WebRtc, Network, Signaling server, Callkit incoming, PushKit, Firebase push notification

I/WebRtc

Trong ứng dụng VoIP, webRtc đóng các vai trò:

-Webrtc cung cấp các api để có thể tương tác được với các phần cứng của device như camera, micro... (bật/tắt micro/camera, chọn camera trước/sau…

-Cung cấp giải pháp biến những hình ảnh, âm thanh thu được từ camera, micro... thành media stream để hiển thị trên local và truyền đi

-Cung cấp giải pháp kết nối ngang hàng peer to peer,mỗi peer hiểu là 1 device tham gia cuộc thoại, dữ liệu (hình ảnh âm thanh) sẽ được truyền trực tiếp từ peer này sang peer kia (nếu trong mạng local) hoặc thông qua 1 turn server (nếu 2 peer khác mạng).

-Cung cấp giải pháp mã hóa video/âm thanh khi truyền đi để tối ưu dung lượng và làm chất lượng truyền tốt hơn.

-Cung cấp giải pháp chuyển data giữa 2 peer như truyền file, truyền các message (các tín hiệu như đầu bên này bật, tắt camera…).

-Cung cấp giải pháp để chụp các khung hình trong video…

Tất cả vấn đề này có thể xem trong project webrtc\_example.

II/Kiến thức về network

-Xem thêm ở phần study network.

1/NAT có thể ảnh hưởng đến ứng dụng VoIP

Khi VoIP được sử dụng trong mạng LAN với NAT, có thể gặp các vấn đề sau:

Vấn đề kết nối: NAT có thể làm cho các cuộc gọi VoIP gặp khó khăn trong việc thiết lập kết nối trực tiếp giữa các thiết bị. Điều này có thể dẫn đến thời gian kết nối chậm hoặc thất bại trong việc thiết lập cuộc gọi.

Vấn đề âm thanh: Do NAT có thể gây ra mất mát dữ liệu và trễ trong mạng, có thể dẫn đến giảm chất lượng âm thanh trong các cuộc gọi VoIP.

Để giảm thiểu vấn đề này, có một số giải pháp, như sử dụng protcol STUN (Session Traversal Utilities for NAT) hoặc TURN (Traversal Using Relays around NAT) để hỗ trợ thiết lập kết nối qua NAT

2/Tường lửa ảnh hưởng thế nào đến ứng dụng VoIP

-Chặn các cổng và giao thức: Một số tường lửa có thể chặn các cổng và giao thức được sử dụng bởi ứng dụng VoIP, như UDP (User Datagram Protocol) và các cổng không tiêu chuẩn. Điều này có thể làm cho các cuộc gọi VoIP gặp khó khăn trong việc thiết lập kết nối và truyền dữ liệu.

-Giảm chất lượng âm thanh: Tường lửa có thể gây trễ và mất mát gói tin, dẫn đến giảm chất lượng âm thanh trong các cuộc gọi VoIP. Nếu tường lửa được cấu hình chặt chẽ, nó có thể làm giảm hiệu suất của dịch vụ VoIP.

-Hạn chế kết nối đa hướng: VoIP thường sử dụng kết nối đa hướng (peer-to-peer) để truyền dữ liệu trực tiếp giữa các thiết bị. Tuy nhiên, một số tường lửa có thể ngăn chặn kết nối đa hướng, đòi hỏi dữ liệu phải đi qua máy chủ trung gian, gây ra trễ và giảm chất lượng cuộc gọi.

-Quản lý băng thông: Một số tường lửa có tính năng quản lý băng thông, có thể ưu tiên các loại dịch vụ khác nhau. Nếu không được cấu hình đúng, nó có thể ưu tiên các dịch vụ khác trước dịch vụ VoIP, gây ra giảm chất lượng trong cuộc gọi.

🡺Để giảm tác động của tường lửa đối với ứng dụng VoIP, cần cấu hình tường lửa một cách cẩn thận, cho phép các giao thức và cổng được sử dụng bởi VoIP đi qua mạng một cách mượt mà và hiệu quả. Ngoài ra, sử dụng các giải pháp STUN, hoặc TURN có thể giúp giải quyết các vấn đề liên quan đến tường lửa và VoIP.

3/Stun server đóng vai trò gì trong ứng dụng VoIP

-Firewall và Nat gây ra những vấn đề khó khăn khi kết nối 2 peer ở 2 mạng khác nhau.

-STUN server giúp ứng dụng VoIP vượt qua các vấn đề liên quan đến NAT, Firewall bằng cách cung cấp thông tin về địa chỉ IP công cộng của các thiết bị trong mạng nội bộ, cho phép thiết lập kết nối trực tiếp và cải thiện trải nghiệm cuộc gọi VoIP.

- Ví dụ: Khi hai điện thoại thông minh A và B muốn thiết lập kết nối VoIP với nhau, mỗi điện thoại sẽ gửi yêu cầu STUN tới STUN server để xác định địa chỉ IP công cộng của nó. Sau đó, A và B biết được địa chỉ IP công cộng của nhau thông qua STUN server. Tiếp theo, A và B có thể truyền dữ liệu trực tiếp với nhau qua địa chỉ IP công cộng này, mà không cần thông qua STUN server nữa.

STUN server chỉ đóng vai trò như một "hướng dẫn viên" giúp các peer tìm ra địa chỉ IP công cộng của nhau để có thể thiết lập kết nối trực tiếp. Sau khi đã tìm ra địa chỉ này, việc truyền dữ liệu diễn ra trực tiếp giữa các peer, không thông qua STUN server.

- Trong trường hợp ứng dụng VoIP chỉ hoạt động nội bộ mà không kết nối đến mạng Internet, các thiết bị trong mạng nội bộ có thể trao đổi dữ liệu âm thanh (voice data) với nhau một cách trực tiếp thông qua địa chỉ IP nội bộ mà không gặp vấn đề NAT. Trong tình huống này, không có sự cần thiết hay lợi ích sử dụng STUN server.

4/Turn server đóng vai trò gì trong ứng dụng VoIP

-Trong ứng dụng VoIP (Voice over Internet Protocol), TURN server (Traversal Using Relays around NAT) đóng vai trò quan trọng khi việc thiết lập kết nối trực tiếp giữa các thiết bị không khả dụng do vấn đề NAT hoặc tường lửa (firewall) ngăn cản.

-Khi các thiết bị VoIP cố gắng thiết lập kết nối trực tiếp thông qua STUN server nhưng không thành công (ví dụ: NAT loại đối tác hoặc NAT không đủ thông minh để thông qua STUN), TURN server được sử dụng như một trung gian (relay) để đảm bảo các thiết bị có thể trao đổi dữ liệu âm thanh (voice data).

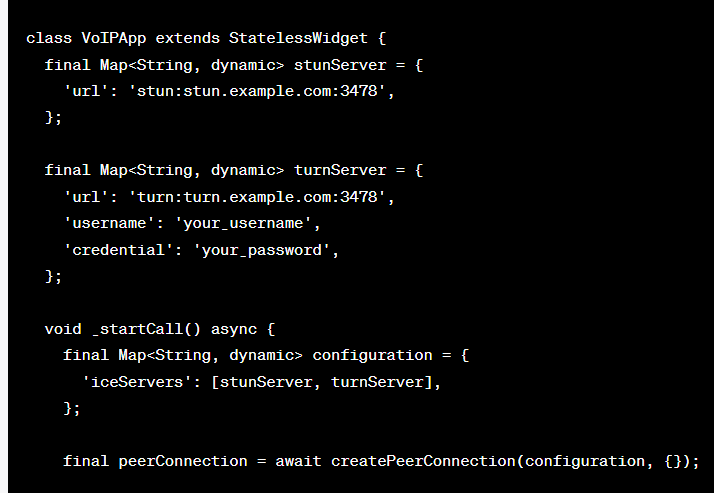
-Cụ thể, đây là cách mà TURN server hoạt động:

+Phương thức relay: Khi các thiết bị không thể thiết lập kết nối trực tiếp, chúng sẽ gửi dữ liệu âm thanh thông qua TURN server. TURN server nhận dữ liệu từ thiết bị gửi và chuyển tiếp nó đến thiết bị đích thông qua kết nối của nó với các thiết bị đó.

+Địa chỉ IP công cộng: Vì TURN server có địa chỉ IP công cộng và không bị ảnh hưởng bởi NAT, nó có thể giúp các thiết bị kết nối với nhau một cách hiệu quả. Dữ liệu từ thiết bị gửi đi sẽ được định tuyến thông qua TURN server để đảm bảo dữ liệu chính xác đến thiết bị đích.

🡺Điều này giúp giải quyết vấn đề với NAT và tường lửa mà không yêu cầu sửa đổi cấu hình mạng. Tuy nhiên, việc sử dụng TURN server có thể tăng đáng kể độ trễ và giảm hiệu suất so với kết nối trực tiếp thông qua STUN server. Do đó, TURN server thường được sử dụng khi kết nối trực tiếp không khả dụng và là giải pháp cuối cùng để đảm bảo ứng dụng VoIP hoạt động trơn tru và ổn định.

-Dưới đây là 1 ví dụ về cách cấu hình car STUN server và TURN server.



<https://developer.mozilla.org/en-US/docs/Web/API/RTCPeerConnection/RTCPeerConnection>

Cấu hình chi tiết xem trong link này.

5/ICE (Interactive Communication Establishment)

ICE nôm na dễ hiểu là một giao thức được cùng để thiết lập phiên media dựa trên UDP đi qua NAT một cách nhanh nhất.

ICE sẽ tìm đường tốt nhất để kết nối giữa các peer, nó thử tất cả khả năng có thể kết nối một cách song song và lựa chọn con đường hiệu quả nhất.

Đầu tiên nó sẽ cố gắng tạo ra một kết nối bằng cách sử dụng địa chỉ thu được từ hệ điều hành và card mạng của thiết bị (private IP).

Nếu không thành công (có thể thiết bị đằng sau NAT) thì ICE sẽ lấy địa chỉ bên ngoài của thiết bị (public IP) bằng cách sử dụng máy chủ STUN.

Nếu không thành công nữa thì nó sẽ chuyển lưu lượng mạng qua một máy chủ chuyển tiếp là TURN.

Nếu khó nhớ thì các bạn cứ nghĩ là ICE xài STUN xong không được thì đi xài TURN.

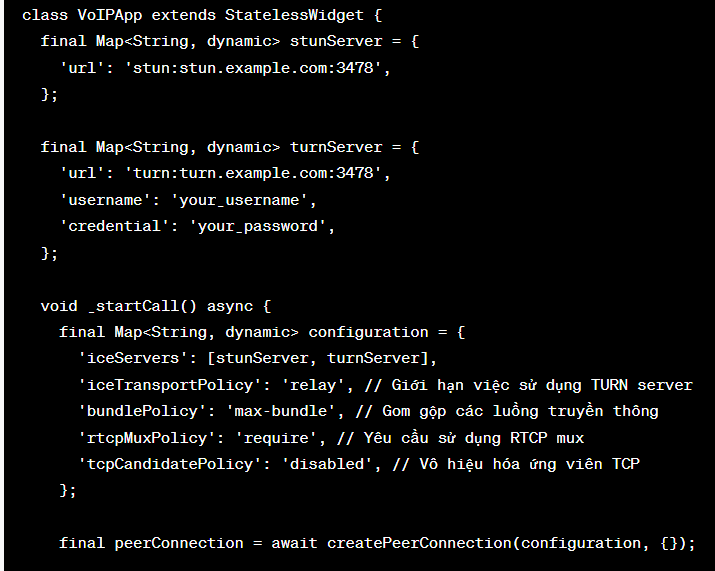
5/Các kỹ thuật để giảm chi phí khi sử ụng TURN server

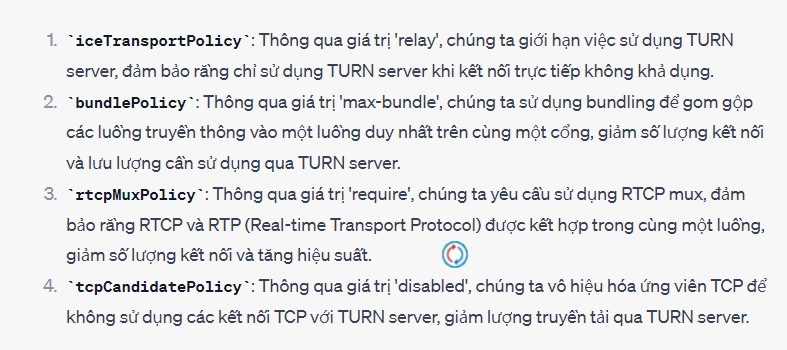
-Sử dụng STUN trước tiền. Cấu hình cả STUN server và TURN server

-Tối ưu hóa mạng: Tối ưu hóa mạng của bạn để giảm lưu lượng truyền tải qua TURN server.

Có thể xem xét sử dụng kỹ thuật hợp nhất (bundling) để gom gộp các luồng truyền thông thành một luồng duy nhất trên cùng một cổng. Điều này giúp giảm số lượng kết nối và lưu lượng cần sử dụng qua TURN server.

Các kỹ thuật trên được thực hiện thông qua các config khi create peer connection.





III/Signaling server

Vai trò của Signaling server

1/Nắm giữ thông tin kết nối và hỗ trợ việc các peer kết nối

<https://viblo.asia/p/webrtc-basic-phan-2-cach-thiet-lap-mot-cuoc-goi-thong-qua-webrtc-MgNvWWpRvYx>

-WebRtc có phương thức để giúp kết nối 2 peer với nhau. Tuy nhiên trước đó thì 2 peer không hề có thông tin của nhau 🡺 Không thể kết nối được. Lúc này cần Signaling server để 2 peer có thể biết đến nhau.

Ví dụ: Máy A muốn gọi Máy B.

St1: Máy A lấy các thông tin về IP, port của mình (có thể là lấy từ STUN, TURN..) gửi cho signaling server. (Dùng socket hoặc api)

St2: Signaling server sử dụng các kỹ thuật như firebase push notification, pushkit để push message về cho Máy B, wake máy B dậy, truyền cho máy B thông tin của máy A.

St3: Máy B nếu đồng ý tham gia cuộc gọi sẽ lấy các thông tin về IP, port của mình gửi cho server.

St4: Server thông qua socket/pushkit/firebase gửi thông tin của máy B về cho máy A (thường là socket).

Sau bước này thì cả máy A và máy B đã có thông tin của nhau và hoàn toàn có thể chủ động tự do kết nối.

St5: Trong quá trình gọi, nếu 1 bên tắt máy hoặc mất kết nối… thì signaling server cũng có thể báo về cho bên máy kia để cập nhật giao diện, kết thúc cuộc gọi…

-Ở St3, nếu máy B không đồng ý kết nối thì sẽ báo cho signaling server , signaling server sẽ báo lại cho máy A và kết thúc giao dịch

-Các step trên chỉ là mô phỏng quá trình kết nối, chi tiết các step xem trên link trên.

2/Quản lý trạng thái kết nối

- Signaling server theo dõi trạng thái kết nối của các thiết bị. Khi một cuộc gọi được thiết lập, Signaling server sẽ theo dõi trạng thái kết nối, cập nhật và duy trì thông tin về cuộc gọi.

Khi 1 thiết bị mất kết nối(mất thông tin socket) hoặc chủ động kết thúc cuộc gọi (chủ động báo server) thì server sẽ nắm bắt được trạng thái của 1 peer và báo cho các peer còn lại.

3/Quản lý yêu cầu chấp nhận và từ chối

Khi máy A gọi máy B, máy B có thể muốn chấp nhận hoặc từ chối cuộc gọi. Signaling server giúp truyền tải thông điệp này đến máy A

4/Truyền các thông điệp đến các peer.

-Mặc dù các peer có thể tự truyền message/data cho nhau thông qua data chanel nhưng signaling server cũng có thể truyền các thông điệp đến các peer thông qua socket trong trường hợp các peer đó chưa thể connect với nhau.

III/Callkit Incoming

Đóng vai trò show ra màn hình có cuộc gọi đến ngay cả khi app đang terminate.

1/Android

Màn hình cuộc gọi đến là 1 custom local notification

2/Ios

-Sử dụng callkit của ios

-Cho phép tích hợp sâu vào ứng dụng gọi điện của hệ điều hành. Khi có cuộc gọi đến app thì thông tin cuộc gọi cũng sẽ xuất hiện trong lịch sử của ứng dụng gọi điện.

IV/Pushkit, Firebase push notification.

1/ Firebase push notification.

-Đóng vai trò đánh thức device android dậy để show ra màn hình call incoming.

-Khi máy A muốn gọi máy B thì máy A báo cho signaling server. Server push firebase notification về cho máy B.

Máy B nhận được notification sẽ có thể thức dậy và show call incoming screen ở bước trên.

-Firebase notification luôn có độ trễ, để tránh việc máy A thực hiện cuộc gọi cho máy B nhưng 1 lúc sau máy B mới nhận được notification thì tốt nhất nên set expired time của notification này rất ngắn, khoảng vài chục giây để nếu trong thời gian đó tin nhắn ko về thì message cũng expired luôn.

2/PushKit.

-Trên ios, khả năng thực hiện các task vụ khi app terminate là rất không ổn định lúc được lúc không.

-Nhưng pushkit hoạt động rất ổn định. Đóng vai trò tương tự như firebase notification bên trên nhưng ổn định hơn.

-Note: khi pushkit bắn đến, nếu app đang terminate thì lập tức hàm main sẽ được chạy nhưng app ko bật lên, nó sẽ keep app sống trong 1 thời gian, do vậy ngay lúc đó mở app lên thì hàm main sẽ ko chạy lại nữa vì đã chạy trước đấy rồi.

-callkit\_incoming\_example + app kèm theo mô tả chi tiết việc dùng pushkit/firebase bắn thông điệp về cho app rồi app dùng callkit để show màn hình call incoming.

Đã test app trên ios.