Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelor Thesis

Mohamed Badawi

Reliable Architecture of Web Real Time Communication

Fakultät Technik und Informatik
Department Informations und Elektrotechnik

Faculty of Engineering and Computer Science
Department of information and Electrical Engineering

Mohamed Badawi

Reliable Architecture of Web Real Time Communication

Bachelor Thesis based on the examination and study regulations

in the study program Bachelor of Science
at Department of Information and Electrical Engineering
of Faculty of Engineering and Computer Science
of the University of Applied Science Hamburg

Supervising examiner: Prof. Dr. Martin Becke
Second examiner: Prof. Dr. Heike Neumann

Date of delivery: 02.01.2020

Mohamed Badawi

Title of the paper
Reliable Architecture of Web Real Time Communication

Keywords
WebRTC, Leader Election algorithms, Distributed systems, P2P, Websockets, Signlaing Server, Peer Connections, Confidence Intervals, Javascript, Web Audio calls, Web Video Calls, Web Screen sharing, Social Networks, Network Topology,Single point of Failure, Reliability, STUN, TURN, MCU, SFU, XML, JSON

Abstract
This thesis is a development of a reliable Web real time communication (WebRTC) without using a MCU or SFU server, so the pipeline of the media tracks audio/video happens on the device of the call initiator in the client side which will have a single point of failure problem and it is handled by replication with leader election algorithm, the experiment happens different network connection using STUN and TURN and a test for scalability performance regarding it is a client side proposed solution.

Mohamed Badawi

Thema der Arbeit
Reliable Architecture of Web Real Time Communication

Stichworte
WebRTC, Leader Election Algorithmen, Verteilte Systeme, P2P, Websockets, Signlaing Server, Peer Verbindungen, Vertrauensintervalle, Javascript,Web Audioanrufe, Web Videoanrufe, Web Bildschirmfreigabe, Soziale Netzwerke, Netzwerktopologie, Single Point of Failure, Zuverlässigkeit, STUN, DREHEN, MCU, SFU, XML, JSON

Kurzzusammenfassung
Diese abschlussarbeit ist eine Entwicklung einer zuverlässigen Web-Echtzeit-Kommunikation (WebRTC) ohne Verwendung eines MCU- oder SFU-Servers, sodass die Pipeline

der Medientracks Audio / Video auf dem Gerät des Anrufinitiators auf der Clientseite erfolgt, das über einen einzigen verfügt Das Problem mit dem Punkt des Ausfalls wird durch Replikation mit dem Leader-Wahl-Algorithmus behoben. Bei dem Experiment werden verschiedene Netzwerkverbindungen mithilfe von STUN und TURN hergestellt, und es wird ein Test der Skalierbarkeitsleistung in Bezug auf die vom Client vorgeschlagene Lösung durchgeführt.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Nowadays professional communication and collaboration systems are important in any working environment. There are many departments and many teams. They need a way of communication especially if those teams in different countries, continents. Nowadays living far from home dose not have the communication problems as the old days thanks to the internet and the software real-time communication which allows people to communicate at any time, any place in the world.

Based on the definition of chat application by D. Henriyan et al.[1] chat application is a feature or a program on the Internet to communicate directly among Internet users who are online or who were equally using the internet. Chat applications allow users to communicate even though from a great distance. Therefore, this chat application must be real-time and supports multi platform to be used by many users. [1]

With the fast development of Internet, more and more people choose chatting tools for communication and traditional real-time chatting software is usually a desktop application program. Specific client programs are needed during application running. The browser-based real-time chatting tool does not need any additional client program and the visual communication could be conveniently realized through browser. [2]

Screen sharing aims at sending the compressed screen images from one computer to another through networks, which could support a lot of applications such as wireless display and remote assistance. In particular, this feature plays an essential role in the video conferencing applications for multi-party remote collaborations and when developing an end-to-end solution for interactive screen sharing, one need to make sure that the system has high visual quality, low delay and bandwidth cost, as well as supporting various clients. [3]

There are some commercial systems supporting screen sharing for example AirPlay of Apple Inc. and Chromecast of Google Inc., which can share the computer screen to other lightweight devices. Some recent research on cloud computing has proposed a solution of sharing the computer screen with other computers and mobile devices through the cloud. In particular, the screen needs to be virtualized and rendered on the cloud side first, and then be sent to the clients through the internet. [4]

In some early systems, the screen sharing is implemented with a Client/Server (C/S) framework, by transporting the Graphical user interface (GUI) vector instructions from the operating system to the receiver side. The receiver then decodes the instructions, reproduces and displays the screen of the sender. The solution of PC Anywhere, also based on the C/S framework, is to copy the simple GUI sections of the screen and encode them with lossless compression algorithms. Some recent screen sharing systems usually adopt the efficient image and video codecs, such as JPEG and H.264, to compress the screen images into video streams. [3]

1.2 Software real-time communication Existing Solutions

Different web based solutions offering audio/video services, web TV services, real-time communication services and social networks exist. Based on implemented services, three groups of systems can be identified: web TV/video on demand systems, web real-time communication systems and social networks. [5]

1.2.1 Video and Audio on demand

The first group of systems offers web TV and/or video on demand services. Many of these systems require usage of custom or proprietary protocols, plug-ins or client applications. On the other hand, it is possible to implement this service using web technologies and provide cross-platform usage without additional plug-ins or client applications. Some researches analyze that possibility and propose solutions for the implementation [6][7][8][9][10].These solutions are mostly based on the usage of Web Real-time Communications (WebRTC) [11] for establishing peer-to-peer (P2P) web TV systems. But these solutions don't have possibility for user interaction, and their focus is on the video streaming. Some commercial solutions such as Vimeo[1] or YouTube[2], also provide streaming features, but they don't have service that allows real-time communication between users. [5]

1.2.2 Real-Time Communication

The second group of systems offers web real-time communication features. These systems can be divided in a two additional groups: web based systems that use WebRTC and systems that use custom or proprietary protocols. The main characteristic of the WebRTC based solutions is that they support audio/video communication, chat and file transfer in a real-time, without usage of the additional plug-ins. Some researches presents options for different types of real-time communication based on WebRTC, such as video conferences, basic P2P communication between users or integration with Session Initiation Protocol (SIP) or different telecommunication systems [12][13]. These solutions are open-source or commercial for example Google Hangout. Most of the systems that use custom or proprietary protocols for communication are commercial, and they usually require usage of plug-ins and client application. Examples of these systems are Viber and Skype. [5]

1.2.3 Social Networks

The third group of systems is social networks. These systems usually provide web real-time communication feature, but they are very limited in video streaming [14]. Some social networks such as Facebook [5]

Many solutions usually require usage of client applications, but some of them only have web application that provides cross-platform usage. Systems that use WebRTC have secured communication between users, as part of the WebRTC. Those solutions have to implement own mechanism for user authentication and secure signaling [15] [16].

[1]https://vimeo.com
[2]www.youtube.com

1.3 Structure of the Thesis

As a notation Chapter 2 shows the related work on WebRTC in different prospective like network topology, time analysis , signaling mechanism and WebRTC limitations. As an introduction to the work Chapter 3 shows be the theoretical background of distributed systems, the concept of transparency on distributed systems, the architecture of how various components are to be organised and fault tolerance, then an introduction of WebRTC architecture and how it works, then will come to the point to explain the problem statement with short example of existed solutions and what is this proposed approach for solving that problem.

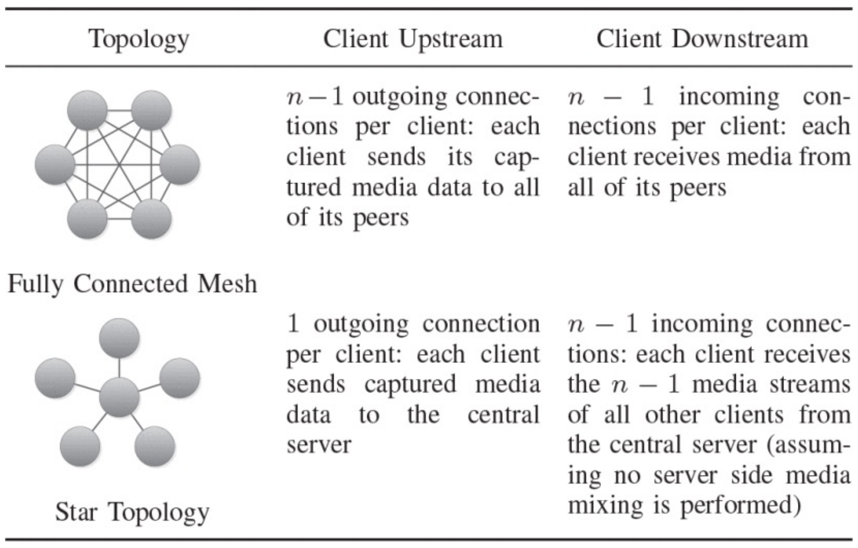
In Chapter 4 will go through the functional and behavioural requirement of the proposed approach. In Chapter 5 will show how to establish a peer connection via WebRTC with sequence diagram and what is WebRTC states with an explanation of every state. Sending data will be introduced in that chapter with some of the possible solutions and the chosen one and explanation of every signal which is playing a big part.

Databases will be introduced with a UML diagrams into Chapter 5 which is as well playing a role for establishing a call and a guard for creating calls.

Chapter 6 will be an analysis of the experiment and what is the result of the proposed approach with comparing different setups and browsers before going to a conclusion.

2 Related work and challenges

2.1 Description of the challenge



Topology	Client Upstream	Client Downstream
Fully Connected Mesh	$n-1$ outgoing connections per client: each client sends its captured media data to all of its peers	$n - 1$ incoming connections per client: each client receives media from all of its peers
Star Topology	1 outgoing connection per client: each client sends captured media data to the central server	$n - 1$ incoming connections: each client receives the $n - 1$ media streams of all other clients from the central server (assuming no server side media mixing is performed)

Figure 2.1: WebRTC topology comparison: Star architecture has network bandwidth advantage over standard WebRTC P2P mesh [17]

Traditionally, audio/video real-time software communication within web browsers was only possible via plugins or third-party software. WebRTC is a collection of communication protocols and Application programming interfaces (APIs) that support P2P real-time communication among web browsers. With the P2P capabilities introduced by WebRTC, browsers now break away from the classic client-server model. The advantage of this shift is that the APIs defined by WebRTC are the same regardless of the underlying browser, operating system and are available on many platforms, especially mobile devices. In the context of video conferencing however, the P2P approach has a potential drawback as depicted in Figure 2.1. For n clients to interact with each other, $n(n-1) = n^2 - n \approx n^2$ transmission channels are needed (fully connected mesh). In order to improve this behavior, a central server can act in the same way as a peer. [17]

This way implementation of star network topology is possible. All clients connect to the central server which distributes the received media streams. This approach reduces client upstream and enabling large conferences especially in face of asymmetric networks with considerably lower upstream than downstream bandwidth. A central server acting as a WebRTC remote peer to forward captured media data is often called Selective Forwarding Unit (SFU). Usually it does not mix the received media data into a composite stream but only relays these streams to a set of participants. [17]

In other words as a high quality video can be made with today's advanced mobile/tablet devices anytime, anywhere and the video contents provider such as YouTube, Facebook is rising, the media streaming technologies are changing from single source to more diverse and scalable multi-sources and crowd sources. Commonly, the existing commercials take the two methods to build the media streaming service platform. One is P2P streaming which is cheaper and better scalable to build the platform. However, in this case, it is difficult to ensure quality of service (QoS) because a user device have to deal with media streams of all members it communicates. The other is server-client streaming with a dedicated server. It solves the problems of P2P streaming and provides high availability and short startup latency. However, building the dedicated server costs a lot and a substantial waste of resources is occurred by overprovisioning (low utilization) because the dedicated server is built based on a peak of fluctuated user's service demand. Furthermore, because user's service demand is geographical distributed, the dedicated server fixed in one place cannot guarantee a user's service accessibility sufficiently. [18]

In order to resolve the problems of above media steaming, many media streaming service platforms exploit a cloud service that enables a scalable and distributed resource provisioning in a cost-efficient way for media streaming service. Actually Neflix, which is the major internet video provider, leverages cloud service to support Video-on-Demand (VoD) services. There are also many researches on a live streaming service platform such as broadcasting and video conferencing. Those researches focusing mainly on the network resource utilization and accessibility. However, as a web based live streaming service is appeared by WebRTC and a WebRTC based media streaming server, called as Multi-point Control Unit (MCU) which is delegated to deliver the media stream of a publisher to multiple subscribers with transcoding and audio/video mixing., is developed, the other factor such as CPU and Memory of the server such as MCU become a important issue in order to avoid QoS degradation. This is because WebRTC standard uses Secure Real-time Protcol (SRTP) for the packet transmission implying that the MCU has to encode and decode each packet in order to make the retransmission from the publisher to each subscriber. In this case, focusing only on the network resource makes service quality and resource utilization management inefficient. [18]

2.2 Related work

2.2.1 Time Analysis

Based on Akhmad Alimudin et al.[19] work by analyzing video conferencing features in applications that have been developed and test with several users using the internet from several providers. Testing was done using the WebRTC analyzer feature that already exists in the chrome browser by accessing "chrome: // WebRTC-internals".

Number of Users	Average Delay (ms)		
	60s	90s	120s
2	100.9	110.2	145.4
5	194.7	220.9	233.6
7	237.4	237.9	282.9

Table 2.1: Average delay per user [19]

From Table 2.1 shows there is an increase in delay when the communication time continues to increase. This means that there is an additional delay due to data queues on the server. And it can be concluded that the more users are connected, the system will increase the delay value. [19]

From Table 2.2 shows that the more users, the greater number of packet loss, and packet loss will decrease with increasing bandwidth. Whereas for testing without bandwidth limiter, packet loss is obtained with a value of 0% for all tests. [19]

Bandwidth Limit	Packet loss %		
	2 Users	5 Users	7 Users
1 Mbps	3.2	7.9	9.3
2 Mbps	2.8	7.4	9.1
3 Mbps	2.3	7.1	9.2

Table 2.2: Connections packet loss [19]

2.2.2 Signaling Mechanism

According to N. M. Edan et al.[20], Users in WebRTC need a signaling mechanism to set up a session, coordinate a communication and connect with each other. A novel scalable WebRTC signalling mechanism (WebNSM) has been created and implemented, which can offer bi-directional video conferencing for unlimited users, as well as using mesh topology over different networks such as Local Area Network (LAN) and Wide Area Network (WAN) networks. WebNSM guarantees a different performance for providing a method to manage the routeing by WebRTc characteristics. In addition, a deep evaluation of the physical implementation was done over CPU performance, memory usage, WebNSM performance, Quality of Experience (QoE). Nevertheless, using mesh has impacted the quality of audio and video due to the bandwidth and CPU consumption in spite of the fact that WebNSM has not been affected. Therefore it takes an average of 112ms as a mean time of delay from the time an offer is sent until returning a response, even when the network is congested. This application has calculated the number of links and Real-time Transport Protocol (RTP) to comprehend the number of connections in the mesh. Additionally, this signalling mechanism can support unlimited number of peers while having high core CPUs, that it can be supplied in various applications, such as conferencing among users, e- Learning among teachers and students, telemedicine among patients, doctors or technicians.

2.2.3 WebRTC Limitations

Currently WebRTC suffers from a number of limitations which are outlined in the following F. Rhinow et al.[21] already discussed several limitations

- There is currently a interoperability issue between browser. This led to implementation difficulties among browsers. The library PeerJS for example currently supports the Google Chrome browser only, because WebRTC is implemented in Mozilla Firefox differently.
- The browser implementations are currently in alpha or beta status and as a result have a number of bugs and may terminate unexpectedly.
- The WebRTC API does not yet offer functions for connection management and establishment. Instead, a second communication channel is necessary to establish a connection for example XmlHttpRequests and WebSockets[1] to overcome this limitation.

2.2.4 Distribution Algorithms and Network Topology

F. Rhinow et al.[21] mentioned tree-based P2P[2] infrastructures were proposed as a solution. These overlay networks relay the data among nodes in a P2P fashion, the data being spread from the root to its leafs. It has been shown that the performance of tree-based architectures can reach theoretically optimal metrics. However, they also have features that make them unsuitable for use in P2P video streaming solutions:

- Tree-based protocols are highly vulnerable to churn. Given that one of the upper nodes leaves the network, the tree has to be restructured. Due to the highly dynamic nature of the Internet such a system is an inappropriate solution.
- Given that the upload capacity of nodes is much lower than the capacity of the server, there exists an unfair distribution of the available bandwidth for each node.

[1]The WebSocket is a protocol designed to enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code. [22]

[2]between any two nodes there is always a communication path allowing those nodes to route messages from one to the other. A well-known class of overlays is formed by P2P (P2P) networks. [23]

3 Theoretical Background

3.1 Distributed systems

A distributed system is a collection of autonomous computing elements that appear to its users as a single coherent system. This definition refers to two characteristic features of distributed systems. [23]

The first characteristic is that a distributed system is a collection of computing elements, each being able to behave independently of the others. However, it must be noted that if they ignore each other, there is no use in putting them together. Modern distributed systems can consist of all kinds of computing elements, ranging from high-performance computers to small plug computers or event smaller devices. The computing elements are programmed to achieve common goals, which are realized only by exchanging messages with each other. A consequence of dealing with independent computing elements is that each one will have its own notation of time. In other words, there is no single global notation of time within a distributed system. This leads to challenges in synchronization and coordination within the system. [23]

The second characteristic is that a distributed system should appear as a single coherent system. End users must believe that they are dealing with a single system and they should not notice that processes and data are dispersed across a computer network. A single coherent system, which is made of multiple computing elements, has to operate in the same way, no matter how the interaction between the user and the system takes place. However, in reality this is an extremely complex task to achieve. Since distributed systems consist of multiple autonomous computing elements, at any point in time, any of them can fail. This would leave the application running with only partial functionality, which is common to complex systems. [23]

Some of the main goals and challenges worth considering when designing distributed systems is to support resource sharing, to make the distribution transparent, to achieve high openness and to achieve scalability. [23]

3.2 Design Goals

Just because it is possible to build distributed systems does not necessarily mean that it is a good idea. A distributed system should make resources easily accessible; it should hide the fact that resources are distributed across a network; it should be open; and it should be scalable. [23]

An important goal of a distributed system is to hide the fact that its processes and resources are physically distributed across multiple computers possibly separated by large distances. In other words, it tries to make the distribution of processes and resources transparent, that is, invisible, to end users and applications. [23]

3.2.1 Requirements Transposition

The concept of transparency can be applied to several aspects of a distributed system, of which the most important ones are listed in table 3.1 The term object to mean either a process or a resource. [23]

Transparency	Description
Access	Hide differences in data representation and how an object is accessed
Location	Hide where an object is located
Relocation	Hide that an object may be moved to another location while in use
Migration	Hide that an object may move to another location
Replication	Hide that an object is replicated
Concurrency	Hide that an object may be shared by several independent users
Failure	Hide the failure and recovery of an object

Table 3.1: Different forms of transparency in a distributed system [23]

The conclusion is that aiming for distribution transparency may be a nice goal when designing and implementing distributed systems, but that it should be considered together with other issues such as performance and comprehensibility. The price for achieving full transparency may be surprisingly high. [23]

3.2.2 Being open

Important goal of distributed systems is openness. An open dis- tributed system is essentially a system that offers components that can easily be used by, or integrated into other systems. At the same time, an open distributed system itself will often consist of components that originate from elsewhere. [23]

3.2.3 Being scalable

Scalability has different dimensions like Size, Geographic and Administrative. When a system needs to scale, very different types of problems need to be solved. First by considering scaling with respect to size. If more users or resources need to be supported, Often confronted with the limitations of centralized services, although often for very different reasons. For example, many services are centralized in the sense that they are implemented by means of a single server running on a specific machine in the distributed system. In a more modern setting, may have a group of collaborating servers co-located on a cluster of tightly coupled machines physically placed at the same location. [23]

3.3 Architecture

The organization of distributed systems is mostly about the software components[1] that constitute the system. These software architectures tells how the various software components are to be organized and how they should interact. Several styles have by now been identified, of which the most important ones for distributed systems are:

[1]A component is a modular unit with well-defined required and provided interfaces that is replaceable within its environment. [23]

- Layered architectures
- Object-based architectures
- Resource-centered architectures
- Event-based architectures

Research on software architectures has matured considerably and it is now commonly accepted that designing or adopting an architecture is crucial for the successful development of large software systems. [23]

3.4 Fault Tolerance

Fault tolerance has been subject to much research in computer science. What it actually means for a distributed system to tolerate faults. Being fault tolerant is strongly related to what are called dependable systems. Dependability is a term that covers a number of useful requirements for distributed systems as : [23]

- Availability
- Reliability
- Safety
- Maintainability

The focus in this thesis will be in **Reliability**.

3.4.1 Availability

Availability is defined as the property that a system is ready to be used immediately. In general, it refers to the probability that the system is operating correctly at any given moment and is available to perform its functions on behalf of its users. In other words, a highly available system is one that will most likely be working at a given instant in time. [23]

3.4.2 Reliability

Reliability refers to the property that a system can run continuously without failure. In contrast to availability, reliability is defined in terms of a time interval instead of an instant in time. A highly reliable system is one that will most likely continue to work without interruption during a relatively long period of time. This is a subtle but important difference when compared to availability. If a system goes down on average for one, seemingly random millisecond every hour, it has an availability of more than 99.9999 percent, but is still unreliable. Similarly, a system that never crashes but is shut down for two specific weeks every August has high reliability but only 96 percent availability. The two are not the same. [23]

3.5 WebRTC

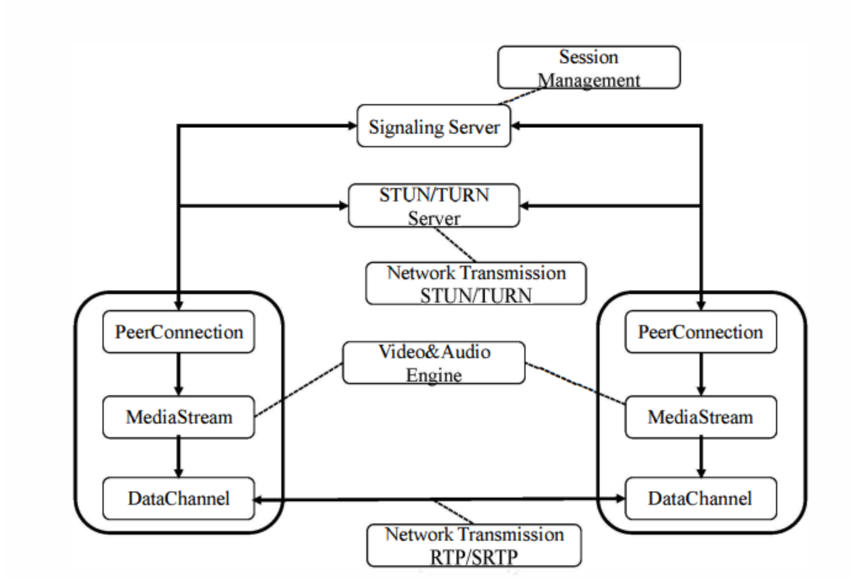


Figure 3.1: The principle of the WebRTC technology. [3]

WebRTC is a set of protocols and standards with a JavaScript API, which allows establishing P2P connection between different web browsers. WebRTC is supported in most web browsers. Establishing connections and calls is done without plug-ins and without installation of additional client applications, so it can be used as a cross-platform solution (directly from web browser). [5]

WebRTC API contains three parts: MediaStream, RTCPeerConnection and RTC-DataChannel as shown in Figure 3.1. MediaStream API provides access to the audio and video streams on the client device. Furthermore, this API enhances the quality of the audio and video streams using embedded functions for echo cancellation, noise reduction, jitter, synchronization and image enhancement. RTCPeerConnection API is used to manage P2P connection. This API provides functions for establishing, maintaining and finishing P2P connection. [5]

During the process of establishing P2P connections, peers use signaling server to discovery each other, and exchange IP addresses and information about the connection. There are several solutions for signaling channel, such as HTTP or WebSocket custom protocol, XMPP, SIP. WebRTC uses Session Description Protocol (SDP) for defining parameters of the P2P connection (session), such as content type, codec, network transport parameters and other metadata. WebRTC application uses JavaScript Session Establishment Protocol (acrshortjsep) to manage SDP parameters from RTCPeerConnection object. Every RTCPeerConnection object has Interactive Connectivity Establishment (ICE) agent, which is responsible for discovering local IP address and port (ICE candidate), checking the connection between peers and sending keepalive messages. ICE agent starts the process of discovering local peer candidate after getting SDP session. ICE agent check network to find a path for establishing P2P connection after the other host got candidate. Exchange of these parameters is done using the signaling server. Checking the network path begins with sending a Session Traversal Utilities for Network address translation (NAT) (STUN) request to the STUN server. STUN server is used to discover public IP address and port of the host. When the host receives STUN response, the network path between peers exists and the P2P connection can be established. If it is not possible to establish P2P connection, peers will use Traversal Using Relays around NAT (TURN) server as relay. After establishing the P2P connection, audio and video content is transferred using Secure Real-Time Transport Protocol (SRTP). Stream Control Transport Protocol (SCTP) is used for data transfer between peers. [5]

WebRTC has quickly become popular as a video conferencing platform, partly due to the fact that many browsers support it. WebRTC utilizes the Google Congestion-Control (GCC) algorithm to provide congestion control for real-time communications over User Datagram Protocol (UDP). The performance during a WebRTC call may be influenced by several factors, including the underlying WebRTC implementation, the device and network characteristics, and the network topology. [24] WebRTC is an open source software project which is a strength point would help the project to evolve with time by involving new contributors for example Source Forge [25]. Regarding T. Ichimura et al. [25] The growing process of project becomes strong according to the number of developers joining into the project. In the growing phase, T. Ichimura et al. found some characteristic patterns between the number of agents and the produced projects. Another example the Linux Kernel is one of the most successful open source projects to date [26].

3.6 Security Model

Security is very important element of the WebRTC. Complete communication audio, video and data is encrypted using combination of DTLS-SRTP protocols. Datagram Transport Layer Security (DTLS) protocol is an extension of the Transport Layer Security (TLS) protocol and provides data integrity and encryption of User Datagram Protocol (UDP) communication. SRTP protocol provides encryption, authentication and data integrity. WebRTC does not provide security mechanism for using signaling server. [5]

3.7 WebRTC Architecture

WebRTC architecture basically based on two layers, Web and WebRTC C++ API as shown in Figure 3.2 Web API is an API to be used by third party developers for developing web based video-chat application, WebRTC Native C++ API is an API layer that enables browser makers to easily implement the Web API proposal. [3]

In general, WebRTC consists of three parts: the API layer for Web developers, the API layer for browser developers and the custom service layer for browser developers. The top API layer for Web developers offers three basic APIs: GetUserMedia API, PeerConnection API and DataChannel API. [3]

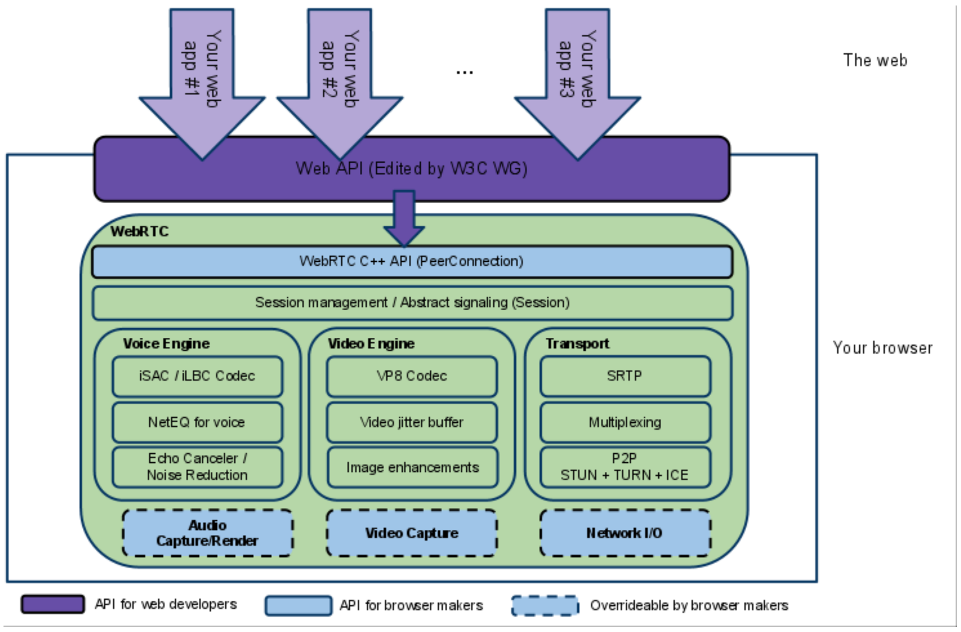


Figure 3.2: WebRTC architecture [27]

3.8 Problem Statement

3.8.1 Multi-point Control Unit

WebRTC includes mechanisms for Interactive Connectivity Establishment (ICE), allowing communication between users even if they are behind firewalls or one or more layers of NATs. However, WebRTC is only aimed at point-to-point communication between two browsers, so multi-point communication is out of the scope of the standard. Multi-point Control Units (MCUs) are then required to enable conferences with more than two endpoints. This brings the possibility to deploy various conferencing models with WebRTC depending on the signaling topology, ranging from star topologies to highly distributed multi-point topologies. [28]

WebRTC has contributed to successful applications in the fields of tele-health, online assistance in e-businesses, distance learning, all of them following the native peer-to-peer approach (P2P). WebRTC does not support IP multi cast, so multi-point scenarios should be supported by multi-uni cast deployments, using MCUs to forward media streams between participants. Examples of WebRTC MCUs, providing services such as media trans-coding and mixing. [28]

Integration of video conference systems on the cloud has been an important research topic in the last years a cloud video conference system for mobile devices, where the cloud resources improve quality and scalability, is described. In case of multi-point video conferences, advantages and challenges of cloud MCUs. [28]

3.8.2 Conference Models (Centralized model)

In this model, dialog signaling and media mixing are both carried out by only one user agent i.e. the Conference Focus (CF). This CF takes media from users who participate on the conference, mixes them, and sends out separately the appropriately mixed stream to each participant. Two different conference models can be created depending on the resource that takes the CF vocation: The CF could be one of the N participants or a dedicated conference server. This "End-System Mixing" is considered as the basic model that support small group of conferencing participant. It is specially characterized by the facility and the popularity of its implementation. Some VoIP provider like Skype implements this model. Otherwise, the second model (Media-server Mixing) is proposed to support larger conference group and run media mixing load on the Server side since this server offer best computational power and bandwidth than a single "basic" participant. Actual large scale solutions use this approach and some commercial products, like IVISIT or WEBEX are ready to support up to one hundred of simultaneous users within the same conference. The conference server is called MCU and designed to support WebRTC based browsers. In both models, the CF of such centralized approach plays the role of a media links with every participant and sends separately the mixed flow generated by mixer instances that are loaded locally. [29]

3.8.3 Thesis proposed approach

As mentioned before for scalability and reliability there is some existing solutions as Multi-point Control Unit (MCU) and Selective Forwarding Unit (SFU) but as Edan et al. has mentioned that using MCU is very expensive and MCU is costly and it can be rented from service providers just during a conference, although some video conferencing CODECs are able to support a specific number of multipoint (e.g. up to 4 users). Adding to that emphasizes that MCU consumes a significant amount of bandwidth [30].

Because of the MCU it is a server in the in the cloud it results a more reliability in case of the call initiator had a network failure or interruption, so on other words if the call initiator has left the call wouldn't close the call itself because the other participants still connected to the MCU server. This thesis will be focused on implementing a audio/video/screen sharing via WebRTC 1-to-1 and 1-to-many (conferencing) without involving neither MCU nor SFU, just browser to browser based on P2P with a signaling server based on websocket for signaling communication and star network topology.

3.8.4 Challenges of proposed approach

Single point of failure

The proposed approach has some challenge about reliability and scalability. For example the star network topology will have as a result a single point of failure. Tanenbaum et al. have mentioned The centralized approach also has shortcomings. The coordinator is a single point of failure, so if it crashes, the entire system may go down. In addition, in a large system, a single coordinator can become a performance bottleneck. Nevertheless, the benefits coming from its simplicity outweigh in many cases the potential drawbacks. [23]

Failure masking and replication

An approach to eliminate or minimize the single point of failure problem is to triplicate the voter. This approach has been used in the design of the fault tolerant multiprocessor system. The advantage of this method is solving the single point of failure problem by distributing the decision between systems, the period elapsed in the delay unit equals the time necessary to check it again and to replace the voter which may be neglected in most applications because it is too short. Of course the number of spare voters and disagreement detectors may be increased to as many as desired which is a trade off between cost and reliability. [31]

So by copying the list of the possible leaders to every node or user which is replication of the list is a solution for the single point of failure, in case of single point of failure happens the leader search algorithm will start a leader election algorithm for example the bully algorithm or a ring algorithm is meant to be an election algorithm. On this thesis bully algorithm will be chosen to apply as the leader search algorithm.

Bully Algorithm

In that thesis will apply the bully leader election algorithm to elect a leader for the call in case of the call initiator device crashes or had a network failure or loss.

A well-known solution for electing a coordinator is the bully algorithm devised by Garcia-Molina (1982) [23]. In the following, we consider N users P0, . . . , PN−1 and let id(Pk) = k. When any user notices that the coordinator is no longer responding to requests, it initiates an election. A user, Pk, holds an election as follows:

- Pk sends an ELECTION message to all users with higher identifiers: Pk+1, Pk+2, . . . , PN−1.
- If no one responds, Pk wins the election and becomes coordinator.
- If one of the higher-ups answers, it takes over and Pk's job is done.

At any moment, a user can get an ELECTION message from one of its lower-numbered colleagues. When such a message arrives, the receiver sends an OK message back to the sender to indicate that he is alive and will take over. The receiver then holds an election, unless it is already holding one. Eventually, all users give up but one, and that one is the new coordinator. It announces its victory by sending all users a message telling them that starting immediately it is the new coordinator. If a user that was previously down comes back up, it holds an election. If it happens to be the highest-numbered user currently running, it will win the election and take over the coordinator's job. Thus the biggest guy in town always wins, hence the name "bully algorithm". [23]

In Figure 3.3 an example of how the bully algorithm works. The group consists of eight processes, with identifiers numbered from 0 to 7. Previously process P7 was the coordinator, but it has just crashed. Process P4 is the first one to notice this, so it sends ELECTION messages to all the processes higher than it, namely P5, P6, and P7, as shown in Figure 3.3(a). Processes P5 and P6 both respond with OK, as shown in Figure 3.3(b). Upon getting the first of these responses, P4 knows that its job is over, knowing that either one of P5 or P6 will take over and become coordinator. Process P4 just sits back and waits to see who the winner will be (although at this point it can make a pretty good guess). In Figure 3.3(c) both P5 and P6 hold elections, each one sending messages only to those processes with identifiers higher than itself. In Figure 3.3(d), P6 tells P5 that it will take over. At this point P6 knows that P7 is dead and that it (P6) is the winner. If there is state information to be collected from disk or elsewhere to pick up where the old coordinator left off, P6 must now do what is needed. When it is ready to take over, it announces the takeover by sending a COORDINATOR message to all running processes. When P4 gets this message, it can now continue with the operation it was trying to do when it discovered that P7 was dead, but using P6 as the coordinator this time. In this way the failure of P7 is handled and the work can continue. If process P7 is ever restarted, it will send all the others a COORDINATOR message and bully them into submission. [23]

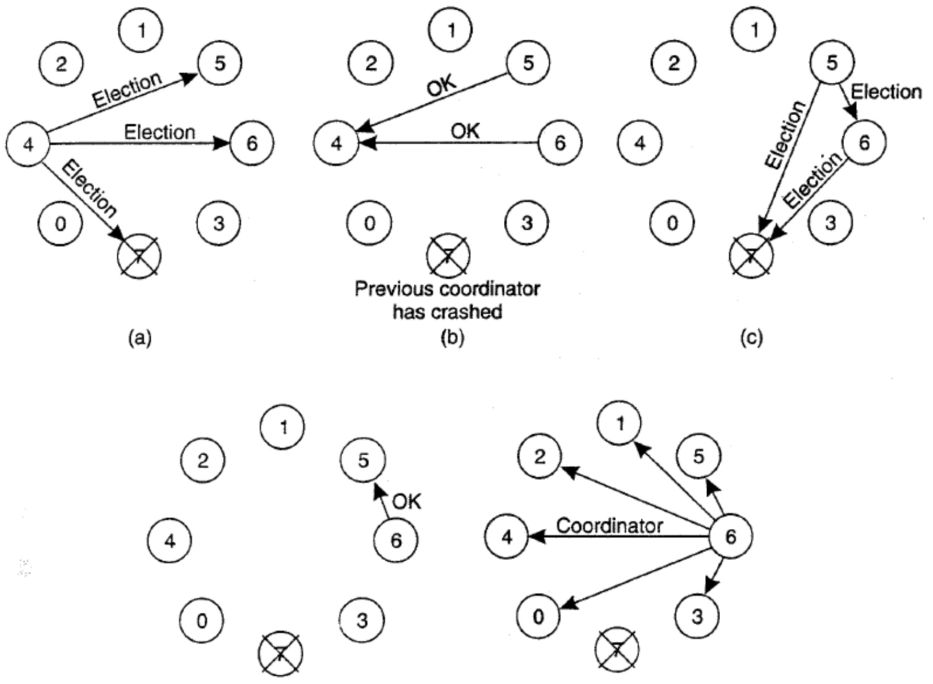


Figure 3.3: The bully election algorithm. (a) Process 4 holds an election. (b) Processes 5 and 6 respond, telling 4 to stop. (c) Now 5 and 6 each hold an election. (d) Process 6 tells 5 to stop. (e) Process 6 wins and tells everyone.

4 Requirement Analysis

Regarding to Tanenbaum et al.[23] the definition of ubiquitous computing system is the system is pervasive and continuously present. The latter means that a user will be continuously interacting with the system, often not even being aware that interaction is taking place. The core requirements for a ubiquitous computing system roughly as follows:

- (Distribution) Devices are networked, distributed, and accessible in a transparent manner
- (Interaction) Interaction between users and devices is highly unobtrusive
- (Context awareness) The system is aware of a user's context in order to optimize interaction
- (Autonomy) Devices operate autonomously without human intervention, and are thus highly self-managed
- (Intelligence) The system as a whole can handle a wide range of dynamic actions and interactions

Therefore in this thesis will show the five requirements for a ubiquitous computing system in 2 parts which is Functional requirements and Behavioural requirement as will be shown next.

4.1 Functional requirement

As shown in Figure 4.1 every device has to be connected to the signaling server through websockets and the advantage of that architecture is that the signaling server can detect whom is connected and whom is not. Behind the signaling server exists database which has the users record details and the channel which is a group of user existing in the same chat context. The other usage of the signaling server later on will be when the call initiator initiate a call, the offer is created and passed to all users in the same channel through it, then every user in the channel will send back the answer to the caller through the signaling server as well, either by accepting or declining.

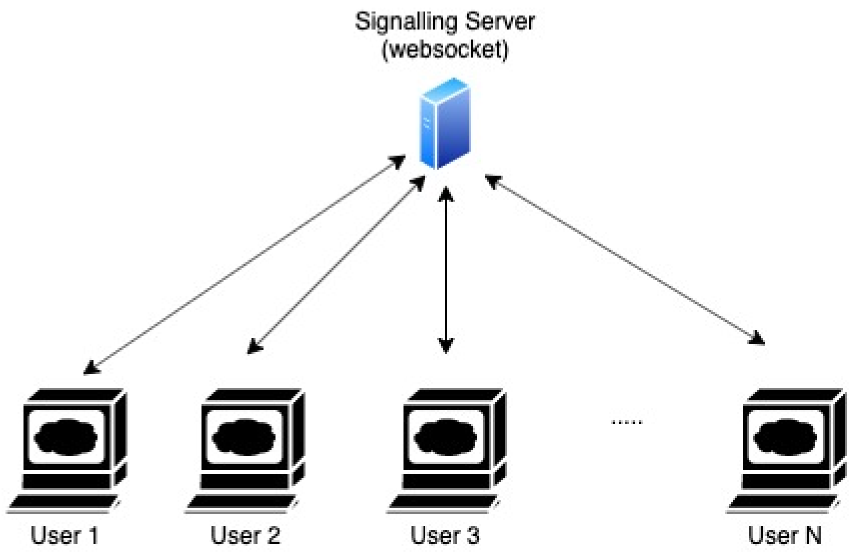


Figure 4.1: Devices connections to signaling server through websockets

The call at the beginning it initiates a peer connection with audio permission for every user in the channel, then later on every user will be able to upgrade his connection to open the camera or share his screen. By upgrading you create a new signal to the initiator to upgrade and accept the new added track of video, then the initiator distribute it to the rest of the connection in the call.

4.2 Behavioural requirement

In Figure 4.2 as shown the mixing of other users media is done through the initiator of the conference call device as a replacement to MCU and distributed to the rest of the connected devices in the call. Every device/client has a list of whom is connected to the call and possible leader list In case of the initiator network is

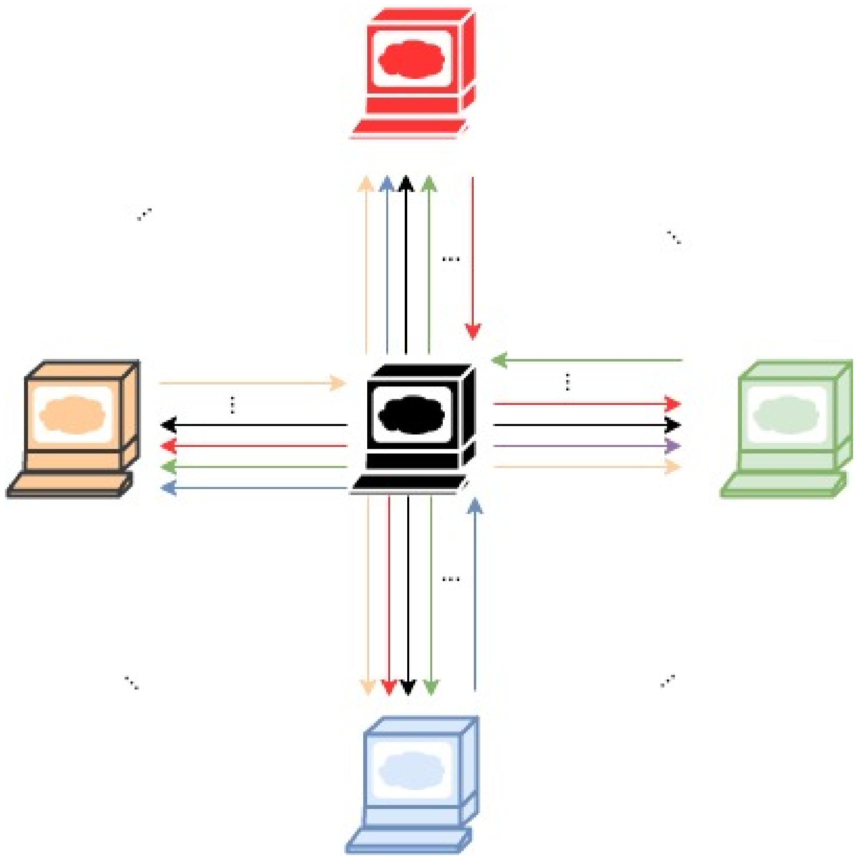


Figure 4.2: Pipeline participants media through Star network topology

crashed the system has to start the bully election algorithm to elect a new initiator for the call.

Every call has to be in the database in case the connection interrupted for sometime and the user returned before the call have ended will be automatically reconnected again to the call. The expectation of such a behaviour will define a certain quality of reliability.

5 Planning and Software design

This section presents the sequence of having connection establishment and handling the leader election in case of single point of failure.

5.1 WebRTC establishing a Peer Connection

In Figure 5.1 shown the sequential diagram of establishing WebRTC peer connection as described before in the Theoretical background chapter, the signaling server is mandatory to complete establishing the connection. After using STUN or the TURN server to detect the public IP of the device every peer send his session description protocol to the other peer then later on after every peer set it is local description and remote description they start to negotiate their ICE candidates via the signaling server until the both agree on the candidates and the connection is established. The signaling server in the structure in Figure 5.1 is a websocket connection between the signaling server and every online peer.

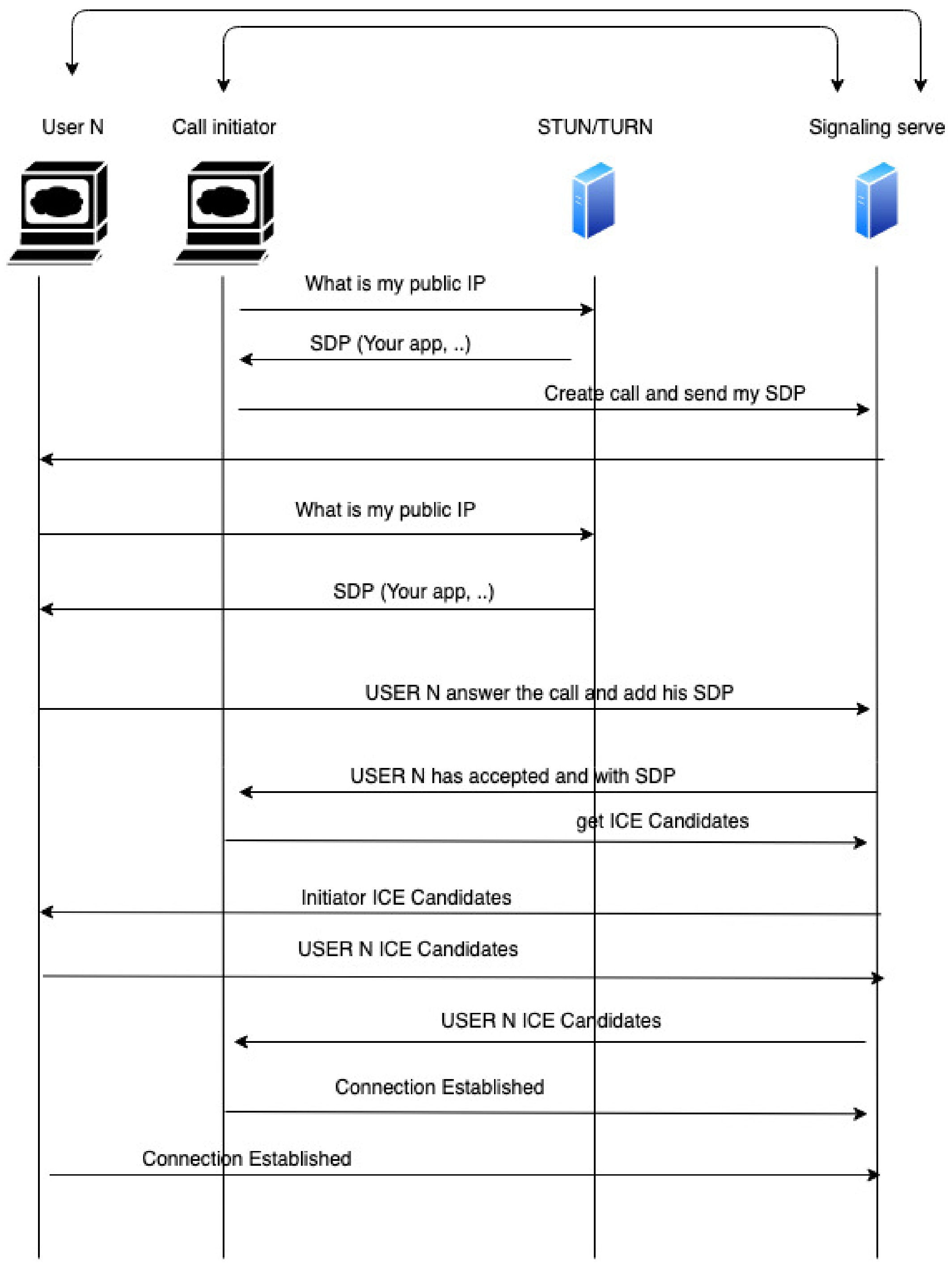


Figure 5.1: Sequence diagram representing establishing connection via WebRTC

5.2 WebRTC States

A possible scenario would be for connection 2 peer connections

- new PeerConnection(): Starting
- (Starting, remote candidates received): Checking
- (Checking, found usable connection): Connected
- (Checking, gave up): Failed
- (Connected, finished all checks): Completed
- (Completed, lost connectivity):: Disconnected
- (any state, ICE restart occurs): Starting
- close(): Closed

In Figure 5.2 is a state diagram shows how every state move to another in the process of establishing a peer connection.

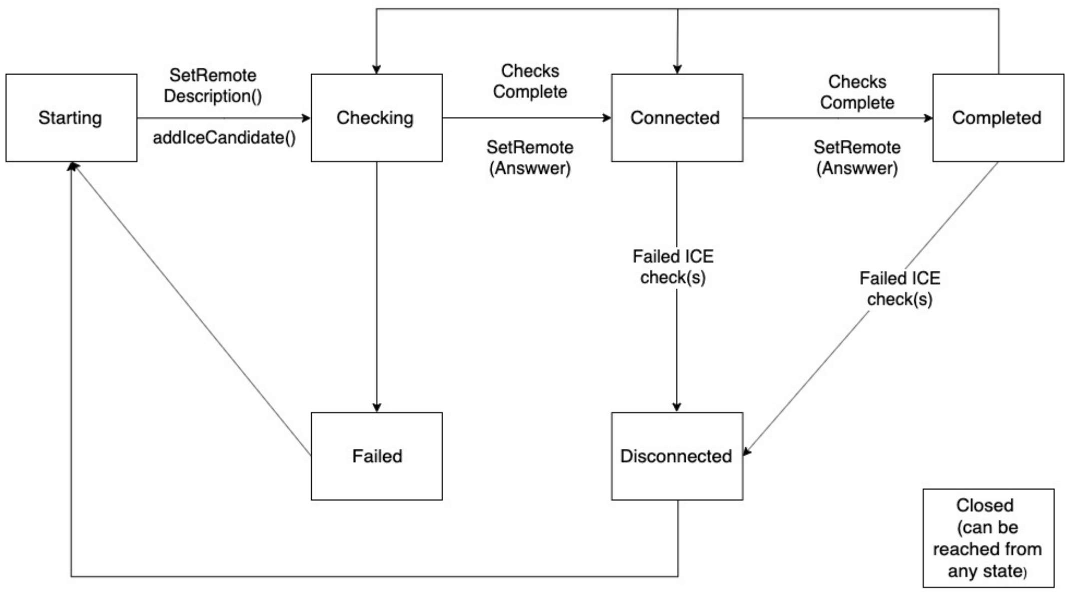


Figure 5.2: State diagram for ICE candidates states

Starting

The ICE Agent is gathering addresses and/or waiting for remote candidates to be supplied.

Checking

The ICE Agent has received remote candidates on at least one component, and is checking candidate pairs but has not yet found a connection. In addition to checking, it may also still be gathering. Occurs if ANY component has received a candidate and can start checking

Connected

The ICE Agent has found a usable connection for all components but is still checking other candidate pairs to see if there is a better connection. It may also still be gathering. Occurs if ALL components have established a working connection

Completed

The ICE Agent has finished gathering and checking and found a connection for all components. Occurs if ALL components have finalized the running of their ICE process

Failed

The ICE Agent is finished checking all candidate pairs and failed to find a connection for at least one component. Occurs if ANY component has given up trying to connect

Disconnected

Aliveness checks have failed for one or more components. This is more aggressive than "failed",and may trigger intermittently (and resolve itself without action) on a flaky network. Occurs if any component has failed aliveness checks

Closed

The ICE Agent has shut down and is no longer responding to STUN/TURN requests. Occurs only if PeerConnection.close() has been called.

5.3 Sending Data with Signaling Server

Before introducing the proposed signals architecture design will describe which data the signals shall hold, data interchange format and how it should looks like because what should be kept in mind that it is a Real-time application so time performance and efficiency it is not a feature it is a requirement.

5.3.1 Data transport

Since some Web applications require relatively high real-time data, the client side does not refresh the entire page in the case of asynchronous real-time data transmission. Ajax has a variety of data formats to achieve the client and server communication, Common data exchange formats are HTML, XML, JSON. [32]

XML

Extensible Markup Language (XML) when it emerged as a W3C standard more than a decade ago, XML was viewed as a revolutionary markup language for data transport and storage. It was designed as a software and hardware independent tool, which was convenient for the developer community at that time. XML became a widely accepted language and gained acceptance into all programming fields, but it mostly affected the way the Internet today works. [33]

```
<signal>
   <id>15</id>
   <name>Call</name>
   <description>When a user create a
call in the channel</description>
   <options type="user_name">
     <item>Max</item>
     <item>Stefan</item>
     <item>Alice</item>
   </options>
</signal>

"signal" : {
   "id" : 15,
   "name" : "Call",
   "description" : "When a user create a
call in the channel",
   "options" : [
       {
            "type" : "user_name",
            "items" : [
                    "Max",
                    "Stefan",
                    "Alice"
            ]
       }
   ]
}
```

Figure 5.3: Data transport: XML vs. JSON

JSON

Over the years, web sites have shifted from being static to very dynamic and interactive. Interactivity increased data transfer and developers looked for a way to modify a small section of a website without re-transmitting all of the data.

Asynchronous JavaScript and XML (Ajax) is a group of technologies that together account for asynchronous communication with the server without having to refresh the page. In the original specification Ajax was designed to work with the following technologies: Hypertext Markup Language (HTML) and Cascading style sheets (CSS)for presentation, Document Object Model (DOM) to dynamically display and interact with data, XML for data exchange, XMLHttpRequest object for asynchronous communication and JavaScript as the cohesive technology. For Ajax, the name itself suggests that it was built for XML, but there was the possibility of using other encoding. XML had its limitations and did not prove to be the ideal transportation tool so new data formats emerged, such as JavaScript Object Notation (JSON). Although JSON is based on the JavaScript language, it is language independent, as it uses a simple key-value pair notation. Today JSON is widely used as an alternative to XML in the field of web development. [33]

Other options and decision making

There is another option for Human readability data transport which is YAML, "YAML Ain't Markup Language" (YAML) is a data serialization language designed to be human-friendly and work well with modern programming languages for common everyday tasks. [34]

From Figure 5.3 first part shows XML example and second part shows the same example with JSON, JSON is more human readable which makes it a better choice for code quality and readability, WebRTC web API implementation is in Javascript and since JSON is prepared for JavaScript [32], therefore JSON the more logical choice for data transport. JSON can be transmitted with a simple String, Number or Boolean type variable, array or a complex object.

Based on work of P. Wang et al. [32] experiments shows that when analyzing JSON type data objects, the computer spend less time than XML data type in large number of parser objects case, using JSON data type as transmission type will bring faster user experience, and cost on development and maintenance also can be reduced. a test case demonstrates the time which the computer spends on parsing ordinary XML and JSON object has a result shown in Table 5.1.

	XML	JSON
Object Count	10000	10000
Total Time(ms)	25675	19484
Average Time(ms)	2.567	1.948

Table 5.1: The time consuming in large number of parsing objects case [32]

Regarding this results one can say that XML objects is slowest. So regarding readability and time JSON is better than XML and since JSON is prepared for

Javascript and can be transmitted with a simple String, Number, Boolean or Array and Complex objects as mentioned before. Therefore it is a logical and better choice to choose JSON as the data transmission format.

5.4 Proposed Signals

In that section will go through the proposed signals as solution for the requirements which is can be signaled via the signaling server. As shown in Figure 5.4 the basic signal data.

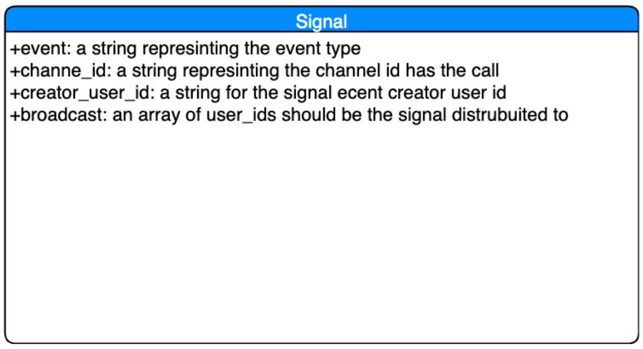

Signal
+event: a string represinting the event type
+channe_id: a string represinting the channel id has the call
+creator_user_id: a string for the signal ecent creator user id
+broadcast: an array of user_ids should be the signal distrubuited to

Figure 5.4: Basic Signal UML

5.4.1 Signal Anatomy

event

Regarding the system has different states of the call like Calling, Answering, Canceling, Upgrading, Connection lost, ICE Candidates exchange and more, a string event descriptive name which is describe the event type and purpose of the signal.

channel_id

Regarding the call will be initiated per channel which is the unique domain of the call and it is later on a search parameter to filter the calls. It is important parameter which is a string has the channel id and used to be string to not give it the numbers limitations based on the system bu using integers or double, So instead a string used which is a unique combination of 0:9 and A-F with a length of 15 char to have a 15^{15} possible unique combination.

creator_user_id

To distinguish between the call initiator and the other call candidates a signal creator user id it has to be represented and it is a string states the user id uses the same combination as the channel but it is a user based.

broadcast

To start distributes the signal to multiple users which the aim of the signaling server a broadcast which is an array of the users ids has to get that signal.

As will be shown next every specific signal has it is specific data attribute which is related to the signal functionality.

5.4.2 Ping Pong Signal

To have a continues connectivity between the server and the user client which will allow the server quickly detect if the user had a network interruption or went offline by closing the session the server sends a "ping" signal to every user connected via websocket to it to detect if the user is still alive and if the user did not response with a signal "pong", then that user considered as offline and the other users being notified in case if the user was the call creator then the leader algorithm start, otherwise just removing the user from the call in the database, in the call candidates clients and the bully possible leader list.

5.4.3 Creating Call

To notify the other users in the channel about an incoming call and to start the peer creation in the other participants client a signal needed which is created when someone tries to call another candidate or make a call through a channel it holds the Session description of that user which it has to be sent to other participants to notify them that they have a call and they can either accept it or decline it and if the other participant accept it the initiator session description will be added by PeerConnection.setRemoteDescription().

The following code snippet is the JSON of creating call signal as an example of how the signals looks like.

```
{
    "event": "SOCKET_CREATE_RTC_CALL",
    "creator_user_id": loggedin_user_id,
    "channel_id": channel_id,
    "desc": sdp,
    "join": false,
    "broadcast": broadcast_user_ids
}
```

5.4.4 Join Call

If a user missed a call and later on decided to join the call or in case of a silent leader algorithm running to decided the new call leader and start establishing the new connections without the other users indicate it (migration of the call) Join call signal is needed which is a signaling between a participant and the call initiator with automatic call accept which created by the participant to ask the call initiator if the original call initiator or the new call leader to create a new peer connection and it will be automatically accepted from the participant side to join the conference call.

The advantage of that approach of joining calls that when a user had a missed call and then later on he decided to join the call will be able to join and other hand it can be used as reliability if the user has a connection interruptions or network problem then later on when the connections comes back automatically will be reconnected and joined to the call with the same signal same approach. The JSON for joining call is exactly as creating call but with different value for the attribute "join" : true

5.4.5 Upgrade Call

It is not a realistic case study that the users just create a call and start automatically transferring the audio and video at the same time, therefore by creating the call only the audio will start to be transferred and later on if the user decided to share his screen or open the camera the new track will be sent, therefore a signal needed to upgrade the call It will be created as well if a new user has joined the call and added a media track because as have seen before in the Behavioral requirement the media tracks pipeline is happened via the call initiator as well because it is not supported in MediaTrack API to know whom have sent that media track then it is instead sent with the upgrade call to detect the track source user.

5.4.6 Has Received User

To confirm that the mixing and distributing of the users tracks on the call initiator side happened to all users a signal needed from the other cal participants side to notify the initiator that a user track received.

5.4.7 Answer Call

To notify the call creator that the other candidate in the call has answered an answer call signal is needed and created from the participants side to add the session description of the participants and notify the call initiator that the participant has accepted the call and add it is session description to be added in the call initiator side by PeerConnection.setRemoteDescription().

5.4.8 Offer ICE Candidates

To exchange ICE candidates between 2 peers after creating a call which is creating a peer, after creating a peer there is need to create an offer which will be the ICE candidates will be sent to the other candidate, so the call initiator client peer starts to get ICE candidates, then send them to the participant via Offer ICE signal.

5.4.9 Answer ICE Candidates

To exchange ICE candidates between 2 peers after the participant accepts the call creating an answer from the peer is needed, so the participant client peer start to get ICE candidates and then send them to the call initiator via Answer ICE signal.

5.4.10 Decline Call

To decline an incoming call a decline signal is needed it is created in case of the participant decided to cancel a call or the initiator want to end a call to close the peer from the side of participant and to close all the peer connection in the side of the call initiator in case of it is created from his side to end the conference call.

5.4.11 Disable Audio

In case of the user decided to mute his microphone the track will be stopped to send packets to the other peer and to notifies the other side of the call that the user is still here but he decided to mute his microphone and show a mute icon on the user avatar a disable audio signal is needed. It is created in case that a user wants to mute his audio stream and created to the whole call participants to stop the audio track or to turn it back on to remove the mute icon.

5.4.12 Disable Video

In case of the user decided to close his camera or stop sharing his screen the track will be stopped to send packets to the other peer and to notifies the other side of the call that the user is still here but he decided to close video tracks and show the user avatar a disable video signal is needed and later on if the user decided to open his camera or share his screen (a video track) sent the avatar automatically removed and the video displayed.

5.4.13 Start Leader Search

In case of the call initiator got disconnected or had a network failure the other candidates in the call have to be notified so start the leader election algorithm, therefore it is created from the signaling server to all joined users in the call which will let the participants to ask for a leader using the bully leader algorithm to decide the leader of the call and re-establishing the call by sending a signal Be Leader.

5.4.14 Be a Leader

In case of the bully algorithm is started every user client has a list of the possible leader users ids and there is a need to ask this users if they are still alive and can be a leader and regarding the Ping Pong signal that list is always updated, therefore the users ids in the list is only the online user which can be a leader, all the users starts to ask the first element and if there is not an answer in 500ms moves to the next in the list and 500ms is a reasonable amount of time to create a signal via the signaling server which shows that the asked user bandwidth is has good network bandwidth.

5.4.15 Set Leader

In case the user client received the Be a leader signal send a signal which will stop the other candidates to search for a new leader, therefore set a leader signal is created as a response of be a leader signal to notify others that the leader is selected and they have to stop searching, then start re-establishing the call again.

5.4.16 Timeout

To cancel unanswered call or in case the user has the client opened but not physically in front of the computer or in case creating a call in a channel nobody answered it a termination signal is needed, therefore on the signaling server there is a cron job of working thread awake every 30s to check the database calls table by checking every call state if it is in "init" state and by comparing the start time of the call against the current time and how many people joined the call, if they are less than 2, then it creates time out signal to the call initiator and if they are more than or equal to 2 it checks how many user ids in that channel which can be in the call if the others did not interacted either by accepting or canceling then it creates a Timeout signal to those users who did not interact and add them in missed call field.

5.5 Database

Regarding that the call is going to be in a web browser the user can be by mistake refresh the page then will be removed from the call and to make the user automatically joining after the page is reloaded there is a need for a database to keep every call with the participants users ids, the other requirement that if a user tries to call another user for a direct call and if that user is busy in ongoing different call the request should be returned by busy response which it will be decided through the server checking the database instead of handle it in the other user client side and overload the work on the client. Therefore in that section will go through the proposed database tables which is play a role key of reliability. The proposed approach has to have a database to handle the calls states and participants of the call. The benefits of having the a database record for every call is analytical purpose and reliability which is the thesis topic. In Figure 5.5 is shown the database diagram of calls, channels and users tables.

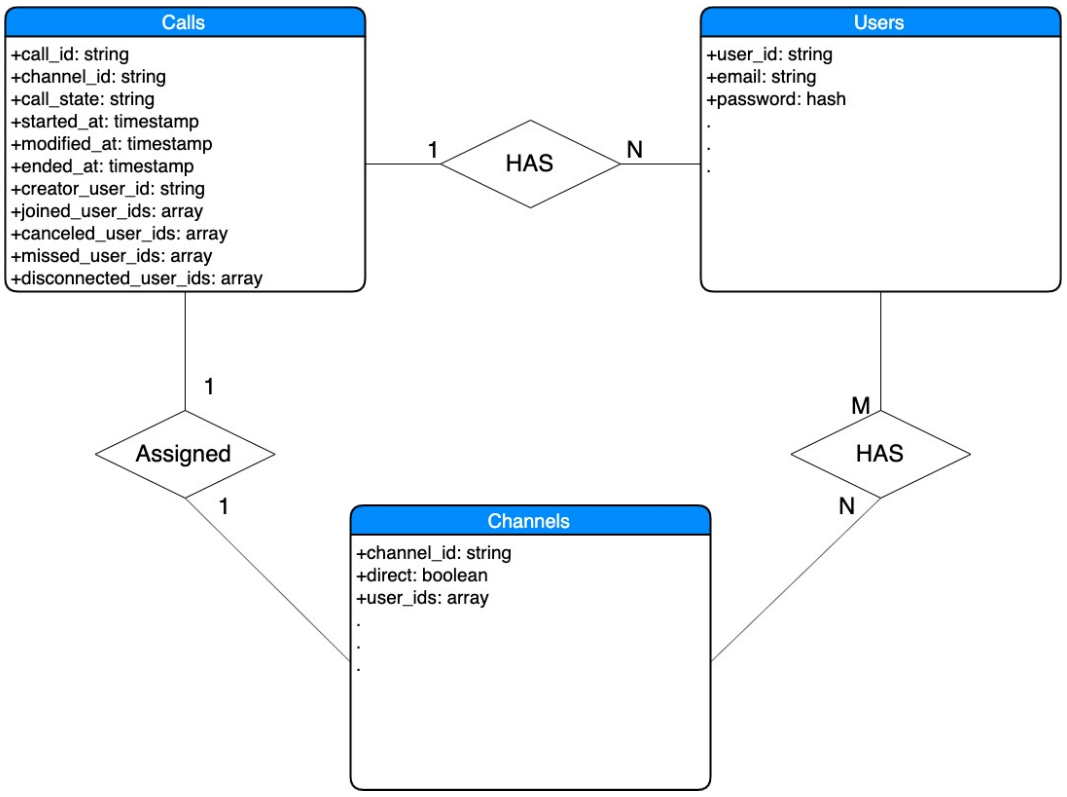


Figure 5.5: Database Entity Relationship Diagram for proposed approach

5.5.1 Users table

To initiate a call it has to have a participants and those participants has to be marked with a unique id, for getting an information about those participants like username, email, profile picture, when join the platform and much more. Therefore users table is necessary for users identity data but what is necessary for a call is the "user_id" which will be the foreign key used in other tables like Channels and Calls. Every channel or call has to be consisted of at least 2 users or more.

5.5.2 Channels table

As mentioned before the channel is the domain of a call and to have the channel details like user ids which is in that channel, channel name, creator user id, when it is created and messages in the channel and more, therefore channel table is necessary for holding this data. What is relevant to the calls table in the channel table is the "channel_id" and the "user_ids" in that channel because when a call is started in a channel it has to be under the domain id of that channel and between the users existed in that channel.

5.5.3 Calls table

That is the main table for our call and what will be focused in this thesis. Regarding that there is a requirement to keep track of whom is the call and in which channel is the call going, the calls table has the "channel_id" the domain of that call, "creator_user_id" which is the user_id of the call initiation and will be added to "joined_user_ids". When another user will accept the call will be added to "joined_user_ids", if the user declined then will be added to "canceled_user_ids", if the user was not available and did not react to the call will be in "missed_user_ids". After the call has been started in the mean time if a user has lost the connection will be added to "disconnected_user_ids", that field is important because later on if that user logged in again for example reloading the page and the call still exists the user will still be able to return to the call automatically and just create a silent signal event to the call initiator to start a join call it is automatically checked when the user load the page if was in a call.

5.6 Caller States

As shown in Figure 5.6 the flow chart diagram of establishing a call for the proposed approach the scenario. In that section will go through every state by describing the requirement for it and it conditions.

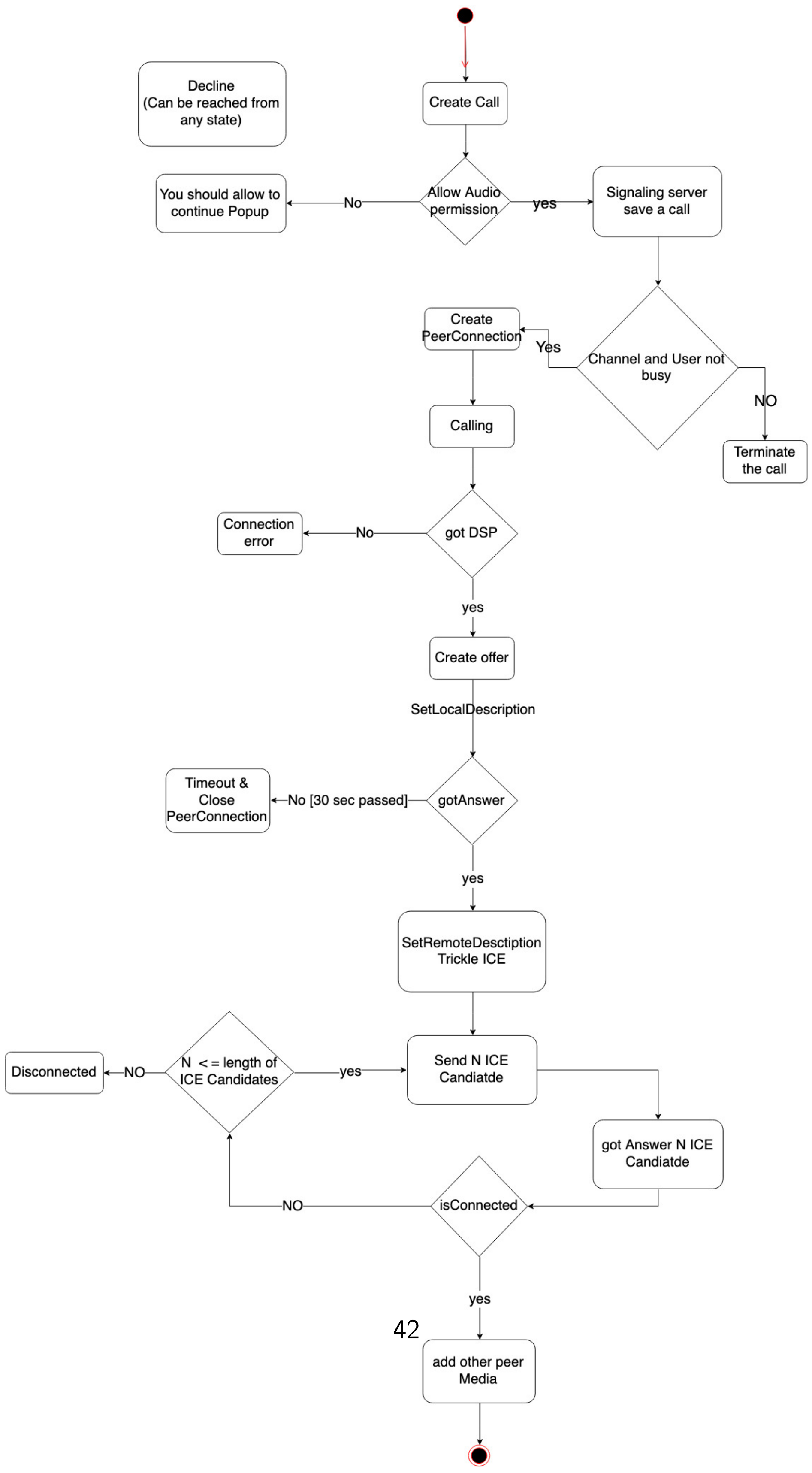


Figure 5.6: Flow chart diagram for establishing a call from the caller side

5.6.1 Create Call

To create a call the user has to allow the microphone permission because there is no sense to create a call without having the audio and the browser for the security matter has to ask and notify the user that the microphone will be allowed. Therefore this is the initial state of creating a call then the user will be asked to allow the microphone permission to start the call if the user refuses then the call be hold till the user allow the microphone permission and for sure should have a physical microphone either internal or an external, otherwise the call will not be started and will show the user that there is no a microphone to request it.

5.6.2 Signaling server to save the call in the database

As mentioned before every call has to have a log and existence in the database regarding the reliability, the analysis and detect which users are available a request to the signaling server has to be done to save the call request in the database with the creator user id and the channel id. The signaling server check if the requested call is for just a direct channel or multiple user channel and check the candidates of the channel if they are busy in another call. If the channel is direct and the other user is busy then return busy response status, if the channel is multiple user channel then it returns an array for available users and another array of busy users then the call initiator will be able to create a peer connection for every available user.

5.6.3 Create PeerConnection

To start establishing a call a peer connection has to be initiated to collect the permissions types, initiates the creation of an SDP offer for the purpose of starting a new WebRTC connection to a remote peer. The SDP offer includes information about any MediaStreamTracks already attached to the WebRTC session, codec, and options supported by the browser, and any candidates already gathered by the ICE agent. At that state the user have allowed the audio permission to access the microphone and got the channel available users, then new peer connection will be created for every available user in the channel.

5.6.4 Calling

At that state every peer connection getting Session description and a public IP from STUN or TURN server and add the tracks, the overlay of creating a call appears on the ringing view and the WebRTC session started.

5.6.5 Create Offer

The call initiator start to creating offer and send the local Session description to every peer via the signaling server and waiting for answer. every offer and answer is individual passed on the peer and as mentioned the SDP offer includes information about any MediaStreamTracks already attached to the WebRTC session, codec, and options supported by the browser.

5.6.6 SetRemoteDescription

At that state the user gets an answer for a specific peer then set the remote description for that peer based on the offer have been sent for that peer. This description specifies the properties of the remote end of the connection including the media format.

5.6.7 Send N ICE Candidate

The initiator starts to exchanging the ICE Candidates with the other peer till the connection either being accepted and the call state continue to connected or network problem happens and the connection become disconnected. When a RTCPeerConnection receives a new ICE candidate from the remote peer over its signaling channel, it delivers the newly-received candidate to the browser's ICE agent by calling RTCPeerConnection.addIceCandidate(). This adds this new remote candidate to the RTCPeerConnection's remote description, which describes the state of the remote end of the connection.

5.6.8 Add other peer Media

Pipeline happens through the call initiator device, therefore if the connection went successful and the peer connection became connected, then the user get the other peer media track and added for his side to represent the other connected peer. every added track kept in an array till all other users in the call have interacted either by accepting or refusing for later use to distribute the track to the new accepted users.

5.6.9 Decline Call

There has to be a way to end an ongoing call, therefore decline call can be called at any time to close the peer connections and cancel the call, that state as well notify the signaling server that the call has been ended to update the database and later on notify other candidates in the call that the user has left the call.

In case of there is no one answer for 1 of the initiated peers in 30 sec. the signaling server send a timeout signal, then the initiator has to close that peer for that targeted user.

5.7 Callee States

As shown in Figure 5.7 the flow chart diagram of establishing a call from the calle side.

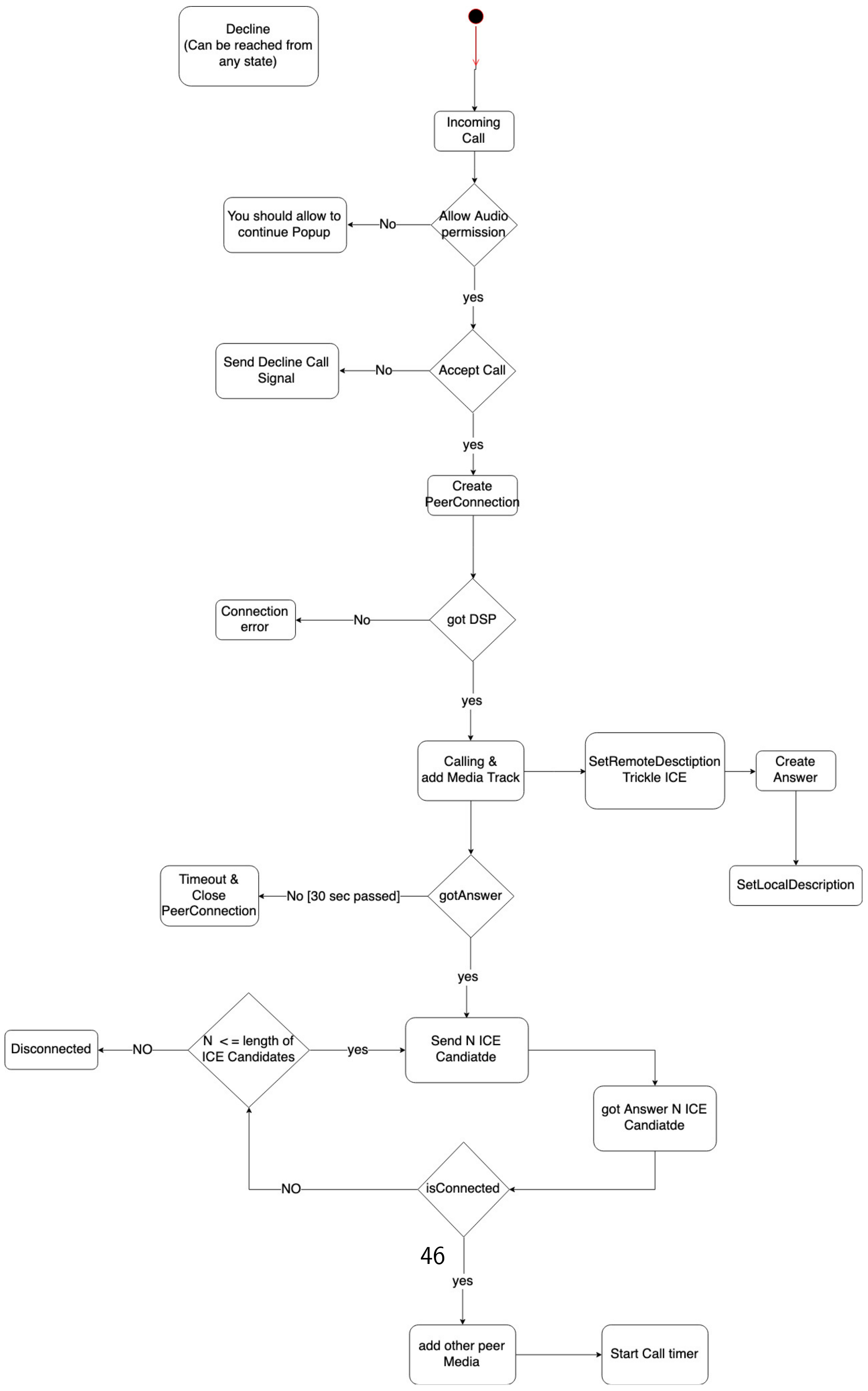


Figure 5.7: Flow chart diagram for establishing a call from the callee side

5.7.1 Incoming call

To notify the user that there is an incoming call and allow the user to accept it or refuse it the incoming call state is necessary to make the user get a notification with pop up that the user getting a call from the call initiator with the initiator details like username and image, then the user asked for allowing the audio to continue if will accept it and will be able as well to decline it.

5.7.2 Accept call

To notify the call initiator that the user has accepted the call and start joining and to update the database calls table an accept call state is represented to allow the user accepting the call and send the data to the signaling server that the user have accepted to answer the call.

5.7.3 Create peer connection

the requirement on the callee side for creating a peer connection it is exactly the same requirement on the caller side except that instead of creating offer it is creating an answer to the offer offered by the call initiator with the same steps.

5.8 Joining a call

As shown in Figure 5.8 the flow chart diagram of establishing joining a call which is mandatory for different purposes like join a caller after missed it and for establishing the leader algorithm in silent mode in case of the call initiator is gone.

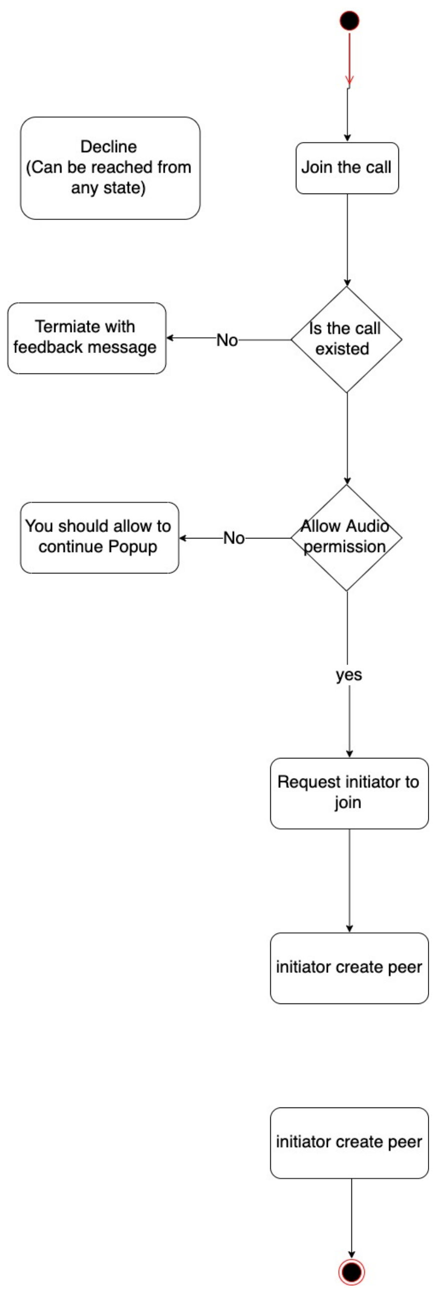

Decline
(Can be reached from any state)
Join the call
Termiate with feedback message
No
Is the call existed
You should allow to continue Popup
No
Allow Audio permission
yes
Request initiator to join
initiator create peer
initiator create peer

Figure 5.8: Flow chart diagram for establishing joining a call

5.8.1 Join call

When the user have missed a call or declined the call and later on decided to join the call a request to the signaling server to check if the call still existed or not and if the call is not existed it gives some feedback to the user, if the call still existed it asks the user to allow the audio permission to continue the call.

5.8.2 Request initiator to join a call

The users has to be allowed to join the call in a silent mode in case of that they were in the channel which has the call. The silent mode means that during the creation of the peers and negotiation time till the user being connected then appears in the call as joined and notify the others that the user has joined. In that state the signaling server create a signal message to the call initiator that the user wanted to join with his user id, then the call initiator create a new peer connection and start the same process for establishing connection with that user wanted to join, then after the peer connection of the user wanted to joined is succeed the initiator starts to distribute that new media track to the other users in the call.

5.9 Leaving a Call

When the user decide leaving the call shall be leaving state which is close the user whom decided to leave peer and notify other participants in the call that a user have left via the signaling server to and the user media tracks removed from the other user client side and updating the database on the server.

5.10 Inviting a user to a call

If the call initiator has to decided to invite a new user to the channel call or trying to recall a user or inviting a user which is external of the channel just for the purpose of the call, there has to be a way to invite the users but actually the process for inviting a new user it is simply creating a new peer and starts the process of establishing a new peer connection as creating a call states without interrupt the ongoing call and extend the call joined user by q then after success distribute the new user media tracks to the call joined users and notify all participants that a new user has been added.

5.11 Leader Election States

As mentioned before in the problem statement section the solution for single point of failure is replication and for doing that using the bully algorithm for leader election. In Figure 5.9 shown the states in case of the call initiator is disconnected till the re-establishing of the connection.

5.11.1 Call initiator disconnected

The call initiator can be detected easily that was disconnected via the signaling server which will notify every user existed in the same channel with that user id and state that the initiator is gone and every user browsers can start the leader algorithm in case of the call not ended. Regarding the replication there is a list of possible leader user ids exists in every user client side to allow the user start to sending the "ASK_TO_BE_LEADER" signal directly when receives that the initiator is gone by Start search leader.

5.11.2 Ask for a leader

To implement the bully algorithm users has to ask the other higher rank users to be a leader and wait for a response, in case of the user received an OK message which is "SET_LEADR" in this case will stop to continue asking and waiting for the connection to be re-established and if not received it in 500ms will continue with the list.

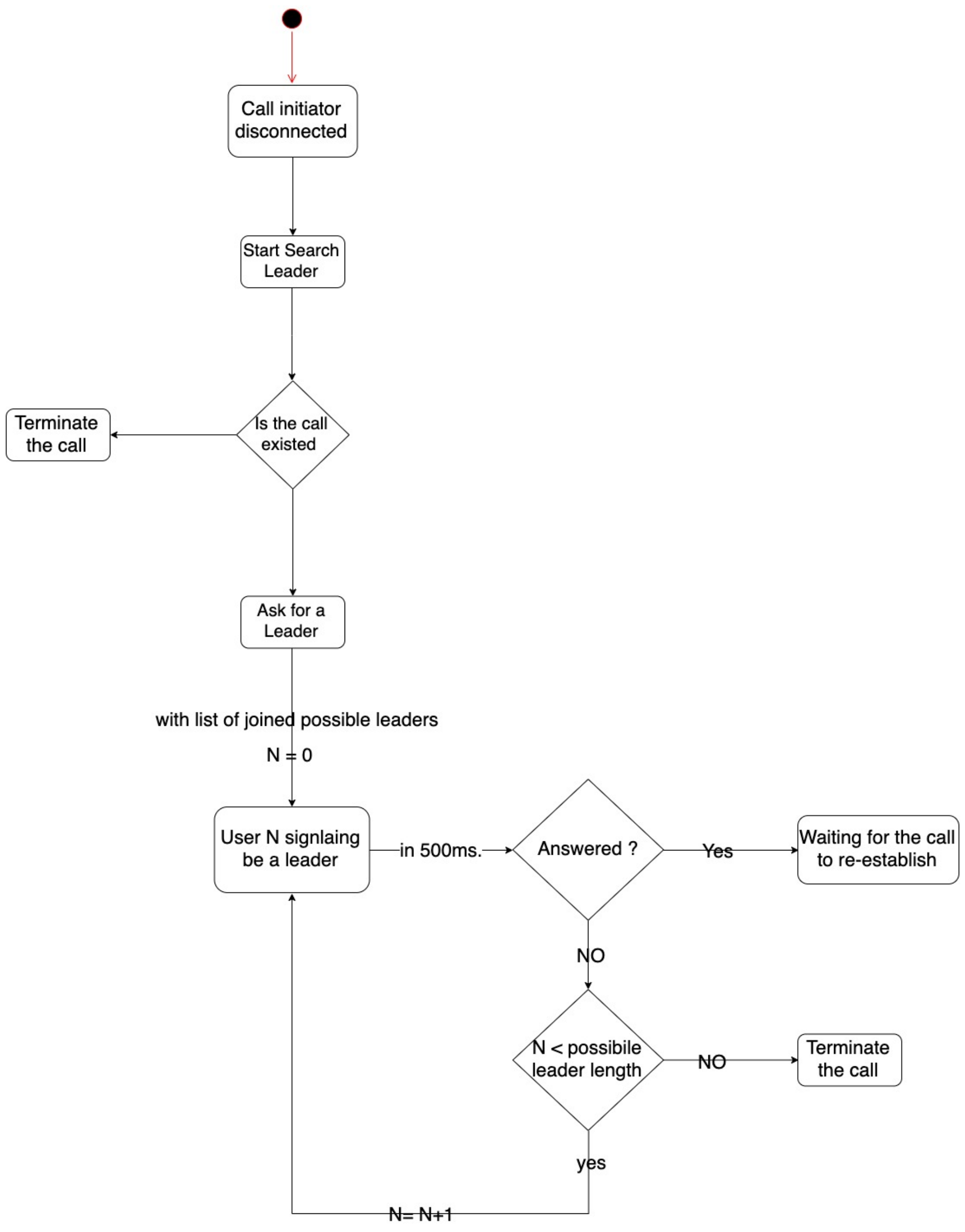

Call initiator disconnected
Start Search Leader
Is the call existed
Terminate the call
Ask for a Leader
with list of joined possible leaders
N = 0
User N signlaing be a leader
in 500ms.
Answered ?
Yes
Waiting for the call to re-establish
NO
N < possibile leader length
NO
Terminate the call
yes
N= N+1

Figure 5.9: Flow chart diagram for re-establishing a call via leader algorithm

5.11.3 User N signaling be a leader

In case of the asked user to be a leader did not answer with a "SET_LEADR" signal in 500ms then the users starts to ask the next in the list. "SET_LEADR" signal when it is distributed it is distributed to all users in the call which will normalize the difference in network bandwidth because that the creator of it will be the highest in bandwidth and the leader over take the call handling and distribution.

5.11.4 Waiting for the call re-establish

After the leader has been decided by "SET_LEADR" signal the callee waiting a create call silent signal from that leader and automatically accepting it with the same process and states of caller and callee.

5.11.5 Terminate the call

In case of all the users of the call has been disconnected then the bully algorithm has to be stopped and all the peer closes which it happens if there no "SET_LEADR" signal has come, peers are created and all the list have been asked.

6 Experiment and Analysis

In that chapter will go through the result of the experiment with a definition, analysis and discussion of various scenarios.

6.1 Relevant Parameters

Numerous controllable and non-controllable parameters may impact audio and video quality of a multimedia conversation and, therefore, affect the QoE. The following is an explanation of the different parameters used in the experiment.

In networking aspect, video conference traffic, which consists of audio and video data, is delay-and loss-sensitive. This means that quality of a video conference service is highly affected by obtained delay and drop of packets during the communication process. In one-way communication, voice requires delay (latency) to be lower than 150ms with loss rate below 1%, while video requires latency to be lower than 400ms with loss rate below 1%. For video standard-definition (SD-video) requires network latency between 400ms to 1s and loss rate between 1% to 2%. Higher value of latency or loss rate lead to lower quality of experience, which depends on end-user perception. Main aspect that affects obtained delay is network condition. [35]

Real-time communication is delay sensitive that makes arrivals of audio/video packets having higher priority than quality of audio/video. Video conference user can tolerate low quality audio/video but halting streams of audio/video can interrupt the communication process.

6.1.1 Packet Loss

Packet loss measures the number of packets lost versus the actual number of packets sent, occurs in both video and audio data transmission at certain times. Generally, packet loss is measured as Packet Loss Rate (PLR), which is defined in equation

$$PLR = \frac{\text{Number of lost packets}}{\text{Total number of transmitted packets}}$$

Number of lost packets is total number of packets that fails to reach the destinations which typically caused by network congestion. [35]

6.1.2 Delay (RTT)

The delay is a measure of how long it takes to transmit a packet from its source to its destination. The Round Trip Time (RTT) can also be used to describe the delay, by measuring the time it takes from the source to destination and back again. The delay may typically influence the QoE by having late play-out on the receiving end,

6.1.3 Jitter

Jitter is related to the delay, and it is known as the variation of the delay between consecutive packets. Jitter may also lead to reordering of packets.

6.1.4 Relevant Protocols

Following is a short description of the Real-time Transport Protocol (RTP) and the RTP Control Protocol (RTCP) which are both used in WebRTC.

RTP

RTP is a protocol typically used in multimedia communication applications and resides at the application layer, running on top of the User Datagram Protocol (UDP). RTP supports services like identifying the payload type, sequence numbering and delivery monitoring of both audio and video packets. In RTP, audio and video packets are separated and transmitted using different UDP ports.

RTCP

RTCP is used together with RTP and is sent as separate packets to provide information about QoS parameters in an active conversation. Statistics presented in RTCP includes the fraction of packets lost, the highest sequence number received, the cumulative number of packets lost, the inter-arrival jitter, and information about the delay, to name a few. The delay is measured by the time since the last report was received.

6.2 Confidence interval(CI)

The CI of a statistic may be regarded as a range of values, calculated from sample observations, that is likely to contain the true population value with some degree of uncertainty. Although the CI provides an estimate of the unknown population parameter, the interval computed from a particular sample does not necessarily include the true value of the parameter. Therefore, CIs are constructed at a confidence level, say 95%, selected by the user. This implies that were the estimation process to be repeated over and over with random samples from the same population, then 95% of the calculated intervals would be expected to contain the true value. Note that the stated confidence level is selected by the user and is not dependent on the characteristics of the sample. Although the 95% CI is by far the most commonly used, it is possible to calculate the CI at any given level of confidence, such as 90% or 99%. The two ends of the CI are called limits or bounds. CIs can be one or two-sided. A two-sided CI brackets the population parameter from both below (lower bound) and above (upper bound). A one-sided

CI provides a boundary for the population parameter either from above or below and thus furnishes either an upper or a lower limit to its magnitude. [36]

Calculating the lower and upper bounds is by

$$\text{Confidence interval boundaries} = \text{Sample Mean} \pm \text{Confidence Level}$$

6.3 Validation

Both network and hardware parameters affects the connection quality of a call and reliability. The validation of the results is done by repeating every experiment scenario multiple times and collecting the data which is the same result in the same setup but if the setup changed by making the call initiator a different user then the results is different and that is why Confidence Interval is important to the result statistics. All the experiments have been led in a controlled environment in order to guarantee as much correctness as possible.

Testbed View

The realized testbed consists of 4 PCs: 2 Desktops (QuadCore at 2.4GHz, 8GB of RAM), 2 laptops (QuadCore at 2.7GHz, 8GB of RAM)

WebRTC internal tool in Chrome and Mozilla browsers. This tool can track important network parameters including delay, jitter, packet loss, and transfer rate for certain periods of time.

Chrome offers a user interface for gathering statistics during a WebRTC-call, named webrtc-internals, The user interface provides a lot of different data and also separates the audio and video statistics. Some of the graphs, however, are hard to interpret and understand and therefore not so useful.

The used STUN and TURN server is Coturn [1] which is an open source project has been installed and configured to be used during the experiment for both STUN and TURN.

[1] https://github.com/coturn/coturn

6.4 Scenario 1: Reliability

In the first scenario will check the time performance of the leader algorithm by forcing network interruption for the call initiator and detect the result of how much time the users will need to choose a leader and the call being reconnected, will do that scenario in two network connection type, first is a STUN based connection and then later a TURN based connection by forcing the peer connection to depend on relay by adding "iceTransportPolicy":"relay" in the RTCPeerConnection configuration.

Time notified :Time when the user notified about the call initiator is gone. Time connected :Time needed that the peer re-connected after initiator is gone.

6.4.1 Using STUN Server

User	Browser	Role	Time notified (ms)	Time connected (ms)
1	Firefox	initiator	-	-
2	Firefox	candidate	128	701
3	Firefox	leader	610 (max.)	-
4	Firefox	candidate	295	768
1	Firefox	candidate	448	1044 (max.)
2	Firefox	initiator	-	-
3	Firefox	leader	497	-
4	Firefox	candidate	489	560
1	Firefox	leader	280	-
2	Firefox	candidate	478	930
3	Firefox	initiator	-	-
4	Firefox	candidate	433	631
1	Firefox	candidate	111	418
2	Firefox	candidate	046 (min.)	734
3	Firefox	leader	336	-
4	Firefox	initiator	-	-
1	Chrome	initiator	-	-
2	Chrome	candidate	278	883
3	Chrome	leader	199	-
4	Chrome	candidate	386	948

1	Chrome	candidate	271	571
2	Chrome	candidate	378	401 (min.)
3	Firefox	leader	206	-
4	Firefox	initiator	-	-
1	Chrome	initiator	-	-
2	Chrome	candidate	394	793
3	Firefox	leader	240	-
4	Firefox	candidate	290	700

Table 6.1: Reliability scenario based on STUN results

Mean (ms)	Confidence Level(95.0%) (ms)
720.142857142857	111.511799635231

Table 6.2: Reliability scenario based on STUN Confidence Level

Lower (ms)	Higher (ms)
608.631058	831.654657

Table 6.3: Reliability scenario based on STUN Confidence Interval

6.4.2 Using TURN Server

User	Browser	Role	Time notified (ms)	Time connected (ms)
1	Firefox	initiator	-	-
2	Firefox	candidate	336	1258
3	Firefox	leader	540	-
4	Firefox	candidate	353	579 (min.)
1	Firefox	candidate	942 (max.)	1386
2	Firefox	initiator	-	-
3	Firefox	leader	330 (min.)	-
4	Firefox	candidate	485	954
1	Firefox	leader	508	-
2	Firefox	candidate	853	1197

3	Firefox	initiator	-	-
4	Firefox	candidate	632	881
1	Firefox	candidate	501	946
2	Firefox	candidate	923	1516 (max.)
3	Firefox	leader	454	-
4	Firefox	initiator	-	-
1	Chrome	initiator	-	-
2	Chrome	candidate	522	631
3	Chrome	leader	339	-
4	Chrome	candidate	712	850
1	Chrome	candidate	461	476
2	Chrome	candidate	454	460
3	Firefox	leader	370	-
4	Firefox	initiator	-	-
1	Chrome	initiator	-	-
2	Chrome	candidate	378	638
3	Firefox	leader	404	-
4	Firefox	candidate	393	1071

Table 6.4: Reliability scenario based on STUN results

Mean (ms)	Confidence Level(95.0%) (ms)
974.5	163.126070136362

Table 6.5: Reliability scenario based on TURN Confidence Level

Lower (ms)	Higher (ms)
811.37393	1137.62607

Table 6.6: Reliability scenario based on TURN Confidence Interval

6.5 Discussion Scenario 1

Using STUN

In Table 6.1 stated the results from the experiment by repeating the same experiment multiple times in the same browsers first Firefox and make every time a different initiator of the call so the roles has to be changed of the call leader and the candidates, then later on repeat it with chrome and with a mixing between the 2 browsers. The results was not for sure constant value of time which is realistic regrading every device has it is own hardware specification and 2 users is connected to the internet via WiFi and the other 2 is connected via Ethernet cable which could have an impact on the network performance. The reached minimum value is 401ms and the maximum is 1044ms. Regarding the value is variant based on the user device and the network connection type, therefore confidence interval is applied in Table 6.2 is the results of the Time connected column which is the time needed that the peer has been connected after the call initiator is gone, which will allow one later on to detect the confidence interval boundaries by subtracting the confidence level from the the mean to detect the lower boundary and summing the mean with the confidence level to detect the upper boundary. In Table 6.3 as shown the confidence interval which is [608, 831]ms.

Using TURN

The test scenario and condition is still the same with only 1 change which is changing the peer configuration to use the TURN server instead. As shown in Table 6.4 the minimum is 579ms and the maximum is 1.516ms, in Table 6.5 the mean, confidence level of the Time connected column and in Table 6.6 shown the confidence interval which is [811, 1138]ms.

Overall

The overall results shows that the single point of failure problem can be solved but the time is needed to re-connect the peers with a new leader depends on different factors like when the user have notified that the initiator is gone ? which is depending on the signaling server performance, speed and the users bandwidth and network speed, which can be shown in Tables 6.1 and 6.4 column Time notified and how fast the signaling server was able that the call initiator has gone and then later the new elected leader which will be the new initiator device specification and network conditions. One can imagine scenario if the user which was elected as a leader was connected via mobile phone and with mobile Long Term Evolution (LTE), the mobile phone hardware specification is less than a computer which will have an impact on the election process speed and later on the call quality for mixing the media tracks from the other peers. What can be a possible solution for that is by not allowing the users which is connected via mobile phone to be a leader form the beginning but still the users devices and environment play a big role which cannot be controllable.

6.6 Scenario 2: Rejoin after network interrupt

In that scenario will see the results of how much time the user to be reconnected after network interruption, the network interruption caused by refreshing the page or plug out and in the internet cable.

Notified: Time that got a silent notification from the call initiator to join the call. Connected: Time needed to have the peer connected. Difference: Time Connected - Time Notified

6.6.1 Using STUN server

Notified (ms)	Connected (ms)	Difference (ms)
378 (max.)	424	46
332 (min.)	389	57
334	370 (min.)	36 (min.)
339	381	42
373	484 (max.)	111 (max.)

Table 6.7: Rejoin after network interrupt scenario based on STUN results

Mean (ms)	Confidence Level(95.0%) (ms)
58.4	37.729018

Table 6.8: Rejoin after network interrupt scenario based on STUN Confidence Level

Lower (ms)	Higher (ms)
20.670982	96.129018

Table 6.9: Rejoin after network interrupt scenario based on STUN Confidence Interval

6.6.2 Using TURN server

Notified (ms)	Connected (ms)	Difference (ms)
306 (min.)	432	126
396 (max.)	533 (max.)	137
324	421 (min.)	97 (min.)
348	481	133
343	483	140 (max.)

Table 6.10: Rejoin after network interrupt scenario based on TURN results

Mean (ms)	Confidence Level(95.0%) (ms)
126.6	21.5527978

Table 6.11: Rejoin after network interrupt scenario based on TURN Confidence Level

Lower (ms)	Higher (ms)
105.0472022	148.1527978

Table 6.12: Rejoin after network interrupt scenario based on TURN Confidence Interval

6.7 Discussion Scenario 2

In Tables 6.7 and 6.10 is the results of scenario 2 with STUN and TURN respectively. Re-connecting the user after network connection interruption based on the database and signaling server is done after the network is back the signaling server check the database if the user was in a call and then send a "ALLOW_TO_JOIN" signal to the call initiator with the current user id, then the initiator send a silent JOIN signal that process is shown in Notified column how much it takes since the user is back online, then after that the peers try to connect and the initiators sends the media tracks again. After applying the CI found that it needs [21,96]ms on STUN based and [105,148]ms on TURN based, Which is acceptable in a browser based audio/video conferencing application, so if the user has disconnected will get back to the call automatically and if the user by fault refresh the page or the browser crashes will be able automatically rejoined.

6.8 Scenario 3: Scalability

In that scenario will measure how much users can be added using the proposed solution and pipeline other peer connections, by generating a peer connection every 500ms and add it to the initiator which will distribute it to the other peers. The data is collected using 2 methods first method is by using ps [2], secoend method is htop [3] and the table 6.13 is the average results of the collected data. The used device resources specifications was QuadCore at 2.7GHz, 8GB of RAM.

Will be shown the chart diagrams for the CPU and the memory next and in Figures 6.1 and 6.2 shows that the results is limited to 60 peers regarding that the device resources was not able to handle more peers there was a plan to add 120 more peers but more than 60 peer let the browser crashes and the device run out of memory.

[2]ps displays information about a selection of the active processes in linux and unix systems. by command "ps −p <pid> −o %cpu,%mem" http://man7.org/linux/man-pages/man1/ps.1.html

[3]htop is an interactive system-monitor process-viewer and process-manager. It is designed as an alternative to the Unix program top. It shows a frequently updated list of the processes running on a computer, normally ordered by the amount of CPU usage. by command " htop -p <pid>" https://hisham.hm/htop/

Number of Users	CPU Usage(%)	Memory Usage (%)
5	62.1	3.55
10	94.2	4.15
15	119.8	4.9
20	126.8	7.2
25	129.0	7.85
30	133.7	8.7
60	206.1	10.75

Table 6.13: Scalability Memory and CPU results

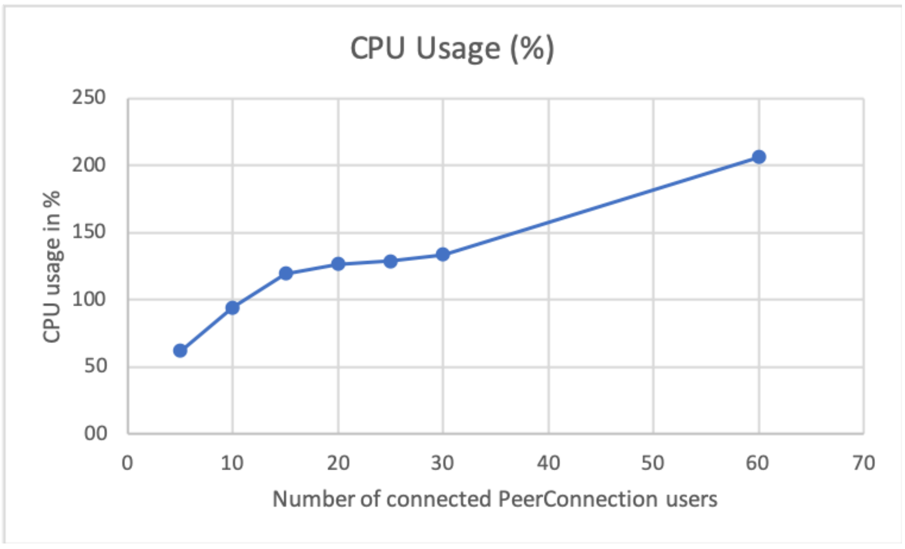


Figure 6.1: Scalability result impact on CPU

6.9 Discussion Scenario 3

As shown in table 6.13 the impact of creating many peers on the same device and distribute the media tracks to the other peer, From Figure 6.1 one can see it clear that by increasing the users (peers) in the call has an impact on the CPU usage which is increased by increasing the users (peers) as well has an impact on the memory as shown in Figure 6.2. So by adding more peers having an exponential increase on the CPU usage and memory foot print which make the solution of mixing the media tracks and creating calls be on the call initiator device is limited

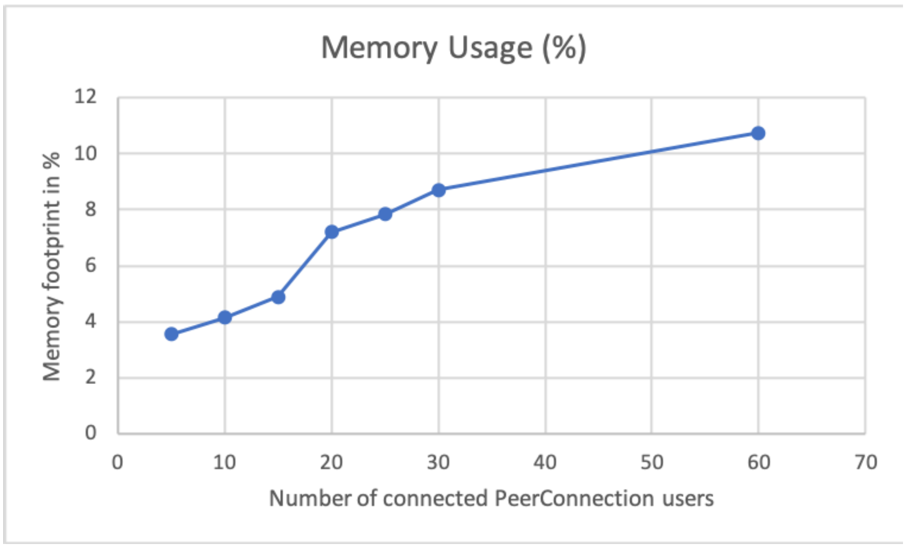


Figure 6.2: Scalability result impact on Memory

to a reasonable amount of users on the call. In real life realistic situations the meetings can consist of maximum 5 or 7 people but then when the people number is increased than that it become a live video streaming which for sure not an option for that solution but for small number of people meeting still practical and functional.

6.9.1 Time and Packets Analysis

In Figure 6.3 will see the impact of adding more users with more audio/video tracks. First by creating a call with 1 user (2 Users in the call) the packets loss is 5, jitter is 0.02ms and round trip time is 0.8ms, by adding more users up to 5 will find that packet loss increased to 60, jitter to 0.04ms and round trip time to 1.75ms at one point but then later is decreased again, the other factor is time which is has an impact because the call increased to 5 users. In Tables 2.1 and 2.2 which is the work of Akhmad Alimudin et al.[19] there was an increased time delay based on the number of the users and a packet loss is increased based as well on the number of the users in the call and the bandwidth limit which it was the same situation regarding the users in the same network with the same bandwidth limit.

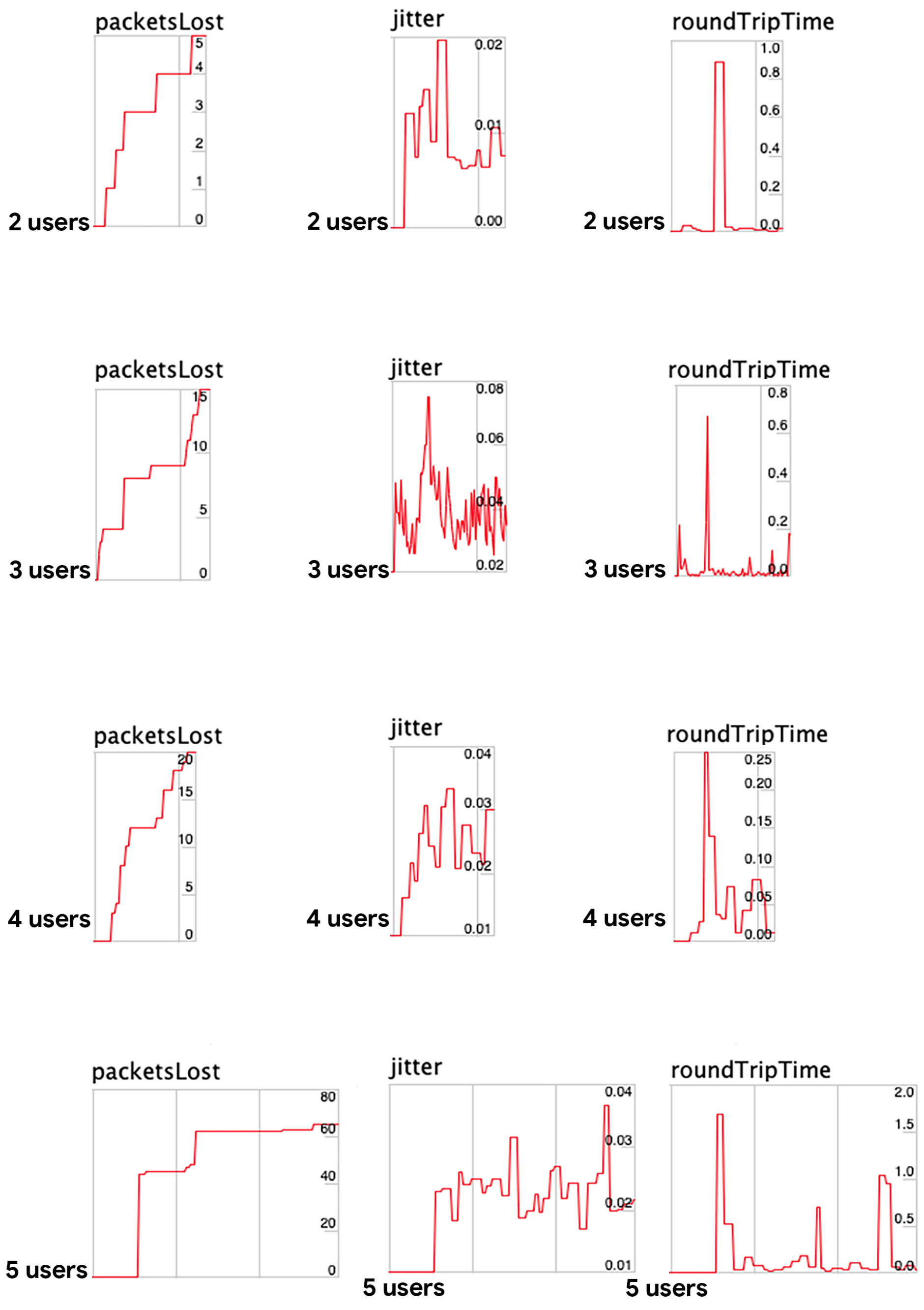


Figure 6.3: Round Trip Time and Packet analysis for multiple users

7 Conclusion

Nowadays professional communication and collaboration systems are important in any working environment. This thesis presented an example for a reliable architecture for WebRTC by implementing a direct and multiple candidates calls using different media source tracks like audio, video and screen sharing. The requirement was to not involve neither SFU nor MCU and instead doing the distribution work on the client side of the call initiator.

The main challenge encountered when the call initiator is gone by a network failure or a fault, regarding that the used network topology is a star network topology and it is obvious that it suffer from a single of point failure challenge, the solution for that challenge was replication of a list of user which can be used to re-establish the whole call again in case of the challenge occurs by applying leader election algorithm and the chosen here in this thesis was a bully leader algorithm.

The experiment it self and the discussed scenarios in the previous chapter was successful by keeping in mind the matter of the trade off between cost and reliability, by applying the call tracks distribution and solving the single point of failure in the client side but had a limitations as mentioned before both network and hardware parameters affects the connection quality of a call and reliability, the time factor of notifying that the call initiator or leader is gone and the time the clients needed to re-establish the call is variant based on the user device and the network connection limitation which leads the solution to different scenarios. One can imagine scenario if the user which was elected as a leader was connected via mobile phone and with LTE network, the mobile phone hardware specification is less than a computer which will have an impact on the election process speed and later on the call quality for mixing the media tracks from the other peers. Based on the work of G. Carullo et al. for evaluating the performance of WebRTC over LTE, an experiment on WebRTC audio/video calls between two mobile users, the experimental results showed how the typical parameters are influenced by different network setups and the deterioration of multimedia flows quality when more realistic setups are considered [37]. For a realistic scenario of an online meeting consist of 5 or 7 users in the call the proposed approach will still be able to handle the call with the average computer of nowadays market. Future work will be considering a multiple SFU or MCU and scale WebRTC for the use of live video streaming and applying the election algorithm for the servers to handle a scaled streaming with testing of QoE and QoS from geographically distribution network.

Acronyms

Ajax Asynchronous JavaScript and XML. 31
API Application Program Interfaces. 5, 16

C/S Client/Server. 2
CI Confidence interval. vii, 55
CPU Central processing unit. 8
CSS Cascading style sheets. 31

DOM Document Object Model. 31

GCC Google Congestion Control. 15
GUI Graphical User Interface. 2

HTML Hypertext Markup Language. 31

ICE Interactive Connectivity Establishment. 15

JSON JavaScript Object Notation. 32

LAN Local Area Network. 8
LTE Long Term Evolution. 61, 67

MCU Multi-point Control Unit. 7

NAT Network address translation. 15

P2P peer-to-peer. 3, 6, 9
PLR Packet Loss Rate. 54

QoE Quality of Experience. 8, 54
QoS quality of service. 6, 55

RTCP RTP Control Protocol. 54

RTP	Real-time Transport Protocol. 8, 54
RTT	Round Trip Time. 54
SCTP	Stream Control Transmission Protocol. 15
SD	standard-definition. 53
SDP	Session Description Protocol. 15
SFU	Selective Forwarding Unit. 6
SIP	Session Initiation Protocol. 3
SRTP	Secure Real-time Transport Protocol. 7, 15
STUN	Session Traversal Utilities for NAT. 15
TURN	Traversal Using Relays for NAT. 15
UDP	User Datgram Protocol. 16, 55
VoD	Video on Demand. 6
WAN	Wide Area Network. 8
WebNSM	WebRTC signalling mechanism. 8
WebRTC	Web Real-Time Communication. 3, 7, 15, 16
XML	Extensible Markup Language. 30
YAML	YAML Ain't Markup Language. 32

Bibliography

[1] D. Henriyan, Devie Pratama Subiyanti, R. Fauzian, D. Anggraini, M. Vicky Ghani Aziz, and Ary Setijadi Prihatmanto, “Design and implementation of web based real time chat interfacing server”, in *2016 6th International Conference on System Engineering and Technology (ICSET)*, Oct. 2016, pp. 83–87. DOI: 10.1109/ICSEngT.2016.7849628 (cit. on p. 1).

[2] S. Yuzhuo and H. Kun, “Design and realization of chatting tool based on web”, in *2013 3rd International Conference on Consumer Electronics, Communications and Networks*, Nov. 2013, pp. 225–228. DOI: 10.1109/CECNet.2013.6703312 (cit. on p. 1).

[3] Huaying Xue and Yuan Zhang, “A webrtc-based video conferencing system with screen sharing”, in *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, Oct. 2016, pp. 485–489. DOI: 10.1109/CompComm.2016.7924748 (cit. on pp. 2, 14, 16).

[4] Y. Lu, S. Li, and H. Shen, “Virtualized screen: A third element for cloud–mobile convergence”, *IEEE MultiMedia*, vol. 18, no. 2, pp. 4–11, Feb. 2011. DOI: 10.1109/MMUL.2011.33 (cit. on p. 2).

[5] I. Dujlović and Z. Đurić, “Cross-platform web based real-time communication in web tv and video on demand systems”, in *2015 57th International Symposium ELMAR (ELMAR)*, Sep. 2015, pp. 65–68. DOI: 10.1109/ELMAR.2015.7334497 (cit. on pp. 2, 3, 15, 16).

[6] I. Dujlović and Z. Đurić, “Cross-platform web based real-time communication in web tv and video on demand systems”, in *2015 57th International Symposium ELMAR (ELMAR)*, IEEE, 2015, pp. 65–68 (cit. on p. 3).

[7] T. Bach, M. Maruschke, J. Zimmermann, K. Hänsge, and M. Baumgart, “Combination of ims-based iptv services with webrtc”, in *The Ninth International Multi-Conference on Computing in the Global Information Technology*, 2014, pp. 140–145 (cit. on p. 3).

[8] J. K. Nurminen, A. J. R. Meyn, E. Jalonen, Y. Raivio, and R. Garcıa Marrero, "P2p media streaming with html5 and webrtc", in *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Apr. 2013, pp. 63–64. DOI: 10.1109/INFCOMW.2013.6970739 (cit. on p. 3).

[9] C. Medina-López, J. P. G. Ortiz, J. Naranjo, L. Casado, and V. González-Ruiz, "Iptv using p2psp and html5+ webrtc", in *The Fourth W3C Web and TV Workshop (Web & TV Convergence), page Paper Submission*, vol. 5, 2014 (cit. on p. 3).

[10] Ó. F. Salas and H. Kalva, "Architecting social tv", in *Applications of Digital Image Processing XXXVI*, International Society for Optics and Photonics, vol. 8856, 2013, 88560G (cit. on p. 3).

[11] S. Loreto and S. P. Romano, "Real-time communications in the web: Issues, achievements, and ongoing standardization efforts", *IEEE Internet Computing*, vol. 16, no. 5, pp. 68–73, Sep. 2012. DOI: 10.1109/MIC.2012.115 (cit. on p. 3).

[12] K. Singh and V. Krishnaswamy, "A case for sip in javascript", *IEEE Communications Magazine*, vol. 51, no. 4, pp. 28–33, Apr. 2013. DOI: 10.1109/MCOM.2013.6495757 (cit. on p. 3).

[13] A. Amirante, T. Castaldi, L. Miniero, and S. P. Romano, "On the seamless interaction between webrtc browsers and sip-based conferencing systems", *IEEE Communications Magazine*, vol. 51, no. 4, pp. 42–47, Apr. 2013. DOI: 10.1109/MCOM.2013.6495759 (cit. on p. 3).

[14] M. J. Werner, C. Vogt, and T. C. Schmidt, "Let our browsers socialize: Building user-centric content communities on webrtc", in *2014 IEEE 34th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, Jun. 2014, pp. 37–44. DOI: 10.1109/ICDCSW.2014.35 (cit. on p. 3).

[15] R. L. Barnes and M. Thomson, "Browser-to-browser security assurances for webrtc", *IEEE Internet Computing*, vol. 18, no. 6, pp. 11–17, Nov. 2014. DOI: 10.1109/MIC.2014.106 (cit. on p. 3).

[16] L. Desmet and M. Johns, "Real-time communications security on the web", *IEEE Internet Computing*, vol. 18, no. 6, pp. 8–10, Nov. 2014. DOI: 10.1109/MIC.2014.117 (cit. on p. 3).

[17] M. Wenzel and C. Meinel, "Full-body webrtc video conferencing in a web-based real-time collaboration system", in *2016 IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, May 2016, pp. 334–339. DOI: 10.1109/CSCWD.2016.7566010 (cit. on pp. 5, 6).

[18] W. Kim, H. Jang, G. Choi, I. Hwang, and C. Youn, "A webrtc based live streaming service platform with dynamic resource provisioning in cloud", in *2016 IEEE Region 10 Conference (TENCON)*, Nov. 2016, pp. 2424–2427. DOI: 10.1109/TENCON.2016.7848466 (cit. on pp. 6, 7).

[19] A. Alimudin and A. F. Muhammad, "Online video conference system using webrtc technology for distance learning support", in *2018 International Electronics Symposium on Knowledge Creation and Intelligent Computing (IES-KCIC)*, Oct. 2018, pp. 384–387. DOI: 10.1109/KCIC.2018.8628568 (cit. on pp. 7, 8, 65).

[20] N. M. Edan, A. Al-Sherbaz, and S. Turner, "Webnsm: A novel scalable webrtc signalling mechanism for many-to-many video conferencing", in *2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC)*, Oct. 2017, pp. 27–33. DOI: 10.1109/CIC.2017.00015 (cit. on p. 8).

[21] 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)*, Jan. 2014, pp. 1–6. DOI: 10.1109/WCCAIS.2014.6916588 (cit. on pp. 8, 9).

[22] (2019). Ietf — the websocket protocol page, [Online]. Available: https://tools.ietf.org/html/rfc6455#section-1 (visited on 10/03/2019) (cit. on p. 9).

[23] A. S. Tanenbaum and M. v. Steen, *Distributed systems 3ed ed.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2017, ISBN: 978-90-815406-2-9 (cit. on pp. 9–14, 19–21, 23).

[24] B. Jansen, T. Goodwin, V. Gupta, F. Kuipers, and G. Zussman, "Performance evaluation of webrtc-based video conferencing", *ACM SIGMETRICS Performance Evaluation Review*, vol. 45, pp. 56–68, Mar. 2018. DOI: 10.1145/3199524.3199534 (cit. on p. 16).

[25] T. Ichimura and T. Uemoto, “Analysis of the social community based on the network growing model in open source software community”, in *2015 IEEE 8th International Workshop on Computational Intelligence and Applications (IWCIA)*, Nov. 2015, pp. 149–153. DOI: 10.1109/IWCIA.2015.7449480 (cit. on p. 16).

[26] K. Stewart, “Lessons learned from the linux kernel: Creating sustained healthy communities”, in *2018 IEEE/ACM 1st International Workshop on Software Health (SoHeal)*, May 2018, pp. 1–1 (cit. on p. 16).

[27] (2019). Webrtc — project home page, [Online]. Available: https://webrtc.org/ (visited on 09/15/2019) (cit. on p. 17).

[28] P. Nuño, F. G. Bulnes, J. C. Granda, F. J. Suárez, and D. F. García, “A scalable webrtc platform based on open technologies”, in *2018 International Conference on Computer, Information and Telecommunication Systems (CITS)*, Jul. 2018, pp. 1–5. DOI: 10.1109/CITS.2018.8440161 (cit. on pp. 17, 18).

[29] W. Elleuch, “Models for multimedia conference between browsers based on webrtc”, in *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct. 2013, pp. 279–284. DOI: 10.1109/WiMOB.2013.6673373 (cit. on p. 18).

[30] N. M. Edan, A. Al-Sherbaz, and S. Turner, “Design and evaluation of browser-to-browser video conferencing in webrtc”, in *2017 Global Information Infrastructure and Networking Symposium (GIIS)*, Oct. 2017, pp. 75–78. DOI: 10.1109/GIIS.2017.8169813 (cit. on p. 19).

[31] A. Patooghy, S. G. Miremadi, A. Javadtalab, M. Fazeli, and N. Farazmand, “A solution to single point of failure using voter replication and disagreement detection”, in *2006 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing*, Sep. 2006, pp. 171–176. DOI: 10.1109/DASC.2006.15 (cit. on p. 19).

[32] P. Wang, X. Wu, and H. Yang, “Analysis of the efficiency of data transmission format based on ajax applications”, in *2011 International Conference of Information Technology, Computer Engineering and Management Sciences*, vol. 4, Sep. 2011, pp. 265–268. DOI: 10.1109/ICM.2011.199 (cit. on pp. 30, 32).

[33] A. Šimec and Magličić, “Comparison of json and xml data formats”, Sep. 2014 (cit. on pp. 30, 32).

[34] (2019). Yaml — project home page, [Online]. Available: `https://yaml.org/spec/1.2/spec.html` (visited on 11/10/2019) (cit. on p. 32).

[35] Y. Bandung, L. B. Subekti, D. Tanjung, and C. Chrysostomou, "Qos analysis for webrtc videoconference on bandwidth-limited network", in *2017 20th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, Dec. 2017, pp. 547–553. DOI: `10.1109/WPMC.2017.8301873` (cit. on pp. 53, 54).

[36] A. Hazra, "Using the confidence interval confidently", *Journal of Thoracic Disease*, vol. 9, pp. 4124–4129, Oct. 2017. DOI: `10.21037/jtd.2017.09.14` (cit. on p. 56).

[37] G. Carullo, M. Tambasco, M. D. Mauro, and M. Longo, "A performance evaluation of webrtc over lte", in *2016 12th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, Jan. 2016, pp. 1–6 (cit. on p. 68).

Declaration

I declare within the meaning of part 16(5) of the General Examination and Study Regulations for Bachelor and Master Study Degree Programs at the Faculty of Engineering and Computer Science and the Examination and Study Regulations of the International Degree Course Information Engineering that: this Bachelor Thesis has been completed by myself/ourselves independently without outside help and only the defined sources and study aids were used. Sections that reflect the thoughts or works of others are made known through the definition of sources.

Hamburg, 02.01.2020 Mohamed Badawi