

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



COMPUTER NETWORKS (CO3094)

Assignment 1: Video Streaming Networking

L04 - Group Hoc tro thay Jim Kurose

GVHD: Bùi Xuân Giang

Thành viên: Đỗ Đình Phú Quý - 2014289
Lê Thanh Tân - 2014451
Chu Đình Chiến - 2012727
Nguyễn Việt Khánh Trình - 1912303

Thành phố Hồ Chí Minh, tháng 6 năm 2022

Mục lục

1	Analysis of problem requirments	2
1.1	Kỹ Thuật streaming Video	2
1.2	Giao thức RTSP	2
1.3	Giao thức RTP	3
1.4	Đánh giá yêu cầu	3
2	List of component	3
3	Model and data flow	4
4	Class diagram	6
5	Description of defferent functions for your application	7
6	Implementation	7
7	A summative evaluation of achieved results	8
8	User manual	8
9	Extend	10
9.1	Extend 1	10
9.2	Extend 2	11
9.3	Extend 3	11
9.4	Extend 4	13

1 Analysis of problem requirements

Nội dung: nói về hai giao thức, cách thức chúng tương tác với nhau

Trong bài tập lớn này, sinh viên sẽ thực hiện một ứng dụng streaming video thông qua hai giao thức đó là RTSP/RTP

1.1 Kỹ Thuật streaming Video

- Streaming Video là một kỹ thuật được sử dụng khá phổ biến trong các ứng dụng mạng. Được sử dụng rộng rãi trong thực tế như các phần mềm trình phát phương diện, trình duyệt web theo mô hình Client/Server. Trong đó các máy Client truy cập và xem video từ các máy Server.
- Streaming Video sử dụng cách thức phát video từ máy chủ tới người dùng đầu cuối muốn xem video mà không cần tải đoạn video đó về. Về bản chất, nó là quá trình chia nhỏ file video thành các frame, lần lượt gửi tới bộ đệm trên máy tính người dùng và hiển thị nội dung của frame đó. Quá trình này tuân thủ chặt chẽ về ràng buộc theo thời gian, nói cách khác là theo hai giao thức RTSP/RTP.

Hiện nay, người ta chia streaming video làm hai loại lớn đó là:

- Live streaming: chuyển phát video trực tiếp trong thời gian thực đến người xem.
- Video on Demand (VoD): chuyển phát các video được lưu trữ sẵn trên máy chủ và sẵn sàng hiện thực khi người dùng truy cập. Có thể thực hiện thao tác: STOP, PAUSE,... Đây chính là kiểu streaming video được sử dụng trong bài tập lớn này.

Các bước thực hiện:

- Phía bên client cần kết nối và xác định file video cần xem trên server.
- Client gửi Request đến server để tìm file video đó, sau đó là thiết lập kết nối cần cho sự truyền tải video.
- Server sẽ chia file video thành các frame rồi gửi đến client theo các giao thức ràng buộc về thời gian (RTSP/RTP) sau khi nhận Request yêu cầu xem từ phía người dùng.
- Khi các frame đến được client, chúng sẽ được giải mã và hiển thị thông qua chương trình phát video.

1.2 Giao thức RTSP

- RTSP (Real Time Streaming Protocol) là giao thức truyền phát thời gian thực. Đây là một giao thức cung cấp khung để truyền dữ liệu phương tiện theo thời gian thực ở tầng 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. Nó không truyền phát đa phương tiện mà giao tiếp với máy chủ truyền dữ liệu đa phương tiện. Ví dụ, khi người dùng tạm dừng video mà anh ta đang phát trực tuyến, RTSP sẽ chuyển yêu cầu của người dùng để tạm dừng video đến máy chủ phát video.
- RTSP sử dụng TCP là giao thức để duy trì một kết nối đầu cuối tới đầu cuối và các thông điệp điều khiển của RTSP được gửi bởi máy client tới máy server. Nó cũng thực hiện điều khiển lại các đáp trả từ máy server tới máy client. Cổng mặc định được sử dụng bởi giao thức này là 554. Để thực hiện kỹ thuật streaming video theo giao thức RTSP nhất thiết máy client phải gửi lên máy server (streaming server) những Request và phải theo một trình tự nhất định.

1.3 Giao thức RTP

- Real time transfer protocol (RTP) là giao thức truyền tải thời gian thực, dùng để chuyển tập tin, video, âm thanh qua mạng IP. RTP được sử dụng rộng rãi trong các hệ thống truyền thông và giải trí liên quan đến các streaming media như gọi điện thoại, các ứng dụng hội nghị truyền hình, các dịch vụ truyền hình và các tính năng push-to-talk dựa trên nền web.
- RTP chạy trên giao thức UDP (User Datagram Protocol), RTP được sử dụng kết hợp với RTP Control Protocol (RTCP). Trong khi RTP mang các luồng truyền thông (ví dụ: âm thanh và video), RTCP được sử dụng để giám sát số liệu truyền tải và chất lượng dịch vụ (QoS) và đồng bộ hóa nhiều luồng. RTP là một trong những cơ sở kỹ thuật của Voice over IP và trong ngữ cảnh này thường được sử dụng kết hợp với một giao thức báo hiệu như Session Initiation Protocol (SIP) thiết lập kết nối qua mạng.

1.4 Đánh giá yêu cầu

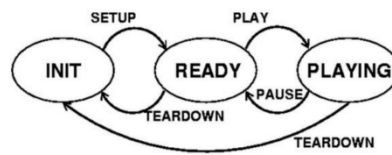
- Một cách hình dung đơn giản cho việc streaming video thông qua hai giao thức RTSP/RTP đó là RTSP thực hiện vai trò giao tiếp và điều khiển, còn RTP thực hiện truyền tải video từ server đến client. Ví dụ để tiến hành bắt đầu phát video, chúng ta sử dụng RTSP để gửi Request PLAY đến server và yêu cầu server tiến hành chuyển gói tin, server response chấp nhận đến client, sau đó server thực hiện chuyển video bằng RTP.
- Cụ thể trong bài tập lớn này, client sẽ dùng RTSP để yêu cầu kết nối đến server, sau đó là giao tiếp như SETUP, PLAY, PAUSE, TEARDOWN. SETUP tương ứng với lệnh thiết lập kết nối (ở đây là thiết lập để có kết nối RTP). Sau lệnh PLAY, PAUSE, TEARDOWN thì server sẽ thực hiện tương ứng việc truyền, tạm dừng và dừng lại việc phát video thông qua giao thức RTP. Chúng ta cần hiện thực ở phía client để có thể giao tiếp được với server (server đã được hiện thực) và thực hiện tạo gói tin theo đúng chuẩn theo giao thức RTP để server chuyển sang client.

2 List of component

Ứng dụng streaming video có 3 thành phần chính:

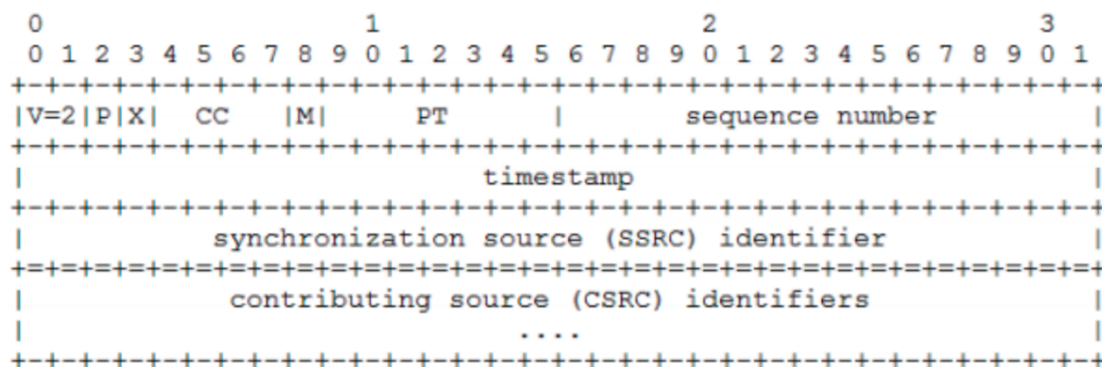
- Phía streaming Server
 - Phía Client
 - Gói tin RTP
- Ở streaming server chúng ta cần thiết lập đầy đủ các phương thức phục vụ cho việc giao tiếp với client ở giao thức RTSP. Đầu tiên, server sẽ thực hiện mở kết nối RTSP, sẵn sàng để client có thể kết nối đến. Khi server nhận được Request từ client, server sẽ phải phân tích dữ liệu nhận được để đồng bộ trạng thái với bên client, thực hiện xử lý đúng các bước rồi sau đó phải hồi cho client. Lần đầu tiên client gửi Request SETUP, server sẽ phải tìm file được yêu cầu, tạo session id (phục vụ cho việc xác nhận giao tiếp) rồi gửi về cho client. Tiếp đến, khi nhận Request PLAY, server tạo socket RTP và gửi gói tin RTP chứa các frame Video về cho client. Nhận Request PAUSE, TEARDOWN thì server tương ứng tạm dừng và ngắt gửi gói tin RTP. Và mỗi khi nhận RTSP Request thì server đều phải reply lại tương ứng.
 - Phía client, khi khởi tạo sẽ mở socket RTSP, sau đó thực hiện shakehand đến streaming server, mỗi lần gửi request, client phải chuẩn bị đầy đủ các thông tin. Khi gửi SETUP thì phải có RTP port để server biết được cổng nhận frame video bên phía client. Và sau khi nhận phản hồi, client sẽ lấy được session id từ streaming server gửi đến, session id sẽ dùng để kiểm tra phản hồi có đúng từ streaming server gửi đến hay không. Đồng thời client sẽ mở socket để nhận RTP từ phía server. Ở các lần Request tiếp theo, PLAY, PAUSE, TEARDOWN thì client chỉ cần gửi đúng thông tin, cập nhập trạng thái sau khi nhận phản hồi, phía server sẽ thực hiện yêu cầu thực hiện phát, dừng, ngắt việc chuyển gói tin RTP. Khi RTP phía client nhận được gói tin RTP sẽ thực hiện giải mã, lấy nội dung payload đã đối chiếu đúng thứ tự frame (chỉ cần frame gói tin lớn hơn frame đã có ở client), rồi tiến hành đẩy vào cache file rồi cập nhập image file để hiển thị ra cửa sổ trình phát video.

- Để việc trao đổi giao tiếp giữa client và server diễn ra đúng yêu cầu thì cả hai phía phải duy trì trạng thái hợp lệ sau mỗi lần gửi nhận Request như sau:



Hình 1: Trạng thái hoạt động.

Đối tượng chính được giao tiếp qua mạng đó là RTP packet, mỗi gói tin của video được yêu cầu chặt chẽ về cấu trúc header như sau:



Hình 2: Cấu trúc header của RTP packet.

- Version (V): 2 bits, xác định phiên bản của RTP
- Padding (P): 1 bit, nếu padding là 1, RTP sẽ có 1 hoặc nhiều padding được thêm vào cuối gói tin, nếu là 0 thì sẽ không có padding nào.
- Extension (X): 1 bit, nếu là 1 thì phần header cố định sẽ kết nối với phần header mở rộng, ngược lại thì không
- CSRC count (CC): 4 bits, chứa các giá trị của trường CSRC ID trong header cố định
- Marker (M): 1 bit, được sử dụng ở lớp Application để xác định 1 profile
- Payload type (PT): 7 bits, xác định dạng payload của RTP (PCM(0), MJPEG (26), GSM(3), LPC(7),...)
- Sequence number: 16 bits, mang số thứ tự của gói tin RTP
- Timestamp: 32 bits, xác định thời điểm lấy mẫu của octets đầu tiên trong gói tin.
- SSRC: 32 bits, giá trị được chọn ngẫu nhiên.
- CSRC list: 32 bits, xác định các nguồn đóng góp payload cho gói tin

Để có thể chuyển đi qua giao thức RTP, chúng ta phải hiện thực các frame video thành các gói tin RTP rồi đóng gói cho server chuyển đi, phía client nhận được sẽ dùng để lấy frame video.

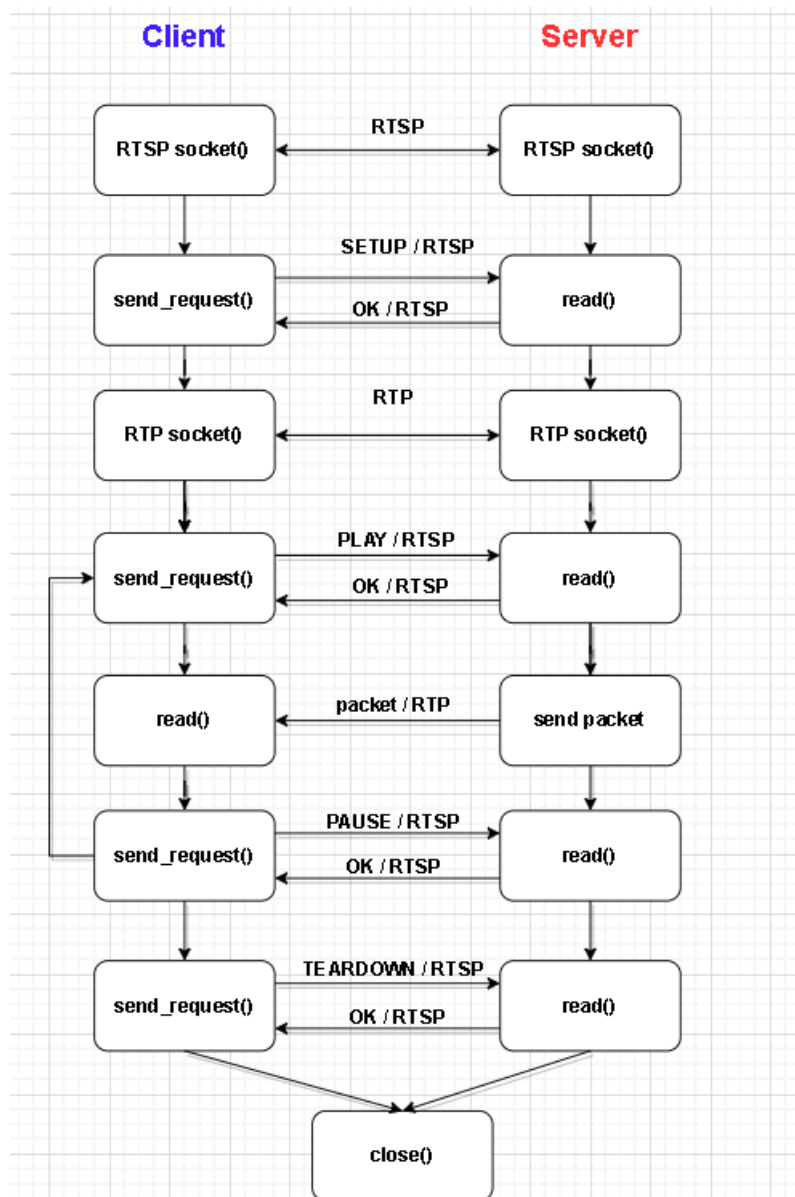
3 Model and data flow

- Server tạo 1 socket RTSP/TCP và lắng nghe các kết nối phản hồi.
- Client tạo 1 socket cùng địa chỉ và port với Server, kết nối giữa Server và Client được hình thành.

- Nếu Request là SETUP thì server sẽ tạo session cho RTP packet, tạo VideoStream, lấy RTP port của Client gửi qua và phản hồi lại cho Client (**OK_200**: thành công, **404 NOT FOUND**: nếu file movie không tìm thấy, **500 CONNECTION ERROR**: nếu kết nối bị lỗi). Client sẽ tạo RTP socket (hàm openRtpPort) cho Server kết nối vào khi Request là PLAY
- Nếu Request là PLAY thì server sẽ tạo RTP socket, RTP packet. Client tạo 1 Thread để nhận các RTP packet, xử lý nó và chiếu lên màn hình. Quá trình gửi RTP packet đến cho Client diễn ra liên tục cho đến khi Request là PAUSE hoặc TEARDOWN thì Server mới dừng việc gửi RTP packet. Khi Client nhận RTP packet chứa movie và ghi nó vào 1 file jpg
- Nếu Request là TEARDOWN thì Server và Client sẽ đóng RTSP socket và kết thúc

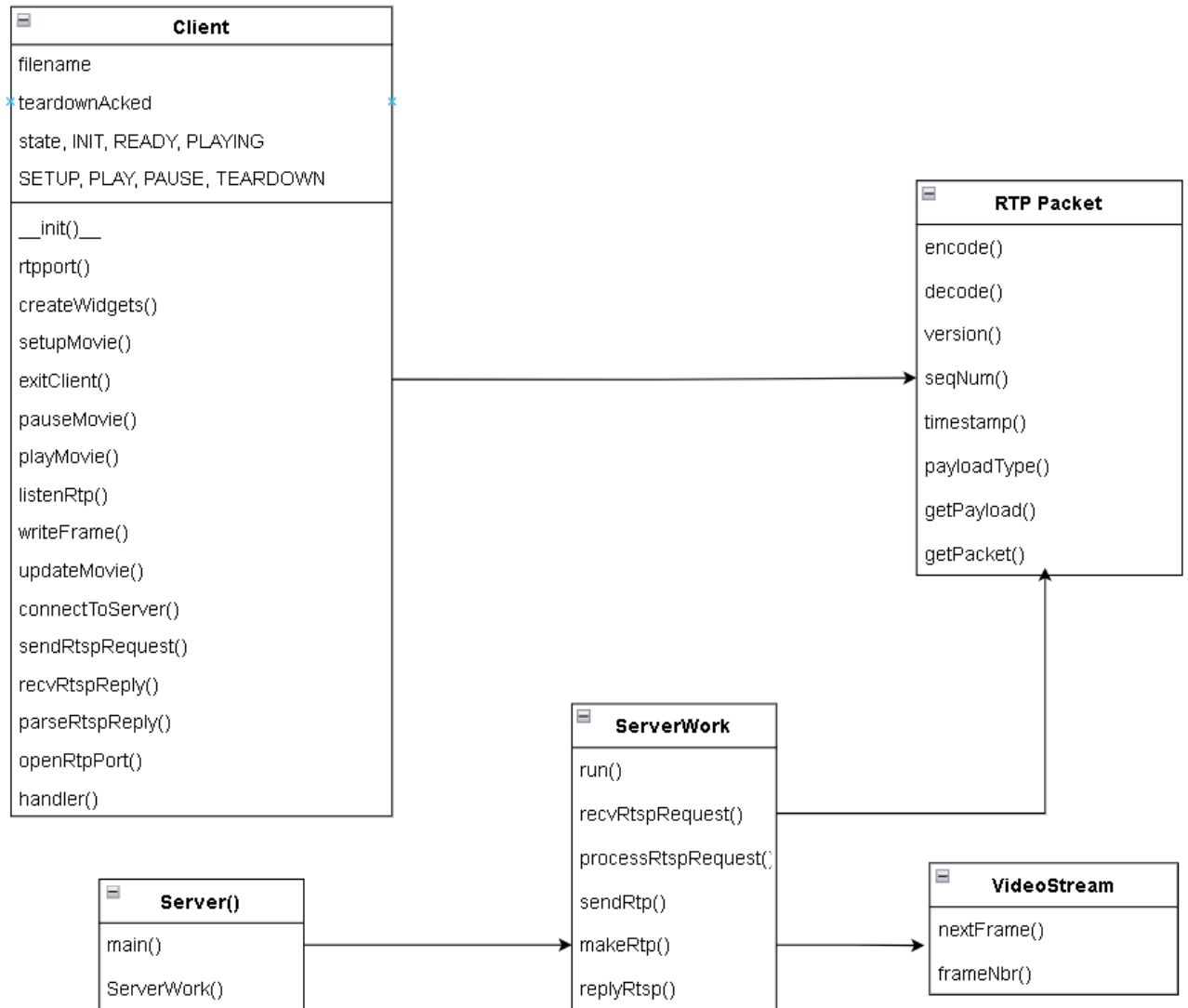
Giao thức RTSP dùng để gửi các Request từ Client đến Server

Giao thức RTP dùng để gửi RTP packet từ Server đến Client



Hình 3: Mô hình hoạt động giữa Server và Client.

4 Class diagram



Hình 4: Class Diagram.

- ServerWorker sử dụng đối tượng của lớp RtpPacket để mã hóa dữ liệu video rồi gửi cho client.
- Client nhận RtpPacket, phân tích rồi chiếu lên màn hình.
- Client và Server giao tiếp với nhau thông qua giao thức RTSP
- Packet được gửi thông qua giao thức RTP

5 Description of defferent functions for your application

Class Name	Function	Parameter	Description
SeverWorker	__init__(self, clientInfo)	self, clientInfo	Constructor
	run(self)	self	Run the server
	processRtspRequest(self, data)	self, data	Process the Rtsp request
	sendRtp(self)	self	Send RTP packets over UDP
	makeRtp(self, payload, frameNbr)	self, payload, frameNbr	RPT-packetize the video data
	replyRtsp(self, code, seq)	self, code, seq	Send RTSP reply to the Client
Sever	main(self)	self	Main function to run the whole program.
VideoStream	__init__(self, filename)	self, filename	Constructor
	nextFrame(self)	self	Get next frame
	frameNbr(self)	self	Get frame number
Client	__init__(self, master, serveraddr, serverport, rtpport, filename)	self, master, serveraddr, serverport, rtpport, filename	Constructor
	createWidgets(self)	self	Build GUI
	setupMovie(self)	self	Setup button handler
	exitClient(self)	self	Teardown button handler
	pauseMovie(self)	self	Pause button handler
	playMovie(self)	self	Play button handler
	take_time(self, buftime)	self, buftime	change time to format minutes : seconds
	listenRtp(self)	self	Listen for RTP packets and analysis somethings
	writeFrame(self, data)	self, data	Write the received frame to a temp image file
	updateMovie(self, imageFile)	self, imageFile	Update the image file as video frame in the GUI
	connectToServer(self)	self	Connect to the Server. Start a new RTSP/TCP session
	sendRtspRequest(self, requestCode)	self, requestCode	Send RTSP request to the server
	recvRtspReply(self)	self	Receive RTSP reply from the server
	parseRtspReply(self, data)	self, data	Parse the RTSP reply from the server
	openRtpPort(self)	self	Open RTP socket binded to a specified port
	handler(self)	self	Handler on explicitly closing the GUI window
RtpPacket	__init__(self)	self	constructor
	encode(self, version, padding, extension, cc, seqnum, marker, pt, ssrc, payload)	self, version, padding, extension, cc, seqnum, marker, pt, ssrc, payload	Encode the RTP packet with header fields and payload
	decode(self, byteStream)	self, byteStream	Decode the RTP packet
	version(self)	self	Return RTP version
	seqNum(self)	self	Return sequence (frame) number
	timestamp(self)	self	Return timestamp
	payloadType(self)	self	Return payload type
	getPayload(self)	self	Return payload
	getPacket(self)	self	Return RTP packet

Hình 5: Danh sách các hàm được sử dụng trong bài

6 Implementation

Phần hiện thực được thực hiện ở link sau:

<https://github.com/tanlethanh/Assignment-Computer-Network.git>

7 A summative evaluation of achieved results

- Qua nội dung của đề bài, nhóm đã thực hiện thành công ứng dụng streaming video từ Server sang Client sử dụng giao thức RTSP và RTP. Dựa trên phần code được chuẩn bị sẵn từ đề bài, các hàm cần thiết đã được hiện thực đúng với yêu cầu đề bài đưa ra.
- Tổng kết từ nội dung của bài tập lớn đã giúp cho sinh viên hiểu hơn về giao thức, truyền tải nội dung. Hiểu về hai giao thức quan trọng là RTSP và RTP trong việc ứng dụng vào truyền tải video từ Server sang Client. Cũng như có sự thực hành để hiểu rõ cơ chế, cú pháp ngữ nghĩa khi sử dụng hai giao thức này.

8 User manual

Mở 2 terminal chạy song song nhau. Terminal 1 chạy lệnh:

```
#python Server.py server_port
```

Server_port là cổng port mà server chờ kết nối RTSP. Cổng RTSP tiêu chuẩn là 554, nhưng bạn nên chọn số cổng lớn hơn 1024.



```
TERMINAL PROBLEMS 79 OUTPUT GITLENS DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

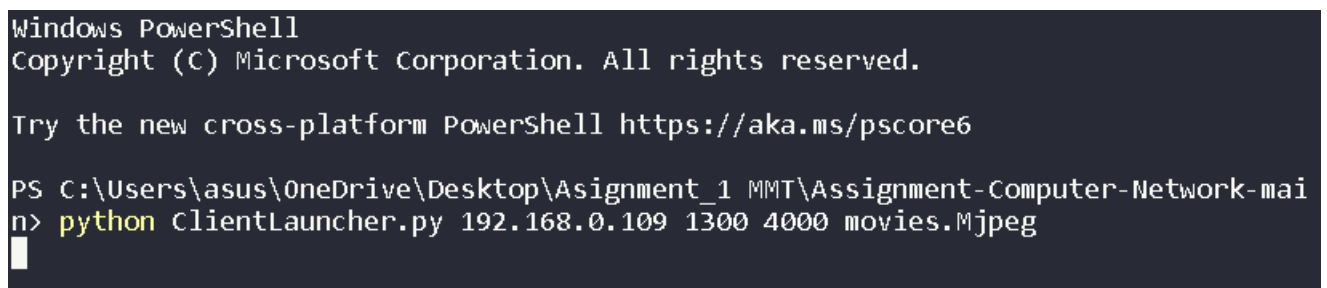
PS C:\Users\asus\OneDrive\Desktop\Assignment_1 MMT\Assignment-Computer-Network-main
> python Server.py 1300
█
```

Hình 6: Run Server.py.

Terminal 2 chạy lệnh:

```
#python ClientLauncher.py server_host server_port RTP_port video_file
```

- Server_host: tên máy tính của bạn, hoặc địa chỉ IP của máy bạn.
- Server_port: tên cổng port bạn đã nhập ở terminal 1.
- RTP_port: là cổng mà RTP packet được nhận.
- Video_file: là tên video bạn muốn chiếu.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\asus\OneDrive\Desktop\Assignment_1 MMT\Assignment-Computer-Network-main
> python ClientLauncher.py 192.168.0.109 1300 4000 movies.Mjpeg
█
```

Hình 7: Run ClientLauncher.py.

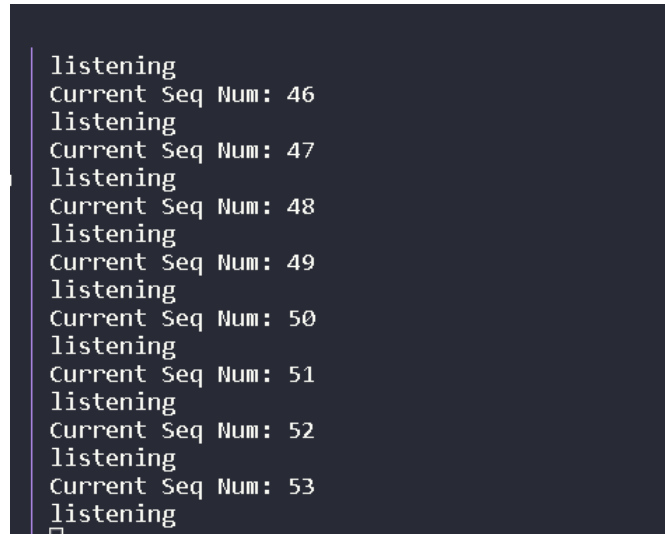
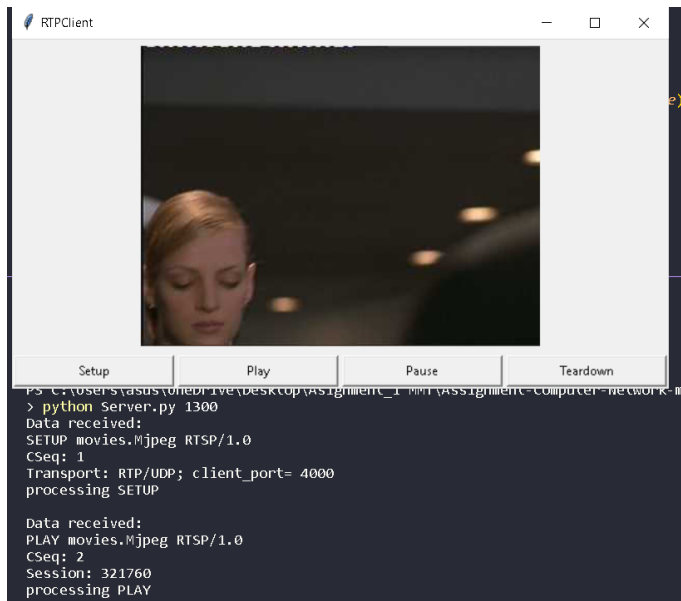
Sau khi chạy sẽ xuất hiện màn hình sau:



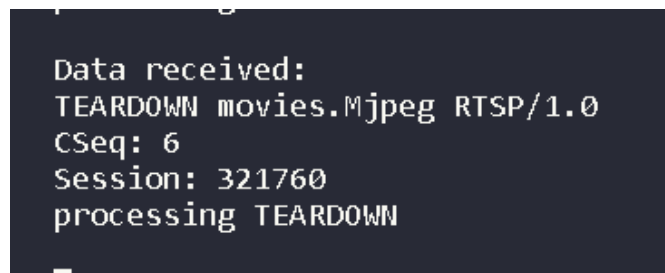
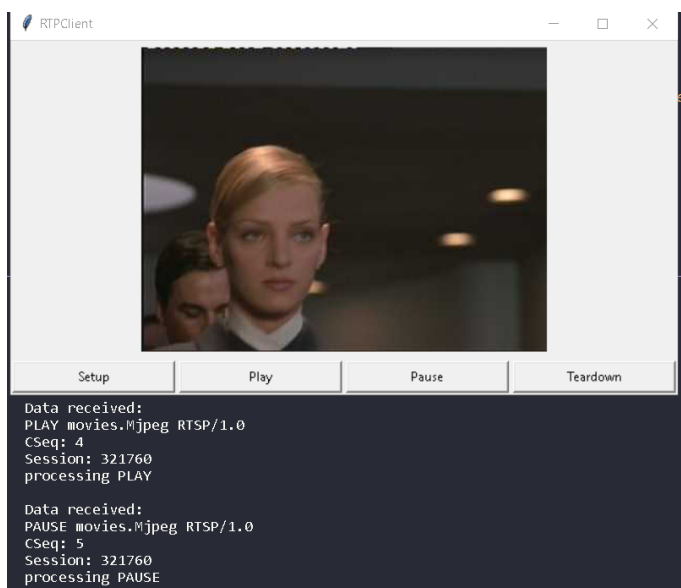
Click SETUP để tải video:



Click PLAY để chạy:



Click PAUSE để dừng và click TEARDOWN để kết thúc.



9 Extend

9.1 Extend 1

Trong phần này, chúng ta sẽ thực hiện các phép toán thống kê về tỉ lệ mất gói và tốc độ truyền video

Tỉ lệ mất gói

Để xác định tỉ lệ mất gói, chúng ta cần đếm được số gói tin có thể nhận được ở phía client và tổng số gói tin được truyền từ server. Việc xác định lượng gói tin ở client chỉ đơn giản là phía client sẽ dùng một biến đếm, giá trị của nó sẽ tăng lên qua các lần nhận được gói tin. Đối với tổng số gói tin, ta có thể lấy số hiệu của frame hiện tại mà client nhận được, ví dụ như `currFrameNbr = 10`, tức đây là gói tin thứ 10, không quan tâm đến mất gói, thì số lượng gói tin mà server đã thực sự truyền đi sẽ là gói thứ 10 (các gói mất đi có số hiệu nhỏ hơn 10)

$$lossRate = \frac{currFrameNbr - countFrame}{currFrameNbr}$$

Với $lossRate$ là tỉ lệ mất gói, $currFrameNbr$ là số hiệu của gói tin hiện tại, $countFrame$ là số gói mà client đã nhận được.

Tốc độ truyền video

Để xác định tốc độ truyền video, ta cần tính được tổng số bytes dữ liệu mà client nhận được từ server, và thời gian của quá trình truyền nhận dữ liệu. Với tổng số bytes, chúng ta sẽ cộng kích thước các gói tin lại mà client đã nhận được lại với nhau. Với thời gian, chúng ta cần đặt các time stamp hợp lý để xác định được các khoảng thời gian mà gói tin được truyền và nhận, đầu tiên sẽ xác định từ lúc bắt đầu lệnh PLAY, sau đó sẽ tiếp tục cộng các khoảng thời gian vào một biến $totalTime$, các khoảng thời gian được tính bằng hiệu của thời điểm nhận gói tin cho time stamp trước đó. Và khi nhận gói tin, timestamp sẽ được cập nhập, mục đích là để loại bỏ các khoảng thời gian mà người dùng nhấn PAUSE.

$$videoDataRate = \frac{totalSizeData}{totalTime} (bytes/second)$$

Trong đó $videoDataRate$ là tốc độ truyền video, $totalSizeData$ là tổng kích thước các gói tin mà client nhận được, $totalTime$ là tổng thời gian quá trình truyền gói tin

9.2 Extend 2

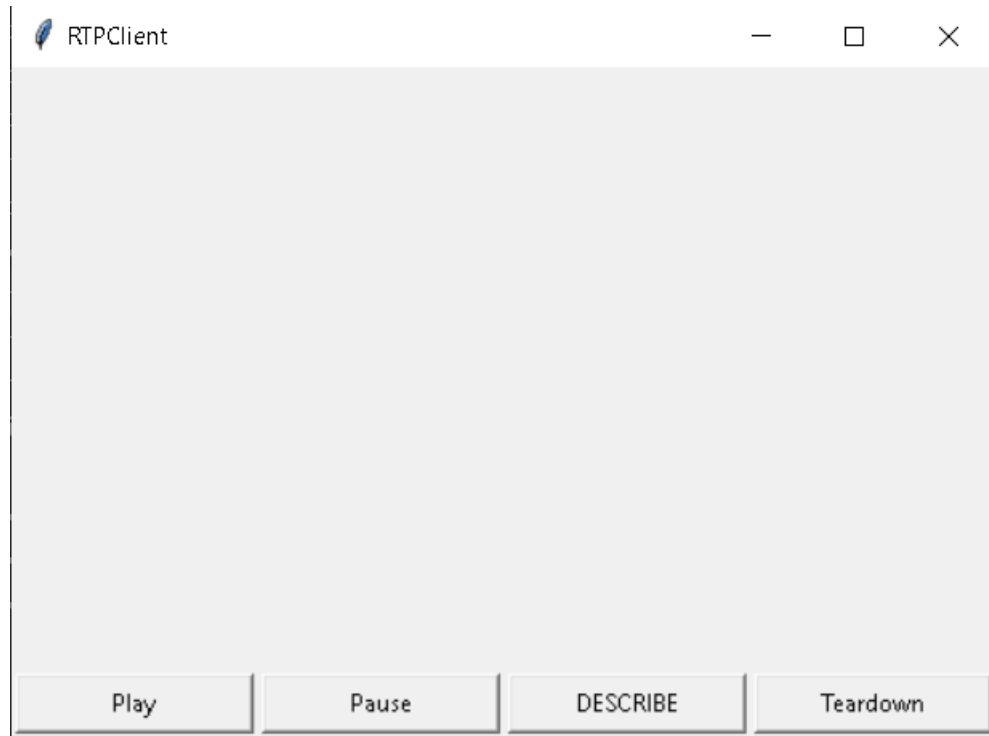
Ban đầu, giao diện gồm 4 nút với 4 hành động khác nhau. So với các chương trình chuẩn ví dụ như RealPlayer hoặc Window Media Player, những chương trình này chỉ có 3 nút với các thao tác tương tự: PLAY, PAUSE VÀ STOP (gần tương tự như TEARDOWN), không có nút SETUP. Cho biết SETUP là bắt buộc, vậy sẽ hiện thực nó như thế nào, khi nào client gửi SETUP? STOP có giống với TEARDOWN không.

- Bởi vì SETUP là bắt buộc, nên ta sẽ đưa hàm `setupMovie()` vào hàm khởi tạo.
- Các nút còn lại hoạt động bình thường.

9.3 Extend 3

Hiện thực phương thức Describe để gửi thông tin về media stream. Khi server nhận được yêu cầu Describe, nó sẽ gửi lại chi tiết thông tin mô tả về loại stream và encoding được sử dụng trong session.

- Tạo 1 nút **Describe** cho phép Client gửi request tới Server.

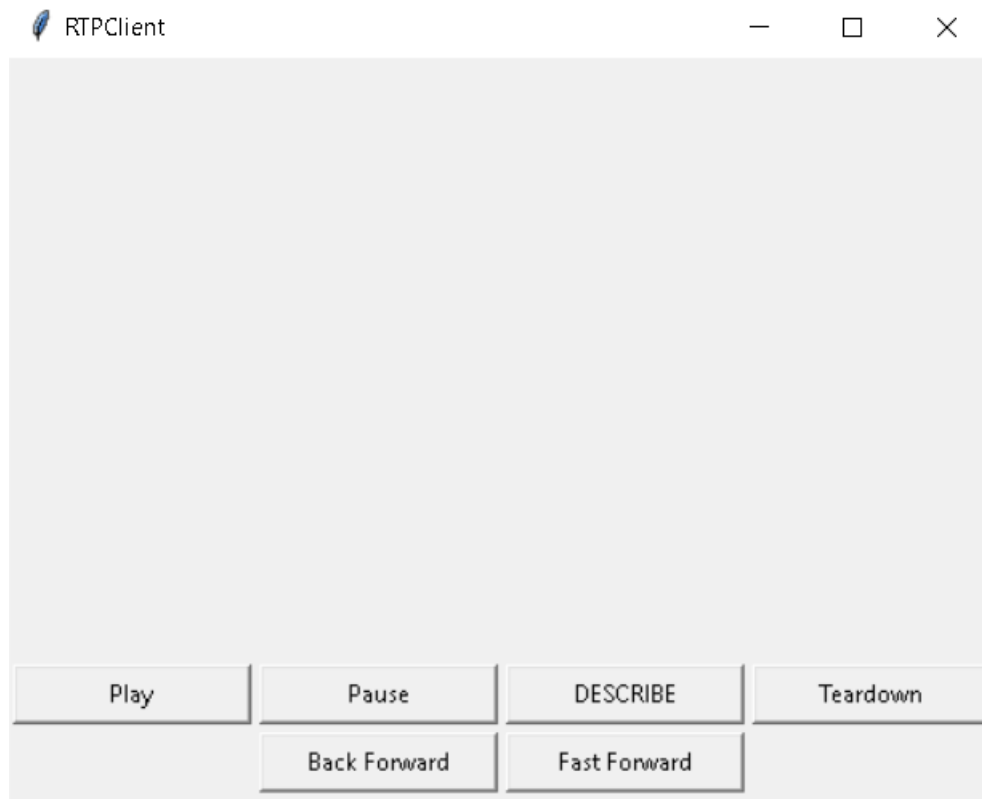


- Sau khi nhận Request từ Client, Server trả về 1 Message có những thông tin sau.

```
File name: movie.Mjpeg  
Protocol: RTSP/RTP 1.0  
Session: 587511
```

9.4 Extend 4

Hiện thực hai chức năng FastForward/BackForward và hiển thị tổng thời gian chạy của video
Thiết kế giao diện có nút FastForward/BackForward.



Hiện thực 2 chức năng FastForward/BackForward.

- Đối với 2 chức năng này, đầu tiên ta sẽ đưa về trạng thái READY bằng các gửi request PAUSE, sau đó gửi request **FASTFORWARD** hoặc **BACKFORWARD** đồng thời chuyển trạng thái **READY** sang **PLAYING** để chạy.

```
def fastAction(self):
    if not self.state == self.INIT:
        self.sendRtspRequest(self.PAUSE)
        # self.threadListenRtp()
        self.sendRtspRequest(self.FASTFORWARD)
        self.countFrame = self.countFrame + 9

def backAction(self):
    if not self.state == self.INIT:
        self.sendRtspRequest(self.PAUSE)
        # self.threadListenRtp()
        self.sendRtspRequest(self.BACKFORWARD)
        if (self.frameNbr - 10 > 0):
            self.frameNbr = self.frameNbr - 10
            self.countFrame = self.countFrame - 11
        else:
            self.frameNbr = 0
            self.countFrame = 0
```

- Bên Server, hai request **FastForward** và **BackForward** hiện thực chung với request **Play**.

```
elif (requestType == self.PLAY) or (requestType == self.FASTFORWARD) or (requestType == self.BACKFORWARD):
    if self.state == self.READY:
        if requestType == self.PLAY:
            print("processing PLAY\n")
        elif requestType == self.FASTFORWARD:
            print("processing FASTFORWARD\n")
            self.fastFlag = True
        else:
            print("processing BACKFORWARD\n")
            self.backFlag = True

    self.state = self.PLAYING

    # Create a new socket for RTP/UDP
    self.clientInfo["rtpSocket"] = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    self.replyRtp(self.OK_200, seq[1])

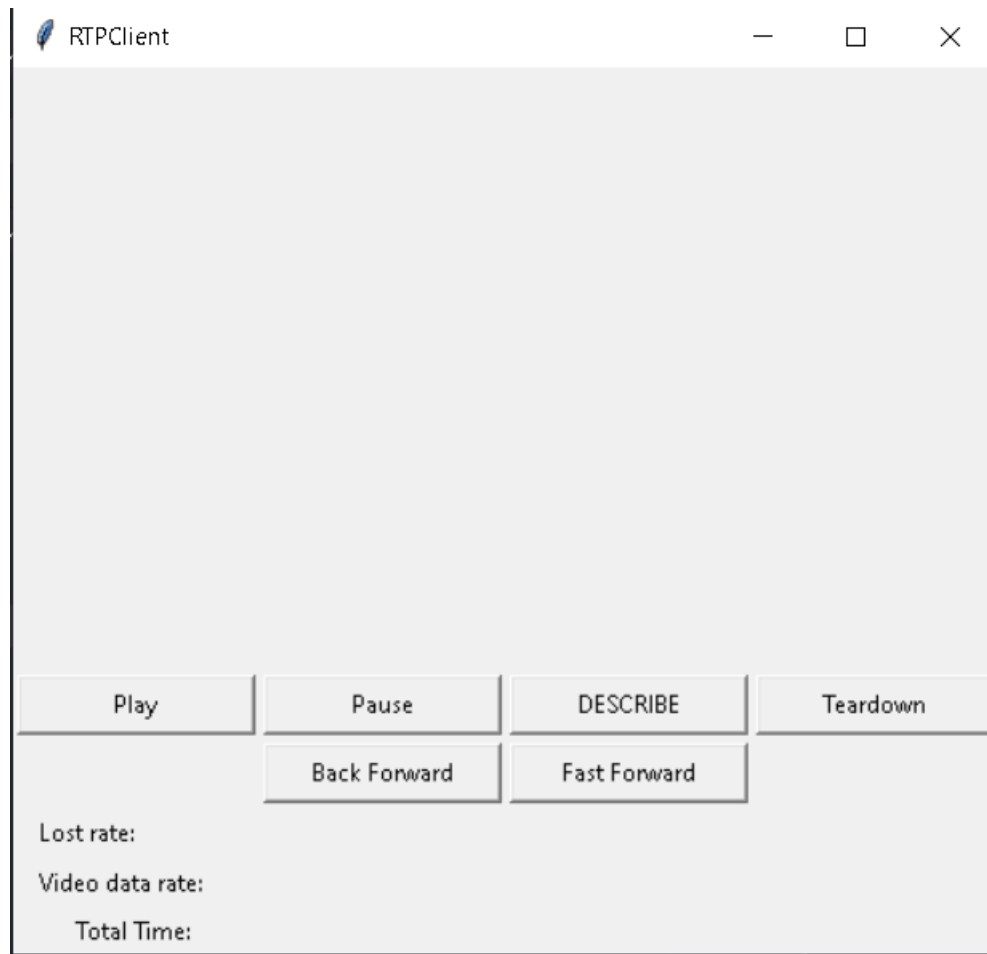
    # Create a new thread and start sending RTP packets
    self.clientInfo['event'] = threading.Event()
    self.clientInfo['worker'] = threading.Thread(target=self.sendRtp)
    self.clientInfo['worker'].start()
```

- Bên server, khi nhận được yêu cầu, Server sẽ gọi hàm để đi tới vị trí 10 frame tiếp theo hoặc quay về 10 frame trước đó. Hai hàm lần lượt là back10Frame() và next10Frame() được hiện thực ở module VideoStream.

Hiện thực thời gian chạy và hiển thị trên màn hình.

- Ta tính tổng thời gian chạy của video bằng cách tính tổng thời gian thực từ lúc chạy cho đến lúc kết thúc. Ta sẽ dùng hàm time() trong module time để lấy thời gian thực.

```
def listenRtp(self):
    """Listen for RTP packets."""
    # print('Start receiving RTP packet\n')
    self.timeStamp = time.time()
    while True:
        try:
            print("listening")
            currentTime = time.time()
            self.totalTime += currentTime - self.timeStamp
            self.timeStamp = currentTime
            videoDataRate = self.sizeData * 1.0 / self.totalTime
```



- Ta cập nhật thời gian thông qua phương thức updateMovie().

```
def updateMovie(self, imageFile, lossRate, videoDataRate, totalTime):  
    """Update the image file as video frame in the GUI."""  
    photo = ImageTk.PhotoImage(Image.open(imageFile))  
    self.label.configure(image = photo, height=288)  
    self.label.image = photo  
  
    self.lostRate['text'] = "Lost rate:      {:.3f}".format(lossRate)  
    self.videoDataRate['text'] = "Video data rate:      {:.3f} bytes/s".format(videoDataRate * 1.0)  
    self.labelTotalTime['text'] = "Total Time:      {:.3f} s".format(totalTime)
```

Sau khi chạy ta được kết quả như hình:

