Performance Evaluation of Resources Management in WebRTC for A Scalable Communication

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*— Web Real-Time Communication (WebRTC) offers peer-to-peer communications without any plug-ins. However, WebRTC cannot provide scalability because of its method that depends on a single server or due to the resource limitations and network topology in the architectural of the WebRTC. This paper aims to design a real environment using MATLAB simulation tools to specify the limitations of resources in WebRTC for bi-directional video conferencing, such as CPU performance, bandwidth consumption and Quality of Experience (QoE) using different topologies such as mesh, star and hybrid (a combination of unidirectional/star & bi-directional/mesh). Moreover, several CPU cores like i3, i5, i7, Xeon, i9 and Xeon Phi, as well as bandwidths: 0.5, 1, 5, 10, 30, 50, 100, 500 and 1000 (Mb/s) were considered to achieve and expand the scalability. In this implementation, the factors of real-time implementation were used. Thus, the utilized measurements were already validated while MATLAB presents coefficient with 95% confidence bound. Additionally, this paper highlights the obstructions are preventing scalability in WebRTC using a centralized server. This illustration is beneficial for interested developers who intend to use WebRTC duplex video conferencing among undefined users and different topologies. Furthermore, our simulation-based' performance evaluation shows the efficiency of the hybrid topology in decreasing the bandwidth overhead and CPU load in WebRTC.**

Keywords*— *The Web Real-Time Communication (WebRTC); Quality of Experience (QoE); Mesh topology; Star topology and Hybrid topology.

I. INTRODUCTION

Web Real-Time Communication (WebRTC) was developed by the World Wide Web Consortium (W3C) and Internet Engineering Task Force (IETF) [1]. WebRTC is an open source and a collection of protocols and standards [2]. It allows the transportation of audio, video and data without plugins [3]. Several trials were produced to attain a scalability in WebRTC to reduce the overhead, CPU usage, bandwidth consumption, etc. Therefore, some developers used XMLHttpRequest (XHR/polling) for video conferencing, but XHR leads to waste of bandwidth and delay [4]. Moreover, it is active with communication that does not need the full-duplex approach [5]. In contrast, other developers used SIP (Session Initiation Protocol) with WebRTC to execute video calls. However, SIP has a high bandwidth consumption and delay [6]. Different approaches have worked towards using peer-to-peer (P2P) overlay networks to accomplish a scalability and to allow end users to help the website operator distribute the static objects (e.g., images, videos, etc.) that make up web pages. Nevertheless, these systems are built either using browser plug-ins that the user must install or client-end software that must be downloaded and run [7]. Furthermore, designing of P2P overlay networks faces a lot of challenges. For instance, the scalability and multidimensional data are considered as primary issues that lead to raising the system complexity. In addition, a configuration of P2P overlay networks on actual machines or networks is not practicable. Therefore, they should be implemented before applying and after development [8]. Consequently, most of the P2P methods do not scale well because each query generates a significant amount of traffic [9]. A key difference between traditional P2P applications and a WebRTC web application is the P2P inability to directly establish connections to peers, despite knowing their public IP-address [10].

Network topology in the architecture of WebRTC, CPU performance and bandwidth consumption that are playing the primary role in video conferencing. [11] illustrated that choosing the suitable network topology in the architectural design of the WebRTC application is considered as one of the essential problems. Hence, it should select an appropriate architecture for the application. In addition to [12] who described that video conferencing demands to process power in order to decode, encode and distribute video and audio in real time. This CPU stress relies on different factors such as the codecs used, the quality of both audio and video and video size. Besides, the CPU limitations affect the user with the reduced CPU usage [3]. Additionally, high processing offers higher video resolution on the system, but this request leads to the more significant congestion, resulting in video conferencing services demanding to meet high CPU requirements [13]. CPU has an essential impact on video conferencing, while it handles a high load due to different sources sending and receiving the videos at the same time. The CPU limitations affect only the user with the reduced CPU usage [3]. Any communication between peers needs to have a separated RTP (Real Time Protocol) for the audio and video. Thus, 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, and one RTP port for incoming audio [14][15]. Bandwidth consumption also plays a leading role in communication, while it requires sustaining the overall session grows for each new participant; therefore, different users will have different bandwidths [16]. It can lead to a bottleneck on the client end, and the performance may drop significantly [17]. Besides, [12] demonstrated that bandwidth availability and latency rely on the differences between the variety of devices and networks, so it leads to interrupt the conference while the system is not able to react appropriately.

The primary objectives of this paper are to design a real environment using MATLAB tools to identify the reasons that impede a scalability in WebRTC using a centralized server and to evaluate the impact of resources in WebRTC for duplex video conferencing such as CPU performance, bandwidth consumption, using different topologies and quality of experience (QoE). Moreover, this test is applied using an unlimited number of users in one and four groups to increase and decrease communication links. This paper gives an extensive explanation to support the concerned developers to get the specifications of WebRTC resources on the scalability and to determine the best topology that can be used in WebRTC bi-directional video conferencing. This implementation used the factors and measurements of real-time implementation as described in [14][15][18].

This paper is organised and outlined as follows: Section II some WebRTC related work. Section III describes the strategy along with the architecture. Section IV, illustrates the implementation. Section V, discusses the implementation results. Section VI, presents the analysis and discussion. Finally, Section VII has the conclusion with the future work.

II. RELATED WORK

Different developers attempted to manage scalability in WebRTC for video broadcasting, video conferencing and audio calls. However, most of them faced several issues over their work. For instance, applying it to limited users utilises plug-ins, they have not implemented their proposed mechanisms. They used P2P overlay network that has some candidate problems: (a) indirection/directional communication, (b) bandwidth complicity, (c) a large number of peers and (d) lack of usage for video conferencing [19]; and also used MCU (Multipoint Control Unit), etc. The following describes and discusses some of their mechanisms:

Designed a scalable communication in WebRTC based on Chord algorithm using PeerConnecion over DataChannel (POD). But, Chord was designed for a unidirectional network and a WebRTC peer itself cannot immediately connect the other peers, as well as this application used for chat communication as messages while POD was used for back-channel content such as images, text chat and game update packets [20]. Furthermore, using CANs (Content Delivery Networks) proposed a hybrid peer-to-peer network to aid video chunks delivery. Nevertheless, this approach is useful for a small number of peers and it does not guarantee the content exchanged between peers [21]. Additionally, [22] implemented WebRTC video streaming using XmlHttpRequests, WebSockets and PeerJS server. This implementation used 32 peers as a maximum number, including using a pull-based protocol over large-scale systems. However, the node requests the package twice and using a push-based protocol is not sufficient for streaming systems and due to the periodical exchange of buffer-maps among the peers it has challenging overhead [23]. In addition, proposed Peer-to-Peer audio and video application using WebSocket and Node.js as a signalling server. However, the system is assumed for two students at one session and it offers unclear audio and echo [24].

Developed a web-based multimedia application, which has a scalability to provide a low latency and high throughput. But, this implementation was done between only two clients [25]. Furthermore, a developed protocol has been proposed for decentralized conferencing with WebRTC using two techniques like voice-activated switching and Load-balancing. These techniques were used to solve the scalability problems and video conferencing service. Nevertheless, the features in a real-world application are entirely theoretical, so it needs to be implemented [26].

Based on the current works as many P2P systems are not accurate and a key reason for the lack of fully-distributed P2P systems is the difficulty in designing highly robust algorithms for large-scale dynamic P2P networks [27], and that includes the various articles of the related work as shown above. The implementation of this paper demonstrates an explanation that helps to recognise most of the resources limitations for scalability in WebRTC for video conferencing. In addition, it extends the number of participants in several topologies to be more than 140 peers in bi-directional video conferencing using the same server, analyses CPU performance, bandwidth consumption, QoE, etc.

III. STRATEGY AND ARCHITECTURE

a) Strategy

It is divided into five parts as follows:

1) Using various topologies, such as the star (one-to-many) bi-directional communication, mesh (many-to-many) bi-directional communication and hybrid based on simple system (one-to-one unidirectional/bi-directional), star topology (one-to-many unidirectional) and mesh topology (many-to-many bi-directional) communication. The hybrid was achieved between broadcaster-to-broadcaster (bi-directional) and broadcaster-to-viewer unidirectional).
2) Using undefined number of users divided into one and four groups
3) Using several CPU cores: i3, i5, i7, Xeon, i9 (18 core)[28], and Xeon Phi (72 cores)[29]
4) Using different bandwidths (Mbps): 0.5, 1, 5, 10, 30, 50, 100, 500 and 1000 (1G)
5) Using Quality of Experience (QoE) based on the interconnection between CPU loads and bandwidth conditions. In addition, using Mean Opinion Score (MOS) as a measure in the domain of QoE resulting in a score

between 1-5 with 1 being very bad, 2 being bad, 3 being fair, 4 being good and 5 being excellent.

b) Architecture

Core i7 & RAM 8 GB computer; and MATLAB software were used to design the GUI (Graphical User interface) of this environment. The mentioned topologies can be applied individually so each topology can be run based on its nature as clarifying below, except for the hybrid, which has a particular mechanism. Figure (1) shows the architecture of the actual model.

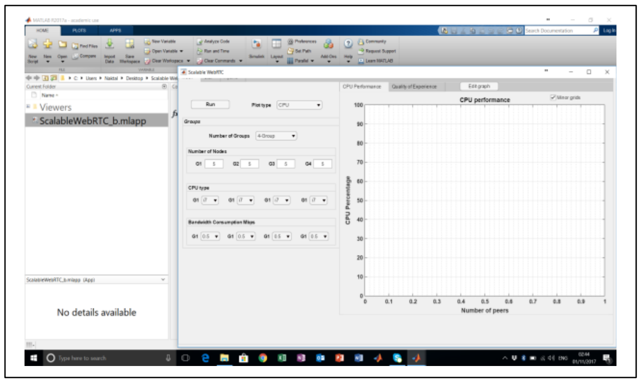

Fig 1, Presents an actual model using different topologies

- Star topology

In this topology, the network relies on an initiator so if the initiator goes down, the whole communications will go down. On the other hand, participants do not need a high capacity of CPU or high bandwidth [11], as long as they cannot communicate among themselves [15]. In contrast to what has been said, this implementation has applied the number of users in one group, and the evaluation of CPU loads and QoE has only focused on the initiator side. Therefore, it should specify the initiator's CPU, participants CPU and the number of nodes as shown in figure (2), but to gain QoE, we should limit the bandwidth as well.

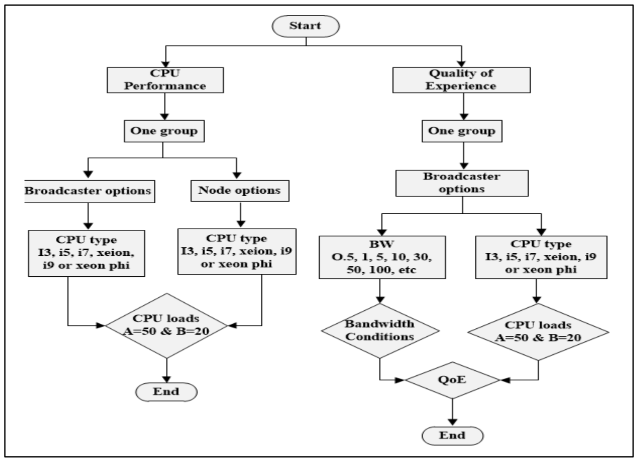

Fig 2, illustrates the flowchart of star in this implementation

- Mesh topology

In mesh topologies, any conference member can invite another user to participate or leave at any time without affecting the remaining participants. It uses many links among users to transfer data and all peers connect among themselves simultaneously [30]. In this implementation, grouping the number of users into one and four groups has been applied, and each group has undefined users, different CPUs and several bandwidths. Figure (3) shows that an initiator should select the kind of group and specify CPU with bandwidth, as well as finding out CPU loads or QoE for each group individually.

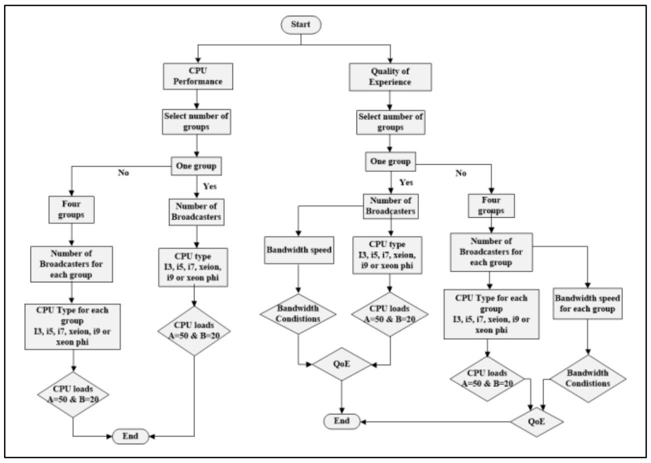

Fig 3, demonstrates the flow chart of mesh in this implementation

- Hybrid topology

According to real-time implementation, a host peer should initiate and starts its browser to allow any user to participate in the session so using different systems allowing all peers to connect with each other as viewers and/or broadcasters. Figure (4) indicates that when attempting to discover CPU loads, there are different equations that should be used as one defined for broadcasting and another one specified for a viewer. Moreover, the same strategy should be used when finding QoE after selecting the bandwidth.

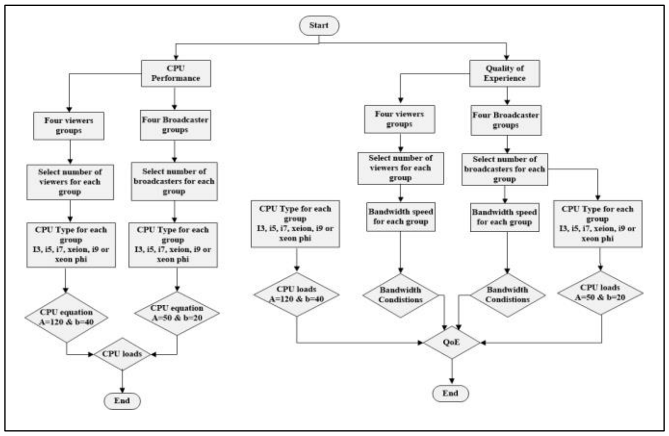
Start
CPU Performance
Quality of Experience
Four viewers groups
Four Broadcaster groups
Four viewers groups
Four Broadcaster groups
Select number of viewers for each group
Select number of broadcasters for each group
Select number of viewers for each group
Select number of broadcasters for each group
CPU Type for each group I3, i5, i7, xeion, i9 or xeon phi
CPU Type for each group I3, i5, i7, xeion, i9 or xeon phi
CPU Type for each group I3, i5, i7, xeion, i9 or xeon phi
Bandwidth speed for each group
Bandwidth speed for each group
CPU Type for each group I3, i5, i7, xeion, i9 or xeon phi
CPU equation A=120 & b=40
CPU equation A=50 & b=20
CPU loads A=120 & b=40
Bandwidth Condistions
Bandwidth Condistions
CPU loads A=50 & b=20
CPU loads
QoE
End
End

Fig 4, presents the flowchart of mesh in this implementation

IV. Implementation

Using Fermi-Dirac distribution which is a special case of *Boltzmann's* equation that gives the best curve fitting and MATLAB tools based on the factors of real-time execution, which achieved using CPU core i5, i7 and Xeon via Wired of LAN (Local Area Network) and WAN (Wide Area Network) as detailed in [14][15][18], various equations to calculate CPU loads for broadcasters and/or viewers in all mentioned topologies have been created and also finding out Quality of Experience (QoE) by using CPU loads and bandwidth conditions as presented:

A. An equation of CPU loads

Different variables were used where x = number of nodes, a & b = curve fitting for measuring and factor (f) = a specific value of each CPU that is found by using MATLAB tools and according to the real-time implementation as shown in equation (1).

$$\boldsymbol{CPU_{Load} = \frac{100}{1+e^{\left(\frac{a*f-x}{b*f}\right)}}} \quad \ldots\ldots\ldots \ (1)$$

Equation (1), displays CPU loads-based number of nodes

The equation (1) was applied to core i5, i7 and Xeon individually and the outcomes were coefficient with 95% confidence bounds compared with their results in real-time implementation. Therefore, this equation has also been applied to obtain CPU loads for the others such as Core i3, i9 and Xeon Phi. Furthermore, the factor of each CPU was found using equation (1) and based on peer's number. Figure (5), describes the pseudocode of each CPU factor. Moreover, figure (6), presents an example of the CPU difference between real-time implementation and this implementation.

```
CPU load

SET a = 50, b = 20, x = number of nodes
SET f according to the CPU type as
SWITCH CPU Type:
    case 'i3'
            f = 0.03;
    case 'i5'
            f = 0.04;
    case 'i7'
            f = 0.06;
    case 'Xeon'
            f = 0.09;
    case 'i9'
            f = 0.25;
    case 'Xeon Phi'
            f = 1;
END

Calculate the CPU load
CPUload% = 100/(1 + (exp((f*a-x)/f*b)))
End
```

Fig 5, shows the pseudocode for the CPU load for broadcasting in all topologies

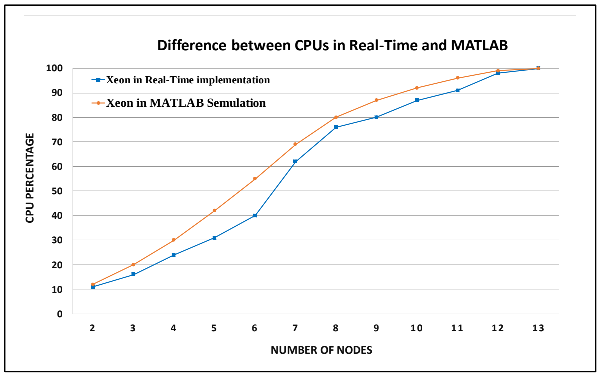

Fig 6, shows the similarities between CPUs performance

More importantly, when communicating only between broadcasters (bi-directional), it has considered that a=50 and b=20 to obtain an efficient outcome comparing with the real-time implementation. On the other hand, if communication between one broadcaster and one/more viewers (unidirectional), the streaming of the audio and video will only be sent from the broadcaster to the viewer. In this case, a=120 and b=40 to get a productive result that is similar to the real-time implementation result should be used. Thus, in this status has combined the equation of CPU loads for broadcasters with the equation of CPU loads for viewers to get an excellent result. Figure (7), illustrates the pseudo code of combining the broadcaster and viewer equations in hybrid topology.

```
CPU load for broadcasters & Viewers in hybrid topology

SET f according to the CPU type as
SWITCH CPU Type:
    case 'i3'
            f = 0.03;
    case 'i5'
            f = 0.04;
    case 'i7'
            f = 0.06;
    case 'Xeon
            f = 0.09;
    case 'i9'   f = 0.25;
    case 'Xeon Phi'   f = 1;
END

SET a = 50, b = 20, x = number of broadcasters
Calculate the CPU load for broadcasters
Cpuload1% = 100/(1 + (exp((f*a-x)/f*b)

SET a = 120, b = 40, x = number of viewers
Calculate the CPU load for viewers
Cpuload2% = 100/(1 + (exp((f*a-x)/f*b)

CPUoad% = Cpuload1 + Cpuload2

IF CPUload% > 100
CPUload% = 100%

End
```

Fig 7, shows the pseudocode of CPU load in hybrid system

V. Implementation Results

Using CPU loads equations for broadcasters in star and mesh and using broadcasters and viewers in hybrid has given a productive result as in real-time implementation. The mentioned CPUs were implemented in all topologies and that is why the CPU core i3 was able to provide an excellent communication between three to four peers, core i5 was able to give an excellent communication between five to six peers, CPU core i7 was able to give an excellent communication between seven to eight peers, CPU core Xeon (called ix in this implementation) was able to give an excellent communication between eleven to twelve peers, core i9 (18 core) was able to give an excellent communication up to thirty-five peers, and CPU core Xeon Phi (named iz in this implementation) was able to give an excellent communication up to 315 peers. On the other hand, these CPUs cannot load more than the indicated numbers while the increase will influence the QoE as shown in tables (1, 2 & 3). In the hybrid, each viewer will communicate with all broadcasters, despite the fact that each broadcaster will communicate with the other broadcasters and viewers as well. This means, the more decreased the broadcasters number is, the more increased the viewer's number can be. Clearly, CPU capability for each broadcaster is equal to two viewers. Thus, communication in hybrid accomplished over a duplicated number of viewers is compared with the number of broadcasters as shown in the table (3). Overall, it has been proved that increasing the CPU core can raise the number of users. Whereas, CPUs extension confirmed its ability to achieve scalability in WebRTC video conferencing as presented in figures (8, 9, 10, 11 & 12).

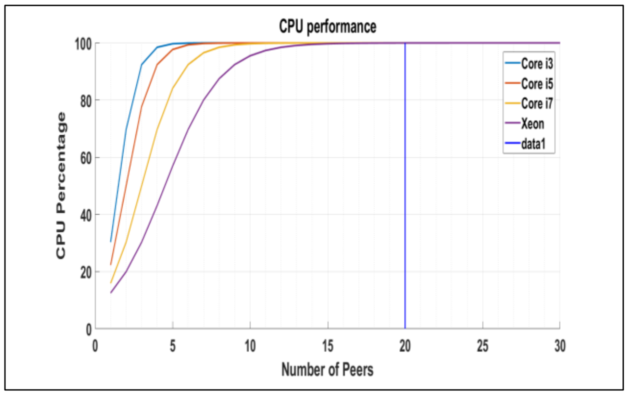

Fig 8, demonstrates CPU loads among twenty peers in mesh topology

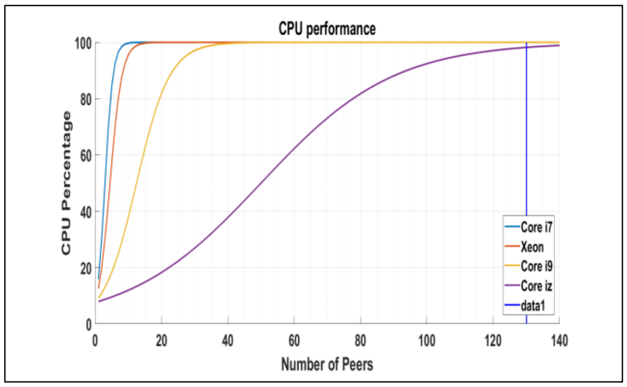

Fig 9, shows CPU loads among hundred and thirty peers in mesh topology

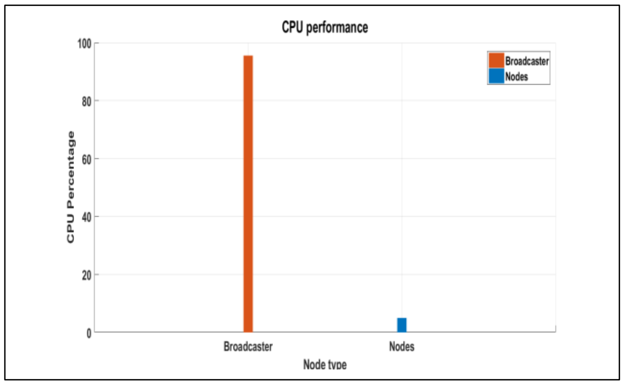

Fig 10, presents CPU loads among eight peers in star topology when the broadcaster uses CPU core i7 and the participant use core i7

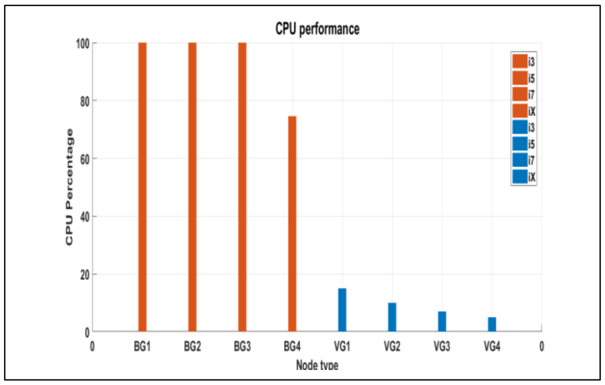

Fig 11, displays different CPU loads among (i3, i5, i7 & ix) for four broadcasters and eight viewers (i3, i5, i7 & ix) in hybrid topology

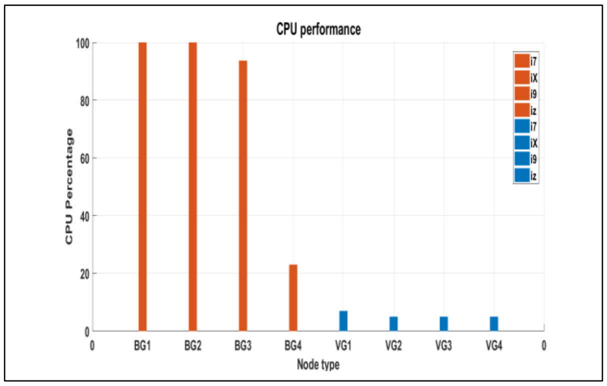

Fig 12, illustrates different CPU loads among (i7, ix, i9 & iz) for eight broadcasters and twenty viewers (i7, ix, i9 & iz) in hybrid topology

TABLE (1), shows the CPU performance in mesh topology

Seq	**CPU Core**	**Mesh Topology Bi-directional**	
		Excellent	**Unacceptable**
1.	**i3**	**4**	**5**
2.	**i5**	**6**	**7**
3.	**i7**	**8**	**9**
4.	**Xeon**	**12**	**13**
5.	**i9**	**33**	**34**
6.	**Xeon Phi**	**136**	**137**

TABLE (2), presents the CPU performance in star topology

Seq	CPU Core	Star Topology Bi-directional	
		Excellent	Unacceptable
7.	i3	4	5
8.	i5	6	7
9.	i7	8	9
10.	Xeon	12	13
11.	i9	33	34
12.	Xeon Phi	136	137

TABLE (3), displays the CPU performance in hybrid topology

Seq	CPU Core	Hybrid System Bi-directional		
		Broadcaster		Viewer
		Excellent	Unacceptable	Broadcaster-to-viewer
	i3	4	5	9
2.	i5	6	7	12
3.	i7	8	9	17
4.	Xeon	12	13	29
5.	i9	33	34	78
6.	Xeon Phi	136	137	315

B. An equation of Quality of Experience

According to the real-time implementation as mentioned in [14][15][18], that each peer consumes minimum 950 kb/s and maximum 1350 kb/s for video and between 58-63 kb/s for audio on each RTP (Real Time Protocol) via LAN and WAN networks. Moreover, in the real-time implementation, actual users have participated to give their individual opinions; therefore, the quality of audio and video has been assessed on different CPUs, such as i5, i7 and Xeon. Therefore, users revealed that the QoE of the audio and video could be excellent as long as the CPU usage is less than 89% that is score 5 in MOS (Mean Opinion Score). Nevertheless, as much as CPU load exhibits more than 88% that lead to decrease the QoE. For example, if the QoE reaches 91% then its score is 4, so it is good. If it reaches 94% then its score 3 so it is as fair, etc. In this implementation, the mean consumption of bandwidth as each peer consumes 1 Mb/s for video and about 60 kb/s for audio was considered. To this end, QoE equation has been built based on CPU loads and QoE in real time implementation. Thus, the results showed coefficient with more than 95% confidence bounds in MATLAB tools as demonstrated in figure (8). The result of QoE is presented in two bars separately, as one for CPU load and another one for QoE. In fact, it was possible to use one equation to display the overall result, but they were separated to give a clear view for identifying the exact resource effects on the quality of audio and video, including the specification of users. Figure (9), represents the curve fitting of QoE based on CPU usage and QoE in the real-time implementation. The equation of QoE was applied on all topologies using several CPUs and bandwidths. As shown in figures (13, 14, 15, 16, 17, 18 & 19) as examples, diverse groups are used and each group (G) can sustain unlimited users, several CPUs and various bandwidths. For this reason, it is easy to recognise the disparity in groups according to their CPU and bandwidth.

```
QoE based on CPU loads

SET QoE = Quality of experience
        bw = Used bandwidth
        NoN = number of nodes

If (cpuload > 88)
QoE= round (34 - 0.33 * cpuload%);

else
    QoE(i,1) = 5;

END
```

Fig 13, illustrates the equation of QoE

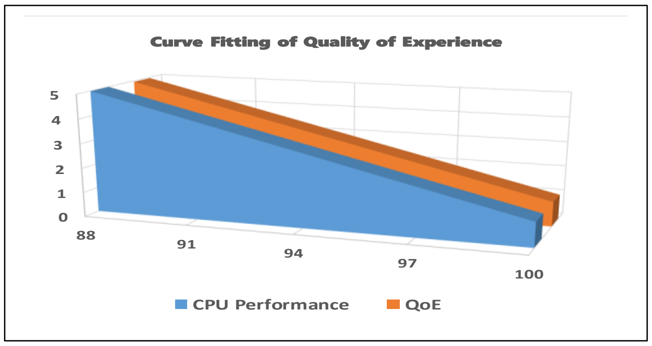

Fig 14, shows QoE based on CPU load in real-time implementation

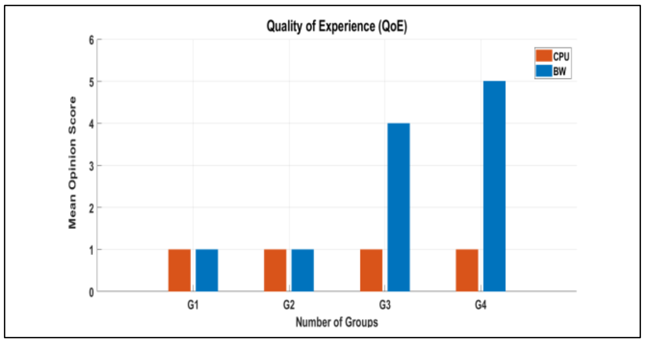

Fig 15, displays QoE in mesh among 20 peers while G1= CPU i3, BW(10), G2= CPU i5, BW (30), G3= CPU i7, BW (50) and G4= CPU ix, BW (100). BW by Mb/s

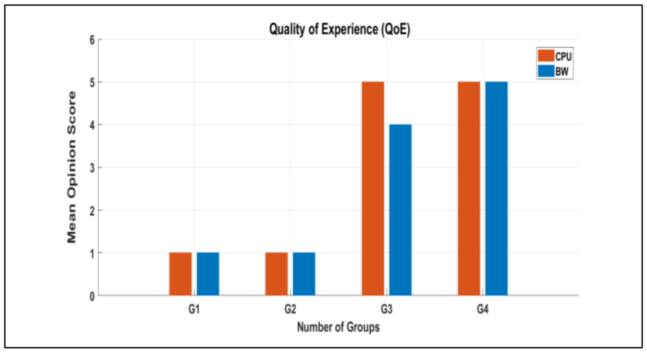

Fig 16, presents QoE in mesh among 20 peers while G1= CPU i7, BW (10), G2= CPU ix, BW (30), G3= CPU i9, BW (50) and G4= CPU iz, BW (100). BW by Mb/s

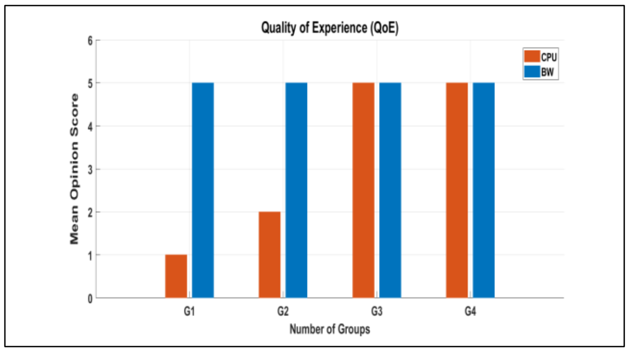

Fig 17, illustrates QoE in mesh among 10 peers while G1= CPU i7, BW (30), G2= CPU ix, BW (50), G3= CPU i9, BW (100) and G4= CPU is, BW (100). BW by Mb/s

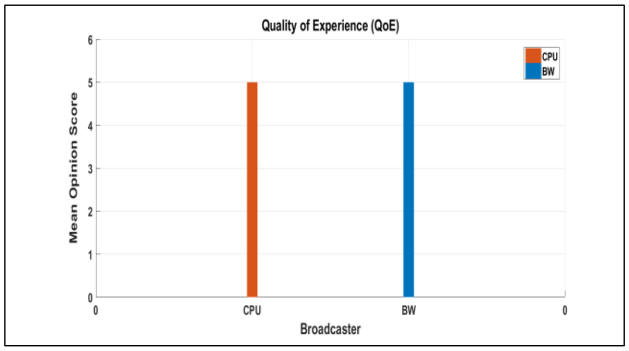

Fig 18, illustrates QoE in star among 5 peers while G= CPU i7 and BW (30) for broadcaster and CPU i7 for participants. BW by Mb/s

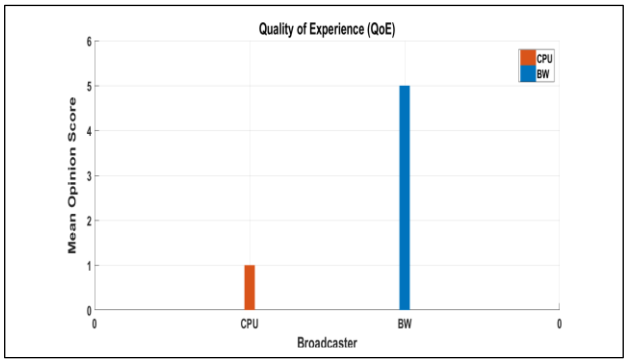

Fig 19, illustrates QoE in star among 5 peers while G= CPU i5 & BW (30) for broadcaster and CPU i7 for participants. BW by Mb/s

VI. Analysis and Discussion

It was proved that resources management is necessary to extend the number of users for scalability in WebRTC. Resources, such as CPU, bandwidth and topology are dovetailed and able to impact on a number of participants, affecting the quality of audio and video, etc. The performance of CPU and bandwidth consumption has significant issues in audio and video conferencing. 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 i3, etc. On the other hand, in the star, a broadcaster exhibits high CPU processing, while it is sending and receiving the audio and video with many users simultaneously. However, a participant does not exhibit a high CPU or bandwidth, as long as he/she is communicating only with the broadcaster. Moving to hybrid, it is the best topology while since offers several options and the user is free to select any of them. As an example, if the user realises that the bandwidth is limited or CPU is not high enough, he/she can simply change from broadcaster to viewer and keep joining the session. This change will maintain the user connected and also consumes less bandwidth. Additionally, the hybrid is useful to use WebRTC video conferencing among various communications. For instance, m-Health (many doctors can communicate with many technicians and patients), e-learning (many teachers can communicate with many students and many students can communicate with each other), communication applications, etc. The quality of experience (QoE) verifies that this testbed environment works correctly as it is in the real-time implementation.

VII. Conclusion

In this implementation, a massive simulation that coefficient with 95% confidence bound as demonstrated in MATLAB tools was obtained. Although, a massive effort has been made in order to evaluate and specify the limitations of resources in WebRTC, especially for bi-directional video conferencing. Therefore, real-time implementation was achieved to exploit its results in this implementation to extend the resources and find out some suggestions and solutions for scalability in WebRTC. In this paper, a novel MATLAB simulation was created and tested, which describes the advantages and disadvantages of using different topologies, CPUs, bandwidths, the number of users, etc. Moreover, it suggests a hybrid system that can reduce the resources consumption and specifications in several applications. This implementation guarantees a different equation for providing a method to manage the resourcing in WebRTC. In addition, a deep evaluation based on the physical implementation was done over CPU performance, QoE, mesh topology, star topology and hybrid topology. Additionally, by managing the resources in WebRTC, can achieve a scalability, therefore in this implementation can support up to 140 broadcasters and 315 viewers while having high core CPUs. In future, there is an intention to consider different CPU cores and develop the hybrid system to be applied based on overlay network techniques.

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.
I want to acknowledge the effort of Mr. Riyadh Abedraba Abbas for supporting by different ideas and advices.

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] L. N. Eirik Fosser, “Quality of Experience of WebRTC based video communication,” Norwegian University of Science and Technology, 2016.

[4] R. Manson, *Getting Started with WebRTC*. Birmingham: Packt Publishing Ltd, 2013.

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

[6] 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.

[7] L. Zhang, F. Zhou, A. Mislove, and R. Sundaram, “Maygh: Building a CDN from client web browsers,” in Proceedings of the 8th ACM European Conference on Computer Systems, EuroSys, pp. 281–294, 2013.

[8] S. Surati, D. C. Jinwala, and S. Garg, “A survey of simulators for P2P overlay networks with a case study of the P2P tree overlay using an event-driven simulator,” *Eng. Sci. Technol. an Int. J.*, vol. 20, no. 2, pp. 705–720, 2017.

[9] Y. Deng, F. Wang, and A. Ciura, “Ant colony optimization inspired resource discovery in P2P Grid systems,” *J. Supercomput.*, vol. 49, no. 1, pp. 4–21, 2009.

[10] M. K. Melhus and P. E. Heegaard, “P2P Video Streaming with HTML5 and WebRTC,” Norwegian University of Science and Technology, 2015.

[11] Schahin Rajab, “Comparing different network topologies for WebRTC conferencing,” 2015.

[12] P. Rodríguez, J. Cerviño, I. Trajkovska, and J. Salvachúa, “Advanced Videoconferencing Services Based on WebRTC,” in IADIS International Conferences Web Based Communities and Social Media and Collaborative Technologies, pp. 180–184, 2012.

[13] J. B. Husić, S. Baraković, and A. Veispahić, “What Factors Influence the Quality of Experience for WebRTC Video Calls,” in 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 428–433, 2017.

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

[15] N. M. Edan, A. Al-sherbaz, and S. Turner, “WebNSM: A Novel WebRTC Signalling Mechanism for One-to-Many Bi-directional Video Conferencing,” in Proceedings of SAI Computing Conference, pp. 1–6, 2018.

[16] M. Grinberg, “socketio Documentation,” 2017.

[17] M. Villanueva, F. Valverde, and O. Pastor, *Information System Development*. 2014.

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

[19] M. Roussopoulos, M. Baker, D. S. H. Rosenthal, T. J. Giuli, P. Maniatis, and J. Mogul, “2 P2P or Not 2 P2P?,” 2004.

[20] J. H. Paik and D. H. Lee, “Scalable signaling protocol for Web real-time communication based on a distributed hash table,” *Comput. Commun.*, vol. 70, pp. 28–39, 2015.

[21] F. R. N. Barbosa and L. F. G. Soares, “Towards the Application of WebRTC Peer-to-Peer to Scale Live Video Streaming over the Internet,” in Simpósio Brasileiro de Redes de Computadores (SBRC), no. Figure 1, pp. 119–124, 2014.

[22] F. Rhinow, P. P. Veloso, C. Puyelo, S. Barrett, and E. O. Nuallain, “P2P live video streaming in WebRTC,” in World Congress on Computer Applications and Information Systems (WCCAIS), pp. 1–6, 2014.

[23] A. Ghaffari Sheshjavani and B. Akbari, “An adaptive buffer-map exchange mechanism for pull-based peer-to-peer video-on-demand streaming systems,” in Multimedia Tools and Applications, vol. 76, no. 5, pp. 7535–7561, 2017.

[24] P. M. Jacob and M. Prasanna, “A Comparative analysis on Black box testing strategies,” in International Conference on Information Science (ICIS), pp. 1–6, 2016.

[25] V. Kandari, “Implementation and Performance Optimization of WebRTC Based Remote Collaboration System,” Blekinge Institute of Technology, 2016.

[26] A. Hallberg, “A protocol for decentralized video conferencing with WebRTC Solving the scalability problems of conferencing,” KTH ROYAL INSTITUTE OF TECHNOLOGY, 2016.

[27] A. Disterhöft and K. Graffi, “Protected chords in the web: Secure P2P framework for decentralized online social networks,” in IEEE International Conference on Peer-to-Peer Computing, 2015.

[28] M. Passingham, “Intel Core i9,” *trustedreviews*. [Online]. Available: http://www.trustedreviews.com/news/intel-core-i9-specs-release-date-skylake-x-2948774. [Accessed: 09-Sep-2017].

[29] “Xeon Phi Processor 7290,” *Intel®*. [Online]. Available: https://www.intel.com/content/www/us/en/products/processors/xeon-phi/xeon-phi-processors/7290.html. [Accessed: 16-Sep-2017].

[30] 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.