An Interactive Video Conferencing Module for e-Learning using WebRTC

Karl Bissereth, Billy B. L. Lim, Amit Shesh
School of IT, Illinois state University
Normal, IL, USA
kwbisse@ilstu.edu, bllim@ilstu.edu, ashesh@ilstu.edu

***Abstract*—Collaboration applications are becoming ubiquitous in corporate environment as well as in education sector. This paper examines WebRTC technology and its application to an e-learning platform to demonstrate how this new technology can be an alternative to traditional web conferencing that relies on plugins or third party software to deliver their services. Using an e-learning project prototype, it gives an overview of the WebRTC technology and highlights a signaling mechanism that is based on a full mesh topology to support multiple peer connections.**

Keywords—WebRTC, full mesh web conferencing, e-learning.

I. INTRODUCTION

The state of web conferencing has been constantly evolving ever since its inception decades ago. The integration of real-time communication via different web applications is presently supported using various proprietary plugins. As the Web transitions from static to more interactive and functional applications, it also influences the web conferencing field. In this context, efforts have been made to loosen the browser and web pages from the hold of plugins. This gives birth to the WebRTC API.

WebRTC stands for Web-based Real-Time Communication. It provides a web application the capability to directly share video and audio streaming between browsers based on their built-in real-time communications features, which are accessible via standard HTML and JavaScript tags [1]. The WebRTC standard is distinguished by two main visions. The first one is Rtcweb, which is the different interaction protocol that IETF (Internet Engineering Task Force) must address in order to assure interoperability with legacy systems. The second concept is defined by W3C (World Wide Web Consortium), known as WebRTC, that defines the different API that allows browsers and scripting languages to interact with media devices (microphones, webcams, and speakers) [2].

Noting the burgeoning WebRTC technology and its potential, this project aims to develop an application of the WebRTC API for learning purposes. With this goal in mind, a project that enables less expensive infrastructure for collaborative online classes is conceived. The project contains an interactive videoconference module for e-learning and it uses the WebRTC and NodeJS/Socket.io APIs to allow instructors to set up their virtual classes and share their knowledge with students using a virtual, online interaction.

The remainder of the paper is organized as follows. In the subsequent section, various existing real-time communication applications are discussed. Then, an overview of the WebRTC API is presented. Next, the architecture of the web application developed in this project, its different components, and the signaling mechanism used to exchange the session messages between the peers are described. Finally, the advantages and limitations of WebRTC are given, followed by conclusions and future work.

II. EXISTING REAL-TIME COMMUNICATION APPLICATIONS

The following presents a brief overview of various collaborating application approaches used.

BigBlueButton: BigBlueButton is an interactive e-learning environment developed by Richard Alam for the Carlton University Technology and Innovation Management department in 2007 [3]. BigBlueButton system uses red5, which is a powerful Java-based video and audio streaming that uses the real-time messaging protocol and servlet implementation (http://www.red5.org/). Therefore, BigBlueButton is mostly based on Java applet plugin to operate in a web browser.

WebEx and GoToMeeting: WebEx is a proprietary platform developed by Cisco that enables user collaboration. It is mostly used for webinars but can also be used for training purposes. GoToMeeting, by Citrix and a direct competitor of WebEx, offers also the same collaboration service. Both systems are proprietary and need additional plugin or software installation in order to run properly on the end user computer [4].

Visimeet: Visimeet offers a video conferencing desktop software solution. It is used in various higher education institutions including by professors of the School of Information Technology at Illinois State University for their online class sessions [5]. However, VisiMeet is only a desktop client and mobile software solution and does not have integrated web software. Therefore, enabling a synchronous learning environment using the browser only is not possible with VisiMeet.

Adobe Connect: Adobe Connect is a "web conferencing platform for web meetings, eLearning, and webinars. It powers mission critical web conferencing solutions end-to-end, on virtually any device, …" [6]. This popular platform, however, relies on flash player plugins to provide web real-time communication.

Other environments include Skype and Hangouts, which are two applications used among the e-learning community and enterprise environment. They are also based on proprietary plugins. Additionally, the user has to create a specific account (Microsoft account for Skype or Google account for Hangouts) in other to access the service.

Some of the companies above are starting to integrate WebRTC into their platforms. New iteration of BigBlueButton software is now adopting WebRTC to provide a better quality and low latency audio stream [7]. Microsoft is studying the possibility to make Skype compatible with WebRTC by implementing the ORTC API, which is similar to WebRTC but used different codec specification for voice and audio [8].

III. Overview of WebRTC

WebRTC is based on three main components [2]:

- The peer connection handles the peer-to-peer connection between the browsers by sharing the session object between the caller and the callee.
- The Media Stream API enables the browser to access the media devices like microphones, webcams and speakers.
- The Data Channel provides the transport capability to the browser.

In order for the WebRTC API to establish a peer-to-peer connection, it has to generate the session description protocol (SDP) that contains all the information about the different parameters set for the multimedia content [9]. The SDP is then exchanged between the peers using a signaling mechanism.

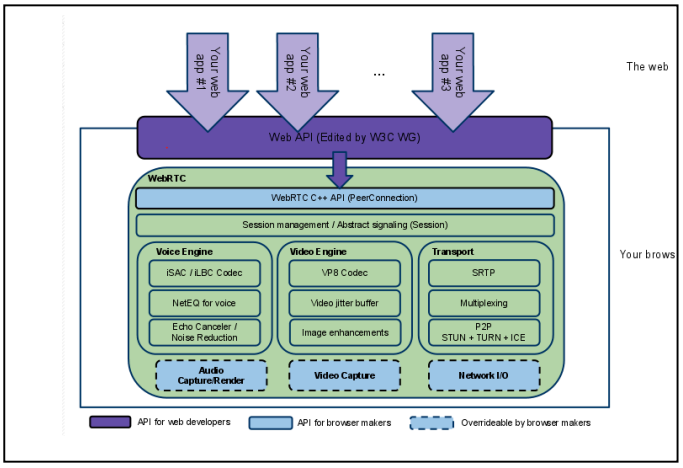

Fig. 1. WebRTC architecture (http://www.webrtc.org/reference/architecture)

The API uses an interactive connectivity exchange (ICE) process to inform the other peers about the network. The following diagram (Fig. 2) highlights the ICE mechanism for a peer-to-peer connection.

IV. Architecture of the Web Application

The web application developed as part of this project allows an instructor to setup a virtual classroom and enroll students to the virtual environment (manually or via an Excel

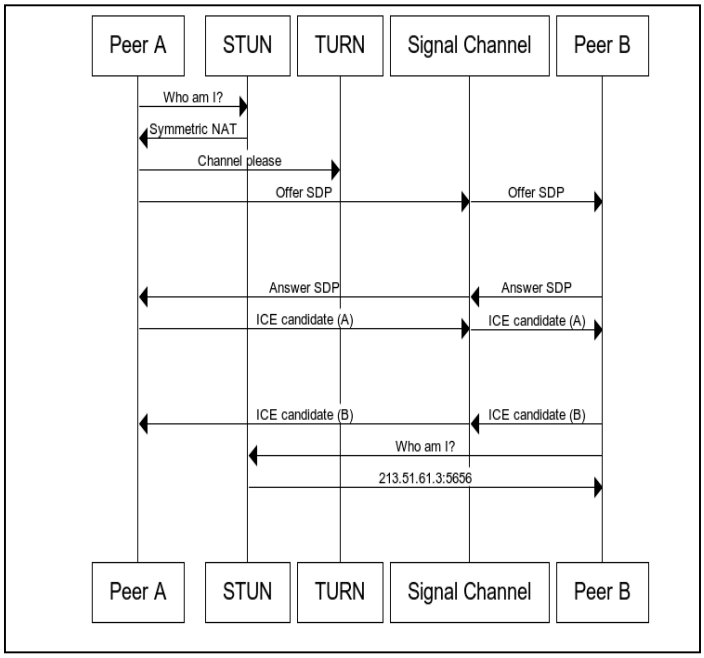

Fig. 2. WebRTC ICE diagram (https://developer.mozilla.org/en-US/docs/Web/Guide/API/WebRTC/WebRTC_architecture)

file import). Then, an instructor can initiate a virtual session and enrolled students can then participate in the session via videoconferencing with audio, video, and whiteboard support. PowerPoint slides as well as Prezi presentations can be used to share conferencing materials.

The architecture used is based on a full mesh logical design. This has the advantage of providing a lower media latency and higher quality [1]. In addition, the mesh design allows multiple peers to connect together using a transparent signaling server. Each participant connects to the web server to download the virtual room web application page. The page contains the WebRTC JavaScript API embedded. Fig. 3 describes a scenario were three peers are connecting.

A. *Components of the Application*

The application has three main components: (1) administrative component that uses Java/JSF technology to allow the instructor to setup the virtual classes and the registered students, (2) linking component that uses Java RESTful API to link the persistent part of the application to the signaling/socket.io server, and (3) persistent server component that stores all the settings saved by the instructor into the system. Fig. 4 illustrates how the components interact.

B. *The signaling Mechanism Application*

The signaling feature of the application is responsible for exchanging the SDP generated by each participant's browser. Node.JS and Socket.io are used to implement the signaling section.

Node.JS is a server platform that has JavaScript as its scripting language and it is built on top of the V8 JavaScript runtime by Google Chrome. It is a well-suited platform for building network web application [10]. An advantage of Node.JS is that it can handle high throughput and performance

since it is based on non-blocking I/O and event-driven model [11]. Socket.io is a WebSocket API that supports real-time approach system. The system is well integrated with the Node.JS server and provides a reliable TCP interconnection between the client's browser and server [12].

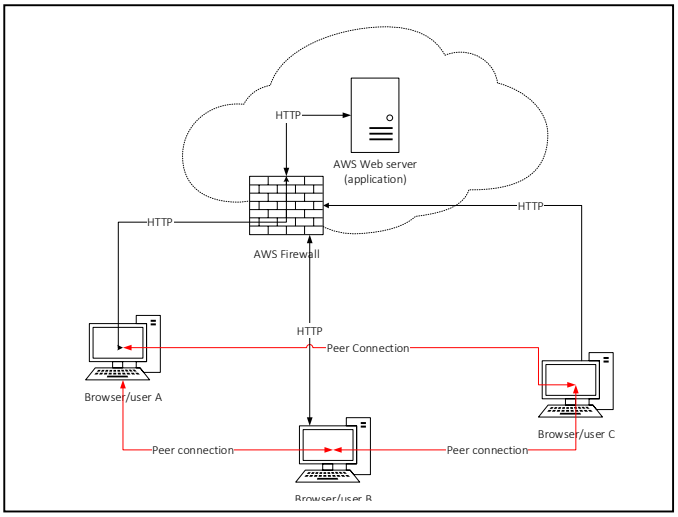

Fig. 3. WebRTC Multiple peer connection technology architecture

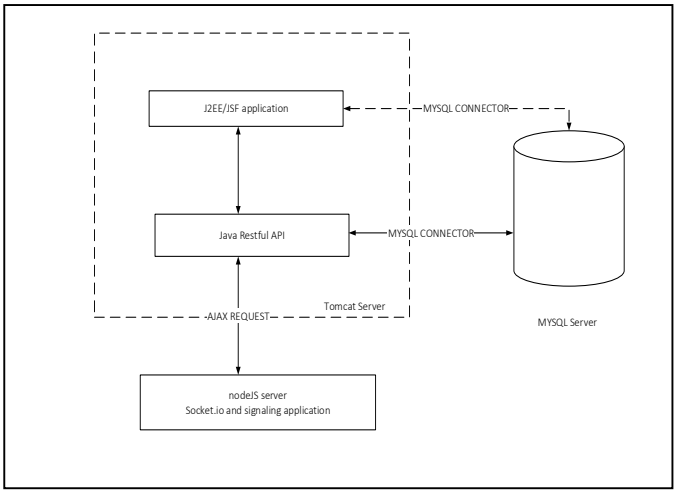

Fig. 4. Application Server Component

The signaling mechanism developed for this project is a JavaScript timer function that runs every minute and looks at the connection and room participant queue to initiate the peer connection process between browsers. When a new participant is connected, the signaling server retrieves the socket ID for the user and stores it in the connection and available-list queue. The signaling process then starts by requesting the SDP offer from the newly connected user. The user then sends back the SDP offer to the signaling server that shares it with the first available user in the room participant list. The user that receives the SDP offer generates an SDP answer and sends it to the signaling server with the receiver id field set with the offer socket ID. Finally, the signaling server upon receiving the SDP answer shares with the corresponding browser that matches the socket receiver ID.

After exchanging the SDP sessions, the signaling server becomes transparent to the browser connections. The browser will start to initiate the ICE exchange process, negotiate the authentication keys, and then establish a secure media. The signaling process loops until the connection queue is empty. Fig. 5 below summarizes the process.

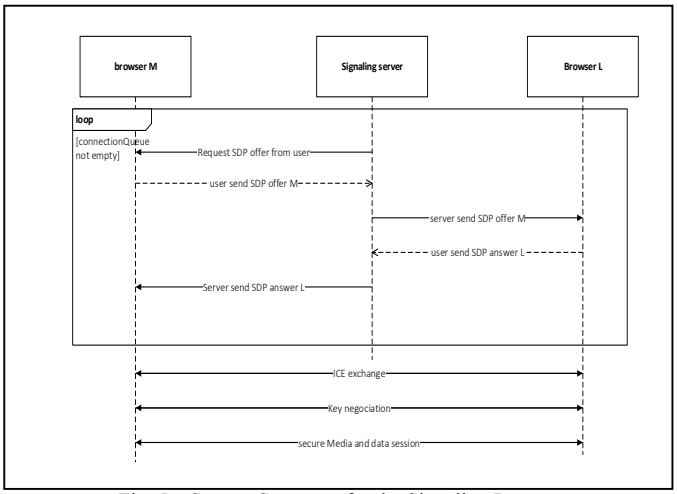

Fig. 5. System Sequence for the Signaling Process

Fig. 6 given below shows a snapshot of a virtual classroom session with whiteboard.

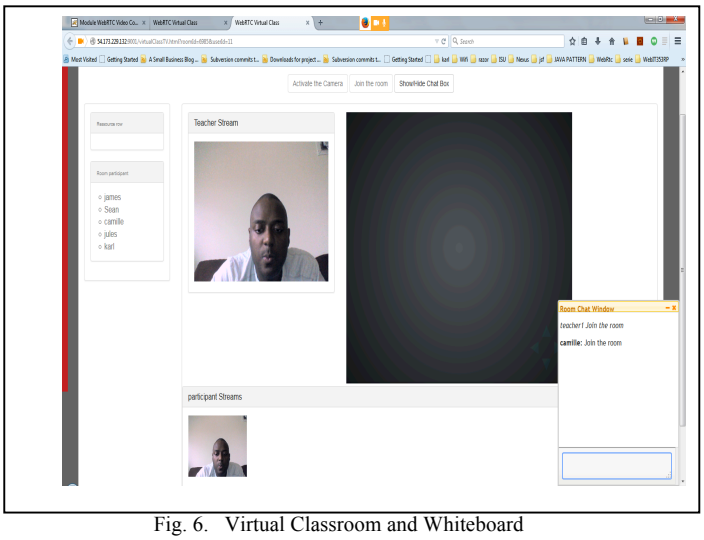

Fig. 6. Virtual Classroom and Whiteboard

V. DISCUSSIONS

The development of the project using WebRTC has proven to be a valuable alternative to the present web conferencing technology. First, the interconnection between browsers is quicker that the traditional web conferencing technology. This might be related to the fact that traditional web conferencing needs a more robust infrastructure to work with whereas WebRTC requires less. Also, there is no need for Media Server to compress the videos and share them. Everything is done at the user level, which reduces the connection delay processing time. For instance, a comparison with flash player

has shown that WebRTC has faster connection time, lower audio latency, and better audio quality [13].

Another major benefit from WebRTC is related to the direct communication between browsers after the SDP has been exchanged. This enables companies to save on infrastructure cost since the service will need fewer nodes to be deployed. Also, WebRTC uses a secure real-time transfer protocol (SRTP) to secure the video and audio media transportation, and this feature is implemented directly in the browser transport stack, which speeds up the transport process [1,14].

The development of this application creates a virtual environment where an instructor can communicate interactively with the enrolled students. This virtual interaction is supported by a whiteboard that is synchronized among all the participants of the room. The virtual classroom also has a presentation area where the slides are switched in real-time. Latency was experienced when more than three users join the room but since all the WebRTC API processing is done at the client level, this behavior should be expected.

One of the major limitations of WebRTC is that it is only implemented by two major browsers—Chrome and Firefox. This is a major constraint for the adoption of the technology since 43.3% of the desktop browser in the US is still using Internet Explorer [15]. Additionally, WebRTC is still an experimental technology that is undergoing major changes. This is slowing down the adoption by the business community, which is looking for more stable technology.

CPU processing constitutes another issue when using mesh solution. The reason is that the host computer needs to use a lot of bandwidth to upload and share the media with other peers and also the encoding process of the video and media stream requires a lot of CPU computations especially when this process is being done multiple times in parallel [16]. A mixer approach also known as a multipoint control unit can be a solution to this issue. But it removes the peer-to-peer characteristic of WebRTC and reverts back to the traditional solution used by legacy web conferencing applications. Additionally, it will add extra delay to the transmission since the encoding and decoding will be done by another node [16].

Lastly, troubleshooting can be difficult in an enterprise environment where the user needs to gather the logs from the WebRTC sessions and send them to the IT technicians. This can be difficult if the user is not technologically savvy.

VI. Conclusions and Future Work

In this paper we propose a mesh configuration for an interactive videoconference module that can be implemented with e-learning platforms. The application supports the participation of multiple users in a video conferencing session using multiple peer connections. We discuss the signaling mechanism that handles the exchange of the SDP message between multiple peers in a mesh connection. We also highlight some of the strengths and weaknesses of the approach used, namely the WebRTC technology.

In future iteration or adoption of WebRTC for collaboration application, it is best to use a centralized topology. This will certainly fix the issue of the high CPU process but the drawback of adding delay to the connection establishment can be expected. Also, improvements still need to be done at the echo and noise cancellation level.

WebRTC and nodeJS/Socket.io combined offer a great solution for real-time application development. Even though the API is still evolving and not yet adopted by all the major browsers, it represents an alternative to all the legacy videoconference applications and makes the setup for the video and audio session easy. Additionally, the implementation of this technology can lead to significant network and infrastructure cost reductions.

References

[1] Johnston, A., & Burnett, D. (2014). *WebRTC: APIs and RTCWEB protocols of the HTML5 real-time web* (3rd ed.). St. Louis, MO, Digital Codex LLC.

[2] Loreto, S., & Romano, s. P. (2012). Real-Time Communications in the Web, *IEEE Computer society*.

[3] BigBlueButton. (2014). *History of BigBlueButton*. Retrieved from http://bigbluebutton.org/history/

[4] Heller, M. (2013, October 25). *Review: WebEx vs. GoToMeeting vs. MyTrueCloud*. Retrieved from http://www.infoworld.com/d/applications/review-webex-vs-gotomeeting-vs-mytruecloud-229198?page=0,0

[5] IOCOM. (n.d.). *PROFESSORS AT ILLINOIS STATE UNIVERSITY USE VISIMEET*. Retrieved from http://janet.iocom.co.uk/docs/ISU_case_study.pdf

[6] "Adobe Connect" Adobe Web Conferencing Software, Video Conferencing, Webinars. Accessed December 12, 2014. http://www.adobe.com/products/adobeconnect.html.

[7] "News." BigBluebutton RSS. Accessed December 12, 2014. http://bigbluebutton.org/2014/10/16/bigbluebutton-0-9-0-beta-now-available/.

[8] "Bringing Interoperable Real-Time Communications to The Web." Skype Blogs. Accessed December 12, 2014. http://blogs.skype.com/2014/10/27/bringing-interoperable-real-time-communications-to-the-web/.

[9] "Abstract." SDP for the WebRTC. Accessed December 12, 2014. http://tools.ietf.org/id/draft-nandakumar-rtcweb-sdp-01.html

[10] "Node.js." Node.js. Accessed December 12, 2014. http://nodejs.org/.

[11] Buasri, Nattha, Tanasak Janpan, Ularn Yamborisut, and Damras Wongsawang. (2014). "Web-based Interactive Virtual Classroom using HTML5-based Technology."

[12] "WebSocket and Socket.IO." David Walsh Blog Atom. Accessed December 12, 2014. http://davidwalsh.name/websocket.

[13] "Blog." Why Should Conferencing Service Providers Embrace WebRTC? Accessed December 12, 2014. http://thisisdrum.com/blog/2012/11/09/why-should-conferencing-service-providers-embrace-webrtc/.

[14] "High Performance Browser Networking." High Performance Browser Networking. Accessed December 12, 2014. http://chimera.labs.oreilly.com/books/1230000000545/ch18.html#_delivering_media_and_application_data.

[15] Christie, Joel. (2014) "Google Chrome Overtakes Internet Explorer for the First Time as the Most-used Web Browser in the U.S." Mail Online. June 8, 2014. Accessed December 12, 2014. http://www.dailymail.co.uk/news/article-2651645/Google-Chrome-overtakes-Internet-Explorer-time-used-web-browser-U-S.html.

[16] "WebRTC beyond One-to-one Communication (Gustavo Garcia Bernardo) - WebrtcHacks." WebrtcHacks. Accessed December 12, 2014. https://webrtchacks.com/webrtc-beyond-one-one