International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/CSEIT217647

Peer To Peer Real-Time Communication Using WebRTC

Rahul Kumar Mohata*[1], Amita Goel[2], Vasudha Bahl[3], Nidhi Sengar[4]
*[1]Student, Department of Information Technology, Maharaja Agrasen Institute of Technology, Delhi, India
[2]Professor, Department of Information Technology, Maharaja Agrasen Institute of Technology, Delhi, India
[3]Assistant Professor, Department of Information Technology, Maharaja Agrasen Institute of Technology, Delhi, India
[4]Assistant Professor, Department of Information Technology, Maharaja Agrasen Institute of Technology, Delhi, India

Article Info
Volume 7, Issue 6
Page Number: 178-183

Publication Issue :
November-December-2021

Article History
Accepted : 01 Dec 2021
Published : 07 Dec 2021

ABSTRACT

The covid-19 pandemic has led to things happening virtually. Students are attending their classes in online mode. More than 50 percent of the working population is working from home. Online meetings have become necessary part of everyone's life. With the existing platforms, users need to setup or install packages on their systems to run the application which sometimes becomes confusing for first timers or non-technical people. This paper proposes to build a full-fledged feature rich web-based video conferencing application using WebRTC technology. WebRTC is used to enable real time audio and video communication from a web browser without the need of installing software or plugins so that users can focus on their work rather than worrying about how to use a video conferencing platform.

Keywords : Real-time Communication, WebRTC, Peer to Peer, Video conferencing, Web Socket, API, Client, Server

I. INTRODUCTION

With the internet revolution, real time communication has gained a lot of importance around the world. Real-time Communication is a mode of communication in which users share information live without any delay. It is a peer-to-peer based communication in which a bi-directional channel is established between a sender and receiver.

RTC has indeed been difficult and sophisticated. It used to be challenging and time-consuming to integrate RTC technology on the web and looked like a major hurdle until Google launched open-source project WebRTC in 2011.With WebRTC, real-time communication on the web became simpler and implementable. WebRTC is an open standard that allows for real-time, plugin-free video, audio, and data transmission. It gave web apps the power to handle real time communication through a set of open-source APIs available in every web browser. In this paper, we present a web-based video conferencing system that uses WebRTC to allow users to interact in real time without requiring any difficult

setup or installations across a range of devices and operating systems.

II. METHODS AND MATERIAL

A. WebRTC Architecture

WebRTC gives engineers the capacity to compose full-fledged real-time mixed media applications on the web. Its goal is to assemble a solid RTC stage that works across numerous internet browsers, across various stages. To establish real-time communication between two browsers, we need to setup a connection between them. WebRTC is used for sending the audio and video streams between peers but one thing that WebRTC doesn't provide is Signalling. Signalling is a process of establishing a peer-to-peer connection between remote peers.

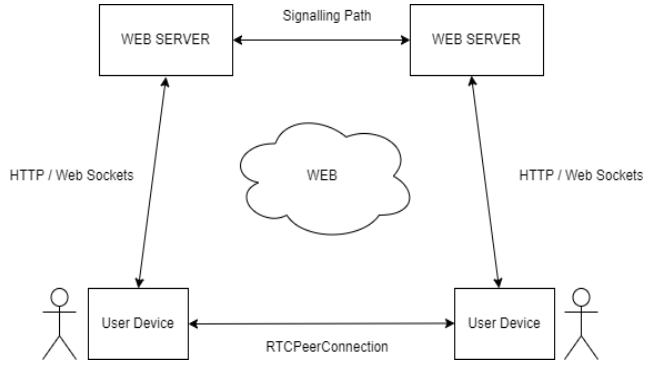

Fig. 1. WebRTC Architecture

For this purpose, we generally use Web Sockets which creates a full duplex connection between a web browser and a server. The communication is done through emitting and listening to events. This way, browser doesn't have to make a request to the server to fetch a response. That is why web sockets are generally very fast and thus are used for signalling.

For sending and receiving data to other clients, each client must know the IP address of other clients. As clients are mostly behind Network Address Translators, it is hard to find their own IP address. For this we setup two type of servers, a Session Traversal Utilities for NAT (STUN) server and a Traversal Using Relays around NAT (TURN) server. Generally, a client can find their IP address using a STUN server, but if the STUN server fails then a TURN server can be used.

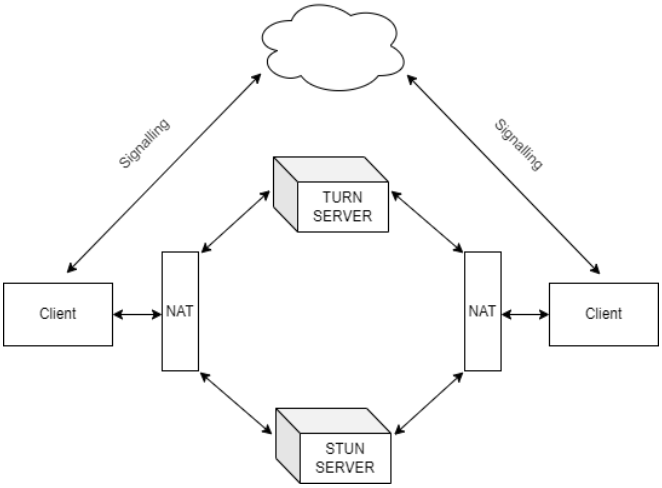

Fig. 2. WebRTC Architecture with NAT, STUN and TURN server

B. WebRTC API

Every browser has WebRTC APIs built in which can be used by application developers to develop a feature rich web-based video conferencing system. The primary level APIs are:

1) MediaStream: MediaStream API is used to access the client's video and audio tracks. If the user's device has several cameras or microphones, it controls the input device selection.

2) RTCPeerConnection: RTCPeerConnection API is used to establish connection between peers. It also systematically manages the sending and receiving of media streams between peers.

3) RTCDataChannel: RTCDataChannel API is somewhat similar to MediaStream API. It is used to construct a bi-directional channel between two peers. This channel can then be used to send and receive extra data between peers.

C. Tools and Technologies Used

1) HTML: Hyper Text Markup Language, or HTML is a markup language used to create the structure of a web page.
2) CSS: Cascading Style Sheets is a language that is used to add styling to web pages. It is used to make websites more attractive.
3) React.js: React is an open-source frontend JavaScript library used to develop user interfaces using reusable components. It is used to develop single page applications.
4) Redux: Redux is an open-source JavaScript library used for state management in applications. It is highly used and compatible with React.js.
5) Node.js: Node.js is a JavaScript runtime which enables execution of JavaScript outside of a browser. With Node.js, we can use JavaScript to develop a web server.
6) Express.js: Express.js is an open source node.js framework used to develop backend servers using JavaScript. Express is the most commonly used node.js framework and quite popular.
7) Peer.js: Peer.js is an open-source JavaScript library which provides a wrapper around browser's WebRTC logic hiding most of the implementation details and provides an easy-to-use API for developers to access WebRTC.
8) Socket.io: Socket.io is an open-source JavaScript library that provides an easy-to-use API to establish Web Socket connection between a client and server.
9) Google Firebase: Firebase is an all-in-one application development service which provides APIs for all the primary functionalities like authentication, database, push notifications etc.
10) Jest: Jest is a open source JavaScript library that is generally used for code testing. It is the most commonly used JavaScript testing library and can be used on frontend as well as backend.
11) GitHub: GitHub is a source code management and version control system using Git under the hood. We can create public and private repositories on GitHub and collaborate on a project in a team.
12) Netlify: Netlify is a Platform as a Service (PaaS) cloud service for hosting web applications and managing deployments.

D. Working

The application is primarily divided in two layers, frontend and backend. The backend server is built using Express.js. The frontend of the app is built using React.js. Each and every user can host meetings by entering meeting details like a title and date & time. When user clicks the “Create” button, a new meeting is created with a unique id and password and stored in database. This id and password is to be used for joining the meeting. When a user enters meeting id and password and clicks “Join” button, id and password are validated and on successful validation user is allowed to enter the meeting room.

Fig. 3. Meeting instance in database

After user joins a room, a “user-join” event is emitted via web sockets. This event is listened at the server and is then broadcasted to other socket connections. When the existing meeting peers gets the “user-join” event, they send a connection request to the new peer along with their video and audio stream. This is handled using Peer.js, a wrapper around browser’s WebRTC implementation. When the new peer, receives a connection request along with media stream of existing peers, it accepts the request and sends its media stream in the response. This media stream is read by the existing peers and in this way a real-time video and audio communication is established. All the other functionalities like chat messaging, screen sharing, leave meeting are also

handled with event-based communication using web sockets.

III.RESULTS AND DISCUSSION

The application is deployed in cloud using Netlify PaaS with Continuous Integration & Continuous Delivery configured. Existing users can login in to the app and new users can register and then login into the application. User Interface of the application is as shown below:

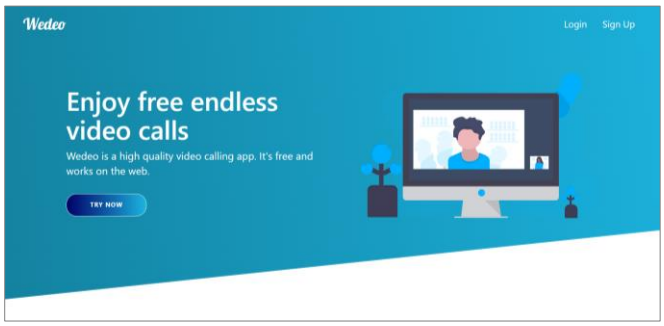

Fig. 4. Landing View

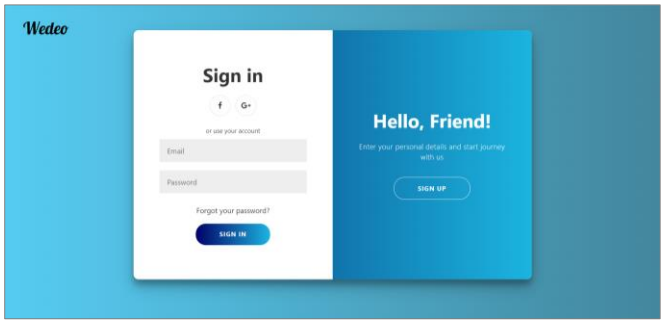

Fig. 5. Login View

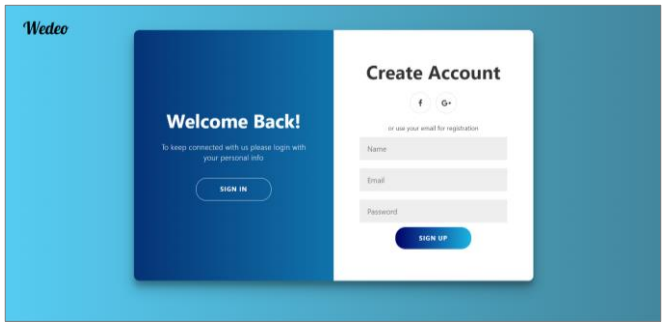

Fig. 6. Register View

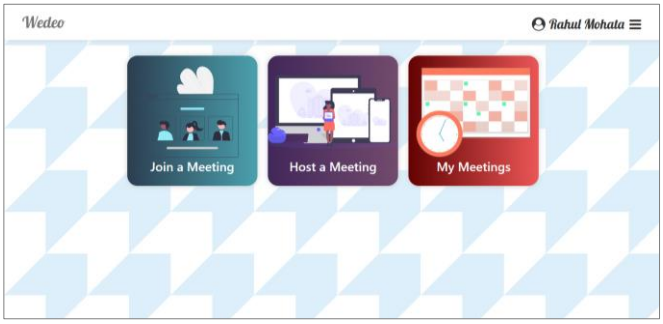

Fig. 7. Dashboard View

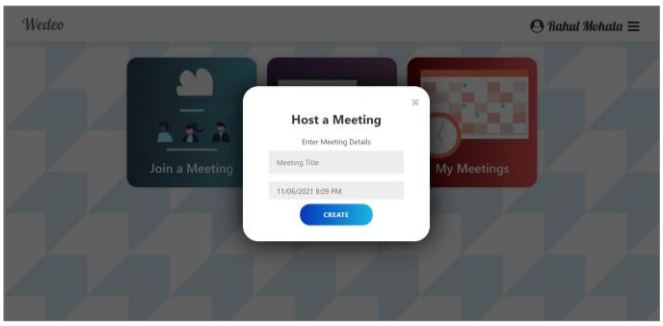

Fig. 8. Host Meeting Modal

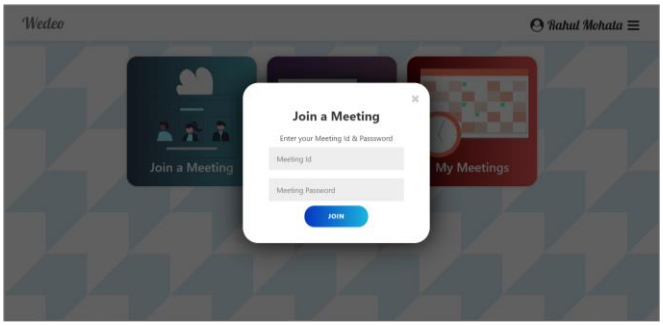

Fig. 9. Join Meeting Modal

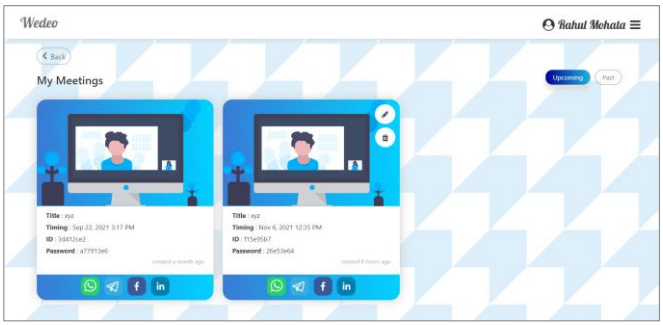

Fig. 10. My Meetings Page

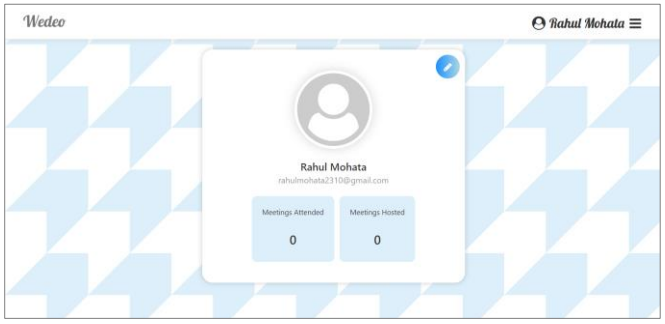

Fig. 11. Profile Page

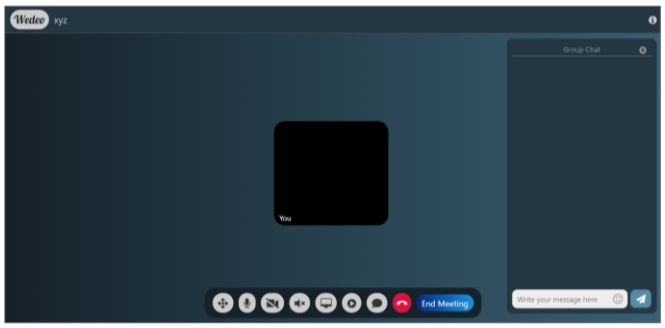

Fig. 12. Meeting Room Page with Chat box

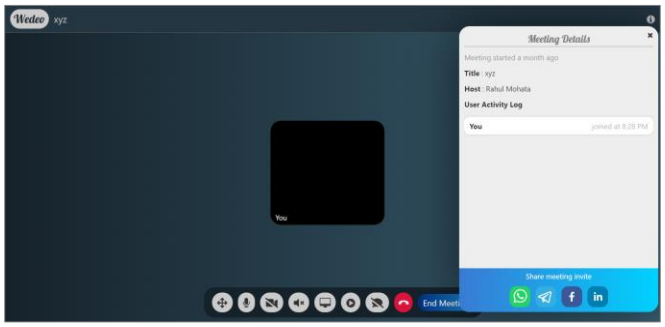

Fig. 13. Meeting Room Page with Meeting Details Modal

IV.CONCLUSION

With the power of WebRTC, we were able to develop a web based real-time audio and video communication system. WebRTC provides a standard set of open- source APIs to develop a high quality and robust video conferencing platform. It is the most widely used real-time communication technology. This application would be of great help to students, teachers, business teams and other users. Features like Screen Sharing will help users to share their screen and collaborate with other meeting participants, Screen Recording will help users to record meetings for future reference, Group Chat will allow user to chat with meeting participants using real-time messaging, Private Meetings to allow user to enter the meeting room if the meeting id and password are valid. This application is designed in such a way that it can be scaled easily and new features can be added as per requirements.

V. REFERENCES

[1]. Xue, Huaying & Zhang, Yuan. (2016). A WebRTC-based video conferencing system with screen sharing. 485-489. 10.1109/CompComm.2016.7924748.

[2]. C. Chiang, Y. Chen, P. Tsai and S. Yuan, "A Video Conferencing System Based on WebRTC for Seniors," 2014 International Conference on Trustworthy Systems and their Applications, 2014, pp. 51-56, doi: 10.1109/TSA.2014.17.

[3]. K. I. Zinnah Apu, N. Mahmud, F. Hasan and S. H. Sagar, "P2P video conferencing system based on WebRTC," 2017 International Conference on Electrical, Computer and Communication Engineering (ECCE), 2017, pp. 557-561, doi: 10.1109/ECACE.2017.7912968.

[4]. Jose Dominic, Joel Mani Joseph, Vishal Thomas, Surekha Mariam Varghese. (2021). Creating an Integrated Online Education Platform with Bandwidth Optimized P2P Video Conferencing. International Research Journal of Engineering and Technology (IRJET)

[5]. Nayyef, Zinah & Amer, Sarah & Hussain, Zena. (2019). Peer to Peer Multimedia Real-Time Communication System based on WebRTC Technology. International Journal for the History of Engineering & Technology. 2.9. 125-130.

[6]. Simon Holm, Alexander Lööf in their thesis “The design and architecture of a WebRTC application”, Malmo University, 2019

[7]. https://webrtc.org/

[8]. https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API
[9]. https://www.w3.org/TR/webrtc/
[10]. https://peerjs.com
[11]. https://nodejs.org/
[12]. https://reactjs.org/
[13]. https://socket.io/
[14]. https://firebase.google.com/docs

Cite this article as :

Rahul Kumar Mohata, Amita Goel, Vasudha Bahl, Nidhi Sengar, "Peer To Peer Real-Time Communication Using WebRTC", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 7 Issue 6, pp. 178-183, November-December 2021. Available at doi : https://doi.org/10.32628/CSEIT217647
Journal URL : https://ijsrcseit.com/CSEIT217647