

Peer-to-peer media streaming with HTML5

Ali Tariq Kalil Al-Khayyat¹, Sanabil A. Mahmood²

¹College of Dentistry, Mosul University, Mosul, Iraq

²College of Computers Sciences and Mathematics, Mosul University, Mosul, Iraq

Article Info

Article history:

Received Jan 15, 2022

Revised Oct 7, 2022

Accepted Nov 3, 2022

Keywords:

Local area network

Quality of experience

Screen monitoring

System controlling

Web real-time communication

ABSTRACT

The knowledge of web real-time communication (WebRTC) and how its customers and server operations are defined in this study. However, the world wide web consortium (W3C) and the internet engineering task force (IETF) have not yet approved an absolute signaling protocol or a complete application programming interface (API) protocol to implement WebRTC and control communication planning. WebRTC requires some type of signaling mechanism. With Chrome, Firefox, and Opera, the primary objective is to create and implement a WebRTC video call across two clients (peers) in the real world while employing local area network (LAN) and wide area network (WAN). This study also demonstrated the design of the server (as an intermediary) and graphical user interface (GUI). Additionally, a signaling method for the peer-to-peer browser connection on the Node.js platform has been developed and successfully put into use. This paper will provide an understanding of web development and an ability to understand WebRTC technology. It will also discuss how to create WebRTC signaling mechanisms and build video conferencing, as well as how to increase design quality using quality of experience (QoE) techniques.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Ali Tariq Kalil Al-Khayyat

College of Dentistry, Mosul University

Mosul, Iraq

Email: ali.khalil987@uomosul.edu.iq

1. INTRODUCTION

Web real-time communication (WebRTC) is a new benchmark created by the world wide web consortium (W3C) and the internet engineering task force (IETF) agencies [1]. WebRTC is a techniques project that includes archives, JavaScript Programming, and principles for audio, video, and data communication [2]. It also has several advantages, including no external software, installation, ease of use, and free [3]. The WebRTC was begun in May of 2011 when Google announced an open-source initiative at the time to provide browser-based real-time communications [4]. In other clarification, Rhinow *et al.* [5] explained that WebRTC is a peer-to-peer (P2P) protocol instead a client-server procedure. JavaScript language is straight used in WebRTC aimed at communicating, and it can be used in a variety of industries, such as web conferencing, video streaming, and voice and video chatting [6]. Moreover, WebRTC differs from other video conferencing apps in that it is a set of application programming interfaces (APIs) rather than a service that allows WWW developers to construct their applications using conventional WWW approaches [7]. On the other hand, WebRTC has no signaling mechanism because of many excuses, such as to escape idleness and maximize compatibility with existing knowledge [8], or for a novel use case. Signaling like the peer identification mechanism, which finds peers and arranges communication between them, relies heavily on it.

WebRTC supports the establishment of by enabling the interchange of data through channels, user communication is enhanced [1]. The session description protocol (SDP), which mixes the network addresses

and destination port for the browser, is supported by signaling, which also links the browser to a server and enables other peers to communicate with it. Signaling is therefore utilized to start announcements among several operators. WebRTC, therefore, needs some sort of signaling mechanism or protocol support [9]. However, the WebRTC group do not spread an arrangement on a definitive signaling appliance or API conventions to test WebRTC and govern communication architecture [10], [11]. As long as WebRTC's major important implementation problem is signaling [12]. The peer discovery mechanism's key component, signaling, locates any existing peers and then orchestrates the interaction between the sites. The essential modules of the WebRTC API are listed in [8] are:

- “MediaStream: it allows a web browser to access the camera and microphone”.
- “RTCPeerConnection: it allows browser-to-browser to communicate directly”.
- “RTCDataChannel: it allows browsers to send data connections and enables the exchange of arbitrary data between them”. Figure 1 shows the WebRTC MediaStream structure.

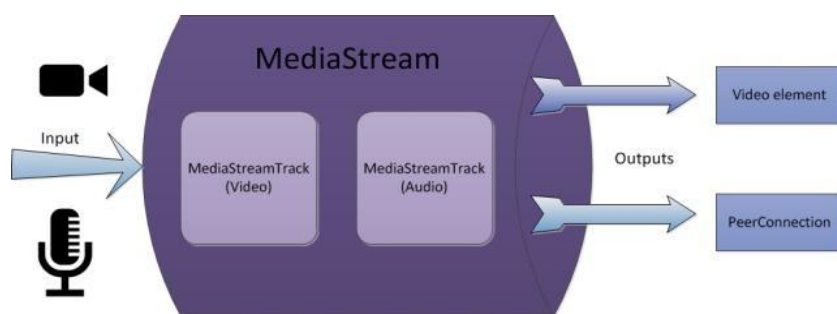


Figure 1. The structure of WebRTC MediaStream [8]

Joint science education project (JSEP) provides tools for creating and applying offers and responses to a session. However, it is completely disconnected from the actual process through which these offers and responses are conveyed to the remote side [13]. As a result of the JSEP technique, the application is solely responsible for driving the signaling state machine [5]. Furthermore, it allows web applications to bear the signaling or negotiation state, allowing genuine indicating etiquettes like (session initiation protocol (SIP) and Jingle) to function on the upper of it [14]. Simply one thing is needed: a mechanism for the app to send a native demand and negotiate distant conference facts, as well as a way to interact with the interactive connectivity establishment (ICE) [15].

The foremost idea of this research is to enterprise WebRTC video calls in an actual application with the Node.js platform. In-depth elaborations of this research will assist anxious customers to produce a consideration of WebRTC technology and communication between the client and server. This design is helpful for concerned employees who purpose to use WebRTC video conferencing for many communications, such as communication presentations, and games. This work is prepared as mentioned: the WebRTC-related exertion is assumed in section 2. The method is given in section 3. Results and discussion are obtainable in section 4. The conclusion is in section 5.

2. RELATED WORK

There are various applications, propositions and work related to a signaling mechanism. But several designers struggled to generate or advance a signaling mechanism for WebRTC. Nonetheless, more of them have been challenged with many issues as the following:

2.1. Signaling issues

The IETF and the W3C group have not yet agreed on the last signing appliance, so they have not established signaling channel standards [16]. Even if WebRTC uses the "RTCPeerConnection" API to convey data flowing between users, nonetheless a signaling tool is not a chunk of it [17]. As a result, the developer must supply the signaling protocol at the application level [18], and it is necessary to be decided upon by the two entities, either with the central node or with the other user. Thus, the client-server architecture does not appear to be a viable WebRTC solution [19]. WebRTC applications require signaling to start up a call, and clients require signaling to communicate different forms of facts. Furthermore, Chatterjee *et al.* [20] stated that the most perplexing aspect of learning WebRTC is signaling, which is often used in statement schemes to communicate that establishing a call is accomplished through information.

2.2. Specifications of the common signaling mechanisms and servers

Socket.io Protocol is a JavaScript-based transference protocol that allows web browsers and servers to communicate in real-time [21]. Socket.io began in Node.js and has remained a primary backend in Node.js ever since [22]. Google Chrome created Nore.js to provide a high-performance JavaScript engine. Accordingly, it provides an HTTP server to aid the development of web applications, but it is neither a web server nor an application server like Apache HTTP [23]. In [24], demonstrates that socket.io enables designers to use WebSockets and decide on many synchronized communication methods governed by the client's browser; a notion for WebSockets with XHR, Flash, and JSON is also described. On the other hand, it makes use of Adobe Flash Sockets, AJAX long polling, and JSON polling. Furthermore, socket.io provides a simple server and client library for supporting real-time bi-directional communication.

2.3. XMLHttpRequest signaling protocol (known as XHR)

In [25] stated that XHR does not support all current browser features and is more difficult than Jingle and XMPP. There are mentioned in [26] that XHR has a lesser throughput than WebSocket and is affected by overhead. Besides, extensible messaging and presence protocol (XMPP): XMPP is an open-source system for real-time messaging, multi-party chat, audio, and video calls. However, it is a slow text-based protocol that causes tremendous Internet traffic. In [27] describes that XMPP uses data transit between two network components. Besides, presented that XMPP connection is built on XML streams, which are made up of unidirectional elements. The XMPP restrictions have been amended, proving that XMPP is limited to a single TCP connection for client-to-server XML transmission. Additionally, Owesen-Lein [26] demonstrated the following XMPP criteria: i) it is utilized for data transmission rather than signaling and ii) it has a client-server architecture rather than a peer-to-peer architecture. In general, no important browsers support the XMPP protocol [10]. Likewise, due to XMPP's limits on many web pages, Vucic *et al.* [27] claimed that the user must restart the execution after each load.

2.4. General limitations

No tools or interfaces have been provided by WebRTC to monitor P2P structures [6]. WebRTC ethics are quiet in the development phase, and they might take some time to be full-grown [26]. Besides, Vucic *et al.* [27] demonstrated that choosing the appropriate network topology is regarded to be one of the challenging issues in the design and architecture of the WebRTC presentation; as a result, it would have to take infrastructure for the program whereas working with a combined audio/video conversation in WebRTC. Not only that but also, Ludwig *et al.* [21] emphasized that there are execution difficulties among peers as a result of their alpha or beta position; consequently, they have several bugs; for example, the present WebRTC draft is not applied within every browser due to interoperability matters between them. Too, established that WebRTC is not performed with all browsers. An instantaneous WebRTC video conferencing structure using multipoint conferencing units (MCU) is discovered in [3]. The proposed test was dependent on an exhausted MCU, which can be realistically accomplished using a single connection, whereas this situation does not deliberate any meaningful signaling. Additionally, used the Licode-Erizo (MCU) across the local area network (LAN) network to take part in a WebRTC video-conferencing application. On the other hand, García *et al.* [24] emphasized how expensive it is to use MCU, and indicated that MCU is expensive and can only be purchased from facility suppliers for a meeting. Furthermore, notes that MCU uses a significant quantity of bandwidth. based on the many items from the relevant study as shown directly above. It is clear that with WebRTC, signaling between peers and servers differs [1], [2].

3. METHOD

3.1. Implementation

An experiment lab was established to implement a WebRTC-based video conferencing system over dissimilar networks. As well as, LAN and wide area network (WAN) networks, Chrome and Firefox were used. This execution can be separated into two kinds: user application and signaling as defined below:

3.1.1. User application

To develop WebRTC video chat apps as clients and servers, a test-bed lab was established. The internet was used to establish the link. The main home page of this solution has been prepared and set up for a variety of structures, including pausing audio and video, employing full-screen, and establishing a connection between two peers (laptop computers) through Chrome. A connection was identified in advance, and the exchange of local and distant descriptions was successful (the audio and video media data). The exchange of signaling offers and responses using the SDP is described in the following (session description protocol). First, web page (Peer) 1 arrangement the local description using the “*setLocalDescription()*”

technique and goes the “*RTCPeerConnection create Offer()*” approach, and then uses the signaling server to route this conference definition to the website page (Peer) 2. Second, Peer 2 employs the “*setRemoteDescription()*” technique to group the information that Peer 1 gave as a remote description. Peer 2 sets up the local description using the “*setLocalDescription()*” process, then uses the *RTCPeerConnection createAnswer()* procedure to run the “*createAnswer()*” function on the connection. Peer 2 then gives Peer 1 this session description. When Peer 1 receives the “Peer 2 session description,” it uses the “*setRemoteDescription()*” method to set that as the distant description via (*//localhost:5555/*).

3.1.2. Signaling protocol

The WebSocket mechanism is used for the signaling operation. Four different switch message types were developed for this study solution: initiator, receiving broadcasting, peerChannel, and exchange SDP. Peer 1 will initiate the “request” in the foundation, and once the server receives the switch communication and access broadcasting stream, the situation will produce the operator media. Peer 1 then pauses till Peer 2 responds. Peer 2 will transmit a reaction signaling message up until the process has begun, and when it realizes it is not the originator, it will initially get “peer Channel” before getting access to the media stream. Then, both parties can twitch to establish a link between peers. In the SDP setup, which accepts information regarding multimedia categories, codecs, real-time transport protocol (RTP), and all linked possessions that can be used during the media session, the offer and answers messages are distributed. When Peer 1 submits a demand to Peer B and Peer 2 responds, Figure 2 shows communication between the two browsers. Figures 3 and 4 depict the procedural code of peers and the web socket server, respectively. Furthermore, Figure 4 to Figure 6 demonstrate the Node.js server's signaling mechanism when peer A sends a request to peer B based on the socket, port, and IP. And, the response from peer B to peer A until the connection will be begun.

```

1. “Get Local Media;”
2. “SWITCH Start Connection;”
3. “CASE1: create-offer/answer;”
4. “STEP1: create-offer;”
5. “IF peer connection is created;”
6. “THEN set local information and send a message;”
7. “ELSE local information not running yet;”
8. “STEP2: process signalling;”
9. “IF offeror = listening to the server's port and IP;”
10. “THEN send a request via server;”
11. “ELSE process signalling =0;”
12. “STEP3: accept request;”
13. “IF answerer accepted the SDP offer;”
14. “THEN set remote description && set local information and send a message;”
15. “ELSE offer is rejected;”
16. “STEP4: process signalling;”
17. “IF answerer = listening to the server's Port and IP;”
18. “THEN send a response via server;”
19. “ELSE process signalling =0;”
20. “STEP5: ICE candidate;”
21. “IF peers = exchanged ICEs;”
22. “THEN start stream video;”
23. “CASE2: disconnect;”
24. “IF any peer disconnected;”
25. “THEN end stream;”
26. “ELSE Re-connect;”
27. END

```

Figure 2. The pseudocode of peers

```

1. “SET WS = WebSocket Server;”
2. “SET Create server;”
3. “SET WS.Port;”
4. “SET Client;”
5. “IF client.IP = WS.IP && client.Port = WS.Port;”
6. “THEN open the channel;”
7. “ELSE IF connection = error;”
8. “THEN sendCallback;”
9. “IF request = accept;”
10. “THEN push data;”
11. “ELSE send Callback;”
12. “END”
13. “WHILE Connection Alive;”
14. “THEN start bi-directional video;”
15. “END”
16. “Peer disconnected;”
17. “END”

```

Figure 3. The pseudocode of WebSocket server

```
Node.js command prompt - node websocket.js
Your environment has been set up for using Node.js 6.11.4 (x64) and npm.
C:\Users\Waktal>cd
C:\>cd inetpub
C:\inetpub>cd wwwroot
C:\inetpub\wwwroot>cd video
C:\inetpub\wwwroot\video>node websocket.js
openssl config failed: error:02001003:system library:fopen:No such process
Sat Nov 11 2017 22:45:59 GMT+0000 (GMT Standard Time) Server is listening on port 1337
Sat Nov 11 2017 22:48:14 GMT+0000 (GMT Standard Time) Connection from origin http://localhost.
Connection :ffff:195.195.93.164
Sat Nov 11 2017 22:49:35 GMT+0000 (GMT Standard Time) Connection from origin http://localhost.
Connection :ffff:195.195.93.231
```

Figure 4. Linking of nobles A and B with the WebSocket server

```
Node.js command prompt - node websocket.js
Your environment has been set up for using Node.js 6.11.4 (x64) and npm.
C:\Users\Waktal>cd
C:\>cd inetpub
C:\inetpub>cd wwwroot
C:\inetpub\wwwroot>cd video
C:\inetpub\wwwroot\video>node websocket.js
openssl config failed: error:02001003:system library:fopen:No such process
Sat Nov 11 2017 22:58:29 GMT+0000 (GMT Standard Time) Server is listening on port 1337
Sat Nov 11 2017 22:58:29 GMT+0000 (GMT Standard Time) Connection from origin http://localhost.
Connection :ffff:195.195.93.231
Sat Nov 11 2017 22:58:29 GMT+0000 (GMT Standard Time) Received Message ["type":"offer","sdp":"m=video:64343601178668317 2 IN IP4 127.0.0.1/vtno-rlntb @/rtna-group/RUADL audio video/rtna-wid-remant1
C:\Users\Waktal>cd
C:\>cd inetpub
C:\inetpub>cd wwwroot
C:\inetpub\wwwroot>cd video
C:\inetpub\wwwroot\video>node websocket.js
openssl config failed: error:02001003:system library:fopen:No such process
Sat Nov 11 2017 22:58:29 GMT+0000 (GMT Standard Time) Received Message ["type":"candidate","label":"0","id":"audio","candidate":"candidate:1544245977 1 udp 2122260223 195.195.93.164 56695 typ host generation 0
Sat Nov 11 2017 22:58:29 GMT+0000 (GMT Standard Time) Received Message ["type":"candidate","label":"1","id":"video","candidate":"candidate:1544245977 1 udp 2122260223 195.195.93.164 56695 typ host generation 0
Sat Nov 11 2017 22:58:29 GMT+0000 (GMT Standard Time) Received Message ["type":"offer","sdp":"m=video:6474285645323568 2 IN IP4 127.0.0.1/vtno-rlntb @/rtna-group/RUADL audio video/rtna-wid-remant1
C:\Users\Waktal>cd
C:\>cd inetpub
C:\inetpub>cd wwwroot
C:\inetpub\wwwroot>cd video
C:\inetpub\wwwroot\video>node websocket.js
openssl config failed: error:02001003:system library:fopen:No such process
Sat Nov 11 2017 22:58:29 GMT+0000 (GMT Standard Time) Received Message ["type":"candidate","label":"0","id":"audio","candidate":"candidate:1722460801 1 udp 2122255103 2001:4437:0676:3255:14d3:3c3c:a218 46467
Sat Nov 11 2017 22:58:29 GMT+0000 (GMT Standard Time) Received Message ["type":"candidate","label":"1","id":"video","candidate":"candidate:61332663 1 udp 2122194087 195.195.93.231 46468 typ host generation 0
Sat Nov 11 2017 22:58:29 GMT+0000 (GMT Standard Time) Received Message ["type":"candidate","label":"1","id":"video","candidate":"candidate:61332663 1 udp 2122194087 195.195.93.231 46468 typ host generation 0"
```

Figure 5. A construction creation via WebSocket server

```
Node.js command prompt - node websocket.js
Your environment has been set up for using Node.js 6.11.4 (x64) and npm.
C:\Users\Waktal>cd
C:\>cd inetpub
C:\inetpub>cd wwwroot
C:\inetpub\wwwroot>cd video
C:\inetpub\wwwroot\video>node websocket.js
openssl config failed: error:02001003:system library:fopen:No such process
Sat Nov 11 2017 23:44:43 GMT+0000 (GMT Standard Time) Server is listening on port 1337
Sat Nov 11 2017 23:44:46 GMT+0000 (GMT Standard Time) Connection from origin http://localhost.
Connection :ffff:195.195.93.164
Sat Nov 11 2017 23:44:54 GMT+0000 (GMT Standard Time) Peer disconnected.
Sat Nov 11 2017 23:44:55 GMT+0000 (GMT Standard Time) Connection from origin http://localhost.
Connection :ffff:195.195.93.164
```

Figure 6. Interruption between the peer and WebSocket

3.2. Quality of experience (QoE)

In this research quality of experience (QoE) has been used through 15 (fifteen) actual users. In this application, they provide their specific views on the supposed user skill through the usage of surveys. The value of audio and video has been measured; hence they remained brilliant as shown in Table 1.

Table 1. Shows the QoE based on 15 users

Questions	Very Bad	Bad Annoying	Fair	Good	Excellent
Rate the ease of using the application					15
Rate the quality of audio				2	13
Rate the quality of the video				1	14
Rate the resilience of the connection			1	3	11
Does the app convince you to use WebRTC in the future				1	14

4. RESULTS AND DISCUSSION

The user interface shows how two browsers can communicate via video conferencing. There are real-time connections between the users. Consequently, there is a direct link between the peers. The video has played in both browsers without the assistance of any third-party plug-ins or downloadable software, such as flash. The `getUserMedia()` method provides access to the microphones and cameras making the connection possible. After users turn on the video conferencing feature, WebRTC requires them to obtain authorization before they may connect media devices. Google Chrome, Opera, and Firefox were used to test the application. Usage of WebRTC on modern browsers generally receives media production and adaptability. When two peers have an established channel of communication. A brand also successfully established real-time video calls between

5. CONCLUSION

The biggest challenge with using WebRTC is that it does not define the standard form of communication between browsers, which prevents it from operating as intended because it is difficult to link several browsers. This study used the Socket technology as a host to communicate two, unlike browsers to solve the aforementioned difficulty. Additionally, it developed and implemented WebRTC video conferencing, which unlike networks LAN and WAN networking may support bidirectional communication. The examination of the physical application's central processing unit (CPU) efficiency, bandwidth usage, and quality of the experience was completed.




REFERENCES

- [1] J. Jang-Jaccard, S. Nepal, B. Celler, and B. Yan, "WebRTC-based video conferencing service for telehealth," *Computing*, vol. 98, no. 1–2, pp. 169–193, Jan. 2016, doi: 10.1007/s00607-014-0429-2.
- [2] G. Carullo, M. Tambasco, M. Di Mauro, and M. Longo, "A performance evaluation of WebRTC over LTE," in *2016 12th Annual Conference on Wireless On-Demand Network Systems and Services*, 2016, pp. 170–175.
- [3] E. Fosser and L. O. D. Nedberg, "Quality of experience of WebRTC based video communication." M.S. thesis, Department of Telematics, Norwegian University of Science and Technology, 2016.
- [4] X. L. Calpe, "Study, design and implementation of WebRTC for a real-time multimedia messaging application." M.S. thesis, Master in Science in Telecommunication Engineering and Management, Universitat Politècnica de Catalunya, 2017.
- [5] F. Rhinow, P. P. Veloso, C. Puyelo, S. Barrett, and E. O. Nuallain, "P2P live video streaming in WebRTC," in *2014 World Congress on Computer Applications and Information Systems (WCCAIS)*, Jan. 2014, pp. 1–6, doi: 10.1109/WCCAIS.2014.6916588.
- [6] H. Patil and A. Buchade, "Review and study of real time video collaboration framework WebRTC," *International Journal of Computer Science and Information Technologies*, vol. 5, no. 1, pp. 108–111, 2014.
- [7] L. López-Fernández, B. García, M. Gallego, and F. Gortázar, "Designing and evaluating the usability of an API for real-time multimedia services in the Internet," *Multimedia Tools and Applications*, vol. 76, no. 12, pp. 14247–14304, Jun. 2017, doi: 10.1007/s11042-016-3729-z.
- [8] B. Sredojević, D. Samardžija, and D. Posarac, "WebRTC technology overview and signaling solution design and implementation," in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, May 2015, pp. 1006–1009, doi: 10.1109/MIPRO.2015.7160422.
- [9] M. Deshpande and S. P. Mohani, "Integration of WebRTC with SIP – current trends," *International Journal of Innovations in Engineering and Technology*, vol. 6, no. 2, pp. 92–96, 2015.
- [10] C. Cola and H. Vaeian, "On multi-user web conference using WebRTC," in *2014 18th International Conference on System Theory, Control and Computing (ICSTCC)*, Oct. 2014, pp. 430–433, doi: 10.1109/ICSTCC.2014.6982454.
- [11] Á. A. Sánchez, "WebRTC." Treball Final de Grau, UPC, Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa, Departament d'Enginyeria Telemàtica, 2016.
- [12] S. N. Malik and M. U. Saeed. Kamran, Sabir. Hussain, "A comparison of SIP with IAX an efficient new IP telephony protocol," in *International Conference of Engineering and Emerging Technology*, 2015.
- [13] J. K. Nurminen, A. J. R. Meyn, E. Jalonen, Y. Raivio, and R. Garcia Marrero, "P2P media streaming with HTML5 and




- WebRTC,” in *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPs)*, Apr. 2013, pp. 63–64, doi: 10.1109/INFCOMW.2013.6970739.
- [14] A. S. Karl, Bissereth, Billy B. L. Lim, “An interactive video conferencing module for e-learning using WebRTC,” in *International Conferences*, 2014, pp. 1–4.
- [15] A. Johnston, J. Yoakum, and K. Singh, “Taking on webRTC in an enterprise,” *IEEE Communications Magazine*, vol. 51, no. 4, pp. 48–54, Apr. 2013, doi: 10.1109/MCOM.2013.6495760.
- [16] J. Uberti, C. Jennings, and E. Rescorla, “JavaScript session establishment protocol,” *draft-ietf-rtcweb-jsep-14*, Internet Engineering Task Force (IETF), 2016.
- [17] S. Vashishth, Y. Sinha, and K. H. Babu, “Addressing challenges in browser based P2P content sharing framework using WebRTC,” in *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, Mar. 2016, pp. 850–857, doi: 10.1109/AINA.2016.143.
- [18] J. Khalid, “Comparison between VoIP clients.” M.S. thesis, Department of Telecommunication Engineering, Politecnico DI Milano, 2014.
- [19] R. Rai, *Socket.IO real-time web application development*. Packt Publishing, 2013.
- [20] S. Chatterjee, T. Abhichandani, Haiqing Li, B. Tulu, and Jongbok Byun, “Instant messaging and presence technologies for college campuses,” *IEEE Network*, vol. 19, no. 3, pp. 4–13, May 2005, doi: 10.1109/MNET.2005.1453393.
- [21] S. Ludwig, J. Beda, P. Saint-Andre, R. McQueen, S. Egan, and J. Hildebrand, “XEP-0166: Jingle,” *XMPP Standards Foundation*, 2016. Accessed: Sep 22, 2021. [Online]. Available: <https://xmpp.org/extensions/xep-0166.html#schemaerrors>
- [22] H. Rocha and R. L. Pereira, “Hyper-linked communications: WebRTC enabled asynchronous collaboration,” in *2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–7, doi: 10.1109/ICC.2017.7996702.
- [23] M. Schindler, C. Von Harscher, J. Kinzig, and S. Alekseev, “Evaluating framework for monitoring and analyzing WebRTC peer-to-peer applications,” in *Proceedings of the 11th International Network Conference*, 2016, pp. 171–175.
- [24] B. García, L. López-Fernández, F. Gortázar, and M. Gallego, “Practical evaluation of VMAF perceptual video quality for WebRTC applications,” *Electronics*, vol. 8, no. 8, Jul. 2019, doi: 10.3390/electronics8080854.
- [25] Schahin Rajab, “Comparing different network topologies for WebRTC conferencing.” KTH Royal Institute of Technology, 2015.
- [26] S. Owsen-Lein, “Unified communication and WebRTC.” B.S. thesis, Faculty of Information Technology and Electrical Engineering, Norwegian University of Science and Technology, 2015.
- [27] D. Vucic, L. Skorin-Kapov, and M. Suznjevic, “The impact of bandwidth limitations and video resolution size on QoE for WebRTC-based mobile multi-party video conferencing,” in *5th ISCA/DEGA Workshop on Perceptual Quality of Systems (PQS 2016)*, Aug. 2016, pp. 59–63, doi: 10.21437/PQS.2016-13.

BIOGRAPHIES OF AUTHORS



Ali Tariq Kalil Al-Khayyat    has a master's degree in electrical and computer engineering from Istanbul Altinbash University and he has an assistant lecturer in the Basic Science Department in the College of the Dentistry University of Mosul and works in the same College, he has a B.Sc. in computer technology engineering from the university of Mosul. and he can be contacted at email: ali.khalil987@uomosul.edu.iq.



Sanabil A. Mahmood    has been assistant literature at the Department of Computer Sciences, College of Computer Science and Mathematics, the University of Mosul, Iraq since 2022, graduated from the Computer Engineering College at the University of Mosul, Iraq in 1997 and worked as Engineering in the same Collage until 2016 when she also started studying Masters of Engineering I, then she finished M.Sc. Degree in 2018 General expertise in computer science, and my specialty is in the area of artificial intelligence and image processing. She has a research gate account under the name Sanabil Ahmed Mahmood. She can be contacted at email: sanabil_2000.uomosul.edu.iq.