View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by NECTAR

International Journal of Electrical and Computer Engineering (IJECE)
Vol. 8, No. 1, February 2018, pp. 390~399
ISSN: 2088-8708, DOI: 10.11591/ijece.v8i1.pp390-399

Design and Implement a Hybrid WebRTC Signalling Mechanism for Unidirectional & Bi-directional Video Conferencing

Naktal Edan[1], Ali Al-Sherbaz[2], Scott Turner[3]
[1,3]School of Science and Technology, Northampton University, Northampton, United Kingdom
[2]College of Computers Sciences and Mathematics, Mosul University, Mosul, Iraq

Article Info

Article history:

Received Sep 27, 2017
Revised Nov 30, 2017
Accepted Dec 14, 2017

Keyword:

Local area network (LAN)
Mesh topology
Quality of experience (QoE)
Socket.IO
The real-time web communication (WebRTC)
Web new signalling mechanism (WebNSM)
Wide area network (WAN)

ABSTRACT

WebRTC (Web Real-Time Communication) is a technology that enables browser-to-browser communication. Therefore, a signalling mechanism must be negotiated to create a connection between peers. The main aim of this paper is to create and implement a WebRTC hybrid signalling mechanism named (WebNSM) for video conferencing based on the Socket.io (API) mechanism and Firefox. WebNSM was designed over a combination of different topologies, such as simplex, star and mesh. Therefore it offers several communications at the same time as one-to-one (unidirectional/bidirectional), one-to-many (unidirectional) and many-to-many (bi-directional) without any downloading or installation. In this paper, WebRTC video conferencing was accomplished via LAN and WAN networks, including the evaluation of resources in WebRTC like bandwidth consumption, CPU performance, memory usage, Quality of Experience (QoE) and maximum links and RTPs calculation. This paper presents a novel signalling mechanism among different users, devices and networks to offer multi-party video conferencing using various topologies at the same time, as well as other typical features such as using the same server, determining room initiator, keeping the communication active even if the initiator or another peer leaves, etc. This scenario highlights the limitations of resources and the use of different topologies for WebRTC video conferencing.

Copyright © 2018 Institute of Advanced Engineering and Science.
All rights reserved.

Corresponding Author:

Naktal Edan,
School of Science and Technology,
Northampton University,
Avenue Campus, St George's Avenue, NN2 6JD, Northampton, United Kingdom.
Email: naktal.edan@northampton.ac.uk

1. INTRODUCTION

WebRTC (Web Real-Time Communication) was developed as a standard by the World Wide Web Consortium (W3C) and Internet Engineering Task Force (IETF) [1]. It is an open source and a collection of protocols and standards [2]. WebRTC allows the transportation of audio, video and data. Also, it does not need plug-ins, licensing, downloads and so on [3]. It is a technology that consists of three principal components [4]: getUserMedia: allows a web browser to access the camera and microphone and to capture media, RTCPeerConnection: manages the peer-to-peer connection and RTCDataChennel: allows browsers to share arbitrary data. On the other hand, WebRTC does not specify any particular signalling mechanism or protocol between the client and the server [5]. Moreover, it does not support the multi-browser communication essential for conferencing over participating browsers [6]. Including, the client-server architecture that does not seem to be a feasible solution [7]. Therefore, choosing the suitable network topology in the architectural design of the WebRTC application is considered as one of the most potential

problems. Thus, it must select an architecture for the application while dealing with a multiparty of audio/video call in WebRTC [8]. A signalling mechanism is the core of peer detection that coordinates the communication between users; it starts exchanging media and supports the establishing communication among users [1]. Signalling connects the browser to a server and permits the participants to access this server. Many experiments have been achieved to offer video calls in WebRTC. Therefore, some of them are used XMLHttpRequest (XHR/polling). However, using XHR leads to waste of bandwidth and delay, as long as the browser keeps polling for data regularly and the server continues responding even when no messages can be sent or received [9]. XHR is active with communication that does not need to full duplex approach [10]. In addition, several developers used SIP (Session Initiation Protocol) with WebRTC to obtain video calls, nevertheless SIP still needs software such servers and installation [11]. Besides, the current real-time communication APIs in an application is more cost efficient and faster than developing a SIP client [12]. Furthermore, SIP has a high bandwidth consumption and delays as compared with other protocols such as Inter-Asterisk eXchange2 (IAX2) [13]. In this paper, WebNSM was created for video conferencing based on RTCPeerConnection (API) using socket.io mechanism to connect between each of the browsers. Socket.io (API) offeres real-time bi-directional communication between a client and a server [14]. RTCPeerConnection (API) is an array of URL objects that send any ICE (Interactive Connectivity Establishment) candidates to the other peer, handles the video stream, and starts offer/answer negotiation process, etc [15]. WebNSM can provide hybrid characteristics as follows: (a) one-to-one (sample) bi-directional video conferencing, (b) one-to-one (sample) unidirectional video conferencing, (c) one-to-many (star) inidirectional video conferencing, (d) many-to-many (mesh) bi-directional video conferencing, (e) provides two kinds of communications, so each peer is free to be as a broadcaster or viewer, (f) determine room initiator, (g) keep a session productive even another participant leaves, (h) participants are able to share with all users, (i) join existing session, (j) stop self-streams and (k) sharing new user with current participants. Furthermore, WebNSM is useful to be used for various communications. For example, m-Health (many doctors can communicate many technicians and patients), e-learning (many teachers can communicate many students and many students can communicate others), communication applications, etc. In addition, it gives a user a full flexibility to use appropriate topology according to its resources. The essential objectives of this paper are to create a hybrid signalling mechanism to serve different topologies at the same time. In addition to designing and implementing a WebRTC video conferencing for many users, including an evaluation of signalling performance, bandwidth consumption, CPU performance, memory usage, Quality of Experience (QoE), using mesh topology (full duplex), star topology (simplex/unidirectional) and calculating the maximum links and RTP (Real Time Protocol).

This paper is organised and outlined as follows, Section 2 reports on survey WebRTC related work. In section 3, the methodology of the paper is explained along with implementation and analysis. Section 4 discusses the evaluation. Finally, Section 5 has the conclusion and future work.

2. RELATED WORK

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 [16], 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 [11]. Moreover, an overview of WebRTC video conferencing architecture using MCU (Multipoint Conferencing Unit) was shown in [17], including a demonstration of some challenges. 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, [17] 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 [18], using MCU is very expensive, and [19] mentioned that MCU is costly and it can be rented from service providers during a conference, although some video conferencing CODECs are able to support up to 4 users. Adding to that, [18] emphasised that MCU consumes a significant amount of bandwidth. According to [20], implemented REST APIs (Representation State Transfer) interoperating with SIP (Session Initiation Protocol) over WebSocket protocol to control the signalling message exchange for the audio/video call via Chrome. However, the signalling should be supported by a central component (named REST service) to exchange messages and establish media channel, besides the communication had 5 seconds

in delay and was done between only two browsers. Additionally, [21] 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 frame instead of employing a live camera. Besides, all calls were forced to stream through the TURN servers. Moreover, [11] 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.

3. METHODOLOGY, IMPLEMENTATION AND ANALYSIS

3.1. Methodology

Thirty computers were used as seventeen PCs (CPU Xeon & 16 GB RAM), three Laptops (core i5 & 4-8 GB RAM), ten PCs (CPU Core i5 and i7 with 4-12 GB RAM) were connected through Wired of Local Area Network and Wide Area Network, Logitech cameras and microphones.

3.2. Implementation

A test-bed lab was created to implement a hybrid signalling mechanism in real- time implementation for video conferencing. Therefore, several methods and APIs have been embedded to be used coherently. This implementation can be divided into following:

3.2.1. Setup a Browser Web Page

The main HTML (web page) of this experiment was programmed using JavaScript and Firefox to set up many features, such as opening room, mute-audio/video, using full-screen, using volume slider and screenshot. In the beginning, to open a room there must always be a room initiator while the participants are free to select "As Viewer" to watch and listen to the broadcaster or select "As Broadcaster" to set up bi-directional video conferencing, as well as the communication can include both as broadcaster and viewer to stream and view the video. All peers do not need to specify "user-id" since they are using the same URL as "user-id" to access the main page. Otherwise, they cannot join the room. In this application, communication has one initiator and different peers as viewers and broadcasters. When the room is opened, it will arbitrarily audio and video to present MediaStream, which can be obtained using navigator.getUserMedia() method to create a synchronised video and audio. After getUserMedia, a web browser will request permission to access the camera and microphone to capture peer's screen. A camera will start streaming when the permission is given; now the application is ready for other peers to join the room. On the other hand, when peers would like to be as viewers they do not need to invoke their camera and microphone, while they will only receive videos. These steps of opening/joining the room applies to every peer, as well as stopping the streaming of their camera/microphone without influencing on the rest. Figure 1 shows the main page and the options.

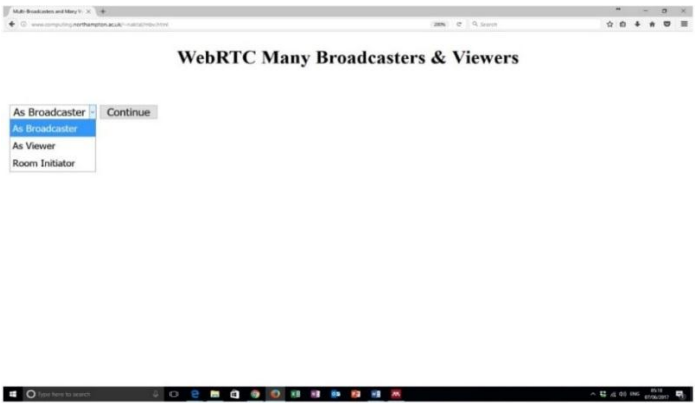

Figure 1. Shown the main web page using Firefox

3.2.2. WebNSM (A Hybrid Signalling Mechanism)

This signalling must occur before a Peer-to-Peer (P2P) connection can be occurred [22]. WebNSM was created using RTCPeerConnection API and socket.io (API) mechanism for an instant handshake.

Therefore, WebNSM must be carried out before streaming can begin between peers. It relies on offer and answers negotiation process to describe the SDP (Session Description Protocol) of the session. The offerer is a peer who initiates the session to connect other peers. In contrast, the answerer is asked for connection from the offerer. The offerer is assumed to know the answerer's URL and then requests a connection through WebNSM. When the initiator opens the main room, WebNSM will be ready to support any offerer and detect a room presence. Thus, several functions and steps have been employed to create it. First of all, it should transmit the data as a String and setup a default channel passed through constructor using "connection.channel = channel || RMCDefaultChannel". Additionally, it connects with a signalling channel when only the first participant is found using invoke "getUserMedia" then initRTCMultiSession function. WebNSM was built to accomplish many characteristics, such as determining the room initiator "connection.initiator = true", allowing a single user to join a room "connection.join = joinSession", keeping a session active even if the initiator leaves (clone data from initial moderator to the second initiator and make sure that if second leaves. The control is shifted to a third person if the initiator wants to close an entire session then shifts the initiation control to another user), hearing new user with existing participants on New Participant (response)'', participants are shared with a single user or with all users, if the initiator disconnects sockets, participants should also disconnect, close the entire session, reject user-id, disconnect for all, open private socket that is used to receive offer-sdp "newPrivateSocket" and ask other users to create offer-sdp and function PeerConnection. They also utilise RTC (Real Time Connection) to send data "connection.send = function(data, _channel)", initialize "RTCMultiSession" which is the backbone object. The custom devices are selected and screen_constraints, such as screen.width, screen.height. Participants also check if the screen-capturing extension is installed. When a stream is stopped, it must be removed from "attachStreams" array to allow re-capturing of the screen, if the muted stream is negotiated, audio/video are fired earlier than screen, stop local stream ''if (response.stopped)'', stop remote stream ''if (response.promptStreamStop'', create an offer SDP using "createOffer()" function, create answer SDP using "createAnswer()" function, createDescription() function, getBrowserInfo() function, construct a new RTCPeerConnection, trigger the stun server request, match just the IP address, remove duplicates, listen for candidate events and etc.

To establish a peer-to-peer connection, both clients need to create an RTCPeerConnection object. Then, each peer needs to obtain their Session Description, an object that indicates what kind of data they want to send to the other client through the connection and what they can do by built-in methods of the RTCPeerConnection object. Thus, the offerer will send the request to the answerer for the availability, including SDP offer to receive audio and video. The answerer (initiator/broadcaster) will receive the request and sends a confirmation of the availability as "room is active" with the SDP constraints to receive audio and video. The offerer gets the remote stream and creates an offer using "getLocalDescription" with RTCPeerConnection. Additionally, the offerer creates DataChannel method which is added to the RTCPeerConnection 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 SDP message. The offerer enables the SDP-offer message by setting various information and send them through WebNSM. For instance, bandwidth information, using the period audio and video codecs, etc. Additionally, both the offerer and answerer change WebNSM state to "stable", to realise that there is no offer/answer exchange in progress. Once the "SDP-offer" message reaches the answerer through WebNSM, the answerer also initiates its RTCPeerConnection instance to accept the request. The answerer uses the "SDP-offer" into its RTCPeerConnection to create an "SDP-answer" and then forward it to the offrer. Also, the two clients need to exchange information about communication methods that they can use to reach each other. These communication methods are known as ICE Candidates and they will be exchanged through the WebNSM. Now the answerer and offerer are able to respond and they both configure the Real Time Communication (RTC) packets transported. After two peers exchange SDP-offer/answer and ICE candidates, they can create their session. The answerer and offerer "add SDP" to candidate UDP by the host IP for both of them. The other participants can join the session based on similar steps.

According to a communication as viewers, when an initiator is active for streaming, a peer is able to accede the room as a viewer after detecting a room presence using WebNSM. WebNSM sends a notification to the initiator that " a participant has asked for availability and the target has no stream". In other words, it is a unidirectional video conferencing from an initiator to a viewer. An initiator receives a request and sends a confirmation of the availability as "room is active" with the SDP constraints. Thus, an initiator has started broadcasting the audio and video to the viewer. In contrast, if there are other broadcasters, a viewer will communicate all of them, so the viewer can communicate all broadcasters by receiving their audio and video at the same time. In addition, a session can be active even if any broadcaster leaves; also all viewers communicate all broadcasters at the same time. Figure 2, illustrated the signalling follows between broadcasters

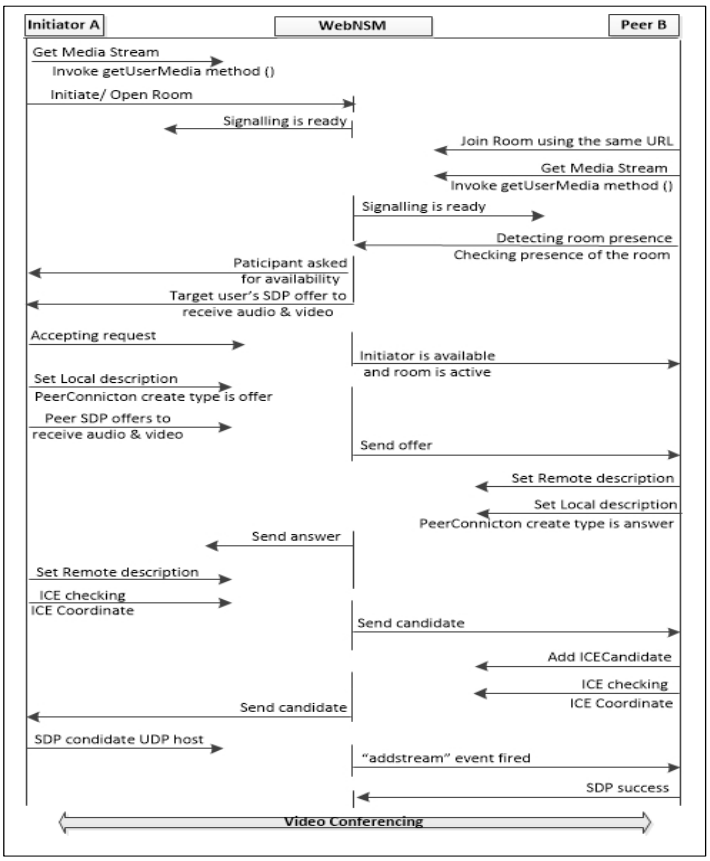

Figure 2. Presents the signalling between broadcasters

3.3. Analysis

This test was achieved among thirty peers during three to four minutes over Local Area Network (LAN) and Wide Area Network (WAN). The Quality of Experience (QoE) was used because it offers significant insight for developers on how the peers experience the quality of their video and audio applications [3]. Also, a measurement of CPU and memory usage using the task manager of Windows 10 within the established connection was obtained, including WebNSM performance via inspect element of Firefox in real-time communication. The analysis can be explained as follows:

3.3.1. WebNSM (A Hybrid Signalling Mechanism)

A performance of WebNSM has been analysed individually among two to thirty users according to two concepts; the first was based on the delay to get ready and the second depends on sending a request and receiving a response. Therefore, WebNSM over LAN network consumes 79 (milliseconds/ms) as minimum consumption and 113 (ms) as maximum consumption to get ready, as well as it consumes 106 (ms) as a minimum use and 120 (ms) as maximum consumption to send a request and receive a response. The mean time was calculated so WebNSM expands 89 (ms) to be ready and expands 111 (ms) to send a request and receive a response. On the other hand, WebNSM over WAN network consumes 78 (ms) as minimum consumption and 89 (ms) as maximum consumption to get ready, as well as it consumes 106 (ms) as minimum consumption and 124 (ms) as maximum consumption to send a request and receive a response. The mean time was calculated so it expands 83 (ms) to be ready and expands 111 (ms) to send a request and receive a response. Based on the consumed time, it has noticed that LAN & WAN networks are exhibited a convergent consumption. WebNSM has an efficient performance while it leads to setup, establish and end a session.

3.3.2. Quality of Video Conferencing

Actual users have participated in this scenario to give their individual opinions on the perceived user experience by the use of questionnaires. The quality of audio and video has been analysed based on

three topologies:

a. Bidirectional (mesh): the quality of audio and video up to ten peers using bi-directional system were excellent. However, due to CPU limitations, the increasing of a number of peers influenced the quality of audio and video. Thus, it would not raise the number of users, while CPU capability was not able to communicate anymore.
b. Unidirectional (simplex & star): this scenario was specified for viewers. All viewers were connecting to all broadcasters from different devices concurrently, but they were not able to connect between themselves. The quality of audio and video up to thirty peers as one broadcaster and 29 viewers using unidirectional system were excellent. Nevertheless, it would not increase the number of viewers, while CPU capability was not able to communicate anymore.
c. Hybrid (Bi-directional & Unidirectional) system: the quality of audio and video using both topologies were excellent. Nevertheless, due to CPU limitations, the number of users was limited especially when the number of broadcasters was raised. Moreover, as much as the number of broadcasters is decreased it would be possible to enhance the number of viewers, while the broadcasters are using mesh topology, which needs a high CPU usage.

3.3.3. Mesh Topology

In a mesh, any conference member can invite another user to join or leave at any time without influencing the remaining participants. In addition, all peers connect among themselves to transmit data from different devices simultaneously. Thus, many links can be created among peers, so there is p*(p-1) number of connections where p is the number of peers. Moreover, each peer needs a minimum of four RTP (Real Time Protocol) to transmit data. Therefore, communication in mesh requests a high CPU and high bandwidth speed, as long as each peer sends and receives different RTPs from the all connected participants at the same time as illustrated: one RTP port for outgoing video, one RTP port for outgoing audio, one RTP port for incoming video and one RTP port for incoming audio.

3.3.4. CPU Performance

It plays a significant role on WebRTC video conferencing, especially using mesh topology. In this experiment, a Xeon CPU was used which is a new generation that has very high performance and bandwidth connectivity to meet the most exacting camera viewing, management needs and processing [23], including CPU core i5 and i7 was used. Mesh handles a high load due to different sources is sending and receiving the videos at the same time, this loading will impact the CPU performance which in turn affects the quality of audio and video. On the other hand, CPU performance in the hybrid unidirectional system was exhibited with rather a low usage than bi-directional. In the meantime, using unidirectional system requires CPU abilities less than the bi-directional system. Each viewer requires a maximum of two RTPs (Real Time Protocol) from each broadcaster to receive data as one RTP port for incoming video and one RTP port for incoming audio. Using simplex will promote resources while it requires less CPU and bandwidth consumptions than mesh topology. Figure 3 displayed the CPU performance on the broadcaster side.

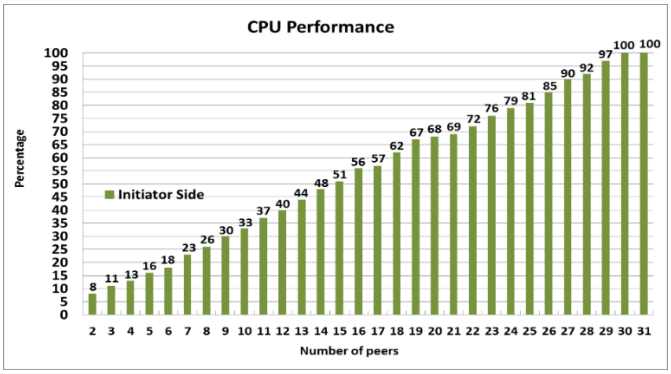

Figure 3. Demonstrated CPU performance based on the initiator end over both LAN and WAN networks

3.3.5. Memory Usage

Practically, memory did not consume much capabilities while peers only need to hold a small amount of session state data, such as when peers are connected. Also, the conferencing was in real time; therefore, there is no need to utilise a high memory as needed for storing or uploading data. Memory usage

did not impact the quality of the audio and video or communication, so all needed over LAN and WAN networks was between 18% to 38%.

3.3.6. Bandwidth Consumption

Different users have different bandwidth speed while each peer might use the various browser, as well as bandwidth requires to handle the overall session grows for every new participant [14]. In this fashion, each browser is built or can be forced based on several video codec and audio codec so that they will consume different bandwidth depends on their codecs. This system used Firefox that relies on Opus audio codec which can change bitrates dynamically from 6 kb/s to 510 kb/s [24]; and VP8 as a video codec. According to this analysis, the following results were found: each peer needs to minimum 1Mb/s bandwidth for each RTP on the video via LAN and WAN networks and needs to 52 - 55 kb/s bandwidth for each RTP on the audio via LAN and WAN networks. As a consequence, bandwidth consumption leads to a bottleneck on the client end, which effects on Quality of Experience (QoE) of video and audio, and the performance may drop significantly [25]. Figure 4, Figure 5, Figure 6 and Figure 7 present the difference of bandwidth consumption via broadcasters and viewers on LAN and WAN networks.

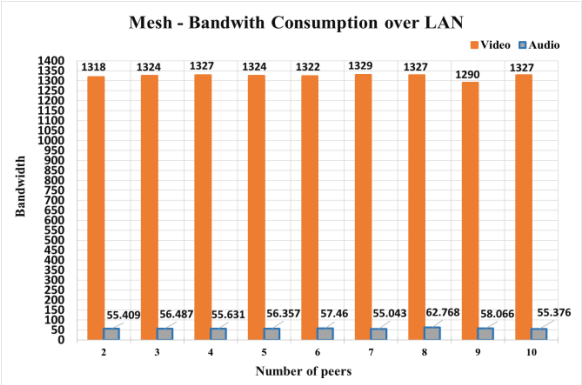

Figure 4. Illustrated the bandwidth consumption of audio and video over LAN network as broadcasters. The unit of bandwidth is kb/s

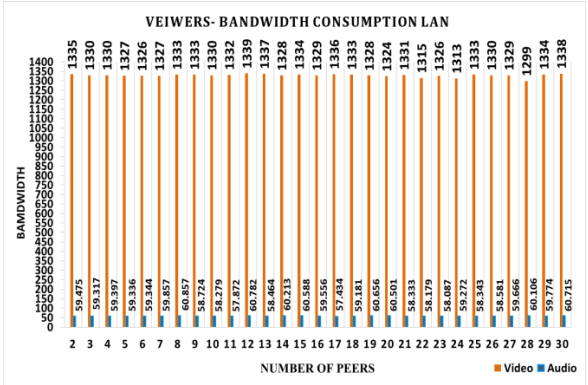

Figure 5. Displayed the bandwidth consumption of audio and video over LAN network as viewers. The unit of bandwidth is kb/s

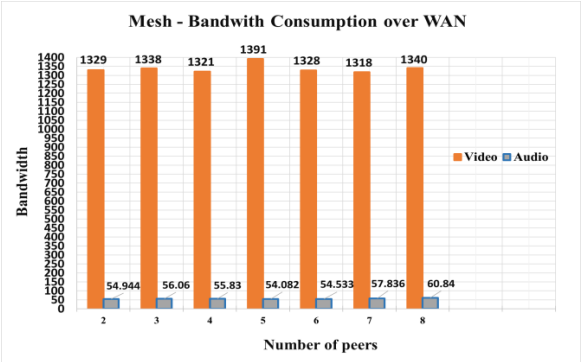

Figure 6. Shows the bandwidth consumption over WAN network among seven peers as broadcasters. The unit of bandwidth is kb/s

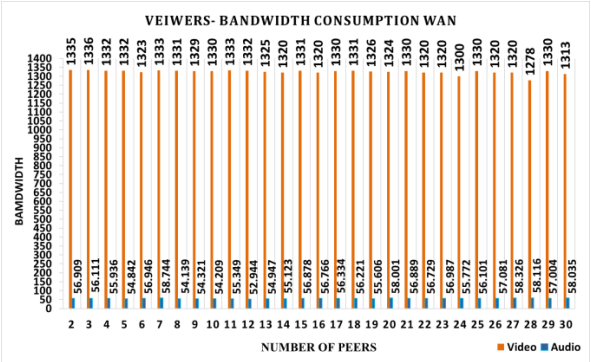

Figure 7. Illustrates the bandwidth consumption of audio and video over WAN network as viewers. The unit of bandwidth is kb/s

3.3.7. Hybrid Topology

A host peer should initiate and start its browser to allow any user to participate in the session at any time without affecting the remaining participants, so using different systems allowing all peers to connect with each other as viewers and broadcasters to transmitted data from different devices simultaneously. A hybrid uses different topologies and gives the users flexibility, reliability and multi-choice of

communications such as initiator, broadcaster or viewer. Moreover, it allows several resources such as devices, networks and users to obtain video conferencing without any registration, downloading or installation and can be used in different applications. Using this scenario shows that it built a strong WebRTC application that works across multiple browsers, networks and topologies. Figure 8, indicates the architecture of the hybrid system.

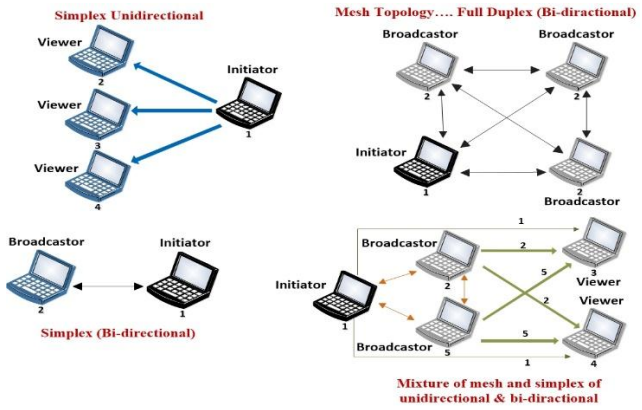

Figure 8. Demonstrates the architecture of hybrid systems

4. EVALUATION

It is proved that WebNSM is able to setup, establish and close a session over LAN or WAN networks. WebNSM is able to offer simplex (unidirectional), star (unidirectional) and mesh topology (bi-directional). On the contrary, it is affected by the CPU, which limits the number of peers. A 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 concurrently. This can be defined as CPU stress and it depends on different elements e.g. the used codec's and the quality of the audio and video. In addition, the variety of bandwidth speeds among the various users can impact the quality of video and audio. Therefore, mesh topology requests a high CPU and high bandwidth speed. For instance, when a user uses CPU core i5, they cannot perform as another user, who uses CPU Xeon, etc. In other words, as high as the CPU core, it will lead to allow more peers to join, better communication and encoding & decoding. Thus, CPU Xeon, which has very high performance and bandwidth connectivity in order to find out the difference among the existing CPUs, was used. According to the indicated limitations, it can be emphasised that CPU plays a significant role in communication and the number of peers, as long a bandwidth does a leading role in the quality of audio and video. The available CPUs at the used computers (e.g. Core i5 & Core i7) are not able to encode, decode, send and receive video conferencing at the same time more than eight peers via mesh topology in real implementation. This is a very productive system that offers two mechanisms for video conferencing. The user is free to choose the appropriate mechanism based on its available bandwidth, and CPU capabilities, as well as this system is changeable as long as the user can change its position from broadcaster to viewer conversely. Additionally, the participant can simply join the session as a broadcaster (using mesh) or as a viewer (using simplex), so using the hybrid system reduces the load on the CPU and bandwidth consumption efficiently and without impacting other participants. 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.

5. CONCLUSION AND FUTURE WORK

In this paper, a hybrid WebRTC signalling mechanism and video conferencing using uni-directional and bi-directional systems were designed and tested in real implementation among thirty PCs. Besides, WebNSM can be considered as a novel signalling mechanism while it presents a flexible communication among users. Moreover, this can be applied in different applications, such as get a group of people together on one call at the same time, conferencing among users, entertainment. e-Learning between teacher and students, m-Health among patients and doctor or specialist and technicians, etc. WebNSM takes an average of 89 (milliseconds) to be ready and 111 (milliseconds) to send a request and receive a response, even when the network is congested. A deep explanation of CPU performance, memory usage, signalling performance,

RTPs calculation, QoE, mesh topology and simplex topology in a physical implementation was done. This scenario is efficient while it provides visually demo over the various devices and networks with a user that requires deep explanation and face-to-face communication. Also, it improves communication & reinforces relationships and increase productivity among users and teams. In the future, there is an intention to expand this work over more scalable video conferencing using MATLAB simulator to discover the effectiveness of resources in WebRTC.

ACKNOWLEDGEMENTS

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. Availablt: https://link.springer.com/article/10.1007/s00607-014-0429-2.

[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. Available: http://ieeexplore.ieee.org/abstract/document/7219558/.

[3] L. N. Eirik Fosser, "Quality of Experience of WebRTC based video communication," Norwegian University of Science and Technology, 2016. Available: https://brage.bibsys.no/xmlui/bitstream/handle/11250/2409900/15147_FULLTEXT.pdf?sequence=1.

[4] W. Elleuch, *"Models for multimedia conference between browsers based on WebRTC,"* in International Conference on Wireless and Mobile Computing, Networking and Communications, pp. 279–284, 2013. Available: http://ieeexplore.ieee.org/document/6673373/.

[5] D. T. Nguyen, K. K. Nguyen, S. Khazri, and M. Cheriet, "Real- Time Optimized NFV Architecture for Internetworking WebRTC and IMS,", pp. 81–88, 2016. Available: http://ieeexplore.ieee.org/document/6673373/.

[6] 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. Available: http://ieeexplore.ieee.org/document/6956711/.

[7] S. Vashishth, Y. Sinha, and K. H. Babu, *"Addressing Challenges in Browser Based P2P Content Sharing Framework Using WebRTC,"* in IEEE 30th International Conference on Advanced Information Networking and Applications (AINA), pp. 850–857, 2016. Available: http://ieeexplore.ieee.org/document/7474178/.

[8] Schahin Rajab, "Comparing different network topologies for WebRTC conferencing," 2015. Available: https://www.kth.se/social/files/56143db5f2765422ae79942c/WebRTC.pdf.

[9] S. a S. T. Miner, "Getting Started with,", pp. 1–41, 2013. Available: http://www.cambridgeinternational.org/teaching-and-learning/getting-started-with/.

[10] R. Rai, *Socket. IO Real-time Web Application Development*. BIRMINGHAM - MUMBAI: PACKT, 2013. Available: https://www.packtpub.com/web-development/socketio-real-time-web-application-development.

[11] 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. Available: http://ieeexplore.ieee.org/abstract/document/7160422/.

[12] C. Notice and A. Notice, "WebRTC to complement IP Communication Services," 2016. Available: https://www.gsma.com/futurenetworks/wp-content/uploads/2016/02/WebRTC_to_complement_IP_Communication_Services_v1.0.pdf.

[13] 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. Available: http://ieeexplore.ieee.org/document/7556048/.

[14] M. Grinberg, "socketio Documentation," 2016. Available: https://media.readthedocs.org/pdf/python-socketio/latest/python-socketio.pdf.

[15] 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]. Available: https://w3c.github.io/webrtc-pc/.

[16] Ana Pol González, "Definition of A Mena Opinion Score for Vp8 Over Real-Time Connections," Universida de Vigo, 2017. Available: http://pequod.det.uvigo.es:8080/xmlui/bitstream/handle/123456789/91/TFM Ana Pol Gonzalez.pdf?sequence=1.

[17] 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. Available: https://pdfs.semanticscholar.org/a9b4/968709b3a6bc5a6be465042da62714e71a13.pdf.

[18] 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. Available:

http://www.ijfcc.org/papers/319-W002.pdf.
[19] 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].
[20] T. Ambra, F. Paganelli, A. Fantechi, D. Giuli, and L. Mazzi, *"Resource-oriented design towards the convergence of Web-centric and Telecom-centric services,"* in Second International Conference on Future Generation Communication Technologies (FGCT), pp. 120–125, 2013. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6767203.
[21] 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, PV, pp. 1–8, 2013. Available: http://ieeexplore.ieee.org/document/6691454/.
[22] A. Amirante, T. Castaldi, L. Miniero, and S. Romano, "On the seamless interaction between webRTC browsers and SIP-based conferencing systems," in *IEEE Communications Magazine*, vol. 51, no. 4, pp. 42–47, 2013. Available: http://ieeexplore.ieee.org/document/6495759/.
[23] V. S. Class, "Complete solution More Features Easy," 2016. Available: http://rasilient.com/wp-content/uploads/2016/02/ApplianceStor-63-TW-1-25-16.pdf.
[24] K. Vos, "RTP Payload Format for the Opus Speech and Audio Codec draft-ietf-payload-rtp-opus-11," 2015. Available: https://tools.ietf.org/pdf/draft-ietf-payload-rtp-opus-11.pdf.
[25] Villanueva, F. Valverde, and O. Pastor, *Information System Development*. 2014. Available: http://link.springer.com/chapter/10.1007/978-3-319-07215-9_8%5Cnhttp://link.springer.com/10.1007/978-3-319-07215-9.