Video Conferencing as a Web Application using WebRTC

Achyut Ajith Kumar
School of Engineering
Department of CSE
Dayananda Sagar University
Bangalore, India
achyut.ajithk@gmail.com

Arun Kumar
School of Engineering
Department of CSE
Dayananda Sagar University
Bangalore, India
arunkumar.aqm@gmail.com

Akhil T Sam
School of Engineering
Department of CSE
Dayananda Sagar University
Bangalore, India
fromakhil@gmail.com

Dr. Mouleeswaran SK
Associate professor
Department of CSE
Dayananda Sagar University
Bangalore, India
mouleeswaran-cse@dsu.edu.in

***Abstract*—People need to communicate with each other for different reasons like business, personal etc. An in-person meeting may not be possible depending on the situation and this means more money and logistics. The global pandemic has also caused an increase in demand for remote/long distance communication. We are implementing a video conferencing web application to tackle these problems.**

We plan on implementing a video conferencing application which lets users to join/host a video call between two or more participants on the same meeting webpage. The backend framework is built using WebRTC. WebRTC is a free, open framework that uses simple Javascript APIs to provide Real-Time Communications (RTC) capabilities to web browsers. WebRTC aims at providing high quality real time communication applications to your webpage. Connections from one browser to another can be made easily, which is dynamically possible through the real time web. This allows our web application to provide different features like text chat, file sharing, screen sharing, audio calls, video chat and more. The front end is designed using ReactJS. We use firebase as our signalling server.

***Index Terms*—WebRTC, React, Firebase, Video conferencing, file sharing, call, video, audio**

I. Introduction

Video conferencing is a technology which allows users to host or join a meeting without having to physically be in the same place. It is a tool that is used for different communication platforms. These platforms may provide features like screensharing, recording, text message etc. Owing to the increase in use of technology and the Covid-19 pandemic, video conferencing is transforming our means of communication with each other.

Video conferencing must be designed to be easy to learn and use, especially for individuals without much experience with technology. Users must be able to host/join a meeting, invite participants with ease and it must be doable from any media device. The technology should also be accessible directly from a browser without the need for any installation or setup. Security and privacy is important, especially to reduce risks due to humans.

WebRTC is the framework picked for this application and can provide a video conferencing web application that is secure, convenient and compatible with browsers for all device platforms.

II. Working of WebRTC

WebRTC is a protocol which allows us to do bi-directional, real-time, peer-to-peer, media exchange between 2 devices. Real-time means there is no lag. Peer-to-peer means 2 devices connected to each other directly. Media exchange is where 2 devices can send/receive video and audio to each other.

Each client has to go through these steps for successful WebRTC communication.

A. Signalling

Before 2 devices can start a P2P communication, they need to know about each other, this is done with the help of Signalling. Both devices agree upon a common central server through which they can talk and exchange information about each other. This server is called signalling server. There is no standard for connecting to Signalling server, so devices can use protocols like HTTP, REST, websockets, MQTT etc.. to connect to the server.

Devices exchange information like:
1. IPs and Ports that the Device is reachable on (candidates).
2. How many audio and video tracks the device wishes to send.
3. What audio and video codecs each device supports.
4. Values used for securing connection (certificate fingerprint).

B. Connecting

Once 2 devices know each others details, they try to find the best possible way to connect to each other using ICE protocol. ICE (Interactive Connectivity Establishment) is a protocol which is used to find the best ways to connect to a device called ICE candidate. ICE uses STUN or TURN

servers to find the best possible way to connect. STUN server is a simple server, which on request provides the public IP and port of the connecting device. TURN server is a Relay server, i.e it acts as a central server through which the data passes through. While connecting the 2 devices check if the ICE candidate is valid.

C. Securing

Once devices know how to connect to each other, they secure it using 2 protocols DTLS and SRTP. DTLS (Datagram Transport Layer Security) which is just TLS over UDP. The TLS protocol provides HTTPS security. SRTP (Secure Real-time Transport Protocol) is RTP with packets encrypted. Devices then do a handshake using DTLS protocol. They check if the certificates match the fingerprint passed in the SDP during signaling. For Audio/Video transmission devices use the same certificates to encrypt the data and send it via RTP.

D. Communicating

Two Devices can communicate 2 things between each other:
1: Media : Audio, Video.
2: Data Messages (simple string messages)

WebRTC specifies 2 different protocols for each of the above type. For Media, devices use RTP (Real-time Transport Protocol) encrypted with SRTP. For Data, devices use the SCTP (Stream Control Transmission Protocol)

III. Literature Survey

[1] In order to make the screen sharing across platforms, a scheme is based on the WebRTC technology under the Browser/Server framework. The system architecture is described in detail in the paper. The proposed WebRTC-based scheme brings a cross-platform, cross-device and multifunctional user experience when compared to other projects.

[2] This paper implements an application compatible with Mozilla firefox. The application is multipoint, meaning every user is connected to each other. The same video stream is displayed to all the users.

[3] This paper helps us understand the system architecture of WebRTC. Developers can build an application with real time video communication solution that is secure and does not require any installation or external plug-ins.

[4] This paper tries to implement live video streaming application using WebRTC as well as its limitations. Implementation shows that peer-to-peer streaming is currently possible for more complex algorithms and larger sets of clients. However, the experiments and implementation have also shown, that with the current limitations of WebRTC such an implementation is currently not practical.

[5] This article uses WebRTC which helped implement a secure and high data transmission application. It provides real time communication amongst users as peer to peer or peer to group connection. Anyone can host/join their own webpage which enables users with features such as text, video, voice chat, file sharing etc.

[6] The application presented in this paper had the potential to smoothly integrate itself into other projects. This means there is no need for any external plug-ins and significant cut down in costs .

[7] The architecture and technology of WebRTC can be well understood from this paper. WebRTC does not offer a signalling solution, so they used WebSocket as signalling protocol to send the required signalling information.

[8] This paper draws attention to use of APIs and how they can help build your application with WebRTC. It also gives us insight about how your application can run with fairly good latency even with limited bandwidth.

IV. The Technology of WebRTC

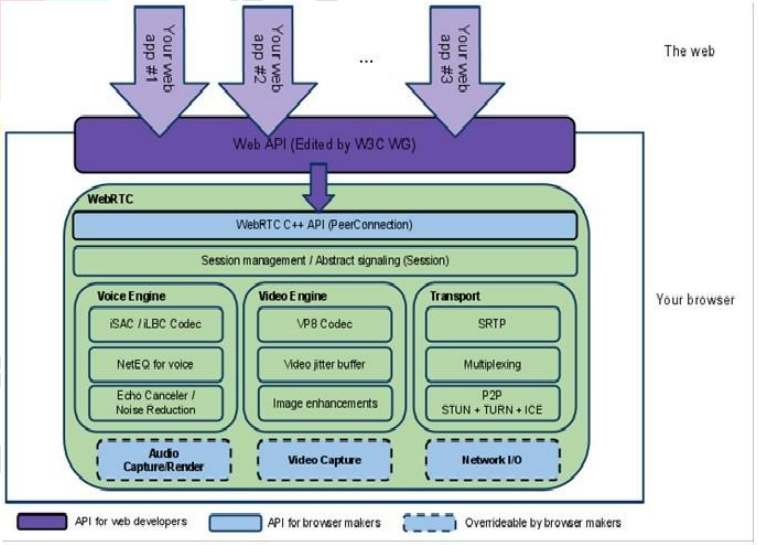

Fig. 1. The technology structure of WebRTC (Huaying Xue, 2015, p.1)

[1] helps us understand the architecture and technology of WebRTC. The architecture of WebRTC is shown in Figure 1. The API layer for Web developers, the API layer for browser developers, and the custom service layer for browser developers make up WebRTC. GetUserMedia API, PeerConnection API, and DataChannel API are the three basic APIs provided by the top API layer for Web developers. WebRTC is illustrated in Figure 2. Figure 2 shows two WebRTC-enabled

browsers, a signalling server, and a STUN/TURN server. It shows a basic WebRTC application architectural concept in which browsers serve as clients, the signalling server parses signal messages before the peer-to-peer connection is completed, and the STUN/TURN server punches holes.

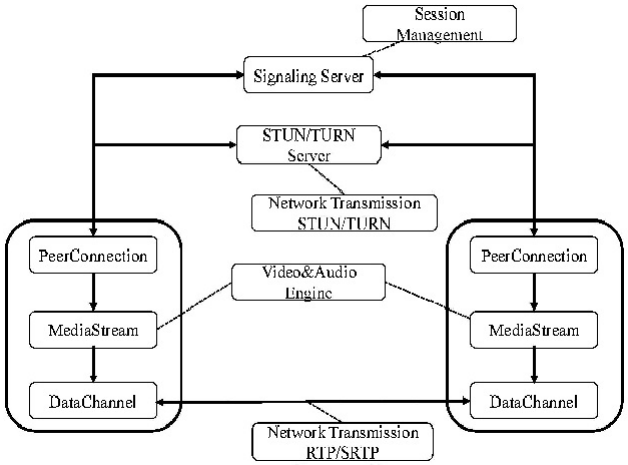

Fig. 2. The principle of the WebRTC technology (Huaying Xue, 2015, p.2)

V. ReactJs

React was used to create our front end. React makes creating interactive UIs a breeze. Create basic views for each state of your project, and React will update and render the appropriate components as your data changes. ReactJS is a JavaScript library for creating reusable user interface components that is declarative, efficient, and versatile. It's an open-source, component-based front-end library that's just responsible for the application's view layer.

VI. Firebase

A Firebase database server is used as the signalling server. A Firebase account was created and a new project was set up. In the code, we import the Firebase Javascript library and use the credentials provided to your Firebase account.

The idea to use Firebase as the signalling server was taken from [9]

The Firebase Real-Time Database is a database stored in the cloud. Data is saved in JSON format and synchronised in real time across all connected clients.

VII. Methodology

A. Connect users

For the two users to connect, the simplest option is that both the users visit the same website. This page can then identify each browser and connect both of them to a shared signaling server, using something like the WebSocket API.

B. Start signals/Signalling

WebRTC does not specify how signalling should be done. Signalling is any form of communication that helps the 2 browsers establish and control their WebRTC communication. May be done using:

- a combination of XHR and the Google AppEngine Channel API
- XHR polling
- Server-Sent Events
- WebSockets

C. Find Candidates/ICE Framework

The next step is for the two browsers to exchange information about their networks, and how they can be contacted. This process is commonly described as "finding candidates", and at the end each browser should be mapped to a directly accessible network interface and port. Each browser is likely to be sitting behind a router that may be using Network Address Translation (NAT) to connect the local network to the internet. Their routers may also impose firewall restrictions that block certain ports and incoming connections. Finding a way to connect through these types of routers is commonly known as NAT Traversal.

D. What is NAT?

NAT solves the problem of scarcity of IP addresses.The basic idea behind NAT is for the ISP to assign each home or business a single IP address (or at most, a small number of them) for Internet traffic.Within the customer network,every computer gets aunique IP address,which is used for routing intramural traffic.However,just before a packet exits the customer network and goes to the ISP, an address translation from the unique internal IP address to the shared public IP address takes place.

STUN (Session Traversal Utilities for NAT) helps in NAT traversal. A STUN server identifies how you can be contacted from the public internet and then returns this information in a useful form. There are a range of people that provide public STUN servers.

If the STUN server cannot find a way to connect to your browser from the public internet, you are left with no other option than to fall back to using a solution that relays your media, such as a Traversal Using Relay NAT (TURN) server. This effectively takes you back to a non peer-to-peer architecture, but in some cases, where you are inside a particularly strict private network, this may be your only option.

Within WebRTC, this whole process is usually bound into a single Interactive Connectivity Establishment (ICE) framework that handles connecting to a STUN server and

then falling back to a TURN server where required.

E. Negotiate Media Sessions

Now that both the browsers know how to talk to each other, they must also agree on the type and format of media (for example, audio and video) they will exchange including codec, resolution, bitrate, and so on. This is usually negotiated using an offer/answer based model, built upon the Session Description Protocol (SDP).

F. Start RTCPeerConnection streams

Once this has all been completed, the browsers can finally start streaming media to each other, either directly through their peer-to-peer connections or via any media relay gateway they have fallen back to using.

G. Peer to Peer Communication

Multiple computers come together and pool their resources to form a content distribution system. These computers are peers. There is no dedicated infrastructure like in a client-server architecture; no central point of control. P2P networks are self-scaling and faster.

VIII. Main function modules

A. Video chatting module

To perform video chatting, each client must use the GetUserMedia API to retrieve the local video and audio streams. The streams should then be added to the local PeerConnection object, and the incoming media streams should be attached to the remote PeerConnection object. As a result, we can have face-to-face conversations with our friends.

B. Data communication module

Between the two peers, a data channel must be introduced to the existing media channel. We create a new related object called 'DataChannel' and link it to the PeerConnection object using the DataChannel API. The data channel resembles the 1M WebSocket channel in appearance. The difference is that the latter requires the server's assistance, whilst the former does not. SCTP is the WebRTC protocol for direct transmission over the Data Channel.

C. File sharing module

For file sharing, the main idea is to fragment the whole file data, transmit the small packets via the Data Channel, and group them together when received. In this module, clients can work as both a sender and a receiver. First, the sender uploads the file to the application. The arrayBuffer(uploaded file) is converted into Base64 encoded string on the sender's side. Finally, we convert, the base64 string to a blob object, which can now be downloaded by the recipient.

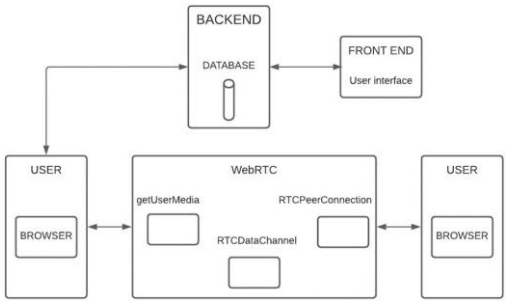

Fig. 3. Video communication using WebRTC

IX. Proposed Design

The backend contains the underlying working of the application and the database. This is accessed through the front end; which is the user interface. Users access the web application through a browser. Once they are on the webpage and connected to the same meeting; WebRTC uses its different APIs to enable real time communication capabilities.

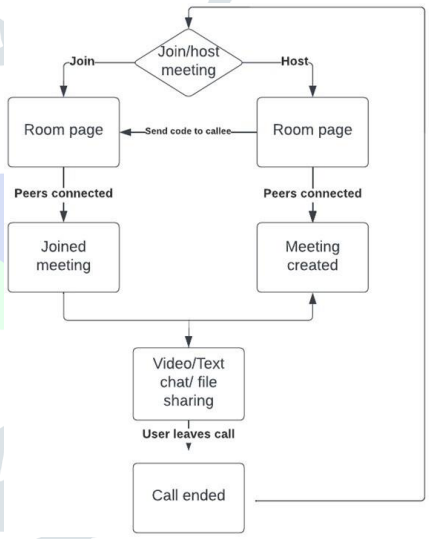

Fig. 4. Flowchart

The flow is fairly simple, users simply choose to join/host meeting on the landing page. The host sends the meeting id to the desired participant via other form of communication. Once the participant enters the correct meeting id, both users are moved from the Room page to the meeting page. Here, users can choose to video/audio chat, text chat or screenshare. Once a user leaves the call, the call ends.

X. Results and screenshots

A project has been successfully implemented, where two users can join/host a meeting. Here, users can join meeting

and can begin to chat. Users can choose to audio/video chat, text chat or file share.

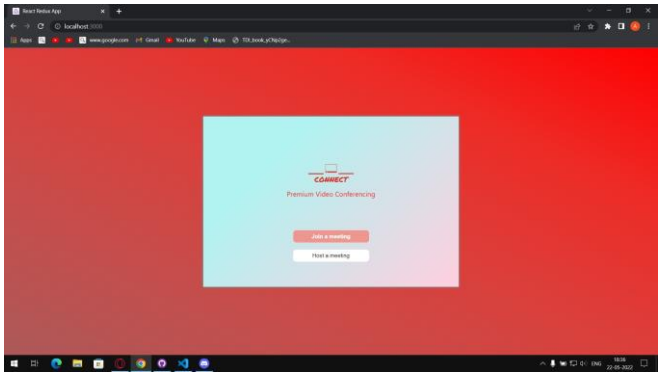

Fig. 5. Landing page

The landing page lets the user choose to join or host a meeting.

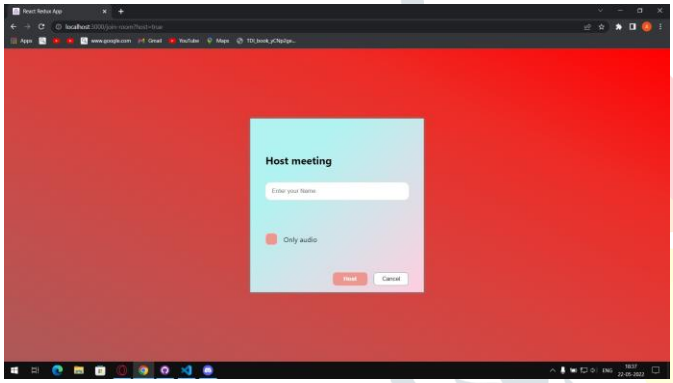

Fig. 6. Host meeting page

The host meeting page allows the user to host a meeting, by entering the name the user wants to be displayed in the meeting.

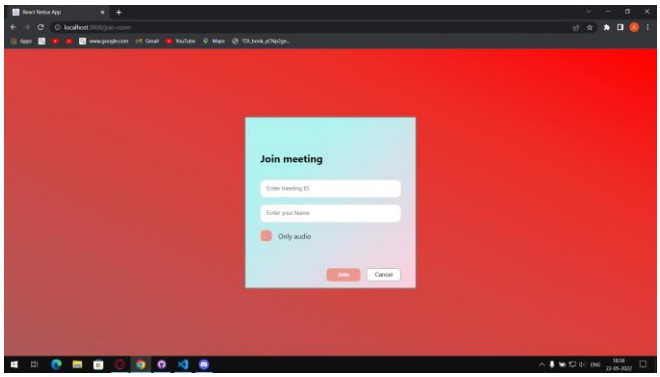

Fig. 7. Join meeting page

The join meeting page lets the user join a meeting by entering the meeting id provided by the host.

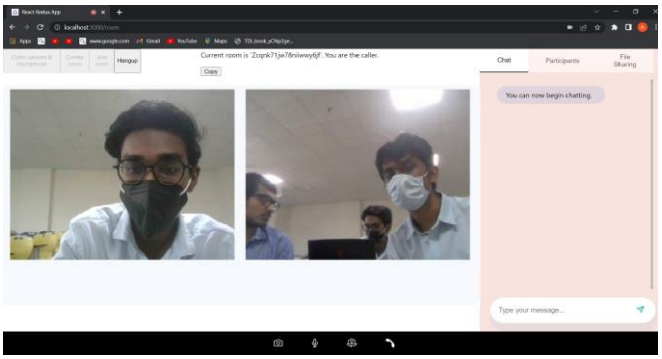

Fig. 8. Meeting page

The meeting page is where the call takes place, once both users have joined the meeting. Here, users can text chat, audio/video chat or file share.

CONCLUSION AND FUTURE SCOPE

This paper illustrates how WebRTC can be used to enable real time communication capabilities for a web application. A project has been implemented which provides a platform where two users can join/host a meeting on the same webpage. Users can text, audio/video chat and file share. One can further improve the application by adding features like screen sharing and a login page for extra security. Preferences management, background blurring, support for many participants is alsopossible.

REFERENCES

[1] Huaying Xue, Yuan Zhang(2016), “A WebRTC-Based Video Conferencing System with Screen Sharing”, IEEE Conference on Computer and Communication.

[2] George Suciu, Stefan Stefanescu, Cristain Beceanu, Marian Ceaparu(2020), “WebRTC role in real-time communication and video conferencing”, IEEE Global IoT Summit.

[3] EA Emmanuel, BD Dirting(2017), “A peer-to-peer architecture for real-time communication using Webrtc”, Journal of Multidisciplinary Engineering Science Studies.

[4] Florian Rhinow, Pablo Porto Veloso, Carlos Puyelo, Stephen Barrett, Eamonn O Nuallain(2014), “P2P live video streaming in WebRTC” , World Congress on Computer Applications and Information Systems (WCCAIS).

[5] Zinah Nayyef, Sarah Faris Amer(2019), “Peer to Peer Multimedia Real-Time Communication System based on WebRTC Technology”, International Journal for the History of Engineering and Technology.

[6] Phuc Truong(2021),”Video Conference Room Implementation with WebRTC and React”

[7] Cui Jian,Zhuying Lin(2016) ”Research and Implementation of WebRTC Signaling via WebSocket-based for Real-time Multimedia Communications”

[8] Kushtrim Pacaj, Kujtim Hyseni, Donika Sfishta(2020) ”Peer to Peer Audio and Video Communication using WebRTC”

[9] Rob Manson(2013), ”Getting Started with WebRTC”, Birmingham: Packt.

[10] David Marcus,2017, Insanely Simple WebRTC Video Chat Using Firebase (With Codepen Demo), Australian Museum, accessed 10 September 2021, (https://websitebeaver.com/insanely-simple-webrtc-video-chat-using-firebase-with-codepen-demo)