| FUTURE COMMUNICATION ARCHITECTURE FOR MOBILE CLOUD SERVICES |
|---|
| Acronym: MobileCloud Networking<br>Project No: 318109 |
| Integrated Project<br>FP7-ICT-2011-8<br>Duration: 2012/11/01-2015/09/30 |

# D5.5 Evaluation of Mobile Platform, IMSaaS and DSN

| | |
|---|---|
| Type | Report |
| Deliverable No: | 5.5 |
| Workpackage: | WP5 |
| Leading partner: | TU Berlin |
| Author(s): | Giuseppe Antonio Carella (Editor), List of Authors overleaf |
| Dissemination level: | PU |
| Status: | Final |
| Date: | 02 May 2016 |
| Version: | 1.0 |

# Versioning and contribution history

| Version | Description | Contributors |
|---|---|---|
| 0.1 | First integrated version | STT |
| 0.2 | Added content for IMSaaS | FOKUS, TUB, UNIBO |
| 0.3 | Integrated additional content for DSS | STT |
| 1.0 | Final integrated version | TUB |

## List of Authors:

| | |
|---|---|
| Giuseppe Carella | TUB |
| Philipp Emanuel Kuhnz | TUB |
| Luca Foschini | UNIBO |
| Riccardo Venanzi | UNIBO |
| Lars Grebe | FHG |
| Mohammad Valipoor | STT |
| Santiago Ruiz | STT |
| Zarrar Yousaf | NEC |

## List of Peer Reviewers:

| | |
|---|---|
| Bruno Sousa | ONE |
| Andy Edmonds | ZHAW |

# Table of contents

## Table of figures

## Table of tables

## List of Acronyms

| | |
|---|---|
| **AE** | Analytics Engine |
| **AS** | Affinity Score |
| **CDN** | Content Delivery Network |
| **CDNaaS** | CDN as a Service |
| **CMS** | Content Management System |
| **CSCF** | Call Session Control Function |
| **DE** | Decision Engine |
| **DSS** | Digital Signage Service |
| **DSSaaS** | Digital Signage Service as a Service |
| **E2E** | End to End |
| **EEU** | Enterprise End User |
| **FIP** | Floating IP |
| **HSS** | Home Subscriber Server |
| **IaaS** | Infrastructure as a service |
| **ICN** | Information-Centric Networking |
| **ICNaaS** | ICN as a Service |
| **I-CSCF** | Interrogating Call Session Control Function |
| **IMS** | IP Multimedia Subsystem |
| **IMSaaS** | IMS as a Service |
| **KPI** | Key Performance Indicator |
| **KVM** | Kernel-based Virtual Machine |
| **LBaaS** | Load Balancer as a Service |
| **MaaS** | Monitoring as a service |
| **MCN** | Mobile Cloud Networking |
| **MCR** | Main Content Repository |
| **MQ** | Message Queue |
| **OTT** | Over The Top |
| **P2P** | Point to Point |
| **P-CSCF** | Proxy Call Session Control Function |
| **PDN** | Packet Data Network |
| **P-GW** | PDN Gateway |
| **QoE** | Quality of Experience |
| **QoS** | Quality of Service |
| **RAVA** | Resource Aware VNF Agnostic |
| **RRAS** | Reference Resource Affinity Score |
| **RRU** | Reference Resource Unit |
| **RU** | Resource Unit |
| **S-CSCF** | Serving Call Session Control Function |
| **S-GW** | Serving Gateway |
| **SIC** | Service Instance Component |
| **SLF** | Subscriber Locator Function |
| **SM** | Service Manager |
| **SO** | Service Orchestrator |

| | |
|---|---|
| **SO_D** | Service Orchestrator Decision module |
| **SO_E** | Service Orchestrator Execution module |
| **VM** | Virtual Machine |
| **VNF** | Virtual Network Function |
| **VoD** | Video on Demand |
| **WP** | Work Package |

# 1 Introduction

This deliverable describes the work accomplished during the last MCN project extension period for the DSS and IMS services. The deliverable including following aspects:

- Innovations and Extensions to the IMS and DSS services. The section covers the motivation, design, development and testing of each innovation/extension. Specific IMS extensions include integration of a new fully compliant NFV Orchestrator (Open Baton), the introduction of media plane components and fault management support. Specific DSS extensions cover the integration of RAVA technology as an extension of the Service Orchestrator, dynamic decentralized content replication for MCR media contents and fault tolerance with auto recovery for all the service components.
- Performance methodology evaluation for IMS and DSS service. For each service, common WP3 performance methodology evaluation is adapted to fit specific service constraints. For each service a final set of KPIs is selected, and a set of scenarios is defined in order to obtain measurements.

Deliverable aims to demonstrate how performance methodology evaluation can be applied to different service types (telco and over-the-top). Implemented extensions provide significant improvements that are directly translated in better service KPIs. For the KPIs that were not considered in the common performance evaluation, but are interesting or valuable, such as fault management and live migration a new set of extended KPIs have been added.

# 2 IMS as a Service

This section provides the innovations addressed for the IMS service as well as the changes needed to enhance the cloudified version of IMS which are described in the previous deliverables [1][2][3][4].

## 2.1 Definition and Scope

Previous deliverables [1][2][3][4] introduced the possible deployment models of the IMS Software Components in order to make it cloud-enabled, the mechanisms required for scaling in and out particular Service Instance Components (SIC), and provided an evaluation of the signalling plane components.

This section introduces some innovative concepts which can be considered for the IMS scenarios. Firstly, it introduces the integration between the Open Source Open Baton NFV Orchestrator and Hurtle. Then it introduces some Fault Management mechanisms employed for enhancing the reliability of the IMS Services. Finally, it describes the integration of a Media Plane component into the IMS architecture and provides an evaluation of the virtualised Media Plane.

## 2.2 Innovative features

The main innovative concepts introduced in the last period are listed below:

- **Integration of the Open Baton NFV Orchestrator and Hurtle**: the Open Baton project provides a fully compliant implementation of the NFV MANO v1.1.1 specification [8]. In particular, it provides a NFV Orchestrator capable of instantiating multiple Network Services and integrate with different cloud systems. Integrating Hurtle and Open Baton provides an innovative approach for managing Virtual Network Functions (VNFs) in a multi domain scenario.
- **Fault Management of IMS SICs**: reliability, and in particular high-availability, it is a very important aspect for standard telco services. A Fault Management System (FMS) has been implemented and integrated into the Open Baton Service Orchestrator, and employed in the IMS scenario for managing high-availability at the Home Subscriber Server (HSS) level
- **Integration of a Media Plane Component**: an open source implementation of a Media Gateway has been selected and integrated within IMSaaS. It provides the capabilities of managing Media Sessions between IMS Subscribers. It has been used particularly for evaluating the Media Plane on top of a virtual infrastructure

### 2.2.1 Integration of the Open Baton Orchestrator and Hurtle

As already extensively presented in the previous deliverables, to provide the functionality "as a Service", the lifecycles of MCN Services are managed through a Service Manager (SM) and a Service Orchestrator (SO) combination, having as reference implementation the hurtle framework proposed in the context of WP3. Through the SM the Enterprise End User (EEU) can use the OCCI APIs to deploy, provision and dispose Service Instances (SIs) and retrieve their status-information.

In order to manage the lifecycle of each Service Instance, the Hurtle approach proposes to instantiate SO on demand via the SM interface. This combination allows the SM to provide a gateway to numerous service instances for multiple consumer while the SO itself remains responsible for one specific deployment, only being accessed by the overlaying management.

However, in the very recent period there was an increasing interest in having Orchestration solutions compliant with the ETSI NFV MANO information model in order to reduce the fragmentation between different existing solutions. Considering that MCN provides a flexible architecture for supporting different vertical domains, what was realized during the extension period was the integration of an open source Orchestrator which is fully compliant with the ETSI NFV MANO information model. The Orchestrator selected is Open Baton[1], implemented by Fraunhofer FOKUS and TU Berlin, providing a very extensible architecture supporting multiple different use cases.

Open Baton provides several functionalities for:

- Instantiating virtual resources on top of the NFVI as required by the Network Service (in our case the IMS SI).
- Send lifecycle events to the different VNF Managers for the instantiation of the VNFs.
- Runtime management of the Network Service with the Autoscaling and the Fault Management Systems.

Open Baton is implemented in Java using the Spring Framework [2]. Figure 1 shows the Functional Architecture of the Open Baton framework.

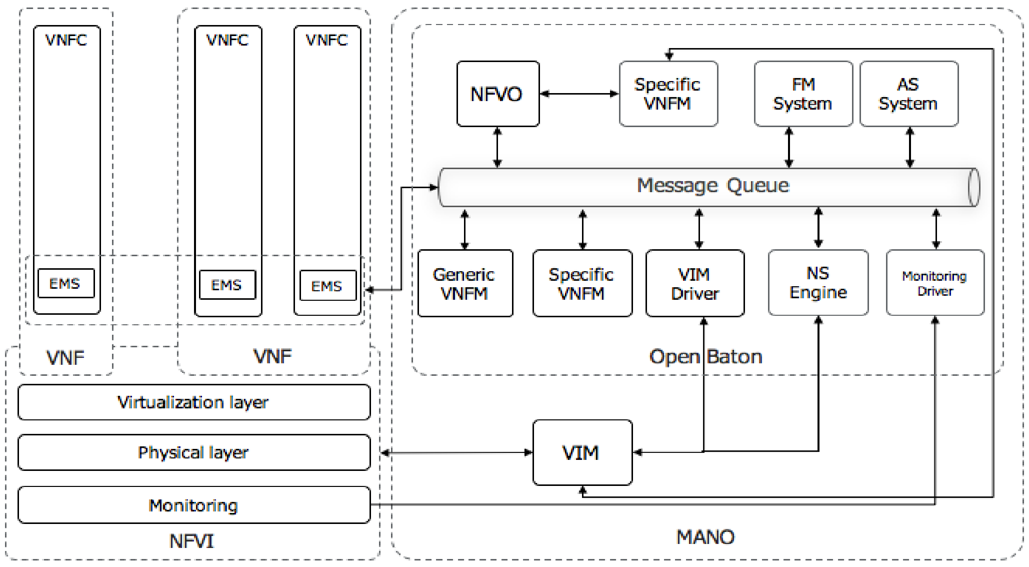

**Figure 1. Open Baton Functional Architecture**

It communicates with one or more VNF Managers via a REST API or messaging system. In the scope of the MCN extension the second way was chosen as it provides a more extensible way for integrating multiple VNF Managers, and as Open Baton provides a Generic VNFM which integrates via the

[1] Open Baton: http://openbaton.org

[2] Spring Framework: http://spring.io/

message bus. The messaging system uses the AMQP protocol and its implementation is based on RabbitMQ. In particular, the NFVO instantiates:

- Two queues for allowing external entities to register/deregister to particular events (instantiation of NSR finished, deletion of a NSR, etc.)
- Two queues for registering/deregistering VNFMs
- One queue used by all VNFMs for sending back answers of each lifecycle event execution

In the context of this project it was employed only the Generic-VNFM which provides generic capabilities for controlling the lifecycle event of a VNF based on its definition in the Network Service Descriptor (NSD) which corresponds to the STG defined in MCN. This VNFM was used for controlling all the IMS SICs. Only few changes to the scripts implemented as part of the Element Management System (EMS) have been required in order to integrate the IMS SICs into the Open Baton NFV Catalogue. Those changes are mainly due to the different way Open Baton is naming variables which are computed during the dependency resolution.

Figure 2 shows how the integration has been realized.

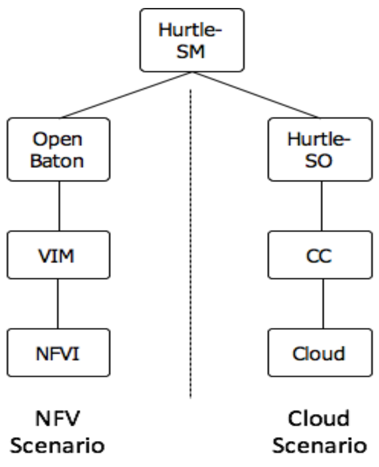

**Figure 2. Open Baton - Hurtle integration**

To let Open Baton to be used as a SO by the SM, an API providing the same functionality with nearly identical signatures is being exposed. It offers deploy, provision, status information and disposing of service instances. While in the hurtle-so approach one SO would manage only one service instance, in the Open Baton scenario, the NFVO manages multiple service instances, each identified by a universally unique identifier, which is required for every request except for the initialization. For this reason, it is also not necessary to deploy on demand a new Service Orchestrator instance.

Corresponding to this, the responsible SM is using a specific Service Orchestrator Manager implementation. It includes the mentioned unique service instance identifier in its requests so that it will do the requests to the correct service instances. While the reference manager implementation makes usage of PaaS like OpenShift to deploy and dispose service orchestrators on demand, this is not needed for the Open Baton approach, container creation is totally skipped and the whole lifecycle management is passed over to Open Baton.

Usage of a service manager using this manager implementation does only differ to the generic implementation in the need to specify host and port for the Open Baton in the service manager configuration.

As can be seen in Figure 3 the Enterprise End User requests the instantiation of the IMS SIC at the H-SM endpoint. This interface hasn't been changed, and its APIs description is available in D2.4 and D3.4.

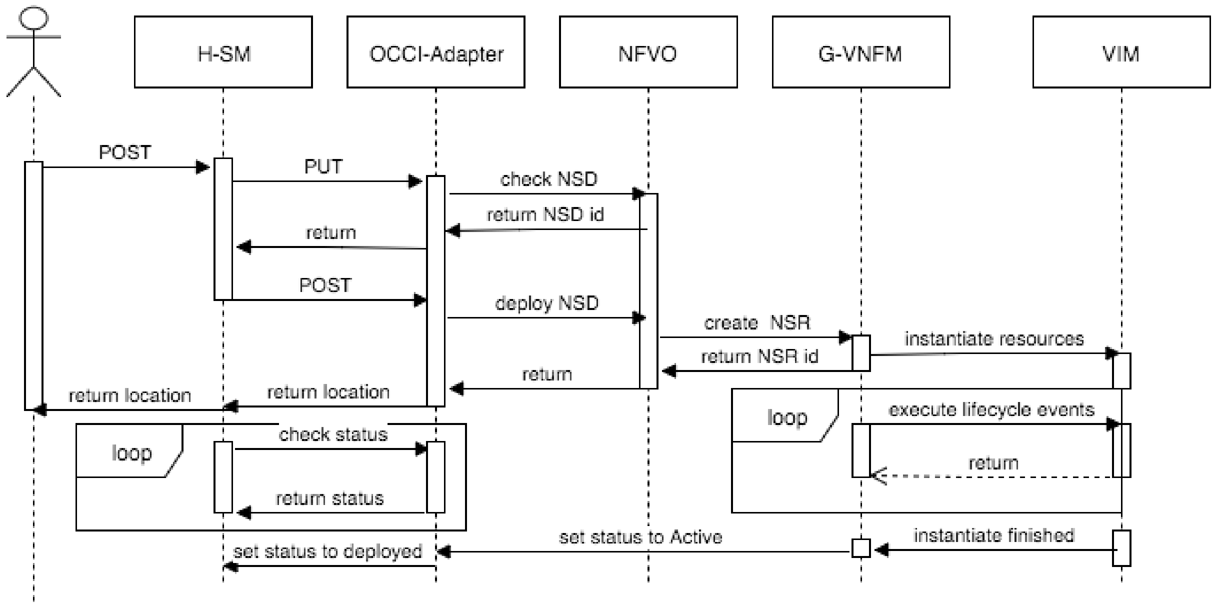

**Figure 3. Sequence Diagram of the IMS SI deployment**

Once the request is received by the SM, it calls internally the Open Baton Service Orchestrator Manager class which has been leveraged for supporting multiple types of Service Orchestrators. Figure 3 shows the sequence diagram of the operations which are executed when a user requests the instantiation of a network service.

The different steps are explained in the following list:

1) The Enterprise End User requests the instantiation of a IMS Service Instance via a POST request to the respective Service Manager endpoint (H-SM).

2) The H-SM requests the instantiation of the IMS SI to the Open Baton NFVO via the OCCI-Adapter. In particular, it first sends a PUT request for initializing it, and then sends a POST request for deploying the IMS SI.

3) Considering that the Open Baton NFVO is capable of orchestrating multiple types of Network Services (multiple types of STG in the MCN terminology), the OCCI-Adapter should request the instantiation of a particular Network Service via its unique identifier. This identifier is configured on the OCCI-Adapter at the installation time, but can be also passed in a OCCI-Attribute while doing the instantiation procedure.

4) Once the POST request has been received by the OCCI-Adapter, it requests the instantiation of the IMS SI NSD, which turns into a series of internal operations executed at the NFVO level. Those operations are usually associated with the execution of lifecycle events: instantiation of virtual resources required as described by the ITG, and execution of installation and configuration of each Service Instance Component with the support of the Generic VNFM (G-VNFM).

5) Once all the operations are concluded successfully, the IMS SI status is set to ACTIVE and the H-SM set the status to deployed.

#### 2.2.1.1 Code documentation and installation guides

Refer to the git repository for the code documentation of the NFV Orchestrator:

https://github.com/openbaton/NFVO

Refer to the git repository for the code documentation of the OCCI-Adapter:

https://github.com/MobileCloudNetworking/OpenBaton-OCCI

### 2.2.2 Fault Management of IMS SICs

Fault Management is considered one of the most important characteristics of a NFV Framework, and it has big impacts on the Quality of Experience (QoE) and Quality of Service (QoS). However, in such a dynamic and complex environment, faults may occur at every levels: Physical layer, Virtualization layer, and Virtual Network Function layer. Considering the case of the IMS Service Instance, as already mentioned in previous deliverables, it is possible that each Service Instance Component (SIC) is instantiated on top of different Virtual Machines (VMs). Those VMs could also be executed on different Physical resources (Compute Nodes), and that particular physical resource could be affected by a fault which may impact the execution of the IMS SICs. Figure 5 shows an example of a typical deployment model (based on the deployment model selected and described in previous deliverables) in which one fault occurs at the level of the physical resource executing the HSS SIC.

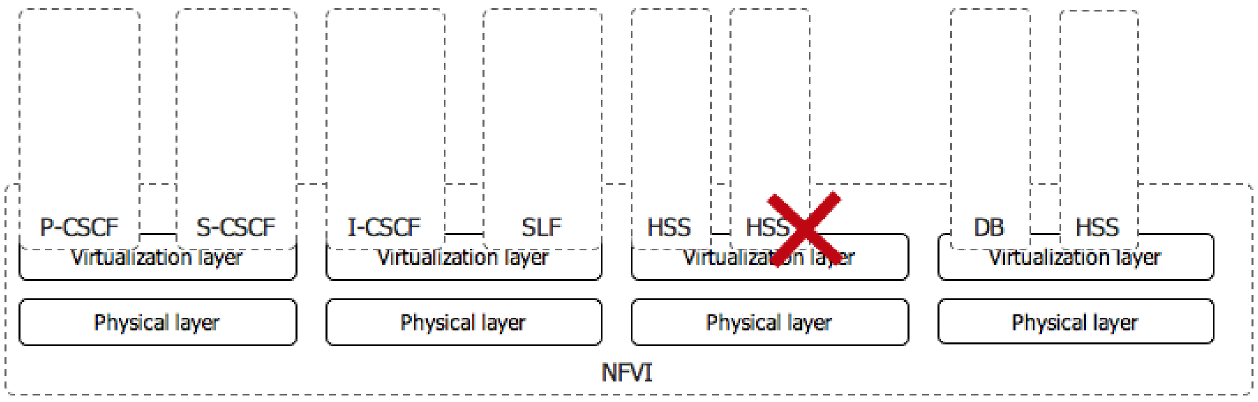

**Figure 4. Example of fault occurring at the Virtual layer**

It is important to underline that for having a fully reliable solution, it is necessary not only to introduce fault management operations, but also support high-availability at the level of the Service Instance itself. In our case, we have employed the elasticity scenario implemented in D5.4 in order to provide also high-availability, and we have applied fault management operations to this scenario.

One of the major issues that needs to be considered while implementing a Fault Management System (FMS) for a NFV environment, is that due to the increased complexity of this multi-layered environment there maybe many scenarios where an alarm storming is occurring. Let us consider as an example the CPU failure in a Compute Node. When this fault occurs, the Virtual Network Functions executing in a Virtual Machine running on top of this Compute Node will be affected. The first thing happening is that Monitoring as a Service (MaaS) will trigger an alarm to the VIM, notifying that a problem occurred at the Infrastructure layer. Additionally, the VNFM responsible of the lifecycle of the VNFs executing on that Compute Node, will also receive a fault notification from the VNF.

In such conditions, both systems, the VNFM and the VIM, could take independent decisions triggering resolution actions which may led in potential conflicts on the final IMS SI.

Our proposed approach, is to provide a Fault Management System which aggregates the faults notifications and triggers corresponding resolution actions based on policy which are defined as part of the Network Service Descriptor.

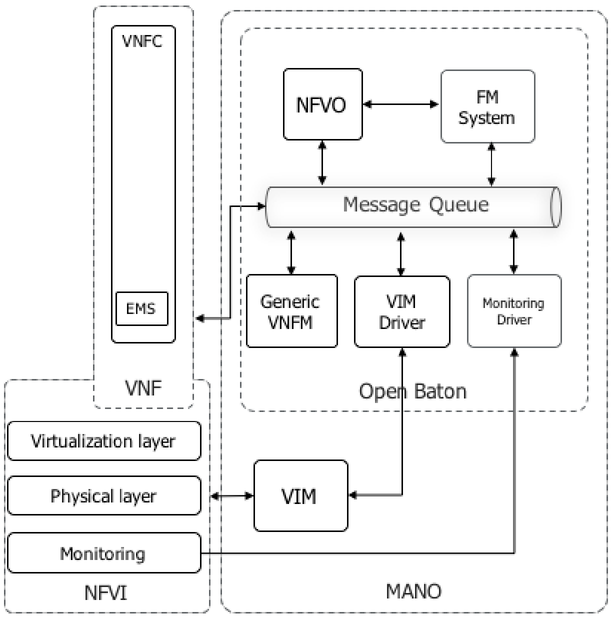

**Figure 5. Fault Management System Open Baton Architecture**

The Fault Management Policy has been introduced in order to define what are the KPIs which can be considered for detecting fault situations. For instance, the policy shown below is used for detecting a fault at at the level of the HSS SIC: whenever the Diameter port (3868) is not available.

```
"fault_management_policy":[
    {
      "name":"HSS SIC not available",
      "isVNFAlarm": true,
      "criteria":[
      {
        "parameter_ref":"net.tcp.listen[3868]",
        "function":"last()",
        "vnfc_selector":"at_least_one",
        "comparison_operator":"=",
        "threshold":"0"
      }
      ],
      "period":5,
      "severity":"CRITICAL"
    }
]
```

The one introduced defines a set of criteria to detect a failure, referring to a monitoring parameter, whether the failure is at the VNF level or the Virtualization level, and additional parameters used for declaring the severity of the failure.

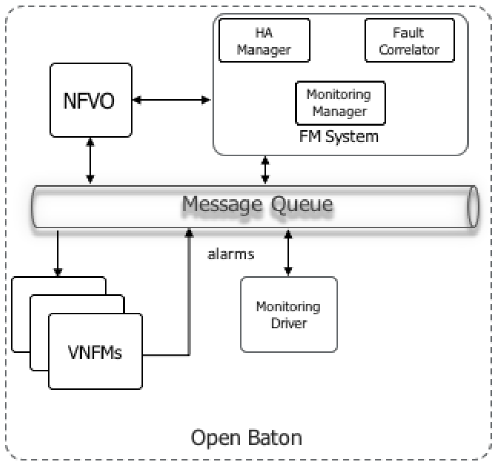

**Figure 6. Internal Architecture of the Fault Management System**

The FMS Architecture is composed by three main components, as illustrated in Figure 6:

- The High Availability Manager (HA Manager) in charge of maintaining the redundancy scheme of the Service Instance and to execute the recovery actions. In particular, it is using the Switch-to-Standby mechanism, so that whenever a fault occurs the system will automatically identify the faulty SIC as faulty and activate the pre-prepared instance, which is in standby mode.
- The Fault Correlator correlates alarms and finds the root cause. In particular, here it is the place where different sources of alarms are correlated and decisions are taken depending on the policies that are defined.
- The Monitoring Manager utilized at the deployment time for creating application specific metrics and alarms on MaaS. It uses the Monitoring Driver as intermediate component for decoupling the internal logic from the specific monitoring implementation. In this project, it has been implemented a Driver for interacting with Zabbix (the monitoring system instantiated via MaaS)

As already mentioned in previous sections, the fault management scenario has been applied to the HSS SIC, as it is the component for which high-availability has been implemented during the last year of the MCN project.

#### 2.2.2.1 Code documentation and installation guides

Refer to the git repository for the code documentation of the NFV Orchestrator:

https://github.com/openbaton/NFVO

Refer to the git repository for the code documentation of the Fault Management System:

https://github.com/openbaton/fm-system

### 2.2.3 Multimedia Data Plane and Media Gateway Integration

To enable the full understanding of Media Gateway (MGW) integration and its elastic scaling, in the following subsections, we first provide some needed background about enabling and controlling the multimedia data plane in IMS, and then we detail how through our MCN Application Server (AS) we enable MGW elastic scale in/out.

#### 2.2.3.1 Multimedia Application Server (AS) for Media Gateway Integration

In the following subsections, we rapidly recall the four main components of the IMS infrastructure that directly participate in the multimedia session integration.

As already described since deliverable D5.1 [1], the first component is the IMS Client, which controls session setup and media transport by implementing all SIP extensions specified by IETF and 3GPP IMS-related standards. Any session is setup between two IMS clients playing the roles of User Agent Client (UAC) and User Agent Server (UAS). The second IMS component is the Multimedia Application Server (MMAS), which allows the introduction of new IMS services and extensions, including the MGW. AS has full control over traversing SIP dialogs and SDP descriptions. The third one, the Serving-Call Session Control Function (S-CSCF), is the most important session control entity. S-CSCF receives register requests from IMS clients and authenticates them; then, depending on filters/triggers specified by client profiles and dynamically downloaded from the HSS, S-CSCF may either route incoming SIP messages from UAC directly to UAS or forward them to the AS. The last entity of interest is the MGW, an external component controlled by IMS that takes part to the media delivery plane between UAC and UAS.

Hence, given its ability to re-route traffic, the S-CSCF can extend UAC-to-UAS session signaling paths through the interposition of convenient ASs; these ASs may participate to multimedia content transport/buffering/adaptation as well as enabling elastic scalability, as better explained in the next section, by interacting with media gateways suitably deployed at the IMS media plane. Other primary IMS components, such as Proxy-/Interrogating-CSCF and Home Subscriber Server (HSS) – that interact with S-CSCF respectively for user authentication and SIP message routing – are not overviewed because they are out of scope in the extension period. For a comprehensive introduction to the whole IMS architecture and for the description of all IMS components, we refer interested readers to the previous deliverables [1][2][3].

Delving into finer details, the integration approach adopted in MCN follows the IMS specification. The resulting distributed architecture, shown in Figure 7, consists of two main components that interwork together to glue the session control plane and the multimedia data plane: MMAS and MGW. MMAS, deployed at the UAC home network, participates to IMS session signaling and controls the MGW as required by ongoing session dialogs by acting as its stub at the session control plane (steps 1 and 2 in Figure 7). MGW, at the data plane, works as a middlebox component that splits the (otherwise direct end-to-end) multimedia data path between UAC and UAS (step 3) by enabling several possible advanced multimedia facilities such as adaptation, mixing, and handoff of traversing multimedia flows. In our prototype, for the MGW we opted for the widely diffused and supported Asterisk server. Asterisk can interact with various session signaling control protocols, e.g., H.323, Media Gateway Control Protocol (MGCP), SIP, and the proprietary Inter-Asterisk eXchange (IAX). For MMAS-to-MGW interaction, we used SIP, given its wide diffusion, also beyond IMS; in addition, this choice simplifies MGW integration with MMAS because MMAS uses SIP to interact with existing IMS frameworks anyway.

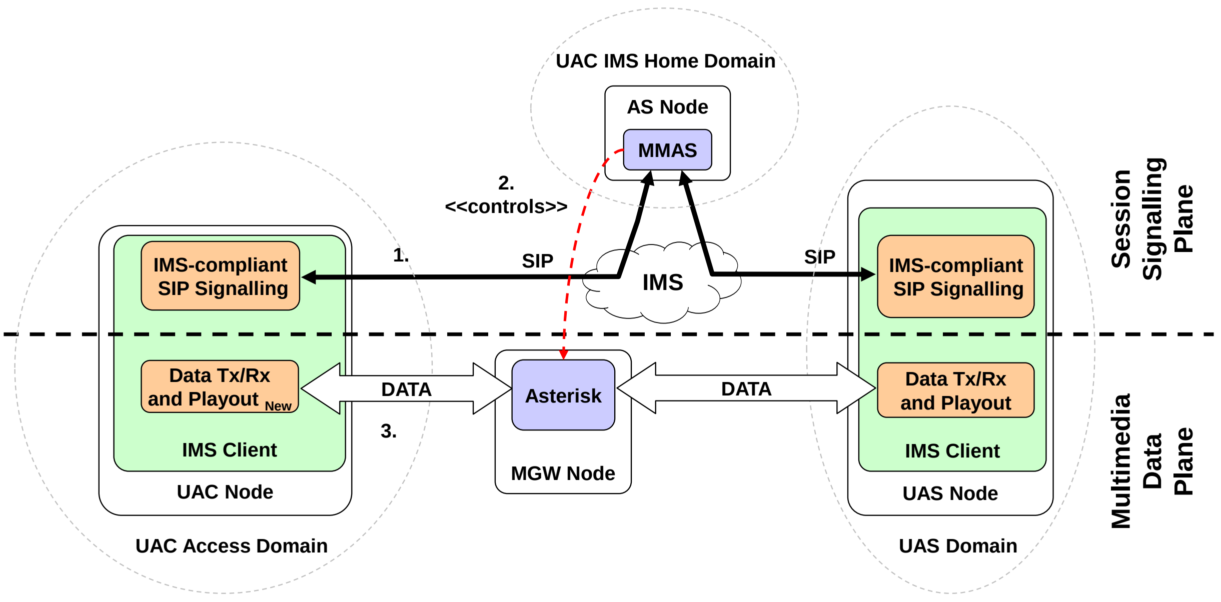

**Figure 7. IMS distributed architecture for integration of the multimedia data plane in MCN**

In particular, Figure 8 shows all main interactions and implemented protocols at the session control and multimedia data planes. Without loss of generality, at the data plane we will focus on the multimedia flow in one-way direction that goes from UAS to UAC because this is the scenario typically considered and supported by most benchmarking tools, and it is the one that is employed in our performance analysis, being detailed in the next sections and on deliverable D6.5. The initial registration phase (not shown in Figure 8) allows to dynamically install UAC IMS filter criteria at the S-CSCF$_{\text{UAC}}$, so to correctly interpose MMAS in the session control path. Thereafter, when UAC generates a new outgoing call, S-CSCF$_{\text{UAC}}$ forwards the received INVITE message (step 1 in Figure 8) to MMAS that in its turn, working as a Back-to-Back User Agent (B2BUA), includes the MGW at the multimedia data plane by requiring the setup of a new data session in Asterisk. To do that, MMAS exploits Asterisk SIP-based control interface that allows negotiating through Session Description Protocol (SDP) the RTP flow encodings and endpoints as required by UAC and UAS (all steps shown as dashed arrows in, namely, steps 3-4, 7-8, 11-13, 17-18). That allows to split the RTP communication between UAC and UAS through the interposition of Asterisk at the data plane binding, respectively, UAS with Asterisk and Asterisk with UAC. Finally, the multimedia flow is established (steps 20, 21): the RTP multimedia flow starts at UAS and goes through Asterisk to reach UAC eventually.

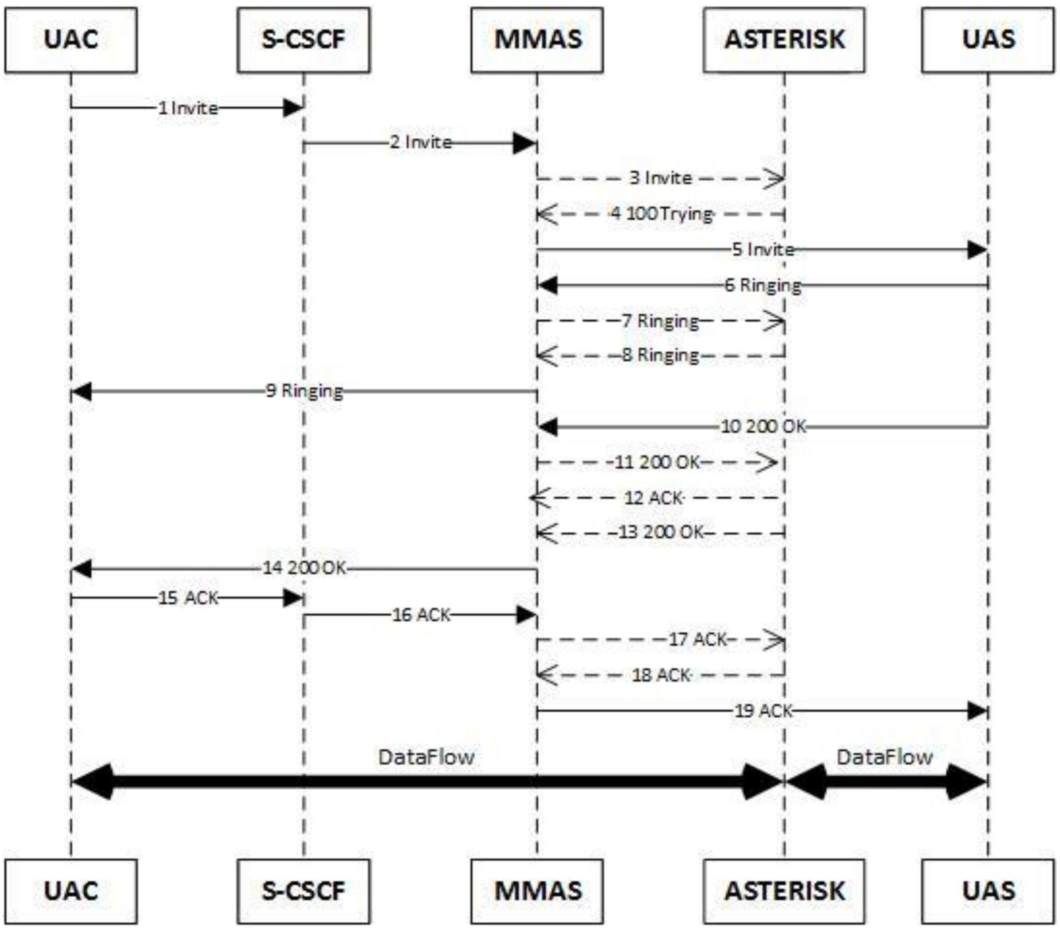

**Figure 8. Multimedia data plane integration protocol**

The integrated session control/multimedia data plane scenario and protocols described atop are the one used during the assessment of the IMSaaS multimedia data plane, as better described in the following sections.

#### 2.2.3.2 Code documentation and installation guides

Refer to the git repository for the code documentation of the IMS Multimedia Application Server:

https://github.com/MobileCloudNetworking/asterisk-sip-as

#### 2.2.3.3 Elastic scalability of the MGW function in MCN

This second section is aimed to detail better how it would be possible to enable scale-out/-in of the MGW function. The overall goal of this protocol proposal is to scale the MGW that is a critical component at the multimedia data plane and might become the overall system bottleneck, especially for high numbers of concurrent multimedia data flows.

Our scale-out/-in integrates with MCN and is based on three main components, namely, MaaS, NFVO, and our MMAS, that work in a pipeline to operate and obtain needed scalability management actions. First of all, we exploit the MaaS monitoring service for triggering scale-out/-in operations as needed; in particular, MaaS monitors the execution of the most critical MGW and can raise monitoring alarms to the NFVO that, for the sake of simplicity, here we consider both as the scale-out/-in decision entity and orchestrator of the whole scale process. Hence, once alerted, and in case of overloading of a MGW instance (for the sake of simplicity, let us assume to have initially one only overloaded MGW instance, namely, $MGW_1$), NFVO can horizontally scale-out the MGW, by adding another instance (namely, $MGW_2$ in Figure 9). Finally, the MMAS behaves like a dynamic load-sharing for the incoming session establishment requests at runtime, able to exploit all available MGW instances. The load sharing is performed by an elastic creation of multiple MGW instances that allows to dynamically adapt to the overall load at runtime, providing a scalable approach. In Figure 9, we illustrate the elastic MGW scalability with a sequence diagram.

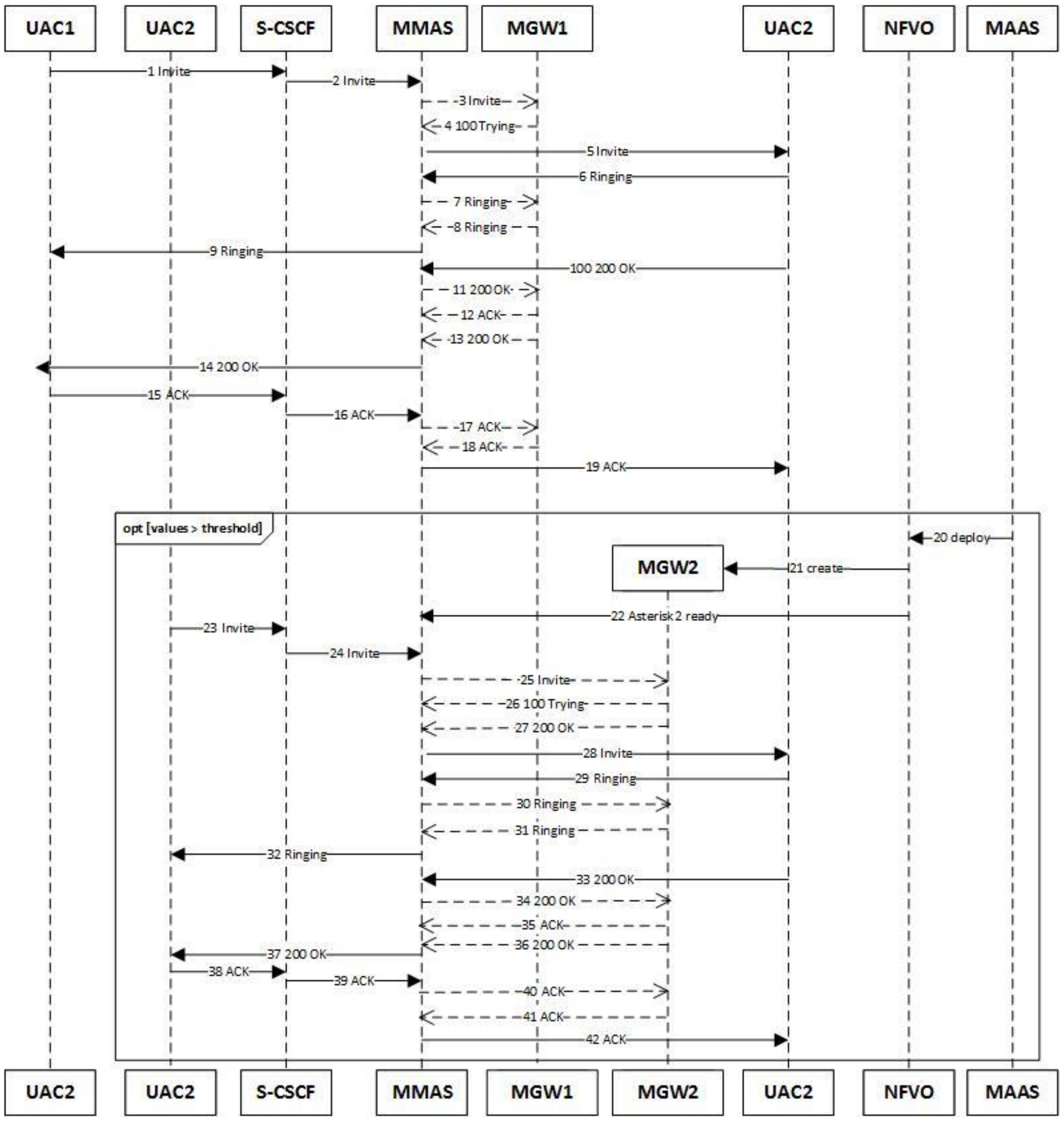

**Figure 9. Elastic MGW scalability in MCN**

After an initial usage of the data plane integration protocol (step 1-19), already detailed in the previous section, we use MaaS to monitor and keep track of the MGW SIC(s) load via Zabbix. If the monitoring reading values exceed the target pre-defined threshold, the related trigger will activate and the MaaS component raise an alarm that sends a request towards the NFVO in order to create a new Asterisk instance (steps 20-21). Once the new Asterisk instance is created and ready, the NFVO informs the MMAS about the availability of the new Asterisk instance (step 22); this step is very important and allows MMAS to update its internal state, thus making available for upcoming new session requests the new MGW. As new INVITE requests arrive, the data plane integration protocol procedure takes place (steps 23-42) and shares the load between available MGWs; the load-sharing algorithm may be either static, such as a hash calculated over the incoming UAC user id, or more dynamic, such as taking a decision based on the current load of MGWs. In the example shown in Figure 9, when new clients UAC requests arrive to the MMAS, the MMAS shares the incoming traffic dynamically and, thus, forwards the requests towards the most newly activated MGW2.

Let us conclude noting that we believe supporting elastic scalability through load sharing is sufficient, rather than using more dynamic load balancing support (i.e., able also to move ongoing sessions between old and new MGWs). In fact, load balancing support would require appropriate APIs to extract the current RTP flows state from the old MGW and to install it at the new MGW. Such APIs are currently not supported by most COTS MGW servers, also because it is usually assumed that the session duration is short enough, such as for phone services, to assume that load sharing is sufficient to distribute fast the load in a short time from the old to new MGW, as ongoing sessions at the old MGW terminate.

## 2.3 IMSaaS Performance Evaluation

Performance evaluation described in this section has been performed following the guidelines provided by WP3 in Deliverable D3.5. In particular, the approach taken 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 which are required for a specific workload
- Life-cycle KPIs related with the management and orchestration operations for the life-cycle of those network functions
- Cloud-native KPIs mainly referring to high-availability and elasticity

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 been presented and extensively evaluated in D5.4 and D6.4, however some new scenarios have been proposed and implemented in order to collect 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 type of scenarios of Registration, Invite, Messages is generated using the IMS Bench Tool [14]. This scenario is used mainly for evaluating the Signaling process, and some of its results have been presented into D5.4. Additional results will be presented in the context of D6.5.
- 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 using the SIPp Bench Tool [14]. This scenario is used mainly for evaluating the Media Plane over different technologies. Its results will be presented into D6.5.
- 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 technologies. Its results will be presented into D6.5.
- Scenario 5: Stairs workload on Media Plane components. The IMS Service Instance is running. Artificial load executing a stairs increasing number of Invite sessions is generated using the SIPp Bench Tool. This scenario is used mainly for evaluating the Media Plane over different technologies. Its results will be presented into D6.5.
- 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 technologies. Its results will be presented into D6.5.
- Scenario 7: Disposal of the IMSaaS SI is performed by the EEU.

### 2.3.1 KPI Mapping

This section provides a mapping between the KPIs proposed on D3.5 and the scenarios which will be used for collecting them. It is important to clarify that some of the evaluation results extracted from those scenarios have been already presented in D5.4 and D6.4. Some other extended results will be presented in D6.5.

#### 2.3.1.1 System KPIs

System KPIs provides metrics for service quality evaluation. Considered KPIs are depicted in Table 1:

**Table 1. System KPIs**

| KPIs | Considered for IMSaaS | Adaptation / Detail | Scenarios |
|---|---|---|---|
| 1 registration success rate | Yes | Number of subscribers successfully registered | 2 |
| 2 session establishment success rate | Yes | Session establishment: Subscriber connection request success & content response. | 2,3,4,5,6 |
| 3 session drop rate | Yes | Number of dropped sessions | 3,4,5,6 |

| | | | |
|---|---|---|---|
| 4 attachment delay | Yes | Registration delay | 3,4,5,6 |
| 5 session establishment delay | Yes | Time required to perform an INVITE | 3,4,5,6 |
| 6 data plane QoS | Yes | QoS parameters related with the data plane | 3,4,5,6 |
| 7 data plane delay | No | End-to-end delay on the data plane | |

Table 1 presents a set of KPIs which are usually considered while dealing with Telco environments. In order to measure system KPIs, IMS is previously deployed and provisioned. As explained before, some of the KPIs have been adapted to the specific functional constraints of the IMS service. For simplicity reasons, measurements for system KPIs will not consider any lifecycle or cloud-based operation (deployment, provisioning, update, scale, migration, disposal, etc.) since those will be considered in the following subsections.

#### 2.3.1.2 Life-Cycle KPIs

Life cycle KPIs aims to measure the behaviour of a IMSaaS instance focusing on deployment, provisioning, disposal and installation times. Considered KPIs are depicted in Table 2:

**Table 2. Life-Cycle KPIs**

| Lifecycle KPIs | Considered for IMSaaS | Adaptation / Detail | Scenarios |
|---|---|---|---|
| 8 installation duration | Yes | Time required to install the SIC images in a new cloud environment | - |
| 9 deployment and configuration duration | Yes | Time to deploy and provision a service instance | 1 |
| 10 disposal duration | Yes | Time to dispose a service instance | 7 |
| 11 service upgrade duration | No | Overall upgrade time. Composition of the previous KPIs | |

Scenarios required to perform measurements of KPIs lifecycles are S1 and S7 performing a complete service instantiation and disposal. For lifecycle KPIs, it is considered a standalone service instance deployment. The main reason is to evaluate the performances of the newly introduced orchestrator system. The KPI11 is not considered as there are no mechanisms implemented in the context of the IMSaaS SI for upgrading a SIC.

#### 2.3.1.3 Cloud-Native KPIs

Cloud native KPIs aims to measure the behaviour of a IMSaaS instance focusing on automated cloud operations to achieve service elasticity and fault tolerance. Considered KPIs are depicted in Table 3:

Table 3. Cloud-Native KPIs

| Cloud native KPIs | Considered for IMSaaS | Adaptation / Detail | Scenarios |
|---|---|---|---|
| 12 Single component deployment latency | Yes | Time to deploy each of the SICs composing service | 1 |
| 13, 14 Scale in/out management latency (ctrl plane) | Yes | Scale in/out perceived user time for each SIC | 2 |
| 15, 16 Scale in/out operation number (ctrl plane) | Yes | Number of scale in out operations per SIC | 2 |
| 17 Useless scale operation number (ctrl plane) | Yes | Number of ping-pong scaling ops | 2 |
| 18, 19 Scale in/out management latency (data plane) | No | Cannot be evaluated without a scaling mechanism for the data plane | |
| 20, 21 Scale in/out operation number (data plane) | No | Cannot be evaluated without a scaling mechanism for the data plane | |
| 22 Useless scale operation number (data plane) | No | Cannot be evaluated without a scaling mechanism for the data plane | |
| 23 Time percentage of control plane availability | Yes | Availability after initial provisioning performing scaling ops / component failure | 2 |
| 24 Time percentage of data plane availability | No | Cannot be evaluated without a scaling mechanism for the data plane | |
| 25 Overall reconfiguration latency | No | | |

The Cloud-Native KPIs presented in the context of the Evaluation Methodology of D3.5, are useful for evaluating the performances of the IMS SI during scaling operations. However, in order to have some evaluation results for the control and data plane scalability it is required not only to have autoscaling at the level of the management and orchestration components, but also high-availability at the level of the SICs. In the case of the IMS SI, we provide high-availability at the HSS level, which has been employed for evaluating the results of the control plane KPIs. Regarding the data plane KPIs, considering that it was out of the scope of this extension period an implementation of a horizontally scalable data-plane, we assume that with the proposed solution of section 2.2.3.3 there should be no impact on the System KPIs of the data plane while executing the scaling operations. The main reason is that the MMAS acts as signalling load balancer, and decides where to place new sessions based on the availability of the MGW components.

#### 2.3.1.4 Resources KPIs

Resources KPIs aims to measure the behaviour of a IMSaaS instance focusing on hardware resource consumption of individual service components. Considered KPIs are depicted in Table 4:

Table 4. Resource KPIs

| Resources KPIs | Considered for IMSaaS | Adaptation/Detail | Scenarios |
|---|---|---|---|
| 26 - CPU usage | Yes | CPU utilization per SIC | 2,3,4,5,6 |
| 27 - Memory usage | Yes | Memory usage per SIC | 2,3,4,5,6 |
| 28 - Network usage | Yes | Inbound/outbound network usage per SIC | 3,4,5,6 |
| 29 - External network usage | No | Inbound/outbound network usage between testbed and the external Clients | |
| 30 - Storage usage | No | Storage utilization per SIC. Irrilevant for the IMS SICs | |
| 31 - Number of VMs used | Yes | Number of VMs used (IMS composed services) | 1,2,3,4,5,6,7 |
| 32 - Resources consumed for orchestration | Yes | HW resources consumed for SO and SM | 1,7 |

## 2.4 Conclusion

In this section, we have documented the extensions realized on the prototype implementation of the IMSaaS components. In particular, it has been presented a new orchestration system, part of the Open Baton project, which has been integrated for instantiating IMS SICs. Additionally, the OCCI-Adapter has been implemented in order to integrate this orchestrator within the MCN framework Hurtle.

A Fault Management System has been presented and employed for managing faults occurring at the virtualization layer. In particular, it has been employed for supporting high-availability at the level of the HSS SICS.

Finally, a Media Plane component has been integrated in the IMSaaS architecture. Its performances have been evaluated following the newly presented Performance Methodology of WP3. The results of such evaluation will be shown in D6.5.

# 3 DSS as a Service

This section provides the innovations addressed for the DSS service as well as required service modifications to enhance cloudified DSS, described in the previous deliverables [1][2][3][4]. DSSaaS target KPIs are also described. DSS performance evaluation results will be presented on deliverable D6.5.

## 3.1 Definition and Scope

In the previous deliverables [1][2][3][4] cloudification of DSS service and its performance evaluation have presented.

The scope of the section is to present the work carried out for DSSaaS during the extension period, including innovations implementedfor DSSaaS to enhance service provisioning, orchestration and fault management and the definition of a performance methodology evaluation aligned with WP3. KPIs measurements for this evaluation will be presented in D6.5

## 3.2 Innovations

This section explains the innovations introduced in the DSS-as-a-Service solution.

Main DSSaaS implemented innovations are:

- **Instance Live Migration:** DSS service is integrated with RAVA service described in D6.4 section 5.1[7]. Using RAVA, DSS SO is now able to make optimized resource management decisions at run time. Details of this integration are presented in following section 3.2.4.
- **Enhanced Fault Tolerance:** As explained on D5.4 [4] before, DSS service fault tolerance was limited to load balancing functionality of CMS component. The new fault tolerance approach enables load balancing not just for CMS component but also for MCR component. Moreover, for both components new mechanisms implemented to monitor the functionality of each essential service inside each DSS SIC. In case any of these services fail to respond the faulty SIC will be recognized and replaced by DSS SO at runtime. Details of this implementation are presented in following section 3.2.3.
- **Scalability and Enhanced Template Generation:** DSS dynamic template generation improved to enable horizontal scaling for DSS MCR component and runtime SIC replacement which enhances Availability, Elasticity, Fault Tolerance and performance of the service. Details of this implementation are presented in following sections 3.2.1 and 3.2.3.
- **Dynamic and Decentralized Content Replication:** DSS MCR component horizontal scaling requires a reliable file replication approach. Therefore, DSS service is modified and a dynamic P2P decentralized content replication supporting service based on BitTorrent technology implemented has been. Details of this implementation are presented in following section 3.2.2.
- **MQ based SO-SICs communication:** MCN platform service orchestrators requires a reliable way to communicate with SICs during the whole service lifecycle. Since SO component is not connected to the management network where the SICs are connected but still requires direct communication to all of them. Considered alternatives for this problem are: (1) Assigning floating IPs (FIPs) to every SIC, (2) redirect SO communication though a proxy service or (3) implement SO communication through an MQ broker. Since FIPs are a scarce resource and

exposes internal components, and now both CMS and MCR service components can scale horizontally, DSS service has been adapted to perform all SO – SIC communication though RabbitMQ[15] service. This simplifies overall communication and speeds up provisioning phase around 40% compared to the results presented in [D6.4] as it will be shown in the [D6.5] DSS performance evaluation results. That was possible due to the parallelization obtained from the asynchronous nature of the MQ messages.

### 3.2.1 Scaling Algorithms

DSSaaS dynamic scaling which is previously described in D5.4 section 3.2.1 [4] supports CMS horizontal scaling and MCR vertical scaling. With the newly modified version of DSS dynamic Template Generator, DSS MCR component is enabled with horizontal scaling feature. This horizontal scaling not only solves the downtime issue imposed to service content repository due to vertical scaling process (improves availability and fault tolerance) but also enhances the general performance and elasticity of the DSS service.

The Scaling approaches are described below:

- **Scale Out for MCR (Main Content Repository):** To achieve higher availability on DSS content repository, all instances of this component are connected to LBaaS and based on inputs received from MaaS, if decision engine realizes this component is under stress (ex. more than 60% CPU load on all MCR instances for more than 1 minute), it will provide required input to SO-D to scale out the service using SO-E.
- **Scale In for MCR:** In case the service is not under stress (ex. less than 10% CPU utilization on all MCR instances for more than 10 minutes), SO-D will take the decision of disposing the underused instances.
- **Scale Out for CMS (Content Management System):** Scale out process for CMS component is described on D5.4 section 3.2.1 [4].
- **Scale In for CMS:** Scale in process for CMS component is described on D5.4 section 3.2.1 [4].

### 3.2.2 Dynamic and Decentralized Content Replication

MCR main objective is to keep track of the static contents, namely VoD and images that are available for the players at any given time. MCR is intended to rely on an independent content delivery network (CDNaaS or ICNaaS)[7] to bring contents close to the users. However, it can also be used as the final endpoint for content retrieval in case content delivery network integration is not implemented or not available. Enabling fault management for MCR therefore requires a solid synchronisation mechanism in order to replicate the static content being managed. A common approach to this problem is to build a decentralised P2P based replication network since studies have demonstrated that the peer model increases the speed of update propagation among a set of replicas [18], decreasing the frequency of using an outdated version of the data. Even considering that the specific fault management of DSS use case considers a very small set of replicas (between 2 and 4), P2P approach have been applied to be able to handle bigger replica sets in the future.

The purpose of the implementation is to automatically replicate the contents uploaded to MCR component across all active MCR instances. All instances of MCR component are connected to LBaaS which are used by CMS component to upload contents. Each MCR instance is enabled with a tracker service [16] that provides a list of available contents for transfer, and allows it to find peers known as seeds who may transfer the contents across the network. These instances are also enabled with a customized bitorrent service [17] which is responsible for creating custom torrent files, spreading them across the network, seeding available contents on the host and download missing content from other peers (other MCR instances).

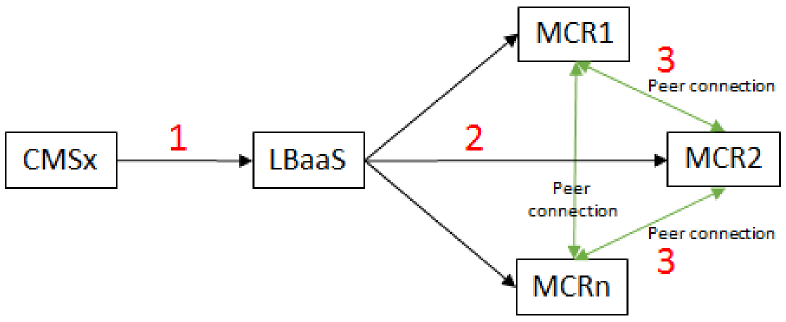

**Figure 10. Content Replication on MCR Instances**

Figure 10 shows a basic MCR content replication scenario. First, one of CMS instances requests a content upload (1). LBaaS chooses one of MCR instances according to its load balancing algorithm, in this case MCR2 (2). When the upload process is finished MCR2 starts broadcasting the existence of a new content to other peers (MCR1 to MCRn) and other peers start downloading the new content to their repositories (3).

In case of horizontal scaling, the newly deployed instances will be notified and will download all the missing content to their empty repository.

### 3.2.3 Enhanced Fault Tolerance

Enhanced DSS fault tolerance focus in enabling HA for the two main DSS components by (1) keeping at least two components of each type connected to a LBaaS, and (2) Monitoring every component at a service level, performing an automatic replacement through the SO logic in case a fault is detected in any of them.

To achieve high availability on DSS front-end, all CMS instances are connected to a LBaaS [4]. The same approach applied to MCR instances to achieve high availability on this component. MCR is now enabled with horizontal scaling feature (Section 3.2.1) and automatic content replication (Section 3.2.2).

Apart from taking advantage of load balancing benefits, at SIC level, status of all essential services running on each DSS SIC are monitored and reported to MaaS. At the SO level, modifications were made to enable DSS SO to act accordingly in the event of failure, therefore achieving enhanced Fault Tolerance on both DSS front-end (CMS) and DSS content repository (MCR). These modifications are as follows:

- **SO Monitoring Module:** Added new functionalities to manage web scenarios for monitoring web applications running on CMS and MCR SICs. Alert Engine improved to support detecting failure events of web scenarios and service specific monitoring items (i.e. monitoring item for checking Tracker service status)

- **SO Decision Engine and Dynamic Template Generator:** functionality added to template generator for removing a specific instance from service template graph. SO-D is modified to enable detecting failure events from Monitoring module and execute SIC replacement procedure.

### 3.2.4 DSS SIC Image: Custom monitoring items added to the image, to monitor service specific items for Streaming and Content Replication services.Instance Live Migration Using RAVA

DSS is integrated with RAVA management and orchestration service described in D6.4 section 5.1 [7]. For this integration MaaS agents installed on Bart testbed [7] and required monitoring items listed in D6.4, table 31 [7] are added to DSS CMS, MCR instances and bart compute nodes.

RAVA enables the cloud controller (CC) to make optimized management decisions (such as scale, migrate etc) based on the novel method of taking into consideration a correlated view of the resource usage of the different DSS SICs and then comparing it to similar correlated view obtained for other active SICs in each of the physical infrastructure nodes in order to select the best possible candidate for the DSS SIC(s) which require an integrated managed. In essence, a correlation score is computed, referred to as Reference Resource Affinity Score (RRAS), which provides a deep insight in terms of the affinity of different resource units (RU) with a reference resource unit (RRU). In order to perform resource affinity analysis, an RRU for each VNF or service component needs to be identified. RAVA will then enable the CC to determine the affinity/correlation of the RUs with the RRU and the degree of impact an RU will have on the RRU. The RAVA engine will quantify the degree of correlation of RU(s) with the RRU by computing the RRAS and based on successive values of RRAS will compute the Affinity Signature (AS). The statistical evaluation of AS will also enable the RAVA engine to predict the future trends of affinity between the RUs and the RRU and hence will enable the CC to make management decisions with long term temporal validity.

In the specific case of DSS-CMS, we are aware of the impact of the network load on the CPU utilization from our experience with the load tests on the CMS SIC. This is also depicted in Figure Figure 11 where a DSS load generator tool is used to generate load that is increased constantly over time. Results of resource consumptions are shown in Figure 12.

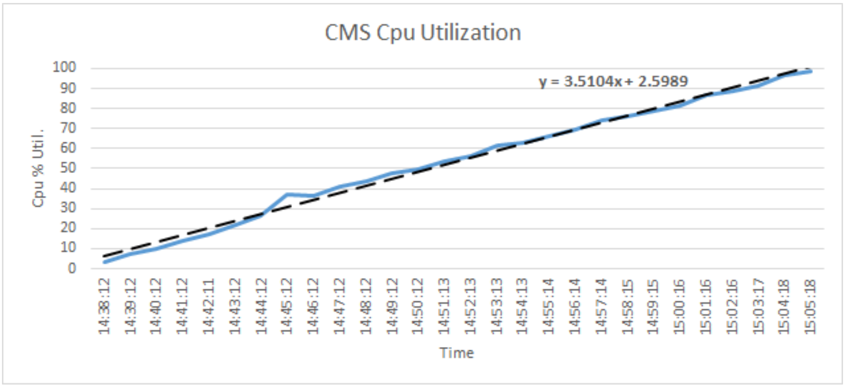

**Figure 11. CMS CPU Utilization**

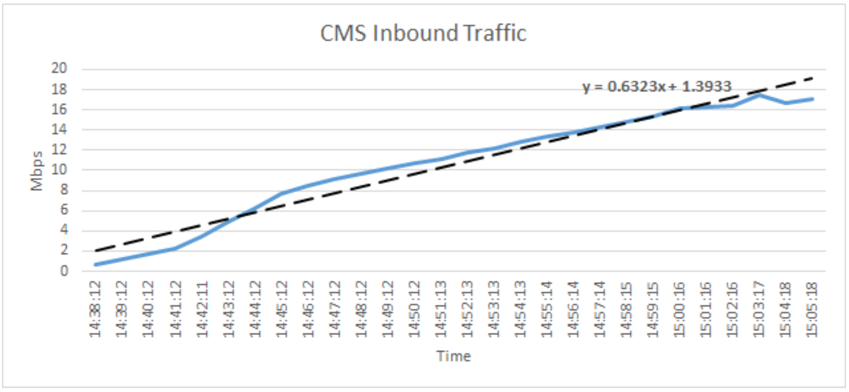

**Figure 12. CMS Inbound Traffic**

Based on the results obtained, resource limiting performance for CMS component is CPU utilization. Also strong correlation with inbound network traffic is present.

RAVA engine requires a MaaS agent configured on each of the Bart testbed nodes. Additionally, VMs where DSS SICs are running also have a MaaS agent configured to enable RAVA to perform the correlation analysis for each component. Details of initial configuration of RAVA scenario is shown in Figure 13.

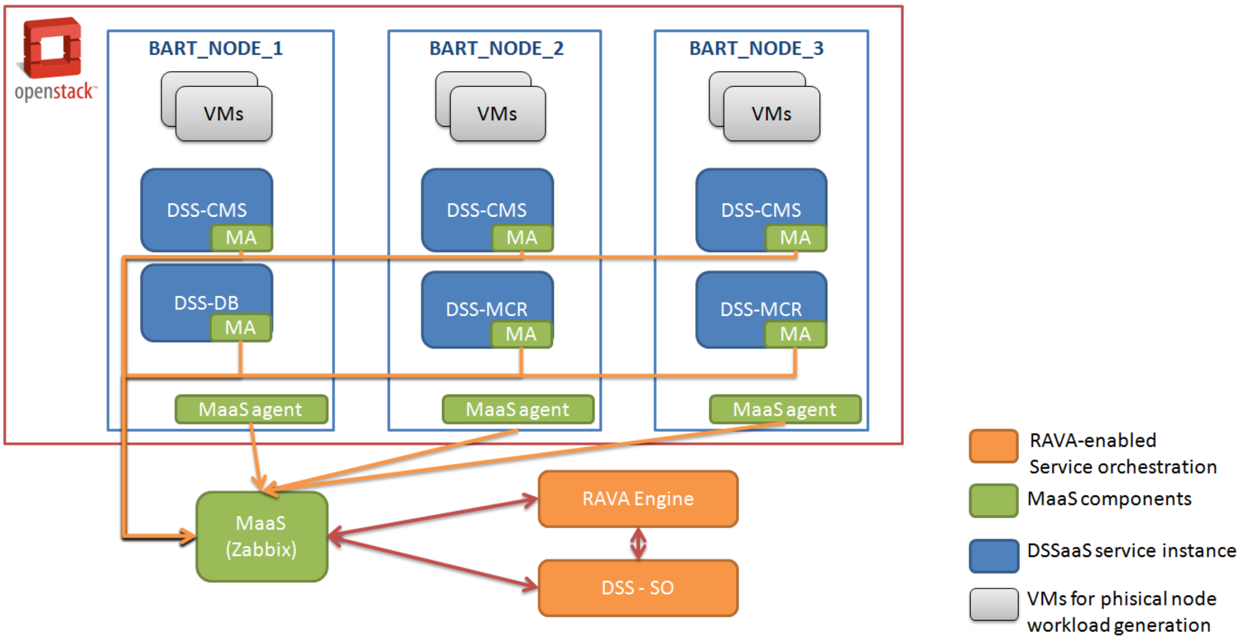

**Figure 13. RAVA initial configuration scenario**

### 3.2.4.1 DSS-RAVA Evaluation Scenario

This scenario is considering three compute nodes on Bart testbed (namely bart-node-1, bart-node-2 and bart-node-3) where a DSSaaS deployment is hosted (composed by three CMS components, two MCR components and one DB component). Beside the deployment of DSSaaS SICs, several other VMs are also deployed that will emulate SICs consuming resources based on the workload that will be generated using the "iperf" traffic generation tool [19]. DSS performance tool will also be used for external DSS SICs load generation.

Having this initial configuration, the following steps are defined for the scenario:

1. Emulating specific workload conditions on the three compute nodes by initiating load on all the six non-DSS SICs using the iperf traffic generator and stress utility program in order to stress the network and CPU RUs assigned to these respective SICs.
2. DSS performance tool start pushing load to "cms1" which is our target VM and located on "bart-node-1", increasing it by 5% every 30 seconds.
3. In parallel at the end of each RAVA Analysis Epoch, its Analytics Engine (AE) will perform the necessary statistics, calculating the RRAS and deriving the Affinity Signature (AS) of "cms1" (located on "bart-node-1") and providing the correlations of I/O utilization for the VMs with reference to the CPU on "bart-node-2" and "bart-node-3".
4. RAVA Decision Engine (DE), based on statistics received from AE and upon observing the load on "cms1", that will have exceeded a specific threshold, will select the most suitable of the two candidate servers ("bart-node-2" and "bart-node-3") and communicate its decision to the DSS-SO that will execute the decision. For example, RAVA DE based on the fine-granular correlated view of the utilization of the RUs (e.g., I/O, memory) with reference to RRU (i.e., CPU), may decide the migration of "cms-1" SIC to the selected candidate server as the best decision, which it will communicate to the DSS-SO. The DSS-SO will then take the necessary measures to execute the decision.

5. Depending on the current status of the compute nodes, DE will choose and decide between "bart-node-2" and "bart-node-3". For example, the load on "bart-node-2" might be less than "bart-node-3", however the load profile in "bart-node-2" is predicted to increase while that of "bart-node-3" is expected to remain constant. DE will thus prefer "bart-node-3" over "bart-node-2" over long term consideration and migrates "cms1" to "bart-node-3".

## 3.3 DSS Performance Evaluation

Current section aims to describe specific considerations taken into account to address WP3 performance evaluation in D3.5 for DSSaaS.

Considering that DSSaaS is an OTT system, it is important to clarify that the functional components do not provide for a separation of the control plane in the way the telco services are doing. This means that each functional component shares control and data plane functions. Therefore, although the general framework and established KPI categories still apply, a refinement of KPIs presented for performance evaluation is required in the specific case of DSS service evaluation.

In addition, a new KPI category is presented out of general context of the WP3 performance evaluation in order to measure the impact of migration of components driven by RAVA engine decision. It is important to note that RAVA decision making for live migration of components takes into account not only the evolution of the workload for each component, but also the evolution of each physical node in the testbed infrastructure, so that migration may occur unexpectedly under steady load for any of the service components. Following that premise, the extraction of metrics for KPIs defined in this section will be significant enough to have a significant perception of the overall experience of the system.

Performance evaluation methodology proposes the creation of synthetic test suites that cover in a reproducible manner a large number of possible workload use cases. Test suits proposed for DSSaaS evaluations are aligned with test kinds proposed in performance evaluation methodology, namely:

- Spike tests: a single burst of requests of a specific type is transmitted in a very short duration.
- Uniform workload tests: the same number of requests of the same type is sent every.
- Stress tests: a set of requests ever increasing is sent to the system under test in order to determine the maximum capacity of the given system.

To extract metrics for all KPIs proposed, a complete set of scenarios have been defined. The scenarios are selected to cover the widest possible situations that may occur in production environments, as specified bellow:

- Scenario 1: Service instantiation. EEU performs a DSSaaS E2E instantiation. Composed service is deployed and provisioned.
- Scenario 2: Uniform workload scenario. Service instance is running. Artificial load is generated with DSS performance tool. Load is intended to map a general production scenario with stable load with almost no usage peaks.
- Scenario 3: Spike test scenario. Service instance is running. Artificial load is generated with DSS performance tool. Load is intended to map a general production scenario with low, but unstable load. In order to extract relevant information, peaks will generate up to 4 times of the

defined base load, and will last up to 5 minutes. SICs will perform scale out and scale in operations.

- Scenario 4: Stress test. Service instance is running. Artificial load is generated with DSS performance tool. Load is intended to map a general production scenario with high stable load with almost no usage variations. SICs will perform scale out operations.
- Scenario 5: Live migration. The scenario provides a specific setup for the Stress test in order to measure specific live migration KPIs. Service instance is running with RAVA engine enabled. The scenario follows the execution script defined in section 3.2.4.1 . SICs will perform migration operations.
- Scenario 6: Service disposal. Service instance is running. EEU performs an E2E disposal request. Service is disposed.

For testing most of the scenarios listed above, an active cloud orchestration environment is required. Therefore, all measurements gathered on "Bart" testbed [7]. Although "Bart" is providing a stable, appropriate environment for running the scenarios, following limitations should be considered:

- As the testbed resources are shared between all services, scheduling of tests should be considered to make sure during execution of the tests, the testbed is not under heavy load.
- Even though All the measurements will be done using "Bart" testbed hence have complete dependency on this testbed, considerations taken into account to gather most accurate results for target KPIs. Apart from testbed isolation which mentioned above, load generators have required resources; provided test data is close enough to the real use case scenarios and warm-ups applied before actual tests. Positioning client simulators in different geographical locations is also considered if necessary.

### 3.3.1 System KPIs

System KPIs provides metrics for service quality evaluation. Considered KPIs are depicted in Table 5:

**Table 5. DSS System KPIs**

| KPIs | Considered for DSSaaS | Adaptation / Detail | Scenarios |
|---|---|---|---|
| Registration success rate | Yes | Number of players successfully registered | 2,3,4 |
| Session establishment success rate | Yes | Session establishment: Player connection request success & content response. | 2,3,4 |
| Session drop rate | Yes | Number of active player sessions | 2,3,4 |
| Attachment delay | Yes | Registration delay | 2,3,4 |
| Session establishment delay | Yes | Time required to perform a player connection | 2,3,4 |
| Data plane QoS | Yes | Groups network QoS metrics for different kinds of traffic considered for DSS | 2,3,4 |

| | | | |
|---|---|---|---|
| Data plane delay | Yes | network delay for different kinds of traffic considered for DSS | 2,3,4 |

In order to measure system KPIs, DSS is previously deployed and provisioned. As explained before, some of the KPIs have been adapted to the specific functional constraints of DSS service. For simplicity reasons, measurements for system KPIs will not consider any lifecycle or cloud-based operation (deployment, provisioning, update, scale, migration, disposal...) since those will be considered in the following subsections.

Some of the results have already been collected in D5.4 [4] and D6.4 [7] including benchmarking results between physical and KVM hypervisor. Previous tests showed values fewer than 10% performance loss on average caused by virtualization overhead after a fine tuning of the hypervisor parameters.

Remaining KPIs measurements will be collected and presented in D6.5.

### 3.3.2 Life-Cycle KPIs

Life cycle KPIs aims to measure the behaviour of a DSSaaS instance focusing on deployment, provisioning, disposal and installation times. Considered KPIs are depicted in Table 6:

**Table 6. Life-Cycle KPIs**

| Lifecycle KPIs | Considered for DSSaaS | Adaptation / Detail | Scenarios |
|---|---|---|---|
| Installation duration | Yes | Time to push new SIC images to the platform | - |
| Deployment and configuration duration | Yes | Time to deploy and provision a service instance | 1 |
| Disposal duration | Yes | Time to dispose a service instance | 6 |
| Service upgrade duration | Yes | Overall upgrade time. Composition of the previous KPIs | 1 |

Scenarios required to perform measurements of KPIs lifecycles are scenario 1 and scenario 7, which performs a complete service instantiation and disposal. For lifecycle KPIs, an E2E service instance deployment (service composition) is considered. Although some of these metrics have been previously collected and presented in D6.4 [7], the complete set of KPIs is presented here to keep performance methodology evaluation consistency. Installation duration KPIs will be obtained considering a manual update of the DSS SICs images to the underlying testbed.

### 3.3.3 Cloud-Native KPIs

Cloud native KPIs aims to measure the behaviour of a DSSaaS instance focusing on automated cloud operations to achieve service elasticity and fault tolerance. Considered KPIs are depicted in Table 7:

**Table 7. Cloud-Native KPIs**

| Cloud native KPIs | Considered for DSSaaS | Adaptation / Detail | Scenarios |
|---|---|---|---|
| Single component deployment latency | Yes | Time to deploy each of the SICs composing service | 1 |
| Scale in/out management latency (ctrl plane) | Yes | Scale in/out perceived user time for each SIC | 2,3,4 |
| Scale in/out operation number (ctrl plane) | Yes | Number of scale in/out operations per SIC | 2,3,4 |
| Useless scale operation number (ctrl plane) | Yes | Number of ping-pong scaling ops | 2,3,4 |
| Scale in/out management latency (data plane) | No | Does not apply (OTT does not perform data/ctrl plane isolation at component level) | |
| Scale in/out operation number (data plane) | No | Does not apply (OTT does not perform data/ctrl plane isolation at component level) | |
| Useless scale operation number (data plane) | No | Does not apply (OTT does not perform data/ctrl plane isolation at component level) | |
| Time percentage of control plane availability | Yes | Availability after initial provisioning performing scaling ops / component failure | 2,3,4,5 |
| Time percentage of data plane availability | No | Does not apply (OTT does not perform data/ctrl plane isolation at component level) | |
| Overall reconfiguration latency | No | Does not apply (DSS does not consider automatic reconfiguration) for manual measurements, service upgrade duration KPI applies | |

WP3 Performance methodology establishes a separation between control and data plane KPIs that cannot be directly applied to an OTT service. Therefore, some of the KPIs in Table 7 are not considered.

These KPIs seek to measure the possible penalties or sudden drops in system performance when scaling operations take place, and how they affect overall service availability. Some of these metrics have been reflected in D6.4 [7] but have been kept for consistency.

### 3.3.4 Extended Cloud-Native KPIs

Extended cloud native KPIs aims to measure the behaviour of a DSSaaS instance focusing on live migration of individual service components performed by RAVA management/orchestration migration operations and fault management described in the innovation section. Considered KPIs are depicted in Table 8:

Table 8. Extended Cloud-Native KPIs

| Extended Cloud-Native KPIs | Considered for DSSaaS | Adaptation / Detail | Scenarios |
|---|---|---|---|
| Service component migration latency | Yes | Time required to complete a SIC migration between 2 physical nodes | 5 |
| Service component migration operations number | Yes | Number of migration operation performed during the scenario | 5 |
| Migration drop rate | Yes | Drop rate caused at service level by a migration operation | 5 |
| Fault SIC recovery drop rate | Yes | Drop rate caused at service level by a component/SIC failure | 2 |
| Fault SIC recovery latency | Yes | Time required to recover from a fault/error in any DSS SIC | 2 |

Although live migration related KPIs have not been explicitly included in performance evaluation methodology, the evaluation of the service behaviour in case of live migration operations is required, to understand how service performance is affected during a migration operation.

KPIs are extracted from the evaluation of the RAVA scenario defined in section 3.2.4.1

### 3.3.5 Resources KPIs

Resources KPIs aims to measure the behaviour of a DSSaaS instance focusing on hardware resource consumption of individual service components. Considered KPIs are depicted in Table 9:

Table 9. Resource KPIs

| Resources KPIs | Considered for DSSaaS | Adaptation/Detail | Scenarios |
|---|---|---|---|
| CPU usage | Yes | CPU utilization per SIC | 2,3,4 |
| Memory usage | Yes | Memory usage per SIC | 2,3,4 |
| Network usage | Yes | Inbound/outbound network usage per SIC | 2,3,4 |
| External network usage | Yes | Inbound/outbound network usage between testbed and the external Clients (DSS players) | 2,3,4 |
| Storage usage | Yes | Storage utilization per SIC | 2,3,4 |
| Number of VMs used | Yes | Number of VMs used (DSS composed services) | 2,3,4 |
| Resources consumed for orchestration | Yes | HW resources consumed for SO-D/SO-E and RAVA AE | 1,2,3,4,5 |

Resources KPIs provinces a clear view about the resources provisioned and consumed by the DSS service, depending on the load generated.

## 3.4 Conclusion

The section covers the design and implementation of innovations applied to DSSaaS service based on the information presented in the D5.4 [4]. The aim of these innovations is to further optimize cloudification of DSSaaS. Innovations cover enhanced elasticity and fault management mechanisms implemented for all DSS service components, namely the improvement in the SO-D module logic to cover the horizontal scalability and replacement in case of failure and the implementation of automatic replication of multimedia contents between MCR components based on bit torrent technology.

Additionally, section explains how RAVA analytics engine (AE) integration has been addressed to enable automatic migration of active service components between different testbed nodes, based on the analysis of linear regression of consumption of resources of virtual machines that are part of the service instance, compared to consumption of physical hosts (nodes). Following a similar approach other services provided by MCN can be adapted to have automatic migration enabled.

A complete evaluation plan to measure service KPIs based on the inputs of D3.5 is defined and evaluation results will be published in D6.5, considering the limitations and specific adaptations for DSS service and extending them to cover the live migration enabled by RAVA.

# 4 Conclusion

The WP5 deliverable presented work accomplished during MCN project extension, describing significant advancements for IMS and DSS services compared to the work addressed during three years' period, highlighting the following:

- Integration of an ETSI NFV compliant orchestrator within the MCN platform
- Integration of a Media Plane Component in the IMSaaS architecture
- Integration of live migration decision logic to optimize utilization of the underlying software infrastructure (OpenStack)
- Improvement of Fault Management mechanisms. Requiring some service redesigning and the implementation of specific innovation like dynamic content replication algorithms.

Beside of this, specific adaptations to the WP3 performance methodology evaluation are presented for both IMS and DSS services. The methodology followed extends and presents in an organized way the KPIs that have been considered more relevant for the service evaluation within MCN platform.

## References

[1] **D5.1 MCN-Project** Design of Mobile Platform Architecture and Services [Report]. - [s.l.] : European Commission, EU FP7 Mobile Cloud Networking public deliverable, 2013.

[2] **D5.2 MCN-Project** Implementation of IMSaaS, DSN and Mobile Platform [Report]. - [s.l.] : European Commission, EU FP7 Mobile Cloud Networking public deliverable, 2013.

[3] **D5.3 MCN-Project** Final implementation of IMSaaS, DSN and Mobile Platform [Report]. - [s.l.] : European Commission, EU FP7 Mobile Cloud Networking public deliverable, 2013.

[4] **D5.4 MCN-Project** Evaluation of Mobile Platform, IMSaaS and DSN [Report]. - [s.l.] : European Commission, EU FP7 Mobile Cloud Networking public deliverable, 2015.

[5] **D3.4 MCN Project** Infrastructure Management Foundations – Final Report on component design and implementation [Informe]. - [s.l.] : European Commission, EU FP7 Mobile Cloud Networking public deliverable, 2015.

[6] **D6.3 MCN-Project** Final Report on Integration and Evaluation Plans [Report]. - [s.l.] : European Commission, EU FP7 Mobile Cloud Networking public deliverable, 2014.

[7] **D6.4 MCN-Project** Final Report on Testbeds, Experimentation, and Evaluation [Report]. - [s.l.] : European Commission, EU FP7 Mobile Cloud Networking public deliverable, 2015.

[8] ETSI Network Functions Virtualisation (NFV). [Online]. www.etsi.org

[9] 3GPP. TS 23.228. IP Multimedia Subsystems (IMS).

[10] Carella, G.; Corici, M.; Crosta, P.; Comi, P.; Bohnert, T.M.; Corici, A.A.; Vingarzan, D.; Magedanz, T., "Cloudified IP Multimedia Subsystem (IMS) for Network Function Virtualization (NFV)-based architectures," *Computers and Communication (ISCC), 2014 IEEE Symposium on* , vol.Workshops, no., pp.1,6, 23-26 June 2014.

[11] Open Source IMS Core: http://www.openimscore.org/

[12] Zabbix: http://www.zabbix.com

[13] Kamailio: http://www.kamailio.org

[14] SIPp: http://www.sipp.sourceforge.net/ims_bench

[15] RabbitMQ: https://www.rabbitmq.com/

[16] OpenTracker: http://erdgeist.org/arts/software/opentracker/

[17] Libtorrent: https://github.com/arvidn/libtorrent

[18] Hegde S., “Replication in Distributed File Systems”, Department of Computer Science, University of Texas at Arlington, 06/2011 at http://crystal.uta.edu/~kumar/cse6306/papers/Smita_RepDFS.pdf.

[19] Iperf: https://iperf.fr/