

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



## COMPUTER NETWORKING (CO3094)

---

### Assignment 1

# *"Video Streaming Application"*

---

Advisor: Lê Bảo Thịnh  
Group: 08  
Students: Hoàng Nhật Quân - 2014262.  
Nguyễn Hữu Hiếu - 2013149.  
Lê Bá Dũng - 2012863.

HO CHI MINH CITY, MAY 2023



## Contents

<b>1</b>	<b>Requirements Analysis</b>	<b>2</b>
1.1	Yêu cầu chức năng . . . . .	2
1.2	Yêu cầu phi chức năng . . . . .	2
<b>2</b>	<b>Function description</b>	<b>3</b>
2.1	Client.py . . . . .	3
2.2	ExtendClient.py . . . . .	4
2.3	RtpPacket.py . . . . .	7
2.4	Server.py . . . . .	8
2.5	ServerWorker.py . . . . .	8
2.6	VideoStream.py . . . . .	10
2.7	ClientLauncher.py . . . . .	10
<b>3</b>	<b>Class diagram</b>	<b>11</b>
3.1	Class Diagram for normal Client . . . . .	11
3.2	Class Diagram for extending Client . . . . .	12
<b>4</b>	<b>Results Achieved</b>	<b>13</b>
<b>5</b>	<b>User manual</b>	<b>14</b>
5.1	Khởi động ứng dụng . . . . .	14
5.2	Sử dụng ứng dụng . . . . .	15
<b>6</b>	<b>Extend Client Implementation</b>	<b>17</b>
6.1	How to run . . . . .	17
6.2	Idea for Buttons . . . . .	17
6.3	Implementation code for Extending Client . . . . .	19
6.3.1	Play Button . . . . .	19
6.3.2	Pause Button . . . . .	20
6.3.3	Reset Button . . . . .	21
6.3.4	Describe button . . . . .	22
6.4	Timing label and Forward, Backward button . . . . .	24
6.4.1	Timing label . . . . .	24
6.4.2	FORWARD and BACKWARD button . . . . .	25
<b>7</b>	<b>Member list - Group 06</b>	<b>28</b>



# 1 Requirements Analysis

## 1.1 Yêu cầu chức năng

- Triển khai một server video trực tuyến và client giao tiếp bằng giao thức RTP
- Client khởi động sẽ mở RTSP socket đến server để gửi yêu cầu cho server
- Trạng thái của client liên tục cập nhật khi nhận phản hồi từ server
- Có giao diện để Client thao tác ( Các nút bấm SETUP, PLAY, PAUSE, TEARDOWN, DESCRIBE)

## 1.2 Yêu cầu phi chức năng

- Thời gian chờ của datagram socket để nhận RTP data từ server là 0.5s.
- Server gửi gói RTP cho client bằng UDP mỗi 0,05s.

## 2 Function description

### 2.1 Client.py

#### Class variable

- INIT , READY, PLAYING lần lượt được gán cho giá trị 0, 1, 2.
- state - biến lưu trạng thái của Client, trong đó các giá trị được gán cho INIT, READY, PLAYING.
- SETUP, PAUSE, TEARDOWN lần lượt được gán cho giá trị 1, 2, 3.
- checkSocketIsOpen dùng để kiểm tra RtpSocket đã được mở chưa, giá trị ban đầu là false.
- checkPlay dùng để kiểm tra video đã đang chạy hay không, giá trị ban đầu là false.
- counter dùng để đếm số gói dữ liệu thất lạc, giá trị ban đầu = 0.

#### Phương thức Client.init (self, master, serveraddr, serverport, rtpport, filename):

Khởi tạo các instance variable, kết nối tới server, khởi tạo giao diện người dùng.

- Field master: đối tượng tk được truyền vào, thiết kế giao diện người dùng
- serverAddr, serverPort, rtpPort, fileName bốn tham số truyền vào khi người dùng gõ trong cmd.
- rtspSeg: dùng để đếm lần chuyển trạng thái khi người dùng thao tác.
- sessionId mã định danh riêng dành cho mỗi máy khách khi kết nối tới máy chủ.
- requestSent lưu giữ các giá trị nguyên ứng với từng thao tác trên nút bấm.
- teardownAcked: cờ đánh dấu kết thúc kết nối tới máy chủ và đóng socket.
- frameNbr: đếm số packet bị mất trong quá trình truyền thông tin giữa máy chủ và máy khác, ngoài ra còn giúp đồng bộ các frame khi truyền dữ liệu.
- Gọi phương thức ClientExtend.createWidgets(self).
- Gọi phương thức master.protocol("WM\_DELETE\_WINDOW", self.handler) dùng để xử lý sự kiện khi người dùng bấm vào dấu "X".
- Gọi phương thức connectToServer(self) để kết nối tới máy chủ.

#### Phương thức Client.createWidgets(self):

Khởi tạo giao diện người dùng

- Nút nhấn "Setup" dành cho phương thức Client.setupMovie().
- Nút nhấn "Play" dành cho phương thức Client.playMovie().
- Nút nhấn "Pause" dành cho phương thức Client.pauseMovie().
- Nút nhấn "Teardown" dành cho phương thức Client.resetMovie().

#### Phương thức Client.setupMovie(self):

Kiểm tra state là INIT thì client yêu cầu SETUP.

#### Phương thức Client.playMovie(self):

Kiểm tra giá trị của state nếu bằng READY thì client gửi yêu cầu PLAY.

#### Client.pauseMovie(self):

Kiểm tra nếu state bằng PLAYING thì client sẽ gửi yêu cầu PAUSE.

#### **Client.resetMovie(self):**

- Phương thức này đầu tiên sẽ kiểm tra cờ checkPlay. Nếu checkPlay được mở (có giá trị True) thì phương thức sẽ gọi phương thức sendRtspRequest() với tham số là self.SETUP để tiến hành việc reset chuyển state của Client về INIT và tiến hành xóa hết tất cả các cache được hệ thống lưu trước đó. Sau đó, sẽ tiến hành reset lại những thông số mặc định và tiến hành kết nối lại với server, mở cổng rtp.
- Nếu cờ checkPlay không được mở (giá trị False) thì phương thức kết thúc.

#### **Phương thức Client.writeframe(self, data)**

Đọc dữ liệu từ frame chuyển thành tệp thuộc lớp Image.

#### **Phương thức Client.updateMovie(self, imageFile)**

Load các hình ảnh được tạo lên giao diện người dùng

#### **Phương thức Client.connectToServer(self)**

Kết nối tới máy chủ

#### **Phương thức Client.sendRtspRequest(self)**

Gửi các yêu cầu RTSP đến server.

#### **Phương thức Client.recvRtspReply(self)**

Nhận các phản hồi RTSP từ server đây cũng sẽ là một vòng lặp được xử lý bởi một luồng riêng.

#### **Phương thức Client.parseRtspReply(self, data)**

Kiểm tra các phản hồi và thực thi trong client.

#### **Phương thức Client.openRtspPort(self)**

Phương thức này đầu tiên sẽ gán True cho checkSocketIsOpen và tạo ra trường rtpSocket, gán với một đối tượng socket có internet protocol là IPv4 có giao thức UDP và time out là 0.5s. Sau đó socket sẽ kết nối đến IP của Client và port của Client. Nếu không thành công thì sẽ hiện lên một biểu mẫu thông báo kết nối thất bại cho Client.

#### **Phương thức Client.parseRtspReply(self, data)**

Kiểm tra các phản hồi và thực thi trong client tương ứng các phản hồi SETUP, PLAY, PAUSE, TEARDOWN .

#### **Phương thức Client.handler(self)**

Xử lý khi người dùng nhấn nút "X" để thoát giao diện người dùng.

## **2.2 ExtendClient.py**

### **Class variable**

- INIT , READY, PLAYING lần lượt được gán cho giá trị 0, 1, 2.
- state - biến lưu trạng thái của Client, trong đó các giá trị được gán cho INIT, READY, PLAYING.
- SETUP, PAUSE, TEARDOWN lần lượt được gán cho giá trị 1, 2, 3.
- checkSocketIsOpen dùng để kiểm tra RtpSocket đã được mở chưa, giá trị ban đầu là false.
- checkPlay dùng để kiểm tra video đã đang chạy hay không, giá trị ban đầu là false.

- counter dùng để đếm số gói dữ liệu thất lạc, giá trị ban đầu = 0.
- isFirstPlay (có giá trị khởi tạo là True) là cờ dùng để setup video trong lần chạy đầu tiên.

**Phương thức ClientExtend.init (self, master, serveraddr, serverport, rtpport, filename):**

Khởi tạo các instance variable, kết nối tới server, khởi tạo giao diện người dùng.

- Field master: đối tượng tk được truyền vào, thiết kế giao diện người dùng
- serverAddr, serverPort, rtpPort, fileName bốn tham số truyền vào khi người dùng gõ trong cmd.
- rtspSeg: dùng để đếm lần chuyển trạng thái khi người dùng thao tác.
- sessionId mã định danh riêng dành cho mỗi máy khách khi kết nối tới máy chủ.
- requestSent lưu giữ các giá trị nguyên ứng với từng thao tác trên nút bấm.
- teardownAcked: cờ đánh dấu kết thúc kết nối tới máy chủ và đóng socket.
- frameNbr: đếm số packet bị mất trong quá trình truyền thông tin giữa máy chủ và máy khác, ngoài ra còn giúp đồng bộ các frame khi truyền dữ liệu.
- Gọi phương thức ClientExtend.createWidgets(self).
- Gọi phương thức master.protocol("WM\_DELETE\_WINDOW", self.handler) dùng để xử lý sự kiện khi người dùng bấm vào dấu "X".
- Gọi phương thức connectToServer(self) để kết nối tới máy chủ.
- currentTime (giá trị khởi tạo = 0): dùng để hỗ trợ việc hiển thị Remaining time label trong GUI.
- totalTime dùng để hiển thị tổng số thời gian để video chạy.
- isForward (giá trị khởi tạo = 0): là cờ dùng để kiểm tra sự kiện người dùng tua video tới.
- isBackward (giá trị khởi tạo = 0): là cờ dùng để kiểm tra sự kiện người dùng tua video lùi
- countTotalPacket, timerBegin, timerEnd, timer, bytes, packets, packetsLost, lastSequence, totalJitter, arrivalTimeofPreviousPacket, lastPacketSpacing được dùng để tiến hành thống kê dữ liệu đường truyền và được khởi tạo giá trị mặc định bằng 0.

**Phương thức ClientExtend.createWidgets(self):**

Khởi tạo giao diện người dùng

- Nút nhấn "Play" dành cho phương thức ClientExtend.playMovie().
- Nút nhấn "Pause" dành cho phương thức ClientExtend.pauseMovie().
- Nút nhấn "Stop" dành cho phương thức ClientExtend.resetMovie().
- Nút nhấn "Describe" dành cho phương thức ClientExtend.describeMovie().
- Nút nhấn "«" dành cho phương thức ClientExtend.prevMovie()
- Nút nhấn "»" dành cho phương thức ClientExtend.forwardMovies().
- Label "Total Time", hiển thị tổng thời gian của video.
- Label "Time left", hiện thị thời gian còn lại của video khi đang chạy.

**Phương thức ClientExtend.setupMovie(self):**

Kiểm tra state là INIT thì client yêu cầu SETUP.

**Phương thức ClientExtend.playMovie(self):**

- Kiểm tra giá trị của state nếu bằng READY thì client gửi yêu cầu PLAY.
- Enable 2 nút Backward («) và Forward (»).

**ClientExtend.pauseMovie(self):** Kiểm tra nếu state bằng PLAYING thì client sẽ gửi yêu cầu PAUSE.

**ClientExtend.resetMovie(self):**

- Phương thức này đầu tiên sẽ kiểm tra cờ checkPlay. Nếu checkPlay được mở (có giá trị True) thì phương thức sẽ gọi phương thức sendRtspRequest() với tham số là self.SETUP để tiến hành việc reset chuyển state của Client về INIT và tiến hành xóa hết tất cả các cache được hệ thống lưu trước đó. Sau đó, sẽ tiến hành reset lại những thông số mặc định và tiến hành kết nối lại với server, mở cổng rtp.
- Nếu cờ checkPlay không được mở (giá trị False) thì phương thức kết thúc.

**Phương thức ClientExtend.forwardMovies(self)**

Gọi phương thức sendRtspRequest(self, requestCode) với requestCode nhận giá trị FORWARD và sau đó set cờ isForward lên 1.

Phương thức ClientExtend.prevMovie(self): phương thức này sẽ gọi phương thức sendRtspRequest(self, requestCode) với requestCode nhận giá trị BACKWARD, set cờ isBackward lên 1 và sau đó tiến hành chỉnh sửa lại field frameNbr

**Phương thức Client2.listenRtp(self)**

Lỗi tạo đối tượng thuộc lớp RtpPacket, dữ liệu đọc sẽ được lưu bởi bộ đệm kích thước 20 bytes = 20480 bit, phương thức là một vòng lặp liên tục lặp đi lặp lại để lắng nghe dữ liệu, do đó nó sẽ được gọi và thực hiện ở một luồng (thread) riêng.

**Phương thức ClientExtend.writeframe(self, data)**

Đọc dữ liệu từ frame chuyển thành tệp thuộc lớp Image.

**Phương thức Client.updateMovie(self, imageFile)**

Load các hình ảnh được tạo lên giao diện người dùng

**Phương thức ClientExtend.connectToServer(self)**

Kết nối tới máy chủ

**Phương thức ClientExtend.sendRtspRequest(self)**

Gửi các yêu cầu RTSP đến server.

**Phương thức ClientExtend.recvRtspReply(self)**

Nhận các phản hồi RTSP từ server đây cũng sẽ là một vòng lặp được xử lý bởi một luồng riêng.

**Phương thức ClientExtend.parseRtspReply(self, data)**

Kiểm tra các phản hồi và thực thi trong client.

#### **Phương thức ClientExtend.openRtspPort(self)**

Phương thức này đầu tiên sẽ gán True cho checkSocketIsOpen và tạo ra trường rtpSocket, gán với một đối tượng socket có internet protocol là IPv4 có giao thức UDP và time out là 0.5s. Sau đó socket sẽ kết nối đến IP của Client và port của Client. Nếu không thành công thì sẽ hiện lên một biểu mẫu thông báo kết nối thất bại cho Client.

#### **Phương thức Client.parseRtspReply(self, data)**

Kiểm tra các phản hồi và thực thi trong client tương ứng các phản hồi SETUP, PLAY, PAUSE, TEARDOWN .

#### **Phương thức Client.handler(self)**

Xử lý khi người dùng nhấn nút "X" để thoát giao diện người dùng.

#### **Phương thức ClientExtend.displayDescription(self, lines)**

Hiện ra một cửa sổ gồm những thông tin về stream cũng như những phương thức encoding đang sử dụng

#### **Phương thức Client2.displayStatus(self)**

Phương thức này sẽ hiện ra một cửa sổ gồm những thông số thống kê trong quá trình Streaming video từ server tới phía Client.

## **2.3 RtpPacket.py**

#### **Global Variable**

HEADER\_SIZE (giá trị 12) : quy định số lượng phần tử của mảng header trong class RtpPacket.

#### **Class Variable**

header: mảng 12 byte chứa siêu dữ liệu (thông tin của 1 gói RTP)

#### **Phương thức RtpPacket.encode(self, version, padding, extension, cc, seqnum, marker, pt, ssrc, payload)**

Mã hóa các gói RTP:

- version (2 bits): Phiên bản của giao thức.
- padding (1 bit): Tín hiệu báo trong gói sẽ có 1 (0) hoặc nhiều padding trong gói (1), thường dùng để lấp các khoảng trống trong khối dữ liệu.
- extension (1 bit) : Tín hiệu báo có header mở rộng ( 1 - có; 0 - không ).
- cc (CSRC count) (4 bits): Chứa các giá trị của trường CSRC ID trong header cố định.
- market (1 bit): Dùng ở lớp ứng dụng (Application layer), xác định một profile.
- pt (payload type) (7 bits): Xác định và nêu ý nghĩa các dạng payload của RTP.
- seqnum (16 bits) : Mang số thứ tự của gói RTP.
- timestamp (32 bits): Xác định thời điểm lấy mẫu của byte đầu tiên trong gói.
- ssrc (synchronization source identifier) (32 bits): Chỉ ra nguồn đồng bộ (nguồn phát gói tin RTP từ micro, camera hay bộ ngẫu phối RTP) của gói tin.

#### **Phương thức RtpPacket.decode(self, byteStream)**

Giải mã header và payload của gói RTP.



**Phương thức RtpPacket.version(self)**

Trả về phiên bản của giao thức

**Phương thức RtpPacket.seqNum(self)**

Trả về số thứ tự của gói tin.

**Phương thức RtpPacket.timestamp(self)**

Trả về mốc thời gian của gói tin.

**Phương thức RtpPacket.payloadType(self)**

Trả về định dạng của gói tin.

**Phương thức RtpPacket.getPayload(self)**

Trả về nội dung của gói tin.

**Phương thức RtpPacket.getPacket(self)**

Trả về header và payload của gói tin.

## 2.4 Server.py

**Phương thức Server.main(self)**

- Phương thức tạo 1 biến SERVER\_PORT là port của máy chủ được truyền vào từ cmd
- Phương thức sẽ khởi tạo 1 socket và gán socket này với port vừa truyền vào.
- Sau đó, liên tục nhận thông điệp từ các nguồn khác nhau (tối đa 5)

## 2.5 ServerWorker.py

**Class variables**

- SETUP : mã xử lý request SETUP từ client.
- PAUSE : mã xử lý request PAUSE từ client.
- PLAY : mã xử lý request PLAY từ client.
- TEARDOWN : mã xử lý request TEARDOWN từ client.
- FORWARD : mã xử lý request FORWARD từ client.
- BACKWARD : mã xử lý request BACKWARD từ client.
- PREV : mã xử lý request PREV từ client.
- INIT ( giá trị = 0): trạng thái khởi tạo.
- READY ( giá trị = 1): trạng thái sẵn sàng.
- PLAYING ( giá trị = 2): trạng thái đang chạy.
- state: biến lưu trạng thái của ServerWorker.
- OK\_200: (giá trị = 0) : mã dùng để hiển thị giá trị OK gửi về từ phía Server tới Client.
- FILE\_NOT\_FOUND\_404 (giá trị = 1) : mã dùng để hiển thị giá trị không tìm thấy gửi về từ phía Server tới Client.
- CON\_ERR\_500 (giá trị = 2) : mã dùng để hiển thị giá trị không thể xử lý từ phía Server tới Client.

- clientInfo: chứa các thông tin từ phía client.

#### **Phương thức ServerWorker.init(self, clientInfo)**

- Gán cho clientInfo đối tượng clientInfo nhận vào.
- Khởi tạo cờ opt với giá trị 0 dùng để xử lý trường hợp client yêu cầu PLAY, FORWARD, BACKWARD.

#### **Phương thức ServerWorker.run(self)**

Tạo ra một Thread mới chạy song song với thread chính để quản lý các request RTP/RTSP đối với mỗi máy khách.

#### **Phương thức ServerWorker.recvRtspRequest(self)**

Mở socket để truyền/nhận dữ liệu và mã hóa/giải mã trước khi gửi đi.

#### **Phương thức ServerWorker.processRtspRequest(self, data)** Xử lý thao tác khác nhau đối với file được chọn dựa theo loại yêu cầu.

- Yêu cầu SETUP: tiến hành thiết lập video mà tại client mong muốn xem, trong trường hợp không thể mở được video thì sẽ gọi phương thức replyRtsp(self, code, seq) với code = FILE\_NOT\_FOUND\_404. Sau khi đã tiến hành thiết lập video thì sẽ tiến hành thiết lập sessionId và gọi phương thức replyRtsp(self, code, seq) với code = OK\_200.
- Yêu cầu PLAY: tiến hành tạo socket (IPV4 và giao thức UDP), sau đó gọi giao thức replyRtsp(self, code, seq) với code = OK\_200 và tạo một thread chạy song song để gửi các gói tin Rtp, tạo ra cờ clientInfo['event'] (giá trị khởi tạo bằng 0) để có thể kiểm soát quá trình gửi gói tin Rtp tới Client.
- Yêu cầu PAUSE: gán giá trị 1 cho clientInfo['event'] tiến hành hủy việc gửi gói tin Rtp tới Client và gọi phương thức replyRtsp(self, code, seq) với mã code = OK\_200.
- Yêu cầu TEARDOWN: gán giá trị 1 cho clientInfo['event'] tiến hành hủy việc gửi gói tin Rtp tới Client và gọi phương thức replyRtsp(self, code, seq) với mã code = OK\_200. Sau đó đóng socket.
- Yêu cầu DESCRIBE: gọi phương thức replyDescribe(self, code, seq, filename).
- Yêu cầu BACKWARD: set cờ opt lên 1
- YÊU cầu FORWARD: gọi phương thức forwardStream(self)

#### **Phương thức ServerWorker.forwardStream(self)**

Gán 1 cho cờ isNext của clientInfo['videoStream'] để xử lý sự kiện client gửi yêu cầu FORWARD.

#### **Phương thức ServerWorker.makeRtp(self, payload, frameNbr)**

Nén và chuyển đổi định dạng dữ liệu nhận từ biến data trong hàm sendRtp, các thông số đi kèm khác thành các gói tin.

#### **Phương thức ServerWorker.replyRtsp(self, code, seq)**

Gửi trả gói tin đến máy khác dựa vào mã code được truyền vào hoặc báo lỗi nếu kết nối không thành công hoặc không tìm thấy file.

#### **Phương thức ServerWorker.replyDescribe(self, code, seq, filename)**

Kiểm tra tham số code nhận vào. Nếu code = OK\_200 sẽ tiến hành gửi các thông tin về DESCRIBE mà Client yêu cầu qua socket tới client.

## 2.6 VideoStream.py

### Phương thức VideoStream.init(self, filename)

- Tham số filename là tên tập tin cần stream.
- Khởi tạo frameNum ( giá trị khởi tạo là 0) để lưu số thứ tự của khung hình hiện tại

### Phương thức VideoStream.nextFrame(self)

Đọc frameLength, nếu nhận được dữ liệu, đặt giá trị frameNum = frameNum + 1.

### Phương thức VideoStream.frameNbr(self)

Trả về số thứ tự của frame vừa lấy.

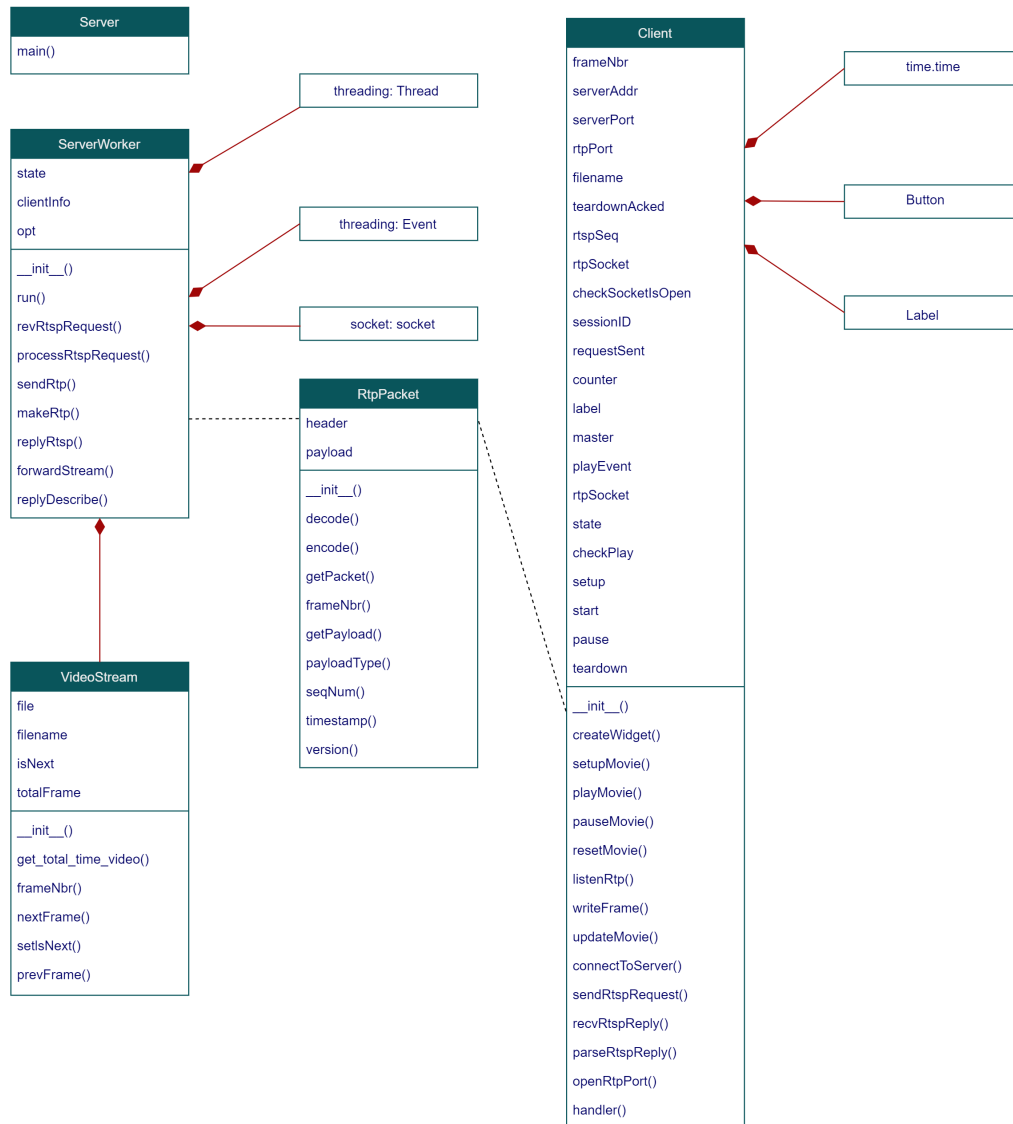
## 2.7 ClientLauncher.py

### Phương thức ClientLauncher.main()

- Nhận 4 tham số từ người dùng trong cmd:
  - + Địa chỉ IP của máy chủ
  - + Mã port với máy chủ của "máy khách" (lớp Client
  - + Mã port các gói dữ liệu qua giao thức Rtp (Real-time Protocol)
  - + Tên của tập tin cần đọc.
- Khởi tạo đối tượng tk (điều khiển, thiết kế giao diện winform) vào lớp Client được khởi tạo và tạo vòng lặp vô tận cho Winform.

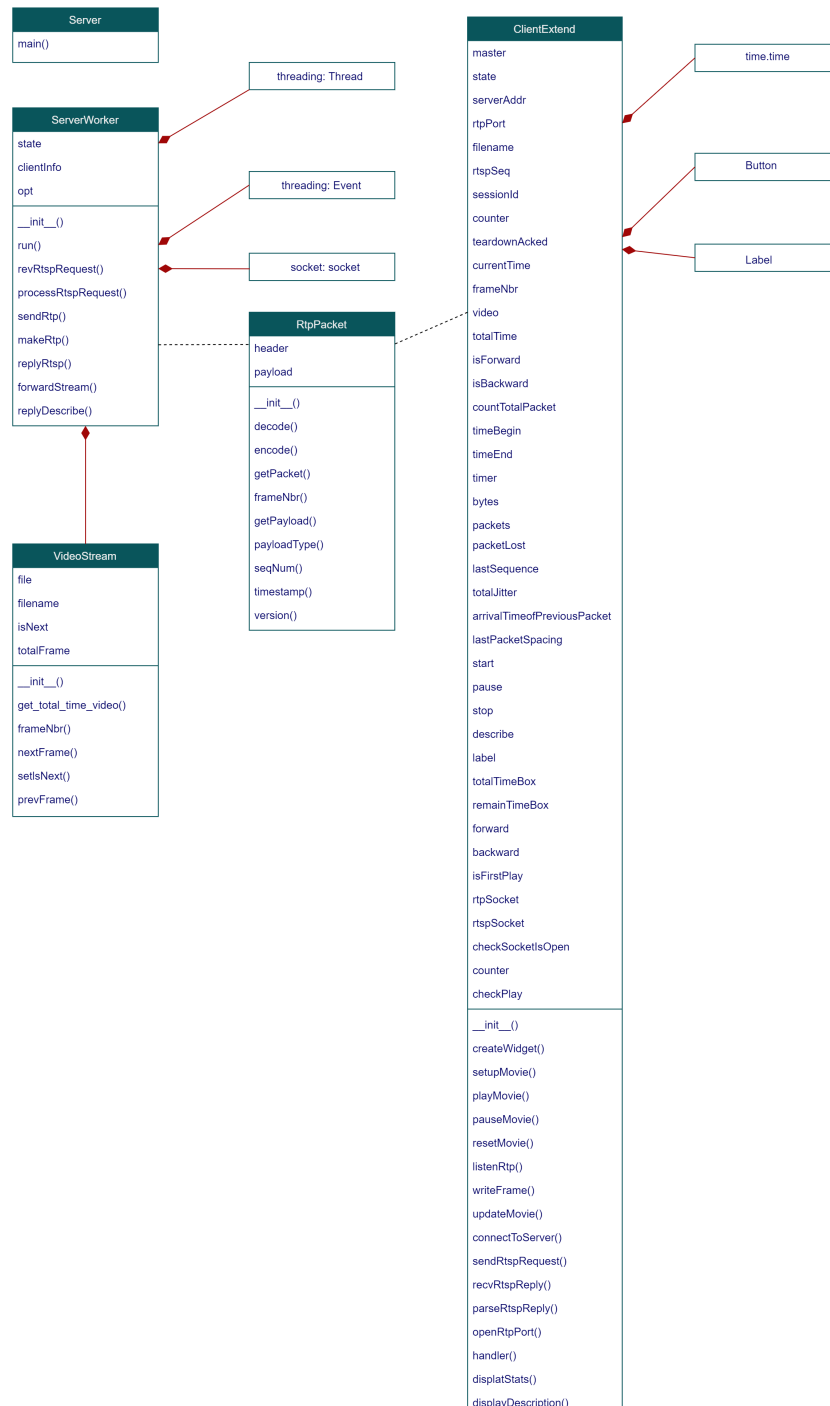
### 3 Class diagram

#### 3.1 Class Diagram for normal Client



Hình 1: Diagram for normal Client

### 3.2 Class Diagram for extending Client

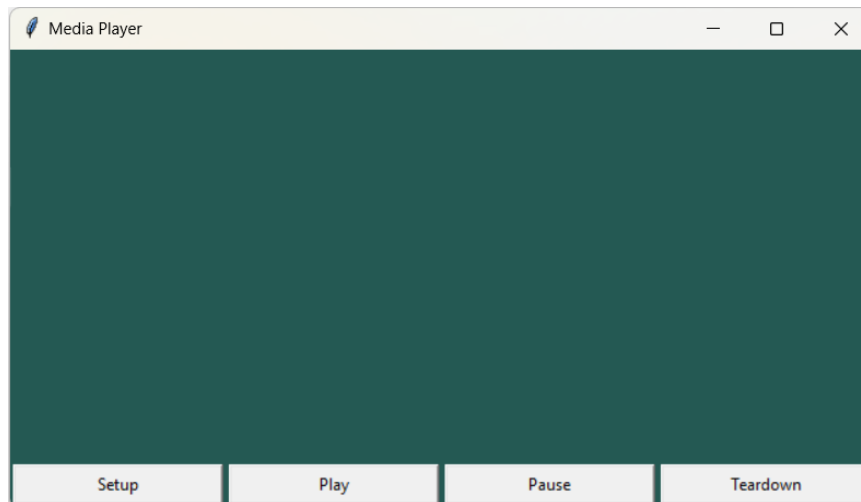


Hình 2: Diagram for extending Client

## 4 Results Achieved

Thông qua bài tập lớn "Video Streaming Application", nhóm đã có thể.

- Thiết kế được GUI cho người dùng xem được video thông qua thư viện tkinter của Python
- Đối với normal Client: Giao diện gồm 4 nút cơ bản "Setup", "Play", "Pause", "Teardown" theo yêu cầu của bài.



Hình 3: Giao diện người dùng Normal Client

- Đối với extending Client: Giao diện gồm 4 nút tương tác: "Setup", "Play", "Pause", "Teardown", "Forward", "Backward" và 2 label "Total time", "Remaining time" lần lượt để thể hiện tổng thời lượng của video và thời lượng còn lại của video.

## 5 User manual

### 5.1 Khởi động ứng dụng

Bước 1. Chạy 2 Terminal - 1 dành cho server 1 dành cho Client.

Bước 2. Chọn môi trường chạy chương trình tại folder chứa source code bằng lệnh  
`cd yourAddr/.../Assignment-1/Source.`

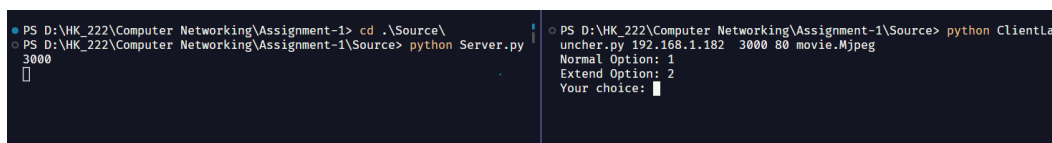
**Note:** Có thể mở trực tiếp 2 terminal từ source code

Bước 3. Terminal thứ nhất (server side) chạy lệnh:  
`python Serve.py <server-port>`. Trong đó:

- server-port là port mà server nhận các kết nối RTSP đến.
- Port RTSP tiêu chuẩn là 554, nhưng ta phải chọn port lớn hơn 1024.  
Trong bài nhóm chọn port 3000. Câu lệnh sẽ là:  
`python Server.py 3000`

Bước 4. Terminal thứ hai (Client side) chạy lệnh:  
`python ClientLauncher.py <server-host> <server-port> <RTP-port> <video-file>`.  
Trong đó:

- server-host: Địa chỉ IP của server (ở đây là địa chỉ IP của thiết bị đang sử dụng).
- server-port: port server đang nghe ( ví dụ ở bên terminal thứ nhất dùng port 3000 thì ở đây server-port cũng là 3000)
- RTP-port: port nơi nhận các gói RTP (Có thể chọn một số nguyên bất kỳ - trong bài này sử dụng port 80)
- video-file: Tên của tệp video bạn muốn yêu cầu stream (trong bài tập lớn này là movie.Mjpeg)



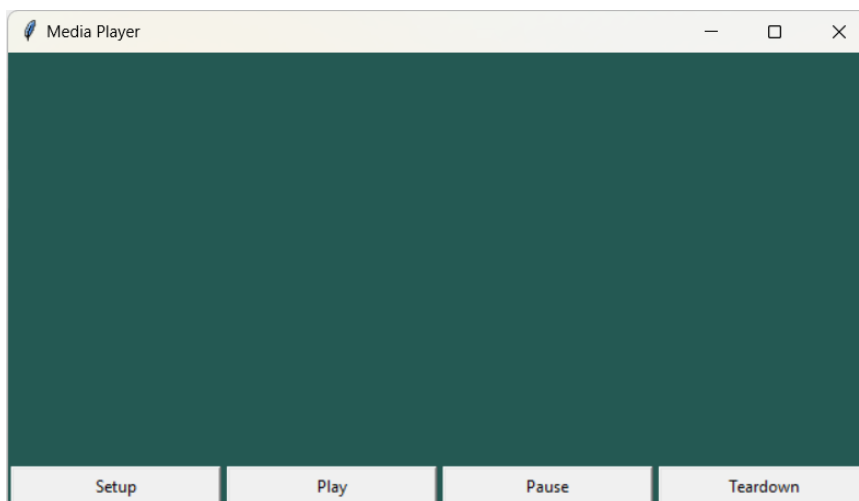
```
PS D:\HK_222\Computer Networking\Assignment-1> cd .\Source\
PS D:\HK_222\Computer Networking\Assignment-1\Source> python Server.py
3000
[ ]

PS D:\HK_222\Computer Networking\Assignment-1\Source> python ClientLa
uncher.py 192.168.1.182 3000 80 movie.Mjpeg
Normal Option: 1
Extend Option: 2
Your choice: [ ]
```

Hình 4: Server side (left) - Client Side (right)

## 5.2 Sử dụng ứng dụng

Sau khi chạy các bước ở phần 5.1. Cửa sổ sẽ hiện lên giao diện như hình 5:



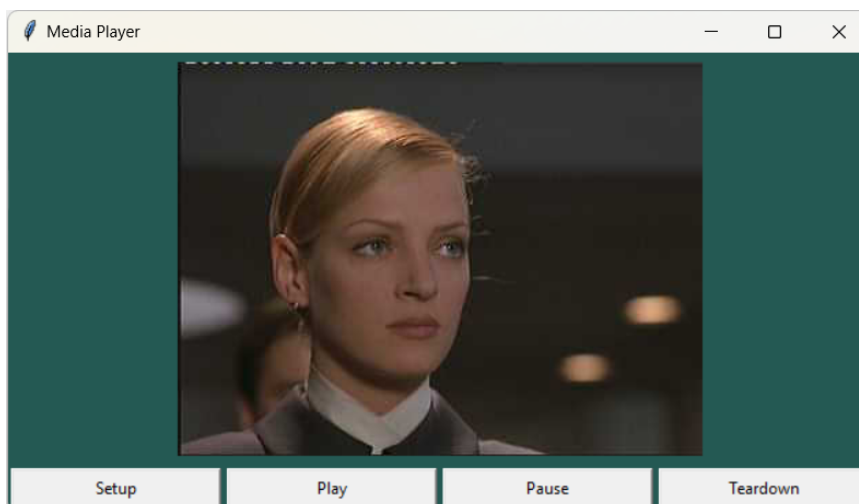
Hình 5: Giao diện ban đầu của người dùng

### Setup Button

Nhấn vào nút Setup để gửi yêu cầu SETUP tới server.

### Play Button

Nhấn vào nút Play để chạy chương trình.

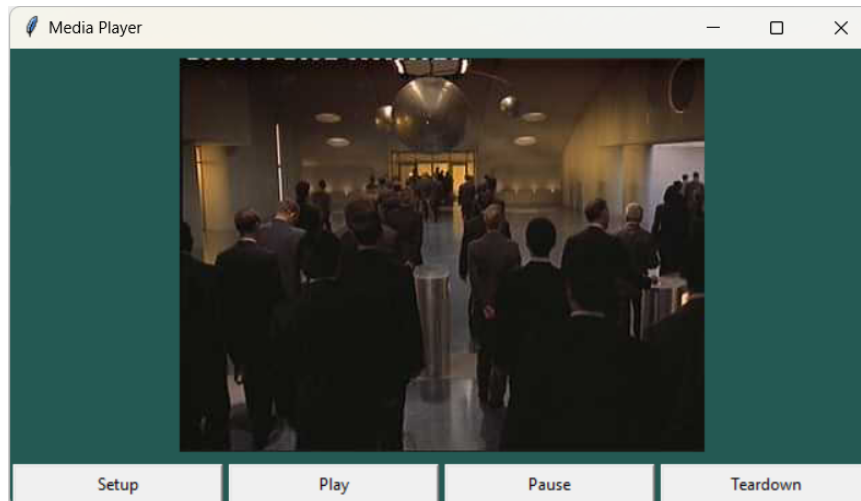


Hình 6: Giao diện người dùng khi nhấn nút Play



### Pause Button

Nhấn nút Pause để tạm dừng stream video.



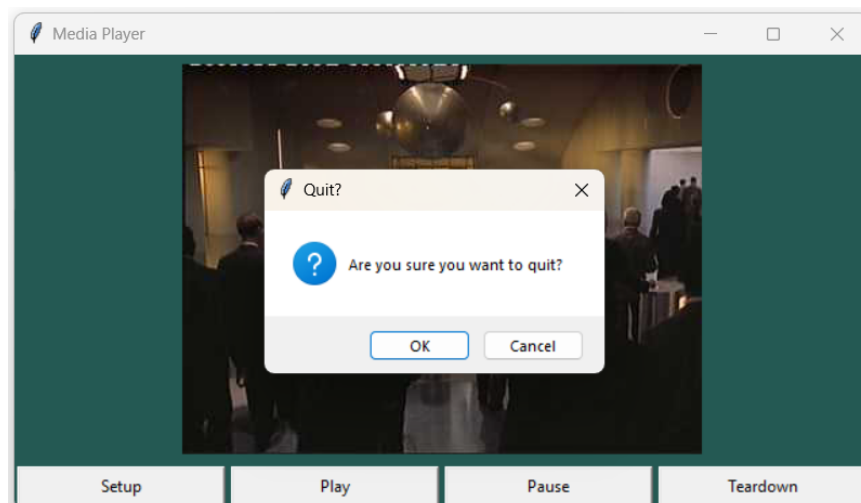
Hình 7: Giao diện người dùng khi nhấn nút Pause

### Teardown Button

Nhấn vào nút Teardown, ứng dụng sẽ dừng stream video và quay về trạng thái ban đầu như chưa được SETUP.

### Thoát ứng dụng

Để thoát ứng dụng và đóng cửa sổ giao diện, nhấn vào nút X ở góc trên bên phải màn hình. Nhấn OK để thoát, Nhấn Cancel để hủy yêu cầu.



Hình 8: Giao diện người dùng khi muốn thoát ứng dụng

## 6 Extend Client Implementation

### 6.1 How to run

Để chạy ứng dụng ở phần mở rộng (Extend):

Ở **Bước 4**: Thay vì nhập "1" thì người dùng nhập "2" từ bàn phím.

```
PS D:\HK_222\Computer Networking\Assignment-1\Source> python Server.py
3000
□

PS D:\HK_222\Computer Networking\Assignment-1\Source> python ClientLa
uncher.py 192.168.1.132 3000 80 movie.Mjpeg
Normal Option: 1
Extend Option: 2
Your choice: 2
```

Hình 9: Server side (left) - Client Side (right)

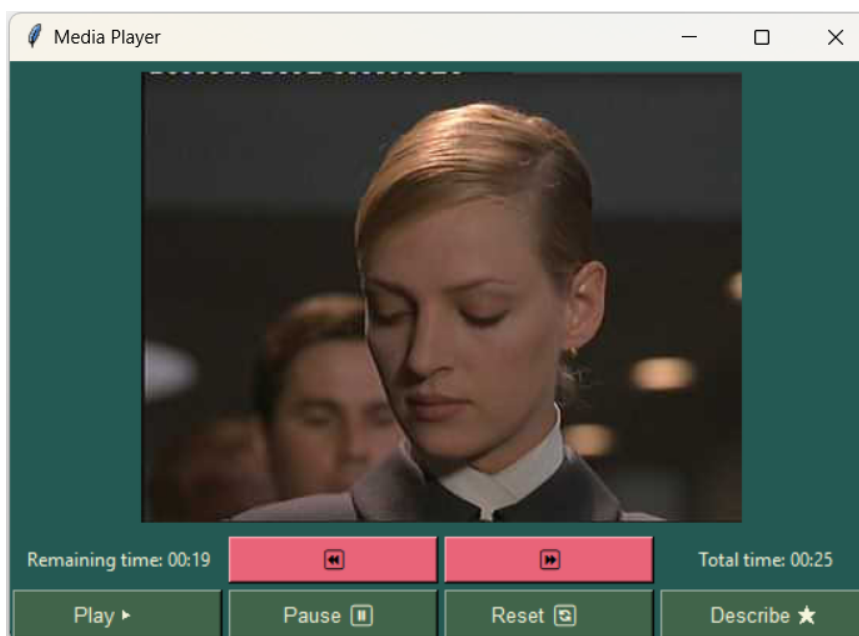
### 6.2 Idea for Buttons

**Play:** Tích hợp SETUP

SETUP là bắt buộc trong tương tác RTSP. Ở normal flow, người dùng cần bấm vào nút SETUP để thực hiện kết nối giữa client và server.

Tuy nhiên, trên thực tế, các ứng dụng stream video chỉ dùng nút Play, set up sẽ được tự tích hợp vào nút Play, hoặc ngay khi người dùng sử dụng ứng dụng (ngay sau khi enter gõ lệnh ở terminal client).

Trong phần mở rộng, nhóm tích hợp SETUP ở lần bấm PLAY đầu tiên, hoặc bấm PLAY sau khi bấm STOP. Hay đơn giản, khi người dùng bấm nút PLAY khi ở trạng thái chưa SETUP, ứng dụng sẽ tự động SETUP cho người dùng.

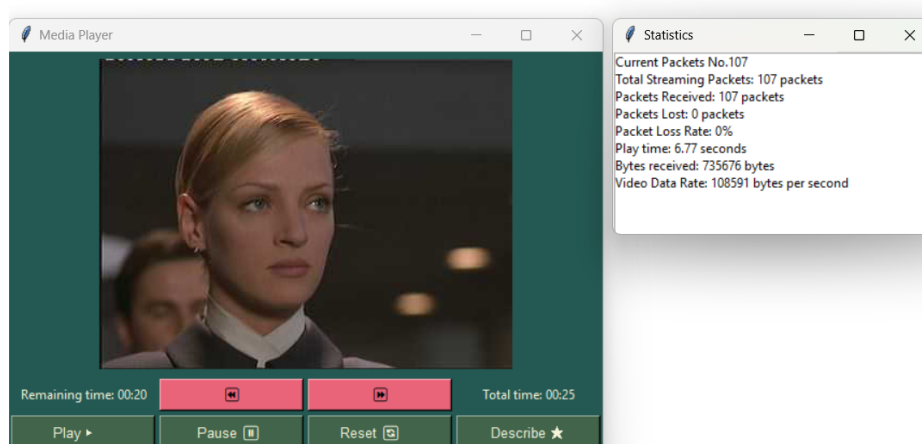


Hình 10: Giao diện mở rộng khi bấm Play

## Pause

Khi nhấn nút Pause, video sẽ được tạm dừng, các thống số thống kê từ lần PLAY khởi chạy sẽ được hiển thị.

Khi Pause video, người dùng vẫn sử dụng được chức năng FORWARD và BACKWARD, sau khi bấm Play lại, video sẽ được stream từ frame được forward hoặc backward. Tuy nhiên, nhóm vẫn chưa hiện thực được khu vực hiển thị video theo forward và backward lúc trạng thái PAUSE.



Hình 11: Giao diện mở rộng khi bấm Pause

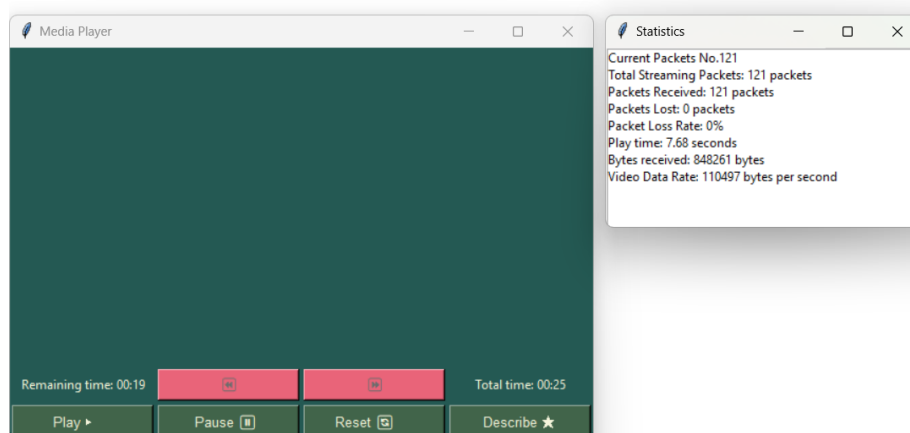
## Stop

Khi nhấn nút Stop, ứng dụng sẽ quay về trạng thái ban đầu và reset các biến thống kê. Đồng thời, chương trình sẽ gọi chức năng TEARDOWN thông số thống kê sẽ được hiển thị.

Người dùng yêu cầu SETUP đến máy chủ khi cần thiết lập các thông số cho luồng phương tiện (địa chỉ IP, cổng (port), số cổng, giao thức (UDP hoặc TCP) và định dạng phương tiện (AAC, MJPEG,...). Những thiết lập này hiện thức trước khi luồng (thread) phát video bắt đầu, ở bước thiết lập RTSP.

Đối với nút TEARDOWN, có thể không thích hợp để gửi TEARDOWN ngay khi người dùng bấm nút STOP trong tất cả các trường hợp. Nếu người dùng tạm dừng video bằng nút PAUSE, ví dụ, có thể thích hợp hơn để gửi lệnh PAUSE đến máy chủ thay vì TEARDOWN, để video có thể được phát lại sau đó từ cùng một điểm. Tuy nhiên, nếu người dùng bấm STOP mà không tạm dừng, có thể phù hợp để gửi TEARDOWN để chấm dứt phiên phát và giải phóng tài nguyên mạng.

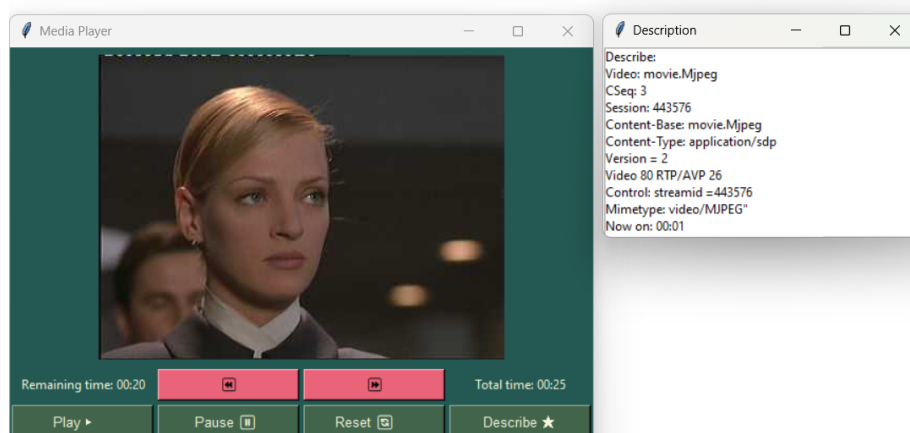
Để triển khai hành vi này, nhóm cung cấp một hộp thoại xác nhận hoặc một lời nhắc để thông báo cho người dùng về hậu quả của việc bấm STOP, đặc biệt là nếu luồng video đang được chia sẻ với người dùng khác hoặc nếu có hạn chế băng thông.



Hình 12: Giao diện mở rộng khi bấm Stop

### Describe

Để mô tả thông tin cho ta biết về video đang stream, nhóm hiện thực thêm nút Describe. Lúc được kích hoạt, ứng dụng sẽ mở ra một winform mới hiển thị những thông tin về video. Tiền điều kiện : state đang ở trạng thái READY hoặc PLAYING ( trạng thái đã SETUP)



Hình 13: Giao diện mở rộng khi bấm Describe

### Thoát chương trình (X)

Khi nhấn nút "X" ở góc phải màn hình, ứng dụng sẽ hỏi người dùng về quyết định này. "OK" để thoát ứng dụng, "Cancel" để hủy thao tác.

## 6.3 Implementation code for Extending Client

Trong tệp ClientExtend.py có thêm những hàm, chức năng được cải thiện từ tệp tin Client.py.

### 6.3.1 Play Button

```
1 def playMovie(self):
2     """Play button handler."""
3     if self.state == self.INIT and self.isFirstPlay == True:
4         self.isFirstPlay = False
5         self.checkPlay = True
6         self.frameNbr = 0
7         self.countTotalPacket = 0
8         self.timerBegin = 0
9         self.timerEnd = 0
10        self.timer = 0
11        self.bytes = 0
12        self.packets = 0
13        self.packetsLost = 0
14        self.lastSequence = 0
15        self.arrivalTimeofPreviousPacket = 0
16        self.lastPacketSpacing = 0
17        self.setupMovie()
18        while self.state != self.READY:
19            pass
20        self.forward["state"] = "normal"
21        self.backward["state"] = "normal"
22        self.describe["state"] = "normal"
23        if self.state == self.READY:
24            self.checkPlay = True
25            # Create a new thread to listen for RTP packets
26            threading.Thread(target=self.listenRtp).start()
27            self.playEvent = threading.Event()
28            self.playEvent.clear()
29            self.sendRtspRequest(self.PLAY)
```

Program 1: Handle Play button

### 6.3.2 Pause Button

Khi nhấn Pause, ứng dụng sẽ mở một cửa sổ mới, chứa thông tin về trạng thái của stream video như **Hình 11**

#### RTP Packet: LOST RATE

Trong các bài lab, nhóm tác giả thường xác định RTP packet lost bằng RTP Stream Application của WireShark.

Trong bài tập lớn này, nhóm hiện thực thủ công dựa trên sequence number ở phía bên Client. Nếu sequence hiện tại (currSequence) không tăng dần theo thì packet đã bị thiếu (loss).

Trong môi trường giả định đường truyền không ổn định, so với cách so sánh tính trung bình số packet bị mất mỗi phút chia cho 20 packets( tệp tin ServerWorker.py định nghĩa 0,05s gửi 1 gói tin). Nhóm dùng tỷ lệ của tổng số packet đã mất tại thời điểm người dùng nhấn "Pause" bằng biến đếm chia cho tổng số gói tin.

Biến này sẽ được tính toán ( giữ nguyên nếu nhận được gói tin, tăng 1 nếu không nhận được) mỗi khi listenRtp được gọi.

### Video data rate

Video Data rate, nhóm tính bằng tổng dung lượng Payload (data - header). chia cho thời gian đã chạy video.

Nhóm dùng thư viện time của python.

```
1 def displayStatus(self):
2     rateLoss = ((self.counter) / (self.countTotalPacket)) * 100
3     top1 = Toplevel()
4     top1.title("Statistics")
5     top1.geometry('300x170')
6     statusBox = Listbox(top1, width=80, height=20)
7     statusBox.insert(1, "Current Packets No.%d " % self.frameNbr)
8     statusBox.insert(2, "Total Streaming Packets: %d packets" %
9 self.countTotalPacket)
10    statusBox.insert(3, "Packets Received: %d packets" % self.
11 packets)
12    statusBox.insert(4, "Packets Lost: %d packets" % self.counter)
13    statusBox.insert(5, "Packet Loss Rate: %d%" % rateLoss)
14    statusBox.insert(6, "Play time: %.2f seconds" % self.timer)
15    statusBox.insert(7, "Bytes received: %d bytes" % self.bytes)
16    statusBox.insert(8, "Video Data Rate: %d bytes per second" % (
17 self.bytes / self.timer))
18    statusBox.pack()
```

Program 2: displayStatus() function in ClientExtend.py

### 6.3.3 Reset Button

```
1 def resetMovie(self):
2     """Reset button handler."""
3     if self.checkPlay:
4         self.checkPlay = False
5         self.sendRtspRequest(self.TEARDOWN)
6         try:
7             for i in os.listdir():
8                 if i.find(CACHE_FILE_NAME) == 0:
9                     os.remove(i)
10        except:
11            pass
12        time.sleep(1)
13        self.forward["state"] = "disabled"
14        self.backward["state"] = "disabled"
15        self.rtspSeq = 0
16        self.sessionId = 0
17        self.requestSent = -1
18        self.teardownAcked = 0
19        self.counter = 0
20        self.isFirstPlay = True
21        self.isForward = 0
```

```
22     self.isBackward = 0
23     self.currentTime = 0
24     self.connectToServer()
25     self.rtpSocket = socket.socket(socket.AF_INET, socket.
SOCK_DGRAM)
26     self.label.pack_forget()
27     self.label.image = ''
```

Program 3: Handle Reset Button

#### 6.3.4 Describe button

Nút Describe trong giao diện người dùng, sẽ đi kèm với những hàm mới cả bên phía client lẫn server.

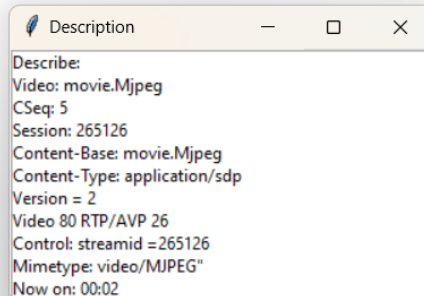
##### Phía Client

Hàm gửi yêu cầu DESCRIBE tới server từ phía Client.

```
1 def describeMovie(self):
2     """Describe button handler"""
3     self.sendRtpRequest(self.DESCRIBE)
```

Program 4: Sent request function from Client to Server in ClientExtend.py

Hàm in ra những thông tin về stream video. Nội dung hiển thị sẽ giống như **Hình 14**



Hình 14: Description Window

- version là phiên bản của giao thức
- Video 80 RTP/AVP 26: kiểu của chương trình và địa chỉ transport. Theo đặc tả SDP Được in theo định dạng  
`<media-type> <RTP-port> <transport> <fmt>`  
Với <media-type> : Video; <RTP-port> : cổng giao tiếp RTP; <transport> : RTP/AVP; <fmt> là MJPEG\_TYPE = 26.
- Control: streamid = ... là session id của video stream.

- Minetype: video/MJPEG: Kiểu phương tiện, dạng định danh hai phần dành cho video và nội dung được truyền trên internet. "video/MJPEG" cho biết kiểu nội dung đang streaming trên session là video và file được mã hóa bằng MJPEG.

```
1 def displayDescription(self, lines):
2     top = Toplevel()
3     top.title("Description")
4     top.geometry('300x180')
5     descriptionBox = Listbox(top, width=50, height=30)
6     descriptionBox.insert(1, "Describe: ")
7     descriptionBox.insert(2, "Video: " + str(self.fileName))
8     descriptionBox.insert(3, lines[1])
9     descriptionBox.insert(4, lines[2])
10    descriptionBox.insert(5, lines[3])
11    descriptionBox.insert(6, lines[4])
12    descriptionBox.insert(7, lines[5])
13    descriptionBox.insert(8, lines[6])
14    descriptionBox.insert(9, lines[7])
15    descriptionBox.insert(10, lines[8])
16    descriptionBox.insert(11, "Now on: " + "%02d:%02d" % (self.
17        currentTime // 60, self.currentTime % 60))
18    descriptionBox.pack()
```

Program 5: Function displayDescribe() for display information about streaming

### Phía Server

Hàm xử lý yêu cầu từ phía Client của server.

```
1 def replyDescribe(self, code, seq, filename):
2     """Send RTSP Describe reply to the client."""
3     if code == self.OK_200:
4         describeReply = 'RTSP/1.0 200 OK\r\nCSeq: ' + seq + '\r\nSession: '
5         + str(self.clientInfo['session'])
6         # body
7         descriptionBody = "\nVersion = 2"
8         descriptionBody += "\nVideo " + self.clientInfo['rtspPort'] + "
9         RTP/AVP " + str(MJPEG_TYPE)
10        descriptionBody += "\nControl: streamid =" + str(self.
11            clientInfo['session'])
12        descriptionBody += "\nMimetype: video/MJPEG\r\n"
13        # end body
14        describeReply += "\nContent-Base: " + filename
15        describeReply += "\nContent-Type: " + "application/sdp"
16        describeReply += descriptionBody
17        connSocket = self.clientInfo['rtspSocket'][0]
18        connSocket.send(describeReply.encode())
```

Program 6: Reply DESCRIBE request from Client in ServerWorker.py



## 6.4 Timing label and Forward, Backward button

### 6.4.1 Timing label

Khởi tạo 2 khối dành cho tổng thời gian và thời gian còn lại của video qua thư viện tkinter. Khi người dùng bấm vào nút PLAY ( đồng thời yêu cầu SETUP được gửi đi), RTSP request sẽ được gửi tới server. Sau đó, ngoài việc gọi replyRtsp() như bình thường, ServerWorker còn gọi thêm hàm replyConnection() - một hàm nhóm làm thêm dựa theo hàm replyRtsp().

Trong đó có hàm getTotalTime() ở file Videostream.py.

```
1 def replyConnection(self, code, seq):
2     """Send RTSP reply to the client."""
3     if code == self.OK_200:
4         totalTime = self.clientInfo['videoStream'].getTotalTime()
5         reply = 'RTSP/1.0 200 OK\r\nCSeq: ' + seq + '\r\nSession: ' +
6         str(self.clientInfo['session']) + '\r\nTotalTime: ' + str(
7         totalTime)
8         connSocket = self.clientInfo['rtspSocket'][0]
9         connSocket.send(reply.encode())
10    # Error messages
11    elif code == self.FILE_NOT_FOUND_404:
12        print("404 NOT FOUND")
13    elif code == self.CON_ERR_500:
14        print("500 CONNECTION ERROR")
```

Program 7: replyConnection() function in ServerWorker.py

### Hàm getTotalTime()

- Tính tổng thời gian bằng cách lấy số frame của video nhân với 0.05 (khoảng thời gian giữa 2 lần server gửi packet cho client). Đây là cách tính thời gian thực của video, trong thực tế, do đường truyền, thời gian stream video có thể lâu hơn.
- Nhấn tổng thời gian sẽ được cập nhật ở lần play video đầu tiên.
- Nhấn thời lượng còn lại bằng cách lấy tổng thời gian (total time) trừ cho thời gian hiện tại (currTime, tương tự với tổng thời gian, currTime được tính bằng cách lấy số frame hiện tại nhân cho 0.05)

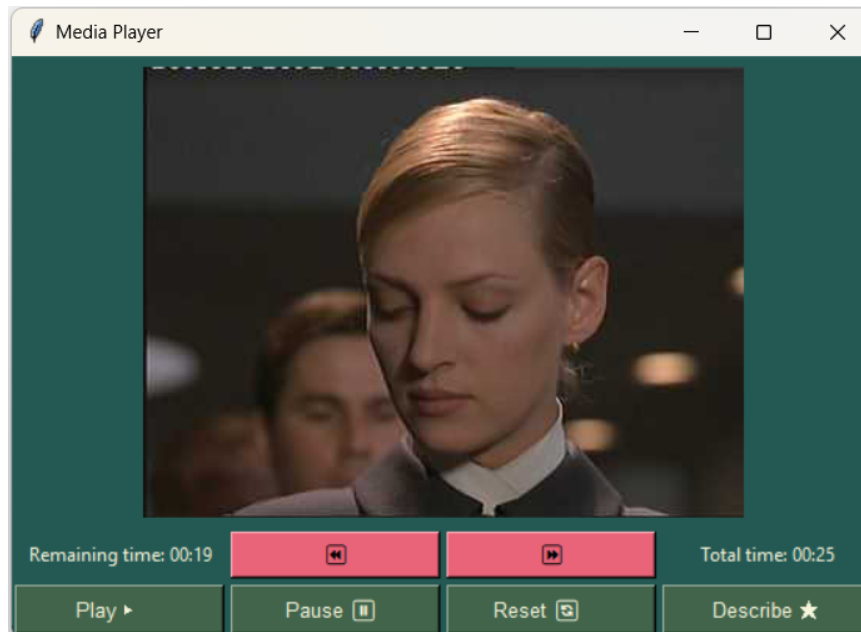
```
1 def getTotalTime(self):
2     self.totalFrame=0
3     while True:
4         data = self.file.read(5)
5         if data:
6             framelength = int(data)
7             # Read the current frame
8             data = self.file.read(framelength)
9             self.totalFrame += 1
10        else:
11            self.file.seek(0)
12            break
13        totalTime = self.totalFrame * 0.05
```

```
14 return totalTime
```

Program 8: Function `getTotalTime()` in `VideoStreaming.py`

#### 6.4.2 FORWARD and BACKWARD button

Nhóm hiện thực thêm chức năng forward và backward ở cố định 10% video.



Hình 15: Backward (◀) and Forward (▶) button

#### FORWARD button (▶)

Khi nhấn nút "▶", hàm `forwardMovies()` sẽ được gọi và hàm này sẽ tiếp tục gọi hàm `sendRtpRequest` với tham số truyền vào là `"self.FORWARD"`

- Hàm `sendRtpRequest` sẽ tiếp tục gửi request tới server.
- sau khi server nhận được request, flag `isNext` ở đối tượng `VideoStream` sẽ được bật bên.
- Hàm `sendRtp` được server gọi, hàm này sẽ tiếp tục gọi hàm `nextFrame` ở đối tượng `VideoStream`.
- `nextFrame` được gọi với flag `isNext` đã được bật, hàm sẽ trả về data của frame ở thứ tự 10% frame tiếp theo thay vì data ở frame tiếp theo như bình thường, đồng thời flag `isNext` sẽ được đưa về lại giá trị 0.
- Nếu số lượng frame còn lại của video không còn đến 10%, hàm `nextFrame` sẽ trả về data của frame cuối cùng.

```
1 def forwardMovies(self):  
2     self.sendRtpRequest(self.FORWARD)
```

```
3 self.isForward = 1
```

Program 9: Function forwardMovies() in ClientExtend.py

```
1 def nextFrame(self):
2     """Get next frame."""
3     if self.isNext == 1:
4         forwardFrames = int(self.totalFrame*0.1)
5         remainFrames = int(self.totalFrame - self.frameNum)
6         if forwardFrames > remainFrames:
7             forwardFrames = remainFrames
8         self.isNext = 0
9     else:
10        forwardFrames = 1
11    if forwardFrames:
12        for i in range(forwardFrames):
13            data = self.file.read(5) # Get the framelength from the
first 5 bits
14            if data:
15                framelength = int(data)
16                # Read the current frame
17                data = self.file.read(framelength)
18                self.frameNum += 1
19    return data
```

Program 10: Function nextFrame() in VideoStream.py

### BACKWARD button («)

Hàm prevMovie sẽ được gọi và hàm này sẽ tiếp tục gọi hàm sendRtspRequest với tham số truyền vào là "self.PREV".

- Hàm sendRtspRequest sẽ tiếp tục gửi request với server.
- Sau khi server nhận được request, flag opt ở đối tượng sẽ được bật lên.
- Khi hàm sendRtp được server gọi với flag opt được bật, hàm này sẽ tiếp tục gọi hàm prevFrame ở đối tượng VideoStream.
- Và khi hàm prevFrame được gọi, hàm sẽ đưa con trỏ đọc data về vị trí đầu video, đồng thời tiếp tục dịch chuyển vị trí con trỏ đến vị trí chứa data của frame cách frame hiện tại 10% frame video về trước, đồng thời trả về data, flag opt sẽ được đưa về lại giá trị 0.
- nếu số lượng frame trước đó của video không còn đến 10%, hàm prevFrame sẽ trả về data của frame đầu tiên.

```
1 if not self.opt:
2     data = self.clientInfo['videoStream'].nextFrame()
3 else:
4     data = self.clientInfo['videoStream'].prevFrame()
5     self.opt = 0
```

Program 11: BACKWARD process in method sendRtp() in ServerWorker.py

```
1 def prevFrame(self):
2     preFrames = int(self.totalFrame * 0.1)
3     if self.frameNum <= preFrames:
4         data = self.file.seek(0)
5         self.frameNum = 0
6         if data:
7             framelength = int(data)
8             # Read the current frame
9             data = self.file.read(framelength)
10            self.frameNum += 1
11        else:
12            data = self.file.seek(0)
13            fFrames = self.frameNum - preFrames
14            self.frameNum = 0
15            for i in range(fFrames):
16                data = self.nextFrame()
17        return data
```

Program 12: prevFrame() in VideoStream.py



## 7 Member list - Group 06

No.	Fullname	Student ID
1	Nguyễn Hữu Hiếu	2013149
2	Lê Bá Dũng	2012863
2	Hoàng Nhật Quân	2014262