Small cEllS coordinAtion for Multi-tenancy and Edge services

Grant Agreement No.671596

Topic: H2020-2014-ICT-14
Advanced 5G Network Infrastructure for the Future Internet
Research and Innovation Action

Deliverable D6.1

Orchestrator Architecture Design and Interfaces Specification

Document Number:	H2020-5GPPP-GA No.671596/*WP6/D6.1/30.06.2016*
Contractual Date of Delivery:	*01.07.2016*
Editor:	Shuping Peng – Fujitsu Laboratories of Europe Ltd. (FLE)
Work-package:	*WP6*
Distribution / Type:	Public (PU) / Report (R)
Version:	1.0
Total Number of Pages:	72
File:	*SESAME_Deliverable 6.1_v1.0_Final*

Abstract

SESAME aims to build a cloud-*enabled* platform that supports both radio access and edge computational services at one Point of Presence (PoP). Network Services are supported by Virtual Network Functions hosted in the Light DC, leveraging on technologies like SDN and NFV that allow achieving an adequate level of flexibility and scalability at the cloud infrastructure edge.

The NFV Orchestrator (NFVO), an essential component of CESCM, provides functionalities for managing and orchestrating the resources and network services in small cells. The NFVO manages a typical Network Function Virtualisation Infrastructure (i.e. processing power, storage and networking), and it also composes service chains (constituted by two or more VNFs located either in one or several CESCs) and manages the deployment of VNFs over the Light DC.

This deliverable analyses the state-of-the-art of the existing NFVOs, especially focusing on analysing their features and the relevance to the SESAME NFVO. It then describes the SESAME NFVO architecture in details, including the key functional components within NFVO and the internal and external interfaces.

Service function chaining mechanisms are introduced together with a few SFC examples. The workflows for creating, deploying, modifying, and terminating the SFCs are also described. A functional breakdown of the SESAME NFVO and a modular architecture is given.

The contents of this deliverable will be used as reference for the implementation of NFVO in T6.2 and T6.3.

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	29/02/2016	Table of Content (ToC) - FLE	Shuping Peng - FLE
0.15	01/04/2016	ToC updated (SOTA is moved to the beginning of the Doc; Section 4.1 and 4.2 are renamed; Section 7 is added)	Shuping Peng - FLE Pouria Sayyad Khodashenas - i2CAT
0.2	04/05/2016	Section 1, 2, and 6 – FLE Section 4, 7 – i2CAT	Shuping Peng - FLE Pouria Sayyad Khodashenas - i2CAT Cristina Ruiz - i2CAT
0.25	09/05/2016	Section 5 - EHU	Jose Oscar Fajardo - EHU Begoña Blanco - EHU Fidel Liberal - EHU
0.3	10/05/2016	Section 4.2 – Atos Section 3, 4, 7 – i2CAT Review & Comments – FLE	Javier Garcia Lloreda - Atos Pouria Sayyad Khodashenas - i2CAT Cristina Ruiz - i2CAT Shuping Peng - FLE
0.4	11/05/2016	Sub-sections 4.2.1.3 and 4.2.2.2 - UPC	Jordi Perez-Romero - UPC
	11/05/2016	Section 3 - NCSRD	Ioannis Giannoulakis - NCSRD
	11/05/2016	Integration & Review – FLE	Shuping Peng - FLE
0.42	12/05/2016	sub-section 7.1 – I2CAT	Pouria Sayyad Khodashenas - i2CAT
0.43	18/05/2016	Updates on section 5 - EHU	Jose Oscar Fajardo - EHU
	27/05/2016	Review Section 1, 2, 3, 4	Irena Trajkovska - ZHAW
0.44	01/06/2016	Updates on section 2 and 4.1.3 – CNET	Leonardo Goratti, Shah Navaz Khan, Cristina Costa - CNET
0.45	01/06/2016	Integration of comments from different versions	Shuping Peng - FLE
0.46	02/06/2016	Updates on sections 2, 3, 4 and 7 – i2CAT	Pouria Sayyad Khodashenas - i2CAT
0.47	07/06/2016	General formatting, typos, spelling.	Cristina Ruiz - i2CAT
0.48	08/06/2016	Additions in Section 4.1.3	Leonardo Goratti - CNET
	10/06/2016	Review Section 5, 6, 7	Irena Trajkovska - ZHAW
0.49	19/06/2016	Additions to Section 2.9	Leonardo Goratti - CNET
0.5	20/06/2016	General review Section 3, 4, 7 and 8.	Cristina Ruiz - i2CAT Pouria Sayyad Khodashenas - i2CAT
0.6	20/06/2016	Address comments in Section 5	Jose Oscar Fajardo - EHU
0.7	27/06/2016	Address comments in Section 4 Second round review	Jordi Pérez-Romero - UPC Javier Garcia Lloreda – Atos
0.8	27/06/2016	Review, Second round review	Shah Nawaz Khan – CNET
0.9	27/06/2016	Review and edition	Shuping Peng – FLE
1.0	29/06/2016	Final editorial and conceptual review, performed by the project coordinator	Ioannis Chochliouros – OTE

Contributors

First Name	Last Name	Partner	Email
Shuping	Peng	FLE	shuping.peng@uk.fujitsu.com
Charles	Turyagyenda	FLE	Charles.Turyagyenda@uk.fujitsu.com
Mick	Wilson	FLE	mick.wilson@uk.fujitsu.com
Pouria	Sayyad Khodashenas	i2CAT	pouria.khodashenas@i2cat.net
Cristina	Ruiz	i2CAT	cristina.ruiz@i2cat.net
Jordi	Ferrer Riera	i2CAT	jordi.ferrer@i2cat.net
Eduard	Escalona	i2CAT	eduard.escalona@i2cat.net
August	Betzler	i2CAT	august.betzler@i2cat.net
Jose Oscar	Fajardo	EHU	joseoscar.fajardo@ehu.eus
Begoña	Blanco	EHU	begona.blanco@ehu.eus
Fidel	Liberal	EHU	fidel.liberal@ehu.eus
Javier	Garcia Lloreda	Atos	javier.garcial@atos.net
Jordi	Perez-Romero	UPC	jorperez@tsc.upc.edu
Oriol	Sallent	UPC	sallent@tsc.upc.edu
Shah Nawaz	Khan	CNET	s.khan@create-org.net
Cristina	Costa	CNET	cristina.costa@create-net.org
Leonardo	Goratti	CNET	leonardo.goratti@create-net.org
Ioannis	Chochliouros	OTE	ichochliouros@oteresearch.gr

Glossary

Acronym	Explanation
4G	Fourth Generation of Mobile Communications
5G	Fifth Generation of Mobile Communications
AMQP	Advanced Message Queuing Protocol
AP	Application Protocol
API	Application Programming Interface
BSS	Business Support System
CA	Controller Adapter
CC	Cloud Controller
CESC	Cloud-enabled Small Cell
CESCM	Cloud Enabled Small Cell Manager
CN	Core Network
CP	Connection Point
CPE	Customer Premises Equipment
CPU	Central Processing Unit
CRUD	Create/Read/Update/Delete
DB	Database
DC	Data Centre
DevOp	Development and Operation
DL	Downlink
DRDB	Domain Resource Database
DSP	Digital Signal Processor
E2E	End-to-End
EMS	Element Management System
EPA	Enhanced Platform Awareness
EPC	Evolved Packet Core
ETSI	European Telecommunications Standards Institute
EU	European Union
FE	Functional Entity
FG	Forwarding Graph
FM	Fault Management
FP	Framework Programme
FP	Function Provider
FP7	7th Framework Programme
FPGA	Field Programmable Gate Array
GA	Grant Agreement
GPRS	General Packet Radio Service
GS	Group Specification
GTP	GPRS Tunnelling Protocol
GUI	Graphical User Interface
GW	Gateway
H2020	Horizon 2020
HeNB	Home eNodeB
HTTP	Hyper-text Transmission Protocol
HW	Hardware
HWA	Hardware Accelerator
I/O	Input/Output
IaaS	Infrastructure-as-a-Service
IB	Information Base
ICT	Information and Communication Technology
ID, id	Identifier

IETF	Internet Engineering Task Force
ILP	Integer Linear Programming
IP	Internet Protocol
IPsec	Internet Protocol Security
ISG	Industry Specification Group
IT	Information Technology
IVM	Infrastructure Virtualisation and Management
JMS	Java Message Service
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
Light DC	Light Data Centre
LANMAN	Local and Metropolitan Area Networks
M2M	Machine-to-Machine
MANO	Management and Orchestration
MCN	Mobile Cloud Networking
MEC	Mobile Edge Computing
MILP	Mixed Integer Linear Programming
MIQCP	Mixed Integer Quadratically Constrained Programming
MME	Mobility Management Entity
MOCN	Multi-Operator Core Network
NAT	Network Address Translator
NCT	Network Connectivity Topology
NE	Network Edge
NF	Network Function
NFP	Network Forwarding Path
NFV	Network Functions Virtualisation
NFVI	Network Functions Virtualisation Infrastructure
NFVO	NFV Orchestrator
NMS	Network Management System
NS	Network Service
NSD	NS Descriptor
NUMA	Non-Uniform Memory Access
OCCI	Open Cloud Computing Interface
OPNFV	Open Platform for NFV
OSM	Open Source Mano
OSS	Operations Support System
OVS	Open virtual Switch
PaaS	Platform-as-a-Service
PHP	Hypertext Preprocessor
PLMN	Public Land Mobile Network
PM	Performance Management
PNF	Physical Network Function
PNFD	PNF Descriptor
PoC	Proof of Concept
PoP	Point of Presence
PPP	Public-Private Partnership
RAM	Random Access Memory
RAN	Radio Access Network
REA	Research Executive Agency
REST	Representational State Transfer
RFC	Request for Comments
RIA	Research and Innovation Action
RO	Resource Orchestrator
RPC	Remote Procedure Call

SC	Small Cell
SCaaS	Small Cells-as-a-Service
SCNO	Small Cell Network Operator
SDC	Smart Data Centre
SDK	Software Development Kit
SDN	Software-defined Networking
secGW	Security Gateway
SFC	Service Function Chaining
SIC	Standard Industrial Classification
SLA	Service Level Agreement
SLM	Services Lifecycle Manager
SM	Service Manager
SME	Small- and Medium-sized Enterprise
SO	Service Orchestrator
SOTA	State-of-the-Art
SQL	Structured Query Language
SR-IOV	Single Root I/O Virtualization
SW	Software
SWA	Software Architecture
TC	Technical Committee
ToC	Table of Contents
ToS	Type of Service
TOSCA	Topology and Orchestration Specification for Cloud Applications
TU	Transcoding Unit
UDP	User Datagram Protocol
UE	User Equipment
UL	Uplink
vCE	virtual Cloud Equipment
vCPE	virtual Customer Premises Equipment
vDPI	virtual Deep Packet Inspection
VDU	Virtual Deployment Unit
VIM	Virtual Infrastructure Manager
vHeNB	virtual Home eNodeB
VL	Virtual Link
VLD	Virtual Link Descriptor
VM	Virtual Machine
VNE	Virtual Network Embedding
VNF	Virtual Network Function
VNFD	VNF Descriptor
VNFFG	VNF Forwarding Graph
VNFFGD	VNF Forwarding Graph Descriptor
VNFM	VNF Manager
VNFR	VNF Record
VSCNO	Virtual Small Cell Network Operator
vTU	virtual Transcoding Unit
WAN	Wide Area Network
WICM	WAN Infrastructure Connection Manager
WP	Work Package
XaaS	All-as-a-Service

Table of Contents

List of Figures

List of Tables

1 Introduction

The SESAME concept is focused on innovations in the placement of network intelligence and applications at the edge through Network Functions Virtualisation (NFV) and Mobile Edge Computing (MEC), the substantial evolution of the Small Cell (SC) towards cloud-*based* coordinated management, and the consolidation of multi-tenancy in mobile communications infrastructures [1]. The resulting solution will allow several operators/service providers engaging in new sharing models of both access capacity and edge computing capabilities, i.e. promoting the concept of "Small Cells-as-a-Service" (SCaaS) based on the conceptual model of "network slicing" the logical partitioning of the localized network infrastructure in one Point of Presence (PoP).

SESAME aims to build a cloud-*enabled* platform that supports both radio access and edge computational services at one PoP. For that purpose, it proposes the Cloud-*Enabled* Small Cell (CESC) concept, composed by a physical small cell unit and a micro-server. The aggregation of the micro-servers belonging to a CESC cluster provides a virtualised execution infrastructure, denoted as "Light Data Centre" ("Light DC"), which enhances the virtualization capabilities and the power process for executing novel applications and services at the network edge (NE). Network Services (NSs) are supported by Virtual Network Functions (VNFs) hosted in the Light DC, leveraging on technologies like Software-*defined* Networking (SDN) and NFV that enable an adequate level of flexibility and scalability at the edge cloud infrastructure. It foresees the functional split of the small cell between Small Cell Physical Network Functions (SC PNFs) and Small Cell Virtual Network Functions (SC VNFs) and provides a multi-tenancy environment to support the Multi-Operator Core Network (MOCN) requirements. The hosting of VNFs related to small cell virtualised functions as well as service VNFs (e.g. virtual Deep Packet Inspection - vDPI) is guaranteed by the design of the advanced modular micro-server whose architecture and characteristics are optimized for the MEC environment.

The NFV Orchestrator (NFVO), an essential component at the heart of the CESCM Manager (CESCM), provides functionalities for managing and orchestrating the resources and network services in the CESCs. The NFVO manages a typical Network Function Virtualisation Infrastructure (NFVI) (i.e. processing power, storage and networking); it composes service chains (constituted by two -or more- VNFs located either in one -or several- CESCs) and manages the deployment of VNFs over the Light DC.

In a heterogeneous radio/IT environment, *like the one proposed in SESAME*, some important management tasks such as the way physical and virtual resources are "mapped" together and the NSs scaling, need to be handled in a more precise and comprehensive manner. These tasks, *in general*, are carried out automatically by the CESCM and, *in particular*, by the NFVO, which implies the utilization of global and local optimization methods.

Generally speaking, the problem of "placing" virtual resources (i.e. VNFs and related Virtual Machines (VM)) over the physical infrastructure can be solved by considering different objectives such as the minimization of the overall energy consumption, or the minimization of the virtual resource consumption. Depending on the targeted objective, the problem is formulated accordingly. The task of VNF placement becomes even more challenging in a setting such as that of SESAME, where the underlying CESCs are meant to be "shared" by different

"tenants", i.e. Virtual Small Cell Network Operators (VSCNOs). In this respect, the performance isolation problem between tenants must also be addressed.

Conceptually, similar to the VNF placement problem, the NSs scaling also needs a precise decision making procedure. That may include resilient and cognitive solutions, depending on the targeted use cases. In this case, one or more VNFs involved in a service chain can dynamically be scaled up and down, according to the actual traffic conditions. This can easily be translated to energy and resource efficiency. This feature is one of the key advantages of NFV as opposed to the vertically integrated middle boxes where scaling a network function up and down essentially implies interfering with the real Hardware (HW). A possible solution for the NSs scaling problem is to have some key elements on the scope such as the process triggering threshold (when and how to start the NSs scaling procedure), the scaling process structure, compliancy of scaling steps with the agreed Service Level Agreement (SLA), etc.

1.1 Deliverable outline

The present deliverable covers the NFVO architecture design and interfaces specification, which includes the following sections:

- *Section 1* offers a brief introductory overview.
- *Section 2* provides a state-of-the-art (SOTA) analysis of the current available open source NFVOs.
- *Section 3* clarifies the role of NFVO in the CESCM and introduces its key functionalities.
- *Section 4* describes the NFVO architecture in details, also including the key functional components within NFVO and the internal and external interfaces.
- *Section 5* defines the unique characteristics of the service function chaining (SFC) in SESAME, while few SFC examples are given to show the uniqueness of service provisioning in SESAME.
- *Section 6* describes the workflows of SFC creation and deployment, modification, and termination.
- *Section 7* provides a NFVO implementation guideline.
- Finally, *Section 8* summarises the key issues discussed in the document and concludes the deliverable.

2 NFVO in the State-of-the-art

In this section, we analyse the state-of-the-art of the available NFVO solutions with a concern to adapt them in the specific context of SESAME. In total, nine projects (NFVO solutions and their features) are analysed. Moreover, their relevance to SESAME in terms of similarity and differences is also analysed.

2.1 Hurtle

Hurtle is a solution developed by the ICCLab's Cloud Orchestration Initiative ([2],[3]), and supported by Mobile Cloud Networking (MCN). Hurtle allows automation of the lifecycle management of network services and it is licensed under the Apache License Version 2.0[1]. The general concept proposed by Hurtle is represented in Figure 1. The key components of Hurtle[2] include the *Service Manager* (SM), *Service Orchestrator* (SO), *Cloud Controller* (CC) and the *Cloud Controller SDK*. The SM is the entity receiving and executing requests for new tenant services. The SO, instead, manages the lifecycle of a tenant service instance. On the other hand, the CC manages and abstracts resources for SOs. Finally, the CC SDK provides an interface between the SO and internal services of the CC.

Hurtle uses standard Open Cloud Computing (OCCI) interfaces. OCCI forms the core interfaces exposed by the SMs, as well as the northbound interface of the CC. Most of the implementation of Hurtle is done in Python language. The SO in Hurtle relies on the languages supported by OpenShift[3], including java and PHP [4].

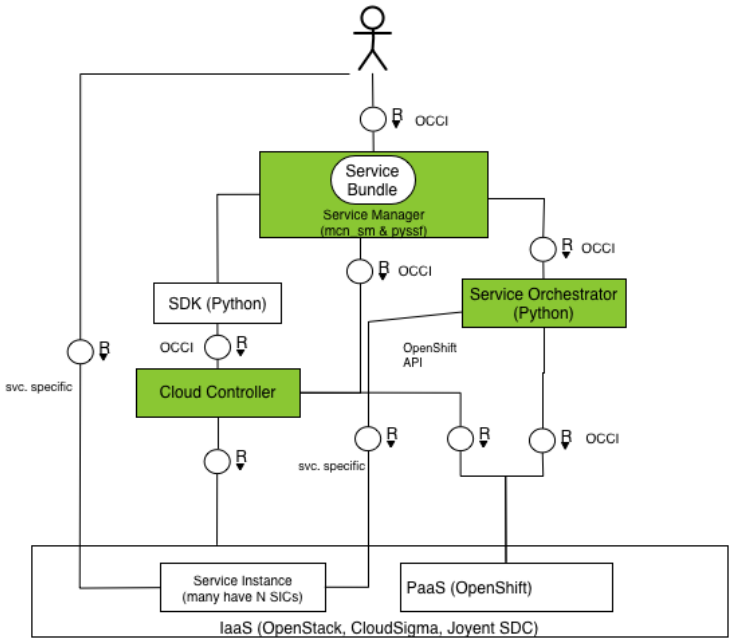

Figure 1: Hurtle technical architecture [5]

[1] For more information, see: http://www.apache.org/licenses/

[2] An Introductory video explaining how to use Hurtle is available at: https://www.youtube.com/watch?v=03YiBT3IM9s

[3] More related information can be found at: https://www.openshift.com/web-hosting/index.html?sc_cid=701600000011p9xAAA&gclid=CjwKEAjwkui7BRCf64DNtfDupgoSJAA_0LOoUzkRltvkt5KDJyPV72NaLiTxDer7AH4wU3nXZ1olTBoC37bw_wcB

2.1.1 Project description and features

The architecture components of Hurtle are depicted in Figure 2. The pivot is that Hurtle provides service to end users, which can be a single atomic service or a chain of individual atomic services. The services are created leveraging on the available resources, which can be both physical (e.g. physical storage) and virtual (e.g. virtual machines). The workflow developed by Hurtle to "instantiate" a service consists in: the end user making a request to the SM; the SM subsequently creates the corresponding SO, which is responsible for the creation of the service instance following the Hurtle service lifecycle; the SM and SO rely on the CC as an abstraction layer to access internal services and resources in order to instantiate the service requested by the end user.

The key features of Hurtle can be summarised as follows:

- Orchestration and life-cycle management of services.
- Recursive composition of services.
- Hurtle uses Open Cloud Computing (OCCI) interfaces
    - Support for TOSCA[4] (in the roadmap)
    - Support OpenStack, CloudStack and Smart Data Centre (SDC)
    - Service Placement functionality (optimise on cost & latency)
    - Integrated Continuous Integration/Continuous Delivery - upgrade your service and or resources transparently
    - State migration of applications (under development)
    - Runtime monitoring and adaptation engine
    - Integration with Open Baton (under development)
- Hurtle workflow:
    - **Design**: Topology and dependencies of each service component are specified. The model here typically takes the form of a graph.
    - **Implementation**: This is where the developer(s) needs to implement the actual software (SW) that will be provided as a service through Hurtle.
    - **Deploy**: The complete fleet of resources and services are deployed according to a plan executed by Hurtle. At this stage they are not configured.
    - **Provision**: Each resource and service is correctly provisioned and configured by Hurtle. Provisioning of operational dependencies is ensured.
    - **Runtime**: Manage all components orchestrated by Hurtle. To manage means -at the most basic level- to monitor the components. Based on metrics extracted, performance indicators can be formulated by using logic-*based* rules. When notified where an indicator's threshold is breached, an Orchestrator could take a remedial action to ensure reliability.
    - **Disposal**: This is where a Hurtle service instance is destroyed.

[4] https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca

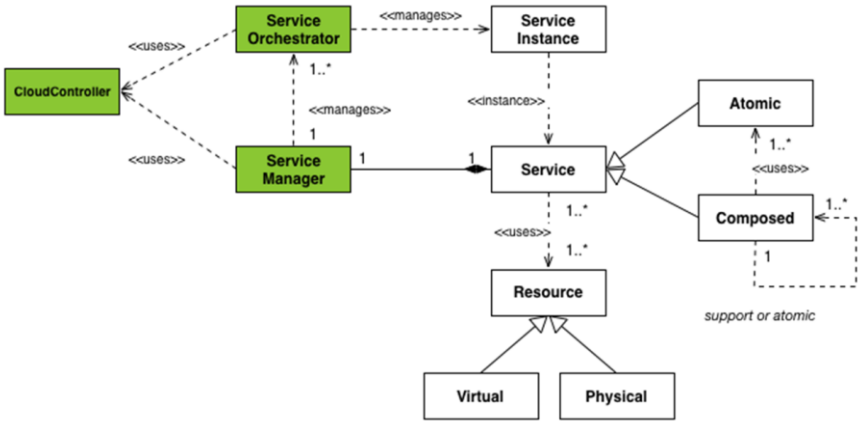

Figure 2: Hurtle conceptual architecture [6]

2.1.2 Relevance to SESAME (Similarity / Differences)

Hurtle was developed within the FP7 MCN project [7] and used to orchestrate and compose virtual network functions. The project aimed to combine the typical cloud approach with mobile networks. In particular, the project studied the components of a mobile network such as the radio access network (RAN), the core network (CN), the orchestration and business support systems in order to revisit them in view of cloud concepts. In addition, the MCN project explored the All-as-a-Service (XaaS) concept, in which the creation of a service is fully automated to facilitate the creation of end-to-end (E2E) services.

Since SESAME develops the concept of SCaaS, with particular focus on the radio access network, Hurtle can provide a service oriented orchestrator solution that is able to monitor the components of a service and take action in case of violation of rules, such as the violation of SLAs put in place between virtual small cell network operator (VSCNO) and the SCNO.

2.2 SONATA

The SONATA ("Service Programming and Orchestration for Virtualized Software Networks") project [8] is an EU-funded project (Horizon 2020) and part of the 5G-PPP initiative. Recently started in July 2015, the project has scheduled a thirty-month duration. SONATA is a highly collaborative effort, with 15 partners representing telecom operators, manufacturers, system integrators, service providers, SME developers, research and academic institutes.

2.2.1 Project description and features

SONATA is developing a Service Platform that includes an NFVO as a chief component. The NFVO will be open source and modular in design. The latter refers to an extensible MANO framework of plug-ins that covers both NFVO and VNFM functionality. The modular strategy allows bespoke customization for the adopting platform operators, facilitating them to integrate new functionalities as needed. In SONATA a gatekeeper module is the main entry point for service packages and service requests.

As the project is focused on facilitating service development, the service platform relies on the gatekeeper module to connect to SONATA's various SDK tools, platform-specific catalogue and repositories, together with an accompanying NFV-focused DevOps (i.e. Development and

Operations) approach. Essentially DevOp emphasises the cooperation between software developers and operators. Furthermore, in the SONATA system a message broker connects the different components of the MANO framework developed within SONATA. The SONATA architecture includes a layer in which the global orchestrator and the network services resource description are present. Logically speaking, below this top layer, SONATA presents a recursively deployed platform. The project aims to develop an automated dynamic service creation platform, along with a service creation language. The global orchestrator in SONATA possesses the necessary algorithms which enable optimal placement of VNFs across the infrastructure.

SONATA's activities are in parallel to SESAME, and its first open source results are expected to be available in the summer of 2016.

2.2.2 Relevance to SESAME (Similarity / Differences)

Since SONATA participates to the first phase of the 5G projects and it delves onto a system for agile development and operation of network services, SESAME could monitor the activities carried forward within this project and in particular for the orchestration part. This is particularly relevant considering that the NFVO of SONATA is going to be modular by design, based on plugins to extend functionality, and access to catalogues and repositories will be made through the gatekeeper module.

2.3 T-NOVA

The FP7 T-NOVA project [9] has a period of thirty-six months duration and will be completed by December 2016. An overview of the project activities, in particular with respect to the orchestration part is provided herein below.

2.3.1 Project description and features

The main aim of the project is to facilitate the deployment of VNFs for the network operators and to enable them to offer it as a service to other interested customers. The term "Virtual Network Appliances" has been used for network functions (e.g. gateways, proxies, firewalls, etc.) that can be provided on demand to the customers "as a service" thereby eliminating the need for specialized hardware devices.

To be able to provide this "Network Function as a Service" functionality, the T-NOVA project aims to develop an orchestrator platform that can work on top of a virtual infrastructure leveraging the cloud management architectures and Software Defined Networking benefits. On top of this management and orchestration layer, the T-NOVA project aims to provide a layer of market-ready products through a "NFV Marketplace". Taking inspirations from the product marketplaces available for different end-user devices (e.g. Apple App Store and Google Play Store), the NFV marketplace is envisioned to serve as a one-stop shop for network functions and services that are compatible with the T-NOVA management and orchestration layer. The network operators can compare and select different products from the marketplace that better serve their needs and are guaranteed to provide a certain performance level as well as service level functions.

To be able to support generic network and service functions, the T-NOVA orchestrator will play an important role in the architecture. The T-NOVA Orchestrator reference architecture, as well as the interfaces with the external Functional Entities (FEs) is depicted in Figure 3. The

orchestrator interacts with the Marketplace, which is the T-NOVA domain responsible for accounting, SLA management and business functionalities. Besides the Marketplace, the Orchestrator also interfaces with the Infrastructure Virtualisation and Management (IVM), and in particular with the VIM, for managing the data centre network/IT infrastructure resources, as well as with the WICM for WAN connectivity management. Finally, the Orchestrator interacts with the VNF itself, which in the T-NOVA scope is located in the IVM domain, to ensure its lifecycle management.

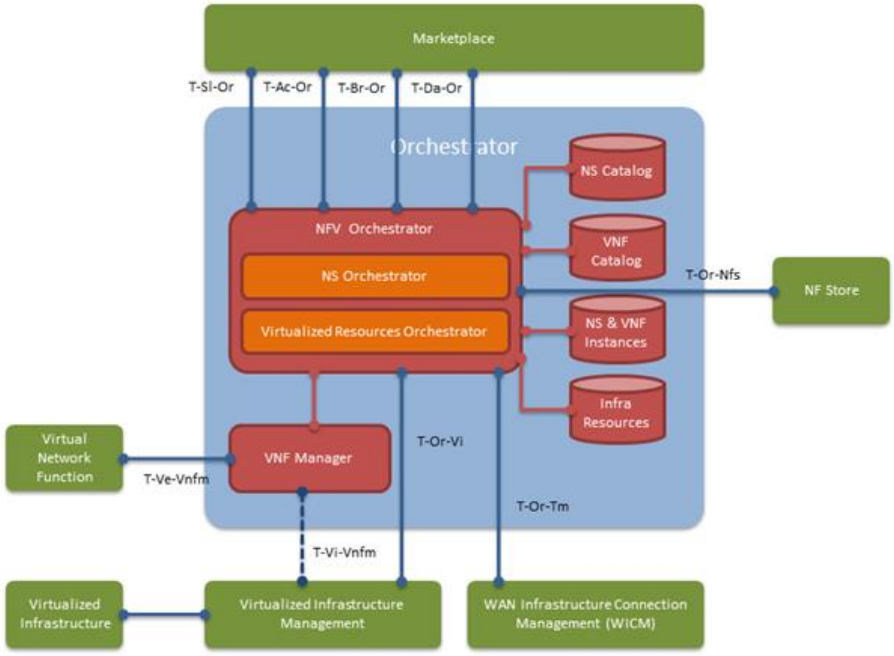

Figure 3: T-NOVA Orchestrator Reference Architecture

Internally, the T-NOVA Orchestrator consists of two main components and a set of repositories that provide essential information for its operation. One of the core elements is the NFVO, acting as the front-end with the Marketplace and orchestrating all the incoming requests towards the other components of the architecture. To support the NFVO operation procedures, a set of repositories is identified in order to store the description of the available VNFs and NSs (VNF Catalogue and NS Catalogue), the instantiated VNFs and NSs (NS & VNF Instances), as well as the available resources in the virtualised infrastructure (Infrastructure Resources Catalogue). Finally, the NFVO also interacts with the other core element, the VNF Manager (VNFM), responsible for the VNF-specific lifecycle management procedures.

2.3.2 Relevance to SESAME (Similarities / Differences)

Orchestration is a fundamental and one of the most challenging aspects of SDN/NFV architectures. The T-NOVA orchestrator performs most of the functions generally associated with VNF orchestrator (e.g. infrastructure, VNF management), but also serves as the main interface with the marketplace entities thereby facilitating easy integration of new functions and services in the network. Therefore, it has several similarities to the vision of orchestration in SESAME as well as other architectures based on cloud and virtualisation foundations.

- The main orchestration functions considered in T-NOVA are not only similar to SESAME, but also to the general functional scope of orchestration in SDN/NFV architectures. The T-NOVA orchestrator may easily compose VNF chaining and it is built to interact with the T-NOVA marketplace. While the concept of marketplace is unique to T-NOVA, a function or service instantiation in the network architecture, keeping a view of its performance and resource utilization etc. are very similar objectives, at least conceptually, to the fundamental tasks of orchestration that will be completed in the SESAME project.
- The focus on bringing virtualisation benefits to the network edge in SESAME project does present some distinct challenges that may not be addressed in the scope of orchestration in T-NOVA architecture. SESAME requires an orchestrator that will be responsible for not only managing VNFs but also handle composite VNF / PNF chains. Additionally, the Light DC concept with potentially large number of CESCs as well as the multi-tenancy support will require a more granular level of orchestration and control over service and SC VNFs in the SESAME architecture.

2.4 UNIFY

Funded by the European Commission, with a period of thirty-month duration, the FP7 UNIFY project [10] aims to improve virtualised network operations by building a new architecture that optimizes data traffic flows.

2.4.1 Project description and features

The UNIFY project aims to address a fundamental problem of the legacy carrier networks i.e., the rigidity of the network control which not only makes administration and control difficult but also adversely affects the elasticity of the provided services and the introduction of new ones. To address this problem, the UNIFY consortium has focused on achieving a full network and service virtualisation with the main aims of enabling rich and flexible services and realizing operational efficiency. These objectives have led to the identification of some main focus points within the project, including orchestration, verification and observation of end-to-end service delivery spanning from home and enterprise networks to the core networks and data-centres. From an abstract view, the UNIFY project aims to "merge" the data-centres and network infrastructure of the carriers to create a unified, cloud-*enabled* platform for flexible service provision and achieving operational efficiency with the added benefits of cloud-*driven* architecture such as extensibility, maintenance and control. It proposes a service abstraction model together with a service creation language to automate the placement of networking, storage and computing elements in the provided infrastructure layer.

Orchestration is one of the main focus point in UNIFY and Figure 4 presents a simplified view of the high level components of the developed orchestration layer. The orchestration layer sits on top of the infrastructure layer and provides an abstraction to the service layer. Internally, the UNIFY orchestrator has two main components, i.e. Controller Adapter (CA) and Resource Orchestrator (RO). The CA is responsible for creating a high level virtual view of the underlying technologies (e.g. by hiding the details that are not relevant to higher level orchestration), controller platforms and other infrastructure components. It communicates with underlying network and compute controllers to collect virtualised and physical resource information into storage structures referred to as Domain Resource Database (DRDB). From the DRDB, different

virtualised components create a domain specific virtual view that is used by the orchestrator (RO) to do a technology agnostic operation. The main purpose of this is to realize a split that achieves a separation of concern (between CA and RO) for orchestration on top of abstract resources and mapping the execution into different technologies, domains and protocols.

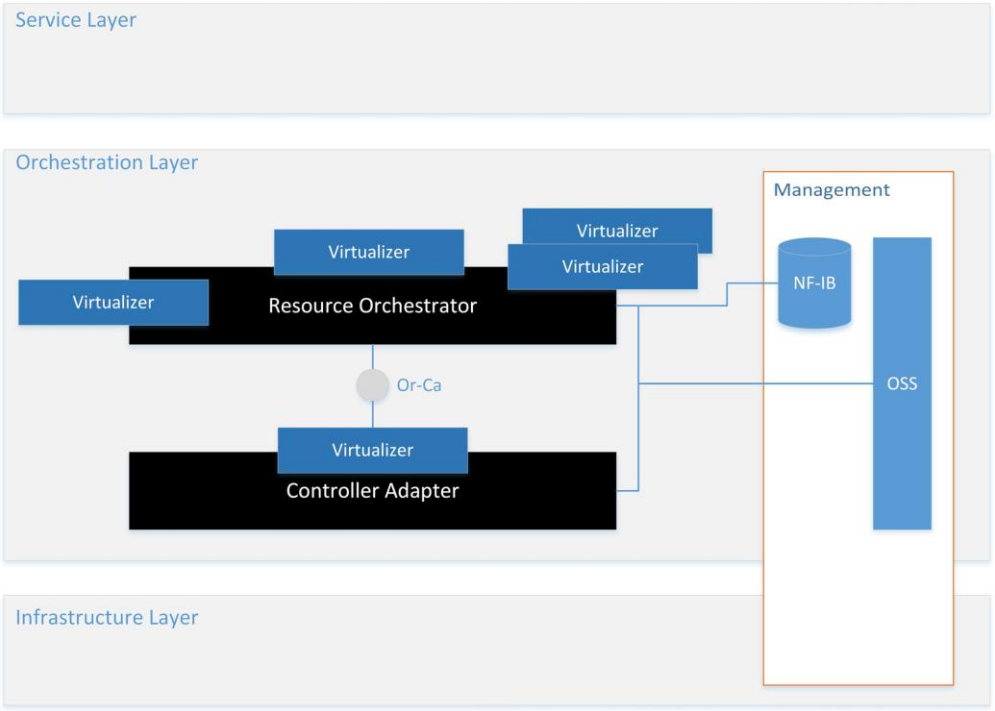

Figure 4: UNIFY orchestration layer components

The OSS in the management box is different from an OSS/BSS that may operate in the service layer. In the orchestration layer, the OSS is managed by the OSS/BSS in the service layer and takes care of management and operation of the RO and CA. The Network Function - Information Base (NF-IB) provides RO the necessary information about the network functions (e.g. the resources required for deploying NF) to map a particular service on top of a virtual network view.

2.4.2 Relevance to SESAME (Similarities / Differences)

Resulting from the main vision of UNIFY, the orchestration layer aims to achieve service elasticity on top of a unified virtual layer, created on top of diverse set of technologies and resources. It therefore has certain distinct features that may not be considered in SESAME (e.g. focus on diverse network technologies) but also "shares" the main orchestration tasks requiring a description of the network functions and a virtualisation layer to facilitate its placement. From an orchestration point of view, the UNIFY orchestration layer has to deploy several network functions and "chain" them together on top of an abstract representation of the underlying resources. The NF-IB provides the necessary description of the network functions for this purpose. These issues are also considered in the scope of SESAME and can, therefore, benefit from the output of the UNIFY project.

The orchestration tasks in SESAME involve additional -as well as different- challenges compared to UNIFY's orchestration layer. While the UNIFY architecture aims to abstract the physical details from the orchestration, the SESAME orchestrator has to take it into consideration for different

PNFs (e.g. through the Element Management System – EMS) to be able to realize different use-cases. The tasks associated with Controller Adapter in UNIFY orchestration layer may not be needed (at least explicitly) in the SESAME scope.

2.5 TACKER

Tacker [11] is an official OpenStack project building a Generic VNF Manager (VNFM) and a NFV Orchestrator (NFVO) to deploy and operate Network Services and Virtual Network Functions (VNFs) on an NFV infrastructure platform like OpenStack. It is based on ETSI MANO Architectural Framework and provides a functional stack to orchestrate network services end-to-end by using VNFs. OpenStack components[5] such as Nova, Neutron and Cinder collectively act as the VIM.

The choice of implementing general purpose components is motivated by the fact that operators are deploying VNFs from Multiple VNF vendors and leverages on the fact that the majority of orchestration workflows are VNF-agnostic and the VNFM functions are generic and applicable to most types of VNFs. At the same time, VNF-specific differentiation can be handled by using plugins and drivers. This approach allows to avoid vendor lock-in, it is multi-tenant aware, and encourages NSD and VNFD template standardization (e.g. TOSCA NFV Profile).

Tacker generally takes TOSCA templates as input for VNF manager to orchestrate the VNFs. TOSCA templates constitute both VNF descriptors and network service descriptors. Currently, Tacker supports only VNF descriptors. Tacker brings NFV into OpenStack in order to develop a general-purpose NFVO.

Tacker has many supporters and developers in the OPNFV community[6] and is projected to be an option in a future OPNFV release. Tacker is released as free software and is under the Apache license.

2.5.1 Project description and features

The OpenStack Tacker project addresses the NFVO and VNFM. Tacker Open NFV Orchestrator has an integrated general-purpose VNF Manager to deploy and operate VNFs, with Service Function Chaining. It is based on the ETSI MANO Architectural Framework and provides a fully functional stack to orchestrate VNFs end-to-end. Tacker provides a VNF Catalogue to on-board VNF Descriptors (written using OASIS TOSCA NFV standards) and provides APIs for Life-Cycle Management of VNFs along with capabilities like VNF monitoring, auto-scaling and self-healing. Tacker plans to expand to NFVO capabilities like Network Service Descriptors (NSD) and VNF Forwarding Graph Descriptor (VNFFGD) support in upcoming cycles. Tacker utilizes OpenStack Compute (Nova), Neutron and Heat to execute the VNF life-cycle:

- Tacker API deploys VNF from the VNF Catalogue.
- Instantiates one or more VMs described in TOSCA NFV template.
- Tacker facilitates configuration injection into VNFs and provides a loadable framework for KPI monitoring and healing.
- To terminate VNF will delete all VMs and other resources associated with VNF instance.

TACKER presents the following characteristics:

[5] https://www.openstack.org/software/
[6] https://www.opnfv.org/

- Supports a template based end-to-end Network Service deployment using decomposed VNFs.
- In TACKER, VNF Descriptor includes both non-parameterized VNFD template and parameterized VNFD template. A non-parameterized VNFD has static values for the parameters, while parameterization allows for the ability to use a single VNFD to be deployed multiple times with different values.
- TACKER MANO API introduces REST API end-points based on ETSI NFV MANO standards[7]. The two resources introduced are 'vnfd' and 'vnf' for describing the 'vnfm' extension. Tacker API can be used by SP's OSS / BSS or a NFV Orchestrator to deploy VNFs in the use case scenarios of e.g. vCE, vCPE, vPE.
- Supports VNF placement policies to ensure efficient placement of VNFs.
- Allows VNFs to be connected using a SFC, described in a VNF Forwarding Graph Descriptor.
- Provides VIM Resource Checks and Resource Allocation functionalities.
- Orchestrates, if needed, VNFs across Multiple VIMs.
- Spans Physical NFs and Virtual NFs.
- Renders VNF Forwarding Graphs using SDN Controller or a SFC API Supports VNF user-data injection.
- Supports VNF configuration injection – during Instantiation and Update.
- VNFD Catalogue.
- Management of the basic life-cycle of VNF (define/start/stop/undefine).
- Performance and Health monitoring of deployed VNFs.
- Auto Healing VNFs based on Policy.
- Initial configuration of VNF facilitated.

There are four major components of Tacker: *VNFD Catalogue*, *VNF Provisioning*, *VNF Configuration Management*, and *VNF Monitoring and Auto Healing*. Here is a quick summary of progress to date in each of these areas.

VNFD Catalogue: Tacker uses TOSCA for VNF meta-data definition. TOSCA (Topology and Orchestration Specification for Cloud Applications) is a TC or technical committee under the OASIS consortium[8] that drives the development, convergence and adoption of open standards for the global information society. Within TOSCA, Tacker uses the NFV profile schema. VNFs should be described using the TOSCA NFV template in order to be on-boarded into the Tacker VNF Catalogue (TOSCA Simple Profile for NFV specification). Once on-boarded, Tacker can instantiate the VNF by interpreting the TOSCA template and translating appropriate portions to

[7] See, for example: https://specs.openstack.org/openstack/tacker-specs/specs/liberty/tacker-api-mano.html

[8] OASIS is a nonprofit consortium that drives the development, convergence and adoption of open standards for the global information society. More information can be found at: https://www.oasis-open.org/

OpenStack Heat[9] by using a translator. Tacker also takes care of configuring the VNF and continues to monitoring and, if needed, auto healing to complete the full lifecycle prescribed by ETSI MANO.

Tacker VNFD Catalogue supports for multiple VMs per VNF (VDU). It also provides APIs to on-board and maintains the VNF Catalogue. The VNFDs are stored in Tacker DB.

VNF Provisioning: By using the Heat template described above, Tacker uses OpenStack Nova[10] to provide the compute infrastructure. Lot of the features from OpenStack Nova can be leveraged during the compute provisioning process. By creating the right flavour with specific attributes such as SR-IOV Passthrough, NUMA, CPU pinning, large page allocation etc., compute resources are optimized for the VNF[11].

VNF Configuration Management: Tacker will push specific configurations required by the VNFs through the configuration driver. Configuration management is built as a pluggable framework where different VNF vendors can write their own configuration drivers for their VNFs.

Another approach is to use the SDN controller: Tacker, by using the SDN controller plugin, can push configurations for specific VNFs using the SDN controller's southbound interfaces. This is a great example of how SDN and NFV come together.

VNF Monitoring and Auto Healing: One of the key responsibilities of Tacker is to monitor the health of the VNFs. Following the same principles that guide the design of other projects in OpenStack, Tacker will have some ready-to-use loadable monitoring drivers like icmp-ping, http-ping, etc. There is also a planned integration with Ceilometer[12], and VNF vendors can write their own monitoring driver with specific monitoring attributes.

VNF Auto-Scaling is planned to implement an Auto-Scale VNF based on policy, implementing Basic Auto-Scaling using common VM metric (such as CPU threshold) or custom monitoring metric.

2.5.2 Relevance to SESAME (Similarity / Differences)

TACKER provides a complete and full functional stack to orchestrate VNFs and service chains in OpenStack based virtualization environment. As it follows the ETSI MANO architectural framework, it can provide the necessary functional components that can be applied in SESAME project as well. However, the level of maturity of the software together with a very limited catalogue of VIM support may become a hindrance considering the different platforms used in SESAME and the existing lack of support for this platform. The SESAME light DC is also a resource constrained environment where the legacy assumptions of data centre level resources under the VIM abstraction layer do not hold true.

---

[9] Heat is the main project in the OpenStack Orchestration program. It implements an orchestration engine to launch multiple composite cloud applications based on templates in the form of text files that can be treated like code. For more relevant information, see: https://wiki.openstack.org/wiki/Heat

[10] OpenStack Nova is a component within the OpenStack open source cloud computing platform developed to provide on-demand access to compute resources by provisioning and managing large networks of virtual machines (VMs). For more details, see, *for example*: http://www.webopedia.com/TERM/O/openstack-nova.html

[11] See: http://superuser.openstack.org/articles/bringing-nfv-into-openstack-with-tacker

[12] A ceilometer is a device that uses a laser or other light source to determine the height of a cloud base. Ceilometers can also be used to measure the aerosol concentration within the atmosphere. See, for example: https://en.wikipedia.org/wiki/Ceilometer

2.6 OpenMANO

OpenMANO is an open source project that provides a practical implementation of the reference architecture for Management & Orchestration under standardization at ETSI's NFV ISG [12].

2.6.1 Project description and features

The ETSI's developed NFV Management and Orchestration architecture has served as the "main starting point" for many SDN-/NFV-*based* projects and architectures. OpenMANO is an effort undertaken by ETSI to realize the main functional components of that architecture through an open source development initiative. As the Management and Orchestration name implies, the ETSI architecture has two key components responsible for orchestration and for NFV management. The orchestration of services relevant to network operators may be handled by specific service orchestration layers which are separate from MANO framework. By moving the effort of developing the ETSI MANO architecture to the open source community, several benefits can be achieved throughout its development and maturity phases including a strong feedback channel towards ETSI ISG NFV, thereby supporting interoperability among NFV implementations.

Figure 5 provides a high-level view of the main components of the OpenMANO. On the top is the OpenMANO Graphical User Interface (GUI). As the name suggests, this component is the main window of interaction with the MANO server. While the main components of the architecture can be interacted through command line interface, the GUI is provided to access the main functionality of the MANO through a web interface (e.g. through common web browser applications). The second main component of OpenMANO architecture bears the same name (openmano) and is responsible for providing the reference NFV orchestrator functionality. The Openmano interfaces with the NFV virtual infrastructure manager (through openvim API) and offers a REST-*based* northbound interface (openmano API). The openmano provides different VNF functions including the capability to create and delete VNF templates, VNF instances, network service templates and network service instances. The last main component of the OpenMANO architecture is the ETSI NFV virtual infrastructure manager called as openvim. The openvim component interfaces with the compute nodes in the NFV infrastructure and an openflow controller to provide computing and networking functionality and to deploy virtual machines. The openvim also provides a northbound interface through which it offers cloud services including the creation, deletion and management of images, instances and networks. This interface is also based on REST (openvim API).

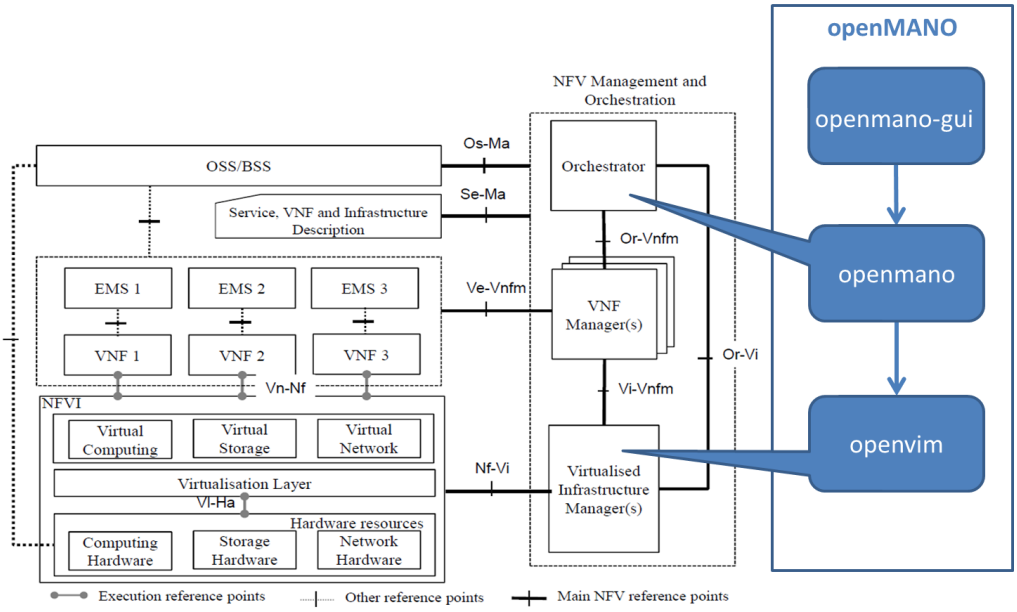

Figure 5: High-level OpenMANO architecture

2.6.2 Relevance to SESAME (Similarities / Differences)

The OpenMANO project has direct association with the effort undertaken by ETSI to present a NFV management and orchestration architecture. As SESAME also includes a cloud-*enabled* platform for providing service and network functions, and therefore will utilize the associated virtualisation tools, it can benefit from the open source tools provided by OpenMANO for infrastructure management and VNF orchestration.

- The OpenMANO project provides a complete stack of the NFV MANO architecture and therefore the tools for VNF orchestration can potentially be used for the orchestration tasks required in SESAME. Additionally, the interfaces among the key components of OpenMANO and the information flow across these interfaces can serve as an "input" to the specification and development of the relevant interfaces in SESAME architecture. Being an open source project, OpenMANO gives an additional benefit that the developed tools can be extended to suit new/different needs.
- While the complete MANO stack provided by OpenMANO can be beneficial in traditional SDN/NFV domains, it needs to be seen whether this creates any compatibility issues with other proprietary solutions for virtualisation, VNFs and services. The complexity of orchestrating composite VNF-PNF chains has to be addressed explicitly in SESAME orchestrator development.

2.7 Open Baton

Open Baton [13] is an implementation of the NFV Orchestrator, integrating with a generic VNFM built from scratch following the ETSI NFV MANO v1.1.1 (2014-12) specification [14]. All the interfaces between the main components follow the ETSI NFV MANO drafts. Open Baton aims to foster, within the NFV framework, the integration between the Virtual Network Function providers and the Cloud Infrastructure providers.

Open Baton was initially part of the OpenSDNCore (www.opensdncore.org) project started almost three years ago by Fraunhofer FOKUS with the objective of providing a compliant implementation of the ETSI NFV specification. Now it is one of the components of the Fraunhofer 5Gplayground Framework[13].

Designed for answering R&D requirements, so that it is easy to configure and to deploy and for providing a centralized view of the testbed, it does not contain any vendor-specific features. It follows open specifications and it is open to the community. Interoperability among different vendor solutions is one of the key challenges that Open Baton is trying to solve. The standard is not yet mature enough, so that by providing an open source implementation of the NFVO will support the development of a standardized NFV environment.

Open Baton extends the basic orchestration towards network functions management, including in its package a generic Virtual Network Function Manager (VNFM), which can be easily extended for supporting different type of VNFs, able to manage the lifecycle of VNFs based on their descriptors, and a generic EMS. A set of libraries is also available, which could be used for building your custom VNFMs (vnfm-sdk). A user-friendly Dashboard allows for easily managing all the VNFs. The NFVO is integrated by default with OpenStack, although it is possible to use different VIMs by using plugins. Indeed, it incorporates a system of plugins to extend its functionalities: it is possible to either use existing plugins or develop custom new extensions.

2.7.1 Project description and features

Open Baton is an open source project providing a reference implementation of the NFVO and VNFM based on the ETSI specification, it is implemented in java by using the spring.io framework, runs on top of multi-site OpenStack, provides independent infrastructure slices, and supports for generic or specific VNF management and a large amount of virtualisation use cases (e.g. core networks, M2M and Multimedia communication).

Open Baton Rel.2[14] provides many different features and components. It provides a network Function Virtualisation Orchestrator (NFVO) and a generic VNFM.

The Open Baton NFVO is completely designed and implemented by following the ETSI MANO specification and implements the key functionalities of the MANO architecture, thus allowing the installation, deployment and configuration of network services. The NFVO uses the ETSI NFV data model internally for the definition of the Network Service and Virtual Network Descriptors. It also interfaces with one or more VNFM(s).

Open Baton was designed to interact with multiple VIMs and fully integrate with OpenStack as main VIM implementation. In fact, Open Baton uses the OpenStack APIs for requesting virtual compute and networking resources. It currently uses OpenStack as the VIM for the NFVI in a Point of Presence (PoP). Open Baton controls the following operations of OpenStack: creation of virtual networks based on the requirements provided by VNFDs; virtual machines images

[13] The 5G Playground (https://www.fokus.fraunhofer.de/go/en/fokus_testbeds/5g_playground) enables the deployment of testbeds for proof-of concept as well as for performance, security and reliability evaluations into the different network areas. 5G Playground encompasses a comprehensive, highly customizable and re-configurable network environment, based on commercially available components and the Fraunhofer own toolkits: Open5GCore representing a new scalable, low delay and highly reconfigurable approach to core networks; OpenSDNCore addressing backhaul related features; Open Baton addressing NFV orchestration and Open5GMTC addressing the connectivity of a multitude of devices. The 5G Playground was designed to be easy replicated at customer premises.

[14] See, for example: http://sdn.ieee.org/newsletter/july-2016/open-baton

management; creation and management of virtual machines used for hosting the VNFCs. The NFVO uses the quota information provided by the VIM for reserving the resources required by each network service.

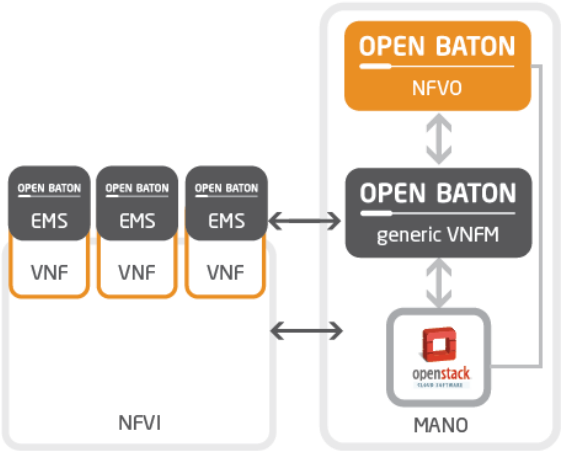

Figure 6: Open Baton framework

Following the ETSI NFV specification the OpenStack API is just one implementation of the VIM interfaces. The plugin mechanism implemented allows easily the extension towards the support of multiple cloud systems and additional VIM types. The NFVI provides virtualised resources (through a virtualisation layer) on top of which VNFs are executed. Plugins allow installing/starting and removing plugins at runtime, enabling different Virtual Infrastructure Managers and/or different Monitoring System. Open Baton uses the Remote Procedure Call (RPC) mechanism for implementing the plugins, this way, adding and removing different types of VIMs does not require re-writing anything in the orchestration logic.

Open Baton Orchestrator maintains an overview on the infrastructure, supporting dynamic registration of NFV PoPs. It receives virtual network function packages from different users, including VNF images and VNFDs.

The Open Baton environment includes multiple data centres and allocates resources on top of multiple OpenStack installations. The Orchestrator deploys on-demand the VNFs on top of an infrastructure consisting of multiple data centre instances (NFV PoPs).

The orchestrator also enables the deployment of multiple customized network slices and deploys in parallel multiple slices one for each tenant, consisting of one or multiple VNFs. Through this functionality the orchestrator provides a multi-tenant environment distributed on top of multiple cloud instances.

Open Baton enables the deployment of multiple virtual network services in parallel. Each network service is independently configurable and managed by the orchestrator: services are isolated (having their own compute, storage and virtual networks) and may be composed from multiple VNFs.

Open Baton supports several VNFM solutions. The VNFM works as intermediate component between the NFVO and the VNFs, particularly the Virtual Machines on top of which the VNF software is installed. In order to complete the lifecycle of a VNF, it interoperates with the Element Management System (EMS) acting as an agent inside the VMs and executing scripts

contained in the VNF package. This VNFM may be assigned the management of a single VNF instance, or the management of multiple VNF instances of the same type or of different types.

Open Baton provides a generic Virtual Network Function Manager and Element Management System which can be used for managing VNF Packages. The Generic VNFM is supposed to be used for any type of VNF that follows some conventions regarding: the VMs deployment; the script execution order; and the VMs termination. Furthermore, Open Baton provides a SDK which can be used for building VNF-*specific* VNFMs.

The Generic VNFM represents an implementation of the MANO function and can be extended to support the management of third party's VNFs. It is tightly coupled with the Open Baton EMS which runs as a software within the deployed Network Functions. The Generic VNFM (together with the Generic EMS) requests to the NFVO the allocation of specific resources for the virtual network instance. It is instantiated on demand by the NFVO and can execute the following operations:

- Request to the NFVO the allocation of specific resources for the virtual network instance.
- Request from the NFVO the instantiation, modification, starting and stopping of the virtual services (or directly to the VIM).
- Instruct the generic Open Baton EMS to save and to execute specific configuration scripts within the virtual machine instances.

It uses different mechanisms for interoperating with the VNFMs, either over a message bus using the PUB/SUB mechanism[15] or using a RESTful interface[16].

The Generic VNFM handles communications with the NFVO and with EMS. The communication NFVO ↔ VNFM ↔ EMS is done by using the AMQP protocol[17] over RabbitMQ[18].

Albeit Open Baton offers a generic VNFM which can be easily extended for different services; it also includes a set of mechanisms which enable the support for external VNFMs. Open Baton provides a set of libraries for integrating new network services. The NFVO supports two different mechanisms for interacting with custom VNFM:

- Publish/Subscribe mechanism using a message queue based on JMS.
- Publish/Subscribe mechanism using JSON[19] RESTful API.

Open Baton also integrates with additional components for the runtime management of a Network Service, the development of some of these are still in progress. For instance, it provides auto-scaling and fault management based on monitoring information coming from the monitoring system available at the NFVI level. It also provides integration with existing monitoring solutions.

The Open Baton auto-scaling engine can be used for automatic runtime management of the scaling operations of your VNFs. It is implemented in java by using the spring.io framework and

---

15 For more details, see, for example, the information found at: http://www.enterpriseintegrationpatterns.com/patterns/messaging/PublishSubscribeChannel.html

16 See, for example: http://rest.elkstein.org/

17 https://www.amqp.org/

18 For more details see, among others: https://www.rabbitmq.com/tutorials/amqp-concepts.html

19 https://en.wikipedia.org/wiki/JSON

runs as an external component, communicating with the NFVO via Open Baton's SDK. Additionally, the Auto-Scaling System uses the plugin mechanism to allow various Monitoring Systems to be used (Open Baton uses Zabbix[20] as the monitoring system in their default set up). An Auto-Scaling Policy defines conditions and actions in order to allow automatic scaling at runtime. The list of AutoScalePolicies is defined at the level of the VNFD/VNFR.

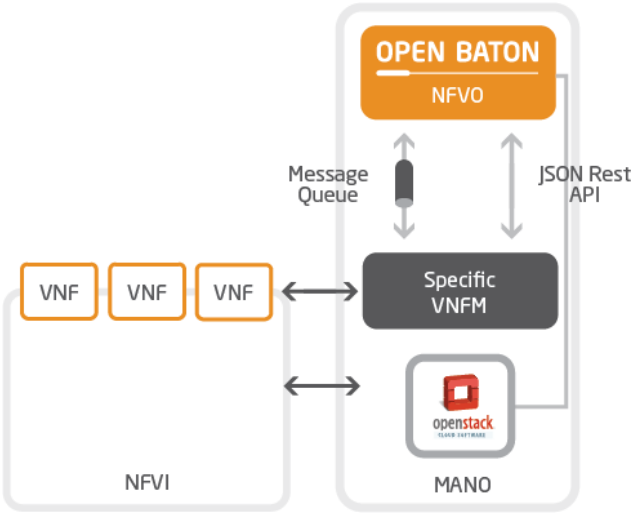

Figure 7: Open Baton external VNFM support

The Open Baton fault management (FM) system can be used for automatic runtime management of faults which may occur at any level. The Open Baton fault management NFVO module manages the alarms coming from the VIM and executes actions through the NFVO. A fault management policy needs to be presented in the VNFD. The Open Baton FM is a rule-*based* system. Such rules are specified in Drools language and processed by the Drools engine in the Open Baton FM.

For the monitoring, Open Baton integrates with the Zabbix monitoring system using RabbitMQ. The Zabbix plugin is an open source project that provides a reference implementation of two interfaces of the VIM, based on the ETSI NFV MANO specification. The two interfaces are: VirtualisedResourceFaultManagement and VirtualisedResourcePerformanceManagement. In particular, with the Zabbix plugin it is possible to create/delete items, trigger an action on-demand. Some of the benefits introduced by the usage of such plugin are given as below.

1) The consumers (NFVO, VNFM) are independent to the monitoring system.
2) The communication between the consumers and zabbix-plugin is JSON-*based*, so the consumers can be written in any languages.
3) The values of the items are cached and updated periodically in order to avoid contacting the zabbix server every time when a specific metric is required.
4) If your consumer is written in java, we provide a simple class MonitoringPluginCaller which handle the communication via RabbitMQ.

Finally, Open Baton includes a user friendly dashboard, a web GUI (implemented AngularJS) from which the users can design and manage their own Network Services and control the

[20] Zabbix is enterprise open source monitoring software for networks and applications. It is designed to monitor and track the status of various network services, servers, and other network hardware. See, *for example*, the information provided in: https://en.wikipedia.org/wiki/Zabbix

infrastructure. The Dashboard enables the management of the complete environment, allowing the easy understanding and modification of the dynamic registered NFV PoPs. It provides an interface for the creation (install, deploy, configure) of the network services and the overview of the deployed ones. The dashboard enables the management of the complete environment including:

- The control of the infrastructure – enabling the easy understanding and modification of the dynamic registered NFV PoPs.
- The management of the deployed network services – their creation and the overview of the deployed services.
- The generic VNFM can be easily extended to support the management of third party VNFs.
- Browse the VNF Catalogue, compose multiple VNFs together for building several scenarios: vIMS, v5G, vEPC, vM2m, etc.
- Manage the Network Service deployment on top of the NFV Infrastructure (NFVI) and get an overview of the utilized resources.

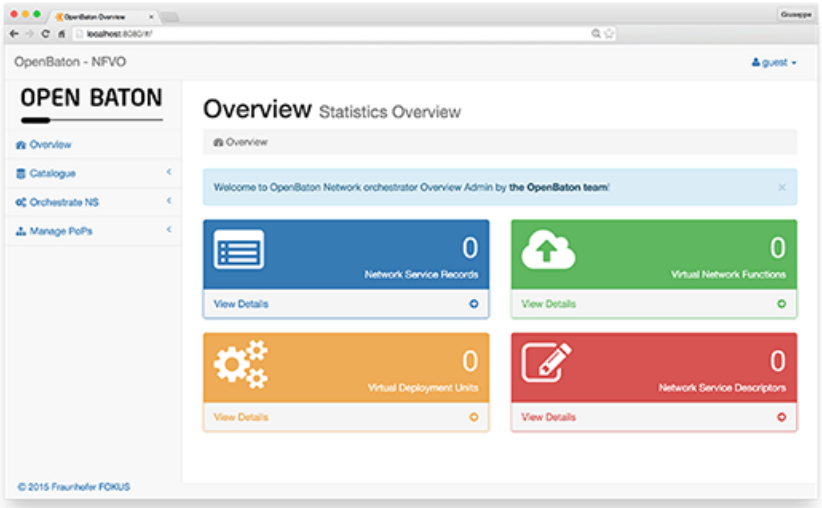

Figure 8: Open Baton dashboard

2.7.2 Relevance to SESAME (Similarity / Differences)

Open Baton is yet another available framework for orchestrating VNFs in OpenStack based virtualization environment and follows the ETSI MANO architectural framework. From the perspectives of the SESAME functional and orchestration requirements (VNF orchestration and service chaining), the Open Baton does provide a generic VNF orchestration platform that can be potentially applied in the SESAME project. But similar to other options, the focus has been on legacy networks and not on resource limited network edge which is the main focus point in SESAME project. The compatibility with the SESAME platform is also not guaranteed. Additionally, SESAME NFVO may require more components or functionalities than the Open Baton plugin system currently supports.

2.8 Cloudify

Cloudify [15] is another open source cloud orchestration framework aimed at automating the NFV orchestration and management tasks.

2.8.1 Project description and features

The main focus point of Cloudify framework is the automation of the entire life cycle of applications and services, including deployment on common cloud and data centre environments, a real-time monitoring of performance including fault detection, and an automatic recovery and maintenance. In simple words, Cloudify provides an end-to-end lifecycle orchestration solution for network applications. Cloudify requires the applications to be described in "blueprint" in a human-readable form (e.g. using YAML[21]) including the application's required infrastructure, middleware, code, scripts, configuration etc., in order to "enable" swift orchestration. The application's blueprint can be very detailed, describing a fine-grained configuration for all required components. In essence, one can define the complete lifecycle of all parts of an application. Cloudify can then easily deploy the application and manage it using various tools of choice of the user.

The deployment phase includes launching the required compute instances, configuring the network, providing the required storage and security etc., to create the virtual infrastructural resources required for the application deployment. Cloudify can execute the instantiation scripts or launch configuration management tools to configure the servers and deploy the middleware and code. Once the application is fully deployed, Cloudify also takes care of its management issues by providing custom "workflows" to change its structure when required. An application can be closely monitored and logs can be collected in order to act upon different scenarios. Cloudify also allows the automation of this process by providing tools to aggregate data, visualize it (externally/internally) and take automated actions upon them. Fault-detection and recovery/healing are also part of this process and can be automated.

A final and important part of Cloudify framework is "pluggability". Plugins are essentially feature add-ons that can be used to enhance the functionality of the framework or provide additional automation. Cloudify-specific plugins can run different scripts, launch CM tools, collect statistics and so on. Plugins for Cloudify are written in python and therefore are easy to write and integrate. Several plugins are available with Cloudify and additional ones can easily be written and integrated. Cloudify uses TOSCA for its domain-specific language and therefore complies with the standardization efforts going on in this area.

2.8.2 Relevance to SESAME (Similarity / Differences)

As described above, Cloudify provides essential lifecycle management and configuration of network applications. This is an important and essential requirement of VNFs running on top of a virtual infrastructure. The compatibility with all well-known virtualisation and cloud platforms is an added benefit provided they could be used in SESAME project scope.

- In SESAME, the scope of orchestration also shares the same required functions of lifecycle management and configurations as addressed in Cloudify. Therefore, the tools developed in Cloudify for application orchestration can potentially be used in SESAME project.

[21] For more explanations see, *for example*: https://en.wikipedia.org/wiki/YAML

- Although deployment and configuration is an important aspect of VNFs, management and control is expected to be an issue of higher relevance in SESAME. As with other orchestration platforms designed with legacy data-centres in mind, the Light DC and CESC concepts in SESAME may present some performance bottlenecks for the existing solutions including Cloudify.

2.9 Open Source Mano

Open Source Mano (OSM) is an ETSI-*hosted* project to develop an open source NFV MANO software stack aligned to the ETSI ISG NFV [19]. The OSM MANO group includes industries and research institutions. The project launched the first release of the software (i.e.: Release 0) in May 2016 [20]. Furthermore, the OSM project aims to position itself as an evolution of the ETSI MANO framework towards an open source code which can be used to implement an ecosystem driven by NFV and NS development for different use cases. The high-level architecture developed by the OSM project is shown in Figure 9.

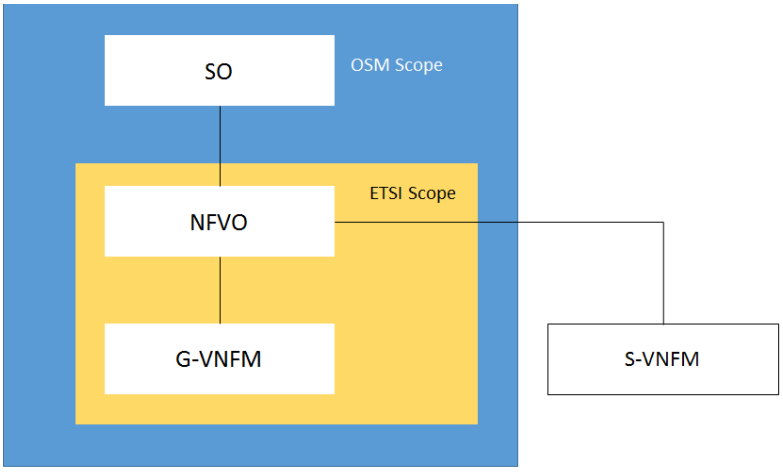

Figure 9: Open Source Mano (OSM) high-level view [21]

2.9.1 Project description and features

The OSM Group is currently developing an open source NFV management and orchestration stack relying on well-known and widely used open source tools. The activity is aligned with the ETSI ISG NFV development and it complements those activities. Particularly, this project relies on the OpenMano project [12], Canonical's Juju Charms[22] and Rift.io orchestrator. One of the pillars of OSM consists in developing an information-model-*driven* architecture, which allows different components of the MANO stack to be interconnected. Through descriptor models, OSM is capable of deploying NFV and NS in a time scale in the order of minutes in a fully automated manner. To achieve this objective, OSM will rely on the following features:

- End-to-End (E2E) service creation.
- Enhanced Platform Awareness (EPA), which ensures high performing virtual nodes.
- SDN control to ensure sufficient bandwidth requirement for the links.

[22] https://jujucharms.com/

- Multi-site capability for multiple sites involved in the E2E process creation.
- Multi-cloud VIM capability, since several VIMs shall coexist.
- Deployment and configuration of multi-tenant and single tenant NFV and NS.

A more detailed view of the OSM architecture is shown in Figure 10. The Resource Orchestrator (RO) is responsible for coordinating the allocation of resources for the different NFV across multiple VIMs and multiple sites involved in the E2E process creation. Moreover, the RO is responsible for processing the resources which are required to deploy a VNF, as detailed by the related VNF descriptor file. In this respect, the RO cooperates and drives the VIM to allocate the necessary storage, compute and network resources which are required in the deployment of NFV and NS, including the necessary network connectivity. The RO layer is broadly aligned with the concept of NFVO in ETSI ISG NFV.

The Service Orchestrator (SO) provides the network-wide level of orchestration, which is needed by the creation of the E2E service through the RO and the different VNF configuration components. In addition, the SO provides a single interface for NS lifecycle management, on-boarding of models for VNF/NS creation, translation between different data models, management of catalogues for VNF/NS, configuration management of VNF/NS and role-*based* access control.

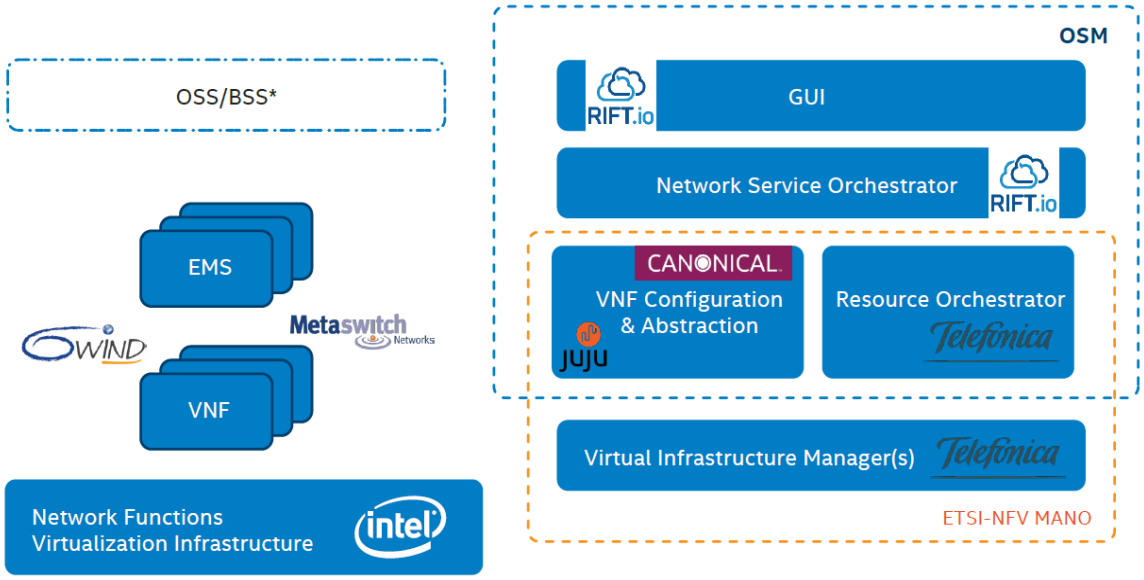

Figure 10: OSM detailed architecture [21]

2.9.2 Relevance to SESAME (Similarity / Differences)

The Open Source MANO project develops carrier-grade virtual services in order to address multiple use cases, while at the same time trying to ensure that telco-grade SLA can be achieved. Given the large support received by telecommunications operators, vendors and research institutions, the SESAME project will pay increasing attention to the OSM project. Recently the group behind OSM did the first software release and a subsequent release is expected in the short term. In specific, SESAME will follow the OSM project with respect to

- The choice of the Resource Orchestrator and the duties assigned to it.

- The creation of end-to-end services in a virtualised environment, which is of great importance within the context of SESAME.
- The opportunity to address several and relevant use cases at societal level.
- An open source implementation of a data model aligned with the information model developed within the ETSI ISG NFV framework.

3 NFVO in CESCM

3.1 The role of NFVO

NFV implementations need to empower (V)SCNOs to flexibly manage both PNFs and VNFs, mainly through VNFM(s) and PNF EMS. Additionally, NFVI must have the elasticity to piece together the dynamic configuration of VMs with respect to processing power and storage resources. On top of that, the NFVO provides the capability to control and assemble the network services delivered to VSCNOs. In light of this, NFVO needs to perform the following tasks:

- The network service orchestration, which includes the lifecycle management of NSs (i.e. chain of VNFs and PNFs), should enable the SCNO to respond automatically to changes according to the defined SLAs and KPIs. The up/down and in/out scaling of VNFs must be supported by the VNF Manager, as defined by the ETSI Management and Orchestration (MANO) architecture [14].
- The resource allocation on NFV infrastructure (NFVI) for VNF placement, i.e. the Light DC in SESAME ecosystem, must be able to offer the resources for the NS instantiation. The NFVO via the coordination with virtual infrastructure managers (VIMs) should handle the VM instantiation and instructing the SDN controller to form the NSs.

SESAME NFVO not only must provide all general previously mentioned functionalities, but also needs to address some specific challenges:

- With SESAME network service orchestration, operators have to simultaneously manage services composed by both physical and virtual network functions, which is the main distinguishing feature of SESAME NFVO in comparison with the state-of-the-art existing orchestrators (as introduced in *Section* 2).
- There must also be an ability to manage the lifecycle of network services across the SESAME platform from a single GUI, tailored to the needs of (V)SCNOs.
- Additionally, the system must ensure network service elasticity and automatic updates of the service connectivity. This will require a stateful inventory that provides persistent and accurate data for all processes, including its native services information model, and topology.

3.2 NFVO and CESCM Interdependencies

To complete the picture of the SESAME NFVO ecosystem, the following text describes the modules and repositories that hold and exchange different information with the SESAME NFVO.

3.2.1 Local Catalogues

The Local Catalogues are a set of repositories that hold useful information for the NFVO operation. The following catalogues are required to support the correct functionality:

- *NS catalogue:* repository for all the NS descriptors (NSD). NSDs are deployment templates of network services that contain description of VNFs and virtual link connections (forwarding graph). It is used by the orchestrator in the process of service instantiation and lifecycle management.

- *VNF catalogue:* repository for all VNF descriptors (VNFD), which are deployment templates describing the requirements of the available VNFs and their characteristics. It is used by the orchestrator in the process of service instantiation and lifecycle management.
- *VNF placement algorithm database:* repository for all the mapping/allocation algorithm scripts.
- *NFV and NS instances catalogue:* repository that holds the details of the NS and the VNF instances in the system that are already up and running.
- *Infrastructure repository:* repository that stores the detailed information of CESC cluster hardware resources availability (i.e. Light DC storage, networking and computational resources).

3.2.2 SLA monitoring

SLA monitoring is a real-time monitoring element that ensures the enforcement of agreed SLAs between the business role players (e.g. SCNO and VSCNO). Taking monitoring information from EMS, and also from the NFVO if required, the SLA Monitoring is able to evaluate the level of conformance between the current service status and the KPIs defined in an SLA. Should an SLA violation occur, the SLA monitoring module will trigger reconfiguration alerts. The NFVO and the EMS then need to initiate the pertinent service adjustments (reconfiguration, migration and scaling of the existing services) to comply with the service agreements while dynamically maximizing the utilization of the resources.

3.2.3 CESCM Portal

The CESCM Portal provides a graphical user interface for both SCNO and VSCNO to access the SESAME platform. The CESCM portal in general provides visual monitoring information of the agreed SLAs and the possibility to request network services using the available VNFs in the catalogues. The different CESCM Portal capabilities are exposed to the users based on authentication and authorization mechanisms, in order to provide appropriate security measures. The communication between the CESCM portal and the NFVO is done through the CESCM Northbound Interface.

3.2.4 VNFM

The VNF Manager (VNFM) is the element responsible for both service VNFs and SC VNFs instances lifecycle management, including: VNF performance, setup configuration and accounting management, VNF lifecycle change notification, and VNF scaling and policy administration. In simple words, SESAME VNFM plays the same role as the VNFM functional block defined in the ETSI NFV MANO documentation [14].

3.2.5 VIM

The Virtual Infrastructure Manager (VIM) is the unit that ensures synchronization between physical and virtual resources. Moreover, it takes care of activities such as end-to-end management, on-demand provisioning, and backup, etc. Within the scope of SESAME it also includes the SDN controller thereby providing advanced network programmability as well. In general, VIM is a key component of the NFV-MANO architectural framework responsible for

controlling and managing the NFVI compute, storage and network resources within one SCNO infrastructure domain. It is in direct communication to the NFVO and the VNFM.

4 NFVO Logical Architecture

As stated in the previous section, the SESAME NFVO is responsible for undertaking the following functionalities:

- **Resource orchestration:** coordinating, engaging and releasing CESC cluster resources via a direct communication with VIM(s) within one or among different point of presence (PoPs).
- **Service orchestration:** creating, terminating, monitoring and scaling of end-to-end service chains through coordination with respective VNFM(s). Under the scope of SESAME, a NS includes both SC VNFs (to support data transmission to User Equipment (UE)/ Evolved Packet Core (EPC)) and service VNFs (to provide added value edge services such as video transcoding).
- **Performance monitoring:** constant monitoring of the running NS instances during their lifecycle and consequent decisions (e.g. NS scaling) to assure that the service performance level "meets" the requirements. That includes activities such as increasing/decreasing resources (e.g. CPU, RAM, storage) for the VNFs constituting a service chain (scale up/down), and/or instantiating a parallel service chain (scale in/out). The terms scaling out/in/up/down in SESAME follow the definition provided by ETSI NFV MANO specification [14].

The interaction between the orchestrator and the external components inside SESAME architecture, mainly the other modules of the CESCM, is done through the categorized interfaces listed below (more detailed information about interfaces are available in *Section* 4.2).

- **Southbound interface:** to provide monitoring/communication between the NFVO, the VIM and the VNFM(s).
- **Westbound interface:** to support data exchange between the NFVO and the local catalogues.
- **Northbound interface:** to support data exchange between the NFVO and other CESCM modules such as the CESCM Portal and the SLA monitoring as well as between the NFVO and external modules such as the Network Management System (NMS).

Some basic features of these interfaces are as follows:

- **Low latency:** to minimize the response time once an error condition is detected, or a threshold condition is exceeded.
- **Scalability:** to support different volume of NFVO/external systems interactions.
- **Resiliency:** to support information delivery in the occasion of infrastructure failure or performance degradation (due to overload).

This section reviewed the high level logical architecture proposed for the SESAME NFVO. Next, detailed characteristics and specifications of the NFVO logical components and NFVO interfaces are presented.

4.1 NFVO logical components

This subsection presents a high level view of the SESAME NFVO logical architecture. Considering the described NFVO responsibilities, it is possible to draw the architecture with five logical components, each with a dedicated functionality. As depicted in Figure 11, NFVO coordinator is the central component in this architecture, which acts as the main data exchange hub between the internal NFVO components and the external elements such as: SLA monitoring, local catalogues, VIM, VNFM, CESCM portal and external NMS. Such an interaction includes receiving information from the CESCM elements and directing it towards the appropriate NFVO components and/or communicating a process result to the external CESCM elements. In the following subsections, the responsibilities of NFVO logical components are briefly reviewed.

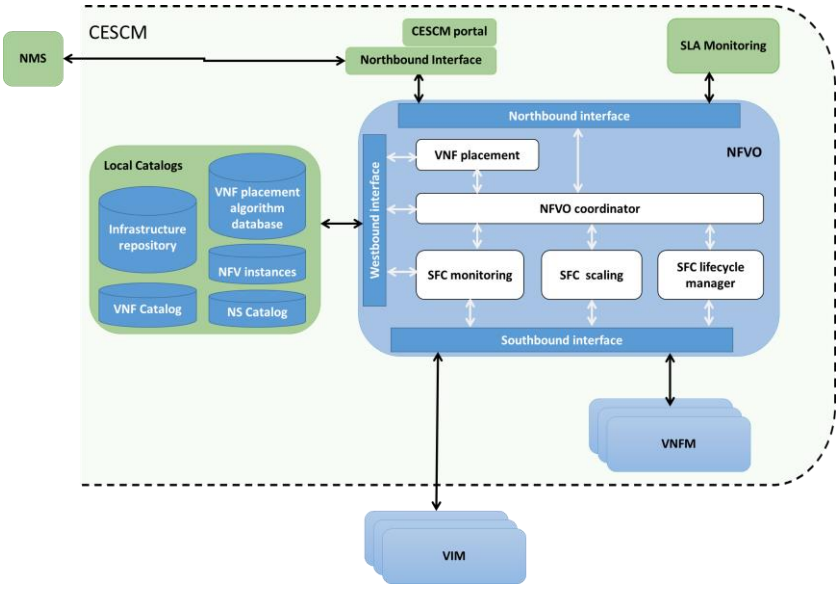

Figure 11: High level SESAME NFVO Logical Architecture (NMS is for both SCNO and VSCNO)

4.1.1 NFVO coordinator

The NFVO coordinator is the core of the orchestrator, and it represents the central information convergence point and the main request manager. The primary role of this component is to coordinate the interactions among the internal NFVO components. That includes breaking down a big activity (e.g. service instantiation) into smaller logical tasks and distributing responsibilities among other relevant internal logical components. Moreover, it manages the data exchange between the NFVO and the external world, i.e. SLA monitoring, local catalogues, VIM, VNFM, CESCM portal and external NMS. This responsibility includes format validation and template generation to ensure communications between NFVO and other elements (e.g. descriptor validation against defined schemas, HEAT template generation[23]).

4.1.2 SFC lifecycle manager

The SFC lifecycle manager is the module responsible for the provisioning and management of network services over the Light DC. Based on ETSI documentation [22], network service (NS) is defined as a forwarding graph of interconnected (virtual) network functions and end points (that includes both functional and behavioural aspects of a network service). The SFC lifecycle

[23] For more relevant data, see: http://www.ijecs.in/issue/v4-i5/12%20ijecs.pdf

manager provides service level lifecycle management functions (create, edit, delete, start and stop). Once the NFVO coordinator validates a service request and receives the placement inputs, the service lifecycle manager takes care of the service creation through appropriate communications with VIM, including the SDN controller, and VNFM. Traditionally, communication with VIM and VNFM means resource and service orchestration, respectively.

- **Resource orchestration:** Managing the resources of different VIMs (when there are multiple VIMs) in the same or different PoPs. This involves: coordination, authorization, releasing and engaging CESC cluster.
- **Service orchestration:** Manages/coordinates the creation of an end-to-end service that involves VNFs from different VNFMs domains. These VNFs may be managed by different VNFMs - instantiated when/where necessary/applicable. Note that service orchestration is achieved via coordination with respective VNFM(s) and not through direct communication with VNFs.

4.1.3 VNF placement

The VNF placement module indicates the exact place to deploy a NS (constituted of service VNFs and SC VNFs). In an ideal case, the whole NS is hosted by a single CESC, although it is possible to have cross-CESC service chains in order to leverage the distributed computation capacity of the whole cluster. In general terms, the placement procedure is as following:

1. Retrieve an appropriate placement algorithm from the algorithm database depending on the optimization objective (e.g. minimum resource utilization, minimum energy consumption, etc.).
2. Retrieve the status of resources (from the infrastructure repository).
3. Indicate where to implement the VNFs according to the selected placement algorithm and the current status of the resource usage.
4. Indicate the connection points between the VNFs in the requested service chain.

The VNF placement problem has risen with the trend of network virtualisation, in which intermediate network functions traditionally hosted on physical middle-boxes are deployed in software. Some common examples of such network functions include Network Address Translator (NAT), load balancer, firewall and deep packet inspection. As already mentioned, in the case of SESAME, they include also video transcoding unit, caching and small cell related VNFs. The main advantage of network virtualisation is provided by the possibility to remove the dependence of operators from special-purpose hardware platforms, which are expensive and inflexible to upgrade.

Placing VNF in virtualized infrastructure often entails formulation of a problem in which one is willing to optimise a network-*specific* utility function under a (wide) set of constraints. Generally, the problem is NP-hard to solve and requires extensive computer simulations, as well as to devise heuristic solutions[24]. In this regard, interesting parallelisms can be found with well-known problems such as the traveller salesman problem[25] and the facility location problem[26], which are both NP-hard to solve. The interesting and tightly related problems associated with Virtual

[24] For further information see: https://en.wikipedia.org/wiki/NP-completeness
[25] See the informative description provided in: https://en.wikipedia.org/wiki/Travelling_salesman_problem
[26] See the informative description provided in: https://en.wikipedia.org/wiki/Facility_location_problem

Network Embedding (VNE) are also worth to mention. In this field, there exist several activities which have been undertaken in the form of whitepapers and memo by standard bodies as highlighted in [23] - [27].

As already mentioned, several research studies can be found in the literature on the topic of VNF placement as shown in [28] - [32]. Clearly this cannot be considered an "exhaustive" search but rather as an overview useful to shed some light onto the general research problem. In [28] the authors develop a Mixed Integer Quadratically Constrained Program (MIQCP) to find the solution to the virtual network functions placement problem and "how to chain them under realistic constraints provided by network resource limitations and specific requirements of the VNFs". In [29] the authors develop an Integer Linear Programming (ILP) formulation for placing virtual network functions, while routing traffic across the network. The work defines an objective function that minimises the total bandwidth used by all flows in the network. The work developed in [30] provides, instead, a Mixed Integer Linear Programming (MILP) formulation, in which the problem of VNF placement and routing of traffic through the network is solved jointly. In this work the objective function is formulated in terms of minimising the maximum link utilisation and the available CPU cores where to run the different VNFs. In [31] an ILP formulation to the VNF orchestration problem is developed. In particular, the authors provide a dynamic programming based heuristic and they solve it for large instances of this problem. In this case the objective consists in minimising the overall network resources utilisation and placing the VNF optimally. Finally, in [32] the authors study the relevant problem of performance isolation in a multi-tenant network, identifying the performance degradations which are due to a tenant separation that is mainly logical, and they propose an online link utilisation feedback solution, which is experimented in a real setup exploiting the OpenFlow protocol. Finally, some preliminary work on Wireless and Radio Virtual Network placement and the associated performance isolation and efficient spectrum utilization challenges can already be found in the literature. For example, in [33],[34] the authors formulate a VNF embedding problem where Radio access is treated as a VNF. The authors define a new formalism for the network request as well as for the placement taking into account channel conditions and VNF scheduling.

SESAME shall leverage on the existing literature on the topic of VNF placement to develop new solutions and algorithms which fit the specific setup, use cases and constraints provided by the project.

4.1.4 SFC monitoring

Retrieving service-*related* metrics is another main task of the NFVO performed by the SFC monitoring component. That means collecting the individual VNF monitoring data and creating a service-level monitoring log file. According to the retrieved metrics values, SFC monitoring may detect performance failures and raise a flag to inform the NFVO coordinator to take an appropriate action (e.g. instantiate a new NS to replace the broken chain) for ensuring the service operation. The overall activity includes performing service level test during both, the network service instantiation process and the service lifecycle period.

Note that the service-level monitoring data may include inter-/intra- CESC cluster connection qualities. To support real-time monitoring, this unit needs a direct communication with the VIM(s). It also includes interfaces to the CESCM local catalogues to store all the service-level monitoring information.

4.1.5 SFC scaling

The SFC scaling unit carries out service-level scaling following the characteristics derived by a joint cooperation between the SFC monitoring and the SFC Lifecycle modules. Under the SESAME context, service-level scaling functionality is handled automatically whenever an SLA is violated. In such occasion and depending on the adopted/agreed scaling policy, one or more of the following ETSI scaling suggestions may apply [22]:

- **Scale Up/Down:** Increase/decrease of resources (e.g. CPU, RAM, storage) to VNF constituting a service chain.
- **Scale Out/In:** Instantiate parallel VNFs in a service chain. In this case, VNF to VNF connectivity in the scaled service chain is a concern to address.

4.2 NFVO interfaces

Three different interfaces for internal and external communications have been defined for the NFVO. Following the current trend of naming the interfaces, they have been labelled using the cardinal points corresponding to their placement on the NFVO architecture diagram: **northbound interface**, **westbound interface** and **southbound interface.**

The northbound interface acts as an abstraction layer to the internal modules of the NFVO for the requests coming from the CESCM Northbound Interface. The southbound interface is used by the NFVO components to communicate with the VNFM and the VIM. The purpose of the westbound interface is to abstract the communications with the local catalogues and repositories of the CESCM.

The next subsections provide functionality description of each interface and their interconnection details with respect to the CESCM internal modules.

4.2.1 Northbound Interface

The Northbound Interface[27] of the NFVO follows the definition of a northbound interface, as it was popularised by the SDN nomenclature. It is placed in the upper part of the NFVO architecture diagram and provides an abstraction layer that serves as an entry point to the internal NFVO components.

As such, this interface is used to collect requests coming from other CESCM components, in particular, the CESCM Portal and the SLA Monitoring, and the external NMS of the SCNO and VSCNOs. However, all the communication from previous components with the NFVO's Northbound Interface comes from the CESCM Northbound Interface.

The NFVO's Northbound Interface connects directly to the orchestrator's internal NFVO coordinator module. Therefore, it provides access to the lifecycle management functionality of the network services composed by Service Function Chains (SFC) of one or more VNFs, either SC VNFs or Service VNFs. Furthermore, this interface allows the SLA Monitoring to send alerts related to an SLA violation to the NFVO in order to trigger an appropriate re-scaling policy. A more detailed view of the workflows that represent those processes is provided in *Section* 6.

[27] See, for example: https://en.wikipedia.org/wiki/Northbound_interface

In order to prevent potential security issues, every call to the NFVO's Northbound Interface needs to be done by authorised users and modules. Furthermore, incoming requests to the NFVO Northbound Interface are validated prior to processing.

4.2.1.1 Communication with CESCM Portal

The CESCM Portal offers a graphical interface for interacting with the functionalities provided by the NFVO. The requests from the Portal are sent through the CESCM Northbound Interface which processes and redirects them to the Northbound Interface of the NFVO.

The capabilities that are exposed by the NFVO through its Northbound Interface are related to the lifecycle management of SFCs. Using the CESCM Portal the VSCNO can either define or select SFCs from the catalogues and perform operations on them that are transparently managed by the NFVO. The available operations from the CESCM Portal on the NFVO's Northbound Interface are the following:

- *Create an SFC:* The CESCM Portal sends a request containing all the relevant information for the NFVO to be able to create the specified SFC. It has to contain the type of VNFs that are required for the chain, how they are connected to each other by means of virtual links and, optionally, specify configuration parameters for the VNFs to be set up at the deployment phase.
- *Modify/update an SFC:* A request can be made from the CESCM Portal to modify the active SFC, by including a new Service VNF, or by changing a forwarding path. The information that has to be sent to the NFVO's Northbound Interface needs to contain the identifier of the SFC to be modified and an updated description template with the changes to be done.
- *Terminate an SFC:* Running SFC can be terminated manually by selecting the corresponding option in the CESCM Portal. An SFC termination request is then sent to the NFVO, containing the identifier of the SFC to be finished.
- *Get an SFC monitoring status:* The CESCM Portal queries the NFVO to get and present the information about the current status of a specific SFC. The request has to contain the SFC identifier. Optionally it could include more specific parameters in order to filter the request to the NFVO.

4.2.1.2 Communication with SLA Monitoring

The SLA Monitoring module is in charge of evaluating the SLAs agreed between the SCNO and the VSCNOs. The "Self-x" features and the automatic configuration mechanisms that can be placed in the EMS are aware of the status of the VNFs under an SLA and can react to changes in the network to better adapt the individual performance of those VNFs. However, there may be situations where a change in the whole SFC is required to guarantee the SLA terms. Therefore, when the SLA Monitoring detects an SLA violation, it sends a notification alert to the NFVO. The NFVO then processes the alert and depending on the available resources it can initiate a re-scaling procedure. The purpose of the re-scaling can be to increase the allocated resources for a VNF, increment the number of VNFs with certain functionality, or conversely, remove duplicated VNF instances or free resources to avoid a SFC using more resources than those initially granted to the VSCNO.

The communication from SLA Monitoring goes through the CESCM Northbound Interface to the northbound interface of the NFVO. Alternatively, the SLA Monitoring could also send notifications directly to the NFVO without passing through the CESCM Northbound Interface, leaving the validity checks of the SLA Monitoring alerts to the NFVO itself.

4.2.1.3 Communication with NMS

The NMS is an entity outside the CESCM that carries out the management of the network encompassing a wider view beyond the CESCs. For that purpose, the NMS interacts with different elements of the CESCM such as the PNF EMS, SC EMS or the NFVO. The communication between the NFVO and the NMS is done through the CESCM Northbound Interface and covers the functionality described in the ETSI NFV Os-Ma-nfvo reference point [35], which supports the following operations:

- *Network Service Descriptor (NSD) management:* A NSD is a template that describes the deployment of a NS including service topology (constituent VNFs and the relationships between them, Virtual Links, VNF Forwarding Graphs), as well as NS characteristics and any other artefacts necessary for the Network Service on-boarding [14] (i.e. for including the NS in the local catalogue) and the lifecycle management of its instances. Through the CESCM Northbound Interface, the NMS can perform different NSD management operations, such as on-boarding an NSD, querying an NSD to obtain details about its attributes, enabling an NSD so that it can be used to instantiate an NS with this descriptor, disabling an NSD, updating an NSD to create a new version of an already on-boarded NSD and deleting an NSD in the NFVO. The NMS can also manage other descriptors associated to an NSD, such as the PNF Descriptor (PNFD), the Virtual Link descriptor (VLD) and the VNF Forwarding Graph descriptor (VNFFGD).
- *NS lifecycle management:* This allows the NMS to perform different lifecycle operations over the NSs, such as instantiating a NS, querying a NS (e.g. retrieving summarised information about Light DC resources associated to a NS instance, or to a VNF instance within the NS instance), terminating an NS and healing an NS (e.g. to restore the state of the NS before a failure occurrence, or to trigger specific actions to recover from a failure).
- *NS lifecycle change notifications:* This allows the NMS to subscribe for receiving notifications from the NFVO when changes in the lifecycle of NS instances occur.
- *NS performance management:* This is used to provide the NMS with different performance information about the different NS instances. Performance information includes measurement results and notifications about the virtualised resources used by the NS instance, or about the performance of the VNFs composing the NS. Collection and reporting of performance information is controlled by Performance Management (PM) jobs that group details of performance collection and reporting information. Then, the NMS can create a PM job indicating a group of performance metrics to be collected for a set of NS instances. Similarly the NMS can subscribe for receiving notifications related to performance information and it can specify threshold levels on specific performance metrics for which notifications will be generated when crossed.
- *NS fault management:* This allows the NFVO to provide the NMS with alarms related to Network Services. An alarm on a given NS results from either a collected virtualised resource fault impacting the connectivity of the Network Service instance or an alarm

associated to a VNF that belongs to the NS. The interface supports operation for subscribing to notifications related to alarms, for notifying the alarms and for getting a list with all the active alarms of a NS.

- *VNF package management:* A VNF package is an archive that includes a VNF Descriptor, the software image associated with the VNF as well as other artefacts, e.g. to check the integrity and to prove the validity of the archive. Through this Northbound Interface, the NMS can carry out operations such as on-boarding VNF packages, querying, disabling, enabling and deleting VNF packages, as well as subscribe and notify operations for VNF package changes.
- *VNF lifecycle management:* This allows the NMS to perform VNF lifecycle management operations. In this case, the NFVO forwards the received requests to the VNFM through the southbound interface described below in *Section* 4.2.2.2.
- *VNF lifecycle change notifications:* This allows the NMS to request subscription to receive notifications of changes related to lifecycle management operations of VNF instances.

4.2.2 Southbound Interface

The NFVO's Southbound Interface[28] serves the same purpose of the Northbound Interface, i.e., it provides a simplified view of the APIs of underlying elements. The name of the interface reflects its position in the NVFO internal architecture diagram.

The internal sub-modules of the orchestrator that make use of the Southbound Interface are the NFVO coordinator, the SFC monitoring, the SFC scaling and the SFC lifecycle manager, i.e. all those that require access to the VIM to manage the Service Function Chains.

All the communication going out from the NFVO goes through this interface, which then translates and redirects the requests to the VNFMs and the VIMs. As such, the interface complies with the functionality defined by the ETSI NFV MANO [36], i.e. reference points Or-Vnfm between the NFVO and the VNFM and the Or-Vi between the NFVO and the VIM. Additionally, SESAME defines the S-Or-Vnfm and S-Or-Vi as extensions to the previous reference points in order to cover future SESAME specific requirements for the NFVO communication with the VNFM and the VIM respectively.

4.2.2.1 Communication with VIM

The VIM is in charge of managing the virtual infrastructure layer that is created on top of the physical resources of the micro servers in the CESCs. It provides a unified view of the micro servers in a cluster, i.e. the Light DC, to users and higher level tools.

The NFVO asks the VIM for available resources to deploy the VNFs and to create VMs for the VNFs. Through the VIM, the NFVO can also access the SDN Controller to request the creation of virtual paths that connect different VNFs, making up the Service Function Chains.

The VIM functionality that is available to the NFVO through its southbound covers the ETSI NFV specification:

[28] http://whatis.techtarget.com/definition/northbound-interface-southbound-interface

- *Light DC resource reservation/release:* The NFVO requests the VIM to reserve a set of virtual resources where a VNF will be deployed later.
- *Light DC resource allocation/release/update*: After the resource reservation has been made and the NFVO have been done all the deployment feasibility checks, it sends a resource allocation request to the VIM. In case a scaling process is initiated, the NFVO can also ask the VIM to update the amount of allocated resources, either increasing or decreasing them.
- *Software image addition/deletion/update:* The NFVO can send the VIM software images to be deployed in the Light DC. Those images can be stored by the VIM so the corresponding update and deletion commands can also be requested.
- *Forwarding of monitoring information:* In order to keep track of the Light DC status, the NFVO can request monitoring information to the VIM, covering changed-status events, resource measurements results and usage records. Complementary, the VIM can proactively send that kind of information directly to the NFVO through the southbound interface.
- *SFC instantiation/release:* The deployment of network services implies the definition of Service Function Chains to create forwarding paths among PNFs, SC VNFs and Service VNFs. For that, the NFVO uses its southbound interface to access the SDN Controller, which is controlled through the VIM, for it to configure the appropriate virtual links that make up the SFCs.

4.2.2.2 Communication with VNFM

The VNFM is the entity of the CESCM responsible for the lifecycle management of both SC VNFs and service VNFs. The communication between the NFVO and the VNFM follows the ETSI NFV MANO Or-Vnfm reference point specification [36], which provides the following functionalities:

- *VNF Lifecycle Management:* The NFVO can invoke VNF lifecycle management operations for both SC VNFs and service VNFs. Operations include instantiating VNFs, scaling VNFs (both horizontal scaling to add or remove virtual machines to a VNF to increase or release capacity, and vertical scaling to increase/decrease memory, CPU capacity or storage size to existing virtual machines), terminating VNFs, querying VNFs, healing VNFs and operating VNFs (e.g. to start and stop a VNF instance).
- *VNF Lifecycle Change Notification:* The NFVO can request subscription to receive notifications from the VNFM about changes related to lifecycle management operations of VNF instances. Notifications contain information about the type of the VNF lifecycle change, the addition/deletion of VNF components and the changes on virtualised resources associated to VNF components as result of the VNF lifecycle change. Notifications also inform about the virtual networks and connection points that are added/deleted as part of the VNF lifecycle operation.
- *VNF Performance management:* This allows the VNFM to provide the NFVO with performance management information related to VNF instances. Management information includes measurement results and notifications concerning the VNF instances and the virtualised resources (i.e. virtual machines) used by these instances. Collection and reporting of performance information is controlled by PM jobs. Then, the NFVO can create PM jobs specifying the performance metrics to collect for each VNF, it

can delete or query PM Jobs, subscribe operations or specify threshold levels on certain performance metrics for which notifications will be generated when crossed.

- *VNF Fault management:* The VNFM provides the NFVO with alarms related to the VNFs. Information for each alarm includes the fault type, the severity and the cause of the alarm.
- *VNF configuration management:* This allows the NFVO to provide configuration information for a VNF instance.
- *Notifications about changes in VNF indicators:* The VNFM provides the NFVO with information on value changes of VNF related indicators declared in the VNF descriptor.
- *VNF package management*: The VNFM queries information about the VNF packages available at the NFVO (e.g. information about release date, vendor info, manifest, VNFD, SW image meta-data, files contained in the VNF package, etc.). It can also subscribe for receiving notifications about VNF package changes.
- *VNF lifecycle operation granting:* The VNFM can request a grant for authorization of a VNF lifecycle operation to the NFVO. The NFVO can approve or reject a request based on policies and available capacity.
- *Management of the virtualised resources in indirect mode*: There exist two options for managing the allocation/release of virtualised resources for the VNFs [37]. In direct mode, the management is performed by the VNFM, who directly requests resources to the VIM. Instead, in indirect mode, the management of the virtualised resources is performed by the NFVO. Therefore, whenever the VNFM needs to allocate or release virtualised resources to a VNF, it will send the request to the NFVO that will transfer it to the VIM.

4.2.3 Westbound Interface

This interface has been designed for the NFVO to access the catalogues and repositories from its internal modules, creating an isolation layer that allows the NFVO to communicate with different kinds of database technologies through a single API.

4.2.3.1 Communication with Local Catalogues and Repositories

The NFVO's westbound interface is dedicated to connect with the CESCM Local Catalogues and Repositories. Through it, the VNF Placement, the NFVO coordinator and the SFC monitoring modules of the NFVO can retrieve a list of available VNFs and NSs from the respective catalogues, obtain infrastructure monitoring data, and even select different VNF placement strategies.

The functionality that the Westbound Interface provides is:

- *Insert/retrieve/modify entries in the Local Catalogues and Repositories*: The API exposed for the internal NFVO modules lets them perform the typical operations to the databases without requiring specific knowledge of the technology used for the database implementation.

5 Service Function Chaining (SFC)

This section copes with the concept of NFV-based network services in SESAME, and the VNF chaining schemes.

5.1 The definition of SFC

In order to formally define the SFC concepts in the scope of SESAME, the following concepts are introduced following the ETSI NFV terminology [22], [23]).

SESAME Network Service (NS): a collection of Virtual Network Functions (VNF) that make up a complete service from the perspective of a VSCNO. Thus, a SESAME NS includes all the network and service instances required to deploy a mobile service for the end users of the VSCNO. Examples of SESAME NSs are: mobile connectivity for a certain coverage area, where only network level VNFs would be involved, and; a mobile video service, where edge video processing instances are added to the basic radio connectivity. Therefore, a SESAME NS can be characterized through a series of radio-level and service-level KPIs that are captured in the SLA between the SCNO owning the NFVO and the interested VSCNO. These KPIs may include bare network parameters (e.g., aggregated uplink (UL) / downlink (DL) bitrate in a certain coverage area for a percentage of time) or more complex service-level KPIs (e.g., a number of supported video sessions with edge caching / edge transcoding features).

Every SESAME NS is defined through its associated Network Service Descriptor (NSD). In SESAME, the NSD is deployed by the SCNO through its NFVO, and every VSCNO is provided with at least one NS that includes the VNF instances supporting the different KPIs required by the VSCNO.

The following two concepts are the main building blocks in SESAME NSs.

The **VSCNO Network Connectivity Topology (NCT)** determines the complete list of VNFs and their Connection Points (CP), as well as the possible interconnections through a series of Virtual Links (VL). In this sense, the VSCNO NCT can be seen as the virtual network slice assigned to that VSCNO in the CESC Cluster. Based on this topology, the VSCNO may provide different configurations and services to its users.

A **VNF Forwarding Graph (VNF-FG)** describes the possible paths that data packets can follow within the VSCNO NCT. Generally speaking, a VNF-FG determines the possible data paths for a specific type of service. Depending on the complexity of the considered service, the VNF-FG can be a simple chain of VNFs determining the data path from the entry to the exit point, or a more complex VNF chain with alternative paths that requires decision-making functions within the VNFs.

In the former case, the NFVO needs to propagate specific traffic steering rules to the NFVI. That way, the traffic can be classified at the entry point of the Light DC and all the data packets would follow the same path towards the exit point. In these cases, the VNF-FG needs to include a specific **Network Forwarding Path (NFP)** for each well-known type of traffic. A NFP can be considered as an ordered list of CPs that determines a chain of VNFs from entry to exit.

When the desired data path cannot be a priori determined at the entry point, the forwarding decisions between the VNFs included in the VNF-FG remain in the logic of the VNFs. Therefore, when a VNF-FG does not include NFPs the forwarding path applicable to a traffic flow is decided at runtime.

In a general sense, and depending on the scope of applicability, the concept of SFC is usually associated to the functionality provided by the VNF-FGs or the NFPs.

Following the specifications in SESAME D2.2 [16] and D2.3 [17], the SESAME NFVO needs to handle with two main types of VNFs:

- Small Cell VNFs, which can be considered as "Type A" or "Type B" VNFs according to [22]. This type of VNF is able to forward packets and implement packet header modifications based on distributed decision logic.
- Service VNFs, which can be considered as "Type C" VNFs according to [22]. A "Type C" VNF can receive network packets, even if the IP or MAC address does not point to it, it can also receive packets without MAC or IP encapsulation. Thus, these Service VNFs are interconnected through a single CP and the forwarding behaviour is governed by the NFVO.

5.2 Examples of SFC

Ideally, an NFVO could implement the exact NFP for each type of traffic if forwarding decisions can be made a priori based on centralized control plane and service level information. Since the SESAME architecture does not propose the decoupling of control and data planes for the mobile network protocol stack, the SESAME NFVO needs to cope with distributed forwarding logic.

The scope of the NFVO in SESAME is illustrated in Figure 12, which represents the NCT for a VSCNO served by the SCNO. The functional network and service elements are based on SESAME D2.3 [16] for the case that deploys a HeNB-GW[29]. This example NCT assumes that the vHeNB-GW implements the IPsec client towards a unique SecGW in the VSCNO EPC.

As described before, this NCT graph resembles the network slice that the SCNO creates for the VSCNO. This network slice includes the 4G RAN segment enriched with the inclusion of one or more mobile edge service instances. In the example provided, the NCT includes two CESCs and two different Service VNFs.

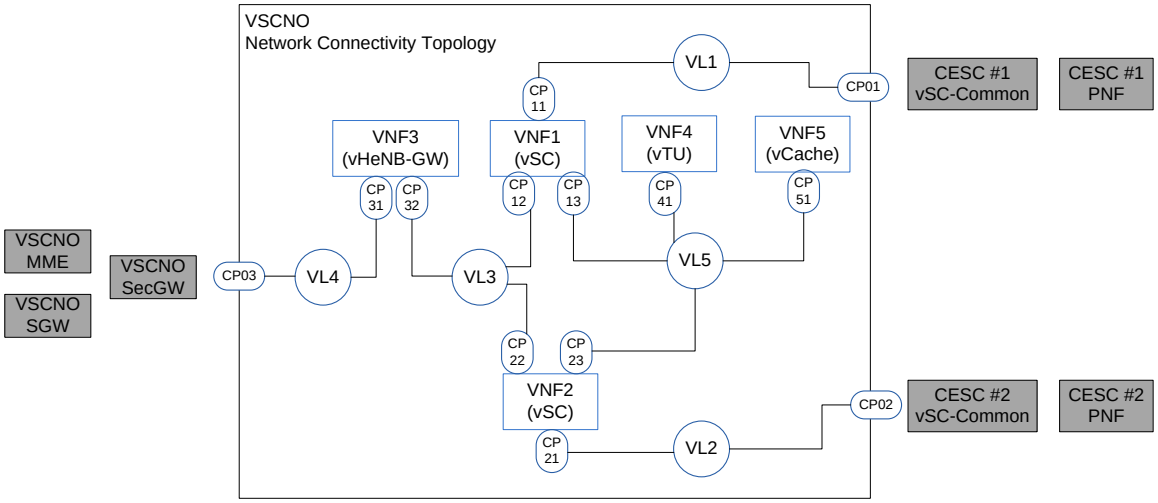

Figure 12: Example Network Connectivity Topology for a VSCNO in SESAME

[29] See: https://en.wikipedia.org/wiki/Home_eNodeB

One of the critical aspects for the correct operation of the network slicing is the slice selection at the entry points. In the external CP interfacing the VSCNO EPC, the underlying infrastructure is able to determine the specific NCT based on the source IP of the VSCNO SecGW. At the interface with each CESC, a specific selection function is proposed to discern between the different VSCNOs according to the subscriber information. Although this slice selection function can be deployed as a common VNF at each CESC, it remains out of the scope of the VSCNO NCT and must be deployed as a standalone entity either by the NFVO or the VIM itself.

Over this underlying logical infrastructure, a series of network services can be deployed for the VSCNO. The following bullets provide a few examples that may constitute SFCs to support different VSCNO data flows.

- VNF-FG for the control plane

One of the key aspects to support the VSCNO mobile services is to create the data paths for the S1-MME interface. In the case illustrated in Figure 12, one S1-MME interface[30] is established between the CESC PNF and the vSC or vHeNB-GW deployed for the VSCNO, and another S1-MME interface is established between the vSC or vHeNB-GW and each MME available in the VSCNO EPC.

Figure 13 illustrates the possible VNF-FG for the C-Plane packets of the VSCNO in the context of SESAME.

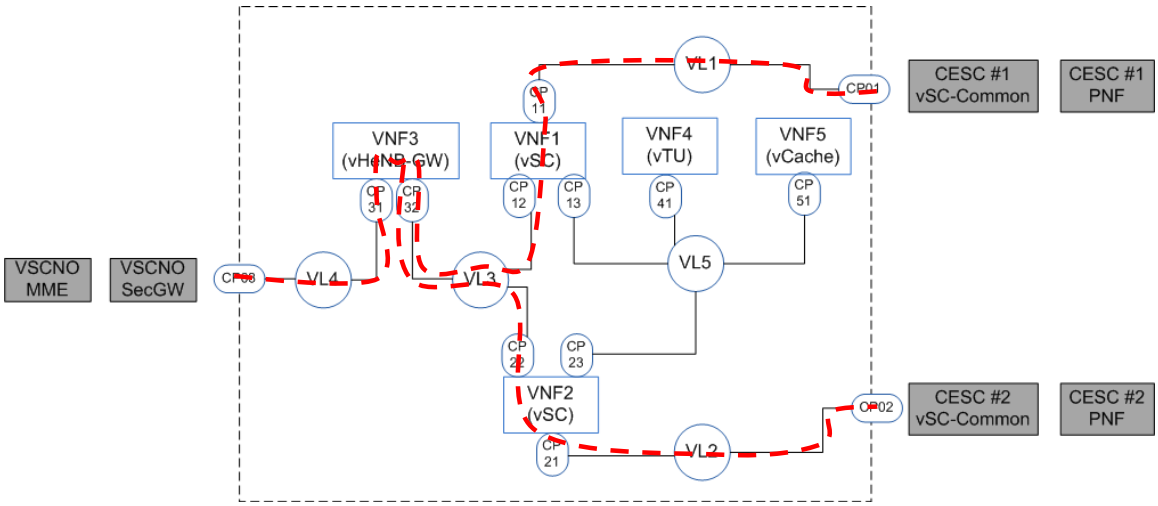

Figure 13: VNF-FG for VSCNO C-Plane

For the C-Plane, as it is detailed in SESAME D3.1 [18], the traffic identification is implemented in the SC Common VNF. Thus, the network selection function is based on the associated PLMN ID field included in the S1AP[31] messages and cannot be identified as a traffic steering policy. Alternatively, the vSC-Common remains as a network selection function which changes the destination IP address to the MME of the specific VSCNO for each S1AP dialog. As a result, each tenant's isolated NS starts after the vSC-Common VNF and the traffic identification can be done based on the protocol type (S1AP for the C-Plane) and the destination IP address.

The possible data paths for the C-Plane for a single CESC are presented in Figure 14. In this case, the UL data path is fixed and could be described by a specific NFP if there is a unique VSCNO

30 https://en.wikipedia.org/wiki/System_Architecture_Evolution
31 http://lteworld.org/specification/s1-application-protocol-s1ap

MME or if the NFVO implements the load balancing between MMEs. In the DL, the vHeNB-GW acts as an IPsec tunnel endpoint, and thus the forwarding decision to the correct CESC must be implemented in the running code.

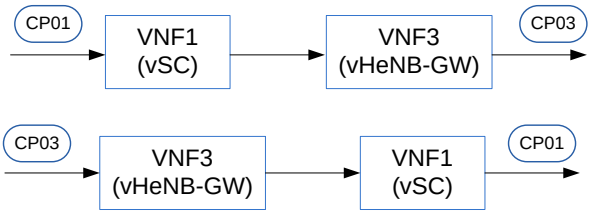

Figure 14: Possible data paths for VSCNO C-Plane (upper figure represents the UL and lower figure represents the DL)

- VNF-FG for the data plane: external services

For the U-Plane traffic, all the user packets are encapsulated at the SC PNF with GTP-U and transported over IP towards the specific VSCNO SGW IP address. Thus, the different user-plane traffic types can be identified at the external connection point based on the protocol type (GTP-U over UDP/IP), the destination IP address (associated to the VSCNO IP address range) and the type of service.

Figure 15 shows the possible data paths for the user data plane of a network service that does not make use of locally deployed service VNFs.

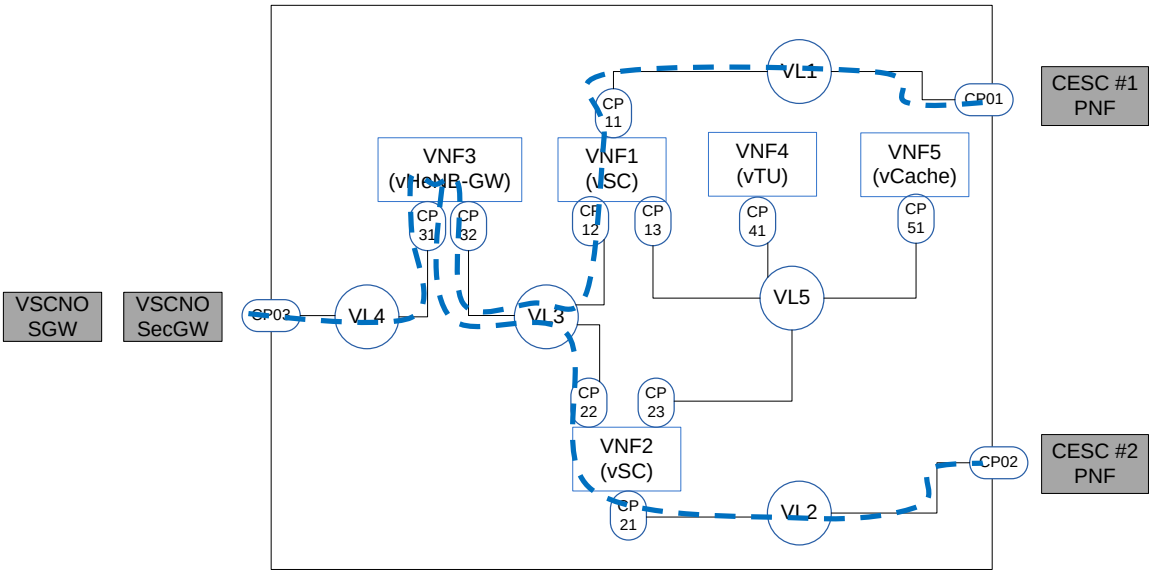

Figure 15: VNF-FG for VSCNO U-Plane: external services without edge services

In this case, all the data connections require the access to external servers through the EPC of the VSCNO. Therefore, the VNF-FG and the possible data paths are similar to the C-plane case, but supporting S1-U interfaces towards the VSCNO SGW.

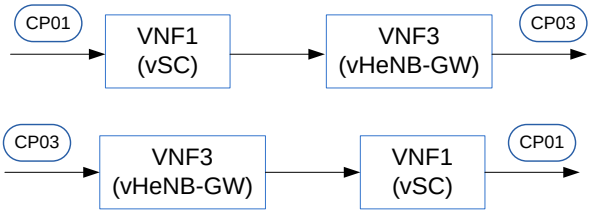

Figure 16: Possible data paths for VSCNO U-Plane without edge services (upper figure represents the UL and lower figure represents the DL)

- VNF-FG for the data plane: external services with edge content modification

This type of service also requires the connection with external servers for the service provision, but the data packets can be derived to a local Service VNF for its further processing. The example provided in Figure 17 represents the case of edge video transcoding, which may modify the encoding characteristics of the multimedia flow before sending it to users (in the DL direction) or to the EPC (in the UL direction).

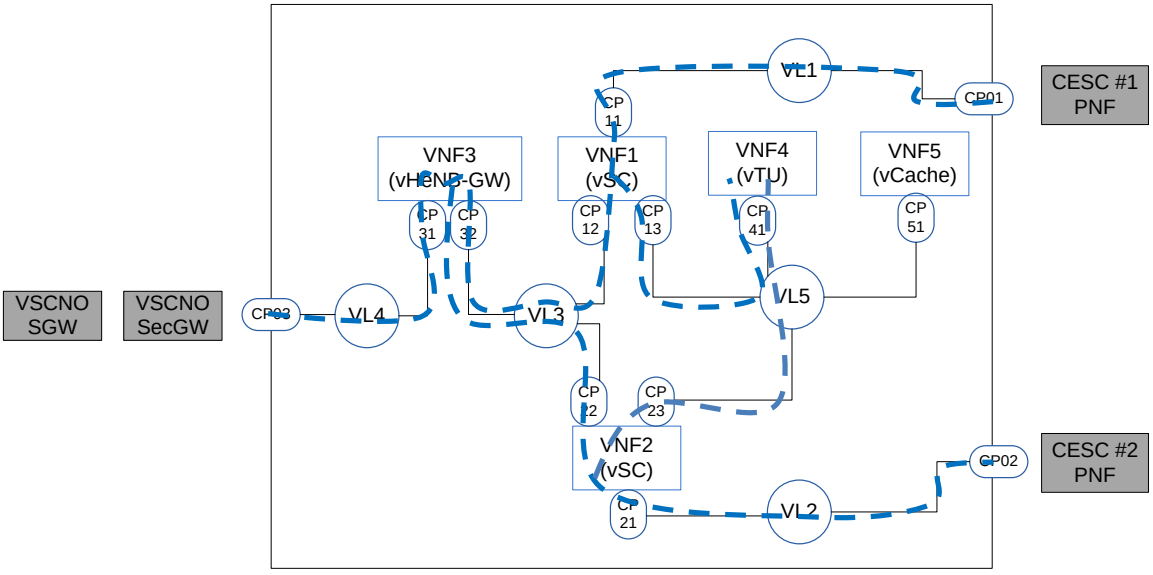

Figure 17: VNF-FG for VSCNO U-Plane: external services with edge content modification

The possible data paths become somehow more complex, with alternative data paths for different types of packets.

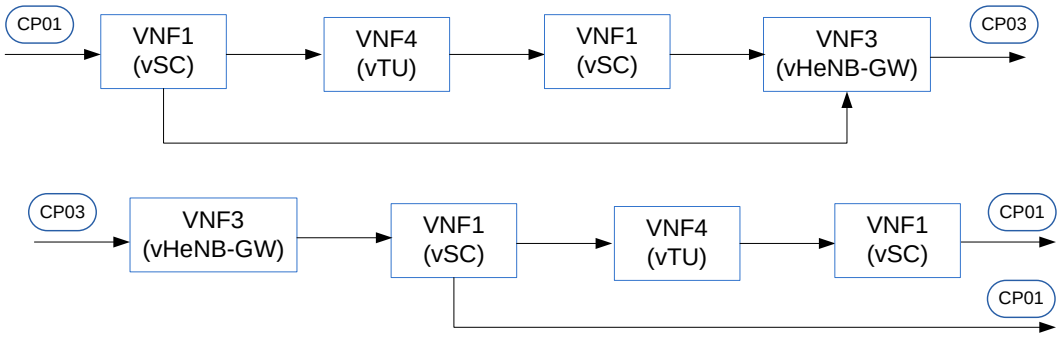

Figure 18: Possible data paths for VSCNO U-Plane with edge content modification (upper figure represents the UL and lower figure represents the DL)

In the UL direction, the vSC needs to inspect the data packet to determine if it has to be forwarded to the vTU. For media requests and other media control packets, the packets can be directly forwarded to the vHeNB-GW and then to the EPC. However, response packets with media content are subject to be forwarded to the vTU VNF before. Forwarding the entire set or a subset of media flows to the vTU, may be determined by specific user-based policies.

In the DL, similarly, the vSC needs to inspect the data packets containing media content and to determine which of them to forward to the vTU, and which directly to the CESC PNF.

- VNF-FG for the data plane: external services with edge transparent caching

This type of service generally requires the connection with external servers for the service provision, but the data packets can be derived to a local Service VNF acting as a transparent cache of contents. Therefore, if the content of a data request is already cached at the edge VNF, the data request will not be forwarded to the EPC.

The example provided in Figure 19 represents the case of edge content caching that operates in the DL direction.

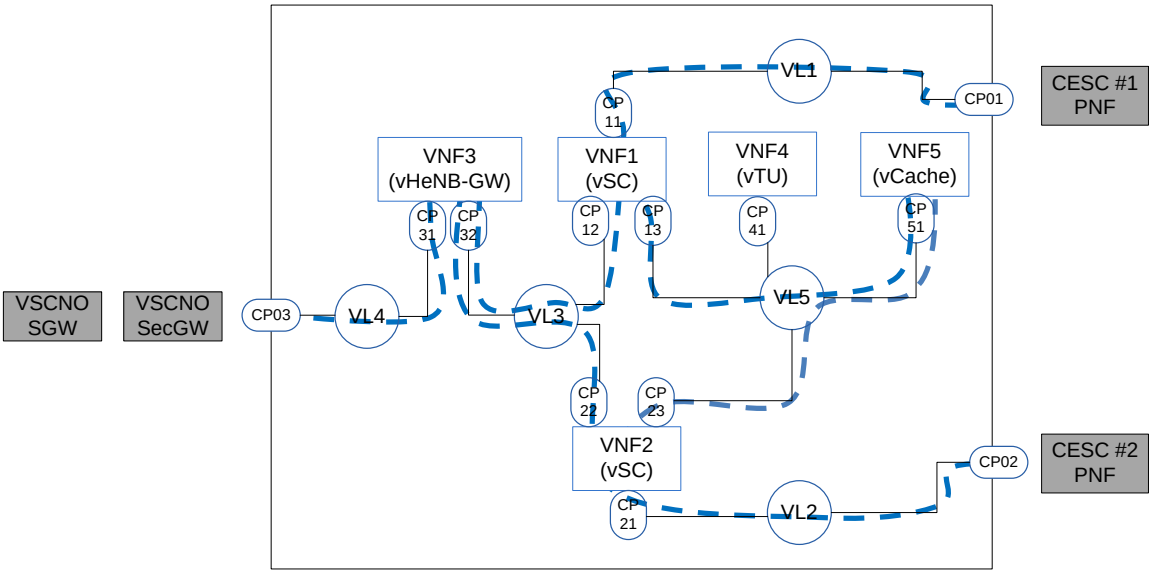

Figure 19: VNF-FG for VSCNO U-Plane: external services with edge transparent caching

As illustrated in Figure 20, the vSC VNF must discern whether data packets coming in the UL from the mobile user are content requests. In that case, those requests are forwarded to the vCache VNF. If the content is already cache, the vSC will forward the content response back to the mobile user. Otherwise, the content request is forwarded to the external server through the VSCNO SGW. In the DL direction, the vSC only needs to ensure that content responses are forwarded to the vCache VNF for its possible edge caching.

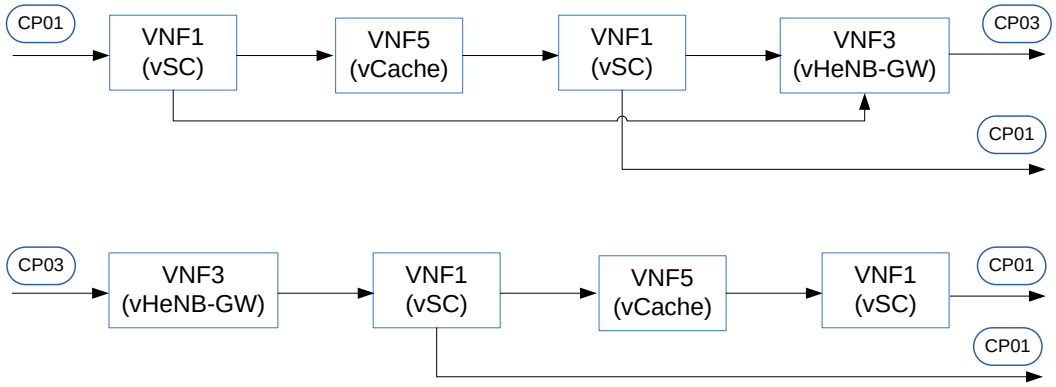

Figure 20: Possible data paths for VSCNO U-Plane with edge transparent caching (upper figure represents the UL and lower figure represents the DL)

- VNF-FG for the data plane: local services

The last example network service involves the deployment of local services, e.g. enterprise services, which do not require traversing the VSCNO EPC for their provisioning. A possible case is exemplified in Figure 21 with the deployment of a local cache of contents and preventing the access to external servers. In this dummy case, the contents are either locally available or not available at all.

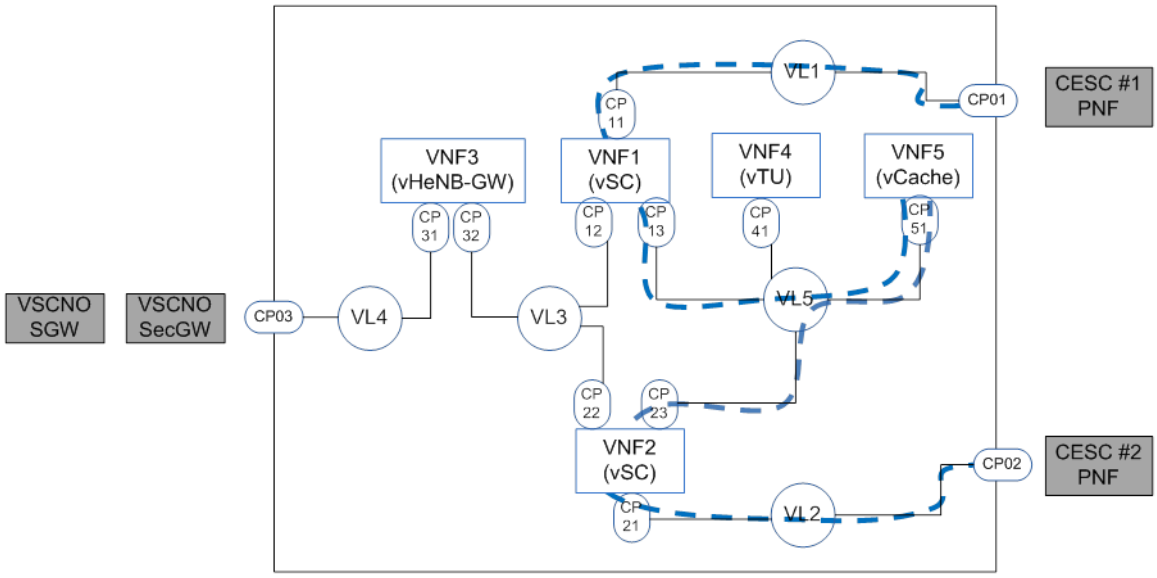

Figure 21: VNF-FG for VSCNO U-Plane: edge local services

In this case, since there is no possible entry from the EPC, the only data path possible involves the interaction between the vSC and the vCache. As a result, this service type can be deployed by the NFVO as a pure SFC with a well-determined NFP in the VNF-FG.

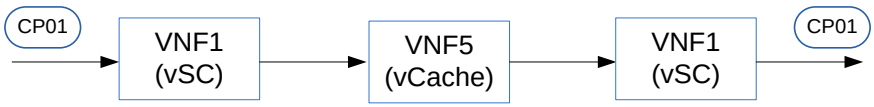

Figure 22: Possible data paths for VSCNO U-Plane with edge local services

This series of type of services are intended to exemplify the possible chains of VNF that the SESAME NFVO can find. Beyond these SFCs, any combination of Service VNFs leading to more complex NFPs is possible within the scope of SESAME. Additionally, all the example cases are

based on the assumption of the use of a HeNB-GW for the VSCNO. The alternative deployment without HeNB-GW would require the inclusion of the IPsec client in the vSC instances and different types of SFCs.

6 Workflows for SFC

In this section, the workflows for SFC creation and deployment, SFC termination, and SFC modification will be described to serve as high-level guidelines for tasks T6.2 and T6.3. The workflows described here explicitly show the interactions among the internal functional modules of the NFVO and external modules, which is an extension of the workflows described in ETSI MANO [14].

6.1 SFC Creation and Deployment

SESAME provides edge cloud services that can be defined as a chain of VNFs and/or PNFs, i.e. an SFC. Therefore, when a user (e.g. VSCNO) sends a service creation request to the NFVO, it will trigger the SFC creation and deployment process. The corresponding workflow of the SFC Creation and Deployment is defined in Figure 23 and described in this subsection.

After the NFVO receives the service creation request, it will first check whether the service already exists, that is, the NSD in the NS catalogue exists and the corresponding VNF Packages are already part of the NS. If not, the NSD will be on-boarded together with the required VNFs, that is, the VNFDs will be uploaded to the VNF catalogue and VNF images will be uploaded to the applicable VIMs. SCNO will be responsible for the on-boarding of the required VNFs.

Once the NS and all the required VNFs are on-boarded, the NFVO will start instantiating the requested SFC. The SFC Lifecycle Manager will be in charge of the SFC instantiation. First, the affinity rules will be applied to each VNF in the VNFFG to determine the location requirements of those VNFs, e.g. two VNFs will have to be located in different HWs in order to guarantee the availability or redundancy, or two VNFCs will need to be located in the same HW in order to guarantee low latency. Considering these location constraints, the VNF placement module will be called for the SFC mapping, that is, indicating where to implement the VNFs (according to the selected VNF placement algorithm and the current status of the resource usage) and also the connection points between the VNFs in the SFC. SESAME focuses on a unique SFC mapping scenario, i.e. the coordinated mapping of both the radio/SC VNFs and the Service VNFs. Each of the VNFs may have different requirements (e.g. latency) for the actual deployment locations. More details can be found in *Section* 4.1.3 on the VNF placement module.

The SFC mapping could fail due to various reasons, such as lack of available resources or a failure of fulfilling the low latency requirements of certain VNFs. If the SFC mapping is successful, the actual SFC deployment will start by interacting with the VIM and the VNFM. In this case they will be in charge to instantiate the VMs to host the VNFs and manage their lifecycle. This has already been covered by the ETSI MANO specifications [14]. The catalogues will be updated with the new available resources status and VNF instances.

Once the service is created and deployed, the SFC Lifecycle manager triggers the SFC monitoring module to invoke service performance monitoring and periodic feedback report.

6.2 SFC Modification

We have defined three types of triggers in SESAME for SFC modification.

- Type 1: The tenant (e.g. VSCNO) triggers the SFC modification.

- Type 2: The SLA monitoring triggers the SFC modification, which reflects the feedback from the radio side i.e. SC EMS. This is unique in SESAME, which shows the coordination between the radio network part and the cloud part.
- Type 3: The SFC Monitoring as internal module of NFVO triggers the SFC modification.

Once SFC modification is triggered, the NSD of the related service together with the relevant VNFDs will be either updated if they already exist (scale up/down) or on-boarded (scale in/out). After successful update or on-board of the new NSD and VNFs, the NFVO coordinator will call the SFC Lifecycle manager to start the SFC modification process. The SFC Scaling module of the NFVO will be in charge of the SFC modification.

The NFVO Coordinator will first query the Local catalogue (i.e. NFVI repository) for the current available NFVI resources as abstracted by the VIM, and then call the VNF Placement algorithm to perform the mapping of the modified SFC (that is, the new deployment flavours of the VNFs and/or the new connectivity between the VNFs). Here again, we will need to consider the unique resource allocation requirements in SESAME, as described in above subsection 6.1. If the mapping is successful, the new SFC will be deployed (the VNFs will be scaled in/out, or scaled up/down) in the NFVI by interacting with VIM and VNFM as specified in ETSI MANO [14]. The catalogues will be updated with the new available resources status and VNF instances. The SFC monitoring module will be called to start monitoring the new service.

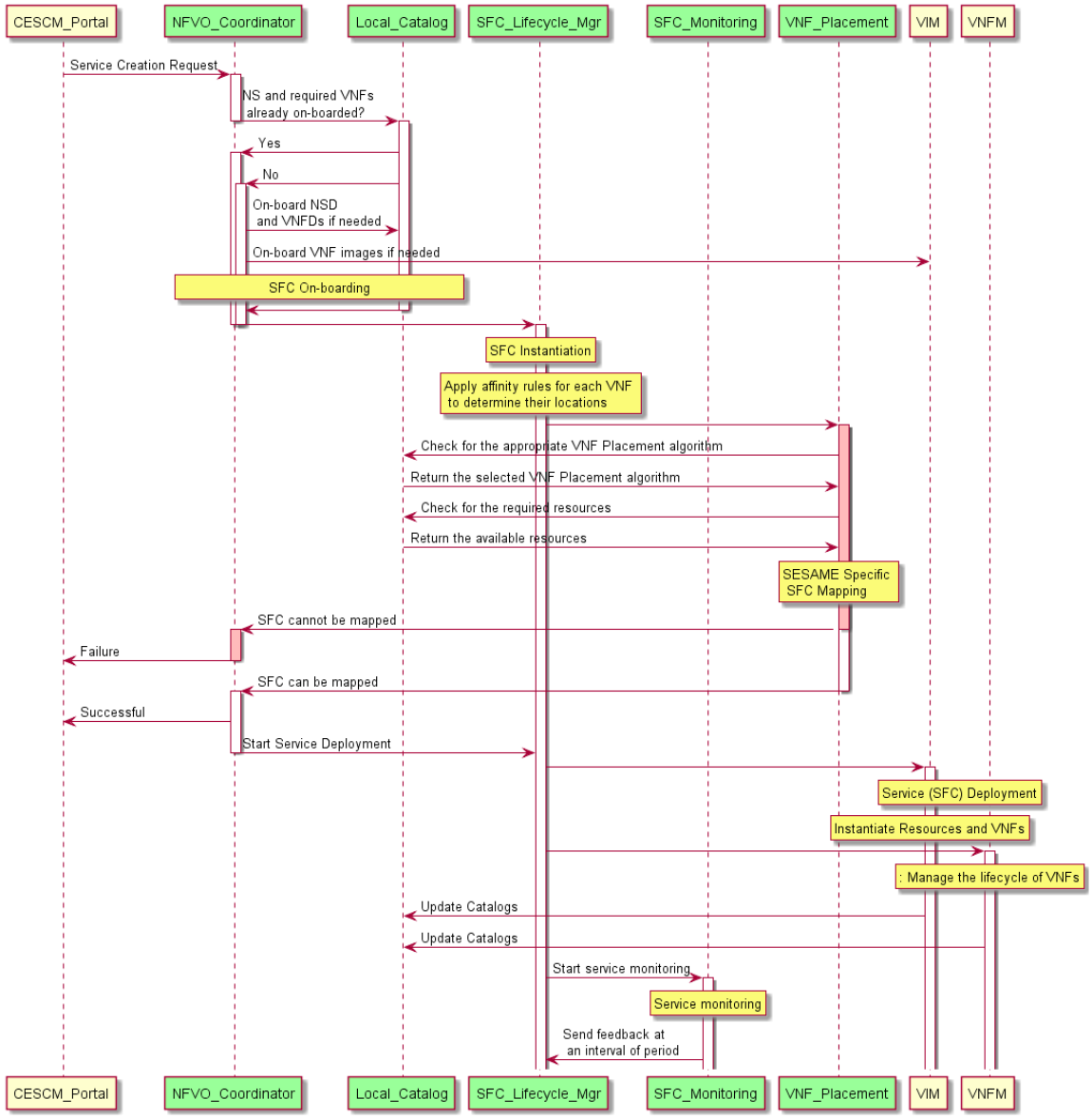

Figure 23: The SFC creation and deployment workflow

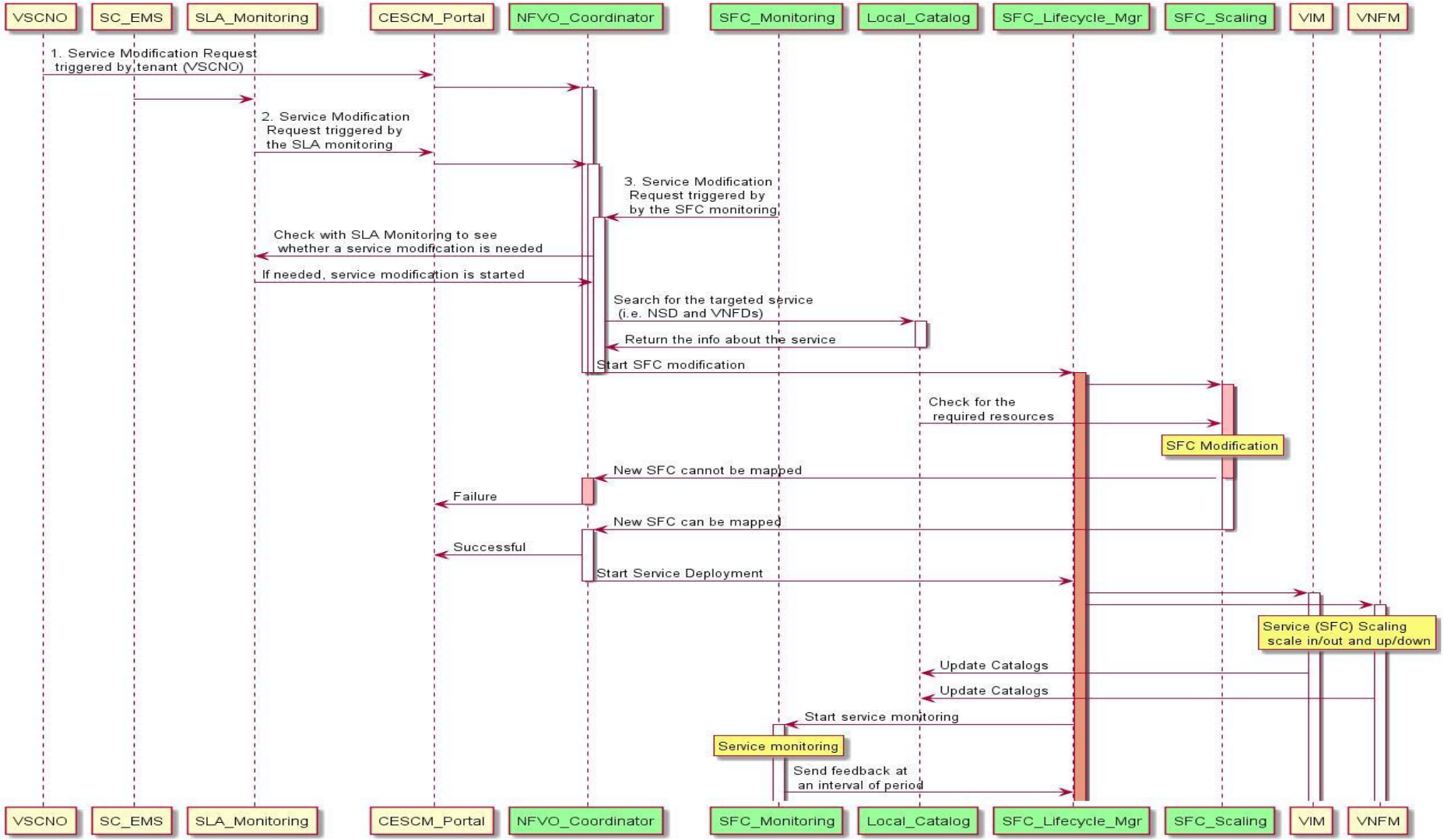

Figure 24: The SFC modification workflow

6.3 SFC Termination

When the user (e.g. VSCNO) sends a request to terminate the service, the request will first reach the NFVO Coordinator, and then the NFVO Coordinator will query the Local catalogue for the NSD and VNFDs of the service. Then the SFC Lifecycle Manager will be in charge of the SFC termination by interacting with VIM and VNFM to disable the VNFs and release hosting resources (VMs and network connectivity) as described in ETSI MANO [14].

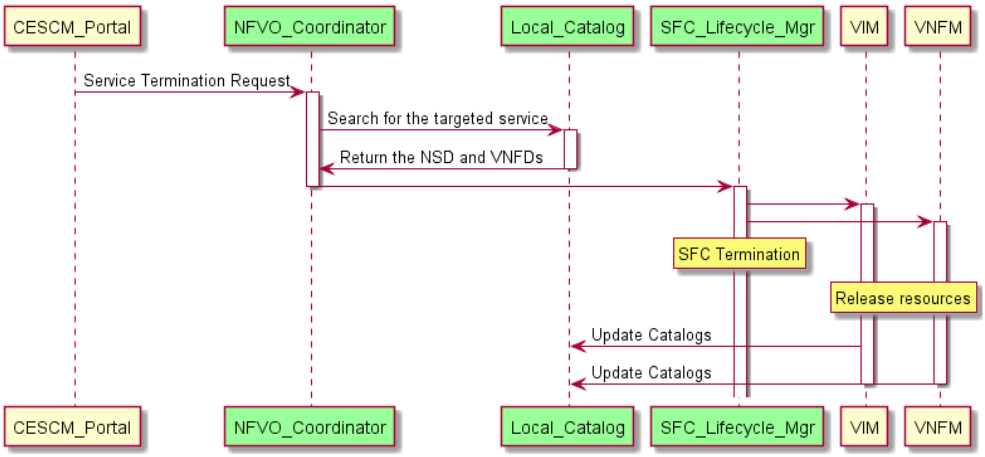

Figure 25: The SFC termination workflow

7 NFVO Implementation Guideline

Based on what has been presented so far, it is inferred that the main novelty of SESAME NFVO is not on the internal architecture and the way that services are deployed. The entire proposed model is ETSI MANO compliant [38], and in this sense no one wants to be off track. However, the point that makes SESAME NFVO different from the other solutions is the fact that it is one of those orchestrators (if it is not the only one) that simultaneously considers both requirements of radio and cloud domains. In theory, the orchestrator is agnostic of the underlying hardware, i.e. despite the VNF origin, NFVO only deals with descriptors and metadata. Of course, this also holds true for the SESAME NFVO. However, the dynamic nature of radio traffic and its influence on the overall system performance (e.g. latency) forces NFVO to take special measures for the service deployment over the Light DC to meet the agreed SLAs. To better understand this issue we need to acknowledge few points:

- Unlike the traditional cloud environments with a uniform infrastructure to work over, one point of presence (PoP) in SESAME is a distributed environment (cluster of CESCs) with limited resources per device.
- The hardware resources available on the CESCs are not identical, e.g. few CESCs in one cluster may contain hardware accelerators (e.g. FPGA, DSP, etc.). This makes the SESAME cloud environment (Light DC) heterogeneous. The placement procedures will be studied with more details in D6.2. Depending on the particular needs in SESAME, one or another placement solution will be adopted.
- To provide edge services, a SC VNF is essential per tenant (VSCNO) and per CESC (to break/create GTP tunnels).
- Like in any radio access network, the user traffic is dynamic and time varying, since the number and the location of users change over time.

Under these circumstances, the placement and the migration of services (composed of SC VNFs and service VNFs) is a very crucial responsibility that SESAME NFVO needs to address. Traditionally, NFVO architectures based on the ETSI MANO reference model adapt a two stage service mapping approach: 1- on NFVO level, a selection between different PoP (VIM) is done, and 2- on the VIM level, e.g. in OpenStack, the VNF deployment is left to the VIM placement process (e.g. Nova scheduler and based on a random selection). In SESAME though, this two-stage approach does not seem enough to meet all the 5G needs.

For instance, in SESAME, even to support a simple connectivity for a VSCNO in one PoP, SC VNFs need to be deployed over the light DC on per tenant basis. For the NS that includes: deploying service VNFs over hardware accelerators (HWA), accurate selection of deployment place considering simultaneously hardware requirement, radio coverage and agreed measures on SLA (e.g. latency). It calls for more advanced placement process better than a radio agnostic procedure running only at the VIM level. The problem becomes more complicated if handover is taken also into account. Even for NS without service VNFs on HWA, handover forces NFVO to greatly influence the service placement and migration. For instance, assume an example where a network service, aimed to serve a group of people (consisting of SC VNF and service VNFs), resides completely on a CESC. As time passes and the group moves out of the CESC coverage, a handover to another CESC happens. From the NS perspective it means that SC VNF of the second CESC carries out the GTP tunnel break down/creation role, whereas the rest of service VNFs are still hosted on the first CESC. Although, it is still possible to have an edge service, the extra delay caused by CESC to CESC communication might affect the overall system performance

and lead to an SLA violation (e.g. increase of the latency). The NFVO should react to radio triggered alarms and respond by a hitless service migration (e.g. make before break). Although the idea of service migration itself is not new, customizing it according to the 5G needs and radio requirements is a good contribution that SESAME NFVO can add up to the orchestration world.

This section aims to draw a guideline for the implementation of NFVO to be followed by the next tasks of WP6, T6.2 and T6.3. To do so, the section is divided into two sub-sections, the first part indicates the possible implementation approach for the NFVO-SDN communication in the context of SESAME, and the second subsection shows an implementation perspective for the SESAME NFVO based on a modular design. The divide-and-conquer strategy defined by the modular implementation facilitates the challenging task of the NFVO development. Moreover, it guarantees many benefits such as: distributed development, code reusability, program readability and better manageability.

7.1 Implementation approach for the NFVO – SDN communication

Based on the discussion above, it is possible to highlight an implementation plan for an effective service function chaining mechanism in the context of SESAME. Before moving to more details, we need to point out that SESAME has not yet envisioned a converged/uniform radio-cloud control plane to provide a pragmatic solution at this stage (PoC). This means that radio related control activities (e.g. handover) are done separately and in a way close to today's 4G approach. Considering the lack of communication between the radio and VIM controllers, SESAME proposes to use a common instance(s) of service VNF in order to serve all hits into the CESC cluster. It means that depending on the SLA that determines both radio (e.g. latency) and cloud (e.g. number of hits to caching) requirements, first the number of required service VNFs is determined, and then the appropriate place (i.e. CESC) to deploy them is chosen. Meanwhile, the rules of the SDN controller are updated to support the routing of data from all CESCs into the one hosting the service VNF (e.g. based on the type of service (ToS) field on IP header). For instance if we assume a VSCNO request to serve <1000 caching hits, NFVO calculates that one vCaching VNF is needed and to meet the required latency performance it should be instantiated on e.g. CESC #3 of the cluster. Similarly, to serve > 2000 caching hits, two vCaching VNFs are necessary and, considering the latency requirements, they are placed on CESC #3 and CESC #10 of the cluster. The proposed solution may seem static at the first place; however, compared to the challenging task of creating a uniform radio-cloud controller, it is a pragmatic way to address the issues stated above. Below is presented a brief description of the proposal through an example.

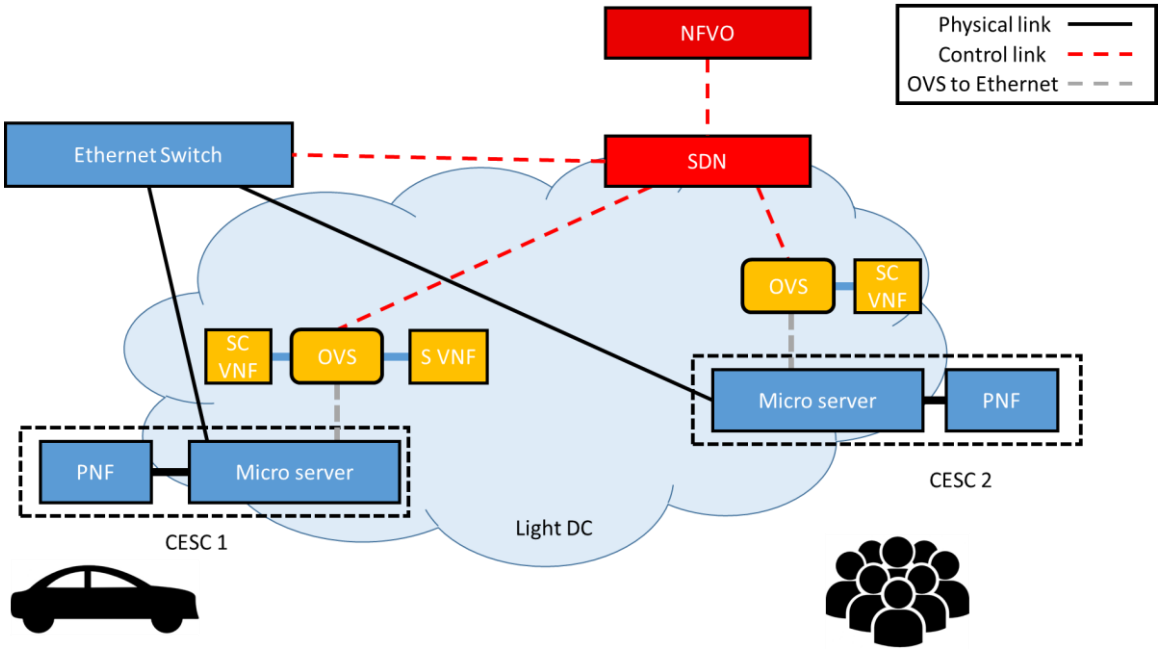

Figure 26: SFC in the context of SESAME

Figure 26 shows a physical – virtualised view of the SESAME environment from a single VSCNO perspective. As depicted, the existence of SC VNF at each micro server is essential to break/create the GTP tunnels. However, service VNFs can be hosted in any micro server, as long as the appropriate connectivity between CESCs is enforced by SDN rules and the SLA requirements are met. For example, consider a case where the VSCNO demands a mobile edge service, say caching. The relevant service VNF (vCaching) is instantiated at one CESC in the cluster (e.g. CESC #1). Meanwhile, NFVO dictates the relevant rules to the SDN controller and with this, it provides access from all CESCs to the service VNF. In case of caching demand elsewhere in the CESC cluster, requests are directed to the CESC #1 to be served from there.

The example above shows in practise the support of edge mobile services. To make the proposed solution flexible enough to respond to SLA violations, we considered two situations. The first case happens when it is not possible to serve all demands with a single common service VNF. In this situation, another instance is deployed (as explained above) in another CESC (e.g. CESC #2) to keep the quality of service. Load balancing between these two instances is subject to the NFVO load balancing policy. Besides that, there might be a situation where only one common VNF is enough to serve the demands, but its placement clashes with another SLA metrics (e.g. latency). In this case, a service VNF migration is applied in a hitless fashion (i.e. make before break) to ensure the SLA requirement.

7.2 Implementation Steps for NFVO

Section 4 provided a high level logical vision of the SESAME NFVO architecture. This subsection tries to map those requirements into a practical implementation. In the NFVO development, similarly as other software development tasks, the design phase is the first crucial step. The importance of this decision becomes clearer if heavy and diverse responsibilities of NFVO defined in [14] are considered. The first step on the design phase is to select a programming paradigm that allows a sustainable development of complex software packages [39]. The modular programming seems like a natural choice for this purpose.

The main idea behind the modular programming is to break down the whole task into separate/easier sub-tasks or modules. Modules are designed in a way to minimize dependencies between one to another. Such a logical separation improves the maintainability and increases the development speed with the possibility of launching parallel efforts. Teams can focus on modules separately without being concerned about the whole system. Any change in one module is translated to a simple modification in relevant subroutines on other modules. To deliver the full functionality of the software package (written in languages such as python and ruby), modules are incorporated through widely used interfaces such as Representational State Transfer Application Programming Interfaces (REST APIs). REST APIs use HTTP requests to send data (create and/or update), read data (e.g., make queries), and delete data. Thus, REST uses HTTP for all four CRUD (Create/Read/Update/Delete) operations between modules. This way of development is well in-line with the open source mentality (encouraged by the European Commission), where different components (i.e. modules) of a software package come with a version number associated and the development happens on per module bases. Considering the benefits of modular programming, SESAME has selected it as the development paradigm for NFVO.

The next step of the design phase is to break down the high level SESAME NFVO logical components, presented in *Section* 4, into implementation modules (as the selected programming paradigm suggests). Table 1 summarizes the process functionalities for the NFVO and relevant CESCM elements:

Logical Component	Functionality	Proposed Break Down (Modules)
NFVO coordinator	• To act as the main entry of NFVO and direct the data flow to internal NFVO elements • To validate NSDs and VNFDs • To translate NSD, VNFD templates one another (e.g. TOSCA to ETSI) if needed • To support data exchange with NS and VNF repositories • To generate YML files and send them to the VIM –HOT template in case of OpenStack as VIM	• *NS coordinator:* The main and only NFVO entrance which is able to direct entries to the right internal module. In general, there are two types of inputs to *NS coordinator*: i) NSD generated on the CESCM portal based on the NS requested by a VSCNO, ii) VNFD entered by the SCNO to add VNFs to the system. Note that the *NS coordinator* is involved in the work flow for service instantiations. • *NS catalogue:* This module is responsible for inserting/retrieving NSDs to/from NS repository. • *NSD validator:* This module is responsible for validating the incoming NSDs with the NSD NFVO internal scheme. In case of error, it will translate the descriptor and/or stop the action. • *VNF coordinator:* This module manages data exchange on the VNF level inside the NFVO and to the VNFM. This means that even VNFD inputs received on the *NS coordinator* are forwarded directly to the *VNF coordinator*. Note that *VNF coordinator* is also involved on the work flow for the service instantiations. • *VNF catalogue:* This module is responsible for inserting/retrieving VNFDs to/from VNF repository. • *VNFD validator:* This module is responsible for validating the incoming VNFDs with the VNFD NFVO internal scheme. In case of error, it will translate the descriptor and/or stop the action. • *HOT Generator:* This module is responsible for generating HOT templates to facilitate NFVO-VIM communication. • *Descriptor translator:* This module translates different descriptor templates to one another (e.g. TOSCA to ETSI)

SFC lifecycle manager	• To act as the services lifecycle manager (SLM). SLM is a set of processes designed to help administrators oversee the implementation, delivery, operation and maintenance of the services over the course of their existence.	• *NS provisioning:* This module executes the required processes to carry out SLM responsibilities. Also, through an interaction with *NS instance repository*, it keeps track of the course of services. Note that *NS provisioning* plays an important role in service scaling.
SFC monitoring	• To keep track of usage, emergency occasions, performance at end-to-end service level. • To generate service level log files. This log file we be the accumulation of individual VNF logs used in a service. • To record service log files.	• *NS monitoring:* This module is responsible for aggregating VNF log files and generating service level log file out of them. Also this module reviews service level log files for failures and sends out alarm. • *NS monitoring repository:* This module is responsible for inserting/retrieving NS instance report to/from *NS monitoring repository*.
Local catalogues	• To organize collection of data (e.g. VNFD, NSD) in databases. In general, a data base might be SQL or NoSQL depends on the implementation decision.	• *NS repository:* This repository collects/holds NSD. • *VNF repository:* This repository collects/holds VNFD. • *NS instance repository:* This repository collects/holds information of running NS instances (e.g. deployment locations). In this way, all relevant information of all running NS is accessible in this repository. • *VNF instance repository:* This repository collects/holds information of running VNF instances (e.g. deployment locations). In this way, all relevant information of all running NS is accessible in this repository. • *NS monitoring repository:* This repository collects/holds monitoring information of running NSs. • *VNF monitoring repository:* This repository collects/holds monitoring information of running VNFs. • *Infrastructure repository:* This repository collects/holds the hardware resource information. • *Algorithm repository:* This repository contains different placement algorithms. Depending on the optimization objective, different algorithms can be selected from this repository.

VNF placement	• To determine the exact placement of VNFs over the distributed (a mesh of CESCs) and heterogeneous (resources on CESCs are not identical, e.g. some are only enriched with hardware accelerators) environment of Light DC.	• *Service Mapping:* This module is responsible for executing retrieved placement algorithm from the *Algorithm Repository*. Selection of algorithms might be based on different optimization objectives such as maximum resource utilization, minimum energy consumption, etc.
VNFM	• To manage the VNF lifecycle. That includes oversee implementation, delivery, operation, and maintenance of VNFs over the course of its existence. • To keep track of usage, emergency occasions, performance at VNF level. • To generate VNF log files and record them.	• *VNF provisioning:* This module executes the required processes to carry out VNF lifecycle management. Also, through an *interaction* with *VNF instance repository*, it keeps track of the course of services. Note that *VNF provisioning* plays an important role in service scaling. • *VNF monitoring:* This module is responsible for generating VNF log files. Also this module reviews VNF log files for failures and sends out alarm. • *VNF monitoring repository:* This module is responsible for inserting/retrieving VNF instance report to/from *VNF monitoring repository*.
SFC scaling	• To perform scaling of NS, either scale up/down or scale in/out.	• General speaking, scaling is a process resulted by an interaction between monitoring and provisioning. In the same way, scaling in SESAME is triggered by monitoring modules (*NS monitoring*, *VNF monitoring* and *SLA monitoring*) or tenant, and receives reaction by provisioning modules (*NS provisioning* and *VNF provisioning*).

Table 1: The SESAME NFVO functional break down

Proposed functional break down presented on Table 1 leads to a modular orchestrator architecture able to perform all forecasted SESAME NFVO functionalities. Following the high level interdependencies detailed in *Section 4*, Figure 27 below illustrates the modular NFVO architecture for SESAME. Darker boxes show the internal NFVO elements while the others highlight external elements connected to NFVO with respect to Figure 11. From a pragmatic implementation perspective, each module presents a piece of code able to perform a specific functionality and data exchange with the other of modules. For example, considering an implementation where the entire system is installed on a single host, one may see each module as a server at one port able to execute a process and communicate with others via REST API. As a side note, the modular implementation allows having some of the modules up and running at a remote machine while maintaining the overall NFVO functionality intact.

To better understand how the system works, two exemplary interactions are reviewed next:

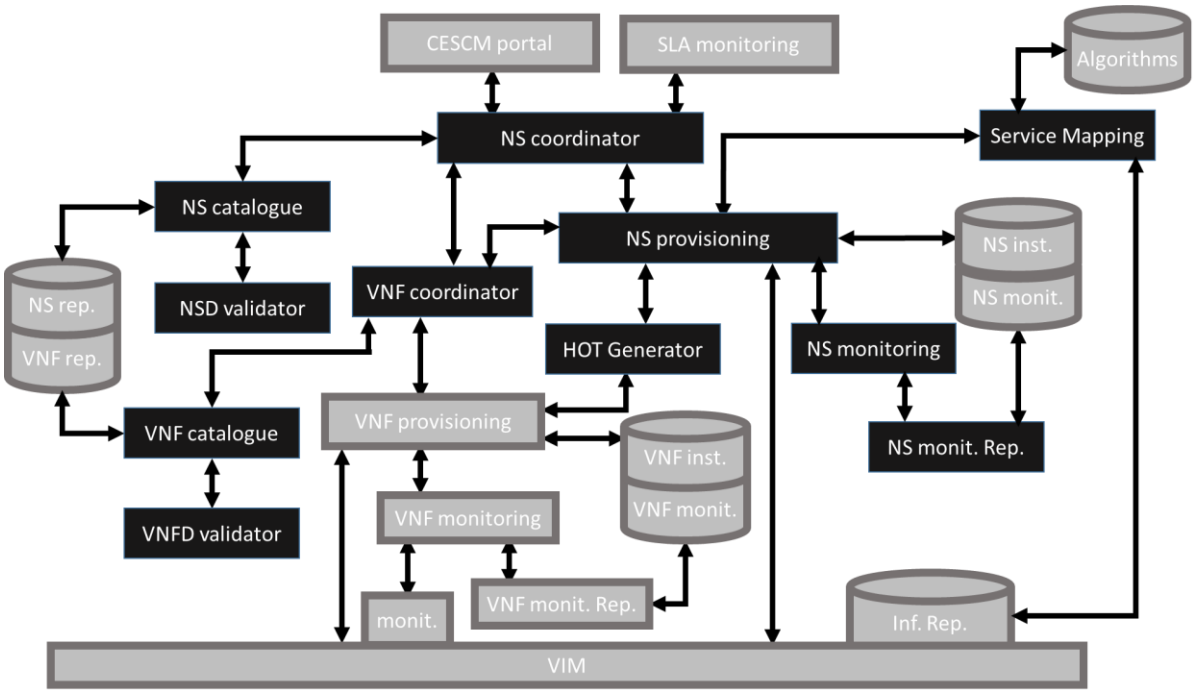

Figure 27: Modular architecture of SESAME NFVO, black elements are internal and grey elements are external with respect to the logical NFVO design.

- *On-boarding a new VNF:* From the NFVO perspective, on-boarding a new VNF is the same as inserting a new VNFD into the *VNF repository*. In the first step, the introduced VNFD by SCNO on the CESCM portal is handed over to the *NS coordinator.* Since the input is at VNF level, it is passed to the *VNF coordinator* in the next step. By looking at the request type (on-boarding a new VNF), *VNF coordinator* forwards the VNFD to the *VNF catalogue* module, which is responsible to add the VNFD into the *VNF repository* after a validation check. *VNFD validator* reviews the VNFD with respect to the VNFD scheme and in case of compliance, resumes the process. The last step is an interaction between *VNF catalogue* module and the *VNF repository* to on-board a new VNFD.
- *Instantiating a NS:* This item is divided into three sub-steps:
    1. **On-boarding a NS and receive a NS ID**: The idea is to keep a record of NS requested by VSCNO to offer it as a suggestion for future requests. In the first step, the introduced NSD by VSCNO on the CESCM portal is handed over to the *NS coordinator*. By looking at the request type (on-boarding a NS), *NS coordinator* forwards the NSD to the *NS catalogue* module, which is responsible to add the NSD into the NS repository after a validation check. The *NSD validator* reviews if the descriptor complies with the NSD scheme and in affirmative case resumes the process. The last step is an interaction between the *NS catalogue* module and the *NS repository* to on-board the NS and to assign it an ID for future access.
    2. **NS network creation**: The next step on the NS instantiation is to create the network that connects virtual machines (VNFs). To do this, *NS coordinator* forwards a copy of the requested NS to the *NS provisioning* module. With the help of *service mapping* module which has an overall vision of the infrastructure (via the infrastructure repository), *NS provisioning* understands the exact location of VNFs and their connections on Light DC (in the ideal case, the entire NS is hosted on a CESC but, in practice, there might be cases with cross CESC chains). In the next step,

the *HOT generator* module generates a HOT template of NS network and then via the *NS provisioning,* the result is handed over to the VIM for deployment. Meanwhile, a copy of NSD is sent by the *NS provisioning* to the *NS instance repository* to keep track of the running NS instances. Moreover, *NS provisioning* informs *NS monitoring* module to start generating NS log files. The generated log files by the *NS monitoring* are added to the *NS monitoring repository* by the *NS monitoring repository* module. With all these interactions, first the NS network is instantiated and also *NS monitoring* is triggered to monitor the NS performance.

3. **Place and run VNFs**: Once the NS network is instantiated, the VNFs get assigned interfaces from that network. To do so, *NS provisioning* sends a copy of NSD to VNF manager. Through an interaction with the *VNF repository* via the VNF catalogues, *VNF coordinator* retrieves the listed VNFD on the NSD, and passes them to the VNFM. Next, the *VNF provisioning* module asks for the HOT template of VNFDs from the *HOT generator* module and after receiving those hands over the templates to the VIM for deployment. Same as above, *VNF provisioning* sends a copy of instantiated VNFs to the *VNF instance repository* to keep track of running VNFs. Moreover, *VNF provisioning* informs *VNF monitoring* module to start generating VNF log files. The generated log files by the *VNF monitoring* are added to the *VNF monitoring repository* by the *VNF monitoring repository* module. After having VNFs up and running as explained the service deployment is done.

The presented implementation guideline is an effort to pave the way for a better SESAME NFVO deployment on T6.2 and T6.3. According to the general development plan, a GitHub[32] will be dedicated for this purpose. The idea is to release the NFVO components along with the other CESCM elements publicly. The exact time plan for such an open source software activity is subject to the SESAME general assembly decision.

[32] https://github.com/github

8 Conclusions

This document presented the high-level logical architecture of the SESAME NFVO, specifying the internal functional modules and the interactions/interfaces among these modules.

First of all, the document analysed the state-of-the-art of the current existing NFVOs, especially focusing on analysing their features and the relevance to the SESAME NFVO. Moreover, the NFVO role related to the SESAME CESCM is further clarified in this document.

The network service (NS) concept in SESAME is defined together with other terminology clarification, concepts such as NCT, NFP and VNF-FG. A few SFC examples are given to show a clear view of the service chaining in SESAME over both U-plane and C-plane and including both UL and DL. The workflow diagrams of network service creation and deployment, modification, and termination are drawn and described. The description of the NS and SFC in this document will further serve T6.2 for the actual implementation of the SFC between SC VNF and service VNFs.

Finally, a functional breakdown of the SESAME NFVO logical elements and a preliminary modular architecture is given in order to serve guidance for the implementation to be done in T6.3.

9 References

[1] Ioannis Giannoulakis, et *al*., "Enabling Technologies and Benefits of Multi-Tenant Multi-Service 5G Small Cells", EuCNC 2016.

[2] https://blog.zhaw.ch/icclab/category/research-approach/themes/cloud-orchestration/

[3] "Mobile Cloud Networking: Hurtle, Cyclops, Gatekeeper," ICCLAB, Available at: https://blog.zhaw.ch/icclab/files/2014/07/mcn_hurtle_cyclops_tmb.pdf

[4] "icclab/hurtle", Available at: http://bit.ly/20lZLB4

[5] "Hurtle Technical Architecture Description," Available at: https://github.com/icclab/hurtle/blob/master/docs/hurtle_technical_implementation.md

[6] "Hurtle Conceptual Architecture," Available at: https://github.com/icclab/hurtle/blob/master/docs/architecture.md

[7] "FP7 Mobile Cloud Networking," Available at: http://www.mobile-cloud-networking.eu/

[8] "5G-PPP SONATA," Available at: https://5g-ppp.eu/sonata/

[9] "FP7 T-NOVA project," Available at: http://www.sonata-nfv.eu/

[10] "FP7 UNIFY, " Available at: https://www.fp7-unify.eu/

[11] "Tacker," Available at: https://wiki.openstack.org/wiki/Tacker

[12] "OpenMANO," Available at http://bit.ly/1NOz5fC

[13] "Open Baton," Available at: http://Open Baton.github.io/

[14] "ETSI NFV MANO v1.1.1," Dec. 2014, Available at: http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf

[15] "Cloudify," Available at: http://getcloudify.org/

[16] H2020 ICT 671596 SESAME, deliverable D2.2, "Overall System Architecture and Interfaces", March 2016.

[17] H2020 ICT 671596 SESAME, deliverable D2.3, "Specification of the CESC components – First Iteration", March 2016.

[18] H2020 ICT 671596 SESAME, deliverable D3.1, "CESC Prototype design specifications and initial studies on Self-X and virtualization aspects", June 2016.

[19] http://www.etsi.org/technologies-clusters/technologies/nfv/open-source-mano

[20] "Open Source MANO," Available at: https://osm.etsi.org/

[21] "End-to-End Service Instantiation Using Open-Source Management and Orchestration Components," Intel Whitepaper, Mobile World Congress 2016.

[22] ETSI GS NFV-SWA 001 (V1.1.1) "Network Function Virtualisation (NFV); Virtual Network Functions Architecture", December, 2014.

[23] P. Quinn and T. Nadeau, "Service Function Chaining Problem Statement," IETF Secretariat, Internet-Draft, RFC 7498, April 2015.

[24] R. Guerzoni, "Network Functions Virtualization: An Introduction, Benefits, Enablers, Challenges and Call for Action. Introductory Whitepaper," SDN and OpenFlow World Congress, 2012.

[25] W. Xu, Y. Jiang and C. Zhou, "Problem Statement of Network Functions Virtualisation Model," IETF Secretariat, Internet-Draft draft-xjz-nfv-model-problem-statement-00, Sept. 2013.

[26] W. Liu, H. Li, O. Huang, M. Boucadair, N. Leumann, Z. Cao, Q. Sun, C. Pham, C. Huang, J. Zhu and P. He, "Service Function Chaining (SFC) Use Cases," IETF Secretariat, Internet-Draft draft-liu-sfc-use-cases-05, March 2015.

[27] M. Boucadair, C. Jacquenet, R. Parker, D. Lopez, J. Guichard and C. Pignataro, "Service Function Chaining: Framework and Architecture," IETF Secretariat, Internet-Draft draft-boucadair-sfc-framework-02, Feb. 2014.

[28] S. Mehraghdam, M. Keller and H. Karl, "Specifying and Placing Chains of Virtual Network Functions," IEEE Third Int'l Conference on Cloud Networking (CloudNet) 2014.

[29] P. C. Abhishek Gupta, M. Farhan Habib, "Joint Virtual Network Function Placement and Routing of Traffic in Operator Networks," CS Department, University of California Davis, Tech. Rep., April 2015.

[30] A. Mohammadkhan, S. Ghapani, G. Liu, W. Zhang, K. K. Ramakrishnan an T. Wood, "Virtual network Function Placement and Traffic Steering in Flexible and Dynamic Software Defined Networks," IEEE Int'l Workshop on Local and Metropolitan Area Networks (LANMAN), April 2015.

[31] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating Virtual Network Functions in NFV," CoRR, vol. abs/1503.06377. Available: http://arxiv.org/abs/1503.06377.

[32] R. Riggio, F. De Pellegrini and D. Siracusa, "The Price of Virtualisation: performance Isolation in Multi-Tenants Networks," IEEE Network Operations and management Symposium (NOMS), May 2014.

[33] R. Riggio, A. Bradai, T. Rasheed, J. Schulz–Zander, S. Kuklinski, T. Ahmed, "Virtual Network Functions Orchestration in Wireless Networks," in Proc. of IEEE CNSM 2015, Barcelona, Spain.

[34] R. Riggio, A. Bradai, D. Harutyunyan, T. Rasheed; T. Ahmed, "Scheduling Wireless Virtual Networks Functions," in IEEE Transactions on Network and Service Management, vol.PP, no.99, pp.1-1.

[35] Draft ETSI GS NFV-IFA 013 v0.7.0, "Network Functions Virtualisation (NFV); Management and Orchestration; Os-Ma-nfvo reference point - Interface and Information Model Specification", Work in progress document, March, 2016

[36] Draft ETSI GS NFV-IFA 007 v0.7.0, "Network Functions Virtualisation (NFV); Management and Orchestration; Or-Vnfm reference point - Interface and Information Model Specification", Work in progress document, March, 2016

[37] R. Morera et *al.*, "ETSI NFV IFA WG - Architecture and Interfaces", https://docbox.etsi.org/isg/nfv/open/Other/NFV12-Tutorials-20151026/20151026-NFV_Tutos-3-MANO%20Overview%20IFA%20status_151009b_Layer123_NFV-IFA_Tutorial_ME_PW.pdf

[38] ETSI GS NFV 004 (V1.1.1) "Network Function Virtualisation (NFV); Virtualisation Requirements", October, 2013.

[39] M. Gabbrielli, S. Martini, "Programming Languages: Principles and Paradigms", Springer, 2010.