HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY, VNU-HCM
FACULTY OF COMPUTER SCIENCE & ENGINEERNG

COMPUTER NETWORK

Assignment

Video Stream Programming

Tutor: Bùi Xuân Giang
Class: L05
Student members: Huỳnh Phạm Phước Linh - 1710165
Trương Ngọc Trung Anh - 2020004
Lê Văn Phong - 1712607

Ho Chi Minh, 11/2020

Contents

1 Introduction

Chương trình gồm 5 files:
Client.py
ClientLauncher.py
RtpPacket.py
Server.py
ServerWorker.py

2 Running the code

2.1 Server.py

Mẫu:
python Server.py server_port
or
Python Server.py server_port

Trong đó:
server_port là cổng mà Server của bạn lắng nghe các kết nối RTSP đến.

a. Chúng ta có thể cho nó giá trị là 1025

b. Cổng RTSP tiêu chuẩn là 554

c. Giá trị được đặt cho server_port phải lớn hơn 1024

2.2 ClientLauncher.py

Mẫu:
python ClientLauncher.py server_host server_port RTP_port name_video_file
or
Python ClientLauncher.py server_host server_port RTP_port name_video_file

Trong đó:

- **server_host** là địa chỉ IP của máy cục bộ. (VD:"127.0.0.1")
- **server_port** là cổng mà Server đang lắng nghe. (Vd: 1025)
- **RTP_port** là cổng mà các gói RTP được nhận. (Vd: 8005)
- **name_video_file** là tên của video muốn xem (Vd: video.Mjpeg)

3 Real Time Streaming Protocol (RTSP)

- Giao thức phát trực tuyến thời gian thực.
- Kiểm soát các streaming media server đối với hệ thống giải trí và truyền thông.
- Thiết lập và kiểm soát các media session giữa các điểm kết thúc.
- Sử dụng TCP.

4 Real-time Transport Protocol (RTP)

- Giao thức vận tải thời gian thực.
- Giao thức mạng để phân phối âm thanh và video qua IP Networks.
- Thiết lập và kiểm soát các media session giữa các điểm kết thúc.
- Sử dụng UDP.

5 Python code

```
class ServerWorker:
    SETUP = 'SETUP'
    PLAY = 'PLAY'
    PAUSE = 'PAUSE'
    TEARDOWN = 'TEARDOWN'

    INIT = 0
    READY = 1
    PLAYING = 2
    state = INIT

        OK_200 = 0
    FILE_NOT_FOUND_404 = 1
    CON_ERR_500 = 2
```

```
class Client:
    INIT = 0
    READY = 1
    PLAYING = 2
    state = INIT

    SETUP = 0
    PLAY = 1
    PAUSE = 2
    TEARDOWN = 3
```

Client sẽ gửi đến Server thông qua Giao thức RTSP là các lệnh như:

- SETUP
- PLAY
- PAUSE
- TEARDOWN

Các lệnh này sẽ cho phía Server biết hành động tiếp theo mà nó sẽ hoàn thành. Server sẽ trả lời Client thông qua Giao thức RTSP là các giá trị như:

- OK_200
- FILE_NOT_FOUND_404
- CON_ERR_500

5.1 SETUP Command

Nếu lệnh SETUP được gửi từ Client đến Server.

```
def sendRtspRequest(self, requestCode):
    if requestCode == self.SETUP and self.state == self.INIT:
        threading.Thread(target=self.recvRtspReply).start()

        self.rtspSeq = 1

        request = "SETUP " + str(self.fileName) + "\n" + str(self.rtspSeq) + "\n" + " RTSP/1.0
            RTP/UDP " + str(self.rtpPort)

        self.rtspSocket.send(request.encode('utf-8'))

        self.requestSent = self.SETUP
```

Gói "SETUP" RTSP sẽ bao gồm:

- Lệnh SETUP
- Tên video
- RTSP Packet Sequence Number bắt đầu 1
- Protocol type: RTSP/1.0 RTP
- Transmission Protocol: UDP
- Cổng RTP để truyền luồng video

```
# Process SETUP request
    if requestType == self.SETUP:
        if self.state == self.INIT:
            # Update state
            print("processing SETUP\n")

            try:
                self.clientInfo['videoStream'] = VideoStream(filename)
                self.state = self.READY
            except IOError:
                self.replyRtsp(self.FILE_NOT_FOUND_404, seq[1])

            # Generate a randomized RTSP session ID
            self.clientInfo['session'] = randint(100000, 999999)

            # Send RTSP reply
            self.replyRtsp(self.OK_200, seq[0])

            # Get the RTP/UDP port from the last line
            self.clientInfo['rtpPort'] = request[2].split(' ')[3]
```

Khi Server nhận được lệnh SETUP:

- Gán cho Client một Specific Session Number ngẫu nhiên
- Nếu lệnh hoặc trạng thái Server lỗi, trả gói ERROR lại cho Client
- Nếu không có lỗi:

  ```
  class VideoStream:
      def __init__(self, filename):
          self.filename = filename
          try:
              self.file = open(filename, 'rb')
          except:
              raise IOError
          self.frameNum = 0
  ```

  Server sẽ mở tệp video được chỉ định trong gói SETUP và khởi tạo số khung hình video của nó bằng 0. Trả về OK_200 cho Client và gán state = READY.

Client sẽ liên tục nhận Server's RTSP Reply

```
def recvRtspReply(self):
    while True:
        reply = self.rtspSocket.recv(1024)
        if reply:
            self.parseRtspReply(reply.decode("utf-8"))
        if self.requestSent == self.TEARDOWN:
            self.rtspSocket.shutdown(socket.SHUT_RDWR)
            self.rtspSocket.close()
            break
```

Sau đó phân tích cú pháp của gói RTSP Reply, nhận Session Number

```
def parseRtspReply(self, data):
    lines = data.split('\n')
    seqNum = int(lines[1].split(' ')[1])
    if seqNum == self.rtspSeq:
        session = int(lines[2].split(' ')[1])
        if self.sessionId == 0:
            self.sessionId = session
```

Nếu như gói Reply được phản hồi là lệnh SETUP thì Client sẽ đặt giá trị state = READY.

```
# Process only if the session ID is the same
if self.sessionId == session:
    if int(lines[0].split(' ')[1]) == 200:
        if self.requestSent == self.SETUP:
            print ("Updating RTSP state...")
            self.state = self.READY
            print ("Setting Up RtpPort for Video Stream")
            self.openRtpPort()
        elif self.requestSent == self.PLAY:
            self.state = self.PLAYING
            print ('-'*60 + "\nClient is PLAYING...\n" + '-'*60)
        elif self.requestSent == self.PAUSE:
            self.state = self.READY
            self.playEvent.set()
        elif self.requestSent == self.TEARDOWN:
            self.teardownAcked = 1
```

Sau đó, mở cổng RTP để nhận video stream.

```
def openRtpPort(self):
    self.rtpSocket.settimeout(0.5)
        try:
            self.rtpSocket.bind((self.serverAddr,self.rtpPort))
            print ("Bind RtpPort Success")
        except:
            tkMessageBox.showwarning('Unable to Bind', 'Unable to bind PORT=%d' %self.rtpPort)
```

5.2 PLAY Command

Nếu lệnh PLAY được gửi từ Client đến Server

```
elif requestCode == self.PLAY and self.state == self.READY:
    self.rtspSeq = self.rtspSeq + 1
    request = "PLAY " + "\n" + str(self.rtspSeq)
    self.rtspSocket.send(request.encode("utf-8"))
    print ('-'*60 + "\nPLAY request sent to Server...\n" + '-'*60)
    self.requestSent = self.PLAY
```

Server sẽ tạo một Socket truyền từ RTP qua UDP và bắt đầu gửi gói video stream

```
elif requestType == self.PLAY:
    if self.state == self.READY:
    print("processing PLAY\n")
    self.state = self.PLAYING
    self.clientInfo["rtpSocket"] = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    self.replyRtsp(self.OK_200, seq[0])
    self.clientInfo['event'] = threading.Event()
    self.clientInfo['worker']= threading.Thread(target=self.sendRtp)
    self.clientInfo['worker'].start()
```

VideoStream.py sẽ giúp cắt tệp video thành từng frame riêng biệt và đưa từng frame vào gói dữ liệu RTP

```
def nextFrame(self):
    data = self.file.read(5)
    if data:
        framelength = int(data)
        data = self.file.read(framelength)
        self.frameNum += 1
    return data
```

Mỗi gói dữ liệu cũng sẽ được mã hóa với một header, header sẽ bao gồm: RTP-version filed, Padding, extension, Contributing source, Marker, Type Field, Sequence Number, Timestamp, SSRC, Payload).

RTP Packet Header							
Bit Offset	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						
96	CSRC identifiers ...						
96+32xCC	Profile-specific extension header ID						Extension header length
128+32xCC	Extension header ...						

```
def encode(self, version, padding, extension, cc, seqnum, marker, pt, ssrc, payload):
    timestamp = int(time())
    self.header = bytearray(HEADER_SIZE)
    self.header[0] = version << 6
    self.header[0] = self.header[0] | padding << 5
    self.header[0] = self.header[0] | extension << 4
    self.header[0] = self.header[0] | cc
    self.header[1] = marker
    self.header[1] = self.header[1] | pt

    self.header[2] = seqnum >> 8
    self.header[3] = seqnum

    self.header[4] = (timestamp >> 24) & 0xFF
    self.header[5] = (timestamp >> 16) & 0xFF
    self.header[6] = (timestamp >> 8) & 0xFF
    self.header[7] = timestamp & 0xFF

    self.header[8] = ssrc >> 24
    self.header[9] = ssrc >> 16
    self.header[10] = ssrc >> 8
    self.header[11] = ssrc

    self.payload = payload
```

Một RTP Packet sẽ bao gồm header và video frame sẽ được gửi đến RTP Port của Client.

```
def sendRtp(self):
    while True:
        self.clientInfo['event'].wait(0.05)
        if self.clientInfo['event'].isSet():
            break
        data = self.clientInfo['videoStream'].nextFrame()
        if data:
            frameNumber = self.clientInfo['videoStream'].frameNbr()
            try:
                address = self.clientInfo['rtspSocket'][1][0]
                port = int(self.clientInfo['rtpPort'])
                ##================ Here ================##
                self.clientInfo['rtpSocket'].sendto(self.makeRtp(data, frameNumber),(address,
                    port))
            except:
                print("Connection Error")
```

Client sẽ decode RTP Packet để lấy header và video frame, tổ chức lại các frame và hiển thị trên giao diện người dùng (User Interface).

```
def decode(self, byteStream):
    self.header = bytearray(byteStream[:HEADER_SIZE])
    self.payload = byteStream[HEADER_SIZE:]
```

5.3 PAUSE Command

Nếu lệnh PAUSE được gửi từ Client đến Server, nó sẽ dừng gửi các video frame từ Server đến Client.

```
elif requestCode == self.PAUSE and self.state == self.PLAYING:
    self.rtspSeq = self.rtspSeq + 1
    request = "PAUSE " + "\n" + str(self.rtspSeq)
    self.rtspSocket.send(request.encode("utf-8"))
    print ('-'*60 + "\nPAUSE request sent to Server...\n" + '-'*60)
    self.requestSent = self.PAUSE
```

Nếu như gói Reply nhận lệnh PAUSE thì Client sẽ đặt giá trị state = READY.

```
elif self.requestSent == self.PAUSE:
    self.state = self.READY
```

5.4 TEARDOWN Command

Nếu lệnh TEARDOWN được gửi từ Client đến Server, nó sẽ ngăn Server gửi các video frame đến Client và đóng tất cả Client terminal.

```
elif requestCode == self.TEARDOWN and not self.state == self.INIT:
    self.rtspSeq = self.rtspSeq + 1
    request = "TEARDOWN " + "\n" + str(self.rtspSeq)
    self.rtspSocket.send(request.encode("utf-8"))
    print ('-'*60 + "\nTEARDOWN request sent to Server...\n" + '-'*60)
    self.requestSent = self.TEARDOWN
```

```
elif self.requestSent == self.TEARDOWN:
    self.teardownAcked = 1
```

References

[1] wikipedia. “**link: http://en.wikipedia.org/**”, , last access: 15/11/2020.