CS425A-Computer Networks
Network Application
PROVIDENCE: PROfessional VIDeo conferENCE
Multi-user P2P video conferencing
Group: CN-18

Amrit Singhal, (150092)
Department of CSE
IIT Kanpur

Mrinaal Dogra, (150425)
Department of CSE
IIT Kanpur

Nikita Awasthi, (150453)
Department of CSE
IIT Kanpur

Varun Khare, (150793)
Department of CSE
IIT Kanpur

Contents

1 Objective

The aim of the project is to develop a video conferencing web application. This application should be capable of handling multi-stream video feeds for different users within the network.

2 Motivation for the app

Faculty building has several conference rooms hosting conferences on a regular basis and it is often infeasible for the faculty members to attend these conferences in person. This suggests a need for a multi-streaming application to connect faculty members from anywhere within the same network.

3 Features of our application

The advantages of using our application are many fold, as listed below.

3.1 Peer-to-Peer Connection

In a Peer to Peer network, the "peers" are computer systems which are connected to each other via the network. Communication can be done directly between systems on the network without the need of a central server.

We use peer-to-peer connection rather than server based relaying of video streams. The host server only sets up the connection between two or more clients. After the connection is setup, exchange of data among the peers does not involve the server. Such an architecture has the following benefits[4]:

- P2P technology doesn't require support from router and network infrastructure (like server) making this extremely cost-effective and easy to deploy.
- This saves a lot of unnecessary data transmission from client 1 to server and then to client 2. Hence we get to save bandwidth while reducing the possibility of communication failure.
- Mostly the bottlenecks in transmission are the network links. Having a server relaying the data streams heavily restrict the maximum number of possible communications. However, the lesser dependancy on a server in a P2P network increases scalability by an enormous margin.

3.2 Security

The application tries to ensure secure transmission of media to end-users as described below:

- **Channel Encryption:**[1] The Secure RTP protocol (SRTP) is used for encryption and authentication of both voice and video. This is especially beneficial over WiFi networks preventing eavesdropping and recording of the voice and video.
- **Reliability:**[2] Malware or viruses might be installed alongside an apparently innocuous plugin or application. This makes applications requiring installations at client side less appealing for users.
- **Transparency:**[1][2] The application is based on an open-source webRTC framework giving transparency at each and every stage of communication. For example, camera and microphone access must be granted explicitly and, when the camera or microphone are running, this is clearly shown by the user interface.

Currently, we have not dealt with security issues with respect to login and authentication. We have implemented a basic login structure where a username and password is matched to their corresponding entry in the database.[1] However, the application is designed such that it can be extended to include the features of secure authentication considering that credentials may come with access to sensitive information.

[1]HTML template for login webpage UI was taken from `https://colorlib.com/wp/template/login-form-v1/`

3.3 Platform Independent

The application doesn't require the clients to install any software on their personal computer. They just need to login to the hosted website and the server handles setting up the connection and sharing communication protocols. Hence the only requirement is having an updated Mozilla Firefox and/or Google Chrome web browser.

With the browsers becoming ubiquitous even in portable devices like mobile phones we have additional benefit of being a cross-platform application.

3.4 Multiple Chatrooms

Every conference will be allotted a unique chat room id with which all the users can join the associated chatroom. Since there can be several conferences that might be hosted simultaneously in different rooms, the application allows initiating multiple chatrooms to run independent of the others.

4 Implementation Details

4.1 Design Details

A brief explanation of the sockets used by us are discussed as follows:

- **create room**: Server and client communicate on this socket for the purpose of creating a chat room.
- **join**: Server and client communicate on this socket where clients can join an already existing room using its unique room id.
- **leave room**: Server and client communicate on this socket on the event of a user leaving a chat room it was a part of.
- **sdp description**: The SDP (Session Description Protocol) string describing the local connection properties at the end of one sending this description to another peer. [3]Through this socket, a peer sends its local SDP information to the server so that it can be set as the remote SDP description for other peers in the system.
- **ice candidate**: Every ICE (Interactive Connectivity Establishment) candidate describes a method that the sending peer is able to use to communicate. [3]This socket relays the ice candidate information of a new peer to the already existing peers in the system.

4.2 Languages used

1. **Python3** has been used to implement the primary server that hosts the conference service, and connects peers together in a chat room, after which they communicate directly.
2. **HTML/CSS** has been used to develop the front end of the application presented to the user.
3. **JavaScript** has been used to perform the client side functionality of the service. This allows us to keep the client free from any external software/dependencies.
4. **JQuery** has been used to implement the login functionality on the home login screen.

4.3 Libraries and APIs

- Python Libraries used:
  1. Flask
  2. Flask-Login
  3. Flask-SocketIO
  4. Flask-SQLAlchemy
- *Gunicorn* is a python WSGI HTTP Server for UNIX, which is used to host our application on the network. It uses the *Gevent-Websocket*, a websocket handler class required for the socketio functionality of the application.
- JavaScript API used:
  1. webRTC

5 Some other noteworthy aspects of our project

1. We have implemented the project from scratch, where all the network aspects of the project have been coded by us completely. We have understood the working of WebRTC API using examples of single connection video calling, and have implemented the multi peer version of video calling ourselves.
2. We have implemented the functioanlity of allowing multiple chatrooms to be running simultaneously. A user can even be a part of multiple chatrooms, provided he/she accesses them from different devices.
3. A chat room can have any number of users in it. There is no static upper bound on the number of users. We dynamically create, delete and adjust peer connections in real-time as users get added or removed from the chat room. Even the video frames layout in the HTML is dynamically adjusted according to the number of connections.

References

[1] Advantages of webrtc: Web real-time communication. `https://www.networkcomputing.com/unified-communications/9-advantages-webrtc/1953259845`.

[2] Getting started with webrtc. `https://www.html5rocks.com/en/tutorials/webrtc/basics/?fbclid=IwAR15Psa2ALuVcrpWWgYILAxoTcbfvJP8QJBdYnDr2gnnViUwDmNnnfj1d8Y#toc-signaling`.

[3] Webrtc api. `https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/`.

[4] K. I. Z. Apu, N. Mahmud, F. Hasan, and S. H. Sagar. P2p video conferencing system based on webrtc. In *Electrical, Computer and Communication Engineering (ECCE), International Conference on*, pages 557–561. IEEE, 2017.