**Function description:**

Chương trình bao gồm 5 file:

- Client.py

- ClientLauncher.py

- RtpPacket.py

- Server.py

- ServerWorker.py

Cái mà chúng ta cần làm:

1. Chạy file Server.py trên command line để start server:

python Server.py server\_port

Trong đó chúng ta sẽ set server\_port là 1025, mặc dù port tiêu chuẩn là 554, nhưng trong project này thì sẽ để cho value > 1024.

1. Chạy file ClientLauncher.py trên 1 command line khác để start client:

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

Trong đó server\_host là địa chỉ IP của máy local (127.0.0.1), server port là port của server(1025), RTP\_port là port mà RTP\_packet nhận được, video\_file là file video muốn stream.

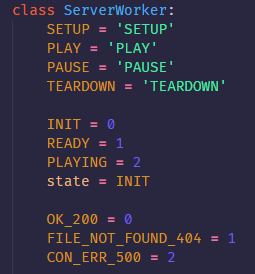
RTSP: Real Time Streaming Protocol sử dụng giao thức TCP

Sử dụng cho các hệ thống giải trí và truyền thông, điều khiển các server. Thành lập và điều khiển các media sessions giữa các điểm cuối.

RTP: Real-time Transport Protocol sử dụng UDP

Giao thức mạng để chuyển giao audio và video thông qua IP networks.

Cách mà RTSP và RTP hoạt động cùng nhau:

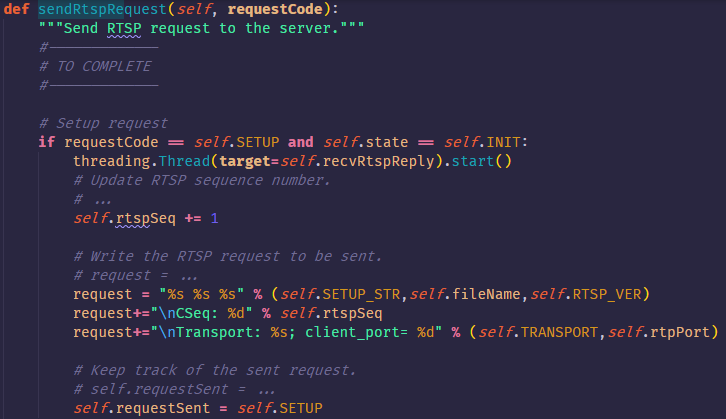


Những command giống như hình trên sẽ được gửi từ client tới server thông qua giao thức RTSP (SETUP, PLAY, PAUSE, TEARDOWN), những command này sẽ để server side biết nó cần hoàn thành action gì tiếp theo.

Những tham số được đáp lại từ sever trả về client(OK\_200, FILE\_NOE\_FOUND\_404, CON\_ERR\_500)

Sauk hi client nhận được phản hồi của server, nó sẽ thay đổi trạng thái (READY, PLAY)

Khi SETUP command được gửi từ client đến server:



1 gói “SETUP” RTS sẽ bao gồm:

- SETUP command

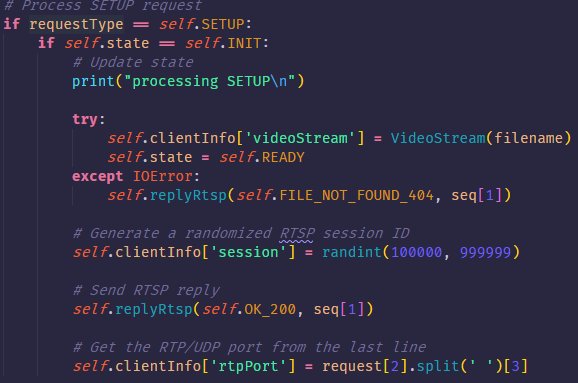
- Tên của video

- Số RTSP Packet Sequence bắt đầu từ 1

- Loại giao thức: RTSP/1.0 RTP

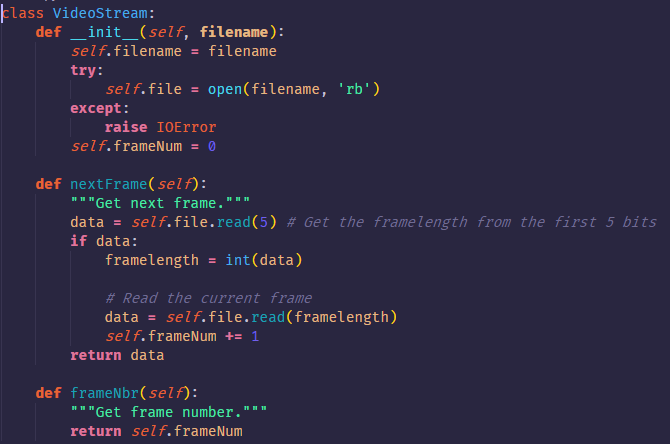
- Giao thức truyền: UDP

- RTP port để truyền video stream



Khi mà server nhận được SETUP command:

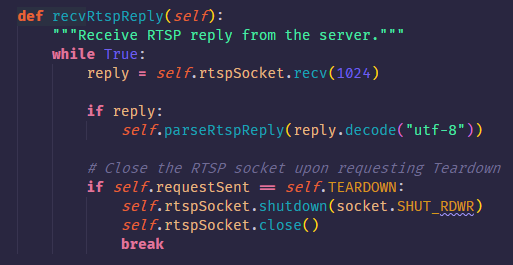
1. Gán cho khách hàng một Specific Session Number ngẫu nhiên.
2. Nếu command bị sai hoặc server bị lỗi, nó sẻ trả 1 gói ERROR về phía client.
3. Nếu command đúng:



Server sẽ mở file video được chỉ định trong SETUP Packet và Khởi tạo số khung hình video của nó thành 0

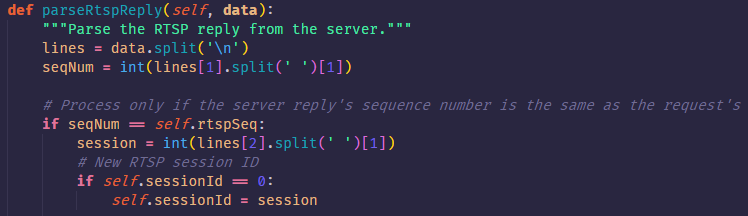
1. Nếu command thực thi chính xác, nó sẽ trả status OK\_200 về cho client và set trạng thái STATE thành READY

Client sẽ thực hiện vòng lặp để nhận được phản hồi của Server’s RTSP:

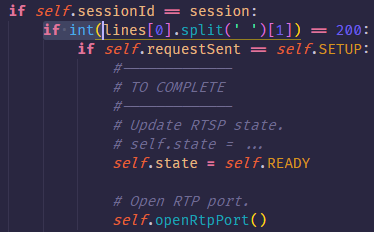


Sau đó, phân tích cú pháp trong gói RTSP Relpy:

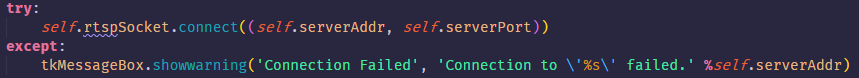
Lấy Session Number:



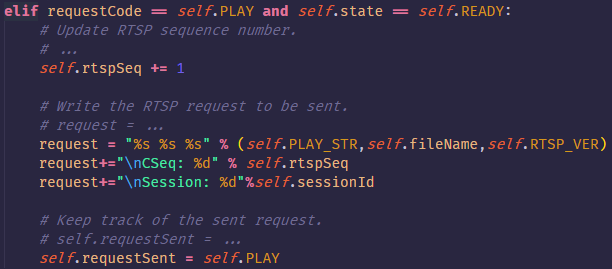
Và nếu gói reply trả vè SETUP command thì client sẽ set trang thái từ STATE sang READY



Sau đó mở RTP port để nhận được video stream được gửi tới.



Sau đó, nếu command PLAY được gửi rừ client tới server:

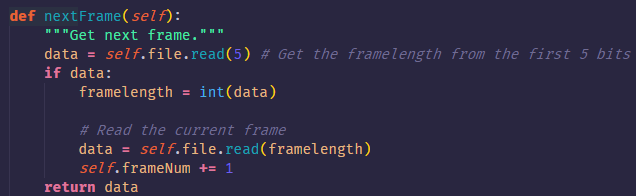


The Server will create a Socket for RTP transmission via UDP, and start a tread to send video stream packet

Server sẽ tạo 1 Socket cho RTP truyền qua UDP, và bắt đầu gửi video stream packet.



VideoStream.py sẽ giúp cắt file video thành từng khung riêng biệt và đưa từng khung vào gói dữ liệu RTP.



Mỗi gói data sẽ được mã hóa cùng với header, header sẽ bao gồm:

- RTP-version filed

- Padding

- Extension

- Contributing source

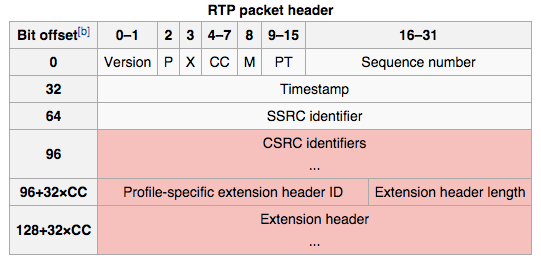
- Marker

- Type Field

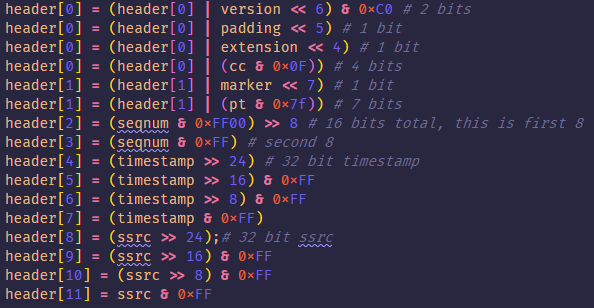
- Sequence Number

- Timestamp

- SSRC



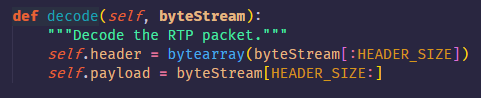
Chúng đã được chèn vào Gói RTP thông qua các phép toán thao tác bit như bên dưới:



Cuối cùng, Gói RTP sẽ bao gồm một header và một khung video được gửi đến Cổng RTP ở phía client:

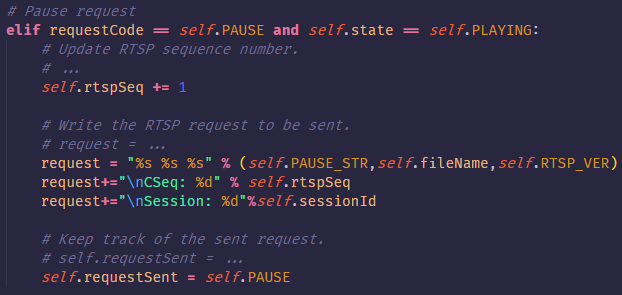


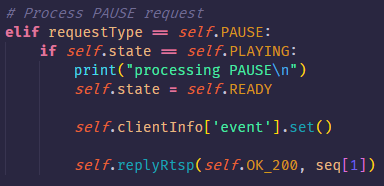
Sau đó, client mã hóa Gói RTP để lấy header và khung video, tổ chức lại các khung và hiển thị trên giao diện người dùng.



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

Nếu 1 command PAUSE được gừi từ client đế server, nó sẽ buộc dừng sever gửi khung video đến cho client.





Nếu 1 command TEARDOWN được gửi từ client đến server. Nó cũng sẽ buộc dừng sever gửi khung video đến cho client và đồng thời cũng đóng terminal.

