Design and Implementation of a High Performant PaaS Platform for Creating Novel Real-Time Communication Paradigms

Cloud computing for Real-Time Multimedia Communication

Alice Cheambe[1], Flavio Murgia, Maiorano Picone, Pasquale[1], García, Boni[2], Gallego, Micael[2], Giuseppe Antonio Carella[3], Lorenzo Tomasini[3], Alin Calinciuc[4], Cristian Spoiala[4]

[1:] {alice.cheambe, flavio.murgia, pasquale.maiorano.picone}@fokus.fraunhofer.de, Fraunhofer FOKUS, Berlin, Germany

[2:] {boni.garcia, micael.gallego }@urjc.es, Universidad Rey Juan Carlos, Madrid, Spain

[3:] {giuseppe.a.carella, lorenzo.tomasini}@tu-berlin.de, Technische Universität Berlin, Berlin, Germany

[4:] :{alin.calinciuc, cristian.spoiala}@assist.ro,Stefan cel Mare University of Suceava, Suceava, Romania

***Abstract*— this paper presents the design and implementation of a Real Time Communication and multimedia processing architecture that uses emerging Network Function Virtualization (NFV) and Software Defined Networks (SDN) to provide enabling cloud technologies. This is work done within the EU project NUBOMEDIA. The main objective of the NUBOMEDIA project is to address the complexity usually involved in providing such a platform, thereby providing a single platform for end-to-end service provisioning, deployment and availability of services. To validate the platform, within the project use case implementations from eHealth, IPTV, augmented reality and collaborative e-Learning are being developed and tested. For such services, a Platform-as-a-Service (PaaS) strategy is proposed which hides the complexity of the infrastructure thereby abstracting services for provisioning, scaling, QoS and network management. This paper highlights the NUBOMEDIA architecture and describe the application deployment procedure for developers.**

Keywords—Cloud computing, Real Time Communication, Quality of Servcie, webRTC, Platform as a Service, Network Funciton Virtualization, Software Defined Networks

I. INTRODUCTION

Traditional communication services act just as mere information transport systems. For example, when we perform a phone call, we just receive the speech signal coming from the person at the other end of the line. However, new techniques such as computer vision, augmented reality and the advanced processing of audio-visual flows allow creating novel communication paradigms useful in diverse areas such as e-Health, security, defense, video games or smart TV.

Web Real-Time Communications (WebRTC) is the umbrella term for several emergent technologies and APIs that aim to bring such communications to the Web. The standardization activity for WebRTC is split between the World Wide Web Consortium[1] (W3C) and the Internet Engineering Task Force[2] (IETF). On the one hand, W3C is defining the JavaScript APIs (Application Programming Interfaces) and the standard HTML5 tags to enable real-time media capabilities to browsers. On the other hand, IETF is defining the underlying communication protocols (SRTP, SDP, ICE, and so on) for the setup and management of a reliable communication channel between browsers [2].

Another advancement in the Web domain is the emergence of cloud computing. Cloud computing is a form of computing in which dynamically scalable and often virtualized resources are provided as a service over the Internet. Platform-as-a-Service (PaaS) is a model of cloud computing that alludes to high-level software systems delivered over the Internet.

NUBOMEDIA is a cloud infrastructure providing developers the ability to deploy real-time media applications that scale transparently and elastically adapting to end-users offered load. The architecture is driven by real-time applications from business sectors including augmented reality, computer visions, multisensory multimedia and e-Health, each providing QoS requirements in respect to the on-demand provisioning of virtualized infrastructure resources.

II. RELATED WORKS AND CONTRIBUTIONS

In academia, a few efforts have been made in the cloudification of Real Time multimedia infrastructures. Boniface et al. [8] propose a PaaS architecture for provisioning real-time service-oriented applications in clouds. These authors address two key aspects of PaaS namely service engineering and service management, showing how the combination of methods, tools and services can be used to improve the usability, maintainability, efficiency of services targeting clouds with strict QoS constraint. Their approach is similar to NUBOMEDIA approach in aspects of 1) QoS specification at application and infrastructure layers, 2) event prediction; QoS oriented service engineering models for predicting QoS requirements, 3) on demand resource provisioning;

[1] http://www.w3.org/TR/webrtc/

[2] http://tools.ietf.org/wg/rtcweb/

provisioning of network resources on virtualized infrastructures throw a combination of workflows and service based metrics. Jung et al. [9] present a PaaS Environment for Social Multimedia Services (PESMS) for supporting the development of social networking services that include multimedia formats (audio, video, image). Their focus was more on encoding and transcoding functionalities for processing large amounts of social media in a parallel and distributed fashion. Difference in their approach and NUBOMEDIA approach is that, they don't provide a full platform for the deployment of applications but offer a platform library for uploading and processing multimedia content which is, after processed, returned to the specific service (user). This approach falls more in the category of SaaS.

The NUBOMEDIA architecture provides a PaaS platform that enabling infrastructural services to developers for managing their applications. The platform also offers cloudified Media Pipelines which offer media specific APIs to define media processing topologies such as augment reality filters and visual computing toolkits for seamless integration to developers.

III. NUBOMEDIA CLOUD ARCHITECTURE FOR REAL-TIME INTERACTVICE MULTIMEDIA APPLICATIONS

NUBOMEDIA cloud platform is a PaaS Host and PaaS Provider offering PaaS Users the functionalities to deploy real-time multimedia applications that can scale transparently and elastically adapting to the end user's offered load. Fig. 1 illustrates the architecture of the NUBOMEDIA cloud platform.

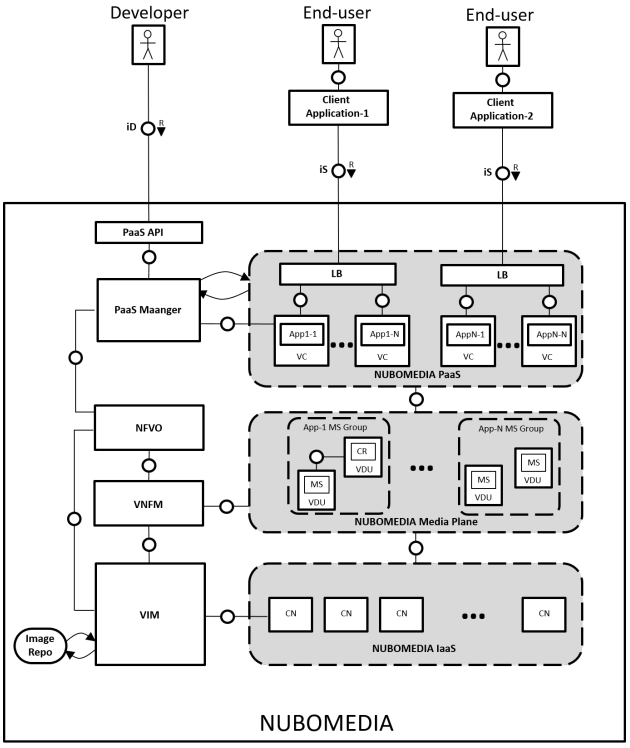

Fig. 1. NUBOMEDIA Cloud Platform

The platform provides two interfaces to two types of end users. 1) Developer (PaaS User) use the Developers interface (iD) to deploy multimedia applications on the NUBOMEDIA PaaS. 2) End Users (service consumers) use the Service interface (iS) to consume the services of one or several deployed applications. The rest of this section will provide a very high level description of the various parts of the System.

A. PaaS API and PaaS Manager

The PaaS API are RESTful APIs accessible via HTTP(s) on the PaaS Manager server. These REST APIs exposes abstractions over specific operations to developers that allows them deploy and manage their applications, on the NUBOMEDIA PaaS. Table 1 gives an overview of the exposed REST interfaces.

TABLE I. NUBOMEDIA PAAS REST API

Method	URL	Description
POST	api/v1/nubomedia/paas/users	Create a new PaaS User
DELETE	/api/v1/nubomedia/paas/users/{name}	Deletes a PaaS User
POST	/api/v1/nubomedia/paas/app	Create a new application
POST	/api/v1/nubomedia/paas/app/{id}	Retrieve status of a project (App) with the given id.
GET	/api/v1/nubomedia/paas/app	Return the list of all deployed projects
DELETE	/api/v1/nubomedia/paas/app/{id}	Delete project with the given id
POST	/api/v1/nubomedia/paas/secret	Create a secret[a]
DELETE	/api/v1/nubomedia/paas/secret/{name}	Deletes a secret[a]
POST	/api/v1/nubomedia/paas//oauth/authorize	Authenticate a PaaS User

a. The Secret object type provides a mechanism to hold sensitive information such as passwords, client config files, dockercfg files, private source repository credentials, etc.

The PaaS Manager Module abstracts operations for developers with which they can build, deploy and instantiate resources on the NUBOMEDIA cloud subsystem. The main functionalities provided by the PaaS Manager are: 1) Building and deployment of applications on NUBOMEDIA PaaS and 2) Requesting the instantiation of network virtual resources for the application to be deployed

As depicted in Fig. 2 the PaaS Manager is composed of six main components.

- API: is the component exposing the Pass-API interface consumed by the Developer

- PaaS Manager: is the central application logic module for holding everything together. It receives requests via the PaaS API and forwards this requests to the appropriate internal service and subsequently returns responses.
- Repository: interoperate with the persistent application storage data with CRUD operations. Application metadata such as application identifier, name, network service record identifier of deployed applications are stored here.
- NFVO Connector: provides services that consume the exposed NFVO-API for requesting the instantiation of virtual network resources
- PaaS Connector: provides services that consume the exposed REST-API from the NUOMEDIA PaaS on operations for security, user management, application deployments, image and source builds, HTTP(s) routing, and project management.

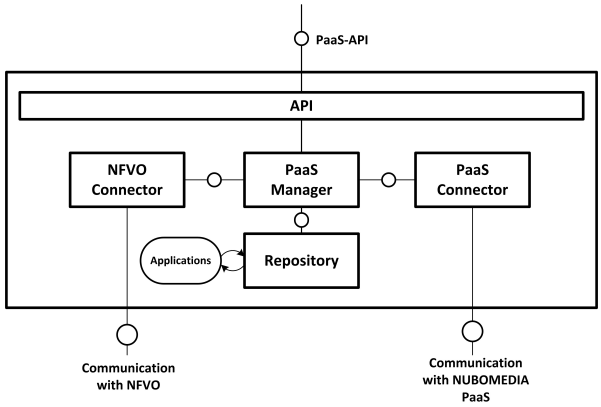

Fig. 2. PaaS API and PaaS Manager

B. NUBOMEDIA PaaS

NUBOMEDIA PaaS allows developers to host and scale their applications in a cloud environment. As depicted in Figure 3 the NUBOMEDIA PaaS comprises the following building blocks: APIs, core services, containers, nodes and storage.

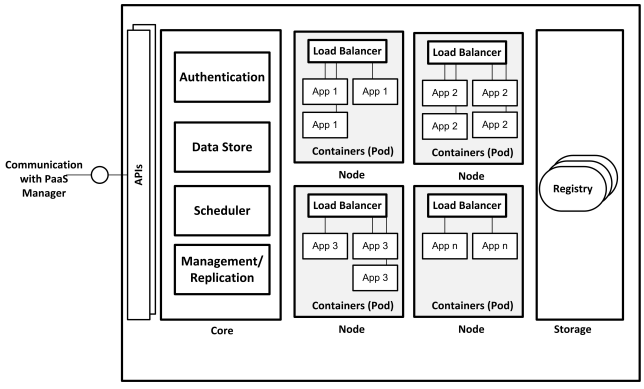

Fig. 3. Functional arcitecture of NUBOMEDIA PaaS

APIs: The core services of NUBOMEDIA PaaS are provided as micro-services. Micro-service is a software architecture approach in which complex applications are composed of small, independent processes communicating with each other using language-agnostic APIs. The core components are offered as micro-services which interact with each other through common REST APIs to enhance the system. This provides the flexibility to allow for alternative implementations

Core services: these include services such as authentication for security of the system; Data Store for keeping persistent metadata of PaaS use's applications available on given nodes and their capabilities; Scheduler service for determining placement of new containers on nodes; Management and replication services to ensure that a specific number of containers ("replicas") are running at all times.

Containers: are the basic units of the NUBOMEDIA PaaS applications which uses the concept of Linux Container Technology. This is basically a lightweight mechanism for isolating running processes, such that they are limited to interacting with only designated resources (processes, files, network, database, etc.). NUBOMEDIA PaaS uses Docker containers which are based on Docker images. A Docker image is a binary that includes all of the requirements for running a single Docker container, as well as metadata describing its needs and capabilities.

Node: is a worker machine which may be a Virtual Machine (VM) or physical machine. Each node has the services necessary to run Pods (a group of containers that make up the application) and is managed by the core components. The node also provides network proxy and load balancing services for scaling the traffic it receives amongst all replicas.

Registry: is used to store custom Docker images for building and deploying user's applications.

For this project, we decided to use the open source version of Red Hat's OpenShift called OpenShift Origin v3 [7]. This is the community driven implementation of the PaaS which introduces the use of Docker [4] and Kubernetes [5]. Docker is an open platform for developers and sysadmins to build, ship, and run distributed applications, whether on laptops, data center Virtual Machines (VMs), or on the cloud. OpenShift uses it as containers for running applications. Kubernetes is an open source orchestration system for Docker containers developed by Google. OpenShift uses Kubernetes for scheduling and packaging Docker containers onto nodes in a compute cluster and manages it workloads to ensure that their state machines reflects the users declared intentions.

In addition to the advantages offered by Docker and Kubernetes, there are several reasons, why we decided for this software component:

- Provision of over the top functionalities for augmented deployment, orchestration and routing.

- Provision of groups of Docker containers called pods that simulate a single virtual machine with a single IP address, shared file system and common security settings. This ability provides the advantage to deploy scalable applications with shared local resources.
- Extensibility. Through the REST APIs to core components services, alternative implementations are applicable.
- Flexibly linked feature; Other PaaS platforms are limited to only web frameworks and rely on external services for other component types. OpenShift Origin v3 provides more application topologies. PaaS users can have a project in which many components are linked with each other. In this manner, we can link any two arbitrary components together through exporting and consumption of environment variables. This way, we can link together any two components without having to change the images they are based on.

Given the above advantages, we can build anything on OpenShift by offering a platform built on containers that allows building entire applications in a repeatable lifecycle.

C. NUBOMEDIA Media Plane

The NUBOMEDIA media plane offers SaaS and is based on a piece of software called the Kurento Media Server (KMS) [6] that provides pluggable media processing capabilities. These capabilities are exposed to application developers through abstractions called *media elements* which expose features that let applications record, mix, augment, blend, route, and apply computer vision to streams, for example. Media elements are functional units performing a specific action on a media stream. The Media elements are represented as self-contained "black boxes" to the application developers, who do not need to understand the low-level details of the element for utilizing it [3].

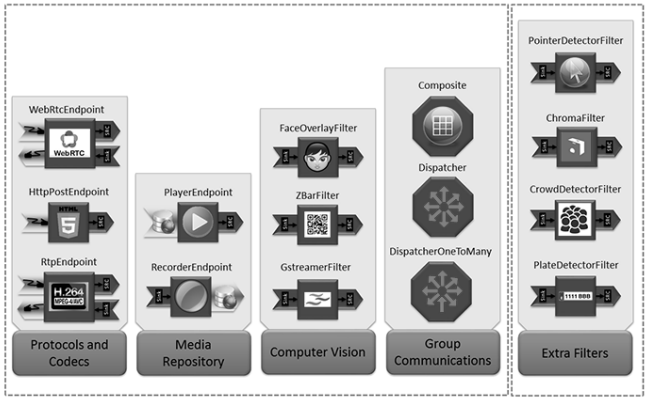

Fig. 4. NUBOMEDIA Media Elements Toolbox

Media elements are capable of receiving media from other elements and of sending media to other elements. This way it forms a media pipeline. The *media pipeline* is a chain of media elements, where the output stream generated by one element (source) is fed into one or more other elements input streams (sinks). Hence, the pipeline represents a "machine" capable of performing a sequence of operations over a stream. Fig. 5 shows the pipeline of an application implementing a WebRTC loopback, which receives and sends WebRTC streams (*WebRtcEndpoint*), processes the media flow for tracking an object's movements (*PointerDetectorFilter*), and adds a special effect, such as a hat, on top of the detected faces (*FaceOverlayFilter*) when the object enters a specific region.

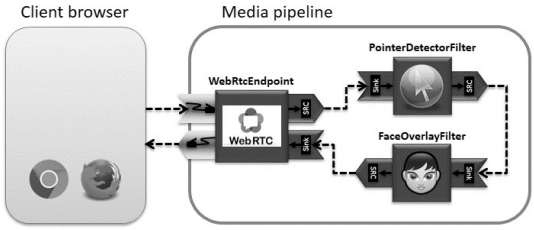

Fig. 5. *NUBOMEDIA application using computer vision and augmented reality*

In order to guarantee QoS to the end users, NUBOMEDIA PaaS elastically provides virtualized infrastructure resources. For instance, computer vision and augmented reality filter are highly CPU-consuming. Due to the fact that media pipelines are executed in a single instance of KMS, if these filters are used in an application with many concurrent users, NUBOMEDIA provides more instances of KMSs. To that aim, NUBOMEDIA provides a special kind of media element that enables the streaming of media among different media pipelines, and therefore the load can be distributed into different KMS instances. Fig 6 picture shows an example how different media pipelines interconnected.

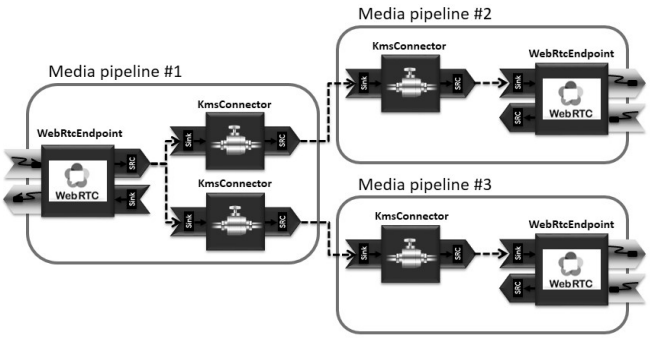

Fig. 6. Example of distributed media pipelines

D. NUBOMEDIA Controllers (NFVO and VNFM)

The NUBOMEDIA Controller is the layer composed by all the control functions providing management and orchestration functionalities of virtual resources and media components. This layer is responsible of managing the lifecycle of the Media Plane elements. Considering that each of the Applications will require its own Media resources, it is of particular importance to provide isolation among those Media components. This full isolation is achieved providing multi-tenancy, and particularly

the possibility of instantiating in parallel multiple set of Media Plane elements.

The NUBOMEDIA Controller layer is composed by three functional elements: a Network Function Virtualization Orchestrator (NFVO), a set of Virtual Network Function Managers (VNFM), and a Virtualized Infrastructure Manager (VIM).

The NFVO manages the lifecycle of a Network Service. In the ETSI [12] terminology a Network Service is a set of multiple Virtual Network Functions. In NUBOMEDIA a Network Service is composed by a Media element and a Cloud repository. It exposes an interface to the PaaS API providing the functionalities for instantiating and disposing a Network Service. The main functionalities provided by the NFVO are:

- Instantiate virtual resources (via the VIM interface) required by the Network Service as specified by the PaaS API
- Provide guaranteed networking resources interoperating with the IaaS network elements
- Send lifecycle events to the different VNFMs

The VNFM, as specified by ETSI NFV [12], is a component providing lifecycle management of a specific VNF. In NUBOMEDIA there are two types of VNFMs: one for managing the Media components, and one for the Cloud Repository. Both are exposing the same type of interface to the NFVO. The communication among them is achieved using a PUB/SUB mechanism over a message queue.

The main function of the VNFM is to deploy a specific VNF on top of virtual resources allocated on the IaaS. This is achieved by triggering the execution of certain scripts (as defined in the lifecycle events of the Network Service) on the virtual resources where those elements have to be installed. Additionally, for the Media components, the VNFM has been extended with another functional elements, the Elastic Media Manager (EMM). This element exposes an interface to the Application providing the capabilities of reserving the required resources on a Media component. In particular, it makes sure that there are always the required resources for the Applications, adapting the utilized virtual resources to the workload changes. This, so called auto scaling mechanism, provides two main operations, scaling in and scaling out, to horizontally scale a set of Media components. Scaling in means removing virtual resources when they are no longer required. Scaling out means adding additional resources when they are required.

The VIM is the component managing the lifecycle of virtual resources. It offers an interface to the NFVO providing mechanisms for deploying and provisioning compute, network and storage resources. It interacts with the IaaS layer abstracting the Hardware which it provides

E. NUBOMEDIA IaaS

The IaaS is composed of hardware and software resources providing an environment on top of which cloud services can be executed. NUBOMEDIA IaaS hides the underlying complexity of the IaaS platform and enables users to build their applications without knowledge on the infrastructure. To achieve this, NUBOMEDIA IaaS is based on OpenStack [15], an open source platform for creating private and public clouds. OpenStack capabilities are exposed through a REST API that is used by Virtual Infrastructure.

In addition to computing, storage and networking capacity, the testbed provides centralized metrics, logs collecting and monitoring system for all the architecture components

The elastic nature of the IaaS enables developers to build multimedia apps that scale easily with an increase of usage. With an increase in usage, the Load Balancer from NUBOMEDIA IaaS starts new Compute Nodes (CNs) with the VIM and in few minutes using KVM virtual machines they will be up and running. By using Docker containers, new CNs can be online in few seconds.

IV. DEPLOYMENT AND LIFE CYCLE MANAGEMNT OF REAL-TIME APPLICATIONS

Fig. 7 illustrates the work flow of a user deploying an application on the NUBOMEDIA cloud infrastructure. The application will be built in the NUBOMEDIA PaaS platform, but it requires external resources provided by NUBOMEDIA IaaS. Prior to deploying an application, the user needs to off course be authenticated on the PaaS manager. After authentication, the user is presented a GUI from which they can enter application specific information necessary for the deployment (step 1 from Fig. 7). This create and send a JSON object which is sent to the PaaS Manager via HTTP POST Request.

```
{
"gitURL":"https://github.com/example/example
-app.git",
   "appName":"my-app-name",
   "projectName":"my-project",
   "ports":[
        {
           "targetPort": 8080(for example),
           "port":8088 (for example),
           "protocol": "TCP" (for example)
        }
   ],
   "flavor":"m1.medium",
   "replicasNumber":2,
   "secretName":"previously-defined-secret-
name"
}
```

- gitURL: is the gir repository where the jar and Dockerfile (and even other files that are necessary for the application)
- appName: the application name that will be used also to create the DNS entry to use your application
- projectName: the project name, has to be the same that was used for secret creation;

- ports: an object that maps the ports that are used from container to the ports that has to be exposed to the outside, with the respective protocol
- flavor: the correspondent media server flavor that will instantiate on the infrastructure
- replicasNumber: the number of containers that has to be created by the PaaS after the building phase
- secretName (optional): the name of the secret that has to be used only if your application is on a private git repository

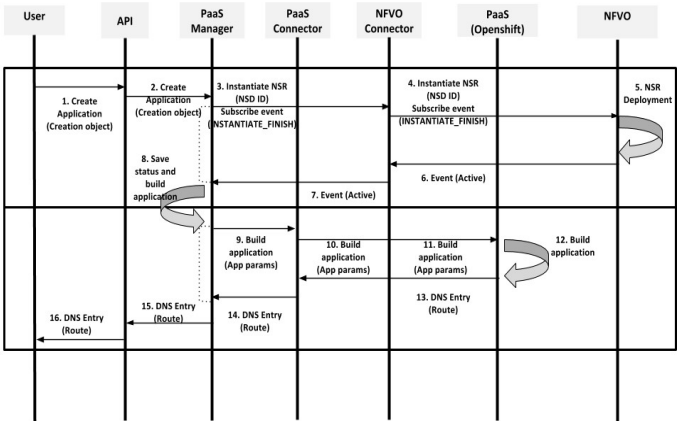

Fig. 7. *Deploying Applications on NUBOMEDIA PaaS API*

The API forwards this request to the PaaS Manager, which internally dispatches the request to the NFVO connector (step 2 and 3). The NFVO Connector consumes the exposed interface from the NFVO component for registering new applications and deploying the necessary media components. After this has been achieved on the NFVO, it sends a response (step 6) back to the NFVO component, which is propagated back to the PaaS Manager. For successful response from the NFVO component, the PaaS Manager sends the build and deploy request to the PaaS (Openshift) for retrieving the application and Docker file from the developer's repository and deploying it on the PaaS (steps9-12). For successful deployment, a DNS entry pointing to the route on which the application can be reached from the Internet is sent back to the PaaS Manager (step 10) which then propagates that back the user.

After the build and deploy procedure has started, depending on the duration, the PaaS User can query the build status (HTTP GET on `/api/v1/nubomedia/paas/app/{appID}`) and retrieve building status from the NUBOMEDIA PaaS build logs, using application identifier and authorization token. If the request is performed without any ID the API will return the status of all applications for that user.

The PaaS User could decide to remove its application from NUBOMEDIA PaaS by sending a HTTP DELETE request on `/api/v1/nubomedia/paas/app/{appID}`. This request remove all the configuration files and pods (group of Docker containers which make up the application) from the NUBOMEDIA PaaS and also request the deletion of the instantiated Media Server instances by the NFVO/VNFM component. It is also possible to delete previously created Secret objects using the HTTP DELETE on `/api/v1/nubomedia/paas/secret/{secretName}` path.

V. Evaluation

The first half of the NUBOMEDIA project was dedicated on specify, designing and implementing the architecture. We are currently on the second half of the project were we are testing the connection and work flow of all the components involved. However some early evaluation of the time needed to deploy a new application and a supporting network topology has been made. The deployment time of 10 different applications was measured and evaluated. Fig 8 and 9 show the deployment of these applications with and without the PaaS API actively checking for status of the deployment. All values are in milliseconds. From the graphs it can be observed that, the maximum time during the deployment is needed in the creation of the Media Server by the NFVO component.

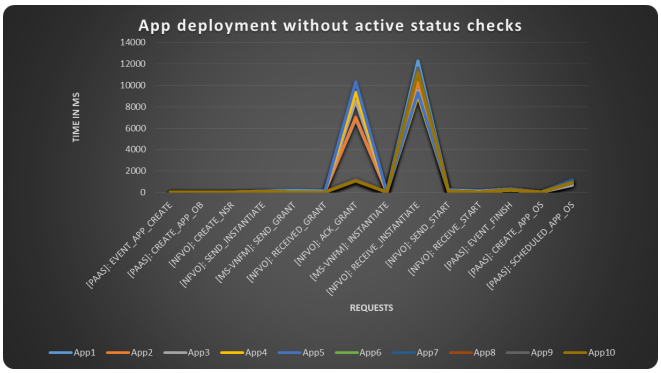

Fig. 8. *Deploying 10 Applications on NUBOMEDIA*

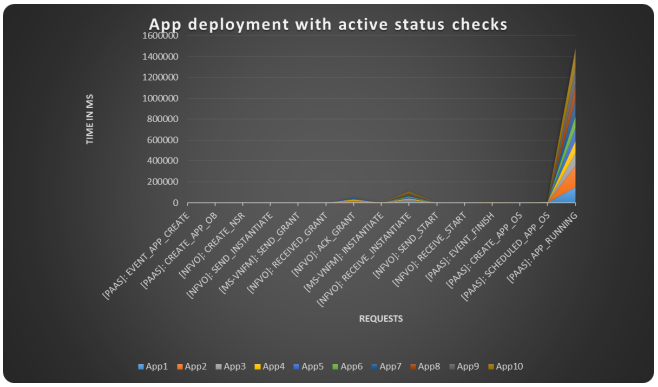

Fig. 9. *Deploying 10 Applications on NUBOMEDIA with active status checks from the PaaS API*

VI. Conclusion and Outlook

This paper has described the design and implementation a PaaS platform for enabling the creation of creating novel real-time communication applications in clouds. The paper presented the NUBOMEDIA architecture, which aligns to the NVF and SDN specification for enabling cloud functionalities.

We also presented how the platform can be used by developers for deploying applications on the cloud and early evaluation of deployment of applications was also presented. These results are intermediate results from the first phase of the project. The next half of the project is dedicated to developing demonstrators to validate the platform. The next phase planned for the NUBOMEDIA project is in the implementation of defined use cases from the domains of augmented reality, computer visions, multisensory multimedia and e-Health, which will then be deploy on the NUBOMEDIA cloud infrastructure for validation. This will help use conduct a series of performance and scalability tests for the following:

- creation of efficient scheduling and placement algorithms for media sessions
- development of auto-scaling mechanisms based on policies
- evaluation of load balancing and scalability of applications on the NUBOMEDIA PaaS

ACKNOWLEDGMENTS

This work has been supported by the European Commission under projects FI-WARE FP7-2011-ICT-FI, GA-285248, and NUBOMEDIA FP7-ICT-2013-1.6, GA-610576; by Spanish Ministerio de Educación under project Reactiv Media (TIN2013-41819-R); and by the Regional Government of Madrid (CM) under project Cloud4BigData (S2013/ICE-2894) co funded by FSE & FEDER.

REFERENCES

[1] NUBOMEDIA Project, http:// http://www.nubomedia.eu/

[2] Loreto, Salvatore, and Simon Pietro Romano. *Real-Time Communication with WebRTC: Peer-to-Peer in the Browser*. " O'Reilly Media, Inc.", 2014.

[3] Lopez-Fernandez, Luis, et al. "Authentication, Authorization, and Accounting in WebRTC PaaS Infrastructures: The Case of Kurento." *Internet Computing, IEEE* 18.6 (2014): 34-40

[4] https://www.docker.com

[5] http://kubernetes.io/

[6] http://kurento.org/

[7] http://www.openshift.org/

[8] Boniface, M., Nasser, B., Papay, J., Phillips, S. C., Servin, A., Yang, X., & Kyriazis, D. (2010, May). Platform-as-a-service architecture for real-time quality of service management in clouds. In Internet and Web Applications and Services (ICIW), 2010 Fifth International Conference on (pp. 155-160). IEEE

[9] Jung, J., Kim, M., & Lee, H. (2015). A Study on Efficient Design of A Multimedia Conversion Module in PESMS for Social Media Services. International Journal of Electrical and Computer Engineering (IJECE), 5(4).

[10] Khan, A., & Ahirwar, K. (2011). Mobile cloud computing as a future of mobile multimedia database. International Journal of Computer Science and Communication, 2(1), 219-221.

[11] Kim, Won. "Cloud Computing: Today and Tomorrow." Journal of object technology 8.1 (2009): 65-72.

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

[13] Open Networking Foundation, "Software-Defined Networking: The New Norm for Networks", white paper, Apr. 2012.

[14] Carella, G.; Magedanz, T.; Campowsky, K.; Schreiner, F., "Elasticity as a service for federated cloud testbeds," *Communications Workshops (ICC), 2013 IEEE International Conference on* , vol., no., pp.256,260, 9-13 June 2013

[15] https://www.openstack.org/