IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798

# INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

## IN COMPUTER & COMMUNICATION ENGINEERING

Volume 10, Issue 5, May 2022

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.165

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 5, May 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1005088|

# Web Real-Time Communication Scalability Using Selective Forwarding Unit

**Ashutosh Dwivedi[1], Deepak Arora[2], Puneet Sharma[3]**

Department of Computer Science & Engineering, Amity University, Uttar Pradesh, Lucknow Campus, India

**ABSTRACT***:* WebRTC is a web-based open platform which allow for real-time communication in the browser. It contains network, audio, and video components needed in voice and video chat apps, as well as the essential block for high-quality communications on the web. When implemented in a browser, these components can be accessed via a JavaScript API, allowing developers to quickly create their own RTC web app. WebRTC is being standardised at the W3C on an API level and at the IETF on a protocol level. The SFU (Selective Forwarding Unit) architecture for video conferencing includes the following data transmission procedures between servers and the endpoints. All video streams are received by the server from all endpoints. The server delivers each endpoint several copies of uncompressed video streams from other participants. The endpoints combine the video streams that come in. The primary goal of this study is to observe the performance of multi-party video conferencing using the Selective Forwarding Unit (SFU), as well as how effectively WebRTC apps can be scaled using this method.

**KEYWORDS***:* WebRTC, SFU, API, Video Conferencing.

## I. INTRODUCTION

Most WebRTC apps, platforms, and services nowadays offer a ton more than the old one-to-one peer-to-peer WebRTC use case, and they all utilise at least one media server to do so. For interoperability with pre-existing technologies such as SIP for Voice over IP (VoIP), PSTN, or Flash (RTMP), that can only handle one 21 stream of a given type (audio, video) at the time, media mixing capacities are 24 required, and a Multipoint Control Unit is chosen (MCU)[1]. Most contemporary media servers, on the other side, are constructed as Selective Forwarding Units (SFU), a design that allows for enhanced bandwidth adaptability with multiple encoding (simulcast) and Scalable Video Coding (SVC) codecs while also being less CPU intensive on the server.

The latter enables significantly greater resiliency against network quality issues such as packet loss. Even when focusing just on use cases that can be implemented with an SFU, there are still a lot of others. Video conferencing (many-to-many, with everyone receiving and sending equally) and streaming / broadcasting (one-to-many, with one sending and many receiving) can be the two 13 most popular use cases.

Although there are a variety of free source SFU media servers available, mediasoup is the best option for creating multi-party video conferencing and real-time streaming programmes owing to its adaptability, performance, and scalability. Simulcast, SVC, transfer BWE, and other cutting-edge capabilities are all included.

## II. RELATED WORK

Simon Holm and Alexander Lööf published The design and architecture of a WebRTC application [2], a study that looked at current design or architectural patterns for WebRTC apps and how they might be implemented using JavaScript. The authors covered the Full mesh-Peer to Peer, MCU, and SFU design patterns, as well as the use cases and limits of these current models. In the study, a model was shown that combined the Full mess model with Selective Forwarding Unit model.

A study paper on P2P Live Video Streaming in WebRTC has been created by Florian Rhinow, Pablo Porto Veloso, Carlos Puyelo, Stephen Barrett, and Eamonn O Nuallain [3]. This research project observes the viability of using 9 WebRTC to integrate live video streaming protocols into online applications. The 14 authors also observed the constraints and potential future issues that might arise 3 when implementing complicated and scaled P2P systems.

Michele Papalini, Giovanna Carofiglio, Alberto Compagno, Angelo Mantellini, Luca Muscariello, Jacques Samain, Mauro Sardara[4]. In the research 10 work they have created ICN-RTC. ICN-RTC is a real-time communication architecture based on the newest SFU-based concept that takes advantage of Information-Centric Networking's scalability features (ICN). They also offer the ICN-RTC synchronisation protocol, which allows media distribution to utilise ICN's pull-based transport without adding additional delay. Hybrid ICN, an incrementally deployable ICN solution leveraging IP, was used to implement ICNRTC. The preliminary findings are promising: Instead, then scaling

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 5, May 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1005088|

with the overall number of users in the conference, ICN-RTC scales with the number of active speakers. When compared to regular WebRTC, this enables for ten times more participants while lowering the call's resource requirements by over a hundred times.

Zahra Jafari [5]. The development process of a WebRTC-based E-exam system is detailed in this research report. This web application is an online examination system that supports video and e-exam systems in Dari. It also uses flashcard technology to assist students prepare for the exam. This online test system may be utilised in schools, and an administrator can supervise and control the 19 actions of teachers and students. It is a safe and responsive web application that is 8 3 tailored to Dari speakers and provides a simple platform for teachers and students to utilise. PHP and MySQL are used to create the e-exam and flashcard sites, while NodeJS and WebRTC are used to create the video-exam website. Overall, this online application is a simple, safe, and powerful Dari language testing system.

## III. METHODOLOGY

The COVID-19 epidemic and the ensuing stay-at-home orders have resulted in major changes in people's working habits. One of these shifts is the rising usage of video conferencing for communication and business meetings. In December 2019, Zoom had 10 million daily meeting attendees; by April 2020, that number had increased to almost 300 million. Other video conferencing systems, such as Google Meet and Microsoft Teams, have seen considerable daily participation increases as well. Furthermore, videoconferencing is expected to continue even after the pandemic has passed, since Gartner expects that just 25% of corporate meetings will be held in person by 2024.

During the COVID-19 outbreak, WebRTC may have been the most essential collection of technologies employed. WebRTC, a wide group of technologies that allows web browsers to perform audio, video, and real-time data conversations, is used by all web-based videoconferencing services. The Internet Engineering Task Force (IETF) established WebRTC technologies.

Server, which takes media streams as input and spread the received streams to the other end users as output [6]. The first and foremost step is to exchange the necessary credentials (including MAC address) between media server and local machine of participant. Since every router and anti-virus nowadays has some kind firewall to protect the suspicious or unwanted connections and to hack the way through it, the second step is that participant should connect to a STUN or TURN server, this step is not mandatory for systems with loose security. The third step is to identify the media server with the help of signaling server [7]. Signaling is very important as it helps the end user to identify which media server to connect. After successful identification of media server, the fourth step is to connect with the media server which has all the necessary implementation to deliver the stream to all the respective participants [8]. As soon as media server starts receiving more than one streams it will start processing and distributing video codecs to each participant.

## IV. IMPLEMENTATION

In this research study, the authors have used an SFU media server to enable browser-to-browser interaction while ensuring scalability. As previously stated, the SFU is a centralized model in which user streams are processed, encoded, and gets distributed among peers as a stream. This type of distribution is preferred when the requirement is to broadcast the media or video/audio to maximum audience. Instead of normal WebRTC peer connection where the media producer has to establish connections with all the consumers in a mesh, the producer can only send one stream of media to the SFU media server that can be distributed among various the consumers. This kind of approach to distribute media stream is more scalable on relatively less computing power in SFU than in MCU.

### Mediasoup Media Server

### A. Communication Client-Server Relationship

There is no signalling system provided by mediasoup for communication between clients and servers. It is up to the programme to transmit them between clients and server via WebSocket, HTTP, or any other communication method, and to exchange mediasoup relevant parameters, requests/responses, and alerts. Because most cases demand bidirectional communication, a full-duplex channel is frequently necessary. However, the application can utilise the same channel for messages that are not connected to mediasoup. Every authenticated WebSocket (WS) connection is associated with a "peer" in the application. In mediasoup, there are no "peers" per se. The application may, however, desire to specify "peers," which may be used to identify and associate a certain user account, WebSocket connection, metadata, and a collection of mediasoup transporters, producers, and consumers. Producers and consumers of data. The

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 5, May 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1005088|

client-side programme loads its mediasoup device by using the server side of mediasoup router's Real Time Processing capabilities.

B. **Creating Transports**

For transmitting and receiving, both mediasoup-client and libmediasoupclient require distinct WebRTC transports. In most cases, the client programme builds those transports ahead of time, even before intending to send or receive media.

For media transmission:

- The mediasoup router must first construct a WebRTC transport.
- The client-side application was then cloned.
- The client application should mandatorily subscribe to the local transport's "connect" and "produce" events.

For media reception:

- The mediasoup router must first construct a WebRTC transport.
- The client-side application was then cloned.
- The client application should mandatorily subscribe to the local transport's "connect" event.
- If SCTP (also known as DataChannel in WebRTC) is wanted on such transports, enableSctp and other SCTP-related options must be enabled.

C. **Producing Media**

The client-side programme can construct several audio and video media tracks on the send transport once it is built.

- A track is obtained by the application.
- It uses the local send transport's produce mechanism.
- If this is the initial call to the generate method, the transport will emit "connect."
- The transport will emit "produce," which will cause the application to send the event parameters to the respective server and construct a Producer instance on the server.
- Finally, the produce method returns a Producer object on the client side.

D. **Consuming Media**

The client-side programme can consume numerous audio and video tracks on the receive transport once it is setup. However, the sequence is reversed (here the consumer must be created in the server first).

- The client programme informs the server of its RTP capabilities.
- The server programme should determine if a certain producer may be consumed by the remote device (this is, whether it supports the producer media codecs). It can use the canConsume technique to do this.
- The server application then invokes the consume method in the WebRTC transport that the client constructed for receiving media, resulting in the creation of a server-side Consumer.
- It is strongly suggested that the serve-side consumer be started with paused: true and then resumed after the remote endpoint has been formed.
- The consumer information and parameters are sent from the server application to the distant client application, which executes the consume method in the local receive transport.
- If this is the initial call to the consume function, the transport will emit "connect."
- Finally, the consume function returns a Consumer object on the client side.

## V. RESULTS AND DISCUSSIONS

KITE is used to run the tests. As a result, they're built in Java and use Selenium WebDriver to launch and manage client browsers. If the page can be loaded, this value is true. Sender video check: true if the sender's video is visible and not a still or blank picture. True if all the videos received from the SFU passed the video check for all 6 clients.

| | SFU | Page loaded | % | Sender video check | % | All video check | % |
|---|---|---|---|---|---|---|---|
| | Jitsi | 280 | 100% | 280 | 100% | 53 | 19% |
| | Janus | 448 | 100% | 448 | 100% | 256 | 57% |
| PASS | Medooze | 481 | 100% | 481 | 100% | 434 | 90% |
| | Kurento | 118 | 100% | 118 | 100% | 69 | 58% |
| | Mediasoup | 488 | 100% | 488 | 100% | 476 | 97% |
| | Jitsi | 0 | 0% | 0 | 0% | 227 | 81% |
| | Janus | 2 | 0% | 0 | 0% | 195 | 43% |
| FAIL | Medooze | 0 | 0% | 0 | 0% | 47 | 10% |
| | Kurento | 0 | 0% | 0 | 0% | 49 | 42% |
| | Mediasoup | 2 | 0% | 0 | 0% | 15 | 3% |

TABLE I: Test result

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 5, May 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1005088|

### *A.Quantitative Result*

Table I shows the percentage of SUCCESS and FAILURES during the testing (a success indicates that the correct video is presented and blank picture). There were no issues with the sender video being shown. However, some of the movies received from the six clients in a room have failed. Jitsi has a great failure rate of 81 percent since still pictures are frequently presented other than of video (the measured rate of bits are zero). Many end users or browsers are missing one or more films, resulting in a high rate of failure of 43 percent for Janus and 42 percent for Kurento.

### *B. Video Quality Assessment*

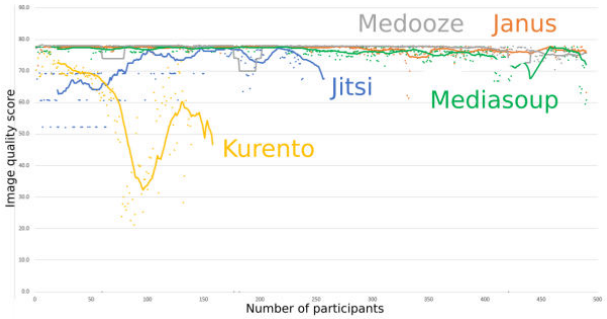


Figure 1: Video Quality Scores

Figure 1 shows the estimation of video quality scores. Although one may anticipate video quality to worsen as the average bit rate recorded decreases, the video quality graphs remain amazingly flat until the end of the test. The capacity of contemporary 20 video codecs to make a video weigh 15 to 30 times less than the original 12 uncompressed video while maintaining a perceived high quality explains this counterintuitive conclusion. Before transmitting the video to the SFU, each Chrome browser encoded it with VP8 for our test. After a few ffmpeg trials, it is discovered that after the bit rate is set to around 150 kbps or below, the natural quality of the video we utilised for this load test degrades 4 noticeably. Medooze has a bit rate of roughly 215 kbps at goal load. This is still sufficient to transmit the selected video in a high-quality manner. The video quality of Kurento changes inexplicably depending on the load. It quickly degrades, reaching its lowest image quality at around 100 participants. Surprisingly, when more people sign up for the test, the image quality increases, reaching around 130 people before decreasing again.

## VI. CONCLUSION

Because of Mediasoup, it is now feasible to scale WebRTC by utilising SFUs. The primary emphasis of this research was placed on the scalability approach developed at SFU. Additional metrics and tests on the client side, such as the audio quality, are 11 currently being developed and will be carried out on all of the supported browsers 23 in order to investigate the influence that browser-specific WebRTC implementations have on the findings. The addition of CPU, RAM, and bandwidth estimation probes to the server will assist us in determining the extent to which the server is able to accommodate increased user demand. In order to simulate a wider range of video conferencing use cases, we plan to make the number of rooms as well as the number of users that can be accommodated in each room into a test run variable. In order to make this effort more comprehensive, we would like to include broadcasting and streaming as part of it.

## REFERENCES

[1] Emmanuel André, Nicolas Le Breton, Augustin Lemesle, Ludovic Roux, Alexandre Gouaillard, "Comparative Study of WebRTC Open Source SFUs for Video Conferencing", 2016, 2018 Principles, Systems and Applications of IP Telecommunications (IPTComm)

[2] Simon Holm and Alexander Lööf, “The Design and Architecture of a WebRTC Application”, Independent thesis Basic level (Bachelor’s Degree), Malmö University, 2019

[3] Florian Rhinow, Pablo Porto Veloso, Carlos Puyelo, Stephen Barrett, and Eamonn O Nuallain, “P2P live video streaming in WebRTC”, School of Computer Science and Statistics, Trinity College Dublin, 2014

[4] Michele Papalini, Giovanna Carofiglio, Alberto Compagno, Angelo Mantellini, Luca Muscariello, Jacques Samain, Mauro SardaraOn the Scalability of WebRTC with Information-Centric Networking, 2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 5, May 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1005088|

[5] Zahra Jafari WebRTC-based E-exam System, Submitted to American University of Central Asia, 2021.
[6] Gonca Bakar, Riza Arda Kirmizioglu, A. Murat Tekalp, "Motion-Based Rate Adaptation in WebRTC Videoconferencing Using Scalable Video Coding", 2015, IEEE Transactions on Multimedia (Volume: 21, Issue: 2, Feb. 2019)
[7] M. Westerlund and S. Wenger, RFC 7667: RTP Topologies, IETF, Nov. 2015. [Online]. Available: https://datatracker.ietf.org/doc/rfc7667/
[8] B. Grozev, L. Marinov, L. Marinov, and E. Ivov, "Last N: relevancebased selectivity for forwarding video in multimedia conferences," in Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video, 2015.

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor: 8.165

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

# INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

## IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details