TECHNIUM

*Technium Vol. 4, No.10 pp.83-96 (2022)*
*ISSN: 2668-778X*
*www.techniumscience.com*

# Proposed a WebRTC Data Communication Using Effective Signalling Protocol

**Sanabil A. Mahmood**

Department of Scienc, College of Computer Science and Mathematics, University of Mosul, Mosul. Iraq

sanabil_2000@uomosul.edu.iq

**Abstract**. Web Real-Time Communication (WebRTC) is designed to permit the co-occurrence of audio, video, and data communications. However, an effective signalling mechanism that setup, establish and end communication between peers has not been specified in WebRTC. This paper designs and implements an effective WebRTC signalling mechanism for data communication compared with the common existing applications and current reviews. Moreover, it focuses on the limitations of multi-browsers communication and data. As long as no elaboration specifies how the signalling mechanism can be used in WebRTC because it does not understand the concept of sessions but rather that of streams, which create and consumes media flows. In addition, this research will elaborate on the related work and analyse the communication quality of data communications, such as data, audio, and video. Keywords: Web Real-time Communication (WebRTC); Signalling Protocols.

**Keywords**. Web Real-time Communication (WebRTC); Signalling Protocols

## 1. Introduction

W3C is used in browser API, and IETF, for the wire protocol, created a brand-new standard called WebRTC. WebRTC is made to allow for simultaneous voice, video, and data conversations [1]. In May 2011, the W e b R T C began. At that time, Google announced an opening Source project to enable browser-based, real-time peer-to-peer communication [2]. A collection of Java Script APIs, libraries, and standards make up WebRTC [3]. Alternatively, [4] proclaimed that, as opposed to a Customer/server protocol, Web RTC is a peer -to -peer (P2P) protocol. Java Script A PIs is utilized directly in WebRTC to facilitate interactive connections between browsers that employ different types of data. In addition, [5] highlighted that WebRTC had several advantages, such as the lack of plug-ins, the ease of usage, no licensing, and the excellent quality of the RTC applications. As a result, [6] confirmed that WebRTC had been used by more than one billion endpoints and devices. Besides, by 2018, it has anticipated that 4.7 billion devices will be capable of supporting WebRTC [7]. WebRTC is built based on the three APIs, such as:

Technium Vol. 4, No.10 pp.83-96 (2022)
ISSN: 2668-778X
www.techniumscience.com

**a) Get.UserMedia API**

This API allows a browser to use nearby devices like a camera and a microphone while representing synchronized streams as audio and video input and output [8]. WebRTC eliminates the need for Adobe Flash in order to use media devices and the plugin requirement thanks to its API. Additionally, this API makes audio and video available for use as HTML elements in all types of web applications [9]. Figure (1) demonstrates the Media Stream structure supporting WebRTC. WebRTC is built based on the three APIs as follows:

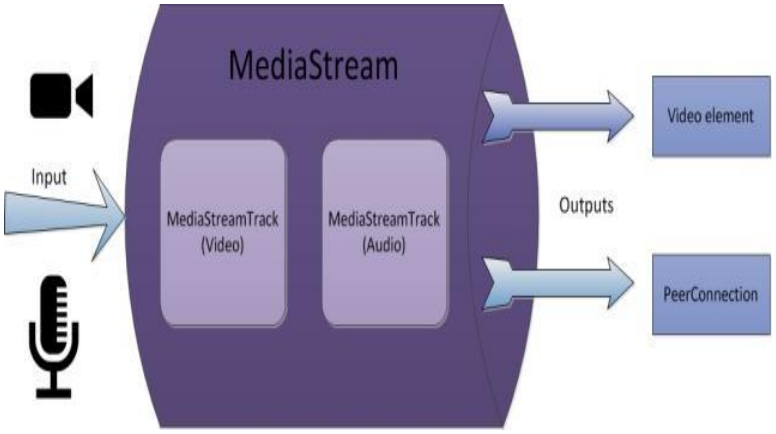


Figure 1:The structure of WebRTC MediaStream [4].

**b) RTC.PeerConnection API**

Direct communication and the establishment of voice or video calls between browsers are made possible by the RTCPeerConnection API. The get.User.Media() method gathers all of the browser's internal workings, including media streams, and allows a browser to reach nearby devices like a camera and microphone to send data to another party [10]. Also, handling all exchange signalling messages requires particular JavaScript methods. Besides, [11] demonstrated that when "PeerConnection" initiates a successful call in a new environment, it facilitates ICE (Interactive Connectivity Establishment) methods and ensures a high degree.

**c) RTC.DataChannel API**

Browsers can send data connections thanks to the RTCDataChannel API. Additionally, it offers a two-way data transmission that allows two browsers to exchange random data. While avoiding the use of servers and intermediaries, each "RTCDataChannel" can lower the service price and provide network utilization such are more adaptable low-latency data exchanges [12]. It is divided into the following types:

1. Delivering trusted or untrusted messages.
2. Delivering messages that are in or out of order.

Protocol for Transmission Control provides the same reliable and orderly transmission (TCP). The User Datagram Protocol (UDP) is comparable to inconsistent distribution and out of order, though. Applications for this API could include games, remote desktop programs, real-time file transfer programs, and text chat, to name just a few. Figure (2) shows the WebRTC architecture.

*Technium Vol. 4, No.10 pp.83-96 (2022)*
*ISSN: 2668-778X*
*www.techniumscience.com*

A WebRTC call setup has been built to concentrate on controlling the media plan, leaving as much of the signalling plan activities up to the application as is practical, as shown in [13]. the justification is: (a) allowing application developers to choose from a list of available protocols (such as SI P or XM PP) or to design their protocol, (b) preventing duplication and maintaining maximum backward compliance with current technology [8], (c) The WebRTC working group of the IETF did not want to limit it to a solution that might not be adequate for all of its needs, (d) allowing designers to create or change signalling protocols, (e) not to be bound by a specific technology [2], (f) for an original use case, and (g) using the specified use case as a guide [14]. The first problem encountered when integrating WebRTC is this one. However, it has not been determined by WebRTC [15].

## 2. Literature Review

WebRTC needs some sort of mechanism for signalling or assistance from protocols to establish communication between various users or devices [16]. But, to evaluate WebRTC and control the communication architecture, the final API protocol and signalling method have not yet been agreed upon by the IETF and W3C. A signalling mechanism is not a component of WebRTC even though it uses the "RTC.Peer.Connection" API to transmit data streaming across peers[17]. As a result, it is undeniable that WebRTC does not standardize the signalling between browsers and servers. Additionally, it appears that the client server architecture is not a workable solution for the Web RTC [7].

It has been clarified in [18], that signalling is required by clients and WebRTC apps to establish calls and communicate many types of information, including the following:

1. Session management Communications are initiated or terminated via messages.
2. Report Errors are used to send a report message when an error occurs.
3. Network configuration is grounded on reality and provides the peer with an IP address and port.
4. Media Capabilities include the bandwidth, resolutions, and codecs for audio and video that are compatible with the desired browser.

WebRTC has been developed on a variety of platforms, including simple WebRTC, easy RTC, and webRTC.io, to support video chat. However, some of them have limitations, some of them charge a fee, and some of them use their infrastructure to manage signalling, services, or easily accessible information. In addition, a number of fixes were proposed, and WebRTC utilized many protocols to obtain the signalling method detailed below:

### 2. 1. Protocol for Socket.io Signalling

A JavaScript transport the protocol is used to facilitate Web browsers and servers can communicate in real-time in both directions [19]. Since its inception, Socket.io has used Node.js as its primary backend [20]. The open-source Nore.js server platform was developed by Google Chrome to offer a potent JavaScript engine. As a result of its foundation in an event-driven paradigm and non-blocking input/output, which can manage high throughput and performance, it is also a good platform for creating web applications [21]. It provides an HTTP server as a result, but it is neither an application server like Tomcat nor a web server like Apache HTTP. While providing a wide range of services, Node.js emerges as one of the most popular options on the web for bi-directional tests. For instance, ease of learning, freedom in building applications and active community [22]. It has been explained in [23] because it enables programmers to use WebSockets and identify numerous synchronized communication protocols controlled by the client's browser, an idea for WebSockets that uses XHR is called

socket.io., Flash, and JavaScript Object Notation (JSON). On the other hand, it uses Adobe Flash Socket, AJAX long polling and JSON Polling. Additionally, [19] stated that socket.io offers simple server and client library for supporting real-time bi-directional communication; also it has the concept of rooms that enables peer-to-peer connection, as three WebSocket connections can occur from three various users [2].

### 2. 2. XML Http Request Signalling Protocol

Using XHR, the Google's WebRTC designed a simple application as highlighted in [24]. It was developed and tested for WebRTC and is an open-source program by the name of AppRTC. A new video conference room called AppRTC has been introduced. It automatically opens for the user and may then be shared with another user using Mozilla Firefox, Google Chrome, or Opera by supplying the designated URL. The AppRTC was developed to identify its browser and network connectivity constraints. So, it has several shortcomings:

1. Concerning the signalling protocol, it used Google Engine Channel API to open a channel between two peers for video chat, and also used Asynchronous JavaScript And XML (AJAX) (push communication) to obtain WebRTC clients. [17] confirmed that the canonical AppRTC ("apprtc.appspot.com") used XHR protocol to handle signalling which leads to high bandwidth consumption.
2. By default, peer-to-peer communication ends after 30 minutes.
3. Because the browser uses a 640x480 resolution by default, the video is of poor quality.

The Program Engine app gives the token to Client A, who then establishes a socket and listens to the server-created channel to communicate with Client B.

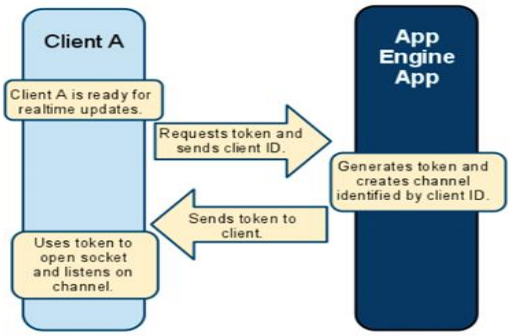


**Figure 2: Google Channel API to establish a channel.**

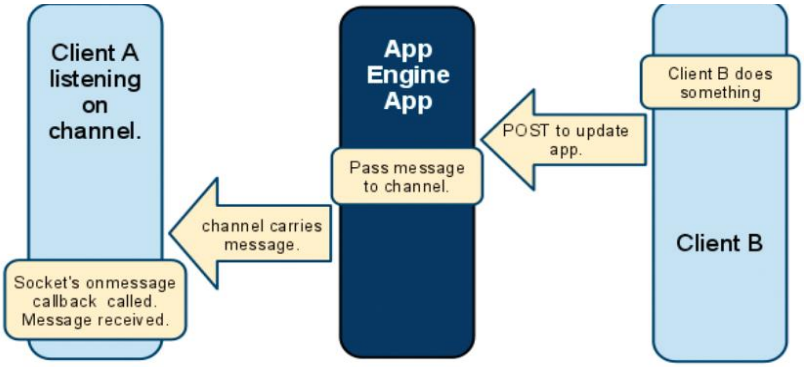


**Figure 3. Google channel API for sending a message.**

*Technium Vol. 4, No.10 pp.83-96 (2022)*
*ISSN: 2668-778X*
*www.techniumscience.com*

**2.3. Protocol for Extensible Messaging and Presence**

XM PP is open source technology for messaging, multi-party chat, and voice and video calls in real-time communication. However, it is a slow text-based protocol that leads to massive traffic through the Internet [15][25], indicating that XMPP utilises data transport between two entities on the network. Moreover, [26] expounded that the XMPP connection is based on XML streams, which consist of unidirectional elements. The XMPP limitations were published in different RFCs and revised versions confirming that XMPP is limited to the only TCP connection for XML sent from client-to-server [27]. Furthermore, [28] displayed that the XMPP criteria as follows:

1. It is used for data transport, but not for signalling.
2. It uses a client/server method but not a peer-to-peer method.
3. It needs asynchronously transports.
4. For media support, it uses XML streaming technology for data exchange instead of being extensible to other media types such as telephony and video

2.4.. WebSocket Protocol

A bi-directional communication session between a user's browser and a server can be opened with the aid of the WS protocol (Castillo et al., 2014). Instead of connecting to another peer, it opens a pipe to a server; a client sends messages to the server, which then forwards them to the participants (Emmanuel and Dirting, 2017). WS provides many benefits, including a low rate of data loss and high user concurrency (Zhang and Shen, 2013). Additionally, Grahl, 2015) revealed that WS, as long as it employs the web server as a proxy for connections to clients, uses the same communication port as HTTP (80/443) and is also an initial connection and handshaking between a web client and a web server (Melhus, 2015). WS leaves both directions of communication in the session open. As a result, users can send and receive a lot of messages (Sredojev et al., 2015). In addition, (Fette and Melnikov, 2011) WS is rumored to provide a tunnel and leave it open among peers. It performs better than AJAX/Polling in terms of lower latency and increased user data bandwidth Karadogan, 2013). On the other hand, (Cola and Valean, 2014) WS's inefficiency with all browsers was explained. Additionally, it cannot guarantee scalability and dependability when a high number of peers are using the WebRTC application or when entering and exiting the network (Laik and Lee, 2015). Different developers attempted to use WS to propose or develop a signalling mechanism for WebRTC as addressed below:

A communication service prototype based RE-presentation State Transfer (REST) API with SIP User Agent over WebSocket has been designed and implemented in (Ambra et al., 2013). The signalling of this prototype should be supported by a middle component (REST service) to exchange messages and establish a media channel. However, the communication is delayed 5 seconds and done with only two REST browsers. By the same token, REST relies on HTTP polling mechanism to push notification from the server to the client (Margret, 2017). Moreover, (Singh et al., 2013) analysed WebRTC video call performance utilizing the Node.js framework and WebSocket protocol (for signalling), TURN servers, etc. Both mesh and star topologies were used for this evaluation. The calls established between the three players in each topology, on the other hand, use a false video sequence as opposed to a live camera. Additionally, different switches were used for the mesh, and an MCU device was used for the star. Additionally, in this test, all calls are forced to stream the video among participants through the TURN servers. By contrast, Dutton, 2013) stated that TURN servers are being used to relay data between endpoints as shown in Figure (6). Similarly, (Emmanuel and Dirting, 2017)

demonstrated that using the TURN server for simultaneous calls can force more overhead on the bandwidth, and it is not wholly free. It is also necessary to highlight that the WebRTC communication is using a TURN server depending on traffic between peers, which suffers regression of media quality and latency.

It has been depicted in (Lozano, 2013), that the designed and evaluated WebRTC application is based on mesh topology for chat and video calling between two peers. This application use the WebSocket protocol and the Node.js platform to handle the signalling messages. Nevertheless, the signalling mechanism has not been described; in addition, it was not able to establish communication for more than three participants with the actual device resources. Another experiment was done in (Rasool and Mukhtar, 2013) that developed an algorithm for switching between WebSocket protocols to ensure efficient battery consumption for portable devices. This algorithm is applied to Qur'an e-learning service by broadcasting from one-to-fifteen participants as unidirectional (star) broadcasting. But, if a device source goes down, then no more peers can be supported, and all peers will go down. Also, the authors have not described how setting up the signalling to establish and end the broadcasting.

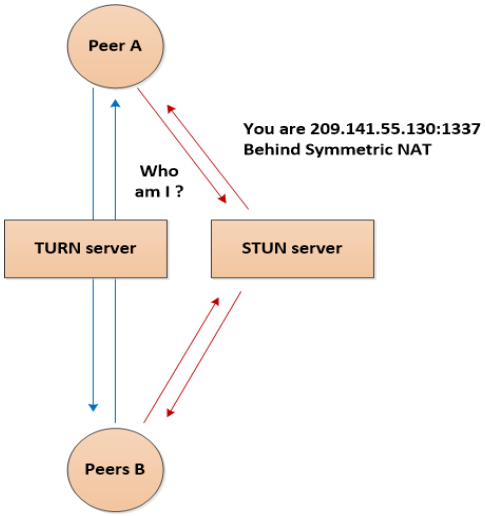


**Figure 4: TURN architecture between peers**

It has been demonstrated in (Sørlie, 2017) that the WebRTC chat application using WebSocket as a signalling server based on the node.js platform was designed and implemented. Although this application applied to two peers using adapter.js to support Google Chrome, users should connect a web server sequentially to communicate with each other. In addition, because a server cannot recognize a client's messages or know where to send them, there is a problem between clients and a web server. In a similar vein, the author acknowledged that the setup and signalling implementation were flawed. Another trial employing the Firefox and Chrome web browsers for video conferences used in e-Health over LAN network proved successful. The signalling was built based on WebSocket using node.js and the socket.io library to establish a communication between two peers (Emmanuel et al., 2017).

### 2. 5. Session Initiation Protocol

It has been illustrated in [29] that no signalling protocol (such as SIP) will be recommended to give developers more flexibility for innovation in web applications. However,

SIP is more complex than other protocols like XMPP. In addition, Various developers attempted to make video calls using WebRTC and SIP, but SIP still required software installation on such servers [8]. Moreover, [30] emphasized that conventional SIP clients do not yet support the protocols that WebRTC requires. Additionally, employing a new type of integrated communication environment, the coupling of WebRTC features with the SIP platform needs to be improved to support multimedia sessions. Aside from that, creating a SIP client is more time- and money-consuming than using the current real-time communication APIs in an application [31].

## 3. Common Existing WebRTC applications

[32] created a serveries-WebRTC project to decouple the "signalling server" and provide email and file transfer between peers without a web server. However, it often faces failure when transferring a large file. Similarly, [33] proposed a multi-video conference utilizing the XMPP server and lib-jingle to establish a WebRTC P2P connection (plugin). However, without installing lib-jingle, the user was unable to access the system or receive signalling. This test was conducted using two different browsers. Another program was developed for WebRTC video conferencing to allow communication between two clients utilizing the Node.js platform. In addition, [34] implemented a centralised system for instant messaging, video and audio conferences, and file sharing. Nevertheless, the video implementation is limited to two participants. What is more, [35] created a WebRTC-based video chat system for senior citizens. The system used an email to send the chatting request rather than signalling. It was also applied to one-to-one video chatting between two users. Not only that but also [21] developed a WebRTC video conference to be implemented on eLearning platform using the Node.js platform, socket.io APIs for signalling and mesh topology. Instead, this test was performed between two browsers. Besides, the author clarified that it seeks an improvement to solve an echo problem with noise cancellation.

A new approach to online learning called (web-based interactive virtual classroom), which must be accessed through HTTPS, was described. All functions cannot be worked out in real connection and latency occurred in some cases. This system is specified for video conferencing (unidirectional) as screen sharing between 10 users. The video was being shared as a one-to-one strategy. As an example, from A to B, B to C, and C to A [36]. Equally, [37] illustrated that a network architecture named Ufo.js offering a channel that enables two browsers to establish communication of file transfers was implemented. Ufo.js was run via the Node.js platform by using an external server to hold and manage a connection to each peer; this connection is used to direct all the signalling among peers.

[5] addressed how WebRTC created browser-to-browser video communication using appear.in SDK. But, the details of signalling were outside the scope, and a communication delay was higher than 1 second. In the same way, [38] attempted to use appear.in SDK to offer video conversations among up to eight users. However, the signalling was outside the scope while using appear.in SDK. [39] presented a test-bed to enable WebRTC real-time multimedia sessions using the Node.js platform as a signalling server with the assistance of the NS-3 simulator. By contrast, this test-bed was executed between two clients.

## 4. Analysis, Implementation, and Methodology

### 4.1 Methodology

In this implementation, the Heroku cloud (HTTPS) was used as a signalling server and the JSFiddle platform as a web server to handle the signalling. Additionally, the task manager,

*Technium Vol. 4, No.10 pp.83-96 (2022)*
*ISSN: 2668-778X*
*www.techniumscience.com*

JavaScript, and the Wireshark bandwidth analyser were used to assess CPU performance. Additionally, cameras, and microphones were provided through access point NetCommWireless. Additionally, Firefox (Mozilla) and Google Chrome were used on the client side. Additionally, ten PCs with various CPU core counts (CPU i5, & 8 GB RAM) are connected to the Internet through Ethernet and wirelessly at various locations as indicated below:

- Ethernet network connecting two computers to the Internet.
- A wireless network connected five computers to the Internet.

**4.2 Implementation**

The implementation of WebRTC video conferencing between several peers over (Ethernet & Wireless) of Internet and 4G networks in a mesh topology was done in a test-bed lab. This accomplishment makes use of a signalling method that is divided into two pieces and is based on the Socket.io API and Multi Connection library:

1) Setup the main platform (browser)

2) Utilised and linked the signalling mechanism to set up, initiate and terminate contact.

The creation of "room-id," which by default takes the form of a combination of numbers and/or letters and is used as a channel name or label that should be controlled by the initiator was essential. When starting and entering a room, the room-id is a highly important factor. It is used to ensure that relevant messages are exchanged with relevant participants and are inaccessible to other users. To issue a participation request and enter the same room, all users must have the same "room-id." If not, they'll join another room or create a new one. This implementation requires different participants to use "room-id," which is specified by a single initiator. Additionally, numerous rooms can be entered but with various "room-ids.".

A new socket must be made whenever the initiator creates a new room. A new socket can be used to obtain a variety of items, including an ICE gathered by the initiator and other participants and an SDP-offer. Alternatively, each peer should send their information over the default channel, create, and exchange "After initiating a new peer connection, they are expected to take on the role of the "offeror" for any new participants. SDP-offer/answer" The expectation is that each person who enters the room will take on the role of the "answerer.".

**4.2.1. Setup a Browser Web Page**

Numerous capabilities, including the ability to mute audio and video and use full screen, are available on the test's main web page. Initially, two buttons were intended to be used to open or enter the room by peers. In addition, a constructor's initialization in the MultiConnetion library was utilized to:

- Designate a session as a video conference.
- Configure SDP video streaming in a bi-directional manner
- Add a prompt event handler (On-click) for the buttons mentioned.
- Link socket.io API will be utilized as a signalling system to manage participant communication. The code was first tested by opening two comparable LAN taps, which allowed two peers to speak with one another. In addition, it sets the video width to 40% and the border radius to 15 pixels, as shown in the figure, to provide greater clearance (1). After determining the room-id, an initiator will offer audio and video "MediaStream" when the room is opened. By requesting authorization to use the peer's camera and microphone and executing the

"navigator.getUserMedia" method to capture the peer's screen, a "MediaStream" may be acquired. Following the granting of authorization, a camera will begin streaming, at which point the implementation will be available for joining or conversing. Everyone involved must use "getUserMedia" and distribute their cameras and microphones.

To exit the room, a peer must refresh or quit the JSFiddle platform's browser web page. This can halt the streaming of their camera and microphone without interfering with the other participants' communications.

#### 4.2.2. Signalling Channel Using Socket.io Mechanism

This signalling was applied based on the URLs of the MultiConnection library and the Socket.io API. A new video conferencing session has been initialized and set up using the MultiConnection library. Additionally, it now includes an "onstream" function for both local and remote media streams, giving each one a distinct stream id and an "event" object to pass through it. More significantly, it was discovered after analysing the Socket.io API that the Node.js server automatically provides it to retrieve the socket.io.js file from the client. The best real-time communication method for each client is also dependent on a variety of strategies, including Adobe Flash Socket, AJAX long polling, and JSON polling. For instance, socket.io can provide many alternative connection methods if a browser is unable to produce JSON. Using socket.io causes dealing with both the client and server files at once, and it can choose whether the connection is established using AJAX long polling, Flash, or WebSocket. Similar APIs are offered by socket.io for both server-side and client-side components.

### 4.3 Analysis

This application utilizes several browsers, including Chrome and Firefox, to provide video conferencing via the Internet and 4G. The data transfer is not impacted in this execution if one of the components fails because there is always a backup available. The analysis is illustrated as follows:

#### 4.3.1 Socket.io Bi-directional Signalling Mechanism

This signalling mechanism has been individually examined between two and ten users based on the signalling delay for two concepts: the first was based on the signalling delay to get ready, and the second relies on sending a request and receiving a response from peers. This analysis was based on the network analysis inspecting elements of Google Chrome and Firefox at the actual communication. As a result, it takes a minimum of 119 milliseconds (ms) and a maximum of 185 (ms) to prepare. To send a request and obtain a response, it takes a minimum of 280 (ms) and a maximum of 690 (ms). According to the computed mean, it takes 171 milliseconds to get ready and 453 milliseconds to transmit and receive a request. All participants can set up, establish, and end a conversation at the same time using the Socket.io signalling system. In actuality, there was a little difference in the variation of delay when using Chrome or Firefox. Additionally, the Opera browser is not supported by the signalling system. On the other hand, CPU load and bandwidth usage had an impact on the audio and video quality. A considerable delay was also discovered utilizing the socket.io signalling technique.

#### 4.3.3 Quality of Video Conferencing

Individual experiments with two to 5 peers found that the quality of voice and video gradually improved over the Internet. As a result, the audio and video between the two peers were of the highest quality. When the third peer entered the room, the video and audio qualities

were excellent as well. However, as soon as the fourth person entered the room, the video's quality changed and fluctuated, growing and decreasing with a lag. It was also noted that one peer was entering and exiting with a crystal-clear voice but a frozen visual for one minute. When the fifth person entered the room, the same quality was also made apparent. On the other hand, when the sixth, seventh, and more peers entered the room, the majority of the connected peers had poor audio and video quality, including unclear audio and frozen video. Not only that, but it was also unable to permit more peers than five to enter the room, as more peers ruined communication. Nearly invisible or shown as a frozen image in respect to the peer who was utilizing the internet. Therefore, employing the socket.io technique for many-to-many communication is effective up to a maximum of three peers.

**Table (1), compares the audio and video quality amongst seven peers using various networks.**

| Sequence | Number of peer | Quality of audio | Quality of Video |
|---|---|---|---|
| 1. | 1–2 | Excellent | Excellent |
| 2. | 3- 4 | Acceptable | Unacceptable |
| 3. | 4- more | very bad | very bad |

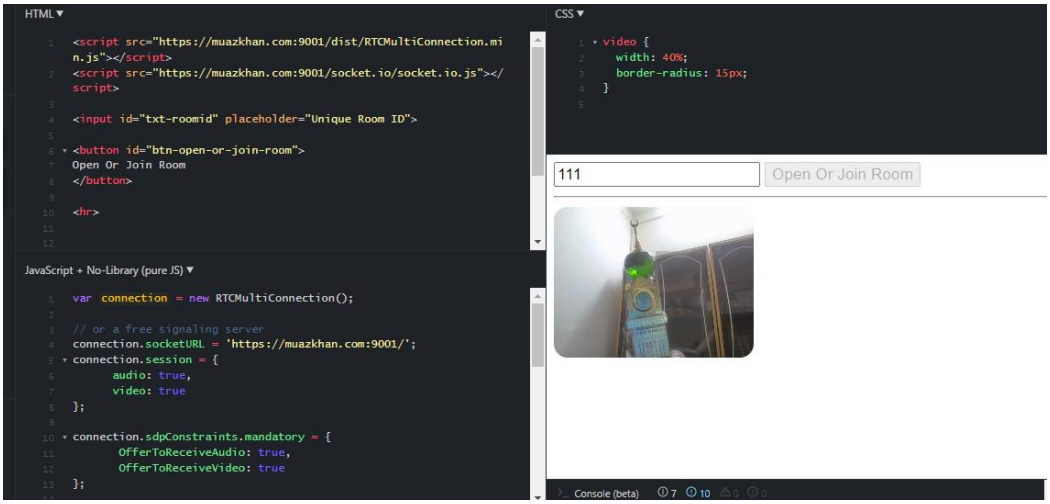


**Figure5 :communication of audio and Video from peer A.**

Technium Vol. 4, No.10 pp.83-96 (2022)
ISSN: 2668-778X
www.techniumscience.com

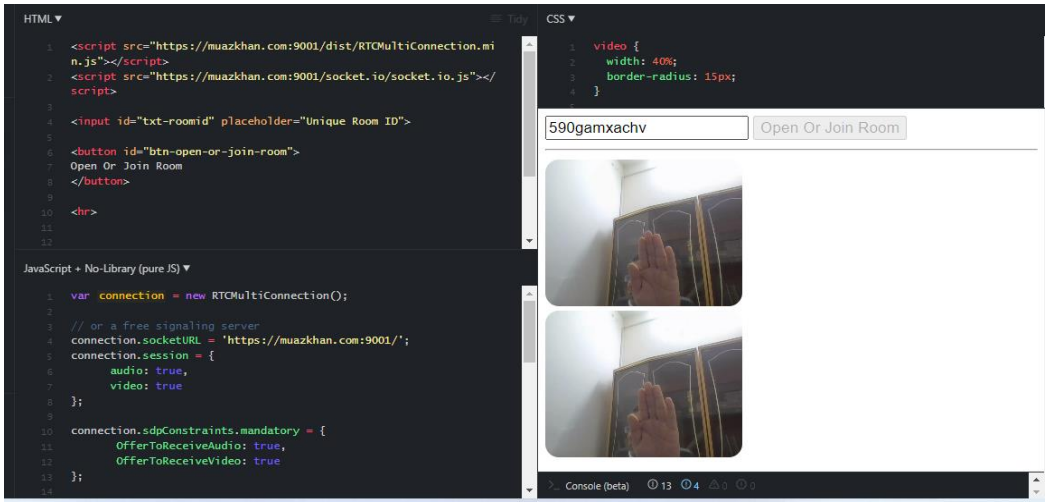


**Figure 6: Communication of audio and Video between two peers.**

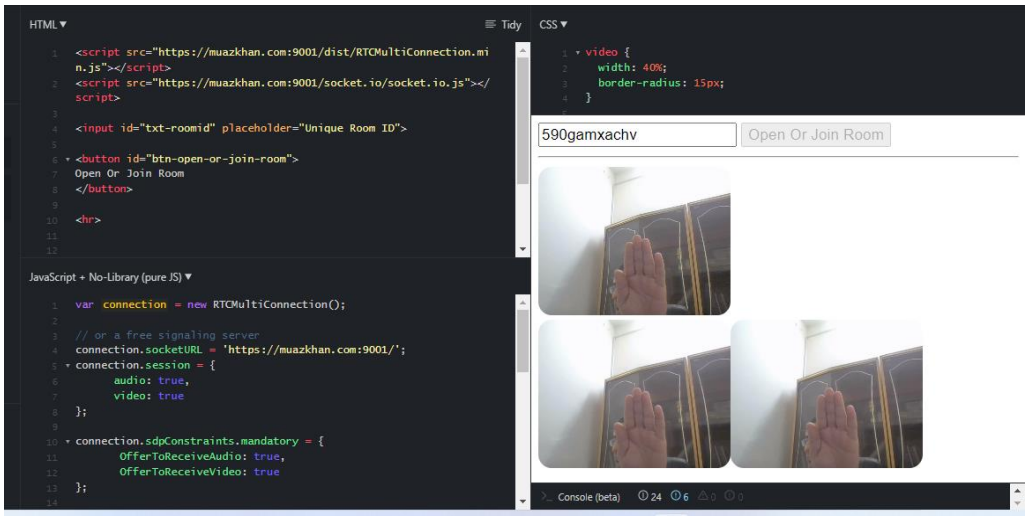


**Figure 7: Communication of audio and Video between three peers.**

## 5. Conclusion

In this study, Applications, potential solutions, and various WebRTC signalling techniques and protocols were examined and described. Moreover, in-depth research has been done on most signalling platforms and commercial servers with their limitations. In this study, a mesh topology for WebRTC bi-directional video conferencing over the Internet and 4G was proposed, tested, and implemented in real-time. Additionally, communication between participants was constructed, established, and ended using the socket.io signalling mechanism. This signalling requires an average of 171 milliseconds to prepare and 453 milliseconds to deliver and receive a request. Additionally, a thorough explanation of the physical implementation's CPU performance, memory use, socket.io signalling performance, RTPs computation, QoE, and mesh topology was made. This scenario is effective because it gives a visual demonstration of the many networks and browsers for a user who needs an in-depth

explanation and face-to-face interaction. Additionally, it enhances relationships, enhances communication, and increases productivity among users and nine teams. Additionally, this experiment can be used with three to five peers in a variety of contexts, including entertainment, e-Learning between teachers and students, and m-Health between patients and doctors or specialists and technicians. Future work on this project will involve scaling up video conferencing and utilizing a different WebRTC signalling technique.

**References**

[1] B. Y. Julian. Jang-Jaccard, Surya. Nepal, Branko. Celler, "WebRTC-based video conferencing service for telehealth," *Computing*, vol. 98, no. 1–2, pp. 169–193, 2016, doi: 10.1007/s00607-014-0429-2.

[2] X. L. Calpe, "Study, design and implementation of WebRTC for a real-time multimedia messaging application," Universitat Politècnica de Catalunya, 2017. [Online]. Available: file:///C:/Users/Naktal/Downloads/memoria.pdf

[3] M. Phankokkruad and P. Jaturawat, "An Evaluation of Technical Study and Performance for Real-Time Face Detection Using Web Real-Time Communication," in *International Conference on Computer, Communication, and Control Technology (I4CT)*, 2015, no. I4, pp. 162–166. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/7219558/

[4] H. Shane, "Video-to-Video Using WebRTC," in *JavaScript Creativity: Exploring the Modern Capabilities of JavaScript and HTML5*, Apress, 2014, p. 184.

[5] L. O. D. N. Eirik. Fosser, "Quality of Experience of WebRTC based video communication," Norwegian University of Science and Technology, 2016. [Online]. Available: https://brage.bibsys.no/xmlui/bitstream/handle/11250/2409900/15147_FULLTEXT.pdf?sequence=1

[6] L.-F. L, G. B, G. M, and G. F, "Designing and evaluating the usability of an API for real-time multimedia services in the Internet," *Multimed. Tools Appl.*, vol. 76, no. 12, pp. 14247–14304, 2016, doi: 10.1007/s11042-016-3729-z.

[7] S. Vashishth, Y. Sinha, and K. H. Babu, "Addressing Challenges in Browser Based P2P Content Sharing Framework Using WebRTC," in *30th International Conference on Advanced Information Networking and Applications (AINA)*, 2016, pp. 850–857. doi: 10.1109/AINA.2016.143.

[8] B. Sredojev, D. Samardzija, and D. Posarac, "WebRTC technology overview and signaling solution design and implementation," in *38th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO - Proceedings*, 2015, no. May, pp. 1006–1009. doi: 10.1109/MIPRO.2015.7160422.

[9] A. A. Lozano, "Performance analysis of topologies for Web-based Real-Time Communication ( WebRTC )," Aalto University, 2013. [Online]. Available: https://aaltodoc.aalto.fi/bitstream/handle/123456789/11093/master_Abelló_Lozano_Albert_2013.pdf

[10] C.-F. Hsiao, "Development of a web-based distributed interactive simulation (DIS) environment using javascript," Monterey, California: Naval Postgraduate School, 2014. [Online]. Available: http://calhoun.nps.edu/bitstream/handle/10945/43928/14Sep_Hsiao_Chen-Fu.pdf?sequence=1&isAllowed=y

[11] C. Jakobsson, "Peer-to-peer communication in web browsers using WebRTC," UMEA

University, 2015. [Online]. Available: http://www8.cs.umu.se/education/examina/Rapporter/ChristerJakobsson.pdf
[12] A. Albas and G. Auguets, "WebRTC," Politècnica de Catalunya, 2016. [Online]. Available: https://upcommons.upc.edu/bitstream/handle/2117/106694/alex.albas_117680.pdf
[13] E. Bash, "Javascript Session Establishment Protocol," in *PhD Proposal*, 2016, vol. 1, pp. 1–85. doi: 10.1017/CBO9781107415324.004.
[14] DEAN. BUBLEY, "Evolving Enterprise & Contact Centers with WebRTC," CA- USA, 2015. [Online]. Available: http://www.oracle.com/us/industries/communications/evolve-enterprise-centers-webrtc-wp-3121245.pdf
[15] C. Fan, "Research on Development and Evaluation of WebRTC Signaling based on XMPP," Norwegian University of Science and Technology, 2017. [Online]. Available: https://brage.bibsys.no/xmlui/bitstream/handle/11250/2456125/17685_FULLTEXT.pdf?sequence=1
[16] E. R. J. Uberti, C. Jennings, "JavaScript Session Establishment Protocol," USA, 2017. [Online]. Available: https://tools.ietf.org/pdf/draft-ietf-rtcweb-jsep-24.pdf
[17] Sam Dutton, "Getting Started with WebRTC," *HTML5 Rocks*, 2014. https://www.html5rocks.com/en/tutorials/webrtc/basics/#toc-simple (accessed Jul. 01, 2016).
[18] Sam Dutton, "WebRTC in the real world: STUN, TURN and signaling," *HTML5*, 2013. https://www.html5rocks.com/en/tutorials/webrtc/infrastructure/ (accessed Nov. 29, 2016).
[19] M. Grinberg, "socketio Documentation," 2017. [Online]. Available: https://media.readthedocs.org/pdf/python-socketio/latest/python-socketio.pdf
[20] R. Rai, *Socket. IO Real-time Web Application Development*. BIRMINGHAM - MUMBAI: PACKT, 2013. [Online]. Available: http://books.google.com/books?hl=en&lr=&id=YgdbZbkTDkoC&oi=fnd&pg=PT9&dq=Socket+.+IO+Real-time+Web+Application+Development&ots=TVve7ogNAQ&sig=K0Sf8yhHjskpEYta799u3UtMcY4
[21] A. S. Karl. Bissereth, Billy B. L. Lim, "An Interactive Video Conferencing Module for e-Learning using WebRTC," in *Internetional Conferences*, 2014, pp. 1–4. [Online]. Available: http://www.cita.my/cita2015/docs/shortpaper/31.pdf
[22] Natalia. Chrzanowska, "Why to Use Node.js," *Nnetguru*, 2017. https://www.netguru.co/blog/pros-cons-use-node.js-backend (accessed Jan. 11, 2018).
[23] Mathieu Nebra, "Socket.io: let's go to real time," *OPENCLASSROOMS*, 2017. https://openclassrooms.com/courses/ultra-fast-applications-using-node-js/socket-io-let-s-go-to-real-time (accessed Jun. 30, 2017).
[24] Silvia Pfeiffer, "AppRTC : Google's WebRTC test app and its parameters," *ginger's thoughts*, 2014. https://gingertech.net/2014/03/19/apprtc-googles-webrtc-test-app-and-its-parameters/ (accessed Jul. 14, 2016).
[25] B. Tulu, S. Chatterjee, T. Abhichandani, and H. Li, "Secured video conferencing desktop client for telemedicine," in *Proceedings - 5th International Workshop on Enterprise Networking and Computing in Healthcare Industry, HealthCom 2003*, 2003, no. 022710, pp. 61–65. doi: 10.1109/HEALTH.2003.1218719.
[26] A. Salinas, "Advantages and Disadvantages of using ReactJS," 2006. [Online].

Available: http://stackoverflow.com/questions/28442239/advantages-and-disadvantages-of-using-reactjs

[27] J. H. S. Ludwig, J. Beda, P. Saint-Andre, R. McQueen, S. Egan, "XEP-0166: Jingle," *XMPP Standards Foundation*, 2016. https://xmpp.org/extensions/xep-0166.html#schema-errors (accessed Sep. 22, 2016).

[28] B. T. Samir. Chatterjee, Tarun. Abhichandani, Haiqing. Li, "Instant messaging and presence technologies for college campuses," *IEEE Netw.*, vol. 19, no. 3, pp. 4–13, 2005, doi: 10.1109/MNET.2005.1453393.

[29] J. Rodriguez, Pedro. Cerviño Arriba, Javier. Trajkovska, Irena. Salvachua, "Advanced videoconferencing based on webrtc," in *IADIS Multi Conference on Computer Science and Information Systems*, 2019, no. June 2014, p. 6. [Online]. Available: http://oa.upm.es/19199/1/INVE_MEM_2012_113875.pdf

[30] S. P. M. Madhura. Deshpande, "Integration of WebRTC with SIP – Current Trends," *Int. J. Innov. Eng. Technol.*, vol. 6, no. 2, pp. 92–96, 2015, [Online]. Available: http://ijiet.com/wp-content/uploads/2015/12/14.pdf

[31] E. E. Istvan. Lajtos, David. O'Byrne, "WebRTC to complement IP Communication Services," 1, 2016. [Online]. Available: https://www.gsma.com/futurenetworks/wp-content/uploads/2016/02/WebRTC_to_complement_IP_Communication_Services_v1.0.pdf

[32] Chris Ball, "WebRTC without a signaling server," *HTML5 Rocks*, 2013. https://developers.google.com/web/updates/2015/05/high-performance-video-with-hardware-decoding

[33] H. V. Cola. Cristian, "On multi-user web conference using WebRTC," in *18th International Conference on System Theory, Control and Computing (ICSTCC)*, 2014, pp. 430–433. doi: 10.1109/ICSTCC.2014.6982454.

[34] H. Wang, R. P. Liu, W. Ni, W. Chen, I. B. Collings, and S. Member, "A New Analytical Model for Highway Inter-vehicle Communication Systems," in *IEEE ICC 2014 - Mobile and Wireless Networking Symposium*, 2014, pp. 2587–2592.

[35] S. M. Y. C. Y. Chiang, Y. L. Chen, P. S. Tsai, "A video conferencing system based on WebRTC for seniors," in *nternational Conference on Trustworthy Systems and their Applications, Taichung*, 2014, pp. 51–56. doi: 10.1109/TSA.2014.17.

[36] D. W. Nattha. Buasri, Tanasak. lanpan, Ularn. Yamborisut, "Web-based interactive virtual classroom using HTML5-based technology," in *Proceedings of the 3rd ICT International Senior Project Conference, (ICT-ISPC)*, 2014, pp. 33–36. doi: 10.1109/ICT-ISPC.2014.6923212.

[37] S. P. R. A. Bevilacqua, P. Boemio, "Introducing ufo. js: A browser-oriented p2p network," in *International Conference on Computing, Networking and Communications (ICNC)*, 2014, pp. 353–357. doi: 10.1109/ICCNC.2014.6785359.

[38] A. Golden J, Pavlyshak Y, "appear.in," *Telenor Digital AS*, 2016. https://appear.in/ (accessed Jul. 06, 2017).

[39] M. L. Giuliana. Carullo, Marco. Tambasco, Mario. Di Mauro, "A Performance Evaluation of WebRTC over LTE," in *12th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, 2016, pp. 170–175. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/7429067/