**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH**



**BÁO CÁO BÀI TẬP LỚN**

**ĐỀ TÀI:**

**Xây dựng web gọi điện hội thảo trực tuyến**

**sử dụng WebRTC**

Giảng viên hướng dẫn: **GV Nguyễn Lê Minh**

Sinh viên thực hiện: Nguyễn Hoàng Nhật

Lớp: CQ.59.CNTT

Khoá: 59

TP. Hồ Chí Minh, năm 2021

**LỜI CẢM ƠN**

Trong thời gian làm đồ án tốt, em đã nhận được nhiều sự giúp đỡ, đóng góp ý kiến và chỉ bảo nhiệt tình của thầy cô, gia đình và bạn bè.

Em xin gửi lời cảm ơn chân thành đến Th.S Nguyễn Lê Minh, giảng viên Bộ môn Công Nghệ Thông Tin - trường ĐH Giao Thông Vận Tải Phân Hiệu Tại TP.HCM người đã tận tình hướng dẫn, chỉ bảo em trong suốt quá trình làm đồ án

Em cũng xin chân thành cảm ơn các thầy cô giáo trong trường ĐH Giao Thông Vận Tải Phân Hiệu Tại TP.HCM nói chung, các thầy cô trong Bộ môn Công Nghệ Thông Tin nói riêng đã dạy dỗ cho em kiến thức về các môn đại cương cũng như các môn chuyên ngành, giúp em có được cơ sở lý thuyết vững vàng và tạo điều kiện giúp đỡ em trong suốt quá trình học tập.

Cuối cùng, em xin chân thành cảm ơn gia đình và bạn bè, đã luôn tạo điều kiện, quan tâm, giúp đỡ, động viên em trong suốt quá trình học tập và hoàn thành đồ án.

**NHẬN XÉT CỦA GIẢNG VIÊN**

**……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………….**

**Chữ kí của giảng viên**

**MỤC LỤC**

[**MỞ ĐẦU** 4](#_Toc26912885)

[**CHƯƠNG 1** **MỤC ĐÍCH VÀ Ý NGHĨA CỦA ĐỀ TÀI** 4](#_Toc26912886)

[**1.1.Mục đích** 4](#_Toc26912887)

[**CHƯƠNG 2.** **TỔNG QUAN VỀ WEBRTC** 4](#_Toc26912889)

[2.1.**Lịch sử WebRTC** 4](#_Toc26912890)

[2.2.**WebRTC là gì ?** 5](#_Toc26912891)

[2.3.**Những lợi ích của WebRTC** 8](#_Toc26912892)

[**2.4.WebRTC dùng để làm gì ?** 10](#_Toc26912893)

[2.5.**Kiến trúc WebRTC** 15](#_Toc26912894)

[2.6.**Các APIs trong WebRTC** 18](#_Toc26912895)

[**2.6.1.RTCPeerConnection API** 19](#_Toc26912896)

[**2.6.2.MediaStream** 20](#_Toc26912897)

[**2.6.3.RTCDataChannel** 22](#_Toc26912898)

[**2.7. Các tầng giao thức trong WebRTC 23**](#_Toc26912899)

[2.8. **WebRTC có an toàn không**?. 31](#_Toc26912900)

[2.9. **Những trở ngại của WebRTC** 31](#_Toc26912901)

# **MỞ ĐẦU**

# **CHƯƠNG 1 LỜI MỞ ĐẦU**

* 1. **Lý do chọn đề tài**

Với tình hình dịch bệnh Covid hiện nay, nhu cầu học online ngày càng nhiều. Về việc gọi hội thảo theo thời gian thực , trên thế giới đã có nhiều công nghệ, giao thức,ứng dụng khác nhau như Skype, Viber, Line, Zoom, …Tuy nhiên, vì nhiều lý do từ tốc độ,bảo mật an toàn và điều quan trọng nhất là sự tiện dụng, vẫn tiếp tục có các nghiên cứu để đơn giản hóa việc gọi điện,hỗ trợ người dùng một cách nhanh nhất mà không đòi hỏi phải thao tác quá nhiều hay cài thêm plugin hoặc ứng dụng trên máy.Cụ thể là vào năm 2011 ý tưởng của Google với mã nguồn mở tên là webRTC mục đích là giao tiếp thời gian thực với trình duyệt.

# **CHƯƠNG 2. TỔNG QUAN VỀ WEBRTC**

## 2.1. Lịch sử WebRTC

Ý tưởng phát triển WebRTC được nhóm kỹ sư chịu trách nhiệm cho Google Hangouts đưa ra từ tận năm 2009. Vào thời gian đó, để truyền tải video, hình ảnh trên web thì người ta thường phải xài đến Flash. Nhóm kỹ sơ Hangouts lại không muốn sử dụng công nghệ này, và họ bắt đầu tự làm một chuẩn riêng cho mình. Đến năm 2010, Google thâu tóm hai công ty On2 và Global IP Solutions (GIPS) để lấy công nghệ truyền dữ liệu thời gian thực làm nền tảng cho WebRTC về sau.  
  
 Vào tháng 5/2011, Google ra mắt một dự án nguồn mở dành cho việc giao tiếp thời gian thực giữa trình duyệt với nhau, và từ lúc này dự án mang tên WebRTC. Song song đó, Hiệp hội World Wide Web (W3C) và Hiệp hội Kĩ sư quốc tế (IETF) cũng đang phát triển một số giao thức để dùng cho việc việc kết nối thời gian thực, thế nên họ bắt tay nhau tiếp tục hoàn thiện để rồi quyết định kết hợp chung vào WebRTC.  
  
 Đến 27/10/2011, W3C ra mắt bản nháp đầu tiên của WebRTC. Tháng 11/2011, Chrome 23 ra mắt, trở thành trình duyệt đầu tiên có tích hợp WebRTC ngay từ bên trong. Và tính đến thời mà mình viết bài này thì WebRTC vẫn còn đang tiếp tục được phát triển chứ chưa hoàn thiện một cách chính thức.

## 2.2. WebRTC là gì ?

*Theo Wikipedia, WebRTC là viết tắt của cụm từ Web Real-Time Communication. Là một web API được phát triển bởi World Wide Web Consortium (W3C), khả năng hỗ trợ trình duyệt (browser) giao tiếp với nhau thông qua VideoCall, VoiceCall hay transfer data Peer-to-Peer (P2P) mà không cần browser phải cài thêm plugins hay phần mềm hỗ trợ nào từ bên ngoài.*

Hiểu một cách đơn giản hơn về WebRTC thì nó không chỉ là một sản phẩm hay một hàm API duy nhất. Nó là cả một tập hợp rất nhiều các hàm có thể được lập trình viên sử dụng cho nhiều mục đích khác nhau. Có hàm chỉ để làm những việc đơn giản như đòi quyền truy cập vào webcam và microphone của máy tính, có hàm phức tạp hơn thì để thiết lập kết nối giữa hai người dùng với nhau, có hàm còn dùng để chia sẻ màn hình với người khác. Và rồi có hàm để hai người gọi video cho nhau, cũng là chức năng "nổi tiếng" nhất của WebRTC tính đến thời điểm hiện tại.

Tuy nhiên, tất cả mọi hàm lập trình nằm trong bộ API có một điểm chung vô cùng quan trọng: chúng thực thi hầu hết các tác vụ theo thời gian thực. Đó là lý do vì sao chữ Real-Time xuất hiện trong cái tên của bộ hàm này. Và nó không chỉ được dùng cho việc gọi video giữa hai trình duyệt mà người ta còn có thể làm nhiều chuyện khác, miễn là chuyện đó có liên quan đến việc làm cho hai hoặc nhiều người dùng liên lạc với nhau.

Trên trang web của mình, WebRTC cho biết họ được hỗ trợ chính thức bởi Google, Mozilla, Opera cùng nhiều đơn vị khác. Mục đích cuối cùng của dự án này là nhằm "mang lại các ứng dụng phong phú, chất lượng cao và chạy theo thời gian thực có thể được phát triển bởi lập trình viên cho các trình duyệt, nền tảng di động, thiết bị Internet of Things, và cho phép tất cả bọn chúng liên lạc với nhau thông qua một bộ các giao thức chung". 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 , 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.

Để sử dụng các hàm lập trình WebRTC, các lập trình viên có thể xài rất nhiều loại ngôn ngữ lập trình quen thuộc: nếu như viết trang web thì họ được quyền xài JavaScript, nếu làm app cho Android thì dùng Java, viết cho iOS thì dùng Objective-C, còn viết app cho Windows thì dùng C++.

Cũng có thể bạn sẽ nghe nói đến CU-RTC-Web. Đây là một phần mở rộng được Microsoft "cống hiến" cho WebRTC. Nó viết tắt cho cụm từ Customizable, Ubiquitous Real Time Communication over the Web. Giải thích thêm về đóng góp của mình, Microsoft cho biết tính "tùy biến" của nó nằm ở chỗ các ứng dụng có thể phản hồi theo thời gian thực với chất lượng của đường truyền. Ví dụ, khi tốc độ mạng bị giảm đi, lập trình viên có thể ra lệnh cho ứng dụng nền web của mình chuyển sang dùng kênh âm thanh thay cho kênh hình ảnh, thậm chí ngừng hoạt động đến khi tín hiệu tốt trở lại. Còn thuộc tính "mọi lúc mọi nơi" (Ubiquitous) có nghĩa là người dùng sẽ giao tiếp được với bạn bè của mình mặc cho trình duyệt và thiết bị sử dụng khác nhau. CU-RTC-Web sẽ hoạt động tốt trên cơ sở hạ tầng mạng hiện tại để đảm bảo tính tương thích cao.

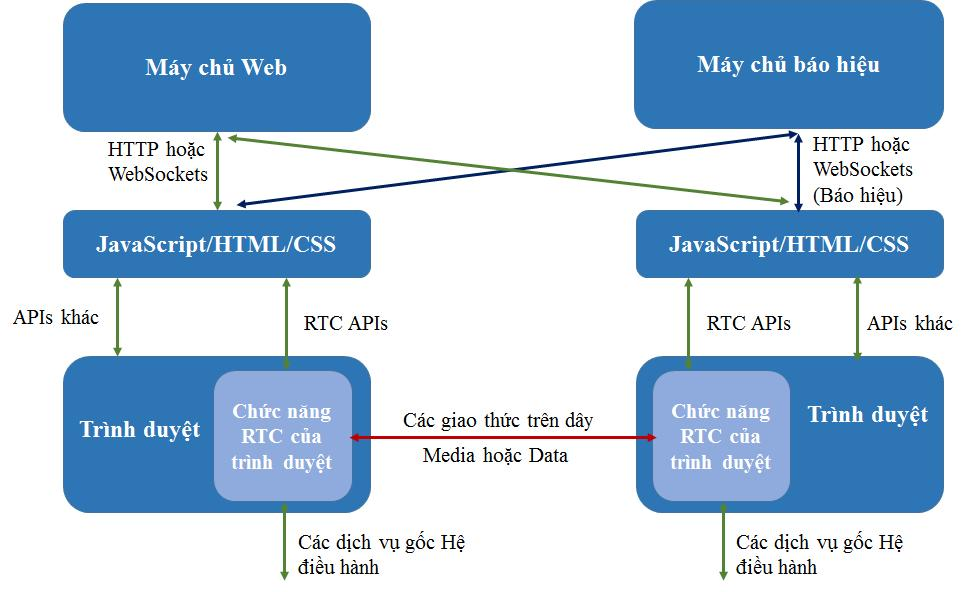
## 2.3. Những lợi ích của WebRTC

* **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
* **Built-in security**: WebRTC quy định mọi dữ liệu truyền P2P đều được bảo mật và mã hóa.

## 2.5. Kiến trúc WebRTC

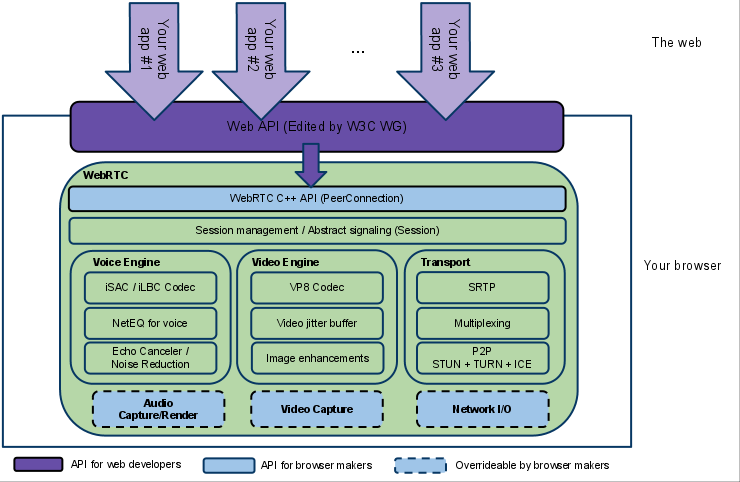
Kiến trúc web cổ điển dựa trên mô hình client-server, trong đó trình duyệt gửi yêu cầu HTTP đến máy chủ để lấy nội dung, máy chủ trả lời, gửi nội dung về cho trình duyệt dưới dạng HTML, thường kèm theo JavaScript và Cascading Style Sheets (CSS). Trong trường hợp đơn giản thì khi máy chủ web trả lời yêu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

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



*Hình 2.4 Truyền thông thời gian thực trong trình duyệt*

Hình 2.4 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. 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.



*2.4.1Kiến trúc tổng thể WebRTC*

Trong kiến trúc WebRTC có 3 lớp API:

* APIs cho nhà lập trình web: lớp này chứa tất cả các APIs mà nhà lập trình web cần, bao gồm các đối tượng chính là RTCPeerConnection,
* RTCDataChannel, MediaStream (chi tiết mô tả ở mục 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

Trong hình 2.4.1, thành phần video engine là framework xử lý chuỗi video từ camera đến mạng và từ mạng ra màn hình. Trong đó, video codec sử dụng VP8 (một dạng nén video mở, miễn phí sở hữu bởi Google và tạo ra bởi On2 Technologies) và VP9, hỗ trợ tính năng Video jitter buffer để giúp ẩn đi những ảnh hưởng của jitter và việc mất gói trong chất lượng video tổng thể, hỗ trợ nâng cao chất lượng ảnh như khử nhiễu ảnh được chụp từ webcam.

Thành phần audio engine là framework xử lý chuỗi audio từ card âm thanh đến mạng. Thành phần này sử dụng codec iSAC (internet Speech Audio Codec) /iLBC (Internet Low Bitrate Codec)/Opus . Nó sử dụng bộ đệm jitter động, thuật toán giấu lỗi để ẩn những ảnh hưởng của jitter mạng và mất gói tin, giúp giảm độ trễ tối đa mà vẫn giữ được chất lượng voice cao nhất. Trong framework sử dụng Acoustic Echo Canceler, phần mềm dựa trên thành phần xử lý tín hiệu giúp loại bỏ âm vọng trong thời gian thực và sử dụng Noise Reduction, phần mềm dựa trên thành phần xử lý tín hiệu giúp loại bỏ những tiếng ồn nền thường gắn với VoIP (hiss, fan noise).Trong kiến trúc này, thành phần vận chuyển/quản lý phiên rất quan trọng, cho phép thiết lập và quản lý kết nối P2P qua các loại mạng khác nhau. Các nhiệm vụ, giao thức trong này như SRTP, STUN/TURN/ICE, quản lý phiên sẽ được nêu chi tiết ở mục 2.4. Các tầng giao thức trong WebRTCs.

## 2.6. 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 sWebRTC thực hiện các công việc chính bao gồm:

Lấy dữ liệu audio, video hoặc dữ liệu khác trên máy.

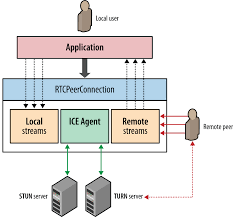
* Lấy thông tin mạng như địa chỉ IP, port và trao đổi thông tin này với WebRTC client (gọi là Peer) để bắt đầu thiết lập kết nối, kể cả qua NATs và Firewall.
* Điều phối giao tiếp báo hiệu để báo cáo lỗi, khởi tạo hoặc đóng phiên kết nối.
* Trao đổi thông tin về khả năng hỗ trợ media của từng Peers như 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.

### **2.6.1. RTCPeerConnection API**

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



*2.5.1. RTCPeerConnection API*

Những nhiệm vụ chính của RTCPeerConnection là

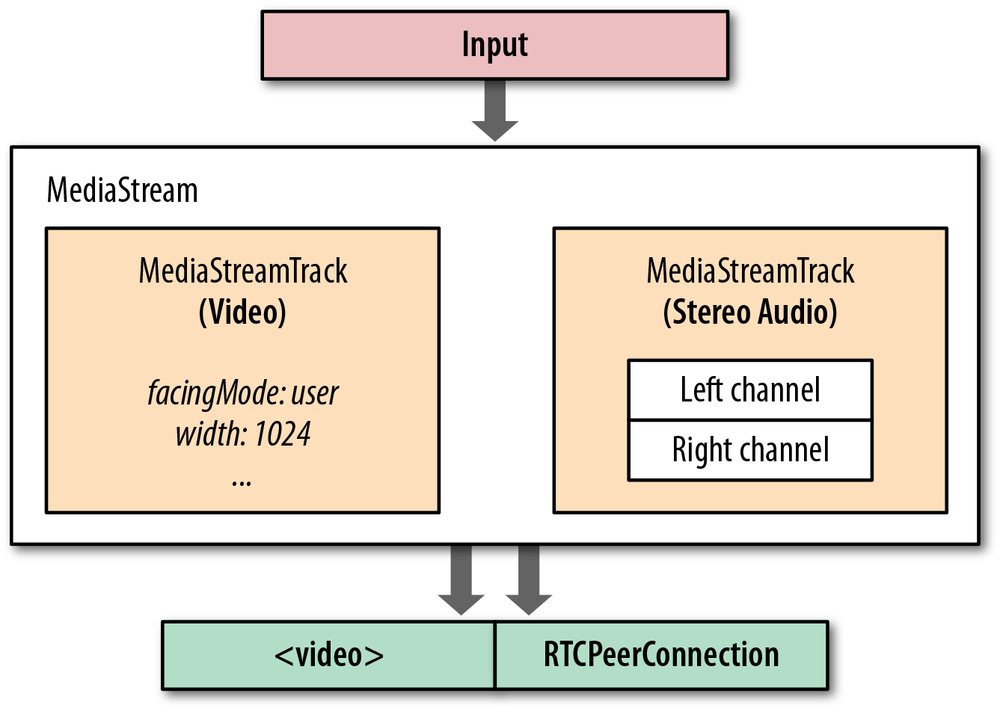
Điều khiển toàn bộ quá trình Interactive Connectivity Establishment (ICE) để vượt NAT.

* Gửi tự động bản tin STUN 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.

### **2.6.2. MediaStream**

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



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

### **2.6.3. RTCDataChannel**

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

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

Dòng dữ liệu gửi P2P được handle bằng cách sử dụng SCTP trên Datagram Transport Layer Security (DTLS) 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).

## 2.7. 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 2.7 dưới đây:



*Hình 2.7. Protocol stack trong WebRTC*

**SRTP**

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

**SCTP**

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

**SDP**

WebRTC sử dụng Session Description Protocol, SDP, được encode trong đối tượng RTCSessionDescription, để mô tả đặc tính media của hai đầu trong kết nối P2P như loại media đề truyền/nhận (audio, video, application data), network transports, loại codecs sử dụng và cấu hình, thông tin băng thông, và các thông tin metadata khác.Thông điệp SDP được trao đổi qua máy chủ báo hiệu hay còn gọi là được trao đổi qua kênh báo hiệu. Máy chủ báo hiệu có trách nhiệm gửi và nhận tất cả thông điệp đến tất cả các peers mà mong muốn kết nối đến peer khác. Mặc dù SDP là định dạng dữ liệu dùng để trao đổi, thống nhất thông số giữa kết nối Peer-to-Peer, nhưng do WebRTC không ràng buộc cho các SDP “offer” và “answer” giao tiếp như nào, nên nó không được thể hiện ở hình 2.7 ở trên. Tuy nhiên, mô hình offer/answer được thiết kế tuân thủ theo RFC3264 .

**DTLS**

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

**STUN**

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

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

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

- Hai trình duyệt muốn thiết lập kết nối trực tiếp phải gửi gói hole punching cùng thời điểm. Nhờ vậy cả hai trình duyệt mới nhận ra session cần thiết lập và biết các địa chỉ để gửi gói tin. Gói tin hole punching là gói tin IP thông thường được gửi để kiểm tra có đến được địa chỉ đích (sau NATs) không. Hai trình duyệt phải biết càng nhiều địa chỉ IP mà có thể sử dụng để kết nối đến peer càng tốt. Các địa chỉ này các địa chỉ nội bộ (trong NAT), địa chỉ public (ngoài NAT) và địa chỉ relay.

- Cả hai trình duyệt phải kết nối đến được địa chỉ IP Public của media relay.

- Dòng đối xứng phải được sử dụng. Lưu lượng UDP phải xuất hiện để hoạt động tương tự cách của kết nối TCP.

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

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

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.

# **CHƯƠNG 4 Các bước xây dựng Web Học Online**

### **4.1 Thiết lập báo hiệu lên server :**

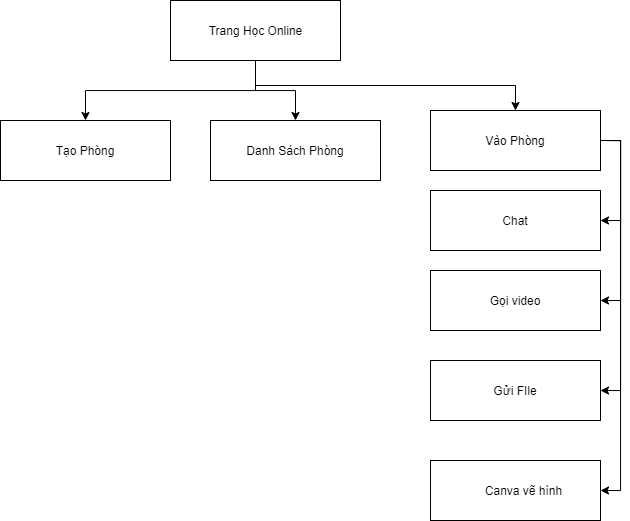
File WebRTCUTC2.js : là đối tượng quan trọng nhất, chứa các thuộc tính, phương thức chính cho việc viết ứng dụng bao gồm phương thức kết nối đến máy chủ báo hiệu, Phương thức tạo phòng, đặt listener cho thông điệp nhận từ máy chủ, Phương thức gửi tin nhắn,video call, file P2P, Phương thức gửi tin nhắn đến ứng dụng trên máy chủ, các phần liên quan đến xử lý lỗi (error handling)

### **4.1 Phân tích thiết kế hệ thống :**

Ứng dụng được xây dựng dựa trên những module chính sau:

* Module xác thực:
* Module gửi file:
* Module quản lý, hỗ trợ gửi message P2P giữa các client.
* Module quản lý, hỗ trợ gọi và nhận cuộc gọi audio streaming audio P2P giữa client.
* Module quản lý nhóm hay room cộng tác
* Module xử lý canva và trao đổi giữa các client

### **4.2 Sơ đồ phân rã chức năng :**



### **4.3 Kết quả :**

Link Web : http://hoconlineutc2.herokuapp.com/home/

**Trang Chủ :**

-Hiển thị danh sách phòng học. Đối với những mã phòng có hình ổ khóa phải cần mật khẩu mới vào được, trang chủ có chức năng tạo phòng hoặc vào phòng theo danh sách hoặc nhập mã phòng

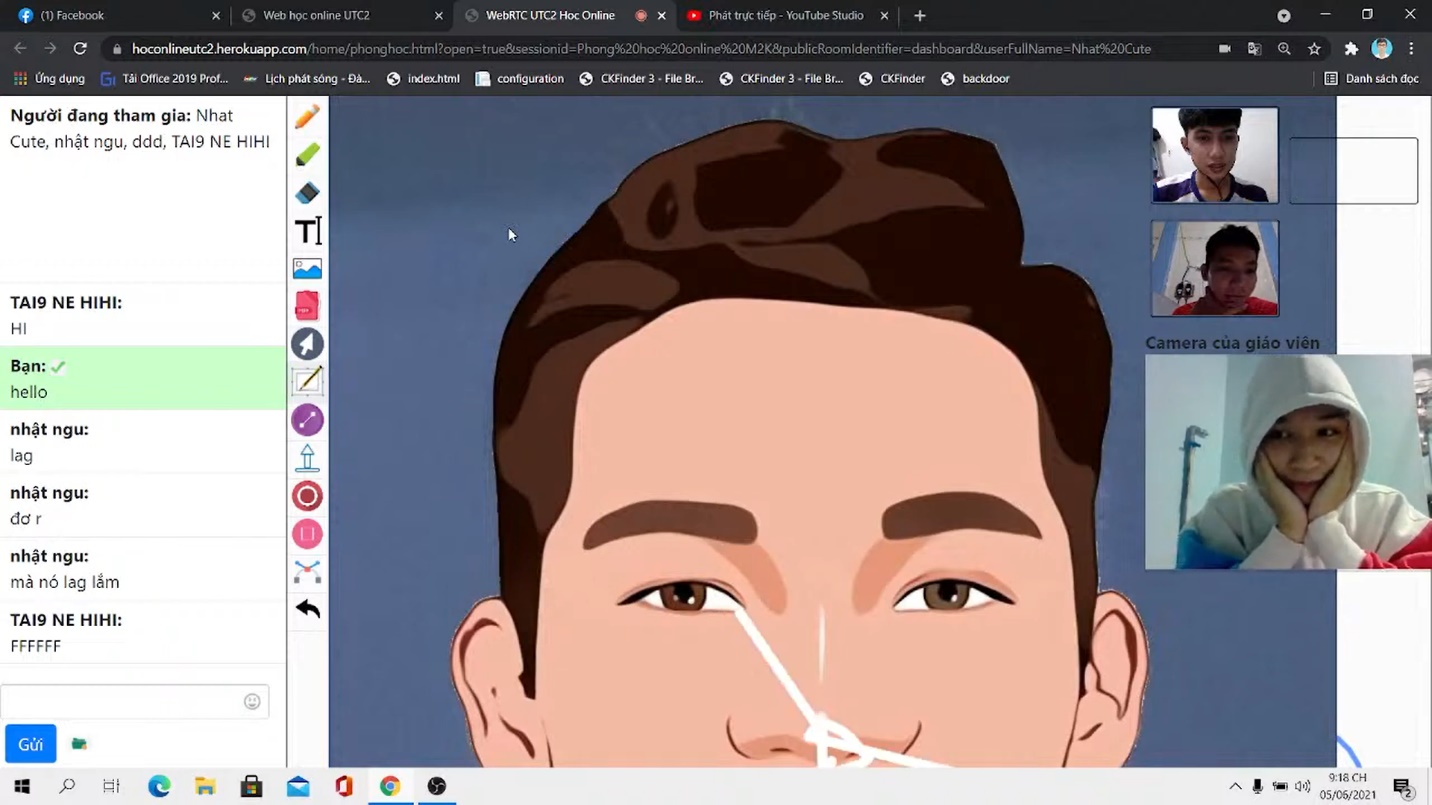
### 

**Trang phòng học** :

-Gọi video với các thành viên

-Chat và gửi file

- Vẽ, upload ảnh,file Pdf lên bảng ở chính giữa.

****

# **CHƯƠNG 5 Kết Luận**

### **5.1 Kết quả :**

Với yêu cầu của đề tài ứng dụng WebRTC trong việc xây dựng web học Online, đồ án thu được những 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
    - Phân tích yêu cầu, thiết kế ứng dụng, cài đặt thử nghiệm hệ thống cộng tác tại web đạt được những kết quả đáng khích lệ.

### **5.2 Hướng phát triển :**

Qua nghiên cứu WebRTC và thử nghiệm ứng dụng, em nhận thấu 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 1 số trình duyệt hiện tại , 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, 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.
    - 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 việc chia sẻ thành công nhưng kết hợp với canva không được tốt lắm
    - Nghiên cứu thêm về cách cài đặt tối ưu hiệu năng từng chức năng như máy chủ báo hiệu, lựa chọn phương án tối ưu trong trường hợp số lượng người dùng lên đến hơn 40 sinh viên