WebNSM: A Novel Scalable WebRTC Signalling Mechanism for Many-to-Many Video Conferencing

Naktal Moaid Edan
School of Science and Technology, The University of Northampton
Northampton, United Kingdom
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*— There is a strong focus on the use of Web Real-Time Communication (WebRTC) for many-to-many video conferencing, while the IETF working group has left the signalling issue on the application layer. The main aim of this paper is to create a novel scalable WebRTC signalling mechanism called WebNSM for many-to-many (bi-directional) video conferencing. WebNSM was designed for unlimited users over the mesh topology based on Socket.io (API) mechanism. A real implementation was achieved via LAN and WAN networks, including the evaluation of bandwidth consumption, CPU performance, memory usage, maximum links and RTPs calculation; and Quality of Experience (QoE). In addition, this application supplies video conferencing on different browsers without having to download additional software or user registration. The results present a novel signalling mechanism among various users, devices and networks to open one or multi rooms at the same time using the same server, determine room initiator to keep the session active even if the initiator or another peer leaves, sharing new user with current participants, etc. Moreover, this experiment highlights the limitations of CPU performance, bandwidth consumption and using mesh topology for WebRTC video conferencing.**

Keywords*— *The Real-Time Web Communication (WebRTC), Socket.IO signalling mechanism, Local Area Network (LAN), Wide Area Network (WAN), Quality of Experience (QoE), Mesh topology and a Web New Signalling Mechanism (WebNSM).

I. INTRODUCTION

Since the start of web applications, developers have worked towards diverse ways of getting full duplex communication between the server and the browser. Whether it is using Flash, Java and so on; all aims are for the same goal. Therefore, the Internet Engineering Task Force (IETF) and World Wide Web Consortium (W3C) developed a new standard named Web Real Time Communication (WebRTC) [1]. WebRTC is an open source and a collection of JavaScript APIs and standards [2]. JavaScript APIs are directly used to provide support for interactive communications using various kinds of data such as audio, video, etc. [3]. WebRTC offers several benefits such as no need for plug-ins, softphones, ease of use, cost reduction, no licensing and high-quality RTC (Real Time-Communication) application [4]. It has also been used from more than 1,000,000,000 endpoints [5]. On the other hand, W3C and IETF have not yet been agreed to a final signalling mechanism or protocol to test WebRTC [6]. Therefore, WebRTC cannot support the multi-browser communication mainly for a conference over participating browsers [7]; including communication between browser-to-browser and server is not standardised yet [1][3][8][9]. Signalling is considered as the main part of the application which has not yet been specified [10]. Thus, WebRTC requires a kind of signalling mechanism and a support of protocols to achieve a communication among different users [11]. Signalling is the heart of the peer detection that discovers peers and coordinates communication among them; it supports the establishing communication among users by exchanging data through channels [1]. Signalling connects the browser to a server and allows the other peers to communicate this server. Moreover, signalling supports the SDP (Session Description Protocol) for combining the network addresses and port numbers for the media exchange [12]. Different platforms are designed for WebRTC video chat such as Simple WebRTC and easyRTC; however, they have some limitations. For instance, some of them are not free of charge and the others use their own infrastructure to handle the service [5]. In addition, many implementations have been accomplished to create WebRTC video chat/calls but they have used XMLHttpRequest (XHR) or polling cycle which leads to waste of bandwidth and delay, while the browser keeps polling for data periodically and the server continues responding with an empty response even when no messages that are ready to be sent or received [13]. XHR (polling) is efficient with communication that does not need to full duplex approach, therefore it is used just for pushing updates from the server to the client [14]. Moreover, different developers attempted to use SIP (Session Initiation Protocol) with WebRTC to obtain video calls, but SIP still needed an installation and a software to such servers [15]. In addition, WebRTC requires protocols which are not yet embedded within existing SIP clients [11]. While the current real-time communication APIs in an application is more cost efficient and faster than developing a SIP client [16]. The combination of WebRTC functions with SIP platform requires some development using a new kind of integrated communication environment to enable multi media sessions [11]. Furthermore, SIP has a high bandwidth consumption and delays comparing with the other protocols such Inter-Asterisk eXchange2 (IAX2) [17].

According to the limitations of the current mechanisms such as XHR and SIP, as well as the necessity for creating a signalling mechanism to offer video conferencing for undefined users in WebRTC.

In this paper, WebNSM was created for video conferencing based on RTCPeerConnection (API) using socket.io mechanism to connect between each of the browsers. A Socket.io API that provides real-time bi-directional event-based communication between a client and a server was used [18]. Besides, RTCPeerConnection (API) is an array of URL objects which sends any ICE (Interactive Connectivity Establishment) candidates to the other peer, handles the video stream, starts offer/answer negotiation process, etc [19]. WebNSM presents a flexible signalling mechanism through Wired of LAN and WAN networks that are able to provide various characteristics as follows: (a) offering bi-directional video conferencing for different users and devices, (b) opening single or multi rooms at the same time using same server, (c) keeping or returning the previous state by rejoining the previous room, (d) determining room initiator, (e) keeping session active even if the initiator or another peer leaves, (f) sharing participants with all users, (g) joining an existing session or renegotiating new session, (h) stoping or removing self-streams, (i) stoping the remote stream, (j) skipping non-candidate events, etc. WebNSM is beneficial in that it can assist web developers making a decision in their choice of technologies, mechanisms and protocols when developing WebRTC supported applications. The primary objectives of this paper are to create a novel and scalable signalling mechanism for WebRTC video conferencing based on the Socket.io API, and for unlimited users using mesh topology in a physical implementation. Moreover, an evaluation of WebNSM performance, bandwidth consumption, CPU performance, memory usage, Quality of Experience (QoE) and maximum links and RTPs calculation. In-depth elaboration of this paper will help concerned users getting a factual prognosis of the advantages and disadvantages of using mesh topology. This illustration is beneficial for interested users who intend to use WebRTC video conferencing among different communications such as communication applications, e-learning, mHealth, monitoring, game, etc.

This paper is organised as follows, The reports on WebRTC related work is given in section II. The methodology along with implementation and analysis are presented in section III. Evaluation is explained in Section IV and Section V has the conclusion and future work.

II. Related work and some limitations

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

As mentioned in [20], signalling management has not yet been specified by WebRTC to allow the developer to modify, reuse existing protocols and permits them freedom to design their signalling to avoid redundancy and to increase compatibility with established technologies [15]. Moreover, an overview of WebRTC video conferencing architecture using MCU (Multipoint Conferencing Unit) was shown in [17]. However, this scenario does not discuss any signalling mechanism or protocol while the proposed test was relying on using MCU that can be applied using a single connection. Also, [21] ran an application of WebRTC video conferencing using the Licode-Erizo (MCU) and Samsung Galaxy for each participant. Licode offers a client API with -Erizo that handles connections for virtual rooms and a server API for communication. Nevertheless, without using the third party (Licode-Erizo) it cannot run this application. The test was achieved among three rooms each room consists of maximum three participants, as well as they have not presented anything about the signalling mechanism. On the other hand, as illustrated in [22], MCU is costly and it can be rented from service providers 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, [23] emphasised that MCU consumes a significant amount of bandwidth.

According to [24], evaluated the performance of WebRTC video calls using the node.js server, WebSocket protocol for the signalling and TURN servers. This evaluation was done over different topologies such as a mesh (using separate switches) and star (using MCU). On the other hand, the calls were established between three participants in each topology using a fake device and video sequence in VGA frame instead of employing a live camera. The media bit rate is set by the browser as 2Mbps maximum value. Besides, all calls were forced to stream through the TURN servers. Moreover, [15] designed and implemented a novel WebRTC signalling mechanism for chat messages using WebSocket via Node.JS cross-platform on the local host. The signalling of this application only supports a chat between two peers.

Based on the current works using the existing protocols and the various articles of the related work as shown above. The physical implementation in this paper gave a real elaboration that helps to overcome most of the existing limitations at the current suggestions and implementations, for instance, it extended the number of participants in mesh topology, to be more than 8 peers, analysed CPU performance, bandwidth consumption, QoE, etc. Moreover, it mentions the limitations of using mesh topology and offers a flexible signalling mechanism that is applied to a full duplex for many-to-many video conferencing.

III. Methodology, implementation and analysis

A. Methodology

This application used Firefox to test a client side, and used Apache HTTP server as a web server and Wireshark analyser. Moreover, fifteen computers were used, ten PCs Xeon (CPU E3-1246 v3 & 16 GB RAM), three PCs (CPU Core i7 & 4-12 GB RAM) and two Laptops (core i5 & 8 GB RAM) connected through Wired of LAN and WAN networks, cameras and microphones.

B. Implementation

A test-bed lab was created to achieve many-to-many video conferencing. The experiment environment can be divided into two types, setting up a browser (to initiate, join or leave the room) and creating WebNSM as described below:

1) Setting up a Browser to Initiate, Join or leave the Room

The main web page of this application uses Firefox and has many features such as open room, mute-audio/video, use full-screen, use volume slider and screenshot. In the beginning, to open a room the initiator needs to specify "user-id", which can be a Boolean random string, numbers or chosen manually. User-id is the most important thing for initiating and joining the room. Therefore, all users should have identical "user-id" to enter the room, otherwise, they cannot prove themselves. In this application, communication has one initiator who specifies "user-id" and different participants who already know the user-id that the initiator will use. When the room is opened, it will present arbitrary audio and video "MediaStream" including multiple tracks. A "MediaStream" can be obtained using "navigator.getUserMedia" method which can be invoked when only the first participant is found and then a web browser will pop out of an HTTP prompt and request permission to access the camera and microphone to capture peer's screen. Once the permission is granted; a camera will start streaming; and then the application would be ready for other peers to join. Moreover, a participant needs to type the same "user-id" to enter the room. Otherwise, the participant cannot confirm to WebNSM, using a different user-id leads to opening a new room. Additionally, a participant needs to fetch "MediaStream", invokes "getUserMedia" and shares camera and microphone as well. Finally, a participant has joined the room and communicates with the existed peer(s). These steps of opening/entering the room will apply to every peer.

To leave the room, a peer needs to refresh or close the browser web page, as well as stopping the streaming of their own camera/microphone without influencing the communication of the rest. The following algorithm (1) shows presents getMediaElement.js and getAudioElement.js.

```
function getMediaElement(mediaElement, config)
    muteAudio.onclick = function()
    muteVideo.onclick = function()
    takeSnapshot.onclick = function()
    stop.onclick = function()
    zoom.onclick = function()
    function launchFullscreen(element)
    function screenStateChange(e)
    function adjustControls()
    mediaElementContainer.toggle = function(clasName)
// getAudioElement.js
function getAudioElement(mediaElement, config)
    muteAudio.onclick = function() {
    stop.onclick = function() {
```

Algorithm (1): demonstrates some functions for getMediaElement.js and getAudioElement.js

2) WebNSM (A Novel Scalable Signalling Mechanism)

This signalling mechanism was created using Socket.io (API) bi-directional mechanism. WebNSM works based on two concepts, offerer and answerer. The offerer is a peer who initiates the WebRTC session (room) to connect another peer. In contrast, the answerer is asked for the connection from the offerer. WebNSM has a nobility feature to enable many peers to join one room or multiple rooms at the same time and using one-to-one or/and many-to-many bi-directional video conferencing, as well as inappropriating users, and it also does n ot have the ability to receive a new session event. The offerer is assumed to know the answerer's user-id and then requests a connection through WebNSM. When the initiator opens the browser (main page), WebNSM will be ready to support the offerer to detect a room presence. WebNSM will send the request from the offerer to the initiator (answerer) for the availability, including SDP offer to receive audio and video. The answerer will receive a request and will validate the "user-id" to decide either accept or reject the request. If the request is accepted, the answerer will send a confirmation of the availability as "room is active" with the SDP constraints to receive audio and video. Now the answerer and offerer are able to respond by Datagram Transport Layer Security (DTLS) and Secure Real-time Transport Protocol (SRTP) to allow the exchange of the cryptographic parameters and conclude keying material. They both configure the Real Time Communication (RTC) packets transported.

The answerer gets remote stream-id and uses "getLocalDescription" to create an offer and RTCPeerConnection. In RTCPeerConnection, a "createDataChannel" method is used to create an "RTCDataChannel" object. When an "RTCDataChannel" on the offerer's side is generated, the offerer invokes "createOffer" of RTCPeerConnection, thereby enabling "createOffer" to return an offerer's Session Description Protocol (SDP) message. To make a connection, based on socket.io (API) the offerer first generates the SDP-offer message by setting the following: session name (s) & information (i), bandwidth information (b), using the period audio and video CODECS by map the Real Time Protocol (RTP), Real-Time Control Protocol (RTCP), etc. Moreover, the offerer changes the state of ICE connection and ICE gathering to "new", also a signalling state to "have a local offer" and then needs to send the message to a certain answerer through WebNSM. Additionally, both the offerer and answerer change a signalling state to "stable" to realise that there is no offer/answer exchange in progress. The ICE connection presents the relationship of peers with ICE state such "is connected", as well as it can configure the user-id, new number and user's information of the offerer and answerer even if they are not connected. Once the "SDP-offer" message reaches the answerer, the answerer also initiates its RTCPeerConnection instance to accept the request. The answerer uses the "SDP-offer" into its RTCPeerConnection to creates an "SDP-answer". WebNSM handles this message over to the offerer's side. After two peers exchange SDP-offer/answer, they can create their session. The other peers can join the session based on similar steps. Chart (1), presents a simple example of offer and answer.

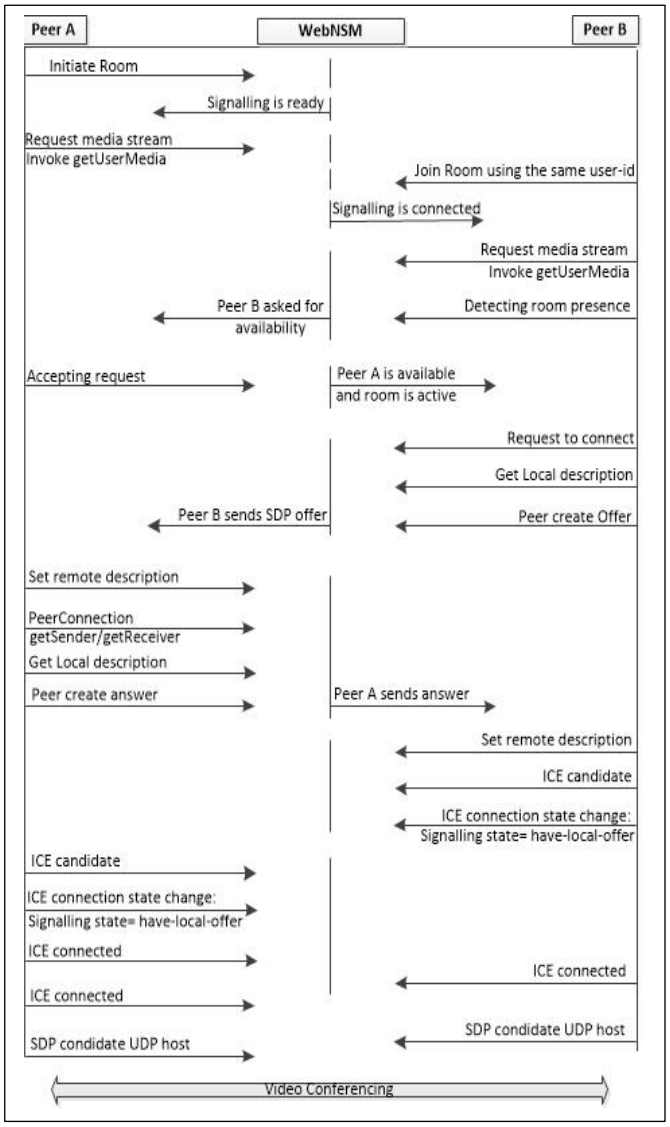

Chart (1), illustrates WebNSM between two peers

C. Analysis

This implementation was achieved among fifteen peers (PCs). The experiment took place during three to four minutes over each communication via Local Area Network (LAN) and Wide Area Network (WAN). In addition, the implantation was repeated twice, once using Wireshark software and another time without using Wireshark in order to validate the CPU loads. The analysis can be described as follows:

1) WebNSM

Based on the network analysis at inspect element of Firefox at the real-time communication, WebNSM demonstrates a productive achievement. While its performance was analysed individually among two to fifteen users based on signalling delay for two concepts, the first was based on the signalling delay to get ready and the second relies on sending a request and receiving a response between two peers. Thus, WebNSM consumes 161 (ms) as a minimum consumption and 180 (ms) as a maximum consumption to get ready, it also consumes 106 (ms) as a minimum use and 110 (ms) as a maximum consumption to send a request and receive a response. The delay was changeable based on CPU capability, the speed of web server and the kind of network. The mean time was calculated so WebNSM expends 171 (ms) to be ready and consumes 112 (ms) to send a request and receive a response. Diagram (1) elaborates more on that. WebNSM is able to setup, establish a video conferencing and end a communication simultaneously among all participants. Moreover, the CPUs and bandwidth consumption have not effected the presentation of WebNSM. On the other hand, the quality of audio and video was affected by the CPU and the bandwidth.

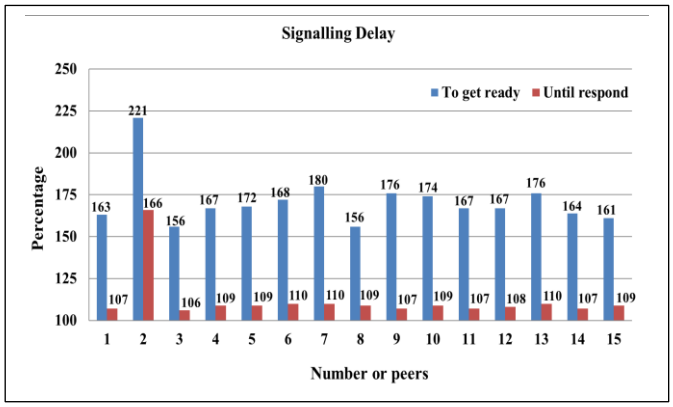

Diagram (1), presents the delay in WebNSM among fifteen peers over LAN and WAN networks. Unite is milliseconds.

2) Quality of Experiment (QoE)

This research was applied by actual users and collected their individual feedbacks on the perceived user experience by the use of questionnaires:

a) Video conferencing via LAN network

Without using Wireshark software, the quality of audio and video by individual tests among two to ten peers were excellent, also between eleven to twelve peers were acceptable while some peers showed some interruption. On the other hand, when the number of peers increased to be over thirteen users, it displayed an unacceptable quality over both audio and video as demonstrated in table (1).

Table (1), shown the quality of audio and video without Wireshark among fifteen peers over wired of LAN network

Monitor	Sequence	Number of peer	Duration	Quality of audio	Quality of video
Without Wireshark	1.	2–10	3-4 m	Excellent	Excellent
	2.	11-12	3-4 m	Acceptable	Acceptable
	3.	13-15	3-4 m	Unacceptable	Unacceptable

When using Wireshark, the quality of audio and video among two to eight peers were excellent and between nine to eleven peers were acceptable. But, some peers led to disorders in both audio and video. When the number of peers increased to be more than eleven, it offered an unacceptable quality.

b) Video conferencing via WAN network

Using Wireshark, the quality of audio and video among two to seven peers was excellent and between eight to ten peers were acceptable, but some peers presented an echo and high delay over video. However, the quality with more than ten peers was unacceptable and showed a frozen image all most. As displayed in the table (2).

Table (2), demonstrated the results of audio and video using Wireshark among fifteen peers over WAN network

Monitor	Sequence	Number of peer	Duration	Quality of audio	Quality of video
With Wireshark	1.	2-7	3-4 m	Excellent	Excellent
	2.	8-10	3-4 m	Acceptable	Acceptable
	3.	11-15	3-4 m	Unacceptable	Unacceptable

3) CPU

It has a primary influence on WebRTC video conferencing especially using mesh topology. Mesh topology uses many links among users to transfer data. It handles a high load due to various sources sending and receiving the videos at the same time, another reason why it has a highload is the process of encoding and decoding each link in the mesh typology. this load will impact the CPU performance. Consequently, as much as the increase in the number of users, the number of links (L) will increase as well. The CPU limitations affect only the user with the reduced CPU usage [4]. On the other hand, memory usage was not effected on the quality of the video and audio, while the communication was in a real time. As presented in the diagram (2).

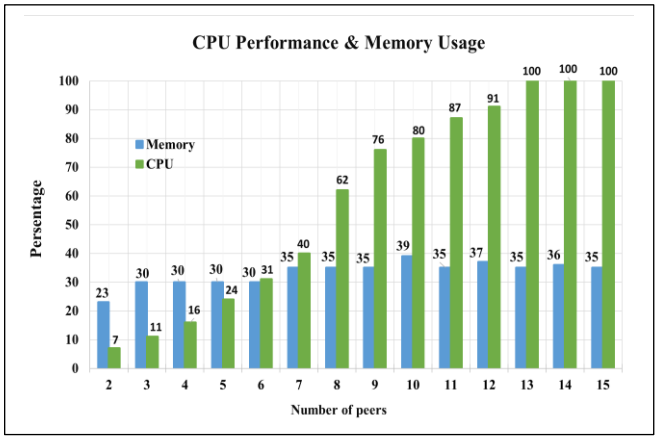

Diagram (2), displays CPU and memory usage among fourteen peers over LAN network.

4) Bandwidth

WebRTC supports various codecs such as G.711, PCMA, PCMU, Opus, V8 and so on [25]. In WebNSM, the SDP sends different CODECS, such as G.711, G.722, Opus, PCMA and PCMU for audio and VP8, VP9 and H.264 for video. The SDP answer will choose an appropriate codec based on the engine. In this implementation, Firefox that relies on Opus as an audio codec and VP8 as a video codec was used. WebRTC defaults its codecs to make use of their superior quality in comparison to other codecs including their adaptability when changes in the bandwidth occur [25]. Bandwidth consumption was measured and analysed to find the following:

- Each peer needs a minimum of 1Mb/s bandwidth for each RTP on the video via LAN and WAN networks
- Each peer needs a minimum of 58 - 63 kb/s bandwidth for each RTP on the audio via LAN and WAN networks

Bandwidth consumption can lead to a bottleneck on the client, which affects the quality of video and audio. Diagrams (3&4) give more clarifications.

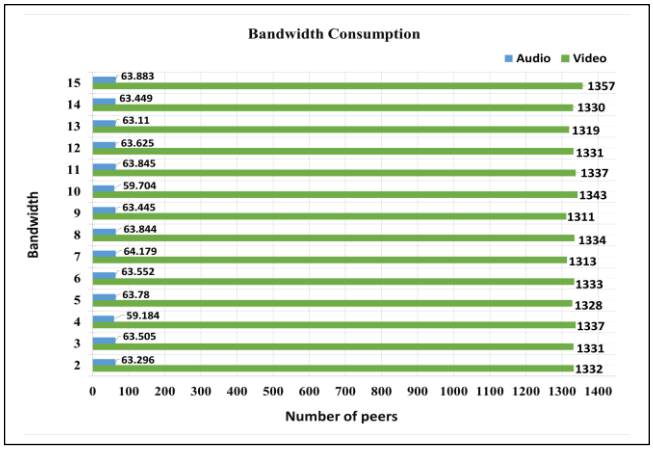

Diagram (3), demonstrates bandwidth consumption between LAN network among fourteen peers. The unit is kbit/s

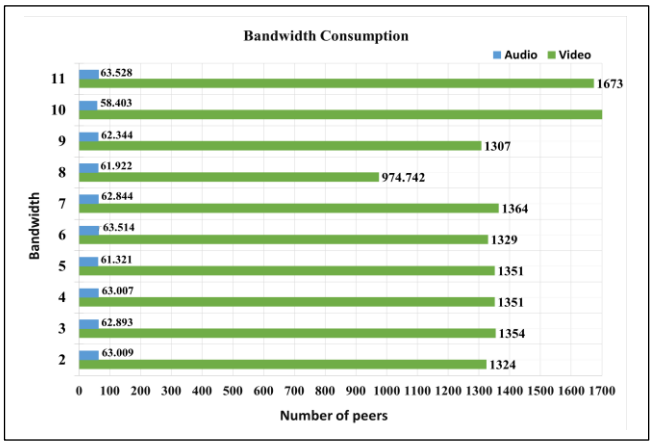

Diagram (4), shows bandwidth consumption between WAN network among ten peers. The unit is kbit/s

5) Mesh topology

In a mesh typology, any conference member can invite another user to participate/leave at any time without affecting the remaining participants. Many links can be created among peers to transfer data, and all peers connect between themselves to transmit data from different devices simultaneously. Figure (2) presents the architecture of mesh topology and diagram (5) shows the number of links among fifteen users.

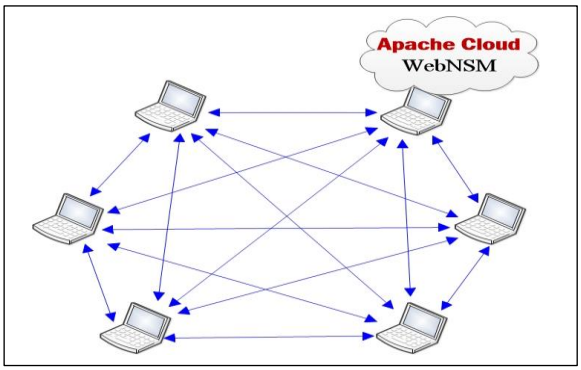

Figure (2), indicates the architecture of mesh topology

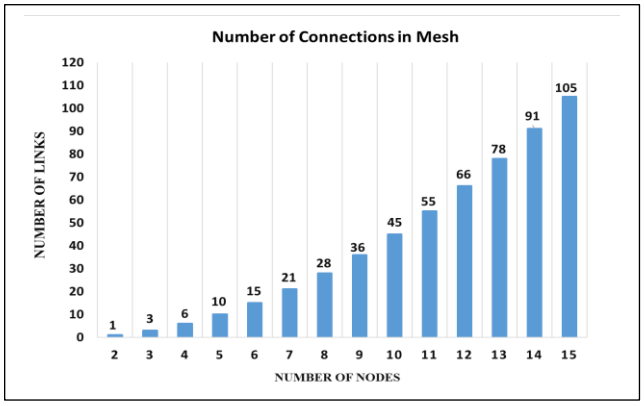

Diagram (5), illustrates the number of connections among fourteen participants in mesh topology

As shown in diagram (6), each communication between peers needs to have a separated RTP (Real Time Protocol) for the audio and video and then transmit them using different UDP (User Datagram Protocol) port. Therefore, each peer requires at least four RTPs as follows:

- One RTP port for outgoing video
- One RTP port for outgoing audio
- One RTP port for incoming video
- One RTP port for incoming audio

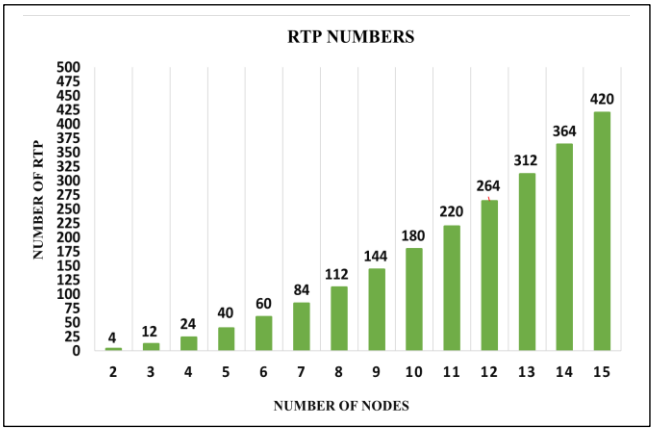

Diagram (6), displays the number of RTP among fourteen participants in mesh topology

IV. Evaluation

It has been proved that WebNSM is able to setup, establish a session and close a communication among an indefinite number of peers over LAN or WAN networks. WebNSM is also able to open one/multiple rooms to offer bi-directional video conferencing, thus keeping the session productive even if any peer leaves, controls self/remote streams, overshoots non-candidates and so on. It is also not affected by the limitations of CPU, bandwidth and memory. However, the quality of audio and video is affected by the limitations of CPU and bandwidth. The performance of CPU and bandwidth consumption has major issues in audio and video conferencing, while video conferencing requests the processor for decoding, encoding and providing the video and audio at the same time. This can be defined as CPU stress and it depends on different elements e.g. used codec's, quality of the audio, video and their respective sizes. Moreover, the variety of bandwidth speed among the different users can impact the quality of video and audio. Mesh topology is the most complicated topology since it requests a high CPU and high bandwidth speed. For instance, when a user uses CPU core Xeon, it cannot perform as another user, which uses CPU core i5, etc. In other words, as much as the CPU core is high, it will lead to better communication and allow more peers to join. According to the referenced restrictions, it can be emphasised that the CPU plays a leading role in communication and number of peers over mesh topology, as long a bandwidth does a key role in the quality of audio and video. In another meaning, the available CPUs at the used computers (e.g. Xeon) are not able to encode, decode, send and receive video conferencing at the same time over more than 55 links via mesh topology in real implementation. The quality of experience (QoE) verifies that this testbed environment works correctly and that it can be used to conduct more extensive experiments on user expertise in the future while having high core CPUs.

V. Conclusion and Future Work

Users in WebRTC need a signalling mechanism to set up a session, coordinate a communication and connect with each other. In this paper, a novel scalable WebRTC signalling mechanism named (WebNSM) has been created and implemented, which can offer bi-directional video conferencing for unlimited users, as well as using mesh topology over different networks such as LAN and WAN networks. WebNSM guarantees a different performance for providing a method to manage the routeing by WebRTC characteristics. In addition, a deep evaluation of the physical implementation was done over CPU performance, memory usage, WebNSM performance, QoE, mesh topology, etc. Nevertheless, using mesh has impacted the quality of audio and video due to the bandwidth and CPU consumptions in spite of the fact that WebNSM has not been affected. Therefore it takes an average of 112 (ms) as a mean time of delay from the time an offer is sent until returning a response, even when the network is congested. This application has calculated the number of links and RTP to comprehend the number of connections in the mesh. Additionally, this signalling mechanism can support unlimited number of peers while having high core CPUs, that it can be supplied in

various applications, such as conferencing among users, e-Learning among teachers and students, telemedicine among patients, doctors or technicians, etc. In the future: an implementation of WebNSM based on simplex (unidirectional) and bi-directional topologies; also attempt to use high core of CPUs to evaluate the performance.

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.

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, 2016.

[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] 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.

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

[5] Sam Dutton, "Getting Started with WebRTC," 2014.

[6] 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.

[7] C. Y. Chiang, Y. L. Chen, P. S. Tsai, and S. M. Yuan, "A video conferencing system based on WebRTC for seniors," in *Proceedings - 1st International Conference on Trustworthy Systems and Their Applications, TSA*, pp. 51–56, 2014.

[8] I. T. Management, "WebRTC in the Enterprise," 2016.

[9] H. Rahaman, "A Survey on Real-Time Communication for Web," vol. III, no. Vii, pp. 39–45, 2015.

[10] M. Schindler, C. Von Harscher, J. Kinzig, and S. Alekseev, "Evaluating Framework for Monitoring and Analyzing WebRTC Peer-to-Peer Applications,", no. Inc, pp. 171–175, 2016.

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

[12] Schahin Rajab, "Comparing different network topologies for WebRTC conferencing," 2015.

[13] S. a S. T. Miner, "Getting Started with," Packt>, pp. 1–41, 2013.

[14] R. Rai, *Socket. IO Real-time Web Application Development*. BIRMINGHAM - MUMBAI: PACKT, 2013.

[15] 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.

[16] C. Notice and A. Notice, "WebRTC to complement IP Communication Services," 2016.

[17] N. M. Edan, A. Al-Sherbaz, S. Turner, and S. Ajit, "Performance evaluation of QoS using SIP & IAX2 VVoIP protocols with CODECS," in *Proceedings of SAI Computing Conference, SAI*, pp. 631–636, 2016.

[18] M. Grinberg, "socketio Documentation," 2016.

[19] D. C. B. Adam Bergkvist, B. A. Cullen Jennings, Anant Narayanan, and B. and Taylor, "Real-time Communication Between Browsers," *W3C*, 2017. [Online]. Available: https://w3c.github.io/webrtc-pc/. [Accessed: 30-Aug-2017].

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

[21] 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.

[22] 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].

[23] 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.

[24] V. Singh, A. A. Lozano, and J. Ott, "Performance analysis of receive-side real-time congestion control for WebRTC," in *20th International Packet Video Workshop*, pp. 1–8,2013.

[25] A. Sandoval Rosas and J. L. Alejos Martínez, "Videoconference System Based on WebRTC With Access to the PSTN," 2016.