IJCSN International Journal of Computer Science and Network, Volume 4, Issue 2, April 2015
ISSN (Online) : 2277-5420 www.IJCSN.org
Impact Factor: 0.417

Browser-based Video Communication using WebRTC

[1] **Naman Avasthi,** [2] **Palakh Mignonne Jude,** [3] **Rhea Thomas,** [4] **Rahul Jadhav**

[1, 2, 3, 4] Computer Department, Fr. C. R. Institute of Technology
Navi Mumbai, 400703, India

Abstract - Communication is a very important part of life. Over the years, the manner in which people communicate has steadily become better, from the older telephones to the modern smartphones. Along with this, progress has been made from just voice communication to video communication. With the advent of the internet, video communication in real time has become a reality. We have developed a web application using WebRTC that permits voice as well as video communication, maybe on a one-to-one basis or among multiple users. Due to the browser-based nature of the application, it is device independent and can run on a variety of devices. This makes it platform-independent. Additionally, the web application ensures the security of the data being transmitted.

Keywords - *Video communication, WebRTC, videoconference, videotelephony.*

1. Introduction

We have created a web application that allows video communication between two and more than two users over the internet. The application requires only browsers, no other plug-in or software has to be installed, making the application platform independent. To use the application, the user has to first visit the website, if the user has a Facebook account, he can login using his Facebook credentials, else he will have to follow the normal signup process. For Facebook users, their Facebook friends who are also using are application will be automatically added as their contacts. The users can search for other users and add them as friends by sending a friend request.

Our application allows the user to carry out, a video call, which is one to one, or a voice call which includes only audio or a multi-chat, in which more than two users can participate. Our application uses OAuth for Facebook login and Web Real Time Communication for video communication (WebRTC).

Video communication initially started out in the form of Videotelephony. Videotelephony is an extension to the telephone; it is a way of simultaneous, two-way communication including both audio and video components. Users can both hear and see each other in real time [3].

WebRTC simplifies the process of video communication. It is a set of JavaScript APIs, protocols and standards by Google, drafted by W3C that allows browser-based communication and data transfer. WebRTC is open source.

The standardization of WebRTC is still going on, advanced implementations have mainly been done in Mozilla Firefox and Google Chrome browsers [1]. Presently, desktop browsers, Google Chrome (version 23 onwards), Mozilla Firefox (version 22 onwards), Opera (version 18 onwards) and mobile platforms like Android support WebRTC.

Videoconferencing is a means of establishing a live connection between two or more than two users at different geographic locations, using computer networks for the transmission of audio and video data [5].

2. WebRTC

Web Real Time Communication is a set of open source, JavaScript APIs, standards and protocols, drafted by W3C and developed by Google. WebRTC allows browser-based peer-to-peer audio, video communication and data sharing. It does not depend on third-party plug-ins or exclusive software. Allowing real-time communication in the browser is definitely one of the most important add-ons to the web, since the start.

WebRTC diverts from the traditional client-server model, the result of which is, complete re-engineering of the

networking layer in the browser, and also the addition of a new media stack required to enable efficient, real-time processing of audio and video [2]. The advantages of using WebRTC are as follows, it does not depend on any software or plug-in it is completely browser-based. It is open source and hence no royalties have to be paid to Google. It makes the job simple for developers and the best audio and video engines have been added by Google.

The only drawback of WebRTC is that it is a standard still under development; the code can go through critical changes in the near future. The details of the WebRTC API with respect to implementation are as follows. PeerConnection is used to create a connection with a peer. It takes in data about which servers to use and choices for the type of connection.

Interactive Connectivity Establishment (ICE) is a framework to allow the web browser to connect with peers, this framework locates candidates, to enable the user to connect with the peer, this is called ICE candidate. Signal Channel is a method used to send metadata and ICE candidate to the peer. In our application we have used Firebase as the Signal Channel. The metadata that explains to the other peer the format to expect (codecs, resolution, size, video etc) is called an Offer SDP (Session Description Protocol).

For an exchange to occur between peers, one peer must send an offer and the other peer should receive it and reply with an answer. The Offer is sent through the Signal Channel. An Answer SDP is a response; like answering the phone. An answer can only be generated once the offer has been received [4].

3. OAuth

OAuth is an open standard for authorization that provides a secure access to the server resources on behalf of the resource owner. This is done without sharing the credentials to the server via a third part application.

Facebook provides developers with its own OAuth 2.0 login process. This enables users to login/signup to/for our web application without the need to create a new account. The login flow also gives users the right to grant extra permissions for a better and customized experience.

Facebook uses Graph APIs, a low-level HTTP-based API that is used in our web application to query Facebook for user details provided in the access permissions.

4. Scope

The scope of our web application is described in this section. It enables voice and video communication between two and more than two users. The users will be allowed to login through Facebook. The login module makes use of OAuth 2.0. We have established a connection between the users using WebRTC (via web browsers such as Google Chrome, Firefox and Opera). Our web application is secure and reliable [1]. The use of WebRTC ensures that we send the data over secure data channels thus preventing anyone from gaining access to the data being transmitted.

5. Design

The block diagram consists of four main parts, the back-end, the front end, users and the WebRTC module. The users connect to our system through the front end. They make use of our web application via web browsers. These browsers must support WebRTC (Google Chrome, Opera and Firefox). WebRTC comprises of getUserMedia, RTCPeerConnection and RTCDataChannel APIs. Through these APIs the caller can place a call to the callee. Additionally, there is a back-end which contains data about all the users who have registered with our system.

We have used Facebook OAuth to allow users to log into our web application. The primary purpose of OAuth is to let applications make use of already existing login credentials of users to sign-in to their application without the trouble of creating one of their own for each application. This is good for developers as the process is made secure on the OAuth resource provider's end and has to only be implemented in the right way.

OAuth login (via Facebook) does all the tasks necessary for any normal login such as checks for unique users (in this case unique email IDs), no user is a bot (CAPTCHA checks) and secure communication while going through the login process via their own login flow. All these features makes it a great choice for developers as it lets them spend more time developing the core application.

As mentioned earlier, our application makes use of the getUserMedia and RTCPeerConnection APIs. The getUserMedia API allows access to the user’s webcam and microphone.

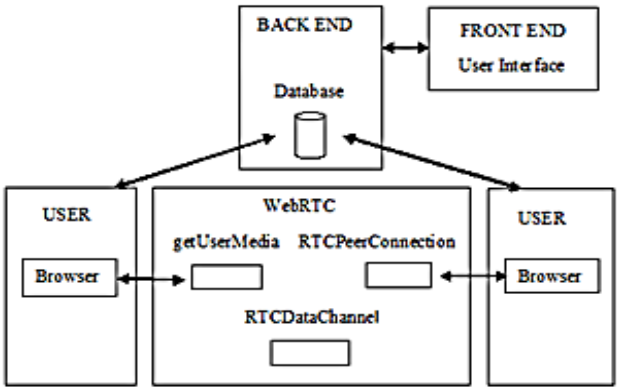


Fig. 1 Block Diagram.

The connection channels implement Secure Real-time Transport Protocol (SRTP) and Datagram Transport Layer Security (DTLS). RTCPeerConnection is used to setup the call. This API provides NAT traversal using STUN/TURN servers. The Session Traversal Utilities for NAT (STUN) and Traversal Using Relays around NAT (TURN) are used to help the data streams traverse the NAT boxes and firewalls

The security is maintained in the following manner. The packets are encrypted using DTLS which is an end-to-end encryption between peers. [1]

The following table gives a comparison between our system and the existing systems.

Table 1: Comparison between our system and existing systems

Parameter	***Skype***	***Proposed System***
Protocol used	Proprietary VoIP-Skype Protocol	WebRTC APIs
Features	Skype-to-Skype calls, Calls to mobiles and landlines, Group calls, Voice messages, Screen Sharing	Video/voice calls to-and-from any device that is browser-enabled. Allows video conferencing
Security	Not very secure.	Makes use of encryption algorithms

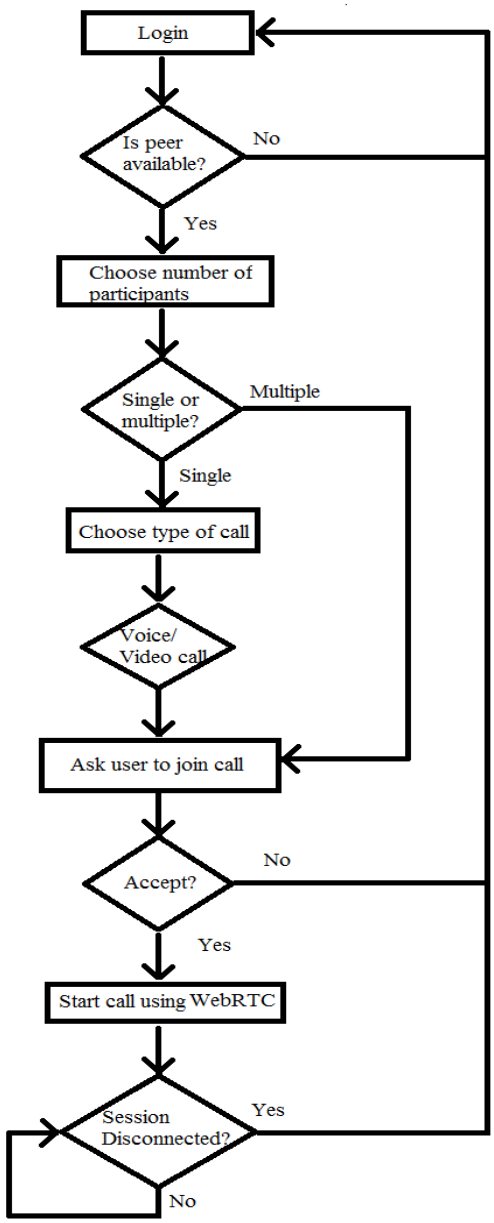


Fig. 2 Flowchart.

The flowchart illustrates the flow of the system's functions. The application permits both one-to-one (single) as well as multiple user communication.

The user needs to log into the web application in order to avail the services provided by it. Once this is done, the user can choose one or more peers to communicate with; provided that these peers are available. The user then chooses the type of call, this maybe a voice call or a video call.

In one-to-one communication, there is a sender, who initiates the call. The sender then waits for the receiver to accept or decline the call. If the receiver accepts the call, the call is set up using WebRTC. Alternatively, if the receiver rejects the call, the sender is notified and the sender can either call back later, choose to call someone else or logout.

In case of multiple user communication, a call request is sent to multiple users. The sender waits for the other users to join the conference call. In this case, as the users join, the conference call proceeds. .

The last part of the flowchart indicates the ending of a call. In a one-to-one call, if either the sender or the receiver ends the call, the session ends. On the other hand, in a multiple user call, the call continues till all the participants leave the call.

6. Implementation

The implementation can be divided into three main modules: the login process (using OAuth 2.0), searching for friends and adding them, placing a call and then key generation.

6.1 Login Process

The login process illustrates how the user enters into our web application. This is a very important step and no calls can be placed before the user logs in.

The Landing Page: The index page is the first landing page in our application. The base landing page has two options, one for a Facebook Login using Oauth2.0 and the other being a normal login system.

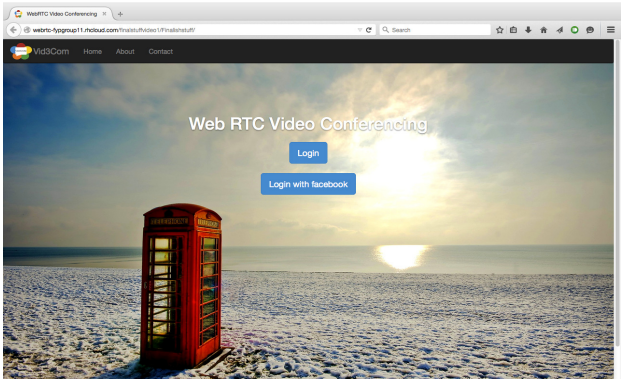


Fig. 3 The Landing Page.

This page contains the Facebook initialisation phase, this includes importing all necessary FacebookGraph APIs, defining the app ID and app secret, along with the redirect URL, this creates a Facebook object called $helper and creates a Facebook session $sess, using $helper and the redirect URL. It also checks if a Facebook session has been created or not.

```
//3.Initialize application, create helper object and get fb sess
FacebookSession::setDefaultApplication($app_id,$app_secret);
$helper = new FacebookRedirectLoginHelper($redirect_url);
$sess = $helper->getSessionFromRedirect();
```

Fig. 4 Initializing variables.

Log in with Facebook: This page asks the user for their Facebook login details. Here as we are using Oauth2.0 our application won't be aware of the credentials, it is handled by the Facebook APIs. Our application waits for the access token from Facebook.

Granting Permissions: Once authenticated, they will be redirected to the applications permissions page. Here the user will be prompted to allow the application to gain the following access rights:

1. Public profile (required) which includes User name, Profile Picture, Gender, etc.
2. Friend List
3. Email address: This is explicitly asked for as the email address is used for identifying a user in our database uniquely (primary key).

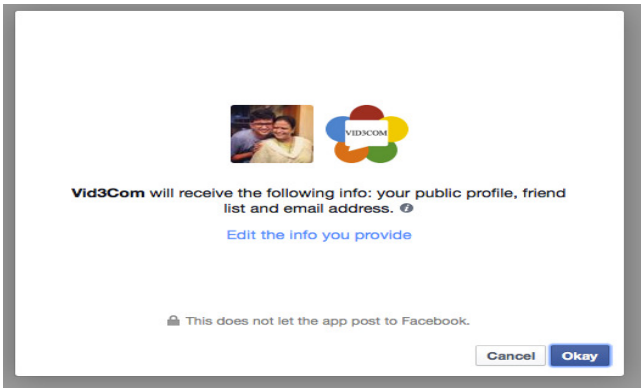


Fig. 5 Granting Permissions.

The Homepage: If the user allows the required permissions the Facebook OAuth2.0 process will be a success and Facebook will return our application an access token. On receiving this the Facebook session is created. The (index page) will now store the following information into a cookie (named 'userdata'), request the basic profile of the user, Full name, First name, Last name, Email ID, Friend List, Gender, Facebook ID-

This is a unique Facebook user ID for all its users, this is used for gaining access to the users Profile picture. Used as a reference in profile picture API call, Facebook Profile Picture. After this, all this data is queried to the database.

```
$first_name = $graph->getFirstName();
$last_name = $graph->getLastName();
$email = $graph->getProperty('email');
```

Fig. 6 Accessing data from the token.

After this is completed, the user is redirected to the Home Page where he can start using the application.

Fig. 7 The Homepage.

6.2 Searching for Users and Adding Friends

This section describes about the process to search for users among the existing users of the system. It also describes how to add these friends.

Searching For Users: The searching functionality is used in the following cases:

a. If someone does not have a Facebook account, he will have to follow the normal signup and login process; in this case the user does not have any friends hence he will have to search for other users to add as friends.
b. If a user has logged on using Facebook, all his Facebook friends who are also using our application will be displayed on the user's profile page, besides this, it is possible that the user may want to search for a new user who is not his Facebook friend.

In both the cases mentioned above, the user utilizes the search bar on the top right of the page.

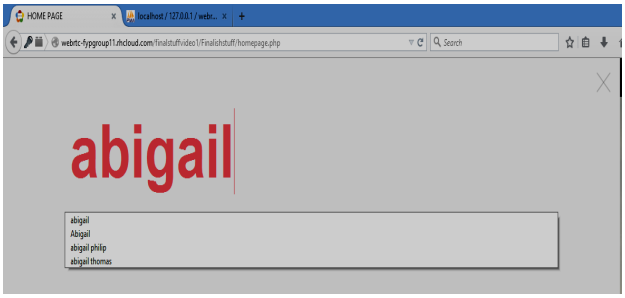


Fig. 8 Searching for a user.

Friend Requests: On searching for a particular user, all the matches are displayed. The result includes profile pictures along with two options, 'View profile' and 'Add as Friend' on clicking the later, a friend request will be sent to that user.

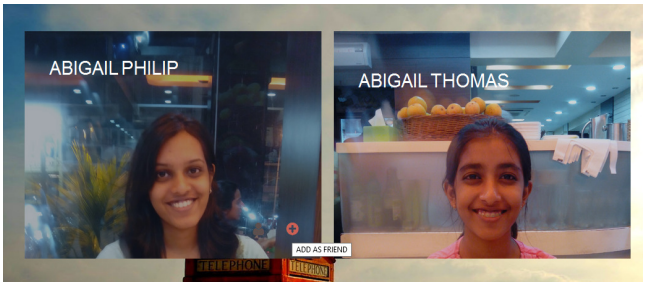


Fig. 9 Results of search.

The header consists of a 'Notifications' slot. On clicking on this, the users sees a table that consists of the users' name who has sent the request and two options 'Accept' and 'Decline'.

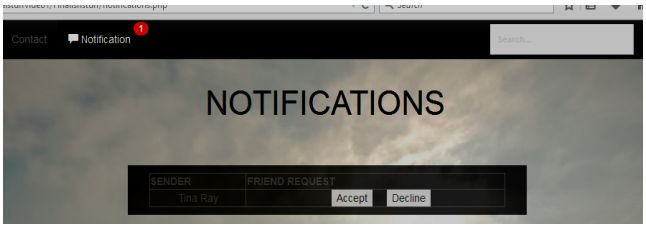


Fig. 10 The Notifications Page.

6.3 Placing a Call and Key Generation

Before a call can be placed; a unique key must be assigned to the sender and then sent to the receiver. After the receiver has received this key, the call can proceed.

Key Generation: Every sender who wishes to place a call (voice/video/multichat) is assigned a unique key. This key

value helps to distinguish one call from another, thus keeping the calls private and unique.

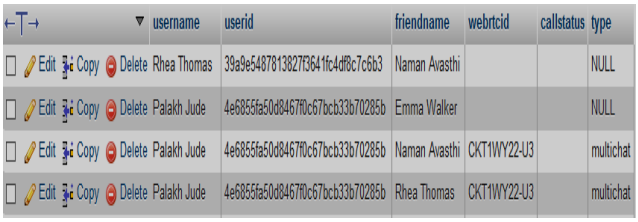
In case of voice/video call, after the user has logged in to the application, the user can now choose a friend to place a call. To do so, the user enters the profile of that friend. From the profile, the user can choose to place a voice/video call. Once the user selects one of these options, a random key is generated using:

Math.random()*newDate().getTime()).toString(36).toUpperCase().replace(/\./g , '-');
The key once generated gets stored in the database against the tuple sender (user) and receiver (friend).

	username	userid	friendname	webrtcid	callstatus	type
Edit Copy Delete	Rhea Thomas	39a9e5487813827f3641fc4df8c7c6b3	Naman Avasthi			NULL
Edit Copy Delete	Palakh Jude	4e6855fa50d8467f0c67bcb33b70285b	Rhea Thomas	88CNZ1TK-H7D		video

Fig. 11 Key entry for video call.

In case of multichat, the user has an option to select two or more friends. A unique key is generated in a manner that is similar to the one mentioned above. However, in this case, the key value gets stored against the names of all the friends that the user has selected.



	username	userid	friendname	webrtcid	callstatus	type
Edit Copy Delete	Rhea Thomas	39a9e5487813827f3641fc4df8c7c6b3	Naman Avasthi			NULL
Edit Copy Delete	Palakh Jude	4e6855fa50d8467f0c67bcb33b70285b	Emma Walker			NULL
Edit Copy Delete	Palakh Jude	4e6855fa50d8467f0c67bcb33b70285b	Naman Avasthi	CKT1WY22-U3		multichat
Edit Copy Delete	Palakh Jude	4e6855fa50d8467f0c67bcb33b70285b	Rhea Thomas	CKT1WY22-U3		multichat

Fig. 12 Key entry for multichat.

Making A Call: This section illustrates what happens when a call is being placed from the sender to the receiver(s). The WebRTC APIs alone cannot complete a call. For the completion of a call, there is a need for a signaling channel. This is to inform either peer about the environment running at the other end.

Firebase is a cloud services provider and a real-time backend as a service. In our application, Firebase is used as a signaling channel. Here, all the Session Description Parameters (SDP) such as resolution, encryption algorithms etc. are sent from the sender to the receiver. This is necessary for the call to be established. Once the key has been generated at the sender's end, his/her SDP parameters are sent to Firebase. Similarly, once the receiver accepts the call, his/her SDP parameters are sent to Firebase. Thus, each party knows about the environment at the other end and a call can be established. After this point, the signaling channel is no longer required and the call proceeds in a peer-to-peer manner.

In case of a voice or video call, t he key that was sent to the database from the sender's end, is now relayed to the receiver's end. The receiver now has the choice to either accept or deny the call. This is implemented by making use of jQuery. The jQuery code constantly checks the database for any new entry. In case of a new entry, an alert message is sent to the receiver. If the receiver accepts the call, the receiver is then redirected to the correct page with the key as part of the URL, thus enabling the sender and receiver to have a voice/video call.

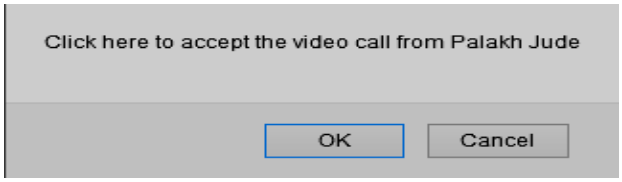


Fig. 13 Alert message for video call.

Once the sender or the receiver enters the voice/video call page, he/she is asked for permission to access his/her microphone (in case of voice calls) and microphone, webcam (in case of video calls). This is done by invoking the getUserMedia API. Additionally, all the calls are carried out via the RTCPeerConnection API. This is responsible for the data (voice/video) that is being transferred.

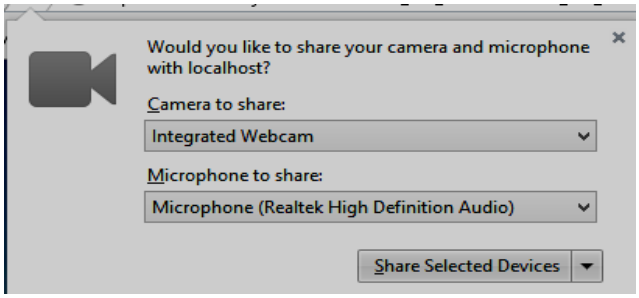


Fig. 14 Request to access webcam and microphone.

In case the receiver the declines the call, the key value is removed from the database and the sender is notified that the receiver has rejected the call.

In case of multichat, it is similar to the video call, except, here it involves more than two participants. Due to the

bandwidth constraints, we have restricted the call to no more than five participants. Here, each of the receivers receives an alert message and has an option to enter the conference.

6.4 Ending a Call

In order to end a call, the user needs to choose the 'Close Conference' option on the page.

In a voice or video call, either of the participants can end the call and the other party will be notified about the call being ended. Simultaneously, the key value is deleted from the database. Both the participants are then redirected to their home page.

Fig. 15 Ending a call.

In case of multichat, this works differently. Here, the call doesn't end until all the participants have left the call. After that, the key value is deleted from the database and the users are redirected to their respective home pages.

7. Conclusions

We have developed a web application that permits video communication through web browsers. This has been done by making use of WebRTC. The improvement over the existing systems is that, in case of our application, no additional plug-ins or softwares need to be downloaded, that is, it is platform independent. Additionally, it also ensures secure communication among the users of the system.

References

[1] Naman Avasthi, Palakh Mignonne Jude and Rhea Thomas, "WebRTC enabled Video Communication", International Journal Of Engineering Research and Technology (IJERT)
ICNTE-2015 Conference Proceedings.

[2] http://chimera.labs.oreilly.com/books/1230000000545/ch18.html#_standards_and_development_of_webrtc.

[3] http://encyclopedia2.thefreedictionary.com/Videotelephony.

[4] https://hacks.mozilla.org/2013/07/webrtc-and-the-early-api.

[5] http://www.webopedia.com/TERM/V/videoconferencing.html.

Naman Avasthi Currently pursuing Bachelor of Computer Engineering degree from Mumbai University. He is studying at Fr. C. Rodrigues Institute of Technology, Vashi.

Palakh Mignonne Jude Currently pursuing Bachelor of Computer Engineering degree from Mumbai University. She is studying at Fr. C. Rodrigues Institute of Technology, Vashi.

Rhea Thomas Currently pursuing Bachelor of Computer Engineering degree from Mumbai University. She is studying at Fr. C. Rodrigues Institute of Technology, Vashi.

Rahul Jadhav Has completed Master of Engineering. He is currently an Assistant Professor at Fr. C. Rodrigues Institute of Technology, Vashi.