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

Computer Network (CO3094)

Assignment 1

Real-Time Streaming Protocol (RTSP) and Real-time Transfer Protocol (RTP)

Advisor : Nguyen Le Duy Lai

Students : Le Tu Ngoc Minh - 1952844
Dao Tien Tuan -
Nguyen Thanh Loc -

HO CHI MINH CITY, OCTOBER 2021

Contents

1 Software Requirements Analysis

1.1 Functional Requirements

1.1.0.a System Requirements

- The system can work (stream video).
- The system can communicate with users via RTSP/RTP protocol.

1.1.0.b Client Requirements

- It is possible to connect to the server via the terminal.
- User can play video from server, stop and end.
- User can view basic video parameters such as video duration.

1.2 Non-functional Requirements

- Videos must be in .Mjpeg format
- Response time from server $\leq$ 0.5s

2 Description of different functions for your application

Class Name	Function	Parameter	Description
Client	__init__(self, master, serveraddr, serverport, rtpport, filename)	self, master, serveraddr, serverport, rtpport, filename	*Constructor*
	createWidgets(self)	self	*Create GUI*
	setupMovie(self)	self	*Setup button handler*
	exitClient(self)	self	*Teardown button handler*
	pauseMovie(self)	self	*Pause button handler*
	playMovie(self)	self	*Play button handler*
	listenRtp(self)	self	Listen for RTP packets
	writeFrame(self, data)	self, data	*Write the received frame to a temp image file. Return the image file.*
	updateMovie(self, imageFile)	self, imageFile	*Update the image file as video frame in the GUI*
	connectToServer(self)	self	*Connect to the Server. Start a new RTSP/TCP session.*
	sendRtspRequest(self, requestCode)	self, requestCode	*Send RTSP request to the server.*
	recvRtspReply(self)	self	*Receive RTSP reply from the server.*
	parseRtspReply(self, data)	self, data	*Parse the RTSP reply from the server.*
	openRtpPort(self)	self	*Open RTP socket binded to a specified port.*
	handler(self)	self	*Handler on explicitly closing the GUI window.*
ClientLauncher	main()		*testcase*
ServerWorker	__init__(self, clientInfo)	self, clientInfo	*Constructor*
	run(self)	self	*Run the server*
	recvRtspRequest(self)	self	*Receive RTSP request from the client.*
	processRtspRequest(self, data)	self, data	*Process RTSP request sent from the client.*
	sendRtp(self)	self	*Send RTP packets over UDP.*
	makeRtp(self, payload, frameNbr)	self, payload, frameNbr	*RTP-packetize the video data*
	replyRtsp(self, code, seq)	self, code, seq	*Send RTSP reply to the client.*
Server	main(self)	self	*Main function to run the whole program*
RtpPacket	__init__(self)	self	*Constructor*
	encode(self, version, padding, extension, cc, seqnum, marker, pt, ssrc, payload)	self, version, padding, extension, cc, seqnum, marker, pt, ssrc, payload	*Encode the RTP packet with header fields and payload*
	decode(self, byteStream)	self, byteStream	*Decode the RTP packet.*
	version(self)	self	*Return RTP version.*
	seqNum(self)	self	*Return sequence (frame) number.*
	timestamp(self)	self	*Return timestamp*
	payloadType(self)	self	*Return payload type.*
	getPayload(self)	self	*Return payload.*
	getPacket(self)	self	*Return RTP packet.*
VideoStream	__init__(self, filename)	self, filename	*Constructor*
	nextFrame(self)	self	*A*
	frameNbr(self)	self	*Get frame number.*

FIGURE 1 – Function Description

3 Class diagram

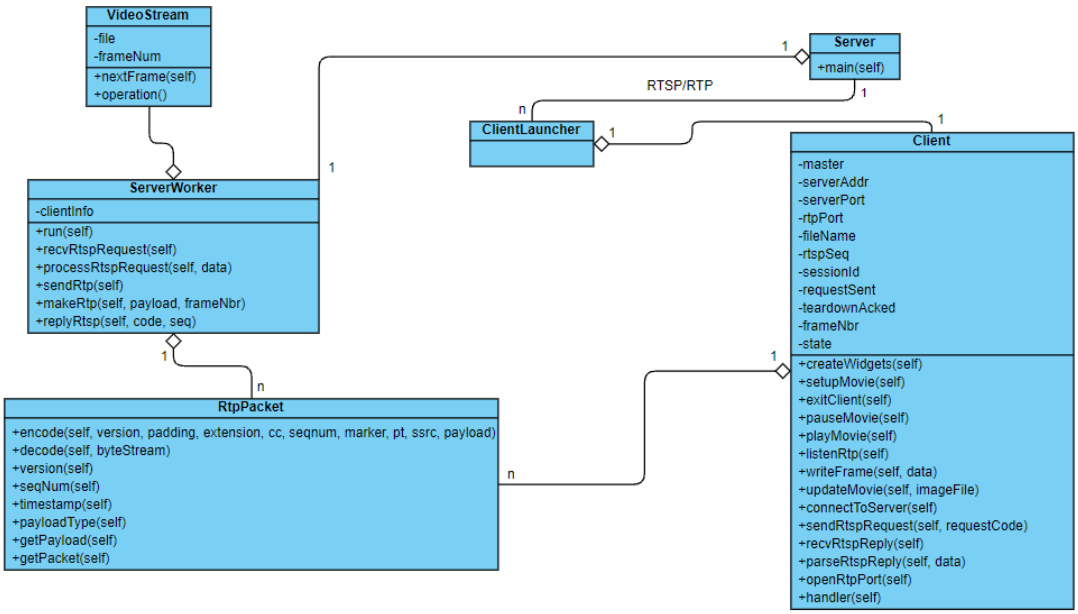

FIGURE 2 – Class Diagram

4 A summative evaluation of achieved results

- Complete the RTSP protocol in the client
- Complete the RTP packetization in the server (named as RtpPacket.py)
- Make the customized interactive player for displaying the transmitted video

5 User manual

— Step 1 : First, we run terminal and start the server with the command : **python Server.py server port**.
We should set the port number larger than 1024.
Example :

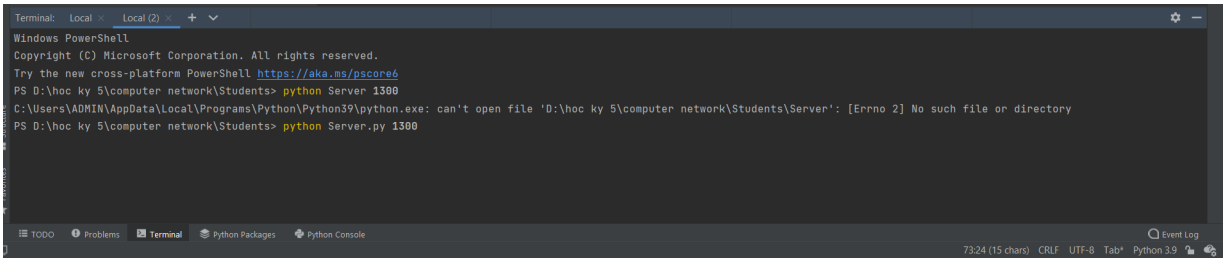

FIGURE 3 – Run server with port = 1300

— Step 2, start the client with the command **python ClientLauncher.py server-host server-port RTP-port video-file** where server host is the name of the machine where the server is running, server port is the port where the server is listening on, RTP port is the port where the RTP packets are received, and video file is the name of the video file you want to request (we have provided one example file movie.Mjpeg).
Example :

— server-host : LAPTOP-MMCHUE76 (use cmd to check)
— sever-port : 1300 (create in first step)
— RTP-port : We choose 6000
— video-file : movie.Mjpeg

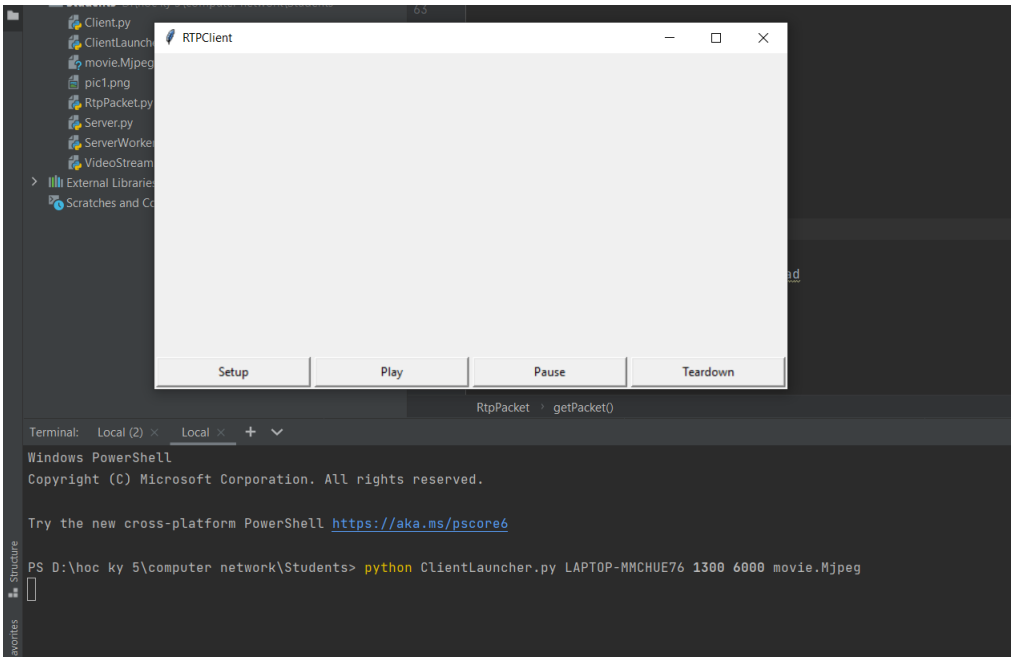

FIGURE 4 – Run Client

— Step 3, click **Setup** to create RTP connection.

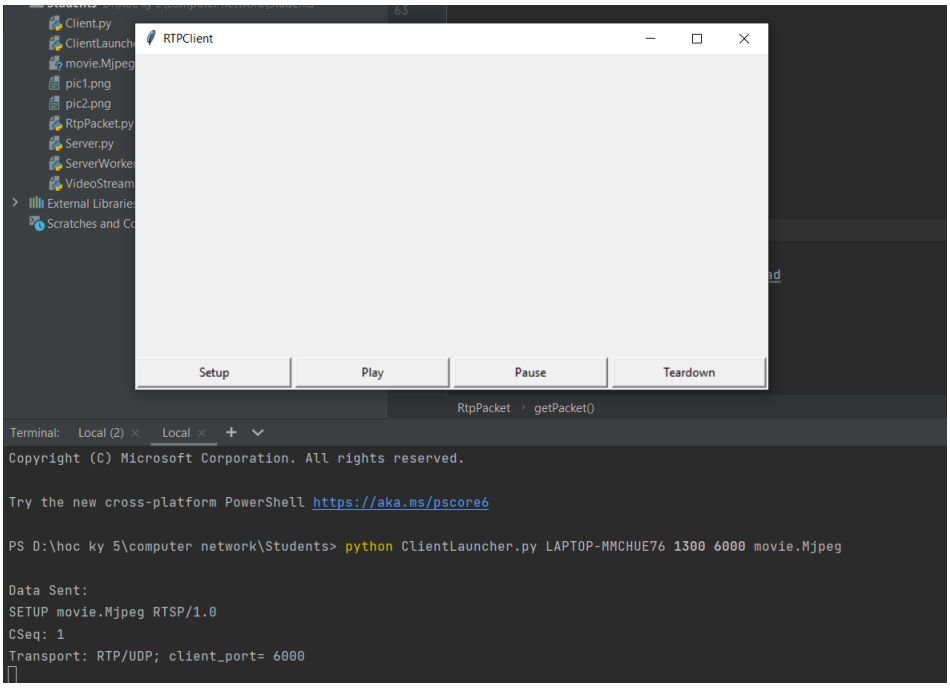

FIGURE 5 – Setup

— Final, we can press **Play** to play video, **Pause** to stop and **Teardown** to close the client UI.

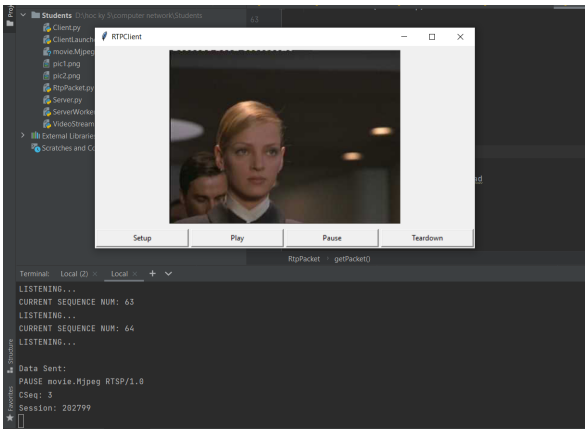

FIGURE 6 – Pause

— New GUI :

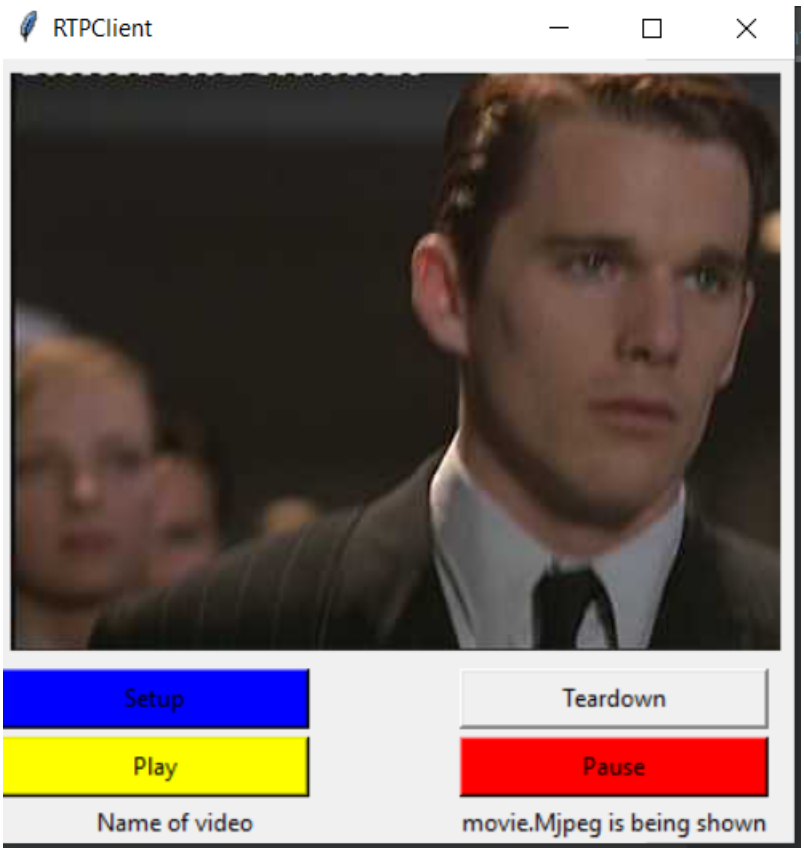

FIGURE 7 – new GUI

6 Extend

6.1 Calculate the statistics about the session. You will need to calculate RTP packet loss rate, video data rate(in bits or bytes per second),and any other interesting statistics that you can think of.

We have add some calculation for the code to calculate some statistics, which display in terminal after we quit the application.(data rate, packetlost, time)

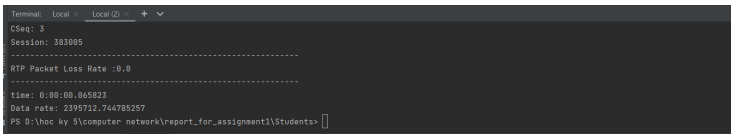

FIGURE 8 – statistics

6.2 The user interface on the RTPClient has 4 buttons for the 4 actions. If you compare this to a standard media player, such as RealPlayer or Windows Media Player, you can see that they have only 3 buttons for the same actions : PLAY, PAUSE, and STOP (roughly corresponding to TEARDOWN). There is no SETUP button available to the user. Given that SETUP is mandatory in an RTSP interaction, how would you implement that in a media player ? When does the client send the SETUP ? Come up with a solution and implement it. Also, is it appropriate to send TEARDOWN when the user clicks on the STOP button ?

i would implement a state machine with 3 state : init, playing and paused as below :
The client will send SETUP right after we pressed play button. It is also appropriate to send

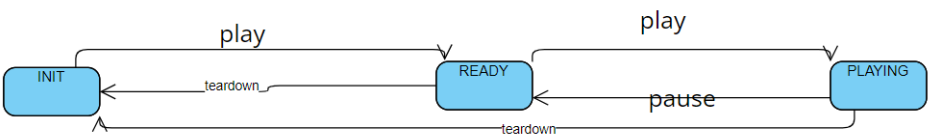

FIGURE 9 – FSM

TEARDOWN when user clicks on STOP button.

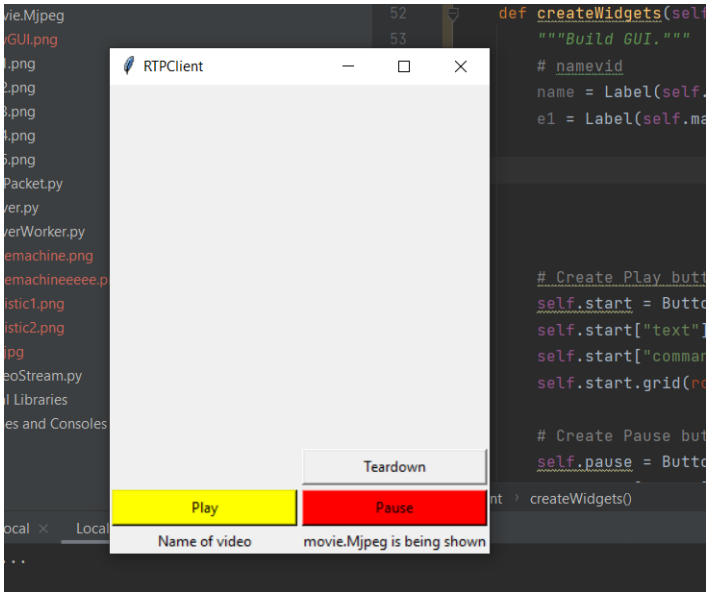

FIGURE 10 – scrs1

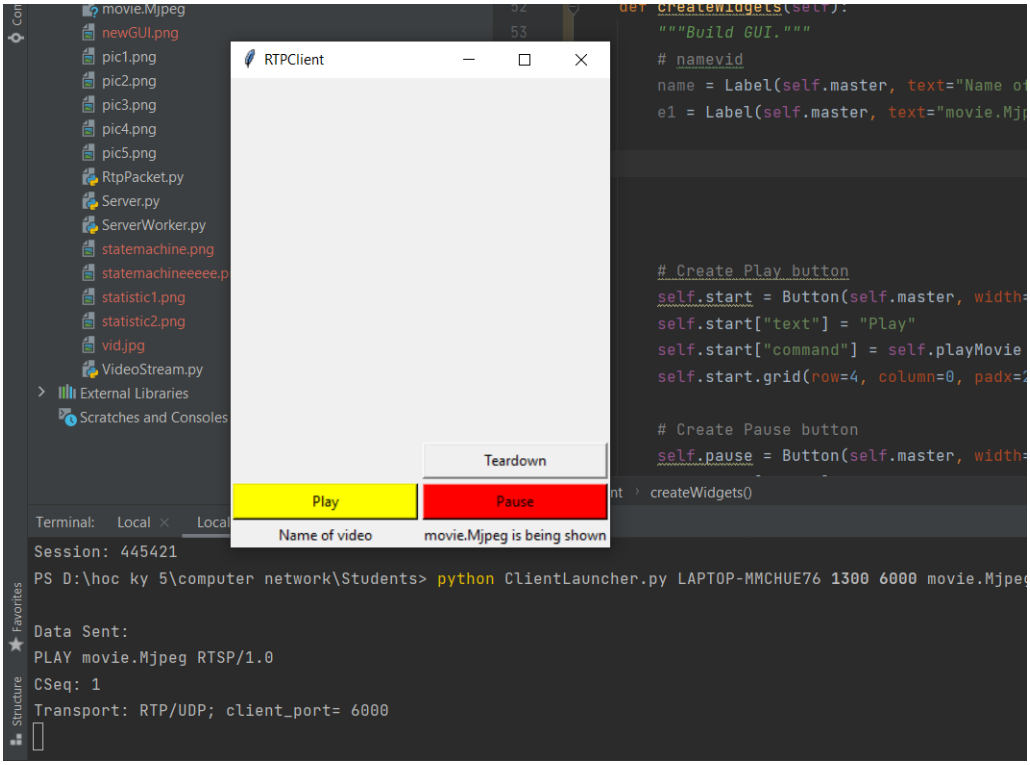

FIGURE 11 – scrs2

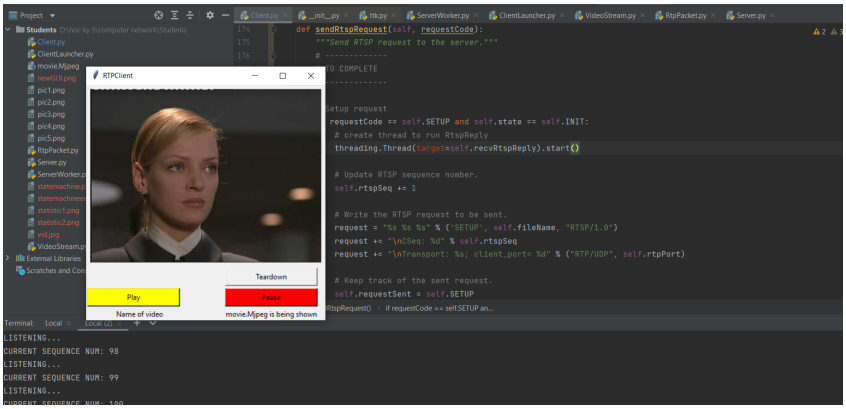

FIGURE 12 – scrs3

6.3 Currently, the client and server only implement the minimum necessary RTSP interactions and PAUSE. Implement the method DESCRIBE which is used to pass information about the media stream. When the server receives a DESCRIBE-request, it sends back a session description file which tells the client what kinds of streams are in the session and what encodings are used.

When pressed Describe button, the information of video is displayed at right corner.

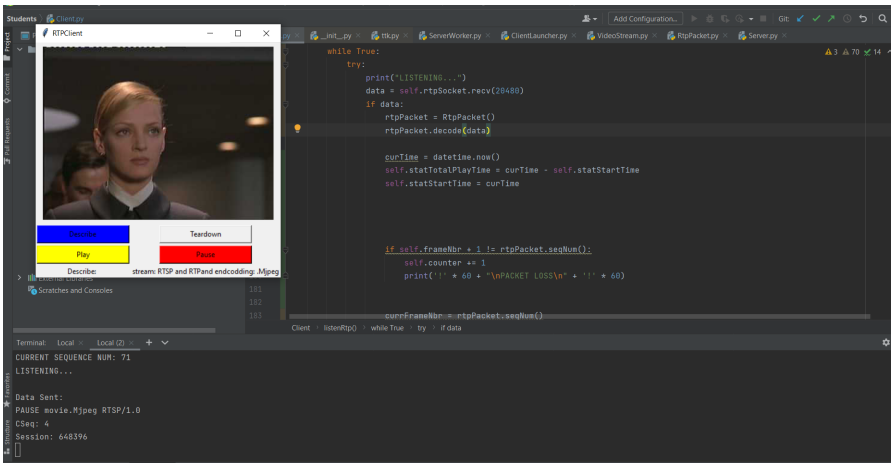

FIGURE 13 – scrs4

6.4 Implement some additional functions for user interface such as : display video total time and remaining time, fast forward or backward video (or make a scroll bar for scrolling video if you can)

6.5 Add one more state to the client (for example SWITCH state) so that user can select another video from a list of videos received from server.

7 Full source code

Github : https ://github.com/minhle270901/1952844.git