Nubomedia: the cloud infrastructure for WebRTC and IMS multimedia real-time communications

Luis Lopez
lulop@kurento.org
Human communications
The value of technology

DISTANCE, I HATE YOU.
Multimedia communication technologies

- 5G
  - HD 3D Video
- 4G
  - HD Video
- 3G
  - Video
- 2G
  - Audio

http://www.kurento.org
The Kurento vision

From this ...

... to this

processing

Knowledge

http://www.kurento.org
WebRTC: present and future

- Before WebRTC
  - Developing the client side
  - Developing the infrastructure side

- After WebRTC
  - Unified APIs
  - Standards
  - FOSS
  - Multiplatform

- Next natural step...
  - Unified APIs
  - Standards
  - FOSS
  - Multiplatform

http://www.kurento.org
WebRTC infrastructures

Peer-to-Peer WebRTC Application (without media infrastructure)

WebRTC Application based on media infrastructure

http://www.kurento.org
Function of WebRTC media servers

Transcoding media server

VP8 \rightarrow H.264

MCU/SFU media server

Recording media server

http://www.kurento.org
Kurento as a WebRTC media server

**What common WebRTC Media Servers do:**
- Transcoding
- MCU
- Recording

**What future Media Servers will do:**
- Flexible processing
- Augmented reality
- Blending
- Mixing
- Analyzing
- Etc.

---

**Current media servers**

- Media is here
- Transcoding
- MCU/SFU
- Recording

- Media goes there

**Future media servers**

- Media is here
- Transcoding, MCU/SFU, Recording, Enrich, Augment, Analyze, Combine, Transform, Adapt, ...
- Context
- Content
- Commands

- Media Events
- Rich Media goes there
Why is this important

Advertising
Broadcasting
Gaming
eLearning
Blue Ocean

Content and entertainment

WebRTC

IoT
M2M
P2M
Multimedia

Making calls
WARNING! Overcrowded

Smart cities
Emergencies
Security
Blue Ocean
WebRTC as a disruptive technology

General purpose
RTC multimedia Services

Specific purpose RTC multimedia services

Disruption direction of WebRTC

Commodity, no business opportunities here

Who cares about the immense minority?

Many business opportuniies here
Kurento Media Server

**Media Element**
- Provides a specific media functionality
  - Send/receive media
  - Process media
  - Transform media
- Exchange media through
  - Sources
  - Sinks

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

```
Media Element
```

```
Media pipeline
```

http://www.kurento.org
Creating applications basing on Kurento Media Server

Toolbox of media elements
Creating applications with Kurento
it seems dangerous...

LETS DO IT!!
What you should learn first

• WebRTC basics

• Signaling basics (STUN/TURN)
Starting with Kurento

• Kurento official documentation
  – http://www.kurento.org/documentation

• Kurento FIWARE documentation
  – Catalogue site
    • http://catalogue.fiware.org/enablers/stream-oriented-kurento
  – Documentation
    • http://catalogue.fiware.org/enablers/stream-oriented-kurento/documentation
Installing Kurento Media Server

• **Requirements**
  – Ubuntu 14.04 box (sudo)
  – Internet connectivity

• **Install**
  – `sudo add-apt-repository ppa:kurento/kurento`
  – `sudo apt-get update`
  – `sudo apt-get install kurento-media-server`

• **Launch**
  – `sudo service kurento-media-server start`
Kurento “Hello World!”

http://www.kurento.org
Kurento “Hello World!”

• Tutorial
  – http://www.kurento.org/docs/current/tutorials/js/tutorial-1-helloworld.html

• Code

• Video
  – https://www.youtube.com/watch?v=vGEnkSOp_xc
Understanding this example

Browser

Application Code

kurento-utils

getUserMedia API

PeerConnection API

Kurento-client-js

WebSocket API

Media

Signaling

Kurento Media Server

http://www.kurento.org
Recording and playing

http://www.kurento.org
Recording and playing

• Get code here
  – WARNING: Example using generators!!

• Video
  – https://www.youtube.com/watch?v=rDd2NjFXcS0
JavaScript Generators

• Generators
  – Black magic for avoiding callback hell
  – Program asynchronously with synchronous philosophy

• Warning
  – “Enable Experimental JavaScript” flag
One-to-many example

WebRTC streaming

User 1

User 2

User 3

User 4

WebRtcEndpoint 1

WebRtcEndpoint 2

WebRtcEndpoint 3

WebRtcEndpoint 4

Media Pipeline
One-to-many

• Java EE
  – [http://www.kurento.org/docs/current/tutorials/java/tutorial-3-one2many.html](http://www.kurento.org/docs/current/tutorials/java/tutorial-3-one2many.html)

• Node.js
  – [http://www.kurento.org/docs/current/tutorials/node/tutorial-3-one2many.html](http://www.kurento.org/docs/current/tutorials/node/tutorial-3-one2many.html)
Face overlay example

- Browser JavaScript
- Java
- Node.js
  - [http://www.kurento.org/docs/current/tutorials/node/tutorial-3-one2many.html](http://www.kurento.org/docs/current/tutorials/node/tutorial-3-one2many.html)
- Video
  - [https://www.youtube.com/watch?v=h84HFkvWGgw](https://www.youtube.com/watch?v=h84HFkvWGgw)
Augmented Reality example

- Video
  - https://www.youtube.com/watch?v=J1Rg4PzeRKQ
Motion detector

- Video
  - https://www.youtube.com/watch?v=r91nExNEHiw
Crowd detector

• Video
  – https://www.youtube.com/watch?v=S6iWSCysgT0
Many other examples

• Face segmentator (aka get a Kiss)
  – https://www.youtube.com/watch?v=WmzzblZGDo

• Room communications
  – https://www.youtube.com/watch?v=hkT8fLROdwo

• B2B calls
  – https://www.youtube.com/watch?v=ocJBD08K6eM

• Etc.
Beyond media servers: WebRTC clouds and the problem of scalability

WebRTC Application based on media infrastructure
Cloud models for WebRTC infrastructures

IaaS
- **Provider**
  - Computing resources
- **Developer**
  - Installation
  - Administration
  - Security
  - Application logic

PaaS
- **Provider**
  - Development API
- **Developer**
  - Application logic

SaaS
- **Provider**
  - Service
- **Developer**
  - Nothing to do

http://www.kurento.org
WebRTC PaaS APIs: Requirements

- Requirements of WebRTC PaaS APIs
  - Functional requirements
    - Media transport
      - Media endpoint
      - Media replication
      - Media routing
    - Media persistence
      - Media storage
      - Media recovery
    - Media processing
      - Transcoding
      - Analysis
      - Augmentation
  - Non-functional requirements
    - Security
    - Dependability
    - Scalability

http://www.kurento.org
WebRTC multimedia session

- RFC 5117
  - A multimedia session is an association among a group of participants engaged in the communication via one or more RTP Sessions.

- Characterized by
  - Communication topology
    - Graph of multimedia flows
  - Multimedia processing
    - Function of each edge of the graph of media flows

http://www.kurento.org
Scalability of RTC multimedia services

- "Broadcasting Non-RTC"
- "Home-made RTC"
- "Beyond RTC calls"
- "RTC calls"

Number of concurrent sessions vs. Number of users per session

WebRTC PaaS

TV Broadcasting

CDNs

Media Server based Services

Phone system

http://www.kurento.org
Scalability of RTC multimedia services

- **“RTC calls”**
  - WebRTC PaaS
  - tokbox
  - skype
  - Phone system

- **“Beyond RTC calls”**
  - NUBOMEDIA

- **“Home-made RTC”**
  - Media Server based Services

- **“Broadcasting Non-RTC”**
  - TV Broadcasting
  - CDNs

Number of concurrent sessions vs. Number of users per session
The scalability problem in “call” clouds
Anatomy of WebRTC PaaS for call models: Flat Architecture

- Load Balancer Function
- Application Server Function
- Broker Function
- Media Server Function

Cloud Orchestrator

IaaS Cloud Manager

http://www.kurento.org
Cloud functions: IaaS manager

- **Function**
  - Provides APIs for IaaS management
    - Images
    - Instances
    - Storage
    - Metrics
    - Security
    - Etc.

- **Requires**
  - Physical infrastructure

**Cloud Orchestrator**

**Infrastructure as a Service APIs**

- Image Management
- Metrics and KPIs
- Security Management
- Computing Management
- Storage Management
- Networking Management

http://www.kurento.org
Cloud functions: Cloud Orchestrator

- **Function**
  - Lifecycle management of the platform
    - It acquires virtual resources and allocate them to the specific services
  - Runtime management with autoscaling
    - It scales out new service instances in situations of peak load
    - It scales in service instances whenever they are not required any longer

- **Requires**
  - Autoscaling rules
    - Ex. If average load is over 60% add two new instances

http://www.kurento.org
Media Server Function

- Function
  - Provides media capabilities
    - WebRTC transport
    - Recording
    - Transcoding
    - Etc.

- Requires
  - Control Protocol
  - Media Protocols
  - Media Codecs
Broker Function

- **Function**
  - Assigns “call” to specific media server instances
    - Give me a media server instance to take care of this call
  - “call” are never split among media servers

- **Requires**
  - Scheduling policy
    - Round robing
    - Random
    - Less load
    - Etc.
  - Registration of MSis
    - All media server instances need to be known by the broker

http://www.kurento.org
Application Server

Client request

- Signaling
  - Send/receive of signaling messages
- Security logic
  - Authentication, Authorization, Accounting
- PaaS API logic
  - Control of media server functions for providing API semantics

Requires
- Signaling protocol implementation
  - SIP, JSON, etc.
- Security rules
  - ACLs, CAP, etc.
- Specific logic
  - Media server dependent
Load balancer

- **Function**
  - Distributes client requests among available AS instances
  - Usually stateful
- **Requires**
  - Balancing policy
    - Round robin
    - Random
    - Less load
    - Etc.
- **Scaling needs**
  - Low

http://www.kurento.org
The flat Nubomedia implementation

Cloud Orchestrator

IaaS Cloud Manager

Load Balancer Function

Application Server Function

Broker Function

Media Server Function

Vert.x

Node.js

Erlang

NGINX

Heat

Kurento

OpenStack

http://www.kurento.org
Scalability of RTC multimedia services

- "Broadcasting Non-RTC"
- "Home-made RTC"
- "RTC calls"
- "Beyond RTC calls"

Number of users per session vs. Number of concurrent sessions
Beyond calls: convergence of broadcasting and phone-like services

Users in call

Monitoring users

Users can dynamically change their role

http://www.kurento.org
The scalability problem in “beyond call” clouds
Anatomy of WebRTC PaaS for call models: Hierarchical Architecture

![Anatomy of WebRTC PaaS](http://www.kurento.org)
Media Server Function

- **Function**
  - Provides elastics media capabilities
  - Strong dependencies among media server instances
  - Media servers connect following a specific topology

- **Requires**
  - Glue mechanism among media server instances
The elastic media server

- **Elasticity**
  - On the number of media pipelines
    - Number of concurrent sessions
  - On the number of elements per media pipeline
    - Number of concurrent users per session
  - Media Pipeline
    - Distributed media pipeline

- **Rigidity**
  - The media element is a monolithic (non-distributed) entity
Broker Function

- **Function**
  - Assigns “call legs” to specific media server instances
    - Give me a media server instance to take care of this call
    - “call” are split among different media server instances

- **Requires**
  - Scheduling policy
    - Topology aware
    - Network aware
    - SLA aware
    - Etc.
  - Very complex problem
    - Leg adding may require additional media server instances
    - Churn is very complex to manage
Hierarchical Nubomedia implementation:
Work in progress

- Vert.x
- Node.js
- Scala
- Nginx
- Python

Own implementation

Cloud Orchestrator

IaaS Cloud Manager

http://www.kurento.org