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 D6.4

Orchestrator Prototype

Document Number:	H2020-5GPPP-GA No.671596/*WP6/D6.4/30.09.2017*
Contractual Date of Delivery:	*30.09.2017*
Editor:	Pouria Sayyad Khodashenas – i2CAT Foundation (i2CAT)
Work-package:	*WP6*
Distribution / Type:	Public (PU) / Report (R) / Prototype (PO)
Version:	1.0
Total Number of Pages:	69
File:	*SESAME_Deliverable 6.4_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 (NSs) are supported by Virtual Network Functions (VNFs) 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 over the CESC cluster. 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 describes the latest NFVO related activities done in the context of the SESAME WP6. In particular, Task 6.3 ("*Service Management, Monitoring and Execution*"). That includes some results extracted from the conducted investigations as well as a technical documentation of the NFVO prototype developments.

The outcome of D6.4 will feed WP7 on the demo-delivery related activities.

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	02.08.2017	ToC and initial inputs all over the document	Pouria Sayyad Khodashenas - i2CAT
0.2	02.08.2017	i2cat inputs on section 1, 2 and 3	Pouria Sayyad Khodashenas - i2CAT
0.3	08.08.2017	ToC update, inputs on section 1 and 2	Pouria Sayyad Khodashenas - i2CAT
0.4	14.08.2017	Added 3.1 Section for Monitoring tools and solution in SESAME	Irena Trajkovska - ZHAW
0.5	22.08.2017	Included section 4.2 SLA monitoring High-level workflow. Updated section 4.3 Endpoint	Elisa Jimeno Gallardo - Atos
0.6	30.08.2017	All contributions have been integrated.	Pouria Sayyad Khodashenas - i2CAT
0.7	03.09.2017	Inputs on section 6	Leonardo Goratti - CNET
0.8	05.09.2017	Inputs on section 4	Jordi Perez-Romero - UPC
0.85	12.09.2017	Inputs on section 7, First complete draft ready for review	Pouria Sayyad Khodashenas - i2CAT
0.9	25.09.2017	Review 1	Cristina Costa - CNET Elisa Jimeno Gallardo - Atos
0.95	27.09.2017	Review 2	Athanassios Dardamanis - SMNET Ioannis Chochliouros - OTE
1.0	30.09.2017	Final release	Pouria Sayyad Khodashenas - i2CAT

Contributors

First Name	Last Name	Partner	Email
Pouria	Sayyad Khodashenas	i2CAT	pouria.khodashenas@i2cat.net
Shuaib	Siddiqui	i2CAT	shuaib.siddiqui@i2cat.net
Athanasios	Chalas	i2CAT	athanasios.chalas@i2cat.net
Alfonso	Egio Artal	i2CAT	alfonso.egio@i2cat.net
Irena	Trajkovska	ZHAW	traj@zhaw.ch
Jordi	Pérez-Romero	UPC	jorperez@tsc.upc.edu
Oriol	Sallent	UPC	sallent@tsc.upc.edu
Ramon	Ferrús	UPC	ferrus@tsc.upc.edu
Ramon	Agustí	UPC	ramon@tsc.upc.edu
Elisa	Jimeno Gallardo	Atos	elisa.jimeno@atos.net
Leonardo	Goratti	CNET	lgoratti@fbk.eu
Shah	Nawaz Khan	CNET	s.khan@fbk.eu
Tejas	Subramanya	CNET	t.subramanya@fbk.eu
Leonardo	Goratti	CNET	lgoratti@fbk.eu
Roberto	Riggio	CNET	rriggio@fbk.eu
Athanassios	Dardamanis	SMNET	adardamanis@smart.net.gr
Ioannis	Chochliouros	OTE	ichochliouros@oteresearch.gr

Glossary

Acronym	Explanation
3GPP	Third Generation Partnership Project
4G	Fourth Generation of Mobile Communications
5G	Fifth Generation of Mobile Communications
AC	Admission Control
ACM	Association to Computer Machinery
API	Application Programming Interface
ASN.1	Abstract Syntax Notation One
AWS	Amazon Web Service
CAPEX	Capital Expenditure
CCO	Coverage and Capacity Optimisation
CESC	Cloud-enabled Small Cell
CESCM	Cloud Enabled Small Cell Manager
CO	Central Office
CPE	Customer Premises Equipment
CPU	Central Processing Unit
CQI	Channel Quality Indicator
C-RAN	Cloud-Enabled RAN
cSON	centralised SON
DB	Database
DC	Data Centre
DL	Downlink
DPI	Deep Packet Inspection
dSON	decentralised SON
DSP	Digital Signal Processor
DSS	Decision Support System
E2E	End-to-End
EMS	Element Management System
EPC	Evolved Packet Core
ETSI	European Telecommunications Standards Institute
EU	European Union
FCAPS	Fault, Configuration, Accounting, Performance and Security
FE	Functional Entity
FG	Forwarding Graph
FM	Fault Management
FMC	Fixed and Mobile Convergence
FP	Function Provider
FPGA	Field Programmable Gate Array
FT	File Transfer
FW	Firewall
GA	Grant Agreement
GPRS	General Packet Radio Service
GPU	Graphics Processing Unit
GS	Group Specification
GTP	GPRS Tunnelling Protocol
GUI	Graphical User Interface
GW	Gateway
H2020	Horizon 2020
HDD	Hard Disk Drive
HeNB	Home eNodeB
HOT	Heat OpenStack Templates

HTTP	Hyper-text Transmission Protocol
HW	Hardware
HWA	Hardware Accelerator
I/O	Input/Output
IaaS	Infrastructure-as-a-Service
ICT	Information and Communication Technology
ID, id	Identifier
IDP	Intrusion Detection Prevention
IP	Internet Protocol
IRP	Integration Reference Point
IS	Information Service
IT	Information Technology
JSON	JavaScript Object Notation
KDE	Kernel Density Estimation
KPI	Key Performance Indicator
Light DC	Light Data Centre
LTE	Long Term Evolution
MAC	Medium Access Control
MANO	Management and Orchestration
MEC	Mobile Edge Computing
MES	Mobile Edge Service
NAT	Network Address Translator
NF	Network Function
nFAPI	network Functional Application Platform Interface
NFP	Network Forwarding Path
NFV	Network Functions Virtualisation
NFVI	Network Functions Virtualisation Infrastructure
NFVO	NFV Orchestrator
NIS	Network Information Service
NMS	Network Management System
NRM	Network Resource Model
NS	Network Service
NSD	NS Descriptor
NSNM	Network Service Notification Manager
OAI	Open Air Interface
OPC	OpenDayLight
OPEX	Operational Expenditure
OPNFV	Open Platform for NFV
OS	Operating System
OTT	Over-the-Top
OVS	Open virtual Switch
PaaS	Platform-as-a-Service
PHY	Physical Layer
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
PRB	Physical Resource Block
PS	Packet Scheduling
PGW	Packet Gateway
QoE	Quality of Experience

QoS	Quality of Service
RAB	Radio Access Bearer
RAM	Random Access Memory
RAN	Radio Access Network
RAT	Radio Access Technology
REST	Representational State Transfer
RF	Radio-Frequency
RIA	Research and Innovation Action
RNIS	Radio Network Information Service
RPC	Remote Procedure Call
RRC	Radio Resource Control
RRM	Radio Resource Management
RSRP	Reference Signal Received Power
RSRQ	Reference Signal Received Quality
RSSI	Received Signal Strength Indicator
RTT	Round-Trip Time
SaaS	Software-as-a-Service
SC	Small Cell
SCaaS	Small Cells-as-a-Service
SCF	Small Cell Forum
SCNO	Small Cell Network Operator
SDC	Smart Data Centre
SDN	Software-defined Networking
SDR	Software Defined Radio
secGW	Security Gateway
SFC	Service Function Chaining
SW	Serving Gateway
SLA	Service Level Agreement
SOM	Self-Organizing Map
SON	Self-Organizing Network
SP	Service Provider
SQL	Structured Query Language
SVM	Support Vector Machine
SVR	Support Vector Regression
SW	Software
TM	Traffic Monitor
TOSCA	Topology and Orchestration Specification for Cloud Applications
TR	Technical Report
TS	Technical Specification
TSDR	Time Series Data Repository
TTI	Transmission Time Interval
TU	Transcoding Unit
UC	Use Case
UE	User Equipment
UL	Uplink
URL	Uniform Resource Locator
USRP	Universal Software Radio Peripheral
UTRA	Universal Terrestrial Radio Access
UUID	Universally Unique Identifier
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
VNF	Virtual Network Function
VNFD	VNF Descriptor
VNFFG	VNF Forwarding Graph
VNFFGD	VNF Forwarding Graph Descriptor
VNO	Virtual Network Operator
VOC	Video Optimization Controller
VoIP	Voice over IP
vPGW	virtual Packet Gateway
VSCNO	Virtual Small Cell Network Operator
vTU	virtual Transcoding Unit
WOC	WAN Optimization Controller
WP	Work Package
WS	Web Service
XML	Extensible Markup Language

Table of Contents

List of Figures

List of Tables

1 Introduction

From an operator's perspective, the evolution of mobile networks implies a continuous attempt to find a trade-off between capacity increase (i.e. higher bitrates, more coverage, etc.) and total cost of ownership (i.e. CAPEX, OPEX). A viable solution points to the concept of flexible Radio Access Network (RAN) with simplified deployment and management. Although today's RAN is able to provide a high level of configurability (e.g. support for a variety of transport network and baseband configurations), there is still plenty of room to exploit the potential synergies with concepts like Network Function Virtualization (NFV) and Software Defined Networking (SDN). Joint radio-cloud architecture is an effort to place intelligence at the network edge and to use virtualisation technologies to build a cost-, spectrum-, and energy-efficient RAN able to offer improved user experience [1].

From an infrastructure perspective, there are various options to form a joint radio-cloud system, depending on the level of centralisation and distribution of IT assets within the RAN environment. A distributed Cloud-Enabled RAN (C-RAN) architecture has been proposed by the H2020 5GPPP SESAME project [2]. Designing a multi-tenant C-RAN represents an effort of evolving commercial Small Cells (SC) towards a so-called Cloud Enabled Small Cell (CESC). A CESC is an enhanced SC that integrates a virtualised execution platform (micro-server) equipped with IT resources (RAM, CPU, storage) and Hardware Accelerators (HWA), such as Graphics Processing Units (GPU), Digital Signal Processors (DSP), and/or Field-Programmable Gate Arrays (FPGA). With these capacities, a CESC is able to support novel applications and services at the network's edge. However, individual CESC resources might be insufficient to offer significant added value. Therefore, SESAME proposes CESC clustering. This approach leads to the creation of a distributed data centre, denominated Light Data Centre (Light DC), aimed to enhance the virtualisation capabilities and processing power at the network's edge. Of course, the technology used to tie up CESCs in the form of a Light DC may vary from wired to wireless solutions depending on the use case. In any case, the employed clustering technology should be able to preserve QoS and "meet" 5G performance requirements such as latency and bandwidth.

Migrating part of SC radio functionality to the Light DC in the form of Virtual Network Functions (VNF), permits to link the radio data transmission to the cloud world. That is the lifecycle events of SC VNFs can be managed similarly to service VNFs, such as virtual deep packet inspection (vDPI), virtual caching (vCaching) and/or virtual transcoding unit (vTU). Network Services (NSs) formed by SC VNF and service VNFs will offer the added-value Mobile Edge Services (MES) to the subscribers of the tenant (SC operator). Different possibilities for the composition of such services are discussed in [3] and [4].

Multiple independent instances of MES can operate over the shared Light DC environment. With the help of authentication and authorisation management (logical isolation), it is possible to serve more than one tenant over a single C-RAN infrastructure. By definition, this case is interpreted as multi-tenancy in the context of 5G.

The multi-tenant mixed radio-cloud environment poses extra challenges for service management and orchestration. To address these challenges within the context of T6.3, the involved WP6 partners have worked on the design and development of SESAME NFVO solution. The initial report on this activity has been presented in D6.1 [4]. This document presents the latest NFVO related activates and well as a technical report on the final SESAME orchestrator prototype.

1.1 Deliverable Outline

The present deliverable covers the most recent NFVO-*related* activities done in the context of Task 6.3 of the SESAME project, which includes the following sections:

- *In Section 2 we investigate the benefits of joint radio – NFV resource orchestration in the 5G era. In particular, we study the resource allocation in a realistic NFV-enabled 5G access network, taking into account the dynamics of ~100,000 persons' movement in a crowded event, i.e. a football match. The proposed solution jointly orchestrates NFV and bandwidth resources, as one resource affects the other. Simulation results clearly verify the benefits of the proposed solution over traditional disjoint schemes.*
- *Despite the potential benefits, joint radio-cloud systems pose technical challenges on the network management and orchestration, especially upon ensuring the Quality of Service (QoS). To this end, having a complete loop of monitoring, decision-making and reaction is essential. However, considering the fact that the radio and the cloud parameters are inherently disparate, forming such a loop in a joint radio-cloud environment is very challenging. This challenge becomes more difficult in multi-tenant (operator) scenarios, targeted by 5G, where ensuring the QoS for one tenant should not violate the QoS of the others. Section 3 intends to "state" the problem from the SESAME perspective, and discusses a possible solution within the context of the project.*
- *One of the main aspects of the QoS assurance is accurate and effective decision-making processes. Section 4 covers this point by discussing the SESAME cognitive decision-making processes.*
- *Section 5 provides an overview of the NFVO prototype and integration guideline.*
- *As a new technology, lightweight virtualized network functions have attracted much attention. Section 6 reviewers this technology in the context of SESAME projects and provides some study results.*
- *Finally, Section 7 summarises the key issues discussed in the document and concludes the deliverable.*

2 Benefits of Joint Radio and NFV Resource Orchestration

We conducted a study that verifies the benefits of joint radio and NFV resource orchestration. In an end-to-end (E2E) service offering scenario, applications forming an added value service can be placed at different networking domains: e.g. the network edge, backhaul or in a remote data center far from the end-user. The decision on where to place them depends on the evaluation of different factors, such as the expected performance, the required resources to run properly, the maximum number of concurrent hits, etc. With the help of resource orchestration techniques [5], it is possible to coordinate the interaction among each piece and deliver the desired added value service.

There are several studies available in the literature focusing on the placement and application coordination problem from a static perspective ([6], [7], [8]). In particular, [6] modeled the resource allocation problem with a mixed-integer program and proposed a solution to incorporate limited physical resources into the NFV resource allocation. Authors in [7] studied the resource allocation problem, taking into account the scalability issue which might be caused by hosting multiple VNFs over the same hardware. Last but not least, in [8] authors in a static scenario studied the impact of latency requirements on the placement of VNFs constituting an added value service.

Despite the importance of the previous studies, given the targeted 5G use cases (UCs), there is no surprise to see that there is a great need to investigate dynamic user behaviour. For instance, 5G-PPP suggests use cases where 5G capabilities are offered in sports stadiums or shopping malls. In these use cases, the traffic profile changes drastically over time as crowded events occur sporadically. This definitely demands an effort to propose solutions able to cope with the traffic changes, on the fly. Actually, this topic has been covered separately at some extent in the cloud and the radio world. NFV management related actions are listed under solutions for scaling up/down resources, application migration from one point of presence to another, etc. On the network management side, allocating more/less frequency bands and/or wavelengths, alternative path allocation, etc., are typical solutions.

Here we will show how a holistic and joint decision-making process is able to significantly improve the network performance compared to a case where only NFV or network related actions are employed. To this end, we selected a real life two tier network topology made of so called “micro-nodes” – close to the end-user – and “macro-nodes” – far from the end-user. Then with the help of simulations results generated by our Net2Plan [9], [10]) tool, we will prove that the proposed joint NFV-network management solution will significantly outperform the disjoint alternatives.

2.1 Selected Scenario and Assumptions

Our simulation scenario involves an event, i.e. football match, during which traffic and performance requirements increase drastically due to the large amount of people (~100,000) present in a relatively small area.

We consider an area of 1-5 km radius around the Camp Nou stadium in Barcelona with a real radio access network topology [11] populated with two types of nodes: the traditional macro-cells around the map, and 200 micro-cells distributed in the stadium premises (i.e. the seating area) with a coverage equally distributed for all spectators. Each macro-cell is directly connected to a Central Office (CO) via a 10 Gbps optical link, while all micro-cells inside the stadium are connected via a 1 Gbps optical link to a central gateway, which is in turn connected to the CO with a 10 Gbps optical link.

Distribution of the macro-cell antennas is depicted in Figure 1. We assume all links are bidirectional and macro-cells, micro-cells and CO are NFV-*enabled* with the capacity to instantiate VNFs and compose the added value services. CO also hosts the Evolved Packet Core (EPC) and management logic.

Simulations are performed modelling 99,354 individual persons (stadium capacity) originating from specific source points around the edge of the scenario, represented by metro/bus stations, parking lots, etc. Each person travels at a random walking speed to the stadium, and once everyone is inside, the football match begins. For simplicity, we assume the match duration as 5 minutes, and during this time, all people remain seated and stationary inside the stadium. Once the match has ended, each person travels back to its original source point, and the simulation ends when everyone has reached its destination.

Throughout all the simulation time frame, at each time slot, every person has a probability to request a service (binomial distribution based on the total offered traffic, number of persons and service types). The different services types are Web Service, Video Conference and VoIP. Each service has some minimum bandwidth requirements and chained VNFs associated ([8], [12]), as seen in Table 1.

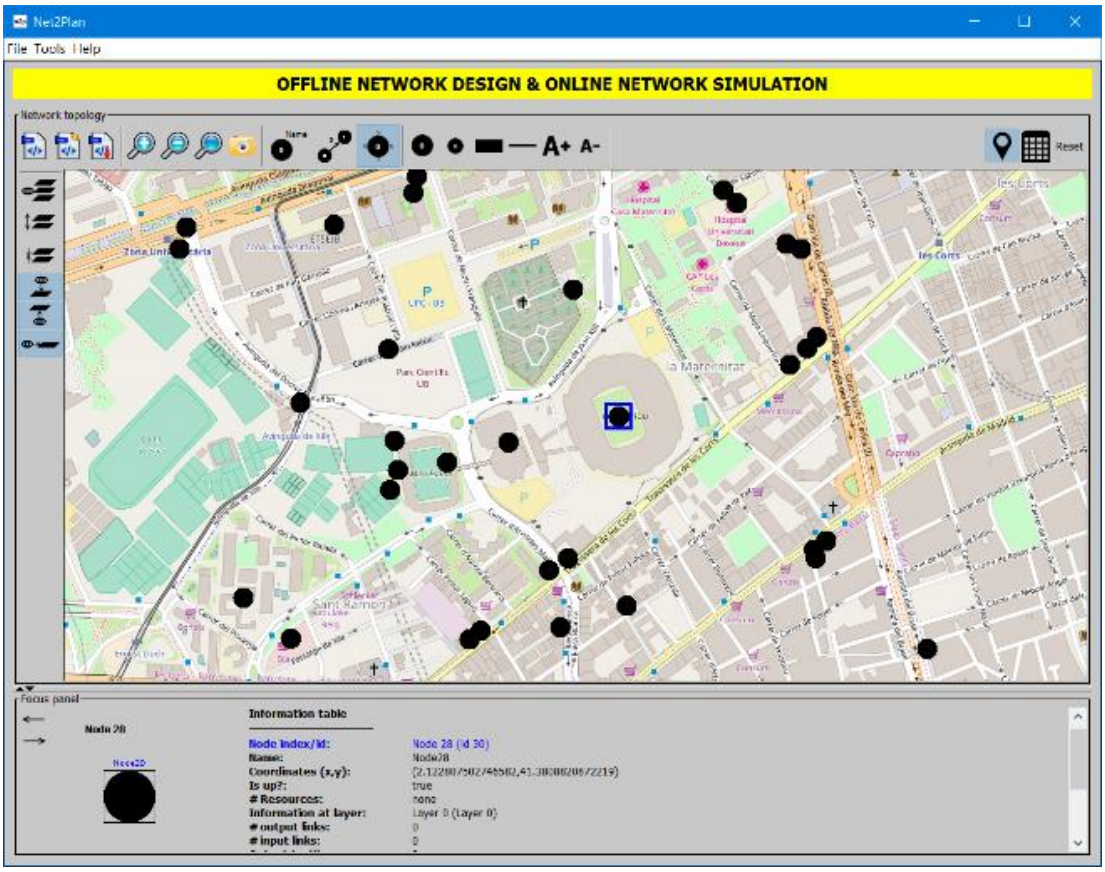

Figure 1: Radio Access Network Topology in Barcelona

We contemplate two different scenarios of management logic, “non-orchestrated” and “orchestrated”.

In the first case (non-orchestrated), each service request is handled by the nearest cell [13]. First, we check if the handing cell has enough link bandwidth with the CO to carry the service offered traffic, blocking the request otherwise. Then, the allocation algorithm tries to find available VNFs to traverse the chain dictated by Table 1 (in order). If a particular VNF from the chain is not instantiated or lacks enough capacity, a new instance will be allocated. In case the cell has not enough hardware resources, the algorithm tries to create the remaining VNFs from the chain in the CO. We remark that in order to process a request, the VNFs have to be traverse in strict order from the handling cell to the CO (no going backwards or loops). If the CO also lacks available hardware resources, the service request is blocked.

In the second case (orchestrated), a different approach is used to reflect an orchestrated management, where the serving cell is decided jointly with the service chain allocation. If the nearest cell lacks enough bandwidth to carry a service, the request is not automatically blocked (as the previous case). Instead, all nearby antennas in coverage (and with enough link bandwidth) are ordered (closest first). Then the allocation algorithm will try to allocate the request on the first available cell in the list that has enough IT resources (following the same procedure explained in the non-orchestrated scenario). Blocking occurs when the service chain cannot be allocated in any cell in coverage. In both scenarios, we assume cells have enough radio capabilities to serve a request [14].

Lastly, once all requests at a given time slot have been processed (either in non-orchestrated or orchestrated case), idle VNFs will be de-instantiated to free unused resources.

Table 1: Service Type Requirements and Associated Service Chains

Service	***Chained VNFs ****	***Bandwidth req.***
VoIP	*NAT-FW-TM-FW-NAT*	*250 Kbps*
Video Conference	*NAT-FW-TM-VOC-IDPS*	*2 Mbps*
Web Service	*NAT-FW-TM-WOC-IDPS*	*4 Mbps*

* IDPS: Intrusion Detection Prevention, FW: Firewall, NAT: Network Address Translation, TM: Traffic Monitor, VOC: Video Optimization Controller, WOC: WAN Optimization Controller

2.2 Simulation Tool and Results

Using Net2Plan, two sets of simulations were performed. For both sets, we establish the following input parameters: 15 Gbps average total offered traffic, service type probabilities of 50% for Web Service, 20% for Video Conference and 30% for VoIP. All service types have a holding time of 100 seconds. Each macro-cell site is assumed to have a server with 16 CPU cores, 64 GB of RAM and 10 TB of HDD. Micro-cell sites have a server with 8 CPU cores, 32GB of RAM and 7 TB of HDD. The CO contains 100 CPU cores, 480 GB of RAM and 27 TB of HDD. The hardware requirements to instantiate each VNF and their concurrent number of operations are shown in Table 2.

Table 2: Number of Concurrent Operations and Hardware Requirements per VNF Instance

VNF	***# of concurrent operations***	***Hardware req.***
IDPS	*2500*	*CPU: 2 cores, RAM: 2GB, HDD: 10GB*
FW	*2500*	*CPU: 2 cores, RAM: 3GB, HDD: 5GB*
NAT	*3000*	*CPU: 1 core, RAM: 1GB, HDD: 2GB*
TM	*2500*	*CPU: 1 core, RAM: 3GB, HDD: 2GB*
VOC	*1000*	*CPU: 2 cores, RAM: 2GB, HDD: 20GB*
WOC	*1500*	*CPU: 1 core, RAM: 2GB, HDD: 10GB*

In this first simulation set, no orchestration between radio resources and NFV management occurs. Each service request is assigned to the nearest cell and a service chain allocation is attempted, always preferring VNF instantiation closer to the users, as described previously. Using the proposed allocation algorithm, the average blocking, in this case, is 40%, and the instant blocking percentage is depicted in **Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**. We appreciate a peak occurring during the football match. This can be attributed to the fact that during this time, the aggregation links between micro-cells and the central gateway, as well as the link between the gateway and the CO, saturate. This bottleneck and its consequent high blocking rate are unacceptable in terms of QoS/QoE.

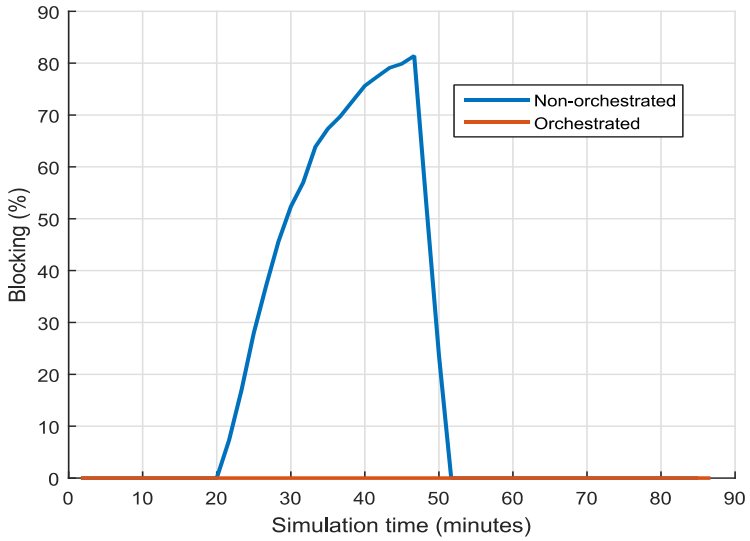

Figure 2: Blocking in Each Instant of the Simulation

The second set of simulations were performed using an orchestrated management (as explained in Section 2.1), where the serving cell and the allocation VNFs are decided jointly.

Figure 3 and Figure 4 show the average resource utilisation time pattern in both sets of simulations for each type of node: macro-cells, micro-cells and CO. We can observe micro-cells have the highest utilisation during most of the simulation for all three resources. This behaviour is to be expected, since the server on micro-cells has the less hardware capabilities than their macro-cells counterparts; and as people approach and sit in the stadium the number the utilisation grows until the match ends. At that time we also note a sudden increase of resource utilisation in macro-cells, from people leaving the stadium, slowly decreasing as they abandon the simulation area.

It is important to note that both cases present similar patterns, with the crucial difference that in the orchestrated case, all requests that were previously blocked in micro-cells due to the lack of bandwidth are now rerouted through nearby macro-cells.

Figure 5 illustrates this by showing the average link utilisation of micro and macro-cells in both cases (orchestrated and non-orchestrated). In both situations, the micro-cell utilisation is the same, blocking services that cannot be allocated in the non-orchestrated case (and with no utilisation of the nearby macro-cells). However, the orchestrated case reflects that all the user requests that cannot be allocated by micro-antennas, are rerouted through nearby macro-antenna raising their link utilisation while achieving 0% blocking. We can also observe a peak of utilisation around minute 50 (the end of the football match) in the non-orchestrated case. This happens because all the people leaving the stadium are trying to request services from the nearest macro-cell antennas, thus causing a peak in the link utilisation. Meanwhile, in the orchestrated case the utilization increase is less pronounced since the macro-cells are already used during the football match, and the service requests are distributed among several antennas further from the stadium.

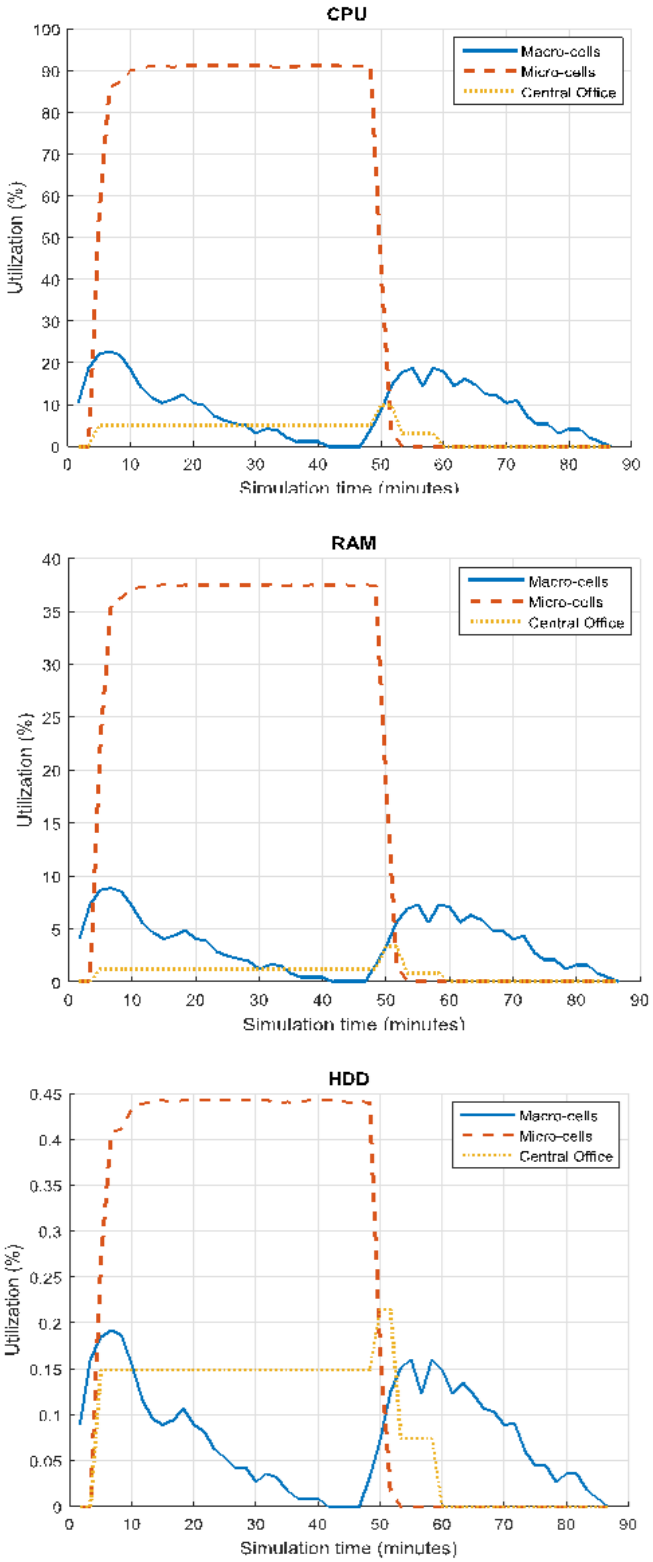

Figure 3: Hardware Utilization, Non-Orchestrated Case.

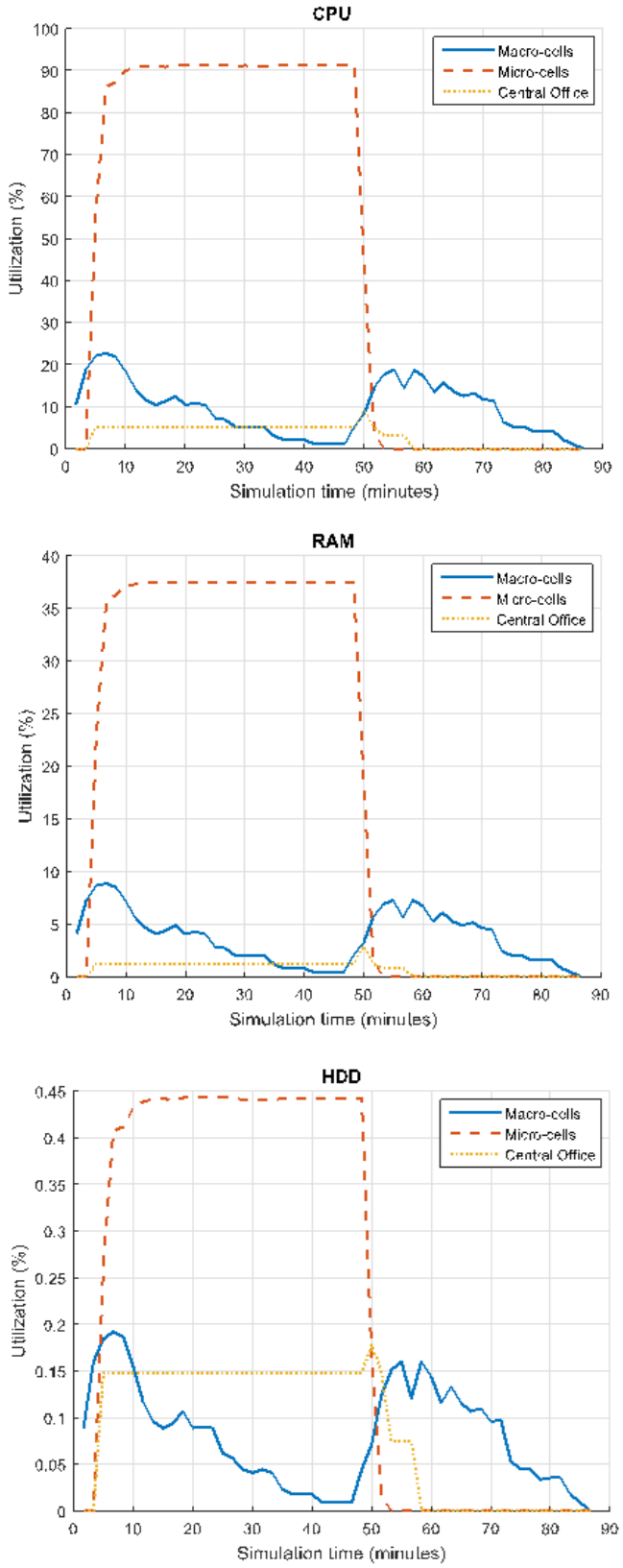

Figure 4: Hardware Utilization, Orchestrated Case.

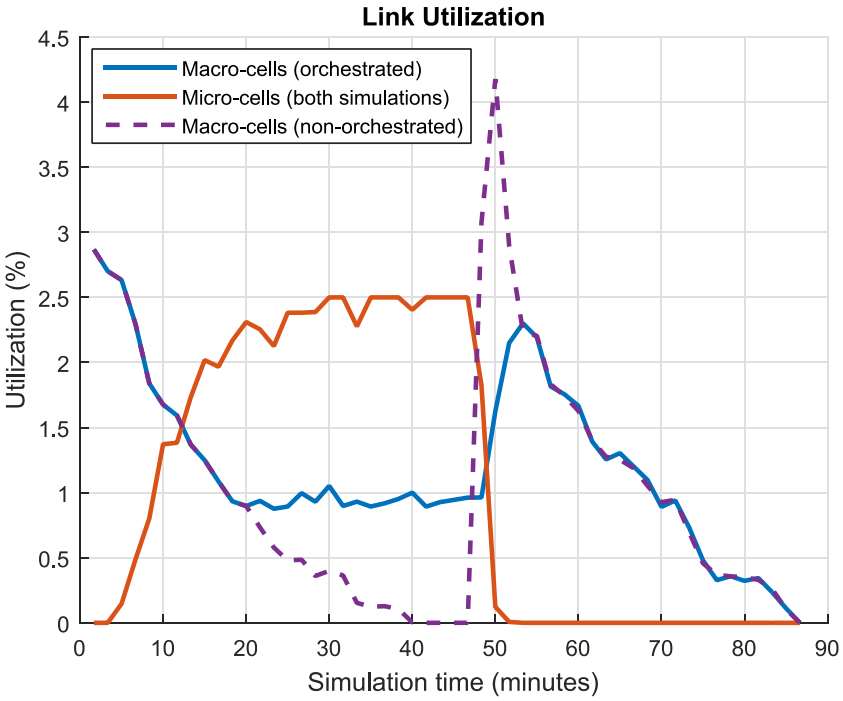

Figure 5: Average Link Utilization in Both Simulations

3 Quality of Service Assurance Framework

Ensuring the Quality of Service (QoS) per tenant bases, i.e. assuring the SLA, is an important aspect of the service lifecycle management. The logical C-RAN manager/orchestrator needs to simultaneously take into account both, the radio status (i.e. volume of traffic, geographical distribution of traffic, etc.) and the cloud capabilities (i.e. available IT resources, VM to VM communication requirements, etc.) for all the actions related to the service lifecycle management. To do so, forming a *QoS insurance feedback loop*, as depicted in Figure 6, is inevitable. Such a loop consists of three main steps:

- *Monitoring*: A phase in which performance monitoring parameters are collected from the radio/cloud elements (e.g. SC physical network function, virtual machines, etc.) and handed over to the next step (decision-making). Depending on the nature of the resources, i.e. radio or cloud, QoS requirements, the available Service Level Agreement (SLA), etc., the monitoring parameters might vary from one use case to another.
- *Decision making*: A phase in which performance metrics collected in the previous step are processed. Depending on the situation and available resources, a decision will be taken to ensure the level of QoS (with the help of a dedicated algorithm). Besides available resources, in a multi-tenant scenario, the decision making process needs to take into account the status of other tenants (i.e. keeping a good level of QoS for one tenant should not cost the failure of others). In principle, the nature of such a decision making algorithm can range from a greedy heuristic to a complex cognitive form.
- *Reaction*: Upon making a decision, the management/orchestration system needs to coordinate the interaction with the other lower level modules such as Element Management System (EMS), Virtual Network Function Manager (VNFM) and Virtual Infrastructure Manager (VIM) to react appropriately.

Implementing this QoS loop means adding the radio dimension to the standard cloud orchestration system defined by ETSI [5], and/or to shift the traditional radio network management mentality towards a cloud-oriented mind-set [15].

This section highlights the service management and orchestration challenges that arise when trying to guarantee the QoS on a per-tenant basis over the distributed C-RAN architecture introduced by the SESAME project. To do so, we initially investigate the performance parameters and measurement procedures on the radio and the cloud side individually. Next, we try to combine and associate them with a single set, analysing how such a joint list of parameters affects the decision making process in a multi-tenant radio-cloud environment. Finally, some solutions for the decision making algorithm and reaction mechanism to enforce the decision are presented. Note that more in deep details about the cognitive decision-making processes will be presented separately in the next section.

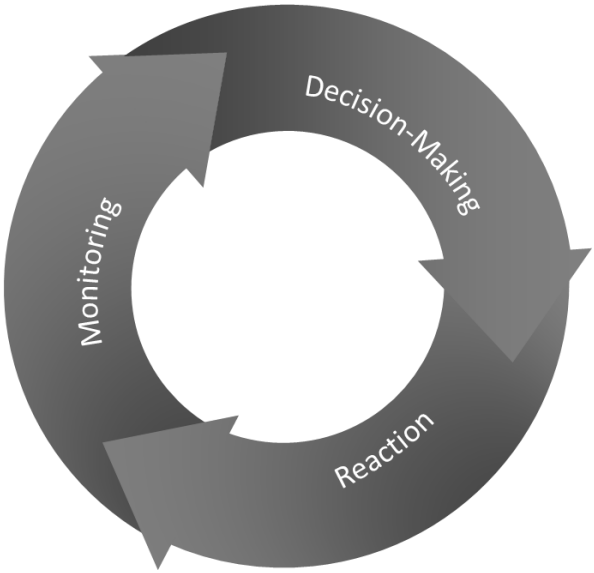

Figure 6: QoS insurance feedback loop

3.1 Parameters to Monitor

In 5G networks, various parameters can be monitored, each of which depicts the system performance from a specific angle. A naive QoS insurance approach is based on a high number of monitoring parameters with the objective to include all the system details and, with this, build a solid failure prevention and system stability mechanism. In reality, monitoring too many parameters increases the complexity of the data management/processing. In a C-RAN system, it is possible to monitor parameters like bandwidth, radio traffic profile over time, CPU usage, etc. However, not all of these inputs might be useful for every use case and every end user. Therefore as a common practice, it is fundamental to set up a list of parameters that will represent the state of the system at any given point of time, in line with the expected level of performance and defined QoS policies. In this section, we initially review the monitoring metrics in the radio and the cloud systems. Then we will present a tentative list of necessary monitoring parameters in envisioned in SESAME.

3.1.1 Radio Access Network Parameters

The performance measurements for RANs are typically divided into a number of sub-groups. For example, [16] divides measurements into:

- RRC connection related measurements,
- E-RAB related measurements,
- Handover measurements,
- Cell level radio bearer QoS parameters,
- Radio resource utilization related measurements,
- UE-associated logical S1-connection measurements,
- Paging related measurements,
- Measurements related to equipment resources,
- Common LAs of overlapping RAT's coverage,
- RF Measurements.

Given the functionality of the eNodeB[1], it is no surprise that the majority of this deal with the performance of the radio link and radio link related procedures. While these performance measurements are of vital importance to the radio infrastructure owner, depending on exactly where the physical virtual cell divide is implemented (SC VNF), only a sub-set are likely to be of interest to a virtual network operator (VNO). The SESAME project has proposed a number of extensions to the standard performance measurements defined in [7] that enable a sub-set of measurements to be collected and recorded on a per-PLMN (and therefor per-VNO) basis. In addition, SESAME has grouped performance measurements into one of three distinct categories:

- Common measurements that are of interest to physical infrastructure owner and VNO alike,
- Infrastructure owner private measurements that are not shared with any VNO,
- Virtual network operator specific measurements where each VNO sees a tailored version that reports a sub-set of measurements, relating only to the VNO's virtual network.

[1] For further details see, *for example*: https://en.wikipedia.org/wiki/EnodeB

For example, all stakeholders may be interested in the availability of the physical cell but, whilst the infrastructure owner may be interested in the data throughput of the entire physical cell, each VNO is only interested in the proportion relating to their users and exchanged with their Evolved Packet Core (EPC). Another area where SESAME has identified the need for VNO specific data is handover performance; as the UEs of different VNOs may be handed in from and out to different macro networks, it is important to separate the associated measurements on a per PLMN basis.

Traditionally, performance measurements are reported periodically as files generated by an element (such as an eNodeB) and transferred to a management system. 3GPP has defined two file formats: [17] defines an XML[2] based file format whereas [18] provides the same information in an ASN.1[3] encoded form. Once uploaded to the management system, performance measurement reports may be retrieved using the File Transfer IRP[4] defined in [19].

A key consideration is the time intervals over which performance measurements are made and reported. In addition to the list of values being reported on, both 3GPP file formats make use of two time-related elements:

- The Granularity Period defines the time interval over which measurements were made and is usually synchronised to the wall clock so that measurements from different elements can be compared directly.
- The Reporting Period defines how often report files are generated and is usually a multiple of the granularity period so that an individual report file may contain one or more granularity periods. The Reporting Period may not be synchronised to the wall clock and may be randomised in order to distribute the load.

The selection of suitable values for these elements depends on a number of factors including the number of network elements deployed and associated data volume and the use to which the data is put. For example, it is likely that data volume measurements employed primarily for billing need only be collected daily whereas metrics such as call set-up and handover success rates used for diagnosis and network optimisation purposes will be collected much more frequently; e.g., every five minutes. To this end, the 3GPP specifications provide for collection and reporting of different sets of measurements in different time frames. This capability is of vital importance in solutions such as SESAME where standard performance management reports are used both for SLA validation and to trigger self-optimisation and self-healing actions.

3.1.2 Light DC Parameters

In the cloud environment, monitoring parameters can be divided into three main categories: (i) Virtualization Infrastructure (NFVI), micro-server's hardware; (ii) Network Functions Virtualisation (NFV) virtual machines performing specific network functionalities, and; (iii) SDN-based monitoring for virtual links.

The SESAME NFVI (Light DC) is formed by clustering CESCs. Therefore, it is important to keep track of the individual performance per micro server. Key performance metrics can be used for this purpose, i.e.:

---

[2] For more informative details see, *inter-alia*: https://en.wikipedia.org/wiki/XML
[3] For more informative details also see: https://en.wikipedia.org/wiki/Abstract_Syntax_Notation_One
[4] Also see: https://en.wikipedia.org/wiki/Integration_Reference_Point

- CPU utilisation: As the brain of the micro-server, CPU carries out the instructions of VNF, performing the arithmetical, logical, and input/output operations. Thus, it is important to monitor its utilisation and possibility keep it as low as possible;
- RAM utilisation: RAM is used to load information required by VNFs for faster access thereby improving the overall performance. If a micro server runs out of RAM, a portion of the hard drive can be dedicated to the virtual memory. This process is called swapping, which will cause performance degradation since the hard drive is much slower than RAM (e.g. 1000 times slower). It is important to monitor the RAM utilisation and possibly add RAM to micro-servers in case of need;
- Hard Disk Drive (HDD) utilisation: The operation of the system kernel, hypervisor and agents on the micro-server needs space on the HDD for normal operating processes including paging files and certain caches. The application running on the server (VNFs) also needs space to write temporary data to cache for efficient operation as well as permanent data that will be accessed by the user. Thus, low free space on the HDD might cause micro severe performance issues;
- System hardware: micro server might include other devices such as HWA (e.g. GPU, FPGA, DSP, etc.), CPU fan, power supply, etc., that affect its overall performance. Health performance parameters of hardware as well as metrics such as temperature, air flow and humidity need to be also monitored.

European Telecommunications Standards Institute (ETSI) in [20] divided the monitoring matrices of NFV into two main categories:

- VNF monitoring parameters,
- Network Service (NS) – a chain of VNF to provide an added-value service – performance metrics.

From a conceptual perspective, a VNF is a virtual machine (VM) that runs a specific application inside which permits performing the same functionality as a network middle-box (e.g. virtual packet gateway (vPGW) in comparison to the actual PGW hardware). ETSI in [20] suggested a way to specify different deployment flavours for the VNF in VNF descriptor (VNFD). These parameters can be either VM related information, e.g. CPU utilisation, bandwidth consumption, etc., or VNF specific such as, calls per second, number of subscribers, number of rule flows per second, VNF downtime, etc. As mentioned previously, one or more of these parameters could be influential in triggering a reaction on the QoS loop.

At NS level, monitoring parameters represent metrics that are tracked to check the level of NS compliance with the agreed SLAs (e.g. NS downtime). These parameters will be part of NS descriptor (NSD) as ETSI suggests in [20] and can be used for specifying different deployment flavours of an NS and/or to indicate different levels of NS availability. Examples of these parameters are: calls per second, number of subscribers, number of rules, flow per second, etc.

For the third monitoring category, i.e. the SDN-*based*, a deep-dive into the details of the SDN data plane is necessary. SDN by definition provides control of the data plane flows, whereas monitoring itself is specific to the management plane. This raises the question of whether the current SDN controllers should include a dedicated monitoring functionality, or this part has to be outsourced to the management plane components.

Currently, the OpenDaylight (ODL) [21] controller provides SDN statistics related to the OpenFlow plugin project [22]. The metrics are based on the statistics manager module and include per-node and table based flows statistics, which can be invoked via APIs and RPCs using

the YANG model definition[5]. Some of the information that can be gathered using the ODL statistics module include:

- Individual and aggregate flow statistics;
- Flow table statistics;
- Port statistics;
- Group statistics;
- Meter statistics;
- Queue statistics.

The device-centric data collection that ODL offers currently is a good start for providing SDN-*related* reporting of current network status. However, a consolidated metrics overview from a service deployed on the top of the ODL nodes and interfaces is still a missing gap. One alternative to "address" this is a design of dedicated library on the northbound that will offer managed data collections based on the statistics unit. Using YANG definitions for different monitoring sets or service-*based* template definitions can achieve this. The data sets could be further invoked by a REST-ful API in a client application responsible for representation and management of the desired service data patterns.

A monitoring tool based on the approach mentioned above would be useful in any SDN managed scenario: datacentre-cloud, mixed cloud-SDN, 5G-NFV etc. The requirement for such a tool should be addressed timely and in parallel with the SDN innovation deployments for datacentres and NVF. It is within the scope of the SESAME project to provide a global monitoring solution that will essentially cover the main aspects of the mixed radio-cloud infrastructure.

3.1.3 Joint Cloud-Enabled RAN Environment

Due to the amount of available monitoring information coming from the radio and cloud environments, it is essential for a radio-cloud system to be able to provide a reduced set containing the most relevant metrics for the involved services. As described before, the infrastructure owner will demand different, i.e. more comprehensive information than the VNO requesting a service. From a tenant point of view, the QoS guarantees are strictly associated to a given service (i.e. a sequence of SC VNF and service VNFs that provides an added-value MES). This implies that most low-level monitoring data needs to be abstracted to offer a simplified view of the status of the service, matching the metrics with the agreed contract terms independently of their domain. For example, service availability is a widely used metric that, in the case of SESAME, needs to be composed by the data made available by each of the components in the service chain. Furthermore, other metrics such as the number of UEs connected to a CESC need to be correlated with the capacity of the service VNFs to serve content to those users, which results in a combined metric with the corresponding guarantee term.

As a result, for SESAME we propose a functional block that not only aggregates monitoring data coming from both radio and cloud environments but also processes and associates it to a specific service being offered to a tenant. This task requires first, to establish the QoS parameters associated with a service, then to identify which radio and cloud metrics are involved in the evaluation of the overall service performance, i.e. are part of a common high-level metric relevant for the VNO.

[5] Also see: *https://en.wikipedia.org/wiki/YANG*

3.2 Monitoring Tools

In this section, the existing monitoring components are discussed, focusing on the underlying technology. After that we will tackle the SESAME monitoring tools from SDN perspective.

3.2.1 Current Technologies

Today, there is no doubt that the amount of existent network monitoring tools and platforms is beyond the expectations. The Standford summary gives a good overview of the current monitoring technologies, classified in different categories [23]. However little is known about the tools based on SDN, that should offer enhanced capabilities to the detailed infrastructure and OTT SDN applications.

Concerning this task, we have looked at several of the existing technologies for monitoring related to SDN, which will be presented in the following sections. We will furthermore present the concepts of the monitoring module related to Netfloc[6], the SDN controller of the SESAME project. More details of the monitoring functionality and use-cases will be presented in Deliverable 6.3.

3.2.2 TSDR (Time Series Data Repository)

TSDR is a project from the OpenDaylight framework that is designed for collecting, storing, querying, and maintaining time series data. The project can be configured to export the data in Grafana[7], by using the controller URL as data source (http://[SDN_controller] 8181/tsdr/nbi) and queries as tsdrkey, such as in the following example:

[NID=OPENFLOW:1][DC=FLOWSTATS][MN=BYTECOUNT][RK=NODE:OPENFLOW:1,TABLE:0,FLOW:L2SWITCH-352]

[NID=OPENFLOW:1][DC=PORTSTATS][MN=TRANSMITTEDPACKETS][RK=NODE:OPENFLOW:1,NODECONNECTOR:OPENFLOW:1:2]

[NID=OPENFLOW:1][DC=FLOWSTATS][MN=BYTECOUNT][RK=NODE:OPENFLOW:1,TABLE:0,FLOW:L2SWITCH-352,FROM:0,UNTIL:NOW]

The metrics can be also queried via APIs using the following URL: http://[SDN_controller] 8181/tsdr/metrics/query

3.2.3 The Skydive Project

Skydive [24] is an open source real-time network topology and protocols analyzer from RedHat[8]. It aims to provide a comprehensive way of understanding what is happening in the network infrastructure.

Skydive agents collect topology information and flows and forward them to a central agent for further analysis. All the information is stored in an Elastic search database. Skydive is SDN-agnostic but provides SDN drivers in order to enhance the topology and information flows.

Figure 7 shows the Skydive graphical user interface. There is a cluster that shows all the nodes, the VM itself and all the connections. On the right side, there are further informations about every single port.

---

[6] See: *http://netfloc.readthedocs.io/en/latest/*
[7] See: *https://grafana.com/*
[8] See: *https://www.redhat.com/en*

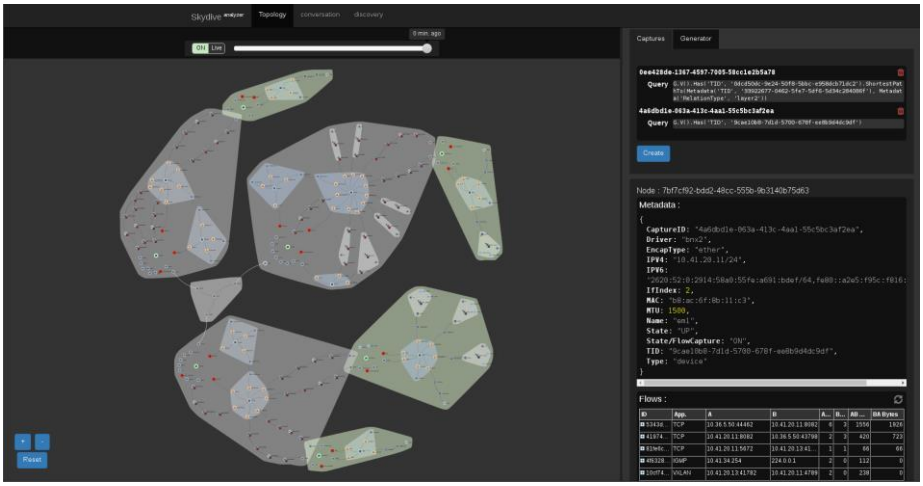

Figure 7: GUI from the monitoring tool Skydive

3.2.4 sFlow

sFlow [25] is a sampling technology for monitoring network traffic, embedded within switches and routers. It also represents an industry standard supported already from many vendors.

Figure 9 represent an example of the output of SDN port 3 on the Netfloc node, both for ingress and egress traffic in bits/s. The sFlow Trend also allows establishing the threshold on specific ports, as shown in Figure 10, where we can also see the ports in the Compute node and the possible parameters to be set as thresholds.

Figure 8 shows the basic sFlow architecture. It consists of a sFlow Agents installed in the network devices that continuously send a stream of sFlow Datagrams to a central sFlow Collector. The flows are afterwards analysed in the Collector in order to produce a real-time, network-wide view of traffic flows.

The SESAME SDN testbed is enriched with sFlow agent installed on all the nodes: Control[9], Compute[10], Neutron[11] and Netfloc/Switch. The solution is based on the open source sFlowTrend-Pro tool from inMon [26]. It represents the data in the sFlowTrend GUI including probing of historical traffic information.

Figure 9 represent an example of the output of SDN port 3 on the Netfloc node, both for ingress and egress traffic in bits/s. The sFlow Trend also allows establishing the threshold on specific ports, as shown in Figure 10, where we can also see the ports in the Compute node and the possible parameters to be set as thresholds.

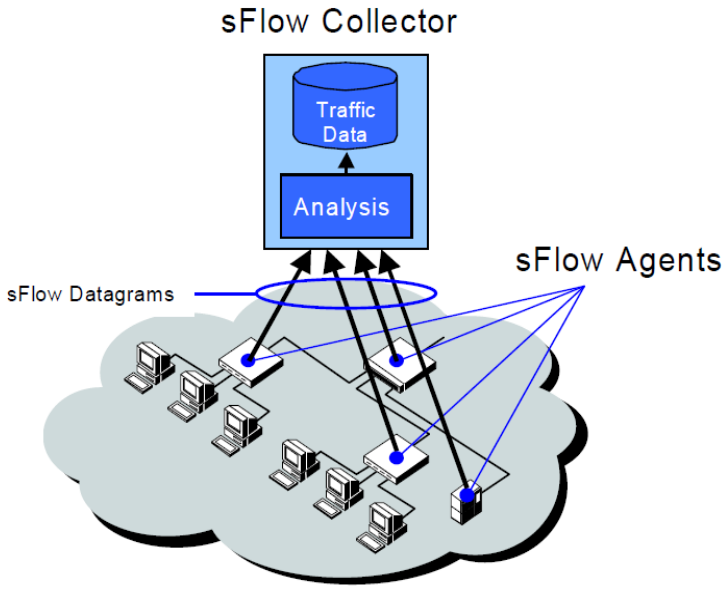

Figure 8 : Basic sFlow architecture

[9] See: *https://docs.openstack.org/newton/install-guide-ubuntu/nova-controller-install.html*
Also see: *https://docs.openstack.org/mitaka/install-guide-ubuntu/neutron-controller-install.html*
[10] See: *https://ask.openstack.org/en/question/50263/what-is-openstack-compute-node/*
[11] See: *https://wiki.openstack.org/wiki/Neutron*

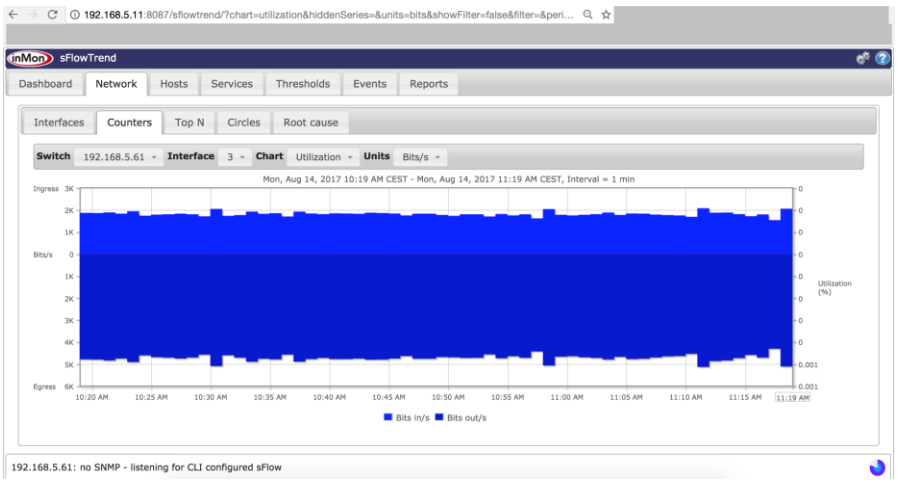

Figure 9: sFlow Trend port statistics in bits/s

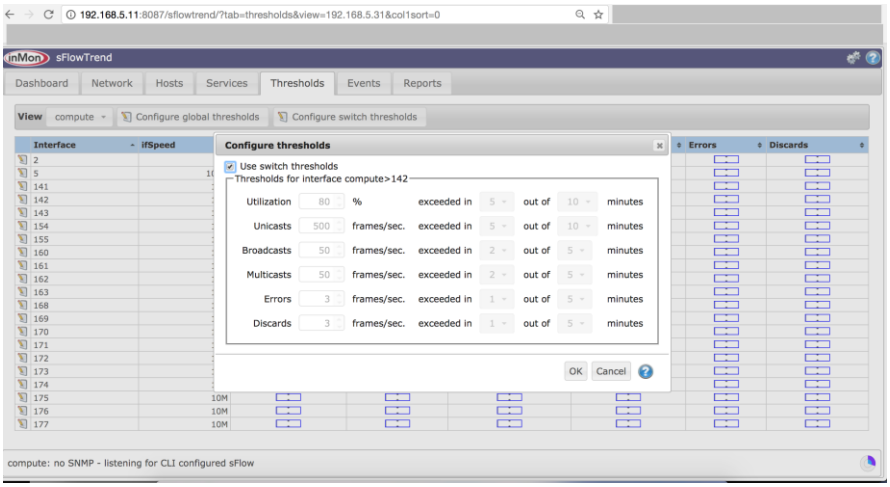

Figure 10: sFlow Trend threshold setup per port

3.2.5 SESAME Network Monitoring

In the context of SESAME, we have created the Netfloc monitoring module, which integrates into Netflogi [27] – the GUI for Netfloc. Apart from the visual representation of the network switches, it also offers information on all SDN testbed components, along with detailed information for ports and flows. Netflogi has been offered as a Docker[12] image now:

sudo docker pull delv/netflogi

docker run -it [netflogi-id]

[12] Also see: *https://en.wikipedia.org/wiki/Docker_(software)*

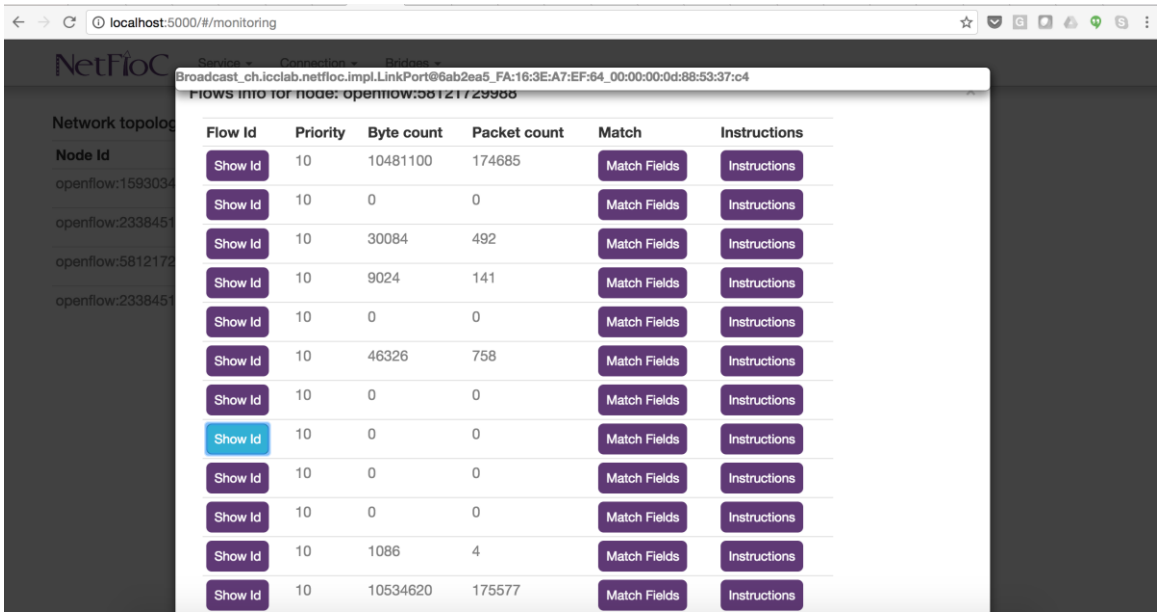

Figure 11: Netflogi flow statistics per port

Figure 11 shows a screenshot from the Netflogi monitoring view, related to flows and flow IDs, as well as bytes and packets per flow.

The next goal in the SESAME project was to extend this functionality and create metrics exporter for displaying the data real-time interactively. For this matter we have created the Netfloc exporter for Prometheus [28], whose goal is despite offering the metrics visualisation in Prometheus and Grafana, to also enable integration with the CESCM monitoring component.

The monitoring library in Netfloc also pretends to offer the possibility to create a per-use-case monitoring configuration, allowing a fine-granular overview and control of the network applications. The reason why we are not creating yet another monitoring tool is justified in the fact that many monitoring tools already exist, as stated earlier, and the main idea is to leverage those along with the SDN capabilities to create a targeted monitoring per application. With this approach, we give a focus on exposing the group of metrics and views that have critical meaning for the application developer, in this case the VSCNO. Using the monitoring information, the SESAME NFVO will be able to provide dynamic management of the VNFs and the NSs on-demand and according to the defined thresholds, with the objective to comply to the SLAs contracted by the operators.

3.3 Reaction Mechanism

The proposed solution to ensure the QoS in the joint-radio cloud environment of SESAME is described as a pseudo-algorithm, where we proposed a performance evaluation of edge cloud services. The evaluation mechanisms are based on two types of actions:

(i) Preventive (proactive) actions: Foreseeing saturation levels that can potentially lead to QoS breaches. Warning alerts are sent to the appropriate management module for it to consider performing corrective actions, e.g. VNF scaling;

(ii) *Correction (reactive) actions*: In case of breach in the QoS, violation alerts are reported to be analysed and corrected, *if possible*.

Figure 12 depicts the SESAME proposed QoS insurance feedback solution.

Services in SESAME, in line with ETSI definitions [20], are composed of SC PNF, SC VNF and service VNFs, which are described in the NSD. Therefore, in the Monitoring phase, the proposed solution (Figure 12), should retrieve metrics from both cloud and radio parameters. The Metric Aggregator (MA), as its name indicates, is the responsible for combining and filtering the monitored parameters collected and associates them with the running services over the SESAME platform. MA continuously processes the collected monitoring values for the QoS or SLA evaluation. Depending on the use case, the algorithm used for this purpose might be a simple threshold checking logic or a complex multi-attribute decision making process. Bearing in mind its fast processing time, we select the threshold checking procedure. To this end, a threshold, i.e. Warning Threshold (WT), is defined for each of the monitored metrics, aiming to detect the critical values per case and trigger a warning alert for the specific failure component. Note that, WTs are defined in a way that the multi-tenant capabilities of the system are taken into account. In this sense, the proposed solution is inflexible, i.e. values of WTs will not adapt dynamically according to the real-time needs of the system. Having said that, SESAME targeted introducing a more complete solution in the project lifetime.

In conjunction with a more complex data process, SESAME envisioned a module denoted as the Decision Support System (DSS), as shown in Figure 12. The main responsibility of DSS is to detect the level of severity of the QoS evaluation process done in MA and decide whether a reactive or a proactive action is needed. Basically, such a decision will be made based on the high level SLA agreements made with VNOs. Such a high level agreement will indicate points such as at what Point of Presences (PoPs) a VNO will be present, how much of overall IT resources are dedicated to a VNO, etc., as presented in [29]. With the help of DSS, the SESAME solution, in addition to the "per NS performance metrics" will bring in the high level SLA agreements into calculations.

NFVO is the responsible module to perform the appropriate reaction based on the analysis made in the previous steps. There are a set of possible reaction mechanisms, including:

- Reconfiguring the flow of data in a NS (i.e. changing the SDN rules),
- Migrating the NS within the PoP of from one PoP to another,
- Scaling up/down the whole NS (i.e. instantiation of a parallel service or terminating a running one),
- Scale in/out of VNF (i.e. adding more resources to a VNF).

Depending on the nature of requested process (proactive/reactive), nature of warning (radio/cloud), available resources, possibilities of VNFs indicated in the VNFD by developers, NS agreement with VNO depicted on NSD, the NFVO selects and applies a reaction mechanism. More details about cognitive decision making processes studied in the framework of SESAME are presented in the next section.

Figure 12, depicts the workflow of the reaction process. That is, NFVO has the possibility to interact with the EMS (managing the radio parameters of SC or SCs) and/or VIM (managing the cloud infrastructure – e.g. OpenStack). In this sense, EMS and VIM are the enablers' components for the adaptation service of NFVO on cloud and/or radio domain. It is worth noting that, in a more advanced scenario, EMS and VIM can also be enhanced with Self-x features, aiming to enable them to make local decisions for a CESC or a set of CESCs (sub set of Light DC).

From the infrastructure owner perspective, besides the status of each provided service, the overall system performance is also highly relevant; therefore, MA is also able to expose general information about the system status. This information helps the infrastructure owner to perform a temporary capacity upgrade if the expected use of resources does not meet planned terms, e.g. due to overloaded use of services/users. The data can be seen visually from the CESCM Portal through for example a customised dashboard.

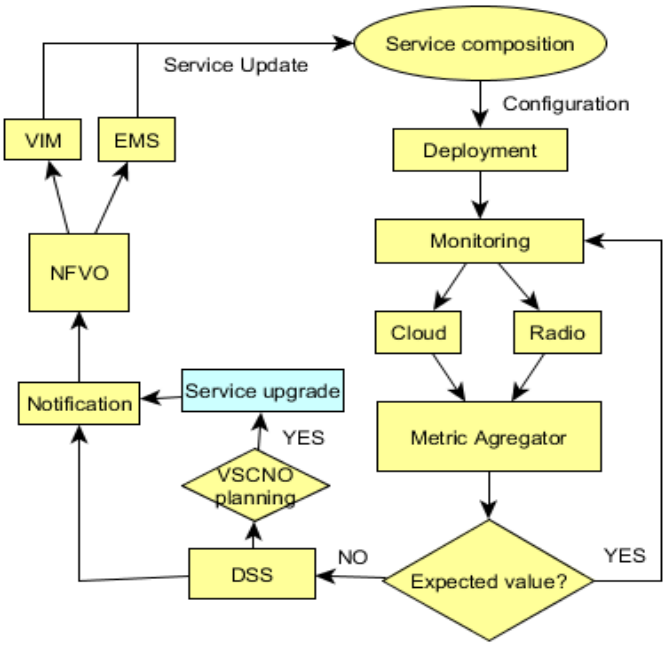

Figure 12: SESAME QoS insurance feedback loop

4 Cognitive Decision-Making Processes

As stated previously, precise decision making processes are an important part of the QoS assurance solution. One of the most effective way of decision-making is cognitive processes which are regarded as processes resulting in the selection of a course of action among several alternative possibilities in very complex situations. In this section, we will review cognitive decision-making processes in the context of SESAME.

4.1 First Thoughts

The dynamic management of the radio resources in a cellular network is carried out by a set of Radio Resource Management (RRM) functions aimed to ensure the efficient use of the resources while, at the same time, "meeting" the Quality of Service (QoS) requirements of the services provided to the users. The parameters of these RRM functions, as well as other general parameters of the CESCs, can be dynamically and automatically modified at runtime by means of Self-Organizing Network (SON) functions, also referred to as Self-X functions.

The decision-making processes of RRM/SON functions exploit information acquired from the environment, reflecting the status of the network and the User Equipment (UEs). The acquired information, which can contain in general a wide range of metrics, needs to be appropriately processed to extract the adequate knowledge that leads to the best decisions in each case.

In deliverable D6.3 a framework is presented for virtualising the RRM/SON functions in multi-tenant small cell networks based on the SESAME architecture [43]. It is depicted in Figure 13, and involves different RRM/SON VNFs, which can be tenant-specific, meaning that they only support RRM/SON procedures for a specific tenant, or they can be common, meaning that they support RRM/SON procedures that are common to all the tenants. Figure 13 illustrates in red and green, *respectively*, the RRM/SON VNFs specific to tenants T1 and T2, while the common VNFs are depicted in brown. In addition, one of the common VNFs, is the Radio Network Information Service (RNIS) that handles the acquisition, processing and collection of different measurements to monitor the status of the RAN and the User Equipment (UE) and exposes them to the rest of RRM/SON functions.

Based on the above, this section focuses first on the management and orchestration of the RRM/SON VNFs. Then, it assesses, through different examples, the orders of magnitude of the requirements of the involved VNFs. Finally, to gain further insight on the operation of the RNIS VNF, this section discusses some possible functions for extracting knowledge models that can be used by the RRM/SON cognitive decision making processes.

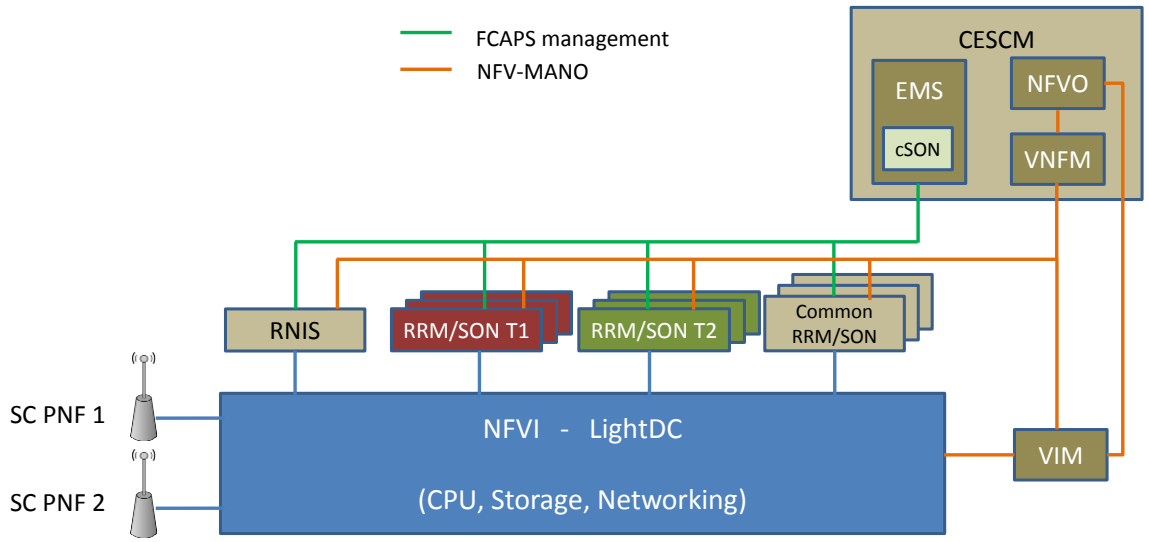

Figure 13: Framework for virtualising RRM/SON functions in a multi-tenant environment

4.2 Management of Virtualized RRM/SON Functions

The management of the RNIS VNF and the RRM/SON VNFs depicted in Figure 13 involves two different frameworks, namely the Fault, Configuration, Accounting, Performance and Security (FCAPS) management and the NFV-MANO.

The FCAPS management is performed by the Element Management System (EMS) and includes a set of functions for configuration and re-configuration of the operational parameters of the RNIS VNF and the different RRM/SON functions (e.g. adjusting parameters and thresholds of the AC, handover, etc.). It also includes a set of functions for the collection of Performance Management (PM) measurements characterising the operation of these functions (e.g. number of successfully established Radio Access Bearers (RABs), etc.).

The NFV-MANO involves the management functionalities provided by the NFVO, VNFM and VIM to support the lifecycle management of these VNFs (e.g. instantiation, scaling, termination) as well as of the entire Network Services (NSs) in which the VNFs are chained as components across the Light DC.

NFV-MANO uses different templates to describe the components and the connectivity of an NS [44]. The Network Service Descriptor (NSD) template is used by the NFVO and describes the NS topology (constituent VNFs, virtual links between VNFs and VNF forwarding graphs) and the NS characteristics (e.g. functional scripts and workflows for initialising, terminating and scaling the NSs). For example, in the framework of Figure 13 the NSD for implementing a RRM/SON function as a virtualised NS can involve the SC-PNF, the RNIS VNF and one RRM/SON VNF. Each one of these VNFs will be defined by means of a VNF Descriptor (VNFD), a template that specifies the deployment and operational behaviour requirements of the VNF and includes the functional scripts for lifecycle events of this VNF (initialisation, termination, scaling). Similarly, the SC-PNF will be defined by means of a PNF Descriptor (PNFD).

The topology of the NS is defined by the VNF Forwarding Graph (VNFFG) that specifies how the different VNFs and PNFs are interconnected. A VNFFG is specified through its corresponding VNFFG Descriptor (VNFFGD). Finally, each of the connections in the VNFFG requires a virtual link (VL) defined by means of the Virtual Link Descriptor (VLD), a template that specifies the connectivity, interface and requirements of these virtual links.

In the context of SESAME prototyping framework, which has adopted the TeNOR orchestrator and OpenStack as VIM, the abovementioned NSD, VNFD, PNFD, VNFFGD and VLD descriptors are

defined as Heat OpenStack Templates (HOT), in order to be forwarded to the OpenStack VIM.

When a NS has to be created, the NFVO translates the descriptors into HOT templates that are sent to the VIM and the VNFM. The VIM manages the NFVI to map the different NS components into the actual hardware (i.e. it creates the VMs, decides the mapping between VNFs and VMs and sets up the virtual links to connect the VNFs). Once completed, the VIM notifies the VNFM so that it can take care of the VNF lifecycle management. At this stage, the VNFM will instantiate the VNFs and will monitor their behaviour based on collecting performance measurements from the VNFs and from the VMs used by these VNFs. The process is illustrated in Figure 14.

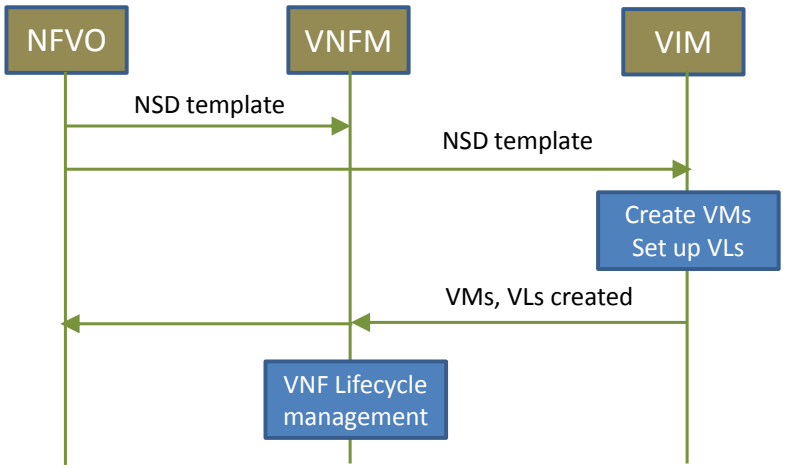

Figure 14: NFV MANO process for creating a network service

4.3 Examples of Virtualized RRM/SON Use Cases

This section intends to illustrate the implementation-related aspects of specific RRM/SON functions under the framework of Figure 13.

4.3.1 Admission Control

The considered AC function (described in [45]) runs at each CESC as a common RRM/SON VNF and makes decisions on the acceptance/rejection of the establishment of Radio Access Bearer (RAB) for the different tenants. This allows keeping the SC PNF functionality as tenant-agnostic with respect to the admission of RABs. Such AC involves two different checks: (i) a capacity check at cell-level, to evaluate the aggregated number of Physical Resource Blocks (PRB) used by all the tenants in a cell after accepting the new RAB request and ensure that the cell has sufficient physical resources for serving the new RAB, and; (ii) the per-tenant capacity share check, to establish an upper bound in the PRB usage by the RABs of a tenant in accordance with the capacity contracted through the SLA. In order to adapt to different network conditions, the AC VNF includes a dSON function used to dynamically modify the thresholds used in the abovementioned checks. The adjustments made by the dSON are supported by a cSON function, which runs at the EMS and has an overall view encompassing multiple cells.

The RNIS provides the common AC VNF with the average number of PRBs used by each tenant. This measurement can be obtained by extending the “PRB usage for traffic” measurement of [46] to include per-tenant measurements. Similarly, in order to estimate the number of PRBs required by a new RAB with specific bit rate requirements, the RNIS provides an estimation of the average bit rate per PRB that can be obtained in the cell. This estimation can be calculated

by dividing the aggregate IP throughput of the cell by the aggregate PRBs that have been allocated for transmission (see [45] for further details).

Table 3 illustrates the order of magnitude of the requirements associated to the VNFs of this example, i.e. the VNF implementing the AC decision making and the dSON and the VNF implementing the RNIS. Computations are based on the averaging window parameters defined in [45] for a scenario with two tenants. In particular, it is assumed that PRB usage measurements are provided by the RNIS in periods of 0.1s, while the dSON should store multiple samples and average them to make adjustments of the thresholds every 300s. The computational requirements of the AC checks are dependent on the RAB arrival rate in a cell, for which a rate of 1 request/s has been considered for each tenant.

Table 3: Requirements of the VNFs involved in the considered examples

		Computation	Memory	Networking
AC	AC VNF (including dSON)	~ 100 ops/s	~ 10 KB	~ 200 b/s (RNIS-AC)
	RNIS	~100 ops/s	~ 10 KB	~ 1 kb/s (SC PNF - RNIS)
CCO	RNIS	~10^3 ops/s	~ 10 MB	~1 kb/s (SC PNF - RNIS and RNIS-cSON)
PS	PS VNF	~10^7 ops/s	~ 10 MB	~ 150 Mb/s (PS VNF - SC PNF)

4.3.2 Coverage and Capacity Optimisation

Coverage and Capacity Optimisation (CCO) is a SON function that targets a continuous coverage and the provision of a sufficient achievable bit rate in the service area. This function operates by adjusting RF parameters of the cells, such as the downlink transmit power, the antenna tilt and the antenna azimuth. A typical operation of CCO consists in identifying specific symptoms such as coverage holes, cell overshoots or excessive cell overlaps and then to trigger an optimization process, e.g. by exploring different candidate configurations of the adjusted parameters through techniques like genetic algorithms or particle swarm [47].

Centralized implementations of this function are typically considered [48] since the optimisation should be made jointly considering multiple cells in the service area. Therefore, the CCO would be implemented as a cSON function residing in the EMS, which in turn is part of the CESCM. In any case, the CCO cSON function can benefit from the availability of the RNIS VNF running at the Light DC, which can provide local information about the coverage conditions in each cell.

In particular, the RNIS collects measurements from the SC-PNF (e.g. measurements performed by the RAN and UE reported measurements such as Reference Signal Received Power (RSRP), Reference Signal Received Quality (RSRQ) and Received Signal Strength Indication (RSSI)). This collection imposes requirements mainly in terms of memory and networking capabilities in the VL between the RNIS VNF and the SC PNF.

Table 3 provides an estimation of the RNIS requirements, assuming small cells with a coverage radius of 100m and that the coverage is rappresented in pixels of 5m x 5m. Besides, it is assumed that the RNIS provides storage capabilities to accumulate measurement reports (over the period of 24h for example). It is assumed that UEs provide an average of 1 RRC Measurement Report per second. As for the computational requirements, the RNIS will have to

perform a high number of operations to process the measurements and characterise each pixel. However, given that the operation is performed on a long-term basis, a relatively low computational complexity is envisaged and, in addition, the processing work can be scheduled during the times when the CPUs are less occupied. Similarly, the VL between the RNIS and the cSON does not pose tight requirements because, even if a large file characterising all the pixels has to be provided to cSON, the transmission can be done without stringent delay constraints and only very sporadically when triggered by the cSON.

4.3.3 Packet Scheduling

The Packet Scheduling (PS) function is responsible for assigning the Physical Resource Blocks (PRBs) of a cell among the different UEs served by that cell and to select the physical layer parameters used for each transmission (e.g. modulation and coding scheme, antenna mapping in case of multi-antenna transmission). PS operates with a time granularity given by the Transmission Time Interval (TTI), which currently is 1 ms in Long Term Evolution (LTE).

The PS is executed at the Medium Access Control (MAC) layer. Therefore, since SESAME prototyping framework assumes a functional split at the S1 interface, the PS would be part of the SC PNF. Anyway, from a more general perspective, this sub-section assumes the possibility of using other functional splits. Specifically, the functional split between MAC and PHY enables a scenario in which a single SC PNF implementing the PHY layer with multi-carrier support can be logically partitioned into multiple instances, each one associated with one or several LTE carriers/RF chains, while the MAC and above layers for each of these instances are implemented as VNFs. In this case, each tenant can be allocated with a different SC PNF instance [49], thus providing isolation between tenants, while the VNFs can be made tenant-specific. This facilitates the implementation of tenant-specific PS VNFs, so that each tenant can customise the PS criteria for assigning PRBs to its own UEs. With this functional split, the interconnection of the PHY layer of one tenant with its corresponding VNFs is done through the network Functional Application Platform Interface (nFAPI) [50].

A tenant-specific PS VNF typically operates based on the instantaneous Channel Quality Indicators (CQI) reported by the UEs and on the information about the buffer status. Besides, the tenant-specific PS VNF should also deliver the scheduled data packets to the SC PNF. The transmission of this information across the nFAPI interface poses high networking requirements for the VL between the SC PNF and the tenant-specific PS VNF. For example, bandwidth requirements in the order of 150 Mb/s and latencies of 2 ms are mentioned in [51]. Storage requirements include both the UE data waiting for transmission in the buffers and the CQI reports of these UEs. Computational complexity in terms of number of operations will be highly dependent on the specific implemented algorithms, but the number of operations will increase in some orders of magnitude with respect to those required by simpler RRM functions like the AC. Besides, the execution of the PS every 1ms puts stronger requirements in terms of the number of operations/s. Table 3 summarizes some orders of magnitude of the envisaged requirements for the tenant-specific PS VNF.

4.4 Examples of Knowledge Discovery Functions

An efficient cognitive decision-making involves the capability of properly analysing the measurements from the environment reflecting the network and the UE status in order to extract the adequate knowledge that will drive the subsequent decisions. In this respect, in Deliverable D3.1 a general framework for knowledge discovery was presented (see section 3.3 of [52]). Firstly, it handles the collection of the different measurements (e.g. network counters included in the PM files, UE measurements, etc.) and performs different pre-processing tasks

(e.g. data cleaning to remove noise and inconsistent data, combination of data from different sources, data selection, etc.). Then, a set of artificial intelligence-based knowledge discovery functions are applied to learn from the users and the network in order to build models that reflect the relevant knowledge that will drive the cognitive decisions.

In the context of the RRM/SON virtualization framework of Figure 13, and depending on the applicability of the considered knowledge models, the knowledge discovery can be part of the RNIS VNF (e.g. in case that the knowledge models can be utilized by multiple RRM/SON functions) or can be implemented as part of an RRM/SON VNF itself.

To further illustrate the possibilities offered by the knowledge discovery, the following sub-sections discuss two different knowledge dimensions that can be extracted to characterise the traffic of a cell in the time and the spatial domains, respectively. Each case is illustrated with specific algorithmic examples.

4.4.1 Cell Level Time-domain Traffic Characterisation

This dimension defines how the traffic of a cell varies as a function of time. The traffic can be measured in different ways, such as the load factor, the total number of users, the total data rate, etc., and it can be aggregated or split among QoS classes. Then, knowledge discovery techniques can be applied to the past observations of traffic in order to provide:

- *Classification of the time domain traffic pattern*: Time correlations in the traffic evolution of a given cell should be detected to identify existing seasonalities at different levels (e.g. intra-day variations, variations during the week between working days and weekend, variations in the traffic between winter and summer, etc.) and classify the cell accordingly. In this way, based on historical samples of the cell's traffic, a classifier would decide which type of traffic pattern the cell exhibits among a number of pre-defined possible classes. Typically, tools should be based on supervised learning, in which the classifier is trained using reference patterns of some cells that fit under the possible classes. Table 4 specifies a possible classifier of the traffic pattern based on Support Vector Machines (SVM).
- *Learning the traffic behavior in the time domain*: This refers to the identification of a model that captures the cell traffic at different periods of time (e.g. hours, days of the week, etc.) and allows identifying time periods exhibiting similar traffic levels. A possible tool for extracting this model is the Self Organizing Map (SOM) clustering technique, which groups multidimensional data based on their statistical properties using unsupervised learning. Table 4 specifies an example of time domain traffic pattern extraction based on SOM.
- *Prediction of the future traffic*: A prediction model can be extracted to anticipate future values of the traffic evolution in a cell. This can feed various decision making processes regarding planning (e.g., in order to anticipate the need to deploy additional network nodes) and optimisation (e.g., in order to tune handover parameters in neighboring cells to absorb traffic if the cell is anticipated to be overloaded) mainly depending on the time scale at which the prediction is conducted. Table 6 presents an example of traffic prediction using Support Vector Regression (SVR).

Table 4: Classification of the cell's time domain traffic pattern using SVM.

- Let $\mathbf{X}=(\mathbf{x}_t, \mathbf{x}_{t-1}, ..., \mathbf{x}_{t-(N-1)})$ be a time series with N samples of the observed traffic or load in a given cell at different times t. This will usually be a long time series, as it will have to include multiple days, months, etc. Each sample can consist of multiple components $\mathbf{x}_t$=(Time, Day, Month, ρ) for a given cell reflecting the decomposition of time *t* in hours/day of the week/ month and the

traffic or load level ρ. The objective of the classifier is to make an association between the input time series **X** and the class C(**X**) that specifies the type of time-domain traffic pattern. Examples of classes C(**X**) could be "traffic pattern with variations between working days and weekends", "traffic pattern with variations between working days/weekends and summer/winter seasonality", etc.

- The time series **X** is processed to extract a vector of M features $F(\mathbf{X})=(f_1(\mathbf{X}),f_2(\mathbf{X}),....,f_M(\mathbf{X}))$. The number of features M will usually be a small value (e.g. in the order of 10). This process is necessary because it would not be practical to directly use such long vector **X** as input to the SVM. The M features should be defined a priori based on expertise and based on the specific problem on hand. Some examples of features for this problem could be: average traffic per week, average traffic during weekends, average traffic per year, average traffic during summer months, ratio of max/min traffic per week, per day, etc.

- The vector F(**X**) will be the input to the SVM, which will provide as a result the class C(**X**) that **X** belongs to. Details on the operation of a SVM can be found in [53] for the case of binary classification. This can be extended to the more general multi-class classification by hierarchically combining several binary SVM classifiers [54]. In essence, the classification with a SVM is made by positioning the input data F(**X**) in relation to the optimal hyperplane that defines the separation between the classes. This usually requires an initial transformation of the input data F(**X**) into a higher dimensional space through a kernel function, so that the data is made linearly separable through a hyperplane.

- The internal configuration of the SVM (i.e. the parameters that define the optimal hyperplane) is obtained through a training stage that uses as input S training time series $\mathbf{X}_1,..., \mathbf{X}_S$ of different cells. Being a supervised learning technique, the class $C(\mathbf{X}_i)$ of each training time series $\mathbf{X}_i$ is pre-defined by an expert. Then, the inputs for the training stage will be the vectors of features of these training time series and the associated classes, i.e. $(F(\mathbf{X}_1),C(\mathbf{X}_1)), ..., (F(\mathbf{X}_S), C(\mathbf{X}_S))$. Using these inputs, the training stage solves a convex optimisation function to get the parameters of the hyperplane (see [53] for details).

Table 5: Learning the time domain traffic pattern using a SOM

- The input to the SOM machine will be the time series $\mathbf{X}=(\mathbf{x}_t, \mathbf{x}_{t-1}, ..., \mathbf{x}_{t-(N-1)})$ with N past samples of the observed traffic or load in a given cell at different times *t*. Each sample consists of multiple components $\mathbf{x}_t$=(Time, Day, Month, ρ) reflecting the decomposition of time *t* in hours/day of the week/ month and the traffic or load level ρ.

- Internally, the SOM is composed by a structure of neurons. Each neuron i has an associated reference vector (denoted as weight vector) $\mathbf{m}_i=(m_{i1},m_{i2},...,m_{in})$ where n equals the number of components of the samples $\mathbf{x}_t$, i.e. n=4 in our case. For each input sample $\mathbf{x}_t$ the SOM machine executes an unsupervised learning process that updates the values of the weight vectors $\mathbf{m}_i$ of the different neurons according to the Kohonen's algorithm explained in [55]. This update is done in such a way that the values of $\mathbf{m}_i$ progressively capture the statistic of the different components in $\mathbf{x}_t$.

- After having processed the N time samples $\mathbf{x}_t$, the output of the SOM machine are the resulting weight vectors $\mathbf{m}_i$. They will capture the combinations of attributes in $\mathbf{x}_t$ that have appeared in the time series **X**. In this way for example we could obtain the traffic level that a given cell exhibits at certain days or weeks (e.g. "the cell exhibits a traffic level of 0.7 every Wednesday at 18h").

- In addition, the SOM machine also provides the distance between the different neurons (computed e.g. as the Euclidean distance between their weight vectors). This distance is used to build a map that groups together neurons with close distance. The analysis of this map could allow identifying similarities between the traffic existing in different hours or days.

- A parameter to be set is the number of neurons of the SOM. Usually, a fixed number of neurons is assumed, although there are some works that allow dynamically modifying the number of

neurons based on the input vectors $\mathbf{x}_t$ [56].

Table 6: Traffic prediction using SVR.

- The input to the SVR machine at time *t* will be a time series composed by the past L observations of the cell traffic, denoted as (ρ_t, ρ_{t-1}, ..., $\rho_{t-(L-1)}$). The SVR will provide as output an estimation of the traffic at time *t*+1 through a prediction function $\rho_{t+1}=f(\rho_t, \rho_{t-1}, ..., \rho_{t-(L-1)})$. The time granularity of the different samples can be on a per-hour basis, on a daily basis, or even on a monthly basis, depending on the desired time scale of the prediction. - The internal configuration of the SVR, i.e. the prediction function f(·), will be learnt by means of a training process that will use the past N observations of the cell traffic **T**=(ρ_t, ρ_{t-1}, ..., $\rho_{t-(N-1)}$). For that purpose, the time series **T** will be first split into different training tuples, each one composed of the traffic at a given time and the traffic at the previous L samples. This split is done by applying a sliding window of length L over time series **T**, resulting in a total of N-L training tuples of the form (ρ_{t-i}, ρ_{t-i-1}, ..., ρ_{t-i-L}), i=0,...,N-L-1. Then, the training process to obtain the function f(·) from these training tuples is based on solving a convex optimization function (details can be found in [57]-[59]). - As a reference of the order of magnitude of L, in [58] prediction is made with L=10 and in [59] prediction is made with L ranging between 3 and 18. Besides, the SVR also involves a small number of free parameters that should be set empirically, such as the kernel function parameters or the penalty term to errors. - Note that we could also consider predictions with a time horizon longer than 1, i.e. predicting $\rho_{t+\Delta}$ at time t. One option to do this is by using the forecasted values ρ_{t+1}, ρ_{t+2},... and apply the same function f(·) obtained for predictions with horizon 1, as done in [58]. Another option would be to build in the training stage a different function $f_\Delta(\cdot)$, that is specifically designed to make predictions with horizon Δ, that is $\rho_{t+\Delta}=f_\Delta(\rho_t, \rho_{t-1}, ..., \rho_{t-(L-1)})$.

4.4.2 Cell Level Space-domain Traffic Characterisation

The characterisation of the traffic in the space-domain can be done in different terms, such as the geographical distribution of the users, traffic load, services/applications or QoS class. In general, given the proliferation of multiple small cells that can be located indoors and deployed in tall buildings, a 3D characterisation can be required. Then, knowledge discovery techniques include:

- *Clustering spatial traffic (hot-spots)*: It targets the identification of unusual concentrations of UEs in limited geographical areas. The identified hot-spots can be later on analysed to extract further knowledge such as the classification of the type of spatial user distribution, the degree of heterogeneity of this distribution, etc. Candidate techniques for addressing the hot-spot detection problem are the geospatial clustering tools [60][61]. In this respect, Table 7 details an example of hot-spot detection using the Kernel Density Estimation[13] (KDE) tool.
- *Learning mobility patterns*: this knowledge intends to identify if the traffic follows some specific mobility patterns inside the cell that can be characterized in terms of prototype or representative trajectories followed by many of the users in the cells (e.g. trajectories directed towards specific points such as a metro station, etc.). The identification of trajectories can have applicability in different contexts. For example, they can be used in the self-planning to decide appropriate antenna settings (e.g. if there is a well identified representative trajectory a sector of a cell site can be pointed in the direction of this trajectory), in the self-optimisation in order to set the handover thresholds and load balancing thresholds, etc. Different tools can be used

[13] See: https://en.wikipedia.org/wiki/Kernel_density_estimation

for analyzing trajectories of mobile objects, usually based on clustering mechanisms (see e.g. [62]-[65] for some examples in the literature). Table 8 presents an example of identification of trajectories using a SOM.

Table 7: Detection of traffic hot-spots using KDE

- The inputs to the KDE machine will be a set of points $\mathbf{p_i}=(x_i,y_i,z_i)$ characterising the 2D or 3D coordinates of the position where a UE is receiving service from the cell under analysis. The set of points $\mathbf{p_i}$ will be obtained from past observations of the traffic served by the cell, so they will belong to the service region of the cell S. The hot-spot detection can be defined either from a global traffic perspective or at QoS class level. In the latter case the set of points $\mathbf{p_i}$ will be limited to the UEs belonging to a certain QoS class.
- The KDE machine uses the set of points $\mathbf{p_i}$ to create a continuous field or surface that estimates the observed density of points. This surface is denoted as $\phi(x,y,z)$ and is computed for the different positions $(x,y,z)\in S$. The values of (x,y,z) are discretized according to a certain grid size. The computation of $\phi(x,y,z)$ is done by placing a moving function (denoted as kernel) which weights the points $\mathbf{p_i}$ under its influence according to their distance to the position (x,y,z). The kernel should be a decreasing radially symmetric function providing a total weight of unity over the region of influence. Details about the process and the types of kernel function can be found in [66]. It is worth mentioning that, although the process of [66] is for bivariate (2D) data, the KDE concept is generalizable to multivariate data (e.g. 3D) (see [67]). The parameters to be set for applying the KDE method are the grid size and the kernel bandwidth that defines the shape of the kernel function.
- After the KDE generates the surface $\phi(x,y,z)$, the hot-spots can be identified by grouping the contiguous points of the grid with a density higher than a certain threshold. Each group of contiguous points forms a hot-spot. It can be defined in terms of its contour or using other metrics such as the centroid or the radius.

Table 8: Identification of trajectories using a SOM.

- The inputs will be a set of traces of the form (UE_id, x, y, z, t) where UE_id is the identifier of a UE connected to the cell under analysis, and x,y,z,t represent the positions where this UE is located at different points of time.
- Based on these traces, we first build the trajectory for each UE_id as the concatenation of n 3D coordinates at different time instants. Then, each input to the SOM will be a trajectory with dimension 3n (or with dimension 2n if only 2D coordinates are considered) associated to UE_id=j and defined as $\mathbf{r_j}=(x_j(t_1),y_j(t_1),z_j(t_1), x_j(t_2),y_j(t_2),z_j(t_2),\ldots.., x_j(t_n),y_j(t_n),z_j(t_n))$.A total of M trajectories will be built in this way, corresponding to different UEs served by the cell.
- The SOM will be composed by a set of neurons. The weight vectors of the neurons will have dimension 3n, i.e. $\mathbf{m_i}=(x_{i1}, y_{i1}, z_{i1}, x_{i2}, y_{i2}, z_{i2},..., x_{in}, y_{in}, z_{in})$. Then, following a similar process as explained in Table 5, for each input trajectory $\mathbf{r_j}$ the values of the weight vectors will be updated according to the Kohonen's algorithm.
- After having processed the M trajectories, the SOM will have learnt the set of prototype trajectories existing in the cell. These prototype trajectories will be defined by the final weights of the neurons $\mathbf{m_i}=(x_{i1}, y_{i1}, z_{i1}, x_{i2}, y_{i2}, z_{i2},..., x_{in}, y_{in}, z_{in})$. For the i-th prototype trajectory the weights (x_{i1},y_{i1},z_{i1}) corresponds to the initial point and (x_{in},y_{in},z_{in}) to the end point, while the rest of components represent the intermediate points.

4.5 Final Thoughts

This chapter has analysed a framework for virtualising RRM/SON functions in the small cell networks considered in SESAME. After addressing the management and orchestration of the involved VNFs, three different examples of functions have been analysed, namely AC, CCO and PS, illustrating the order of magnitude of the computational, storage and networking requirements of the involved VNFs.

Based on the analysis of these examples, it is observed that in most of the functions the computational and memory requirements are quite reduced in relation to the capabilities of the micro servers that compose an execution platform like the one of the SESAME project. This reflects the feasibility to implement packages encompassing multiple RRM/SON functions as a single VNF. Consequently, the virtualisation of RRM/SON appears as a feasible option from an implementation point of view, which opens the door to substantial innovation in the way that RRM/SON functions are conceived and brought to the market. Specifically, it is envisaged that virtualisation can facilitate more open markets where third-parties could provide RRM/SON solutions that can be easily plugged into the network thanks to the flexibility offered by the NFV-MANO framework.

The presented framework enables the use of knowledge discovery mechanisms for extracting appropriate knowledge models that can drive the cognitive decisions made by different functionalities. In this respect, this chapter has illustrated, with specific examples, possible algorithmic approaches for characterising the time and spatial domain traffic in a cell.

5 NFVO Prototype

As detailed on D6.1 [4], TeNOR [30] the orchestration solution developed in the context of T-NOVA project [31] has been selected as bases to develop SESAME NFVO. D6.1 provided comprehensive details about the modular architecture of the orchestrator and the particular functionalities of each module. Moreover, workflows for service lifecycle management actions, i.e. service deployment, service termination and service modification have been presented on D6.1.

In this section, we focus on the elements that SESAME added on top of what was existing. In particular, the Network Service Notification Manager (NSNM) is a feature developed in SESAME project to complete the monitoring feedback loop in practice. It is responsible for receiving and processing notifications regarding the performance of a network service deployment. NSNM is provided as a micro-service inside TeNOR but it can also be used as a standalone service communicating with the orchestrator via the API. This feature is part of the general SESAME monitoring module and only works together with Prometheus [32].

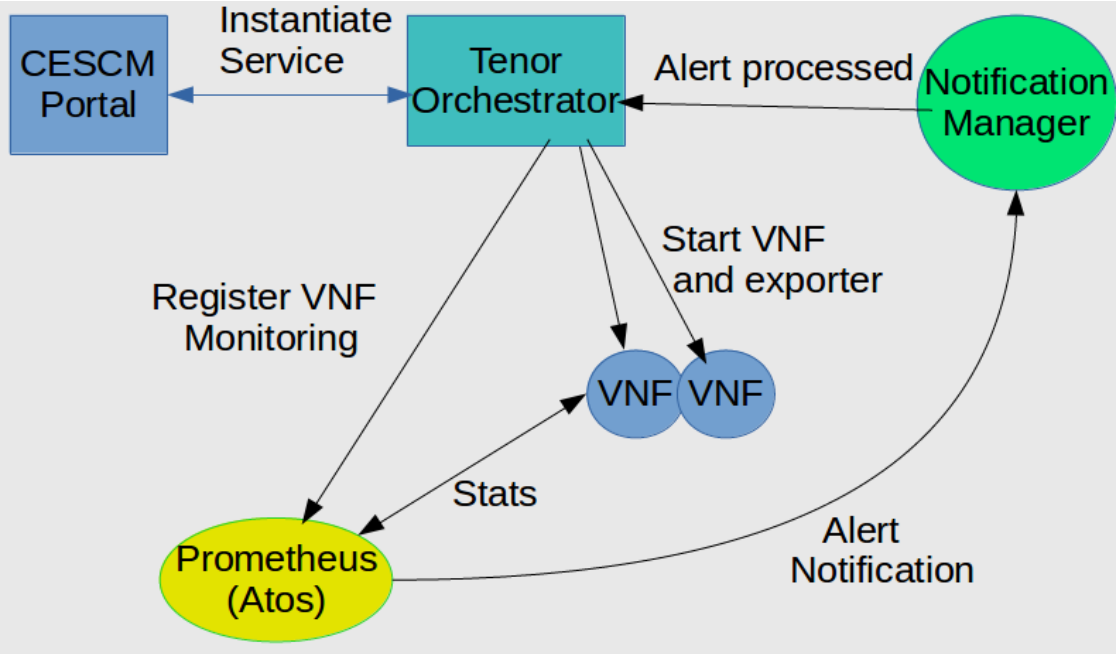

Figure 15: SESAME Monitoring Schema

5.1 The Big Picture: Services and Monitoring

As Figure 15 shows, the SESAME solution's user, e.g. SCNO/VSCNO system administrator, interacts with the system via CESCM Portal developed in WP5. The portal first posts the NS and VNF descriptors to the TeNOR catalogue, where their fields are verified and stored.

Then upon the user request, a service instantiation request is sent to trigger the deployment process. TeNOR will create the stack with the networks and machines as defined in the descriptors. In principle, a network service composed by many Virtual Network Functions, each of which delivers a certain functionality. The bigger the service is the most critical monitoring gets.

To collect the monitoring data from the VNFs we need a service to run on the virtual machines created, collecting and exposing relevant system data. Prometheus already offers this service,

called node exporter. To get the right functionality out of the exporter, first we need to include some relevant lines in the VNF descriptors so that the exporters can start to extract and push the monitoring data upon the NS creation time. In this way, Prometheus guarantees the NS monitoring right on time. We have to add the following information in the VNFD, under the Virtual Deployment Unit (VDU) section. That is, the added command will be among the first executed in the machine. The command just downloads the exporter for its GitHub repository [33] and starts it on the default port.

```
"bootstrap_script": "#!/bin/bash\nwget
https://github.com/prometheus/node_exporter/releases/download/v0.14.0/node_exporte
r-0.14.0.linux-amd64.tar.gz\ntar -xzf node_exporter-*.tar.gz\nrm node_exporter-
*.tar.gz\nmv node_exporter* node\ncd node/\n./node_exporter"
```

If the instantiation goes on without any problem, TeNOR will register the VNFs of the service for monitoring to the external Prometheus service. For this, we need to send a post request with the details of the service instantiated, like the service ID and the IP/ports of the machines that the service is running. The request should send the information in a JSON format as presented in the example below.

```
POST http://[IP]:5000/target
 {
  "targets": [ "IP:PORT", “localhost:5050”,..... ],
  "labels": {
  "env": "prod",
  "job": "node",
  "service": "sesame-service",
  "service_id" : "T3SDavkl3pall9688g"
  }
 }
```

Prometheus will then communicate with the node exported to collect statistics for the service and in case there is a QoS violation, will send a notification to the NS Notification Manager. The notification contains a list of information useful for selecting the reaction to the alerts received. The details about the Prometheus operations will be presented on D5.2.

The NSNM is currently available and integrated as part of SESAME NFVO. To install it, clone the project from its public GitHub repo [34] and follow the installation instructions. For using the notification manager, it is necessary to setup the configuration file under the TeNOR directory *ns-prometheus-prometheus*. The config file should look like the example below. Before starting the monitoring, it is needed to verify the correct IP and port of the Prometheus installation.

```
servicename: ns_prometheus_monitoring
environment: development
address: 0.0.0.0
port: 4544
prometheus-adress: 192.165.3.3
prometheus-portL: 5000
```

When SESAME NFVO is installed, one can start it using the invoker module, executing the command.

```
invoker start invoker.ini
```

The monitoring services are not executed by default when TeNOR starts the NSNM should be started manually. For doing so, the three following commands should be ran on the same directory of the TeNOR installation.

```
invoker add redis
invoker add sidekiq
invoker add prometheus-monitoring
```

When all the monitoring related modules have been set correctly, upon the NS instantiate TeNOR will take care of two main actions:

- Start the exporter inside every machine deployed by the service;
- Register the service for monitoring in the Prometheus module that will start collecting all the critical info about the running VNFs.

5.2 SLA Monitoring High-level Workflow

In order to assure the Service Level Agreement (SLA) between the service consumer and the service provider, the workflow sequence diagram is specified. Several steps need to be fulfilled, in order to manage the commercial relationships between the Service Providers (SPs), Function Providers (FPs) and customers.

We have focused on the use case scenario from the perspective of the VNSCO deploying a service on SESAME platform. SESAME platform is able to manage complex network services throughout their entire lifecycle. The workflow presented includes the configuration, deployment/starting a service, scaling in/out, monitoring and termination of a deployed service.

Deliverable 5.2 has depth information about the internal architecture, and description of the phases to start the evaluation for further information; this section will describe the requirements and process that need being integrated with the orchestrator.

5.2.1 Configuration

The SLA module relies on a MySQL database that is deployed in and configured to create the needed tables. Prior to offering services to the user, several actions need to be configured in the system to be operational.

The provider needs to be registered prior to offering any service. The creation of a new provider in the platform is done by the SLA API exposed. The operation can be performed by sending the name of the provider and a UUID[14] associated with it. If the UUID is not provided, the system will create one automatically. An example of the POST request is as follows:

[14] For further information also see: *https://en.wikipedia.org/wiki/Universally_unique_identifier*

```
<provider>
  <uuid>5576abeb-595b-4c0e-b44f-e176a1035da6</uuid>
  <name>provider-sesame</name>
</provider>
```

The possible response code from the request can be:

- 409: The UUID or name already exists in the database
- 201: OK.

Templates also need to be defined before an agreement between the Service Provider and the user is closed. SLA templates are created based on each VNFD and NSD. The SLA template is a draft of the contract that the involved parts will sign once the product (VNF or service) is acquired with minor modifications. The body might include a TemplateId or not. In case that a TemplateId is included in the file, it must match with the one from the url. An example of an existing SESAME SLA template is shown here:

```
<wsag:Template xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement"
xmlns:sla="http://sla.atos.eu" wsag:TemplateId="template-sesame">
    <wsag:Name>TemplateSesame</wsag:Name>
    <wsag:Context>
      <wsag:AgreementResponder>provider-a</wsag:AgreementResponder>
      <wsag:ServiceProvider>AgreementResponder</wsag:ServiceProvider>
      <wsag:ExpirationTime>2018-06-07T14:00:00+02:00</wsag:ExpirationTime>
      <sla:Service> test-service </sla:Service>
    </wsag:Context>
    <wsag:Terms>
      <wsag:All>
        <wsag:ServiceDescriptionTerm wsag:Name="SDTName2" wsag:ServiceName="test-
service"> DSL expression </wsag:ServiceDescriptionTerm>
        <wsag:ServiceProperties wsag:Name="NonFunctional" wsag:ServiceName="test-
service"/>
        <wsag:GuaranteeTerm wsag:Name="GT_Availability">
          <wsag:ServiceScope>test-service
        </wsag:ServiceScope>
          <wsag:ServiceLevelObjective>
            <wsag:KPITarget>
              <wsag:KPIName>Availability</wsag:KPIName>
              <wsag:CustomServiceLevel>
                <sla:Slo>
                  <sla:Constraint>Availability GT qos:Availability</sla:Constraint>
                  <sla:Description> </sla:Description>
                </sla:Slo>
              </wsag:CustomServiceLevel>
            </wsag:KPITarget>
          </wsag:ServiceLevelObjective>
        </wsag:GuaranteeTerm>
      </wsag:All>
    </wsag:Terms>
  </wsag:Template>
```

The possible response code from the request can be:

- 409: The UUID already exists in the database.
- 409: The provider UUID specified in the template does not exist in the database.
- 500: Incorrect data has been supplied.
- 201: Created.

5.2.2 WS-Agreement

A list of available SESAME services is presented in the CESCM portal. The CESCM portal is the entry point to the SESAME service, presented as control plane web GUI. The users of the infrastructure, SCNO and VSCNO, contract and manage the available VNFs. It provides an on-demand control with intuitive dashboards for a comprehensive overview of network assets, instance status, and network usage.

The user selects the service that wants to deploy: every service is associated to a deployment flavour or KPI. A service or function with a specific flavour, allows providers to have some level of control on the possible deployed services, that will determinate the service assurance on a contracted service.

In this scenario, we have defined **#3** different KPIs associated with a service: those must be explicitly accepted or rejected by the user whenever he selects the service to be deployed. We have not considered a negotiation phase, as is a matter of business how the VNO o SCVNO negotiates with the user the service and in the case of SESAME with no specific requirements for VNO o SCVNO is out of scope.

The SLAs are defined by “Agreements”, agreements represent a signed SLA contract between the user and the service provider. Every agreement must be associated with an existing template to be registered. Otherwise, the agreement is not registered.

```
<wsag:Agreement,xmlns:sla=http://sla.atos.eu,xmlns:wsag="http://www.ggf.org/namespaces/
ws-agreement" wsag:AgreementId="agreement-sesame">
    <wsag:Name>SesameAgreement</wsag:Name>
    <wsag:Context>
      <wsag:AgreementInitiator>client-prueba</wsag:AgreementInitiator>
      <wsag:AgreementResponder>provider-sesame</wsag:AgreementResponder>
      <wsag:ServiceProvider>AgreementResponder</wsag:ServiceProvider>
      <wsag:ExpirationTime>2017-04-07T14:00:00+02:00</wsag:ExpirationTime>
      <wsag:TemplateId> template-sesame </wsag:TemplateId>
      <sla:Service>sesame-service</sla:Service>
    </wsag:Context>
    <wsag:Terms>
      <wsag:All>
        <wsag:ServiceProperties wsag:Name="ServiceProperties" wsag:ServiceName="sesame-
service">
          <wsag:VariableSet>
            <wsag:Variable wsag:Name="Availability" wsag:Metric="xs:double">
              <wsag:Location>service- /Availability</wsag:Location>
            </wsag:Variable>
          </wsag:VariableSet>
        </wsag:ServiceProperties>
        <wsag:GuaranteeTerm wsag:Name="GT_ResponseTime">
          <wsag:ServiceScope wsag:ServiceName="sesame-service">
     </wsag:ServiceScope>
          <wsag:ServiceLevelObjective>
```

```
            <wsag:KPITarget>
              <wsag:KPIName>Availability</wsag:KPIName>
              <wsag:CustomServiceLevel>
                <sla:Slo>
                  <sla:Constraint>Availability GT 0.97</sla:Constraint>
                  <sla:Description>
          </sla:Description>
                </sla:Slo>
              </wsag:CustomServiceLevel>
            </wsag:KPITarget>
          </wsag:ServiceLevelObjective>
        </wsag:GuaranteeTerm>
      </wsag:All>
    </wsag:Terms>
  </wsag:Agreement>
```

The response code from the agreement request can be as follows:

- 409: The UUID already exists in the database.
- 409: The provider UUID specified in the agreement does not exist in the database.
- 409: The template UUID specified in the agreement does not exist in the database.
- 500: Incorrect data has been supplied.
- 201: Created.

Once the user has selected a service, the portal sends the order to the orchestrator to deploy the service with the characteristics selected. The template received by the orchestrator is used for managing the lifecycle of network services: the deployment is specified as a Network Service Descriptor (NSD) in form of TOSCA[15] file, and describes the relationship between VNFs and possibly the PNFs that it contains and the links needed to connect VNFs.

5.2.3 Assessment

The next step for an efficient and incident free operation of a network service is the monitoring information. The network operator requires receiving feedback on the service provided to ensure the SLA agreed with the end-user. The information that can be retrieved for a particular network that it owns by requesting the service platform.

In light of efficient and incident free operation of a network service, the monitoring feedback plays a vital role. Moreover, a network service operator will require continuous monitoring data for its deployed network service in order to ensure SLAs it agreed with the end user. The network service operator can retrieve the monitoring information for a particular network service. As a first step, the service platform checks if the network service operator is authorised to retrieve monitoring data for that particular network service. If the authorisation is valid, the system will allows the monitoring evaluation of that service. Otherwise it will be rejected.

Once the service is deployed and running, orchestrator activates the monitoring through the enforcement request in order to take action so the monitoring data is fetched and pushed to the SLA manager. The enforcement is the process by which it is evaluated that the provider complies with an agreement, i.e. the measured metrics for the variables in guarantee terms fulfil the constraints. Actually, 'assessment' would be a more accurate term but we keep 'enforcement' to be compliant with the ws-agreement specification. Enforcement process is depicted in Figure 16.

[15] Further information can be found at: *http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.html*

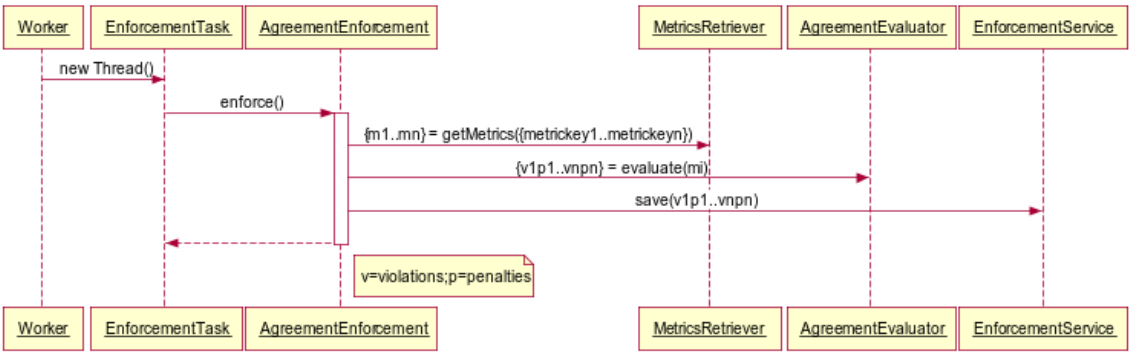

Figure 16 : Agreement enforcement, periodic execution

The monitoring system is independent of the SLA module. For this reason, an adapter has to be implemented in order for the SLA retrieve the metrics needed to do the assessment. The way the external monitoring system communicates with the SLA module is through the Prometheus server, described in Deliverable 5.2.

The monitoring process will start as soon as the new targets are created. Each target corresponds to a service, being possible to have a single target in multiple services as the Service Function Chained is defined in the NSD.

Once a service has been deployed, the NFVO will notify Prometheus with the new targets to be monitored. The information required contains the IPs of the VNFs that composes the service as well as the service identifier. Additional information can be sent in the form of "tags" to Prometheus for it to make more elaborated monitoring queries.

5.2.4 Service Assurance

The last step is to activate the evaluation loop, this is performed by the enforcement. The enforcement job is the entity which starts the assessment of the agreement guarantee terms. An agreement can be assessed only if an enforcement job, linked to it, has been previously created and started. An enforcement job is automatically created when an agreement is created, so there is no need to create one to start an assessment.

It is activated by making a PUT request with the associated agreementId of the desired service, and start command; to deactivate the assurance loop; the procedure is similar by modifying the stop command.

The response code when activating the service assurance will be as follow:

- 403: It was not possible to start the job.
- 200: OK.

5.3 SLA Management / Endpoint and Notifications

There are two types of communication with the orchestrator trough the SLA module, as mentioned before in the Deliverable 5.2, and two types of alarms are triggered by the system: soft threshold alarms and hard threshold alarms.

Soft thresholds alarms are implemented as webhooks. The webhook receiver from AlertManager, will send the alarm once the value has passed the threshold. The monitoring module is configured to send HTTP POST requests to an endpoint with the alert and a

description in JSON format. The following example shows the notification of a service where two of its instances are not available:


```
{
   "status": "firing",
   "groupLabels": {
      "service": "demo-service",
      "alertname": "instance_downn"
   },
   "groupKey": {},
   "commonAnnotations": {
      "summary": "Monitor service non-operational"
   },
   "alerts": [
      {
         "status": "firing",
         "labels": {
            "alertname": "instance_downn",
            "service": "demo-service",
            "instance": "localhost:8088",
            "job": "POP",
            "env": "prod",
            "service_id": "hb323kl789688g",
            "severity": "critical"
         },
         "endsAt": "0001-01-01T00:00:00Z",
         "generatorURL":
"http://prometheus:9090/graph?g0.expr=up+%3D%3D+0&g0.tab=0",
         "startsAt": "2017-05-22T18:13:42.822+02:00",
         "annotations": {
            "description": "localhost:8088 service is down.",
            "summary": "Monitor service non-operational"
         }
      },
      {
         "status": "firing",
         "labels": {
            "alertname": "instance_downn",
            "service": "demo-service",
            "instance": "localhost:8087",
            "job": "POP",
            "env": "prod",
            "service_id": "hb323kl789688g",
            "severity": "critical"
         },
         "endsAt": "0001-01-01T00:00:00Z",
         "generatorURL":
"http://prometheus:9090/graph?g0.expr=up+%3D%3D+0&g0.tab=0",
         "startsAt": "2017-05-23T10:12:02.822+02:00",
         "annotations": {
            "description": "localhost:8087 service is down.",
            "summary": "Monitor service non-operational"
         }
      }
   ],
   "version": "4",
   "receiver": "tenor-notification",
   "externalURL": "http://prometheus:9093",
   "commonLabels": {
      "alertname": "instance_downn",
      "service": "demo-service",
```

```
        "job": "POP",
        "env": "prod",
        "service_id": "hb323kl789688g",
        "severity": "critical"
    }
}
```

Hard threshold alarms are also defined as violation. Violations occur when a term defined in an SLA has not met its guaranteed value, meaning the SLA has been broken.

Actions to be taken as a consequence of a violation happening are out of the scope of the SLA module.

The notification of the violation is represented as follows:

```
<violation>
  <uuid>2396c9f3-bdaf-4305-8be4-2b137054642c</uuid>
  <agreement_id>agreement-sesame</agreement_id>
  <service_name>sesame-service</service_name>
  <service_scope></service_scope>
  <kpi_name>availability</kpi_name>
  <datetime>2017-07-27T14:22:30+02:00</datetime>
  <actual_value>0.6459022667315054</actual_value>
</violation>
```

5.4 Implementation Details

The service is implemented using Sinatra [35] to expose a simple REST API, providing a single endpoint to accept notification alerts as a JSON post request. Since the process of the notification can be complicated and time consuming the service needs to be asynchronous so sidekiq [36] workers are used together with redis database [37] that serves as a queue for incoming notifications. With an implementation like this the system can scale very well and we can serve big numbers of notifications per second, so it is ideal for a case of monitoring a big system. This implementation also allows to give priority to the jobs that will treat the more critical alerts.

When a notification is received, we pass it to a worker where it is parsed so we can get more information about its type and the service it refers to. Then we iterate over the specific alerts received. Depending on the SLA of the service and the alert we proceed to act. For example, maybe an alert shows that a VNF is consuming a lot of CPU resource. If the SLA requires that the service never run out of CPU resources, we have to take action in order to offer this service more CPU power.

Figure 17 shows the internal implementation of the notification manager. The notification is coming from the external Prometheus module and when the API receives it, it immediately sends it to the Redis queue which returns with the information that the notification has been received and that it is about to be processed.

When a notification is parsed in the worker, we get the information about the type of alerts, its importance, the service and the instances is concerns and other valuable info. Depending on the policies agreed in the service SLA we will choose how to react upon the notification. After

choosing the required action TeNOR will coordinate its deployment. More details about this process will be presented on the upcoming WP7 deliverables where the SESAME demo specifications and results will be presented.

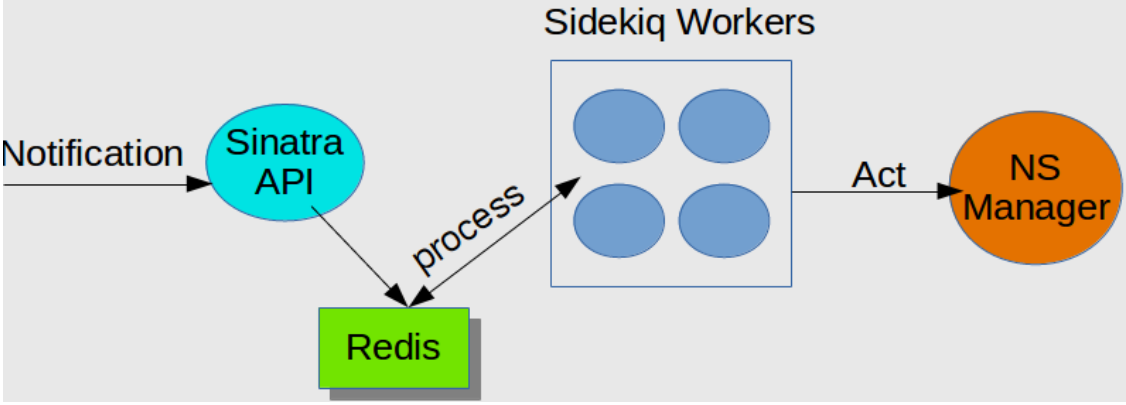

Figure 17: Internal Architecture of the Notification Manager

6 Orchestration of Light-weight Virtualised Network Functions

The centralised data-centres providing cloud infrastructure in the familiar guises of Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS) are based on well-established proprietary and open-source virtualisation technologies. The 5G development road-map shows a strong reliance on cloud-native technologies to address the expected 5G networks challenges including resource sharing, multi-tenancy, service scaling, isolation and elastic value addition. For 5G networks' core, the same data-centre architectures, virtualisation technologies and management solutions are expected to provide the necessary functionalities to fulfil the requirements of SDN-/NFV-*based* 5G network architecture. However, for the large set of distinct differences including resource availability constraints between the network core and edge makes those same set of virtualisation technologies (OpenStack, Amazon AWS, etc.) and management functions (including NFV orchestration) infeasible for the network edge infrastructure. In the SDN/NFV progression thus far, the network edge (i.e. the infrastructure located closest to the end-user) has not received the attention it warrants. The plethora of works carried out under the banners of Mobile Edge Computing, Fog Computing[16], and Cloudlets[17] etc., have demonstrated how bringing network and service functions closer to the edge will bring benefits to the end-users and service providers alike. To bring the benefits of SDN/NFV to the network edge, a lightweight solution to virtualisation and network function orchestration must be considered. Within the context of SESAME's heterogeneous, lightweight CESC nodes, a correspondingly lightweight management and network orchestration solution carries more weight than tinkering with solutions developed for resource affluent data centres.

In this context, we consider candidate cloud service and management technologies that take resource utilisation as an important consideration for service provisioning and utilise them for prototype SDN/NFV edge-cloud service provisioning. More specifically, we investigate container eco-system and relevant container management and orchestration technologies as the core drivers of the 5G network edge.

6.1 Containers as Light-weight Virtual Network Functions

In most of the works on SDN/NFV, the virtual network functions are developed and used as Virtual Machines hosted on established cloud infrastructures. While fully adequate for the data-centre oriented clouds, virtual machines and the required virtualisation technologies are not explicitly oriented towards having a small resource footprint thereby making them less adequate for resource constrained nodes such as SESAME CESCs. Containers provide an alternative to virtual machines and have different requirements in terms of virtualisation. Figure 18 gives an abstraction of the differences between containers and virtual machines based virtual network functions where the actual network function (indicated as App) sits on top of the stack. Apart from the difference in the number of layers stacked between the physical hardware at the bottom and the network application at the top, the stacked layers in the case of virtual machines are more resource intensive such as hypervisor and guest operating systems compared with container runtime. Moreover, the container stack provides the possibility to share (if required) certain application specific libraries among different applications. This makes the container-based virtualisation for SDN/NFV more adequate for a lightweight network edge. The increased efficiencies in the container stack do come with certain compromises on security

[16] For further information also see, *among others*: https://en.wikipedia.org/wiki/Fog_computing
[17] For further information also see: https://en.wikipedia.org/wiki/Cloudlet

and isolation, yet the advantages overweigh the disadvantages especially since access to the SESAME CESC clusters is not open.

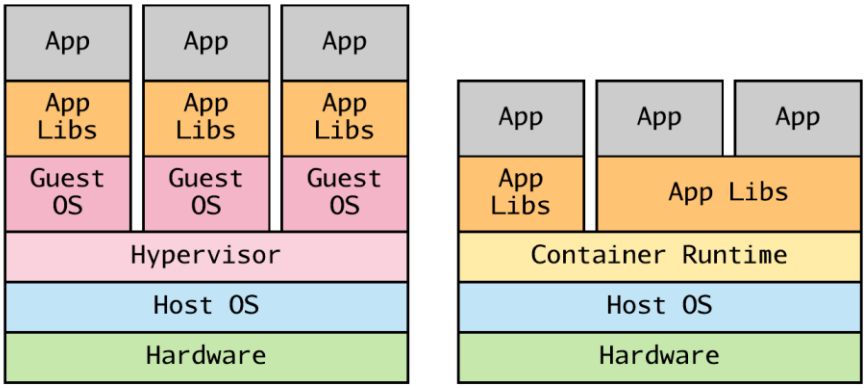

Figure 18: Architectural comparison between full VM (left) and containers (right)

6.1.1 The Concept of Dockers

Although Docker is one software container platform among others, it is by far the most widely used. The container virtualisation stack shown in Figure 18 maps directly to how Docker containers are developed/packaged and executed on any real hardware. Docker uses a Docker system daemon as container-runtime which runs with root privileges and manages the isolation and proper execution of Docker containers. Under the hood, the Docker daemon uses the standard Linux kernel features such as control groups and namespaces to control the resources consumed by containers and ensuring isolation among running containers. The workflow for creating a Virtual Network Function using Docker based software containers consists of defining the recipe for building the application inside a software container and packaging the required application libraries inside, resulting in a Docker container image. This image is entirely portable and regardless of the execution platform (except the Host OS kernel and physical CPU architecture) will function/execute exactly the same on all machines forming the CESC cluster as it would on the machine that developed the container image. This solves the problem of platform dependency completely and a container running a VNF application inside can be deployed on any machine/CESC forming the cluster. The workflow to compose an edge-service using Docker containers is somewhat similar to how VMs are tied together with a template file specifying interconnection and deployment preferences. Docker containers running different virtual network functions can be chained together across CESC cluster using overlay connectivity by defining a template for Docker services using common description/markup tools such as Yaml and Json. Moreover, for each deployed service, one can specify policies which, among others, allow/restrict targeted deployment nodes, manage resource utilisation and define security/access preferences.

6.2 Container Based VNF Orchestration

Containers are meant to be atomic in the service they provide. This implies that contrary to how one can install any number of applications inside a single VM and deploy it as a monolith application, containers are meant to only provide a single/atomic functionality. Although this is not a strictly imposed requirement for using container-based services, the proper use of container technology dictates granular application decomposition. This means that a complex application should be divided into small, lightweight parts with each functional part embedded in a separate container image. This has a number of advantages including separation of

concerns, better reliability and recovery mechanisms compared with VMs. However, this also implies that if done properly, a container based network service will require executing more VNF units (Containers) than VM based services. In this case, the need for container management and orchestration become evident. In essence, container orchestration refers to the automated deployment, coordination and management of container based services. Container orchestration tools are rapidly evolving for container-centric clustered infrastructure environments. Container orchestration tools provide the same set of functions associated/expected of orchestration tools developed for VM based cloud infrastructures. Additionally however, container orchestration tools are very good at managing scale as the sheer number of containers executed in a cloud is very high. As a container orchestration tool for deploying Dockers in the CESC prototype it was used Kubernetes, an open source container management and orchestration tool developed by Google [38]. Kubernetes automates the deployment of container based network services on top of a clustered infrastructure, scaling services to guarantee performance constraints and taking care of service availability.

6.3 A Light-weight Orchestration Method

The mobile edge caching prototype [39], [40]) was developed in such a way that three applications/services have to be deployed on the CESC platform: GTP encap/decap, Squid and OpenAirInterface eNodeB. The applications are packaged into three separate containers based on the instructions specified in a Dockerfile. (A Dockerfile is a text file with a set of instructions specified by the user needed to build an image). The image is built using a simple command-line instruction:

```
docker build  Dockerfile
```

The LTE small cell was implemented with an Ettus B210 Software Defined Radio [41] and the OpenAirInterface [42] software spanning the full protocol stack of an eNodeB that can support up to 3GPP Rel. 10. The Squid application is used for caching the web content from the UE associated to the eNodeB. The GTP encap/decap application takes care of performing a stateful termination and recreation of the GTP session between the eNodeB and SGW, and also redirects the traffic towards the caching application service.

By leveraging on the abstractions provided by Kubernetes, containerized applications are deployed as pods on each of the nodes through the following command line instruction:

```
Kubectl create -f  JSON-filename
```

On the other hand, the details about the active pods are obtained using the following line command:

6.3.1 Sample Code of a JSON Configuration File For eNodeB Pod Deployment

```
{
 "id":"eNB",
 "kind": "Pod",
 "apiVersion": "v1",
 "metadata": {
   "name": "enb",
   "labels": {
     "name": "eNB"
   }
 },

 "spec": {
   "hostNetwork": true,
   "containers": [
     {
       "name": "enb",
       "image": "oai_enodeb",
      "securityContext": {
       "privileged": true
      }
     }
   ],
   "nodeSelector": {
       "node1": "SCVNF"
   }
 }
}
```

In addition to the pods already discussed that provide the pillar components of the edge caching functionality, a monitoring service of the containers through a graphical user interface was also deployed using the following command:

```
Kubectl create -f kube-ui-svc.yaml -namespace=kube-system
```

6.3.2 Performance Metrics Evaluation

Performance metrics of deploying and orchestrating virtualized network functions by means of Dockers were obtained in an experimental setup that consists of three x86[18] machines with 1.9 GHz Intel core i5 (4th generation) dual core processor and 8GB of RAM. All the machines run on Ubuntu 16.04[19] (Xenial Xerus) with Linux kernel 4.8[20], Docker 17.06[21] and Kubernetes 1.7[22]. All Docker containers use Ubuntu 16.04 as the base image.

[18] x86 is a family of backward compatible instruction set architectures based on the Intel 8086 CPU and its Intel 8088 variant. More relevant information can be found, *inter-alia*, at: *https://en.wikipedia.org/wiki/X86.*

[19] For more details see: *http://releases.ubuntu.com/16.04/*

[20] Also see: *https://kernelnewbies.org/Linux_4.8*

[21] Also see: *https://docs.docker.com/enterprise/17.06/*

Table 9: Experiments comparing Docker implementation of network functions with standard virtual machines

	Description
Experiment #1	It focuses on the difference in response times the user gets when served by the Squid service VNF, instead of directly being served by the origin Web server. The comparison is shown in Figure 19, which is obtained by making a web request using Curl[23] (a command-line tool) to five popular web pages, from the LTE UE terminal located about 2 meters to the serving eNodeB. The measurements have been repeated for 10 consecutive times. The difference in average RTT is significantly lower with caching, mainly because of the evasion of the core network and web server processing delay in the backhaul.
Experiment #2	It is performed to profile different types of overheads for running applications in the containerised platform and virtualised platform, compared to the native, non-virtualized platform. Storage overhead (see Figure 20) by making 1000 web requests from the UE after it is associated to the eNodeB with three resource usage profiles: Native Containerized (Docker) and Virtualized (VirtualBox). Other performance indicators that were measured are CPU consumption (see Figure 21), memory consumption (see Figure 22), boot-up time (see Figure 23). The second comparison that can be drawn is the difference in size for a similar image as shown Figure 19. The image containing Squid on top of Ubuntu weighs 3.95GB in VirtualBox, and only 358MB in a Docker. The significant reduction in size is mainly due to the removal of kernel, peripheral drivers, etc., that are unnecessary in a containerized environment, since the Docker image has access to it directly from the host kernel. The next performance criteria consist in the CPU utilisation. Raw single core utilisation by Squid application is monitored for the duration of the experiment using TOP and PERF[24] tools. The Squid application server when running in VM requires 16.8% of host CPU, whereas Docker requires only 1.25% of host CPU as seen in Figure 21. Pinning each virtual CPU to a physical CPU further reduces the overall CPU utilisation to 1.05%.

[22] Also see: *http://blog.kubernetes.io/2017/06/kubernetes-1.7-security-hardening-stateful-application-extensibility-updates.html*
[23] See: *https://github.com/curl/curl*
[24] See: *https://perf.wiki.kernel.org/index.php/Tutorial*

	Memory (RAM) utilisation is limited to a maximum of 2GB in both Docker and VirtualBox for its operation. In the case of VM, the allocated memory is completely utilised for running the Squid application, whereas Docker only utilises 93.66 MB as shown in Figure 22. Another interesting measurement is the time needed to boot the system. In case of VirtualBox, it took 21 seconds for the operating system to start, whereas in case of a Docker it took only 3.2 seconds as shown in Figure 23.

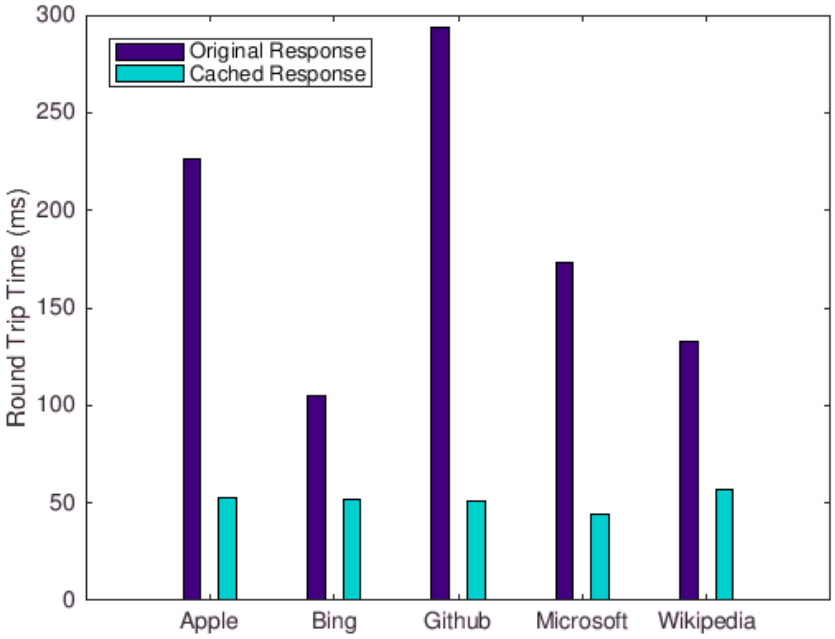

Figure 19: Measured round-trip time

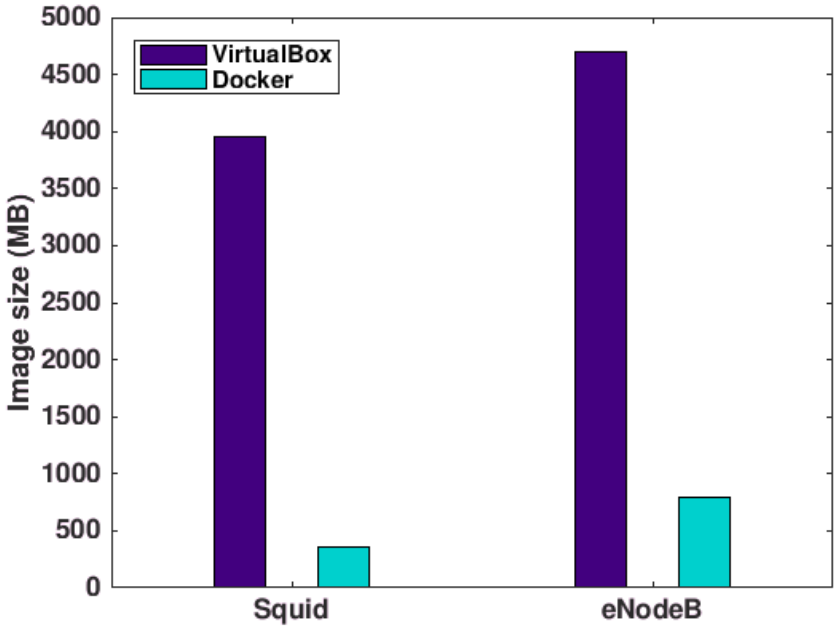

Figure 20: Measured storage space

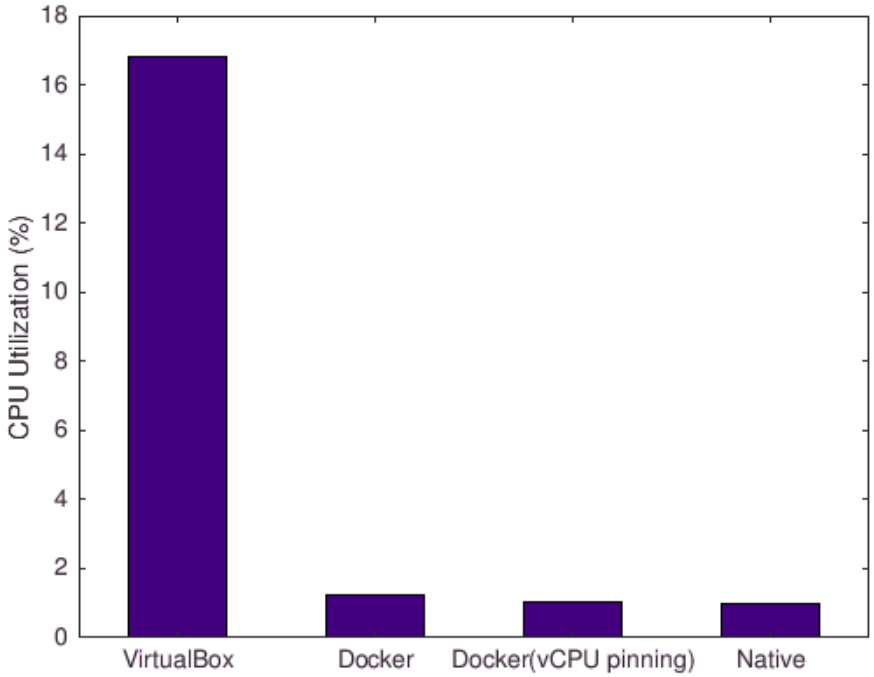

Figure 21: Measured CPU utilisation

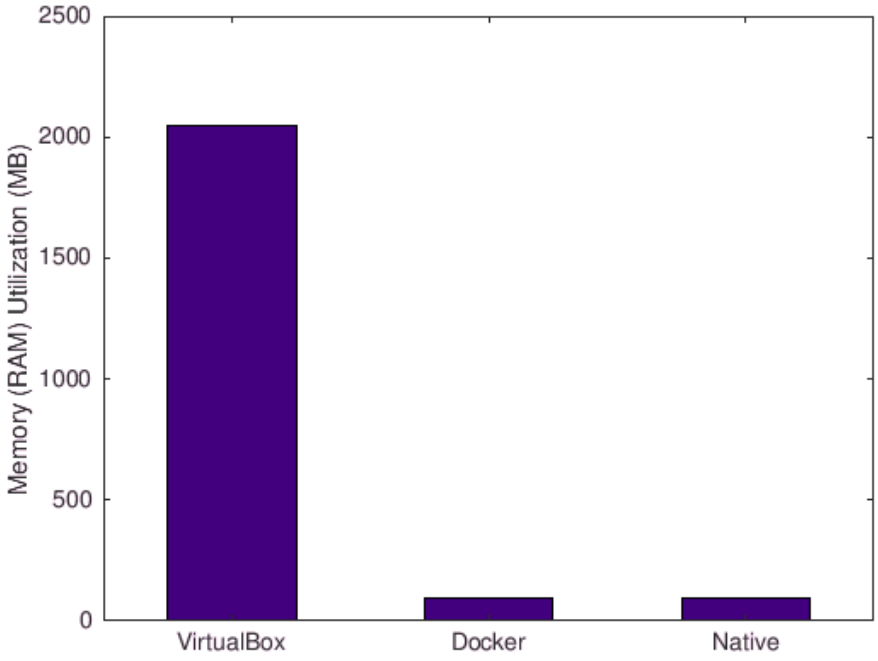

Figure 22: Measured RAM utilisation

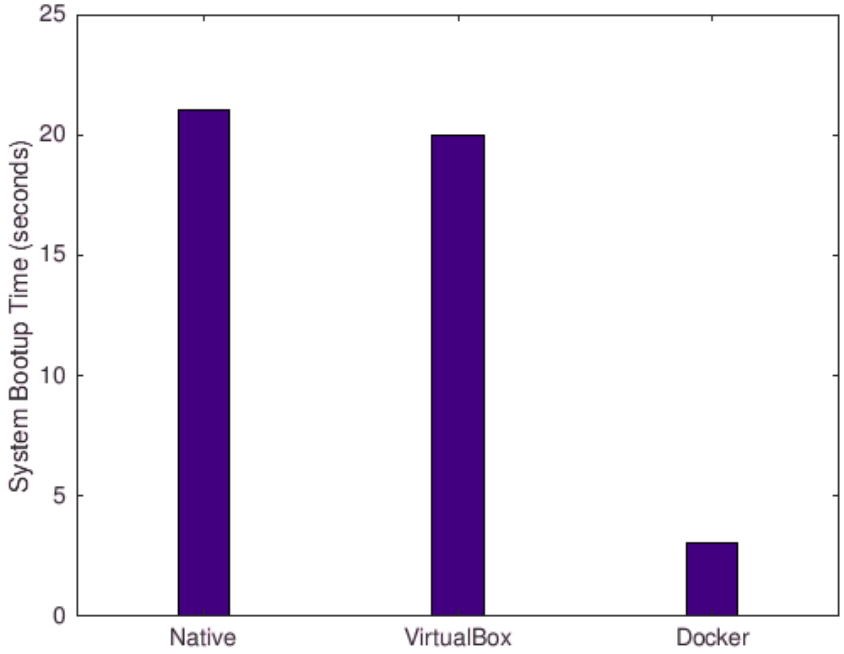

Figure 23: Measured system boot-up time

7 Conclusions

This deliverable describes the latest NFVO related activities done in the context of the SESAME WP6. In particular, Task 6.3 "*Service Management, Monitoring and Execution*". In particular, in section 2 we investigated the benefits of joint radio – NFV resource orchestration in 5G systems. Our study proved that how a holistic joint orchestration will improve the user experience. In section 3, we introduced QoS assurance loop in the context of SESAME project. This concept is one of the main objectives of SESAME NFVO and has been discussed with extensive details regarding all required steps, i.e. monitoring, decision-making process and reaction mechanism. Section 4 focused on advanced decision-making processes with the help of cognitive processes. Section 5 detailed the SESAME NFVO prototype developments and provided links to the SESAME NFVO on the open source repositories. Lightweight virtualisation technologies as an emerging solution have been reviewed in section 6, accompained by study results that confirm their benefits.

This document together with the SESAME NFVO prototype available on open source repositories form the final input from WP6 to WP7, where the final SESAME demo integration happens.

8 References

[1] Ericsson White Paper, "Cloud-RAN – the benefits of virtualization, centralization and coordination", (2015). Online [available at: http://www.ericsson.com/res/docs/whitepapers/wp-cloud-ran.pdf]

[2] I. Giannoulakis, et. *al.*(2016). Enabling Technologies and Benefits of Multi-Tenant Multi-Service 5G Small Cells. In Proceedings of the EuCNC 2016 Conference. Athens, Greece, June 27-30, 2016.

[3] S. Peng, et. *al*. (2017). QoS-Oriented Mobile Edge Service Management Leveraging SDN and NFV., *Mobile Information Systems*, *Vol.2017*, ID 3961689, 14 pages.

[4] H2020/5G-PPP SESAME project, Deliverable D6.1 [Online], available at: http://www.sesame-h2020-5g-ppp.eu/

[5] "Network Functions Virtualization (NFV); Management and Orchestration," ETSI GS NFV-MAN V1.1.1, December 2014.

[6] T. Lin, Z. Zhou, M. Tornatore et *al.*(2015). Optimal Network Function Virtualization Realizing End-to-End Request. In Proceedings of the 2015IEEE GLOBECOM Conference. IEEE, San Diego, December 06-10, 2015.

[7] M. Savi et *al.*(2016). Impact of Processing Costs on Service Chain Placement in Network Functions Virtualization. In Proceedings of the 2015 IEEE NFV-SDN Conference. San Francisco, November 18-21, 2016.

[8] M. Savi, M. Tornatore and G. Verticale (2016). To distribute or not to distribute? Impact of latency on Virtual Network Function Distribution at the edge of FMC Networks. In Proceedings of the 18th International Conference on Transparent Optical Networks (ICTON)N. IEEE, Trento, Italy, July 10-14, 2016.

[9] Net2Plan – The open-source network planner. [Online] Available: http://www.net2plan.com [Last accessed: July 2017]

[10] P. Pavon and J.-L. Izquierdo-Zaragoza (2015, September). Net2Plan: An Open Source Network Planning Tool for Bridging the Gap between Academia and Industry. *IEEE Network, 29*(5), pp.90-96.

[11] Spanish Ministry of Energy, Tourism and Digital Agenda – Information about Radioelectric Installations and Radiation Exposure Levels. [Online] Available: https://geoportal.minetur.gob.es/VCTEL/vcne.do [Last accessed: July 2017].

[12] M. Savi et *al.*, "Impact of processing-resource sharing on the placement of virtual network functions," Proceedings of IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN), 2015.

[13] C.-C. Lin (2013). *Handover Mechanisms in 3GPP Long Term Evolution (LTE)*, Ph.D. dissertation. University of Technology, Sydney, Australia, 2013.

[14] 3GPP TR25.814 version 7.1.0: Physical Layer Aspects for Evolved Universal Terrestrial Radio Access (UTRA) (Release 7),September 2006

[15] ETSI (2014). Mobile-edge computing: Introductory technical white paper.

[16] 3GPP TS 32.425: Performance Management (PM); Performance measurements Evolved Universal Terrestrial Radio Access Network (E-UTRAN), V.10.6.

[17] 3GPP TS 32.435: Telecommunication management – Performance measurement – XML file format definition v.13.

[18] 3GPP TS 32.436: Telecommunication management – Performance measurement – ASN.1 file format definition v.13.

[19] 3GPP TS 32.342: Telecommunication management – File Transfer (FT) IRP – Information Service (IS), v13..

[20] "NFV Management and Orchestration - An Overview", GS NFV-MAN 001 V1.1.1, European Telecommunications Standards Institute (ETSI), 2014.

[21] *https://www.opendaylight.org/* .

[22] ODL OpenFlow Plugin Statistics, *https://wiki.opendaylight.org/view/OpenDaylight_OpenFlow_Plugin:Statistics* .

[23] https://www.slac.stanford.edu/xorg/nmtf/nmtf-tools.html

[24] https://skydive-project.github.io/skydive/getting-started/docker

[25] *www.sflow.org/sFlowOverview.pdf* .

[26] *http://www.inmon.com/products/sFlowTrend.php#download* .

[27] *https://github.com/icclab/netflogi* .

[28] *https://github.com/icclab/netfloc-prometheus-exporter* .

[29] P. S. Khodashenas, et. *al.*, Service Provisioning and Pricing Methods in a Multi-Tenant Cloud Enabled RAN, In *Proceedings of the IEEE 2016 Conference on Standards for Communications and Networking (CSCN 2016)*. IEEE, Berlin, Germany, October 31 November 02, 2016.

[30] *https://github.com/T-NOVA/TeNOR* .

[31] *http://www.t-nova.eu/*

[32] *https://prometheus.io/* .

[33] *https://github.com/prometheus* .

[34] *https://stash.i2cat.net/projects/TENOR/repos/tenor/browse* .

[35] *https://github.com/sinatra/sinatra* .

[36] *https://github.com/mperham/sidekiq* .

[37] *https://redis.io/* .

[38] Kubernetes, available at: *https://kubernetes.io/*

[39] A. Betzler (editor), "Framework of a Distributed Network Management System Capable to Host and Run Self-X Functionalities," Deliverable D3.3 of 5G-PPP SESAME project, June 2017.

[40] A. Whitehead (editor), "CESC Small Cell Prototype and PoC," Deliverable D3.4 of 5G-PPP SESAME Project, June 2017.

[41] Universal Software Radio Peripheral (USRP) B210, available at: *https://www.ettus.com/product/details/UB210-KIT*

[42] Open Air Interface, available at: *http://www.openairinterface.org/*

[43] I. Trajkovska (editor), "Service Management and Orchestration functions, including VNF models - Final", Deliverable D6.3 of SESAME, September 2017.

[44] ETSI GS NFV-MAN 001 v1.1.1: Network Functions Virtualisation (NFV); Management and Orchestration, December 2014.

[45] J. Pérez-Romero, O. Sallent, R. Ferrus, R. Agustí (2017, May). Admission Control for Multi-tenant Radio Access Networks. In Proceedings of the 2017 IEEE International Conference on Communications Workshops (ICC Workshops). IEEE, Paris, France, May 21-25, 2017.

[46] 3GPP TS 32.425 v14.1.0 "Performance Management (PM); Performance measurements Evolved Universal Terrestrial Radio Access Network (E-UTRAN)", December, 2016

[47] J. Sánchez-González, J. Pérez-Romero, O. Sallent, "A Rule-Based Solution Search Methodology for Self-Optimization in Cellular Networks", IEEE Comm. Letters, Vol. 18, No. 12, 2014, pp. 2189-2192.

[48] 3GPP TS 32.522 v11.7.0: Self-Organizing Networks (SON) Policy Network Resource Model (NRM) Integration Reference Point (IRP); Information Service (IS) (Release 11), September, 2013.

[49] SCF 191.08.02: Multi-operator and neutral host small cells, December, 2016.

[50] SCF 082.07.03: nFAPI and FAPI specification, June, 2016.

[51] SCF 159.06.02: Small Cell Virtualization: Functional Splits and Use Cases, January, 2016.

[52] A. Whitehead (editor), "CESC Prototype design specifications and initial studies on Self-X and virtualization aspects", Deliverable D3.1 of the SESAME project, June 2016.

[53] C.J.C. Burges (1998, June). A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery, 2*(2), pp.121-167.

[54] V. Feng, S.Y. Chang, (2012, March). Determination of Wireless Networks Parameters through Parallel Hierarchical Support Vector Machines. *IEEE Transactions on Parallel and Distributed Systems, 23*(3) pp.505-512.

[55] T. Kohonen (1990, September). The Self-Organizing Map. *IEEE Proceedings, 78*(9), pp.1464-1480.

[56] T. Kuremoto, et *al.* (2010). Parameterless-Growing-SOM and its application to a voice instruction learning. *Journal of Robotics, Volume 2010,* Article ID 307293, pp.1-9.

[57] N.I. Sapankevych, R. Sankar (2009, May). Time Series Prediction. Using Support Vector Machines: a Survey. *IEEE Computational Intelligence Magazine,* 4(2), pp.24-38.

[58] P. Bermolen, D. Rossi (2009). Support vector regression for link load prediction. *Computer Networks, 53*(2), pp.191-201

[59] W. Jiewu, F. Wentao, H. Chunjing, Z. Xing. User traffic collection and prediction in cellular networks: architecture, platform and case study. In *Proceedings of 2014 4th International Conference on Network Infrastructure and Digital Content (IC-NIDC 2014)*. Beijing, China, September 19-21, 2014.

[60] X. Wang, W. Gu, D. Ziebelin, H. Hamilton, (2010, November). An ontology-based framework for geospatial clustering. *International Journal of Geographical Information Science, 24*(11),pp.1601-1630.

[61] J. Han, M. Kamber, A.K.H. Tung, (2001). Spatial Clustering Methods in Data Mining: A Survey Geographic Data Mining and Knowledge Discovery, Research Monographs in GIS, Taylor and Francis, pp.1-29..

[62] T. Schreck et *al.* (2008). Visual Cluster Analysis of Trajectory Data With Interactive Kohonen Maps. In Proceedings of the 2008 IEEE Symposium on Visual Analytics Science and Technology (VAST'08). Columbus, Ohio, USA. October 19-24, 2008.

[63] E. Masciari (2009). A Complete Framework for Clustering Trajectories. In *Proceedings of*

the 21st IEEE International Conference on Tools with Artificial Intelligence (ICTAI'09). Newark, USA, November 02-04, 2009.

[64] J-G. Lee, J. Han, K-Y. Whang (20017). Trajectory Clustering: A Partition-and-Group Framework.. In *Proceedings of the ACM SIGMOD 2007*. ACM,, Beijing, China, June 11-14, 2007.

[65] G. Andrienko, et *al.* (2009). Interactive Visual Clustering of Large Collections of Trajectories. In *Proceedings of the 2009 IEEE Symposium on Visual Analytics Science and Technology*. Atlantic City, New Jersey, USA, October 12-13, 2009.

[66] A.C. Gatrell, T.C. Bailey, P.J. Diggle, B.S. Rowlingson (1996). Spatial point pattern analysis and its application in geographical epidemiology. *Transactions of the Institute of British Geographers*, *21*(1), pp.256-274.

[67] G.R. Terrell, D.W. Scott 1992, September). Variable Kernel Density Estimation. *The Annals of Statistics, 20*(3), pp.1236-1265.