Design of WebRTC based Real Time Streaming System

BTP-1 report submitted to

Indian Institute of Technology Kharagpur

in partial fulfilment for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Divyang Mittal

(17CS10012)

Under the supervision of

Prof. Sandip Chakraborty

Department of Computer Science and Engineering

Indian Institute of Technology Kharagpur

Autumn Semester, 2020-21

Nov 23, 2020

DECLARATION

I certify that

(a) The work contained in this report has been done by me under the guidance of my supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Date: Nov 23, 2020 (Divyang Mittal)
Place: Kharagpur (17CS10012)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

KHARAGPUR - 721302, INDIA

CERTIFICATE

This is to certify that the project report entitled "**Design of WebRTC based Real Time Streaming System**" submitted by **Divyang Mittal** (Roll No. 17CS10012) to Indian Institute of Technology Kharagpur towards partial fulfilment of requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering is a record of bona fide work carried out by him under my supervision and guidance during Autumn Semester, 2020-21.

Date: Nov 23, 2020

Place: Kharagpur

Prof. Sandip Chakraborty
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
Kharagpur - 721302, India

Abstract

Name of the student: **Divyang Mittal** Roll No: **17CS10012**

Degree for which submitted: **Bachelor of Technology**

Department: **Department of Computer Science and Engineering**

Thesis title: **Design of WebRTC based Real Time Streaming System**

Thesis supervisor: **Prof. Sandip Chakraborty**

Month and year of thesis submission: **Nov 23, 2020**

Real-Time communication has advanced leaps and bounds in past few years. Since a long time, Voice over Internet protocol is the most widely used protocol for voice communication. Over the years, We have come across development of video communication as well. Video communication spans over many domains such as video calling, video conference, live streams, etc.

In this work, We look at some of the most widely used methods to build a video conferencing system. We analyze their performance to identify their particular advantages and disadvantages. We show the use of WebRTC in development of these systems. Finally we propose our own design to build a video conferencing system.

Acknowledgements

I would like to thank my supervisor Dr. Sandip Chakraborty, assistant professor, dept. of CSE, IIT Kharagpur for his valuable guidance and support all throughout the course of my B.Tech project work.

I would like to thank Mr Bishakh Chandra Ghosh, PhD Scholar, dept. of CSE, IIT Kharagpur and Mr. Abhijit Mondal, PhD Scholar, dept. of CSE, IIT Kharagpur, who have helped in this project. Without their tremendous support, this work might not be finished.

I would like to express my gratitude to all faculty members of Department of Computer Science and Engineering for guiding us and providing knowledge in various subjects over the past four years.

Finally, my deep and sincere gratitude to my family and friends in IIT Kharagpur for their continuous love and support.

Contents

List of Figures

List of Tables

Abbreviations

QoE	**Q**uality **o**f **E**xperience
NAT	**N**etwork **A**ddress **T**ranslation
STUN	**S**ession **T**raversal **U**tilities for **N**AT
TURN	**T**raversal **U**sing **R**elays arounf **N**AT
MCU	**M**ultipoint **C**ontroller **U**nit
SFU	**S**elective **F**orwarding **U**nit

Chapter 1

Introduction

1.1 Motivation

Since the advent of COVID-19, We all have been constraint to our homes. The necessity of carrying on our work, education, social lives, etc has lead to a tremendous increase in the use of video conferencing systems. As reported by zoom, They had 10 million daily users in December 2019 which increased to 300 million daily users in April 2020 at the beginning of lockdown in various countries. Video conferencing involves a large amount of data and CPU usage. This exhibits how ”efficient” video conferencing systems are the need of the hour.

The security provided by these video conferencing systems concerning handling user data has also come under scrutiny. Most of these systems use a server-client architecture due to which these systems have access to the user’s data. This is a violation of user’s privacy as their data might be used without their consent. This calls for a decentralized network architecture to ensure security.

1.2 Case study of video conferencing systems

In this section, We show the working of a few popular conferencing tools available to us. We want to identify the methods used by these systems to make them scalable and efficient. As we understand the internal architecture of these systems is not public, We only cover partial architecture information.

1. Skype

   Video chat in Skype started as a peer-to-peer server-client architecture. Hence the name ”sky peer-to-peer”. At that time it used supernodes to store the skype user directory information. An authorized user with a good network connection was used a supernode. After identifying the required users a peer-to-peer connection was created. Later, Microsoft started storing these supernodes in their own data centers. This helped in increasing the number of users who can simultaneously join a meeting. But lead to the above-mentioned problem of violation of user's privacy.

2. Zoom

   Zoom is by far the most popular of these systems. It has the maximum capacity of the number of users among all systems. Zoom has 9 data centers all over the world to handle its traffic. Zoom has been built from the ground up to optimize video conferencing. One such example is Zoom uses the SVC (Scalable Video Codec) codec over AVC which is used in most of the systems. In AVC, multiple streams need to be sent to send multiple bitrates whereas SVC uses multiple layers in a single stream.

3. Google Meet

   Google Meet uses WebRTC protocol in its architecture. Due to this, a person can join a meeting without any external plugins. One major advantage for google meet is it enjoys the freedom provided by chrome as it is able to modify the browser to suit its needs.

1.3 Problem Statement

Building a video conferencing system which satisfies the following criteria

1. decentralized system to ensure user privacy

2. minimization of network usage in the system

3. QoE maximization to the end user

Chapter 2

WebRTC

2.1 What is WebRTC?

Web Real-time communication better known as WebRTC is a peer-2-peer protocol for browsers and mobile applications. This means it allows browsers to directly connect and share media information such as audio and video in real-time. This helps to remove the need for any external plugins in the system. It is free and open-source allowing us easy access. WebRTC is developed and standardized through W3C and the Internet Engineering Task Force IETF together with industry leaders like Google and Apple. Since this is a peer-2-peer protocol, We have used this in building our own video conferencing system.

2.2 Working

Now we look into the process involved in the creation of a single p2p connection using WebRTC. The sharing of media resources is p2p in WebRTC but we require a central server to create this connection. This is done with the help of a signaling server.

2.2.1 Signaling server

The tasks of signaling servers involve the exchange of messages such as a connection request, error messages, etc. All this information about the media (type and resolution), network connection information, etc is transferred using Session Description Protocol(SDP). This is used along with Interactive connection establishment(ICE) to establish a connection. An architecture of peers with a signaling server is shown in figure 2.1. To start a connection, the peers share their respective SDP's with other peers using a signaling server. The algorithm for sharing of these SDP's looks like the following:

1. A peer creates a new RTCPeerConnection using its media streams which we can get using getUserMedia() API.
2. The peer sends a SDP offer using this newly created RTCPeerConnection object through the signaling server and saves it in its own localDescription variable.
3. The receiver of the sdp offer saves it in the remoteDescription and generates it's own SDP answer, stores it in its own localDescription and sends the answer to the sender.
4. The initial peer finally saves it in its own local description establishing the connection.

   After this step signaling server is only used to send text information between the peers such as error messages.

2.2.2 NAT Traversal

Although, the sharing of SDP packets helps in the exchange of network information between peers. The peers might still not be able to communicate with each other.

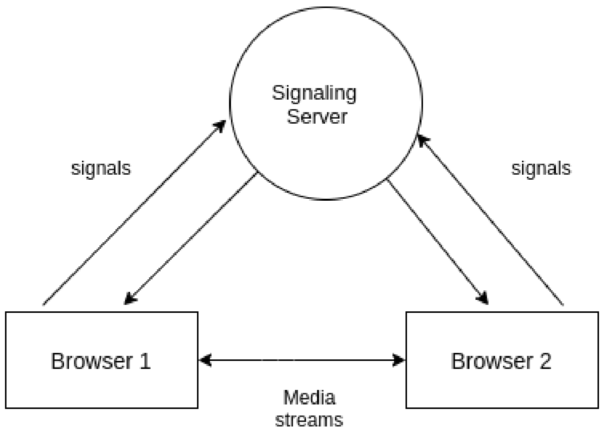

FIGURE 2.1: A p2p connection with a signaling server

There is a case that a client is unable to identify it's own IP address when it is behind a network address translation(NAT). We have to employ certain techniques to identify the right IP in this case. These techniques are known as NAT traversal.

One of the techniques involves the use of STUN servers. A browser may connect to this STUN server. The architecture for the p2p network using stun server has been shown in figure 2.2. These STUN servers then return potential candidates for the browser's IP address, known as ICE candidates. These candidates are passed to the peers using the signaling channel. After the exchange of SDP, each browser(peer) tries out the various candidates in the list to identify the right IP address.

The algorithm for identifying the right candidate involves selecting a candidate from the list and sending it to the stun server which consequently tries to connect with the browser. If an answer is received by the browser, We identify it as the right candidate.

The STUN server may not be enough in case of more secure networks. In that case, a TURN server is employed. A TURN server creates a public IP address and port for the browser. This acts as a relay address that forwards all the packets received and send by this browser. An additional feature of TURN servers is that it allows us to use TCP instead of the usual UDP used in WebRTC.

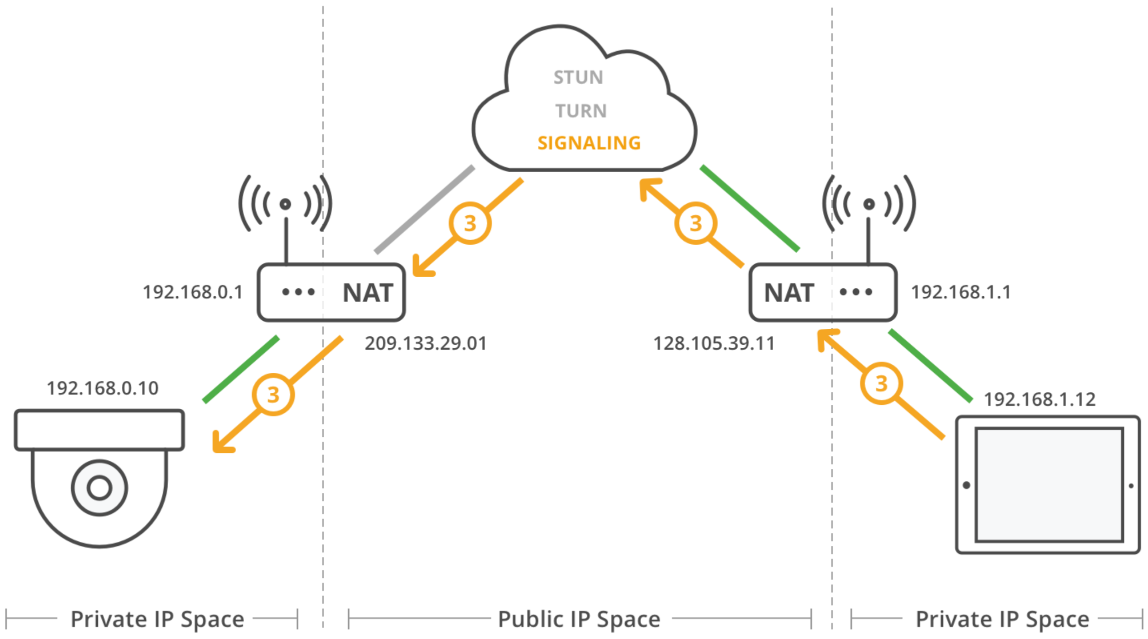

FIGURE 2.2: A p2p connection using STUN server (6)

A duplex connection is established between the peers once the sdp offers have been exchanged and ice candidates are resolved. After this both peers can add media streams in the connection to share their media content. Next we look into the WebRTC API.

2.3 Advantages and Disadvantages

We have compiled the various advantages and disadvantages of WebRTC to justify our selection of this protocol.

2.3.1 Advantages

1. Provides getUserMedia() API to access the various media streams from input through webcam, microphone and screen sharing.

2. Open source and has a good amount of documentation.

3. Automatically adjusts the quality of video and audio with the bandwidth available to it.

4. Provides low latency since the media streams are shared over p2p network.

5. No additional plugins or extension need to be developed. Meetings can be joined directly from browser.

6. High level of security: all connections are protected (HTTPS) and encrypted (SRTP).

2.3.2 Disadvantages

1. Compatible with very few browsers, Moreover works with only some versions of specific browsers.

2. Custom signaling server needs to be built since there is no inbuilt signaling system available.

Since our main task is to study the various elements involved in the development of video conferencing systems. Compatibility does not matter to us as long as the system works well with few browsers. We built a signaling server to overcome the other disadvantage.

Chapter 3

Design of Video conferencing systems

In this chapter, We will show our method to scale a video conferencing system to accommodate more peers. We have shown how two peers are connected. All the peers in a meeting have to be organized into a network topology to share their media elements with each other. We will first show two implemented methods, an industrially accepted solution, and finally our proposed algorithm.

3.1 Mesh Topology

A simple approach is to make a complete mesh of all peers. This means all the peers are connected to one another as shown in figure 3.1. If there are N nodes, each node will receive N-1 video and audio streams and there will be N*(N-1) peer connections in the network. All of the incoming and outgoing streams have to be decrypted and encrypted respectively. This requires a huge computation power and may not be feasible as the number of peers in the network increase. Also since we are receiving N-1 video streams, network usage is highest in this case.

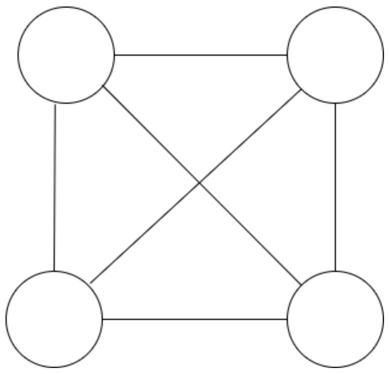

Figure 3.1: A full mesh network of 4 peers

3.2 Star Topology

A much better method is to use a star topology. In this case, all the peers are only connected to a centrally located server as shown in figure 3.2. In this manner, Each peer has only one connection summing to N*2 total connections in the network. Hence, All the streams go through the server that acts as a gateway. Each peer has to send its stream, only to the server. But, what about the other way round? Each peer still has to receive the streams of all the users. To overcome this problem, We look at two different methods.

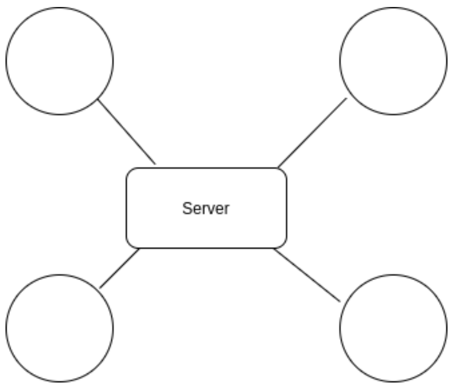

Figure 3.2: A star network of 4 peers

1. Multipoint Control Unit

   A Multipoint Control Unit (MCU) is a combination of a network gateway and media processing unit. An MCU receives the streams of all the users decrypts them, combines them into a single media stream, encrypts it, and sends this single stream to all the users. In terms of network utilization, this is the best solution as the number of streams transferred for an N node server is 2*N. But, this method also has its own caveats. The mixing of streams is an added step thereby it requires time that increases the latency. Also, the end stream received is a single stream which is a major drawback as it provides low QoE.

2. Selective Forwarding Unit

   Selective Forwarding Unit (SFU) is the most accepted solution in most of the systems today. This has a similar architecture to MCU with the central server in this case being an SFU. An SFU works similar to MCU in a way as a gateway. But, it does not mix these streams. Instead, it changes the SDP information of these streams according to the network capability of the receiver and sends it to them. This acts as an adaptive algorithm that keeps adjusting the quality of the stream as the number of streams sent to a user changes. Hence, this method provides a good scalable solution with low network usage.

   Most of the systems have tweaked this method into creating optimized solutions. One such system jitsi-meet (5) optimizes the network usage by sending the streams only from last-n users that have been sending audio. This is done by checking the SDP information for audio streams to store these stats.

Overall, star topology with selective forwarding seems the most viable solution as it reduces network usage, is scalable, and maintains good QoE. This has a major flaw though, The use of an external server violates the user privacy that we wanted. We have proposed a solution that uses the functionalities of some of these systems to design a new system.

3.3 Proposed design

We have used the idea of supernodes in building a new network topology for our video conferencing system. Supernodes in network topology is a fairly popular idea. This architecture involves the creation of a network of subtrees as shown in figure 3.3. We also use the fact that we can mutate the SDP packets to change the quality of media elements as the number of peers in the network change.

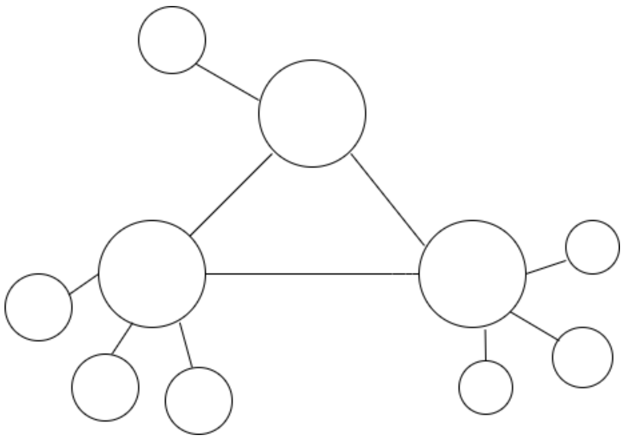

FIGURE 3.3: A supernode architecture with mesh network among the supernodes. Here the larger circle denote the supernodes.

In this method, We identify certain peers in a meeting as supernodes. The network is then converted into a supernode topology by connecting the non-supernodes to exactly one of the supernodes. All these supernodes create a mesh among themselves to complete the connection. The supernodes have the following functions-

1. Act as a gateway for all the peer nodes connected to it.
2. Combine the media streams of connected peers with it's own stream to form a single stream.

These supernodes are deriving from the functionalities of MCU to create combined streams. Since this is happening at a user's browser, it remains a decentralized network. We have not yet discussed how these supernodes will get selected. As shown in (4), Supernode selection must follow some distribution criteria as follows-

1. All the non supernodes must have a low latency access to one of the supernodes.
2. These supernodes must be evenly distributed throughout our network.
3. To meet the application specific performance requirements, some pre-specified ration of supernodes per nodes must be maintained.
4. Supernodes should not serve more than a pre-specified amount of non-supernodes.

The supernode selection problem is a widely known problem with many possible solutions as shown in (4). In our design, We simply use the signaling network to select the supernodes based on availability of bandwidth and CPU resources. We have selected only these two parameters as supernodes form a network among themselves and hence have a much higher bandwidth and CPU requirements. We have ignored latency as this system will not work for a large number of peers as depicted in our results for a mesh network.

There is one other modification at each node, They modify their SDP contents to change the media quality before sending it to a supernode. This has been derived from the SFU functionality to adjust the media quality with increasing connections. The system also needs to be fault-tolerant as there are scenarios when a supernode leaves a system. In this case, the signaling server tries to accommodate these nodes to an existing supernode whose load criteria have not been exceeded. Otherwise, a new supernode is created among the remaining nodes and all the other nodes are assigned to it.

Chapter 4

Evaluation and Results

4.1 Evaluation

We have built our web applications using JavaScript and nodejs to test the various methodologies. These tests were run on a single pc with specifications i5 processor, 8GB memory. The server was hosted on an ubuntu system with 8 GB memory, 2 core processors, and 160 gb ssd disk. We have used kurento as our MCU and Jitsi-videobrige as the SFU to test the star topologies. We have been unable to completely implement our design and it is to be done in future work.

Table 4.1 shows a comparison of the network usages when the number of members was varied for each of the methodologies.

Table 4.1: Network usage for all three methodologies at each browser in mbps

Number of participants	2	3	4	5
Mesh	2.2	5.6	-	-
MCU	2.9	2.7	3.2	2.7
SFU	2.8	5.4	7.2	6.4

4.2 Analysis

4.2.1 Mesh Topology

The network was unable to handle cases with a number of members more than three. Even for fewer members, the CPU and memory usage was considerably high. In conclusion, due to being directly connected mesh topology theoretically provides the least latency solution. But our results show that this is not scalable and thus cannot be used.

4.2.2 Star Topology with MCU

This network shows near-constant network usage which should be true theoretically speaking. As the number of streams to be uploaded and downloaded remains the same. Though, This faces the problem of less QoE and maybe not be suitable for all.

4.2.3 Star Topology with SFU

Initially, this was slightly worse than mesh topology but it continues to work even on increasing the number of members. Additionally, on increase in the number of members, its network usage does not grow exponentially as opposed to mesh topology.

Table 4.2 shows the comparison of features in these networks based on our analysis.

Table 4.2: Comparison of features

Feature	Mesh	MCU	SFU
Scalability	NO	YES	YES
QoE	NO	NO	YES
Decentralized	YES	NO	NO

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this report, we have seen the working of various WebRTC applications in the video conference domain. We showed the various topologies that can be used and their individual advantages and disadvantages. We have to select a topology based on our requirements. A peer-2-peer topology may not be feasible unless the conference size is very small.

An MCU solution can be used if only one or few attendees are presenting at a time due to the low network usage even with a large number of attendees. SFU is the best method among these due to its high scalability and QoE.

Theoretically speaking, Our design is not highly scalable it is an improvement over the existing peer-2-peer network in a bid to maximize the efficiency of a decentralized network.

5.2 Future Scope

The first step for us shall be the implementation of our design. As we will be able to better understand itss performance in a real time scenario. There are also some flaws in our design as the supernode selection algorithm is not optimal. It is highly susceptible to dynamically changing networks. We have ignored the latency parameter and this algorithm does not seem fair to the supernodes as they have a much higher load than their counterparts.

Bibliography

[1] Real-time communication for the web. (2020, November 15) Retrieved from https://webrtc.org/

[2] Kurento documentation. (2020, November 15) Retrieved from https://www.kurento.org/documentation

[3] Jitsi Meet Handbook. (2020, November 15) Retrieved from https://jitsi.github.io/handbook/docs/intro

[4] Virginia Lo, Dayi Zhou, Yuhong Liu, Chris GauthierDickey, Jun Li. ”Scalable Supernode Selection in Peer-to-Peer Overlay Networks”, IEEE 2005.

[5] Boris Grozev, Lyubomir Marinov, Varun Singh, and Emil Ivov. ”Last N: relevance-based selectivity for forwarding video in multimedia conferences”,NOSSDAV 2015.

[6] STUN, TURN, and ICE. (2020, November 15) Retrieved from https://anyconnect.com/stun-turn-ice/