

ĐỒ ÁN CUỐI KÌ MÔN

CÔNG NGHỆ PHẦN MỀM

**NGHIÊN CỨU WEB-RTC KURENTO**

*Người hướng dẫn*: **THẦY LÊ NGỌC THẠCH**

Người thực hiện: **NGUYỄN VI THỊNH - 517H0086**

**MAI DUY BẢO - 518H0474**

**PHAN QUANG DUY – 518H0006**

**ĐOÀN HỒ HOÀNG HIỆP – 518H0176**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2020**



ĐỒ ÁN CUỐI KÌ MÔN

CÔNG NGHỆ PHẦN MỀM

**NGHIÊN CỨU WEB-RTC KURENTO**

*Người hướng dẫn*: **THẦY LÊ NGỌC THẠCH**

*Người thực hiện*: **NGUYỄN VI THỊNH - 517H0086**

**MAI DUY BẢO - 518H0474**

**PHAN QUANG DUY – 518H0006**

**ĐOÀN HỒ HOÀNG HIỆP – 518H0176**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2020**

# LỜI CẢM ƠN

Cảm ơn thầy Lê Ngọc Thạch đã dạy dỗ chúng em trong suốt quá trình học môn công nghệ phần mềm, truyền đạt những kiến thức bổ ích và hỗ trợ những lúc khó khăn, thắc mắc trong quá trình giảng dạy, cảm ơn thầy đã giú p đỡ chúng em học được những kiến thức bổ ích trong công nghệ phần mềm

# ĐỒ ÁN ĐƯỢC HOÀN THÀNH

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của chúng tôi và được sự hướng dẫn của Thầy Lê Ngọc Thạch. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày tháng năm Tác giả*

*(ký tên và ghi rõ họ tên)*

***Nguyễn Vi Thịnh***

***Mai Huy Bảo***

***Phan Quang Duy***

***Đoàn Hồ Hoàng Hiệp***

# PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

#### Phần xác nhận của GV hướng dẫn

Tp. Hồ Chí Minh, ngày tháng năm (kí và ghi họ tên)

#### Phần đánh giá của GV chấm bài

Tp. Hồ Chí Minh, ngày tháng năm (kí và ghi họ tên)

# TÓM TẮT

Đây là bài báo cáo tóm tắt lại các nội dung liên quan đến đề tài tiềm hiểu về webrtc-kurento của nhóm chúng tôi.

# MỤC LỤC

Contents

[LỜI CẢM ƠN 1](#_Toc59726969)

[ĐỒ ÁN ĐƯỢC HOÀN THÀNH 2](#_Toc59726970)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN 1](#_Toc59726971)

[TÓM TẮT 2](#_Toc59726972)

[MỤC LỤC 1](#_Toc59726973)

[DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ 3](#_Toc59726974)

[CHƯƠNG 1 – GIỚI THIỆU JAVA 4](#_Toc59726975)

[1.1 Introduction 4](#_Toc59726976)

[1.2 Scope 4](#_Toc59726977)

[1.2.1 Definitions and Abbreviations 4](#_Toc59726978)

[1.2.2 Reference 4](#_Toc59726979)

[1.3 The Overall Description 5](#_Toc59726980)

[1.3.1 Product Perspective 5](#_Toc59726981)

[1.3.2 Product Functions 6](#_Toc59726982)

[1.3.3 User Characteristics 6](#_Toc59726983)

[1.4 Design and Implementation Constraints 6](#_Toc59726984)

[1.4.1 Know your use case 6](#_Toc59726985)

[1.4.2 Install KMS 7](#_Toc59726986)

[1.4.3 Configure KMS 7](#_Toc59726987)

[CHƯƠNG 2 – ARCHITECTURE DESIGN 7](#_Toc59726988)

[1.1 Kurento architecture 7](#_Toc59726989)

[1.2 Các danh sách chức năng 7](#_Toc59726990)

[1.3 Screenflow 10](#_Toc59726991)

[1.4 Thiết kế kiến trúc 10](#_Toc59726992)

[CHƯƠNG 3: DEPLOYMENT GUIDELINE 13](#_Toc59726993)

[1.1 Giới thiệu 13](#_Toc59726994)

[1.2 Nâng cấp cục bộ 13](#_Toc59726995)

[1.3 STUN / TURN cài đặt máy chủ 14](#_Toc59726996)

[1.4 Kiểm tra cài đặt của bạn 15](#_Toc59726997)

[Chương 4: Development guideline 16](#_Toc59726998)

[4.1 MediaStreamAPI 16](#_Toc59726999)

[4.2 Ứng dụng trong video call bằng web RTC: 22](#_Toc59727000)

[4.2.1 Các thành phần tham gia vào thực hiện ứng dụng video call 22](#_Toc59727001)

[4.3 Quá trình thực hiện ứng dụng video call 23](#_Toc59727002)

[4.3.1 Đăng kí người dùng 23](#_Toc59727003)

[4.3.2 Thực hiện cuộc gọi 23](#_Toc59727004)

[Quy trình lấy được SDP 24](#_Toc59727005)

[Trao đổi ứng viên (candidate) 24](#_Toc59727006)

[Bắt đầu thiết lập cuộc gọi 25](#_Toc59727007)

[Kết thúc cuộc gọi 27](#_Toc59727008)

[TÀI LIỆU THAM KHẢO 29](#_Toc59727009)

# DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

**Hình ảnh:**

[Hình 1: *screenflow của phòng gặp mặt WebRTC* 10](#_Toc59728830)

[Hình 2: Framwork chính của hệ thống 10](#_Toc59728831)

[Hình 3: Kiến trúc của hệ thống 11](#_Toc59728832)

[Hình 4: Trao đổi các luồng đa phương tiện trong hệ thống 12](#_Toc59728833)

[Hình 5: Sơ đồ kiến trúc WebRTC 21](#_Toc59728834)

[Hình 6: Các thành phần tham gia thực hiện cuộc gọi 22](#_Toc59728835)

[Hình 7: Pha đăng kí người dùng 23](#_Toc59728836)

[Hình 8: Quy trình lấy SDP 24](#_Toc59728837)

[Hình 9: Quy trình trao đổi ứng viên 25](#_Toc59728838)

[Hình 10: Các kết nối của Peer A đến KMS 25](#_Toc59728839)

[Hình 11: các bước Signaling cần thực hiện để thiết lập cuộc gọi 26](#_Toc59728840)

[Hình 12: Trao đổi luồng của SS với KMS 27](#_Toc59728841)

[Hình 13: Gửi dữ liệu chuyển tiếp SDP 27](#_Toc59728842)

[Hình 14: Phiên làm việc khi kết thúc tương tác 28](#_Toc59728843)

**Bảng biểu:**

[Bảng 1: Danh sách các chức năng 9](#_Toc59729233)

# CHƯƠNG 1 – GIỚI THIỆU JAVA

## Introduction

SRS này mô tả yêu cầu phần mềm và phi chức năng cho bản phát hành 1.0 của WebRTC. Tài liệu này nhằm mục đích được sử dụng bởi cộng đồng WebRTC, những người sẽ triển khai và xác minh hoạt động chính xác của hệ thống. Trừ khi có ghi chú khác, tất cả các yêu cầu được chỉ định ở đây đều có mức độ ưu tiên cao và được cam kết cho bản phát hành 1.0.

Kurento là một máy chủ đa phương tiện WebRTC và một tập hợp các API khách hàng giúp đơn giản hóa việc phát triển các ứng dụng video nâng cao cho WWW và nền tảng điện thoại thông minh. Các tính năng của Kurento Media Server bao gồm truyền thông nhóm, chuyển mã, ghi âm, trộn, phát sóng và định tuyến các luồng nghe nhìn. Là một tính năng khác biệt, Kurento Media Server cũng cung cấp khả năng xử lý phương tiện tiên tiến liên quan đến thị giác máy tính, lập chỉ mục video, thực tế tăng cường và phân tích giọng nói. Kiến trúc mô-đun Kurento giúp đơn giản hóa việc tích hợp các thuật toán xử lý phương tiện của bên thứ ba (tức là nhận dạng giọng nói, phân tích cảm xúc, nhận dạng khuôn mặt, v.v.), có thể được các nhà phát triển ứng dụng sử dụng một cách minh bạch như phần còn lại của các tính năng tích hợp sẵn của Kurento.

## 1.2 Scope

Tài liệu này có giải thích cấp cao về cách trở thành nhà phát triển KMS. Phát triển Kurento Client Các ứng dụng nằm ngoài phạm vi của tài liệu này và sẽ không được giải thích ở đây. Mã của Kurento Media Server là Mã nguồn mở, được phát hành theo các điều khoản của Giấy phép Apache Phiên bản 2.0 và có sẵn trên GitHub.

### 1.2.1 Definitions and Abbreviations

SRS: System Requirement Specification.

KMS: Kurento Media Server.

RTC: Real-Time Communications.

MCU: Multi-Conference Unit.

PaaS: Platform as a Service.

KTF: Kurento Testing Framework.

E2E: End to end.

### 1.2.2 Reference

Hàm tạo cho WebRtcPeer là WebRtcPeer (chế độ, tùy chọn, gọi lại) trong đó:

mode: Chế độ trong đó PeerConnection sẽ được định cấu hình. Giá trị hợp lệ là

- recv: chỉ nhận media.

- send: chỉ gửi media.

- sendRecv: gửi và nhận phương tiện.

options: Đây là một nhóm các tham số và chúng là tùy chọn. Nó là một đối tượng json.

- localVideo: Thẻ video trong ứng dụng cho luồng cục bộ.

- remoteVideo: Thẻ video trong ứng dụng cho luồng từ xa.

- videoStream: Cung cấp luồng video đã có sẵn sẽ được sử dụng thay vì sử dụng phương tiện phát trực tuyến từ webcam cục bộ.

- audioStreams: Cung cấp luồng âm thanh đã có sẵn sẽ được sử dụng thay vì sử dụng phương tiện phát trực tiếp từ micrô cục bộ.

- mediaConstraints: Xác định chất lượng cho video và âm thanh

- peerConnection: Sử dụng peerConnection đã được tạo trước đó

- sendSource: Nguồn nào sẽ được sử dụng

\* webcam

\* screen

\* window

- onstreamended: Phương thức sẽ được gọi khi sự kiện kết thúc luồng xảy ra

- onicecandidate: Phương thức sẽ được gọi khi sự kiện ứng viên băng xảy ra

- oncandidategatheringdone: Phương thức sẽ được gọi khi tất cả các ứng cử viên đã được thu hoạch

- dataChannels: Cờ cho phép sử dụng các kênh dữ liệu. Nếu đúng, thì một kênh dữ liệu sẽ được tạo trong đối tượng RTCPeerConnection.

- dataChannelConfig: Nó là một đối tượng JSON với cấu hình được truyền cho DataChannel khi được tạo. Nó hỗ trợ các phím sau:

\* id: Chỉ định id của kênh dữ liệu. Nếu không có quy định nào được chỉ định, cùng một id của đối tượng WebRtcPeer sẽ được dùng.

\* options: Đối tượng Options được truyền cho phương thức khởi tạo kênh dữ liệu.

\* onopen: Chức năng được gọi trong sự kiện onopen của kênh dữ liệu, được kích hoạt khi kênh đang mở.

\* onclose: Hàm được gọi trong sự kiện onclose của kênh dữ liệu, được kích hoạt khi kênh dữ liệu đã đóng cửa.

\* onmessage: Hàm được gọi trong sự kiện onmessage của kênh dữ liệu. Sự kiện này được kích hoạt mọi thời gian nhận được tin nhắn.

\* onbufferedamountlow: Là trình xử lý sự kiện được gọi khi nhận được sự kiện bufferedamountlow. Sự kiện như vậy được gửi khi RTCDataChannel.bufferedAmount giảm xuống nhỏ hơn hoặc bằng số lượng được chỉ định bởi thuộc tính RTCDataChannel.bufferedAmountLowThreshold.

\* onerror: Chức năng gọi lại được kích hoạt khi có lỗi trong kênh dữ liệu. Nếu không có gì được cung cấp, thông báo dấu vết lỗi sẽ được ghi vào bảng điều khiển của trình duyệt.

- simulcast: Cho biết liệu simulcast sẽ được sử dụng. Giá trị đúng | sai

- configuration: Nó là một đối tượng JSON nơi Máy chủ ICE được định nghĩa bằng cách sử dụng.

## The Overall Description

### 1.3.1 Product Perspective

Để đánh giá đúng Kurento từ góc độ người dùng cuối cùng, một bộ kiểm tra E2E phong phú đã được thiết kế và được thực hiện. Với mục tiêu đó, Khung kiểm tra Kurento (KTF) đã được tạo ra. KTF là một phần của Kurento nhằm thực hiện các bài kiểm tra end-to-end (E2E) cho Kurento. KTF đã được triển khai trên hai khuôn khổ thử nghiệm Nguồn mở nổi tiếng: JUnit và Selenium.

### 1.3.2 Product Functions

Các phần chính của WebRTC bao gồm:

* getUserMedia, cho phép trình duyệt web truy cập vào camera và/hoặc microphone để lấy dữ liệu hình ảnh âm thanh cho việc truyền tải.
* RTCPeerConnection dùng để cài đặt videocall/voicecall dùng cho việc truyền tải.
* RTCDataChannel cho phép trình duyệt chia sẻ dữ liệu peer-to-peer.

WebRTC API bao gồm chức năng:

* getStats cho phép ứng dụng web lấy tập hợp các số liệu thống kê về các session WebRTC.

### 1.3.3 User Characteristics

Người dùng mục tiêu là người thử nghiệm và quản trị viên.

## Design and Implementation Constraints

### 1.4.1 Know your use case

Chọn giữa Kurento và OpenVidu. Kurento Media Server đã được thiết kế như một nền tảng đa năng có thể được sử dụng để tạo bất kỳ loại các ứng dụng phát trực tuyến đa phương tiện. Điều này làm cho KMS trở thành một công cụ mạnh mẽ, tuy nhiên nó cũng có nghĩa là có một số không thể tránh khỏi sự phức tạp mà nhà phát triển phải đối mặt. WebRTC là một tiêu chuẩn phức tạp với rất nhiều bộ phận chuyển động và bạn cần biết về từng bộ phận này các thành phần và cách chúng làm việc cùng nhau để đạt được truyền thông đa phương tiện mà tiêu chuẩn cố gắng hướng tới phục vụ. Nếu ứng dụng dự kiến ​​của bạn bao gồm một thiết lập phức tạp với các loại nguồn khác nhau và các trường hợp sử dụng khác nhau, thì Kurento là đòn bẩy tốt nhất mà bạn có thể sử dụng. Tuy nhiên, nếu bạn có ý định giải quyết một trường hợp sử dụng đơn giản hơn, chẳng hạn như các trường hợp sử dụng ứng dụng hội nghị truyền hình, thì dự án OpenVidu được xây dựng dựa trên Kurento để cung cấp một giải pháp đơn giản và dễ sử dụng hơn sẽ giúp bạn tiết kiệm thời gian và Nỗ lực phát triển.

### 1.4.2 Install KMS

Cài đặt KMS Hướng dẫn cài đặt giải thích các cách khác nhau mà Kurento có thể được cài đặt trong hệ thống của bạn. Nhanh nhất và cách đơn giản nhất là sử dụng mẫu được định cấu hình trước của chúng tôi cho Amazon AWS.

### Configure KMS

KMS có thể chạy nguyên trạng sau khi cài đặt bình thường. Tuy nhiên, có một số tham số mà bạn có thể muốn để điều chỉnh các tệp cấu hình.

# CHƯƠNG 2 – ARCHITECTURE DESIGN

## 1.1 Kurento architecture

Hiện nay, trên Internet chia sẻ video đã cùng tồn tại chia sẻ trực tuyến và lưu trữ (VOD). Với, Video phát trực tiếp: người dùng quan tâm về độ trễ, mượt mà khi phát video. Bên cạnh đó, chia sẻ VOD: người dùng quan tâm chất lượng, tốc độ tải xuống. Hơn nữa, Dịch vụ mạng xã hội (Facebook,Twitter…) đang phải đối mặt với sự tắc nghẽn máy chủ quan trọng khi một video mới được chia sẻ trong Internet. Số người xem video đó sẽ tăng đột ngột. Cũng đó là người dùng hiện tại, có thể là người dùng Internet có dây (PC) hoặc người dùng di động. Vì vậy, chúng tôi đề xuất một thuật toán để giải quyết vấn đề quan trọng bằng cách tận dụng lợi thế của người dùng Internet(PC) như kết nối Internet tốc độ cao, khả năng cao người dùng trên thiết bị di động có kết nối Internet tốc độ thấp. Chúng tôi sẽ ưu tiên cho người dùng PC trở thành điểm nhấn để ngăn chặn đột ngột rời đi và tham gia của người dùng di động. Qua thử nghiệm, chúng tôi sẽ cho thấy rằng cách tiếp cận của chúng tôi là cách tiếp cận rất hiệu quả trong việc cung cấp dịch vụ phát trực tuyến và VOD trên Web.

## 1.2 Các danh sách chức năng

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Chức năng | Thành phần | Mô tả |
| 1 | Tạo không gian | Không gian | **-**Tạo không gian để quản lý các cuộc họp. Không gian là ảo phòng họp.  -Thêm hoặc chỉnh sửa thành viên vào không gian để gặp gỡ hoặc trò chuyện.  -Mời khách tham gia cuộc họp.  -Chia sẻ nội dung, màn hình hoặc ứng dụng trong cuộc họp  -Tham gia vào phòng họp vơi thiết bị mà bạn đã chọn |
| 2 | Giao diện chính | -Cuộc họp hiện tại | Hiển thị không gian nơi bạn hiện đang họp.  Điều này chỉ xuất hiện nếu bạn hiện đang họp |
|  |  | **-**Danh sách gần đây | Ứng dụng hiển thị danh sách các khoảng trống được sắp xếp theo  hoạt động, lịch sử cuộc gọi cho các cuộc gọi trực tiếp và không gian mà bạn đã tham gia với tư cách khách mời. Chấm màu xanh lam bên cạnh hình đại diện không gian cho biết tin nhắn chưa đọc. Khi bạn đăng xuất, các mục cũ nhất từ danh sách hoạt động bị xóa |
|  |  | -Thanh tìm kiếm | -Sử dụng chức năng này để tìm kiếm ai đó ứng dụng sẽ hiển thị cho bạn thông tin phù hợp. |
|  |  | **-**Nút gọi | **-**Bắt đầu cuộc họp mới hoặc tham gia vào cuộc họp có sẵn |
|  |  | **-**Cài đặt | -Trước khi tham gia cuộc họp, hãy kiểm tra âm thanh và video cài đặt và xem chế độ tự xem của bạn trên thiết bị bạn sẽ sử dụng. Nhấp vào đây để có thêm thông tin |
| 3 | Tùy chỉnh cuộc họp | -Thanh tùy chỉnh cuộc họp | -Khi bạn đang họp, ứng dụng cung cấp cho bạn các tùy chọn để quản lý người tham gia và kiểm soát cuộc họp của bạn. |
|  |  | -Đánh đấu quan trọng | -Đưa người được chọn chiếu màn hình lớn trong cuộc họp hoặc chọn trình chiếu máy tính trên cuộc họp. |
|  |  | -Tắt âm thanh | -Bật/ Tắt micrô |
|  |  | -Tắt camera | -Bật/ Tắt camera |
|  |  | -Thêm/ xóa người | - Thêm hoặc xóa một người được chọn khỏi/ vào cuộc họp. |
| 4 | Ghi và phát cuộc họp | -Thanh Ghi | - Ghi hình cuộc họp đang diễn ra và lưu trữ vào dữ liệu máy tính hoặc dữ liệu đám mây tùy người dùng chọn. |
|  |  | -Thanh Phát | - Phát trực tiếp cuộc họp ra bên ngoài trang chính để mọi người theo dõi |
| 5 | Chia sẻ màn hình và ứng dụng | -Nút chia sẻ | -Chia sẻ đến mọi người ở các trang ứng dụng bên ngoài đã chọn, nếu người dùng đang hoạt động có thể tham gia xem trực tiếp cuộc họp đang diễn ra của bạn. |
| 6 | Khóa/ mở khóa cuộc họp | -Thanh kiểm soát cuộc họp | Bạn có thể khóa hoặc mở khóa cuộc họp từ các tùy chọn menu tham gia nếu bạn có quyền được kích hoạt bởi quản trị viên hệ thống.  1. Tham gia cuộc gọi, sau khi bạn tham gia, hãy nhấp vào để mở thanh kiểm soát cuộc họp  2. Trong Bảo mật, nhấp vào Khóa cuộc họp  3. Để mở khóa, hãy nhấp vào cùng một biểu tượng |
| 7 | Thông tin cuộc gọi | -Tên gọi | Tên của bạn co thể đặt khi đang tham gia trong cuộc họp |
|  |  | -Thời lượng | -Thời lượng cuộc họp tinh từ lúc người đầu tiên vào phòng họp |
| 8 | -Báo cáo lỗi với nhà phát triển | -Thanh báo cáo | - Ấn vào thanh báo cáo khi bạn phát hiện lỗi hoặc gặp trải nghiệm không tốt khi sử dụng dịch vụ hoặc có góp ý phát triển thêm giúp nhà phát triển. |

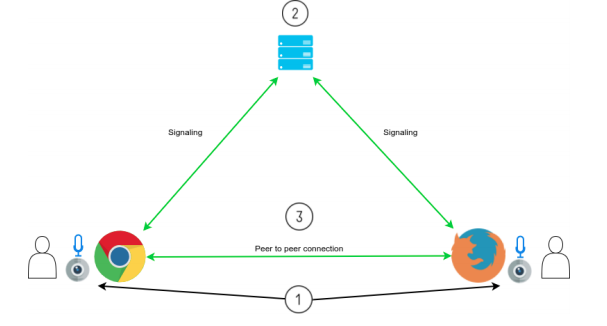
Bảng 1: Danh sách các chức năng

## 1.3 Screenflow

Quy trình cho phép giao tiếp trên WebRTC có thể được coi là quy trình ba bước:

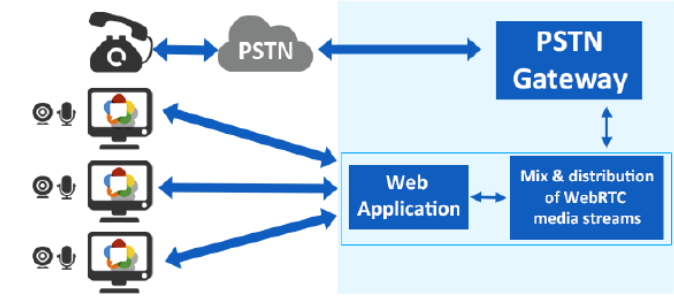
* Trình duyệt có quyền truy cập vào các thiết bị phương tiện (máy ảnh và micro).
* Mỗi đồng đẳng trao đổi thông tin về chính nó với tất cả những người khác thông qua một quá trình báo hiệu.
* Sau khi phát tín hiệu, nhiều người có thể kết nối trực tiếp và cuộc giao tiếp bắt đầu.

Để thực hiện tất cả những điều này, cần có một máy chủ báo hiệu để trao đổi thông tin. Ngoài ra, cần có một cặp máy chủ STUN / TURN để đạt được phương tiện truyền tải NAT và phương tiện chuyển tiếp nếu không thể giao tiếp trực tiếp.



Hình 1: *screenflow của phòng gặp mặt WebRTC*

## 1.4 Thiết kế kiến trúc



Hình 2: Framwork chính của hệ thống

Hoạt động của hệ thống diễn ra như sau:

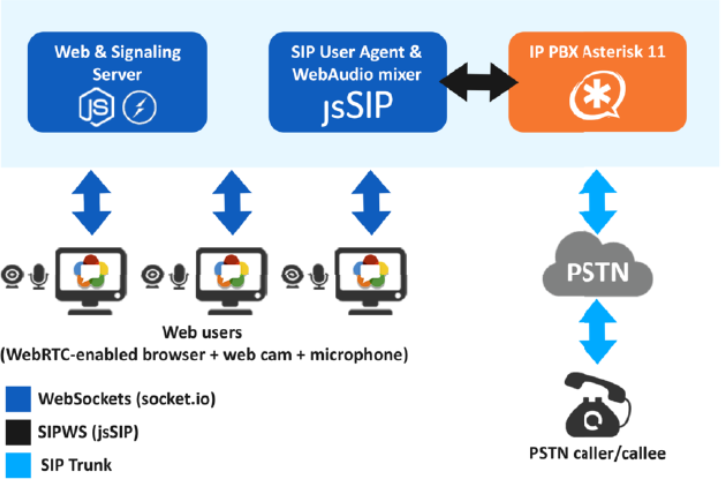
(1)Người dùng đăng nhập ứng dụng web bằng trình duyệt web tương thích và tạo phòng họp. Trong thời điểm đó, người dùng bắt đầu chia sẻ video và giọng nói với mọi người tham gia.

(2) Nếu người dùng khác đăng nhập, các luồng đa phương tiện của họ cũng bắt đầu được chia sẻ với mọi người trong phòng.

(3) Tất cả những gì người dùng có trong trang web của họ a softphone mà họ có thể tạo các cuộc điện thoại đến PSTN. Nếu người dùng quay số và nhấn nút Gọi, những người khác sẽ không quay được và điện thoại của họ sẽ bị chặn.

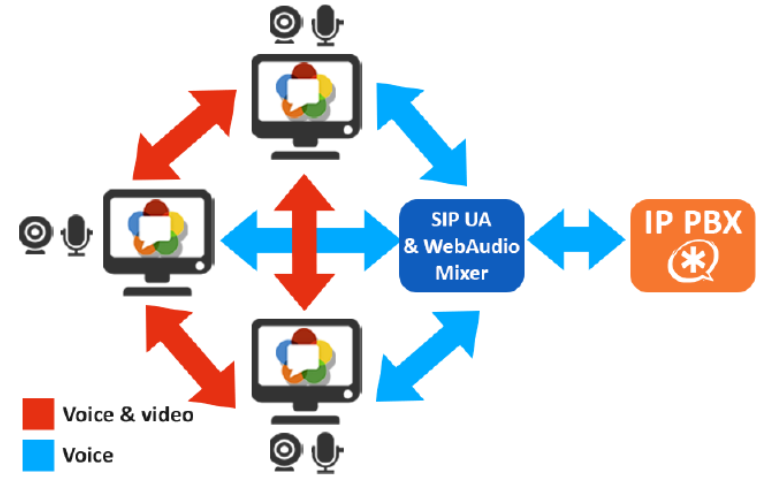
(4) Người dùng bên ngoài có thể quay số được liên kết với đường trục SIP của nhà cung cấp dịch vụ, thực hiện cuộc gọi trong máy chủ IP PBX của hệ thống. Khi nhận được cuộc gọi theo cách này, cuộc gọi sẽ tự động được trả lời và tất cả người dùng có mặt trong phòng đều không bị lỗi. (5) Khi nhận được giọng nói của cuộc gọi, nó sẽ được tích hợp trong phòng để người dùng có thể nghe thấy người gọi, đồng thời nghe thấy giọng nói của những người dùng đổi lại.

(6) Phòng họp sẽ ngừng hoạt động cho đến khi người dùng web cuối cùng kết thúc. Không thể kích hoạt phòng bằng một cuộc gọi mới.



Hình 3: Kiến trúc của hệ thống

Người dùng trong phòng họp truyền tín hiệu chia sẻ giọng nói và video từ máy ảnh trên thiết bị của họ và máy ảnh web, trong khi họ chỉ chia sẻ giọng nói với mô-đun truyền âm thanh. Mô-đun này cung cấp phản hồi giọng nói bên ngoài từ cuộc gọi trong phòng họp, do đó người dùng coi mô-đun này như một người dùng khác về mặt truyền tải bằng cách sử dụng WebRTC.



Hình 4: Trao đổi các luồng đa phương tiện trong hệ thống

WebRTC giới thiệu hỗ trợ cho codec OPUS cho âm thanh và VP8 cho video và nó mặc định cho chúng để sử dụng chất lượng vượt trội của chúng so với các codec khác. Tuy nhiên, WebRTC cũng hỗ trợ các codec khác như G.711, PCMA, PCMU,... Một tính năng khác của WebRTC liên quan đến các luồng phương tiện là tính bảo mật. Các luồng được yêu cầu kết hợp một số loại mã hóa trước khi được gửi đến đồng đẳng khác. Các trình duyệt web hỗ trợ WebRTC sử dụng cơ chế DTLS theo mặc định để đếm với từng đặc điểm.

# CHƯƠNG 3: DEPLOYMENT GUIDELINE

## 1.1 Giới thiệu

Với phương pháp này, bạn sẽ cài đặt Kurento Media Server từ kho gói Ubuntu gốc do dự án Kurento cung cấp.

KMS có hỗ trợ cho hai phiên bản Hỗ trợ dài hạn (LTS) của Ubuntu: Ubuntu 16.04 (Xenial) và Ubuntu 18.04 (Bionic) (chỉ dành cho 64-bit).

Mở một thiết bị đầu cuối và chạy các lệnh sau:

a. Đảm bảo rằng GnuPG đã được cài đặt.

sudo apt-get update && sudo apt-get install --no-install-recommends --yes \

gnupg

b. Thêm kho lưu trữ Kurento vào cấu hình hệ thống.

Chạy các lệnh sau:

**#** Import the Kurento repository signing key

sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 5AFA7A83

**#** Get Ubuntu version definitions

source /etc/upstream-release/lsb-release 2>/dev/null || source /etc/lsb-release

**#** Add the repository to Apt

sudo tee "/etc/apt/sources.list.d/kurento.list" >/dev/null <<EOF

**#** Kurento Media Server - Release packages

deb [arch=amd64] http://ubuntu.openvidu.io/6.15.0 $DISTRIB\_CODENAME kms6

EOF

c. Cài đặt KMS:

**Ghi chú**

Bước này chỉ áp dụng cho cài đặt lần đầu tiên . Nếu bạn đã cài đặt Kurento và muốn nâng cấp nó, hãy làm theo các bước được mô tả tại đây: Nâng cấp cục bộ

sudo apt-get update && sudo apt-get install --no-install-recommends --yes \

kurento-media-server

Thao tác này sẽ cài đặt phiên bản phát hành của Kurento Media Server.

Máy chủ bao gồm các tệp dịch vụ tích hợp với hệ thống init Ubuntu, vì vậy bạn có thể sử dụng các lệnh sau để khởi động và dừng nó:

sudo service kurento-media-server start

sudo service kurento-media-server stop

Thông báo nhật ký từ KMS sẽ có sẵn trong /var/log/kurento-media-server/. Để biết thêm chi tiết về nhật ký KMS, hãy kiểm tra [Debug Logging](https://doc-kurento.readthedocs.io/en/6.15.0/features/logging.html).

## 1.2 Nâng cấp cục bộ

Để nâng cấp cài đặt cục bộ của Kurento Media Server, bạn phải ghi số phiên bản mới vào tệp /etc/apt/sources.list.d/kurento.list, được tạo trong quá trình [Cài đặt cục bộ](https://doc-kurento.readthedocs.io/en/6.15.0/user/installation.html?fbclid=IwAR1QVSQ6aWAUUrBtqaFKEGY7aLJvCkl78MNx8WtUpMxGEZ0ForRCjwrhixw#installation-local) . Sau khi chỉnh sửa tệp đó, bạn có thể chọn giữa 2 tùy chọn để thực sự áp dụng nâng cấp:

1. **Nâng cấp tất cả các gói hệ thống** .

Đây là quy trình tiêu chuẩn được mong đợi bởi phương pháp luận của người bảo trì Debian & Ubuntu. Nâng cấp tất cả các gói hệ thống là một cách để đảm bảo rằng mọi thứ đều được đặt thành phiên bản mới nhất và tất cả các bản sửa lỗi và cập nhật bảo mật cũng được áp dụng, vì vậy đây là phương pháp được khuyến nghị nhất:

sudo apt-get update && sudo apt-get dist-upgrade

Tuy nhiên, đừng làm điều này bên trong vùng chứa Docker. Việc chạy *nâng cấp apt-get* hoặc *nâng cấp bản phân phối apt-get không được chấp nhận* bởi các [phương pháp hay nhất của Docker](https://docs.docker.com/develop/develop-images/dockerfile_best-practices/#apt-get) ; thay vào đó, bạn chỉ nên chuyển sang phiên bản mới hơn của [hình ảnh Kurento Docker](https://hub.docker.com/r/kurento/kurento-media-server) .

1. **Gỡ cài đặt phiên bản Kurento cũ trước khi cài đặt phiên bản mới.**

Tuy nhiên, lưu ý rằng **apt-get không đủ tốt** để loại bỏ tất cả các gói Kurento. Chúng tôi khuyên bạn nên sử dụng *aptitude* cho điều này, hoạt động tốt hơn nhiều so với *apt-get* :

sudo aptitude remove '?installed?version(kurento)'

sudo apt-get update && sudo apt-get install --no-install-recommends --yes \

kurento-media-server

Ghi chú:

Hãy cẩn thận! Nếu bạn không nâng cấp tất cả các gói Kurento, bạn sẽ gặp phải các hành vi sai và bị treo . Kurento bao gồm một số gói:

• hiện-media-server

• hiện tại-mô-đun-người tạo

• km-lõi

• kms-phần tử

• kms-bộ lọc

• libnice10

• libusrsctp

• openh264

• openwebrtc-gst-plugins

• Và nhiều hơn nữa

Để sử dụng phiên bản mới hơn, bạn phải nâng cấp tất cả các gói Kurento , không chỉ gói đầu tiên.

## 1.3 STUN / TURN cài đặt máy chủ

Làm việc với WebRTC *đòi hỏi các* nhà phát triển phải học và hiểu rõ về mọi thứ liên quan đến NAT, ICE, STUN và TURN.

Kurento Media Server, cũng giống như bất kỳ điểm cuối WebRTC nào, sẽ tự hoạt động tốt, đối với các kết nối *LAN* hoặc đối với các máy chủ có địa chỉ IP công cộng được gán cho chúng. Tuy nhiên, sớm hay muộn, bạn sẽ muốn làm cho ứng dụng của mình hoạt động trong môi trường đám mây với tường lửa NAT và cho phép KMS kết nối với các máy khách từ xa. Đồng thời, các máy khách từ xa có thể cũng muốn kết nối từ phía sau bộ định tuyến [NAT](https://doc-kurento.readthedocs.io/en/6.15.0/glossary.html#term-nat) của chính họ , vì vậy ứng dụng của bạn cần được chuẩn bị để thực hiện [NAT Traversal](https://doc-kurento.readthedocs.io/en/6.15.0/glossary.html#term-nat-traversal) ở cả hai phía. Điều này có thể được thực hiện bằng cách thiết lập máy chủ [STUN](https://doc-kurento.readthedocs.io/en/6.15.0/glossary.html#term-stun) hoặc chuyển tiếp [TURN](https://doc-kurento.readthedocs.io/en/6.15.0/glossary.html#term-turn) và cấu hình nó **trong cả KMS và trình duyệt máy khách** .

## 1.4 Kiểm tra cài đặt của bạn

Để xác minh rằng quy trình Kurento đang hoạt động, hãy sử dụng lệnh này và tìm quy trình *kurento-media-server* :

**$** ps -fC kurento-media-server

UID PID PPID C STIME TTY TIME CMD

kurento 7688 1 0 13:36 ? 00:00:00 /usr/bin/kurento-media-server

Trừ khi được cấu hình khác, KMS sẽ lắng nghe trên cổng TCP 8888, để nhận Yêu cầu RPC và gửi Phản hồi RPC bằng [Giao thức Kurento](https://doc-kurento.readthedocs.io/en/6.15.0/features/kurento_protocol.html) . Sử dụng lệnh này để xác minh rằng cổng này đang mở và đang lắng nghe các gói tin đến:

**$** sudo netstat -tupln | grep -e kurento -e 8888

tcp6 0 0 :::8888 :::\* LISTEN 7688/kurento-media-

Bạn có thể thay đổi các thông số này trong tệp /etc/kurento/kurento.conf.json.

Để kiểm tra xem KMS có hoạt động hay không và đang lắng nghe các kết nối, hãy sử dụng lệnh sau:

curl \

--include \

--header "Connection: Upgrade" \

--header "Upgrade: websocket" \

--header "Host: 127.0.0.1:8888" \

--header "Origin: 127.0.0.1" \

http://127.0.0.1:8888/kurento

Bạn sẽ nhận được phản hồi tương tự như sau:

HTTP/1.1 500 Internal Server Error

Server: WebSocket++/0.7.0

Bỏ qua thông *báo* “ *Lỗi máy chủ* ”: điều này được mong đợi và nó thực sự chứng minh rằng KMS đang hoạt động và lắng nghe các kết nối.

Nếu bạn cần để tự động này, bạn có thể viết một kịch bản tương tự như healthchecker.sh , là chúng ta sử dụng trong hình ảnh Kurento Docker.

# Chương 4: Development guideline

## 4.1 MediaStreamAPI

MediaStreamAPI (còn được gọi là getUserMediaAPI)

Các MediaStreamAPI đại diện cho đồng bộ dòng phương tiện truyền thông. Ví dụ: một luồng được lấy từ đầu vào máy ảnh và micrô đã đồng bộ hóa các bản nhạc và video.

Có lẽ cách dễ nhất để hiểu MediaStreamAPI là xem xét nó một cách tự nhiên:

1.Trong trình duyệt của bạn, điều hướng đến các mẫu **WebRTCgetUserMedia** .

2.Mở bảng điều khiển.

3.Kiểm tra stream biến nằm trong phạm vi toàn cầu.

Mỗi cái đều MediaStream có một đầu vào, có thể là một MediaStreamđược tạo bởi **getUserMedia()** và một đầu ra, có thể được chuyển cho một phần tử video hoặc một **RTCPeerConnection**.

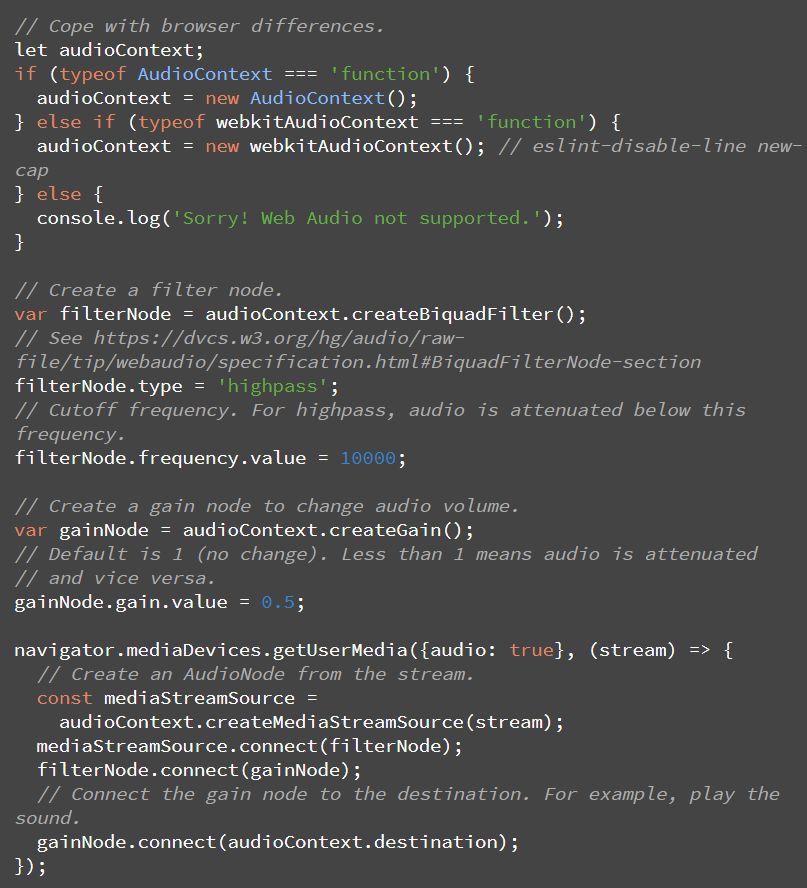
Các **getUserMedia()** phương pháp có một MediaStreamConstraints đối tượng tham số và trả về một Promise mà giải quyết cho một MediaStream đối tượng.

Mỗi MediaStream có một label, chẳng hạn như 'Xk7EuLhsuHKbnjLWkW4yYGNJJ8ONsgwHBvLQ'. Một mảng MediaStreamTracks được trả về bởi các phương thức getAudioTracks() và getVideoTracks().

Ví getUserMediadụ, **stream.getAudioTracks()** trả về một mảng trống (vì không có âm thanh) và giả sử một webcam đang hoạt động được kết nối, **stream.getVideoTracks()** trả về một mảng MediaStreamTrack đại diện cho luồng từ webcam. Mỗi kênh MediaStreamTrack có một loại ( 'video' hoặc 'audio'), một label (ví dụ như 'FaceTime HD Camera (Built-in)') và đại diện cho một hoặc nhiều kênh âm thanh hoặc video. Trong trường hợp này, chỉ có một đoạn video và không có âm thanh, nhưng có thể dễ dàng hình dung các trường hợp sử dụng có nhiều hơn, chẳng hạn như ứng dụng trò chuyện lấy luồng từ camera trước, camera sau, micrô và một ứng dụng chia sẻ màn.

Một MediaStream có thể được gắn vào phần tử video bằng cách đặt srcObjectthuộc tính. Trước đây, điều này đã được thực hiện bằng cách đặt src thuộc tính thành URL đối tượng được tạo bằng **URL.createObjectURL(),** nhưng hiện nay phương pháp này không còn được sử dụng nữa

getUserMediacũng có thể được sử dụng làm nút đầu vào cho API âm thanh web :



Các ứng dụng và tiện ích mở rộng dựa trên Chromium cũng có thể kết hợp getUserMedia. Thêm **audioCapture** và / hoặc **videoCapture** quyền vào tệp kê khai cho phép quyền được yêu cầu và cấp chỉ một lần khi cài đặt. Sau đó, người dùng không được yêu cầu quyền truy cập máy ảnh hoặc micrô.

Quyền chỉ phải được cấp một lần cho **getUserMedia().** Lần đầu tiên, nút Cho phép được hiển thị trên thanh thông tin của trình duyệt . Quyền truy cập HTTP cho **getUserMedia()** đã bị Chrome ngừng sử dụng vào cuối năm 2015 do tính năng này được phân loại là Tính năng mạnh.

Mục đích có khả năng cho phép MediaStream bất kỳ nguồn dữ liệu phát trực tuyến nào, không chỉ máy ảnh hoặc micrô. Điều này sẽ cho phép phát trực tuyến từ dữ liệu được lưu trữ hoặc các nguồn dữ liệu tùy ý, chẳng hạn như cảm biến hoặc các đầu vào khác.

**Constraints (Ràng buộc)**

Ràng buộc có thể được sử dụng để đặt giá trị cho độ phân giải video **getUserMedia().** Điều này cũng cho phép hỗ trợ các ràng buộc khác , chẳng hạn như tỷ lệ khung hình; chế độ camera (camera trước hoặc sau); tỷ lệ khung hình, chiều cao và chiều rộng; và một **applyConstraints()**.

Việc đặt giá trị hạn chế không được phép đưa ra một DOMExceptionhoặc một OverconstrainedErrornếu, ví dụ, một nghị quyết yêu cầu không có sẵn. Để xem điều này thực tế, hãy xem các mẫu WebRTC **getUserMedia**: chọn độ phân giải cho bản trình diễn.

**Screen and tab capture (màn hình và tab)**

Ứng dụng Chrome cũng giúp bạn có thể chia sẻ video trực tiếp của một tab trình duyệt hoặc toàn bộ màn hình thông qua **chrome.tabCapture** và **chrome.desktopCapture** các API.

Cũng có thể sử dụng chụp màn hình làm **MediaStream** nguồn trong Chrome bằng cách sử dụng **chromeMediaSource** ràng buộc thử nghiệm . Lưu ý rằng chụp màn hình yêu cầu HTTPS và chỉ nên được sử dụng để phát triển do nó được bật thông qua cờ dòng lệnh.

**Signaling: Session control, network, and media information (Báo hiệu: Kiểm soát phiên, mạng và thông tin phương tiện)**

WebRTC sử dụng **RTCPeerConnection** để giao tiếp dữ liệu trực tuyến giữa các trình duyệt (còn được gọi là ngang hàng (peers)), nhưng cũng cần một cơ chế để điều phối giao tiếp và gửi các thông điệp điều khiển, một quá trình được gọi là báo hiệu. Các phương pháp và giao thức báo hiệu không được WebRTC chỉ định. Báo hiệu không phải là một phần của **RTCPeerConnection** API.

Thay vào đó, các nhà phát triển ứng dụng WebRTC có thể chọn bất kỳ giao thức nhắn tin nào họ thích, chẳng hạn như SIP hoặc XMPP và bất kỳ kênh giao tiếp song công (hai chiều) nào thích hợp. Các appr.tc dụ sử dụng XHR và API Kênh là cơ chế truyền tín hiệu. Bộ mã sử dụng Socket.io chạy trên máy chủ Node .

-Báo hiệu được sử dụng để trao đổi ba loại thông tin:

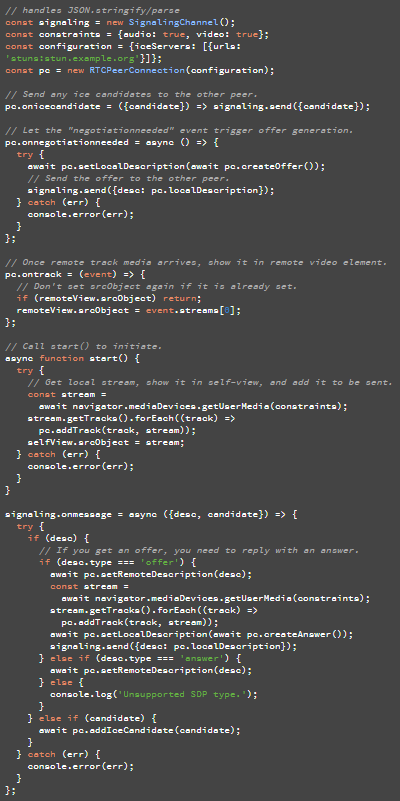
1.Thông báo điều khiển phiên: để khởi tạo hoặc đóng giao tiếp và báo lỗi.

2.Khả năng phương tiện: codec và độ phân giải nào có thể được xử lý bởi trình duyệt của bạn và trình duyệt mà nó muốn giao tiếp?

3.Việc trao đổi thông tin thông qua báo hiệu phải hoàn tất thành công trước khi phát trực tuyến ngang hàng có thể bắt đầu.

=>**Việc trao đổi thông tin thông qua báo hiệu phải hoàn tất thành công trước khi phát trực tuyến ngang hàng có thể bắt đầu.**

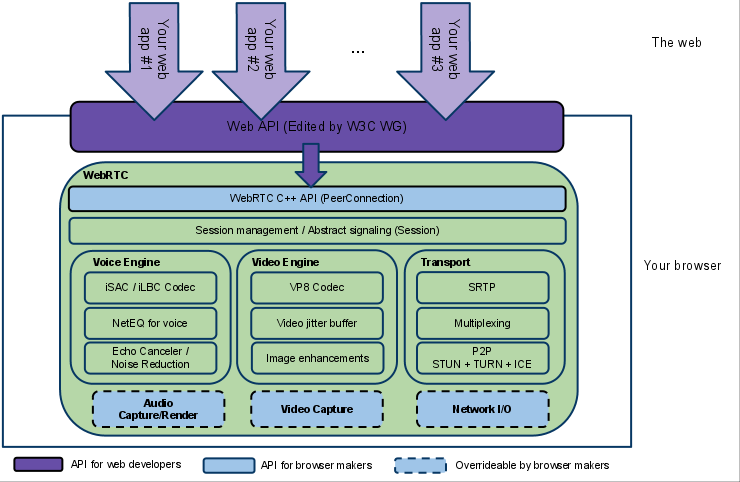
Ví dụ: Mã giả định sự tồn tại của một số cơ chế báo hiệu được tạo trong createSignalingChannel()phương thức. Cũng lưu ý rằng trên Chrome và Opera, RTCPeerConnectionhiện đã có tiền tố.



**RTCPeerConnection**

**RTCPeerConnection** là thành phần WebRTC xử lý giao tiếp ổn định và hiệu quả về truyền dữ liệu.

Sơ đồ kiến ​​trúc WebRTC cho thấy vai trò của **RTCPeerConnection**:



Hình 5: Sơ đồ kiến trúc WebRTC

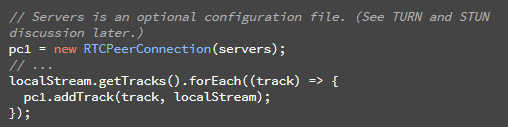
RTCPeerConnection without server (RTCPeerConnection không có máy chủ)

Đoạn mã sau được lấy từ các mẫu WebRTC Kết nối ngang nhau, có kết nối cục bộ và từ xa **RTCPeerConnection** (và video cục bộ và từ xa) trên một trang web. người gọi và người được gọi nằm trên cùng một trang, nó làm cho hoạt động của **RTCPeerConnectionAPI** rõ ràng hơn một chút vì các RTCPeerConnection đối tượng trên trang có thể trao đổi dữ liệu và thông điệp trực tiếp mà không cần phải sử dụng cơ chế báo hiệu trung gian.

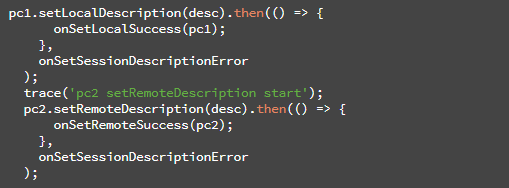
Trong ví dụ này, pc1đại diện cho ngang hàng cục bộ (người gọi) và pc2đại diện cho ngang hàng từ xa (callee).

**a.Người gọi:**

Tạo **RTCPeerConnection** và thêm luồng từ **getUserMedia()**:

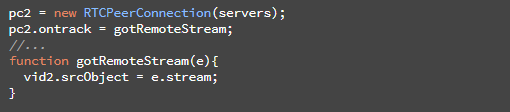


Tạo và đặt nó làm mô tả cục bộ cho pc1và làm mô tả từ xa cho pc2. Điều này có thể được thực hiện trực tiếp trong mã mà không cần sử dụng tín hiệu vì cả người gọi và người gọi đều ở trên cùng một trang:



**b.Người được gọi:**

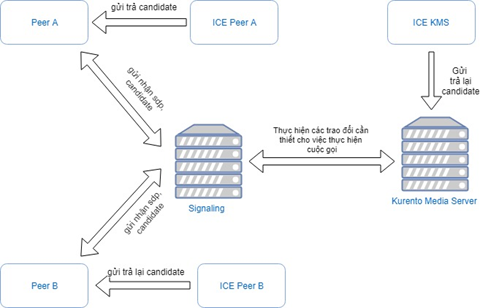
Tạo pc2 và khi luồng từ pc1 được thêm vào, hiển thị nó trong phần tử video:



## 4.2 Ứng dụng trong video call bằng web RTC:

## 4.2.1 Các thành phần tham gia vào thực hiện ứng dụng video call

Với các ứng dụng WebRTC thông thường, việc thực hiện các ứng dụng media sử dụng API WebRTC cung cấp thông thường hầu như khá phức tạp, rất khó để cho lập trình viên. Như đã đề cập đến ở chương 2, kurento cung cấp một số API nhằm mục đích đơn giản hóa việc xây dựng các ứng dụng đa phương tiện, một phần trong các API này dựa vào WebRTC API có sẵn. Một điều nữa là mặc dù các ứng dụng WebRTC chưa hẳn là peer-to-peer, nó vẫn cần một server đứng giữa để thực hiện việc trao đổi các thông tin cần thiết giữa các peer, có thể cần một media server cho việc nâng cao hiệu suất của ứng dụng WebRTC. Dưới đây là một số thành phần tham gia vào khi thực hiện cuộc gọi.



Hình 6: Các thành phần tham gia thực hiện cuộc gọi

Tóm lượt lại hình trên ta có thể thấy có 4 thành phần chính tham gia vào quá trình thực hiện. Dưới đây là mô tả tóm gọn các thành phần:

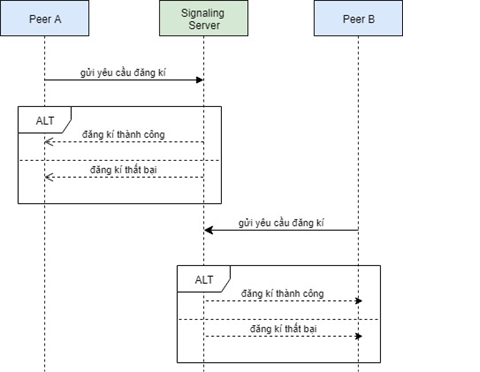
* Peer: Chính là đối tượng kết nối ( ở đây có thể là trình duyệt web,mobile…).
* Signaling Server: Như đã giới thiệu ở các phần trên, ở đây nó đứng giữa thực hiện trao đổi với các peer và KMS.
* ICE (Interactive Connectivity Establishment): Quá trình hỗ trợ vượt NAT, nó sẽ nhận đăng kí từ các peer và trả về candidate (là các cách để kết nối với peer đó). ICE ở đây có thể là tập hợp bao gồm STUN, TURN server được đặt ở các đối tượng là peer và KMS. Phần này đã được giới thiệu ở phần trước

Các đối tượng này sẽ tham gia vào trong quá trình thực hiện cuộc gọi video. Về cơ bản ta có thể chia quá trình này thành 3 quá trình con là đăng kí, thực hiện cuộc gọi,kết thúc cuộc goi.Chi tiết về quá trình thực hiện sẽ được mô tả bên dưới.

## 4.3 Quá trình thực hiện ứng dụng video call

### 4.3.1 Đăng kí người dùng

Bước này thực hiện khá đơn giản với việc có 3 tác nhân là người dùng A, người dùng B và signaling server. Server này sẽ là trung gian xử lý các yêu cầu đăng kí của hai bên (ví dụ có thể lưu vào database). Nó cũng đưa ra các quyết định liệu có thể cho người dùng đăng kí hay không.Có hai trạng thái mà nó gửi về cho người dùng ở bước này là đăng kí thành công và đăng kí thất bại. Hình dưới mô tả pha đăng kí người dùng.

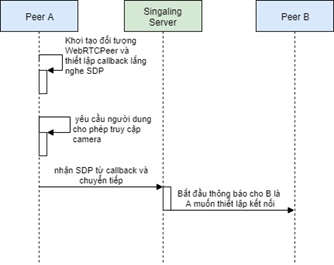


Hình 7: Pha đăng kí người dùng

### 4.3.2 Thực hiện cuộc gọi

Bước chủ yếu được giới thiệu ở đây là thực hiện cuộc gọi. Ở bước này các peer sẽ phải thực hiện một số trao đổi nhất định để có thể đạt được kết nối với KMS. Các trao đổi này được trung gian qua Signaling Server và lấy được candidate thông qua các ICE đứng bên cạnh các đối tượng. Dưới đây là mô tả chi tiết các quy trình để thực hiện một cuộc gọi video.

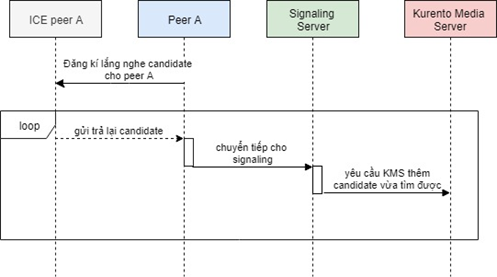
**Quy trình lấy được SDP**



Hình 8: Quy trình lấy SDP

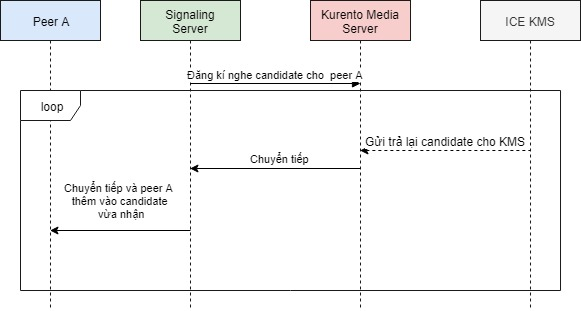
Khi thực hiện việc đăng kí thành công, để bắt đầu thiết lập cuộc gọi, mỗi bên cần biết một thông số nhất định. Đầu tiên, bắt đầu khởi tạo một đối tượng là WebRtcPeer ( đã nhắc qua ở chương 2).Sau đó ta bắt đầu thiết lập một callback để lắng nghe SDP offer được gửi về. Khi nhận được SDP thành công, Peer A bắt đầu gửi yêu cầu cấp quyền truy cập một số tài nguyên thiết bị tới người dùng.Nếu người dùng đồng ý, chúng ta bắt đầu thực hiện quá trình tiếp theo là gửi các thông tin ban đầu từ Peer A ( như là SDP, đến từ đâu, muốn kết nối tới đâu…) đến signaling.Signaling có nhiệm vụ lưu lại các thông tin này và bắt đầu thông báo tới trình duyệt B là có ai đó muốn kết nối.Khi Peer B nhận được thông báo bên A muốn kết nối, nó sẽ thiết lập tương tự như Peer A khi lấy SDP offer và truy cập thiết bị. Khi đó Signaling sẽ có thông tin của hai Peer và bắt đầu vào quá trình thiết lập cuộc gọi với KMS.

**Trao đổi ứng viên (candidate)**



Hình 9: Quy trình trao đổi ứng viên

Quá trình này song song với quá trình đạt được SDP, mục đích của quá trình này nhằm mở ra các cách khác nhau mà cách peer khác có thể kết nối với A.Đầu tiên ta cần làm là đăng kí lắng nghe candidate tới ICE peer A bằng cách ta sẽ thiết lập một callback là OnIceCandidate ( đây là một option khi tạo đối tượng WebRtcPeer).Khi đó,ICE cho trình duyệt A sẽ thực hiện chức năng của nó là tìm các cách để có thể kết nối với A. Khi tìm được, nó sẽ gửi lại cho trình duyệt A thông qua hàm callback OnIceCandidate.Bước tiếp theo trình duyệt A gửi lại cho signaling server, server này đơn giản là ủy quyền lại cho Kurento Media Sever (KMS) bằng cách gọi addCandidate để yêu cầu thêm vào các cách để kết nối với A (candidate).Vì vậy KMS sẽ biết cách để kết nối với peer A.Việc này tương tự với Peer B.

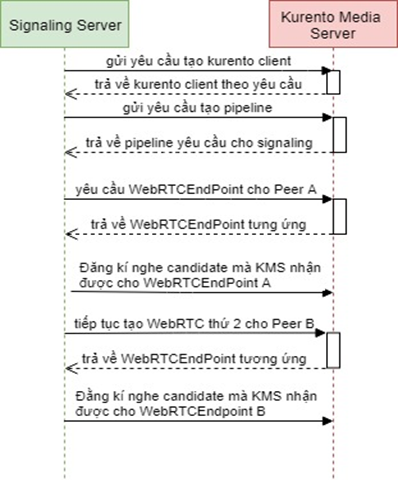


Hình 10: Các kết nối của Peer A đến KMS

Quá trình này được thực hiện khi mà WebRtcEndpoint đã được tạo ra bên trong KMS. Khi đó ta cần biết cách để biết làm sao để kết nối với WebRtcEndpoint tương ứng. Bằng cách thêm vào một máy chủ ICE bên cạnh KMS (có thể là STUN hoặc TURN). Nó sẽ nói cho KMS biết làm cách nào để kết nối với WebRtcEndpoint trong đó. Từ đó KMS sẽ gửi các cách kết nối với WebRtcEndpoint này (candidate) trở lại chỗ mà nó đã đăng kí mỗi khi mà máy chủ ICE gửi trả về một candidate tương ứng. Các candidate này được gửi trở lại các peer qua Signaling làm trung gian. Các peer sẽ thực hiện công việc cuối cùng là thêm cái candidate đó vào (bằng cách thực hiện addCandidate ở đối tượng WebRtcPeer) và quá trình này có thể sẽ phải lặp lại nhiều lần. Nếu quá trình trao đổi thành công, Peer sẽ biết cách để kết nối với WebRtcEndpoint trong KMS vừa tạo. Kết quả từ hai quá trình kể trên là các Peer và KMS đã hiểu cách kết nối với nhau.

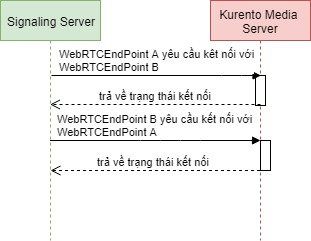
**Bắt đầu thiết lập cuộc gọi**

Ngay sau khi B chấp nhận kết nối, chúng ta bắt đầu thiết lập các bước cần thiết để có thể đạt được cuộc gọi giữa hai. Trọng phần này các bước sẽ được thực hiện ở Signaling, dưới đây là mô tả các bước Signaling cần thực hiện để thiết lập cuộc gọi.



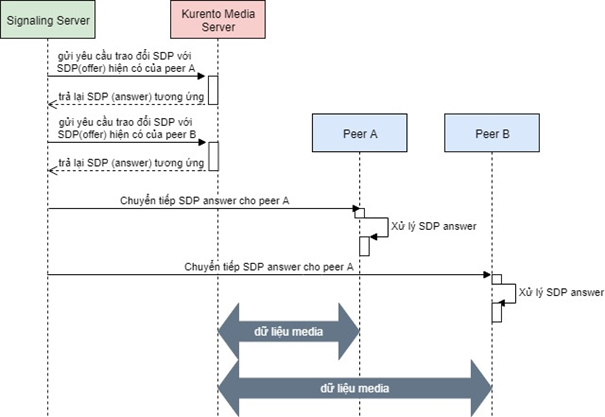
Hình 11: các bước Signaling cần thực hiện để thiết lập cuộc gọi

Ở bước này đầu tiên ta cần phải tạo một kurentoClient bằng cách gọi hàm kurento ( được kurento cung cấp ) nhằm kết nối tới KMS.Sau khi đã kết nối được với KMS,chúng ta cần gửi yêu cầu đến tạo một pipeline, chúng ta sử dụng đối tượng kurentoClient đã tạo trước đó gọi hàm create (được thiết kế để tương tác với KMS như đề cập ở chương trước) với tham số là MediaPipeline.Tiếp đó, ngay sau khi có được pipeline, chúng ta cần hai WebRTCEndpoint cho hai trình duyệt tương ứng A và B. Mục đích của việc này là để các luồng media hoạt động trên KMS có thể trao đổi với nhau.



Hình 12: Trao đổi luồng của SS với KMS

Mặc dù đã tạo hai WebRTCEndPoint nhưng chúng ta cần phải làm một điều là kết nói chúng lại với nhau, việc kết nối này được thực hiện trên cả hai WebRTCEndPoint. Thực hiện điều này bằng cách sử dụng đối tượng hai đối tượng WebRTCEndPoint ở trên và thực hiện gọi invoke với tham số là ‘connect’. Việc này được thực hiện lần lượt với cả hai WebRTCEndPoint.

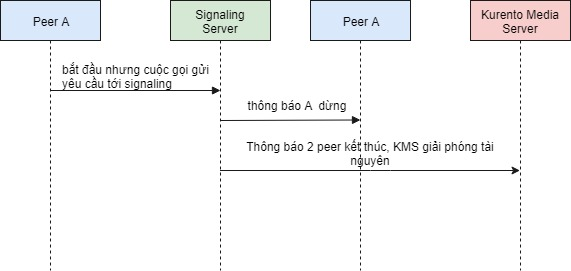


Hình 13: Gửi dữ liệu chuyển tiếp SDP

Bước cuối cùng trong việc thực hiện thiết lập cuộc gọi là việc trao đổi SDP với KMS.Bên phía signaling đã có hai SDP offer được gửi cho nó từ lần trước.Cái signaling làm là gọi sử dụng đối tượng WebRTCEndPoint mà KMS đã trả về từ trước, gọi hàm processOffer với tham số là SDP offer tương ứng với mỗi peer và một callback.Khi thực hiện điều này, bên phía KMS sẽ đáp lại SDP answer và gửi trả về tương ứng với mỗi peer.Sau khi nhận được SDP answer, mỗi peer sẽ gọi processAnswer để xử lý SDP answer được trả về. Từ đây mỗi peer có thể tương tác với nhau, gửi/nhận dữ liệu media qua KMS.

**Kết thúc cuộc gọi**

Pha cuối cùng trong quy trình là ngừng cuộc gọi. Khi có yêu cầu ngừng từ một peer, peer này sẽ kết thúc phiên tương tác bằng cách ngắt kết nối camera và gửi yêu cầu tới signaling, signaling sẽ tiếp tục gửi lại yêu cầu này tới peer đang tương tác và thực hiện tương tự. Tiếp theo signaling sẽ gửi tới KMS yêu cầu ngắt kết nối với hai peer và giải phóng tài nguyên, kết thúc phiên làm việc. Hình dưới mô tả phiên làm việc kết thúc tương tác.



Hình 14: Phiên làm việc khi kết thúc tương tác

# TÀI LIỆU THAM KHẢO