Design and Evaluation of Browser-to-Browser Video Conferencing in WebRTC

Naktal Moaid Edan
School of Science and Technology, The University of Northampton
Northampton, United Kingdom
The University of Mosul, Mosul, Iraq
naktal.edan@northampton.ac.uk

Ali Al-Sherbaz, Scott Turner
School of Science and Technology, The University of Northampton
Northampton, United Kingdom
{ali.al-sherbaz, scott.turner}@northampton.ac.uk

***Abstract*— This paper describes the Web Real-Time Communication (WebRTC) technology and the implementation of its clients and server. The main aim is to design and implement WebRTC video conferencing between browsers in real implementation using Chrome and (Wired & WiFi) of LAN & WAN networks. Also, an evaluation of CPU performance, bandwidth consumption and Quality of Experience (QoE) was achieved. Moreover, a signalling channel between browsers using the WebSocket protocol via *Node.js* platform has been created and executed. This paper will give web developer an opportunity to comprehend the WebRTC technology, as well as to understand how to design WebRTC video conferencing.**

Keywords*—*The Web Real-Time Communication (WebRTC); WebSocket protocol; Node.JS. Local Area Network (LAN); Wide Area Network (WAN); Quality of Experience (QoE).

I. INTRODUCTION

Real-Time Communication (RTC) over voice and video has several benefits, but due to several issues such as expensive video and audio licensing, RTC poses several challenges that have attracted the research community [1]. The World Wide Web Consortium (W3C) and Internet Engineering Task Force (IETF) developed a new standard known as WebRTC; they have commented that the WebRTC is designed to permit the co-occurrence of audio and video sessions without the need to plug-ins or other fees. WebRTC is a peer-to-peer open source framework that is considered as a collection of standards, protocols and JavaScript [2]. Also, it is supported by Opera, Mozilla Firefox and Google Chrome [3]. To establish communication among various users and/or devices, WebRTC requires a kind of signalling mechanism or a support of protocols [4]. But, the Internet Engineering Task Force (IETF) and World Wide Web Consortium (W3C) have not yet agreed to a final signalling mechanism or a final API protocol to test WebRTC and to regulate communication architecture [1][5]. Therefore, signalling is the main significant implementation issue with WebRTC [6]. Signalling is the heart of the peer discovery mechanism that finds any present peers and then coordinates communication between the browsers [7]. The major components of the WebRTC API as mentioned in [7][8] are: (a) MediaStream: allows a web browser to access the camera and microphone, (b) RTCPeerConnection: allows browser-to-browser to communicate directly and (c) RTCDataChannel: allows browsers to send data connections and enables the exchange of arbitrary data between them. Moreover, WebRTC uses Interactive Connectivity Establishment (ICE) technique For traversing Network Address Translation (NATs) that relies on Session Traversal Utilities for NAT (STUN) to find out the external address is assigned to a particular peer and Traversal Using Relay NAT (TURN) to support STUN and to bypass the Symmetric NAT restriction by opening a connection with a TURN server and relaying all information through that server [7]. The primary objectives of this paper are to design WebRTC video conferencing in a physical implementation using the *Node.js* platform. Including, an evaluation of the bandwidth consumption, CPU performance and Quality of Experience (QoE). In-depth elaboration of this paper will help concerned users to get an understanding of WebRTC technology and communicating between the client and server. This illustration is beneficial for interested users who intend to use WebRTC video conferencing among various communications such as communication applications, e-learning, mHealth, monitoring, game, etc.

This paper is organised as follows: The definitions in section II. Reports on survey comments of some WebRTC issues (related work) are given in section III. The implementation is presented in section IV. Evaluation is explained in section V. Finally, the conclusion and future work in section VI.

II. DEFINITIONS

A. *WebSocket (WS)*

It is a protocol that is based on the web and it allows a continuous and bi-directional TCP or secured Transport Layer Security (TLS) to communicate between a web client and a remote/ web server. It leaves the session open for messages from both directions. Thus, users can have so many messages whether binary or textual travelling on both directions [8].

B. *Node.JS*

Google Chrome builds *Node.JS* as a server platform and is Google's open source high-performance JavaScript engine. *Node.JS* is an appropriate platform for building network web

978-1-5090-4934-9/17/$31.00 ©2017 IEEE

application. *Node.JS* can handle high throughput and performance because it is based on non-blocking Input/Output and event-driven model [9].

III. LITERATURE REVIEW

Different developers attempted to create or develop a signalling mechanism or a protocol for WebRTC. But, most of them faced some issues. The following elaborations will describe some of these matters:

As mentioned in [10], signalling management has not yet been specified by WebRTC in order to allow the developer to modify, reuse existing protocols and permit them the freedom to design their own signalling; moreover, to avoid redundancy and to increase compatibility with established technologies [8]. An overview of WebRTC video conferencing architecture using MCU is shown in [11]. However, this scenario does not discuss any kind of signalling while the proposed test was relying on using MCU that can be applied using a single connection. In addition, [11] ran an application of WebRTC video conferencing using the Licode-Erizo (MCU) over LAN (Local Area Network). Licode offers a client API with -Erizo that handles connections for virtual rooms and a server API for communication. But, without using the third party (Licode-Erizo) it would not be possible to run this application. On the other hand, as illustrated in [12] using MCU is very expensive, and [13] mentioned that MCU is costly and it can be rented from service providers just during a conference, although some video conferencing CODECs are able to support a specific number of multipoint (e.g. up to 4 users). Adding to that, [12] emphasizes that MCU consumes a significant amount of bandwidth.

Based on the various articles of the related work as shown above. It can be comprehended that signalling between browser-to-browser and server is not standardized in WebRTC [14][15].

IV. IMPLEMENTATION

A test-bed lab was created to implement a WebRTC video conferencing and over different networks such as Local Area Network and Wide Area Network. This implementation can be divided into two types: client application and signalling as described below:

- Client application

A test-bed lab was created to build WebRTC video conference application as clients and server. The connection was made through (wired and wifi) of Local Area Network & Wide Area Network.

The main HTML (web page) of this implementation has been programmed and set up for many features such as establishing a communication between two different users (PCs) using Chrome, pause audio/video, using full-screen, using volume slider and screenshot. Before a connection became established, the exchange of local and remote descriptions is performed (the audio and video media information). The following will describe how the signalling offer and answer are exchanging using the SDP (Session Description Protocol). First, Peer A sets the local description using the "setLocalDescription()" method and runs the RTCPeerConnection create Offer() method and then sends this session description to Peer B via the signalling server. Second, Peer B sets the description that Peer A sent as a remote description using the "setRemoteDescription()" method. Peer B sets the local description using the "setLocalDescription()" method and runs the RTCPeerConnection "createAnswer()" method and then runs the RTCPeerConnection "createAnswer()" method and then sends this session description to Peer A. When Peer A gets the Peer B session description, it sets that as the remote description with the "setRemoteDescription()" method. For example, once installed within an enterprise network, the server will enable WebRTC peer-to-peer communication sessions via (URL http://localhost:8888/).

- Signalling

In this study, the signalling implementation uses the WebSocket mechanism. In our signalling solution it made four types of the control messages: "initiator", "getting media stream", "peerChannel" and "exchange SDP". In the beginning, Peer A will send the "request" to the server and after getting the control message "getting access media stream," it will generate the user media from the server. Now Peer A waits until Peer B responses. Until it is activated, Peer B will send a response signalling message and when it finds out that it isn't the initiator, it will first get "peer Channel" and then "got access media stream". After that, both sides can start establishing a peer-to-peer connection. The offer and answer messages are transmitted in the SDP (Session Description Protocol) format, which bears information about media type, CODECs, RTP (Real Time Protocol), RTCP (Real Time Control Protocol) and all connected properties that can be used in the media session. Figure (2) demonstrates a communication between two peers, when Peer A sends a request to Peer B and Peer B return the response. In addition, figure (1) displays the architecture of the design that Peer A will send the SDP offer and ICE candidates to the server and the server will pass that to Peer B. Peer B will send the SDP answer and ICE candidates to the server and the server will in turn pass that to Peer A.

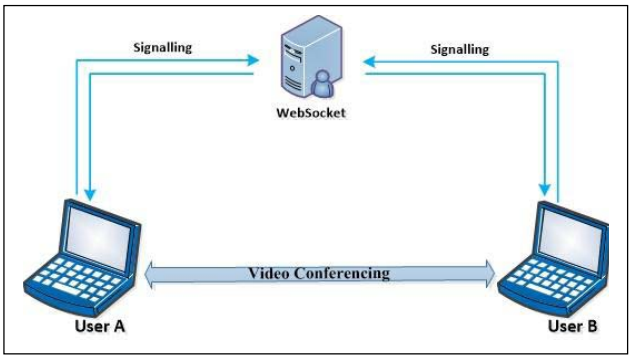

Figure (1), shows the architecture via WebSocket server

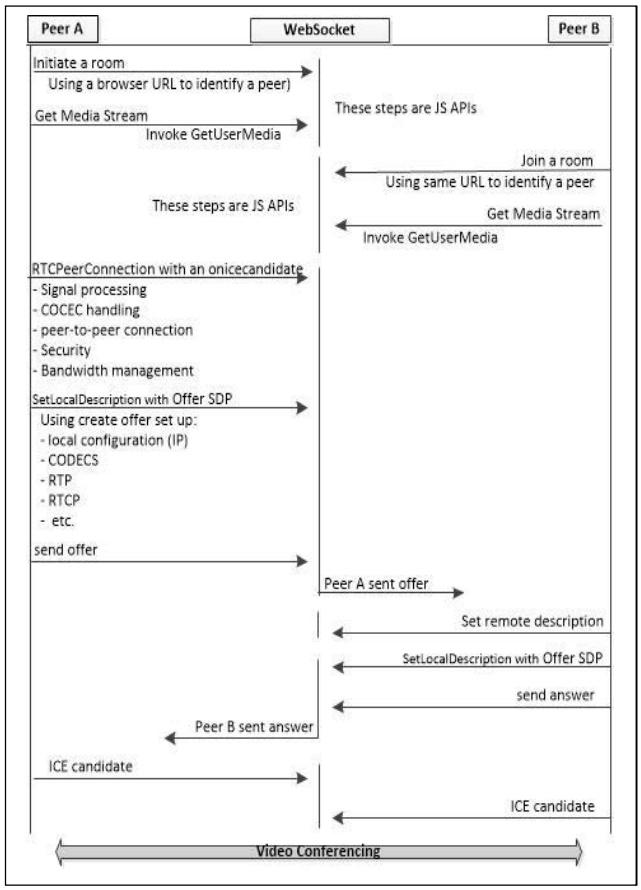

Figure (2), shows the data flow based on connection among two users and server

V. EVALUATION

Based on the necessity of CPU performance and bandwidth consumption, which can impact on video streaming [16], it has focused on their assessment which can be summarised as follows:

- Bandwidth Consumption

The results in table (1&2) show that the bandwidth consumption of audio in WebRTC, so the communication was achieved at various times such as one minute & three minutes via (Wired and WiFi) of LAN & WAN networks, the communication exhibited (53 - 54 kbit/s) bandwidth rate over LAN and (48 – 50 kbit/s) over WAN, it has also shown magnificent quality of voice and video.

Table (1), presents the bandwidth consumption between two peers via (Wired and WiFi) of LAN network. The unit of bandwidth is kbit/s.

Network	Over	Bandwidth	
		Time	Audio
Local Area Network	Wire	One minute	53.563
		Three minutes	54.208
	WiFi	One minute	54.743
		Three minutes	54.979

Table (2), Shows the bandwidth consumption between two peers via (Wired and WiFi) of WAN network. The unit of bandwidth is kbit/s.

Network	Over	Bandwidth	
		Time	Audio
Wide Area Network	Wire	One minute	50.719
		Three minutes	49.755
	WiFi	One minute	48.176
		Three minutes	48.062

- CPU Performance

It has a primary influence on WebRTC video conferencing. CPU handles a high load due to various sources sending and receiving the videos at the same time. The CPU limitations affect only the user with the reduced CPU usage [17]. The performance of the CPU was measured using task manager software of Windows 10 in peer side, which is both of client and server. The analysis is divided into four types as follows: (a) during one minute through (Wired & Wifi) of LAN network, (b) during one minute through (Wired & Wifi) of WAN network, (c) during three minutes through (Wired & Wifi) of LAN network and (d) during three minutes through (Wired & Wifi) of WAN network. As displayed in the charts below, the maximum average requirement of CPU speed is as follows: Over LAN (Wire) the CPU needs (13%), Over LAN (Wifi) the CPU needs (14%), Over WAN (Wire) the CPU needs (15%) and Over WAN (Wifi) the CPU needs (17%). The CPU exhibited a range of 13% to 17% as an average rate of need. The CPU performance can be shown in diagrams (1, 2, 3&4).

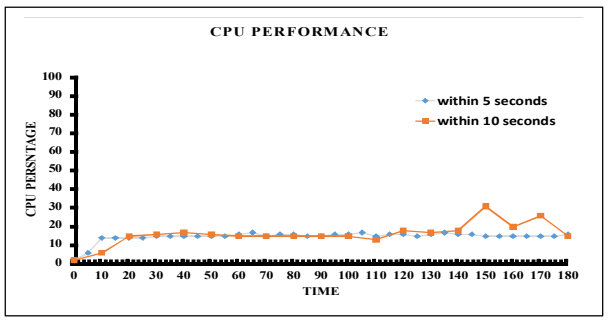

Diagram (1), shows the CPU performanceof peer A consumption over (wire) of LAN within THREE minutes

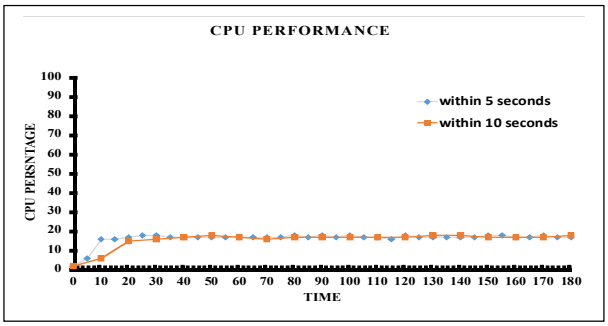

Diagram (2), displays the CPU performance of peer A use over (wire) of WAN within THREE minutes

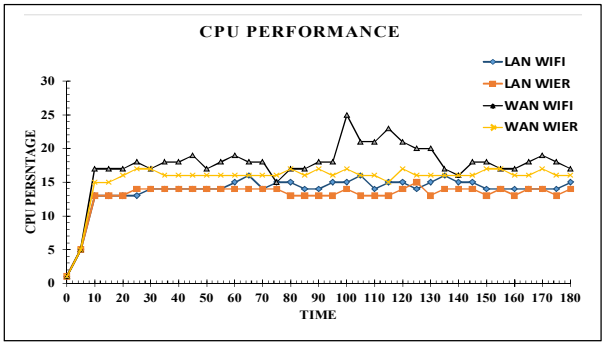

Diagram (3), indicates the CPU performance of peer A use over (Wired and WiFi) of LAN and WAN within THREE minutes

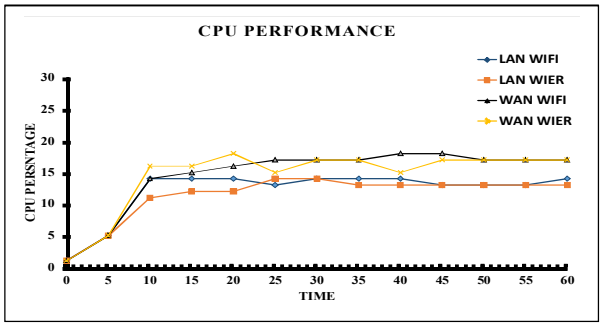

Diagram (4), illustrates the CPU performance of peer A use over (Wired and WiFi) of LAN and WAN within ONE minute

- Quality of Experience (QoE)

Actual users have participated in this implementation to give their individual opinions on the perceived user experience by the use of questionnaires. The quality of audio and video has been assessed, therefore they were excellent.

VI. CONCLUSION

The Web Real-Time Communication (WebRTC) was developed as a new standard for facilitating the RTC between users through different web browsers without any extra installation. However, there is a major challenge for implementing WebRTC, that is, the WebRTC does not specify the communication standard between different browsers, which prevents the WebRTC from functioning correctly due to the incompatibility problem with multi-browser connection. Addressing the aforementioned obstacle, this paper utilised the WebSocket protocol as a server to fully communicate between two different browsers. Besides, it created and implemented WebRTC video conferencing that can offer bi-directional communication, as well as using different networks such as (Wired and Wi-Fi) of LAN and WAN networks. A deep evaluation of the physical implementation was done over CPU performance, bandwidth consumption and QoE. In the future, will attempt to create a WebRTC signalling mechanism for unlimited peers using Socket.io and apply it over different topologies such as mesh and star. Additionally, will compare WebRTC video conferencing with the most common protocols in Voice over Internet Protocol (VoIP), such as SIP or IAX2 protocols.

ACKNOWLEDGMENT

This research was funded by the Ministry of Higher Education in the Republic of Iraq, according to the scholarship number (1469) in (03/04/2013) to sponsor the first author to pursue his PhD research.

REFERENCE

[1] C. Cola and H. Valean, “On multi-user web conference using WebRTC,” in *18th International Conference on System Theory, Control and Computing, ICSTCC*, pp. 430–433, 2014.

[2] M. Phankokkruad and P. Jaturawat, “An Evaluation of Technical Study and Performance for Real-Time Face Detection Using Web Real-Time Communication,”, no. I4ct, pp. 162–166, 2015.

[3] T. Sandholm, B. Magnusson, and B. A. Johnsson, “An on-demand WebRTC and IoT device tunneling service for hospitals,” in *Proceedings - International Conference on Future Internet of Things and Cloud, FiCloud*, pp. 53–60, 2014.

[4] M. Deshpande, “Integration of WebRTC with SIP – Current Trends,” *Int. J. Innov. Eng. Technol. Integr.*, vol. 6, no. 2, pp. 92–96, 2015.

[5] A. Albas and G. Auguets, “WebRTC,” Politècnica de Catalunya, 2016.

[6] I. T. Management, “WebRTC in the Enterprise,” 2016.

[7] M. S. Nasir and K. Saeed, “A Comparison of SIP with IAX an Efficient new IP Telephony Protocol A Comparison of SIP with IAX an Efficient new IP Telephony Protocol,” no. August, 2015.

[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*, no. May, pp. 1006–1009, 2015.

[9] K. Bissereth, B. B. L. Lim, and A. Shesh, “An Interactive Video Conferencing Module for e-Learning using WebRTC,” in *Internetional Conferences*, pp. 1–4, 2014.

[10] Ana Pol González, “DEFINITION OF A MENA OPINION SCORE FOR VP8 OVER REAL-TIME CONNECTIONS,” Universida de Vigo, 2017.

[11] and M. S. D. Vučić, L. Skorin-Kapov, “The impact of bandwidth limitations and video resolution size on QoE for WebRTC-based mobile multi-party video conferencing Faculty of Electrical Engineering and Computing , University of Zagreb,” in *5th ISCA/DEGA Workshop on Perceptual Quality of Systems*, pp. 59–63, 2016.

[12] K. Fai Ng, M. Yan Ching, Y. Liu, T. Cai, L. Li, and W. Chou, “A P2P-MCU Approach to Multi-Party Video Conference with WebRTC,” *Int. J. Futur. Comput. Commun.*, vol. 3, no. 5, pp. 319–324, 2014.

[13] 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].

[14] 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, 2016.

[15] G. Carullo, M. Tambasco, M. Di Mauro, and M. Longo, “A Performance Evaluation of WebRTC over LTE,” in *12th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, pp. 170–175, 2016.

[16] J. Nurminen, A. Meyn, and E. Jalonen, “P2P media streaming with HTML5 and WebRTC,” in *IEEE International …*, no. 4, pp. 1–2, 2013.

[17] L. O. D. Nedberg, “Quality of Experience of WebRTC based video communication Eirik Fosser,” Norwegian University of Science and Technology, 2016.