HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÁO CÁO MÔN HỌC

CÁC HỆ THỐNG PHÂN TÁN

Đề tài: ỨNG DỤNG CHAT PEER TO PEER SỬ DỤNG WEBRTC

Giảng viên:	TS. Kim Ngọc Bách	
Lớp:	M24CQHT02-B	
Thực hiện:	Nhóm 12	
	Đào Văn Luân	B24CHHT084
	Trần Đức Tiến Long	B24CHHT082
	Trần Xuân Huỳnh	B24CHHT079

Hà Nội - 2025

MỤC LỤC

I. GIỚI THIỆU BÀI TOÁN CHAT P2P 1
1.1. Khái niệm P2P và vai trò trong Chat 1
1.2. Mục tiêu phát triển 1
II. CÁC KIẾN TRÚC CHAT HIỆN CÓ 2
2.1. Kiến trúc Client–Server truyền thống 2
2.2. Kiến trúc P2P đơn giản (không có signaling) 2
2.3. Kiến trúc P2P với Signaling Server 3
III. CƠ SỞ LÝ THUYẾT 4
3.1. NAT – Network Address Translation 4
3.2. ICE – Interactive Connectivity Establishment 4
3.3. STUN – Session Traversal Utilities for NAT 5
3.3.1 Bản chất & vai trò 6
3.3.2 Khi nào STUN hiệu quả / hạn chế 7
3.4. TURN – Traversal Using Relays Around NAT 8
3.4.1. Khi nào phải dùng TURN? 9
3.4.2. TURN hoạt động thế nào (tóm tắt theo chuẩn) 9
3.4.3. Tác động hệ thống & khuyến nghị triển khai 10
3.5. Quan hệ giữa ICE, STUN, TURN 12
3.6. Luồng kết nối tổng thể 12
3.6.1. Các thành phần 13
3.6.2. Luồng xử lý minh họa trong hình 13

IV. CÔNG NGHỆ SỬ DỤNG 15
4.1. Android Framework 15
4.2. WebRTC (Web Real-Time Communication) 15
4.3. Signaling Server 15
4.4. Firebase 16
4.4.1. Firebase Authentication 16
4.4.2. Cloud Firestore 16
V. ỨNG DỤNG 17
VI. KẾT LUẬN 20
Tài liệu tham khảo 21

I. GIỚI THIỆU BÀI TOÁN CHAT P2P

1.1. Khái niệm P2P và vai trò trong Chat

- P2P (Peer-to-Peer) là mô hình kết nối mà trong đó các thiết bị (peers) đóng vai trò như cả client lẫn server, giao tiếp trực tiếp với nhau mà không qua trung gian. Mô hình này phổ biến trong nhiều ứng dụng chia sẻ file (như BitTorrent), gọi thoại/video (như Skype), và các dịch vụ truyền thông thời gian thực khác.
- Với chat P2P, dữ liệu (tin nhắn, media, v.v.) được truyền trực tiếp giữa hai người dùng, không cần thông qua máy chủ trung gian — giúp giảm độ trễ, tăng bảo mật, và tiết kiệm băng thông

1.2. Mục tiêu phát triển

- Mục tiêu chính: Phát triển một ứng dụng chat di động Android P2P, trong đó người dùng kết nối trực tiếp với nhau để trò chuyện, không cần server trung gian truyền media dài hạn
- Lợi ích chính:
    - Riêng tư hơn và hiệu năng thấp hơn so với kiến trúc client-server.
    - Tiết kiệm băng thông cho server và giảm độ trễ vì dữ liệu trao đổi trực tiếp giữa hai thiết bị.
- Thách thức kỹ thuật:
    - Peer đứng sau NAT hoặc firewall làm việc khó khăn với kết nối trực tiếp.
    - Cần cơ chế signaling để hai peer biết nhau và khởi tạo kết nối.
    - Phải giải quyết NAT traversal (qua ICE, STUN, TURN).

II. CÁC KIẾN TRÚC CHAT HIỆN CÓ

2.1. Kiến trúc Client–Server truyền thống

- Nguyên lý hoạt động: Mọi tin nhắn được gửi từ thiết bị người gửi đến máy chủ trung gian. Máy chủ sau đó chuyển tiếp tin nhắn đến người nhận. Server thường đóng vai trò lưu trữ, quản lý trạng thái và xử lý xác thực.

- Ưu điểm:
  + Dễ triển khai và bảo trì.
  + Dễ quản lý, giám sát và lưu trữ dữ liệu tập trung.
  + Hỗ trợ đồng bộ dữ liệu giữa nhiều thiết bị.
- Nhược điểm:
  + Phụ thuộc hoàn toàn vào máy chủ, nếu server lỗi, hệ thống ngừng hoạt động.
  + Tốn băng thông và tài nguyên máy chủ khi số lượng người dùng tăng lớn.
  + Vấn đề về bảo mật và quyền riêng tư khi dữ liệu đi qua và lưu trên server.

2.2. Kiến trúc P2P đơn giản (không có signaling)

- Nguyên lý hoạt động: Hai người dùng kết nối trực tiếp với nhau thông qua việc chia sẻ thủ công thông tin kết nối (SDP – Session Description Protocol). Không có máy chủ trung gian để hỗ trợ quá trình tìm kiếm và trao đổi thông tin.
- Ưu điểm:
  + Loại bỏ hoàn toàn chi phí vận hành server.
  + Độ trễ thấp do dữ liệu truyền trực tiếp.
- Nhược điểm:
  + Quá trình thiết lập kết nối phức tạp và không thân thiện với người dùng phổ thông.

+ Không thể hoạt động trong môi trường có NAT hoặc firewall phức tạp.

+ Khó mở rộng quy mô hoặc tích hợp thêm tính năng nâng cao.

2.3. Kiến trúc P2P với Signaling Server

- Nguyên lý hoạt động: Vẫn duy trì kết nối P2P trực tiếp cho luồng dữ liệu chat, nhưng bổ sung một signaling server để tự động hóa quá trình trao đổi thông tin khởi tạo kết nối (SDP, ICE candidates). Signaling server không truyền dữ liệu chat, chỉ giúp hai peer "gặp" nhau và thiết lập kết nối.
- Ưu điểm:

  + Tự động hóa quá trình bắt tay kết nối, cải thiện trải nghiệm người dùng.

  + Vẫn giữ được độ trễ thấp và giảm tải băng thông máy chủ.

  + Cho phép tích hợp các cơ chế NAT traversal như ICE, STUN, TURN.
- Nhược điểm:

  + Cần vận hành một máy chủ signaling (dù nhẹ hơn server truyền thống).

  + Vẫn có trường hợp phải dựa vào TURN server (relay) → tăng chi phí và độ trễ.

  + Yêu cầu bảo mật signaling server để tránh tấn công giả mạo kết nối.

III. CƠ SỞ LÝ THUYẾT

3.1. NAT – Network Address Translation

- Khái niệm:

NAT là kỹ thuật cho phép nhiều thiết bị trong mạng LAN dùng chung một địa chỉ IP công cộng khi truy cập Internet. Router thực hiện việc ánh xạ (mapping) giữa IP/port nội bộ và IP/port công cộng.

- Vai trò trong P2P:

NAT giúp tiết kiệm địa chỉ IPv4 nhưng gây khó khăn cho kết nối trực tiếp vì một thiết bị bên ngoài Internet không thể tự gửi gói tin vào mạng LAN nếu không có ánh xạ sẵn.

- Các loại NAT phổ biến:

+ Full-cone NAT: dễ kết nối P2P.

+ Restricted cone NAT / Port-restricted cone NAT: kết nối được nhưng cần biết trước IP/port đối tác.

+ Symmetric NAT: khó kết nối P2P, thường phải dùng TURN.

3.2. ICE – Interactive Connectivity Establishment

- Khái niệm:

ICE là một khung (framework) được IETF chuẩn hóa (RFC 8445) để tìm và lựa chọn đường kết nối tối ưu giữa hai peer.

- Cơ chế hoạt động:

1. Thu thập các ICE candidates – tập hợp các địa chỉ/port khả thi từ nhiều nguồn:

+ Host candidates: IP/port cục bộ của thiết bị.

+ Server-reflexive candidates: lấy từ STUN server.

+ Relay candidates: lấy từ TURN server.

2. Trao đổi danh sách candidates giữa hai peer qua signaling.

3. Kiểm tra tính liên thông (connectivity checks) bằng cách gửi gói STUN thử nghiệm giữa từng cặp candidate.

4. Chọn đường kết nối ưu tiên: host → server-reflexive → relay.

3.3. STUN – Session Traversal Utilities for NAT

- Khái niệm:

STUN là giao thức nhẹ giúp thiết bị xác định địa chỉ IP công cộng và port mà NAT gán cho nó (server-reflexive address).

- Vai trò trong P2P:

+ Giúp hai peer "nhìn thấy" địa chỉ công khai của nhau để thử kết nối trực tiếp.

+ Duy trì phiên NAT bằng keep-alive.

- Quy trình cơ bản:

1. Peer gửi STUN Binding Request tới STUN server.

2. Server phản hồi Binding Response chứa địa chỉ công cộng.

3. Địa chỉ này được gửi cho peer bên kia qua signaling.

- Hiệu quả & hạn chế:

+ Thành công với đa số NAT cone types (~75–80% trường hợp).

+ Thất bại với symmetric NAT hoặc khi bị chặn UDP → cần TURN.

STUN là giao thức nhẹ giúp client xác định địa chỉ IP công cộng và port của mình khi đứng sau NAT. Client gửi yêu cầu đến STUN server, server phản hồi lại thông tin public-facing IP/port được NAT cấp phát.

- Phần lớn trường hợp (đa số NAT cone types), kết nối P2P được thiết lập thành công chỉ với STUN (~75–80%).

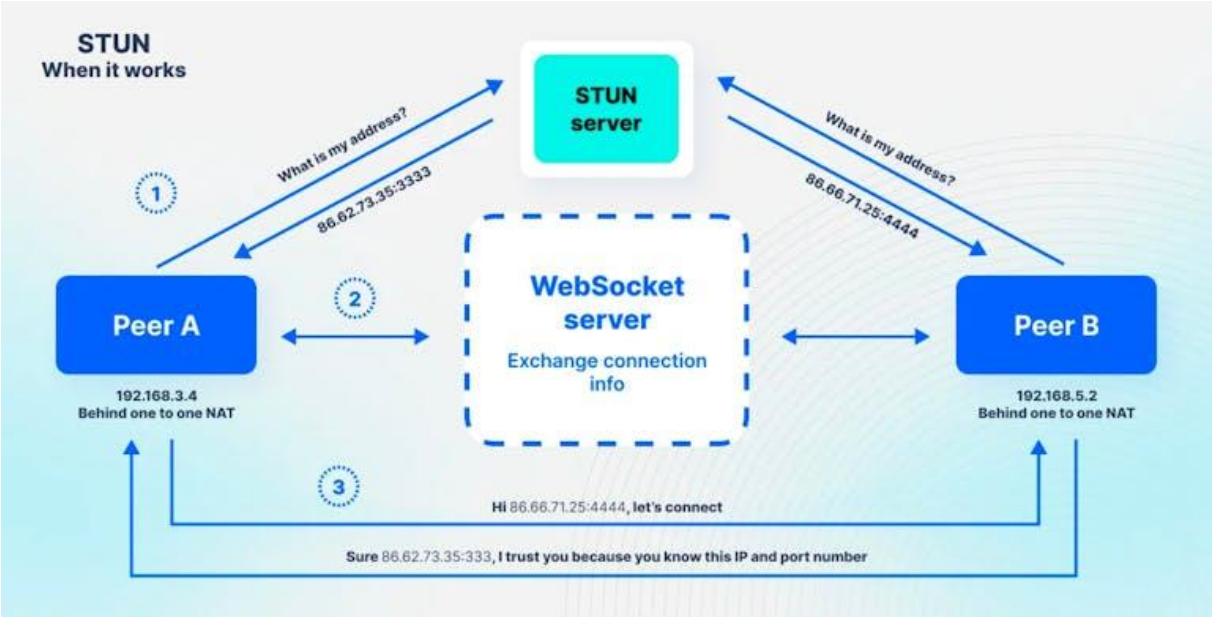
3.3.1 Bản chất & vai trò

- STUN là gì? Giao thức giúp một endpoint đứng sau NAT "nhìn thấy" địa chỉ công khai (server-reflexive) của chính mình (IP:port) bằng cách gửi Binding Request tới STUN server và nhận về Binding Response chứa thuộc tính XOR-MAPPED-ADDRESS. Kết quả này được dùng làm ICE candidate kiểu srflx. STUN cũng được dùng để kiểm tra kết nối (connectivity checks) và giữ sống phiên NAT (keep-alive).
- STUN nằm ở đâu trong WebRTC/ICE? Trong thủ tục ICE (RFC 8445), trình duyệt/ứng dụng thu thập các ứng viên (host, srflx từ STUN, và relay từ TURN), rồi dùng các gói STUN để "thử" từng cặp ứng viên giữa hai bên nhằm tìm tuyến tốt nhất. Nếu các NAT cho phép, kết nối P2P được thiết lập trực tiếp; nếu không, sẽ rơi về TURN (relay).
- STUN không phải là signaling. Signaling (WebSocket/Firebase/…) chỉ để trao đổi SDP/ICE giữa hai peer. STUN phục vụ khám phá địa chỉ và kiểm tra kết nối.

Ghi chú NAT: hành vi NAT (mapping/filtering) rất khác nhau—các biến thể "cone" vs "symmetric" quyết định STUN có đủ để xuyên NAT hay phải dùng TURN.

3.3.2 Khi nào STUN hiệu quả / hạn chế

- Hiệu quả khi NAT hai bên có endpoint-independent mapping/filtering (đa số hộ gia đình / small office), cho phép UDP "đục lỗ" thành công.
- Hạn chế khi gặp symmetric NAT, NAT không hỗ trợ hairpin, hoặc môi trường chặn UDP → thường phải bật TURN (relay) làm phương án dự phòng.



Hình: Mô tả cơ chế STUN – client phát hiện địa chỉ công khai và trao đổi qua signaling để tìm đường kết nối.

https://www.nxrte.com/jishu/webrtc/6848.html

3.4. TURN – Traversal Using Relays Around NAT

- Khái niệm:

TURN là giao thức relay trung gian, cho phép truyền dữ liệu qua một máy chủ khi kết nối trực tiếp thất bại.

- Vai trò trong P2P:

Bảo đảm kết nối ngay cả khi NAT/firewall quá chặt, đặc biệt với symmetric NAT hoặc mạng chặn UDP.

- Cơ chế hoạt động:

1. Peer gửi Allocate Request tới TURN server để yêu cầu địa chỉ relay.

2. TURN cấp Relayed Transport Address – mọi dữ liệu gửi đến địa chỉ này sẽ được chuyển tiếp tới peer còn lại.

3. Thiết lập Permission để cho phép nhận dữ liệu từ IP nhất định.

4. Truyền dữ liệu qua TURN.

- Nhược điểm:

+ Tăng độ trễ do dữ liệu đi vòng qua server.

+ Tiêu tốn băng thông và tài nguyên server.

- Ứng dụng:

Thường triển khai TURN như giải pháp fallback khi ICE thử STUN nhưng thất bại.

Trong nhiều trường hợp, đặc biệt khi peer bị NAT loại symmetric hoặc firewall chặn UDP, STUN không đủ để kết nối. Lúc này, ta cần sử dụng TURN server như một relay – tự server trung gian truyền dữ liệu giữa hai peer.

- TURN đòi hỏi nhiều tài nguyên hơn (bandwidth, xử lý) nên thường đặt làm giải pháp fallback khi STUN thất bại.

3.4.1. Khi nào phải dùng TURN?

- Khi NAT kiểu symmetric, carrier-grade NAT (CGNAT), hoặc tường lửa chặn UDP ⇒ hai peer không thể thiết lập đường đi trực tiếp dù đã dùng STUN/ICE. Lúc này ta chuyển sang relay toàn bộ lưu lượng qua TURN server (máy chủ chuyển tiếp). TURN là một phần của họ giao thức STUN/ICE do IETF chuẩn hoá; đặc tả mới nhất là RFC 8656 (thay thế RFC 5766).

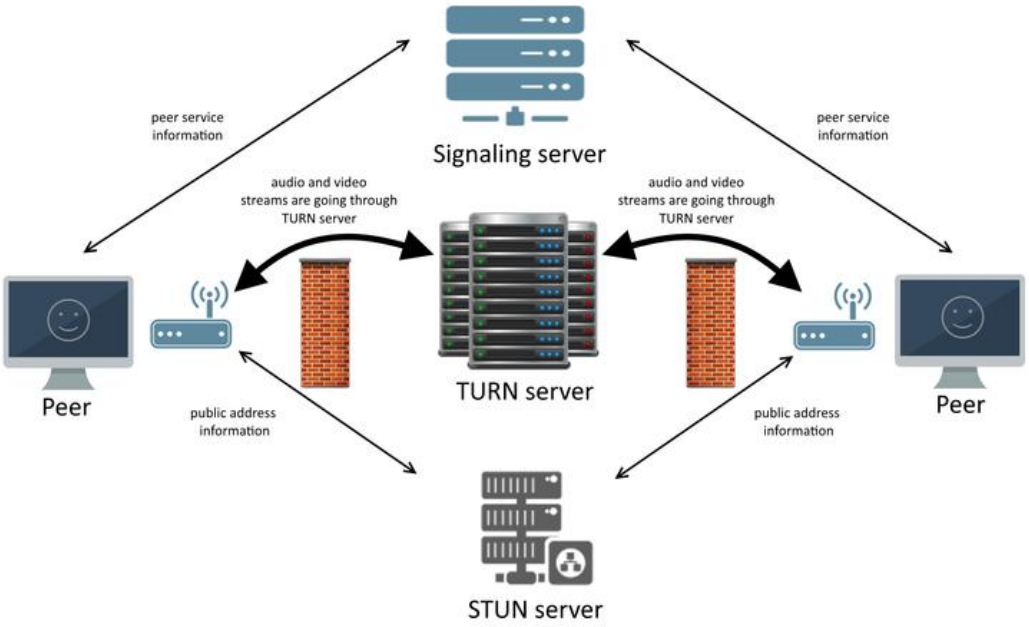
3.4.2. TURN hoạt động thế nào (tóm tắt theo chuẩn)

1. Allocate (cấp phát phiên/địa chỉ chuyển tiếp)
   Client gửi Allocate Request đến TURN server. Nếu thành công, server tạo một allocation và cấp một Relayed Transport Address (IP:port trên TURN) – đây là "địa chỉ công khai" mà peer còn lại sẽ gửi dữ liệu tới. Allocation được gắn với 5-tuple (src/dst IP, src/dst port, protocol). Thời gian sống mặc định của allocation là 10 phút, client phải Refresh định kỳ.
2. Permissions (cấp quyền nhận từ peer cụ thể)
   Mỗi allocation duy trì danh sách permission theo IP peer; nếu không có permission phù hợp, gói đến sẽ bị DROP. Thời gian sống mặc định của permission là 300s.
3. ChannelBind / Data Indication (truyền dữ liệu)
   a. DATA/Send Indication: đơn giản, mỗi gói kèm header STUN/TURN.
   b. ChannelBind: ràng kênh số (channel number) với địa chỉ peer để giảm overhead header, hiệu quả cho lưu lượng âm thanh/hình ảnh liên tục.

4. Vận chuyển TURN hỗ trợ UDP (phổ biến nhất). Khi UDP bị chặn, có thể dùng TURN-over-TCP/TLS; phần mở rộng RFC 6062 định nghĩa TCP allocations (mở kết nối TCP từ TURN tới peer), dù trong WebRTC thực tế chủ yếu dùng TURN/TCP (mang UDP bên trong) hoặc TURNS (TLS trên 5349).
5. ICE ưu tiên relay thấp nhất. Trong WebRTC, ICE sẽ thử host → server-reflexive (STUN) → relay (TURN); relay chỉ được dùng khi mọi đường trực tiếp đều thất bại.

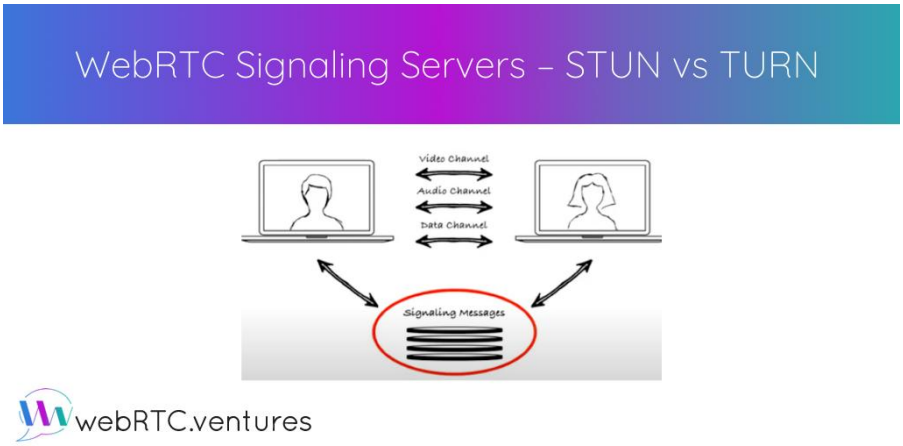
3.4.3. Tác động hệ thống & khuyến nghị triển khai

- Độ trễ & băng thông: Mọi media/data đi vòng qua TURN ⇒ tăng latency và tốn băng thông/chi phí; cần đặt TURN gần người dùng, đa điểm POP, bật ChannelBind cho media.
- Bảo mật: Dùng long-term credentials của TURN (nonce hết hạn ~600 s), cấp mật khẩu tạm thời (ephemeral) phía ứng dụng; triển khai TURNS (TLS, cổng 5349) cho môi trường firewall nghiêm ngặt.
- Vận hành: Giám sát tỷ lệ rơi về TURN, CPU/băng thông, tỷ lệ refresh allocation/permission; điều chỉnh giới hạn lifetime khi cần (theo coturn: channel=600 s, permission=300 s).



Hình: Toàn cảnh kiến trúc signaling server trung gian, STUN, TURN để hỗ trợ khởi tạo kết nối P2P. Minh họa giai đoạn fallback, dữ liệu sẽ đi qua TURN relay thành công

https://stackoverflow.com/questions/73372258/peer-to-peer-p2p-android-chat-application-over-internet-using-webrtc



Hình: So sánh tổng quan giữa STUN và TURN trong bối cảnh signaling.

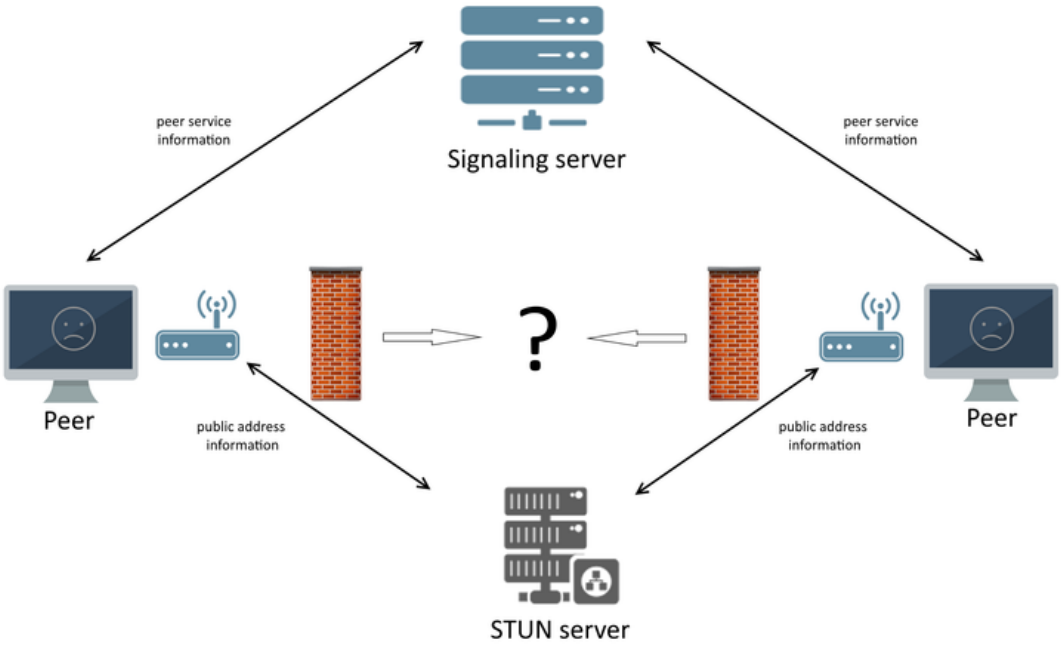
https://webrtc.ventures/2020/12/webrtc-signaling-stun-vs-turn/

3.5. Quan hệ giữa ICE, STUN, TURN

- ICE là khung điều phối → quyết định dùng STUN hay TURN.
- STUN là phương án ưu tiên để thiết lập kết nối trực tiếp, tiết kiệm tài nguyên.
- TURN là giải pháp cuối cùng khi mọi đường kết nối trực tiếp đều thất bại.

3.6. Luồng kết nối tổng thể

1. Peer A và B kết nối đến Signaling Server để trao đổi SDP và ICE candidates.
2. Mỗi peer dùng STUN để xác định địa chỉ public → gửi candidates qua signaling.
3. ICE chọn đường kết nối khả thi nhất:
    - Direct (P2P) nếu khả thi
    - Turn relay nếu direct fail



Hình: hình ảnh minh họa hệ thống tổng thể WebRTC kết hợp signaling, STUN, TURN, và ICE. Minh họa trạng thái khi kết nối trực tiếp P2P không thành công và chưa có TURN server relay.

https://dev.to/alexsam986/how-to-integrate-audio-video-calls-using-webrtc-javascript-4ecn

3.6.1. Các thành phần

- Peer A & Peer B: Hai thiết bị/ứng dụng muốn kết nối trực tiếp để trao đổi dữ liệu (chat, video call, gửi file…). Trong hình, cả hai Peer đều có biểu tượng → biểu thị kết nối chưa thành công.
- Signaling Server: Trung gian chỉ dùng để trao đổi thông tin khởi tạo kết nối (SDP, ICE candidates). Không truyền dữ liệu media.
- STUN Server: Dùng để giúp mỗi Peer biết được địa chỉ IP công khai và port khi ở sau NAT. Sau khi biết thông tin này, mỗi Peer sẽ gửi qua Signaling Server để bên kia thử kết nối trực tiếp.
- Router/NAT + Firewall (bức tường đỏ): Đại diện cho rào cản mạng (Network Address Translation, firewall rules) có thể chặn UDP hoặc giới hạn kết nối vào.
- Dấu chấm hỏi (?) ở giữa: Thể hiện rằng quá trình direct P2P connection thất bại → không tìm được đường truyền khả thi trực tiếp.

3.6.2. Luồng xử lý minh họa trong hình

1. Kết nối ban đầu & Signaling
    a. Peer A và Peer B cùng kết nối tới Signaling Server.
    b. Gửi/nhận Session Description Protocol (SDP) và ICE candidates qua kênh signaling.
2. Xác định địa chỉ Public qua STUN
    a. Mỗi Peer gửi yêu cầu tới STUN server để biết public IP/port của mình (server-reflexive candidate).
    b. Thông tin này được gửi sang Peer kia thông qua Signaling Server.
3. ICE kiểm tra kết nối trực tiếp

a. ICE (Interactive Connectivity Establishment) ở mỗi Peer sẽ thử tạo kết nối trực tiếp (host → srflx).
b. Nếu NAT hỗ trợ (Full-cone, Restricted-cone), kết nối thường thành công.

4. Kết quả
    a. Trong hình, cả hai Peer đều không thể kết nối trực tiếp → dấu "?" tượng trưng cho việc chưa tìm ra đường truyền khả thi.
    b. Ở thực tế, bước tiếp theo sẽ là fallback sang TURN server để relay.

IV. CÔNG NGHỆ SỬ DỤNG

4.1. Android Framework

- Khái niệm:

Android là hệ điều hành mã nguồn mở dành cho thiết bị di động, được Google phát triển. Android Framework cung cấp bộ API và thư viện để lập trình viên xây dựng ứng dụng native.

- Vai trò trong dự án:

+ Xây dựng giao diện người dùng (UI) cho ứng dụng chat P2P.

+ Tích hợp với WebRTC Native API để xử lý luồng dữ liệu thời gian thực.

+ Lưu trữ tin nhắn local tại thiết bị

+ Tương tác với các dịch vụ Firebase thông qua SDK chính thức cho Android.

4.2. WebRTC (Web Real-Time Communication)

- Khái niệm:

WebRTC là tiêu chuẩn mở do W3C và IETF phát triển, cho phép truyền dữ liệu thời gian thực (video, audio, data) giữa các thiết bị thông qua kết nối P2P.

- Vai trò trong dự án:

+ Thiết lập kết nối P2P giữa hai người dùng để trao đổi dữ liệu chat.

+ Sử dụng DataChannel để gửi và nhận tin nhắn văn bản trực tiếp.

+ Tận dụng cơ chế ICE (Interactive Connectivity Establishment), STUN và TURN để vượt qua NAT/firewall.

+ Mã hóa dữ liệu truyền qua DTLS và SRTP, đảm bảo bảo mật end-to-end.

- Ưu điểm:

+ Không cần cài plugin bổ sung.

+ Hỗ trợ mã hóa mặc định.

+ Giảm tải cho server vì dữ liệu chat đi trực tiếp giữa các thiết bị.

4.3. Signaling Server

- Khái niệm:

Signaling server là máy chủ trung gian dùng để trao đổi thông tin cần thiết cho việc thiết lập kết nối WebRTC giữa hai peer, như SDP (Session Description Protocol) và ICE candidates.

- Vai trò trong dự án:

+ Xác định vai trò kết nối (Caller – người tạo offer, Callee – người trả lời).

+ Chuyển tải thông tin offer/answer và ICE candidates giữa hai peer.

+ Không truyền dữ liệu chat, chỉ phục vụ giai đoạn "bắt tay" kết nối.

Trong dự án này, signaling server được triển khai thông qua Firebase Realtime Database

4.4. Firebase

Firebase là nền tảng phát triển ứng dụng của Google, cung cấp nhiều dịch vụ backend dạng "BaaS" (Backend as a Service), giúp giảm thời gian và chi phí phát triển server. Trong dự án này, nhiều dịch vụ Firebase được sử dụng.

4.4.1. Firebase Authentication

- Vai trò: Xác thực và quản lý danh tính người dùng
- Lợi ích:

+ Triển khai nhanh, bảo mật cao.

+ Hỗ trợ nhiều hình thức đăng nhập mà không cần tự xây hệ thống xác thực.

4.4.2. Cloud Firestore

- Khái niệm:

Cơ sở dữ liệu NoSQL dạng document/collection, hỗ trợ truy vấn linh hoạt hơn Realtime Database.

- Vai trò trong dự án:

+ Là kênh signaling cho WebRTC.

+ Quản lý tài khoản người dùng

+ Quản lý danh sách bạn bè, lời mời kết bạn

- Ưu điểm:

+ Khả năng mở rộng tốt, truy vấn linh hoạt.

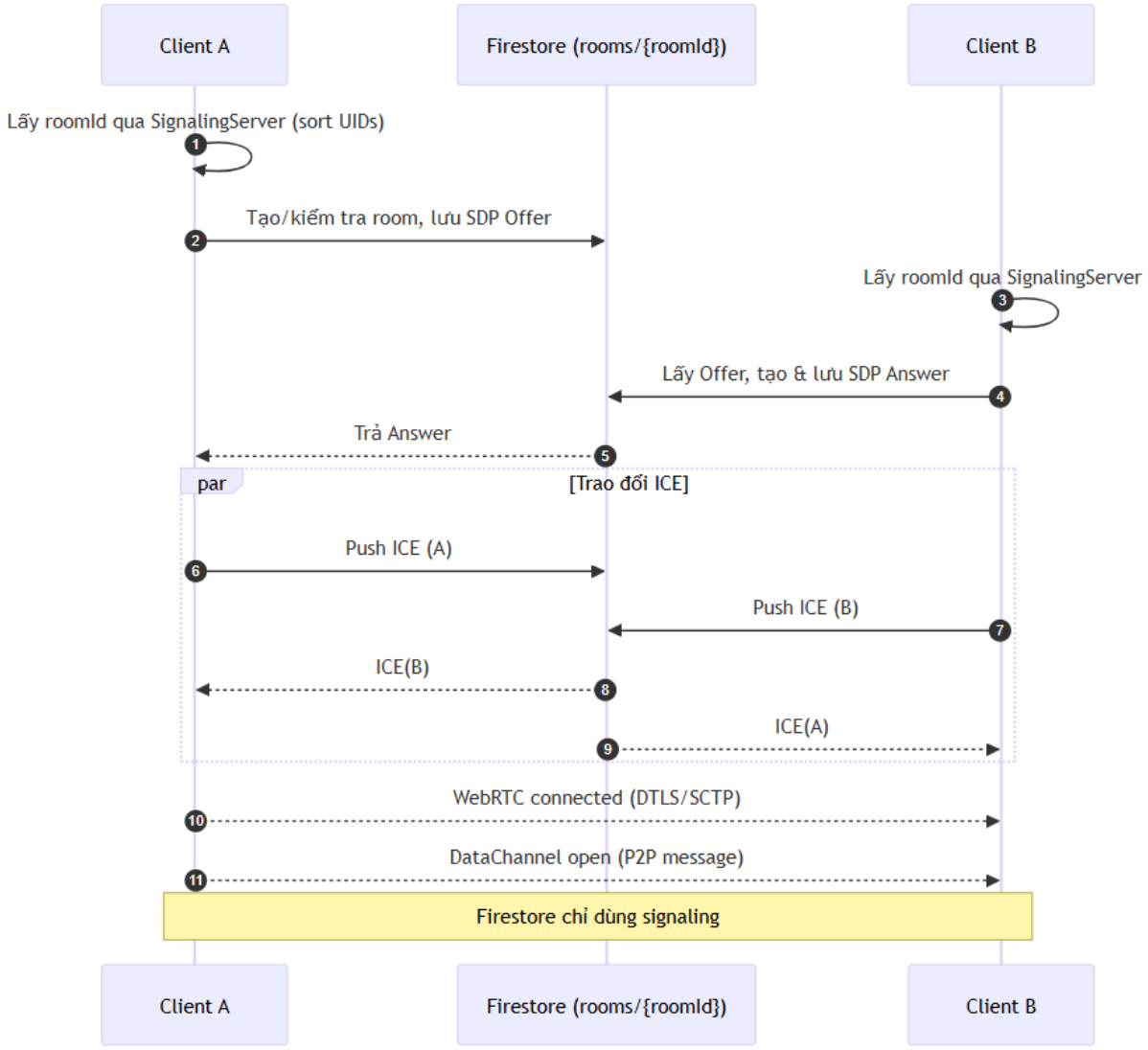
+ Dữ liệu được phân vùng theo collection → dễ tổ chức.

V. ỨNG DỤNG

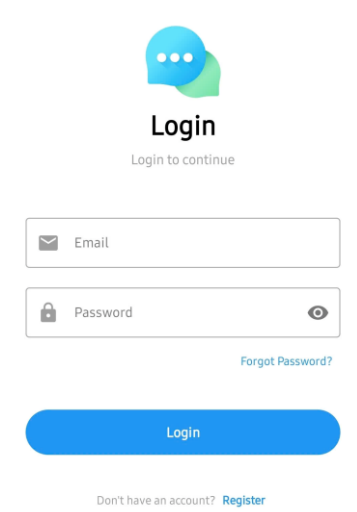
- Kiến trúc tổng quan: WebRTC:
    - Tạo kết nối P2P trực tiếp giữa 2 user
    - Firebase Firestore: Làm signaling server để trao đổi thông tin kết nối
    - STUN Server: Hỗ trợ NAT traversal (sử dụng Google STUN)
    - Sử dụng Room database của Android để lưu trữ tin nhắn tại thiết bị (local)\
- Các tính năng chính:
    - Đăng ký, đăng nhập (sử dụng firebase auth)
    - Quản lý bạn bè (Firebase firestore)
    - Nhắn tin (Web RTC + Firebase firestore)
    - Lưu trữ tin nhắn local (Room database)
- Quy trình kết nối (9 giai đoạn):

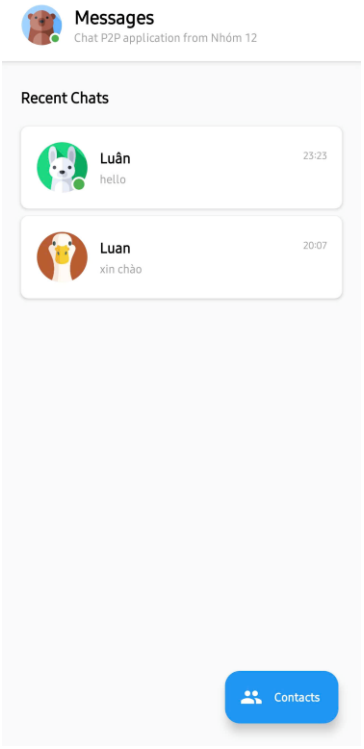
1. Room Creation: Tạo room và xác định vai trò (Caller/Callee)
2. Join Room: Cả 2 user join vào room và lắng nghe trạng thái
3. Caller Flow: User đầu tiên tạo offer và gửi lên Firestore
4. Callee Flow: User thứ 2 nhận offer và tạo answer
5. WebRTC Handshake: Hoàn tất quá trình trao đổi SDP
6. ICE Candidate Exchange: Trao đổi thông tin mạng qua Firestore
7. P2P Connection: Thiết lập kết nối trực tiếp qua WebRTC DataChannel
8. Messaging: Gửi tin nhắn trực tiếp qua DataChannel (không qua server)
9. Cleanup: Dọn dẹp khi rời khỏi chat

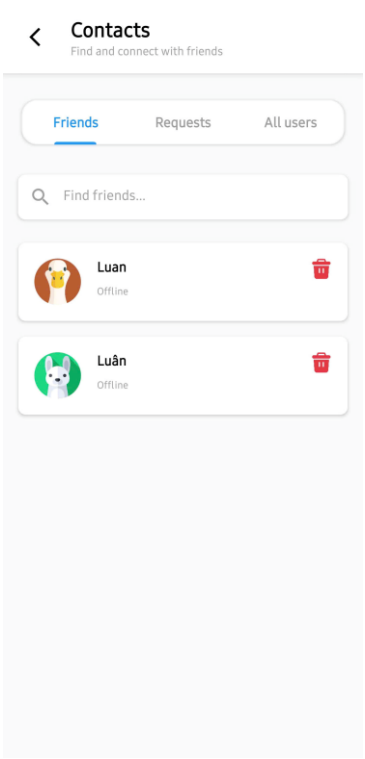
Sơ đồ tạo kết nối giữa hai người dùng:

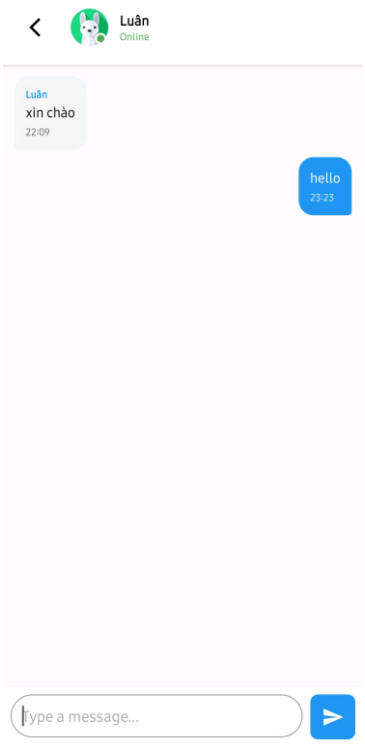


Một số hình ảnh của ứng dụng:









VI. KẾT LUẬN

Kiến trúc P2P với signaling chứng minh là lựa chọn đúng đắn cho bài toán chat thời gian thực hướng tới độ trễ thấp, chi phí máy chủ tối thiểu và tính riêng tư cao, đồng thời đơn giản hóa đáng kể lớp backend.

Dù vậy, để đạt chất lượng sản phẩm sản xuất (production-grade), nhóm phát triển cần đầu tư thêm cho quy trình vận hành, giám sát chất lượng, xử lý góc cạnh trên Android, và kế hoạch mở rộng chức năng (nhóm, file, thông báo, hiện diện).

Hạn chế:

- Việc tạo quá nhiều connection bằng Web RTC gây tốn tài nguyên hệ thống nên việc xử lý tin nhắn khi người dùng không ở trong chat là một thách thức lớn
- Cần theo dõi hạn mức đọc/ghi, quy tắc bảo mật trên Firestore/Realtime Database và chi phí Storage khi mở rộng gửi tệp
- Không có khả năng đồng bộ dữ liệu trên nhiều thiết bị

Hướng phát triển:

- Kết hợp với server để phát triển tính năng thông báo tin nhắn khi người dùng offline
- Phát triển thêm tính năng gửi ảnh, gửi file
- Web RTC được xây dựng để hỗ trợ mạnh mẽ cho video call, ứng dụng sẽ phát triển tính năng video call trong tương lai

TÀI LIỆU THAM KHẢO

[1] Building a Video Chat App: WebRTC on Android (Part1) Building a Video Chat App: WebRTC on Android (Part1) | by Jaewoong Eum | ProAndroidDev

[2] WebRTC for Android and Kotlin Developers WebRTC on Android with Kotlin

[3] Android P2P Chat Web-RTC Android P2P Chat Web-RTC. Creating a fully functional p2p chat… | by dalakoti07 | ProAndroidDev

[4] Tìm hiểu sơ lược về Firebase Tìm hiểu sơ lược về Firebase