Applying Software Engineering Based on Peer-to-Peer Communication

Asma Salim Yahya[*]

[*]College of Computer Sciences and Mathematics, The University of Mosul, Mosul, Iraq, asma_alkhairi@uomosul.edu.iq

Abstract: The JavaScript programming language was selected to create software creation that facilitates the creation of a video connection between users because it enables the creation of cross-platform apps relatively quickly. For example, "Web Real-Time Communication (WebRTC)" standards do not specify exactly how two browsers initiate and manage communication with one another. The reason for this is that WebRTC does not specify the signaling technique or protocol. This paper's main goal is to design and construct a WebRTC simultaneous video conference between peers utilizing Google Chrome and the Socket.io signaling technology. A Local Area Network was used in this experiment (LAN). Furthermore, an assessment was conducted on the quality of experience (QoE), the Socket.io signaling method, and resources, including bandwidth consumption. This paper describes the simultaneous execution of peer-to-peer video calls with user identification (user-id).

Keywords: Software Engineering (SE), Software Design Process, Web Real-Time Communication, Quality of Experience (QoE), 4th internet generation.

1. Introduction

Online resources are frequently used by software developers for a range of software engineering duties, such as understanding APIs, resolving bugs, understanding code or concepts, etc. Most of these requests for assistance involve regular interaction or recommendations from other developers [1]. The term Software Engineering (SE) dates back to 1968 when a NATO study group was established to address software-related challenges and advocate for the integration of software in various aspects of our lives. This discipline falls within the realm of engineering and involves thorough analysis, design, and development processes, similar to other engineering fields like electrical, mechanical, and civil engineering. Software products are designed for specific purposes, akin to other engineering endeavors. According to IEEE, software engineering entails employing a systematic, disciplined, and measurable approach to create, operate, and maintain software [2]. On the other hand, a new standard known as Web Real-Time Communication (WebRTC) was created by the World Wide Web Consortium (W3C) for browser API and the Internet Engineering Task Force (IETF) for wire protocol [3]. WebRTC is a set of open-source tools, JavaScript APIs, and standards that facilitate interactive audio, video, and data communications [4]. Additionally, it provides several advantages cost-free, license-free, no requirement for external or internal plug-ins, simplicity of usage, and excellent Real Time-Communication (RTC) application [4]. However, neither the W3C nor the IETF has yet approved a protocol for testing WebRTC nor a final signaling method [5]. As a result, WebRTC did not specify the signaling channel standard. Put differently, the exact methods by which two browsers initiate and manage their communication are not specified by WebRTC standards [6]. The rationale for this is that signaling is assumed to be left up to the developer, giving application providers the freedom to design their protocol based on a new use case or one of the established protocols (such as Socket.io, Session Initiation Protocol (SIP). to design their protocol; or for an innovative application [7]. Most importantly, to minimize incompatibility with legacy technology and prevent duplication. Signaling plays an important role in peer detection, which identifies peers and facilitates communication between them. It also helps users establish communication by transferring data over channels. Real-time bi-directional interaction between a client and a server is provided by Socket.io (API). It is a JavaScript-written transport protocol that permits web browsers and servers to connect in real-time in both orders [8]. Peer-to-peer bi-directional video calls, joining already-existing sessions, blocking self-remote streams, and using users to prevent the anonymous peer from attending the meeting are all made possible by this experiment.

This document is organized and outlined as follows: The work on the WebRTC survey is detailed in Section II. A description of the paper's methodology, including its application and analysis, is assumed in Section III. Section IV discusses the evaluation. Suggestions for additional development are included in Section V's conclusion.

2. Related Work

As mentioned in [9], the primary function of the application—which is now unspecified in terms of WebRTC between the browser and the server—is signaling. Consequently, the developer must supply the signaling protocol at the application level. Additionally, a secure video chat built on the webRTC standard was developed. There are two presentation components to the software implementation, which is a publicly accessible web application. To gather the data required to place a video call, the application sends messages to the STUN and signaling servers in a sequential manner. Traditionally, JavaScript has been used to construct software products that implement the establishment of video connections between end users since it is a relatively quick programming language for cross-platform applications [10]. Additionally, the author looked at a web application that permits real-time communication but is prone to network issues that degrade user experience. The study's authors use subjective assessments to assess a WebRTC-based audio/video service's conversation quality [11]. Also, in [12] the author used WebRTC technologies to enable the execution of safe and strong data transmission among users as peer-to-peer in real-time communication. However, it did not allow users to communicate with one another via video/audio calls while using simple JavaScript APIs and Node JS.

3. Implementation

The JSFiddle platform has been utilized in this application to provide a collaborative HTML, CSS, and JavaScript code design and testing environment. Utilized a Wireshark analyzer in addition to determine the bandwidth usage. Moreover, client-side functionality was provided by Google Chrome. In addition, a LAN network connects two PCs with i5 CPU cores and 4 GB of RAM. To evaluate WebRTC video conferencing amongst several peers, a test-bed lab was established. This accomplishment uses a signaling system that is split into two halves and is based on the Socket.io API:

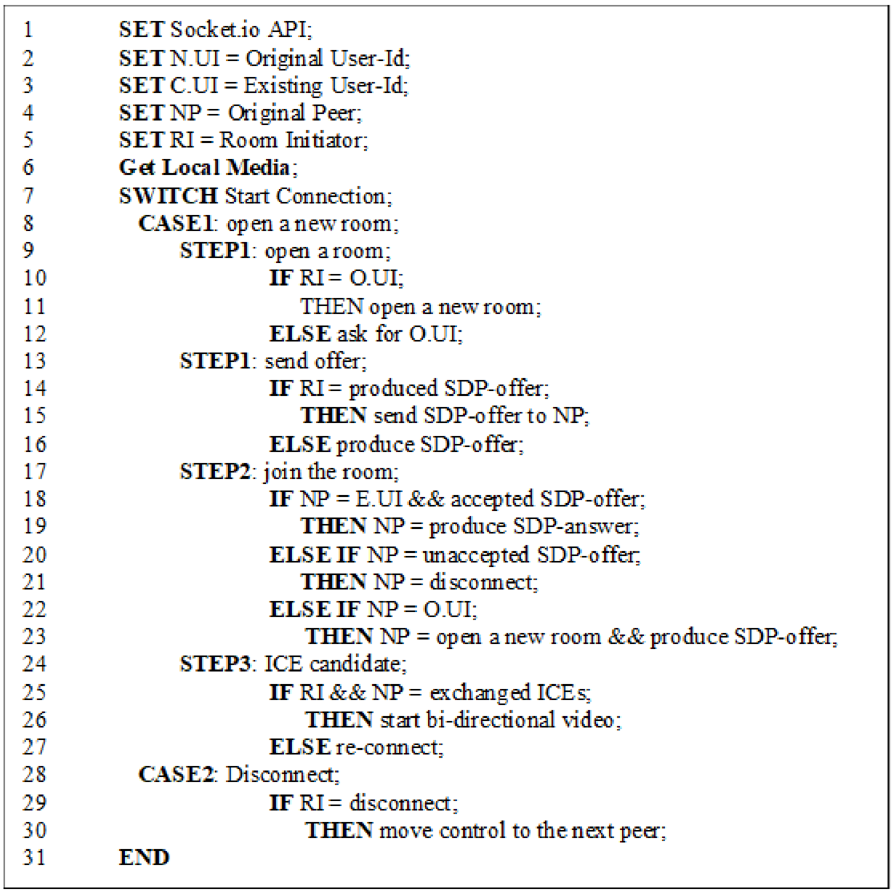
1) The main browser.

2) Utilised Socket.io API.

An ICE that has been collected by the starter and other participants, as well as an SDP offer, can be obtained by opening a new socket. Put differently, each peer is supposed to act as an "offeror" for any new member after establishing an original peer linking, broadcast their details via the default channel, and build and exchange "SDP-offer/answer". Rather, everyone entering the room ought to take on the position of "answerer.

3.1 Structure a Web Page

Numerous capabilities are available on the test's main webpage, including the ability to create and send SDP offers and answers, access local devices such as cameras and microphones, open and join rooms, disable audio and video, and use full-screen mode. In order to (a) establish a video conference session, (b) enable SDP video simultaneous broadcasting, and (c) link Socket.io API to be utilized as a signaling contrivance that manages communication between two peers, it was created with a single button that peers can use to open or join the room. Additionally, a "MediaStream" can be acquired by requesting access to the camera and microphone in order to record the peer's screen using the "navigator.getUserMedia" method. A camera will begin broadcasting as soon as authorization is obtained, at which point the process will be prepared for joining or communication. It is required of all participants to contribute their cameras and microphones by calling "getUserMedia". The experiment's pseudocode is displayed in Figure (1). A peer can discontinue the live broadcast of their camera or microphone without interfering with other peers' communication by refreshing or closing the JSFiddle platform browser page before leaving the room.



```
SET Socket.io API;
SET N.UI = Original User-Id;
SET C.UI = Existing User-Id;
SET NP = Original Peer;
SET RI = Room Initiator;
Get Local Media;
SWITCH Start Connection;
  CASE1: open a new room;
      STEP1: open a room;
               IF RI = O.UI;
                  THEN open a new room;
               ELSE ask for O.UI;
      STEP1: send offer;
               IF RI = produced SDP-offer;
                  THEN send SDP-offer to NP;
               ELSE produce SDP-offer;
      STEP2: join the room;
               IF NP = E.UI && accepted SDP-offer;
                  THEN NP = produce SDP-answer;
               ELSE IF NP = unaccepted SDP-offer;
                  THEN NP = disconnect;
               ELSE IF NP = O.UI;
                  THEN NP = open a new room && produce SDP-offer;
      STEP3: ICE candidate;
               IF RI && NP = exchanged ICEs;
                  THEN start bi-directional video;
               ELSE re-connect;
  CASE2: Disconnect;
               IF RI = disconnect;
                  THEN move control to the next peer;
END
```

Fig 1. Implementation pseudocode

3.2 Create a Signaling Using Socket.io

Based on the Socket.io API, this signalling has been employed. Upon analyzing the Socket.io mechanism, it was discovered that the Socket.io.js file is automatically recovered from the client by the Node.js server.

Based on the signalling delay for two concepts—the first relied on the signalling delay to get ready, and the second was based on sending a request and receiving a response from peers, this signalling method has been studied independently between two users. The analysis was based on the network analysis examining components of Google Chrome during the actual communication. As a result, it takes 150 milliseconds (ms) at the very least and 200 (ms) at the most to prepare. To send a request and receive a response, it also uses 330 (ms) as a minimum and 940 (ms) as a maximum. After calculating the mean time, it takes 111 milliseconds to be ready and 433 milliseconds to send a request and receive a response. The Socket.io signalling method allows all participants to initiate, establish, and terminate communication at the same time. However, a lengthy delay was discovered through the Socket.io signalling technique.

3.3 Quality of Video Conferencing

Individual tests conducted between two peers showed a progressive improvement in audio and video quality over the Internet network. As a result, there was outstanding audio and visual quality between the two peers. Therefore, for a maximum of two peers, using the Socket.io protocol for communication is efficient.

3.4 Bandwidth Consumption

The examination of the bandwidth consumption revealed that (a) each peer requires an absolute minimum of 2 Mb/s capacity for each RTP on the video, and (b) each peer requires a minimum of 65 kb/s bandwidth for a single RTP on the audio. The analysis was conducted using a Wireshark analyzer.

3.5 Quality of Experience (QoE)

From two to ten peers, the audio and video quality progressively improved. As a result, using the questionnaires shown in Table 1, real users took part in the test and provided their unique perspectives on the realized user experience (1). This approach showed exceptional audio and video quality, especially when shared across three peers via 4G networks and the Internet. Some peers' responses, though, were ineffective, and at times they took longer than three minutes. The examination indicates that it can take into account CPU performance as a means of permitting or limiting the number of peers, as well as the use of bandwidth as a means of lowering the quality of music and video. As a result, quality improves when a peer waits for other members till the CPU calms down. On the other hand, this signaling has caused a significant delay from the time a request is issued until a response is received.

Table (1), QoE for online conversation between two users.

Questions	Very Bad	Bad	Fair	Good	Excellent
Evaluate the conversation with Socket.io		2	2	3	3
How user-friendly is the application?				9	1
Rate the session's audio quality				7	3
During the session, rate the video's quality			3	6	1
Throughout the session, rate the echo					10
Does using WebRTC with Software Engineering make you want to use it even more going forward?					10

3.5.1 Applying Software Engineering

The Repeated Version model is another name for the incremental process paradigm. First, a few basic functionalities are built into a simple, functional system, which is then delivered to the client. Up until the intended system is released, more iterations/versions are put into effect and supplied to the client [13]. The incremental mode used in this work is depicted in Figure (2).

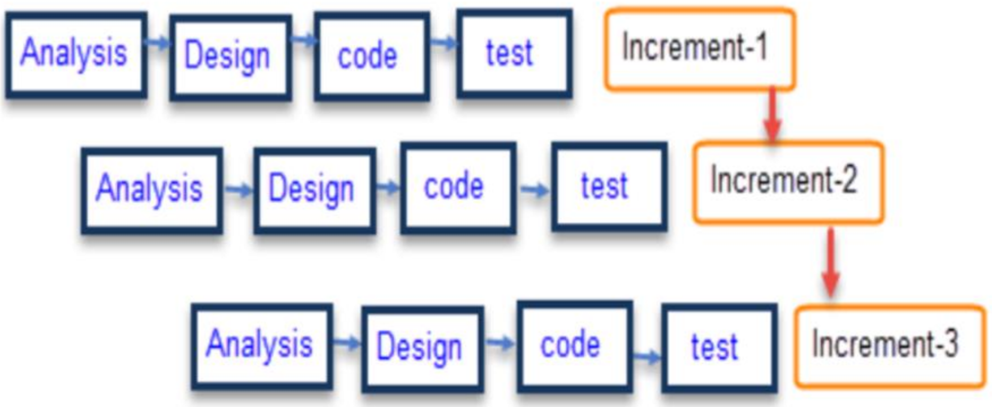


Fig 2. The Incremental Model [14].

3.5.2 Use Case Diagram

Use case gives an overview of how the system or business processes work from the user's point of view. The first step has been done in making a use case diagram to think about how a user "uses" the system. (Unhelkar, et al,, 2017). This diagram has two levels, an elementary level, and a detailed level (to represent the graphical user interface (GUI)). The initial level characterizes the application's primary user interface. Figure (3) depicts that when the caller enters the system, the type of connection is selected, and then the connection is initiated, after which the call is terminated by one of the callers.

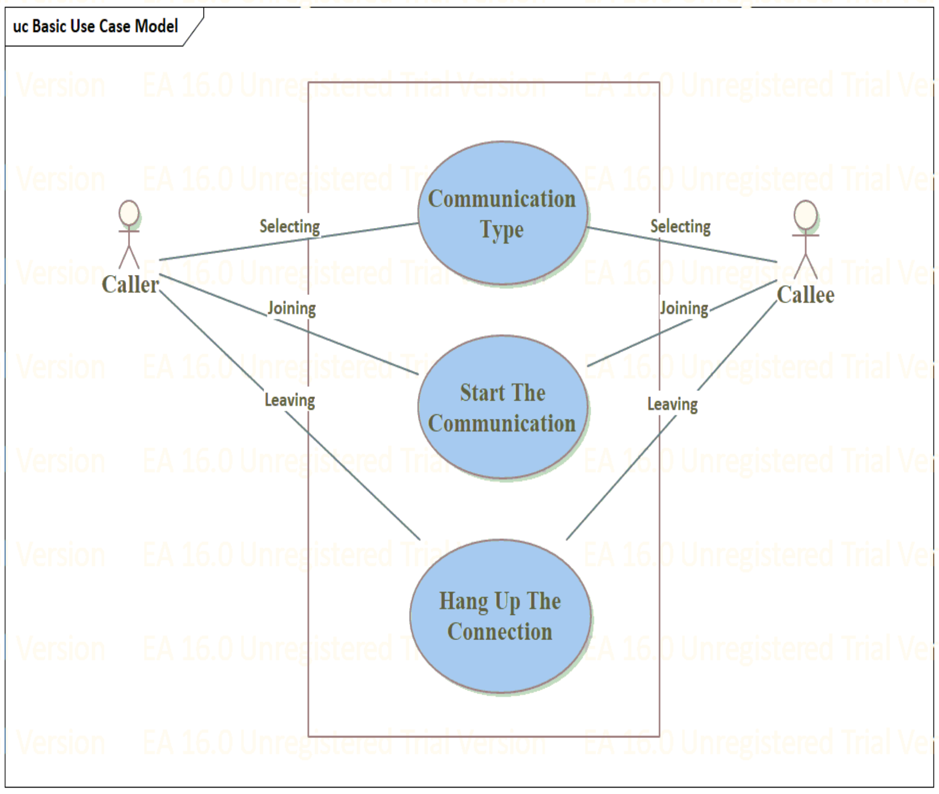


Fig 3. The initial level of the application's Use Case Diagram.

The diagram shows the exact details of the application’s work, where the caller sends a request to the callee, and then the callee will select the type of communication and make a communication response. If the type of communication is a video call, the communication functions appear such as mute/unmute the audio, on/off the camera, screen sharing, screen recording, and the chat for exchanging messages, regarding a button in order to hang out the call, it will appear in the communication interface. In contrast, if the type of communication is chatting, then it will only show the chat with a button to end the call.

4. Conclusion

It is established that Socket.io can be used to facilitate browser-to-browser communication. Through the use of a signaling method, this solution may initiate, maintain, and terminate Internet-based connection. In addition, it provides bi-directional video conferencing, manages self- and remote streams, stops non-candidates from entering the room, and maintains the productivity of the session even if a peer exits. However, it was made without the aid of any outside technology. Furthermore, bandwidth is a major factor in both audio and video quality. More than three peers cannot be supported by the socket.io signaling technique to provide outstanding audio and visual quality. Additionally, there is a significant delay as peer-to-peer communication takes a while to start. The QoE confirms that this laboratory environment functions as intended and may be utilized going forward for larger-scale user expertise trials. This paper describes the architecture and real-time testing of WebRTC bi-directional video calls over the Internet. Furthermore, the Socket.io signaling system was used to initiate, maintain, and terminate peer-to-peer communication. This scenario works well because it gives a user who needs an in-depth explanation and direct communication a visual demo via the network and browser. Furthermore, it enhances connections, boosts productivity, and facilitates better communication between teams and users.

References

[1] S. Kabir, D. N. Udo-Imeh, B. Kou, and T. Zhang, "Who Answers It Better? An In-Depth Analysis of ChatGPT and Stack Overflow Answers to Software Engineering Questions," in arXiv preprint arXiv, 2023, p. 2308.02312. [Online]. Available: http://arxiv.org/abs/2308.02312

[2] M. Savary-Leblanc, L. Burgueño, J. Cabot, X. Le Pallec, and S. Gérard, "Software assistants in software engineering: A systematic mapping study," Softw. - Pract. Exp., vol. 53, no. 3, pp. 856–892, 2023, doi: 10.1002/spe.3170.

[3] A. T. K. Al-Khayyat and S. A. Mahmood, "Peer-to-peer media streaming with HTML5," Int. J. Electr. Comput. Eng., vol. 13, no. 2, pp. 2356–2362, 2023, doi: 10.11591/ijece.v13i2.pp2356-2362.

[4] S. Nursiti, "Design and Implementation of Peer-to-Peer Video and Chat Communication," Tech. Soc. Sci. J., vol. 17, pp. 235–243, 2021.

[5] N. M. Edan, A. Al-Sherbaz, and S. Turner, WebNSM: A novel scalable WebRTC signalling mechanism for one-to-many Bi-directional video conferencing, vol. 857. 2019. doi: 10.1007/978-3-030-01177-2_40.

[6] S. J. Karam and B. F. Abdulrahman, "Using Socket . io Approach for Many-to-Many Bi-Directional Video Conferencing," Al-Rafidain J. Comput. Sci. Math., vol. 16, no. 1, pp. 81–86, 2022.

[7] K. Jannes, E. H. Beni, B. Lagaisse, and W. Joosen, "BeauForT: Robust Byzantine Fault Tolerance for Client-Centric Mobile Web Applications," IEEE Trans. Parallel Distrib. Syst., vol. 34, no. 4, pp. 1241–1252, 2023, doi: 10.1109/TPDS.2023.3241963.

[8] S. A. Mahmood, "Proposed a WebRTC Data Communication Using Effective Signalling Protocol," Tech. Rom. J. Appl. Sci. Technol., vol. 4, no. 10, pp. 83–96, 2022, doi: 10.47577/technium.v4i10.7956.

[9] A. M. Jawad, "Using JavaScript on 5G networks to improve real-time communication through WebRTC," Al-Rafidain J. Eng. Sci., vol. 1, no. 1, pp. 9–23, 2023.

[10] E. Revyakina, "Development of a secure video chat based on the WebRTC standard for video conferencing," in E3S Web of Conferences, 2023, vol. 07017, pp. 1–8.

[11] S. Abidi, S. Suleman, V. Dhanush, and H. Jm, "WEBRTC BASED VIDEO CONFERENCE APP," Int. Res. J. Mod. Eng. Technol. Sci., vol. 05, no. 06, pp. 833–838, 2023.

[12] A. Tiwari, H. Gupta, S. Mittal, and M. Rawat, "Voice Chat Web App Using Webrtc," IJRDO -Journal Comput. Sci. Eng., vol. 8, no. 11, pp. 23–28, 2022, doi: 10.53555/cse.v8i11.5444.

[13] J. Leong, K. May Yee, O. Baitsegi, L. Palanisamy, and R. K. Ramasamy, "Hybrid Project Management between Traditional Software Development Lifecycle and Agile Based Product Development for Future Sustainability," Sustainability, vol. 15, no. 2, p. 1121, 2023, doi: 10.3390/su15021121.

[14] M. Beyazit, T. Tuglular, and D. O. Kaya, "Incremental Testing in Software Product Lines - An Event Based Approach," IEEE Access, vol. 11, no. October 2022, pp. 2384–2395, 2023, doi: 10.1109/ACCESS.2023.3234186.