Al-Rafidain Journal of Computer Sciences and Mathematics (RJCM)

www.csmj.mosuljournals.com

Using Socket.io Approach for Many-to-Many Bi-Directional Video Conferencing

Sameer Jasim Karam[1],* Bikhtiyar Firyad Abdulrahman [2]

College of Pharmacy
Hawler Medical University, Arbil, Iraq[1]
, Erbil Technical Engineering College
Erbil Polytechnic University, Arbil, Iraq[2],
**Corresponding author. Email:* Sameer.karam@hwv.edu.krd

Article information

Article history:
Received : 23/11/2021
Accepted : 27/12/2021
Available online :

Abstract

Video conferencing has become a critical need in today's world due to its importance in education and business to mention a few; also, recent years have witnessed a great revolution in communication technologies. However, there still exist limitations in these technologies in terms of the quality of communication established between two peers. Therefore, many solutions have been suggested for a variety of video conferencing applications. One of these technologies is Web Real-Time Communication technology (WebRTC). WebRTC provides the ability to efficiently perform peer-to-peer communication, which improves the quality of the communication. This work tries to propose a WebRTC bi-directional video conferencing for many-to-many (mesh topology) peers. In this work, signaling was obtained using Socket.io Library. The performance evaluation of the proposed approach was performed in terms of CPU performance, and Quality of Experience (QoE). Moreover, to validate the simulations results, a real implementation was achieved based on the following scenarios a) involving several peers, b) at the same time, opening several video rooms, c) a session will still be active even when the room initiator leaves, and d) new users can be shared with currently involved participants.

Keywords:
Video Conferencing, many-to-many communication, WebRTC, Bidirectional Communication, and Mesh Topology

Correspondence:
Author :Sameer jasim karam
Email: Sameer.karam@hwv.edu.krd

1. INTRODUCTION

The different directions of multimedia have been supported using the current communication technologies. This support enables many features in these applications' experiences such as being real-time-based [1]. Due to the low cost of Internet services with high quality, it is widely used for sending and receiving multimedia files (e.g., videos, audio, images, etc.) [2]. Moreover, a new standard called Web Real-Time Communication (WebRTC) was developed by the Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C) [3]. This standard is open-source and includes JavaScript APIs and other standards that have the ability to secure interactive communication among peers to exchange multimedia files [4]. The WebRTC also provides other advantages, for instance, it reduces cost, plug-ins are not required, ease of use, securing high-quality real-time communication, and no licenses are needed [5]. Furthermore, the IETF and W3C have not yet provided protocols for testing the WebRTC. They also have not confirmed the final signaling mechanism. Besides, in WebRTC the signaling channel standard has not yet been determined [6]. More precisely, there are no standards or a specific approach of WebRTC that can be followed to establish a communication between two browsers. This is because it is up to developers to create their protocols or to freely select the appropriate protocols for their applications such as Scoket.io, Extensible Messaging and Presence Protocol (XMPP), or Session Initiation Protocol (SIP) [7]. The peer detection can be

performed through signaling, which identifies peers as well as coordinating the communications among pairs. Using communications channels, signaling can also initiate users' communications and enables exchanging data [8]. In addition, connecting browsers to servers can be performed through signaling, which enables the communication between other peers to the defined servers. This process includes supporting the SDP protocol in merging ports numbers and network addresses aiming to provide the ability of media exchange [9].

On the other hand, the Socket.io (API), a transport protocol, can provide the feature of real-time bi-directional communication between users and servers [7]. Its programming started and ended using Node.js [10]. The authors of [9] extended the Socket.io aiming to enable designers and developers to use the WebSockets, which can identify a variety of synchronized-communication techniques that are managed by the user's browser. Moreover, the authors in [10] and [7] showed that Socket.io can provide a simple library for clients and servers. This library supports real-time bi-directional communication between both parties of communication. This feature has the ability to provide many-to-many bi-directional video conferencing. Also, it secures two communication types that help a single peer to freely choose the role as an initiator or a participant. Moreover, it provides the ability to identify a room initiator and keep open sessions in active staus even with the absence of the initiator. The aforementioned features can be utilized in a variety of communication applications (e.g., distance learning and telemedicine).

The rest of this paper is organized as follows, Section II reviews the literature, Section III describes the research method in terms of design, implementation, and results. Section IV results discussions. Finally, Section V concludes this work and presents the future possible considerations.

2. Literature Review and Problem Statement

Most of the WebRTC implementations for creating video conferencing used the pooling cycle or XMLHttpRequest (XHR). However, this approach almost causes time delay and consumes bandwidth due to the unnecessary empty responses between the browser and server that make the process busy all the time [11]. Other approaches such SIP requires to be installed and adjusted with the server used [12]. Besides, integrating WebRTC and SIP needs a lot of effort to establish a multimedia environment for sessions [13].

Pasha et al. [14] suggested an architecture for WebRTC dedicated to video conferencing. They used what has been called Multipoint Control Unit (MCU) in their proposed architecture. The authors elaborated on the proposed method and how it offers high resolutions such as session recording and stream processing. Another study performed by Fai Ng et al. [15] showed that the use of MCU was extremely expensive but it can be subscribed to some providers at the time of the conference. Also, the developers in [16] approved that MCU cannot support heterogeneous endpoints or when having a large number of participants. In the same context, Potthast [17] elaborated on some issues related to video conferencing. For instance, the codecs of video conferencing codecs are able to support up to 4 participants of multipoint.

According to the aforementioned description and the literature, MCU in usual situations presents failure quality of video conferencing, which eventually affects the whole performance of the conference [15]. Hence, this paper comes to overcome these issues and suggest an approach that guarantees to provide reliable and high-quality performance.

3. Research Method

This section describes the details of how the proposed approach was designed. The main tool used was the JavaScript language. Also, for handling the signaling, the JSFiddle platform was used as a web server. Besides, a Windows task manager was used aiming to evaluate the performance. A group of 10 computers with a variety of CPU specifications, including different CPU cores such as i5 and i7 that are connected through different locations to the Internet.

3.1 Practical Implementation

To implement WebRTC video conferencing, a test-bed lab was initiated among different users under many-to-many (mesh topology). The proposed approach used the signaling mechanism based on the Socket.io library. It is divided into two portions: the main browser set up and setting up the initiation and the termination of the communication. Also, it was crucial to creating a room with a specific identifier (room-id). This id is unique and used for establishing and joining the room. This also guarantees that particular messages will be exchanged with relevant participants, which prevents other users to access these messages. For this specific reason, each participant must have a "room-id" for sharing information with other participants in that room. This approach enables one user to initiate a room "room-id" that other users can join and participate in.

Creating a room by an initiator requires creating a new socket. This socket can be utilized for different communications purposes.

3.1.1 Browser Setting Up

The home page in our design is characterized by many features such as "open/join" a room, "mute-audio/video", and display in "full-screen". The frontend was designed to have two buttons in order to enable users to "opening" or "joining" a particular room. Also, the Socket.io library was utilized for "initiating" a constructor for the following purposes:

1. Set "Session Type" as a video conferencing session.
2. Set "SDP video directions" as bi-directional streaming.
3. Link "socket.io API" in order to involve the signalling mechanism.
4. Add "quick event handler" from the aforementioned buttons.

The first step in our proposed method was to test the code. This step is required to secure LAN communication between two users using two taps in the browser. Then, video settings were set to be with a width of 40% and border-radius of 15px

aiming to have an appropriate display. In the next step, the initiator determined the room-id and opened it. An audio/video "MediaStream" were displayed. The "MediaStream" was obtained using the "navigator.getUserMedia" method, which was performed by requesting permission for accessing the microphone and camera that were necessary for recording users' screens. Thereafter, the multimedia (video and audio) started streaming. At that step, the whole implementation was ready for the other users to join and communicate. Here, it was crucial for the participants to call "getUserMedia" in order to share their own media (microphone and cameras). For any participant who would like to leave the room, it was needed to close the browser or refresh the web page. This procedure did not impact the communications of the other participants.

3.1.2 Involving the Signaling

The MultiConnection library and the URLs of Socket.io API were utilized for the signalling. The former was used to set up and initiate a video conferencing new session. In addition, the MultiConnection library added the "onstream" function for the purpose of remote/local media streaming. According to that, a unique id was assigned to each media stream. Furthermore, the analysis of the Socket.io API was automatically generated by the Node.js server. This process can be performed based on a variety of techniques. In particular, the AJAX long polling and JSON, and Adobe Flash Socket can be involved aiming to determine the real-time communication method was the most appropriate for each user. For instance, in the case of the browser was not able to generate the JSON, Socket.io was used to suggest some communication techniques. As stated in [18] and [9], the use of Socket.io enables, at the same time, dealing with the server and client files. It could also detect the connection whether it was established with WebSocket, Flash, or AJAX. In this context, the components of the client and server sides could be provided using Socket.io with similar APIs [19]. Figure (1), presents a screeshoot from the main web.

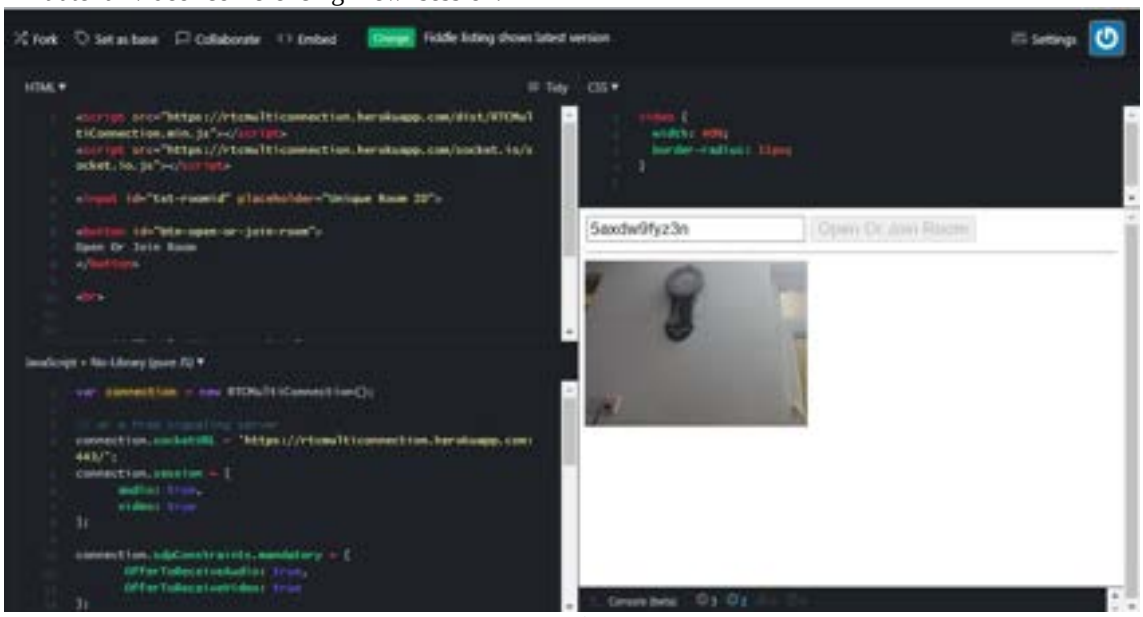

Fig1. A screenshoot from the main web

3.2 Analysis

The analysis of this work is demonstrated and described in the following subsections:

3.2.1 Bi-directional signaling

The analysis of this work was started by collecting elements from Firefox and Google Chrome browsers that were used in the experiments. The analysis of the signalling mechanism was performed based on the delay time when involving two to ten participants. Here, two aspects were used; first, the signalling delay time and the second was based on sending requests/receiving responses among participants.

The minimum and maximum times consumption to get ready were 215ms and 385ms respectively. Also, the minimum and maximum times consumption to send requests and receive responses were 390ms and 776ms respectively. The average time consumption was 251ms to be ready and 513ms for sending requests and receiving responses. The Socket.io signaling mechanism was able to fully and simultaneously control the sessions among users in terms of establishing/ending communications. The analysis also showed that there was a slight difference in delay variations between Firefox and Google Chrome. Interestingly, it was observed that the Opera browser was not supported by the signalling mechanism. Moreover, the quality of video/audio streaming was significantly affected by the bandwidth and the CPU loads. Besides, the use of Socket.io signaling caused a long delay.

3.2.2 Analysis of Video Conferencing Quality

The quality of video/audio streaming between the two participants was excellent. It was noticed that when a 3rd participant joined a room, the video/audio quality was good. However, when a 4th participant joined the same room, the

video quality was not stable and the delay was increased. This situation continues on the same behavior when a 5th participant joined the same room. In contrast, in the case of the 6th or more participants who joined the same room, it was found that the video and audio quality were significantly decreased and caused a failure. This means, 6 participants were at the threshold of the quality of the system, after this threshold the quality was insufficient. Table (1), presents the quality of video and audio among seven peers.

Table (1): Quality of video/audio among seven participants.

Number	***No. of peer***	***Duration***	***Audio Quality***	***Video Quality***
1.	2–3	3 minutes	Excellent	Excellent
2.	3- 4	3 minutes	Between acceptable and unacceptable	Unacceptable
3.	6 and more	3 minutes	Almost unacceptable	Almost unacceptable
4.	7 and more	3 minutes	Not connected	Not connected

3.3.3 Mesh Topology Evaluation

The use of mesh topology enables participants to download/upload video streaming from/to neighbors' participants at the same time, also the mesh topology is considered a complex topology [20]. This is because it consumes CPU capacity and needs high-speed bandwidth, which is also proved in the study of [21]. This means high CPU performance leads to having more participants joining the conference as well as better communication quality as indicated in [21][22]. In this regard, [23] showed that the CPUs in mesh topology consumed a lot. This is due to the encoding/decoding processes on videos especially when this kind of process is needed multiple times in parallel. Another drawback in the mesh can be on the client-side, which is a massive bottleneck that can be caused by bandwidth differences among users.

3.3.4 CPU Usage

As mentioned, in the mesh topology high load of communications is handled because of a high frequency of sending/receiving processes performed. These processes are performed at the same time for video and audio streaming, as a side effect, the whole performance is affected. The work of [22] proved that the CPU loads are significantly affected when using the mesh topology.

3.3.5 Quality of Experience (QoE) Evaluation

It was mentioned that video/audio quality for two to ten participants was gradually increased. In the Quality of Experience (QoE) evaluation, several participants were involved in the experiments using some questionnaires. The participants were asked about their experience in the presented system, their responses are presented in Table 2. The analysis of the responses showed excellent reflections on the quality of audio and video. This result reflects the case when having three users who use 4G networks. Based on these results, it can be inferred that bandwidth is crucial in affecting the quality of audio and video. Moreover, CPU performance is driven by the number of participants in a session and the loads can be significantly increased when increasing this number. However, CPU performance, in this case, can become better after the initiation of the communications among users. Also, the signaling may cause a delay at the time of sending requests and receiving responses.

Table 2: The QoE of 10 users when the communication.

Questions	***Very Bad***	***Bad***	***Fair***	***Good***	***Excellent***
Assess the quality of video using Socket.io protocol			3 - 5 users		2 users
Assess the quality of audio through the session				4 users	3 users
Assess the quality of video through the session	4 users	3 users			2 users
Assess the resilience of the connection	3 users		3 users		2 users
Assess the echo through the session	4 users		3 users		2 users

4. Discussions

In this work and according to the obtained results, it can be observed that Socket.io was useful in a mesh topology and supported the communications among different web browsers. Also, the signalling mechanism was able to handle the communications over the Internet. Moreover, signalling has the ability to provide bi-directional video conferencing and maintains sessions to stay active even when a user left the room. Another ability of signalling was shown, which was the ability to control the streaming and allow only authorized participants to join particular rooms. Another feature, it can be developed with no need for external devices or cloud/server support. This paper can be considered the first work that develops a "WebRTC Bi-Directional" video conferencing using 4G networks. It should be mentioned that the proposed approach does not support the Opera web browser.

The results also showed that some issues were faced in the quality of video/audio considering low bandwidth and low CPU specifications. This phenomenon is termed "CPU stress". It depends on many factors such as the used codecs and the defined quality of videos/audios. Besides, the differences in bandwidths among system users can be considered as an important factor in the quality of the exchanged videos/audios. Furthermore, it can be said that CPU specs play a crucial role in handling communications among a high number of users. Similarly, the bandwidth also plays a leading role in the quality of audio/video. The Socket.io signalling mechanism was not able to handle more than three users, which is a threshold of this mechanism. This mechanism also caused a high delay time when initiating the communications among network peers. Finally, the QoE confirmed that the presented testbed environment was sufficient for users.

5. Conclusion and Future Work

This work presented a WebRTC bi-directional approach for video conferencing in a mesh topology. The approach used the Internet in the experiments. The implementation of the proposed approach was real-time-based. Also, the Socket.io signaling mechanism was used to handle all the communications among participants. Besides, the work also performed a comprehensive analysis and evaluation of the performance in terms of CPU usage, signalling in Socket.io, Quality of Experience, and the mesh topology. The evaluation was performed in a physical environment. In addition, the evaluation was performed using different web browsers. Practically, the applications of this work can be applied to entertainment games, e-Learning and so on. As future work, the proposed approach can be extended to use different signaling mechanisms over the WebRTC. This work includes a lot of directions to be developed and investigated by developers.

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.

[2] 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), no. I4, pp. 162–166, 2015.

[3] 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), pp. 170–175, 2016.

[4] L. O. D. N. Eirik. Fosser, "Quality of Experience of WebRTC based video communication," Norwegian University of Science and Technology, 2016.

[5] K. S. Alan. Johnston, John. Yoakum, "Taking on webRTC in an enterprise," IEEE Commun. Soc., vol. 51, no. 4, pp. 48–54, 2013.

[6] P. J. Ha and L. D. Hoon, "Scalable signaling protocol for Web real-time communication based on a distributed hash table," Comput. Commun., vol. 70, pp. 28–39, 2015.

[7] M. Grinberg, "socketio Documentation," 2017.

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

[9] Mathieu Nebra, "Socket.io: let's go to real time," OPENCLASSROOMS, 2017. [Online]. Available: https://openclassrooms.com/courses/ultra-fast-applications-using-node-js/socket-io-let-s-go-to-real-time. [Accessed: 30-Jun-2017].

[10] R. Rai, Socket. IO Real-time Web Application Development. Birmingham - Mumbai: PACKT, 2013.

[11] J. A. S. N. Luis. López. Fernández, Migue.l París. Díaz, Raúl. Benítez. Mejías, Francisco. Javier. López, "Kurento: A media server technology for convergent WWW/mobile real-time multimedia communications supporting WebRTC," in 14th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), p. 6, 2013.

[12] E. R. J. Uberti, C. Jennings, "JavaScript Session Establishment Protocol," USA, 2017.

[13] A. A. Lozano, "Performance analysis of topologies for Web-based Real-Time Communication (WebRTC)," Aalto University, 2013.

[14] M. Pasha, F. Shahzad, and A. Ahmad, "Analysis of challenges faced by WebRTC videoconferencing and a remedial architecture," Int. J. Comput. Sci. Inf. Secur., vol. 14, no. 10, pp. 698–705, 2016.

[15] W. C. Kwok-Fai. Ng, Man-Yan. Ching, Yang. Liu, Tao. Cai, Li. Li, "A P2P-MCU Approach to Multi-Party Video Conference with WebRTC," Int. J. Futur. Comput. Commun., vol. 3, no. 5, pp. 319–324, 2014.

[16] "Video Conferencing," Daitan GROUP, 2013. [Online]. Available: http://www.daitangroup.com/video-conferencing-what-is-an-mcu/. [Accessed: 15-Jul-2017].

[17] S. Potthast, "Point to Point and Multipoint," Jisc community, 2016. [Online]. Available: https://community.jisc.ac.uk/library/janet-services-documentation/point-point-and-multipoint. [Accessed: 23-Aug-2017].

[18] David. Walsh, "WebSocket and Socket.IO," Media Temple, 2010. [Online]. Available: https://davidwalsh.name/websocket. [Accessed: 26-Jun-2017].

[19] N. Chhetri, "A Comparative Analysis of Node . js (Server-Side JavaScript)," St. Cloud State University, 2016.

[20] A. W. AlTuhafi, "A Review on Peer-to-Peer Live Video Streaming Topology," IEEE-International Conf. RFID-Technologies Appl., vol. 68, no. 5, pp. 1–4, 2013.

[21] T. S. Edan. N, Al-sherbaz. A, "Desing and Implement A Hybrid WebRTC Signalling Mechanism for Unidirectional & Bi-directional Video Conferencing," Int. J. Electr. Comput. Eng., vol. 8, no. i, pp. 1–8, 2018.

[22] N. M. Edan, A. Al-sherbaz, and S. Turner, "WebNSM : A Novel Scalable WebRTC Signalling Mechanism for Many-to-Many Video Conferencing," in 3rd IEEE International Conference on Collaboration and Internet Computing (CIC), vol. 2, pp. 1–7, 2017.

[23] A. S. Karl. Bissereth, Billy B. L. Lim, "An Interactive Video Conferencing Module for e-Learning using WebRTC," in Internetional Conferences, pp. 1–4, 2014.

[24] R. P. and S. Picek, "Multipoint Web Real-Time Communication," in Information System Development, Switzerland: Springer, pp. 99–110,2014.

استخدام اسلوب Socket.io لعقد مؤتمرات فيديو ثنائية الاتجاه بين العديد من الأطراف

سمير جاسم كرم — **بەختیار فریاد عبدالرحمن**

sameer.karam@hmu.edu.krd — Bikhtiyar.abdulrahman@epu.edu.iq

كلية الصيدلة — **كلية اربيل للهندسة الفنية**

جامعة هولير الطبية،أربيل، العراق — **جامعة بوليتكنيك أربيل ، أربيل ، العراق**

تاريخ استلام البحث: 2021/11/23 — **تاريخ قبول البحث: 2021/12/27**

الخلاصة:

أصبحت مؤتمرات الفيديو حاجة ماسة في عالم اليوم نظرًا لأهميتها في التعليم والأعمال على سبيل المثال لا الحصر ؛ كما شهدت السنوات الأخيرة ثورة كبيرة في تقنيات الاتصال. ومع ذلك ، لا تزال هناك قيود في هذه التقنيات من حيث جودة الاتصال بين اثنين من الأقران. لذلك ، تم اقتراح العديد من الحلول لمجموعة متنوعة من تطبيقات مؤتمرات الفيديو. إحدى هذه التقنيات هي تقنية Web Real-Time Communication (WebRTC) (التواصل عبر الويب في الوقت الفعلي). يوفر WebRTC القدرة على أداء اتصال نظير إلى نظير بكفاءة ، مما يحسن جودة الاتصال. يحاول هذا العمل اقتراح مؤتمر فيديو ثنائي الاتجاه WebRTC لأقران متعدد إلى كثير (طوبولوجيا المعشقة). في هذا العمل ، تم الحصول على الإشارات باستخدام مكتبة Socket.io. تم إجراء تقييم أداء النهج المقترح من حيث أداء وحدة المعالجة المركزية وجودة التجربة (QoE). علاوة على ذلك ، للتحقق من صحة نتائج المحاكاة ، تم تحقيق تنفيذ حقيقي بناءً على السيناريوهات التالية أ) إشراك العديد من الأقران ، ب) في نفس الوقت ، فتح عدة غرف فيديو ، ج) ستظل الجلسة نشطة حتى عندما يغادر منشئ الغرفة ، و د) يمكن مشاركة المستخدمين الجدد مع المشاركين المعنيين حاليًا.

الكلمات المفتاحية : مؤتمرات الفيديو ، والاتصالات بين أطراف متعددة ، و WebRTC ، والاتصال ثنائي الاتجاه ، والطوبولوجيا المعشقة (Mesh Topology).