Kurento: The WebRTC Modular Media Server

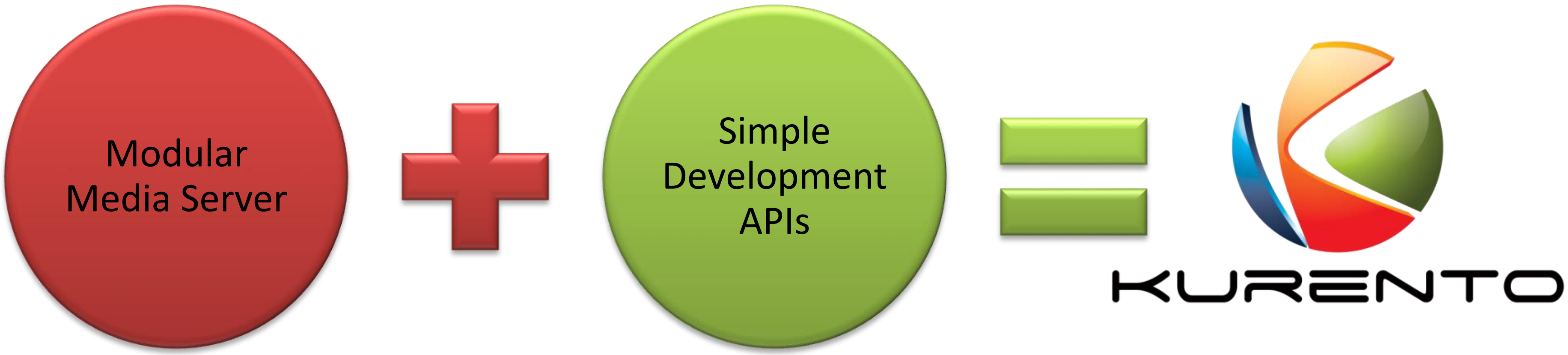
Boni García

boni.garcia@urjc.es

Universidad Rey Juan Carlos

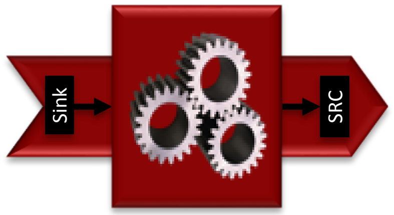
Multimedia development implies **complexity**

The Kurento formula



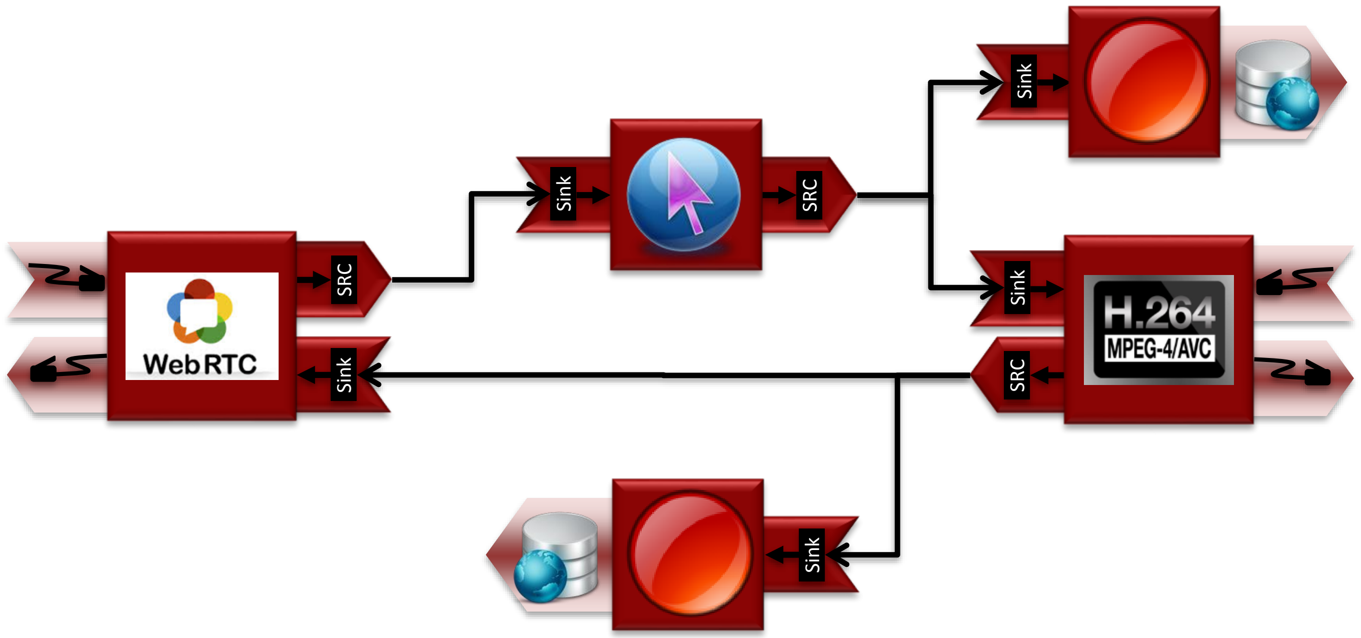
Key concepts: Media elements and Pipelines

- Media Element
  - Provides a specific media functionality
    - Send/receive media. These are the **Endpoints**
    - Process media
    - Transform media
  - Ready to be used
  - New media elements can be added



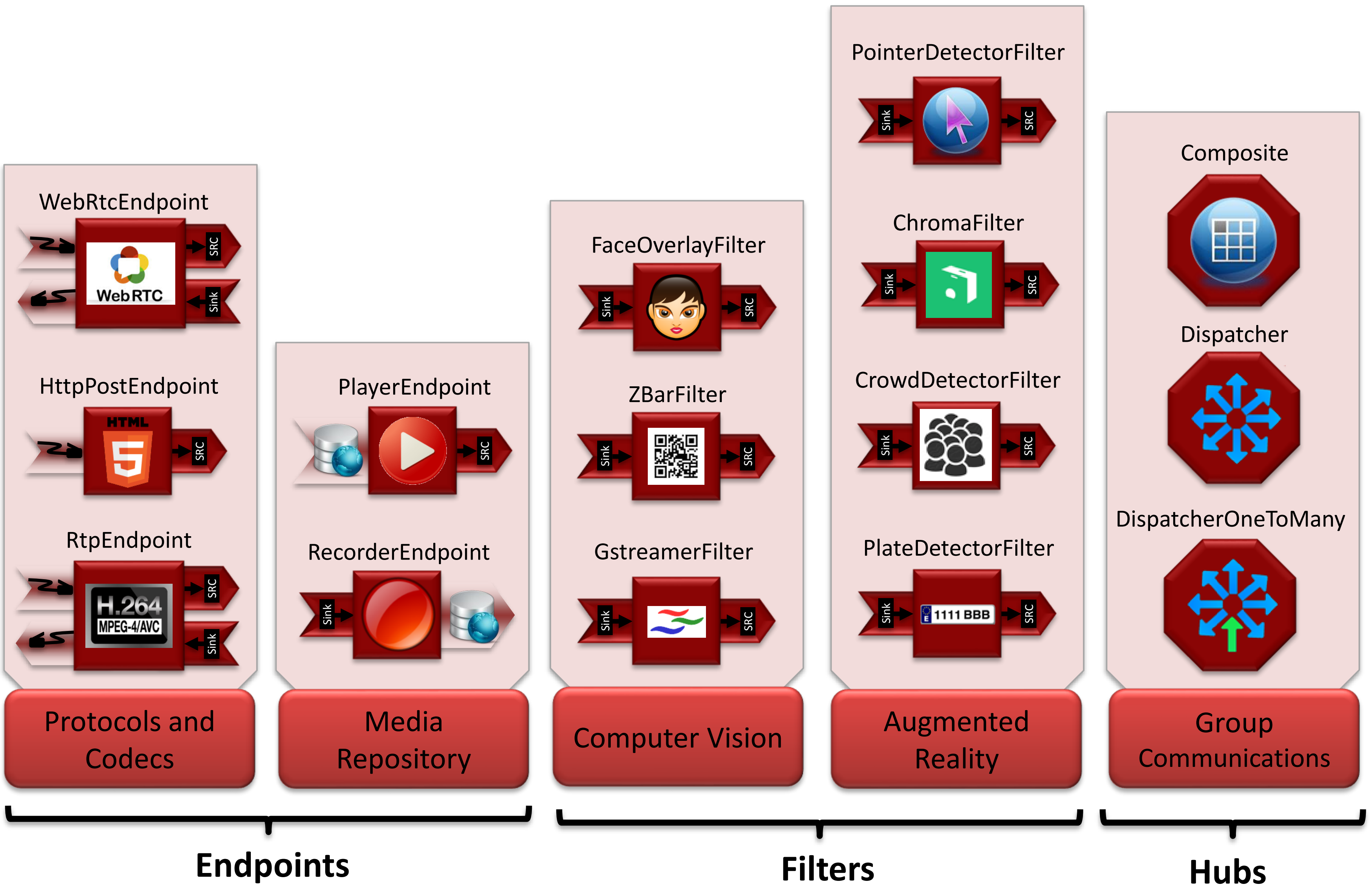
Key concepts: Media elements and Pipelines

- Media Pipeline
  - Chain of media elements implementing the desired media logic
  - The Media Server provides the capability of creating media pipelines by joining media elements of the toolbox



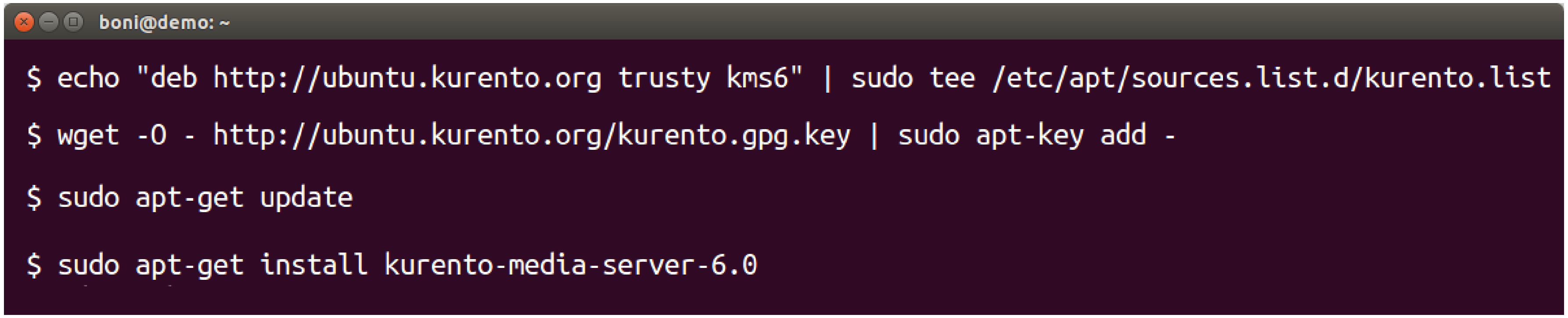
WebRTC
SRC
Sink
Sink
SRC

Kurento Toolbox



KURENTO

ubuntu 14.04 LTS

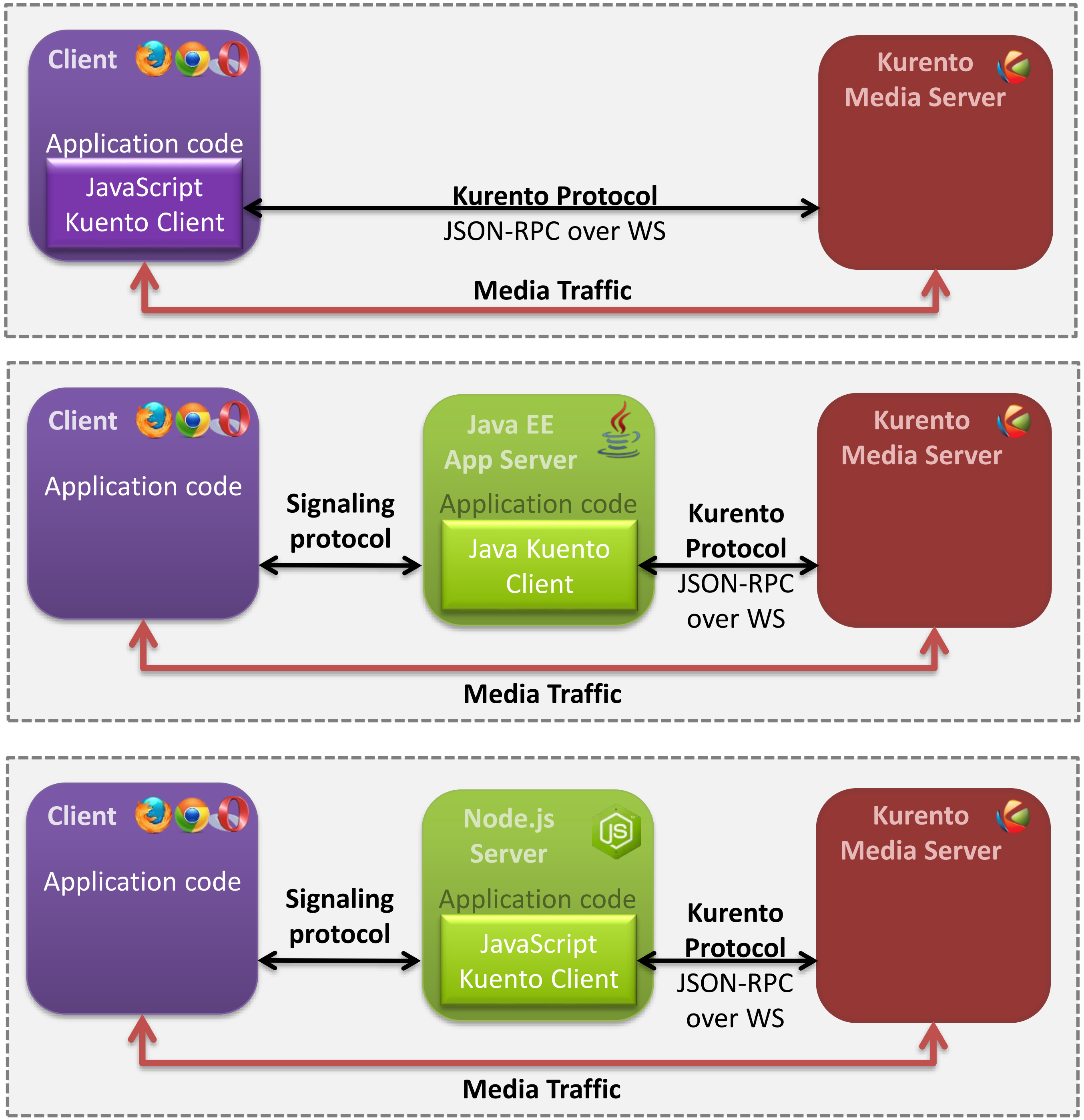


```
$ echo "deb http://ubuntu.kurento.org trusty kms6" | sudo tee /etc/apt/sources.list.d/kurento.list

$ wget -O - http://ubuntu.kurento.org/kurento.gpg.key | sudo apt-key add -

$ sudo apt-get update

$ sudo apt-get install kurento-media-server-6.0
```


Client
Application code
JavaScript Kuento Client
Kurento Protocol
JSON-RPC over WS
Kurento Media Server
Media Traffic
Client
Application code
Signaling protocol
Java EE App Server
Application code
Java Kuento Client
Kurento Protocol
JSON-RPC over WS
Kurento Media Server
Media Traffic
Client
Application code
Signaling protocol
Node.js Server
Application code
JavaScript Kuento Client
Kurento Protocol
JSON-RPC over WS
Kurento Media Server
Media Traffic

Kurento Development with Java

Java

```java
@Autowired
private KurentoClient kurentoClient;

MediaPipeline pipeline = kurentoClient.createMediaPipeline();
WebRtcEndpoint webRtcEndpoint = new WebRtcEndpoint.Builder(pipeline)
.build();
FaceOverlayFilter faceOverlayFilter = new FaceOverlayFilter.Builder(
pipeline).build();
webRtcEndpoint.connect(faceOverlayFilter);
faceOverlayFilter.connect(webRtcEndpoint);
```

pom.xml

```xml
<dependencies>
   <dependency>
      <groupId>org.kurento</groupId>
      <artifactId>kurento-client</artifactId>
      <version>6.6.0</version>
   </dependency>
   <dependency>
      <groupId>org.kurento</groupId>
      <artifactId>kurento-utils-js</artifactId>
      <version>6.6.0</version>
   </dependency>
</dependencies>
```

Kurento Development with JavaScript for browser

JavaScript

```javascript
kurentoClient.create("MediaPipeline", function(error, pipeline) {
   pipeline.create('WebRtcEndpoint', function(error, webRtc) {
      if (error) return onError(error);
         pipeline.create('FaceOverlayFilter', function(error, filter) {
            if (error) return onError(error);
            webRtc.connect(filter, function(error) {
            if (error) return onError(error);
               filter.connect(webRtc, function(error) {
               if (error) return onError(error);
            });
         });
      });
   });
});
```

bower.json

```json
"dependencies": {
   "kurento-client": "6.6.0",
   "kurento-utils": "6.6.0"
}
```

Kurento Development with JavaScript for Node.js

JavaScript

```
kurentoClient.create("MediaPipeline", function(error, pipeline) {
   pipeline.create('WebRtcEndpoint', function(error, webRtc) {
      if (error) return onError(error);
         pipeline.create('FaceOverlayFilter', function(error, filter) {
            if (error) return onError(error);
            webRtc.connect(filter, function(error) {
            if (error) return onError(error);
               filter.connect(webRtc, function(error) {
               if (error) return onError(error);
            });
         });
      });
   });
});
```

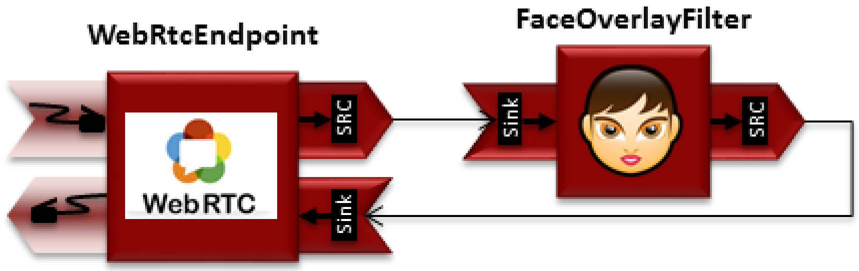
bower.json

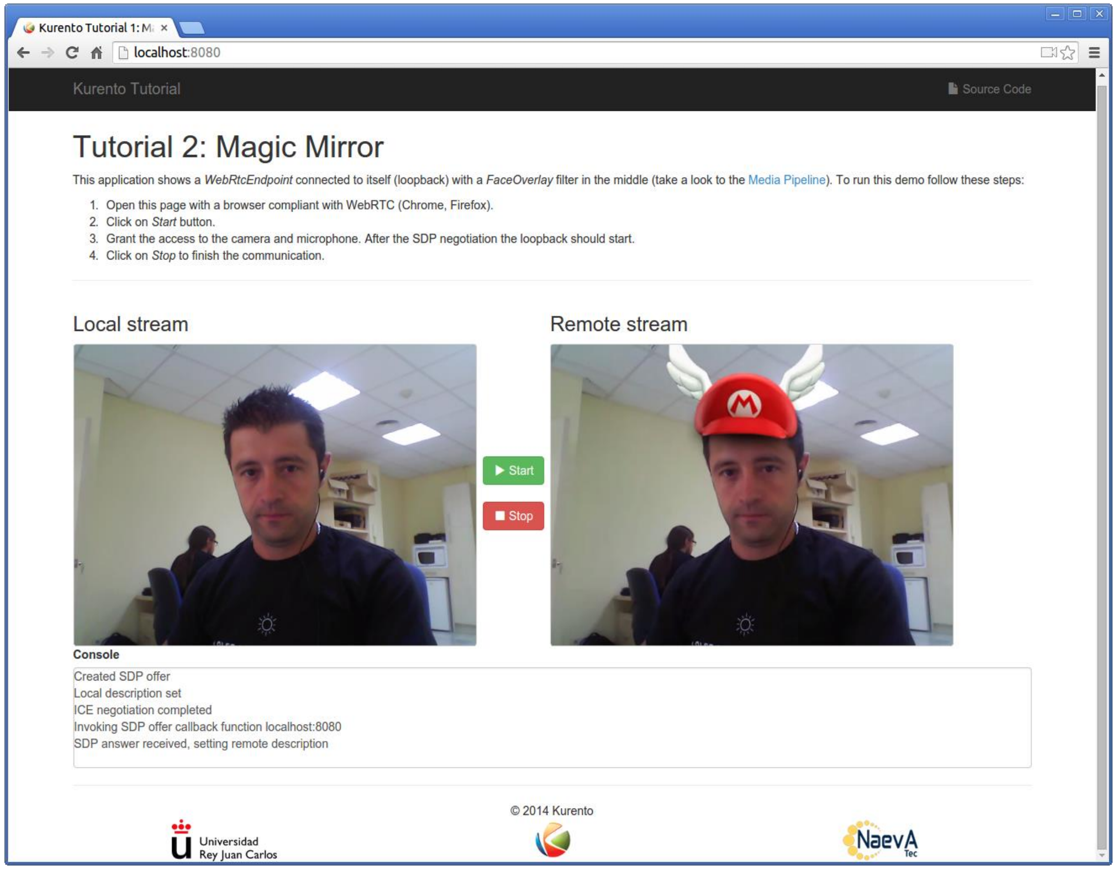
```
"dependencies": {
   "kurento-utils": "6.6.0"
}
```

package.json

```
"dependencies": {
   "kurento-client": "6.6.0"
}
```

“Magic Mirror” Example





Tutorial 2: Magic Mirror

This application shows a *WebRtcEndpoint* connected to itself (loopback) with a *FaceOverlay* filter in the middle (take a look to the Media Pipeline). To run this demo follow these steps:

1. Open this page with a browser compliant with WebRTC (Chrome, Firefox).
2. Click on *Start* button.
3. Grant the access to the camera and microphone. After the SDP negotiation the loopback should start.
4. Click on *Stop* to finish the communication.

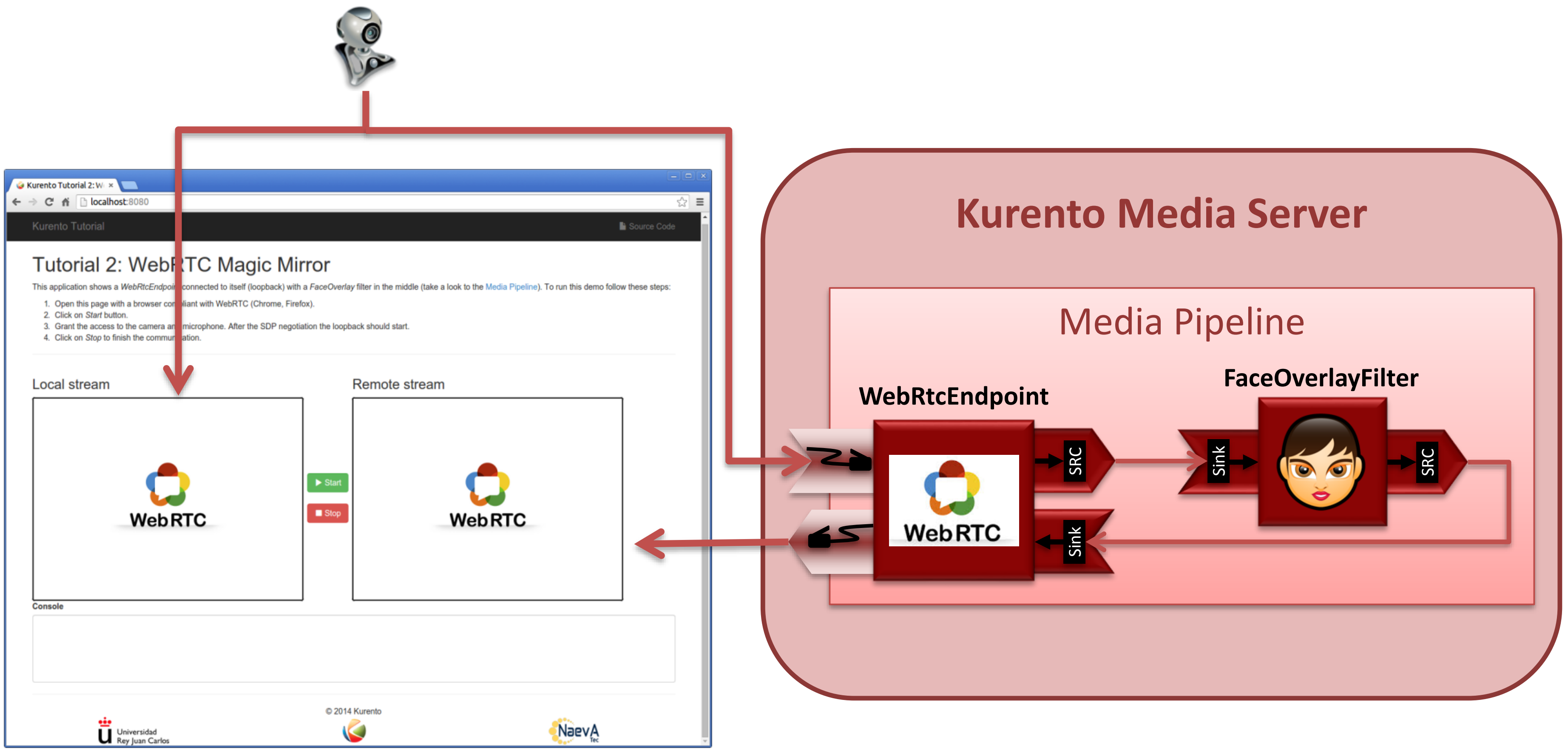
Local stream

Remote stream

Console

```
Created SDP offer
Local description set
ICE negotiation completed
Invoking SDP offer callback function localhost:8080
SDP answer received, setting remote description
```

Magic Mirror Media Pipeline

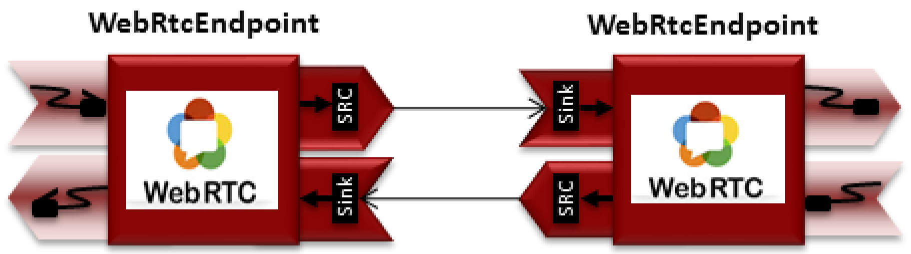


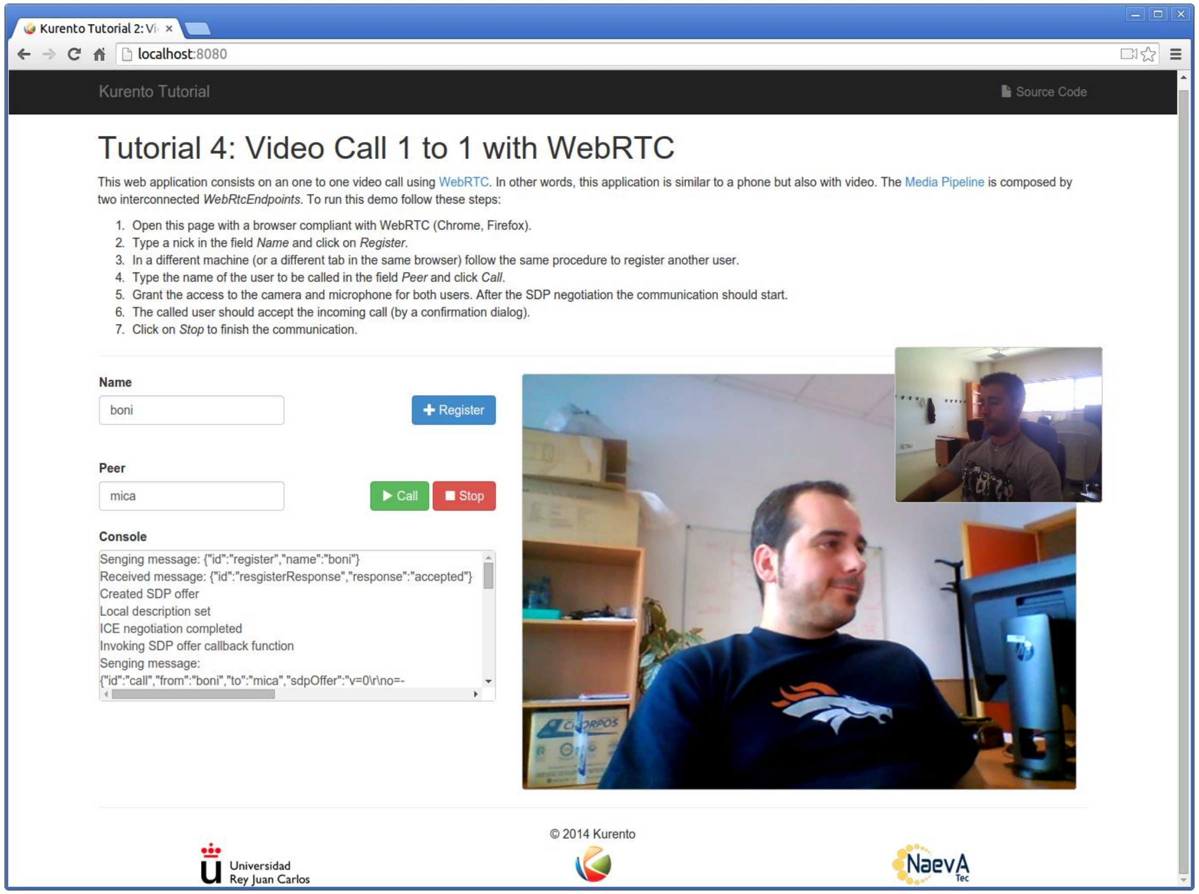
Tutorial 2: WebRTC Magic Mirror

This application shows a *WebRtcEndpoi* connected to itself (loopback) with a *FaceOverlay* filter in the middle (take a look to the Media Pipeline). To run this demo follow these steps:

1. Open this page with a browser co liant with WebRTC (Chrome, Firefox).
2. Click on *Start* button.
3. Grant the access to the camera an microphone. After the SDP negotiation the loopback should start.
4. Click on *Stop* to finish the commu ation.

Video call one to one





Tutorial 4: Video Call 1 to 1 with WebRTC

This web application consists on an one to one video call using WebRTC. In other words, this application is similar to a phone but also with video. The Media Pipeline is composed by two interconnected *WebRtcEndpoints*. To run this demo follow these steps:

1. Open this page with a browser compliant with WebRTC (Chrome, Firefox).
2. Type a nick in the field *Name* and click on *Register*.
3. In a different machine (or a different tab in the same browser) follow the same procedure to register another user.
4. Type the name of the user to be called in the field *Peer* and click *Call*.
5. Grant the access to the camera and microphone for both users. After the SDP negotiation the communication should start.
6. The called user should accept the incoming call (by a confirmation dialog).
7. Click on *Stop* to finish the communication.

Name: boni

Peer: mica

Console

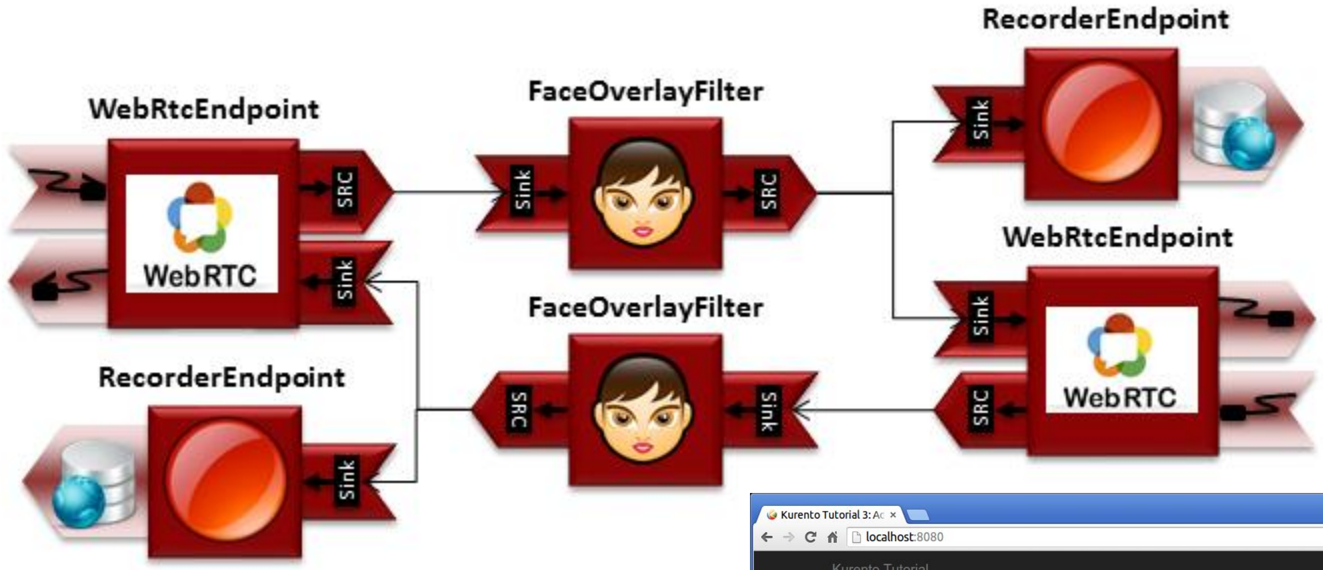
```
Senging message: {"id":"register","name":"boni"}
Received message: {"id":"resgisterResponse","response":"accepted"}
Created SDP offer
Local description set
ICE negotiation completed
Invoking SDP offer callback function
Senging message:
{"id":"call","from":"boni","to":"mica","sdpOffer":"v=0\r\no=-
```

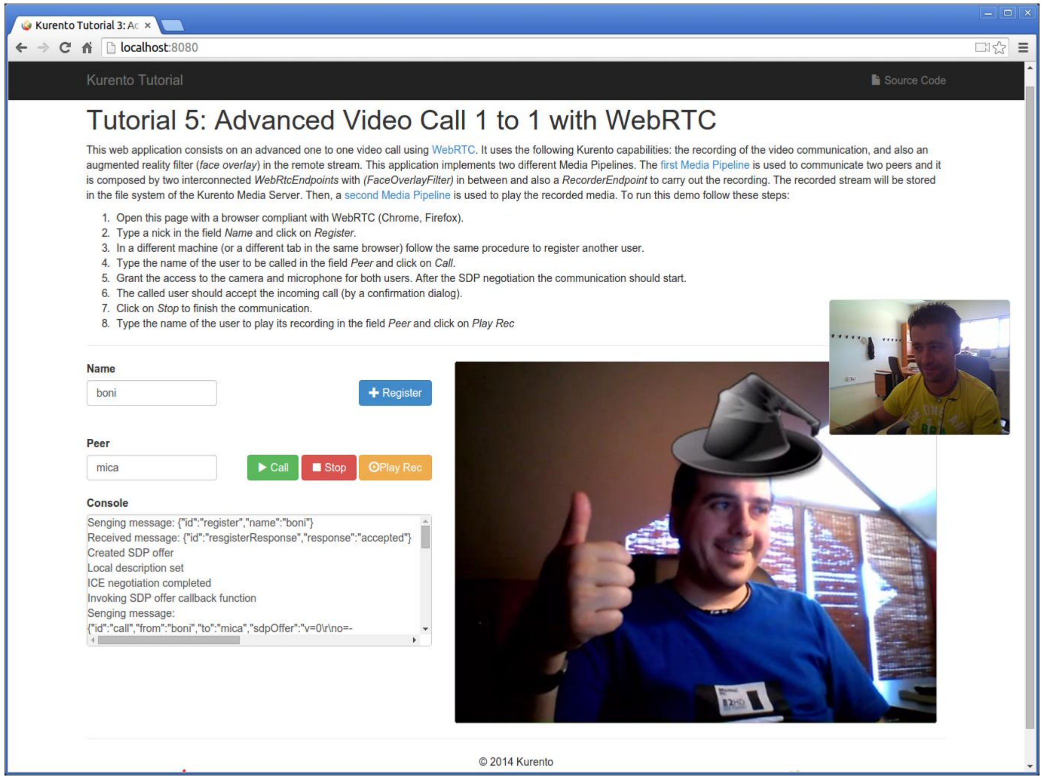
© 2014 Kurento

Universidad Rey Juan Carlos

NaevA Tec

Advanced video call one to one





Tutorial 5: Advanced Video Call 1 to 1 with WebRTC

This web application consists on an advanced one to one video call using WebRTC. It uses the following Kurento capabilities: the recording of the video communication, and also an augmented reality filter (*face overlay*) in the remote stream. This application implements two different Media Pipelines. The first Media Pipeline is used to communicate two peers and it is composed by two interconnected *WebRtcEndpoints* with *(FaceOverlayFilter)* in between and also a *RecorderEndpoint* to carry out the recording. The recorded stream will be stored in the file system of the Kurento Media Server. Then, a second Media Pipeline is used to play the recorded media. To run this demo follow these steps:

1. Open this page with a browser compliant with WebRTC (Chrome, Firefox).
2. Type a nick in the field *Name* and click on *Register*.
3. In a different machine (or a different tab in the same browser) follow the same procedure to register another user.
4. Type the name of the user to be called in the field *Peer* and click on *Call*.
5. Grant the access to the camera and microphone for both users. After the SDP negotiation the communication should start.
6. The called user should accept the incoming call (by a confirmation dialog).
7. Click on *Stop* to finish the communication.
8. Type the name of the user to play its recording in the field *Peer* and click on *Play Rec*

Name

boni

Peer

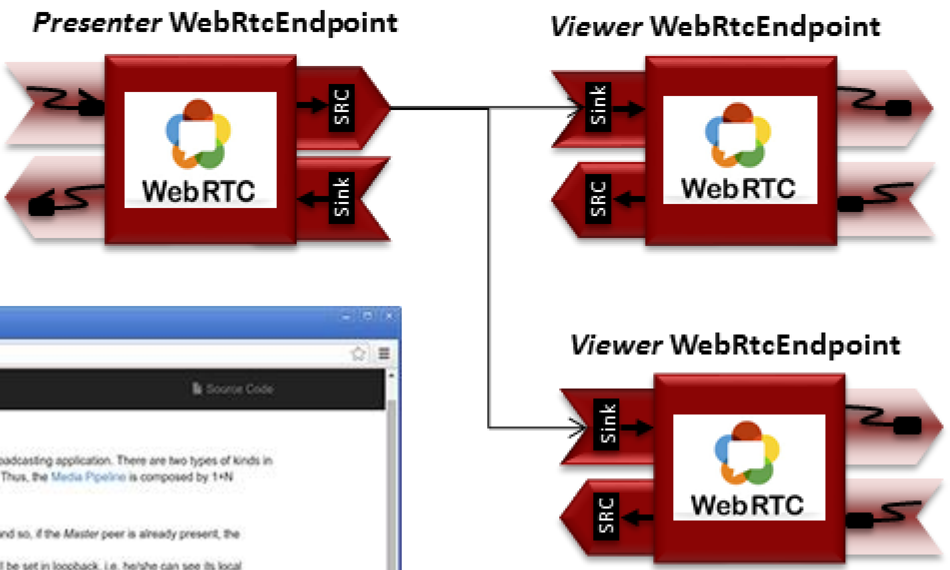
mica

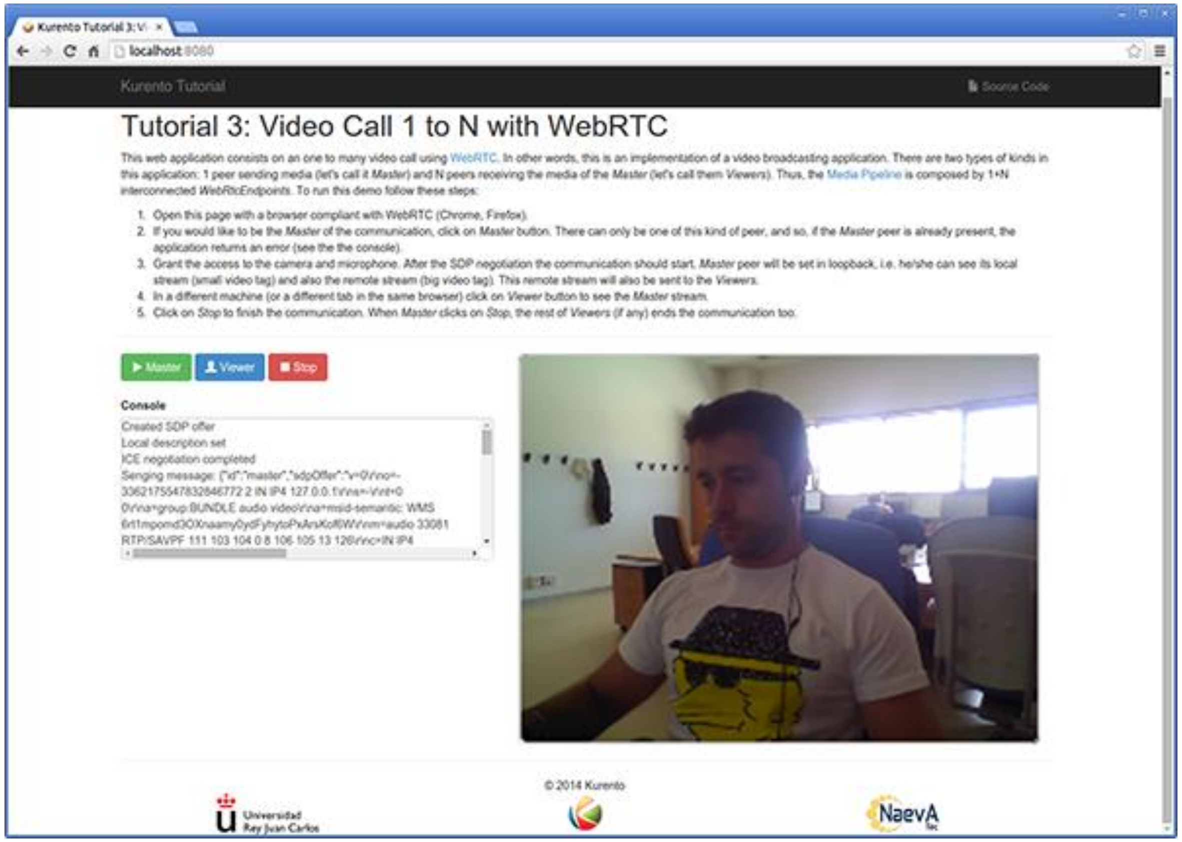
Console

```
Senging message: {"id":"register","name":"boni"}
Received message: {"id":"resgisterResponse","response":"accepted"}
Created SDP offer
Local description set
ICE negotiation completed
Invoking SDP offer callback function
Senging message:
{"id":"call","from":"boni","to":"mica","sdpOffer":"v=0\r\no=-
```

© 2014 Kurento

Video call one to many





Tutorial 3: Video Call 1 to N with WebRTC

This web application consists on an one to many video call using WebRTC. In other words, this is an implementation of a video broadcasting application. There are two types of kinds in this application: 1 peer sending media (let's call it *Master*) and N peers receiving the media of the *Master* (let's call them *Viewers*). Thus, the Media Pipeline is composed by 1+N interconnected *WebRtcEndpoints*. To run this demo follow these steps:

1. Open this page with a browser compliant with WebRTC (Chrome, Firefox).
2. If you would like to be the *Master* of the communication, click on *Master* button. There can only be one of this kind of peer, and so, if the *Master* peer is already present, the application returns an error (see the the console).
3. Grant the access to the camera and microphone. After the SDP negotiation the communication should start. *Master* peer will be set in loopback, i.e. he/she can see its local stream (small video tag) and also the remote stream (big video tag). This remote stream will also be sent to the *Viewers*.
4. In a different machine (or a different tab in the same browser) click on *Viewer* button to see the *Master* stream.
5. Click on *Stop* to finish the communication. When *Master* clicks on *Stop*, the rest of *Viewers* (if any) ends the communication too.

Console

```
Created SDP offer
Local description set
ICE negotiation completed
Senging message: {"id":"master","sdpOffer":"v=0\r\no=-
3062175547832846772 2 IN IP4 127.0.0.1\r\ns=-\r\nt=0
0\r\na=group:BUNDLE audio video\r\na=msid-semantic: WMS
6rt1mpomd3OXnaamy0ydFyhytoPxAnKof6W\r\nm=audio 33081
RTP/SAVPF 111 103 104 0 8 106 105 13 126\r\nc=IN IP4
```

© 2014 Kurento

Further information

- Home page
  http://www.kurento.org/
- Source code
  https://github.com/kurento
- Developers guide
  http://doc-kurento.readthedocs.io/
- Support
  https://groups.google.com/forum/#!forum/kurento
- Twitter
  https://twitter.com/kurentoms

Read *the* Docs