Multi media streaming protocol

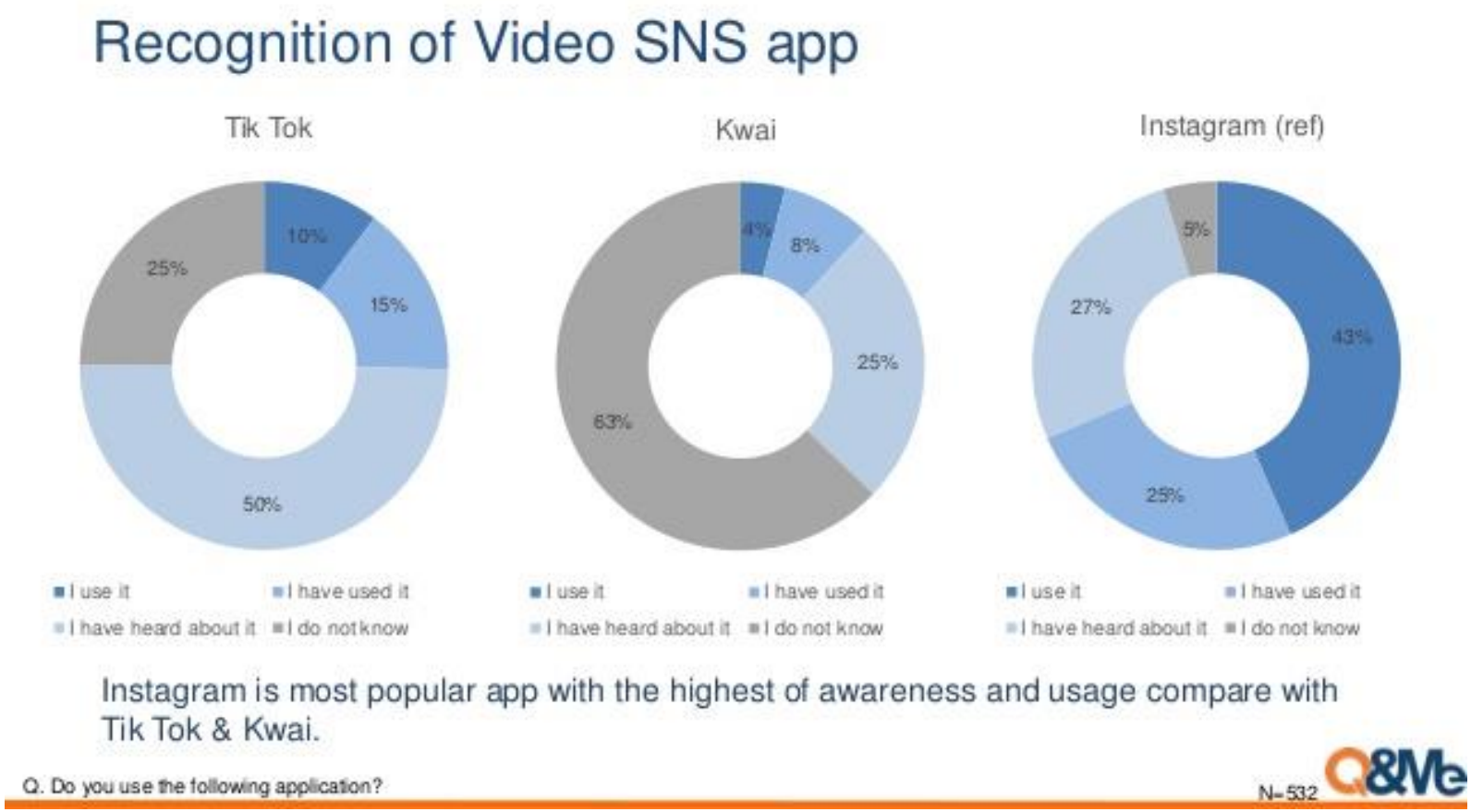
Computer Vision Lab, University of Seoul
김형욱(hyounguk1112@gmail.com)

Social Network Service

세상과 소통하고 사람들과 어울리는 주요한 매개체 or 인생의 낭비?

Social Network Service

사진과 텍스트 기반에서 점차 짧은 동영상 공유가 주요한 컨텐츠가 되고 있다!



Video streaming protocol

Todays Contents!

1. RTP
2. RTSP
3. RTCP
4. Example software

What is RTP?

The **Real Time Streaming Protocol**(**RTP**) is a network protocol for delivering audio and video over IP networks.

1. Video와 Audio “데이터 " 를 전송하기 위한 프로토콜 (Application protocol)

2. UDP/IP로 패킷 전송 (Muiti-Media transfer usually use it.)

3. 헤더에는 코덱 정보, **Sequence number, Timestamp**, SSRC(Syncronization Source Identification, multi-RTP sessions)

4. One way comm(Server to Client)

What is RTP?

	TCP	UDP
연결 방식	연결형 프로토콜 연결 후 통신 1:1 통신 방식	비연결형 프로토콜 연결 없이 통신 1:1, 1:N, N:N 통신 방식
특징	- 데이터의 경계를 구분 안함 - 신뢰성 있는 데이터 전송 - 데이터의 전송 순서 보장 - 데이터의 수신 여부 확인 - 패킷을 관리할 필요 없음 - UDP보다 전송속도가 느림	- 데이터의 경계를 구분함 - 신뢰성 없는 데이터 전송 - 데이터의 전송 순서가 바뀔 수 있음 - 데이터의 수신 여부를 확인 안함 - 패킷을 관리해야함 - TCP보다 전송속도가 빠름

TCP는 전송에 대한 신뢰성을 보장해주지만 Three-way handshaking을 통한 연결을 하고, 여러 기능을 위한 긴 packet heade을 가져 overhea가 발생한다. Delay sensitiv한 응용에서는 유리하지 않음.

What is RTP?

The **Real Time Tranport Protocol**(**RTP**) is a network protocol for delivering audio and video over IP networks.

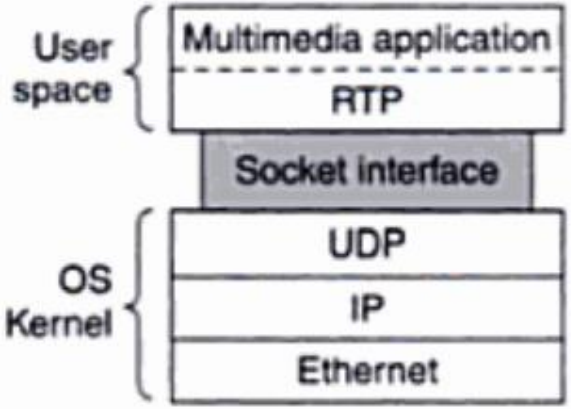
1. Video와 Audio “데이터 " 를 전송하기 위한 프로토콜 (Application protocol)

2. UDP/IP로 패킷 전송 (Muiti-Media transfer usually use it.)

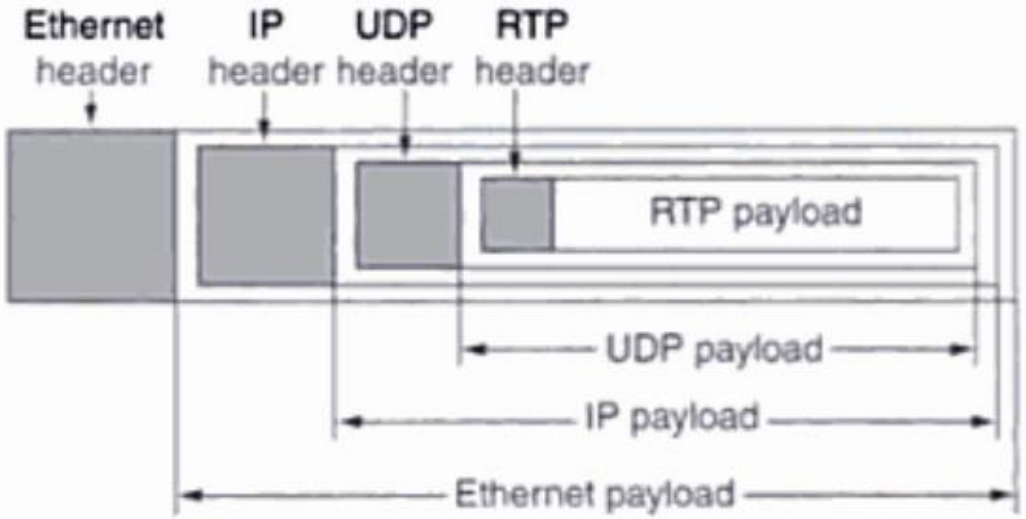
3. 헤더에는 코덱 정보, **Sequence number, Timestamp**, SSRC(Syncronization Source Identification, multi-RTP sessions)

4. One way comm(Server to Client)

What is RTP?



(a)



(b)

What is RTP?

RTP packet header

Offsets	Octet	0								1								2								3							
Octet	Bit [a]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version			P	X		CC		M				PT				Sequence number															
4	32	Timestamp																															
8	64	SSRC identifier																															
12	96	CSRC identifiers ...																															
12+4×CC	96+32×CC	Profile-specific extension header ID																Extension header length															
16+4×CC	128+32×CC	Extension header ...																															

V = Version
P = Padding
X = Extension

CC = CSRC count
M = Marker
PT = Payload type

https://en.wikipedia.org/wiki/RTP_audio_video_profile

Codec, Clock rate(sampling rate) info...

What is RTSP?

The **Real Time Streaming Protocol(RTSP)** is a network control protocol designed for use in entertainment and communications systems to control streaming media servers.

1. 스트리밍 데이터를 제어하기 위한 프로토콜 (Application protocol)
2. 재생, 일시정지, 빨리감기, 되감기, 재생위치 변경 같은 명령 전송
3. TCP/UDP 모두 사용 가능하나 UDP가 주로 사용된다. (port 554번)
4. SETUP 패킷(connection establish)에서 어떤 파일(URL)을 재생할 것인지, 포트번호와 프로토콜(UDP/RTP)을 무엇을 사용할 것인지를 알려주면 서버에서 Session 번호를 알려준다. 그 세션을 통해 클라이언트를 구분.

What is RTSP?

Option : server에 요청할 수 있는 선택지

```
C->S:  OPTIONS rtsp://example.com/media.mp4 RTSP/1.0
       CSeq: 1
       Require: implicit-play
       Proxy-Require: gzipped-messages

S->C:  RTSP/1.0 200 OK
       CSeq: 1
       Public: DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE
```

Describe : 해당 item에 대한 정보

```
C->S: DESCRIBE rtsp://example.com/media.mp4 RTSP/1.0
      CSeq: 2

S->C: RTSP/1.0 200 OK
      CSeq: 2
      Content-Base: rtsp://example.com/media.mp4
      Content-Type: application/sdp
      Content-Length: 460

      m=video 0 RTP/AVP 96
      a=control:streamid=0
      a=range:npt=0-7.741000
      a=length:npt=7.741000
      a=rtpmap:96 MP4V-ES/5544
      a=mimetype:string;"video/MP4V-ES"
      a=AvgBitRate:integer;304018
      a=StreamName:string;"hinted video track"
      m=audio 0 RTP/AVP 97
      a=control:streamid=1
      a=range:npt=0-7.712000
      a=length:npt=7.712000
      a=rtpmap:97 mpeg4-generic/32000/2
      a=mimetype:string;"audio/mpeg4-generic"
      a=AvgBitRate:integer;65790
      a=StreamName:string;"hinted audio track"
```

What is RTSP?

SETUP : RTP socket setup! 이때 server는 session id를 알려줘야 한다.

```
C->S: SETUP rtsp://example.com/media.mp4/streamid=0 RTSP/1.0
      CSeq: 3
      Transport: RTP/AVP;unicast;client_port=8000-8001

S->C: RTSP/1.0 200 OK
      CSeq: 3
      Transport: RTP/AVP;unicast;client_port=8000-8001;server_port=9000-9001;ssrc=1234ABCD
      Session: 12345678
```

PLAY : RTP protocol에 의해 전송 시작

```
C->S: PLAY rtsp://example.com/media.mp4 RTSP/1.0
      CSeq: 4
      Range: npt=5-20
      Session: 12345678

S->C: RTSP/1.0 200 OK
      CSeq: 4
      Session: 12345678
      RTP-Info: url=rtsp://example.com/media.mp4/streamid=0;seq=9810092;rtptime=3450012
```

What is RTSP?

RTSP는 양방향 송신으로 TCP/IP를 쓰는 http와 유사하다. 그러나 stateless한 http와 달리 state를 가지고 있다.

What is RTCP?

The **RTP Control Protocol(RTCP)** is a sister protocol of the Real-time Transport Protocol (RTP). RTCP provides out-of-band statistics and control information for an RTP session. It partners with RTP in the delivery and packaging of multimedia data, but does not transport any media data itself.

1. RTP 데이터의 전송 상태 감시를 위한 프로토콜 (Application protocol)
2. 주기적으로 송수신자가 리포트 패킷을 주고 받음.
3. 보낸 패킷, 손실된 패킷, 패킷 수신 간격 정보를 주고 받음
4. RTP에 대한 flow control을 위해 사용
5. 표준에서는 RTCP로부터 수집된 정보에 따라 어떻게 처리하는 가는 명시되어 있지 않음. 응용 개발자의 설계에 자유롭게 반영할 수 있다.
   E.G : codec 변경, 전송속도 조절 등…

Project

RTSP in code

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

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

    counter = 0
    # Initiation..
    def __init__(self, master, serveraddr, serverport, rtpport, filename):
        self.master = master
        self.master.protocol("WM_DELETE_WINDOW", self.handler)
        self.createWidgets()
        self.serverAddr = serveraddr
        self.serverPort = int(serverport)
        self.rtpPort = int(rtpport)
        self.fileName = filename
        self.rtspSeq = 0
        self.sessionId = 0
        self.requestSent = -1
        self.teardownAcked = 0
        self.connectToServer()
        self.frameNbr = 0
        self.rtpSocket = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
```

RTSP의 state와 action을 보관하기 위해서 client 클래스는 왼쪽과 같은 멤버를 갖는다.

RTSP를 위한 python socke을 생성하는 것을 볼 수 있다. SOCK_STREAM은 TCP 방식의 socket이라고 보면 된다.

```
def connectToServer(self):
    """Connect to the Server. Start a new RTSP/TCP session."""
    self.rtspSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    try:
        self.rtspSocket.connect((self.serverAddr, self.serverPort))
    except:
        tkMessageBox.showwarning('Connection Failed', 'Connection to \'%s\' failed.' %self.serverAddr)
```

RTSP in code

```
def setupMovie(self):
    """Setup button handler."""
    if self.state == self.INIT:
        self.sendRtspRequest(self.SETUP)

def exitClient(self):
    """Teardown button handler."""
    self.sendRtspRequest(self.TEARDOWN)
    #self.handler()
    self.master.destroy() # Close the gui window
    os.remove(CACHE_FILE_NAME + str(self.sessionId) + CACHE_FILE_EXT) # Delete the cache image from video
    rate = float(self.counter/self.frameNbr)
    print '-'*60 + "\nRTP Packet Loss Rate :" + str(rate) +"\n" + '-'*60
    sys.exit(0)

def pauseMovie(self):
    """Pause button handler."""
    if self.state == self.PLAYING:
        self.sendRtspRequest(self.PAUSE)

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

RTSP Client class의 request를 위한 멤버 method들이다.

RTSP in code

```
def sendRtspRequest(self, requestCode):
    """Send RTSP request to the server."""
    #-------------
    # TO COMPLETE
    #-------------

    # Setup request
    if requestCode == self.SETUP and self.state == self.INIT:
        threading.Thread(target=self.recvRtspReply).start()
        # Update RTSP sequence number.
        # ...
        self.rtspSeq = 1

        # Write the RTSP request to be sent.
        # request = ...
        request = "SETUP " + str(self.fileName) + "\n" + str(self.rtspSeq) + "\n" + " RTSP/1.0 RTP/UDP " + str(self.rtpPort)

        self.rtspSocket.send(request)
        # Keep track of the sent request.
        # self.requestSent = ...
        self.requestSent = self.SETUP
```

예를 들어 SETUP method가 호출되면 위와 같은 문자열이 TCP/IP socket에 의해 전송된다.

RTSP in code

```
class Server:

    def main(self):
        try:
            SERVER_PORT = int(sys.argv[1])
        except:
            print "[Usage: Server.py Server_port]\n"
        rtspSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        rtspSocket.bind(('', SERVER_PORT))
        print "RTSP Listing incoming request..."
        rtspSocket.listen(5)

        # Receive client info (address,port) through RTSP/TCP session
        while True:
            clientInfo = {}
            clientInfo['rtspSocket'] = rtspSocket.accept()    # this accept

            ServerWorker(clientInfo).run()
```

서버 쪽에서는 마찬가지로 socket class를 만들고 listening 상태로 대기, client의 request를 받는다. 받은 정보를 통해 server가 serving을 시작!

RTSP in code

```
def run(self):
    threading.Thread(target=self.recvRtspRequest).start()

def recvRtspRequest(self):
    """Receive RTSP request from the client."""
    connSocket = self.clientInfo['rtspSocket'][0]
    while True:
        data = connSocket.recv(256)  ###
        if data:
            print '-'*60 + "\nData received:\n" + '-'*60
            self.processRtspRequest(data)
```

Received request 정보를 토대로 Request를 processing한다!

RTP in code

```
# Process PLAY request
elif requestType == self.PLAY:
    if self.state == self.READY:
        print '-'*60 + "\nPLAY Request Received\n" + '-'*60
        self.state = self.PLAYING

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

        self.replyRtsp(self.OK_200, seq[0])
        print '-'*60 + "\nSequence Number ("+ seq[0] + ")\nReplied to client\n" + '-'*60

        # 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()
```

ProcessRtspRequest 중 PLAY에는 바로 RTP protocol에 의한 socket이 생성되어 packet을 전송한다.

RTP in code

```
self.clientInfo['rtpSocket'].sendto(self.makeRtp(data, frameNumber),(self.clientInfo['rtspSocket'][1][0],port))
```

```
def makeRtp(self, payload, frameNbr):
    """RTP-packetize the video data."""
    version = 2
    padding = 0
    extension = 0
    cc = 0
    marker = 0
    pt = 26 # MJPEG type
    seqnum = frameNbr
    ssrc = 0

    rtpPacket = RtpPacket()

    rtpPacket.encode(version, padding, extension, cc, seqnum, marker, pt, ssrc, payload)

    return rtpPacket.getPacket()
```

RTP packet을 만들어 encoding해서 client로 전송

RTP in code

```
class RtpPacket:
    #header = bytearray(HEADER_SIZE)
    #HEADER_SIZE = 12

    def __init__(self):
        self.header = bytearray(HEADER_SIZE)

    def encode(self, version, padding, extension, cc, seqnum, marker, pt, ssrc, payload):
        """Encode the RTP packet with header fields and payload."""

        timestamp = int(time())
        print "timestamp: " + str(timestamp)
        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 << 7
        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
```

RTP packet의 forma와 encoding code

RTP in code

```
def listenRtp(self):
    while True:
        try:
            data,addr = self.rtpSocket.recvfrom(20480)listenRtp

            if data:
                rtpPacket = RtpPacket()
                rtpPacket.decode(data)
                print "||Received Rtp Packet #" + str(rtpPacket.seqNum()) + "|| "

                try:
                    if self.frameNbr + 1 != rtpPacket.seqNum():
                        self.counter += 1
                        print '!'*60 + "\nPACKET LOSS\n" + '!'*60
                    currFrameNbr = rtpPacket.seqNum()
                    #version = rtpPacket.version()
                except:
                    print "seqNum() error"
                    print '-'*60
                    traceback.print_exc(file=sys.stdout)
                    print '-'*60

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

Client에서의 RTP packet을 받아 decoding하여 GUI의 이미지를 update!

Thank you