WebRTC Implementation and Architecture Analysis

Kedar G. Pathare[1], Pushpanjali M. Chouragade[2]
M.Tech Scholar[1], Assistant Professor[2]
Department of Computer Science and Engineering
Government College of Engineering, Amravati
[1]kedarpathare@gmail.com, [2]pushpanjalic3@gmail.com

***Abstract*—Web browsers are becoming powerful day-by-day, with increased usage of interactive multimedia applications. Web Real Time Communications is a standard to enable real time multimedia feature in web browsers without the need of third party plugins. The media capabilities of WebRTC standards are state-of-the-art, with many new features. Features like supporting peer-to-peer interactive multimedia communications between web browsers. This paper analyzes implementation and architecture of web real time communication (WebRTC) API along with its limitations. It also analyzes the Trapezoid and Triangle model of WebRTC architecture.**

***Keywords*—WebRTC, peer-to-peer, real time communication, browser communication. API.**

I. Introduction

WebRTC is a set of standards from WC3 that will enable real time communications (RTC) on the web between browsers. Browsers like Firefox, Chrome and Opera natively support WebRTC. Using WebRTC one can make peer-to-peer calls, video chats, exchange files and share screens. WebRTC enables users to build apps with the help of HTML5 and JavaScript. Software tools are available to build very compelling desktop and mobile apps. Any connected device—computers, tablets, televisions can be WebRTC enabled, can become a communications device. These tools are empowering every user to build their own apps and include WebRTC features. With the adoption of WebRTC API in browsers, the web browsers will be able to communicate (peer-to-peer) with one another, and with WebSocket servers. Early days web browsers were required to download and install third party plugins, like flash player, for real time communications. Third party plugins does not provide security for data and applications are also not trustworthy and standardized. The compatibility for establishing a session is not assured between two peers, if the peers are using browser plugins from different vendors, or different browser vendors which prefers different plugin vendors that reduce the success rate of session establishment.

WebRTC supports browser-to-browser applications for voice calling, video chat and peer-to-peer file sharing without the requirement of either internal or external plugins. Web real time communications (WebRTC) is a specification for browsers to enable peer-to-peer communication. WebRTC is being standardized by W3C WebRTC and IETF RTCWEB working groups. IETF specifies the protocol level standards [5] while W3C specifies the implementation specific features in browsers. The real time communication of media is independent of the browser vendors and applications are platform independent. A WebRTC peer can establish a session with that of a non WebRTC peer through gateways, though the other peer does not support any of the standards specified by WebRTC standardization like compatible codecs or transport level security etc.

WebRTC aims to develop a communication framework for web browsers that work on different platforms and devices. It provides an API for JavaScript applications to establish, modify and terminate sessions. It supports peer-to-peer interactive multimedia communications like audio, text, video, data, games etc. WebRTC supports many real time and non real time use cases. WebRTC implements ICE to identify and establish connection with peers behind Network Address Translators (NAT) [8].

II. Related Work

WebRTC is an open source project for browser based real time communication released by Google in May 2011. This has been followed by ongoing work to standardize the relevant protocols in the IETF[4] and browser APIs in the W3C.[6] The W3C draft of WebRTC is a work in progress with advanced implementations in the Chrome and Firefox like browsers. The WebRTC API is based on preliminary work done in the WHATWG. It was referred to as the Connection Peer API, and pre standards concept implementation was created at Ericsson Labs. Previous W3C web standards allowed communication only between a browser and a server, WebRTC allows direct communication between browsers, without any server in the middle. This theoretically allows for the implementation of peer-to-peer algorithms such as Coolstreaming or GridMedia. The WebRTC API currently by Google Chrome and Mozilla Firefox consists of the following data communication functions :

- RTCPeerConnection: The RTCPeerConnection interface provides a connection between peers to exchange data.
- MediaStream: The MediaStream interface represents a stream of audio or video data.
- RTCDataChannel: The RTCDataChannel interface describes a full-duplex data connection between two nodes.

III. ARCHITECTURE

The classic web structural design are based on a client-server model, where browsers send an HTTP (Hypertext Transfer Protocol) request for content to the web server, which replies with a response containing the information requested. The server provides resources that are closely associated with an entity known by a URI (Uniform Resource Identifier) or URL (Uniform Resource Locator). In the web application situation, the server can embed some JavaScript code in the web page and sends back to the client. Such code can interact with browsers through standard JavaScript APIs and with users through the user interface.

A. WebRTC Architecture

WebRTC extends the client-server semantics by introducing a peer-to-peer communication model between browsers. The most general WebRTC architectural form draws its idea from the so-called SIP (Session Initiation Protocol) Trapezoid given in Figure 1.

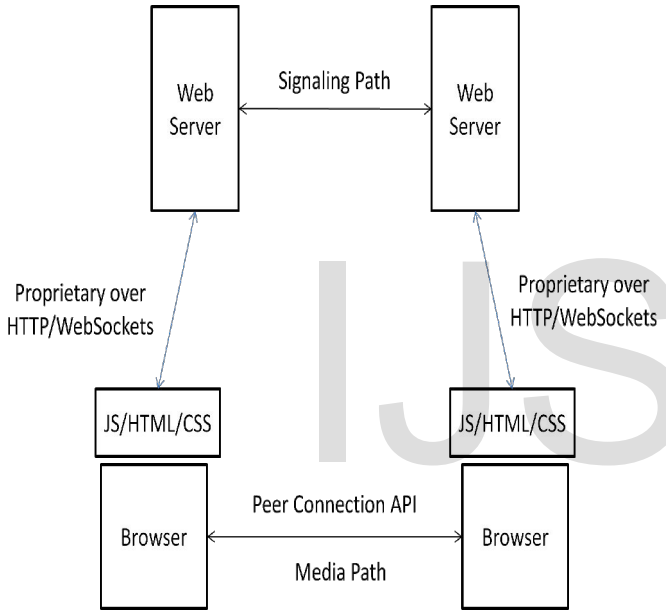

Fig. 1. The WebRTC Trapezoid Model

In the WebRTC Trapezoid model, both browsers are running a web application. Application is downloaded from a different web server. Signaling messages are used to set up and finish communications. They are transported by the HTTP or WebSocket protocol via web servers that can modify, translate, or manage them as needed. It is important to note that the signaling between browser and server is not standardized in WebRTC, as it is considered to be part of the application. As to the data path or media path, a peer connection allows media to transmit directly between browsers without any prevailing servers. The two web servers can communicate using a standard signaling protocol such as SIP or Jingle (XEP-0166). Otherwise, they can use a proprietary signaling protocol. The most common WebRTC scenario is likely to be the one where both browsers are running the same web application, downloaded from the same Web Server. In this case the Trapezoid model becomes a Triangle model as shown in Figure 2.

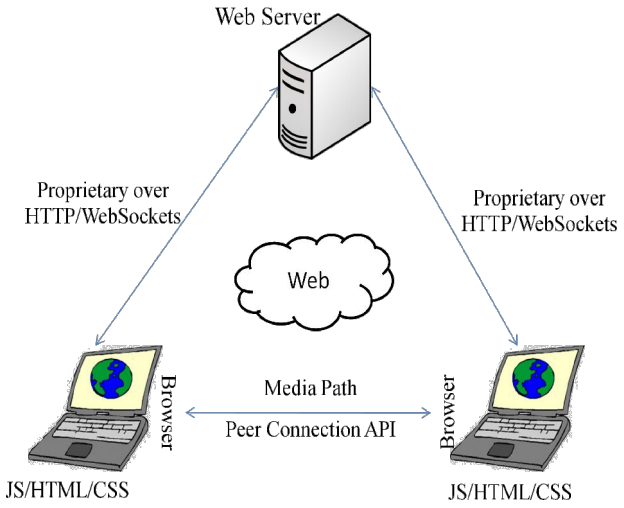

Fig. 2. The WebRTC Triangle Model

This arrangement is called a triangle due to the form of the signaling (sides of triangle) and media or data transmits (base of triangle) between the three elements. A Peer Connection establishes the communication for voice and video media and data channel transmits directly between the web browsers.

The connection between browsers sometimes referred as signaling but the connection is not really signaling as used in telephony systems. Signaling is not standardized in WebRTC. It is just considered part of the application. This signaling may run over WebSockets or HTTP to the same web server that serves web pages to the browser, or to a completely different web server that just handles the signaling.

B. WebRTC in Browser

A WebRTC web application (typically written as a mixture of HTML and JavaScript language) interacts with web browsers through the standardized WebRTC API, allowing it to properly utilize and control the real-time browser function as shown in Figure 3. The WebRTC web application also interacts with the browser, using both WebRTC and other standardized APIs, both proactively and reactively. The WebRTC API must therefore provide a wide set of functions, like connection management (in a peer-to-peer approach), encoding/decoding capabilities negotiation, selection and control, media control, firewall and NAT[8] element traversal, etc.

The design of the WebRTC API does represent a challenging issue. It envisages that a continuous, real-time flow of data is streamed across the network in order to allow direct communication between two browsers, with no further mediators along the path. This clearly represents a innovative approach to web-based communication. Let us imagine a real-time video or audio call between two browsers. Communication, in such a situation, might involve direct media streams between the two browsers, with the media path

negotiated and instantiated through a complex sequence of interactions.

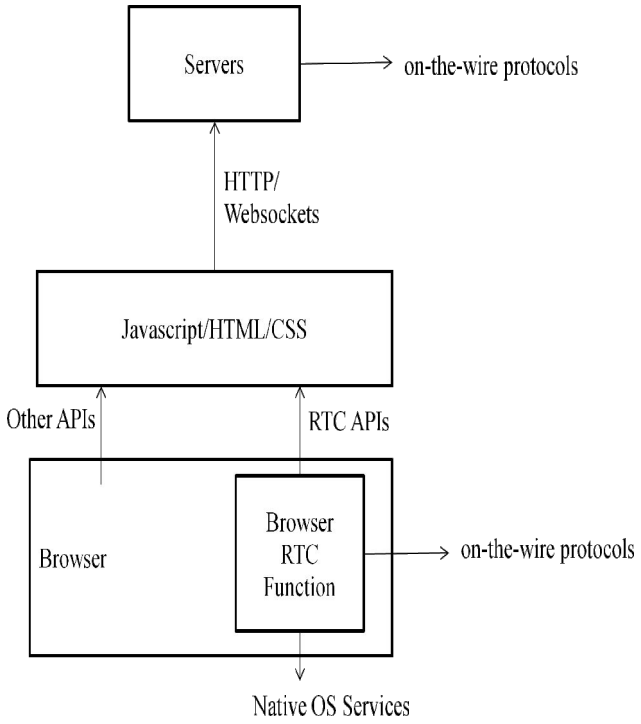

Fig. 3. Browser Model

Figure 3 shows the browser model and the role of the real-time communication function. The lighter block shows "Browser RTC Function". The unique nature and requirements of real-time communications means that adding and standardizing this block is non-trivial. The RTC function interacts with the web application using standard APIs. It communicates with the Operating System using the browser.

C. Signalling

WebRTC does not specify signaling methods to avoid redundancy and to maximize compatibility with established technologies. Unlike in native peer-to-peer applications where nodes can establish connections by contacting their peers directly, WebRTC requires the use of a separate signaling channel to negotiate a connection. The signaling process provides a level of security by eliminating the need for nodes to keep ports open, and allows creative routing strategies to enable peers behind NAT devices or firewalls to connect to each other. The exact nature of the signaling channel is not described by the WebRTC specification.

IV. Limitation of WebRTC

Currently WebRTC suffers from a number of limitations which are outlined in the following. However, while we were able to implement a prototype there were still multiple restrictions involved:

- There is currently a interoperability issue between browser. This led to implementation difficulties among browsers. For example the library PeerJS currently supports the Google Chrome browser only, because WebRTC is implemented in Mozilla Firefox differently.
- The browser implementations are currently in alpha or beta status and as a result have a number of bugs and may terminate unexpectedly.
- The WebRTC API does not yet offer functions for connection management and establishment. Instead, a second communication channel is necessary to establish a connection. We were using XmlHttpRequests and WebSockets to overcome this limitation.
- Another problem for WebRTC technology is the list of essential codecs. At the instant all the participating companies have come to the agreement only on one thing – WebRTC needs one main codec which will be supported by all browsers and thus will be cross platform.

V. Conclusion

WebRTC will help us to minimize the use of third party plugins, improve the security of the contents in browser. WebRTC is as innovatory a market disruption for telecom as HTML was for the internet. Companies that are willing to adopt this technology will have plentiful of business opportunities. Although, no browser has completely implemented the current WebRTC draft, it is likely that it will be implemented in future releases. With the many peer-to-peer applications and solutions that exist to date, WebRTC could greatly empower the web applications of the future.

References

[1] Kiran Kumar, Sachin Dev "WebRTC Implementation Analysis and Impact of Bundle Feature" 2015 IEEE DOI 10.1109/CSNT.2015.45

[2] C. Holmberg, S. Hankansson, G. Eriksson, "Web Real-Time Communication Use-Cases and Requirements", IETF draft-ietf-rtcwebuse-case-and-requirements-14, Febraury, 2014.

[3] Sam Dutton, "Getting Started with WebRTC", Febraury 2014, http://www.html5rocks.com/en/tutorials/webrtc/basics/.

[4] H.Alvestrand, "Overview: Real Time Protocols for Browser-based Applications", IETF draft-ietf-rtcweb-overview-11, August 2014.

[5] Berkvist, A., Burnett, D., Jennings, C. Narayanan, A. (2011) "WebRTC 1.0: Real-Time Communication Between Browsers". Working Draft.

[6] A Johnston, D. Burnett, "WebRTC: APIs and RTCWEB Protocols of the HTML5 Real-Time Web", Digital Codex, St. Louis, Mo, 2012

[7] A Johnston, D. Burnett, "WebRTC: The Web Way to Communicate", WebRTC IEEE St. Louis & COMSOC April 2013

[8] J. Rosenberg "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", IETF RFC 5245, April 2010.