

VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING



Computer Network Lab (CO3094)

RTSP and RTP

ASSIGNMENT 1 REPORT

Lecturer: Nguyễn Mạnh Thìn

Class : CC03

Group : 6

Students: Lê Đức Cầm - 1952588
Nguyễn Tiến Dương - 1952639
Ngô Minh Đại - 1913008
Đào Văn Hiếu - 1852362

HO CHI MINH CITY, OCTOBER 2021

Contents

1	Problem Requirement Analysis	2
1.1	Functional requirement	2
1.2	Non-functional requirement	2
1.3	Source file	3
1.4	Task	3
2	Description of different functions	4
3	List of components	7
3.1	SETUP	7
3.2	PLAY	7
3.3	PAUSE	8
3.4	TEARDOWN	8
3.5	DESCRIBE	8
4	Model and data flow	9
4.1	Server	9
4.2	Server worker	9
4.3	RTP packet	11
4.4	VideoStream	11
5	Class diagram	12
6	Implementation	12
7	Achieved results	13
7.1	Programming aspect	13
7.2	Running application aspect	14
7.2.1	The GUI	14
7.2.2	Communication between server and client	16
8	User manual	18
9	Reference	20

1 Problem Requirement Analysis

1.1 Functional requirement

★ System side

- System can connect and communicate with client via Real-Time Streaming Protocol (RTSP) and Real-time Transfer Protocol (RTP).

- System can stream video and provide some function like set up, play, pause, tear-down Streaming Protocol (RTSP) and Real-time Transfer Protocol (RTP).

★ Client side

- User can connect to the server by using terminal.

- User can set up, play, stop or shutdown the application

- User can see some basis information like how long the video is or how much time left

- User can go to every part of the video at anytime with scroll bar

1.2 Non-functional requirement

- Video must be in .Mjpeg format

- Server response time $\leq 0.5s$

- Easy to use

- Get the service with less than 5 clicks of mouse

- Basic GUI

- The application should be able to run on PC



1.3 Source file

Building this video application will require following source files :

1. Client.py
2. ClientLauncher.py
3. RtpPacket.py
4. Server.py
5. ServerWorker.py
6. VideoStream.py

1.4 Task

We will implement :

- The RTSP protocol in the client (named as Client.py).
- The RTP packetization in the server (named as RtpPacket.py).
- The customized interactive player for displaying the transmitted video.
- Calculate the statistics about the session (extend section).
- Solution to make our program same as those standard media player (have only 3 basic button - get rid of set up button)
- Implement the method DESCRIBE (extend section).
- Add mechanism to tell client the total length and remaining time of the movie (extend section).
- Provide a way for user to move forward or backward

2 Description of different functions

★ Description of functions in Server, ServerWorker and VideoStream file.

· Function in Server.py: Create a RTSP socket and open the connection to host with the provided port. It's also forward the data sent from client to the ServerWorker to be processed with the help of clientInfo variable.

· Function in ServerWorker.py: When ever a new client come, a new thread is started to process requests sent from Client comprise of determine the request type, required file name,... ServerWorker also response to Client by sending the needed video frame by frame and some status

· Function in VideoStream.py: open the required video, read a frame from video, return a frame number,total frame of the whole movie and also the fps of that movie.

Class Name	Function	Description
ServerWorker	__init__(self,clientInfo)	Constructor
	run(self)	Run the server
	processRtspRequest(self,data)	Process the Rtsp request sent from client
	sendRtp(self)	Send RTP packets over UDP
	makeRtp(self,payload,frameNbr)	RPT-packetize the video data
	replyRtsp(self,code,seq)	Send RTSP reply to the client
	recvRtspRequest(self)	Receive RTSP request from client
Server	main(self)	Main function to start the entire program
VideoStream	__init__(self, filename)	Constructor
	nextFrame(self)	Get next frame.
	frameNbr(self)	Get frame number.
	getTotalFrame(self)	Get total frame
	getFps(self)	Get fps of the movie

Hình 1: Function description

★ Description of functions in RtpPacket file.

· Function in RtpPacket.py: provide mechanism to encode the variables like version, padding, extension, cc, seqnum, marker, pt, ssrc into a RTP packet and way to decode it. It also help to get the payload of a packet and packet itself.

Class Name	Function	Description
RtpPacket	__init__(self)	Constructor
	encode(self, version, padding, extension, cc, seqnum, marker, pt, ssrc, payload)	Encode the RTP packet with header fields and payload
	decode(self, byteStream)	Decode the RTP packet.
	version(self)	Return RTP version.
	seqNum(self)	Return sequence (frame) number.
	timestamp(self)	Return timestamp.
	payloadType(self)	Return payload type.
	getPayload(self)	Return payload.
	getPacket(self)	Return RTP packet.

Hình 2: RtpPacket's function description

★ Description of functions in ClientLauncher file.

· Function in ClientLauncher.py: Extract information like serverAddr, serverPort, rtpPort, fileName, after that create a new client with those input data .

Class Name	Function	Description
ClientLauncher	A If statement	Initial setup connection

Hình 3: If statement in ClientLaucher

★ Description of functions in Client file.

· Function in Client.py: Set up widget needed to run the app from client-side like the Graphic User Interface (include buttons, label,...). Then connect to the RTSP socket created by Sever. It also have functions help client to send request and continuously receive then parse reply from server (use while loop). And of course, some functions were built to enable client to interact with those provided buttons such as set up, play, pause, describe, tear down.

Class Name	Function	Description
Client	__init__(self, master, serveraddr, serverport, rtp port, filename)	Constructor
	createWidgets(self)	Build Graphical User Interface(GUI)
	setupMovie(self)	Setup button handler
	exitClient(self)	Teardown button handler
	pauseMovie(self)	Pause button handler
	playMovie(self)	Play button handler.
	describeVideo(self)	Describe button handler
	listenRtp(self)	Listen for RTP packets.
	writeFrame(self, data)	Write the received frame to a temp image file. Return the image file.
	updateMovie(self, imageFile)	Update the image file as video frame in the GUI
	connectToServer(self)	Connect to the Server. Start a new RTSP/TCP session.
	sendRtspRequest(self, requestCode)	Send an RTSP request to the server.
	recvRtspReply(self):	Receive RTSP reply from the server.
	parseRtspReply(self, data):	Parse the RTSP reply from the server.
	openRtpPort(self):	Open RTP socket binded to a specified port.
	handler(self)	Handler on explicitly closing the GUI window.

Hình 4: Client's function description

3 List of components

There are 4 major components: SETUP, PLAY, PAUSE, TEARDOWN

3.1 SETUP

When send SETUP request to the server. Client will insert the Transport header, that he had specified the port for RPT data socket he created. The client will receive the respond from the server and the ID of the RTSP port.

The RTSP SETUP packet include

- Name of the video.
- The number of RTSP packet.
- Protocol.
- RTP port.

when the server receives the SETUP request, it will:

- Parse the Session header to client.

If the error happen, the server will send ERROR to client .Otherwise, It will send OK-200 and set to READY state.It will open the video file which has been specified in SETUP packet.

The client will repeat to receives RTSP from server. If the RTSP packet respond for SETUP request, client will set state to READY and open a RTP port to receive video.

3.2 PLAY

When the client send PLAY request to the server: The server will create Socket to send RTP through UDP begin with sending packet video. The file VideoStream.py will separate the video into different frame and send it to packet data RTP.

Each packet will be attach with a header include:

- RTP version.
- Padding.
- Extension.
- Contribute source.
- Marker.
- Type.
- Sequence number.
- Timestamp.
- SSRC.

They will be insert into RTP packet through bitwise operation. The RTP packet will include video frame and a header which will be send to RTP port from client. When reciving the client will resolve the RTP packet to get header and video frame, and reorganize to show for client



3.3 PAUSE

When a client sends PAUSE request to server, it will stop the server sending video frame. And change the server STATE to READ to wait for the client's request.

3.4 TEARDOWN

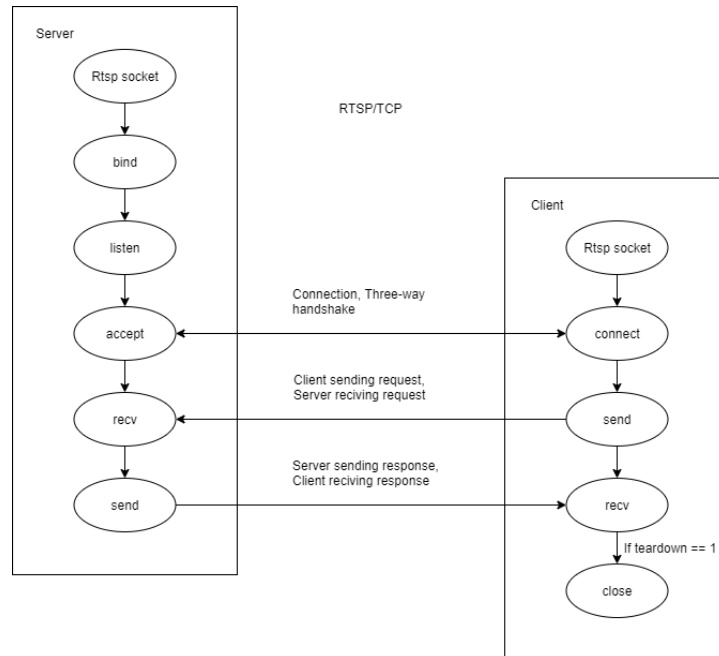
If a TEARDOWN request is sent from the client to the server, it will prevent the server from sending the video frame and close terminal of client and set it STATE to INIT.

3.5 DESCRIBE

When a client send DESCRIBE request to the server, the server will send back to client the session, type of the video that is playing and the type of character encoding.

4 Model and data flow

4.1 Server



Hình 5: Communication between Client and Server by RTSP/TCP protocol

Figure 5 illustrates the process of communicating and exchanging data between Client and Server via RTSP/TCP protocol is established in Video Streaming system.

- Create RTSP Socket and open connection to public host and provided port address, Client and Server will communicate with each other via RTSP sockets on each component - receive and respond to requests from the Client.
- The ClientInfo variable is created to contain the data that the client sends to the server and transfers via ServerWorker to process the request.

4.2 Server worker

Handle Request(Setup, Play, Pause, Teardown) sent from Client via interface RTSP mode.

For each client, a Request thread is created. In each stream, one rtp socket is initialized to send data to the Client's rtp socket, each thread has its own method to receive, process requests and respond to requests from the Client.

When receiving a request from the Client, the Server extracts the data of the request, determines the request type and required information. When receiving a setup request from

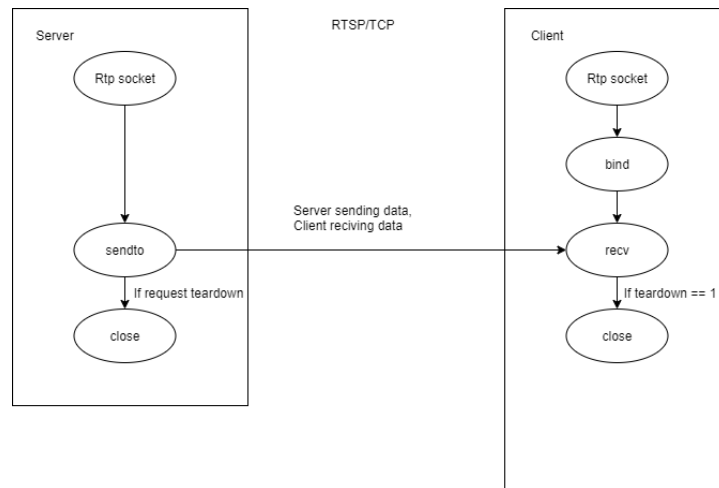
the Client, the video file name to be played is provided through the request from the Client and save to ClientInfo the VideoStream object of the video file to serve the process of downloading and sending video data. Also information about Client port is also determined and saved. A sessionId is generated randomly. However, this number helps the Client receive the correct response that the Server sent. Server send a response to receive a successful request to the client and change the server state to READY.

When receiving a play request from the Client, check the current status of the Server, if current state is READY, change Server state to PLAYING, create an RTP socket to send image data to the client. The server will generate a stream to traverse the data from the requested video, using the nextFrame . method of Videostream to read the frames corresponding to each image in the video, the code encodes, encapsulates the packet, encapsulates the packet into UDP segments, and sends it to Client follows RTP/UDP protocol, independent stream and while loop combined sending UDP protocol allows data to be vertically and continuously sent. Send feedback respond to the request to the Client.

When it receives a pause request, the server checks its current state, if the current state is PLAYING, use threading.Event().set() to set the flag to true. At the stream reading the data, if the flag is true stop reading the data. Transition state server state to READY and send successful response to Client.

When a teardown request is received, stop the video like processing pause request, sending response return to the Client and close the RTP socket.

Figure 6 illustrates the process of Server sending segments containing data about images through the RTP/UDP protocol.



Hình 6: Communication between Client and Server by RTP/TCP protocol

4.3 RTP packet

encode function: Sets headers for objects, assigns headers and data frames, and header and payload variables of the object.

decode function: determines the header and payload components in a packet, and assigns them for the respective variables.

The version, seqNum, timestamp, and payloadType functions output the given parameters save in header.

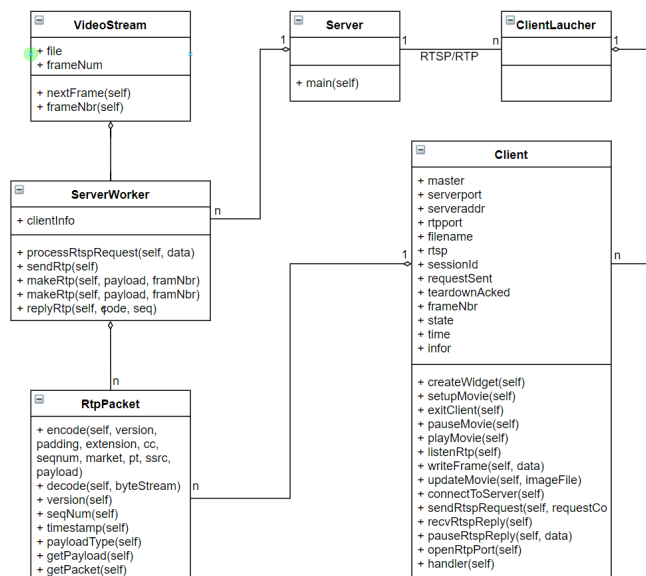
getPayload function: returns the payload in the packet.

getPacket function: Packs the image frame into a packet.

4.4 VideoStream

Open the video file to read, read each frame in the video data, each frame corresponds to an image appears in the video. The nextFrame function is a method with a function read a data frame in a video file, when combined with a full frame while loop in the video file will be read.

5 Class diagram



Hình 7: Class diagram

6 Implementation

★ We submit report with zip file contain code on BKEL

★ Or you can see the work we have done via the link below:

<https://github.com/Cam-Duc-Le/computer-network>

7 Achieved results

7.1 Programming aspect

★ Required section: 100%

- Finish the RTSP protocol in the client (named as Client.py).
- Completely fill the header bytearray with RTP header fields in encode() method in the RtpPacket class (in RtpPacket.py).
- Successfully construct the customized interactive player for displaying the transmitted video (like buttons, and label).

★ Extend section: 75-80%

- Modify and add some functions in VideoStream.py, ServerWorker.py Client.py for further extend requirement.
- Question 1: System can calculate and return the RTP packet loss rate (%), video data rate (we chose to present in bytes per second)
- Question 2: Program automatically SETUP when you **first time** connect to the server (You don't need to press SETUP and jump right into movie), in case you press TEARDOWN to watch another movie (it means that it will disrupt the connection to original movie), Later you can click on SETUP to reconnect to the server which contain the original movie.
- Question 3: System have a DESCRIBE method and button to pass information about the media stream (the payload type - PT) and the encoding used
- Question 4: System can display video total time (in seconds) and show the remaining time left (also in second) when client click PAUSE or TEARDOWN button. We also provide SCALE BAR to fast forward and move backward

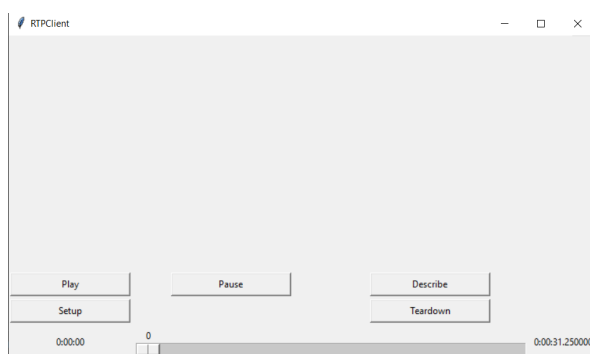
★ What task that we have not finished yet:

- We have not done question 5 in extend section
- Some bugs still happen when we perform some tricky operations
- Responsiveness is not really ok that means you may have to do operations on the buttons slowly so the follow won't collapse.

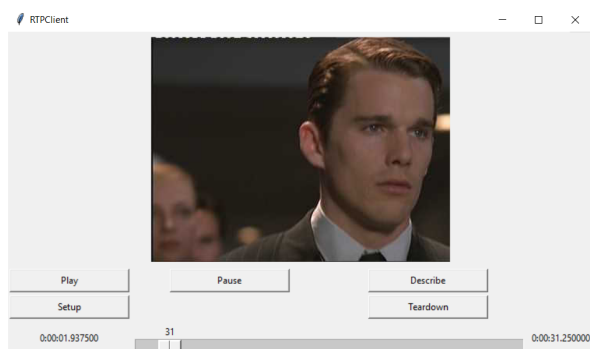
7.2 Running application aspect

- Server and Client can communicate with each other.
- Server can process every request from user .
- Client can receive information sent by Server.
- Client can interact with program via the graphical user interface
- Client have some options(as button and scroll bar) like figures below.
- Client can press teardown then setup to watch movie again

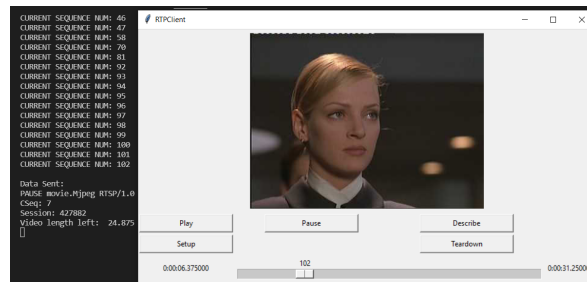
7.2.1 The GUI



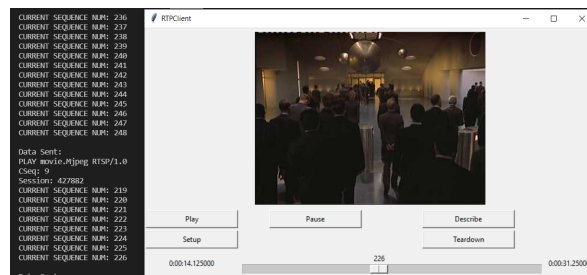
Hình 8: Client interface with automatically setup process



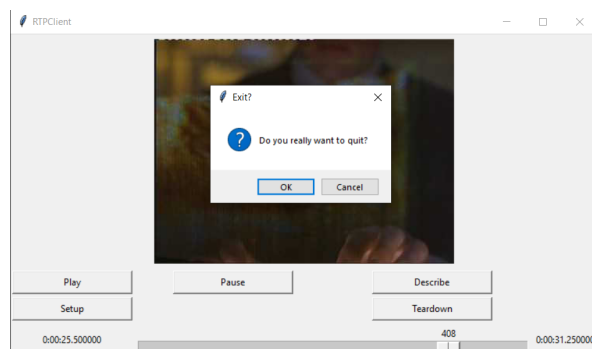
Hình 9: sample GUI when streaming video or press PAUSE



Hình 10: User can fast forward with provided scroll bar without stopping movie



Hình 11: User can go backward and they need to click PLAY to show video



Hình 12: Exit button clicked or Client can simply press TEARDOWN

7.2.2 Communication between server and client

```
Data Sent:
SETUP movie.Mjpeg RTSP/1.0
CSeq: 1
Transport: RTP/UDP; client_port= 5678
```

Hình 13: Automatically
Client SETUP request
when initialize a new
client

```
Data received:
SETUP movie.Mjpeg RTSP/1.0
CSeq: 1
Transport: RTP/UDP; client_port= 5678
processing SETUP
```

Hình 14: Server SETUP
receivment

```
Data Sent:
PLAY movie.Mjpeg RTSP/1.0
CSeq: 2
Session: 781964
```

Hình 15: Client PLAY
request

```
Data received:
PLAY movie.Mjpeg RTSP/1.0
CSeq: 2
Session: 781964
processing PLAY
```

Hình 16: Server PLAY
receivment

```
Data Sent:
PAUSE movie.Mjpeg RTSP/1.0
CSeq: 3
Session: 781964
Video length left: 18.96 second
```

Hình 17: Client PAUSE
request

```
Data received:
PAUSE movie.Mjpeg RTSP/1.0
CSeq: 3
Session: 781964
processing PAUSE
```

Hình 18: Server PAUSE
receivment

```
Data Sent:
DESCRIBE movie.Mjpeg RTSP/1.0
CSeq: 4
Session: 781964
type: Mjpeg(26)
Encode: utf-8
```

Hình 19: Client DE-
SCRIBE request

```
Data received:
DESCRIBE movie.Mjpeg RTSP/1.0
CSeq: 4
Session: 781964
processing DESCRIBE
```

Hình 20: Server DE-
SCRIBE receivment

```
Data Sent:
TEARDOWN movie.Mjpeg RTSP/1.0
CSeq: 7
Session: 781964
```

Hình 21: Client TEAR-
DOWN request

```
Data received:
TEARDOWN movie.Mjpeg RTSP/1.0
CSeq: 7
Session: 781964
processing TEARDOWN
```

Hình 22: Server TEAR-
DOWN receivment

- Information sent when client teardown somewhere in the middle of the video

```
RTP packet loss rate: 0 %  
Video data rate 95223 bytes per second  
Total video length: 31.25 second  
Video length left: 5.0625 second
```

Hình 23: Suddenly teardown

- Information sent when client teardown at the end of the video without going backward (you can see that video rate * total length will be equal to 4.07 mb)

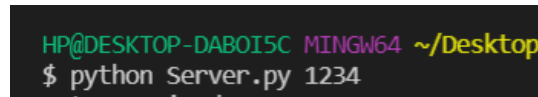
```
RTP packet loss rate: 0 %  
Video data rate 136556 bytes per second  
Total video length: 31.25 second  
Video length left: 0.0 second
```

Hình 24: Teardown at the end when watching movie continuously 1 time

8 User manual

Step 1: the user needs to start the Server: run Terminal in the directory containing the Server.py file. With Command: `python Server.py <port_server>`

+ `<port_server>`: select any number greater than 1024 because port numbers from 0 to 1023 are reserved for common TCP/IP applications and are called well-known ports. So in this case, our group choose: 1234.



```
HP@DESKTOP-DABOI5C MINGW64 ~/Desktop
$ python Server.py 1234
```

- Step 2: We open a new terminal and run it to start a Client that connects to the Server we just ran.

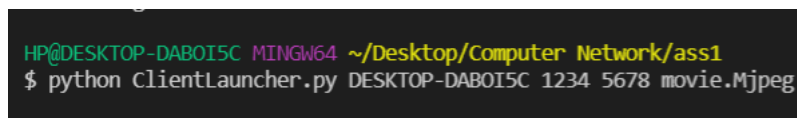
Command: `python ClientLauncher.py <name_server> <port_server> <port_RTP> <name_video>`

+ `<name_server>`: Server name of the computer in use.

+ `<port_server>`: port_server initialized in step 1 (1234).

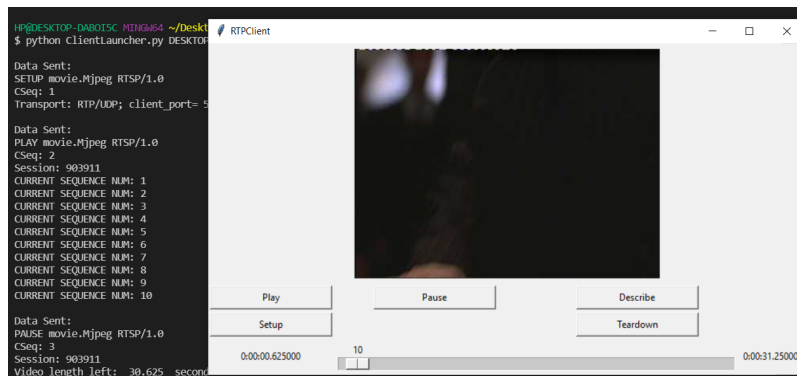
+ `<port_RTP>`: choose any number, eg: 5678.

+ `<name_video>`: the name of the video file.



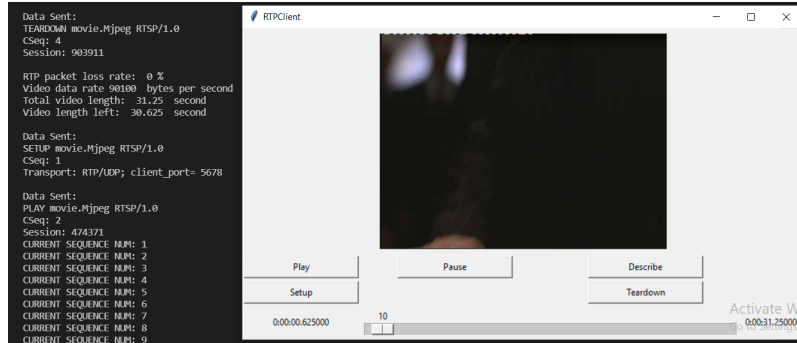
```
HP@DESKTOP-DABOI5C MINGW64 ~/Desktop/Computer Network/ass1
$ python ClientLauncher.py DESKTOP-DABOI5C 1234 5678 movie.Mjpeg
```

- Step 3: We can press the PLAY buttons right away to play the movie because the Set Up process is automatically done. Client can also click PAUSE button to stop the movie, scroll the HORIZONTAL BAR to fast forward or move backward, they can just click DESCRIBE button to see the encoding and payload type or TEARDOWN button to stop and end.





- Step 4: Client also can TEARDOWN to watch to watch others movie provided by server or press SET UP to connect again to the original movie and watch it easily .





9 Reference

1. Text book: Computer Networking _ A Top-Down Approach
2. Repositories on <https://github.com>
3. <https://stackoverflow.com/questions/4303439/what-is-the-difference-between-rtp-or-rtsp-in-a-st>
4. https://en.wikipedia.org/wiki/Real_Time_Streaming_Protocol
5. <https://www.tutorialspoint.com/python/>
6. <https://python.hotexamples.com/>