Design and implementation of a Browser Talk for video and text chat applications

P.Vijaya Pal Reddy

Matrusri Engineering College, Hyderabad
drpvijayapalreddy@gmil.com

Abstract

In this paper, the design and implementation of a Browser Talk which is a browser based video and text chat application using WEBRTC framework which runs on JavaScript enabled browsers in all platforms is presented. Two people can chat using video, audio and text using JavaScript enabled web Browser. Based on research, it is utilized the node.js, a cross-platform server side JavaScript environment which optimizes application's performance and scalability. It is also discussed the architecture, components and advantages of WEBRTC based Browser application. This application is a P2P application but uses signaling server for exchanging metadata and networking information. Bandwidth analysis of server and client load of Browser Talk is also presented

Keywords: *JavaScript-node.js, cross-platform applications.*

1. Introduction

Browser based video conferencing with text functionalities is useful for internal troubleshooting in a company since it is fast and uses LAN for communication. Web Real time Transport Connection (WEBRTC) framework is a popular API that is used for developing browser application with voice calling, video calling and file sharing functionalities. This implementation enables peer to peer connection, but requires signaling server for exchanging Meta data, network information. Signaling server is implemented in node.js and front end GUI is developed using HTML5 and JavaScript.

A. Voice over Internet Protocol

VoIP is a service for delivery of voice and multimedia communication using public IP networks as medium. Personal computers and mobile devices can use VoIP services using GPRS, Wi-Fi, 3G, Edge, 4G LTE and broadband technologies.

B. Web RTC

Web Real Time Communication is an Application Programming Interface (API) drafted by Worldwide consortium (W3C) [3] to share data between browsers without plug-ins. Web RTC API enables video chat, peer to peer file transfer and voice calling between browsers communicating using IP addresses. Various browsers supporting WebRTC API has been shown in

Operating System	*Browser*
Desktop PC(Windows/Linux)	Google Chrome,Mozilla Firefox,Opera
Apple IOS(Mac,Iphone,Ipad)	Browser App(available at Appstore)
Chrome OS	Chrome
Firefox OS	Firefox
Android OS	Chrome Firefox,Opera Mobile

TABLE I. Components for Instant Messenger development

C. WEBRTC architecture

Using WebRTC API, we can develop rich audio,video and multimedia application on web without third party plug-ins. WebRTC applications work across multiple browsers and platforms. WebRTC architecture has been presented in figure1 .
Various modules of WebRTC frameworks such as Web App, Web API, WebRTC native C++ API, Session management, RTP stack, STUN/ICE.

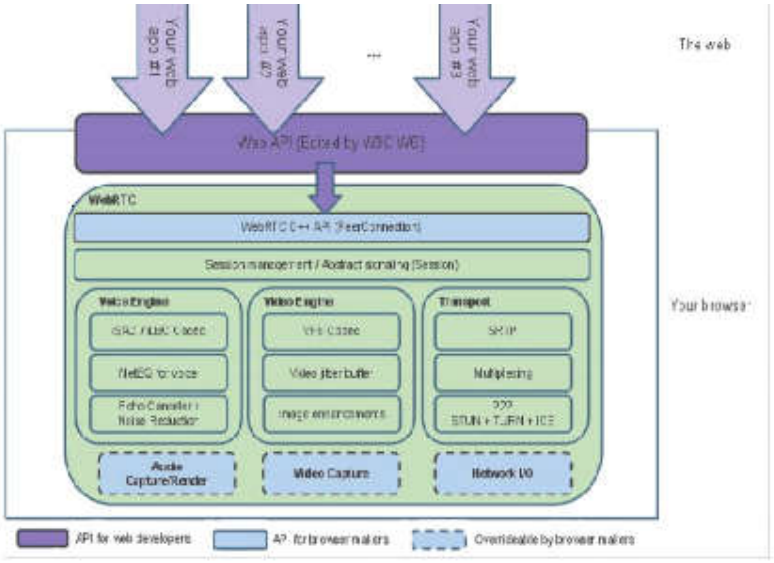


Fig. 1. WebRTC Architecture

Voice Engine - Its an audio media chain framework for connecting sound card to the network. Framework includes audio codecs, noise reduction, jitter control modules. Description various codecs used are in Table II. NETeq is a network equalizer implementation for jitter buffered optimized for voice. NetEQ module functionality is to maintain highest voice quality with latency as low as possible. It uses network concealment algorithm against packet loss and network jitter. Acoustic Echo Canceler(AEC) is software based signal process component that is responsible for removing acoustic echo coming out of the device into the microphone. Noise reduction (NR) is a software base signaling component to remove the unwanted noise like (hiss, fan sound, etc) for the VoIP service.

Video Engine - Its an video media chain framework for connecting camera to the network and from network to screen.VP8 is a video codec, its part of goggles WEBM project. It suits for Real time communication due its low latency. Video Jitter Buffer, a software based signaling component is a dynamic jitter buffer for video. Helps overcomes the effects of jitter and packet loss on total video quality. Image enhancements module, it removes video noise from the image captured by the web camera device.

2. Related Work

JavaScript, as it's supported by browsers across all platform needs to be used to develop platform independent application. Node.js, a popular JavaScript framework follows strict asynchronous, non-blocking I/O and event driven programming model. In Asynchronous, non-blocking I/O model, the application doesn't get blocked while waiting for an I/O operation, hence reducing context switching time. Event driven programming is

generally scalable and efficient model over multi-threaded programming model in terms of building concur-rent and high performance applications Tilkov [1] and Lerner [9] discusses importance of node.js framework for developing real time web application with minimal effort. It illustrates a basic example of node.js application. This paper discusses about node package manager (npm) which is used for deploying node.js application includes socket.io, express.js and other module libraries and dependencies. McCune [8] discussed about scalability and performance issues of node.js framework. This paper measures response time and core utilization of node.js serve load. This paper compares the node.js framework with Apache and Event machine multi-threaded frameworks. Their analysis showed node.js has high concurrence and low data throughput of node.js framework in comparison with other. Goode [5] describes the factors for making high quality VOIP call. It details about the effect of bandwidth and delay on call quality. This paper explains the cost benefits of VOIP voice over Public switched telephone network (PSTN). It discusses about codec selection, delay budget, network quality of service and VOIP call signaling protocols. It also discusses VOIP issues with Network address translator(NAT) and firewalls. Amirante [10] discusses about integration of WEBRTC clients and making WEBRTC inter-operable with existing standards-based collaboration platform.

This paper discusses use of HTML5 for WEBRTC client browser and Session Description protocol (SDP). This paper also discusses ICE framework for network peer discovery. In [6] discusses support of multi-party communication between endpoints over an IP networks for WEBRTC technology.

This paper discusses conference mode for webrtc applications. Guha [11] discusses performance analysis of skype video conferencing software. Bandwidth usage at peer and server level has been estimated in it. In [7] discusses need of advanced video conferencing services such as session recording, media mixing and adjusting to varying network conditions for the WEBRTC framework.

3. Proposed Work

A. Application Functionality

1) Establish WEBRTC connection between two re-motley connected browsers. 2) Stream Video between two remote client browsers. 3) Stream Audio between two client browsers. 4) Implementation of signaling server for identification of IP addresses of the connected browsers. Connection of two remote clients using a namespace identifier(room) along with URL. Bandwidth usage of the remote clients and the signaling server needs to be measured.

a. Algorithm

Algorithm 1: User's room assignment

User connects to the signaling server with ROOM identifier;

If session exist with ROOM identifier and number of users ==1 then

Add the User to the session;

Exchange networking information and metadata with peer via signaling server;

Offer/Answer Mechanism between peers; Server sends a HTML5 page with video plugin embedded of the two users connected;

Ask user permissions for accessing camera and microphone; Stream Audio, Video and text capture by connected devices between the peers;

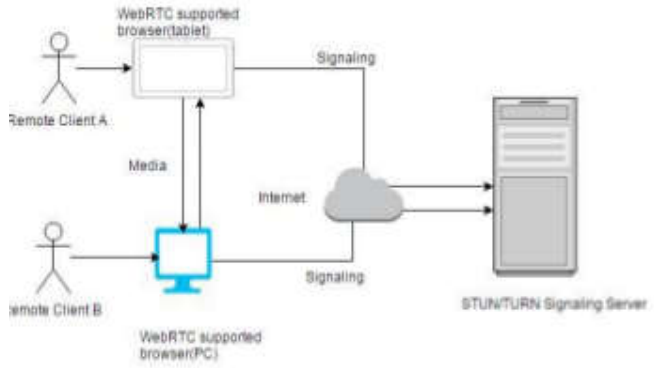
Ask user permissions for accessing camera and microphone;

Else

server; Server sends a HTML5 page with video plug-in embedded of the user connected; Displays the video and audio captured by the device as html frame;

end

B. The Proposed Model



c. Web rower components

WebAPI Interface	*Description*
getUserMedia	Gives permission to the browser to access built-in webcamera.Microphone and media services
RTCPeerConnecti on	Establishes WEBRTC connection and handles efficient steaming of data between browsers of two remotely connected devices.
RTCDataChannel	denotes a bi-directional channel between two remote browsers in a connection
getStats	Used to get remote user connection statistics like bandwidth usage, IP addr

TABLE III. Interfacing with Web API's

d. Signaling server Implementation

WEBRTC establishes browser to browser (P2P) connection.WEBRTC uses signaling server for for exchanging metadata for coordination between the browsers, for handling mechanism for a browser behind a Network Address translation (NAT) and firewalls. Signaling process implementation is application specific. It is a coordinating mechanism between clients to exchange metadata information like, Session control messages for opening and closing connection, Media metadata like bandwidth, media types, codecs and codec settings, For secure connections (Public key Infrastructure keys), Host IP addresses which may be public or behind Network address translators. Current Implementation uses Javascript session Establishment protocol since its deployed using socket.io and node.js

frameworks. Signaling state information is stored at server side as there is chance of losing state browser cache while updation or deletion.

1) Offer and Answer Mechanism: Consider two clients A and B which wants transmit metadata using signaling protocol.

• 'A' creates a RTCPeerConnection object.

• 'A' creates an offer with RTCPeerConnection's createOffer() method.

• 'A' invokes setLocalDescription() method with its offer

• 'A' stringifies the offer and sends the offer to 'B'.

• 'B' invokes setRemoteRemote() method with A's offer, such that B's RTCPeerConnection object knows about A's setup.

• 'B' invokes createAnswer() method and successful callback of method is passed in local session Description as B's answer.

• 'B' set the answer as local description by invoking setLocalDescription() method. • 'B' sends back stringified answer to 'A' using signalling.

'A' sets up B's answers as the remote session description using setRemoteDescription () method. Signaling is used for initiating WEBRTC connection. This application uses WEBRTCIO as signaling server which is abstract library for WEBRTC. Socket.io library uses Web Socket API with fallbacks with full-duplex bi-direction message communication.

2) Exchanging Network Information: Internet Communications Engine (ICE) framework is used to connect two or more clients in public or NAT network. IP address and port address needs to be shared by the browser through the signaling server. Sequence of steps for exchanging network information is as follows. Consider two remote browser clients 'A' and 'B'. 'A' creates an RTCPeerConnection object instantiated by onicecandidate() handler (part of ICE framework) to find suitable network candidates. The handler is invoked when when network candidates become available.

In the handler 'A' send the stringified candidate data to 'B' using signaling channel. 'B' receives candidate message from 'A', 'B' invokes addICECandidate() method to add candidate to remote peer connection.

e. RTCDataChannel stream

This is a component of WEBRTC for streaming data. It is Used for handling signal information and data stream. This used Web Socket for connection since it support bi-direction full duplex connection.RTCDatachannel uses following methods to stream data. stream.getVideoTracks ()-For getting video with cam-era as input. stream.getAudioTracks ()-For getting audio with microphone as input. RTCDatachannel stream API contains constraints arguments which specifies camera facing, Codecs used, Resolution. RTCDataChannel API is used for stream data required for UserDiscoveryandcommunication, Signaling, Network addresses translation/traversal,Relay server in case of failure of video transmission.

4. Experimental Setup

As a part of our basic experiment, we run node.js server on COMPAQ 510 core2duo laptop with 3 GB RAM running UBUNTU 14.04 LTS operating system. We installed node package manager and its dependencies. We used DLINK wireless router to connect server laptop and two Samsung note2 android mobile devices with newest chrome browser installed on them supporting JavaScript V8 engine. We connected HP laptop to the network with camera and newest chrome browser to test our application on multiple platforms. The laptop server act as signaling server to make the IP of the user browsers

visible with in network. We installed a 'nload' application to analyze bandwidth of signaling server. The webrtc application is running as node.js application on IP address 192.168.0.103 and port address 8080.We installed Bandwidth Monitor application on android device to analyze bandwidth taken by application on client browsers.

In our Experiment, the devices are connected to server with room identification i.e. http://192.168.0.103:8080/#mbsl. When smart-phone devices and PC connects to the URL via chrome browsers the browser ask permission to access microphone and camera devices. After giving permissions, the two devices connected are able to do Video, Audio and text chat using the interface with high quality video.

A.Performance Analysis

Metrics	***Description***
Clientside requirements	Mobile, PC device with Javascript Enabled browser
Browser used	Chrome Version 43.0.2357.130 on mobile and PC
Video frame size	2 frames of 640X480 size
Connected number of users	4 users
Permissions	To access camera and microphone
Rooms	Two rooms 1. two android mobile devices. 2. one windows PC and one android mobile device.
Operation on Browser	Capture and transmit Audio, Video and text between two connected using a browser
Recording time period	20 minutes
Client average Bandwidth usage	Mobile : 112-250 kbps PC: 110-300 kbps
Application loading time	Varied(3-5 seconds)

TABLE IV .METRICS FOR EXPERIMENTAL SETUP

METRIC	
Min Bandwidth	0.00 kbps
Max Bandwidth	37.81 kbps
Avg Bandwidth	2.621 kbps
Total data Received	0.34 MB

TABLE V. SIGNALING SERVER INCOMING TRAFFIC

METRIC	
Min Bandwidth	0.00 kbps
Max Bandwidth	744 bps
Avg Bandwidth	120 bps
Total data Received	0.04 MB

TABLE VI..SIGNALING SERVER OUTGOING TRAFFIC

5. Conclusion

Complete web application can be developed using javascript frameworks like node.js. Minimal effort in developing code using node.js than traditional client-server application since implementation is in single programming language. Web Real time Communication (WebRTC) API enables to add video/audio chat services to our web application. As its browser application there is no need of external plug-ins. WebRTC API is available on major browser implementation. As part of our work, we have presented the architecture of WebRTC. Using WebRTC API we have designed and implemented a javascript based web application that can perform video, audio, text chat operations. Implemented signaling server which identifies IP of the browser to be connected and create a session. This application is cross-platform that can work on web browsers across tablets, smart phones, personal computers. This application can be used as private internal VoIP service. It is efficient and cost effective since it uses existing network. Quality of video and audio is good. This application can be used as internal communication services within a company and these services are cost-effective.

Integrating the application with other web services. Improvements in User Interface need to be done. Implementation of application with Multi-user support needs to be done. Compression techniques should be used to reduce the bandwidth usage of the application.

References

[1] Tilkov, Stefan, and Steve Vinoski. ”Node. js: Using JavaScript to build high-performance network programs.” IEEE Internet Computing 14.6 (2010): 0080-83.

[2] Johnston, Alan B., and Daniel C. Burnett. WebRTC: APIs and RTCWEB protocols of the HTML5 real-time web. Digital Codex LLC, 2012.

[3] Rescorla, Eric. ”WebRTC Security Architecture.” (2014). https://tools.ietf.org/html/draft-ietfrtcweb-security- March-07

[4]Flanagan,David.JavaScript:thedefinitiveguide.”O’Reilly Media,Inc.”, 2002.

[5] Goode, Bur. ”Voice over internet protocol (VoIP).” Proceedings of the IEEE 90.9 (2002): 1495-1517.

[6] Elleuch, Wajdi. ”Models for multimedia conference between browsers based on WebRTC.” Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th International Conference on. IEEE, 2013.

[7] Rodrguez Prez, Pedro, Javier Cervio Arriba, Irena Trajkovska, and Joaqun Salvacha Rodrguez. ”Advanced Videoconferencing based on WebRTC.” (2012): 180-184.

[8] McCune, Robert Ryan. ”Node. js Paradigms and Benchmarks.” STRIEGEL, GRAD OS F 11 (2011).

[9] Lerner, Reuven M. ”At the forge: Node. JS.” Linux Journal 2011, no. 205 (2011): 6.

[10] Amirante, Alessandro, Tobia Castaldi, Lorenzo Miniero, and Simon Pietro Romano. ”On the seamless interaction between webRTC browsers and SIP-based conferencing systems.” Communications Magazine, IEEE 51, no. 4 (2013): 42-47.

[11] Guha, Saikat, and Neil Daswani. An experimental study of the skype peer-to-peer voip system. Cornell University, 2005.

[12] Node Package manager documentation https://docs.npmjs.com/

[13] WEBRTC documentation https://developer.mozilla.org/en-US/docs /Web/Guide/API/WebRTC