As per UGC guidelines an electronic bar code is provided to seure your paper

International Journal for Modern Trends in Science and Technology, 7(12): 186-189, 2021

ISSN: 2455-3778 online
DOI: https://doi.org/10.46501/IJMTST0712035
Available online at: http://www.ijmtst.com/vol7issue12.html

Performance Analysis & Implementation of A/V Real Time Communication

Neelam Sharma | Naman Arora | Nandeesh Gupta | Pradhuman Singh

Department Information Technology, Maharaja Agrasen Institute of Technology, New Delhi, India.

To Cite this Article

Neelam Sharma, Naman Arora, Nandeesh Gupta and Pradhuman Singh. Performance Analysis & Implementation of A/V Real Time Communication. *International Journal for Modern Trends in Science and Technology* 2021, *7* pp. 186-189. https://doi.org/10.46501/IJMTST0712035

Article Info

Received: 12 November 2021; Accepted: 05 December 2021; Published: 13 December 2021

ABSTRACT

Audio-Video conferencing, digital meetings, and interviews have become commonplace in today's workplace, with the benefits of saving resources, as well as decreasing travel-related environmental impact. Despite this, the present technology is unable to provide immediate real-time feedback while transmitting, recording, or showing critical information, sophisticated collaborations such as interactive teamwork still require the physical presence of all participants. For almost 20 years the World Wide Web (WWW) has offered Web-based conferences, such as webinars and webcasting.

Since then, technology has gone from unidirectional to half-duplex and full-duplex media sharing with the help of advanced video codecs, which seek to improve performance and reduce latency, data loss, and the need for resources, such as bandwidth and CPU

WebRTC is an open-source project aiming at providing JavaScript developers with RTC APIs, allowing the implementation of online applications of several RTC functions, including telephone calls, video conferences, and peer-to-peer browser file sharing. WebRTC APIs operate using two technologies: JavaScript and Hypertext Mark-up Language (HTML).

Since its release, efforts have been made to integrate it with other communication technologies, frameworks, and platforms to enable the development of rich, high-quality RTC applications for browsers, mobile platforms, and IoT devices, allowing them all to communicate using a common set of protocols. WebRTC in conjunction with the correct network topology and other technologies like OpenCV enables a variety of appealing features with a smooth video streaming experience. Investigating the viability of properly combining these technologies with WebRTC can improve the way RTC works. Remote collaboration within companies is one area where WebRTC implementation is becoming common.

KEYWORDS: Web RTC, Face detection, Real Time Communication, JavaScript, Node.js.

INTRODUCTION

Audio-Video conferencing, digital meetings and interviews have become commonplace in today's workplace, with the benefits of saving resources, as well as decreasing travel-related environmental impact. Despite this, current technology is still lacking in addressing issues such as scalability, integration, and interoperability in order to successfully enable multi-party remote collaboration while maintaining a decent user experience. Traditional video conferencing solutions don't have the flexibility to adjust to changing system resource requirements such as CPU, memory, disc, and network usage. Furthermore, since present technology is unable to provide immediate real-time feedback while transmitting, recording or showing critical information, sophisticated collaborations such

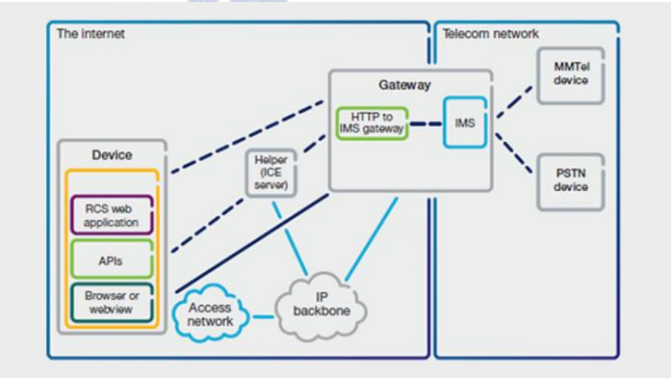
as interactive teamwork still require the physical presence of all participants. Enterprises demand fast and adaptable solutions that allow multi-part interactive RTC in order to maximize their development and business processes. Due to the versatility of WebRTC, which allows lightweight RTC with unique capabilities like screen sharing, screen recording, whiteboard sharing, and video/audio chat via the web, it has been recognized as the solution to most of the difficulties in RTC. For almost 20 years the World Wide Web (WWW) has offered Web-based conferences, such as webinars and webcasting. Since then, technology has gone from unidirectional to half-duplex and full-duplex media sharing with the help of advanced video codecs, which seek to improve performance and reduce latency, data loss and the need for resources, such as bandwidth and CPU. Digital and video cameras at affordable prices on PCs and cellular phones are contributing to the rise of Internet video communications in real time. Some of the motivations for using WebRTC include the current initiatives in online communication in real time, as well as the rising need for more innovative apps that give better user experience and are simpler to connect with the WWW. The WebRTC is an open-source project aiming at providing JavaScript developers with RTC APIs, allowing the implementation of online applications of several RTC functions, including telephone calling, video conference and peer-to-peer browser file sharing. WebRTC APIs operate using two technologies: JavaScript and Hyper Text Markup Language (HTML). Since its release, efforts have been made to integrate it with other communication technologies, frameworks, and platforms in order to enable the development of rich, high-quality RTC applications for browsers, mobile platforms, and IoT devices, allowing them all to communicate using a common set of protocols. WebRTC in conjunction with the correct network topology and other technologies like OpenCV, enables a variety of appealing features with a smooth video streaming experience. Investigating the viability of properly combining these technologies with WebRTC can improve the way RTC works. Remote collaboration within companies is one area where WebRTC implementation is becoming common.

OBJECTIVES

- Exploring the potential gains on video streaming performance by using machine learning in the system.
- Detect multiple faces using face API and averaging Pixel of the background
- Using the results of above research to implement a multi-party, cross platform video communication software.

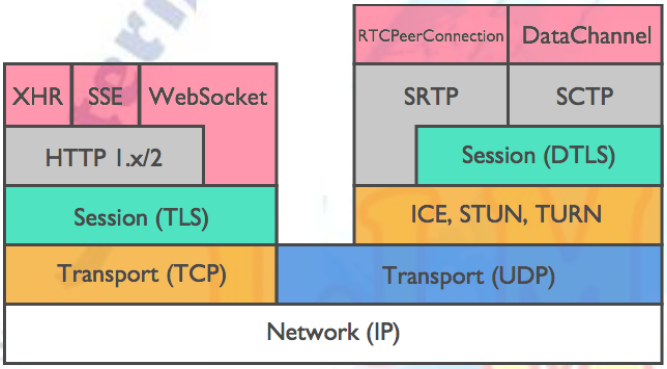
TECHNOLOGY USED

- FaceAPI: It is a JavaScript API for face detection and face recognition in the browser implemented on the top of the tensorflow.js core API.
- JavaScript: JS is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions. While it is most well known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js
- NodeJS: NodeJS is a JavaScript runtime built on Chrome's V8 engine. It provides event-driven I/O to the web by abstracting away the complexity of network communications and storage. In short, Nodejs lets you build server-side applications with JavaScript, making them lightweight and highly performant
- Web RTC: Web RealTime Communication (WebRTC) is an open-source project that offers the real time media communication such as video, data and voice transmission between browsers and devices. This gives users the ability to communicate through their web browser without the need for complicated plugins or additional hardware.
- WebSocket: It is computer communication protocol that provides full duplex communication channels over a single TCP connection.



SCOPE

- With the advent of fast internet to the masses, online video conferencing has seen massive growth. Any further advancements we can make in this area will directly lead to huge savings on computational resources and network bandwidth required.
- In a peer-to-peer connection, the size of data to be sent dictates the latency experienced by users. By Leveraging the advances of Machine Learning in Face Detection, we can explore the possible ways to decrease stream size and improve performance
- This study aims to compare the different ways to improve Audio-Video RTC and realize the findings as a publicly available software.



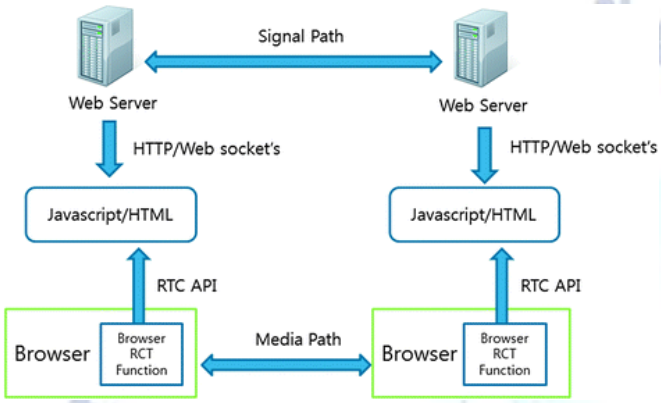
IMPLEMENTATION

Implementation of the project has been done in two following

- Analysed the protocols (VoIP) and techniques of various video calling APIs that includes, TokBox, Sinch, Quickblox, Twilio, CometChat, Pubnub.
- Evaluated the developments that are being done for improving video calling by Disney and Nvidia that involves AI-enhanced video compression mode.
- Developed a user interface using JavaScript, Face API for enabling users to have real time video communication.
- Implemented a Pixel Averaging Algorithm inspired by the use of blur filters in real time one on one communication that averages out the nearby pixels leaving out the face box, this algorithm thus modifies the actual feed provided to the system so that is can be properly compressed by DEFLATE Algorithm that implemented in WebRTC.
- DEFLATE ALGORITHM works by combining two lossless compression techniques i.e., Huffman's & Lempel-Ziv encoding technique. Stream is then divided into 3 parts i.e., uncompressed data, compression using fixed Huffman codes and compression using dynamic codes. Our modified feed then can be easily compressed by dynamic Huffman Codes which thus reduces size of our stream feed.
- For text transmission along with RTC feed, SDP data compression technique is being used which has some predefined prefix string in encoders and decoders for lesser bit transmission, along with this the unencoded data is then compressed using data tables.
- The goal is to minimise the total bandwidth used while maintaining a good resolution around the face box and pixelating the rest of the feed.
- During poor connectivity viewers can enjoy a smooth video interaction with have minimal latency.
- Pixelated video contributes into fast transmission of the stream over networks which in turn give an edge for a reduced bandwidth.
- Implemented a way such as to effectively use external streaming devices / camera for as a single source. Such as a user can decide from which device to stream for either a Phone camera of 64 Megapixel or a WebCam of 2 Megapixel.
- The device cameras and microphones are then accessed using MediaStream to provide a minimum latency effect between devices while having a reliable connection of the media devices that capture and render media data.
- RTCPeerConnection which is the core of WebRTC, manages to establish a direct connection to peers without the need of an intermediatory server.
- This helps us to further reduce the latency since the server element is eliminated which is quite suitable for one-on-one conversation.
- RTCPeerConnection also manages the media transfer, NAT Traversal, Codec Implementation, SDP negotiation, packet loss and Bandwidth management.
- RTCDataChannel is being used for Bi-directional data transmission between any two peers which is designed to mimic WebSocketAPI, but instead of relying on a TCP connection, which is reliable but has high latency and is prone to bottlenecks, the data channels use UDP-based streams with the reconfigurability of the protocol Stream Control

Transmission Protocol (SCTP). This design enables the best of both worlds: reliable delivery like TCP, but with reduced network congestion like UDP.

- This process effectively reduces our transmission latency as well as our bandwidth used for smooth and easy communication between peers.



FUTURE SCOPE

- Reduced Bandwidth and improved video quality: SR3 are a super-resolution diffusion model that uses a low-resolution image as input and creates a corresponding high-resolution image from pure noise. The model is trained on an image distortion process in which noise is gradually added until it becomes a pure noise from a high-resolution image. Then the reverse process is learnt by starting from pure noise and then gradually eliminating the noise to achieve the distribution target from the low-resolution input image.

CONCLUSION

With the help of Face APIs pre trained models, we successfully recognised faces and implemented pixel averaging around the detected face. This resulted in a video which will be smaller in size while sending through the WebRTC, which also provides lower latency due to RTCDataConnection over the internet.

ACKNOWLEDGEMENT

We would like to extend our gratitude towards our guide and corresponding author Ms. Neelam Sharma (Assistant Professor MAIT, GGSIPU). Her constant guidance and support have made this research possible.

REFERENCES

[1] "Here are the most popular Video chat APIs to be considered for your mobile app " https://appsexpert.medium.com/here-are-the-most-popular-video-chat-apis-to-be-considered-for-your-mobile-app-da983915215b

[2] "NVIDIA swaps codecs for AI to improve video call compression." https://www.fiercevideo.com/tech/nvidia-swaps-codecs-for-ai-to-improve-video-call-compression

[3] "Detecting Face Features and Applying Filters with JavaScript" https://livecodestream.dev/post/detecting-face-features-and-applying-filters-with-javascript/

[4] "Snapchat Style Filters with Tracking.js and Vonage" https://learn.vonage.com/blog/2019/04/03/snapchat-filters-opentok-tracking-js-dr/

[5] "SDP Compression Algorithm for WebRTC: ESDiPi " http://www.ijcee.org/vol8/916-IG015.pdf

[6] "Tweaking WebRTC video quality: unpacking bitrate, resolution and frame rates" https://bloggeek.me/tweaking-webrtc-video-quality-unpacking-bitrate-resolution-and-frame-rates/

[7] "Could AI make all our old blurry photos clear?" https://www.malaymail.com/news/tech-gadgets/2021/09/11/could-ai-make-all-our-old-blurry-photos-clear/2004666

[8] "Criminals beware! Google's AI can now identify faces from heavily pixelated images" https://www.wired.co.uk/article/google-brain-ai-pictures-blur

[9] "Designing a Large-Scale Video Chat Application" https://www.researchgate.net/publication/221571646_Designing_a_Large-Scale_Video_Chat_Application

[10] "Real-Time Face Detection and Tracking for High Resolution Smart Camera System" https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.585.6365&rep=rep1&type=pdf

[11] "Working and Implementation Tracking JS " https://trackingjs.com/

[12] "Working of Face API JS" https://justadudewhohacks.github.io/face-api.js/docs/index.html