

1. A Ariketa

"Client_V1.java"tik abiatuta, zer eskatzen da:

◆ **Bezereoaren klaseak betetzen dituen eginkizun guztiak aztertzea.**

Bezero programak hurrengo eginkizunak bete behar ditu:

1. TCP Konexioa abiarazi zerbitzariarekin (IP eta Portua eskatuz) Setup Botoia klikatzean
2. UDP Konexioa abiarazi RTP_RCV portuan (25000)
3. Play botoiaren bidez:
 - a. READY moduan badago: PLAY eskaera bidali zerbitzariari (Sekuentzia zenbaki egokiarekin)
 - b. Erantzuna zein den begiratu, OK bada "PLAYING" moduan jarri
 - c. Denbora neurtzen hasi
4. Pause botoiaren bidez:
 - a. PLAYING moduan badago: PAUSE eskaera bidali zerbitzariari (Sekuentzia zenbaki egokiarekin)
 - b. Erantzuna zein den begiratu, OK bada "READY" moduan jarri
 - c. Denbora neurtzeari utzi
5. Teardown botoiaren bidez:
 - a. Edozein modutan egonda: TEARDOWN eskaera bidali zerbitzariari (Sekuentzia zenbaki egokiarekin)
 - b. Erantzuna zein den begiratu, OK bada, denbora neurtzeari utzi eta programa itxi.

◆ **PLAY botoieko kodea aztertuz, PAUSE botoiak eragin zuzena izan daian, JAVA kodea gehitu, eta probatu.**

Hurrengo kodeak funtzionatu egin du:

```
class pauseButtonListener implements ActionListener {

    public void actionPerformed(ActionEvent e){

        System.out.println("Pause Button pressed!");

        if (state == PLAYING)
        {
            //increase RTSP sequence number
            RTSPSeqNb++;

            //Send PAUSE message to the server
            send_RTSP_request("PAUSE");

            //Wait for the response
            if (parse_server_response() != 200)
                System.out.println("Invalid Server Response");
            else
            {
                //change RTSP state and print out new state
                state = READY;
                System.out.println("New RTSP state: READY");
                //stop the timer
                timer.stop();
            }
        }
        //else if state != PLAYING then do nothing
    }
}
```

◆ **TEARDOWN botoiak eragin zuzena izan daian, JAVA kodea gehitu, eta probatu.**

Hurrengo kodeak funtzionatu egin du:

```
class tearButtonListener implements ActionListener {
    public void actionPerformed(ActionEvent e){
        System.out.println("Teardown Button pressed !");

        //increase RTSP sequence number
        RTSPSeqNb++;

        //Send TEARDOWN message to the server
        send_RTSP_request("TEARDOWN");

        //Wait for the response
        if (parse_server_response() != 200)
            System.out.println("Invalid Server Response");
        else {
            //change RTSP state and print out new state
            state = INIT;
            System.out.println("New RTSP state: INIT");

            //stop the timer
            timer.stop();

            //exit
            System.exit(0);
        }
    }
}
```

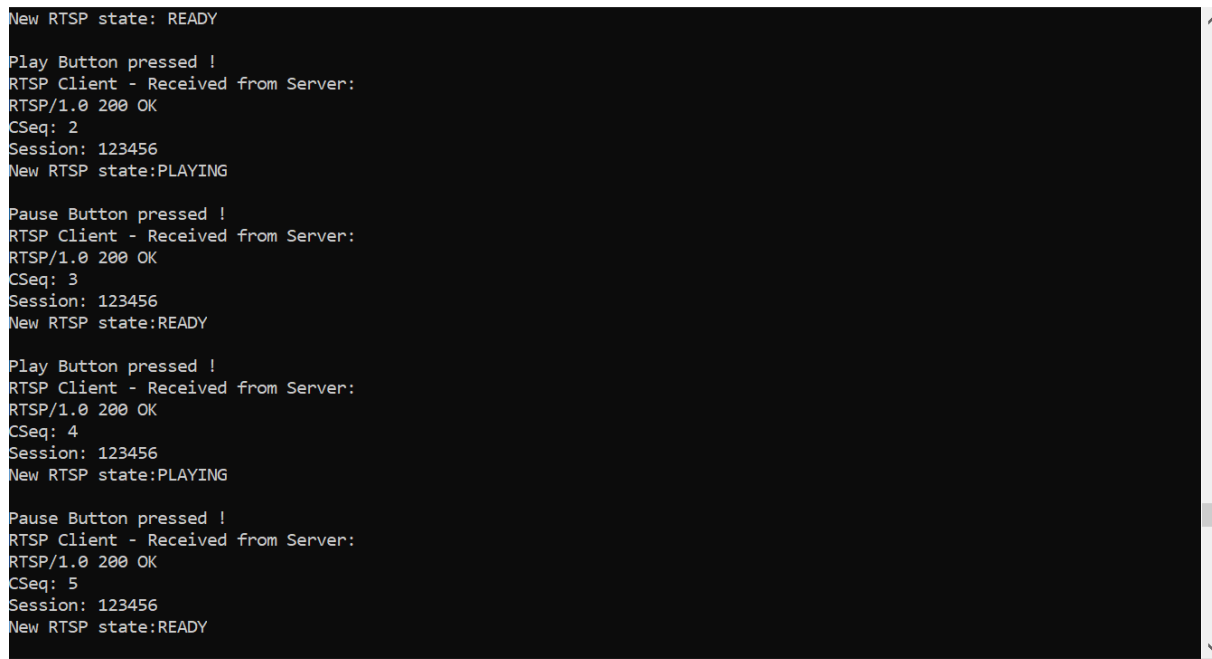
2. B Ariketa

Arterako saioan, RTSP bidez komunikatzen diran "bezero" eta "zerbitzaria" modulu bi JAVAn burutu ditugu.

Izan ere, bezeroak, RTSP protokoloa zerbitzariari agindua igortzeko erabiliko du, eta barriz, edukiak, DUP gaineko RTP bidez hartuko ditu.

Hori burutzeaz gain, ondo legoke komunikazioaren bilakaera gainbegiratu ahal izatea.

Lehenik, Client_v1.java dagoeneko gainbegiratzeko trazak aztertuko ditugu.



```
New RTSP state: READY

Play Button pressed !
RTSP Client - Received from Server:
RTSP/1.0 200 OK
CSeq: 2
Session: 123456
New RTSP state:PLAYING

Pause Button pressed !
RTSP Client - Received from Server:
RTSP/1.0 200 OK
CSeq: 3
Session: 123456
New RTSP state:READY

Play Button pressed !
RTSP Client - Received from Server:
RTSP/1.0 200 OK
CSeq: 4
Session: 123456
New RTSP state:PLAYING

Pause Button pressed !
RTSP Client - Received from Server:
RTSP/1.0 200 OK
CSeq: 5
Session: 123456
New RTSP state:READY
```

Laborategiko saio honetan, gainbegiratzeko parametroak gehituko ditugu: RTP pakete bakoitzaren tamaina, pakete bakoitzaren sekuentzia zenbakia, time-stamp-a, eraibilitko codec-a.. holako zerbait lortzeko eran.

```

S mbolo del sistema - java Client
Play Button pressed !
RTSP Client - Received from Server:
RTSP/1.0 200 OK
CSeq: 6
Session: 123456
New RTSP state:PLAYING
PayloadLength = 5085

Received RTP packet with SeqNum # 14 and TimeStamp 1400 ms, of type 34
10000000 00100010 00000000 00001110 00000000 00000000 00000101 01111000
PayloadLength = 5358

Received RTP packet with SeqNum # 15 and TimeStamp 1500 ms, of type 34
10000000 00100010 00000000 00001111 00000000 00000000 00000101 11011100
PayloadLength = 5685

Received RTP packet with SeqNum # 16 and TimeStamp 1600 ms, of type 34
10000000 00100010 00000000 00010000 00000000 00000000 00000110 01000000
PayloadLength = 6010

Received RTP packet with SeqNum # 17 and TimeStamp 1700 ms, of type 34
10000000 00100010 00000000 00010001 00000000 00000000 00000110 10100100
PayloadLength = 6334

Received RTP packet with SeqNum # 18 and TimeStamp 1800 ms, of type 34
10000000 00100010 00000000 00010010 00000000 00000000 00000111 00001000

Pause Button pressed !
RTSP Client - Received from Server:
RTSP/1.0 200 OK
CSeq: 7
Session: 123456
New RTSP state:READY

```

“Client_V1.java”tik abiatuta, zera eskatzen da

1. Dagoeneko gainbegiratzen diren parametroen azalpena

Kontrol mezuen gainbegiraketa egiten da dagoenarekin. Hauen sekuentzia zenbakiak, HTTP erantzunak eta komandoak agertzen dira.

2. RTP paketeen karga baliogarria gainbegiratzea

Tamaina egokia dela da horretarako erarik errazena. Inplementatutako metodoan horixe egiten da.

3. Sekuentzia zenbakiak gainbegiratzea

RTP paketeentzako getSequenceNumber erabiliz lortu daiteke. Ordenean daudela da garrantzitsua. Hauetan saltoak egotea pakete galeraren indikazioa da. Ordenez kanpo egotea jitter oso handia dagoelaren adierazlea da.

4. Time-Stamp gainbegiratzea

Zerbitzaria frame guztiak denbora egokian egin ditzakeela ziurtatzeko balio du. Gure kasuan, 100ms-ko denbora du frame artean (10 FPS).

5. Codec-a gainbegiratzea

Goiburua gainbegiratzeaz gain, mota ere kontrolatu dezakegu. Era horretan, espero den kodifikazio formatua (Mjpeg=34, kasu honetan) jasoko dela ziurtatzeko

6. RTP goiburuko bitak gainbegiratzea, behin SSRC eremua kenduta.

Goiburuan datu hauek guztiak daude. Azken 4 byte-ak Timestamp dira, horren aurreko 2-ak sekuentzia zenbakia, horren aurreko 7-ak mota... eta lehen biak bertsioa dira – berriena 2 (10) da.

7. Amaieran, honako gainbegiratze aukerak ditugu:

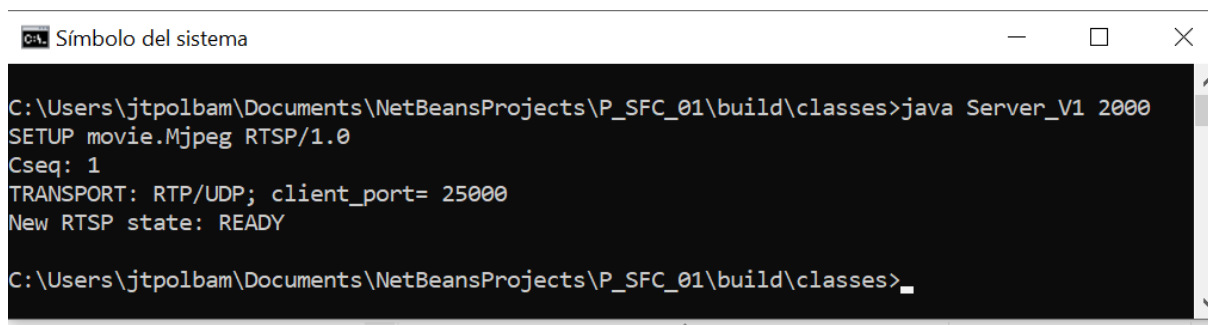
Balio horiek guztiak emandako metodoekin lor daitezke. Kodean begiratzuz zehatz-mehatz zelan egin den ikus daiteke – timerListener metodoaren barruan dago guztia.

```
#=====#  
Received RTP packet!  
Size: 6010  
Sequence Number: 17  
Packet timestamp: 1700ms  
Payload type: 34  
10000000 00100010 00000000 00010001 00000000 00000000 00000110 10100100  
#=====#
```

Ariketa honeek, *class timerListener implements ActionListener* Client_V1.javako klasean gauzatuko dira.

3. C Ariketa

Horretarako, Server_v1.java fitxategitik abiatuko gara, eta dagoeneko funtzionamendua aztertuko dugu bezero batek bideo zerbitzu bat eskatzen dionean. Ondoko irudiko moduko jarduera ikusiko dugu. Eta irteteko prozesua hil beharko dugu.



```

C:\Users\jtpolbam\Documents\NetBeansProjects\P_SFC_01\build\classes>java Server_V1 2000
SETUP movie.Mjpeg RTSP/1.0
Cseq: 1
TRANSPORT: RTP/UDP; client_port= 25000
New RTSP state: READY
C:\Users\jtpolbam\Documents\NetBeansProjects\P_SFC_01\build\classes>

```

Baina zerbitzariak ez die bezeroaren botoi bidezko eskariei erantzunik emango.

“Server_V1.java”tik abiatuta, zera eskatzen da:

1. PLAY, PAUSE, eta TEARDOWN atazak burutzeko kodigoa sortzea.

a. Eskariak eta estatuak aztertu ataza bakoitzeko.

Bezeroaren era berean, eskariak estatu berezietan etorri behar dira: Ezin dugu pausatu erreproduzitzen ari ez den zeozer.

b. RTSP erantzuna igorri bezeroari atazako

Bezeroaren eskaera guztiei erantzun egingo diogu, momentu egokian heldu badira.

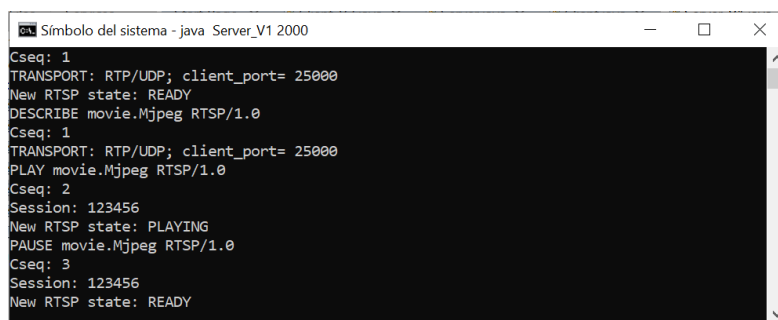
c. Timer-a abiarazi edo gelditu atazako

Bezeroan moduan, timer.start() eta timer.stop() egingo dugu.

d. Estatua eguneratu atazak

Eskaeraren arabera estatuak aldatuko ditugu.

Ariketa honek, CONNECT botoiaren antzeko egitura izango du. Eta azkeneko emaitza ondoko irudikoa modukoa izan behar da.



```

Símbolo del sistema - java Server_V1 2000
Cseq: 1
TRANSPORT: RTP/UDP; client_port= 25000
New RTSP state: READY
DESCRIBE movie.Mjpeg RTSP/1.0
Cseq: 1
TRANSPORT: RTP/UDP; client_port= 25000
PLAY movie.Mjpeg RTSP/1.0
Cseq: 2
Session: 123456
New RTSP state: PLAYING
PAUSE movie.Mjpeg RTSP/1.0
Cseq: 3
Session: 123456
New RTSP state: READY

```

Kode snippet-a:

```
while(true)
{
    request_type = theServer.parse_RTSP_request();

    if((request_type == PLAY) && (state == READY)){
        theServer.send_RTSP_response();
        theServer.timer.start();
        state = PLAYING;
        System.out.println("NEW RTSP STATE: PLAYING");
    }
    if(request_type == PAUSE){
        theServer.send_RTSP_response();
        theServer.timer.stop();
        state = READY;
        System.out.println("NEW RTSP STATE: PAUSED(READY)");
    }
    if(request_type == TEARDOWN){
        theServer.send_RTSP_response();
        theServer.timer.stop();

        System.out.println("BYE-BYE! KAPUT!");

        theServer.RTPsocket.close();
        System.exit(0);
    }
}
```


2. Azkenik, RTP paketeen gainbegiratzea gauzatzea eskatzen da.

Horretarako, arterako saioko *public void actionPerformed(ActionEvent e)* eta *public void printHeader()* metodoak hartuko ditugu kontutan, eta emaitza ondoko irudikoa modukoa izango da.

```

0
SETUP movie.Mjpeg RTSP/1.0
Cseq: 1
TRANSPORT: RTP/UDP; client_port= 25000
New RTSP state: READY
DESCRIBE movie.Mjpeg RTSP/1.0
Cseq: 1
TRANSPORT: RTP/UDP; client_port= 25000
PLAY movie.Mjpeg RTSP/1.0
Cseq: 2
Session: 123456
New RTSP state: PLAYING
10000000 00100010 00000000 00000001 00000000 00000000 00000000 01100100
10000000 00100010 00000000 00000010 00000000 00000000 00000000 11001000
10000000 00100010 00000000 00000011 00000000 00000000 00000001 00101100
10000000 00100010 00000000 00000100 00000000 00000000 00000001 10010000
10000000 00100010 00000000 00000101 00000000 00000000 00000001 11110100
10000000 00100010 00000000 00000110 00000000 00000000 00000010 01011000
10000000 00100010 00000000 00000111 00000000 00000000 00000010 10111100
10000000 00100010 00000000 00001000 00000000 00000000 00000011 00100000
10000000 00100010 00000000 00001001 00000000 00000000 00000011 10000100
10000000 00100010 00000000 00001010 00000000 00000000 00000011 11101000
10000000 00100010 00000000 00001011 00000000 00000000 00000100 01001100
10000000 00100010 00000000 00001100 00000000 00000000 00000100 10110000
10000000 00100010 00000000 00001101 00000000 00000000 00000101 00010100
PAUSE movie.Mjpeg RTSP/1.0
Cseq: 3
Session: 123456
  
```

Bezero kodean egindako gauza bera erabili dezakegu zerbitzarian, aldagaien izenak zuzenduta:

```

#=====#
SENT RTP packet!
Size: 6830
Sequence Number: 20
Packet timestamp: 2000ms
Payload type: 34
10000000 00100010 00000000 00010100 00000000 00000000 00000111 11010000
#=====#
  
```

4. Ondorioak

RTSP erabiltzea oso erraza da eskuragarri ditugun liburutegi, aplikazio eta paketeekin. One-To-Many bideo streaming sistemak egiteko tresna baliagarria izan daiteke, kontrolerako seinaleztapen mezu sinple eta estandarrak mantenduz.

Praktikan ikasi dugun moduan, sistema hauen oinarrizko funtzioak inplementatzea programa oso arinetan egin daiteke.

Azkenik, goiburu guztia aztertuz barne funtzionamendua hobe ulertzeko aukera izan dugu, protokoloaren mekanismoak martxan ikusteko eta haien arteko elkartrukeak aztertzeko.