

Introduction to WebRTC

Luis López

lulop@kurento.org

<http://www.kurento.org>



Real-time Communications (RTC)



Internet RTC

- Fragmentation
- No universal RTC service

Telco RTC

- Standardization
- Phone system as universal RTC service



WebRTC: a new player

WebRTC: a definition

- Framework, protocol and API that provide real-time voice, video and data in web browsers and other applications (by Salvatore Loreto)

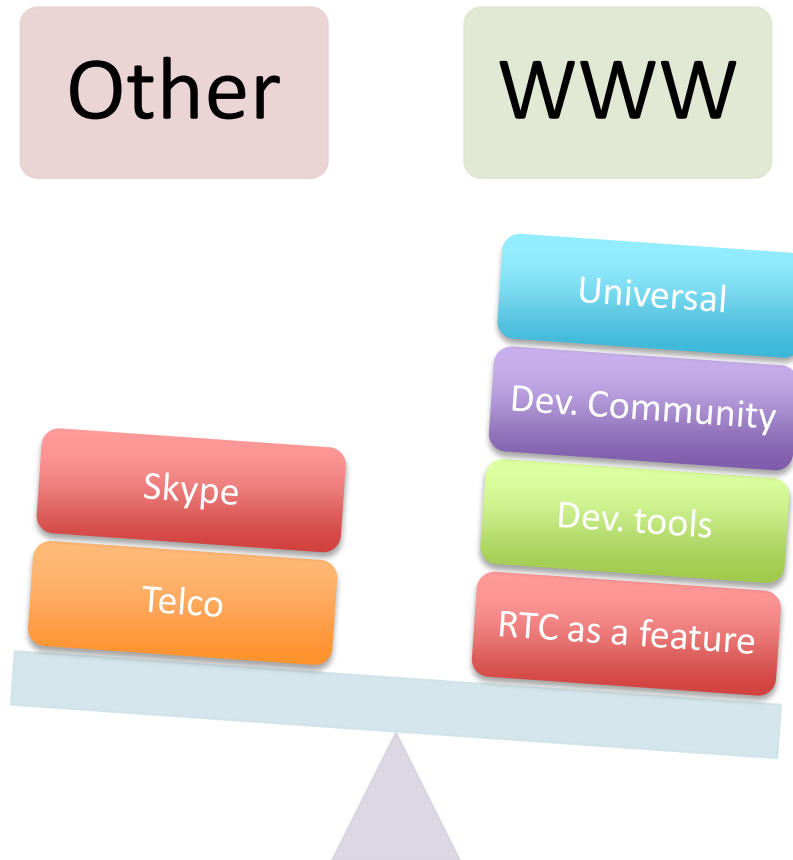
WebRTC as a framework

- Technological capabilities enabling RTC on web browsers: Codecs, NAT traversal, security, transports, etc.
- Basing on standards: RTCWeb Working Group of the IETF for protocols

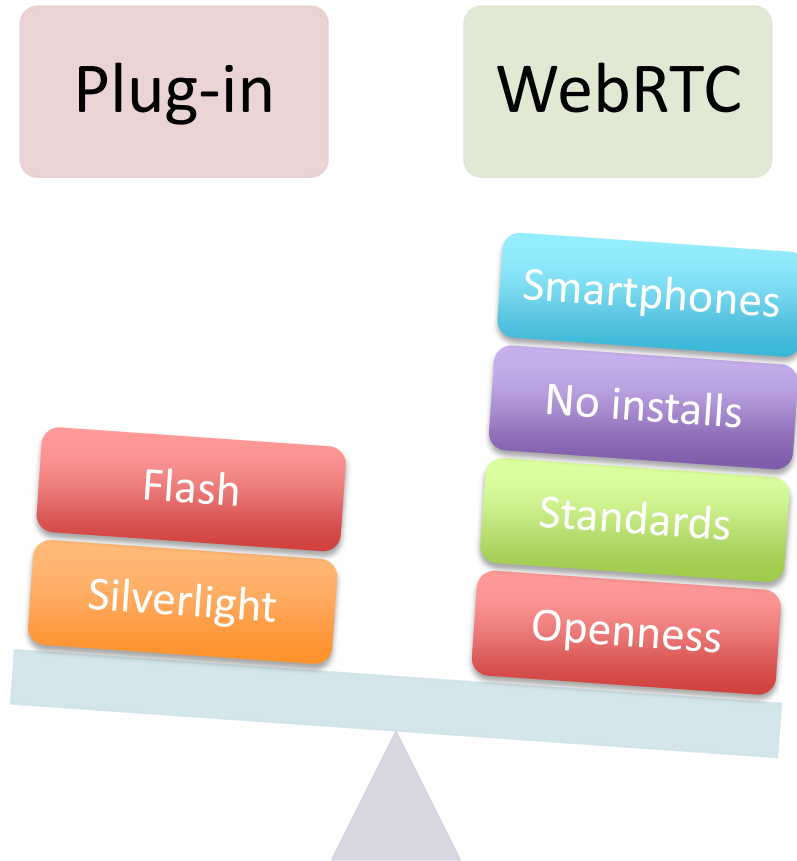
WebRTC as an API

- Capabilities are exposed to web developers in an abstract manner and adapting to HTML5 philosophy
- Basing on standards: WebRTC Working Group of the W3C for JavaScript APIs

Why WebRTC



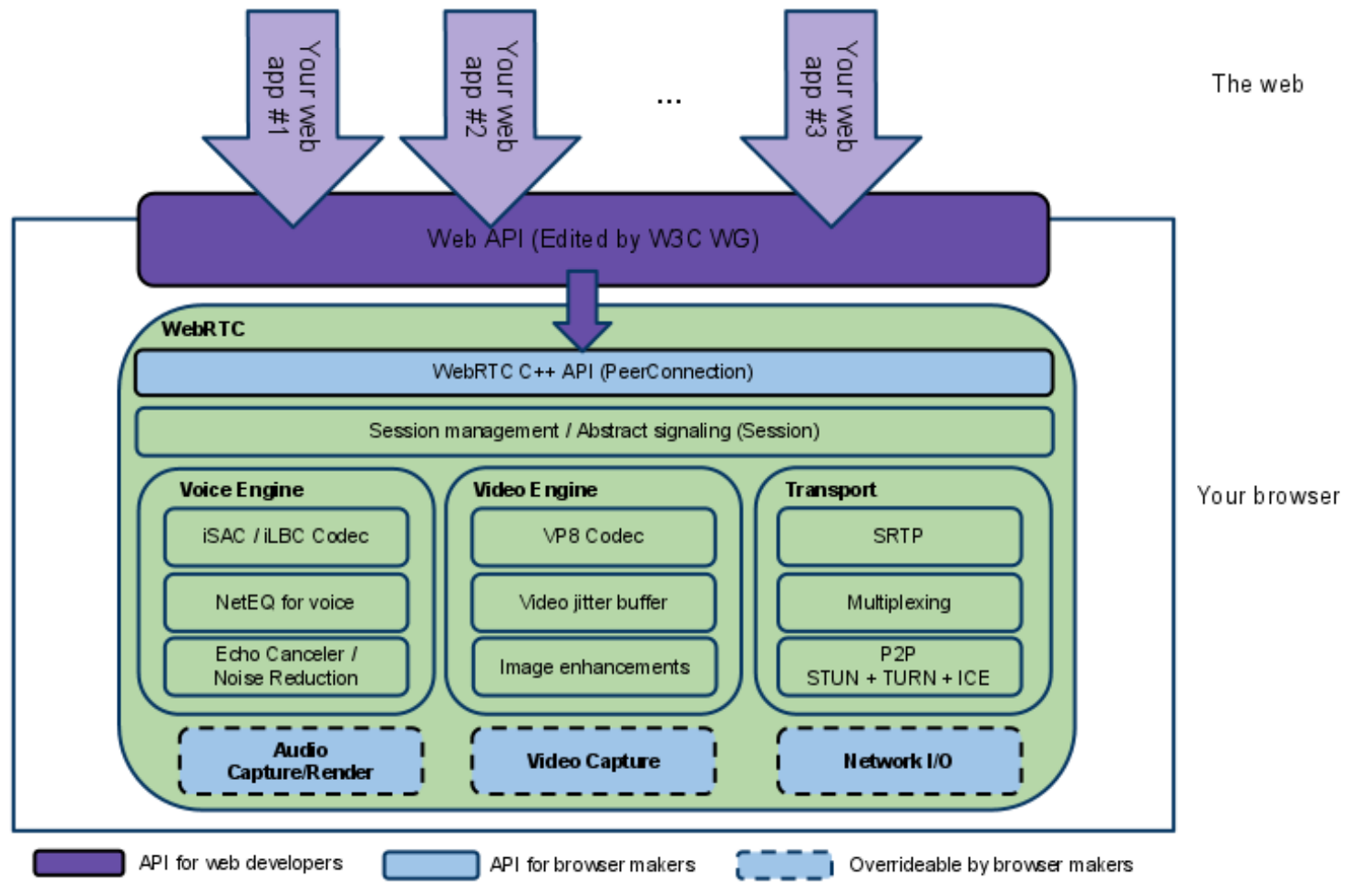
Why WebRTC



Who is who in WebRTC

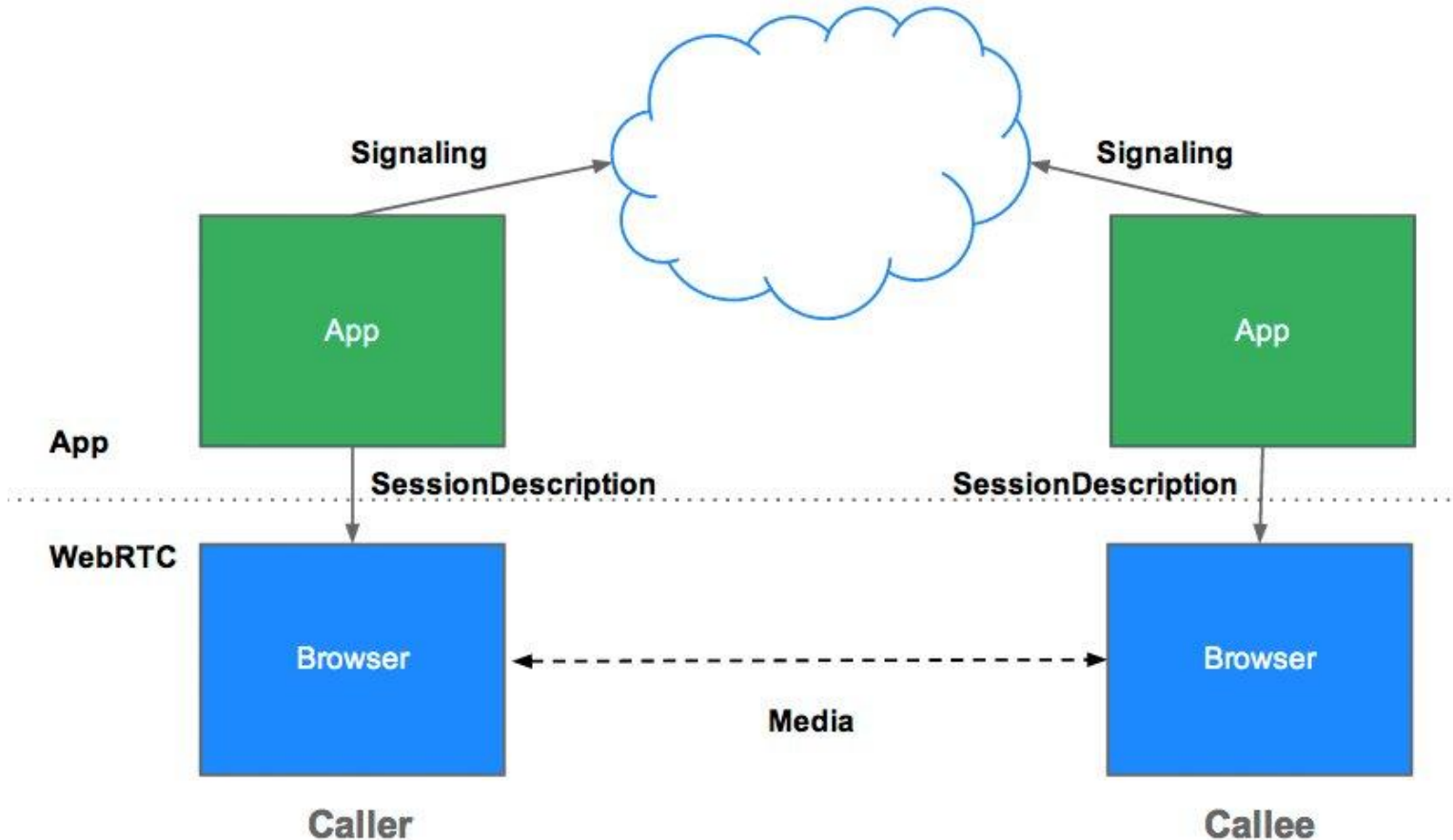


What's WebRTC: browser architecture



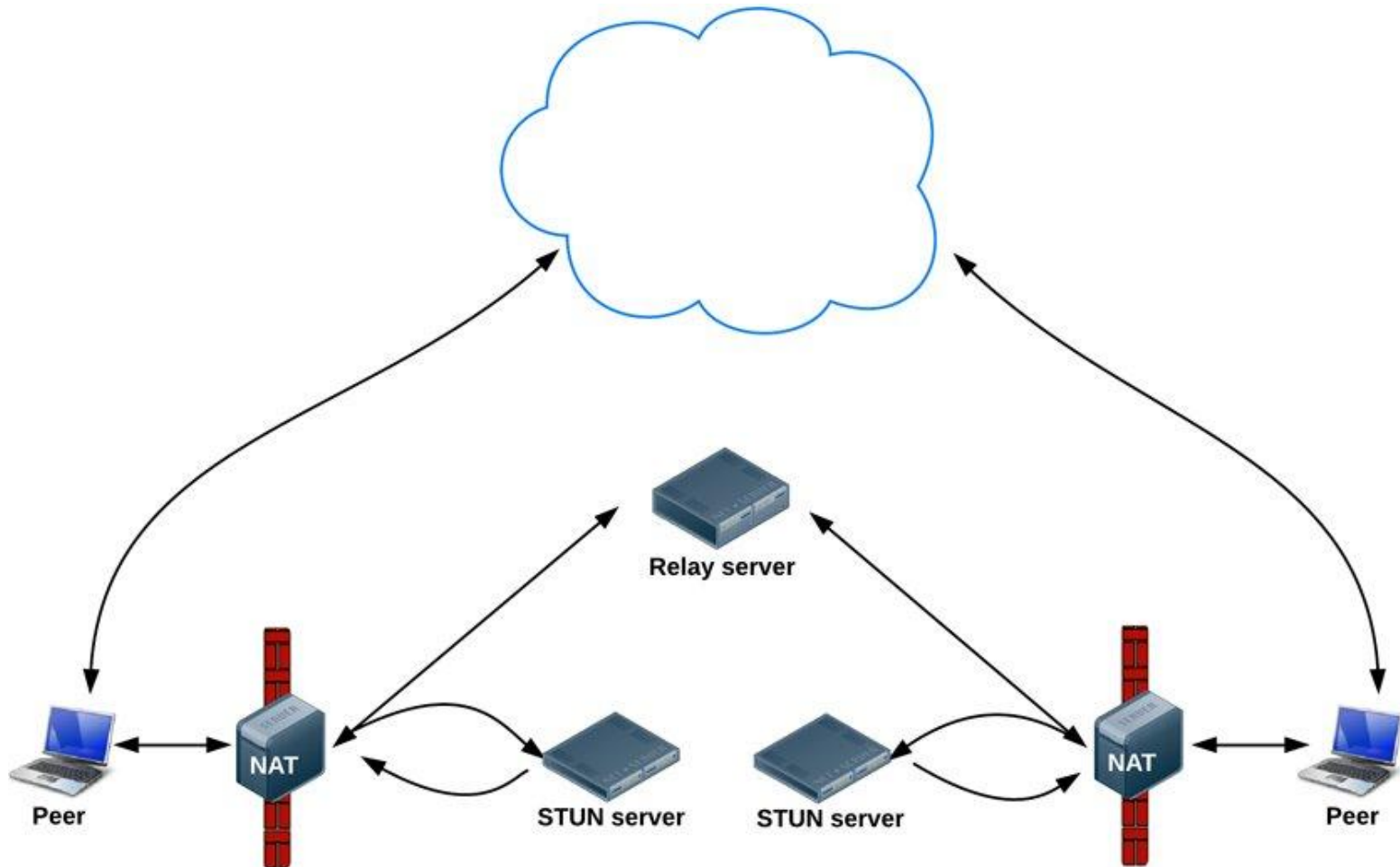
*This image has been borrowed from: <http://www.html5rocks.com/en/tutorials/webrtc/basics/>

WebRTC: P2P communications



*This image has been borrowed from: <http://www.html5rocks.com/en/tutorials/webrtc/basics/>

WebRTC: NATs



*This image has been borrowed from: <http://www.html5rocks.com/en/tutorials/webrtc/basics/>

Developing WebRTC apps

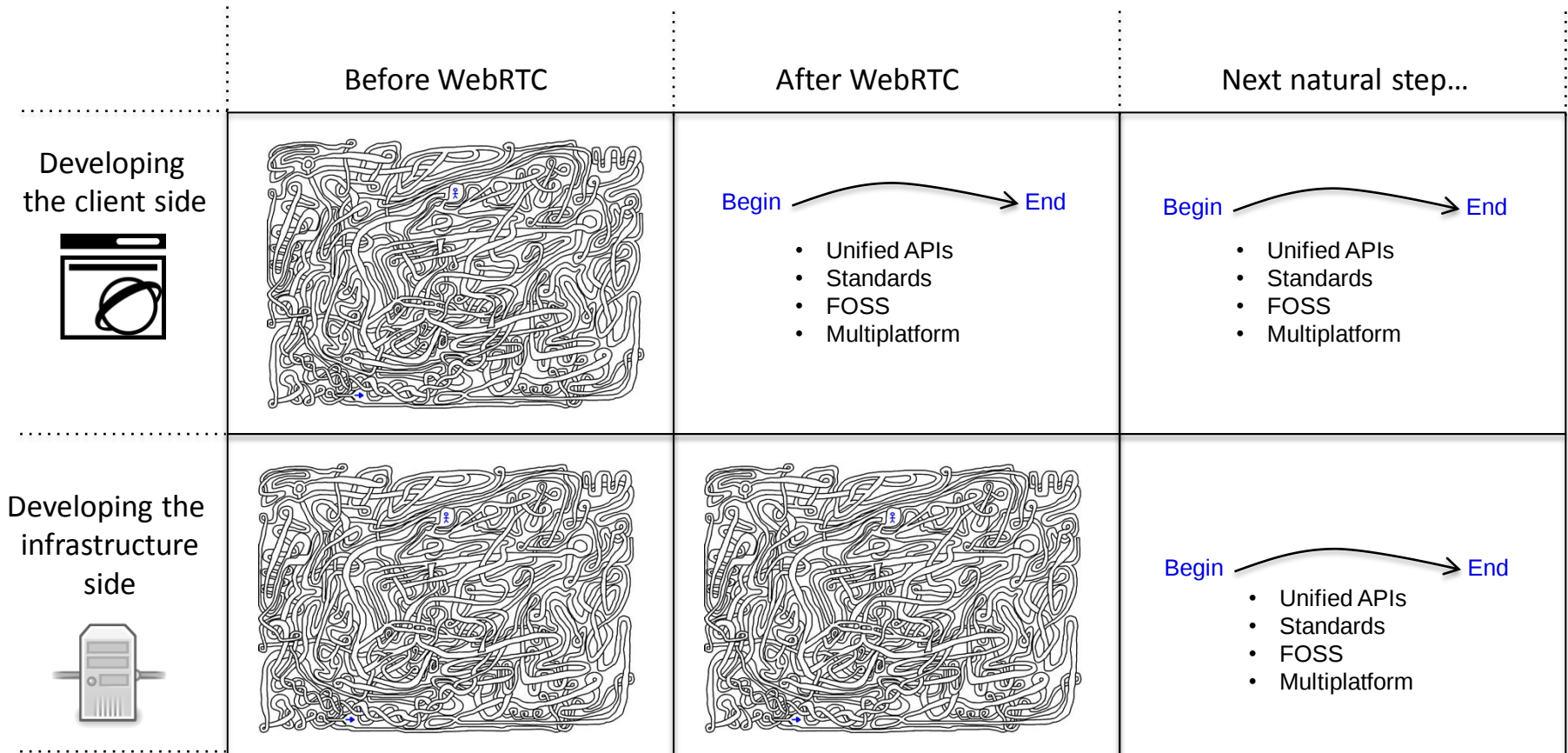
- <http://io13webrtc.appspot.com/#9>
- [http://www.html5rocks.com/en/tutorials/web
rtc/basics/](http://www.html5rocks.com/en/tutorials/webrtc/basics/)

Example

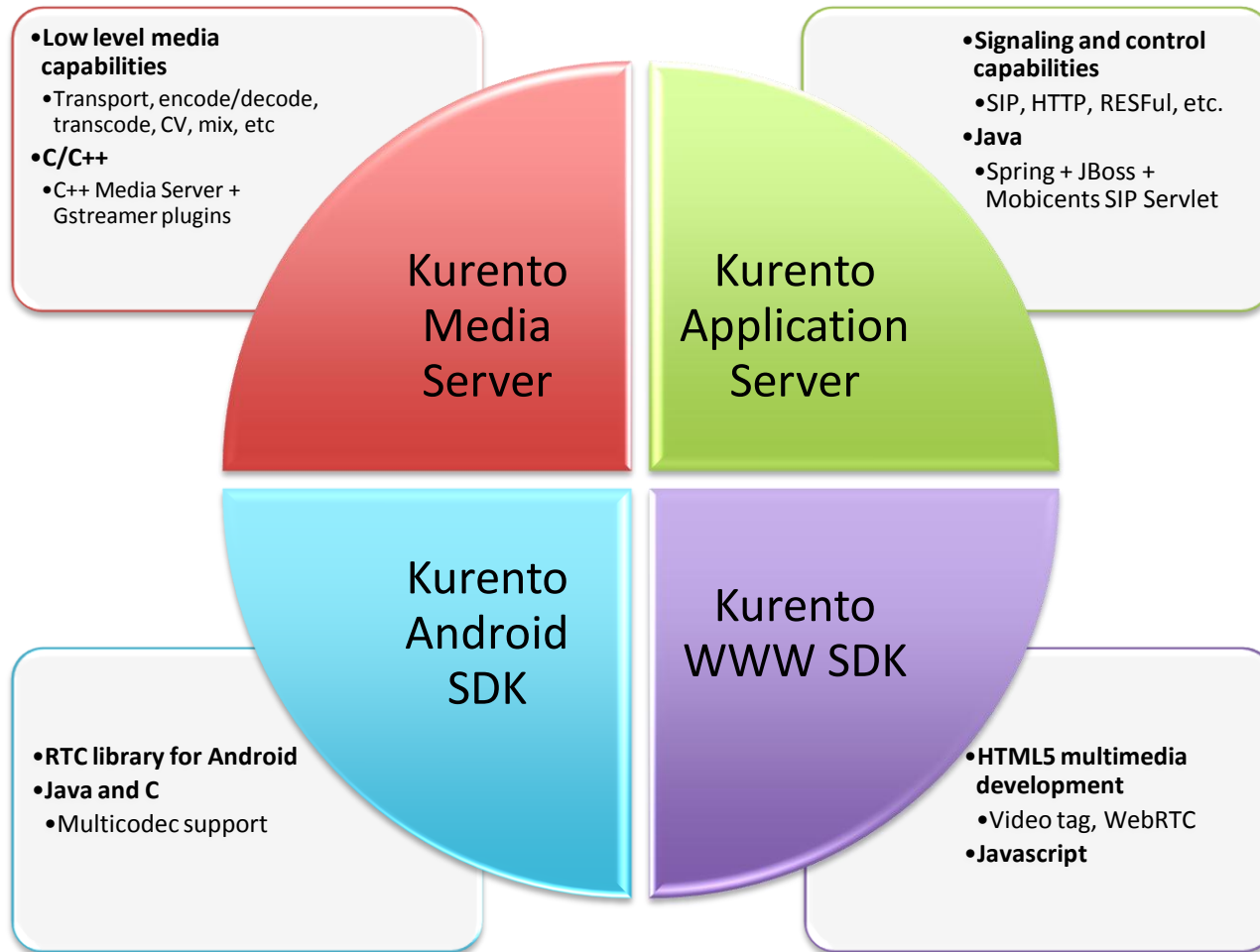
- <https://talky.io/>
- <https://www.cubeslam.com>

Why Kurento?

WWW RTC developer experience



Kurento media framework components



Kurento Application Server: extending the WWW development model



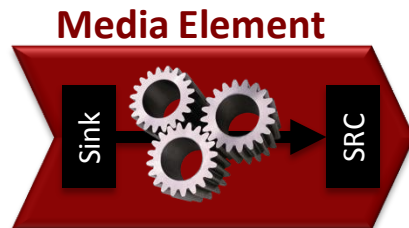
Intuition behind
traditional WWW
Applications
(Servlets, ASP, PHP,
Rails, etc.)

Intuition behind
Kurento
development APIs:
Multimedia RTC is just
another feature of your
application

Media API: media elements and media pipelines

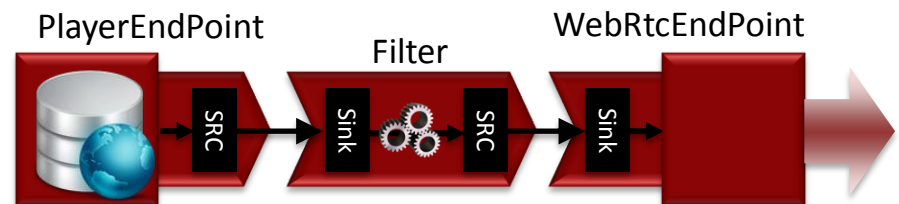
■ Media Element

- Provides a specific media functionality
 - › Building block
 - › Send/receive media
 - › Process media
 - › Transform media
- The Media API provides a toolbox of media elements ready to be used.
- New media elements can be added

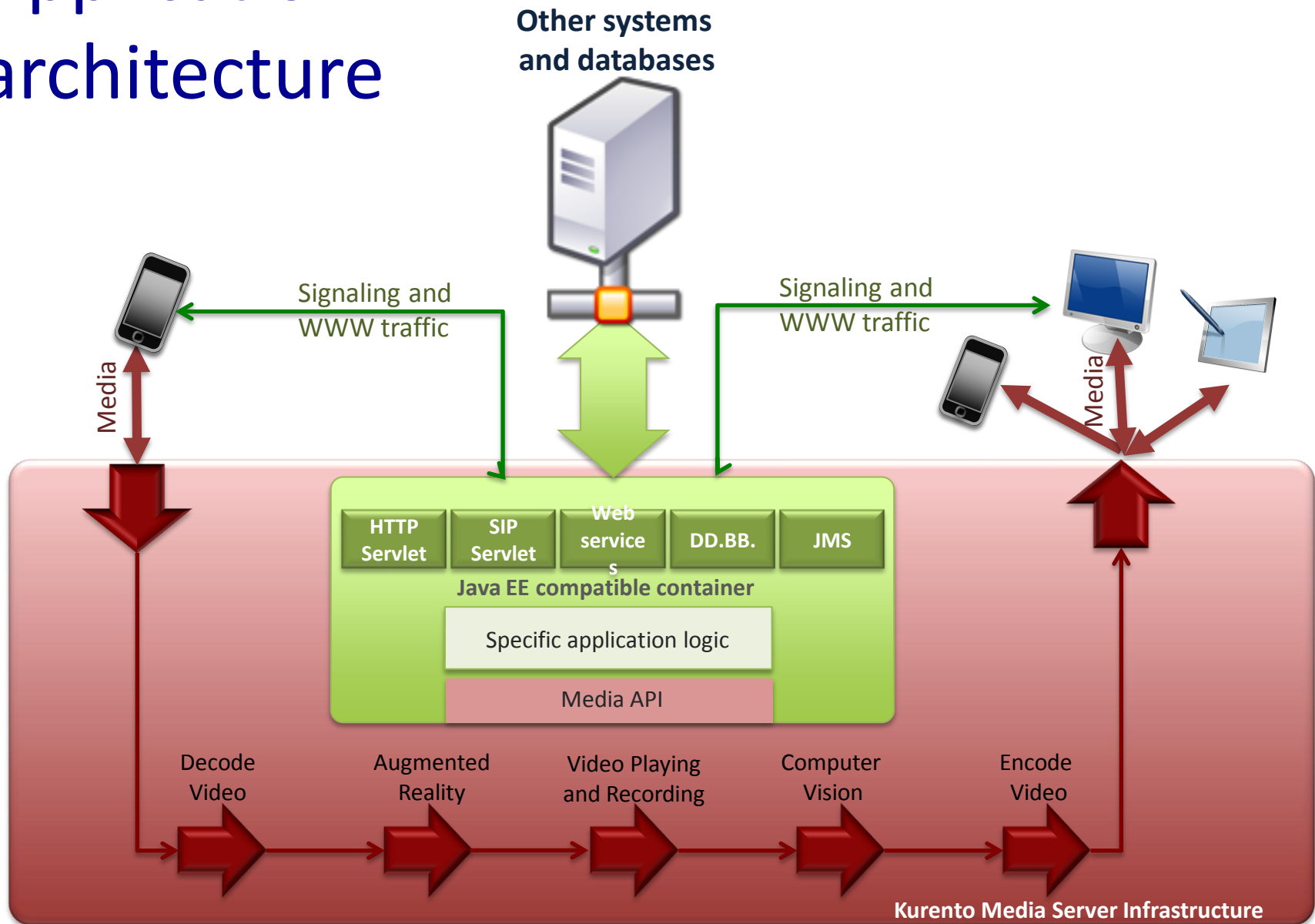


■ Media pipeline

- Chain of media elements implementing the desired media functionality.
- The Media API provides the capability of creating media pipelines by joining media elements of the toolbox



Application architecture

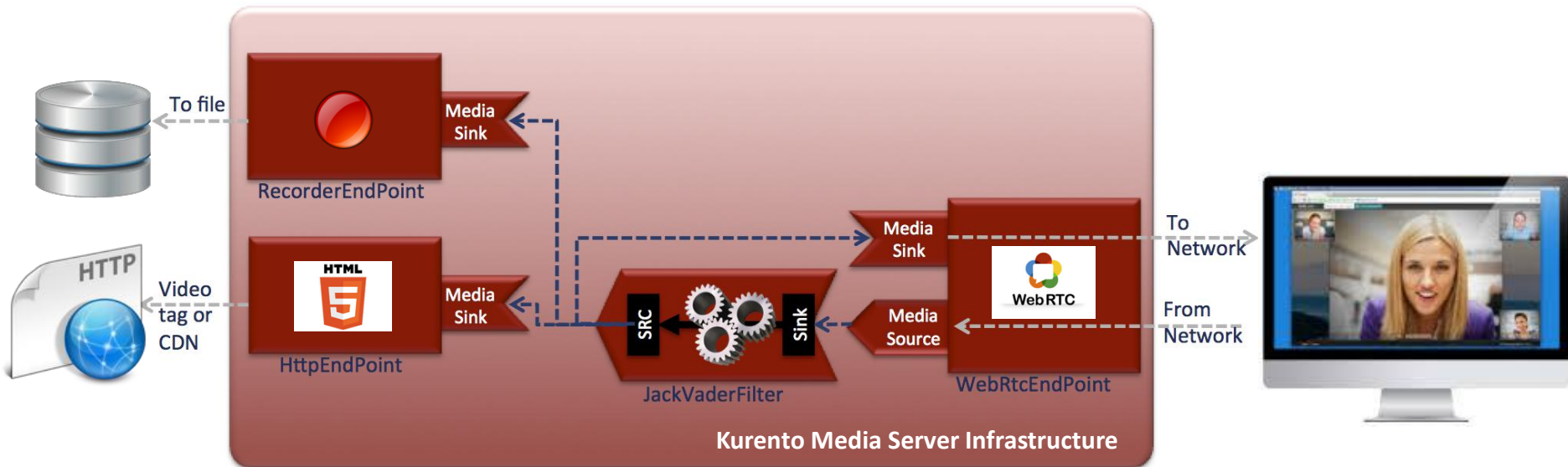


Possible use cases: just integrate with Java EE and GStreamer

- **Verticals**
 - **E-Health**
 - Kurento + HAPI (<http://hl7api.sourceforge.net/>)
 - P2D video conferences as Electronic Health Records
 - **Smart cities**
 - Kurento + NGSI + OpenCV + Google Maps
 - City crowds movement tracking
 - Traffic density tracking
- **Telco infrastructures**
 - **Kurento + Mobicents**
 - IMS application server
- **B2B & B2C WWW RTC**
 - **Kurento + CRM APIs**
 - Enriched video conferencing with customer personal data
 - **Kurento + ESB**
 - Billing, video event processing, physical security, etc.



Application example: requirements



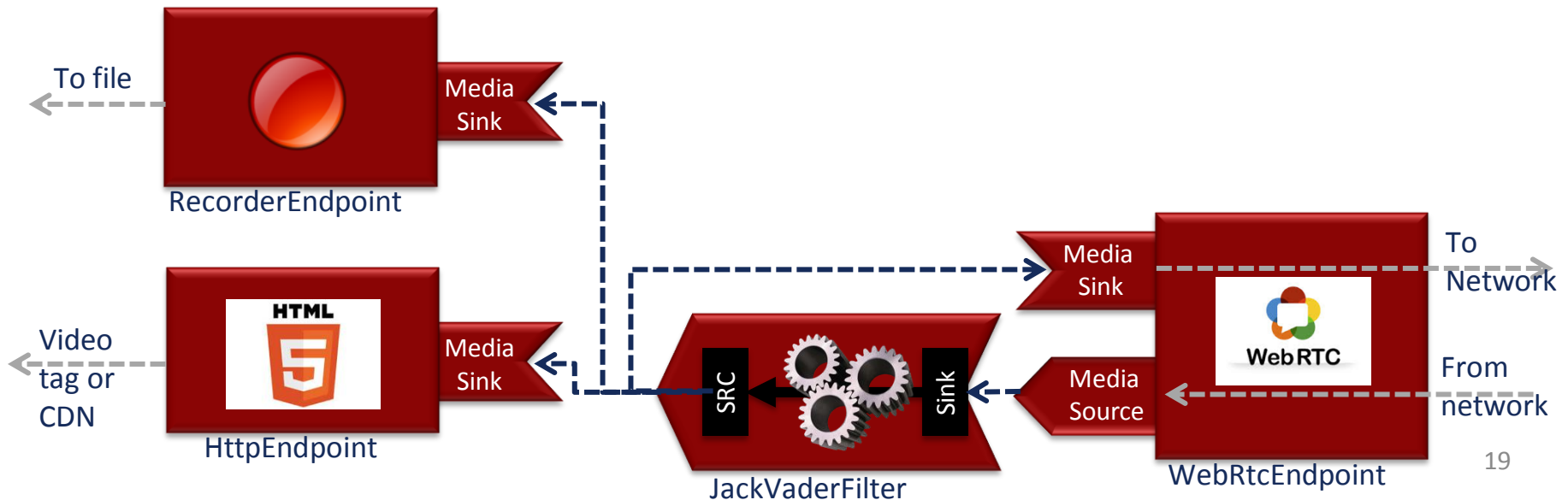
Application example: code

```
@WebRtcMediaService(name = "MyWebRtcService", path = "/pathToService")
public class MyWebRtcService implements WebRtcMediaHandler {

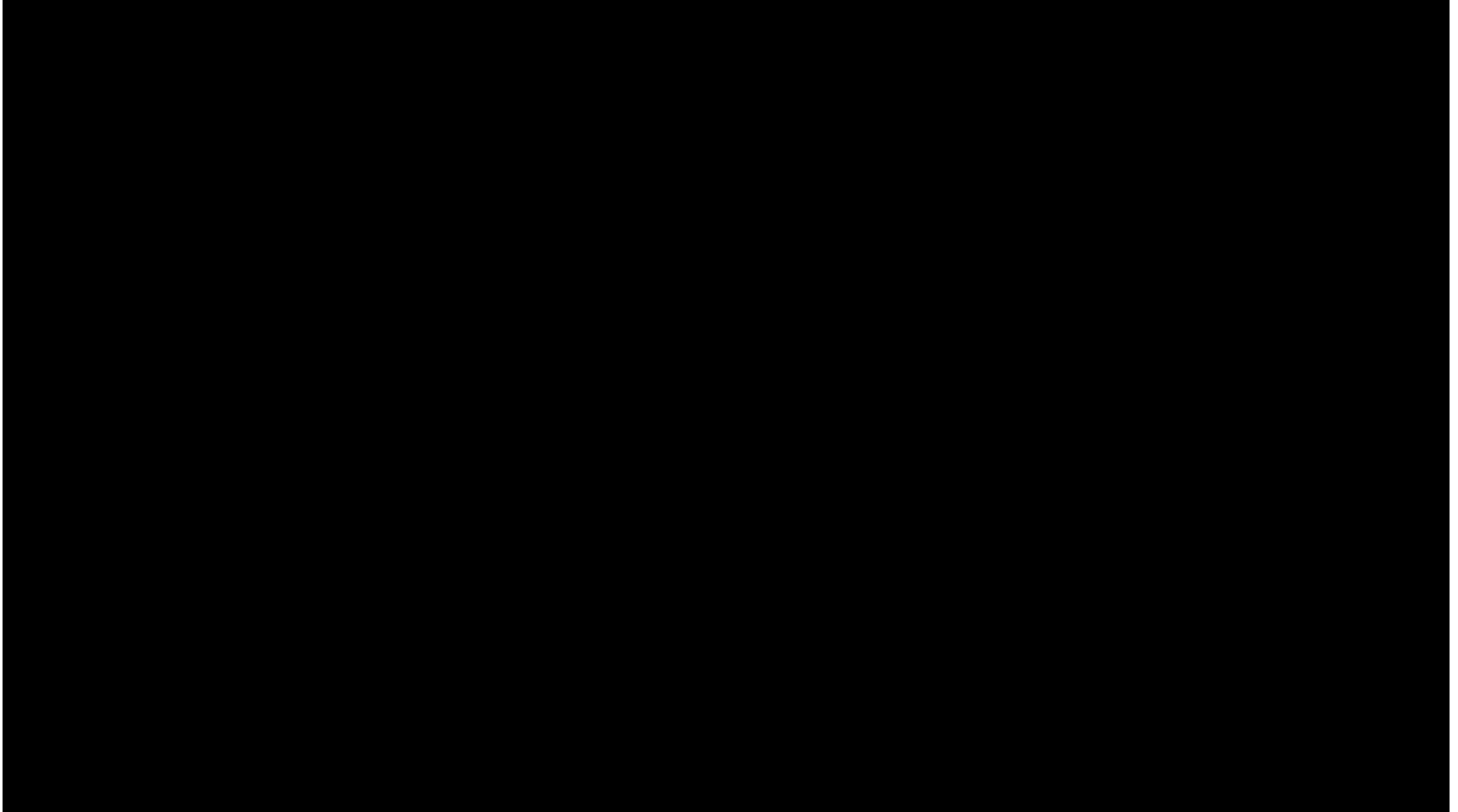
    public void onMediaRequest(WebRtcMediaRequest request) {
        // I can authenticate using any of the Java EE available mechanisms
        MediaPipelineFactory mpf = request.getMediaPipelineFactory();
        MediaPipeline mp = mpf.createMediaPipeline();

        // I could decide the type of processing connecting to a DDBB
        JackVaderFilter filter = mp.newFilter().withType(JackVaderFilter.class).build();
        RecorderEndpoint recorder = mp.newRecorderEndpoint().withUri("file:///myFile.webm");
        filter.connect(recorder);
        HttpEndpoint httpEndpoint = mp.newHttpEndpoint().build();
        filter.connect(httpEndpoint); // I could connect only audio or video separately

        request.startMedia(filter, filter);
    }
}
```



Application example: result



Media element toolbox

Transport

- WebRtcEndpoint
- RtpEndpoint
- HttpEndpoint

Repository

- PlayerEndPoint
- RecorderEndPoint

Group communications

- MainMixer
- GridMixer
- RoundRobinMixer

Filters

- FaceRecognitionFilter (events)
- JackVaderFilter
- QR/Barcode detector
- PlateRecognitionFilter (events)
- ColorTrackingFilter (events)

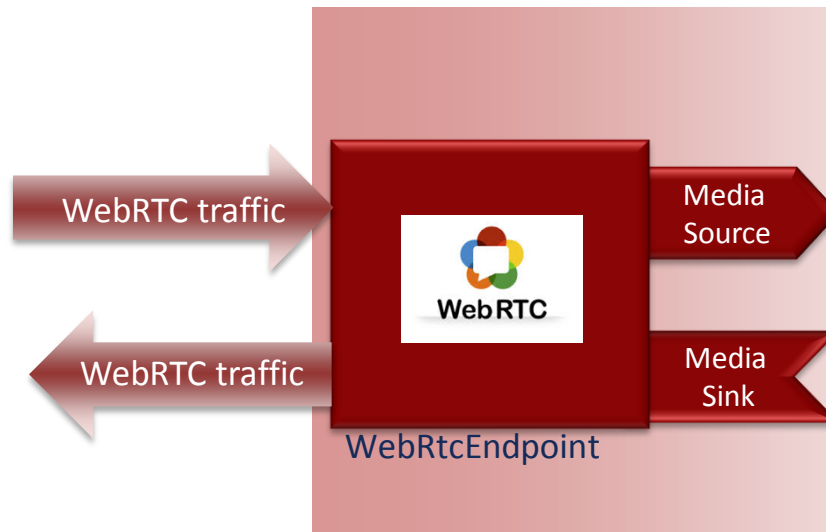
And growing ...

Available as part of



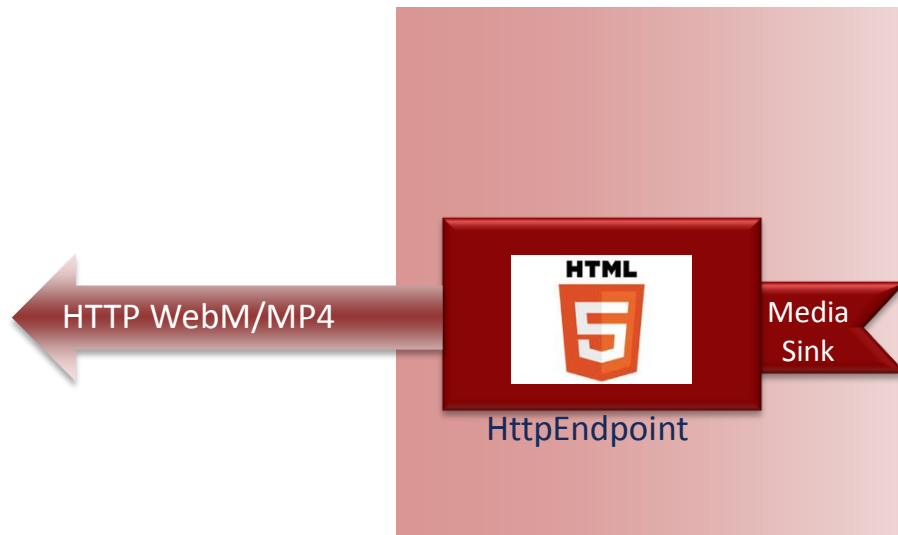
Media elements: WebRtcEndpoint

- Full implementation of the RTCWeb protocol stack
 - SRTP
 - ICE
 - DTLS
- Allow sending and receiving WebRTC flows at the media server infrastructure



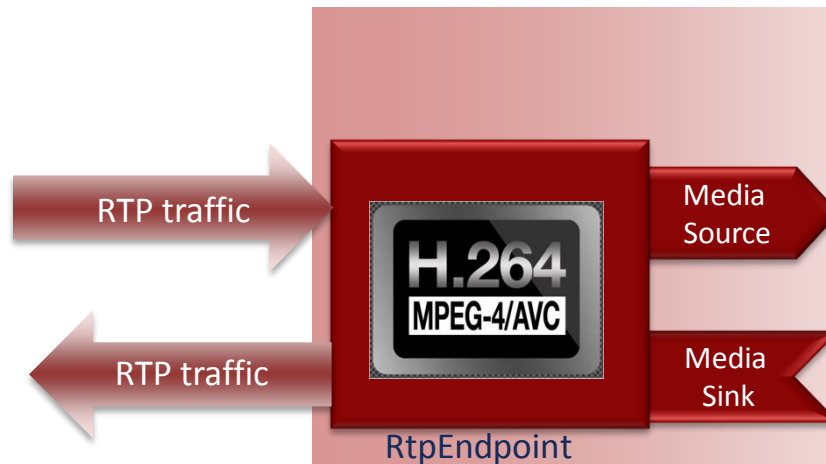
Media elements: HttpEndpoint

- Media downloading compatible with the HTML5 video tag
 - WebM (Chrome, Firefox)
 - MP4 (Chrome, Firefox, IE, Safari)
- Media uploading compatible with HTML file input tag
 - Multipart support



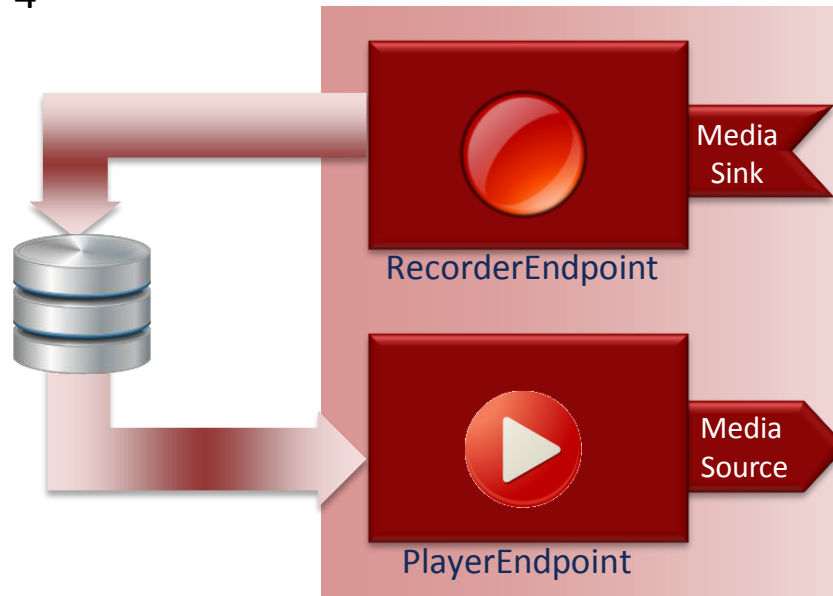
Media elements: RtpEndpoint

- Full-duplex RTP multimedia exchange
 - H.264
 - H.263
 - VP8
 - Many different audio codecs supported



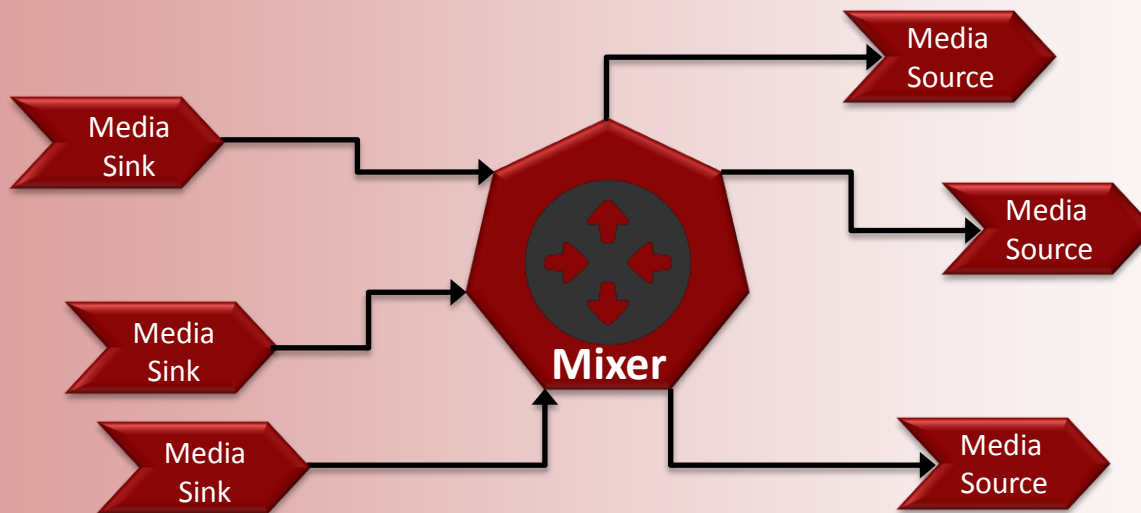
Media elements: UriEndpoints

- **PayerEndpoint**
 - Play media from file or URL
 - Support for most popular formats
- **RecorderEndpoint**
 - Record media to file or URL
 - WebM
 - MP4



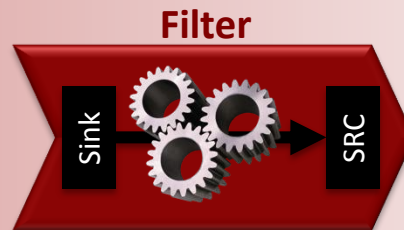
Media elements: Mixers (in progress)

- Make possible group communications
 - ForwardingMixer
 - One-to-many replication of flows
 - A source can be assigned to any of the sinks
 - Multiple sources supported
 - MainMixer
 - Mixes media
 - A source can be assigned to a combination of sinks
 - Multiple sources supported



Filters

- Seamless integration into OpenCV
 - Face recognition
 - Augmented reality
 - Subtitle adding
 - Color manipulation
 - QR detection
 - People counter
 - Plate recognition
 - Etc.

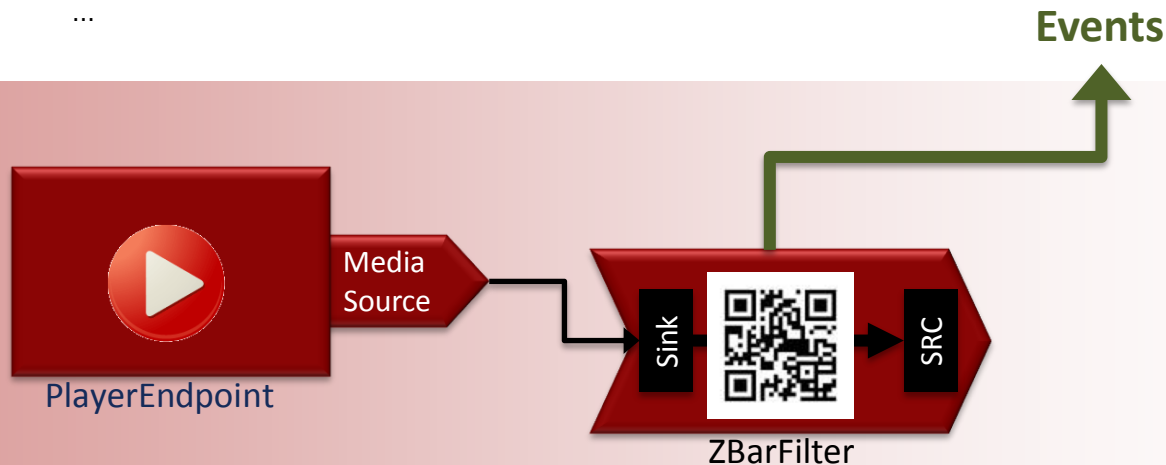


Filter with events

- Filters can provide events to the application
 - Events are generated at the media server
 - Events can be propagated to the client app
- Code example:

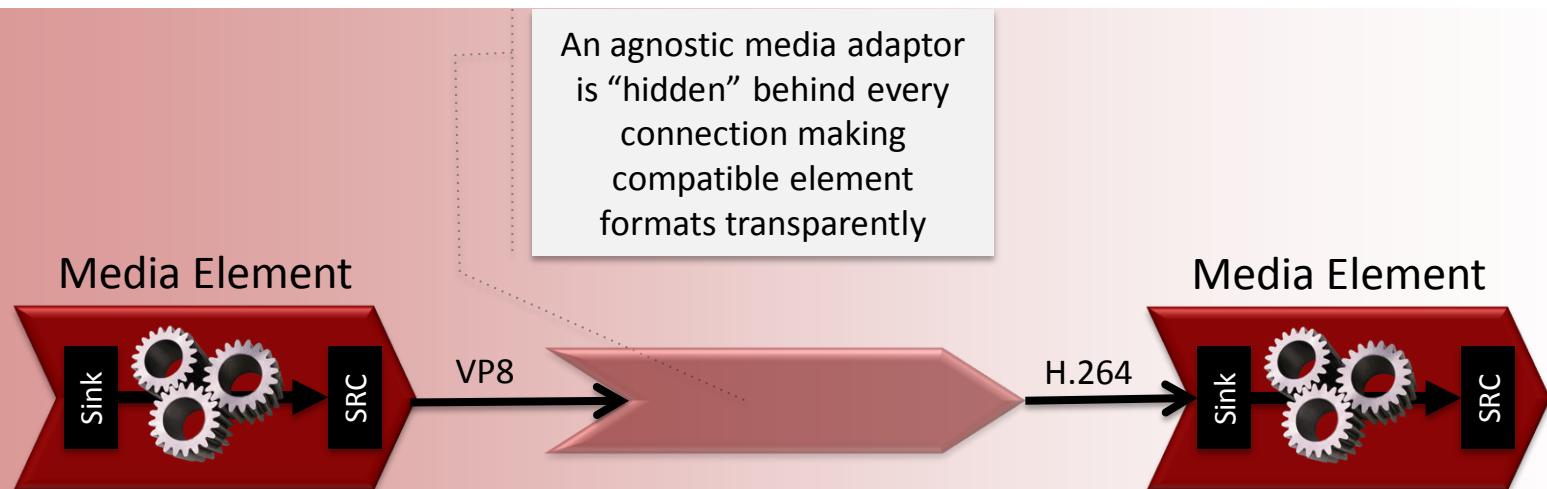
```
MediaPipeline mp = mpf.create();
PlayerEndPoint playerEndPoint = mp.newPlayerEndPoint(
    "https://ci.kurento.com/video/barcodes.webm").build();
ZBarFilter filter = mp.newZBarFilter().build();
playerEndPoint.connect(filter);

filter.addCodeFoundDataListener(new MediaEventListener<CodeFoundEvent>() {
    @Override
    public void onEvent(CodeFoundEvent event) {
        session.publishEvent(new ContentEvent(event.getType(),
            event.getValue()));
    }
    ...
});
```

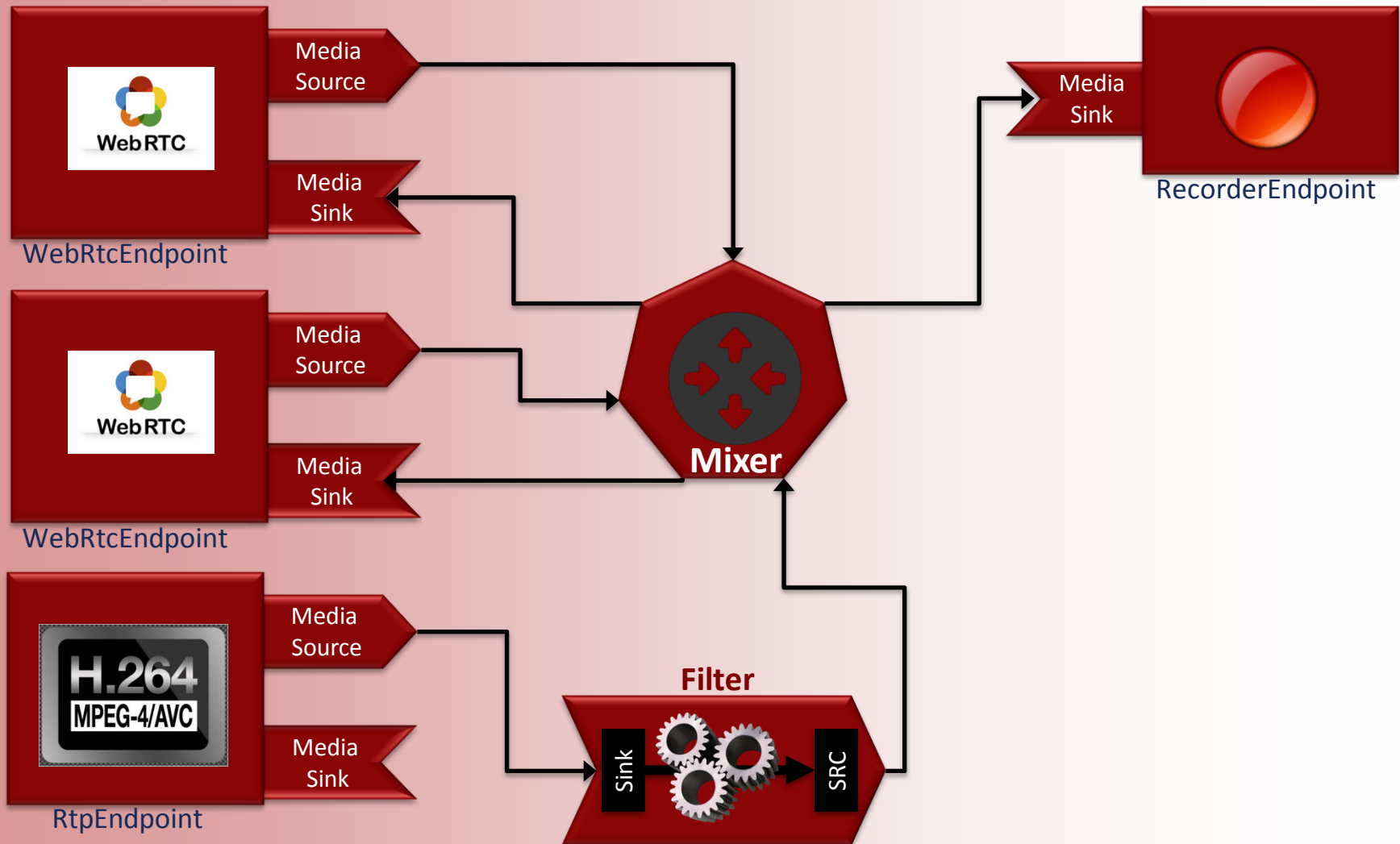


The magic of pipelines: Transparent media adaptation

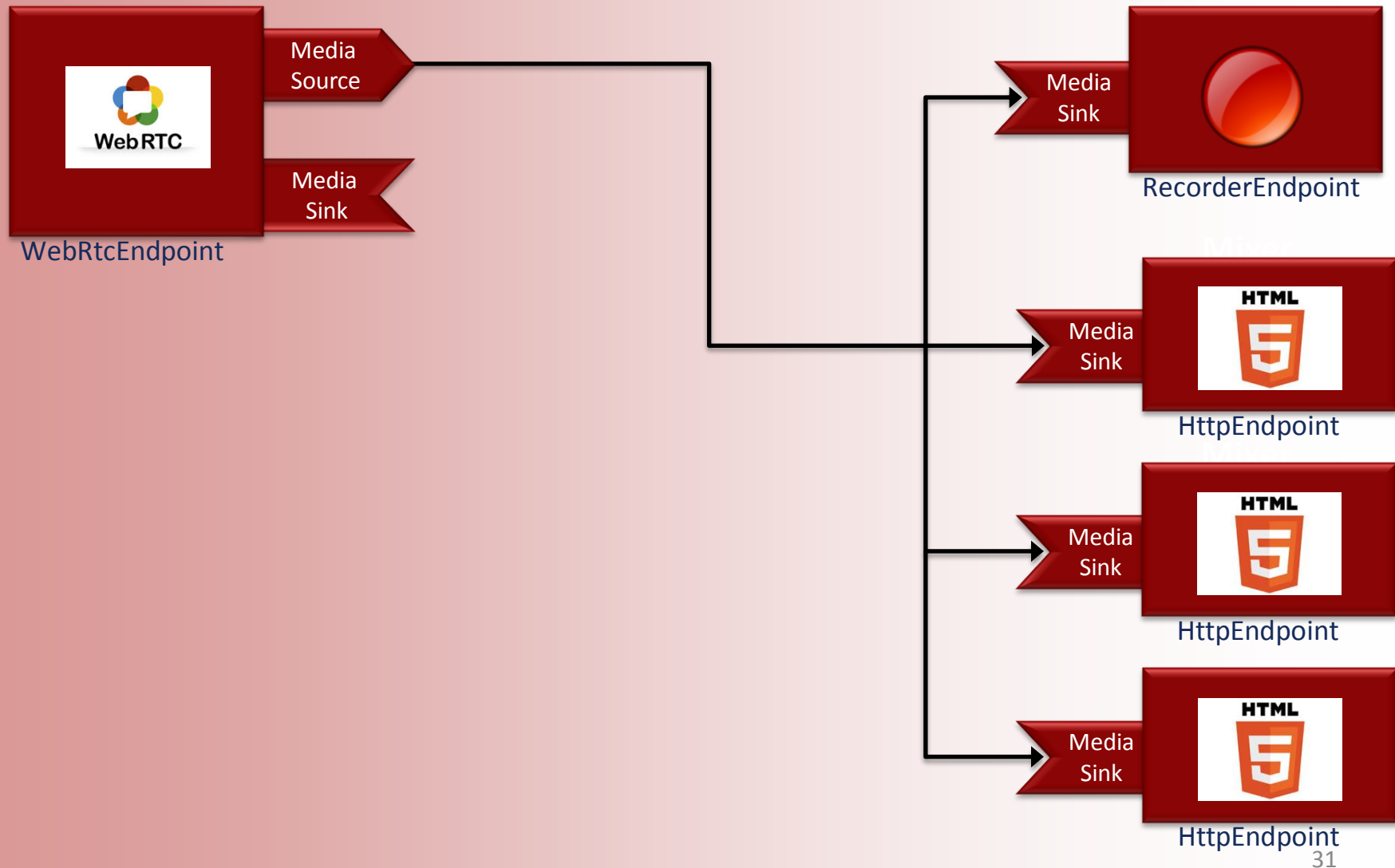
- **Agnostic media adaptor**
 - Acts every time a source is connected to a sink
 - Adapts media formats as required by the involved media elements
 - 100% transparent for the application developer



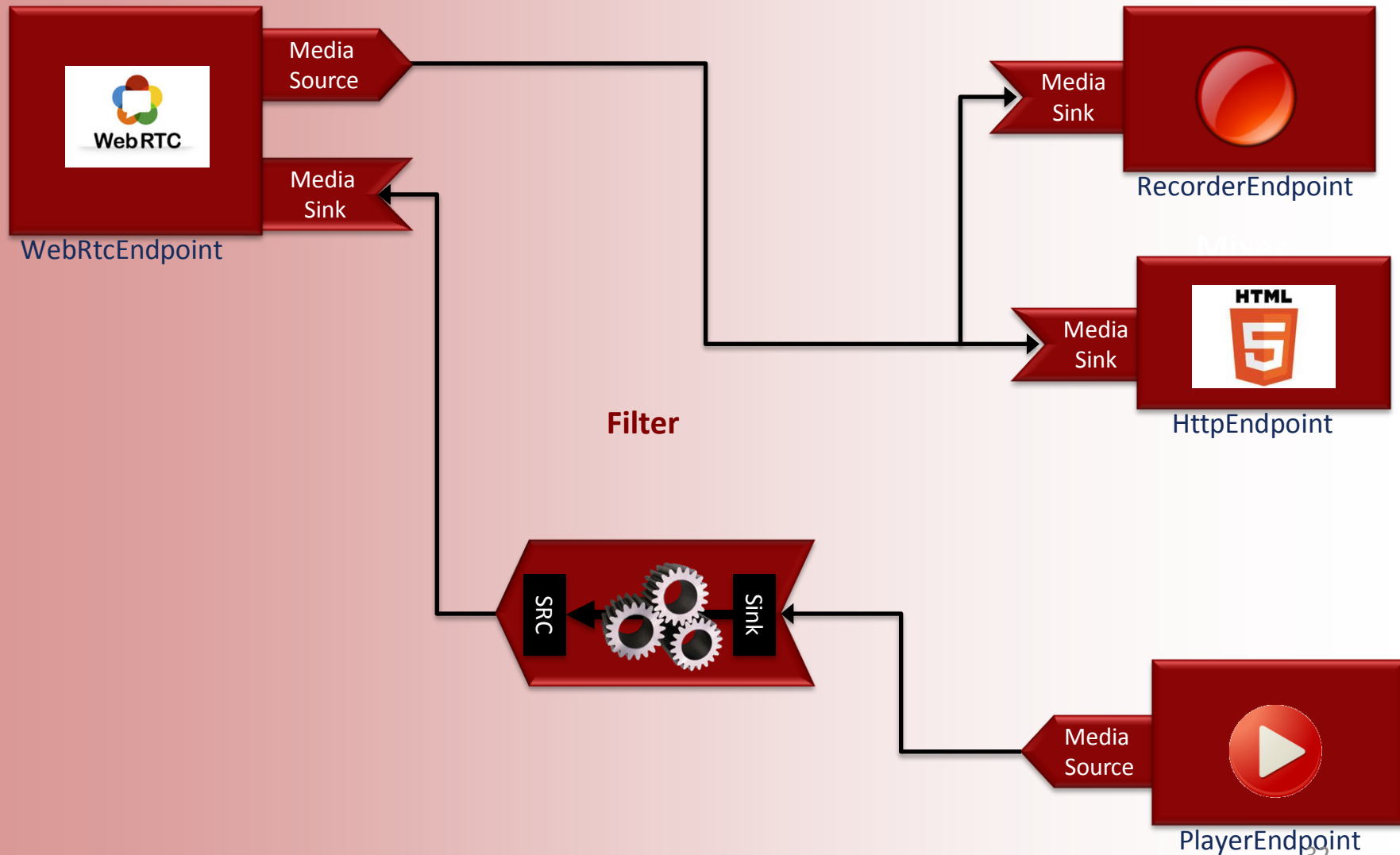
Complex examples: Heterogeneous group communications



Complex examples: WebRTC to HTTP

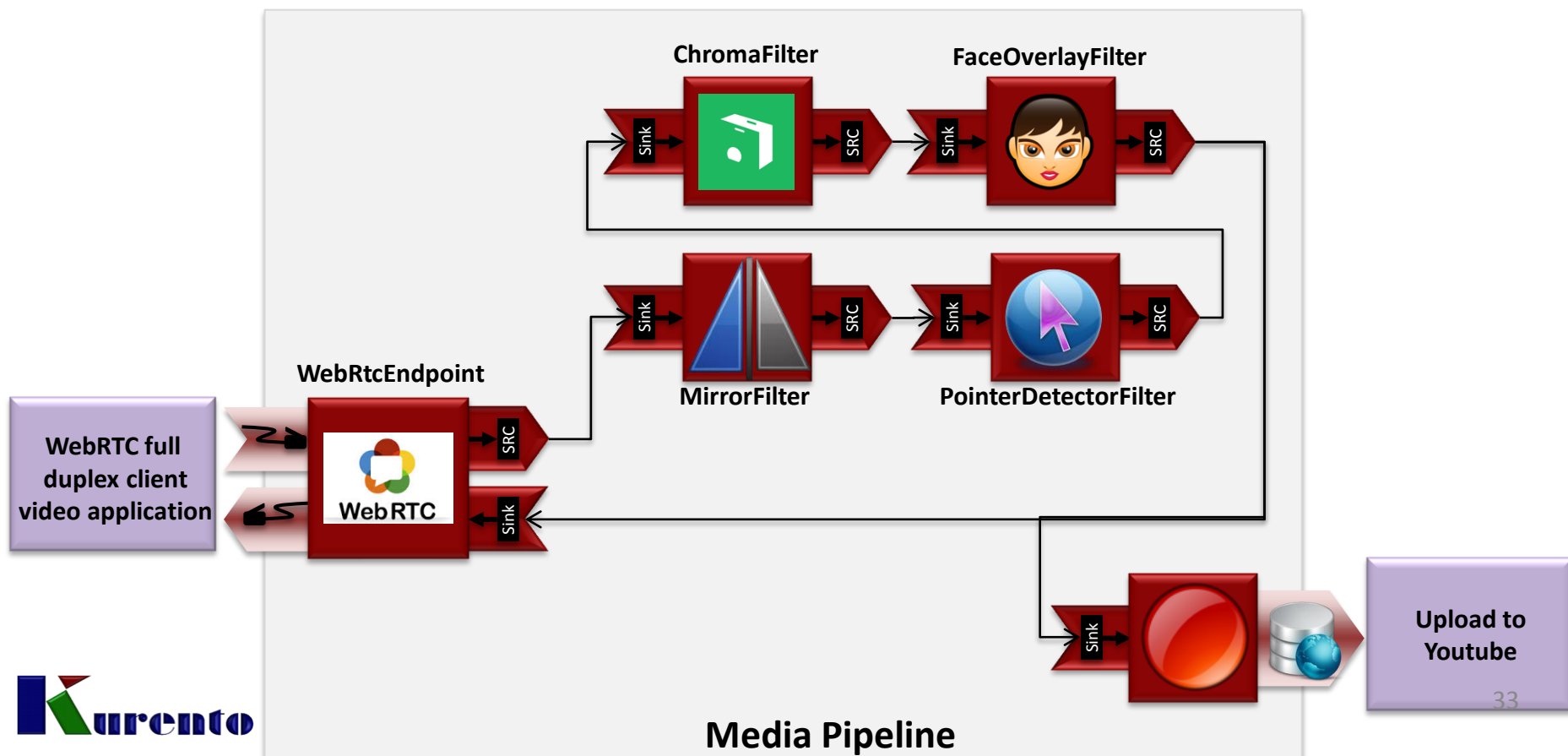


Complex examples: WebRTC Playing



WebRTC CV game

- <http://www.youtube.com/watch?v=CRqT7Q9KkRY>



Collaborations welcome
<http://www.github.com/kurento>

Thank you very much for your attention

Complains, suggestions and comments can be sent to:

Luis López

lulop@kurento.org