

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



BÁO CÁO

Computer Network

Assignment 1: Implement a streaming video server and client

GVHD: Bùi Xuân Giang

TP. Hồ Chí Minh, Ngày 10 tháng 5 năm 2023

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

BÁO CÁO BÀI TẬP LỚN

Assignment 1: Implement a streaming video server and client

THÔNG TIN SINH VIÊN

Họ Và Tên	MSSV	Email
Nguyễn Thành Tuấn	2014931	tuan.nguyencsboyk20@hcmut.edu.vn

Mục lục

1	Requirements Analysis	3
1	Yêu cầu chức năng :	3
2	Yêu cầu phi chức năng :	3
2	Function description	4
1	Client:	4
2	Server:	4
3	Class Diagram	5
4	A Summative Evaluation of Results Achieved	6
5	User Manual	7
6	Extend	9
1	Calculate the statistics about the session:	9
1.1	RTP PACKET LOSS RATE	9
1.2	VIDEO DATA RATE	10
2	Implement PLAY, PAUSE and STOP buttons:	11
3	Implement DESCRIBE method:	12

Chương 1

Requirements Analysis

1 Yêu cầu chức năng :

- Triển khai một server video trực tuyến và client giao tiếp bằng phương thức RTSP và gửi dữ liệu 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.

2 Yêu cầu phi chức năng :

- Thời gian chờ của diagram socket để nhận RTP data là 0.5s.
- Server gửi gói RTP cho client bằng UDP mỗi 50ms.

Chương 2

Function description

1 Client:

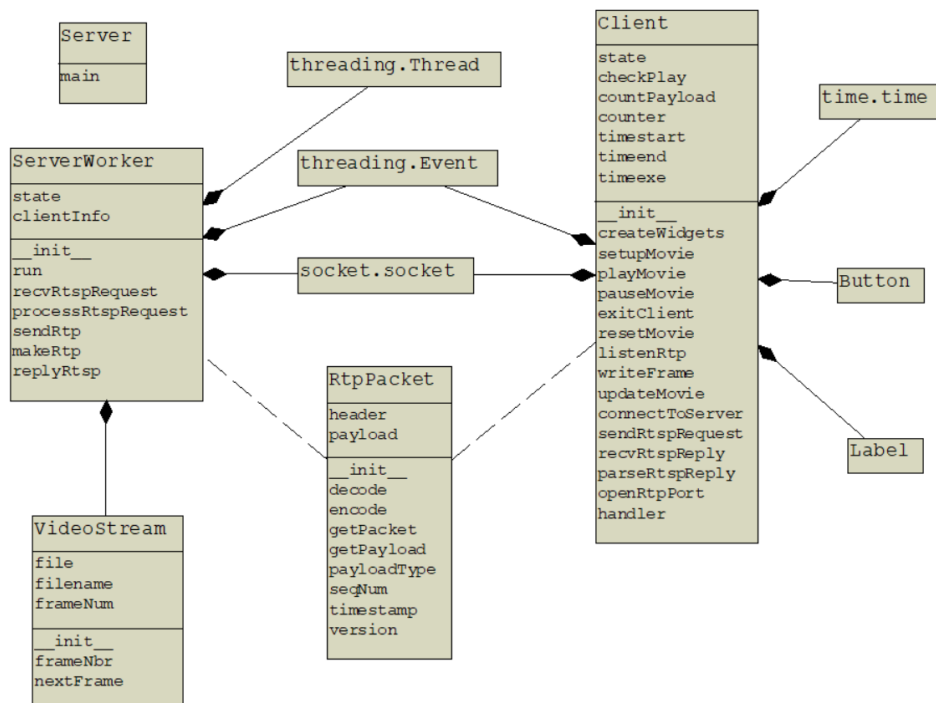
- **SETUP:** Gửi yêu cầu setup đến server, đọc phản hồi từ server và phân tích session header để lấy RTSP session ID, tạo một datagram socket để nhận RTP data và đặt thời gian chờ cho socket là 0.5s, video được khởi tạo. yêu cầu SETUP bắt buộc phải được hoàn thành trước khi một yêu cầu PLAY được gửi từ máy client.
- **PLAY:** Gửi yêu cầu play đến server, chỉ ra một dải (range) chỉ rõ một cách cụ thể số hiệu frame bắt đầu được gửi và số hiệu frame kết thúc; đọc phản hồi từ server, video phát.
- **PAUSE:** Gửi yêu cầu pause đến server, đọc phản hồi từ server, video tạm dừng, sau này có thể được nối lại với một yêu cầu PLAY.
- **TEARDOWN:** Gửi yêu cầu teardown đến server, đọc phản hồi từ server, quay trở về trạng thái ban đầu, chưa được SETUP.

2 Server:

- Khi server nhận yêu cầu SETUP từ client, server sẽ đáp trả lại bằng các xác nhận các tham số đã được lựa chọn, và điền vào các phần còn thiếu.
- Khi server nhận yêu cầu PLAY từ client, server đọc 1 video frame từ file và tạo RtpPacket. Sau đó, server gửi frame đến client.
- Khi server nhận yêu cầu PAUSE từ client, server sẽ tạm dừng gửi các frame dữ liệu tới máy client.
- Khi server nhận yêu cầu TEARDOWN từ client, server dừng gửi các frame tới máy client và kết thúc một phiên giao dịch của giao thức RTSP.

Chương 3

Class Diagram



Hình 3.1: Class Diagram của hệ thống RTP-RTSP Video Streaming

Chương 4

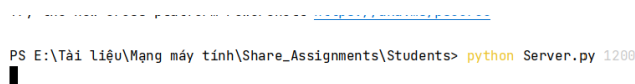
A Summative Evaluation of Results Achieved

- Hoàn thành được giao thức RTSP ở client
- Hoàn thành giao thức RTP ở server

Chương 5

User Manual

- Bước 01: Chúng ta phải chạy server trước : chạy terminal trong thư mục chứa file Server.py Gõ: `python Server.py «port-server»` chúng ta đặt nó là số bất kì lớn hơn 1024. Ví dụ chọn «port-server» là 1200



```
PS E:\Tài liệu\Mạng máy tính\Share_Assignments\Students> python Server.py 1200
```

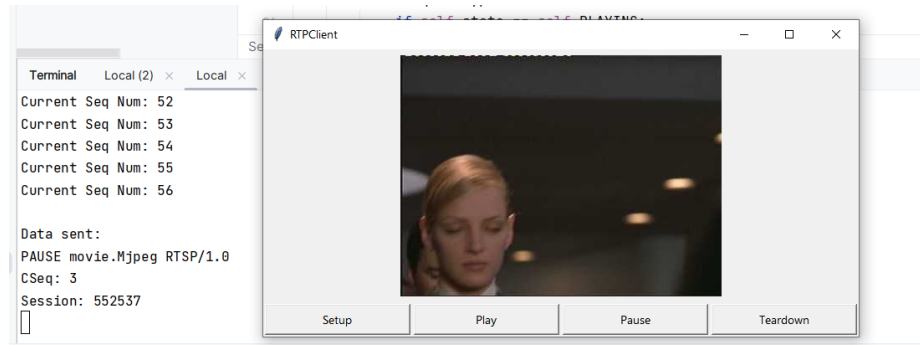
Hình 5.1: Chạy Server với port là 1200

- Bước 02: Chúng ta mở terminal mới trong thư mục chứa file ClientLauncher.py để xem đó là một client kết nối đến Server mình mới vừa chạy.
Gõ `python ClientLauncher.py «name-Server» «port-server» «port-RTP» «name-video»`, «name-Server» tên của server, «port-server» port của server mình muốn kết nối, «port-RTP» số bất kì là port để nhận RTP packet, «name-video» là tên video mình muốn xem. Ví dụ:
 - «name-Server»: là tên Server ở máy tính đang dùng, ở đây là "localhost"
 - «port-server»: là port đã khởi tạo ở bước 1, ở đây là 1200
 - «port-RTP»: ví dụ ta chọn 5008
 - «name-video»: tên của video, ở đây là movie.Mjpeg



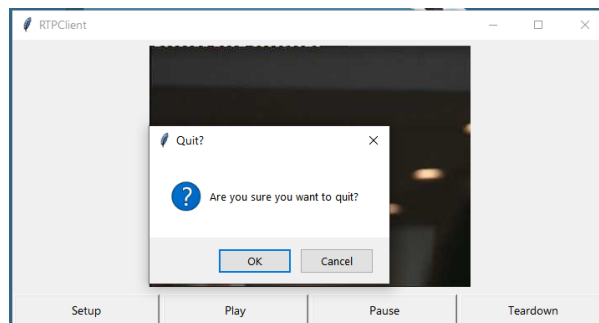
Hình 5.2: Chạy Client

- Bước 03: Nhấn vào setup để tạo đường truyền RTP và nhấn Play để xem video, pause để dừng và teardown để kết thúc.



Hình 5.3: Phát video

- Để thoát khỏi chương trình, đóng cửa sổ giao diện người dùng, ta bấm vào dấu X ở góc trên phải màn hình. Bấm OK để thoát, bấm Cancel để hủy bỏ yêu cầu.



Hình 5.4: Thoát chương trình

Chương 6

Extend

1 Calculate the statistics about the session:

1.1 RTP PACKET LOSS RATE

Xác định dựa trên sequence number ở bên phía Client, nếu như Current Sequence Number không tăng dần theo thứ tự thì chứng tỏ nó đã bị lost.

Theo như file ServerWorker.py, cứ mỗi 0.05 server sẽ gửi 1 gói tin, vậy thì tối đa có 20 gói được gửi, nhưng có những lúc mạng không ổn định thì có thể chỉ được 15-16 gói tin, vậy nên không thể xác định bằng cách đếm có đủ 20 gói tin trong 1s được.

Ta xử lý bằng cách đặt thêm biến counter khởi tạo bằng 0, biến này sẽ tăng mỗi khi listenRtp được gọi và bắt được data

```
def listenRtp(self):
    """Listen for RTP packets."""
    while True:
        try:
            data = self.rtpSocket.recv(20480)
            if data:
                rtpPacket = RtpPacket()
                rtpPacket.decode(data)

                currFrameNbr = rtpPacket.seqNum()
                self.counter += 1
                print("Current Seq Num: " + str(currFrameNbr))

                if currFrameNbr > self.frameNbr: # Discard the late packet
                    self.frameNbr = currFrameNbr
                    self.updateMovie(self.writeFrame(rtpPacket.getPayload()))
        except:
            # Stop listening upon requesting PAUSE or TEARDOWN
            if self.playEvent.isSet():
                break
```

Khi hàm exitClient() được gọi, rate sẽ được tính bằng công thức: current frame number hiện tại trừ đi số counter đã đếm được, đó chính là số packet bị loss, và chia cho frameNbr, chúng ta sẽ thu được tỷ lệ.

```
2 usages
def exitClient(self):
    """Teardown button handler."""
    self.sendRtspRequest(self.TEARDOWN)
    self.master.destroy() # Close the gui window
    os.remove(CACHE_FILE_NAME + str(self.sessionId) + CACHE_FILE_EXT) # Delete the cache image from video

    if self.frameNbr:
        rate = float((self.frameNbr - self.counter) / self.frameNbr)
        print("\nPacket loss rate: " + str(rate) + "\n")
```

1.2 VIDEO DATA RATE

Về vấn đề video data rate, ta sẽ tính bằng tổng số Payload (chính là phần data trừ đi số byte của header) đếm được trong 1 đơn vị thời gian chia cho thời gian đã chạy video.

Ta xử lý bằng cách thêm thư viện time của python, 3 biến khởi tạo timestart, timeend, timeexe.

Time start được đặt tính từ lúc người dùng bấm nút PLAY

```

import sys
import threading
import time

def playMovie(self):
    """Play button handler."""
    if self.state == self.READY:
        # Create a new thread to listen for RTP packets
        self.timestart = time.time()
        threading.Thread(target=self.listenRtp).start()
        self.playEvent = threading.Event()
        self.playEvent.clear()
        self.sendRtpRequest(self.PLAY)

```

Timeend được set khi Pause được gọi, timeexe cũng được tính bằng timeend - timestart, vì 1 video có thể bị pause nhiều lần, bởi Pause, hoặc nút thoát(X) nhưng lại cancel, nên chúng ta sẽ cộng dồn biến này lại

```

import sys
import threading
import time

def pauseMovie(self):
    """Pause button handler."""
    if self.state == self.PLAYING:
        self.timeend = time.time()
        self.timeexe += self.timeend - self.timestart
        self.sendRtpRequest(self.PAUSE)

```

Tổng số Payload thì được cộng dồn mỗi khi listenRtp được gọi và bắt được data

```

def listenRtp(self):
    """Listen for RTP packets."""
    while True:
        try:
            data = self.rtpSocket.recv(20480)
            if data:
                rtpPacket = RtpPacket()
                rtpPacket.decode(data)

                currFrameNbr = rtpPacket.seqNum()
                self.counter += 1
                print("Current Seq Num: " + str(currFrameNbr))

                if currFrameNbr > self.frameNbr: # Discard the late packet
                    self.frameNbr = currFrameNbr
                    self.countPayload += len(rtpPacket.getPayload())
                    self.updateMovie(self.writeFrame(rtpPacket.getPayload()))

        except:
            # Stop listening upon requesting PAUSE or TEARDOWN
            if self.playEvent.isSet():
                break

```

Khi exitClient() được gọi, Video data rate sẽ được tính bằng công thức tổng số Payload đếm được chia cho thời gian từ lúc chạy tới lúc kết thúc (không tính thời gian PAUSE) việc gửi các gói tin.

Kết quả:

```

Video data rate = 1977662 / 16.6620774269104 = 118692.4024735319

Data sent:
TEARDOWN movie.Mjpeg RTSP/1.0
CSeq: 4
Session: 998554

Packet loss rate: 0.0

```

2 Implement PLAY, PAUSE and STOP buttons:

SETUP là bắt buộc trong tương tác RTSP, ở trong bài tập lớn này, chúng ta có 2 cách đơn giản để hiện thực nó:

- Cách 1: Tự động SETUP ngay sau khi gõ lệnh ở terminal thứ 2.
- Cách 2: Chương trình sẽ SETUP ở lần bấm PLAY đầu tiên.

Ở đây, em xin chọn cách 1 để hiện thực nó.

Trong Windows Media Player, khi ta bấm vào STOP thì video sẽ dừng và quay trở về trạng thái ban đầu và đã được SETUP sẵn. Khi đó nếu bấm PLAY thì video sẽ chạy lại từ đầu. Ta chỉ có thể thoát khi bấm vào dấu X ở góc bên phải phía trên của ứng dụng.

Dưới đây là các bước thao tác cơ bản:

- Sau khi gõ 2 lệnh lần lượt trên 2 terminal. Chương trình sẽ tự động SETUP sẵn.

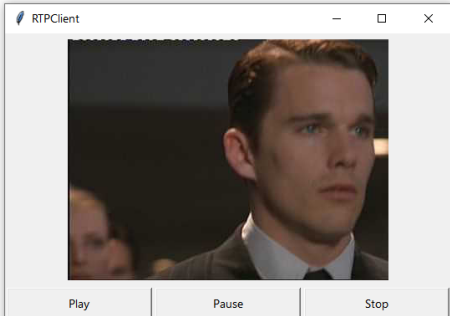
```
PS E:\Tài liệu\Mạng máy tính\Share_Assignments\extend2> python ClientLauncher.py localhost 1200 5008 movie.Mjpeg

Data sent:
SETUP movie.Mjpeg RTSP/1.0
CSeq: 1
TRANSPORT: RTP/UDP; client_port= 5008
Bind RtpPort Success
█
```



- Khi bấm PLAY, video sẽ được chạy.

```
.Current Seq Num: 9
.Current Seq Num: 10
.Current Seq Num: 11
.Current Seq Num: 12
.Current Seq Num: 13
.Current Seq Num: 14
.Current Seq Num: 15
.Current Seq Num: 16
.Current Seq Num: 17
.Current Seq Num: 18
.Current Seq Num: 19
.Current Seq Num: 20
.Current Seq Num: 21
.Current Seq Num: 22
.Current Seq Num: 23
.Current Seq Num: 24
.Current Seq Num: 25
.Current Seq Num: 26
.Current Seq Num: 27
.Current Seq Num: 28
.Current Seq Num: 29
.Current Seq Num: 30
.Current Seq Num: 31
.Current Seq Num: 32
.Current Seq Num: 33
.Current Seq Num: 34
.Current Seq Num: 35
```



- Khi bấm PAUSE, video sẽ tạm dừng

- Khi người dùng bấm vào STOP, thì video sẽ dừng lại và quay trở về trạng thái ban đầu. Được SETUP sẵn tương tự như Windows Media Player.

```
Data sent:
PAUSE movie.Mjpeg RTSP/1.0
CSeq: 3
Session: 649868
not receive data!

Data sent:
SETUP movie.Mjpeg RTSP/1.0
CSeq: 1
TRANSPORT: RTP/UDP; client_prot= 5008
Bind RtpPort Success
█
```



- Và cuối cùng, để thoát khỏi chương trình thì người dùng bấm vào dấu X ở góc phải màn hình. Chương trình sẽ thoát ngay lập tức mà không hỏi lại (tương tự Windows Media Player)

Với cách hiện thực như trên thì ta nhận thấy, khi bấm vào TEARDOWN ở phần hiện thực trong Bài tập lớn thì video sẽ dừng lại và quay về trạng thái ban đầu chưa được SETUP. Còn đối với STOP hiện thực ở trong phần extend này thì video sẽ dừng lại và quay về trạng thái ban đầu và được SETUP sẵn.

3 Implement DESCRIBE method: