WebRTC: Peer-To-Peer Architecture for Real-Time Communication

Naveen Shivnani [1], Prof. Harish Chandra Maurya [2]

[1] *Research Scholar, Computer Science and Engineering, Bhagwant University, Ajmer, Rajasthan, India*
[2] *Assistant Professor, Computer Science and Engineering, Bhagwant University, Ajmer, Rajasthan, India*

ABSTRACT

Communication and information sharing has always kept us ahead and with technology the communication has been much easier as well as exchange of data/information. Existing communication channels have been improving with new technologies. Current real-time communication applications come with several challenges such as requirement of additional software, downloads and plugins and security in order to establish real-time communication. Web Real Time Communication (WebRTC) can be used to overcome these hurdles,the aim of this study is to design and develop a real time communication application solution with peer-to-peer system architecture, during this study multiple technologies were integrated with WebRTC to develop the system architecture. Quantitative and Qualitative data were collected during the process. The application/solution overcame the hurdles associated with plugins, downloads and security and minimize latency and bandwidth usage and also established real time peer-to-peer video and audio communication. The results show that peer-to-peer connection was established and high quality video and audio was recorded. The system of 20 virtual users (VU) recorded a TPS (Transaction processing Time) of 308ms and average response time of 196ms for 1 min uptime of load test. WebRTC provides high quality service and can be an alternative for delivering real-time peer-to-peer interaction.

Keyword: - *Web Real-Time Communication (WebRTC); Peer-To-Peer (p2p); MediaStream; RTCPeerConnection; RTCDataConnection*

I. INTRODUCTION

The most basic and major challenge with web has been the incapability for two web browser to communicate and to share data without the support of any third party software or plugin such as flash, applets etc. [1] As a result, there has been many progressive attempt by IETF (Internet Engineering Task Force) to define networking protocols. The W3C (Worldwide Web Consortium) harnessed JavaScript API to bring the realities of new communication experience to webservers and internet users [2]. This led to the technology popularly known as WebRTC (Web Reat-time communication). Its main objective is to establish a peer-to-peer (p2p) communication within the dependable ecosystem. Prior to the introduction of a peer-to-peer protocol, a signaling process was implemented using various technologies such as Http Request, Session initiation protocol (SIP), Extensible Messaging and Presence Protocol and WebSocket. Google in association with Internet Engineering Task Force (IETF), WorldWide Web Consortium (W3C), and Mozilla are working on this open source project [3]. The project is open source in nature to ensure developers are able to design and build various systems/applications that allow internet users to communicate by video conferencing, text chat in real time inside the browser itself [4, 5], or can be used with PSTN (Public Switch Telephone Network) or VOIP (Voice Over Internet Protocol) [6]. Websocket is very similar to WebRTC the major difference is that, websocket open a connection with server rather than the peer. For example in a chat application, Websocket first the message is sent to the server from the client and then server sends the message to the recipients. WebRTC ensures that the p2p communications between the users are secure and plugin free [7]. WebRTC ensures a simple, flexible and cost effective means of real-time communication for its users without the need to rely on service providers. A crucial issue with plugins such as shockwave, flash etc is to download them whenever connection has to be established which can cause increase in bandwidth, execution time, latency and speed [8]. This application of video based communication with no plugins will reduce the problems that are caused with them as well as establish coordination in communication. The developed WebRTC system architecture establishes a p2p end-to-end communication with audio-video content and direct data transmission. This application is developed in order to omit intermediate server and avoid security issues such as data being hacked by hackers or call being tracked by hackers. These features make WebRTC unique than any other Real Time Communication Systems such as Zoom. Though Zoom is a good audio-video communication service, but it lacks p2p ability, as well as the security that is provided by WebRTC. P2P (Peer-to-Peer) also ensures the data to be encrypted [11], secure and cannot be compromised. P2P can be secure because it doesn't require any plugin and if there are no plugins, there is no problems associated with them in WebRTC. Factors like bandwidth, memory utilization and latency as well as anonymity of the internet user/client all are supported by WebRTC. In order to carry out the architecture of system designed in this study, a p2p video calling application was designed and evaluated. The application has been deployed on github pages, so as to make sure it is available for real world application and is accessible by everyone. WebRTC technology provided peer-to-peer real-time communication between users and delivers great potentials [12]. WebRTC is neither a service nor an application; it is a technology which does not require support for any of its components like codec or media engine. WebRTC is being adopted by large firms and the speed of adoption is pretty high.

2. LITERATURE REVIEW

Researchers define WebRTC as "Zoom-like" technology, but zoom has been well before WebRTC and both of them are completely separate from each other. The existence and dominance of zoom in RTC (real-time communication) industry, with billions of user's worldwide, zoom still contains few drawbacks. It is exclusive and users must install the package and supporting software's or plugins such as flash to make sure it works properly. The security also gets affected with plugin used by zoom. Zoom is also resource intensive because of its client server architecture [13]. WebRTC uses peer topology which is less overhead to the system and it is an open source/ open-protocol descendant of media engine [14]. This means real-time capabilities, existed in flash plugin have been natively made available in browsers that are WebRTC-compatible so that developers can easily develop various real-time solutions [3, 15].

2.1 Issues in communication technology and solutions

Internet communication has always been dependent on service providers before the existence WebRTC, vendors such as PSTN (Public Switch Telephone Network) had used more complex communication processes. The clients of PSTN must participate separately as members of PSTN IP based community. This IP based membership is maintained by service providers who delivers very basic/simple necessities. There should also be an affirmation that every telephone number must be unique for every physical location i.e. for every portable mobile device there must be unique user or phone number. This sort of communication may require users to buy or subscribe product, moreover require users to download plugins before contents or calls work properly. For example WebEX and Zoom, these require installation of third party applications, hence limiting the diversification of communication and scalability, with WebRTC every website of all kinds is substantially its own "Service Provider", without any need or relationship with any party outside of itself or the establishment of communication by its users.

The biggest concern with installation of third party software or plugins is lack of trust, it is because malicious software or malware can be introduced by using these technologies, and can cause technical problems/issues with the software. WebRTC technology can overcome these problems which are associated with plugin installation. Apart from this there is another problem i.e. standardized media engine which can be accessed freely through a simple HTTPS (Hyper Text Transfer Protocol). Additionally, potential vulnerabilities or security flaws in all browsers are programmed to get automatic updates when they are discovered and many times these flaws are updated in the newer version. So, a browser implementation of WebRTC can be fixed very quickly when compared to that with traditional software/application like VoIP (Voice over Internet Protocol) application. The complicated Voice over Internet Protocol faces similar problems caused by malicious software by getting patches to solve the security flaws. This may take much more time to get resolved. Browsers play an important role because of their universal nature and speed of data that is accessed [16].

Zoom/WebEX use a DNS (Domain Name Server), which can cause decoding of keys of media content that is being transferred through their services to be intercepted, whereas WebRTC enables authentication and encryption of video, voice and data by default and maintain the anonymity of its users. This is possible through the DTLS (Datagram Transport Layer Security) and SRTP (Secure Real-Time Protocol). This is used to avoid recording and eavesdropping of video and voice data over WiFi networks. WebRTC do not hold up support for any other collaboration or communications protocols. A major consideration for establishment of reliable session is by using NAT (Network Address Translators). It is important as it reduces delay in response from server and drastically avoid server latency, load and intensify quality. This is a good feature as it allows developing customized applications that can allow other vendors to begin communication sessions with applications that uses WebRTC. This is a unique accomplishment is realized with WebRTC which is not the same with Zoom. Applications that are developed using WebRTC allow users to participate in any interaction form any site without registration or pay a handsome amount so as to join the interaction as experienced with Google or LinkedIn for association.

2.2 WebRTC API Implementation

WebRTC impact can be seen from different edges. WebRTC can be seen in terms of Codec, Stack, Protocol and Signaling [17]. WebRTC can also be seen in terms of API [18]. It is majorly dependent of three different implementation of APIs which help in real-time communication/transmission of data in any web application [3]. These three APIs include RTCPeerConnection, RTCDataChannel and media engine. Media Engine –This enables the browsers to getUserMedia i.e. browsers are able to access user media like camera and microphone. This API is natively available in HTML5 which can be used to directly access hardware. getUserMedia bypass the need of using external codec for capturing video or audio data.

RTCPeerConnection- through RTCPeerConnection actual WebRTC connection is possible, while WebRTC handles the streaming of data in an efficient manner between two or more peers. So, for initiation of a connection with a remote party the caller browser must initiate an RTCPeerConnection Object. RTCPeerConnection API transfers the data is real-time and is responsible to manage the life-cycle of every peer-to-peer connection, encrypt the management and connection setup, and the state of the object in a single user-friendly interface [19].

RTCDataChannel- through RTCDataChannel API of WebRTC arbitrary exchange of data between peers is possible. RTCDataChannel can be viewed as the already known WebSocket, but with customization capability of transfer protocols. It is really helpful in much application such as text chat application, file sharing and game applications.

2.3 Internal WebRTC Standards and Architecture

WebRTC internal architecture contains web API for developers. It also has a platform where developers can handle issues related with rendering and capturing hooks [20]. Layers of WebRTC work on different platforms and across browsers. The web APIs layers consist of an RTCDataChannel, RTCPeerConnection and MediaSteam Object for its developers. WebRTC implementation is easy due to its native C++ API for different browsers. The calls can be setup with ease by developers because it consist of signaling management and session management modules that takes care of session establishment and signaling. WebRTC can also handle application of different transport mechanisms. The internal architecture also consist of a Video Engine or Video Engine collectively known as the media engine.

The Voice Engine framework transfers audio from sound card to the network. Examples of voice engine include iSAC, iLBC and OPUS. iLBC and iSAC were products of Global IP solutions but in 2011 became a part of WebRTC. All these codec basically manage the audio streams. These voice engines provides facilities to keep bandwidth and voice latency at a low level, while keeping the quality high. These codec includes error concealment algorithm and dynamic jitter buffer used to conceal the negative effects of packet losses and network. It is helpful in handling the negative effects of noise reduction, encryption, compression, and echo cancellation.

Video Engine framework has bi-directional moment of video i.e. from network to user screen and camera to the network. In has additional features like video image enhancement, video processing and camera image processing. It also provides dynamic jitter buffer to increase the quality of the video and conceal any packet loss and also manages the bandwidth. The video codec includes H.264 and VP8. Fig. 1 shows the internal WebRTC video and voice codecs.

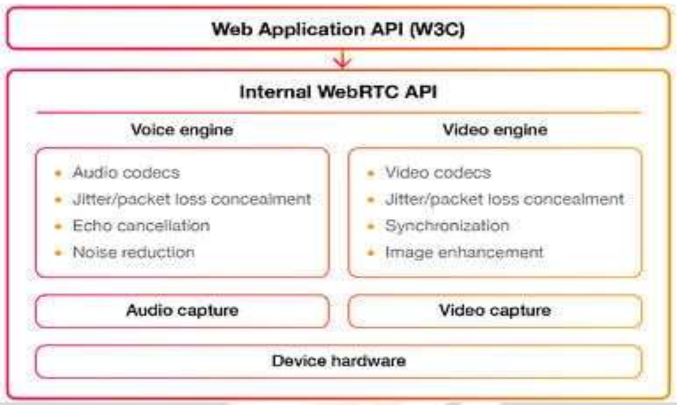


Fig -1: Built in audio and video WebRTC engines

3. RESEARCH METHOD

The research was conducted in an iterative manner. First Step included identifying the required technologies including WebRTC to design the system architecture. At this stage, a video chat application was considered for implementation of WebRTC, other technologies and real-time communication. Potential clients/users were also categorized and were collected. The next step included the designing system architecture of real-time communication system that uses WebRTC. The following step included development and testing of the application. The development process of the application was iterative. Application evaluation was also done by the potential users/peers. The final step concluded the perform analysis of the test results to evaluate that the system was able to carry out Real-time interactions.

3.1 Required Technology Survey

Design of the Real-Time Application was kept simple, reliable and testable. The study was to provide a solution using WebRTC to deliver a real-time interaction to the peers/users. Other technologies identified suitable to the requirements of the system includes HTML, CSS, JavaScript and PeerJS technology. The frontend is kept simple and user-friendly using HTML and CSS, the backend rely on PeerJS and WebRTC inbuilt functionalities and to make it available to everyone the application is hosted on github pages to make it accessible to everyone.

3.2 System Architecture Design

The designing stage is a prominent stage that provides a documentation, methods and process that were applied in this research. The design research of this study followed planning, design, coding, testing and implantation phase. The WebRTC communication platform was also checked. In the planning phase, identification of specification for a simple web-based real-time application and gathering basic requirement of the application to run. The system design and coding of components was done by creating very simple MVC architecture for the application, which involved writing logic and programs critical in creation of a quality application.

3.3 System Architecture

This section of the research explains the application of peer-to-peer technology. The p2p WebRTC architecture mentioned in the above section was left over specifically for this section as it is a part of implementation of this study.The general architecture of the application adapts the p2p (peer-to-peer) architecture. Two peers will only be able to communicate after the signaling process gets completed. In Fig.2, WebRTC has been designed to sit inside the browser to make sure that media/data gets communicated in peer-to-peer fashion without any plugins. The ability of communication between two peers must start with an attempt to initiate and aid familiarity between the callers. This can be explained with the process of offer and answer.

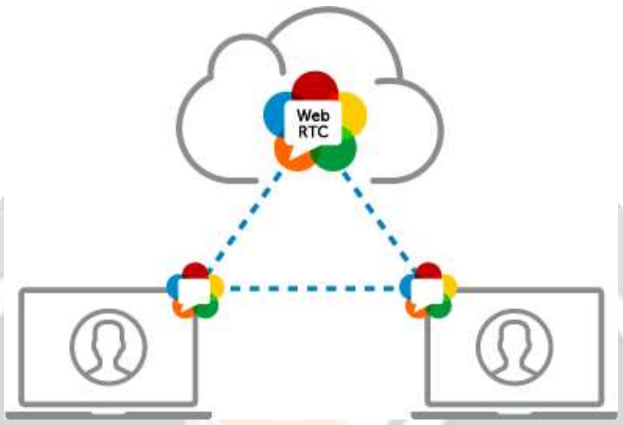


Fig -2: P2P Architecture

Standard infrastructure required to setup the above WebRTC architecture includes 1) the peer's browsers, an HTTP signaling written in PeerJS. It is required to exchange data between the peers,the stun server is required to find an excellent path for the media to get relayed. There are many more infrastructure that can be used but are beyond the aim of this study including SFU or MCU. These are used by large multi-peer video and audio conferencing, recordings and other gateways.

3.4 The Offer, Answer, Connection and Handshake Process

When the process gets started of signaling an offer is created by the first client/peer who has initiated the call, and an encrypted key get generated, the offer has a session description which gets encrypted by the alpha-numeric key, this key needs to be delivered to the second user/peer who will receive the call. The second user/peer replies with a message as answer, which contains session description of the other peer. This exchange of keys convey that both the peers know certain information regarding each other which can be used for the call to get connected example of these are transport protocol, ports, video parameter and codec information. Interactive Connectivity Establishment (ICE), figures out the best possible path to establish a connection between the two peers based on the information that was gathered earlier for example, through wired connection or wireless network interface. The handshake model ensures that exchange of networking information happens between the participating peers.

In order to make a connection with another peer, its web location should be known. This step is a logical process where, first an RTCPeerConnection object is created by both the peers and a session description is obtained, which ensure what kind of data peers want to exchange with other peer through a p2p connection. This is done by calling the inbuilt method of RTCPeerConnection Object. The caller initiates the video call and obtains the encrypted key which stores the session description and makes an offer for another peer, this key is transferred to another peer who can simply enter the key and makes a connection through remote description and sends an answer back to the first peer through the signaling channel.

3.5 Application Development and Testing

The technologies and design is kept as simple, portable and lightweight as possible, in order to make it user friendly and easily understandable to the general public. The front-end is based on HTML, CSS and Javascript to make sure that it is light weight and has room for improvement, in the back-end PeerJS has been used which uses WebRTC and its web APIs to make connections and sessions between the peers and enables them to exchange data with each other. PeerJS uses the following API RTCPeerConnection to establish a connection between the peers, RTCDataChannels which enables them to exchange data with each other in a peer-to-peer fashion and MediaStream Object which ensure that data is streamed in a secure and encrypted manner.

The application is currently hosted on github pages, which makes it easily available for general purpose use and is open-source in nature to ensure that any developer can contribute to the application and apply his/her creativity and take the advantage of peer-to-peer architecture for data sharing, it also helps in building, running, storing, scaling and better management of the application by using tools like gitbash. It also helps in reducing the cost, better access for general audience and developer infrastructure for quality of service. We studied the benefits allowing the developer to focus solely on developing the application rather than focusing on hosting infrastructure capabilities. Scaling is an added benefit which allow developers to scale up the application quickly through passing commands from CLI. The github pages infrastructure has reduced the cost and time effectively. It also support different platforms and incompatibility issues can be resolved easily.

3.6 Testing of Application

To determine practical usage of the application on LAN (Local Area Network), a test was conducted with several users (peers) within the university. After that, LoadFocus “a tool to generate virtual users and perform load testing and performance matrix” was used to test the application prototype for its performance. WebRTC Internals was also used to test the real live test performance of the application. Many parameters such as Virtual Users, Response Time, Hits, Bandwidth, Latency, Errors, Transaction processing time (TPS), memory usage were captured and measured. Many of these tests are reported and graphically displayed in the result section. The starting configuration were as follows: Ramp-up time: 30s, peers/users: 20, duration: 60s, iteration: testing continues. These simple setups are targeted towards meeting the performance goals.

4. RESULTS AND ANALYSIS

Fig. 3 represents the video calling application user interface. The UI shows a video conferencing between two peers. The peers are in real-time interaction. The connection is directly between the user browsers and avoids all convectional domain name server connection between peers. The peers do not require any kind of third party plugin or download any software or application such as flash for the video to be seen on both peer browsers. The getUserMedia() method establishes the connection and can access microphones an cameras of the peers. WebRTC requests the peers for permission to use media devices. The peers can either “allow” or “block” the request. Once the “allow” option has been clicked that means that the system can now access the peers microphone and camera for real-time interaction. The web application was tested and ran on chrome, edge, firefox browsers.

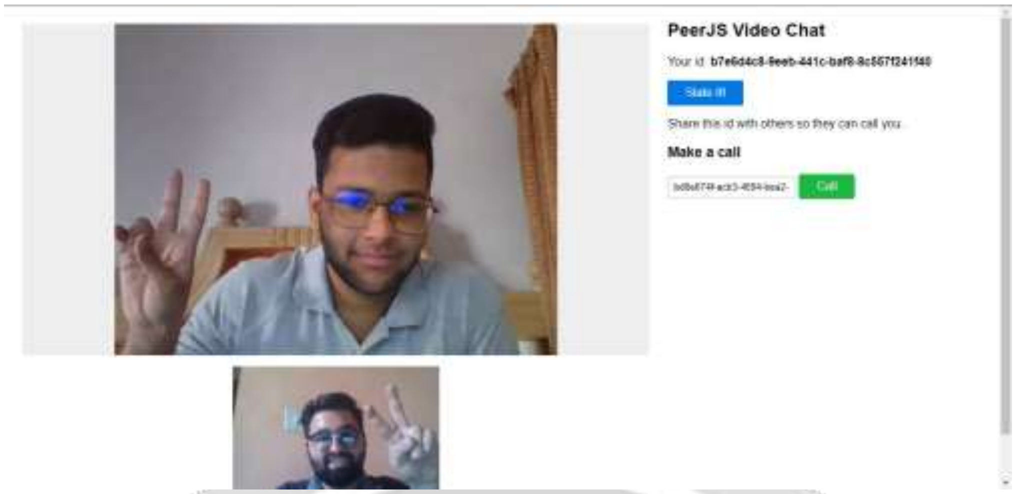

PeerJS Video Chat
Your id: b7e6d4c8-6eeb-441c-baf8-8c557f241f40
Share this id with others so they can call you.
Make a call
Call

Fig. 3 Real-time video communication between peers

4.1 System Performance Analysis

The Fig 4. Graph shows performance of the application. The curves in the graphs shows different attributes of performance for the web application that represents real-time communication system. The analysis and result showed that real-time communication was established and the system performed as per the expectation.

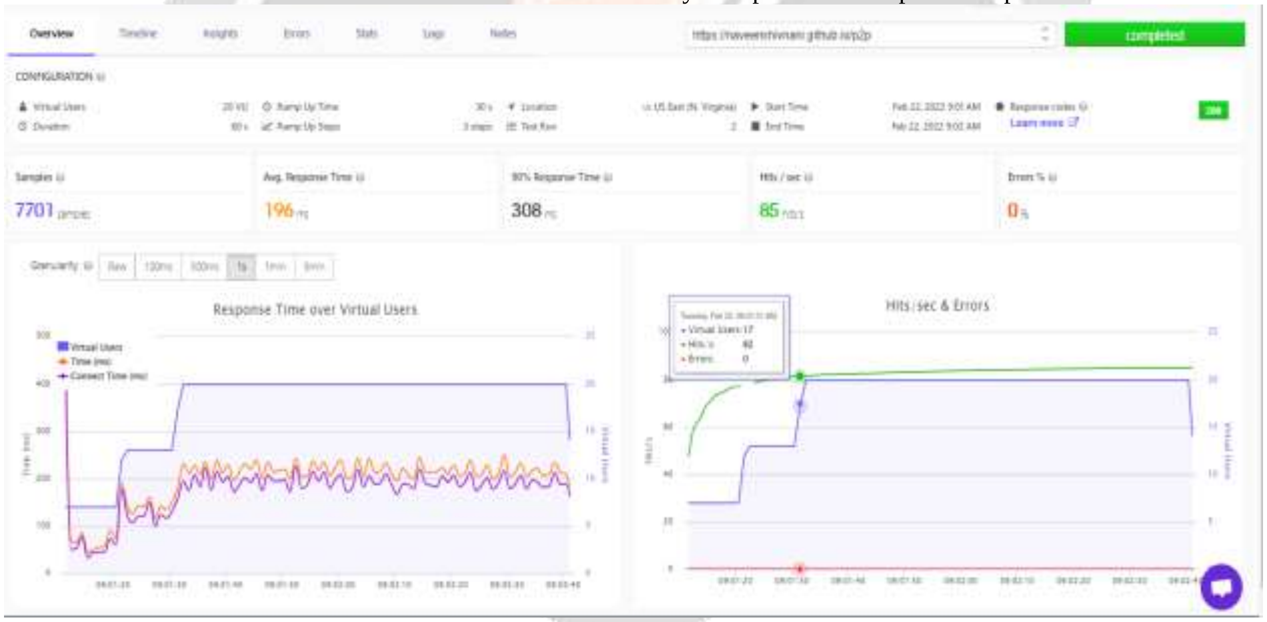


Fig 4. Load Testing and Performance Testing

The system of 20 virtual users recorded a TPS (Transaction processing Time) of 308ms and average response time of 196ms for 1 min uptime of load test. The throughput of the application is directly proportional to number of virtual user, which means the application can easily handle increasing load of more peers with a high throughput. The low and high curves point out fluctuation of response from server while loading the data. The response time is based on the users which indicates that the application can handle number of users with appropriate quality of service. The server response time is an essential factor in UX (user experience) as it shows the waiting time of the users. Response time is the time taken by the peer to connect to the application and real-time communication system which includes request by the peer and receive desired response when the load increases. Based on the recorded data, the peers will experience high quality interaction in real-time with other peers. Considering the results the application recorded error % as 0% which represents the fact that the technologies applied while implementing and designing the system is reliable and the application is able to provide the expected service. The utilization graph of resources like memory, network and CPU. The utilization of these resources can drastically affect the performance of the application. It is safe to say from the results, that the resources are being used efficiently during the real-time communication. It was also seen that the technologies and WebRTC used together to develop and design the architecture of the application successfully enabled real-time communication between peers without any need of third party plugin or other downloadable applications needed to establish the communication. This increases the data security from unauthorized access, eavesdropping and other issues related to security. The freely available server within the WebRTC reduces the cost of communication.

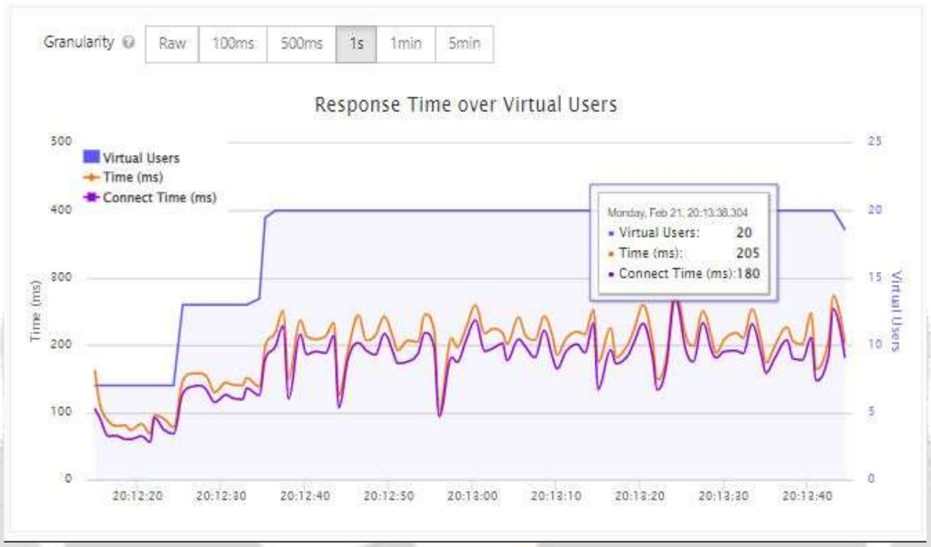


Fig 5. Performance Graph of 20 VUs

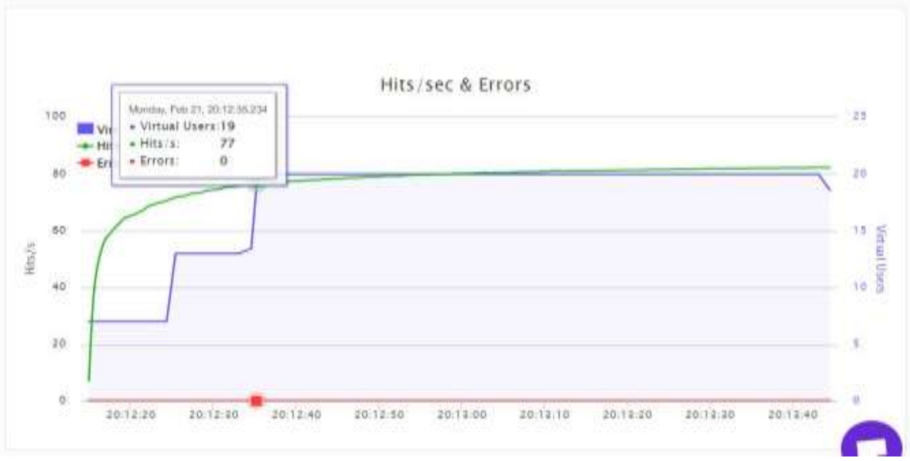
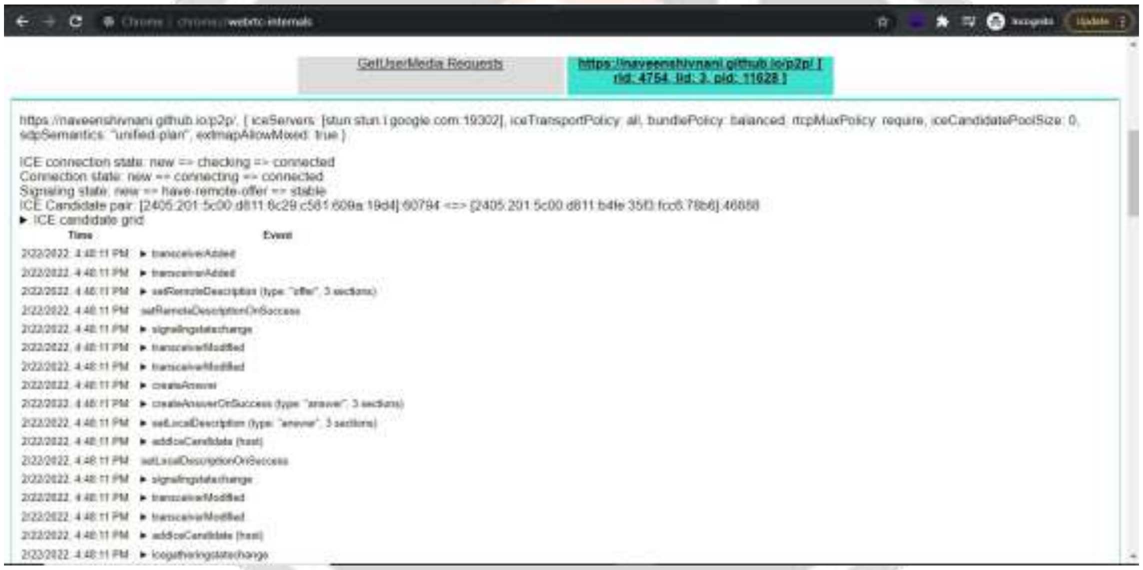


Fig 6. Hit/Sec and Error% graph

4.2 WebRTC Internal Output

Benefits of applying WebRTC on browsers are to produce better media quality. When a communication is established between two peers, it records information of media during the sessions. These results show the stats of performance and processes of the application during the session of communication. These stats are shown in the figure below The WebRTC server measures which indicate the actual performed processes during the session between peers, which include request and response received information, amount of packets sent and received, timestamps. Server also indicates the RTT (round trip time). ICE server also indicates different parameters such as offer and answer, statechange, session description which indicate that there has been a real-time interaction. The details from the server and ICE display WebRTC communication was established between two peers.

The stats in the figure also depict the server and ICE of the two peer's one of them using edge and other using google chrome for the communication session. Apart from that WebRTC stats can also be provided for unsuccessful and successful communication took place as confirmed by the offer creation, CreateOfferOnSuccess, setRemoteDescriptionOnSuccess, setLocalDescriptionOnSuccess, and addICECandidate parameter in fig. The analysis show optimized result of the RTCPeerConnection, getUserMedia, and event of WebRTC and divides the complicated process of SDP, ICE and server processes. The RTCPeerConnection which include ICE remain alive to ensure that UDP does not expire. Once they get access, they make sure peers are actively working and can receive and send media. Every peer involved in this process of communication is represented as a "candidate", which is same as the container that has the complex information that must be known by the other peers to establish the connection. These contain port address, transfer protocol, component ID, IP address and priority. When the offer and answer process is completed, ICE gets involved and does its tasks, while checking and establishing the connectivity.


Chrome | chrome://webrtc-internals
Incognito
Update
GetUserMedia Requests
https://naveenshivnani.github.io/p2p/ [rid: 4754, lid: 3, pid: 11628]

```
https://naveenshivnani.github.io/p2p/, { iceServers: [stun:stun.l.google.com:19302], iceTransportPolicy: all, bundlePolicy: balanced, rtcpMuxPolicy: require, iceCandidatePoolSize: 0, sdpSemantics: "unified-plan", extmapAllowMixed: true }

ICE connection state: new => checking => connected
Connection state: new => connecting => connected
Signaling state: new => have-remote-offer => stable
ICE Candidate pair: [2405:201:5c00:d811:6c29:c561:609a:19d4]:60794 <=> [2405:201:5c00:d811:b4fe:35f3:fcc6:78b6]:46888
▸ ICE candidate grid
        Time                      Event
2/22/2022, 4:48:11 PM  ▸ transceiverAdded
2/22/2022, 4:48:11 PM  ▸ transceiverAdded
2/22/2022, 4:48:11 PM  ▸ setRemoteDescription (type: "offer", 3 sections)
2/22/2022, 4:48:11 PM    setRemoteDescriptionOnSuccess
2/22/2022, 4:48:11 PM  ▸ signalingstatechange
2/22/2022, 4:48:11 PM  ▸ transceiverModified
2/22/2022, 4:48:11 PM  ▸ transceiverModified
2/22/2022, 4:48:11 PM  ▸ createAnswer
2/22/2022, 4:48:11 PM  ▸ createAnswerOnSuccess (type: "answer", 3 sections)
2/22/2022, 4:48:11 PM  ▸ setLocalDescription (type: "answer", 3 sections)
2/22/2022, 4:48:11 PM  ▸ addIceCandidate (host)
2/22/2022, 4:48:11 PM    setLocalDescriptionOnSuccess
2/22/2022, 4:48:11 PM  ▸ signalingstatechange
2/22/2022, 4:48:11 PM  ▸ transceiverModified
2/22/2022, 4:48:11 PM  ▸ transceiverModified
2/22/2022, 4:48:11 PM  ▸ addIceCandidate (host)
2/22/2022, 4:48:11 PM  ▸ icegatheringstatechange
```

Fig 7. WebRTC Internal Parameters for Actual Video Communication

Fig 7. proves that there has been an exchange of data between two peer in different locations. The stats prove that real-time video interaction between the users/peers took place, also the application and system architecture designed using WebRTC has setup a communication between two peers delivered in real-time.

5. DISCUSSION

WebRTC is fairly new and still under development, this technology has great potential to provide enhanced user experience and quality of service on browsers. For this study, a system was designed and developed for real-time video and audio based interactions. The system architecture uses WebRTC components. A real-time video conferencing application was designed and developed to measure and evaluate the system architecture. The technologies used to build this application make it functional and a cross platform solution with minimum standard architecture. The application which was developed was adequately proved and tested across multiple browsers, it might be possible to have unexpected errors as WebRTC is modified or browsers are updated with new features. The system performed extremely well and performed its major function to establish a real-time audio-video based interaction between two peers at remote location with the browsers that support WebRTC. The perks of this application establishment include security from intruders or interception, speed of delivery, reduces bandwidth and latency. The communication was also using end-to-end encryption. A video conferencing application was developed using the feature of WebRTC and other technologies, without the need to install any kind of custom drivers, plugins and software downloads. The application has been hosted on github pages as a part of development process. The deployment platform chosen was flexible in this study because of its advantages and effectively reducing the cost. A load test was performed using "Loadfocus". The results reveled that a real-time interaction took place between two peers at remote locations. Audio and video data was exchanged between the peers. The major factors that could alter the delivery time or speed of any peer request include the CPU utilization, quality of network and signal strength. The user interface was designed in a simple manner for ease of use and simplicity, which in-turn enhances user's/peer's experience.

6. CONCLUSION

In this study, a practical approach in the design and analysis of the application and system architecture for a real-time video conferencing application has been presented. The application was implemented using the features of WebRTC and other technologies which benefits the experience and makes the system more speedy, flexible and cost effective in real-time communication to all its users/peers. WebRTC is available through the peers browsers and minimizes the use of plugins and installation of software application in order to support communication, it has the potential to improve the security for multimedia content and can help the developers to build better and secure and real-time audio-video communication solutions. Keeping these aside there additional benefits of WebRTC including quality of service, user experience, providing better security of user information and data and also reduction in the cost of communication. If this new technology is implemented in a correct way this can help breaking the monopoly of the most OTT corporations and can bring inventive new opportunities which can synchronize with the existing and future applications. WebRTC is still in development phase, more development is under process to standardize policies which will improve the quality of service and technology itself and with the current scenario the need of online based real-time application has increased tremendously making WebRTC a scalable and good fit for developing new applications.

7. REFERENCES

[1] C. Chuang-Yen, C. Yen-Lin, T. Pei-Shiun, and Y. Shyan-Ming, "A Video Conferenceing System Based on WebRTC for Seniors", In: Proc. of the 2014 International Conference on Trustworthy Systems and their Applications, 51-56. 2014.
[2]. R. Manson, "Getting started with WebRTC: Explore WebRTC for real-time peer-to-peer communication", Birmingham, UK: Packt Pub. 2013.
[3]. [3] A. B. Johnston, & D. C. Burnett, "WebRTC: APIs and RTCWEB protocols of the HTML5 real-time web". Publ: Digital Codex LLC, ISBN13: 978985978808 2014.
[4] S. Loreto, S. P. Romano, and L. Miniero, "RealTime Communication with WebRTC: Peer-to-Peer in the Browser". O'Reilly Media, UK. ISBN 1491938080 2016.
[5] D. Ristic, "Learning WebRTC: Develop interactive real-time communication applications with WebRTC",. Packt Publishing ISBN: 9781783983667. 2015.
[6] S. Hudson, "Video-to-Video Using WebRTC. JavaScript Creativity: Exploring the Modern Capabilities of JavaScript and HTML5". ApressBerkely, CA, USA.ISBN:1430259442 9781430259442. 2014.
[7] Altanai, "WebRTC integrator's guide: Successfully build your very own scalable WebRTC infrastructure quickly and efficiently" Birmingham, UK: Packt Pub. 2014.
[8] S. Dutton, "Real-time communication without plugins", Available ffrom: https://www.html5rocks.com/en/tutorials/webrtc/basics / (Accessed: March, 2016). 2012.
[9] B. Patrik, and T. Alexandra, "The Online Paris Café", A Master thesis in Department of Computer science, Electrical and Space engineering. Luleå University of Technolog. Available from: http://www.divaportal.org/smash/get/diva2:1031094/FULLTEXT02. 2013.
[10] A. Bergkvist, D. C. Burnett, C. Jennings, and A. Narayanan, "WebRTC 1.0: Real-time Communication Between Browsers", Available from: https://www.w3.org/TR/2012/WD-webrtc-20120821/ (Accessed: March, 2016). 2012.

[11] M. Di Mauro, & M. Longo, "Revealing Encrypted WebRTC Traffic via Machine LearningTools". In: Proc. 12th International Joint Conference on e-Business and Telecommunications (ICETE) IEEE Conference Publications 4, 259-266. 2015.
[12] D. D. Bakwa and A. E. Edim, "Application Of M.E.A.N Stack Restive API And WEBRTC In The Design Of A Real-Time Telemedicine Service" International Journal of Innovative Research and Advanced Studies (IJIRAS) 4(3), 311-322. 2017.
[13] B. John, "4 video chat alternatives that beat Skype", Available from: http://www.foxnews.com/tech/2015/10/28/4-videochat-alternatives-that-beat-skype.html. (Accessed June, 2016). 2015.
[14] K. Shuang, X. Cai, P. Xu, and Q. Jia, "WebCDN: A Peer-To-Peer Web Browser CDN Based WebRTC", In: Yao L., Xie X., Zhang Q., Zomaya A., Jin H. (eds) Advances in Services Computing. Lecture Notes in Computer Science, Vol. 9464. Springer, Chan. 2015.
[15] S. Loreto, S. P. Romano, "Real-time Communication with WebRTC: Peer-to-Peer in the Browser", Pub: O'Reilly Media. UK. ISBN: 978-1-4493- 7187-6. 2014a.
[16] A. Gary, "Network Computing" Available from: http://www.networkcomputing.com/unifiedcommunications/9-advantages-webrtc/195325984(Accessed May, 2016). 2014.
[17] H. W. Barz, H. W. and G. A. Bassett, "WebRTC, in Multimedia Networks: Protocols, Design, and Applications", John Wiley & Sons, Ltd, Chichester, UK. doi: 10.1002/9781119090151. 2016.
[18] J. C. Zhang, W. Barnes, D. King, "Getting Started with WebRTC and Test Driven Development", Available from https://medium.com/@coldbrewtesting/getting-startedwith-webrtc-and-test-driven-development1cc6eb36ffd#.yswz9omvt (Accessed: Jan. 2017). 2016.
[19] A. W. Sime, "WebRTC: Delivering Telehealth in the Browser", Journal of mHealth, 1-2 doi:10.21037/mhealth.2016.03.08. 2016.
[20] A. Arrichiello, "Learning WebRTC application development. Birmingham", UK: Packt Pub. 2014.
[21] M. Maruschke, O. Jokisch, M. Meszaros, M., & V. Iaroshenko, "Review of the Opus Codec in a WebRTC Scenario for Audio and Speech Communication", In: Proc. Speech and Computer International Conference, SPECOM, Athens, 348-355. 2015.
[22] D. Odell, Using WebRTC for Video Chat. Pro JavaScript Development, Apress. ISBN: 978-1-4302- 6269-5. 321-339. 2014.
[23] S. Branislav, S. Dragan, & P. Dragan, "WebRTC technology overview and signaling solution design and implementation", In: 38th International Convention on Information andCommunicationTechnology, Electronics and Microelectronics (MIPRO), IEEE Conference Publications. PP. 1006 – 1009. 2015.
[24] A. Sergiienko, "WebRTC blueprints: Develop your very own media applications and services using WebRTC", Birmingham, UK: Packt Pub. 2014.
[25] A. Sergiienko, WebRTC cookbook: Get to grips with advanced real-time communication applications and services on WebRTC with practical, hands-on recipes. UK: Packt Pub. ISBN: 9781783284450. 2015.
[26] A. Roach, "WebRTC Video Processing and Codec Requirements", Available from: https://tools.ietf.org/html/draft-ietf-rtcweb-overview. doi:10.17487/rfc7742 (Accessed June, 2016). 2016.
[27] G. Ilya, "High Performance Browser Networking", First Edition. Publisher: O'Reilly Media, Inc. ISBN: 9781449344757. 2013.
[28] Y. Helsingin, T. Matemaattisluonnontieteellinen, L. Tietojenkäsittelytieteen, & A. Hussain, "WebRTC in presence of NAT, firewalls and HTTP proxies" Thesis / Dissertation ETD. Available from: http://hdl.handle.net/10138/153590. 2015.
[29] G. P. Kedar, M. C. Pushpanjali, "WebRTC Implementation and Architecture Analysis", International Journal of Scientific & Engineering Research, 7(2), 2229-5518. 2016.
[30] S. Loreto, S. P. Romano, "Real-time communication with WebRTC", O'Reilly Media, Sebastopol, CA. 2014b.