World Academy of Science, Engineering and Technology
International Journal of Electronics and Communication Engineering
Vol:17, No:6, 2023

Designing and Implementation of a Method Comprising One to One Web-Based Real-Time Communications

Lata Kiran Dey, Rajendra Kumar, Biren Karmakar

***Abstract*—**Web Real-Time Communications is a collection of standards, and protocols, which provide real-time communications capabilities between web browsers and devices. This paper outlines the design and further implementation of a web real-time communications method on a secure web application having audio and video call capabilities. This proposed application may put up a system that will be able to work over both desktop as well as mobile browsers. Web Real-Time Communications (WebRTC) also gives a set of JavaScript standard Real-Time Communications (RTC) Application Programming Interfaces (APIs), which primarily work over the RTC framework. This helps to build a suitable communication application, which enables the audio, video, and message transfer between today's modern browsers having WebRTC support.

***Keywords*—**WebRTC, Session Initiation Protocol, SIP, RTC, JavaScript, Secure Real Time Protocol, SRTP, Secure Web Sockets, Browser.

I. Introduction

WebRTC is a new web technology that allows browser and mobile applications with functionalities such as audio/video calling, chat, P2P (peer-to-peer) file sharing, and all that without any additional third-party software or plugins [1]. It is an open standard that allows web developers to create RTC applications that work over the web. The primary components considered are real-time audio, video, and signalling. As we mainly focus here, on the technologies related to communications through the browser(s), WebRTC promises to add a huge set of RTC capabilities over the web, and also provide support for browser-to-browser RTC, without any further requirement of any unknown plugins or extension installation request [2].

WebRTC empowers myriad services, and is now an official standard, bringing audio and video communications anywhere on the Web, had been announced by The World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF) [3]. While the IETF is standardizing the signalling protocols and media technologies required in WebRTC, the W3C is standardizing the APIs and browsers for RTC [3]. The IETF and the W3C are tasked with bringing WebRTC among browsers to an acceptable level in both the industry and academia [4]. It was published as an opensource technology by Google in May 2011 and includes fundamental components for RTC on the web. With those components that can be accessed via the JavaScript API, developers can create impressive web applications. Google, Mozilla, and Opera support WebRTC and will participate in further developing this technology. At the moment, there are still platforms and browsers that either partially or do not support WebRTC [5]. WebRTC creates many opportunities in different domains, such as the Internet of Things (IoT), connected vehicles, healthcare, entertainment, surveillance, banking, and insurance, etc. Since WebRTC is used by high-profile companies, such as Google, Apple, Mozilla, and Microsoft, amongst others, it is expected a continuous growth of interest by developers and business executives [6].

The World Wide Web is emerging as a platform concerning communication services with the evolution of Web-based technologies such as hypertext support with HTML5, styling using CSS and scripting with JavaScript helps to make any web application design appealing and more as you think to create it. These provide the developers with a conductive application environment allowing them to deploy their services while maintaining the global standard. The APIs, amazingly help to make things possible, and enhances these technologies to provide a perfect user's web experience via dynamic user interfaces accessible over any browser-enabled device.

WebRTC uses Secure Real-time Transport Protocol (SRTP) to add encryption, message authentication, integrity, and replay attack protection for RTP data [7]. It is a security framework that provides confidentiality by encrypting the RTP payload and supporting origin authentication [7]. The security features of WebRTC are a crucial component of its reliability, and the basis for it all revolves around the Real-time Transport Protocol [7]. RTP uses the RTP Control Protocol (RTCP) to report the performance of the media stream. In this process, the media sender transmits encoded media encapsulated in RTP [7]. It also sends RTCP Sender Reports, which facilitate playback synchronization of different media streams [7]. The receiver maintains a jitter buffer to reorder media packets and play them out as per the timing information encoded in the packet [7]. If a packet is missing, the receiver recovers the packet or conceals the error [7]. Finally, the receiving endpoint reports rough or detailed statistics in the RTCP Receiver Report that enables the media sender to adapt its media encoding rate, change to a better codec, or vary the amount of forward error correction [7].

Major components of the WebRTC API are [8]:

- MediaStream – allows a web browser to access the camera and microphone [8];

Lata Kiran Dey is with C-DOT, India (e-mail: kirans.singy@gmail.com).

- RTCPeerConnection – sets up audio or video calls [8];
- RTCDataChannel – allows browsers to send data through peer-to-peer connections [8].

A. RTCWEB: A Standards Perspective

RTCs in Web browsers, also referred to as RTCWEB, are created by the IETF working group, which is in charge of defining the real-time protocols, their data formats, security, and all other elements required to make peer-to-peer conversations possible [9]. Real-time voice and video communication within browsers are among the topics currently gaining momentum in the two main Internet standardization bodies, the IETF and the W3C [10]. The vision of these two bodies is to standardize an open framework enabling seamless browser-to-browser multimedia applications [11].

II. Related Work

WebRTC is gaining more and more interest, and several relevant projects are currently under development. A large community of developers exchanges information, comments, suggestions, and announcements in discussions over WebRTC groups on Google Groups [10].

Before the existence of WebRTC, there were already many video conferencing systems available on the market [12]. The most popular example is Skype. Microsoft is the company that owns Skype [12]. Skype uses a proprietary protocol for the transmission of multimedia streams, and it requires the installation of a mobile application or desktop to access services such as phone calls, messages, etc [12]. The WebRTC architecture provides end-to-end encrypted P2P communication with audio-visual content and data being transmitted directly. The implementation is realized to bypass intermediary hardware servers and eliminate security challenges like hijackers [12]. This particular feature makes the difference between WebRTC and other RTCs such as Skype. Skype is proprietary and lacks the direct P2P ability, as well as a credible security feature found in WebRTC [12].

Peer-to-peer also enables users' data to be encrypted, safe, and cannot be compromised. A peer-to-peer connection can be credible because it can bypass all the problems associated with plug-ins. Factors such as latency, bandwidth, and memory utilization as well as support for anonymity are supported in WebRTC [13].

III. Proposed Work

Our main focus here is making a secure web communications application that works over the Internet and provides one-to-one audio and video calls. The concept is being put forward to make a system using the most recent versions of JavaScript, HTML, and standard libraries of WebRTC, with security mechanisms and standards of secure web sockets and media SRTP. Fig. 1 shows the call signalling flow between devices for WebRTC communications.

The purpose is to make a standard application to make a great user experience with fast communications and low-delay media. All the error handling has been done, considering the cases of media, messages, registration, deregistration, authentication, and authorization has been done as well. Fig. 2 shows flow chart of the work done.

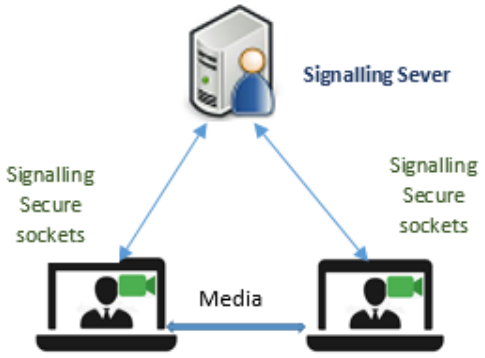

Fig. 1 Call signalling between devices

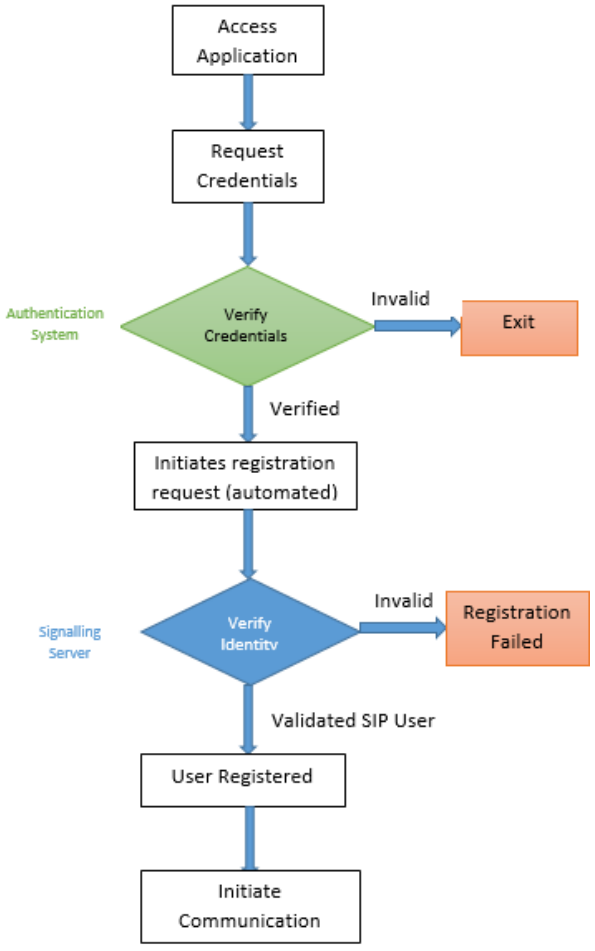

Fig. 2 Flowchart showing the system

A. Current Implementation

In working with WebRTC, our prime concern is on creating a RTC application that will work majorly making hand-to-hand with new technologies and supporting platforms. The major concerns over here are to design and later develop a system that can work on the concept to initiate and accept video, and audio calls, in addition to this; we also focus on the real-time data chat within the connected point-to-point browsers. This is caused by

the rising need for real-time applications, that can run on any platform and are hardware-independent, regardless of the operating system. To implement this, the criteria are also here not only to make end-to-end call sessions but also to provide a secure medium for the session and data. This will be done and achieved via the secure socket layers and the authentication & authorization set for the user. Now to elaborate on this, let's now see the proposed WebRTC framework that makes the session and hence the communications possible. Fig. 3 shows the connectivity between users-browsers and server(s) via Secure Sockets.

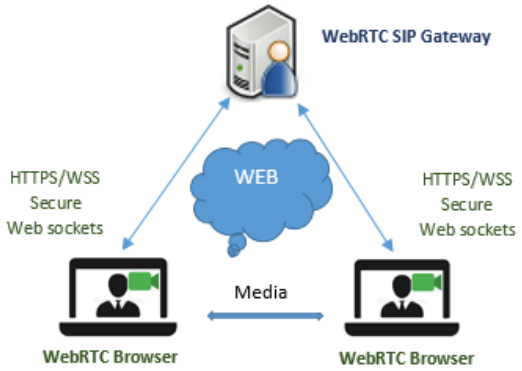

Fig. 3 WebRTC-User Connectivity via Secure Sockets

For these purposes, we planned and proposed an HTML-based web application with WebRTC support that initiates and ends the session initiation protocol (SIP) session over the web. To start accessing the application the user first needs to login into the application by entering the credentials. Certain security mechanisms were applied over here to consider the issues related to unauthorized resources access at the SIP gateway, which may lead to unwanted session requests, and data processing and may be ignoring the valid traffic data in the form of messages. In the first step, login to and access the calling feature, we provide the user with the sign-up/login into the application to generate/ verify the credentials.

On successful generation/verification of the user, the authorization steps will be done by the system based on permissions, and then an automated registration request will be sent to the registrar server. For user SIP registration purposes, a registration request will be sent internally via this system, and in response the user will be shown registered at the application. This registration process will be done via SIP gateway-supported WebRTC to the registrar server. The list of registered numbers that are displayed online from the user's address book can now be used to start an audio or video call. As the user clicks over the video/audio call button given over the user interface, the user agent will send an INVITE message request to the other user agent (receiver) endpoint which will be forwarded via the WebRTC SIP gateway. Once the SIP gateway has successfully confirmed that the selected user is already registered and prepared to receive the call, the call session will be started. If no registration-related information is found for other users, then an appropriate FAIL message following the SIP Transaction(s) will be sent to the originator user that initiated the call. In other cases, where the receiver is found registered and ready to accept the call it will send out the successful responses 180 RINGING and 200 OK SIP messages towards the originator user agent. As soon as the receiver sends a message confirming a successful call session, the originator will send a response ACK SIP message to the receiver to acknowledge the successful reception and acceptance of the previous messages. Fig. 4 displays the SIP WebRTC message sequence.

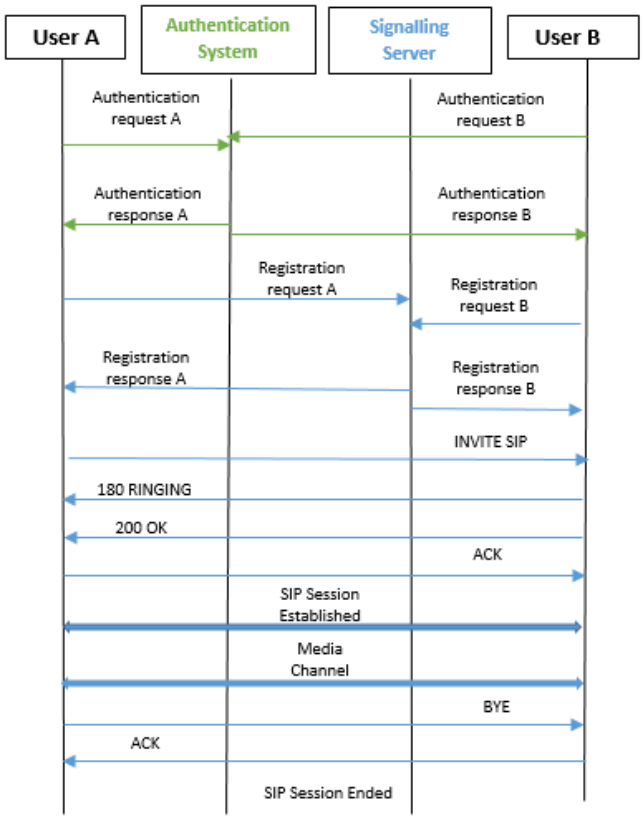

Fig. 4 SIP WebRTC Message Sequence

Finally, the user call session has been established. From this point on, the media will be started by the audio or video call media codecs listed in the call session messages. The offer and answer messages will follow the SDP (Session Description Protocol) format, which contains information about media type, CODECs, RTP (Real Time Protocol), RTCP (Real Time Control Protocol), and all connected properties that can be used in the media session. The media session will now be initiated and exchanged between the endpoint user agents (browsers). The MediaStream interface API will be applied here to stream the RTP packets of media content. A stream consists of several tracks, such as video or audio tracks [14]. Now, at any point in time, the End call may be initiated from either Originator or Receiver and then a BYE SIP message will be sent to the SIP Gateway. And on the reception of the BYE message, the SIP Gateway will clear the session data as well as media with the backend Services/Processes. And both the entertaining parties' browsers will also clear all their media and session data, as well

as browser and memory resources. And both points of browser application(s) will now retain the original online state.

B. Security: Analysis and POV

Whenever we talk about any web application over the Internet, one of the major concerns that came to mind is how secure this application carries the data. So here are the answers related to those queries. We are expressing our concern regarding the security features included in this proposed system or application and its associated solution(s) that provide a guard over user data or messages. The first one is the Authentication and Authorization for verification and validation of the user credentials and access rules when the user is trying to access the application. Secondly, the Secure Web Sockets are used for end-to-end points within web applications, and SIP Gateway. Third is the web client certificate that will be added here at the browser agent for secure communications, and most importantly the user browser certificate provided by the server. So majorly all these points needed to be provided within the application over the web to handle the different security issues.

VI. Conclusion

Finally, as we wrap up our explanation of this effort, we should mention that real-time communications with signalling and media performance were also taken into account for the future. The error cases that may occur will also be handled in this proposed system. The next step is media analysis in case of high network traffic, video packet handling & synchronization will also be done as future work. At last, majorly the security issues will be handled and tested.

References

[1] Branislav Sredojev, Dragan Samardzija and Dragan Posarac, WebRTC technology overview, and signaling solution design and implementation, 25-29 May 2015.

[2] Michael Adeyeyel, Ishmeal Makitla, and Thomas Fogwill, WebRTC using JSON via XMLHttpRequest and SIP over WebSocket Initial Signalling Overhead Findings, Conference: WEBIST 2013 - Proceedings of the 9th International Conference on Web Information Systems and Technologies, Aachen, Germany, 8-10 May 2013

[3] WebRTC Online Available: https://www.w3.org/2021/01/pressrelease-webrtc-rec.html.en

[4] Michael Adeyeye; Ishmeal Makitla; Thomas Fogwi, Determining the Signalling Overhead of two common WebRTC methods: JSON via XMLHttpRequest and SIP over WebSocket, 9-12 Sept. 2013.

[5] A. Sergiienko, "WebRTC Blueprints, develop your very own media applications and services using WebRTC," in Birmingham, Packt Publishing, ISBN 978-1-78398-310-0, May 2014.

[6] A. Morton, "Understanding WebRTC Adoption by Industry," 2018. Online Available: https://www.callstats.io/white-papers/understandingwebrtc-adoption-by-industry

[7] Explaining the Real-time Transport Protocol of SRTP for WebRTC, By callstats on April 12, 2018. Online Available: https://medium.com/callstatsio/explaining-the-real-time-transport-protocol-of-srtp-for-webrtc-2795e03d2d58?source=post_internal_links

[8] Madhura Deshpande, and Dr. S. P. Mohani, Integration of WebRTC with SIP – Current Trend. International Journal of Innovations in Engineering and Technology (IJIET), Volume 6 Issue 2 December 2015.

[9] Available Online at: https://hpbn.co/webrtc/

[10] Alessandro Amirante, Meetecho S.r.l. Tobia Castaldi, Lorenzo Miniero, and Simon Pietro Romano, University of Napoli Federico II, On the Seamless Interaction between WebRTC Browsers and SIP-Based Conferencing Systems,

[11] C. Holmberg, S. Hakansson, and G. Eriksson, "Web Real-Time Communication Use-Cases and Requirements," draft-ietf-rtcweb-use-cases-and-requirements10, Dec. 2012.

[12] Real-Time Communication and Multipoint Video Conference Using Webrtc and Media Server. Boobalan M Department of Computer Science and Engineering Sri Ramanujar Engineering College, Chennai-600127, India.

[13] Sanket Jadhav, Manoj Berad, Tejas Banaitkar, Swapnil Salunkhe, Swati Khokale, Survey on Real-Time Peer to Peer Multimedia Communication Application, © 2021 JETIR April 2021, Volume 8, Issue 4 www.jetir.org (ISSN-2349-5162).

[14] MediaStream, MDN web docs, Online Available: https://developer.mozilla.org/en-US/docs/Web/API/MediaStream