Implementation of WebRTC (Web based Real Time Communication) solution for On Premise Video Conferencing System

Khondakar Mohammad Rakibul Hasan

Student ID: 012183010

A Project

in

The Department

of

Computer Science and Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Science in Computer Science and Engineering

United International University

Dhaka, Bangladesh

January 2021

Approval Certificate

This project titled "**Implementation of on-premise WebRTC solution for video conferencing system**" submitted by **Khondakar Mohammad Rakibul Hasan, Student ID: 012183010**, has been accepted as Satisfactory in fulfillment of the requirement for the degree of Master of Science in Computer Science and Engineering on 25 Jan 2021.

Board of Examiners

1.
______________________________ Supervisor
Prof Dr. Mohammad Nurul Huda
Professor and Director, MSCSE
Department of Computer Science and Engineering (CSE)
United International University (UIU)
Dhaka-1212, Bangladesh

2.
______________________________ Examiner
Mohammad Mamun Elahi
Assistant Professor
Department of Computer Science and Engineering (CSE)
United International University (UIU)
Dhaka-1212, Bangladesh

3.
______________________________ Ex-Officio
Prof Dr. Mohammad Nurul Huda
Professor and Director, MSCSE
Department of Computer Science and Engineering (CSE)
United International University (UIU)
Dhaka-1212, Bangladesh

Declaration

This is to certify that the work entitled **"Implementation of on-premise WebRTC solution for video conferencing"** is the outcome of the research carried out by me under the supervision of **Professor Dr. Mohammad Nurul Huda.**

[illegible]

Khondakar Mohammad Rakibul Hasan
Student ID: 012183010
MSCSE Program
Department of Computer Science and Engineering (CSE)
United International University (UIU)
Dhaka-1212, Bangladesh

In my capacity as supervisor of the candidate's project, I certify that the above statements are true to the best of my knowledge.

Prof Dr. Mohammad Nurul Huda
Professor and Director, MSCSE
Department of Computer Science and Engineering (CSE)
United International University (UIU)
Dhaka-1212, Bangladesh

Abstract

The Covid-19 pandemics has given the world a new perspective of being on the work place remotely from home. Remote collaboration and distant learning have reached up to a different dimension that may change the industry practices permanently in days to come. On premise WebRTC solutions can be a cost saving and more secured way of doing this in coming days. My project intends to deploy an on premise WebRTC solution for United International University with relevant requirements, architecture and methodology.

To talk about the WebRTC Technology itself; full meaning of WebRTC is “Web Based Real Time Communication”. WebRTC is comparatively a newer technology which enables standard web browser to launch or host video teleconferencing systems. The exciting part is that, this is open source and the Software Development Kit (SDK) or the Application Programming Interface (API) are easily available in official WebRTC website or in the popular open-source repositories like GitHub, Bitbucket or Source forge etc.

WebRTC has got enormous potentiality in terms of expansion, research and customization. Because of its availability in the open-source market, the technology and its plugins and addons are easily available in the internet. WebRTC has been developed as the core technology, hence it has got full flexibility to expand or customize with considerable amount of research and analysis. WebRTC is a community driven technology where the contributions are coming from developers, implementors and users. Therefore, it has a growing tendency of releasing newer version very often. In addition to that, due to the availability of the core technology at free of cost, many organizations are customizing the white labeled version and developing their own branded video conferencing product out of this technology. My endeavor in this project work is quite a bit similar to build a new app named as “UIU Connect” by using the core and associated technology of WebRTC.

Despite having enormous possibilities, WebRTC implementation has got few challenges as well. First of all, due to its open-source availability WebRTC has got many stable and unstable versions in the internet. Finding the right version remains as challenge. Again, the plugins and addons are community developed, therefore while choosing them we have to be very careful about the security aspects of it. Last but not the least, the versions are updated and released in almost every month. So, working with such fast evolving technology sometimes get challenging for individual developers.

Acknowledgement

I would like to express my special thanks of gratitude to my Project Supervisor Professor Dr. Mohammad Nurul Huda, Director (MSCSE) for guiding me through this project work. Despite of his busy schedule, he has given me lot of time to complete my project work successfully.

Besides, I must thank Assistant Professor Mohammad Mamun Elahi sir as my Examiner who had been extremely passionate and co-operative while I presented my project proposal to him. He guided me and explained the key areas I must cover to make this project successful.

I also must thank the Department of Computer Science and Engineering and CITS for giving me this unique opportunity to do this project on WebRTC, which also helped me to be engaged in lot of R&D and I could gather so much knowledge on this new technology. I am really thankful to them.

Lastly, I would also like to thank my family members and colleagues who assisted me relentlessly in finalizing this project work within the limited time frame.

Table of Contents

List of Figures

List of Images

Chapter 1
Introduction

1.1 Introduction

WebRTC (Web based Real-Time Communications) is a latest technology, which brings audio /video communication and data transfer capabilities to web applications running on standard browsers and mobile device platforms. WebRTC is a standard for embedding both way interactive communications capabilities like voice, video, chat directly from a browser. WebRTC is based on peer-to-peer connection sessions between web sockets available in commonly used browsers and applications APIs. Instead of relying on third-party plug-ins or proprietary software like Zoom, Google Meet or Skype, any organization can implement an on premise; cost free WebRTC Solution to meet their purpose of Video Conferencing and remote collaboration.

WebRTC empowers web-based programs with continuous communications traffic like voice, video and screen/file share through JavaScript APIs. In a traditional Client-Server architecture, data transmission is carried out between the browser and the server. The browser sends a request to the server, and then the server respond corresponding data according to the request parameters. On the other hand, WebRTC achieves peer-to-peer real-time data transmission between browsers where the role of server remains only to ensure session control and signaling.

1.2 Background

GIPS Corp (Global IP solutions Corp) which is a USA based R&D company have extensively experimented on real-time voice and video processing in Peer-to-peer network since 1999. Google, in May 2010 bought GIPS and took over the RTC Project along with the codecs and experimented modules for further developments. 1 year later, in May 2011 Google collaborated with IETF (Internet Engineering Task Force) and W3C (World Wide Web Consortium) and released an Open-Source project named as WebRTC. Since then, WebRTC has been deliberately experimented and supported by Apple, Google, Microsoft, Mozilla, and Opera.

Presently, all ongoing work on WebRTC is being standardized through the W3C and IETF.

1.3 Organization of the Report

The content included in this Project Report are organized into 6 Chapters. After this introductory chapter the subsequent chapters covers the topic in a chronological order.

Chapter 2 describes the motivation to choose this project including problem statement and the purpose itself. Few inherited benefits and justification of choosing this technology has been also described in the second part of the chapter.

Chapter 3 summarizes the core technology of WebRTC where the basic components and architecture are described with their purpose and technical implications.

Chapter 4 introduces the solution which is known as Jitsi Project. Jitsi is the most popular and commonly used platform of WebRTC globally. In this chapter the core components and the methodology of implantation are described.

Chapter 5 basically explain and demonstrate the main implementation part of this project work. Here the step-by-step techniques and procedures including prerequisite environments and CLI commands are showed and explained. The chapter also narrates the deployment and hosting aspects of the core framework and other peripheral services into the Server.

Finally, Chapter 6 describes the use case scenario and advantages of implantation of WebRTC technology specific to an educational institute like UIU. Last but not the least, this chapter also bring some assurance about the technology by describing the security features and technologies involved to prevent cyber security threats and vulnerabilities.

Chapter 2
Motivation

2.1 Problem Statement

During Covid-19 Pandemic situation most of the organization had to adopt some of the Video Conferencing systems for better collaboration within their employee and stakeholders. Commercial Video Conferencing systems are good to have; but have few draw backs such as:

2.1.1 Commercial Video Conferencing Solutions like Zoom, Google Meet or Skype for Business are expensive. Sometime those are not cost effective in regards to the overall outcome or contribution from the organization's perspective.

2.1.2 Those solutions are mostly Cloud Based where the main data stream is handled by the vendor including the encryption and authentication, therefore secrecy and privacy are not always confirmed in regards to Data Security perspective.

2.1.3 Due to business concerns, the solution providers usually do not allow the user to access the core architecture to do much of customization or R&D. Rather, users are only allowed to avail the Software as a Service (SaaS).

2.1.4 White labeling such as Own Logo, Branding or End user supports are not possible due to restricted usage policy from the service Provider.

2.2 Purpose of the Project

My Project work intends to implement an "Open Source WebRTC Solution" for United International University which will facilitate the University to own an on-premise Video Conferencing Solution with the facilities as described below:

2.2.1 On Premise The solution will be deployed in UIU Datacenter and the Core System will be handed over to the UIU Team for future research and development.

2.2.2 Free of Cost For the project work, I intend to deploy a popular Open-Source solution which will involve “zero costing” provided the solution is deployed in UIU Data Center and necessary real IP Host, Domain names and SSL certificates are provided from the university.

2.2.3 Secured and reliable The WebRTC solution will be deployed with following security feature:

- **SSL Encryption** Secured Socket Layer (SSL) Encryption used for Server-Client Communication.
- **E2E Encryption** End to End Encryption (E2EE) for Peer-to-Peer communication.

2.2.4 Full Featured The solution will be as good as any commercial video conferencing system with features like**:**

- Video and Audio Conferencing with HD quality streaming over low bandwidth connectivity.
- Dynamic Invitation link for each meeting room with password protection of the meeting as needed.
- Guest access control by Lobby / Waiting room feature.
- Both way Screen Sharing and presentation mode.
- Live broadcasting over HTTPS from YouTube.
- Activity control on participants by the host like Force Quit, Force Mute etc.

2.2.5 White Labeled the solution will be customized with UIU Logo, Frontend Design and UIU Domain name for proprietary use.

2.2.6 Scalable All source codes and privileged access to the core system will be handed over to UIU IT Team. Further development and resources expansion will be possible at any point of time.

Chapter 3

Technology

3.1 Hierarchical Components

WebRTC is not based on single server, rather collection of servers, API and protocols standardized by W3C and IETF. These APIs defines interaction between different intermediaries of the system. The peers or users are connected through a media stream at the transport layer over the internet. Beneath, the WebSocket API of peer's browsers connect with the Web Server through the signaling process. This API indicates the nature and attributes of requests that can be made, how to make, with which type of data formats and the conventions to follow, etc. It also provisions the extension mechanisms which ensure the extended functionality of different components of the system. The webserver is connected at the backend with the Conferencing manager server and XMPP Server. Extensible Messaging and Presence Protocol in short XMPP, is a combination of technologies for voice and video calls, instant messaging, collaboration, content syndication, and generalized routing of XML data. Lastly, the Selective Forwarding Unit (SFU) server streams the appropriate contents over secured channel which enables the media stream between peers over internet traffic securely. Fig

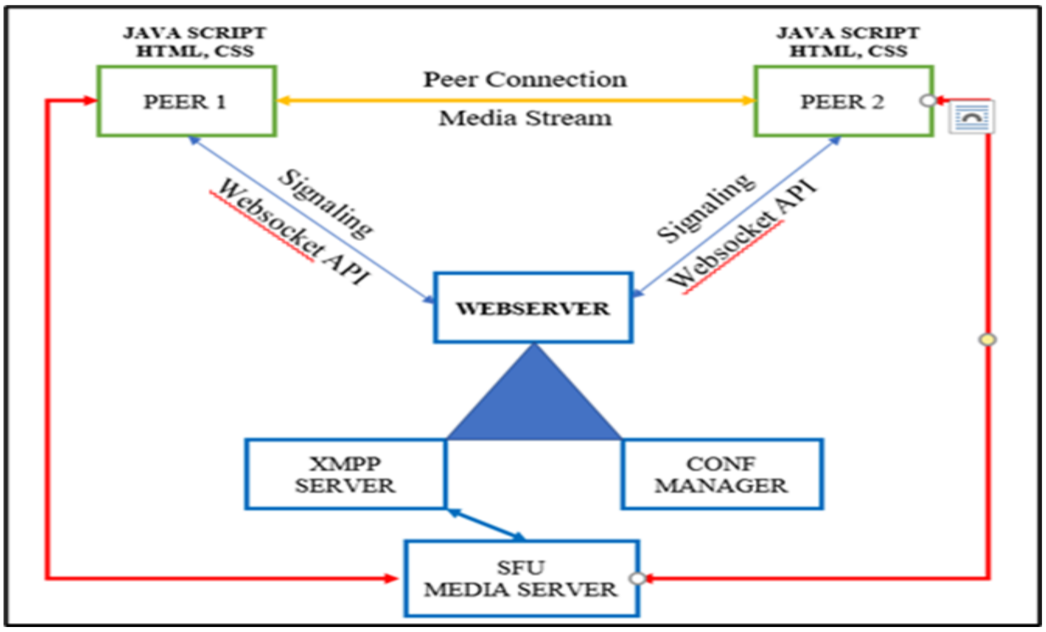


Figure 1: Hierarchical Components of standard WebRTC System

3.2 API Components

As said earlier, API components of WebRTC are interdependent and mutually support each other to perform different function of the overall functions. Often these APIs are deployed in single server and are well connected through a wide range of IP Address and port forwarding options (Figure 2). However, in larger infrastructure the components may also be deployed independently in different physical or virtual servers as well.

3.2.1 Media Stream API

This API controls the synchronized streams of media with a set of input and output devices. The input source might be a physical device, such as the camera or microphone. Media Stream API ensures the user's browser get access permission to the camera and microphone. A real-time media stream is represented by a stream object in the form of video or audio with an added security level through user permissions. Each time the API start fetching a stream seeks user permission to include additional layer of privacy.

3.2.2 RTC Peer Connection API

Real time communication through audio or video calls are set up through this API. This API directly communicate with the web browsers on user's computers. This component provides a framework to make real-time communications of Audio, Video and the transport mechanism.

3.2.3 RTC Data Channel API

Through this API, real time data transmission is ensured by the browsers to establish data channel between nodes with the mechanism of peer-to-peer connections. It represents a bi-directional data channel between two peers which enables exchange of arbitrary data between them. Each RTC Data Channel can be configured to provide reliable or unreliable delivery of messages and in-order or out-of-order delivery of messages.

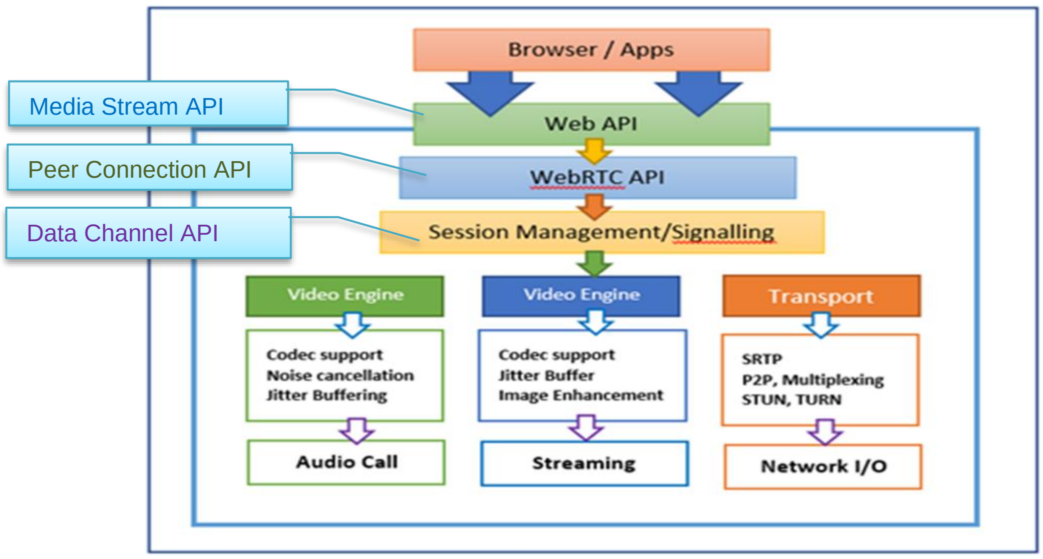


Figure 2: API Components and Different Engines of WebRTC

3.3 Architecture

WebRTC architecture is designed basing on different protocols and standard, which cover most commonly used API / SDK used on web-based applications and browsers at present. Access to the system hardware to capture both voice and video is a prerequisite for the technology. As raw voice and video streams are not compatible to transfer over networks, they need to be processed for noise reduction and echo cancellation. Additionally, those are automatically encoded with optimized narrowband or wideband audio codecs and equipped with necessary error-concealment algorithm as required. These techniques collectively merge the bad effects of network jitter and packet loss that ensures better quality. Therefore, the architecture of WebRTC is quite complex one. Different Codec used in WebRTC are described below:

3.2.1 Audio Codecs Most common audio codecs used in WebRTC are iSAC and iLBC. iSAC is a wideband audio codec for streaming audio that uses a sampling rate of 16-32 KHz with 12-52 kbps bitrate. On the other hand, iLBS is a narrowband speech codec that has got a bit rate of 13.33-15.2 kbs variable bitrate. Another new codec named as Opus has further bitrate as lower as 6kbps.

3.2.2 Video Codecs Mostly used video codec in WebRTC is V8 which requires 100–2K Kbit/s of bandwidth. The bitrate depends on the user defined quality of the streams such as Low, SD, HD etc. This codec is quite popular for web-based video streaming due to low latency.

3.2.3 Real-Time Network Transport WebRTC is a browser-based communication that use Transmission Control Protocol (TCP). For data transmission, WebRTC uses User Datagram Protocol (UDP), which is the basic requirement for all types of real-time communication in the browser. Browser are needed to be compatible with wide ranges of protocols and services and are capable of traversing multiple layers of NATs and firewalls.

Chapter 4

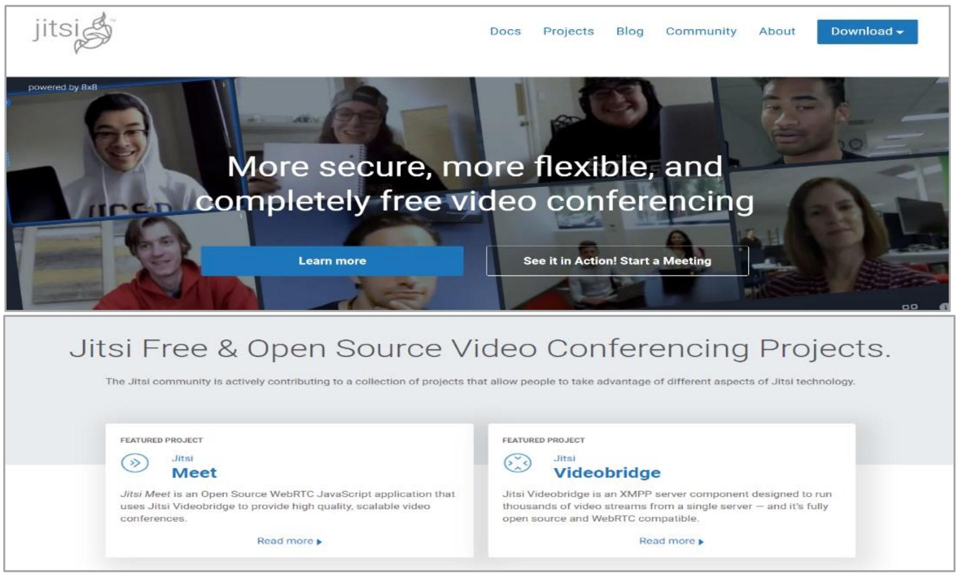
Methodology

4.1 Introducing Jitsi Meet

WebRTC has got number of open-source frameworks, which can be easily implemented, on premise. However, there are number of commercial solutions based on WebRTC popularly known are - Zoom, Cisco WebEx, MS Teams, Google Meet, Rocket, Apache open meetings etc. However, among the free open-source solutions most commonly used framework is Jitsi.

4.1.1 Overview on Jitsi Open Source WebRTC

Jitsi is a collection of community based free and open-source multiplatform solutions with the wide range of services like VTC, VoIP, File Sharing and instant messaging (Figure 3). Jitsi is available for all OS platforms such as Windows, Linux, macOS, iOS and Android. At the beginning, Jitsi project started with the Jitsi Desktop which was also familiarized as SIP Communicator. With the expansion of scope and public demand, the developer community shifted their focus on the Jitsi Videobridge enabling web-based multi-party video calling. Further, the team added Jitsi Meet, which is a full video conferencing system.


jitsi
Docs
Projects
Blog
Community
About
Download
powered by 8x8
More secure, more flexible, and completely free video conferencing
Learn more
See it in Action! Start a Meeting

Jitsi Free & Open Source Video Conferencing Projects.

The Jitsi community is actively contributing to a collection of projects that allow people to take advantage of different aspects of Jitsi technology.

FEATURED PROJECT

Jitsi **Meet**

Jitsi Meet is an Open Source WebRTC JavaScript application that uses Jitsi Videobridge to provide high quality, scalable video conferences.

Read more ▸

FEATURED PROJECT

Jitsi **Videobridge**

Jitsi Videobridge is an XMPP server component designed to run thousands of video streams from a single server — and it's fully open source and WebRTC compatible.

Read more ▸

Figure 3: Screenshot from the Official Webpage of Jitsi

Presently the platform of Jitsi is built on an open-source community-based development model. Thousands of contributors around the world continuously develop and integrate different projects of Jitsi at free of cost. Jitsi allows user to easily build and deploy secure video conferencing solutions on premise servers with prescribed guidelines and manuals. The community initiatives are centrally published and updated through the official website: https://jitsi.org/

4.2 Projects of Jitsi

Commonly known as the "birthplace" for Jitsi, GitHub currently contains 103 repositories for various Jitsi projects. Among all, Jitsi Meet is the heart of the Jitsi family, which is an open-source JavaScript WebRTC application that allows building and deploying scalable video conferencing systems. Few other important and essential projects of Jitsi are described below:

4. 2.1 Jitsi Meet This is the Debian/Ubuntu based Video conferencing server designed for quick installation as the core solution. This is a WebRTC compatible JavaScript application that utilize SFU service named Jitsi Videobridge to enable high-definition video conferences with scalable features.

4.2.2 Jitsi Video Bridge This component performs the task of Selective Forwarding Unit engine for powering multi-party conferences. JVB is responsible for routing video streams within end users. This is highly scalable and modular based solution, which can be instantly deployed or withdrawn basing on the traffic load.

4.2.3 Jitsi Conference Focus (JICOFO) JICOFO manages the ongoing media sessions between each of the peers (Host and Guests) along with the server-side video bridge component.

4.2.4 Jitsi Gateway to SIP (JIGASI) JIGASI is a server-side application that ensures the SIP clients are connected with the meet. If no SIP is provisioned, this is an optional feature.

4.2.5 Jitsi Broadcasting Infrastructure (JIBRI) JIBRI is a set of tools for streaming a conference that works by launching rendered virtual frame buffer and capturing and encoding the output with FF-Mpeg codecs. This part of the solution record, convert and stream audio and video.

4.2.6 Prosody This is the XMPP (Extensible Messaging and Presence Protocol) communication server which ensures the set up and configuration of all other system resources. This is easily configurable and scalable solutions for developers which can be rapidly developed and integrated with latest communication protocols. Under the permissive MIT/X11 license authorities, Prosody is an open-source, community developed solution which is used as core platform of many other solutions (Figure 4).

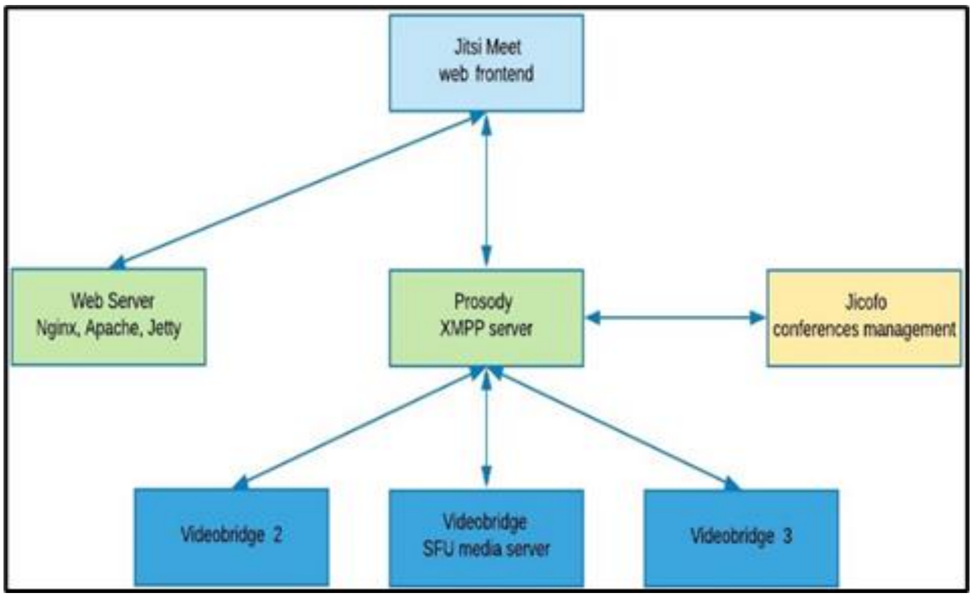


Figure 4: Official Components of Jitsi Project

Chapter 5

Deployment and Integration

5.1 Prerequisite Environment

Jitsi is an easy to install platform requires moderate technical expertise. All open-source resources make it easier and cost-free solutions to deploy in any small or large enterprise environment. Basic prerequisite of deploying Jitsi are:

a. Ubuntu 18.04 server with Root privileges

b. A domain or sub-domain

c. Real IP with Firewall

d. SSL Certificates Webserver like Nginx

5.2 Deployment and Hosting

5.2.1 Server Set-up Jitsi has got different versions which can be deployed in multi-platform environments like Debian, Unix, Docker etc. However, most popular and widely accepted deployment option for Jitsi-Meet installation is on a Debian-based GNU/Linux system. Among other supported distributions, following are the most recent versions that is available in open sources:

a. Ubuntu 18.04 (Bionic Beaver) or later

b. Debian 10 (Buster) or later

Initial preparation of the server required to update two of the packages and repository:

a. gnupg2

b. sudo (only needed if we use sudo command.)

First, we need to make sure that the system is updated and all prerequisite packages are installed in the system. To do so:

a. Run the following update command as ‘root’ or ‘sudo’ to retrieve the latest package versions for all available repositories: *apt update*

b. Then, to ensure that the support for apt repositories served over an HTTPS traffic, the command is typed as: *apt install apt-transport-https*

c. For Ubuntu based systems, Jitsi need some additional dependent packages from universal package repository. Following command is executed to do so:

```
sudo apt-add-repository universe
sudo apt update
```

5.2.2 Setting up a Custom DNS

The domain name is chosen at the beginning. For Example, connect.uiu.ac.bd. This domain will be pointed to the web server where the subscribers will hit first in order to access the meeting service. The DNS A record will be set using a public IP address. If the server is behind a NAT where the server has got a private IP, Dynamic DNS (DDNS) can be configured in the gateway router to ensure the DNS is reachable from outside network. However, in case of setting the Fully Qualified Domain Name (FQDN) following command needs to be executed:

```
sudo hostnamectl set-hostname meet
```

Later, the same FQDN is inserted in the /etc/hosts file, associating it with the loopback address:

```
127.0.0.1 localhost
x.x.x.x connect.uiu.ac.bd
```

where the x.x.x.x is the server's public IP address. Finally on the same machine we need to test that the server can be reachable by ICMP ping test with following cmd:

```
ping "$(hostname)"
```

Once the reachability of the server is ensured via the DNS Configured, the Jitsi repository packages to be downloaded and added in the server. One thing is to make sure that, the server has got internet connection to download and update the repositories as needed.

5.2.3 Jitsi package repository update

Following command will download and add the Jitsi repositories from the Jitsi Website and update the package sources:

```
curl https://download.jitsi.org/jitsi-key.gpg.key | sudo sh -c
sudo apt update
```

5.2.4 Setup and configure your firewall

Since, Jitsi Meet works over traditional TCP Protocol, here is the list of ports (Figure 5) to be allowed in the firewall security policy to allow traffic to the Jitsi Meet server:

Port	Protocol	Service / Purpose
80	TCP	SSL certificate verification / renewal
443	TCP	Access over HTTP
4443	TCP	Fallback media streaming network
10000	UDP	Video/Audio Streaming
22	TCP	SSH for remote access

Figure 5: Required Port and Services Chart

Since, Uncomplicated Firewall (UFW) is used as netfilter following commands will open the above ports:

```
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
sudo ufw allow 4443/tcp
sudo ufw allow 10000/udp
sudo ufw allow 22/tcp
sudo ufw enable
```

Lastly, to check the firewall status, rntmto check the firewall status – 'sudo ufw status verbose' is executed to confirm before proceeding to next steps. If the Jitsi Meet server running behind NAT, the above ports needed to be forwarded on gateway router towards the server's IP address.

5.2.5 Encryption by TLS Certificate

As Jitsi Server communicate between peers over standard E2EE (End to End Encryption) Encryption protocol, the certificate needs to be activated. To create a certificate that is signed by a Certificate Authority, following shell command needs to be executed:

```
sudo /usr/share/jitsi-meet/scripts/install-letsencrypt-cert.sh
```

To use a different self-signed certificate, first the certificate needs to be created by ‘Let’s Encrypt’ of similar platform. Then start installing jitsi-meet and choose to use own certificate option during installation.

But, it is always preferred to use a CA Certificate over self-signed certificates for following reasons:

a. This will cause a warning on user’s browsers as browser is unable to verify the server's identity remotely.
b. Mobile version of Jistsi Meet require a valid certificate from trusted Certificate Authority, where self-signed certificate will not work.

5.2.6 Installing Jitsi Meet

Main job of installing Jitsi Meet will start with the following command which will install the main repository.

```
sudo apt install jitsi-meet
```

during the installation the installer will check if the web server is configured with either, Nginx or Apache and the virtual host is configured within that. If any of them not found the installer will give error. In that case the prerequisite installations are to be ensured first. During the step-by-step process following parameters needed to be entered to complete the installation:

a. **SSL/TLS Configuration**: These steps are described in 5.1.2.5 paragaraph.
b. **Hostname**: The hostname of the Jitsi Meet instance needed to be entered. in the form of Fully Qualified Domain Name (FQDN) or an IP Address. This same hostname will be used for virtual host configuration inside Jitsi Meets as well.

For this project, consider that the installation will run under the FQDN: *connect.cse.uiu.ac.bd* and the SSL certificate and key is stored in this location: /etc/ssl/ connect.uiu.ac.bd{crt,key}. Thereby, following Deb Conf variables net command to be executed before installing the packages:

```
install debconf-utils package
```

And. The server will be ready with the core Jitsi Service. But prior to enable other related peripheral services few more steps to be completed.

5.3 Deploying Required Peripheral Services

After successfully installing the Jitsi Server, different components needed to be configured separately to make the full project functional. A single server Jitsi installation can handle limited size of concurrent conferences. The first limiting factor in this is the video bridge component that handles the actual video and audio traffic. It is easy to scale the video bridges horizontally by adding as many as needed.

A standard jitsi setup architecture is based on Single Jitsi Meet Over Multiple Video Bridge. The Jitsi-Meet server requires comparatively less resources like will generally not have that much load (unless you have many) conferences going at the same time. A 4 CPU, 8 GB machine will probably be fine.

5.3.1 Installation of Video bridge (s)

For installing video bridge similar debconf variables are executed as above and install jitsi-videobridge. For video bridge, only following ports are internally required between the Nginix and Videobridge.

a. Ports open to Public Interface:
   - 80 TCP
   - 443 TCP
b. Ports open to the video bridges only
   - 5222 TCP (for Prosody)
   - 5347 TCP (for Jicofo)

Following diagram (Figure 6) shows the basic architecture with defined port number and traffic directions:

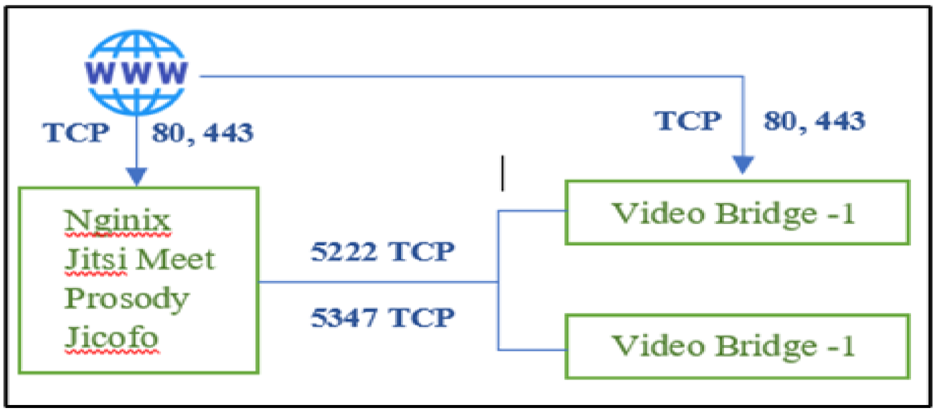


Figure 6: Single Jitsi, Multiple Video bridge Architecture

5.3.2 Installing and Configuring NGINX

To deploy NGINX, we have to create the following configuration file in location: /etc/nginx/sites-available/connect.uiu.ac.bd as usual. Then execute this command:

```
apt install -y nginx
```

After installation Nginix service need to be started and enabled by :

```
systemctl start nginx.service
systemctl enable nginx.service
```

5.3.3 Installing and Configuring Prosody

First the Prosody Package repository needed to be installed by :

```
apt-get install prosody
```

Add following config file in the root directory

```
/etc/prosody/prosody.cfg.lua
/etc/prosody/conf.avail/ connect.uiu.ac.bd.cfg.lua
```

Later the domain virtual host needed to be added along with the security certificate and authentication of conference focus user and host.

5.3.4 Confirmation of installation

After above processes completed, for checking purpose a web browser (such as Firefox, Chrome or Safari) needs to be running on the end user PC. Type the URL (connec.uiu.ac.bd) or the IP Address used as Hostname in the address bar.

If self-signed certificate used, most of the web browser requires a further confirmation that the certificate is trusted. In case of using self-signed certificates, this test may fail if running from an Apple or Android based device’s browser.

After successful entry on the homepage, options to enter a new meeting name will be displayed on the landing page. If the meeting room can be successfully created and the meeting URL is accessible by remote users over public html access, then it can be considered that Jitsi is deployed successfully.

5.4 Testing and Verification

Through Testing and Verification process, the system or components of the developed application is compared against the requirements and specifications. Later the outcomes are evaluated to assess progress of design, performance, suitability etc. For this project the architecture and deployed modules are tested through step-by-step functional tests. The steps are described below with relevant screenshots:

5.4.1 Access and Log in

As said earlier, the project is deployed in UIU Datacenter and the application is pointed towards the domain name:

www.connect.cse.uiu.bd

Through any standard web browser, the application can be accessed by using this domain name (Screenshot 1).

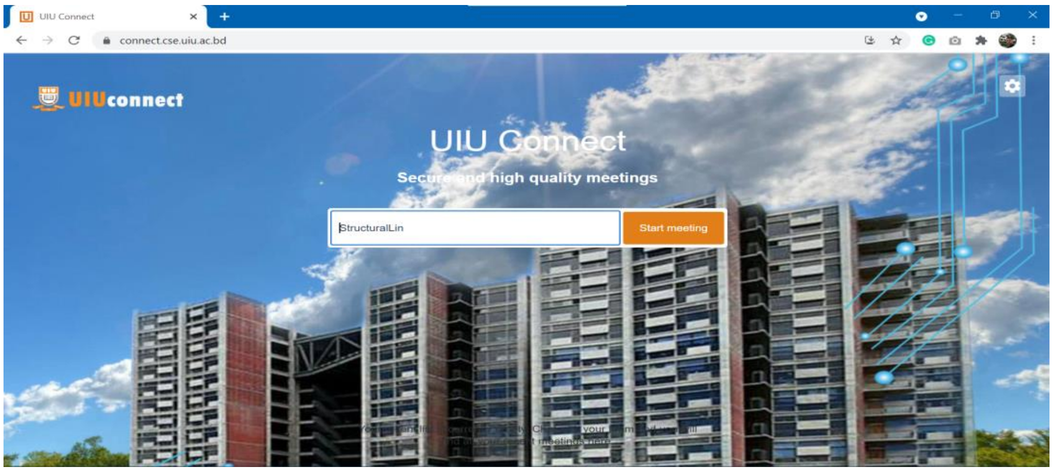
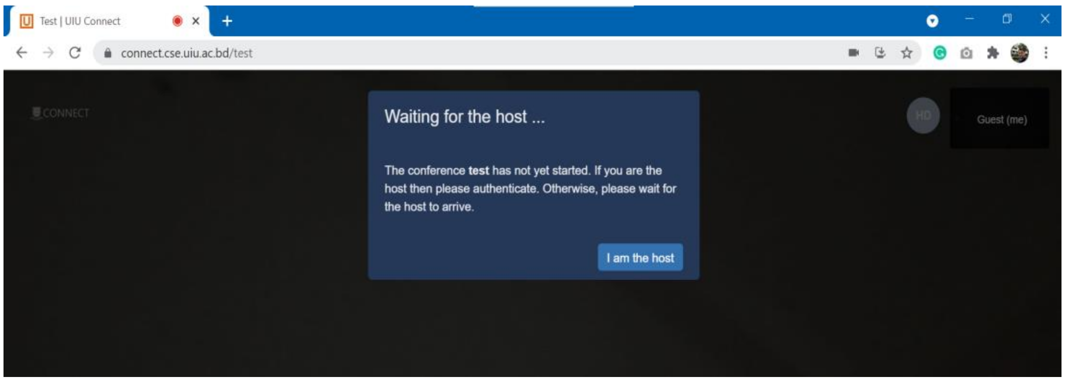


Image 1: Landing Page of UIU Connect

At the center of the home page the "Start Meeting" button will initiate a new conference room. To create a room, user must have a valid Host ID and Password which will be prompted as soon as the room name is entered and Start Meeting button is pressed (Image 2). The user must click "I am the host" to confirm his Identity as a "Host".



Waiting for the host ...

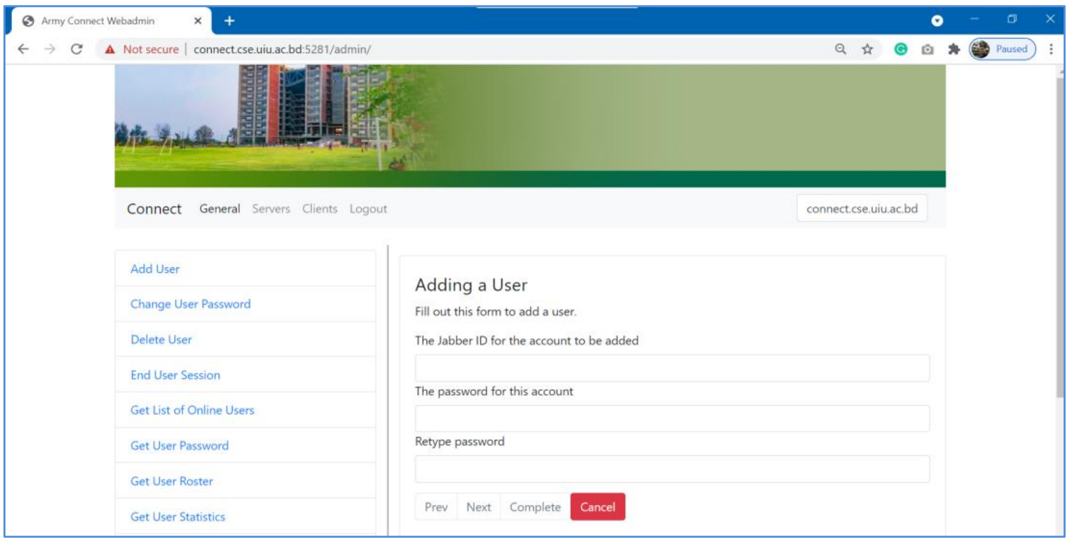
The conference **test** has not yet started. If you are the host then please authenticate. Otherwise, please wait for the host to arrive.

Image 2: Log in Page of UIU Connect

5.4.2 Role of System Admin Any meeting room created in Jitsi requires a valid host to confirm his/her Host ID and Password. Those IDs and Passwords are created by the system admin at the Admin Panel. The URL for the admin Panel is:

https://connect.cse.uiu.ac.bd:5281/admin/

After log in as System Admin, s/he can add new "Host" as user in the admin panel. The adding process is very simple and easy (Image 3). The super admin can also change password, delete user and forcefully end any meeting session from the admin panel as well.



Adding a User

Fill out this form to add a user.

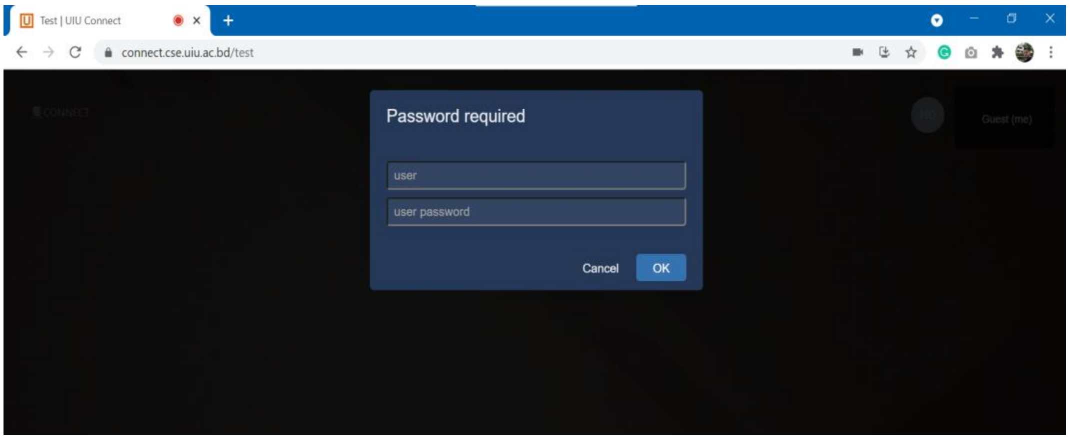
The Jabber ID for the account to be added

The password for this account

Retype password

Image 3: System Admin Panel

5.4.3 Hosting a Meeting with Host ID As described in Para 5.4.1, the host has to click "I am the host" (Image 2), and then s/he has to enter the Host ID and Password created by system admin in previous step (Image 3).



Password required

user

user password

Image 4: Host Name and Password Entry

5.4.4 Starting a Meeting Upon successful entry of Host's credentials, the meeting room is created. Now the URL of the meeting room is changed with a subdirectory name entered as the meeting room name at the home page. For example, naming a meeting room as "dhaka" will automatically create a sub directory URL as: **https://connect.cse.uiu.ac.bd/dhaka/.**This new URL has to be shared with all participants of the meeting via SMS or Email.

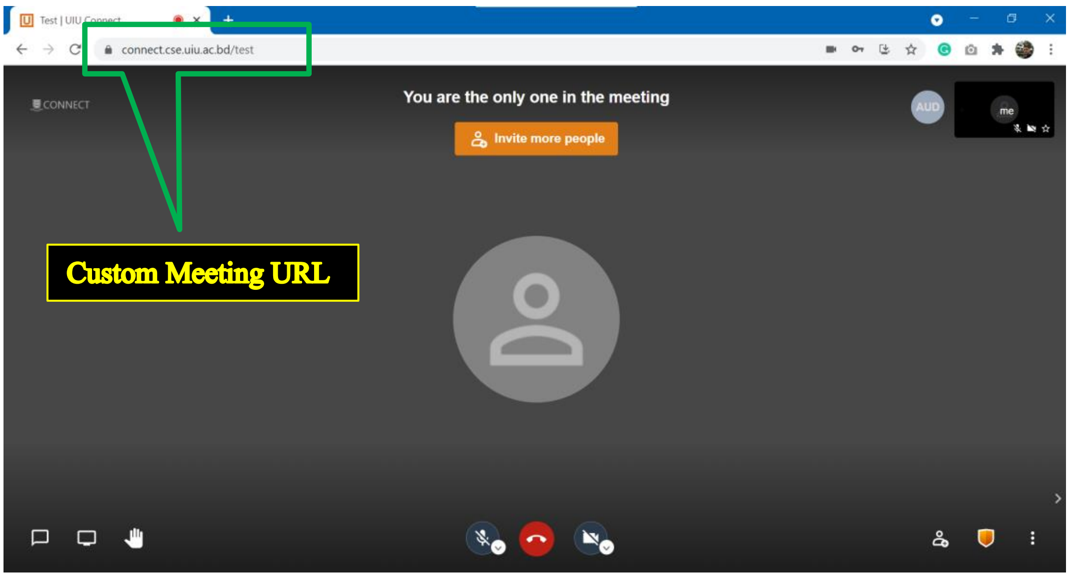
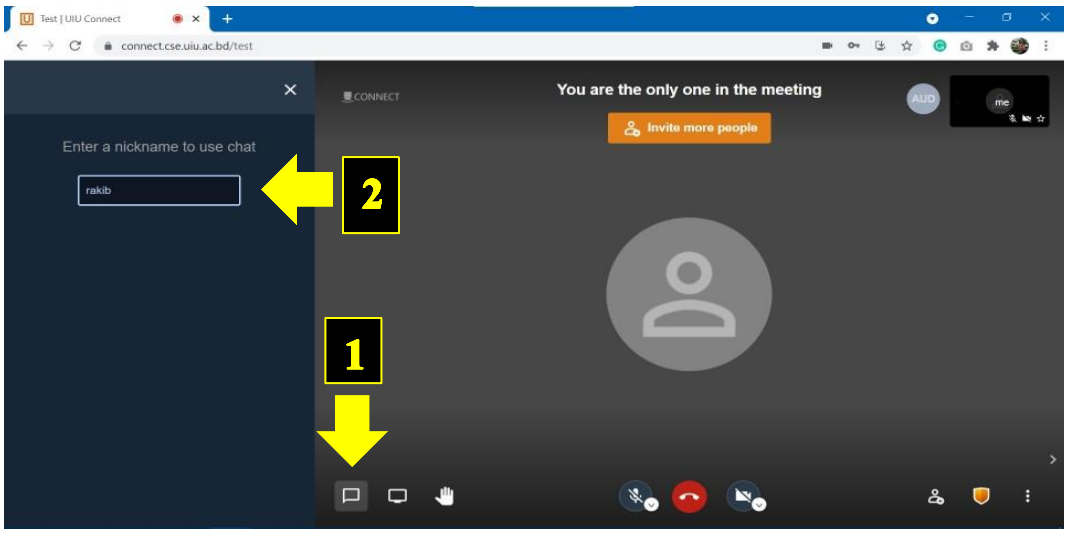


Image 5: Staring a Meeting

5.4.5 On Screen Chat Function Subsequent Images will give an idea about various control elements and components of a meeting room.



Test | UIU Connect
connect.cse.uiu.ac.bd/test
Enter a nickname to use chat
rakib
2
1
CONNECT
You are the only one in the meeting
Invite more people
AUD
me

Image 6: Start a Chat

With Chat button, any user can start a Live Chat on the screen instantly. The user will be asked to enter a chat "Nickname" at the beginning. (Image 6).

5.4.6 Screen Sharing Option During a meeting, any user can share screen for other users. The screen share mode works for full screen, specific browser tab or application windows (Image 7)

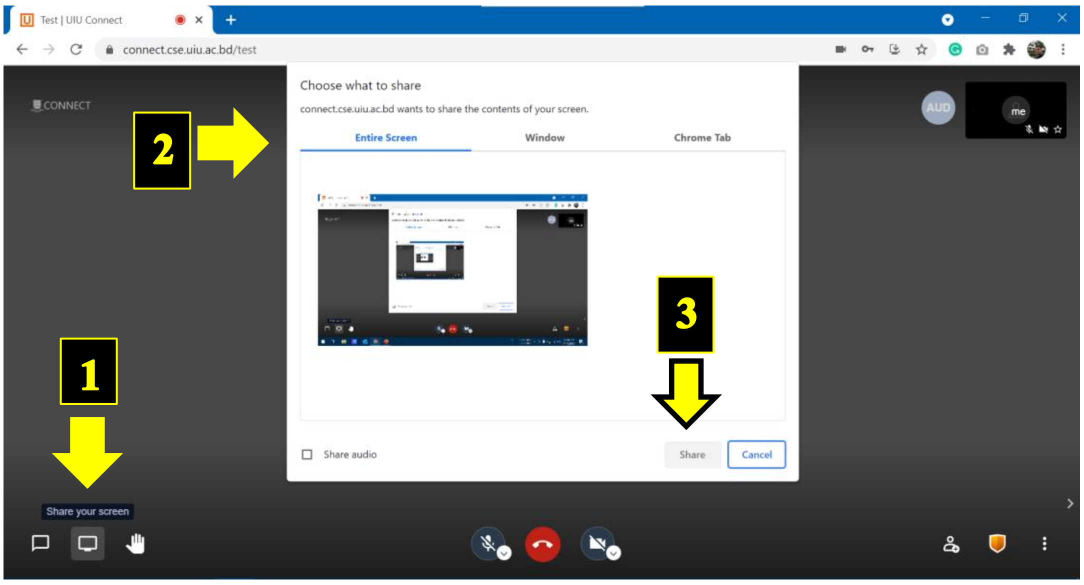


Image 7: Sharing Screen during meeting

5.4.7 Sharing Meeting Link with Users Meeting Link URL can be shared with the user directly from browser address bar or from the "Invite" button. The Link URL can be directly shared by message, email or social media platforms. Jitsi Meetings also facilitate users to embed in any webpage to host any webinar as well which is a unique feature (Image 8).

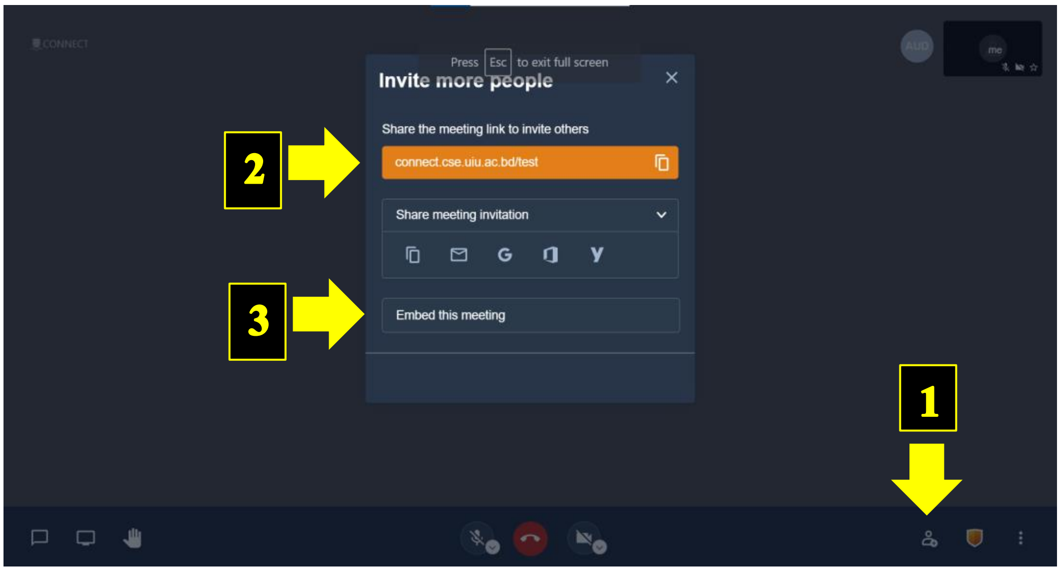
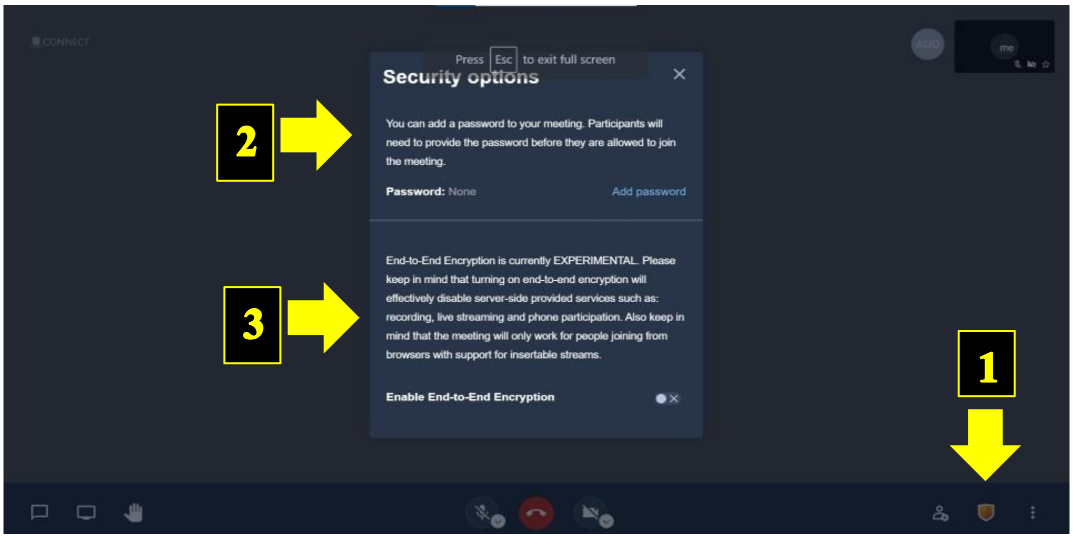

CONNECT
Press Esc to exit full screen
Invite more people
Share the meeting link to invite others
connect.cse.uiu.ac.bd/test
Share meeting invitation
Embed this meeting
AUD
me
1
2
3

Image 8: Sharing Meeting Link with Users

5.4.8 Setting up Security Options As described in Chapter 6 later, the WebRTC has got multiple security feature like Password Protection, End to End Encryption etc. This security options can be setup for each meeting separately from the "Security Option" button shown below (Image 9).



Security options

You can add a password to your meeting. Participants will need to provide the password before they are allowed to join the meeting.

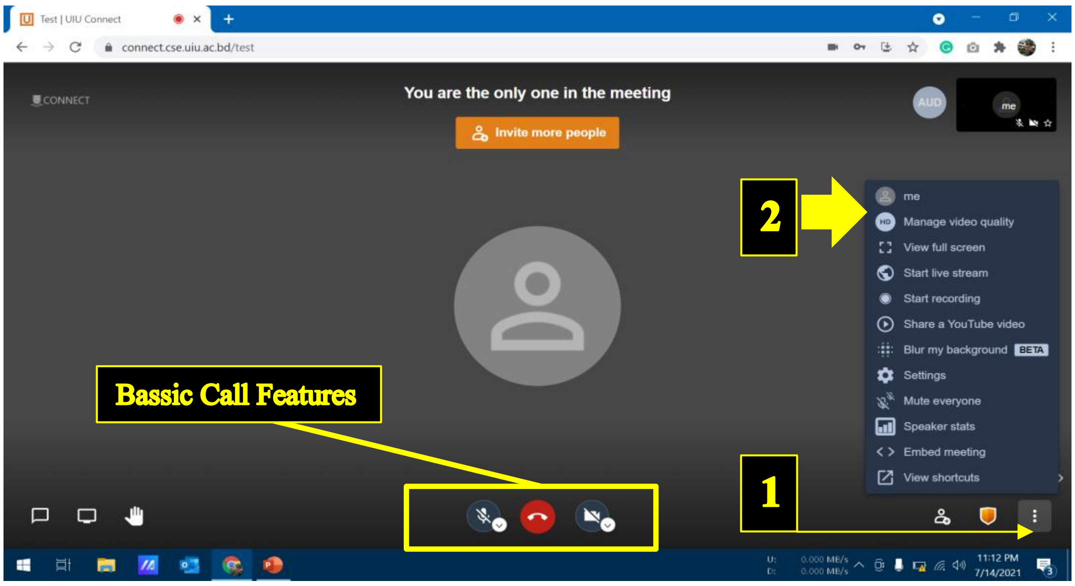
Password: None — Add password

End-to-End Encryption is currently EXPERIMENTAL. Please keep in mind that turning on end-to-end encryption will effectively disable server-side provided services such as: recording, live streaming and phone participation. Also keep in mind that the meeting will only work for people joining from browsers with support for insertable streams.

Enable End-to-End Encryption

Image 9: Setting up Security Options

5.4.9 Control Features During Meeting Miscellaneous admin control features like managing video quality, live streaming, mute everyone etc., are available in the admin control panel shown below (Image 10). Other basic call functions like End, Microphone (On/Off), Video (On/Off) are also embedded as any other standard Video Conferencing Systems.



You are the only one in the meeting

Image 10: Control Features

Chapter 6

Implications and Security

6.1 Use Case Scenario of WebRTC

WebRTC permits a large number of applications for enterprise IT organizations, service providers, system integrators and software package developers. There are numerous opportunities for organizations and small to large enterprises to implement WebRTC as their only communication platform. Few mentionable are:

6.1.1 **On-line Video-Tele conferencing services** Enterprises and service providers may have online meetings, conferences, or webinars with lightweight browser-based audio and video communication services.

6.1.2 **Communications-enabled business processes** Corporate IT organizations can directly integrate real-time communication capabilities into business applications and business processes to increase productivity, improve decision-making and streamline workflows.

6.1.3 **Online Learning and Knowledge Sharing** Schools,colleges, universities and other educational institutions can use for online distant learning and knowledge sharing to the students and researchers for better collaboration.

6.1.4 **Alternative to Mobile devices for enterprises** In order to minimize international mobile roaming charges and calling costs by allowing Wi-Fi calling, business organizations can expand corporate communications services to smartphone or tablet users.

6.1.5 **Over the Top (OTT) Service** Service provider can improve profit margins, ensure market competitiveness and extend service reach to the end users more effectively.

6.1.6 **Communications-enabled Web apps** Businesses may add audio, video or chat to customer-facing Web pages or mobile apps to enhance customer interactions and business development.

6.2 Advantages of WebRTC

- Open source and free
- Unlimited time of usage
- Browser independent
- Standard WebRTC Protocols
- Easily available API and SDK
- Secure and encrypted
- Easy installation and support

6.3 Security of WebRTC

There are many ways that WebRTC platforms can be vulnerable to cyber-attacks. Like any other communication platforms, WebRTC also has few security vulnerabilities which may occur on the source, destination or even at the transmission network. Nevertheless, WebRTC solutions has few extra security features in the architecture and resource usage policy, which place them in advantageous position over other application platforms and frameworks.

6.3.1 Browser Trust Model

A web browser typically enable the user to access the content from the web server securely. In case of HTML and JS hosted in the web server, the browser performs various actions at the end-user PC. In Browser Trust Model of the WebRTC framework, browser can segregate those audio and video streams with the help of sandboxing techniques to isolate the underlying scripts from each other. Besides, all local security policies configured by user along with identity checking parameters are checked for authentic connections.

6.3.2 Access to Media/Local Resources

At present, most of the video / audio conferencing systems commonly get access to the local file resources including peripheral I/O devices to be functional at end user level. Unauthorized access to the camera, microphone, and files is a great concern for user privacy and plays a vital role for any such service to be reliable. In WebRTC, end user is enforced by the platform to give those permissions explicitly which ensures an extra level of data security ensure the security.

6.3.3 Media Encryption & Communication Security

All network traffics along with the media streams passed through various integral components including signaling frameworks of WebRTC are encrypted. Each nodes generating media streams ensure the encryption protocols are enforced at all levels. Basing on the latest encryption algorithm WebRTC framework encrypts different streams basing on different channel types. Usually, Datagram Transport Layer Security (DTLS) is used to encrypt the Data Stream. Besides, media streams are encrypted using Secure Real-time Transport Protocol (SRTP).

6.3.4 DTLS: Datagram Transport Layer Security Generally, DTLS provides extra layer of security to the data streams from falling a victim of eavesdropping, tampering, or message forgery attacks. In WebRTC framework, the data streams sent over RTC Data Channel are secured using DTLS. This is a standardized protocol, which is embedded in all browsers that support WebRTC. The built-in feature of DTLS also ensures the user to get the benefit without installing it on the browser.

Chapter 7

Conclusion and Future Works

7.1 Conclusion

In the present era of modern science and advance technologies, communication has become a basic need for humankind. In the plea for growing awareness of major hacking incidents in corporate world, eavesdropping in government systems and personal scandals - Data Security has become a major concern for all systems. As a result, each and every entity is implementing next generation security measures to all possible sectors of their IT Infrastructure.

WebRTC is in a position that is more advantageous over other popular VoIP systems available commercially in the technology market. As of now, commercial solutions are more focused on consistency rather on security aspects. Considered the security aspect of virtual video/audio communication as optional, presently most of the organization prefer un-encrypted VoIP to save the budget. Larger entities at government and business domain are playing lead role for this, choosing cheaper solutions rather than secured one. But as WebRTC forbids unencrypted communication at any point of the implementation, it can be point of more confidence that communication remain private and safer.

7.2 Limitations

WebRTC is an easy to deploy open-source solutions which is getting popular day by day. Still this technology has got a major drawback that it is still under development. The existing WebRTC API version 1.0 is only a working draft. The other major disadvantage is the requirement of cross platform codecs which has to be supported by all browsers. Till the date of writing this paper, the only royalty free VP8 codec is incorporated under this agreement. However, some larger companies like Cisco are insisting on using H.264 and H.265 codecs which are proprietary products and require purchase by the developers. If this proposal is agreed upon, freelancer community developers will be trouble in future.

7.3 Future Works

Keeping security in mind, the design of WebRTC enforces or encourages important security concepts in all components. Due to the strong focus on secure communication, WebRTC is currently considered by most organizations as the secure web-based communication means all around the world. Lastly, to round everything off, WebRTC is available at open-source platform at free of cost, providing a scalable and reliable platform for developers' community to build their custom-tailored application to be used with self-branding and labels.

References

[1] J.K. Nurminen, A. J. R. Meyn, E. Jalonen, Y. Raivio, and R.G. Marrero, “P2P Media Streaming with HTML5 and WebRTC”, Proc. IEEE Conference on Computer Communications Workshops, April 14-19,Turin, pp.63-64, 2013.

[2] K. Haensge and M. Maruschke, "QoS-based WebRTC access to an EPS network infrastructure," 2015 18th International Conference on Intelligence in Next Generation Networks, Paris, 2015, pp. 9-15, doi: 10.1109/ICIN.2015.7073800.

[3] W. Kim, H. Jang, G. Choi, I. Hwang and C. Youn, "A WebRTC based live streaming service platform with dynamic resource provisioning in cloud," 2016 IEEE Region 10 Conference (TENCON), Singapore, 2016, pp. 2424-2427, doi: 10.1109/TENCON.2016.7848466.

[4] K. De Moor, S. Arndt, D. Ammar, J. Voigt-Antons, A. Perkis and P. E. Heegaard, "Exploring diverse measures for evaluating QoE in the context of WebRTC," 2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX), Erfurt, 2017, pp. 1-3, doi:

[5] B. Sredojev, D. Samardzija and D. Posarac, "WebRTC technology overview and signaling solution design and implementation," 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, 2015, pp. 1006-1009, doi: 10.1109/MIPRO.2015.7160422.10.1109/QoMEX.2017.7965665.

[6] A. Gouaillard and L. Roux, "Real-time communication testing evolution with WebRTC 1.0," 2017 Principles, Systems and Applications of IP Telecommunications (IPTComm), Chicago, IL, 2017, pp. 1-8, doi: 10.1109/IPTCOMM.2017.8169751.

[7] C. Toğay and A. Levi, "WebRTC based augmented secure communication," 2016 24th Signal Processing and Communication Application Conference (SIU), Zonguldak, 2016, pp. 1621-1624, doi: 10.1109/SIU.2016.7496066.

[8] N. M. Edan, A. Al-Sherbaz and S. Turner, "Design and evaluation of browser-to-browser video conferencing in WebRTC," 2017 Global Information Infrastructure and

Networking Symposium (GIIS), St. Pierre, 2017, pp. 75-78, doi: 10.1109/GIIS.2017.8169813.

[9] S. Ouya, K. Sylla, P. M. D. Faye, M. Y. Sow and C. Lishou, "Impact of integrating WebRTC in universities' e-learning platforms," 2015 5th World Congress on Information and Communication Technologies (WICT), Marrakech, 2015, pp. 13-17, doi: 10.1109/WICT.2015.7489664.

[10] Huaying Xue and Yuan Zhang, "A WebRTC-based video conferencing system with screen sharing," 2016 2nd IEEE International Conference on Computer and Communications (ICCC), Chengdu, 2016, pp. 485-489, doi: 10.1109/CompComm.2016.7924748.

[11] R. R. Chodorek, A. Chodorek, G. Rzym and K. Wajda, "A Comparison of QoS Parameters of WebRTC Videoconference with Conference Bridge Placed in Private and Public Cloud," 2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Poznan, 2017, pp. 86-91, doi: 10.1109/WETICE.2017.59.

[12] E. André, N. Le Breton, A. Lemesle, L. Roux and A. Gouaillard, "Comparative Study of WebRTC Open Source SFUs for Video Conferencing," 2018 Principles, Systems and Applications of IP Telecommunications (IPTComm), Chicago, IL, 2018, pp. 1-8, doi: 10.1109/IPTCOMM.2018.8567642.

[13] RFC 4566, Session Description Protocol. Available at : https://tools.ietf.org/html/rfc4566.

[14] RFC 3711, Secure Real-time Transport Protocol (SRTP), Available at : https://tools.ietf.org/html/rfc3711