Open APIs for Open Minds

Kurento Real Time Media Stream Processing

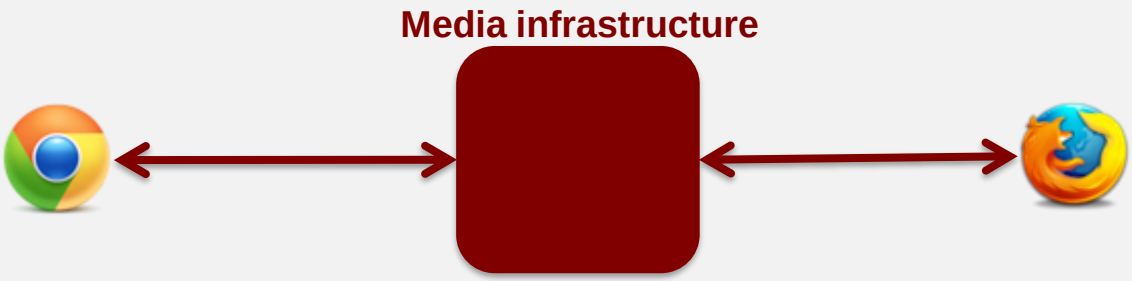
Juan Ángel Fuentes
Software Developer. Stream Oriented GE
jafuentes@naevatec.com

FIWARE

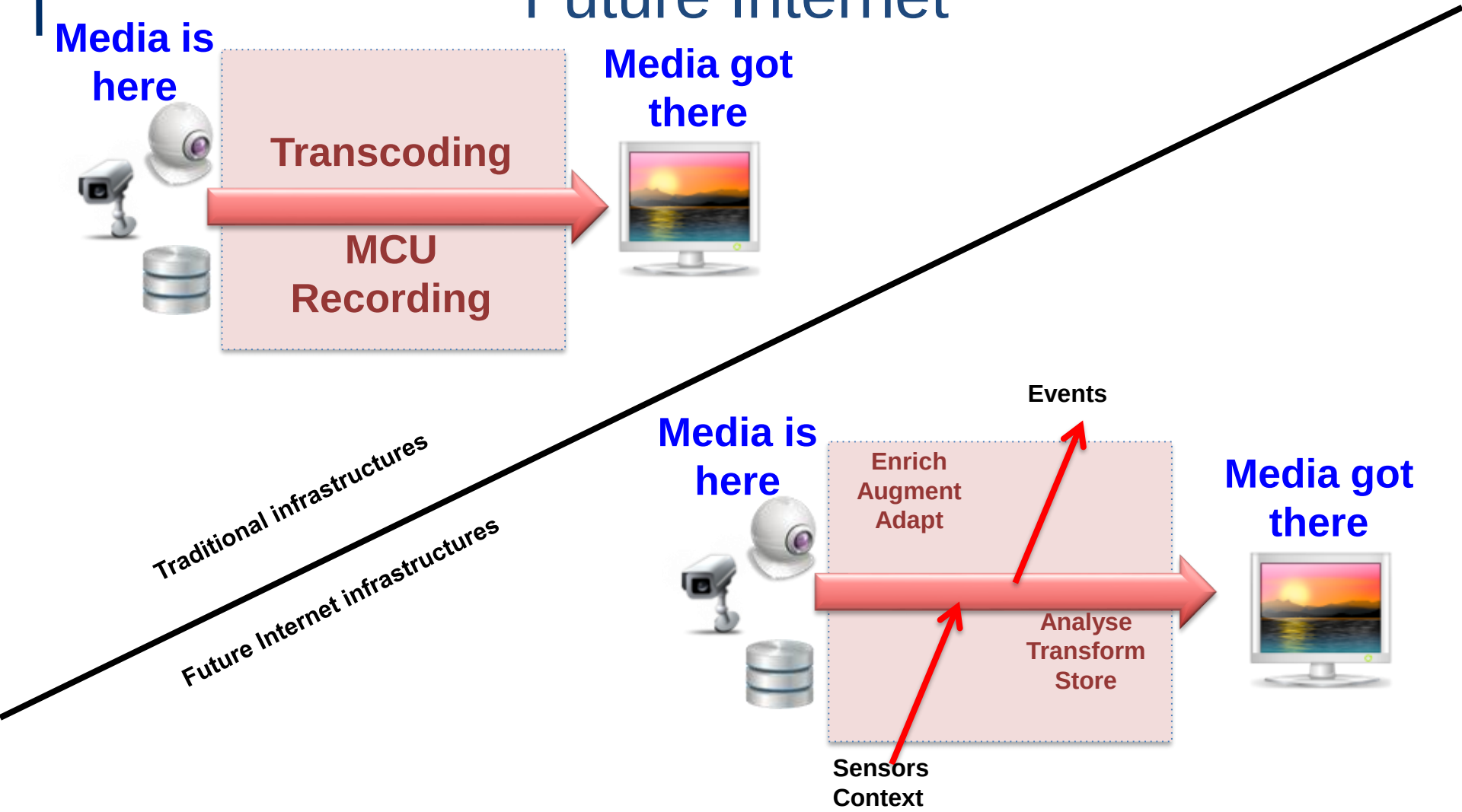
Introducing multimedia infrastructures

Peer-to-Peer Application (without media infrastructure)

Application with media infrastructure



Multimedia infrastructures for the Future Internet



Enrich, augment, adapt, analyse, transform, store: what's the problem?

Complexity

FIWARE

The Stream Oriented Generic Enabler – SO GE

Multimedia infrastructure

- Interoperable media exchange (multiplatform/multiprotocol)
  - WebRTC, RTP, HTTP (video tag), etc.
- Standard capabilities
  - Transcoding, MCU, recording
- Advanced capabilities
  - Computer vision, augmented reality, mixing, blending, etc.

APIs

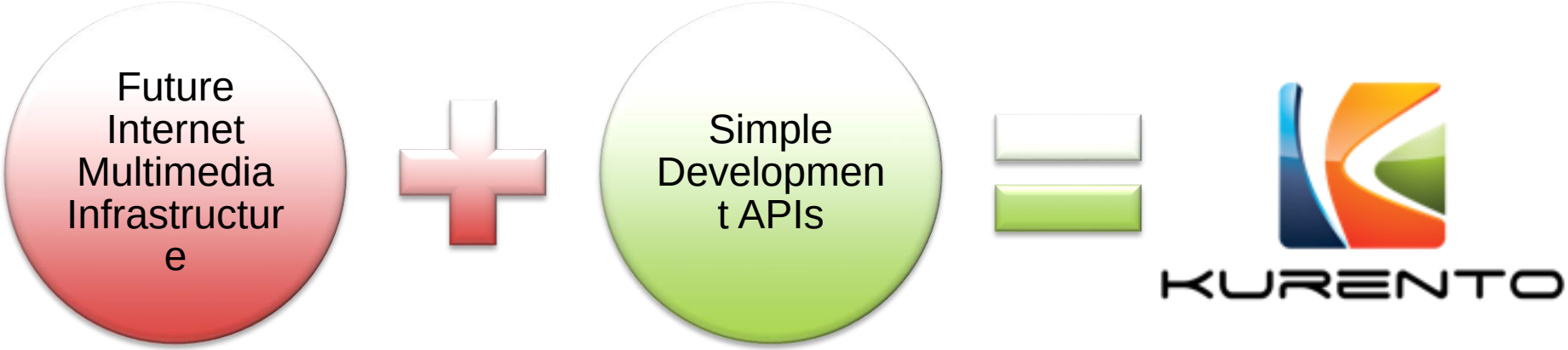
- REST API
- JavaScript API
- Java API

Is distributed through a flexible FOSS license

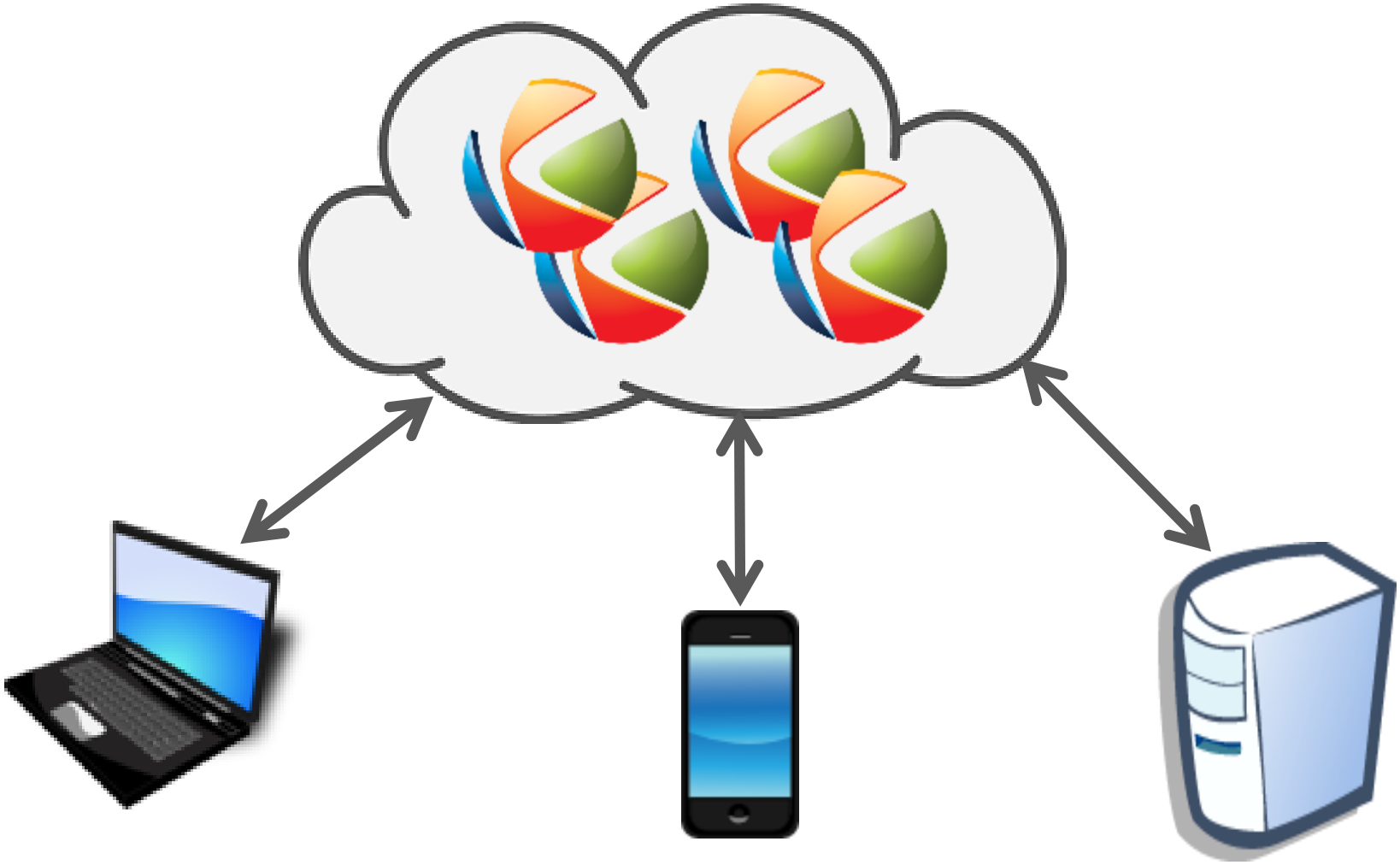
- LGPL 2.1

FIWARE

Kurento: The equation



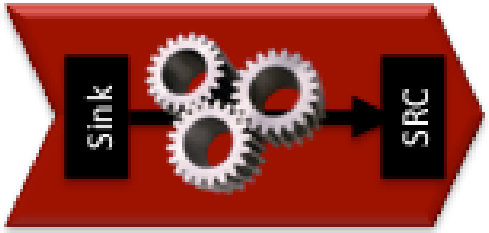
General Architecture



FIWARE

Key Concepts: Media Elements

- Functional unit performing a **specific action** on a media stream
- Developers **abstract** from the low level implementation
- Able to **receive media** form other elements
- Able to **send media** to other elements

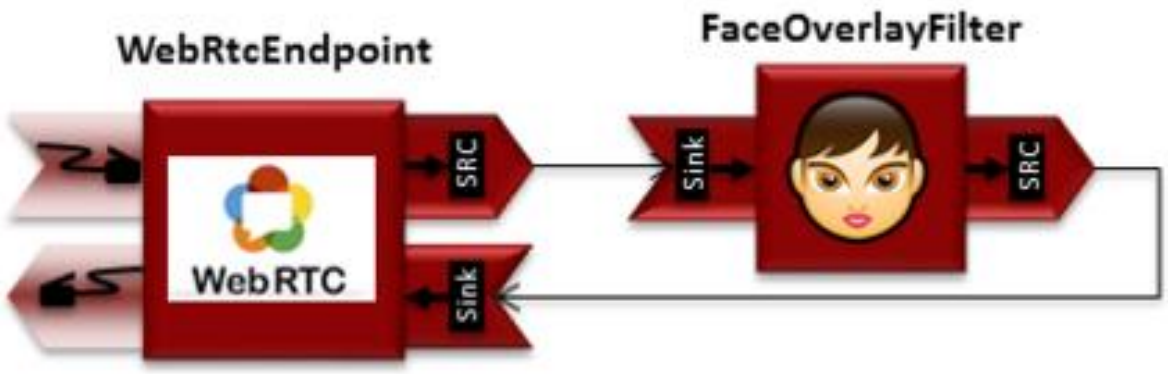


Key Concepts: Types of Media Elements

- **Input Endpoints:** injects media
- **Filters:** Transform or analyze media
- **Hubs:** Manage multiple media flows
- **Output Endpoints:** take the media stream out

Key Concepts: Media Pipeline

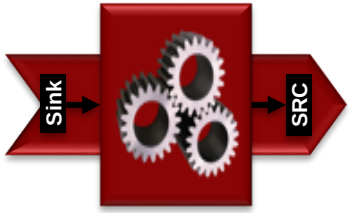
- **Media Pipeline:** is a chain of media elements, where the output stream generated by one element (source) is fed into one or more other elements input streams (sinks)



Key concepts: media elements and pipelines

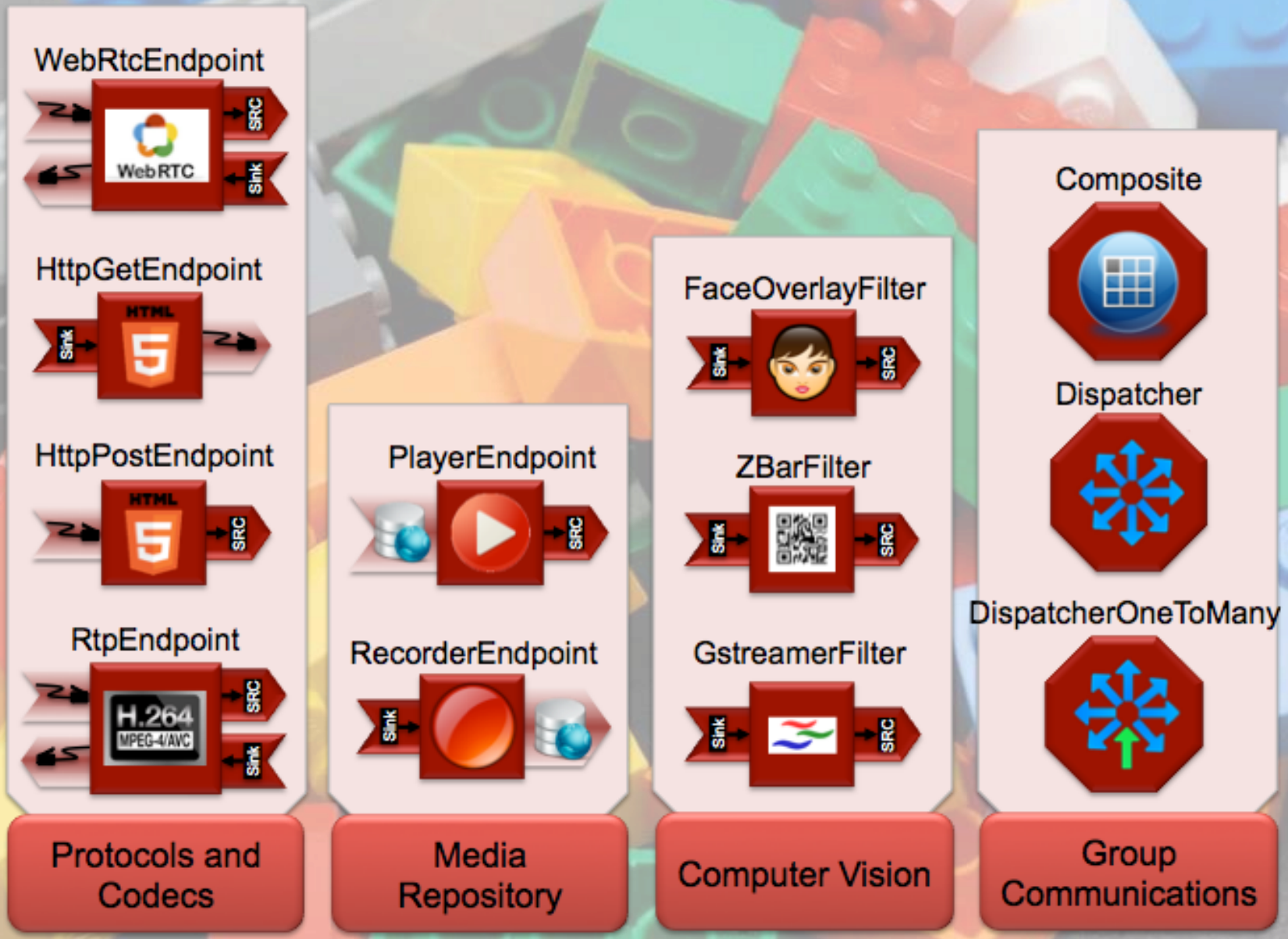
- **Media Element**
  - **Provides a specific media functionality**
    - **Send/receive media. These are the Endpoints**
    - **Process media**
    - **Transform media**
  - **Ready to be used**
  - **New media elements can be added**

Media Element



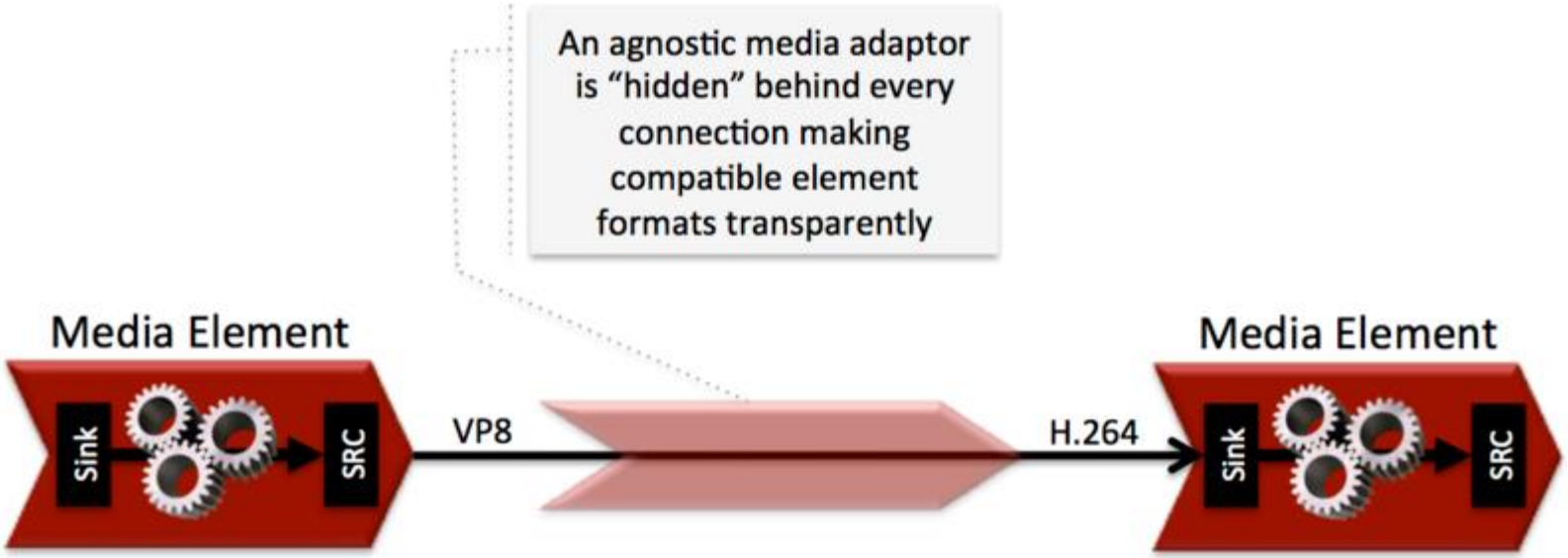
Media Pipeline

- **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**

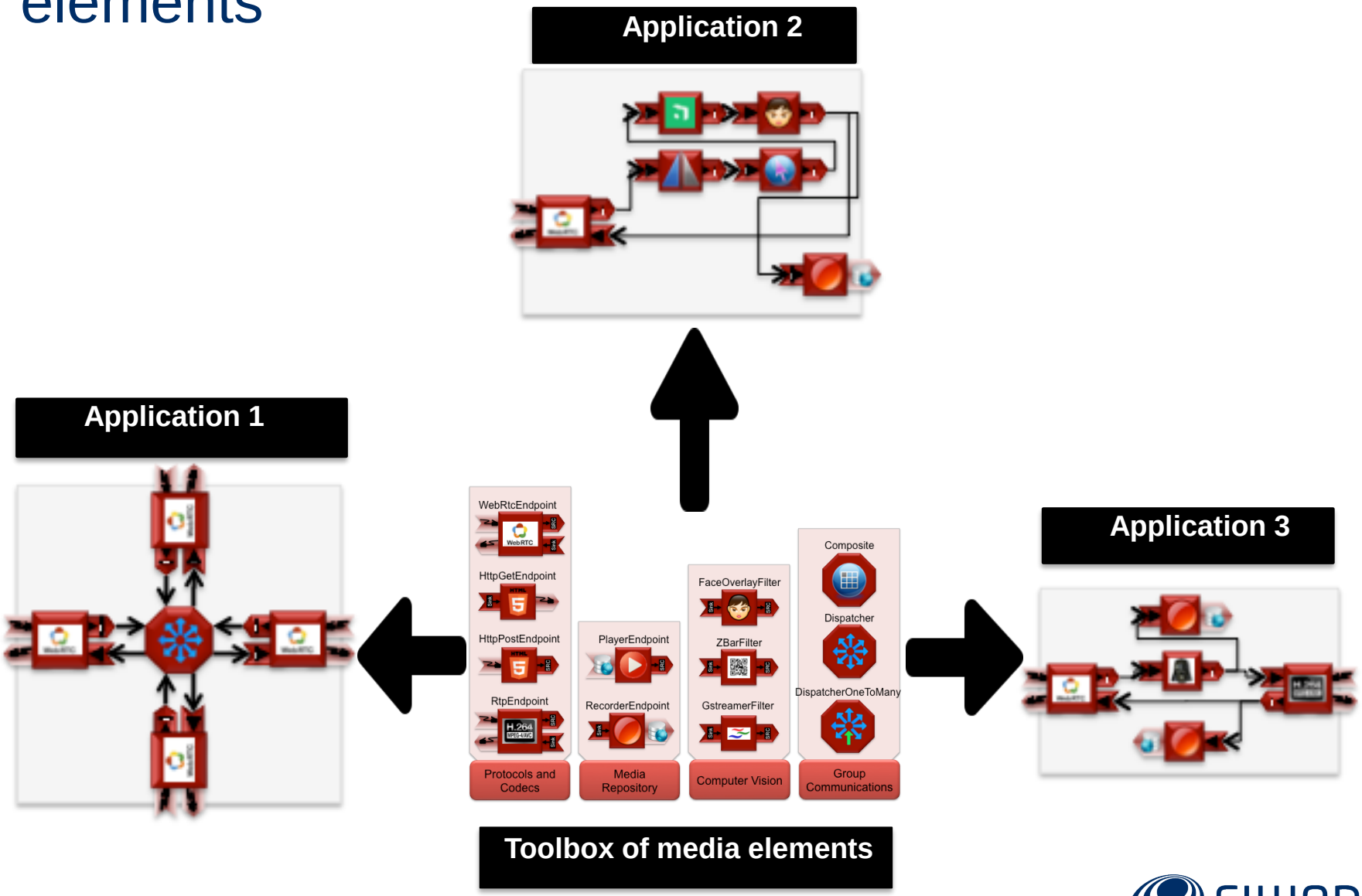

WebRtcEndpoint
WebRTC
SRC
Sink
HttpGetEndpoint
HTML
Sink
HttpPostEndpoint
HTML
SRC
RtpEndpoint
H.264
MPEG-4/AVC
SRC
Sink
PlayerEndpoint
SRC
RecorderEndpoint
Sink
FaceOverlayFilter
Sink
SRC
ZBarFilter
Sink
SRC
GstreamerFilter
Sink
SRC
Composite
Dispatcher
DispatcherOneToMany
Protocols and Codecs
Media Repository
Computer Vision
Group Communications

Agnostic Media adapter

- In charge of making possible that Stream Oriented GE APIS **allow developers to combine media elements** to create the desired pipeline
- Fully abstracts all the complexities of media codecs and formats



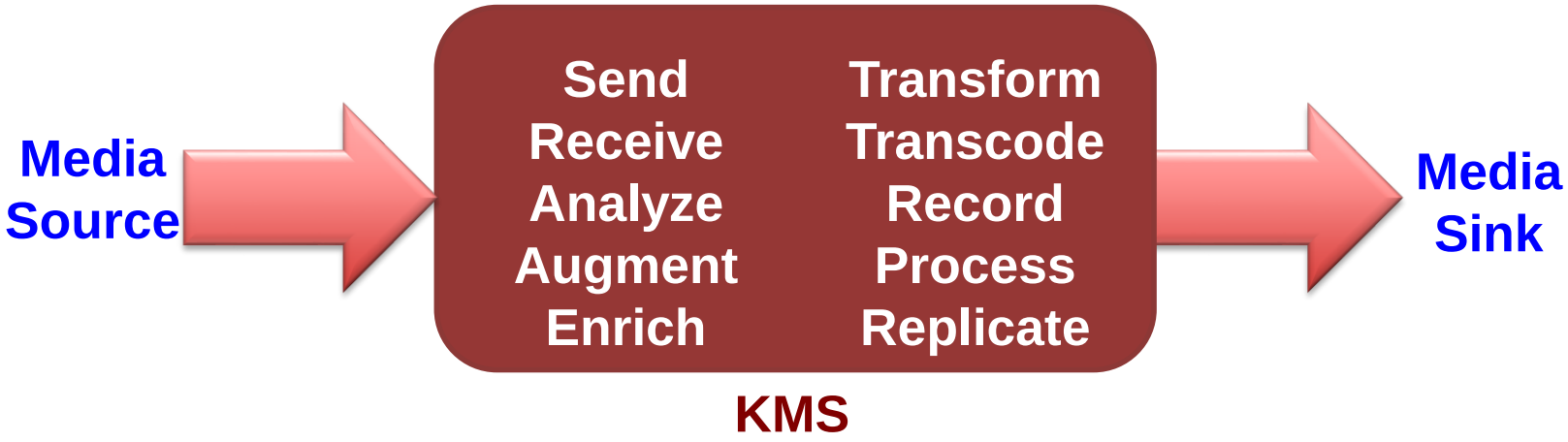
Developers create applications just connecting elements



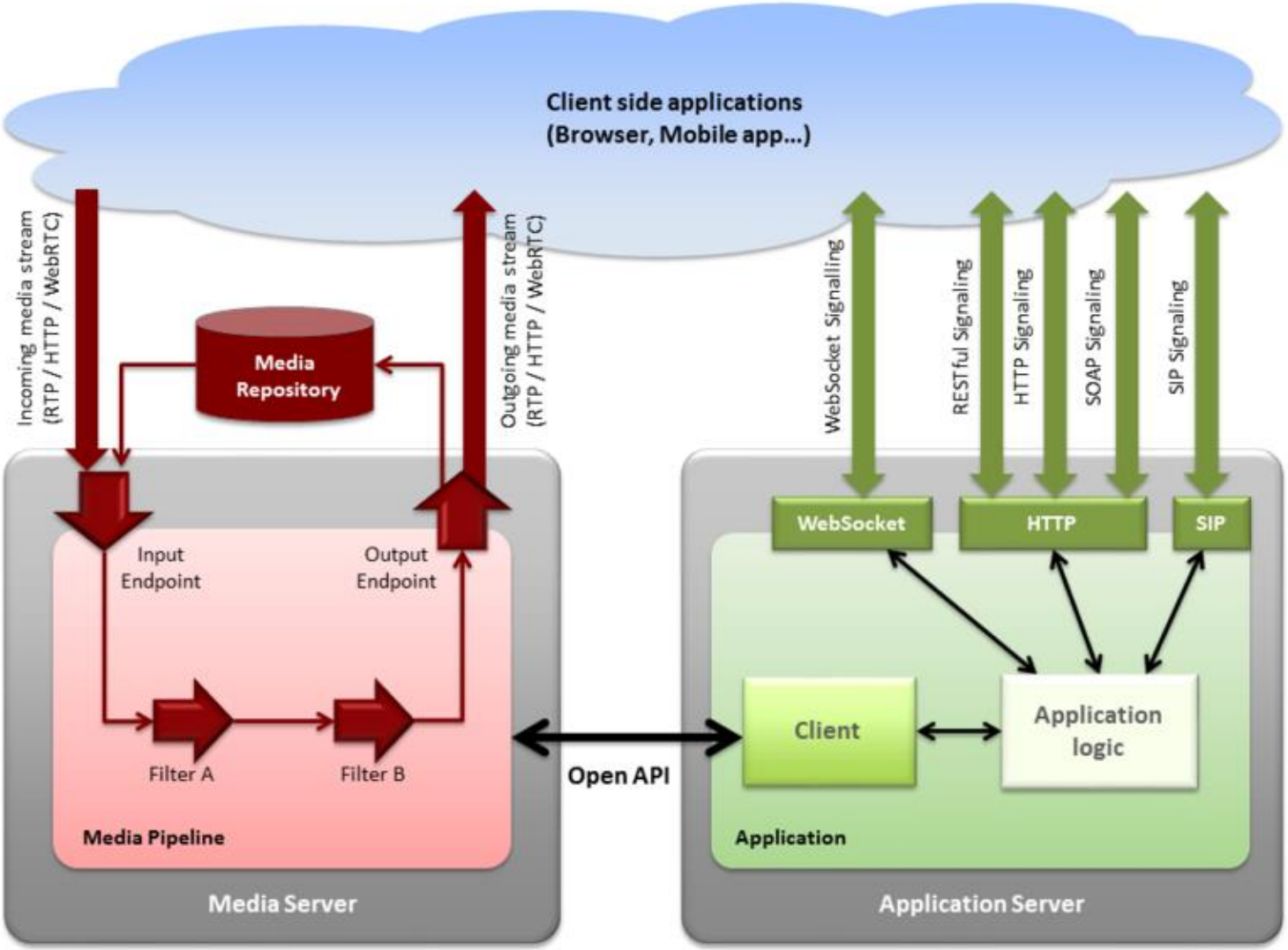
FIWARE

Media Server (KMS): The nucleus of Kurento

- **KMS is a middleware for media streams:**
- **Receives the stream**
- **Process the stream**
- **Issues the stream**



Media plane and Signaling plane



How to control the media server?

- **REST API**
- **JSON RPC**
- **Websocket interface in KMS**

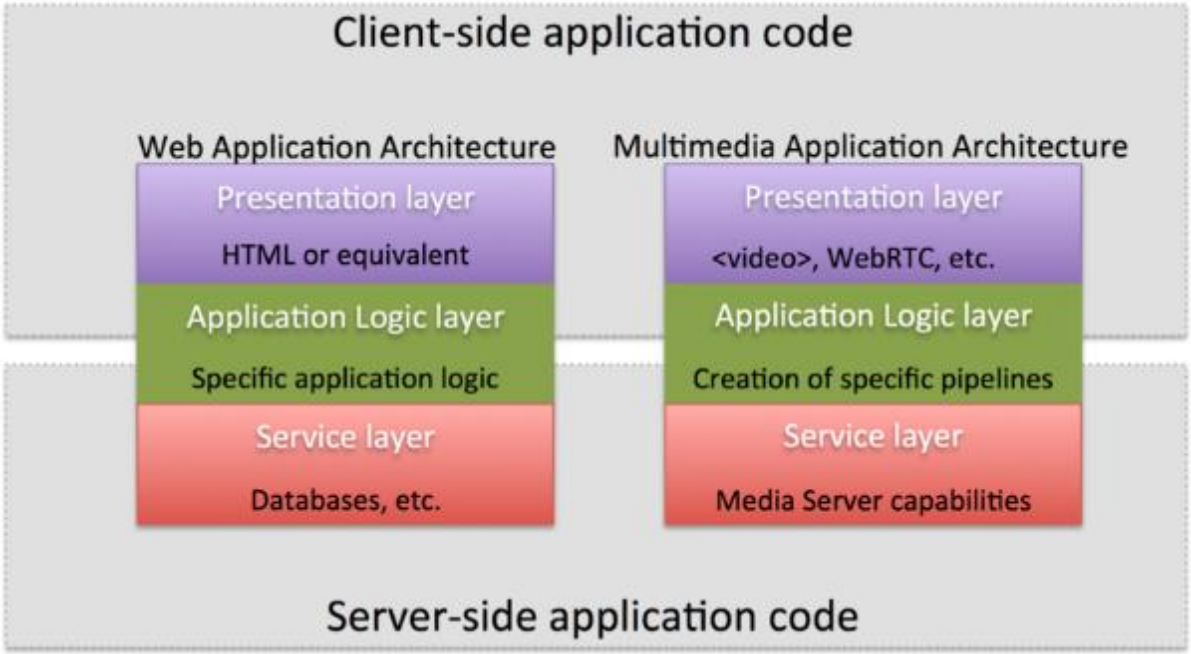
API Implementations

For now…

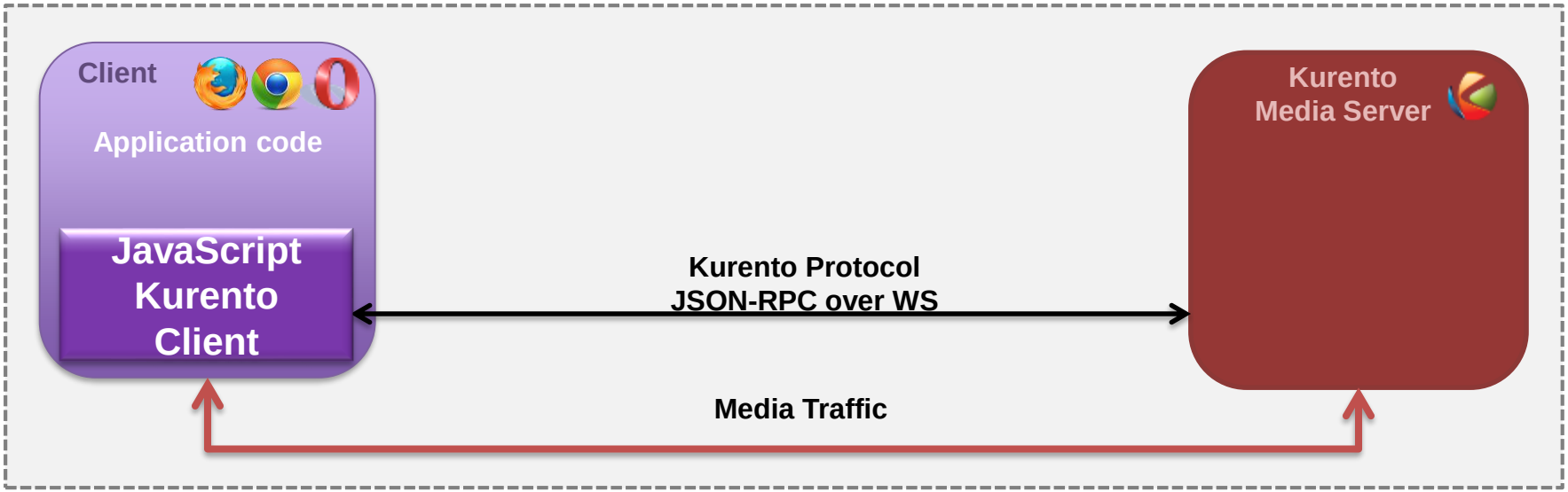
Multimedia Application Architecture in SO GE

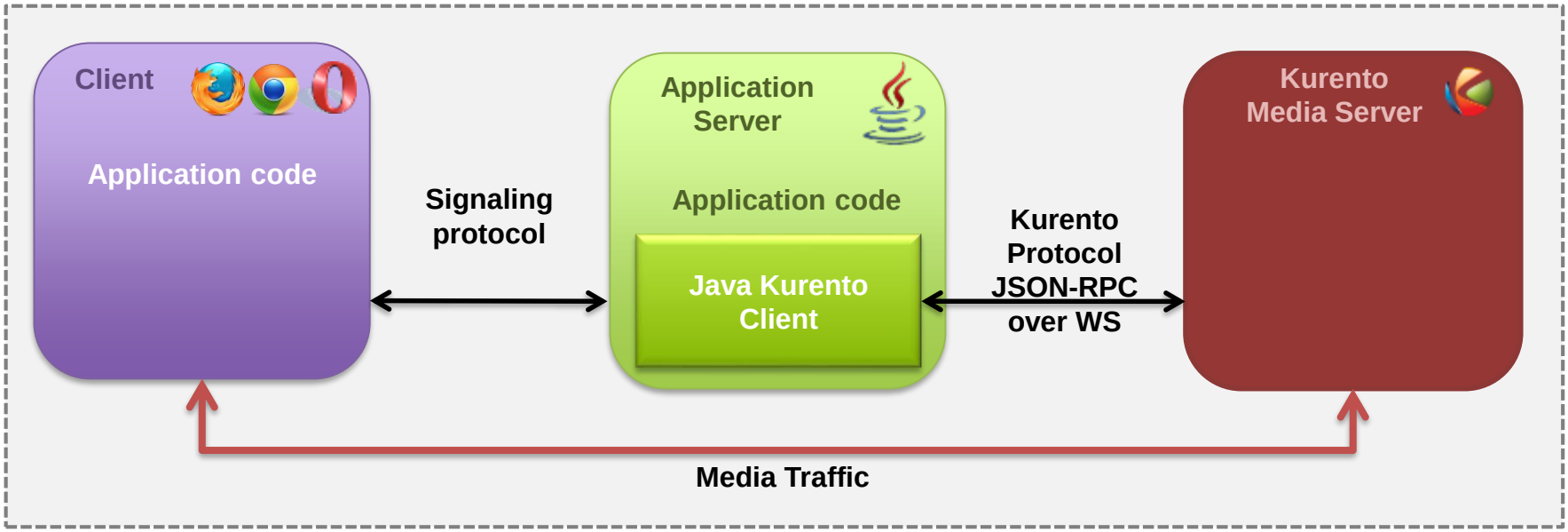
Web application three layers parallelism:

- **Presentation layer:** Multimedia presentation and capturing
- **Application logic:** Multimedia logic in charge of building the pipeline
- **Service Layer:** Kurento Media Server

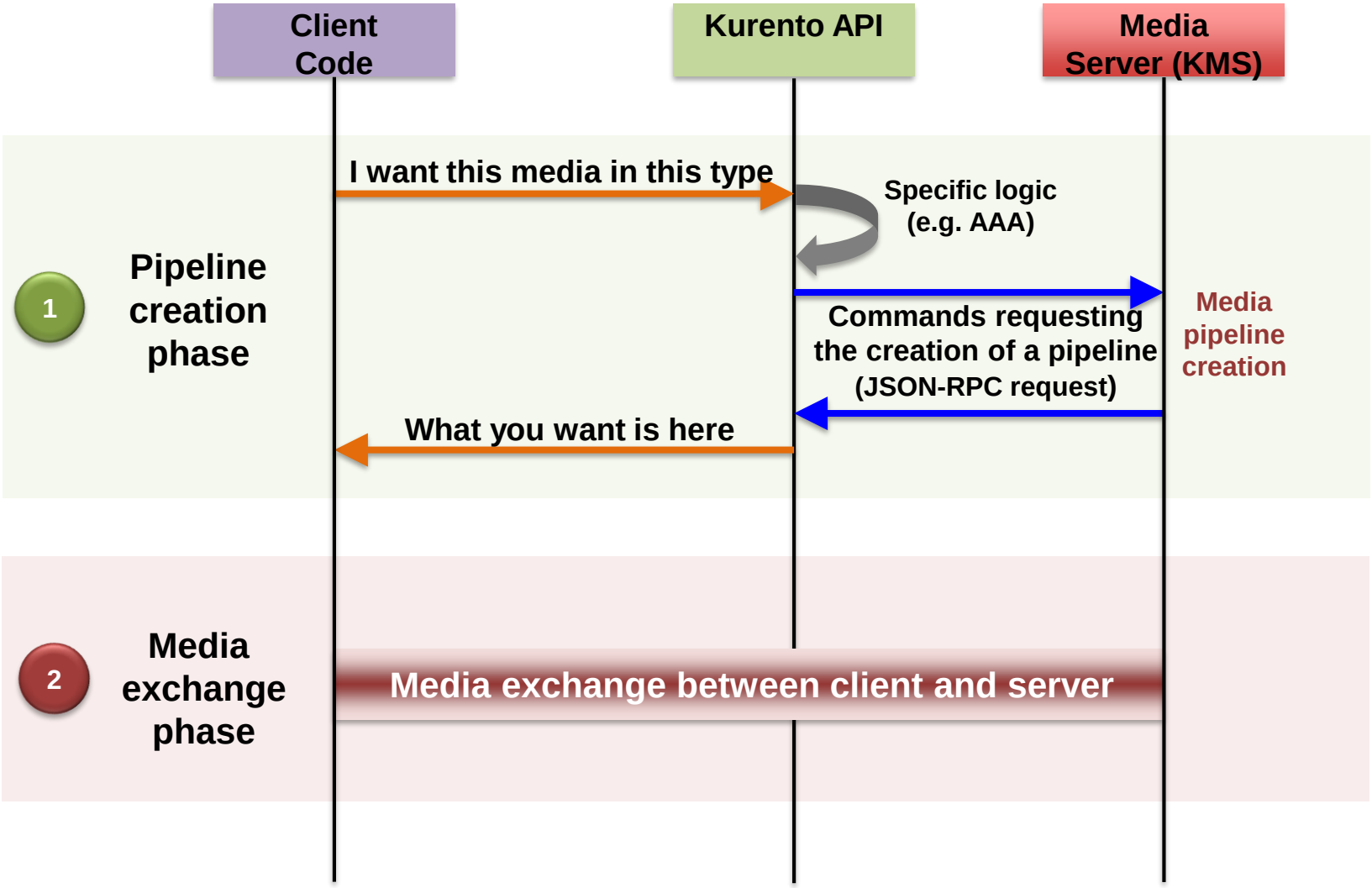


Kurento Applications





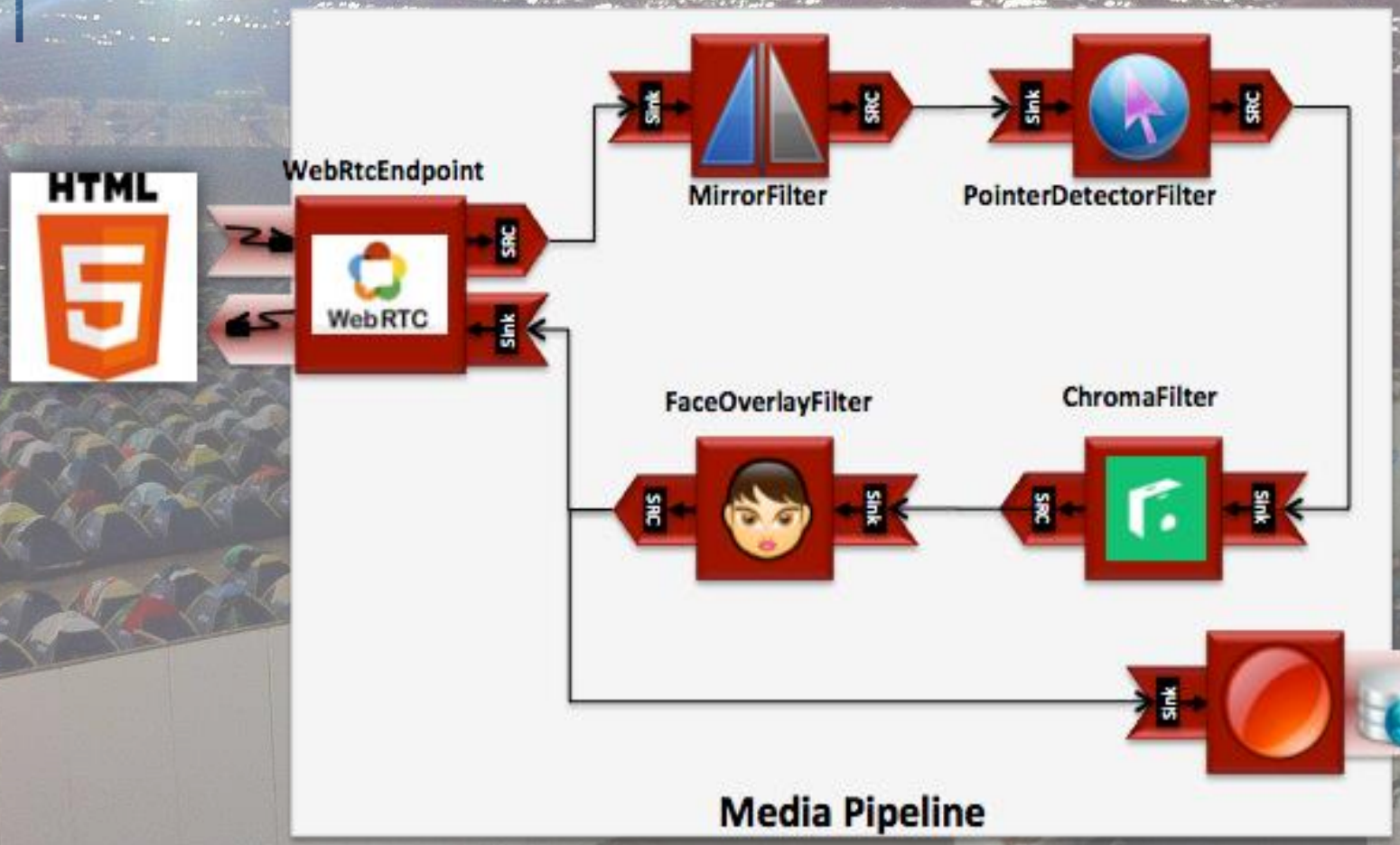
Application execution flow



Campus Party Brazil

FIWARE

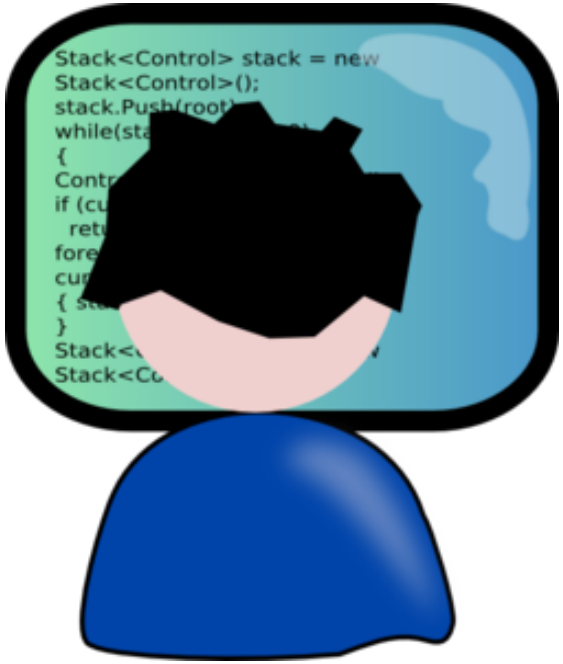
Campus Party Brazil



FIWARE

Let’s get started!

Tutorials



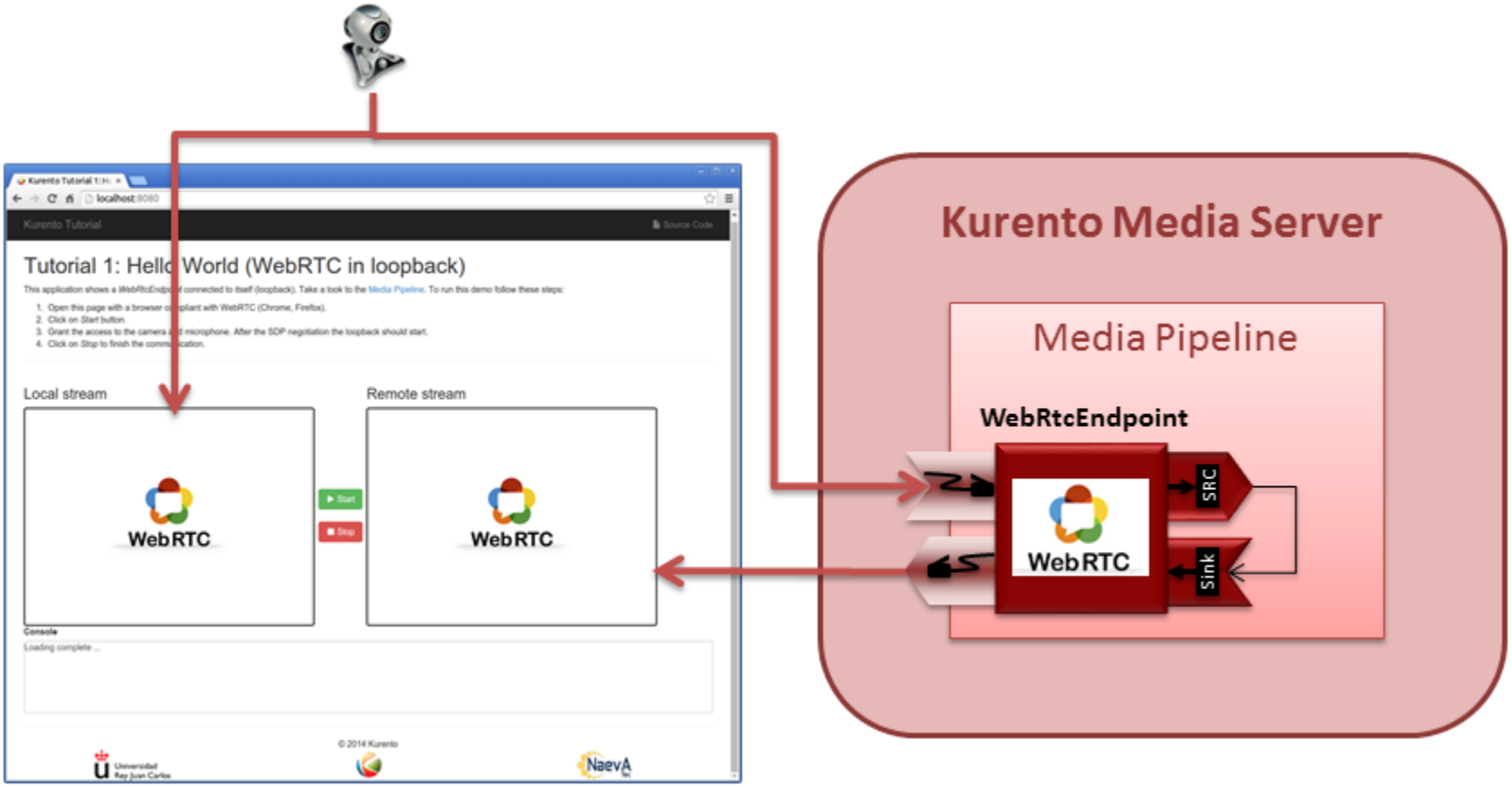
http://doc-kurento.readthedocs.io/en/stable/tutorials.html

FIWARE

What do we need?

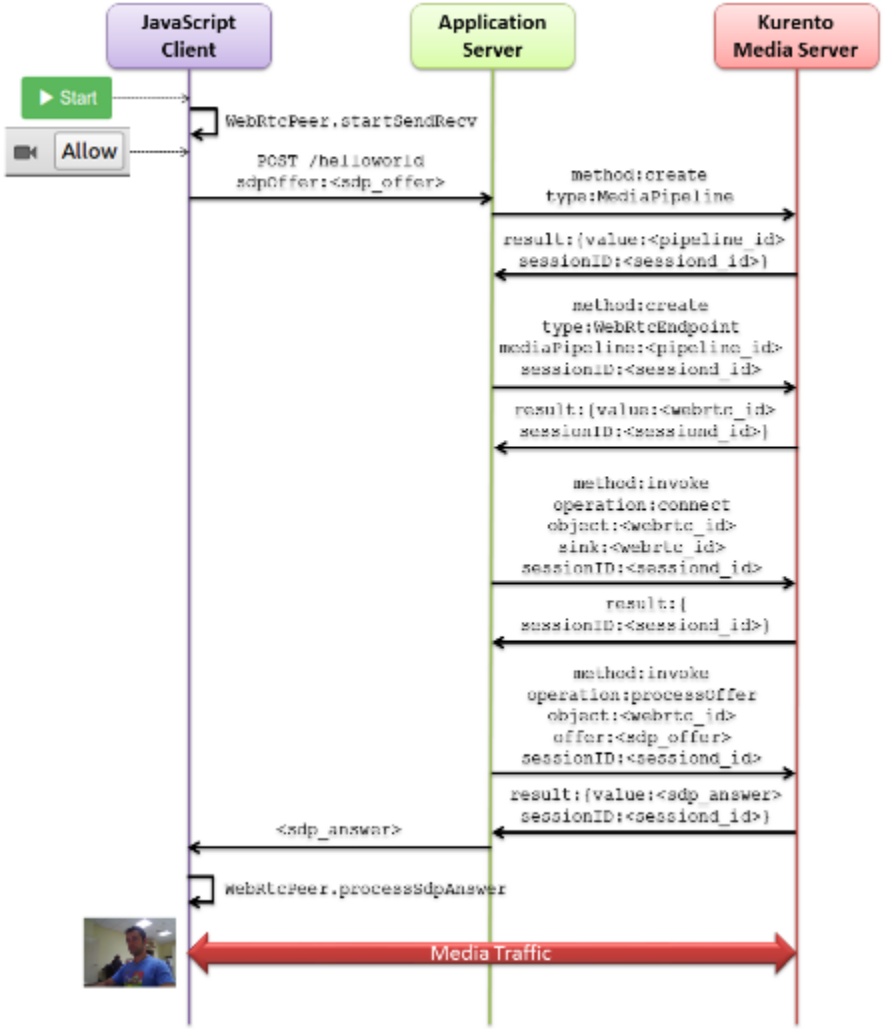
- **An instance of the Kurento Media Server**
  - **FIWARE-LAB**
  - **Local installation**
- **Maven**
- **NPM**
- **Bower**
- **Your favorite IDE**

Tutorial 1 - Hello world



FIWARE

Tutorial 1 - Hello world



FIWARE

Tutorial 1 - Hello world, server

```
@RestController
public class HelloWorldController {

   @Autowired
   private KurentoClient kurento;

   @RequestMapping(value = "/helloworld", method = RequestMethod.POST)
   private String processRequest(@RequestBody String sdpOffer)
         throws IOException {

      // Media Logic
      MediaPipeline pipeline = kurento.createMediaPipeline();
      WebRtcEndpoint webRtcEndpoint = new WebRtcEndpoint.Builder(pipeline)
            .build();
      webRtcEndpoint.connect(webRtcEndpoint);

      // SDP negotiation (offer and answer)
      String responseSdp = webRtcEndpoint.processOffer(sdpOffer);
      return responseSdp;
   }

}
```

FIWARE

Tutorial 1 - Hello world, client

```
var webRtcPeer;

function start() {
   console.log("Starting video call ...");
   showSpinner(videoInput, videoOutput);
   webRtcPeer = kurentoUtils.WebRtcPeer.startSendRecv(videoInput, videoOutput, onOffer, onError);
}

function onOffer(sdpOffer) {
   console.info('Invoking SDP offer callback function ' + location.host);
   $.ajax({
      url : location.protocol + '/helloworld',
      type : 'POST',
      dataType : 'text',
      contentType : 'application/sdp',
      data : sdpOffer,
      success : function(sdpAnswer) {
         console.log("Received sdpAnswer from server. Processing ...");
         webRtcPeer.processSdpAnswer(sdpAnswer);
      },
      error : function(jqXHR, textStatus, error) {
         onError(error);
      }
   });
}

function onError(error) {
   console.error(error);
}
```

Tutorial 1 - Hello world (Node.js)

```
var kurento = require('kurento-client');

//...

const ws_uri = "ws://localhost:8888/kurento";

//...

kurento(ws_uri, function(error, kurentoClient) {
        if (error) {
                return callback(error);
        }
        kurentoClient.create('MediaPipeline', function(error, _pipeline) {
                if (error) {
                        return callback(error);
                }
                pipeline = _pipeline;
                return callback(null, pipeline);
        });
});
```

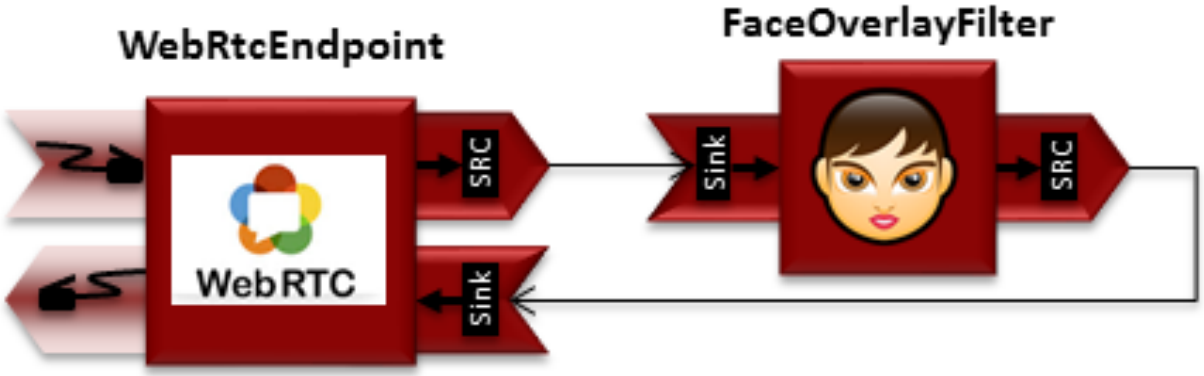
FIWARE

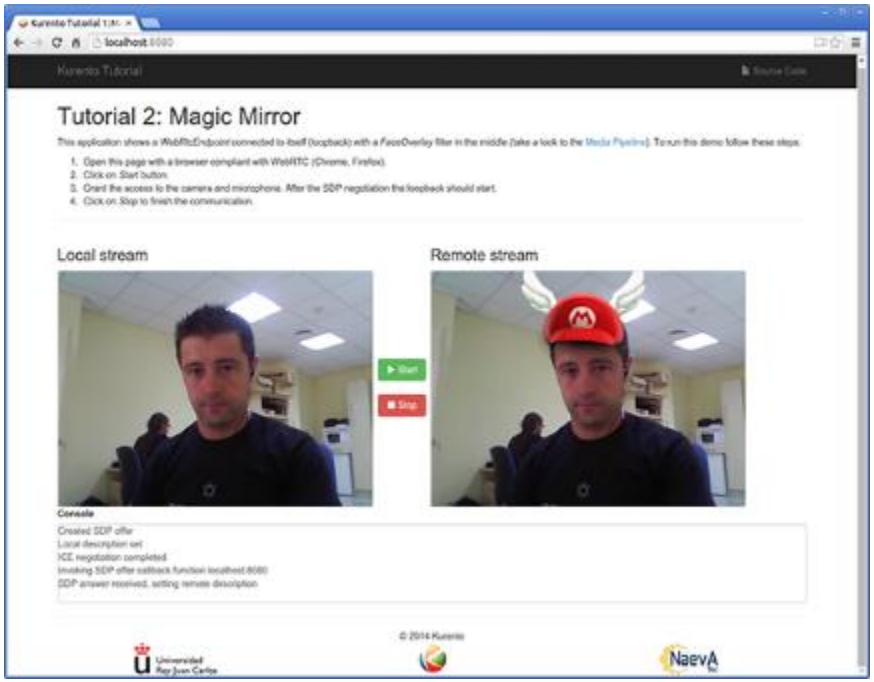
Tutorial 1 - Hello world (Node.js)

```
app.post('/helloworld', function(req, res) {
   var sdpOffer = req.body;
   getPipeline(function(error, pipeline) {
      pipeline.create('WebRtcEndpoint', function(error, webRtcEndpoint) {
         webRtcEndpoint.processOffer(sdpOffer, function(error, sdpAnswer) {
            webRtcEndpoint.connect(webRtcEndpoint, function(error) {
               res.type('application/sdp');
               res.send(sdpAnswer);
            });
         });
      });
   });
});
```

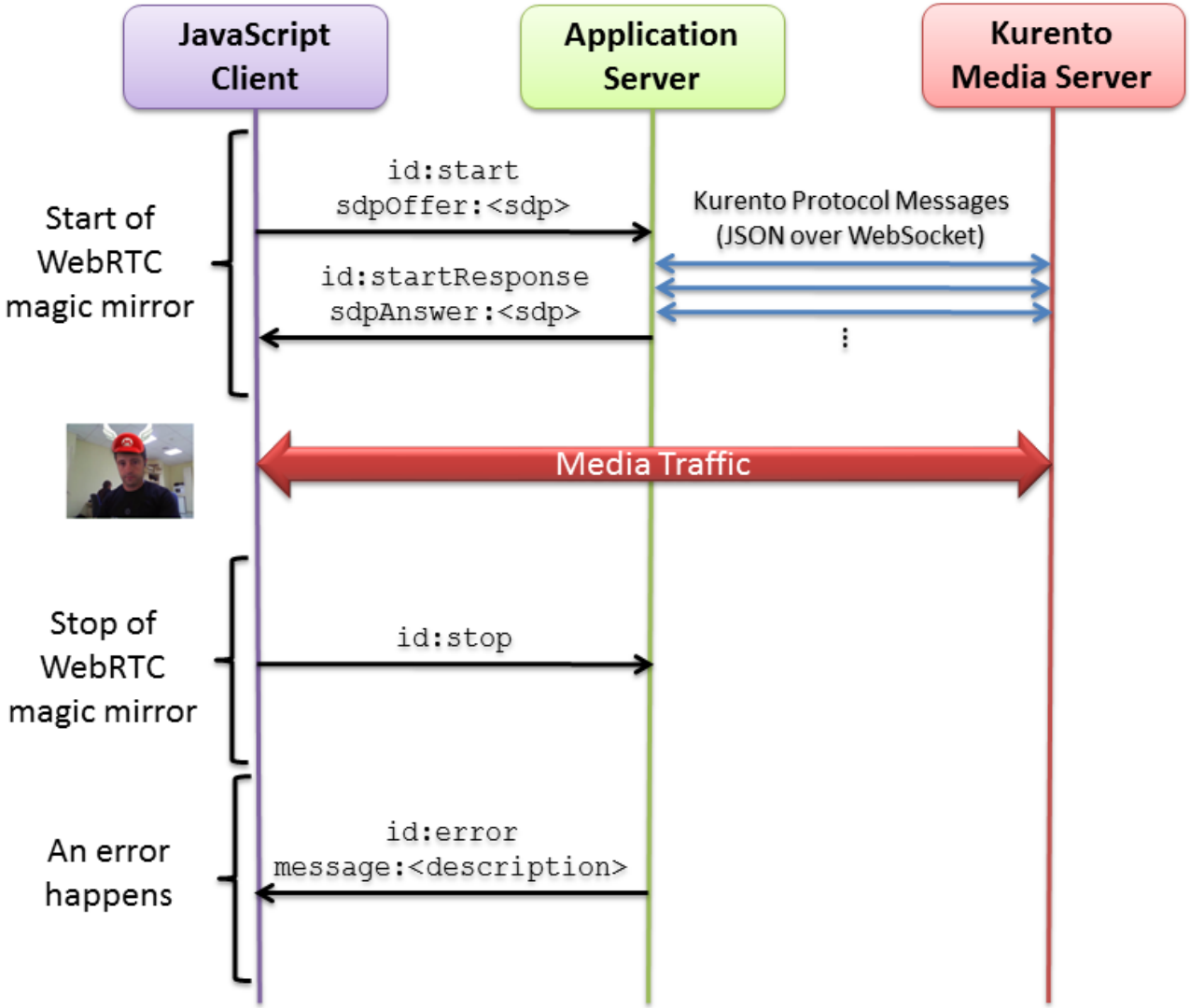
FIWARE

Tutorial 2 - WebRTC magic mirror





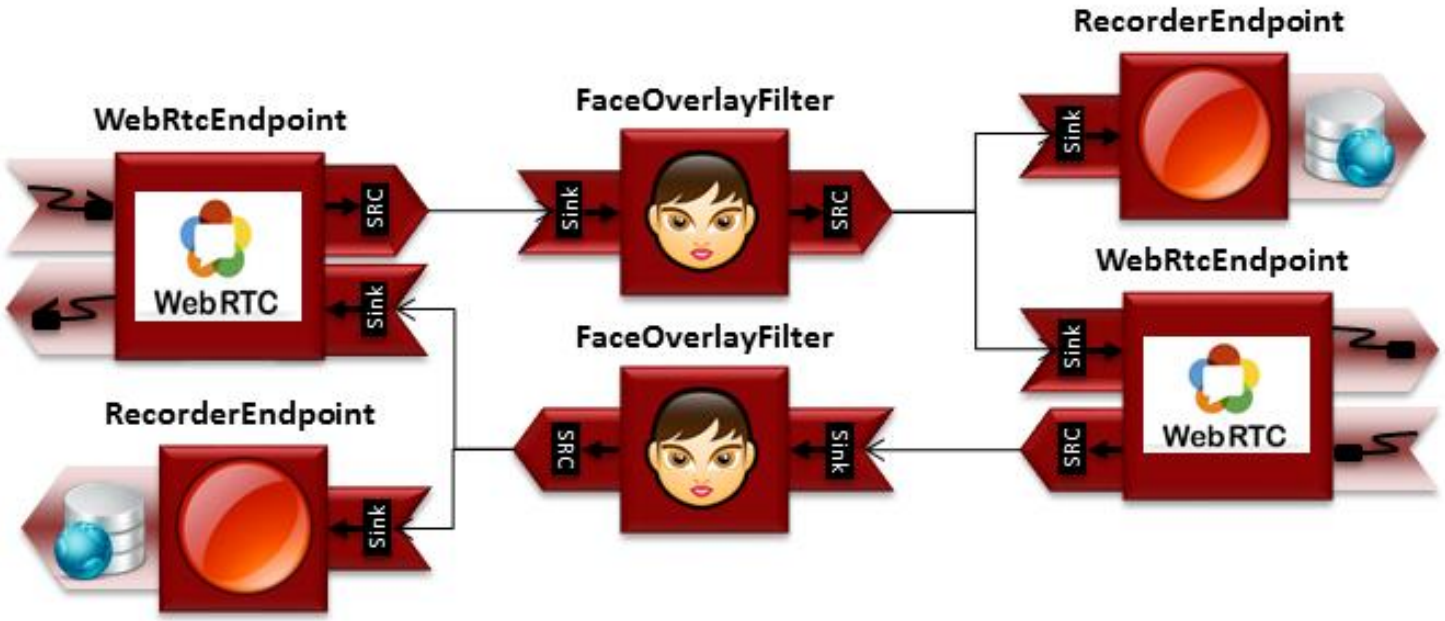
Tutorial 2 - WebRTC magic mirror



Tutorial 2 - WebRTC magic mirror

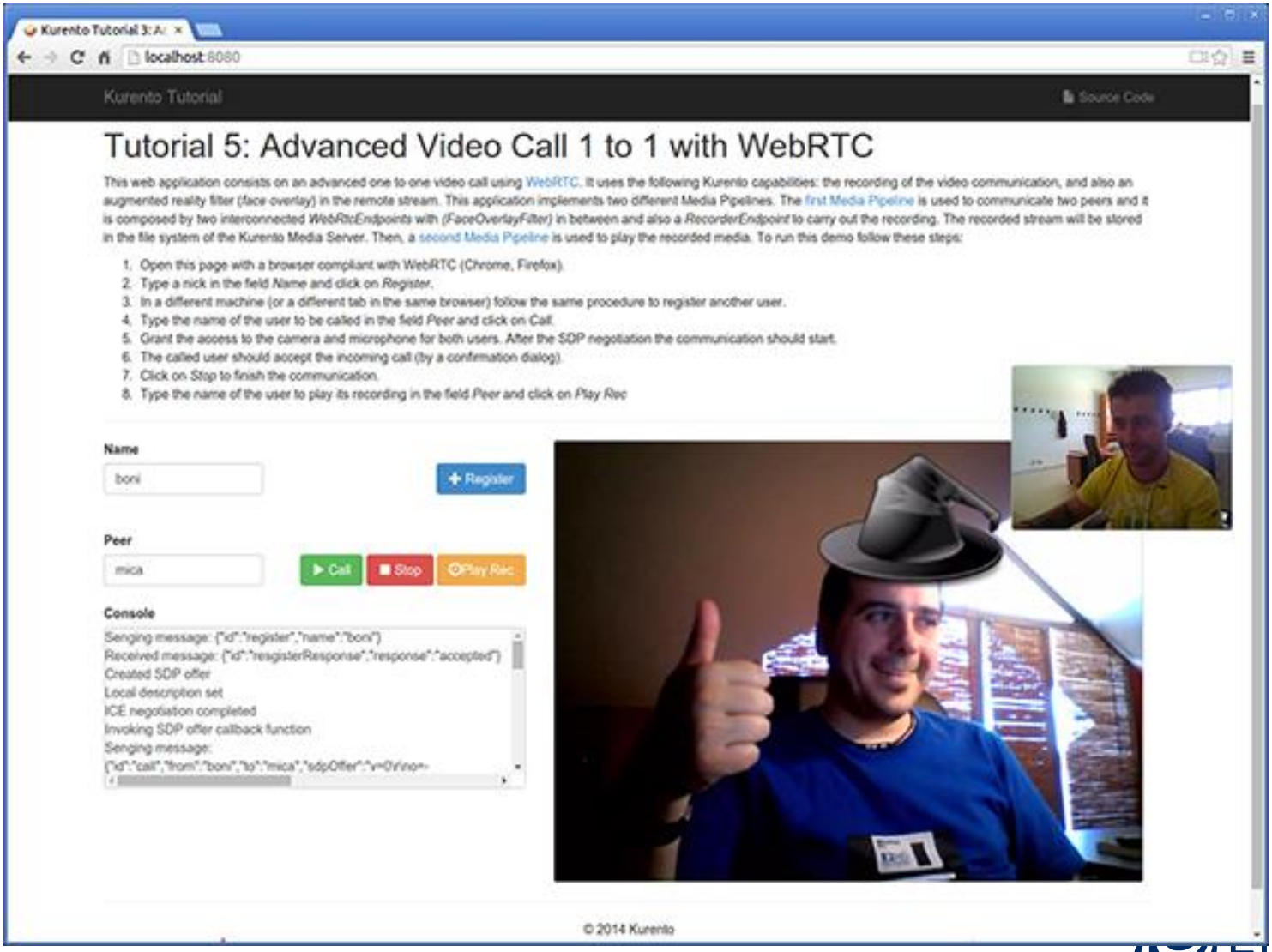
```
private void start(WebSocketSession session, JsonObject jsonMessage) {
   try {
      // Media Logic (Media Pipeline and Elements)
      MediaPipeline pipeline = kurento.createMediaPipeline();
      pipelines.put(session.getId(), pipeline);

      WebRtcEndpoint webRtcEndpoint = new WebRtcEndpoint.Builder(pipeline)
            .build();
      FaceOverlayFilter faceOverlayFilter = new FaceOverlayFilter.Builder(
            pipeline).build();
      faceOverlayFilter.setOverlayedImage(
            "http://files.kurento.org/imgs/mario-wings.png", -0.35F,
            -1.2F, 1.6F, 1.6F);

      webRtcEndpoint.connect(faceOverlayFilter);
      faceOverlayFilter.connect(webRtcEndpoint);

      // SDP negotiation (offer and answer)
      String sdpOffer = jsonMessage.get("sdpOffer").getAsString();
      String sdpAnswer = webRtcEndpoint.processOffer(sdpOffer);

      // Sending response back to client
      JsonObject response = new JsonObject();
      response.addProperty("id", "startResponse");
      response.addProperty("sdpAnswer", sdpAnswer);
      session.sendMessage(new TextMessage(response.toString()));
   } catch (Throwable t) {
      sendError(session, t.getMessage());
   }
}
```

Tutorial 3 - WebRTC one-to-one video call with recording and filtering



FIWARE

Tutorial 3 - WebRTC one-to-one video call with recording and filtering



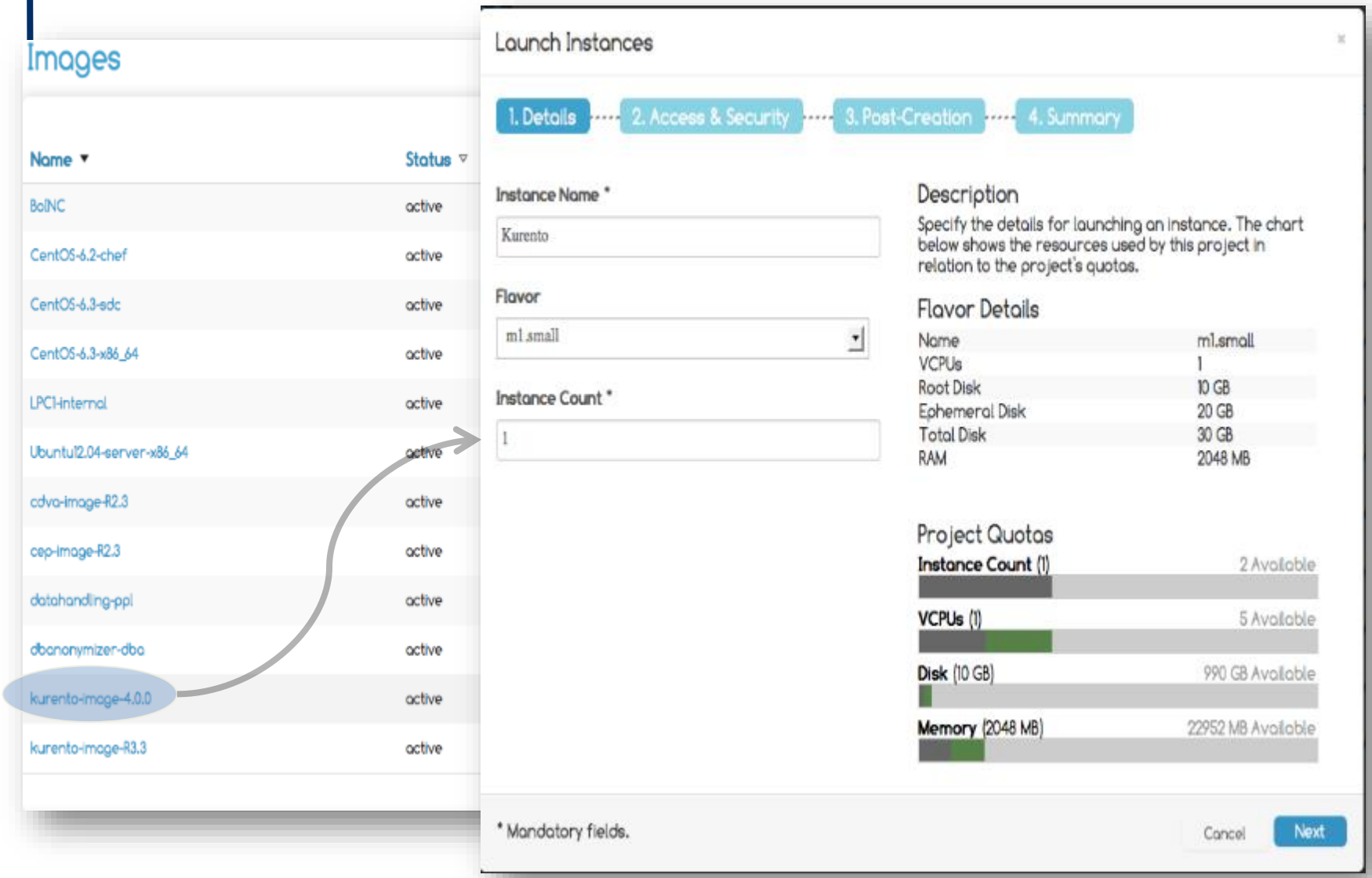
Tutorial 3 - WebRTC one-to-one video call with recording and filtering

```
public CallMediaPipeline(KurentoClient kurento, String from, String to) {
   // Media pipeline
   MediaPipeline pipeline = kurento.createMediaPipeline();

   // Media Elements (WebRtcEndpoint, RecorderEndpoint, FaceOverlayFilter)
   webRtcCaller = new WebRtcEndpoint.Builder(pipeline).build();
   webRtcCallee = new WebRtcEndpoint.Builder(pipeline).build();

   recorderCaller = new RecorderEndpoint.Builder(pipeline, RECORDING_PATH
         + from + RECORDING_EXT).build();
   recorderCallee = new RecorderEndpoint.Builder(pipeline, RECORDING_PATH
         + to + RECORDING_EXT).build();

   FaceOverlayFilter faceOverlayFilterCaller = new FaceOverlayFilter.Builder(
         pipeline).build();
   faceOverlayFilterCaller.setOverlayedImage(
         "http://files.kurento.org/imgs/mario-wings.png", -0.35F, -1.2F,
         1.6F, 1.6F);

   FaceOverlayFilter faceOverlayFilterCallee = new FaceOverlayFilter.Builder(
         pipeline).build();
   faceOverlayFilterCallee.setOverlayedImage(
         "http://files.kurento.org/imgs/Hat.png", -0.2F, -1.35F, 1.5F,
         1.5F);

   // Connections
   webRtcCaller.connect(faceOverlayFilterCaller);
   faceOverlayFilterCaller.connect(webRtcCallee);
   faceOverlayFilterCaller.connect(recorderCaller);

   webRtcCallee.connect(faceOverlayFilterCallee);
   faceOverlayFilterCallee.connect(webRtcCaller);
   faceOverlayFilterCallee.connect(recorderCallee);
}
```

Kurento and the FIWARE-LAB

- FIWARE-LAB
  - Working instance of FI-WARE enabling free experimentation with technology
  - http://lab.fiware.org
- Creating a Kurento instance from an image
  - Use latest version of Kurento images.
- Creating a Kurento instance using recipes
  - Use Ubuntu 14.04 LTS clear image
  - Use latest version of Kurento recipes

Kurento and the FIWARE-LAB



FIWARE

To learn more…

Thank you!

http://fiware.org
Follow @FIWARE on Twitter

FIWARE