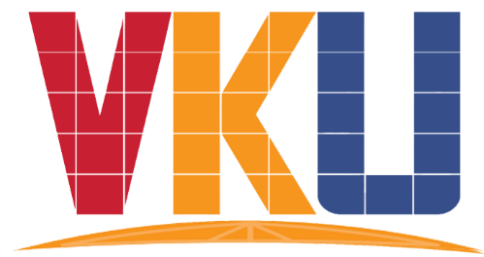
**ĐẠI HỌC ĐÀ NẴNG**

**ĐẠI HỌC CNTT & TRUYỀN THÔNG VIỆT - HÀN**



**ĐỒ ÁN MÔN HỌC**

**LẬP TRÌNH MẠNG**

**Đề tài: LẬP TRÌNH ỨNG DỤNG CHAT WEBRTC VÀ FILE SHARING**

Sinh viên thực hiện:  **Nguyễn Trung Hiếu – 19IT155**

**Lê Trần Thu Loan – 19IT…**

Lớp :19IT3

Giảng viên hướng dẫn: **ThS. Nguyễn Thanh Cẩm**

***Đà nẵng, tháng 11 năm 2021***

# LỜI NÓI ĐẦU

Khi nhu cầu của người dùng tăng cao tạo cơ hội không chỉ cho các hãng sản xuất mà các những nhà **lập trình mạng**. Từ việc phát triển các ứng dụng giúp việc khai thác hết những tính năng, lợi ích của thiết bị hữu ích này trở nên dễ dàng hơn. Lúc này, những người xây dựng ứng dụng mạng được quan tâm, chú ý với cơ hội phát triển bản thân, phát triển sự nghiệp hiệu quả.

Lập trình ứng dụng mạng chính là việc sử dụng những [ngôn ngữ lập trình](https://mona.media/cac-ngon-ngu-lap-trinh-pho-bien-2017/) để viết ứng dụng phát triển những ứng dụng để gia tăng tới mức tốt nhất tiện ích cho người dùng. Trong đó, những ứng dụng phổ biến như chat, game, từ điển, đọc truyện, tổng hợp thông tin chứng khoán, giá vàng, hay truy cập mạng xã hội, ứng dụng cho doanh nghiệp.

Ngày nay, với sự phát triển nhanh chóng của xã hội, nhu cầu giải trí thông qua điện thoại di động ngày càng phổ biến, vì vậy em đã chọn đề tài **“LẬP TRÌNH ỨNG DỤNG CHAT WEBRTC VÀ SHARE FILE ”** Ứng dụng sẽ giúp người dùng có một cái nhìn tốt hơn về mạng xã hội cũng như biết cách chọn lọc thông tin khi dùng mạng xã hội. Đề tài gồm các nội dung sau:

* Mở đầu
* Chương 1: Cơ sở lý thuyết
* Chương 2: Phân tích và thiết hệ thống
* Chương 3: Xây dựng chương trình

Bằng sự cố gắng nỗ lực và đặc biệt là sự giúp đỡ tận tình, chu đáo của giảng viên hướng dẫn ThS. Nguyễn Thanh Cẩm, em đã hoàn thành đồ án đúng thời hạn. Do thời gian làm đồ án có hạn và trình độ còn nhiều hạn chế nên không thể tránh khỏi những thiếu sót. Em rất mong nhận được sự đóng góp ý kiến của các thầy cô cũng như là của các bạn sinh viên để bài đồ án này hoàn thiện hơn nữa.

*Đà Nẵng, ngày…tháng…năm 2021*

Sinh viên thực hiện

Nguyễn Trung Hiếu

Lê Trần Thu Loan

**LỜI CẢM ƠN**

Em xin chân thành cảm ơn sự giúp đỡ nhiệt tình của giảng viên hướng dẫn ThS. Nguyễn Thanh Cẩm, người đã định hướng, hướng dẫn cũng như hỗ trợ cho Em trong quá trình chuẩn bị và tiến hanh để Em có thể hoàn thành tốt đồ án này.

Em cũng xin gửi lời cảm ơn tới các thầy cô giáo, giảng viên trong Trường Đại học Công nghệ thông tin và Truyền thông – ĐHĐN đã cung cấp cho em những kiến thức cần thiết để thực hiện đồ án này.

Em cũng xin gửi lời cảm ơn đến gia đình, bạn bè luôn động viên giúp đỡ em trong suốt thời gian học tập và nghiên cứu, đóng góp những kinh nghiệm quý báu trong thời gian thực hiện đề tài này.

Kính chúc thầy cô mạnh khỏe, công tác tốt, tiếp tục giảng dạy và đào tạo thế hệ trẻ thành công.

**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

.......................................................................................................................................... .......................................................................................................................................... .......................................................................................................................................... .......................................................................................................................................... .......................................................................................................................................... .......................................................................................................................................... .......................................................................................................................................... .......................................................................................................................................... .......................................................................................................................................... .......................................................................................................................................... .......................................................................................................................................... .......................................................................................................................................... .......................................................................................................................................... .......................................................................................................................................... .......................................................................................................................................... .......................................................................................................................................... .......................................................................................................................................... .......................................................................................................................................... .......................................................................................................................................... .......................................................................................................................................... .......................................................................................................................................... .......................................................................................................................................... ..........................................................................................................................................

*Đà Nẵng, ngày…tháng…năm 2021*

Giảng viên hướng dẫn

ThS. Nguyễn Thanh Cẩm

**MỤC LỤC**

[LỜI NÓI ĐẦU 2](#_Toc88087868)

[DANH MỤC HÌNH 1](#_Toc88087869)

[MỞ ĐẦU 1](#_Toc88087870)

[Lý do chọn đề tài 1](#_Toc88087871)

[Đối tượng, phương pháp nghiên cứu 1](#_Toc88087872)

[Phạm vi nghiên cứu 1](#_Toc88087873)

[CHƯƠNG I: CƠ SỞ LÝ THUYẾT 3](#_Toc88087874)

[1.1. TỔNG QUAN VỀ WEBRTC 3](#_Toc88087875)

[1.1.1. Quá trình phát triển 3](#_Toc88087876)

[1.1.2. Kiến trúc WebRTC 6](#_Toc88087877)

[1.1.3. Các APIs trong WebRTC 9](#_Toc88087878)

[1.1.4. Các tầng giao thức trong WebRTC 12](#_Toc88087879)

[1.2. SƠ LƯỢC VỀ NODEJS 16](#_Toc88087880)

[1.2.1. Các đặc tính NodeJS 16](#_Toc88087881)

[1.2.2. Hai NodeJS framwork sử dụng phổ biến 17](#_Toc88087882)

[CHƯƠNG II: PHÂN TÍCH THIẾT KẾ HỆ THỐNG 19](#_Toc88087883)

[2.1. Biểu đồ UseCase 19](#_Toc88087884)

[2.3. Biểu đồ hoạt động 20](#_Toc88087885)

[CHƯƠNG III: XÂY DỰNG CHƯƠNG TRÌNH 21](#_Toc88087886)

[3.1. NHỮNG KẾT QUẢ KHI NGHIÊN CỨU XONG 21](#_Toc88087887)

[3.1.1. Những kết quả đạt được 21](#_Toc88087888)

[3.1.2. Những thiếu sót 21](#_Toc88087889)

[3.2. HÌNH ẢNH CHƯƠNG TRÌNH 21](#_Toc88087890)

[3.3. HƯỚNG PHÁT TRIỂN 24](#_Toc88087891)

# DANH MỤC HÌNH

[Hình 1.1. Mô hình OSI 2](#_Toc55172744)

[Hình 2.1. Giao diện và lớp mô tả của giao diện 8](#_Toc55803768)

[Hình 2.2. Biểu đồ minh họa 9](#_Toc55803769)

[Hình 2.3. Kiến trúc phân tầng của RMI 9](#_Toc55803770)

[Hình 2.4. Hoạt động của các thành phần trong kiến trúc RMI 10](#_Toc55803771)

[Hình 2.5. Vai trò của các lớp trung gian stub và skel 10](#_Toc55803772)

[Bảng 3. 1. Khởi động Server 16](#_Toc55803773)

[Bảng 3.2. Giao diện cửa sổ chat 17](#_Toc55803774)

[Bảng 3.3. Hình ảnh khi người dùng tham gia phòng chat 17](#_Toc55803775)

[Bảng 3.4. Hình ảnh khi người dùng thực hiện hội thoại nhóm 18](#_Toc55803776)

[Bảng 3.5. Chức năng chat mật 18](#_Toc55803777)

[Bảng 3.6. Người dùng thoát khỏi phòng chat 19](#_Toc55803778)

# 

# MỞ ĐẦU

## Lý do chọn đề tài

Cùng với sự bùng nổ công nghệ, người dùng Internet, nhu cầu giao tiếp, chia sẻ thông tin, trao đổi dữ liệu ngày càng lớn. Về chia sẻ thông tin và dữ liệu, trên thế giới đã có rất nhiều hình thức với các công nghệ, giao thức, ứng dụng khác nhau, từ FTP, Email đến các hình thức chia sẻ P2P (Peer-to-Peer) như Bitorrent, hoặc ứng dụng dịch vụ cloud như Dropbox, OneDrive, Google Drive... Về giao tiếp thời gian thực thì đã có những ứng dụng messenger rất thành công và được người dùng chào đón như Skype, Viber, Whatsapp, Line, Hangouts…Tuy nhiên, vì nhiều lý do từ tốc độ, bảo mật an toàn thông tin và đặc biệt là sự tiện dụng, vẫn tiếp tục có các nghiên cứu để đơn giản hóa việc giao tiếp, chia sẻ dữ liệu, hỗ trợ người dùng một cách nhanh nhất mà không đòi hỏi phải thao tác nhiều hay cài đặt thêm các plugin hoặc ứng dụng trên máy. Cụ thể hơn, mong muốn sử dụng trình duyệt không chỉ để lướt web, check mail mà như là một công cụ hỗ trợ tất cả nhu cầu từ chia sẻ file đến giao tiếp thời gian thực từ lâu đã được nhen nhóm và thực sự phát triển mạnh từ năm 2009. Ý tưởng ban đầu từ Google với dự án mã nguồn mở browser-based real-time communication, gọi là WebRTC, mục đích chính là tạo khả năng giao tiếp thời gian thực giữa trình duyệt. Đến nay WebRTC được thiết kế để có thể tích hợp với các hệ thống truyền thông hiện tại như VoIP, các SIP client khác nhau, thậm chí cả mạng PSTN. WebRTC đang tiếp tục phát triển, được các tổ chức tiêu chuẩn thế giới bàn thảo để chuẩn hóa các giao thức, các APIs trong trình duyệt để hỗ trợ WebRTC. WebRTC cũng được những vendor trình duyệt lớn hỗ trợ trong việc phát triển, đảm bảo trình duyệt có thể kết nối trực tiếp với nhau và thực hiện được các yêu cầu về thời gian thực trong giao tiếp. Điều này sẽ mở ra một giai đoạn mới của Web, thực sự mang Web đến với thế giới viễn thông

## Đối tượng, phương pháp nghiên cứu

a,Đối tượng

Đối tượng nghiên cứu của đề tài là môn lập trình mạng nói chung cũng như lập trình WebRTC nói riêng.

b, Phương pháp nghiên cứu

*Phương pháp nghiên cứu tài liệu*: Tìm và đọc các tài liệu liên quan đến kiến thức môn lập trình mạng và phương pháp lập trình đối tượng phân tán (Slide, tài liệu của khoa, trường.

*Phương pháp thực nghiệm*: Tạo ra sản phẩn demo rồi không ngừng cải tiến.

## Phạm vi nghiên cứu

Đồ án tập trung tìm hiểu về công nghệ WebRTC, các APIs trình duyệt, các giao thức được WebRTC sử dụng để có thể chia sẻ và truyền dữ liệu trực tiếp thời gian thực giữa các trình duyệt trong môi trường mạng. Luận văn cũng phân tích yêu cầu tính chất “thời gian thực” khi truyền dữ liệu media và cách thức WebRTC đang được xây dựng để giải quyết, cũng như cách thức vượt NAT, Firewall để thiết lập kết nối Peer to Peer. Luận văn đi sâu vào nghiên cứu phần báo hiệu (phần quan trọng nhưng không chuẩn hóa trong WebRTC), những luồng tiến trình trong quá trình báo hiệu. Dựa trên kết quả nghiên cứu về WebRTC đến thời điểm hiện tại, luận văn chỉ ra được những hướng tiếp cận với WebRTC để phục vụ phát triển những ứng dụng web giao tiếp thời gian thực. Cuối cùng là căn cứ trên hiện trạng của Trung tâm dịch vụ Đa Phương tiện và giá trị gia tăng Mobifone – Tổng Công ty viễn thông Mobifone, luận văn đưa ra ứng dụng demo cho giải pháp cộng tác giúp chia sẻ dữ liệu đa Phương tiện trong Trung tâm trên nền một nền tảng WebRTC là EasyRTCIV.

# CHƯƠNG I: CƠ SỞ LÝ THUYẾT

## 1.1. TỔNG QUAN VỀ WEBRTC

WebRTC (Web Real-Time Communication) [26] là một tiêu chuẩn định nghĩa một tập hợp các giao thức truyền thông và các giao diện lập trình ứng dụng cho phép truyền thông thời gian thực trên các kết nối peer-to-peer. Điều này cho phép các trình duyệt web không chỉ yêu cầu tài nguyên từ các máy chủ mà còn truyền thông tin thời gian thực với trình duyệt khác. Về bản chất, WebRTC là tập hợp các tiêu chuẩn và giao thức cho phép các trình duyệt Web thực hiện trực tiếp các tính năng truyền thông đa phương tiện thời gian thực như gọi điện, truyền hình, truyền dữ liệu, gửi tin nhắn bằng các APIs JavaScripts

### 1.1.1. Quá trình phát triển

WebRTC được bắt đầu từ Google nhằm xây dựng một chuẩn dựa trên máy phương tiện thời gian thực (Real time Media Engine) trong tất cả các trình duyệt. Ý tưởng này được đưa ra bởi nhóm phát triển Google Hangouts từ năm 2009. Sau khi mua lại công ty Global IP Solutions (GIPS) năm 2010 (Công ty hỗ trợ những ứng dụng điện thoại trên nền PC cho các Công ty lớn như Nortel(Avaya), Webex(Cisco), Yahoo, IBM…) và Công ty On2 năm 2011 (Công ty tạo ra chuẩn video codec VP8), Google đã phát hành WebRTC như là dự án mã nguồn mở dựa trên các công nghệ đạt được của GIPS và On2, chứa những thành phần nền tảng cho việc truyền thông chất lượng cao trên môi trường Web. Những thành phần này khi được thực hiện trong trình duyệt, có thể được truy cập qua các JavaScripts APIs, cho phép những nhà lập trình xây dựng những ứng dụng web đa phương tiện. Những công ty hỗ trợ phát triển WebRTC tích cực nhất gồm có Google, Mozilla, Opera. Sự phát triển của WebRTC qua các năm gần đây trên các trình duyệt thể hiện ở các mốc thời gian:

* 27/10/2011: Bản dự thảo WebRTC đầu tiên được W3C công bố.
* Tháng 11/2011, WebRTC được hỗ trợ một phần trên Chrome 23 (chưa hỗ trợ Data Channel API)
* Tháng 1/2013: WebRTC được hỗ trợ một phần trên Firefox 20 (hỗ trợ API GetUserMedia - API cho phép truy cập media trên máy).
* Tháng 6/2013: Firefox 22 phát hành, hỗ trợ khả năng tạo cuộc gọi video cũng như sử dụng Data Channel API.
* Tháng 7/2013: Phiên bản beta của Chrome 29 trên Android hỗ trợ WebRTC.
* Tháng 8/2013: Chrome 29 trên Android hỗ trợ đầy đủ WebRTC.
* Tháng 10/2013: Phiên bản beta Opera 18 giới thiệu hỗ trợ WebRTC.
* Tháng 3/2014: Phiên bản Opera 20 cho Android hỗ trợ WebRTC.
* 10/02/2015: WebRTC 1.0 working draft chính thức được công bố, đến nay đã được hỗ trợ bởi các trình duyệt Chrome (version 23 trở lên), Firefox (version 22 trở lên), Opera (version 18 trở lên) và được hỗ trợ trình duyệt trên 12 nền tảng Android (Chrome 29 trở lên, Firefox 24 trở lên, Opera Mobile 12 trở lên, Google Chrome OS).

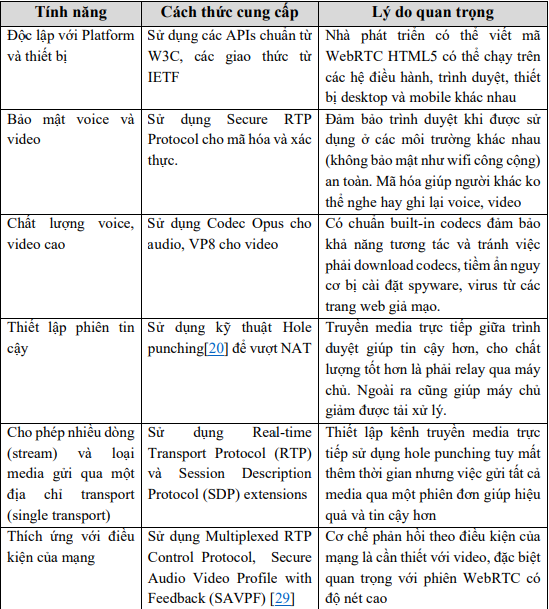
Tuy chưa được Microsoft, Apple tuyên bố hỗ trợ nhưng WebRTC vẫn tiếp tục được nghiên cứu mở rộng và hoàn thiện, bản cập nhật mới nhất được thực hiện vào 16/09/2016. WebRTC được phát triển dưới sự phối hợp chặt chẽ của tổ chức W3C và Internet Engineering Task Force – Lực lượng quản lý kỹ thuật mạng Internet (IETF). Tổ chức W3C, chủ yếu là nhóm Web Real-Time Communications Working Group, có nhiệm vụ định nghĩa các APIs phía client (client-side) để cho phép truyền thông thời gian thực trên trình duyệt Web. Những APIs giúp xây dựng ứng dụng chạy trong trình duyệt, không yêu cầu thêm download hay cài đặt plugin, cho phép truyền thông giữa các bên sử dụng audio, video theo thời gian thực không qua các máy chủ trung gian (trừ một số trường hợp cần thiết khi vượt NAT, tường lửa. Tổ chức IETF, chủ yếu là nhóm RTC in WEB-Browser Working Group, có nhiệm vụ định nghĩa các giao thức, định dạng dữ liệu, bảo mật … sử dụng trong WebRTC để thiết lập, điều khiển, quản lý việc truyền thông giữa trình duyệt.

Trước khi WebRTC xuất hiện, khi muốn xây dựng một ứng dụng web đa phương tiện đa nền tảng, người ta thường sử dụng Flash, Java Applet và tích hợp plugins các nhà cung cấp thứ ba để thực hiện. Giải pháp như vậy được coi là “nặng” và khó triển khai cũng như hỗ trợ về sau. Điều này thúc giục việc nghiên cứu giải pháp đơn giản, hiệu quả hơn cho các ứng dụng đa phương tiện, đặc biệt trên cơ sở người dùng hiện nay có thể truy cập được Internet mọi lúc mọi nơi. WebRTC ra đời để giải quyết vấn đề này, khi nó được tích hợp với các Voice Engine, Video Engine tốt nhất và được triển khai trên hàng triệu thiết bị đầu cuối hàng năm.

Những lợi ích của WebRTC:

* Giảm giá thành: chi phí triển khai và hỗ trợ IT thấp vì không cần cài đặt phần mềm client đặc biệt nào phía client.
* Không Plugins: trước đây phải sử dụng Flash, Java Applets và các giải pháp khác để xây dựng ứng dụng web tương tác đa phương tiện, phải download và cài đặt các plugin của bên thứ ba để có thể sử dụng nội dung đa phương tiện, ngoài ra còn phải lưu ý đến những giải pháp/plugin cho các hệ điều hành và nền tảng (platform) khác nhau. Với WebRTC thì không cần quan tâm đến vấn đề này nữa.
* Truyền thông P2P: trong đa phần các trường hợp, truyền thông được thiết lập trực tiếp giữa trình duyệt, không cần có những điểm trung gian.
* Dễ sử dụng: có thể dễ dàng tích hợp tính năng WebRTC trong dịch vụ web/trang web bằng cách sử dụng JavaScript APIs, những framework đã có sẵn. 13
* Một giải pháp cho mọi nền tảng: không cần phát triển những phiên bản dịch vụ web cho những nền tảng khác nhau (Windows, Android, IOS…)
* Mã mở và miễn phí: WebRTC được Google đưa thành dự án mã nguồn mở, và được hỗ trợ bởi những công ty quốc tế như Mozilla, Google và Opera, thêm cộng đồng trên thế giới có thể phát hiện những lỗi mới và giải quyết nhanh chóng hoàn toàn miễn phí [3]
* Built-in security: WebRTC quy định mọi dữ liệu truyền P2P đều được bảo mật và mã hóa.

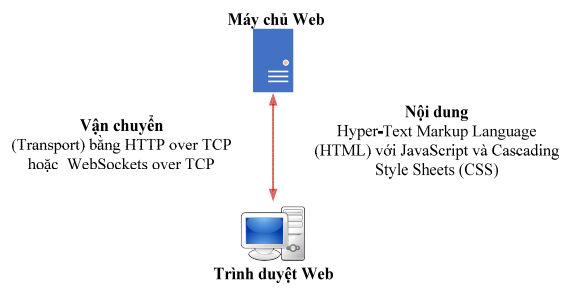
Một số tính năng mới quan trọng được lược tả ở bảng sau:



Bảng 1.1: Những tính năng mới của WebRTC

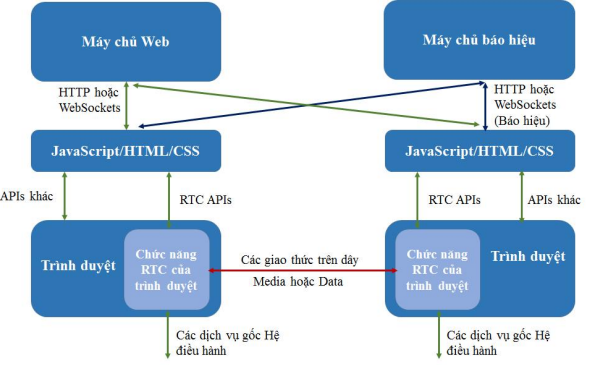
### 1.1.2. Kiến trúc WebRTC

Kiến trúc web cổ điển dựa trên mô hình client-server, trong đó trình duyệt gửi yêu cầu HTTP đến máy chủ để lấy nội dung, máy chủ trả lời, gửi nội dung về cho trình duyệt dưới dạng HTML, thường kèm theo JavaScript và Cascading Style Sheets (CSS). Trong trường hợp đơn giản thì khi máy chủ web trả lời yêu cầu từ client bằng thông tin text, hình ảnh hay thông tin khác như client mong muốn. Trong trường hợp phức tạp hơn, máy chủ gửi JavaScript để chạy ở phía trình duyệt, tương tác với với trình duyệt qua các JavaScript APIs chuẩn và tương tác với người dùng qua thao tác lựa chọn, click…trên giao diện người dùng. Trình duyệt trao đổi thông tin với máy chủ bằng giao thức HTTP trên TCP hoặc WebSockets trên TCP



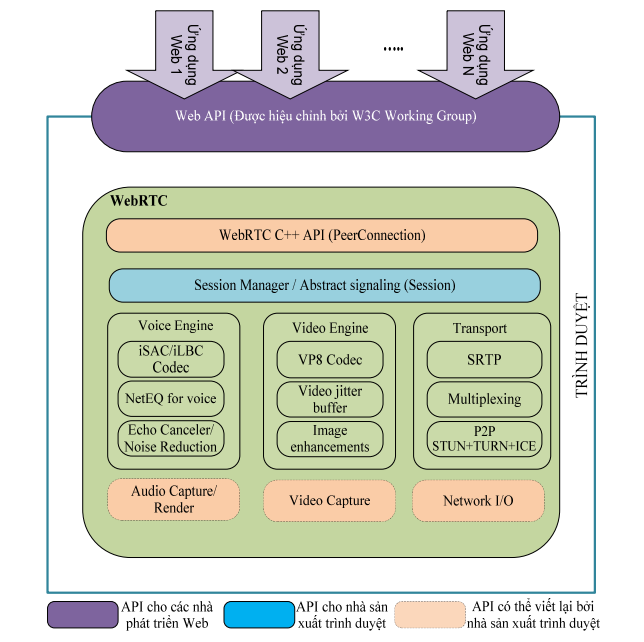
Hình 2.1: Kiến trúc ứng dụng Web cổ điển

WebRTC mở rộng ngữ nghĩa client-server bởi mô hình truyền thông Peer-to-Peer giữa các trình duyệt, thêm máy chủ báo hiệu và thành phần chức năng truyền thông thời gian thực (Real Time Communication hay RTC) của trình duyệt. Ứng dụng với WebRTC (thường viết bằng HTML5 và JavaScript) tương tác với trình duyệt qua những WebRTC APIs đang được chuẩn hóa, cho phép nó khai thác hợp lý và điều khiển chức năng thời gian thực của trình duyệt. Ứng dụng web với WebRTC cũng tương tác với trình duyệt sử dụng cả những APIs chuẩn hóa khác một cách chủ động (như truy vấn khả năng trình duyệt) hoặc bị động (như tiếp nhận thông báo khởi tạo bởi trình duyệt). Vì thế, WebRTC APIs phải cung cấp tập phong phú chức năng, như chức năng quản lý kết nối (connection management), thống nhất khả năng encoding/decoding, chức năng điều khiển media (media control), hỗ trợ vượt NAT và tường lửa…



Hình 2.2: Truyền thông thời gian thực trong trình duyệt (nguồn [1])

Hình 2.2 cho thấy mô hình trình duyệt và vai trò của các chức năng truyền thông thời gian thực. Khối màu sáng là chức năng truyền thông thời gian thực (Real Time Communication – RTC) của trình duyệt. Do tính chất riêng và yêu cầu của truyền thông thời gian thực nên việc chuẩn hóa khối này là không đơn giản, hiện tại vẫn đang trong quá trình bàn thảo. Các chức năng RTC tương tác với các ứng dụng web sử dụng các APIs chuẩn. Nó giao tiếp với các hệ điều hành bằng cách sử dụng trình duyệt. Khía cạnh mới của WebRTC là sự tương tác xảy ra từ trình duyệt đến trình duyệt, gọi là một kết nối Peer-to-Peer (P2P Connection) khi chức năng RTC trong một trình duyệt giao tiếp với chức năng RTC trong một trình duyệt khác tiếp sử dụng giao thức chuẩn trên dây (on-the-wire) hoặc giao tiếp với ứng dụng VoIP, ứng dụng Video. Trong khi lưu lượng web sử dụng giao thức TCP để vận chuyển, giao thức trên dây giữa các trình duyệt có thể sử dụng giao thức vận chuyển khác như UDP. Khía cạnh mới như nêu ở trên là cần thêm máy chủ báo hiệu (Signaling Server), là máy chủ cung cấp kênh truyền báo hiệu giữa trình duyệt và đầu kia của kết nối Peer. Phần báo hiệu trong WebRTC sẽ được trình bày chi tiết tại Chương III của Luận văn. Kiến trúc WebRTC bao gồm nhiều chuẩn khác nhau, chứa đựng cả ứng dụng và APIs trình duyệt, cũng như yêu cầu nhiều giao thức và định dạng dữ liệu để nó hoạt động. Trong WebRTC thì trình duyệt có khả năng truy cập vào phần cứng hệ thống tầng dưới để lấy audio, video đơn giản qua các APIs. Các dòng audio, video được xử lý để gia tăng chất lượng, tính đồng bộ, và “output bitrate” được điều chỉnh cho phù hợp với sự tăng giảm của băng thông, độ trễ giữa các client. Ở đầu xa, quá trình xử lý diễn ra ngược lại, client phải giải mã dòng media thời gian thực, có khả năng điều chỉnh jiter và độ trễ mạng. Dù việc lấy và xử lý audio và video là vấn đề phức tạp, nhưng WebRTC đã mang đến những “engine” audio, video đầy đủ tính năng để thực hiện.



Hình 2.3: Kiến trúc tổng thể WebRTC

Trong kiến trúc WebRTC có 3 lớp API:

* APIs cho nhà lập trình web: lớp này chứa tất cả các APIs mà nhà lập trình web cần, bao gồm các đối tượng chính là RTCPeerConnection, RTCDataChannel, MediaStream (chi tiết mô tả ở mục 1.1.3.Các APIs trong WebRTC).
* APIs cho nhà phát triển trình duyệt sử dụng.
* Overridable API: nhà phát triển trình duyệt có thể thay đổi, phát triển APIs của riêng mình.

Trong hình 2.3, thành phần video engine là framework xử lý chuỗi video từ camera đến mạng và từ mạng ra màn hình. Trong đó, video codec sử dụng VP8 (một dạng nén video mở, miễn phí sở hữu bởi Google và tạo ra bởi On2 Technologies) và VP9, hỗ trợ tính năng Video jitter buffer để giúp ẩn đi những ảnh hưởng của jitter và việc mất gói trong chất lượng video tổng thể, hỗ trợ nâng cao chất lượng ảnh như khử nhiễu ảnh được chụp từ webcam.

Thành phần audio engine là framework xử lý chuỗi audio từ card âm thanh đến mạng. Thành phần này sử dụng codec iSAC (internet Speech Audio Codec) /iLBC (Internet Low Bitrate Codec)/Opus .Nó sử dụng bộ đệm jitter động, thuật toán giấu lỗi để ẩn những ảnh hưởng của jitter mạng và mất gói tin, giúp giảm độ trễ tối đa mà vẫn giữ được chất lượng voice cao nhất. Trong framework sử dụng Acoustic Echo Canceler, phần mềm dựa trên thành phần xử lý tín hiệu giúp loại bỏ âm vọng trong thời gian thực và sử dụng Noise Reduction, phần mềm dựa trên thành phần xử lý tín hiệu giúp loại bỏ những tiếng ồn nền thường gắn với VoIP (hiss, fan noise).

Trong kiến trúc này, thành phần vận chuyển/quản lý phiên rất quan trọng, cho phép thiết lập và quản lý kết nối P2P qua các loại mạng khác nhau. Các nhiệm vụ, giao thức trong này như SRTP, STUN/TURN/ICE, quản lý phiên sẽ được nêu chi tiết ở mục 1.1.4 . Các tầng giao thức trong WebRTC.

### 1.1.3. Các APIs trong WebRTC

WebRTC bao gồm các APIs, các giao thức liên quan và làm việc với nhau để hỗ trợ việc trao đổi dữ liệu đa phương tiện giữa các trình duyệt. Về cơ bản, ứng dụng WebRTC thực hiện các công việc chính bao gồm:

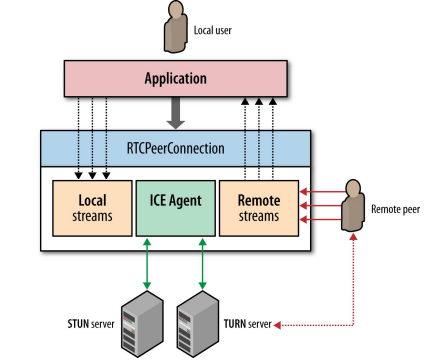
* Lấy dữ liệu audio, video hoặc dữ liệu khác trên máy.
* Lấy thông tin mạng như địa chỉ IP, port và trao đổi thông tin này với WebRTC client (gọi là Peer) để bắt đầu thiết lập kết nối, kể cả qua NATs và Firewall.
* Điều phối giao tiếp báo hiệu để báo cáo lỗi, khởi tạo hoặc đóng phiên kết nối.
* Trao đổi thông tin về khả năng hỗ trợ media của từng Peers như độ phân giải, codecs.
* Cuối cùng là streaming audio, video hoặc dữ liệu khác giữa hai Peer.

Để làm được các điều trên, WebRTC đang trong quá trình chuẩn hóa và sử dụng các APIs quanh ba khái niệm chính:

* RTCPeerConnection: thiết lập kết nối cho cuộc gọi audio/video/data, khả năng mã hóa và quản lý băng thông.
* MediaStream: truy cập vào dòng media, như camera hay microphone người dùng.
* RTCDataChannel: giao tiếp peer-to-peer cho các dữ liệu non-media.

#### 1.1.3.1 RTCPeerConnection API

Có rất nhiều giao thức tham gia vào quá trình thiết lập, duy trì kết nối P2P, nhưng APIs trong WebRTC được thiết kế khá trực quan. Giao diện RTCPeerConnection có nhiệm vụ quản lý trọn vẹn vòng đời của mỗi kết nối P2P.



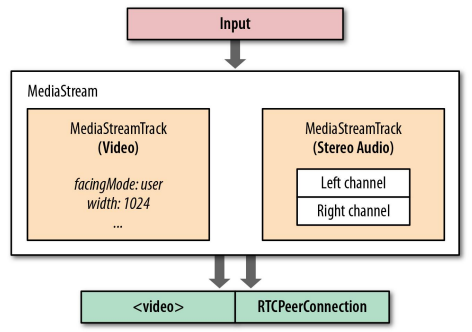
Hình 2.4: RTCPeerConnection API

Những nhiệm vụ chính của RTCPeerConnection là:

* Điều khiển toàn bộ quá trình Interactive Connectivity Establishment (ICE) [13] để vượt NAT.
* Gửi tự động bản tin STUN [14] keepalives giữa các peers.
* Giám sát dòng dữ liệu local và dòng dữ liệu ở xa (remote stream).
* Tự động kích hoạt (triggers) quá trình thương lượng lại dòng dữ liệu theo yêu cầu.
* Cung cấp các APIs để khởi tạo những thông tin offer/anwser cần trao đổi trong quá trình thiết lập kết nối, hoặc truy vấn trạng thái kết nối hiện tại. Ngắn gọn lại thì RTCPeerConnection đóng gói tất cả việc tạo, quản lý, trạng thái của kết nối trong một giao diện.

#### 1.1.3.2 MediaStream

Đặc tả “Media Capture and Stream” theo W3C định nghĩa là một tập các APIs JavaScript mới giúp ứng dụng có thể yêu cầu các dòng audio, video từ các nền tảng phía dưới, cũng như tập các APIs để thao tác, xử lý các dòng media đó. Đối tượng MediaStream là giao diện chính để thực hiện các chức năng này.



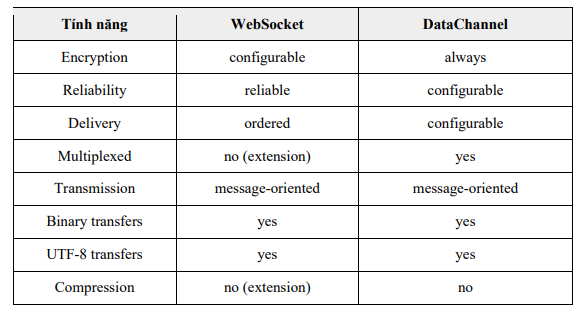
Hình 2.5: MediaStream mang một hoặc nhiều tracks đồng bộ

Mỗi đối tượng MediaStream chứa một hoặc nhiều những track riêng biệt (MediaStreamTrack). Mỗi MediaStreamTrack có thể chứa nhiều kênh (ví dụ như kênh trái hoặc phải của audio). Những kênh này là đơn vị nhỏ nhất mà được định nghĩa bởi MediaStream API. Các Track trong đối tượng MediaStream được đồng bộ với nhau. Nguồn vào có thể là thiết bị vật lý như microphone, webcam hoặc các file từ máy người dùng hoặc từ mạng. Đầu ra của MediaStream có thể là một hoặc nhiều đích: thành phần video hay audio trên local, peer xa, mã JavaScript để xử lý sau. Đối tượng MediaStream thể hiện dòng media thời gian thực và cho phép các đoạn mã ứng dụng thu thập dữ liệu, thao tác với các track riêng biệt, xác định đầu ra. Tất cả quá trình xử lý audio, video như hủy tiếng ồn, hiệu chỉnh (equalization), nâng cao chất lượng ảnh…được xử lý tự động bởi các engine. Tuy nhiên, tính năng thu thập dòng media bị ràng buộc với khả năng của nguồn vào: microphone chỉ có thể xuất ra dòng audio, các webcam có thể xuất ra các video độ phân giải khác nhau theo cấu hình. Trong ứng dụng, vấn đề này được xử lý bằng cách gọi API getUserMedia(), API này cho phép xác định danh sách những ràng buộc bắt buộc hay không để phù hợp với ứng dụng. Nói chung, getUserMedia() là một API đơn giản để lấy dòng audio và video từ platform, còn media được tự động tối ưu, encoded, decodeed bởi WebRTC audio, video engines và sau đó hướng đến các đầu nguồn ra tùy theo ứng dụng

#### 1.1.3.3 RTCDataChannel

Tương tự như audio và video, WebRTC hỗ trợ truyền thông thời gian thực với các loại dữ liệu khác. RTCDataChannel API cho phép trao đổi dữ liệu tùy ý peer-to-peer với độ trễ thấp và thông lượng cao. Vì vậy, nó được sử dụng trong những trường hợp như: trong ứng dụng game, ứng dụng remote desktop, chat text thời gian thực, truyền file. Ở lớp dưới, DataChannel sử dụng Stream Control Tranmission Protocol - SCTP, giao thức cho phép cấu hình việc gửi tin cậy (tương tự như TCP) hay không tin cậy (tương tự UDP), có thứ tự hay không có thứ tự phù hợp với các yêu cầu ứng dụng khác nhau. Ngoài ra, DataChannel hỗ trợ tập các kiểu dữ liệu linh động, các APIs được thiết kế để giống WebSocket, hỗ trợ strings cũng như các kiểu nhị phân trong JavaScript như Blob (binary large object), ArrayBuffer, ArrayBufferView, là những kiểu dữ liệu hữu ích cho việc truyền file và chơi game nhiều người. Tuy nhiên Data Channel có bổ sung một số tính năng so với WebSocket, và có những điểm khác chính là:

* DataChannel có thể khởi tạo session từ cả 2 đầu của kết nối, hàm callback ondatachannel sẽ được gọi khi có một phiên RTCDataChannel mới được thiết lập.
* WebSocket chạy trên giao thức vận chuyển tin cậy TCP, còn DataChannel chạy trên ba giao thức UDP, DTLS (Datagram Transport Layer Security), SCTP

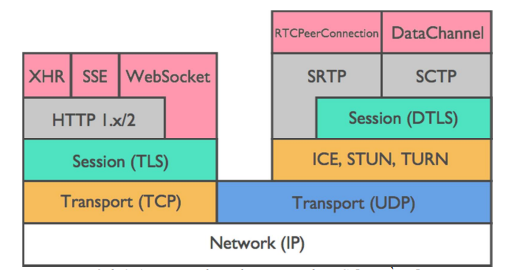


Bảng 2.2: So sánh giữa WebSocket và DataChannel

Dòng dữ liệu gửi P2P được handle bằng cách sử dụng SCTP trên Datagram Transport Layer Security (DTLS) [16] trên ICE/UDP, cung cấp giải pháp vượt NAT cùng với tính chất an toàn, xác thực nguồn gốc và toàn vẹn dữ liệu gửi. Dịch vụ vận chuyển dữ liệu này hoạt động song song với vận chuyển media bằng giao thức SRTP (Secure Real-time Transport Protocol).

### 1.1.4. Các tầng giao thức trong WebRTC

Như đã nêu ở mục 1.1.1, các giao thức phục vụ cho các ứng dụng thời gian thực trên trình duyệt được IETF (nhóm RTCWEB Working Group) phụ trách đưa ra các đề xuất chuẩn hóa. Do đặc điểm yêu cầu thời gian thực cao hơn tính tin cậy, giao thức UDP được lựa chọn sử dụng trong WebRTC là giao thức vận chuyển. Tuy nhiên, để thỏa mãn tất cả yêu cầu của WebRTC, trình duyệt phải hỗ trợ các giao thức và dịch vụ khác ở lớp trên nữa. Về cơ bản, các giao thức chính sử dụng trong WebRTC thể hiện ở hình 2.6 dưới đây:



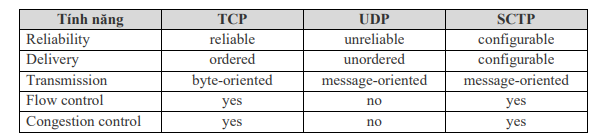
Hình 2.6: Protocol stack trong WebRTC

SRTP

Giao thức quan trọng nhất mà WebRTC sử dụng là Secure Real-time Transport Protocol, hay SRTP . SRTP được sử dụng để mã hóa và chuyển các gói tin media giữa các WebRTC client. Sau khi thiết lập thành công PeerConnection, kết nối SRTP sẽ được thiết lập giữa các trình duyệt hoặc trình duyệt và máy chủ. Với dữ liệu non-audio hay video, SRTP không được sử dụng, thay vào đó là SCTP.

SCTP

WebRTC sử dụng SCTP - Stream Control Transmission Protocol để truyền các dữ liệu non-media giữa các Peer. Giao thức SCTP là giao thức vận chuyển, tương tự như TCP và UDP, có thể chạy trực tiếp trên giao thức IP. Tuy nhiên trong WebRTC, SCTP chạy trên DTLS trên UDP. SCTP được lựa chọn do có những tính năng tốt nhất của TCP và UDP như: message-oriented transmission, khả năng cấu hình tùy biến tính tin cậy và thứ tự gói tin, có cơ chế quản lý lưu lượng và chống nghẽn.



Bảng 2.3: So sánh tính năng chính các giao thức TCP, UDP và SCTP

SDP

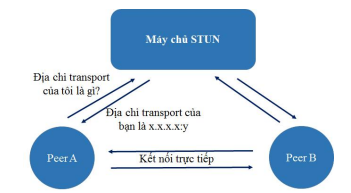
WebRTC sử dụng Session Description Protocol, SDP, được encode trong đối tượng RTCSessionDescription, để mô tả đặc tính media của hai đầu trong kết nối P2P như loại media đề truyền/nhận (audio, video, application data), network transports, loại codecs sử dụng và cấu hình, thông tin băng thông, và các thông tin metadata khác. Thông điệp SDP được trao đổi qua máy chủ báo hiệu hay còn gọi là được trao đổi qua kênh báo hiệu. Máy chủ báo hiệu có trách nhiệm gửi và nhận tất cả thông điệp đến tất cả các peers mà mong muốn kết nối đến peer khác. Mặc dù SDP là định dạng dữ liệu dùng để trao đổi, thống nhất thông số giữa kết nối Peer-to-Peer, nhưng do WebRTC không ràng buộc cho các SDP “offer” và “answer” giao tiếp như nào, nên nó không được thể hiện ở hình 2.6 ở trên. Tuy nhiên, mô hình offer/answer được thiết kế tuân thủ theo RFC3264

DTLS

Datagram Transport Layer Security- DTLS dựa trên giao thức TLS, cung cấp tính bảo mật và toàn vẹn dữ liệu truyền giữa các ứng dụng tương tự TLS .Tuy nhiên, WebRTC sử dụng DTLS do nó chạy trên giao thức UDP thích hợp với việc vượt NAT cho các ứng dụng P2P. Tất cả các dữ liệu truyền P2P đều được bảo mật sử dụng DTLS. Cụ thể hơn, DTLS được sử dụng trong việc thống nhất khóa bảo mật cho việc mã hóa dữ liệu media và trong việc bảo mật sự vận chuyển dữ liệu ứng dụng. Mặc dù cung cấp tính mã hóa, tính toàn vẹn, nhưng phần xác thực trong WebRTC được gán cho ứng dụng

STUN

Session Traversal Utilities for NAT, STUN : là giao thức giúp cho việc vượt NAT trong quá trình thiết lập kết nối. Trong WebRTC, một STUN client sẽ được xây dựng trong User Agent của trình duyệt để kết nối đến STUN server ngoài Internet. STUN server thực thi nhiệm vụ khá đơn giản, kiểm tra thông tin địa chỉ IP, port của request đến từ ứng dụng sau NAT, sau đó trả thông tin đó về dưới dạng response, nói cách khác là STUN giúp ứng dụng biết địa chỉ IP, cổng của nó sử dụng khi đi ra Internet. STUN có thể được vận chuyển trên UDP, TCP hoặc TLS



Hình 2.7: Mô hình hoạt động STUN

Trong đa số các trường hợp thì chỉ cần sử dụng STUN trong việc thiết lập kết nối P2P, trừ trường hợp một peer đứng sau symmetric NAT, một peer đứng sau Symmetric NAT hoặc port-restricted NAT. Trường hợp này quá trình hole punching sẽ không thành công, cần phải sử dụng đến TURN - Traversal Using Relays around NAT.

STUN

Traversal Using Relays around NAT, TURN, là một mở rộng (extension) của giao thức STUN, cung cấp media relay cho tình huống thực hiện hole punching không thành công. Trong WebRTC, User Agent của trình duyệt sẽ bao gồm một TURN client. TURN server được cung cấp trên Internet qua các nhà cung cấp dịch vụ, hoặc có thể cài đặt trong mạng doanh nghiệp. Giao thức UDP được sử dụng để giao tiếp giữa TURN client và TURN server qua NAT. Cổng UDP mặc định cho TURN là 3478. Trên thực tế TURN server thường là STUN server có bổ sung tính năng relay. TURN server thì có chức năng STUN, nhưng không phải mọi STUN server đều có chức năng TURN.

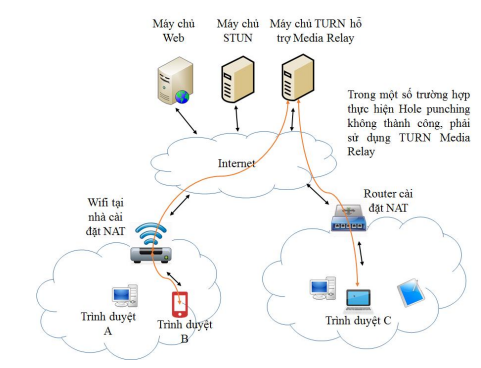
ICE

Interactive Communication Establishment, hay ICE là giao thức rất quan trọng trong WebRTC, có hai chức năng chính: cho phép WebRTC client trao đổi media qua thiết bị có thực hiện NAT và cung cấp sự xác minh việc chấp thuận giao tiếp giữa client, giúp gói tin media chỉ được gửi đến trình duyệt mà đang đợi dữ liệu. Một ứng dụng web độc hại có thể lừa trình duyệt gửi dữ liệu media đến một host trên Internet mà không phải là thành phần muốn giao tiếp. Kiểu tấn công này là tấn công từ chối dịch vụ DOS (Denial of Service), và ICE thành công trong việc ngăn ngừa điều này vì dữ liệu media không bao giờ được gửi trừ khi quá trình trao đổi ICE kết thúc thành công. ICE sử dụng kỹ thuật hole punching, kỹ thuật được tiên phong bởi các game thủ chơi online, những người cần trao đổi gói tin trực tiếp giữa các PCs cho dù giữa chúng có NAT. Hole punching là kỹ thuật cho phép “đục lỗ” qua thiết bị NAT để thiết lập kết nối trực tiếp giữa ứng dụng ngay cả khi cả hai đầu ứng dụng đều nằm sau NAT. Để hole punching thành công cần thỏa mãn một số các yêu cầu:

* Hai trình duyệt muốn thiết lập kết nối trực tiếp phải gửi gói hole punching cùng thời điểm. Nhờ vậy cả hai trình duyệt mới nhận ra session cần thiết lập và biết các địa chỉ để gửi gói tin. Gói tin hole punching là gói tin IP thông thường được gửi để kiểm tra có đến được địa chỉ đích (sau NATs) không.
* Hai trình duyệt phải biết càng nhiều địa chỉ IP mà có thể sử dụng để kết nối đến peer càng tốt. Các địa chỉ này các địa chỉ nội bộ (trong NAT), địa chỉ public (ngoài NAT) và địa chỉ relay.
* Cả hai trình duyệt phải kết nối đến được địa chỉ IP Public của media relay.
* Dòng đối xứng phải được sử dụng. Lưu lượng UDP phải xuất hiện để hoạt động tương tự cách của kết nối TCP.

Yêu cầu đầu tiên thỏa mãn khi sử dụng máy chủ web để điều phối quá trình hole punching, bởi máy chủ web biết có nhu cầu mong muốn thiết lập giữa hai trình duyệt, do đó nó đảm bảo việc hai trình duyệt bắt đầu quá trình hole punching cùng thời điểm ở mức tương đối. Yêu cầu thứ 2 được thỏa mãn bằng cách dùng STUN Server. Yêu cầu thứ 3 được thỏa mãn khi dùng TURN Server. Trình duyệt ưu tiên truy vấn đến STUN server trước để khởi động quá trình hole punching, sau đó mới truy vấn đến TURN server để lấy địa chỉ media relay. Địa chỉ media relay là địa chỉ public, giúp chuyển tiếp gói tin đến và đi từ trình duyệt thiết lập địa chỉ relay. Địa chỉ này sau đó sẽ được thêm vào danh sách địa chỉ ứng viên (Địa chỉ ứng viên gồm địa chỉ IP và cổng UDP). Yêu cầu 4 được đáp ứng bởi trình duyệt gửi media từ cùng 1 cổng UDP mà trình duyệt sử dụng để lắng nghe media đến. Điều này làm cho 2 phiên một chiều RTP trên UDP xuất hiện với NAT như là một phiên RTP hai chiều.

Trong đa số các trường hợp thì hole puching thành công, kết nối peer-to-peer được thiết lập, luồng media không relay qua máy chủ. Luồng media chỉ relay qua máy chủ khi mọi đường media đều thiết lập không thành công.



Hình 2.8: Luồng Media qua TURN server

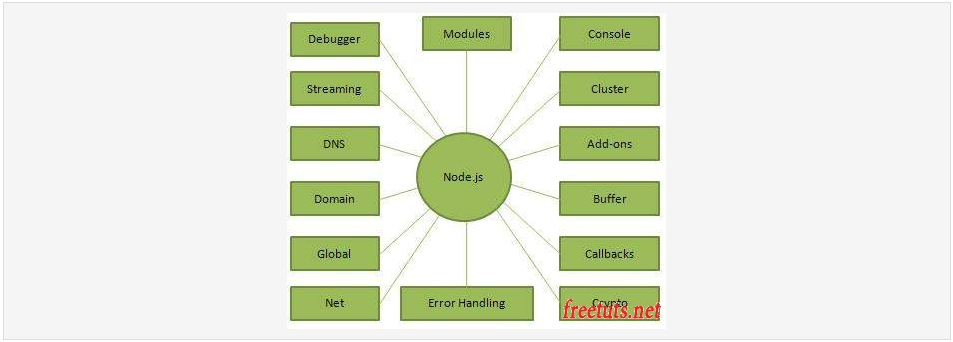
## 1.2. SƠ LƯỢC VỀ NODEJS

NodeJS là một mã nguồn được xây dựng dựa trên nền tảng Javascript V8 Engine, nó được sử dụng để xây dựng các ứng dụng web như các trang video clip, các forum và đặc biệt là trang mạng xã hội phạm vi hẹp. NodeJS là một mã nguồn mở được sử dụng rộng bởi hàng ngàn lập trình viên trên toàn thế giới.

NodeJS có thể chạy trên nhiều nền tảng hệ điều hành khác nhau từ WIndow cho tới Linux, OS X nên đó cũng là một lợi thế. NodeJS cung cấp các thư viện phong phú ở dạng Javascript Module khác nhau giúp đơn giản hóa việc lập trình và giảm thời gian ở mức thấp nhất.

### 1.2.1. Các đặc tính NodeJS

* **Không đồng bộ**: Tất cả các API của NodeJS đều không đồng bộ (none-blocking), nó chủ yếu dựa trên nền của NodeJS Server và chờ đợi Server trả dữ liệu về. Việc di chuyển máy chủ đến các API tiếp theo sau khi gọi và cơ chế thông báo các sự kiện của Node.js giúp máy chủ để có được một phản ứng từ các cuộc gọi API trước (Realtime).
* **Chạy rất nhanh**: NodeJ được xây dựng dựa vào nền tảng V8 Javascript Engine nên việc thực thi chương trình rất nhanh.
* **Đơn luồng nhưng khả năng mở rộng cao**: Node.js sử dụng một mô hình luồng duy nhất với sự kiện lặp. cơ chế tổ chức sự kiện giúp các máy chủ để đáp ứng một cách không ngăn chặn và làm cho máy chủ cao khả năng mở rộng như trái ngược với các máy chủ truyền thống mà tạo đề hạn chế để xử lý yêu cầu. Node.js sử dụng một chương trình đơn luồng và các chương trình tương tự có thể cung cấp dịch vụ cho một số lượng lớn hơn nhiều so với yêu cầu máy chủ truyền thống như Apache HTTP Server.
* **Không đệm**: NodeJS không đệm bất kì một dữ liệu nào và các ứng dụng này chủ yếu là đầu ra dữ liệu.
* **Có giấy phép**: NodeJS đã được cấp giấy phép bởi [MIT License](https://freetuts.net/nodejs-la-gi-584.html).



**sơ đồ về các thành phần quan trọng trong NodeJS**

### 1.2.2. Hai NodeJS framwork sử dụng phổ biến

#### 1.2.2.1. Express:

ExpressJS là một trong những framework phổ biến dùng để xây dựng API và Website phổ biến nhất của NodeJS. Nó được sử dụng rộng rãi đến mức hầu như mọi dự án Web nào đều bắt đầu bằng việc tích hợp Express. Có rất nhiều lý do để chọn ExpressJS:

* Có nhiều tính năng hỗ trợ tất cả những gì bạn cần trong việc xây dựng Web và API
* Quản lý các route dễ dàng
* Cung cấp một nền tảng phát triển cho các API
* Hỗ trợ nhiều thư viện và plugin
* Bảo mật và an toàn hơn so với việc code thuần
* Hỗ trợ cộng đồng tuyệt vời

#### 1.2.2.2. SocketIO

SocketIO là một web-socket framework có sẵn cho nhiều ngôn ngữ lập trình.

Trong NodeJS, SocketIO cho phép xây dựng một các ứng dụng realtime như chatbot, tickers, dashboard APIs, và nhiều ứng dụng khác. SocketIO có lợi ích hơn so với NodeJS thông thường.

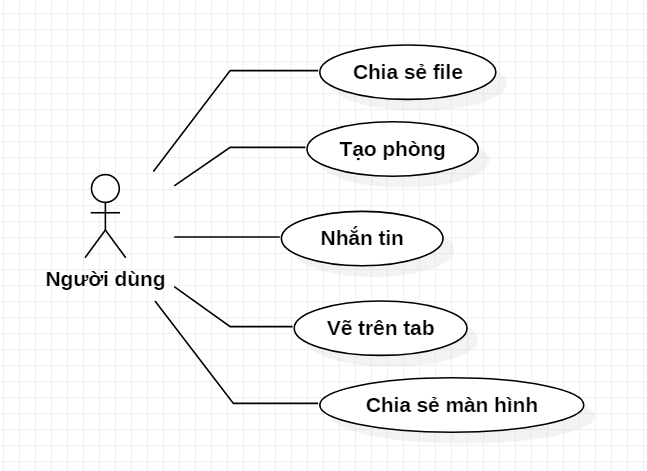
* Hỗ trợ route URL tùy chỉnh cho web socket
* Tích hợp dễ dàng hơn với Express JS
* Hỗ trợ clustering với Redis

#### 1.2.2.3. **Ngoài ra còn một số framework khác như**:

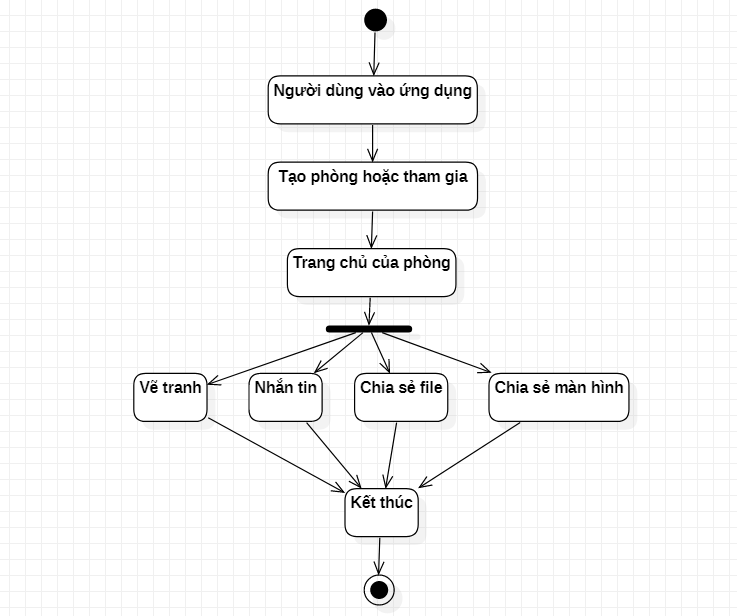
* BodyParser
* ails.JS
* Hapi.JS
* Total.JS
* LoopBack
* Meteor
* Restify
* Koa

# CHƯƠNG II: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

## 2.1. Biểu đồ UseCase

2.2. Đặc tả UseCase

## 2.3. Biểu đồ hoạt động



# CHƯƠNG III: XÂY DỰNG CHƯƠNG TRÌNH

## 3.1. NHỮNG KẾT QUẢ KHI NGHIÊN CỨU XONG

Sau một khoảng thời gian tìm hiểu đề tài “Lập trình bằng Nodejs cho phép triệu gọi các chương trình ở xa thực hiện tính toán đồng thời” em đã thực hiện được nội dung của đề tài theo yêu cầu đặt ra. Với mục đích là tìm hiểu về mạng Peer – to – Peer một kiến trúc ứng dụng phân tán phân vùng nhằm phân nhiệm vụ hoặc khối lượng công việc giữa các peer , ngôn ngữ lập trình Nodejs cũng như nắm bắt lại hệ thống kiến thức nền tảng của các môn học. Thông qua đó, nắm bắt được các kiến thức cơ bản cũng như xây dựng được chương trình demo.

### 3.1.1. Những kết quả đạt được

Trong quá trình nghiên cứu và tìm hiểu lý thuyết về môn lập trình mạng kết hợp với việc xây dựng ứng dụng, em đã thu được kết quả như sau:

- Hiểu và sử dụng được kiến trúc phân tán mạng Peer to Peer .

- Hiểu được định dạng dữ liệu và cơ chế làm việc với Nodejs.

- Hiểu được cơ cấu và cách hoạt động của WebRTC

- Xây dựng thành công ứng dụng demo.

- Hoàn thành báo cáo dưới sự chỉ dẫn của GVHD.

### 3.1.2. Những thiếu sót

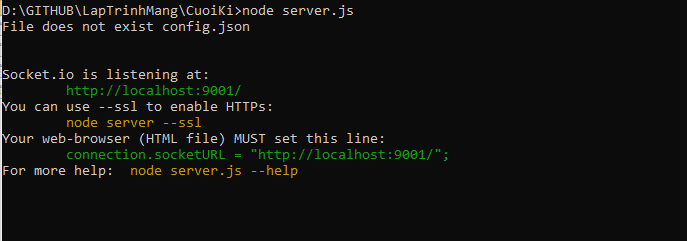
Do thời gian gấp rút cùng với kiến thức có hạn của bản thân nên nhóm em chưa thể tìm hiểu sâu hơn về các kiến thức liên quan tới môn lập trình mạng nói chung và phương tức lập trình đối tượng phân tán nói riêng dẫn tới việc ứng dụng demo vẫn còn nhiều hạn chế, chưa thực sự thể hiện được các kĩ thuật nâng cao. Chúng em sẽ cố gắng tìm hiểu sâu hơn trong thời gian sắp tới:

- Xây dựng ứng dụng demo hoàn thiện hơn.

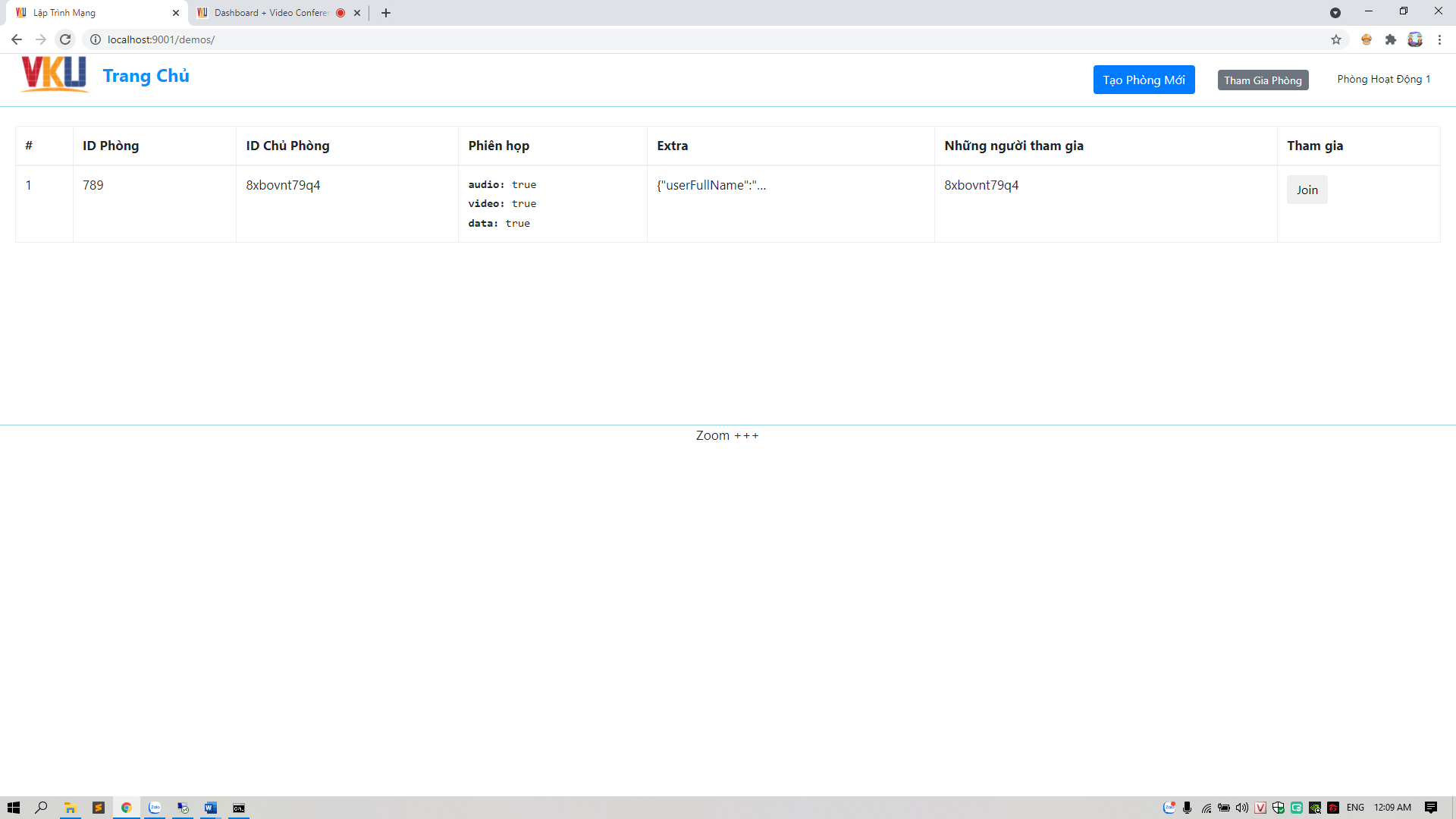
- Nghiên cứu sâu hơn các kĩ thuật và kiến thức nâng cao.

## 3.2. HÌNH ẢNH CHƯƠNG TRÌNH

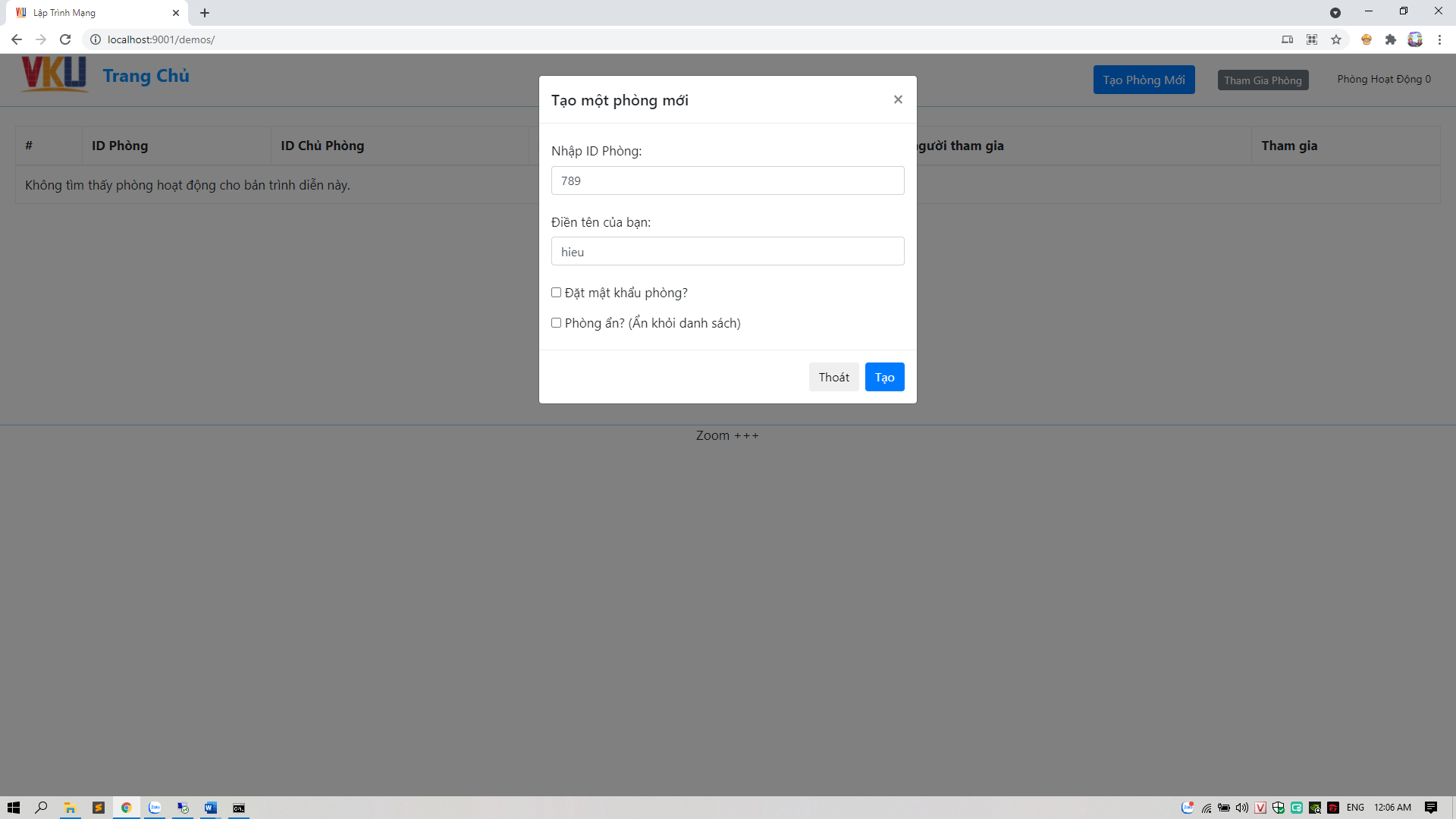
Trước khi sử dụng ứng dụng ta phải khởi động Server



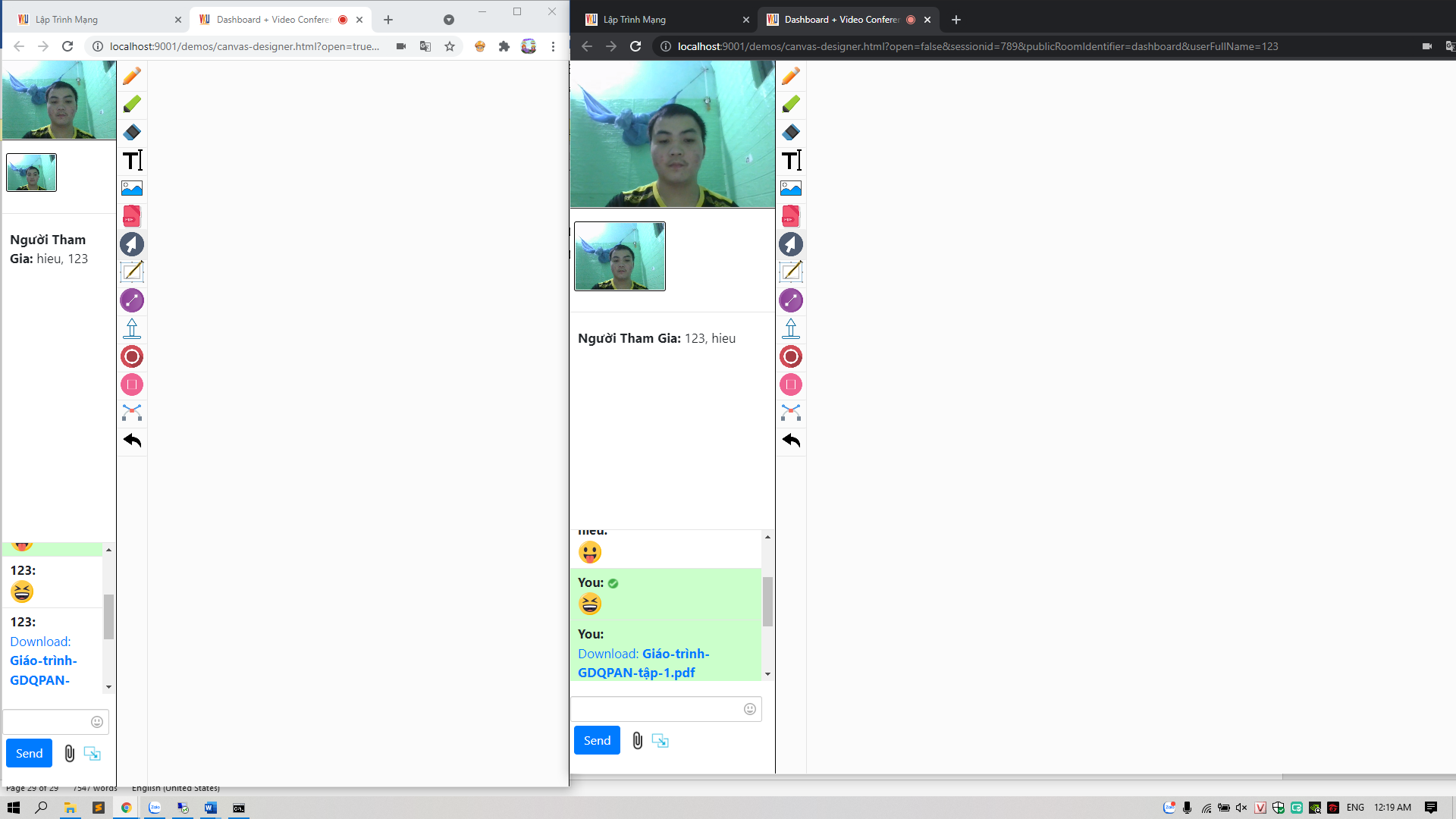
Sau khi khởi động Sever ta tiến hành khởi chạy cổng <http://localhost:9001/> đẻ khởi động ứng dụng . Tại cửa sổ này, chúng ta có thể thấy các chức năng căn bản như là: Tạo phòng mới , Tham gia phòng , Hiển thị các phòng , Người tham gia ….



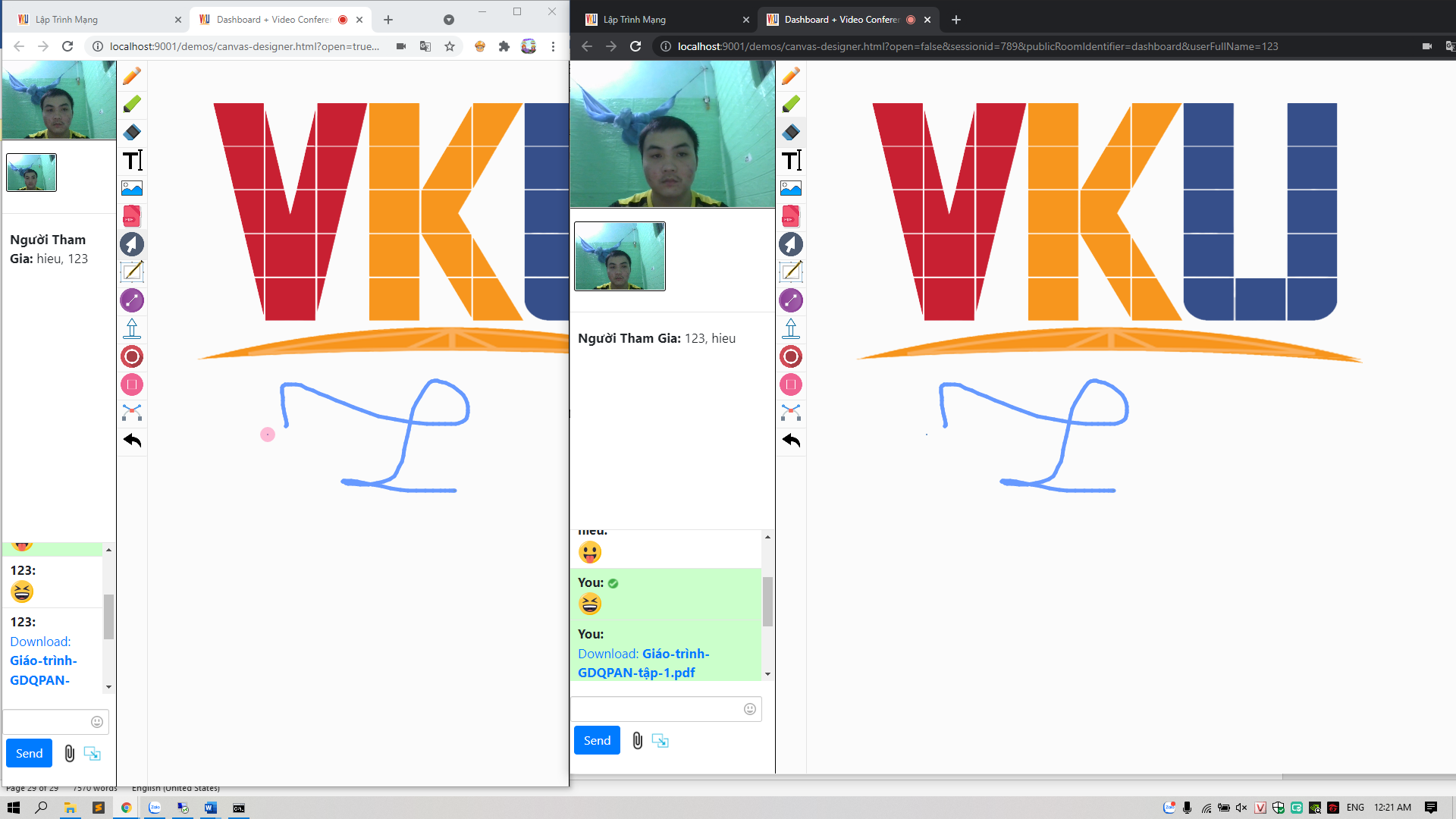
Ở chức năng tạo phòng mới chúng tao tạo ID của phòng điền tên của mình vào và chúng ta có thể thêm mật khẩu của phòng tạo và có thể ẩn phòng tạo đó đi nhằm tính bảo mật cho các cuộc gọi gặp gỡ hoặc có thể công khai tên phòng ở trang chủ:



Sau khi tạo xong chúng ta sẽ join vào phòng ở đây chúng ta có thể gặp mặt được nhiều người .Và chúng ta có thể trò chuyện với nhau , chat với nhau , có thể gửi file cho nhau , chia sẻ màn hình ….



Tiếp theo chúng ta có thể chia sẻ các ảnh hoặc cái file hoặc có thẻ sử dụng văn bản, hay vẽ ….



## 3.3. HƯỚNG PHÁT TRIỂN

- Xây dựng được giao diện thân thiện hơn với người dùng.

- Kiểm tra và sửa lỗi các trường hợp thiếu sót như bật tắt được camera,…..

- Tận dụng thế mạnh của các công cụ hỗ trợ để ứng dụng được nhanh nhạy, chính xác hơn.

- Xây dựng thêm nhiều chức năng để hệ thống đầy đủ và toàn vẹn hơn.

- Tìm hiểu xu hướng của người dùng để chỉnh sửa, bổ sung hệ thống phù hợp theo yêu cầu.

- Cập nhật tính năng khác để nâng cấp hệ thống của mình.

**TÀI LIỆU THAM KHẢO**

1.Alan B.Johnson, Daniel C.Burnett (2014), APIs and RTCWEB Protocols of the

HTML5 Real-Time Web, Digital Codex LLC

2. Salvatore Loreto, Simon Pietro Romano (2014), Real-time Communication with

WebRTC, O’Reilly, USA

3. Andrii Sergiienko (2014), WebRTC Blueprints, Packt Publishing Ltd, UK

4. Ilya Grigorik (2015), High Performance Browser Networking, O’Reilly Media.

5. Altanai (2014), WebRTC Intergrator’s Guide, Packt Publishing Ltd, UK

6. WebRTC for Enterprises

7. Tsahi Levent-Levi (2013), WebRTC for Business People: Unraveling the

challenges and opportunities of the WebRTC ecosystem

8. Dan Ristic (2015), Learning WebRTC, Packt Publishing Ltd, UK

9. Rob Manson (2013), Getting Started with WebRTC, Packt Publishing Ltd, UK

10. WebRTC Architecture, https://webrtc.org/architecture, Thời gian truy cập: 11-

09-2016

11. RFC 1631 - The IP Network Address Translator (NAT), 1994,

https://tools.ietf.org/html/rfc1631

12. RFC 6716 - Definition of the Opus Audio Codec, 2012

https://tools.ietf.org/html/rfc6716

13. RFC 5245, Interactive Connectivity Establishment (ICE): A Protocol for

Network Address Translator (NAT) Traversal for Offer/Answer Protocols, 2012,

https://tools.ietf.org/html/rfc5245

14. RFC 5389, Session Traversal Utilities for NAT (STUN), 2008,

https://tools.ietf.org/html/rfc5389

15. RFC 4960, Stream Control Tranmission Protocol (SCTP), 2007,

https://tools.ietf.org/html/rfc4960

16. RFC 4347, Datagram Transport Layer Security (DTLS), 2006

https://tools.ietf.org/html/rfc4347

17. RFC 3711, The Secure Real-time Transport Protocol (SRTP), 2004,

https://www.ietf.org/rfc/rfc3711.txt

18. RFC 4566, SDP: Session Description Protocol, 2006,

https://tools.ietf.org/html/rfc4566

19. RFC 5246, The Transport Layer Security (TLS) Protocol Version 1.2, 2008,

https://tools.ietf.org/html/rfc5246

20. RFC 5128, State of Peer-to-Peer (P2P) Communication across Network Address

Translators (NATs), 2008, https://tools.ietf.org/html/rfc5128

74

21. RFC 5766, Traversal Using Relays around NAT (TURN): Relay Extensions to

Session Traversal Utilities for NAT (STUN), 2010,

https://tools.ietf.org/html/rfc5766

22. EasyRTC website, https://easyrtc.com/docs/browser/easyrtc.php, Thời gian truy

cập: 11-09-2016

23. https://www.pkcsecurity.com/blog. Thời gian truy cập 11-10-2016

24. RFC 3264, An Offer/Answer Model with the Session Description Protocol

(SDP), 2002, https://tools.ietf.org/html/rfc3264

25. Javascript Session Establishment Protocol draft-ietf-rtcweb-jsep version 16, 20-

09-2016, https://tools.ietf.org/html/draft-ietf-rtcweb-jsep-16

26. https://en.wikipedia.org/wiki/WebRTC

27. https://webrtchacks.com/signalling-options-for-webrtc-applications/, thời gian

truy cập 10-2016

28. RFC 6749, The OAuth 2.0 Authorization Framework, 2012,

https://tools.ietf.org/html/rfc6749

29. RFC 5762, Multiplexing RTP Data and Control Packets on a Single Port, 2010,

https://tools.ietf.org/html/rfc5761

30. RFC 6120, Extensible Messaging and Presence Protocol (XMPP): Core, 2011,

https://tools.ietf.org/html/rfc6120