Creating an Integrated Online Education Platform with Bandwidth Optimized P2P Video Conferencing

Jose Dominic[1], Joel Mani Joseph[2], Vishal Thomas[3], Surekha Mariam Varghese[4]

[1,2,3]*Student, Dept. of Computer Science and Engineering, Mar Athanasius College of Engineering, Kerala, India*
[4]*Professor, Dept. of Computer Science and Engineering, Mar Athanasius College of Engineering, Kerala, India*

Abstract - *The need for a reliable and efficient method of teaching online has increased substantially during recent times. Many students face connectivity and data issues while attending online classes. To mitigate this problem, we propose an integrated online platform for education which uses low network bandwidth wherever possible. Video streams will be sent at the lowest possible size using the WebRTC API while maintaining reasonable quality. Data saving is also achieved by using a PPT file sharing mechanism where a global view of a single file is shared between all users. The platform will also automatically monitor each student and present a report to the teacher at the end of each class showing how long each student was present in the class along with their level of attentiveness. Web RTC open source framework is used for implementing real-time communication between browsers.*

Key Words: WebRTC, Node.js, PeerJS, Education, Web Application, Video conferencing, Javascript

1. INTRODUCTION

The COVID19 pandemic has brought in a time of change and forced paradigm shifts in many areas. It has forced us to rethink the traditional school model and move on to an online mode of education.Furthermore, with this sudden shift away from the classroom in many parts of the globe, the adoption of online learning will continue to persist post-pandemic, and such a shift would impact the worldwide education market. The online model helps learning to continue beyond the four walls of the classroom, allows students' choice and flexibility to learn at their pace, creates more opportunities for collaborative tasks along with providing opportunities to rethink the mode of assessment feedback. This brings in the need for efficient and easy to use online platforms for education.

Upon extensive study of existing platforms, it was found that none of them contained all the necessary features which could satisfy the needs of students and teachers of all levels. Such platforms should be usable on as much low bandwidth as possible as all the users may not be having access to high speed internet connectivity. Not having a satisfactory way to take attendance of the students online,measuring the degree of attentiveness of the students, difficulty in conducting online examinations, managing recorded versions of live classes properly were all found to be the significant issues faced by teachers in existing platforms. Also it was found that students report running out of mobile data while attending online classes.
We propose a WebRTC based peer to peer online video conferencing platform specifically optimized for education that functions well on as low bandwidth as possible. To tackle the major issues currently faced by students and teachers in online mode of education we introduce a platform in which users can meet all their educational requirements in one place rather than using multiple tools. The system is deployed as a web application which can be accessed from any web browser.

2. BACKGROUND

2.1 WebRTC

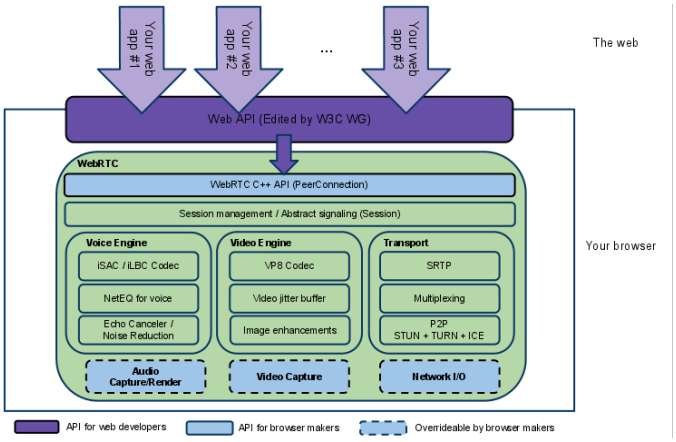

Fig -1:WebRTC Architecture

WebRTC (Web Real-Time Communication) is a technology which enables Web applications and sites to capture and optionally stream audio and/or video media, as well as to exchange arbitrary data between browsers without requiring an intermediary. The set of standards that comprise WebRTC makes it possible to share data and perform teleconferencing peer-to-peer, without requiring that the user install plug-ins or any other third-party software.[1]

WebRTC serves multiple purposes; together with the Media Capture and Streams API, they provide powerful multimedia capabilities to the Web, including support for audio and video conferencing, file exchange, screen sharing, identity management, and interfacing with legacy telephone systems including support for sending DTMF

(touch-tone dialing) signals. Connections between peers can be made without requiring any special drivers or

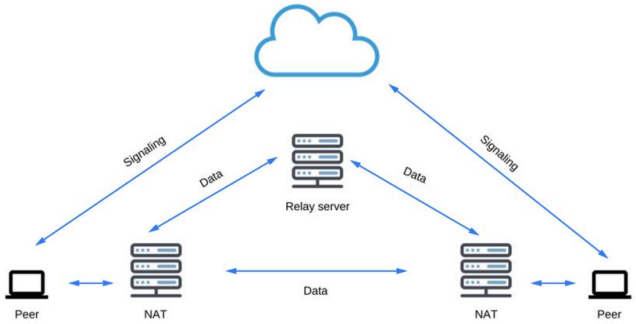

Fig -2: WebRTC Flow diagram

plug-ins, and can often be made without any intermediary servers.[1]

STUN SERVER: Session Traversal Utilities for NAT (STUN) is a protocol to discover your public address and determine any restrictions in your router that would prevent a direct connection with a peer.[2]

The client will send a request to a STUN server on the Internet who will reply with the client's public address and whether or not the client is accessible behind the router's NAT.[2]

Fig -3: STUN server

TURN SERVER: Network Address Translation (NAT) is used to give your device a public IP address. A router will have a public IP address and every device connected to the router will have a private IP address. Requests will be translated from the device's private IP to the router's public IP with a unique port. That way you don't need a unique public IP for each device but can still be discovered on the Internet.[2]

Some routers will have restrictions on who can connect to devices on the network. This can mean that even though we have the public IP address found by the STUN server, not anyone can create a connection. In this situation we need to turn to TURN.[2]

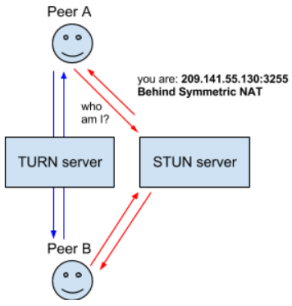

Fig -4:Turn Server

2.2 PeerJS

PeerJS is a javascript library that wraps the browser's WebRTC implementation to provide a complete, configurable, and easy-to-use peer-to-peer connection API. Equipped with nothing but an ID, a peer can create a P2P data or media stream connection to a remote peer. [3]

2.3 Node.js

Node.js is a javascript based-platform for building scalable network applications. It has a similar asynchronous runtime like javascript. It is a server side framework. Node.js runs the V8 JavaScript engine, the core of Google Chrome, outside of the browser. This allows Node.js to be very performant.[4]

A Node.js app runs in a single process, without creating a new thread for every request. Node.js provides a set of asynchronous I/O primitives in its standard library that prevent JavaScript code from blocking and generally, libraries in Node.js are written using non-blocking paradigms, making blocking behavior the exception rather than the norm.[4]

2.4 ExpressJS

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. It is a javascript library used for building at server side. Write handlers for requests with different HTTP verbs at different URL paths (routes). Integrate with "view" rendering engines in order to generate responses by inserting data into templates. Set common web application settings like the port to use for connecting, and the location of templates that are used for rendering the response. Add additional request processing "middleware" at any point within the request handling pipeline.[5]

3. PROPOSED METHOD

We propose a WebRTC based peer to peer online video conferencing platform specifically optimized for education that functions well on as low bandwidth as possible. To tackle the major issues currently faced by students and teachers in online mode of education we introduce a

platform in which users can meet all their educational requirements in one place rather than using multiple tools. The system is deployed as a web application which can be accessed from any web browser.

3.1 Platform Architecture

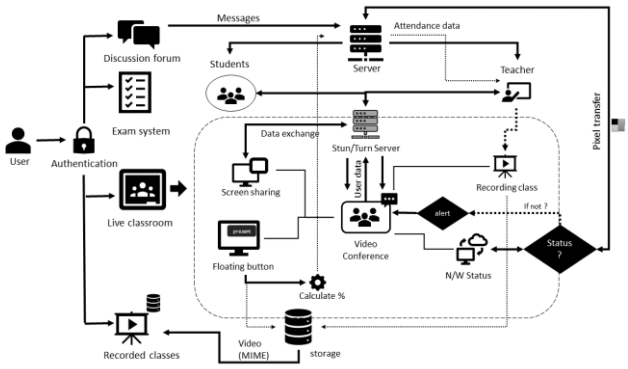

Fig -5:Architecture of proposed platform

In the architecture, we can see 2 physical servers where one of them is the main server and STUN/TURN server. Main server is responsible for handling authentication of users, maintaining web socket connections. It has both signalling server and peer server (handles peer connections). STUN/TURN server is responsible for initiating connection between peers for p2p conferencing.

The platform provides the following functionalities:

- Authentication : User need to authenticate himself/herself to log into the platform
- Discussion Forum : Students could ask doubts to teachers and clarify it.
- Exam System: System for conducting tamper proof exam by making use of object detection models.
- Sharing Screen: During live class users can share the screen.
- PPT sharing: Teacher could share the ppt slide by slide as images. The current slide image will be rendered on the student side at the same time.
- Floating button mechanism: For monitoring student attentiveness during live class.
- Recording live class: Both students and teacher can record the live classes and video can be viewed later.
- Network status monitoring is used to alert the teacher while she is presenting the screen. During the presentation time she might get cut off from the network, in-order to notify her about the network status, an alert mechanism is used.

4. IMPLEMENTATION

The project is implemented as a Node.js web application. We have used the expressJS library for running the backend server. The main server listens for incoming connections on port 3000. The peerserver runs at the endpoint"/peer" on the main server. After this the database schemas,user views and authentication code are defined. The encryption of data in the database is done with the help of passportjs and bcryptjs. The user views are configured as expressJS templates and client side scripts are written to connect to the server. We then configure the STUN/TURN servers on Azure using a linux instance. The application is tested locally for bugs and is deployed to heroku which is a free hosting service.Implementation of different functionalities are discussed in detail below.

4.1 Video Chat Workflow

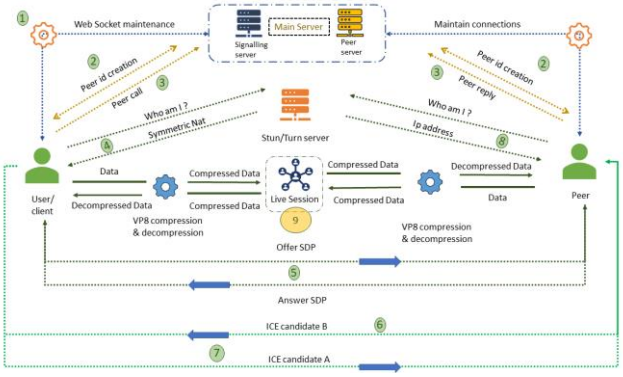

Fig -6:Workflow of video chat

There are two major algorithms at play here: the signalling server algorithm and the client side script.

The main purpose of the signalling server is to monitor group memberships. It takes care of establishing and maintaining web socket connections for different clients and also notifies clients of changes to group membership by using the below algorithm.

Signalling server algorithm

1. Start
2. Listen for incoming web socket connections
3. When a new socket connects

a. On event join-room, notify all users in room of new user with new userid

4. On socket disconnect

a. Notify all users in room

5. Stop

Client side algorithm

1. Start
2. Make new web socket connection to signaling server

3. Get videogrid id,roomId
4. Create new peer object myPeer and connect to peerserver
5. Configure peer object with stun/turn credentials
a. On connection, emit 'join-room' event through socket to notify signaling server
b. myPeer will be assigned a random unique ID
6. Create a new video element
7. Fetch mediastream of current user through navigator.getUserMedia() into myStream
a. Video quality given as parameter
8. Add the current user videostream to videogrid
9. When a new user connects
a. Fetch new user id(peerId)
b. Call the new user with myPeer.call(userid, myStream)
c. Fetch returned mediastream into newStream
d. Create a new video element with newStream
e. Add new element to videogrid
10. On receiving call event from user
a. Acknowledge call and send myStream to caller
11. When a user disconnects
a. Remove that users video element from videogrid
12. Stop

The client side script is responsible for peer connection negotiation with the peer server and fetching and transmitting the user media stream acquired via the browser. It does so by making use of the above algorithm. Videogrid represents the area on the screen where the videostream is to be displayed and roomId refers to the id of the group that the client is part of. Each client is represented by using a Peer object in the network and is assigned a unique identifier by the peer server upon request.

4.2 Setting Up STUN/TURN Server

A linux VM instance is first created on azure. The ports 3748(TCP and UDP), 49152–65535 (UDP in cloud security groups) are open for communication. After this coturn server which is basically an open source server that acts as both the TURN and STUN server is installed and configured.

Coturn library configuration

1. Open ports mentioned above in firewall settings on azure
2. Install coturn library using sudo apt-get command
3. Run coturn as an automatic system service daemon
4. Create /etc/turnserver.conf file and enter the server configuration information like external ip
5. Restart coturn service

4.3 Live Class Recording and Cloud Storage

We make use of the MediaDevices interface of the WEBRTC API particularly the getdisplaymedia() function. This helps fetch the required stream from the available pool of streams. The fetched stream is then converted to a media stream and then into a blob which is basically an object containing raw immutable binary data. The user on stopping the recording is asked for a name (for the recording). This name is appended with an mpeg4 extension and uploaded to the server using a Nodej.js middleware called multer.

4.4 Measuring Student Attentiveness

We make use of a floating button for this. The button is rendered at random places on the screen at random intervals. The student has to click these buttons. Once the student leaves the meeting the percentage of clicks (with respect to the no: of floating button rendered) is calculated at the client side and sent to the server. The attendance is then given based on this measure. For instance if the floating button appears 10 times during a class on the screen and the student clicks the button 6 times out of 10 then his attentiveness will be 60 percent. The attendance threshold for attentiveness can be decided by the teacher.

4.5 Network Status Monitoring

For monitoring the network status, all clients in the room/meeting will continuously try to fetch an image (containing only one pixel) stored at the server. In case of network disruption for a client the HTTP response code in that scenario will not be in the range 200 - 299, which is being checked at the client side. In this situation the user is given an alert indicating that he has been disconnected from the network.

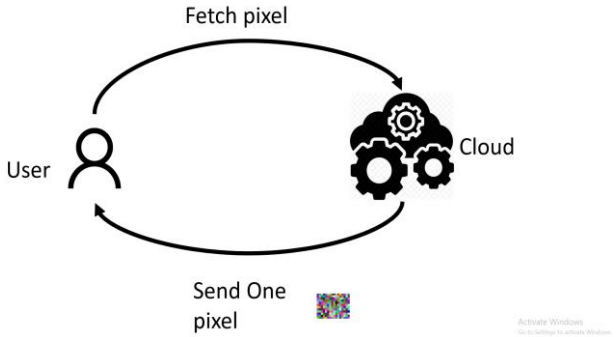

Fig -7:Network status - user online

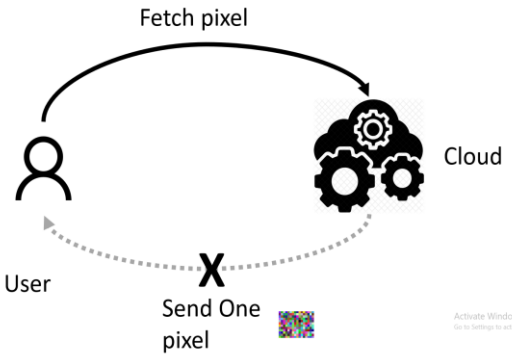

Fig -8:Network status - user offline

4.6 PPT File Sharing

The teacher first uploads the powerpoint to be presented. Using an API called convertAPI the slides of the powerpoint are converted into images and stored on the server. During the meeting when the teacher wants to present the powerpoint he/she can use the "Share PPT " button and select the required presentation from the list. On selecting the presentation, the image corresponding to the first slide is loaded automatically at both the teacher's and student's side. Only the teacher has the option to change the slides using the next and previous buttons and the images corresponding to the required slide are loaded. The slide changes are automatically reflected at the students side also. These slide change events are sent using websockets. In case the student joins the class late for some reason and figures out the teacher is presenting something then he/she can use the "View shared PPT" button to enter and view the current slide being presented.

5. RESULT

The application is deployed in the cloud using Heroku CPaas. Already registered users can login directly and others need to first register and then login. Some of the features of our final system are explained below.

5.1 Live Classroom

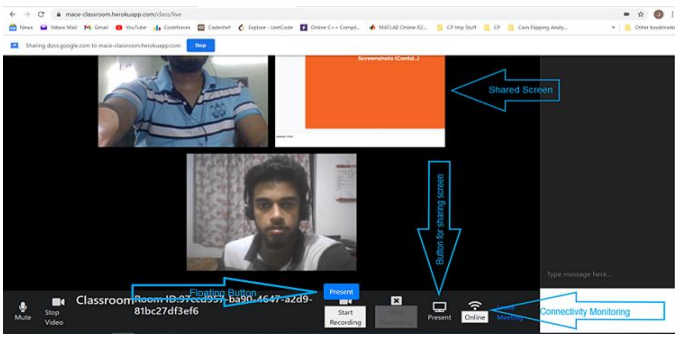

Fig -9:Screenshot from a live class demo

The teacher first creates an online meeting and shares the room code with her students.The student can then join the live class by entering his room id. Once the student joins the class, his attendance is recorded depending on the number of times he clicks the floating button that appears at random points on the screen at random intervals. The connectivity status can be seen at the bottom inside the classroom and any disconnection will be alerted to the user. Both the student and teacher have the option to record the live class. At the end of the class the teacher can access the student attendance and attentiveness data from her dashboard. Also the recorded classes will be available to both the students and the teacher.

5.2 Low Bandwidth PPT File Sharing

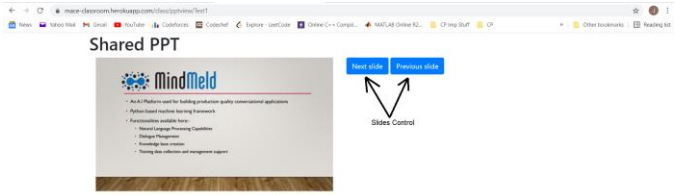

Fig -10: Screenshot of teacher sharing PPT

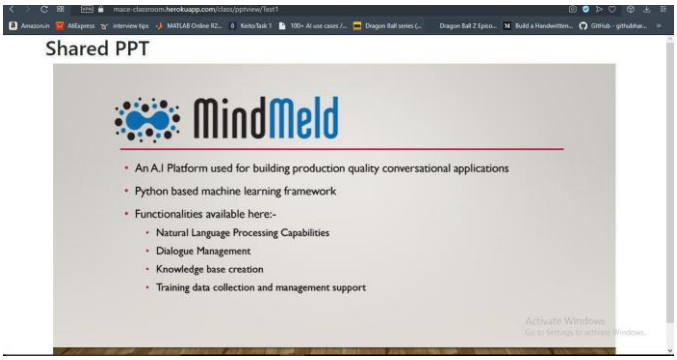

Fig -11:Screenshot of student viewing shared ppt

For sharing PPTs for taking online classes, the teacher does not have to share her screen, sacrificing more bandwidth. The teacher can share the subject PPT by making use of our data saving mechanism explained in the previous chapters. First the teacher has to upload the required PPT via her dashboard. The uploaded PPTs can be shared to a live class by the teacher using the share PPT button in the live classroom. When the teacher shares the PPT the corresponding PPT is opened at all the students' browsers.

5.3 Tamperproof Exam System(Future Scope)

Exams scheduled by the teacher can be accessed by the student from their dashboard at the stipulated time. The

student will be allowed to proceed to view the question paper only if he passes the person detection. The student has to be present in front of the camera to be allowed to enter the exam room. Once the student passes the validation, his screen gets locked and can view the question paper until the timer runs out. If the student tries to exit the exam room he is given a warning and is flagged for malpractice if he proceeds to exit the exam before completion without submitting the answer sheet.

6. PERFORMANCE ANALYSIS

Data consumed by the system was monitored using network monitoring software Netbalancer. The tests were carried out while three users were engaged in a video conferencing with their audio and video turned on. In the experimental conditions mentioned above, our system, google meet and zoom's data consumption was monitored over a duration of 120 seconds.The resulting graphs are shown below.

Fig -12:Our system's network usage

Fig -13:Google meet network usage

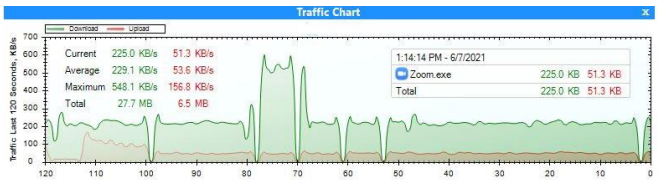

Fig -14:Zoom network usage

From the above results it can be seen that our system consumes significantly lower bandwidth over time compared to google meet and zoom. The tests were conducted with three users but the results are expected to follow the same pattern even at higher user count as our system makes use of the peer to peer model. As more and more users join, the overall system efficiency will only improve.

7. CONCLUSION

We were able to successfully design and develop an all in one platform for online education that consumes less data than existing platforms. We faced the challenge of simulating real world test case scenarios as a large number of users may join an online class at the same time. Still we expect the system to perform well even under higher traffic based on our experiments with a small number of users. It was found that our system will deliver a smooth video conferencing experience at bandwidths as low as 150KBPS which will be of great help to students with poor data connectivity.Also the features like tamperproof examination and attentiveness monitoring that are implemented are sure to ease the academic struggles of both students and teachers in the online mode. The system is designed in such a way that it can be easily scaled as per the requirements of various institutions.

ACKNOWLEDGEMENT

First and foremost, we sincerely thank 'God Almighty' for his grace for the successful and timely completion of the project. We express our sincere gratitude and thanks to Dr. Mathew K, Principal and Prof. Joby George, Head of the Department Computer Science and Engineering of Mar Athanasius College of Engineering Kothamanagalam for providing the necessary facilities and their encouragement and support. We owe special thanks to our project guide and coordinator Dr. Surekha Mariam Varghese for her guidance, constant supervision, encouragement and support throughout the period of this project work. We are grateful to the staff-in-charge Prof. Ani Sunny for her corrections, suggestions and sincere efforts to coordinate the project under a tight schedule. We express our sincere thanks to staff members in the department of Computer Science and Engineering for helping me with the successful completion of the project. Finally, we would like to acknowledge the heartfelt efforts, comments, criticisms, co-operation and tremendous support given to us by our dear friends during the preparation of the project and also during the presentation without whose support this work would have been all the more difficult to accomplish.

REFERENCES

[1] https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API

[2] https://eytanmanor.medium.com/an-architectural-overview-for-web-rtc-a-protocol-for-implementing-video-conferencing-e2a914628d0e

[3] https://peerjs.com

[4] https://nodejs.dev

[5] https://expressjs.com

[6] A.. Ushakov, M. V. Ushakova, A. E. Shukhman, P. N. Polezhaev and L. V. Legashev, "WebRTC based Platform for Video Conferencing in An Educational Environment," 2019 IEEE 13th International Conference on Application of Information and Communication Technologies(AICT), 2019

[7] Nayyef, Zinah Amer, Sarah Hussain, Zena. (2019). Peer to Peer Multimedia Real-Time Communication System based on WebRTC Technology. International Journal for the History of Engineering Technology. 2.9. 125-130.

[8] Maraˇsevi´c and A. Gavrovska, "Virtual Reality and WebRTC implementation for Web educational application development," 2020 28th Telecommunications Forum (TELFOR), 2020

[9] https://webrtc.github.io/webrtc-org/architecture

[10] https://blog.sessionstack.com/how-javascript-works-webrtc-and-the-mechanics-of-peer-to-peer-connectivity-87cc56c1d0ab

[11] https://www.highereducationdigest.com/the-importance-of-online-learning-in-the-times-of-covid-19-and-beyond