IJARSCT

ISSN (Online) 2581-9429

Impact Factor: 6.252

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

Volume 2, Issue 6, May 2022

Random Peer to Peer Chatting Application using WebRTC

Prof. Rushali Patil[1], Satish Kumar Patra[2], Krishnanunni R S[3], P Deepak Raj[4], Mahima Yadav[5]

Department of Computer Engineering

Army Institute of Technology Pune, India

Abstract: *Because of its ease for real-time communication and simple features, messaging has become an inextricable aspect of our lives. Because of their quickness and potential, messaging or chat services have become an indisputable part of how people communicate in today's world. By exchanging messages in real time, the chat program makes it simple to communicate with individuals all over the world. People are increasingly using immersive experience to connect with one another. Chat apps are rising in popularity because they can virtually retain the feeling of real-time interaction, from group messaging in live chat to e-learning and team collaboration through chat rooms to file sharing between coworkers. However, when it comes to deciding how to create the app, the consumer experience is critical. Our goal is to create a chat software with real-time messaging services that provide users with a genuine and highly interactive engagement. Our goal is to create an application that is dependable, safe, and runs in real time, without concern for user volatility or concurrent constraints. Make sure the chat application has the right security safeguards when working with sensitive material like regulatory requirements and confidential user information. As a solution to the above-mentioned requirements, we propose a random peer-to-peer chatting application using technologies like WebRTC and PeerJS. The recent improvements in WebRTC show how efficient this implementation could become. Here in this project, We propose PeerJS, which is a wrapper in JavaScript to use WebRTC without considering connection drops as everything is handled properly. At the same time, the signaling needed by WebRTC is provided by PeerJS although a custom signaling server can be created from scratch in a matter of minutes*

Keywords: WebRTC

I. INTRODUCTION

Currently, 59% of the world population has the availability of the Internet. And alone in India, this percentage stands at 50% in 2020. And with the present growth, the Internet penetration of India will be more than 80% by 2030. This number will be astounding if we look into the numbers of India's Population. Hence, many sectors in IT are already witnessing a record-breaking number of Unicorns in 2021. Hence the importance of socialization will outweigh the other services as usual in modern-day Internet usage. People are now being concerned of privacy issues regarding many chat applications, so peer-to-peer chat application is obviously will gain mainstream support in the upcoming days.

The proposed system is a chrome browser-based system that allows users to instantly share their live messages. The proposed system will be able to provide peer-to-peer real-time communication over the internet browser using WebRTC technology. It doesn't require downloading any plug-in or client program.

Messaging has become an inevitable part of our life due to its convenience for real-time communication and easy-to-use functionality. The aim of the project is to create a random chatting application, which ensures privacy along with stable and real-time connectivity. In this era of computers, people like to know each other while staying in their comfort zone and private space. This project provides such an interaction. They can interact in real-time with random people all over the world.

In the report various techniques for developing a Peer-To-Peer chat system with real time connectivity is studied and an analysis is performed regarding difficulties and issues that can arise while developing such a system. For this report a Peer-To-Peer chat system application with real time connectivity was developed, and the design of the same is described in the upcoming sections. The design of our application employs one of several different possible Peer-To-

Peer solutions and a few of the existing solutions for Peer-To-Peer communication. When one thinks of a chatting application one will presumably think of one of the more common chat systems like WhatsApp Messenger, Skype or Google Talk. All of these applications need the user to create an account and sign in to be able to use their service. Due to this reason a lot of people use different unsafe chat systems to talk with other people.

According to a study conducted by the mobile app analytics firm Sensor Tower, WhatsApp competitor Signal witnessed 17.8 million downloads between January 5 and January 12, up from just 285,000 the previous week. Similarly messenger app Telegram saw 15.7 million downloads during the same period, over twice the 7.6 million downloads it saw the in the previous week. So, it is very likely to witness peer-to-peer technology replacing these server-based services as peer-to-peer is itself serverless and privacy is high by its nature.

II. RELATED WORK

Multi-party WebRTC Services using Delay and Bandwidth Aware SDN-Assisted IP Multicasting of Scalable Video over 5G Networks :: Arda Kirmizioglu and A. Murat Tekalp:"[4]

Multi-party WebRTC video conferencing between peers and endpoints with heavily diverse network resources currently works best over the Internet with a central Selective Forwarding Unit (SFU), where each peer transmits a scalably encoded video stream to the SFU.This connection model avoids download bandwidth bottlenecks associated with mesh connections. However, this increases peer latency and overall network load (resource consumption) and requires an investment in the server since all video traffic must be routed through the SFU server. To this end, we have proposed a new multi-part 4444-WebRTC service model for future 5G networks, in which Video Service Providers (VSPs) collaborate with Network Service Providers (NSPs) to provide NSP-based services for scalable video-level streaming. (P2P multicast with IP 4444 SDN uses an NSP infrastructure. In the proposed service model, each peer sends a scalable upstream encoded video to the SDN switch, which is selectively replicated and distributed as a layer stream. The multicast tree is managed by a central non-SFU server that manages the network in multipoint WebRTC sessions controlled by the SDN controller.

"WebRTC role in real-time communication and video conferencing:: George Suciu R&D Department BEIA Consult International Marian Ceaparu R&D Department BEIA Consult International"[5]

The paper presents the related work, describing the system architecture of the WebRTC and the main protocols used. It provides TURN server functionality using the Interactive Connection Establishment (ICE) protocol for data transfer. It also shows the experimental results of a video conferencing application developed using the push messaging service Scaledrone and the Mozilla Firefox browser.

"WebRTC Architecture Assisted by Software Defined Networks:: Yugiita Kheibari Uluslararası Bilgisayar Enstitüsü Ege Üniversitesi"[2]

With the benefits it offers, software-defined networking (SDN) has the potential to move beyond today's network technology restrictions and improve the performance of network applications. This study describes the WebRTC architecture that utilizes SDN and the potential impact of SDN technology on WebRTC conferencing performance system are produced. To evaluate the performance of the proposed architecture, competitive tests were conducted in a variety of topologies, including topologies that are commonly utilised in the real world.

III. PROPOSED SOLUTION

Aim of the project is to create a random chatting application, which ensures privacy along with stable and real-time connectivity. In this era of computers, people like to know each other while staying in their comfort zone and private space. This project provides such an interaction. They can interact in real-time with random people all over the world. With the chat application, you can easily communicate all around the world by exchanging messages in real time. The chat app allows users to get the same immersive and vibrant interactions through messaging abilities, just as if they were directly.

Simply put, WebRTC is a set of rules that allow internet browsers and mobile apps to communicate with each other. It refers to an open source project and is known as Web RealTime Communication, which allows for audio, video and data transfer. WebRTC is a straightforward but sophisticated technology. The virtue of simplicity is the ease with which it may be implemented. There is always a "but" when working with technology at the same time. The biggest issue with WebRTC is the backend. Developers must test the solution on different networks.

The main focus of WebRTC is to enable real-time media communication for e.g. audio and video communication between users using a web browser to initiate conversations, find each other, and bypass firewalls. WebRTC uses JavaScript, HTML5 APIs built into the browser. Common features of WebRTC applications include:

1. Send and receive streaming audio and video.
2. Use WebRTC API to get network configuration information such as IP addresses, application ports, firewalls, and network address translators (NATs) used to exchange data from other clients.
3. Open/close connections and report errors.
4. Exchange media information, e.g., image resolution and video codecs

IV. IMPLEMENTATION

This section discusses the implementation details of our proposed solution in detail.

WebRTC: WebRTC which abbreviates for Web Real Time Chat, is a comprehensive module that offers web browser support for Real Time Chatting. In order to search the best suited module or protocol for real time communication, we compared two best candidates; Web Socket and WebRTC. A total of 55,805 websites (6.3%) were found using web sockets. This is, which is much more than the 2018 study [1], with only 1.6-2.5% of sites using web sockets. Websites with very high rankings are slightly more likely to use websockets (7.3%). of the first 1000 websites, the difference is not significant. This increase is self-evident from the growth of Internet penetration and modern technology embracement all over the world. A connection established in Web Socket has to go over a server. But our proposed solution needed a server less communication. Hence we used WebRTC for communication process.

PeerJS: At first, we started developing an environment of two devices communicating with each other using WebRTC. Although we were successful, but connection couldn't be stretched for more than 5 minutes. It was partly because of the unstable architecture of Internet (tons of protocols ensuring the stability of Internet). One simple reason could be Dynamic IP allocation. Every Internet Service Provider provides IP that can change over the time to low-bandwidth users and as maximum user base is in low-bandwidth phase, it was better to search for alternative or creating one to stabilize WebRTC connection. Fortunately, there is a beautiful library targeted to solve this problem. PeerJS maintained by Eric Zhang and Michelle Bu was nicely suited for our approach. It is a JavaScript library that handles the WebRTC API support on browser and enhances the stability of connection.

Signaling Server: Before starting the connection, It is necessary for the two connecting devices to exchange the device information for e.g. Codec, Camera, Browser, Network Status and Support. WebRTC has a distinct feature than Web Socket that it maintains the connection without a server. WebRTC selects the connection configuration based on device information. So, to initiate a connection, device information has to be exchanged between two devices. A protocol named SDP (Software Description Protocol) is used for this case. And a specialized server that enables this exchange of information through SDP is known as Signaling server.

V. SECURITY ANALYSIS

This section discusses the security component. Security has been taken care with utmost responsibility. Here a deep analysis will be taken into account.

- **Server-Less:** The architecture is server-less and promises no eavesdropping from any party whether is it is service provider or any third party. With the growing challenges of information eavesdropping and awareness of public, it promises no information snooping.
- **Confidentiality:** The architecture is server-less and promises no eavesdropping from any party whether is it is service provider or any third party. With the growing challenges of information eavesdropping and awareness of public, it promises no information snooping. Confidentiality is taken of utmost care with User Identity storing them by encrypting them. Data is stored in Distributed Storage nullifying single point of error.
- **Data Integrity:** The architecture is server-less and promises no eavesdropping from any party whether is it is service provider or any third party. With the growing challenges of information eavesdropping and awareness of public, it promises no information snooping.
- **Privacy:** The whole communication is encrypted with AES encryption while key is transferred through RSA encryption. So privacy is considered to be top-notch and security becomes ultimately strong.

- **Cyber Safety:** Cyber safety becomes a concern when both users are strangers. It can allow the ill-elements to take advantage of the situation and generate unwanted cyber safety issues. So after every conversation we have added a feedback session for every user. Based on the score, we ban the user or take cautious steps/

VI. FUTURE SCOPE

With the advent of streaming, online schooling, online gaming streams, use of peer to peer will increase. From the chatting perspective, peer to peer chatting will gain mainstream as we see many controversies revolving around Meta, WhatsApp privacy concerns. Messaging has become a part of our daily life, in part due to the convenience and simplicity of real-time communication. Aim of the project is to create a random chatting application, which ensures privacy along with stable and real-time connectivity. In this era of computers, people like to know each other while staying in their comfort zone and private space. This project provides such an interaction. They can interact in real-time with random people all over the world. Chat apps make it easy to connect with people around the world by sending and receiving messages in real time. Chat apps allow users to have the same engaging and vibrant interactions as face-to-face with customizable messaging features.

VII. CONCLUSION

We discussed what the WebRTC project is and introduced key concepts. Then we created a simple application to exchange data between two HTML clients. We also discussed the steps involved in creating and setting up a WebRTC connection. It also covered using STUN and TURN servers as fallback mechanisms in case of WebRTC failure. We also had reviewed some of the works done in the field and studied some journals and made the literature review. PeerJS acts as a wrapper around WebRTC, simplifying the whole process and providing much simpler events and methods to work with.

ACKNOWLEDGMENT

It brings us great pleasure to share the "Random Peer To Peer Chatting Application" paper. We'd like to take this time to express our gratitude to Prof. Rushali Patil, our Project Guide, for providing us with all of the assistance and direction we required. We owe her a huge debt of gratitude for her unwavering support. Her wise advice was quite beneficial. Prof. Dr.S.R.Dhore, Head of Computer Engineering Department, Army Institute of Technology, is also acknowledged for his invaluable assistance and ideas.

REFERENCES

[1]. Shi Yuzhuo1 , Hao Kun2 1 Tianjin College of Commerce, Tianjin, China, 2011 IEEE. Design and Realization of Chatting Tool Based on Web

[2]. Bita Kheibari Uluslararası Bilgisayar Enstitüsü Ege Üniversitesi 2020 IEEE. A WebRTC Architecture Assisted by Software Defined Networks

[3]. Alexandre Gouaillard, Ludovic Roux CoSMo Software Consulting, Singapore, 2019 IEEE. Real-Time Communication Testing Evolution with WebRTC 1.0

[4]. R. Arda Kirmizioglu and A. Murat Tekalp Multi-party WebRTC Services using Delay and Bandwidth Aware SDN Assisted IP Multicasting of Scalable Video over 5G Networks .Web Application for Social Networking using RTC GeorgeSuciu R&D Department BEIA Consult International Marian Ceaparu R&D Department

[5]. BEIA Consult International, WebRTC role in real-time communication and video conferencing

[6]. Samuel Micka, Utkarsh Goel, Hanlu Ye,† Mike P. Wittie, and Brendan Mumey Department of Computer Science, Montana State University, Bozeman, MT 59717

[7]. Roberto Beraldi, Gabriele Proietti Mattia DIAG, La Sapienza University of Rome, Italy. Power of random choices made efficient for fog computing

[8]. Shi Yuzhuo1 , Hao Kun2 1 Tianjin College of Commerce, Tianjin, China. Design and Realization of Chatting Tool Based on Web

[9]. Sunghyun Yoon, Taeheum Na, and Ho-Yong Ryu Smart Network Research Department, Electronics and Telecommunications Research Institute, Daejeon, Korea. An Implementation of Web-RTC based Audio/Video Conferencing System on Virtualized Cloud

[10]. Alexandru C. 2014. Impact of WebRTC (P2P in the Browser). inB. Stiller et al. (Eds.). Internet Economics VIII. Technical Report. Department of Informatics, University of Zurich.

[11]. Vanessa W., Salim F. and Moskovits P. 2013. The definitive guide to HTML5 WebSocket. New York: Apress.

[12]. Beizer B. 1995. Black-box testing: techniques for functional testing of software and systems. Canada: John Wiley and Son.