Multimedia lab

❖ RTSP,RTP

Introduction

Multimedia networking is highly important since we are witnessing massive development of audio and video applications over the Internet.

The service requirements of multimedia applications differ significantly from those of traditional elastic applications such as e-mail, web browsing, and file downloading. In particular, unlike the elastic applications, multimedia applications are highly sensitive to end-to-end delay and delay variation but can tolerate occasional loss of data. So it is delay-sensitive loss-tolerant applications.

One the most famous streaming classes is streaming stored videos, like the idea of YouTube. In this class, the multimedia content is stored at the server.

The term streaming means that the video doesn't have to be completely received so the receiver can play it, instead the client can start playing the video without receiving the whole file.(remember YouTube).

In order to communicate between client and server, RTSP is used to communicate between them. RSTP is public domain protocol for providing user connectivity.

We'll discuss RTSP protocol in the following section.

Real Time Streaming Protocol (RTSP)

RTSP allows a media player to control the transmission of media form server. Controlling media includes playing, pausing, and fast-forwarding streams.

RTSP is an **out-of-band protocol** e.g. it's packets are sent from a port away from data stream.

RTSP Messages

There are some basic RTSP messages we will study including

SETUP

Setup message contains a lot of important information including : file name and RTP port number that the user is willing to accept RTP datagram packets.

PLAY , PAUSE , TEARDOWN : Easy ☺

Real Time Protocol (RTP)

RTP is a protocol used to transport common file formats such as MPEG and provide sequence numbers, time stamps and other important information that will be shown in the next section.

RTP typically runs on top of UDP. The sending side encapsulates a media chunk within an RTP packet, then encapsulates the packet in a UDP segment, and then hands the segment to IP.

The receiving side extracts the RTP packet from the UDP segment, then extracts the media chunk from the RTP packet, and then passes the segment to the media player.

RTP Header Format

RTP Header format is important for us to define. Since we will implement the algorithm from the scratch, we will show the important fields in the Header. See figure below :

version	Padding	Extension	CC	Marker	Type	SeqNum	Time Stamp	SSRC

Fields marked in blue are not important to us, for the rest of them:

Type

Indicates the format of the content. See table below for some famous formats and its corresponding type number.

Payload Type	Video Format
26	Motion JPEG
31	H.261
32	MPEG 1 video
33	MPEG 2 video

SeqNum

Contains the sequence number of the current RTP packet. Sequence numbers are used to solve the problem of frames arriving in disordered manner. With sequence numbers we can rearrange the stream easily.

Time Stamp

Time stamp field used to eliminate jitter –the variability of packet delays within the same packet stream.
So assume that we show image frames as soon as they arrive, and assume a jitter condition : that the first packet arrives with delay of 3 ms and the second one with delay 5 ms and the third of 1 ms and so on.
Then the effect on video is that it will inhibit some glitches in the picture.

To eliminate jitter we use buffering. By buffering we can store the arriving packets and then start playing it form the buffer. So it doesn't matter anymore when the frame arrived.

Practical part

When you look at the attached JAVA project, you will find 4 classes :

❖VideoStream

This class is used to read video data from the file on disk. For this class we are concerned with the input and output of this class. It can we illustrated as follows:

❖RTPpacket

This class is used to handle the RTP packets. The first constructor of this class to implement **RTP-Packetization** of the video data. The second constructor is used by the client to **De-Packetize** the data.

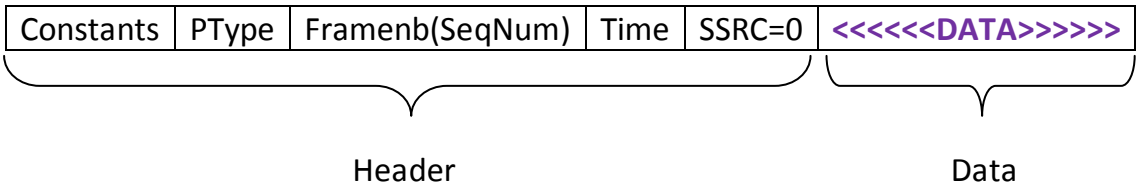
First constructor

```
public RTPpacket(int PType, int Framenb, int Time, byte[] data, int data_length){ }
```

As you can see, the constructor gets inputs needed to build an RTP packet.
This parameters are arranged like this:

Constants	PType	Framenb(SeqNum)	Time	SSRC=0

The complete packet will be as follows :



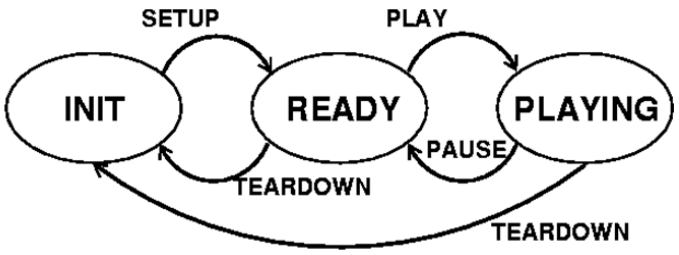
Second constructor

```
public RTPpacket(byte[] packet, int packet_size)
```

this constructor is used to de-packetize an RTP packet on the Client side. that is, extract the data from this packet.

❖Server

The server implementation is based on state machine coding style we will describe the state machine below :



**Note that SETUP, PLAY, PAUSE, and TEARDOWN are messages sent by Client using RTSP .

Now let's describe what the server does in these States:

INIT

At this stage, server is waiting for connection from the client at the RTSP socket.

```
request_type = theServer.parse_RTSP_request(); //blocking
```

as you can see above, when the server receives a SETUP RSTP command it will go to READY state. At this point, the server had received the file name for the video that the user wants to play. also, it had received the port number for the UDP connection to send datagram packets containing RTP packets.

READY

At this stage, server waits for PLAY message to start the timer (discussed later).

```
if ((request_type == PLAY) && (state == READY))
 {
  theServer.timer.start();
  //update state
  state = PLAYING;
 }
```

Playing

At this stage the RTP packets are being sent via datagram socket.

Timer idea :

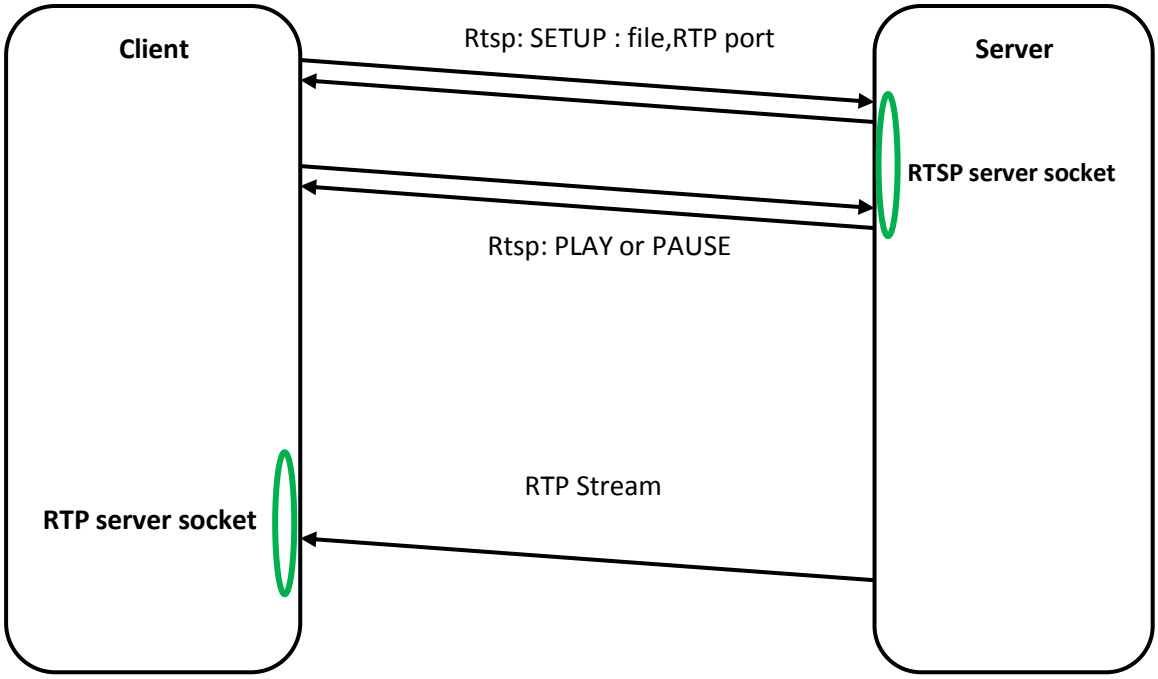
An event is triggered every time the counter increases by one. This event's handler contains the following code:

```
//Builds an RTPpacket object containing the frame

RTPpacket rtp_packet = new RTPpacket(MJPEG_TYPE, imagenb,
imagenb*FRAME_PERIOD, buf, image_length);

 //send the packet as a DatagramPacket over the UDP socket

senddp = new DatagramPacket(packet_bits, packet_length, ClientIPAddr,
RTP_dest_port);

RTPsocket.send(senddp);
```

❖Client

The client job is to send RSTP messages and receive RTP stream.

Message	Server action
SETUP	Extracts the video name and RTP port number and initiates a connection to this socket.
PLAY	Starts the timer.
PAUSE	Stops the timer.
TEARDOWN	Stops the timer and terminate connection.



Chain : client connects to RTSP socket .. sends a SETUP message containing the file and RTP socket .. server uses this RTP socket to connect to the client sends PLAY request .. server starts sending RTP datagram packets .. client receives them ,de packetize them and displays them on screen.

Programming part

Open the JAVA classes attached with this lab and read the code, understand it and try to run it.

Homework

Add the button "Stop\Start Fast Forward" that plays the video in faster rate than the play button ..

Complete code is given to you ,, you have to add the FF button to the existing interface.