ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

NGUYỄN VIẾT THẮNG

NGHIÊN CỨU ỨNG DỤNG CÔNG NGHỆ WEBRTC CHO GIẢI PHÁP CỘNG TÁC VÀ CHIA SẺ DỮ LIỆU ĐA PHƯƠNG TIỆN TẠI TRUNG TÂM MVAS-TCT VIỄN THÔNG MOBIFONE

LUẬN VĂN THẠC SĨ CÔNG NGHỆ THÔNG TIN

Hà nội - 2016

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

NGUYỄN VIẾT THẮNG

NGHIÊN CỨU ỨNG DỤNG CÔNG NGHỆ WEBRTC CHO GIẢI PHÁP CỘNG TÁC VÀ CHIA SẺ DỮ LIỆU ĐA PHƯƠNG TIỆN TẠI TRUNG TÂM MVAS-TCT VIỄN THÔNG MOBIFONE

Ngành : Công nghệ thông tin
Chuyên ngành : Truyền dữ liệu & Mạng máy tính
Mã số :

LUẬN VĂN THẠC SĨ CÔNG NGHỆ THÔNG TIN
GIÁO VIÊN HƯỚNG DẪN KHOA HỌC: TS.HOÀNG XUÂN TÙNG

Hà nội - 2016

LỜI CẢM ƠN

Luận văn Thạc sĩ này được thực hiện tại Đại học Công nghệ - Đại học Quốc gia Hà Nội dưới sự hướng dẫn của TS. Hoàng Xuân Tùng. Xin được gửi lời cảm ơn sâu sắc đến thầy Hoàng Xuân Tùng về những ý kiến quý báu liên quan đến các định hướng khoa học, liên tục quan tâm, tạo điều kiện thuận lợi cho tôi trong suốt quá trình nghiên cứu hoàn thành luận văn này. Tôi xin được gửi lời cảm ơn đến các thầy, cô trong Bộ môn Truyền dữ liệu và Mạng máy tính cũng như Khoa Công nghệ Thông tin đã mang lại cho tôi những kiến thức vô cùng quý giá và bổ ích trong quá trình theo học tại trường.

Tôi xin gửi lời cảm ơn tới các đồng chí lãnh đạo đơn vị nơi tôi công tác đã tạo điều kiện và thời gian để tôi có thể hoàn thành chương trình học của mình. Bên cạnh đó tôi xin gửi lời cám ơn tới các đồng nghiệp trong Mobifone đã tạo điều kiện và giúp đỡ tôi hoàn thành khóa luận này một cách tốt nhất.

Cuối cùng tôi cũng xin chân thành cảm ơn đến các học viên cao học khóa K19, K20, K21 đã giúp đỡ tôi trong suốt thời gian học tập.

Do thời gian và kiến thức có hạn nên luận văn chắc không tránh khỏi những thiếu sót nhất định. Tôi rất mong nhận được những sự góp ý quý báu của thầy cô và các bạn.

Hà Nội, ngày tháng năm 2016

Nguyễn Viết Thắng

LỜI CAM ĐOAN

Tôi Nguyễn Viết Thắng xin cam đoan nội dung trong luận văn này là công trình nghiên cứu và sáng tạo do chính tôi thực hiện dưới sự hướng dẫn của TS. Hoàng Xuân Tùng. Số liệu, kết quả trình bày trong luận văn là hoàn toàn trung thực và chưa công bố trong bất cứ công trình khoa học nào trước đây. Nếu hình ảnh được lấy từ nguồn bên ngoài, tôi đều có trích dẫn nguồn rõ ràng và đầy đủ.

Hà Nội, ngày ... tháng ... năm 2016

Học viên

Nguyễn Viết Thắng

MỤC LỤC

DANH MỤC CÁC TỪ VIẾT TẮT

TT	Từ viết tắt	Cụm từ tiếng anh
1.	HTTP	Hypertext Transfer Protocol
2.	URI	Uniform Resource Identifier
3.	URL	Uniform Resource Locator
4.	CSS	Cascading Style Sheets
5.	NAT	Network Address Translation
6.	STUN	Session Traversal Utilities for NAT
7.	TURN	Traversal Using Relays around NAT
8.	WEBRTC	Web Realtime Communication
9.	W3C	World Wide Web Consortium
10.	IETF	Internet Engineering Task Force
11.	API	Application Programming Interface
12.	UDP	User Datagram Protocol
13.	HTML	Hyper-Text Markup Language
14.	P2P	Peer-to-Peer
15.	FTP	File Transfer Protocol
16.	SIP	Session Initiation Protocol
17.	PSTN	Public switched telephone network
18.	CORS	Cross Origin Resource Sharing
19.	JSON	JavaScript Object Notation
20.	JSEP	Javascript Session Establishment Protocol

DANH MỤC CÁC BẢNG

DANH MỤC CÁC HÌNH VẼ

CHƯƠNG 1. MỞ ĐẦU

1.1. Đặt vấn đề

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.

1.2. Phạm vi và mục tiêu của luận văn

Luận văn 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 EasyRTC.

1.3. Phương pháp và bố cục nghiên cứu

Luận văn được chia thành ba chương với nội dung sau:

Chương 1 – Lời mở đầu

Chương 2 – Tổng quan về WebRTC. Chương này giới thiệu chung về lịch sử, sự tiện lợi, các APIs và giao thức được sử dụng trong WebRTC

Chương 3 – Báo hiệu, thiết lập phiên trong WebRTC. Chương này đi sâu vào tìm hiểu, phân tích việc sử dụng báo hiệu và kênh báo hiệu để thiết lập phiên kết nối Peer-to-peer trong WebRTC.

Chương 4 – Ứng dụng WebRTC trong giải pháp cộng tác và chia sẻ dữ liệu đa Phương tiện tại Trung tâm MVAS – TCT Viễn thông Mobifone. Chương này giới thiệu các cách tiếp cận sử dụng WebRTC trong xây dựng ứng dụng, giới thiệu framework EasyRTC và sử dụng EasyRTC demo ứng dụng cộng tác tại Trung tâm MVAS – TCT viễn thông Mobifone.

Chương 5 - Kết luận: Kết quả đạt được và hướng phát triển tiếp theo.

CHƯƠNG 2. 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.

2.1. 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

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 [11], 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.

- **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:

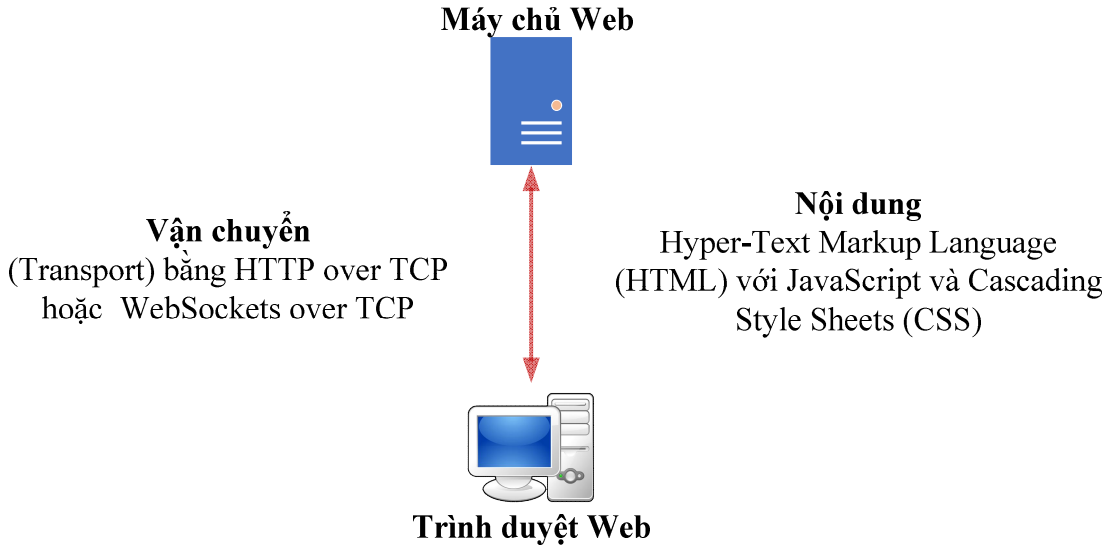
Bảng 2.1: Những tính năng mới của WebRTC (tổng hợp theo [1])

Tính năng	Cách thức cung cấp	Lý do quan trọng
Độc lập với Platform và thiết bị	Sử dụng các APIs chuẩn từ W3C, các giao thức từ IETF	Nhà phát triển có thể viết mã WebRTC HTML5 có thể chạy trên các hệ điều hành, trình duyệt, thiết bị desktop và mobile khác nhau
Bảo mật voice và video	Sử dụng Secure RTP Protocol cho mã hóa và xác thực.	Đảm bảo trình duyệt khi được sử dụng ở các môi trường khác nhau (không bảo mật như wifi công cộng) an toàn. Mã hóa giúp người khác ko thể nghe hay ghi lại voice, video
Chất lượng voice, video cao	Sử dụng Codec Opus cho audio, VP8 cho video	Có chuẩn built-in codecs đảm bảo khả năng tương tác và tránh việc phải download codecs, tiềm ẩn nguy cơ bị cài đặt spyware, virus từ các trang web giả mạo.
Thiết lập phiên tin cậy	Sử dụng kỹ thuật Hole punching[20] để vượt NAT	Truyền media trực tiếp giữa trình duyệt giúp tin cậy hơn, cho chất lượng tốt hơn là phải relay qua máy chủ. Ngoài ra cũng giúp máy chủ giảm được tải xử lý.
Cho phép nhiều dòng (stream) và loại media gửi qua một địa chỉ transport (single transport)	Sử dụng Real-time Transport Protocol (RTP) và Session Description Protocol (SDP) extensions	Thiết lập kênh truyền media trực tiếp sử dụng hole punching tuy mất thêm thời gian nhưng việc gửi tất cả media qua một phiên đơn giúp hiệu quả và tin cậy hơn
Thích ứng với điều kiện của mạng	Sử dụng Multiplexed RTP Control Protocol, Secure Audio Video Profile with Feedback (SAVPF) [29]	Cơ chế phản hồi theo điều kiện của mạng là cần thiết với video, đặc biệt quan trọng với phiên WebRTC có độ nét cao

Hỗ trợ nhiều loại media và nhiều nguồn của media	Sử dụng APIs và báo hiệu để thống nhất size/format của mỗi nguồn media riêng biệt	Khả năng thống nhất mỗi nguồn media riêng biệt giúp tối ưu sử dụng băng thông và những tài nguyên khác.
Khả năng tương tác với hệ thống VoIP và hệ thống truyền thông video sử dụng SIP, Jingle và PSTN	Sử dụng Secure Real-time Transport Protocol (SRTP) và Secure Real-time Control Transport Protocol (SRCTP) [17]	Những hệ thốngVoIP, Video đang tồn tại có thể kết nối với hệ thống WebRTC sử dụng những giao thức chuẩn.

2.2. Kiến trúc WebRTC

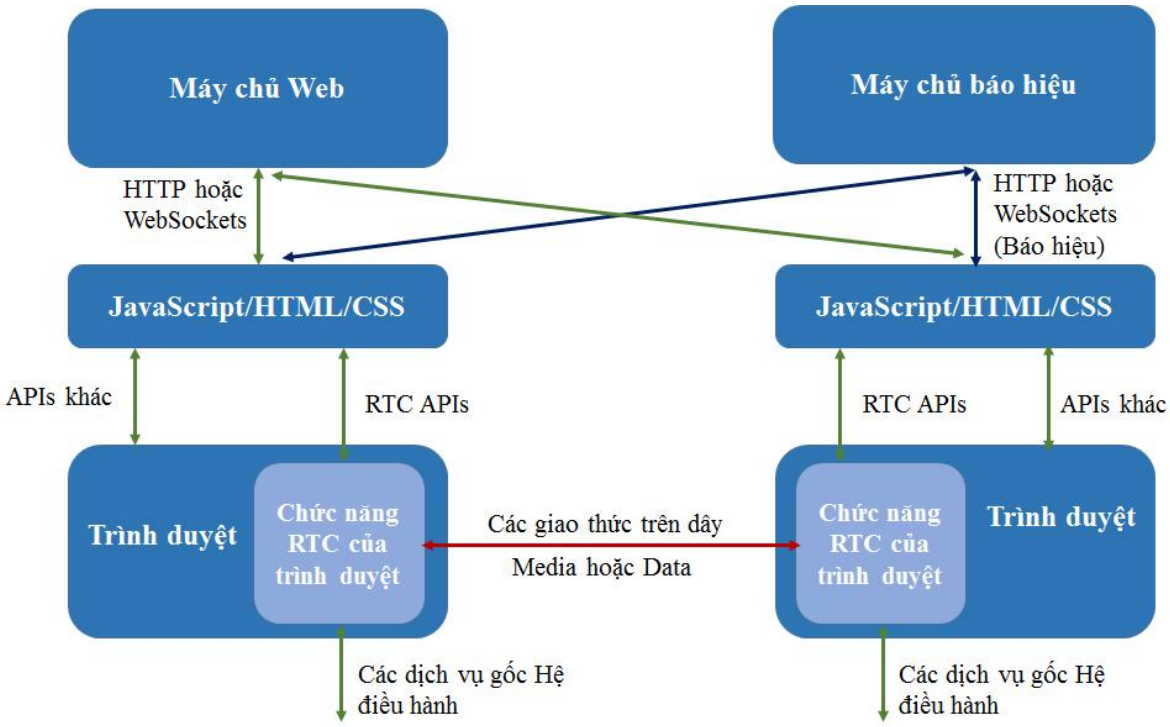
Kiến trúc web cổ điển dựa trên mô hình client-server, trong đó trình duyệt gửi yêu cầu HTTP đến máy chủ để lấy nội dung, máy chủ trả lời, gửi nội dung về cho trình duyệt dưới dạng HTML, thường kèm theo JavaScript và Cascading Style Sheets (CSS). Trong trường hợp đơn giản thì khi máy chủ web trả lời yêu 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…[2].

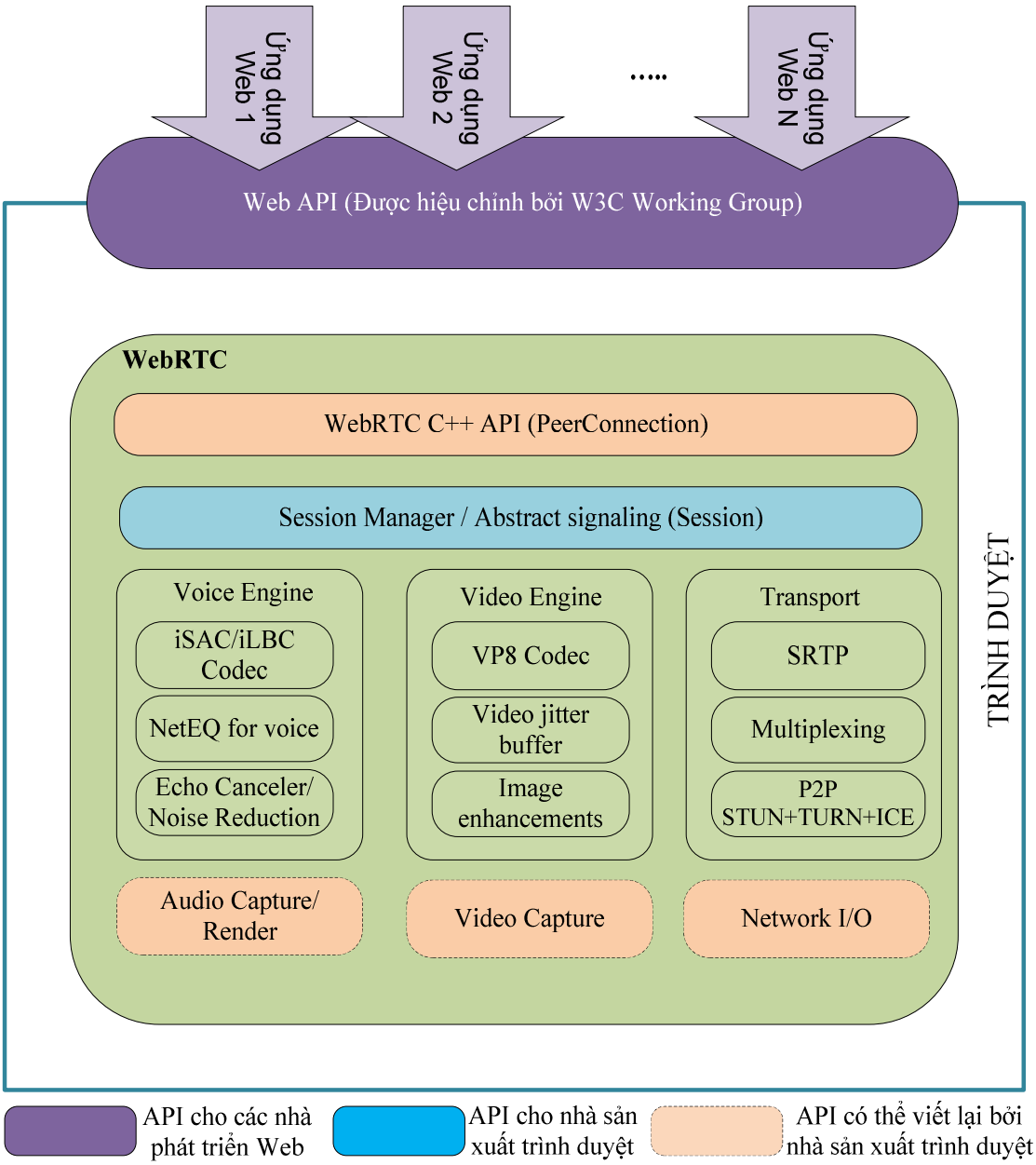


Hình 2.2: Truyền thông thời gian thực trong trình duyệt (nguồn [1])

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 [Nguồn 10]

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 2.3.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 [12]. 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 2.4. Các tầng giao thức trong WebRTCs.

2.3. 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 Peers

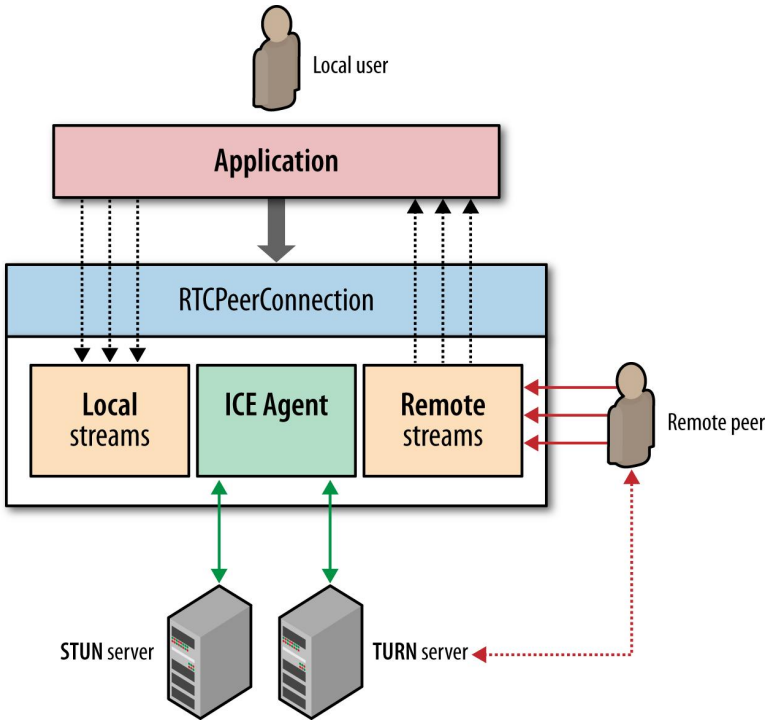
Để làm được các điều trên, WebRTC đang trong quá trình chuẩn hóa và sử dụng các APIs quanh ba khái niệm chính:

- 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.

a. 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 [Nguồn 4]

Những nhiệm vụ chính của RTCPeerConnection là

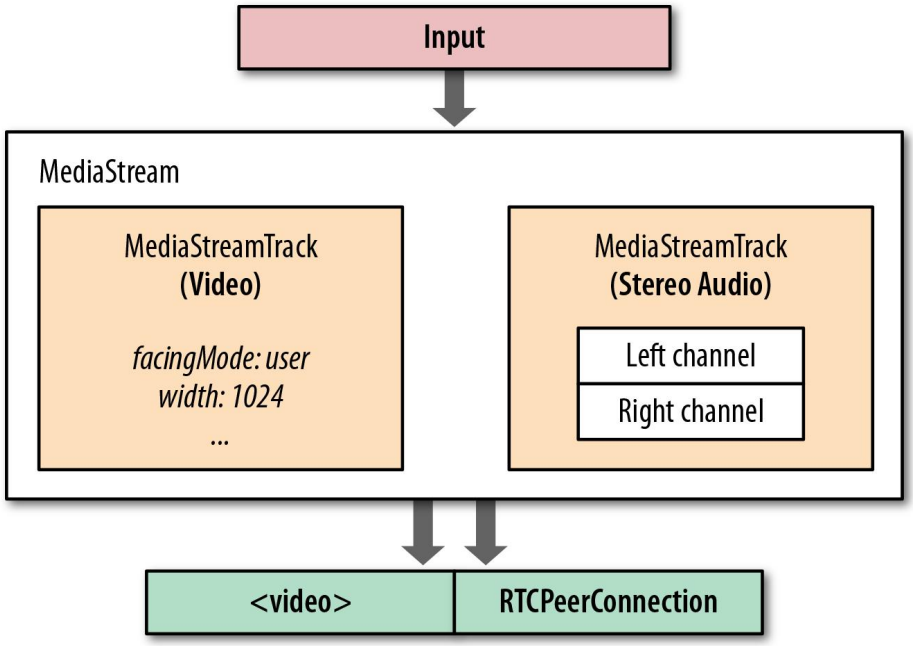
- ✓ Điều khiển toàn bộ quá trình Interactive Connectivity Establishment (ICE) [13] để vượt NAT.
- ✓ Gửi tự động bản tin STUN [14] keepalives giữa các peers
- ✓ Giám sát dòng dữ liệu local và dòng dữ liệu ở xa (remote stream)
- ✓ Tự động kích hoạt (triggers) quá trình thương lượng lại dòng dữ liệu theo 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.

b. 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 [Nguồn 4]

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 [9].

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.

c. 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 [15] , 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 [4]

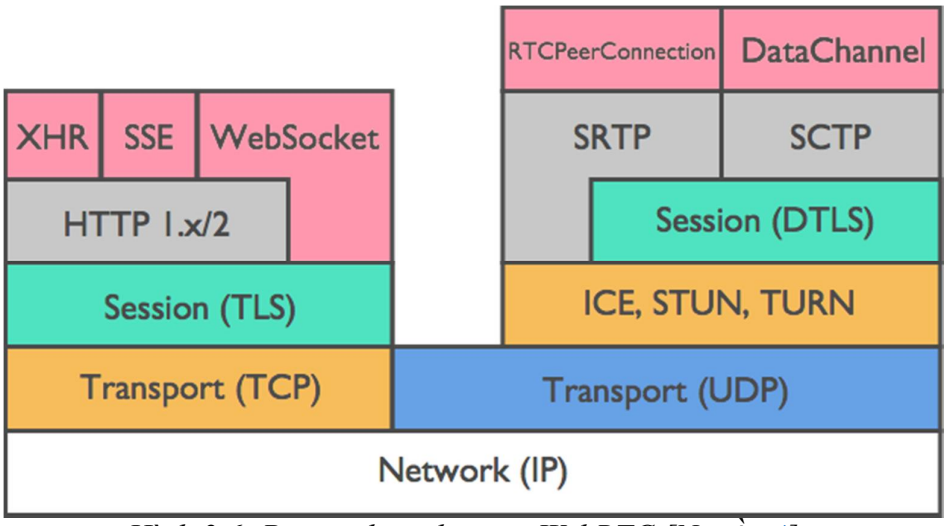
Tính năng	WebSocket	DataChannel
Encryption	configurable	always
Reliability	reliable	configurable
Delivery	ordered	configurable
Multiplexed	no (extension)	yes
Transmission	message-oriented	message-oriented
Binary transfers	yes	yes
UTF-8 transfers	yes	yes
Compression	no (extension)	no

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).

2.4. Các tầng giao thức trong WebRTC

Như đã nêu ở mục 2.1, 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 2.5 dưới đây:



Hình 2.6: Protocol stack trong WebRTC [Nguồn 4]

✓ SRTP

Giao thức quan trọng nhất mà WebRTC sử dụng là Secure Real-time Transport Protocol, hay SRTP [17]. 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 [15] 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

Tính năng	TCP	UDP	SCTP
Reliability	reliable	unreliable	configurable
Delivery	ordered	unordered	configurable
Transmission	byte-oriented	message-oriented	message-oriented
Flow control	yes	no	yes
Congestion control	yes	no	yes

✓ **SDP**

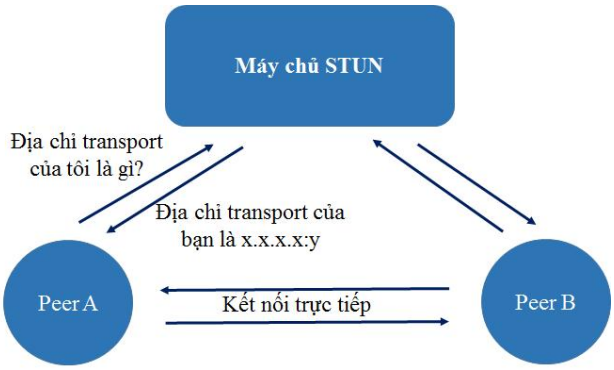
WebRTC sử dụng Session Description Protocol, SDP [18], được encode trong đối tượng RTCSessionDescription, để mô tả đặc tính media của hai đầu trong kết nối P2P như loại media đề truyền/nhận (audio, video, application data), network transports, loại codecs sử dụng và cấu hình, thông tin băng thông, và 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 [24].

✓ **DTLS**

Datagram Transport Layer Security- DTLS [16] dựa trên giao thức TLS, cung cấp tính bảo mật và toàn vẹn dữ liệu truyền giữa các ứng dụng tương tự TLS [19]. 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 [14]: 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 [3]

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 [20] sẽ không thành công, cần phải sử dụng đến TURN - *Traversal Using Relays around NAT [21]*.

✓ **TURN**

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 [13] 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.

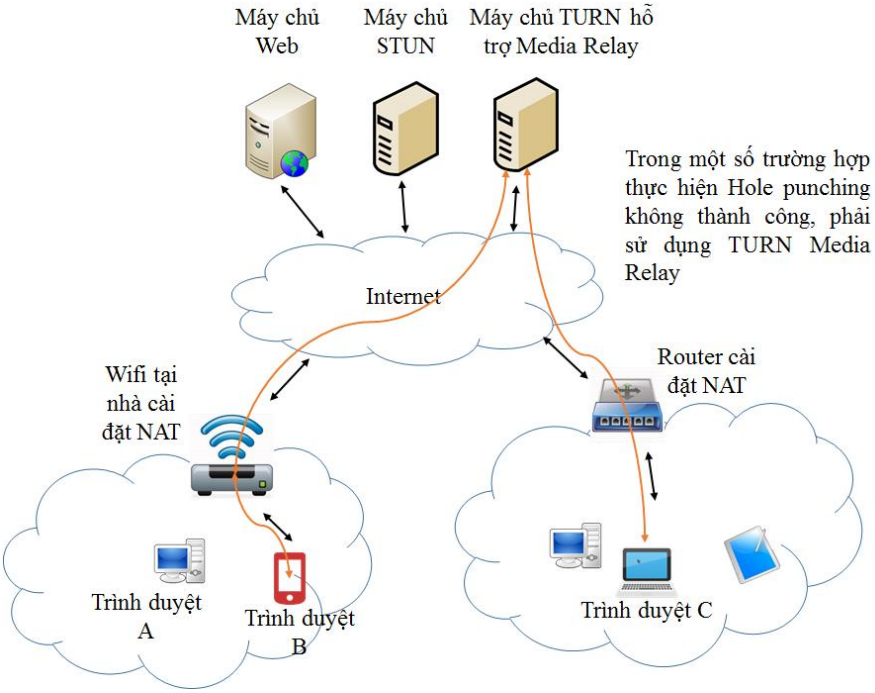
ICE sử dụng kỹ thuật hole punching [20], kỹ thuật được tiên phong bởi các game thủ chơi online, những người cần trao đổi gói tin trực tiếp giữa các PCs cho dù giữa chúng có NAT. Hole punching là kỹ thuật cho phép "đục lỗ" qua thiết bị NAT để thiết lập kết nối trực tiếp giữa ứng dụng ngay cả khi cả hai 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.

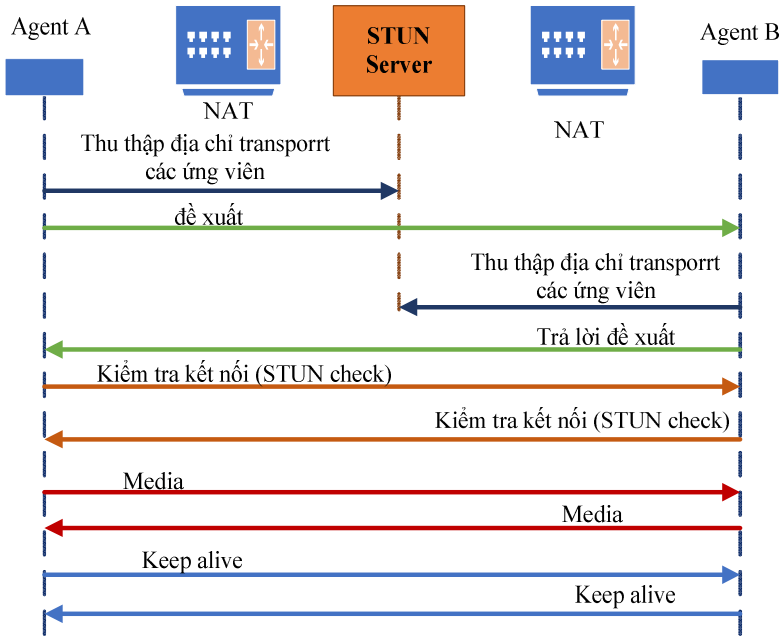
Yêu cầu đầu tiên thỏa mãn khi sử dụng máy chủ web để điều phối quá trình hole punching, bởi máy chủ web biết có nhu cầu mong muốn thiết lập giữa hai trình duyệt, do đó nó đảm bảo việc hai trình duyệt bắt đầu quá trình hole punching cùng thời điểm ở mức tương đối. Yêu cầu thứ 2 được thỏa mãn bằng cách 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]

Quy trình hoạt động của ICE: Để thiết lập được đường định tuyến giữa các peer, WebRTC, cụ thể là đối tượng PeerConnection có một ICE agent cho nhiều nhiệm vụ như thu thập thông tin địa chỉ ứng viên, kiểm tra kết nối giữa các peer, gửi các thông tin keepalives.



Hình 2.9: Quy trình hoạt động ICE mức cao

Quy trình các bước cơ bản trong ICE thể hiện ở hình 2.9

- Bước 1: Thu thập địa chỉ vận chuyển ứng viên (Gather Candidate Transport Addressé)

Bước đầu tiên là tập hợp các địa chỉ vận chuyển của ứng viên. Địa chỉ vận chuyển ứng viên là địa chỉ IP và port mà media có thể được nhận từ PeerConnection. Những địa chỉ này phải được tập hợp vào đúng thời điểm cuộc gọi, nó không được tập hợp trước trong 1 số trường hợp. Như trong hình 2.9 trên, ICE Agent A bắt đầu quá trình tập hợp này khi người dùng tại A khởi tạo PeerConnection với B. ICE Agent B bắt đầu tập hợp địa chỉ ứng viên ngay khi yêu cầu PeerConnection được nhận từ A qua kênh báo hiệu. Có bốn loại địa chỉ ứng viên là Host, Server Reflexive, Peer Reflexive, Relayed. Host là địa chỉ card mạng qua hệ điều hành. Nếu ICE Agent đứng sau NAT, địa chỉ này sẽ là địa chỉ nội bộ và không được định tuyến đến được ngoài subnet. Loại địa chỉ thứ 2 là địa chỉ Reflexive, là địa chỉ STUN gửi về cho "ICE Agent" trong PeerConnection. Nếu ICE Agent đứng sau NAT, thì địa chỉ này là địa chỉ ngoài cùng trong quá trình NAT (nếu có nhiều lớp NAT). Địa chỉ thứ 3 Peer reflexive là địa chỉ do ICE Agent đầu xa gửi sau quá trình STUN check. Địa chỉ này không được trao đổi qua kênh báo hiệu nhưng được nhận ra trong phần kiểm tra kết nối STUN ở bước 3. Địa chỉ ứng viên relayed là địa chỉ của máy chủ Media relay, thường được lấy thông qua giao thức TURN, cụ thể hơn là qua TURN allocation request. Khi session description được đặt (local hoặc remote), ICE Agent local sẽ tự động bắt đầu quá trình tìm kiếm tất cả địa chỉ ứng viên có thể: lấy thông tin local IP, truy vấn STUN server để lấy IP pubic và cổng, bổ sung thông tin TURN server như là ứng viên cuối cùng.

- Bước 2: Trao đổi thông tin ứng viên qua kênh báo hiệu (Exchange of Candidates)

Khi thu thập thông tin ứng viên hoàn thành, hàm callback thường là onicecandidate được gọi, trong đó tạo ra SDP offer với thông tin ứng viên và gửi cho peer qua kênh báo hiệu.

- Bước 3: Thực hiện kiểm tra kết nối (STUN Connectivity Checks)

ICE Agents bắt đầu quá trình kiểm tra kết nối ngay khi chúng gửi và nhận danh sách ứng viên. Với Agent A, thời điểm kiểm tra là lúc nhận được trả lời SDP từ Agent B. Với Agent B, thời điểm là lúc nó gửi trả lời SDP đến Agent A. Trong quá trình này, ICE Agents tại local gửi các thông điệp (STUN binding request), trong khi peer cần xác nhận với một STUN response thành công. Bước đầu tiên để ghép cặp ứng viên dựa trên kiểu địa chỉ IP (Ipv4 hoặc Ipv6) và một số yếu tố khác. Mục đích của việc ghép cặp và tạo thiết lập cho mỗi cặp để giảm số lượng kiểm tra kết nối phải thực hiện, giảm thiểu thời gian cần thiết để đạt được những ứng viên phù hợp. Địa chỉ ứng viên Peer reflexive có thể được phát hiện trong bước này và tự động được ghép cặp nếu được phát hiện. Có

thể tối ưu quá trình ICE bởi Trickle ICE (một mở rộng của giao thức ICE), cho phép thu thập và kiểm tra kết nối peer từng phần thay vì tất cả ứng viên được cung cấp tại thời điểm bắt đầu quá trình ICE. ICE Trickle sẽ được bắt đầu với tập nhỏ, và các ứng viên sẽ được thêm vào dần hoặc bỏ đi khi quá trình diễn ra. Những ứng viên mới được ghép cặp, xếp hàng và đi vào cùng các bước như hình trên.

- Bước 4: Chọn cặp và bắt đầu truyền Media (Choose Selected Pair and Begin Media)

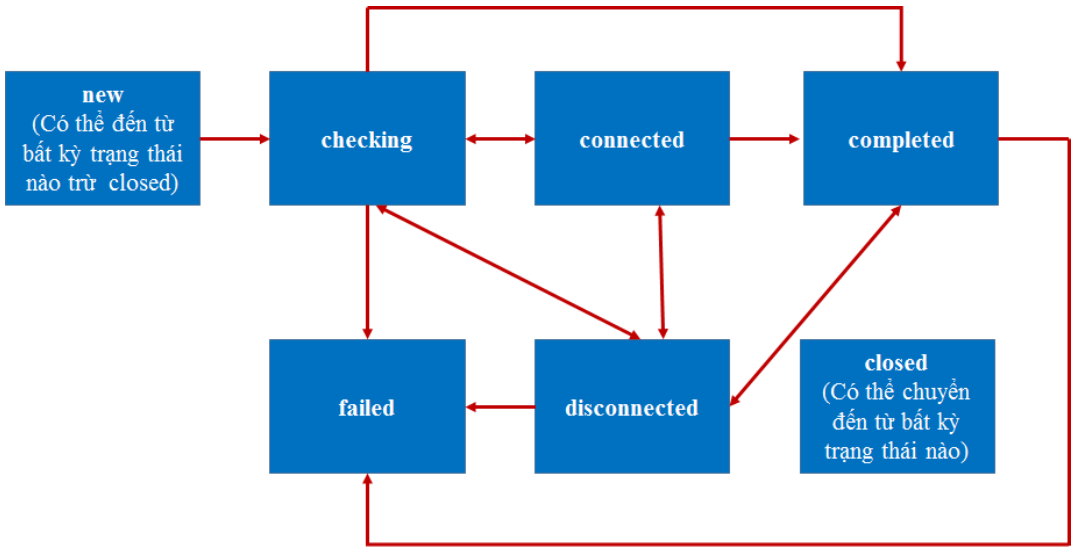
Sau khi việc kiểm tra các ứng viên hoàn, sẽ có một cặp được lựa chọn, media lúc này sẽ được gửi bởi 2 trình duyệt sử dụng cặp ứng viên đã được lựa chọn.

- Bước 5: Keep-Alives

Để chắc chắn việc NAT và các luật lọc không làm kết nối bị timeout trong suốt phiên media, ICE Agent tiếp tục gửi "STUN connectivity checks" khu kỳ 15 giây qua cặp ứng viên đang sử dụng ngay cả khi không gửi media. Nếu media session vẫn active, ICE Agent đầu xa sẽ khởi tạo một STUN trả lời. Việc nhận STUN trả lời chỉ ra rằng media có thể tiếp tục gửi. Nếu STUN response không nhận được, ICE sẽ khởi động lại. Hành vi này không được định nghĩa trong đặc tả ICE nhưng được định nghĩa trong ICE Extension.

- Bước 6: Khởi động lại quá trình ICE (ICE Restart)

Việc khởi động lại ICE được kích hoạt khi ICE Agent phát hiện thấy thay đổi địa chỉ vận chuyển mà cặp ứng viên sử dụng. Điều này làm cho ICE Agent quay lại bước 1, tập hợp ứng viên và gửi thông tin những ứng viên trong SDP offer cho ICE Agent. ICE Agent phía peer cũng quay lại bước 1 và toàn bộ quá trình ICE được lặp lại. Nó thường sẽ xảy ra nếu người dùng reload trang web trong khi đã thiết lập cặp ứng viên. Hiện tại các tổ chức vẫn tiếp tục nghiên cứu và chuẩn hóa cách xử lý/hanlde tình huống này [1]



Hình 2.10: Sơ đồ chuyển trạng thái trong ICE

Hình 2.10 mô tả việc chuyển trạng thái trong quá trình ICE diễn ra. Trạng thái new là trong quá trình tập hợp ứng viên và/hoặc đợi ứng viên xa để được cung cấp, và chưa bắt đầu vào quá trình kiểm tra. RTCIceTransportState chuyển sang trạng thái checking khi đã tìm được ít nhất một ứng viên. Trạng thái connected là khi đã tìm được kết nối có thể sử dụng giữa cặp ứng viên, nhưng tiếp tục tìm kiếm cặp ứng viên tốt hơn. Trạng thái completed là kết thúc quá trình tìm kiếm, chỉ ra không còn ứng viên xa, kết thúc mọi việc kiểm tra các cặp ứng viên và tìm thấy kết nối. Trạng thái failed là khi kết thúc quá trình tìm kiếm, kết quả là không có ứng viên, hoặc có ứng viên nhưng kiểm tra bị fail, không được đồng ý kết nối. Disconnected khi mất mạng hoặc không nhận được các phản hồi kiểm tra STUN. Cuối cùng là trạng thái Closed, khi không còn sự phản hồi STUN request.

CHƯƠNG 3.BÁO HIỆU TRONG WEBRTC

Chương 2 luận văn đã trình bày những thông tin tổng quan về WebRTC, từ kiến trúc đến các APIs, Protocols. Chương 3 của luận văn này đi vào mô tả báo hiệu trong WebRTC, cách sử dụng các APIs, các tiến trình chính trong việc thiết lập phiên kết nối Peer-to-Peer giữa các trình duyệt.

3.1. Vai trò của báo hiệu

Để có thể thiết lập phiên làm việc trực tiếp giữa trình duyệt, đầu tiên trình duyệt muốn kết nối với trình duyệt còn lại phải lấy thông tin địa chỉ mạng, kiểm tra kết nối, xác định đầu xa sẵn sàng cho kết nối chưa. Sau đó trình duyệt tiếp tục trao đổi thông tin codecs hay kiểu media (đối với kết nối streaming media) trước khi thiết lập Peer Connection. Do ban đầu chưa có kết nối giữa hai trình duyệt, các thông tin trên cần được chuyển tiếp qua máy chủ trung gian trước khi đến trình duyệt kia. Quá trình chuyển các thông điệp từ trình duyệt này qua máy chủ trung gian (gọi là máy chủ báo hiệu - Signaling Server) đến trình duyệt khác được gọi là quá trình báo hiệu, hay gọi tắt là báo hiệu.

Báo hiệu có vai trò rất quan trọng trong WebRTC, tuy nhiên nó không được chuẩn hóa, cho phép nhà phát triển ứng dụng tự do lựa chọn phương án phù hợp. Trong truyền thông thời gian thực, báo hiệu có các vai trò chính:

- *Xác định địa chỉ vận chuyển (IP và port) của peer:* trao đổi địa chỉ ứng viên trong quá trình ICE hole punching đã nêu ở Chương 2, mục 2.4.
- *Thương lượng/thống nhất khả năng media và các thiết lập cấu hình*: đây là chức năng quan trọng nhất của báo hiệu, giúp trao đổi thông tin thường được chứa trong đối tượng SDP giữa các trình duyệt tham gia vào Peer Connection. SDP chứa tất cả các thông tin cần thiết cho RTP media stack trên trình duyệt để cấu hình media session, bao gồm loại media (voice, video, data), codecs sử dụng (Opus, G.711,..) hay bất kỳ tham số hay thiết lập nào cho codecs, thông tin về băng thông.
- *Điều khiển media session*: khởi tạo, đóng, thay đổi session

Lý do báo hiệu không được chuẩn hóa trong WebRTC đơn giản vì nó không cần được chuẩn hóa để giúp trình duyệt có khả năng tương tác với nhau. Máy chủ web có thể đảm bảo hai trình duyệt sử dụng cùng giao thức báo hiệu với mã JavaScript được tải về. Trong mô hình web, chỉ một số ít thành phần được chuẩn hóa, cho phép nhà phát triển web tự do lựa chọn và thiết kế tất cả khía cạnh trang web và ứng dụng. Thực tế, chỉ phần chuyển vận (HTTP), markup (HTML), và media (WebRTC) cần được chuẩn hóa. Máy chủ lựa chọn giao thức báo hiệu và chắc chắn được rằng người dùng của ứng dụng web hoặc site hỗ trợ cùng một giao thức. Nó khác với các hệ thống VoIP và hệ thống Video, những hệ thống không có cách nào đẩy (push) mã báo hiệu đến đầu cuối,

nên các đầu cuối phải sử dụng cùng giao thức báo hiệu chuẩn, như SIP hoặc Jingle để có thể giao tiếp với nhau. Hiện nay, quá trình báo hiệu đang được xây dựng dựa trên Javascript Session Establishment Protocol (JSEP) [25], là cơ chế cho phép ứng dụng JavaScript kiểm soát hoàn toàn phần báo hiệu của phiên đa phương tiện qua API RTCPeerConnection.

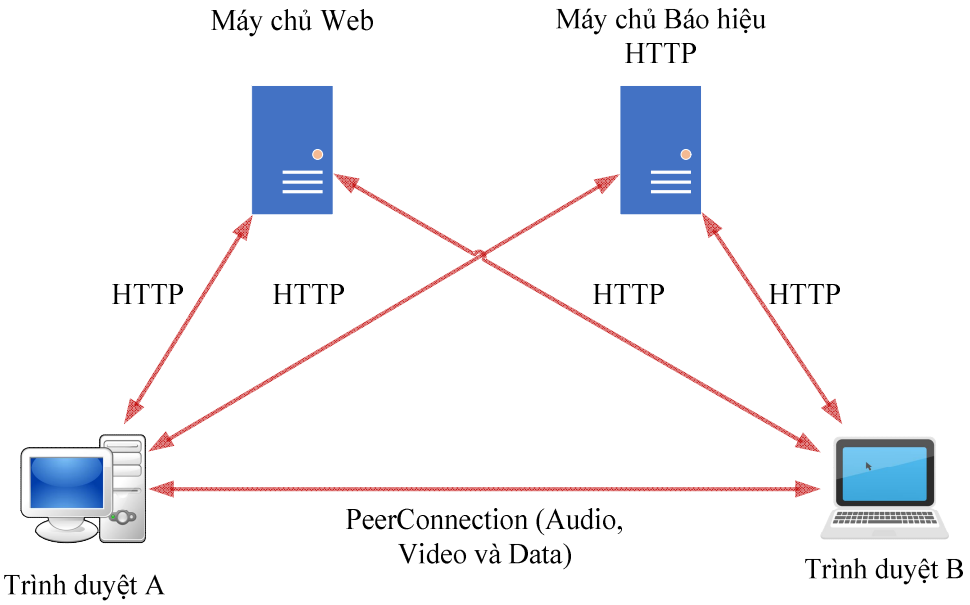
3.2. Giao thức vận chuyển báo hiệu

Các giao thức được sử dụng phổ biến cho vận chuyển báo hiệu WebRTC là HTTP, WebSocket và cả kênh DataChannel.

✓*Vận chuyển bằng HTTP*

HTTP có thể sử dụng để vận chuyển báo hiệu WebRTC. Trình duyệt khởi tạo HTTP request để gửi và nhận thông tin báo hiệu từ máy chủ. Thông tin có thể được vận chuyển sử dụng phương thức GET hay POST, hay trong các phản hồi. Nếu máy chủ sử dụng cho mục đích báo hiệu hỗ trợ Cross Origin Resource Sharing (CORS), một chuẩn truy cập tài nguyên web trên domain khác, máy chủ báo hiệu có thể ở địa chỉ IP khác với máy chủ web.

Việc gửi thông tin đến máy chủ không phức tạp, sử dụng lời gọi API XHR (XML HTTP Request) trong JavaScript. Ở hướng ngược lại, việc nhận thông tin bất đồng bộ từ máy chủ phức tạp hơn, có thể sử dụng công nghệ đã được phát triển là AJAX (Asynchronous JavaScript And XML). Một số phương pháp tiếp cận đơn giản tương tự như polling hay giữ GET request liên tục mở để sẵn sàng nhận data từ máy chủ.



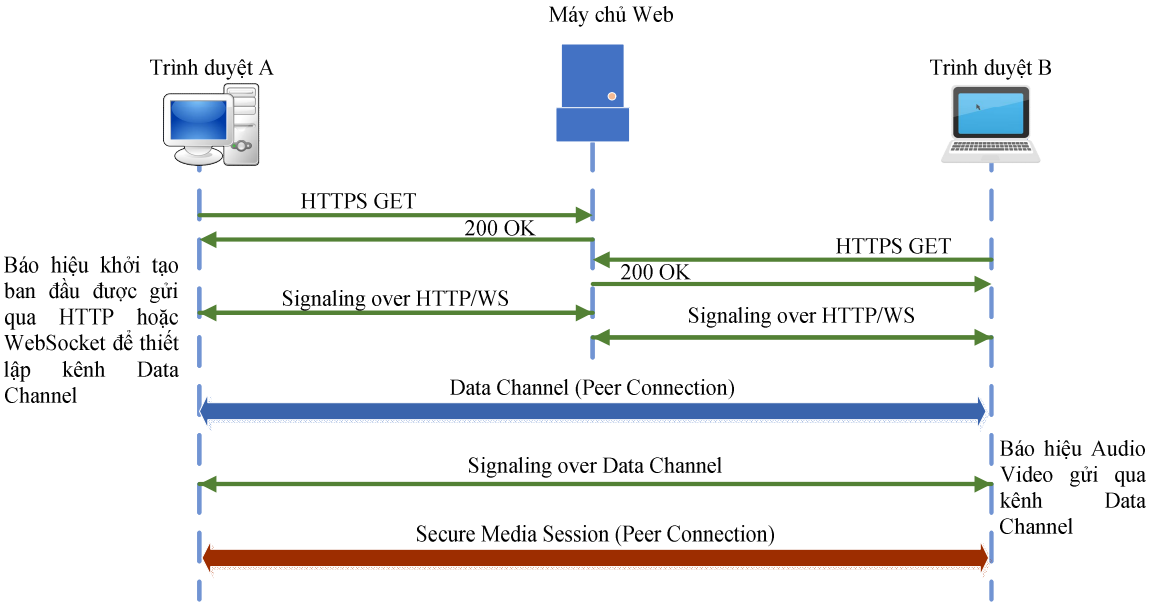
Hình 3.1: HTTP Transport cho báo hiệu

✓*Vận chuyển bằng WebSocket*

WebSocket cho phép trình duyệt mở kết nối hai chiều đến máy chủ. Kết nối bắt đầu là HTTP request nhưng sau đó nâng lên WebSocket. Khi máy chủ hỗ trợ CORS, máy chủ Websocket có thể là máy chủ khác máy chủ web (IP khác). Để các trình duyệt kết nối đến, Websocket phải chạy trên máy chủ HTTP. Do không thể mở WebSocket trực tiếp với trình duyệt khác vì trình duyệt triển khai chức năng HTTP user agent và không phải chức năng máy chủ HTTP, máy chủ WebSocket vẫn cần để relay giữa 2 web client sử dụng WebSocket. Trình duyệt dùng WebSocket bằng cách sử dụng WebSocket API [WS-API]. WebSocket được hỗ trợ bởi tất cả các hãng cung cấp trình duyệt lớn, tuy nhiên một vài web proxy và firewall không hỗ trợ WebSocket đầy đủ có thể gây ra những vấn đề nhất định, đặc biệt là về xác thực.

✓*Vận chuyển bằng chính DataChannel*

Kênh dữ liệu, sau khi được thiết lập P2P giữa trình duyệt, cung cấp kết nối trực tiếp, độ trễ thấp nên cũng phù hợp với việc vận chuyển báo hiệu. Vì sự khởi tạo thiết lập kênh dữ liệu (Data Channel) đòi hỏi cơ chế báo hiệu riêng, kênh dữ liệu không thể sử cho tất cả báo hiệu WebRTC. Tuy nhiên, nó có thể sử dụng để handle các báo hiệu khác sau khi thiết lập thành công kênh trực tiếp, bao gồm báo hiệu cho audio, video media qua Peer Connection.



Hình 3.2. Vận chuyển báo hiệu trên Data Channel

Hình 3.2 thể hiện báo hiệu cho kênh dữ liệu được chuyển tải qua máy chủ Web, trên thực tế thì thường dùng máy chủ Websocket. Tải cho phần báo hiệu cho kênh dữ liệu thấp hơn nhiều so với việc handle phần báo hiệu cho voice, video. Một ưu điểm của việc vận chuyển báo hiệu trên kênh dữ liệu là nó có thể sử dụng để giảm thời gian thiết

lập kết nối. Trước khi media có thể trao đổi giữa trình duyệt, ICE và bước quản lý khóa DTLS-SRTP phải hoàn thành. Nếu các bước này chỉ bắt đầu sau khi có đồng ý kết nối tạo phiên, nó có thể gây ra độ trễ đến vài giây. Với báo hiệu qua kênh dữ liệu, ICE và quản lý khóa DTLS-SRTP được thực hiện để thiết lập kênh dữ liệu, và có thể bắt đầu trước khi người dùng được cảnh báo hoặc hỏi cho việc chấp thuận kết nối. Khi người dùng chấp nhận kết nối, việc truyền voice và video có thể rất nhanh, vì thông điệp báo hiệu truyền đi trực tiếp từ trình duyệt, và dòng media tái sử dụng Peer Connection nên không cần quá trình ICE hoặc DTLS-SRTP thực hiện nữa.

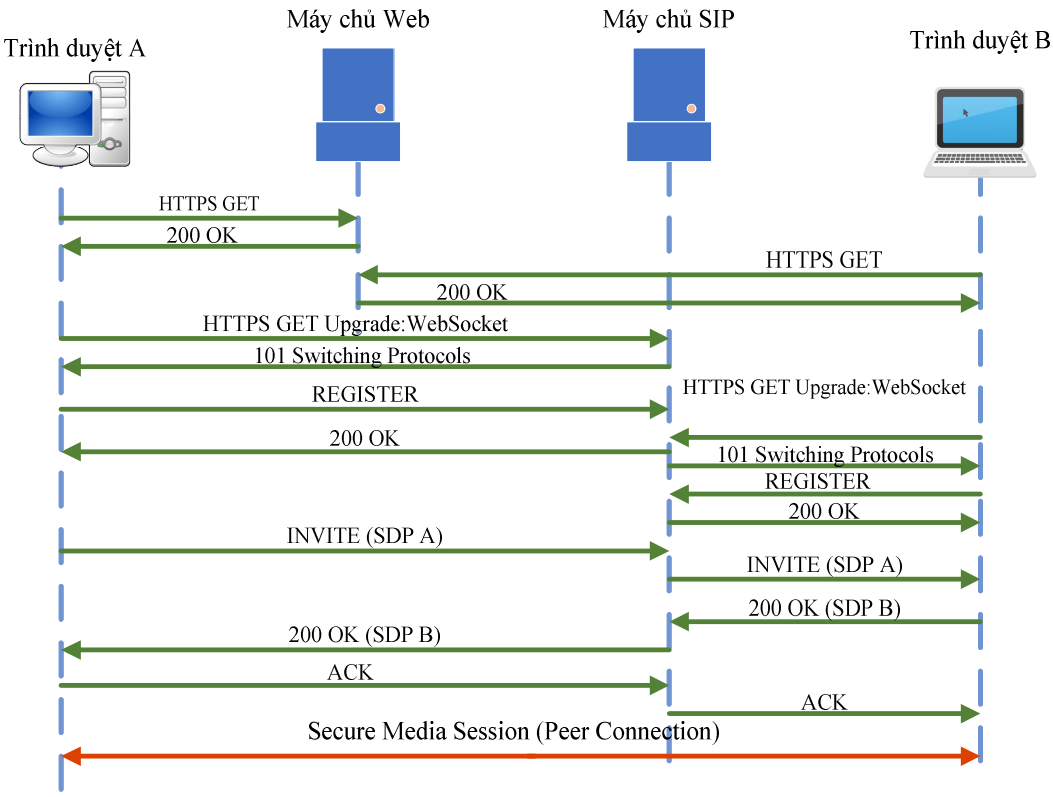
3.3. Giao thức báo hiệu

Một phần quan trọng trong xây dựng ứng dụng WebRTC là lựa chọn giao thức báo hiệu, nó không cần thiết gắn với việc lựa chọn giao thức vận chuyển báo hiệu. Ta có thể chọn giao thức báo hiệu chuẩn như SIP, Jingle hoặc một cách báo hiệu riêng tự định nghĩa. Không phụ thuộc vào giao thức báo hiệu, sẽ luôn có một dạng máy trạng thái (State Machine) báo hiệu. Không phải tất cả trạng thái trong máy trạng thái báo hiệu cần thông điệp báo hiệu được trao đổi giữa trình duyệt, dù một vài hướng tiếp cận báo hiệu có thực hiện. Việc sử dụng những giao thức báo hiệu chuẩn có điểm thuận lợi là máy trạng thái đã được định nghĩa đầy đủ. Thông điệp báo hiệu sẽ chứa thông tin hữu ích cho việc định tuyến trạng thái. Còn điểm mạnh sử dụng những giao thức báo hiệu riêng là nó đơn giản, chỉ cần cung cấp tính năng mà ứng dụng cần. Nếu xây dựng ứng dụng WebRTC mục đích kết nối, truyền dữ liệu media, data P2P chỉ giữa trình duyệt, không qua những trung gian hay đầu cuối SIP, Jingle, VoIP…thì đây là một sự lựa chọn tốt. Các giao thức báo hiệu WebRTC được dùng phổ biến là SIP, XMPP, hoặc truyền đối tượng JSON.

✓*Sử dụng giao thức báo hiệu SIP*

SIP là giao thức báo hiệu thường sử dụng trong VoIP và hệ thống hội nghị truyền hình. SIP có thể sử dụng UDP, TCP, SCTP hay TLS cho việc vận chuyển, tuy nhiên thường thì sử dụng Websocket. Ở cách tiếp cận này, trình duyệt tải JavaScript SIP User Agent Library (chẳng hạn như sipML5, jsSIP..), sau đó thiết lập kết nối (REGISTER) với SIP Proxy Server/Registrar mà hỗ trợ vận chuyển WebSocket. Trình duyệt khởi tạo phiên WebRTC sẽ gửi lời mời INVITE chứa SDP offer gửi đến Sip Proxy Server. Trình duyệt đích cần được định danh bởi SIP URI. Trình duyệt đích sẽ nhận được INVITE và khởi tạo SIP response tương ứng, trả về SDP answer trong bản tin 200 OK response, từ đó thiết lập media session

Luồng thông điệp báo hiệu sử dụng SIP over WebSocket thể hiện ở hình sau:



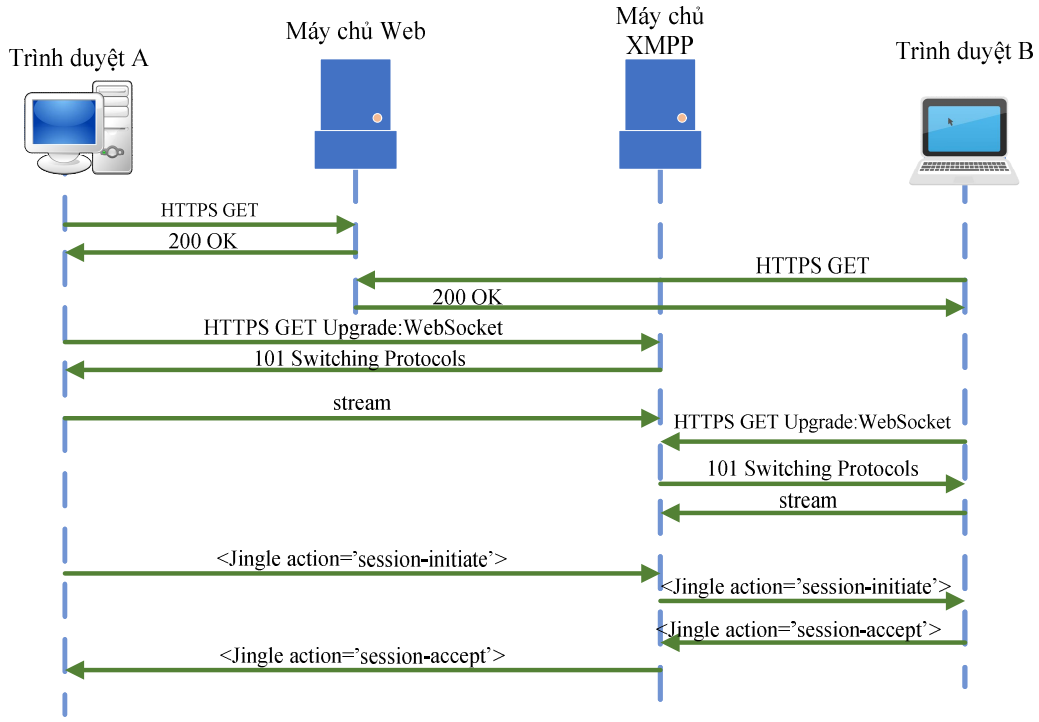
Hình 3.3: Giao thức báo hiệu SIP trên WebSocket

Thực tế việc phân tích và mã hóa thông điệp SIP trong JavaScript là không tối ưu về hiệu năng, tuy nhiên một số framework như QofficeSIP, sipML5, jsSIP chỉ ra rằng nếu sử dụng thì không ảnh hưởng đến trải nghiệm người dùng. Do SIP không được thiết kế, và không dễ dàng thích ứng với việc tối ưu Trickle ICE để giảm thời gian thiết lập kết nối, nên dễ dẫn đến sự chậm trễ mà người dùng khó chấp nhận. Sự chậm trễ này hay xảy ra với các thiết bị đầu cuối được cấu hình một hoặc nhiều giao diện mạng mà không thể kết nối đến máy chủ STUN và TURN, là tình trạng khá phổ biến với các thiết bị multi-homed như điện thoại thông minh mà đồng thời kết nối với 3G / 4G và mạng WiFi, cũng như với máy tính xách tay chạy VPN, máy ảo. Đã có thực hiện demo sipML5 trên kết nối OpenVPN mất hơn 10 giây để khởi tạo "Invite out" [27]. Ngắt kết nối VPN thì sự chậm chễ chỉ xuống ít hơn 1 giây.

✓*Sử dụng giao thức báo hiệu Jingle over WebSockets*

Jingle là một mở rộng của XMPP (extensible Messaging and Presence Protocol), được biết đến là Jabber [30]), thêm khả năng báo hiệu media cho XMPP. Jingle cung cấp cách chuyển mô tả phiên SDP sang định dạng XML, sau đó có thể được vận chuyển qua TCP hoặc TLS đến máy chủ XMPP. Để triển khai báo hiệu sử dụng Jingle, client phải load đoạn mã JavaScrip từ máy chủ để thiết lập kết nối XMPP qua WebSockets với máy chủ XMPP. Client sau đó sẽ map thông tin SDP offer và SDP answer được tạo ra

bởi trình duyệt thành thông điệp Jingle call setup và chuyển tiếp nó cho trình duyệt kia. Luồng thông điệp báo hiệu Jingle trong WebRTC được thể hiện ở hình dưới đây:



Hình 3.4: Báo hiệu Jingle over WebSockets cho WebRTC

✓*Sử dụng JSON over WebSockets*

Cả SIP và XMPP (với Jingle) là các giao thức báo hiệu, quản lý các phiên đa phương tiện dựa trên mô tả SDP (SIP sử dụng SDP như đã định nghĩa trong RFC 4566, XMPP sử dụng một phiên bản XML của định dạng SDP). Vì vậy cả hai giao thức SIP và XMPP có thể là một lựa chọn tốt, cũng như bất kỳ giao thức nào mang thông tin like- SDP. Tuy nhiên, báo hiệu trực quan nhất là truyền tải đối tượng JSON qua kênh hai chiều như là WebSockets. Hướng tiếp cận này được Google sử dụng và tương đối phổ biến hiện nay. JSON có thể coi là tập con cú pháp JavaScript nên có ưu điểm lớn khi thông dịch bởi trình duyệt web. Tất cả cấu trúc dữ liệu và thông tin trạng thái trong ứng dụng web được lưu trữ trong đối tượng đều được map rõ ràng, trực tiếp vào JSON nên không đòi hỏi nhiều nỗ lực cho việc mã hóa (encoding), phân tích (parsing) và xử lý (processing). Một thuận lợi khác của JSON là không áp đặt ngữ nghĩa lên ứng dụng, cho phép đầu cuối trao đổi bất kỳ loại thông tin nào như SDP blob cho thiết lập kết nối media, hoặc sự kiện cụ thể tùy theo logic ứng dụng. Ngoài ra do không bắt buộc sử dụng cách thức đặt tên (identity scheme) đặc biệt nào, nó cho phép các loại cơ chế định danh người dùng. Chẳng hạn nó cho phép định danh người dùng bởi username hoặc email hoặc định danh dịch vụ truyền thông sẵn có như số điện thoại, Skype name, và SIP/XMPP URIs.

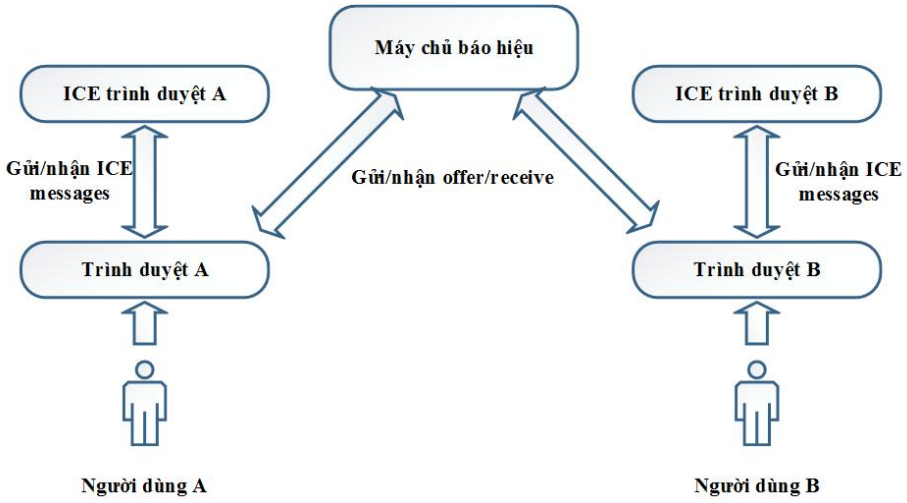
Sử dụng JSON cùng với thư viện đảm nhận việc thiết lập và duy trì kênh hai chiều tin cậy với máy máy chủ báo hiệu là khá đơn giản và hiệu quả, dù phát sinh thêm chi phí xây dựng cổng ứng dụng tùy biến (custom gateway) nếu muốn kết nối ứng dụng web với dịch vụ thông tin liên lạc bên ngoài.

3.4. Các quá trình trong báo hiệu

Có ba quá trình "**bán**" bất đồng bộ chính trong thiết lập session WebRTC Theo [23], bao gồm:

- **WebRTC Javascript callback logic:** các logic này handle tất cả những xử lý mức trình duyệt của WebRTC
- **STUN/TURN Sever Session Description Protocol (SDP) messaging:** Đây là logic báo hiệu diễn ra ngoài kết nối WebRTC để cài đặt kết nối P2P giữa 2 trình duyệt theo yêu cầu.
- **ICE (Interactive Connectivity Establishment) messaging:** quá trình hỗ trợ vượt NAT, chuyển tiếp (relay) media/data trong trường hợp cần thiết.

Các quá trình tạm gọi là bán đồng bộ vì trong chuỗi kết nối, luồng logic call back và luồng logic báo hiệu được kích hoạt lẫn nhau. Trong đó quá trình SDP, ICE nêu trên đều là một phần của báo hiệu, đều cần có máy chủ báo hiệu riêng để chuyển tiếp thông điệp SDP, hoặc để chuyển tiếp thông điệp ICE. Luận văn đi vào nghiên cứu quá trình báo hiệu với các thực thể là STUN Server phía người dùng A, người dùng A, trình duyệt người dùng A, máy chủ báo hiệu, trình duyệt người dùng B, người dùng B, và STUN Server người dùng B trong ứng dụng truyền nhận video giữa A và B.

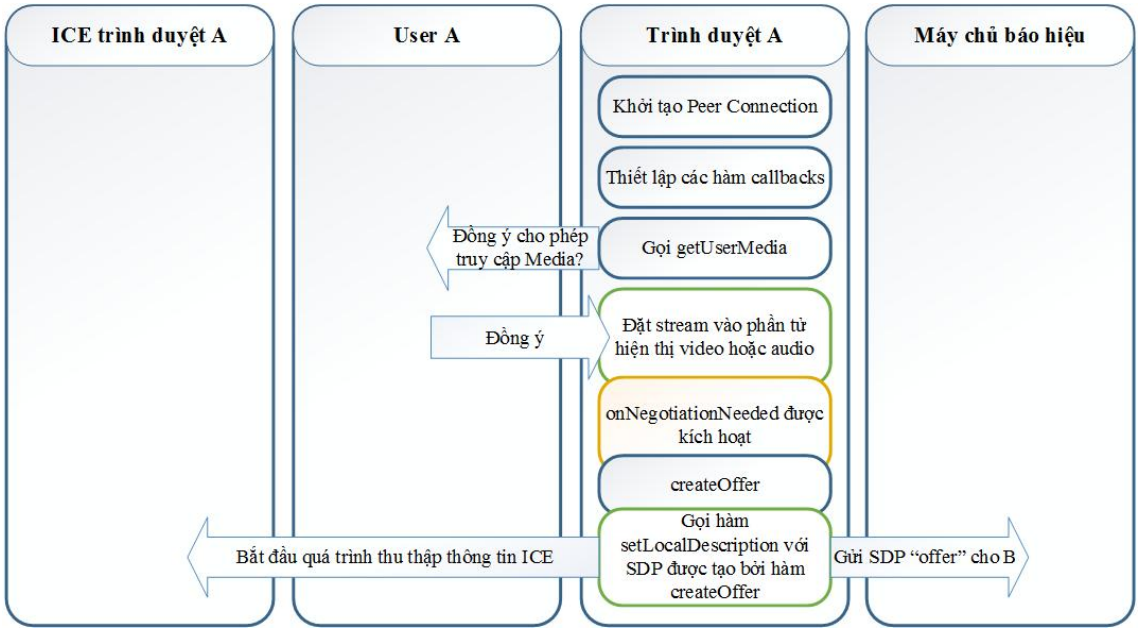


Hình 3.5 Các thực thể tham gia quá trình báo hiệu

✓ **Khởi động báo hiệu**

Có bốn bước để bắt đầu quá trình báo hiệu: khởi tạo đối tượng RTCPeerConnection, định nghĩa các hàm callback và gán vào đối tượng RTCPeerConnection, gọi hàsm getUserMedia để lấy stream video từ camera (đối với

các ứng dụng video), tạo thông điệp SDP bằng cách invoke các hàm onNegotiationNeeded, createOffer và setLocalDescription



Hình 3.6: Quá trình khởi tạo trong báo hiệu WebRTC

Bước 1: khởi tạo RTCPeerConnection: công việc chính của bước này là định nghĩa danh sách STUN Servers. Hiện nay Google cho phép truy cập đến các máy STUN free, thích hợp sử dụng cho các ứng dụng thử nghiệm.

Bước 2: Định nghĩa các hàm Callbacks và assign cho đối tượng RTCPeerConnection

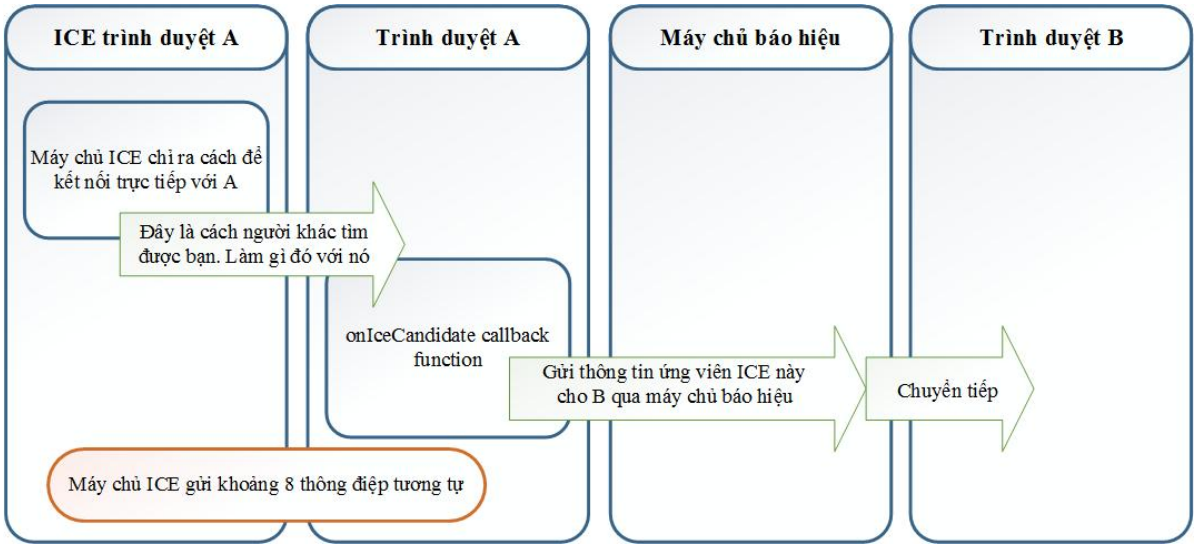
- Hàm onIceCandidate: xử lý phản hồi nhận được từ máy chủ STUN
- Hàm onNegotiationNeeded: Nếu có gì xảy ra yêu cầu phải đàm phán lại session, hàm này sẽ được gọi. Trong đa số các trường hợp, hàm được gọi khi click nút Allow, nút gắn với sự kiện mà hàm onNegotiationNeeded lắng nghe.
- Hàm onAddStream: được gọi thực hiện ngay ghi gọi hàm setRemoteDescription với thông tin SDP remote peer.

Bước 3: Gọi hàm getUserMedia để lấy dòng video. Hàm này gồm 3 tham số constraints, hàm callback success và hàm callback failure

Bước 4: Tạo thông điệp SDP với các hàm ***onNegotiationNeeded, createOffer*** *và* ***setLocalDescription***. Khi người dùng A click vào nút Allow, hàm callback onNegotiationNeeded được gọi và bắt đầu quá trình thống nhất. Quá trình thống nhất hoàn thành bằng cách gọi hàm createOffer. Hàm creatOffer tạo ra đối tượng SDP, có hàm callback sau khi thành công để xử lý đối tượng SDP vừa được tạo, trong đó việc chính là cho RTCPeerConnection biết tất cả thông tin mô tả local trong SDP. Việc này

được kết thúc bằng hàm setLocalDescription có được gắn hàm callback khi thành công. Trong hàm call back này sẽ thực hiện gửi thông tin SDP local (loại SDP "offer") cho peer xa qua máy chủ báo hiệu.

✓ **Đàm phán (Negotiation) ICE**



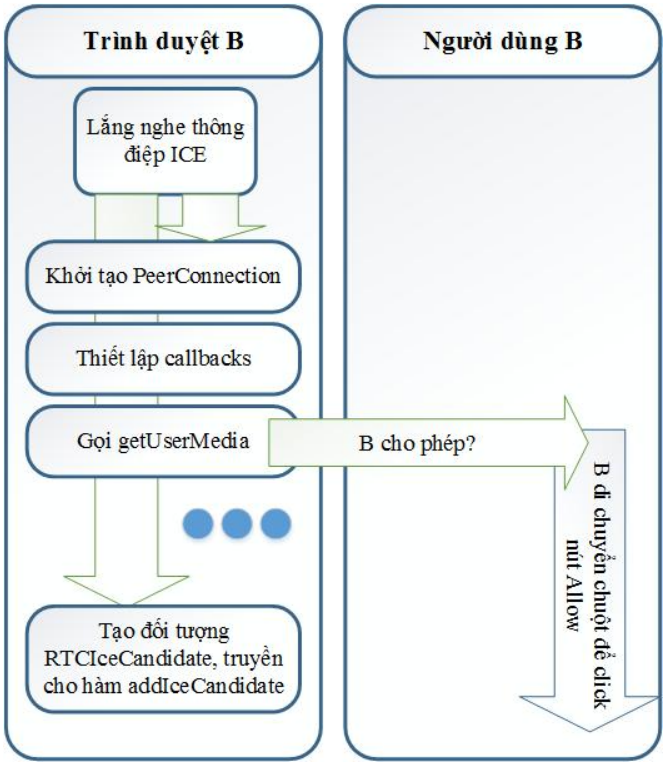
Hình 3.7 Quá trình ICE Negotiation trong báo hiệu WebRTC

Ngay sau khi hàm setLocalDescription được gọi, ICE negotiation bắt đầu thực hiện. Hàm setLocalDescription hỏi thông tin từ những máy chủ STUN được định nghĩa trong RTCPeerConnection ở bước 1 để tạo ra những ứng viên ICE được đóng gói trong đối tượng RTCIceCandidate. Máy chủ STUN sẽ trả lời thông tin một vài ứng viên ICE và gửi kết quả về trình duyệt. Ứng dụng xử lý thông tin những ứng viên được STUN server trả về bằng cách sử dụng hàm callback onIceCandidate. Hàm onIceCandidate có nhiệm vụ gửi đối tượng RTCIceCandidate cho peer xa qua máy chủ báo hiệu. Tại thời điểm này, người dùng A hoàn thành xong quá trình khởi tạo và đợi người dùng B gửi thông tin SDP và ICE Candidate của B.

✓ **Quá trình phía người dùng B khi nhận thông điệp ICE hoặc SDP**

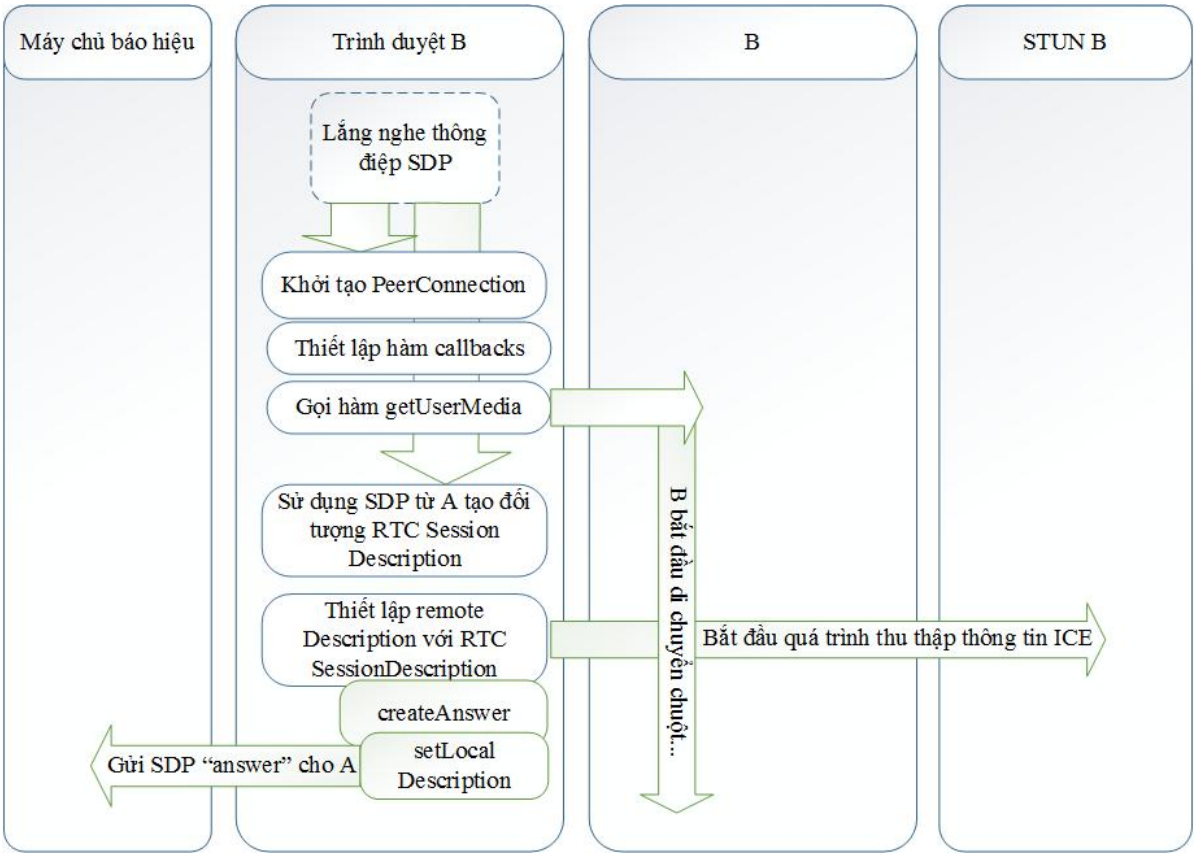
Trình duyệt người dùng B vẫn đang đâu đó đợi nhận các thông điệp. Ngay khi nó nhận được thông điệp ICE hoặc SDP từ trình duyệt A, nó kick-off tiến trình khởi động như khi người dùng A khởi động cuộc gọi đã nêu ở trên. Tại thời điểm này có thể có 2 quá trình bất đồng bộ xảy ra: một là khi B nhận được thông tin ứng viên ICE, hai là khi B nhận được thông điệp SDP.

Trường hợp B nhận được thông điệp ICE, nếu là ứng viên ICE thì B sẽ thêm ứng viên này vào đối tượng RTCPeerConnection qua hàm addIceCandidate. Từ thời điểm này thì B biết làm thế nào để kết nối đến được với A.



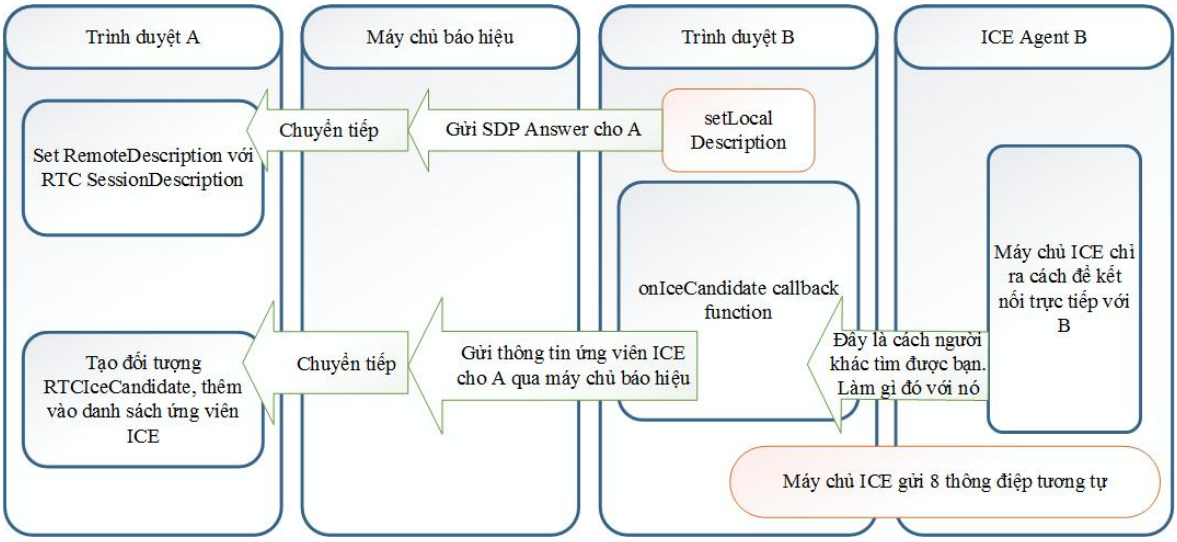
Hình 3.8: Quá trình xử lý thông điệp ICE phía người dùng xa

Trường hợp người dùng B nhận được thông điệp SDP, B sẽ cần báo cho RTCPeerConnection biết những thông tin về media của người dùng A được đóng gói trong SDP. B thực hiện việc này bằng cách tạo ra đối tượng RTCSessionDescription và truyền nó vào trong hàm setRemoteDescription. Hàm setRemoteDescription có định nghĩa hàm calback trong trường hợp thành công (hàm createAnswer), đầu tiên sẽ kiểm tra loại SDP có phải là "offer" không. Trong trường hợp đang xem xét thì SDP là "offer", B sẽ tạo ra SDP của chính mình bằng phương thức createAnswer. Phương thức createAnswer có hàm callback khi thành công, tạo ra SDP loại "answer", gọi đến hàm setLocalDescription với đối tượng RTCSessionDescription được truyền vào. Tương tự như đã nêu ở bước 4 trong quá trình khởi động báo hiệu, setLocalDescription bản thân có hàm callback khi thành công với mục đích gửi thông tin SDP đến người dùng A qua máy chủ báo hiệu



Hình 3.9: Quá trình xử lý thông điệp SDP phía người dùng xa

✓ **Phía người dùng A khi nhận thông điệp ICE hoặc phản hồi SDP**



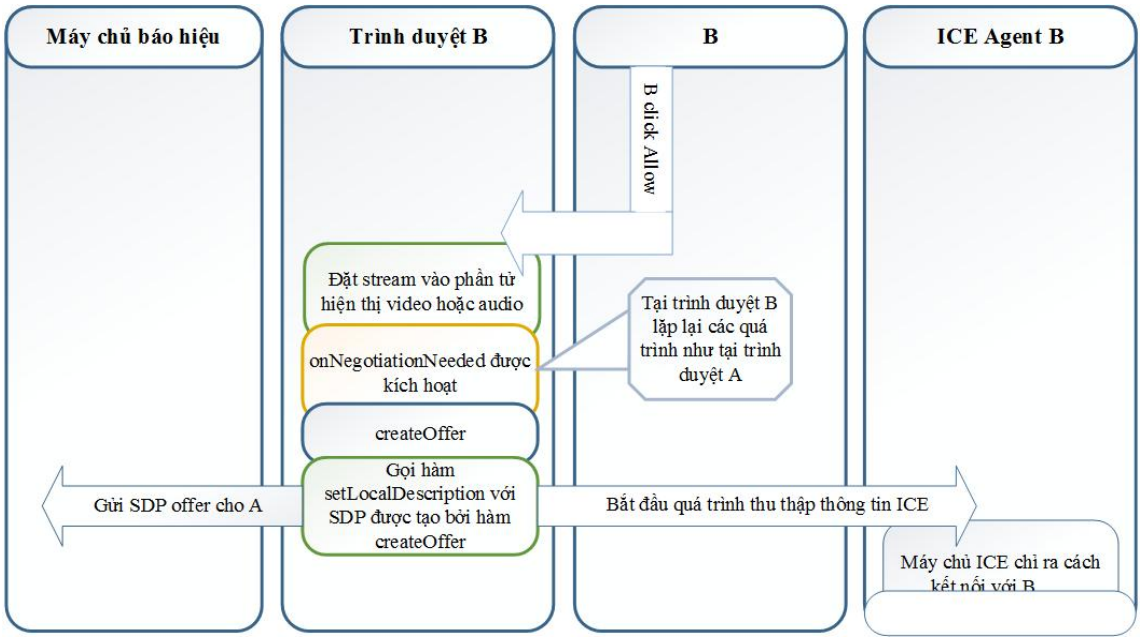
Hình 3.10: Quá trình xử lý thông điệp SDP, ICE khi nhận phản hồi từ người dùng xa trong báo hiệu WebRTC

Khi trình duyệt A nhận thông điệp SDP từ trình duyệt B, A sử dụng những dữ liệu này để tạo đối tượng RTCSessionDescription và truyền nó cho hàm

setRemoteDescription. Từ thời điểm này, A đã hiểu và có đủ thông tin về dữ liệu media của B (như các ràng buộc, mã hóa...) để thiết lập kết nối an toàn. Khác với B, do thông điệp SDP A nhận được là loại "answer" nên không có thêm tiến trình nào xử lý dữ liệu SDP này nữa.

Trình duyệt tại A xử lý tương tự như B khi nhận thông điệp ICE. Từ điểm này A đã biết cách làm sao để kết nối đến máy tính của B mà không cần sử dụng đến máy chủ của bên thứ 3 để relay dữ liệu. Lúc này quá trình ICE Negotiation kết thúc.

Đến đây thì B đã có thể thấy hình ảnh của người dùng A treaming đến, nhưng A thì chưa nhìn thấy B vì còn đợi B đồng ý (click vào nút Allow) như nêu ở trên.



Hình 3.11: Quá trình xử lý khi B đồng ý ứng dụng truy cập camera/microphone

Khi B click Allow để đồng ý, sự kiện RenegotiationNeeded được gọi, nó sẽ thực thi hàm callback onNegotiationNeeded đã được định nghĩa. Khi B và A có cùng code chạy trên trình duyệt, toàn bộ các quá trình nêu trên được kick-off, sự khác biệt chỉ là A sẽ "học" thông tin về B và B thì học thông tin về A.

Trên đây là toàn bộ quá trình báo hiệu trong WebRTC cho ứng dụng điển hình WebRTC có truyền tải video. Thực tế thì hàm getUserMedia có thể được gọi độc lập với tiến trình khởi động. Chẳng hạn khi ứng dụng có chức năng gọi riêng (với nút Call) và chức năng bật webcam riêng (nút Turn On Webcam). Khi đó khi người dùng click Call, ứng dụng có thể thiết lập RTCPeerConnection và hàm callback, gọi trực tiếp hàm createOffer chứ không cần qua hàm onNegotiationNeeded. Các quá trình còn lại thì tương tự như đã trình bày ở trên.

CHƯƠNG 4.ỨNG DỤNG WEBRTC CHO GIẢI PHÁP CỘNG TÁC VÀ CHIA SẺ DỮ LIỆU ĐA PHƯƠNG TIỆN TẠI TRUNG TÂM MVAS

Với công nghệ được tích hợp trong WebRTC, có thể ứng dụng WebRTC trong việc xây dựng những ứng dụng cộng tác, chia sẻ file thời gian thực. Chương 4 mô tả cách thức sử dụng các thư viện đã được phát triển ứng dụng WebRTC để xây dựng ứng dụng cộng tác, chia sẻ file tại Trung tâm MVAS - Mobifone

4.1. Thư viện WebRTC và các hướng tiếp cận

4.1.1. Các thư viện WebRTC

Để đơn giản hóa việc triển khai ứng dụng, cộng đồng mã nguồn mở/các vendor đã phát triển rất nhiều các thư viện, framework WebRTC JavaScript, một số tên phổ biến thể hiện ở bảng dưới đây:

Bảng 4.1: Thư viện WebRTC Javascript

Vendor	Địa chỉ link đến thư viện JavaScript
Addlive	http://www.addlive.com/platform-overview/
Apidaze	https://developers.apidaze.io/webrtc
Bistri	http://developers.bistri.com/webrtc-sdk/#js-sdk
Crocodile	https://www.crocodilertc.net/documentation/javascript/
EasyRTC	http://www.easyrtc.com/docs/
Janus	http://janus.conf.meetecho.com/docs/JS.html
JsSIP	http://jssip.net/documentation/
Openclove	http://developer.openclove.com/docs/read/ovxjs_api_doc
Oracle	http://docs.oracle.com/cd/E40972_01/doc.70/e49239/index.html
Peerjs	http://peerjs.com/docs/#api
Phono	http://phono.com/docs
Plivo	https://plivo.com/docs/sdk/web/
Pubnub	http://www.pubnub.com/docs/javascript/javascript-sdk.html
Quobis	https://quobis.atlassian.net/wiki/display/QoffeeSIP/API
SimpleWebRTC from &Yet	http://simplewebrtc.com/
SIPML5	http://sipml5.org/docgen/symbols/SIPml.html
TenHands	https://www.tenhands.net/developer/docs.htm
TokBox	http://tokbox.com/opentok
Twilio	http://www.twilio.com/client/api
Voximplant	http://voximplant.com/docs/references/websdk/

Vline	https://vline.com/developer/docs/vline.js/
Weemo	http://docs.weemo.com/js/
Xirsys	http://xirsys.com/_static_content/xirsys.com/docs/
Xsockets.net	http://xsockets.net/docs/javascript-client-api

Rõ ràng có rất nhiều sự lựa chọn thư viện trên nền WebRTC để phát triển ứng dụng WebRTC. Điểm chung của các thư viện, framework này là cung cấp tập các chức năng qua các APIs của nó, giúp các nhà phát triển JavaScript tích hợp WebRTC vào trang web một cách đơn giản nhất. Chức năng của đa số các thư viện WebRTC JavaScript thường là trừu tượng hóa những hoạt động phức tạp gắn với phần báo hiệu bằng những dịch vụ riêng biệt, và nó cũng thường quản lý các WebRTC API media engine của trình duyệt. Một vài cơ chế chung cho các thư viện WebRTC JavaScript được thể hiện như ở dưới đây:

Bảng 4.2. Cơ chế hoạt động chung các thư viện WebRTC Javascript

Cơ chế	Mô tả
Khởi động của thư viện	Hầu hết các thư viện WebRTC JavaScript có phương thức khởi tạo, tạo ra một instance của đối tượng JavaScript mà đại diện cho các dịch vụ được cung cấp bởi thư viện. Trong thời gian khởi tạo, các thư viện yêu cầu một số loại thông tin cấu hình cơ bản để được truyền vào cho thư viện JavaScript, chẳng hạn như thông tin người dùng cuối thường được đưa vào thư viện để đính kèm ID cho bất kỳ phiên WebRTC nào mà được thiết lập. Thông tin người dùng cuối này có thể được cung cấp bởi người sử dụng nhập, hoặc thông tin được lưu trữ trong các máy chủ web và được gửi trong khi tải khởi động trang.
Đăng ký	Sau khi thư viện được khởi tạo, chức năng đăng ký sẽ được thực hiện. Nó có thể phục vụ cho việc kết nối đến máy chủ báo hiệu ở đâu đó, cho dịch vụ biết rằng người dùng đã sẵn sàng nhận các thông điệp báo hiệu. Việc đăng ký cũng gồm một số thủ tục bảo mật như cung cấp sự xác thực của người dùng hoặc ứng dụng.
Tạo và quản lý phiên WebRTC	Tất cả các thư viện WebRTC JavaScript cung cấp phương thức tạo và quản lý phiên WebRTC. Trong đa số các trường hợp, thư viện sẽ xử lý phần truy cập vào camera và micro (nếu cần), và thiết lập PeerConnection cho việc truyền media, dữ liệu, đảm bảo các phương thức đơn giản, dễ hiểu. Các phương thức có thể khác nhau chút giữa các thư viện, nhưng mục đích tương tự nhau trong thực tế. Thông thường khi lời gọi phương thức được thực hiện, thông tin được chuyển đến thư viện để xác định ai là người được gọi, và vài sự kiện callback cũng được truyền. Phương thức này có thể cũng yêu cầu

	trao đổi một số thông tin về luồng phương tiện truyền thông audio/video sắp được thiết lập, đảm bảo chúng có thể được gắn vào phần tử audio/video thích hợp trong DOM. Sau khi session được thiết lập, thư viện hỗ trợ một tập những phương thức khác để thay đổi hoặc kết thúc session.
Sự kiện Callback	Có rất nhiều các sự kiện callbacks sử dụng có được thông qua các thư viện WebRTC JavaScript. Phần này cho phép các nhà phát triển ứng dụng chỉ dẫn các thư viện JavaScript như thế nào thì nên thông báo, hoặc "callback" JavaScript của nhà phát triển thư viện trong trường hợp xảy ra sự không đồng bộ trong nội tại của thư viện. Thư viện WebRTC có tập phong phú các hàm callback thì đưa ra những thông tin tốt hơn cho ứng dụng về trạng thái của session, cho phép ứng dụng báo cho người sử dụng hoặc ghi lại những gì đang diễn ra.

Ẩn trong các thư viện là các cơ chế gần giống nhau, giúp các nhà phát triển không phải quan tâm đến phần vận chuyển báo hiệu lớp dưới (HTTP, WebSocket, XMPP…), giao thức báo hiệu (SIP, XMPP, JSON,..) mà chỉ cần có kỹ năng lập trình Web với JavaScript. Vì vậy đa phần các thư viện WebRTC Javascript đều dễ sử dụng, chuyển từ thư viện này sang thư viện khác không tốn quá nhiều công sức.

Ngoài những hàm API theo cơ chế chung, nhiều thư viện cung cấp các hàm "nâng cao" hơn, thường đặc thù cho các dịch vụ mà thư viện hướng tới. Thư viện hướng tới dịch vụ hội nghị có thể cung cấp các hàm API xung quanh điều khiển hội nghị, trong khi thư viện hướng đến dịch vụ điện thoại có thể cung cấp các hàm bổ sung cho tin nhắn SMS hoặc RCS (rich communication service), hay thư viện hướng tới dịch vụ cộng tác có thể cung cấp các API bổ sung để chia sẻ màn hình và chuyển file. Phần khác biệt thật sự giữa các thư viện nằm ở các hàm nâng cao hướng dịch vụ cụ thể mà cung cấp bởi thư viện WebRTC JavaScript, còn cơ chế hoạt động các API ở mức cao cho việc khởi tạo, đăng ký, gọi/nhận cuộc gọi, event callbacks là khá tương đồng trong tất cả các thư viện

Dựa trên những thư viện đã và đang tiếp tục được hoàn thiện, đã có những ứng dụng hoàn chỉnh trên Internet cung cấp dịch vụ và giới thiệu WebRTC đến với người dùng, đặc biệt là các Web chuyên về chia sẻ file https:// www.sharefest.me, https://reep.io, https://www.sharedrop.io hay những Web hỗ chợ text/voice/video chat như https://appear.in, https://opentokrtc.com, https://helo.firefox.com, https://vline.com, https://talky.io, https://hubl.in/...

4.1.2. Các hướng tiếp cận sử dụng WebRTC

Có nhiều cách tiếp cận để xây dựng ứng dụng WebRTC phục vụ nhu cầu của doanh nghiệp/cá nhân, trong đó nhóm lại thành bốn hướng như sau:

- Chủ động tự xây dựng ứng dụng sử dụng WebRTC.

Đây là hướng tiếp cận thông dụng, có thể tham khảo hơn 2000 dự án WebRTC liên quan trên Github (GitHub là một dịch vụ lưu trữ dựa trên web cho các dự án phát triển phần mềm trong đó sử dụng các hệ thống kiểm soát phiên bản Git). Hướng này được chọn thường là dựa trên những kinh nghiệm phát triển VoIP trước đây của nhà phát triển, mà mong muốn có thể kiểm soát tất cả những khía cạnh của dịch vụ. Một lý do khác là cá nhân/doanh nghiệp đã phát triển hệ thống VoIP thành công, và mong muốn bổ sung điểm truy cập trong dịch vụ từ trình duyệt bằng cách đưa WebRTC vào đó. Theo hướng này, cần thực hiện các tác vụ:

- Định hướng và phát triển phần báo hiệu
- Xây dựng tất cả tính năng cần thiết cho WebRTC phía server như STUN, TURN, máy chủ báo hiệu, thống kê…
- Quản lý tính năng media phức tạp phía server.
- Xây dựng giao diện người dùng cho tất cả thiết bị và trình duyệt mong muốn hỗ trợ.
- Thường xuyên update dự án để hỗ trợ những thay đổi mới nhất của chuẩn WebRTC, kiểm thử và kiểm thử.

- Sử dụng dịch vụ của những nhà cung cấp dịch vụ đám mây SaaS

Để triển khai một dự án WebRTC, ta phải giải quyết các vấn đề kỹ thuật như vượt NAT, báo hiệu, hội nghị đa điểm, tracking chất lượng dịch vụ...Điều này có thể được giải quyết nhanh chóng bởi sự hỗ trợ từ các nhà cung cấp dịch vụ đám mây SaaS. Cụ thể hơn:

- Về vượt NAT: thay vì tự dựng máy chủ TURN, có thể sử dụng dịch vụ của đám mây XirSys hoặc VideoRoaming.
- Về báo hiệu: Có những nhà cung cấp dịch vụ nhắn tin độ trễ thấp, và chúng ta có thể được sử dụng cho báo hiệu WebRTC. Những nhà cung cấp đó có thể kể đến PubNub và Firebase.
- Về hội nghị đa điểm: Video đa điểm là một thách thức thực sự với WebRTC. Ta có thể sử dụng dịch vụ MCU1.com cho việc này.
- Về giám sát chất lượng: thay vì phải xây dựng một hoạt động giám sát toàn bộ chất lượng dịch vụ để hiểu những gì người dùng đang trải qua, có thể sử dụng dịch vụ CallStats.io và nhận được báo cáo trực tuyến trên các trải nghiệm người dùng.

- Lựa chọn nền tảng WebRTC API

Hướng tiếp cận khác là sử dụng các nền tảng WebRTC API. Nền tảng API là một tập bộ SDK (Software Development Kit) cho máy chủ, máy khách mà cung cấp mọi thứ cho lập trình viên phát triển dịch vụ WebRTC. Về phía máy chủ, tất cả các nền tảng API xử lý các chức năng cơ bản như báo hiệu giữa các bên, kết nối session, và dòng media giữa các topo mạng, vượt NAT. Một vài nền tảng API cung cấp cả các tính năng tiên tiến như hỗ trợ truyền thông đa điểm, recording, streaming, và hỗ trợ tích hợp bên

thứ ba trong quản lý định danh, cơ chế fallback trong tình huống WebRTC không được hỗ trợ và các khả năng khác. Bên cạnh các ưu điểm kể trên, các nền tảng WebRTC API có nhược điểm sau:

- Là giải pháp đóng: bạn có trong tay platform của nhà cung cấp bạn chọn, không thể thay đổi, và phải liên hệ với nhà cung cấp để giải quyết nếu có vấn đề gì đó xảy ra.
- Giá thành: API platform thường được tính giá dựa trên sử dụng, có thể cản trở khả năng doanh nghiệp trong cạnh tranh, lợi nhuận.
- Vendor lock-in: Không có chuẩn API hoặc service flow cho API Platform, sẽ mất nhiều công sức nếu muốn chuyển sử dụng từ API Platform này sang API Platform khác, bao gồm không chỉ APIs mà còn tính năng, luồng hoạt động.

Vì vậy, hướng tiếp cận này phù hợp với những đơn vị/cá nhân chỉ quan tâm đến kết quả cuối cùng của việc truyền thông video/voice/data như là một tính năng trong một chuỗi dịch vụ rộng lớn hơn, mà không phải là nghiệp vụ kinh doanh chính của họ, hoặc ít nhất là không cần phải sở hữu và kiểm soát giải pháp

- Hướng tiếp cận lai

Hướng tiếp cận này đề cập đến nhiều thành phần khác nhau, được chia theo hai nhóm: "Wrappers phía client + Máy chủ báo hiệu" và "Thành phần chức năng phía server"

- Wrappers phía client + Máy chủ báo hiệu: một Client Wrappers là một bộ SDK mà đóng gói phần WebRTC ở phía client và thường bao gồm cả máy chủ báo hiệu. Khi mà WebRTC APIs vẫn còn đang trong giai đoạn chuẩn hóa, có nhiều thay đổi, sự tương thích trình duyệt vẫn là vấn đề, việc có bộ đóng gói phía trên dịch vụ WebRTC giúp loại bỏ nhu cầu phải update ứng dụng Client WebRTC. Ví dụ cho các bộ SDK đó là PeerJS, EasyRTC, SimpleRTC, rtc.io. Các SDK khác nhau về chức năng, bảo trì, và sự linh hoạt mà nó cung cấp, vì vậy, trước khi sự lựa chọn cần đánh giá chúng dựa trên nhu cầu ứng dụng, các kế hoạch tương lai của SDK, và sự dễ dàng khi sử dụng. Cần đặc biệt chú ý đến các báo hiệu độc quyền đi kèm cùng với SDK và chắc chắn rằng nó đáp ứng nhu cầu ứng dụng. Việc thay đổi các báo hiệu là có thể, nhưng nó là trách nhiệm của các nhà phát triển khi nâng cấp lên phiên bản mới của SDK.
- Thành phần chức năng phía server: Đây là những thành phần chức năng cụ thể mà có thể được host trên đám mây hoặc tại chỗ (local). Chẳng hạn dịch vụ Twilio STUN/TURN và chức năng media server được cung cấp bởi Jitsi và Kurento.

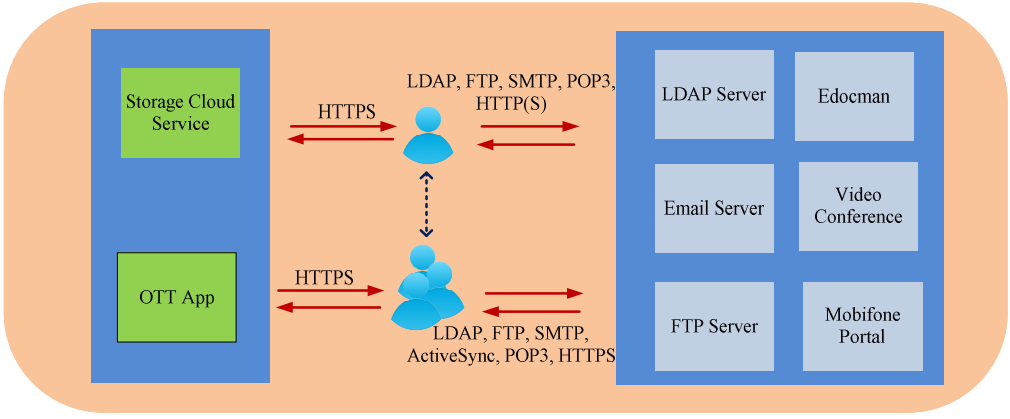
Hướng tiếp cận này phù hợp cho các nhà phát triển muốn kiểm soát toàn bộ giải pháp những không muốn xây dựng ứng dụng từ đầu.

4.2. Ứng dụng WebRTC thử nghiệm cho việc cộng tác, chia sẻ dữ liệu đa phương tiện tại Trung tâm MVAS - Mobifone

4.2.1. Hiện trạng cộng tác chia sẻ dữ liệu tại Mobifone

Tổng công ty viễn thông Mobifone được thành lập từ năm 1993, là doanh nghiệp đầu tiên của Việt Nam khai thác dịch vụ thông tin di động GSM 900/1800. Đến nay, sau hơn 20 năm phát triển, Mobifone đã mở rộng hoạt động trải dài ở tất cả các tỉnh thành trong cả nước. Về hạ tầng mạng CNTT, Mobifone đã phát triển hệ thống mạng nội bộ kết nối tất cả các khối văn phòng, các Trung tâm chức năng, các Công ty khu vực cũng như các chi nhánh, cửa hàng, trong đó node mạng Core đặt tại HN, HCM, Đà Nẵng. Các kết nối mạng từ tất cả các site đều được kết nối về các node mạng Core, thông mạng trong toàn Tổng Công ty. Hạ tầng mạng được đầu tư lớn như vậy để đảm bảo chất lượng dịch vụ cung cấp cho khách hàng, cũng như ứng dụng tốt nhất CNTT trong việc điều hành hoạt động sản xuất kinh doanh. Bên cạnh việc không ngừng nâng cao công nghệ mạng viễn thông (từ 2G lên 3G, và đã hoàn thành thử nghiệm mạng 4G) chất lượng dịch vụ phục vụ khách hàng, Mobifone không ngừng đầu tư phát triển, nâng cấp các hệ thống nghiệp vụ như hệ thống Tính cước & Quản lý khách hàng, hệ thống Đối soát cước, Hệ thống Kế toán, hệ thống Bán hàng, hệ thống Chăm sóc khách hàng... Về mặt quản trị, điều hành Mobifone đã đầu tư, xây dựng các hệ thống quản lý đầu tư, quản lý chi phí, quản lý văn bản, email, các hệ thống báo cáo, hỗ trợ ra quyết định, áp dụng phân tích BI, Big Data…

Đánh giá chung về hạ tầng CNTT của Mobifone là khá hiện đại, áp dụng các công nghệ tiên tiến, không ngừng nâng cao chất lượng để đáp ứng nhu cầu sản xuất kinh doanh. Tuy nhiên, vẫn có một phần nhỏ chưa được Mobifone quan tâm đúng mực, mà để các đơn vị (hơn 40 đơn vị) tự chủ động thực hiện: đó là việc cộng tác, chia sẻ dữ liệu giữa cá nhân, nhóm.



Hình 4.1: Mô hình cộng tác tại Mobifone

Do không phải đơn vị nào cũng có bộ phận CNTT đủ mạnh, cũng như được đầu tư về CNTT, việc chia sẻ dữ liệu chia sẻ dữ liệu, cộng tác nhóm hiện tại ở các đơn vị nói chung và ở Trung tâm MVAS còn gặp một số vướng mắc, tồn tại:

✓ *Về chia sẻ file, dữ liệu*:

Trung tâm MVAS vẫn sử dụng các hình thức chia sẻ file kiểu cũ với nhiều nhược điểm:

- Sử dụng chức năng chia sẻ network file sharing của Windows trên SMB Protocol (do các máy client chủ yếu sử dụng hệ điều hành Micrsoft Windows). Hình thức này người dùng cần có thao tác tạo folder, copy file muốn chia sẻ vào folder chia sẻ, gửi người cần chia sẻ link qua email. Hình thức này hỗ trợ khả năng phân quyền dựa trên Active Directory (do Mobifone đang sử dụng hệ thống các máy chủ Microsoft Windows Server 2012 với dịch vụ Active Directory để quản trị người dùng, nhóm người dùng). Chia sẻ này khá phù hợp với chia sẻ nhiều file hoặc folder, tuy nhiên cũng khá phức tạp trong thao tác đối với người dùng thông thường, chưa kể các vấn đề liên quan xử lý firewall từ máy PC hay thiết bị chuyên dụng.
- Sử dụng FTP server trong mạng nội bộ: hình thức này cũng được sử dụng khá phổ biến, cần cài đặt một FTP Server với dung lượng ổ cứng phù hợp hoặc kết nối SAN để chia sẻ dữ liệu giữa người dùng. Tương tự hình thức trên, người chia sẻ thường sẽ gửi mail về link chia sẻ cho người khác thông qua email. Thao tác là tương đối đơn giản cho người dùng cuối, nhưng có giới hạn về dung lượng có thể chia sẻ.
- Sử dụng hệ thống Email: người dùng có thể chia sẻ file qua email cho nhóm người mong muốn trong cộng tác. Dung lượng file chia sẻ qua email bị giới hạn khá thấp (20MB) để đảm bảo tải hệ thống Email (hiện cung cấp cho hơn 10.000 user, mỗi user được cấp giới hạn 500MB-2GB dung lượng mailbox).
- Sử dụng các công cụ đồng bộ trên private cloud (tương tự như Dropbox, OneDrive… nhưng sử dụng trong nội bộ): giải pháp này thực ra nghiêng về quản lý dữ liệu và đồng bộ, hỗ trợ đa nền tảng từ Web, App Desktop đến trên thiết bị di động nền tảng Android, IOS... Việc chia sẻ file cũng khá đơn giản, tương tự các thao tác như sử dụng Dropbox, GoogleDrive, OneDrive,..Điểm trừ là khi muốn chia sẻ, người dùng phải upload file lên không gian lưu trữ của mình trước, sau đó mới chia sẻ link. Dung lượng người dùng cũng có giới hạn nhất định, số lượng license cũng hữu hạn (Mobifone sử dụng giải pháp Aspera của IBM nhưng chỉ thử nghiệm một số license nhất định tại Văn phòng Tổng công ty).
- Sử dụng các ứng dụng OTT, P2P: đáp ứng tốt nhu cầu trao đổi, chia sẻ thời gian thực. Tuy nhiên, người dùng cần phải cài đặt thêm các ứng dụng này trên thiết bị đầu cuối.

✓ *Về hoạt động cộng tác, trao đổi thông tin*

Trong công việc, tại các doanh nghiệp chủ yếu là sử dụng các ứng dụng chat nội bộ hoặc dùng những ứng dụng phổ biến như Skype, Viber, Facebook Messenger, Zalo…Các ứng dụng này ngày càng được cải tiến đơn giản, dễ dùng và phù hợp với đa phần người dùng. Tuy nhiên, thường thì người dùng cần phải cài đặt ứng dụng trên đầu cuối. Ngoài ra, với sử dụng các ứng dụng này không đảm bảo an toàn bảo mật khi trao đổi file, thông tin qua các ứng dụng này, doanh nghiệp bắt buộc phải tin tưởng vào uy tín của các công ty phát triển phần mềm.

Với những ưu điểm được nêu ở Mục 2.1 như giảm giá thành, không plugins, truyền dữ liệu P2P, dễ sử dụng, một giải pháp cho nhiều nền tảng, tích hợp sẵn khả năng bảo mật… Để nâng cao hiệu quả chia sẻ thông tin, làm việc nhóm cũng như bảo mật, đề tài hướng đến việc sử dụng công nghệ WebRTC xây dựng ứng dụng Web đơn giản, dễ duy trì, giao diện thân thiện người dùng cuối ứng dụng trong Trung tâm dịch vụ Đa phương tiện và giá trị gia tăng Mobifone – Tổng Công ty viễn thông Mobifone (Trung tâm MVAS)

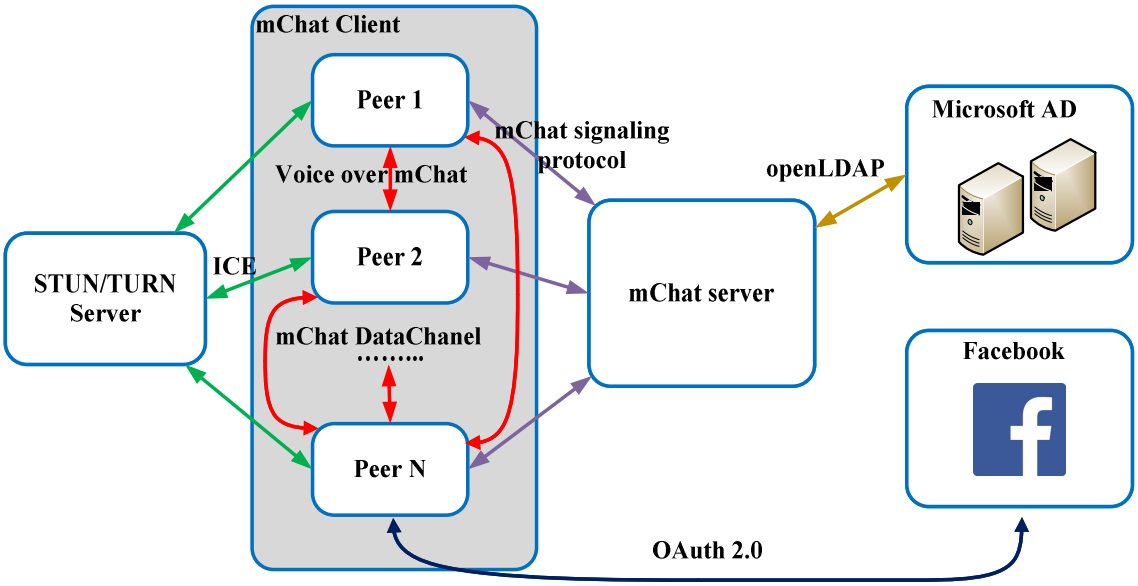
4.2.2. Yêu cầu hệ thống cộng tác tại Trung tâm MVAS – TCT viễn thông Mobifone

Hệ thống cộng tác tại Trung tâm MVAS Mobifone cần đảm bảo một số yêu cầu chính như sau:

- Hệ thống cho phép xác thực qua hệ thống Microsoft Active Directory của Mobifone (giao thức LDAP trên nền tảng Window Server 2012), hoặc xác thực qua tài khoản Facebook.
- Người dùng không cần cài đặt thêm phần mềm thứ 3, plugin mà chỉ cần dùng trình duyệt Chrome, Firefox, Opera để sử dụng các tính năng hệ thống... Người dùng có thể sử dụng các nền tảng hệ điều hành khác nhau như Windows, Mac OS X, Android, IOS…
- Dễ dàng cộng tác qua việc chat text, chat voice, chia sẻ file, và được thực hiện "peer to peer", không qua máy chủ trung gian.
- Hỗ trợ chia sẻ những file kích thước lớn.
- Hỗ trợ Voice chat P2P thời gian thực
- Có khả tăng tạo nhóm cộng tác, bảo mật bằng mật khẩu do người tạo thiết lập.
- Ngoài các yêu cầu trên, ứng dụng cung cần đảm bảo thông tin trao đổi cần được mã hóa, bảo mật, tránh thất thoát thông tin cho bên thứ ba.

4.2.3. Thiết kế kiến trúc hệ thống

Hệ thống cộng tác chia sẻ dữ liệu đa Phương tiện tại Trung tâm MVAS, gọi là mChat, được thiết kế với những khối chức năng chính là mChat Server, mChat client như hình dưới đây:



Hình 4.2: Kiến trúc hệ thống cộng tác và chia sẻ dữ liệu mChat

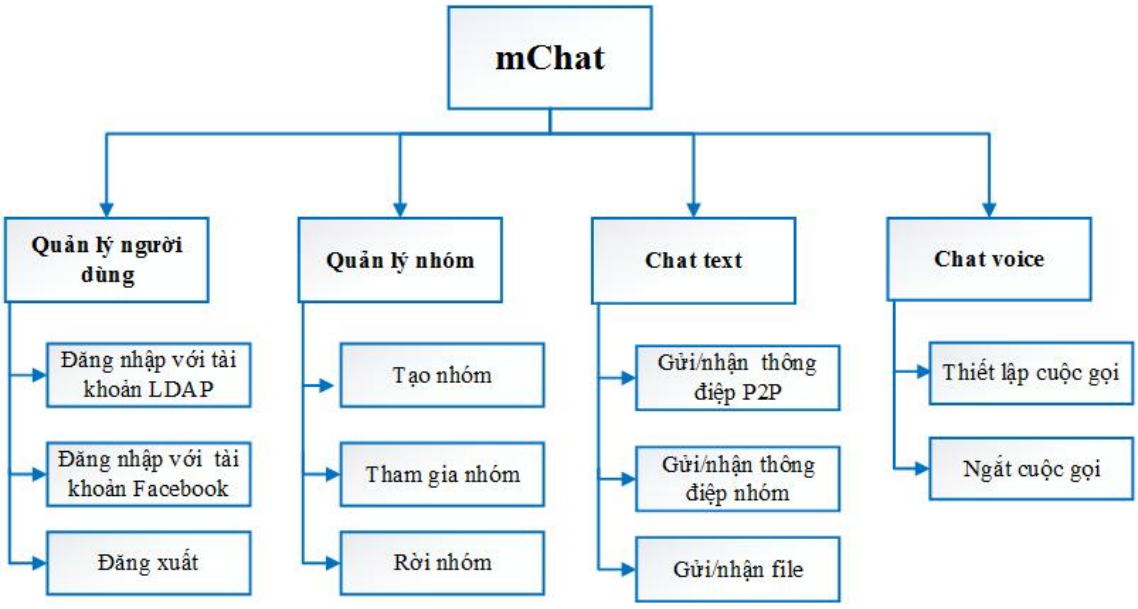
Các khối chức năng chính trong hệ thống cộng tác mChat bao gồm:

- Khối máy chủ mChat server là thành phần trung tâm, xử lý các phần logic của hệ thống cộng tác. mChat server đóng vai trò hai vai trò:
  - ✓ Vai trò thứ nhất và cũng là quan trọng nhất, là vai trò máy chủ báo hiệu cho hệ thống mChat. Khối chức năng này giao tiếp với mChat client qua giao thức riêng của ứng dụng, gọi là mChat Signaling Protocol. mChat Signaling Protocol hoạt động trên giao thức WebSocket, đảm bảo gửi/nhận các thông điệp với mChat client theo những thông tin mà mChat quy định. Các thông điệp trong mChat Protocol lưu trong các JavaScrip Object, được serialized dưới dạng chuỗi (string) để gửi qua WebSockets.
  - ✓ Vai trò thứ hai mChat server xử lý xác thực tài khoản người dùng qua hệ thống Active Directory trong nội bộ Mobifone hoặc qua Facebook. mChat sử dụng OpenLDAP để giao tiếp với máy chủ Microsoft Active Directory (AD).
  - ✓ Khối máy chủ STUN/TURN Server: hỗ trợ các mChat client tìm kiếm Peer, vượt NAT, tường lửa (nếu có). Khối này sử dụng ICE Protocol để trao đổi thông điệp với mChat client. Trong phạm vi của luận văn sử dụng dịch vụ của Google qua địa chỉ stun.l.google.com:19302
- Máy chủ đóng vai trò cung cấp thông tin để xác thực tài khoản (Identity Provider): gồm có máy chủ Microsoft AD tại Mobifone, Facebook. Xác thực qua AD là hướng tiếp cận hiện tại yêu cầu cho tất cả ứng dụng tại

Mobifone, đảm bảo quản lý tập trung user. Xác thực qua Facebook đảm bảo tương tác cho cả những đối tác của Mobifone, với điều kiện được khai báo trong phần cấu hình ứng dụng mChat trên trang facebook dành cho nhà phát triển.

- mChat client: là các trình duyệt Web hỗ trợ WebRTC (Chrome, Firefox, Opera), sử dụng WebSockets để gửi/nhận thông tin báo hiệu, sử dụng ICE để tìm kiếm peer và thiết lập connection P2P. Sau khi thiết lập phiên P2P giữa 2 mChat client, theo thiết kế của WebRTC, mChat client sử dụng các giao thức trên nền các giao thức SRTP đối với chức năng voice chat, tạm gọi là Voive over mChat Protocol và trên nền SCTP đối với truyền/nhận dữ liệu khác, tạm gọi là mChat DataChannel Protocol. mChat client giao tiếp với Facebook sử dụng OAuth 2.0 [28] trên HTTP để xác thực khi truy cập ứng dụng bằng tài khoản Facebook.

4.2.4. Phân tích chức năng người dùng



Hình 4.3: Biểu đồ phân rã chức năng người dùng hệ thống

Dưới đây là mô ta các chức năng chính của hệ thống

- Chức năng đăng nhập xác thực qua tài khoản LDAP**:** Người dùng có thể sử dụng tài khoản email của Mobifone để đăng nhập vào hệ thống mChat. Việc đăng nhập đảm bảo ứng dụng có thông tin cần thiết phục vụ cho việc cộng tác về sau. Thông tin người dùng sẽ được hiển thị dựa trên Display Name lấy từ hệ thống Active Directory của Mobifone, giúp người dùng dễ dàng lựa chọn đối tượng để cộng tác.

- Chức năng xác thực qua tài khoản Facebook**:** Người dùng có thể sử dụng tài khoản facebooke để đăng nhập vào hệ thống mChat. Chức năng này hữu ích trong 1 số trường hợp cần trao đổi giữa nhân viên Mobifone với các đối tác đến và làm việc tại Mobifone mà chưa có tài khoản email. Tương tự như việc đăng nhập qua tài khoản LDAP, các đối tác sau khi đăng nhập bằng Facebook sẽ hiển thị tên trên hệ thống cộng tác. Việc phân biệt với cán bộ Mobifone với đối tác thể hiện bằng hình ảnh Profile của người đăng nhập. Cán bộ Mobifone hệ thống dùng hình ảnh mặc định, với đối tác là hình ảnh lấy từ Profile Facebook của đối tác.

Chức năng tạo nhóm**:** chức năng này cho phép người dùng tạo ra những room hay nhóm chat riêng để trao đổi thông tin. Sau khi tạo thành công nhóm và mật khẩu để vào nhóm, người dùng sẽ gửi thông tin tên nhóm và mật khẩu cho những người dùng khác có thể tham gia nhóm

Chức năng tham gia nhóm: Người dùng truy cập phần chat nhóm, lựa chọn nhóm, nhập mật khẩu để tham gia nhóm.

Chức năng rời nhóm: Người dùng có thể chủ động rời nhóm trong trường hợp thấy không cần thiết. Trường hợp có vấn đề kết nối mạng, hệ thống tự động loại bỏ người dùng khỏi nhóm, đảm bảo người dùng trong nhóm luôn nhìn thấy thành phần online thời gian thực.

Chức năng gửi/trả lời thông điệp text P2P: cho phép người dùng gửi nhận, trao đổi thông tin text. Thông tin text được truyền trực tiếp giữa 2 mChat client không qua máy chủ trung gian.

Chức năng gửi/nhận file: cho phép người dùng có thể chia sẻ file mọi định dạng với nhau. Việc truyền nhận file dưới dạng P2P trực tiếp giữa 2 mChat client, không qua máy chủ trung gian.

Chức năng gửi/nhận thông điệp nhóm: cho phép người dùng gửi nhận thông tin text đến tất cả những người khác tham gia vào nhóm.

Chức năng goi/trả lời: cho phép người dùng thiết lập cuộc gọi audio P2P để trao đổi thông tin.

4.2.5. Phân tích luồng các sự kiện chính

4.2.5.1. Đăng nhập LDAP

Tên Use Case	Đăng nhập
Tác nhân	Người dùng mChat
Sự kiện kích hoạt	Người dùng click vào nút đăng nhập
Luồng sự kiện chính: 1. Người dùng nhập thông tin username và mật khẩu Email trên trang login. 2. Nhấn nút Log in 3. Đăng nhập thành công, hệ thống redirect sang trang chủ hệ thống mChat	

Luồng sự kiện phụ: 1. Đăng nhập thất bại: yêu cầu đăng nhập lại

4.2.5.2. Đăng nhập Facebook

Tên Use Case	Đăng nhập bằng tài khoản Facebook
Tác nhân	Người dùng mChat
Sự kiện kích hoạt	Người dùng click vào nút đăng nhập
Luồng sự kiện chính: 1. Người dùng click ảnh Facebook trên trang login 2. Hệ thống redirect đến trang xác thực của Facebook. 3. Người dùng nhập thông tin tài khoản, mật khẩu Facebook, ấn nút Log In 4. Dịch vụ của Facebook thực hiện đăng nhập 5. Đăng nhập thành công, hệ thống redirect sang trang chủ hệ thống mChat	
Luồng sự kiện phụ: 1. Đăng nhập thất bại: yêu cầu đăng nhập lại 2. Đăng nhập thành công lần đầu, Facebook hiển thị permission form xin phép đồng ý người dùng để ứng dụng mChat lấy thông tin Profile người dùng. Người dùng chọn đồng ý.	

4.2.5.3. Tạo nhóm

Tên Use Case	Tạo nhóm chat riêng
Tác nhân	Người dùng mChat
Sự kiện kích hoạt	Người dùng click chọn nút Create trong giao diện tạo nhóm
Luồng sự kiện chính: 1. Người dùng vào giao diện group collaboration, chọn tab Create Group 2. Người dùng nhập tên nhóm và đặt mật khẩu cho nhóm, click Create 3. Tạo nhóm mới thành công, người tạo mặc định ở trong nhóm. Nhóm mới sẽ được thêm vào danh sách trong giao diện Join Group để người dùng khác có thể tham gia nhóm	
Luồng sự kiện phụ: 1. Tạo nhóm thất bại: quay lại giao diện ban đầu, hiển thị nguyên nhân lỗi	

4.2.5.4. Tham gia nhóm

Tên Use Case	Tham gia nhóm chat riêng
Tác nhân	Người dùng mChat
Sự kiện kích hoạt	Người dùng click chọn nút Join trong giao diện tham gia nhóm
Luồng sự kiện chính: 1. Người dùng vào giao diện group collaboration, chọn tab Join Group 2. Người dùng nhập tên nhóm và mật khẩu của nhóm, click Join 3. Join nhóm mới thành công, người dùng được thêm vào nhóm để cộng tác. Tên các nhóm mà người dùng tham gia được thể hiện ngay trên giao diện. 4. Danh sách	
Luồng sự kiện phụ: 2. Tham gia nhóm thất bại: quay lại giao diện tham gia nhóm, hiển thị nguyên nhân lỗi	

4.2.5.5. Rời nhóm

Tên Use Case	Tham gia nhóm chat riêng

Tác nhân	Người dùng mChat
Sự kiện kích hoạt	Người dùng click link Leave Room ở bên dưới tên nhóm mà người dùng tham gia
Luồng sự kiện chính: 1. Người dùng vào giao diện group collaboration, chọn tab Join Group 2. Người dùng nhập tên nhóm và mật khẩu của nhóm, click Join 3. Join nhóm mới thành công, người dùng được thêm vào nhóm để cộng tác. Tên các nhóm mà người dùng tham gia được thể hiện ngay trên giao diện.	
Luồng sự kiện phụ: 1. Tham gia nhóm thất bại: quay lại giao diện tham gia nhóm, hiển thị nguyên nhân lỗi	

4.2.5.6. Gửi nhận thông điệp text P2P

Tên Use Case	Chat text
Tác nhân	Người dùng mChat
Sự kiện kích hoạt	Người dùng click nút Send trong giao diện chính
Luồng sự kiện chính: 1. Người dùng vào giao diện chính, click chọn người dùng cần cộng tác theo danh sách bên trái 2. Nhập thông tin cần gửi vào textbox, click nút Send để gửi thông điệp 3. Hệ thống hiển thị thông điệp đã gửi vào khung chat ở phía trên phía người gửi và phía người nhận.	

4.2.5.7. Gửi nhận thông điệp nhóm

Tên Use Case	Group chat
Tác nhân	Người dùng mChat
Sự kiện kích hoạt	Người dùng click nút Send trong giao diện Group Collaboration
Luồng sự kiện chính: 1. Người dùng vào giao diện Group Collaboration, chọn tab Join Group, click Join nhóm chat mà mình mong muốn và được tham gia. 2. Nhập thông tin cần gửi vào textbox, click nút Send để gửi thông điệp 3. Hệ thống hiển thị thông điệp đã gửi vào khung chat ở phía trên phía người gửi và khung chat của tất cả người dùng đang online trong nhóm.	

4.2.5.8. Chia sẻ File

Tên Use Case	Chia sẻ file
Tác nhân	Người dùng mChat
Sự kiện kích hoạt	Người dùng kéo thả file vào giao diện chat
Luồng sự kiện chính: 1. Người dùng click chọn người cần trao đổi file, kéo thả file vào giao diện chat 2. Phía người dùng đầu xa nhận được thông báo, click chọn đồng ý nhận file 3. Hệ thống gửi file cho người nhận	

4.2.5.9. Thiết lập cuộc gọi

Tên Use Case	Thiết lập cuộc gọi
Tác nhân	Người dùng mChat
Sự kiện kích hoạt	Người dùng click nút Call trong giao diện Voice Chat

Luồng sự kiện chính: 1. Người dùng vào giao diện Voice Chat, click Connect để kết nối vào room những người dùng sẵn sàng voice chat 2. Lựa chọn đối tượng để thiết lập cuộc gọi trong danh sách, click chọn nút Call 3. Hệ thống có thông báo đến đối tượng nhận cuộc gọi. Đối tượng nhận cuộc gọi Click đồng ý thiết lập cuộc gọi 4. Hoàn thành việc thiết lập cuộc gọi, hai đầu có thể trao đổi thông tin

4.2.5.10.Ngắt cuộc gọi

Tên Use Case	Ngắt cuộc gọi
Tác nhân	Người dùng mChat
Sự kiện kích hoạt	Người dùng click nút Hangup trong giao diện Voice Chat
Luồng sự kiện chính: 1. Người dùng click Hangup khi đang trao đổi voice chat. 2. Hệ thống ngắt cuộc gọi. 3. Hệ thống enable nút gọi, người dùng có thể thực hiện cuộc gọi với người khác	

4.2.6. Phát triển ứng dụng

4.2.6.1. Lựa chọn thư viện

Để triển khai hệ thống cộng tác với những chức năng nêu ở mục 4.2.5, luận văn nghiên cứu sử dụng framework EasyRTC do Priologic xây dựng. Lý do lựa chọn hướng tiếp này để đảm bảo cán bộ IT Trung tâm MVAS có thể quản lý toàn bộ giải pháp, không phải viết lại code từ đầu mà tận dụng thư viện có sẵn EasyRTC, không tiêu tốn chi phí thuê API Service ngoài do nhu cầu tương tác nội bộ. Việc lựa chọn EasyRTC vì các lý do chính sau:

- EasyRTC là bộ công cụ mã nguồn mở full-stack WebRTC, gồm tập hợp các ứng dụng web, các đoạn mã, thư viện phía client và các thành phần máy chủ được tài liệu hóa chi tiết, tỉ mỉ, dễ tiếp cận.
- EasyRTC cung cấp cho cán bộ IT Trung tâm MVAS không chỉ bộ công cụ phát triển, mà có sẵn những ứng dụng đơn giản, chỉ cần bổ sung không nhiều đoạn mã là có thể ứng dụng được ngay.
- Các WebRTC APIs trong trình duyệt Chrome, Firefox, Opera vẫn tiếp tục thay đổi, phát triển với tốc độ nhanh, các đoạn mã viết ra với phiên bản này có thể không chạy với phiên bản khác. EasyRTC hỗ trợ ẩn những phần thay đổi đó bằng cách cung cấp những APIs phía client dễ sử dụng và hạn chế tối đa ảnh hưởng bởi các WebRTC APIs nhất. EasyRTC Client API cũng hỗ trợ ẩn những phần khác nhau giữa các trình duyệt, giúp cán bộ IT Trung tâm chỉ cần tập trung vào ứng dụng.
- EasyRTC hỗ trợ fail-over tự động sang web-socket trong trường hợp trình duyệt không hỗ trợ Data Channel hoặc có lỗi trong vấn đề thiết lập Peer Connection (chẳng hạn do vấn đề NAT, Firewall).
- EasyRTC tích hợp dễ dàng với hệ thống xác thực hiện có của Trung tâm.

- EasyRTC đã giành được giải thưởng "Best WebRTC Tools Award: tại hội nghị triển lãm WebRTC Expo and Conference vào tháng 11 năm 2012 và Tawk.com trên EasyRTC giành được giải thưởng "Best All Around Award" vào tháng 8 năm 2013 tại Hội nghị triển lãm WebRTC tại Atlanta.

  EasyRTC được tổ chức gồm có:

  - Thư mục / (root): chứa thông tin về các gói cũng như license.
  - Thư mục API, chứa những file API EasyRTC xử lý các sự kiện từ phía client, đóng vai trò như một Controller xử lý các message từ phía client.
  - Thư mục thư viện (lib): chứa những thư viện cần thiết phía server xử lý các vấn đề liên quan đến báo hiệu.

Về phía nhà phát triển ứng dụng thì cần quan tâm đến các chức năng, phương thức được cung cấp, hỗ trợ trong các EasyRTC API như sau:

✓ **Browser EasyRTC API**

Easyrtc.js:

Là đối tượng quan trọng nhất, chứa các thuộc tính, phương thức chính cho việc viết ứng dụng bao gồm:

- Phương thức kết nối đến máy chủ báo hiệu *<static> connect (applicationName, successCallback, errorCallback).* Cần kết nối đến máy chủ báo hiệu trước khi call đến người dùng khác.
- Khởi động cuộc gọi đến người khác, phương thức *call(otherUser, callSuccessCB, callFailureCB, wasAcceptedCB)*
- Phương thức ngắt kết nối đến máy chủ báo hiệu *<static> disconnect()*
- Phương thức ngắt kết nối với người dùng khác hoặc tất cả, phương thức *<static> hangup(otherUser) hoặc <static> hangupAll()*
- Đặt listener cho thông điệp nhận từ máy chủ: *<static> setServerListener(listener)*
- Gửi tin nhắn P2P, phương thức *<static> sendDataP2P(destUser, msgType, msgData)*. Phương thức gửi dữ liệu đến người dùng khác qua kênh đã được thiết lập. Phương thức sẽ không thành công nếu chưa có kênh dữ liệu (data channel) nào được thiết lập giữa người dùng. Độ lớn dữ liệu có thể gửi chỉ phụ thuộc vào trình duyệt.
- Gửi tin nhắn đến ứng dụng trong máy chủ, phương thức sendServerMessage *<static> sendServerMessage(msgType, msgData, successCB, failureCB)*
- Các phương thức hỗ trợ quản lý room: *getRoomList, getRoomOccupantsAsArray, getRoomsJoined, joinRoom, leaveRoom*
- Các phần liên quan đến xử lý lỗi (error handling) [22]

Easyrtc_ft.js :

Chứa các phương thức để làm việc với file (truyền nhận file) giữa các peer. Các phương thức chính:

- Kích hoạt kênh truyền để nhận file, phương thức *<static> buildFileReceiver(acceptRejectCB, blobAcceptor, statusCB).*
- Gửi file đến cho peer, phương thức: *<static> buildFileSender(destUser, progressListener)*

✓ **Server API**

- Pub Object: đối tượng public, được trả về từ hàm listen() trong EasyRTC. Pub chứa tất cả các phương thức public cho việc tương tác với máy chủ EasyRTC.
- Application Object (appObj) phục vụ quản lý ứng dụng, có các phương thức tạo kết nối, phiên, tạo và xóa room trong ứng dụng.
- Connection Object (connectionObj) phục vụ quản lý các kết nối đến room, có các phương thức tạo room, quản lý user đang kết nối đến room nào.
- Connection Room Object (connectionRoomObj): Quản lý link giữa một kết nối tới một room.
- Session Object (sessionObj): làm việc với EasyRTC session, quản lý phiên, thiết lập giá trị biến Session, danh sách các kết nối hiện tại.
- Room object: Quản lý một room, thiết lập các lựa chọn cho room, chứa các phương thức quản lý room bao gồm cả xác định các kết nối đã gia nhập room
- Utility Function Object (util): những chức năng tiện ích như copy object, quản lý log…được nhóm trong đối tượng util, và được khai báo (attach) trong các lớp application, connection, session và room.
- Events Object (events): chứa các phương thức tương tác với các sự kiện EasyRTC, được khai báo trong các lớp application, connection, session và room
- Default Events (eventListener): là những event listeners mặc định được sử dụng bởi EasyRTC như onAuthenticate, onAuthenticated, onDisconnect, onConnection, onEasyrtcAuth, onEasyrtcCmd, onEasyrtcMsg, onEmitEasyrtcCmd, onEmitEasyrtcMsg, onGetIceConfig, onMsgTypeRoomJoin…một số trong đó có thể được override.

4.2.6.2. Môi trường phát triển

- Hệ điều hành: Windows 10 Pro, công cụ lập trình Web Storm 2016.2.3;

- Thiết lập môi trường phát triển NodeJS, tích hợp module passport hỗ trợ xác thực qua tài khoản LDAP, Facebook; module express-session phục vụ quản lý phiên từ passport.
- Máy chủ báo hiệu được xây dựng sử dụng module socket.io trên NodeJS, phục vụ gửi/nhận thông tin giữa server và client trình duyệt.
- Các API EasyRTC trong các file: easyrtc.js, easyrtc_int.js, easyrtc_ft.js, easy_app.js, adapter.js;
- LDAP server.

4.2.6.3. Thiết kế modules

Hệ thống mChat cộng tác chia sẻ dữ liệu đa Phương tiện tại Trung tâm MVAS Mobifone được phát triển theo mô hình MVC:

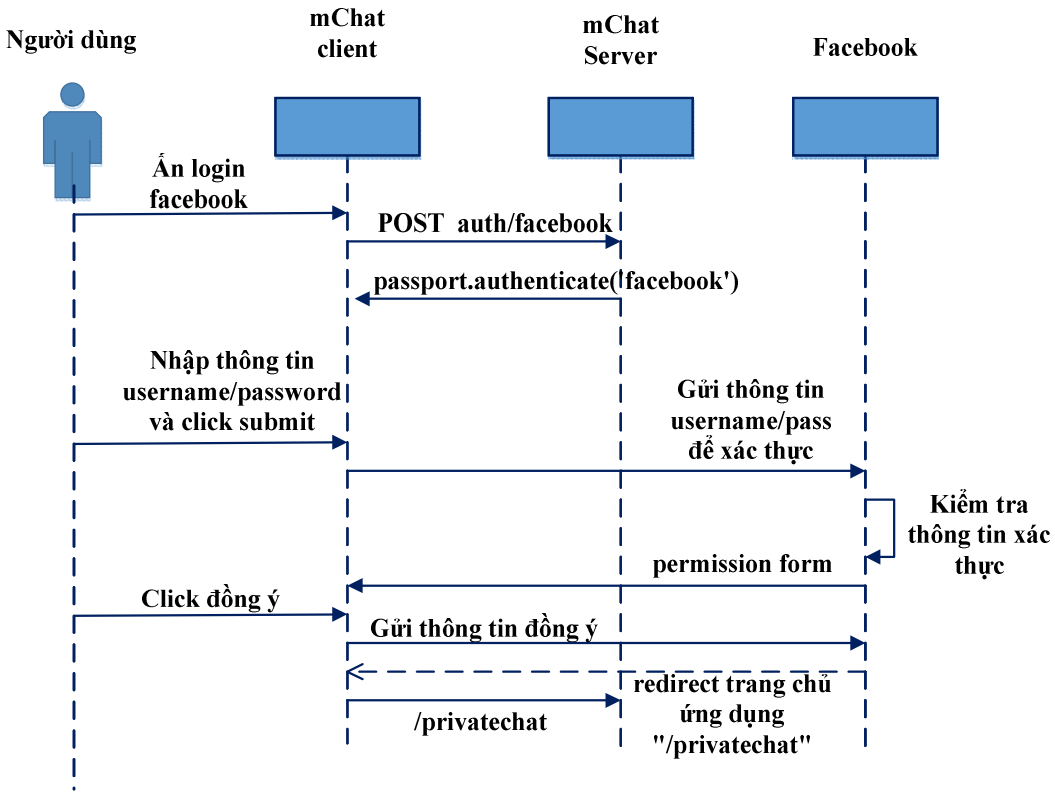
- Lớp giao diện người dùng: cung cấp các thành phần giúp người dùng tương tác với hệ thống, hiển thị các thông tin cần thiết đối với người dùng để cộng tác (chat, gọi audio, chia sẻ file, tạo nhóm chat…).
- Lớp xử lý logic: kiểm tra thông tin tài khoản xác thực, thực hiện các nhiệm vụ của kênh báo hiệu, xử lý các hành động tương tác của người dùng trong quá trình cộng tác đảm bảo các chức năng hệ thống theo yêu cầu.
- Lớp dữ liệu: quản lý thông tin người dùng thời gian thực như id, username, phiên kết nối P2P, thông tin nhóm làm việc thời gian thực, thông tin trao đổi.

Để thực hiện các yêu cầu về cộng tác và chia sẻ dữ liệu tại Trung tâm MVAS như mục 4.2.2, ứng dụng mChat được xây dựng dựa trên những modules chính sau:

- Module xác thực:

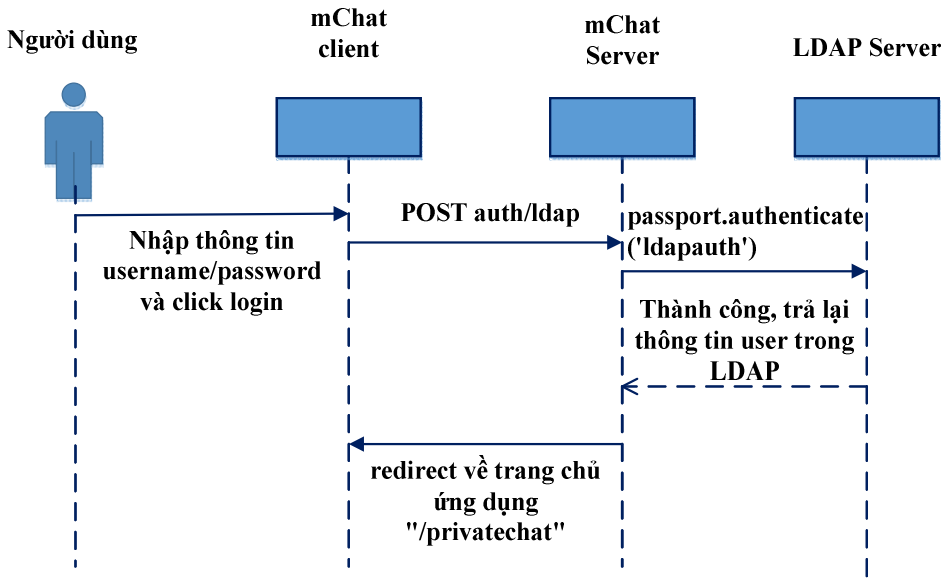
Sử dụng passport, một middleware cho việc xác thực của Node.js, cụ thể theo yêu cầu của Mobifone là dùng module passport cho Facebook (Facebook Strategy) và module passport cho LDAP (LDAP Strategy).

Module xác thực qua Facebook xác thực người dùng sử dụng tài khoản Facebook và thẻ OAuth 2.0 (OAth 2.0 tokens)



Hình 4.4. Biểu đồ tuần tự quá trình xác thực bằng tài khoản Facebook

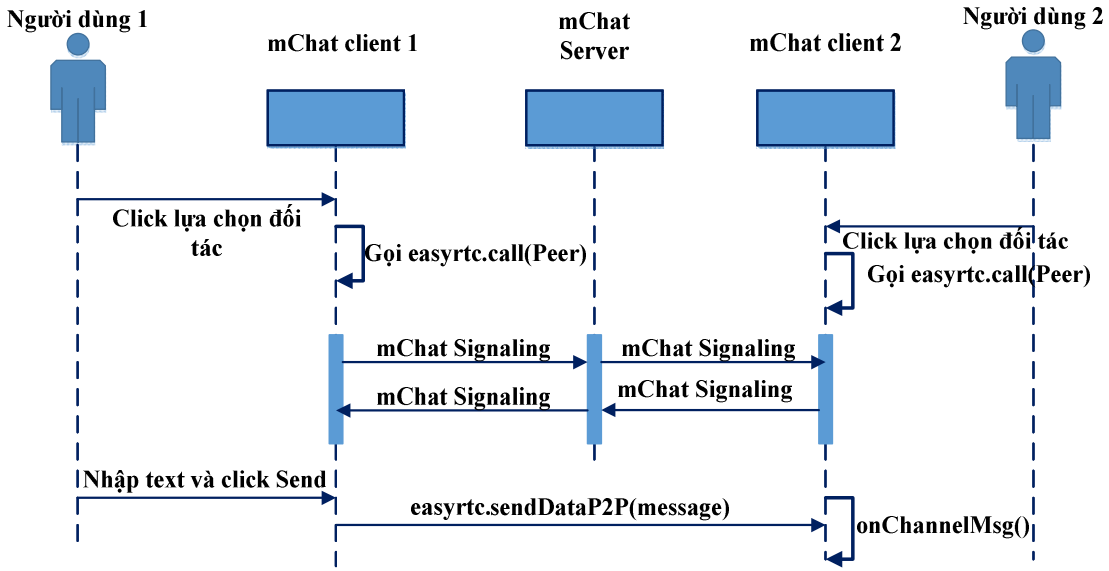
Module xác thực qua LDAP sử dụng giao thức openldap, cho phép người dùng truy cập được vào trang chủ cộng tác sau khi được xác thực sử dụng địa chỉ email của người dùng (xác thực qua hệ thống LDAP của Mobifone)



Hình 4.5: Biểu đồ tuần tự quá trình xác thực bằng tài khoản Email Mobifone

- Module quản lý, hỗ trợ gửi/nhận message P2P giữa các mChat client.

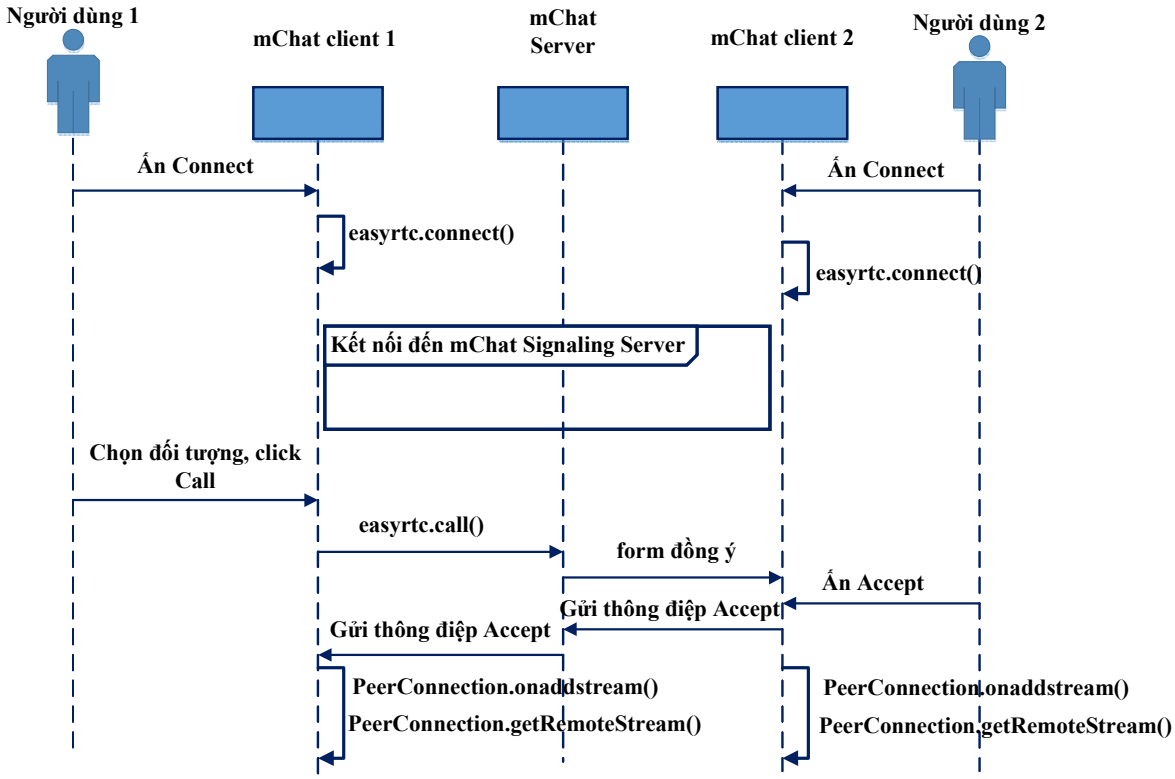
Việc chat hay gửi nhận thông điệp giữa mChat client được bắt đầu hàm easyrtc.call(). Hàm này có nhiệm vụ khởi tạo đối tượng RTCPeerConnection và thiết lập các thông số cần thiết trước khi hai bên có thể trao đổi thông điệp. Quá trình báo hiệu cũng bắt đầu thực hiện với việc gửi những thông điệp offer, answer. Sau khi kết thúc quá trình báo hiệu, kết nối peer được thiết lập, việc gửi và nhận message được thông qua hàm gửi sendDataP2P() phía người gửi và xử lý listener onChannelMsg phía người nhận



Hình 4.6. Biểu đồ tuần sự module gửi/nhận text message

- Module quản lý, hỗ trợ gọi và nhận hay ngắt cuộc gọi audio P2P giữa client.

Việc thực hiện voice-chat bắt đầu bằng lời gọi hàm easyrtc.connect(). Hàm connect() kết nối đến máy chủ mChat qua WebSocket, giúp mChat server biết được đang có những người dùng nào sẵn sàng kết nối cho chức năng voice-chat. Hàm connect() cũng khởi động những nguồn media, cụ thể trong trường hợp này là xin phép người dùng truy cập vào microphone để sẵn sàng cho voice-chat. Sau khi kết nối tới máy chủ, hàm easyrtc.call() thực hiện cuộc gọi đến người dùng đầu xa, khi đầu xa đồng ý kết nối thì đối tượng RTCPeerConnection được thiết lập phù hợp, và dòng audio sẽ được truyền qua mạng và truyền ra loa nhờ vào các xử lý của sự kiện onaddstream và getRemoteStream đối tượng RTCPeerConnection

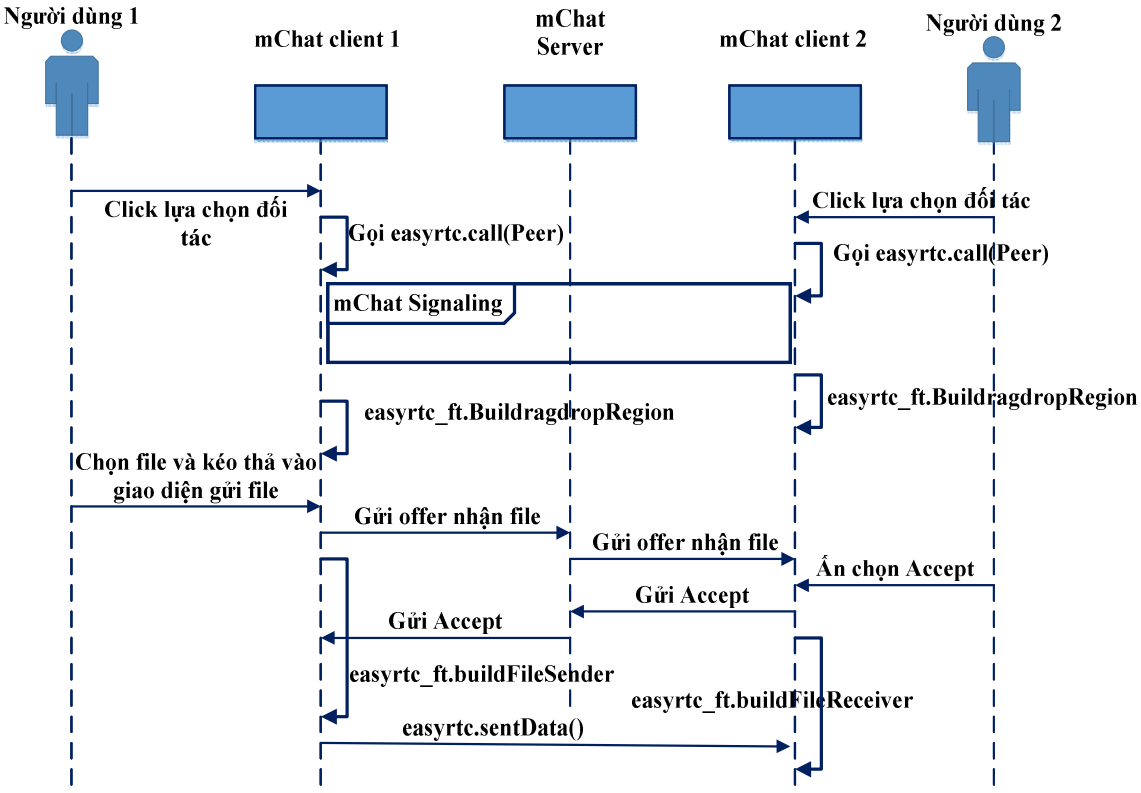


Hình 4.7 Biểu đồ tuần sự thiết lập và gọi audio – voice chat

- Module gửi file:

Cho phép người dùng gửi file P2P giữa trình duyệt, hỗ trợ người dùng gửi những file dữ liệu lớn. Gửi file bắt đầu bằng thao tác lựa chọn người chat rồi kéo thả file của trên giao diện. Lựa chọn người dùng tương tự như trong phần gửi thông điệp, sẽ gọi đến hàm easyrtc.call(), mục đích là khởi tạo đối tượng RTCPeerConnection và thiết lập các thông số cần thiết trước khi hai bên có thể trao đổi thông điệp. Khi người dùng kéo thả file cần có sự xác nhận từ phía đầu xa đồng ý nhận file. Khi đồng ý thì quá trình gửi file được thực hiện nhờ easyrtc.sendData(). Hàm sendData() sẽ kiểm tra kết nối Peer, nếu có đã có kết nối P2P sẽ gửi file trực tiếp qua kênh DataChannel, nếu không có kết nối P2P giữa client thì sẽ gửi qua mChat server bằng WebSocket. Dưới đây là đoạn code mô tả hàm sendData trong lớp EasyRTC:

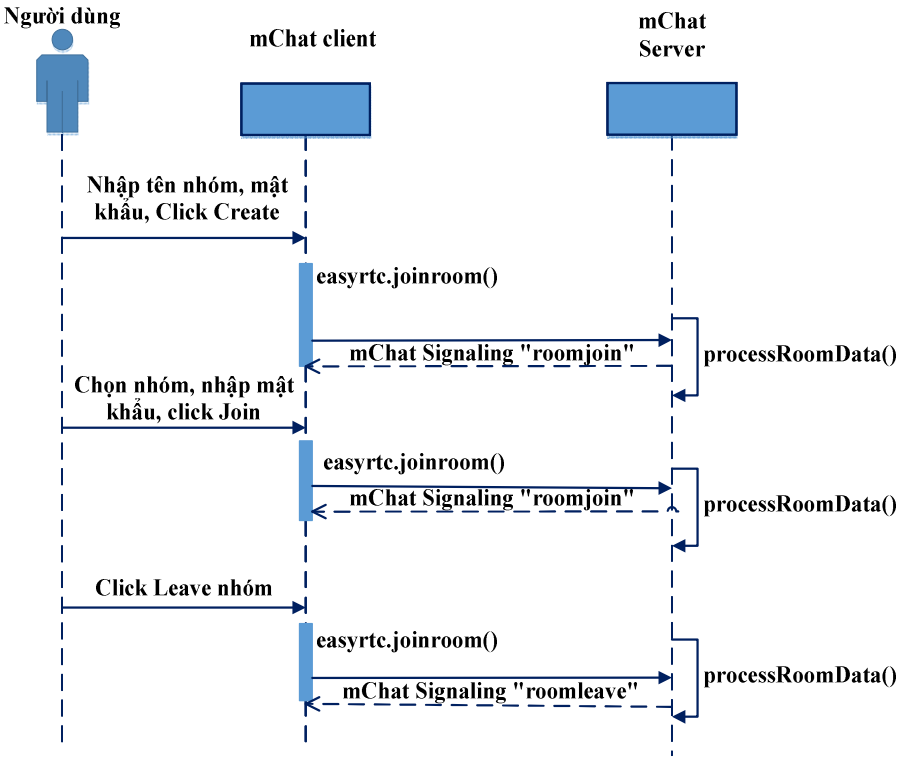
```
this.sendData = function(destUser, msgType, msgData, ackHandler) {
    if (peerConns[destUser] && peerConns[destUser].dataChannelReady) {
        self.sendDataP2P(destUser, msgType, msgData);
            }
    else {
        self.sendDataWS(destUser, msgType, msgData, ackHandler);
    }
```



Hình 4.8: Biểu đồ tuần sự chia sẻ file

- Module quản lý nhóm hay room cộng tác:

Cho phép người dùng chủ động tạo nhóm với mật khẩu, mời người khác tham gia nhóm cộng tác qua link. Để tạo nhóm, hay join nhóm, easyrtc cung cấp hàm joinroom. Để rời nhóm, easyrtc cung cấp hàm leaveroom. Công việc chính trong các hàm trên là gửi thông tin joinroom, leaveroom cho mChat serer cập nhật thông tin về nhóm thời gian thực. Tất cả các mChat client sẽ được update thông tin nhờ vào hàm processRoomData và được cập nhập mỗi 0.1 giây (sử dụng timer).

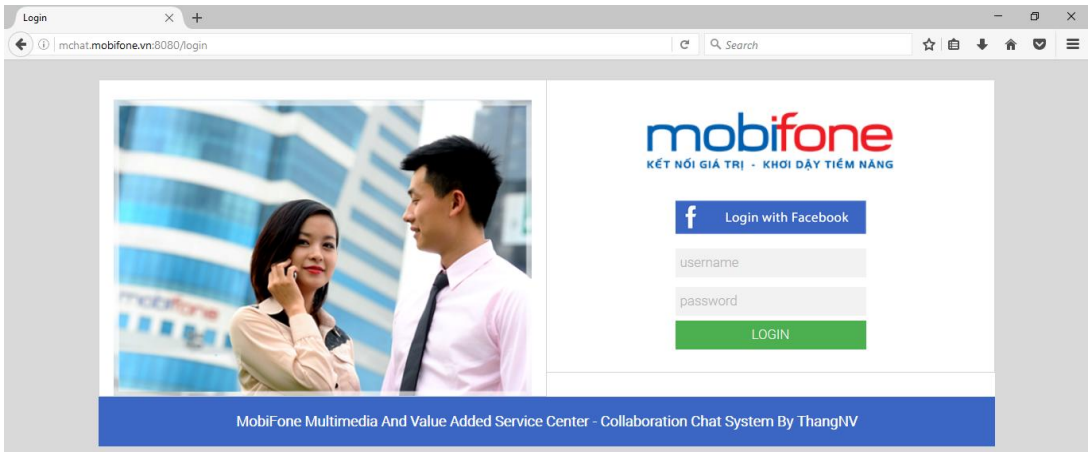


Hình 4.9: Biểu đồ tuần tự các chức năng quản lý nhóm

4.2.6.4. Thiết kế giao diện

Giao diện hệ thống thiết kế theo chuẩn "Responsive Design" giúp tự động dàn trang, hiển thị trên các kích cỡ màn hình khác nhau (PC, smartphone, tablet..). Thiết kế sử dụng Bootstrap framework, là framework phổ biến nhất cho việc thiết kế html, css, JavaScrip cho các web tương tác. Luận văn sử dụng EJS, ngôn ngữ bản mẫu phía client, cho phép kết hợp dữ liệu và mẫu template để tạo ra HTML.

Giao diện login hệ thống, cho phép xác thực bằng tài khoản LDAP hoặc tài khoản Facebook:

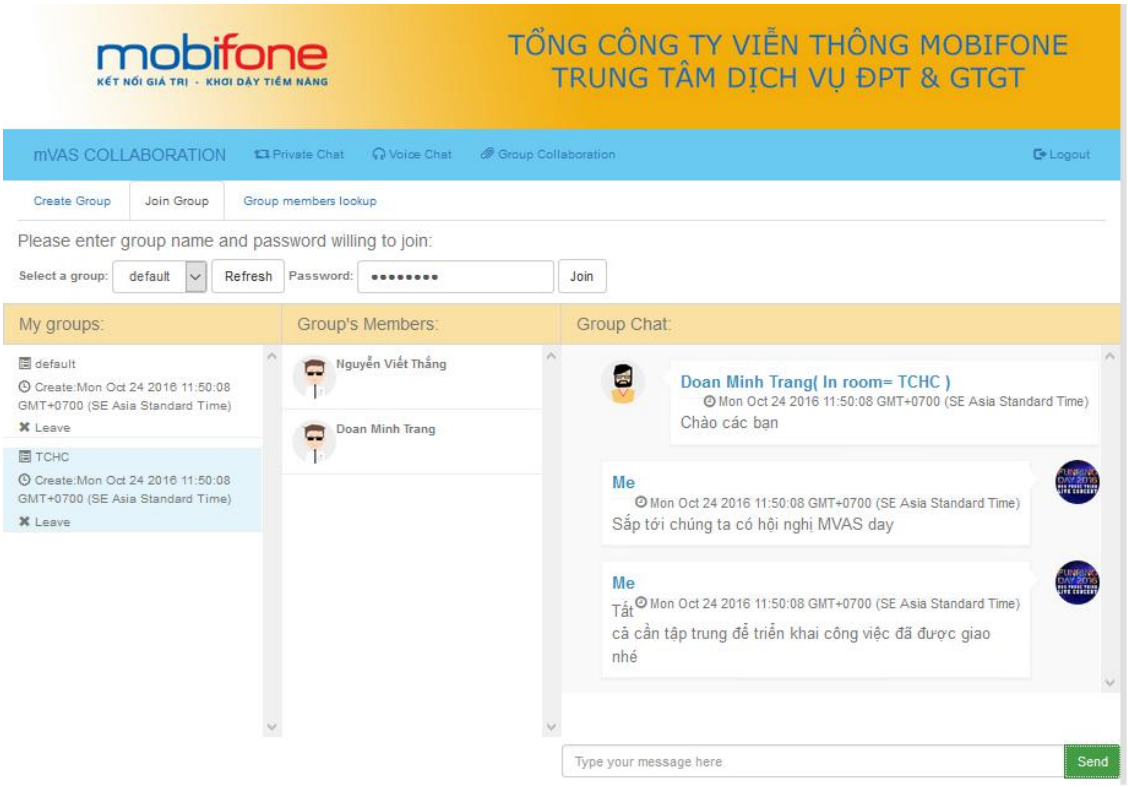


Hình 4.10: Giao diện login

Giao diện chính sau khi người dùng xác thực thành công qua Microsoft AD hoặc qua Facebook:

Hình 4.11: Giao diện Private Chat

Giao diện group Collaboration, cho phép người dùng tạo nhóm, tham gia nhóm, rời nhóm:



Hình 4.12: Giao diện group chat

Giao diện voice-chat, cho phép người dùng thiết lập cuộc gọi P2P với người khác:

Hình 4.13: Giao diện voice-chat

4.2.6.5. Các thách thức

Trong phạm vi của đề tài, tác giả không tập trung xây dựng ứng dụng cộng tác với nhiều tính năng có thể thực hiện với EasyRTC như chat video, chia sẻ màn hình

(screen sharing), hay nhóm chat audio, video (số lượng tham gia nhỏ) vì việc phát triển chúng về cơ bản là tương tự. Ứng dụng cộng tác demo cũng chưa hỗ trợ khả năng lưu trữ, hiển thị lịch sử trao đổi giữa các mChat client. Vấn đề này cũng không gặp thách thức về kỹ thuật, có thể giải quyết bằng cách bổ sung thêm Database phía máy chủ, các message khi được gửi P2P đồng thời cũng sẽ gửi đến mChat server lưu trữ qua WebSocket. Vì tính chất tập trung vào tương tác thời gian thực nên yếu tố này không phải quá cần thiết. Tuy nhiên, có một tính năng quan trọng là tính năng push notification, hay gửi thông báo cho mChat client A biết được đang có mChat client B đang muốn tương tác qua chat text, hoặc qua cuộc gọi. Với ứng dụng demo như luận văn trình bày, đòi hỏi người dùng muốn cộng tác phải cùng sử dụng trình duyệt và truy cập vào ứng dụng web gần cùng thời điểm. Đây là điểm hạn chế lớn trong cộng tác vì khung thời gian hoạt động của người dùng rất khác nhau nếu không có cơ chế thông báo. Gửi thông báo rất hữu ích khi người dùng mở trình duyệt nhưng chưa truy cập ứng dụng hoặc thậm chí là không mở trình duyệt nhưng vẫn biết được thông tin có người muốn tương tác để có thể truy cập ứng dụng và bắt đầu phiên cộng. Đây là một thách thức nói chung với tất cả các ứng dụng cộng tác với WebRTC trên môi trường desktop (Môi trường mobile thì có thể gửi tin nhắn thông báo từ ứng dụng đến máy chủ Google/Apple notification, từ đó gửi đến thiết bị). Trình duyệt Chrome từ phiên bản version 42, đã cung cấp Push API và Notification API cho nhà phát triển. Bản chất là Chrome thông qua Extension hỗ trợ cho các file JavaScript chạy ngầm và lắng nghe kết nối WebRTC. Khi có kết nối WebRTC sẽ hiện ra thông báo. Giải pháp này không yêu cầu người dùng vào ứng dụng nhưng phải chạy trình duyệt Chrome. Tương tự như vậy Firefox từ phiên bản 44 cũng hỗ trợ Web Push, khi trình duyệt Firefox hoạt động nó có thể nhận được tin nhắn ngay cả khi đóng các tab ứng dụng. Ngoài nỗ lực của Google, Firefox thì W3C và IETF cũng đã bắt đầu những nghiên cứu bàn thảo ra các tiêu chuẩn trong việc "push" thông điệp này từ năm 2012. Hiện tại, W3C đã công bố Push API working draft và được cập nhật gần nhất vào 15/12/2015, còn IETF đã có dự thảo WebPush protocol (nhóm IETF WEBPUSH) và được cập nhật gần nhất vào 17/10/2015. Vì vậy, trong lúc chưa được chuẩn hóa, muốn xây dựng ứng dụng có tính năng "push" thông báo cần dựa trên những cung cấp, API của nhà phát triển trình duyệt riêng.

4.2.7. Kết quả thử nghiệm và đánh giá

Môi trường thử nghiệm mChat server:

- Hệ điều hành: Windows 10 Pro, Windows 7 Pro, Ubuntu 16.04x64
- Máy chủ web (http) cài đặt qua Node.JS phiên bản 4.6.0 trên máy chủ Windows và Node.JS phiên bản 4.2.6 trên máy chủ Ubuntu, kết hợp với module http và module express.

- Máy chủ báo hiệu (sử dụng WebSockets cho báo hiệu) cài đặt qua Node.JS phiên bản 4.6.0 trên máy chủ Windows và Node.JS phiên bản 4.2.6 trên máy chủ Ubuntu, kết hợp với module socket.io.
- Máy chủ hỗ trợ xác thực sử dụng module passport trong Node.JS

(Các dịch vụ phần báo hiệu, web, hỗ trợ xác thực cài chung trên một máy vật lý)

- Máy chủ LDAP: máy chủ Windows Server 2012 R2, cài đặt dịch vụ Active Directory Domain Services. Đây là máy chủ đang chạy thật của Mobifone, phục vụ quản lý tất cả các tài khoản người dùng cán bộ công nhân viên Mobifone.
- Máy chủ STUN: kết nối đến máy chủ STUN public của Google tại địa chỉ stun.l.google.com:19302. Trong trường hợp không kết nối được STUN của Google sẽ kết nối đến các STUN miễn phí khác tại địa chỉ stun.sipgate.net:10000;

Môi trường thử nghiệm của mChat client:

- Hệ điều hành Windows 10 Pro, Window 7 Pro; trình duyệt thử nghiệm Firefox phiên bản 47.0.2, 49.0.2; Chrome phiên bản 54.0.2840.71m, 54.0.2840.99m; Microsoft Edge 38.14393.0.0
- Hệ điều hành Android 6.0.1, cài đặt trình duyệt Chrome 54.0.2840.85, Firefox 49.0.2…
- Hệ điều hành IOS 10.1.1, cài đặt trình duyệt Chrome 54.0.2840.91, trình duyệt Firefox 5.3 (2).
- Các máy tính, thiết bị mobile được kết nối vào mạng của Trung tâm MVAS Mobifone (cùng lớp mạng 10.151.181.0/24).

Kết quả thử nghiệm:

Kết quả thử nghiệm trong nội bộ mạng Trung tâm dịch vụ Đa Phương tiện và giá trị gia tăng (MVAS) Mobifione trong một số kịch bản chính như sau:

- Chức năng xác thực: xác thực qua LDAP của Mobifone và xác thực qua Facebook đều hoạt động tốt với tất cả các trình duyệt, môi trường client được thử nghiệm.
- Chức năng chat text Peer to Peer: hoạt động tốt với các trình duyệt Chrome, Firefox trên nền tảng Windows, Android. Chưa hoạt động với Chrome, Firefox cài đặt trên IOS.
- Chức năng gửi file: hoạt động tốt với các trình duyệt Chrome, Firefox trên nền desktop. Chia sẻ file từ nền tảng desktop sang nền tảng mobile Android với các trình duyệt Chrome, Firefox hoạt động tốt. Tuy nhiên khi thực hiện thao tác chia

sẻ file, nếu người dùng đóng trình duyệt hoặc mở tab làm việc khác, quá trình gửi/nhận bị dừng lại mà không tự phục hồi (thiếu chức năng resumable).

- Chức năng quản lý nhóm: hoạt động tốt với tất cả các trình duyệt, do phần này chủ yếu là quản trị dữ liệu nhóm.
- Chức năng voice-chat:
    - Với người dùng sử dụng Firefox trên desktop: hoạt động tốt.
    - Với người dùng Chrome trên desktop: gặp lỗi PermissionDeniedError khi truy cập vào local media, cụ thể ở đây là truy cập vào microphone của máy. Lỗi này được khắc phục khi Node.JS được dựng trên nền máy chủ https thay vì http như phần thử nghiệm.
    - Với người dùng nền tảng mobile cả Android và IOS: chưa hoạt động cả với Firefox, Chrome.
- Chức năng gửi text trong nhóm: hoạt động tốt với trình duyệt Firefox, Chrome trên cả nền tảng Desktop và Mobile.

Trong quá trình thử nghiệm, để có thông tin kết nối Peer 2 Peer giữa các trình duyệt, có thể sử dụng công cụ của Firefox và Chrome đã được tích hợp sẵn trình duyệt. Với Firefox truy cập about:webrtc, với Chrome truy cập chrome://webrtc-internals.

Bảng 4.3 Thống kê ứng viên trong quá trình thiết lập kết nối P2P với Firefox

Local Candidate	Remote Candidate	ICE State	Priority	Nominated	Selected
92.168.194.1:57925/udp (host)	192.168.100.11:49742/ udp(host)	failed	9.07E+18		
192.168.68.1:57926/udp (host)	192.168.100.11:49742/ udp(host)	failed	9.07E+18		
192.168.100.9:57927/udp (host)	192.168.100.11:49742/ udp(host)	succeeded	9.07E+18	TRUE	TRUE
192.168.194.1:57925/udp (host)	113.190.150.184:4974 2/udp (serverreflexive)	cancelled	7.20E+18		
192.168.68.1:57926/udp (host)	113.190.150.184:4974 2/udp (serverreflexive)	cancelled	7.20E+18		
192.168.100.9:57927/udp (host)	113.190.150.184:4974 2/udp (serverreflexive)	cancelled	7.20E+18		

Bảng 4.3 thể hiện thông tin session được thiết lập với Firefox, hai ứng viên thiết lập P2P ở đây có địa chỉ transport lần lượt là 192.168.100.9:57927/udp và 192.168.100.11:49742/udp.

Đánh giá ứng dụng

Qua việc xây dựng ứng dụng demo về phần cộng tác chia sẻ dữ liệu cho thấy việc sử dụng EasyRTC thật sự dễ dàng, giúp nhanh chóng triển khai những ứng dụng tương

tác thời gian thực giữa trình duyệt đặc biệt trên nền tảng Desktop. Với ứng dụng web đơn giản này, cán bộ công nhân viên của Trung tâm MVAS đã có thể dễ dàng trao đổi thông tin, tài liệu với nhau trực tiếp, theo nhóm mà tương đối tiện lợi, đảm bảo an toàn bảo mật. Chất lượng voice thử nghiệm trên desktop cho chất lượng tương đối tốt, hai đầu nghe rõ, độ trễ thấp. Ứng dụng đã đáp ứng được các yêu cầu chính đã đặt ra. Tuy nhiên, ứng dụng còn một số hạn chế chưa thể giải quyết ở thời điểm hiện tại như chưa hỗ trợ trình duyệt IE, Safari; ứng dụng cũng chưa hoạt động hết các chức năng trên nền tảng mobile như kết quả thử nghiệm ở trên. Giao diện cũng chưa thực sự tối ưu hướng đến thuận tiện cho người dùng trong sử dụng như những ứng dụng OTT. Về tính năng, Bảng 4.4 ở dưới so sánh ở một số khía cạnh khả năng có thể phát triển của ứng dụng demo với các ứng dụng OTT, web khác:

Bảng 4.4: So sánh các ứng dụng chat và chia sẻ file

Tính năng	Skype	Viber	Facebook	Khả năng ứng dụng demo WebRTC	Ghi chú
Chat text					
Chat nhóm	x	x	X	x	
Chat offline	x	x	x		
Lưu lịch sử chat	x	x	x		
Cảnh báo (Notification)	x	x	x		
Nền tảng hỗ trợ					
Windows	x	x	x	x	
MAC OS X	x	x		x	
Linux	x	x		x	
Unix	x	x		x	
IOS	x	x	x	x	
Android	x	x	x	x	
Web			x	x	
Live Video Streaming				x	Tính năng này chưa kịp demo
Voice chat	x	x	x	x	
Chia sẻ					
Upload tập tin	x		x	x	
Upload ảnh	x	x	x	x	
Upload Video	x	x	x	x	
Chia sẻ màn hình	x				Hiện tại có có thể phát triển áp dụng cho Chrome
Cần cài đặt	x	x			
Bảo mật					
Mã hóa text gửi đi	x	x	x	x	

Tính năng	Skype	Viber	Facebook	Khả năng ứng dụng demo WebRTC	Ghi chú
Mã hóa text đảm bảo nhà cung cấp không thể đọc được	x		N/A	x	
Mã hóa file gửi	x	x	x	x	

So với nhiều những dự án mã nguồn mỡ hỗ trợ text/voice/video chat miễn phí trên nền WebRTC đã public trên mạng như https://Sharefest.vn, https://hubl.in, *https://talky.io...,* ứng dụng tương tác trong Trung tâm MVAS có điểm khác biệt sau:

- ✓ Tích hợp xác thực với hệ thống quản trị tài nguyên mạng tập trung Microsoft Active Directory của Mobifone. Phần xác thực chính là phần WebRTC không định ra tiêu chuẩn, mà chuyển cho ứng dụng quản lý.
- ✓ Hỗ trợ tạo nhóm cộng tác và quản lý mật khẩu đảm bảo truy nhập an toàn cho nhóm.

Tuy nhiên, nếu so với những ứng dụng OTT phổ biến, ứng dụng demo chỉ có ưu điểm về việc đơn giản trong sử dụng, không cần cài đặt plugin, an tâm về bảo mật mà không đòi hỏi đầu tư hạ tầng mạnh. Để có thể triển khai và áp dụng rộng rãi tại Mobifone, ứng dụng cần bổ sung thêm những tính năng còn chưa hoạt động trên mobile, đặc biệt là phần cảnh báo như đã phân tích ở mục 4.2.6.5.

CHƯƠNG 5. KẾT LUẬN CHUNG

5.1. Các đóng góp của luận văn

Với yêu cầu của đề tài luận văn nghiên cứu ứng dụng WebRTC trong việc xây dựng giải pháp cộng tác và chia sẻ dữ liệu đa phương tiện tại Trung tâm MVAS – Mobifone, luận văn đã thu được một số kết quả sau:

- Tìm hiểu được những nội dung cơ bản của WebbRTC như: các chuẩn giao thức, APIs trong WebRTC, cách thức vượt NAT trong WebRTC
- Nghiên cứu sâu về báo hiệu, vai trò của báo hiệu và các quá trình trong báo hiệu WebRTC.
- Khảo sát và đánh giá các thư viện WebRTC, các hướng tiếp cận sử dụng thư viện WebRTC
- Nghiên cứu và sử dụng thư viện EasyRTC trong việc xây dựng ứng dụng Peer-to-Peer tương tác thời gian thực
- Phân tích yêu cầu, thiết kế ứng dụng, cài đặt thử nghiệm hệ thống cộng tác tại mạng nội bộ Trung tâm dịch vụ Đa phương tiện và giá trị gia tăng Mobifone - Tổng công ty Viễn thông Mobifone và đạt được những kết quả đáng khích lệ.

5.2. Một số hướng phát triển

Qua nghiên cứu WebRTC và thử nghiệm ứng dụng, tôi nhận thất WebRTC là công nghệ rất tiềm năng, đặc biệt hiệu quả khi triển khai nhanh những ứng dụng đòi hỏi tương tác thời gian thực giữa các tình duyệt với tính đơn giản khi cài đặt, dễ sử dụng với người dùng. Các hạn chế của WebRTC như chưa được hỗ trợ bởi trình duyệt IE của Microsoft, Safari của Apple hiện tại đã có giải pháp cài đặt thêm các WebRTC plugin, dù như vậy không đúng theo tiêu chí là không plugin mà WebRTC hướng đến. Vì vậy, WebRTC vẫn đang tiếp tục được nghiên cứu chuẩn hóa (bản update mới nhất là vào tháng 9/2016), tiếp tục phát triển, tương lai ứng dụng WebRTC sẽ có tác động không nhỏ đến ngành công nghiệp web, thậm chí thay thế các ứng dụng cộng tác hiện tại.

Với phân tích đánh giá kết quả thử nghiệm ứng dụng cộng tác, hướng phát triển tiếp theo của đề tài có thể nghiên cứu, phát triển tiếp những công việc sau:

- Hoàn thiện các tính năng tương tự như các OTT như hỗ trợ lưu thông tin chat office, cảnh báo notification.
- Hoàn thiện tính năng voice trên nền tảng mobile.
- Bổ sung khả năng resumable cho việc gửi/nhận file.
- Nghiên cứu khả năng phát triển tính năng chia sẻ màn hình – screen sharing. Đến thời điểm hiện tại mới chỉ có trình duyệt Chrome hỗ trợ để ứng dụng có thể phát triển tính năng này, về bản chất là chụp ảnh màn hình liên tục và gửi cho trình duyệt đầu xa.

- Nghiên cứu cách cài đặt tối ưu hiệu năng chức năng như máy chủ EasyRTC, máy chủ báo hiệu, lựa chọn phương án tối ưu trong trường hợp số lượng người dùng lên đến hơn 5000 cán bộ công nhân viên.

Hoàn thành các việc nghiên cứu này thì ứng dụng cộng tác chia sẻ dữ liệu đa Phương tiện tại Trung tâm MVAS có thể ứng dụng cho không chỉ TCT Viễn thông Mobifone nói riêng mà cho tất cả các doanh nghiệp, tổ chức nói chung, đảm bảo được người dùng đón nhận.

TÀI LIỆU THAM KHẢO

TIẾNG ANH

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

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