

## MỤC LỤC

DANH MỤC CÁC TỪ VIẾT TẮT .....	6
DANH MỤC CÁC BẢNG .....	7
DANH MỤC CÁC HÌNH VẼ .....	8
CHƯƠNG 1. MỞ ĐẦU .....	9
1.1. Đặt vấn đề .....	9
1.2. Phạm vi và mục tiêu của luận văn .....	9
1.3. Phương pháp và bối cảnh nghiên cứu .....	10
CHƯƠNG 2. TỔNG QUAN VỀ WEBRTC .....	11
2.1. Quá trình phát triển .....	11
2.2. Kiến trúc WebRTC .....	14
2.3. Các APIs trong WebRTC .....	18
2.4. Các tầng giao thức trong WebRTC .....	22
CHƯƠNG 3. BÁO HIỆU TRONG WEBRTC .....	30

## **DANH MỤC CÁC TỪ VIẾT TẮT**

<b>TT</b>	<b>Từ viết tắt</b>	<b>Cụm từ tiếng anh</b>
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

## DANH MỤC CÁC BẢNG

Bảng 2.1: Những tính năng mới của WebRTC (tổng hợp theo [1]) .....	13
Bảng 2.2: So sánh giữa WebSocket và DataChannel [4] .....	21
Bảng 2.3: So sánh tính năng chính các giao thức TCP, UDP và SCTP .....	23
Bảng 4.1: Thư viện WebRTC Javascript.....	42
Bảng 4.2. Cơ chế hoạt động chung các thư viện WebRTC Javascript.....	43
Bảng 4.3 Thống kê ứng viên trong quá trình thiết lập kết nối P2P với Firefox .....	68
Bảng 4.4: So sánh các ứng dụng chat và chia sẻ file.....	69

## DANH MỤC CÁC HÌNH VẼ

Hình 2.1: Kiến trúc ứng dụng Web cổ điển.....	14
Hình 2.2: Truyền thông thời gian thực trong trình duyệt (nguồn [1]) .....	15
Hình 2.4: RTCPeerConnection API [Nguồn 4] .....	19
Hình 2.5: MediaStream mang một hoặc nhiều tracks đồng bộ.....	20
Hình 2.6: Protocol stack trong WebRTC [Nguồn 4].....	22
Hình 2.7: Mô hình hoạt động STUN .....	24
Hình 2.8: Luồng Media qua TURN server .....	26
Hình 2.9: Quy trình hoạt động ICE mức cao.....	26
Hình 2.10: Sơ đồ chuyển trạng thái trong ICE .....	28
Hình 3.1: HTTP Transport cho báo hiệu .....	31
Hình 3.2. Vận chuyển báo hiệu trên Data Channel .....	32
Hình 3.3: Giao thức báo hiệu SIP trên WebSocket .....	34
Hình 3.4: Báo hiệu Jingle over WebSockets cho WebRTC .....	35
Hình 3.5 Các thực thể tham gia quá trình báo hiệu .....	36
Hình 3.6: Quá trình khởi tạo trong báo hiệu WebRTC .....	37
Hình 3.7 Quá trình 1 ICE Negotiation .....	38

## 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 BitTorrent, 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.

### **1.3. Phương pháp và bối cảnh 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

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

- **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])*

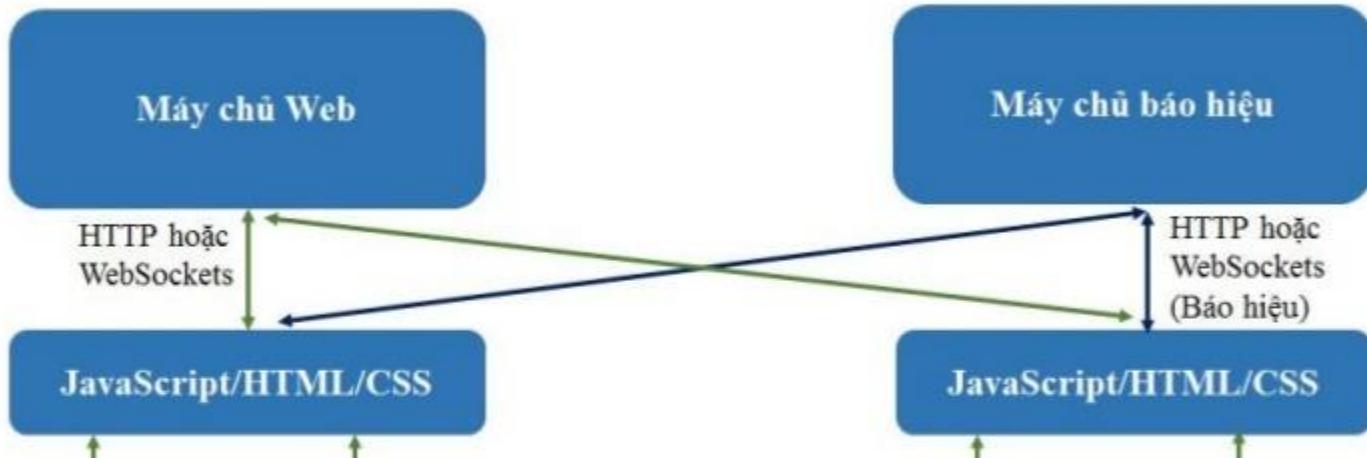
Tính năng	Cách thức cung cấp	Lý do quan trọng
Độc lập với Platform và thiết bị	Sử dụng các APIs chuẩn từ W3C, các giao thức từ IETF	Nhà phát triển có thể viết mã WebRTC HTML5 có thể chạy trên các hệ điều hành và thiết bị

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 thống VoIP, Video đang tồn tại có thể kết nối với hệ thống WebRTC sử dụng những giao thức chuẩn.

## 2.2. Kiến trúc WebRTC

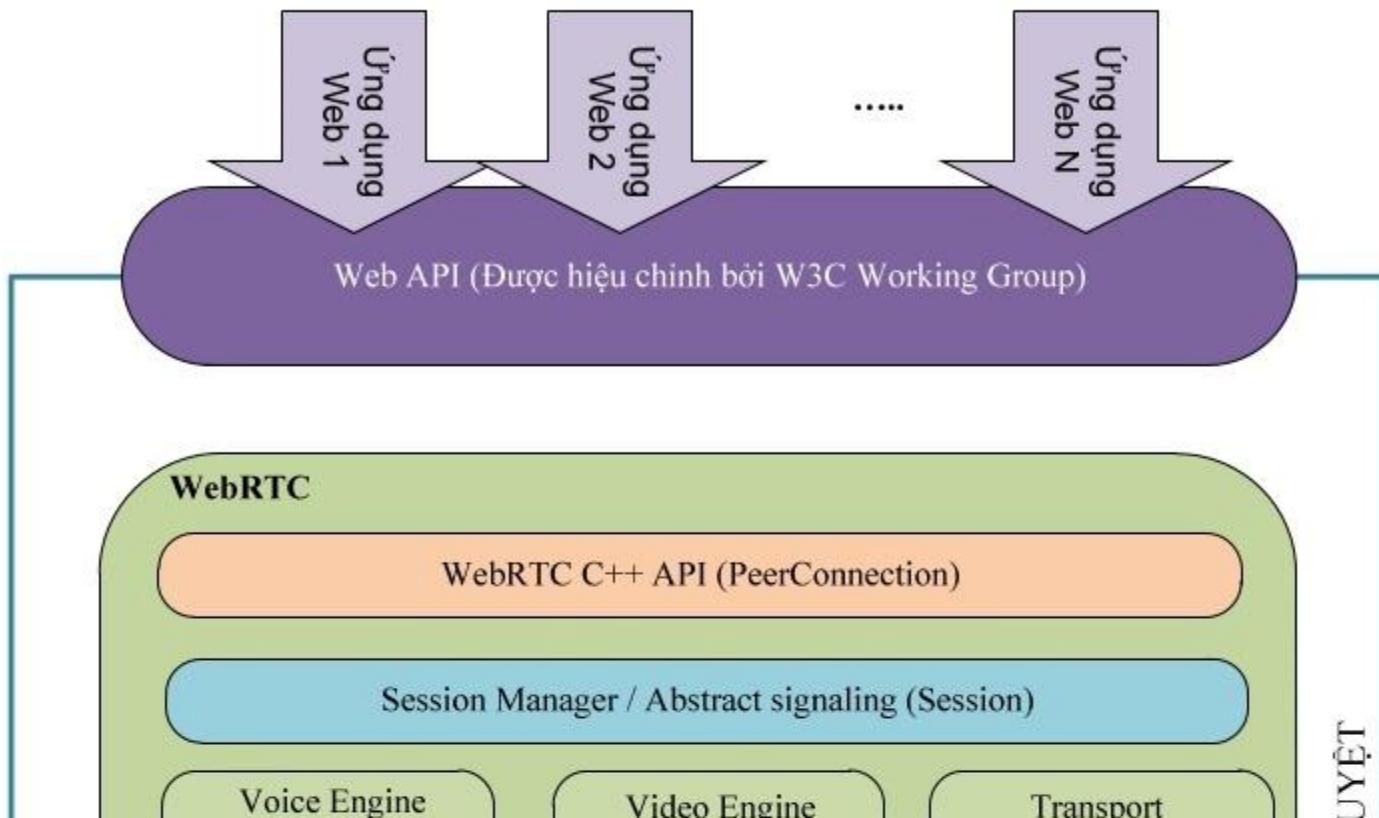
Kiến trúc web cổ điển dựa trên mô hình client-server, trong đó trình duyệt gửi yêu cầu HTTP đến máy chủ để lấy nội dung, máy chủ trả lời, gửi nội dung về cho trình duyệt dưới dạng HTML, thường kèm theo JavaScript và Cascading Style Sheets (CSS). Trong trường hợp đơn giản thì khi máy chủ web trả lời yêu cầu từ client bằng thông tin text, hình ảnh hay thông tin khác như client mong muốn. Trong trường hợp phức tạp

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].



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 jitter 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.



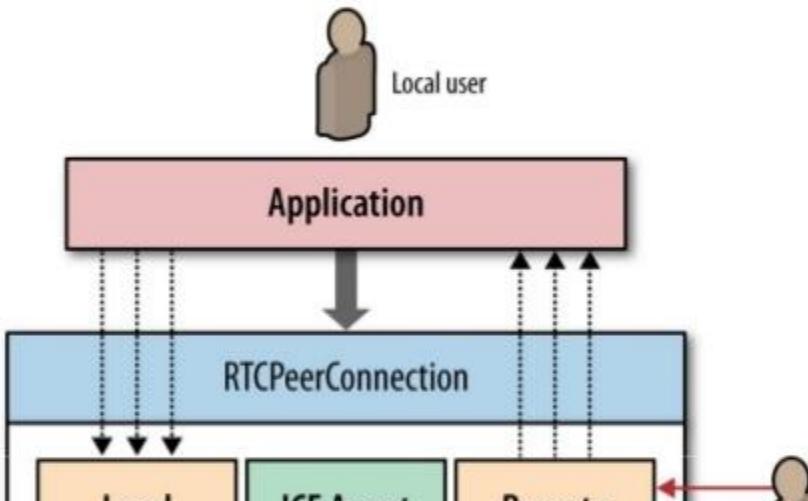
Trong hình 2.3, thành phần video engine là framework xử lý chuỗi video từ camera đến mạng và từ mạng ra màn hình. Trong đó, video codec sử dụng VP8 (một dạng nén video mở, miễn phí sở hữu bởi Google và tạo ra bởi On2 Technologies) và VP9, hỗ trợ tính năng Video jitter buffer để giúp ẩn đi những ảnh hưởng 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).

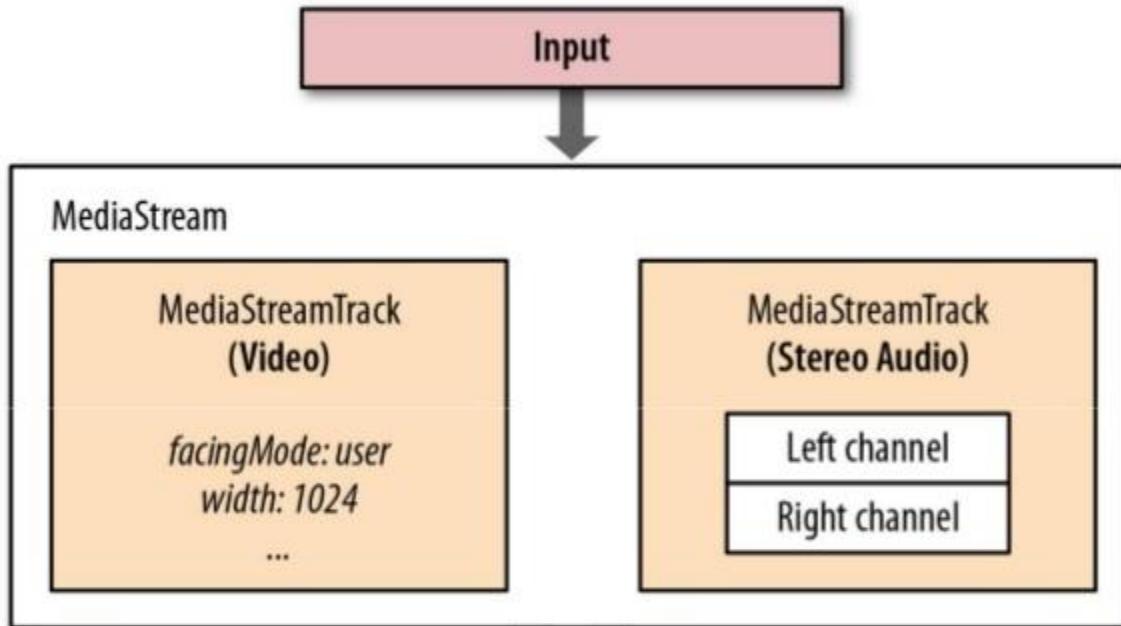
- RTCDATAChannel: giao tiếp peer-to-peer cho các dữ liệu non-media.

### 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.



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.



### c. **RTCDATAChannel**

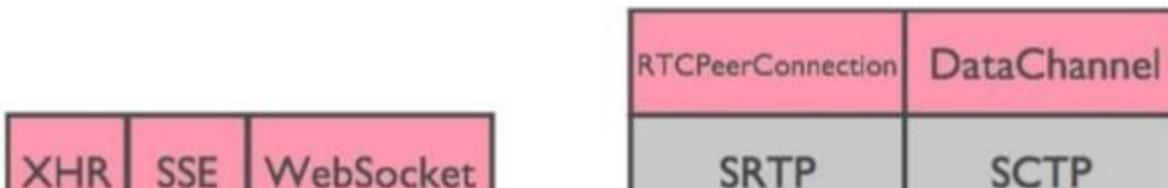
Tương tự như audio và video, WebRTC hỗ trợ truyền thông thời gian thực với các loại dữ liệu khác. RTCDATAChannel API cho phép trao đổi dữ liệu tùy ý peer-to-peer với độ trễ thấp và thông lượng cao. Vì vậy, nó được sử dụng trong những trường hợp như: trong ứng dụng game, ứng dụng remote desktop, chat text thời gian thực, truyền file. Ở lớp dưới, DataChannel sử dụng Stream Control Transmission 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

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:

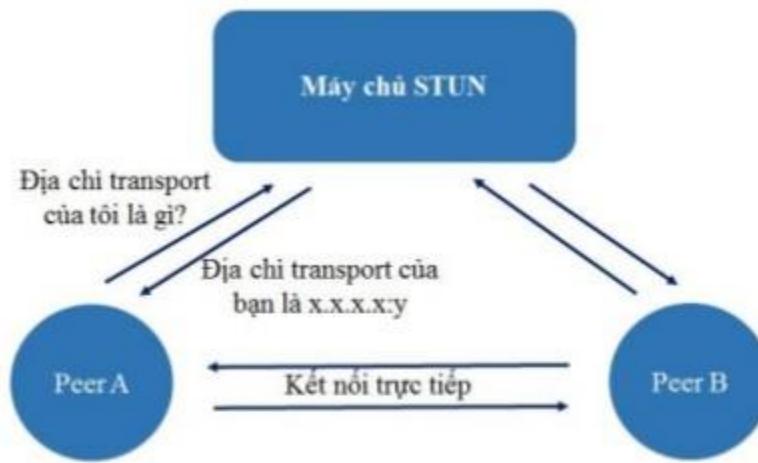


*Bảng 2.3: So sánh tính năng chính các giao thức TCP, UDP và SCTP*

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 để 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

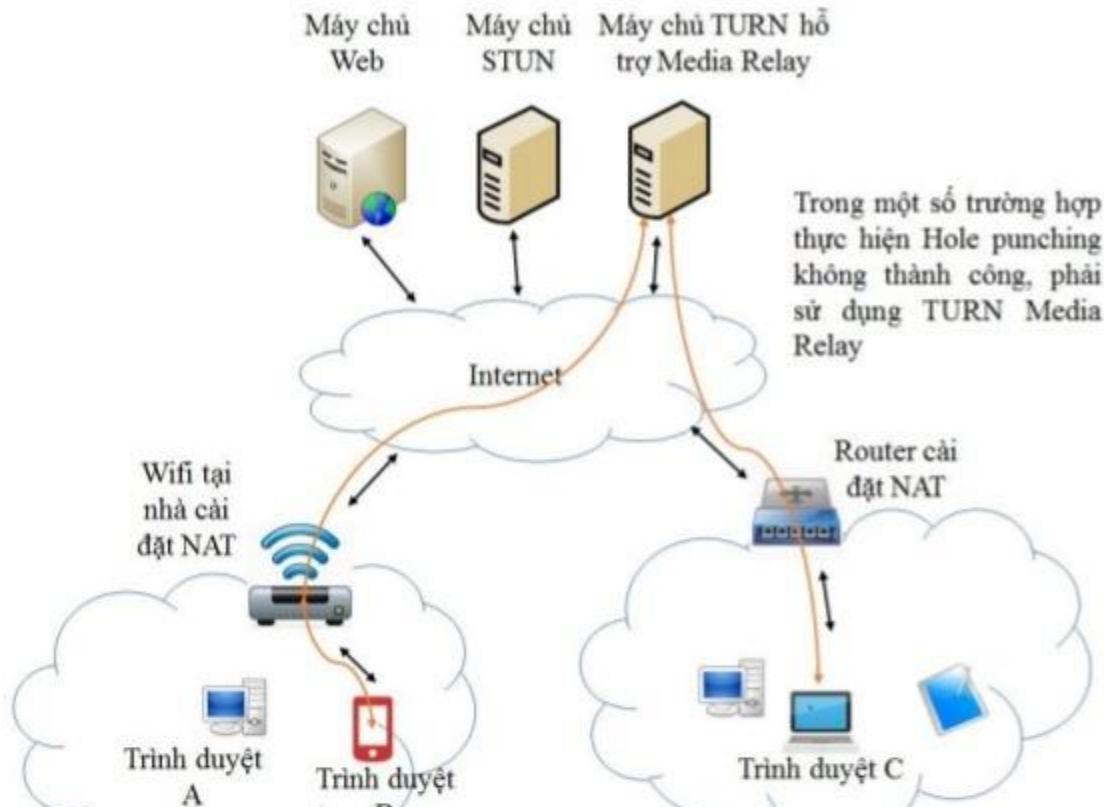


Hình 2.7: Mô hình hoạt động STUN [3]

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].

- 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



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.

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

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 - Signal Server) có thể như sau:

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

✓ 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.

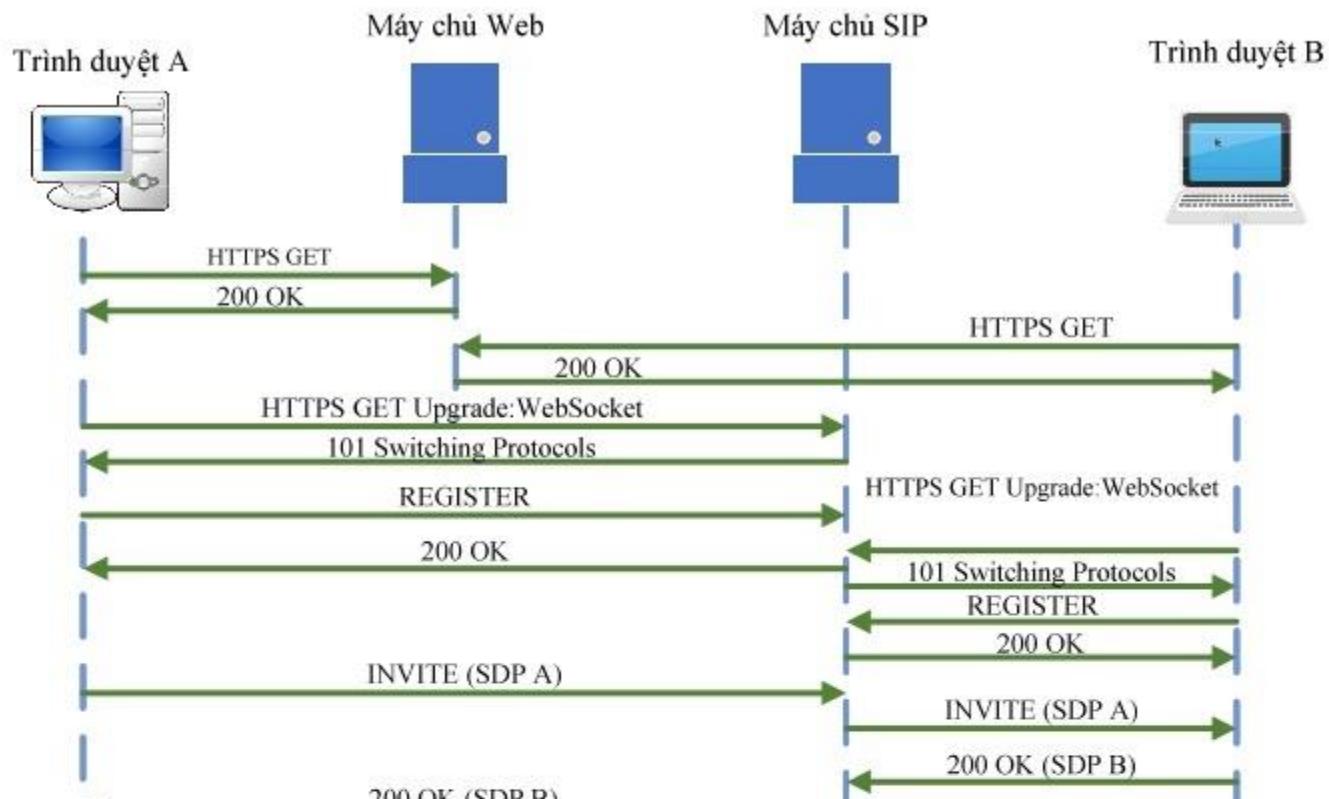
✓ Vận chuyển bằng chính DataChannel

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

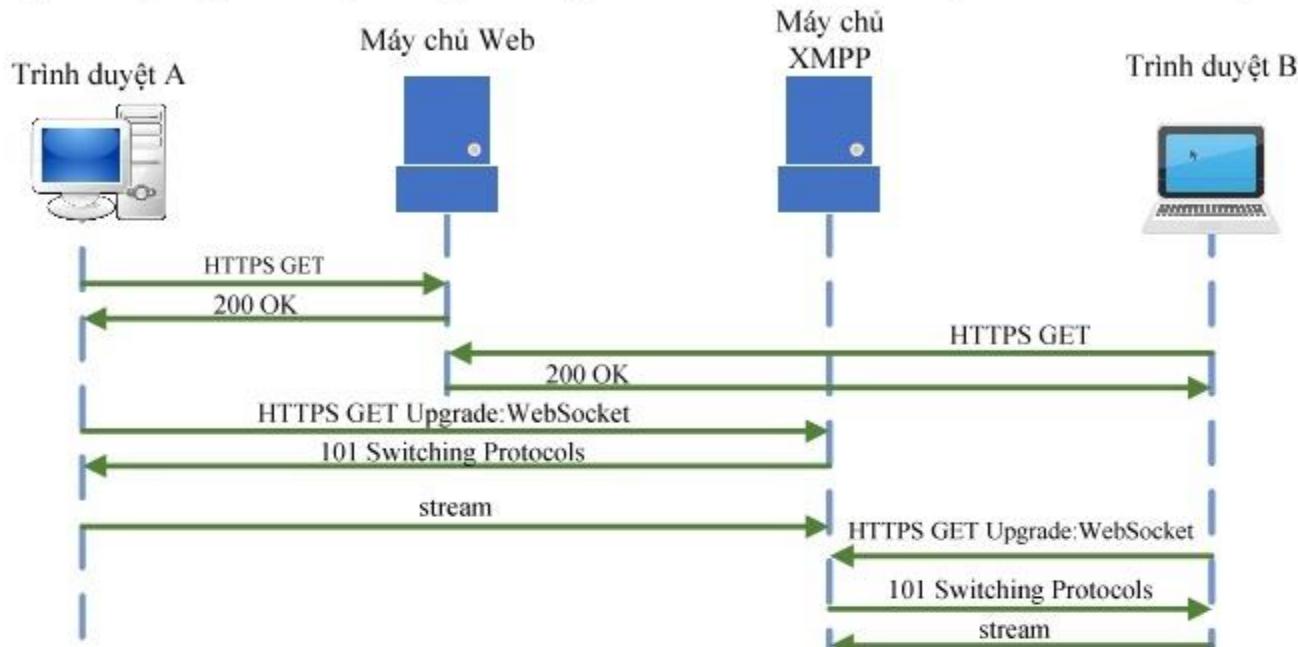
lập kết nối. Trước khi media có thể trao đổi giữa trình duyệt, ICE và bước quản lý khóa DTLS-SRTP phải hoàn thành. Nếu các bước này chỉ bắt đầu sau khi có đồng ý kết nối tạo phiên, nó có thể gây ra độ trễ đến vài giây. Với báo hiệu qua kênh dữ liệu, ICE và quản lý khóa DTLS-SRTP được thực hiện để thiết 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



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:



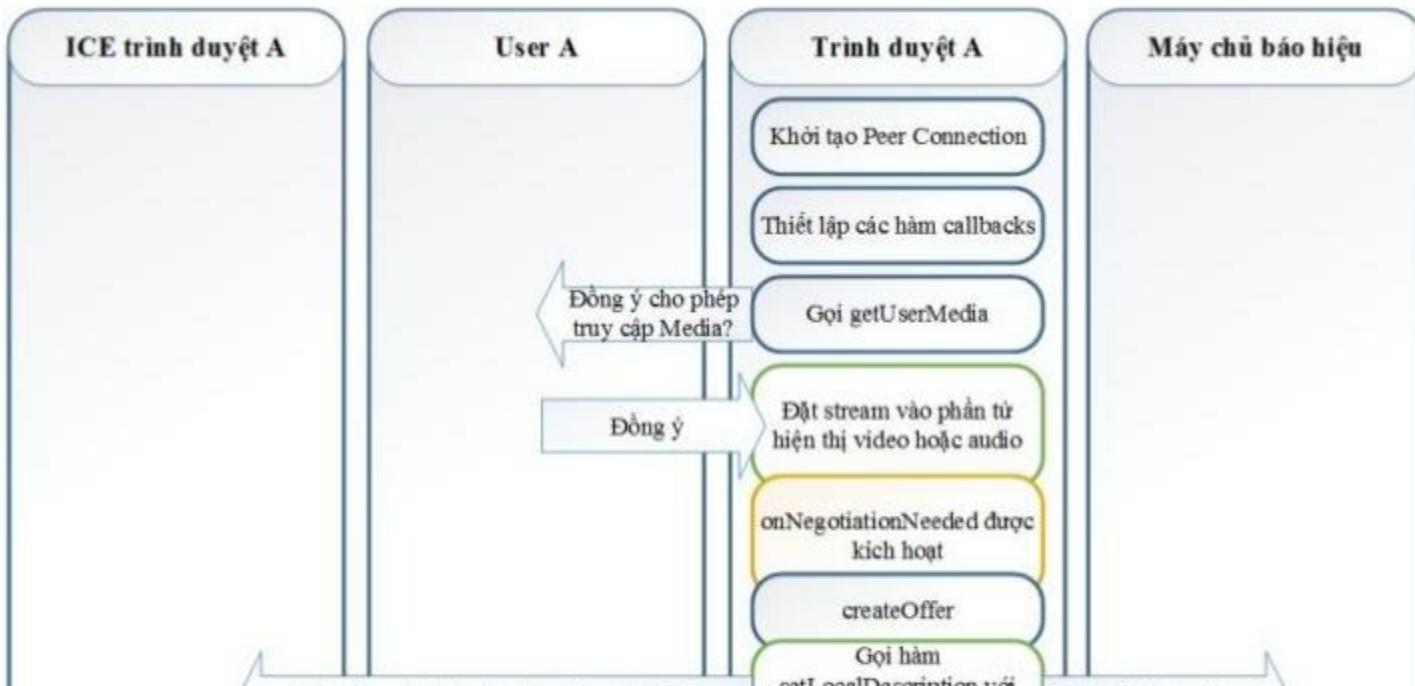
Sử dụng JSON cùng với thư viện đảm nhận việc thiết lập và duy trì kênh hai chiều tin cậy với máy chủ báo hiệu là khá đơn giản và hiệu quả, dù phát sinh thêm chi phí xây dựng cổng ứng dụng tùy biến (custom gateway) nếu muốn kết nối ứng dụng web với dịch vụ thông tin liên lạc bên ngoài.

### 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:

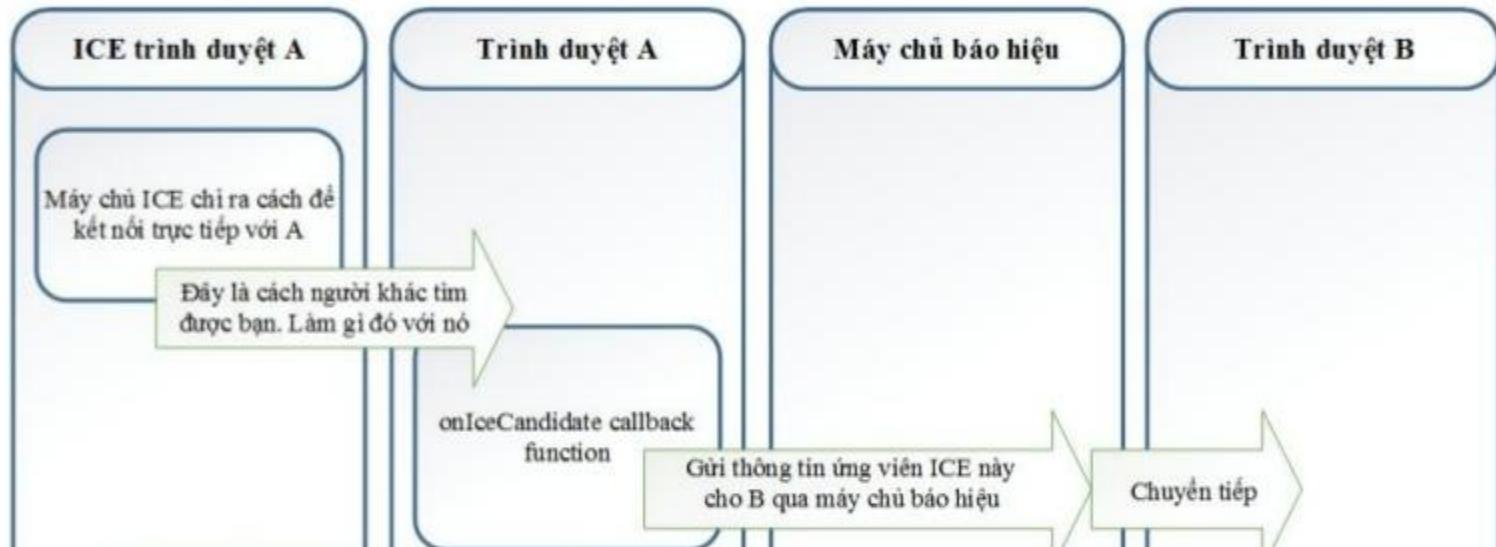
- **WebRTC Javascript callback logic:** các logic này handle tất cả những xử lý mức trình duyệt của WebRTC
- **STUN/TURN Sever Session Description Protocol (SDP) messaging:** Đây là logic báo hiệu diễn ra ngoài kết nối WebRTC để cài đặt kết nối P2P giữa 2 trình duyệt theo yêu cầu.
- **ICE (Interactive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

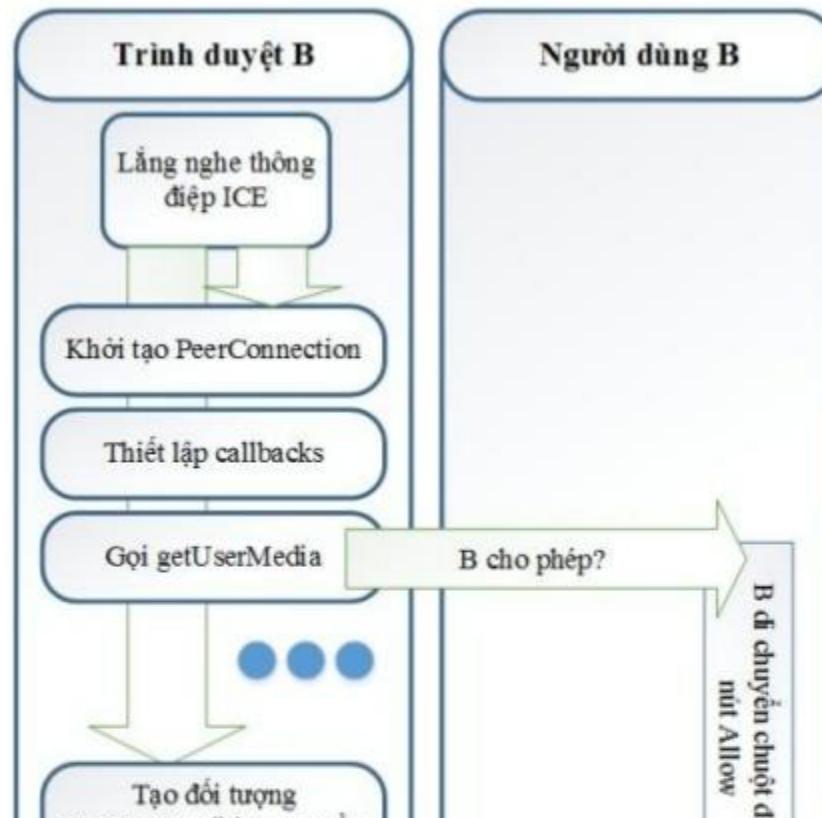
các ứng dụng video), tạo thông điệp SDP bằng cách invoke các hàm onNegotiationNeeded, createOffer và setLocalDescription

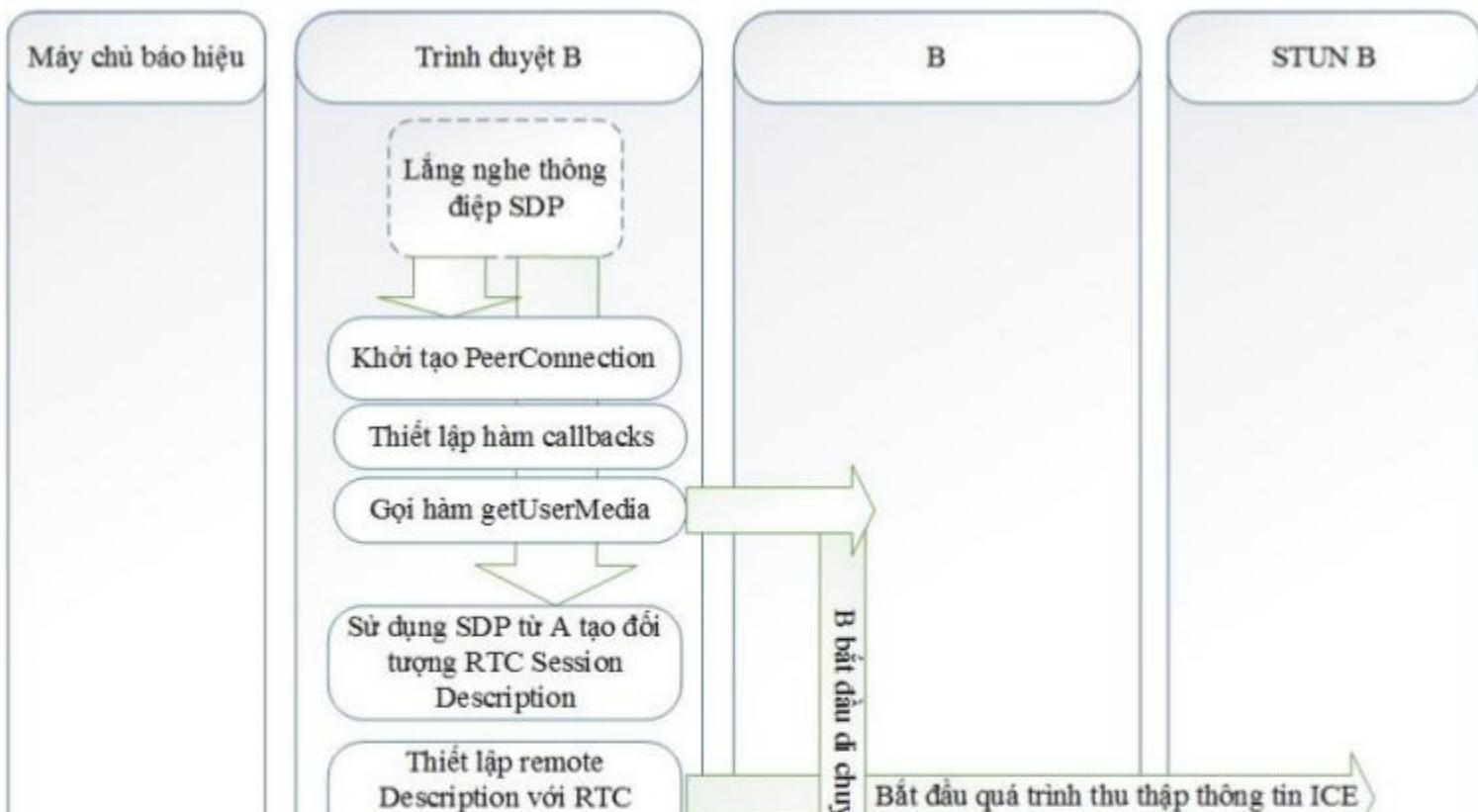


được kết thúc bằng hàm setLocalDescription có được gắn hàm callback khi thành công. Trong hàm call back này sẽ thực hiện gửi thông tin SDP local (loại SDP “offer”) cho peer xa qua máy chủ báo hiệu.

✓ **Đàm phán (Negotiation) ICE**



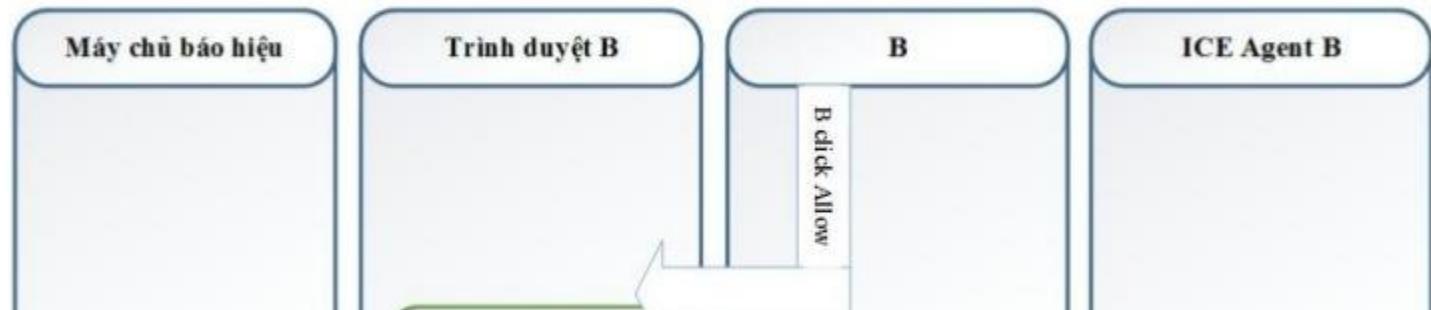




setRemoteDescription. Từ thời điểm này, A đã hiểu và có đủ thông tin về dữ liệu media của B (như các ràng buộc, mã hóa...) để thiết lập kết nối an toàn. Khác với B, do thông điệp SDP A nhận được là loại “answer” nên không có thêm tiến trình nào xử lý dữ liệu SDP này nữa.

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 streaming đến, nhưng A thì chưa nhìn thấy B vì còn đợi B đồng ý (click vào nút Allow) như nêu 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
--------	--------------------------------------

Vline	<a href="https://vline.com/developer/docs/vline.js/">https://vline.com/developer/docs/vline.js/</a>
Weemo	<a href="http://docs.weemo.com/js/">http://docs.weemo.com/js/</a>
Xirsys	<a href="http://xirsys.com/_static_content/xirsys.com/docs/">http://xirsys.com/_static_content/xirsys.com/docs/</a>
Xsockets.net	<a href="http://xsockets.net/docs/javascript-client-api">http://xsockets.net/docs/javascript-client-api</a>

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

	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.

Ấn trong các thư viện là các cơ chế gần giống nhau, giúp các nhà phát triển

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.
- Thay đổi nguyên tắc hoạt động dựa trên ~~đã~~ để hỗ trợ những thay đổi mới nhất của chuẩn

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 ứng dụng riêng lẻ hơn, mà không phải là nghiên cứu kinh doanh chính của họ.

## **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

✓ Về chia sẻ file, dữ liệu:

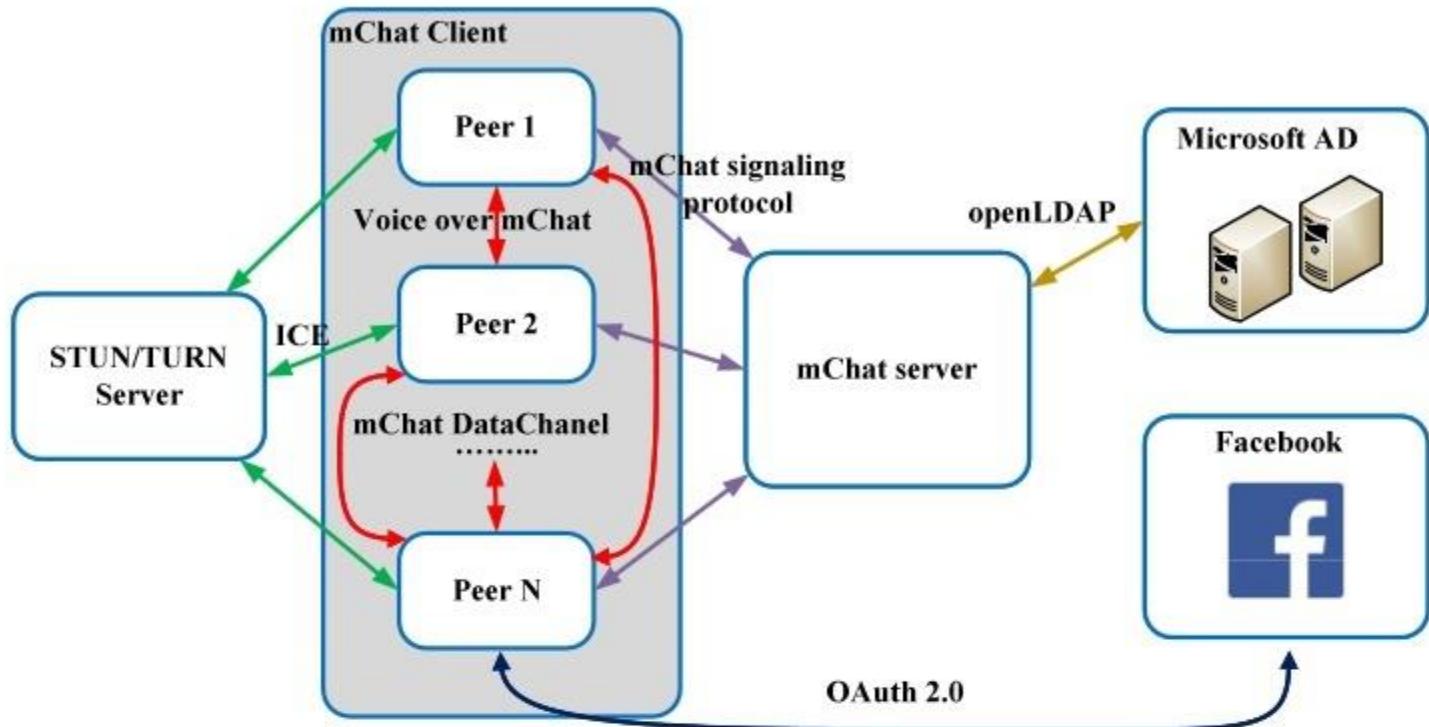
Trung tâm MVAS vẫn sử dụng các hình thức chia sẻ file kiểu cũ với nhiều nhược điểm:

- 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 Microsoft 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.

✓ Về hoạt động công 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 cài 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, để 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).



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

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

- Chức năng xác thực qua tài khoản Facebook: Người dùng có thể sử dụng tài khoản facebook 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.

Luồng sự kiện phụ:

- 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:

- Người dùng click ảnh Facebook trên trang login
- Hệ thống redirect đến trang xác thực của Facebook.
- Người dùng nhập thông tin tài khoản, mật khẩu Facebook, ấn nút Log In
- Dịch vụ của Facebook thực hiện đăng nhập
- Đăng nhập thành công, hệ thống redirect sang trang chủ hệ thống mChat

Luồng sự kiện phụ:

- Đăng nhập thất bại: yêu cầu đăng nhập lại
- 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

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
<b>Luồng sự kiện chính:</b>	
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.	
<b>Luồng sự kiện phụ:</b>	
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.

**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**

- 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 cung cấp 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\_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

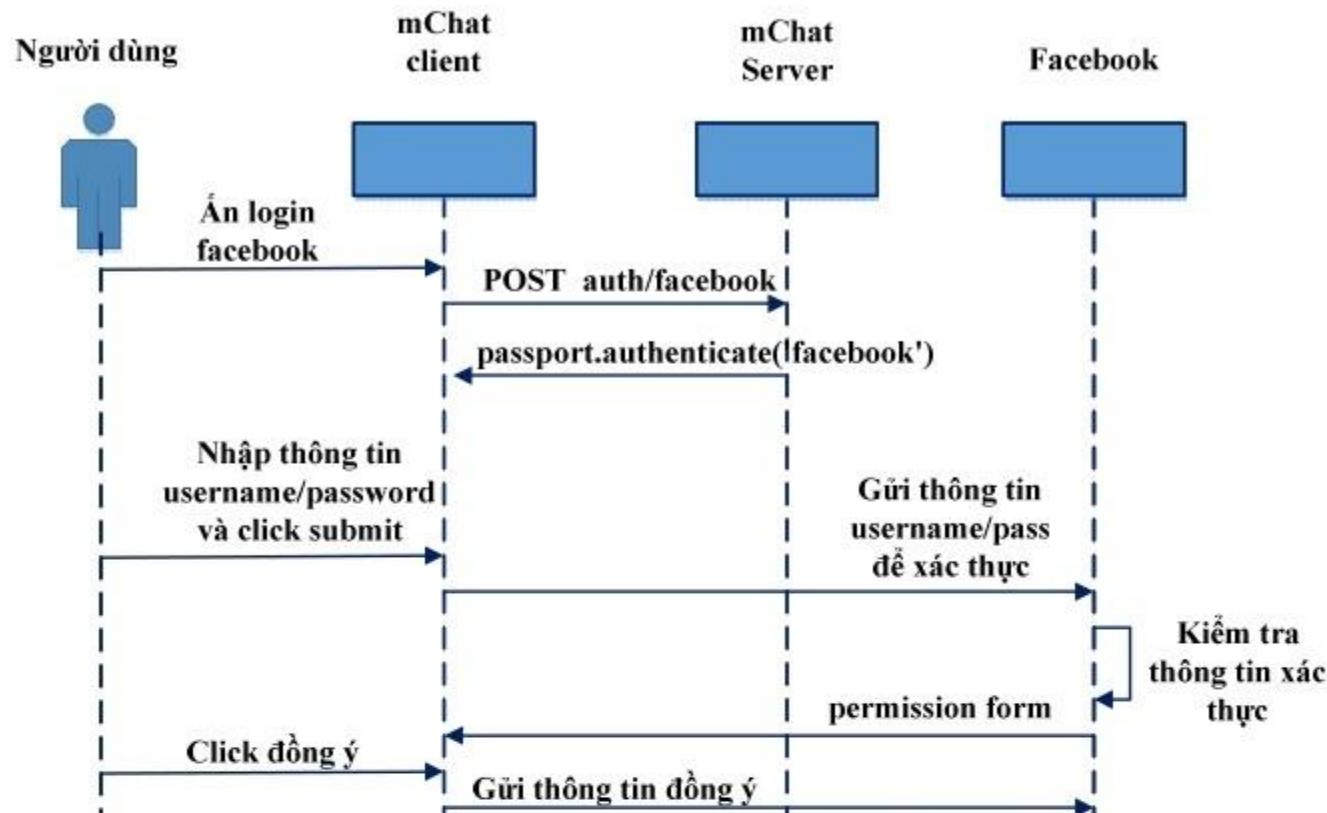
- Pub Object: đối tượng public, được trả về từ hàm `listen()` trong EasyRTC.  
Pub chứa tất cả các phương thức public cho việc tương tác với máy chủ EasyRTC.
- Application Object (`appObj`) phục vụ quản lý ứng dụng, có các phương thức tạo kết nối, phiên, tạo và xóa room trong ứng dụng.
- Connection Object (`connectionObj`) phục vụ quản lý các kết nối đến room,

- 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*

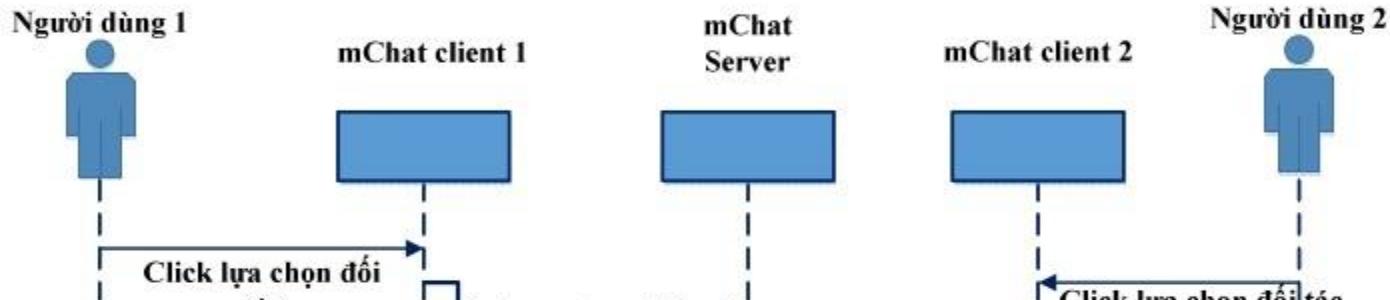
Hệ thống mChat cộng tác chia sẻ dữ liệu đa Phương tiện tại Trung tâm MVAS Mobifone được phát triển theo mô hình MVC:

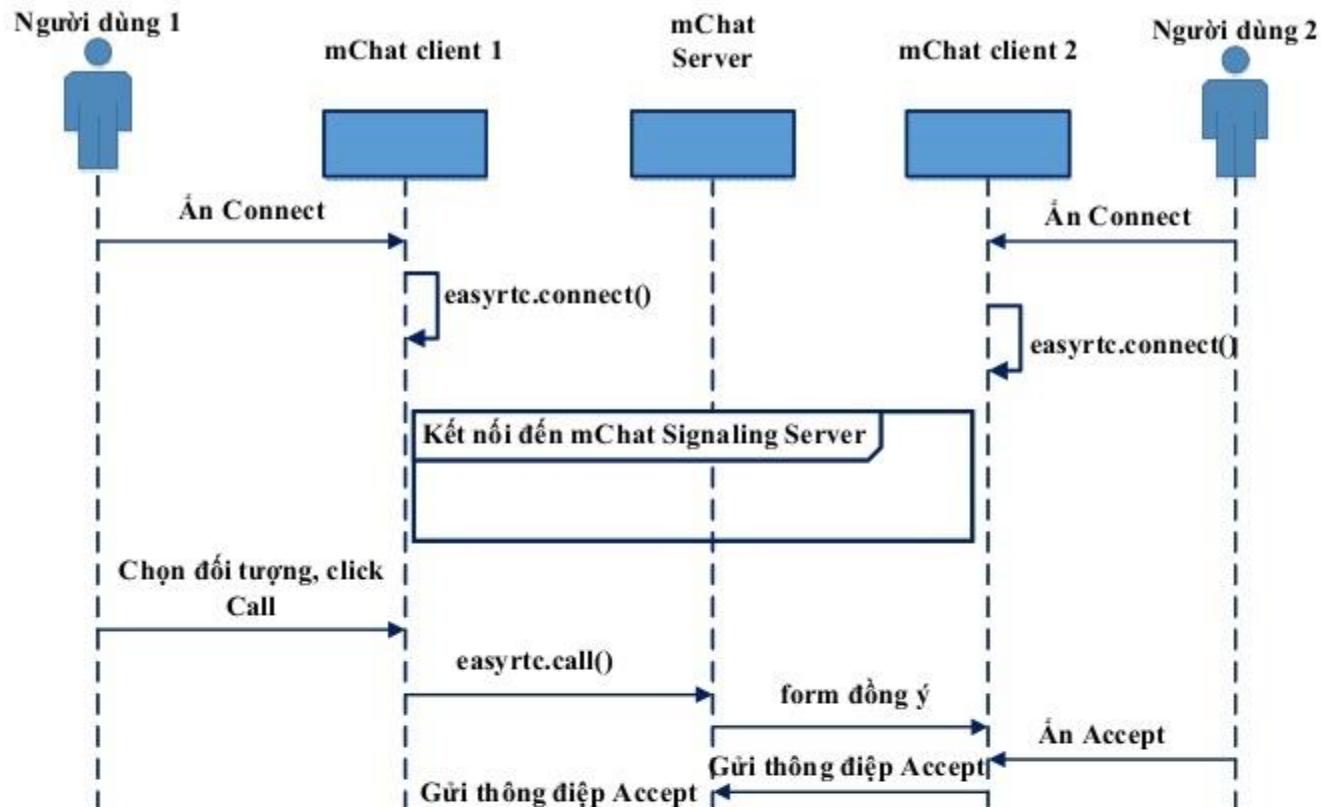
- Lớp giao diện người dùng: cung cấp các thành phần giúp người dùng tương tác với hệ thống, hiển thị các thông tin cần thiết đối với người dùng

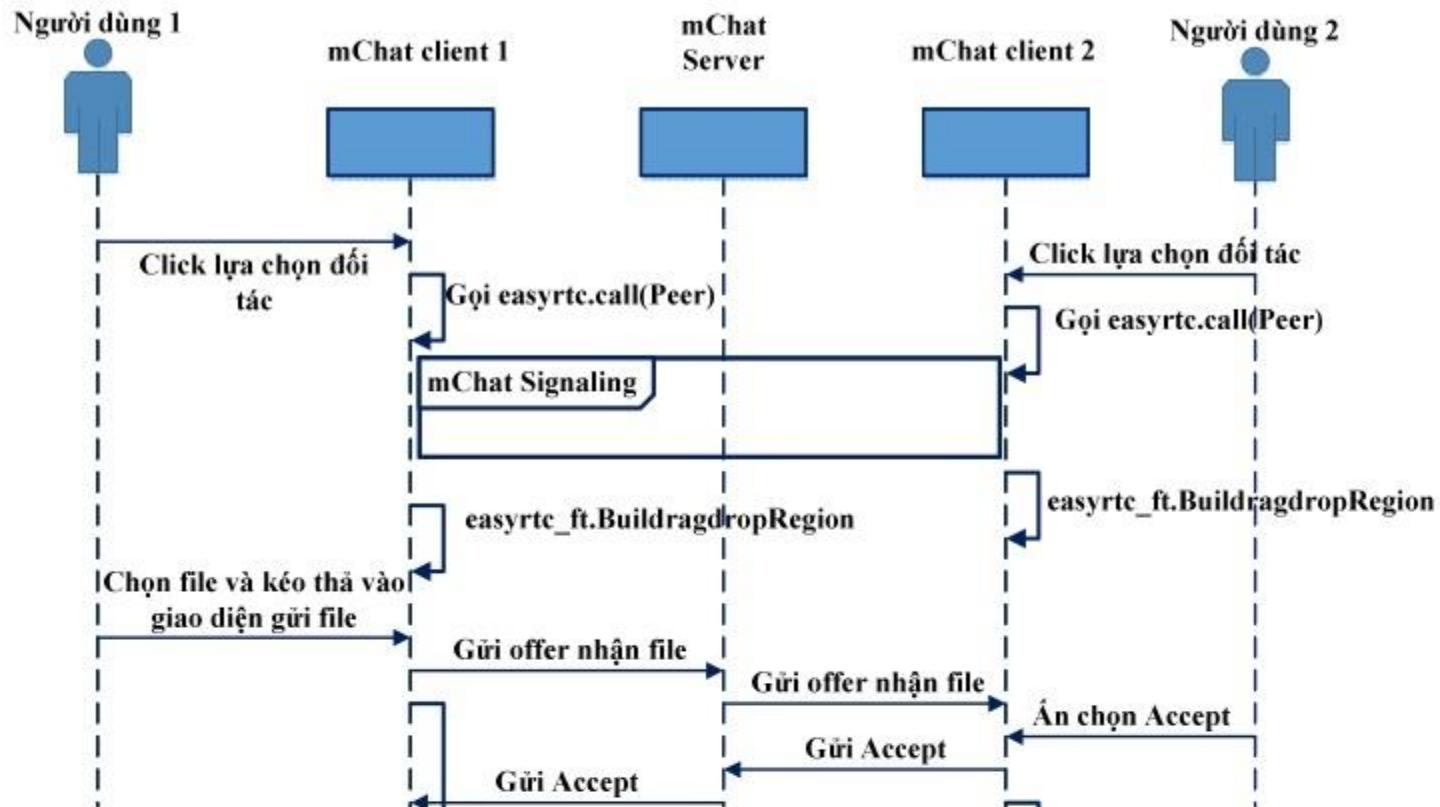


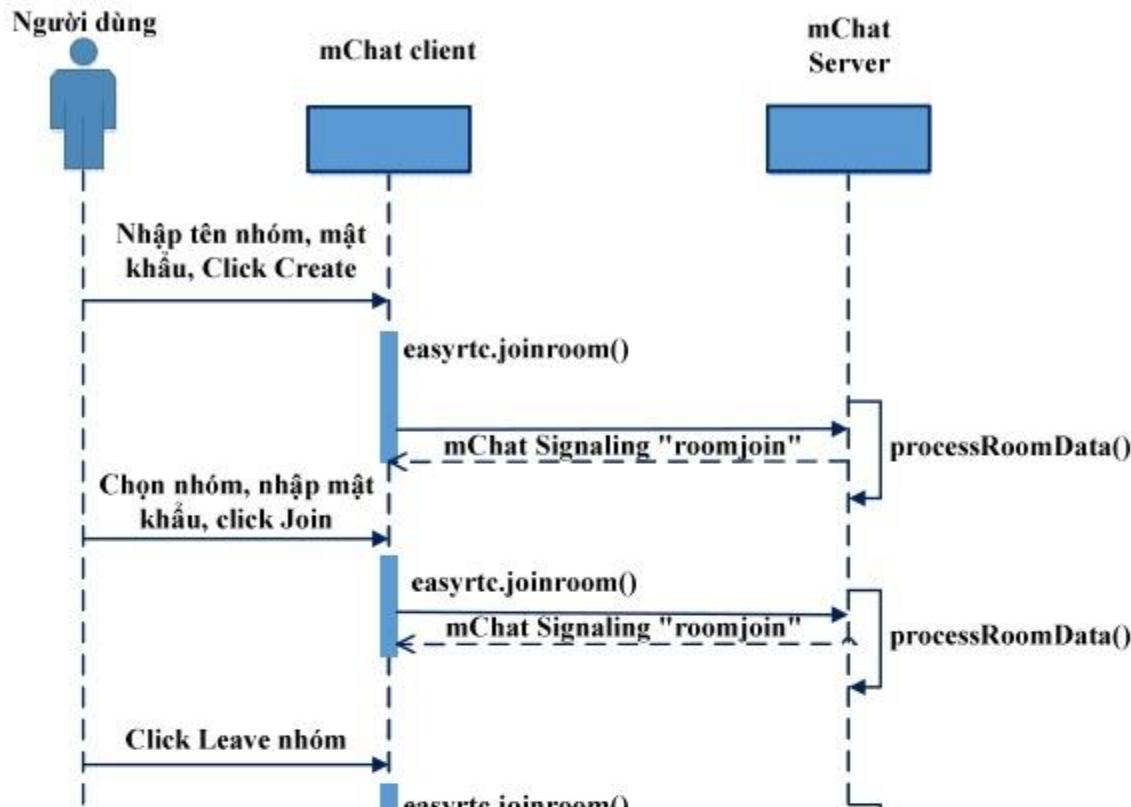
- Module quản lý, hỗ trợ gửi/nhận message P2P giữa các mChat client.

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









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

The screenshot shows the MobiFone mVAS Collaboration platform. At the top, there is a yellow header bar with the MobiFone logo and the text "TỔNG CÔNG TY VIỄN THÔNG MOBIFONE TRUNG TÂM DỊCH VỤ ĐPT & GTGT". Below the header, there is a blue navigation bar with links for "mVAS COLLABORATION", "Private Chat", "Voice Chat", "Group Collaboration", and "Logout". A search bar with the placeholder "Enter username for searching..." is also present. On the right side of the screen, there is a user profile for "Nguyễn Việt Thắng" (Phong Ky thuat) from "Trung tam DPT & GTGT". The main content area displays a list of messages. One message from "Me" says "Chào Trang" at 10:10:22 GMT+0700 on Oct 24, 2016. Another message from "Doan Minh Trang" says "Chào anh Thắng" at 10:10:40 GMT+0700 on Oct 24, 2016.

**mobifone**  
KẾT NỐI GIÁ TRỊ - KHỞI ĐẦU TIỀM NĂNG

TỔNG CÔNG TY VIỄN THÔNG MOBIFONE  
TRUNG TÂM DỊCH VỤ DPT & GTGT

MVAS COLLABORATION      Private Chat      Voice Chat      Group Collaboration      Logout

Create Group    Join Group    Group members lookup

Please enter group name and password willing to join:

Select a group: default    Refresh    Password: \*\*\*\*\*    Join

My groups	Group's Members	Group Chat
<input checked="" type="checkbox"/> default <span style="color: #ccc;">( Create Mon Oct 24 2016 11:50:08 GMT+0700 (SE Asia Standard Time) )</span> <a href="#">Leave</a>	 Nguyễn Việt Thắng  Doan Minh Trang	<b>Doan Minh Trang</b> [ In room= TCHC ] <span style="color: #ccc;">( Mon Oct 24 2016 11:50:08 GMT+0700 (SE Asia Standard Time) )</span> Chào các bạn
<input checked="" type="checkbox"/> TCHC <span style="color: #ccc;">( Create Mon Oct 24 2016 11:50:08 GMT+0700 (SE Asia Standard Time) )</span> <a href="#">Leave</a>		<b>Me</b> <span style="color: #ccc;">( Mon Oct 24 2016 11:50:08 GMT+0700 (SE Asia Standard Time) )</span> Sắp tới chúng ta có hội nghị MVAS day
		<b>Me</b> <span style="color: #ccc;">( Mon Oct 24 2016 11:50:08 GMT+0700 (SE Asia Standard Time) )</span> Tất cả cần tập trung để triển khai công việc đã được giao

(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 công 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 công. Đây là một thách thức nói chung với

- 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:

sẽ 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.

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	Ghi chú
-----------	-------	-------	----------	------------------------	---------

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 mở 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.

## 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 WebRTC 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

- 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 Sergienko (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. Bob Moeller (2013), Getting Started with WebRTC, Packt Publishing Ltd, UK

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://webrtcchacks.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>