Audio Calling App (Calto)

Group Members:

- Awais Raza (BSCS51F21R009)
- Zain-ul-Abidden (BSCS51F21R034)
- M Yahya Khalid (BSCS51F21R045)

Subject:

- Internet Architecture and Protocols

Submitted to:

- Sir Farooq Javed

Contents

Research on Working

1. Call Establishment:

- The calling system initiates a call request, which includes session information such as the caller's and callee's identifiers and capabilities (e.g., supported codecs).
- Signalling protocols like SIP (Session Initiation Protocol) or WebSocket are used to exchange call signalling messages between the caller and the callee.

2. NAT Traversal:

- If the caller and callee are behind Network Address Translation (NAT) devices, ICE (Interactive Connectivity Establishment) techniques are employed.
- ICE helps determine the best communication path by gathering candidate IP addresses and testing connectivity using STUN (Session Traversal Utilities for NAT) servers.

3. Peer-to-Peer Connection:

- After NAT traversal is successful, the calling system establishes a direct peer-to-peer connection between the caller and the callee.
- This connection is established using secure transport protocols for encryption and key exchange. example: DTLS (Datagram Transport Layer Security).

4. Media Transmission:

- With the peer connection established, the calling system uses SRTP (Secure Real-Time Transport Protocol) for transmitting encrypted audio and video data between the caller and the callee.
- SRTP ensures end-to-end security by encrypting media streams using algorithms like AES.

5. Call Management and Monitoring:

- During the call, the calling system manages media streams, handles codec negotiation for audio and video, and adapts to network conditions (e.g., adjusting bitrate for optimal quality).
- Call control messages, such as mute/unmute commands or call termination requests, are exchanged between the caller and the callee using signalling protocols.
- The calling system monitors network conditions and applies Quality of Service (QoS) measures to maintain call quality.
- This may include packet loss mitigation, jitter buffering, and prioritization of audio/video streams.

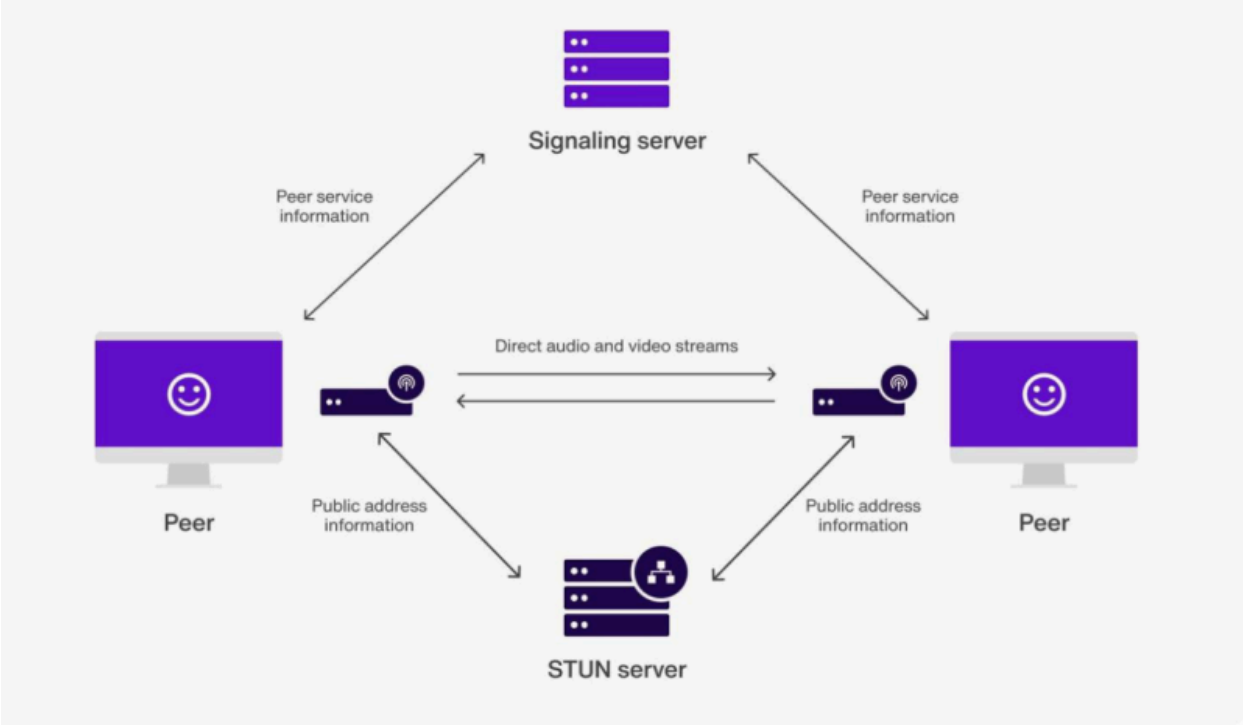
6. Security Measures:

- To ensure secure communication, the calling system employs encryption algorithms like AES for media encryption.
- Authentication mechanisms, such as certificate-based authentication or token validation, are used to verify the identities of callers and protect against unauthorized access.

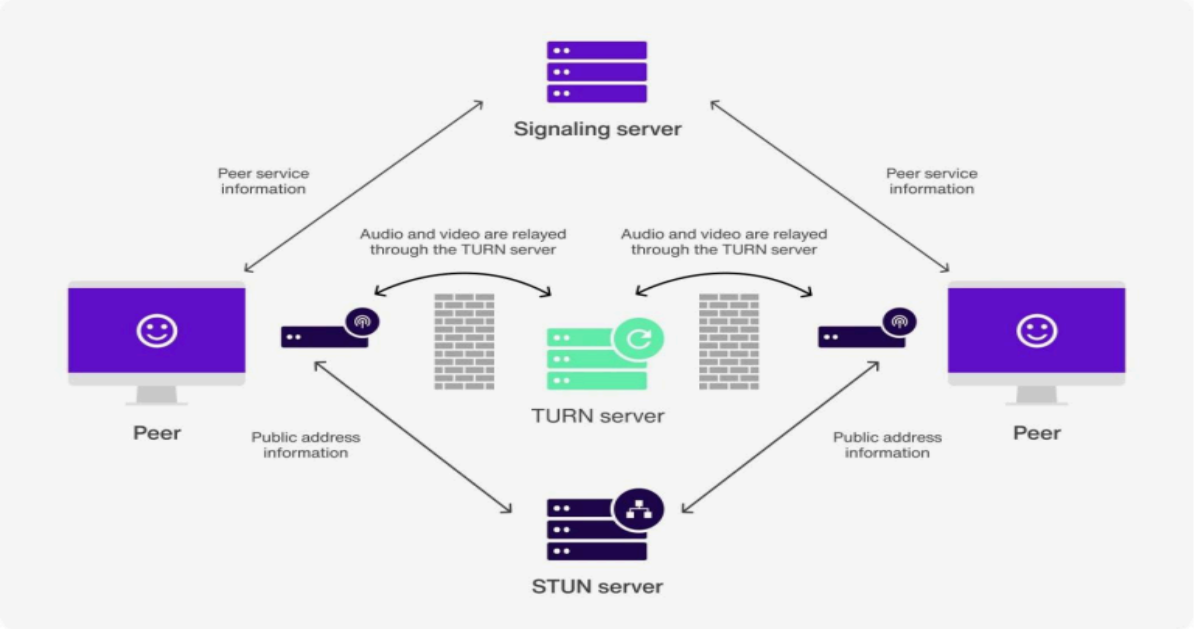
Basic Architecture of Call Using WebRTC

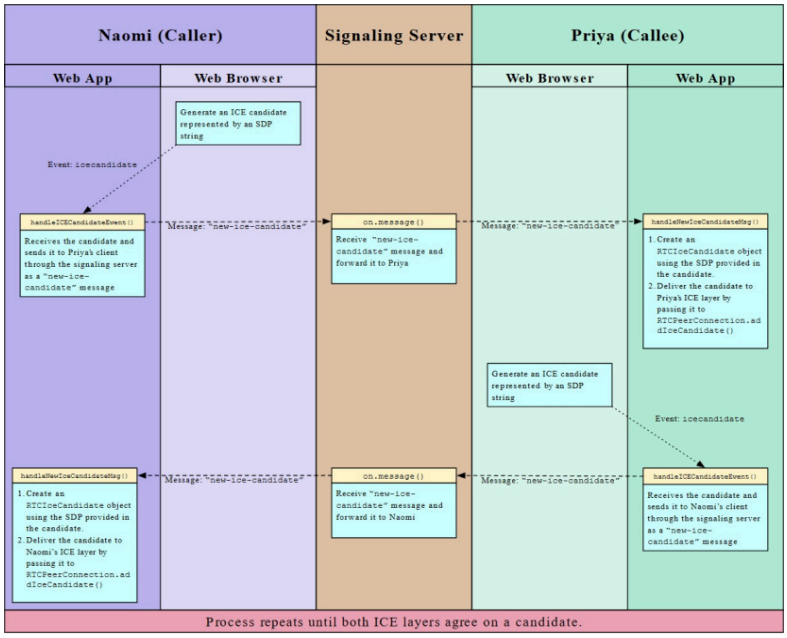
1. First user has dynamic local IP
2. He checks whether his router uses symmetric NAT or any other NAT
3. If symmetric NAT found, we will use TURN server
    a. Because symmetric NAT is too restrictive
    b. So, Peer to Peer is very challenging
    c. We use centralised server for that called TURN server
4. For any other NAT type, we use STUN server for knowing which publicly accessible IP we have. Generally, Entry of NAT table of Router.
5. Now we know our public IP on which anyone can access us
6. Prepare ICE candidate (collection of our all information and possible gateways for WebRTC)
    a. ICE is responsible for facilitating the connection between devices over the Internet
7. Prepare SDP(Session Description Protocol) offer (ICE candidate + media description and media constraints for a given user + other useful information)
8. SDP offer Send to Signalling Server (Connecting.)
9. Signalling Server Rings on Other User interface using WebSocket (Ringing.)
10. If Other User Decline Acknowledgement Sends to first user
11. If Other user Accept/Pick the Call
12. SDP offer of first user is received
13. Repeat All the process from 1 to 8 for this user
14. Signalling Server send SDP offer of second user to first user
15. Media Accessed from Users Devices
16. WebRTC Peer to Peer Connection Established on basis of SDP and ICE candidate
17. Peer to Peer Connection Established Now Enjoy the Call

Calling When using STUN(Peer-to-Peer)



Calling When using TURN server: (No peer-to-peer)





Protocols

Step	Protocol	Layer	Devices	Notes
Dynamic IP Assignment	DHCP	Application Layer	DHCP server / Router	
NAT Type Detection	STUN	Transport Layer	Peer	
Handling Symmetric NAT	TURN	Application Layer	Peer	Client-Server Connection
ICE candidate Preparation	ICE Framework	Application Layer	Peer	
SDP offer	Session Description Protocol (SDP)	Application Layer	Peer	
Initiating and Maintaining Call	Web Sockets	Application Layer	Peer, WebSocket Server	
Audio Transfer	Null	Physical Layer		Codecs: Opus
Connection Establishment	Packet Switching	Data Link Layer	Routers	
End-to-End Encryption	SRTP	Transport Layer	Peer	Uses AES

Comparison of Existing Apps

	Whatsapp	Instagram	Messenger	Telegram	Skype
Availability Checking	MQTT	MQTT	MQTT	MT Proto (proprietary)	SIP
Signalling	XMPP	WebSocket (wss)	SIP	MT Proto (proprietary)	SIP
Media Transport	SRTP (Secure)	RTP	RTP	MT Proto (proprietary)	SRTP (Secure)
NAT Traversal	STUN and TURN	STUN and TURN	STUN and TURN	STUN and TURN	STUN and TURN
Codec (Audio)	Opus	Opus	Opus	Opus	SILK
Codec (Video)	VP8	VP8 or H.264	VP8 or H.264	H.264	H.264

- **TURN(Traversal using relay around NAT)**

Why did we choose WebRTC?

- WebRTC is a framework of protocols built for developers to use while building calling services.
- In WebRTC we get:
    - RTP (Real-time Transport Protocol) for media transport
    - STUN/TURN for NAT Traversal
    - Opus for audio Codec
    - VP8,VP9, H.264 for video codec

So, We can combine WebRTC with any signalling server like WebSockets to implement Audio Calling System.

Tech Stack:

1. **UI/UX:**
    a. Tailwind CSS with custom CSS for styling
    b. Utilising components from third-party UI libraries
2. **Interactions and Frontend Logic:**
    a. JavaScript for interactions
    b. Browser APIs for audio or video input
    c. React.js for frontend logic
    d. Google Authentication integration
3. **Data:**
    a. PostgreSQL for data storage
4. **Backend (signalling server):**
    a. Node.js or Bun for backend logic (Peer.js)
    b. Web Sockets for signalling (Peer.js)
5. **Peer-to-Peer Communication:**
    a. WebRTC for peer-to-peer communication (Peer.js)
    b. Google STUN servers for connectivity

6. **Hosting:**
    a. Netlify for frontend hosting.
    b. Render for backend signalling server hosting.
    c. PostgreSQL on Supabase.

Demo and Implementation:

- Code: https://github.com/AwaisOem/calling-app-frontend
- Demo: https://calto-app.netlify.app/

Problems and FAQs:

- Why is Symmetric NAT not feasible for Peer-to-Peer connection?
    - blog link

.