T.C.

ALTINBAS UNIVERSITY

Graduate School of Science and Engineering

Electrical and Computer Engineering

DESIGN AND IMPLEMENTATION OF BI-DIRECTIONAL AUDIO AND VIDEO CONFERENCING BASED ON WEB REAL-TIME COMMUNICATION

Abdullah Ali Tahseen Al-Najjar

M.Sc. Thesis

Prof. Dr. Osman Nuri Uçan

Istanbul, 2019

DESIGN AND IMPLEMENTATION OF BI-DIRECTIONAL AUDIO AND VIDEO CONFERENCING BASED ON WEB REAL-TIME COMMUNICATION

by

Abdullah Ali Tahseen Al-Najjar

Electrical and Computer Engineering

Submitted to the Graduate School of Science and Engineering

in partial fulfillment of the requirements for the degree of

Master of Science

ALTINBAŞ UNIVERSITY

2019

This is to certify that we have read this thesis and that in our opinion it is entirely adequate, in scope and quality, as a thesis for the degree of .....

Academic Title Name SURNAME	Academic Title Name SURNAME
Co-Supervisor	Supervisor

Examining Committee Members (the first name belongs to the chairperson of the jury and the second name belongs to supervisor)

Academic Title Name SURNAME	Faculty, University	_____
Academic Title Name SURNAME	Faculty, University	_____
Academic Title Name SURNAME	Faculty, University	_____
Academic Title Name SURNAME	Faculty, University	_____
Academic Title Name SURNAME	Faculty, University	_____

I certify that this thesis satisfies all the requirements as a thesis for the degree of .....

Academic Title Name SURNAME

Head of Department

Academic Title Name SURNAME

Director

Approval Date of Graduate School of Science and Engineering: ____/____/____

I hereby declare that all information in this document has been obtained and presented per academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Abdullah Ali Tahseen Al-Najjar

DEDICATION

My Father and Mother without his incredible support, I could not have completed my studies.

For her ceaseless prayers, encouragement and endless love.

My Brothers and Sisters the hidden strength behind my every success.

ACKNOWLEDGMENTS

First and foremost, I am grateful to Almighty God, for giving me the ability to complete this research, without whom nothing is possible.

My sincerest gratitude and most profound appreciation go to my supervisor Prof.Dr.Osman Nuri UCAN for his endless support, and for providing me with the freedom and independence that have been essential to the making of this thesis. I also much appreciate his care and concern throughout my Master study.

My special gratitude is due to my parents, brothers, and sisters for their endless support. Their prayers and blessings were no doubt the real reason behind any success I have realized in my life.

ABSTRACT

DESIGN AND IMPLEMENTATION OF BI-DIRECTIONAL AUDIO AND VIDEO CONFERENCING BASED ON WEB REAL-TIME COMMUNICATION

Al-Najjar, Abdullah Ali Tahseen

M.S, Electrical and Computer Engineering, Altınbaş University,

Supervisor: Prof.Dr.Osman Nuri UCAN

Date: September 2019

Pages: 52

During the past few years, the Internet Engineers Task Force (IETF) and the World Wide Web Consortium (W3C) established a new standard and called it Web Real-Time Communication (WebRTC). This standard is a standardized mechanism for the establishment of direct audio, video, and data connections among browsers and web users. However, it does not consider signalling mechanisms and protocols as a part of it, which makes signalling the most implementation problem in WebRTC since signalling is standard to the process of peer detection. The peer detection process determines peers and coordinates communications among them to exchange data over the local or the wide area.

This thesis concentrates on establishing and developing WebRTC audio and video conferencing to concurrently provide bi-directional and/or unidirectional video conferencing at the same time and with the same server. Moreover, we can apply the (WebRTC) over Local Area Network (LAN), and Wide Area Network (WAN) using many techniques and methods. Thus, we used to

adjust control, connect, open, and open default socket. The steps were physically implemented to increase the flexibility and adaptability of the application to establish better audio and video conferencing. Furthermore, this procedure supports video conferencing without the need to use any external signalling servers or equipment like Multipoint Control Unit (MCU), Flash or Traversal Using Relays around NAT (TURN) servers. The methodology of this research is as follows:

1. A signalling channel based on Node.JS server for bi-directional video conferencing between two peers and based on simplex topology is to be implemented and established.
2. It is clear that the CPUs and bandwidth consumption do not affect the performance of the application created in this research.
3. To evaluate the Quality of Experience (QoE), bandwidth consumption, Real-Time Protocol (RTP), and CPUs performance.

Thus, it has been created and examined a particular environment using JavaScript programming to check whether there are any resources limitations in WebRTC video conferencing. In the same way, a comparison between WebRTC and Voice over Internet Protocol (VoIP), and the universal WebRTC application (AppRTC) were achieved. This thesis would contribute to the support of researchers interested in understanding and learning more on how to establish a WebRTC bi-directional audio and video conferencing with simplex topology and the limitations of using one server and resources.

Keywords: Node.JS, Web Real-Time Communication (WebRTC), Simplex topology, Networks, and Video conferencing.

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

LIST OF ABBREVIATIONS

API	:	Application Programming Interface
BW	:	Bandwidth
CPU	:	Central Processing Unit
DTLS	:	Datagram Transport Layer Security
GUI	:	Graphical User Interface
HTML5	:	Hypertext Markup Language 5
ICE	:	Interactive Connectivity Establishment
IETF	:	Internet Engineering Task Force
IP	:	Internet Protocol
JSEP	:	JavaScript Session Establishment Protocol
LAN	:	Local Area Network
MCU	:	Multipoint Control Unit
MGCP	:	Media Gateway Control Protocol
NAT	:	Network Address Translation
P2P	:	Peer-to-Peer
PC	:	Private Computer
QoE	:	Quality of Experience
RTCP	:	Real-Time Control Protocol
RTP	:	Real Time Protocol

SDP	:	Session Description Protocol
SRTP	:	Secure Real-time Transport Protocol
STUN	:	Session Traversal Utilities for NAT
TCP	:	Transmission Control Protocol
TURN	:	Traversal Using Relays around NAT
UDP	:	User Datagram Protocol
VoIP	:	Voice over Internet Protocol
WAN	:	Wide Area Network
WebRTC	:	Web Real-Time Communication
W3C	:	World Wide Web Consortium
WS	:	WebSocket
WWW	:	World Wide Web

1. INTRODUCTION

1.1 OVERVIEW

According to [1], real-time multimedia experiences in different directions are supported by communication technology through the Internet . Sending audio, video, and data became a better option because of the low cost and the better quality services, this occurs using the Internet rather than the Public Switched Telephone Network (PSTN) [2]–[4]. Due to the evolution of web technologies, it allows users to request different types of services at anytime, anywhere [5]. Voice over Internet Protocol (VoIP) means transferring audio and video information through the Internet [6], [7], and it has become a standard for the ideal voice and video applications through various devices [4], [8]. To communicate with users and servers, VoIP involves the use of different signalling protocols and Media Gateway Control Protocol (MGCP) [9], [10], but its applications need registration, installations, and other applications that require licenses such as RingCentral, VOIP\STUDIO and sipgate. According to [11], Web Real-Time Communication (WebRTC), which is an open source and a group of libraries, JavaScript Application Programming Interfaces (APIs) and standards [12] that provides interactive communications of audio, video, and data [13], is a new standard that the Internet Engineering Task Force (IETF) and World Wide Web Consortium (W3C) has developed [11]. Moreover, it has several benefits in that it does not require plug-ins or installation, and it is easy to use, as well as it is free, and does not require licensing and high-quality Real Time-Communication (RTC) application [12], [14]. However, W3C and IETF have not yet reached an agreement concerning a final signalling mechanism or a protocol to test WebRTC [15] so the standard of the signalling channel in WebRTC is not yet specified [16], [17]. In another way, it is not clear how WebRTC standards make two browsers establish and control their communication [18], since signalling is left to the developer to make it more flexible to application providers to choose from the existing protocols (e.g., Session Initiation Protocol (SIP) or Extensible Messaging and Presence Protocol (XMPP)); or to create their own protocol [16], [19].

The peer detection process determines peers and coordinates communication among them; signalling is one of its most essential elements. Furthermore, it supports creating discussion among users by exchanging data through channels [11]. In addition to the fact that signalling connects the browser to a server and makes it possible for other peers to communicate through

this server, and support the Session Description Protocol (SDP) to integrate the network addresses and port numbers for exchanging media [20]. Figure (1.1) shows the architecture of this application and its signalling channel in WebRTC.

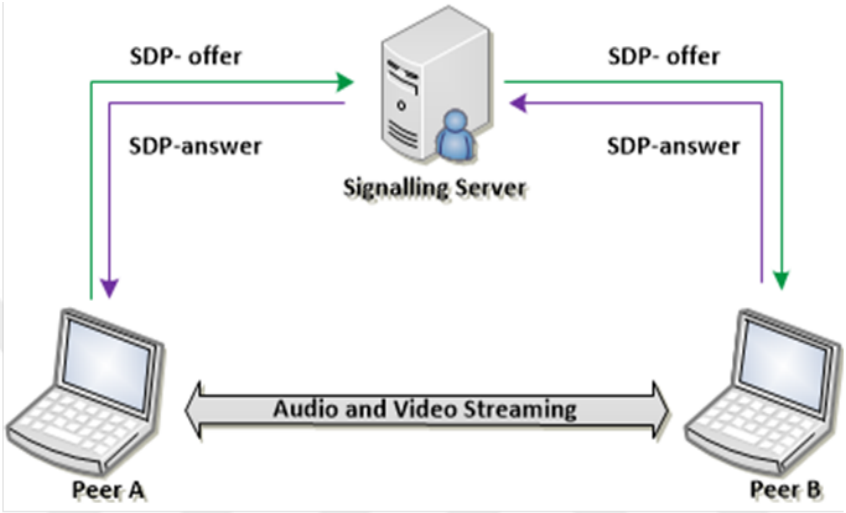

Figure 1.1: A general signalling mechanism.

WebRTC has offered different protocols such as XMLHttpRequest (XHR), XMPP, Jingle, and SIP. Some applications have used XHR, which is efficient with communication that does not need a full duplex (bi-directional) approach [21]. Similarly, it is more complicated than other protocols as [22] indicated. Moreover, an attempt to use SIP with WebRTC to obtain video calls was made by many developers, but SIP still required installation and software for such servers [23]. Additionally, WebRTC requires protocols that are not yet integrated within existing SIP clients [24], [25]. According to [26], the XMPP is a slow protocol, which may cause massive traffic through the Internet [27]. While according to [28], using Jingle have produced latency in the response of the server. Chapter two has more details on the current signalling protocols and existing implementations.

1.2 DESCRIBING THE ISSUE

[29] Mentioned that the most notable implementation issue on WebRTC is the signalling. While implementing the WebRTC, this was the first issue to be observed [30]. For this reason, the existing version of the WebRTC is not mainly endorsed "multi-browser" communication for a

conference over participating browsers [31], [32]. Communication among browsers and server has not been standardized yet using WebRTC [11], [13], [33]. As a result; this research focuses on the following challenges:

1. Reliability and flexibility using the same centralized server concurrently.
2. Selecting the appropriate browser is ambiguous, such as Firefox or Chrome, and not all browsers support the newest application of WebRTC. This affects the adoption of WebRTC by developers.
3. Solving Network Address Translation (NAT) problem to achieve communication through Local Area Network (LAN), and Wide Area Network (WAN).

This project faced different challenges while establishing WebRTC bi-directional video conferencing as described below:

1. The current WebRTC signalling protocols used browsers and networks that were established and published, it is offered in commercial servers or even proposed solutions; and it does not explain their mechanisms in details.
2. Determining the maximum and minimum bandwidth consumption and CPU load for each peer (each incoming and outgoing Real Time Protocol (RTP)).
3. WebRTC does not support Quality of Service (QoS).
4. Taking into consideration that WebRTC is being developed concurrently with the writing of this thesis, so there might be some change in the statements made in this thesis in the future version of WebRTC; however, this should not affect the overall conclusions.

1.3 RESEARCH GOALS AND OBJECTIVES

This research objective is to design and implement WebRTC bi-directional and/or unidirectional audio and video conferencing using Node.JS server in a real-world application. A flexible communication can be established for two peers using the same server using this research through (Wired and WiFi) of LAN and WAN networks. Moreover, it is possible to utilize it in different WebRTC applications without the need for registration, installation, plugins, license or cost.

The objectives of this research are:

1. Investigating how audio and video communication in VoIP work within applications. Hence, we achieved an extensive review of the most common VoIP applications on audio and video calls.
2. Designing and implementing WebRTC bi-directional and \or unidirectional audio and video conferencing based on simplex (one-to-one) communication using WebSocket protocol through the Node.js platform.
3. Accomplishing the evaluation of the bandwidth consumption, CPU performance, Quality of Experience (QoE).

1.4 THE CONTRIBUTIONS OF THE RESEARCH

The productive WebRTC audio and video conferencing do the following:

1. Offering an excellent browser performance of networking.
2. Handling a request and response among participants to exchange the relevant metadata among them.
3. Concurrently providing bi-directional and/or unidirectional video conferencing.
4. Supporting the viewer to lessen CPU load and bandwidth consumption.
5. Allowing a more profound degree of resilience in adapting a WebRTC application for a use case or scenarios such as getting two people together on one call at the same time, e-Learning between teacher and student, and m-Health between doctor and specialist or technician.

1.5 OUTLINE OF THE THESIS

The organization of this thesis is as follows:

- Chapter 2. Demonstrates significant milestones in the field of WebRTC technology background. It also indicates the general specifications and limitations of WebRTC.
- Chapter 3. The methodology and results of the authentic implementation for WebRTC audio and video conferencing, as well as creating and utilizing a WebRTC signalling mechanism based on most common WebRTC signalling protocols such as WebSocket using Node.js and based on simplex topology. This chapter ends with a comparison between WebRTC and VoIP.
- Chapter 4. The conclusion of the thesis and discussion of the point of views and recommendations for possible future work.

2. WEBRTC BACKGROUND AND LIMITATIONS

2.1 INTRODUCTION

The general research and techniques currently used and suggested for WebRTC bi-directional audio and video conferencing are reviewed in this chapter. This chapter also discusses the limitations on browser-to-browser and multi-browsers communications, and the servers and networks for multimedia/data exchange using public architecture to handle signalling protocols or public infrastructure to handle the privacy as a service or information. However, there is no explanation that shows how bi-directional audio and video conferencing can be applied in WebRTC since it has no concept of sessions; but rather, it has a concept of streams that generates and consumes media flows [16]. We researched the current experiments and propositions of WebRTC bi-directional video conferencing to discover the main gabs in this point and to obtain a greater understanding of the design and the signalling of WebRTC along with its advantages and disadvantages.

Rest of the chapter is organized as follows: section (2.2) explains the WebRTC architecture, while section (2.3) discusses the general WebRTC challenges. General limitations of WebRTC, several proposals and implementations of WebRTC audio and video conferencing and signalling mechanisms are discussed in section (2.4). Finally, section (2.5) gives the chapter summary.

2.2 WEBRTC TECHNOLOGY

During the past few years, the Internet Engineers Task Force (IETF) and the World Wide Web Consortium (W3C) established a new standard and called it Web Real-Time Communication (WebRTC). According to [11], Web Real-Time Communication (WebRTC), which is an open source and a group of libraries, JavaScript Application Programming Interfaces (APIs) and standards , that provides interactive communications of audio, video, and data among browsers and web users. In other words, [34] mentioned that WebRTC is a peer-to-peer (P2P) rather than a client/server protocol. In WebRTC, JavaScript APIs are directly used to provide support for interactive communications among browsers using different kinds of data [13], [29], [35]. Data does not depend on a server; but it allows a direct communication channel among web browsers [36]. Moreover, it has several benefits in that it does not require plug-ins or installation, and it is easy to use, as well as it is free, and does not require licensing and high-quality Real Time-

Communication (RTC) application [12], [14]. [37] mentioned that WebRTC can solely support one-to-one communication. Furthermore, it does not provide any interfaces or tools to monitor P2P structures [38], [39]. WebRTC does not offer functions for connection management [40]. WebRTC is not a service, but rather it is a set of APIs that enables WWW developers to design their specific applications via WWW standard techniques [41].

According to [42], [43], WebRTC consists of three major components as follows:

- GetUserMedia API .
- RTCPeerConnection API .
- RTCDataChannel API .

A deep explanation of the components has done in section (2.3). Even though WebRTC is using "RTCPeerConnection" API to communicate data streaming among peers, but a signalling mechanism is not a part of it [37], [44]. Therefore, the developer needs to provide the signalling protocol in the application level [45], [46]. It does not consider signalling mechanisms and protocols as a part of it because W3C and IETF have not yet reached an agreement concerning a final signalling mechanism or a protocol to test WebRTC [15] so the standard of the signalling channel in WebRTC is not yet specified [16], [17]. In another way, it is not clear how WebRTC standards make two browsers establish and control their communication [18], since signalling is left to the developer to make it more flexible to application providers to choose from the existing protocols or to create their own protocol.

There is no explanation that shows how bi-directional audio and video conferencing can be applied in WebRTC since it has no concept of sessions; but rather, it has a concept of streams that generates and consumes media flows [16]. [45] illustrated that a call setup in WebRTC is designed to focus on managing the media plan, and leaving the signalling plan action up to the application as much as possible.

2.3 WEB REAL-TIME COMMUNICATION (WEBRTC) BACKGROUND

The W3C established a new standard known as WebRTC for the API and IETF browsers, and for the wire protocol. The WebRTC allows the co-occurrence of audio, video, and data communications [11]. Developing the WebRTC started in May 2011 when Google declared an open source project to offer peer-to-peer, browser-based, real-time communications [47]–[49].

WebRTC is a collection of libraries, JavaScript APIs, and standards [12]. In other words, [34] mentioned that WebRTC is a peer-to-peer (P2P) rather than a client/server protocol. In WebRTC, JavaScript APIs are directly used to provide support for interactive communications among browsers using different kinds of data [13], [29], [35]. Furthermore, and according to [14] WebRTC does not require plug-ins, softphones, no licensing and high-quality RTC applications and this is an advantage of WebRTC, it also offers flexible solutions for voice and video conferencing capabilities [50]. While, [51] mentions that it is possible to use WebRTC within many different fields like web conferencing, video streaming, live broadcasting and voice & video calling [52], virtual classes, file sharing, security cameras, e-Health, and banking. Moreover, several authors [12], [31], [47] claimed that we could use WebRTC at e-learning, news, senior citizens, speech recognition and face detection. Where in the human life, face recognition is very useful especially in security side [53]–[56] .Equally, the WebRTC services delivered by a variety of developments include lower latency, ultra-high reliability, and guaranteed security [25]. It is possible to set or execute a secure channel or channels through Datagram Transport Layer Security (DTLS) and Secure Real-Time Protocol (SRTP) for the media channel, and also Stream Control Transmission Protocol (SCTP) over DTLS for data channels. Security challenges like data capture by hackers through bypass intermediary hardware server can be eliminated [57], and it can ensure the authentication and confidentiality of the system [58]. Consequently, [37], [41] confirmed that more than 1,000,000,000 endpoints/devices had used WebRTC. It is also expected that 4.7 billion devices will be able to support WebRTC by 2018 [36].

WebRTC could be one situation as long as it utilizes the same web page for downloading and allows both browsers to work out the same web application [38]. So, any method that allows browsers to send and receive messages via a server can be used for WebRTC signalling [59]; it only requires the use of an appropriate HTML5 API [43]. There is wider use of HTML5 with the paced evolution of web browsers led by Firefox, Chrome, and Opera, In HTML5, there are two methods for pushing data from the server to the client known as Server-Sent Events (SSE); however, it is a unidirectional protocol, while the other one is the full duplex WebSockets [21], [60]. Therefore, data does not depend on a server; but it allows a direct communication channel among web browsers [36].

There is a difference in WebRTC if we are to compare it to other video conferencing applications, for instance, WebRTC is not a service, but rather it is a set of APIs that enables WWW developers to design their specific applications via WWW standard techniques [41]. According to [42], [43], WebRTC consists of three major components as follows:

- GetUserMedia API (aka, MediaStream),
- RTCPeerConnection API (aka, PeerConnection).
- RTCDataChannel API (aka, DataChannel).

2.3.1 GetUserMedia API

This API represents synchronised streams as input and output of audio and video while it helps browsers to access local devices like microphone and camera [23]. As a result of this API, WebRTC eliminates the need to use Adobe Flash to have the media devices and the requirement of the plugin. Moreover, this API supports the employment of audio and video as HTML elements in any type of web application [1]. Figure (2.1) shows the structure of MediaStream.

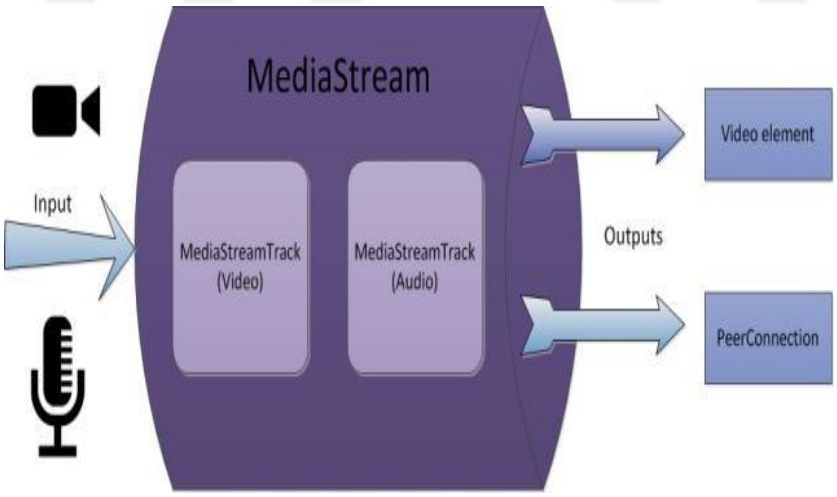

Figure 2.1: The structure of WebRTC MediaStream [23].

2.3.2 RTCPeerConnection API

RTCPeerConnectiob API makes it possible for browser-to-browser to directly communicate and set up audio or video calls. It gathers all the internal mechanisms of the browser such as media streams through getUserMedia() method that allows browsers to access the local devices like microphones and cameras to ease data transmitting to the other side [44]. Moreover, it uses

specific JavaScript methods to handle all the exchange signalling messages [1]. [61] mentioned that "PeerConnection" supports signalling and Interactive Connectivity Establishment (ICE) techniques and guarantees a high degree while establishing a successful call in different environments.

2.3.3 RTCDataChannel API

RTCDataChannel API allows browsers to transmit data connections, and presents a bi-directional data communication to offer an exchange of random data between two browsers. Every "RTCDataChannel" can reduce the cost of service and produce the network usage like low-latency data exchanges in a more flexible way while avoiding using the server and the intermediate [62]. It falls into two types:

1. Delivering reliable or unreliable messages.
2. Delivering in-order or out-of-order messages.

The Transmission Control Protocol (TCP) is identical to the reliable and in-order delivery. While the User Datagram Protocol (UDP) is similar to the unreliable and out-of-order delivery. There are several possible use cases for this API. Such as games, applications of remote desktop and real-time file transfer, and text chat. Figure (2.2) shows WebRTC architecture.

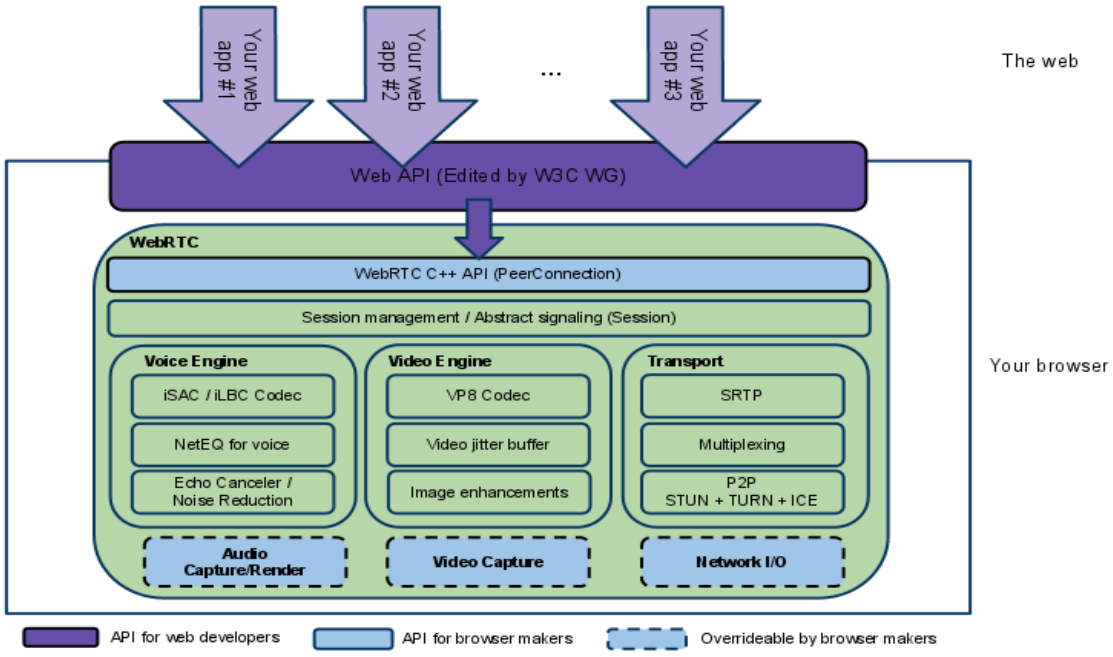

Figure 2.2: WebRTC architecture [37].

[1] mentioned that there are one or more layers of Network Address Translation (NAT) behind several devices. NAT is a technique used for planning private addresses to public addresses, and the variable address information within IP packets while sending over the routing device. It is possible for WebRTC technology developers to utilize ICE that in return, facilitates the complication of the Internet addressing system. It is certain that ICE is reliable and is massively used in media communications while selecting the best way of connectivity in limited environments. It is used for traversing NATs that relies on Session Traversal Utilities for NAT (STUN) and Traversal Using Relay NAT (TURN) [63]:

1. STUN: is used to discover an external address to a particular peer [63], [64].
2. TURN: is used to support STUN by bypassing the Symmetric NAT determination by establishing a connection with a TURN server and sending all information over it [16], [42], [64].

[45] illustrated that a call setup in WebRTC is designed to focus on managing the media plan, and leaving the signalling plan action up to the application as much as possible. The rational is that: (a) to make application developers free to choose from the existing protocols (e.g. SIP or XMPP), or to create their own protocol [19], [45], (b) to avoid repetition and not minimize the compatibility with the developed technologies [23], [25], (c) the WebRTC working group (IETF) did not want to make it something that may appear to be inappropriate for all the uses [65], (d) to give developers the freedom to design or modify the signalling protocols [18], [43], (e) not to commit to any particular technology [49], and (f) for a novel use case [45]. The mentioned arguments above that are flexible enough to offer an opportunity that permits the developers to work with old and obsolete signalling protocols like SIP [49]. Also a comparison between some protocols mentioned below in Table (2.1).

Table 2.1: Shows some differences in protocols, such as SIP, XMPP and WebSocket protocol.

SIP	XMPP	WebSocket
Particularly used for telecommunication	Utilised in the decentralised system for real-time messaging	Does not support every browser, web servers and proxies
Applied in more complicated systems with a proxy server, location server, registrar and user agent	Scalability is limited to XMPP and cannot supply modification of binary data	It suffers from reconnections in services
Messages processing can consume bandwidth	It has a raised network overhead	
The peer must register first and then can send the invitation message, which has massive information	It has a long process to establish a session	

Similarly, WebRTC makes it possible for developers to choose the signalling problems while IETF is standardizing the JavaScript Session Establishment Protocol (JSEP). JSEP permits the application to manage the signalling state via the browser and into JavaScript. For instance, the application controls the addressing, retransmission, and handling. JSEP allows the mechanisms to create an offer or an answer and respond to them during a session. However, the implementation of JSEP is entirely separated from the real mechanism by which the offerers and answers communicate with the remote side [49]. Thus, the application becomes entirely responsible for leading the signalling state machine because of the JSEP methodology [65]. Moreover, it allows the signalling or negotiation status in the web applications to operate, thus providing the real signalling protocols like (SIP and Jingle) to work on top of it [66]. The only thing required here is a way for the application to transmit a local request and negotiate remote session descriptions, and a technique to interact with the ICE [45]. Accordingly, when there is a need to exchange an offer/answer, we need to establish signalling to exchange the SDP among peers over the signalling channel [49].

2.4 GENERAL WEBRTC SPECIFICATIONS

The current version of the WebRTC is only developed to aid browser-to-browser communication, while it is not mainly built for many-browsers communication for a conference over participating browsers [31], [35], [42]. [37] mentioned that WebRTC can solely support one-to-one communication. Furthermore, it does not provide any interfaces or tools to monitor P2P structures [38], [39]. Its standards are still in a primitive stage, and they need some time to grow [41]. In addition to that, [20], [67] claimed that choosing the suitable network topology in the architectural design of the WebRTC application is one of the potential issues; thus, it must choose an architecture for the application while dealing with a multiparty of audio/video call in WebRTC. As a result; [1] showed that several topologies and devices were utilized in WebRTC such as point-to-point (Simplex) topology, and Multipoint Conferencing Unit (MCU) device.

[68] clarified that there are implementation issues among browsers because they are in alpha or beta status. Therefore, they have different bugs; for example, the existing WebRTC draft is not entirely implemented within every browser because of the interoperability issues between them. Besides, [69] confirmed that WebRTC is not completely operated with all browsers. While [62], [70]–[72] assured that WebRTC does not aid all browsers [59], [73], [74].

[66] said that a WebRTC implementation has some impediments currently through various libraries. For instance, using PeerJS API, which consists JavaScript libraries, coordinates a connection establishment in WebRTC. However, each client that demands to join the session/network including a unique ID need to register, and it can only support Google Chrome. Similarly, WebRTC does not offer functions for connection management, and its video streaming requires significant CPU usage and bandwidth consumption [40]. As a matter of fact, the signalling mechanism is the most apparent implementation problem in WebRTC, and it is the first one to be noticed while operating the WebRTC [15], [27], [29], [30], [38].

2.5 LIMITATIONS OF WEBRTC SIGNALLING MECHANISM

WebRTC needs a type of a signalling mechanism or aid from protocols to create communication among various users or devices [24], [45], [46], [75]. However, the IETF and W3C have not yet set similar opinions on a final signalling mechanism or a final API protocol to test the WebRTC and regulate the communication architecture [15], [62]. Furthermore, they have not set the signalling channels standards [16], [76]. In other words, the way two browsers create and control

their communication via WebRTC standards is not exactly defined [18], [42], [59]. Even though WebRTC is using "RTCPeerConnection" API to communicate data streaming among peers, but a signalling mechanism is not a part of it [37], [44]. In addition, [16], [19], [30], [38], [41], [51] demonstrated that signalling has been seen as the main component of the application, which has not yet been specified in WebRTC between browser and server. Therefore, the developer needs to provide the signalling protocol in the application level [45], [46]; and the two entities have to agree on it either with the central node or with the other user [1]. Consequently, it can be confirmed that the signalling among browsers and servers is not standardized in WebRTC [11], [13], [47]. Not only that but also the client-server architecture does not seem to be a reachable solution in WebRTC [36].

[37], [62], [77] clarified that WebRTC applications need signalling to set up a call and the users require signalling to transfer various types of information as follows:

1. Session Control Messages are used to start or end communication.
2. Network Configuration is related to the real world that gives the IP address and port to the peer.
3. Media Capabilities consists of the audio and video codecs, bandwidth, and resolutions that can handle the browser that is communicated with.

Using a signalling mechanism sets the communications between two peers [68] to find each other, establish the direct streams for P2P, and exchange contact details [59]. Accordingly, signalling is the core component of the peer detection that finds peers and establishes communication among them. It supports establishing communication among users by exchanging data through channels [11], and it connects the browser to a server and allows the other peers to communicate through this server, including supporting the SDP for integrating the network addresses and porting numbers for the media exchange [20], [43].

SDP is an integral part of WebRTC. Thus, it must operate before making a browser-to-browser connection. This includes the client’s IP address and all other data related to the data, audio, or video that a client requires sharing back and forth with the serve[78]. The developer is free to specify the signalling setup [62]. Moreover, [33] emphasised that the most confusing side when

using WebRTC is the signalling and typically is applied in communication systems to express that setting up a call is achieved by the information provided and used.

There are many platforms for WebRTC that support video chat, such as SimpleWebRTC, easyRTC, and webRTC.io. However, they have some restrictions while some of them are not for free, and others use their infrastructure to deal with the signalling, service or information that can be easily reached [37]. Apart from this, several suggestions and different protocols were used in WebRTC to get a signalling mechanism as described below:

2.5.1 Socket.io Signalling Protocol

This is a transport protocol written in JavaScript to allow real-time bidirectional communication between web browsers and a server [79]. Socket.io initiates in Node.js and continues over Node.js as a primary backend [21]. Node.js is an open source server platform established by Google Chrome to offer high-performance JavaScript engine. Furthermore, it is a good platform for building a web application, and it can handle high throughput and performance since it is based on non-blocking Input/Output [80]. As a result, it provides an HTTP server to support establishing a web application but is not a web server like Apache HTTP, and it is not an application server like Tomcat [21]. Node.js is one of the most well-known selection on the web for bi-directional type of test since it offers a massive number of services [59]. It is easy to learn, and it has freedom in building applications and active community [81].

[82] explained that socket.io allows developers to utilize WebSockets and determine various synchronised communication techniques managed by the client's browser [21].

2.5.2 WebSocket Protocol (WS)

This protocol helps with bi-directional communication to open an interactive communication session between the user's browser and a server [83]. It connects to a server by opening a pipe instead of another peer. Then, a client sends messages to the server, and the server forwards them to the participants [57]. It has many advantages, such as low data loss rate and much user concurrency [33]. In addition, [47], [83], [84] stated that WS uses the same communication port as HTTP (80/443) and is also a primary connection and handshaking between a web client and a web server as long as it uses the web server as a proxy for connections to clients [85]. WS allows messages from both directions and leaves the session open. Thus, users can have so many

messages travelling in both directions [23]. Besides, [86] indicated that WS offers a tunnel and leave it opened among peers. Equally, it has better performance as compared with AJAX/Polling in terms of lower latency and higher user data bandwidth [87], [88]. On the other hand, [15] explained that WS is not compatible with all browsers. In addition, it does not confirm scalability and reliability when the WebRTC application is employed by a large number of peers or when joining and leaving the network [19].

[89] demonstrated that the WebRTC chat application uses WebSocket as a signalling server based on the designed and implemented node.js. But, there was an issue among clients and the web server because the server is unable to recognize the client's messages or where to send them or not, and the setup was not correct.

2.5.3 The Most Common WebRTC Video Conferencing Application

Microsoft designed this protocol to generate and transmit requests based on XMLHttpRequest signalling protocol, and to allow asynchronous data transfer between clients and servers. Nevertheless, [90] confirmed that it does not support all features of the current browsers; and it is more complicated than the Jingle and XMPP protocols [22]. [91] stated that XHR has lower throughput as compared with WebSocket, and it is affected by the overhead.

As highlighted in [92], the Google WebRTC designed an open source simple application called AppRTC that was designed and tested for WebRTC. AppRTC was deployed as a new video conferencing room that is automatically opened for the user and is then shared through the provided URL with another user using Google Chrome, Opera or Firefox. The reason why the AppRTC is designed is to find out its shortcomings over network connectivity and browser. Thus, it has the following shortcomings:

1. It uses the canonical AppRTC ("apprtc.appspot.com") leading to high bandwidth usage.
2. Its communication between two peers does not last more than 30 minutes by default.
3. It has poor-quality video since the browser utilizes a 640×480 resolution by default.

App Engine app sends the token to client A, and then client A opens a socket and listens to the channel set up on the server to send a message to client B as shown in Figures (2.3).

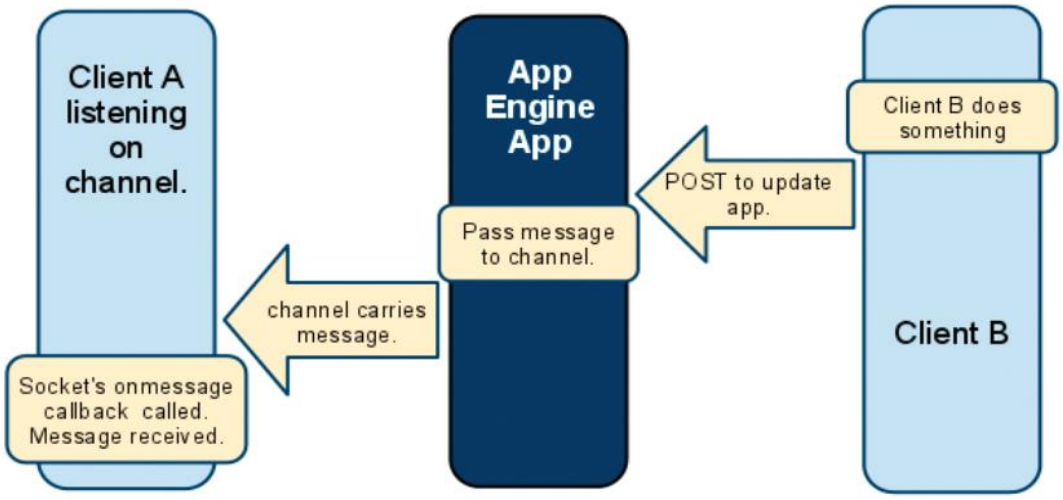

Figure 2.3: Google channel API for sending a message [135].

The most popular applications listed in Table (2.2).

Table 2.2: Showing the most popular servers and platform for WebRTC.

Commercial Tools			**The contribution of this thesis**
1	**appear.in** [93]	• Its signalling mechanism has not presented • Need to pay for a premium room to offer video conferencing	• Free of charge • Signalling has explain completely • Clients do not need to waste time to establish connection • Sound does not show any interrupt or connection drop • Does not need to specified memory usage • Does not consume high bandwidth • Does not use long polling or Plugins • Does not require complex or enormous efforts • Does not need to downloading or installation to access the service
2	**Ably** [94]	• Its signalling mechanism has not presented • A client needs at least 2 minutes to re-establish a connection • Not free of charge	
3	**APE Project** [95]	• Its signalling mechanism has not presented. Therefore, it might use plugins.	
4	**Cisco WebEx** [96]	• Its signalling mechanism has not presented. Therefore, it might use plugins.	
5	**EasyRTC** [97]	• Its signalling mechanism has not presented. Therefore, it might use plugins. • Using long polling which consumes high bandwidth • Sound shows interrupted and	

		connections can be dropped • It is specified by the memory usage • Not free of charge	• It has no network issues • It is not sensitive
6	**Firebase** [98]	• Its signalling mechanism has not presented. Therefore, it might use plugins. • Not free of charge	
7	**Firehose.io** [99]	• Its signalling mechanism has not presented. Therefore, it might use plugins. • Using Long-polling which consumes high bandwidth	
8	**Fanout** [100]	• Its signalling mechanism has not presented. Therefore, it might use plugins. • Used for messages communication • Not free of charge	
9	**GStreamer** [101]	• Its signalling mechanism has not presented and uses a plug-in architecture to handle protocols • Extremely complex and requires enormous efforts	
10	**Internet Relay Chat (IRC)** [102]	• Its signalling mechanism has not presented. Therefore, it might use plugins. • Designed to offer text messaging	
11	**IceLink** [103]	• Its signalling mechanism has not presented. Therefore, it might use plugins. • Not free of charge	
12	**Meteor** [104]	• Its signalling mechanism has not presented. Therefore, it might use plugins. • Not free of charge	
13	**OnSip** [105]	• Its signalling mechanism has not presented. Therefore, it might use plugins. • Not free of charge	

14	**PeerJS** [106]	• Its signalling mechanism has not presented. Therefore, it might use plugins. • Supports a connection between two clients	
15	**Pusher** [107]	• Its signalling mechanism has not presented. Therefore, it might use plugins. • Not free of charge	
16	**PubNub** [108]	• Its signalling mechanism has not presented. Therefore, it might use plugins. • Using Long-polling which consumes high bandwidth for limited messages • Not free of charge	
17	**PowerMedia XMS SERVER** [109]	• Its signalling mechanism has not presented. Therefore, it might use plugins. • Not free of charge	
18	**Reapply**	• Its signalling mechanism has not presented. Therefore, it might use plugins. • Used for messages • Not free of charge	
19	**Realtime.co** [110]	• Its signalling mechanism has not presented. Therefore, it might use plugins. • Not free of charge	
20	**RTCweb.in** [111]	• Its signalling mechanism has not presented. Therefore, it might use plugins. • Not free of charge	
21	**Skype** [112]	• Its signalling mechanism has not presented. Therefore, it might use plugins. • It needs downloading and installation to access the service	
22	**SightCall** [113]	• Its signalling mechanism has not presented. Therefore, it might use plugins.	
23	**Sonus** [114]	• Its signalling mechanism has not presented. Therefore, it might use plugins. • Not free of charge	

24	**SignalR** [115]	• Its signalling mechanism has not presented. Therefore, it might use plugins.	
25	**Sword** [116]	• Its signalling mechanism has not presented. Therefore, it might use plugins. • Using for voice call	
26	**TokBox** [117]	• Its signalling mechanism has not presented. Therefore, it might use plugins. • Not free of charge	
27	**Trident** [118]	• Its signalling mechanism has not presented. Therefore, it might use plugins. • Using for voice call	
28	**Talky** [119]	• Its signalling mechanism has not presented. Therefore, it might use plugins.	
29	**Uberconference** [120]	• Its signalling mechanism has not presented. Therefore, it might use plugins. • Using for voice calls • Not free of charge	
30	**vLine** [121]	• Its signalling mechanism has not presented. Therefore, it might use plugins.	
31	**Vidyo.io** [122]	• Its signalling mechanism has not presented. Therefore, it might use plugins. • Not free of charge	
32	**Valid8** [123]	• Its signalling mechanism has not presented. Therefore, it might use plugins. • Not free of charge	
33	**Voximplant** [124]	• Its signalling mechanism has not presented. Therefore, it might use plugins. • Not free of charge	
34	**XirSys** [125]	• Its signalling mechanism has not presented. Therefore, it might use plugins. • Not free of charge	

35	**OnSip** [105]	• Its signalling mechanism has not presented. Therefore, it might use plugins. • Not free of charge	
36	**Napster** [126]	• Focuses only on music files, encoded in MP3 format	
37	**Content Distribution Network (CDN)** [127]	• It is expensive • It has several network issues	
38	**Past** [128]	• Can be used only to store files	
39	**Gnutella** [126]	• Provides only file sharing services • Suffers from low response time, security, load balancing and fault active error	

2.6 CHAPTER SUMMARY

This chapter reviewed the diverse types of WebRTC signalling mechanisms/protocols, applications, and suggested solutions. Furthermore, deep research was done over its limitations. Specifically, WebRTC audio and video conferencing and signalling mechanism are required while the IETF and W3C have not yet set opinion on the exact signalling mechanism or a final protocol to implement on WebRTC and regulate the communication architecture. Equally, the existing applications depend on the public signalling server, signalling service, libraries or APIs that a signalling protocol provides. This literature review primarily addresses the achievement of its design, test or results without limitations on the clarification of how it is designed or used in the signalling mechanisms/protocols to determine communications among peers to find one another, establish the direct streams and exchange contact details of session; especially, most of them were used one of the existing signalling server, commercial server, external hardware or plugins; except AppRTC as explained in (2.4.2).

The main objective of this chapter is to provide a short background on the design of video conferencing and signalling mechanisms/protocols used in WebRTC and to prove that it is crucial to design a WebRTC audio and video conferencing that can offer various features at for two users concurrently.

3. WEBRTC IMPLEMENTATION AND COMPARISON

3.1 INTRODUCTION

This chapter revolves around the design and implement of WebRTC bi-directional and\ unidirectional conferencing, and signalling mechanism, also the way it has been developed in order to learn and achieve audio and video conferencing between two peers, several networks such as LAN, WAN (different buildings or campuses), and using the same server, stream and application within the same server application; thus, all participants should connect to the same server.

A signalling channel between clients and server based on WebSocket protocol and using the Node.js platform was created. [57] highlighted that WebRTC technology and WebSocket are used together in most cases for signalling usages. This implementation used Google Chrome and (Wired & WiFi) of LAN and WAN networks. Moreover, this signalling channel will make four types of the control messages: “initiator”, “getting media stream”, “peerChannel” and "exchange SDP (Session Description Protocol) to fully communicate between browsers. In the same way, it is beneficial to understand the working mechanism of WebRTC (API), making an offer and answer functions as the primary strategy to exchange SDP between peers, designing video conferencing to be run entirely within the web browser without the need for any native plugins. In the same way, WebRTC video streaming needs essential requirements on CPU usage and bandwidth consumption [40]. For this reason, an evaluation of each experiment based on a used Node.JS server, CPU performance, bandwidth consumption, and Quality of Experience (QoE) has been considered. Finally, a comparison between VoIP and WebRTC was done.

The chapter sections are organised as follows: in section (3.1), presents the design and implementation of WebRTC bi-directional audio and video conferencing based on simplex (one-to-one) topology. Additionally, section (3.2) displays the results analysis of simplex implementation. A comparison between WebRTC and VoIP will be illustrated in section (3.3). Finally, section (3.4) gives the chapter summary.

3.1.1 Design And Implementation Tools

This implementation used Wireshark analyser to find out the bandwidth consumption, task manager to evaluate a CPU performance and JavaScript language. Furthermore, two computers were connected through (Wired and WiFi) of LAN and WAN networks. This execution consists of two main parts: (a) the main browser, (b) a signalling channel, and making an audio and video call. The primary browser has many elements, as follows:

- Two video elements to display the local and remote videos
- Two input elements to start and terminate video call
- The script elements to get streaming audio and video
- A network element to add network IP addresses and port

The signalling channel is used to open a channel between peers (offeror and answerer) and server through sending the offer to a server to be forwarded to another peer. The offeror is a peer who generates and streams the session description to connect to another peer. In contrast, the answerer is asked for the connection from the offerer. In other words, the answer is a peer who receives a session description from another peer describing features of media communication and then responds with its session description [129]. Accordingly, a test-bed lab was created to design and implement WebRTC video conferencing for browser-to-browser. The signalling channel was created on the client side and server side to open, establish, and close communication between peers, as shown in Figure (3.1).

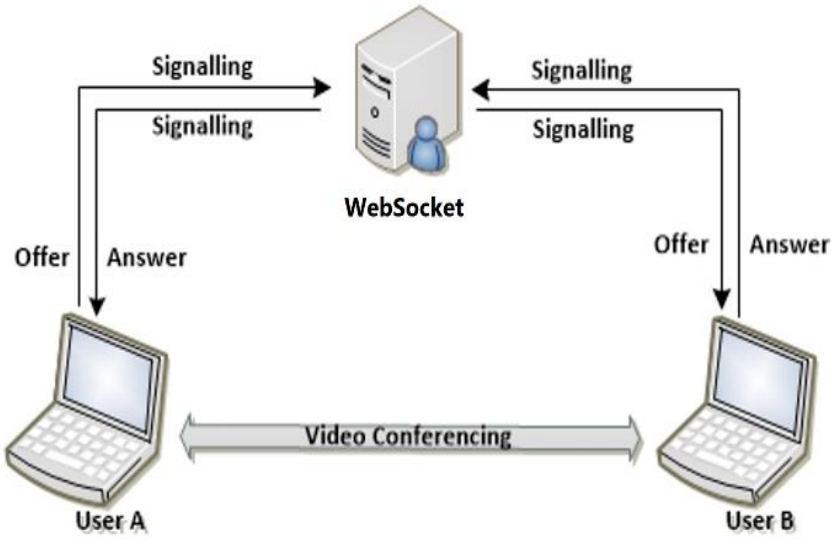

Figure 3.1: The architecture via WebSocket server**.**

3.1.1.1 Client's side

The main web page browser was built using JavaScript language and Notepad++ editor. Before a connection became established, peer A sets the local description using "MediaStream" which obtained using "navigator.getUserMedia" method then a web browser will request permission to access the camera and microphone. Once the permission is granted; a camera will start streaming, and then the test would be ready for peer B to join. Also, peer B needs to invoke "getUserMedia" and shares the camera and microphone as well. Each single video element used two video parts, local video and remote video. Therefore, when the remote peer adds a stream, it will be handed to the local video. Nevertheless, when it removes a stream, it will be hidden to the local peer. Four buttons were designed to lead the communication between peers as presented: (1) Play video, (2) Stop video, (3) Connection, and (4) Hanging Up.

In the beginning, the exchange of local and remote descriptions should be performed (e.g. the audio and video media information). It has been set up via "RTCPeerConnection" method for both peers individually to handle the streaming of video and to show the peer-to-peer connection. Thus, the caller creates an offer using "peerConn.createOffer" by "peerConn.setLocalDescription (offer)" methods and send it to the destination through a signalling channel service. Moreover, the callee that receives the offer needs to create an answer using "peerConn.setRemoteDescription", "peerConn.createAnswer" by "peerConn.setLocalDescription (answer)" to send it back to the server to be forwarded to the caller. As a result, the caller will use "peerConn.setRemoteDescription" method to get the remote description related to a connection. In addition, "RTCDataChannel" object that shows a bi-directional data channel between peers was created using "RTCPeerConnection.createDataChannel ()" method.

The offer and answer mechanisms have been located for both peers in order to enable each peer to make a call as an offeror or to answer a call as answerer using their SDP to define the media characteristics of a call. An actual main browser for clients has been shown in Figure (3.2).

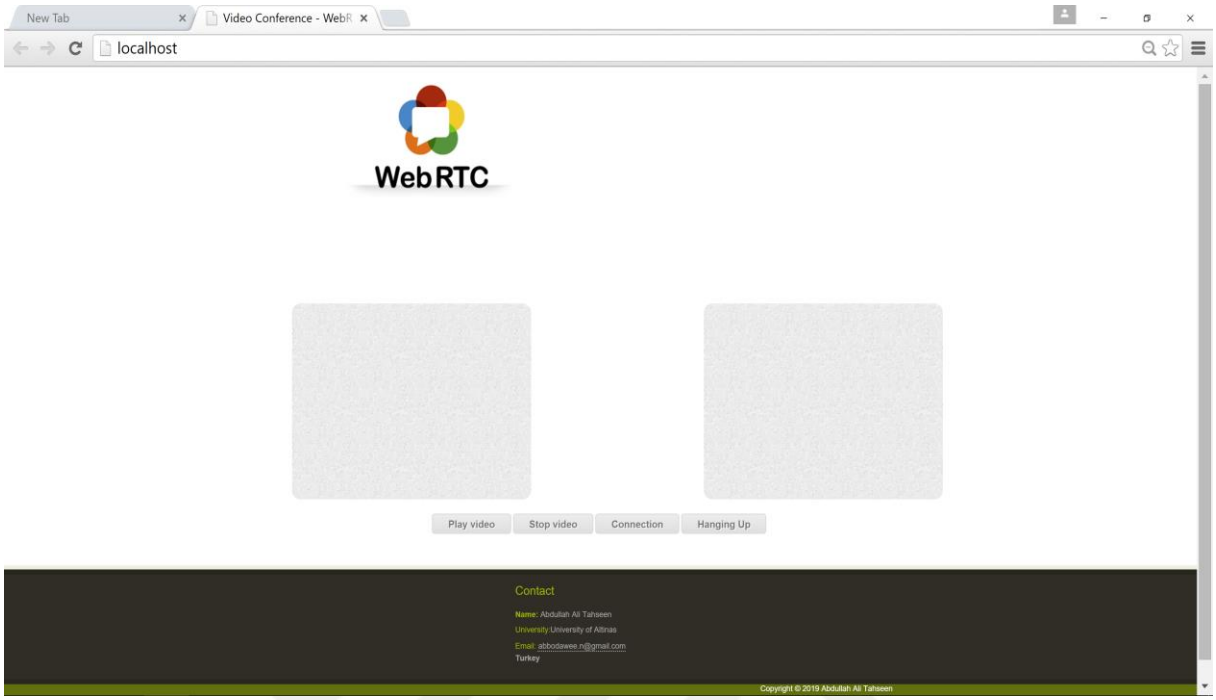

Figure 3.2: Screenshot of the primary browser for video conferencing.

3.1.1.2 Signalling channel and make an audio and video call

The signalling is essential in WebRTC for individual peers to find each other and coordinate the communication [57]. In this study, the backend (signalling channel) was created and implemented based on the WebSocket protocol using Node.js. The node.js platform is required to run the WebSocket that was running and listening on port with no error coming from the server, so is able to detect the connection from both peers. Also, Google's STUN server is used to enable a peer behind NAT and to find out its public address. Few steps were considered in order to create a signalling channel to enable communication between peers as follows:

1. Define a WebSocket server
2. Create a web server object (HTTP server object)
3. Determinate a specific port (1337).
   (A port must be selected when setting up a client or a server to allow them to meet. Also, it identifies a socket on a host.
4. Handle all messages from peers.

To process and broadcast messages to all connected peers

5. Close peer connection

Once the server is run, it shows to date, time and listening notification as “the server is listening on port 1337" as shown in Figure (3.3).

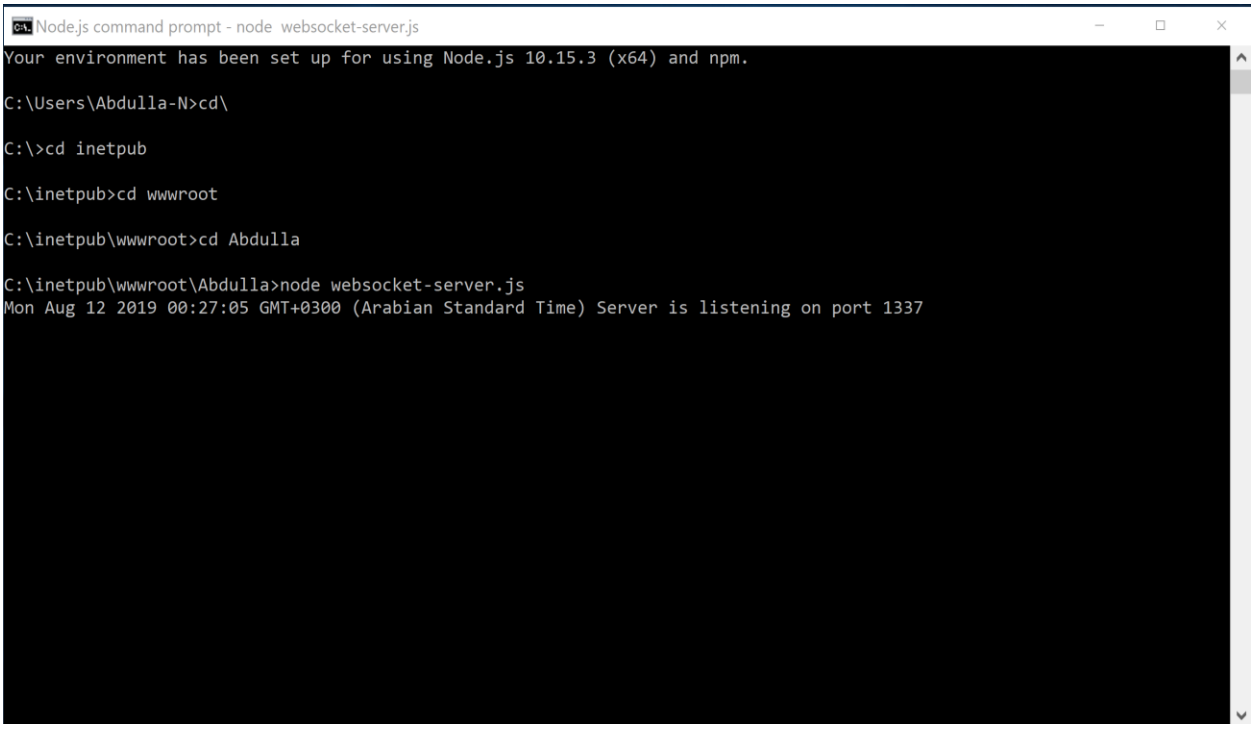

Figure 3.3: The listening of WebSocket server.

When the main browser is opened, the server will prove that "connection from origin http://localhost is done by displaying the IP of the connected peer" to be recognised as shown in Figure (3.4). [130] mentioned that a server is listening for incoming connections from the client. Moreover, after both peers A & B have connected to the server, their IPs will be presented, as shown in Figure (3.5).

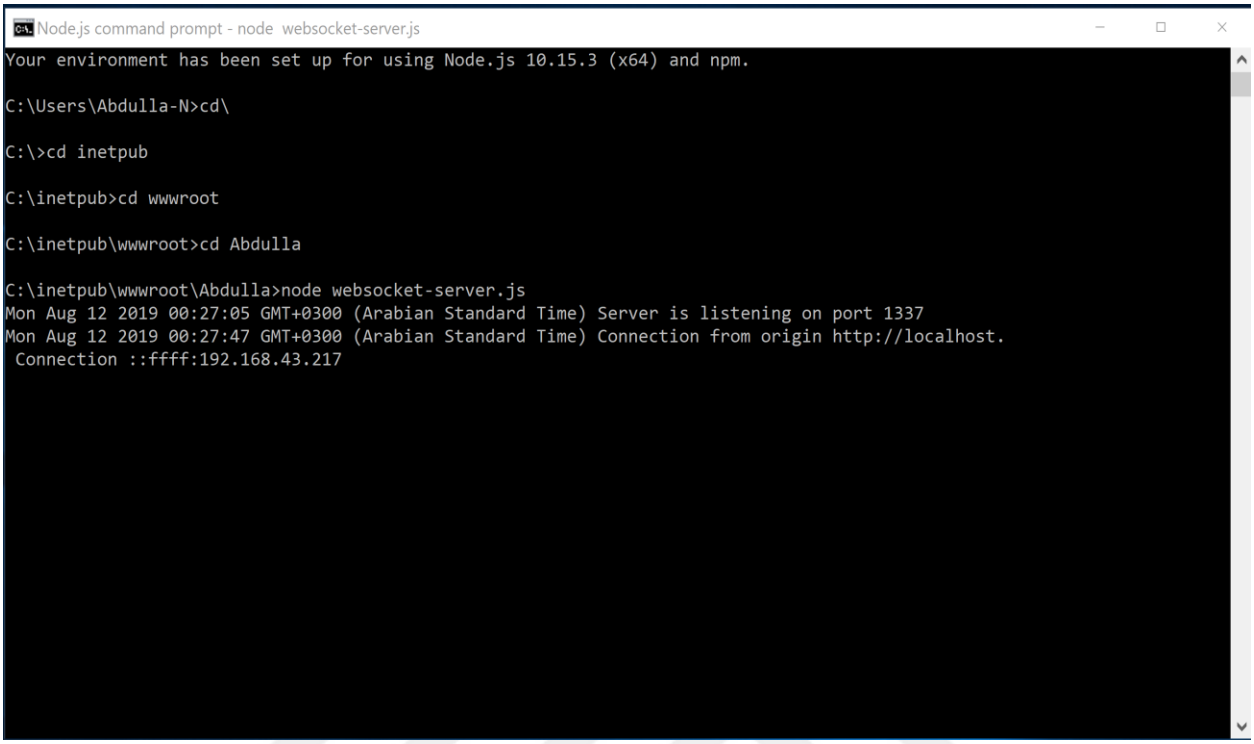

Figure 3.4: Connection of peer A with the WebSocket server.

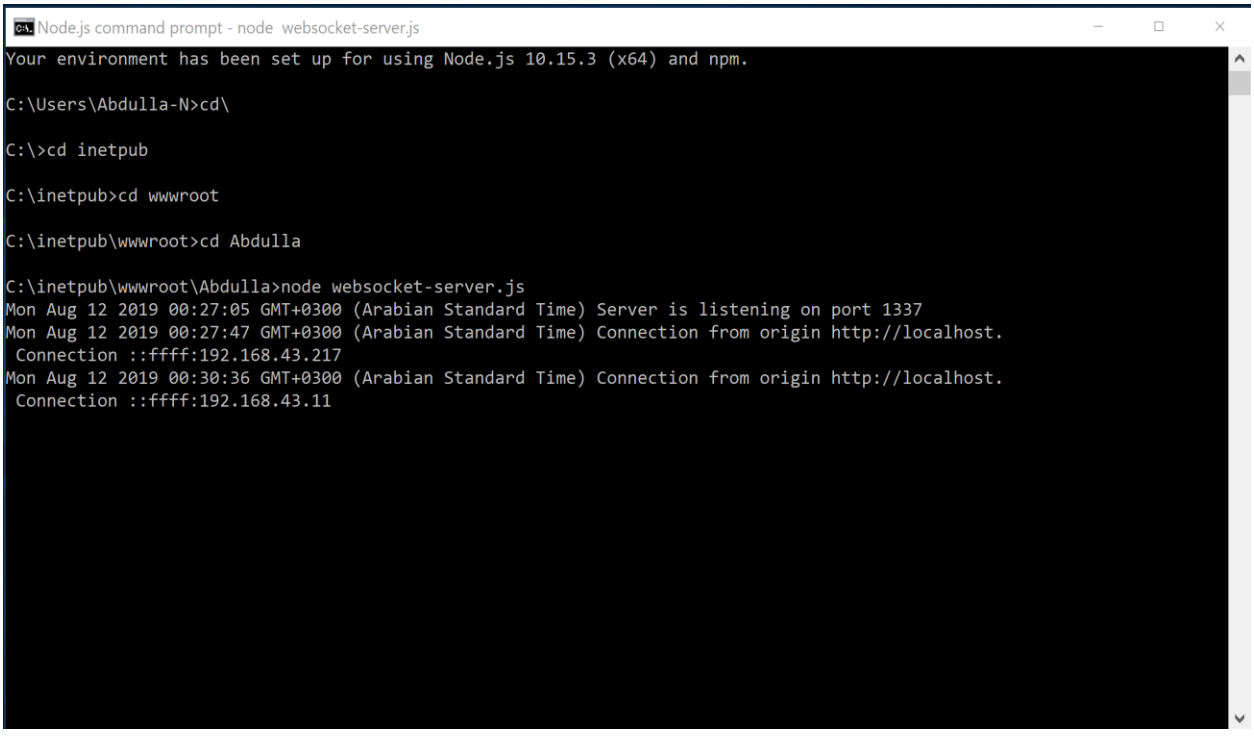

Figure 3.5: Connection of peers A & B with the WebSocket server.

After clicking on "Play video" can access the local microphone and camera; and then by clicking the "Connection", can open a peer connection that creates and send SDP-offer describing its intent to start a multimedia communication session. SDP comes to exchange call setup information between participants. It holds necessary information about a browser before the peer connection starts. This information includes; audio and video media capabilities, codecs information, user information (IP address and port numbers) transmission protocol and available bandwidth [57]. Therefore, the remote peer will receive the SDP-offer and may generate an SDP-answer accepting the offer, or it may reject the offer. In the case of offer is taken, the callee receives a view of the multimedia session which contains of the peer's information such as RTP, Real-Time Control Protocol (RTCP), ICE processing and all connected properties that can be used in the media session and then creates the answer and sends it back to the caller through signalling channel to establish a communication as shown in Figure (3.6).

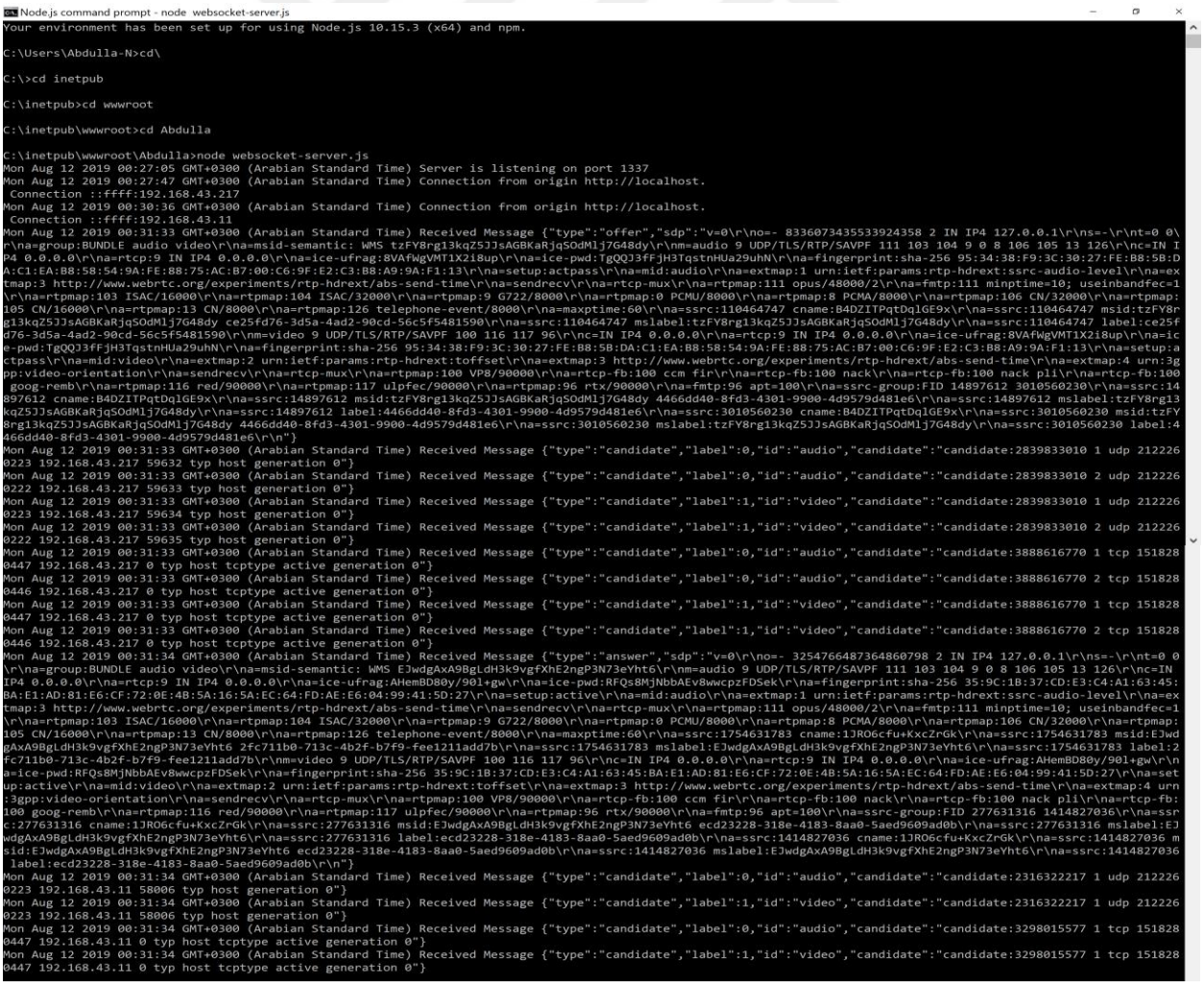

Figure 3.6: A connection establishment between peers via WebSocket server.

The signalling channel to exchange media configuration information is made by exchanging an offer, and an answer using the SDP, as well as ICE candidate was sent to each other. Therefore, the server will enable peer-to-peer communication session. Alternatively, when a peer presses "Hanging UP" button, the communication will be disconnected directly, and the remote peer will receive a message that says, "Remote hang up". Additionally, the server will show a console message stating that this IP (peer) has disconnected, as shown in Figure (3.7).

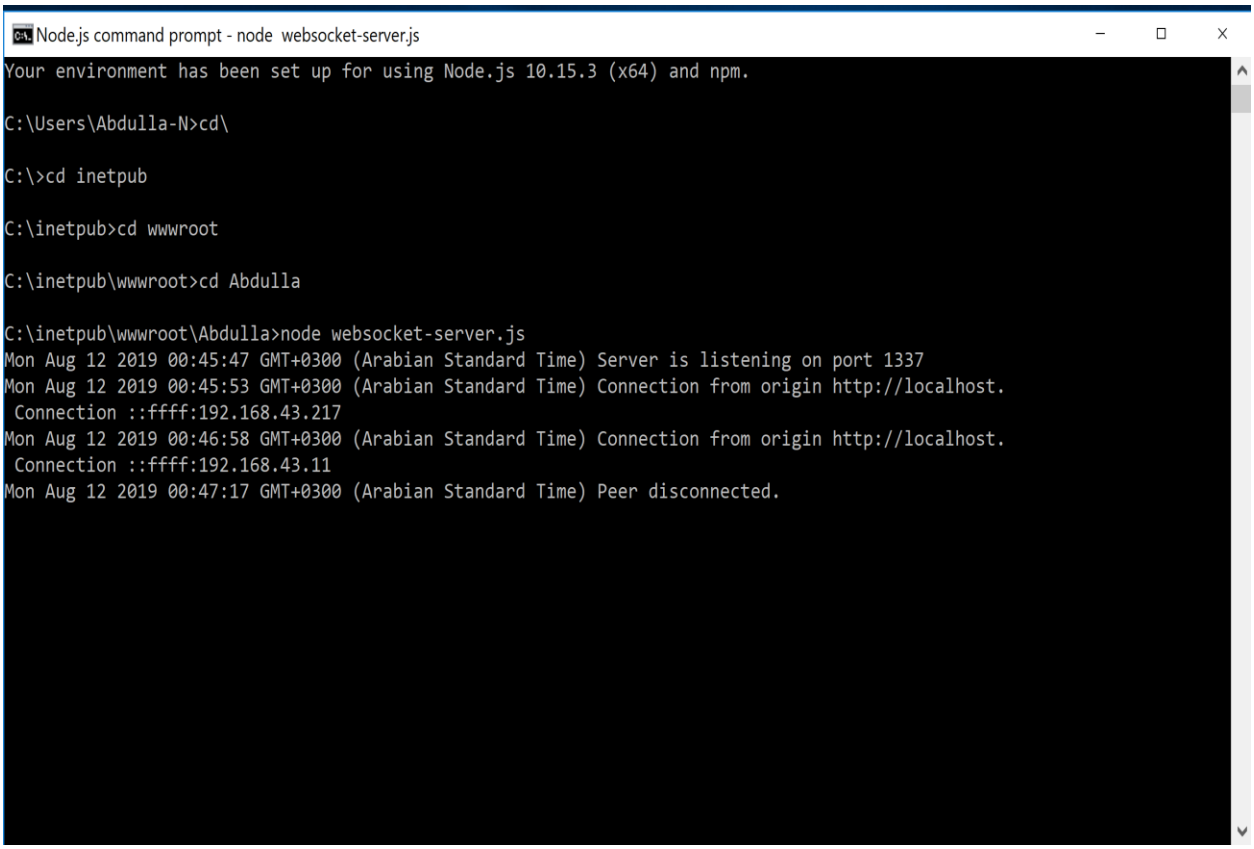

Figure 3.7: Disconnection between the peers and WebSocket server.

3.2 OUTCOMES ANALYSIS

Based on the results of this implementation, the evaluation of the created signalling channel, resources and quality of audio and video were gained as follows:

3.2.1 Signalling Channel

The signalling channel of this test was created on the client side and server side to setup communication between peers, so a productive job was presented. While, the created signalling

channel refers to the process of setting up, controlling and disconnecting a communication; also allowing two peers to start video conferencing with one another. Using this signalling, four kinds of information were exchanged: (a) Session control information, It determines when to initialise, close, and modify communication session, (b) Session control messages: used in error reporting when one peer disconnected, (c) Network data (It exposes where peers are located using IP address and port so that that caller can find callee), (d) Determined media data (To recognise the codecs and media types that the caller and callee have in common).

This signalling exchanged media configuration information between peers by using an offer and an answer in the SDP format. Hence, it shows that any peer is able to initiate or end a session, stopping self/remote stream, offering bi-directional audio and video conferencing and leaving or rejoining the session. In other words, it ensures that this infrastructure of communication is ready to create a WebRTC application.

[131] illustrated that in the application, you must know about the channel performance. Based on the inspected element of Google Chrome in real communication, the performance of the signalling channel between two peers was discovered. Thus, it analysed the delay of signalling depending on sending a request and receiving a response (establish a connection). Therefore, the minimum signalling consumption was 94 (ms) and 110 (ms) as a maximum consumption to send a request and receive a response. Figures (3.8, 3.9,), show the screenshot of the actual conferencing over LAN network, and figures (3.10, 3.11), show the screenshot of the actual conferencing over WAN network.

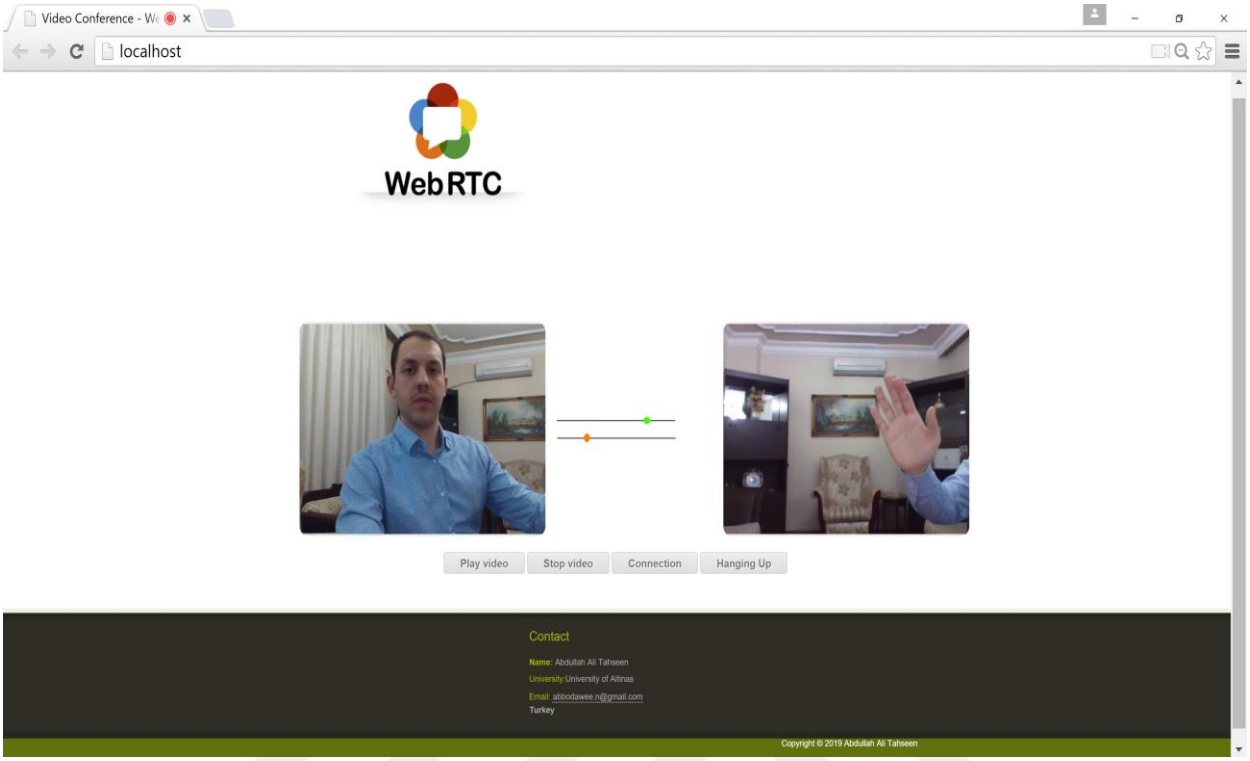

Figure 3.8: Screenshot of peers over LAN network.

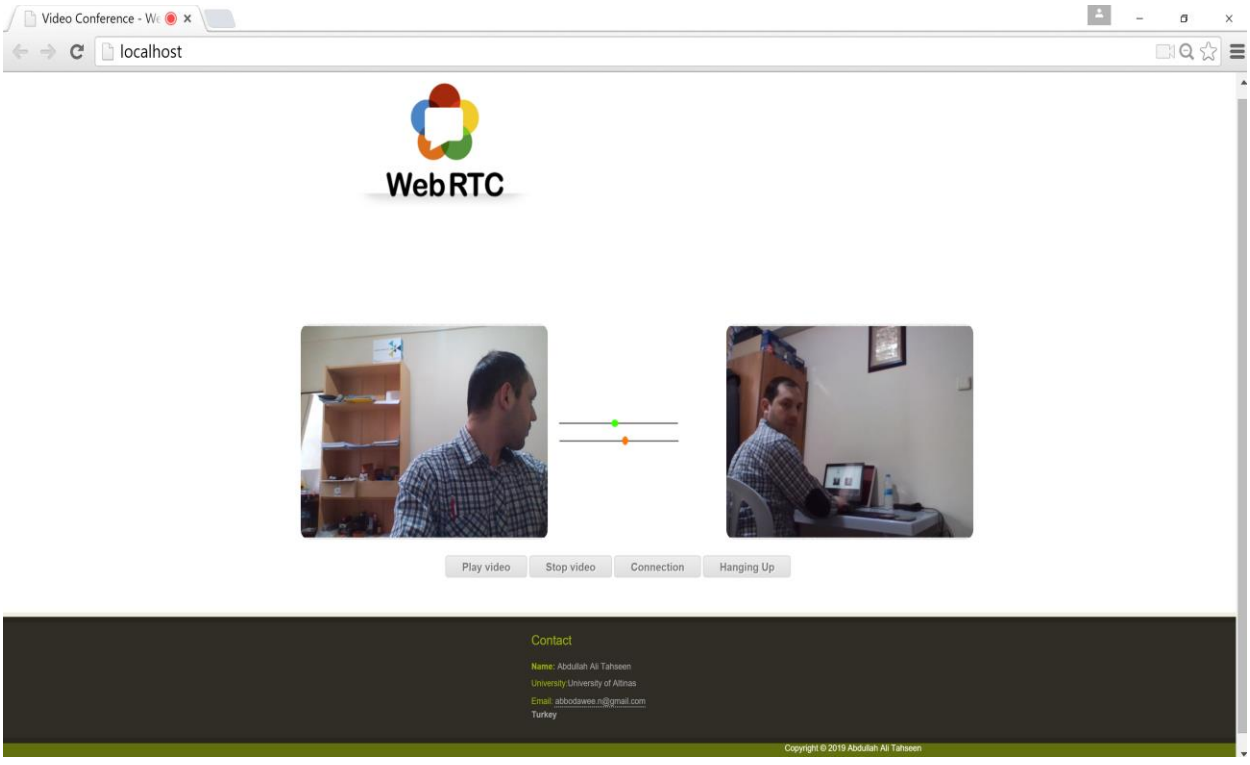

Figure 3.9: Screenshot of peers over LAN network.

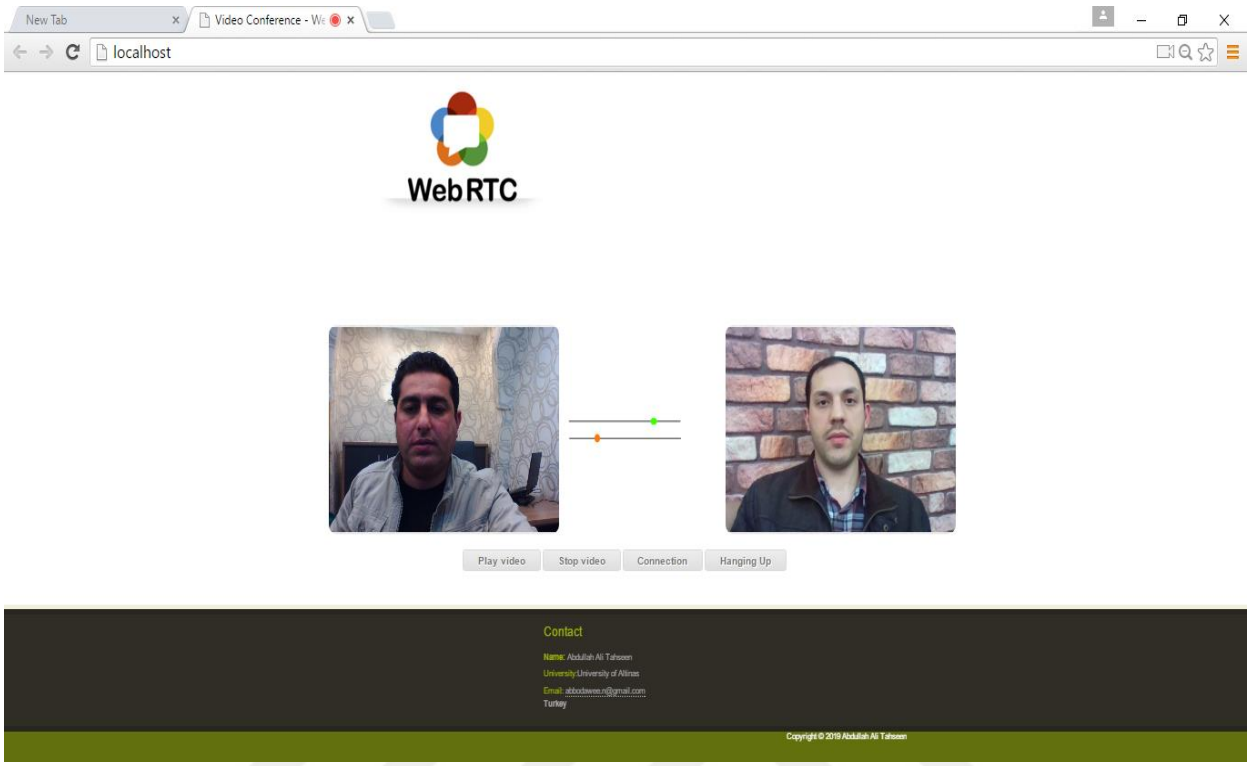

Figure 3.10: Screenshot of peers over WAN network.

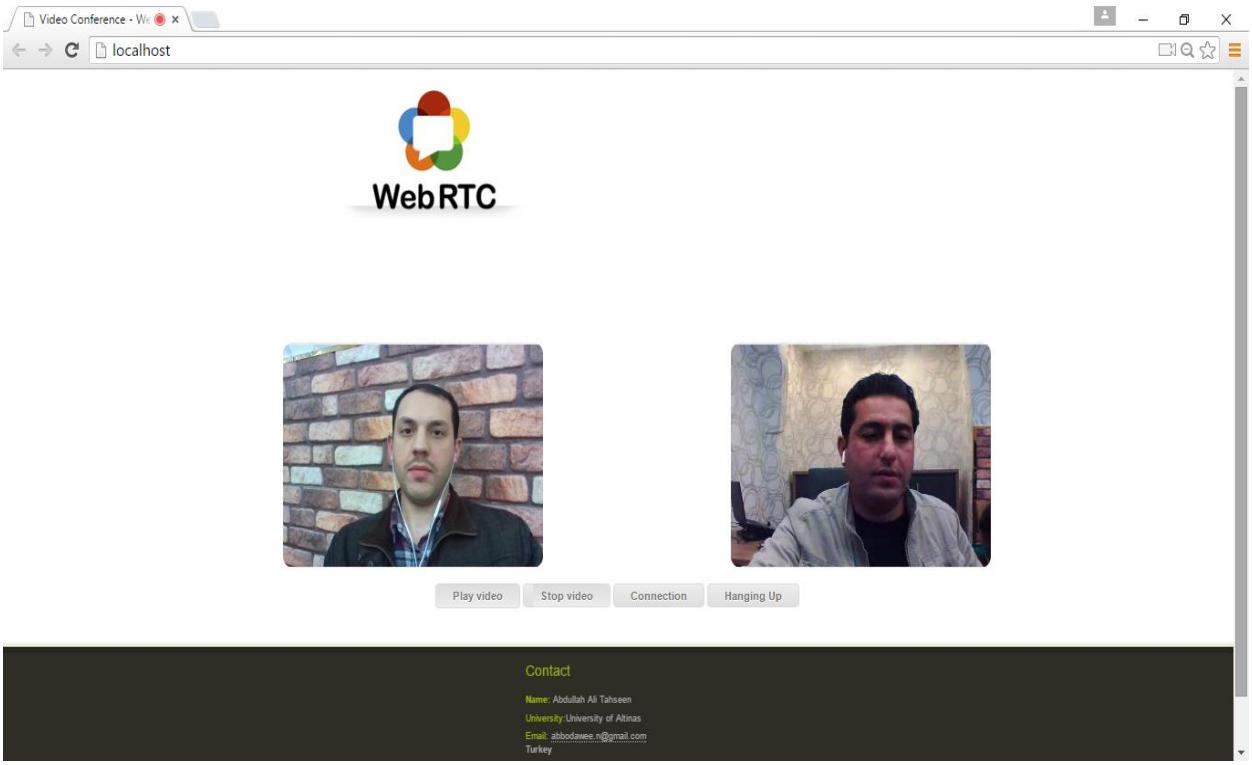

Figure 3.11: Screenshot of peers over WAN network.

3.2.2 Bandwidth Consumption

WebRTC supports various audio and video codecs, such as Opus for audio and VP8 for video (Nandakumar and Jennings, 2018). The communication was achieved during one and three minutes over (Wired & WiFi) of LAN & WAN networks, as shown in the Table (3.1).

Table 3.1: The bandwidth consumption of the audio and video between two peers through Chrome. Where the unit of bandwidth is kilobit/second (kb/s).

Network	**Over**	**Bandwidth Consumption**		
		Time	**Audio** (kb/s)	**Video** (kb/s)
Local Area Network	Ethernet	1 minute	52.662	1161
		3 minutes	53.307	1181
	Wireless	1 minute	53.842	911
		3 minutes	53.988	1246
Wide Area Network	Ethernet	1 minute	49.818	1213
		3 minutes	48.854	1158
	Wireless	1 minute	47.275	1116
		3 minutes	47.161	1025

3.2.3 CPU Performance

In this test, the performance of the CPU was measured using task manager software of Windows 10 in server side, which has both of client and server. The analysis is divided into four types:

- During one minute through (Wired & WiFi) of LAN network
- During three minutes through (Wired & WiFi) of LAN network
- During one minute through (Wired & WiFi) of WAN network
- During three minutes through (Wired & WiFi) of WAN network

Figures (3.12), shows the CPU usage.

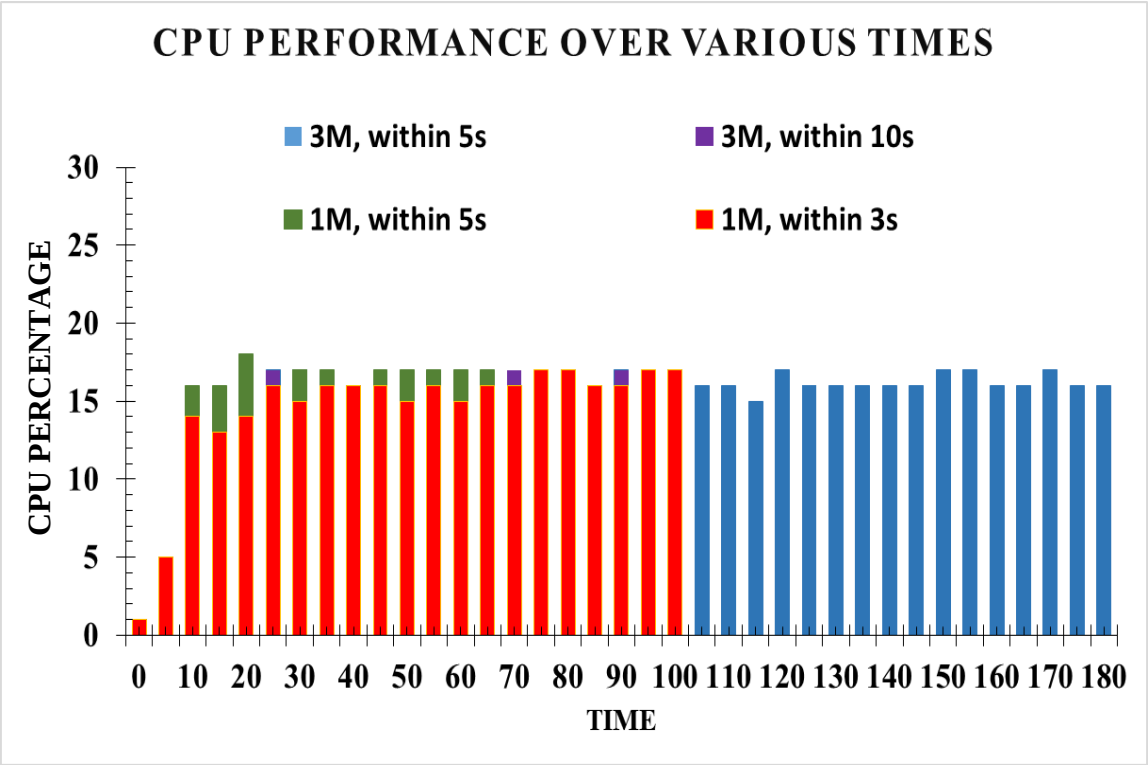

Figure 3.12: CPU performance.

3.2.4 Quality of Experience (QoE)

Actual ten users have participated in this experiment to give their individual opinions on the perceived user experience by the use of questionnaires as described in the Table (3.3). In this application, was proved that WebRTC could be used as an alternative technology for streaming media as a use case of peer-to-peer video conferencing interface.

Table 3.2: QoE via (Wired & WiFi) of LAN & WAN networks.

Questions	**Very Bad** Very annoying	**Bad** Annoying	**Fair** Slightly annoying	**Good** Perception but not annoying	**Excellent** Imperceptible
Rate the ease of using this application				**2**	**8**
Rate the quality of audio during this session				**8**	**2**
Rate the quality of the video during this session				**2**	**8**
Rate the resilience of this connection				**4**	**6**

3.3 A COMPARISON BETWEEN WEBRTC AND VOIP

According to the enormous studying and views that were done over VoIP signalling protocols, and the experiments that were performed on WebRTC in this chapter. It has been revealed that WebRTC and VoIP both share the same transport standards such as RTP and UDP, or TCP. Additionally, WebRTC is an extension of VoIP, as shown in Figure (3.13), they both support audio and video calls and data exchange. On the other hand, the big difference between them remains in their infrastructure. VoIP provides signalling protocols by dividing them into two main groups: user-to-server and user-to-user [133]. Likewise, [134] mentioned that VoIP has complicated signalling and is critical to delay, lost packets and jitter. In contrast, WebRTC works completely within the most widely used web browsers like Firefox, Chrome and Opera without downloading flash or web application. Besides, it does not need registration, downloading, plug-

ins, or license. With WebRTC, it is possible to embed communication applications or platforms directly into a webpage. However, it does require a signalling protocol to setup, establish and end the connection. As a result, WebRTC changes the identification of VoIP by considering its concept and exploits it into the Web browsers.

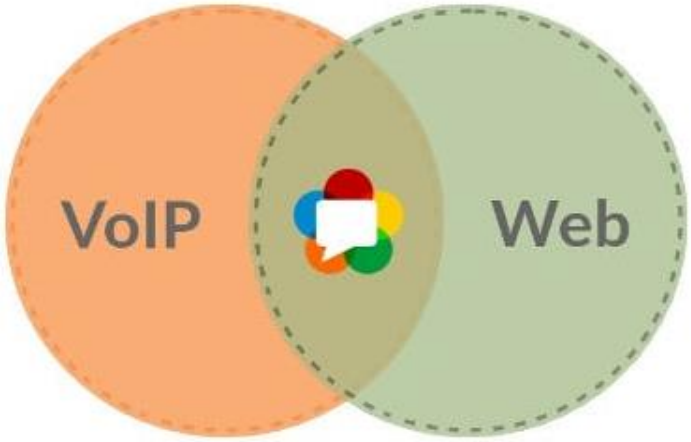

Figure 3.13: The relationship between WebRTC and VoIP [136].

3.4 CHAPTER SUMMARY

An extensive effort was given to achieve research in the most common VoIP signalling protocols with voice and video codecs. This result confirmed that the signalling protocols in VoIP have different limitations. In contrast, several implementations have been achieved to create and utilise WebRTC signalling mechanisms to offer bi-directional audio and video conferencing in real-time based on simplex topology. Therefore, WebRTC signalling mechanism based on WebSocket protocol was utilised and explained.

The contributions of this chapter support concerned researchers to learn and understand the advantages and disadvantages of VoIP, also how to design WebRTC bidirectional video conferencing.

4. CONCLUSIONS AND FUTURE WORK

4.1 CONCLUSION

In this thesis, WebRTC bi-directional and\ unidirectional audio and video conferencing have been designed and implemented using different strategies. As well as, it offers a communication using different networks (e.g. LAN & WAN) and centralised server at the same time. This application is very efficient, whereas it has been proved that is able to control setup, establish and close a connection. In other words, it can set up a call, supports a client to exchange several types of information such as SDP information, coordinates a connection establishment between peers and disconnect a communication session as well. Equally, it allows two peers to start bi-directional and/or unidirectional audio and video conferencing with one another. An additional point, the users, do not need to use any external commercial signalling server, plugins, registration, installation or fees. Accordingly, each user who would like to use this application is free to select a suitable network.

This research offers the evaluation of the performance of the following:

1. Simplex topology (One-to-One)
2. CPU performance
3. Bandwidth consumption
4. QoE by actual users
5. A comparison between WebRTC and VoIP and with most common WebRTC application like AppRTC that used in the current applications.

More importantly, this application can be applied in various fields. For instance, e-Learning between teachers, teacher and student or students, m-Health between patients, doctors, specialists and technicians, games, monitoring and live meeting.

The QoE verifies that this test-bed environment runs correctly and that it can be used to conduct more extensive experiments on user expertise in the future.

The created application in this thesis illustrated distinctive characteristics:

1. The WebRTC implementation of its clients and server,
2. Utilise an offer and answer functions as the primary strategy to exchange SDP between peers,
3. Create a signalling channel between browsers using the WebSocket protocol,
4. Offer bi-directional audio and video conferencing between browsers,
5. Using Chrome over (Wired & WiFi) of LAN & WAN networks,
6. Evaluation of resources and QoE
7. Give web developers an opportunity to comprehend the WebRTC technology, as well as to understand how to design WebRTC video conferencing.

4.2 A VISION FOR FUTURE WORK

The work explained in this thesis can be extended in many directions. This section points out some of the future research directions.

1. Develop this application to support different topologies, such as a star (One-to-Many) and mesh (Many-to-Many) topologies.
2. Develop this application to be able to opening multi rooms via different networks.
3. Develop it to be prepared for supporting video conferencing via the Internet.
4. Apply it on the future technology and smart applications such as 5G technology to get speeder Internet and massive quality of the audio and video.

REFERENCES

[1] A. A. Lozano, "Performance analysis of topologies for Web-based Real-Time Communication (WebRTC)," Aalto University, 2013.

[2] Phillippa. Biggs, "the Status of Voice Over Internet Protocol (Voip) Worldwide, 2006," *Geneva Futur. Voice*, no. January, p. 49, 2007.

[3] S. M. R. Sajjad and M. N. Dilber, "Comparative analysis of traditional telephone and VoIP systems (April, 2014)," *J. Indep. Stud. Res.*, vol. 12, no. 1, p. 25, 2014.

[4] E. M. A. A. Suliman, "UMTS VoIP Codec QoS Evaluation," *IOSR J. Electron. Commun. Eng.*, vol. 10, no. 2, pp. 07–12, 2015.

[5] U. Çelenk, D. Ç. Ertuğrul, M. Zontul, A. Elçi, and O. N. Uçan, "Dynamic Quota Calculation System (DQCS): Pricing and Quota Allocation of Telecom Customers via Data Mining Approaches," in *Handbook of Research on Contemporary Perspectives on Web-Based Systems*, IGI Global, 2018, pp. 434–459.

[6] L. Uys, "Voice over internet protocol (VoIP) as a communications tool in South African business," *African J. Bus. Manag.*, vol. 3, no. 3, pp. 89–94, 2009.

[7] B. Hartpence, *Packet Guide to Voice over IP: A system administrator's guide to VoIP technologies*. " O'Reilly Media, Inc.," 2013.

[8] S. Jadhav, H. Zhang, and Z. Huang, "Performance evaluation of quality of VoIP in WiMAX and UMTS," in *2011 12th International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2011, pp. 375–380.

[9] A. Kumar and T. Malhotra, "A multi-signaling protocol architecture for voice over IP terminal," in *IEEE INFOCOM 2004*, 2004, vol. 2, pp. 1191–1199.

[10] S. Antoniou, "VoIP Signaling Protocols," *PLURALSIGHT*, 2010. [Online]. Available: https://www.pluralsight.com/blog/it-ops/voip-signaling-protocols.

[11] J. Jang-Jaccard, S. Nepal, B. Celler, and B. Yan, "WebRTC-based video conferencing service for telehealth," *Springer*, vol. 98, no. 1–2, pp. 169–193, 2016.

[12] M. Phankokkruad and P. Jaturawat, “An evaluation of technical study and performance for real-time face detection using web real-time communication,” in *2015 International Conference on Computer, Communications, and Control Technology (I4CT)*, 2015, pp. 162–166.

[13] G. Carullo, M. Tambasco, M. Di Mauro, and M. Longo, “A Performance Evaluation of WebRTC over LTE,” *2016 12th Annu. Conf. Wirel. On-demand Netw. Syst. Serv. A*, pp. 170–175, 2016.

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

[15] C. Cola and H. Valean, “On multi-user web conference using WebRTC,” *2014 18th Int. Conf. Syst. Theory, Control Comput. ICSTCC 2014*, pp. 430–433, 2014.

[16] A. Johnston, J. Yoakum, and K. Singh, “Taking on webRTC in an enterprise,” *IEEE Commun. Mag.*, vol. 51, no. 4, pp. 48–54, 2013.

[17] S. Johansson, “Behaviour of WebRTC in Non-optimal Networks,” Luleå University of Technology, 2018.

[18] A. P. González, “Definition of A Mena Opinion Score for Vp8 Over Real-Time Connections,” 2017.

[19] 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.

[20] S. Rajab, “Comparing different network topologies for WebRTC conferencing,” *Kthroyal Inst. Technol.*, 2015.

[21] R. Rai, *Socket. IO Real-time Web Application Development*, 1st ed. Birmingham: Packt Publishing Ltd, 2013.

[22] P. Smolka, “Real-time communication in web browser.” Masaryk University,Faculty Of Informatics, p. 88, 2013.

[23] B. Sredojev, D. Samardzija, and D. Posarac, “WebRTC technology overview and

signaling solution design and implementation," in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2015, pp. 1006–1009.

[24] M. Deshpande and S. Mohani, "Integration of WebRTC with SIP–Current Trends," *ijiet.com*, vol. 6, no. 2, pp. 92–96, 2015.

[25] E. Lajtos, I., O'Byrne, D. and Ersoz, "WebRTC to complement IP Communication Services," *UK GSM Assoc.*, pp. 1–33, 2016.

[26] H. D. L. Rocha, "Hyper-linked Communications : WebRTC enabled asynchronous collaboration," Universidade de Lisboa, 2016.

[27] C. Fan, "Research on Development and Evaluation of WebRTC Signaling based on XMPP," Norwegian University of Science and Technology, 2017.

[28] A. El Hamzaoui, H. Bensaid, and A. En-Nouaary, "A formal model for WebRTC signaling using SDL," in *International Conference on Networked Systems*, 2016, pp. 202–208.

[29] A. Amirante, T. Castaldi, L. Miniero, and S. Romano, "On the seamless interaction between webRTC browsers and SIP-based conferencing systems," *IEEE Commun. Mag.*, vol. 51, no. 4, pp. 42–47, 2013.

[30] M. Schindler, C. von Harscher, J. Kinzig, and S. Alekseev, "Evaluating Framework for Monitoring and Analyzing WebRTC Peer-to-Peer Applications.," in *Proceedings of the Eleventh International Network Conference (INC)*, 2016, pp. 171–175.

[31] C. Y. Chiang, Y. L. Chen, P. S. Tsai, and S. M. Yuan, "A video conferencing system based on WebRTC for seniors," *Proc. - 1st Int. Conf. Trust. Syst. Their Appl. TSA 2014*, pp. 51–56, 2014.

[32] X. Chen, "Unified Communication and WebRTC," Norwegian University of Science and Technology, 2015.

[33] 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*, 2013, pp. 281–294.

[34] S. Hudson, "Video-to-Video Using WebRTC," in *JavaScript Creativity: Exploring the Modern Capabilities of JavaScript and HTML5*, 2014, pp. 113–124.

[35] B. Tamba, K.-C. Tseng, C. Chen, and C.-C. Tseng, "Enhancing application performance through OpenFlow enabled multi-homed devices," in *2014 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, 2014, pp. 392–397.

[36] S. Vashishth, Y. Sinha, and K. H. Babu, "Addressing Challenges in Browser Based P2P Content Sharing Framework Using WebRTC," in *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, 2016, pp. 850–857.

[37] S. Dutton, "Getting Started with WebRTC," *HTML5 Rocks*, 2014. [Online]. Available: https://www.html5rocks.com/en/tutorials/webrtc/basics/#toc-simple.

[38] M. H. Rahaman, "A Survey on Real-Time Communication for Web," *Sci. Res. J.*, vol. III, no. VII, pp. 39–45, 2015.

[39] S. A. Christian. von. Harscher, Marco. Schindler, Johannes. Kinzig, "Algorithm for Generating Peer-to-Peer Overlay Graphs based on WebRTC Events," *Proc. Elev. Int. Netw. Conf.*, no. Inc, pp. 147–151, 2016.

[40] S. Loreto and S. Pietro Romano, "Real-time communications in the web: Issues, achievements, and ongoing standardization efforts," *IEEE Internet Comput.*, vol. 16, no. 5, pp. 68–73, 2012.

[41] L. López-Fernández, B. García, M. Gallego, and F. Gortázar, "Designing and evaluating the usability of an API for real-time multimedia services in the Internet," *Multimed. Tools Appl.*, vol. 76, no. 12, pp. 14247–14304, 2017.

[42] W. Elleuch, "Models for multimedia conference between browsers based on WebRTC," *Int. Conf. Wirel. Mob. Comput. Netw. Commun.*, pp. 279–284, 2013.

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

[44] C.-F. Hsiao, “Development of a web-based distributed interactive simulation (DIS) environment using javascript,” Naval Postgraduate School, 2014.

[45] J. Uberti, C. Jennings, and E. Rescorla, “JavaScript Session Establishment Protocol,” *USA Netw. Work. Gr.*, 2019.

[46] J. Uberti, C. Jennings, and E. E. Rescorla, “JavaScript Session Establishment Protocol,” *USA Netw. Work. Gr.*, 2017.

[47] P. Segeč, P. Palúch, J. Papán, and M. Kubina, “The integration of WebRTC and SIP: way of enhancing real-time, interactive multimedia communication,” in *2014 IEEE 12th IEEE International Conference on Emerging eLearning Technologies and Applications (ICETA)*, 2014, pp. 437–442.

[48] João. Luís. Gonçalves. Domingos, “Website file download acceleration using WebRTC,” Técnico Lisboa, 2015.

[49] X. L. Calpe, “Study, design and implementation of WebRTC for a real-time,” Politècnica de Catalunya, 2017.

[50] I. Waßmann, C. Schönfeldt, and D. Tavangarian, “WIKI-LEARNIA: SOCIAL E-LEARNING IN A WEB 3.0 ENVIRONMENT.,” *Eng. Sci. Technol. Inz. i Technol.*, vol. 4, no. 1, pp. 21–27, 2014.

[51] L.-L. Tsahi, “WebRTC in the Enterprise,” *BlogGreek.Me: voxbone*, pp. 1–15, 2016.

[52] H. Patil and A. Buchade, “Review and Study of Real Time Video Collaboration Framework WEBRTC,” *Int. J. Comput. Sci. Inf. Technol.*, 2014.

[53] A. Alazzawi, O. N. Ucan, and O. Bayat, “Robust face recognition algorithm based on linear operators discrete wavelet transformation and simple linear regression,” in *Proceedings of the First International Conference on Data Science, E-learning and*

Information Systems, 2018, p. 1.

[54] E. Gumus, N. Kilic, A. Sertbas, and O. N. Ucan, "Evaluation of face recognition techniques using PCA, wavelets and SVM," *Elsevier*, vol. 37, no. 9, pp. 6404–6408, 2010.

[55] E. Gumus, N. Kilic, A. Sertbas, and O. N. Ucan, "Eigenfaces and support vector machine approaches for hybrid face recognition," *Online J. Electron. Electr. Eng.*, vol. 2, no. 4, pp. 308–310, 2010.

[56] A. Alazzawi, O. N. Ucan, and O. Bayat, "Evaluation of Face Recognition Techniques Using 2nd Order Derivative and New Feature Extraction Method Based on Linear Regression Slope," *Int. J. Comput. Sci. Netw. Secur.*, vol. 18, no. 3, pp. 169–177, 2018.

[57] E. E. Azom and B. D. Dirting, "A Peer-To-Peer Architecture For Real-Time Communication Using Webrtc," *J. Multidiscip. Eng. Sci. Stud.*, vol. 3, no. 4, pp. 1671–1683, 2017.

[58] S. El Jaouhari, A. Bouabdallah, J. M. Bonnin, and T. Lemlouma, "Securing the communications in a WoT/WebRTC-based smart healthcare architecture," *Proc. - 14th Int. Symp. Pervasive Syst. Algorithms Networks, I-SPAN 2017, 11th Int. Conf. Front. Comput. Sci. Technol. FCST 2017 3rd Int. Symp. Creat. Comput. ISCC 2017*, vol. 2017-Novem, pp. 403–408, 2017.

[59] R. Manson, *Getting Started with WebRTC*, 1st ed. Birmingham: Packt Publishing Ltd, 2013.

[60] I. Grigorik, "Server-Sent Events (SSE)," *O'Reilly Media*, 2013. [Online]. Available: https://hpbn.co/server-sent-events-sse/.

[61] C. Jakobsson, "Peer-to-peer communication in web browsers using WebRTC," UMEA University, 2015.

[62] A. Albas and G. Auguets, "WebRTC," Politècnica de Catalunya, 2016.

[63] A. Heikkinen, T. Koskela, and M. Ylianttila, "Performance evaluation of distributed data delivery on mobile devices using WebRTC," *Dubrovnik IEEE*, 2015.

[64] M. S. Nasir and K. Saeed, “A Comparison of SIP with IAX an Efficient new IP Telephony Protocol A Comparison of SIP with IAX an Efficient new IP Telephony Protocol,” *Int. Conf. Eng. Emerg. Technol. .Lahore IEEE*, 2015.

[65] J. K. Nurminen, A. J. R. Meyn, E. Jalonen, Y. Raivio, and R. G. Marrero, “P2P media streaming with HTML5 and WebRTC,” in *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2013, pp. 63–64.

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

[67] R. Picek and S. Picek, “Multipoint Web Real-Time Communication,” in *Information System Development*, Switzerland: Springer, 2014, pp. 409–420.

[68] L. L. Fernandez, M. P. Diaz, R. B. Mejias, F. J. Lopez, and J. A. Santos, “Kurento: A media server technology for convergent WWW/mobile real-time multimedia communications supporting WebRTC,” *2013 IEEE 14th Int. Symp. a World Wireless, Mob. Multimed. Networks, WoWMoM 2013*, 2013.

[69] S. Owesen-Lein, “Unified Communication and WebRTC,” Norwegian University of Science and Technology, 2015.

[70] A. Sergiienko, *WebRTC Blueprints*. Birmingham: Packt Publishing Ltd, 2014.

[71] A. Sergiienko, *WebRTC Cookbook*. Birmingham: Packt Publishing Ltd, 2015.

[72] V. Antani, S. Timms, and N. Prusty, *JavaScript: Moving to ES2015*. Packt Publishing, 2017.

[73] G. A. B. Hans W. Barz, *Multimedia Networks: Protocols, Design and Applications*, 1st ed. Wiley, 2016.

[74] Altanai., *WebRTC integrator’s guide : successfully build your very own scalable WebRTC infrastructure quickly and efficiently*. Packt Pub, 2014.

[75] J. Uberti, C. Jennings, and E. E. Rescorla, “JavaScript Session Establishment Protocol,”

USA Netw. Work. Gr., 2016.

[76] P. Ravindran, “RTCWeb standard signaling protocol,” 2011.

[77] Jahangir. Khalid, “Comparison between VoIP clients,” politecnico di milano, 2014.

[78] A. A. T. Al-Najjar, O. N. Ucan, and O. Bayat, “Design and Implementation of Bi-directional Audio and Video Conferencing in WebRTC,” *AURUM J. Eng. Syst. Archit.*, 2019."in-Review".

[79] M. Grinberg, “socketio Documentation,” *USA: Python community*, 2018. [Online]. Available: https://python-socketio.readthedocs.io/en/latest/.

[80] A. S. Karl. Bissereth, Billy B. L. Lim, “An Interactive Video Conferencing Module for e-Learning using WebRTC,” *Internetional Conf.*, pp. 1–4, 2014.

[81] N. Chrzanowska, “Why to Use Node.js,” *Nnetguru*, 2017. [Online]. Available: https://www.netguru.com/blog/pros-cons-use-node.js-backend.

[82] M. Nebra, “Socket.io: let’s go to real time,” *openclassrooms*, 2018. [Online]. Available: https://openclassrooms.com/en/courses/2504541-ultra-fast-applications-using-node-js/2505653-socket-io-let-s-go-to-real-time.

[83] I. Castillo, V. Pascual, and J. Villegas, “The websocket protocol as a transport for the session initiation protocol (sip),” *Internet Engineering Task Force (IETF)*. 2014.

[84] Lennart. Grahl, “SaltyRTC Seriously Secure WebRTC,” Münster University of Applied Sciences, 2015.

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

[86] I. Fette and A. Melnikov, “The WebSocket Protocol,” *Internet Eng. Task Force*, 2011.

[87] C. Marion and J. Jomier, “Real-time collaborative scientific WebGL visualization with WebSocket,” in *Proceedings of the 17th international conference on 3D web technology*, 2012, pp. 47–50.

[88] G. M. Karadogan, “Evaluating WebSocket and WebRTC in the Context of a Mobile Internet of Things Gateway,” KTH Royal Institute of Technology Stockholm, Sweden, 2013.

[89] M. Sørlie, “Congestion Control for WebRTC Services.” Norwegian University of Science and Technology, 2017.

[90] M. V. D. Khoa, Y. Kim, and Y. Kim, “HTML5-based Distributed and Interactive E-learning Framework,” *Korean Inst. Commun. Sci.*, pp. 978–979, 2014.

[91] S. Agarwal, “Real-time web application roadblock: Performance penalty of HTML sockets,” *IEEE Int. Conf. Commun.*, pp. 1225–1229, 2012.

[92] S. Pfeiffer, “AppRTC : Google’s WebRTC test app and its parameters,” *ginger’s thoughts*, 2014. [Online]. Available: https://gingertech.net/2014/03/19/apprtc-googles-webrtc-test-app-and-its-parameters/.

[93] P. Y. A. Golden J, “appear.in,” *Telenor Digital AS*, 2019. [Online]. Available: https://appear.in/.

[94] “Ably,” *Ably*, 2019. [Online]. Available: https://www.ably.io/.

[95] “APE Project,” *APE*, 2019. [Online]. Available: http://ape-project.org/.

[96] “Cisco WebEx,” *Cisco*, 2019. [Online]. Available: https://www.webex.com/.

[97] “Easyrtc,” *Priologic Software Inc*, 2019. [Online]. Available: https://easyrtc.com.

[98] “Firebase,” *Google*, 2019. [Online]. Available: https://firebase.google.com/.

[99] “Firehose,” 2019. [Online]. Available: http://firehose.io/.

[100] “Fanout,” 2019. [Online]. Available: https://fanout.io.

[101] “GStreamer,” 2019. [Online]. Available: https://gstreamer.freedesktop.org/documentation/application-development/introduction/gstreamer.html?gi-language=c.

[102] “Internet Relay Chat (IRC) called channels,” 2019. [Online]. Available: http://www.irchelp.org.

[103] “IceLink,” *Frozen mountain*, 2019. [Online]. Available: https://www.frozenmountain.com/products-services/icelink/.

[104] “Meteor,” 2018. [Online]. Available: https://www.meteor.com.

[105] “OnSip,” 2019. [Online]. Available: https://www.onsip.com/getonsip.

[106] “PeerJs,” 2019. [Online]. Available: https://peerjs.com.

[107] “Pusher,” 2019. [Online]. Available: https://pusher.com.

[108] “PubNub,” 2019. [Online]. Available: https://www.pubnub.com.

[109] “PowerMedia XMS,” 2019. [Online]. Available: https://www.dialogic.com/xms.

[110] “Realtime framework,” 2017. [Online]. Available: https://framework.realtime.co/messaging/.

[111] “RTCweb.in,” 2018. [Online]. Available: https://rtcweb.in.

[112] “Skype,” 2019. [Online]. Available: https://www.skype.com/.

[113] “SightCall,” 2019. [Online]. Available: https://sightcall.com.

[114] “Sonus,” 2019. [Online]. Available: https://www.sonus.net.

[115] “SignalR,” *Microsoft*, 2019. [Online]. Available: http://signalr.net/.

[116] “Sword,” 2018. [Online]. Available: https://dialo.ga/en/sword/.

[117] “TokBox,” 2019. [Online]. Available: https://tokbox.com/.

[118] “Trident,” 2018. [Online]. Available: https://dialo.ga/en/trident/.

[119] B. Zemel, L. Stout, and H. Young, “Talky,” *Talky, inc*, 2019. [Online]. Available: https://talky.io.

[120] “Uberconference,” *Dialpad,Inc*, 2019. [Online]. Available: https://www.uberconference.com.

[121] “vLine,” 2019. [Online]. Available: https://www.vline.com.au.

[122] “Vidyo.io,” 2019. [Online]. Available: https://vidyo.io.

[123] “Valid8,” 2019. [Online]. Available: https://www.valid8.com.

[124] “Voximplant,” 2019. [Online]. Available: https://voximplant.com.

[125] “XirSys,” 2019. [Online]. Available: https://www.xirsys.com.

[126] N. J. Navimipour and F. S. Milani, “A comprehensive study of the resource discovery techniques in Peer-to-Peer networks,” *Peer-to-Peer Netw. Appl.*, vol. 8, no. 3, pp. 474–492, 2015.

[127] S. Yadav, R. Yadav, and S. Mishra, “Comparison between Centralized & Decentralized Overlay Networks for Media Streaming,” *Int. J. Comput. Appl. IJCA*, vol. 5, no. 5, pp. 33–36, 2010.

[128] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, “A survey and comparison of peer-to-peer overlay network schemes.”

[129] J. Rosenberg and H. Schulzrinne, “An offer/answer model with session description protocol (SDP),” *USA: Network Working Group*. 2002.

[130] L. Kalita, “Socket Programming,” *Int. J. Comput. Sci. Inf. Technol.*, vol. 5, no. 3, pp. 4802–4807, 2014.

[131] G. Ergin, O. N. UÇAN, G. N. NAFEA, and O. BAYAT, “Adaptive Video Transmission Over Communication Networks,” *AURUM J. Eng. Syst. Archit.*, vol. 3, no. 1, pp. 97–112, 2019.

[132] C. J. Nandakumar. S, “Annotated Example SDP for WebRTC (draft-ietf-rtcweb-sdp-08),” USA, 2018.

[133] P. JC, F. B, and W. Christian, “Patterns for VoIP signaling protocol architectures,” *Proc.*

12th Eur. Pattern Lang. Programs Conf., pp. 1–13, 2007.

[134] A. Lazzez and T. Slimani, "Deployment of VoIP Technology : QoS Concerns," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 2, no. 9, pp. 3514–3521, 2013.

[135] R. CHAN, "Getting Started with WebRTC," *LOCAL CODERS*, 2015. [Online]. Available: https://localcoders.blogspot.com/2015/06/getting-started-with-webrtc.html.

[136] T. Levent-Levi, "Why Developing With WebRTC is Different than VoIP Development," *BlogGeek.me*, 2017. [Online]. Available: https://bloggeek.me/why-developing-webrtc-different-voip-development/.

APPENDIX A

WEBRTC BI-DIRECTIONAL VIDEO CONFERENCING BASED ON SIMPLEX TOPOLOGY

A.1 PSEUDOCODE OF WEBSOCKET SERVER

```
1   SET WS = WebSocket Server;
2   SET Create server;
3   SET WS.Port;
4   SET Client;
5   IF client.IP = WS.IP && client.Port = WS.Port;
6           THEN open channel;
7           ELSE IF connection = error;
8                   THEN sendCallback;
9   IF request = accept;
10          THEN push data;
11  ELSE send Callback;
12  END
13  WHILE Connection Alive;
14          THEN start bi-directional video;
15  END
16  Peer disconnected;
17  END
```

A.2 PSEUDOCODE OF PEERS

```
18  Get Local Media;
19  SWITCH Start Connection;
```

20 **CASE1**: create-offer/answer;

21 **STEP1**: create-offer;

22 **IF** peer connection is created;

23 **THEN** set local information and send message;

24 **ELSE** local information not running yet;

25 **STEP2**: process signalling;

26 **IF** offeror = listening to the server's port and IP;

27 **THEN** send request via server;

28 **ELSE** process signalling =0;

29 **STEP3**: accept request;

30 **IF** answerer accepted the SDP-offer;

31 **THEN** set remote description && set local information and send message;

32 **ELSE** offer is rejected;

33 **STEP4**: process signalling;

34 **IF** answerer = listening to the server's Port and IP;

35 **THEN** send response via server;

36 **ELSE** process signalling =0;

37 **STEP5**: ICE candidate;

38 **IF** peers = exchanged ICEs;

39 **THEN** start stream video;

40 **CASE2**: disconnect;

41 **IF** any peer disconnected;

42 **THEN** end stream;

43 **ELSE** Re-connect;

44 **END**

CURRICULUM VITAE

Abdullah Ali Tahseen Al-Najjar was born in Mosul/Iraq on April, 1993. He received the B.Eng degree in Computer Technology - Computer Communications Networks, from Al-Hadba University College in Mosul/Iraq in 2017 .