5G Communication with a Heterogeneous, Agile Mobile network in the Pyeongchang Winter Olympic competitioN

Grant agreement n. 723247

Deliverable D4.2
Design and implementation report on Mobility Management, Integrated Orchestration and VNFs in a Distributed Mobile Core

Date of Delivery:	30 November 2017 (Contractual)	30 November 2017 (Actual)
Editor:	Wouter Tavernier	
Associate Editors:	Taeyeon Kim, Taesang Choi, Muhammad Arif	
Authors:	Wouter Tavernier, Taeyeon Kim, Taesang Choi, Muhammad Arif, Olli Liinamaa	
Dissemination Level:	PU	
Security:	Public	
Status:	Final	
Version:	1.0	
File Name:	5GCHAMPION_D4.2_Final.docx	
Work Package:	WP4	

	**Title:**	D4.2 Design and implementation report on MM, Integrated Orchestration and VNFs in a Distributed Mobile Core		
	Date:	November 2017	**Status:**	Final
	Security:	Public	**Version:**	1.0

Abstract

This deliverable reports on the implementation work that has been carried out on Korean and on the European mobile core network and its supporting SDN/NFV platforms. The ultimate goal of this work is to enable interoperability of both platforms and mobile core networks in the Proof of Concept in GangNeung due in February 2018. This deliverable is structured in two parts. The first part focuses on the development and extensions that have been made on the SDN/NFV platforms of KR and EU testbeds. This involves all MANagement and Orchestration (MANO) components, interfaces and associated processes, as well as preliminary evaluations that have been made on the performance of these platforms. The second part of the deliverable details the development on the software components of the KR and EU virtualized Evolved Packet Cores (vEPC). This includes the innovative distributed mobile core implementation and associated distributed mobility management functionality as developed by the Korean consortium partners. The actual interoperability of the documented implementations will be carried out at a later stage in the project, and will be documented in deliverable D4.3.

Index terms

Software-Defined Networking, Network Function Virtualization, Management, Orchestration, MANO, mobile core network, Evolved Packet Core, mobility management, distributed mobile core

Contents

List of Acronyms

3GPP	3rd generation partnership project
NFV	Network Functions Virtualization
MANO	Management and Network Orchestration
ETSI	European Telecommunication Standardization Organization
NFVI	Network Function virtualization infrastructure
VIM	Virtualized Infrastructure Manager
NFVO	Network Functions Virtualization Orchestrator
VNFM	Virtual Network Function Manager
NSD	Network Service Descriptor
VNFD	Virtual Network Function Descriptor
VNF	Virtual Network Functions
VNFR	Virtual Network Function Records
EPC	Evolved Packet Core
RAN	Radio Access Network
EMS	Element Management System
5GTN	5th Generation Test Network
PoP	Point of Presence
spgw	serving packet data network gateway
mme	Mobility Management Entity
JSON	Java Script Object Oriented Notation
CLI	Command Line Interface
VNFC	Virtual Network Function Component
REST	Representational State Transfer
PCRF	Policy control and charging rules function
AF	Application Functions
HSS	Home subscriber service
MME	Mobility management entity

List of Figures

1 Introduction

Based on the global project architecture as provided by WP2, WP4 is in charge of detailing the design, implementation and integration of two software-based mobile core networks. This work consists of two main parts: i) management and control-related functionality, and ii) components implementing the actual mobile core network. The first is provided by T4.2, the latter by T4.3. While D4.1 [1] has documented the design of the involved testbeds and associated SDN/NFV environments, this deliverable will report on the implementation work that has been carried out in T4.2 and T4.3 to actually provide software-based platforms on KR and EU side. As planned in the project proposal, the core of this work has carried out by the Korean partners.

The ultimate goal of this work is to demonstrate interoperability of both platforms and mobile core networks in the Proof of Concept in GangNeung due in February 2018. The deliverable follows the structure where section 2 details the implementation work of all MANagement and Orchestration (MANO) components, involved interfaces and associated processes, as well as preliminary evaluations that have been made with these platforms. The Korean partners have implemented a Distributed Cloud NFV Management System from scratch, supporting multiple service and network function models and descriptors, as well as a number of ETSI compliant interfaces between its subcomponents. Preliminary evaluations validate the scalability and performance of this platform. The EU partners rely on the OpenBaton SDN/NFV MANO platform for controlling its virtualized mobile core network. Interoperability between both platforms builds further on ETSI NFV ISG standardization work as provided by the interface between the NFVO and the VNFM.

The third section of this deliverable details implementations of the involved software-based mobile core networks. The main contribution here comes from the Korean partners, detailing the innovative distributed mobile core implementation and associated distributed mobility management functionality. The EPC that will be used by the European testbed for the interoperability scenario planned by the end of the project, is the OpenEPC platform as provided by Core Network Dynamics. Section 3 will document the internal components of this implementation, as well as its instantiation on the OpenBaton MANO platform.

As a result, this deliverable sets the stage for the latest phase of the project targeting interoperability of the software-based instances of the EPCs as controlled by their corresponding SDN/NFV MANO platforms. The actual interoperability of the documented implementations will be carried out at a later stage in the project, and will be document in deliverable D4.3.

2 Management and Orchestration

2.1 Global architecture and functionality

Defined in ETSI ISG NFV architecture, MANO (Management and Network Orchestration) is a layer that manages and orchestrates the cloud infrastructure, resources and services. It is comprised of, mainly, three different entities — NFV Orchestrator, VNF Manager and Virtual Infrastructure Manager (VIM).

The NFV Orchestrator is responsible for managing the functions such as network service life-cycle management and the overall resource management. Service management or orchestration deals with the creation and end-to-end management of the services — made possible by composing different virtual network functions (VNFs). Resource management helps in ensuring that the NFV-infrastructure resources are abstracted cleanly (independent of VIM) to support the services that access these resources.

The VNF Manager oversees the lifecycle (typically involves provisioning, scaling, terminating) management of instances of virtual network function (VNF). It is typically assumed that each VNF will be associated with a VNFM that will manage that particular VNF's lifecycle. A VNFM may manage multiple instances of the same type of VNF or different types of VNFs.

The Virtualized Infrastructure Manager (VIM), controls and manages the NFVI compute, storage, and network resources. The VIM-component has received tremendous focus and various open source solutions such as OpenStack and has been used to realize the virtualized infrastructure management functionality of MANO.

In summary, when MANO is seen as a single entity, the typical functionalities include (a) Infrastructure automation and providing consistent, accurate and global view of the resources, (b) Network integration, (c) VNF lifecycle management and its placement in in NFVI, (d) service management, (e) Performance monitoring, analysis and governance (auditing, compliance) support.

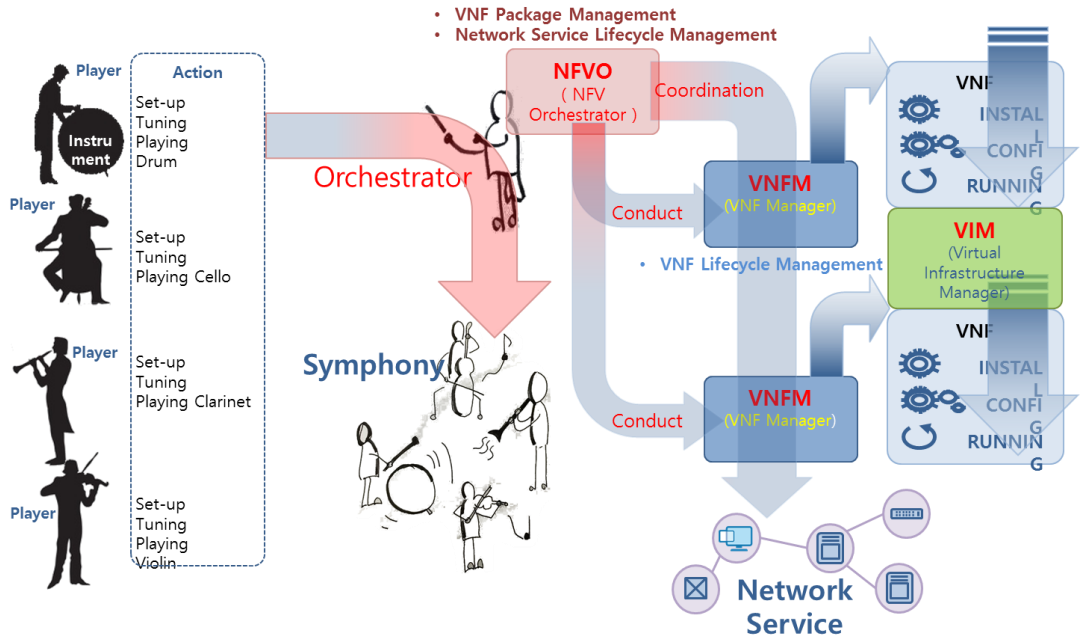

Figure 1 - MANO Architecture

2.2 KR Distributed Cloud NFV Management System

Characterized by its agility and flexibility 5G networks are required to adapt to the service to be deployed on demand. Smart Network Integrated Management & Orchestration (SNIMO) in the DCNM mediates service and network infrastructure connected with business and service support systems. SNIMO plays a role of integrated orchestration for distributed Micro DC (Data Center) by conducting multi-domain, multi-technology resource optimization. It manages not only network services deployable over the multiple micro DCs but also resources and functions in the individual micro DCs. Transport networks between micro DCs are also controlled by the multi-layer-multi-technology transport VNCM (Virtual Network Control & Management) functional block. The resulting architecture and deployment model on the Korean platform is depicted in Figure 2 and has been documented in D4.1 [1].

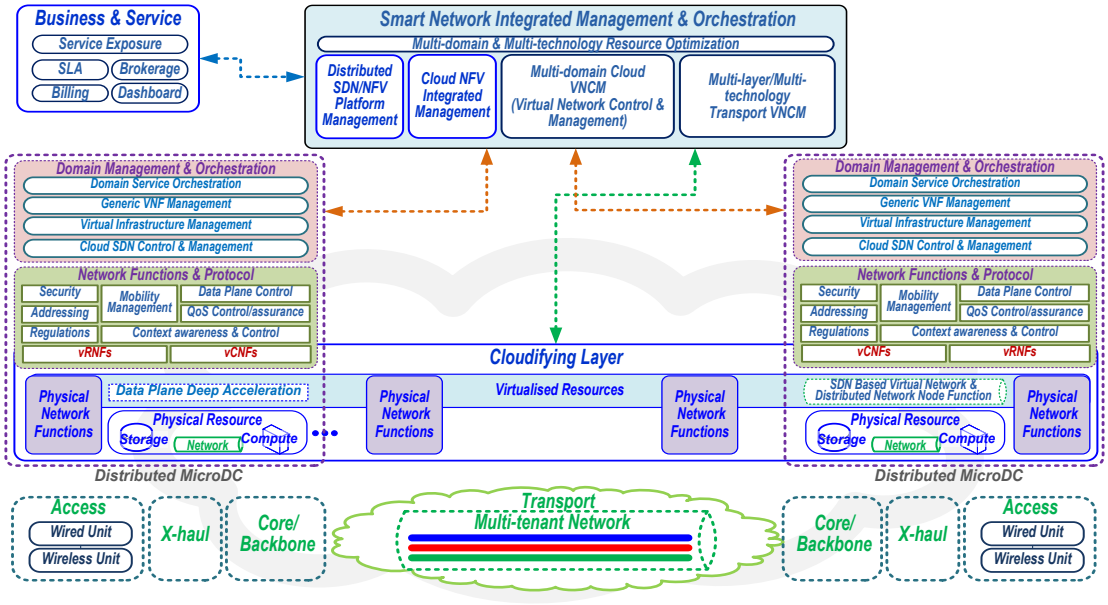

Figure 2 - Deployment model of DCNM

2.3 EU OpenBaton MANO platform

OpenBaton is an open source platform which is an implementation of the ETSI NFV Management and Orchestration specifications. It comprises an NVFO for network orchestration, a generic VNFM and a generic element management system (EMS) to manage the lifecycle of VNFs based on their descriptors. There are driver mechanisms supporting different types of VIMs e.g. OpenStack. It also has a monitoring plugin, an auto-scaling engine and a fault management system[1]. The architectural layout of OpenBaton MANO is illustrated in Figure 3 which depicts all the functional blocks within the framework.

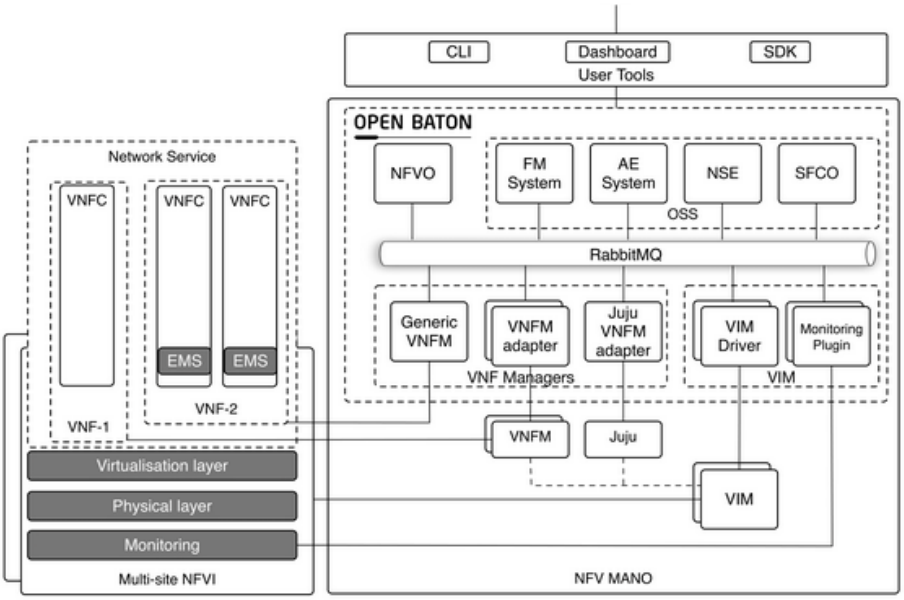

Figure 3 - OpenBaton architectural framework[1]

2.3.1 NFVO Architecture

The NVFO is a modular software component implemented in Java using the spring.io framework. Every module inside it is based on the Spring Framework and the communication between NFVI and other external modules is mostly based on RabbitMQ messaging system.[2]

[1] http://openbaton.github.io/documentation/
[2] http://openbaton.github.io/documentation/nfvo-architecture/

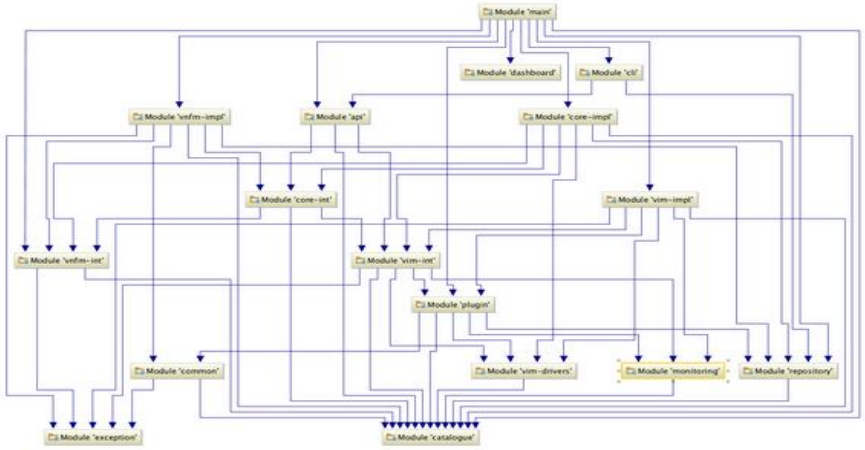

Figure 4 - OpenBaton NVO architecture[2]

The modules, along with their functionalities, within OpenBaton's NFVO are listed in Table 1.

Components	Functionality
API	Contains requisite classes for exposing APIs as REST server
MAIN	Contains classes which takes care of the startup of the whole system and gathering configurations
COMMON	contains classes common to the NFVO
VNFM-INT	Contains the interfaces of the core functionalities regarding NVFO interfaces to the VNFM
VNFM-IMPL	Contains the beans implementing vnfm-int interfaces
CORE-INT	Contains the interfaces of core functionalities regarding the NVO internal interfaces only
CORE-IMPL	Contains the beans implementing core-int interfaces
CATALOGUE	Contains the complete NFVO model in the OpenBaton libraries
PLUGIN	Contains the utility classes for the OpenBaton plugins interface
VIM-DRIVERS	Contains the interface for the VIM plugins
EXCEPTION	Contains the exception classes common to projects containing OpenBaton libraries
MONITORING	Contains the interface for monitoring OpenBaton plugins
SECURITY	Configures the security mechanisms used in NFVO
VIM-INT	Contains the interfaces of the core functionalities regarding NVFO interfaces to the VIM
VIM-IMPL	Contains the beans implementing the vim-int interfaces

Table 1 - Modules within OpenBaton's NFVO[2]

2.3.2 Generic VNFM

Working as an intermediate component between the NFVO and the VNFs, the generic VNFM is an implementation of ETSI GS NFV-MAN 001 V1.1.1 (2014-12) [2]. It interoperates with the Element Management System (EMS) to complete the lifecycle of a VNF. The EMS is actually an agent inside the VMs which executes scripts contained in a VNF package or defined through a link to the scripts inside the descriptor for each package. Communication between the NFVO and EMS is handled by the VNFM using the AMQP protocol over RabbitMQ[3].

The VNFM sends commands to the EMS and the EMS executes commands locally in the VNFC[3]. Figure 5 illustrates the flow of messages while instantiating virtual network functions.

The 'INSTANTIATE' message typically comprises a VNF Descriptor and some other parameters required to create VNF Records like Virtual Link Records. The message is directed towards the Generic VNFM for Virtual Network Function Record creation which is then sent back to the NFVO into a 'GrantOperation' message which triggers the NFVO to check availability of resources. If there are enough resources of that VNF Record, a 'GrantOperation' message with the updated VNF Record is sent back to the Generic VNFM. The Generic VNFM then forms an 'AllocateResources' message with the received VNF Record and sends it to the NFVO. Having created the VMs, the NFVO sends back the 'AllocateResources' message to the VNFM. Now, the scripts contained in the VNF Package are sent to the EMS and the Generic VNFM will call for the execution of each script defined in the VNF Descriptor. Once all scripts are executed without an error, the VNFM sends an 'Instantiate' message back to the NFVO[3]. The modify and start messages inside the flow as shown in Figure 5 again follow the same steps as explained earlier where the NFVO sends a message to the VNFM (modify/start/stop in this case). The VNFM then executes scripts in the VNF descriptor and returns it back to NFVO[3]. Since, VNF instantiation can either be done with resource allocation done by NFVO or with resource allocation done by VNF manager so the generic VNFM uses the former approach for VNF instantiation flows[3]. The following two types of messages will be sent to the NFVO:

GRANT_OPERATION message: It checks the availability of resources on a selected PoP. If the message is returned, then it is an indication of enough resources unless otherwise, an ERROR message will be sent[3].

ALLOCATE_RESOURCE message: It directs the NFVO to create all the resources. If no errors occurred, an allocate_resource message will be returned to the VNF manager[3].

[3] http://openbaton.github.io/documentation/vnfm-generic/

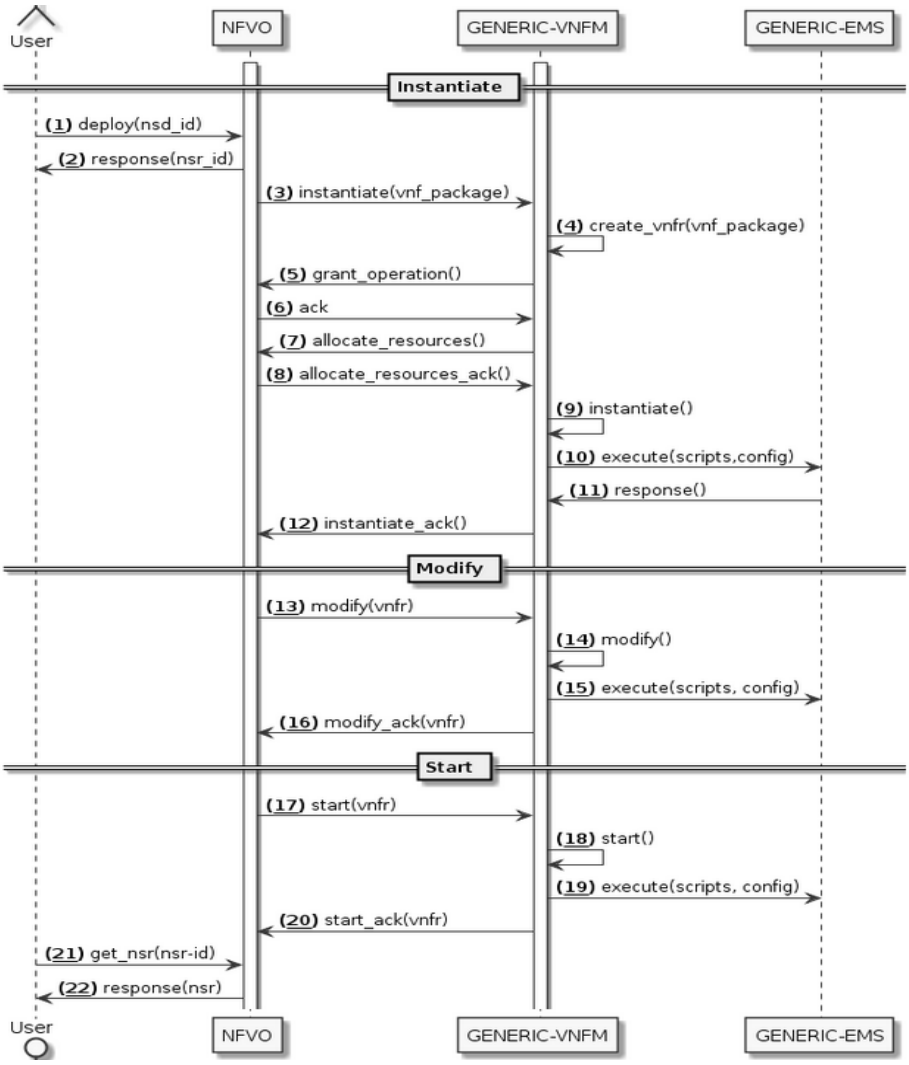

Figure 5 - Instantiation flows[3]

2.3.3 Virtual Infrastructure Management

The controlling and managing of NFVI compute, storage and network resources within a Point of Presence (PoP) is handled by the virtualized infrastructure manager (VIM)[4]. Before instantiating any resources on a PoP, we should inform the NFVO. Depending on the choice of VIM, different drivers are required. In our case we need an OpenStack VIM driver to integrate our OpenStack instance with OpenBaton, as they are running in different hosts. So, initially the installation of the OpenStack VIM driver is required.

[4] http://openbaton.github.io/documentation/pop-registration/

We can then register a PoP by writing a JSON script containing the details of the PoP. We can either use the CLI or the dashboard for this purpose. The name, auth-url, tenant/project ID, username, password and type are the parameters, mandatory while registering a PoP. It could be all added inside a JSON file or manually added while registering a PoP.

2.4 Service and Network Function Model

2.4.1 Meta-model

Both the EU and KR MANO platforms rely on their own particular service and network model and associated descriptors. Despite their peculiarities, they share a common set of concepts as defined by the meta-model shown in Figure 6.

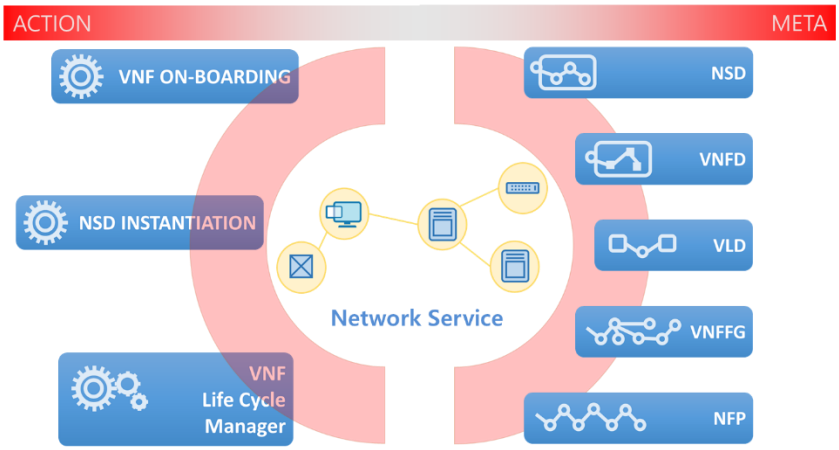

Figure 6 - Service and Network Function meta-model

2.4.1.1 Network Service model

The **Network Service Descriptor (NSD)** defines a set of interconnected VNFs to realize a network service spanning multiple VNFs. In addition, the NSD also defines the NS-level configuration information.

NS Connection Points: List for network service (NS) connection points. Each NS has one or more external connection points used to link the NS to other NSes or to external networks. Each NS exposes these connection points to the orchestrator. The orchestrator can construct network service chains by connecting the connection points between different NSes.

Constituent VNFs: List of Virtual Network Function Descriptors (VNFDs) that are part of this network service.

VNF Dependencies: List of VNF dependencies. This specifies the order in which the VNFs inside the NS should be started.

Virtual Links: List of Virtual Link Descriptors (VLD). The VLD describes how VNFs in the NSD are connected.

NS Configuration Primitives: Network Service level configuration primitives.

2.4.1.2 VNF model

The **VNFD** connects Virtual Deployment Units (VDUs) using the internal Virtual Links (VLs). Each VDU represents a VM. The VDUs attach to the internal VLs using the internal Connection Points (CPs). So the VNFD captures the list of VDUs and the internal VLs that connect the VDUs. Note that for a VNF containing a single VDU, there may not be any internal VLs.

VNF Connection Points: The list for external connection points. Each VNF has one or more external connection points. As the name implies the external connection points are used for connecting the VNF to other VNFs or to external networks. Each VNF exposes these connection points to the orchestrator. The orchestrator can construct network services by connecting the connection points between different VNFs. The NFVO will use VLDs and VNFFGs at the network service level to construct network services.

Constituent VDUs: List of virtual deployment units. VDUs refer to individual VMs inside the VNF. They are also referred to as VNFCs.

VDU Dependencies: List of VDU dependencies. The orchestrator uses this list to determine the order of start-up for VDUs.

Internal VLDs: A list of internal virtual links to connect various VNF components.

VNF Configuration Primitives: VNF level configuration primitives.

Monitoring Parameters: List of monitoring parameters for the VNF

2.4.2 VNF orchestration of a Network Service

The orchestrator is needed to manage the relationships between VNFs which are sthe constituent component of Network Service (NS) to support a change of NS flavor or auto-scaling.

VNFs existing in network services are dynamically evolving into virtual networks that might be further subdivided and involving functionally depending on each other. As the dynamics in the network increases, the network slice that accommodates various network functions encounters a number of challenges. The core of this challenge

relates to automated processing capabilities. How to ensure the dependencies between VNFs to provide various and complex virtual network services?

The VNFs configured in the network service do not cause problems when they are created separately and can function independently. However, some information might need to be transferred between the VNFs in order to ensure correct operation.

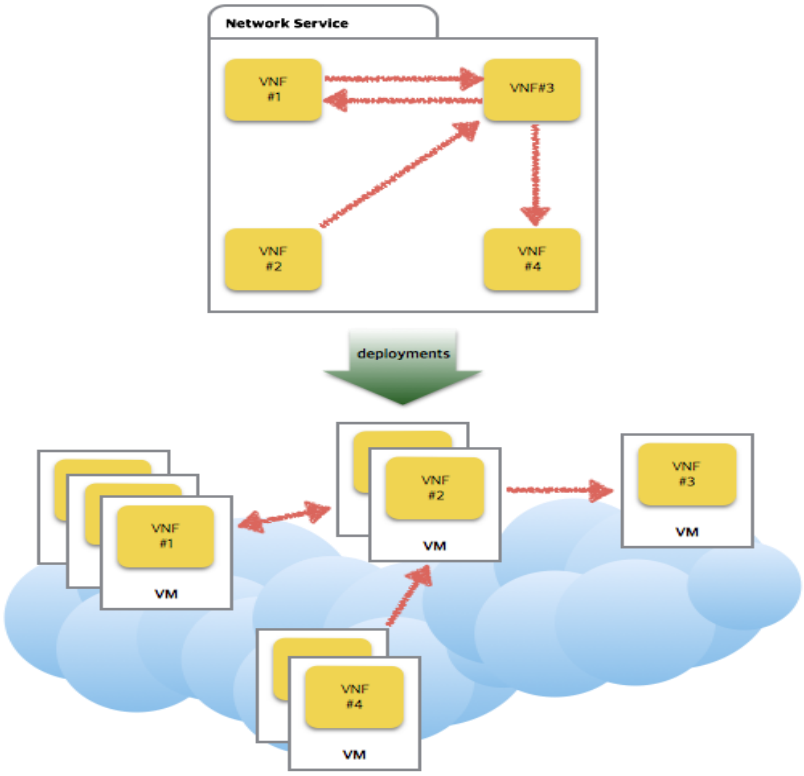

Figure 7 – Inter-VNF information dependencies

The role of the orchestrator is to provide the supporting information management system. It requires information delivery methods and processing mechanism to maintain and assure correct correlation between VNFs

This is a two-way method, i.e., a method of providing necessary information according to an access method of information (Call by value) and a method of providing a reference value of necessary information (Call by reference).

The first case is that an application calls by reference to receive information from other applications.

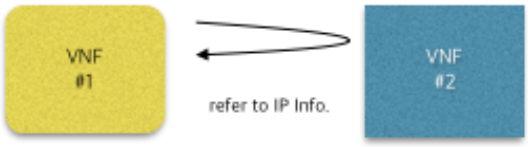

Figure 8 – Call by reference for inter-VNF communication

As shown in the above figure, when the VNF # 1 requires the IP of the VNF # 2 as its set value, the setting of the VNF # 1 is completed by referring to the IP information using the registered VNF # 2 information.

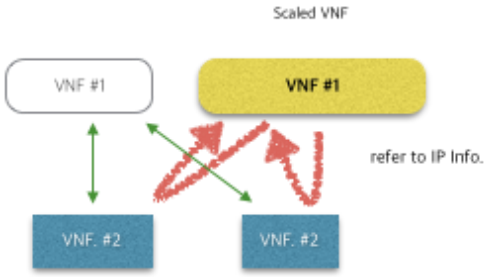

Figure 9 – Collect information during inter-VNF communication

When the VNF is added at the front end of the service, it collects the VNF information at the bottom and automatically sets it so that it does not become a failure to participate in the service.

Figure 10 – Additional VNFs share reference information to other VNFs

The other case is that an application provides the reference information to other applications. As shown in the following figure, when VNF # 2 is newly created, information is transmitted to VNF # 1, which should refer to the information of VNF # 2, and the configuration information is provided so that it can participate in the service.

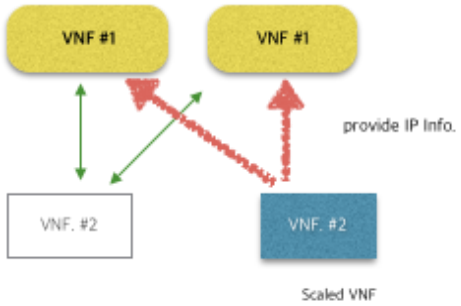

Figure 11 – Announcement of IP information by new VNF

Since the extension information of the VNF at the lower end of the service is unable to be known, the load of the service is possibly to automatically distributed by providing the information of the added VNF at the lower level.

These VNFs communicate with each other so that the orchestrator can provide flexible management of interdependency between VNFs.

2.4.3 OpenBaton descriptors

2.4.3.1 NS Descriptor

The network service descriptor (NSD) in OpenBaton is in compliance with ETSI GS NFV-MAN 001 V1.1.1 (2014-12) and is used by the NFVO for deploying network services. It supports both JSON file representation and TOSCA templates. The following: name, vendor, version, vnfd, vld and vnf_dependancy are some requisite parameters while making an NSD. The virtual link descriptor (VLD) depicts the virtual links required for inter VNF connectivity. The VNFD uses the network description defined in the VLD while creating network[5].

2.4.3.2 VNF Descriptor

Each VNF package has a virtual network function descriptor (VNFD) which the NSD will also have as part of description for each package. The following: name, vendor, version, type and endpoint are required parameters inside a VNFD. Also, it has virtual deployment unit (VDU) which comprises of image list, PoP list, scale_in_out, vnfc. When a network service is launched, each VNFC will run on a different virtual machine. Apart from that, there are also other parameters like virtual link (pointing to a VLD

[5] http://openbaton.github.io/documentation/ns-descriptor/

defined in the NSD), Lifecycle events (INTANTIATE, CONFIGURE, START, TERMINATE; SCALE_IN), Deployment Flavor and inter-VNF dependencies[6].

2.4.3.3 Network Service Record

Initially, the network service record (NSR) created upon launching the network service descriptor is in NULL state. Once the instantiation is finished in the VNFM, the virtual network function records (VNFR) will be set to 'INSTANTIATED'. The NSR will also change to the INSTANTIATED state when all the VNFR are in INSTANTIATED state. Now, when the START message is sent to the VNFM and returned, the NFVO sets the VNFR state to ACTIVE. When all the VNFR are in ACTIVE state, the NSR will also change its state to ACTIVE. The similar change of state will be followed in case of MODIFY and TERMINATE states[7]. The VNF record states is illustrated in Figure 12.

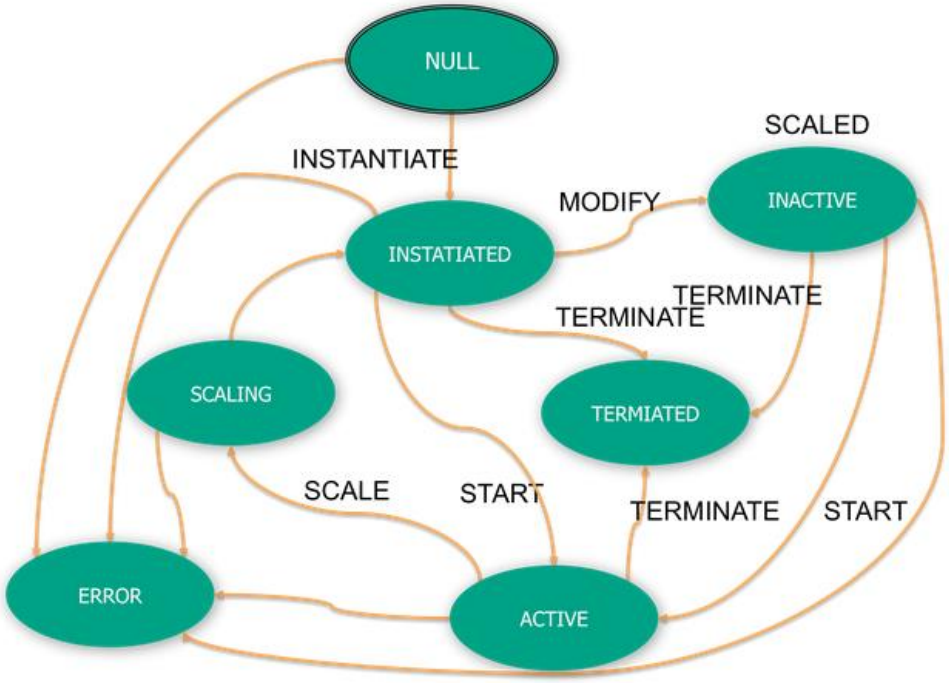

Figure 12 - VNF Record States[7]

[6] http://openbaton.github.io/documentation/vnf-descriptor/
[7] http://openbaton.github.io/documentation/vnfr-states/

2.5 Management and orchestration processes

2.5.1 Overall message flow for Network Service Creation

Network Service creation starts with checking NSDs from the Descriptor Repository initiated by NFVO. NFVO has interfaces both with VIM for resource management and NFVM for VNF management as well as repositories to keep NSDs and records of Network Services. VIM manages heterogeneous resources to create virtual containers for accommodation of vEPC VNFs by the allocation of the resources amount of specified on the VNF descriptors. After the completion of virtual resource allocation for vEPC VNFs, VNFM notifies the event with initial configuration parameters to VNFs to finalize the vEPC instantiation.

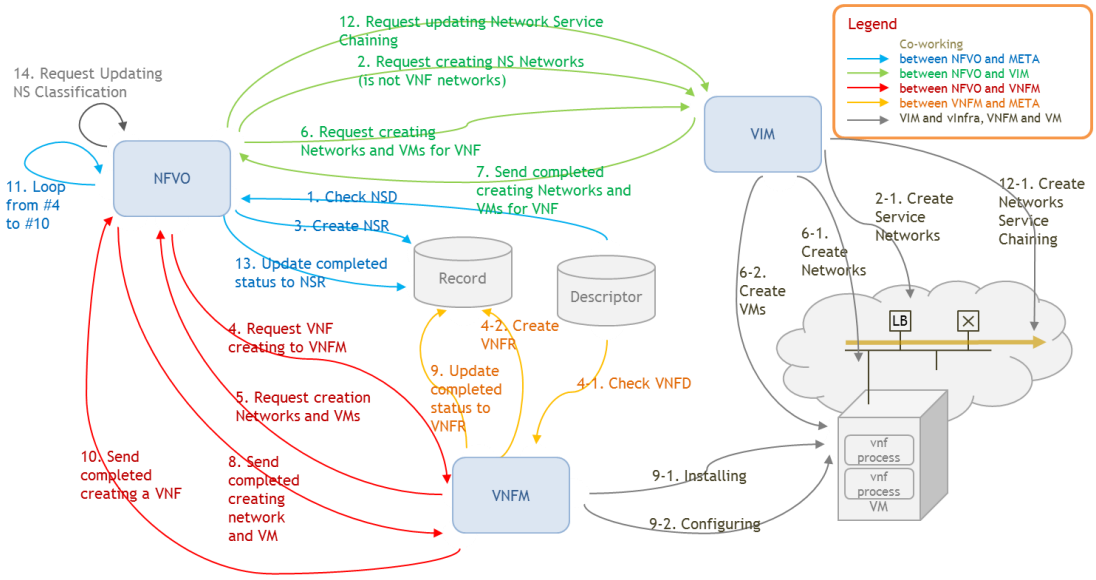

Figure 13 - NFV MANO Functional Blocks and Interfaces

2.5.1.1 Auto scaling based on PM/FM

For reliable 5G mobile core network service (NS), Performance Management (PM) and Fault Management (FM) over NS and constituent VNFs are crucial in terms of MANO. After instantiation of the 5G mobile core NS, NFVO sends supervision request to the Supervisor which performs monitoring and notification of performance, fault, alarm and indicator over virtualized resource and functions. Scaling is conducted autonomously by orchestrator based on the information notified from the Supervisor.

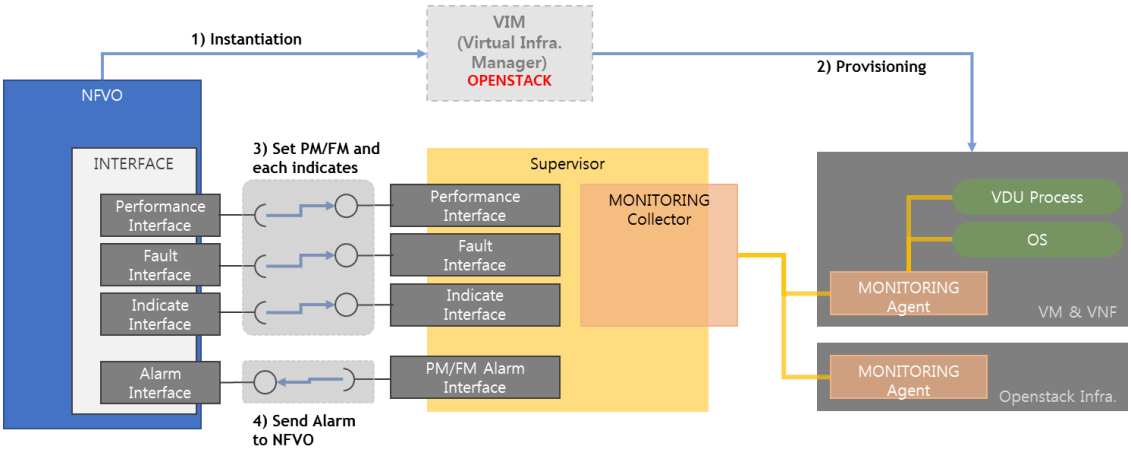

Figure 14 - KR MANO Auto Scaling Process

2.5.1.2 Automation by event chaining

Event chaining process which is defined as the sequence of event units occurring from inside or outside of the target VNF and VDU enables automation of the 5G mobile core network management.

A combination of internal events significant in the single VNF or VDU and external events between VNFs and VDUs enables automated management of lifecycle of mobile core network service.

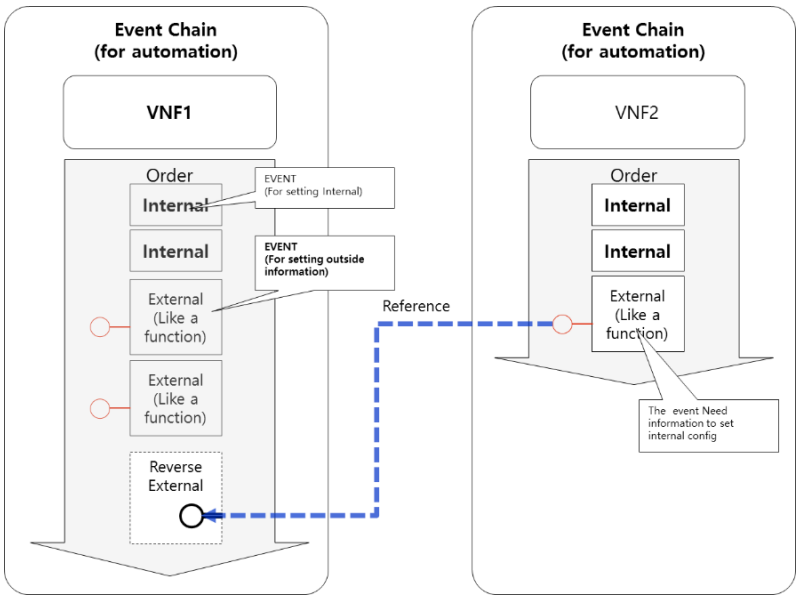

Figure 15 - KR MANO Event Chaining Process

2.6 Management and orchestration interfaces and interoperability

The ETSI NFV ISG has defined several Management and Network Orchestration (MANO) between functional blocks inside and toward outside entities: Os-Ma-Nfvo, Or-Vnfm, Or-Vi, Vi-Vnfm, and Ve-Vnfm-vnf/em. The Os-Ma-Nfvo interface is used for exchanges between the OSS/BSS and the NFVO to support network services and VNF package management. The Os-Vnfm interface is used for exchanges between the NFVO and VNFMs, and supports VNF instance management. The Ve-Vnfm-vnf/em interface is used for exchanges between VNFMs and VNF/EMs to manage VNF instances. The Or-Vi is the interface between the NFVO and VIMs for virtualized resource management. The Vi-Vnfm interface also aims to manage virtualized resources, but it is used for exchange between VNFMs and VIMs. The interfaces are illustrated in Figure 16.

ETSI NFV ISG has standardized information models for those interfaces: IFA005, IFA006, IFA007, IFA008, and IFA013. In particular, ETSI NFV ISG defined RESTful protocols and data models based on the information models for some interfaces: SOL002, SOL003, and SOL005. The following summarizes those specifications.

- IFA005 provides an information elements associated to the Or-Vi interface.
- IFA006 provides an information elements associated to the Vi-Vnfm interface.
- IFA007 provides an information elements associated to the Or-Vnfm interface.
- IFA008 provides an information elements associated to the Ve-Vnfm-vnf/em interface.
- IFA013 provides an information elements associated to the Os-Ma-Nfvo interface.
- SOL002 presents a set of RESTful protocol specifications and data models fulfilling the requirements specified in IFA008.
- SOL003 presents a set of RESTful protocol specifications and data models fulfilling the requirements specified in IFA007.
- SOL005 presents a set of RESTful protocol specifications and data models fulfilling the requirements specified in IFA013.

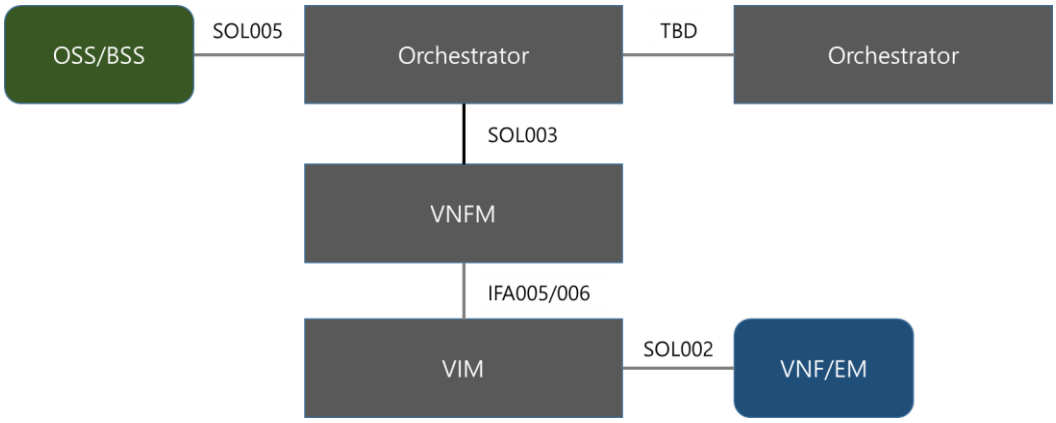

Figure 16 - NFV MANO Functional Blocks and Interfaces

2.6.1 Management and orchestration platform interoperability

Interoperability between multiple SDN/NFV MANO platforms is an active research topic in multiple research projects (e.g., 5GEx[8]) and standardization bodies (e.g., ETSI, IETF). MANO platforms can either interoperate via their NFVO's. However, no standardization body or research project has yet converged on a commonly agreed interface for NFVO interoperability as indicated in Figure 16. Alternatively, multiple platforms can interact using the NFVO-VNFM interface relying on the recently finalized ETSI NFV SOL03 [3] interface. Because this is the most stable and well defined interface currently available, supporting a range of events and associated interoperability, this is the approach followed by WP4 in the 5GCHAMPION project.

The approach taken relies on a global NFVO in charge of orchestrating two virtualized mobile core networks, the KR DMM on one side, and OpenEPC on the other side. Both implementations are documented in detail in the third section of this document. To coordinate the lifecycle of both vEPCs, NFVO relies on the SOL03 interface interconnecting the global NFVO with both the KR VFNM in charge of the DMM instance and the EU VNFM in charge of the OpenEPC instance. Each of these VNFMs is responsible locally for the management of the lifecycle of its VNFs, its virtualized resources and its configuration, in interaction with the VIM and NFVO.

As depicted in Table 2, ETSI NFV IFA007/IFA008 defines a range of lifecycle management operations, including instantiation, healing, scaling and termination of VNFs. Support of certain operations by a concrete VNF may depend on the capabilities of the VNF itself (e.g., whether a VNF is "scalable").

[8] http://www.5gex.eu/

Operation	Support by VNF	Explanation
Instantiate VNF	Mandatory	Allocate virtualised resources, configure them, start the application, trigger configuration of the application.
Scale VNF	Optional	Change the amount of virtualised resources allocated to a VNF.
Query VNF	Mandatory	Obtain runtime information about the VNF instance (VnfInfo).
Terminate VNF	Mandatory	Terminate the VNF, and release the virtualised resources.
Change VNF flavour	Optional	Change the deployment flavor of the VNF, which typically includes changi ng the amount of virtualised resources, and the topology.
Heal VNF	Optional	Virtualisation-related corrective actions to repair a faulty VNF, and/or its V NFC instances and internal VNF Virtual Link(s).
Operate VNF	Optional	Start or stop the VNF software.
Modify VNF Info	Mandatory	Change certain items of the VNF runtime information (VnfInfo).
Auto-Scale and Auto-Heal	Optional	Variants of Scale VNF and Heal VNF, triggered automatically in the VNFM, by monitoring the VNF

Table 2 – VNF Lifecycle Management Operations

The VNF lifecycle operations listed in Table 2 can be triggered by the (global) NFVO relying on task resources according to the general lifecycle management flow provided by the SOL03 standard as depicted in Figure 17. For the details, we refer to the corresponding standard document, however below we briefly document the main process sequence. In the proposed setup, VNFMs expose a REST API enabling to refer to individual VNF instances and a requested task resource. Task resources are associated to the potential operations in Table 2 (e.g., instantiate, scale, heal, etc.). As indicated in Figure 17, the NFVO can trigger a lifecycle-event by sending a request to the resource corresponding to the required event (once the NFVO is subscribed to the VNFM and authentication has been performed through the granting mechanism). In response of this request, the VNFM creates a new corresponding LCM operation occurrence resource (which can be polled by the NFVO). Depending on the particular lifecycle event (e.g., instantiation vs. scaling), packages might need to be onboarded, resource allocations might be executed, and associated references might be exchanged.

The KR MANO platform and VNFM implementations provide full compliance of the referred mechanism and SOL03 interface. On the other side, the Generic VNFM of the OpenBaton EU MANO platform, as documented in Section 2.3.2, supports a very similar workflow through its own implementation of either a REST API or RabbitMQ interface. In order to enable interoperability, the available Generic VNFM had to be modified to support the SOL03 workflow and interface. Because the focus of the later deliverable D4.3 is exactly this interoperability, more detailed information on the adapted process and interaction can be found in D4.3.

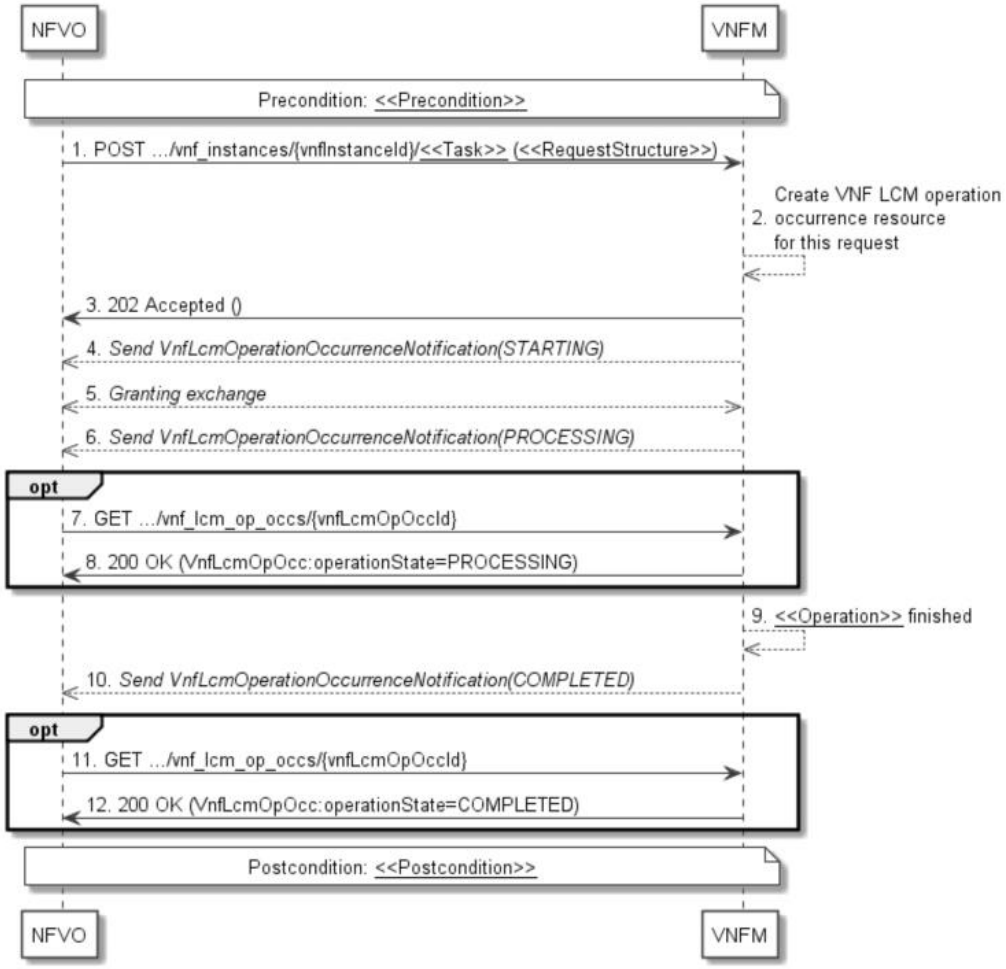

Figure 17 - General lifecycle management flow as provided by ETSI NFV SOL03

3 Virtualized Mobile Core implementation

3.1 Distributed Mobile Core

Distributed Mobility Management (DMM) is proposed to split the mobility anchor function to the edges of a network which are close to users. Standardization of DMM is ongoing work in the IETF (Internet Engineering Task Force) DMM working group. According to the DMM working group, distributed mobility management requirements are as follows;

- **On-demand mobility**: DMM should provide per-session mobility management. In current mobility management schemes, all sessions of a mobile device are supported with the same level of the mobility management. In DMM, a mobile node can select their IP address depending on an application's mobility needs. Another part is that the DMM network triggers mobility management operations only when mobile nodes detach their home anchor and attach to a visited network node.
- **Control/Data plane separation**: In DMM, mobility functionalities which were deployed in a current centralized anchor are decomposed into control and data plane functions. According the CP/DP separation, several CP/DP deployment models are possible to adopt. Between the CP and DP, a specific interface is needed to configure a forwarding path of mobile node's traffic. CP/DP separation of DMM functions enables to distribute a signalling path and a data traffic path in a network and to easily adapt in a SDN/NFV based network environment.

3.1.1 Architecture

Handover management: Specifically, in the MHN, a fast handover mechanism is required even in high-speed movements keeping user data rate for application service. To this end, the designated handover scheme using policy-based SDN technologies is exploited and used for the PoC of the mobility management.

Distributed Mobility Management (DMM): Functional decomposition & distribution for global service management will span multiple PoPs (Point of Presence) over the network. Anchoring and mobility management tailored to a network slicing instance will be determined at the central node. Composition functions and resource will be orchestrated for dynamic mobility management.

Distributed Mobility Management technology is used to distribute and place IP anchoring functions over the network to address issues stemming from conventional centralized single anchoring architecture such as Mobile IP (MIP), Proxy Mobile IP (PMIP). There are 5 DMM architectural models applicable to mobile core networks of 5GCHAMPION.

3.1.1.1 Split Home Anchor Model

The SGW remains as same as conventional integrated entity, the PGW distributes signaling path and data path by split control plane with data plane.

- SGW distinguishes S5 interfaces to PGW CP and PGW DP separately
- Destinations of the S5-C and S5-U protocols must be set differently in the SGW
- Sx interface supports interaction between PGW-CP and PGW-DP
- PGW-CP supports Gx interface for exchange of control message with PCRF, SGi interface supports PGW-DP connection to PDN

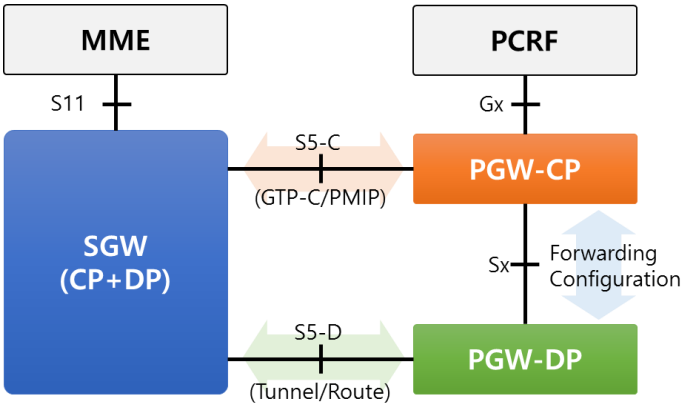

Figure 18 - Split Home Anchor Mode

3.1.1.2 Separated Control and User Plane Model

There are separated CP and DP for both SGW and PGW with Sx interface between CP functions and DP functions for each GWs.

- Conventional S5 interface splits into S5-C and S5-U for interconnection of CPs and DPs respectively.
- Sx interface supports interaction between PGW-CP and PGW-DP and also between SGW-CP and SGW-DP
- S11 interface supports SGW-CP and MME as control interface and SGW-DP has S1-U interface with eNodeB
- PGW-CP supports Gx interface for exchange of control message with PCRF, SGi interface supports PGW-DP connection to PDN

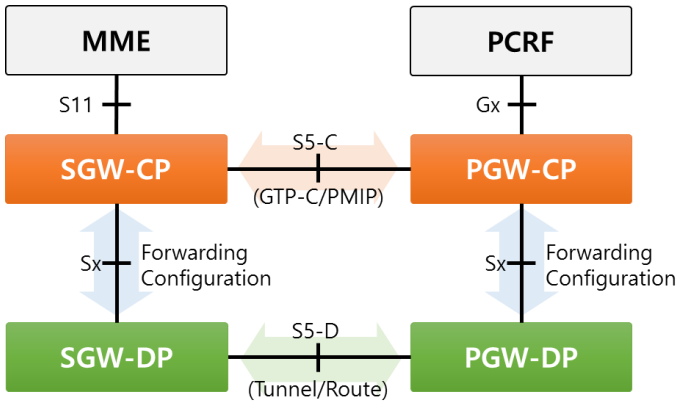

Figure 19 - Separated Control and User Plane Mode

3.1.1.3 Centralized Control Plane Model

The Centralized Control Plane entity supervises multiple Data Plane entities across the distributed networks.

- S5 interface implements only data plane-related interface through S5-U interface while conventional control plane interfaces over S5-C merges into the internal interface in the centralized control plane.
- Entities for centralized control plane have independent Sx interface with SGW-DP and PGW-DP respectively.
- CCP entities are interconnected with control entities such as MME and PCRF through S11 and Gx interface respectively

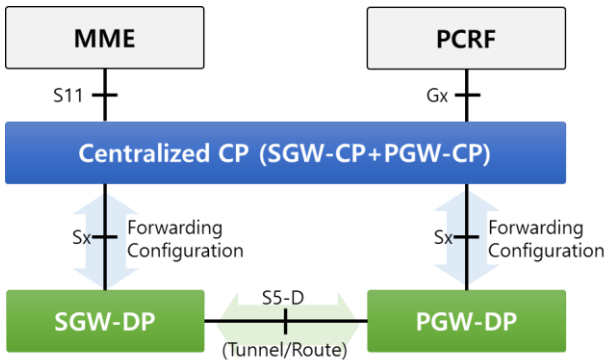

Figure 20 - Centralized Control Plane Mode

3.1.1.4 Data Abstraction Model

In the context of Data Abstraction Mode, a routing controller, for example, the SDN controller, provides upper CP functions with abstraction information of the SGW-DP and the PGW-DP. CP functions in the upper layer enforce routing setup information to the routing controller.

- Routing Controller should support establishment of data paths with data plane entities for southbound using specific protocol such as Sx or OpenFlow while providing configuration information to the control plane functions northbound.
- Data transmission between Data Plane functions may employ tunnel or policy based dynamic forwarding mechanisms.

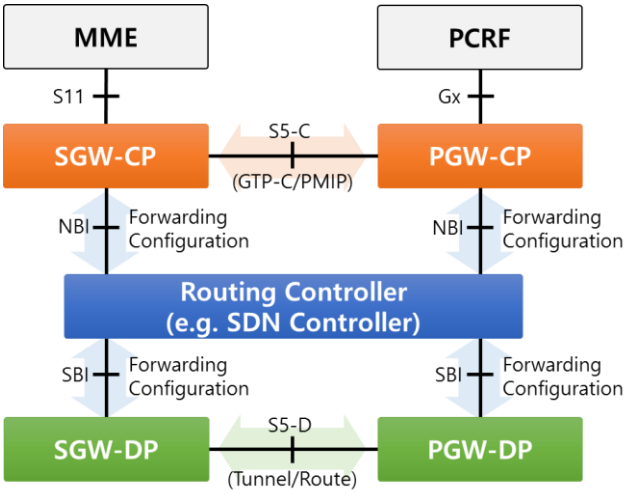

Figure 21 - Data Plane Abstraction Mode

3.1.1.5 On-demand Control Plane Orchestration Model

Control Plane entities for SGW and PGW are allocated by Mobility Controller dynamically at the time of UE attachment to networks. Data Plane functions are abstracted and managed by Routing Controller based on region and policy.

- On receiving a routing setup request message from MME when UE attaches to the networks, Mobility Controller allocates SGW-CP and PGW-CP based on the policy.
- When allocated CP functions decide forwarding rules and push it to the Routing Controller, the Routing Controller forwards these rules to the appropriated DP functions through a southbound interface.

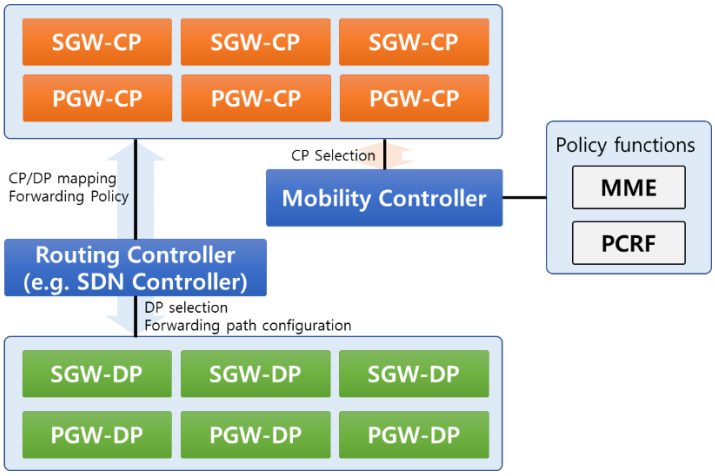

Figure 22 - On-demand Control Plane Abstraction Mode

3.1.2 Evaluation

For evaluation of distributed mobility management in vEPC environment, we assume that vEPC components are deployed independently on each edge PoP. Handover between PoPs means that the PGW of the previous PoP performs as a home anchor and the PGW of the new PoP acts as an access node. Until now, we did not modify current EPC components yetm but adding new components to provide DMM mechanism between two edge PoPs.

3.1.2.1 Testbed Design

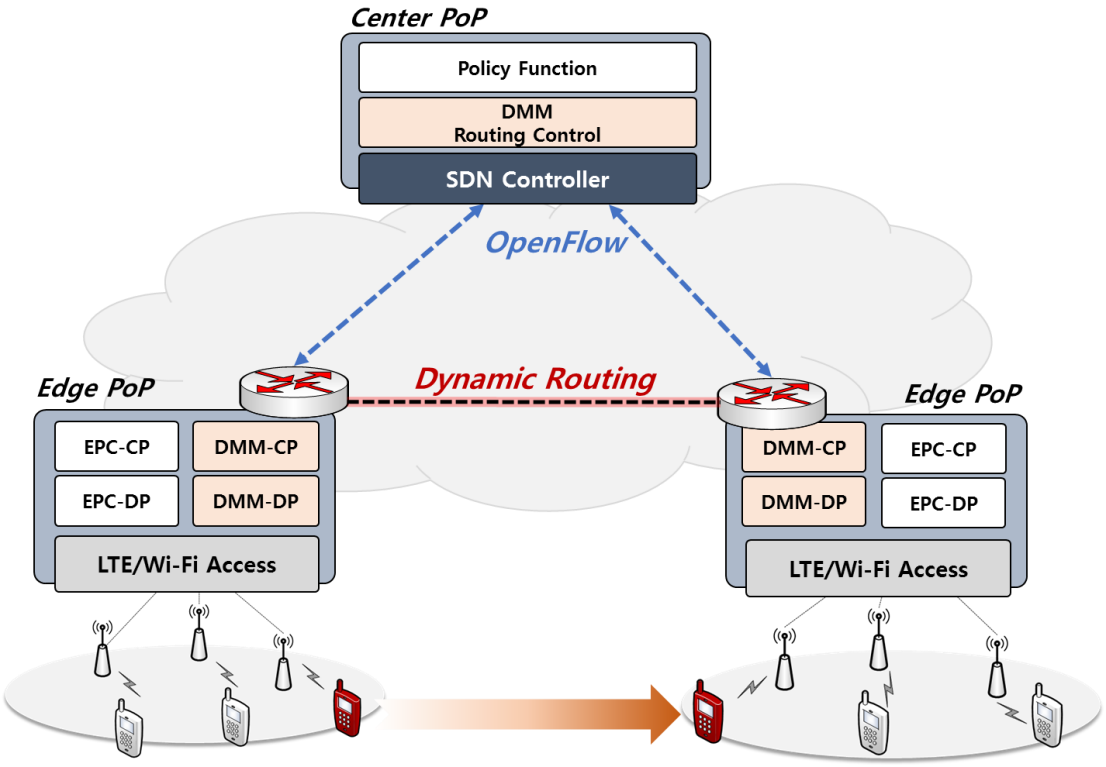

Figure 23 - Testbed for DMM

- DMM CP functions of edge PoP exchange mobility signaling messages with the DMM Routing Control function located in the core PoP to query information of current MN's location, address of anchor node. The DMM CP function also updates location of the MN to the DMM Routing Controller when the MN attaches to its PoP.

- DMM DP function is deployed as a GW of its edge PoP and it forwards traffic of MN between edge PoPs by using dynamic routing configuration based OpenFlow.
- For SDN-based routing, the SDN controller in the center PoP configures forwarding rules based on policy function in the center PoP and movement of MN.

3.1.2.2 Message flow

3.1.2.2.1 Initial attachment flow

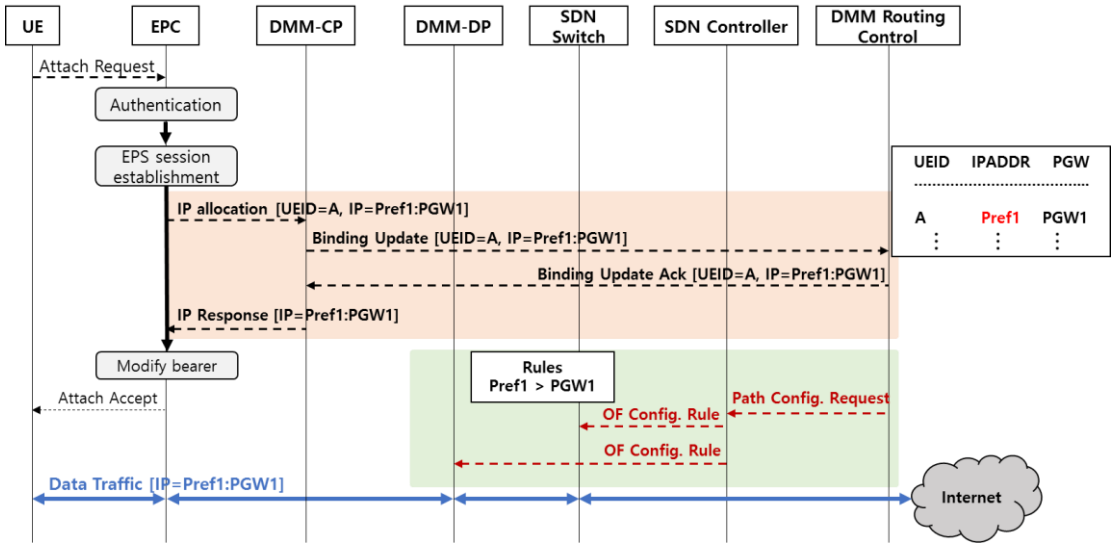

Figure 24 - Initial attachment message flow

- In the edge PoP, authentication procedure and the EPS bearer establishment process are performed in a standardized way
- The DMM routing function in the center PoP, determines that the UE attaches at the first in the network, updates information of the UE in the binding table, and sends the Binding Update Ack message to the DMM-CP.
- When the DMM binding table is updated, the corresponding information is sent to the SDN controller to push the forwarding rules to the DMM-DP at the edge PoP using OpenFlow Config message.

3.1.2.2.2 Handover flow

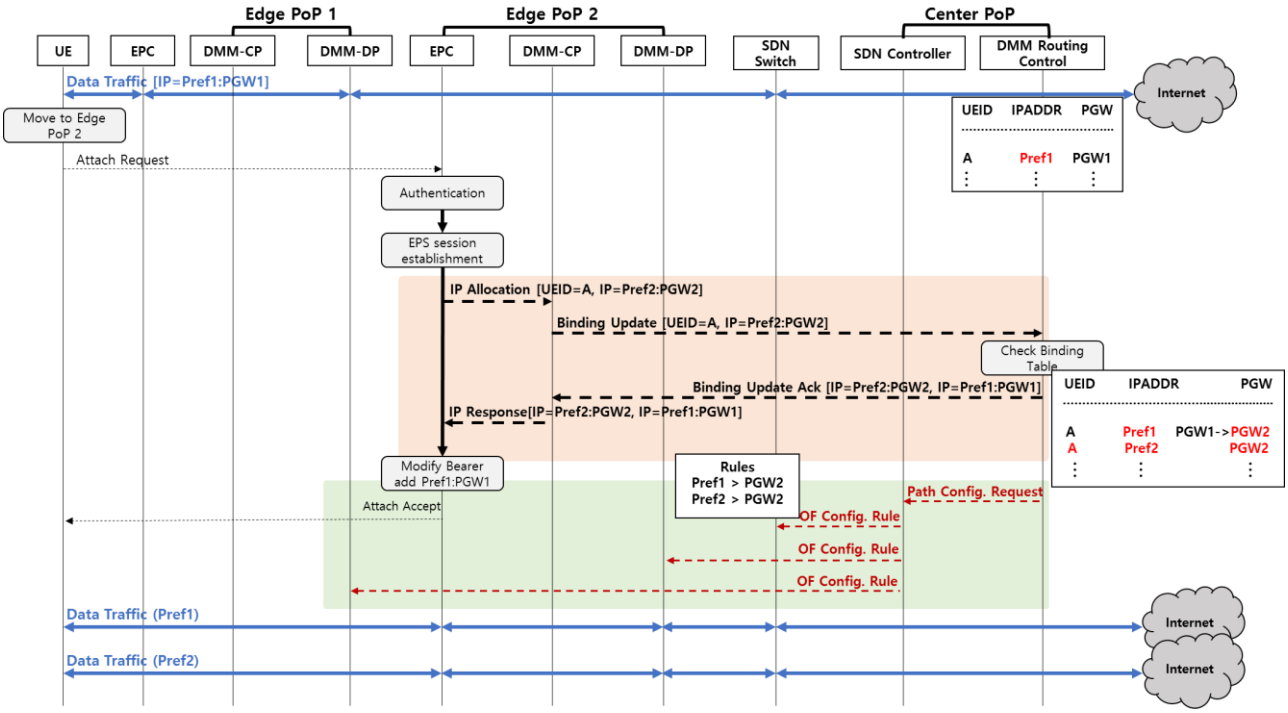

Figure 25 - Handover message flow

- The DMM Routing Control function receives information of the UE included in the Binding Update message from the DMM-CP of edge PoP2 and it can determine that the UE is moved from edge PoP1 to edge PoP2.
- Since that information of UE is already existing in the binding table, the DMM Routing Function updates the current location of the UE and assigns a new IP address to the binding table, and sends the Binding Ack message including IP address assigned at the edge PoP1 to the DMM-CP of the edge PoP2.

3.2 OpenEPC

OpenEPC is a software emulation of LTE evolved packet core (EPC) and is in compliance with 3GPP release 11/12. It is developed and maintained by Core Network Dynamics, offering professional services. All the functional elements of EPC could be instantiated as virtual machines inside OpenEPC setup. The setup could be deployed as standalone on a single PC using hypervisors e.g. VMWare Workstation or on servers e.g. VMWare ESXi, OpenStack. The OpenEPC function elements are illustrated in Figure 26.

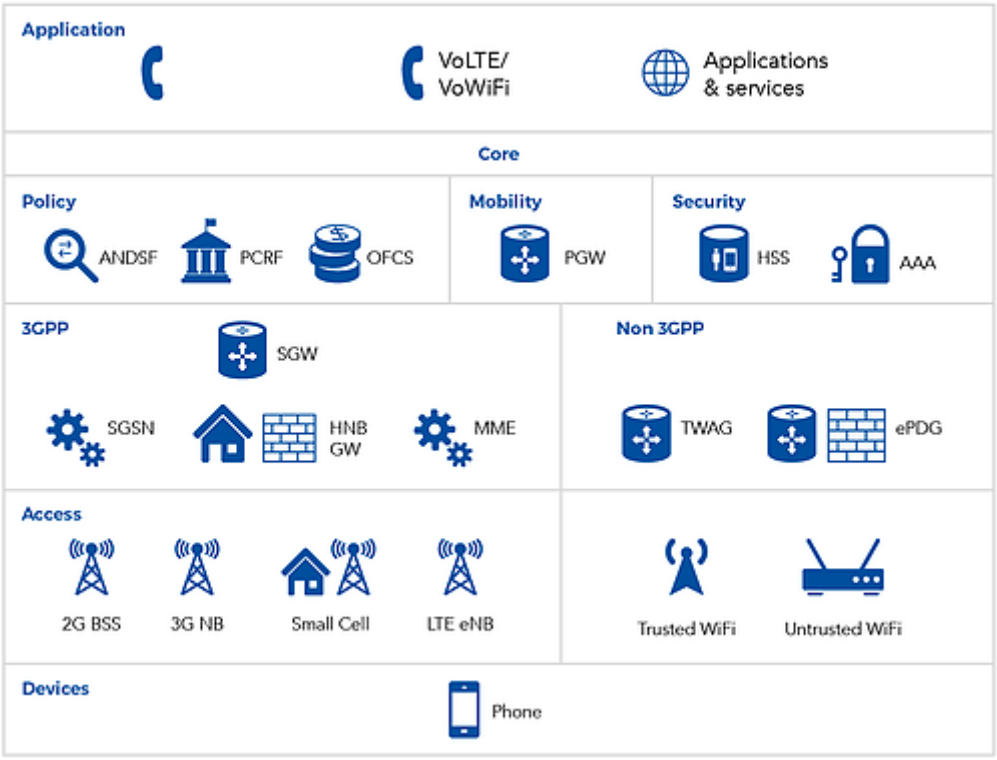

Figure 26 - OpenEPC functional Elements (copyright core network dynamics) [4]

3.2.1 OpenEPC Functions

Some of the primary OpenEPC functions are defined below.

HSS: The Home subscriber unit is in compliance with 3GPP release 12 and comprises interfaces such as S6a, S6c, S6d, SWx, Cx and Sh. It has Diameter Front-End and a Database as back-end. It has multiple provisioning interfaces like REST/JSON and a WebGUI. It is capable of both horizontal and vertical scalabilities [4].

MME: The MME in OpenEPC has a carrier-grade optimized software architecture in compliance with 3GPP release 12. It comprises S1-MME, S10, S11, S3, S6a, SLd, SGs, SGd, Sm and M3 interfaces. It follows procedures such as Attachment/Detachment, TAU, Handovers, Dedicated bearer, Multiple PDN connections etc [4].

SGW/ PDN-Gw: The serving gateway and packet data network gateway in OpenEPC have S5, S8, S11, S2a, S2b, S4 and S6b interfaces. They also support mobility protocols such as GTPv2 and PMIPv6. The gateways in OpenEPC have support for multiple SGi interfaces and dynamic switching [4].

PCRF and PCC architecture: OpenEPC has a PCRF for VoLTE which is in compliance with 3GPP release 12. Rx, Gx and Sp are the available interfaces. It has PCC API which also includes public safety related features. It selects a subscriber based on rules already defined in the subscriber profile and from the APN. It also has a NetLoc support and performs event reporting to application functions [4].

3.2.2 OpenEPC LAN segments

OpenEPC uses different LAN segments having different subnets on each network segment. The typical OpenEPC network setup is illustrated in Figure 27. OpenEPC's MESH topology allows deployment of of a complete core network along with the RAN[9].

net_a: It is the PDN and it connects the PGW with the application functions. Usually it is the NAT network which gives access to the internet [4].

net_b: It is the operators backhaul, connecting the access network gateways and the PGW [4].

net_d: It is the 3GPP radio access backhaul and it connects RAN with the EPC core network [4].

mgmt: It is the management and signaling network. All the signaling i.e. Diameter goes through this network and it serves as a management network for accessing each VM [4].

[9] http://www.openepc.com/decentralized-core-networks/

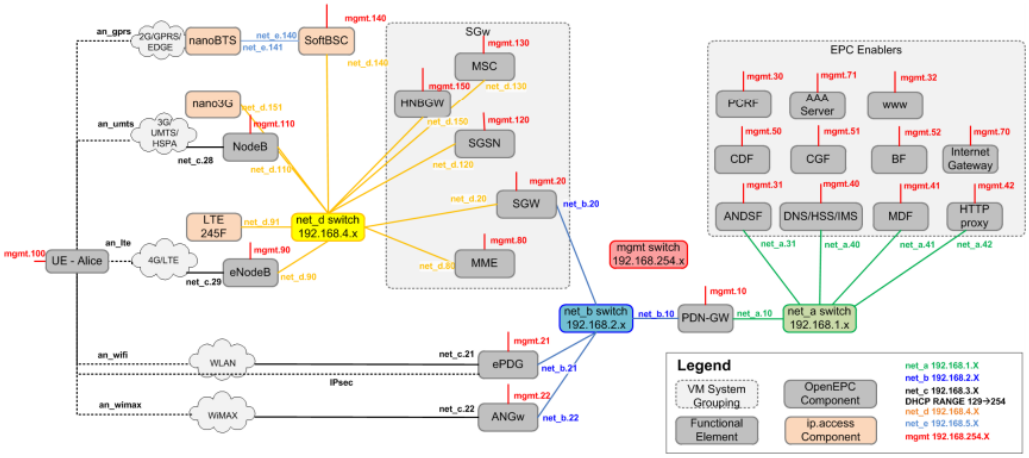

Figure 27 - OpenEPC architecture (copyright core network dynamics) [4]

3.3 EU architecture overview

The 5G test network at the University of Oulu, is an open network providing an infrastructure to its consortium partners which could be used for building a platform for 5G and beyond[10]. The network includes radio access network (RAN) and multiple core networks (EPC). OpenEPC, is one of the core element used for different research activities. 5GTN with OpenEPC as a core component provides LTE data service along with voice over LTE (VoLTE) and IMS functionalities. Also, Policy control and charging capability ensures a better quality of service (QoS). To make the network highly scalable and dynamic, the MANO platform is built around OpenEPC in 5GTN using OpenBaton and OpenStack. OpenEPC is integrated with OpenStack by making use of the OpenBaton MANO platform which provides an NFVO and a VNFM where Open Stack serves as a VIM. The OpenStack VIM driver in OpenBaton ensures the integration by making a Point of Presence referring to the OpenStack endpoints. The platform is illustrated in Figure 28 where OpenEPC network components i.e. EPC-Enablers, mme, SPGW. eNodeB and EPC-Client are deployed as virtual network components. OpenBaton is running in a hardware locally at University of Oulu and OpenStack is used from a cloud service. For each OpenEPC VNFs, five in total (of which 2 are optional as they are emulation of LTE eNodeB and UE), at least two vCPUs and 2GB RAM is required (6 Floating IPs). The choice of flavor is optional yet we used ‘standard. small’ flavor in the descriptor for each VNF package. Heat Orchestration within OpenStack takes care of creating the required topology for OpenEPC network functions. The NFVO and generic VNFM communicates using

[10] http://www.5gtn.fi

RabbitMQ or using REST APIs[3]. Since, passing an image file inside the VNF package is not supported currently in OpenBaton[1] so an OpenEPC cloud image is uploaded on OpenStack prior to on boarding the VNF packages and required descriptors. Moreover, the NFVO is open to any RESTFUL API requests[11] to manage the lifecycle of network functions from outside the network.

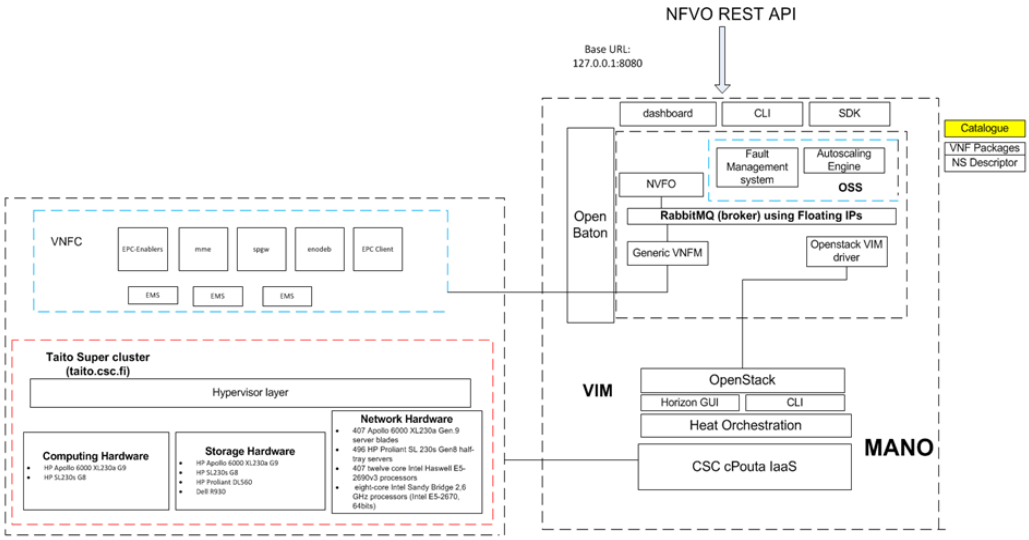

Figure 28 - OpenBaton + OpenEPC EU architecture

[11] http://get.openbaton.org/api/ApiDoc.pdf

3.3.1 Open EPC virtual machines as VNFs

EPC Enablers, mme, spgw, enodeb and epc-client are OpenEPC virtual machines which would be used as VNFs on OpenStack. The OpenEPC network topology was provided using Heat orchestration on OpenStack. For that, a script was provided in Heat template to launch a new stack. Once, the creation of stack is accomplished, the resulting setup is available for use. The deployed network setup and graph from OpenStack is illustrated in Figure 29 and Figure 30, respectively.

Figure 29 - Network Topology (copyright core network dynamics)

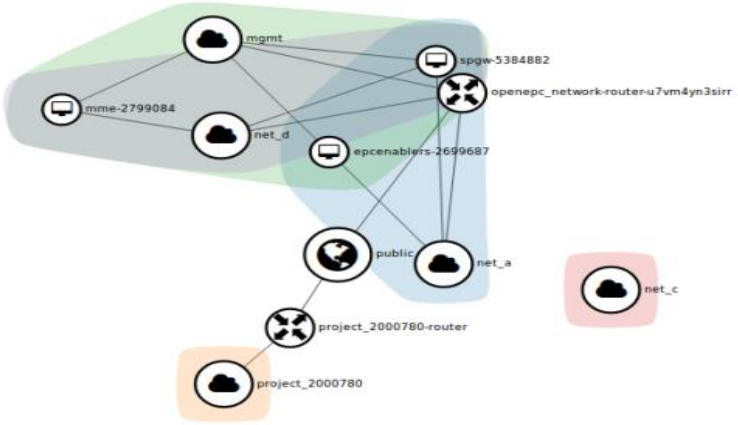

Figure 30 - OpenEPC network graph (copyright core network dynamics)

4 Conclusion

In preparation of the final project stage targeting interoperability between European and Korean software-based mobile core networks, this deliverable documents the implementation work that has been carried out on both the MANO components as well as on the network functions providing the virtualized mobile core network.

The Korean testbed relies on a custom Distributed Cloud NFV Management System developed in-house by the Korean partners, while the European partners rely on the OpenBaton MANO platform. Preliminary evaluations confirm the readiness, stability of the involved platforms in order to support dynamic instantiation, auto-scaling and associated event handling. The interoperability of both platforms is guaranteed through the ETSI SOL3 interface which standardizes the communication between the NFVO and the involved VNFMs. Both sides have integrated necessary changes to the required VNFMs or NFVO in order to support coordinated instantiation as well as auto-scaling of the involved mobile core networks.

Both the Korean and European testbed provide their distinct mobile core network service. The Korean partners provide a Distributed Mobile Core implementation enabling to optimally exploit mobile edge resources in a dynamic manner. The second part of this deliverable documents the architecture, the processes and selected implementation details of the involved network functions, as well as a preliminary evaluation of the resulting implementation. The European partners, re-use the OpenEPC virtualized mobile core implementation of Core Network Dynamics, of which the internal structure (as far as allowed by the license), as well as its instantiation on on OpenStack infrastructure has been briefly documented in Section 3.

Given these implementations, this deliverable prepares the last stage of the project, which is about the interoperability of both MANO platforms in charge of instantiating and auto-scaling the referred mobile core implementations. The actual experimentation, and resulting evaluation of the interoperability will be documented in D4.3, due by the end of the project.

References

[1] 5GCHAMPION, D4.1: Operator grade NFV-based and SDN-enriched EPC environment at 5GTN environment, 2017

[2] ETSI, NFV GS. Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Or-Vnfm Reference Point v2.3.1., NFV-SOL 003, 2017

[3] ETSI, NFV GS. "Network Functions Virtualisation (NFV); Management and Orchestration v1.1.1., NFV-MAN 1, 2014

[4] Fontenla-González, J., Pérez-Garrido, C., Gil-Castiñeira, F., González-Castaño, F. J., & Giraldo-Rodriguez, C. (2016, June). Lightweight container-based OpenEPC deployment and its evaluation. In *NetSoft Conference and Workshops (NetSoft), 2016 IEEE* (pp. 435-440). IEEE.