

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



Mạng Máy Tính (CO3003)

Assignment

Video streaming using RTSP and RTP

Giáo viên Hướng dẫn: Bùi Xuân Giang
Sinh viên: Nguyễn Hoàng Quang Khánh - 2013458
Nguyễn Tuấn Kiệt - 2013577
Đặng Phú Quốc - 2014294
Vũ Trần Hoàng - 2013245

HO CHI MINH CITY, JUNE 2022



Contents

1	Member list & Workload	2
2	Phân tích yêu cầu	2
2.1	Giới thiệu	2
2.2	Yêu cầu	2
2.2.1	Yêu cầu chức năng	2
2.2.2	Yêu cầu phi chức năng	3
2.3	Thực thi giao thức RTP trên Server	3
3	Mô tả chức năng	3
3.1	ClientLauncher	4
3.2	Client	4
3.3	ServerWorker	5
3.4	Server	5
3.5	RtpPacket	5
3.6	VideoStream	5
4	Danh sách các thành phần	6
5	Mô hình luồng dữ liệu	6
5.1	Phiên bản bình thường	6
5.2	Phiên bản mở rộng	7
6	Class diagram	7
7	Hiện thực	8
8	Đánh giá tổng thể các kết quả đạt được	11
9	Hướng dẫn sử dụng	11
10	Extend	13
10.1	Extend 1	13
10.2	Extend 2	13
10.3	Extend 3	14
10.4	Extend 4	15
10.5	Extend 5	16

1 Member list & Workload

No.	Fullname	Student ID	Problems	Work %
1	Nguyễn Hoàng Quang Khánh	2013458	- Viết báo cáo. - Nút SETUP, extend 1,2.	25%
2	Nguyễn Tuấn Kiệt	2013577	- Phiên bản mở rộng. - Nút SWITCH, extend 3,4,5.	25%
3	Đặng Phú Quốc	2014294	- Phiên bản mở rộng - Nút BACK, FORWARD, extend 2,3,4.	25%
4	Vũ Trần Hoàng	2013245	- Phiên bản bình thường - Nút SETUP, PLAY, PAUSE, extend 3, 4.	25%

2 Phân tích yêu cầu

2.1 Giới thiệu

Trong Assignment này, chúng ta sẽ triển khai một server streaming video và client giao tiếp bằng Real-Time Streaming Protocol (RTSP) và gửi dữ liệu bằng Real-time Transfer Protocol (RTP). Báo cáo này sẽ cung cấp các thông số chi tiết như cũng như hướng dẫn sử dụng để triển khai ứng dụng của nhiệm vụ này.

2.2 Yêu cầu

2.2.1 Yêu cầu chức năng

1. Sau khi kết nối giữa server và client được thiết lập. Sẽ có một cửa sổ giao diện người dùng của video-streaming xuất hiện
2. Người dùng có thể tương tác với giao diện người dùng (gửi yêu cầu đến Máy chủ) bằng cách nhấp vào các nút
3. Video-streaming phải có ít nhất 4 nút: SETUP, PLAY, PAUSE, TEARDOWN
4. Client gửi yêu cầu PLAY bất cứ khi nào nút SETUP được nhấn. Yêu cầu được sử dụng để thiết lập sessionID và các tham số truyền tải
5. client gửi yêu cầu PLAY sau khi SETUP. Yêu cầu này được sử dụng để bắt đầu phát streaming video từ server. Không thể phát video trừ khi client gửi SETUP trước
6. client có thể gửi yêu cầu PAUSE nếu muốn tạm dừng video trong khi phát trực tuyến
7. Client gửi yêu cầu TEARDOWN. Yêu cầu này sẽ kết thúc phiên, đóng kết nối cũng như đóng cửa sổ giao diện của video-streaming
8. Một phiên bản mở rộng sẽ được cung cấp nếu tất cả các yêu cầu trên được hoàn thành. Ở phiên bản mở rộng sẽ có các nút STOP, BACKWARD, FORWARD, DESCRIBE và SWITCH.
9. Yêu cầu PLAY ngay bây giờ sẽ bắt đầu phát video mà không cần gửi yêu cầu CÀI ĐẶT trước (Phiên bản mở rộng)

10. Khi nhấn nút FORWARD, nút này sẽ chuyển tiếp video trong 1/5 thời lượng của video.
11. Khi nhấn nút BACKWARD, nó sẽ quay ngược video trong 1/5 thời lượng của video.
12. Khách hàng gửi yêu cầu DESCRIBE, nó sẽ gửi lại tệp mô tả session cho client biết loại luồng nào trong session và mã hóa nào được sử dụng, v.v.
13. Yêu cầu TEARDOWN được thay thế bằng yêu cầu STOP, khi gửi đi sẽ thiết lập lại luồng từ đầu
14. Cả hai phiên bản (Bản gốc / Phần mở rộng) sẽ phải cung cấp một nút thoát để đóng luồng video. Khi người dùng nhấp vào nút đó, một cửa sổ bật lên sẽ xuất hiện để xác nhận việc thoát. Nếu có, nó sẽ kết thúc phiên, đóng kết nối cũng như cửa sổ giao diện. Nếu không, nó sẽ bắt đầu phát video

2.2.2 Yêu cầu phi chức năng

1. streaming video server và client giao tiếp sử dụng Real-time Streaming Protocol (RTSP) và gửi dữ liệu bằng Real-time Transfer Protocol (RTP).
2. Thời gian phản hồi từ server phải nhỏ hơn hoặc bằng 0,5 giây
3. Client phải thiết lập kết nối RTP với máy chủ với thời gian chờ là 0,5 giây. Nếu không, nó sẽ phát sinh lỗi kết nối.
4. GUI (giao diện người dùng đồ họa) của video streaming window được áp dụng bằng cách sử dụng mô-đun Tkinter.
5. Tỷ lệ mất gói dưới 1%.

2.3 Thực thi giao thức RTP trên Server

Server sẽ thực hiện đóng gói các dữ liệu thành các gói RTP, thiết lập vùng trong tiêu đề gói và sao chép payload vào gói. Khi máy chủ nhận được yêu cầu PLAY từ client, máy chủ sẽ đọc video frame từ file và tạo RtpPacket-object. Sau đó máy chủ sẽ gửi các frame đến client thông qua giao thức UDP mỗi 50 milliseconds. Để thực hiện đóng gói, server sẽ gọi hàm mã hóa trong RtpPacket class. Rtp header bao gồm:

RTP packet header							
Bit offset ^[b]	0–1	2	3	4–7	8	9–15	16–31
0	Version	P	X	CC	M	PT	Sequence number
32	Timestamp						
64	SSRC identifier						

3 Mô tả chức năng

Chương trình bao gồm 6 phần:

3.1 ClientLauncher

ClientLauncher sẽ chịu trách nhiệm khởi tạo client với server address, serverport, RTP port và file .mpeg sẽ được truyền.

3.2 Client

Client thiết kế giao diện người dùng với 4 nút: SETUP, PLAY, PAUSE, TEARDOWN và hiện thực các hành động khi nhấn nút tương ứng. Trong client bao gồm các hàm:

- def __init__(self, master, serveraddr, serverport, rtpport, filename): khởi tạo client với các tham số mặc định
- def createWidgets(self): tạo giao diện người bao gồm các nút chức năng và khung hiển thị video được phát
- def setupMovie(self): thiết lập đường truyền khi lần đầu phát video
- def exitClient(self): dừng phát video, ngắt kết nối và thoát cửa sổ
- def pauseMovie(self): tạm dừng video đang phát
- def playMovie(self): bắt đầu phát hoặc tiếp tục phát video khi video được dừng.
- def describeMovie(self): hiển thị các thông tin của video đang chọn
- def backMovie(self): tua về sau một đoạn cố định tỉ lệ với độ dài video
- def forwardMovie(self): tua về trước một đoạn cố định tỉ lệ với độ dài video.
- def switchMovie(self): hiển thị danh sách các video để chuyển đổi khi người dùng bấm vào nút SWITCH
- def updateSlider(self, value): cập nhật slider mỗi khi hình ảnh video stream thay đổi
- def listenRtp(self): nhận các gói RTP được gửi từ Server
- def writeFrame(self, data): ghi khung nhận được vào file hình ảnh tạm thời và trả về file hình ảnh
- def updateMovie(self, imageFile): cập nhật file hình ảnh dưới dạng khung video trong giao diện người dùng
- def connectToServer(self): thiết lập kết nối với server bằng cách bắt đầu một phiên RTSP / TCP mới
- def sendRtspRequest(self, requestCode): gửi yêu cầu đến máy chủ khi người dùng nhấp vào các nút trong giao diện người dùng. Chúng ta cũng có thể sử dụng chức năng này để in ra thông tin của tin nhắn được gửi từ client
- def recvRtspReply(self): nhận RTSP trả lời từ máy chủ
- def handleRtsReply(self, data): phân tích cú pháp RTSP trả lời từ server để thiết lập trạng thái của client hoặc quyết định đóng socket
- def openRtpPort(self): mở socket RTP được liên kết với một port được chỉ định.
- def handler(self): xử lý đóng giao diện khi người dùng nhấp vào nút X
- def displayDescribe(self, recvData): Hiển thị thông tin describe sau nhận được từ server

3.3 ServerWorker

ServerWorker xác định trạng thái của server (INIT, READY, PLAYING) và xử lý yêu cầu từ client.

- def __init__(self, clientInfo): xác định thông tin của client
- def run(self): chạy một thread để nhận các yêu cầu RTSP từ client
- def recvRtspRequest(self): nhận các yêu cầu RTSP từ phía client
- def processRtspRequest(self, data): xử lý các yêu cầu RTSP đã gửi
- def sendRtp(self): gửi các gói RTP qua UDP
- def replyRtsp(self, code, seq): gửi RTSP trả lời cho client

3.4 Server

Server sẽ phụ trách việc khởi động Server với server_port và nhận thông tin từ client (address, port) thông qua phiên RTSP / TCP.

3.5 RtpPacket

RtpPacket sẽ xử lý các gói RTP. Nó cũng có phương pháp mã hóa, giải mã packet RTP và nhận thông tin từ packet.

- def encode(self, version, padding, extension, cc, seqnum, marker, pt, ssrc, payload): mã hóa gói RTP với các trường tiêu đề và tải trọng
- def decode(self, byteStream): giải mã gói RTP
- def version(self): trả về phiên bản RTP
- def seqNum(self): trả về seqNum
- def timestamp(self): trả về timestamp
- def payloadType(self): trả về payloadType
- def getPayload(self): trả về payload
- def getPacket(self): trả về packet RTP

3.6 VideoStream

VideoStream sẽ phụ trách việc đọc dữ liệu video từ tệp trên đĩa.

- def __init__(self, filename): mở tệp trong đĩa
- def nextFrame(self): lấy khung dữ liệu video tiếp theo
- def frameNbr(self): lấy số khung

4 Danh sách các thành phần

- **Server:** là máy tính chứa các tệp, chương trình được chia sẻ và hệ điều hành mạng. Server cung cấp quyền truy cập vào tài nguyên mạng cho tất cả người dùng mạng
- **Clients:** là máy tính truy cập và sử dụng mạng và tài nguyên mạng dùng chung. Clients về cơ bản là khách hàng (người dùng) của mạng, khi họ yêu cầu và nhận dịch vụ từ máy chủ.
- **Network Interface Card:** ở mỗi máy tính trong mạng có một thẻ mở rộng đặc biệt được gọi là network interface card (NIC). NIC chuẩn bị (định dạng) và gửi dữ liệu, nhận dữ liệu và kiểm soát luồng dữ liệu giữa máy tính và mạng. Về phía truyền tải, NIC chuyển các khung dữ liệu đến physical layer, lớp này sẽ truyền dữ liệu đến physical link. Về phía người nhận, NIC xử lý các bit nhận được từ physical layer và xử lý thông điệp dựa trên nội dung của nó.
- **LAN Cable:** Một khu vực cáp mạng cục bộ được gọi là cáp dữ liệu hoặc cáp Ethernet là một loại cáp có dây được sử dụng để kết nối một thiết bị với internet hoặc với các thiết bị khác như máy tính, máy in, v.v.

5 Mô hình luồng dữ liệu

5.1 Phiên bản bình thường

Yêu cầu SETUP: người dùng nhấn vào nút SETUP ở giao diện của sổ

- **Client → Server:** Client gửi yêu cầu SETUP chứa phiên bản RTSP, trình tự, cổng và cổng vận chuyển
- **Server → Client:** Server sẽ in quá trình xử lý SETUP ở terminal. Sau khi kết thúc, gửi nội dung phản hồi chữ phiên bản RTSP, trình tự, session.

Yêu cầu PLAY: khi người dùng nhấn vào nút PLAY ở giao diện cửa sổ

- **Server → Client:** Client gửi yêu cầu PLAY chứa phiên bản RTSP, trình tự và session.
- **Client → Server:** Server sẽ in quá trình xử lý PLAY ở terminal và gửi nội dung phản hồi chứa phiên bản RTSP, trình tự, session. Video sẽ chạy khi yêu cầu được xử lý

Yêu cầu PAUSE: người dùng nhấn vào nút PAUSE ở giao diện cửa sổ

- **Client → Server:** Client gửi yêu cầu PAUSE chứa phiên bản RTSP, trình tự và session của server
- **Server → Client:** Server sẽ in quá trình xử lý PAUSE ở terminal và gửi nội dung phản hồi chứa phiên bản RTSP, trình tự, session. Video sẽ chạy khi yêu cầu được xử lý

Yêu cầu TEARDOWN: người dùng nhấn vào nút TEARDOWN ở giao diện cửa sổ

- **Client → Server:** Client gửi yêu cầu TEARDOWN chứa phiên bản RTSP, trình tự và session của server

5.2 Phiên bản mở rộng

Yêu cầu DESCRIBE: người dùng nhấn vào nút DESCRIBE ở trên giao diện của sổ

- Client \rightarrow Server: Client gửi yêu cầu DESCRIBE chứa phiên bản RTSP, trình tự và session của server
- Server \rightarrow Client: Server sẽ in quá trình xử lý DESCRIBE ở terminal và gửi nội dung phản hồi chứa tập mô tả những loại luồng nào trong session và những mã hóa nào được sử dụng.

Yêu cầu FORWARD : người dùng nhấn vào nút FORWARD ở trên giao diện của sổ

- Client \rightarrow Server: Client gửi yêu cầu FORWARD chứa phiên bản RTSP, trình tự và session của server
- Server \rightarrow Client: Server sẽ in quá trình xử lý FORWARD ở terminal và gửi nội dung phản hồi chứa phiên bản RTSP, trình tự, session. Khung hiển thị video sẽ được chuyển về phía trước khi yêu cầu được xử lý

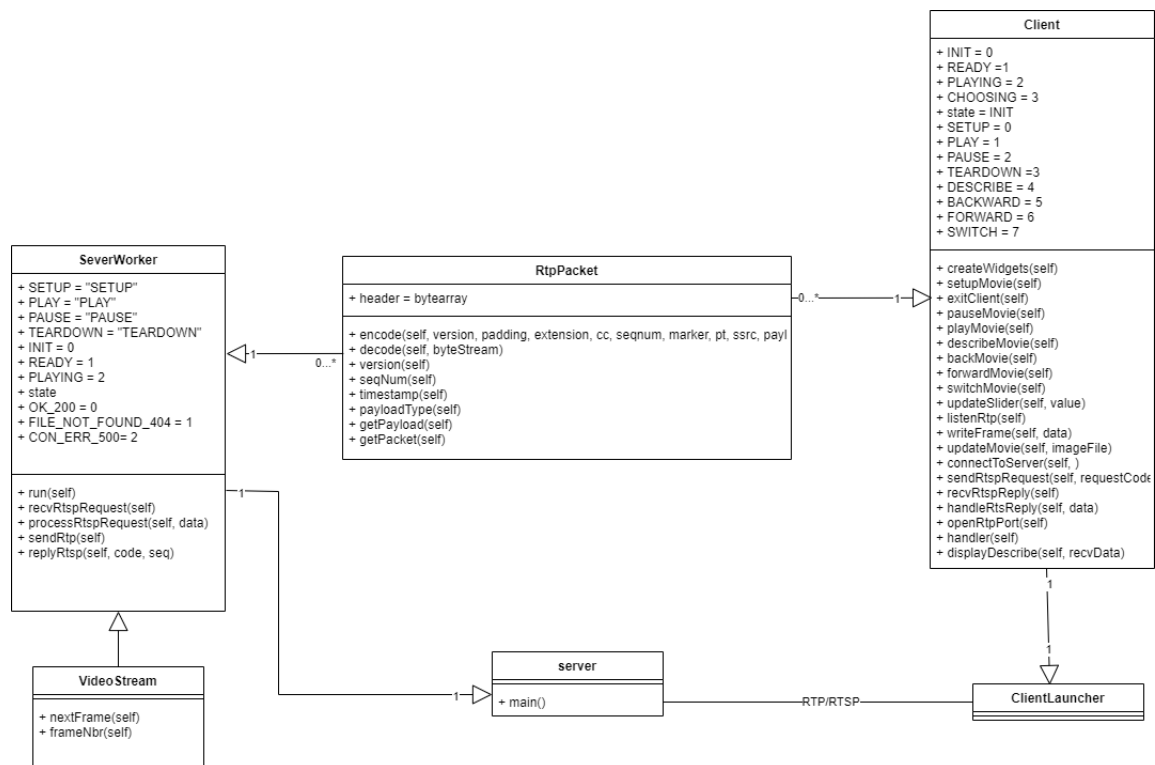
Yêu cầu BACKWARD : người dùng nhấn vào nút BACKWARD ở trên giao diện của sổ

- Client \rightarrow Server: Client gửi yêu cầu BACKWARD chứa phiên bản RTSP, trình tự và session của server
- Server \rightarrow Client: Server sẽ in quá trình xử lý FORWARD ở terminal và gửi nội dung phản hồi chứa phiên bản RTSP, trình tự, session. Khung hiển thị video sẽ được lùi lại khi yêu cầu được xử lý

Yêu cầu SWITCH : người dùng nhấn vào nút SWITCH ở trên giao diện của sổ

- Client \rightarrow Server: Client gửi yêu cầu SWITCH chứa phiên bản RTSP, trình tự và session của server
- Server \rightarrow Client: Server sẽ in quá trình xử lý FORWARD ở terminal và gửi nội dung phản hồi chứa phiên bản RTSP, trình tự, session và danh sách video. Quá trình lựa chọn sẽ do client xử lý, sau đó client sẽ gửi yêu cầu PLAY hay SETUP

6 Class diagram



Hình 1: sơ đồ class diagram của video streaming

7 Hiện thực

Sau khi triển khai giao diện người dùng của yêu cầu RTSP, chúng ta đã đạt được như sau:



Hình 2: Cửa sổ giao diện ứng dụng

Với ứng dụng cơ bản chỉ với 4 lệnh: set up, play, pause and tear dow, chúng ta có thể thực hiện các tác vụ sau:

- Bằng cách nhấp vào nút SETUP, nếu Client ở trạng thái ban đầu, một yêu cầu sẽ được gửi đến ổ cắm RTSP với thông báo sau được Server tiếp nhận:

```
Data received:
SETUP movie.Mjpeg RSTP/1.0
CSeq: 1
Transport: RTP/UDP; client_port= 25000
processing SETUP
```

Hình 3: Nhận yêu cầu SETUP

- Và sau đó máy chủ sẽ trả lời xem thông báo có được thực thi thành công hay không:

```
Data sent:
RTSP/1.0 200 OK
CSeq: 1
Session: 257067
Totalframe: 500
```

Hình 4: Phản hồi yêu cầu SETUP

Trong trạng thái này, Client sẽ mở một ổ cắm RTP mới, liên kết ổ cắm với địa chỉ máy chủ được cung cấp và cổng RTP, sau đó thay đổi trạng thái thành sẵn sàng. Server worker sẽ xử lý thông báo này bằng cách mở tên tệp phim trong lớp luồng video.

- Khi bộ phim đã được thiết lập, nhấp vào nút PLAY sẽ bắt đầu một chuỗi mới để lắng nghe gói RTP và sau đó gửi yêu cầu RTSP:

```
Data received:
PLAY movie.Mjpeg RSTP/1.0
CSeq: 2
Session: 257067
processing PLAY
```

Hình 5: Nhận yêu cầu PLAY

Server trả lời và bắt đầu gửi khung phim qua cổng RTP tới Client. Client bây giờ nhận được gói này và cập nhật khung hình video

- Trong khi phim đang phát, nhấp vào nút PAUSE sẽ làm cho chuỗi đang nghe gói RTP ngừng nhận khung phim và server cũng sẽ dừng quá trình gửi gói RTP

```
Data received:
PAUSE movie.Mjpeg RSTP/1.0
CSeq: 4
Session: 257067
processing PAUSE
```

Hình 6: Nhận yêu cầu PAUSE

- Nhấp vào nút TEARDOWN sẽ đóng tất cả các cổng và xóa hình ảnh bộ nhớ cache, sau đó kết thúc session:

```
Data received:  
TEARDOWN movie.Mjpeg RSTP/1.0  
CSeq: 5  
Session: 257067  
processing TEARDOWN
```

Hình 7: Nhận yêu cầu TEARDOWN

8 Đánh giá tổng thể các kết quả đạt được

Sau khi đạt được các mục tiêu chính của chúng ta là thực hiện một streaming video server và client giao tiếp bằng Real-Time Streaming Protocol (RTSP) và gửi dữ liệu bằng Real-time Transfer Protocol (RTP), ta đã hiểu sơ qua kiến thức về RTSP, RTP, cách hoạt động bình thường của video streaming. Không chỉ vậy, bằng cách hoàn thành phần mở rộng (3/4) cho bài tập mà ta đã hiểu khái niệm về mất gói thông qua các loại kết nối khác nhau. Chúng ta có cơ hội mở rộng kỹ năng kỹ thuật của mình bằng cách triển khai giao diện, lớp và nhiều chức năng cần thiết để hoàn thành assignment này.

9 Hướng dẫn sử dụng

Trong phần này, ta sẽ trình bày cách sử dụng ứng dụng của phép gán và cách chạy nó. Trước khi bắt đầu, trước tiên bạn cần cài đặt thư viện PIL vì một số phiên bản python không hỗ trợ thư viện này.

```
pip install pillow
```

Đầu tiên, mở 2 terminal (Client và Server). Chạy Server.py trên Server Terminal để bắt đầu server:

```
python Server.py server-port
```

Trong đó, Server-port là nơi server nhận kết nối RTSP đến. Tiêu chuẩn của Cổng RTS là 554, nhưng chúng ta sẽ cần chọn một số cổng tốt hơn 1024. Ví dụ:

```
python Server.py 1300
```



Sau đó, chạy ClientLauncher.py trên Client Terminal để bắt đầu client:

`python ClientLauncher.py server-host server-port RTP-port video-file`

Địa chỉ:

- Server-host là địa chỉ IP của máy cục bộ(e.g "127.0.0.1")
- Server-port là cổng nơi máy chủ tiếp nhận yêu cầu.
- RTP-port là cổng nhận gói RTP
- video-file là yêu cầu tên tệp tới máy chủ (loại Mjpeg) Như một ví dụ tương ứng với Server's terminal command là:

`python ClientLauncher.py 127.0.0.1 1300 5008 movie.Mjpeg`

Sau đó, một cửa sổ bật lên xuất hiện với các nút trong ứng dụng video streaming này

- SETUP: Để phát video, nhấn nút SETUP để khởi chạy
- PAUSE: Dừng video
- PLAY: Phát / Tiếp tục video
- TEARDOWN: Dừng video và đóng cửa sổ giao diện

Trong phiên bản mở rộng, giai đoạn khởi tạo sẽ tự động bắt đầu khi client và server được kết nối và video sẽ được thiết lập. Cửa sổ sẽ có thêm các nút sau:

- DESCRIBE: Gửi một câu trả lời mô tả cho khách hàng bao gồm loại giao thức luồng trong, session, mã hóa nào được sử dụng, fps
- FORWARD: Chuyển tiếp video trong 1/5 thời lượng của video
- BACKWARD: Lùi lại video cho 1/5 thời lượng của video
- SWITCH: cho người dùng đổi video bằng danh sách tự tạo

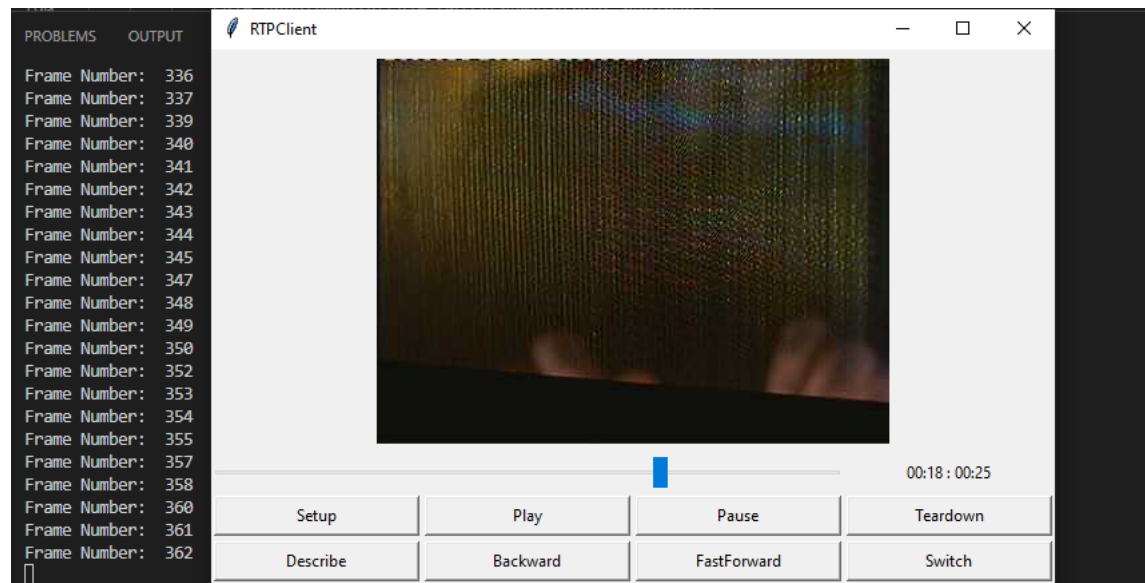
Trong phiên bản này, người dùng chỉ có thể đóng cửa sổ bằng cách nhấp vào nút 'X' ở trên cùng bên phải của cửa sổ và xác nhận yêu cầu trên hộp thông báo để kết thúc session.

10 Extend

10.1 Extend 1

Chúng ta tính toán thống kê về phần này dựa trên một số thông tin có được khi hiện thực code

Đối với tỉ lệ mất gói RTP , ta sẽ xác định số gói bị mất ghi chạy chương trình thông qua việc in số thứ tự của frame trên cổng terminal của Client khi Client nhận được gói tin chứa khung hình đó. Khi ta chạy thử , tỉ lệ mất gói trong trường hợp này là bằng 0% (nghĩa là không thiếu bất kỳ khung hình nào).



Hình 8: Client hiển thị thứ tự của frame trên cổng terminal khi nhận được gói tin

Ngoài ra , ta dùng thư viện opencv xác định được FPS (tổng số khung hình trên 1 giây) là xấp xỉ khoảng 20 FPS

10.2 Extend 2

Trong yêu cầu này, giao diện người dùng chỉ có 3 nút là PLAY, PAUSE, STOP(không có nút SETUP). Chức năng STOP cũng tương tự như TEARDOWN ở trong project chính của chúng ta. Tuy nhiên, nó khởi động lại video từ đầu thay vì kết thúc.

- Đầu tiên, ta cần xóa cài đặt thủ công video. Trong quá trình triển khai, chúng ta sử dụng lại code đã viết cho yêu cầu SETUP. Thay vì thiết lập video trên command, chúng ta có thể gọi nó bất cứ lúc nào code khởi động (cụ thể là trong hàm khởi tạo của lớp Client2). Điều đó sẽ đảm bảo video được thiết lập khi chúng ta khởi động ứng dụng.
- Để thực hiện chức năng STOP, chúng ta cần hiểu chức năng của nó. Khi người dùng bấm nút STOP, video sẽ tạm dừng và được tua lại. Khi người dùng nhận PLAY, video sẽ phát lại video từ đầu thay vì phát tiếp tục từ chỗ cũ. Vì vậy, không thể dùng TEARDOWN để thay cho STOP vì TEARDOWN sẽ đóng hoàn toàn cửa sổ, dẫn tới không thể tiến hành

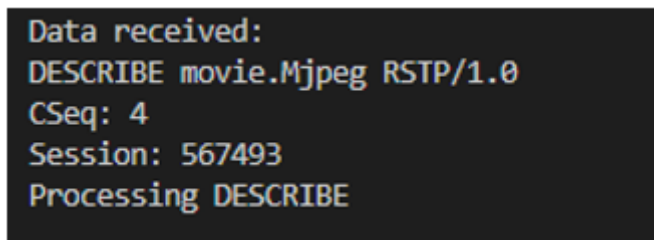
phát lại video. Ý tưởng chính là gửi yêu cầu STOP đến Sever, đặt lại số khung hình tại thời điểm khi STOP được nhấn cũng như tệp nhị phân khi Sever đọc video. Sever sẽ xử lý STOP theo các bước sau:

1. Nhận yêu cầu STOP từ Client
2. Đặt lại khung hình của video
3. Đặt lại tệp nhị phân khi Sever đọc video trong quá trình SETUP
4. Thay đổi trạng thái thành READY

10.3 Extend 3

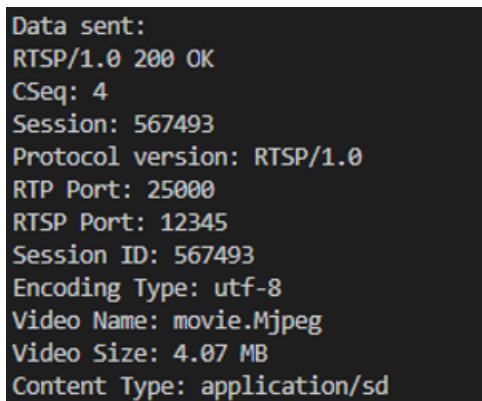
Trong yêu cầu này, ta sẽ xử lý phương thức DESCRIBE được sử dụng để đưa thông về media stream. Khi máy chủ nhận được yêu cầu DESCRIBE, nó sẽ gửi lại thông tin về video stream. Để hiện thực DESCRIBE, ta thực hiện theo những bước sau:

- Tạo phương thức DESCRIBE từ Client được gửi đến Server. Yêu cầu được thực hiện chỉ khi trạng thái của client là READY. Yêu cầu được tạo với cấu trúc: DESCRIBE + file name + Transport protocol + RTSP Sequence number
- Sau khi nhận được yêu cầu DESCRIBE. Server sẽ xử lý yêu cầu và trả lời Client.
- Sau khi Client nhận được thông tin từ Server, Client sẽ cập nhật trạng thái.



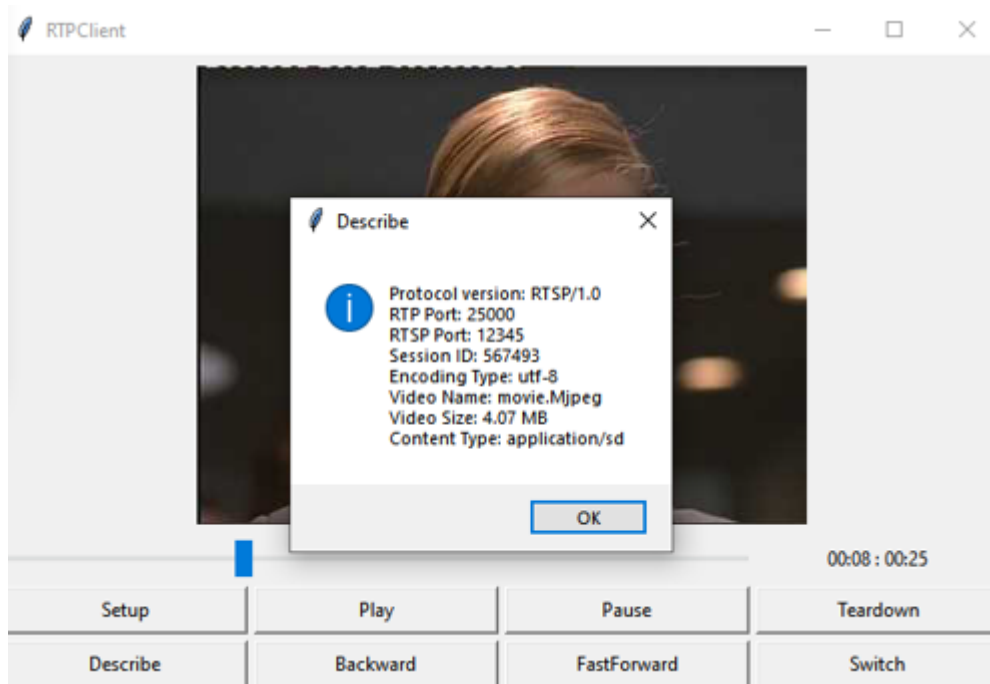
```
Data received:
DESCRIBE movie.Mjpeg RSTP/1.0
CSeq: 4
Session: 567493
Processing DESCRIBE
```

Hình 9: Nhận yêu cầu DESCRIBE



```
Data sent:
RTSP/1.0 200 OK
CSeq: 4
Session: 567493
Protocol version: RTSP/1.0
RTP Port: 25000
RTSP Port: 12345
Session ID: 567493
Encoding Type: utf-8
Video Name: movie.Mjpeg
Video Size: 4.07 MB
Content Type: application/sd
```

Hình 10: Phản hồi yêu cầu DESCRIBE



Hình 11: Mô tả cửa sổ DESCRIBE

10.4 Extend 4

Khi người dùng click vào nút BACK/FORWARD thì client sẽ gửi yêu cầu BACK/FORWARD đến phía server. Server nhận được yêu cầu sẽ thay đổi số thứ tự frame để gửi đi, và đồng thời phản hồi trạng thái kèm với số thứ tự của frame mà Client sẽ nhận được. Bên client sẽ nhận được phản hồi và chờ các gói frame từ đó trở đi.

- Khi người dùng tua video, thời gian mặc định mà ta đặt cho mỗi lần tua là $1/5$ thời lượng của video. Vì vậy, mỗi lần nhấn nút FORWARD, số khung hình video tại thời điểm đó sẽ được cộng với số lượng khung hình thêm vào là $1/5$ tổng frame. Hơn nữa khi đến cuối video, nghĩa là số khung hình còn lại của video nhỏ hơn $1/5$ tổng số khung hình, Máy chủ sẽ gửi khung hình cuối cùng đến máy khách và đặt số khung hình của máy khách thành tổng số khung hình của video.

```
def forward(self):
    ammount = int(self.frameNum/5)
    if (self.currentFrame > self.frameNum-1-ammount):
        self.currentFrame = self.frameNum-1
    else:
        self.currentFrame += ammount
```


- Tương tự như vậy với quay lại, mỗi lần nhấn nút BACKWARD, số khung hình video tại thời điểm đó sẽ bị trừ đi lượng khung hình ($1/5$ độ dài của video). Hơn nữa khi video ở đầu, có nghĩa là khung hình mà video đã được phát nhỏ hơn $1/5$ tổng số khung hình, Máy chủ sẽ gửi khung hình đầu tiên đến máy khách và đặt số khung hình của máy khách bằng 0

```
def back(self):  
    ammount = int(self.frameNum/5)  
    if (self.currentFrame < ammount):  
        self.currentFrame = 0  
    else:  
        self.currentFrame -= ammount
```

10.5 Extend 5

SWITCH:

- Gửi yêu cầu switch, client sau khi gửi gói RTSP chứa yêu cầu switch sẽ tiến vào trạng thái CHOOSING - tạm dừng video đang chơi nếu có.
- Sau khi nhận gói RTSP chứa switch, server tiến hành tìm kiếm tất cả file video đạt yêu cầu trong thư mục mã nguồn, sau đó bỏ tất cả tên các file ấy vào dòng cuối cùng của gói RTSP phản hồi.

```
if (self.request == self.SWITCH):  
    fileList = [f for f in os.listdir() if f.endswith('.Mjpeg')]  
    reply += '\n' + ' '.join(fileList)
```

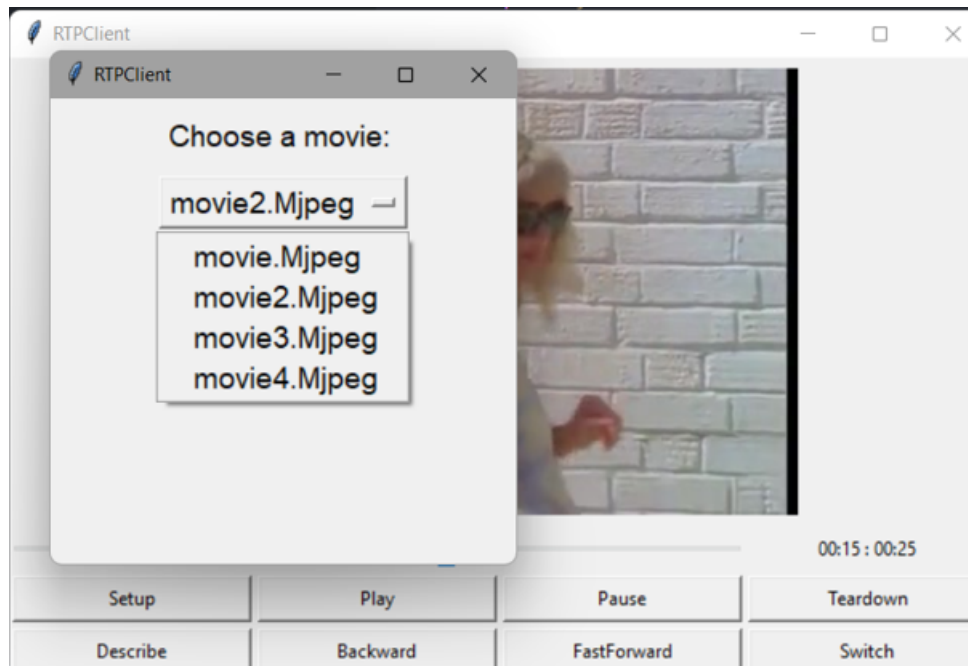
Ví dụ về gói RTSP phản hồi:

```
Data sent:  
RTSP/1.0 200 OK  
CSeq: 6  
Session: 350623  
movie.Mjpeg movie2.Mjpeg movie3.Mjpeg movie4.Mjpeg
```

- Sau khi nhận được danh sách video, client sẽ tạo một cửa sổ mới. Ở đây ta có hai lựa chọn, nếu ta chọn một video bất kì trong danh sách, chương trình sẽ chuyển video đang phát về

video mới, nếu ta chọn đóng cửa sổ thì chương trình giữ nguyên. Cả hai lựa chọn đều dẫn chương trình về state READY, nhấn PLAY để tiếp tục phát.

Ví dụ về cửa sổ lựa chọn:





References

- [1] https://en.wikipedia.org/wiki/Real_Time_Streaming_Protocol
- [2] <https://www.techopedia.com/definition/4755/real-time-transport-protocol-rtp>
- [3] <https://www.codementor.io/@erikeidt/bit-twiddling-understanding-bit-operations-ij68ynb7>
- [4] <https://www.cl.cam.ac.uk/~jac22/books/mm/book/node159.html>
- [5] <https://dauso1900.vn/rtsp-la-gi/>
- [6] <https://stackoverflow.com/questions/62041552/creating-rtp-packet-with-python>