**TRƯỜNG ĐẠI HỌC BÁCH KHOA TP.HCM**

**MẠNG MÁY TÍNH**



**BÀI TẬP LỚN 1**

**REAL TIME STREAMING PROTOCOL**

**GVHD: BÙI XUÂN GIANG**

**Thành viên:**

**1. Nguyễn Văn Hoàn 1711376**

**2. Ngô Đức Kiên 1812703**

**3. Trần Thanh Tuấn 2020114**

Contents

[**1.** **Phân tích yêu cầu:** 3](#_Toc56275944)

[**2.** **Chi tiết các chức năng của ứng dụng** 3](#_Toc56275945)

[**3.** **Thiết kế chi tiết ứng dụng** 4](#_Toc56275946)

[**4.** **Hướng dẫn sử dụng** 20](#_Toc56275947)

[**5.** **Đánh giá kết quả đạt được** 25](#_Toc56275948)

[**II.** **Phần mở rộng (Extend)** 26](#_Toc56275949)

**I.**

## **Phân tích yêu cầu:**

* Hiện thức giao thức RTSP trong client

RTSP là một giao thức cung cấp khung để truyền dữ liệu phương tiện theo thời gian thực ở cấp ứng dụng. Nó truyền dữ liệu thời gian thực từ đa phương tiện sang thiết bị đầu cuối bằng cách giao tiếp trực tiếp với máy chủ truyền dữ liệu.

Giao thức tập trung vào việc kết nối và kiểm soát các phiên phân phối dữ liệu trên các dòng đồng bộ hóa thời gian cho phương tiện liên tục như video và âm thanh. Tóm lại, giao thức truyền phát thời gian thực hoạt động như một điều khiển từ xa mạng cho các tệp phương tiện thời gian thực và máy chủ đa phương tiện.

* Tạo nhịp độ RTP trong máy chủ

RTP (Giao thức Vận chuyển Thời gian Thực) đặc tả một tiêu chuẩn định dạng gói tin dùng để truyền âm thanh và hình ảnh qua internet

## **Chi tiết các chức năng của ứng dụng**

1. Khởi tạo Streaming Server cho phép Client kết nối đến và giao tiếp thông qua giao thức RTSP/TCP

- Khởi tạo Server và socket RTSP/TCP, lắng nghe và cho phép tạo kết nối với nhiều máy khách kết nối đến cùng lúc.

2. Khởi tạo Client và giao diện người dùng để gửi các yêu cầu RTSP.

- Khởi tạo Client, socket để tạo kết nối với Server và khởi tạo giao diện. Sau khi kết nối thành công, người dùng thao tác gửi các yêu cầu RTSP đến máy chủ khi thao tác nhấn các nút nhấn trên giao diện.

3. Sử dụng giao thức RTP để xử lý dữ liệu video và phản hồi cho Client.

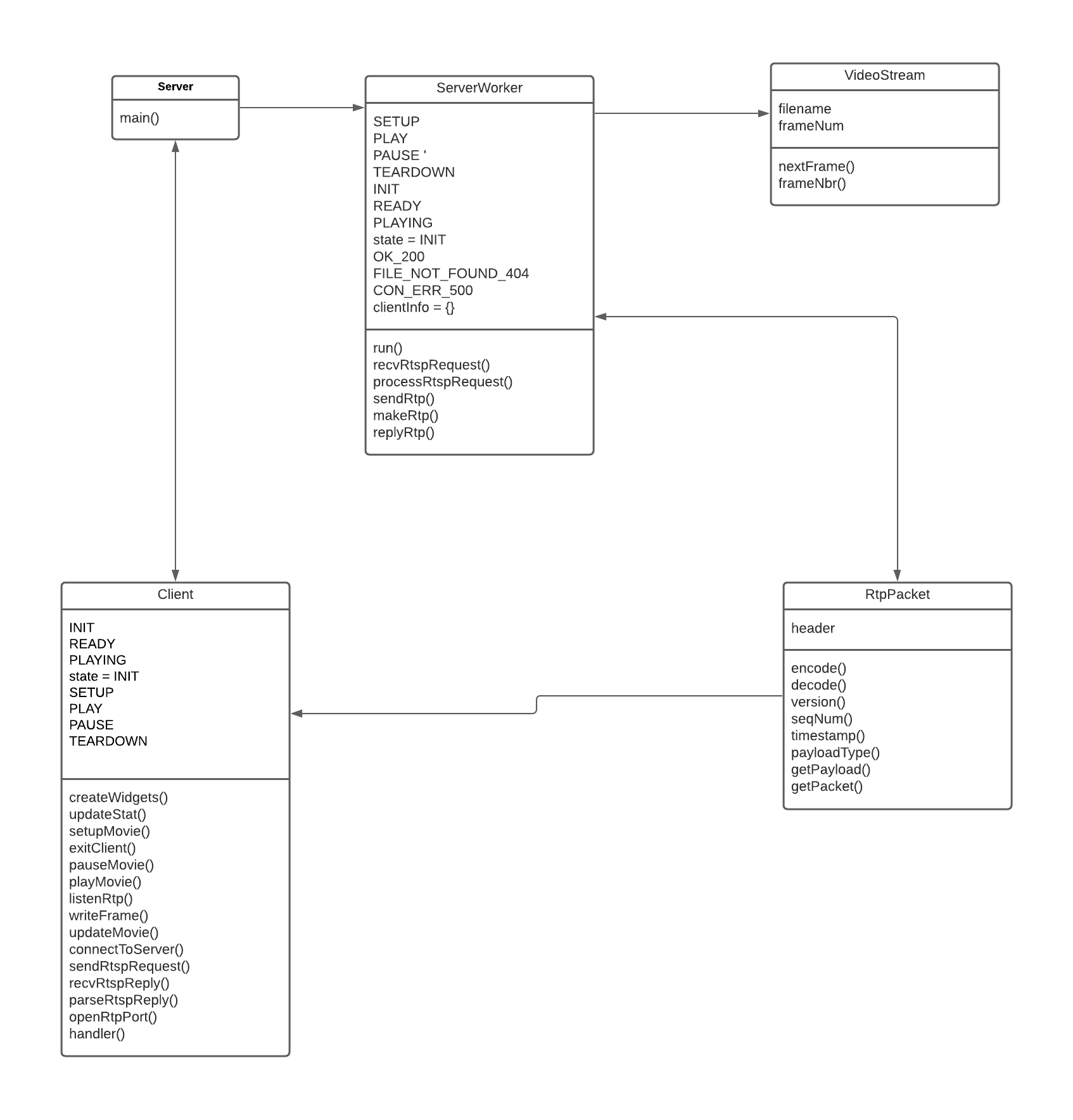
- Server nhận các yêu cầu RTSP của Client, xử lý dữ liệu video và đóng gói dưới dạng gói tin RTP, sau đó phản hồi cho Client.

4. Client nhận các gói tin RTP, xử lý và hiển thị video trên giao diện

- Client lắng nghe và nhận các gói tin RTP phản hồi từ Server, phân giải các gói tin và hiển thị video trên giao diện người dùng.

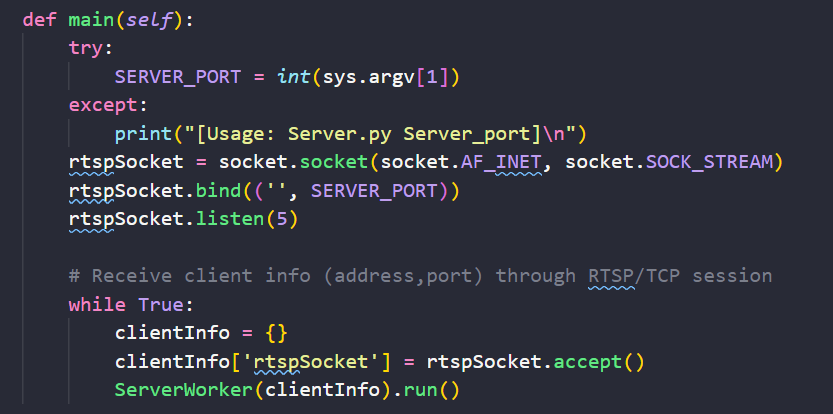
## **Thiết kế chi tiết ứng dụng**

1. Class Diagram



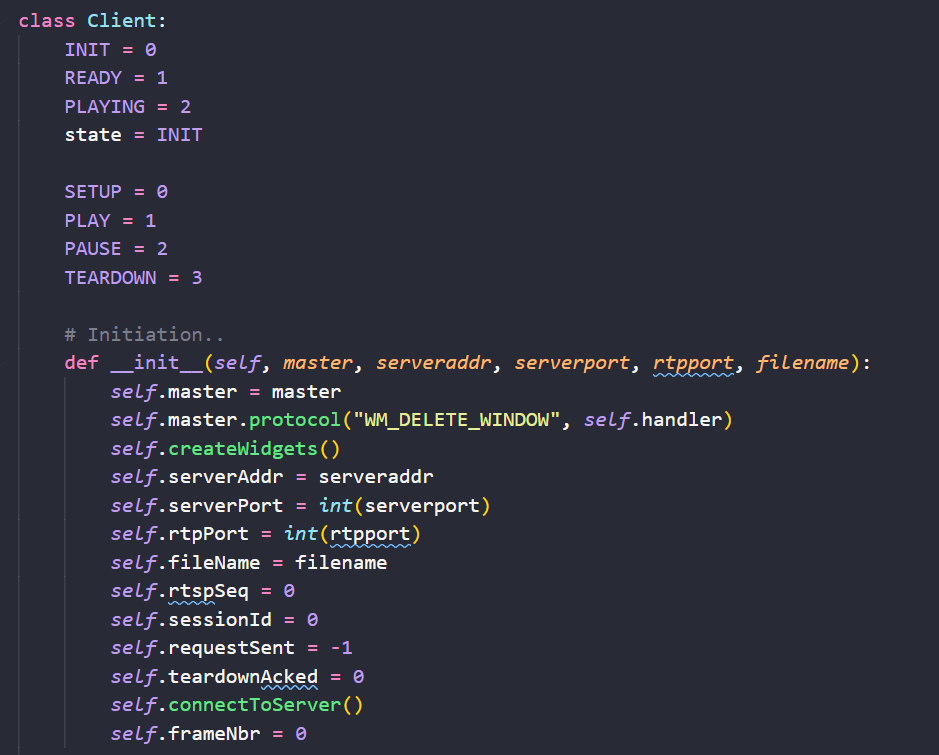
1. Phân tích luồng của ứng dụng

* Chạy Server.py trên Command Promt tại thư mục chứa ứng dụng để khởi động server. Câu lệnh: python Server.py server\_port. Server sẽ tiến hành khởi tạo socket RTSP/TCP, gán socket với địa chỉ IP và port mà chúng ta cung cấp, sau đó bắt đầu lắng nghe kết nối từ Server.



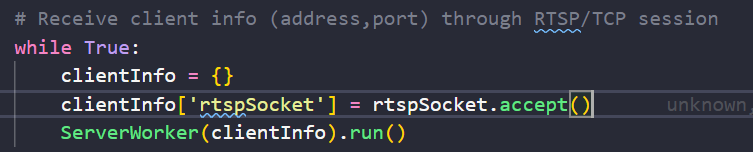
* Chạy ClientLaucher.py trên Command Promt tại thư mục chứa ứng dụng để khởi động Client. Câu lệnh: python ClientLauncher.py server\_host server\_port PRT\_port video\_file. ClientLaucher.py sẽ nhận vào các thông số người dùng cung cấp, gọi đến Client. Tại Client, tiến hành khởi tạo các trạng thái, thông số, tạo socket kết nối tới Server thông qua địa chỉ và port được cung cấp.

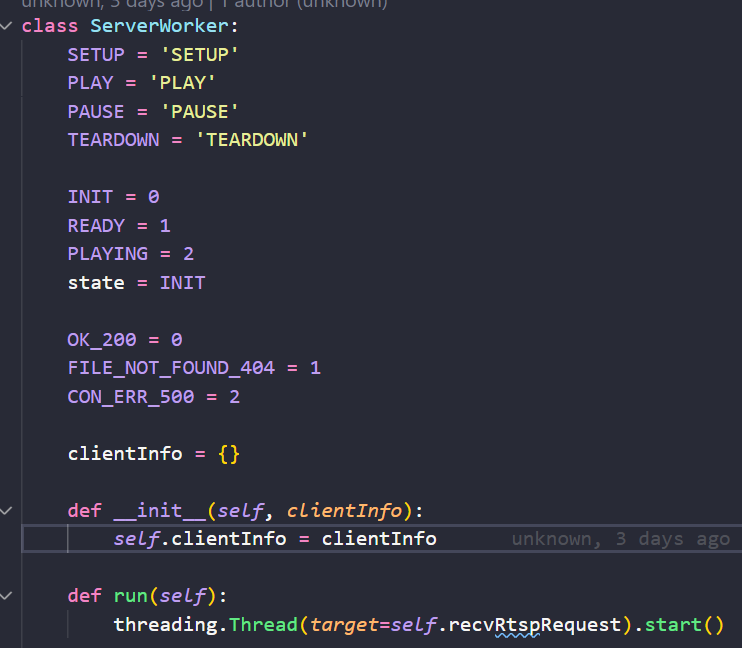




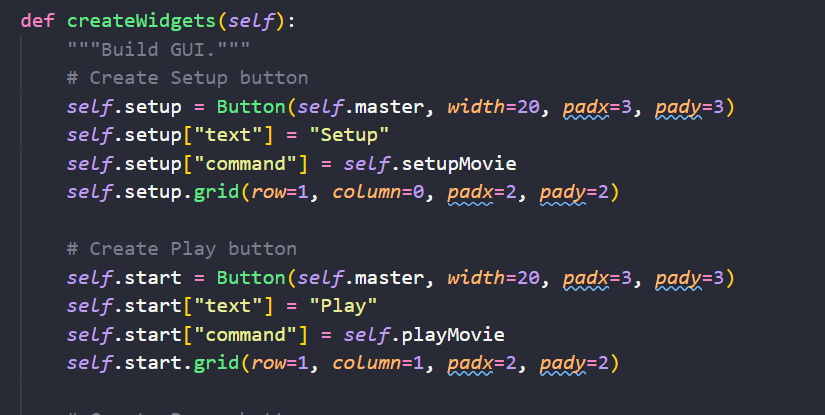


Server lắng nghe và chấp nhận kết nối khi nhận được yêu cầu kết nối từ Client. Tại ServerWorker, khởi tạo các trạng thái, thông số Server và sẵn sàng nhận các yêu cầu RTSP từ Client.

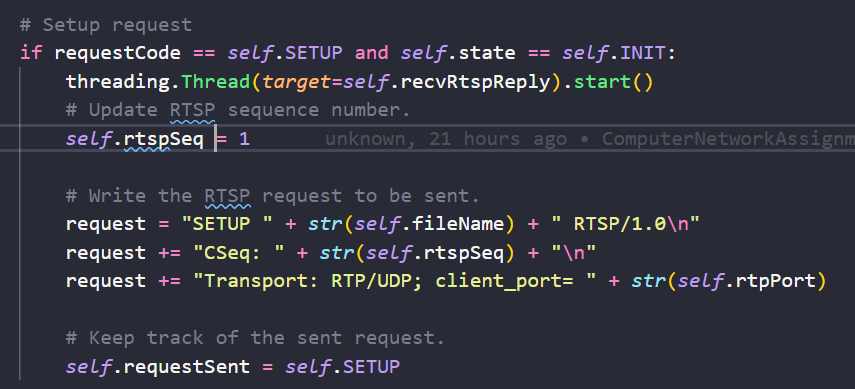




* Đồng thời, Client tiến hành khởi tạo giao diện người dùng.



* Sau khi đã khởi tạo Server và Client, kết nối với nhau qua giao thức RTSP/TCP. Client sẽ gửi đến Server các lệnh như “SETUP”, “PLAY”, “PAUSE”, “TEARDOWN” thông qua giao thức RTSP, các lệnh này sẽ cho phía Server biết hành động tiếp theo mà nó cần hoàn thành.
* Những gì sẽ được phản hồi từ Server đến Client thông qua Giao thức RTSP là các tham số như: “OK\_200”, “FILE\_NOT\_FOUND\_404”, “CON\_ERR\_500” để xác nhận với Client nếu Server đã nhận được yêu cầu của nó một cách chính xác.
* Sau khi Client nhận được câu trả lời của Server, nó sẽ thay đổi trạng thái tương ứng thành: “READY”, “PLAYING”.



* Nếu yêu cầu “SETUP” được gửi đến Server, gói tin RTSP “SETUP” sẽ bao gồm:

+ Yêu cầu “SETUP”

+ Tên Video muốn phát

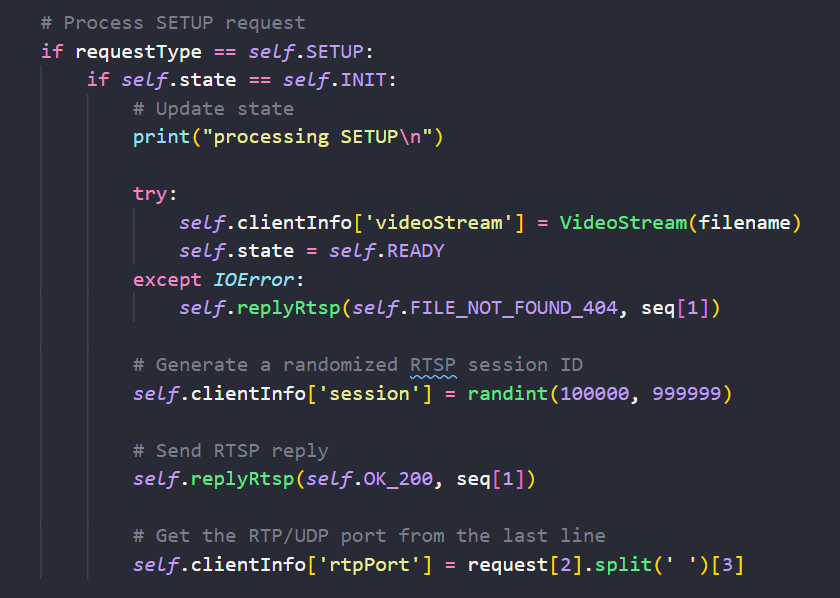
+ Protocol type: RTSP/1.0

+ Số RTSP Packet Sequence bắt đầu từ 1

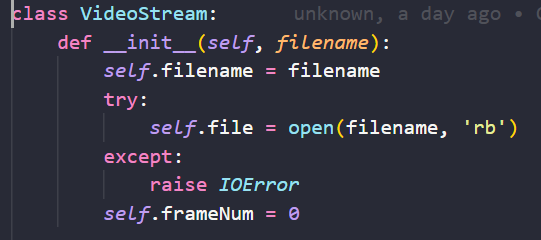
+ Transport Protocol: RTP/UDP

+ Cổng RTP để truyền dòng video

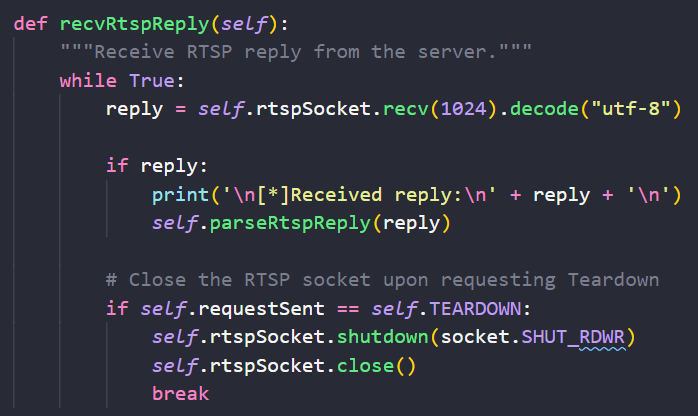
Thông tin về requestSent cũng được ghi lại.



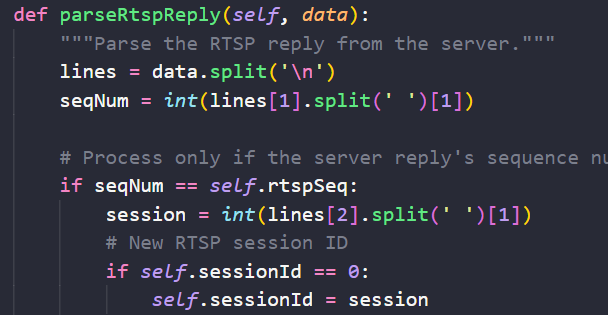
* Khi Server nhận được yêu cầu “SETUP”, nó sẽ gán cho Client ngẫu nhiên một Specific Session Number. Nếu có sai sót với yêu cầu hoặc trạng thái Server, nó sẽ phản hồi gói tin ERROR về cho Client. Nếu yêu cầu là chính xác, Server sẽ mở tệp video được chỉ định trong yêu cầu “SETUP” và khởi tạo số khung hình video của nó thành 0.
* Nếu yêu cầu được xử lý chính xác, Server sẽ trả lời lại OK\_200 cho Client và đặt STATE của nó thành “READY”.



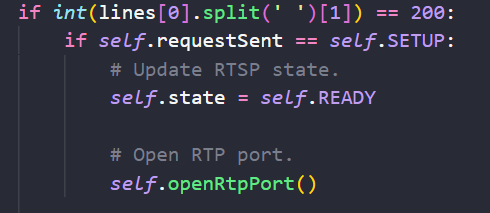
* Phía Client sẽ thực hiện vòng lặp để nhận RTSP Reply của Server:



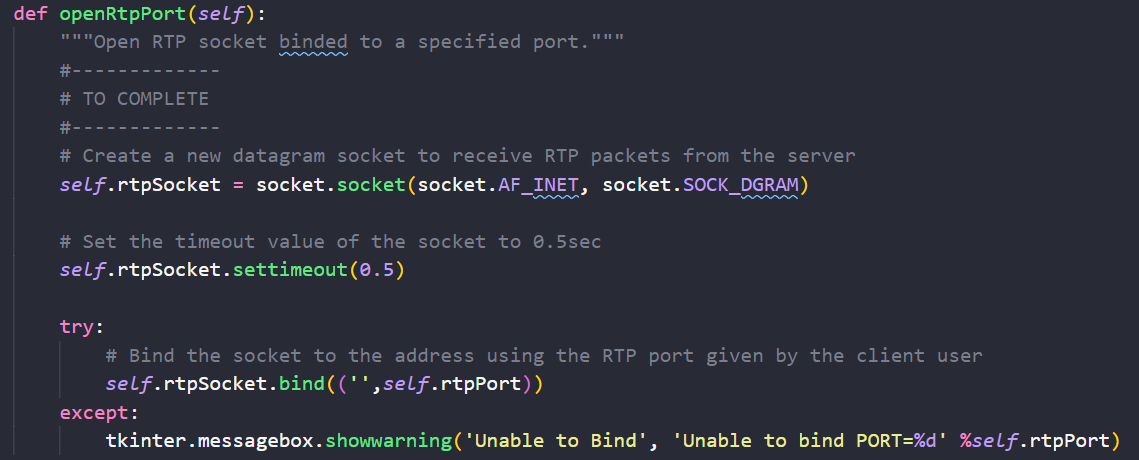
* Sau đó tiến hành phân tích cú pháp RTSP phản hồi từ máy chủ, lấy được Session Number do Server khởi tạo trước đó cho Client.



- Và nếu gói tin phản hồi cho lệnh “SETUP”, Client sẽ đặt trạng thái (STATE) là “READY”. Sau đó, mở một RtpPort để nhận luồng video đến.



* Khởi tạo RTP socket để nhận gói tin RTP từ server, thiết lập timeout cho socket này là 0.5 giây. Ràng buộc cho socket địa chỉ sử dụng RTP port được cung cấp bởi người dùng, nếu không thành công sẽ có cửa sổ thông báo hiện lên.



* Sau đó, nếu yêu cầu “PLAY” được gửi đến Server, Số RTSP Packet Sequence được tăng lên 1 và gói tin RTSP “PLAY” sẽ bao gồm:

+ Yêu cầu “SETUP”

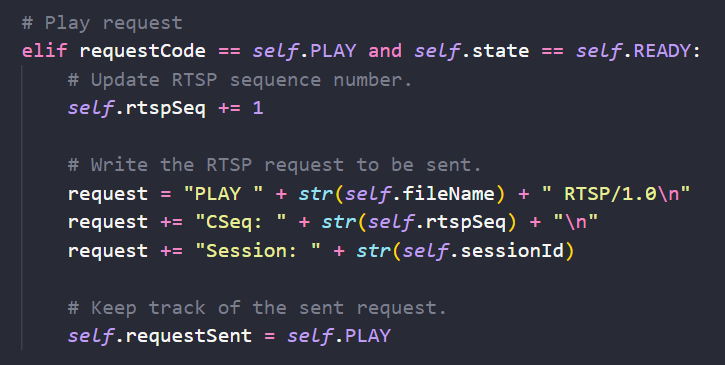
+ Tên Video muốn phát

+ Protocol type: RTSP/1.0

+ Số RTSP Packet Sequence

+ SessionId (được khởi tạo ngẫu nhiên từ ServerWorker và trả về từ phản hồi cho yêu cầu “SETUP”)

Thông tin về requestSent cũng được ghi lại.

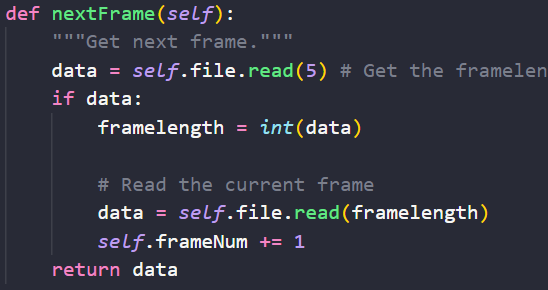


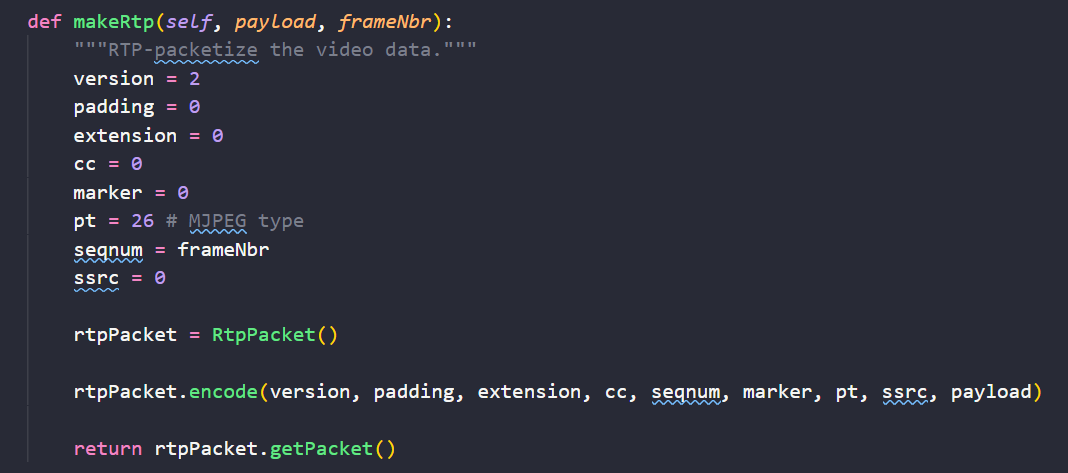
* Khi nhận được yêu cầu, Server sẽ tạo một Socket để truyền RTP qua UDP và khởi tạo một luồng (threading) để gửi gói tin RTP.



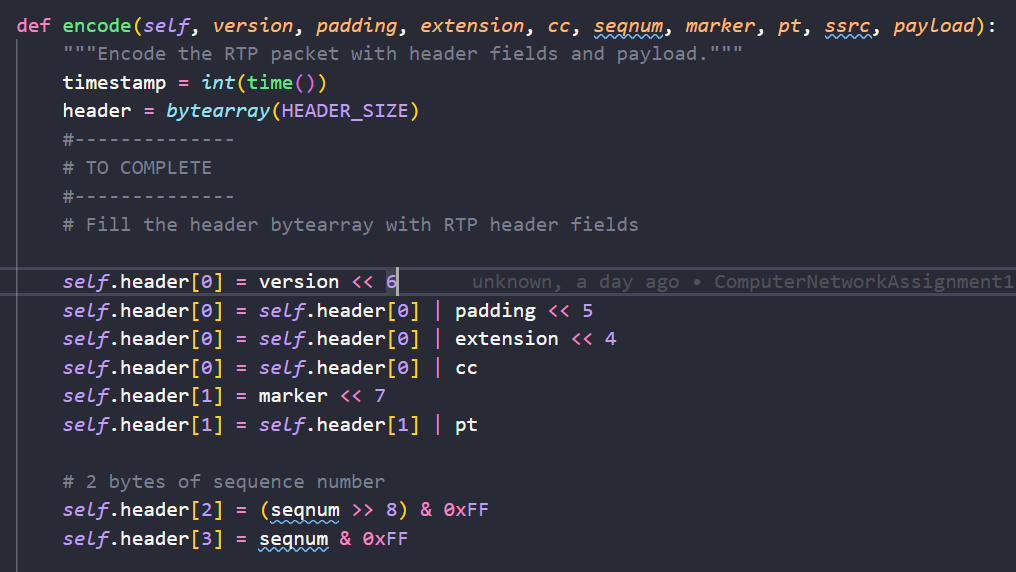
* VideoStream.py sẽ giúp cắt tệp video thành từng frame riêng biệt và đưa từng frame vào gói dữ liệu RTP.

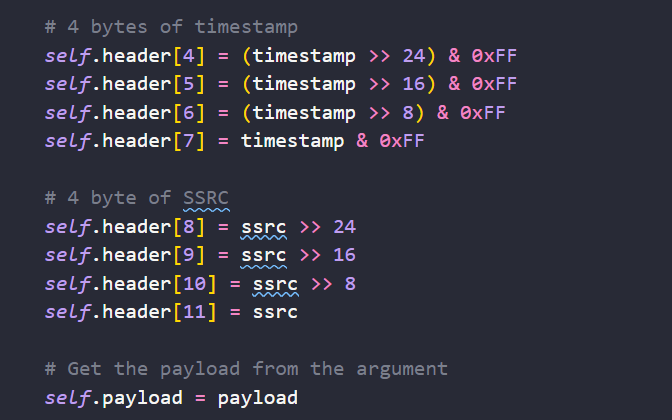




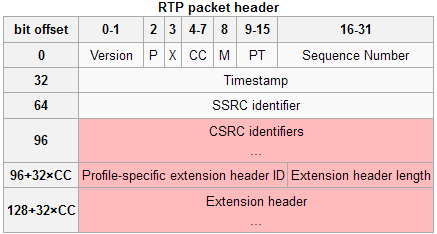


* Việc thực hiện mã hóa gói tin RTP gồm các trường của header và payload sẽ được thực hiện tại RtpPacket.py, chúng được chèn vào gói tin RTP thông qua các thao tác bitwise:





* Kích thước nhỏ nhất của một header của gói tin RTP là 12 bytes. Sau phần header chính, là phần header mở rộng và không cần thiết phải có phần header này. Chi tiết các trường trong một header như sau:



+ Version ( 2 bits): Cho biết phiên bản của giao thức này.

+ P (Padding) (1 bit) : Cho biết số các byte mở rộng cần thêm vào cuối của gói tin RTP. Ví dụ trong trường hợp ta muốn sử dụng các thuật toán mã hóa, ta có thể thêm vào một số byte vào phần kết thúc của gói tin để tiến hành mã hóa frame trên đường truyền.

+ X ( Extension) ( 1bit): Cho biết có thêm phần header mở rộng vào sau phần header chính hay không.

+ CC (CSRC Count) ( 4 bit) : Chứa con số định danh CSRC cho biết kích thước cố định của header.

+ M ( Marker) ( 1 bit) : Cho biết mức của ứng dụng và được định nghĩa bởi một profile. Nếu được thiết lập, có nghĩa là dữ liệu hiện tại đã được tính toán chi phí một cách thích hợp

+ PT (Payload Type) ( 7 bit) : Cho biết định dạng của file video. Đây là một đặc tả được định nghĩa bởi một profile RTP.

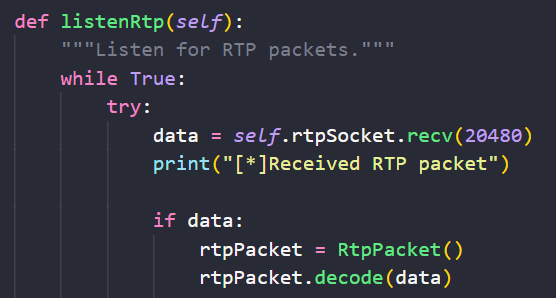
+ Sequence Number (16 bits) : số hiệu của frame. Và sẽ được tăng lên 1 đơn vị cho mỗi gói tin RTP trước khi gửi và được sử dụng bởi bên nhận để dò ra các gói bị lạc và có thể phục hồi lại gói có số thứ tự đó.

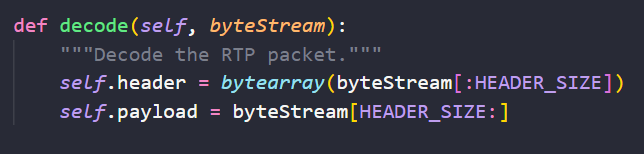
+ Timestamp ( 32 bits): Được sử dụng thông báo cho bên nhận biết để phát lại frame này trong khoảng thời gian thích hợp.

+ SSRC ( 32 bits): Định danh cho nguồn streaming. Mỗi nguồn cho phép streaming video sẽ định danh bởi một phiên RTP duy nhất.

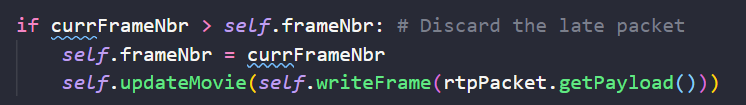


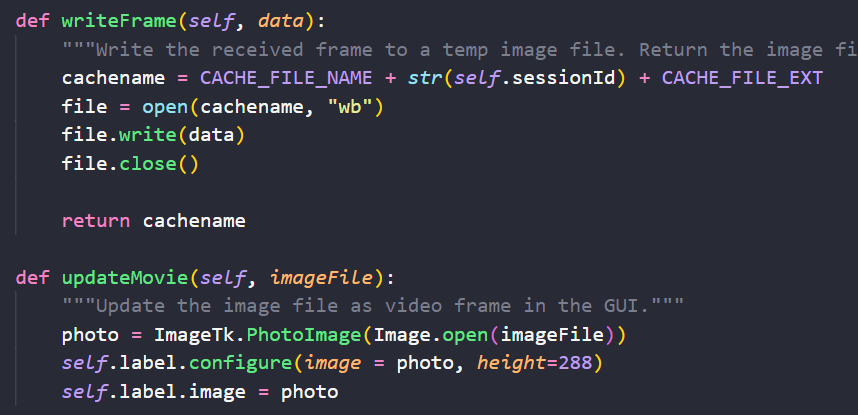
* Cuối cùng, gói tin RTP sẽ bao gồm một header và một frame của video được gửi đến RTP port ở phía Client.
* Phía Client sẽ lắng nghe để nhận các gói tin RTP Server gửi đến, sau đó tiến hành giải mã gói để lấy header và frame,



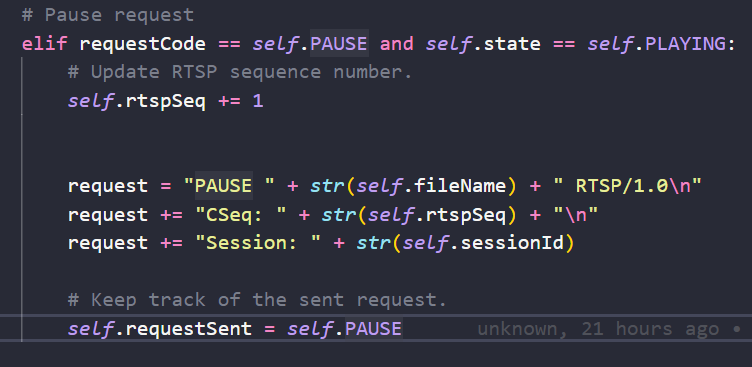


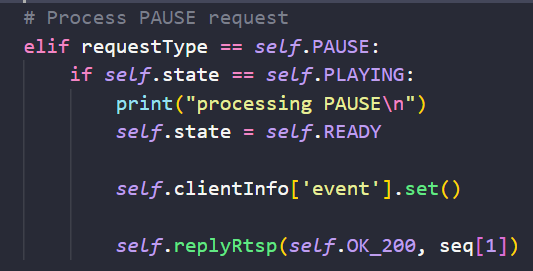
* Các frame sau đó sẽ được hiển thị trên giao diện người dùng:

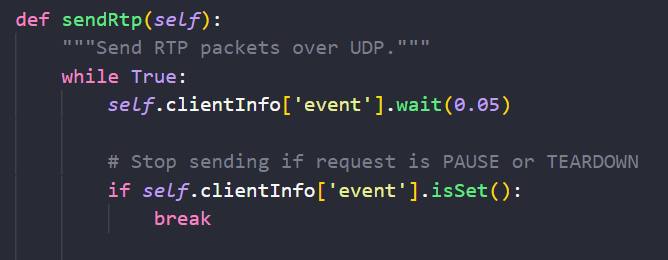




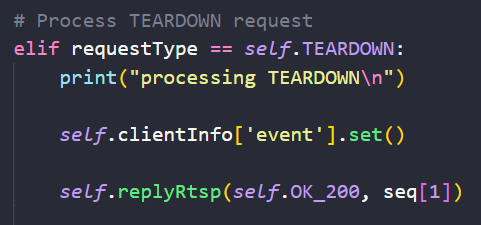
- Nếu lệnh “PAUSE” được gửi từ Client đến Server, nó sẽ ngăn Server gửi các frame đến Client.

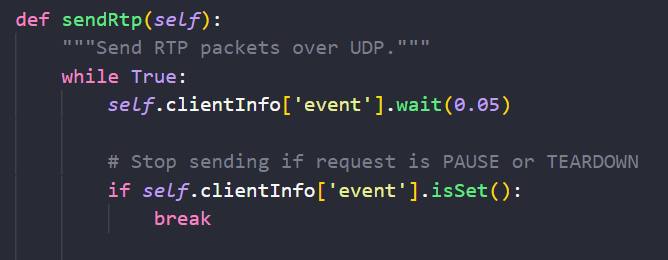


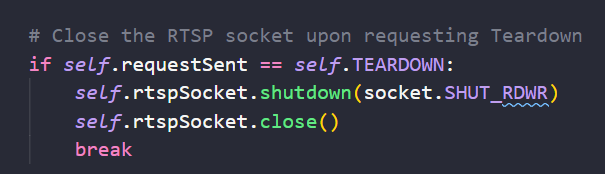




- Nếu lệnh “TEARDOWN” được gửi từ Client đến Server, nó cũng sẽ ngăn Server gửi các frame video đến Client và đóng cả kết nối của Client.







## **Hướng dẫn sử dụng**

Chương trình bao gồm:

Client.py

ClientLauncher.py

RtpPacket.py

Server.py

ServerWorker.py

VideoStream.py

Thực hiện chương trình như sau:

1. Chạy Server.py trên Command Promt tại thư mục chứa ứng dụng để khởi động server:

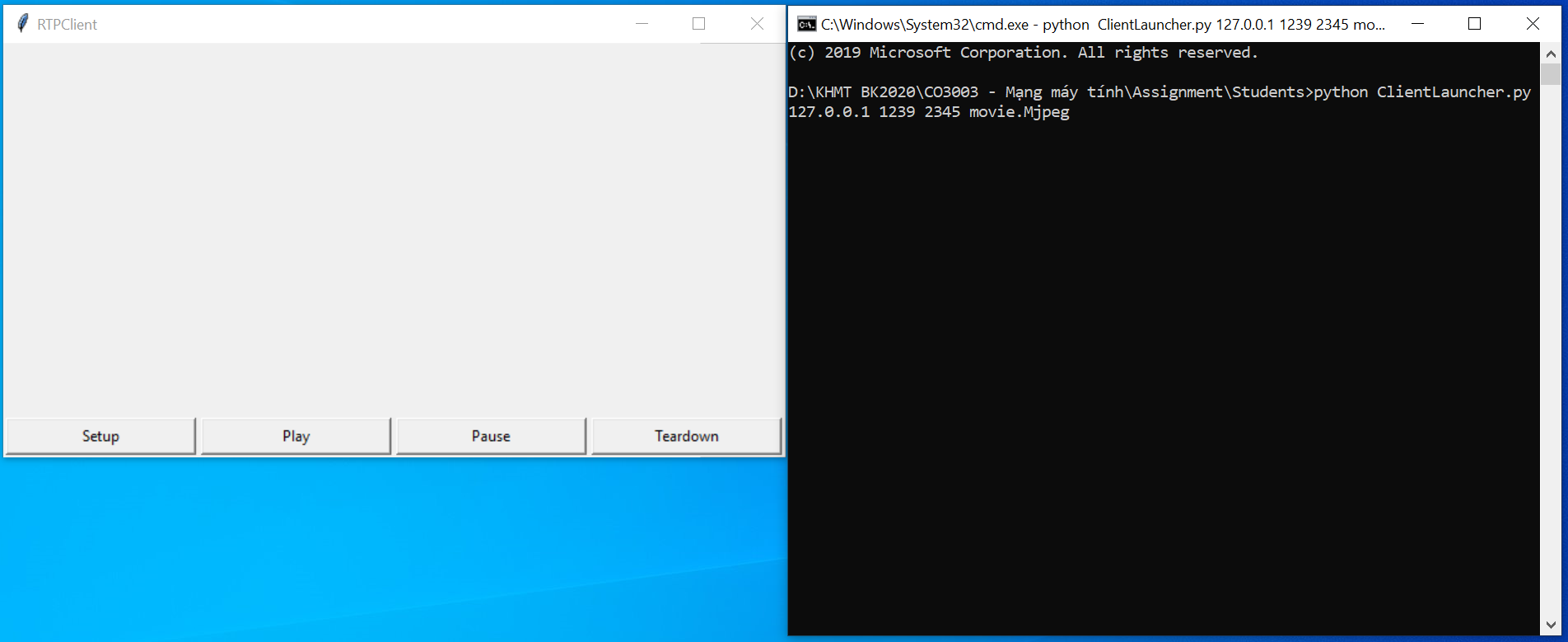
- Câu lệnh: python Server.py server\_port

- server\_port là cổng mà máy chủ lắng nghe các kết nối RTSP đến, trong Bài tập lớn này, chúng ta sử dụng Port có giá trị lớn hơn 1024, chúng ta cho server\_port giá trị là 1239.



1. Chạy ClientLaucher.py trên Command Promt tại thư mục chứa ứng dụng để khởi động Client và giao diện người dùng:

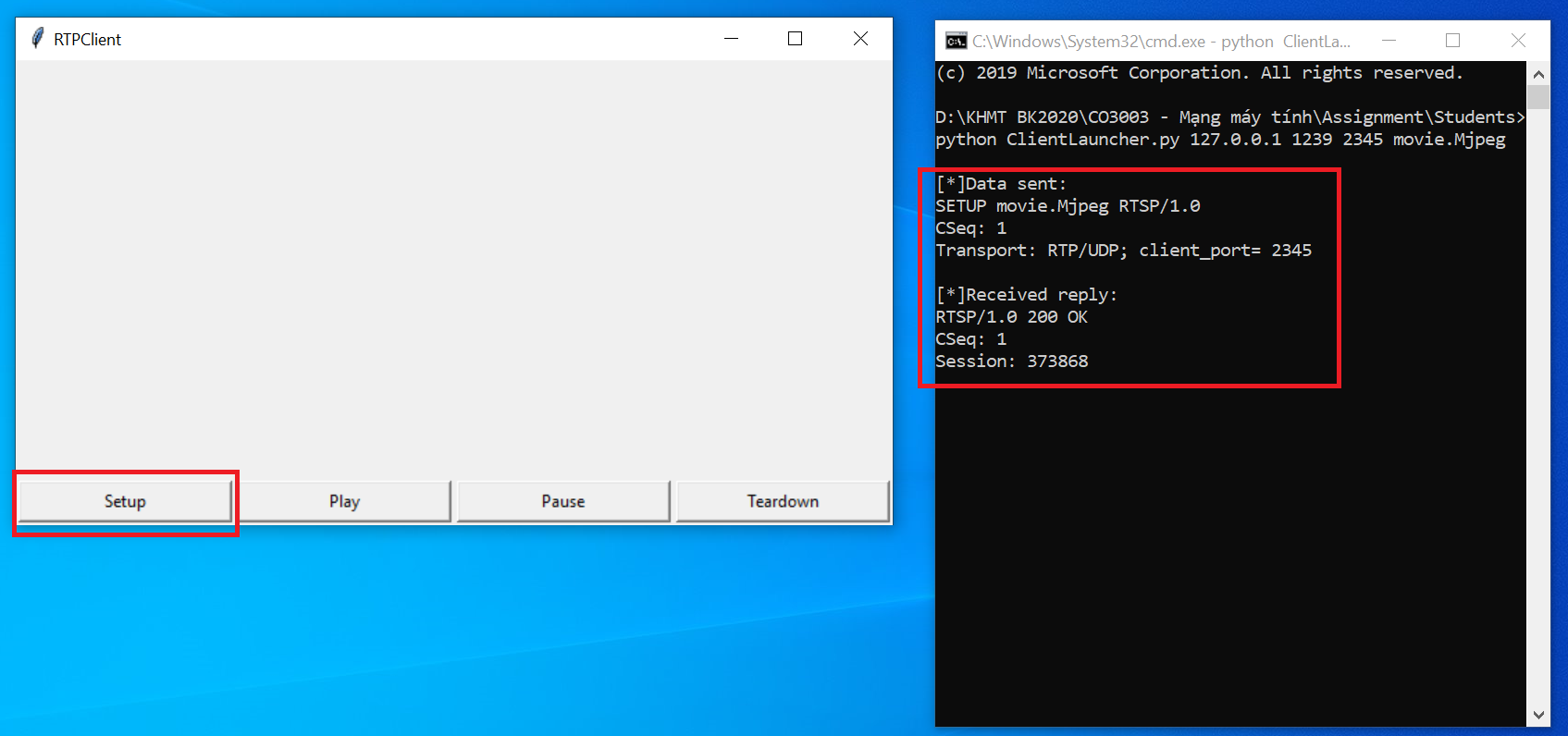
* Câu lệnh: python ClientLauncher.py server\_host server\_port PRT\_port video\_file
* server\_host là địa chỉ IP của thiết bị mà server đang chạy (chúng ta có thể sử dụng “127.0.0.1” ).
* server\_port là cổng mà máy chủ đang lắng nghe, ở đây là 1239.
* RTP\_port là cổng nơi các gói RTP được nhận , chúng ta cho RTP\_port có giá trị là 2345.
* video\_file là tên của tệp video mà chúng ta muốn phát (ở đây là “video.mjpeg”)



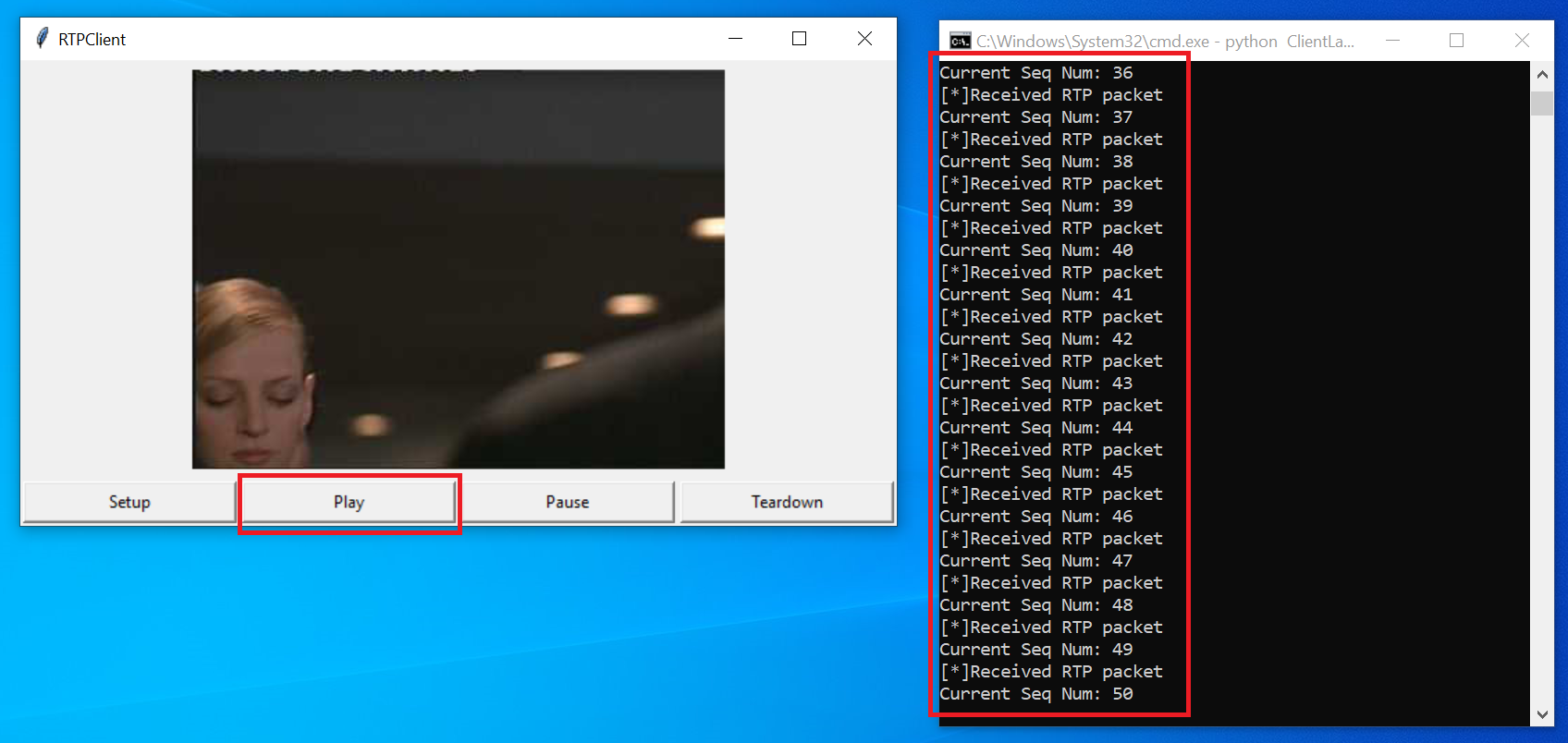
* Nếu kết nối không thành công sẽ có thông báo hiện lên. Nếu không có thông báo hiện lên nghĩa là đã kết nối thành công.

1. Thao tác trên giao diện:

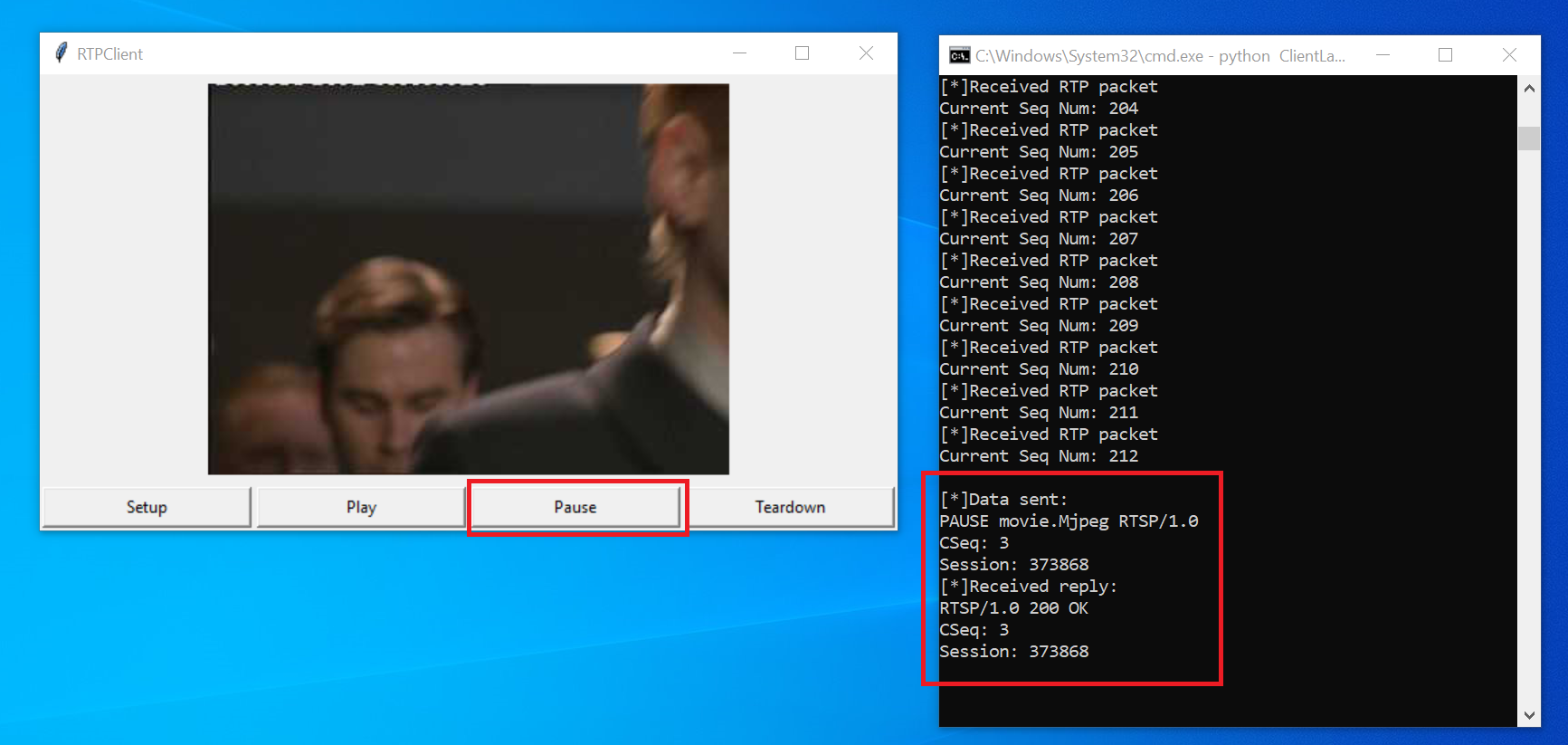
* Gửi yêu cầu “SETUP” từ Client đến Server bằng cách nhất vào nút “SETUP” trên giao diện.

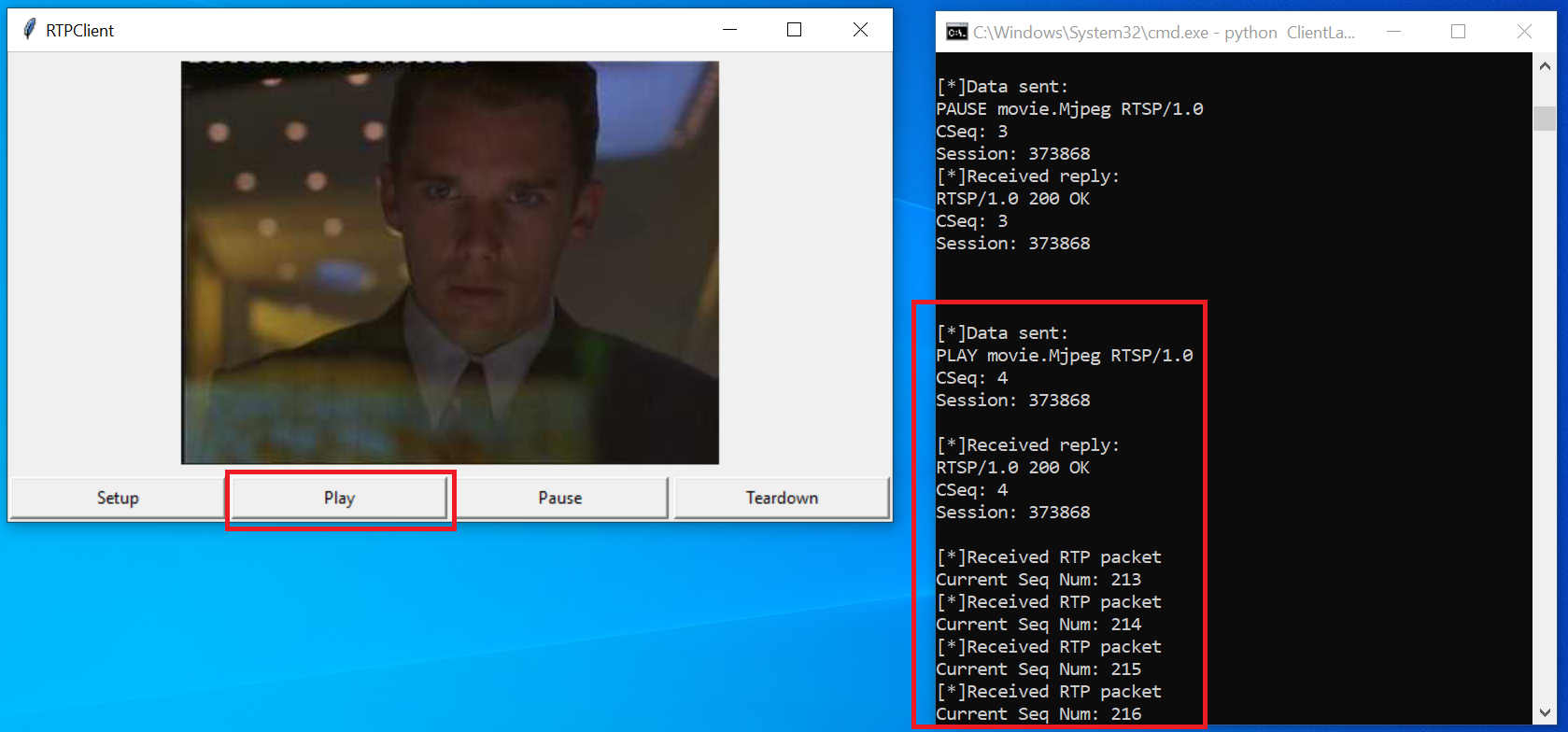


* Sau khi gửi yêu cầu “SETUP”, để phát video, nhấn vào nút “PLAY” để gửi yêu cầu đến Server.

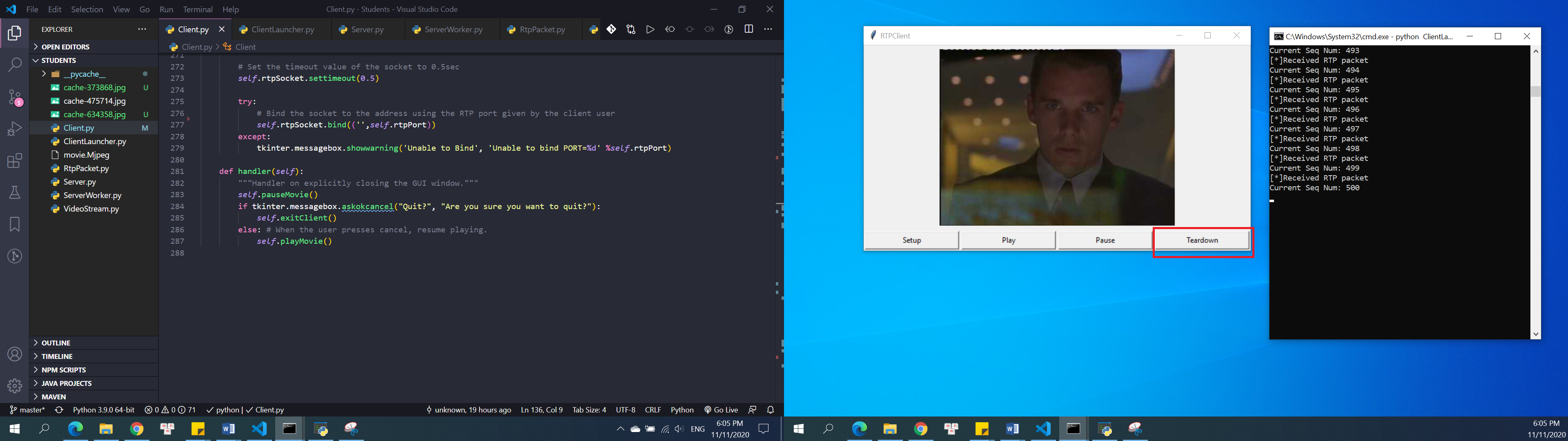


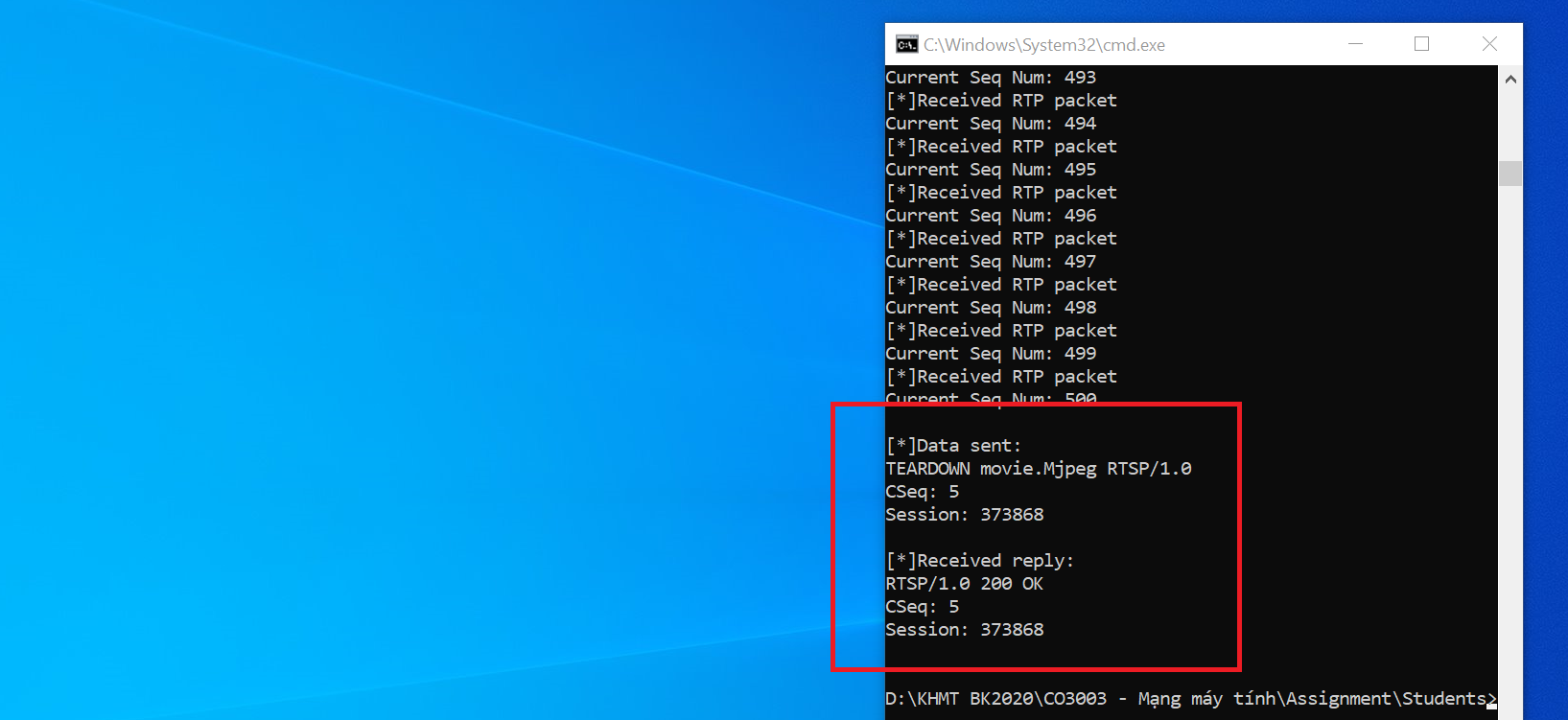
* Để tạm dừng khi video đang phát, nhấn vào nút “PAUSE” trên giao diện, nếu muốn tiếp tục phát video, nhấn lại vào nút “PLAY”.





* Để dừng phát video, ngắt kết nối với Server và tắt giao diện người dùng, nhấn vào nút “TEARDOWN” trên giao diện.





## **Đánh giá kết quả đạt được**

* Có được kiến thức tổng quát về hai giao thức RTSP (Real Time Streaming Protocol) và RTP (Real-time Transport Protocol):

+ RTSP: sử dụng cho hệ thống giải trí và truyền thông để kiểm soát các streaming media servers; thiết lập và kiểm soát các phiên truyền giữa các điểm cuối; sử dụng TCP.

+ RTP: giao thức để vận chuyển dữ liệu video và audio qua mạng IP; sử dụng UDP.

* Hiểu được cách thức tạo Socket để kết nối giữa Server và Client.
* Hiện thực được giao thức RTSP tại Client (Client.py) để thực hiện gửi các yêu cầu “SETUP”, “PLAY”, “PAUSE”, “TEARDOWN” đến Server thông qua các thao tác nhấn các nút tương ứng trên giao diện của người dùng. Các thông tin gửi đi theo đúng yêu cầu đề bài, và phản hồi từ Server cũng được đọc.
* Hiện thực được việc đóng gói gói tin RTP từ các frame dữ liệu của video (RtpPacket.py, phương thức encode()). Hiểu về các trường trong một header của một gói tin RTP.

## **Phần mở rộng (Extend)**

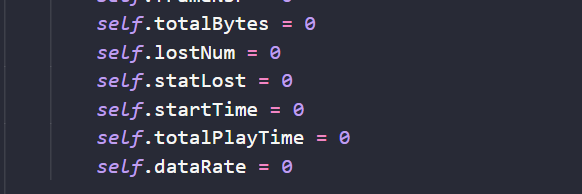
1. Tính RTP packet loss rate và Video data rate (bytes per second):

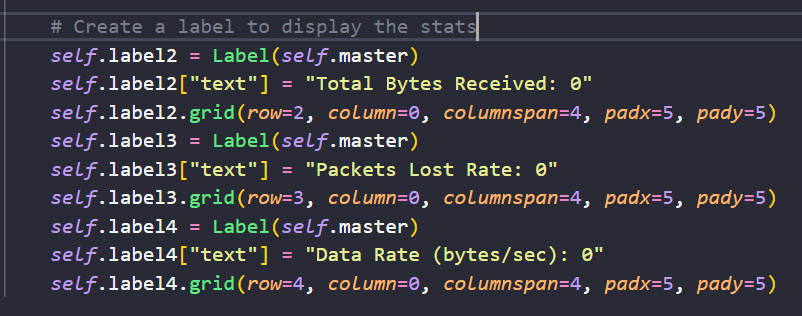
**Phân tích:**

* RTP packet loss rate: được định nghĩa là tỷ lệ của tổng số gói được truyền nhưng không đến máy nhận. Để tính tỷ lệ này ta sẽ đếm số lượng gói bị mất đi (Client không nhận được), sau đó chia cho số gói đã được gửi đi đến hiện tại.
* Video data rate (bytes per second): để tính thông số này, ta sẽ tính tổng số lượng bytes của các frame đã nhận được và thời gian từ lúc gửi yêu cầu “PLAY” đến khi nhận được gói tin RTP mới nhất.

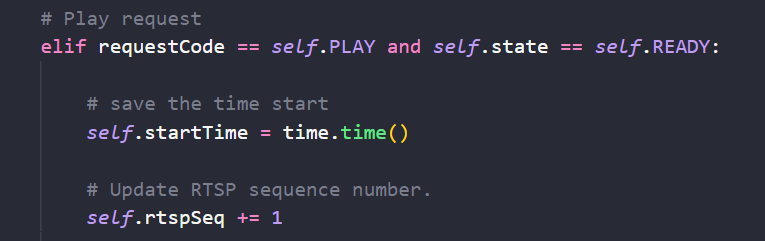
**Thực hiện:**

* Thêm các thuộc tính sau tại Client để tính toán các thông số trên. Thêm các Label vào giao diện để hiện các thông số Total Bytes, Packet loss rate, Data Rate.

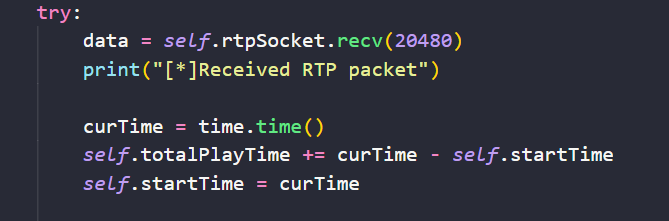




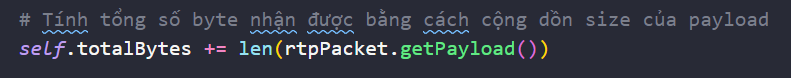
* Đầu tiên, khi thực hiện gửi yêu cầu “PLAY”, ta sẽ tiến hành lưu lại thời điểm bắt đầu:

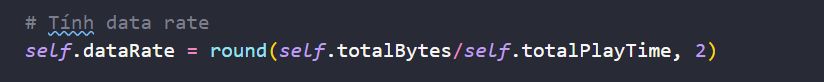


* Khi nhận được gói tin RTP từ Server, ta sẽ thực hiện ghi lại thời điểm nhận được gói tin, từ đó tính được tổng thời gian từ lúc yêu cầu “PLAY” đến thời điểm nhận được gói tin RTP theo cú pháp sau:

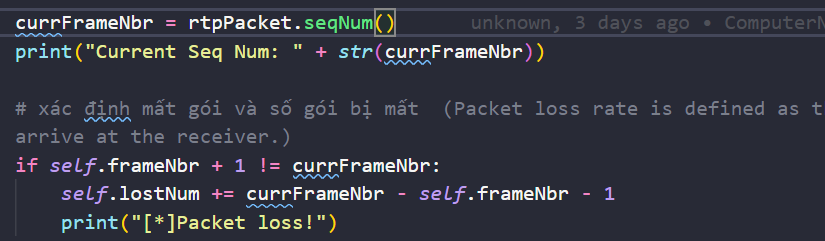


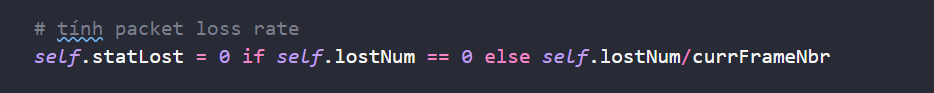
* Đồng thời, ta tính được tổng số byte nhận được bằng cách cộng dồn size của payload của mỗi gói tin RTP nhận được. Từ đó tính được thông số Data Rate bằng cách lấy tổng byte nhận được chia cho tổng thời gian từ lúc yêu cầu “PLAY”:





* Về Packet Loss, ta có thể xác định bằng cách so sánh số frame mà Client đang lưu với số frame của gói tin mà Server trả về, nếu số frame mà Client đang lưu cộng thêm 1 mà không bằng với số frame của gói tin mà Server trả về, có nghĩa đã có gói tin mà Client không nhận được. Từ đó ta tính được số gói tin Client không nhận được, đồng thời số gói tin RTP mà Server đã gửi đến thời điểm nhận gói tin mới nhất cũng chính là số frame của gói tin mà Server trả về. Do đó mà ta tính được thông số Packet Loss Rate:





* Sau đó ta thực hiện cập nhật lại các thông số trên lên giao diện:

