# Video Stream Programming Report

The Program includes 5 files:

Client.py

ClientLauncher.py

RtpPacket.py

Server.py

ServerWorker.py

What we will do is

1.Run Server.py on Server Terminal to start server:

E.g.

python Server.py server\_port

server\_port is the port your server listens to for incoming RTSP connections

# we can give it the value 1025

# Standard RTSP port is 554

# In this project we shall make the value > 1024

2.Run ClientLauncher.py on Client Terminal to start a client:

E.g

python ClientLauncher.py server\_host server\_port PRT\_port video\_file

server\_host is the IP address of local machine (we can use “127.0.0.1” )

server\_port is the port the server is listening on (here “1025”)

RTP\_port is the port where RTP packets are received (here “5008”)

video\_file is the name of video file that we want to play (here “video.mjpeg”)

RTSP

Real Time Streaming Protocol

For entertainment and communications systems to control streaming media servers

Establishing and controlling media sessions between end points

It uses TCP

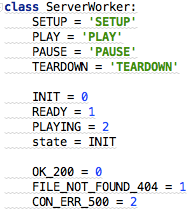
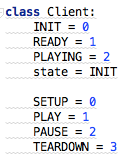
RTP

Real-time Transport Protocol

Network protocol for delivering audio and video over IP Networks

It uses UDP

How RTSP and RTP work together?

What will be sent from client to server via RTSP Protocol are the commands like

SETUP

PLAY

PAUSE

TEARDOWN

These commands will let server side know what is next action it should complete.

What will be replied from server to client via RTSP Protocol are the parameters like:

OK\_200

FILE\_NOT\_FOUND\_404

CON\_ERR\_500

To tell the client if the server receive its commands correctly

After client receives server`s reply, it will change its state accordingly to :

READY

PLAYING

If SETUP command was sent from client to server



The “SETUP” RTSP Packet will include

1. SETUP command

2.Video file name to be play

3.RTSP Packet Sequence Number starts from 1

4.Protocol type: RTSP/1.0 RTP

5.Transmission Protocol: UDP

6.RTP Port for video stream transmission

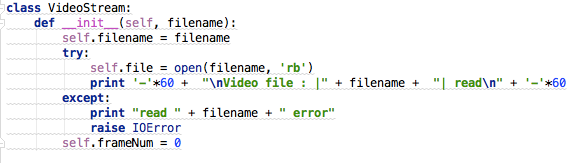


When Server side receives “SETUP” command ,it will

1.Assign the client a Specific Session Number randomly

2.If something wrong with this command or server`s state,it will reply ERROR packet back to client

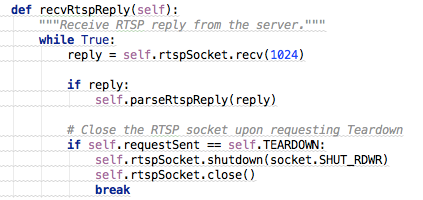
3.If command correct



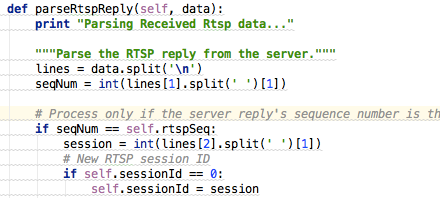
The server will open the video file specified in the SETUP Packet and Initialize its video frame number to 0

3.If command processes correctly, it will reply OK\_200 back to client and set its STATE to READY

The Client side will loop to receive Server`s RTSP Reply

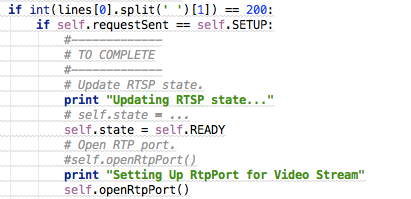


Then Parse the RTSP Relpy Packet:

Get the Session Number 

And if the Reply Packet is respond for the SETUP command

The client will set its STATE as READY

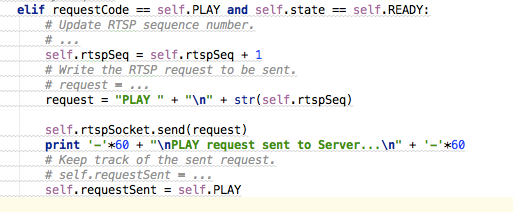


Then open a Rtp Port to receive incoming video stream

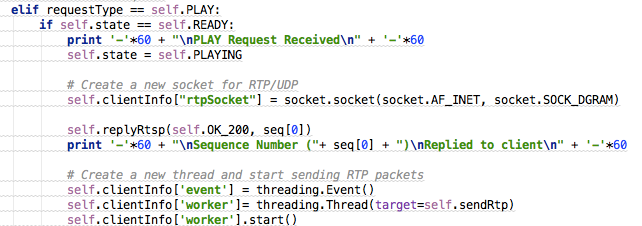


Afterward

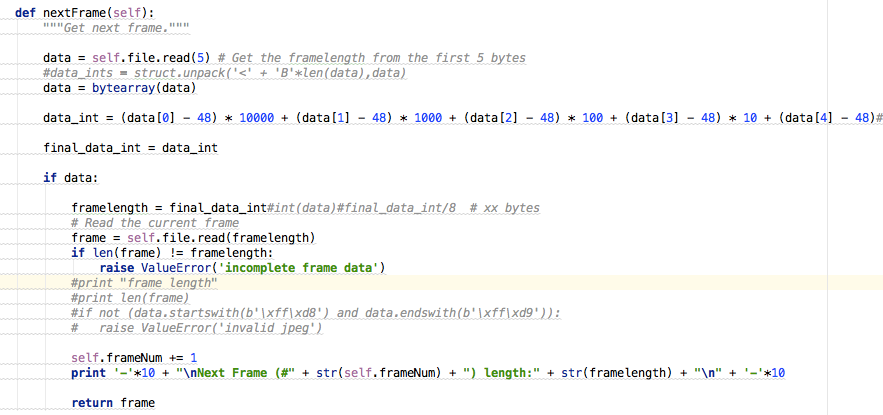
If PLAY RTSP command was sent from client to server:



The Server will create a Socket for RTP transmission via UDP, and start a tread to send video stream packet



VideoStream.py will help chop the video file to separate frame , and put each frame into RTP data packet



Each data packet will also be encoded with a header, the header will include

RTP-version filed

Padding

extension

Contributing source

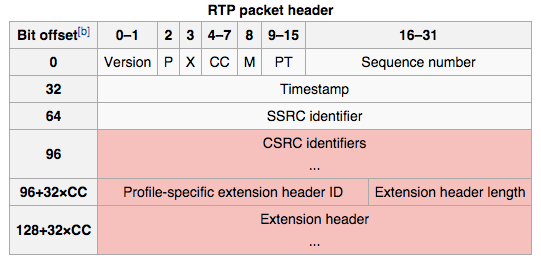
Marker

Type Field

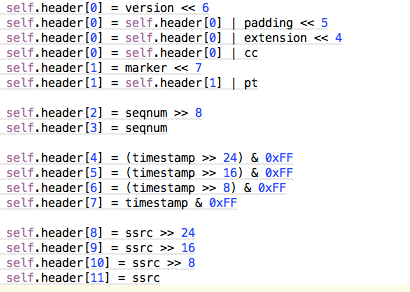
Sequence Number

Timestamp

SSRC



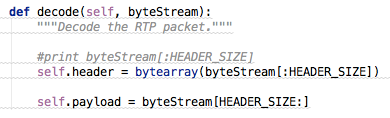
they have been inserted in the RTP Packet via bitwise operations



Finally the RTP Packet will include a header and a video frame be sent to the RTP Port on the client side:



Then Client decode the RTP Packet to get the header and the video frame, reorganize the frames and display on the UI



If a PAUSE command was sent from client to server , it will stop the server from sending video frames to client

If a TEARDOWN command was sent from client to server, it will also stop the server from sending video frames to client and close the client terminal as well