NUBOMEDIA: The First Open Source WebRTC PaaS

Boni García
Universidad Rey Juan Carlos
Calle Tulipán S/N
28933 Móstoles (Spain)
boni.garcia@urjc.es

Luis López
Universidad Rey Juan Carlos
Camino del Molino S/N
28943 Fuenlabrada (Spain)
luis.lopez@urjc.es

Francisco Gortázar
Universidad Rey Juan Carlos
Calle Tulipán S/N
28933 Móstoles (Spain)
francisco.gortazar@urjc.es

Micael Gallego
Universidad Rey Juan Carlos
Calle Tulipán S/N
28933 Móstoles (Spain)
micael.gallego@urjc.es

Giuseppe Antonio Carella
Technische Universität Berlin
School IV Elektrotechnik und Informatik
Marchstraße 23, 10587 Berlin (Germany)
giuseppe.a.carella@tu-berlin.de

ABSTRACT

In this paper, we introduce NUBOMEDIA, an open source elastic cloud Platform as a Service (PaaS) specifically designed for real-time interactive multimedia and WebRTC services. NUBOMEDIA exposes its capabilities through simple Application Programming Interfaces (APIs), making possible to deploy and execute developers' applications. To that aim, NUBOMEDIA combines the simplicity and ease of development of API services with the flexibility of PaaS infrastructures. Once an application is implemented, developers just need to deploy it on top of NUBOMEDIA providing elasticity as a service and reliable communication.

CCS CONCEPTS

• **Information systems → Information systems applications → Multimedia information systems**; • **Networks → Networks architectures → Programming interfaces**

KEYWORDS

WebRTC, Platform as a Service, Application Programming Interface, Open Source.

Submitted to ACM MULTIMEDIA 2017 Open Source Software Competition

1 INTRODUCTION

WebRTC is a set of emerging technologies and APIs having the ambition of bringing high-quality real-time communications to the Web [1]. WebRTC is a joint standardization effort between the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF). On the one hand, W3C is defining the JavaScript APIs in so-called WebRTC 1.0 and the standard HTML5 tags to enable peer-to-peer (P2P) connections between web-enabled devices [2]. On the other hand, IETF is defining the underlying communication protocols such as SRTP (Secure Real-time Transport Protocol), SDP (Session Description Protocol), or ICE (Interactive Connectivity Establishment), for the setup and management of a reliable communication channel between browsers [3].

WebRTC has been conceived as a peer-to-peer architecture where browsers can directly communicate without the mediation of any kind of infrastructure. This model is enough for creating basic applications but features such as group communications, media stream recording, media broadcasting or media transcoding are difficult to implement using this architectural model. For this reason, many applications require using a media server infrastructure. The common features of WebRTC media servers include [4]: i) Media bridging capabilities (interoperability among networks or domains having incompatible media formats or protocols). ii) Group communication capabilities include mixing and forwarding. iii) Media archiving capabilities (recording audiovisual streams into structured or unstructured repositories and recovering them later for visualization).

This paper presents NUBOMEDIA [5], an open source Platform as a Service (PaaS) aimed to provide an elastic media server infrastructure for hosting WebRTC-based applications. NUBOMEDIA has been built on the top of Kurento Media Server [7], which provides a unique rich toolbox of media features. NUBOMEDIA has been released under the terms of the Apache version 2.0 license. Its source code is hosted on GitHub [8]. If you want to put your hands on it quickly, the best way is to take a look the NUBOMEDIA documentation [9] that includes tutorials, API description, installation guide, etc.

2 WEBRTC IN THE CLOUD

In the WebRTC ecosystem, scalable clouds for developers are not new. Providers such as Tokbox, Kandy, Twilio and many others offer them. These solutions are commonly called "WebRTC API PaaS", "WebRTC Cloud APIs", or just "Cloud APIs" as they expose a number of WebRTC capabilities through custom APIs.

MM '17, October 23-27, 2017, Mountain View, CA, USA

ACM ISBN 978-1-4503-4906-2/17/10...$15.00.
DOI: http://dx.doi.org/10.1145/3123266.3129392

Although APIs are the main building block developers use for creating applications, applications are more than just a set of API calls. After analyzing WebRTC developers' needs, we felt more appealing the concept of platform than the concept of API. A platform is more than an API in the sense that it provides all the required facilities for executing applications. These typically include an operating system, some programming-language-specific runtime environments and some service APIs.

There are many such platforms in the market, including Heroku or AWS Elastic Beanstalk, exposing to developers the ability of uploading, deploying, executing and managing applications written in different programming languages. These PaaS services allow developers to concentrate on creating their applications' logic while all the complex aspects of provisioning, scaling and securing them are assumed by the PaaS [10]. In spite of the wide offer of PaaS services, we noticed that most common PaaS providers did not expose WebRTC capabilities as part of their APIs. Hence, WebRTC developers were not able to enjoy all the advantages of full PaaSes.

Based on these ideas, the NUBOMEDIA concept emerged clearly: instead of evolving Kurento into a cloud API we should rather create a full PaaS out of it, so that developers could enjoy the nice features of PaaSes (i.e. application deployment, execution, scaling, etc.) while consuming the Kurento APIs in a scalable and secure way.

From a practical perspective, the main differences between NUBOMEDIA and other WebRTC cloud solutions are illustrated in the next figure. As it can be seen, there is a trade-off between flexibility and simplicity: the simplest the development, the less flexible the application is, and the more difficult it is to adapt it to custom needs and requirements. NUBOMEDIA also positions within this balance but giving more prevalence to flexibility. This makes NUBOMEDIA more suitable for developments requiring to comply with special or rare requirements.

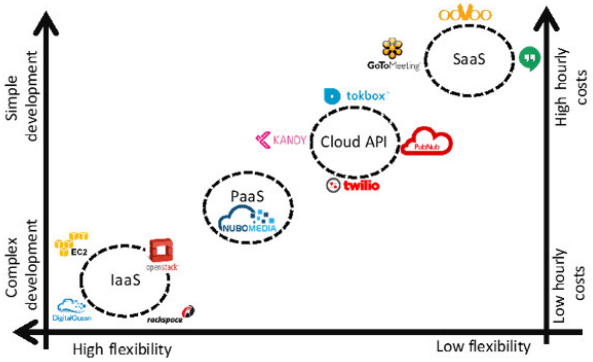

Figure 1. WebRTC cloud API vs PaaS

3 NUBOMEDIA ARCHITECTURE

NUBOMEDIA is a complex framework providing many different functions, capabilities and interfaces. For the sake of clarity and brevity, in this paper we only identify the main parts of it and we do not dig deeply into any complex details. The NUBOMEDIA high-level architecture is depicted in Figure 2. This figure shows the NUBOMEDIA system architecture following the Fundamental Modeling Concepts (FMC) visual notation [11].

The NUBOMEDIA architecture is based on the ETSI NFV Management and Orchestration (MANO) specification [12] but extends it with a PaaS layer. Thanks to it, when developers deploy their applications through a PaaS API, the PaaS Manager orchestrates all the required actions for the provisioning of the appropriate resources and services required for applications to run in a reliable and scalable way.

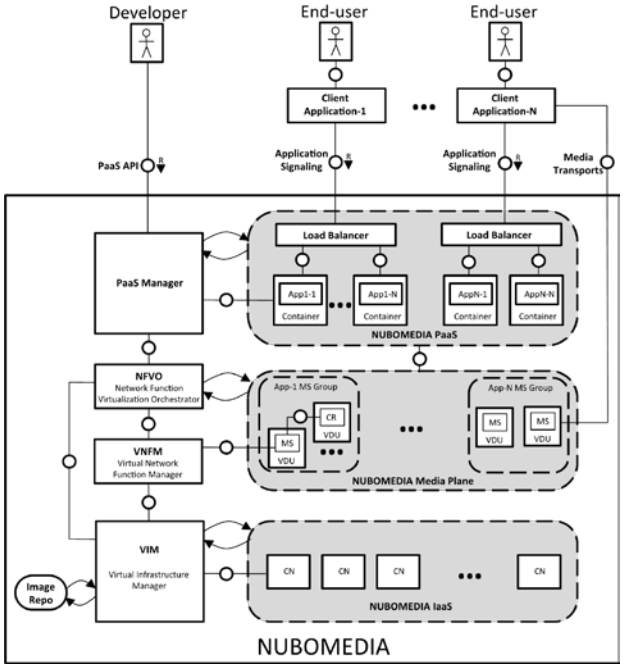

Figure 2. NUBOMEDIA Architecture

At a very high-level perspective, and following a top-down approach, these are the core functions:

- NUBOMEDIA PaaS being the intermediate level between the infrastructural resources and the users of the platform. Particularly it includes the NUBOMEDIA PaaS Manager exposing the PaaS API to the developers for deploying their applications, and the NUBOMEDIA PaaS hosting the applications.
- NUBOMEDIA Media Plane composed by the media plane capabilities (Media Servers and Cloud Repository) whose lifecycle is managed by the Network Function Virtualization Orchestrator (NFVO) and Virtual Network Function Manager (VNFM) following the guidelines of the ETSI NFV specification for the virtualization of Network Functions.
- NUBOMEDIA IaaS composed by the infrastructure resources in terms of Compute Nodes (CN) and the Virtual Infrastructure Manager (VIM) providing the compute, storage and networking resources to the upper layers

3.1 NUBOMEDIA PaaS

The NUBOMEDIA PaaS implementation is based on OpenShift [6], and it is the part of the system enabling developers to deploy and manage their media-enabled applications. NUBOMEDIA

applications are defined as a combination of the application, hosted on the PaaS, and its required media functions, hosted on the IaaS. The NUBOMEDIA PaaS elastically provides the service layer infrastructure (i.e. the needed instances of Media Servers and a Media repository if needed) depending on the load of the system.

The capabilities provided by the NUBOMDIA PaaS can be accessed by developers using the PaaS Manager, in charge of the full lifecycle of NUBOMEDIA applications. The capabilities provided by the PaaS Manager can be used by developers in two ways:

- Using the PaaS GUI. The PaaS Manager GUI is a web application that allow to use the PaaS Manager.
- Using the PaaS Manager API. The PaaS Manager exposes its capabilities by means of a REST API.

3.2 NUBOMEDIA Media Plane

This NUBOMEDIA Media Plane is one of the most relevant components of the architecture because it holds the media capabilities that include transport, transcoding, processing, mixing and archiving. From implementing the NUBOMEDIA Media Server we have extended and adapted Kurento Media Server [7].

Kurento introduces the concept of Media Element. A Media Element is a module that holds a specific media capability. For example, the media element called *WebRtcEndpoint* holds the capability of sending and receiving WebRTC media streams, the media element called *RecorderEndpoint* has the capability of recording into the file system any media streams it receives, the *FaceOverlayFilter* detects faces on the exchanged video streams and adds a specific overlaid image on top of them, etc.

As a differential factor, NUBOMEDIA provides the capability of scaling horizontally Kurento Media Server instances. To that aim, NUBOMEDIA provides an API to applications for retrieving dynamically the available Media Servers. This API is consumed by the NUBOMEDIA Media API implementation to determine in which specific Media Server instance a newly created Media Pipeline is placed. This is transparent to the developer who just needs to create the Media Pipelines without worrying about how they are located.

For this, the Media Plane has been extended to provide semantics to that placement interface and guarantees the availability of Media Servers through a horizontal autoscaling mechanism based on two operations: scaling-out (i.e. to add resources when necessary) and scaling-in (i.e. to remove resources when they are no longer required). Both, the scaling-in and -out, are fired by simple autoscaling policies based on the existing number of media sessions. In order to provide such functionality NUBOMEDIA applications must use the *nubomedia-media-client* library which communicates with the NUBOMEDIA control layers whenever a session in instantiated or closed.

3.3 NUBOMEDIA IaaS

The Virtual Infrastructure Manager (VIM) provides an interface for controlling the NUBOMEDIA IaaS. As in the ESTI NFV specification, the VIM follows the OpenStack [13] approach and, through its APIs, offers the ability to start new computing resources by using already pre-configured images containing the appropriate artifacts for every required function. The NUBOMEDIA NFVO has been implemented by adapting and extending Open Baton [14], an open source software implementation of the NVF ETSI recommendations.

The media elements are instantiated on the NUBOMEDIA IaaS on top of one or more Compute Nodes (CN) using Docker containers [15] and KVM (Kernel-based Virtual Machine) virtual machines [16].

4 NUBOMEDIA APIS

From the developer's perspective, NUBOMEDIA capabilities are accessed through a set of APIs. Hence, for creating NUBOMEDIA applications, developers just need to understand these NUBOMEDIA development APIs so that they may use them for creating their rich media applications. The NUBOMEDIA APIs are summarized in the following table:

Table 1: NUBOMEDIA APIs

API	Description
Media API	Enables developers consuming the **media server** capabilities, such as media transport, media archiving, media processing, transcoding, etc.
WebRtcPeer API	Abstracts the **client WebRTC** media capabilities, exposing the media capture and communication capabilities of a browser in a seamless way
Repository API	Makes possible to access an elastic scalable **media repository** for archiving media information and meta-information
Signaling API	Provides a simple **signaling** mechanism based on JSON-RPCs for applications
Room API	Enables application developers' functionalities to create **group communication** applications adapted to real social interactions
Tree API	Allows developers to build video **broadcasting** web applications

In addition, NUBOMEDIA offers SDKs for Android and iOS development. These SDKs provides the client-side of the above-mentioned APIs for Android and iOS devices, allowing to create mobile applications consuming the media capabilities provided by NUBOMEDIA.

5 CREATING NUBOMEDIA APPLICATIONS

From the application developer perspective, Media Elements are like Lego pieces: the developer just needs to take the elements needed for an application and connect them following the desired topology. When creating a Media Pipeline, developers need to determine the capabilities they want to use (the Media Elements) and the topology determining which media elements provide media to which other media elements (the connectivity). The connectivity is controlled through the *connect* primitive, exposed on all Media Elements. This primitive is always invoked in the element acting as source and takes as argument the sink.

Following this approach, developers can create applications with media capabilities is a seamless way. Figure 3 shows several examples of Media Pipelines together with the screenshot of the application implementing each one. First, we can see a group

communication application example. This application is a N-to-N (i.e. video room) WebRTC communication. The second example is a 1-to-1 WebRTC application with recording capabilities. Third, we find a simple example of augmented reality, in which the WebRTC user media feeds a *FaceOverlayFilter*, which overlays an image on the top of the detected faces in the media. Finally, a computer vision is illustrated with an application with receives the live RTSP media of a street cam using a *PlayerEndpoint* detecting crowds in the media and sent to a browser by means of a *WebRtcEndpoint* in receive-only mode.

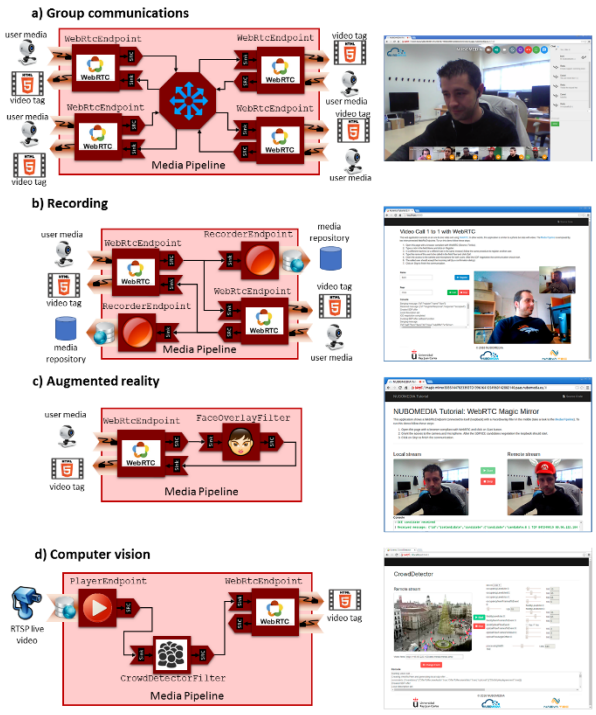

Figure 3. Example of applications created with NUBOMEDIA (Media Pipelines and screenshots)

The deployment of these applications in NUBOMEDIA is quite simple. First, the source code should be committed on a GitHub repository. Second, and due to fact The NUBOMEDIA PaaS uses Docker containers internally, developers need to specify the deployment configuration by means of a Dockerfile located in the root of that repository. Finally, the GitHub repository URL is provided to the NUBOMEDIA PaaS Manager handling the deployment of the application.

The documentation of NUBOMEDIA [9] contains several tutorials, in which this process in detailed through simple example applications. Moreover, the NUBOMEDIA YouTube Channel [17] shows different use cases in action.

6 CONCLUSIONS

In this paper, we have introduced NUBOMEDIA, a PaaS platform that aimed to simplify the development, deployment and scalability of WebRTC applications. Along the paper, we have tried to stress the importance of listening to developers' needs and solving developers' problems. We have illustrated now NUBOMEDIA can be used by developers for saving thousands of hours of effort when creating advance WebRTC applications.

From the developer's perspective, NUBOMEDIA capabilities are accessed through a set of APIs. These capabilities include the ability of accessing scalable, secure and reliable WebRTC applications. As a differential factor, NUBOMEDIA provides the capability of scaling in and out Kurento Media Server instances. Finally, the PaaS Manager is the component used by developers to deploy WebRTC application in a seamless way. This component provides access to the NUBOMEDIA PaaS, which is in charge of the full lifecycle of applications: deployment, hosting on the PaaS, and of course, elastic provisioning of media functions with full transparency.

ACKNOWLEDGMENTS

This work has been supported by the European Commission under projects NUBOMEDIA (FP7-ICT-2013-1.6, GA-610576), and ElasTest (H2020-ICT-10-2016, GA-731535); by the Regional Government of Madrid (CM) under project Cloud4BigData (S2013/ICE-2894) cofunded by FSE & FEDER; and Spanish Government under project LERNIM (RTC-2016-4674-7) cofunded by the Ministry of Economy and Competitiveness, FEDER & AEI.

REFERENCES

[1] Loreto, S. and Romano, S. P. Real-Time Communication with WebRTC: Peer-to-Peer in the Browser. O'Reilly Media Inc., 2014.
[2] W3C *WebRTC 1.0: Real-time Communication Between Browsers*. 2017. https://www.w3.org/TR/webrtc/.
[3] IETF *Real-Time Communication in WEB-browsers (Active WG)*. 2017. https://tools.ietf.org/wg/rtcweb/.
[4] Grigorik, I. High Performance Browser Networking: What every web developer should know about networking and web performance. O'Reilly Media Inc., 2013.
[5] NUBOMEDIA *The Open Source Cloud for Real-Time Multimedia Communications*. 2016. http://www.nubomedia.eu/. Accessed: May 2017.
[6] OpenShift *Container application platform*. 2017. https://www.openshift.com/. Accessed: May 2017.
[7] Kurento Open Source WebRTC media server. 2016. http://www.kurento.org/. Accessed: May 2017.
[8] NUBOMEDIA *GitHub organization*. 2016. https://github.com/nubomedia. Accessed: May 2017.
[9] NUBOMEDIA *Developer Guide*. 2016. http://nubomedia.readthedocs.io/. Accessed: May 2017.
[10] Kavis, M. J. *Architecting the cloud: Design decisions for cloud computing service models (SaaS, PaaS, AND IaaS)*. John Wiley & Sons, 2014.
[11] Tabeling, P., Gröne, B. and Knöpfel, A. *Fundamental modeling concepts-effective communication of it systems*. John Wiley & Sons, Ltd, 2006.
[12] Han, B., Gopalakrishnan, V., Ji, L. and Lee, S. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, 53, 2 (2015), 90-97.
[13] OpenStack *Open-source software Infrastructure as a Service (IaaS)*. 2017. https://www.openstack.org/. Accessed: May 2017.
[14] Open Baton *Open source Network Function Virtualisation Orchestrator (NFVO)*. 2017. https://openbaton.github.io/. Accessed: May 2017.
[15] Docker *Software container platform*. 2016. https://www.docker.com/. Accessed: May 2017.
[16] KVM *Kernel-based Virtual Machine*. 2017. https://www.linux-kvm.org/. Accessed: May 2017.
[17] NUBOMEDIA Official *YouTube Channel*. 2017. https://www.youtube.com/channel/UCo-SKxfRjxAb0fRj9NCvVIQ. Accessed: May 2017.