**ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT HÀN**

**KHOA KHOA HỌC MÁY TÍNH**



**BÁO CÁO**

**LẬP TRÌNH MẠNG**

**ĐỀ TÀI: XÂY DỰNG ỨNG DỤNG CHAT ROOM**

Sinh viên thực hiện : **PHAN THÀNH ĐẠT**

Giảng viên hướng dẫn : **TS NGUYỄN VŨ ANH QUANG**

Lớp. :**18IT1**

Đà nẵng, tháng 12 năm 2020

**ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

**VIỆT HÀN**

**KHOA KHOA HỌC MÁY TÍNH**

**BÁO CÁO**

**LẬP TRÌNH MẠNG**

**ĐỀ TÀI: XÂY DỰNG ỨNG DỤNG CHAT ROOM**

Đà Nẵng, tháng 01 năm 2020

**MỞ ĐẦU**

Thế giới ngày nay đã có nhiều phát triển mạnh mẽ về CNTT. Từ một tiềm năng thông tin đang trở thành một tài nguyên thật sự, trở một hàng hoá trong xã hội, tạo ra sự thay đổi to lớn trong lực lượng sản xuất, cơ sở hạ tầng, cấu trúc kinh tế, tính chất lao động và cả cách thức quản lí trong các lĩnh vực của xā hội. Trong những năm gần đây nền CNTT nước ta đã có nhiều phát triển trong mọi lĩnh vực của cuộc sống cũng như trong lĩnh vực quản lí xã hội. Một trong những lĩnh vực mà máy tính được sử dụng nhiều nhất là các hệ thống thông tin quản lí nói chung. Sự phát triển của internet, internet đã đưa con người vào kỉ nguyên xa lộ thông tin trên mạng. Thông tin qua mạng internet con người có thể mua bán hàng hóa, trao đổi thông tin rộng rãi trên toàn cầu. Công nghệ đã thúc đẩy mạnh mẽ việc khai thác thông tin trên mạng với tính năng linh hoạt và dễ sử dụng nó. Cùng với sự phát triển của ngành công nghệ thông tin thì máy tính càng trở nên gần gũi và thông dụng với cuộc sống của chúng ta. Khả năng hiệu quả hỗ trợ những công việc khó khăn và phức tạp trong mọi lĩnh vực của cuộc sông nói chung và công tác quản lí trong các doanh nghiệp, cơ quan nói riêng như bảo mật thông tin, xem và chỉnh sửa thông tin một cách dễ dàng, hiệu quả và nhanh chóng. Đã có nhiều phần mềm được xây dựng để phục vụ cho tính chất phức tạp trong mọi lĩnh vực của cuộc sống nói chung và công tác quản lý trong các doanh nghiệp và các cơ quan. Trong suốt quá trình học tập, chúng tôi đã được các thầy cô giáo cung cấp và truyền đạt những kiến thức cần thiết trong lĩnh vưc công nghệ thông tin. Ngoài ra chúng tôi còn đuợc rèn luyện một tinh thần học tập, làm việc độc lập sáng tạo. Trong khuôn khổ một đề án môn học dưới sự chỉ bảo tận tình của thây cô giáo, cùng sự tích lũy kiến thức của bản thân chúng tôi đã xây dựng đề tài " Chat room" bằng chương trình Java.

Lý do chọn đề tài: Hiện nay, Chat không còn xa lạ gì với thế hệ trẻ, các công ty, trường học, tập đoàn ..., Chat khá là phổ biến trong cả nước ta nói riêng và tất cả các nước trên thế giới nói chung. Để cho thuận tiện hơn trong công việc, giúp cho nhiều người có thể kết nối với nhau, trò chuyện, bàn bạc với nhau qua internet, chúng tôi quyết định làm phần mềm CallGroup.

**LỜI CẢM ƠN**

Em xin chân thành cảm ơn các thầy cô và các bạn đã dành thời gian giúp đỡ chúng em trong quá trình thực hiện đồ án này. Đặc biệt, chúng em xin chân thành cảm ơn thầy Huỳnh Công Pháp và Nguyễn Anh Tuấn là người đồng ý hướng dẫn trực tiếp cho đề tài của em. Là người tận tình giúp đỡ chúng em về thông tin của đồ án. Nhờ vậy mà em đã hoàn chỉnh được đồ án của mình và quan trọng hơn hết là em đã tiếp thu được những kinh nghiệm trong suốt quá trình thực hiện đồ án.

Tuy nhiên, do thời gian có hạn nên em không thể phát huy hết những ý tưởng, khả năng hổ trợ của ngôn ngữ và kỹ thuật lập trình vào đề tài. Trong quá trình xây dựng website, không thể tránh khỏi những sai xót, mong nhận được sự đóng góp và cảm thông của quí thầy cô.

Em xin chân thành cảm ơn.

.

**NHẬN XÉT**

Chữ ký xác nhận của Giảng viên

**MỤC LỤC**

[Chương 1 TỔNG QUAN VỀ LẬP TRÌNH MẠNG 1](#_Toc60143123)

[1.1 Mạng máy tính 1](#_Toc60143124)

[1.2 Mô hình mạng 1](#_Toc60143125)

[*1.2.1* *Mạng hình sao (Star Network)* 1](#_Toc60143126)

[*1.2.2* *Mạng tuyến tính (Bus Network)* 1](#_Toc60143127)

[*1.2.3* *Mạng hình vòng (Ring Network)* 2](#_Toc60143128)

[1.3 Phân loại mạng 2](#_Toc60143129)

[*1.3.1* *LAN* 2](#_Toc60143130)

[*1.3.2* *MAN* 3](#_Toc60143131)

[*1.3.3* *WAN* 3](#_Toc60143132)

[1.4 Mô hình mạng OSI 3](#_Toc60143133)

[1.5 Giao thức truyền thông 5](#_Toc60143134)

[Chương 2 WEBRTC QUA WEBSOCKET TRONG NODEJS 6](#_Toc60143135)

[2.1 Tổng quan về WebRTC 6](#_Toc60143136)

[*2.1.1* *Giới thiệu* 6](#_Toc60143137)

[*2.1.2* *Kiến trúc WebRTC* 6](#_Toc60143138)

[*2.1.3* *Một số giao thức trong WebRTC* 7](#_Toc60143139)

[*2.1.4* *Các phần chính của WebRTC* 9](#_Toc60143140)

[*2.1.5* *Cách thức hoạt động của giao thức WebRTC* 10](#_Toc60143141)

[2.2 Giao thức TCP 11](#_Toc60143142)

[*2.2.1* *Khái niệm* 11](#_Toc60143143)

[*2.2.2* *Thiết lập kênh truyền* 11](#_Toc60143144)

[*2.2.3* *Truyền dữ liệu bằng giao thức TCP* 12](#_Toc60143145)

[*2.2.4* *Khái niệm về Socket* 13](#_Toc60143146)

[*2.2.5* *Sử dụng TCP socket* 13](#_Toc60143147)

[2.3 WebSocket là gì? 14](#_Toc60143148)

[*2.3.1* *Giới thiệu* 14](#_Toc60143149)

[*2.3.2* *Hoạt động của WebSocket* 15](#_Toc60143150)

[*2.3.3* *Thư viện Socket.IO* 16](#_Toc60143151)

[2.4 Các bước và thuật toán xây dựng chương trình Client-Server 17](#_Toc60143152)

[*2.4.1* *Các bước xây dựng chương trình server* 17](#_Toc60143153)

[*2.4.2* *Các bước xây dựng chương trình client* 17](#_Toc60143154)

[*2.4.3* *Thuật toán xây dựng chương trình client-server tổng quát* 17](#_Toc60143155)

[Chương 3 TRIỂN KHAI VÀ XÂY DỰNG 18](#_Toc60143156)

[3.1 Bài toán chat room 18](#_Toc60143157)

[3.2 Giải quyết bài toán 18](#_Toc60143158)

[3.3 Tổng quan về NodeJs 18](#_Toc60143159)

[*3.3.1* *NodeJs là gì?* 18](#_Toc60143160)

[*3.3.2* *Các đặc tính của NodeJs* 19](#_Toc60143161)

[*3.3.3* *Hai NodeJS Framework sử dụng phổ biến* 20](#_Toc60143162)

[3.4 Tổng quan hệ thống 20](#_Toc60143163)

[*3.4.1* *Mô hình hoạt động* 20](#_Toc60143164)

[*3.4.2* *Biểu đồ hoạt động* 21](#_Toc60143165)

[3.5 Demo giao diện 22](#_Toc60143166)

[KẾT LUẬN 24](#_Toc60143167)

[1. Kết quả 24](#_Toc60143168)

[2. Hạn chế 24](#_Toc60143169)

[3. Hướng phát triển 24](#_Toc60143170)

[DANH MỤC TÀI LIỆU THAM KHẢO 25](#_Toc60143171)

**DANH MỤC CÁC HÌNH**

[*Hình 1.1 Mô hình mạng hình sao* 1](#_Toc60143735)

[*Hình 1.2 Mô hình mạng tuyến tính* 2](#_Toc60143736)

[*Hình 1.3 Mô hình mạng hình vòng* 2](#_Toc60143737)

[*Hình 1.4 Mô hình 7 tầng OSI* 4](#_Toc60143738)

[*Hình 2.5 Quá trình bắt tay 3 bước* 11](file:///E:\callgroup\bao_cao_do_an_4.docx#_Toc60143739)

[*Hình 2.6 Minh họa việc truyền dữ liệu bởi giao thức TCP* 12](#_Toc60143740)

[*Hình 2.7 Mô tả Socket trong mô hình TCP/IP* 13](#_Toc60143741)

[*Hình 2.8 Socket trong mô hình ứng dụng client-server* 13](#_Toc60143742)

[*Hình 2.9 Các hoạt động của TCP socket* 14](file:///E:\callgroup\bao_cao_do_an_4.docx#_Toc60143743)

[*Hình 2.10 Mô hình hoạt động của WebSocket* 15](#_Toc60143744)

[*Hình 2.11 Thuật toán xây dựng chương trình Server* 17](file:///E:\callgroup\bao_cao_do_an_4.docx#_Toc60143745)

[*Hình 2.12 Thuật toán xây dựng chương trình Client* 17](file:///E:\callgroup\bao_cao_do_an_4.docx#_Toc60143746)

[*Hình 2.13 Thuật toán xây dựng chương trình client-server tổng quát* 18](#_Toc60143747)

[*Hình 3.1 Các thành phần quan trọng của NodeJs* 19](#_Toc60143748)

[*Hình 3.2 Mô hình hoạt động* 20](#_Toc60143749)

[*Hình 3.3 Biểu đồ hoạt động tham gia phòng* 21](#_Toc60143750)

[*Hình 3.4 Giao diện trang tạo/tham gia phòng* 22](#_Toc60143751)

[*Hình 3.5 Giao diện phòng chat* 22](#_Toc60143752)

[*Hình 3.6 Giao diện phòng chat* 23](#_Toc60143753)

[*Hình 3.7 Giao diện phòng chat* 23](#_Toc60143754)

# TỔNG QUAN VỀ LẬP TRÌNH MẠNG

## Mạng máy tính

Mạng máy tính hay hệ thống mạng (Computer Network hay Network System) là sự kết hợp các máy tính lại với nhau thông qua các thiết bị nối kết mạng và phương tiện truyền thông (giao thức mạng, môi trường truyền dẫn) theo một cấu trúc nào đó và các máy tính này trao đổi thông tin qua lại

## Mô hình mạng

### *Mạng hình sao (Star Network)*

Có tất cả các trạm được kết nối với một thiết bị trung tâm có nhiệm vụ nhận tín hiệu từ các trạm và chuyển đến trạm đích. Tùy theo yêu cầu truyền thông trên mạng mà thiết bị trung tâm có thể là hub, switch, router hay máy chủ trung tâm. Vai trò của thiết bị trung tâm là thiết lập các liên kết Point – to – Point.

* Ưu điểm: Thiết lập mạng đơn giản, dễ dàng cấu hình lại mạng (thêm, bớt các trạm), dễ dàng kiểm soát và khắc phục sự cố, tận dụng được tối đa tốc độ truyền của đường truyền vật lý.
* Khuyết điểm: Độ dài đường truyền nối một trạm với thiết bị trung tâm bị hạn chế (bán kính khoảng 100m với công nghệ hiện nay).

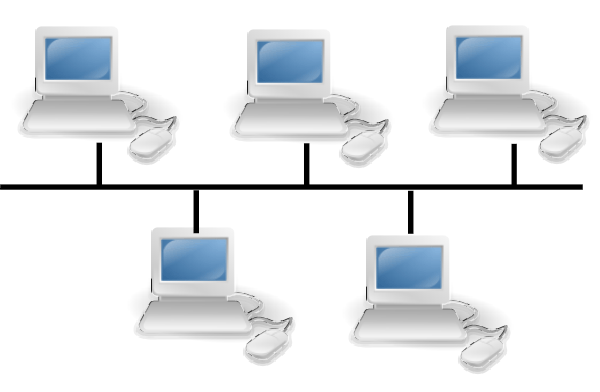
**

*Hình 1.1 Mô hình mạng hình sao*

### *Mạng tuyến tính (Bus Network)*

Có tất cả các trạm phân chia trên một đường truyền chung (bus). Đường truyền chính được giới hạn hai đầu bằng hai đầu nối đặc biệt gọi là terminator. Mỗi trạm được nối với trục chính qua một đầu nối chữ T (T-connector) hoặc một thiết bị thu phát (transceiver). Mô hình mạng Bus hoạt động theo các liên kết Point–to–Multipoint hay Broadcast.

* Ưu điểm: Dễ thiết kế và chi phí thấp.
* Khuyết điểm: Tính ổn định kém, chỉ một nút mạng hỏng là toàn bộ mạng bị ngừng hoạt động.

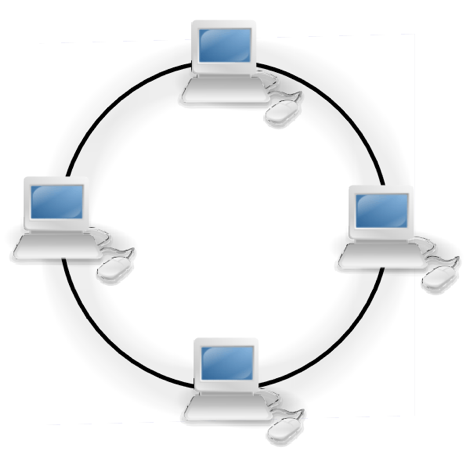


*Hình 1.2 Mô hình mạng tuyến tính*

### *Mạng hình vòng (Ring Network)*

Tín hiệu được truyền đi trên vòng theo một chiều duy nhất. Mỗi trạm của mạng được nối với nhau qua một bộ chuyển tiếp (repeater) có nhiệm vụ nhận tín hiệu rồi chuyển tiếp đến trạm kế tiếp trên vòng. Như vậy tín hiệu được lưu chuyển trên vòng theo một chuỗi liên tiếp các liên kết Point–to–Point giữa các repeater.

* Ưu điểm: Mạng hình vòng có ưu điểm tương tự như mạng hình sao.
* Khuyết điểm: Một trạm hoặc cáp hỏng là toàn bộ mạng bị ngừng hoạt động, thêm hoặc bớt một trạm khó hơn, giao thức truy nhập mạng phức tạp.



*Hình 1.3 Mô hình mạng hình vòng*

## Phân loại mạng

### *LAN*

LAN (local area network), hay còn gọi là "mạng cục bộ", là mạng tư nhân trong một toà nhà, một khu vực (trường học hay cơ quan chẳng hạn) có cỡ chừng vài km. Chúng nối các máy chủ và các máy trạm trong các văn phòng và nhà máy để chia sẻ tài nguyên và trao đổi thông tin. LAN có 3 đặc điểm:

* Giới hạn về tầm cỡ phạm vi hoạt động từ vài mét cho đến 1 km.
* Thường dùng kỹ thuật đơn giản chỉ có một đường dây cáp (cable) nối tất cả máy. Vận tốc truyền dữ liệu thông thường là 10 Mbps, 100 Mbps, 1 Gbps, và gần đây là 100 Gbps.
* Ba kiến trúc mạng kiểu LAN thông dụng bao gồm:
* Mạng bus hay mạng tuyến tính. Các máy nối nhau một cách liên tục thành một hàng từ máy này sang máy kia. Ví dụ của nó là Ethernet (chuẩn IEEE 802.3).
* Mạng vòng. Các máy nối nhau như trên và máy cuối lại được nối ngược trở lại với máy đầu tiên tạo thành vòng kín. Thí dụ mạng vòng thẻ bài IBM (IBM token ring).
* Mạng sao.

### *MAN*

MAN (metropolitan area network), hay còn gọi là "mạng đô thị", là mạng có cỡ lớn hơn LAN, phạm vi vài km. Nó có thể bao gồm nhóm các văn phòng gần nhau trong thành phố, nó có thể là công cộng hay tư nhân và có đặc điểm:

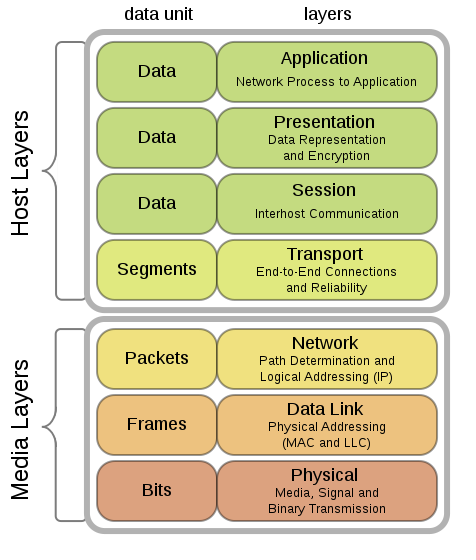
* Chỉ có tối đa hai dây cáp nối.
* Không dùng các kỹ thuật nối chuyển.
* Có thể hỗ trợ chung vận chuyển dữ liệu và đàm thoại, hay ngay cả truyền hình. Ngày nay người ta có thể dùng kỹ thuật cáp quang để truyền tín hiệu. Vận tốc có hiện nay thể đạt đến 10 Gbps.

### *WAN*

WAN (wide area network), còn gọi là "mạng diện rộng", dùng trong vùng địa lý lớn thường cho quốc gia hay cả lục địa, phạm vi vài trăm cho đến vài ngàn km. Chúng bao gồm tập hợp các máy nhằm chạy các chương trình cho người dùng. Các máy này thường gọi là máy lưu trữ(host) hay còn có tên là máy chủ, máy đầu cuối. Các máy chính được nối nhau bởi các mạng truyền thông con (communication subnet) hay gọn hơn là mạng con (subnet). Nhiệm vụ của mạng con là chuyển tải các thông điệp (message) từ máy chủ này sang máy chủ khác.

## Mô hình mạng OSI

Mô hình OSI (Open Systems Interconnection Reference Model, viết ngắn là OSI Model hoặc OSI Reference Model) - tạm dịch là Mô hình tham chiếu kết nối các hệ thống mở - là một thiết kế dựa vào nguyên lý tầng cấp, lý giải một cách trừu tượng kỹ thuật kết nối truyền thông giữa các máy vi tính và thiết kế giao thức mạng giữa chúng. Mô hình này được phát triển thành một phần trong kế hoạch Kết nối các hệ thống mở (Open Systems Interconnection) do ISO và IUT-T khởi xướng. Nó còn được gọi là Mô hình bảy tầng của OSI.



*Hình 1.4 Mô hình 7 tầng OSI*

Tường trình các tầng cấp của mẫu hình OSI:

* Tầng vật lý (Physical Layer): Tầng vật lý định nghĩa tất cả các đặc tả về điện và vật lý cho các thiết bị. Trong đó bao gồm bố trí của các chân cắm(pin), các hiệu điện thế, và các đặc tả về cáp nối (cable).
* Tầng liên kết dữ liệu (Data-Link Layer): Tầng liên kết dữ liệu cung cấp các phương tiện có tính chức năng và quy trình để truyền dữ liệu giữa các thực thể mạng (truy cập đường truyền, đưa dữ liệu vào mạng), phát hiện và có thể sửa chữa các lỗi trong tầng vật lý nếu có.
* Tầng mạng (Network Layer): Tầng mạng cung cấp các chức năng và quy trình cho việc truyền các chuỗi dữ liệu có độ dài đa dạng, từ một nguồn tới một đích, thông qua một hoặc nhiều mạng, trong khi vẫn duy trì chất lượng dịch vụ (quality of service) mà tầng giao vận yêu cầu.
* Tầng giao vận (Transport Layer): Tầng giao vận cung cấp dịch vụ chuyên dụng chuyển dữ liệu giữa các người dùng tại đầu cuối, nhờ đó các tầng trên không phải quan tâm đến việc cung cấp dịch vụ truyền dữ liệu đáng tin cậy và hiệu quả.
* Tầng phiên (Session layer): Tầng phiên kiểm soát các (phiên) hội thoại giữa các máy tính. Tầng này thiết lập, quản lý và kết thúc các kết nối giữa trình ứng dụng địa phương và trình ứng dụng ở xa.
* Tầng trình diễn (Presentation layer): Lớp trình diễn hoạt động như tầng dữ liệu trên mạng. Lớp này trên máy tính truyền dữ liệu làm nhiệm vụ dịch dữ liệu được gửi từ tầng Application sang dạng Fomat chung. Và tại máy tính nhận, lớp này lại chuyển từ Fomat chung sang định dạng của tầng Application.
* Tầng ứng dụng (Application layer): Tầng ứng dụng là tầng gần với người sử dụng nhất. Nó cung cấp phương tiện cho người dùng truy nhập các thông tin và dữ liệu trên mạng thông qua chương trình ứng dụng. Tầng này là giao diện chính để người dùng tương tác với chương trình ứng

## Giao thức truyền thông

* Giao thức truyền thông (Communication protocol) hay Giao thức liên mạng, Giao thức truyền thông, Giao thức trao đổi thông tin - trong công nghệ thông tin gọi tắt là giao thức (protocol) - là một tập hợp các quy tắc chuẩn dành cho việc biểu diễn dữ liệu, phát tín hiệu, chứng thực và phát hiện các lỗi dữ liệu. Đó là những việc rất cần thiết để gửi thông tin qua các kênh truyền thông, nhờ đó mà các thiết bị có thể kết nối và trao đổi thông tin với nhau.
* Các giao thức dùng cho việc truyền thông tin tín hiệu số trong mạng máy tính có nhiều tính năng để đảm bảo việc trao đổi dữ liệu một các đáng tin cậy tqua một kênh truyền thông không hoàn hảo.
* Có các mô hình giao thức mang tính lý thuyết, được ngành công nghệ thông tin tôn trọng một cách tương đối như mô hình OSI. Cũng có những giao thức được triển khai và ứng dụng rộng rãi như TCP, IP, HTTP, FTP, SMTP, POP3...
* Có nhiều giao thức được sử dụng để giao tiếp hoặc truyền đạt thông tin trên Internet, dưới đây là một số các giao thức tiêu biểu:
* TCP (Transmission Control Protocol): thiết lập kết nối giữa các máy tính để truyền dữ liệu. Nó chia nhỏ dữ liệu ra thành những gói (packet) và đảm bảo việc truyền dữ liệu thành công.
* IP (Internet Protocol): định tuyến (route) các gói dữ liệu khi chúng được truyền qua Internet, đảm bảo dữ liệu sẽ đến đúng nơi cần nhận.
* HTTP (HyperText Transfer Protocol): cho phép trao đổi thông tin (chủ yếu ở dạng siêu văn bản) qua Internet.
* FTP (File Transfer Protocol): cho phép trao đổi tập tin qua Internet.
* SMTP (Simple Mail Transfer Protocol): cho phép gởi các thông điệp thư điện tử (e-mail) qua Internet.
* POP3 (Post Office Protocol, phiên bản 3): cho phép nhận các thông điệp thư điện tử qua Internet.
* WAP (Wireless Application Protocol): cho phép trao đổi thông tin giữa các thiết bị không dây, như điện thoại di động.

# WEBRTC QUA WEBSOCKET TRONG NODEJS

## Tổng quan về WebRTC

### *Giới thiệu*

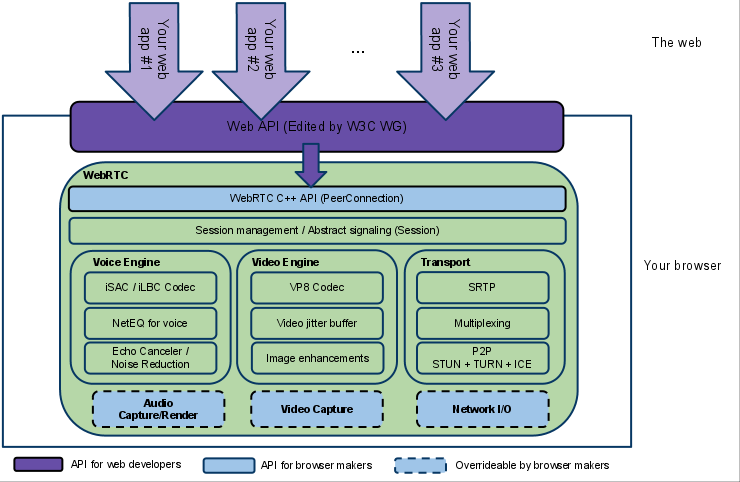
WebRTC, viết tắt của Web Real-Time Communication, là một giao thức cung cấp một bộ quy tắc cho giao tiếp ngang hàng theo thời gian thực và an toàn hai chiều cho web.

Với WebRTC, các ứng dụng web hoặc các tác nhân WebRTC khác có thể gửi video, âm thanh và các loại phương tiện khác giữa các đồng nghiệp.

WebRTC dựa vào một loạt các giao thức khác để đạt được mục đích của nó là tạo ra một kết nối hoặc kênh giao tiếp, sau đó truyền / trao đổi dữ liệu và / hoặc các loại phương tiện. Để điều phối giao tiếp, các máy khách WebRTC cần một số loại " máy chủ báo hiệu " ở giữa, điều này cần thiết để trao đổi thông tin siêu dữ liệu.

Socket.IO - và các máy chủ Node.js dựa trên ws cung cấp một giải pháp thay thế để cung cấp tín hiệu theo cách lâu dài, thời gian thực cho các máy khách WebRTC để chia sẻ mô tả phiên và thông tin phương tiện và thực sự trao đổi dữ liệu. Do đó, cả hai đều có thể được sử dụng làm công nghệ bổ sung.

### *Kiến trúc WebRTC*



*Hình 2.1 Mô hình kiến trúc WebRTC*

Có 2 lớp riêng biệt (distinct layers):

* Browser deveopers sẽ quan tâm đến WebRTC C++ API và các thành phần core sâu hơn của nó như Voice Engine, Video Engine, Transform. Hay dễ hiểu hơn đó là âm thanh, video và kết nối mạng.
* Web App developers sẽ quan tâm tới Web API.
* Your Web App: Một ứng dụng dựa trên web của nhà phát triển bên thứ ba với khả năng trò chuyện video và âm thanh được cung cấp bởi API web để liên lạc trong thời gian thực.
* Web API: Một API sẽ được các nhà phát triển bên thứ ba sử dụng để phát triển các ứng dụng giống như trò chuyện video dựa trên web.
* WebRTC Native C++ API: Lớp API cho phép các nhà sản xuất trình duyệt dễ dàng triển khai đề xuất API Web.
* Transport/Session: Các thành phần phiên được xây dựng bằng cách sử dụng lại các thành phần từ libjingle, không sử dụng hoặc yêu cầu giao thức xmpp / jingle.
* RTP Stack: Ngăn xếp mạng cho RTP, Giao thức thời gian thực.
* STUN/ICE: Một thành phần cho phép các cuộc gọi sử dụng cơ chế STUN và ICE để thiết lập kết nối qua nhiều loại mạng khác nhau.
* Session Management: Một lớp phiên trừu tượng, cho phép thiết lập và quản lý cuộc gọi. Điều này để lại quyết định triển khai giao thức cho nhà phát triển ứng dụng
* VoiceEngine: VoiceEngine là một khuôn khổ cho chuỗi phương tiện âm thanh, từ card âm thanh đến mạng.

#### iSAC / iLBC / Opus

#### Acoustic Echo Canceler (AEC)

#### Noise Reduction (NR)

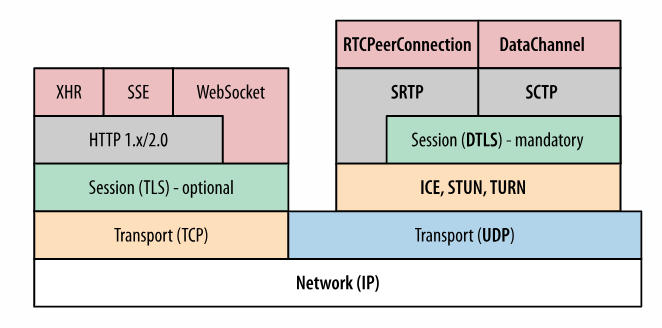
* Video Engine: VideoEngine là một chuỗi phương tiện video khung cho video, từ máy ảnh đến mạng và từ mạng đến màn hình

#### VP8

#### Video Jitter Buffer

#### Image enhancements

### *Một số giao thức trong WebRTC*



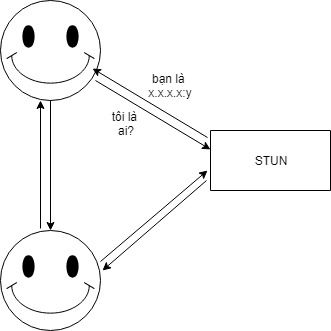
*Hình 2.2 Ngăn xếp giao thức trong WebRTC*

**SRTP:**  (Secure Real Time Protocol) 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 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**](https://tools.ietf.org/html/rfc4960) (Stream Control Tranmission Protocol) để truyền 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. SCTP được lựa chọn do có những tính năng tốt 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.

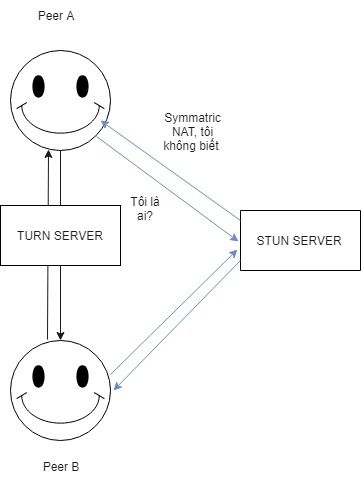
**DTLS:** Datagram Transport Layer Security- [DTLS](https://tools.ietf.org/id/draft-ietf-tls-dtls13-01.html) giao thức này cung cấp tính năng bảo mật và toàn vẹn dữ liệu. Tất cả các dữ liệu truyền P2P đều được bảo mật sử dụng DTLS.

**STUN:** NATcung cấp một địa chỉ IP để sử dụng trong mạng nội bộ. Nhưng địa chỉ này không được sử dụng bên ngoài mạng. Không biết địa chỉ công khai sẽ không có cách nào để hai Peer có thể tương tác. Để giải quyết vấn đề đó WebRTC sử dụng STUN (Session Traversal Utilities for NAT) [8]. STUN server tồn tại trên mạng internet và chỉ có nhiệm vụ duy nhất là kiểm tra địa chỉ IP và cổng của yêu cầu vừa đến và gửi trở lại IP và cổng đó. Các ứng dụng sử dụng STUN server để cung cấp IP và cổng công khai từ internet. Từ đó một WebRTC peer có thể tự lấy được địa chỉ IP và cổng công khai và đưa nó cho các peer khác thông qua cơ chế signaling. Hình bên dưới mô tả về cách làm việc của STUN server:

**

*Hình 2.3 Cách thức hoạt động của STUN*

**TURN:** Traversal Using Relays around NAT ([TURN](https://tools.ietf.org/html/rfc5389)) . Được xây dựng nhằm vượt qua symmetric NAT bằng cách mở một kết nối tới TURN server và đáp lại tất cả thông tin thông qua server này (dữ liệu audio/video/data streaming giữa các peer, không phải signaling data).Turn server làm được điều này vì nó có một public địa chỉ, vì thế nó có thể liên lạc với các peer thậm chí peer có tường lửa hoặc proxy đứng sau. Hình dưới mô tả hoạt động của TURN server.



*Hình 2.4 Cách thức hoạt động của TURN*

**SDP:** [Session Description Protocol](https://tools.ietf.org/html/rfc4566) là một chuẩn mô tả các thông số của mỗi kết nối như là độ phân giải, định dạng, codecs, mã hóa… Điều này làm cho mỗi peer có thể hiểu nhau khi dữ liệu được truyền. Đây thực chất là meta-data miêu tả nội dung chứ không phải là dữ liệu media.

**ICE:** [Interactive Connectivity Establishment](https://tools.ietf.org/html/rfc5245) là một chuẩn được sử dụng để thiết lập kết nối giữa các các peer trên internet.Mặc dù WebRTC là kết nối trực tiếp Peer-to-Peer, nhưng thực tế nó vẫn gặp phải vấn đề NAT (Network Address Translation) gây khó khăn khi kết nối.ICE sẽ cố gắng thiết lập kết nối bằng cách tìm đường đi tốt nhất cho kết nối.Bằng cách thử hết các khả năng song song nhau, ICE có thể chọn ra được lựa chọn hiệu quả nhất cho kết nối. Đầu tiên ICE cố gắng kết nối sử dụng địa chỉ host lấy được từ hệ điều hành hoặc card mạng. Nếu thất bại nó sẽ lấy địa chỉ IP public thông qua STUN server. Nếu vẫn thất bại, lưu lượng được gửi thông qua TURN server. Các cách để kết nối này được gọi là “candidate”, cách thức trao đổi sẽ được mô tả ở các phần bên sau.

### *Các phần chính của WebRTC*

**MediaStream**: đây là một stream dữ liệu âm thanh và hình ảnh, bằng cách gọi hàm getUserMedia để khởi tạo khi làm việc cục bộ. getUserMedia sẽ cho phép trình duyệt web truy cập vào camera và/hoặc microphone để lâý dữ liệu âm thanh, hình ảnh cho việc truyền tải.

Sau khi một kết nối WebRTC được thiết lập với một máy tính, MediaStream sẽ cho phép truy cập vào Stream của một máy tính khác.

**RTCDataChanel:** cho phép trình duyệt chia sẻ dữ liêu Peer-2-Peer. Đây là một kênh hai chiều chịu trách nhiệm trao dư liệu thời gian thực. RTCDataChanel sẽ trao đổi các dữ liệu dạng text, tệp P2P và các loại khác à không chứa các dữ liệu nghe nhìn (âm thanh, hình ảnh). RTCDataChanel trao đổi thông tin nhanh chóng, an toàn, đáng tin.

**RTCPeerConnection:** dùng để cài đặt VideoCall/Voice dùng cho việc truyền tải. Đây là phần quan trọng giúp kết nối MediaStream và RTCDataChanel trở thành WebRTC. RTCPeerConnection là API giúp kết nối giữa hai trình duyệt, cung cấp các phương thức để kết nối, duy trì kết nối, và đóng kết nối khi không còn nhu cầu sử dụng.

### *Cách thức hoạt động của giao thức WebRTC*

Tác nhân WebRTC biết cách tạo kết nối với các đồng nghiệp. Báo hiệu kích hoạt nỗ lực ban đầu này, cuối cùng làm cho cuộc gọi giữa cả hai tác nhân có thể thực hiện được. Đại lý sử dụng mô hình đề nghị / trả lời: một đề nghị được đưa ra bởi một đại lý để bắt đầu cuộc gọi và một người khác trả lời cuộc gọi để kiểm tra tính tương thích với mô tả phương tiện được cung cấp.

Ở cấp độ cao, giao thức WebRTC hoạt động trong khoảng bốn bước. Đối với các giai đoạn này, giao tiếp diễn ra theo một thứ tự phụ thuộc, trong đó một giai đoạn phải được hoàn thành trước khi giai đoạn tiếp theo có thể bắt đầu. Bốn giai đoạn này bao gồm:

**Báo hiệu**: Điều này bắt đầu quá trình xác định hai tác nhân WebRTC có ý định giao tiếp và trao đổi dữ liệu. Khi các đồng nghiệp cuối cùng kết nối và có thể giao tiếp, việc truyền tín hiệu sẽ sử dụng một giao thức khác, SDP . Giao thức mô tả phiên (một giao thức bản rõ) rất hữu ích để trao đổi các phần phương tiện trong các cặp khóa-giá trị. Với nó, chúng tôi có thể chia sẻ trạng thái giữa hai hoặc nhiều đồng nghiệp có ý định kết nối. Lưu ý : Trạng thái chia sẻ có thể cung cấp tất cả các tham số cần thiết để thiết lập kết nối giữa các đồng nghiệp.

**Kết nối:** Sau khi báo hiệu, các tác nhân WebRTC cần đạt được giao tiếp hai chiều, ngang hàng. Mặc dù việc thiết lập kết nối có thể khó khăn do một số lý do như phiên bản IP, vị trí mạng hoặc giao thức, kết nối WebRTC cung cấp các tùy chọn tốt hơn khi so sánh với các máy khách web / máy chủ truyền thống. Chúng bao gồm giảm băng thông, độ trễ thấp hơn và bảo mật tốt hơn. Lưu ý : WebRTC cũng sử dụng ICE (Thiết lập Kết nối Tương tác) để kết nối hai tác nhân. ICE là một giao thức cố gắng tìm ra cách tốt nhất để liên lạc giữa hai đại lý ICE. Thông tin chi tiết có thể được tìm thấy tại đây .

**Bảo mật**: Mọi kết nối WebRTC đều được mã hóa và xác thực. Nó sử dụng các giao thức DTLS và SRTP để cho phép giao tiếp liền mạch và an toàn trên lớp dữ liệu. DTLS, tương tự như TLS, cho phép chúng tôi thương lượng một phiên và sau đó trao đổi dữ liệu một cách an toàn giữa hai đồng nghiệp. Mặt khác, SRTP được thiết kế để trao đổi phương tiện.

**Giao tiếp**: WebRTC cho phép chúng tôi gửi và nhận vô số luồng âm thanh và video. Giao thức độc lập với một codec cụ thể, vì có các tùy chọn . Nó dựa trên hai giao thức đã có từ trước: RTP và RTCP . RTP là giao thức mang phương tiện truyền thông. Nó được thiết kế để cho phép phân phối video theo thời gian thực. RTCP là giao thức truyền siêu dữ liệu về cuộc gọi.

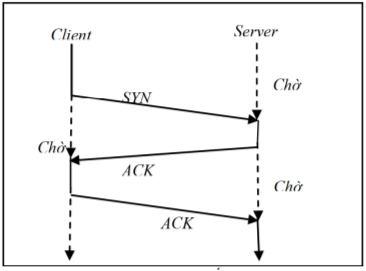
## Giao thức TCP

### *Khái niệm*

Giao thức TCP là giao thức truyền thông có kết nối (connection oriented) và tin cậy (reliable).

Truyền thông có kết nối có nghĩa là trước khi truyền dữ liệu, giao thức TCP gửi và giao thức TCP nhận phải thiết lập kênh truyền trước khi gửi/nhận dữ liệu và, trong suốt quá trình trao đổi đữ liệu hai giao thức gửi/nhận phải duy trì kênh truyền đã được thiết lập. Truyền thông tin cậy có nghĩa là nút gửi biết được gói tin đã được gửi đến đích hay không. Nếu không đến được, nút gửi phải gửi lại gói tin. Nếu gói tin đã được gửi thành công, nút gửi gửi gói tin tiếp theo. Cụ thể của quá trình này được thực hiện bằng cách mỗi gói tin được đánh một số thứ tự. Mỗi khi nhận được gói tin, nút nhận phải gửi một phản hồi xác nhận (acknowledgment) cho nút gửi để thông báo đã nhận được thành công gói tin.

### *Thiết lập kênh truyền*

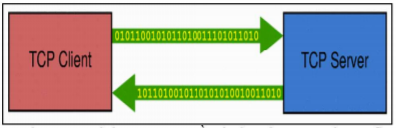
Quá trình thiết lập kênh truyền được thực hiện cụ thể bởi một quá trình gọi là quá trình bắt tay ba bước (three-way handshake) như sau:

*Hình 2.5 Quá trình bắt tay 3 bước*

1. Giao thức TCP phía máy gửi phải yêu cầu thiết lập một kênh truyền bằng cách gửi một phân đoạn (segment) gọi là SYN đến giao thức TCP phía máy nhận.
2. Giao thức TCP máy nhận trả lại một phân đoạn (segment) xác nhận đã nhận được thành công.
3. Giao thức TCP phía máy gửi tiếp tục gửi một phân đoạn ACK để xác nhận và sau đó là quá trình gửi dữ liệu.

### *Truyền dữ liệu bằng giao thức TCP*

Giao thức TCP sử dụng truyền thông dạng luồng dữ liệu hay dãy các bytes (byte-stream) liên tục trên kênh truyền, được mô tả như hình vẽ bên dưới:



*Hình 2.6 Minh họa việc truyền dữ liệu bởi giao thức TCP*

Mỗi gói tin TCP (hay còn gọi là segment) gồm một số thông tin điều khiển ở phần đầu được mô tả ở bảng dưới đây:

Socket được xem như điểm giao tiếp của ứng dụng với mạng, do đó chương trình client/server sẽ thực hiện hai thao tác cơ bản sau:

* Để truyền dữ liệu, chương trình chỉ cần gửi dữ liệu vào socket
* Để nhận dữ liệu, chương trình chỉ cần đọc dữ liệu từ socket

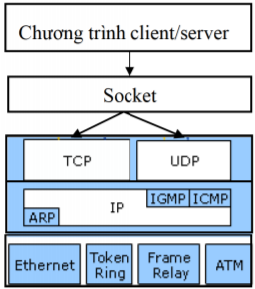
Có hai loại socket: socket nối với giao thức TCP còn được gọi là TCP socket, và tương tự như vậy socket nối với giao thức UDP gọi là UDP socket

Bảng 2.1 Thông tin điều khiển của một Segment

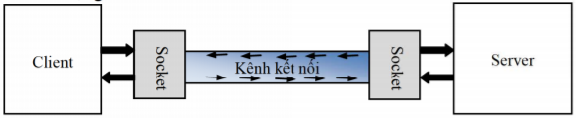
|  |  |
| --- | --- |
| Thông tin điều khiển | Mô tả |
| Số cổng nguồn(Source Port) | Số hiệu cổng TCP của tiến trình gửi |
| Số cổng đích(Destination Port) | Số hiệu cổng TCP của tiến trình nhận |
| Số hiệu gói tin(Sequence Number) | Số thứ tự byte đầu tiên của gói tin TCP |
| Số hiệu xác nhận(Acknowledgment Number) | Số thứ tự của byte đầu tiên của gói tin TCP mà giao thức TCP gửi chờ nhận từ phía nhận |
| Kích thước bộ đệm(Window) | Kích thước hiện tại của bộ đệm TCP phía máy gửi dùng để chứa gói tin TCP nhận được. |
| Thông tin kiểm tra lỗi(TCP Checksum) | Dùng để kiểm tra tính toàn vẹn của phần đầu và dữ liệu của gói tin TCP. |

### *Khái niệm về Socket*

Để xây dựng một chương trình theo mô hình client-server, cho phép truyền nhận dữ liệu qua mạng, người lập trình chỉ cần viết chương trình ở tầng ứng dụng và tìm cách giao tiếp với một trong hai giao thức (dịch vụ) ở tầng Vận chuyền là TCP hoặc UDP. Để làm được việc này, các hệ điều hành cung cấp một phương tiện cho phép ứng dụng người dùng giao tiếp được với các giao thức TCP và UDP một cách dễ dàng. Phương tiện đó được gọi là Socket, được biêu diễn như hình vẽ bên dưới:



*Hình 2.7 Mô tả Socket trong mô hình TCP/IP*



*Hình 2.8 Socket trong mô hình ứng dụng client-server*

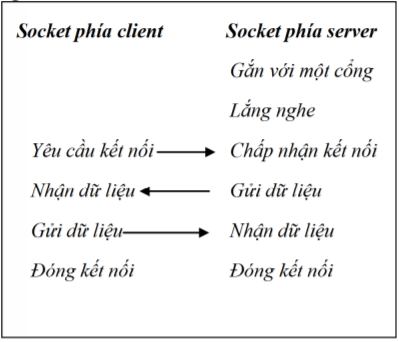
Có thể nói một kênh kết nối giữa hai chương trình được thiết lập nhờ vào hai socket được gắn ở hai chương trình. Như hình vẽ trên, môi chương trình client và server đêu có một socket được gắn vào nó để có thể truyền và nhận dữ liệu.

### *Sử dụng TCP socket*

Như trình bày ở mục trên, việc xây dựng chương trình client-server sử dụng giao thức TCP của tầng Vận chuyên được thực hiện thông qua TCP socket. Do vậy, người lập trình chỉ cần viết mã lệnh đề đọc và ghi dữ liệu giữa chương trình và TCP socket, sau đó TCP socket sẽ có nhiệm vụ liên lạc và gửi, nhận dữ liệu với TCP socket phía đầu còn lại.

Do đó, các hoạt động của TCP socket sẽ bao gồm như sau:

TCP Socket phía server:

* Lắng nghe yêu cầu nối kết từ socket phía client
* Chấp nhận kết nối từ socket phía client
* Gửi dữ liệu nhận dữ liệu
* Đóng kết nối
* TCP Socket phía client:
* Yêu cầu kết nối với socket phía server
* Gửi dữ liệu

*Hình 2.9 Các hoạt động của TCP socket*

* Nhận dữ liệu
* Đóng kết nối

Để TCP socket phía client có thể liên lạc được với TCP socket phía server thì TCP socket phía server phải được gắn với một công (port). Sau khi TCP socket phía server được gắn một công, nó sẽ chờ và lắng nghe yêu cầu tạo kênh kết nối từ TCP socket phía client gửi đến.

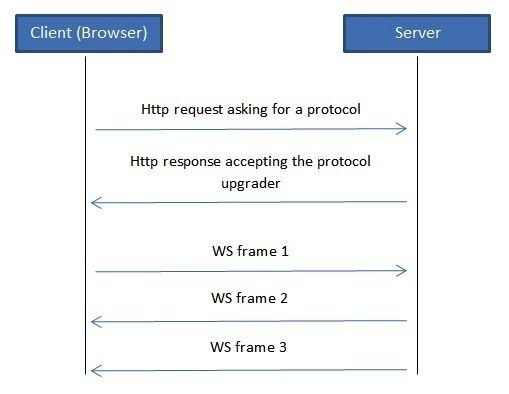
## WebSocket là gì?

### *Giới thiệu*

WebSoket là công nghệ hỗ trợ giao tiếp hai chiều giữa client và server bằng cách sử dụng một TCP socket để tạo một kết nối hiệu quả và ít tốn kém. Mặc dù được thiết kế để chuyên sử dụng cho các ứng dụng web, lập trình viên vẫn có thể đưa chúng vào bất kì loại ứng dụng nào.

* WebSockets mới xuất hiện trong HTML5, là một kỹ thuật Reverse Ajax. WebSockets cho phép các kênh giao tiếp song song hai chiều và hiện đã được hỗ trợ trong nhiều trình duyệt (Firefox, Google Chrome và Safari). Kết nối được mở thông qua một HTTP request (yêu cầu HTTP), được gọi là liên kết WebSockets với những header đặc biệt. Kết nối được duy trì để bạn có thể viết và nhận dữ liệu bằng JavaScript như khi bạn đang sử dụng một TCP socket đơn thuần.
* Không giống với giao thức HTTP là cần client chủ động gửi yêu cầu cho server, client sẽ chời đợi để nhận được dữ liệu từ máy chủ. Hay nói cách khác với giao thức Websocket thì server có thể chủ động gửi thông tin đến client mà không cần phải có yêu cầu từ client
* Tất cả dữ liệu giao tiếp giữa client-server sẽ được gửi trực tiếp qua một kết nối cố định làm cho thông tin được gửi đi nhanh chóng và liên tục khi cần thiết. WebSocket làm giảm độ trễ bởi vì một khi kết nối WebSocket được thành lập, server không cần phải chờ đợi cho một yêu cầu từ client.
* Tương tự như vậy, client có thể gửi tin nhắn đến server bất cứ lúc nào. Yêu cầu duy nhất này giúp làm giảm đáng kể độ trễ, mà sẽ gửi một yêu cầu trong khoảng thời gian, cho dù thông điệp có sẵn.
* Để có thể sử dụng được Websocket thì không phải chỉ cần trình duyệt hỗ trợ mà còn phải có server Websocket, server Websocket có thể được tạo ra bằng bất kỳ ngôn ngữ server-side nào, nhưng Node.js được sử dụng rộng rãi hơn cả vì nó viết bằng Javascript nên mang nhiều ưu điểm so với các ngôn ngữ server-side truyền thống khác.
* Dữ liệu truyền tải thông qua giao thức HTTP (thường dùng với kĩ thuật Ajax) chứa nhiều dữ liệu không cần thiết trong phần header. Một header request/response của HTTP có kích thước khoảng 871 byte, trong khi với WebSocket, kích thước này chỉ là 2 byte (sau khi đã kết nối). Vậy giả sử bạn làm một ứng dụng game có thể tới 10,000 người chơi đăng nhập cùng lúc, và mỗi giây họ sẽ gửi/nhận dữ liệu từ server.

### *Hoạt động của WebSocket*



*Hình 2.10 Mô hình hoạt động của WebSocket*

Giao thức có hai phần: Bắt tay và truyền dữ liệu Ban đầu client sẽ gửi yêu cầu khởi tạo kết nối websocket đến server, server kiểm tra và gửi trả kết quả chấp nhận kết nối, sau đó kết nối được tạo và quá trình gửi dữ liệu có thể được thực hiện, dữ liệu chính là các Ws frame

### *Thư viện Socket.IO*

Socket.IO là một bộ thư viện dành cho các ứng dụng web, mobile đê phát triển các ứng dụng realtime. Với đặc trưng mạnh mẽ và dễ sử dụng, [Socket.IO](http://socket.io/) đang ngày càng được sử dụng rộng rãi từ những trang mạng xã hội cần sự tương tác cao, đến các blog hay các trang web thương mại điện tử. Với bộ thư viện này, làm việc với WebSockets trở nên đơn giản hơn rất nhiều. Thư viện gồm 2 phần:

* Phía client: gồm bộ thư viện viết cho web(JavaScript), iOS, Android
* Phía server: viết bằng JavaScript và dùng cho các máy chủ node.JS

Socket.IO hỗ trợ sử dụng rất nhiều các công nghệ realtime:

* WebSocket
* Flash Socket
* AJAX long-polling
* AJAX multipart streaming
* IFrame
* JSONP polling

Nó sẽ tự động chuyển sang Websocket nếu có thể, hầu hết các trình duyệt hiện nay đã hỗ trợ websocket nên việc sử dụng [socket.io](http://socket.io/) trên trình duyệt cũng là đang sử dụng websocket Việc sử dụng [socket.io](http://socket.io/) rất đơn giản và giống nhau ở cả client lẫn server nó bao gồm 3 phần chính:

* Khởi tạo kết nối
* Lắng nghe event
* Gửi event

Riêng ở server thì sẽ không có phần khởi **tạo kết nối** vì chỉ có clent mới cần khởi tạo kết nối đến server Việc dùng [socket.io](http://socket.io/) đồng bộ ở cả client lẫn server, cú pháp cũng khá đơn giản. Chúng ta sẽ làm một ví dụ nhỏ để làm quen với một số cách gửi và nhận dữ liệu với [socket.io](http://socket.io/)

## Các bước và thuật toán xây dựng chương trình Client-Server

### *Các bước xây dựng chương trình server*

Chương trình server được xây dựng gồm các bước sau:

1. Tạo TCP socket
2. Gắn TCP vào số hiệu cổng
3. Chở và lắng nghe yêu cầu kết nối từ chương trình client
4. Nếu có yêu cầu kết nối, chấp nhận kết nối từ chương trình client
5. Sau khi kênh kết nối được thiết lập, trao đổi dữ liệu với chương trình client
6. Đóng kết nối

*Hình 2.11 Thuật toán xây dựng chương trình Server*

### *Các bước xây dựng chương trình client*

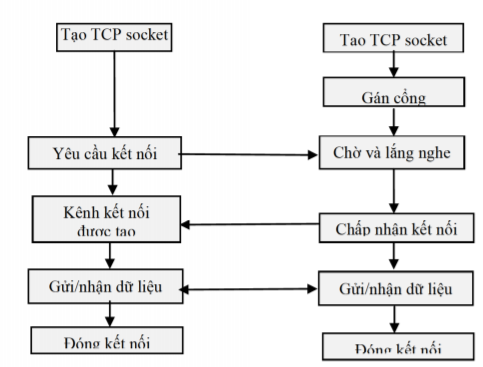
Chương trình client được xây dựng gồm các bước sau:

1. Tạo TCP socket
2. Yêu cầu kết nối đến server. Nếu được server chấp nhận, một kênh kết nối được thiết lập
3. Trao đổi dữ liệu với chương trình server
4. Đóng kết nối

*Hình 2.12 Thuật toán xây dựng chương trình Client*

### *Thuật toán xây dựng chương trình client-server tổng quát*

Từ các bước xây dựng chương trình client và server ở trên, quá trình xây dựng một ứng dụng theo mô hình client-server được mô tả băng thuật toán ở sơ đồ khôi như sau:



*Hình 2.13 Thuật toán xây dựng chương trình client-server tổng quát*

Sau khi chương trình server tạo TCP socket và gắn một số hiệu công vào TCP socket, chương trình server băt đâu chờ và lăng nghe yêu câu kêt nôi từ chương trình client.

Do TCP là giao thức truyền thông có kết nối nên để trao đổi được dữ liệu với chương trình server, chương trình client phải tạo một kênh kết nối đến chương trình server. Chương trình client đầu tiên tạo một TCP socket, sau đó gửi yêu cầu kết nối đến chương trình server đang chạy tại một địa chỉ IP và số hiệu cổng đã xác định.

Chương trình server nhận được yêu cầu kết nối gửi đến, nếu không có lỗi sẽ chấp nhận kết nối. Ngay khi chương trình server chấp nhận kết nối, một kênh kết nối được thiết lập để cho phép chương trình client và server trao đổi dữ liệu với nhau

# TRIỂN KHAI VÀ XÂY DỰNG

## Bài toán chat room

* Chat room là hình thức thảo luận trực tiếp trên Internet, bạn có thể thảo luận, gửi file, hình ảnh, trao đổi và nhận được câu trả lời hầu như ngay lập tức.
* Nó là công cụ để giúp mọi người có thể làm việc với nhau, trao đổi thông tin một cách dễ dàng cụ thể bằng giọng nói, hình ảnh của mình thông qua các cuộc gọi nhóm mà không cần phải gặp mặt.
* Trong đồ án này, em sẽ sử dụng NodeJs và WebRTC để lập trình chương trình chat room.

## Giải quyết bài toán

* Chat realtime: Xây dựng 1 server bằng Nodejs, mỗi user khi đăng nhập sẽ được thêm vào một room theo mã room. Server sẽ Node sẽ chịu trách nhiệm trung gian nhận và gửi tin nhắn cho tất cả user trong cùng một room.
* Video call: Mỗi khi user tham gia vào một room server sẽ gửi peerId của user đó cho tất cả user khác trong room. Các user sẽ tự connect với nhau (mỗi user sẽ connect với n-1 user khác). Ngoài ra ứng dụng có sử dụng TURN server chịu tải, nguyên lý hoạt động của TURN server là thay vì 1 user gửi gói tin cho n-1 user khác TURN server sẽ nhận gói tin đó và gửi cho n-1 user khác

## Tổng quan về NodeJs

### *NodeJs là gì?*

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

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

Khi nói đến NodeJS thì phải nghĩ tới vấn đề Realtime. Realtime ở đây chính là xử lý giao tiếp từ client tới máy chủ theo thời gian thực. Giống như khi bạn lướt Facebook thì mỗi khi bạn comment hay like một topic nào đó thì ngay lập tức chủ topic và những người đã comment trên đó sẽ nhận được thông báo là bạn đã comment.

### *Các đặc tính của NodeJs*

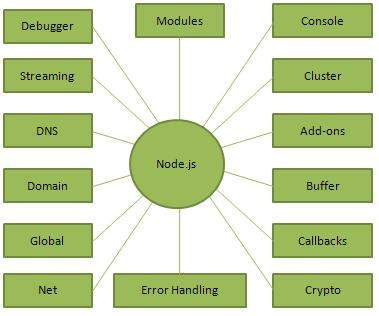
**Không đồng bộ**: Tất cả các API của NodeJS đều không đồng bộ (*none-blocking*), nó chủ yếu dựa trên nền của NodeJS Server và chờ đợi Server trả dữ liệu về. Việc di chuyển máy chủ đến các API tiếp theo sau khi gọi và cơ chế thông báo các sự kiện của Node.js giúp máy chủ để có được một phản ứng từ các cuộc gọi API trước (Realtime)

**Chạy rất nhanh**: NodeJ được xây dựng dựa vào nền tảng V8 Javascript Engine nên việc thực thi chương trình rất nhanh.

**Đơn luồng nhưng khả năng mở rộng cao**: Node.js sử dụng một mô hình luồng duy nhất với sự kiện lặp. cơ chế tổ chức sự kiện giúp các máy chủ để đáp ứng một cách không ngăn chặn và làm cho máy chủ cao khả năng mở rộng như trái ngược với các máy chủ truyền thống mà tạo đề hạn chế để xử lý yêu cầu. Node.js sử dụng một chương trình đơn luồng và các chương trình tương tự có thể cung cấp dịch vụ cho một số lượng lớn hơn nhiều so với yêu cầu máy chủ truyền thống như Apache HTTP Server.

**Không đệm**: NodeJS không đệm bất kì một dữ liệu nào và các ứng dụng này chủ yếu là đầu ra dữ liệu

**Có giấy phép**: NodeJS đã được cấp giấy phép bởi [MIT License](https://raw.githubusercontent.com/joyent/node/v0.12.0/LICENSE).



*Hình 3.1 Các thành phần quan trọng của NodeJs*

### *Hai NodeJS Framework sử dụng phổ biến*

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

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

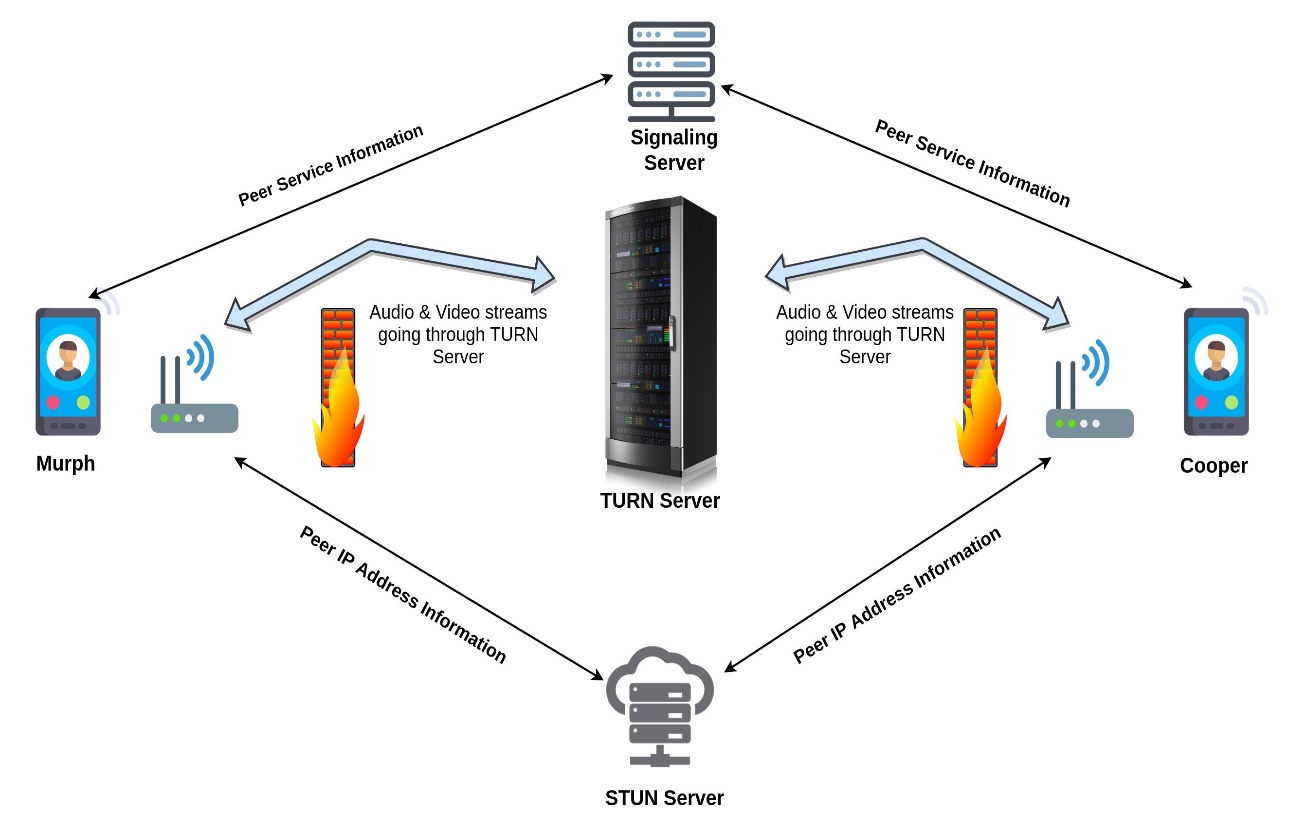
**SocketIO:**  là một web-socket framework có sẵn cho nhiều ngôn ngữ lập trình.

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

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

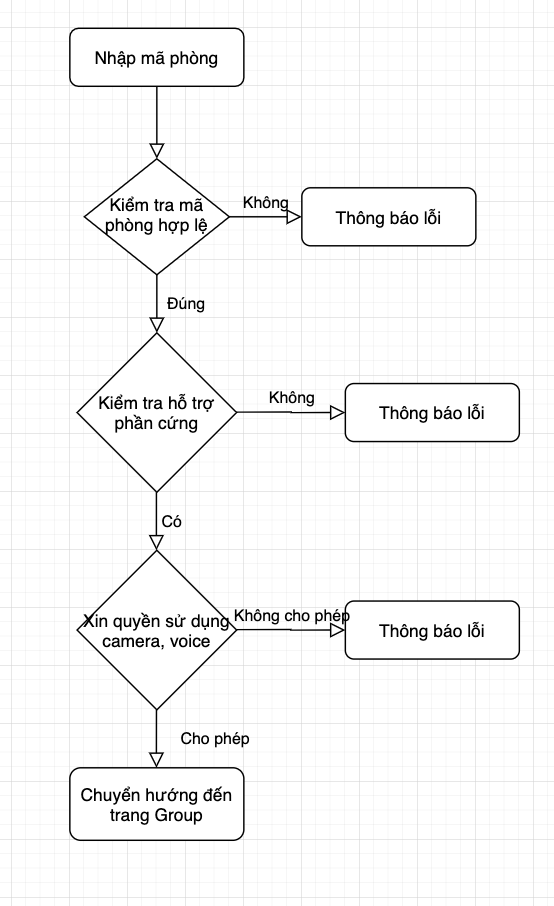
## Tổng quan hệ thống

### *Mô hình hoạt động*



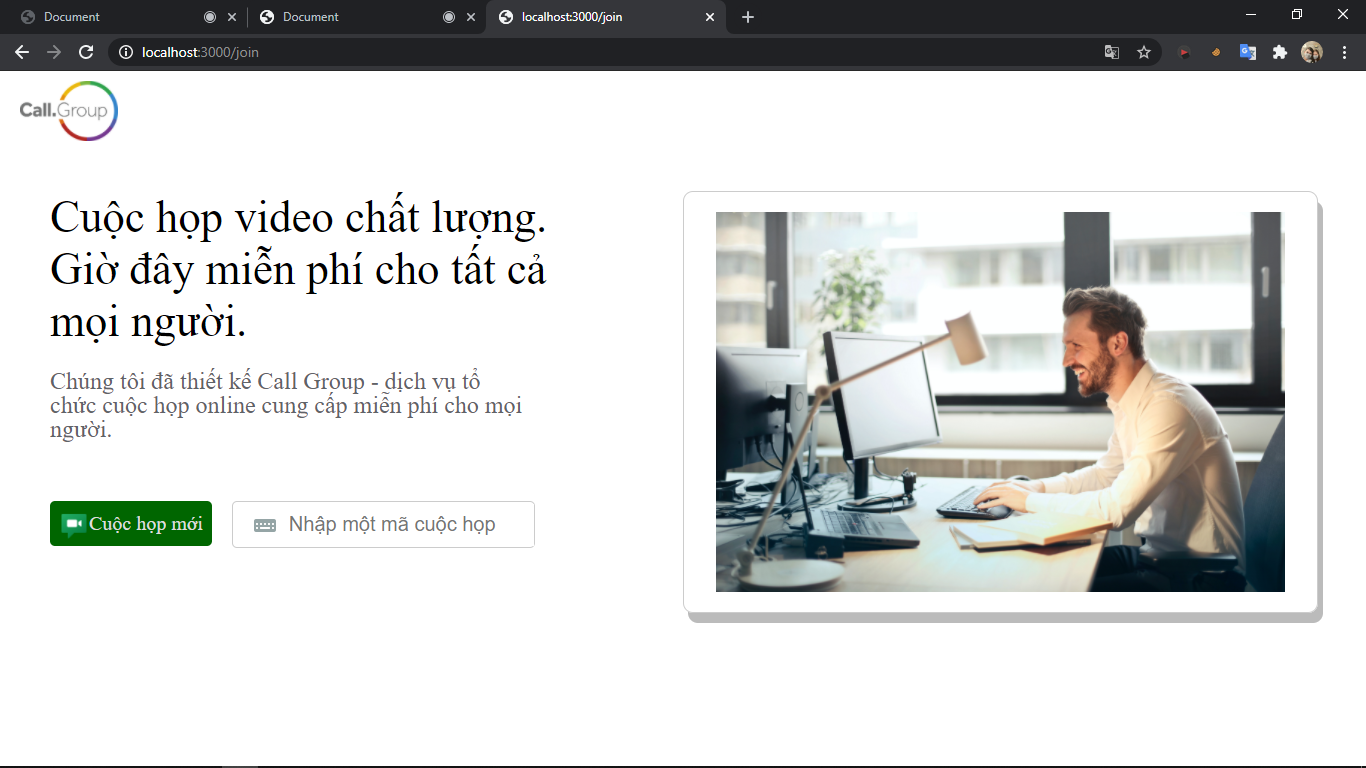
*Hình 3.2 Mô hình hoạt động*

### *Biểu đồ hoạt động*

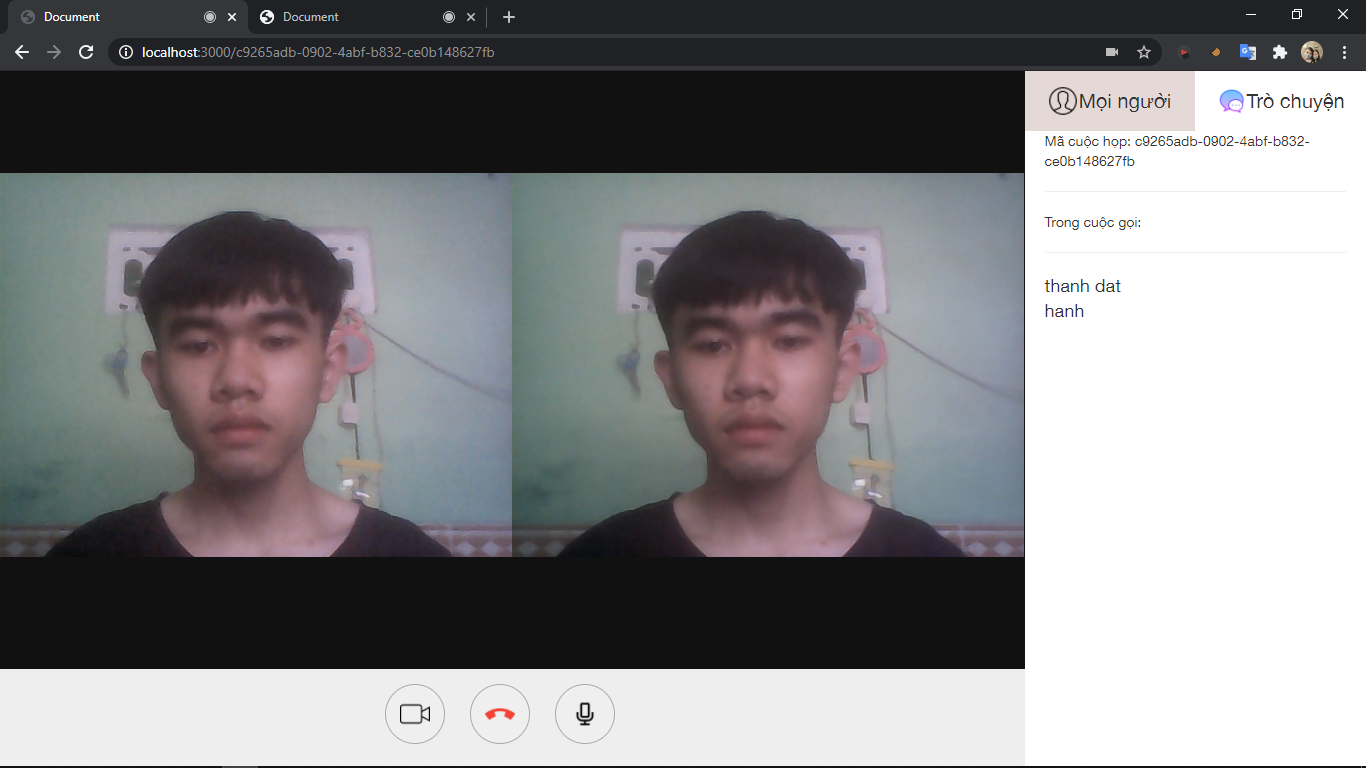


*Hình 3.3 Biểu đồ hoạt động tham gia phòng*

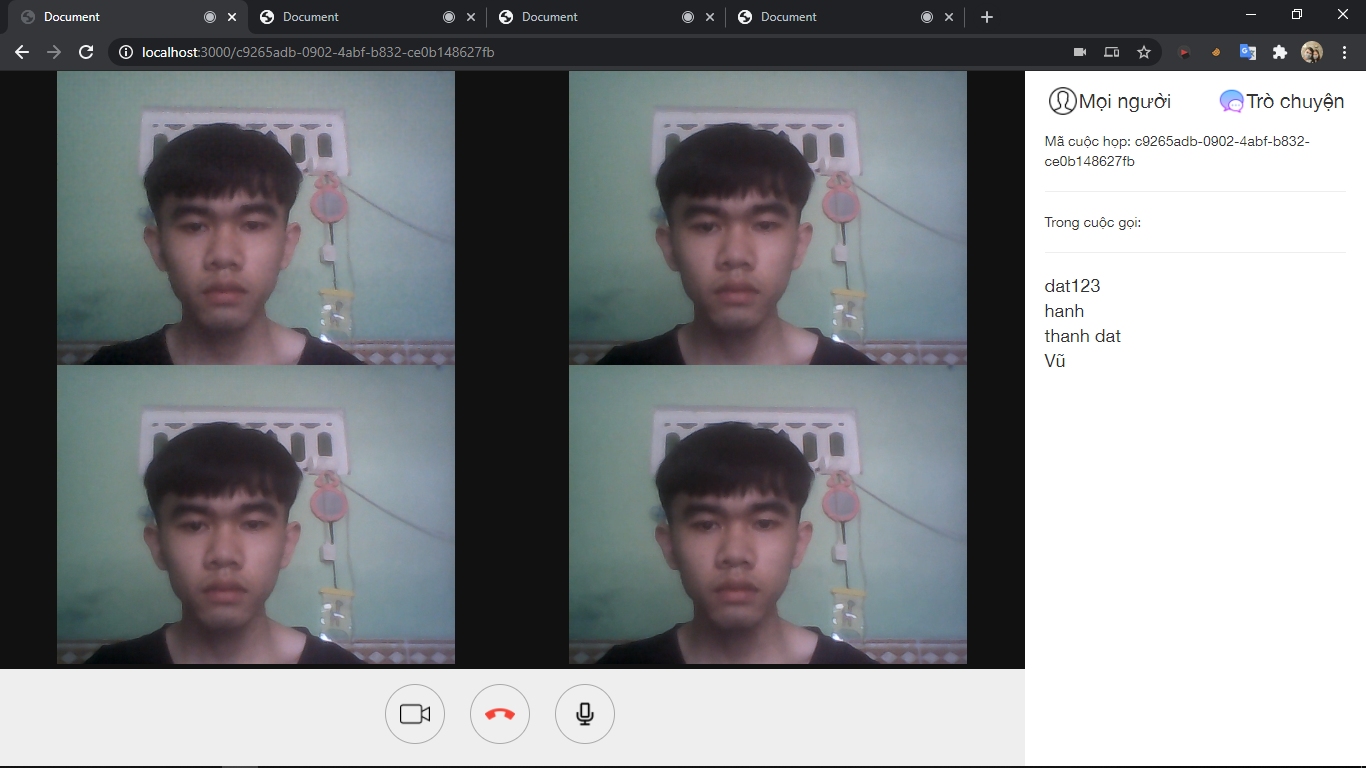
## Demo giao diện



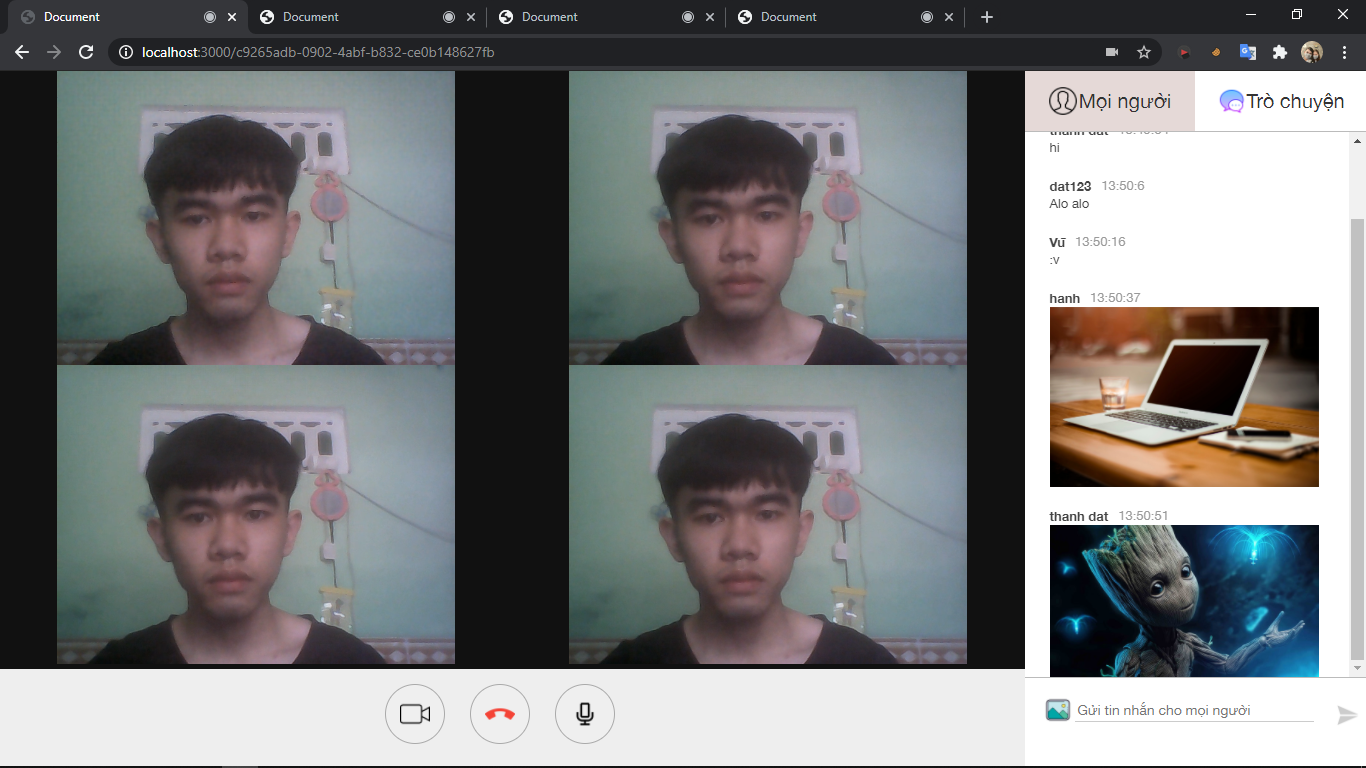
*Hình 3.4 Giao diện trang tạo/tham gia phòng*



*Hình 3.5 Giao diện phòng chat*



*Hình 3.6 Giao diện phòng chat*



*Hình 3.7 Giao diện phòng chat*

# KẾT LUẬN

## Kết quả

* Xây dựng được chương trình sử dụng được Multi Threading cho phép nhiều Client kết nối tới một Server
* Cho phép các Client gọi video và nhắn tin với nhau thông qua một room chat

## Hạn chế

* Ứng dụng chưa gửi được file, icon
* Giao diện đơn giản
* Số lượng người tham gia cùng lúc còn hạn chế

## Hướng phát triển

* Nâng cấp, hoàn thiện giao diện
* Nâng cấp băng thông, tăng giới hạn số lượng người tham gia cùng lúc
* Cho phép gọi với chất lượng video cao hơn

# DANH MỤC TÀI LIỆU THAM KHẢO

Tiếng Việt:

1. Xây dựng chương trình Client-Server sử dụng giao thức TCP – PGS.TS Huỳnh Công Pháp.
2. Nodejs trang <https://vi.wikipedia.org/wiki/Node.js>
3. WebRTC trang https://vi.wikipedia.org/wiki/WebRTC

Tiếng Anh:

1. Advanced Network Programing - Principles and Techiques của Bogdan Ciubotaru và Gabriel-Miro Muntean