Orchestration of Virtual Network Functions for 5G wireless networks

by

Alexandros-Stylianos Valantasis

Department of Electrical and Computer Engineering

UNIVERSITY OF THESSALY

VOLOS 2019, GREECE

Thesis Title:

“Orchestration of Virtual Network Functions for 5G wireless networks”

“Ενορχήστρωση εικονοποιημένων δικτυακών λειτουργιών για ασύρματα δίκτυα 5ης γενιάς”

Author:

Alexandros-Stylianos Valantasis

Supervisors:

Athanasios Korakis, Associate Professor

Antonios Argyriou, Assistant Professor

Dimitrios Katsaros, Associate Professor

Presented to the Department of Electrical and Computer Engineering of
The University of Thessaly at Volos in Partial Fulfillment
of the Requirements for the Degree of

MASTER OF SCIENCE IN SCIENCE AND TECHNOLOGY OF ELECTRICAL AND COMPUTER ENGINEERING

UNIVERSITY OF THESSALY

OCTOBER 2019

Declaration of Authorship

I, Alexandros-Stylianos Valantasis, hereby certify that this thesis titled, “Orchestration of Virtual Network Functions for 5G wireless networks” has been composed by me and is based on my work unless stated otherwise.

The research was carried out wholly during the candidacy for the degree of Master of Science in Science and Technology Electrical and Computer Engineering at the University of Thessaly, Department of Electrical and Computer Engineering, Volos, Greece.

I have documented every source and material I have quoted or consulted in the References section.

Dedicated to my Family & Friends

Ευχαριστίες

Με την περάτωση αυτής της εργασίας, θα ήθελα να ευχαριστήσω τον επιβλέποντα αναπληρωτή καθηγητή του τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, κ. Αθανάσιο Κοράκη για την εμπιστοσύνη που έδειξε στο πρόσωπό μου αλλά και την πίστη του στις δυνατότητές μου, προσφέροντας μου την ευκαιρία να ασχοληθώ με το συγκεκριμένο θέμα και να αποκτήσω ουσιαστικά εφόδια μέσα από αυτήν τη διαδικασία.
Επίσης, θα ήθελα να ευχαριστήσω τους εργαζόμενους στο Εθνικό Κέντρο Έρευνας και Τεχνολογικής Ανάπτυξης (ΕΚΕΤΑ) Βόλου, οι οποίοι με βοήθησαν να ξεπεράσω τυχόν δυσκολίες που προέκυψαν.
Ιδιαίτερη ευχαριστία θα ήθελα να απευθύνω στους κ. Νικόλαο Μακρή και Χρήστο Ζαραφέτα για τις ουσιώδεις υποδείξεις και παρεμβάσεις, καθώς και για την καθημερινή τους ενασχόληση και καθοδήγηση που βοήθησαν σε μεγάλο βαθμό στην εκπόνηση της παρούσας εργασίας.
Τέλος, ένα μεγάλο ευχαριστώ στην οικογένεια μου και τους φίλους μου, στους ανθρώπους που με τους όποιους πέρασα χαρούμενες και λυπημένες στιγμές και μου παρείχαν ουσιαστική υποστήριξη σε αυτό το ταξίδι της γνώσης μέχρι και το τέλος.

Περίληψη

Η εικονικοποίηση δικτυακών λειτουργιών (NFV) είναι μια αρχιτεκτονική δικτύου για την εικονικοποίηση διαφορετικών υπηρεσιών δικτύου οι οποίες παραδοσιακά υλοποιούνται πάνω σε ιδιόκτητο και αποκλειστικό φυσικό υλικό. Ειδικότερα, η συγκεκριμένη αρχιτεκτονική αποσκοπεί στην αύξηση της ευελιξίας, της συνολικής βιωσιμότητας και της διαχείρισης της υποδομής που χρησιμοποιούν οι φορείς εκμετάλλευσης δικτύων για την παροχή υπηρεσιών. Αυτή η τεχνολογικά προηγμένη αρχιτεκτονική δικτύου θεωρείται ότι αποτελεί τον βασικό παράγοντα για την εξέλιξη της τεχνολογίας 5G, καθώς δίνει την ευκαιρία να αναπτυχθούν νέες υπηρεσίες δικτύου (NFs) με ένα μόνο κλικ.

Για τις ανάγκες διαχείρισης και ενορχήστρωσης των υπηρεσιών δικτύου, δημιουργήθηκε μια νέα αρχιτεκτονική, η NFV-MANO. Η αρχιτεκτονική αυτή είναι το βασικό συστατικό εργαλείο που αναπτύχθηκε από την Ομάδα Προδιαγραφών του Ευρωπαϊκού Ινστιτούτου Τηλεπικοινωνιακών Προτύπων (ETSI) για το NFV (ETSI ISG NFV) με σκοπό την ενορχήστρωση διαφορετικού τύπου εικονικοποιημένων πόρων.

Παρόλο που το NFV-MANO θεωρείται ότι είναι ο κύριος εκφραστής για την ανάπτυξη υπηρεσιών σε data-centers, δεν υπάρχει η δυνατότητα ενσωμάτωσης και διαχείρισης ασύρματων δικτύων στις ενοποιημένες υπηρεσίες που παρέχει.

Για το λόγο αυτό, σε αυτή τη διατριβή επικεντρωνόμαστε στην ενσωμάτωση της ασύρματης δικτύωσης ως ένα χαρακτηριστικό του εργαλείου NFV-MANO για την παροχή εικονοποιημένων υπηρεσιών πάνω από φυσικό εξοπλισμό. Ειδικότερα, χρησιμοποιούμε το εργαλείο Open Source Mano για την ανάπτυξη εικονοποιημένων διαδικτυακων λειτουργιών (VNFs) σε συνδυασμό με ασύρματες τεχνολογίες όπως LTE, WiFi και mmWave, που παρέχονται από τον διαχειριστή εικονικής υποδομής (VIM) του ασύρματου πειραματικού περιβαλλοντος NITOS.

Abstract

Network functions virtualization (NFV) is an initiative to virtualize network services traditionally run on proprietary, dedicated hardware. Especially aims at adding up to the flexibility, overall sustainability, and management of the infrastructure that network operators use to provide services. This state of the art softwarized network architecture considers being the key enabler of 5G technology as it gives the opportunity to deploy Network Functions (NFs) with a single click.

For the needs of management and orchestration of Network Functions, a new architecture has emerged, namely the NFV-MANO. NFV MANO is this key component framework developed by the European Telecommunications Standards Institute (ETSI) Industry Specification Group for NFV (ETSI ISG NFV) with the purpose of orchestrating different type of virtualized resources.

Although NFV-MANO considered being the key enabler for the deployment of services over data-centers, it lacks to introduce a softawarized use case for the wireless part of the network.

For this reason, in this thesis, we focus on integrating wireless networking as a feature of an NFV-MANO-compliant orchestrator for provisioning virtual services on top of the physical equipment. Especially, we use the Open Source MANO framework for the deployment of Virtual Network Functions (VNFs) combined with wireless links, through LTE, WiFi and mmWave technologies, provided by the Virtual Infrastructure Manager (VIM) of the NITOS wireless testbed.

Table of Contents

List of Figures & Tables

1

Introduction

Network functions virtualization or NFV is the state of the art network architecture that considers being a key part for the development of the 5G technology. Especially, provides a new way to create, distribute, and operate networking services, as it aims to decouple network functions from proprietary hardware in order to run as software on standardized hardware. These functions (such as firewall, deep packet inspection, and intrusion prevention) characterized as virtual network functions (VNF). In order to manage and orchestrate these Virtual Network Functions, a new framework has introduced, the NFV MANO. In this thesis, we take advantage of NFV-MANO's compliant orchestrator (OSM) capabilities in order to extend the specific software so as to support the integration with wireless networks such as WiFi, LTE, and mmWave.

1.1 Organization

The rest of this dissertation is organized as follows:

In *chapter 2*, we present basic background information about the Network Function Virtualization while we emphasize in the NFV management and orchestration standard. Also, *chapter 2* introduces NITOS testbed, the facility that hosted our experiments.

In *chapter 3*, details about the implementation and design of the whole architecture are provided. Specifically, *chapter 3* presents the architecture of provisioning the testbed with the NFV-MANO paradigm and how the integration of wireless technologies occurred. Moreover, *chapter 3* presents the testbed integration, namely the integration of the orchestrator with the existing NITOS tools for provisioning the testbed to experimenters.

Finally, in the last *chapter 4*, we summarize and conclude this thesis and propose directions for future research.

2

Background

2.1 Network Function Virtualization (NFV)

Network functions virtualization or NFV is a network architecture concept that uses the technologies of IT virtualization to virtualize entire classes of network node functions into building blocks that may connect, or chain together, to create communication services. This innovative approach aims to completely decouple the network resources and services from the underlying physical infrastructure in order to transform the traditional way networks are built or configured and services are delivered. With NFV, any enterprise can simplify a wide array of network functions, as well as maximize efficiencies and introduce new revenue-generating services faster and easier than ever before.[1]

In the past, data centers, mobile operators and enterprises have built their network infrastructure mainly based on custom-designed, physical hardware and software. Some examples of these applications could be network gateways, switches, routers, network load balancers, varied mobile applications in the mobile core and radio access network such as vEPC (virtual evolved packet core), vCPE (virtual customer premise equipment), vRAN (virtual Radio Access Network) and security applications like firewalls, NGFW, IDS/IPS, SSL/IPsec offload appliances, DLP and antivirus applications, to name just a few.[2]

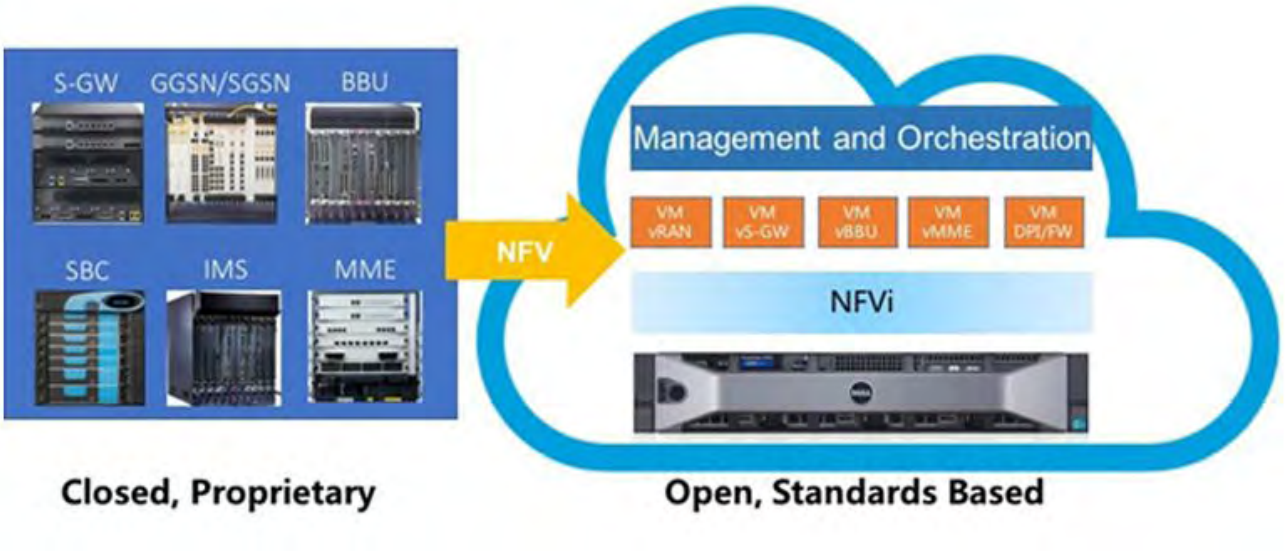

Figure 2.1: NFV Concept

Nowadays, NFV is intended to unite and convey the networking components needed to support an infrastructure absolutely free from hardware. These components include virtual compute, storage and network functions. NFV utilizes standard IT virtualization technologies that run on off-the-shelf hardware like commodity x86 servers. It is applicable to any data plane processing or control plane function in both wired and wireless network infrastructures.[2]

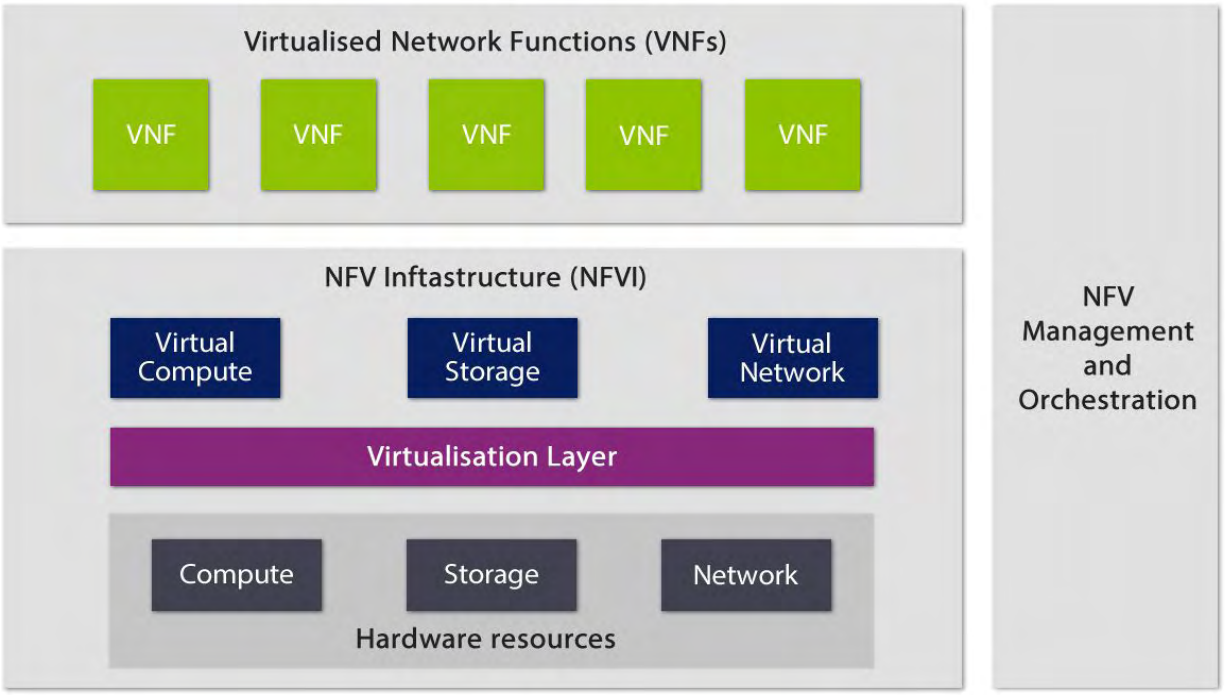

Figure 2.2: High-Level NFV Architecture

Generally, the idea of NFV began from service providers who needed to make adding new network functions or applications simpler and quicker. The requirements of hardware-based appliances led them to apply standard IT virtualization technologies to their networks. To accelerate progress towards this common goal, the companies who had formed the European Telecommunications Standards Institute (ETSI) devoted resources to a new group, the Industry Specification Group for Network Functions Virtualization (ISG NFV) in January 2013.

The ETSI ISG NFV is a group charged with developing requirements and architecture for virtualization of various functions within telecoms networks, such as standards like NFV management and network orchestration (MANO). ETSI is also instrumental in collaborative projects like the newly announced OPNFV.

2.1.1 Architecture of NFV

As we mentioned above, the European Telecommunications Standards Institute (ETSI) created the Network Functions Virtualization (NFV) framework which describes standards to decouple network functions from proprietary hardware appliances and instead, run them in software on standard hardware. This framework describes how to manage and orchestrate VNFs.

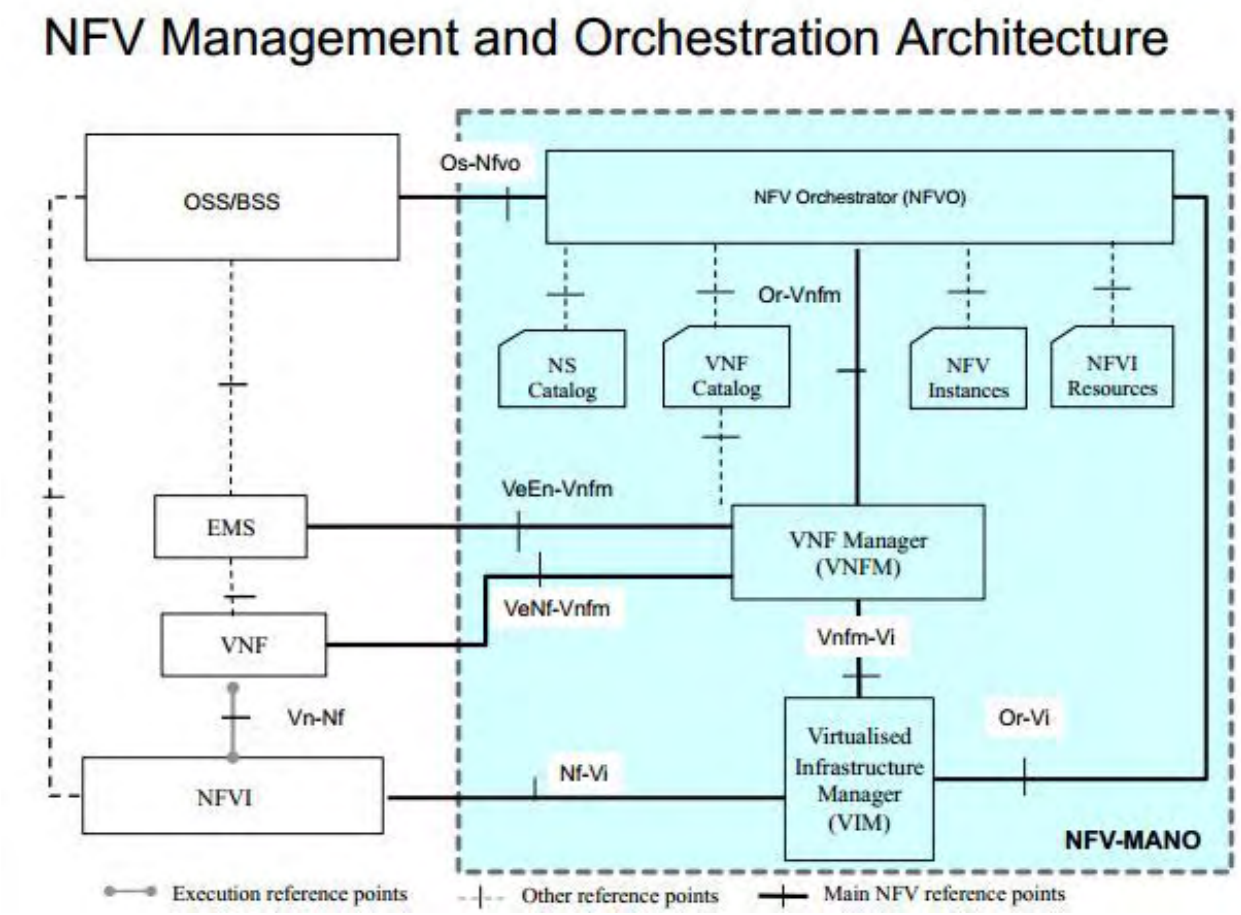

Figure 2.3: NFV Architecture & Functional Blocks

The NFV architecture consists of a plethora of interconnected components. As we can notice in the figure above, the most important are the **Virtual Network Functions (VNF)**, the **NFV Infrastructure (NFVI)**, the **NFV Management and Orchestration(NFV-MANO),** the **EMS** and **OSS/BSS.**

- **Virtual Network Functions (VNF)** is the basic block in NFV Architecture. In addition, VNFs are software implementations of network functions that can be deployed on an **NFVI**. These entities run on virtual machines on top of the physical infrastructure.[2]

- The **NFV Infrastructure (NFVI)** is the whole hardware and software we need to create a platform where we can run VNFs on. The NFVI works together with the VNFs and the Virtual Infrastructure Managers (VIMs) to interconnect and support the resources contained in the NFVI-PoPs, in order to provide the required abstraction of the underlying network infrastructure.[2] The NFV Infrastructure includes hardware resources, virtualization layer, and virtual resources.

    - As hardware resources, we refer to computing, storage, and network that provides processing, storage, and connectivity to VNFs through virtualization (hypervisor) layer. Computing and storage resources are commonly used in a pool. The network resource comprises of switching functions e.g. router, wired or wireless network.[2]

    - Virtualization layer also known as a hypervisor, it abstracts the hardware resources and decouples the VNF software from the underlying hardware to ensure a hardware-independent life cycle for VNFs

    - As virtualization resources, we refer to virtual compute, storage and network that are abstracted from the hardware layer through the virtualization layer.

- The **OSS** is a software component that enables a service provider to monitor, control, analyze, and manage the services on its network. These types of software applications, along with a business support system (**BSS**), support most customer-facing activities, including ordering, billing, and support.

- **Element Management System (EMS)** is responsible for the functional management of VNF. The management functions include Fault, Configuration, Accounting, Performance, and Security Management. An EMS may manage the VNFs through proprietary interfaces. There may be one EMS per VNF or one EMS that can manage multiple VNFs. EMS itself can be deployed as Virtual Network Function (VNF).

- **NFV Management and Orchestration (NFV MANO)** is a key component that manages and orchestrates the lifecycle of VNFs. In addition, MANO interacts with both NFVI and VNF layers in order to manage all the resources in the infrastructure layer. We will further expand on MANO, with particular emphasis on the ETSI MANO specification.

2.2 NFV MANO Standard

As aforementioned, NFV MANO is the key component framework developed by the European Telecommunications Standards Institute (ETSI) Industry Specification Group for NFV (ETSI ISG NFV). Over time, the framework became more commonly referred to as just NFV management and orchestration. It is the ETSI-defined framework for the management and orchestration of all resources in a virtualized data center including compute, networking, storage, and virtual machine (VM) resources. The main focus of NFV MANO is to allow flexible on-boarding, sidestepping the chaos that can be associated with the rapid spin-up of network components.

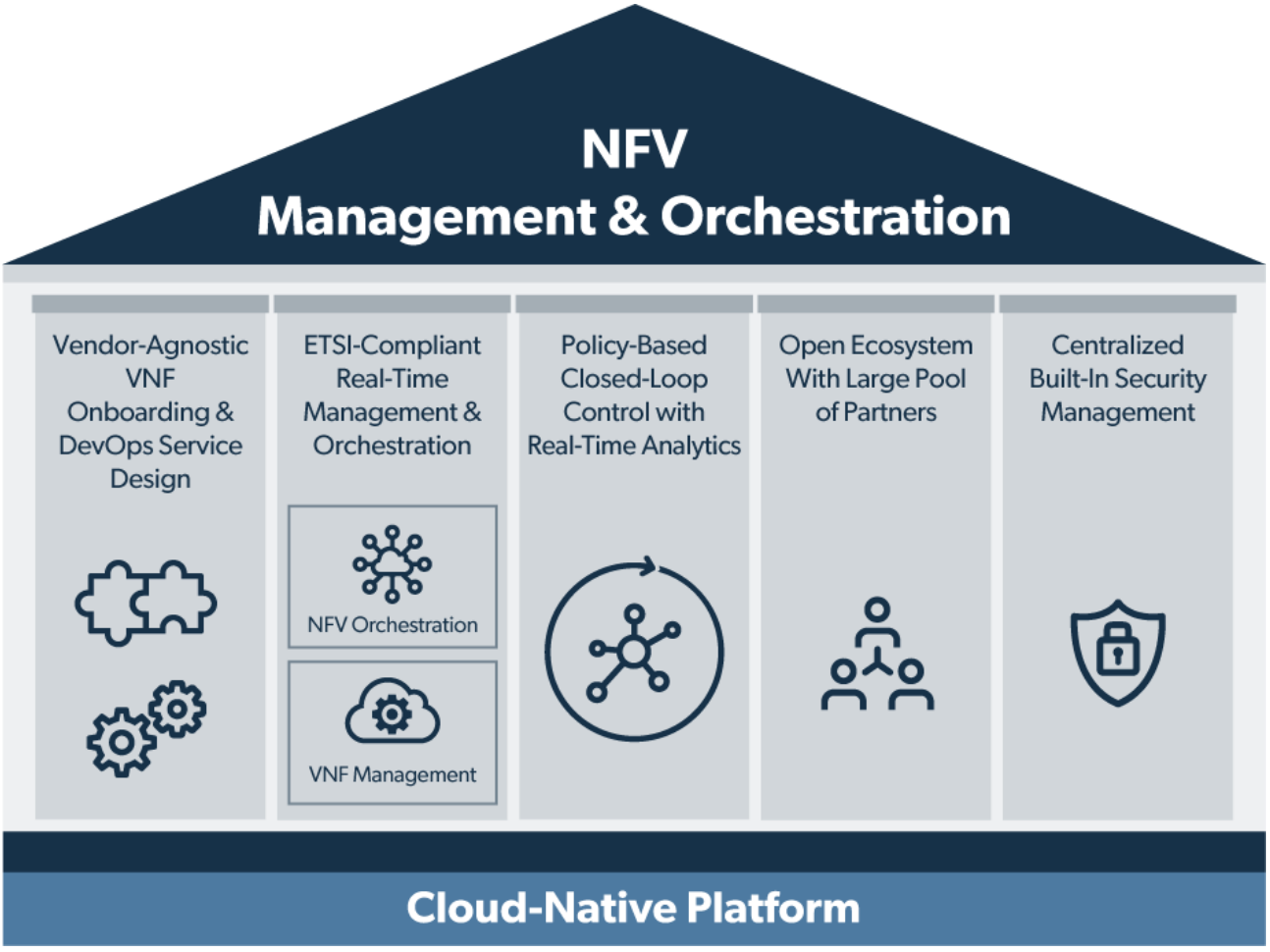

Figure 2.4: NFV-MANO

2.2.1 Architectural Framework & Functional Blocks

NFV-MANO includes a plethora of functional blocks, all of these have a set of responsibilities. Each one of those applies management and orchestration operations on well-defined entities, leveraging the services offered by the other functional blocks.

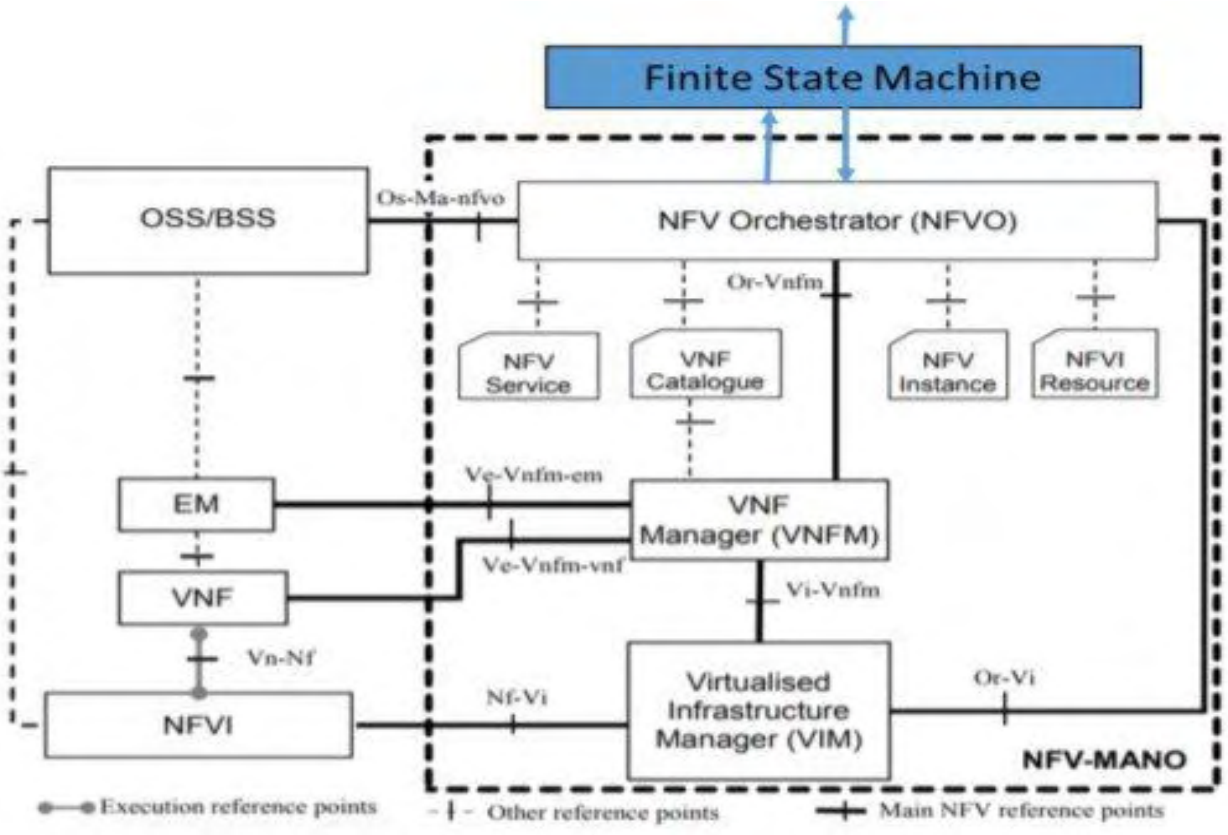

Figure 2.5: NFV-MANO Architecture & Functional Blocks

As we can see from the figure above the NFV-MANO framework is broken up into three functional blocks which are the **Virtualised Infrastructure Manager (VIM),** the **NFV Orchestrator (NFVO),** the **VNF Manager (VNFM)** and the interconnections among them.

2.2.1.1 NFV Orchestrator (NFVO)

NFVO is the component that is responsible for the orchestration and the on-boarding of new resources across multiple VIMs. These resources could be new network services (NS) and virtual network function (VNF) packages. In addition, it is responsible not only for the NS lifecycle and global resource management but also for the validation and authorization of network functions virtualization infrastructure (NFVI) resource requests.[4]

NFVO has two key responsibilities. The first one is the Resource Orchestration and the other is the Network Service Orchestration.

- **Resource Orchestration:** Consists of a variety of actions that manage the resources of NFVI, such as compute, storage and network resources. This functionality used to abstract access to the resources provided by an NFVI to services, avoiding them from depending on any VIM. In addition, the key features of resource orchestration could be[2]:
    - The validation and authorization of NFVI resource requests from the VNF manager.
    - The distribution, reservation, and allocation of NFVI resources to NS and VNF instances.
    - The management of the relationship between a VNF instance and the NFVI resources allocated to it, using NFVI Resources repositories.
    - The collection of information by VNF instances.

- **Network Service Orchestration:** This function uses the services of VNF Manager and Resource Orchestration to provide a plethora of capabilities such as[2] :
    - Management of NS and VNFs including the on-boarding of new Network Services and VNF Packages.
    - Handling the Network Service’s instance lifecycle through operations like updating, querying, scaling and terminating an NS.

- Supervision of VNF manager’s instantiation.
- Supervision of VNFs instantiation.
- Supervision of VNF Forwarding Graphs.
- The board of the trustworthiness and perceivability of the Network Service cases through their lifecycle; the NFVO additionally deals with the connection between the Network Service cases and the VNF examples.
- Computerized the executives of Network Service examples, utilizing triggers to consequently execute operational administration activities for NS and VNF occurrences, adhering to the directions caught in the on-boarded NS and VNF arrangement layouts.

2.2.1.2 Virtualized Infrastructure Manager (VIM)

VIM is the NFV-MANO component in charge of controlling and dealing with the resources contained in an NFV infrastructure such as compute, storage and network. The VIM has some expertise not only in taking care of a specific kind of NFVI resources (only for compute or storage or network), but also it be able to deal with numerous kinds of NFVI resources on the double, providing a northbound interface to different capacities to oversee them. Some of the functionalities exposed through their northbound interfaces are[3]:

- Keeping an inventory of the allocation of virtual resources to physical resources. This allows for the orchestration of the allocation, upgrade, release, and reclamation of NFVI resources and the optimization of their use.[2]
- Square underpins the administration of VNF forwarding graphs by creating virtuals links, networks, subnets, and ports but also oversees security group policies to guarantee access control.

- Management of software images, permitting other NFV-MANO blocks like the NFVO to include, erase, update, question or duplicate an image from the pool where the images been kept up by the VIM.[2]
- Validating the capacity of virtualized resources, including the forwarding of information related to NFVI resources capacity and their usage.

- Gathering the performance details and fault information from the virtual resources in order to provide them to the other NFV-MANO blocks.

- Administration of resource's catalogs, in order to be provided as configurations to the NFVI.

2.2.1.3 VNF Manager (VNFM)

The VNF Manager is an NFV-MANO block that fulfills the role of Management and Orchestration of VNFs through the life cycle management of their instances. A VNFM can be assigned to either a single VNF instance or multiple. Also, the managed VNFs can be of the same or different types, as VNFMs can be applied to any VNF(s) because of their typically generic[2].

Like the other NFV-MANO blocks, VNFM provides functionalities to other elements of the NFV-MANO architecture. Some of them are:

- VNF's instantiation and configuration using a VNF template

- Manage the software inside a VNF instance

- Controlling the state of a VNF instance and handle procedures such as the scaling out/in and up/down of instances.

- Gather performance measurement results, faults, and events correlated with its VNF instances.

- Self or automated administration of VNF instances

- Management and verification of VNFs through their lifecycle.

- Manage the configuration and event reporting between VIM and the EM.

The VNF Manager can access the VNF package repository, the VNF catalog, where the Virtualized Network Function Descriptors (VNFD) are stored[2]. The VNFD is a critical aspect of NFV scope as it can define the operational behavior of a VNF and specifies how it should be deployed providing a full description of its attributes and requirements. NFV-MANO via VNFDs can deploy instances of VNFs, handle their lifecycle, and associate them to required NFVI resources.

2.2.1.4 Data Repositories

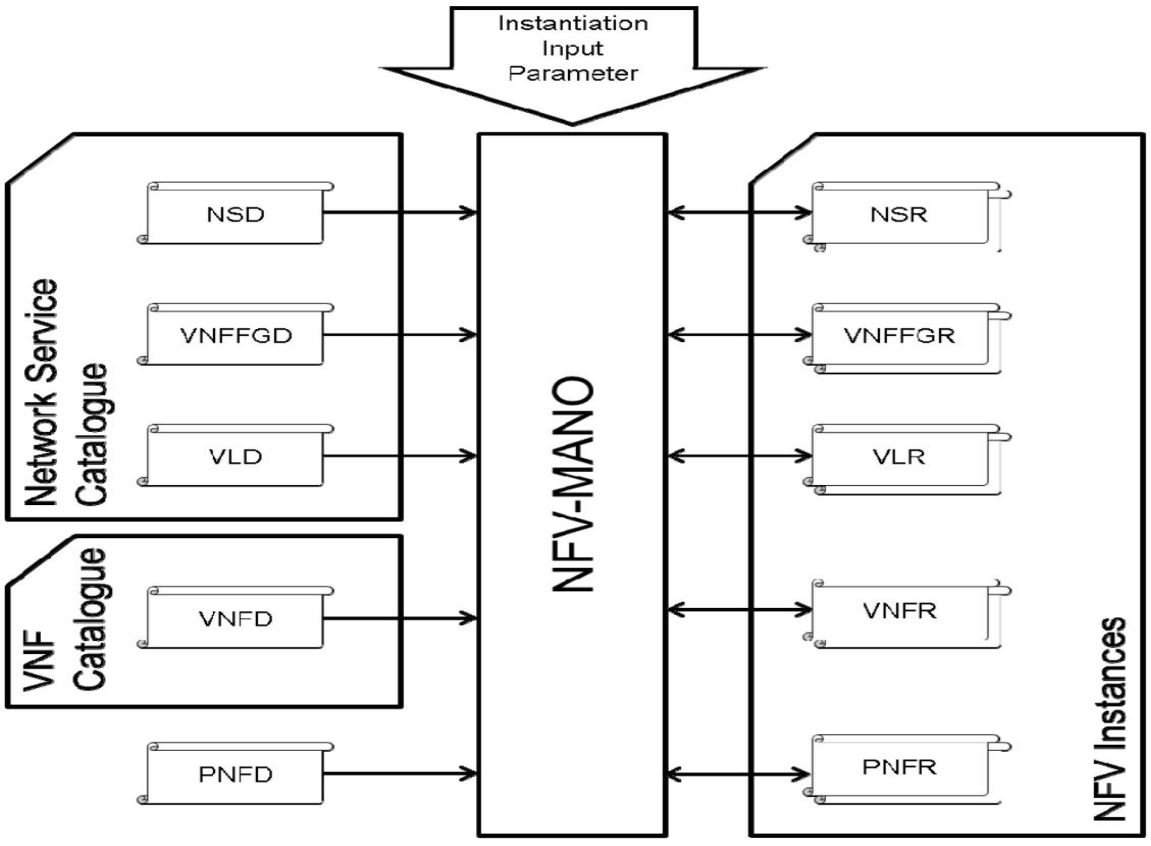

Figure 2.6: NFV-MANO Data Repositories

As we can see from the figure above the data exchanged between NFV-MANO stored in a plethora of repositories:

- The **NS Catalogue**, stores all the on-boarded and validated Network Services. As Network Services we refer to the combination of multiple interdependent VNFs in order to create a fully functional network service. The NS needs to be describable by a descriptor (NSD) that should be stored in the NS catalog. In addition, the NS Catalogue is designed to support various NS deployment templates such as virtual Link Descriptors (VLDs) and VNF Forwarding GraphDescriptor (VNFFGD).[2]

- The **VNF Catalogue** stores all on-boarded and validated VNF Packages while supporting their instantiation and management. As we mentioned above each VNF is defined in a template called Virtualized Network Function Descriptor (VNFD), stored in a VNF catalog, that corresponds to VNF Package. Both NFVO and VNFMs communicate with VNF Catalogue for the VNFDs.[2]

- The **NFV Instances repository** includes information for all the NS and VNF instances stored via their respective VNF Records (VNFs) and NS Records (NSRs). These records are refreshed during the lifetime of the instances. In addition, the NFV Instance repository supports both the NFVO's and VNFM's responsibilities in keeping up the integrity and perceivability of the instances as the connection between the Network Services and VNFs.[2]

- The **NFVI Resources repository** contains information about the accessibility and availability of NFVI resources, as preoccupied with VIMs. This corresponding information can help to reserve, allocate and monitor the resources. For that reason, the NFVI Resources repository has a critical role in the NFVO's Resource Orchestration as it can discover the NFVI resources against their associated NS and VNF instances.[2]

2.2.1.5 NFV-MANO reference points

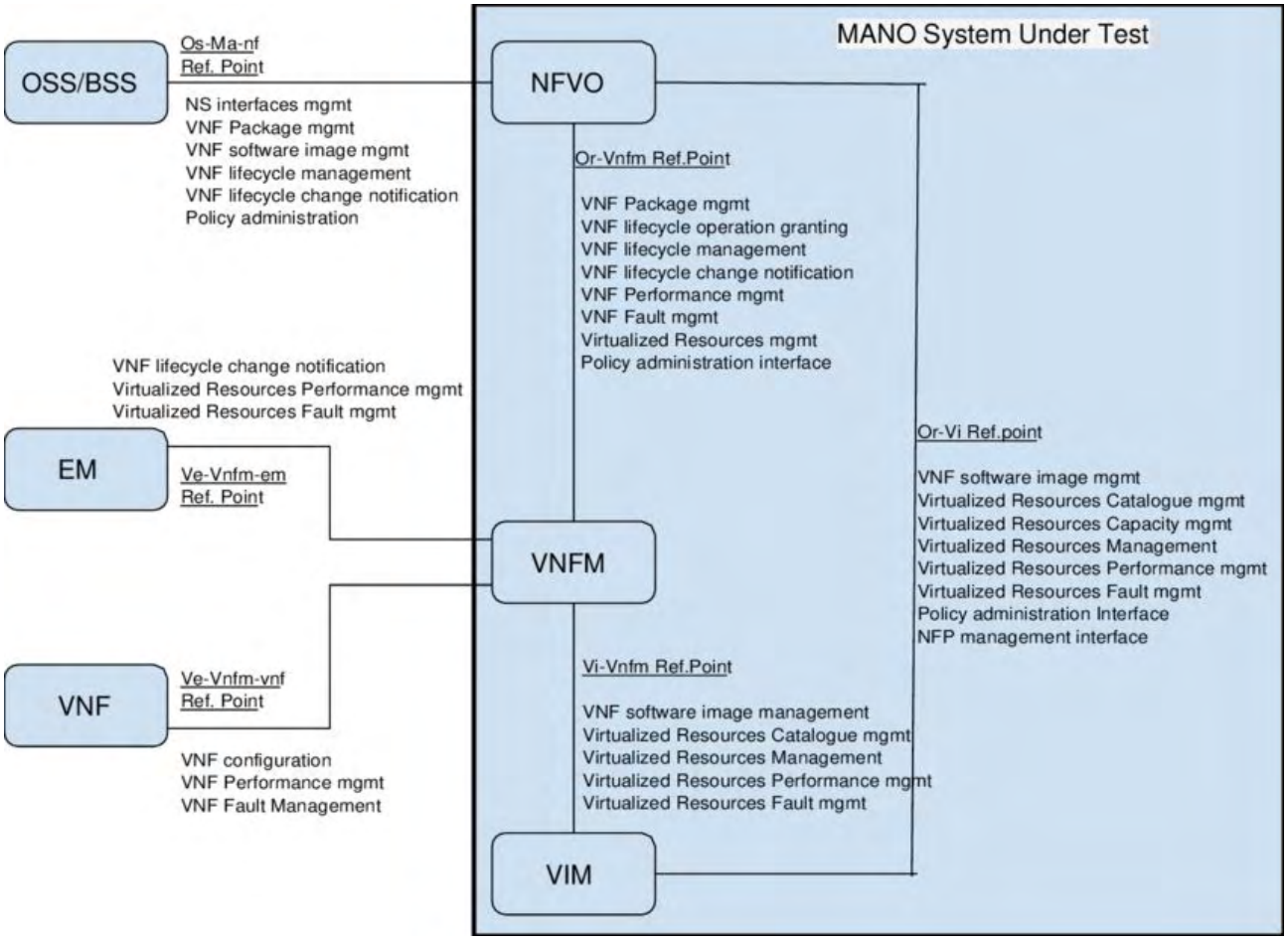

Figure 2.7: NFV-MANO reference points

As we can observe at the figure there are several reference points that connect the different NFV-MANO's blocks:

- **Os-Ma-nf** is a reference point that is responsible for the communication between OSS/BSS and NFVO blocks for the NSDs management (and VNF packages).

- **Ve-Vnfm-em** and **Ve-Vnfm-vnf**, are two reference points between a VNFMand EM or a VNF respectively, used by a VNFM for management and control of VNF

- **Nf-Vi** is a reference point used by VIM to control an NFVI, including the management configuration of Virtual Machines.

- **Or-Vnfm** is a reference point between NFVO and VNFM. It is used in order to authorize the allocation and release of resources to VNFs, instantiate VNFs and to retrieve and update information regarding VNF instances.

- **Or-Vi** is a reference point between NFVO and VIM, used by the NFVO to handle NFVI resources.

- **Vi-Vnfm** is a reference point used by VNFMs in order to retrieve and allocate NFVI's resources information from VIM.

2.2.1.6 VNF instantiation Flow

The figure below describes a typical example of VNF instantiation, especially how the internal NFV-MANO's components interact with each other and what kind of messages exchange

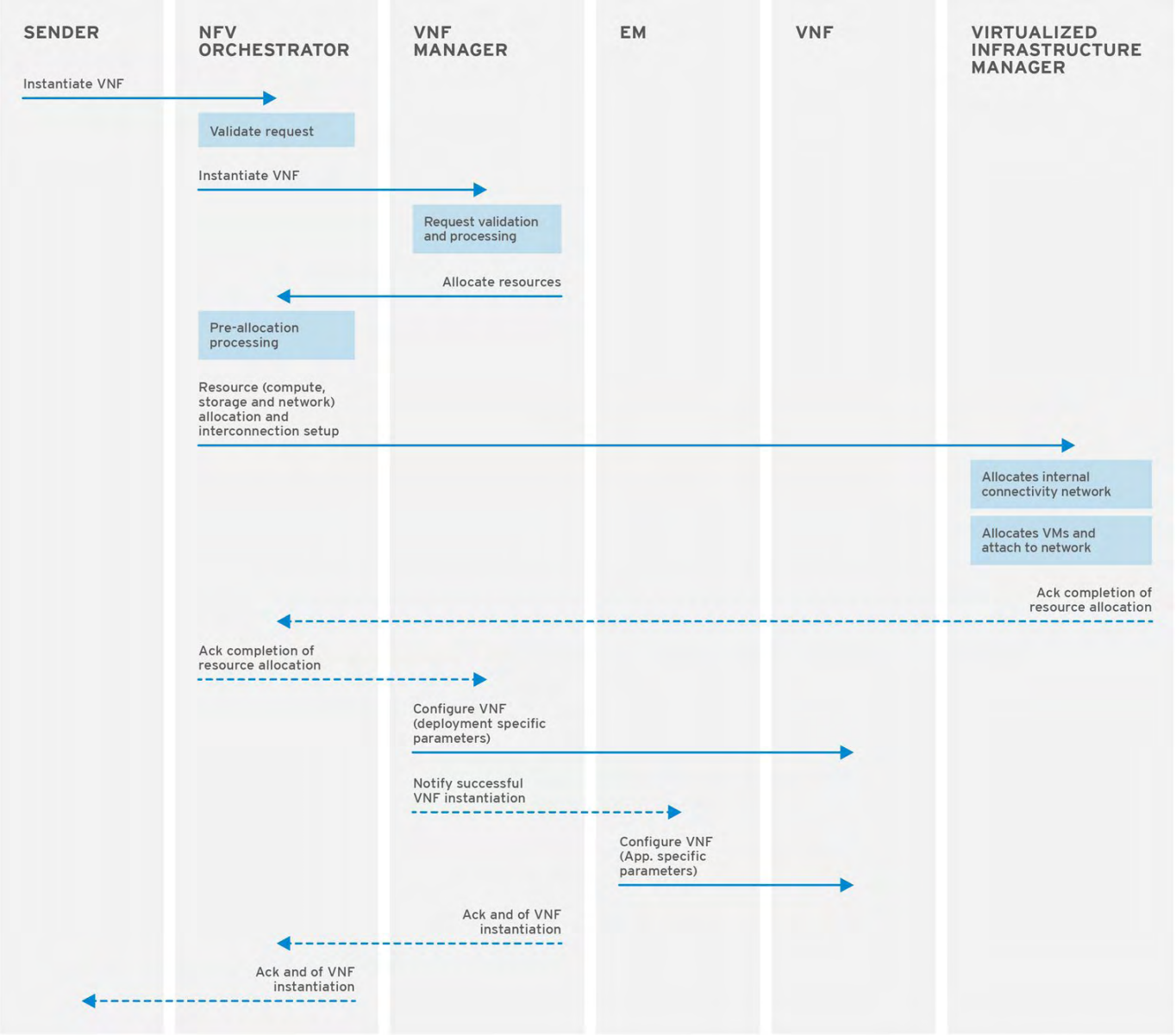

Figure 2.8: VNF instantiation Flow

The main steps for the instantiation of a VNF are:

1. A message of an **Insatiate VNF** sends to NFVO either from OSS or VNFM.

2. Tye NFVO receives the message and validates the request by making a feasibility check.

3. The NFVO sends to the VNF Manager an **Instantiate VNF** request, to instantiate the VNF with its instantiation data.

4. The VNFM validates the request and processes it. Also, includes the instantiation data with data contained in the VNFD data.

5. The NFVM goes back to VNFO to enquire for resource allocation through an **Allocate Resource** message.

6. The NFVO doing the necessary pre-allocation procedures.

7. The NFVO sends an **Allocate Resource** request to VIM, to reserve the necessary compute, storage and network resources.

8. The VIM allocates the internal connectivity network

9. The VIM interfaces with the NFVI in order to allocate the requested compute and storage resources while also to attach the instantiated VMs to the previously allocated internal connectivity network.

10. The VIM sends a completion acknowledgment response back to the NFVO

11. The NFVO makes an acknowledgment request to the VNFM about the successful completion of the allocation process.

12. The VNFM configures the VNF, using appropriate configuration operations. The EM is notified about the instantiated VNF.

13. The EM configure with its configuration parameters on the VNF.

14. The VNFM returns a response to the NFVO as an acknowledgment notifying for the successful completion of configuration.

15. The NFVO acknowledge the sender for the successful VNF instantiation.

2.2.2 NFV-MANO OpenSource Tools

In this section, we are going to provide a brief review of the available OpenSource tools for the NFV-MANO architecture.

As we can observe from the figure below there are a plethora of OpenSource Network Function Orchestration platforms.

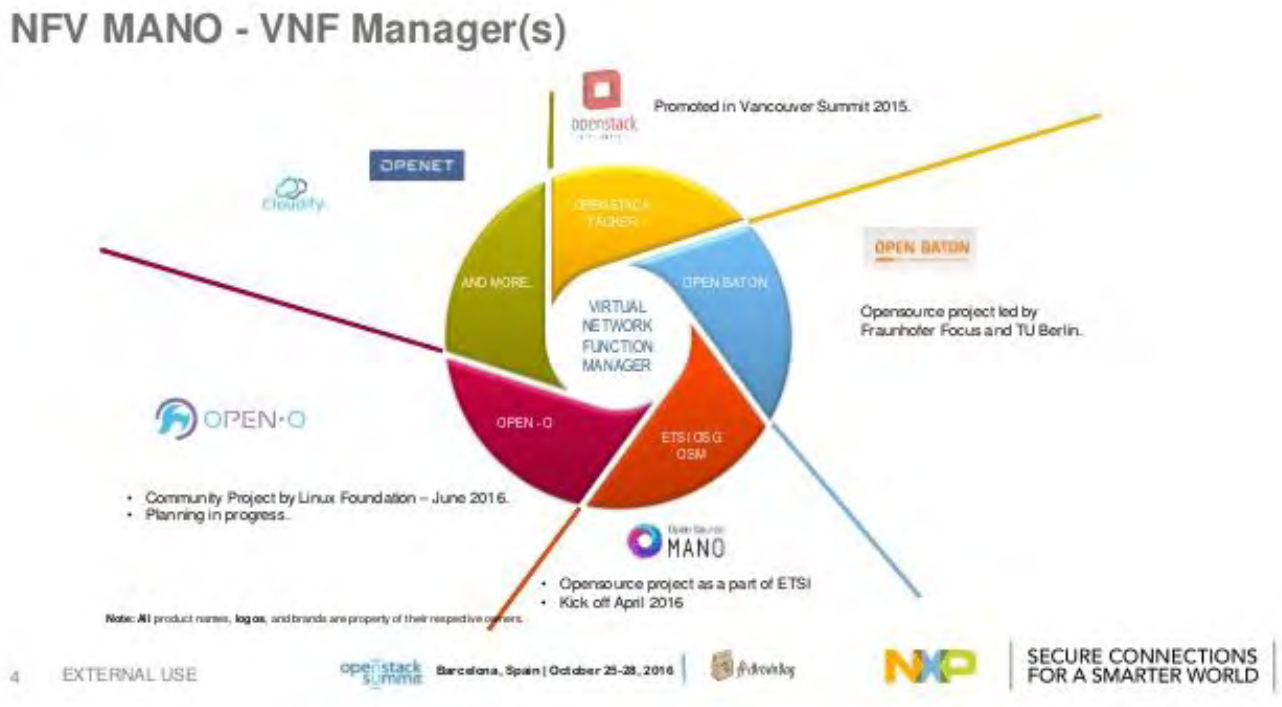

Figure 2.9: Open Source NFV-MANO tools

OSM

The OpenSource MANO [6] project was authoritatively propelled at the World Mobile Congress (WMC) in 2016. Beginning with a few establishing individuals, including Mirantis, Telefónica, BT, Canonical, Intel, RIFT.io, Telekom Austria Group, and Telenor, the OSM people group presently incorporates with 55 unique organizations. The OSM project is facilitated at ETSI offices and targets conveying an open-source the executives and coordination (MANO) stack firmly lined up with the ETSI NFV reference design.[5]

OSM issued two releases, Rel 0 and Rel 1, during 2016. The most recent at the time of this writing, OSM Rel. 6, has been publicly available since May 2019.OSM tries to be compliant with the ETSI NFV MANO reference architecture. It consists of 3 basic components. First the SO, the Service Orchestrator (SO), responsible for end-to-end service orchestration and provisioning where VNF definitions and NS catalogs are stored. Second, the Resource Orchestrator (RO) which is used to provide services over a particular IaaS provider in a given location. And finally, the VNF Configuration and Abstraction (VCA) module perform the initial VNF configuration using Juju Charms.[5]

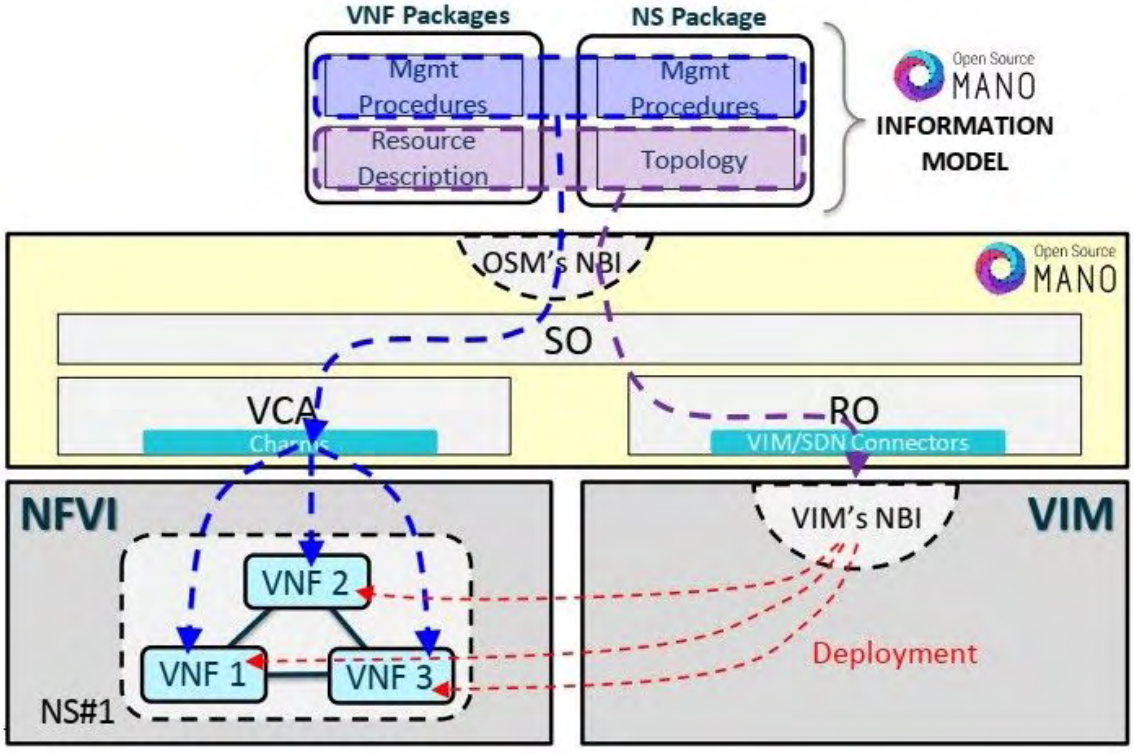

Figure 2.10: Open Source MANO Architecture

ONAP

The Open Network Automation Platform (ONAP) [10] is an initiative created by the combination of the ECOMP (Enhanced Control, Orchestration, Management & Policy) and Open-O (Open Orchestrator) projects into ONAP, to bring the capabilities for designing, creating, orchestrating and handling of the full lifecycle management of Virtual Network Functions, Software Defined Networks, and the services that all of these things entail.

In essence, ONAP is the platform above the infrastructure layer that automates the network. ONAP allows the end-users to connect products and services through the infrastructure, and allows the deployment of VNFs and scaling of the network, in a fully automated manner. The high-level architecture of ONAP contains different software subsystems that are part of a design-time environment, as well as an execution time environment to execute the designed platforms. The project is under the governance of the Linux Foundation and founded by AT&T and China Mobile. This has brought together many of the key players in the industry, like Vodafone, Orange, Bell Canada, Ericsson, Reliance JIO, Nokia and more.

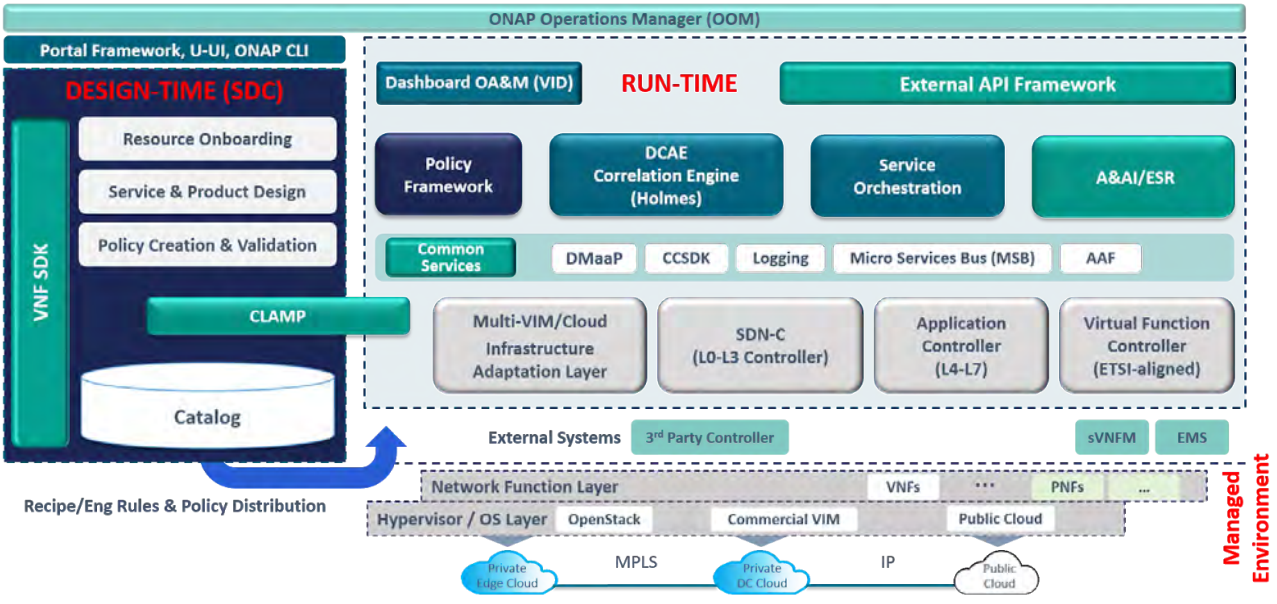

Figure 2.15: ONAP Architecture

OPEN-O

The OPEN-O project is facilitated by the Linux establishment and was likewise officially declared at 2016 MWC. Beginning task supporters were for the most part from Asian organizations, for example, Huawei, ZTE and China Mobile. In the long run, the project got further help from Brocade, Ericsson, GigaSpaces, Intel and others.[5]

The principle project objective is to enable end-to-end service agility across multiple domains using a unified platform for NFV and SDN orchestration. The OPEN-O project presented its first version in November 2016. Overall project governance is managed by the project Board, with technology-specific issues managed by the Technical Steering Committee

In addition, the OPEN-O project, by and large, have no goal to be consistent with NFV MANO architecture, as the NFVO functional block of OPEN-O is aligned with an ETSI reference model, and all key MANO components, for example, VNFM and VIM may be found in an NFVO architecture. A critical task that should take into account is the ability to support SDN-based networking services provisioning and orchestration which now utilized either in conjunction with NFV services or as an independent component.[5]

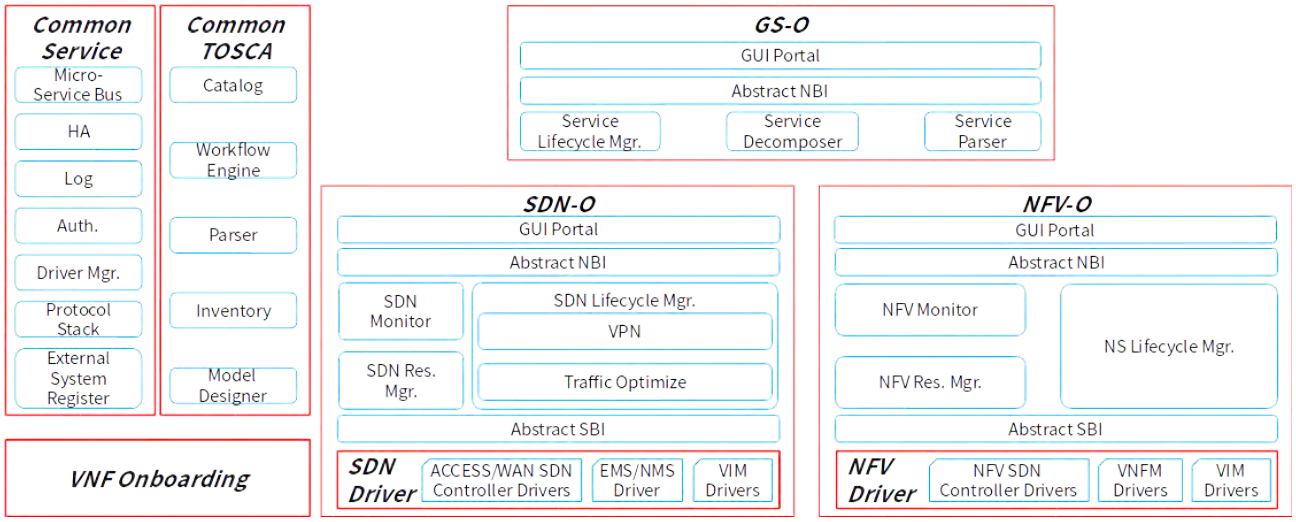

Figure 2.11: OPEN-O Architecture

As we can observe from the figure above the OPEN-O project consists of several functional blocks such as Common service, Common TOSCA, Global Service Orchestrator (GSO), SDN Orchestrator (SDN-O) and NFV Orchestrator (NFV-O).

Rope/XOS

Initially CORD [8] (Central Office Re-architected as a Datacenter) was presented as one of the use cases for the ONOS SDN Controller, however, it grew up into a separate project under ON.Lab administration.

The primary goal of the project was not only to associate the NFV and SDN concepts but also to bring datacenter economics and cloud agility to the Telco Central Office. The reference implementation of CORD combines commodity servers, white-box switches, and disaggregated access technologies with open source software to provide an extensible service delivery platform. CORD Rel.1 and Rel.2 integrate a number of open source projects, such as ONOS to manage SDN infrastructure, OpenStack to deploy NFV workloads and XOS as a service orchestrator. To reflect the use cases' uniqueness, CORD introduces a number of service profiles, such as Mobile (M-CORD), Residential (R-CORD), and Enterprise (E-CORD).[5]

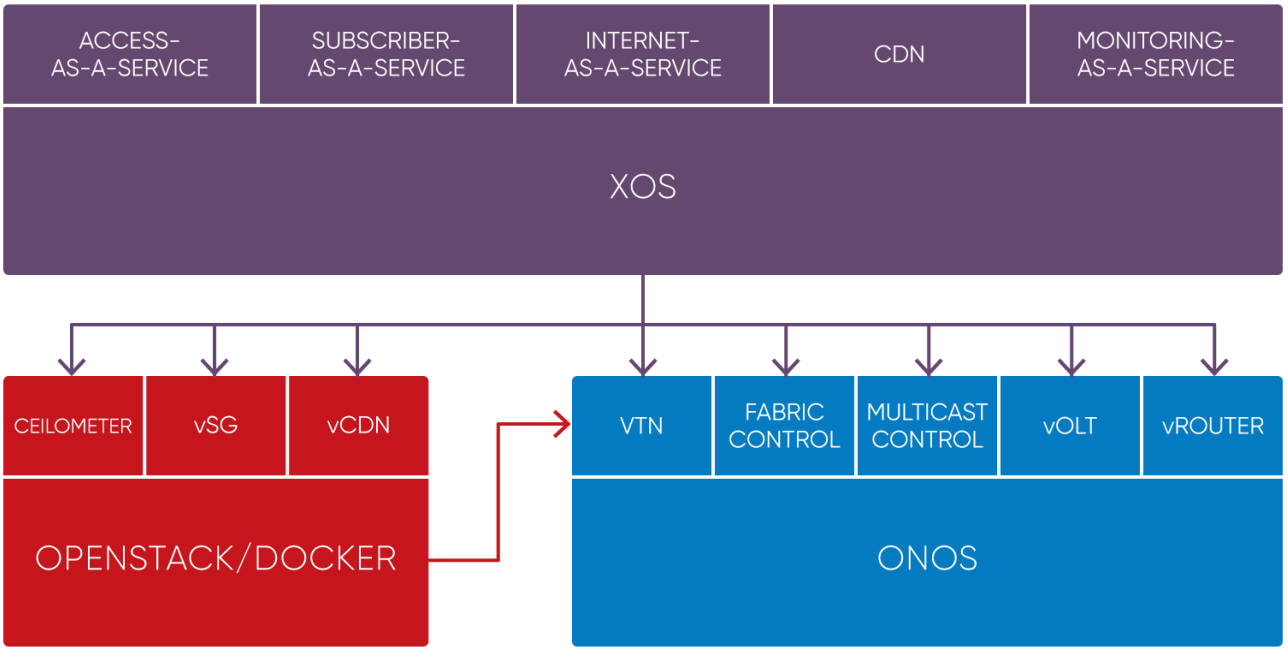

Figure 2.12: Rope/XOS Architecture

Gigaspaces Cloudify

Gigaspaces' Cloudify [9] is the open-source TOSCA-based cloud arrangement programming stage. Initially presented as an unadulterated cloud arrangement (like OpenStack HEAT), the stage was additionally extended to incorporate NFV-related use cases, and the Cloudify Telecom Edition rose.

Considering its unique stage reason, Cloudify has an extensible design and can cooperate with numerous IaaS/PaaS suppliers, for example, AWS, OpenStack, Microsoft Azure, etc. By and large, Cloudify programming is open source under the Apache 2 permit and the source code is facilitated in an open archive. While the Cloudify stage is open source and invites network commitments, the general undertaking guide is characterized by Gigaspaces.[5]

The original focus of the Cloudify platform was orchestration of application deployment in a cloud. Later, when the NFV use case emerged, the Telecom Edition of the Cloudify platform was delivered. This platform combines both NFVO and generic VNFM components of the MANO defined entities.[5]

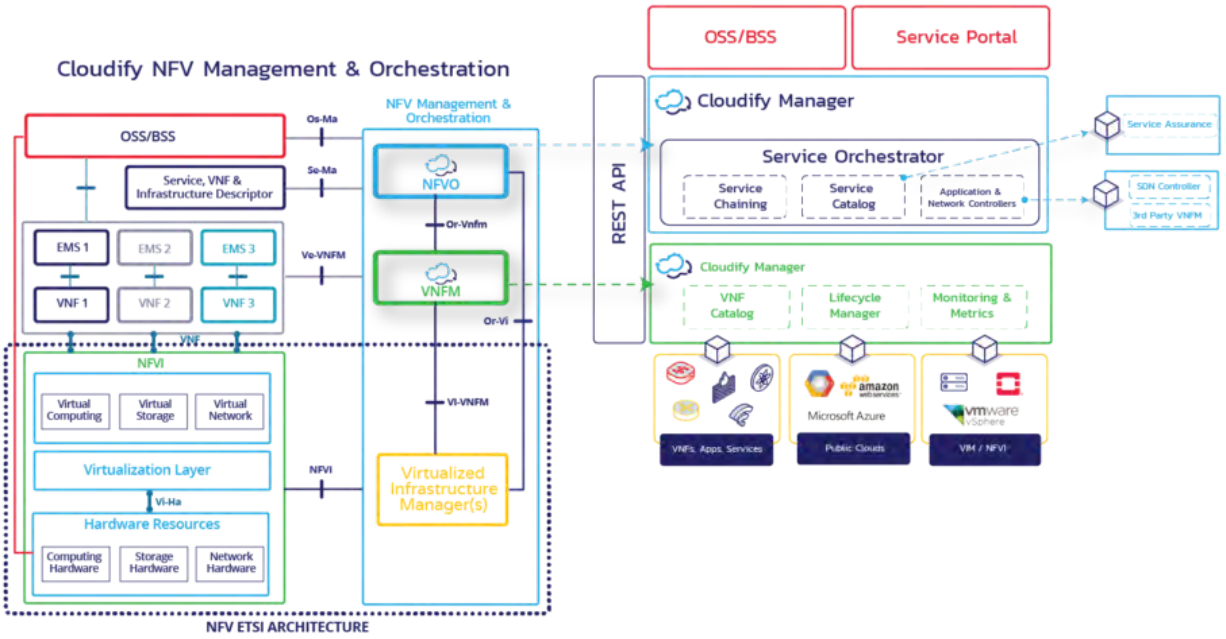

Figure 2.13: Gigaspaces Cloudify Architecture

Cloudify Agents are used to managing workflow execution via an appropriate plugin

From the high-level perspective, Cloudify platform consists of a plethora of functional blocks:

- Cloudify Manager, which is the orchestrator that manages the deployment and the lifecycle of the services.
- Cloudify Agents, which are used to manage workflow execution via an appropriate plugin
- Elasticsearch, utilized as an information store component of the deployment procedure
- Logstash, which is used in order to gather log information from platform components
- Riemann used as a policy engine to process runtime decisions about availability
- RabbitMQ used as an async transport for communication among all platform components

Open Baton

Open Baton [7] is an open-source project, launched by TU Berlin and Fraunhofer FOKUS, which aims to provide a complete NFV orchestration framework based on the ETSI NFV-MANO and TOSCA standards. Open Baton allows the creation of complete NFV environments, providing the main functional blocks required by the specification:

- NFV Orchestrator (NFVO), responsible for the orchestration of resources
- VNFM for the management of VNFs

- Scaling Engine, in order to manage the scaling operations
- EMS, located inside the VNFC in order to ensure a link with VNFM.
- Plugin mechanism in order to support different type of VIMs

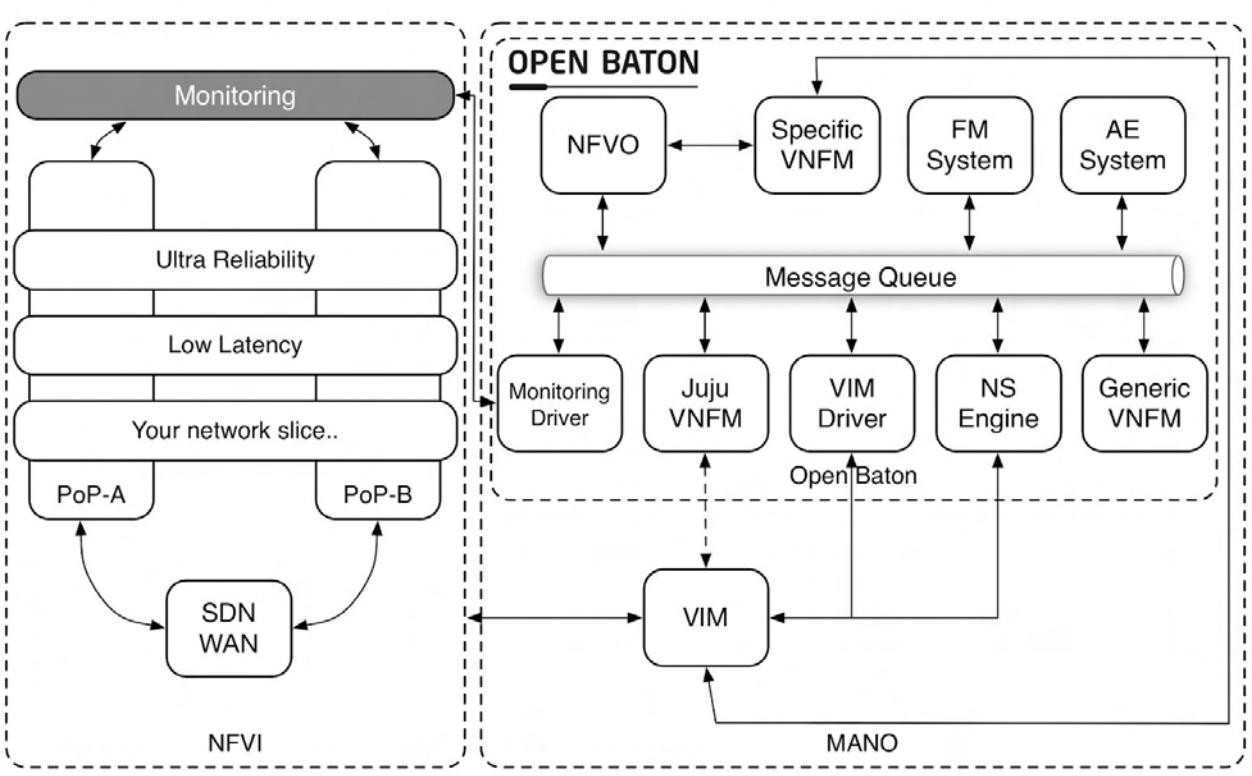

Figure 2.14: Open Baton Architecture

In conclusion, the main purpose of the Open Baton framework is to deploy Network Services on top of cloud-based infrastructures in order to achieve higher scalability and flexibility.

SONATA NFV PLATFORM

SONATA is a vendor-agnostic MANO platform [11] that fully embraces the flexible programmability of 5G networks and the virtualization of the communication services enabled by Software Defined Networks (SDN) and Network Function Virtualization (NFV) technologies.

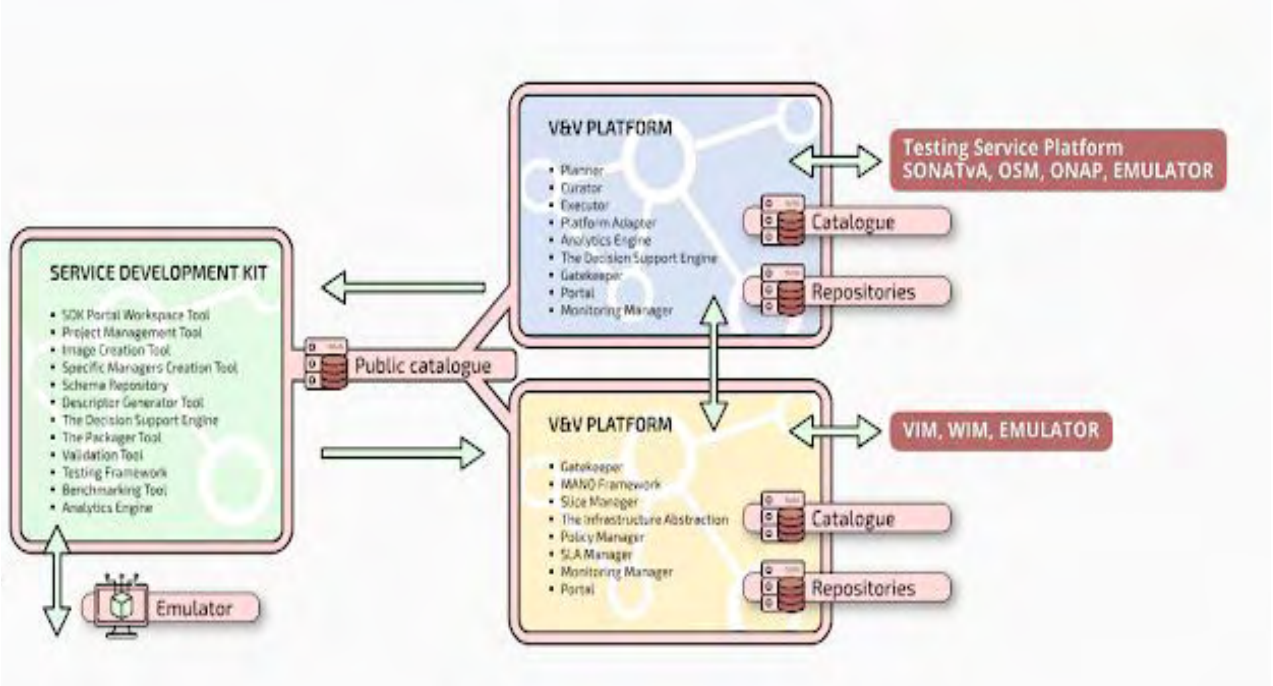

Figure 2.16: SONATA NFV Architecture

It follows a three-phased approach that covers the three main stages of any network service lifecycle: development, testing, and operations. Its flexible architecture consists of three main modules, each of them responsible for each of the above-mentioned phases:

- Development: An NFV-enabled Service Development Kit (SDK) to support developers in the creation of innovative Virtual Network Functions (VNF) and Network Services (NS).

- Testing: A Validation and Verification (V&V) Platform with advanced mechanisms for the qualification of VNFs and NSs.

- Operations: A modular Service Platform, with an innovative MANO framework for agile management of the full lifecycle of the deployed NSs and the use of resources.

These three modules can work either in an integrated or isolated way but, in combination, they release an extended NFV DevOps model between service developers, telecom operators and vertical industries that increases operational efficiency, fosters collaboration and facilitates the launch of innovative services.

In this thesis, Open Source Mano has been chosen as an orchestration platform due to its high level of adherence to the ETSI NFV-MANO standard.

As we presented above, the Virtualized Infrastructure Manager (VIM) is part of the ‘MANO Stack’ and addresses provides lifecycle management for virtualized resources. For this reason, we will present bellow the most popular platforms used for the role of VIM and is compatible with Open Source MANO

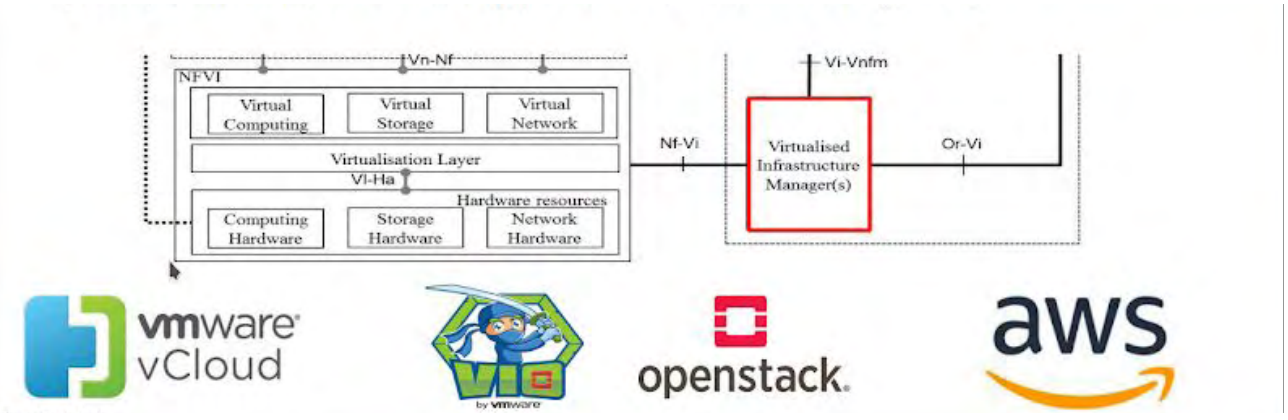

Figure 2.17: Virtualized Infrastructure Manager Vendors

AWS

Amazon Web Services (AWS)[13] is a secure cloud services platform, offering compute power, database storage, content delivery, and other functionality to help businesses scale and grow. Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. [12]

VMware

VMware [14] offers two ETSI OSM compatible VIM solutions:

- VMware vCloud Director: It is combined with ETSI OSM enables easy and intuitive cloud provisioning and consumption. It includes built-in services such as data protection, disaster recovery, multi-site management, multi-tenancy and more. vCloud Director enables Cloud Providers to deliver differentiated cloud services on their VMware Cloud infrastructure and provides enterprises a self-service cloud. [12]

- VMware Integrated Openstack: VMware vCloud NFV OpenStack Edition provides the simplest path for communications service providers (CSPs) to deploy network functions virtualization (NFV) services on OpenStack. VMware vCloud NFV OpenStack Edition combines carrier-grade NFV infrastructure with VMware Integrated OpenStack Carrier Edition as an NFV virtualized infrastructure manager (VIM) that transforms CSP networks, operations, and business models by delivering agile, secure, and highly available services. The open architecture and integrated operations management with 360-degree visibility across the NFVI layer allows CSPs to accelerate service innovation while delivering network flexibility, performance feedback, and fast remediation capabilities. With a fully compatible Openstack API, VMware Integrated OpenStack integrates with OpenSourceMANO (ETSI OSM) for a robust fully-functional NFV solution. [12]

OpenVIM

OpenVIM [16] is a part of OpenMANO. OpenVIM is a lightweight implementation of an NFV VIM supporting EPA features and control of an underlay switching infrastructure through an OFC. In addition, a reference implementation of an NFV VIM (Virtualized Infrastructure Manager), with support for high and predictable performance. It interfaces with the compute nodes in the NFV Infrastructure and an OpenFlow controller to provide computing and networking capabilities and deploy virtual machines [16]. It offers an OpenStack-like northbound interface (openvim API), where enhanced cloud services are offered including the creation, deletion, and management of images, flavors, instances, and networks. This implementation follows ETSI's NFV-PER001 recommendations.[12]

OpenStack

OpenStack [15] is the standard for private clouds and is also available as a service via dozens of public cloud providers around the world. At its core, OpenStack is an open-source integration engine that provides APIs to orchestrate bare metal, virtual machine and container resources on a single network. The same OpenStack code powers a global network of public and private clouds, backed by the largest ecosystem of technology providers, to enable cost savings, control and portability.[12]

OpenStack is a global community of more than 70,000 individuals across 180 countries supported by the OpenStack Foundation, which facilitates the development of many innovative projects in the open infrastructure space. The community delivers two software releases each year, which are Apache 2 licensed and productized by a large ecosystem of technology vendors in our Marketplace.[12]

2.4 NITOS Testbed

As we mentioned all the developments are integrated with the NITOS testbed [35], which provides access to physical resources in terms of host machines in order to support experimentation for wireless networks. In this thesis, the proposed facility used is the NITOS testbed.

NITOS (Network Implementation Testbed using Open Source code) is a wireless testbed managed by the Network Implementation Testbed Laboratory (NITlab) of the Electrical and Computer Engineering Department at the University of Thessaly in collaboration with the Center of Research and Technology Hellas (CERTH). The NITOS testbed gives researchers the opportunity to perform experiments and test their implementations in real-time environments. It consists of wireless nodes based on open-source software.

Figure 2.18: NITOS

The NITOS facilities are the NITOS outdoor testbed, the CERTH indoor testbed, and the Tholos indoor testbed. The control and Management Framework (OMF) is being used in order to control and manage the testbed.

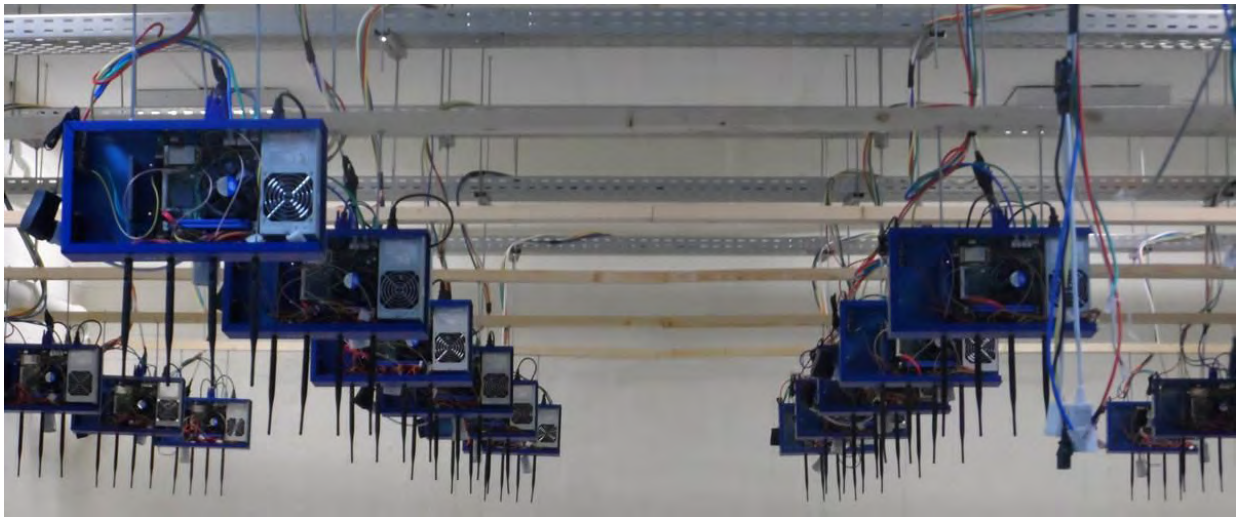

Figure 2.19: Tholos Indoor testbed

The Tholos indoor testbed is an isolated physical of nodes environment at the University of Thessaly's campus building. It consists of 40 Icarus nodes that contain multiple heterogeneous interfaces making the user capable of performing several realistic scenarios.

The WiFi ICARUS nodes have been developed by the NITlab team. The basic manufacture characteristics of Icarus nodes are that they are equipped with 802.11a/b/g and 802.11a/b/g/n wireless interfaces and feature new generation intel 4-core CPU’s and new generation solid-state drives.

Figure 2.20: Icarus Node

The specification of Icarus Nodes can be included in the below table.

Specifications:

Motherboard	Features two Gigabit network interfaces and supports two wireless interfaces
CPU	Intel® Core™ i7-2600 Processor, 8M Cache, at 3.40 GHz
RAM	8G DDR3
Wireless Interfaces	Atheros 802.11a/b/g & Atheros 802.11a/b/g/n (MIMO)
Chassis Manager card	NITlab CM card
Storage	Solid-state drive
Power Supply	350 Watt mini-ATX
Antennas	Multi-band 5dbi, operates both on 2.4Ghz & 5Ghz
Pigtails	High-quality pigtails (UFL to RP-SMA)

Table 1: Icarus Nodes Specifications

3

System Architecture

In this section, we present the design and implementation of the proposed architecture. This chapter is organized into two different parts. The first one presents the architecture for provisioning the testbed with the NFV-MANO paradigm and how the integration of wireless technologies occurred, while the second details the integration of the orchestrator with the existing NITOS tools for provisioning the testbed to experimenters.

3.2 Design

As we mentioned the main scope of this thesis is the orchestration and management of network services via the Open Source MANO orchestration platform. Especially, we combine the deployed virtualized functions created by the OSM orchestrator with LTE, WiFi and mmWave functionalities provided by NITOS physical resources. The VIM that supports our extensions in order to manage NITOS nodes as compute infrastructure is Openstack. A high-level approach of VNF's deployment on top of NITOS compute nodes can be shown below.

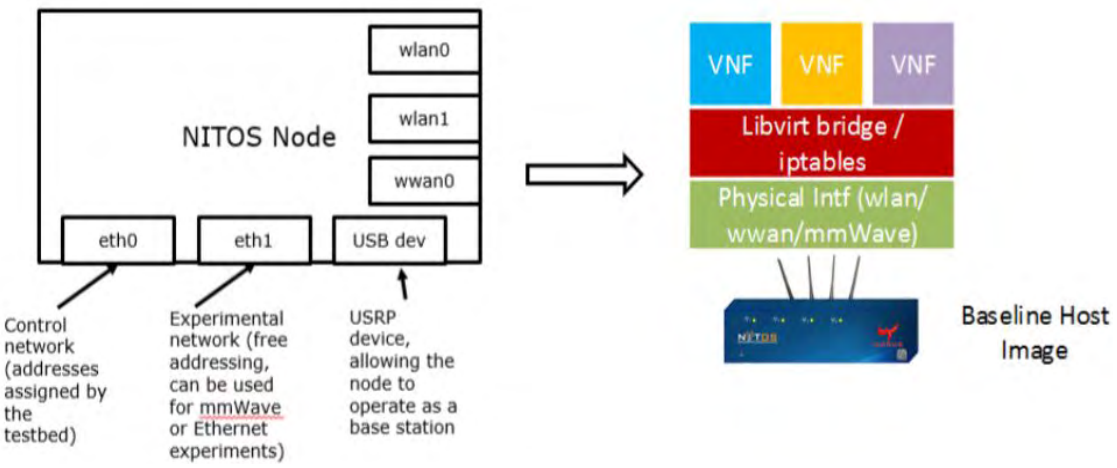

Figure 3.14: Placing VNFs on NITOS nodes

The proposed architecture aims to abstract the wireless interfaces (WiFi/LTE/mmWave) of NITOS compute nodes and bridge them with the specific interfaces of VNF. As a result, the VNF can use either the bridged Ethernet or wireless connection of the NITOS node. In this thesis, we use Open-vSwitch as the middleware for the bridged connection between the VNFs and the physical interfaces [36]. In addition, as we can observe in figure 3.1 Open Source MANO orchestrates the NFVs instantiation but not interact with the integration of physical interfaces with the VNFs. This is a task of Openstack which is located at the NITOS VIM side.

Finally, we should inform that the proposed architecture will not alter the information model for the intercommunication of the different entities in the NFV-MANO architecture as all the changes we have made have been established inside the OpenStack VIM. This results in possessing a generic solution that can be applied to any different Open Source MANO platform that can be hosted anywhere.

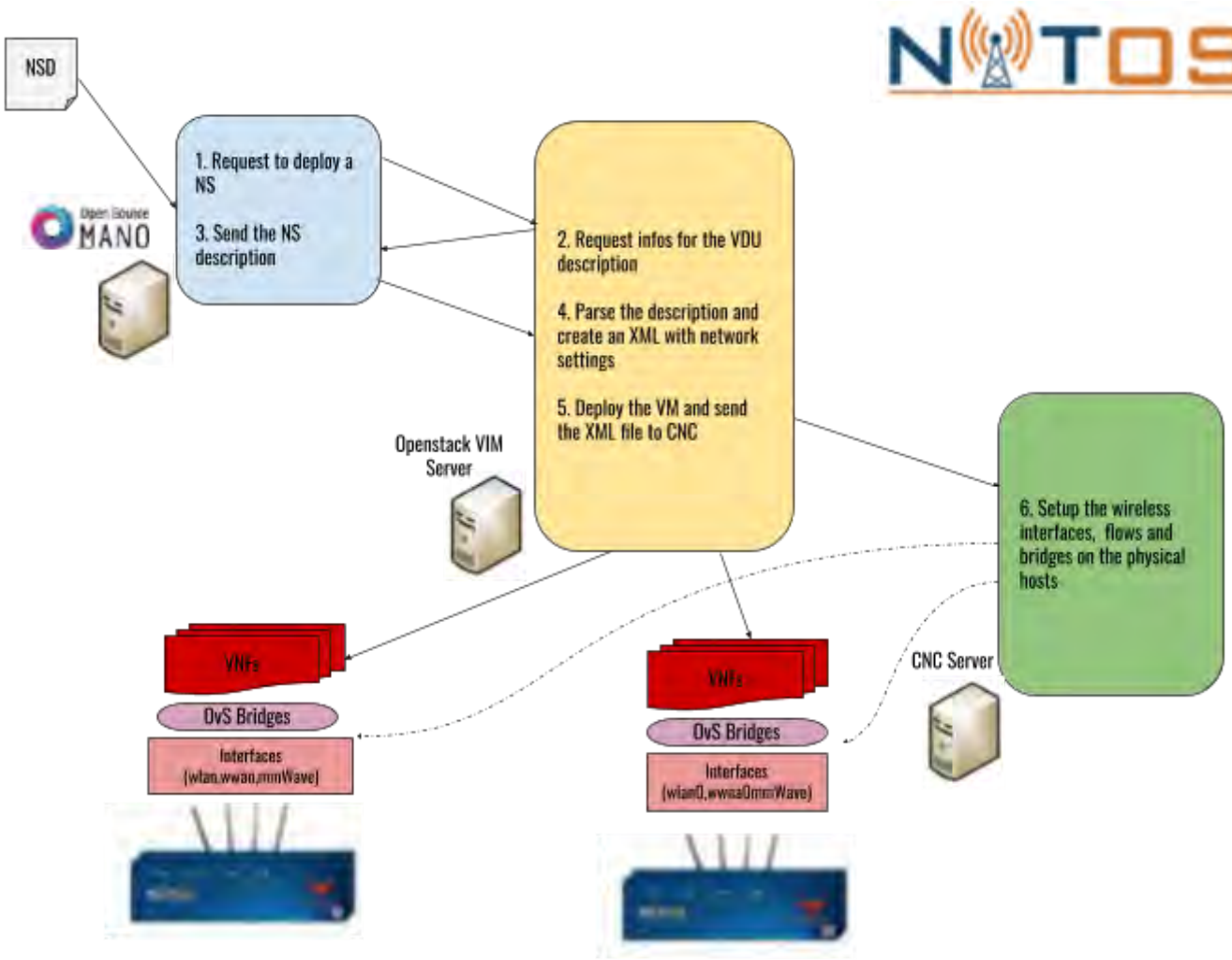

Figure 3.1: Instantiation Flow of deploying VNFs over the wireless interfaces of the testbed

As mentioned, after the OSM's NS deployment the actions taken for the establishment of the wireless connections are the responsibility of the NITOS VIM side which contains the Openstack and the Compute Network Control (CNC) server. Compute Network Control (CNC) is a Sinatra based framework that has developed through the Future Internet Research and Experimentation(FIRE) initiative [38] for testbeds. It is hosted at NITOS VIM side capable of managing HTTP XML requests and establishing the requested wireless interfaces on the psychical hosts. Especially, as we observe from the figure below the deployment flow has the following steps:

1. OSM receives a Network Service Description (NSD) parses the different components, and for each VDU, sends a request to the VIM instance sends in our case Openstack server in order to deploy the new Network Service.

2. Openstack communicates back to the REST API of the OSM RO functional block in order to get more information about the NS such as VNF and VDU description.

3. The RO block of OSM sends the requested information to Openstack.

4. Then Openstack checks the VDU's description field in order to specify the wireless network type requested. If there is one, then parses the description data and creates an XML with the given network settings. An example of the XML file for the different types of wireless networks is presented below :

WiFi Access Point XML

```
<xml version=”1.0”?>
<wifiAP>
   <interface>wlan0</interface>
   <channel>1</channel>
   <mode>g</mode>
   <essid>mainAP</essid>
   <vnfd>test_vnf</vnfd>
   <intf>eth1</intf>
   <vAP>
      <bss>wifi_vap0</bss>
      <essid>testVAP1</essid>
   </vAP>
</wifiAP>
```

WiFi Client XML

```
<xml version=”1.0”?>
<wifiClient>
   <interface>wlan0</interface>
   <client-ip>192.168.10.2</client-ip>
   <essid>mainAP</essid>
```

```
    <vnfd>test_vnf</vnfd>
    <intf>eth1</intf>
</wifiClient>
```

mmWave XML

```
<xml version="1.0"?>
<mmWave>
    <ip>192.168.5.1</ip>
    <physical-node-id>1</physical-node-id>
    <intf>eth1</intf>
</mmWave>
```

LTE Profile XML

```
<xml version="1.0"?>
<vAPN>
    <qci>5</qci>
    <name>TEST_SS</name>
    <ulambr></ulambr>
    <dlambr></dlambr>
</vAPN>
```

LTE Client XML

```
<xml version="1.0"?>
<vLteCluster>
    <name>TEST_SS</name>
    <vnfd>test_vnf</vnfd>
    <client>1</client>
    <intf>eth1</intf>
</vLteCluster>
```

5. Then Openstack deploys the VMs on the selected physical NITOS nodes hosted on its pool and sends via an HTTP POST the XML config file to the CNC.

6. The CNC receives an XML config file for each deployed VDU. Then for each one, access the selected physical node and set up the wireless interfaces, bridges and OvS flows.

3.2.1 VNFs with LTE connectivity

For the LTE, the VNF is provided has two network connections, the first one as a management connection running over the Ethernet network of NITOS, while a second one associated to the NITOS LTE infrastructure.

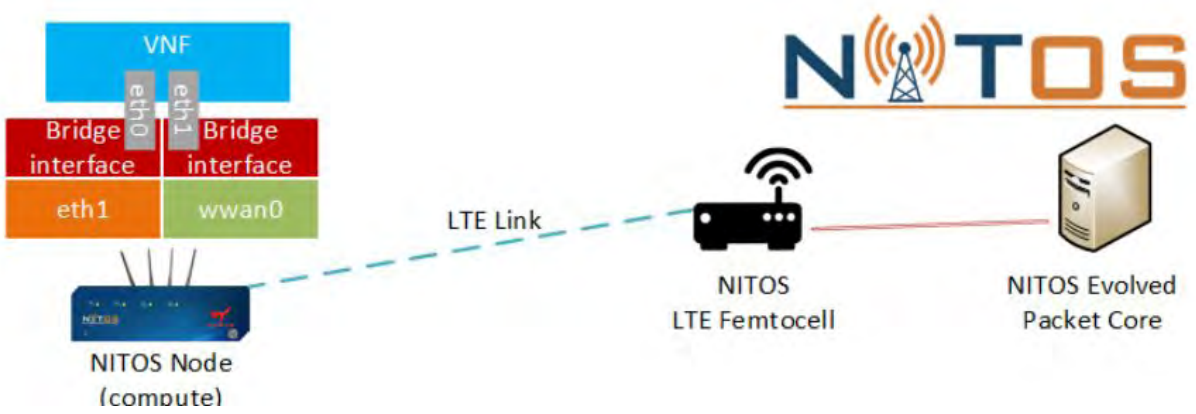

Figure 3.2: VNF deployment with LTE connectivity over the CITS NITOS LTE network

In addition, some parameters are needed to be configured for the connection and the creation of the LTE network. These parameters are hosted at an extra field inside the description of the VNF(VNFD). Specifically, these parameters are:

- LTE subscriber group which the VNF will belong to
- Access Point Name (APN)
- the maximum Uplink the VNF will use
- Downlink traffic the VNF will use

Figure 3.3 shows the VNFD with some configuration for the LTE network that can be changed by the experimenter. Figure 3.4 presents an example of NSD that is making use of this VNF.

```
vnfd:vnfd-catalog:
 vnfd:
 - connection-point:
   - id: eth0
     name: eth0
     short-name: eth0
     type: VPORT
   - id: eth1
     name: eth1
     short-name: eth1
     type: VPORT
   description: A VNF consisting of 1 VDUs connected to an LTE VL
   id: lteUE-vnf
   logo: osm.png
   mgmt-interface:
     cp: eth0
   name: lteUE-vnf
   short-name: lteUE-vnf
   vdu:
   - cloud-init-file: cloud-config.txt
     count: '1'
     description: type:Lte,virtual-apn:TEST_SS,dlambr:50000000,ulambr:50000000,qci:5,cluster-client:lte_client1,virtual-cluster:TEST_SS
     id: lteUEVM
     image: ubuntu1404_cloud
     interface:
     - external-connection-point-ref: eth0
       name: eth0
       position: '1'
       type: EXTERNAL
       virtual-interface:
         type: VIRTIO
     - external-connection-point-ref: eth1
       name: eth1
       position: '2'
       type: EXTERNAL
       virtual-interface:
         type: VIRTIO
     name: lteUEVM
     vm-flavor:
       memory-mb: '4096'
       storage-gb: '5'
       vcpu-count: '4'
   version: '1.0'
```

Figure 3.3: VNFD for establishing an LTE connection in conjunction with the VNF deployment at NITOS testbed

```
nsd:nsd-catalog:
 nsd:
 - constituent-vnfd:
   - member-vnf-index: '1'
     vnfd-id-ref: lteUE-vnf
   description: NS with 1 VNFs connected by datanet and mgmtnet VLs
   id: lte-ns
   logo: osm.png
   name: lte-ns
   short-name: lte-ns
   version: '1.0'
   vld:
   - id: mgmtnet
     mgmt-network: 'true'
     name: mgmtnet
     short-name: mgmtnet
     type: ELAN
     vim-network-name: provider1
     vnfd-connection-point-ref:
     - member-vnf-index-ref: '1'
       vnfd-connection-point-ref: eth0
       vnfd-id-ref: lteUE-vnf
   - id: datanet
     name: datanet
     short-name: datanet
     type: ELAN
     vnfd-connection-point-ref:
     - member-vnf-index-ref: '1'
       vnfd-connection-point-ref: eth1
       vnfd-id-ref: lteUE-vnf
```

Figure 3.4: NSD for provisioning the LTE VNF

3.2.2 VNFs with WiFi connectivity

For the WiFi case, all the nodes of the testbed are equipped with WiFi card that supports a plethora of modes of the IEEE 802.11 suite of protocols such as Access Point, Station, Mesh, Ad-hoc, etc. In our case, the deployed VNFs can be either of type Access Point or Station.

Similarly to the LTE case, the deployed VNF has again two different interfaces, one used for managing the VNF and is bridged with the ethernet interface of the node while a second one that is bridged over the wireless network through OvS flows. In addition, we can create multiple virtual Access Points through the hostapd tool [37] on the compute node and assign a specific ESSID (network name). As a result, VNFs that are configured as WiFi stations (STA) can be associated to the VAP with the same ESSID.

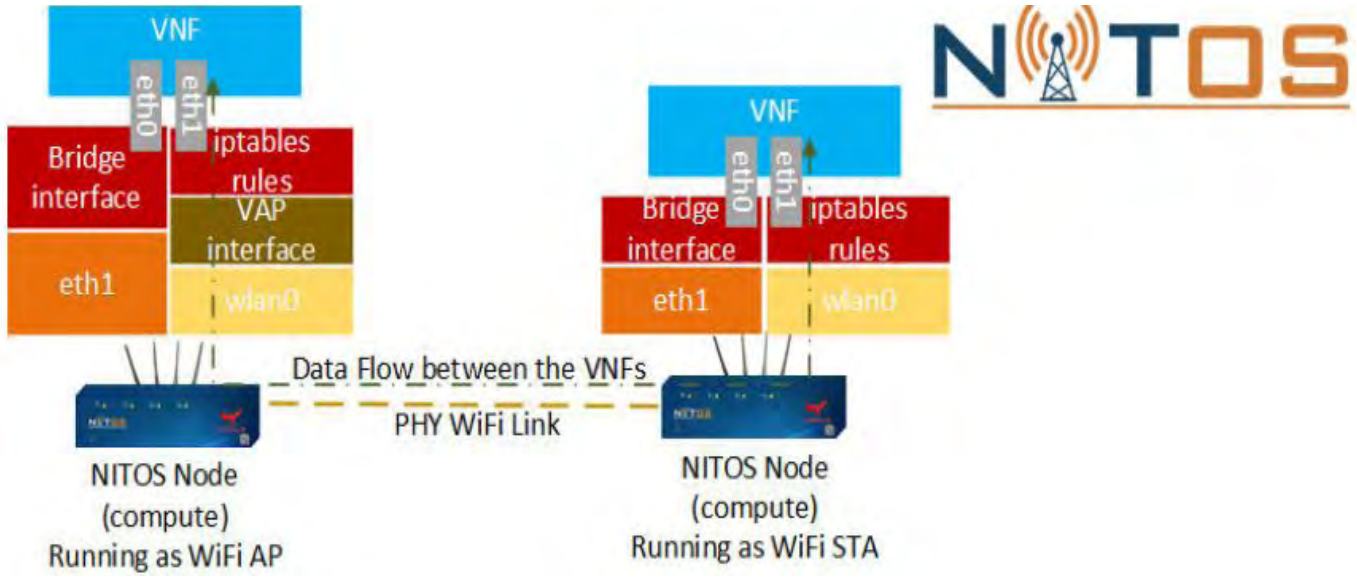

Figure 3.5: Pair of VNFs deployment with WiFi connectivity

In addition, some parameters are needed to be configured for the connection and the creation of the WiFi network. These parameters are hosted at an extra field inside the description of the VNF(VNFD). Specifically, these parameters are:

- For the WiFi AP side:
    1. the physical interface on the compute node
    2. the ESSID name
    3. BSS
    4. WiFi mode (a/b/g/n/ac)
    5. the operating channel

- For the WiFi Station side:
    1. the physical interface on the compute node
    2. the ESSID name

Figure 3.6 and Figure 3.7 present the example VNFs for running over a WiFi AP and Station respectively. Figure 3.8 illustrates an NSD used to deploy these two VNFs over the testbed.

```
vnfd:vnfd-catalog:
 vnfd:
 - connection-point:
   - id: eth0
     name: eth0
     short-name: eth0
     type: VPORT
   - id: eth1
     name: eth1
     short-name: eth1
     type: VPORT
   description: A VNF consisting of 1 VDU connected to a controll ethernet VL and to a wifi VL
   id: wifiAP-vnf
   logo: osm.png
   mgmt-interface:
     cp: eth0
   name: wifiAP-vnf
   short-name: wifiAP-vnf
   vdu:
   - cloud-init-file: cloud-config.txt
     count: '1'
     description: type:WifiAP,interface:wlan0,bss:vap1,essid:test,mode:g,channel:1
     id: wifApVM
     image: ubuntu1404_cloud
     interface:
     - external-connection-point-ref: eth0
       name: eth0
       position: '1'
       type: EXTERNAL
       virtual-interface:
         type: VIRTIO
     - external-connection-point-ref: eth1
       name: eth1
       position: '2'
       type: EXTERNAL
       virtual-interface:
         type: VIRTIO
     name: wifiApVM
     vm-flavor:
       memory-mb: '4096'
       storage-gb: '5'
       vcpu-count: '4'
   version: '1.0'
```

Figure 3.6: VNFD for establishing a WiFi AP connection in conjunction with the VNF deployment at NITOS testbed

```
vnfd:vnfd-catalog:
 vnfd:
 - connection-point:
   - id: eth0
     name: eth0
     short-name: eth0
     type: VPORT
   - id: eth1
     name: eth1
     short-name: eth1
     type: VPORT
   description: A VNF consisting of 1 VDUs connected to an wifi VL
   id: wifiClient-vnf
   logo: osm.png
   mgmt-interface:
     cp: eth0
   name: wifiClient-vnf
   short-name: wifiClient-vnf
   vdu:
   - cloud-init-file: cloud-config.txt
     count: '1'
     description: type:WifiClient,interface:wlan0,essid:test
     id: wifiClientVM
     image: ubuntu1404_cloud
     interface:
     - external-connection-point-ref: eth0
       name: eth0
       position: '1'
       type: EXTERNAL
       virtual-interface:
         type: VIRTIO
     - external-connection-point-ref: eth1
       name: eth1
       position: '2'
       type: EXTERNAL
       virtual-interface:
         type: VIRTIO
     name: wifiClientVM
     vm-flavor:
       memory-mb: '4096'
       storage-gb: '5'
       vcpu-count: '4'
   version: '1.0'
```

Figure 3.7: VNFD for establishing a WiFi Station connected to the AP at NITOS testbed

```
nsd:nsd-catalog:
 nsd:
 - constituent-vnfd:
   - member-vnf-index: '1'
     vnfd-id-ref: wifiAP-vnf
   - member-vnf-index: '2'
     vnfd-id-ref: wifiClient-vnf
   description: NS with 2 VNFs connected by datanet and mgmtnet VLs
   id: wifi-ns
   logo: osm.png
   name: wifi-ns
   short-name: wifi-ns
   version: '1.0'
   vld:
   - id: mgmtnet
     mgmt-network: 'true'
     name: mgmtnet
     short-name: mgmtnet
     type: ELAN
     vim-network-name: provider1
     vnfd-connection-point-ref:
     - member-vnf-index-ref: '1'
       vnfd-connection-point-ref: eth0
       vnfd-id-ref: wifiAP-vnf
     - member-vnf-index-ref: '2'
       vnfd-connection-point-ref: eth0
       vnfd-id-ref: wifiClient-vnf
   - id: datanet
     name: datanet
     short-name: datanet
     type: ELAN
     vnfd-connection-point-ref:
     - member-vnf-index-ref: '1'
       vnfd-connection-point-ref: eth1
       vnfd-id-ref: wifiAP-vnf
     - member-vnf-index-ref: '2'
       vnfd-connection-point-ref: eth1
       vnfd-id-ref: wifiClient-vnf
```

Figure 3.8: NSD for provisioning the WiFi VNF

3.2.3 VNFs with mmWave connectivity

For the mmWave case, NITOS testbed provides six nodes with mmWave technology. Each of the six nodes has a unique VLAN interface. Especially, traffic that sent over VLAN 692 is reaching the first node, VLAN 693 the second, etc. Similar to the previous technologies, A VLAN interface is created on the compute node that is attached to an OvS bridge communicating with a second network interface of the VNF. Traffic sent over this interface gets transmitted over the VLAN, reaching the mmWave node and then over the air. At the receiving mmWave node, the traffic is encapsulated in the respective VLAN addressing the mmWave node. Traffic is transmitted over the VLAN and can be delivered at the VNF that has this VLAN configured at the compute node. This procedure is illustrated in Figure 3.9.

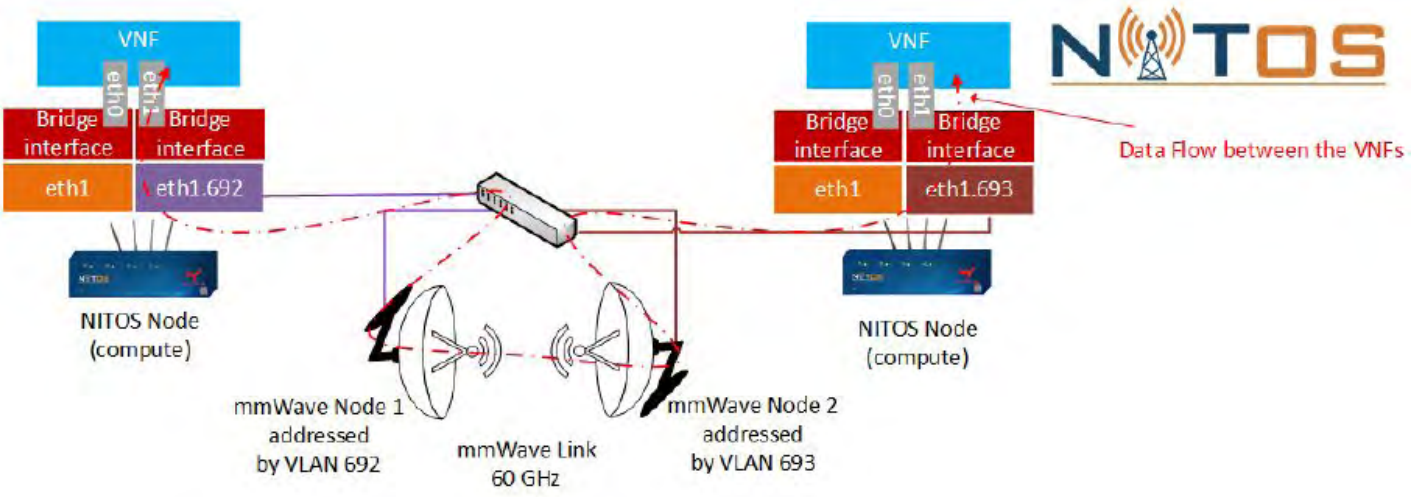

Figure 3.9: mmWave network provisioning for two VNFs deployed in NITOS

Similar to the previous cases, some parameters are being passed through the description field; Figure 3.10 and Figure 3.11 provide the descriptions for provisioning two VNFs connecting over two mmWave nodes in the testbed. Figure 3.12 shows the respective NSD

```
vnfd:vnfd-catalog:
 vnfd:
 - connection-point:
   - id: eth0
     name: eth0
     short-name: eth0
     type: VPORT
   - id: eth1
     name: eth1
     short-name: eth1
     type: VPORT
   description: A VNF consisting of 1 VDUs connected with two VLs. One is the controller network and the othes is for mmwave network
   id: mmwaveOne-vnf
   logo: osm.png
   mgmt-interface:
     cp: eth0
   name: mmwaveOne-vnf
   short-name: mmwaveOne-vnf
   vdu:
   - cloud-init-file: cloud-config.txt
     count: '1'
     description: type:mmWave,physical-node-id:1
     id: mmwaveOneVM
     image: ubuntu1404_cloud
     interface:
     - external-connection-point-ref: eth0
       name: eth0
       position: '1'
       type: EXTERNAL
       virtual-interface:
         type: VIRTIO
     - external-connection-point-ref: eth1
       name: eth1
       position: '2'
       type: EXTERNAL
       virtual-interface:
         type: VIRTIO
     name: mmwaveOneVM
     vm-flavor:
       memory-mb: '4096'
       storage-gb: '5'
       vcpu-count: '4'
   version: '1.0'
```

Figure 3.10: VNF making use of the mmWave physical node 1 in NITOS

```
vnfd:vnfd-catalog:
 vnfd:
 - connection-point:
   - id: eth0
     name: eth0
     short-name: eth0
     type: VPORT
   - id: eth1
     name: eth1
     short-name: eth1
     type: VPORT
   description: A VNF consisting of 1 VDUs connected with two VLs. One is the controller network and the other is for mmwave network
   id: mmwaveTwo-vnf
   logo: osm.png
   mgmt-interface:
     cp: eth0
   name: mmwaveTwo-vnf
   short-name: mmwaveTwo-vnf
   vdu:
   - cloud-init-file: cloud-config.txt
     count: '1'
     description: type:mmWave,physical-node-id:2
     id: mmwaveTwoVM
     image: ubuntu1404_cloud
     interface:
     - external-connection-point-ref: eth0
       name: eth0
       position: '1'
       type: EXTERNAL
       virtual-interface:
         type: VIRTIO
     - external-connection-point-ref: eth1
       name: eth1
       position: '2'
       type: EXTERNAL
       virtual-interface:
         type: VIRTIO
     name: mmwaveTwoVM
     vm-flavor:
       memory-mb: '4096'
       storage-gb: '5'
       vcpu-count: '4'
   version: '1.0'
```

Figure 3.11: VNF making use of the mmWave physical node 2 in NITOS

```
nsd:nsd-catalog:
 nsd:
 - constituent-vnfd:
   - member-vnf-index: '1'
     vnfd-id-ref: mmwaveOne-vnf
   - member-vnf-index: '2'
     vnfd-id-ref: mmwaveTwo-vnf
   description: NS with 2 VNFs connected by datanet and mgmtnet VLs
   id: mmwave-ns
   logo: osm.png
   name: mmwave-ns
   short-name: mmwave-ns
   version: '1.0'
   vld:
   - id: mgmtnet
     mgmt-network: 'true'
     name: mgmtnet
     short-name: mgmtnet
     type: ELAN
     vim-network-name: provider1
     vnfd-connection-point-ref:
     - member-vnf-index-ref: '1'
       vnfd-connection-point-ref: eth0
       vnfd-id-ref: mmwaveOne-vnf
     - member-vnf-index-ref: '2'
       vnfd-connection-point-ref: eth0
       vnfd-id-ref: mmwaveTwo-vnf
   - id: datanet
     name: datanet
     short-name: datanet
     type: ELAN
     vnfd-connection-point-ref:
     - member-vnf-index-ref: '1'
       vnfd-connection-point-ref: eth1
       vnfd-id-ref: mmwaveOne-vnf
     - member-vnf-index-ref: '2'
       vnfd-connection-point-ref: eth1
       vnfd-id-ref: mmwaveTwo-vnf
```

Figure 3.12: NSD for provisioning VNFs with mmWave connectivity in NITOS

3.3 Testbed Integration

As mentioned above NITOS is a wireless oriented facility established under the Future Internet Research and Experimentation(FIRE) initiative [38]. The FIRE initiative created a plethora of tools for experimentation with testbed resources. NITOS testbed uses a scheduler interface in order to give the opportunity to researchers to select the number of nodes they want to use within a timeslot. Timeslots are the amount of time that nodes are reserved. In NITOS the maximum amount of time is 4 hours. This calendar-based reservation is very common within the FIRE testbeds federation, especially for wireless resources.

Our goal is to combine NFV-MANO orchestrator capabilities with the testbeds existing tools. For this reason, we created a daemonized service at the OpenStack VIM side for reserving the testbed resources before the orchestration process takes place. Especially, before the orchestration occurred the daemon who is running on the VIM server periodically (every 10 minutes) communicates with OSM Portal for any new onboarded and validated VNFs. Through further parsing of the portal response, the constituent VNFs that will be used for the orchestration are retrieved. Then specified the number of compute resources will be used for the respective experiment in order to make the reservation.

For reserving the testbed resources, the FIRE tools that manage the testbed are invoked. NITOS testbed is managed and provisioned through the NITOS Broker [39]. The Broker acting as the federation API with other testbeds within Fed4FIRE, supporting two different APIs at its northbound interface: a custom REST or an SFA [40]. For the purposes of integrating the testbed we used the SFA API. The SFA API supports two different versions of the protocol: version 2 and version 3. There are small differences in the API calls for listing, retrieving and preparing resources in the testbed. Currently, the developments for the integration are based on Version 2

After extracting the number of compute nodes that should be used for the experiment and configure the type of technology requested, the daemon communicates with NITOS broker in order to check the availability of resources for the requested time slot. If the nodes are available, the tool reserves them and places them under a specific name. In addition, when the reservation starts the daemon prepares the physical resources by loading a specific image. This image refers to a binary copy of the hard disk of a node, including the operating system, and all the kernel modules, services and applications that will run on it. For our case, the image is an Ubuntu Server 14.04 installation, with virsh and Qemu 3.0 services installed, and some applications for connecting the LTE dongles to the network and setting up the node as a WiFi AP. The overall procedure is shown in the following Figure.

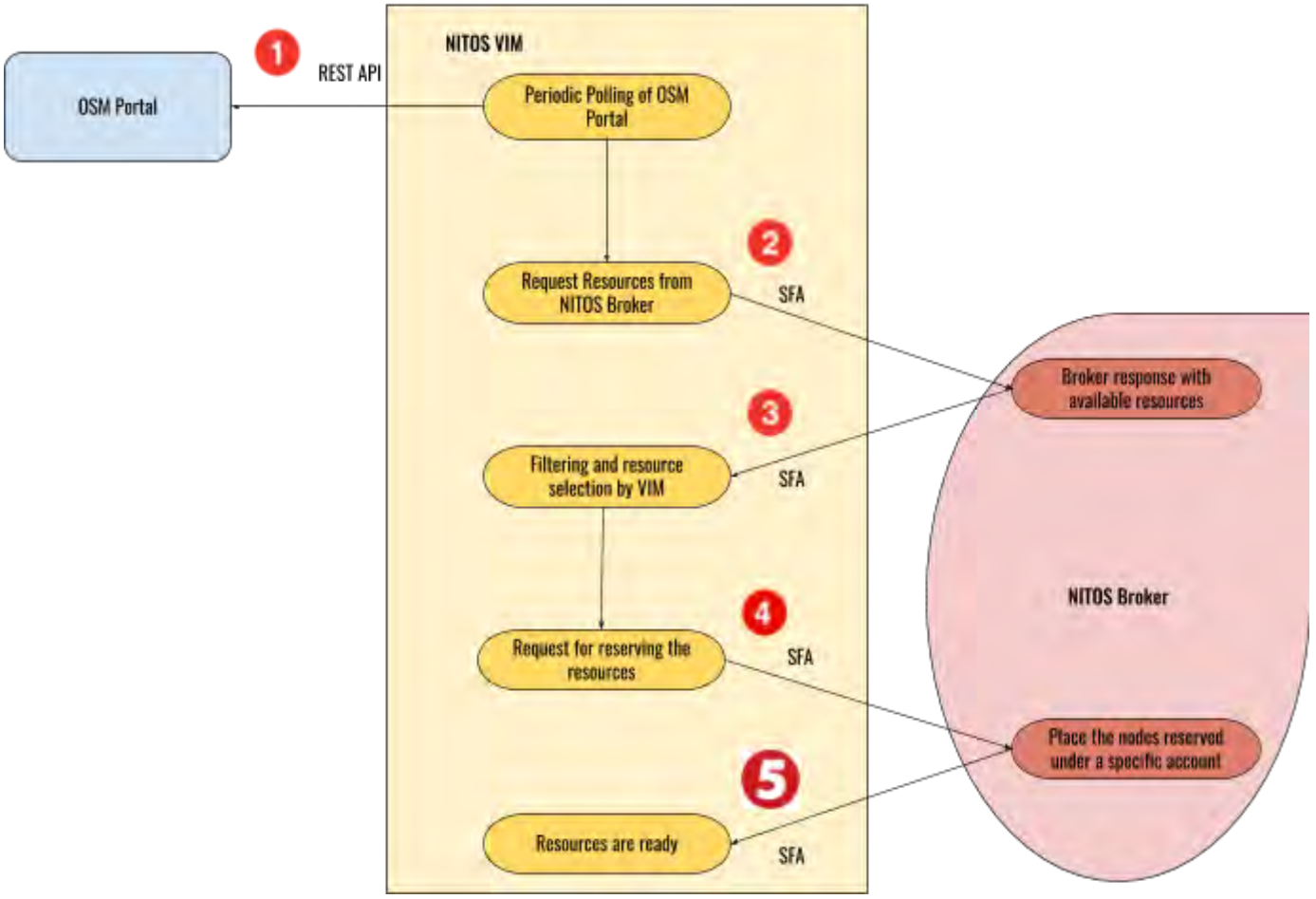

Figure 3.13: Testbed Integration

4

Conclusion

In this thesis, we presented the extensions we have made in order to integrate wireless networks with the orchestration software (OSM) installed at a wireless testbed NITOS. With the above implementation, we can deploy and instantiate VNFs over a wireless link. As we mentioned, this could be an interface attached to a VM with an LTE, WiFi, and mmWave technology. This proposed functionality gives the opportunity for researchers to deploy and manage VNFs with wireless technologies running over a distributed datacenter with the minimum configuration overhead. In addition, as we mentioned above we not alter the information model for the intercommunication of the different entities in the NFV-MANO architecture because all the extensions we made hosted inside the VIM. So, the functionality we proposed is a generic solution that can be applied to any other NITOS-like FIRE testbed

4.1 Future Work

In the future, we aim to extend the process of service orchestration by using JuJu or Cloud-Init, for the configuration of services on the VNF. Also, we aim to introduce a comparison between our two different VIMs (Openstack and OpenVim) considering the overhead that the same VNF on each VIM poses in the time for deployment. In addition, we foresee to use the latest release of OSM, Release SEVEN as it supports the WIM plugin for the creation of a multi-VIM environment. The respective distributed architecture will give us the opportunity to deploy VNFs simultaneously on different VIMs across the world. Last but not least, we aim to combine Kubernetes [41], alongside with OSM in order to deploy docker containers integrated with wireless networks instead of VMs.

Preferences

[1] https://en.wikipedia.org/wiki/Network_function_virtualization

[2] “Design and implementation of an ETSI Network Function Virtualization compliant container orchestrator” by CILLONI Marco

[3]https://www.sdxcentral.com/networking/nfv/definitions/virtualized-infrastructure-manager-vim-

[4] https://www.sdxcentral.com/networking/nfv/mano-lso/definitions/nfv-mano/

[5]https://www.mirantis.com/blog/which-nfv-orchestration-platform-best-review-osm-open-o-cord-cloudify

[6] https://osm.etsi.org

[7] https://openbaton.github.io

[8] https://www.opennetworking.org/xos/

[9] https://www.gigaspaces.com/

[10] https://www.onap.org

[11] http://www.sonata-nfv.eu

[12] https://osm.etsi.org/wikipub/index.php/VIMs

[13] https://aws.amazon.com/

[14] https://www.vmware.com/products/vcloud-director.html

[15] https://www.openstack.org

[16] https://www.sdxcentral.com/projects/openvim/

[17] Mayoral, A.; Vilalta, R.; Casellas, R.; Martinez, R.; Munoz, R. Multi-tenant 5G Network Slicing Architecture with Dynamic Deployment of Virtualized

Tenant Management and Orchestration (MANO) Instances.ECOC 2016; 42nd European Conference on Optical Communication, 2016, pp. 1–3.5639

[18] T-NOVA: An Open-Source MANO Stack for NFV Infrastructures by Kourtis, M.; McGrath, M.J.; Gardikis, G.; Xilouris, G.; Riccobene, V.; Papadimitriou, P.; Trouva, E.; Liberati published in IEEE Transactions on Network and Service… 2017

[19] Francescon, A.; Baggio, G.; Fedrizzi, R.; Ferrusy, R.; Ben Yahiaz, I.G.; Riggio, R. X-MANO: Cross-domain management and orchestration of network services. 2017 IEEE Conference on Network Softwarization (NetSoft), 2017, pp. 1–5. doi:10.1109/NETSOFT.2017.8004223.

[20] Francescon, A.; Baggio, G.; Fedrizzi, R.; Orsini, E.; Riggio, R. X-MANO: An open-source platform for cross-domain management and orchestration. 2017 IEEE Conference on Network Softwarization (NetSoft),2017, pp. 1–6. doi:10.1109/NETSOFT.2017.8004236.

[21] Bernardos, C.J.; Gerö, B.P.; Di Girolamo, M.; Kern, A.; Martini, B.; Vaishnavi, I. 5GEx: realising a Europe-wide multi-domain framework for software-defined infrastructures.Transactions on Emerging Telecommunications Technologies 2016,27, 1271–1280.

[22] Vilalta, R.; Mayoral, A.; Baranda, J.; Nuñez, J.; Casellas, R.; Martínez, R.; Mangues-Bafalluy, J.; Muñoz, R. Hierarchical SDN orchestration of wireless and optical networks with E2E provisioning and recovery for future 5G networks. 2016 Optical Fiber Communications Conference and Exhibition (OFC), 2016, pp. 1–3.

[23] Santos, R.; Kassler, A. A SDN controller architecture for Small Cell Wireless Backhaul using a LTE Control Channel. 2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2016, pp. 1–3. doi:10.1109/WoWMoM.2016.7523544.

[24] Dalla-Costa, A.G.; Bondan, L.; Wickboldt, J.A.; Both, C.B.; Granville, L.Z. Maestro: An NFV Orchestrator for Wireless Environments Aware of VNF Internal Compositions.2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA), 2017, pp.484–491.649doi:10.1109/AINA.2017.126.\

[25] Katsalis, K.; Nikaein, N.; Huang, A. JOX: An event-driven Orchestrator for 5G Network Slicing. NOMS 2018 - 2018 IEEE/IFIP Network Operations

and Management Symposium, 2018, pp. 1–9. doi:10.1109/NOMS.2018.8406236.

[26] Makris, N.; Zarafetas, C.; Valantasis, A.; Korakis, T.; Tassiulas, L. Integrating NFV-MANO with Wireless Services: Experiences and Testbed Development. O4SDI - 4th Workshop on Orchestration for Software-Defined Infrastructures (at) 2018 IEEE NFV-SDN (NFV-SDN'18 - O4SDI); , 2018

[27] Richart, M.; Baliosian, J.; Serrat, J.; Gorricho, J. Resource Slicing in Virtual Wireless Networks: A Survey.IEEE Transactions on Network and Service Management2016,13, 462–476. doi:10.1109/TNSM.2016.2597295.

[28] Kamel, M.I.; Le, L.B.; Girard, A. LTE Wireless Network Virtualization: Dynamic Slicing via Flexible Scheduling.2014 IEEE 80th Vehicular Technology Conference (VTC2014-Fall), 2014, pp.1–5. doi:10.1109/VTCFall.2014.6966044.

[29] Katsalis, K.; Nikaein, N.; Schiller, E.; Ksentini, A.; Braun, T.Network Slices toward 5G Communications: Slicing the LTE Network.IEEE Communications Magazine2017,55, 146–154. doi:10.1109/MCOM.2017.1600936.

[30] Nikaein, N.; Schiller, E.; Favraud, R.; Katsalis, K.; Stavropoulos, D.; Alyafawi, I.; Zhao, Z.; Braun, T.; Korakis, T. Network Store: Exploring Slicing in Future 5G Networks. Proceedings of the 10th International Workshop on Mobility in the Evolving Internet Architecture; ACM: New York, NY, USA, 2015; MobiArch '15, pp. 8–13. doi:10.1145/2795381.2795390.

[31] Bhanage, G.; Seskar, I.; Mahindra, R.; Raychaudhuri, D. Virtual Basestation: Architecture for an OpenvShared WiMAX framework.Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures. ACM, 2010, pp. 1–8.

[32] Bhanage, G.; Seskar, I.; Raychaudhuri, D. A service oriented experimentation framework for virtualized WiMAX systems. International Conference on Testbeds and Research Infrastructures. Springer, 2011, pp. 152–161.

[33] Richart, M.; Baliosian, J.; Serrati, J.; Gorricho, J.; Agüero, R.; Agoulmine, N. Resource allocation for network slicing in WiFi access points. 2017 13th International Conference on Network and Service Management (CNSM), 2017, pp. 1–4. doi:10.23919/CNSM.2017.8256046.

[34] Yiakoumis, Y.; Yap, K.K.; Katti, S.; Parulkar, G.; McKeown, N. Slicing Home Networks. Proceedings of the 2Nd ACM SIGCOMM Workshop on Home Networks; ACM: New York, NY, USA, 2011; HomeNets '11, pp. 1–6. doi:10.1145/2018567.2018569.

[35] https://nitlab.inf.uth.gr

[36] https://www.openvswitch.org

[37] https://w1.fi/hostapd/

[38]https://ec.europa.eu/digital-single-market/en/future-internet-research-and-experimentation

[39] http://nitos.inf.uth.gr/

[40] https://github.com/onelab-eu/sfa

[41] https://kubernetes.io