

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH



MẠNG MÁY TÍNH - CO3094

Báo cáo bài tập lớn 1

RTSP Video Streaming Application

GV HD: Nguyễn Tấn Đạt

Nhóm SV thực hiện: Lê Đức An - 1912522 (Nhóm trưởng)
Đinh Như Tân - 1915040
Đào Văn Tiến Quyền - 1914880
Trần Toàn - 1915562

Tp.Hồ Chí Minh, Tháng 10/2021



Mục lục

1	Phân tích yêu cầu bài toán	2
1.1	Mục tiêu bài toán	2
1.2	Mô tả các file source code	2
1.2.1	Client, ClientLauncher	2
1.2.2	ServerWorker, Server	2
1.2.3	RtpPacket	2
1.2.4	VideoStream	2
1.3	Phân tích yêu cầu cụ thể	3
1.3.1	Hiện thực RTP packetization ở phía server (file RtpPacket.py)	3
1.3.2	Hiện thực RTSP protocol ở phía client (file Client.py)	3
1.3.3	Hiện thực player dùng để tương tác, hiển thị video	5
2	Hiện thực	6
2.1	Class diagram	6
2.2	Hiện thực các hàm :	6
2.2.1	Hiện thực RTP packetization - hàm encode() - file RtpPacket.py	6
2.2.2	Hiện thực các hàm trong ClientGUI.py	7
3	Hiện thực phần mở rộng	8
3.1	Extend 1 - Thống kê về session	8
3.2	Extend 2 - Bỏ qua nút SETUP	9
3.3	Extend 3 - Hiện thực DESCRIBE button	9
3.4	Extend 4 - Total time/Remaining Time/Progress bar	10
4	Hướng dẫn sử dụng và giới thiệu các chức năng của ứng dụng	11
4.1	Giới thiệu và các bước sử dụng	11
4.2	Các tính năng mở rộng	14
5	Đánh giá kết quả đạt được	14
5.1	Kết quả đạt được	14
5.2	Chưa đạt được	15
6	Source code và demo	15

1 Phân tích yêu cầu bài toán

1.1 Mục tiêu bài toán

Mục tiêu bài tập lớn:

Hiểu và hiện thực một mô hình client và server dùng để stream video sử dụng Real-Time Streaming Protocol (RTSP) và gửi nhận data (video) sử dụng Real-time Transfer Protocol (RTP)

1.2 Mô tả các file source code

Chú ý: các file source code được cung cấp là tham khảo, ta có thể chỉnh sửa/thêm bớt để hoàn thiện phù hợp

1.2.1 Client, ClientLauncher

ClientLauncher.py Có nhiệm vụ nhận vào 4 tham số: Servername, server_port, RTP_port và tên video file sau đó hàm main sẽ khởi chạy giao diện GUI sử dụng thư viện chuẩn tkinter của Python và khởi chạy một Client với các thông số ở trên.

Client

Định nghĩa các button trong giao diện Player của Client: các nút điều khiển Setup, Play, Pause, Teardown, exit đồng thời xử lý logic khi người dùng bấm vào các nút này. Ngoài ra nó còn có các function để hiện thực các logic cần thiết để hiển thị các frame hình ảnh lên giao diện GUI (trình chiếu phim từ server)

1.2.2 ServerWorker, Server

Đây là 2 module dùng để hiện thực phía server dùng để phản hồi các RTSP request và stream video.

Server Tương tự với ClientLaucher, nó sẽ nhận vào tham số là server port để khởi tạo một socket và đợi các kết nối đến socket đó

ServerWorker.py : Xử lý các logic để phản hồi các yêu cầu của client đối với các nút Setup, Play, Pause, Teardown. 2 module này kết hợp để hiện thực server, server có nhiệm vụ phản hồi các RTSP request và stream video. Server sẽ sử dụng module RtpPacket để thực hiện các chức năng của nó và ta cũng không cần chỉnh sửa gì ở module này.

1.2.3 RtpPacket

Đây là module để xử lý liên quan đến RTP packets. Nó có các quy trình riêng để xử lý các gói tin đã nhận ở phía client và ta không cần phải sửa nó. Nhưng ta cần hoàn thành hàm encode đầu tiên của module này để gói tin hoá dữ liệu của video (tức biến video thành cách gói tin bằng cách đóng gói nó với header cần thiết). Ngoài ra còn có hàm decode() nhưng ta không cần thay đổi nó (đã được hiện thực)

1.2.4 VideoStream

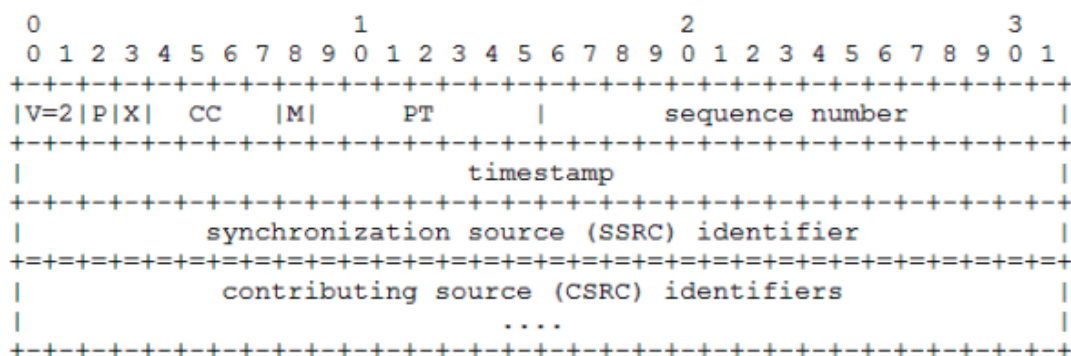
Đây là module dùng để đọc video data từ file ở ổ đĩa. Ở trong bài tập lớn này đó là file movie.Mjpg. Ta cũng không cần thay đổi file này.

1.3 Phân tích yêu cầu cụ thể

1.3.1 Hiện thực RTP packetization ở phía server (file RtpPacket.py)

Ta cần hiện thực hàm `encode()` trong file `RtpPacket.py`

- Ở trường RTP-version ta set là 2
- Trường padding (P) , extension (X), CC, Marker (M) trong bài tập lớn này ta set là 0
- Trường payload type (PT) : ta sử dụng định dạng MJPEG nên `type=26`
- Sequence number: Server sẽ cung cấp số Sequence number (tức `framenbr` - tham số) cho hàm `encode`.
- Timestamp được cung cấp bởi module `time` của Python
- SSRC ta có thể chọn bất kì số nguyên nào.
- Độ dài của packet header là 12 bytes. Cấu trúc của nó tương ứng với 3 dòng đầu tiên trong hình dưới. CSRC ta bỏ qua, không tồn tại trong BTL này.



Hình 1: Sơ đồ header

1.3.2 Hiện thực RTSP protocol ở phía client (file Client.py)

Để làm được điều này, ta để ý trong file `Client.py` có các hàm với các mục `#TODO` cần hiện thực. Đây là các hàm sẽ được gọi khi user click các nút điều khiển ở trên player user interface. Mỗi nút sẽ gọi tới các hàm để xử lý thao tác tương ứng của người dùng. Ta sẽ cần thực hiện các button sau:

Giải thuật sẽ dùng để hiện thực các chức năng:

SETUP:

- Tập hợp các tham số đầu vào: `serverAddress/serverPort/rtpPort/filename`
- Kết nối đến server: Tạo một socket để nhận RTP data và set timeout cho socket khoảng 5 mili giây.

- Gửi một request "SETUP" tới server. Ta cần đóng gói phần Transport header để chỉ ra port của socket mà ta đã tạo ở trên.
- Đợi và đọc phản hồi (reply) từ server và phân tích cú pháp của nó, đặc biệt là phần header ở trong response để lấy session ID bằng một thread mới.
- Set state của Client sang "READY"

PLAY:

- Nếu state của Client đang là INIT thì thực hiện lệnh SETUP (extend 2) nếu không thì tạo một thread mới để lắng nghe các packet từ server
- Tiếp theo, ta cần gửi PLAY request tới server. Ở phần header ta phải chèn Session và session ID mà ta nhận được từ response của bước SETUP
- Trong thread mới ở trên sẽ có các hàm nhận và phân tích gói tin từ server

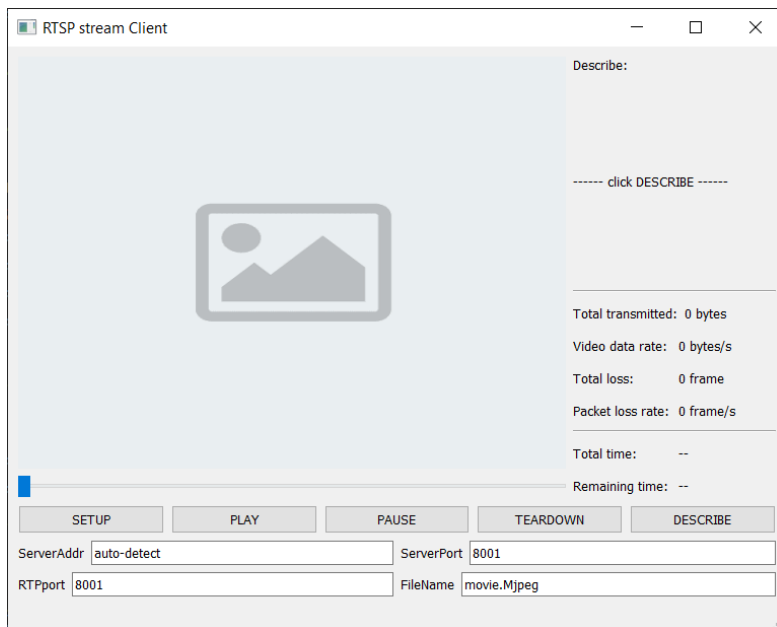
PAUSE:

- Gửi tới server PAUSE request, ta phải chèn Session header và sử dụng Session ID trả về bởi SETUP response
- Sau khi gửi thì thread chờ phản hồi từ server và Video sẽ được tạm dừng vì không nhận được packet nào nữa.
- Ta chỉ thực hiện các thao tác trên nếu state của Client đang là "PLAYING"

TEARDOWN:

- Gửi yêu cầu tới server với mã TEARDOWN
- Sau khi gửi thì thread chờ phản hồi từ server và Server sẽ đóng kết nối
- Tiếp theo ta sẽ dọn dẹp các file cache (xóa chúng) còn trong đĩa và gọi lệnh thoát chương trình.
- Reset giao diện về lại như ban đầu

1.3.3 Hiện thực player dùng để tương tác, hiển thị video

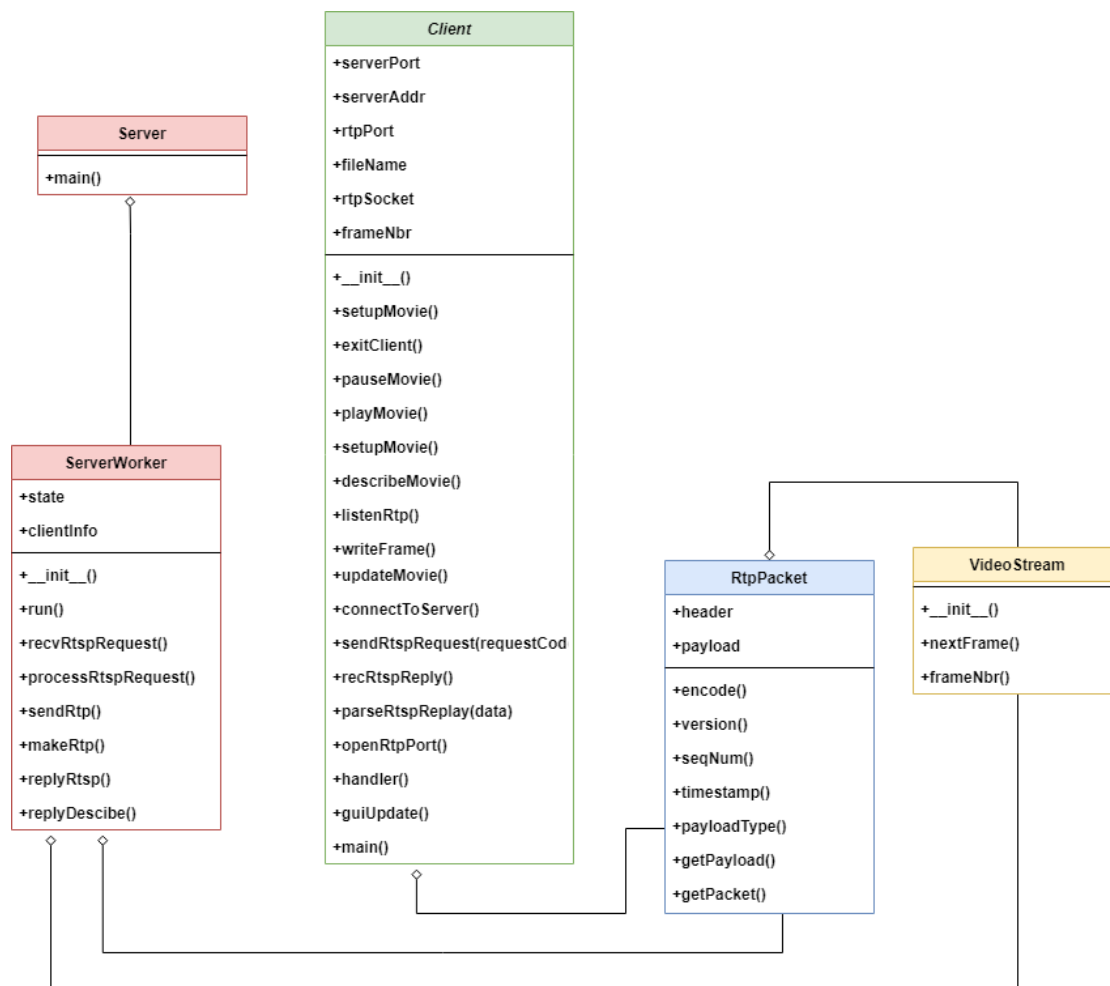


Hình 2: Giao diện Client sẽ hiện thực

- Trong bài tập lớn này, source code được cung cấp sử dụng Tkinter nhưng để thuận tiện trong phần mở rộng (Extend) cũng như ta sẽ chuyển sang sử dụng framework PyQt5
- Ngoài ra, thay vì sử dụng 2 file ClientLauncher.py và Client.py ta sẽ gộp lại thành một file ClientGUI.py khi dùng PyQt5
- Các hàm cốt lõi trong ClientGUI.py hoàn toàn được lấy từ Client.py, chỉ thay đổi để phù hợp với PyQt5
- Các file source khác giữ nguyên cấu trúc

2 Hiện thực

2.1 Class diagram



Hình 3: Class Diagram cho hệ thống

2.2 Hiện thực các hàm :

2.2.1 Hiện thực RTP packetization - hàm `encode()` - file `RtpPacket.py`

- Ta cần hiện thực hàm `encode()` trong file `RtpPacket.py`
- Đây là hàm tạo các header của gói tin RTP

```

def encode(self, version, padding, extension, cc, seqnum, marker, pt, ssrc, payload):
    header = bytearray(HEADER_SIZE)
    timestamp = int(time())
    # Encode the RTP packet
  
```

```
# Fill the header bytearray with RTP header fields
header[0] = (header[0] | version << 6) & 0xC0 # 2 bits
header[0] = (header[0] | padding << 5) # 1 bit
header[0] = (header[0] | extension << 4) # 1 bit
header[0] = (header[0] | (cc & 0x0F)) # 4 bits
header[1] = (header[1] | marker << 7) # 1 bit
header[1] = (header[1] | (pt & 0x7f)) # 7 bits
header[2] = (seqnum & 0xFF00) >> 8 # 16 bits total, this is first 8
header[3] = (seqnum & 0xFF) # second 8
header[4] = (timestamp >> 24) # 32 bit timestamp
header[5] = (timestamp >> 16) & 0xFF
header[6] = (timestamp >> 8) & 0xFF
header[7] = (timestamp & 0xFF)
header[8] = (ssrc >> 24) # 32 bit ssrc
header[9] = (ssrc >> 16) & 0xFF
header[10] = (ssrc >> 8) & 0xFF
header[11] = ssrc & 0xFF

# Set RtpPacket's header and payload.
self.header = header
self.payload = payload
```

2.2.2 Hiện thực các hàm trong ClientGUI.py

Do source code của Class này quá dài, nên dưới đây chỉ liệt kê anh sách các hàm đã hiện thực và cộng dụng của nó:

Toàn bộ source code nằm trong file đính kèm: file ClientGUI.py

Chú ý: như đã trình bày ở trên, file ClientGUI.py là hợp nhất của Client.py và Client-Launcher.py khi xây dựng bằng PyQt5. Cấu trúc hiện thực dưới đây đều lấy từ file gốc Client.py

```
#Hiện thực logic khi user nhấn vào nút SETUP trên giao diện:
def setupMovie(self): pass
#Hiện thực tác vụ khi user nhấn vào nút TEARDOWN trên giao diện:
def exitClient(self): pass
#Hiện thực tác vụ khi user nhấn vào nút PAUSE trên giao diện:
def pauseMovie(self): pass
#Hiện thực tác vụ khi user nhấn vào nút PLAY trên giao diện:
def playMovie(self): pass
#Hiện thực tác vụ khi user nhấn vào nút DESCRIBE trên giao diện (thuộc phần mở rộng):
def describeMovie(self): pass
#Function dùng để lắng nghe các RTP packet gửi từ server, hoạt động trên một thread riêng biệt
def listenRtp(self): pass
#-----
# Dưới đây các các hàm phụ trợ cho các tác vụ trên:

#Ghi frame nhận được xuống file tạm và trả về file ảnh:
def writeFrame(self, data): pass

#Cập nhật khung hình trên GUI, tạo nên hiệu ứng video:
```



```
def updateMovie(self, imageFile): pass

# Tạo kết nối đến server, phục vụ tác vụ SETUP:
def connectToServer(self): pass
#Gửi Rtp request tới server, phục vụ cho các tác vụ SETUP/PLAY/PAUSE/TEARDOWN/DESCRIBE
def sendRtpRequest(self, requestCode): pass

# Nhận phản hồi từ server (chạy trên một thread khác)
def recvRtpReply(self): pass

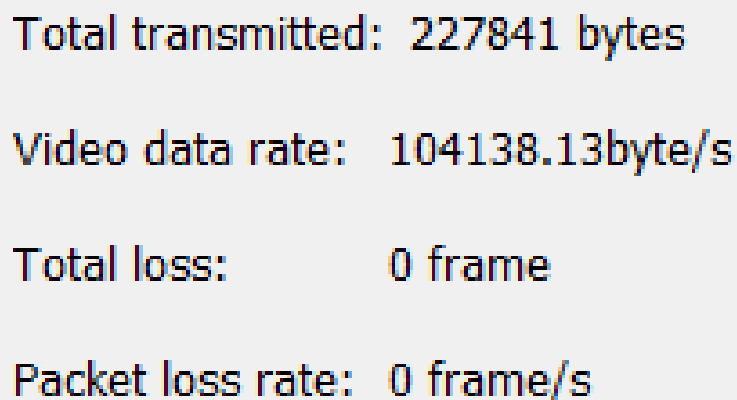
# Phân tích cú pháp gói tin nhận được từ server:
def parseRtpReply(self, data): pass

#Gắn rtpPort được chỉ định vào socket:
def openRtpPort(self): pass
#-----
# Xử lý tác vụ khi user nhấn vào nút "X" để đóng chương trình
def handler(self): pass
```

3 Hiện thực phần mở rộng

3.1 Extend 1 - Thống kê về session

Yêu cầu: Thử tính toán các giá trị thống kê về session đang stream như: Tỷ lệ rớt các gói tin (loss rate), video data rate.



```
Total transmitted: 227841 bytes
Video data rate: 104138.13byte/s
Total loss:      0 frame
Packet loss rate: 0 frame/s
```

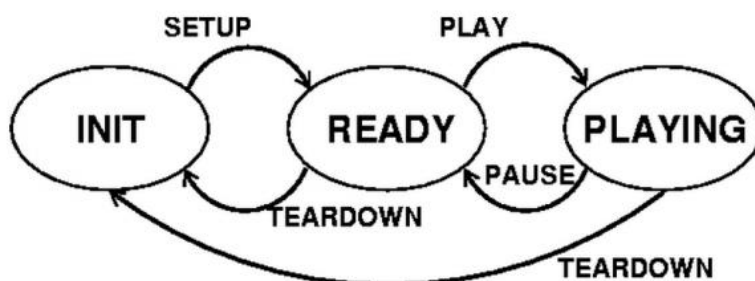
Hình 4: Phần mở rộng 1

Solution:

- Ở phía Client, mỗi khi ta nhận được một phản hồi từ server, ta sẽ tính số byte đã nhận được và cộng dồn chúng lại ta sẽ có được tổng data đã truyền ở mục **Total transmitted**
- Ta lấy total transmitted trên chia cho tổng số thời gian đã chạy sẽ ra video data rate
- Mỗi khi có một gói tin được gửi lại (dựa vào frame của gói tin, #frame của gói tin sau không tăng dần) ta sẽ biết có gói tin bị loss và đếm nó (total loss) , lấy nó chia cho tổng số thời gian để tìm loss rate

3.2 Extend 2 - Bỏ qua nút SETUP

Yêu cầu: So sánh với các trình player khác ta thấy chỉ có 3 nút là PLAY, PAUSE, STOP mà không có nút SETUP. Mặt khác thao tác SETUP là bắt buộc, câu hỏi đặt ra là tìm giải pháp để không cần nhấn SETUP mà video vẫn PLAY được và hiện thực nó.



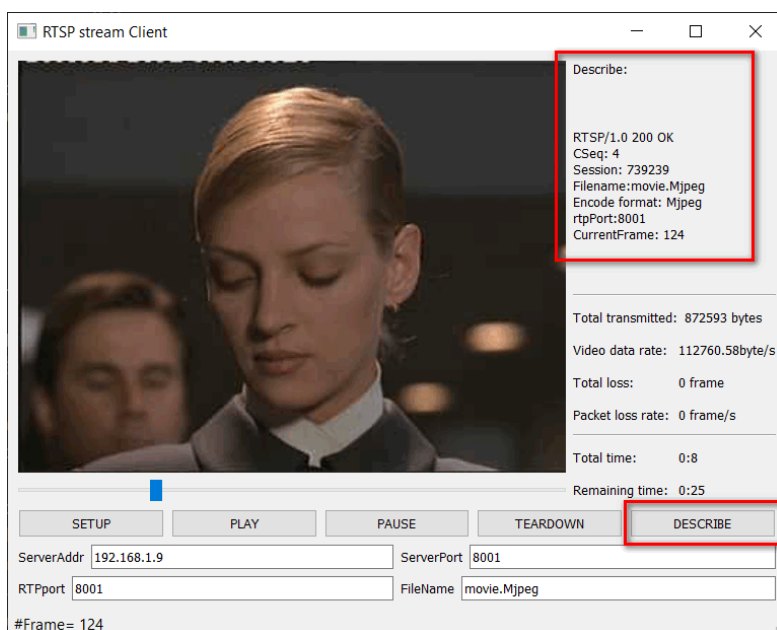
Hình 5: Phần mở rộng 2 - sơ đồ trạng thái

Solution:

- Khi chưa nhấn SETUP thì trạng thái (biến state) sẽ được thiết lập giá trị là INIT (tức chưa setup)
- Khi nhấn SETUP rồi thì state chuyển sang READY (dựa vào Client State chart trong đề)
- Từ đó giải pháp đơn giản là khi user nhấn vào nút PLAY , ta chỉ cần kiểm tra state lúc đó đang ở trạng thái nào, nếu là INIT thì gọi hàm tương ứng với nút SETUP rồi tiến hành PLAY còn đang ở trạng thái READY sẵn rồi thì chỉ cần PLAY như bình thường.

3.3 Extend 3 - Hiện thực DESCRIBE button

Yêu cầu: Hiện thực thêm một nút là DESCRIBE để yêu cầu server gửi các thông tin về media stream như sessionId, encoding đã sử dụng,...



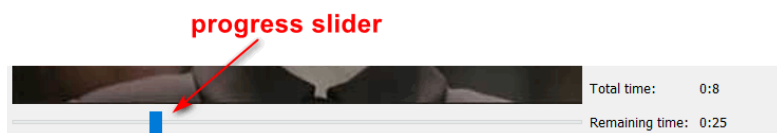
Hình 6: Phần mở rộng 3

Solution:

- Đầu tiên ta tạo một nút DESCRIBE như các nút khác trên Client UI và gắn nó vào một hàm xử lý
- Tiếp đến ta thêm một loại request là describe và định nghĩa một request mới tên là DESCRIBE (dựa trên các hàm xử lý PLAY/PAUSE đã có)
- ở phía server ta cũng thêm một phần xử lý việc nếu nhận được gói tin với requestCode là DESCRIBE thì sẽ trả về các thông số gì (hiện thực dựa trên các hàm đã có sẵn của các request khác) Trở về lại phía client, ở hàm phân tích cú pháp, nếu nhận ra đây là phản hồi của request DESCRIBE thì tiến hành cập nhật ClientGUI là các thông tin nhận được ở mục "Describe"

3.4 Extend 4 - Total time/Remaining Time/Progress bar

Yêu cầu: Hiện thực các chức năng về thông tin video đang stream như tổng thời gian đã hiển thị (đã play) , thời gian còn lại, thanh tiến trình



Hình 7: Phần mở rộng 4

Solution:

- Trong thực tế ta sẽ lấy được độ dài của video từ server gửi về. Ở đây ta biết video có sẵn có 500 khung hình, từ đó ta dựa vào số lượng frame (khung hình) đã nhận ở một thời điểm và từ đó ước lượng ra được thời gian đã qua trên tổng số thời gian cần để play video. Nghĩa là hiện thực thanh progress bar.
- Ta sử dụng module time của python để đánh dấu các thời điểm cập nhật khung hình, mỗi khi video được cập nhật khung hình ta lại cộng dồn các khoảng thời gian nhỏ lại và làm tròn để ra thời gian đã stream. Tức là Total time. Chú ý rằng các khoảng thời gian này không tính lúc video bị PAUSE
- Từ thông tin video ta có thể lấy được tổng thời gian của video nên từ đó ta trừ đi thời gian đã stream sẽ ra thời gian còn lại

4 Hướng dẫn sử dụng và giới thiệu các chức năng của ứng dụng

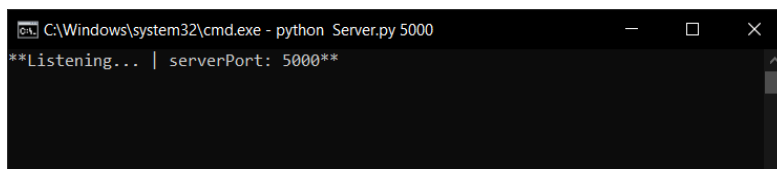
4.1 Giới thiệu và các bước sử dụng

Bước 1: Khởi chạy server

Đầu tiên ta cần khởi chạy một server bằng câu lệnh sau thông qua CMD tại thư mục chứa file Server.py:

```
python Server.py 8001
```

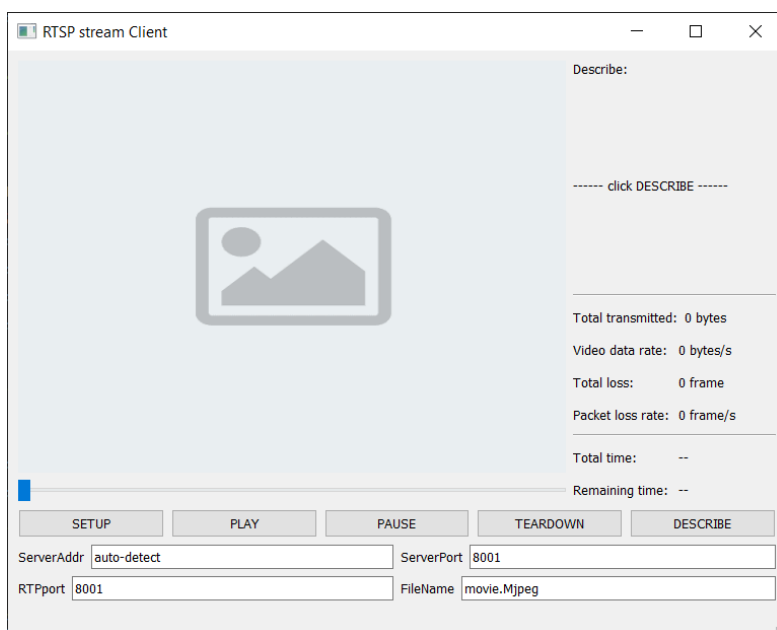
Hoặc đơn giản là khởi chạy file **exe_server.bat** để khởi chạy nhanh mà không cần gõ lại câu lệnh trên.



Hình 8: Sau khi khởi chạy server

Bước 2: Khởi chạy Client

Tiếp theo ta click đúp vào file ClientGUI.py để khởi chạy giao diện client. Hình ảnh UI của Client hiện ra như hình sau:



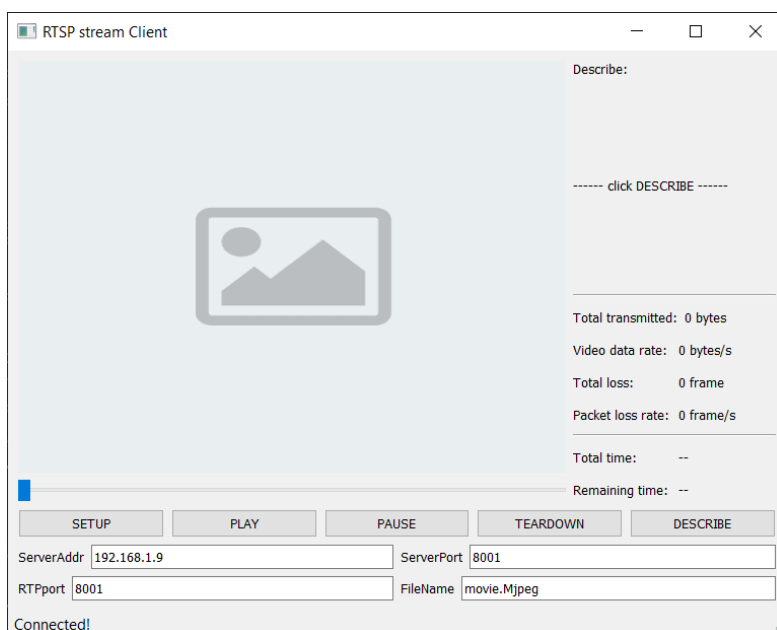
Hình 9: Giao diện Client

Bước 3: Cài đặt thông số

Ta cần chú ý đến 4 tham số đầu vào sau, các tham số này có thể được tùy chỉnh. Các giá trị trên Client là các giá trị mặc định. Chú ý rằng nếu nhập các tham số sai thì sẽ không kết nối được đến server hoặc xảy ra các exception chưa được xử lý.

- ServerAddr: địa chỉ của server. Ở trong bài tập lớn này, do server và client cùng chạy trên một máy tính nên giá trị mặc định của nó là "auto-detect" có nghĩa là khi nhấn SETUP thì nó sẽ tự động lấy địa chỉ IPv4 của máy hiện tại làm địa chỉ server. Ta thường không cần chỉnh sửa nó.
- ServerPort: cổng của server để thực hiện kết nối socket. Giá trị mặc định là 8001, đây cũng là giá trị mặc định mà ta đã cấu hình trong file exe_server.bat. Nếu bạn khởi chạy server với một port khác thì phải tùy chỉnh lại cho giống ở mục này.
- rtpPort: port của client, thông thường client sẽ kết nối đến server nên giá trị này có thể bất kì.
- FileName: tên của file mà sẽ yêu cầu stream. Trong bài tập lớn này ta chỉ có một file là movie.Mjpeg.

Sau khi tùy chỉnh các tham số (nếu có) ta nhấn vào nút **SETUP** để Client thực hiện kết nối đến server và được như hình sau:



Hình 10: Giao diện sau khi nhấn SETUP

Bước 4: Bắt đầu stream

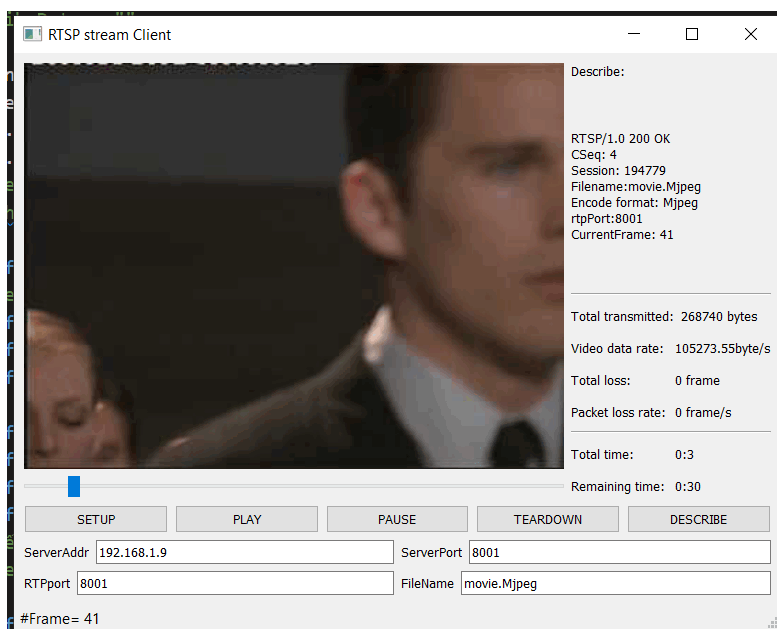
Ta nhấn vào nút PLAY để bắt đầu tiến hành stream video. Thông tin các nút trên Client:

- SETUP: cài đặt các thông số cần thiết (đã nói ở trên)
- PLAY: bắt đầu play/stream video
- PAUSE: dừng stream video
- TEARDOWN: dừng stream, đóng kết nối và tắt Client
- DESCRIBE: yêu cầu Server gửi các thông tin về session đang stream (đây là phần Extend của bài tập lớn) . Các giá trị Client nhận được sẽ hiển thị ở ô trống ngay dưới nhãn "Describe:"

Các thông tin khác trên giao diện:

- Describe: Hiển thị thông tin của session khi người dùng nhấn vào nút DESCRIBE (Phần Extend 3)
- Total transmitted: Tổng số byte Client đã truyền (Phần Extend 1)
- Video data rate: số byte đã truyền/tổng thời gian video đã stream (không tính thời gian dừng - PAUSE) (Phần Extend 1)
- Total loss: Tổng số frame mà bị lỗi khi truyền giữa Client và Server (Phần Extend 1)
- Loss rate: Số frame bị loss/tổng thời gian video đã stream (Phần Extend 1)
- Total time: tổng thời gian đã stream video (Phần Extend 4)

- Remaining time: tổng thời gian còn lại của video (Phần Extend 4)
- Thanh progress bar: hiển thị thanh tiến trình của video đang play. (Phần Extend 4)



Hình 11: Giao diện khi đang thực hiện stream video

Chú ý: Trọng tâm của bài tập lớn này là hiện thực stream video bằng RTSP protocol nên ta bỏ qua các xử lý logic không cần thiết và rườm rà như xử lý validate các tham số đầu vào. Do đó nếu nhập các chuỗi không đúng định dạng và nhấn các nút lộn xộn có thể làm chương trình crash và xảy ra các lỗi không mong muốn.

4.2 Các tính năng mở rộng

Có thể bỏ qua nút SETUP - extend 2

- Ta có thể thiết đặt các thông số (nếu muốn) và nhấn nút PLAY để bắt đầu stream video ngay mà không cần nhấn nút SETUP trước
- Tính năng này tương tự các trình phát Media khác khi không có nút SETUP

Có thể Ngắt kết nối (TEARDOWN) và tiến hành stream lại

- Trong lúc video đang ở trạng thái PLAY ta có thể nhấn TEARDOWN để ngắt kết nối và nhấn PLAY lại để stream video lại từ đầu

5 Đánh giá kết quả đạt được

5.1 Kết quả đạt được

- Hiện thực được chức năng cơ bản của ứng dụng stream video sử dụng giao thức RTSP với 4 nút cơ bản: SETUP/PAUSE/PLAY/TEARDOWN

- Hiện thực được phần mở rộng 1: thống kê các giá trị của video như tổng data đã truyền/loss rate,...
- Hiện thực được phần mở rộng 2: Tích hợp nút SETUP vào nút PLAY (user không cần nhấn SETUP trước khi nhấn nút PLAY)
- Hiện thực được phần mở rộng 3: Tạo thêm nút DESCRIBE để yêu cầu Server trả về các thông tin của phiên hoạt động
- Hiện thực được phần mở rộng 4: Tính và hiển thị các giá trị Total time/Remaining time/Progress bar
- Hiểu được và hiện thực cách đóng một header vào gói tin như thế nào
- Hiểu được các hoạt động của giao thức RTSP dùng để stream video

5.2 Chưa đạt được

- Các giá trị tính toán còn có thể chưa chính xác như các chương trình thực tế vì hạn chế về thuật toán cũng như thời gian và độ phức tạp của ứng dụng.
- Phần extend 4 chưa hiện thực được do thời gian có hạn, đề yêu cầu chọn danh sách nhưng chỉ có một file video và file video này có định dạng không phổ biến.

6 Source code và demo

Video demo: <https://www.youtube.com/watch?v=MK0pzXl8BUk>

Full source code: Đính kèm theo bài nộp tại bài tập lớn, trong file "RTSP streaming source code.zip"

————— END —————