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

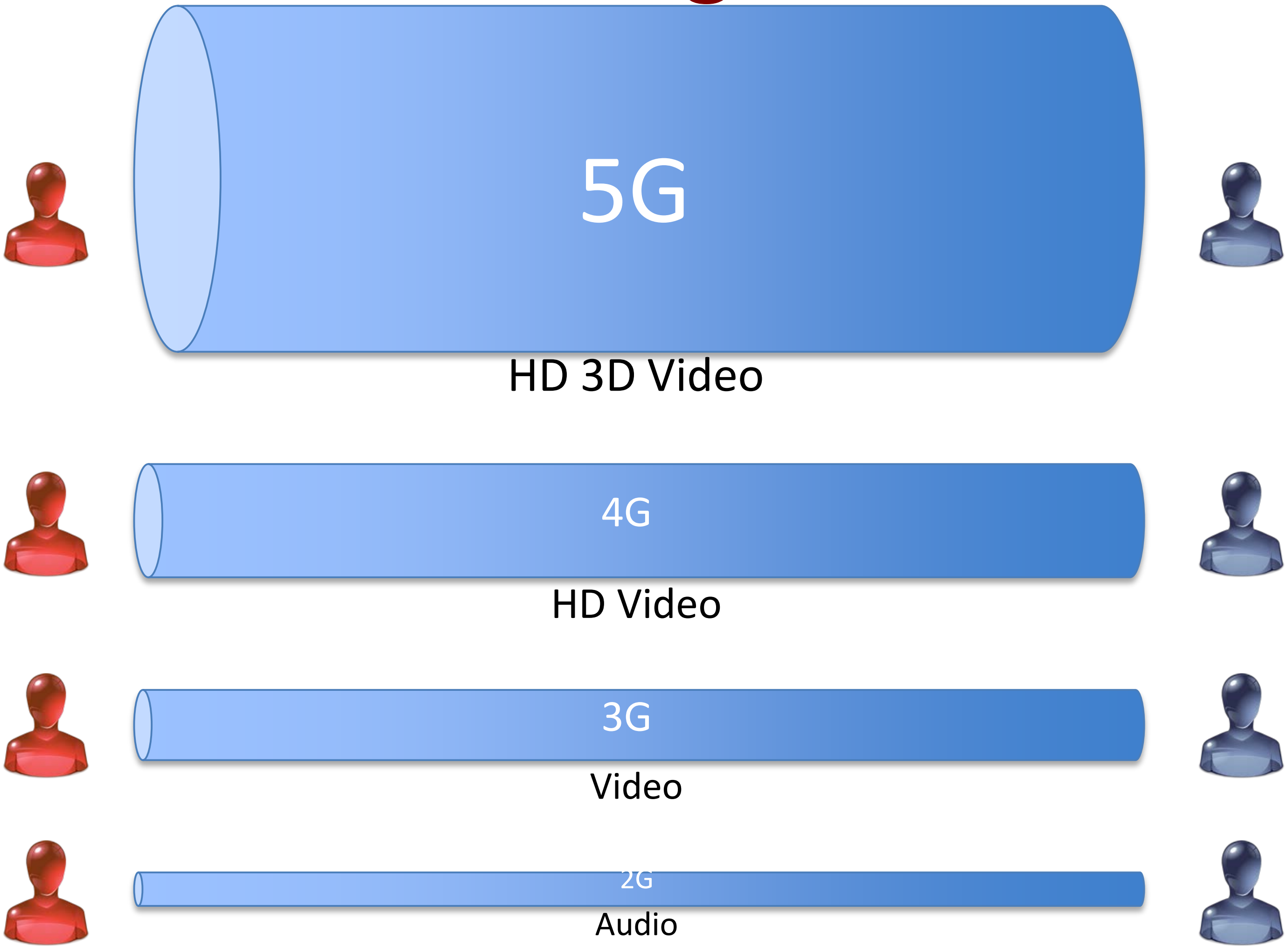
Luis Lopez
lulop@kurento.org

Human communications

The value of technology

DISTANCE,
I HATE YOU.

Multimedia communication technologies



5G

HD 3D Video

4G

HD Video

3G

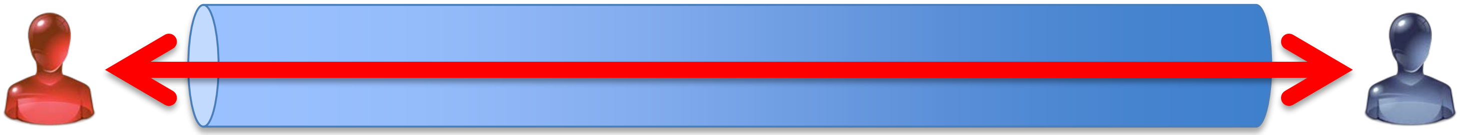
Video

2G

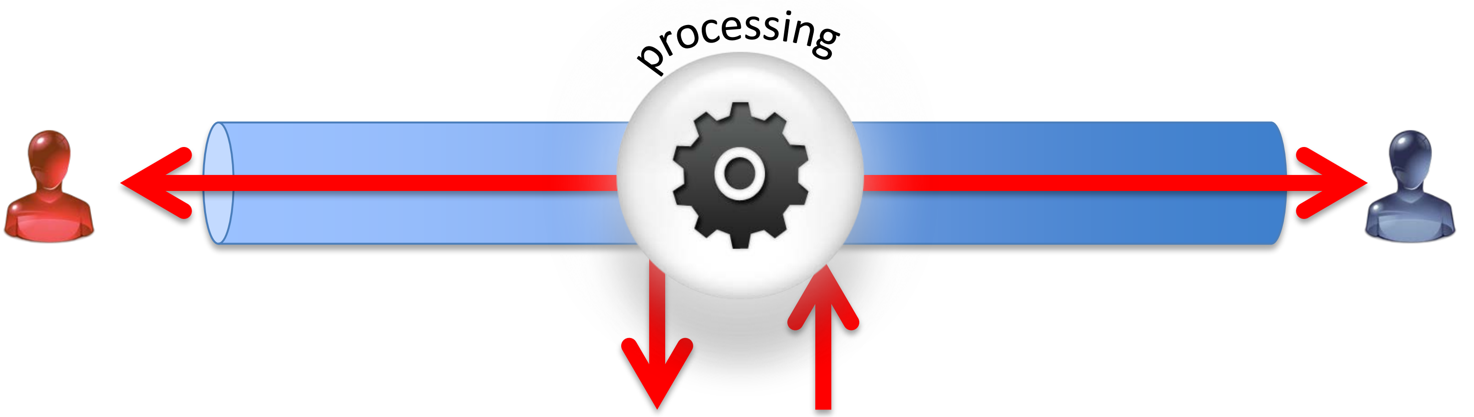
Audio

The Kurento vision

From this …



… to this



Knowledge

Cooking Kurento

WebRTC

OpenCV

gstreamer

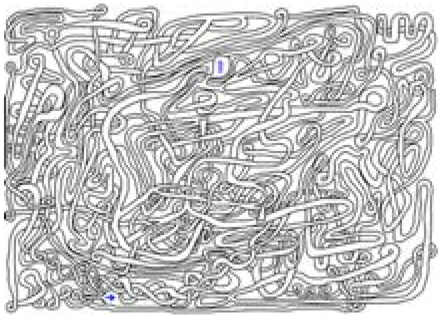
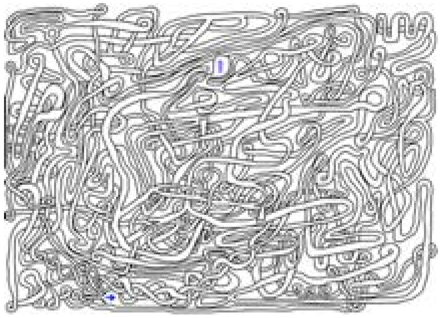
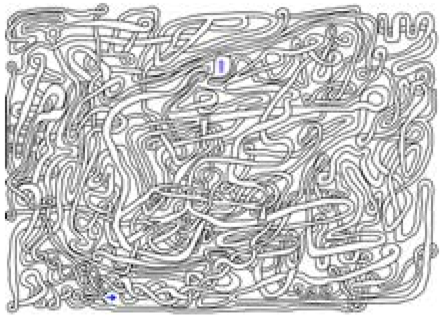
HTML5

node js

Java EE

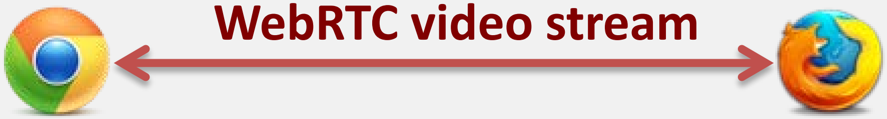
spring

WebRTC: present and future

	Before WebRTC	After WebRTC	Next natural step…
Developing the client side		Begin → End - Unified APIs - Standards - FOSS - Multiplatform	Begin → End - Unified APIs - Standards - FOSS - Multiplatform
Developing the infrastructure side			Begin → End - Unified APIs - Standards - FOSS - Multiplatform

WebRTC infrastructures

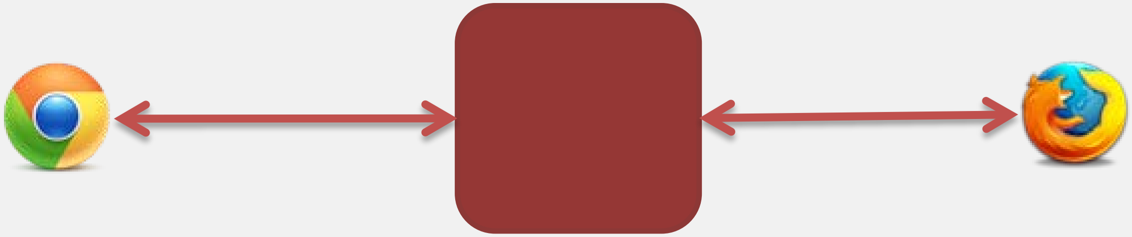
Peer-to-Peer WebRTC Application (without media infrastructure)



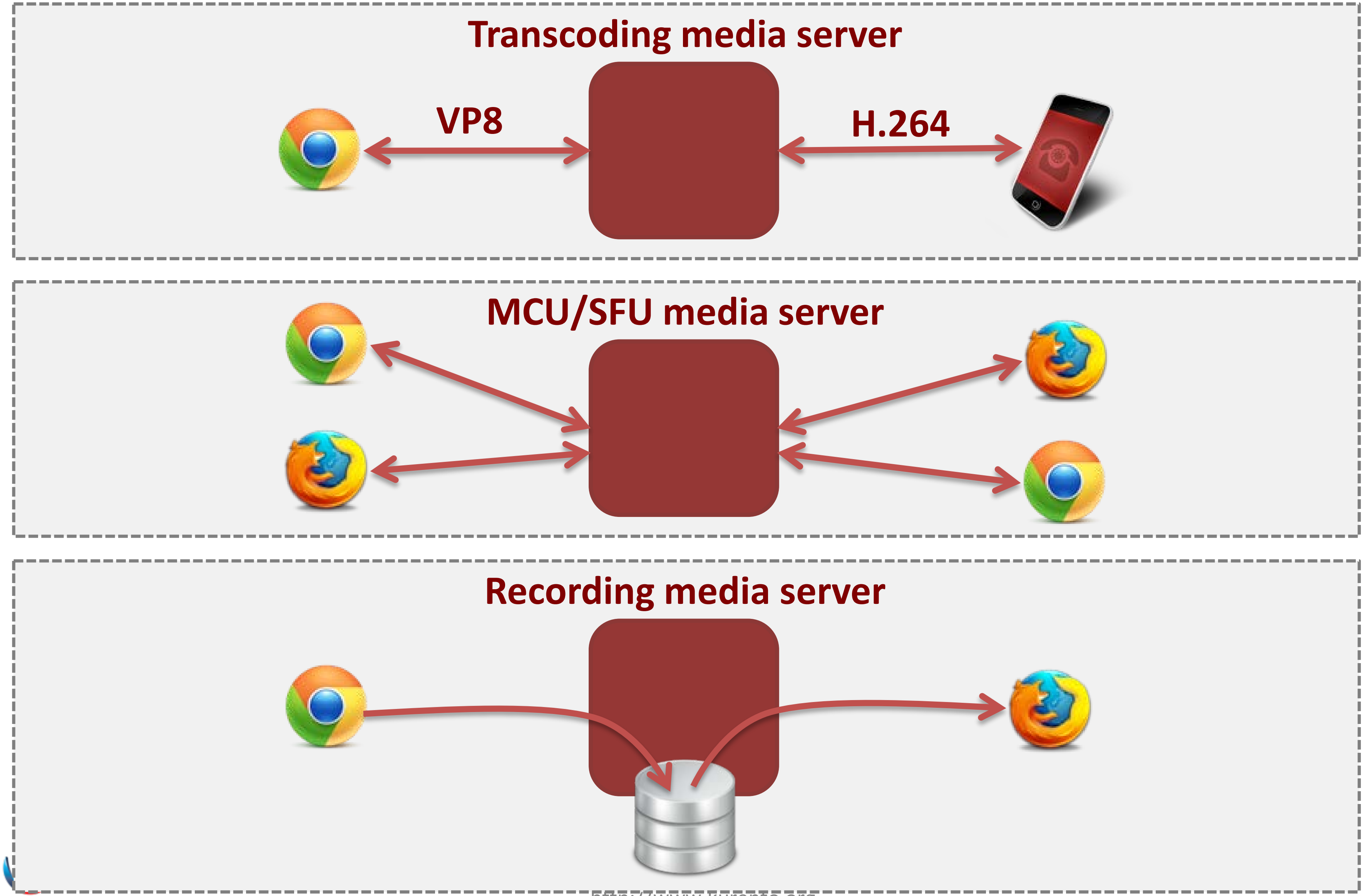
WebRTC video stream

WebRTC Application based on media infrastructure

media infrastructure



Function of WebRTC media servers



Transcoding media server

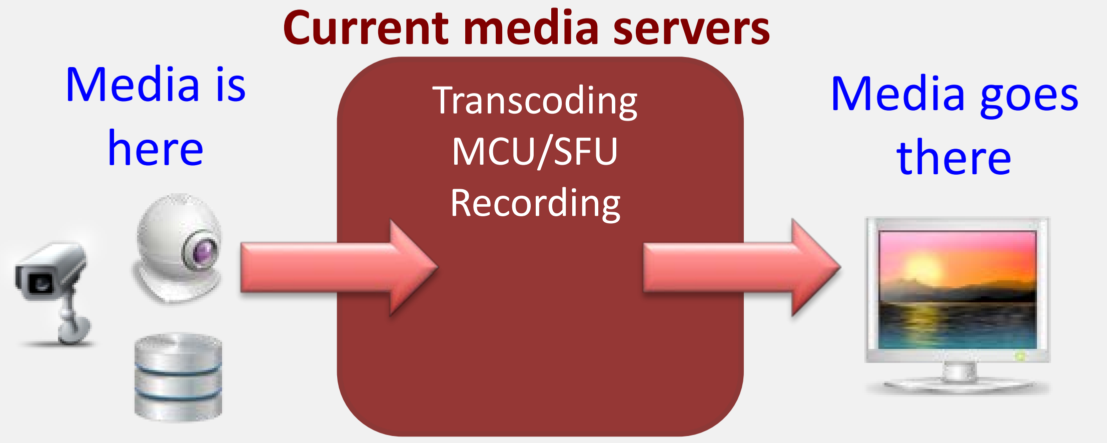
MCU/SFU media server

Recording media server

Kurento as a WebRTC media server

What common WebRTC Media Servers do:

- Transcoding
- MCU
- Recording



Current media servers

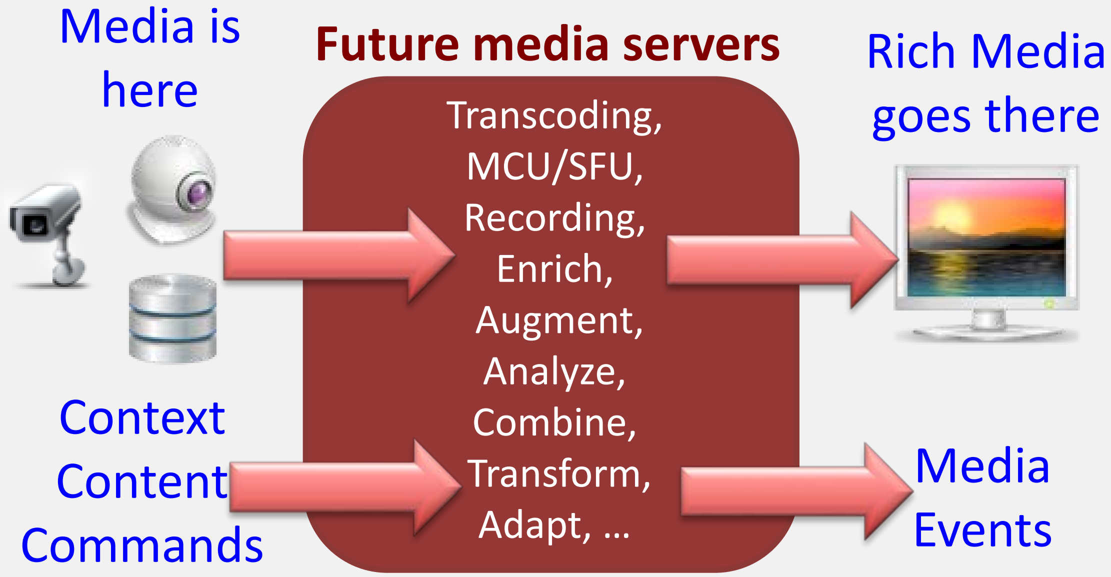
Media is here

Transcoding
MCU/SFU
Recording

Media goes there

What future Media Servers will do:

- Flexible processing
- Augmented reality
- Blending
- Mixing
- Analyzing
- Etc.



Media is here

Future media servers

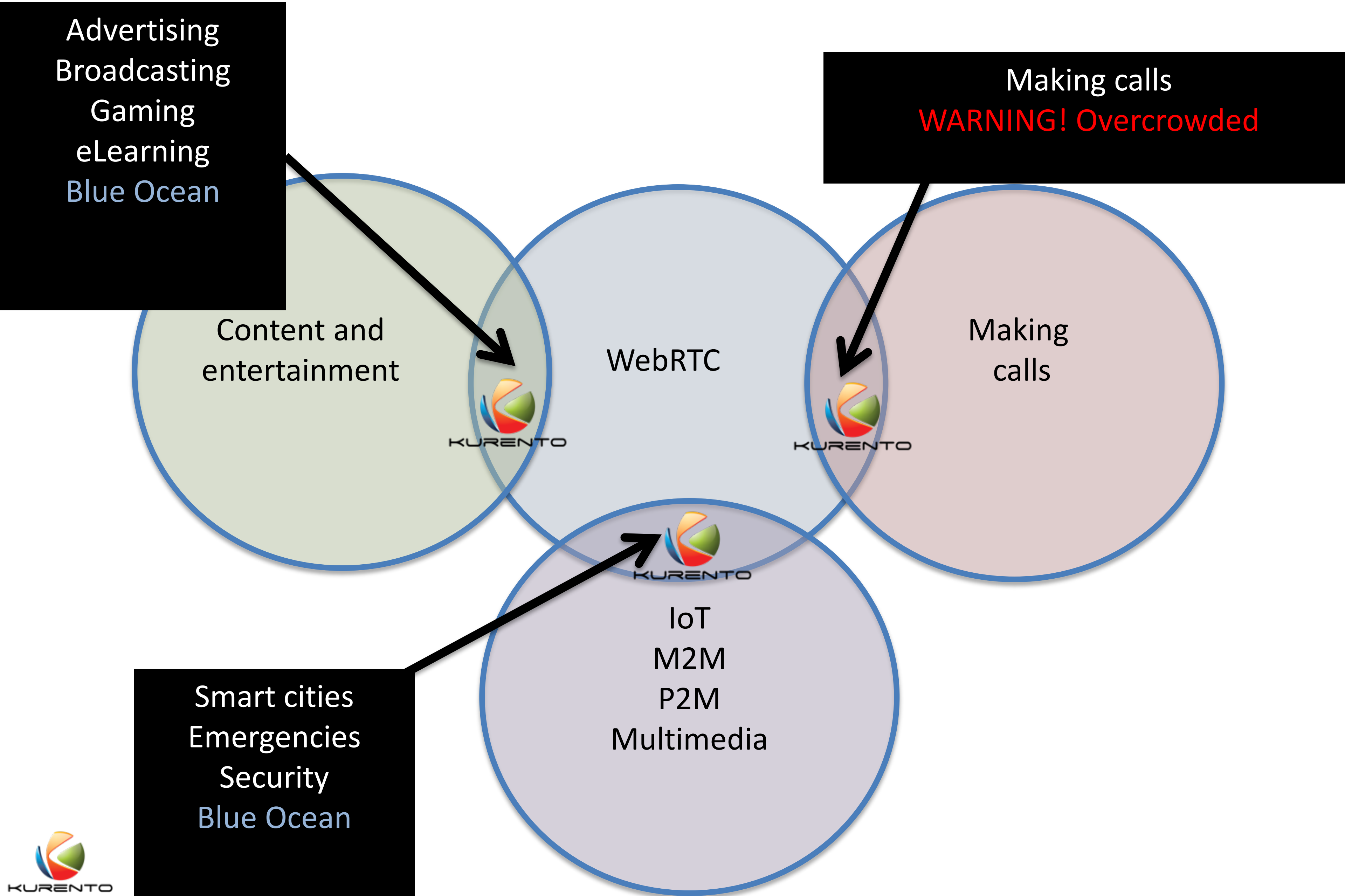
Transcoding,
MCU/SFU,
Recording,
Enrich,
Augment,
Analyze,
Combine,
Transform,
Adapt, …

Rich Media goes there

Context
Content
Commands

Media
Events

Why is this important



Advertising
Broadcasting
Gaming
eLearning
Blue Ocean

Making calls
WARNING! Overcrowded

Content and entertainment

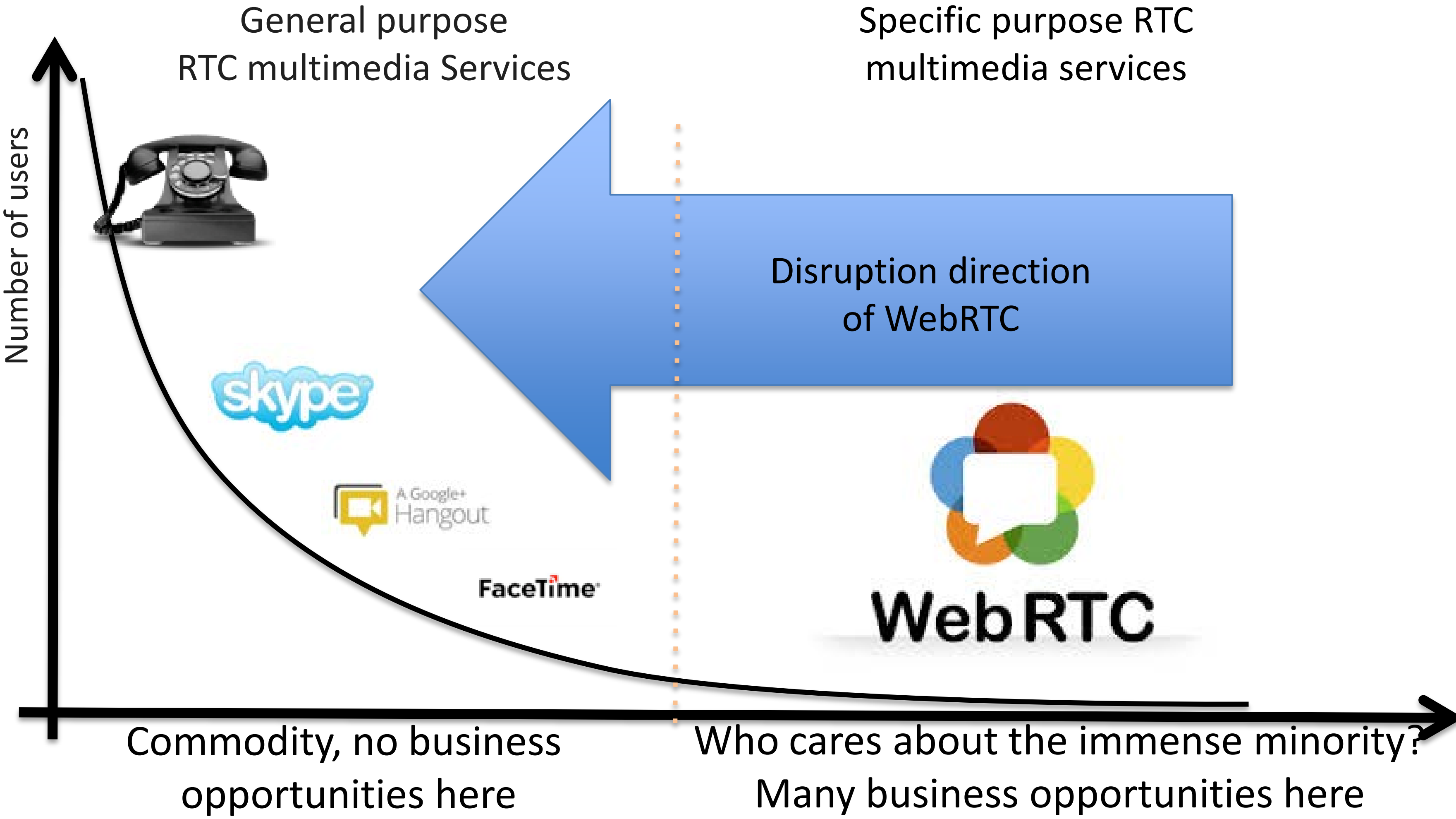
WebRTC

Making calls

IoT
M2M
P2M
Multimedia

Smart cities
Emergencies
Security
Blue Ocean

WebRTC as a disruptive technology



General purpose
RTC multimedia Services

Specific purpose RTC
multimedia services

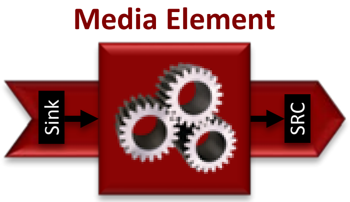
Disruption direction
of WebRTC

Commodity, no business
opportunities here

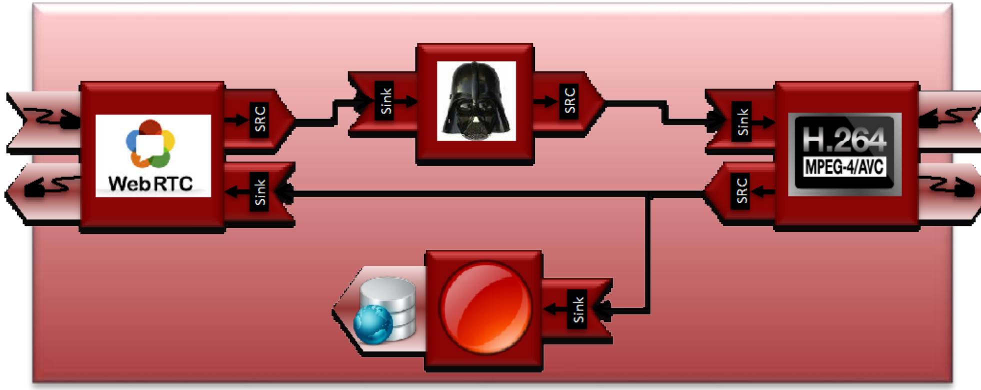
Who cares about the immense minority?
Many business opportunities here

Kurento Media Server

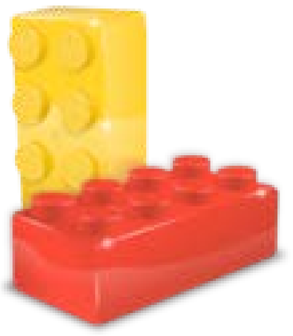
- Media Element
  - Provides a specific media functionality
    - Send/receive media
    - Process media
    - Transform media
  - Exchange media through
    - Sources
    - Sinks

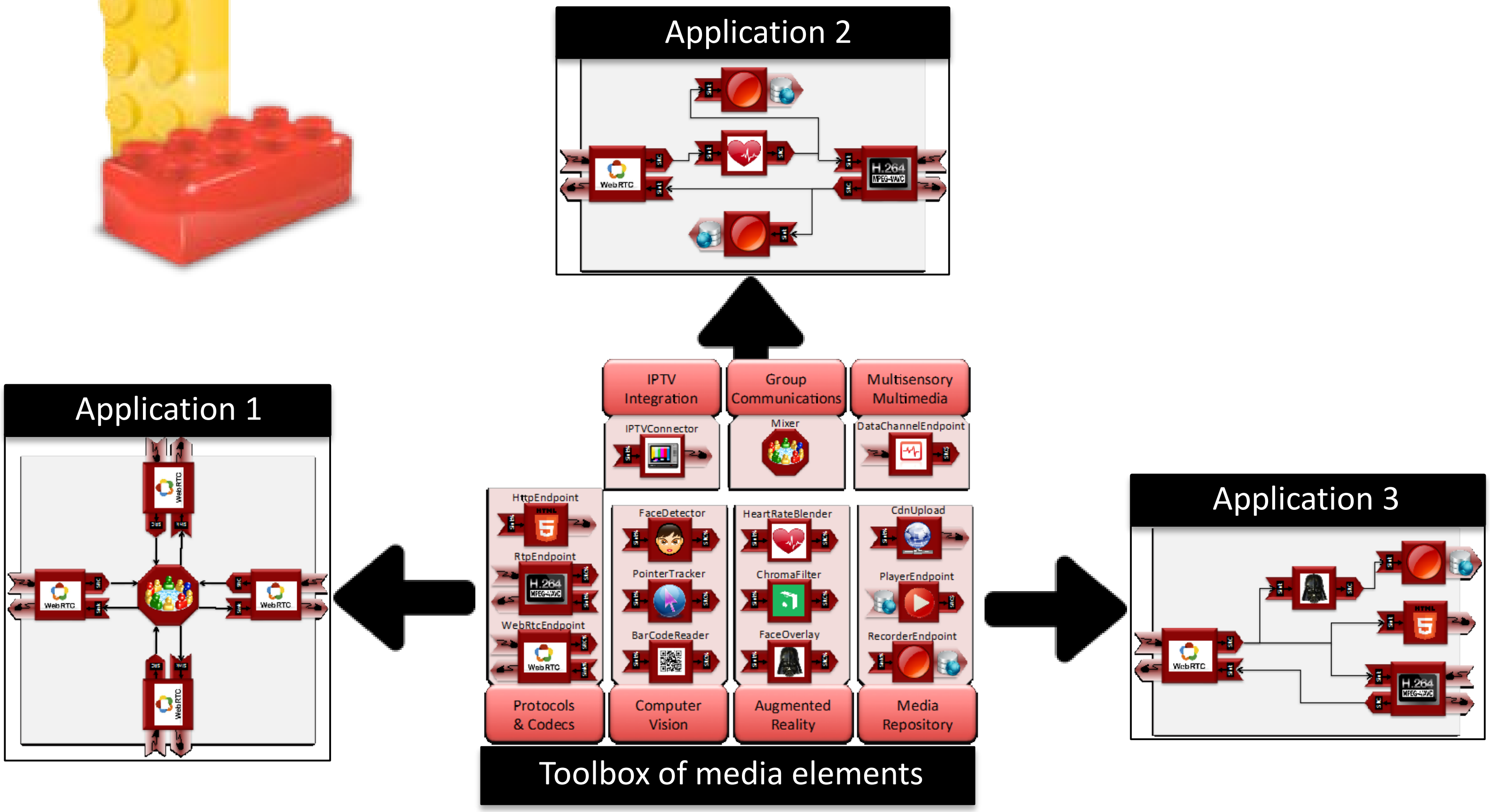


- Media pipeline
  - Chain of media elements implementing the desired media logic.
  - The Media API provides the capability of creating media pipelines by joining media elements of the toolbox



Creating applications basing on Kurento Media Server





Application 2

Application 1

Application 3

IPTV Integration

IPTVConnector

Group Communications

Mixer

Multisensory Multimedia

DataChannelEndpoint

HttpEndpoint

RtpEndpoint

WebRtcEndpoint

Protocols & Codecs

FaceDetector

PointerTracker

BarCodeReader

Computer Vision

HeartRateBlender

ChromaFilter

FaceOverlay

Augmented Reality

CdnUpload

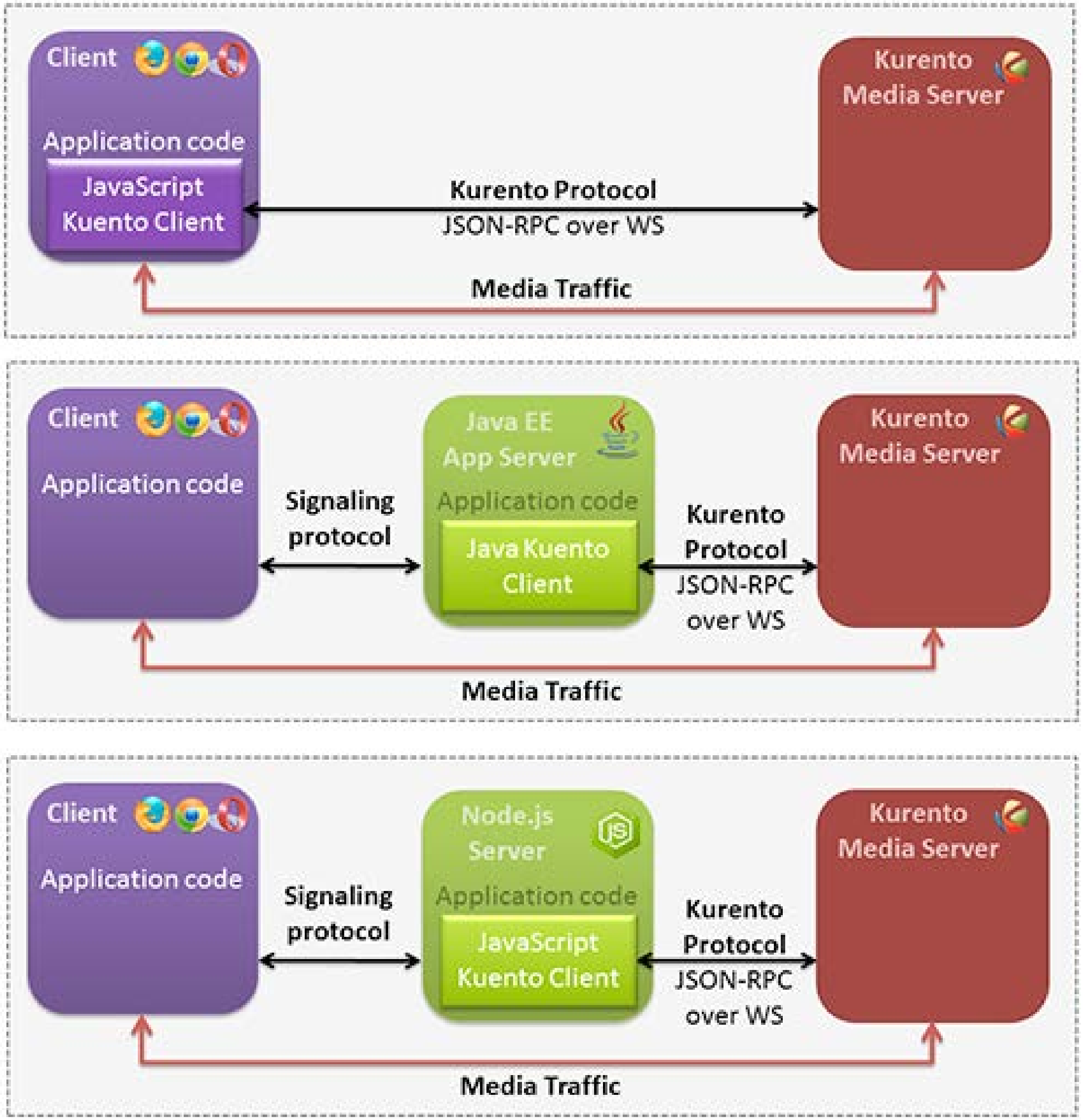
PlayerEndpoint

RecorderEndpoint

Media Repository

Toolbox of media elements

Creating applications with Kurento



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

it seems dangerous...
LETS DO IT!!

What you should learn first

- WebRTC basics
  - http://www.html5rocks.com/en/tutorials/webrtc/basics/
- Signaling basics (STUN/TURN)
  - http://www.html5rocks.com/en/tutorials/webrtc/infrastructure/

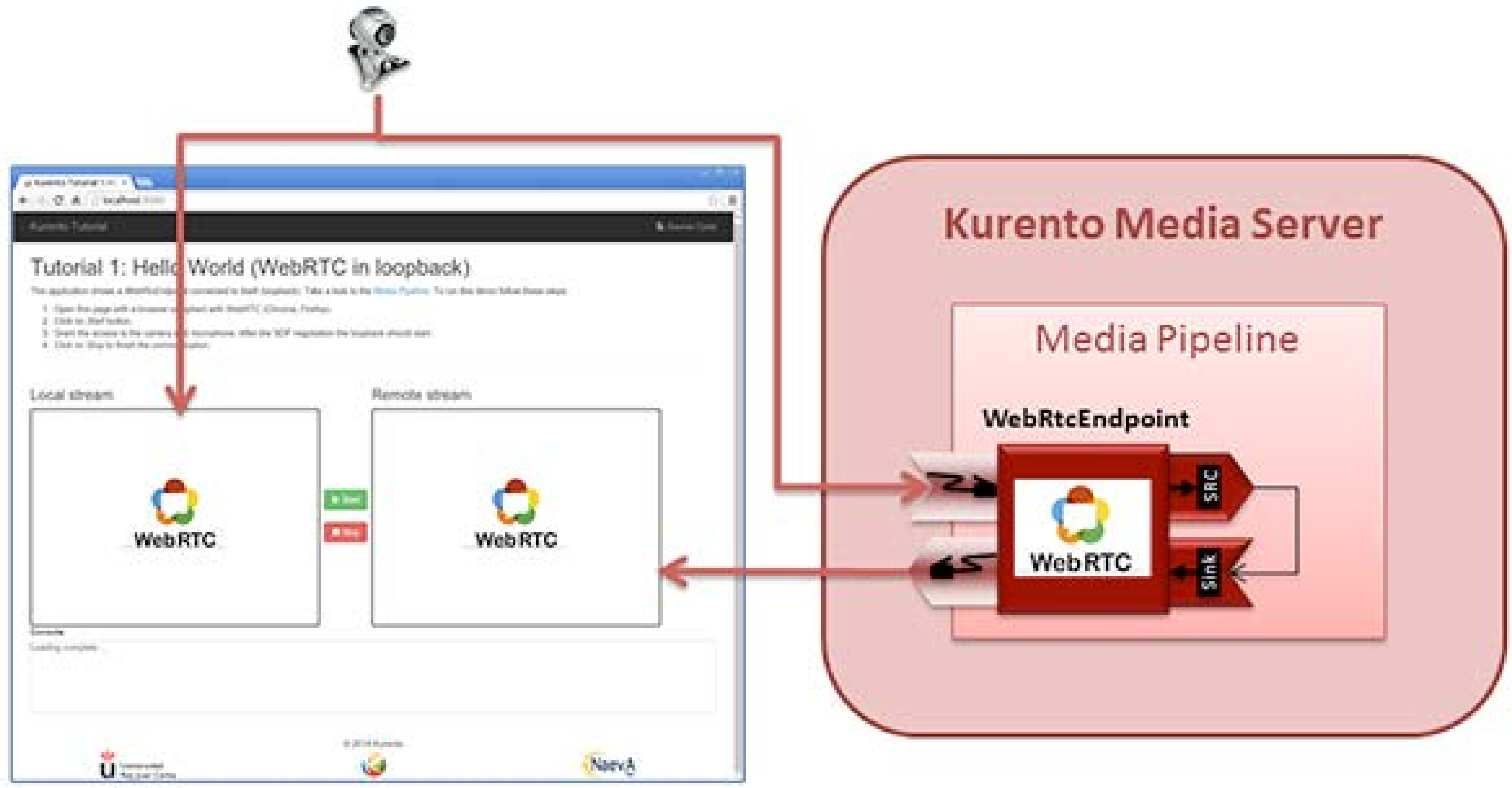
Starting with Kurento

- 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`

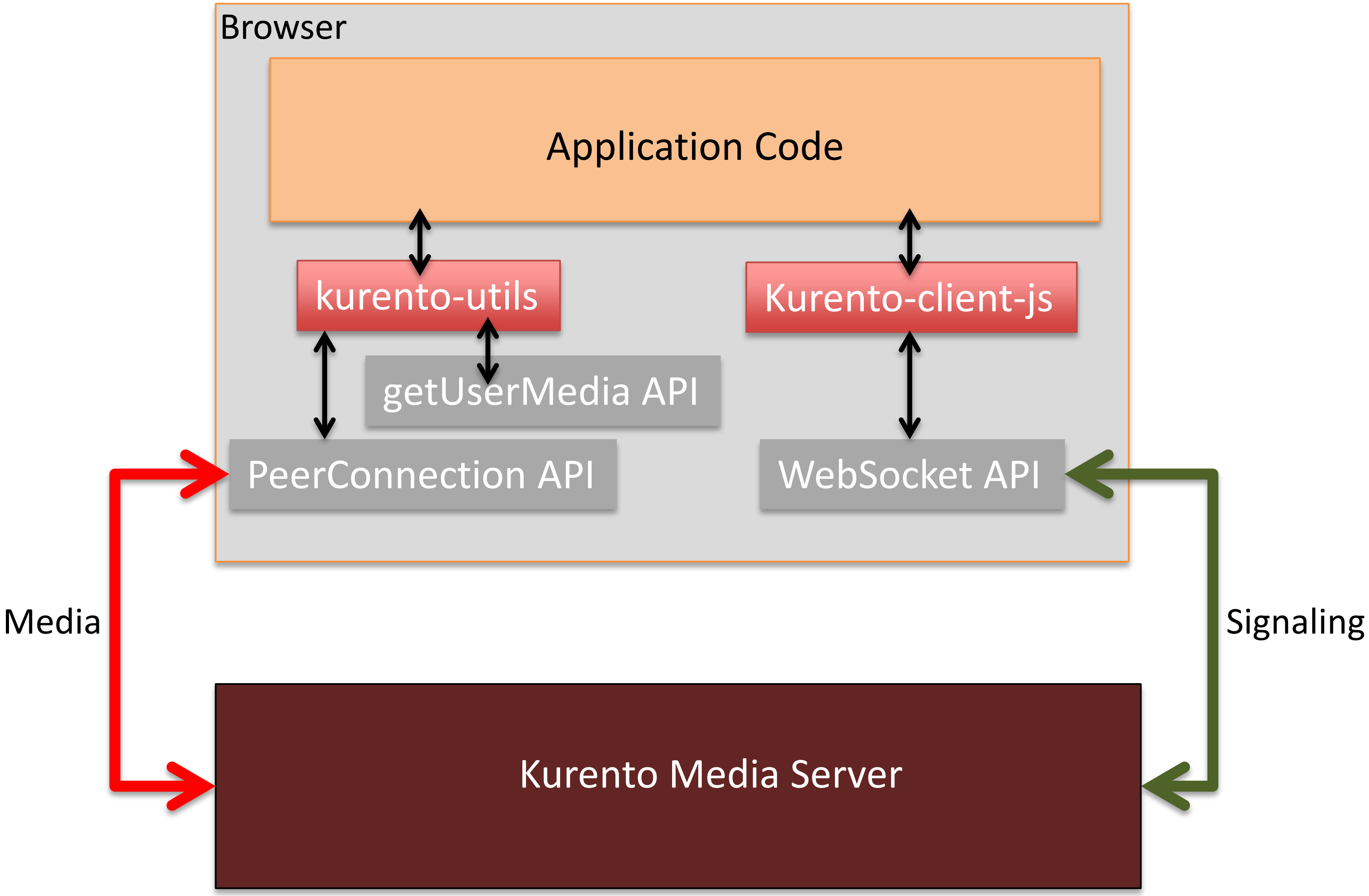
Kurento “Hello World!”



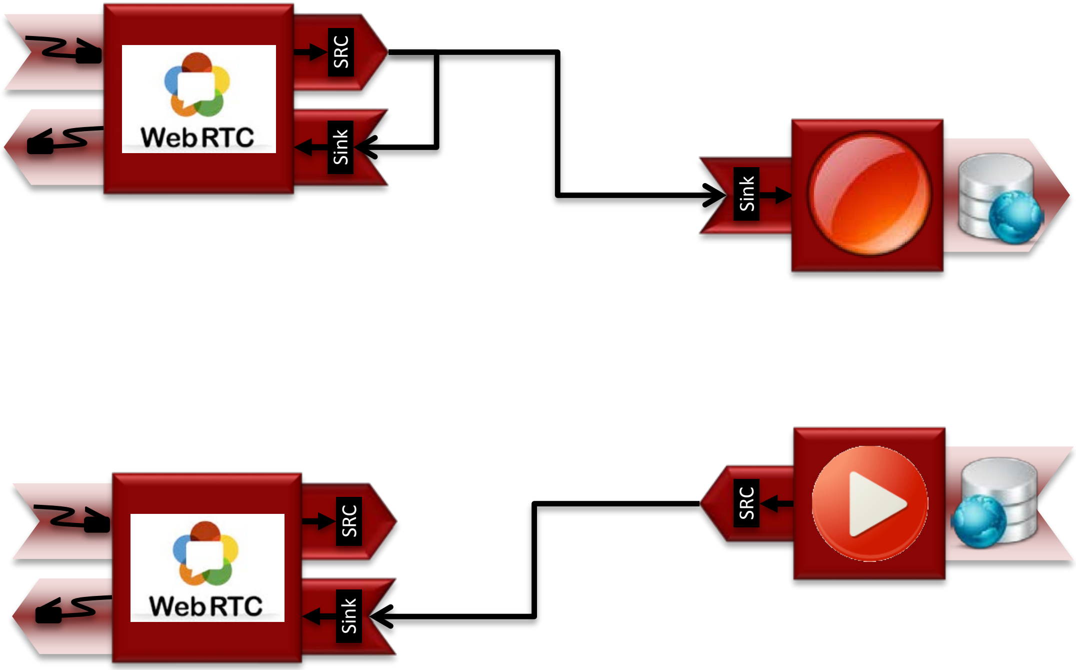
Kurento "Hello World!"

- Tutorial
  - http://www.kurento.org/docs/current/tutorials/js/tut
    rial-1-helloworld.html
- Code
  - https://github.com/Kurento/kurento-tutorial-js/tree/release-5.1/kurento-hello-world
- Video
  - https://www.youtube.com/watch?v=vGEnkSOp_xc

Understanding this example



Recording and playing



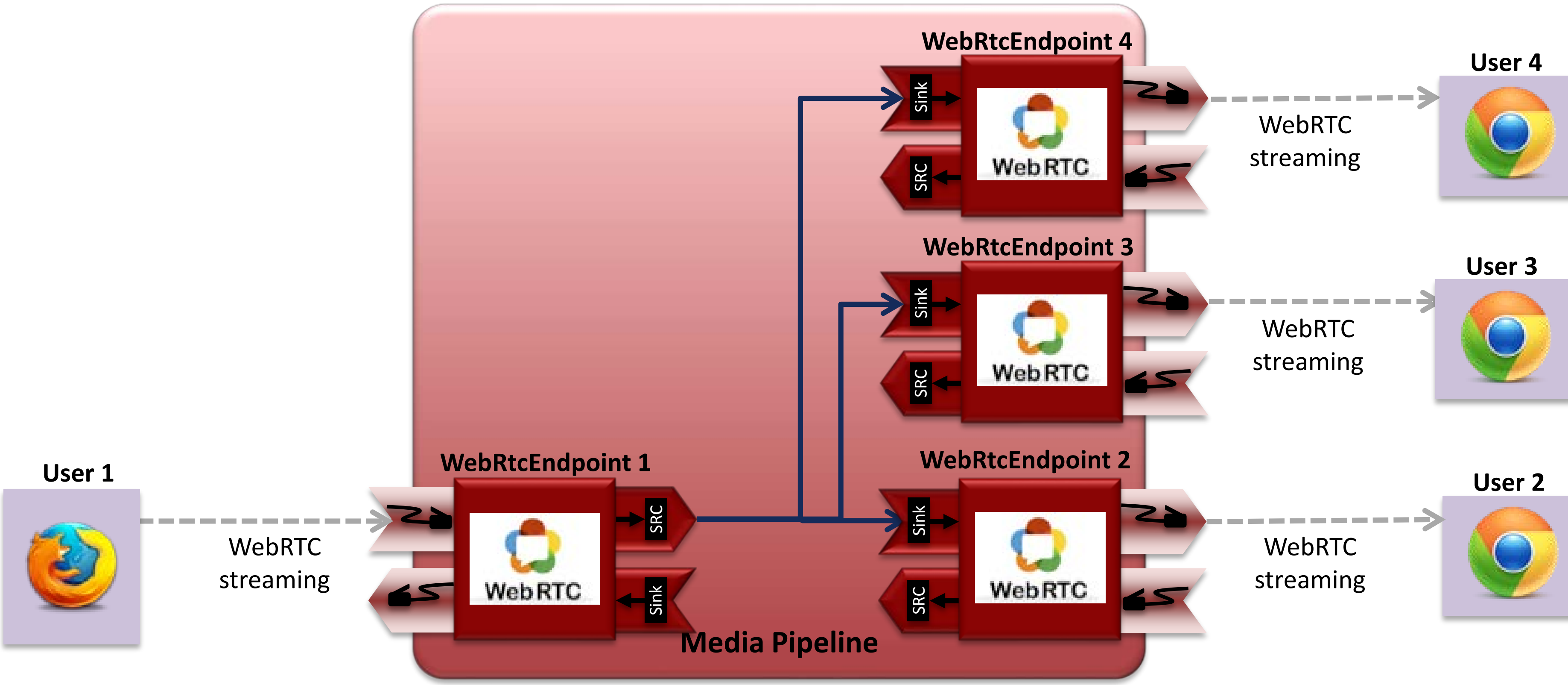
Recording and playing

- Get code here
  - https://github.com/Kurento/kurento-tutorial-js/tree/release-5.1/kurento-hello-world-recorder-generator
  - 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

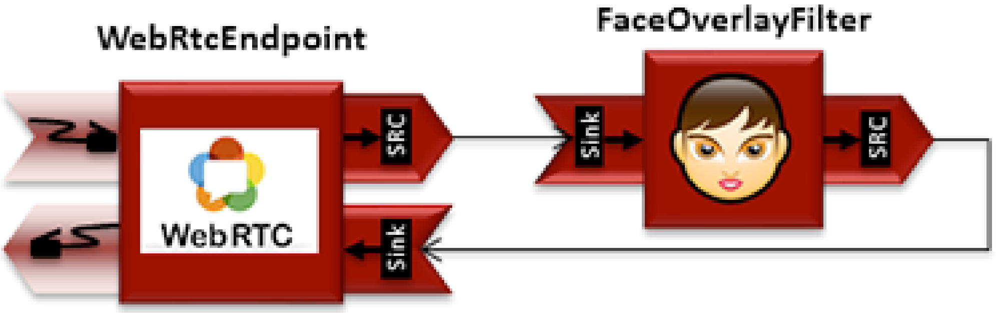
One-to-many example



One-to-many

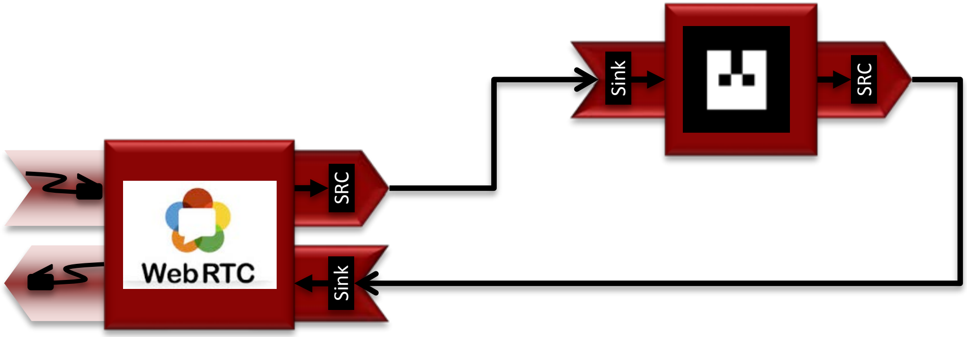
- Java EE
  - http://www.kurento.org/docs/current/tutorials/java/t
    utorial-3-one2many.html
- Node.js
  - http://www.kurento.org/docs/current/tutorials/node/
    utorial-3-one2many.html

Face overlay example



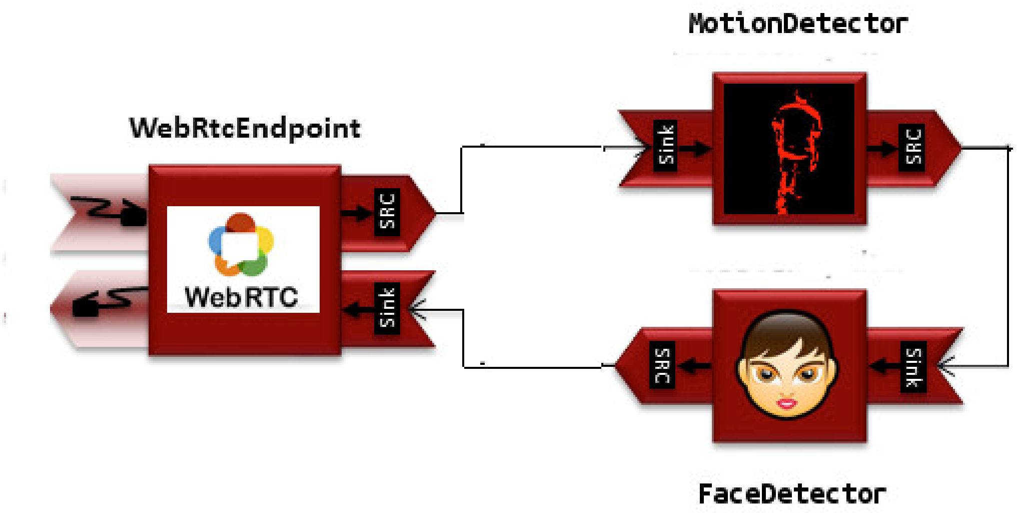
- Browser JavaScript
  - http://www.kurento.org/docs/current/tutorials/js/tutorial-2-magicmirror.html
- Java
  - http://www.kurento.org/docs/current/tutorials/java/tutorial-2-magicmirror.html
- Node.js
  - http://www.kurento.org/docs/current/tutorials/node/tutorial-3-one2many.html
- Video
  - https://www.youtube.com/watch?v=h84HFkvWGgw

Augmented Reality example



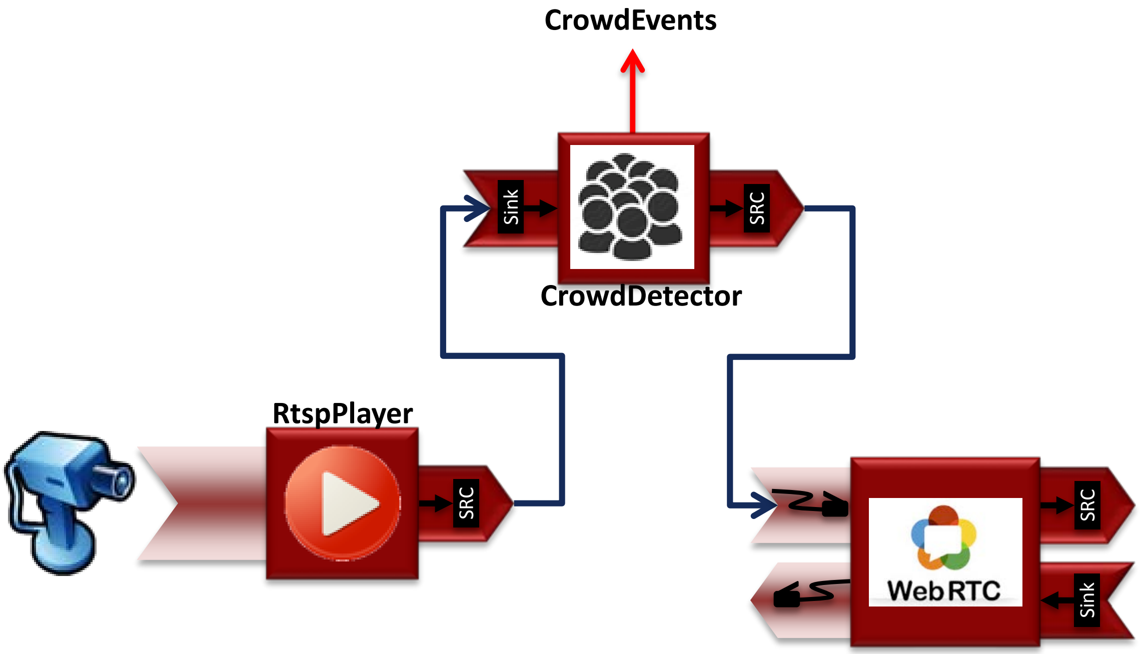
- Video
  - https://www.youtube.com/watch?v=JlRg4PzeRKQ

Motion detector



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

Crowd detector

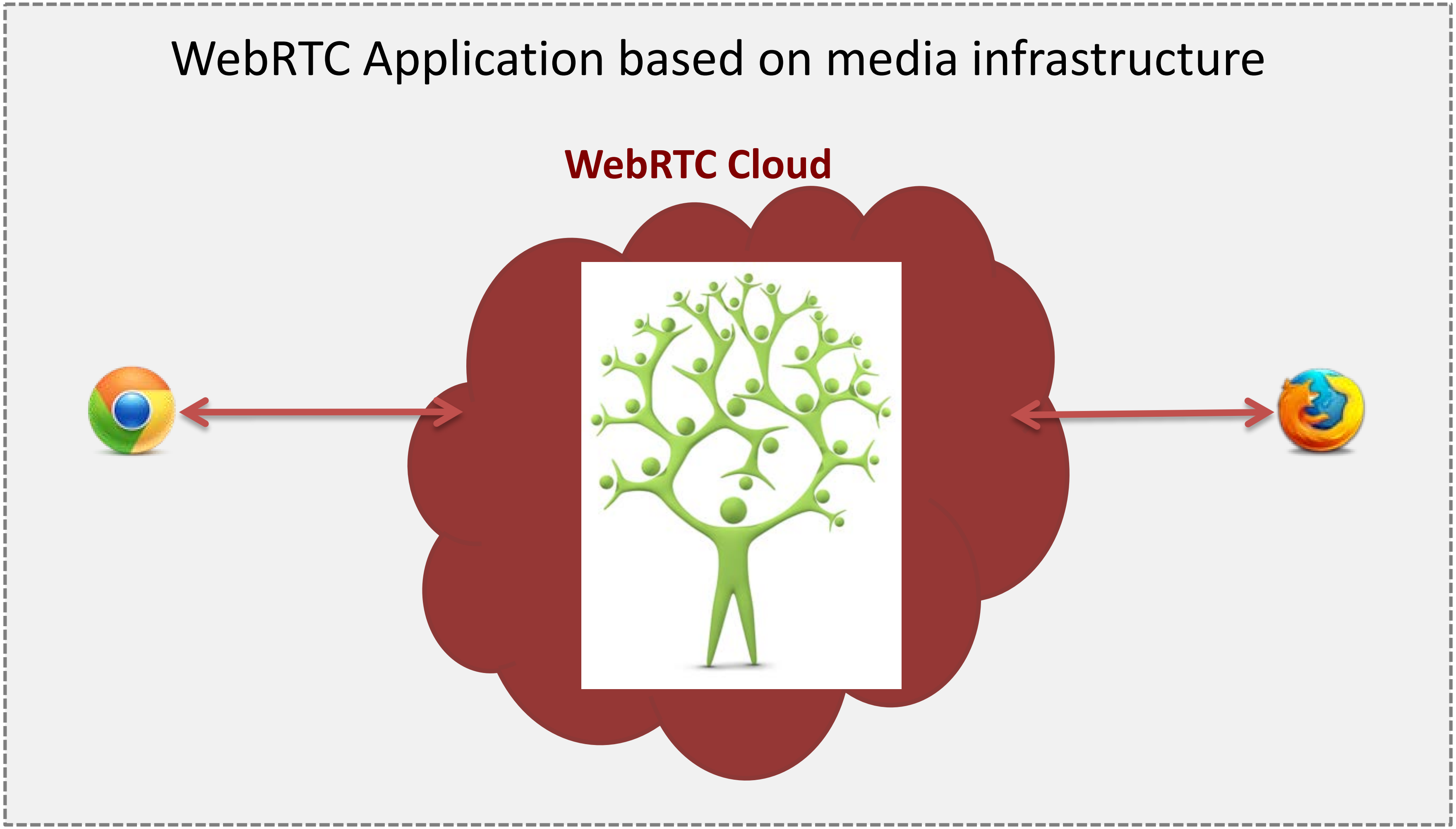


- Video
  - https://www.youtube.com/watch?v=S6iWSCysgT0

Many other examples

- Face segmentator (aka get a Kiss)
  - https://www.youtube.com/watch?v=WRmzzblZGDo
- Room communications
  - https://www.youtube.com/watch?v=hkT8fLROdwo
- B2B calls
  - https://www.youtube.com/watch?v=ocJBDo8K6eM
- Etc.

Beyond media servers: WebRTC clouds and the problem of scalability



WebRTC Application based on media infrastructure

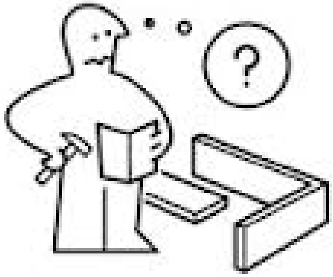
WebRTC Cloud

Cloud models for WebRTC infrastructures

IaaS

Provider
- Computing resources

Developer
- Installation
- Administration
- Security
- Application logic



PaaS

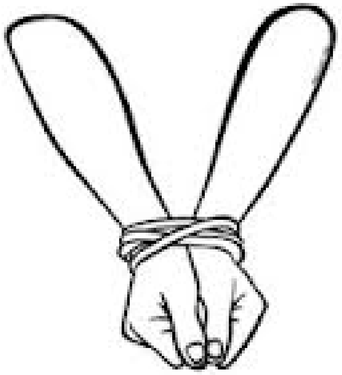
Provider
- Development API

Developer
- Application logic

SaaS

Provider
- Service

Developer
- Nothing to do

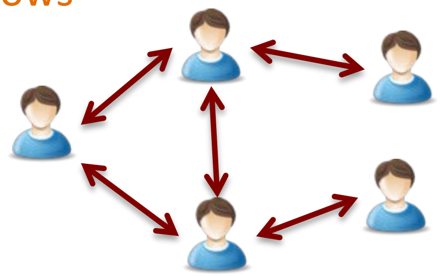


WebRTC PaaS APIs: Requirements

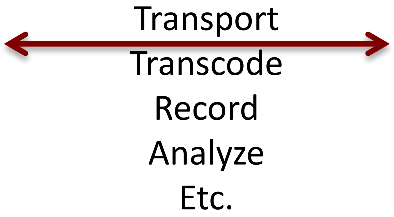
- Requirements of WebRTC PaaS APIs
  - Functional requirements
    - Media transport
      - Media endpoint
      - Media replication
      - Media routing
    - Media persistence
      - Media storage
      - Media recovery
    - Media processing
      - Transcoding
      - Analysis
      - Augmentation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

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

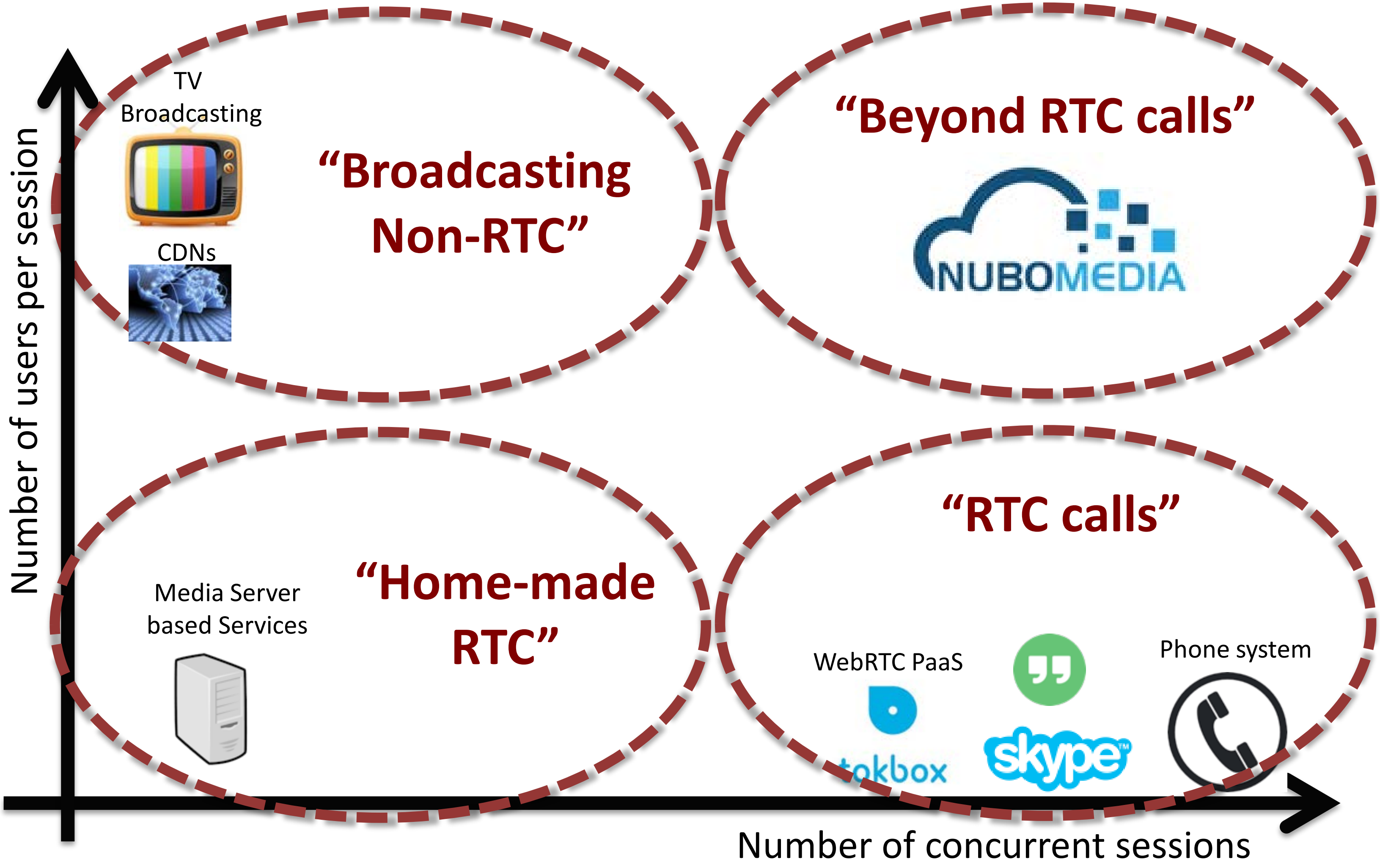


  - Multimedia processing
    - Function of each edge of the graph of media flows

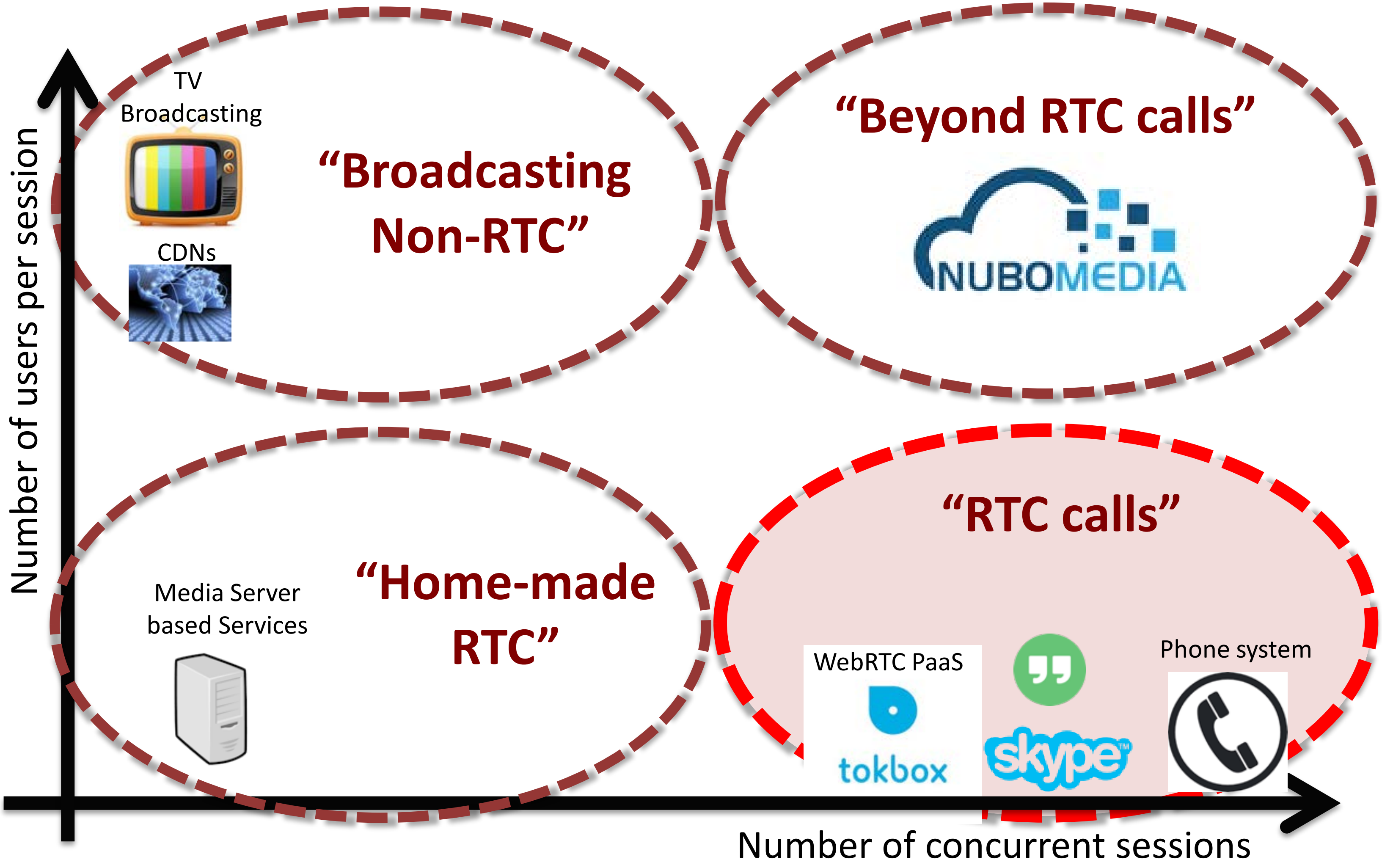


Transport
Transcode
Record
Analyze
Etc.

Scalability of RTC multimedia services



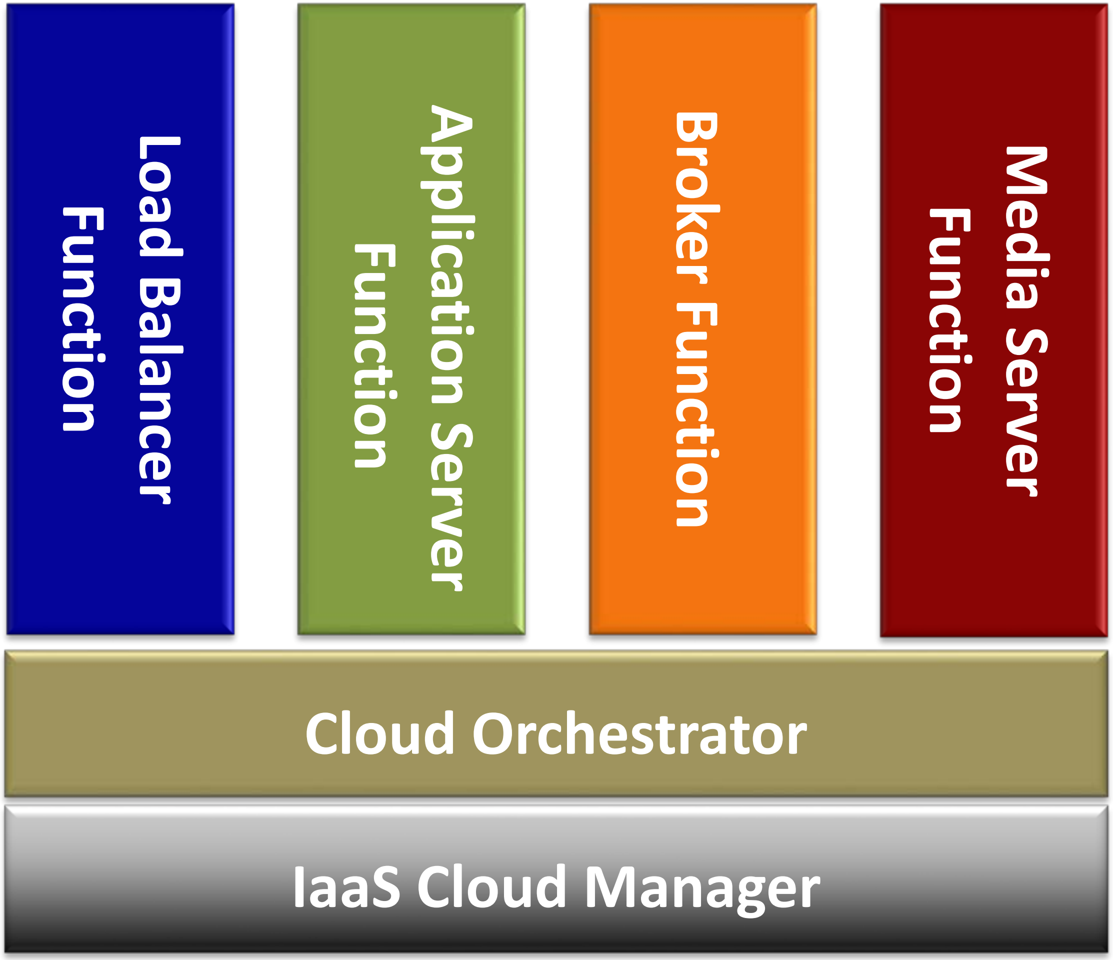
Scalability of RTC multimedia services



The scalability problem in “call” clouds

Call
Call
Call
Call
Call
Call
Call
Call
Call
Call
Call
Call
Call
Call
Call
Call

Anatomy of WebRTC PaaS for call models: Flat Architecture



Load Balancer Function

Application Server Function

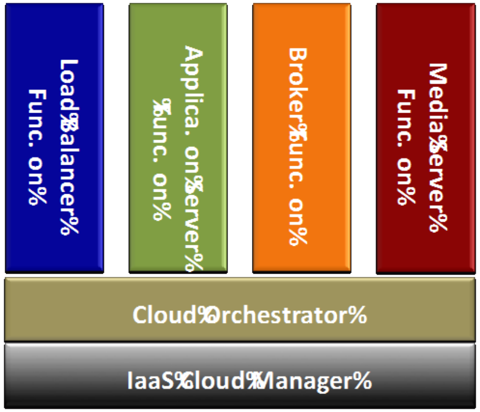
Broker Function

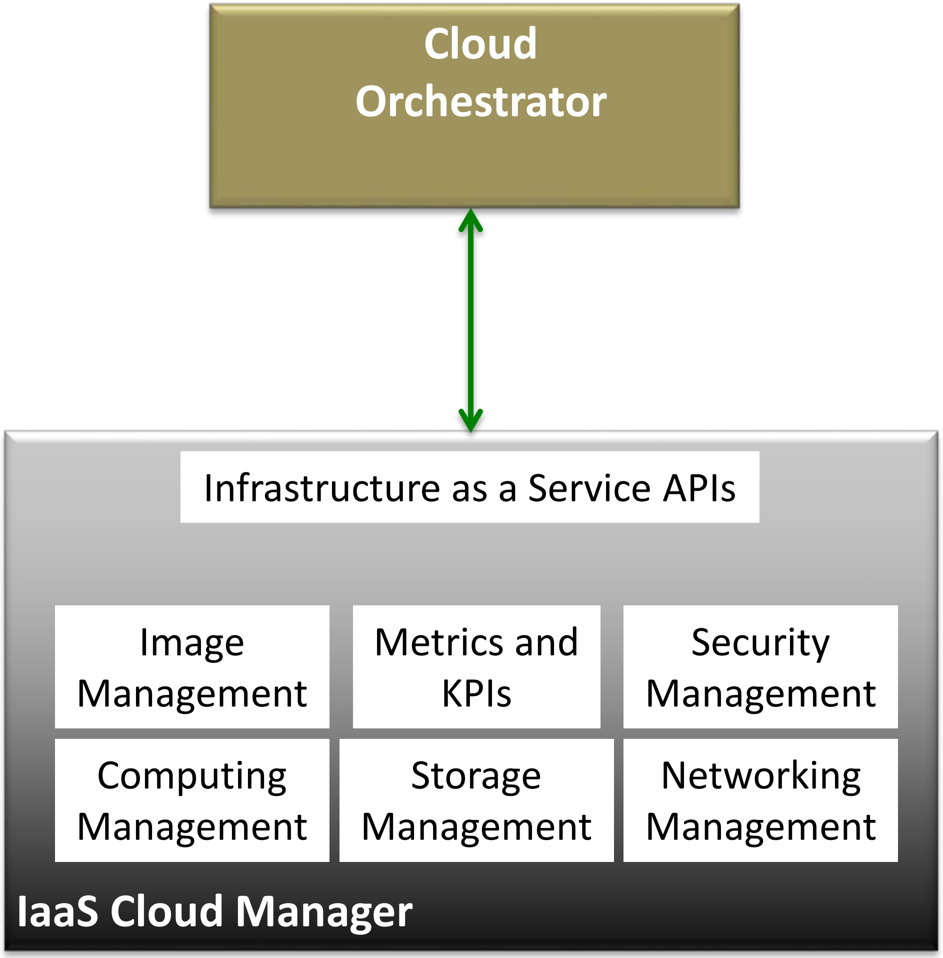
Media Server Function

Cloud Orchestrator

IaaS Cloud Manager

Cloud functions: IaaS manager




Cloud Orchestrator

Infrastructure as a Service APIs

Image Management

Metrics and KPIs

Security Management

Computing Management

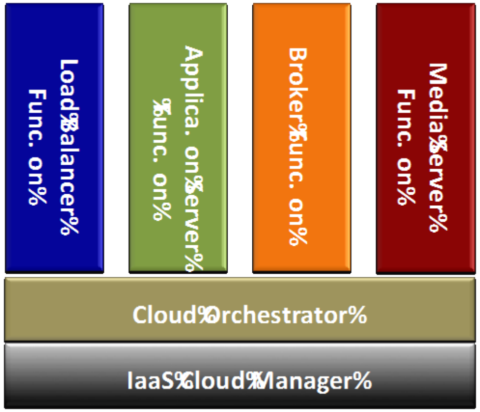
Storage Management

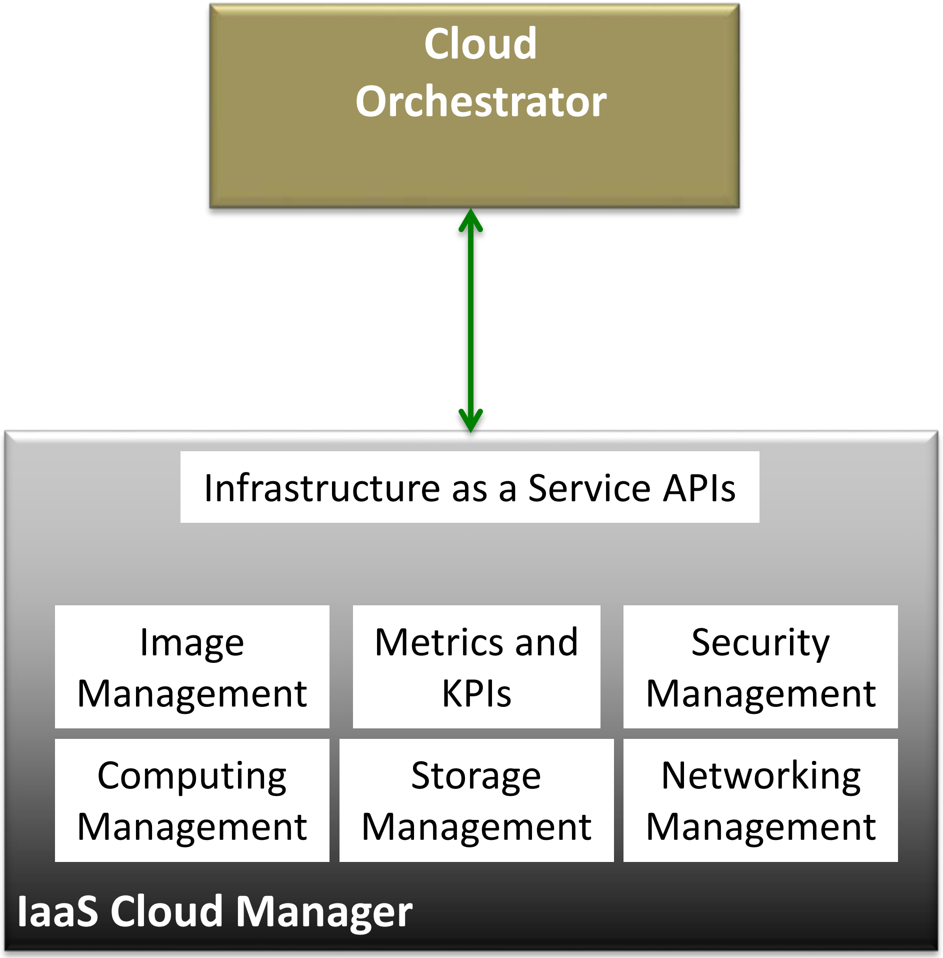
Networking Management

IaaS Cloud Manager

- Function
  - Provides APIs for IaaS management
    - Images
    - Instances
    - Storage
    - Metrics
    - Security
    - Etc.
- Requires
  - Physical infrastructure

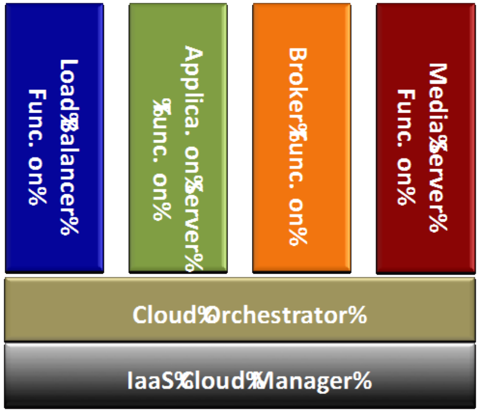
Cloud functions: Cloud Orchestrator

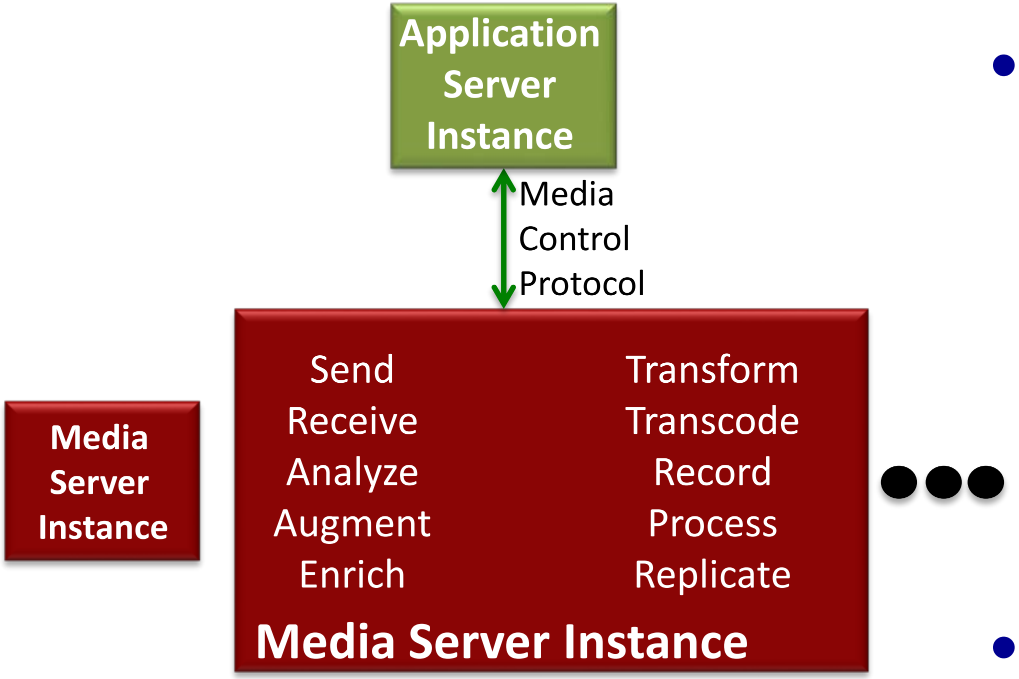




- Function
  - Lifecycle management of the platform
    - It acquires virtual resources and allocate them to the specific services
  - Runtime management with autoscaling
    - It scales out new service instances in situations of peak load
    - It scales in service instances whenever they are not required any longer
- Requires
  - Autoscaling rules
    - Ex. If average load is over 60% add two new instances

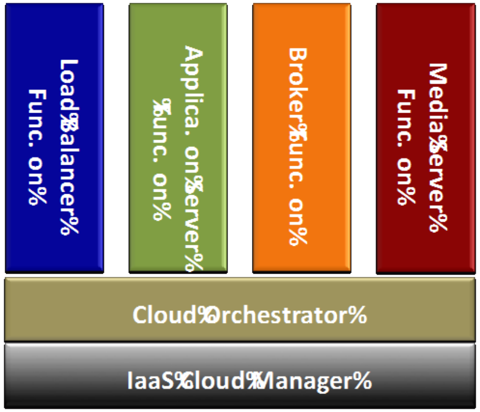
Media Server Function

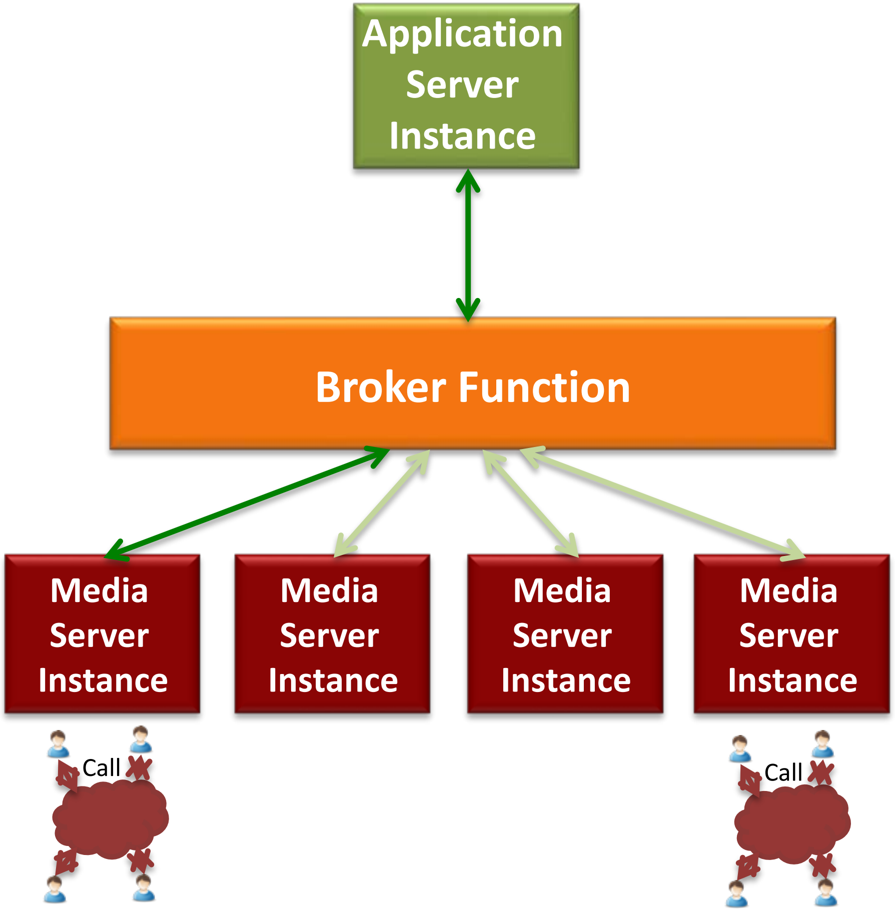




- Function
  - Provides media capabilities
    - WebRTC transport
    - Recording
    - Transcoding
    - Etc.
- Requires
  - Control Protocol
  - Media Protocols
  - Media Codecs

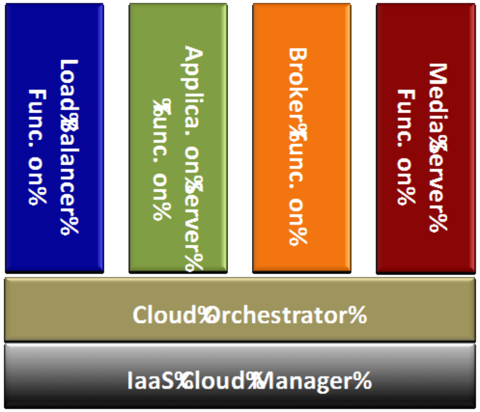
Broker Function





- 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

Application Server



Load%Balancer%Function%

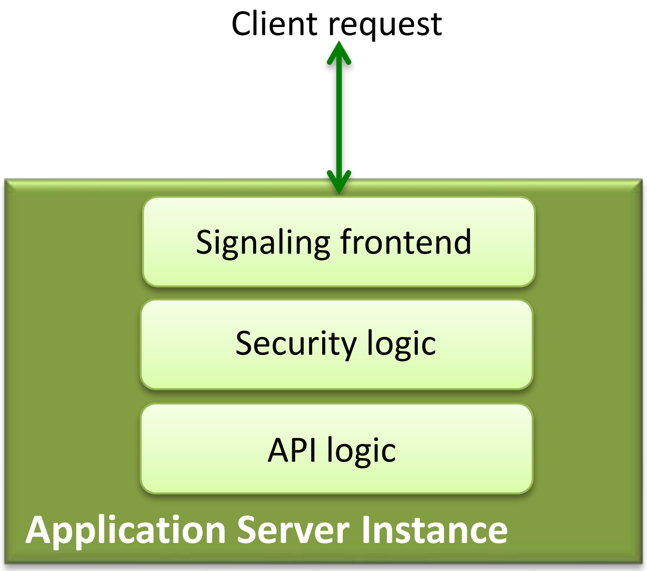
Application%Server%Function%

Broker%Function%

Media%Server%Function%

Cloud%Orchestrator%

IaaS%Cloud%Manager%



Client request

Signaling frontend

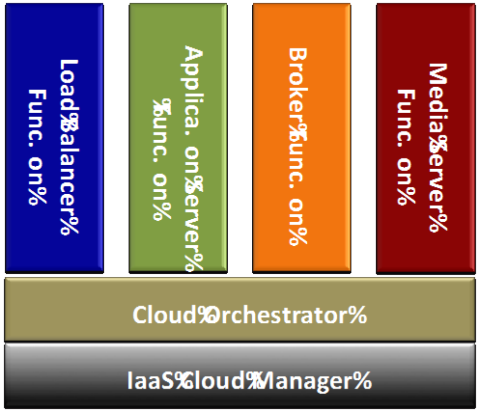
Security logic

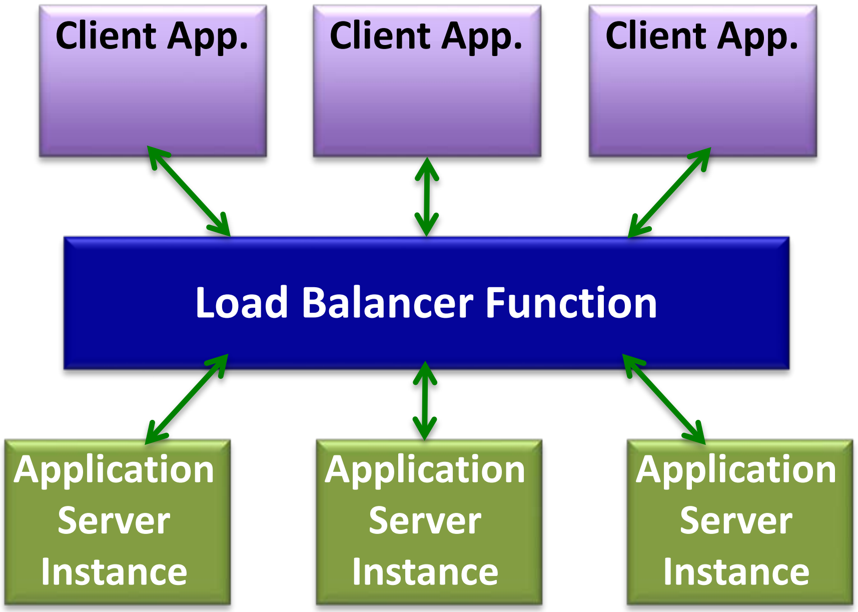
API logic

Application Server Instance

- Function
  - 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

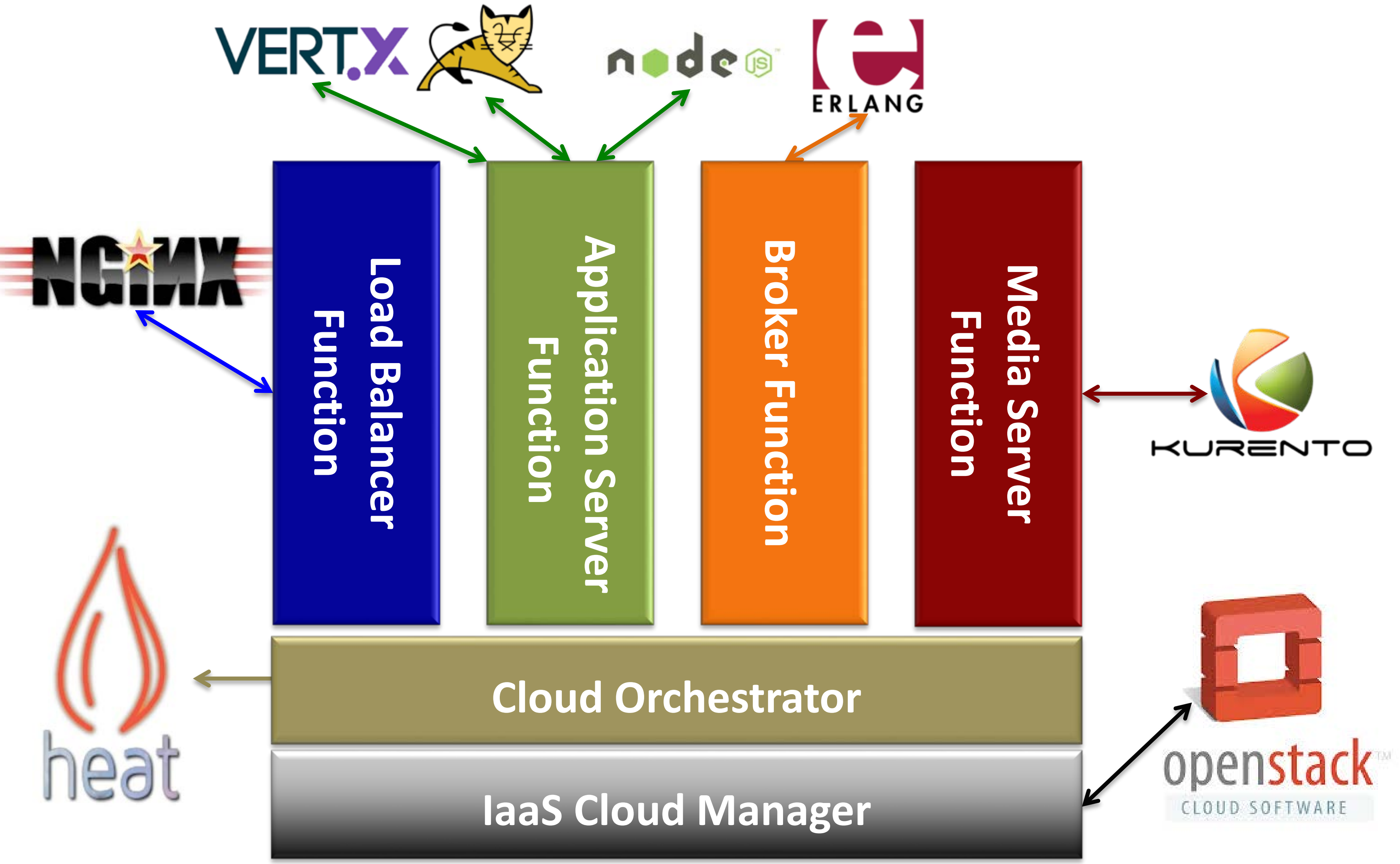
Load balancer




Client App.
Client App.
Client App.
Load Balancer Function
Application Server Instance
Application Server Instance
Application Server Instance

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

The flat Nubomedia implementation



Load Balancer Function

Application Server Function

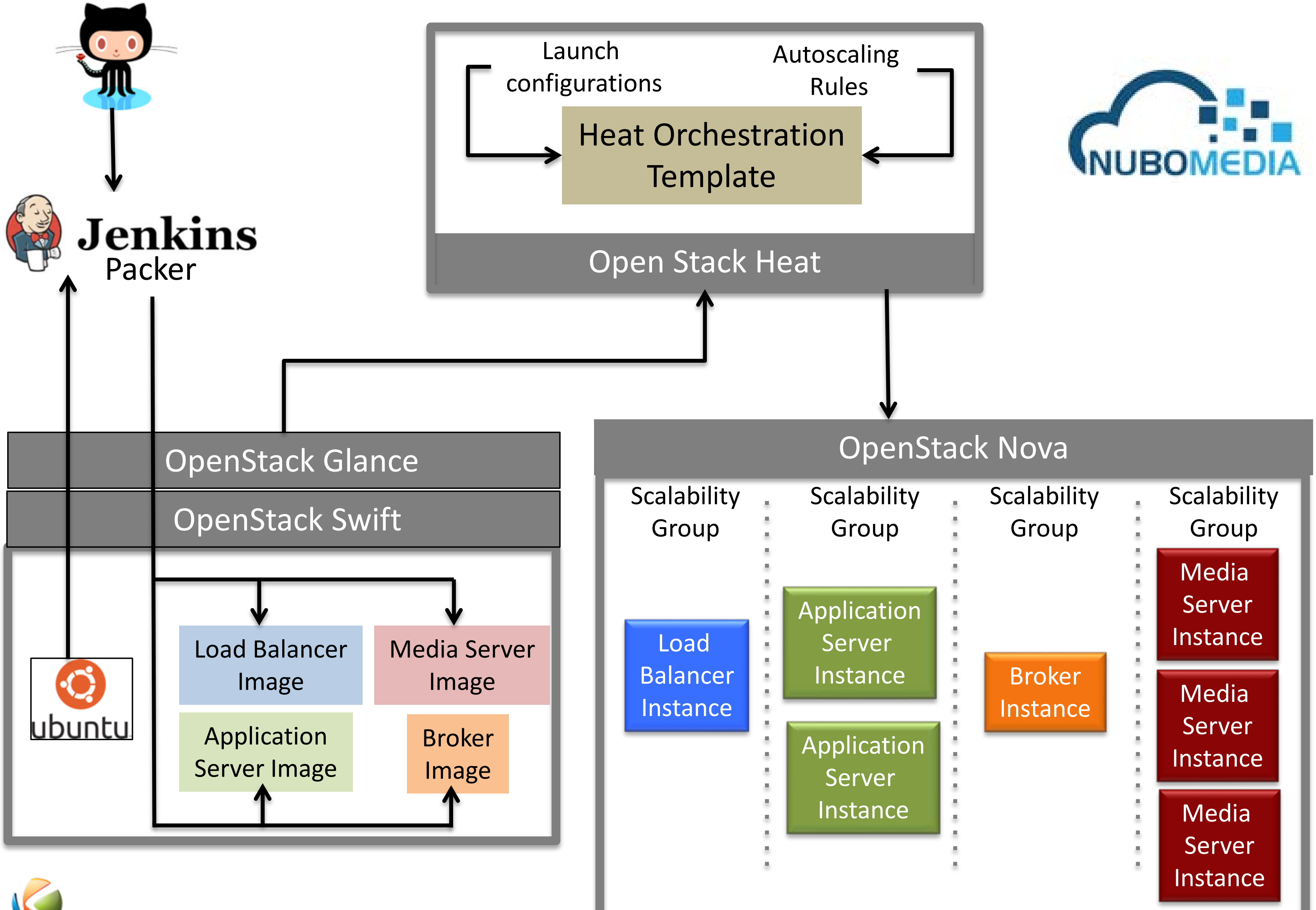
Broker Function

Media Server Function

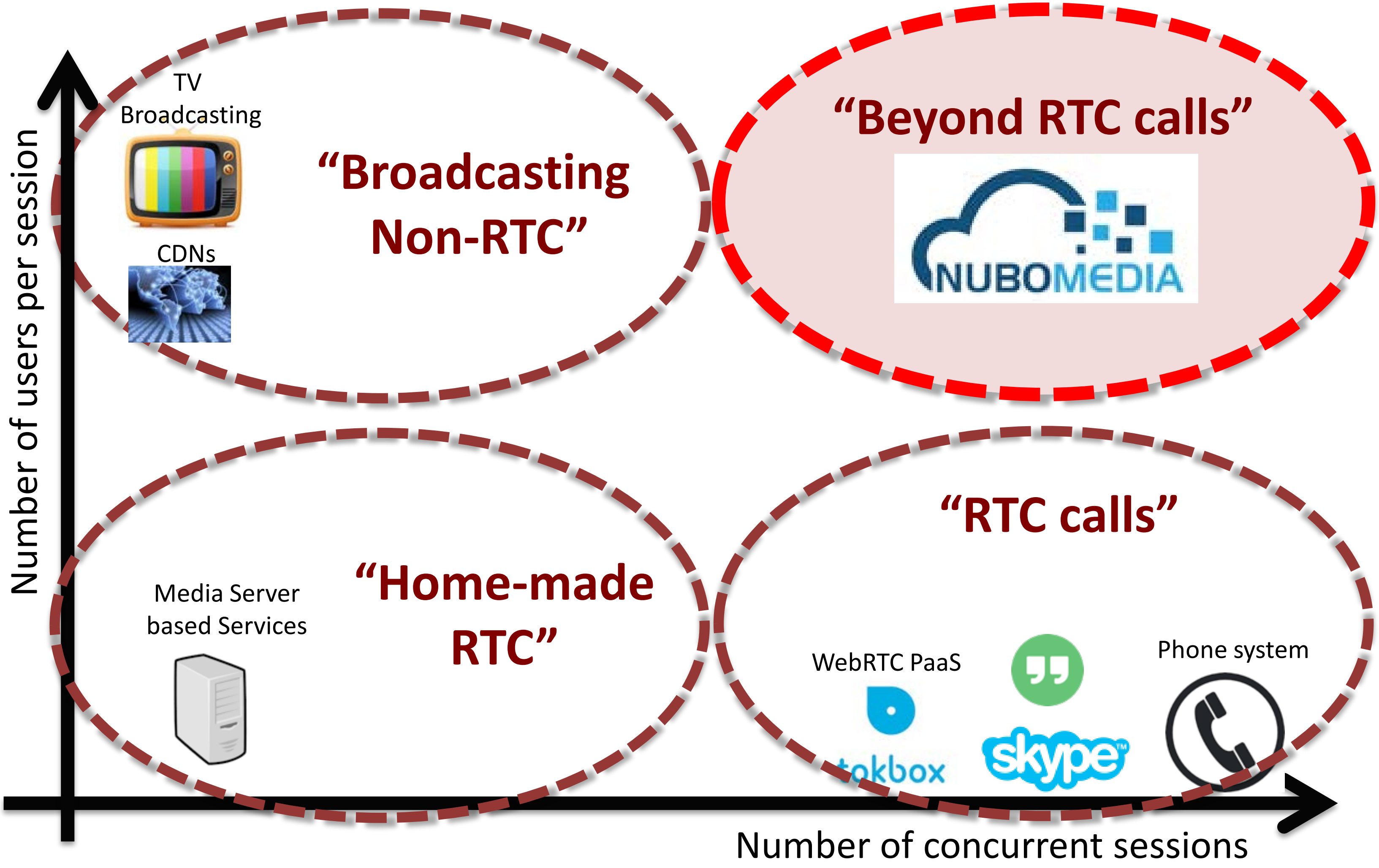
Cloud Orchestrator

IaaS Cloud Manager

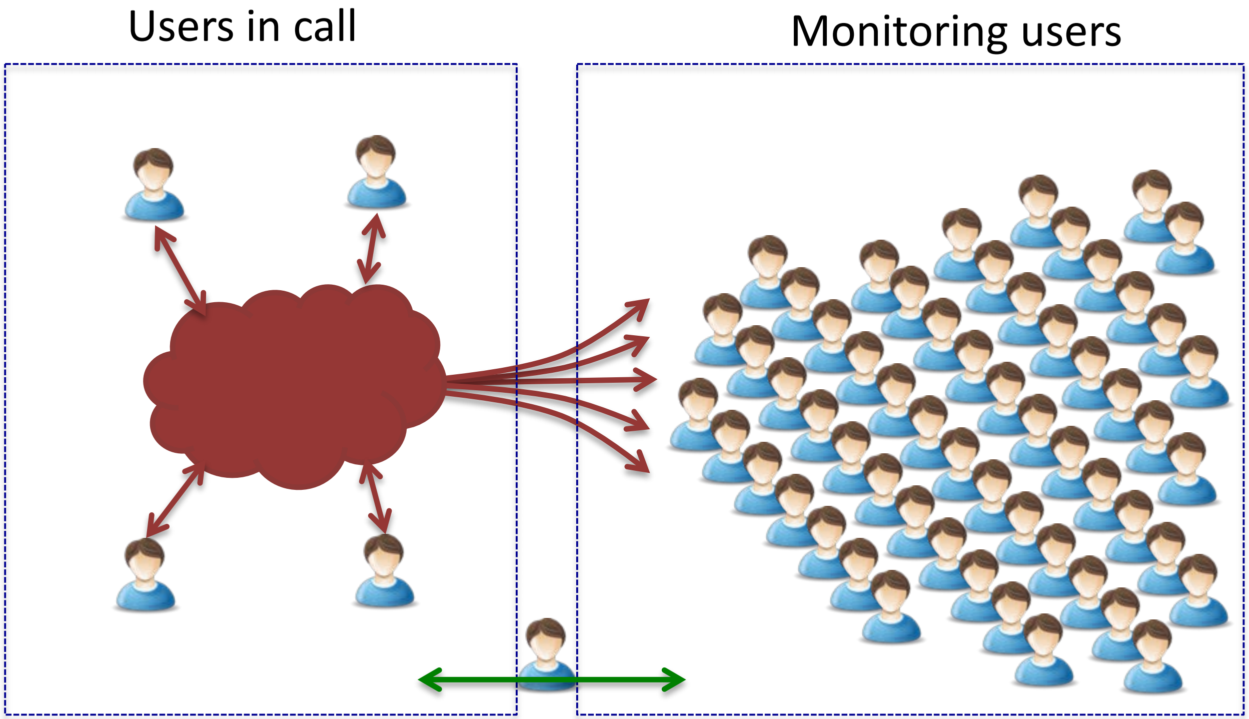
Instance Lifecycle



Scalability of RTC multimedia services

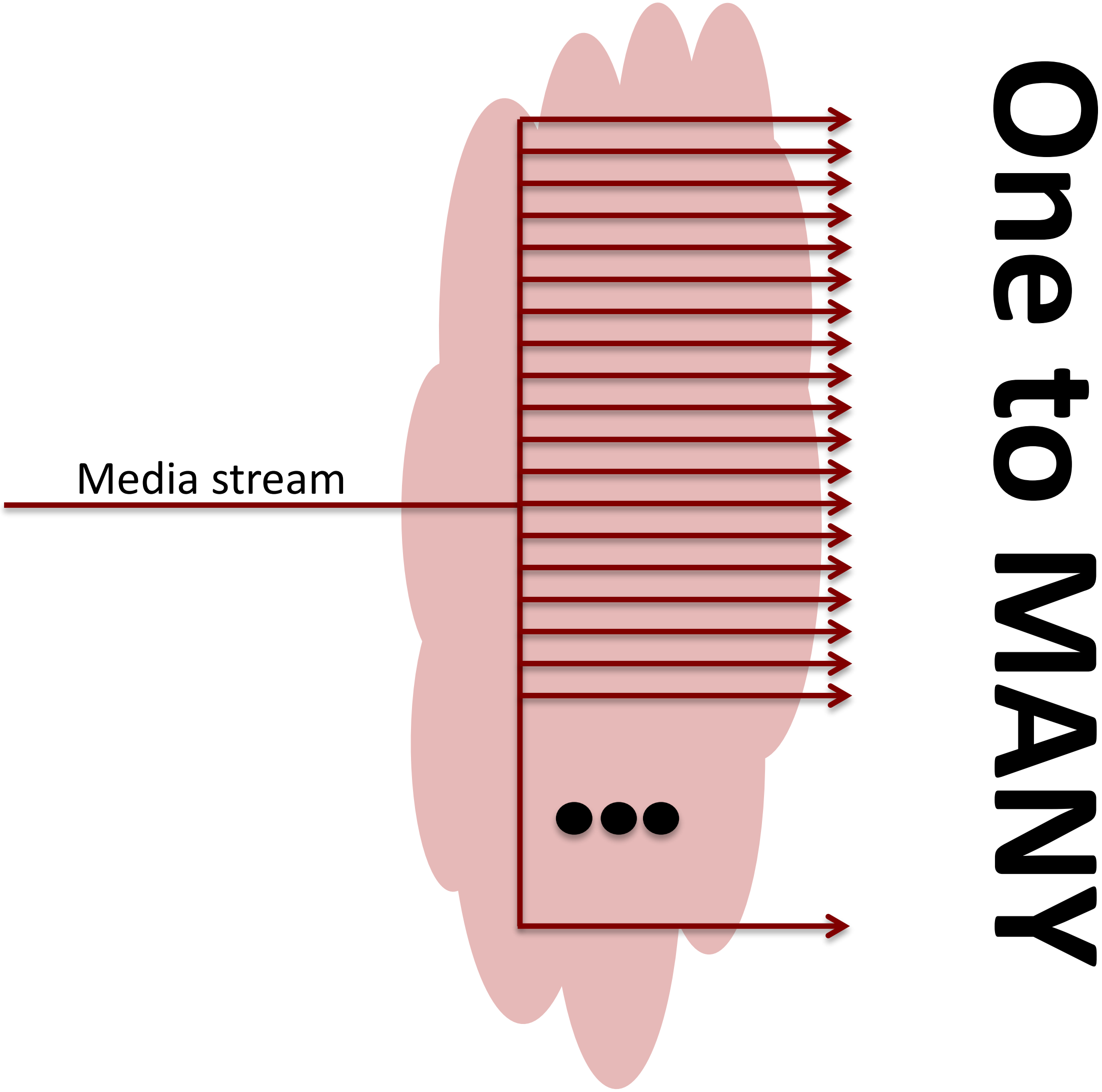


Beyond calls: convergence of broadcasting and phone-like services

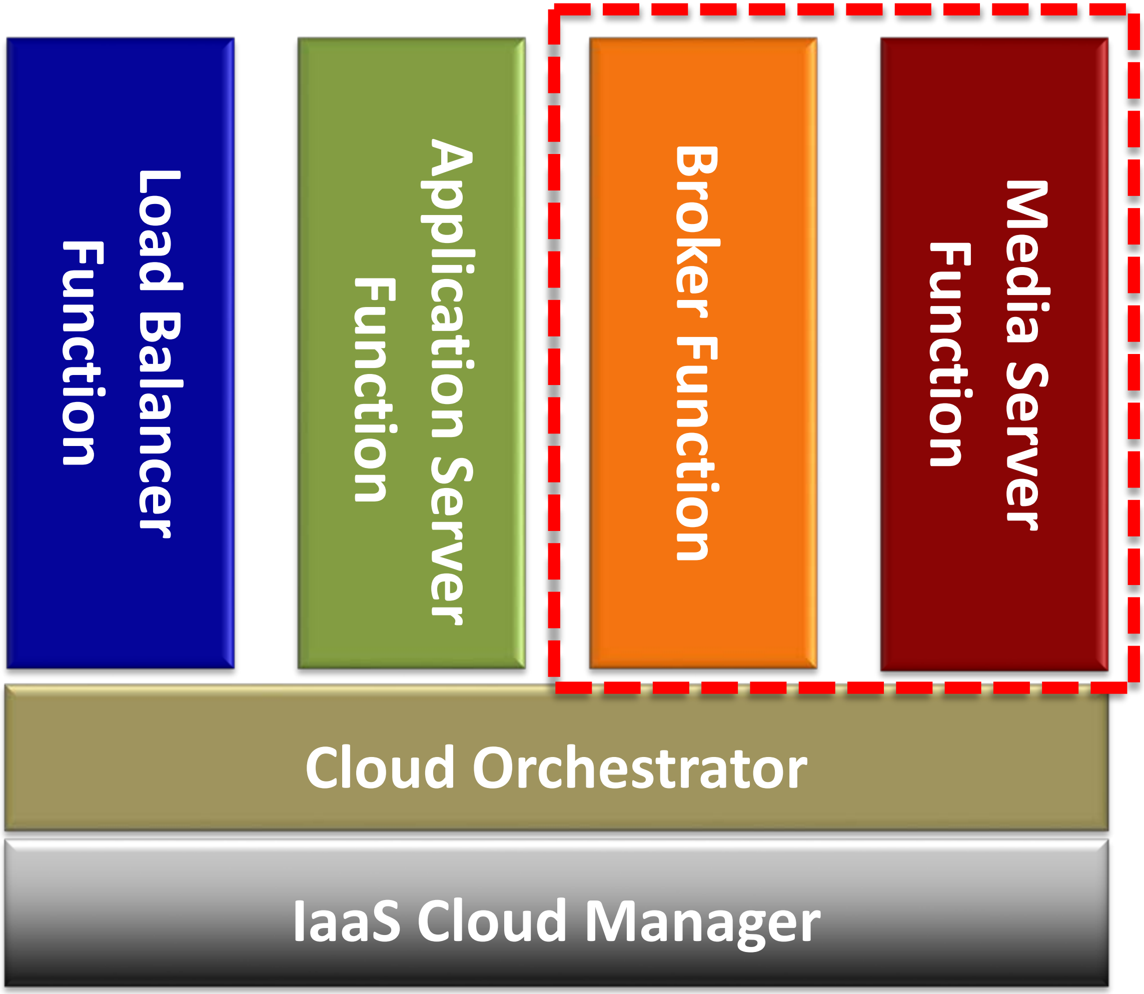


Users can dynamically change their role

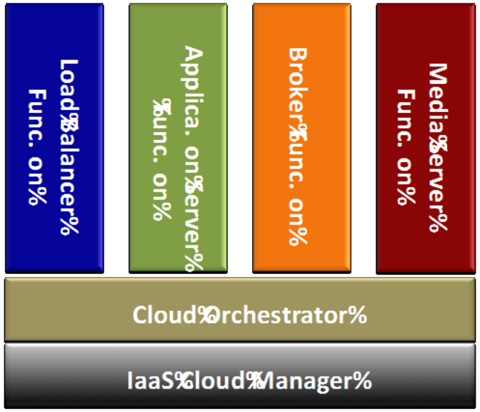
The scalability problem in “beyond call” clouds

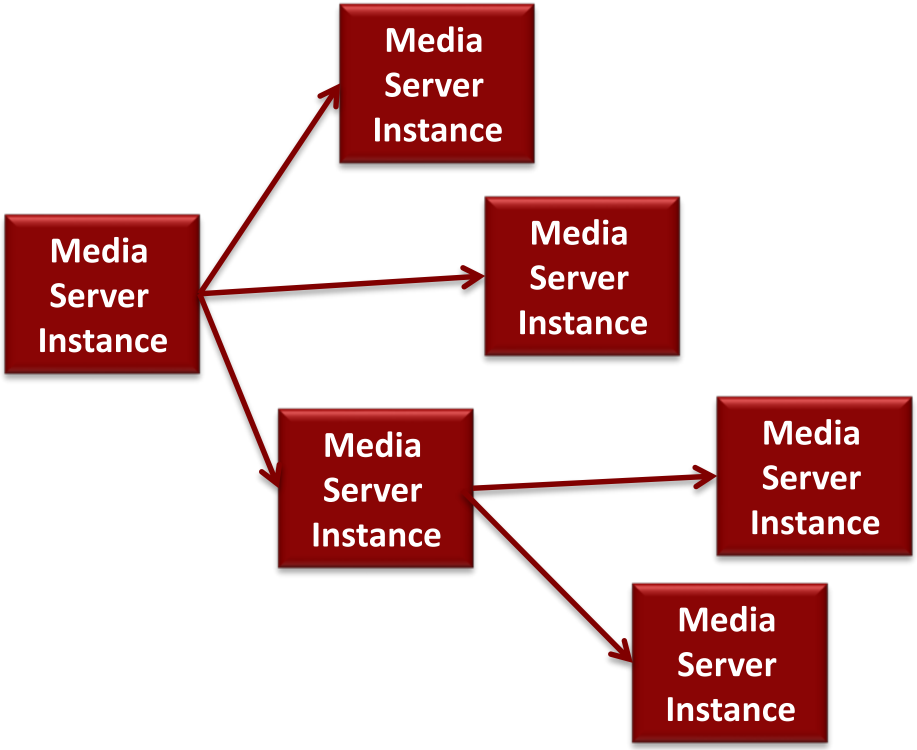


Anatomy of WebRTC PaaS for call models: Hierarchical Architecture



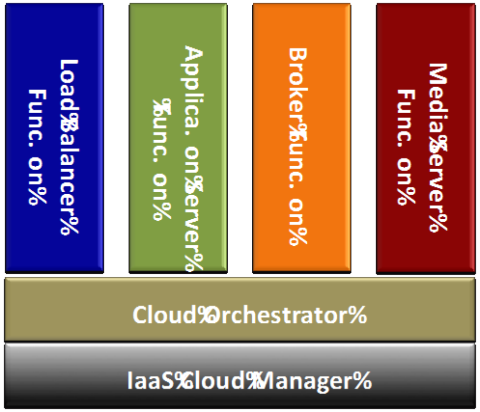
Media Server Function





- 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



Load%Balancer% Func. on%

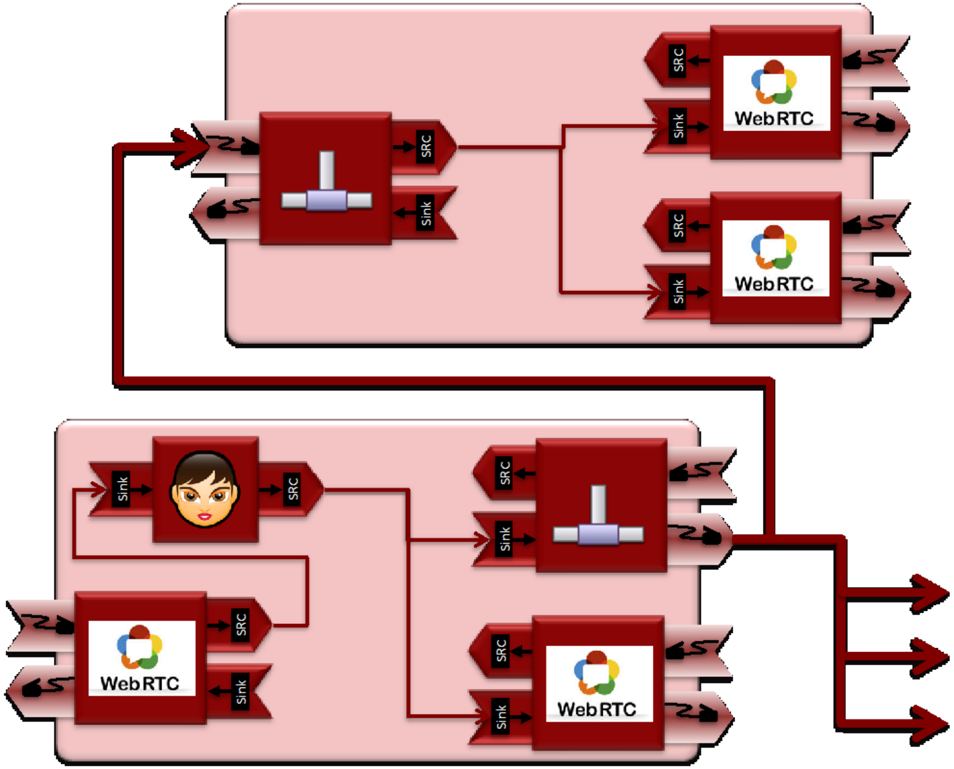
Applica. on%Server% %Func. on%

Broker%Func. on%

Media%Server% Func. on%

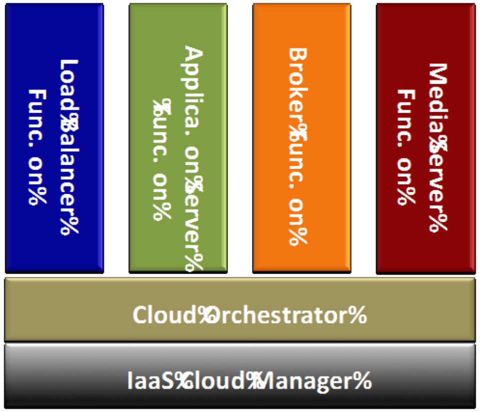
Cloud%Orchestrator%

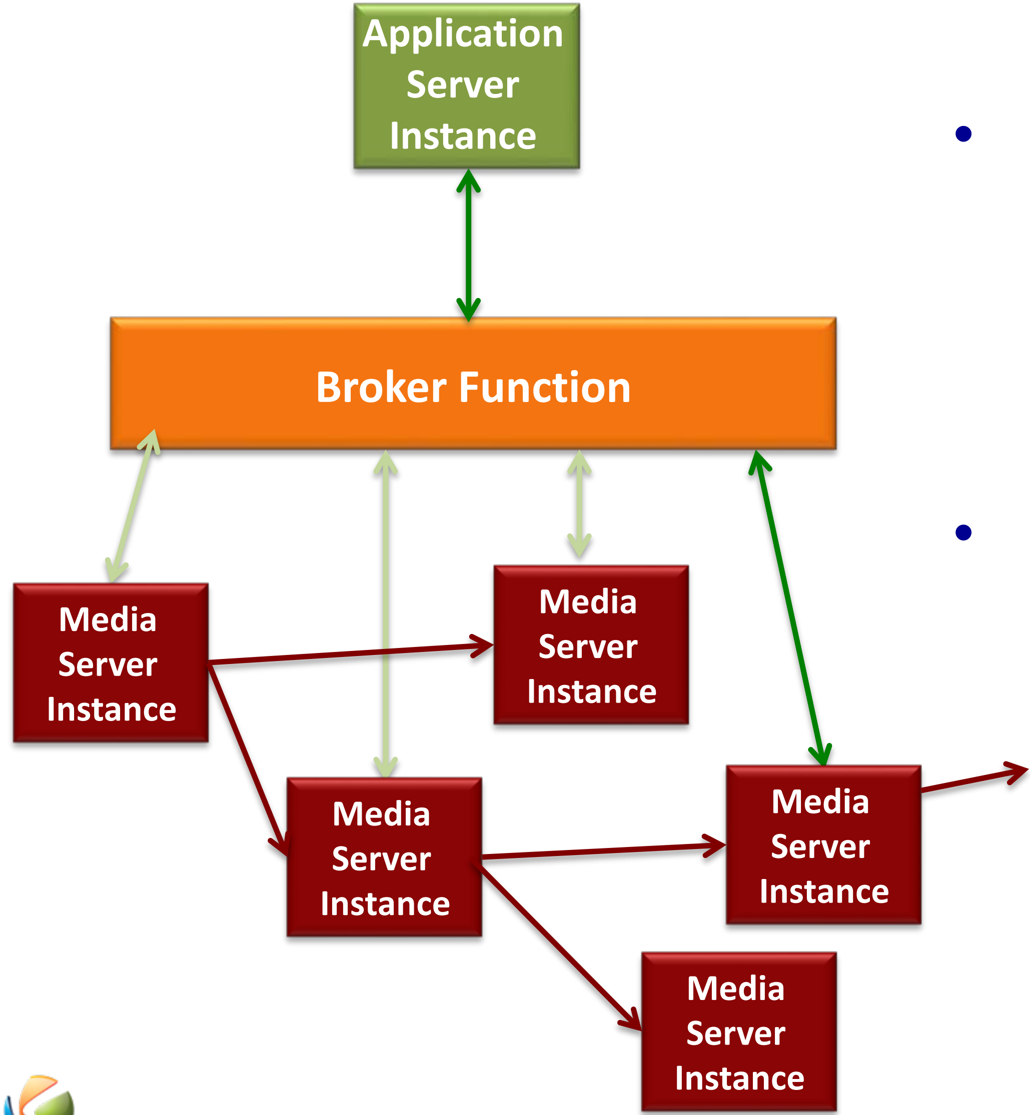
IaaS%Cloud%Manager%



- 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

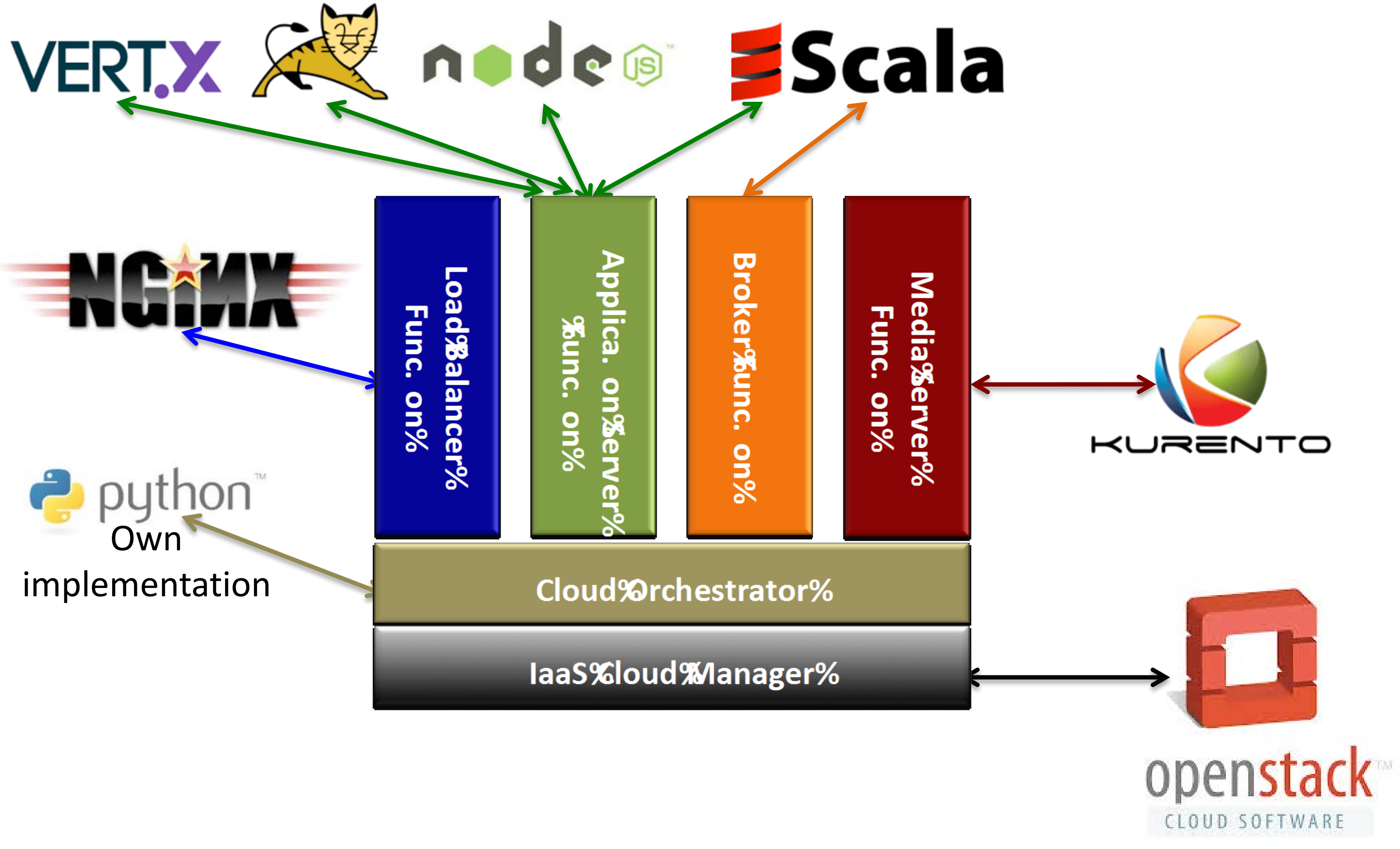
Broker Function





- 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



Load%Balancer% Function%

Application%Server% Function%

Broker%Function%

Media%Server% Function%

Cloud%Orchestrator%

IaaS%Cloud%Manager%

Own implementation