A
SYNOPSIS
ON

Implementation of WebRTC API for real time peer to peer communication and data transfers.

Submitted in partial fulfillment of the requirement for the award of degree of

BACHELOR OF ENGINEERING

IN

(COMPUTER SCIENCE AND ENGINEERING)

By:
Sagar Ramrao Bharad
Prashish Gautam Borkar
Shreya Bhojraj Gawali
Janhavi Jayant Karande
AnilKumar Ramji Sharma

UNDER THE GUIDANCE OF
Prof. D. J. Chaudhari

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GOVERNMENT COLLEGE OF ENGINEERING, CHANDRAPUR, (M.S.)

Year 2020-2021

1. Introduction

WebRTC is new Real Time communication industry standard developed by joint effort of IETF and W3 consortium along with precious support from global industry leaders namely Google, Mozilla, and Microsoft. WebRTC leverages a set of plugin-free APIs that can be used in both desktop and mobile browsers, and is progressively becoming supported by all major modern browser vendors.

WebRTC or Web Real-Time Communication is a State-of-the-Art open technology that makes real-time communication capabilities in audio, video, and data transmission possible in real-time communication through web browsers and desktop environment using JavaScript APIs (Application Programming Interfaces). WebRTC uses peer to peer network architecture to achieve Real Time low latency low bandwidth audio-visual communication and data exchange.

These systems users can view, record, remark, or edit video and audio content flows using time-critical cloud infrastructures that enforce the quality of services
These systems can be used in identifying Elevated Body Temperature in wake of COVID 19 using Thermal FLIR surveillance cameras over the web without Human interventions

WebRTC is used in many major implementation such as: Google Meet | Facebook Messenger | Discord | Amazon Chime | Appear.in and platforms such as WebKitGTK+ and Qt native apps.

WebRTC promises to provide secured direct P2P communication between users and free of plug-ins. WebRTC assures a simplified, flexible, and cost-effective means of real-time communication for users without dependence on service providers

2. Problem Statement:

As the internet, social media and mobile systems become prevalent lots of recent research works focused on customer engagement centric application such as real-time communication.

Real Time Communication (RTC) was a challenge in web industry due to proprietary work and solution done may many industry experts which were costly. |Moreover these solutions were not feasible as implementation of such systems were not efficient. They tend to take huge resources in terms of man power as well as infrastructures

RTC is challenging problem in modern world networking which are subject to fading, interference, being unreliable, and rapid varying channel quality

WebRTC is an open source set of API's originally proposed by Justin Uberti and Peter Thatcher at Google which were then standardized by Internet Engineering Task Force (IETF) at protocol level and World Wide Web Consortium (W3C) at Internet.

.

3. Objective:

WebRTC is a set of JavaScript(ECMA Script6) API's, STUN server (Session Traversal of User Datagram Protocol) , TURN server (Traversal Using Relay NAT) and ICE candidate .A WebRTC application will usually go through a common application flow. Accessing the media devices, opening peer connections, discovering peers, and start streaming.

WebRTC will implement three API's**:**

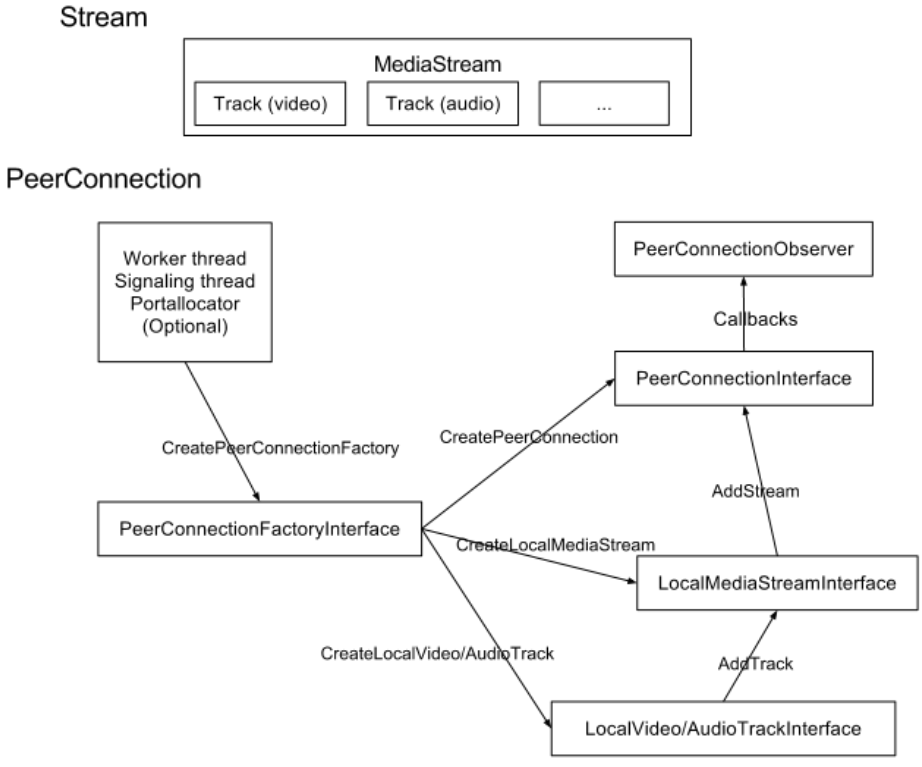
MediaStream (aka getUserMedia): The MediaStream API represents synchronized streams of media, e.g.: a stream taken from camera and microphone input has synchronized video and audio tracks.

RTCPeerConnection: RTCPeerConnection is the WebRTC component that handles stable and efficient communication of streaming data between peers. RTCPeerConnection shields web developers from the myriad complexities that lurk beneath the codecs and protocols used by WebRTC do a huge amount of work to make real-time communication possible, even over unreliable networks:

Packet loss concealment | echo cancellation | bandwidth adaptivity | dynamic jitter | buffering | automatic gain control | noise reduction and suppression |image 'cleaning'.

RTCDataChannel: The RTCDataChannel API enables peer-to-peer exchange of arbitrary data, with low latency and high throughput. The API has several features to make the most of RTCPeerConnection and enable powerful and flexible peer-to-peer communication:

Leveraging of RTCPeerConnection session setup. | Multiple simultaneous channels, with prioritization | Reliable and unreliable delivery semantics | Built-in security (DTLS) and congestion control. | Ability to use with or without audio or video.



4. Literature Survey

1. P. Nuño, F. G. Bulnes, J. C. Granda, F. J. Suárez and D. F. García, "*A Scalable WebRTC Platform based on Open Technologies*," 2018 International Conference on Computer, Information and Telecommunication Systems (CITS), Colmar, 2018, pp. 1-5, doi: 10.1109/CITS.2018.8440161.
2. G. Suciu, S. Stefanescu, C. Beceanu and M. Ceaparu, "*WebRTC role in real-time communication and video conferencing*," 2020 Global Internet of Things Summit (GIoTS), Dublin, Ireland, 2020, pp. 1-6, doi: 10.1109/GIOTS49054.2020.9119656.
3. Nayyef, Zinah & Amer, Sarah & Hussain, (2019). *Peer to Peer Multimedia Real-Time Communication System based on WebRTC Technology*. International Journal for the History of Engineering & Technology. 2.9. 125-130.
4. "*Real-time communication for the web*" May. 28, 2019. [Online accessed on 05 September 2020]. Available: https://webrtc.org/
5. "*WebRTC 1.0: Real-time Communication Between Browsers W3C Candidate Recommendation*" Aug. 25, 2020. [Online Accessed on: September 5, 2020. Available: https://www.w3.org/TR/webrtc/
6. Sam Dutton "*Getting Started with WebRTC*" Feb. 21, 2014 [Online Accessed on: September 1, 2020] Available: https://www.html5rocks.com/tutorials/webrtc/basics/
7. *WebRTC Audio Codec and Processing Requirements*, IETF Working Group, https://tools.ietf.org/html/rfc7874.
8. Jansen, Bart & Goodwin, Timothy & Gupta, Varun & Kuipers, Fernando & Zussman, Gil. (2018). *Performance Evaluation of WebRTC-based Video Conferencing*. ACM SIGMETRICS Performance Evaluation Review. 45. 56- 68. 10.1145/3199524.3199534.
9. L. L. Fernandez, M. P. D ´ ´ıaz, R. B. Mej´ıas, F. J. Lopez, and J. A. Santos, ´ "*Kurento: a media server technology for convergent WWW/mobile realtime multimedia communications supporting WebRTC*." in Proc. IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'13), Madrid, Spain, Jun. 2013.
10. Yuan Liao, Zhonghua Wang and Yanfen Luo, "*The design and implementation of a WebRTC based online video teaching system*," 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), Xi'an, 2016, pp. 137-140, doi: 10.1109/IMCEC.2016.7867188
11. C. Cola and H. Valean, "*On multi-user web conference using WebRTC*," 2014 18th International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, 2014, pp. 430-433, doi: 10.1109/ICSTCC.2014.6982454.
12. Perumal, M.; Wing, D.; Ravindranath, R.; Reddy, T.; Thomson, M. (October 2015). Session Traversal Utilities for NAT (STUN) Usage for Consent Freshness. IETF. .

5. Methodology

WebRTC application will implement three API's in a procedural process.

- MediaStream (aka getUserMedia)
- RTCPeerConnection
- RTCDataChannel

In development phase the initial stage will implement the backend system such as STUN and TURN servers and backend media servers. In second stage the application will demonstrate single peer to peer ICE candidate connection. In later stage these single peers will be broaden out to include multiple ICE candidate and finally in last stage the application will demonstrate file and data sharing. WebRTC development consists of various stages:

Stage 1:

Get streaming audio, video or other data. The MediaStream API represents synchronized streams of media. Each MediaStream has an input, which might be a MediaStream generated by getUserMedia(), and an output, which might be passed to a video element or an RTCPeerConnection. The getUserMedia() method takes a MediaStreamConstraints object parameter, and returns a Promise that resolves to a MediaStream object.

Signalling: session control, network and media information: Signalling methods and protocols are not specified by WebRTC: signalling is not part of the RTCPeerConnection API. Here the singling mechanism will be implemented using Node.js - a c++ based JavaScript runtime environment using Google chrome's v8 engine and Scokets.io – a JavaScript library for Node.js to implement web sockets. The Node.js app will run as server side script providing access point for user and media streams.

Stage 2:

RTCPeerConnection plus servers: Users discover each other and exchange 'real world' details such as names. WebRTC client applications (peers) exchange network information. Peers exchange data about media such as video format and resolution. WebRTC client applications traverse NAT gateways and firewalls.

Stage 3:

RTCDataChannel : Communication occurs directly between browsers, so RTCDataChannel can be much faster than Web Socket even if a relay (TURN) server is required when 'hole punching' to cope with firewalls and NATs fails. In this development stage the system will transfer data over the connection.

6. Hardware and Software Requirement

- Hardware specification:

Processor:	Intel core Duo 1.2 GHz or higher AMD Ryzen 2 1.0 GHz or higher
Ram:	2 GB DDR4 2400 MHz or higher
Host OS:	Windows 7 SP1, Ubuntu Linux 16.07 or higher with webcam support

- Software specification:

Front-End: `	HTML 5, JavaScript (ECMA script 6)
Development Environment:	Microsoft Visual Studio 2019
Back-End:	Node JS 14.8.0, Sockets.io 6.14.7
UI Interfacing:	CSS 3
VCS (Version Control System):	Local – Git 2.27 Cloud- Microsoft Github

- Preferred System Runtime Environment:

`	Google Chrome 85.0.2 Microsoft Edge 84.0.1 Mozilla Firefox 80.2.2

Other compliant web Browser with WebRTC Support