WebRTC Technology

A Video Chat application

CS297 Report

Presented to

Dr. Chris Pollett

Department of Computer Science

San Jose State University

In Partial Fulfilment

Of the Requirement for the Class CS297

By

Yangcha K. Ho

Feb 19, 2019

Introduction

WebRTC (Web Real-Time Communication) is a framework that enables peer to peer connections. It allows voice, video and data to work collectively in the browser without depending on third party plugins. The technology uses a collection of standards, protocols, and JavaScript APIs and HTML5. The video tag of HTML5 and getUserMedia function allow video stream to pass through between two browsers. RTCPeerConnection API allows communication between two browsers possible once both initial contact between them is established. This paper presents the major process of building video Chat application using WebRTC technology and it demonstrates that a WebRTC application can be built without plug-in third party software.

When two browsers want to set up a communication between them, they need a bridge which helps to communicate between them. The role of a signaling server is this bridge, but the WebRTC has not defined any specification as to how this signaling server should be set up. It is up to developers' choice to choose necessary components and put them together. This lack of specification on signaling server makes developers baffled with too many options without information. It seems a very simple concept to set up a communication between two browsers, but its implementation requires some work. The experience gained from building a videoChat application surely helps me to work on the final project which is porting the videoChat application into Yioop.com

Deliverable 1

The objective of deliverable 1 is to be familiar with Yioop.com application on which my final project CS298 – WebRTC video chat application will be ported to.

The first part of my work is to download Yioop source code to my desktop using “git clone https://seekquarry.com/git/yioop.git". The next task is to be familiar with various features of Yioop and know how source codes are being used to generate various social groups. As a learning curve, I generated several discussion groups to see how they appear on the page and where those code functions are called from.

I tracked the source code from the main index.php and followed through each function called to determine what the next call functions might ber until code groups appear. My findings are as follows:

From the main index.php page the following links are used:

1. The main index.php which calls function passthruYioopRequest()
2. Functrion passthruYioopRequest() invokes src/index.php and
3. Calls bootstrap function
4. bootstrap function calls following functions one by one
    - "/library/LocaleFunctions.php" which Calls : configs/Config.php";
    - "/library/UpgradeFunctions.php";
    - /library/VersionFunctions.php"; which sets group info and set DB data and
    - Calls basic.js
    - "/library/Utility.php";
6. Finally basic.js handles code groups

Deliverable 2

The purpose of deliverable 2 is to be familiar with Php language and sqlite3 databases. These two tools will be used in Yioop.com where my final CS298 project will be ported to. I wrote Sign in Gym membership application using Javascript program, Php and Sqlite3 database. The program allows users can sign in Gym membership application or can drop being a gym membership any time. Once user signs in, the application has an option to list all the members currently on the system and also what equipment the user usually works on. The user can select which equipment he/she wants to use. The user information and her choice of equipment are kept in sqlite3 database called "myGym.db" and the application has an option to list, edit, or drop information on database.

The first screen is a blank screen where a name and an equipment fields are blank, and user is expected to fill in them. Below is a screenshot of the log in page:

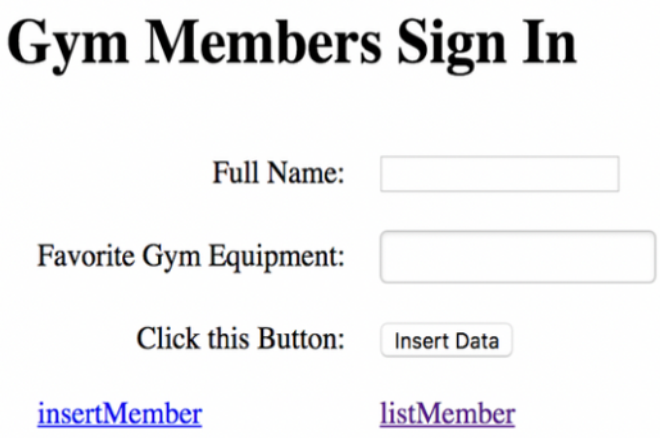

Fig 1: The first screen when you run the program

Deliverable 3

The purpose of this stage is to write a simple video chat application as a first experiment using a WebRTC technology. I used video tag of Html5 and getUserMedia function to capture video stream data. The WebRTC technology uses three APIs such as

1. MediaStream - allows the client (e.g., the web browser) to access the stream, such as the one from a WebCam or microphone;
2. RTCPeerConnection - enable audio or video data transfer, with support for encryption and bandwidth management;
3. RTCDataChannel - enables peer-to-peer communication for any generic data.

A sample html page using getUserMedia function to specify two video tags would look like this:

```
<html>
 <video id="localVideo" autoplay muted></video>
 <video id="remoteVideo" autoplay></video>
 <script src="main.js"></script>
 ...
     <scripts>
var constraints = {
    video: true,
    audio: true,
};

if(navigator.getUserMedia) {
```

```
    navigator.getUserMedia(constraints, getUserMediaOK,
        getUserMediaFailed);
    } else {
    alert('Your browser does not support getUserMedia API');
    }

   function getUserMediaOK (stream) {
      window.stream = stream; // make stream available to console
      videoElement.srcObject = stream;
  }
  function getUserMediaFailed (error) {
  console.error('Error: ', error);
}
</script>
```

RTCPeerConnection API is the core of WebRTC connection and is able to handle an array of URL objects of STUN and TURN servers, and through which the ICE candidate can find each other.

Deliverable 4

The purpose of deliverable 4 is to write a patch that improves some aspect of Discussion Groups in Yioop. For this part, I have to study the codes involved in Discussion Groups in Yioop.com. I found out that when users enter a long line of text without a line break, the resulting text reflects the same input as entered without rending a wrap up. This makes hard to read since the page goes over the regular length. I include a patch using search.css at which I specify

```
.overflow-item
{
   max-width: 5.1in;
   height:3in;
   overflow:scroll;
}

.mobile .overflow-item
{
    left: 5.1in;
    min-width: 2in;
    position: absolute;
    top: 0.55in;
}
```

This patch allows a long text automatically wrapped based on the line length and a corresponding patch has been submitted through mantis Bug Tracker system.

Conclusion

WebRTC is a pretty new technology with its APIs still being in progress. It allows audio and video communication to work inside web pages by allowing direct peer to peer communication without the need to install third party plugins or download native apps. There are a few browsers such as Chrome, Firefox, and Opera support this technology. This paper covers one scenario of WebRTC technology, video call at which two web browsers can share a simple video call using WebRTC. In general, a WebRTC-enabled application needs to take of several things such as connecting users, finding candidate with each other, negotiating media sessions, etc. One of the HTML5 tag “video” and getUserMedia() method allows video stream to pass between browsers and RTCPeerConnection APIs plays an important role to make browser connection possible. The getUserMedia() mthod of WebRTC requires security permission to use Webcam or audio features. WebRTC prohibits using self-signed security socket layer(ssl) certs and we need to generate correct SSL cert for the domain name as well. This implies that we need some public server such as AWS or other host for this application.

I found that Amazon Web Service (AWS) is a good place to run this application. One handy thing about AWS is that it allows to create a domain name along with an option to create SSL certificate for users.

References

1. https://appr.tc/
2. http://subnets.ru/books/Getting-Started-with-Webrtc-2013-Rob-Manson.pdf
3. http://davekilian.com/webrtc-the-hard-way.html
4. https://www.html5rocks.com/en/tutorials/webrtc/infrastructure/
5. Sergiienko, Andrii. WebRTC Cookbook. Packt Publishing Ltd, 2015.
6. Sergiienko, Andrii. WebRTC Blueprints. Packt Publishing Ltd, 2014.
7. https://webrtc.github.io/samples/src/content/peerconnection/pc1/
8. https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_servers
9. https://web-engineering.info/node/57