REAL-TIME E-LEARNING PLATFORM

Final Report for Lecture Live Streaming Module

Project ID: 19-079

A.M.H.B. Athapaththu

IT16131002

Bachelor of Science (Hons) in Information Technology Specialized in Software Engineering

Department of Software Engineering

16th September 2019

REAL-TIME E-LEARNING PLATFORM

Project ID : 19-079

A.M.H.B.Athapaththu

IT16131002

Supervisor's name – Dr. Malitha Wijesundara

Department of Information Technology

16th September 2019

Declaration

I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. Also, I hereby grant to Sri Lanka Institute of Information Technology the non-exclusive right to reproduce and distribute my dissertation in whole or part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as article or books).

Signature: Date: 16th of September 2019

Signature of the Supervisor: Date: 16th of September 2019

Abstract

Real-time e-learning platform is a WebRTC based live teaching platform which facilitates the e-learning requirements of universities and other institutes. This solution includes lecture live streaming, lecture playback, a vector based interactive whiteboard, chatting and file sharing module and a real-time lecture movement tracking module using a PTZ camera. The system is capable of streaming two simultaneous streams of a 1080p camera and a 720p screen capture seamlessly using a network connection with 256KB/s bandwidth. Live streaming component is very less CPU intensive and it use around 14% of the CPU for streaming 10 simultaneous sessions with 10 listeners per each on a AWS t2.micro instance with 1 vCPU 2.5 GHz, Intel Xeon Family, 1 GiB memory. Because of that, this solution is a very cost effective product compared to existing competitors in the market.

ACKNOWLEDGEMENT

First, I would like to express my gratitude to supervisor Dr. Malitha Wijesundara and co-supervisor Mr. Pramadi Athapaththu for the guidance and mentoring given throughout this period to make this research a success. Also I would like to thank senior lecturer and lecture in-charge for Comprehensive Design and Analysis Projects module, Mr. Jayantha Amararachchi for the valuable assistance given to complete this project. At last but not least I am also thankful to all my family members and colleagues who have been always helping and encouraging me throughout the research period. I have no valuable words to express my thanks, but my heart is still full of the favors received from every person.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF ABBREVIATIONS

WebRTC	Web Real-Time Communications
RTP	Real-time Transport Protocol
PTZ	Pan Tilt Zoom
CPU	Central Processing Unit
SFU	Selective Forwarding Unit

1. INTRODUCTION

Real-time e-Learning platform is a live lecture streaming web application which is developed focusing on Universities and other educational institutes. Lecturers can stream lectures from or out of the institute premises remotely while students can join relevant live streams from anywhere. This report will describe the live streaming component of the application. Lectures streamed from institute premises will be equipped with a PTZ camera to get a third person perspective camera feed of the lecturer. Lecturers can use their web camera instead of PTZ camera if they are streaming remotely outside of the institute premises. Apart from the camera feed, Lecturer's Computer screen is captured and streamed. Lecturer can select between which screen or which application to share where they have a choice to manage their privacy when they share their screens with others. Also audio feed from a mic attached to the lecturer's PC is broadcasted. The application is capable of handling multiple simultaneous lecture sessions without any interruptions to each other. This component is also responsible for recording live streams to facilitate the lecture playback where students are given the capability to rewind and playback already streamed lectures for future reference.

We have considered the same concept of the physical lecture/class room scenario when computerizing it into our implementation. In a classroom, what students required to focus on is the lecturer, lecture slides or projected computer screen and the whiteboard. Hence this application is also given the capability of broadcasting lecturer’s PC screen, third person point of view video stream of the lecturer and a digital whiteboard so that a remotely joined student can get a similar experience as of a student in the class. Students in a class room can interact with lecturer by asking questions. To provide such student lecturer interaction to remote students, we have introduced a live questioning facility with both audio and video capabilities. Also students and lecturers can use the whiteboard in the process of clarifying their questions and answers. When we consider about a school or a university, there are multiple such class rooms where lecturers/teachers teach students. Similar to that, this system is also capable of conducting multiple lecture sessions simultaneously without any interruptions to one another. Just like walking into the relevant classroom, remote student can join the particular session if the student is enrolled to that module and has the credentials to access the system. All these remote streaming features are related to this live stream component which is to be explained in this document.

In order to achieve Real-time live streaming, the system is built on top of WebRTC, an open source browser based communication technology which has the capability of Real-time

communications via APIs [1]. As its built on top of browsers, our application users do not have to install any additional libraries. Currently Google Chrome and Firefox and Opera web browsers highly supports WebRTC while other browsers are also gradually adapting to it.

One of the main goals we are trying to achieve here is the simplicity of the application increasing the user experience. If user is a lecturer, in order to start a lecture session, the only thing he/she has to do is enter session details such as Module Name. Title and Venue and click "Start Streaming" button. Rest will be handled by the system. Students just has to visit the URL generated and visible in the system in order to join a particular session.

Another aspect that we have to focus on when developing these type of applications is the security. We have taken high security measures to protect data and privacy from intruders. Only authorized users will be allowed to access the system so that remote joiners should have valid credentials to watch a live stream. None of the videos are downloadable by any means so that all the data will be within the system itself.

1.1 Background Literature

Virtual education is an emerging concept around the world. Students are starting to adapt to learn more productively through internet than traditional classroom due to many reasons. One of the main reasons is that every student has a different pace of catching-up with the teaching where some of them can be considered as fast learners while others may be slow. In traditional classroom scenarios, there is no solution to this fact. Another reason is the schedule flexibility. Through virtual learning, students can access their courses at any time anywhere with internet giving students the full control of their schedule of the day. Supporting the facts, universities provide students with the access to course materials via internet. Going beyond this, Real-time e-Learning Platform is a solution for the shortcomings in conventional education system mainly with universities. [2][3]

This proposed system is a web-based application where students can watch live lectures and interact with the lecturer and also playback previous lectures online. A tracking camera will be tracking the lecturer movements while recording and both lecturer's video feed and the lecturer's computer screen feed will be streamed. Also

this system contains features to interact with the lecture. Student can ask questions from the lecturer remotely via his or her webcam in real-time and also can use a virtual whiteboard that is provided with the system to describe any misunderstandings.

We have done our background study regarding this domain in order to get a good understating of technologies used, current available features, flexibility of existing systems etc. Some similar implementations we found along with this study are Eduscope, BigBlueButton, Apache OpenMeetings, MConf. These existing products has following features in common.

1) Live Streaming Web Camera
2) Desktop Capture
3) Document Share
4) Live Chat
5) Share Polls
6) Whiteboard

With related to the live stream component, I have studied the behavior of live streaming components of existing products closely in order to identify issues related to them. Main Issue found was the considerable latency of the live streams. Most of the cases, they had a considerable amount of delay where it cannot be addressed as a real-time scenario.

Existing e-learning platforms also require a very good bandwidth to work with. Specially, remote joiners should have a very good internet connection if they want to get a real-time experience. This issue is also to be addressed by our solution to provide users a near real-time experience.

Another common issue related with the existing applications was the performance and scalability. These applications are heavy and require considerable amount of resources to setup the infrastructures to run them. Also they are hard to scale due to their internal architecture. As one of our goals in this research is to provide a cost effective solution,

our system had to be capable of running in low performance computers and scalable as well.

A major outcome of this background study was getting to know about WebRTC. WebRTC stands for Web Real-Time Communication. It's a collection of APIs which allows direct connection between browsers providing the capability of transmitting any type of data. This direct communication allows WebRTC to reduce the latency between communications. [11]

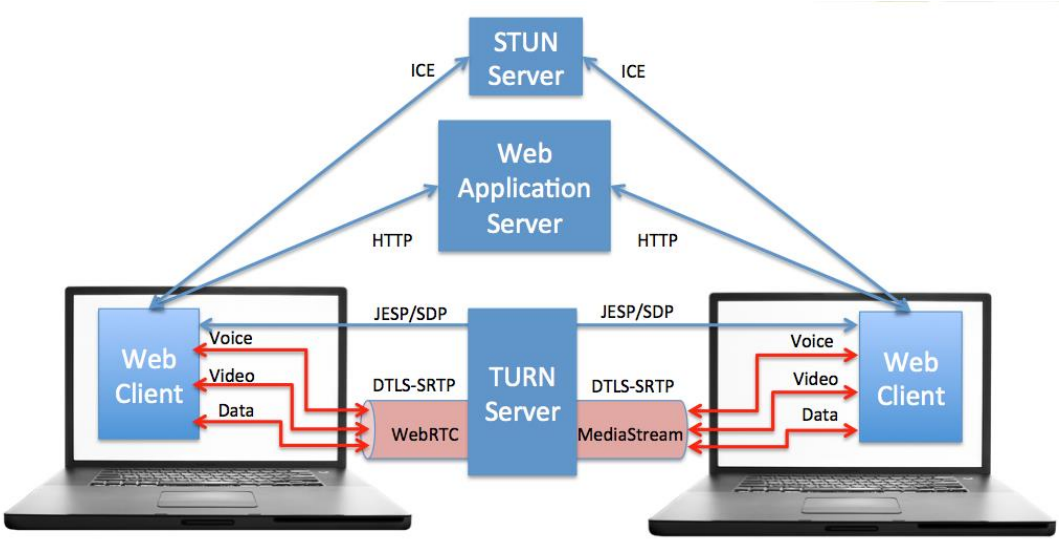

Figure 1-1 WebRTC direct browser connection establishment

Some of the major features provided by WebRTC which is helpful for our implementation are as follows.

1) Real-time communication
2) Supports Multiple Browsers. (Chrome, Firefox, Opera)
3) Supports Multiple Platforms. (Windows, Linux, MacOS, Android, IOS)
4) Supports mobile devices.
5) Built-in support for noise reduction and echo cancellation
6) High level Security and protocols.
7) Support modern video codes. (H.264, VP8)
8) Built-in capturing mechanisms.

Along with WebRTC, we came across with Janus, a general purpose WebRTC media server. Janus is an open source WebRTC server, which can be used as the core base for a video streaming application. It does contain a plugin for video conferencing which creates a video room and allow publishers to stream to the room while allowing subscribers to view those streams. Janus core also provides a feature to record media streams. It simply stores every RTP packet received by the server as it is without any changes. This process allows Janus to function consuming very less amount of processing power and memory. Later these recordings can be converted to a format that is supported by general media players by re-writing them to general media containers such as MP4 or WEBM [10].

1.2 Research Gap

When we consider about existing competitors of this field, the latency of the live streams is high where we cannot categorize their live streams as real-time. Main fact which has been affected to those is maintaining the quality of the live streams while delivering in real-time.

Also supporting live streams for almost all the devices including mobile devices and different browsers is another focus of this research.

Due to high performance infrastructure required to run the existing applications, it is not cost effective on business perspective. Building a high performance e-learning application which require less resources is quit a challenging task ahead of us.

None of the existing applications uses an adaptive bit rate mechanism during their live streams. Because of that, those systems are unable to deliver an uninterrupted stream to live viewers who has a low bandwidth internet connection.

As a solution for this, WebRTC introduces a feature known as simulcasting [9]. Simulcasting is a technique that stream sender encodes the same video stream in different resolutions and bitrates and sent them to a unit called SFU. SFU is responsible to decide which sender stream to forward to the receiver.

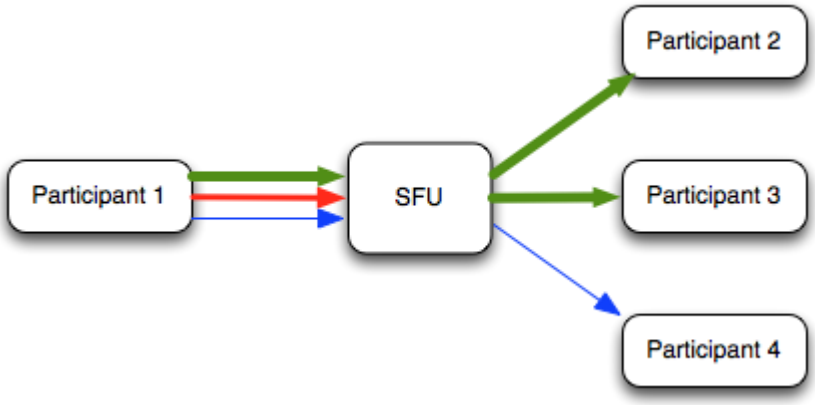

Figure 1.2.1 Conceptual model for video routing when using simulcast

As shown in the above figure, with all available stream variations, the SFU decide which quality stream is to be forward to each participant depending on their current internet connectivity.

1.3 Research Problem

The most common teaching and learning practice adopted by many enterprises has always been a classroom with one or more instructors and learners meeting physically and in real-time. But in this teaching model, there are several drawbacks which will be addressed as problems within this research.

A classroom based learning experience means the class schedule is predetermined and not subject to change. Students must shape their personal schedules around school instead of the other way around. If plans unexpectedly change or an emergency comes up, the student cannot adjust the class schedule to turn in the work at a different time. This is one of the main problems that this research will find solutions for.

Content non-reusability is another problem that can be found in classroom learning. Memorizing or writing all the necessary content while listening to a lecture is difficult. Therefore, students might miss many important points that the lecturer is pointing out during a lecture.

1.4 Research Objectives

1. Minimize Latency

   Delivering a stream in real-time is a challenging task due to several factors. Main obstacle is the source of the stream should have a high bandwidth internet connection to broadcast the streams. Also the client should have a high bandwidth to view the stream without any delay. Main objective of this component would be finding solutions to address these issues.

2. Multiple Device / Browser Support

   Students should be able to access the application from any of their devices and watch the streams. Also lecturers should be able to stream from their devices remotely. Therefore, the application should support most of the common devices and browsers.

3. Record Live Streams

   Live streams have to be recorded and stored in order to facilitate students with lecture playback feature. This process should be very light weight and less CPU consuming task.

4. High Scalability

   System should be capable of handle all the load coming onto it. If not, there should be the flexibility of scaling the system to accommodate all the requests.

5. High Quality Streams

   System should have the capability of delivering very high quality streams in bandwidths capable of student's internet connection to handle.

6. Adaptive Bitrates

   Live streams should support adaptive bitrates so that receivers can receive a uninterrupted video stream.

2. METHODOLOGY

2.1 Methodology

In order to develop this real-time application which is usable from multiple devices as laptops, tablets and smartphones we have chosen WebRTC technology. WebRTC is an open source project developed by google to make web browsers support peer-to-peer communications. On top of that, to implement our solution, we used Janus as the media server. Janus is a general purpose WebRTC server which is capable of implementing WebRTC media communications with browsers [10]. Backend application server is developed with Node.JS while frontend application is implemented using React.JS.

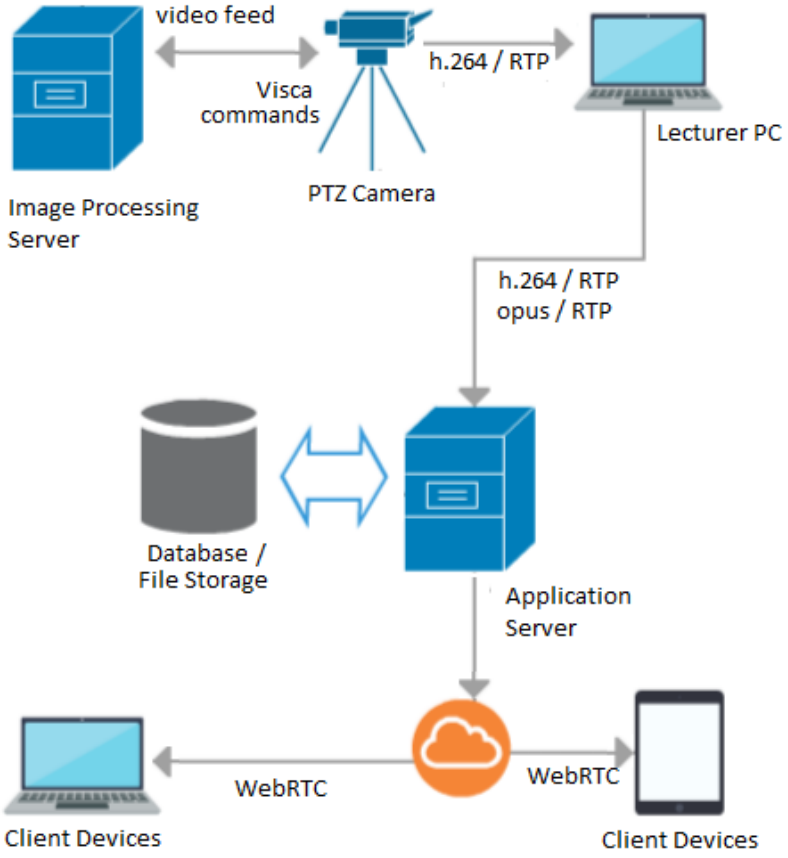

Figure 2.1.1 High Level Architectural Diagram

Main purpose of the live stream component is to transport video feeds from lecture's pc to student’s pc with minimum latency and support multiple streaming sessions at

once. Once the lecturer starts to stream, system creates a new streaming session and streams related to that session is broadcasted within the session. Once a peer joins to a specific session, they will receive the related streams. Lecturer's computer screen and a third-person perspective camera view of the lecturer are the two video feeds that

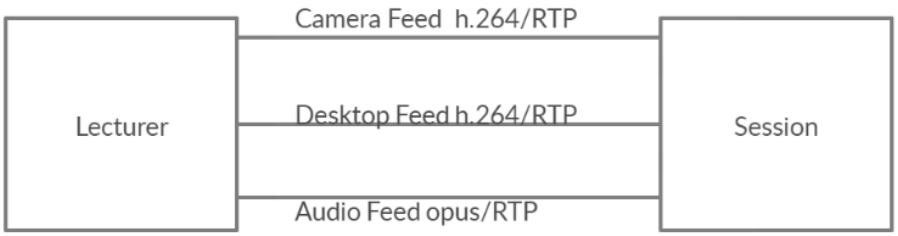

Figure 2.1.2 Feeds broadcasting within a session

will be broadcasted within a session. A feed from the lecturer's mic is also streamed to the session. Those video feeds are encoded with h.264 codec and audio feed is encoded with opus codec and send to the media server using RTP protocol.

Once media server receives the streams, they are transcoded and broadcasted via WebRTC.

For the purpose of playing back live streams, all the feeds are recorded and stored in the server as a custom file format **mjr**. A separate file will be created for each audio and video stream which contains structured raw RTP packets exactly as they are

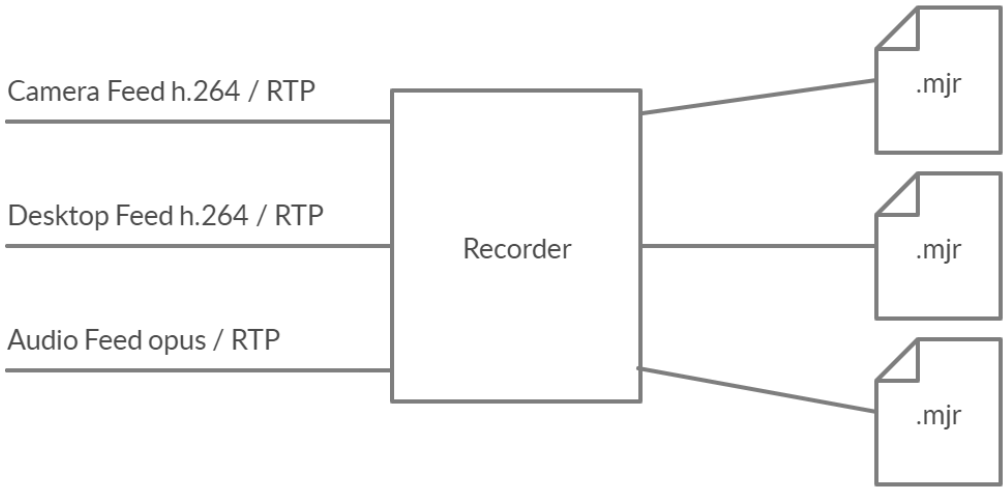

Figure 2.1.3 Feed recording process

received by the server. This operation allows the server to perform recording process as very less CPU intensive operation [10].

After a steam is completed, media frames from the recorded RTP packets are extracted and saved to a media container in a way that media applications can consume them. Here video recordings are converted into .mp4 file format and audio feed as converted into .opus file format.

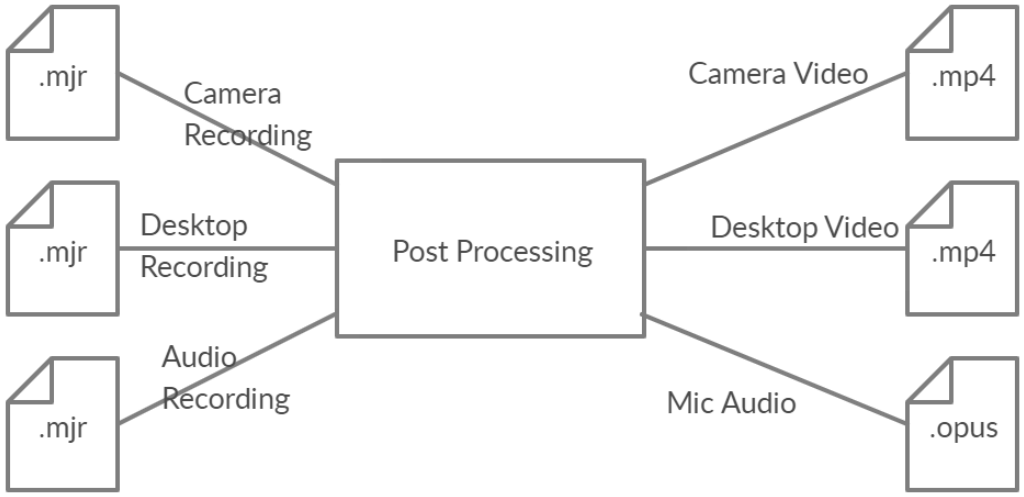

Figure 2.1.4 Post processing of recordings

2.2 Commercialization aspects

Key stake holders of this system are:

- University / Educational Institutes / Schools
- Lecturers / Teachers
- Students

Our main goal of the product in the aspect of commercialization is reduce the cost for the usage of the product than the competitors in the current market.

Real-time e-learning platform is commercialized based on two business revenue models.

- Cloud based revenue model
- On premises revenue model

2.2.1 Cloud based revenue model

This revenue model is used if the client agrees to install their application setup in an amazon cloud environment. Amazon will charge for monthly resource usage. Clients are charged with an additional service charge which will be a multiple of the entire usage bill from cloud service.

Total usage bill = Amazon cloud bill x billing factor

2.2.2 On premises revenue model

If the client does not want to use a 3rd party service to host their application and wants to setup the system on premises, this revenue model will be used. Here client will be billed monthly based on the user accounts created in the system.

2.3 Testing and Implementation

From the starting point of the project, we followed a waterfall model of development until the end of the project. Unit testing was done to each individual component developed while integration testing also carried out each time a feature integration was carried out to the system. We maintained a development environment which was our local devices which we use to develop as well as a AWS linux server as the production environment. Each component was developed and tested in both of these environments for expected outcomes.

During the implementation period, we used Github for the version controlling of our application. Two Git repositories were maintained as for the frontend developments and backend developments. Clearly identified features and independent development

of them in different branches allowed the features to be integrated to the main system successfully without major merge conflicts.

After a successful development stage, following are the outcomes of above implementations in the UI point of view.

Once lecturer's logged into the system, they can navigate to the Live stream page which is shown in Figure 2.3.1 Live Streaming UI for lecturer which is a very simple interface to work with. In order to start a new stream, lecturer has to enter the Module name, title and the venue and click Start Streaming button.

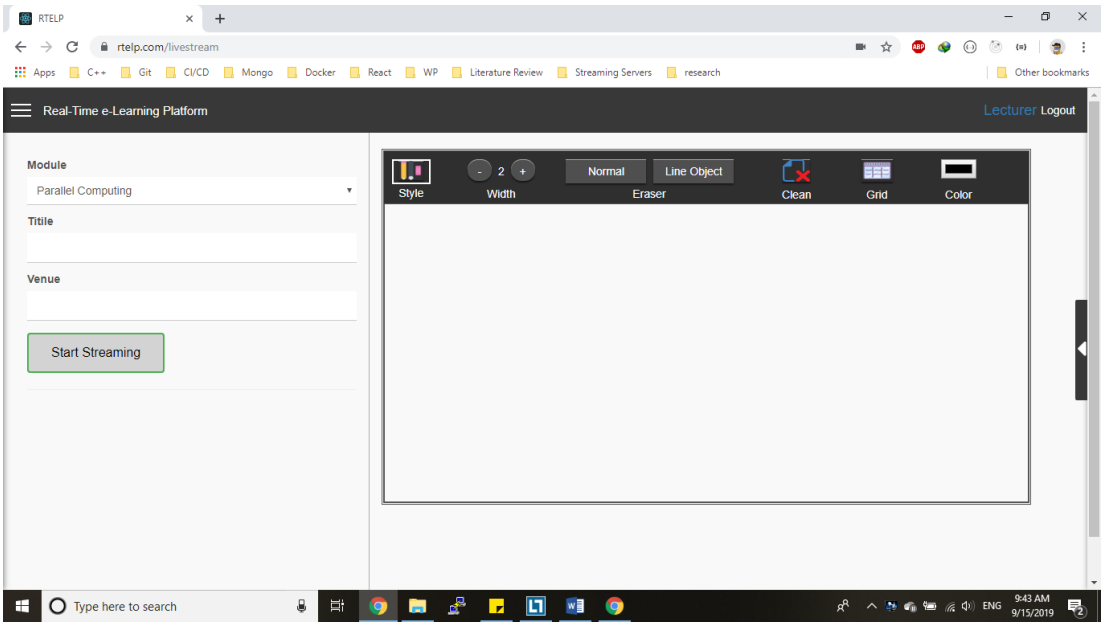

Figure 2.3.1 Live Streaming UI for lecturer

Once the start stream button is clicked, a menu will be shown to select which screen or application to be streamed. Figure 2.3.2 Screen Selection UI for Sharing screen shows the screens that lecturer can select during this process. If the lecturer PC is extended to multiple screens, those will be listed here. In this way lecturer can stream selected screen only while keeping rest of the screens for other uses.

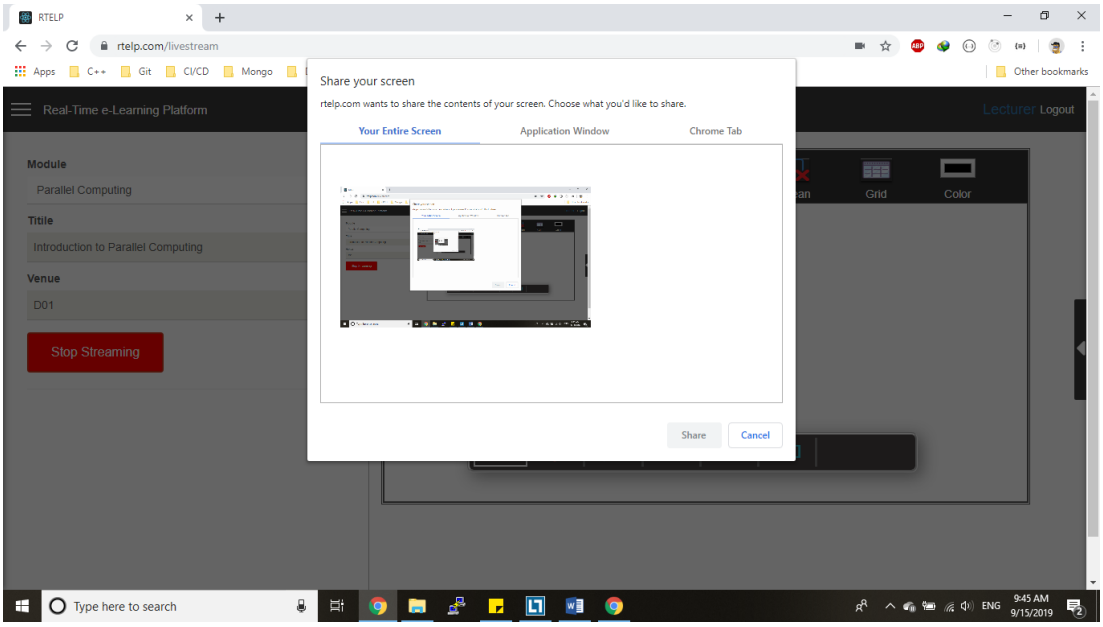

Figure 2.3.2 Screen Selection UI for Sharing screen

In the same window, it provides to select an application window to be capture and streamed (Figure 2.3.3 Application Selection UI for Screen Sharing). This option also maximizes the privacy of the lecturer during a live stream since only that particular application window will be captured.

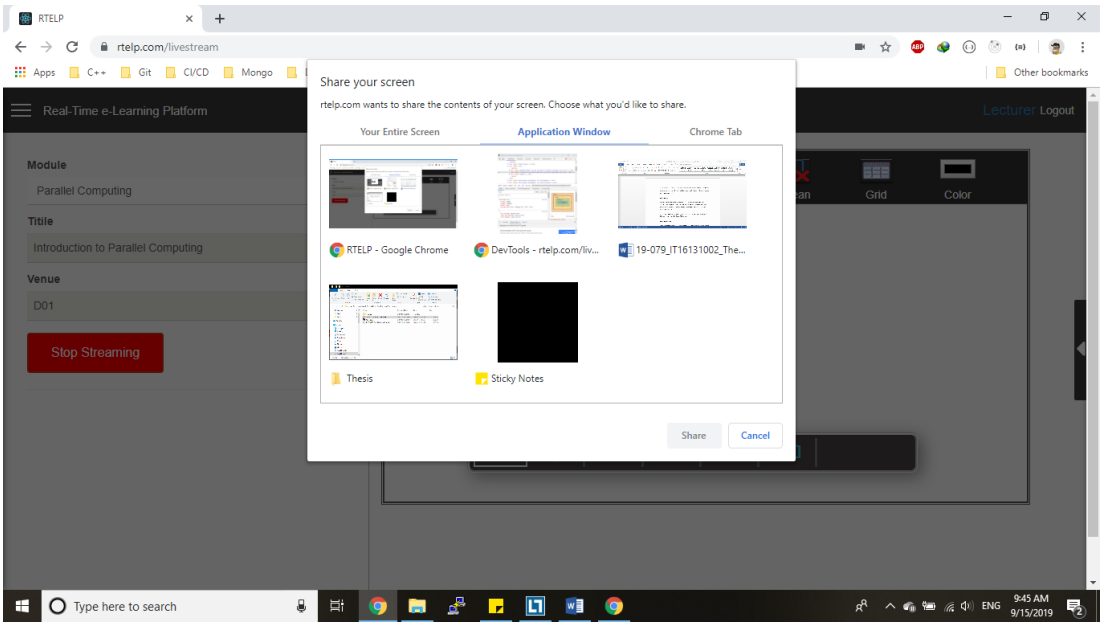

Figure 2.3.3 Application Selection UI for Screen Sharing

Once the screen to share is selected, system creates a new room for the session where students can join. Once streaming is ready, a preview of what lecturer currently share to students is displayed as shown in Figure 2.3.4 Preview of what lecturer shares on an ongoing stream4.

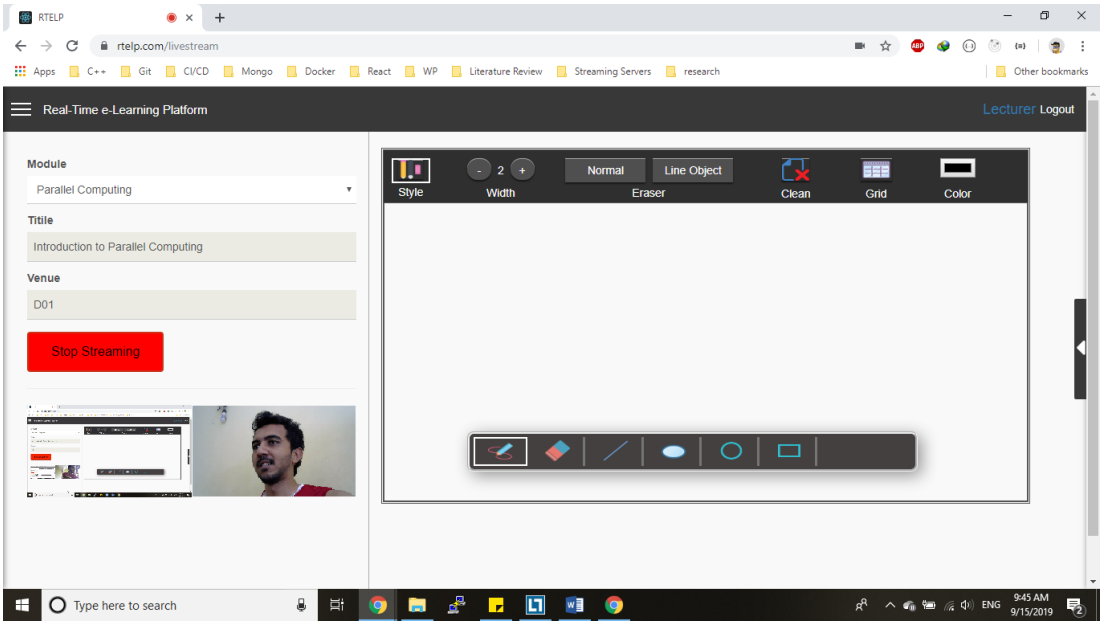

Figure 2.3.4 Preview of what lecturer shares on an ongoing stream

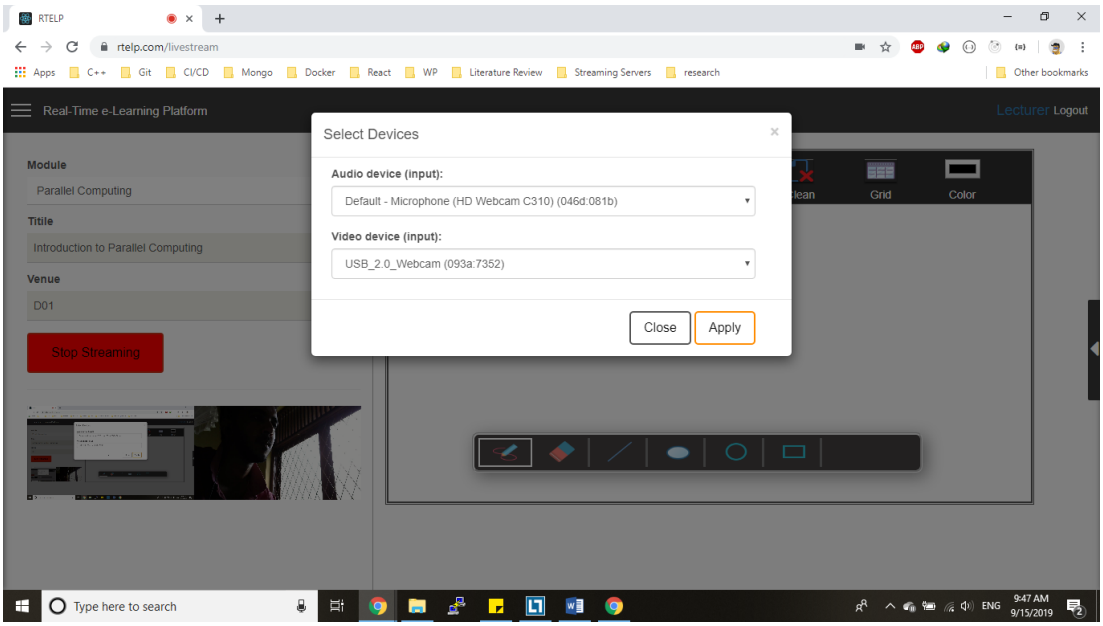

Figure 2.3.5 Input Device Selection Window

Lecturer can switch between streaming screens as well as cameras even during the stream which gives more flexibility and control to the lecturer about what they want to share.

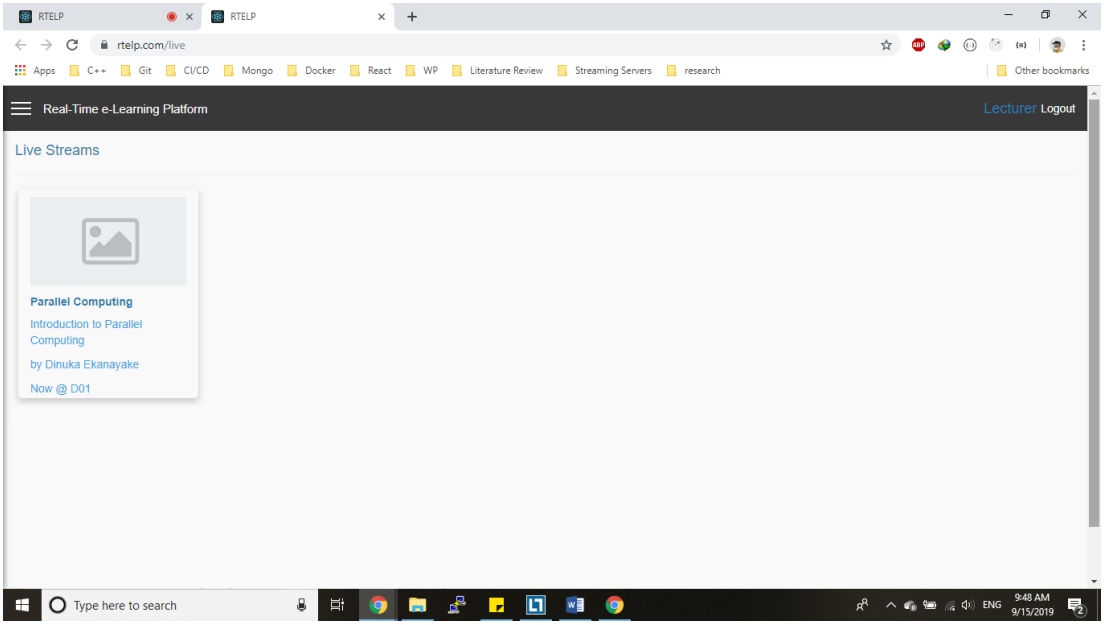

Figure 2.3.6 UI for Current Live Streams Available to Join

After a new live stream initiated, it will be shown in Live Now section as a thumbnail which contains the URL required to join the session. Once clicked, it will redirect student to the relevant session and streams will be loaded as shown in Figure 2.3.7 Student's view of a live stream.

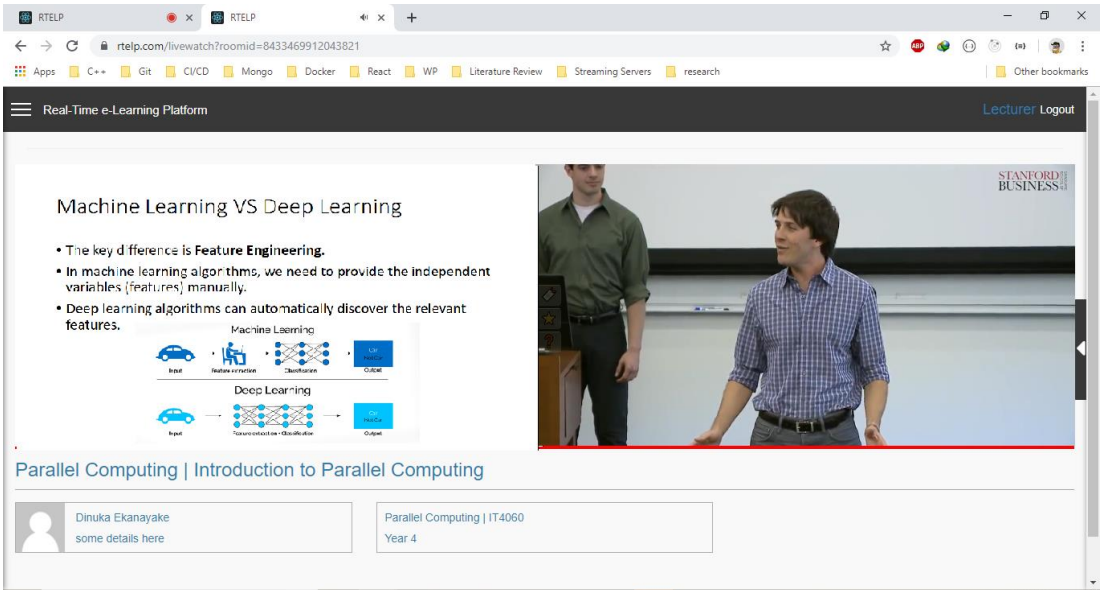

Figure 2.3.7 Student's view of a live stream

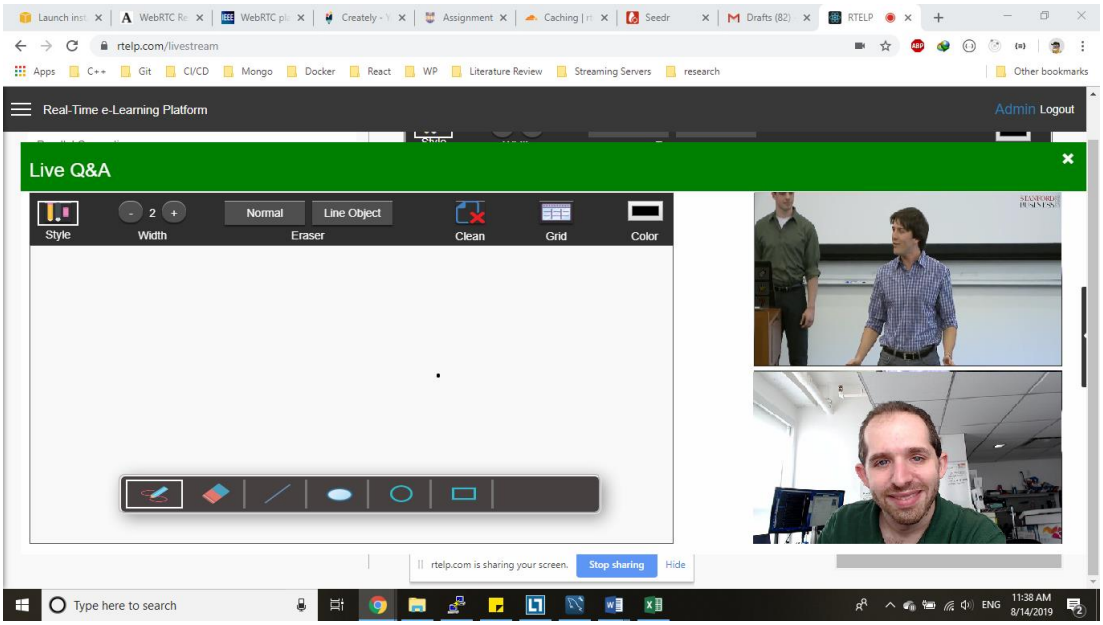

Figure 2.3.8 Live Q&A Sessions UI

Again when student is using live Q&A feature, WebRTC comes to play and start streaming students web camera to the lecturer. Whiteboard is placed alongside with the two camera previews of lecturer and the student. Both of the parties will be shown this window during a live Q&A. Student can use the whiteboard to explain and watch lecture explaining the solution to his or her problem in real-time during these live Q&A sessions.

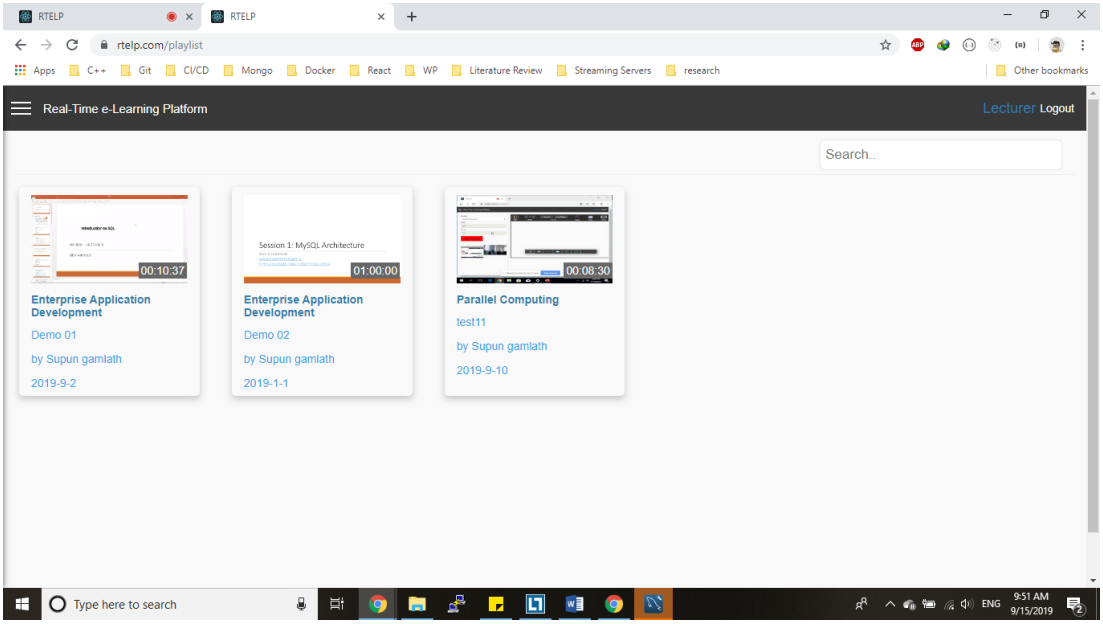

Figure 2.3.9 List of previously recorded lecture streams

Once the lecturer ends the session by clicking on Stop Streaming button, a background process will starts to process the recorded videos and will be published to the Recordings UI (Figure 2.3.9 List of previously recorded lecture streams9) once finished.

Those are the major UI implementations related to Live streaming component.

3. RESULTS AND DISCUSSION

We tested this process of the system using a AWS t2.micro instance (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only) running Ubuntu 18.04 LTS. System was capable of handling 10 simultaneous live stream sessions with 10 live viewers per each session only using 14% of the available CPU.

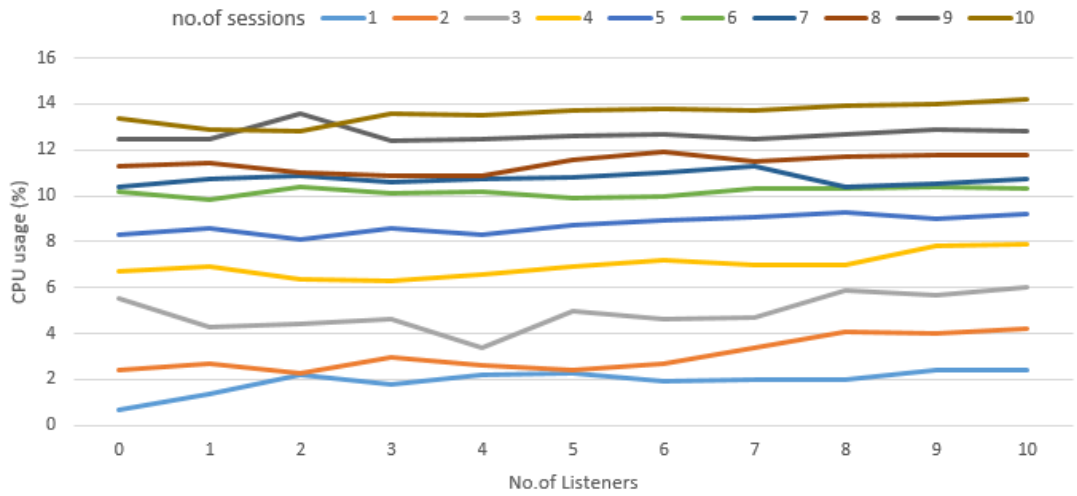

Figure 3.1 CPU consumption

Memory consumption for the same test scenario is given below.

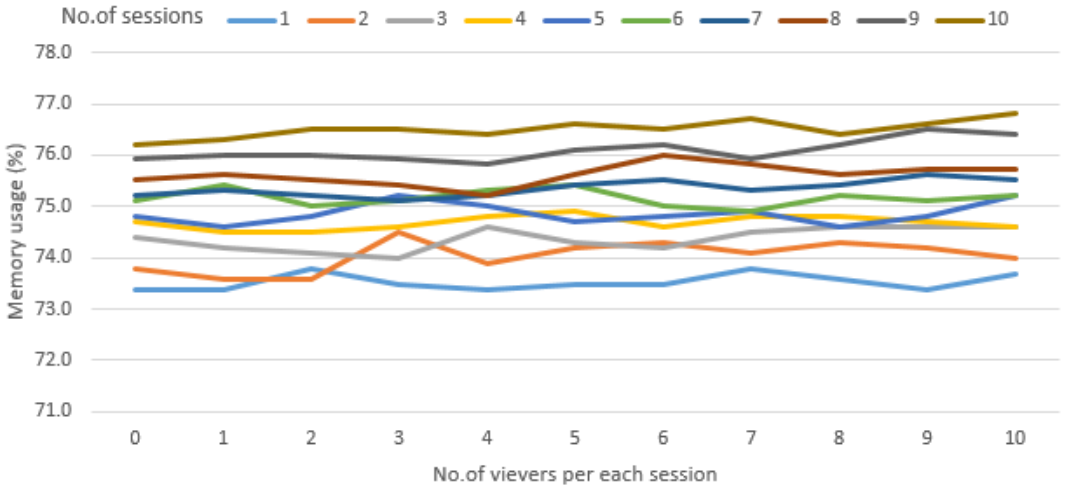

Figure 3.2 Memory Consumption

System consumed 3-4% of the available memory to cater the memory requirement of the platform for the above scenario.

In order to provide such a reliable stream, the server should have a better network bandwidth as well. The graph shown in below figure shows the network consumption during above testing scenario. During this period 10 live sessions were running parallely and from them there were 20 streams tranfered in to the server. The maximum network bandwidth which has been consumed was 19.2MB/s.

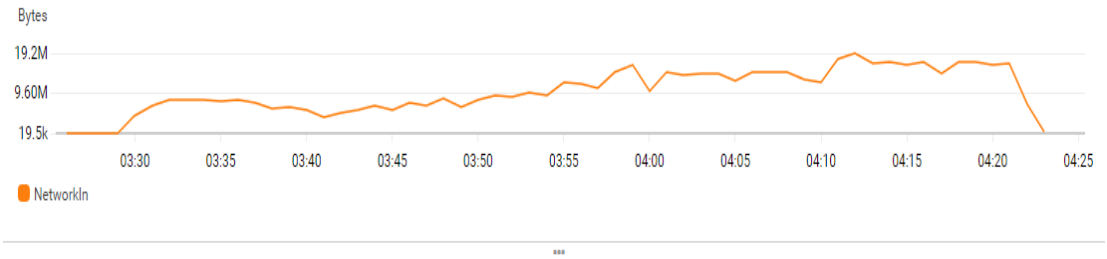

Figure 3.3 Network In of AWS instance

Similarly, take a look at the Network out graph of the same instance for the same scenario. Average of around 14MB/s has been consumed as well as for the network out.

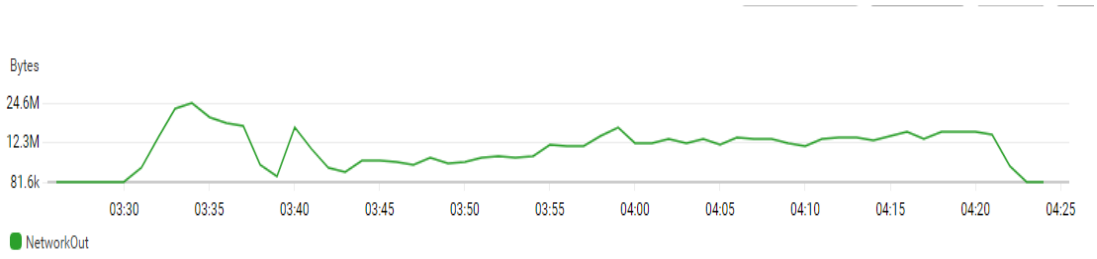

Figure 3.4 Network Out of AWS instance

Those network graphs tend to grow rapidly with the increase of parallel sessions and connected student count as obvious. This is a must bear cost for the system since the application is totally based on internet.

A 60 minutes long video recording of 720p screen capture and 1080p camera takes approximately 600MB and 1000MB of storage respectively. Audio recording is 20MB per 60 minutes which makes the system to consume approximately 1620MB of storage per 1 hour long session to store the original streams. Cost for the storages also has to be bear by the client. Thses numbers are normal for such a quality video. But it is also possible to reduce the recording sizes by compromising some of the video quality as per the liking of the client.

4. CONCLUSION

Implementing an actual real-time e-learning platform is not possible since anyway along the pipe line, we have to do some computational functions in order to deliver media across different clients. But we were able to achieve a near real-time solution for our problem reducing the latency to fractions of a second. All these results were possible because of using WebRTC as the base for the solution. For establishing near real-time remote communications, WebRTC is highly recommended than other technologies out there.

Looking ahead, next steps for this solution will be introducing adaptive bitrate capability during a live stream. This will help the clients to access live streams at any condition of their network bandwidths.

5. REFERENCES

[1] S. Ouya, C. Seyed, A. B. Mbacke, G. Mendy and I. Niang, "WebRTC platform proposition as a support to the educational system of universities in a limited Internet connection context," 2015 5th World Congress on Information and Communication Technologies (WICT), Marrakech, 2015, pp. 47-52. doi: 10.1109/WICT.2015.7489643

[2] Chris Curren, "*Strategies for e-Learning in Universities*",CSHE.7.04,2004

[3] H. M. Truong, "Integrating learning styles and adaptive e-learning system: Current developments, problems and opportunities," Computers in Human Behavior, 2015

[4] I U Cooray, W & M P Abhayawickrama, D & Ragel, Roshan. "Real time interactive lecture delivery system." 91-96. 10.1109/ICIAFS.2010.5715641.

[5] McCrohon M, Lo V, Dang J, Johnston C. "Video streaming of lectures via the Internet: An experience" ERIC Clearinghouse; 2001 Dec 9.

[6] D. Ehlers, "*Quality in e-Learning from a Learner's Perspective*", 2004,7

[7] Yueshi Shen, "*Live Video Transmuxing/Transcoding:FFmpeg vs TwitchTranscoder Part I "* https://blog.twitch.tv/live-video-transmuxing-transcoding-ffmpeg-vs-twitchtranscoder-part-i-489c1c125f28. [Accessed: 2019-05-13]

[8] Kurento, "Kurento Media Server Development Documentation" https://doc-kurento.readthedocs.io/en/6.10.0/#dev-docs [Accessed: 2019-05-13]

[9] Grozev, Boris & Politis, George & Ivov, Emil & Noel, Thomas & Singh, Varun. (2017). Experimental Evaluation of Simulcast for WebRTC. IEEE Communications Standards Magazine. 1. 52-59. 10.1109/MCOMSTD.2017.1700009.

[10] Meetecho s.r.l, “Meetecho Janus Recordings”, https://janus.conf.meetecho.com/docs/recordings.html [Accessed: 2019-08-06]

[11] Zafran M R M ,Gunathunga L G K M, Rangadhari M I T ,Gunarathne M D D J ,Kuragala K R S C B, Mr Dhishan Dhammearatchi,” Real Time Information and Communication Center based on webRTC”,2016