**Computer Networks - II**
*prof.* kotaro kataoka
Department of Computer Science and Engineering
Indian Institute of Technology Hyderabad

# Computer Networks 2

Nalari Sushma - CS19BTECH11006

Ambati Sanjay Krishna - CS19BTECH11013

Uppala Sehouriey - CS19BTECH11015

Lingareddy Harshavardhan Reddy - CS19BTECH11023

Kollimarla Sri Teja - CS19BTECH11002

Langaru Likitha - CS19BTECH11008

May 3, 2022

**Computer Networks - II**
*prof.* kotaro kataoka
Department of Computer Science and Engineering
Indian Institute of Technology Hyderabad

# Contents



# 1 PROJECT GOAL

- Video streaming is a continuous broadcast of a live or recorded video from a server to a client.
- In file transfer protocols, the client must download the video in order to view it, whereas video streaming allows the user to view it online.
- Our goal is to develop this video streaming service using RTSP (Real-Time Streaming Protocol) and RTP (Real time Transfer Protocol).

# 2 SOFTWARE OVERVIEW

## 2.1 SOCKET

A socket is one endpoint of a two-way communication link between two programs running on the network. So we use socket properties to achieve connection between server and client.

## 2.2 SERVER

Server is now one endpoint of communication which has a socket. Server creates a socket and attaches it to Ip address and port number to establish a server side connection. The Listening method on the server starts waiting for the client.

## 2.3 CLIENT

Client is now the other endpoint of communication. Many clients can connect to the server thus establishing one to many connections from server to clients. A client needs to connect to the server using the IP and port number(s) from which the server is connected to the network.

## 2.4 PROTOCOL (RTSP AND RTP)

- We used RTSP and RTP protocols to implement this streaming service.
- RTSP is a presentation-layer protocol that allows users to control media servers through the use of pause and play functionalities.
- RTP is a transport protocol which is used to transfer data.
- RTSP uses RTP (over UDP) for stream and controls using TCP.

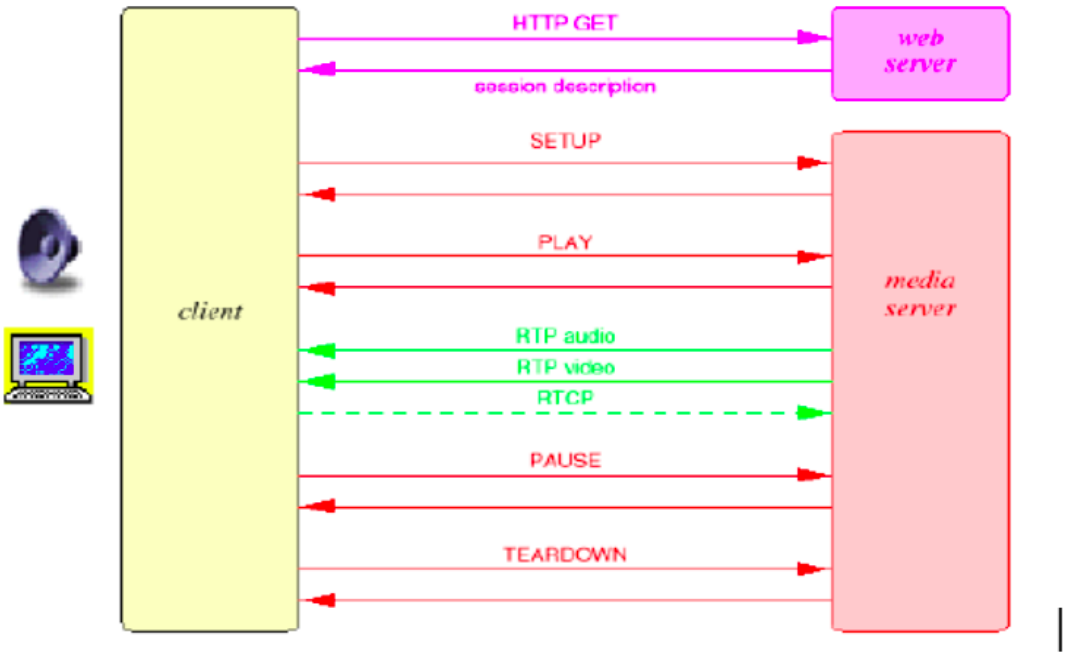


Figure 1: RTSP and RTP Protocol Messages

# 3 SYSTEM DESIGN

- Our project's system design can be described using the following
  - System Block Diagram
    - Principal Components and Architecture.

- Pseudo code
  - Server Class
  - Client Class
- FLowchart
  - Working of server and client classes

## 3.1 SYSTEM BLOCK DIAGRAM

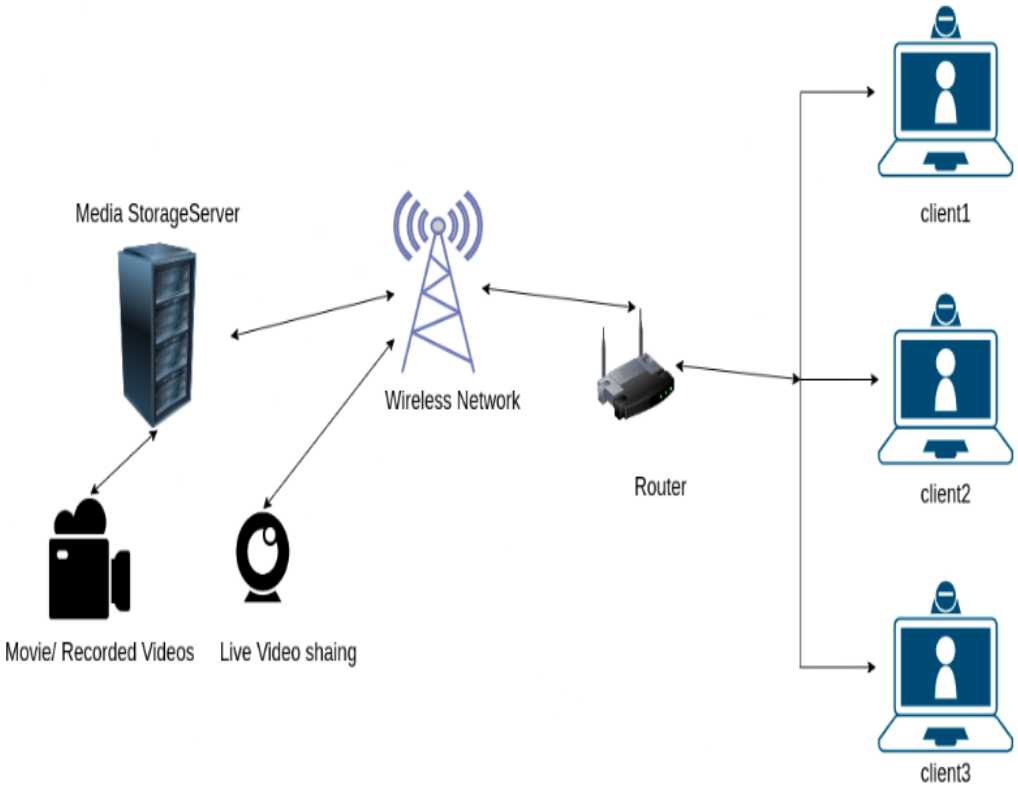


Figure 2: SYSTEM BLOCK DIAGRAM

## 3.2 SERVER CLASS - PSEUDO CODE

```
CLIENT CLASS() :
    Initializations :
        Initialize the GUI buttons
        Initialize the connection to server with given serveradr and portNo

    SET_UP_BUTTON :
        send SET_UP request with the video filename
        new thread :to wait for reply
                        process the reply
                        create a new rtp socket

    PAUSE_BUTTON :
        send PAUSE request

    TEARDOWN_BUTTON :
        send TEARDOWN request
        close the GUI window

    PLAY_BUTTON :
        create an listening thread for incoming video packets and play the video
            listen untill a PAUSE or TEARDOWN acknowledgement is sent
                if PAUSE : end the thread
                if TEARDOWN : close the rtp socket created for video streaming
        send PLAY request
```

Figure 3: server-pseudo-code

## 3.3 CLIENT CLASS - PSEUDO CODE

```
SERVER CLASS()
    Initializations :
        open to accept requests for clients
    Functioning :
        initialize a thread to recieve RTSP requests
            process each RTSP request
            if SET_UP request :
                check for the video file requested for
                and reply with an acknowledgement if present
                or negative acknowledgement if not present
            if PLAY request :
                create a new thread (sendRTP)
                and send the video packets in RTP format
            if PAUSE request :
                stop the thread started in PLAY request
            if TEARDOWN request :
                stop the thread started in PLAY request
                ans close the RTP socket created
```

Figure 4: client-pseudo-code

### 3.4 FLOWCHART

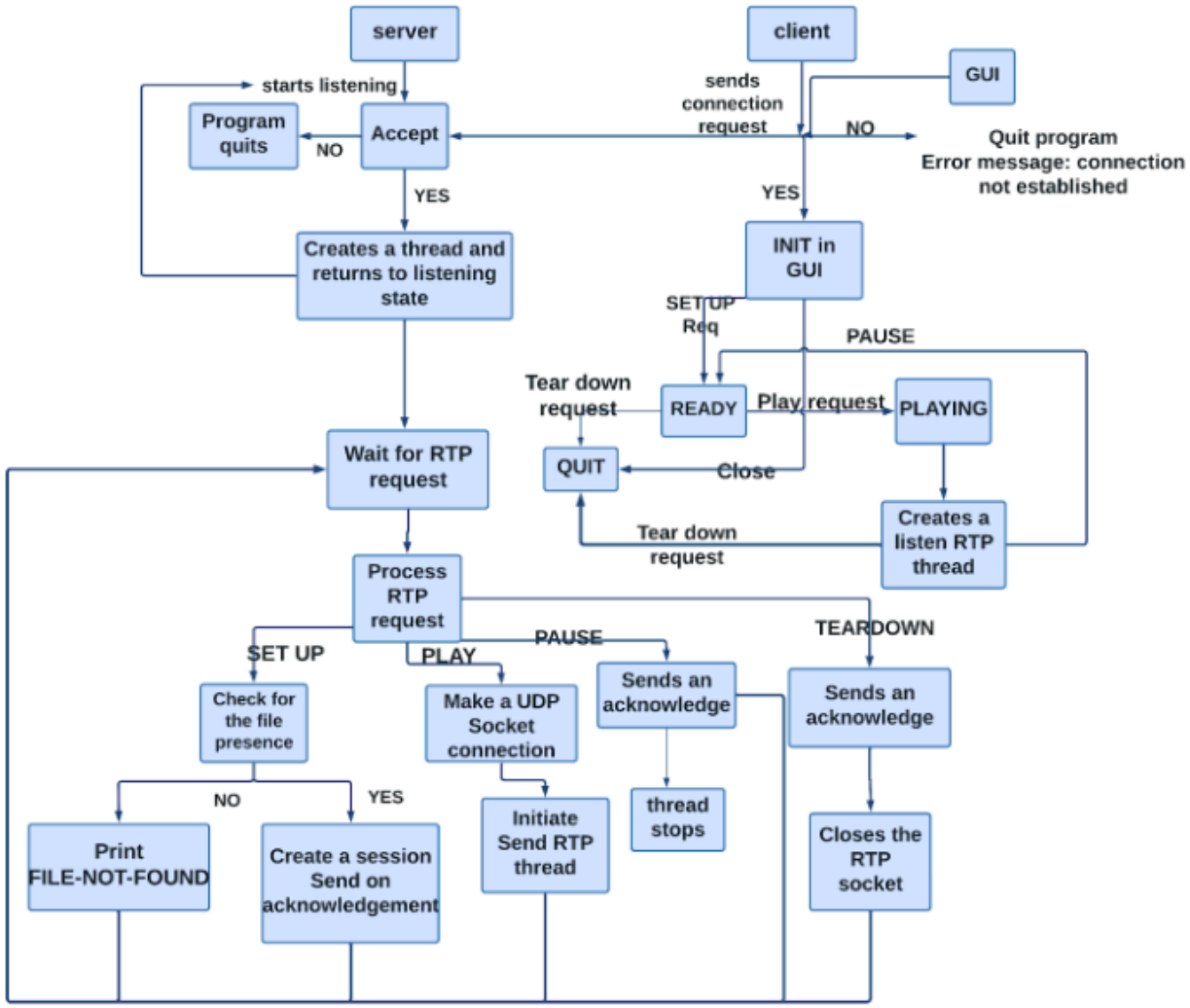


Figure 5: flowchart

# 4 Code Files

The language chosen for implementation of this project is python. The five main files are “server.py”, “client.py”, “rtppacket.py”, “videostream.py” and “audiostream.py”

**For Server and Client**

A RTSP socket using TCP protocol is created for pause and play functionalities as well as for acknowledgements. Video streaming is implemented using RTP Packet and UDP socket.
Audio streaming is implemented using TCP Socket

- SERVER
  - This file is used to start running the server. Once the server begins, it starts listening and multiple clients can join the same server.
  - A new "server" object is created for each new client joined.
  - Two threads are created for sending audio and video frames respectively
- CLIENT
  - This file, on running, establishes the connection with the server (providing valid arguments)
  - Graphical User Interface is implemented with help of "kivy" module and is added to provide a better interface.
  - Totally 5 screens are created and are directed according to the User's choice.
  - Final Screen will be either "video streaming" or "live streaming" containing SETUP,RESET, PLAY,PAUSE and TEARDOWN options for better streaming.
- CLIENT HELPER
  - For each new client, a new "Client" object is created.
  - Two threads are created for receiving audio and video frames respectively.
- RTPPACKET
  - This file is used to handle the RTP packets.
  - A header is created according to the RTP protocol. Finally, the RTP packet contains - RTP header and Payload.
- VIDEOSTREAM
  - Based on the User' choice, it will either read data from the video file or from the camera and data is sent frame by frame.
- AUDIOSTREAM
  - This file is used to read audio data from the file, which was provided with audio extracted from the given video file.

# 5 STARTING SERVER AND CLIENT

Using the command "python3 server.py [SERVER_PORT]", the server starts listening. The server is open and clients can connect to the server.

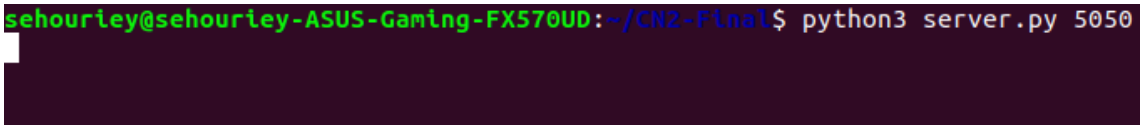


Figure 6: server start

Using the command "python3 client.py [CLIENT_IP_ADDR] [SERVER_PORT] [VIDEO_PORT] [AUDIO_PORT]", the client is ready to connect to the server.

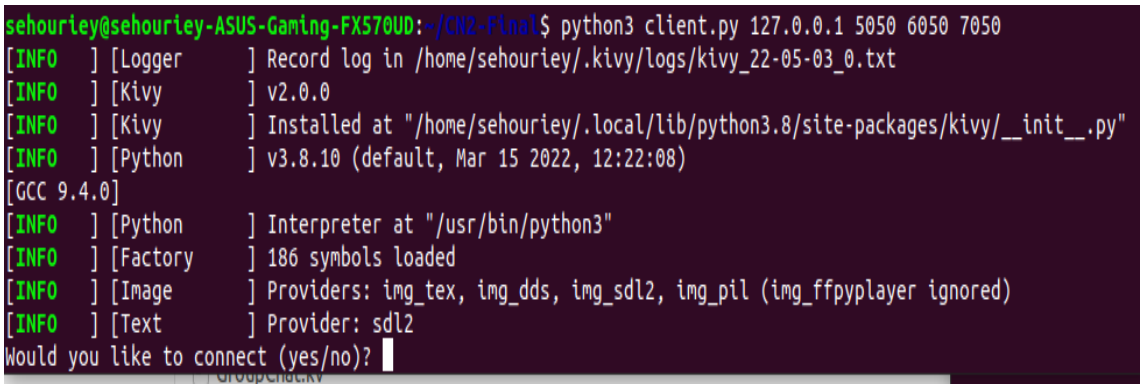


Figure 7: client-pseudo-code

If the chosen option is:

- **yes** : The client gets connected to the server.
- **no** : The client refused the connection and the program terminates.

Once the connection is established between the client and the server, an "Rtp Streaming Application" window gets created.

# 6 WORKING OF THE PROJECT(STREAMING SERVICE)

The client is prompted to enter his name before joining.

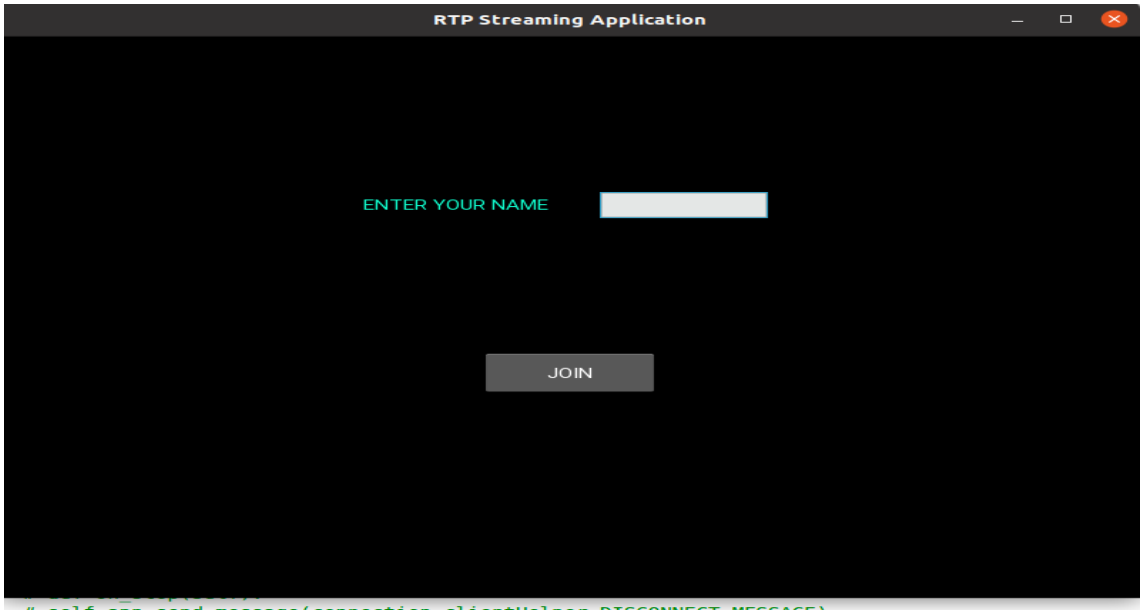


Figure 8: Prompt to enter the name

This is a mandatory field i.e, it cannot be empty. If the name is empty or an empty string, the user is warned that it is a Invalid Name. A pop up is created requesting "PLEASE ENTER A VALID NAME!". After closing the popup, the user is again prompted to enter a name until it is valid.

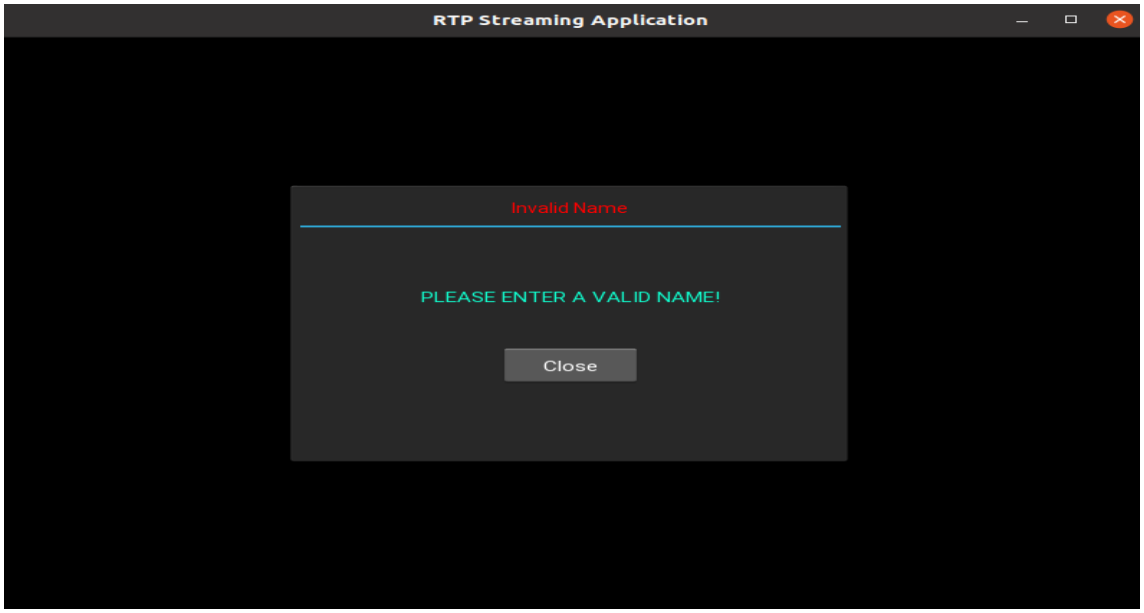


Figure 9: Prompt showing invalid image

On entering a valid username and clicking JOIN, the user is directed towards the streaming service .

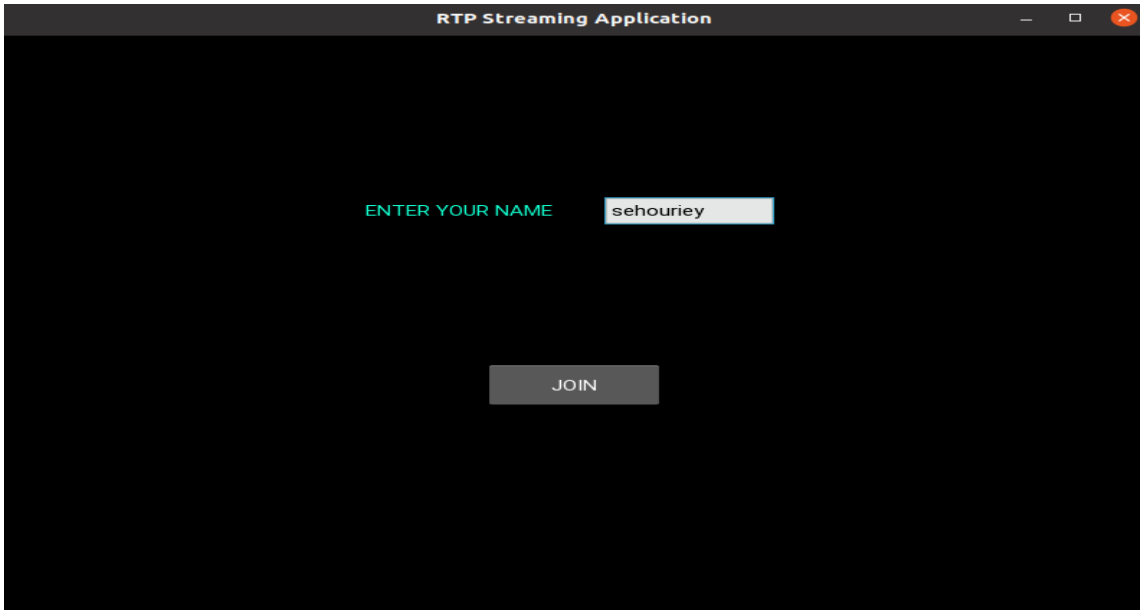


Figure 10: entering name

The client/user now has 2 choices

- VIDEO STREAMING
- LIVE STREAMING

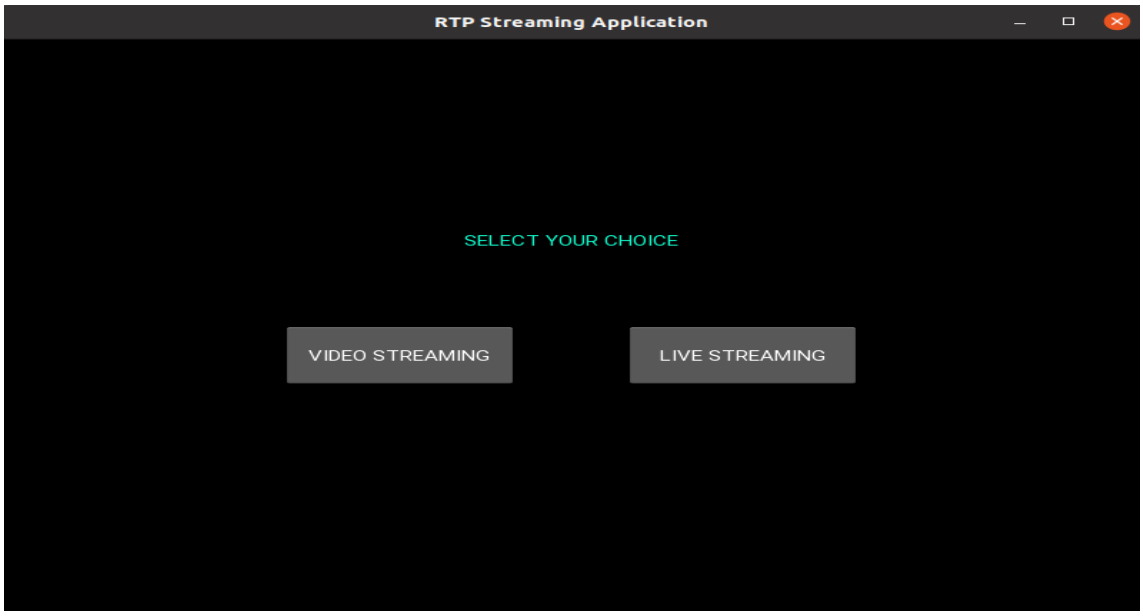


Figure 11: prompt to select the stream

If the user chooses the Video streaming option, the name of the video file to be streamed by the server is prompted to enter.

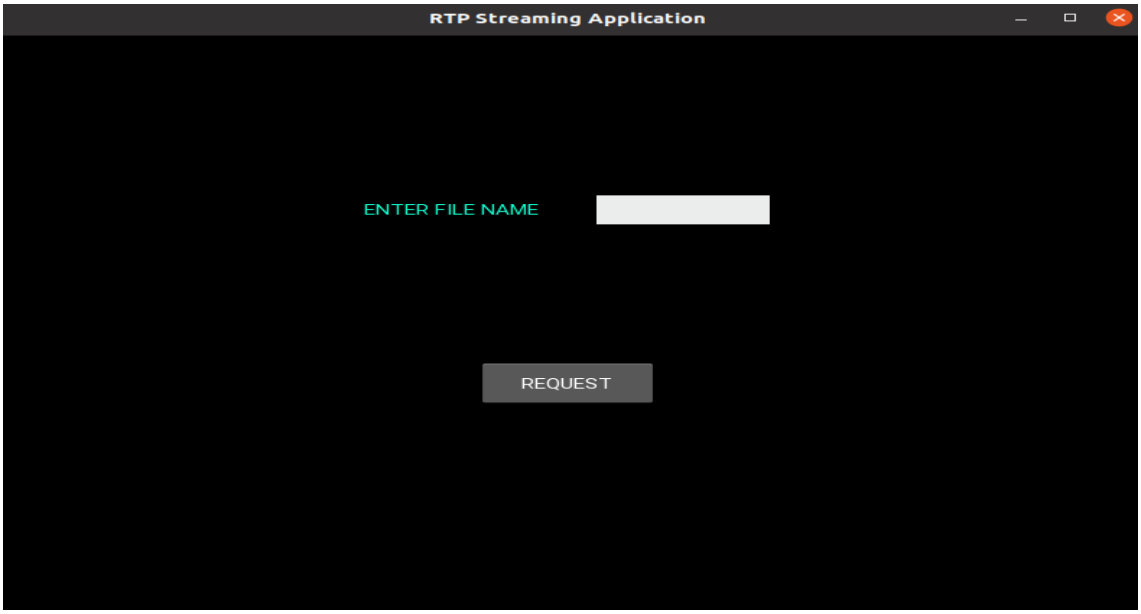


Figure 12: prompt to enter file name

The file requested by the client/user is checked in the server's directory. If the file is not found, it informs the client with a pop up saying **"REQUESTED FILE DOES NOT EXIST!"**.

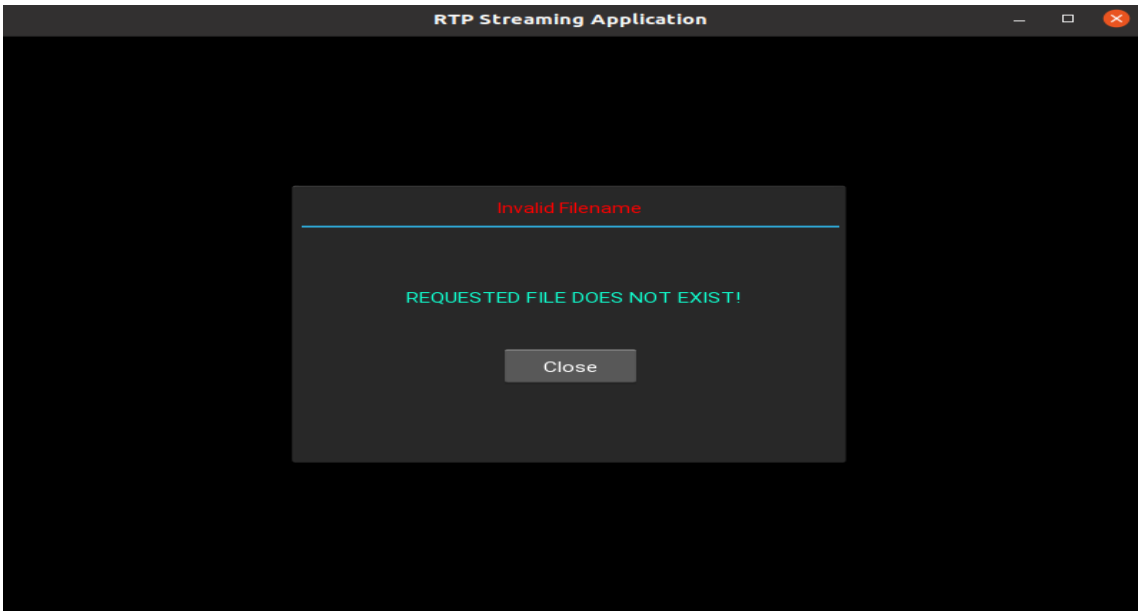


Figure 13: prompt showing file does not exist

If the file to be streamed by the server is found, the video streaming service is controlled by the client.

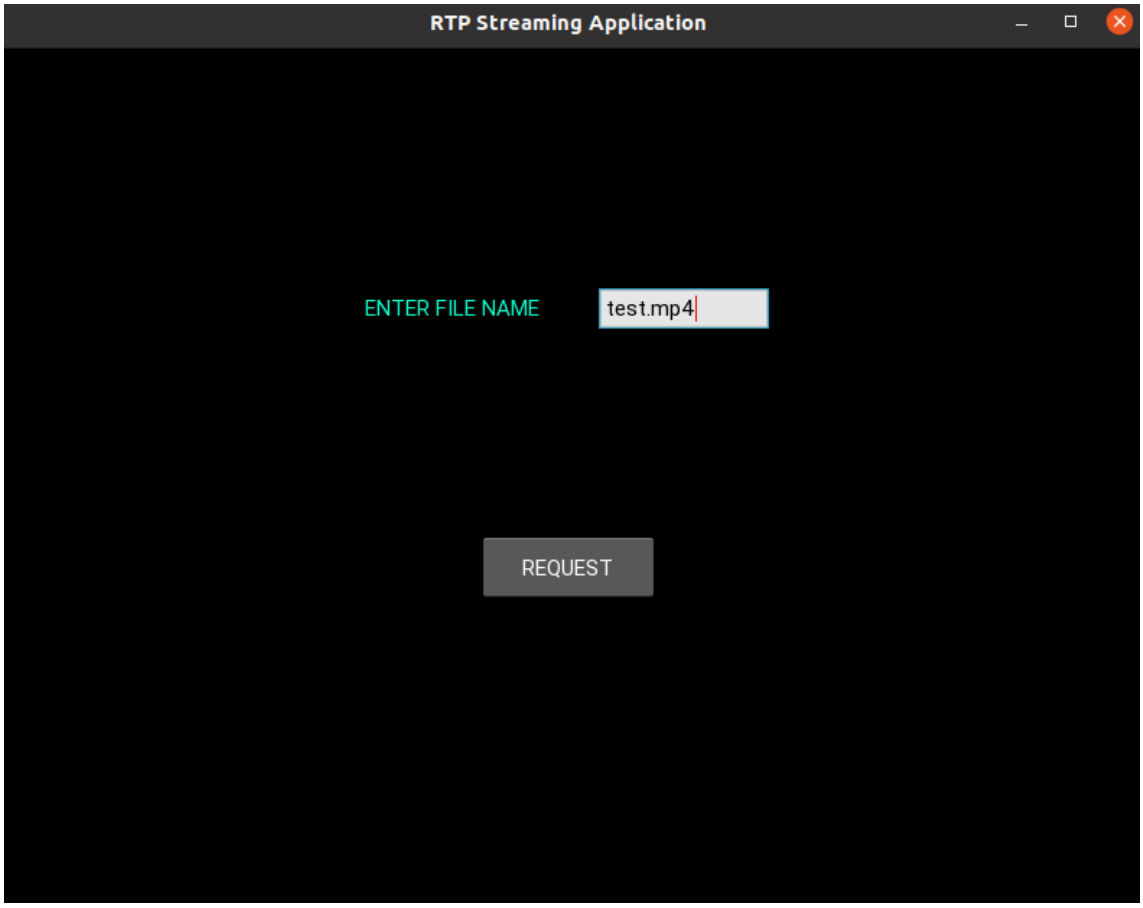


Figure 14: prompt entering file name

The image shown below is presented showing the start of “VIDEO STREAMING”.

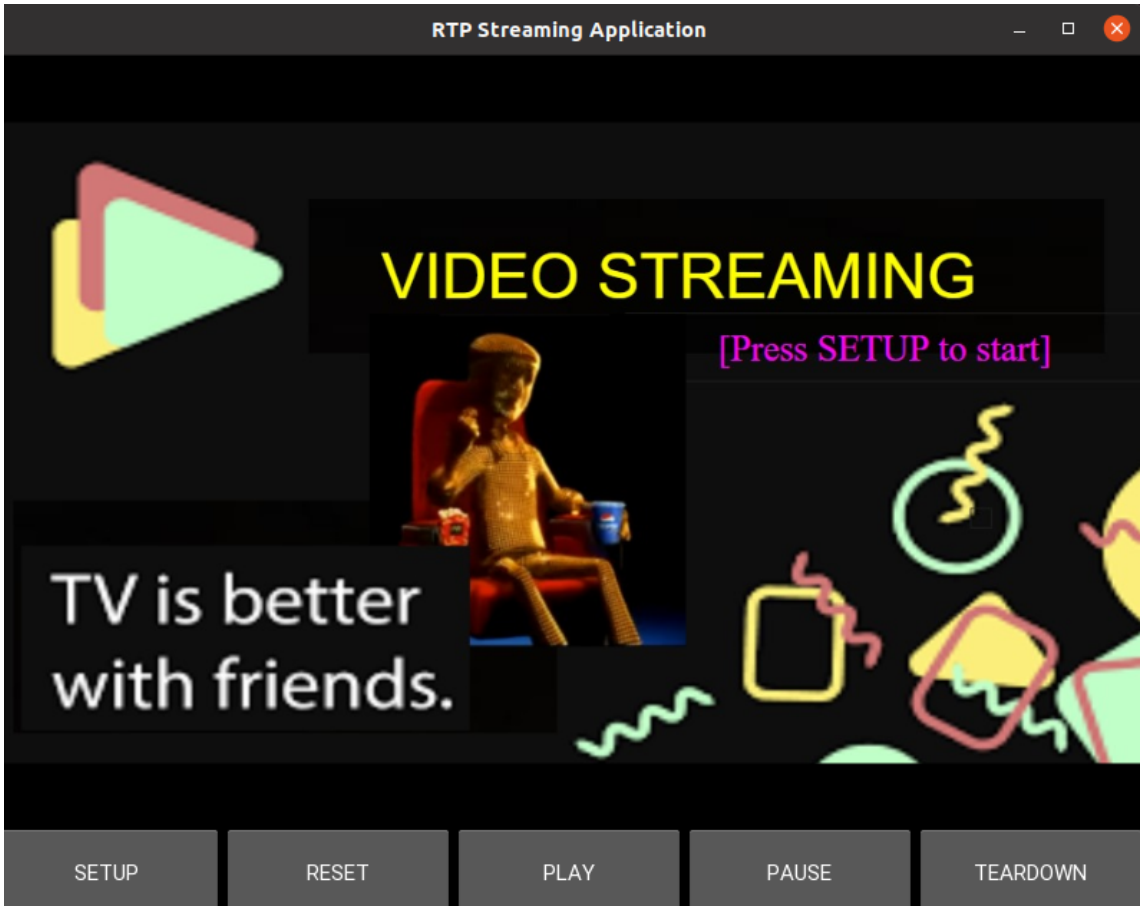


Figure 15: Window showing start of video streaming

The client now has 5 choices:

- **"SETUP" :** Sets up the video streaming session
- **"PLAY" :** Starts streaming the video file chosen()
- **"PAUSE" :** Stops the video file temporarily, can be continued using the PLAY option
- **"RESET" :** Restarts the session. On choosing PLAY, it gets streamed from the start of the video.
- **"TEARDOWN" :** Stops streaming the video by closing the session. This deletes the created window and the client gets disconnected from the server.

(Below image is a screen capture while playing the video)

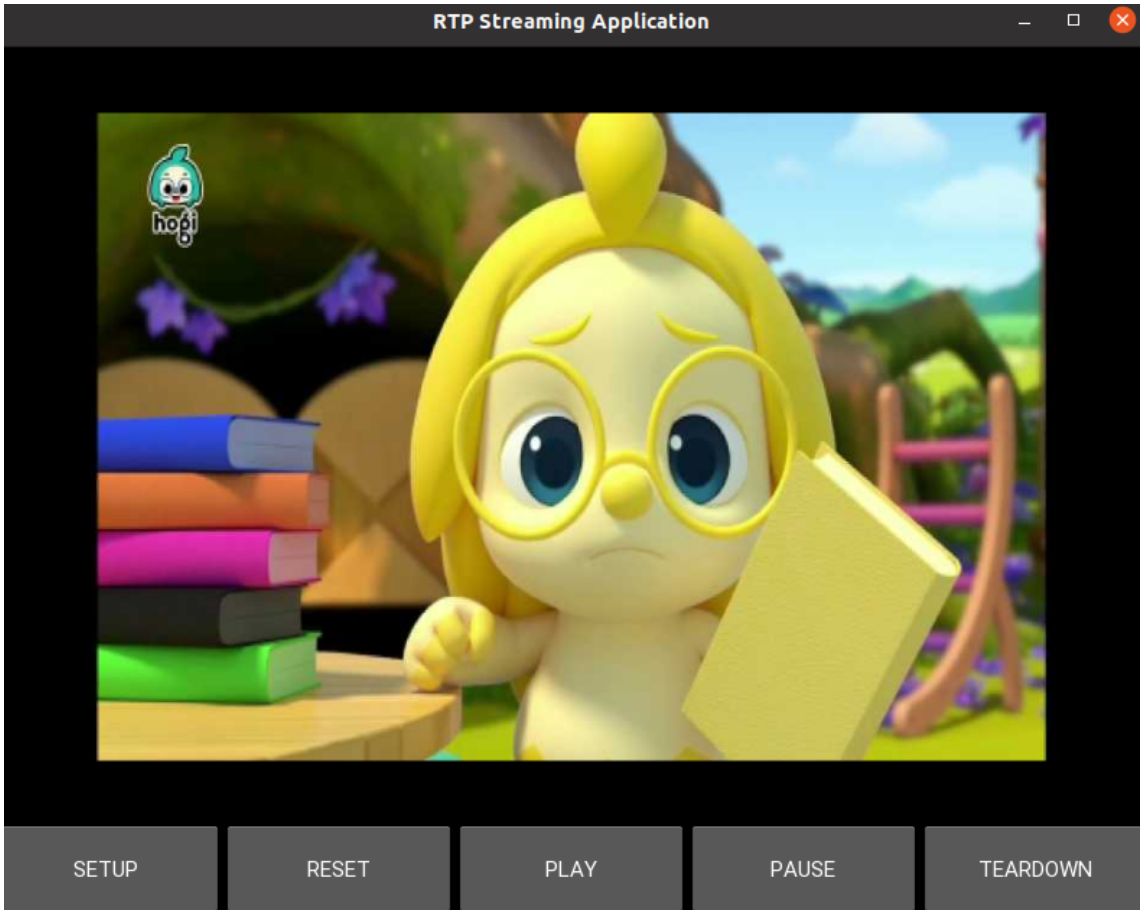


Figure 16: Screen shot during a video playing

The client has to reconnect with the server, to get back to the streaming service.
If the client chooses the option “LIVE STREAMING”, after joining, the client is now ready to join the live streaming shared by the server.
The image shown below shows the start of the Live Streaming.

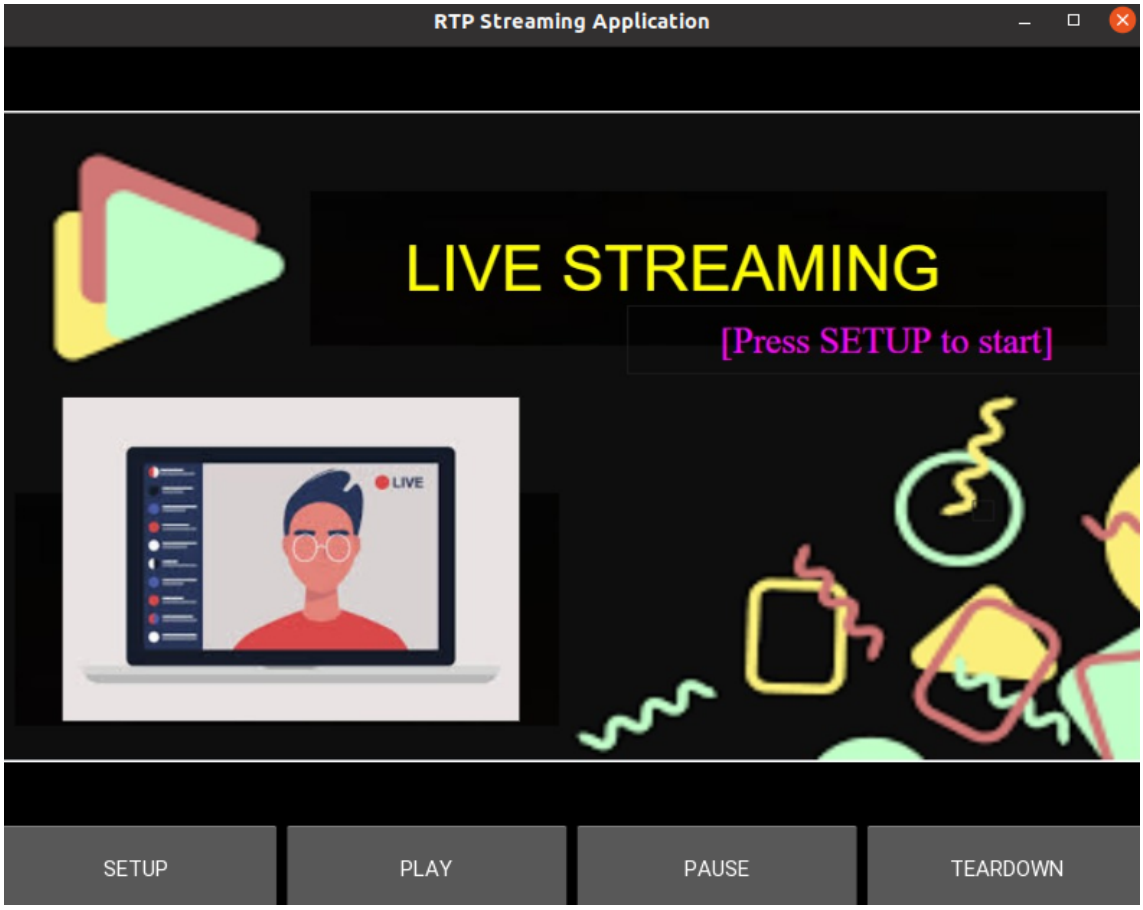


Figure 17: Window showing start of Live streaming

The client now has 4 choices

- "SETUP": Sets up the live streaming session
- "PLAY": Starts streaming the live video from the server
- "PAUSE": Stops the live video temporarily, can be continued using the PLAY option
- "TEARDOWN": Stops streaming the live video by closing the session. This deletes the created window and the client gets disconnected from the server.

(Below image is a screen capture while streaming the live video)


Figure 18: Image screenshot during live streaming

This streaming service can be used by multiple clients/users by connecting to the server. Multiple clients can request for streaming the same or different videos.

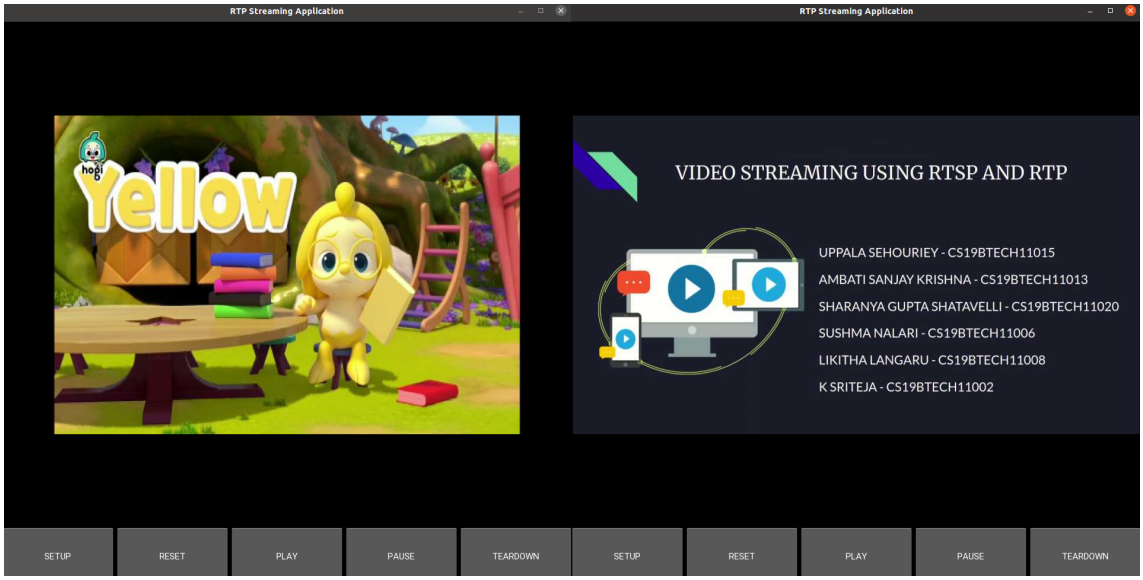


Figure 19: Video streaming in two different clients

Live streaming and Video streaming are also simultaneously enabled.

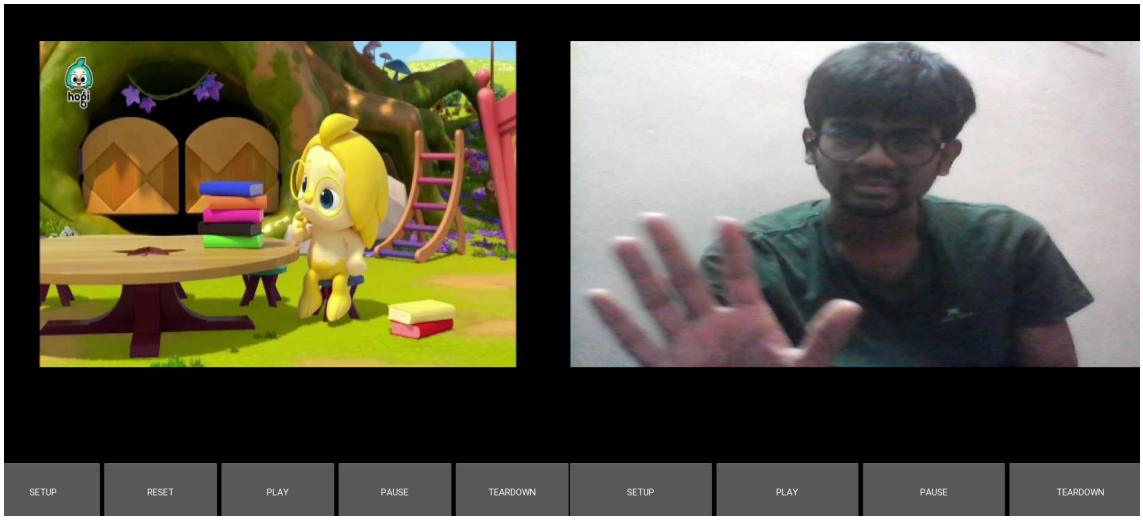


Figure 20: simultaneous live and video streaming in two different clients

This was the main goal of this project. The client can now enjoy the video streaming service.

LaTeX generated document

**********END**********