

D6.5	
Version	1.0
Author	TI
Dissemination	PU
Date	31/01/2017
Status	Final



D6.5: NUBOMEDIA Computer vision enhanced phone calls

Project acronym:	NUBOMEDIA
Project title:	NUBOMEDIA: an elastic Platform as a Service (PaaS) cloud for interactive social multimedia
Project duration:	2014-02-01 to 2017-01-31
Project type:	STREP
Project reference:	610576
Project web page:	http://www.nubomedia.eu
Work package	WP6: Demonstration
WP leader	ZED
Deliverable nature:	Report
Lead editor:	Teo Redondo / Paula Collazos (ZED)
Planned delivery date	01/2017
Actual delivery date	31/01/2017
Keywords	NUBOMEDIA, Demonstrator, Open Source, Apache.

The research leading to these results has been funded by the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 610576



FP7 ICT-2013.1.6. Connected and Social Media



This is a public deliverable that is provided to the community under a **Creative Commons Attribution-ShareAlike 4.0 International License**

<http://creativecommons.org/licenses/by-sa/4.0/>

You are free to:

Share — copy and redistribute the material in any medium or format

Adapt — remix, transform, and build upon the material
for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Notices:

You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation.

No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material.

For a full description of the license legal terms, please refer to:

<http://creativecommons.org/licenses/by-sa/4.0/legalcode>

Contributors:

Fabio Luciano Mondin (TI)
Claudio Venezia (TI)

Internal Reviewer(s):

Luis López (URJC)

Version History

Version	Date	Authors	Comments
1.0	12/01/2017	Claudio Venezia,Fabio Mondin	Complete Description

Table of contents

Executive summary	6
Introduction	6
1. Technological objectives of the demonstrator	7
2. Requirements that the demonstrator must satisfy	8
3. Architecture of the demonstrator (related to NUBOMEDIA architecture)	10
4. Relevant implementation details.....	14
5. Testing mechanisms and methodologies	17
6. End-user's guide and tutorial: how to install and use	19
7. Total effort consumed in creating the demonstrator	23
8. Developer's feedback in relation to using NUBOMEDIA.....	23

Executive summary

The aim of this demonstrator is to show how to set-up a technological bridge between the TELCO world and the Web Communication world.

The Demonstrator will use NUBOMEDIA outcomes in order to build a novel communication paradigm, in the middle between normal communication and surveillance/control.

The use-case starts from a webcam positioned inside the user's home, connected to NUBOMEDIA. When the system detects a face, a traditional phone-call is forwarded to a specified number setting up a bi-directional audio-communication between the home and the remote responder.

Such a scenario could be part of a more complex Smart-Home bundle offered by TIM or in general by TELCO Operators. One of the main advantages of this solution is that it is leveraging on the Operator's Access Gateway, which is able to forward a traditional phone call under commercial conditions which are already known by the user.

Moreover, after the audio-call is set-up, specific apps could leverage on the system itself to set up a direct video-communication.

Introduction

In the last few years some Telco Operators have been scouting for new opportunities for deploying Value Added Services, exploiting the improved bandwidth capabilities, optical fiber connection and smarter home gateways which are capable of managing smart environments.

One of the most promising markets is the Smart Home, where forecasts show a constant growth till 2020 and over.

In general, Smart Home applications tend to be perceived as a good way to perform customer retention, rather than a way to create the need for more bandwidth, being the data exchange among sensors and actuators quite low, but this is not completely true. There is a number of Smart Home Video-based services, linked to cloud which can be very useful for the final user and which of course require a reasonable amount of bandwidth, both in and outbound.

In this light Telecom Italia is aiming to identify scenarios in which typical Smart Home Applications could compound the Video experience towards novel fruition models where computer vision capabilities can provide new real time intelligent feedbacks based on the real-time analysis of domestic video streams (e.g. face/body/movement recognition, object tracking).

Telecom Italia's idea is leveraging on NUBOMEDIA platform to provide novel communications paradigms, in the field of Smart Home and Ambient Assisted Living, tightening the relation between Video Services, such as IPTV or Video Conference Services, and the User's place.

Thanks to NUBOMEDIA support TI is foreseeing an Event Triggered SIP/PSTN Video-Call System, which would enable a set of innovative possibilities:

NUBOMEDIA: an elastic PaaS cloud for interactive social multimedia

- Assisted Automatic Call From an in-house call point.
- Automatic Call from in case of security-critical events detected
- In perspective: communication with care-giver triggered by in-house user help-request

Once, we have provided a general overview of TI's master use scenario, it is time to select a particular one. We call this application case **"Computer Vision Enabled Smart Environments"**, which will be capable of being interconnected with other modules developed inside or outside of the project. Based on face-recognition events identified on a video stream coming from a Webcam, the system will be able to trigger a call to a specific set of user outside of the house. On top of this, different scenarios could be implemented into applications under development into the innovation department of TI:

Exemplars:

A 8-10 years old kid, home alone for a short while, could go in front of the webcam and once his/her face is detected a call can be set up from the house to the parent's phone:.

- No need to use the phone home-side
- Triggered Video-Control from remote (privacy respect)

The very same use case could also apply to an aged person.

Or:

Use together with a face recognition system. When an unknown face is detected, trigger a phone call. Instead if the detected face is recognized set up a call after a specific command or timeout.

- Intrusion detection
- Automatic Phone call on face recognition trigger

1. Technological objectives of the demonstrator

The demonstrator's objectives created by TIM are the following:

- **Real-time multimedia communication:**

One of the main objectives of this demonstrator is indeed Real Time Multimedia Communication. The demonstrator proposes a novel communication paradigm which sets a bridge between the Web Multimedia Communication and the traditional fixed/mobile network communication, using the access gateway as entry point.

As partially explained in the introduction, one of the main use case is to have a "fixed" point inside a house triggering a phone call to a specific target. Real Time communication is an important objective as the objective is to allow a "natural" interaction between the subjects involved in the call.

- **Complex communication:**

If on one side communication topology is not an objective, targeting this use case just a simple point to point communication topology, on the other side, the use case targets a complex objective in terms of protocol/network interoperability and communication paradigms.

One of the technological challenges of this demonstrator is to set up a bridge between Web Communication and traditional network communication, increasing somehow the audience of Web Communication services as the one involved in NUBOMEDIA.

One of the technological objectives of the demonstrator, specifically of the software module inside the home gateway is to build such a bridge.

- **Computer vision capability:**

As explained previously the demonstrator sets up a “traditional phone call” after some kind of trigger. The main objective however is to demonstrate this use case with face detection over webRTC stream as trigger.

The objective is to send a video stream exposed by a fix-positioned webcam to NUBOMEDIA, perform face detection and grab the event in order to set up a call.

- **Smartphone platforms:**

Smartphone platforms are targeted by this type of demonstrators, on one side, we have already described the objective of forwarding a phone call to a specific number, triggered by face detection, a number who could be a mobile number. On the other side, an app could be released in order to set up a WebRTC over mobile network point to point call exploiting NUBOMEDIA.

- **Security and privacy guarantees:**

Even if the application is intended to be used within the house’s walls, a factor giving reasonable security margins, further evolutions may take into account also face recognition features, which should be feasible also due the limited number of faces to be recognized (family members), such an objective could be a system evolution.

2. Requirements that the demonstrator must satisfy

TI’s demonstrator contains a number of features or requirements that must satisfy, those are the following:

- **Communication capabilities:**

1. Phone call: After a face has been detected in a WebRTC video stream, a phone call is initiated on the commuted line.

This feature has been successfully implemented and tested over TIM commercial Access Gateway currently installed in the FTTC Customer’s homes. The overall solution was also presented at FUSECO 2016 thanks to a successful collaboration with TUB.

The call was successfully forwarded to a specific phone number. The only limitation was in relation to the fact that the number should not be in any operator’s or customer black list. If for example the customer is not enabled to perform international calls, the number set should be in the same nation.

- **Real-Time multimedia Communications (RTC)**

1. Send Video Stream coming from a webcam to the platform and get notified if a face is recognized
2. Set up a one to one call in order to check the webcam stream if needed.

This is the feature where NUBOMEDIA made a huge difference in implementing the service concept. A PaaS Instance of NUBOMEDIA was used to fulfill this requirement: the commuted line phone call cited above has to be initiated in a reasonable amount of time, forcing real time communication between the webcam and the platform to be very important. Moreover the same stream video should be accessed from the person receiving the phone call, in case he/she is need to check the video stream.

- **Computer vision capabilities:**

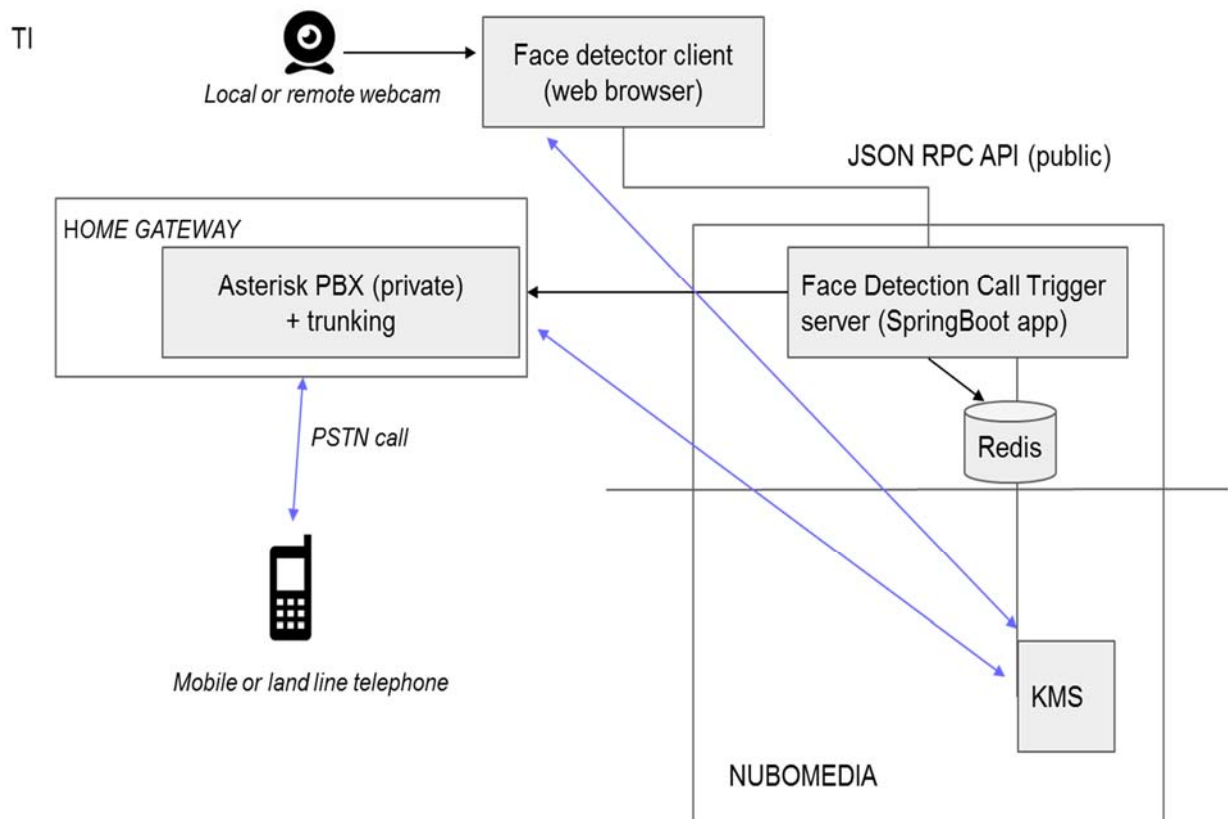
1. A specific NUBOMEDIA filter has to be applied to the real time video stream and be able to detect a face, and generate the corresponding event.

As for the first requirement above, this has been tested and demonstrated at FUSECO 2016. Also with this point, NUBOMEDIA gives a strong help by providing a quick and functional face detection filter. Moreover, this is the point where the interaction between the access gateway and NUBOMEDIA is stronger, the point where the bridge between traditional call and Web communication is built.

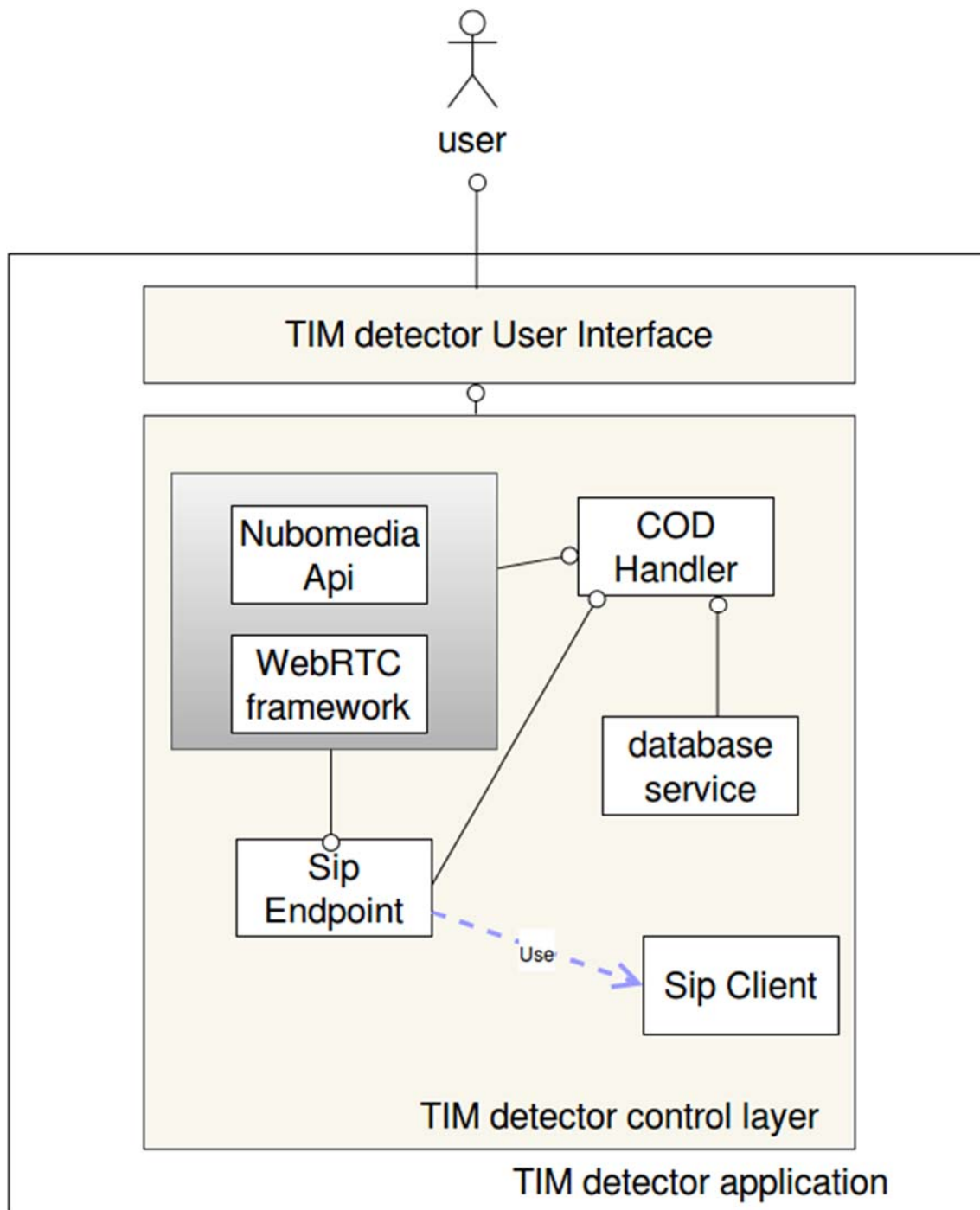
3. Architecture of the demonstrator (related to NUBOMEDIA architecture)

The following diagram shows the overall architecture of the demonstrator and the interaction with the system modules.

From the components point of view:



From an UML diagram point of view:



This is a web application, and therefore it follows a client-server architecture. At the client-side, the logic is implemented in **JavaScript** using AngularJS framework. At the

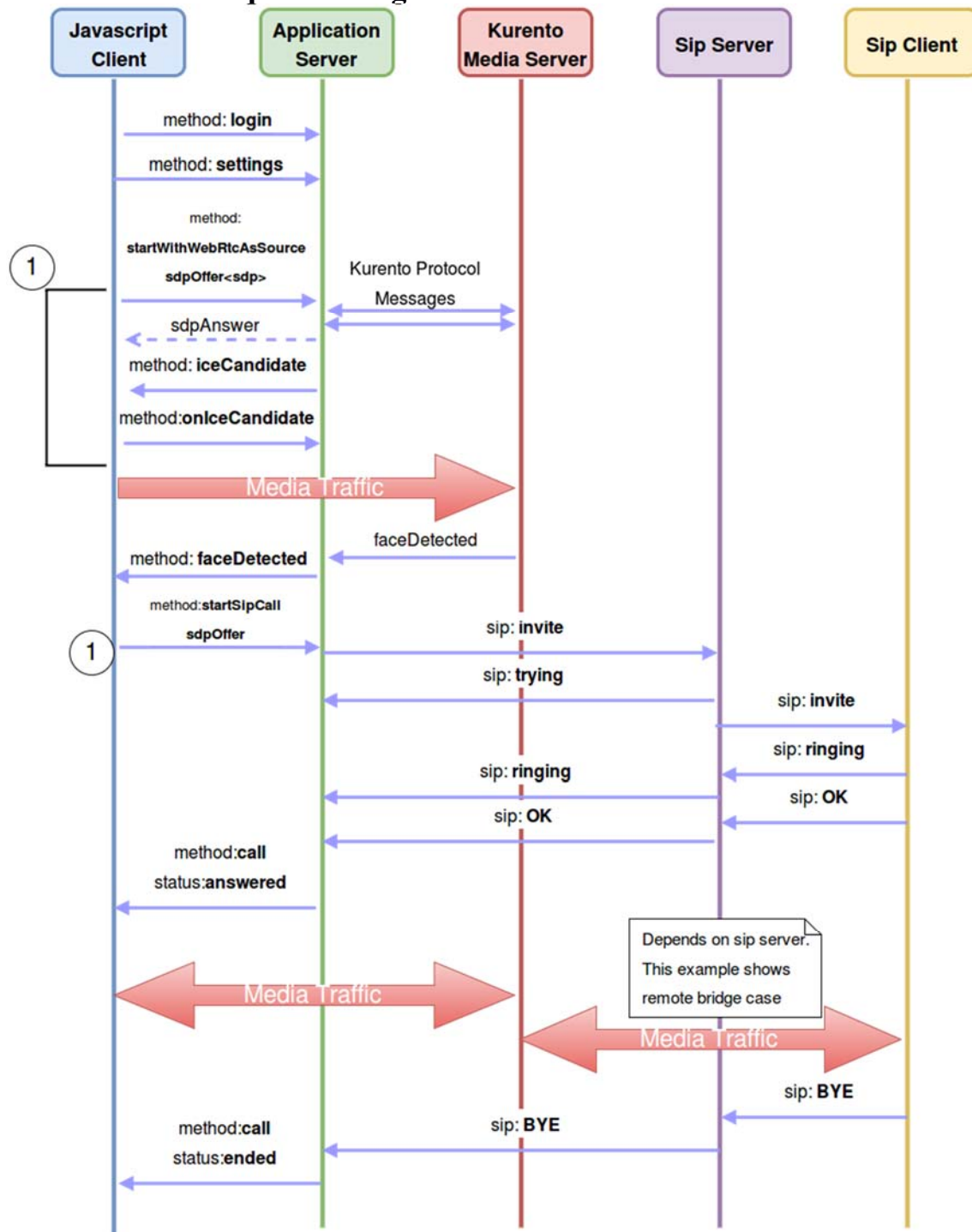
server-side, we use a Spring-Boot based server application consuming the **Nubomedia Java Client API**, to control **Kurento Media Server** capabilities. All in all, the high level architecture of this demo is three-tier. To communicate these entities two WebSockets are used. First, a WebSocket is created between client and server-side to implement a custom signaling protocol over JSONRPC 2.0. Second, another WebSocket is used to perform the communication between the Kurento Java Client and the Kurento Media Server. This communication is implemented by the **Kurento Protocol**.

The signaling between the Application Server and the sip Sip server is done using SIP protocol.

To communicate with the Sip Server a sip client is implemented on the application server using JAIN SIP library.

The sequence diagram below details the messages between all the actors involved into this demonstrator.

Demonstrator Sequence Diagram

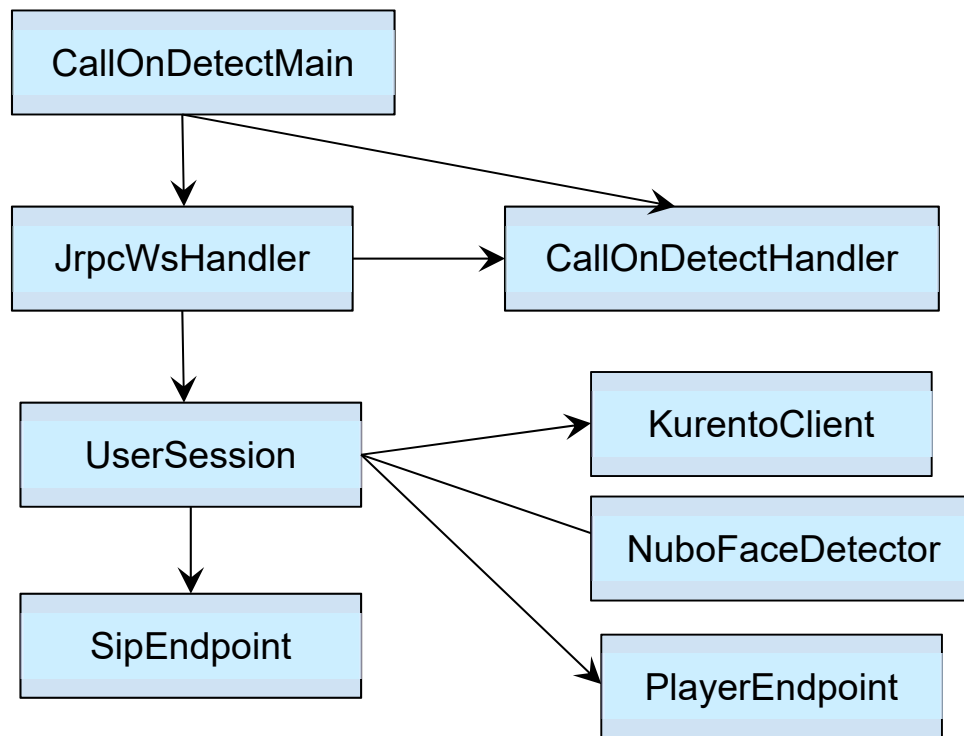


4. Relevant implementation details

Application Server Logic

This demonstrator has been developed using Java in the server-side, based on the Spring Boot framework, which embeds a Tomcat web server within the generated maven artifact, and thus simplifies the development and deployment process. To simplify the development/deployment phases even more the Nubomedia APIs have been used to communicate with one or more Kurento Media Servers.

In the following figure it is possible to see a class diagram of the server side code:



This web application follows a Single Page Application architecture (SPA), and uses JsonRPC over WebSocket to communicate client with application server by means of requests and responses.

The main class of this demonstrator is named **CallOnDetectMain**. This class imports JsonRpcConfiguration class configurations with Spring annotation *@Import* in order to add support for JsonRPC protocol. It also implements JsonRpcConfigurer to process JsonRPC requests.

The central piece of this class is the method `registerJsonRpcHandlers`. This method implements the actions for requests on path `/callOnDetect`, returning responses through the WebSocket.

The main class instantiates two Spring Beans:

- **JrpcWsHandler:** It is a wrapper class that provides a syntactic sugar through annotations to define JsonRPC endpoints (e.g. `@JrpcMethod(name = "login")`). It is in charge also to throw an error if the called endpoint does not exists. This class extends `DefaultJsonRpcHandler` to handle JsonRPC requests.
- **CallOnDetectHandler:** `CallOnDetectHandler` class implements `JrpcEventListener` to handle the close of the websocket connection in order to

release all the objects created during the session properly. This is the class that handles all the requests to the server.

In the designed protocol there are different kinds of incoming messages to the Server:

- register
- login
- settings
- startWithWebRtcAsSource
- startWithRtspPlayerAsSource
- startSipCall
- onIceCandidate
- onIceCandidateRec
- stop

In the **login** method the `UserSession` is created while in the **settings** method are saved the settings related to the `sipHost` and the `destUser`:

- **sipHost** defines the *url:port* of the sip server
- **destUser** defines the destination user telephone number in case the sip server is connected to a trunk, the extension of the sip server on which the user is reachable otherwise

The **startWithWebRtcAsSource** method handles the ICE candidates gathering, creates a Media Pipeline, creates the Media Elements (`WebRtcEndpoint`, `NuboFaceDetector`) and make the connections among them. It creates also the `SipEndpoint` through which the server register the user on the Sip Server. A `sdpAnswer` response is generated and sent as response. In case the registration against the Sip Server fails an error is returned as a notification to the client. If everything goes fine from this point when the Application Server will receive from the `NuboFaceDetector` an `OnFaceEvent` a *faceDetected* event will be sent to the client.

The **startWithRtspPlayerAsSource** method works the same as the previous one. The only difference is that the media source in this case it is not the webcam of the user PC but it is an IP WebCam. To handle the IP Webcam video a `PlayerEndpoint` is created and associated to the `UserSession`, the `rtspUrl` to reproduce is taken from the user settings and the player is started with *play()* method.

Once a face is detected the client will call the **startSipCall** method. This method handles the ICE candidates gathering, creates another `WebRtcEndpoint` to receive the media from sip world (it cannot be used the previous `WebRtcEndpoint` since this demonstrator supports also the IP webcam so in that case the `WebRtcEndpoint` is already receiving a media stream), and issues a sip call by calling `sipEndpoint.startSipCall` method.

Subsequently this method registers two listeners:

- a **call listener**: to handle the answer/decline to the call
- a **call ended listener**: in case the call started successfully and after the destination user hangs up. A timer is setted to avoid triggering a new face detected event after the call is ended.

The **afterConnectionClosed** method is called once the websocket connection is closed. The behaviour of this method is similar to the **stop** method. In both cases the

userSession is closed and removed from the map holding the active sessions and the sip endpoint is released. In case a sip call is active it will be stopped.

Client Side

Let's move now to the client-side of the application. The Single Page Application (SPA) was implemented using AngularJS framework.

To call the previously created WebSocket service in the server-side, we use the JavaScript class WebSocket wrapped in an angular Service. In this way it is possible to register different listeners on different methods on the websocket channel that get notified once a message is received with a specified method or response id.

We use a specific Kurento JavaScript library called kurento-utils.js to simplify the WebRTC interaction with the server. This library depends on adapter.js, which is a JavaScript WebRTC utility maintained by Google that abstracts away browser differences.

All the needed libraries are linked in the index.html web page. The web application is composed by five modules

- **login:** the module that handles the login phase
- **registration:** the module that handles the registration phase
- **settings:** the module that handles the setting phase. Here can be defined the destination user for the sip call and chose where to use the local webcam for the media stream or to visualize the media stream of a remote webcam as a source for the face detection
- **video:** this is the modules that hosts the directive that handle the video and displays the instructions for the first access.
- **codVideo:** this is the main directive that handles the video stream and the sip call events.

To move between pages the *ngRoute* module is used. The *ngRoute* module routes the application to different pages without reloading the entire application.

All these modules are imported from the main module *cod* in the main file **app.js**.

Experienced Issues

The demo contains a module called SipEndpoint. This module is in charge to allow the connection between the sip and some kurento endpoints (in the demo webrtc and player endpoint).

It connects to the sip server as a normal sip client and manipulate and forwards the requests to sip through sip signaling.

In particular the SDP offer from Kurento Media Server has to be manipulated by the SipEndpoint in order to work with asterisk.

Right now two section of the SDP offer have been modified by the SipEndpoint and are highlighted on this SDP offer example:

```
v=0
o=- 3688973018 3688973018 IN IP4 192.168.1.52
s=Kurento Media Server
c=IN IP4 192.168.1.52
```



```

t=0 0
m=audio 37748 RTP/AVP 96 0 97
a=extmap:3 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
a=rtpmap:96 SPEEX/16000
a=rtpmap:97 AMR/8000
a=mid:audio0
a=ssrc:103624916 cname:user702861291@host-3768eb2d
m=video 21508 RTP/AVP 102 103
a=extmap:3 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
a=rtpmap:102 VP8/90000
a=rtpmap:103 H264/90000
a=mid:video0
a=rtcp-fb:102 nack
a=rtcp-fb:102 nack pli
a=rtcp-fb:102 ccm fir
a=rtcp-fb:103 nack
a=rtcp-fb:103 nack pli
a=rtcp-fb:103 ccm fir
a=ssrc:2757220289 cname:user702861291@host-3768eb2d

```

The *m* property is modified if the sip server does not support AVPF.

The *c* property has been modified to solve an issue detected once the demo was deployed on the NUBOMEDIA PaaS.

The issue experienced was the missing audio flow from the sip client to the browser. The root cause of this issue was that the SipEndpoint did not replaced the IP address provided by the KMS in the SDP offer with the public IP address of the machine hosting the KMS.

The issue was solved by getting the KMS URL using the NUBOMEDIA APIs:

```

String kmsId = UUID.randomUUID().toString();
kmsUrl = KurentoClient.getKmsUrl(kmsId, new Properties());
kurentoClient = KurentoClient.create(kmsUrl);

```

From the KMS url it is possible to extract the KMS ip and then substitute it on the SDP offer.

5. Testing mechanisms and methodologies

The testing part of this demonstrator has been divided in two parts:

1. A first part in which the single functionalities are tested (Unit Tests). Our tests have been made using JUNIT Framework version 4 (<http://junit.org>). JUnit is a **Regression Testing Framework** used by developers to implement unit testing in Java, and accelerate programming speed and increase the quality of code. Regarding the demonstrator these tests are located in the module **sip_endpoint** and **are** related to the utilities classes that this module exposes. These kind of tests are related mostly on network functionalities methods (testGetKmsIpFromUrl, testIsPrivateV4OrLoopbackAddress, testGetFisrtFreePort etc..).

2. A second part involves the testing of the sip client implementation developed to add sip signaling capabilities to the SipEndpoint. To test the signaling part sipUnit framework has been evaluated. SipUnit provides a test environment geared toward unit testing SIP applications. It extends the JUnit test framework to incorporate SIP-specific assertions, and it provides a high-level API for performing the SIP operations needed to interact with or invoke a test target. (<https://github.com/RestComm/sipunit>). In this case the tests are related to test if the client sends the right sip messages to the sip server and if it behaves as expected when receives BUSY or BYE messages from the Sip Server.

Anyway, to test NUBOMEDIA APIs and the interaction with Asterisk at media level we don't have specific test because is quite hard to check programmatically if everything in a is going well.

For example we cannot check programmatically if our audio/video is freeze, or the received audio/video quality to the end client, or maybe if a remote sip client doesn't send audio properly, is really difficult to know if there was a problem of the API, or the peer, the sip server or something else. Because of that we wrote a list of basic functionalities a the demonstrator must achieve, and in every release of the APIs we made that test manually to ensure everything was working properly.

6. End-user's guide and tutorial: how to install and use

Call on detect Nubomedia demonstrator

Prerequisites

To run properly this demo you need to have an installed and working instance of the following components:

- KMS (v6.5.0)
- The [nubomedia face detector plugin] installed on the KMS
- Asterisk or other PBX

Installation instructions

Be sure to have installed Java8

(<http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html>)

Maven (<https://maven.apache.org/download.cgi>) and Bower (<http://bower.io>) in your system:

```
git clone https://github.com/nubomediaTI/timedetector-demonstrator.git
cd timedetector-demonstrator
```

Call On Detect (COD) configuration

Get the application.properties template and make a copy:

```
cp call_on_detect/src/main/resources/application.properties .
```

Open it with your favourite text editor. You will see the following properties:

```
server.port: 8443                #the port on which the server will listen
server.ssl.key-store: keystore.jks  #keystore file
server.ssl.key-store-password: kurento
server.ssl.keyStoreType: JKS
server.ssl.keyAlias: kurento-selfsigned

serverEvents.timer.milliseconds=6000    #interval from a face recognition event and
another from the server

#stun server address
sip.stunServer=stun.l.google.com:19302
#asterisk host address
```

```
sip.host=showdemos2.ddns.net #sip server host address to which the client will
connect
sip.port=5060                #sip server port
#interface from which take ip address (if more than one interface connected)
#sip.listenOnInterface=
```

replace the **sip.host** property with the name/address of your PBX server.
replace the old *application.properties* or place this file on the same folder you will launch the application.

Running the application

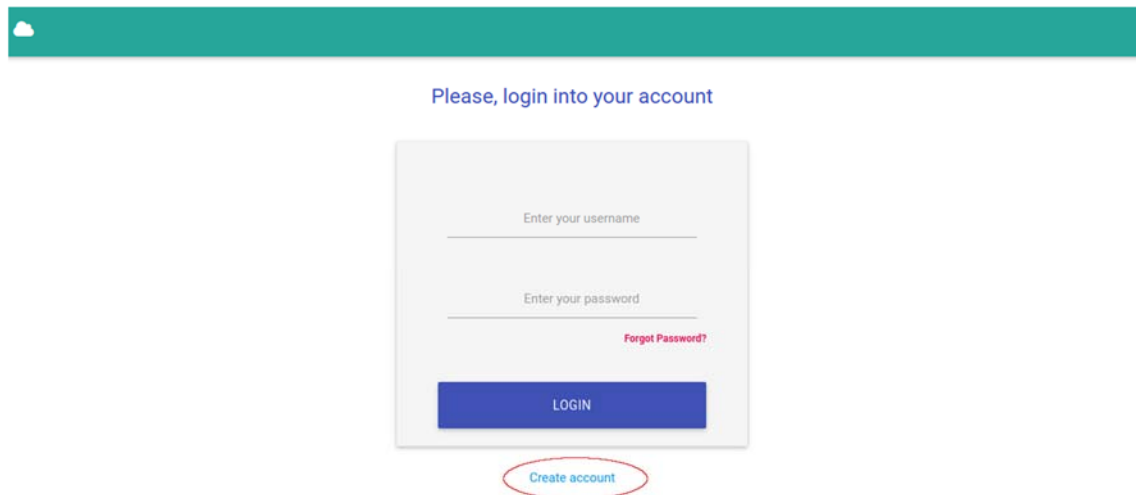
Run from your terminal the following commands:

```
mvn clean package -DskipTests
cd call_on_detect
java -jar target/call_on_detect-6.1.0.jar
```

Pay Attention: if you change the location of the jar file be sure that the file keystore.jks is in the same folder you launch the application.

How to use the DEMO

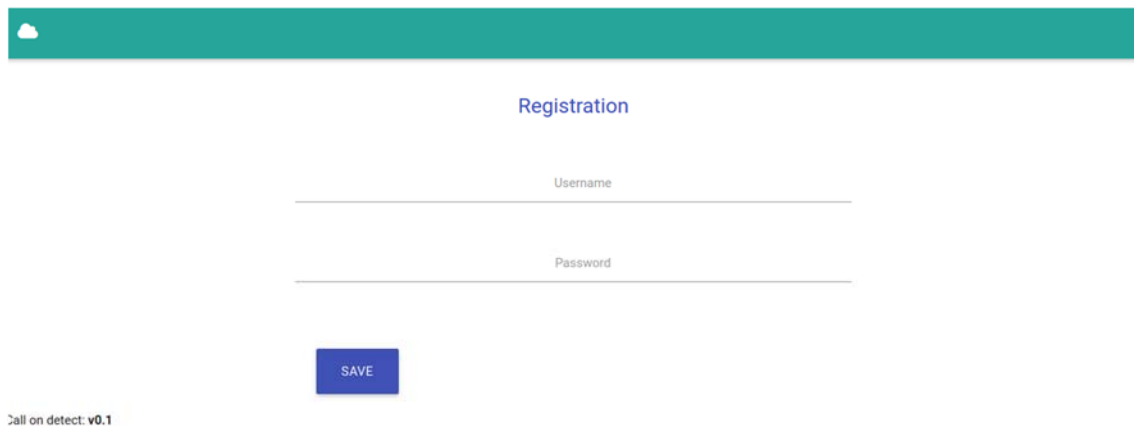
Once the server starts go to the page <http://localhost:8443>. The following page will be displayed:



The image shows a web application interface. At the top, there is a teal header bar with a white cloud icon on the left. Below the header, the text "Please, login into your account" is displayed in blue. In the center, there is a light gray rectangular box containing a login form. The form has two input fields: "Enter your username" and "Enter your password". Below the password field, there is a red link that says "Forgot Password?". At the bottom of the form box is a blue button labeled "LOGIN". Below the form box, there is a red oval button labeled "Create account".

click on "Create account".

The registration screen will be displayed:



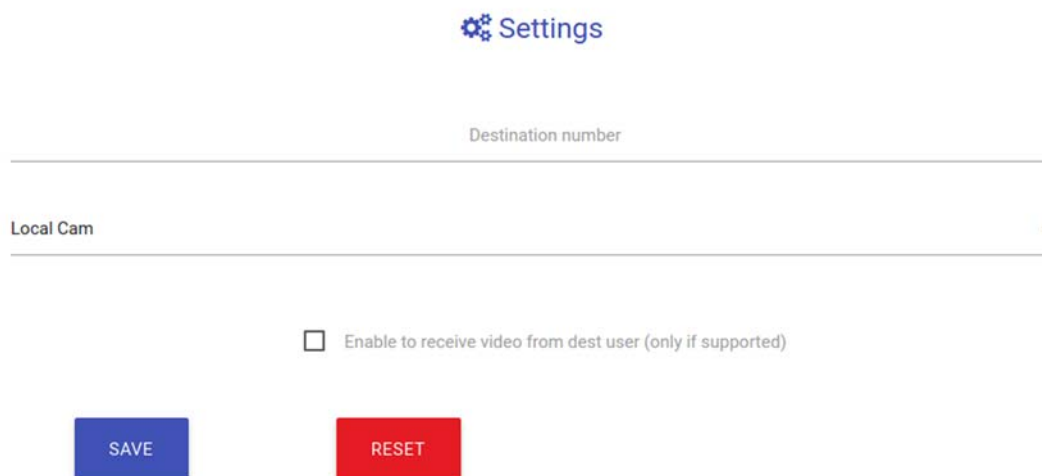
The registration form is displayed on a teal background. It features a title 'Registration' in blue. Below the title are two input fields: 'Username' and 'Password'. A blue 'SAVE' button is positioned below the password field. At the bottom left, there is a small text label 'Call on detect: v0.1'.

Insert username and password and click “save”.

NB: The username and password chosen must be the same as the one you use to authenticate the user on your sip server.

Once registered the user will be redirect to the login page. Provide the username and password chosen during the registration phase and click on *login*.

Once logged, the user will be redirect to settings page:



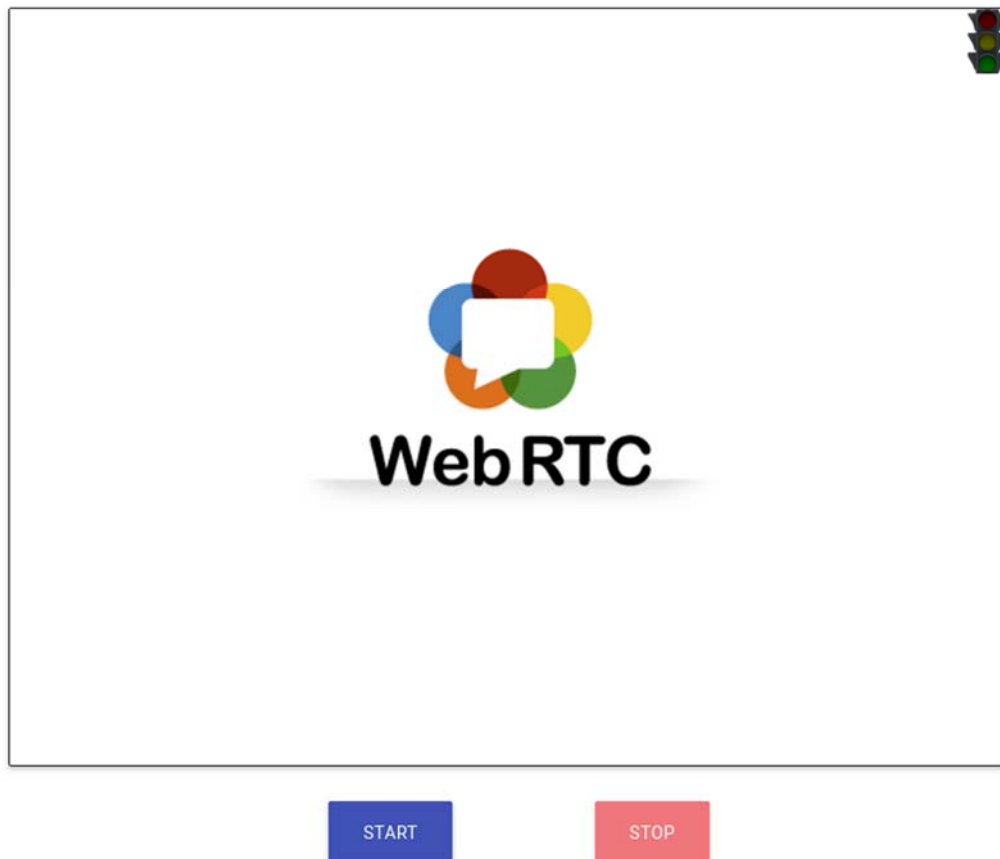
The settings form is displayed on a white background. It features a title 'Settings' in blue with a gear icon. Below the title is a 'Destination number' input field. Underneath is a 'Local Cam' dropdown menu. At the bottom, there is a checkbox labeled 'Enable to receive video from dest user (only if supported)'. Below the checkbox are two buttons: a blue 'SAVE' button and a red 'RESET' button.

Set the destination number as the number of the extension of the destination user or the user telephone number if trunking is enabled on your sip server.

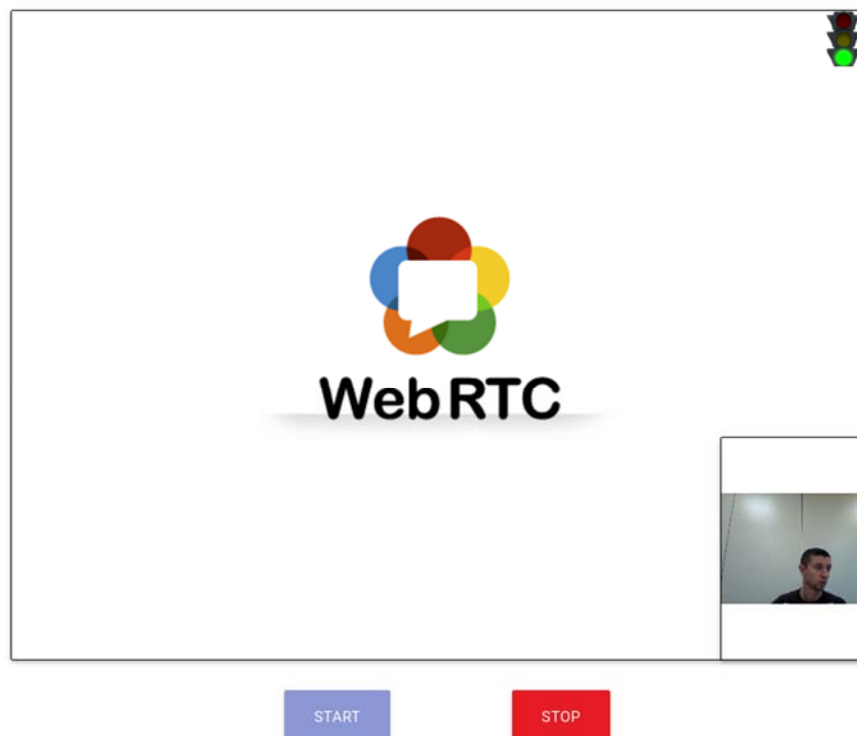
If using a sip client register the correspondent user to the sip server in order to be available for the call.

Check “*Enable to receive video from dest user*” if you want to receive the video from the sip client. Check this option only if receive video option is checked on your sip client. This option doesn't work with sip client that perform a renegotiation to enable the video.

Once finished click on “*Save*” and you will be redirect to the video page:

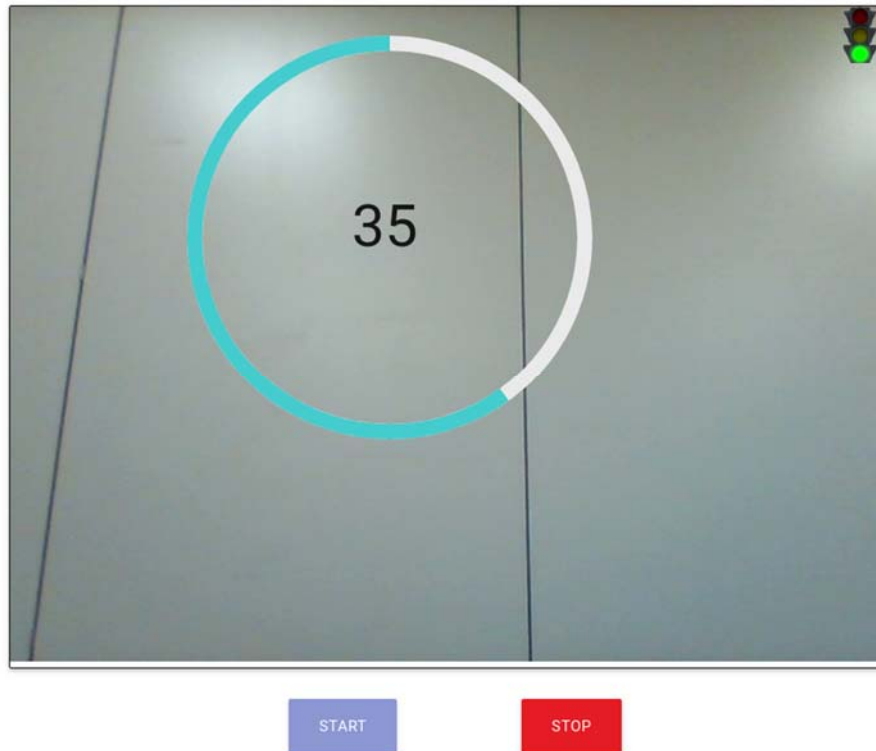


Cl¹ provided to the Sip server. If the credentials are valid and everything goes fine you will see a green light on the traffic light. Now the system is ready for the face detection. Put your face in front of the camera and wait until the system detect your face triggering the call:



The sip client should ring. If the user does not answer or is busy an error message will notify the user.

Answer and the call will start. Once finished hang-up from the mobile device. The User will see a progress bar with a timer:



Once the timer will expire you can start a new call triggered by the face detection. This behavior is necessary to avoid that once finished the call a new face detection triggers immediately a new call. Once finished click on the stop button.

7. Total effort consumed in creating the demonstrator

Effort type	Person/Month effort
Implementation effort	4
Debug effort	2
Other tasks related to the demonstrator effort	1

8. Developer's feedback in relation to using NUBOMEDIA

The NUBOMEDIA Paas allowed to put in place a demonstrator/prototype for this service concept without the need to re-implement/re-engineer any specific video-communication or face-recognition product.

Even the mobile one to one communication, useful for the “outside-home” person to check the situation inside the house was actually provided by NUBOMEDIA.

The development process was then much quicker with respect to the effort which could have been required to build a similar prototype without NUBOMEDIA.

From a Technical stance, the main effort was related to the development of the SIP Trunking part running on the access-gateway, a part which is by design in the operator’s domain.

The Mobile APIs allowing to set up communication among peers are enough easy to use and provide all of the features needed to implement this demonstrator.

A possible drawback could be related to network configuration, requiring to understand well the set of TCP ports to be left available, but this drawback is much less significant with respect to the benefits that the platform is able to provide.