MCN
Mobile Cloud Networking

FUTURE COMMUNICATION ARCHITECTURE FOR MOBILE CLOUD SERVICES

Acronym: Mobile Cloud Networking
Project No: 318109
Integrated Project
FP7-ICT-2011-8
Duration: 2012/11/01-2015/10/31

D6.5 Final Report on Testbeds, Experimentation, and Evaluation

Type	Report
Deliverable No:	6.5
Workpackage:	6
Leading partner:	ONE
Author(s):	Luis Cordeiro (Editor), list of authors overleaf
Dissemination level:	Public
Status:	Final
Date:	25 August 2016
Version:	1.1

List of Authors and Reviewers

Authors	Partner
Aikaterini Trilyraki	EURE
Andy Edmonds	ZHAW
Bruno Marques	ONE
Bruno Sousa	ONE
David Palma	ONE
Dr. Thomas Magedanz	FOKUS
Eryk Schiller	UBERN
Giuseppe Carella	TUB
Lars Grebe	FOKUS
Lorenzo Tomasini	TUB
Luca Foschini	UNIBO
Luís Cordeiro	ONE
Marius Corici	FOKUS
Mohammadjavad Valipoor	STT
Navid Nikaen	EURE
Patrício Batista	ONE
Philipp Emanuel Kuhnz	TUB
Santiago Ruiz	STT
Rakesh Varudu	FOKUS
Thomas Bohnert	ZHAW
Zarrar Yousaf	NEC

Reviewers	Partner
Paulo Simões	ONE
Paolo Bellavista	UNIBO

Versioning and contribution history

Version	Description	Contributors
0.1	First Draft Version with evaluation section structure	ONE
0.2	Version with integrated service evaluations	ONE, STT, TUB
0.3	Version with integrated service evaluations and sections closed	ONE
0.4	Version with integrated service evaluations of IMS	ONE, TUB
0.5	First Review	ONE, UNIBO
0.6	Integrated first review comments	ONE, STT,EURE
0.7	Pre Final version	ONE, TUB
0.8	Integrate further contributions and perform editorial changes	ONE,UNIBO
1.0	Final editing and last content integration	ONE
1.1	Minor updates requested after the final review	ONE

Executive Summary

This document reports the work conducted in the scope of WP6 during the extension period of the Mobile Cloud Networking (MCN) project (M37-M42). The main objectives of this work were i) the extensive and methodologically-structured experimental evaluation of articulated end-to-end Proof-of-Concept (PoC) services and of their primary composing services (individual services); and ii) the application of the proposed evaluation methodology (see Deliverable D3.5) in realistic and articulated cases of practical interest, also in order to assess the effectiveness and real applicability of the methodology itself.

In particular, the reported outcomes are presented in the evaluation section by outlining the functional prototype PoCs behaviours in relation to the targeted end-to-end scenario, which has been significantly modified in the extension period (M37-M42) to include Radio Access Networks in different testbeds. The evaluation section not only reports such evaluation results in the end-to-end scenario, but also outlines the performance of each individual service that contributes to the end-to-end composed service.

The key highlights of this evaluation were:

- The evaluation of the DSS and IMS PoCs demonstrates that the impact at the platform and infrastructure levels is low in terms of resources, by requiring mainly CPU, in particular for the deployment and provisioning phases involving the Service Instance Components.
- The DSS evaluation results demonstrate that the introduced enhancements for provisioning and supporting fault tolerance do not affect the end-to-end perceived quality of players, as well as the support for RAVA allows achieving a good level of testbed resource consumption efficiency, without affecting the availability of migrated components.
- The proposed evaluation methodology allowed us to quantify objectively the performance of IMSaaS SICs in different deployment situations, including hardware and virtual, where the latter has demonstrated to bring to a minimal difference in terms of delay due to the additional entities involved in the provisioning process.
- The enhancements performed in the RAN OAI by splitting the BBU and RRH components lead to the need of additional resources, but with proved and relevant flexibility in terms of deployment options for operators that aim to adopt C-RAN in next generation networking solutions.

In summary, through the evaluation methodology specified and adopted in the MCN extension period, the performance of different services composing the targeted end-to-end scenarios was characterized in terms of several diverse dimensions including: System KPIs, Resource KPIs, and Cloud-Native KPIs. Such extensive experimentation has considered different workload tests that are commonly applied by telco operators and vendors to fully specify the performance of their products. In this context, the achieved results demonstrate the feasibility and efficiency of the solutions prototyped within the MCN project framework and provide useful insights towards the adoption (and efficient tuning) of cloud-native features by emerging and state-of-the-art services like RAN.

Table of Contents

Table of Figures

Table of Tables

Acronyms

Acronyms	Description
AAA	Authentication, Authorization and Accounting
API	Application Programming Interface
BBU	Base Band Units
CC	Cloud Controller
CDN	Content Delivery Network
CLI	Command Line Interface
COTS	Commercial Of The Shelf
CPU	Central Processing Unit
C-RAN	Cloud Radio Access Network
DB	Database
DHCP	Dynamic Host Configuration Protocol
DMM	Distributed Mobility Management
DNS	Domain Name System
DSN	Digital Signage Network
DSS	Digital Signage System
EEU	Enterprise End-User
END-TO-END	End-to-end
EPC	Evolved Packet Core
EU	End-User
HDD	Hard Disk Drive
HSS	Home Subscriber Server
HTTPS	Hypertext Transfer Protocol Secure
ICN	Information Centric Network
IMS	IP Multimedia Subsystem
IP	Internet Protocol

iSCSI	Internet Small Computer System Interface
IT	Information Technology
ITG	Infrastructure Template Graph
KVM	Kernel-based Virtual Machine
LB	Load Balancer
LTE	Long Term Evolution
MCN	MobileCloud Networking
MCR	Main Content Repository
MPLS	Multi-Protocol Label Switching
NCP	Network Connectivity Provider
NFS	Network File System
NUMA	Non-Uniform Memory Access
OAI	OpenAir Interface
OCCI	Open Cloud Computing Interface
OTT	Over-The-Top
PoC	Proof of Concept
PoP	Point of Presence
PXE	Pre-Execution Environment
QCOW	QEMU Copy On Write
QEMU	Quick Emulator
QoS	Quality of Service
RAM	Random Access Memory
RAN	Radio Access Network
RCB	Rating Charging Billing
REST	Representational State Transfer
RRH	Remote Radio Head
SAN	Storage Area Network
SCP	Secure Copy Protocol

SDK	Service Development Kit
SFTP	SSH File Transport Protocol
SIC	Service Instance Component
SIMD	Single Instruction Multiple Data
SIP	Session Initiation Protocol
SLA	Service-Level Agreement
SM	Service Manager
SO	Service Orchestrator
SSD	Solid State Drive
SSH	Secure Shell
STG	Service Template Graph
VLAN	Virtual Local Area Network
VM	Virtual machine
WP	Work Package
XaaS	X as a Service

1 Introduction

This document reports on the MCN work on prototype integration, experimentation, demonstration, and evaluation carried out during the extension period of the MCN project (M37-M42). It complements the core bulk of results already provided in Deliverable D6.4 (Final Report on Testbeds, Experimentation, and Evaluation, M36) and builds upon the evaluation methodology provided in Deliverable D3.5 (Final Report on Component Design and Implementation, M42), as well as on the outcomes of previous work packages.

1.1 Motivation, objectives and scope

In line with and as specified in the MCN Description of Work, this report covers the following activities:

- Extended experimentation and evaluation activities (Task T6.5 Extension), according to the evaluation methodology for the integrated MCN services. This deliverable reports the evaluation activities carried out on the extension period M37-M42.
- Experimentation and evaluation activities of the individual services as per the evaluation methodology.
- Testbed maintenance and setup for the evaluation activities accomplished during the extension period.

1.2 Structure of the document

The core of the deliverable consists of three main sections, according to the logical sequence of activities undertaken by WP6 in the extension period:

- Section 2 provides a description of the MCN testbed infrastructure which was used during the extension period.
- Section 3 provides a description of the PoC evaluation process performed according to the evaluation methodology originally proposed for this extension period.
- Section 4 describes and discusses the results of the experimentation and evaluation work.

Section 5 finalizes the document with an overall discussion of the MCN PoC experimentation and evaluation.

1.3 Relation with other deliverables

This document relates with and builds upon many of the previous project deliverables by including evaluation and experimentation results about individual services, previously described in other deliverables, and their composition in an end-to-end scenario. In addition to previous deliverables, the following "extension deliverables" should be explicitly mentioned due to their close relation with D6.5:

- D3.5 Evaluation Methodology (M42).
- D4.6 Mobile Network Cloud Software Components and Report (M42).
- D5.5 Evaluation of Mobile Platform, IMSaaS and DSN (M42).

Along the document we try to keep balance between conciseness (avoiding repeating content from those deliverables) and readability (summarizing some of the key assumptions on which D6.5 builds upon). Nonetheless, we do assume the reader is already familiar with those previous documents.

Deliverable D6.4 (Final Report on Testbeds, Experimentation, and Evaluation, M36) should also be explicitly mentioned here, though for different reasons. D6.4 already presented and discussed the core bulk of MCN evaluation results (conducted up to M36), and this new deliverable was written as an extension to those results, focusing only on the changes of testbeds and methodologies that took place during the extension period and on the new evaluation results achieved during the last 6 months (M37-M42). It is strongly assumed the reader already knows D6.4, and for sake of conciseness several descriptions and discussions are not repeated (or even summarized) again in D6.5.

2 MCN Integrated Testbed

This section provides the primary updates that were operated over the MCN Integrated Testbed in order to make it suitable and effective for the extensive evaluations of the targeted individual and end-to-end services, including the infrastructure testbeds and the platform testbed. The open source platforms used in the WP6 evaluation activities of the extension period have been based on OpenStack and OpenShift. Let us recall that the overall and more comprehensive Integrated Testbed of MCN up to M36 included more interconnected testbeds, as fully documented in Deliverable D6.4.

2.1 Integrated Testbed

This sub-section details the testbeds employed in the WP6 evaluation activities. For additional information regarding the testbeds, please refer to Deliverable D6.4.

Table 1 – Key MCN extension testbeds

Testbed name	Location	Responsible Partner
EURE	Sophia Antipolis, France	EURE
Bart	Zurich, Switzerland	ZHAW
Fokus	Berlin, Germany	FhG

During the MCN extension period the experimentation and evaluation activities were performed in the EURE, Bart and Fokus testbeds, as summarized in Table 1. The number of testbeds is lower when compared to the MCN M36 period, due to the fact that partners providing testbeds were not participating in the extension activities (e.g. CloudSigma, Intel), and others participating had a limited number of resources to maintain the testbeds running in the extension period (such as UBern).

The testbeds in the extension are based on OpenStack full deployments and are interconnected as multiple regions, as identified in Table 2. Each region has support for the OpenStack Heat functionality to enable the basic resources for each service instance.

Table 2 – MCN extension testbed platforms

Testbed name	IaaS	Other	Region name
EURE	OpenStack, LXC	None	EURE
Bart	OpenStack, KVM	Identity Provider	RegionOne
Fokus	OpenStack, KVM	None	FOKUS

The associated Service Managers (SM) are located in an OpenStack testbed located in Zurich, Switzerland, and are based on the Kilo version of OpenStack. The Service Orchestrators (SO) are deployed on top of OpenShift, which is located in the Amazon Web Services EC2 testbed.

As already stated, it should be noticed that the number of testbeds is lower than the one reported in M36. The testbed of UBern was not available during the extension due to the fact that UBern exhausted their resources and were unable to maintain the testbed operational. They only concentrated on supporting

activities in T3.5. Also, the number of partners actively participating in the extension was small, hence 3 testbeds provided enough capacity to accommodate for project needs.

2.1.1 Testbeds specifications

2.1.1.1 Bart (ZHAW)

This testbed has not been substantially modified for the extension evaluation. Thus the testbed runs the basic OpenStack services, based on Kilo version. All the installation steps and configuration are reported in Section 3.2.1 of Deliverable D6.4.

The only modifications performed during the extension period address mainly the support for SCTP communications, which requires compute nodes to load SCTP modules in the Linux kernel, as well as associated updates in the firewall rules to allow SCTP signalling.

2.1.1.2 Eurecom (EURE)

Eurecom testbed is an OpenStack (LXC based) deployment for the radio access network service (RANaaS). As described in D3.5 (Section 4) the architecture of the RAN was extended resulting in a new version of OAI eNB capable to interface via Ethernet a Remote Radio Head GateWay. In order to incorporate these changes to the MCN framework the single eNB LXC was replaced by two LXCs, one for OAI RRH GW and one for the altered version of OAI eNB in which we will refer to as the OAI BBU. In addition, a regular virtual network of OpenStack was created to realize the communication between OAI RRH GW and OAI BBU, namely the FH network. Finally, the appropriate modifications were made so that Openstack Heat is able to configure the RRH GW and BBU VMs on-the-fly. For more details regarding the content of this paragraph, see D3.5 (Sections 4.1.2 and 4.1.3).

The hardware/radio set up remains the same as described in D6.4 (Section 3.2.1). At this point, we have to note that although the OAI RRH GW is deployed on the same infrastructure (i.e., physical server) as the OAI BBU, it is completely independent from it and therefore it could be deployed on a separate/remote server as long as the frame/subframe timing and HARQ deadlines are met. The latter depends on the distance between the BBU and RRH, and given the speed of light in fiber (200 m/us) a maximum distance of 15 Km is possible to be achieved.

Closing, we would like to note that the hardware and software platforms are provided by EURECOM/OpenAirInterface, while the OpenStack installation was mainly performed by the CDS group of the University of Bern. This OpenStack installation is responsible for region EURE.

2.1.1.3 Fokus (FOKUS)

An OpenStack-based testbed to support the Radio Access Network service is installed at Fraunhofer Fokus, Berlin. The testbed consists of a radio network built using the software and hardware components developed within MCN by Eurecom. The LTE-compliant base station (eNodeB) is setup by installing USRP B210, i.e., a software-defined radio support for OpenAirInterface on a Linux-based X86 machine. The testbed is a mirrored replica of the Eurecom OpenStack-based radio access network testbed.

The Linux machine on which LTE-compliant eNodeBs are built is powered by an Intel Xeon E5-1620 v2, 8 core, running at 3.7 GHz, with 16GB RAM and 500GB HDD. It uses a USRP B210 board that supports a two-channel USRP device with continuous RF coverage from 70 MHz to 6 GHz. It features a full duplex MIMO with 2 Tx and 2 Rx operations. The setup consists of a duplexer and an antenna for

the 2.5 GHz spectrum range. The radio setup includes commercial LTE-enabled USB dongles (Huawei E3276) and LTE-supported smartphones (Galaxy S4 Gt 19505), which can be used to assess and validate the service provided by the 5G service testbed.

The setup runs as a complete single-host all-in-one OpenStack. In particular, OpenStack uses the Linux containers (LXC) to optimize the performance for the radio network service. The version of OpenStack used is Juno and is running on a Ubuntu 14.04.02 with low-latency kernel version 3.17.1-031701. The employed setup uses Nova to manage and deploy virtualized instances on Linux containers, Glance to manage image services, Neutron to handle networking, and Heat for orchestration. In addition, it includes the protocol support for TCP, UDP, and SCTP communication. By delving into finer details, the OpenStack setup consists of a controller/compute node, a virtual router, and a virtual machine providing RAN as a service, accessible by floating IPs 193.175.132.160, 193.175.132.161, and 193.175.132.162 respectively. This testbed has been connected with Bart as Compute Node and used as different region (region_name=Fokus).

Additionally, the Fokus labs have provided additional HW resources for executing the standalone IMS scenarios. On the one hand, these additional resources include two instances of a Lenovo ThinkCentre M93 with 8 cores (Intel(R) Core(TM) i7-4785T CPU @ 2.20GHz) and 16GB of RAM (SODIMM DDR3 Synchronous 1600 MHz (0.6 ns)). These two instances have been used for the scenarios [3-6] of the standalone IMS, in order to compare the performance results of a virtualized IMS with the hardware based one. On the other hand, they comprise one instance of an Ubuntu Orange Box providing a multi-compute node environment. In particular, the Orange Box provides 10x Intel NUCs, specifically, the Ivy Bridge D53427RKE model. Each Intel NUC contains:

- i5-3427U CPU;
- Intel HD Graphics 4000;
- 16GB of DDR3 RAM;
- 120GB SSD root disk.

The Orange Box has been used for the scenario 2 of the standalone IMS (see the following Section 3.3) in order to evaluate the performance results of the virtualized IMS in auto scaling and fault management scenarios.

Let us finally recall that this testbed was not used and present earlier, and has been originally added during the extension period.

2.1.2 Multi-Region Keystone

To enable the users to use their credentials in the whole geographically distributed MCN testbeds, a common Keystone service is used in all cases, and geographical regions are represented as Keystone regions. The OpenStack services offered in each region have their endpoint registered in the corresponding Keystone region.

All the configuration steps reported in D6.4 Section 3.2.2 have been performed for the Fokus testbed to enable the Fokus region. Figure 1 illustrates a screen capture of the OpenStack Dashboard Horizon with the MCN regions, including the Fokus region made available only in the extension period.

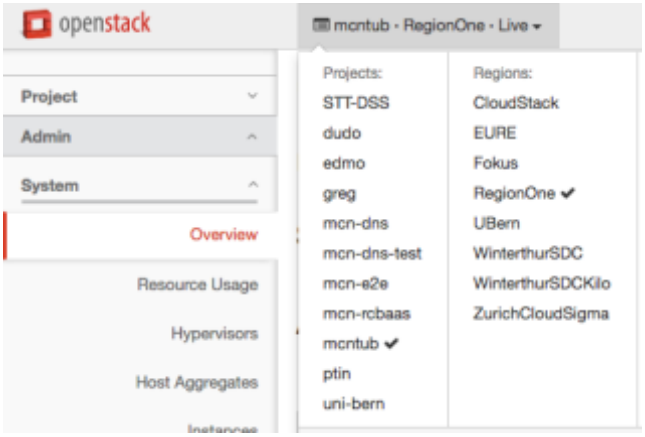


Figure 1 – The OpenStack Dashboard with MCN extension regions

2.2 OpenStack Infrastructure as a Service

The current deployed version of OpenStack is Kilo[1]. The installation in the Fokus region was performed already with the Kilo version before the extension period, as stated previously. No modifications were performed to the OpenStack version provided by the community, with the notable exception of the part for software-defined networking, as detailed in D6.4 (Section 3.3).

The SCTP support also required new and extended configurations, namely in terms of enabling the SCTP support by loading the respective modules in the controller and network nodes of OpenStack, as well as configuring the respective firewall rules.

In Bart, the region where most of the services are deployed and working as the identity provider, OpenStack is configured in five nodes, according to the roles identified in Table 3.

Table 3 – Bart OpenStack nodes

Node name	Roles of Nodes	Short Name
bart.cloudcomplab.ch	Controller, Network and Compute node	B0
bart-node-1.cloudcomplab.ch	Compute node	B1
bart-node-3.cloudcomplab.ch	Compute node	B3
bart-node-4.cloudcomplab.ch	Compute node	B4

[1] https://www.openstack.org/software/kilo/

2.3 OpenShift Platform-as-a-Service

OpenShift v3 was employed in the evaluation activities of WP6. Among the primary motivations of this adoption, this version, based on Docker containers and Kubernetes, proved to be more efficient when instantiating Service Orchestrators (SO), as extensively reported in D6.4 (Appendix A).

No primary configuration changes were performed regarding OpenShift to operate as a platform. During the extension evaluation work, OpenShift was deployed in the Amazon Web Services cloud testbed to further evaluate the interoperability of the MCN architecture with different cloud solutions, e.g., commercial public cloud providers. The configuration actions associated with this deployment change, i.e., moving OpenShift from Bart to Amazon Web Services, required modifications in the firewall rules of the Eurecom, Fokus, and Bart testbeds.

The OpenShift PaaS is deployed over five nodes, according to the roles identified in Table 4.

Table 4 – OpenShift nodes

Node name	Roles of Nodes	Short Name
opsv3.cloudcomplab.ch	OpenShift Controller	O0
opsv3-compute0.cloudcomplab.ch	OpenShift compute	O1
opsv3-compute1.cloudcomplab.ch	OpenShift compute	O2
opsv3-compute2.cloudcomplab.ch	OpenShift compute	O3

3 PoC Evaluation Scope and Methodology

The main objective of this section is to describe the scope and methodology for the evaluation of end-to-end composed services in an integrated environment according to the evaluation methodology defined in Section 2 of Deliverable D3.5. In particular, here we detail the mechanisms, tools, testbeds, platforms, and infrastructures that were employed to evaluate the DSS and the IMS PoC scenarios.

3.1 Scope and methodology

For the motivations extensively described in previous MCN work and because of their challenging exemplary aspects, we have decided to extensively evaluate two end-to-end composed service scenarios. On the one hand, the IMS PoC can be seen as an articulated integrated scenario, including several individual services and in particular RANaaS. On the other hand, the DSS PoC scenario is able to show the results of the integration with the Resource Aware VNF Agnostic (RAVA) NFV management and orchestration, thus permitting a thorough evaluation of the associated performance results when integrated in a composed end-to-end service. Let us rapidly note that some of the Key Performance Indicators defined in our evaluation methodology are not applicable for the DSS PoC but can be usefully collected for the other IMS scenario.

It should be clearly stated that the evaluation performed in the extension period has a deep focus on the dimensions of the software network performance including: system KPIs, resources KPIs, lifecycle KPIs and cloud-native KPIs. This greatly enhances the evaluation reported in D6.4 for MCN M36 period, which included mainly functional and non-functional evaluation aspects, without considering objective patterns of workload (i.e., spike tests, stable load, constant load, among others) to evaluate and experiment the end-to-end composed service scenarios. The enrolment of UNIBO, in the extension period, was crucial to the definition of the evaluation methodology, extensively reported in Deliverable D3.5 and its application in the experimentation and evaluation activities of WP6.

To collect the different KPI values, the following approaches have been followed:

- **Service/Component Monitoring** - corresponding data are collected by the MCN monitoring system (MaaS) from the individual services;
- **Infrastructure/Platform Monitoring –** data are collected in the MCN monitoring system (MaaS) from both the platform (OpenShift) nodes and the infrastructure (OpenStack) nodes;
- **SM/SO Monitoring** - in this case performance-related data are collected by the MCN monitoring system (MaaS) from SM and SO, in particular with regards to timing of deployment, provisioning, scaling, and disposal of either a service or an end-to-end scenario (built-in feature);
- **External Tools** - external tools are employed to generate emulated traffic and to collect evaluation metrics that are only measurable from the outside, such as delay and throughput, among others.

Evaluation is performed over the different dimensions we have originally specified in the D3.5 evaluation methodology, as summarized in Table 5, for the services composing the end-to-end IMS and DSS (with RAVA support) PoCs.

Table 5 – KPIs in the PoC Evaluation

Dimension	KPI	KPI Description	Services
System KPIs	KPI 1	Attachment/registration success rate	RANaaS, EPCaaS, IMSaaS
	KPI 2	Session establishment rate	RANaaS, EPCaaS, IMSaaS
	KPI 3	Session drop rate	RANaaS, EPCaaS, IMSaaS
	KPI 4	Attachment delay	RANaaS, EPCaaS, IMSaaS
	KPI 5	Session establishment delay	RANaaS, EPCaaS, IMSaaS
	KPI 6	Data Plane QoS	RANaaS, EPCaaS, IMSaaS
	KPI 7	Data Plane delay	RANaaS, EPCaaS, IMSaaS
Lifecycle KPIs	KPI 8	Installation duration	All
	KPI 9	Deployment and configuration duration	All
	KPI 10	Disposal duration	All
	KPI 11	Service Upgrade duration	All
Cloud-Native KPIs	KPI 25	Overall reconfiguration latency. This KPI includes all the KPIs from KPI12 to KPI24	DSS
Resources KPIs	KPI 26	CPU Usage	All
	KPI 27	Memory Usage	All
	KPI 28	Network Usage	All
	KPI 29	External network usage	All
	KPI 30	Storage Usage	All
	KPI 31	Number of VMs used	All
	KPI 32	Compute/network/storage resources consumed for orchestration	All

3.1.1 System KPIs

The System KPIs have been measured to assess the performance of services in a End User perspective. The measurements have been performed using iperf[2] and D-ITG[3], and the PJSIP tool[4] , according to the associations reported in Table 6.

Table 6 – System KPIs and measurement characteristics

KPI	KPI Description	Measurement characteristics
KPI 4	Attachment/Registration delay	• Measured for the RAN+EPC attachment and for the IMS Registration
KPI 5	Session Establishment delay	• Between IMSaaS and User Equipment attached to RAN OAI at Fokus
KPI 6	Data Plane QoS	• Packet sizes (bytes): 64, 768 • Send Rate with Inter Departure Time (IDT) in seconds: 0.2, 0.4, 0.8, 1.0 • Measured for simple internet sessions and for RTP data traffic
KPI 7	Data Plane delay	

During the execution of the measurements, it was observed that one of the most valuable results for the KPIs related to delay a very important result was to be able to determine which of the components contributes to the increase of the specific KPI the most. Therefore, the system KPIs were split into different contributions of the different components as they add to the final end-to-end delay. Such a solution is also recommended for the final end-to-end product benchmarking as to be able to distinguish between the effects introduced by the products coming from different vendors as well as from the effect introduced by the networking and virtualisation plane.

3.1.2 Resource KPIs

The resource dimension includes metrics for CPU, memory, and network usage information of compute/network nodes of both OpenStack and OpenShift. Metrics are also mapped to the monitoring items of the monitoring system (MaaS) integrated in our PoCs, relying on Zabbix in the targeted testbeds, as outlined in Table 7.

Table 7 – Resource KPIs and monitored metrics mapping

KPI	Description	Monitored metrics in MaaS
KPI 26	CPU	User CPU utilization (system.cpu.util[,user]) System CPU utilization (system.cpu.util[,system]) Idle CPU (system.cpu.util[,idle]) CPU switches (system.cpu.switches) CPU load (system.cpu.load[percpu,avg1])
KPI 27	Memory	Memory available (vm.memory.size[available]) Used Memory (vm.memory.size[used])

[2] http://iperf.fr

[3] http://traffic.comics.unina.it/software/ITG/

[4] http://www.pjsip.org/

		Number of Processes (proc.num[])
KPI 28	Network usage	Network interface input packets (net.if.in[]) Network interface output packets (net.if.out[])
KPI 30	Storage Used	Total storage used by all the Service Instance Components
KPI 31	Number of VMs	Total of Service Instance Components

3.1.3 Lifecycle KPIs

The lifecycle KPIs can be associated with the metrics of the service lifecycle of MCN, as reported in Deliverable D6.4. Such metrics apply to both considered PoCs. Table 8 provides a mapping between the lifecycle KPIs and the MCN lifecycle metrics.

Table 8 – Lifecycle KPIs and mapping of metrics

KPI	KPI Description	MCN Lifecycle metric
KPI 8	Installation duration	Design phase duration[5]
KPI 9	Deployment and configuration duration	Deployment time Provisioning time
KPI 10	Disposal duration	Disposal time
KPI 11	Service upgrade duration	Operation & Service management time, which includes retrieve time, and updates times during scaling operations.

In particular, the lifecycle KPIs are measured using the logging facilities in the Service Manager (SM) library. The Graylog[6] was kept from previous evaluations (performed up to M36) since it has demonstrated to support scalability well and allows retrieving the performance information easily for data analysis, as reported in D6.4 (Section 3.5).

3.1.4 Cloud-Native KPIs

The cloud-native KPIs, specified in Section 3.3.3 of Deliverable D3.5, are put under observation and evaluation for all the individual services involved in the end-to-end services. For the DSS service, these KPIs are documented in Section 4.3.1.5 and in Section 4.3.1.6, which includes the extended KPIs to evaluate the RAVA management/orchestration migration operations and fault management support in DSSaaS. The IMSaaS, instead, is used to highlight the evaluation of cloud-native KPIs as presented in Section 4.3.2.2.2 for a stairs workload test case.

It should be noticed that these KPIs are not reported in the DSS or IMS PoC evaluations, since there the aim is to assess the performance of individual services (i.e. IMS and DSS) when facing different workload models. Thus, it is not to be evaluated there a coordinated scaling of services, as previously motivated and reported in Deliverable D6.4.

[5] This metric has not been evaluated in the End-to-End evaluation context.

[6] https://www.graylog.org/

3.2 Digital Signage System (DSS) PoC

This section aims at describing the DSS PoC scenario, which includes the following services: DSSaaS, DNSaaS, MaaS, and RCBaaS, as depicted in Figure 2. This PoC does not include all the services as in M36 (e.g. AAAaaS, SLAaaS, ICNaaS) since these services were not included in the extension period. DNSaaS, RCBaaS and MaaS have been kept as they operate as support services.

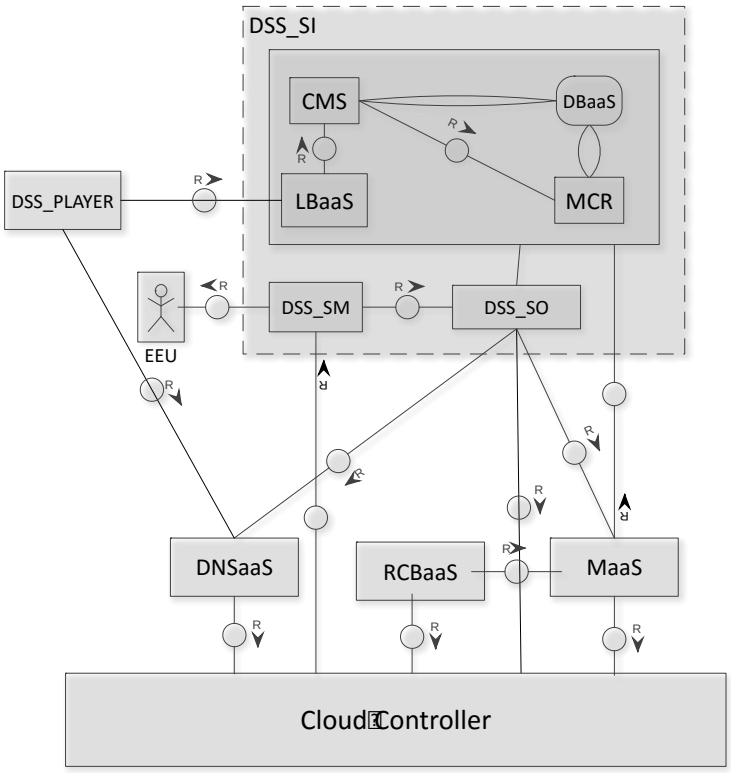


Figure 2 – DSS PoC services

This PoC deploys all the composing services in the Bart testbed, according to the specific roles of each service reported in D6.4. The evaluation dimensions in this scenario include mainly the lifecycle KPIs and the resource KPIs in terms of CPU and memory impact in both the (OpenStack) infrastructure and (OpenShift) platform nodes.

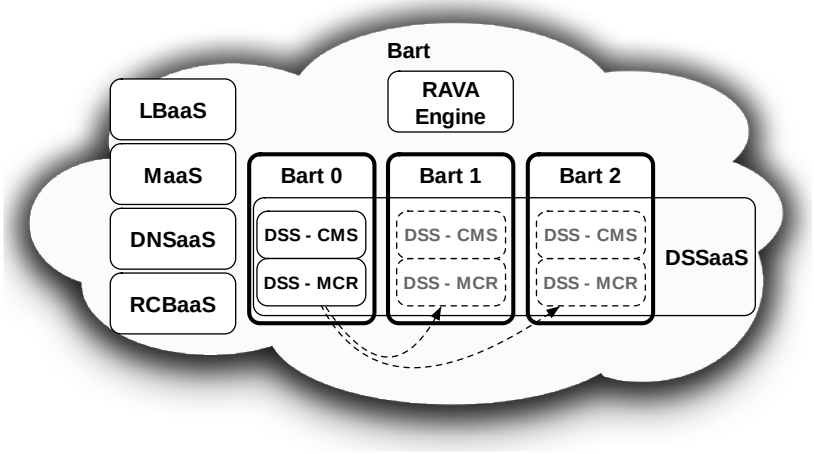


Figure 3 – DSS PoC services and testbed

Figure 3 depicts the DSS end-to-end scenario with the components that can be migrated between the distinct compute nodes in the Bart testbed, more specifically, the Content Management Server (CMS) and the Main Content Repository (MCR) components of DSSaaS. The LBaaS, MaaS, DNSaaS and RCBaaS are employed as support services. All the logic and setup for experimentation are detailed in Section 3.2.4 of Deliverable D5.5.

3.3 IP Multimedia System PoC

This section has the primary goal of presenting the evaluation of the IMS PoC scenario, which includes: IMSaaS, RANaaS at Fokus, RANaaS at Eurecom, EPCaaS, DNSaaS, MaaS, and RCBaaS (cf Figure 4).

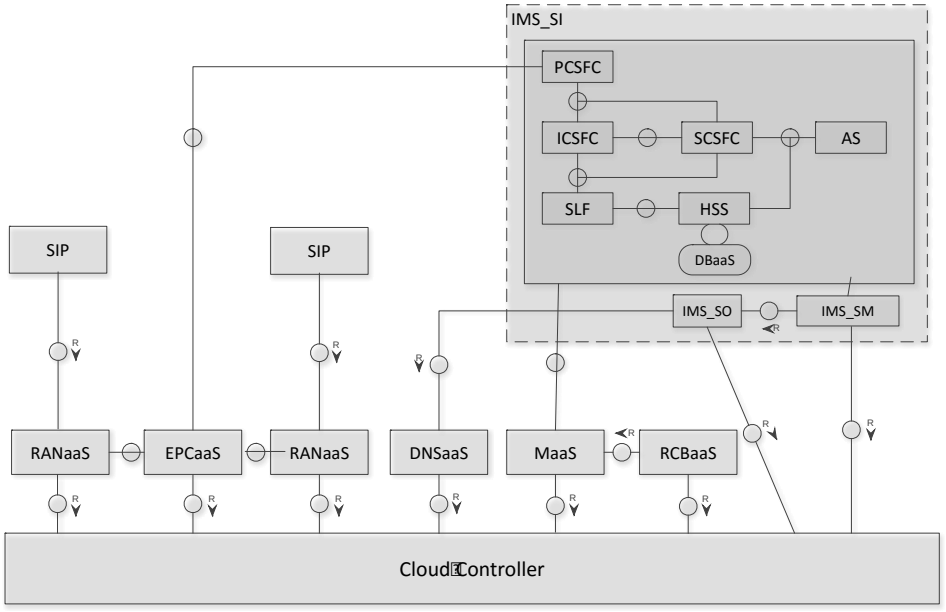


Figure 4 – IMS PoC services

The IMS PoC in the extension period includes two RANaaS services, operating in different testbeds, as depicted in Table 9, and does not include services like the ANDSFaaS (as reported in Deliverable D6.4 for up to M36).

Table 9 – IMS PoC services and testbeds

Service	Testbed
DNSaaS	Bart
EPCaaS	Bart
IMSaaS	Bart
MaaS	Bart
RANaaS (OAI with BBU+RRH)	Eurecom
RANaaS (OAI)	Fokus
RCBaaS	Bart

As already stated, the main novelty in the IMS PoC realized during the project extension is the support of two distinct geographic RANaaS, one located in the Fokus region (Berlin, Germany) and another one in the Eure region (Sophia Antipolis, France). Measurements between the Eurecom and Bart testbeds report on an observed round trip delay in the order of 20-30ms, while between Fokus and Bart such measurements rely in the order of 25-35ms. In addition, in the "extension" IMS PoC the eNodeB in both regions is configured for a 5MHz channel bandwidth in band 7 (2.68GHz) operating with signal antenna.

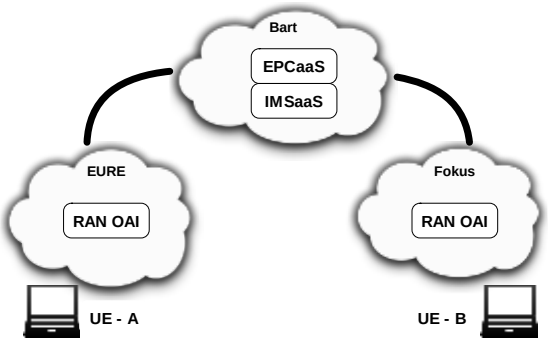


Figure 5 – IMS End-to-End evaluation scenario with User Equipment

Figure 5 depicts the IMS end-to-end scenario with User Equipment (UE) connected at Eure and Fokus regions. The UE–B is based on a laptop running Linux and connects to RAN OAI through a LTE dongle, while the UE-A has Windows 7 installed (also for practically showing portability and interoperability of the solution over different operating systems for UEs) and also connects to RAN OAI through a LTE dongle. The delay verified between the two clients is in the order of 70-85ms.

The difference between the two RANaaS is that they are built upon different RAN architectures. More in detail, the RANaaS maintained at Fokus is built upon the classic RAN architecture where a base station includes both the radio front-end equipment and the baseband processing unit (enabling the usage of real phones for the testing), whereas the RANaaS maintained at Eure is built upon the C-RAN architecture where the radio front-end is decoupled from the baseband unit and connected to it via a fronthaul interface. It is important to note that in our approach the fronthaul interface is Ethernet-based. More details on the differences of the two RANaaS can be found in Section 4 of Deliverable D3.5.

The system KPIs targeting data plane QoS measurements between the UE-A and UE-B were collected by using the widespread and well-known iperf tool, as well as via the PJSIP tool. In addition, this PoC evaluation includes the consideration of lifecycle and resource KPI dimensions; the remaining dimensions of the evaluation methodology are taken into account when reporting the evaluation of the individual composing services.

4 Evaluation Results

The extensive evaluation performed for the previously described PoCs includes experimental results from five runs to increase the accuracy levels; all the reported results in the following parts of this deliverable are average values over these five runs.

The results for the resource KPIs are presented for all the **B**-Bart OpenStack nodes and for all the **O**-OpenShift nodes with **SM** running at the Lisa testbed. This testbed was made available by ZHAW, and relies on OpenStack without the MCN extensions, since the CloudSigma testbed was not available in the extension period. There are four Bart nodes enabling the Infrastructure as a Service functionality (B0, B1, B2 and B3), with different rules (see the details in Section 2.2). The Platform as a Service functionality is enabled by OpenShift v3 nodes O0, O1, O2, and O3, with specific rules as well (cf. Section 2.3).

4.1 DSS PoC

This subsection reports and discusses the DSS PoC evaluation results for both resource and lifecycle KPIs. The aim is mainly to determine the impact in terms of required resources of having the DSS PoC end-to-end scenario deployed. As stated in Section 3.2, this PoC only includes the DSSaaS, DNSaaS, RCBaaS and MaaS. The DSSaaS has been enhanced to support RAVA features, while DNSaaS, RCBaaS and MaaS have been employed as support services.

4.1.1 Resource KPIs

Figure 6 reports the collected experimental results for metrics related to CPU usage, namely KPI 26, for the diverse lifecycle phases. The "normal" situation reported in the figure refers to the time periods when there is no deployment or disposal of resources, as well as when the involved services have already been deployed and are under the operation & service management phase.

The deployment corresponds to the time periods when there is a deployment/provisioning phase (KPI 9 and KPI 10), while the disposal corresponds to the time windows when a disposal request is formulated and ongoing. It should be noticed that the disposal, due to its short duration (see the comments about the lifecycle KPIs in Section 4.2.3), is not clearly visible in all the nodes, namely Bx-OpenStack nodes.

The difference between the normal and other lifecycle phases is more evident in the metrics "system.cpu.util[,user]" and "system.cpu.load[percpu,avg1]", where the CPU usage percentage is significantly higher. There is a high variation in the deployment/provision lifecycle phase (green) and disposal phase (blue). This trend occurs in all the infrastructure and platform nodes. The CPU usage in the machine running the SMs is almost neglected since the "system.cpu.util[,idle]" lies in values around 100%. The low number of services, when compared for instance to the IMS PoC, leads to such low overhead.

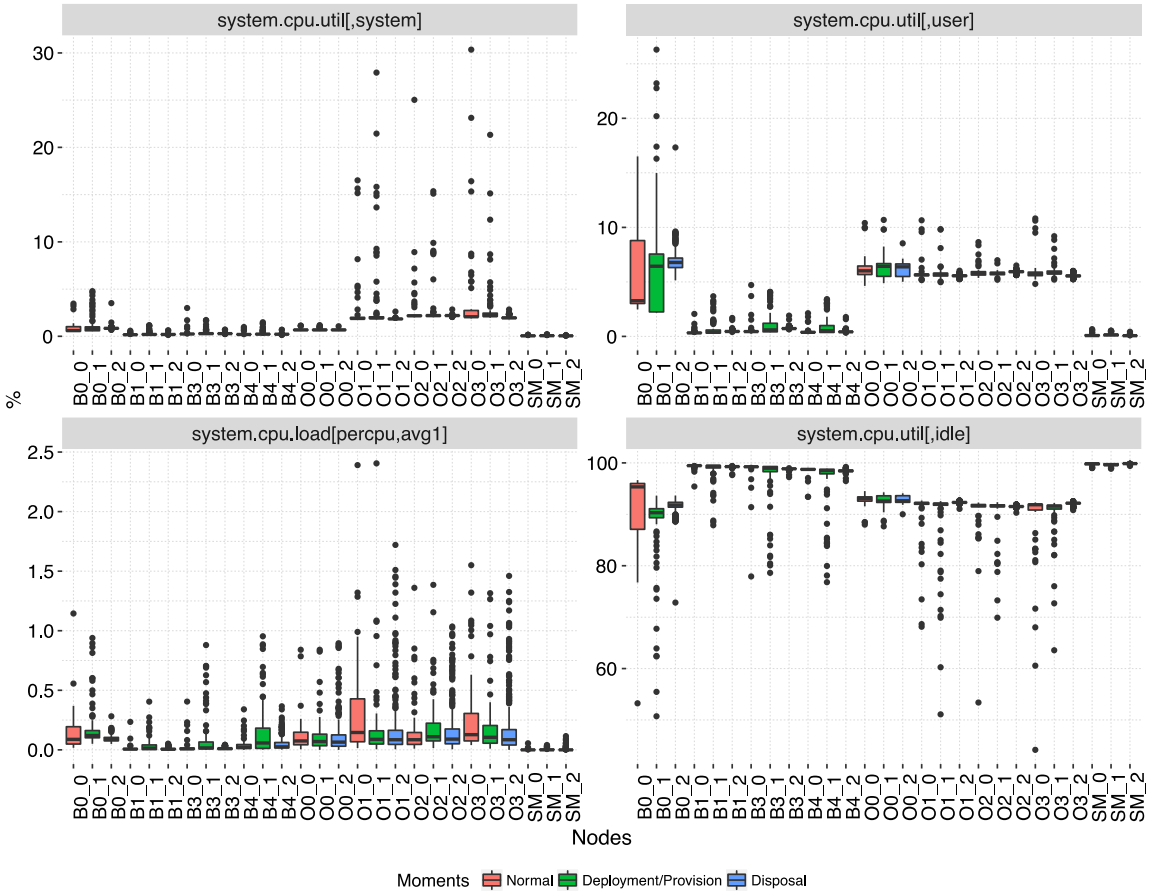


Figure 6 – Metrics for CPU in the DSS PoC (KPI 26)

Figure 7 depicts the collected results for metrics related with memory, i.e. KPI27, in particular with "proc.num[]" and "vm.memory.size[available]" in the DSS PoC. The former counts for the number of processes, while the latter measures the total memory in bytes that is inactive, cached, and free, thus with higher values representing better performance. In line with the CPU usage results, the memory and the number of processes do not differ in the machine running the SMs. In Bart, B0 exhibits a higher number of processes due to its role of acting as a controller node for OpenStack. Generally speaking, the deployment, provision and disposal phases have slightly more impact in the infrastructure nodes, since the available memory is lower in the Bart nodes running Openstack. Also it should be noticed that the DSS PoC has more impact in terms of CPU usage in the platform and infrastructure resources, while in the infrastructure (OpenStack) there are also some noticeable effects on the usage of memory.

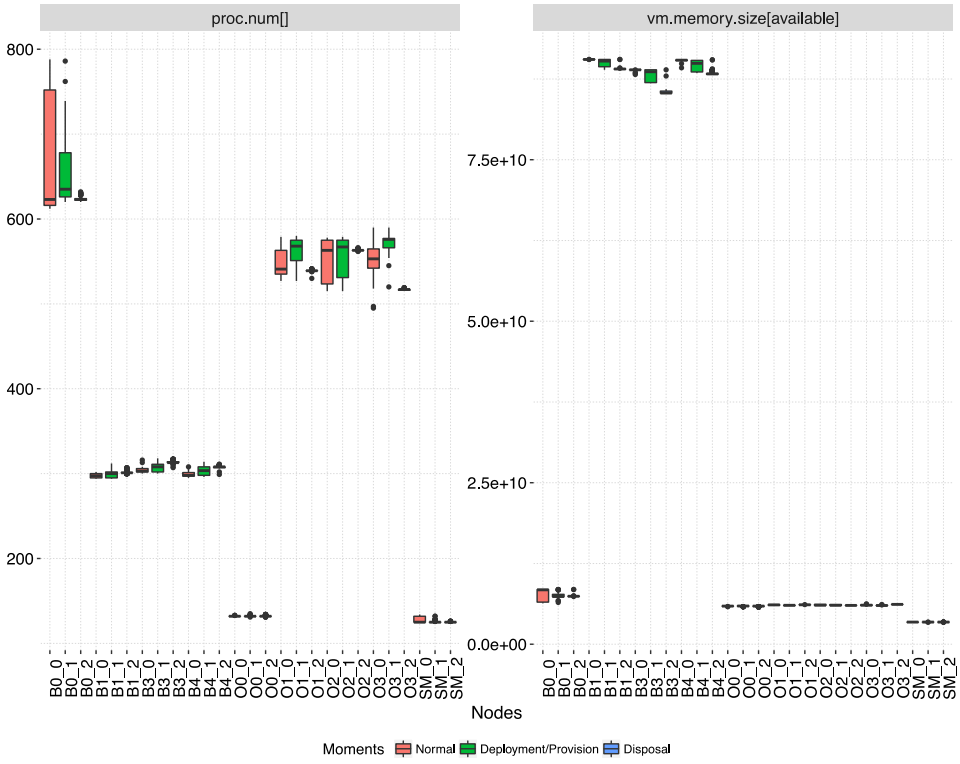


Figure 7 – Metrics for memory in the DSS PoC (KPI27)

The network resources considered in KPI 28 include the number of inbound and outbound packets in the respective main interfaces of OpenStack, OpenShift, and SM nodes, as depicted in Figure 8. We can observe that the OpenStack nodes send and receive more traffic, due to the higher volume of traffic. Such fact is intrinsically associated with the number of Service Instance Components (SICs). The Controller node (B0) and the Compute node (B4) are the ones that exchange more traffic, as B4 is the compute node chosen to manage all the SICs of the four service composing the DSS PoC.

The memory and storage resources used and the number of VMs employed for the DSS PoC are summarized in Table 10 for the different services.

Table 10 – Resource KPIs of DSS PoC

Service	Storage (KPI 29)	Number of VMs (KPI 31)
DSSaaS	Disk: 4 x 20GB + 5GB = 85GB RAM: 4 x 2GB + 1GB = 9GB	5
DNSaaS	Disk: 20GB + 40GB = 60GB RAM: 2GB + 4GB = 6GB	2
MaaS	Disk: 20GB RAM: 2GB	1
RCBaaS	Disk: 40GB RAM: 4GB	1

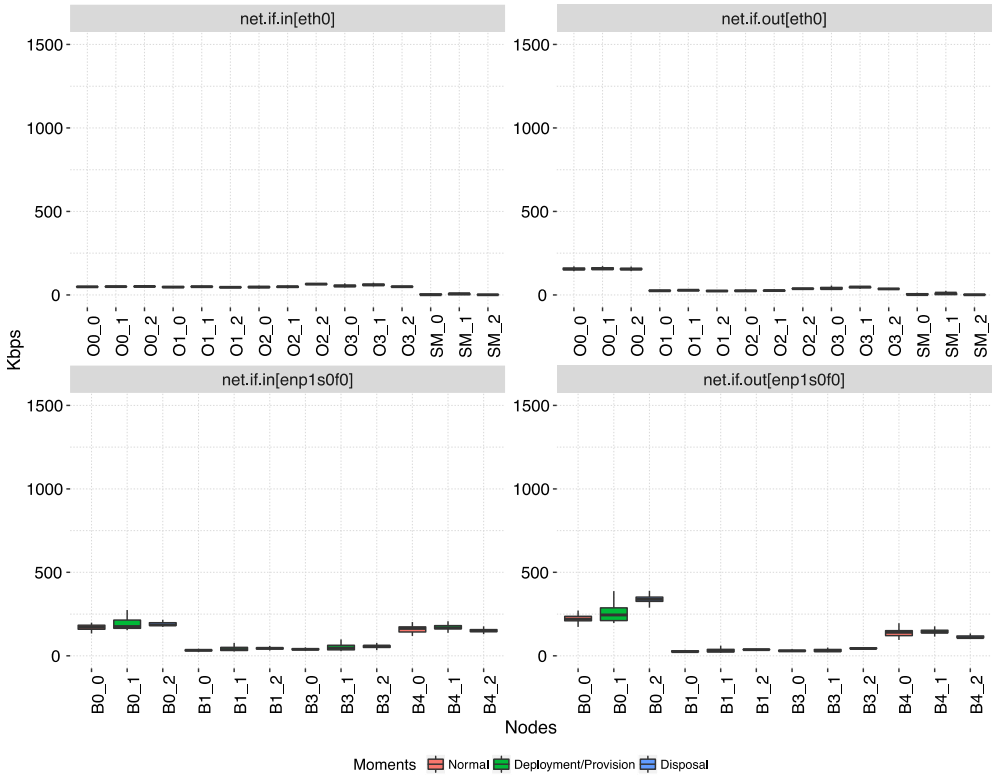
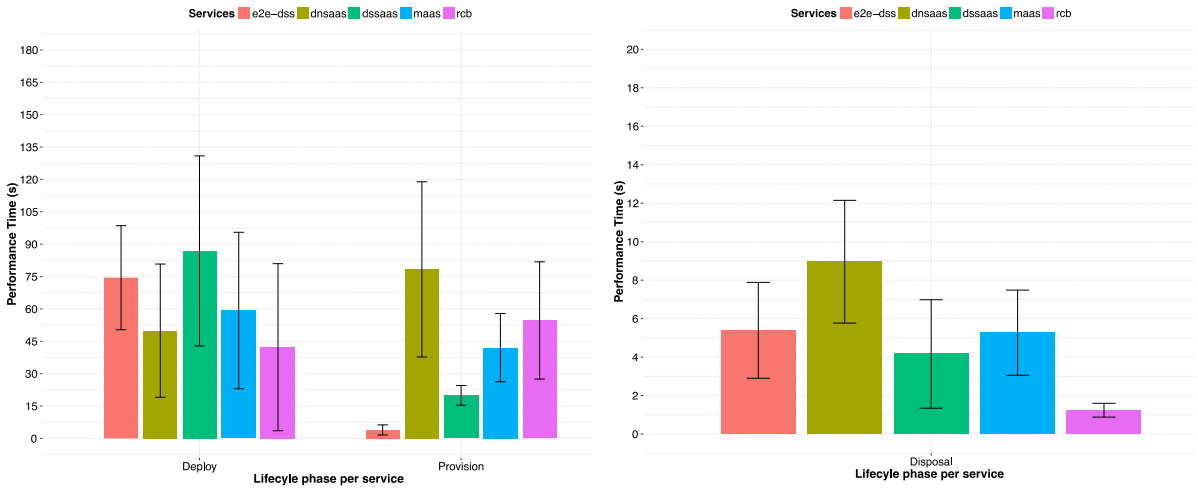


Figure 8 – Metrics for network in the DSS PoC (KPI 28)

4.1.2 Lifecycle KPIs

The results collected for the lifecycle KPIs are summarized in Table 11, which reports the measured indicators for the different runs.

Table 11 – Lifecycle Deployment/Provision (KPI 9) and Disposal in DSS PoC (KPI 10)



The full deployment and provisioning time is in the order of 200 seconds for 4 services. Let us notice that this result is lower than the one reported in D6.4 (Section 4.3.2), where the values for this metric

were in the order of 340 seconds, namely due to the high number of services (7) composing the DSS PoC realized before the project extension period. The overall disposal time is around 10 seconds.

4.2 IMS PoC

This subsection presents and discusses the collected experimental results for system, lifecycle, and resource KPIs related to the IMS PoC. As already stated and discussed, the cloud-native KPI evaluation is presented for the individual service evaluation and not considered. As stated in Section 3.3, this PoC includes the IMSaaS, EPCaaS, RANaaS with Ethernet fronthaul, RANaaS at Fokus, DNSaaS, RCBaaS and MaaS. The IMSaaS has been modified to support the Open Baton orchestration model, and RANaaS has been modified to split the BBU and RRH functionalities. DNSaaS, RCBaaS and MaaS have been employed as support services.

4.2.1 System/Service KPIs

The service KPIs were split into two types of KPIs: related to the control plane and related to the data plane.

4.2.1.1 System/Service KPIs for the control plane

As the system used for the testing included only two UEs and as the tests were executed in a sequential form, the KPIs related to the sanity functioning of the system at different load levels are not relevant. This includes the attachment and session establishment success rate and the session drop rate. These KPIs are highly dependent on the number of subscribers at a specific moment in time. For the testbed system, only one subscriber was considered. Thus, the only interesting KPI remained KPI4: attachment and registration delay, as introduced in Table 12.

Table 12 – IMS PoC System KPI (KPI 4)

KPI 4 metrics	Details	Value
Attachment delay	UE-eNB-CTRL-HSS-Switch	Median: 285ms Minimum: 192ms Maximum: 402ms Median network delay: 158ms
Registration delay	UE-DNS-CSCFs-HSS	Median: 200ms Median network delay: 83ms

However, as the system included multiple components coming from multiple project partners, a very interesting question arise on which of the components is contributing with how much to the delay of the overall procedure. A similar question would have to be considered also in the case of the other service KPIs especially because there is a stringent need in the end-to-end system to determine the malfunctioning part as fast as possible.

For the attachment delay, the procedure was considering the Fokus UE and eNB connected to the Bart EPC. The procedure is illustrated in Figure 9. For the specific procedure, Wireshark traces were used to compute the delay of the procedures in each of the components: eNB, CTRL, HSS and Switch.

To these procedures, the delay over the LTE radio was computed based on the measurements from the eNB. This solution was chosen, as there were no means available to determine the delay of the specific LTE messages without having internal access to the UE modem, which in this testbed setup was a common commercially available dongle. Thus, the delay does not include the first message (Attach request), as the eNB is not aware of the time this message was transmitted, only the remainder request-response pairs. However, the time considered includes also the processing within the UE of the messages. We assume that the measurements have an error of maximum 14ms for the full attachment procedure, including 10-12ms for the transmission over the LTE environment of the "Attach Request" message.

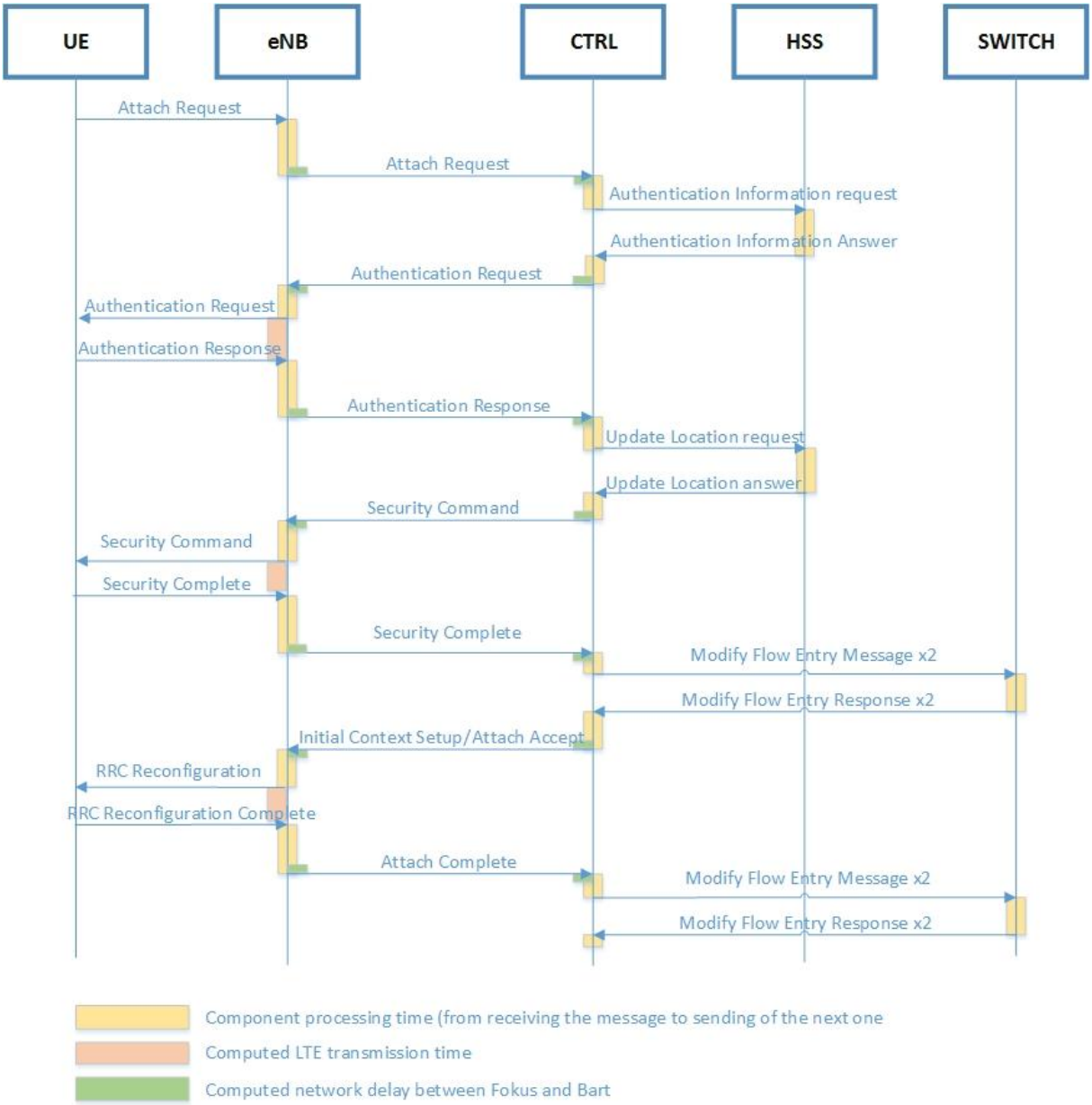


Figure 9 – End-to-end EPC attachment delay evaluation

Furthermore, a very large amount of the delay is due to the interconnection network between the two locations (Fokus in Germany and Bart in Switzerland). Although other parts of the network delay may

be considered (rounded up to 1ms between two virtual machines for a one-way connection), the rest of the delays are insignificant.

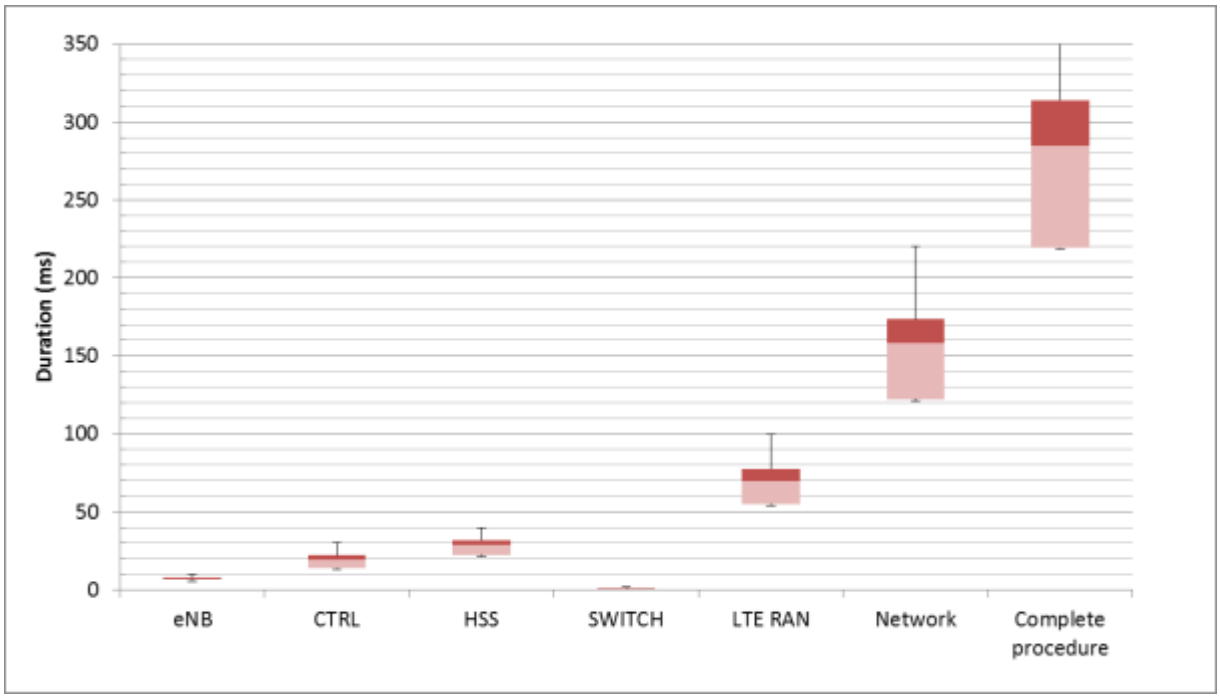


Figure 10 – System KPI: Attachment Delay – split per components in RAN + EPC

As expected, the delay within the eNB and the Switch components are minimal, while the delay in the LTE RAN and in the CTRL is proportionate with the number of messages which have to be computed. Also, considering the RTT time of more than 25ms between Fokus and Bart, the network delay, which contributes the most significantly to the end-to-end delay, was in the foreseen limits (14ms for a message between the data centers and 1ms for the messages within a data center).

An unexpected high delay was achieved in the HSS due mainly to the MySQL queries, which increased with 30% the procedure duration if not considering the network transmissions. Determining this factor was not possible when using the KPI only from the subscriber perspective as planned in D3.5.

Following the learned issues from the LTE attachment procedure, a same split of the delay was also considered for the IMS registration. For the IMS, the UE and the eNB considered were located at Fokus while the IMS was located together with the EPC at Bart. As illustrated in Figure 11, the measurements were performed at the UE, eNB, CSCFs, HSS and DNS level, while the delay of the switch was considered negligible in comparison with the network delay.

The delay of each component was computed based on the received and send messages as observed by using Wireshark. One registration was executed at a specific moment in time.

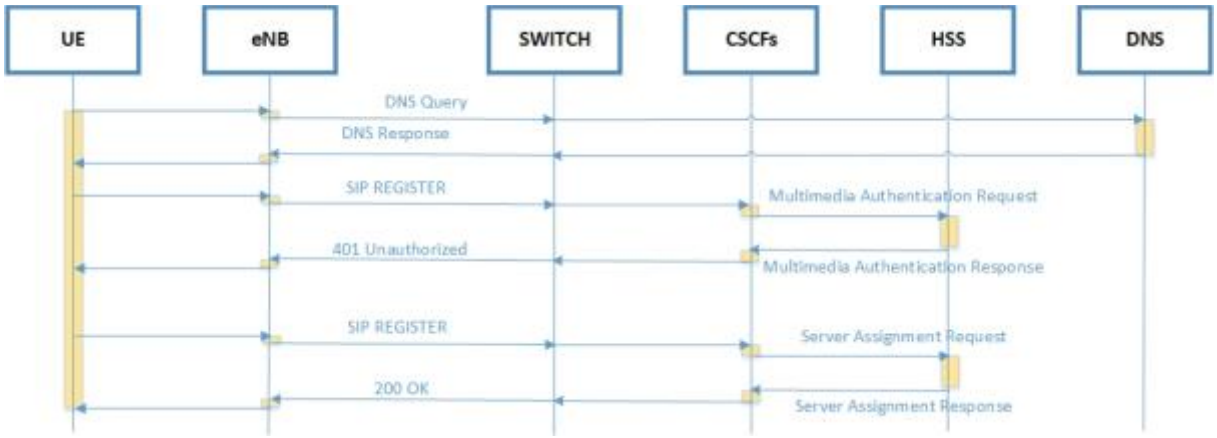


Figure 11 – End-to-end IMS Registration Delay Evaluation

The results obtained for the IMS registration are illustrated in Figure 12. As observed also in the LTE attachment case, a very large part of the delay is due to the HSS data base queries as well as to the DNS query. The network usage is proportionate with the number of messages and equivalent to a median of 14ms for a one-way connection between the data centers, a median of 1ms for a one-way connection within components of the same data center or virtual machine and 1ms for the switch processing.

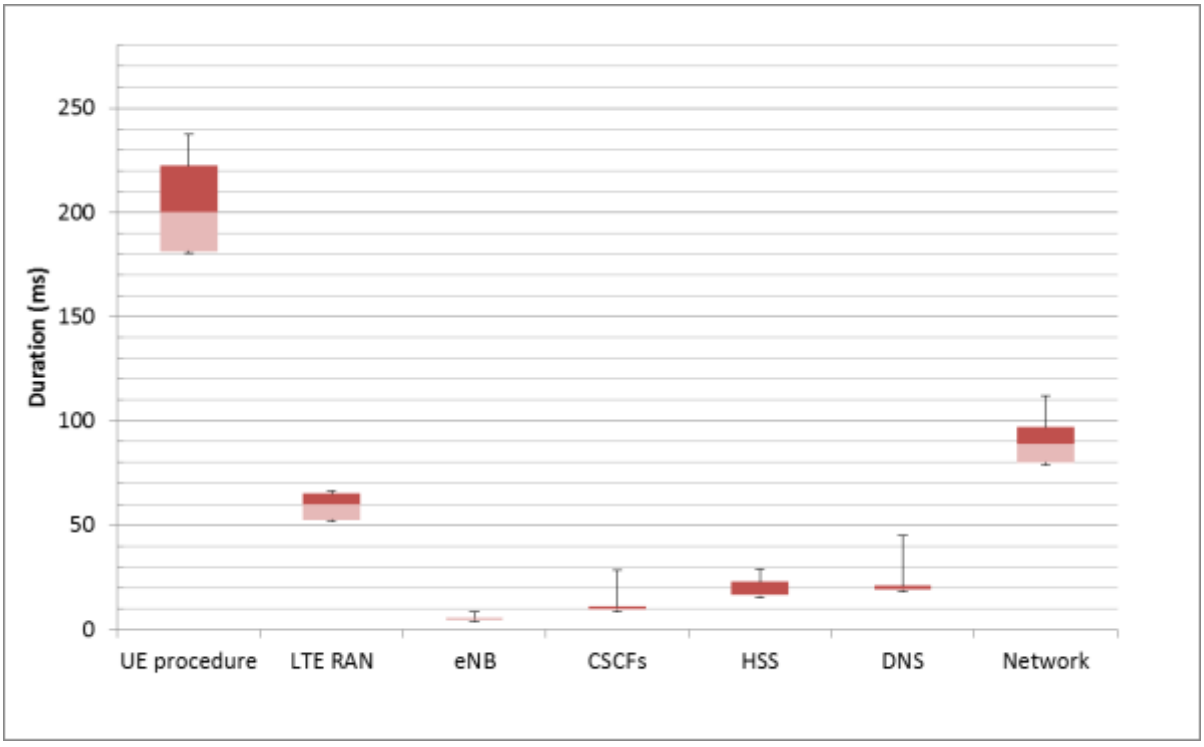


Figure 12 – System KPI: IMS Registration Delay – split per components in RAN + EPC + IMS

In conclusion, in order to be able to highly benefit of the system KPIs it is highly advisable to split them per system or even per component when possible, through this to be able to determine the source of a badly functioning system immediately when this happens. From another perspective, the measurements for the individual components are not needed unless the service KPIs are starting to have abnormal values.

4.2.1.2 System/Service KPIs for the data plane

The most relevant system/service KPIs in the perspective of the E2E provisioning relate to KPI 6 – Data Plane QoS and KPI 7 – Data Plane Delay, which are measured between UE-A connected to RAN OAI at EURE and UE-B connected to RAN OAI at Fokus. Table 13 summarizes the metrics collected for this KPI6, using iperf and D-ITG.

Table 13 – IMS PoC System KPIs (KPI 6) between UE-A and UE-B

Data Plane QoS (KPI 6)	Details	Value
Downlink bandwidth	From UE-B → UE-A (TCP traffic)	Avg: 2Mbit/s
Uplink bandwidth	From UE-A → UE-B (TCP traffic)	Avg: 1Mbit/s

The observed Round Trip Time (RTT) between the UE, as part of KPI 7, between the UE is depicted in Figure 13 for the different executed tests with different Inter Departure Times (IDT) and for packets with different size in bytes. As expected the packets with bigger size have higher values for RTT, due to the higher overhead in being processed at the eNodeB of EPCaaS (i.e. more resources are required to encapsulate and decapsulate packets).

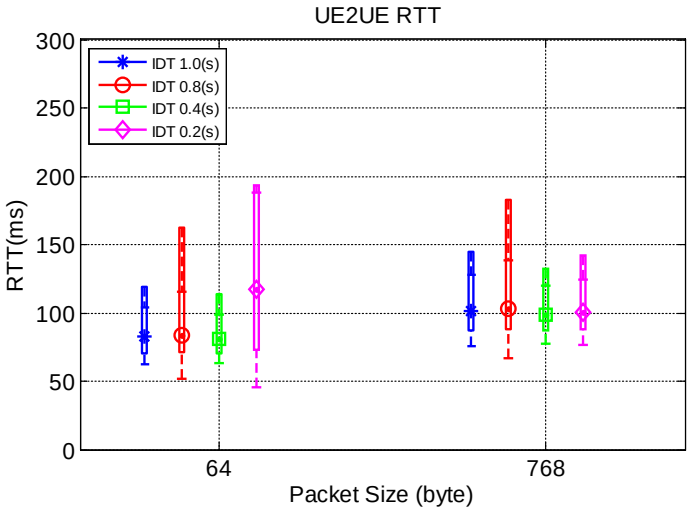


Figure 13 – IMS PoC UE-A to UE-B Round Trip Time

The session establishment delay (KPI5) and KPI6 have also been measured in the Fokus testbed, while using UE-B and a machine connected to the EPC network in the Bart testbed. Figure 14 depicts the values of a KPI 5-call setup call (from an invite to an established media connection in IMS service), the KPI 6-Audio RTT, and the respective jitter for both TX and RX directions of the audio channel. As already mentioned, the reported results are based on 5 runs; the different colours in Figure 14 relate to these different runs.

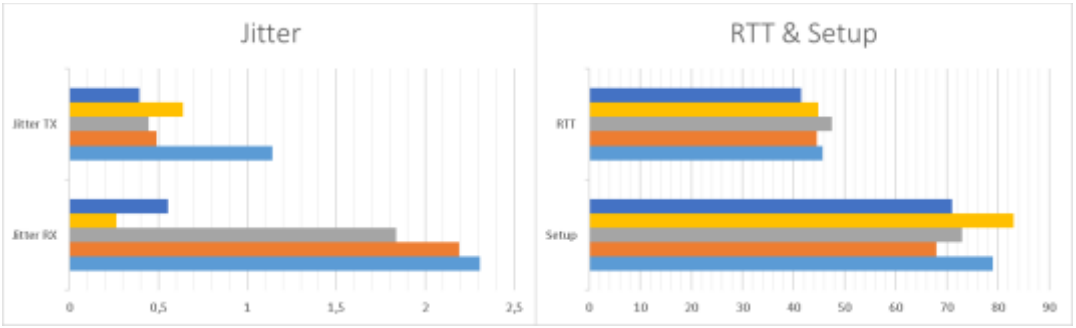


Figure 14 – IMS PoC KPI5 and KPI6 measurements

4.2.2 Resource KPIs

Figure 15 reports the collected experimental results for metrics related to CPU, namely KPI 26, for different lifecycle phases. The "normal" situation reported in the figure refers to the time periods when there is no deployment or disposal of resources as presented in Section 4.1.1 for the DSS PoC.

The difference between the normal and other lifecycle phases is more manifest in the metrics "system.cpu.util[,user]" and "system.cpu.load[percpu,avg1]", where the CPU usage percentage is significantly higher. There is a high variation in the deployment/provision lifecycle phase (green) and the outliers tend to have higher values. This trend occurs in all the infrastructure and platform nodes, as well in the machine running the SMs, even if in this node the difference is less noticeable.

During the disposal phase, as perceived in some of the nodes for OpenShift, namely O3, it can be noticed that there is a higher impact, if compared with the deployment phase, on the metric of "system.cpu.util[,user]". Such result is associated with the simultaneous removal of the containers with the SOs. Our deployment also supports parallelization of operations, but the service composition, managed by the E2E SO, leads to a low number of concurrent deployments.

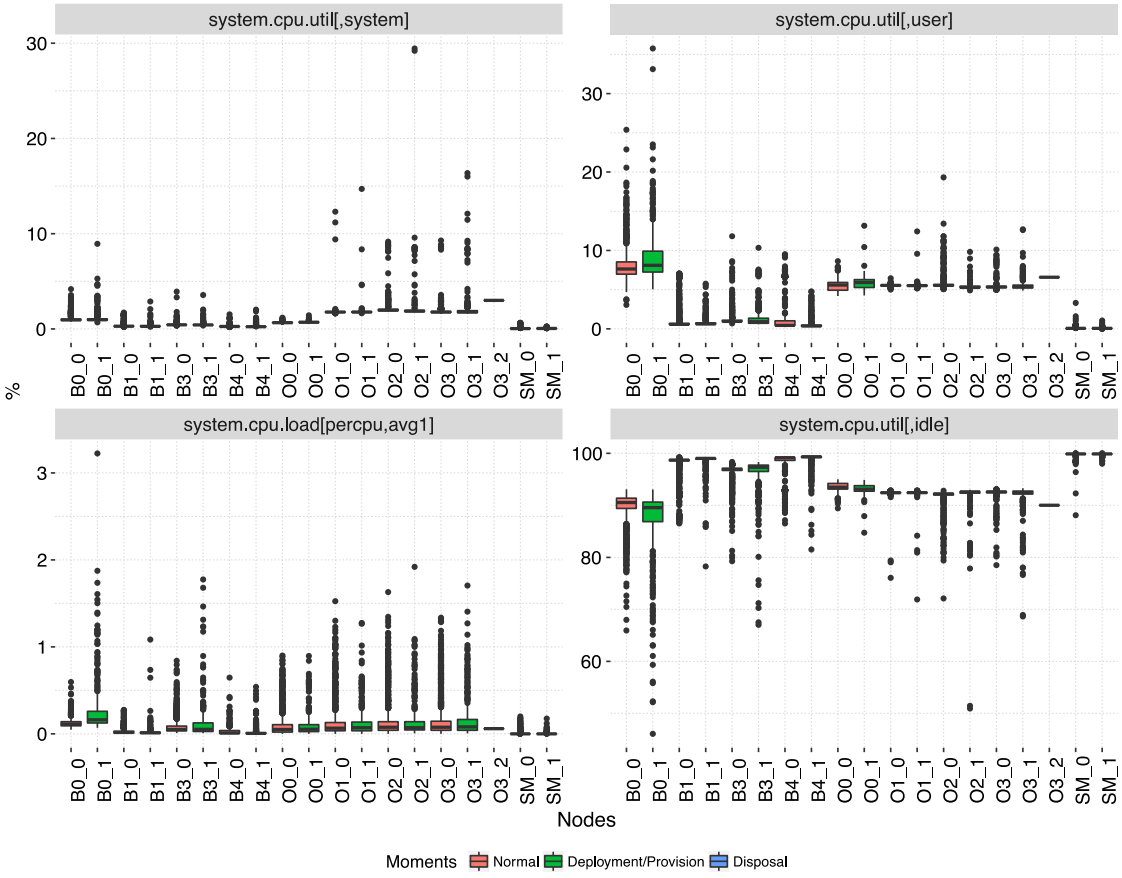


Figure 15 – Metrics for CPU in the IMS PoC (KPI 26)

Figure 16 reports the collected results for metrics related with memory, i.e., KPI 27, in particular with "proc.num[]" and "vm.memory.size[available]" in the IMS PoC. As in the DSS PoC, the former counts for the number of processes, while the latter measures the total memory in bytes that is inactive, cached, and free, thus with higher values representing better performance. The number of processes is not impacted by the different lifecycle phases. Indeed, its number is closely related with the role that a given node is performing. For instance, B0, acting as a controller node for OpenStack in Bart, exhibits a higher number of processes. The available memory is mainly impacted in the infrastructure nodes running OpenStack. Indeed, the platform nodes (Ox) have less memory usage variations between the diverse lifecycle phases. Considering such results, it is clear that the IMS PoC has more impact on the platform and infrastructure resources in terms of used CPU.

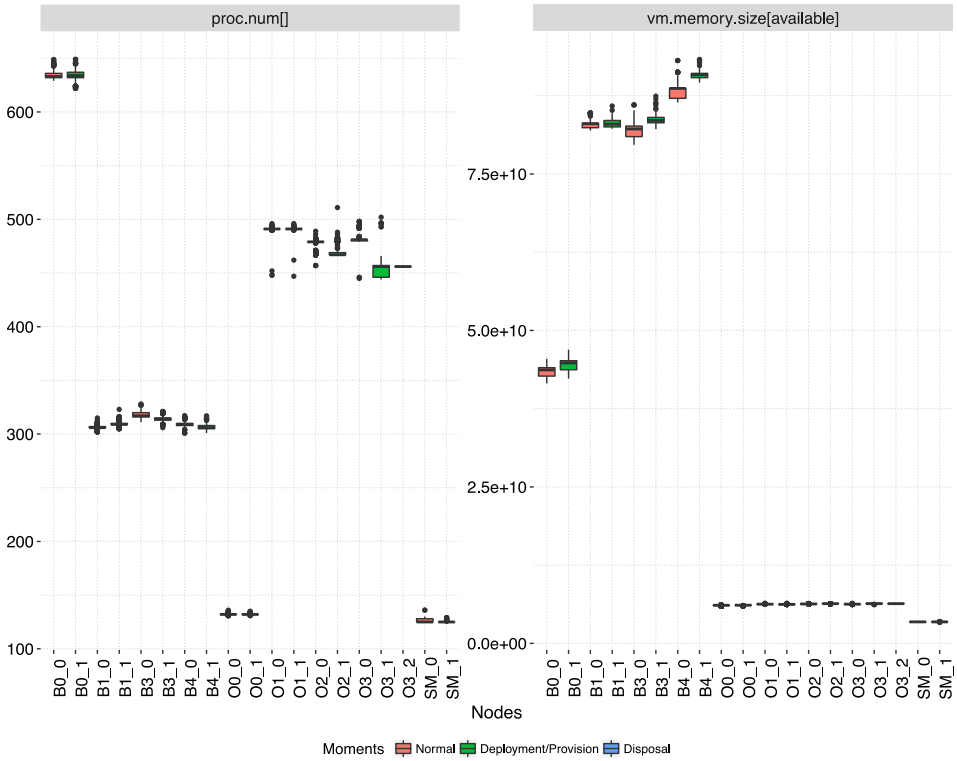


Figure 16 – Metrics for memory in the IMS PoC (KPI 27)

Considering networking evaluation, the network resources associated to KPI 28 include the number of inbound and outbound packets in the main interfaces of OpenStack, Openshift, and SM nodes.

As we can observe in Figure 17, the OpenStack nodes receive and send more traffic (a higher number of packets), as depicted in the lower part of the figure. In fact, while in the OpenShift and SM nodes there is a 1:1 mapping, in OpenStack such mapping is N:M, where one service has multiple Service Instance Components (SICs). For instance, DNSaaS employs 2 SICs, while MaaS employs one SIC. All these SICs are reachable from other services sending and receiving information, as requested. The OpenShift nodes are clearly impacted with the deployment/provisioning phase, as there is a constant communication between SM and SO of the different services to ensure the successful completion of the phase – a successful orchestration. In OpenStack there is a similar communication pattern; however, when services are deployed and ready, they can be reached and used, thus leading to a potential increase of exchanged IP packets.

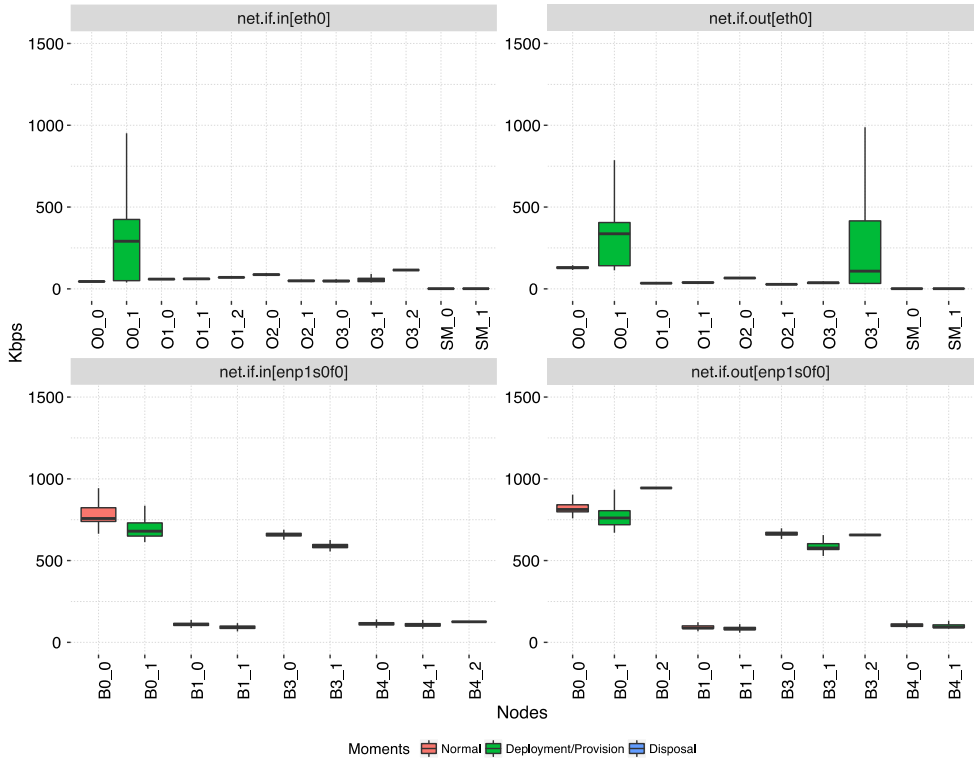


Figure 17 – Metrics for network in the IMS PoC (KPI 28)

The storage used and the number of VMs employed for the IMS PoC are summarized in Table 14 for the different services.

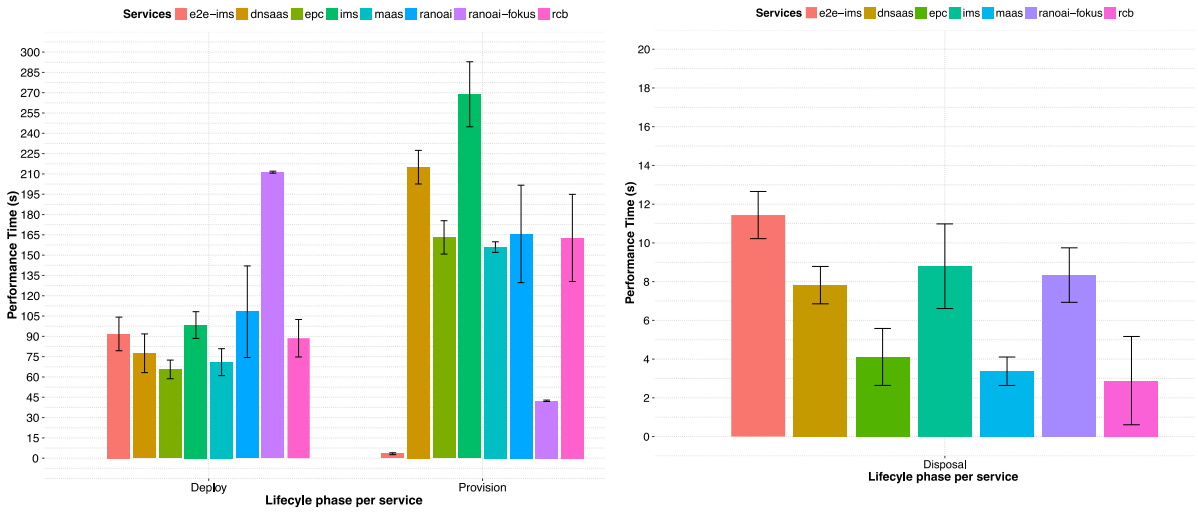
Table 14 – Resource KPIs of IMS PoC

Service	Storage (KPI 29)	Number of VMs (KPI 31)
DNSaaS	Disk: 20GB + 40GB = 60GB RAM: 2GB + 4GB = 6GB	2
EPCaaS	Disk: 20GB + 20GB + 20GB = 60GB RAM: 2GB + 2GB + 2GB = 6GB	3
MaaS	Disk: 20GB RAM: 2GB	1
IMSaaS	Disk: 20GB + 20GB + 40GB = 80GB RAM: 2GB + 2GB + 4GB = 8GB	4
RANaaS (EURE)	Disk: 80GB + 80GB = 160GB RAM: 8GB + 8GB = 16GB	2
RANaaS (Fokus)	Disk: 80GB RAM: 8GB	1
RCBaaS	Disk: 40GB RAM: 4GB	1

4.2.3 Lifecycle KPIs

The collected results for the lifecycle KPIs are summarized in Table 15 for all the services composing the IMS PoC.

Table 15 – Lifecycle Deployment/Provision (KPI 9) and Disposal (KPI 10)



The full deployment and provisioning time is in the order of 400 seconds for 7 services. Let us notice that this result has demonstrated to be highly dependent on the hardware where OpenStack operates. For this reason, in D6.4 (Section 4.4.2) the values reported for this metric were in the order of 255 seconds, namely due to the employment of the Bern region, which was using a cloud controller and a computing node running on top of a Dell PowerEdge R520 (see the detailed description in D6.4, Section 3.2.1). The measured overall disposal time was around 15 seconds.

4.3 Individual Services

This section reports the evaluation results of the individual services that are central for the realization of the end-to-end composed scenarios, i.e., DSS, IMS, and RAN OAI. Other M42 experimental evaluation results, namely those related to EPCaaS, are reported in Deliverable D4.6.

4.3.1 Digital Signage Service – DSSaaS

This section presents the individual service evaluation of DSSaaS, based on the scenarios and KPIs defined in D5.5 - DSSaaS Performance evaluation methodology. Detailed datasets for each scenario are explained in Section 4.3.1.1 and results for the execution of the scenarios are presented in Section 4.3.1.2.

4.3.1.1 Scenarios and related datasets

As described in D5.5, the evaluation methodology for DSSaaS performance has required the generation of different datasets and scenarios to cover performance profiles (stable, spike, stress) defined in our application of the general-purpose performance evaluation methodology described in Deliverable D3.5. In order to measure the targeted KPIs, specific datasets for each scenario have been elaborated. Datasets show the main configuration parameters for the scenario execution. For each scenario several calibration executions have been performed in order to tune the overall individual service behaviour. When required, the selected configuration values deriving from this calibration work are explained. For some

scenarios, runs with different configuration parameters have been executed in order to cover measurements for specific KPIs (e.g. fault management, scenario 2).

4.3.1.1.1 Scenario 1: Deployment and provisioning

Scenario 1 considers the complete deployment and provisioning of a DSSaaS service instance. In order to maximize measurements accuracy for the KPIs, 5 different runs of the scenario have been performed.

Table 16 – Scenario 1 (Deployment and Provisioning)

Scenario #1	Value	Unit
Initial number of CMS SICs	2	# number
Initial number of MCR SICs	2	# number

4.3.1.1.2 Scenario 2: Stable load

Scenario 2 considers a stable load, with minor or no usage peaks for all service components, emulating a stable production usage for an existing (previously provisioned) service instance. Scenario 2 considers two independent runs with slightly different datasets in order to collect fault management KPIs in an isolated test, which uses the Zipf as the retrieval distribution of contents (cf. Table 17 and

Table 18).

Table 17 – Scenario 2 (Stable Load)

Scenario #2 - constant load	Value	Unit
Number of emulated players (max):	6000	player
Player update ratio	1%	%
Avg. num. files per player update	5	files
Number of media files stored	50	files
Average size file	20	MB
Cached (yes/no)	yes	yes/no
Media file distribution used (size)	Gamma	distribution
LB distribution policy	Round Robin	balance
Fault forced in CMS component	0	# faults
Fault forced in MCR component	0	# faults
Scenario execution time	15	minutes

Table 18 – Scenario 2 (Fault Management)

Scenario #2 - fault management	Value	Unit
Number of emulated players (max):	6000	player
Player update ratio	1%	%
Avg. num. of files per player update	5	files
Number of media files stored	50	files
Average size file	20	MB
Cached (yes/no)	yes	yes/no
Media file distribution used (size)	Gamma	distribution
LB distribution policy	Round Robin	balance
Fault forced in CMS component	1	# faults
Fault forced in MCR component	1	# faults
Scenario execution time	15	minutes

4.3.1.1.3 Scenario 3: Spike test

Scenario 3 considers a high load for a short amount of time in order to measure service performance limits for an existing (previously provisioned) service instance. The dataset is depicted in Table 19.

Table 19 – Scenario 3 (Spike Test)

Scenario #3 - spike load	Value	Unit
Number of emulated players (max):	60000	player
Player update ratio	9%	%
Avg. num. of files per player update	10	files
Number of media files stored	3500	files
Average size file	60	KB
Cached (yes/no)	yes	yes/no
Media file distribution used (size)	Gamma	distribution
LB distribution policy	Least connections	balance
Scenario execution time	5	minutes
Retrieval distribution	Zipf	distribution

Let us note that file size has been drastically reduced in the case of our spike tests due to the testbed network limitations. In fact, in the case of using larger files, MCR is limited by the external outbound network bandwidth, which is fully dependent on the testbed, and thus less significant in terms of the reported performance indicators. Therefore, the dataset has been modified to measure maximum number of requests that can be handled by the MCR component.

Compared to scenario 2, the load balancing distribution policy has been changed from "round robin" to "least connections". This is intrinsically related to the warm up required by Java applications as the one processing the requests for CMS component. When a new SIC is deployed and provisioned due to a scale-out operation, it is not able to process the same number of requests of a previously existing instance, thus taking more time for each connection. Setting LB policy to round robin limits more the global system capacity in case of spike than setting it to least connections. Performance loss during a spike for a round robin configuration can reach 60%, so the experimental evaluation work has shown that a "least connections" policy selection is recommendable in this case.

4.3.1.1.4 Scenario 4: Stress test

Scenario 4 considers a constantly increasing load for a long time in order to measure service reaction to load, for an existing (previously provisioned) service instance. Service will scale in order to absorb the load generated.

Table 20 – Scenario 4 (Stress Test)

Scenario #4 - Stress Test	Values	Unit
Number of emulated players (max):	from 20,000 to 45,000	player
Player update ratio	7%	%
Avg. num. of files per player update	10	files
Number of media files stored	3500	files
Average size file	60	KB
Cached (yes/no)	yes	yes/no
Media file distribution used (size)	Gamma	distribution
LB distribution policy	Least Connections	balance
Retrieval distribution	Zipf	distribution
Scenario execution time	40	minutes

Initial load is set to 20,000 emulated players with a 6% player update ratio that is increased up to 45,000 players. Differently from scenario 3, the performance tool waits for 12 minutes after each load increase to give the service some time to stabilize and perform scale-out operations when required.

4.3.1.1.5 Scenario 5: RAVA-enhanced

The configuration of this scenario is exactly the same as for scenario 2, but with the additional consideration of the state of physical infrastructure nodes. This scenario aims to show how a live migration decision is taken in a situation of consistent stable load. As discussed in Deliverable D5.5, RAVA analysis, to reduce the scenario complexity and make the evaluation practically feasible with limited costs, the experimental evaluation activities focus on CPU and Inbound network resources for the DSSaaS CMS component. The corresponding dataset is presented in Table 21.

Table 21 – Scenario 5 (RAVA)

Scenario #5 – RAVA	Values	Unit
Number of emulated players (max):	6000	player
Player update ratio	1%	%
Avg. num. of files per player update	5	files
Number of media files stored	50	files
Average size file	20	MB
Cached (yes/no)	Yes	yes/no
Media file distribution used (size)	Gamma	distribution
LB distribution policy	Least Connections	balance
Fault forced in CMS component	0	# faults
Fault forced in MCR component	0	# faults
Scenario execution time	15	minutes
Retrieval distribution	Zipf	distribution
Scenario execution time	30	minutes
bart node cpu increase rate	60% + 1% / min	%/min
bart-1 node cpu increase rate	60% + 1% / min	%/min
bart-3 node cpu increase rate	90% + 0% / min	%/min
bart-4 node cpu increase rate	60% + 1% / min	%/min
bart net inbound increase rate	200Kbps + 25Kbps/min	KB/min
bart-1 node net inbound increase rate	200Kbps + 25Kbps/min	KB/min
bart-3 node net inbound increase rate	200Kbps + 25Kbps/min	KB/min
bart-4 node net inbound increase rate	200Kbps + 25Kbps/min	KB/min

4.3.1.2 Scenarios Execution

Evaluation scenarios and their corresponding datasets have been already presented in the previous Section 4.3.1.1. The results obtained for each scenario are now reported and discussed.

4.3.1.2.1 Scenario 1 - Deployment and configuration

Execution results show reasonable deployment times while provisioning (configuration) times are faster than those previously presented in D6.4 (M36). This is mainly due to the additional parallelization obtained from the usage of an MQ broker to enable efficient communications between SO and SICs. The measured KPIs obtained are shown in Section 4.3.1.4.

4.3.1.2.2 Scenario 2 - Stable load

Execution results show that all SICs components (two CMS and two MCR) exhibit a stable behaviour during the execution. All resources (CPU, Mem, and Network I/O) remain constant and the components' response times also remain stable. Load is properly distributed between both components of each kind. Obtained KPIs are described in Section 4.3.1.3.

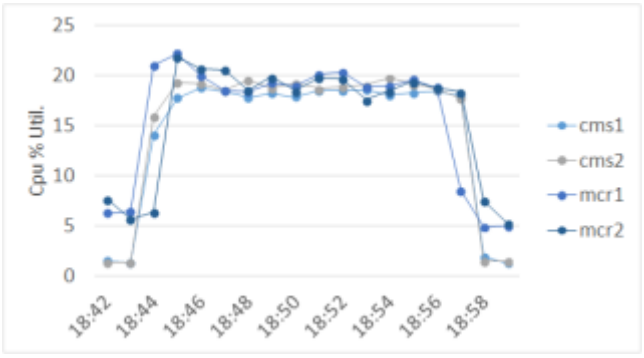


Figure 18 – DSS Scenario 2 – CMS and MCR CPU usage

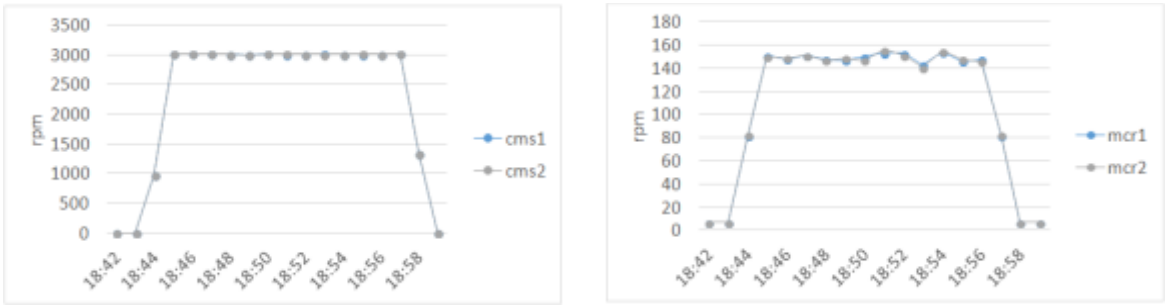


Figure 19 – DSS Scenario 2 - CMS and MCR request count

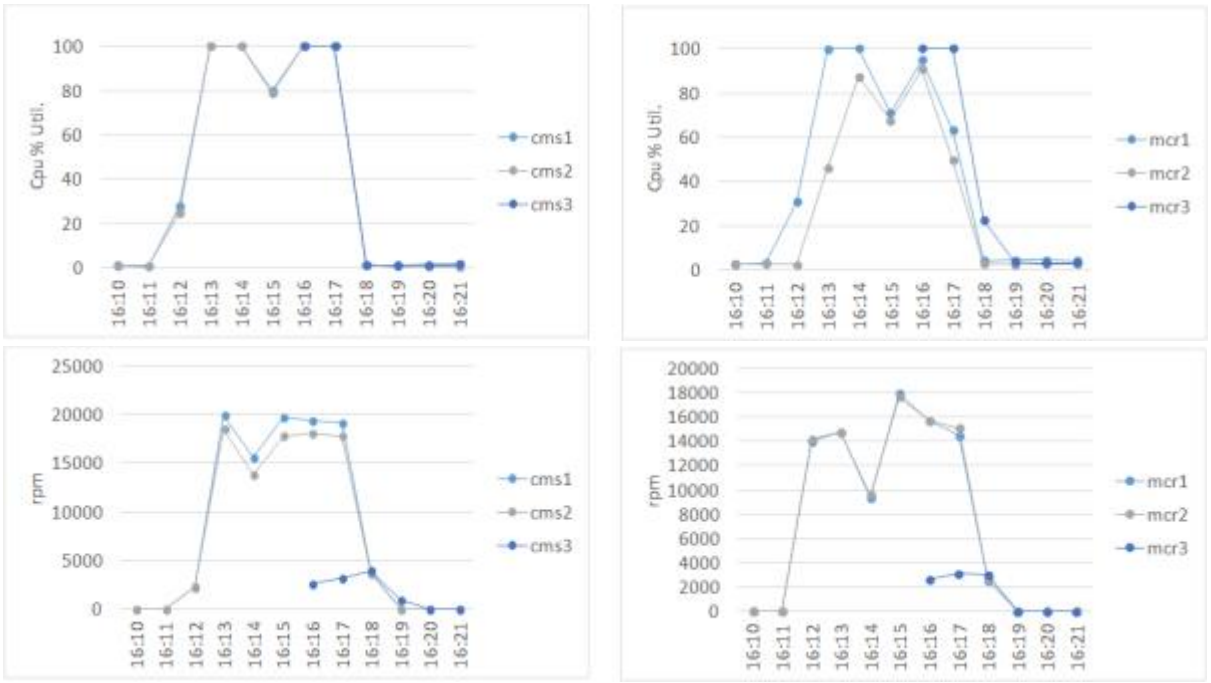
4.3.1.2.3 Scenario 2 - Fault management

In order to establish a clear separation between service behaviour with no faults and when a fault affects one specific component, a separate set of runs was performed to obtain fault management-related KPIs (see Section 4.3.1.5). Figure 20 shows a minor drop rate at the fault detection moment, due to the issue with the necessary recreation of the LB for the service (as explained in Deliverable D6.4). Nonetheless, the automatic recovery process works as expected, without relevantly affecting the service user experience.

Figure 20 – DSS Fault Recovery CPU and Requests for CMS and MCR components

4.3.1.2.4 Scenario 3 - Spike test

This scenario shows how all the available SICs reach their limit by considering the CPU utilization as the critical resource in this case. In fact, after a few seconds, response times increase to unacceptable values. At the same time, the service tries to react by performing a scale-out operation that causes a small drop rate at minute 16:15 (Figure 21). Although the new component also raises to 100% utilization, the number of requests processed is clearly smaller due to the component warmup time. There is also a direct relation between CPU and network resources, as expected. All the KPIs collected are presented in Section 4.3.1.7.



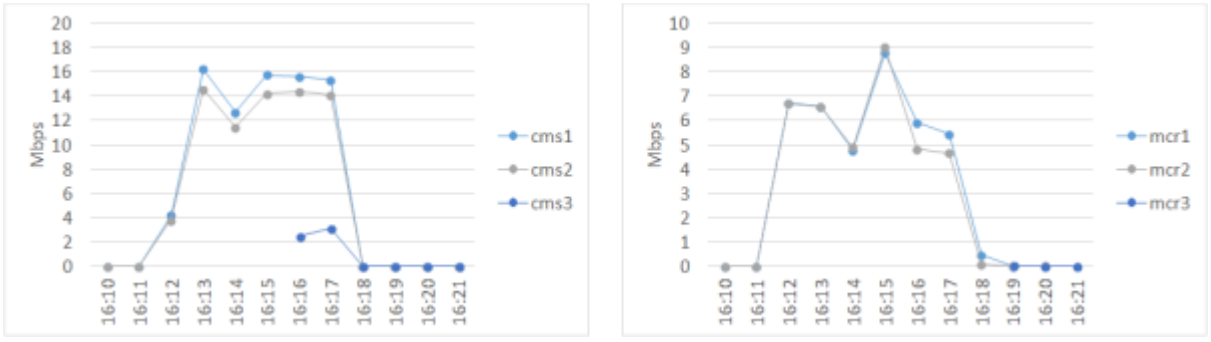


Figure 21 – DSS Scenario 3: CMS and MCR spike test

4.3.1.2.5 Scenario 4 - Stress test

Measured results show 3 scale-out operations for CMS, as clearly recognizable in Figure 22. The figure also shows that total performance of CMS increases after each scale-out operation. Temporary instability after each scale-out is due to the previously mentioned recreation of the load balancer. As presented in Figure 23, service stability is well recovered in about 5 minutes after each load increase.

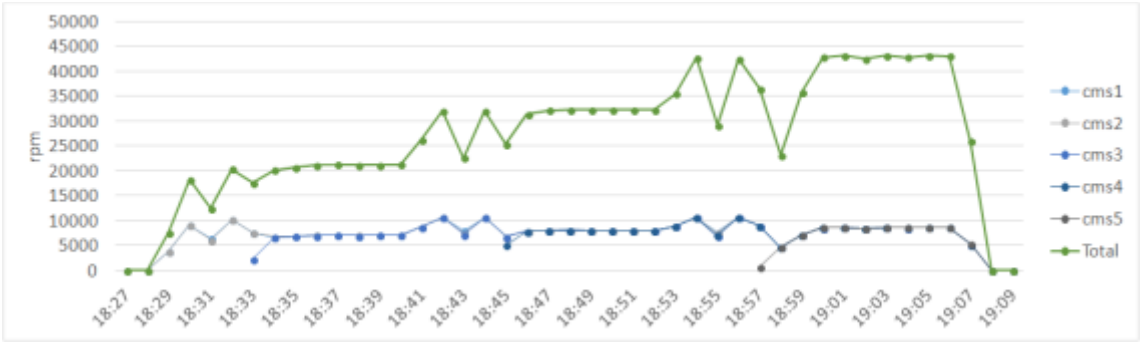


Figure 22 – DSS Scenario 4: Overall and per component CMS performance (Requests/minute)

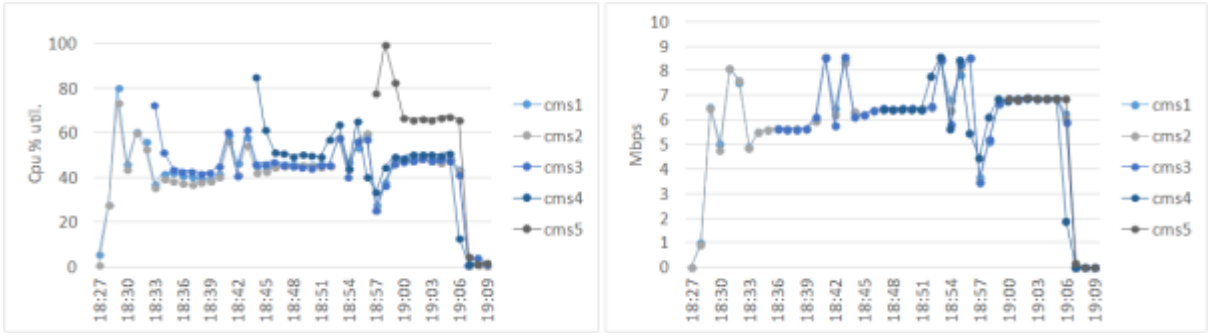


Figure 23 – DSS Scenario 4: CMS CPU usage and Network inbound

Two scale-out operations are also shown for the MCR component. Compared to the CMS component, after each MCR scale-out the testbed takes some additional time to stabilize. This is mainly caused by the time required to perform media file synchronization with the existing nodes by the peer to peer service. Therefore, the required time for the component to be available will be also be dependent on the size of the replica set. After the final scale-out (at 10:49 instant), the load is properly balanced between all components: at this stage the overall performance does not increase for MCR only due to the

incidental fact that the testbed outbound network is limited and under saturation. The KPIs collected during the execution of scenario are presented in Section 4.3.1.5.

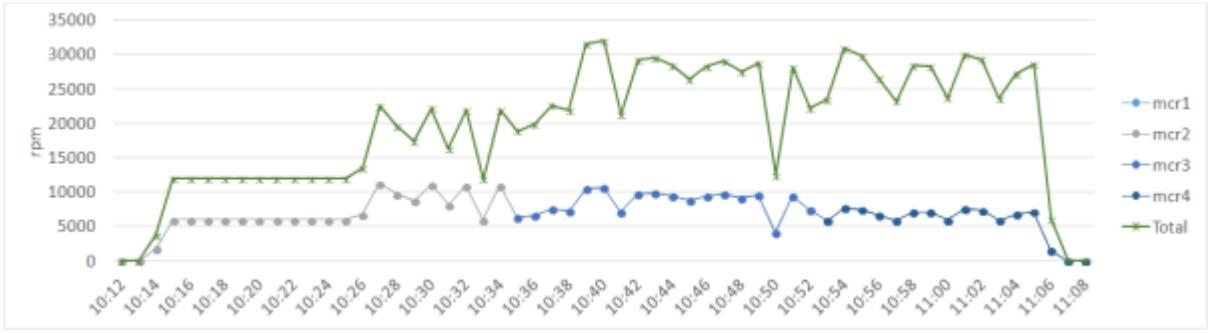


Figure 24 – DSS Overall and per component MCR performance (Requests/minute)

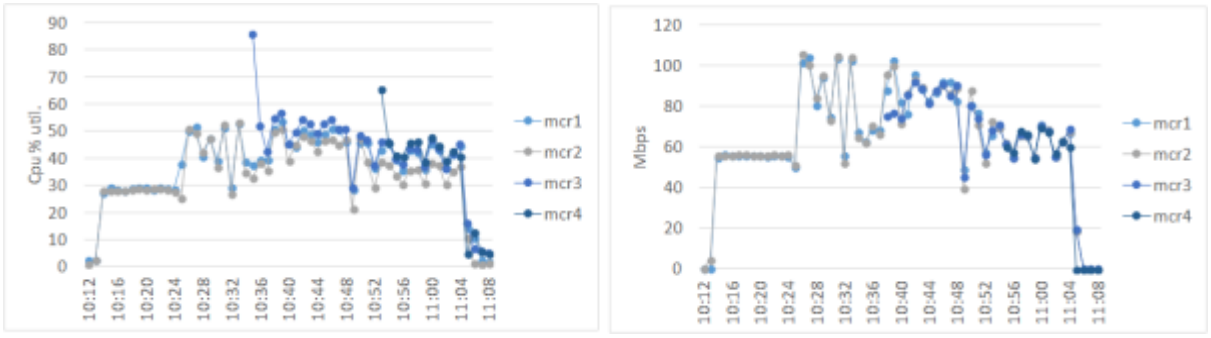


Figure 25 – DSS CPU usage and Network outbound per MCR component

4.3.1.2.6 Scenario 5 – RAVA-based Orchestration

In order to utilize the RAVA orchestration method for optimized management decision, specific initial conditions are configured. RAVA analytic epoch has been configured to perform a 10-minute analysis of both the DSS SICs in Bart-1 and for all the other generic SICs in the other (Bart) physical nodes. Monitoring information of the Bart nodes and the respective SICs in them are obtained through our MaaS. Emulated load for CPU and network for the SICs is generated using the Linux stress utility program and iperf tool, respectively. Load details are specified in the configuration dataset already presented in Table 21.

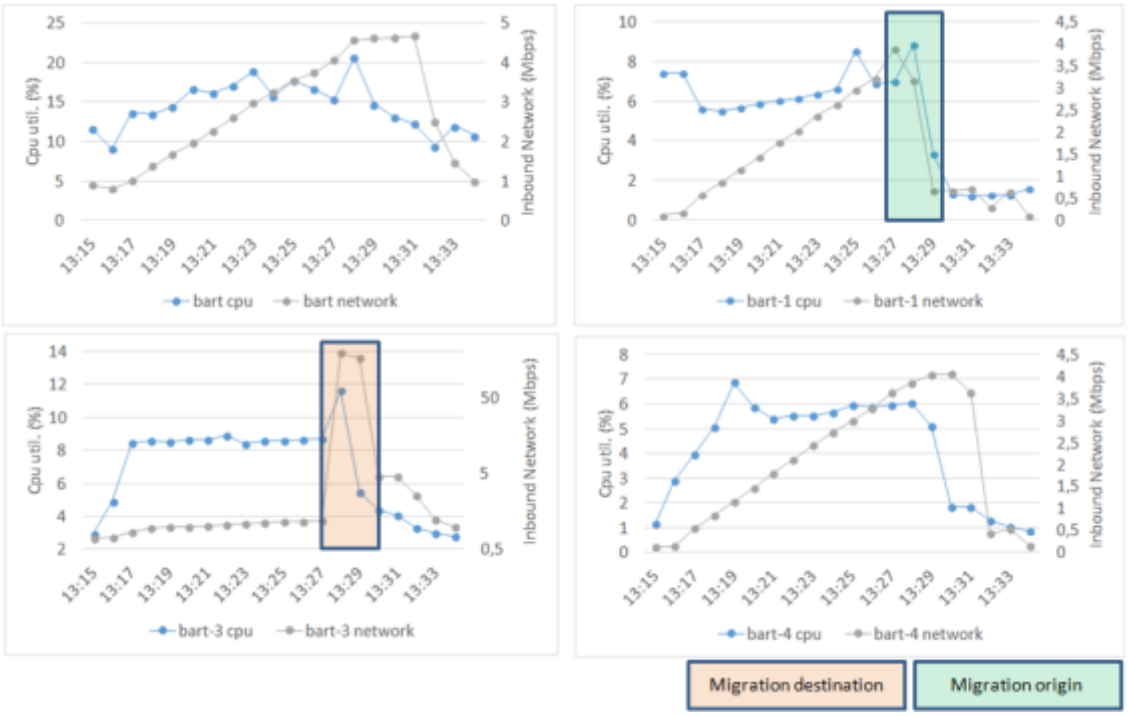


Figure 26 – DSS Bart physical nodes Network Inbound and CPU usage

The load that is emulated for the generic SICs in Bart-3 makes this node exhibit a lower affinity between CPU and network inbound resources, and therefore the best candidate amongst the other Bart nodes for migration destination in order to sustain the long-term load requirement of the target SIC. Since load is balanced equally between both CMS SICs, any of them is equally suitable to be a target SIC that can be migrated since the RRAS will be very similar for both. As can be seen in Figure 26 and Figure 27, the RAVA DE selects CMS1 as the target-SIC and the Bart-3 node as the candidate destination node for migration as expected. During the migration operation, the load on Bart-1 decreases and load on Bart-3 increases as new workload is introduced to Bart-3 due to the migration of the target SIC (i.e., CMS1). The migration operation is triggered after 10 minutes, thus making the network usage of the physical node increase in order to complete migration process. During migration no packet drops were observed and no request is either lost or failed. All KPIs captured are presented in Section 4.3.1.6.

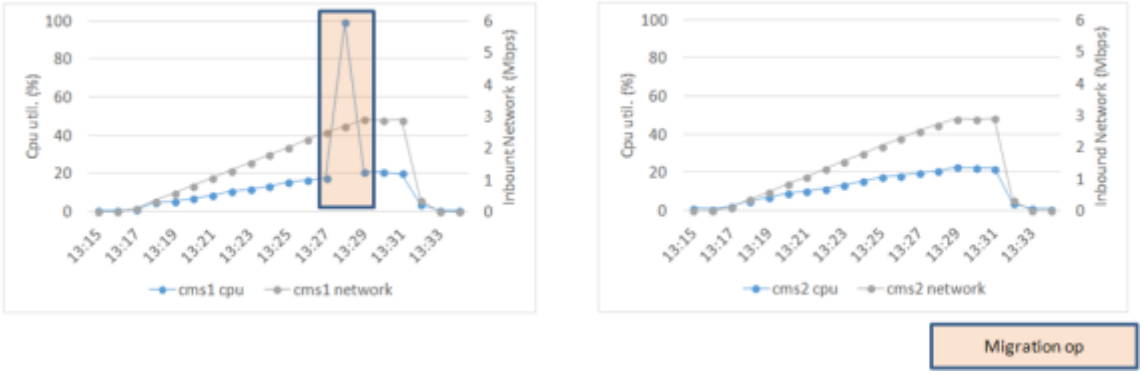


Figure 27 – DSS CMS Component CPU and Network Inbound

4.3.1.2.7 Scenario 6 - Service disposal

The scenario considers the disposal of an existing service with 2 CMS and 2 MCR components. KPIs are obtained from 5 different scenario executions. No additional considerations were needed to be taken into account for this scenario, which is in line with the previous descriptions. The collected KPIs are presented in Section 4.3.1.4.

4.3.1.3 System KPIs

As already described, system KPIs provide metrics for service quality evaluation. In this individual service evaluation work, they are considered according to Table 22.

Table 22 – DSS System KPIs

KPIs	Units	Scenarios	KPI measurement
Registration success rate	Rate (%)	2,3,4	Scenario 2: 100% Scenario 3: 100%* Scenario 4: 100%*
Session establishment success rate	Rate (%)	2,3,4	Scenario 2: 100% Scenario 3: 84% Scenario 4: 98%
Session drop rate	Time	2,3,4	Scenario 2: 0 seconds downtime (0%) Scenario 3: 12 seconds downtime (32% during that 12s) Scenario 4: 36 seconds downtime (25% during that 36s)
Attachment delay	Time (ms)	2,3,4	Scenario 2: 238 ms (avg) Scenario 3: 493 ms (avg) Scenario 4: 244 ms (avg)
Session establishment delay	Time (ms)	2,3,4	Scenario 2: 14 ms (avg) Scenario 3: 241 ms (avg) Scenario 4: 12 ms (avg)
Data plane QoS	Rate (%) Speed (Mbps)	2,3,4	Scenario 2: • TCPERR tcp_stream = 1.37% • TCPERR VoD = 1.98% • Throughput tcp_stream = 606Kbps • Throughput VoD = 10.1Mbps Scenario 3: • TCPERR tcp_stream = 1.33% • TCPERR VoD = 2.01% • Throughput tcp_stream = 585Kbps • Throughput VoD = 9.8 Mbps Scenario 4: • TCPERR tcp_stream = 1.35% • TCPERR VoD = 1.90% • Throughput tcp_stream = 608Kbps • Throughput VoD = 10.2Mbps

Data plane delay	Time (ms)	2,3,4	Scenario 2: 15ms (avg) 198ms (max) Scenario 3: 20ms (avg) 190 ms (max) Scenario 4: 18ms (avg) 196 ms (max)

Data plane QoS measurements have been taken as independent user-perceived quality by retrieving the KPI measurements as an EEU. Results show that user-perceived quality is not heavily affected even in scenario 3 as long as the component is still responding. It is important to mention that for scenario 3, after the involved component reaches its limit, QoS values cannot be measured since the service does not respond due to the global load generated. Measured droprate is limited to 12 seconds at the specific time when scaling operation occurs in scenarios 3 and 4. Although connection requests during that time are rejected, overall behaviour can be considered acceptable since the auto recovery mechanisms implemented in the EU (player) side do perform recovery of the content retrieval sessions fast enough to make it imperceptible for the viewer.

4.3.1.4 Lifecycle KPIs

Lifecycle KPIs aims to measure the behaviour of a DSSaaS instance by specifically focusing on deployment, provisioning, disposal, and installation times. The KPIs considered in our experimental evaluation work are reported in Table 23 and have been measured based on the average of 5 different deployments in the Bart testbed.

Table 23 – DSS Lifecycle KPIs

KPIs	Units	Scenarios	KPI measurement
Installation duration	Time (m)	any	44 minutes per component
Deployment and configuration duration	Time (s)	1	Deployment: 70.90 s Configuration: 137.15 s Total: 208.05 s
Disposal duration	Time (s)	6	33.2 s
Service upgrade duration	Time (s)	1	241.25s

Installation duration considers the required time to manually push a new service component version to the platform and make it available for the creation of a new service instance. Installation is considered a manual process and the time is strongly related to the network bandwidth, since a new service image has to be pushed to the testbed.

Service deployment and configuration is the total time the DSSaaS instance needs to be available for the EEU. Service disposal duration is the time required to completely remove a Service instance and release all its resources, including orchestration. Service upgrade duration is the time required for the whole service to be recreated, considering the new images that have been previously pushed to the testbed.

4.3.1.5 Cloud-native KPIs

Cloud-native KPIs aim to measure the behaviour of a DSSaaS instance by specifically focusing on automated cloud operations to achieve service elasticity and fault tolerance. Considered KPIs are depicted in Table 24.

Table 24 – DSS Cloud-native KPIs

Cloud native KPIs	Units	Scenarios	KPI measurement
Single component deployment latency	Time (s)	1	18.44 s
Scale in/out management latency	Time (s)	2,3,4	Scale out: • New SIC deployment: 24 s • New SIC provision: 118 s • Total scale out latency: 142 s Scale in: • Disposal SIC: 26 s
Scale in/out operation number	Num #	2,3,4	Scale out • Scenario 2 (stable load): 0 • Scenario 2 (fault): 0 • Scenario 3: 2 • Scenario 4: 7 Scale in: • Scenario 2 (stable load): 0 • Scenario 2 (fault): 0 • Scenario 3: 2 • Scenario 4: 7
Useless scale operation number	Num #	2,3,4	Scenario 2 (stable load): 0 Scenario 2 (fault): 0 Scenario 3: 0 Scenario 4: 0
Time percentage of control plane availability	Rate (%)	2,3,4,5	Scenario 2 (stable load): 100% Scenario 2 (fault): 99% Scenario 3: 84% Scenario 4: 98%

Except where explicitly stated, the reported measurements have been taken based on the average of the results obtained while running all the specified scenarios. Single component deployment latency considers only the time required to deploy a new SIC. Scale-out deployment takes some extra additional time since our scale-out process requires the creation of the component and of the network port, as well as the attachment of the load balancer (LBaaS).

4.3.1.6 Extended Cloud-native KPIs for RAVA evaluation

Extended cloud-native KPIs have been introduced with the goal of measuring the behaviour of a DSSaaS instance by specifically focusing on live migration of individual service components performed by the RAVA management/orchestration migration operations and fault management described in the innovation section. The associated KPIs are listed in Table 25.

Table 25 – DSS Extended Cloud-Native KPIs for RAVA Evaluation

Cloud native KPIs	Units	Scenarios	KPI measurement
Service component migration latency	Time (s)	5	38 s
Service component migration operations number	Num #	5	1
Migration drop rate	Rate (%)	5	0% (100% availability)
Fault SIC recovery drop rate	Rate (%)	2	1% (99% availability)
Fault SIC recovery latency	Time (s)	2	Detection: 120 s Deployment of new SIC: 29 s Conf. of new SIC:109 s

4.3.1.7 Resources KPIs

Resource KPIs aim to measure the behaviour of a DSSaaS instance by concentrating the attention on hardware resource consumption of individual service components. The considered KPIs are reported in Table 26.

Table 26 – DSS Resource KPIs

KPIs	Units	Scenarios	KPI measurement
Cpu usage	CMS Num vcpus (#) CMS Usage rate (%) MCR Num vcpus (#) MCR Usage rate (%)	2,3,4	Scenario 2 (stable load): • CMS Num vCpus: 2 • CMS Usage rate (%): 34% of 200% • MCR Num vCpus: 2 • MCR Usage rate (%): 42% of 200% Scenario 3: • CMS Num vCpus: 3 • CMS Usage rate (%): 276% of 300% • MCR Num vCpus: 3 • MCR Usage rate (%): 279% of 300% Scenario 4: • CMS Num vCpus: 5 • CMS Usage rate (%): 265% of 500% • MCR Num vCpus: 4 • MCR Usage rate (%): 220% of 400%
Memory usage	CMS Reserved (IaaS)(MB) CMS Used (MB) MCR Reserved (IaaS) (MB) MCR Used (MB)	2,3,4	Scenario 2 (stable load): • CMS Reserved: 4000 • CMS Used: 1710 • MCR Reserved: 4000 • MCR Used: 1902 Scenario 3: • CMS Reserved: 6000 • CMS Used: 3567 • MCR Reserved: 6000

			• MCR Used: 3610 Scenario 4: • CMS Reserved: 10000 • CMS Used: 5250 • MCR Reserved: 8000 • MCR Used: 3200
Network usage	CMS Inbound (Kbps) MCR Inbound (Kbps) CMS Outbound (Kbps) MCR Outbound (Kbps)	2,3,4	Scenario 2 (stable load): • CMS Inbound: 4558 • MCR Inbound: 3797 • CMS Outbound: 2082 • MCR Outbound: 329111 Scenario 3: • CMS Inbound: 43134 • MCR Inbound: 15096 • CMS Outbound: 24556 • MCR Outbound: 298626 Scenario 4: • CMS Inbound: 32010 • MCR Inbound: 13924 • CMS Outbound: 19832 • MCR Outbound: 281968
External network usage	Inbound (Kbps) Outbound (Kbps)	2,3,4	Scenario 2 (stable load): • CMS Inbound: 4412 • MCR Inbound: 3480 • CMS Outbound: 1940 • MCR Outbound: 322741 Scenario 3: • CMS Inbound: 41812 • MCR Inbound: 14976 • CMS Outbound: 22880 • MCR Outbound: 292117 Scenario 4: • CMS Inbound: 30019 • MCR Inbound: 11975 • CMS Outbound: 18441 • MCR Outbound: 280047
Storage usage	CMS (GB) MCR (GB)	2,3,4	Scenario 2 (stable load): • CMS: 3.98 of 39.3 • MCR: 6.02 of 39.3 Scenario 3: • CMS: 5.97 of 58.95 • MCR: 6.63 of 58.95 Scenario 4: • CMS: 9.95 of 98.25 • MCR: 8.84 of 78.6

Number of VMs used	Num (#)	2,3,4	Scenario 2 (stable load): • CMS VMs: 2 • MCR VMs: 2 Scenario 3: • CMS VMs: 3 • MCR VMs: 3 Scenario 4: • CMS VMs: 5 • MCR VMs: 4
Resources consumed for orchestration	CPU Mem Storage Net	1,5	PaaS Service Orchestrator: • Cpu Usage: • Num. of vCpus: shared (best effort) • Usage rate (%): <0.3% • Memory Used: 33 MB • Network Inbound: <0.1 Kbps • Network Outbound: <0.1 Kbps • Storage: 16 MB • Num. of Lightweight Containers:1 IaaS SO Proxy: • Cpu Usage: • Num. of vCpus: 1 • Usage rate (%): 1% of 100% • Memory Reserved: 1000 GB • Memory Used: 216 MB • Network Inbound: 0.1 Kbps • Network Outbound: 0.1 Kbps • Storage: 881 MB of 5 GB • Num. of VMs: 1 VM

4.3.2 IP Multimedia Service – IMSaaS

The performance evaluation described in this sub-section has been performed by following the methodological guidelines provided in Deliverable D3.5 and adapting them to the specific characteristics of this individual service, showing again the general applicability and adaptability of the proposed evaluation methodology. In particular, the approach taken for IMSaaS considers the five dimensions proposed in D3.5, summarized here:

- Workload representing the amount of external work requests received by the system;
- System KPIs as the QoS offered to the subscribers;
- Resources in terms of compute, network, and storage required for a specific workload;
- Lifecycle KPIs related to the management and orchestration operations for the lifecycle of those network functions;
- Cloud-native KPIs mainly referring to high-availability and elasticity.

4.3.2.1 Scenarios and datasets

As already mentioned in Deliverable D5.4, in order to provide an evaluation based on the different dimensions proposed in D3.5, different scenarios have been prepared. Some of those scenarios have been already extensively evaluated and discussed in Deliverables D5.4 and D6.4. During the extension period new scenarios have been proposed and implemented for collecting additional measurements related to the high availability and media plane virtualization characteristics of the extended IMSaaS components.

In particular, multiple different scenarios are presented in the following subsections:

- Scenario 1 - The instantiation of the IMSaaS SI is performed by the EEU to the IMSaaS SM.
- Scenario 2 - Stairs workload. The IMS Service Instance is running. Artificial load executing different types of scenarios for Registration and Invite messages is generated by using the widely adopted and well-known IMS Bench Tool. This scenario is used mainly for evaluating the signalling process, and some of its results have been already presented into D5.4.
- Scenario 3 - Linear workload on Media Plane components. The IMS Service Instance is running. Artificial load executing a linearly increasing number of Invite sessions is generated by using the widely adopted and well-known SIPp Bench tool. This scenario is used mainly for evaluating the Media Plane over different environments.
- Scenario 4 - Spike workload on Media Plane components. The IMS Service Instance is running. Artificial load executing a spike number of Invite sessions is generated using the SIPp Bench tool. This scenario is used mainly for evaluating the Media Plane over different environments.
- Scenario 5 - Stairs workload on Media Plane components. The IMS Service Instance is running. Artificial load executing a stairs-like increasing number of Invite sessions is generated using the SIPp Bench tool. This scenario is used mainly for evaluating the Media Plane over different environments.
- Scenario 6 - Constant workload on Media Plane components. The IMS Service Instance is running. Artificial load executing a constant number of Invite sessions is generated using the SIPp Bench tool. This scenario is used mainly for evaluating the Media Plane over different environments.
- Scenario 7 - Disposal of the IMSaaS SI is performed by the EEU.

Scenario 1 and 7 are mainly used for evaluating Lifecycle KPIs, while scenarios 3 to 6 are used for evaluating system KPIs and resource KPIs of the Media Plane component on top of virtualized vs hardware-based environment.

In almost all the presented scenarios (except those executed up to M36 and already reported in D6.4), we used the Open Baton NFV Orchestrator. In fact, Open Baton allows easily reconfiguring the Network Service Descriptor (basically the STG in the MCN terminology). In particular, for scenario 1 and scenario 7, we employed the 1:1 approach in which each of the SICs required (SCSCF, PCSCF, ICSCF, HSS, DNS, MGW, MMAS) for providing a standalone IMS service were deployed on separated VMs. Scenario 2 was slightly different from the previous one as it includes also the SLF and DB SICs in order to provide high-availability at the HSS level. For scenarios 3 to 6 we have decided to deploy only the MGW component, as the main goal was to evaluate the performance results of the single component from the data plane perspective.

4.3.2.1.1 Scenario 1: Deployment and provisioning

In this first scenario a complete IMSaaS SI is deployed and provisioned. In the table below you can find the list of KPIs collected. This scenario was executed five times.

Table 27 – KPIs collected in scenario 1

KPIs	Adaptation / Detail
KPI 9 - Deployment and configuration duration	Time to deploy and provision a service instance
KPI 12 - Single component deployment latency	Time to deploy each SIC composing service
KPI 31 - Number of VMs used	Number of VMs used (IMS composed services)
KPI 32 - Resources consumed for orchestration	HW resources consumed for SO and SM

4.3.2.1.2 Scenario 2: Stairs workload

This second scenario has been used for evaluating the primary performance results of the control plane. In particular, the scenario consists of a mixed operating situation including registrations and call setups.

Table 28 – KPIs collected in scenario 2

KPIs	Adaptation / Detail
KPI 9 deployment and configuration duration	Time to deploy and provision a service instance
KPI 13, KPI 14 Scale in/out management latency (ctrl plane)	Scale in/out perceived user time for each SIC
KPI 15, KPI 16 Scale in/out operation number (ctrl plane)	Number of scale in out operations per SIC
KPI 17 Useless scale operation number (ctrl plane)	Number of ping-pong scaling ops
KPI 23 Time percentage of control plane availability	Availability after initial provisioning performing scaling ops / component failure

4.3.2.1.3 Scenario 3: Linear workload

This scenario has been used for evaluating the performances indicators of the data plane. As shown in Table 29, two different variations of this scenario have been implemented. Modifying the increase rate in this scenario, calculated in terms of calls per every 5 seconds (Cp5s), allows to know better the performance limitations of the media plane components, especially because it causes a different usage of the internal resources available at the component level.

Table 29 – Linear workload scenarios

Scenario	Initial Cp5s	Increase Rate	Step duration
Linear-1	5	1	5
Linear-2	5	2	5

Figure 28 graphically depicts the considered scenarios. As it can be seen, Linear-2 goes faster to around 150 Cp5s set as the maximum number of Cp5s which could be handled by the media plane component.

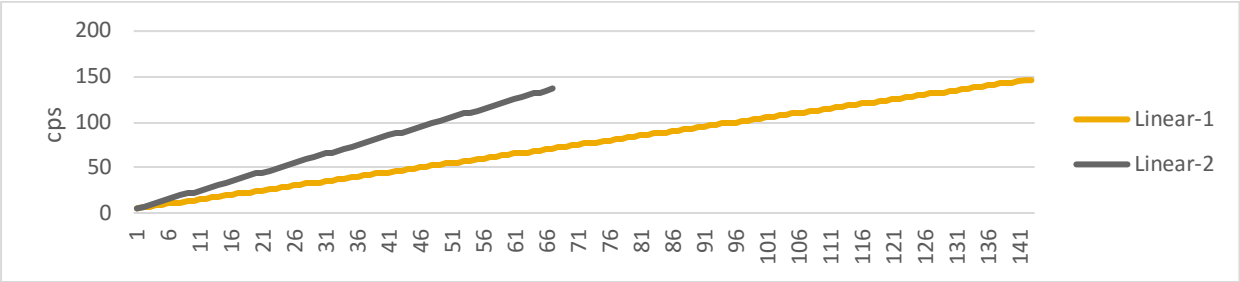


Figure 28 – Number of Cp5s in both scenarios

Table 30 identifies the KPIs collected in the linear scenarios. Considering that the main focus is on media plane performance evaluation, the measured KPIs belong mostly to the classes of system and resource KPIs. This table is also valid for the next three scenarios, exposed in the next three sections.

Table 30 – KPIs collected

KPIs	Adaptation / Detail
KPI 2 - session establishment success rate	Session establishment: Subscriber connection request success & content response
KPI 3 - session drop rate	Number of dropped sessions
KPI 4 - attachment delay	Registration delay
KPI 5 - session establishment delay	Time required to perform an INVITE
KPI 6 - data plane QoS	QoS parameters related with the data plane
KPI 26 - CPU usage	CPU utilization per SIC
KPI 27 - Memory usage	Memory usage per SIC
KPI 28 - Network usage	Inbound/outbound network usage per SIC
KPI 31 - Number of VMs used	Number of VMs used (IMS composed services)

Another linear scenario (linear-elastic) was used for evaluating the prototypal deployment of the elastic scalability of the MGW function in MCN. The SIP messages flow in (cf. Deliverable D5.5, Figure 7), starts from UAC and reaches the UAS passing through the SIP AS and the MGW, while the media flow (RTP packets) only flows through the MGW starting from UAS to UAC. In addition, to run these evaluation tests, we used the same software configurations used to run the tests presented above. The only notable difference lies in the fact that these experiments have been run on the less performant and more overloaded (by other ongoing MCN experiments) Bart MCN playground. Since the goal of the experimental results shown below was not taking performance measurements, but rather challenging our elastic prototype over a realistic cloudified environment, that testbed was more than adequate.

In our experiments, we focused on the evaluation of the call phase (CALL in the following), represented mainly by the SIP INVITE messages dialogue, plus all the others messages needed to establish a proper and valid SIP session and then to allow for the load sharing among available MGW instances. The whole protocol is detailed in Deliverable D5.5 (in particular in the protocol description shown in Fig. 9), with

the only addition of the BYE message sent at the end of the interaction between the UAS to the UAC to close the call.

In particular, we tested the CALL session loading the system with the linear workload (cf. Figure 29) that presents as initial cps 1, an increase rate of 1, and a step duration equal to 30 seconds; in other words, every 10 seconds a new call is started and the call rate increases by one every 30 seconds.

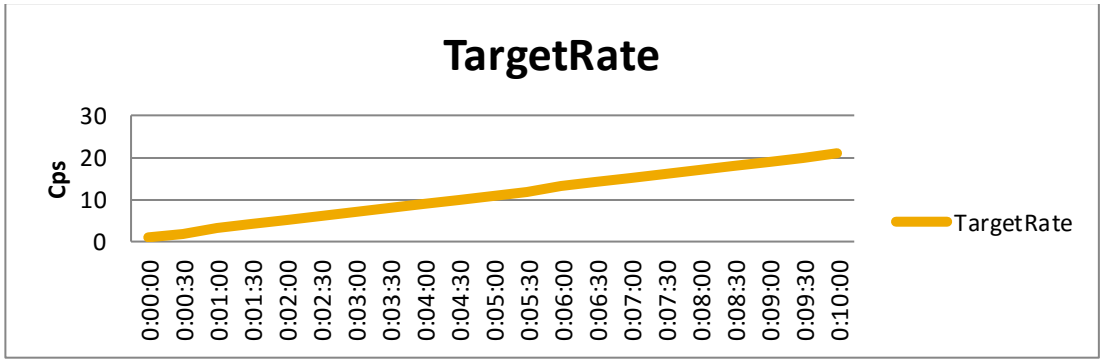


Figure 29 – Target rate of the linear elastic

4.3.2.1.4 Scenario 4: Spike workload

This scenario has the same primary purpose of the previous one, by focusing on media plane components performance evaluation. However, it specifically concentrates on sending a high number of requests for a very short period of time. In this way it is possible to analyse situations in which there is an unexpected immediate growth in the number of participating users, which is deemed industrially relevant in several situations. Table 31 shows the number of call per 5 seconds executed, and its duration.

Table 31 – Spike workload scenario

Scenario	Initial Cp5s	Increase Rate	Step duration
Spike	100	0	180

Figure 30 depicts the considered scenarios. As can be seen, there is a high number of call per second just for a very short interval.

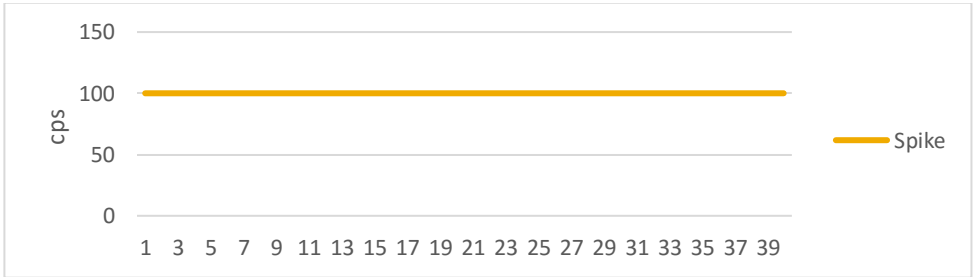


Figure 30 – Spike workload scenario

4.3.2.1.5 Scenario 5: Stairs workload

This scenario proposes the stairs workload and is composed by four different variations. The main objective is to be able to evaluate the media plane components in situations where the traffic grows with different increase rates and speeds. Table 32 provides some details about the settings and datasets used for the different scenarios.

Table 32 – Stairs workload scenarios

Scenario	Initial Cp5s	Increase Rate	Step duration
Stairs-1	1	5	30
Stairs-2	1	10	60
Stairs-3	1	20	120
Stairs-4	1	30	180

As shown in Figure 31 , there is a high number of call per every 5 seconds just for a very short interval.

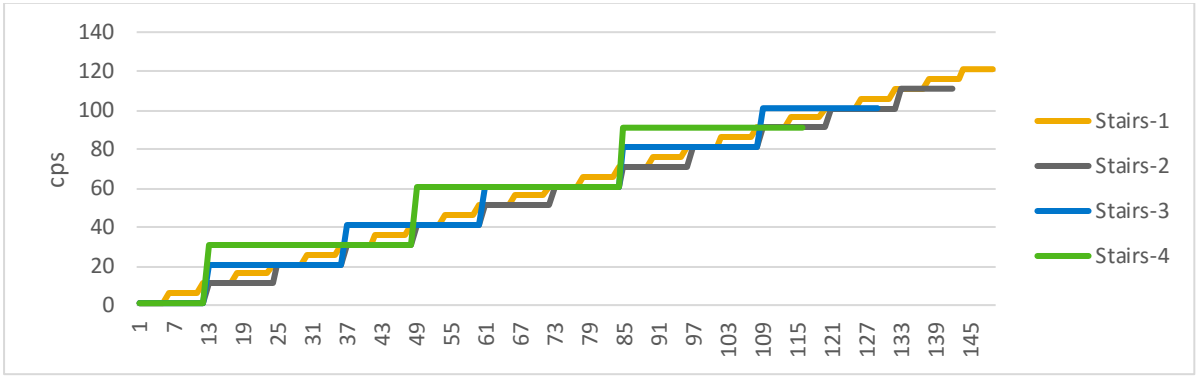


Figure 31 – Stairs scenarios workload

4.3.2.1.6 Scenario 6: Constant workload

This scenario proposes the constant workload, having as main objective the evaluation of the media plane components in situation of constant load. Table 33 shows the details about the rate of cps utilised in this scenario, and Figure 32 presents its graphical view.

Table 33 – Constant Scenario Workload

Scenario	Initial Cp5s	Increase Rate	Step duration
Constant	80	0	600

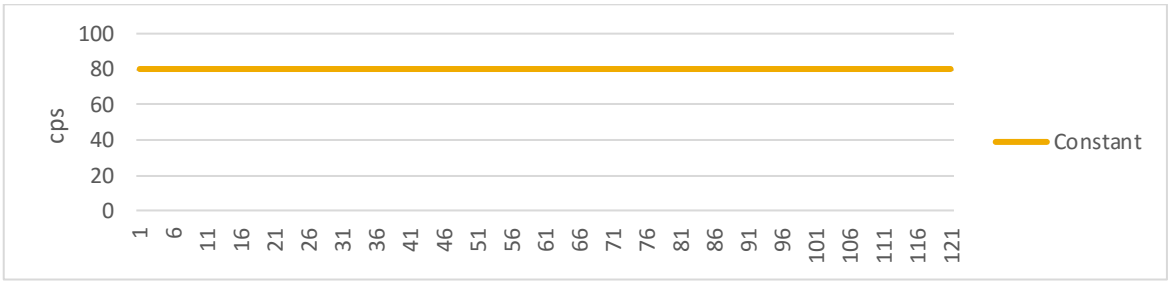


Figure 32 – Constant workload scenario

4.3.2.1.7 Scenario 7: Disposal

In this last scenario the service is simply disposed. For obvious motivations, it is evaluated in terms of lifecycle and resource KPIs.

Table 34 – KPIs collected in scenario 7

KPIs	Adaptation / Detail
KPI 10 – Disposal Duration	Time to dispose a service instance
KPI 31 - Number of VMs used	Number of VMs used (IMS composed services)
KPI 32 - Resources consumed for orchestration	HW resources consumed for SO and SM

4.3.2.2 Scenarios execution

This section presents the results obtained by the execution of the scenarios introduced into the previous section for the IMSaaS SI.

4.3.2.2.1 Scenario 1: Deployment and provisioning

Within this scenario we have experimentally measured the time needed for deploying and provisioning an IMS SI from the EEU perspective. Therefore, times were taken from the moment a EEU issues a request to deploy the IMS SI, to the moment the instantiation and provisioning is completed with a CREATE_COMPLETE status. Results over the usual 5 runs are shown in Figure 33.

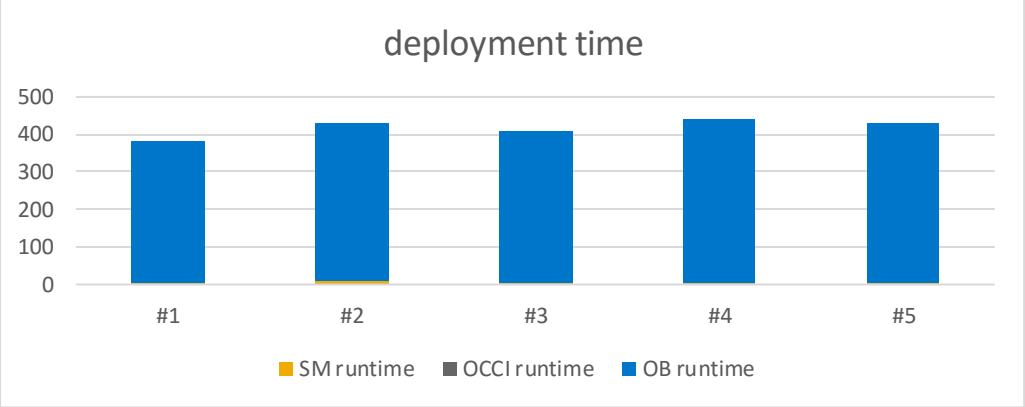


Figure 33 – KPI 9 results for IMS SI

As it can be noticed from Figure 33, the average time for deploying a new IMS SI is about 419 seconds. This time has been also divided in three major steps, which are executed while deploying the new SI, in particular when including the OCCI Adapter on top of the Open Baton Orchestrator. The results have shown that most of the time is taken by the Orchestrator for deploying the SICs. It should be clarified that, for this particular scenario, it has been used an empty Ubuntu image and the IMS SIC software has been downloaded from the Internet and installed on the fly by the NFVO. Basically we re-created the typical situation in which a new datacentre is instantiated and an EEU decides to deploy a new IMS SI. Obviously, the reported indicators can be easily optimised including the software binaries of the IMS SIC in the images of the new datacentre (already the case for the MGW component), which was not anyway the purpose of this performance evaluation. Figure 34 shows the results in terms of KPI 12.

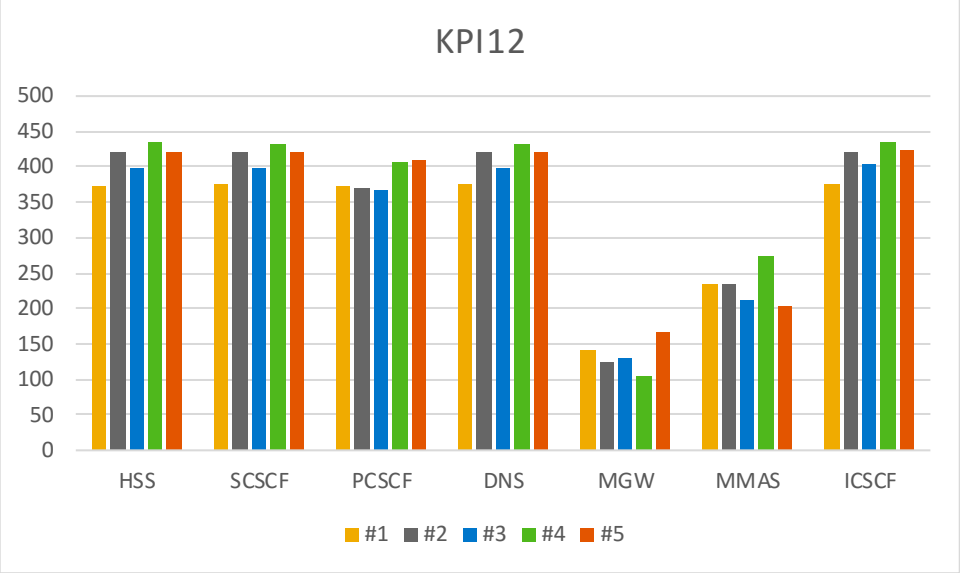


Figure 34 – KPI 12 (deployment latency) results from IMS SI

As noticeable, most of the time is employed for deploying the IMS signalling components. Another aspect that has demonstrated to significantly impact on the SIC instantiation time is the dependency from other SICs. Whenever a SIC requires runtime parameters of another SIC, it will need to wait up to the moment this parameter has been generated by the NFV Orchestrator. A typical example could be the PCSCF requiring the dynamic DNS record of the ICSCF. This information is available only after the ICSCF component is instantiated, therefore the PCSCF VNFM has to wait for the instantiation lifecycle termination of the ICSCF before configuring the PCSCF SIC.

Table 35 shows the results from the resource KPIs perspective. In particular, it can be noticed that it is enough a set of 7VMs for providing a basic IMS Service. In addition to this, it was employed a VM for the Orchestration components which consumed additional CPU resources.

Table 35 – KPIs collected in scenario 1

KPIs	Description	Value
KPI 31 - Number of VMs used	Number of VMs used (IMS composed services)	7 VMs
KPI 32 - Resources consumed for orchestration	HW resources consumed for SO and SM	4GB Memory 4vCPUs

4.3.2.2.2 Scenario 2: Stairs workload

This comprehensive scenario test has been performed in order to validate our Fault Management (FM) system against repeated failures. As already mentioned in Deliverable D5.5, a failure in the virtualized compute layer may cause the triggering of other alarms (Alarm Storming).

In one hour, we simulated 6 VM failures and the FM System performed after several Switch-to-Standby actions, in order to restore the normal situation. VNF failures obtained during the VM failure were properly ignored.

The average of the total latency, from the occurrence of the failure till the recovery of the service, has been calculated during the final test and illustrated in Figure 35. The reported results are the average of the six cases of failure emulated in our tests.

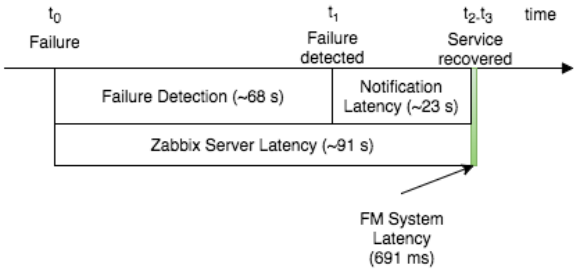


Figure 35 – Fault Management System KPIs

Regarding Figure 35, the interval t_0–t_1 represents the time between the occurrence of the failure and its detection by the Zabbix Server. Such interval can be reduced/optimized by tuning the delay of the item via proper configuration of the Zabbix monitoring "greediness". In the considered scenario, the Zabbix Server updates the monitored indicators every 60 seconds, which is considered very usual for many related deployment environments (limited overhead with acceptable latency introduced in management operations).

The interval time t_1–t_2 is the latency introduced by the Zabbix Server to send the notification to the Monitoring Driver. Ideally, the Zabbix Server should only get the information of the trigger having the problem, and execute the alert script we created. Although executing the alert script takes a bunch of milliseconds, the notification is received by Monitoring Driver not before 23 seconds. Such interval could not be reduced since it has been experimentally demonstrated to be due to the internal mechanism used by the Zabbix Server to send notifications.

The interval time t_2–t_3 is the overhead introduced by the fault restoration procedure. In this interval different components of Open Baton are involved, including the FM system. The interval time t_2–t_3 has been analyzed and the results about its sub-components are reported in the following figure. The total overhead is 691ms, due to the different actions performed by the different components as succinctly described above.

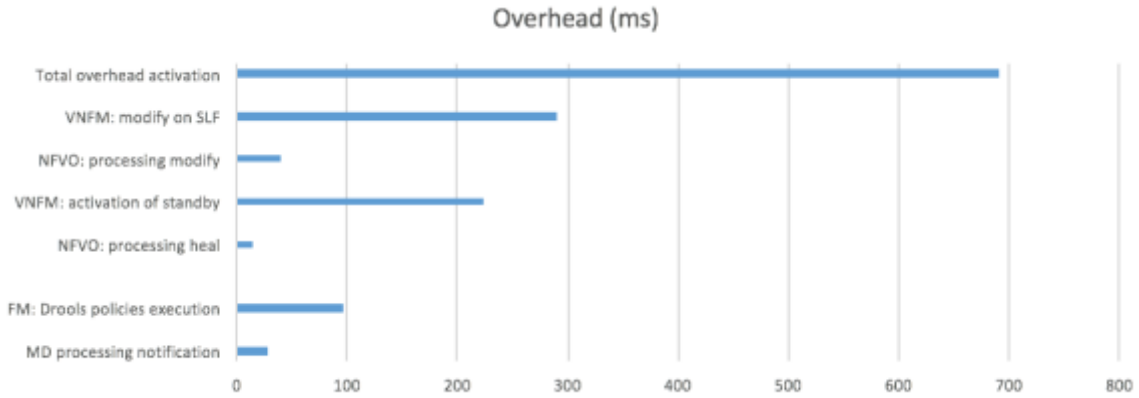


Figure 36 – Overhead for the execution of the switch-to-standby action from the Orchestrator perspective

The Monitoring Driver introduces an overhead of 28ms due to the mapping of the Zabbix Server notification to the standard VirtualizedResourceAlarmNotification. The FM system introduces an overhead of about 96ms, in which the major fraction has demonstrated to be due to the execution of Drools rules.

Then, the NFVO processes the Switch-to-Standby action by sending a message with the cause "switch-to-standby" to the Generic VNFM in about 15ms. The overhead introduced by the VNFM during the activation of the standby VNFC instance is ~220ms, mostly caused by the execution of the script to start the Apache Web server. After that, the NFVO resolves the needed dependencies and sends to the VNFM the SCALE_IN, MODIFY, and START actions in order to reconfigure the SLF. Such actions are grouped with an overhead of ~40ms. Please refer to Deliverable D5.5 for more details about the associated procedures.

Table 36 – IMS Fault Management Overhead results

Component/s	Overhead
Monitoring Driver	28 ms
FM System	96 ms
NFVO - VNFM	567 ms
Total	**691 ms**

The VNFM spends 290ms to perform such actions, mostly due to the execution of the scripts in the SLF virtual machine. The SLF needs to be reconfigured in order to work with the new dependencies, this demonstrating to introduce the major part of the overhead. Although the overhead of the FM system depends on the number of the rules to process, our evaluation activities show how the overhead of the FM system is largely acceptable.

Table 37 – KPIs collected in scenario 1

KPIs	Description	Value
KPI 13, KPI 14 - Scale in/out management latency (ctrl plane)	Scale in/out perceived user time for each SIC	Around 1 second
KPI 15, KPI 16 - Scale in/out operation number (ctrl plane)	Number of scale in out operations per SIC. While executing a switch to standby operation it is basically generating a scale out and scale in procedure in parallel.	6 scale in and 6 scale out
KPI 17 - Useless scale operation number (ctrl plane)	Number of ping-pong scaling ops	0
KPI 23 - Time percentage of control plane availability	Availability after initial provisioning performing scaling ops / component failure	Not applicable in this scenario as mainly devoted to management operations KPIs

4.3.2.2.3 Scenario 3: Linear workload

Before describing the results obtained in this scenario, it is important to clarify, as already mentioned in previous parts of this deliverable, that from scenario 3 to scenario 6 have been executed some benchmark workloads for evaluating the performance results obtained by a Media Plane Component running on top of virtualized resources compared to the one deployed directly on hardware. The Hardware resources

used for such experiments have been presented in Section 2.1.1.2, in particular with reference to the Fokus region deployment. One of these two Lenovo ThinkCentre instances have been configured with an OpenStack compute node running a KVM instance for evaluating the virtualized scenarios, while have been used directly for running the software version of the MGW (Asterisk, as discussed in Deliverable D5.5). On the other instance we have installed the IMS Benchmarking tool running a typical User Agent Client/Server scenario as presented in the previous section. For each of the 4 scenarios we present first the results obtained in the hardware instantiation, and then the virtualized ones.

Figure 37 shows KPI 2 and KPI 3 values (in terms of successful number of calls each 5 seconds) obtained executing the linear workload against an instance of a MGW installed directly on top of the Guest OS.

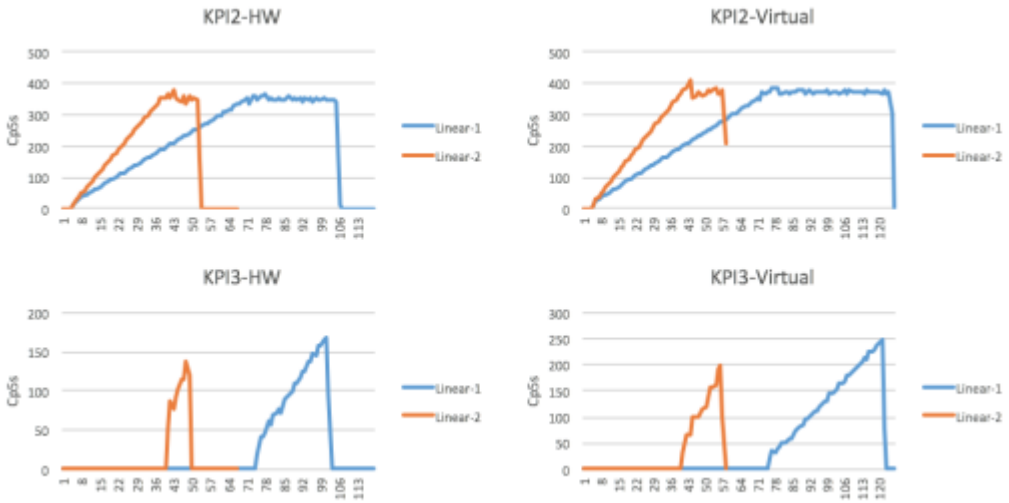


Figure 37 – KPI 2 and KPI 3 for a MGW - HW vs. virtualized

Table 38 shows the system KPIs measured as average of all scenarios (3-6), including Media Plane Components. In particular, those measurements show there is a difference of around 10% in jitter/RTT between the virtualised solution and the one deployed directly on hardware. This performance variation is still acceptable.

Table 38 – System KPIs for a MGW running on HW versus on a VM involving Media Plane

KPIs	Adaptation / Detail	Units	Values HW	Values Virtual
KPI 4 - attachment delay	Registration delay. Calculated from the client perspective, however having the benchmarking tool on the same local network of the IMS	Time (ms)	Avg: 5	Avg: 5
KPI 5 - session establishment delay	Time required to perform an INVITE. Calculated from the client perspective, however having the benchmarking tool on the same local network of the IMS	Time (ms)	Avg: 5ms	Avg: 5ms
KPI 6 - data plane QoS	QoS parameters related with the data plane. Calculated from the client perspective, however having the benchmarking tool on the same local network of the IMS	Time (ms)	Jitter (Avg): • RX: 0.322 • TX: 0.235 RTT: 0.194	Jitter (Avg): • RX: 0.460 • TX: 0.255 RTT: 0.210

4.3.2.2.3.1 Functional Test of the Elastic Scalability of the MGW

As mentioned above, in this scenario it has been executed a specific use case related to the scalability of the MGW component. We ran the test for ten minutes (sufficient time interval to observe stabilized behaviours) and obtained, with the rate shown above, a linear increment of the KPI 2 (UAC side) sampled on a period of 30 seconds. Figure 39 and Figure 40 show, respectively, the current ongoing calls at the UAC per period, and the number of successfully completed sessions, namely KPI 2 (Session establishment rate).

With the increasing of the target rate, we obtained a relevant number of ongoing current calls, namely, of calls started by the UAC, but still to be completed due to the duration of the call. Indeed, to make realistic tests, with a heavier load, we kept the call duration quite long (if compared with "regular" realistic calls), namely around 2 minutes. That is the reason why the first successful sessions complete at period 2 minutes 30 seconds (see Figure 40). In fact, our goal was to study the architecture behaviour and responsiveness of the MGW under stress (due to accumulated calls, through the MCN KPIs and metrics described in Deliverable D3.5).

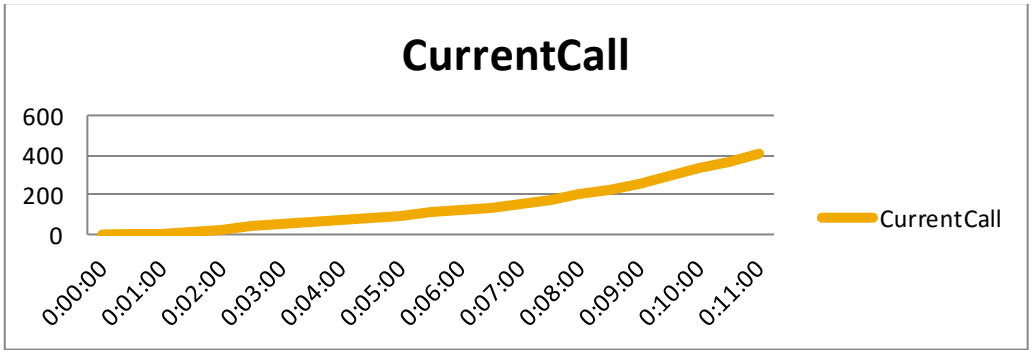


Figure 38 – Number of current calls at the UAC with one only MGW

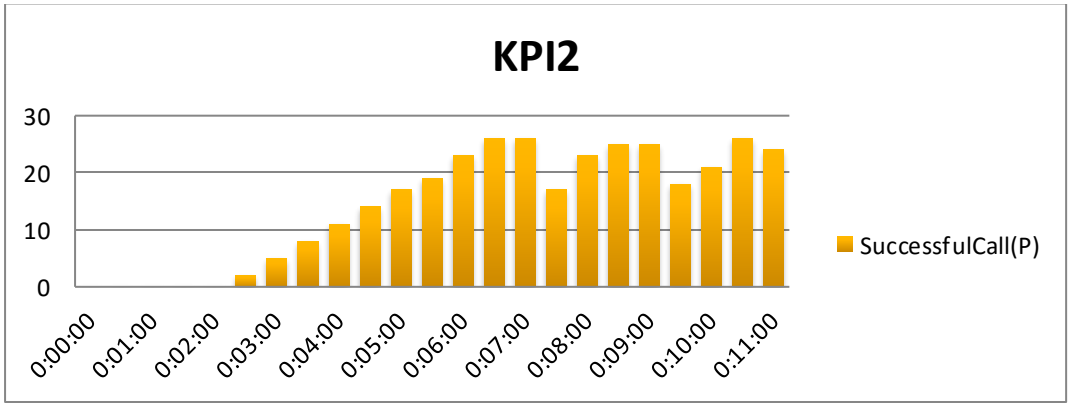


Figure 39 – KPI 2 w/out elastic scaling testing, with one only MGW instance

Figure 39 shows the evolution trend of KPI 2 that starts degrading at around minute 7 because, even for this relatively low numbers, there are failed calls that we quantified as KPI 3 (see Figure 40). KPI 3 has demonstrated to be a relevant indicator to consider because it allowed us, according to the MCN performance methodology, to isolate the component emitting these error messages due to overloading; it is the MGW that represents the bottleneck in this case.

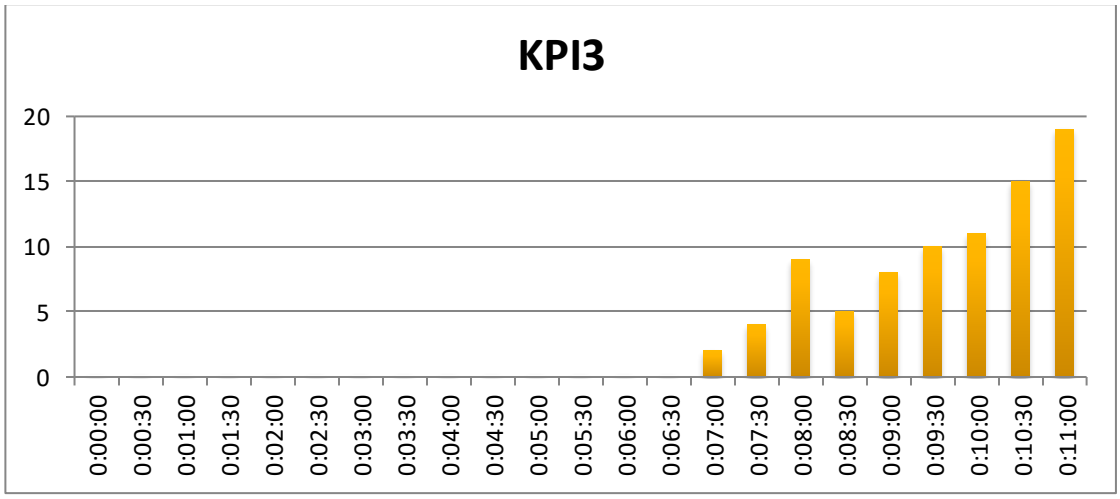


Figure 40 – KPI 3 w/out elastic scaling, with one only MGW instance

In fact, as the above chart clearly depicts, failed calls per period seem to represent an accumulation effect that starts around minute 7 and that increases with the increase of the target rate until the end of the experiment. Once noticed that, we focused on the type of the error to better focus the reason why these calls were failing. The MMAS detects the error when it receives an unexpected BYE message from MGW and forwards it back to UAC. After deep analysis and investigation, we have understood that what happens is that, with the increasing of the calls and the call rate, more and more calls accumulate on the MGW that cannot handle them quickly enough (also due to the challenging call duration). Hence, non-served calls become pending on the MWG and go to a condition of INVITE timeout (although it has been fixed at the very high value of 360 seconds). When INVITE timeout expires, the MGW (Asterisk in our evaluation tests) sends back an automatic and self-generated BYE message. Hence, that means that the MGW is unable to keep up with this reasonably low incoming target call rate.

Hence, following the proposed MCN performance methodology, once pointed out that the MGW is the bottleneck of our deployment in this scenario, we leveraged elastic scalability to alleviate the MGW bottleneck issue by dynamically adding a new MGW instance and then using it to share the load among the two. By the NFVO (OpenBaton in our evaluation tests), the MMAS receives a MGW2 ready message (step 22 in Fig. 9 in Deliverable D5.5) and starts sharing the load among the two available MGWs, namely, MGW1 and MGW2.

With elastic scaling enabled, we tested again the entire individual service with the same testbed and workload used before, and we have compared the results obtained with the previous ones. With elastic scaling activated, the KPI 2 performance indicator improves much (see Figure 41), and that is confirmed also by the better behaviour of KPI 3 (see Figure 42): with two MGW instances, there are only a very limited number of failures during the last period. This improvement is also evident by the trend of the Current Calls at UAC shown in Figure 43. The yellow line represents the experiment without scaling; the number of the current calls in this case is normally higher because the single MGW is not able to handle the high number of the calls that remain pending. In the scaling case, with more instances of MGW, more calls can be handled and correctly closed in the same period; hence, the number of open current calls is significantly lower.

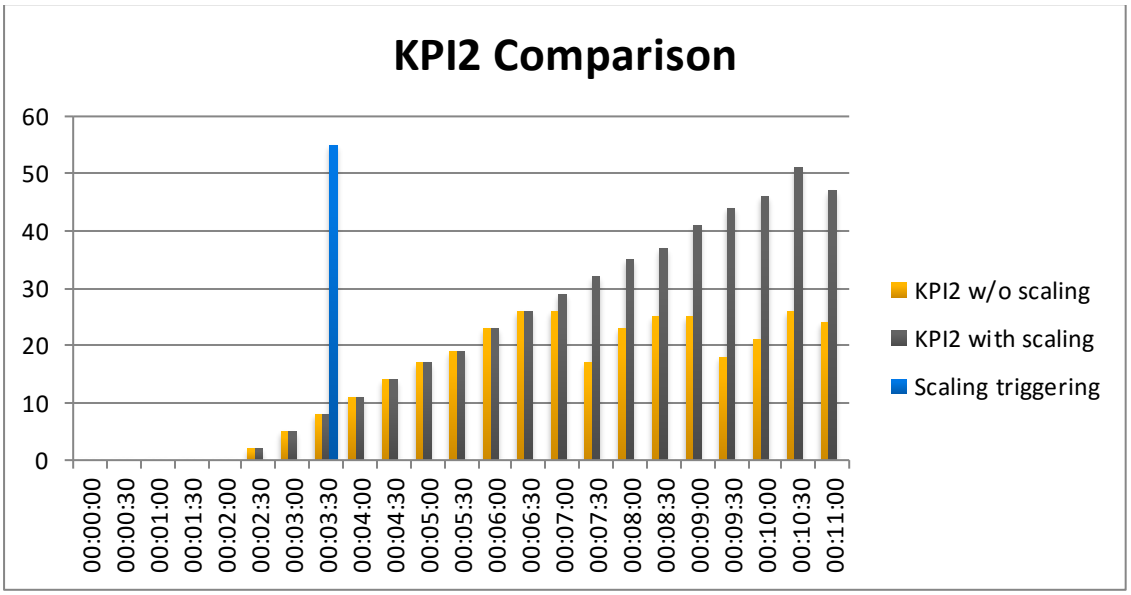


Figure 41 – Behaviour of KPI2 for the elastic scaling testing on Bart with two MGW instances

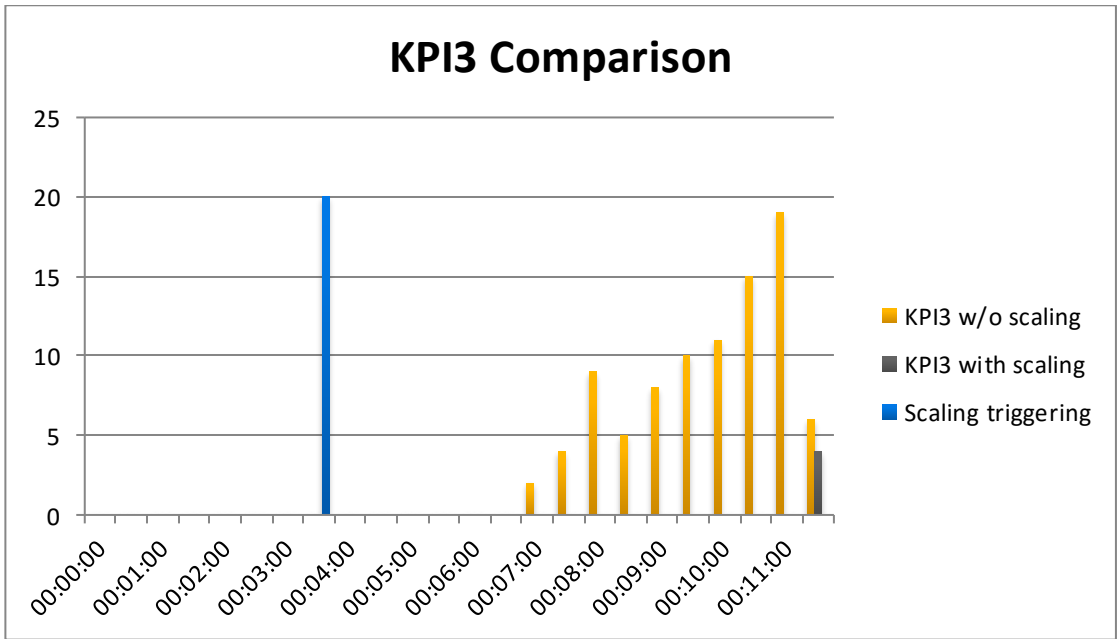


Figure 42 – KPI 3 trend for the elastic scaling with two MGW instances.

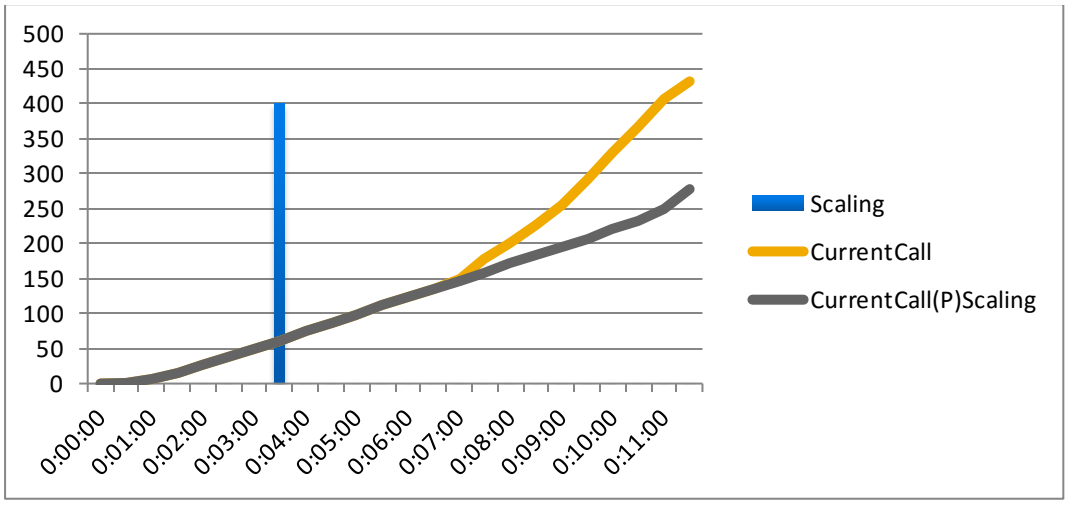


Figure 43 – Number of current calls at the UAC with two MGW instances.

Hence, the reported results confirm that the introduction of our dynamic scale-out solution makes the overall individual service able to handle a higher number of calls with the number of failed calls drastically reduced. Let us stress that, although without any ambition of been a full-fledged performance result, that also confirms the effectiveness and the utility of the proposed methodology to determine and isolate the most critical bottlenecks, so to replicate more critical and overloaded components into the system.

Let us conclude this section with a detailed reporting and analysis of the KPIs related to system resources. First of all, we report the performance results of the MMAS and MGW in order to evaluate the trend of resource usage with or without our scaling solution. To this purpose, we monitored the usage (in percentage) of CPU and memory, namely KPI 26 and KPI 27, for MMAS VM, MGW VM1, and VM2 (before and after the scaling). As for the KPIs considered above, we have monitored only the INVITE call phase.

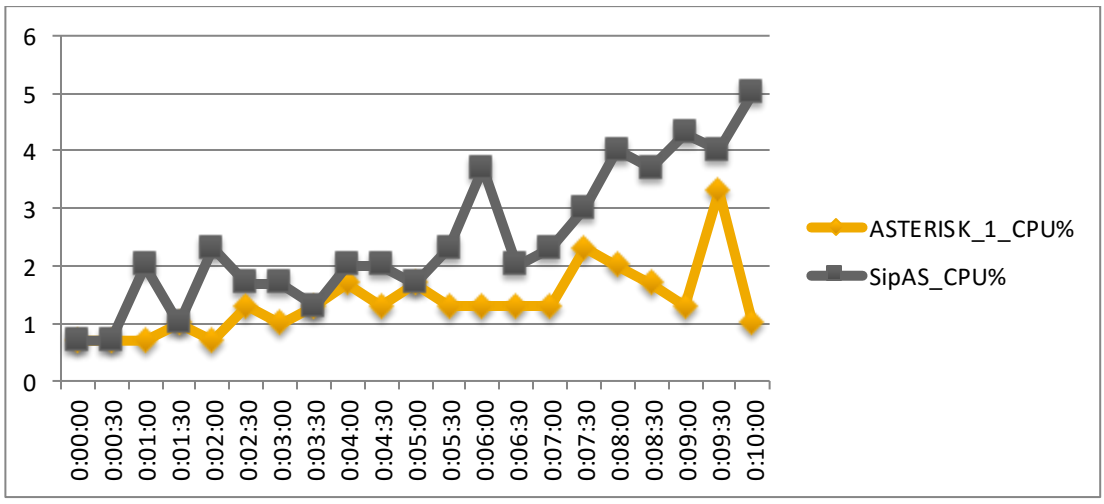


Figure 44 – CPU% w/out elastic scaling, with one only MGW instance

Figure 44 shows the trend of CPU (KPI 26) over MGW (yellow line) and MMAS (grey line). We can easily observe that the usage grows with the increase of the call rate and the number of the incoming

calls; we can also notice that the CPU usage of the MMAS is slightly higher than the MGW one. In any case, the values are really low, always below 5%.

Regarding memory usage (KPI 27), Figure 45 shows its evolution trend. As for the CPU usage, the memory usage percentage of MMAS is higher than the AMGW one. We can see that the usage percentage of the MGW remains more or less stable with the increase of the calls and call rate, while the MMAS one grows of almost 8%. From these two figures, we can also observe and point out that the MMAS, which is still a prototype and has not been optimized as the widely diffused Asterisk product, is the component mostly under stress.

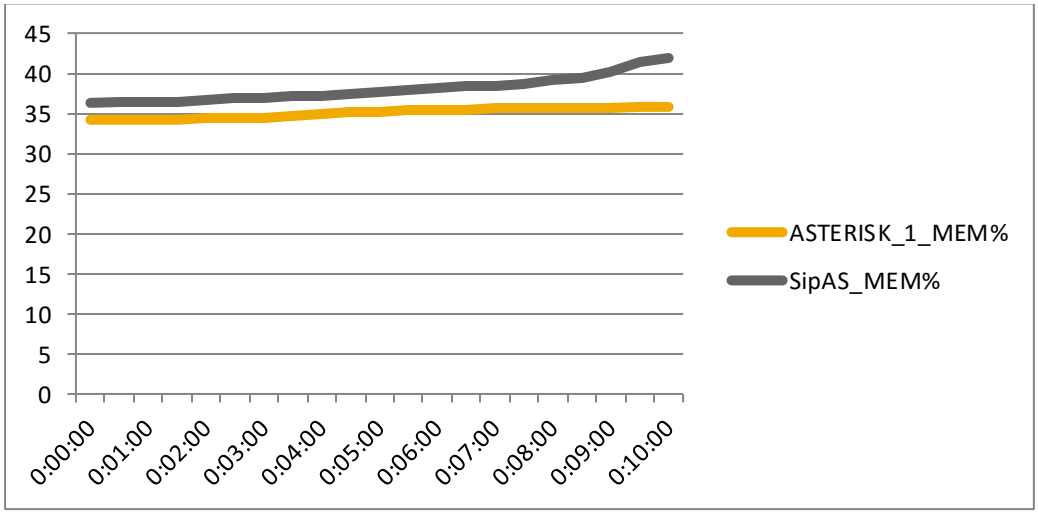


Figure 45 – MEM% w/out elastic scaling, with one only MGW instance.

By introducing elastic scaling, it is reasonable to expect that the memory and CPU used by MMAS are higher, while the percentages of the MGW1 resource usage should be lower. Figure 46 reports KPI 26 evaluation for MMAS, MGW1 and MGW2 with elastic scaling activated.

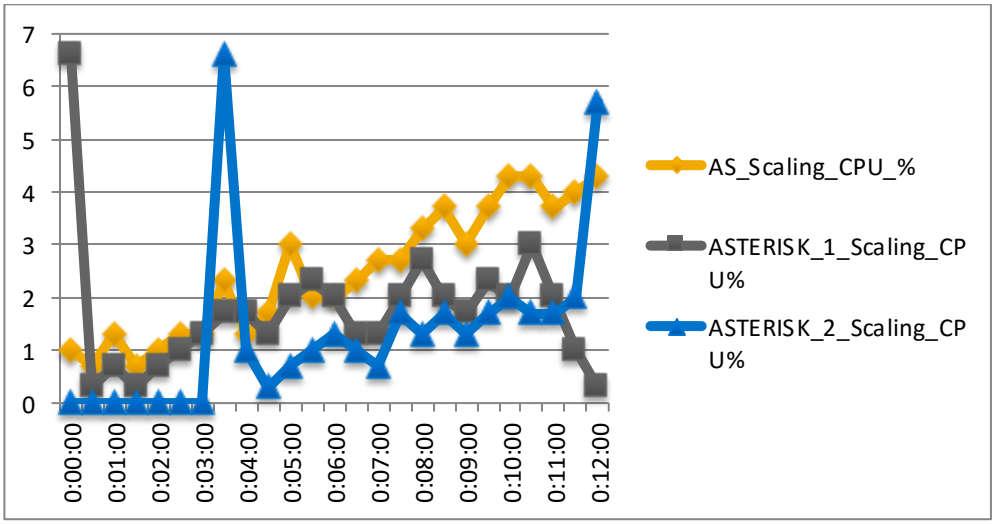


Figure 46 – KPI 26 with elastic scaling with two MGW instances.

The first two spikes of the MGW (both instances) are due to initial bootstrap of its components (loading libraries, allocating memory, etc.). Comparing the performance measured with (see Figure 44) and without (see Figure 46) scaling, we can observe that the usage of CPU by the first instance of MGW significantly decreases and the trends of the two MGWs are similar. The MMAS CPU usage, instead, is

similar with and without scaling, thus confirming the good behaviour of the MMAS prototype in terms of introduced overhead.

Another very relevant performance indicator is about the memory usage, reported in Figure 47 . This chart points out how the trends of MMAS and MGW1 memory usage in the scaling case are totally similar to the one without scaling until the second instance of MGW comes up. From that point on, the memory usage of MMAS starts to increase up to 5% higher than its own upper bound in the without scaling case, while the memory usage of MGW1 remains more or less the same in both experiments.

The increment of the memory usage on the MMAS is largely reasonable and acceptable because, with two instances of MGW, the MMAS has to store data of two different MGWs instead of one, and it has to handle data from both MGWs concurrently.

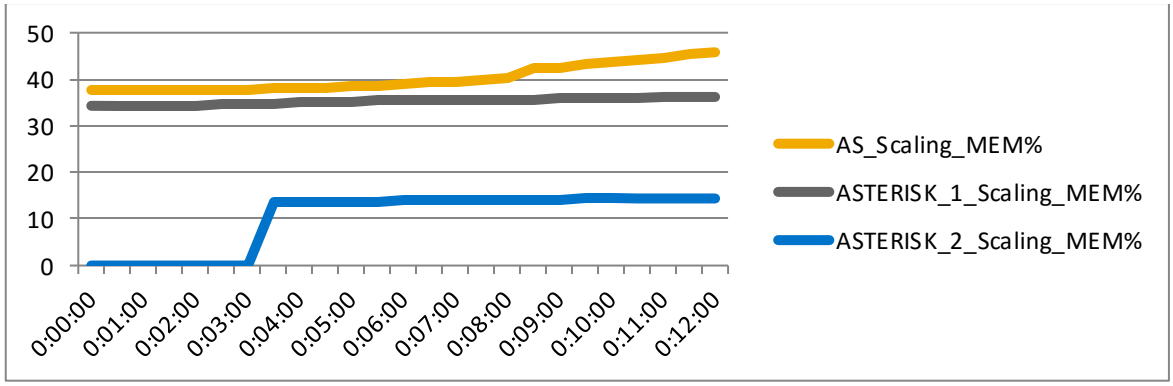


Figure 47 – KPI 27 with elastic scaling with two MGW instances

The trends of CPU and memory of MGW1 remain more or less the same in both the experiments because there is not an interruption of the calls' flows toward it, but only a reduction of the traffic according to the simple load sharing algorithm realized, which shares almost equally the load across the two available MGWs.

4.3.2.2.4 Scenario 4: Spike workload

In this scenario it has been executed a spike workload. Figure 48 shows the results obtained from the execution of the spike workload against one single MGW instance. It can be seen that both KPIs (successful call rate and failed call rate) are almost identical.

Figure 48 – KPI 2 and KPI 3 results for the spike scenario

Figure 49 shows the results from the resource KPI perspective. In particular, as for the previous results, also here there is not too much difference between the results obtained on a virtualized environment if compared with the ones where the MGW was executing directly on hardware.

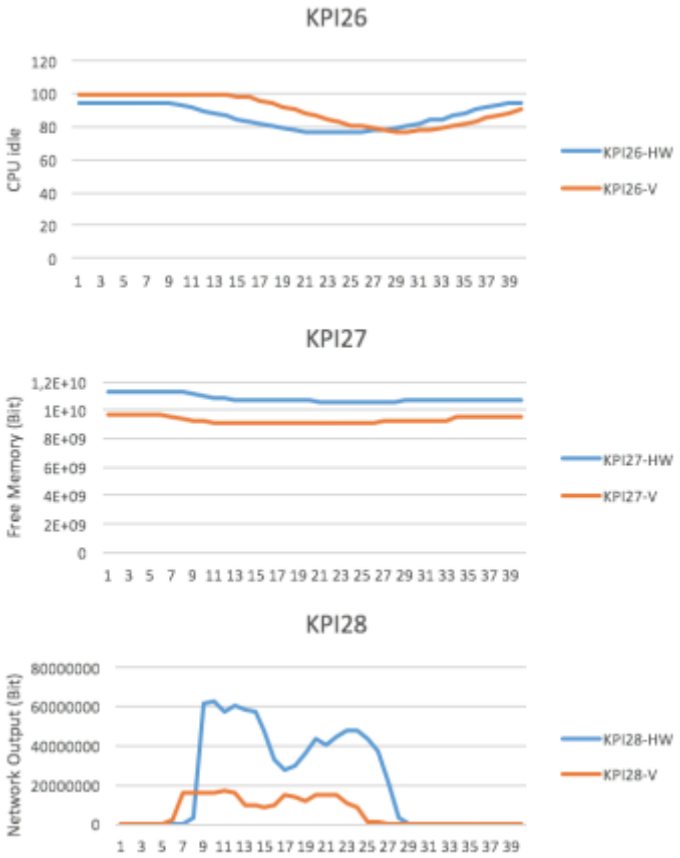


Figure 49 – KPI 26, KPI 27, and KPI 28 results for the Spike workload scenario

4.3.2.2.5 Scenario 5: Stairs workload

The scenarios evaluated in this case are providing the typical stairs workload with different inclinations. Those workloads were executed against a single instance of a MGW firstly installed directly on the Lenovo box provided by the Fokus lab, and then on top of a KVM VM running on the same hardware. Both executions where done in series, so that they cannot interfere with each other.

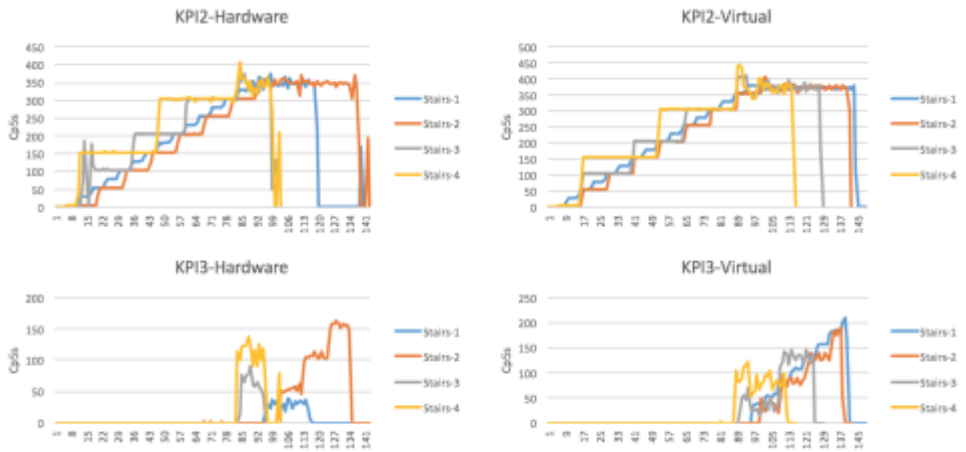


Figure 50 – KPI 2 and KPI 3 results of a MGW running on a virtualized vs HW-based environment

Figure 51, Figure 52, and Figure 53 are showing the results of the resource KPIs measured while executing these stairs scenarios against an instance of the MGW running; the instance was firstly deployed directly on hardware and then on a VM.

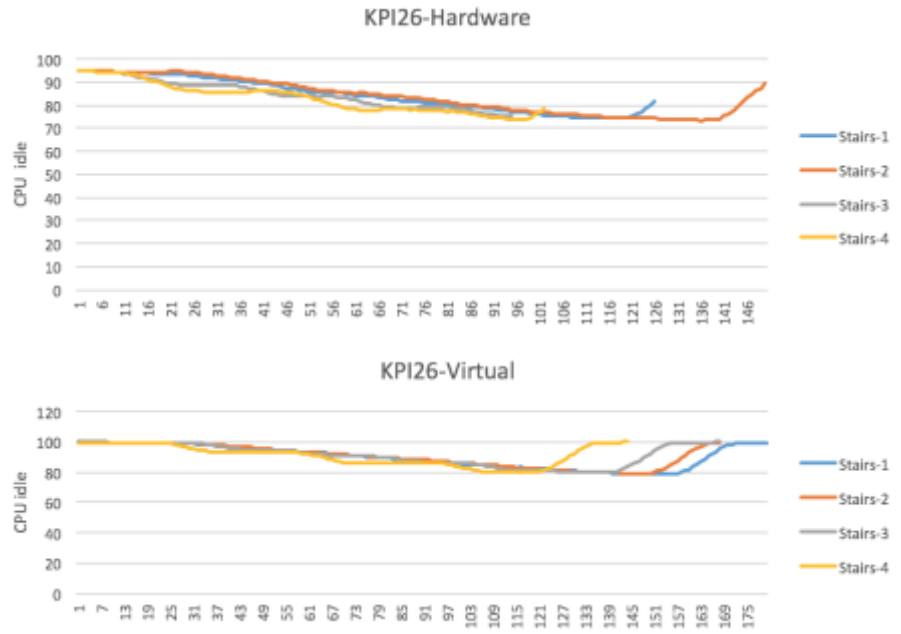


Figure 51 – KPI 26 results of a MGW running on a virtualized vs HW-based environment

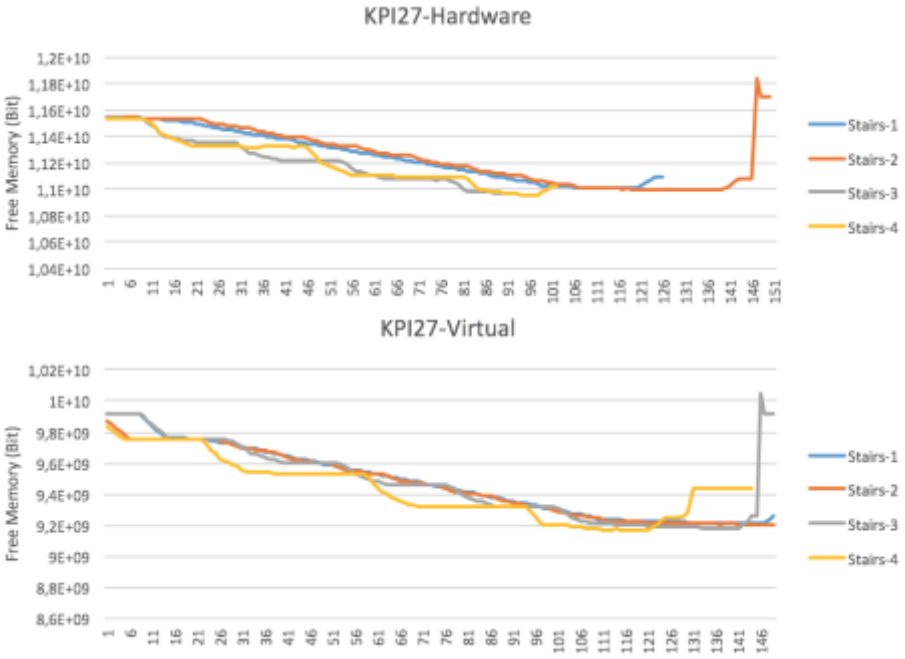


Figure 52 – KPI 27 results of a MGW running on a virtualized vs HW-based environment

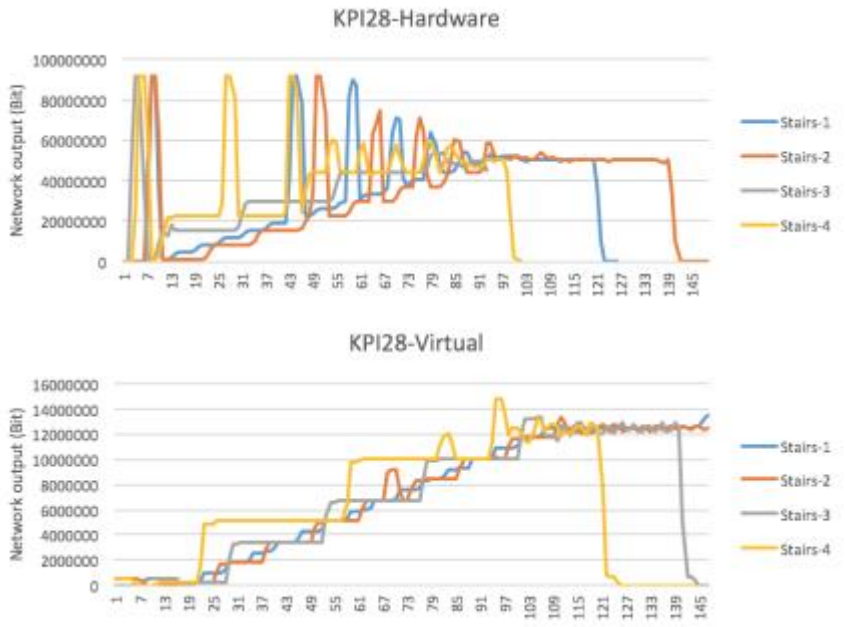


Figure 53 – KPI 28 results of a MGW running on a virtualized vs HW-based environment

4.3.2.2.6 Scenario 6: Constant workload

This scenario is the last one used for evaluating the media plane components and is also the simplest one in the set of considered scenarios, given that it provides a constant number of call per second as workload distribution. Figure 54 shows the results for KPI 2 and KPI 3.

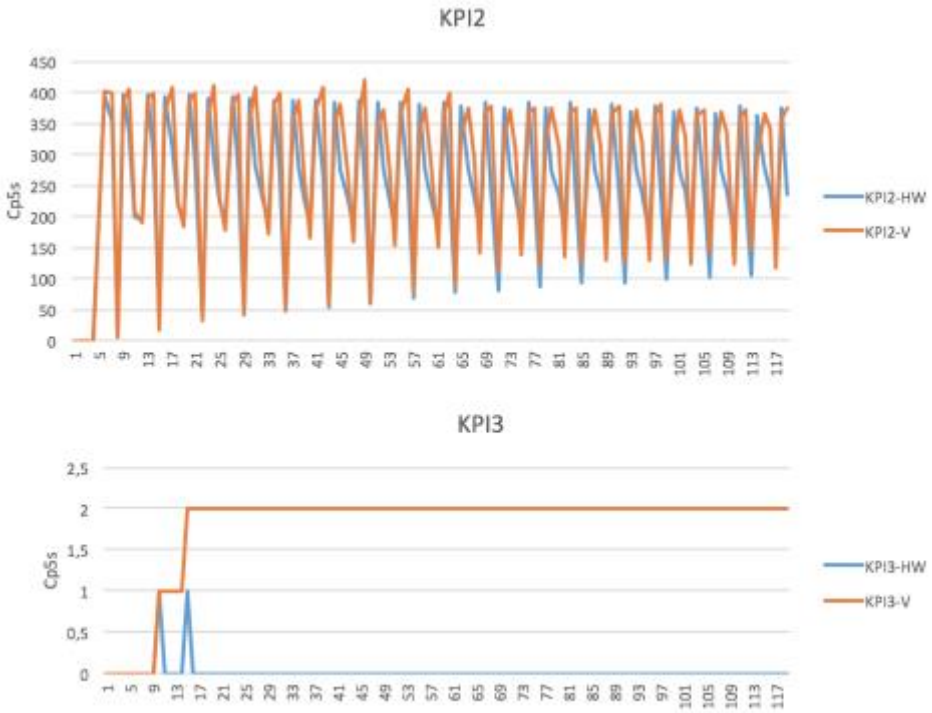


Figure 54 – KPI 2 and KPI 3 results of a MGW running on a virtualized vs HW-based environment

As it can be noticed the number of successful calls has a floating behaviour, which has demonstrated to depend on buffering at the ingress queues of the MGW or at the external queues of the benchmarking tool. The most relevant KPI, number of failed calls (KPI3), is anyway very low in both cases. Analogous considerations may be made for the results in Figure 55 regarding the resource KPIs.

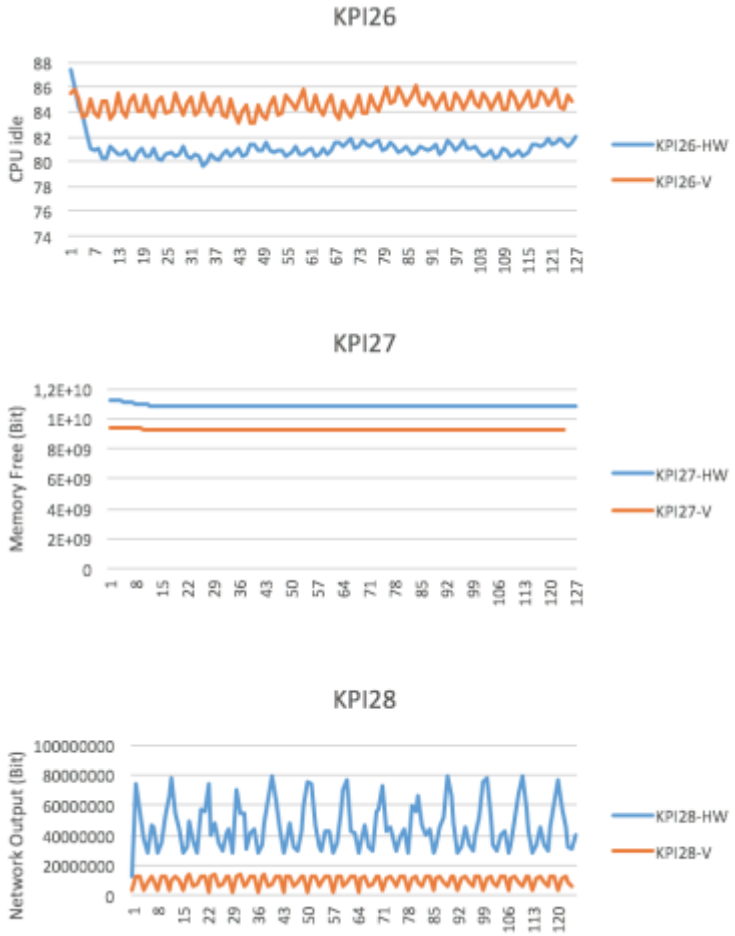


Figure 55 – KPI 26, KPI 27, KPI 28 results of MGW running on a virtualized vs HW-based environment

4.3.2.2.7 Scenario 7: Disposal

This scenario has been employed for experimentally determining the lifecycle KPIs of the IMSaaS Orchestrator. In particular, we have measured the time for disposing the IMS SI from the moment the EEU issues a request to the IMS SM. It is important to clarify that the measurements taken into this scenario are differing from the ones presented earlier in previous deliverables (D5.4 and D6.4) as they make use of the new Orchestration system based on Open Baton.

Table 39 – KPIs collected in scenario 7

KPIs	Adaptation / Detail	Values
KPI 10 - Disposal duration	Time to dispose a service instance	0.5 seconds
KPI 31 - Number of VMs used	Number of VMs used (IMS composed services)	7VMs

KPI 32 - Resources consumed for orchestration	HW resources consumed for SO and SM	MeM: 4GB 4vCPUs

4.3.3 Radio Access Network – EURE RANaaS

For the following set of results, the hardware set up used is the same as the one described in Deliverable D6.4. More specifically, it consists of a Gigabyte Cube with Intel Core i7 4770R running at 3.2 GHz, 4 cores, hyper-threading, 16 GB RAM, and 240 GB HDD. The cube runs a complete single-host all-in-one OpenStack installation. The OpenStack version is Juno running on Ubuntu 14.04. The services include OpenStack Nova with Linux Containers (LXC), Glance, Neutron and Heat. The OpenStack server is directly connected to a NI/ETTUS USRB B210 radio frontend (2x2 MIMO). As part of the setup, there is also one commercial LTE-enabled USB dongle, namely Huawei E398. Finally, the eNB is configured for single antenna operation at either 5MHz or 10 MHz (depending on the test) channel bandwidth in band 7 (2.68GHz). In the following sub-sections, we detail the description of the tools and scenarios that we used to experimentally acquire the metrics for the KPIs of interest defined in Deliverable D3.5 and thus evaluate the performance of Eure RANaaS in standalone mode.

4.3.3.1 System KPIs

In order to measure the System KPIs, a set of different tools were used depending on the evaluation metrics and performance indicators that we wanted to collect from in-the-field deployment.

KPI 1 and KP 2 - Attachment and session establishment success rates

In order to measure the system KPIs, we used one UE at different locations with respect to the BBU/RRH as depicted in the figure below. Two procedures are considered, namely **Initial Attach**, and **EPS dedicated bearer setup** (E_RAB_SETUP). Note that in the latter case a UE can have more than one communication channel with the base station, each one responsible for a different type of traffic. In the conducted experiments two additional channels are established.

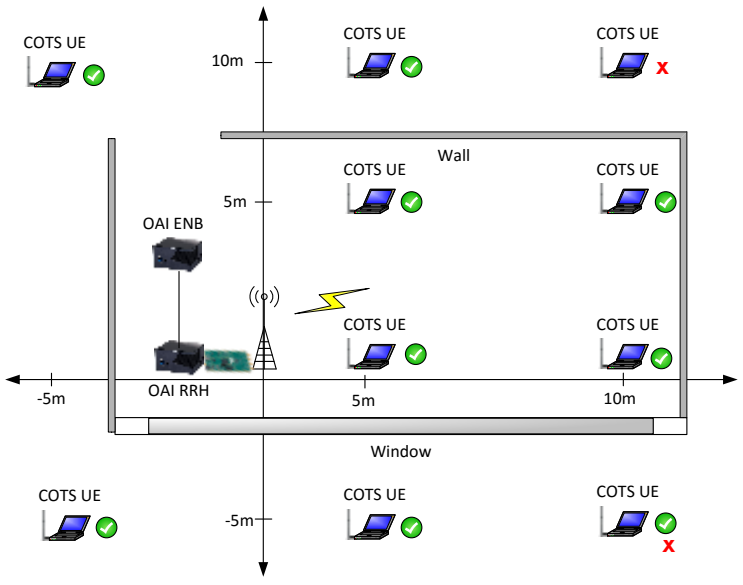


Figure 56 – Measurement setup for the system KPIs

The initial attachment procedure would fail only in cases where the user is experiencing a deep fading, e.g., due to walls, and thus the user *attach request* message is not detected or failed (overloaded scenarios and channel access contention are not considered in these experiments). As for the dedicated bearer session, two cases are observed: either failure due to failure of the attach procedure, or failure after a successful initial attach procedure. The latter occurs primarily due to uplink errors.

The measurements have shown a success rate of 90% for the initial attach procedure, and 80% for the EPS radio access bearer establishment.

KPI 4 and KPI 5 - Attachment and session establishment delay

For measuring the Attachment and Session delay KPIs, we used Wireshark so that we can measure delays based on the timestamps in the collected traffic traces. Wireshark traces were taken from both BBU and EPC. The following figure describes the signalling that takes place for the Initial attach and the EPS dedicated bearer setup procedures.

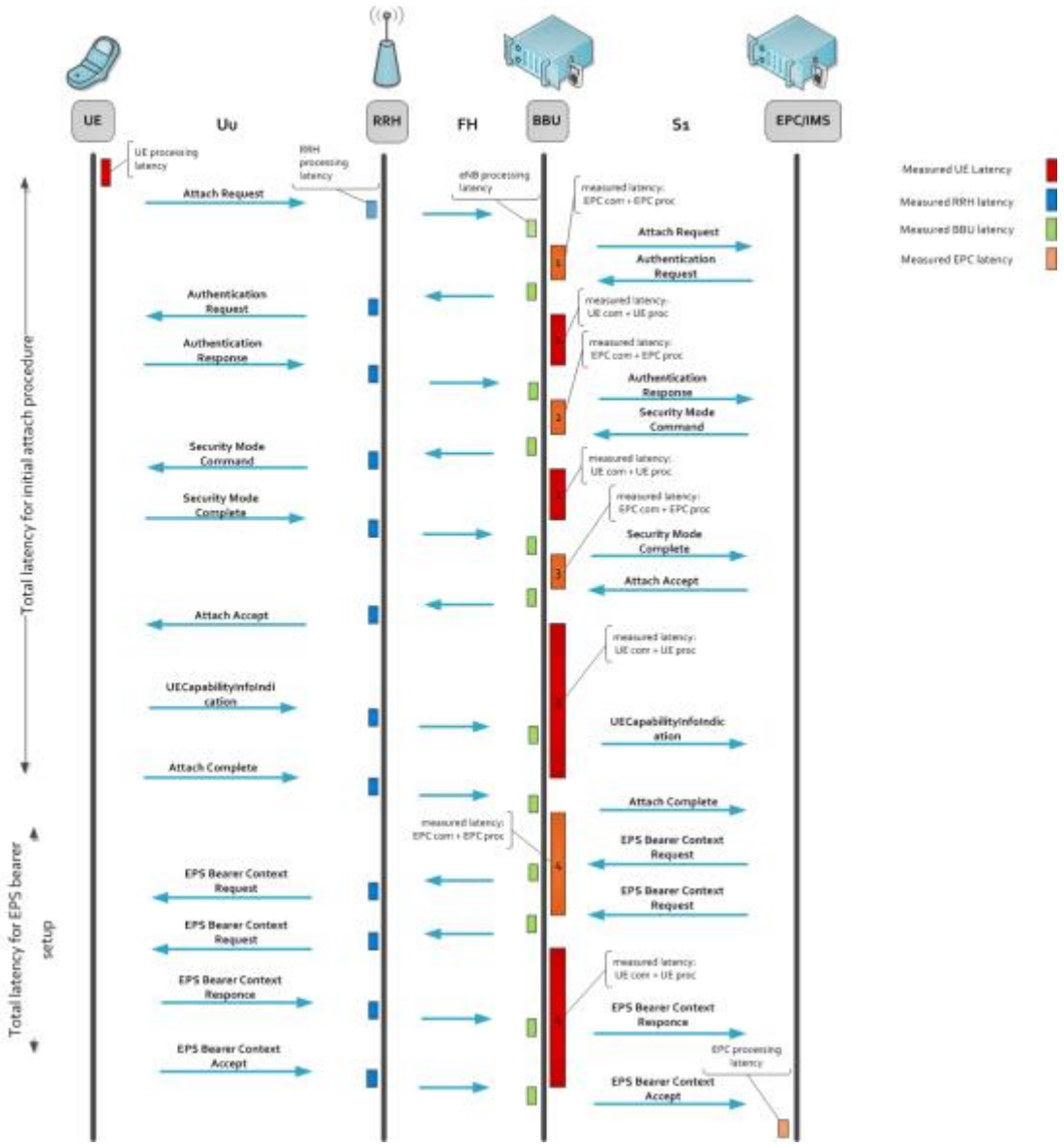


Figure 57 – Initial attach and EPS dedicated bearer procedure latency breakdown

Having as a reference the figure above and using the traces acquired by Wireshark, the following tables with performance indicators are derived:

Table 40 – Initial attach procedure latency (BBU traces)

Latency (ms)	1	2	3	Total
UE communication and processing	315 (0.3147)	18 (0.0178)	250 (0.2502)	583
EPC communication and processing	40 (0.0402)	2 (0.0021)	8 (0.0083)	50
Total	355	20	258	633

Table 41 – Initial attach procedure latency (EPC/IMS traces)

Latency (ms)	1	2	3	Total
UE communication and processing	315 (0.3149)	18 (0.0181)	250 (0.2502)	583
EPC communication and processing	39 (0.0399)	2 (0.0019)	8 (0.0081)	49
Total	354	20	258	632

Table 42 – EPS dedicated bearer procedure latency

BBU	
Latency (ms)	**4**
UE communication and processing	219 (0.2190)
EPC communication and processing	9 (0.0090)
Total	228

EPC/IMS	
Latency (ms)	**4**
UE communication and processing	219 (0.2193)
EPC communication and processing	9 (0.0087)
Total	228

For both procedures, we can see that the results from both probing points (BBU and EPC) are quite similar and their differences, if any, are very limited and in the order of 1ms. For the initial attach procedure, the total control-plane latency measured is 633 ms and, for the EPS dedicated bearer set up, is 228 ms. In both cases the radio access network is the one that has been experimentally found to introduce the largest part of the measured latency.

KPI 6 - Data plane QoS

For measuring the Data plane QoS we used iperf, a client-server based tool for measuring bandwidth between two endpoints. The data traffic used is of type UDP and the data rate employed is 15Mbit/sec and 30Mbit/sec depending on the bandwidth configuration at eNB. We used two PCs: in one of them iperf ran in client mode and was generating traffic for the UE (LTE dongle), which was connected to another PC in which iperf was running in server mode and was just discarding the traffic.

For the Data plane QoS it is interesting not only to benchmark the current implementation of the RAN service, but also to compare the altered RAN architecture with the performance results achieved by the original one (up to M36, before the project extension period). Additionally, it is important to investigate

the behaviour of the extended RAN architecture under different circumstances; the fact that the I/Q samples are packetized and transported via Ethernet makes hard and sometimes impossible to meet the frame/subframe and HARQ timing constraints imposed by LTE. As result, in the following figures we choose to have the throughput results for the following cases:

- **eNB_1, 5 MHz/10 MHz** - refers to the standard RANaaS version without RRH.
- **eNB_2, 5MHz/10MHz** - refers to RANaaS with local RRH GW (RRH GW is running in the same physical hardware as BBU)
- **eNB_3, 5 MHz/10MHz** - refer RANaaS with remote RRH GW (RRH GW and BBU are running on two distinct physical hardware and the I/Q samples are transported over Ethernet)

The table and figure below present the average jitter, drop rate, and goodput of RANaaS in the three considered scenarios for 5MHz and 10MHz channel bandwidth. It can be seen that the same performance is achievable with and without RRH, confirming the feasibility of packet-based fronthaul network based on Ethernet. In other words, when having a 5 or 10 MHz radio bandwidth the I/Q samples along with the data introduced by the fronthaul headers are able to fit in a 1Gbit link and arrive on-time to the BBU (LTE stack timing is respected).

Table 43 – Jitter and drop rate of RANaaS in different scenarios

Metric / Scenario	eNB_1	eNB_2	eNB_3
average jitter values (ms) 5MHz	1.998ms	1.939ms	1.943ms
average jitter values (ms) 10MHz	1.611ms	1.393ms	1.403ms
average drop rate values (%) 5MHz	7.5170e-006%	2.3232e-005%	3.8787e-005%
average drop rate values (%) 10MHz	9.5807e-004%	2.3646e-005%	0.0011%

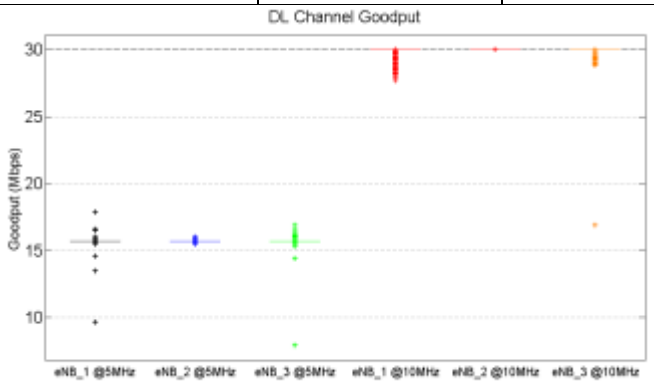


Figure 58 – Measured goodput at the UE for different scenarios

KPI 7 - Data plane delay

For the data plane delay KPI, we measure the application Round Trip Time (RTT) over the default radio bearer. In particular, RTT is evaluated through different traffic patterns generated by the *ping* tool with 64, 768, 2048, 4096, 8192 Packet Sizes (PS) in byte, and 1, 0.8, 0.4, and 0.2 Inter-Departure Time (IDT) in seconds. While our focus is on the evaluation of the RAN service, a third party EPC at Eurecom is used to measure the RTT at the application level.

The figure below presents the measured RTT as a function of PS and IDT for the three considered cases, namely standalone eNB, eNB with local RRH, and eNB with remote RRH. As expected, higher RTT

values are observed when PS and IDT increase. It can be also noted that the RTT trend is similar in the three cases, thus confirming the feasibility of RRH. However, a deeper investigation is required to better characterize any potential performance degradation with RRH.

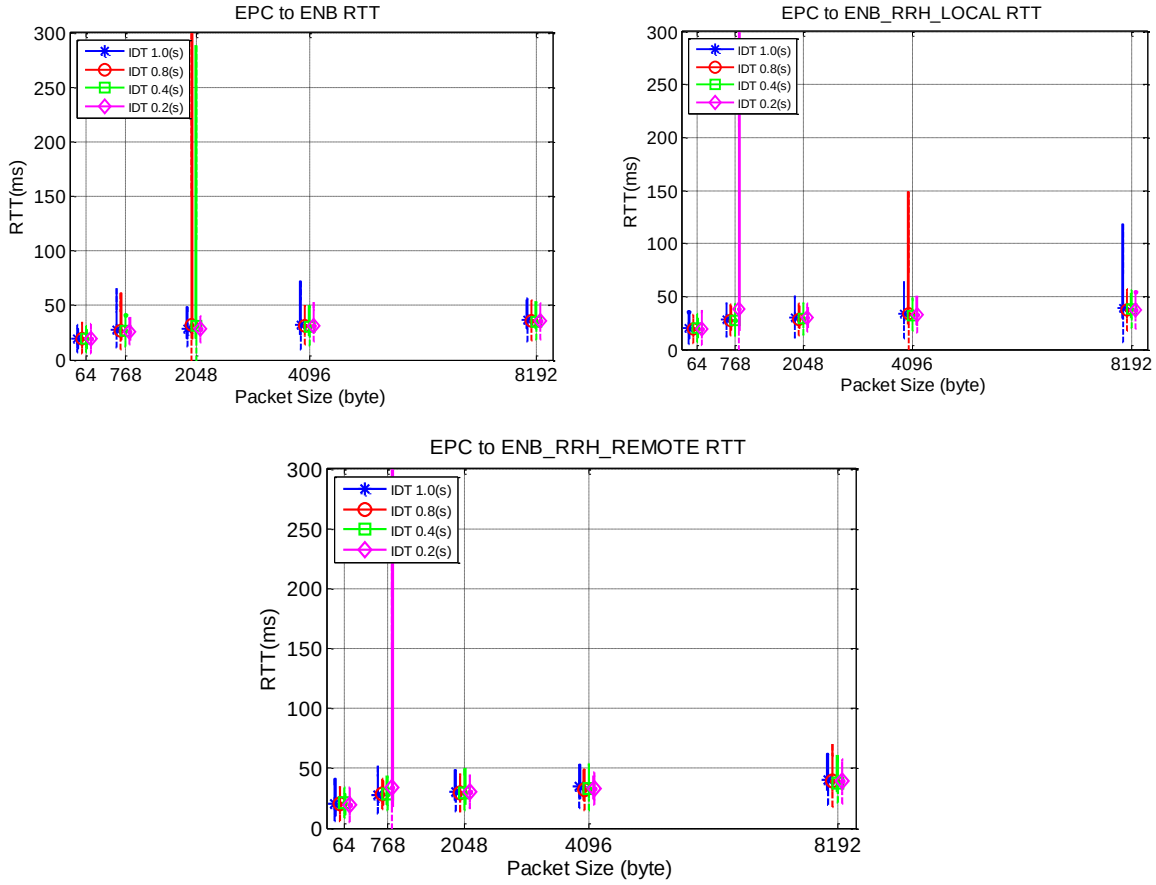


Figure 59 – Data-plane round trip time for the three considered scenarios

Summary of System KPIs

The table below summarizes the experimental results about System KPIs for RANaaS service located at Eurecom.

Table 44 – Eurecom RANaaS System KPIs

KPIs	Units	KPI measurement
Registration success rate	Rate(%)	90%
Session establishment success rate	Rate(%)	80%
Session drop rate	N/A	N/A
Attachment delay	Time (ms)	633 ms
Session establishment delay	Time(ms)	228 ms
Average Data plane QoS	Time (ms)	average jitter:

	Rate(%)	• 5MHZ: 1.939ms • 10MHz: 1.393ms average packet loss: • 5MHZ: 2.3232e-005% • 10MHz: 2.3646e-005% average throughput: • 5MHZ: 16MBps • 10MHz: 30MBps
Data plane delay	Time(ms)	Round trip delay: • 64 bytes/packet: 35 ms • 768 bytes/packet: 45 ms • 2048 bytes/packet: 44 ms • 4096 bytes/packet: 48ms • 8092 bytes/packet: 59 ms

4.3.3.2 Lifecycle KPIs

To measure the lifecycle KPIs, we have made use of widespread time utilities, so that we had the possibility to measure the application and system CPU time spent to operate the actions of relevance for the KPIs of interests. To the sake of maximum clarity, we provide a brief description:

- Application time measures the amount of CPU time spent in user-mode code (outside the kernel) to execute the process.
- System time measures the amount of CPU time spent in the kernel within the process for system calls.

Table 45 – Eurecom RANaaS Lifecycle KPIs

KPIs	Units	KPI measurement
KPI 8 - Installation duration	Time (s)	Application: 180.0 – 481.0 seconds System: 61 seconds Total: 241 – 542 seconds
KPI 9 - Deployment and configuration duration	Time(s)	Application: 1.42 second System: 2.2 second Total : 4.02 second
KPI 10 - Disposal duration	Time(s)	Application: 0.4 second System: 0.6 second Total : 1 second
KPI 11 - Service upgrade duration	Time(s)	Application: 107 – 193 seconds System: 15 – 44 seconds Total: 122 – 237 seconds

Note that if the application standard outputs and errors are not redirected appropriately, the induced delay is orders of magnitude higher (mainly due to the I/O delay). For example, in the case of installation,

the delay could be as large as 10-12 minutes, and for the deployment and configuration, this could be 40-60 seconds.

The reported measurements reveal that a total of 546 to 245 seconds (9-4 minutes) are required for service installation, configuration, and deployment, and 237 to 122 seconds (4–2 minutes) for the service upgrade that includes both service update, reconfiguration, and redeployment.

4.3.3.3 Resources KPIs

The resource KPIs has shown to strongly depend on actual load. Indicatively, we present the CPU usage for uplink for an eNB configured with 10 MHz bandwidth (50 PRBs), SISO mode, and full data workload generated by the iperf utility. Downlink and uplink MCS are fixed to 26 and 16 (the maximum value) in order to produce high processing load. Only 4 uplink Sub-Frames (SFs) are granted by the eNB, namely SF #0, 1, 2, and 3, allowing UL transmission to occur in SF # 4, 5, 6, and 7.

We note that in most cases RX processing requires 1 ms runtime to process the received signal and the resulted protocol data unit across the layers. It may happen that this processing goes as large as 1.5 ms. This corresponds to a maximum of 1.5 core at the 3GHz CPU frequency. Similar results show that, in the TX, processing requires 0.7 core for the same frequency. Thus a total of 2 CPU cores are required.

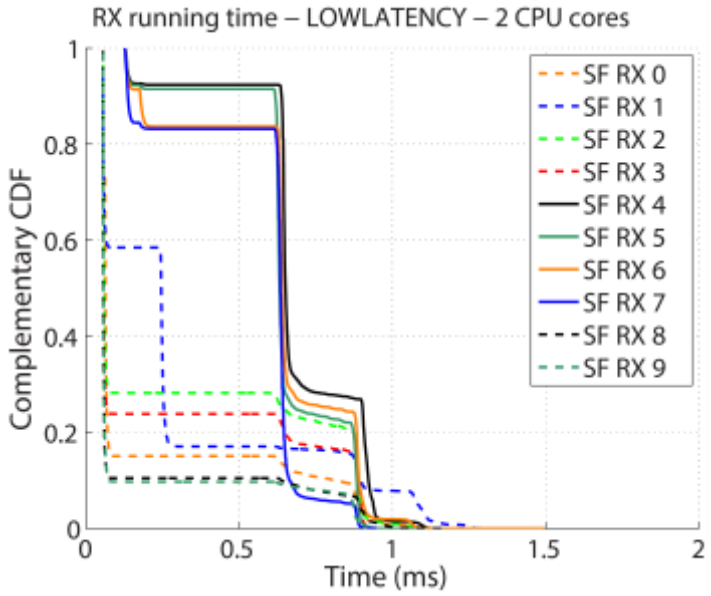


Figure 60 – RX processing time required for RANaaS

Below a screenshot is included in which the momentary usage of CPU is measured by the htop utility. The CPU usage is 26.6% but this value is not representative since the load of the CPU is highly traffic dependent. As a general consideration, from our experimentation and evaluation work, we can consider safe to say that one core is required.

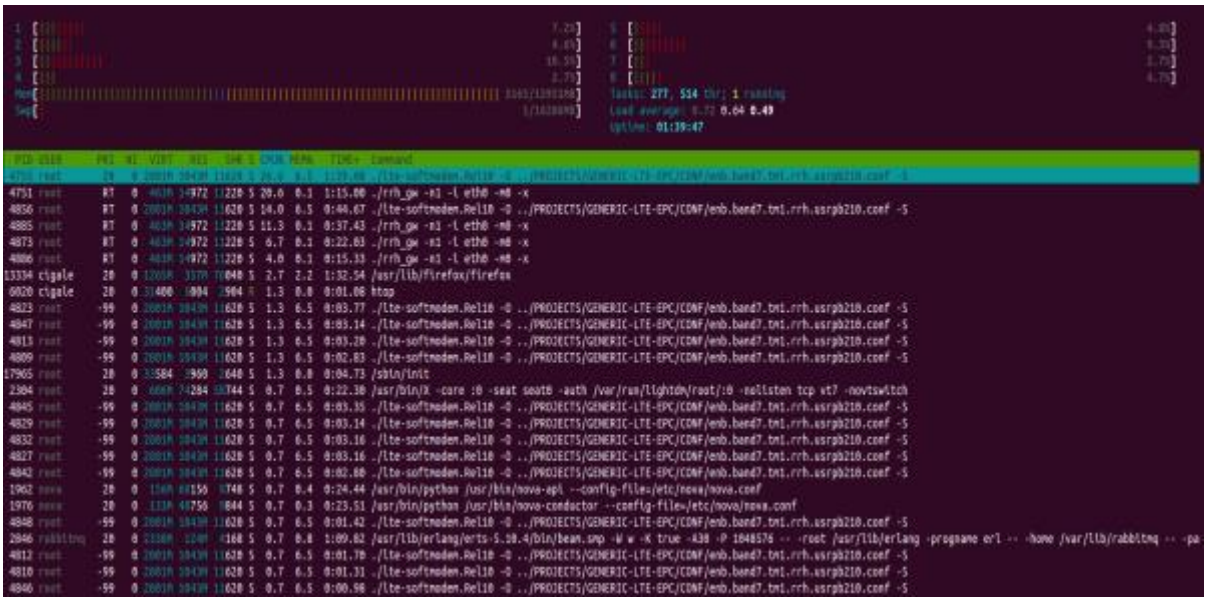


Figure 61 – Momentary CPU usage for RAN service testbed

The table below summarizes the evaluated resource KPIs for the RANaaS located at Eurecom.

Table 46 – Resource KPIs for RANaaS at Eurecom

KPIs	Units	KPI measurement
KPI 26 - CPU usage	Number	10MHz: • Reserved: 2 cores @ 3GHz • Momentary: 1 core
KPI 27 - Memory usage	Kbytes	eNB: • Vss: 2868392K • Rss: 1067068K • Pss: 1063554K • Uss : 1062712K RRH: • Vss: 474648K • Rss: 15952K • Pss: 13391K • Uss: 12728K **Note:** this is the minimum momentary memory usage, and additional memory will be used depending on the number of attached UEs.
KPI 28 - Network usage	Mbps	RANaaS: 1 interface to EPC • 5MHz: 16Mbps • 10MHz: 30Mbps RANaaS and RRH: 1 guest-only interface between BBU and RRH • 5MHZ: 16Mbps + 326Mb/s (BBU-RRH) • 10MHz: 30Mbps + 653Mbs (BBU-RRH)

		Note that the BBU-RRH guest-only interface has a continuous resource usage.
KPI 29 - External network usage	Mbps	RANaaS: 1 interface to EPC • 5MHz: 16Mbps • 10MHz: 30Mbps
KPI 30 - Storage usage	Time(s)	RANaaS image: 6 Giga Bytes RRH image: 6 Giga Bytes Note: the above images could be further shrink.
KPI 31 - Number of VMs used	Quantity	Legacy eNB: • RANaaS: 1 Cloud-RAN: • RANaaS : 1 • RRH: 1 Note: In the considered setup, no split is used.

4.4 Evaluation conclusions

Section 4 has extensively reported and discussed the evaluation results deriving from our deployment and experimentation activities for the end-to-end PoC scenarios (considering lifecycle and resource KPIs) and the individual services (considering all the dimensions of our proposed evaluation methodology).

In the DSS and IMS PoCs, we have noticed a decrease in lifecycle KPI performance indicators in regard to deployment, provisioning, and disposal processes, due to the fact that this project extension evaluation work has been performed in testbeds with lower computational capabilities then the previous end-to-end experimentation. The impact that such end-to-end scenarios have in platforms is mainly in terms of CPU resources, where it was observed a higher CPU usage within the deployment, provisioning, and disposal phases. Such CPU usage was verified in all the orchestration nodes (OpenShift), including as well OpenStack nodes. An additional resource that also increases in terms of usage corresponds to the network inbound and outbound bandwidth. Indeed, it was verified that during the mentioned lifecycle phases the inbound/outbound traffic increases as well.

On the one hand, DSS results obtained from the different tests put in evidence that the finally perceived quality is not heavily affected, even in scenario 3 (spike tests) on which DSS components start to present low performance values. Thanks to the results of the preliminary evaluation tests, the implemented provisioning logic has been enhanced, thus achieving a complete service deployment and provisioning time in less than 3.5 minutes and relevantly improving the performance results previously reported in Deliverable D6.4. In addition, DSS scalability has demonstrated to present acceptable performance despite the extra time required for components reattachment to the LBaaS, completing a scale out operation in around 2 minutes with minimum drop rate and a scale in operation in under 30s. Also, the implementations we completed for improving elasticity and fault management of DSSaaS effectively improved the overall stability of system behaviour, thus enabling effective component failure detection, recovery times have demonstrated to be in around 4 minutes, which is largely acceptable in this field. Moreover, the RAVA-enabled SO has shown to be able to perform live migration in order to achieve

testbed resource optimization usage and without significantly affecting the availability of migrated components.

On the other hand, the performance evaluation methodology proposed in the context of WP3 has driven the evaluation of the IMSaaS SICs. First of all, the media plane components have been evaluated on different deployment models: hardware vs virtual, and single vs scaling. The collected results show that the performance results of the media plane components on a virtualized environment are not considerably different from the ones directly deployed on hardware, with the evident advantages in terms of flexibility and elasticity. However, the reported results demonstrate that there is a minimal difference at least in delay, which has shown to be mainly because in the virtualized environment there are additional entities involved in the networking path. Nevertheless, the additional delay is largely compensated by the possibility of dynamically scaling out the data plane. The reported and early results obtained by the scalability of the MGW component show that dynamic instantiation of entities also improves and has beneficial effects on the system KPIs of such telco services.

Finally, it is important to mention that the benchmarking methodology was also very useful in identifying bottlenecks and malfunctions of the different systems and components under test. In particular, the iterative process allowed us to identify and mitigate bottlenecks and configuration/tuning issues in different steps and relatively easily, each time by putting the attention to the most relevant motivation of inefficiency at this stage and with the desired coarse-grained level of approximation.

Regarding the extended version of the RAN service, the results from the IMS PoC come to verify and complement the results of the evaluation of the RAN service in standalone mode. More specifically, once again we verified the feasibility of transporting I/Q samples from a passive RRH to a BBU over Ethernet, in particular when having a 1Gbit direct connection between RRH GW and BBU for 5 and 10 MHz radio bandwidth. In terms of resources, the extended RAN version has demonstrated to be more demanding; an extra VM along with an Ethernet link between RRH GW and BBU is needed, thus also affecting the life cycle performance of the service. In any case, the additional resources needed are relatively limited and largely acceptable, also because the implemented extensions were carefully designed and implemented so that the least possible processing is required.

From the data plane QoS perspective, though, we observed an unexpected performance degradation in throughput. In order to isolate the problem, we used exactly the same RAN service testbed used for the IMS end-to-end scenario but this time with a third-party EPC without the MTU mismatch, where we got the expected performance, namely the performance that the service yielded when tested in standalone mode. Having verified that the problem was not residing on the RAN part, we performed an additional series of evaluation tests and concluded that a mismatch in the MTU size between the RAN service and the EPC was not correctly handled: this has demonstrated to be the primary motivation of the unexpected result for the throughput in the IMS PoC, thus showing again the usefulness of the proposed evaluation methodology and of its iterative process also in order to identify criticalities and better tuning the sub-systems and service components involved in an end-to-end composed service.

When assessing the service KPIs from an end-to-end perspective of the RAN and EPC working together as considered by the MCN project, a clear requirement to have a split of the delay related KPIs per component became obvious. This was mainly due to the need to determine which of the components is behaving under the expected KPI, in the case of the evaluation being the HSS. Thus, we conclude that in order to be able to use in a beneficial manner the methodology proposed for the evaluation of the end-to-end service, the end-to-end KPIs have to be split to a higher level of granularity in order to determine the less performant components.

5 Summary and Outlook

The objectives, previously demonstrated and evaluated in a functional perspective at M36 for the MCN project, in regard to the expected proof of concept implementations, have been achieved and fully evaluated according to the MCN evaluation methodology described in Deliverable D3.5. The DSS and IMS Proof-of-Concepts have been thoroughly evaluated to assess the feasibility and effectiveness of our on-demand support and the elastic provisioning of mobile services.

The PoC scenarios, despite the reduced number of services in the extension period, have relevantly increased in complexity. The DSS PoC has incorporated fault-tolerance mechanisms, thus enabling Over-The-Top (OTT) services to operate in high load conditions (i.e., high number of players requesting content) or even to benefit from dynamic adjustments enabled by RAVA, which has shown to be able to migrate DSS service components to nodes with low resource consumption with acceptable performance. The IMS PoC increased in the complexity of having two distinct Radio Access Networks operating in different testbed locations and with different versions. To stress interoperability of different versions, the RAN OAI in Fokus was based in the M36 version, while the RAN OAI at EURE employed the enhanced version that supports the splitting of BBU and RRH functionalities.

By integrating RAVA support, the DSS PoC has successfully evaluated the vast majority of the KPIs defined in our performance evaluation methodology. In order to adapt the methodology to the constraints applying to an OTT service, some KPIs were refined. Although DSS evaluation is complete, it was mainly focused in the benefits acquired from the implementation of the orchestration enhancement capabilities (fault management, RAVA live migration, and faster provision time). To better illustrate that, an extended cloud-native KPI family was appended. The reported performance indicators show very promising results in terms of user availability time, automatic fault recovery, and component live migration, by significantly improving the results previously presented in Deliverable D6.4. The live migration feature provided by RAVA has demonstrated to behave consistently and to perform adequate component migration, with no downtime, depending on the conditions of the resource affinities of every physical node considered. Although the results described here are limited to one single testbed, the performance comparison between Physical and Hypervisor was already presented in Deliverable D6.4 showing low (<10%) performance loss due to virtualization overhead, so we are very confident that performance evaluation results will be consistent if the methodology is reproduced in a different testbed using the same underlying IaaS and MCN platform.

The IMSaaS service was extensively evaluated, namely to characterize its performance within different deployment models: hardware vs. virtual. The achieved results, demonstrate that the media plane components on virtualized environments do not differ substantially from those deployed directly on the hardware. The Virtualized environments introduces marginal delays due to the additional entities involved in the networking path. On the other end the migration to the Open Baton NFV Orchestrator simplified the way IMS SI deployments are realised. First of all it is possible to make use of the internal catalogue of Open Baton to combine on demand different type of Service Template Graph, without having to recompile the Orchestrator and build a new container for it. Additionally, using the Open Baton Orchestrator as single responsible entity for multiple SIs allows to remove OpenShift from the path, reducing the time to deploy a new SI just to the time needed for deploying SICs (as shown in the evaluation section of IMSaaS the overhead of the SO/SM is almost null). Finally, the employment of the Open Baton Orchestrator allows also the integration of other components part of the Open Baton ecosystem. Indeed, its modular architecture allows the on the fly integration of several external entities which can be used for enhancing the usability of this management system.

Finally, the novelties introduced in the RAN service verify the feasibility of transporting I/Q samples from a passive RRH to a BBU over Ethernet. In terms of performance, we have demonstrated that the user can experience the same quality of service, but this does not mean that the introduction of an extra software component along with the transportation delay of radio traffic over Ethernet does not make the deadlines related to LTE network stack more difficult to meet. In short, the concept of the Ethernet-based fronthaul has proved to be feasible but in scenarios that are not particularly challenging in terms of volume of radio data to be transported and of the characteristics of the network between the RRH GW and the BBU. In order to make Ethernet-based fronthaul a standard common solution, the primary goal is to make Radio-over-Ethernet a standardized protocol, where the Ethernet protocol itself has to be changed, so that its best effort nature is altered to comply with the timing constraints imposed when radio data are transported.

In conclusion, the overall achieved results demonstrate clearly the benefits of adopting the network function virtualization and cloud-native features for telco core services and for enterprises aiming to migrate from private to cloud infrastructures. Where stringent SLAs are present, five 9s of availability (99.999%) can be met with customized fault tolerance mechanisms, no matter the type of service targeted.

References

MCN Deliverables

MCN D2.1 Reference Scenarios and Technical System Requirements Definition

MCN D2.2 Overall Architecture Definition, Release 1

MCN D2.3 Market Analysis and Impact of Mobile Cloud Concepts

MCN D2.4 Development of Business Models and Strategies for Adaption and Change Management

MCN D2.5 Final Overall Architecture Definition, Release

MCN D3.1 Infrastructure Management Foundations – Specifications & Design for MobileCloud framework

MCN D3.2 Infrastructure Management Foundations – Components First Release

MCN D3.3 Infrastructure Management Foundations – Components Final Release

MCN D3.4 Infrastructure Management Foundations – Final Report on component design and implementation

MCN D3.5 Infrastructure Management Foundations – Final Report on component design and implementation, May 2016

MCN D4.1 Mobile Network Cloud Component Design

MCN D4.2 First Mobile Network Cloud Software Components

MCN D4.3 Algorithms and Mechanisms for the Mobile Network Cloud

MCN D4.4 Final Mobile Network Cloud Software Components

MCN D4.5 Mobile Network Cloud Component Evaluation

MCN D4.6 Mobile Network Cloud Software Components and Report, May 2016

MCN D5.1 Design of Mobile Platform Architecture, IMSaaS and DSN applications

MCN D5.2 First Implementation of IMSaaS, DSN and Mobile Platform

MCN D5.3 Final Implementation of IMSaaS, DSN, and Mobile Platform

MCN D5.4 Evaluation of Mobile Platform, IMSaaS and DSN

MCN D5.5 Evaluation of Mobile Platform, IMSaaS and DSN, May 2016

MCN D6.1 Initial Report on Integration and Evaluation Plans

MCN D6.2 Initial Report on Testbeds, Experimentation, and Evaluation

MCN D6.3 Final Report on Integration and Evaluation Plans

MCN D6.4 Final Report on Testbeds, Experimentation, and Evaluation