International Journal of Engineering Applied Sciences and Technology, 2022
Vol. 7, Issue 4, ISSN No. 2455-2143, Pages 276-280
Published Online August 2022 in IJEAST (http://www.ijeast.com)

VIDEO/AUDIO CONFERENCING USING WEBRTC

Ms.PreetiRathee, Devesh Bhatla, Sameer Khan, Sudeep Chowdhary, Vaibhav Diwan
Information Technology Dept.
Maharaja Surajmal Institute of Technology,
New Delhi, India

Abstract: **Due to the ongoing COVID-19 situation, the need for video-conferencing tools skyrocketed. From student's classes to high stakes business meetings, video call platforms are now commonplace for virtually everyone. Web Real-Time Communication allows the modern web browsers to communicate in real time using mesh architecture in a secured and efficient way with no need for any proprietary plug-ins. In this paper we proposed a video-calling platform made using webRTC with additional functionalities like video/audio privacy controls, inbuilt chatting system, screen sharing and interactive whiteboard.**

I. INTRODUCTION

Video-conferencing allows people present at different locations to communicate in real time by transmitting audio and video information[1,2]. After the release of video-conferencing tools, we got the ability to see and hear events which are occurring far away, from the comfort of our homes. Also, the use of video conferencing skyrocketed because of the COVID-19 pandemic and the resulting lockdown. A Gartner HR Survey conducted in 2020 reports that 86% of companies conduct employee interviews over video call[3]. Zoom conducted a survey recently which shows that 88% of the respondents believe that video conferencing software helps them to get higher education[4].
There are two types of distributed networking namely centralized and decentralized networking. A centralized network architecture is built around a single server that handles all the major processing. The client-server model is a type of centralized network architecture. In contrast to the centralized client-server networking model, there is also a decentralized networking model. A peer-to-peer mesh service is a decentralized platform whereby many individuals interact directly with each other, without intermediation or support of a global centralized server or authority.
WebRTC (Web Real Time Communication) is an open-source project that adds real-time peer-to-peer communication capabilities to an application via browser-to-browser communication for video chat, voice calling and even sharing files among users.[5] WebRTC allows us to exchange media through the web without any required plug-in or framework.

We have expanded the scope of WebRTC from simple peer-to-peer connection to mesh connection. This allows more than 2 users to be in a single video conference.

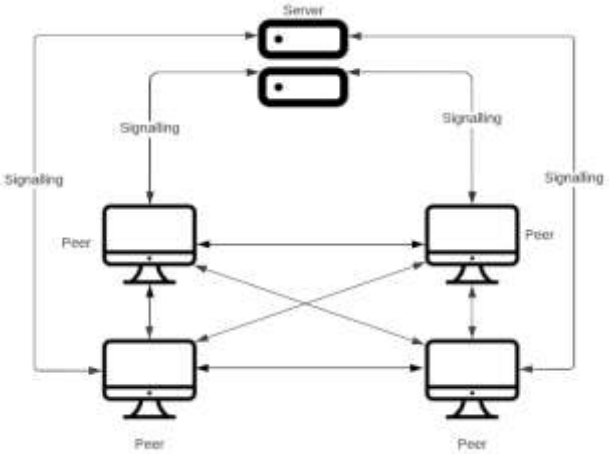

Figure 1.1 WebRTC signalling[17]

The above diagram represents the WebRTC signalling process.
A connection is established between peers through a negotiation process called signalling [14]. The application requires a signalling server to establish the connection. The server is only responsible for the exchange of information of the peers. Once the peers know the connection details the connection is made directly between them. The server only plays the role of the exchange of information and does not play any role in transmitting the actual media(audio/video) between the peers. The information includes the media details (audio/video) of peers.

International Journal of Engineering Applied Sciences and Technology, 2022
Vol. 7, Issue 4, ISSN No. 2455-2143, Pages 276-280
Published Online August 2022 in IJEAST (http://www.ijeast.com)

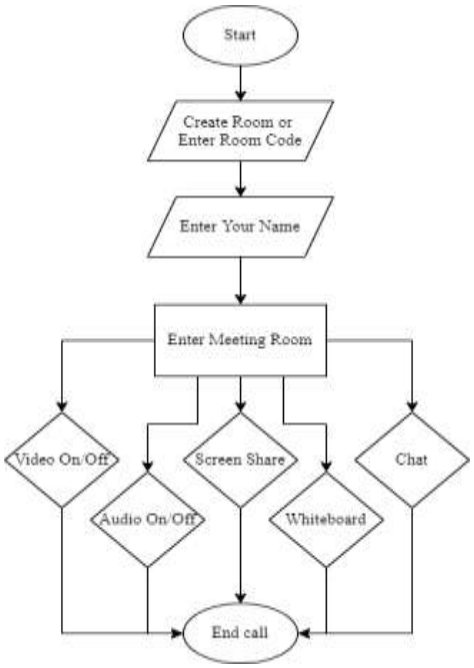

Figure 1.2: Application flow

The above figure portraits the overall application flow. When the user starts the application, the first page consists of two options – create a room or join a room. When the user creates a room or enters the room joining code, he is then asked to enter his name and then redirected to the meeting room page. On the meeting room page, the user can perform multiple actions like turning his video/audio on or off, screen sharing, using whiteboard, chatting in group. He can also end the call at any point of time.

II. MESH ARCHITECTURE WEBRTC

The caller creates an offer using the Session Description Protocol (SDP)[6] and sends it to the other peer. The SDP contains media information such as audio/video.
The callee responds to that offer with an answer message that also contains the SDP.
After the exchange of SDP, they still won't be able to connect as they are not aware of each other's IP address, NAT, and firewalls. Interactive Connectivity Establishment (ICE) is used to solve this issue. ICE gathers the available network connections, also known as ICE Candidates and uses STUN/TURN server to send the information.

API's:
1. MediaStream: This includes the media tracks like video/audio[7]. **Media Devies. get User Media()** method retrieves a MediaStream, like video from webcam.
2. RTCPeerConnection: This allows the peer-to-peer communication between users[8]. Streams that are accessed by **Media Devices. get User Media()** are added to this component.
3. RTC Data Channel: This allows bi-directional peer-to-peer transfer of arbitrary data[9].

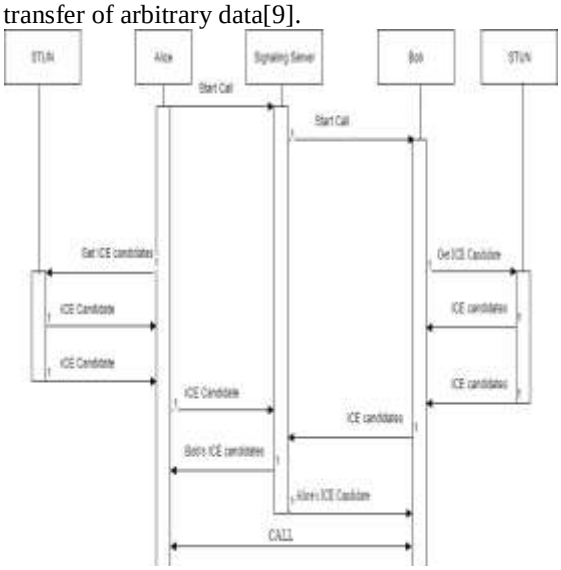

Figure 2.1: Sequence diagram of connection establishment [11]

The above diagram represents an example of connection establishment where Alice and Bob are the two participants.
ICE (Interactive Connectivity Establishment) is a peer to peer network method used for sending and receiving media information over the net [10]. ICE collects available network connections, called ICE candidates, and uses protocols such as STUN (Session Traversal Utilities for NAT) and TURN (Traversal Using Relays around NAT) for NAT and firewall.
Steps followed by ICE to handle the connections[11]:
1. Initially it tries to connect peers via UDP.
2. If the direct method of UDP fails it tries TCP.
3. If both UDP and TCP connection fails, ICE tries to use STUN server with UDP to connect.
4. If STUN server also fails, ICE uses a TURN server, which is same as STUN server but have some extra relay functionalities that traverse symmetric NATs.
Many a times the traditional functionality of one-to-one video conferencing provided by WebRTC applications does the job, but what if we want to connect several users together inside one call. The answer to this question is hidden in the depths of Mesh networking which can be applied to peer-to-peer technologies like WebRTC.[18]

International Journal of Engineering Applied Sciences and Technology, 2022
Vol. 7, Issue 4, ISSN No. 2455-2143, Pages 276-280
Published Online August 2022 in IJEAST (http://www.ijeast.com)

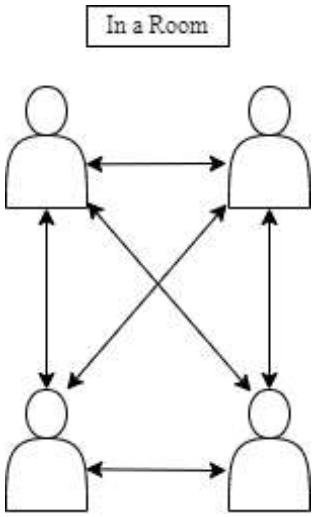

Figure 2.1: Four users connected in mesh network in a single room

The above figure represents the scenario where four participants are present in a single room where each user is connected with every other user to send/receive the video stream which enables them to perform video calling.
To connect multiple users, each one would begin by connecting to the signalling server. They would then request calls to all the users they want to communicate with. This would setup multiple WebRTC connections allowing everyone to communicate with each other at the same time. So now, each participant would send their stream directly to every other participant without having to go through a media server.

III. SECURITY/PRIVACY FEATURES

In WebRTC Media streams are encrypted using Secure Real-time Transport Protocol (SRTP) [12].
The Application has features to turn of Video/Audio of the peers. These features are useful from the privacy perspective. The audio/video tracks can be disabled by **track. Enabled = false**; [15]. The type of track (video/audio) can be found by **track. kind**; [16] . It can be enabled again when the Mic/Video button is pressed.

IV. SCREEN SHARE

The screen share feature may be used to display the contents of your entire screen or a particular window to another peer. It works by capturing your screen and sending the screen capture data to the other peers in the same stream.
Media Devices. get Display Media() method is used for getting the data on the screen [13]. This method responds by presenting the user with a dialog box from which the screen to be shared can be selected. The selected screen replaces the video feed given by **Media Devices. getuser Media().** The replacement is done by the **replace Track (track)** [13] method.

As soon as a screen share is started, a stop share screen window appears at the bottom that is integrated into the browser itself. When the stop
screen share button is clicked, the display stops giving the data and the video stream is replaced by the stream of the camera.

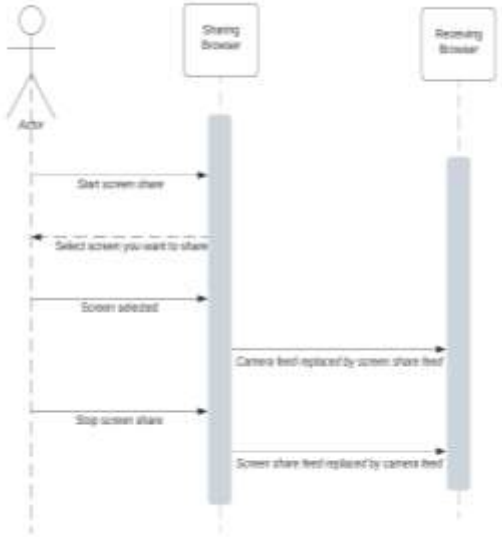

Figure 4.1: Sequence diagram of screen share

The above picture represents the screen share feature. All clients in the same room can see the screen shared by the single individual.

V. CHAT

The chat functionality is implemented usingSocket.IO due to the poor performance of WebRTC data channel. The total throughput seems to be very heavy on CPU usage and data channels and is largely single threaded [19].
Text messages along with the timestamp and the username can be transferred to the peers present in the current room. The **socket. emit()** API emits the message data, username and timestamp to the server. The **socket.on()** API listens to that data and populates it in the textbox present on the screen.

International Journal of Engineering Applied Sciences and Technology, 2022
Vol. 7, Issue 4, ISSN No. 2455-2143, Pages 276-280
Published Online August 2022 in IJEAST (http://www.ijeast.com)

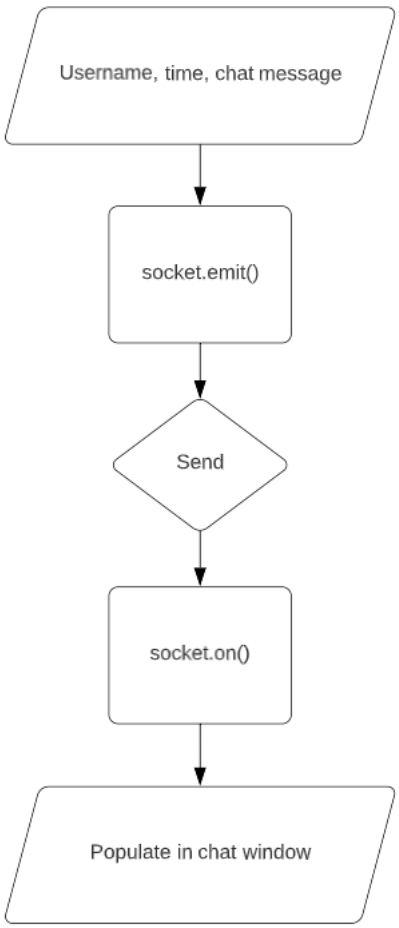

Figure 5.1: Sequence diagram of chat

The above diagram represents the flow of chat feature. When user clicks send with a message in the chat box, **socket. emit()** event triggers and send the username, time and message to the deployed server. Then **socket.on()** even triggers and populates the text box of all the users present in the room.

VI. INTERACTIVE WHITEBOARD

With interactive whiteboard, multiple users can draw simultaneously on a single canvas. Whiteboard feature is implemented using Socket.IO.

Socket. emit() API can send the coordinates of the new x, y and previous x, y positions along with the colour and pen size.

Socket. on() API receives the old x, y and new x, y positions along with the colour and pen size and draws it onto the canvas.

It ensures that the drawing on the screen is real-time, and all the users present in the room can draw simultaneously. There are 10 colours to choose from in the whiteboard along with an eraser and a delete button that clears the entire whiteboard in 1 click.

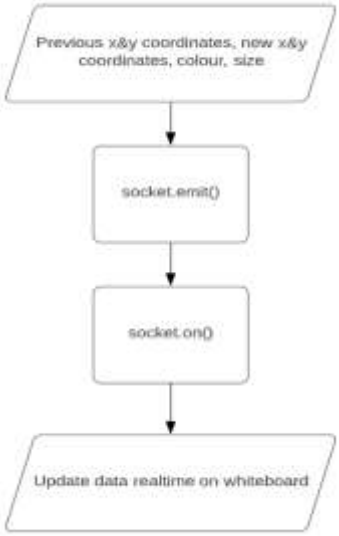

Figure 6.1: Sequence diagram of interactive whiteboard

The above diagram represents the flow of interactive whiteboard feature. When a user draws any figure on the canvas, the **socket.emit()** even is triggered and at the same time **socket.on()** event is triggered. It ensures that the real-time content on the canvas is being displayed.

VII. CONCLUSION

WebRTC technology enabled the implementation of secure and easy transmission between users as peer to peer in real time communication, therefore anyone can create their own room. It includes features such as real time communication environment, screen sharing or audio/video conferencing. This allows users to connect one to one with other users. WebRTC allows us to create a such a powerful feature allowing every user to connect with other via audio/video conferencing or screen sharing by using simple React API and JavaScript API. STUN server is provided free by Google and can be used in the application.

Table 7.1: Client-Side Performance (Chrome browser CPU usage) [20]

Numberof users	**P2P/Mesh**	**SFU**
2	4%	5%
3	10%	8%
4	22%	9.5%
5	34%	18%
6	47%	25%

Table 7.2: Server-Side Performance (CPU usage) [20]

Number of users	P2P/Mesh	SFU
2	0.1%	2%
3	0.1%	13%
4	0.1%	24%
5	0.1%	32%
6	0.1%	41%

The SFU server has lost its real-time performance since it had 6 clients whereas the signalling server remained constant at 0.1% [20].

In mesh as the number of clients increased the CPU utilization of the clients increased significantly as compared to SFU[20].

VIII. REFERENCES

[1]. Andersson, Russell L., Tsuhan Chen, and Barin G. Haskell, (19 Mar. 1996), "Video conference system and method of providing parallax correction and a sense of presence." U.S. Patent No. 5,500,671..

[2]. Egido, Carmen, (1988), "Video conferencing as a technology to support group work: a review of its failures." Proceedings of the 1988 ACM conference on Computer-supported cooperative work. ACM.

[3]. M. Baker, (April 2020),"Gartner hr survey shows 86 of organizations are conducting virtual interviews to hire candidates during coronavirus pandemic,". [Online]. Available: https://gtnr.it/3yyAvMf.

[4]. S. Ann Earon, "The value of video communications in education." [Online]. Available: https://bit.ly/30yxi2v.

[5]. L. L. Fernandez, M. P. Diaz, M. R. Benitez, F. J. Lopez, J. A. Santos, J.A.,(2013), "Catalysing the success of WebRTC for the provision of advanced multimedia real-time communication services," Intelligence in Next Generation Networks (ICIN), 2013 17th International Conference on, pp.23,30,.

[6]. M. Handley et al., (2006), RFC 4566: SDP: Session Description Protocol. IETF RFC.

[7]. "MediaStream," MDN Web Docs. [Online]. Available: https://mzl.la/3se9ts8.

[8]. "RTCPeerConnection," MDN Web Docs. [Online]. Available: https://mzl.la/3DZdkeV.

[9]. "RTCDataChannel," MDN Web Docs. [Online]. Available: https://mzl.la/3q8wCtH.

[10]. J. Rosenberg, (2010), RFC 5245: Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. IETF RFC.

[11]. Borja Nebbal, "How ICE Handles The Connection". [Online]. Available at: https://bit.ly/3e2fuQp.

[12]. M. Baugher et al., (2004). RFC 3711: SRTP: Secure Real-time Transport Protocol. IETF RFC.

[13]. "Using the Screen Capture API" MDN Web Docs. [Online]. Available at: https://mzl.la/3yBoupp.

[14]. S. Loreto and S. P. Romano (2014). WebRTC Introduction. Real-Time Communication with WebRTC. O'Reilly Media, Inc.

[15]. "MediaStreamTrack.enabled" MDN Web Docs. [Online]. Available at: https://mzl.la/3IVQ5pS.

[16]. "MediaStreamTrack.kind" MDN Web Docs. [Online]. Available at: https://mzl.la/3E4IDoH.

[17]. "Signalling" meidialooks. [Online]. Available at: https://bit.ly/3pbg8RY.

[18]. Dan Ristic (2015). Learning WebRTC. Introduction to mesh networking. Packt Publishing.

[19]. Rasmus Eskola, Jukka K. Nurminen,(2015) "Performance evaluation of WebRTC data channels".

[20]. "WebRTC Performance Comparison". [Online]. Available: https://bit.ly/38K3eoQ.