WebBoard: An interactive tutoring system

Ashwin Vasani
Ira. A Fulton School of Engineering, Arizona State University
akvasani@asu.edu

Ravinsingh Jain
Ira. A Fulton School of Engineering, Arizona State University
rvjain@asu.edu

Sagar Kalburgi
Ira. A Fulton School of Engineering, Arizona State University
skalburg@asu.edu

Abstract -

This project is to meet the necessary requirements of online lectures and to make it easier for the instructors and the students to communicate more effectively and interactively, especially for remote students. WebBoard is an integrated tutoring application. During the presentation, the instructor can write on the application interface called WebBoard, the content of which is replicated to all the students in their WebBoards. It comes with a Question & Answer feature which allows students to ask questions in a manner that can be accessed by students participating in the class. The instructor can easily switch among the slides, video and the WebBoard. While elaborating on the concepts, the instructor can annotate the content of slides, without modifying them which makes it easier for students to understand the concepts. The lectures are recorded in the class rooms and are available to all the students later on the cloud.

Keywords — webRTC , Vlab, Node.js, Video conferencing, Screen sharing, WebBoard, tutoring, VP8.

INTRODUCTION

a. **Problem:** Rigid Nature of the hybrid classes and online lectures, current hybrid/online classes and challenges faced by students/instructors.

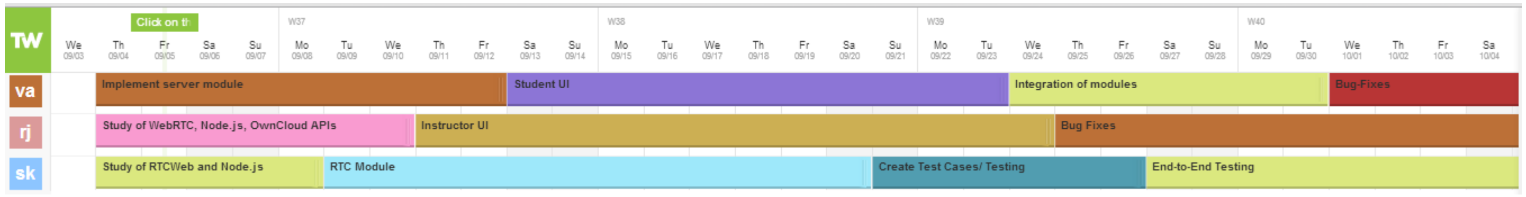
b. **Why it is important:** It gives a single system which allows the students to communicate with the instructors and other students. The lectures will be more interactive as the professor can ask some questions to the students who are remotely accessing the lectures to answer or vice-versa. The videos of the lectures would be stored on cloud so that it is easily accessible to students to recollect concepts taught in the class room. Moreover they get a classroom feel while watching the video streaming. Students can take their personal notes for the slides or video which they can store for later use. Instructor can use the Web Board functionality to annotate on some concepts which they might do on a White board in classrooms so that even her explanatory notes can be available to all the users after the class on the cloud.

c. **Technologies used:** Cloud, WebRTC API, Node.js, JavaScript, CSS, HTML ~~and (PHP).~~

d. **Expected outcomes of this projects:**

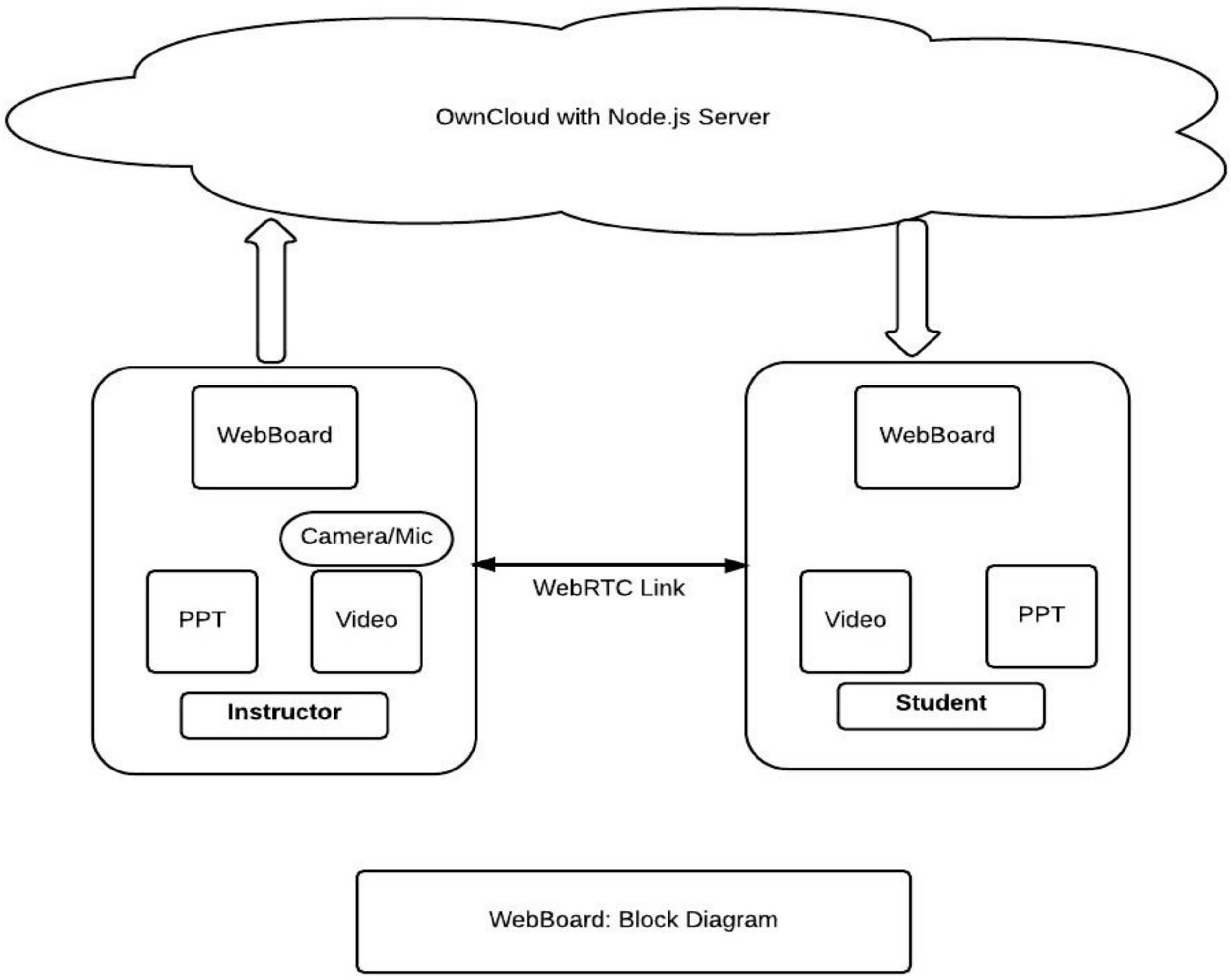
1) Software would have the following main features -
   a) Video recording and storing in cloud
   b) WebBoard (Similar to White Board) for instructors[completed]
   c) Paper Presentation (PPT) sharing tool [completed]
   d) ~~Note Taking Tool for students.~~
2) On instructor side: She can start the session. Session can be started on Tablet using web or Android browser. She can upload **PPT** or PDF file. This **PPT** or PDF is internally converted into images. She can draw on the image and undo them. She can also switch to White Board Mode where she can draw diagrams or write notes. She can view all the students who are joining the sessions. [completed]
3) On Student Side: They can join the session initiated by the instructor. They can also view the pre-recorded session. On a live session, they can ask questions using the WebBoard. If allowed by the instructor, they can write on the WebBoard of the instructor and their question becomes available to everyone. Though the direct support of annotating on WebBoard by students is not allowed in our UI implementation, we have exposed the APIs to enable this feature .Students can view the live streaming of the lectures with one-to-many broadcasts from the instructor. [completed]
4) Instructor can share the VM (Vlab) screen with students for live coding sessions.

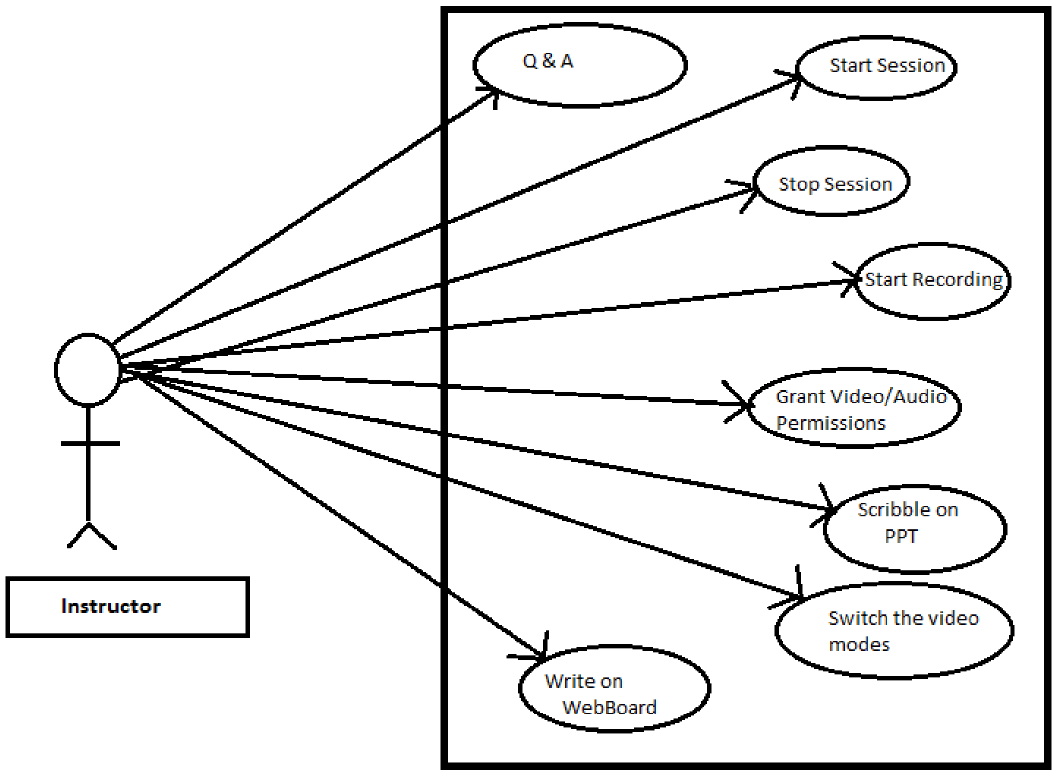
e. **Project management plan (timeline, and group members, etc.)**

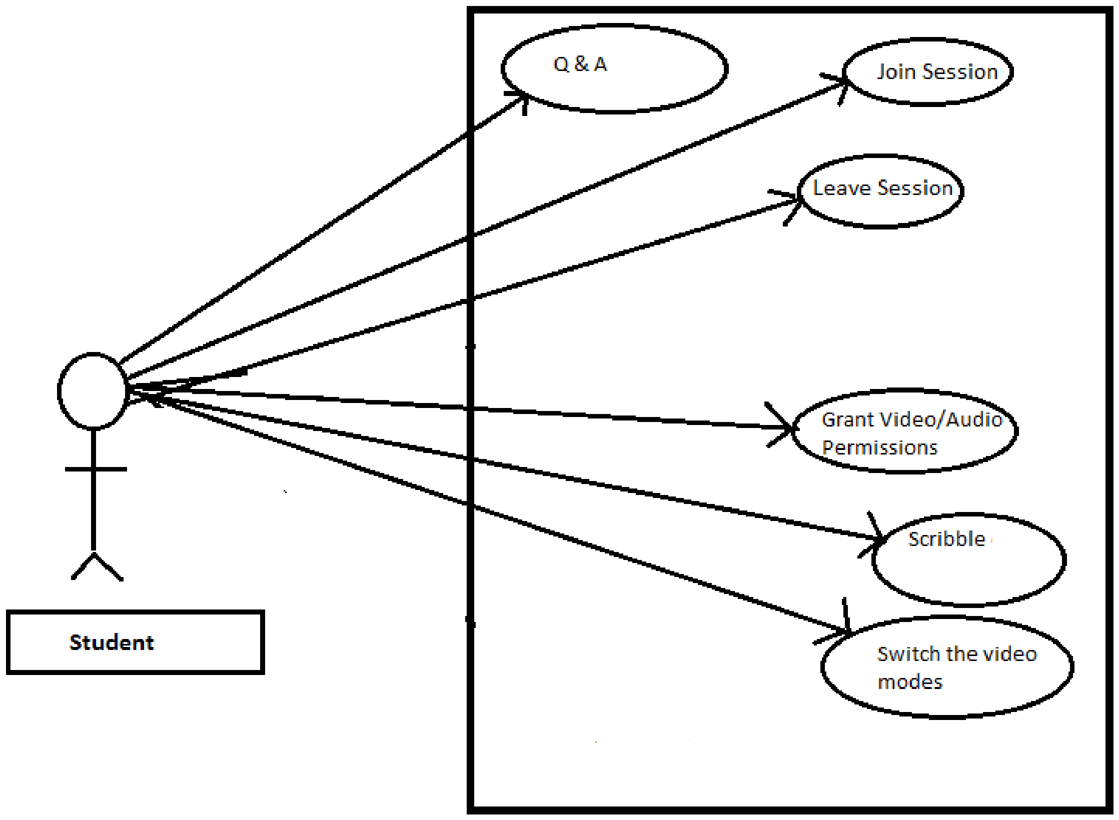


System Models

a) System Model




Q & A
Start Session
Stop Session
Start Recording
Grant Video/Audio Permissions
Scribble on PPT
Switch the video modes
Write on WebBoard
Instructor


Q & A
Join Session
Leave Session
Grant Video/Audio Permissions
Scribble
Switch the video modes
Student

b) Software

Tools: ~~Eclipse or~~ Sublime Text, Emacs, Git

APIs: ~~OwnCloud,~~ WebRTC API, Node.js, Vlab, Heroku

Language: JavaScript, ~~PHP,~~ HTML & CSS.

PROJECT DESCRIPTION

a) Project Overview

a. Task 1: Study WebRTC, ~~Own Cloud,~~ Vlab and Node.js - We shall research on the above topics and start with implementing some of them. [completed]
b. Task 2: Implement server module - We shall install Node.js server on localhost with some server side business logic. [completed]
c. Task 3: RTC Module - Implement WebRTC module. [completed]
d. Task 4: Instructor GUI: Implement Instructor Graphical user interface by using web languages. [completed]
e. Task 5: Student GUI: Implement Student Graphical user interface by using web languages. [completed]
f. Task 6: Create Test Cases.
g. Task 7: Integration of modules [completed].
h. Task 8: Test the application.
i. Task 9: Fixing Bugs.

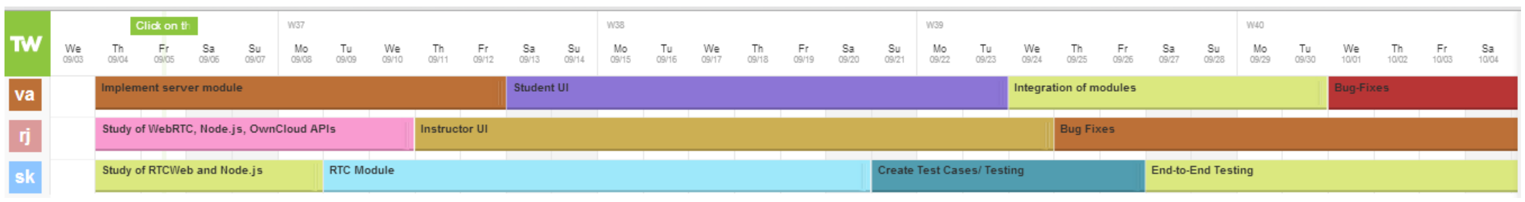
b) Project Task Allocation

Name/Task	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9
Ravinsingh Jain	Co-owner	Co-owner	Co-owner	Owner	Co-owner	Co-owner	Co-owner	Co-owner	Owner
Ashwin Vasani	Co-owner	Owner	Co-owner	Co-owner	Owner	Co-owner	Owner	Co-owner	Co-owner
Sagar Kalburgi	Owner	Co-owner	Owner	Co-owner	Co-owner	owner	Co-owner	Owner	Co-owner

c) Deliverables

It is a web application which enables the instructor to use WebBoard in order to give lucid presentations. The instructor can seamlessly switch among the presentation slides, explanatory notes and video. The students, especially those who use remote access, would benefit greatly since they can ask real time questions using the WebBoard. Additionally, application can be extended to add a feature where student can also take separate personal notes and store them in cloud. She may also share her notes with her classmates, leading to healthy exchange of ideas.

d) Project Timeline



Risk Management of the Project

Risk Factor	Description	Workaround
Bandwidth	Requires high bandwidth	Can optimize the camera resolution accordingly.
Security	Network security for the streaming	Private secure stream can be deployed.
WebRTC for native application	It is under development	AppRTC and JNI can be used at some level

Progress till now

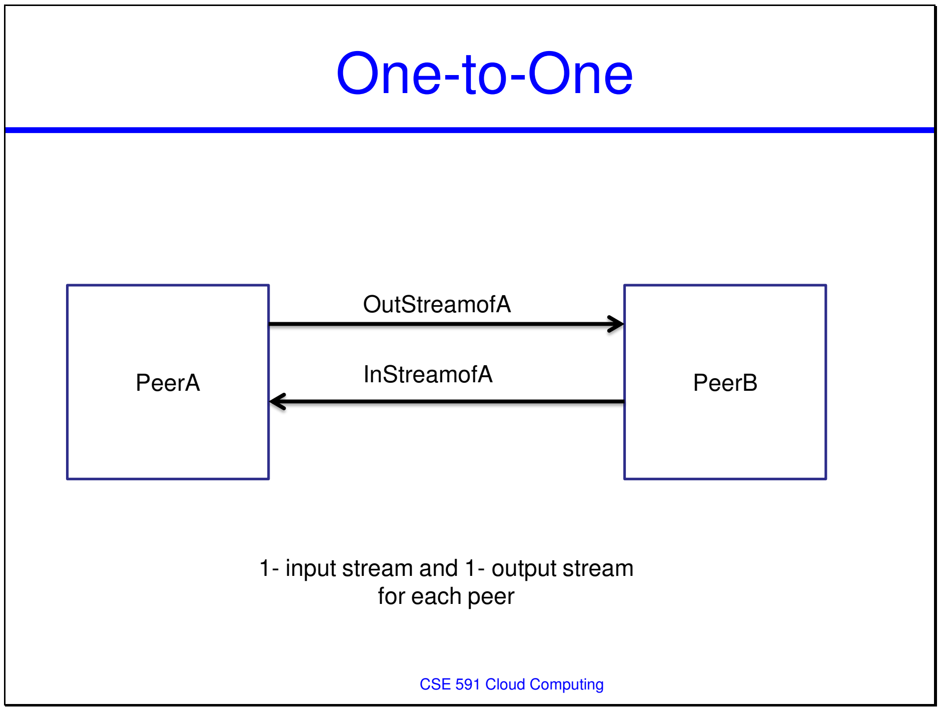
Investigations/Research:

Technologies that were studied for project requirements:

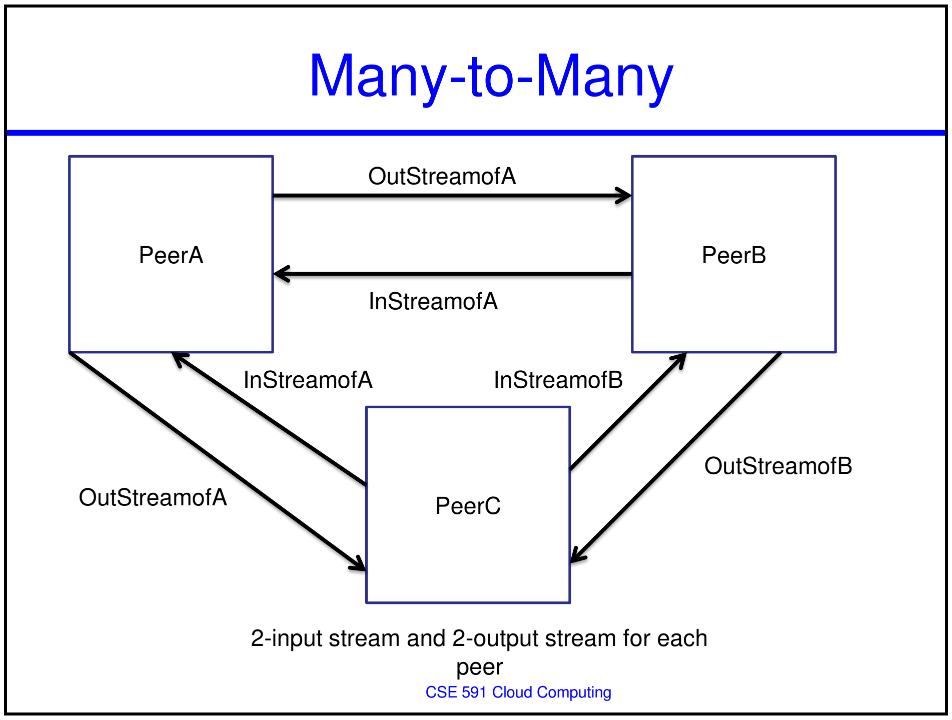
WebRTC APIs for video and audio, ICE, STUN &TURN servers for remote stream management, Node.js for hosting server, Communication architectures: one-to-one, one-to-many, many-to-many and broadcast, Bootstrap for User Interface design and MongoDB for the database of valid user credentials.

Following research was done for WebRTC based video sharing between multiple peers:

During the investigation, we found some potential challenges for our use-case using WebRTC. Presently, WebRTC creates two streams for audio/video for each peer (Please ref. the diagram below). This model is feasible only if there are two users.



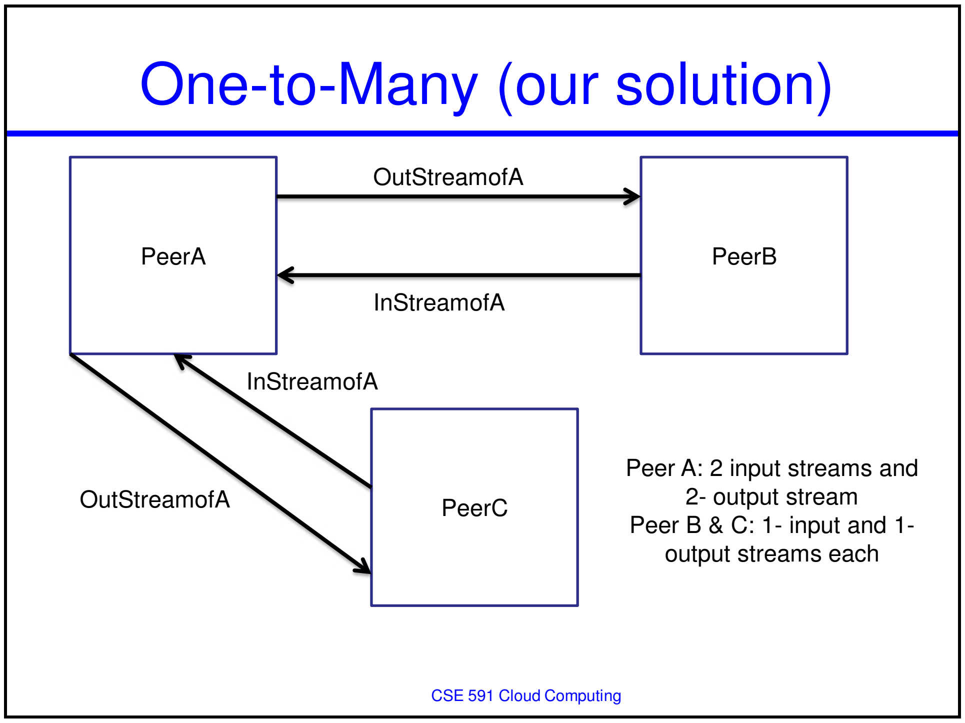
In many-to-many approach (Please ref. the diagram below), a complex mesh is created amongst the peers which causes a very heavy network and CPU usage. This model is not feasible in our case since we are targeting that the user might be having a handheld device like smart-phone or tablet. Such devices don't have enough power to handle such a heavy requirement.



To overcome the above challenges, we tried implementing our own design. We have hacked the camera stream which we got from the WebRTC's getUserMedia call and reduced the quality of the video frames. Later we sent this video frame to a node.js server using socket.io and client renders from node.js to canvas using webRTC standard APIs. Here server also adds many headers to the packets and fragmentation of packets on network causes a bad and unacceptable performance. Hence we switched to one-to-many approach.

In one-to-many approach (Please ref. the diagram below), the total number of streams created by the instructor will be equal to twice the number of users, and only two streams will be created at the student end. This means that the instructor module should be running on the device with maximum resources but this does not apply to the students.

The maximum bandwidth of each A/V RTP stream is 1MBPS and for each peer it requires 2MB bandwidth. For future enhancements in WebRTC, we envisaged that adding WebRTC support on the server platform like node.js would prove beneficial for many-to-many and one-to-many use-cases. In this case, the server will act as a multiplexer and all the peers would connect to the server. Instead of having multiple A/V streams for all peers, sharing single video stream to multiple users can also be one of the solutions to this problem.



One-to-Many (our solution)

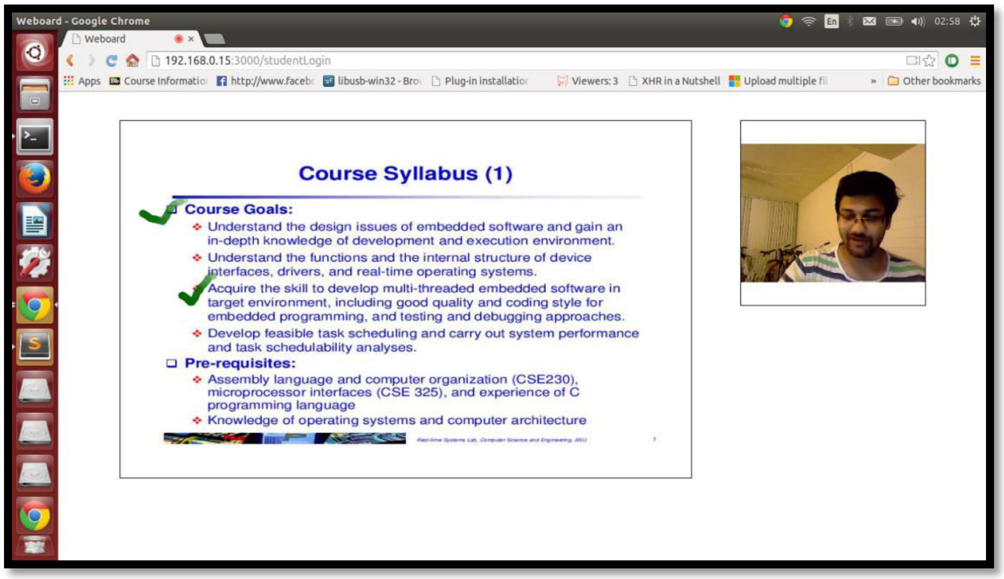
Peer A: 2 input streams and 2- output stream
Peer B & C: 1- input and 1- output streams each

CSE 591 Cloud Computing

Implementation Details:

- Bootstrap was used for interactive UI design.
- Heroku cloud is an application platform known for its adaptability.
- Login credentials would be stored in MongoDB which is a highly scalable database.
- Presently we are uploading PDFs, converting into images to display them in canvas as a slideshow.
- Used meeting.js for one-to-many WebRTC implementation and firebase.js was used for WebRTC signaling.
- Using HTML5 canvas and sections for encapsulating videos elements.
- Currently we are using only single classroom but it can be scaled to multiple classrooms. Also, it is supporting event based dynamic stream addition and removal.

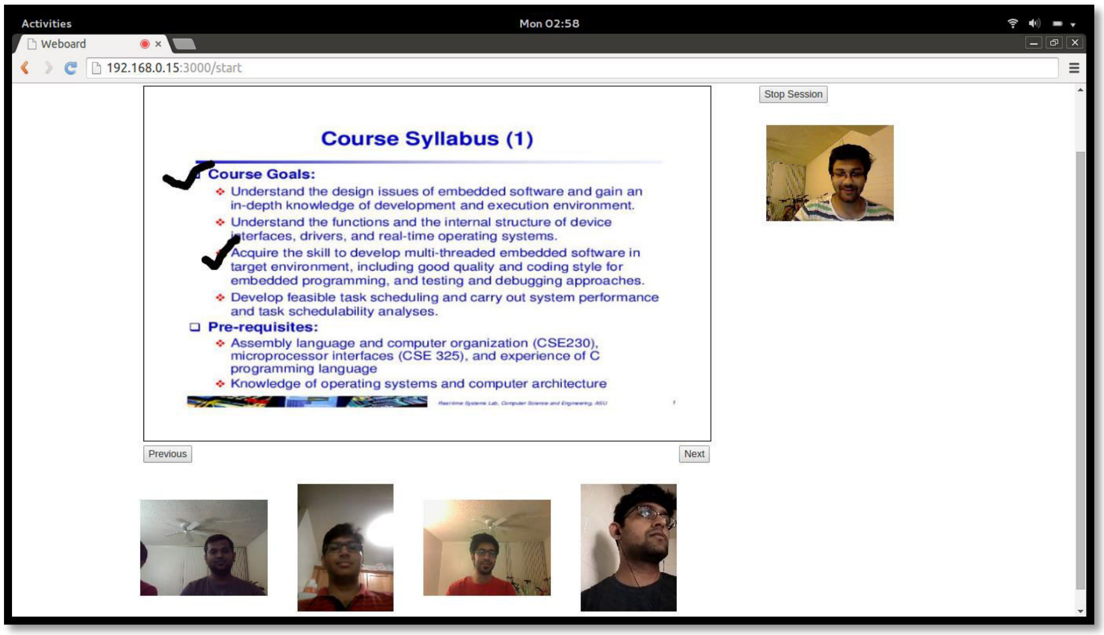
Demo Snapshots:



Course Syllabus (1)

- Course Goals:
  - Understand the design issues of embedded software and gain an in-depth knowledge of development and execution environment.
  - Understand the functions and the internal structure of device interfaces, drivers, and real-time operating systems.
  - Acquire the skill to develop multi-threaded embedded software in target environment, including good quality and coding style for embedded programming, and testing and debugging approaches.
  - Develop feasible task scheduling and carry out system performance and task schedulability analyses.
- Pre-requisites:
  - Assembly language and computer organization (CSE230), microprocessor interfaces (CSE 325), and experience of C programming language
  - Knowledge of operating systems and computer architecture

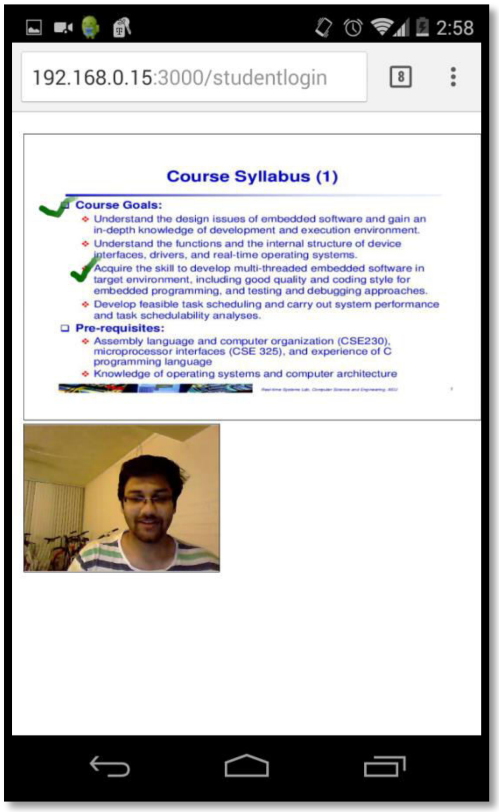
Figure 1: Instructor Screen



Course Syllabus (1)

- Course Goals:
  - Understand the design issues of embedded software and gain an in-depth knowledge of development and execution environment.
  - Understand the functions and the internal structure of device interfaces, drivers, and real-time operating systems.
  - Acquire the skill to develop multi-threaded embedded software in target environment, including good quality and coding style for embedded programming, and testing and debugging approaches.
  - Develop feasible task scheduling and carry out system performance and task schedulability analyses.
- Pre-requisites:
  - Assembly language and computer organization (CSE230), microprocessor interfaces (CSE 325), and experience of C programming language
  - Knowledge of operating systems and computer architecture

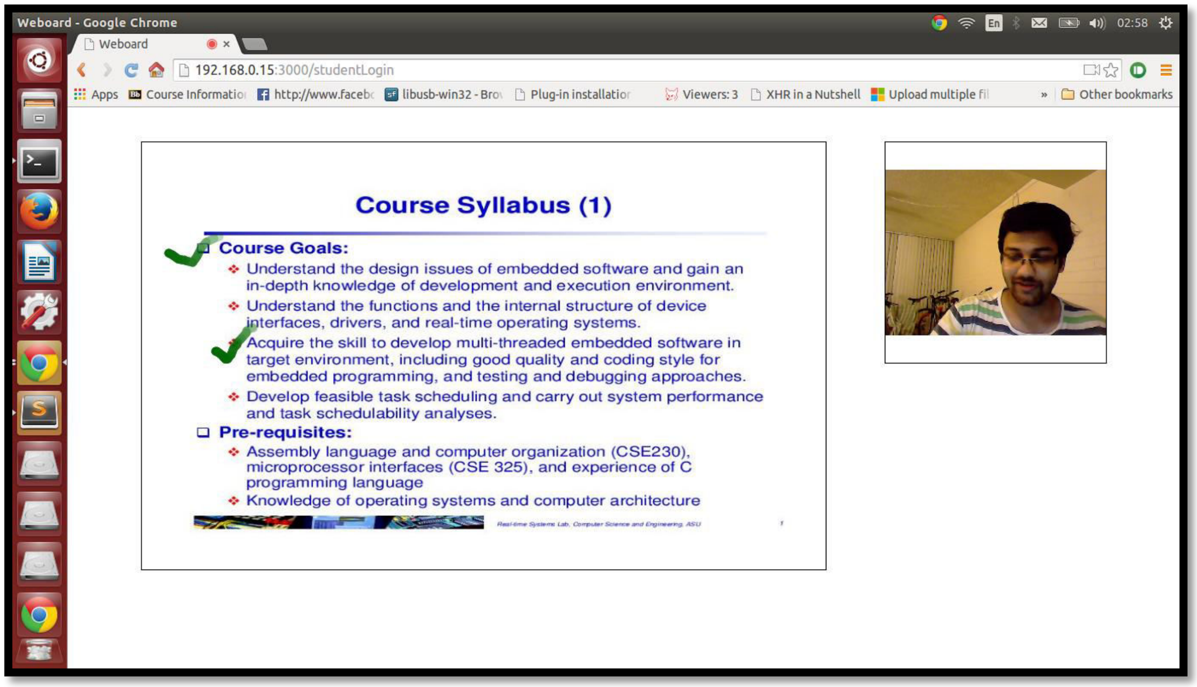
Figure 2: Interactive Class



Course Syllabus (1)

- Course Goals:
  - Understand the design issues of embedded software and gain an in-depth knowledge of development and execution environment.
  - Understand the functions and the internal structure of device interfaces, drivers, and real-time operating systems.
  - Acquire the skill to develop multi-threaded embedded software in target environment, including good quality and coding style for embedded programming, and testing and debugging approaches.
  - Develop feasible task scheduling and carry out system performance and task schedulability analyses.
- Pre-requisites:
  - Assembly language and computer organization (CSE230), microprocessor interfaces (CSE 325), and experience of C programming language
  - Knowledge of operating systems and computer architecture

Figure 3Application demo on Android Chrome Brower



Course Syllabus (1)

- **Course Goals:**
  - Understand the design issues of embedded software and gain an in-depth knowledge of development and execution environment.
  - Understand the functions and the internal structure of device interfaces, drivers, and real-time operating systems.
  - Acquire the skill to develop multi-threaded embedded software in target environment, including good quality and coding style for embedded programming, and testing and debugging approaches.
  - Develop feasible task scheduling and carry out system performance and task schedulability analyses.
- **Pre-requisites:**
  - Assembly language and computer organization (CSE230), microprocessor interfaces (CSE 325), and experience of C programming language
  - Knowledge of operating systems and computer architecture

Real-time Systems Lab, Computer Science and Engineering, ASU

Figure 4: Student UI on Desktop Browser

Conclusion

Considering the limitations of the current online learning technologies, our WebBoard system aims to better the classroom experience, thus ensuring that the students and the instructors are on the same page, especially for remote students.

Future prospects:

- A Student may use a real time language translator in case she doesn't know the medium of instruction.
- Hearing impaired students can be provided with subtitles.
- The application can be integrated with native applications
- Many to many interaction can be provided.
- Desktop sharing on cloud using WebRTC .

Acknowledgment

We are grateful to Huijun Wu and Bing Li who guided and motivated us to take up this project.

References

[1] http://www.webrtc.org/

[2] http://tools.ietf.org/html/draft-ietf-rtcweb-data-channel-07

[3] https://www.webrtcexample.com/

[4] https://myasucourses.asu.edu

[5] Lezli Kubo, "The Rise of Online Education: Exploring the Phenomenon"

[6] http://nodejs.org/

[7] www.htmlrocks.com

[8] http://www.tmcnet.com/tmc/whitepapers/documents/whitepapers/2013/9187-webrtc-reach-web-with-new-conversation-experience.pdf