, apply analytics and trigger events based on the analysis. DCAE can be part of the close control loop for managing network services. [4]

3.2.1.5 Data Management as a Platform

Data Management as a Platform Data Bus Controller (DMaaP) is a platform to transport and process of data between ONAP components, from any source to any target. This component optimizes data exchange, filtering, compressing and reducing data. It also transports and processes the data efficiently. [5]

3.2.1.6 Microservices Bus

Microservices Bus (MSB) is a component that provides reliable, resilient and scalable communications between the platform components. This includes service discovery and registration, external and internal API gateway and more features including component communications. MSB have a pluggable architecture, that means plugins can be added to MSB to provide more functionalities from external sources. It is an independent platform, integrated with Kubernetes to provide transparent service registration for ONAP microservices. [7]

3.2.1.7 Multi-VIM/Multi-Cloud

MultiCloud (or Multi-VIM) is an Run-Time ONAP component whose scope is to mediate all interactions between ONAP and all underlying VIM or Clouds. [44] This component enables ONAP to deploy and run actions in VIM or Cloud infrastructures using a specific plugin according to the cloud infrastructure used (OpenStack, Azure, etc.). It also enables to decouple the evolution of ONAP platform from the evolution of the underlying environments, minimizing the impact of cloud infrastructures evolution on ONAP's platform. [27]

A high-level architecture and workflow of Multicloud are outlined in Figure 3.3.

3.2.1.8 ONAP Operations Manager

OOM is responsible for LCM of the ONAP platform. This manager is not responsible for managing services or network functions like VNFs, but uses Kuber-

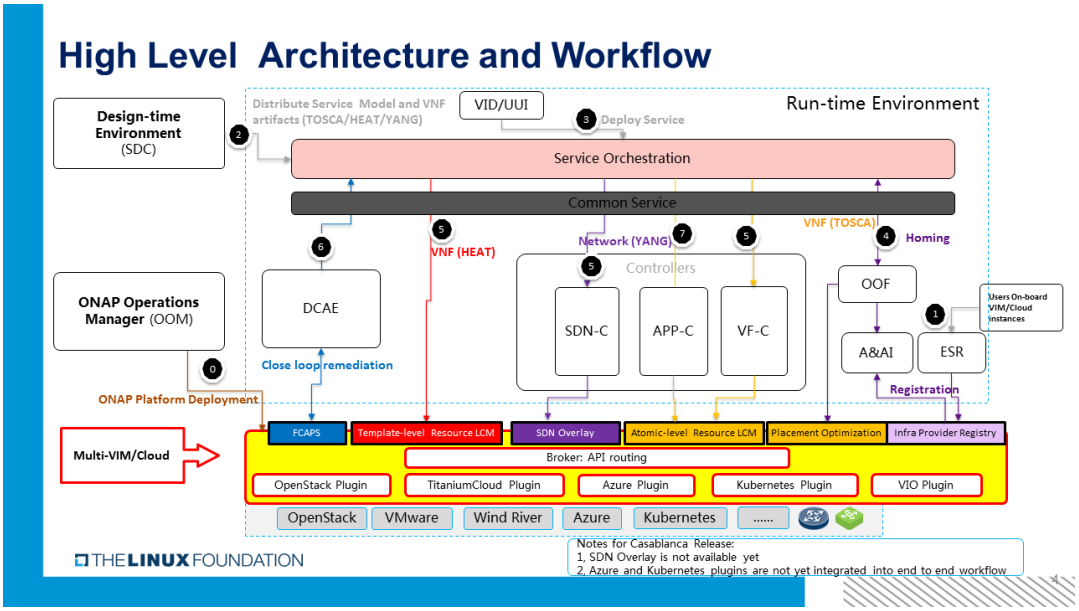


Figure 3.3: Multi-VIM/MultiCloud high-level architecture and workflow [44]

netes container management system to manage Docker containers that composes ONAP. In resume, OOM provides seven major capabilities [45]:

- **Deploy** - component deployment and dependency management;
- **Configure** - configuration across all ONAP components;
- **Monitor** - real-time health monitoring;
- **Heal** - recreation of failed ONAP containers;
- **Scale** - enable ONAP services scaling;
- **Upgrade** - reconfigure or change containers with a minimal impact on the platform;
- **Delete** - Clean individual containers or an entire deployment.

3.2.1.9 Policy

Policy component is a subsystem that maintains, distributes and operates a set of rules defined by the operator for ONAP control, orchestration and management of functions. Policies can support ONAP's platform, products, services, operation automation and security. These sets of rules can be modified, configured or even designed by users from ONAP portal. [9]

3.2.1.10 ONAP Portal

This component is a web based platform that centralize different ONAP applications for different users. The target of this component is to allow decentralized applications running in their own components, providing common management services and connectivity in one single portal. [10]

3.2.1.11 Service Design and Creation

Service Design & Creation (SDC) component is the ONAP visual modeling and design tool. SDC belong to both design and run time, creating internal metadata that describes all assets used by all ONAP components. SDC contains a database catalogue containing TOSCA blueprints with all information and configuration data of VNFs and Network Services. SDC does VNFs and Network Services onboarding, distributing to all other components information about the functions to be onboarded. [11]

3.2.1.12 Service Orchestrator

SO is a component that arranges, sequences and implements tasks based on rules and policies (sent by SDC and Policy components) to create, modify or remove logical and physical resources in the managed environment. This component orchestrate those resources in a high level, leaving the remaining and low level orchestration to the underlying controllers such as APP-C and Virtual Function Controller (VF-C). The SO runs autonomously within ONAP and use process workflows to execute all the tasks through service models. SO interacts with other platforms via defined APIs. [6]

3.2.1.13 Software Defined Network Controller

SDN-C is a component that configures and maintains healthy VNFs and network services from the three lower levels of Open System Interconnection (OSI) model. It manages inter-cloud connectivity and with other external SDN-Cs for connectivity inside a cloud region. SDN-C component. This component is built over OpenDayLight, having a proper API built in this technology. SDN-C share this platform with APP-C component. [38]

3.2.1.14 Virtual Function Controller

VF-C component is an integration of ETSI NFV MANO architecture and reference model on ONAP. It implements LCM and Fault-management, Configuration, Accounting, Performance and Security (FCAPS) of VNFs and NSs, following the ETSI NFV specifications. This component works generally with

generic VNFs, with multiple VNFMs, VNFs and VIMs. VF-C can also support vendor VNFMs. [8]

3.2.1.15 Virtual Infrastructure Deployment

Virtual Infrastructure Deployment (VID) component has the goal to execute the infrastructure deployment, instantiation and change-management operation. VID also can choose the target instantiation environment by checking their availability, customize VNFs and services to fit a instantiation, and can even operate and execute basic LCM operations through workflow if a instantiation is failed. [13]

3.3 Chapter Summary

In this chapter two open source platforms for the management and orchestration of virtualised networks were presented.

The first platform presented was OSM which is an open source platform developed by the ETSI community aligned with the NFV MANO requirements. The objective of this platform is the rapid installation in VNF vendors, system integrators and worldwide operator environments. This platform has a modular, model-driven architecture and adopts a cloud-native design based on the NFV MANO architecture. Its architecture is further divided into design-time and run-time models, depending on the phase in which the component is used.

The second platform that has been described in this chapter is ONAP, which is an open-source platform for NFV MANO developed by the Linux Foundation community. It provides a platform for real-time, policy-driven orchestration and automation of physical and virtual network functions.

It is considered a MANO++ platform because it includes features that go beyond the scope of ETSI MANO. Like OSM, its architecture has run-time and design-time environments, depending on the role each component plays in the execution of the platform.

Chapter 4

Implemented Platform

From a general perspective, this chapter will focus on the description of the implemented infrastructure, including the platform setup and available LCM operations characterization. The first subchapter is an overview description of the implemented infrastructure. In the followed subchapter, the orchestration platform used in the implementation is specified. In the third subchapter, LCM tools available in APP-C component are identified and explained.

4.1 Platform overview

5G networks are developed on top of NFV and SDN technology, making LCM operations on network services and functions a significant role in the automation of infrastructure orchestration.

The prime goal for this solution is to perform different types of LCM operations on VNFs. To implement this solution, it was adopted a customized ONAP platform to orchestrate the wanted services with the minimal required components installed. The chosen component to execute LCM operations is APP-C through his protocols because is more complete and stable than VF-C component in Casablanca version.

4.2 Orchestration platform

As said before, the platform adopted for the implemented solution was ONAP. It was adopted because it is a feature-packed platform for network automation, providing many capabilities besides orchestration. Because of its modularity and flexible deployability, the first step was to identify only ONAP components that would be required for enabling service orchestration (instantiation and LCM). After the analysis, the components identified as required were SO, AAI, SDN-C and APP-C.

SO component maps to and End-to-End Orchestrator. It is responsible to implement orchestration logic, providing the highest level of service orchestration in the platform. AAI works as an inventory for the platform. It's responsible for storing all information, including network, service and platform information. SDN-C provides instantiated VNFs and services with configuration automation and provisioning of network resources. At last, APP-C performs LCM operations on VNFs, VMs and Virtualised Network Function Component (VNFC)s, as stated in Chapter 3, performing on Layers 4 (Transport) to 7 (Application) of OSI model.

During platform analysis and deployment, some functional and platform dependencies were identified. Those components are:

- **VID:** This component enables users to instantiate infrastructure services and their components designed in SDC. VID can be used to search on SDC catalogue for services models or services instances and modify them;
- **Service Design & Creation (SDC):** This component is responsible for creating internal metadata describing resource and service resources to be distributed and used by other ONAP components, in design time and run time. Also, it works as a metadata catalogue;
- **MSB:** This component works like an information tunnel and distributor of information between all ONAP microservices;
- **AAF:** This component manages authorization and authentication to various ONAP components. It is a direct requirement of AAI component because all incoming requests to its REST APIs must pass through AAF before AAI;
- **External System Register (ESR):** In this component, users can register, update and delete external systems using its graphic interface;
- **Multi Cloud/Virtual Infrastructure Manager (Multi Cloud/VIM):** This component decouples ONAP platform from the underlying cloud infrastructure, providing mediation between them, supporting multiple clouds. It is used by both SO and LCM components (APP-C and VF-C);
- **Data Movement as a Platform (DMaaP):** responsible for implementing an effective data movement tool implementing a Publisher-Subscriber logic.

4.2.1 ONAP Setup description

To setup ONAP deployment, it was used OOM. This component is responsible for LCM of ONAP platform. OOM relies on Rancher and Kubernetes infras-

tructure. Rancher is a platform responsible to manage clusters, in this case, Kubernetes clusters, and maintaining ONAP repository through Helm. Kubernetes is used to ONAPs microservice-based management running in Docker containers.

It was established a testbed to deploy and configure ONAP platform. It was created four VMs, three of them dedicated to OOM and one for an Openstack Pike platform. OOM includes one VM for Rancher and two for Kubernetes cluster. All the services and VNF resources instantiated and orchestrated are held by OpenStack infrastructure.

Rancher infrastructure requires low resources as it only handles Kubernetes environment and ONAP repository. In opposite, Kubernetes cluster requires significant memory and processing resources (Virtual Computing Processor Unit (vCPU)). Kubernetes performs LCM on ONAP pods that are more than 100 for this case. Openstack infrastructure does not require high amounts of resources because it's used to validate the integration between ONAP and VIM or cloud, and to perform simple LCM operations on simple VMs, VNFs or VNFCs.

In the Table 4.1, VMs and resources used for ONAP deployment are described.

The setup process begins adding Kubernetes VMs as host machines to Rancher environment. This process comprises a Rancher agent installation in those VMs and consequently registration in the Rancher server. After the addition of the Kubernetes VMs to the Rancher environment, ONAP deployment carried out using OOM. This process consists of configuring Helm charts. Helm is software that applies a packet management concept naming packages as charts and applying them to Kubernetes resources. Those charts are organized in a collection of files inside a directory, containing a specific structure on a specific service description. OOM is an aggregation of those charts describing ONAP services.

After those charts being cloned into Kubernetes cluster, OOM is used to choose the desired ONAP components for installation and creation of the corresponding local Helm repository. The setup is concluded running a Helm command for instantiating all desired components.

Table 4.1: List of LCM operations and Protocols

	Virtual Machines			
Resources	**Rancher**	**K8's - 0**	**K8's - 1**	**Openstack**
vCPU	4	8	4	2
RAM	8 GB	64 GB	16 GB	8 GB
Disk Memory	80 GB	300 GB	30 GB	30 GB
OS	Ubuntu Server 16.04	Ubuntu Server 16.04	Ubuntu Server 16.04	Ubuntu Server 16.04

4.3 Lifecycle Management tools

To perform LCM operations on VNFs, VMs and VNFCs, APP-C component uses some protocols to handle those components, using adapters for each one of the protocols. The following subchapters describe APP-C component with detail and the protocols used by this component.

APP-C is a component designed to perform LCM of VNF's on layers 4 (Transport) to 7 (Application) of OSI model and, in addition, act as a SVNFM [38]. This component is built on top of OpenDayLight platform.

APP-C component provides a comprehensive set of controller actions such as Configure, Modify Configuration, Start, Stop, Migrate, Restart and Rebuild. It supports a set of standard VNF interfaces like NETCONF, Chef and Ansible, and is designed to be self-service using a model-driven architecture that provides a layer of abstraction making APP-C completely service-, VNF-, and site-agnostic [27].

Figure 4.1 schematises the APP-C high level architecture.

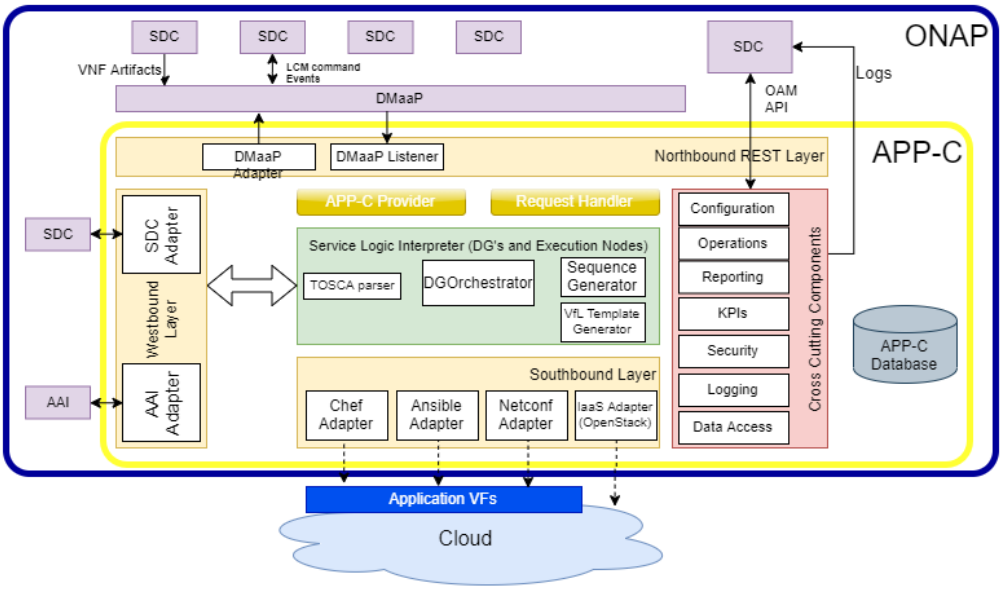


Figure 4.1: APP-C Architecture [43]

On northbound layer, APP-C interacts with APP-C clients like SO and SDC, exposing his YANG-based API via REST [43]. APP-C provides a comprehensive set of controller actions such as Configure, Modify Configuration, Start, Stop, Migrate, Restart, Rebuild, and so on. It supports various standard VNF Interfaces such as Ansible, Chef and NETCONF.

APP-C can apply LCM operations to the following three components [43]:

- Virtual Network Function (VNF);
- VNFC;
- Virtual Machine (VM).

APP-C supports two types of LCM requests. The first request type is LCM over REST, that are sent to APP-C through Hypertext Transference Protocol (HTTP) POST requests. The second type is LCM over Message Bus (DMaaP). APP-C receives LCM command requests as messages from DMaaP. Those requests are sent by APP-C's clients located on other ONAP components like SO and SDC [43].

To perform a LCM request, a Directed Graph (DG) related to the specific operation is chosen. A DG is a diagram that works like a workflow to APP-C. APP-C component has a DG for each LCM operation and has a DG Builder to configure DG or develop new ones [43].

To perform LCM operations on VMs, VNFs and VNFCs, APP-C have the following protocols to handle those components [46]:

- NETCONF (Extensible Markup Language (XML) Restconf);
- Ansible;
- Chef;
- REST;
- OpenStack (VM Level).

In the exception of Openstack and REST protocols, LCM protocols need Reference Data files to perform each one of LCM operations for each VNF. These a Reference Data need to be defined through APP-C Controller Design Tool (CDT) by VNF vendors. These Reference Data are composed by three types of files [46]:

- Template;
- Parameter definition file;
- Parameter name-value pair file.

A Template is required for Ansible, Chef and NETCONF protocols. Parameter Definition and Parameter Name-Value Pair artifacts are typically used with Configure and ConfigModify templates and are optional for templates of other

actions. OpenStack and REST protocols do not use a template or parameter definitions or name-value pairs [46].

The Table 4.2 shows all LCM operations available and protocols that can run each one of them for ONAP's Beijing release [46].

Table 4.2: List of LCM operations and Protocols supported by APP-C [46]

Actions \ Protocols	Netconf	Ansible	Chef	REST	OpenStack
ActionStatus	NO	NO	NO	NO	NO
AttachVolume	NO	NO	NO	NO	YES
Audit	YES	YES	YES	YES	NO
CheckLock	NO	NO	NO	NO	NO
Configure	YES	YES	YES	NO	NO
ConfigModify	YES	YES	YES	NO	NO
ConfigBackup	NO	YES	YES	NO	NO
ConfigRestore	NO	YES	YES	NO	NO
ConfigScaleOut	YES	YES	YES	NO	NO
DetachVolume	NO	NO	NO	NO	YES
DistributeTraffic	NO	YES	YES	NO	NO
Evacuate	NO	NO	NO	NO	YES
HealthCheck	NO	YES	YES	YES	NO
Lock	NO	NO	NO	NO	NO
Migrate	NO	NO	NO	NO	YES
QuiesceTraffic	NO	YES	YES	NO	NO
Rebuild	NO	NO	NO	NO	YES
Restart	NO	NO	NO	NO	YES
ResumeTraffic	NO	YES	YES	NO	NO
Snapshot	NO	NO	NO	NO	YES
Start	NO	NO	NO	NO	YES
Start Application	NO	YES	YES	NO	NO
Stop	NO	NO	NO	NO	YES
Stop Application	NO	YES	YES	NO	NO
Sync	YES	YES	YES	YES	NO
Unlock	NO	NO	NO	NO	NO
UpgradeBackout	NO	YES	YES	NO	NO
UpgradeBackup	NO	YES	YES	NO	NO
UpgradePostCheck	NO	YES	YES	NO	NO
UpgradePreCheck	NO	YES	YES	NO	NO
UpgradeSoftware	NO	YES	YES	NO	NO

For more detailed information about APP-C and APP-C CDT see [43] and [46].

4.3.1 Netconf

APP-C component have an adapter for NETCONF protocol, responsible for configuration tasks using XML format for VNFs supporting NETCONF API. Secure Shell (SSH) adapter can additionally use NETCONF protocol through XML, used to load configurations and retrieve the running configuration. [43]

According to [28], NETCONF protocol defines a simple mechanism through

which a network device can be managed. Configuration data information can be retrieved and new configuration data can be uploaded and manipulated. The protocol allows the device to expose a full API, that can be used to applications sends and receives full and partial configuration data sets. The protocol uses a Remote Procedure Call (RPC) paradigm encoded in XML.

For more information about this protocol, read [28] and all related Request for Comments (RFC) from Internet Engineering Task Force (IETF).

4.3.2 Ansible

Ansible adapter includes a client for an external Ansible server, which connects to VNF Northbound APIs. This adaptor enables APP-C to operate playbooks to perform LCM operations over VNFs connected to Ansible server. [43]

According to [55], Ansible is an open source Information Technology (IT) management, deployment and orchestration tool. This tool comprises a large variety of automation challenges, trying to conceal this variety to a user friendly tool, easy and simple to use.

One of the most considerable differences between this tool and similar ones is the architecture. Ansible is a agentless tool that runs in a 'push' model. In other words, no software is installed or required on remote machines to manage them. By default, Ansible uses SSH (Linux and UNIX) or Windows Remote Manager (WinRM) (Windows) to manage remote machines. [55]

For more information about this tool, see [55] and online Ansible documentation.

4.3.3 Chef

Similar to Ansible adapter, Chef Adapter incorporates a client for an external Chef server that connects to VNF Northbound APIs. With this adapter, APP-C can operate cookbooks (Chef scripts) to perform LCM operations over VNFs. [43]

According to [25], Chef is an automation platform that transforms infrastructure into code. Chef automates how infrastructure is configured, deployed and managed across the managed network.

The most significant difference between Chef and Ansible is the architecture complexity. The architecture is defined as showed in Figure 4.2.

The chef architecture is divided in three components [25]:

- **Chef Workstation:** Location where users interact with Chef. With the workstation, users can create and test cookbooks using available Chef tools. The user can also interact with Chef Server;

Figure 4.2: Chef Architecture [25]

- **Chef Client Nodes:** Machines that are managed by Chef. To connect these nodes, a Chef agent is installed on each node;
- **Chef Server:** This component is a hub for configuration data. In other words, is where cookbooks, policies and metadata are stored. The managed machines, through Chef Client, asks the Server for configuration details.

For more information about Chef protocol, visit the online documentation of Chef.

4.3.4 Restconf

APP-C component have a Restconf adapter, responsible for configuring tasks, using JavaScript Object Notation (JSON) format messages, for VNFs supporting Restconf API. [43]

According to [23], Restconf is a protocol based on HTTP for configuring data defined YANG version 1 and 1.1 templates, using datastore concepts of NETCONF. Restconf uses HTTP methods to provide Create, Read, Update, Delete (CRUD) operations on a conceptual datastore containing YANG-defined data.

For more information about this protocol, read [23] and all related RFC from IETF.

4.3.5 OpenStack

APP-C uses a Infrasctructure as a Service (IaaS) adapter to connect with OpenStack controllers to perform basic LCM operations on VMs such Restart, Migrate, Start and Stop. This adapter acts as a Directed Graph plugin, while those Directed Graphs call services exposed by the adapter. [25]

4.4 Chapter Summary

In this chapter, the platform implemented for testing and evaluation of LCM operations in network services was characterised.

First, it was defined that a customized installation of the ONAP platform should be used so that it would be possible to perform LCM operations with a smaller amount of resources.

This was followed by the identification of the components that would allow the LCM operations to be carried out, namely the Operating System (OS), the AAI, the SDN-C and the APP-C. After the analysis and implementation of the platform, dependencies were identified, such as VID, SDC, MSB, AAF, ESR, Multi-cloud and DMaaP.

After identifying all the necessary components, the customised platform was deployed using four virtual machines. This deployment included the use of platforms such as Kubernetes, Rancher and Helm, to support the installation and maintenance of the platform, and OpenStack to host the VNFs that would be instantiated.

Finally, the architecture of the APP-C component for LCM operations was explained. This component, developed over the OpenDayLight platform, allows to perform LCM operations over VNFs, NF and VMs, using some of the interfaces that this component provides, such as OpenStack, Ansible, Chef, Netconf and Restconf.

Chapter 5

Infrastructure Testing and Validation

In the first part of this chapter, the objectives to test and validate LCM operations on the installed platform are described. In the second subchapter, the VNF used to perform LCM operations is briefly detailed. In the next subchapters, the realized tests and results obtained to validate some of the LCM operations available on the platform are shown.

5.1 Validation and Evaluation Targets

To validate the platform and the LCM operations wanted, it was defined three major objectives to fulfil:

- **Onboarding of a VNF in ONAP platform:** The first objetive to complete is the onboarding and instantiation of a VNF inONAP platform. This objective must be accomplished in order to execute LCM operations. VNF's description and characteristics are described in the next subchapter;
- **Execute LCM operations with OpenStack:** LCM operations executed through OpenStack are the most simple and stable operations on ONAP platform. The goal is to execute some of the operations available through this protocol;
- **Execute LCM operations with Ansible:** Ansible is a protocol that enables easy management and configuration on VNFs through APP-C

The methodology used to meet the defined test goals will consist in instantiate a network service, and then run the lifecycle operations from OpenStack and Ansible. The metrics will be taken from the APP-C logs during the test execution.

These tests will consist of a repetition of the operations, and the metrics collected will be used to obtain a average values. Finally, according to the results obtained, an evaluation of the performance of the operations will be made.

5.2 VNF description, Onboarding and Instantiation

To test and validate the platform and the LCM operations, it is needed to instantiate a VNF and perform the LCM operations. Due to a scarce of resources, the VNF in use is a simple Virtual Firewall (vFW) based on [12]. For testing, it was used tiny flavors, in other words, virtual machines with low resources. To do that change, the OS was replaced by CirrOS, a very simple OS for testing purposes only. The HEAT templates used to instantiate and describe all the details of the vFW VNF are in Appendix A.

To onboard and instantiate the VNF, it was followed the process indicated in Figure 5.1.

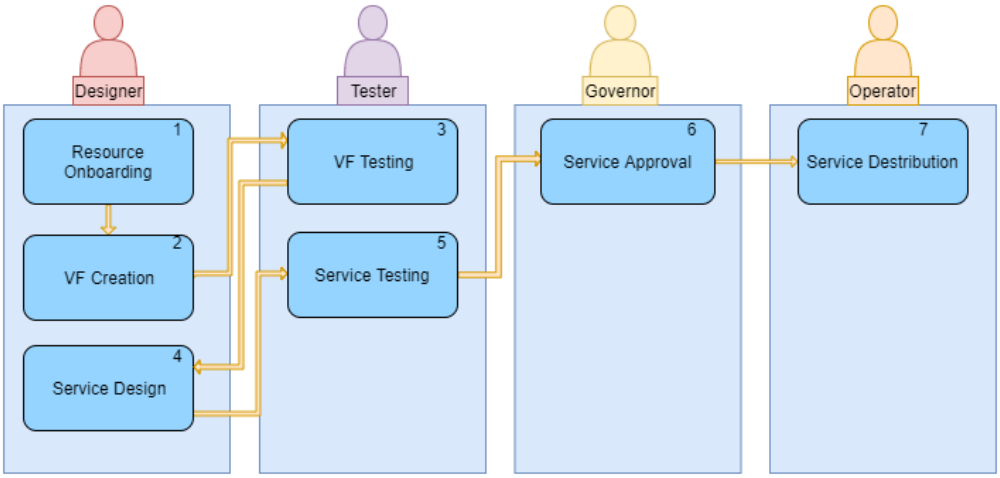


Figure 5.1: Service Onboard process in ONAP [21]

The Resource Onboarding, executed by a Designer user, onboard VNFs and Virtual Function (VF), the corresponding models and artifacts. This includes License Model creation and onboarding and Vendor Software Products (VSP), which correspond to a VNF or VF. [21]

VF Creation uses VSPs as building blocks, and then the VF is tested and validated and adds it to the VF catalogue. [21]

At the Service Design phase, the service model and artifacts are configured and onboarded using TOSCA packages. The Service Testing performed by a Tester is responsible to test and certify the service. [21]

In the end, the Service Distribution phase consists of two steps, the approval of the distribution of the service and the trigger of the distribution. [21]

5.3 OpenStack LCM Operations

The first part of testing is to execute the OpenStack LCM on the onboarded vFW. This test phase is divided into two parts. The first part is the execution of a series of three LCM operations. These operations are Start, Restart and Stop. The second part is a series of a Rebuild operation. The tests and the results of those tests are described in the next subchapters.

5.3.1 Running Tests on Start, Restart and Stop Operations

To execute this test, it was needed to have an onboarded VNF stopped on the OpenStack environment connected to ONAP. After the platform is ready to execute LCM operations, it's constructed three API calls corresponding for the three operations to execute: Start, Restart and Stop.

This calls are APP-C API calls and run in software named Postman. This software enables us to do API calls remotely in several programming languages. It also enables to run a collection with some calls and running repeatedly in iterations. This feature is used to this test, running a collection with Start, Restart and Stop operations, in that exact order, with ten iterations with a delay of one minute between each one.

To get results from this test, it was used the log system of APP-C to take the important parts of the operation, such as queries to inventory, calls to OpenStack API and duration of the LCM operation.

In the next subchapters, the results and analyses made for each operation are presented.

5.3.1.1 Start Operation Results

To better understand the Start operation phases and results, it was described in a diagram the messages and phases of the operation, as seen in Figure 5.2. This operation was repeated ten times during the tests.

Analyzing the diagram, the operation starts with the reception of a message from Postman and then starts a query to ONAP inventory. Then, the information requested to AAI component is returned to APP-C, sending back a success message to Postman. After this phase, APP-C component send a request to OpenStack through OpenStack adapter to execute the operation. While executing the operation on vFW, APP-C periodically sends a status request to validate

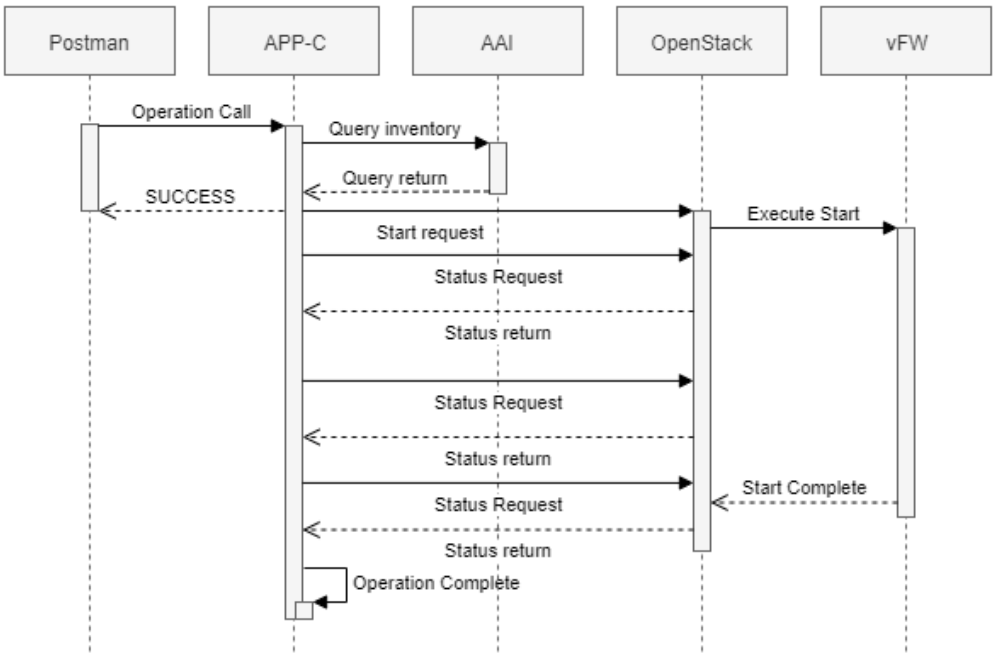


Figure 5.2: Start Operation Diagram

when the operation is concluded or not. Finally, when the operation is concluded, the APP-C component throws an operation complete message.

The results from the test to Start operation were stored and then calculated the average time, median and standard variation for each identified phase, as shown in Table 5.1.

Table 5.1: Test results from Start operation (ms)

	Average	**Standard Deviation**	**Median**
Input received → AAI (query finish)	1101	90	1100
Input received → Operation Start	1425	127	1405
AAI (query finish) → Complete	33131	659	33114
Operation Start → Complete	32807	613	32833
Input Received → Complete	34232	669	34142
AAI Query Time by log system	927.2	939.5	90.936

From this table, it is possible to make a statistical and graphical analysis building a Box and Whiskers plot. This plot is presented with two parts: the box and the whiskers.

The first part corresponds to a box, where is represented the area between the first and third quartile of the measured values. The first quartile represents the mean value between the smallest value measured and the median of the data set. In other words, represent that twenty-five percent of the retrieved data are under

the quartile value. The third quartile represents the opposite, which means that twenty-five percent of the retrieved data is above the quartile value. The second quartile corresponds to the median value, represented in the box as a dividing line.

The second part of the plot is the whiskers. Those whiskers are the line that defines the range between the first and third quartiles to the correspondent minimum and maximum measured values.

The Box and Whiskers graphic obtained is illustrated in Figure 5.3.

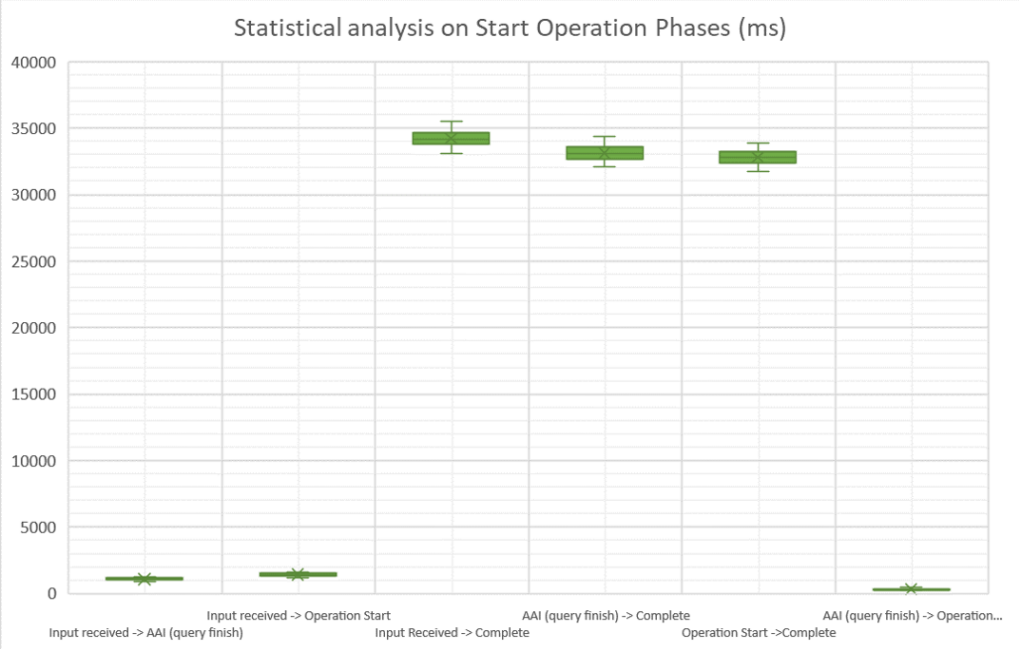


Figure 5.3: Box and Whiskers plot for Statistical analysis on Start Operation Phases

From the statistical analysis on Start operation phases, it is concluded that the series of tests made for the Start operation was sufficiently accurate to get a low of variation between values measured. Also, it can be concluded that operation is stable because of the low variation of times in each phase.

The chart on Figure 5.4 is a cumulative graphic from the time spent on each phase. This graphic doesn't include phases composed of other phases. It can express the influence of each phase through the time bars, however, it can analyze the slope between phases through the cumulative line. In this graphic, the slope on the first two phases isn't high. The ideal case is a zero slope from APP-C phases, but a zero slope situation isn't a realistic case. The slope between the operation start and the operation complete is higher, however, this phase is executed by OpenStack platform that is external to ONAP platform. It is strongly

possible that this phase included a failed call to AAI, which is described in the following sections, although was not detected in the metrics service of APP-C for that operation.

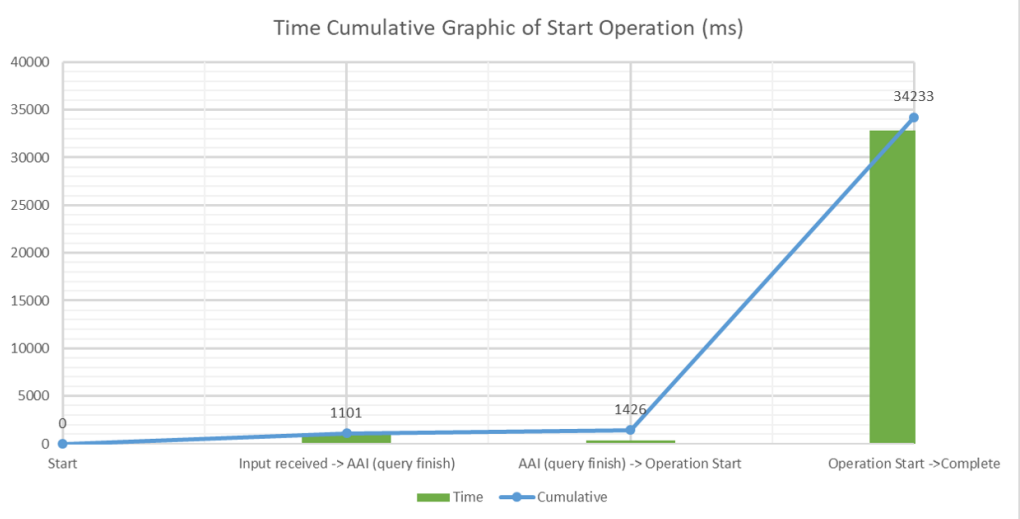


Figure 5.4: Time cumulative graphic of Stop operation

5.3.1.2 Restart Operation Results

To better understand the Restart operation phases and results, the messages and phases of the operation were presented in a diagram, see Figure 5.5. The restart operation was repeated ten times during the tests.

The execution of this operation is similar to the previous operation. However, in the final phase of Restart operation, APP-C throws a ”Server Status RUNNING” message and then have a five seconds delay before throwing the operation complete status. This occurs because APP-C try to update the VNF information on AAI, but the update failed because DMaaP can’t send the update to AAI.

The results from the test to Restart operation were stored and then calculated the average time, median and standard variation for each identified phase, as shown in Table 5.2.

Similar to the previous operation, analyzing the Box and Whiskers plot obtained through the series of tests on Restart operation (Figure 5.6), the platform was defined as stable because of the measured times were accurate on each phase. Because of this stable platform, the number of time measurements was sufficient to guarantee a low margin of variation between the values.

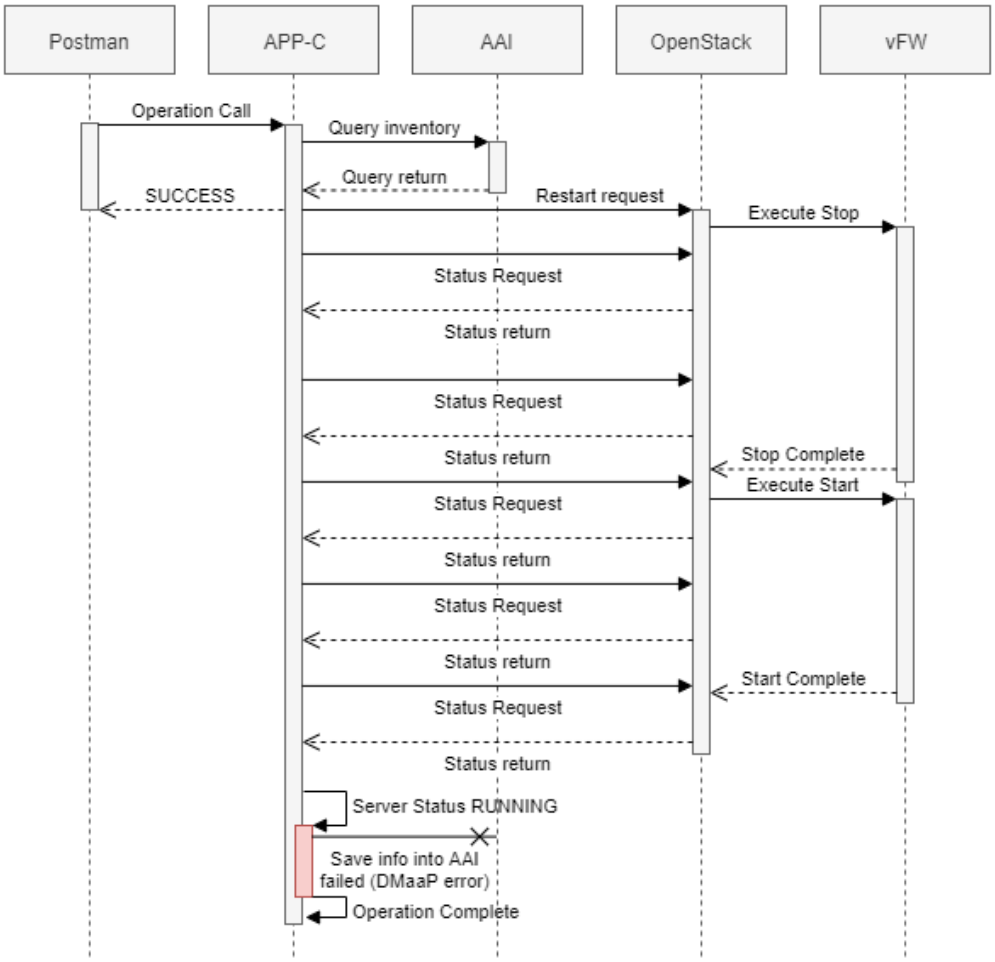


Figure 5.5: Restart Operation Diagram

Table 5.2: Test result from Restart operation

	Average	Standard Deviation	Median
Input received → AAI (query finish)	1116	100	1109
Input received → Operation Start	1431	113	1415
AAI (query finish) → Complete	51139	780	51233
Operation Start → Complete	50825	796	50842
Operation Start → Server Status Running	45494	770	45537
Server Status Running → Complete	5331	64	5322
Input Received → Complete	52256	822	52322
AAI Query Time by log system	928.9	932.5	76.855

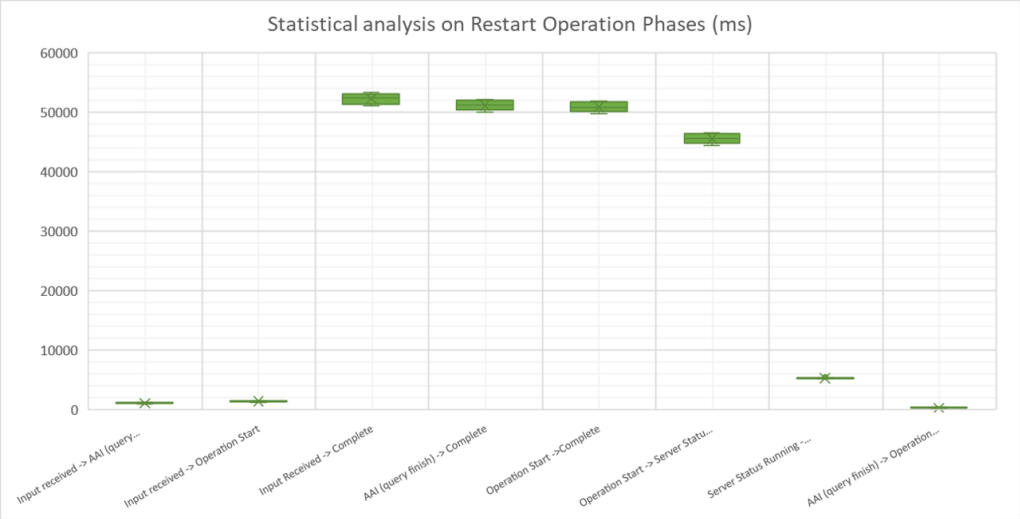


Figure 5.6: Box and Whiskers plot for Statistical analysis on Restart Operation Phases

The time cumulative graphic of Restart operation (Figure 5.7) is almost similar to the previous operation, However, it was detected the delay referred before. The slope in the last phase is the responsibility of the APP-C infrastructure that tries to update the AAI inventory through DMaaP unsuccessfully. Ideally, this delay shouldn't exist, throwing the complete status right after the running status. In other words, the last phase influence, represented by the last line, should have a slope of near-zero, which in this case doesn't occur.

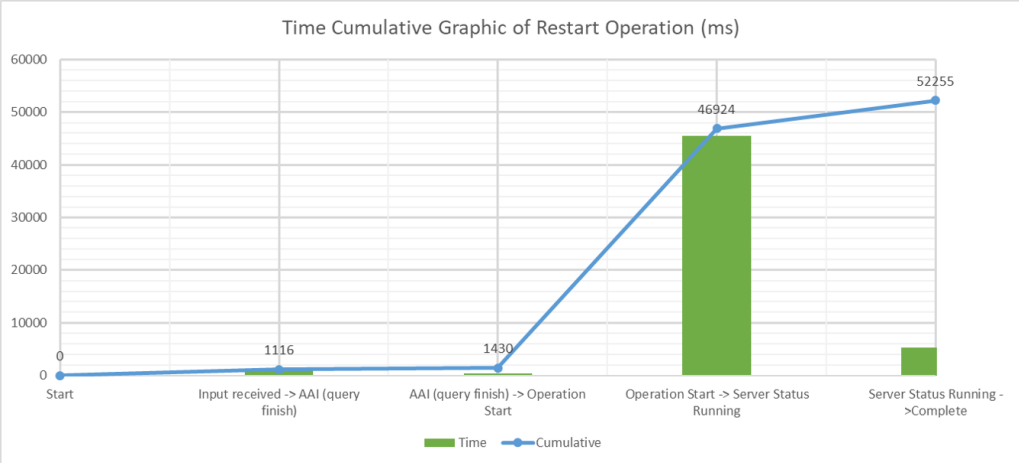


Figure 5.7: Time cumulative graphic of Restart operation

5.3.1.3 Stop Operation Results

To understand the Stop operation phases and results, it was described in a diagram, as the previous operations, the messages and phases of the operation, as seen in Figure 5.8. This operation was repeated ten times during the tests.

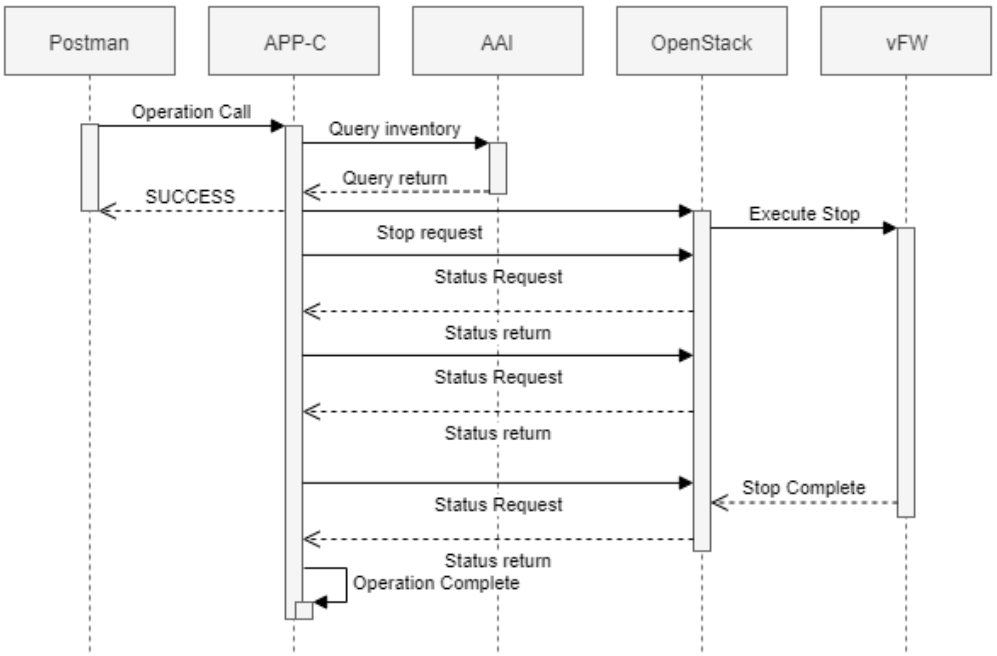


Figure 5.8: Stop Operation Diagram

This operation is similar to the Start operation with the same phases with the same functionality. The only phase that differs from Start operation e the Stop request phase.

The results from the test to Stop operation was stored and then calculated the average time, median and standard variation for each identified phase, as shown in Table 5.3.

Table 5.3: Test result from Stop operation (ms)

	Average	**Standard Deviation**	**Median**
Input received → AAI (query finish)	1126	155	1080
Input received → Operation Start	1472	179	1433
AAI (query finish) → Complete	33583	1164	33228
Operation Start → Complete	33237	1100	32917
Input Received → Complete	34709	1170	34383
AAI Query Time by log system	958.1	922	155.836

As the previous operations, the statistical analysis of the data measured in Stop operation on 5.9 evidences a low variation of the results and the stability of the Stop operation through the test.

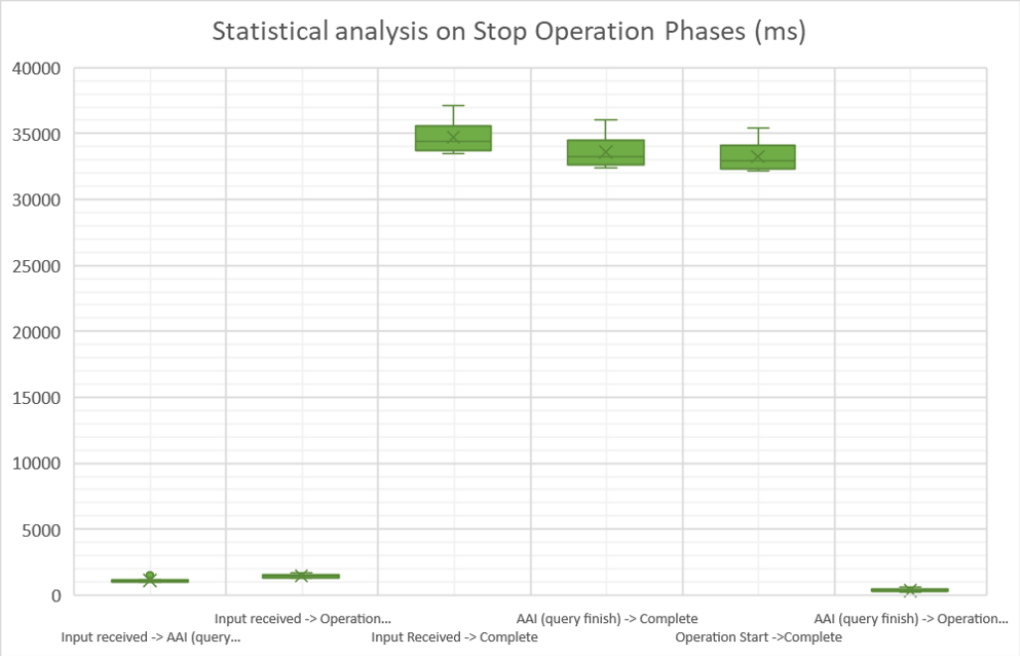


Figure 5.9: Box and Whiskers plot for Statistical analysis on Stop Operation Phases

The conclusions taken with the analysis of the time cumulative graphic for Stop operation (Figure 5.10) are the same as the conclusions taken to the Start operation graphic. This occurs because the operation phases are similar to the Start operation. Even the last phase, corresponding to the Stop operation itself, has a similar slope than the last Start operation phase.

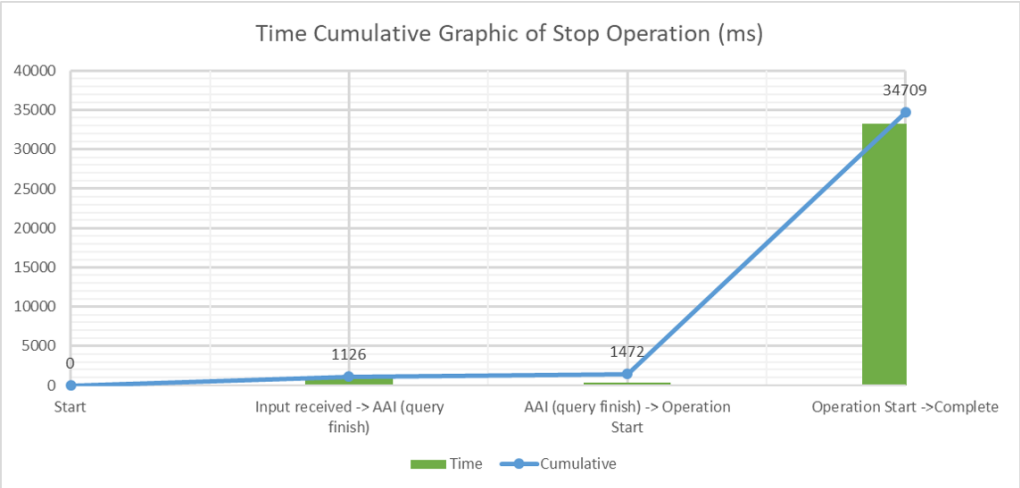


Figure 5.10: Time cumulative graphic of Stop operation

Table 5.4: Comparison between Restart operation and Stop + Start operation (ms)

	Time
Start Operation	34232
AAI (query finish) ->Complete (Stop Operation)	33237
Start + Stop Operation	67469
Restart Operation	52256
Difference	**15213**

5.3.2 Restart Operation versus Stop and Start Operations

The LCM restart operation has the same purpose as the Stop and Start operations together. The objective of this test is to compare if, in the time domain, the Restart operation is equal to Stop and Start.

To compare those operations, it was added the total time of Start operation to the phase after the AAI query to the complete state of the Stop operation. The results are in Table 5.4.

Analyzing the results of the previous table, it can be noticed that the added time of the Stop and Start operation exceeds 15 seconds compared to the Restart operation. This difference can have to do with several situations, mainly because of the queries to the AAI component or the error posted by the DMaaP component when trying to update the ONAP inventory.

5.3.3 Running Tests on Rebuild Operation

The initial plan for this part of the test was to run iterations of Snapshot and Rebuild operations in this exact order on the vFW VNF. However, the Snapshot operation, in Casablanca release, has a bug in the validation creation of a snapshot, due to an error on a call that finds the created image. Because of that error, APP-C throws an error of a failed operation.

To execute this part of the test, similar to the previous test, it was created a call on Postman software to Rebuild operation, to run in a series of ten iterations. It was used as a Snapshot created by the failed Snapshot operation, becoming this test realistic as possible.

5.3.3.1 Rebuild Operation Results

Similar to the previous test, it was built a diagram to explain how this operation executes the operation phases 5.11. This operation was repeted ten times during the tests.

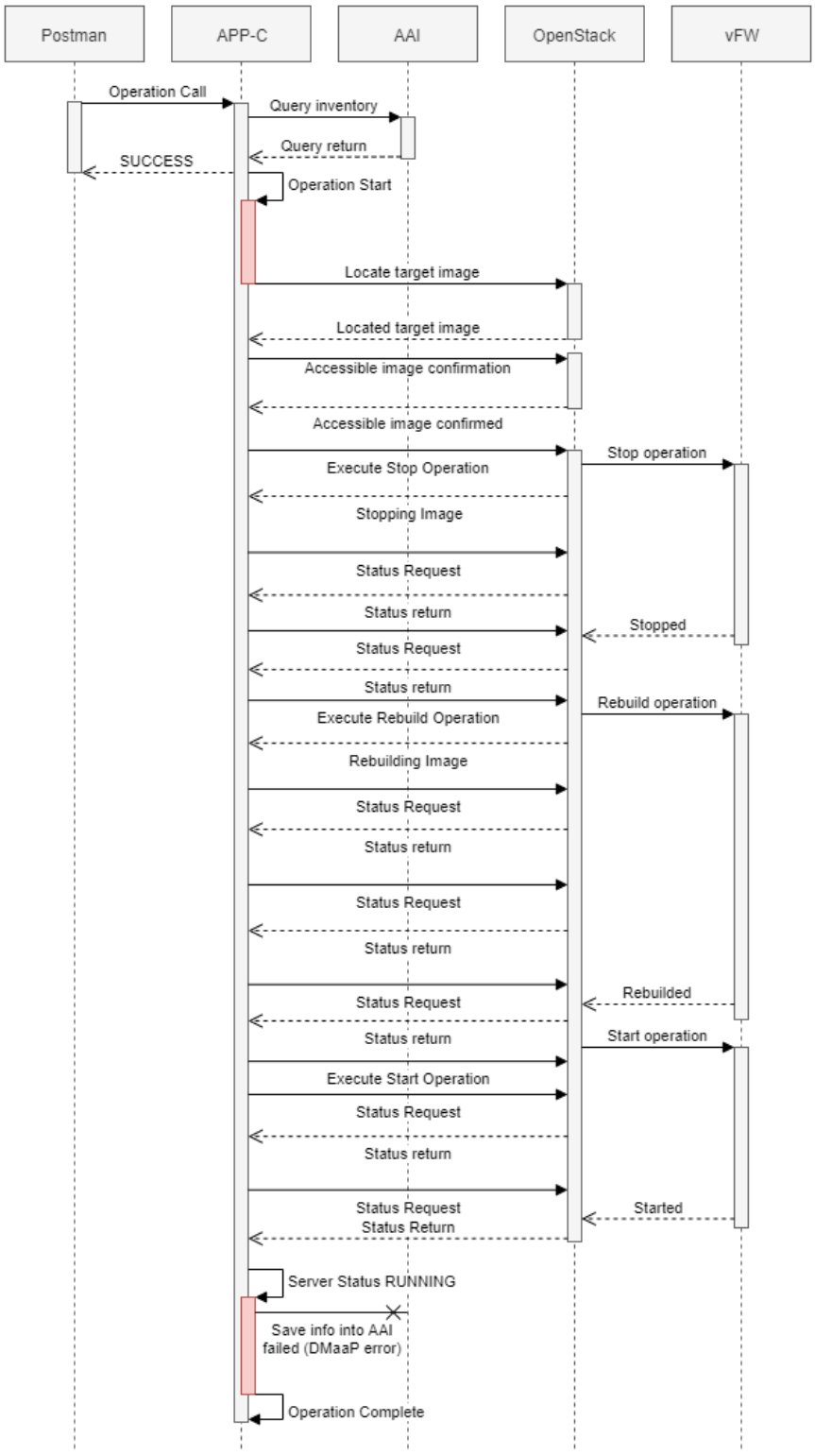


Figure 5.11: Rebuild Operation Diagram

Analyzing the diagram, there are some similar phases to the previous operations tested. After Postman sends a call to APP-C, this component consults the AAI inventory and, after being succeeded, it sends a successful message to Postman. However, this time the APP-C sends a request to OpenStack with a Snapshot identification to locate the snapshot. If successful, APP-C asks OpenStack to check if the snapshot is available to be used for Rebuild operation. When APP-C receives the confirmation, it sends a request to execute the Stop operation on the VNF and periodically send a status request. After receiving the confirmation of the stopped VNF, it sends a rebuild request to execute on the VNF and then it starts to send status requests. When it receives the confirmation for that operation, APP-C sends the Start request to OpenStack and once more start to send status requests. Finally, after the confirmation of "Server Status RUNNING", APP-C tries, unsuccessfully, to update AAI and at the end of the delay, throws an operation complete status.

The process to obtain the time results is similar to the previous test and are stored in the following tables:

Table 5.5: Test result from Rebuild operation (ms)

	Average	**Standard Deviation**	**Median**
Input received → AAI (query finish)	3009	237	2983
AAI (query finish) → Operation Start	1101	362	994
Operation Start → Start Request	12226	368	12226
Start Requests → Located Target Image	1406	512	1299
Located Image → Accessible image confirmation	608	365	415
Accessible Confirmation → Stopping Image	901	80	879
Stopping image → Rebuilding image	21785	479	21575
Rebuilding image → Starting image	12261	586	12284
Starting image → Running image	21927	297	21895
Running image → Operation Complete (with DMaaP error)	8009	174	8049
Input Received → Operation Start	4110	454	4047
Operation Start → Operation Complete (with DMaaP error)	79123	1011	78791
Input Received → Operation Complete (with DMaaP error)	83233	912	83038
AAI Query Time by log system	2516.4	2474	207.121

Analyzing the results of this test, it is possible to conclude that the phases that compose this operation are different from those of the previous tests. This operation consumes more time because this operation search for the VNF in the infrastructure, validate if the VNF have a valid image to execute the operation and then stops the VNF, rebuild the VNF with the Snapshot sent by the APP-C call and, at the end of the rebuild phase, starts the rebuilt VNF.

Concluding a statistical analysis of the measured results (Figure 5.12), once again can be recognized as the low variation on the measured values in the series of tests for each phase. It also evidences the stability of this operation execution by the APP-C component.

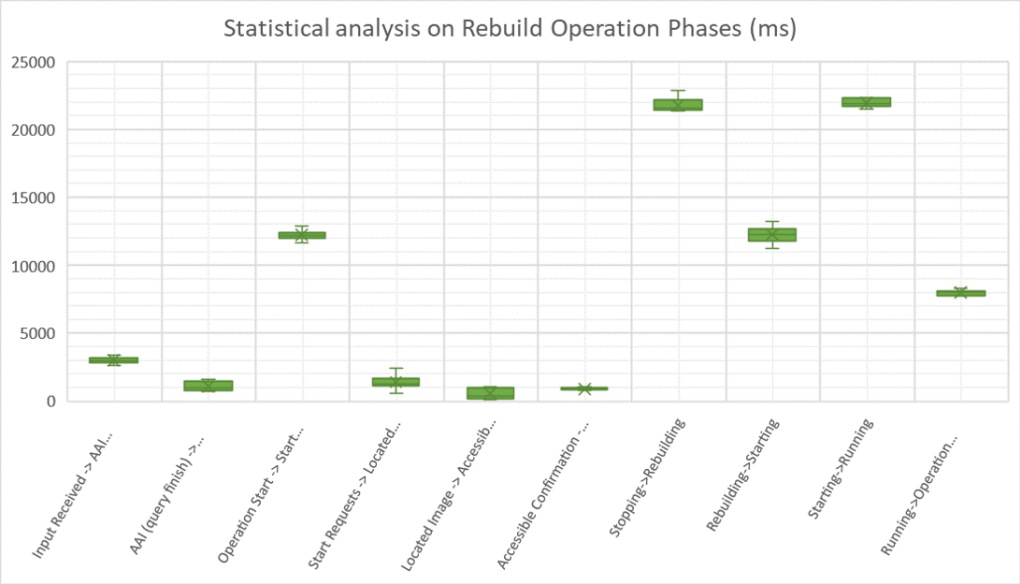


Figure 5.12: Box and Whiskers plot for Statistical analysis on Rebuild Operation Phases

Interpreting the time cumulative graphic of this operation, in Figure 5.13, is notorious that the slope between phases is higher than the previous operations. This is because the Rebuild operation is more complex and may require more resources to operate. Nevertheless, the ideal platform functionality should have a slope between phases of near zero. As seen in the graphic, some phases have a greater weight in operation time than they should have.

The most critical phase is between the operation start status and the beginning of the requests to OpenStack. As seen in the values measured, this phase takes more than twelve seconds to start requesting. The cause for this higher slope was unidentified but it should be optimized to decrease the weight of this phase in the operation. Another critical phase is the final phase, similar to the identified phase in Restart operation. The cause of this weight on the operation is the same as Restart operation. In other words, in that phase, APP-C component unsuccessfully tries to update the inventory and creates a delay in the operation, increasing the weight of the phase in the overall operation.

The weight caused by OpenStack phases is independent of the APP-C component and the ONAP platform, whereby they aren't critical for this analysis despite their weight on the operation.

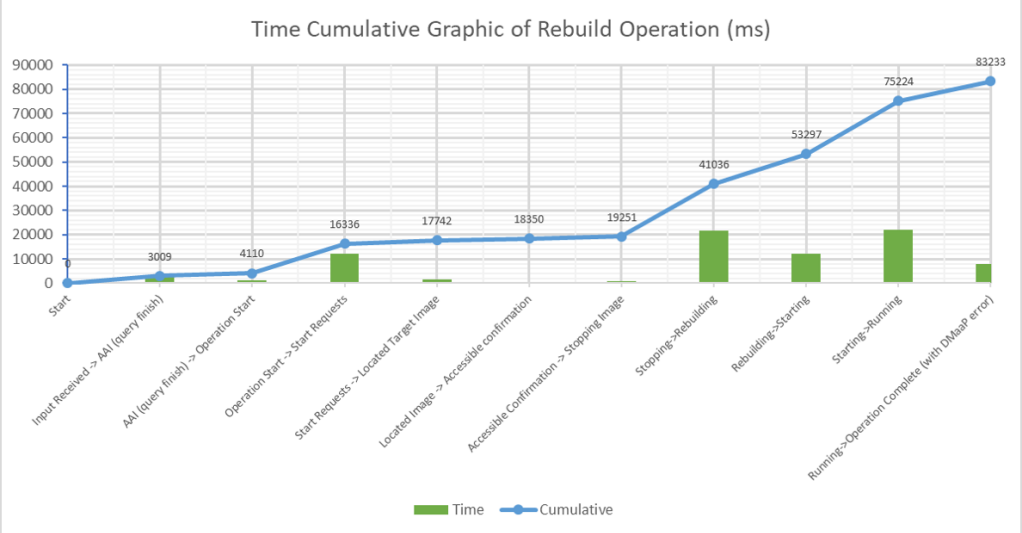


Figure 5.13: Time cumulative graphic of Rebuild operation

5.4 Ansible LCM Operations

To evaluate LCM operations through Ansible, it was planned to execute a simple configuration operation to vFW VNF. This operation consists in using the APP-C API to execute the operation through the Ansible Adapter and Ansible Server. The configuration operation planned to execute was install or uninstall the Uncomplicated Firewall (UFW) daemon, the native firewall daemon of Ubuntu OS. It was planned to build a playbook to execute that operation, to be added in APP-C Ansible Server component and to onboard the operation in APP-C CDT to execute successfully.

5.4.1 Testing Ansible with Virtual Machines

Before testing Ansible LCM operations on ONAP platform, it was tested some functionalities of this framework in a virtual environment. With Ansible it was possible to perform some simple operations such as pinging to several clients, performing ifconfig commands and other Linux terminal commands. More complex operations such as uploading and configuring files were then tested to configure daemons and OS programs. All those operations were successfully tested with multiple clients.

5.4.2 Testing Ansible LCM operations

After testing Ansible standalone in a VM environment, it was tried to test the Ansible LCM operation on ONAP platform. However, the execution of this

test wasn't occurred in time because of some difficulties on ONAP platform.

The first difficulty found was the instability of ONAP platform and Kubernetes containers and pods, or even disconnection of an entire VM from the rest of the deployment. The problem that causes that error wasn't found and the only solution found was restarting the VM disconnected. However, when the solution was executed successfully, some containers started to fail and the infrastructure was deleted and a new deployment was entirely deployed.

After the ONAP platform redeployment, APP-C CDT doesn't connected to other APP-C components. However, the origin of the error wasn't found in time even instantiating all APP-C pods. This error was critical so that it could not test the LCM transactions with Ansible in time.

5.5 Identified challenges

As said in the previous section, some challenges and difficulties disabled the execution of LCM operations in time due to some problems and instability on ONAP platform. In the following sections, some of those challenges and difficulties with ONAP are described.

5.5.1 Testbed Resource Requirements

The ONAP deployment relies on the usage of a very significant amount of resources. However, in the testbed, those amount of resources isn't available. To solve this problem, it was followed a minimal installation, identifying dependencies in the platform required to onboard and apply LCM operations. [21]

5.5.2 ONAP Component Dependencies

One of the major difficulties was the ONAP dependencies, more precisely between pods on Kubernetes throwing failed and pending status. Most of these problems were caused by existing multiple dependencies between pods of different containers, that should wait for each other to be instantiated. However, sometimes those components don't wait for each other to be instantiated and throw pending or failed status (i.e [2]). This occurs when the infrastructure is slower and the waiting timer coded by ONAP development team isn't enough to instantiate correctly all the components. One solution is to delete individually each failed pods to instantiate correctly the failed pods.

5.5.3 APP-C Database Component Instantiation

Sometimes, the APP-C Database fails to instantiate, disabling the APP-C Core to instantiate as well. After the analysis of logs, the problem was solved

by editing the parameter safe_to_bootstrap to 1 in file grastate.dat in the APP-C docker files.

5.5.4 APP-C Database: OpenStack Operation Onboarding

In Casablanca release, sometimes onboarding OpenStack operations through APP-C CDT, the component don't populate the Database. In these situations, it's needed to add those operations directly on APP-C Database using the following line:

```
insert into PROTOCOL_REFERENCE (ACTION, VNF_TYPE, PROTOCOL, TEMPLATE,
    ACTION_LEVEL) values (<OPERATION_NAME>, <VNF_NAME>, "OS", "N", "
    VM");
```

5.6 Testing and Evaluation Conclusions

Completing the tests and validations of the APP-C component and some of the LCM operations, we can see that although the resources used are reduced, the time of the operations has interesting values compared to some support operations that are used in current mobile networks. For example, the Rebuild operation, which takes just over a minute and twenty seconds, compared to replacing physical equipment, which can take hours, days or even weeks, is very small. The result of this test proves the advantages of using NFV concerning the use of legacy physical equipment.

Although the VMs used in testing contains a CirrOS image, which is a very light test operating system, a VNF that can be used in a network service, if optimized well, should be equally light and will not have a major impact on the latency of operations.

Nevertheless, the OpenStack platform was running on old material (Table 4.1) so the operation time may be higher than if a better-prepared server was used.

5.7 Chapter Summary

In this last chapter, the objectives of infrastructure validation and assessment were stated, being the onboarding of a VNF on the ONAP platform and the execution of LCM operations on network services using OpenStack and Ansible. The methodology used to achieve these objectives involves the instantiation of a network service through ONAP and, with this instance, performing LCM operations with OpenStack and Ansible. The metrics were removed during the execution of the tests through the APP-C logs. With a repetition of operations, it was defined to remove the metrics and use average values, and with these results, evaluate the performance of the platform, specifically the operations used.

After defining the methodology to be used for the tests, the important steps in the description, onboarding and instantiation of a VNF were described. In this way, service onboarding processes were identified, from resource onboarding to service distribution.

In the execution of the tests, LCM operations were first performed from OpenStack, more specifically the Start, Restart, Stop and Rebuild operations. With the tests performed, it was possible to identify some phases of each of the operations, some of which are coincident with each other, and analyse the temporal weight that each one of these phases has on the operation itself. Those result were expressed in form of graphics.

For LCM operations via Ansible, it has been planned to perform simple configuration operations on a vFW by installing or uninstalling UFW or configuring its configuration files.

To understand how Ansible works, it was necessary to first create a test environment with virtual machines where simple operations such as pinging several clients through Ansible, performing ifconfig operations and finally, more complex operations such as uploading and configuring files were performed.

However, it was not possible to perform LCM operations through Ansible in ONAP due to several difficulties found in the platform, such as the instability of the platform and the virtual machines associated with it.

During the implementation of the platform, network services and LCM operations, some challenges and difficulties were encountered, such as resource requirements, dependencies between ONAP components, instantiation of the APP-C database and the onboarding of OpenStack operations in the APP-C database.

Finally, it was concluded that although there are no comparison values for the results obtained, these are interesting because they validate the fact that it is possible to carry out the operation of management and maintenance of virtualized network resources in seconds or minutes, compared to the time required to carry out the same type of operations in legacy network services, which may take hours, days or even weeks to execute.

Chapter 6

Conclusions

5G networks will be the near future generation network technology that will bring new use cases and new services to operators and clients, giving space to new features such as Network Slicing and Service-Based Architecture networks. More than the enhancement of Radio Access Networks, the core virtualization will be an important enabler to this network generation and will revolutionize the way we see mobile networks.

ONAP platform will be an important platform to operators to distribute and scale old and new services easily in the entire range of the network, in and End-to-End perspective, enabling Orchestration of all services and turn them virtualized to centralize and distribute more easily those services.

With the first objective referred to in the Introduction, an analysis of the standardisation of ETSI NFV was carried out, highlighting its architecture, functions and other components that make network virtualisation possible. The importance that this technology will have in 5G networks was also highlighted.

The second objective, which refers to the analysis of open-source orchestration software, consisted in a study on two open-source platforms, namely OSM and ONAP. This analysis helped to understand the state of the current software for this purpose and to understand in a general way, its functioning. Finally, the ONAP platform was selected because it is a more comprehensive platform that goes beyond the MANO functions.

The third objective was completed by integrating the ONAP platform with the OpenStack platform. For this, a custom installation of the ONAP platform was made to make LCM operations to VNF, these instantiated in the OpenStack cloud platform.

In order to achieve the last objective, which is to execute LCM operations in network services, we used the ONAP APP-C component, more precisely the tools available for operations via OpenStack as well as Ansible. In order to evaluate the execution of these operations, a series of tests to several operations

was performed, taking out temporal metrics and analyzing the latency of each of the operations with them. However, the operations with Ansible were not possible to be performed due to some errors in the platform. From these results it was proven the importance that this type of platforms, as well as the network virtualization, can bring to 5G networks.

Although the ONAP platform is stable and production-ready, in the R&D context the platform has some instabilities and still presents some difficulties for developers to test the platform in new environments and new use cases. The existing documentation of the platform, despite much, is still confusing and not very organized, being spread over several locations. This complicates the role of the developer when exploring, studying and developing the ONAP platform.

In the end of this dissertation, some tasks that can be important to the future work of this project can identify :

- Test new LCM operations, don't tested in this dissertation context;
- Onboard new types and/or more complex VNFs and Network Services and validate the usage of those VNFs;
- Research and test new use cases focused on 5G networks.

References

[1] AAF Architecture — casablanca documentation. URL: https://onap.readthedocs.io/en/casablanca/submodules/aaf/authz.git/docs/sections/architecture/aaf{_}architecture.html. [cited on p. 26]

[2] [AAI-1759] AAI model loader failed model distribution - ONAP. URL: https://jira.onap.org/browse/AAI-1759. [cited on p. 57]

[3] Architecture — casablanca documentation. URL: https://onap.readthedocs.io/en/casablanca/submodules/aai/aai-common.git/docs/platform/architecture.html. [cited on p. 25]

[4] DCAE Architecture — casablanca documentation. URL: https://onap.readthedocs.io/en/casablanca/submodules/dcaegen2.git/docs/sections/architecture.html. [cited on p. 26]

[5] DMaaP – Data Movement as a Platform (5/16/17) - Developer Wiki - Confluence. URL: https://wiki.onap.org/pages/viewpage.action?pageId=3247130. [cited on p. 26]

[6] Master Service Orchestrator (MSO) - Developer Wiki - Confluence. URL: https://wiki.onap.org/pages/viewpage.action?pageId=1015834. [cited on p. 28]

[7] MSB Architecture — casablanca documentation. URL: https://onap.readthedocs.io/en/casablanca/submodules/msb/apigateway.git/docs/platform/architecture.html. [cited on p. 26]

[8] ONAP VF-C — casablanca documentation. URL: https://onap.readthedocs.io/en/casablanca/submodules/vfc/nfvo/lcm.git/docs/platform/index.html. [cited on p. 29]

[9] Policy Architecture — casablanca documentation. URL: https://onap.readthedocs.io/en/casablanca/submodules/policy/engine.git/docs/platform/architecture.html. [cited on p. 27]

[10] Portal Architecture — casablanca documentation. URL: https://onap.readthedocs.io/en/casablanca/submodules/portal.git/docs/platform/architecture.html. [cited on p. 28]

[11] Service Design and Creation (SDC) Portal - Developer Wiki - Confluence. URL: https://wiki.onap.org/display/DW/Service+Design+and+Creation+{%}28SDC{%}29+Portal. [cited on p. 28]

[12] vFW CDS Casablanca - Developer Wiki - Confluence. URL: https://wiki.onap.org/display/DW/vFW+CDS+Casablanca. [cited on p. 42]

[13] VID project (5/17/17) - Developer Wiki - Confluence. URL: https://wiki.onap.org/pages/viewpage.action?pageId=5735532. [cited on p. 29]

[14] 3GPP. About 3GPP Home. URL: https://www.3gpp.org/about-3gpp/about-3gpp. [cited on p. 5]

[15] 3GPP. 3GPP TS 22.261 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Service requirements for the 5G system;. 2018. [cited on p. 5, 7]

[16] 5GO.pt. 5go | Activities. URL: https://5go.pt/en/activities/. [cited on p. 2]

[17] 5GO.pt. 5go | Project. URL: https://5go.pt/en/project/. [cited on p. 2, 3]

[18] 5GO.pt. Deliverable D2.1 Use cases and requirements for solutions targetting 5G network core. 2018. [cited on p. 10]

[19] 5GPPP. Vision and mission ■ 5G-PPP. URL: https://5g-ppp.eu/about-us/. [cited on p. 5]

[20] 5GPPP. View on 5G Architecture (Version 2 . 0). pages 1–140, 2017. [cited on p. 7, 18]

[21] João Aires, Pedro Barbosa, Sérgio Figueiredo, Bruno Parreira, Jorge Mamede, and Manuel Ricardo. Addressing end-to-end Orchestration of Virtualized Telco Services using ONAP in a R&D environment. In *InForum 2019: Comunicações e Redes de Computadores (CRC)*, page 12, 2019. [cited on p. xv, 3, 42, 43, 57]

[22] Iqbal Bedi. *Setting the scene for 5G: Opportunities & Challenges*. 2018. doi:http://handle.itu.int/11.1002/pub/811d7a5f-en. [cited on p. xv, 9]

[23] Andy Bierman, Martin Björklund, and Kent Watsen. RESTCONF Protocol. RFC 8040, January 2017. URL: https://rfc-editor.org/rfc/rfc8040.txt, doi:10.17487/RFC8040. [cited on p. 39]

[24] Gabriel Brown. Service-Based Architecture for 5G Core Networks. 2017. [cited on p. 8, 9]

[25] Chef. Chef Docs - Platform Overview, 2019. URL: `https://docs.chef.io/platform{_}overview.html`. [cited on p. xv, 38, 39]

[26] Margaret Chiosi and et al. Network Functions Virtualisation (NFV). *ETSI white paper*, (1):1–20, 2014. `doi:DGS/NFV-0011`. [cited on p. 12, 17]

[27] Chris Donley. ONAP Casablanca Architecture (v3.0.3). [cited on p. 26, 34]

[28] Rob Enns, Martin Björklund, Andy Bierman, and Jürgen Schönwälder. Network Configuration Protocol (NETCONF). RFC 6241, June 2011. URL: `https://rfc-editor.org/rfc/rfc6241.txt`, `doi:10.17487/RFC6241`. [cited on p. 37, 38]

[29] ETSI. ETSI GS NFV 001 V1.1.1 Network Functions Virtualisation (NFV); Use Cases. pages 1–50, 2013. [cited on p. 12, 18]

[30] ETSI. ETSI GS NFV-MAN 001 V1.1.1 Network Functions Virtualisation (NFV); Management and Orchestration. *Gs Nfv-Man 001 V1.1.1*, 1:1–184, 2014. URL: `http://www.etsi.org/deliver/etsi{_}gs/NFV-MAN/001{_}099/001/01.01.01{_}60/gs{_}nfv-man001v010101p.pdf`. [cited on p. 14, 15, 16]

[31] ETSI. ETSI GS NFV-SWA 001 Network Functions Virtualisation (NFV); Virtual Network Functions Architecture. pages 1–93, 2014. [cited on p. 13]

[32] ETSI. ETSI GS NFV-EVE 005 V1.1.1 Network Functions Virtualisation (NFV); Ecosystem; Report on SDN Usage in NFV Architectural Framework. 2015. [cited on p. 17]

[33] ETSI. ETSI GR NFV 001 V1.2.1 Network Functions Virtualisation (NFV); Use Cases. *ETSI*, pages 1–81, 2017. `doi:RGR/NFV-001ed121`. [cited on p. xv, 12, 13, 14, 15, 16, 18, 19]

[34] ETSI. ETSI GS NFV-IFA 011 V3.1.1 Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; VNF Descriptor and Packaging Specification. 2018. [cited on p. 16]

[35] ETSI. ETSI GS NFV-SOL 001 V2.5.1 Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; NFV descriptors based on TOSCA specification. 2018. [cited on p. 16, 17, 18]

[36] ETSI. ETSI - NFV, 2019. URL: `https://www.etsi.org/technologies/nfvhttps://www.etsi.org/technologies-clusters/technologies/nfv`. [cited on p. 12]

[37] ITU-R. Recommendation ITU-R M.2083-0 IMT Vision – Framework and overall objectives of the future development of IMT for 2020 and beyond. pages 1–21, 2015. [cited on p. 6]

[38] Amar Kapadia. *ONAP Demystified*. [cited on p. xv, 18, 24, 25, 28, 34]

[39] NetWorld2020 ETP Expert Working Group. 5G: Challenges, Research Priorities, and Recommendations. pages 1–45, 2014. URL: http://networld2020.eu/wp-content/uploads/2014/02/NetWorld2020{_}Joint-Whitepaper-V8{_}public-consultation.pdf. [cited on p. 5, 10]

[40] NGMN Alliance. 5G White Paper. *By NGMN Alliance 1.0*, pages 1–125, 2015. [cited on p. xv, 8]

[41] NGMN Alliance. Service-Based Architecture in 5G. pages 1–17, 2018. [cited on p. 8, 9]

[42] NGMN Alliance. The NGMN Alliance At a Glance NGMN at a Glance : Vision , Mission , Organisation and Ways of Working. pages 1–13, 2018. [cited on p. 5]

[43] ONAP. APPC User Guide — casablanca documentation. URL: https://onap.readthedocs.io/en/casablanca/submodules/appc.git/docs/APPCUserGuide/APPCUserGuide.html. [cited on p. xv, 25, 34, 35, 37, 38, 39]

[44] ONAP. ONAP MultiCloud Architecture — casablanca documentation. URL: https://docs.onap.org/en/casablanca/submodules/multicloud/framework.git/docs/MultiCloud-Architecture.html. [cited on p. xv, 26, 27]

[45] ONAP. ONAP Operations Manager Project — master branch documentation. URL: https://docs.onap.org/en/latest/submodules/oom.git/docs/oom{_}project{_}description.html. [cited on p. 24, 27]

[46] ONAP. APPC CDT User Guide — casablanca documentation, 2018. URL: https://onap.readthedocs.io/en/casablanca/submodules/appc.git/docs/APPCCDTGuide/APPCCDTGuide.html. [cited on p. xvii, 35, 36, 37]

[47] ONF. SDN Architecture 1.0 Overview. 2014. [cited on p. xv, 11, 17]

[48] ONF. ONF TR-518 Relationship of SDN and NFV. (1):1–20, 2015. URL: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/onf2015.310{_}Architectural{_}comparison.08-2.pdf. [cited on p. 17]

[49] ONF. ONF TR-521 SDN Architecture Issue 1.1. In *ONF White Paper*, number 1.1, pages 1–59, 2016. [cited on p. 10, 11]

[50] OSM. OSM Information Model Release 1. 2016. [cited on p. 18, 21, 22]

[51] OSM. OSM Release THREE a technical review. 2017. [cited on p. 22, 23]

[52] OSM. OSM Release TWO a technical overview. 2017. [cited on p. 22]

[53] OSM. OSM Release FOUR Technical Overview. 2018. [cited on p. xv, 21, 22]

[54] François Rancy. IMT for 2020 and beyond. In Ramjee Prasad, editor, *5G Outlook- Innovations and Applications*, chapter 6, pages 69–84. River Publishers, 2016. [cited on p. xv, 6, 9]

[55] Red Hat Ansible. ANSIBLE IN DEPTH (White Paper). Technical report, 2017. URL: `https://www.ansible.com/hubfs/pdfs/Ansible-InDepth-WhitePaper.pdfhttps://www.ansible.com/resources/whitepapers/ansible-in-depth`. [cited on p. 38]

Appendix A

vFW Heat Templates

```
#base_vfw.env
parameters:
  image_name: cirros
  flavor_name: cirros
  public_net_id: 4d8925d9-e42b-4456-b637-a5d9b3d9cbf3
  unprotected_private_net_id: zdfw1fwl01_unprotected
  unprotected_private_subnet_id: zdfw1fwl01_unprotected_sub
  unprotected_private_net_cidr: 192.168.10.0/24
  protected_private_net_id: zdfw1fwl01_protected
  protected_private_subnet_id: zdfw1fwl01_protected_sub
  protected_private_net_cidr: 192.168.20.0/24
  onap_private_net_id: oam_onap_k0H4
  onap_private_subnet_id: oam_onap_k0H4
  onap_private_net_cidr: 10.0.0.0/16
  vfw_private_ip_0: 192.168.10.100
  vfw_private_ip_1: 192.168.20.100
  vfw_private_ip_2: 10.0.100.1
  vpg_private_ip_0: 192.168.10.200
  vsn_private_ip_0: 192.168.20.250
  vsn_private_ip_1: 10.0.100.3
  vfw_name_0: zdfw1fwl01fwl01
  vsn_name_0: zdfw1fwl01snk01
  vnf_id: vFirewall_demo_app
  vf_module_id: vFirewallCL
  dcae_collector_ip: 127.0.0.1
  dcae_collector_port: 8080
  repo_url_blob: https://nexus.onap.org/content/sites/raw
  repo_url_artifacts: https://nexus.onap.org/content/groups/staging
  demo_artifacts_version: 1.1.0
  install_script_version: 1.1.0-SNAPSHOT
  key_name: onap_key_k0H4
  pub_key: ssh-rsa <ssh_key>
  cloud_env: openstack
#  physical_resource_id_vsn: interface_vsn
#  physical_resource_id_vfw: interface_vfw
```

```
#base_vfw.yaml
#############################################################################

#
#========================LICENSE_START=======================================
#
#
# Copyright (c) 2017 AT&T Intellectual Property. All rights reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#         http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
#========================LICENSE_END=========================================
#
# ECOMP is a trademark and service mark of AT&T Intellectual Property.
#
#############################################################################


heat_template_version: 2013-05-23

description: Heat template that deploys vFirewall-based demo for ONAP

###############
#             #
# PARAMETERS  #
#             #
###############

parameters:
  image_name:
    type: string
    label: Image name or ID
    description: Image to be used for compute instance
  flavor_name:
    type: string
    label: Flavor
    description: Type of instance (flavor) to be used
```

```
45  public_net_id:
46    type: string
47    label: Public network name or ID
48    description: Public network that enables remote connection to VNF
49  unprotected_private_net_id:
50    type: string
51    label: Unprotected private network name or ID
52    description: Private network that connects vPacketGenerator with
      vFirewall
53  unprotected_private_subnet_id:
54    type: string
55    label: Unprotected private subnetwork name or ID
56    description: Private subnetwork of the protected network
57  unprotected_private_net_cidr:
58    type: string
59    label: Unprotected private network CIDR
60    description: The CIDR of the unprotected private network
61  protected_private_net_id:
62    type: string
63    label: Protected private network name or ID
64    description: Private network that connects vFirewall with vSink
65  protected_private_subnet_id:
66    type: string
67    label: Protected private subnetwork name or ID
68    description: Private subnetwork of the unprotected network
69  protected_private_net_cidr:
70    type: string
71    label: Protected private network CIDR
72    description: The CIDR of the protected private network
73  onap_private_net_id:
74    type: string
75    label: ONAP management network name or ID
76    description: Private network that connects ONAP components and
      the VNF
77  onap_private_subnet_id:
78    type: string
79    label: ONAP management sub-network name or ID
80    description: Private sub-network that connects ONAP components
      and the VNF
81  onap_private_net_cidr:
82    type: string
83    label: ONAP private network CIDR
84    description: The CIDR of the protected private network
85  vfw_private_ip_0:
86    type: string
87    label: vFirewall private IP address towards the unprotected
      network
88    description: Private IP address that is assigned to the vFirewall
       to communicate with the vPacketGenerator
89  vfw_private_ip_1:
90    type: string
```

```
91      label: vFirewall private IP address towards the protected network
92      description: Private IP address that is assigned to the vFirewall
         to communicate with the vSink
93    vfw_private_ip_2:
94      type: string
95      label: vFirewall private IP address towards the ONAP management
        network
96      description: Private IP address that is assigned to the vFirewall
         to communicate with ONAP components
97    vpg_private_ip_0:
98      type: string
99      label: vPacketGenerator private IP address towards the
        unprotected network
100     description: Private IP address that is assigned to the
        vPacketGenerator to communicate with the vFirewall
101   vsn_private_ip_0:
102     type: string
103     label: vSink private IP address towards the protected network
104     description: Private IP address that is assigned to the vSink to
        communicate with the vFirewall
105   vsn_private_ip_1:
106     type: string
107     label: vSink private IP address towards the ONAP management
        network
108     description: Private IP address that is assigned to the vSink to
        communicate with ONAP components
109   vfw_name_0:
110     type: string
111     label: vFirewall name
112     description: Name of the vFirewall
113   vsn_name_0:
114     type: string
115     label: vSink name
116     description: Name of the vSink
117   vnf_id:
118     type: string
119     label: VNF ID
120     description: The VNF ID is provided by ONAP
121   vf_module_id:
122     type: string
123     label: vFirewall module ID
124     description: The vFirewall Module ID is provided by ONAP
125   dcae_collector_ip:
126     type: string
127     label: DCAE collector IP address
128     description: IP address of the DCAE collector
129   dcae_collector_port:
130     type: string
131     label: DCAE collector port
132     description: Port of the DCAE collector
133   key_name:
```

```
134       type: string
135       label: Key pair name
136       description: Public/Private key pair name
137     pub_key:
138       type: string
139       label: Public key
140       description: Public key to be installed on the compute instance
141     repo_url_blob:
142       type: string
143       label: Repository URL
144       description: URL of the repository that hosts the demo packages
145     repo_url_artifacts:
146       type: string
147       label: Repository URL
148       description: URL of the repository that hosts the demo packages
149     install_script_version:
150       type: string
151       label: Installation script version number
152       description: Version number of the scripts that install the vFW
        demo app
153     demo_artifacts_version:
154       type: string
155       label: Artifacts version used in demo vnfs
156       description: Artifacts (jar, tar.gz) version used in demo vnfs
157     cloud_env:
158       type: string
159       label: Cloud environment
160       description: Cloud environment (e.g., openstack, rackspace)
161 #   physical_resource_id_vfw:
162 #     type: string
163 #     label: vFW interface name
164 #     description: Name of the vFW interface (e.g., interface vFW)
165 #   physical_resource_id_vsn:
166 #     type: string
167 #     label: vsn interface name
168 #     description: Name of the vsn interface (e.g., interface vsn)
169
170 ##############
171 #            #
172 # RESOURCES #
173 #            #
174 ##############
175
176 resources:
177   random-str:
178     type: OS::Heat::RandomString
179     properties:
180       length: 4
181
182   my_keypair:
183     type: OS::Nova::KeyPair
```

```
184     properties:
185       name:
186         str_replace:
187           template: base_rand
188           params:
189             base: { get_param: key_name }
190             rand: { get_resource: random-str }
191       public_key: { get_param: pub_key }
192       save_private_key: false
193
194   unprotected_private_network:
195     type: OS::Neutron::Net
196     properties:
197       name: { get_param: unprotected_private_net_id }
198
199   unprotected_private_subnet:
200     type: OS::Neutron::Subnet
201     properties:
202       name: { get_param: unprotected_private_subnet_id }
203       network_id: { get_resource: unprotected_private_network }
204       cidr: { get_param: unprotected_private_net_cidr }
205
206   protected_private_network:
207     type: OS::Neutron::Net
208     properties:
209       name: { get_param: protected_private_net_id }
210
211   protected_private_subnet:
212     type: OS::Neutron::Subnet
213     properties:
214       name: { get_param: protected_private_subnet_id }
215       network_id: { get_resource: protected_private_network }
216       cidr: { get_param: protected_private_net_cidr }
217
218   # Virtual Firewall instantiation
219   vfw_private_0_port:
220     type: OS::Neutron::Port
221     properties:
222       network: { get_resource: unprotected_private_network }
223       fixed_ips: [{"subnet": { get_resource:
      unprotected_private_subnet }, "ip_address": { get_param:
      vfw_private_ip_0 }}]
224
225   vfw_private_1_port:
226     type: OS::Neutron::Port
227     properties:
228       allowed_address_pairs: [{ "ip_address": { get_param:
      vpg_private_ip_0 }}]
229       network: { get_resource: protected_private_network }
230       fixed_ips: [{"subnet": { get_resource: protected_private_subnet
      }, "ip_address": { get_param: vfw_private_ip_1 }}]
```

```
231
232   vfw_private_2_port:
233     type: OS::Neutron::Port
234     properties:
235       network: { get_param: onap_private_net_id }
236       fixed_ips: [{"subnet": { get_param: onap_private_subnet_id }, "
      ip_address": { get_param: vfw_private_ip_2 }}]
237
238   vfw_0:
239     type: OS::Nova::Server
240     properties:
241       image: { get_param: image_name }
242       flavor: { get_param: flavor_name }
243       name: { get_param: vfw_name_0 }
244       key_name: { get_resource: my_keypair }
245       networks:
246         - network: { get_param: public_net_id }
247         - port: { get_resource: vfw_private_0_port }
248         - port: { get_resource: vfw_private_1_port }
249         - port: { get_resource: vfw_private_2_port }
250       metadata: {vnf_id: { get_param: vnf_id }, vf_module_id: {
      get_param: vf_module_id }}
251       user_data_format: RAW
252       user_data:
253         str_replace:
254           params:
255             __dcae_collector_ip__ : { get_param: dcae_collector_ip }
256             __dcae_collector_port__ : { get_param:
      dcae_collector_port }
257             __demo_artifacts_version__ : { get_param:
      demo_artifacts_version }
258             __install_script_version__ : { get_param:
      install_script_version }
259             __vfw_private_ip_0__ : { get_param: vfw_private_ip_0 }
260             __vfw_private_ip_1__ : { get_param: vfw_private_ip_1 }
261             __vfw_private_ip_2__ : { get_param: vfw_private_ip_2 }
262             __unprotected_private_net_cidr__ : { get_param:
      unprotected_private_net_cidr }
263             __protected_private_net_cidr__ : { get_param:
      protected_private_net_cidr }
264             __onap_private_net_cidr__ : { get_param:
      onap_private_net_cidr }
265             __cloud_env__ : { get_param: cloud_env }
266           template: |
267             #!/bin/bash
268
269             # Create configuration files
270             mkdir /opt/config
271             echo "__dcae_collector_ip__" > /opt/config/
      dcae_collector_ip.txt
272             echo "__dcae_collector_port__" > /opt/config/
```

```
    dcae_collector_port.txt
273             echo "__repo_url_blob__" > /opt/config/repo_url_blob.txt
274             echo "__repo_url_artifacts__" > /opt/config/
    repo_url_artifacts.txt
275             echo "__demo_artifacts_version__" > /opt/config/
    demo_artifacts_version.txt
276             echo "__install_script_version__" > /opt/config/
    install_script_version.txt
277             echo "__vfw_private_ip_0__" > /opt/config/
    vfw_private_ip_0.txt
278             echo "__vfw_private_ip_1__" > /opt/config/
    vfw_private_ip_1.txt
279             echo "__vfw_private_ip_2__" > /opt/config/
    vfw_private_ip_2.txt
280             echo "__unprotected_private_net_cidr__" > /opt/config/
    unprotected_private_net_cidr.txt
281             echo "__protected_private_net_cidr__" > /opt/config/
    protected_private_net_cidr.txt
282             echo "__onap_private_net_cidr__" > /opt/config/
    onap_private_net_cidr.txt
283             echo "__cloud_env__" > /opt/config/cloud_env.txt
284
285             # Download and run install script
286
287
288   # Virtual Sink instantiation
289   vsn_private_0_port:
290     type: OS::Neutron::Port
291     properties:
292       network: { get_resource: protected_private_network }
293       fixed_ips: [{"subnet": { get_resource: protected_private_subnet
      }, "ip_address": { get_param: vsn_private_ip_0 }}]
294
295   vsn_private_1_port:
296     type: OS::Neutron::Port
297     properties:
298       network: { get_param: onap_private_net_id }
299       fixed_ips: [{"subnet": { get_param: onap_private_subnet_id }, "
    ip_address": { get_param: vsn_private_ip_1 }}]
300
301   vsn_0:
302     type: OS::Nova::Server
303     properties:
304       image: { get_param: image_name }
305       flavor: { get_param: flavor_name }
306       name: { get_param: vsn_name_0 }
307       key_name: { get_resource: my_keypair }
308       networks:
309         - network: { get_param: public_net_id }
310         - port: { get_resource: vsn_private_0_port }
311         - port: { get_resource: vsn_private_1_port }
```

```
312       metadata: {vnf_id: { get_param: vnf_id }, vf_module_id: {
    get_param: vf_module_id }}
313       user_data_format: RAW
314       user_data:
315         str_replace:
316           params:
317             __protected_net_gw__: { get_param: vfw_private_ip_1 }
318             __unprotected_net__: { get_param:
    unprotected_private_net_cidr }
319             __repo_url_blob__ : { get_param: repo_url_blob }
320             __repo_url_artifacts__ : { get_param: repo_url_artifacts
    }
321             __install_script_version__ : { get_param:
    install_script_version }
322             __vsn_private_ip_0__ : { get_param: vsn_private_ip_0 }
323             __vsn_private_ip_1__ : { get_param: vsn_private_ip_1 }
324             __protected_private_net_cidr__ : { get_param:
    protected_private_net_cidr }
325             __onap_private_net_cidr__ : { get_param:
    onap_private_net_cidr }
326             __cloud_env__ : { get_param: cloud_env }
327           template: |
328             #!/bin/bash
329
330             # Create configuration files
331             mkdir /opt/config
332             echo "__protected_net_gw__" > /opt/config/
    protected_net_gw.txt
333             echo "__unprotected_net__" > /opt/config/unprotected_net.
    txt
334             echo "__repo_url_blob__" > /opt/config/repo_url_blob.txt
335             echo "__install_script_version__" > /opt/config/
    install_script_version.txt
336             echo "__vsn_private_ip_0__" > /opt/config/
    vsn_private_ip_0.txt
337             echo "__vsn_private_ip_1__" > /opt/config/
    vsn_private_ip_1.txt
338             echo "__protected_private_net_cidr__" > /opt/config/
    protected_private_net_cidr.txt
339             echo "__onap_private_net_cidr__" > /opt/config/
    onap_private_net_cidr.txt
340             echo "__cloud_env__" > /opt/config/cloud_env.txt
341
342             # Download and run install script
```