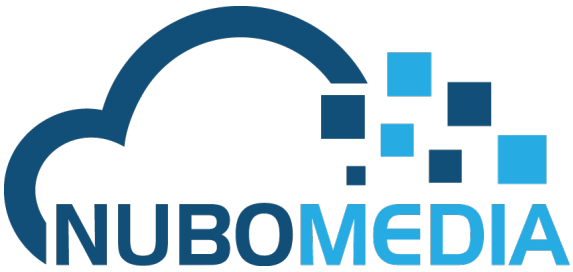
D5.1	
Version	1.0
Author	URJC
Dissemination	PU
Date	27/01/2015
Status	Final



D5.1: NUBOMEDIA develop tools analysis document

Project acronym:	NUBOMEDIA
Project title:	NUBOMEDIA: an elastic Platform as a Service (PaaS) cloud for interactive social multimedia
Project duration:	2014-02-01 to 2016-09-30
Project type:	STREP
Project reference:	610576
Project web page:	http://www.nubomedia.eu
Work package	WP5
WP leader	Luis López
Deliverable nature:	Report
Lead editor:	Luis López
Planned delivery date	01/2015
Actual delivery date	27/15/2015
Keywords	NUBOMEDIA, Kurento, Application Programming Interface

The research leading to these results has been funded by the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement nº 610576

SEVENTH FRAMEWORK PROGRAMME

Information Society Technologies

European Commission

FP7 ICT-2013.1.6. Connected and Social Media

Contributors:

Luis Lopez (URJC)

Internal Reviewer(s):

Mäkelä Satu-Marja (VTT)

Version History

Version	Date	Authors	Comments
0.1	03-01-2015	Luis Lopez	Initial Version
1.0	19-01-2015	Luis Lopez	Completed implementation status and set as final version for R3.

Table of contents

List of Figures:

Acronyms and abbreviations:

API	Application Programming Interface
AR	Augmented Reality
CDN	Contend Distribution Network
FOSS	Free Open Source Software
IMS	IP Multimedia Subsystem
IoT	Internet of Things
KMS	Kurento Media Server
MCU	Multipoint Control Unit
NFV	Network Function Virtualization
RTC	Real-Time Communications
RTP	Real-time Transport Protocol
SCTP	Stream Control Transmission Protocol
SFU	Selective Forwarding Unit
UE	User Equipment
VCA	Video Content Analysis
VoD	Video on Demand
WebRTC	Web Real Time Communications

1 Executive summary

This document presents the Application Programming Interfaces (APIs) and tools that NUBOMEDIA will expose to developers. We first introduce the two development profiles that play a role in NUBOMEDIA: platform developers and application developers. We show why these profiles are necessary and we explain their roles in the lifecycle of applications. To conclude, we introduce the different APIs and tools that NUBOMEDIA will make available for these developers depending on their profile and sketch a roadmap for their implementation.

2 Introduction

NUBOMEDIA is a platform, meaning that its objective is exposing PaaS APIs and SaaS services to application developers and end users. When looking to PaaS APIs, in current state of the art there is not a clear definition of what's a PaaS API [1,2,3]. In the context of NUBOMEDIA, we understand that a PaaS API is a networked API that exposes RTC capabilities and that can be consumed by application developers for creating end-user applications. In other words, for us a PaaS API consist of a network interface based on some kind of protocol (e.g. RESTful+XML, SIP, WebSocket+JSON, etc.) that can be used by application developers for providing specific RTC capabilities to their applications.

The emergence of WebRTC technologies has brought an explosion in RTC PaaS APIs following the above described model and currently the marked is flooded with WebRTC PaaS API providers. Current market offer includes Tokbox, Temasys, Bistri, Acision, Twilio, Genband, etc. This multiplicity of solutions is an indication of the appropriateness of these types of PaaS APIs and the preference of developers for them. This preference is probably due to the numerous advantages of the networked-based PaaS API model, which include:

- Programing language independent: so that the APIs can be consumed in a language-independent way.
- Privacy and IPR protection: given that application developers do not need to share with the platform any kind of private data from the end-users nor application source code.
- Pay-per-use models: so that application developers can be charged in a per-use-model which may be based on API calls or on API usage time.
- Isolation: so that the platform and the application can evolve independently from each other as long as the interface (networked API) among them does not change.
- Elasticity: so that application developers can start small without requiring large investments and can grow very large utilizing the elasticity of the platform.

Hence, following this model, in the lifecycle of the application we need to distinguish different roles:

- The PaaS provider, which is in charge of deploying and maintaining the PaaS infrastructure and offers its capabilities through the PaaS API.
- The application provider, who exposes RTC enabled applications based on the PaaS API.
- The end-user, representing a person accessing the application (and hence consuming the PaaS API) from a device.

Following this model, end-users access the PaaS resources but the PaaS provider charges the application provider for it. Hence, end-user identities are managed by the application provider and access the PaaS on its behalf. Thanks to this, the application provider can implement different types of business models without requiring the PaaS to own any kind of data about end-users.

Hence, when looking to the developer's perspective, we can distinguish two types of developers:

- Platform developers: these developers create the platform logic and make it accessible through the PaaS API to the external world.
- Application developers: these developers create the end-user applications consuming the PaaS APIs and additional APIs or capabilities external to the platform.

In most PaaS in the market, platform developers are not very relevant and no efforts are consumed for making their lives simpler. However, in the case of NUBOMEDIA, platform developers are the keystone to the success of the project. The reasons are straightforward:

- State-of-the-art PaaS APIs are designed just for communications, this makes platforms to hold just simple and uniform APIs. However, NUBOMEDIA exposes much more than communications. Hence, the spectrum of capabilities that NUBOMEDIA can expose is much wider and the creation of different kinds of PaaS APIs providing services in different vertical markets and with disparate characteristics is more than reasonable. As a result, NUBOMEDIA may have chances of disrupting the market and catalyzing a new generation of services only if platform developers find the appropriate tools suitable for creating different PaaS APIs in a simple and direct manner.
- As any other FOSS initiative, the commercial success of NUBOMEDIA comes from the ability of companies and organizations to create technologies differentiating them from the rest of the market. For this reason, the role of platform developers is critical given that this differentiation may emerge when creating specific purpose and innovative APIs capable for satisfying the requirements of vertical market segments.
- NUBOMEDIA target primary audience is platform providers and not application developers. Simplifying the work of platform developer is important for incentivizing NUBOMEDIA adoption among potential providers.

3 Objectives

This document has the objective of presenting an architecture and a technological strategy introducing main NUBOMEDIA development APIs and sketching a roadmap for their implementation. For achieving this objective a number of sub objectives need to be fulfilled:

- To identify the different developer profiles playing a role in NUBOMEDIA application lifecycle.
- To identify the different APIs that need to be exposed by NUBOMEDIA for satisfying the needs of the different developer profiles.
- To identify any additional tools beyond the APIs that may be useful for the different developer profiles.

- To provide a comprehensive description of the different functions of the APIs in relation to the NUBOMEDIA architecture.
- To sketch a technological strategy for the implementation of the APIs and to forecast the major problems that may emerge.

4 NUBOMEDIA APIs and development tools

As specified above, NUBOMEDIA development require two profiles of developments with independent lifecycles: platform and application. Application developers work in contact with the client device (i.e. User Equipment or UE) and may use External Application Servers for providing the appropriate business intelligence required by their applications. On the other hand, platform developers work in contact with the NUBOMEDIA infrastructure and create the appropriate logic for exposing networked PaaS APIs and services to the external world.

From the perspective of developers, the NUBOMEDIA architecture and the APIs are the ones depicted in Figure 1. The components and interfaces shown in this picture are described in the following sections.

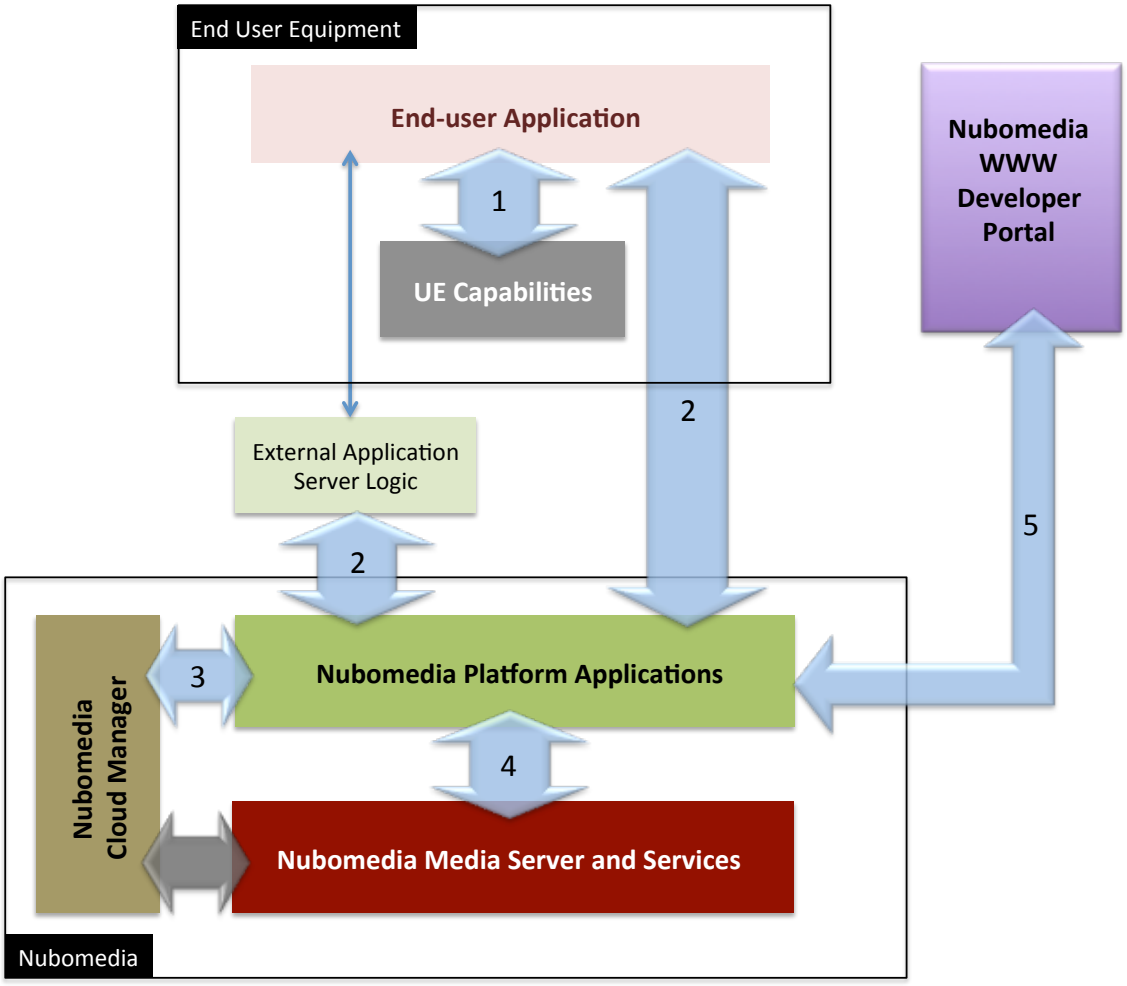


Figure 1. This figure shows the different development APIs involved in the NUBOMEDIA project. These are: (1) Client Capabilities API (NUBO-CAPI), (2) NUBOMEDIA PaaS API (NUBO-PAPI), (3) NUBOMEDIA Cloud API (NUBO-CLAPI), (4) NUBOMEDIA Media Server API (NUBO-MAPI) and (5) NUBOMEDIA WWW Developer Portal API (NUBO-WAPI)

4.1 Components

4.1.1 End User Equipment (UE)

This represents the device of the end-user accessing the application. It may take the form of a smartphone or of a desktop computer. When accessing from WWW browsers, device capabilities are limited to the features exposed by the browser sandbox security model.

Inside the end user equipment, the following modules will be located:

- User Equipment capabilities: this represents the native capabilities of the device or WWW browser that are consumed by the application.
- End-user application: this represents the application logic created by the application developer and executed at the UE.

4.1.2 NUBOMEDIA

This represents the NUBOMEDIA infrastructure. In a synthesized way, and for the eyes of developers, this architecture comprises the following modules:

- NUBOMEDIA Platform Applications: these hold the platform logic for exposing the PaaS APIs and SaaS services. They are contained in the NUBOMEDIA Application Server function, as explained in NUBOMEDIA architecture (see D2.4.1).
- NUBOMEDIA Media Server and Services: these hold the low level media capabilities including media transport, media processing, VCA, AR and repository. They correspond with the Media Server Function, as explained in NUBOMEDIA architecture (see D2.4.1).
- NUBOMEDIA Cloud Manager: this module provides the cloud functions, which include the EMM and the CM, as explained in the NUBOMEDIA architecture (see D2.4.1).

4.1.3 WWW Developer Portal

This module represents a visual facility for helping developers. In principle, it should provide access to monitoring information helpful for the debugging of NUBOMEDIA applications and for supporting NUBOMEDIA enabled services.

4.2 Developer APIs

4.3 Client Capabilities API (NUBO-CAPI) - 1

The NUBO-CAPI, marked as (1) in Figure 1 is an API exposing and abstracting UE RTC capabilities. In other words, this API must make possible accessing media capture sensors (i.e. camera, microphone, etc.), screen rendering (i.e. RTC media rendering) and media transport. Hence, this API is wrapping lower level APIs of the device and is exposed to the developer through primitives and data types on the language of the client platform (i.e. JavaScript for WWW clients, Java for Android clients, etc.).

This API is consumed by the Application Developer.

This API is created by the NUBOMEDIA consortium.

4.4 NUBOMEDIA PaaS API (NUBO-PAPI) - 2

The NUBO-PAPI, marked as (2) in Figure 1 is an API exposing and abstracting NUBOMEDIA Media Server and Services capabilities to the end-user application. In other words, this API exposes a network interface (e.g. RESTful-XML, WebSocket-JSON, SIP, etc.) that can be consumed by external applications and devices for providing NUBOMEDIA-enabled RTC services to end-user applications. The network based interface shall be wrapped so that the NUBO-PAPI can be consumed by the developer through primitives and data types on the language of the client platform (i.e. JavaScript for WWW clients, Java for Android clients, etc.).

This API is consumed by the Application Developer.

This API is created by the Platform Developer.

For illustration purposes, and for covering some common applications scenarios, the NUBOMEDIA consortium will provide pre-built PaaS APIs for some of the most popular use-cases including room-based group communications and RTC broadcasting services.

4.5 NUBOMEDIA Cloud API (NUBO-CLAPI) – 3

The NUBO-CLAPI, marked as (3) in Figure 1 is an API exposing and abstracting cloud infrastructures to the Platform Developer. This API makes possible for the application to consume auto-scaling Media Server and Services, with intelligent load balancing adapted to the real load and capabilities of the virtual instances, in an abstract way. This means that the NUBO-CLAPI abstracts the specific cloud manager details, making possible for NUBOMEDIA Platform Applications to be deployed on any cloud (i.e. Amazon, OpenStack, etc.).

The main objective of this API is to offer to the Platform Application developer a placement service. Hence, the NUBO-CLAPI shall be exposing at least one primitive that the Platform Application can consume for determining what is the best Media Server instance where a specific media capability will be placed.

The NUBO-CLAPI will take the form of network-based API using simple standard transport and marshaling mechanisms (i.e. RESTful JSON, etc.). For simplifying the work of developers, the network interface shall be wrapped and exposed through stub objects and primitives through the computer languages supported by the NUBOMEDIA Application Server Function (i.e. Java).

This API is consumed by the Platform Developer.

This API is created by the NUBOMEDIA Consortium.

4.6 NUBOMEDIA Media Server API (NUBO-MAPI) - 4

The NUBO-MAPI, marked as (4) in Figure 1, is an API for controlling Media Server services. This API plays a similar role to the H.248 protocol (i.e. MEGACO) [4] in IMS architectures. Other similar API is the one defined in RFC 5707 (i.e. Media Server Markup Language - MSML) [5]. In essence, these APIs make possible to create and manage the lifecycle of media capabilities such as endpoints, media routers, media mixers, media transcoders, etc.

The NUBO-MAPI provides access to NUBOMEDIA Media Server capabilities which include media transport, media transcoding, media mixing, media routing, media processing, Augmented Reality (AR) and Video Content Analysis (VCA). The NUBO-MAPI shall be based on a network interface with PUSH (bi-directional) capabilities (e.g. WebSocket + JSON-RPC). For simplifying the work of developers, the network interface shall be wrapped and exposed through stub objects and primitives through the computer languages supported by the NUBOMEDIA Application Server Function (i.e. Java)

This API is consumed by the Platform Developer.

This API is created by the NUBOMEDIA Consortium.

4.7 NUBOMEDIA WWW developer portal and API (NUBO-WAPI) - 5

The NUBO-WAPI , marked as (5) in Figure 1, consists of a control and monitoring API that, combined with a WWW interface, offer helper tools capable of simplifying the life of application developers. These tools may comprise debuggers, profilers, configurers and monitors.

The NUBO-WAPI shall consist of a network interface providing PUSH (bi-directional) capabilities (e.g. WebSocket + JSON-RPC). This interface shall be wrapped into JavaScript, for simplifying the creation of the NUBOMEDIA WWW developer portal.

This API is consumed by the WWW development portal.

This API is created by the NUBOMEDIA consortium.

5 Implementation strategy

Given the descriptions above, the summary of NUBOMEDIA developer APIs is presented in the table bellow:

	Crated by	Used by	Networked	Wrappers	Logic at
NUBO-CAPI	NUBOMEDIA	Application developer	No	Java JavaScript ObjectiveC	Client
NUBO-PAPI	Platform developer	Application developer	Yes	Java JavaScript ObjectiveC	Infrastructure
NUBO-CLAPI	NUBOMEDIA	Platform developer	Yes	Java	Infrastructure
NUBO-MAPI	NUBOMEDIA	Platform developer	Yes	Java JavaScript	Infrastructure
NUBO-WAPI	NUBOMEDIA	WWW portal	Yes	JavaScript	Infrastructure

5.1 NUBO-CAPI implementation strategy

Given the descriptions above and the NUBOMEDIA objectives, the NUBO-CAPI must provide RTC multimedia capabilities suitable for accessing NUBOMEDIA features from WWW, Android and iOS platforms. This means that this API must make possible to produce and to consume media (i.e. audio, video and sensor data) to/from NUBOMEDIA. There are many possibilities for using this, but given current technological trends and given NUBOMEDIA technical strategy, the best option is to base the implementation on a WebRTC stack.

Following this strategy, the support for WWW platforms will be provided through W3C WebRTC standard APIs, which will be wrapped accordingly to the NUBO-CAPI requirements. At the time of this writing, W3C WebRTC APIs are available in Chrome, Firefox and Opera. There are plans for having Internet Explorer (or its successors) supporting WebRTC during 2015.

On the other hand, the support of WebRTC natively on smartphone platforms is not provided out off the self by any of them. For this reason, a WebRTC stack needs to be installed prior to being able to use the NUBO-CAPI. Creating a full client-side WebRTC stack from zero is out of the reach of a project like NUBOMEDIA. Hence, our strategy needs to be based on re-using one of the available ones. In this direction, there are two main options: the webrtc.org stack and the openwebrtc.io

The webrtc.org stack

This stack has been created by Google and it's the underlying software below the Chrome WebRTC implementation. Its main characteristics are the following:

- Documentation available.
- Source code available.
- License compatible with the LGPL v2.1 license used by NUBOMEDIA technologies.
- Can be compiled into Android (full support) and iOS (partial support).

The openwebrtc.io stack

This stack has been created by Ericsson consuming GStreamer capabilities and it's the underlying software below the Bowser WebRTC implementation. Its main characteristics are the following:

- Documentation available.
- Source code available.
- License compatible with the LGPL v2.1 license used by NUBOMEDIA technologies.
- Can be compiled into Android and iOS.

Basing on these two options, the NUBO-CAPI implementation strategy consists of compiling, validating and wrapping any of them to expose the appropriate APIs to application developers.

5.2 NUBO-PAPI implementation strategy

Given the descriptions above and the NUBOMEDIA objectives, the NUBO-PAPI is the PaaS API exposing NUBOMEDIA infrastructure capabilities to the end users. The NUBO-PAPI shall be created by platform developers and its characteristics depend on the specific application logic and function.

For illustration purposes, and for providing the appropriate APIs adapted to NUBOMEDIA requirements, we plan to create two different types of NUBO-PAPIs: a room-based (ROOM-PAPI) one and a broadcasting-based (TREE-PAPI) one.

5.2.1 ROOM-PAPI

This API will expose capabilities for the creation of group communication services based on room models. In these models, users share a common virtual space (the room) so that all users entering the room can share their video. This model is quite common in services such as Google Hangouts or Skype.

Following this model, the number of users in a single room is limited to a given maximum (e.g. 10) and not being possible for new user to enter the room after this limit has been reached.

Hence, this API should provide primitives with semantics such as the following:

- Create room.
- Enter room.
- Share my video with the room.
- Get video from a room member.

The ROOM-PAPI will be based on some kind of network interface with PUSH capabilities, given that events need to be pushed from the infrastructure to the client (e.g. new users entering the room, user leaving the room, etc.) In principle, JSON-RPC over WebSocket is the preferred mechanism, although it may be reviewed at later stages. Client side wrappers will be created on the different supported client platforms for simplifying the application development process.

The creation of this API requires developing the appropriate platform application logic suitable for the management of rooms, users and streams. This logic needs also to consume the NUBO-CLAPI for making the placement of the rooms onto the corresponding media server instances.

5.2.2 TREE-PAPI

This API will expose capabilities for the creation of large broadcasting distribution trees with low latency. In this model, a presenter user produces a RTC stream which is send to NUBOMEDIA, and the platform makes possible for viewers to receive the stream in a scalable way, so that the number of viewers can grow or shrink without the rest of the viewers noticing it. This model is quite common in streaming CDN services, but with the difference that in this case the distribution latency is small (i.e. it's real-time).

Hence, this API should provide just three types of primitives:

- Create broadcasting tree.
- Enter as presenter.
- Enter as viewer.

The TREE-PAPI will be based on some kind of network interface with PUSH capabilities given that events need to be pushed from the infrastructure to the client (e.g. presenter may wish to notice the presence of new viewers). In principle, JSON-RPC over WebSocket is the preferred mechanism, although it may be reviewed at later stages. Client side wrappers will be created on the different supported client platforms for simplifying the application development process.

The creation of this API requires developing the appropriate platform application logic suitable for the management of trees, users and streams. This logic needs to be stateful and aware of the topology of the distribution tree, which shall comprise many media server instances. For creating that topology, it needs to consume the NUBO-CLAPI, which will provide services such as the following:

- Placement of users into the tree.
- Control of tree growth.
- Control of tree pruning.

5.3 NUBO-CLAPI implementation strategy

The NUBO-CLAPI abstracts all the complexities of the cloud manager for application developer. NUBOMEDIA is based on a NFV architecture where network node functions (MS, AS, etc.) are seen as building blocks that may be connected or chained together to create RTC services. Hence, when an application requires creating new media capabilities for a user, it needs to select the most appropriate Media Server instance for placing them. This selection can be performed using simple algorithms (e.g. round robin, random, etc.) or intelligent algorithms taking into account current load of machines and current network topology and SLAs.

Given that NUBOMEDIA uses SDN SLAs and that intelligent placement reinforces QoE, placement of media capabilities is performed in runtime by the different cloud managers (i.e. Elastic Media Manager and Connectivity Manager) as described in the NUBOMEDIA Architecture in deliverable D2.4.1. Hence, the platform application needs to consume that logic under users' request.

The NUBO-CLAPI is the API making possible to consume that logic, which just requires the execution of simple "one-shot" primitives not requiring PUSH. For this reason, the original plan is to implement this API using a RESTful model marshaling messages through JSON.

In addition, a simple Java wrapper will be created, so that platform application developers are capable of consuming the API seamlessly.

5.4 NUBO-MAPI implementation strategy

The NUBO-MAPI is the API exposing the capabilities of individual media servers. Given that NUBOMEDIA is integrating Kurento Media Server (KMS) at the media layer, the NUBO-MAPI will be based on the corresponding KMS APIs as specified in Deliverable D4.1.1.

Kurento Media Server capabilities are exposed by the Kurento API to application developers. This API is implemented by means of libraries called Kurento Clients. Kurento offers two clients out of the box for Java and JavaScript. If you have another favorite language, you can still use Kurento using directly the Kurento Protocol. This protocol allows controlling Kurento Media Server and it is based on Internet standards such as WebSocket and JSON-RPC.

Kurento Client's API is based on the concept of Media Element. A Media Element 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. Kurento exposes a rich toolbox of media elements as part of its APIs.

A first version of the NUBO-MAPI has already been created as part of D4.1.1. We refer to that deliverable for further information in relation to the NUBO-MAPI.

5.5 NUBO-WAPI implementation strategy

The NUBO-WAPI consists of a number of helper APIs providing monitoring and debugging capabilities for platform and application developers. The NUBO-WAPI needs to provide access to the internal status of the different NUBOMEDIA functions, including:

- Service instances and service topologies.
- Mediapipeline status and structure.
- Session status.
- Service statistics including QoS statistics.
- Debugging information.

The precise definition of the capabilities to be exposed by the NUBO-WAPI is under discussion. However, it is clear that for the creation of this API a strategy comprising the following steps will be used:

- Implementation of the appropriate capabilities suitable for the generation of the monitoring and debugging information. These capabilities may include Media Server QoS monitors through statistics (e.g. latency, jitter, packet loss), cloud infrastructure monitoring information (e.g. types of manage images, types of scalability groups, number of images) and application specific information (e.g. number of sessions, types of sessions).
- Implementation of the appropriate capabilities suitable for collecting and consolidating the different monitoring and debugging information.
- Implementation of the APIs and mechanism exposing the monitoring and debugging information, in real-time, to platform and application developers.

A relevant aspect to be taken into account is that NUBO-WAPI needs to provide PUSH capabilities for pushing events (i.e. asynchronous monitoring and debugging information) to the WWW developer portal. For this reason, in principle a JSON-RPC protocol over WebSockets is the initial choice.

A JavaScript wrapper will be generated for the network interface, so that the creation and extension of the WWW developer portal is simplified.

6 Implementation status

6.1 NUBO-CAPI implementation status

6.1.1 Smartphone platforms

The implementation of the NUBO-CAPI for smartphone platforms has not yet started. First implementation efforts are planned for NUBOMEDIA R4.

6.1.2 WWW platform

A first simplified version of the NUBO-CAPI for WWW platforms. This preliminary version has been instrumental for the creation of demonstrators, tests and validation applications.

Source code of the WWW preliminary implementation can be found at Kurento GitHub repository as part of the kurento-utils-js project. Master branch is at the following URL:

- https://github.com/Kurento/kurento-utils-js/tree/master

The preliminary version of the WWW NUBO-CAPI has been released under the terms and conditions of the LGPL v2.1 license.

6.2 NUBO-PAPI implementation status

A first simplified implementation of the room NUBO-PAPI (ROOM-PAPI) has been created for demo and validation purposes and it's available at Kurento GitHub repository as part of the kurento-java project. The development branch can be reached at the following URL:

- https://github.com/Kurento/kurento-java/tree/develop/kurento-room

A first simplified implementation of the tree NUBO-PAPI (TREE-PAPI) has been created for demo and validation purposes and it's available at Kurento GitHub repository as part of the kurento-java project. The development branch can be reached at the following URL:

- https://github.com/Kurento/kurento-java/tree/develop/kurento-tree

Both applications and their corresponding PAPI interfaces have been created for clarifying and consolidating the technical requirements emerging from CFM, DAS and DMS, as described in NUBOMEDIA architecture (deliverable D2.4.1).

Both preliminary implementations have been released under the terms and conditions of the LGPL v2.1 license.

6.3 NUBO-CLAPI implementation status

The NUBO-CLAPI implementation status is only on its early stages by NUBOMEDIA R3. Refer to NUBOMEDIA deliverables D3.3.1 and D3.4.1 for details on NUBO-CLAPI implementation.

6.4 NUBO-MAPI implementation status

The NUBO-MAPI implementation status is mature by NUBOMEDIA R3. The NUBO-MAPI is implemented through the Kurento Protocol stack exposed by Kurento Media Server. Refer to NUBOMEDIA deliverables D4.1.1, D4.2.1 and D4.3.1 for details on the NUBO-MAPI implementation.

6.5 NUBO-WAPI implementation status

The NUBO-WAPI implementation has not started by NUBOMEDIA R3. Initial efforts are planned for it during the execution of the NUBOMEDIA R4.

7 References

[1] Lawton, G. (2008). Developing software online with platform-as-a-service technology. Computer, 41(6), 13-15.
[2] Rimal, B. P., Choi, E., & Lumb, I. (2009, August). A taxonomy and survey of cloud computing systems. In INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on (pp. 44-51). Ieee.
[3] Vaquero, L. M., Rodero-Merino, L., Caceres, J., & Lindner, M. (2008). A break in the clouds: towards a cloud definition. ACM SIGCOMM Computer Communication Review, 39(1), 50-55.
[5] H.248. Gateway Control Protocol. http://www.itu.int/rec/T-REC-H.248.1/en
[6] RFC 5707. Media Server Markup Language – MSML. http://tools.ietf.org/html/rfc5707