Assignment 01: Streaming Video Server & Client

Presentation

Le Tien Loc, Pham Tan Phuoc, Tran Tien Phat

Vietnam National University
Ho Chi Minh City University of Technology

November 24, 2021

Functional requirements

Implement an application that transmits live video between the server and the client using the RTSP protocol in the server, RTP de-packetization in the client and takes care of the display of the transmitted video.

- **RTSP** (Real-Time Streaming Protocol) commonly used in entertainment and communication systems to control streaming media servers and control transmission sessions between endpoints. RTSP runs over the TCP protocol.
- **RTP** (Real-time Transport Protocol) is a protocol used to transfer video and audio files over IP networks, commonly used in Streaming media systems. RTP runs over UDP protocol.

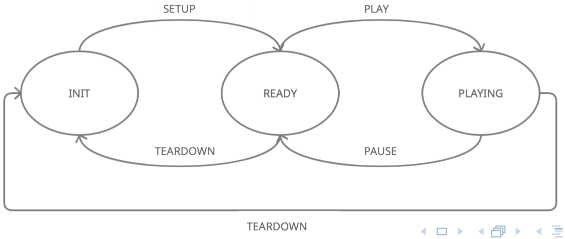
Non-functional requirements

- Create a datagram socket for receiving RTP data and set the timeout on the socket to 0.5 seconds.
- Need to choose the server port number greater than 1024.
- When sending a video frame to the client, the frame is sent over UDP every 50 milliseconds.
- The video to be played is formatted with the MJPEG file format.

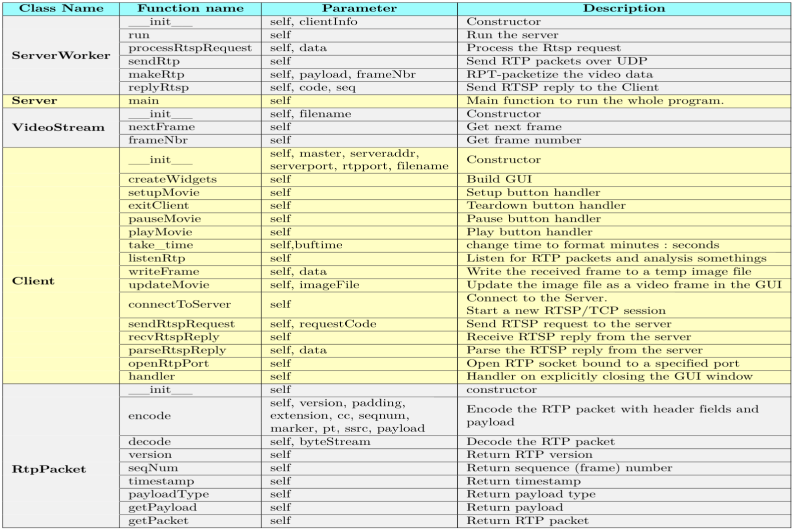
Main functions of the application

When the client starts, it also turns on the RTSP socket to the server. When the client clicks on the buttons on the user's interface, the functions are sent by the socket to the server. The main functions include:

- SETUP
- PLAY
- PAUSE
- TEARDOWN



List of components



Class Name	Function name	Parameter	Description
ServerWorker	__init__	self, clientInfo	Constructor
	run	self	Run the server
	processRtspRequest	self, data	Process the Rtsp request
	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
Server	main	self	Main function to run the whole program.
VideoStream	__init__	self, filename	Constructor
	nextFrame	self	Get next frame
	frameNbr	self	Get frame number
Client	__init__	self, master, serveraddr, serverport, rtpport, filename	Constructor
	createWidgets	self	Build GUI
	setupMovie	self	Setup button handler
	exitClient	self	Teardown button handler
	pauseMovie	self	Pause button handler
	playMovie	self	Play button handler
	take_time	self,buftime	change time to format minutes : seconds
	listenRtp	self	Listen for RTP packets and analysis somethings
	writeFrame	self, data	Write the received frame to a temp image file
	updateMovie	self, imageFile	Update the image file as a video frame in the GUI
	connectToServer	self	Connect to the Server. Start a new RTSP/TCP session
	sendRtspRequest	self, requestCode	Send 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 bound to a specified port
	handler	self	Handler on explicitly closing the GUI window
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

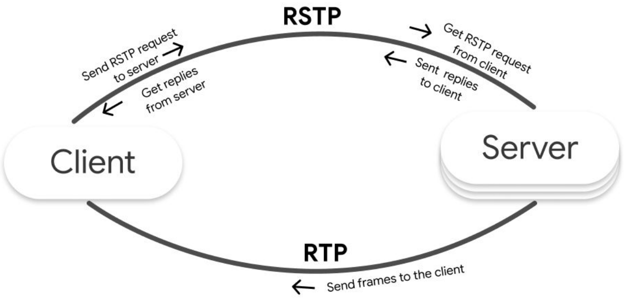
Model

1. Client
   - Master to save state: PLAY, PAUSE, SETUP, TEARDOWN,...
   - Server address
   - Server port
   - RTP port
   - File name which is streamed

     python ClientLauncher.py 127.0.0.1 1025 5008 video.mjpeg
2. Server
   - Server port

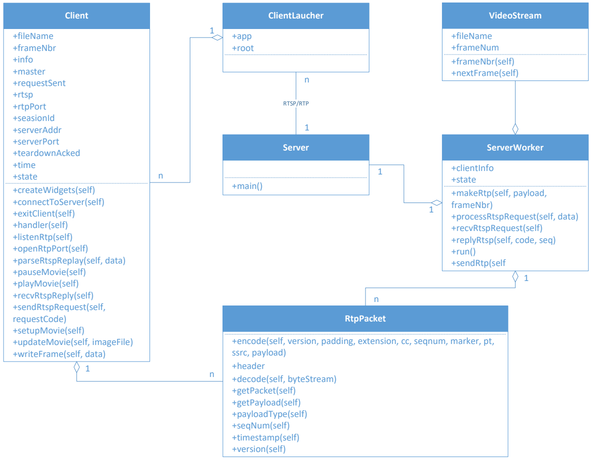
     python Server.py 1025

Data flow


RSTP
Send RSTP request to server
Get replies from server
Get RSTP request from client
Sent replies to client
Client
Server
RTP
Send frames to the client

- RTSP is an application-layer protocol used for commanding streaming media servers via pause and play capabilities.
- The Real-Time Streaming Protocol (RTSP) establishes and controls either a single or several time-synchronized streams of continuous media such as audio and video.

Class diagram



Server.py

```
python Server.py <port_server>
```

ClientLauncher.py

```
python ClientLauncher.py <host_server> <port_server>
<port_RTP> <name_video>
```

Summative evaluation of achieved results

In this assignment we have achieved:

- Complete the RTSP protocol at the client.
- Complete the RTP protocol at the server.

The request and reply messages sent from the client and server are displayed on the client side when SETUP and PLAY are pressed, the server returns RTSP/1.0 200 OK if the request is successful. And when the user presses TEARDOWN, the session ends, the interface window closes and returns the RTSP/1.0 200 OK message to the user.