VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Computer Networks Lab (CO3094)

Assignment 1 (Semester 211)

BUILDING A REAL-TIME STREAMING APPLICATION

Advisor:	Mr. Nguyễn Lê Duy Lai	
Students:	Nguyễn Thanh Ngân	1911667
	Phạm Nhựt Huy	1952059
	Nguyễn Hoàng	1952255

HO CHI MINH CITY, NOVEMBER 2021

Contents

1 Requirement analysis

The tables below list all the functional and non-functional requirements specified. Since some functions are implemented for us in advance, we will be focusing on the requirements that we need to implement.

No.	Requirement
1	The Client application has a GUI
2	The Client application can display the video
3	The Client application can pause the video
4	The Client application can terminate the session its side
5	The Client application can display video total time and remaining time
6	The Client application can forward and backward the video
7	The Client application can switch to a new video from a list returned by the Server
8	The Client can request and receive session description
9	The Server can send back a session description file when client requests
10	The Server can return a list of available videos to the client
11	The Server can switch to a new video when client requests

Table 1: Functional requirements

No.	Requirement
1	The Client application includes only 3 buttons: Play, Pause and Stop
2	The Client must send a SETUP request to server to start a session
3	The Client has a scrollbar to forward and backward the video
4	The server must RTP-packetize the video data and send it to the client's RTP port
5	Both the Server and Client application support the DESCRIBE method (describe the session)
6	The session must change to a new state (aside from INIT, READY and PLAYING) when switching to a new video

Table 2: Non-functional requirements

2 Application description

2.1 Communication between Client & Server

2.1.1 RTSP methods

In total, we have implemented 8 RTSP methods of communication between the Client and Server:

- SETUP: Initiate a session between Client and Server

  Example:

```
SETUP movie1.Mjpeg RTSP/1.0
CSeq: 0
Transport: RTP/UDP; client_port= 8002
```

- PAUSE: Pause the video streaming

  Example:

```
PAUSE movie2.Mjpeg RTSP/1.0
CSeq: 5
Session: 613038
```

- PLAY: Continue/Start the video streaming

  Example:

```
PLAY movie2.Mjpeg RTSP/1.0
CSeq: 4
Session: 613038
```

- TEARDOWN: Terminating the current session

  Example:

```
TEARDOWN movie1.Mjpeg RTSP/1.0
CSeq: 12
Session: 613038
```

- FORWARD: Forward or backwarding the video streaming

  Example:

```
FORWARD movie2.Mjpeg RTSP/1.0
CSeq: 7
Session: 613038
Frame: 31
```

- LIST: Request the list of available videos from Server

  Example:

```
LIST RTSP/1.0
CSeq: 1
Session: 613038
```

- DESCRIBE: Request a session description from Server

  Example:

```
DESCRIBE RTSP/1.0
CSeq: 2
Session: 613038
```

- SWITCH: Switch to a new video stream

  Example:

```
SWITCH movie1.Mjpeg RTSP/1.0
CSeq: 9
Session: 613038
```

2.1.2 RTSP response

If a request is processed successfully, the server will returns an RTSP message with the following format

```
RTSP/1.0 200 OK
CSeq: <RTSP request sequence number>
Session: <session number>
<response body>
```

in which the response body is JSON-formatted data returned by the Server. For example, after a SETUP request is processed successfully, the Server may send back a message that looks like this.

```
RTSP/1.0 200 OK
CSeq: 0
Session: 1000
{"nFrame": 5000, "duration": 31}
```

2.1.3 Session

A session starts counting from the time when SETUP request is sent and stops when the Client sends TEARDOWN request. There are 4 possible states in a session between Client and Server - namely, INIT, READY, PLAYING and PENDING (see the diagram below).

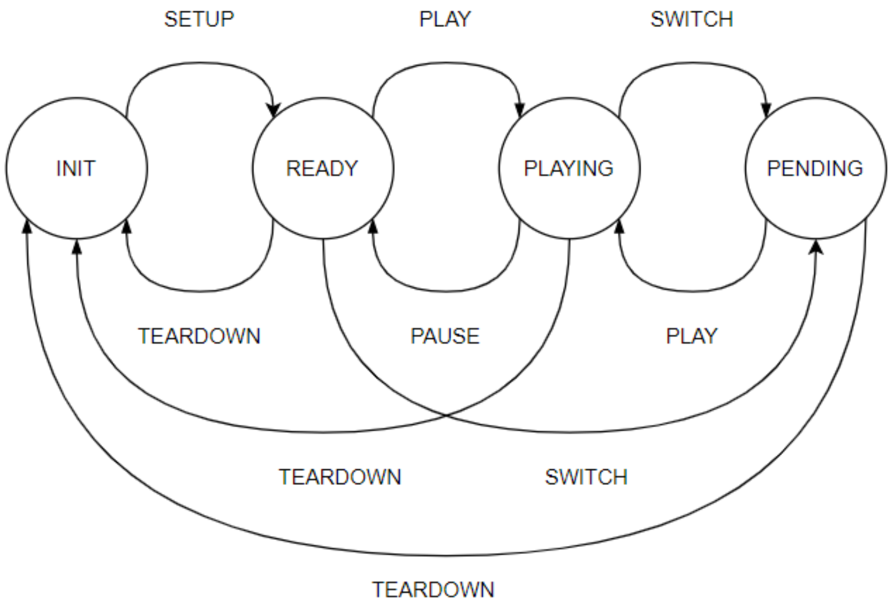


Figure 1: State diagram of a session

2.2 Main functionalities of Client

Our Client application has 7 main functionalities:

- Playing video
- Pausing video
- Forwarding and backwarding video
- Terminating session
- Requesting session description
- Listing available videos
- Switching to a new video

2.2.1 Playing video

Our application receives the video frames one by one from the server via the RTP protocol and displays them on the screen (see the figure below). Every time the user presses the Play button, the video will start playing. If the session is not yet initiated when user presses Play, a SETUP request will be sent before the PLAY request. While the video is playing, user can drag the slider on the time progress bar to jump to a specific time in the video.

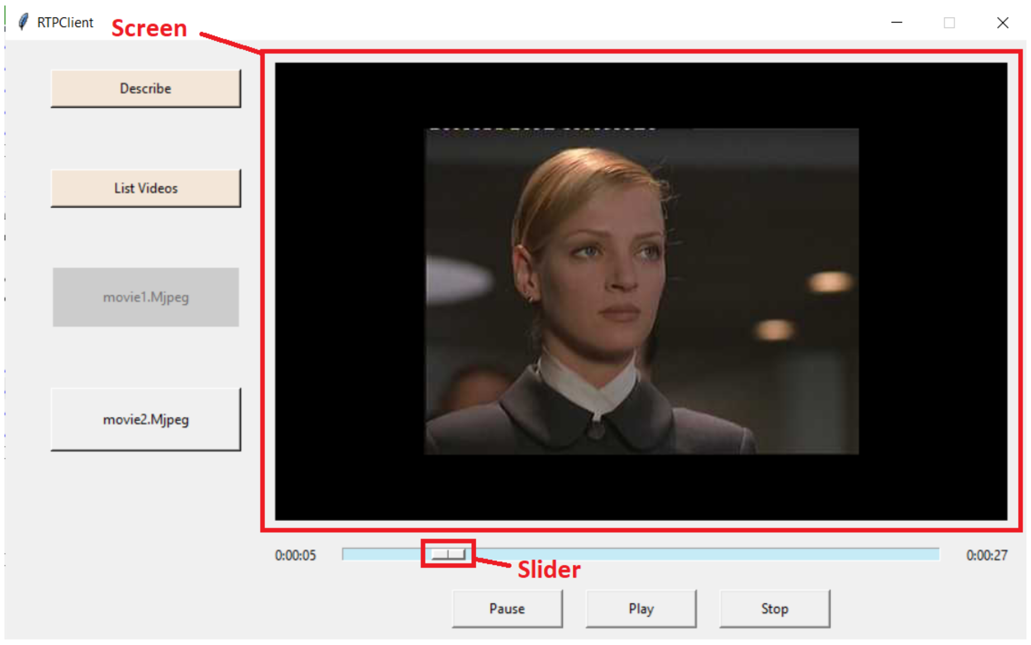


Figure 2: The GUI of the Client application when user presses Play

2.2.2 Pausing video

The video is paused when the user presses the Pause button. The client application only sends a PAUSE request to the server if the current state is PLAYING. When the video is paused the video progress bar is freezed, so user cannot backward or forward the video when it is paused.

2.2.3 Forwarding and backwarding video

The slider on the time progress bar is used to jump to a specific time in the video. This function is available only when the state is PLAYING, otherwise the progress bar is freezed. When user releases the slider, Client will send a FORWARD request to Server which specifies which frame to jump to.

2.2.4 Terminating session

When user presses Stop, a TEARDOWN request is issued if there is an ongoing session between Client and Session. The Server will close the RTSP connection and terminates its worker thread when it receives a TEARDOWN request from the Client.

2.2.5 Requesting session description

We include a Describe button in the GUI so that user can issue a DESCRIBE request and see the session description file displayed in terminal. The description file follows the Session Description Protocol (SDP). There are 6 fields in a description file generated by our Server.

```
v = (protocol version number, currently only 0)
o = (owner/creator and session identifier)
```

```
 s = (session name)
 a = (zero or more session attribute lines)
 t = (time the session is active)
m = (media name and transport address)
```

A typical description file may look like this

```
 v=0
 o=elnosabe 2890844526 2890842807 IN IP4 126.16.64.4
 s=Mjpeg Video Stream
 t=2873397496 2873404696
 a=recvonly
m=video 8002 RTP/AVP 26
```

Specifications of the Session Description Protocol can be found in RFC 8866 [Begen et al., 2021].

2.2.6 Listing available videos

Aside from the buttons specified in one of the non-functional requirements, we also include a "List Videos" button which is used to send a LIST request to the Server. The server will response with a list of videos available for streaming.

2.2.7 Switching to a new video

While streaming, the Client application can send a SWITCH request to change the video stream. This function works when session state is READY or PLAYING. In the GUI, there are buttons corresponding to different videos that user can choose to replace the current video.

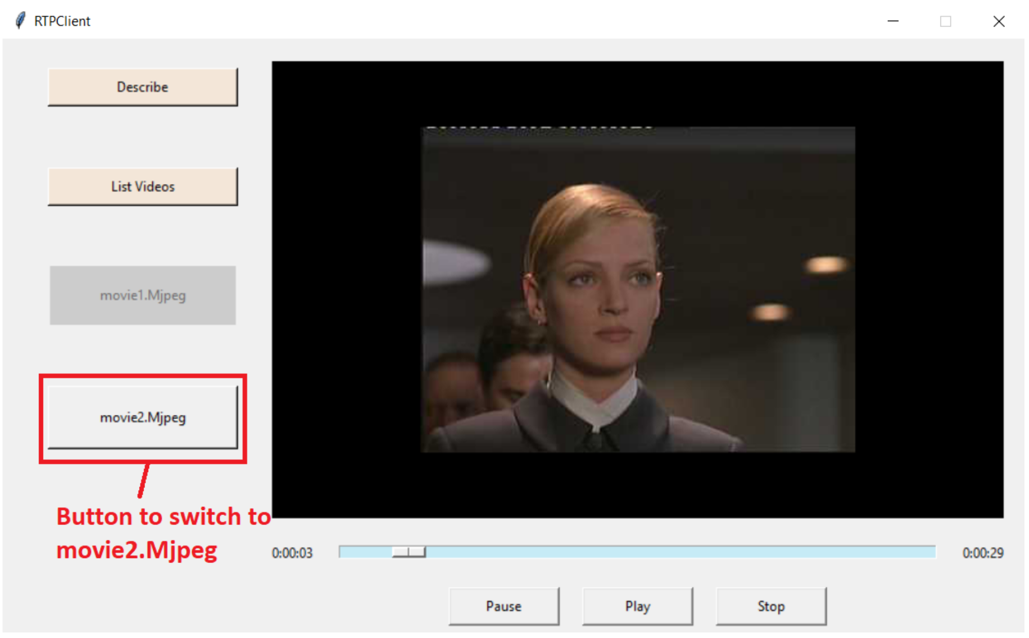


Figure 3: Buttons for switching are displayed on the left side of the screen

The button corresponding to the current video is disabled (greyed out).

3 List of components

The system can be decomposed into many components. Below is the list of components that we have included in our final product.

- Client application:
  - Graphical User Interface (GUI)
  - RTP packet receiver
  - Video renderer
- Server:
  - RTSP request handler
  - RTP packet sender
- RTP packet handler (packetization & de-packetization)
- Video stream

4 Data flow

The following data flow diagram (DFD) illustrates the main data flows in our system.

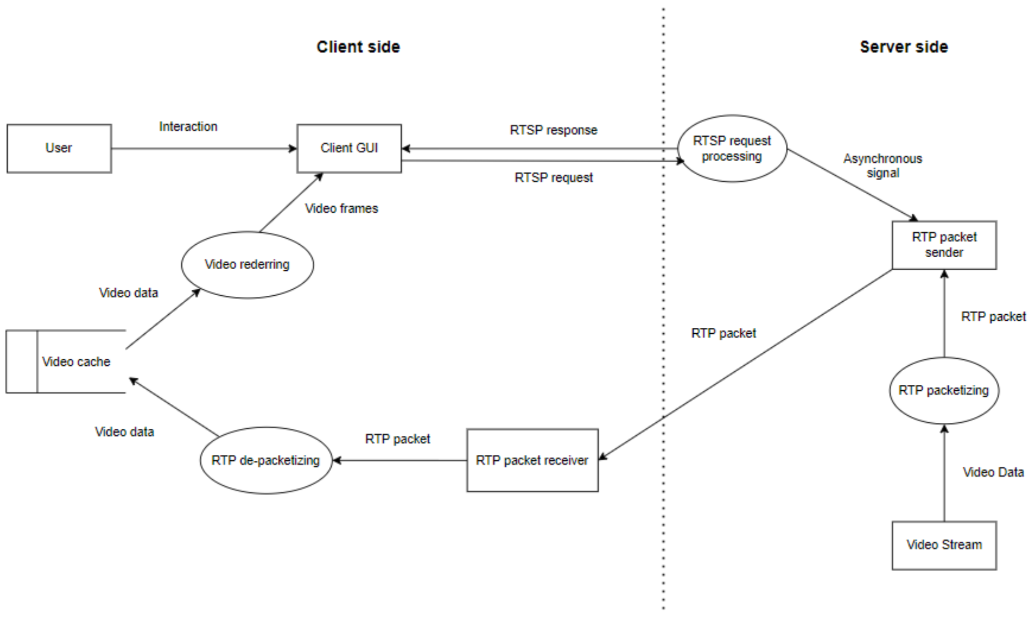


Figure 4: Data flow diagram

5 Class diagram

Below is the class diagram of our system.

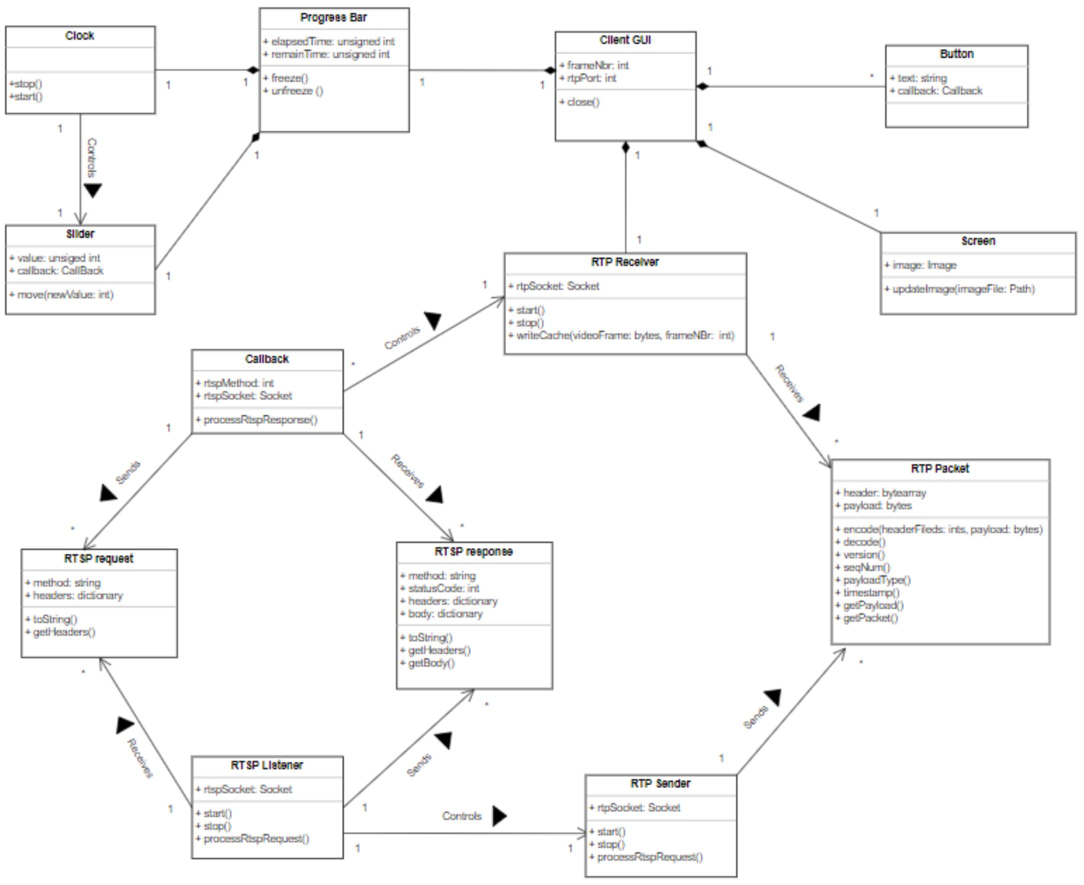


Figure 5: Class diagram

6 Evaluation

In order to evaluate the performance of our system, we have calculated some statistics about a session between Client and Server.

- Packet loss rate:

  The packet loss rate is calculated based on the logging information displayed in the terminals of Client and Server. Below are the logs printed out during one of our test runs.

Listing 1: Client's log

```
Data received:
RTSP/1.0 200 OK
CSeq: 0
Session: 200925
{"nframe": 500, "duration": 31}

Data received:
RTSP/1.0 200 OK
CSeq: 1
Session: 200925

Data received:
RTSP/1.0 200 OK
CSeq: 2
```

```
Session: 200925

=============== SESSION LOG ===============
Session ID: 200925
Number of RTP packets received: 337

Data received:
RTSP/1.0 200 OK
CSeq: 0
Session: 938117
{"nframe": 500, "duration": 31}

Data received:
RTSP/1.0 200 OK
CSeq: 1
Session: 938117

Data received:
RTSP/1.0 200 OK
CSeq: 2
Session: 938117

Data received:
RTSP/1.0 200 OK
CSeq: 3
Session: 938117

Data received:
RTSP/1.0 200 OK
CSeq: 4
Session: 938117

Data received:
RTSP/1.0 200 OK
CSeq: 5
Session: 938117

Data received:
RTSP/1.0 200 OK
CSeq: 6
Session: 938117

=============== SESSION LOG ===============
Session ID: 938117
Number of RTP packets received: 252

Data received:
RTSP/1.0 200 OK
CSeq: 0
Session: 588016
{"nframe": 500, "duration": 31}

Data received:
RTSP/1.0 200 OK
CSeq: 1
Session: 588016
```

```
Data received:
RTSP/1.0 200 OK
CSeq: 2
Session: 588016
{"list": ["movie1.Mjpeg", "movie2.Mjpeg"]}

Data received:
RTSP/1.0 200 OK
CSeq: 3
Session: 588016
{"nframe": 240, "duration": 15}

Data received:
RTSP/1.0 200 OK
CSeq: 4
Session: 588016

Data received:
RTSP/1.0 200 OK
CSeq: 5
Session: 588016

============== SESSION LOG ==============
Session ID: 588016
Number of RTP packets received: 108
```

Listing 2: Server's log

```
Data received:
SETUP movie1.Mjpeg RTSP/1.0
CSeq: 0
Transport: RTP/UDP; client_port= 8002

processing SETUP

OK 200

Data received:
PLAY movie1.Mjpeg RTSP/1.0
CSeq: 1
Session: 200925

processing PLAY

OK 200

Data received:
TEARDOWN movie1.Mjpeg RTSP/1.0
CSeq: 2
Session: 200925

processing TEARDOWN

OK 200
```

```
Session with client (127.0.0.1, 8002) closed.

============== SESSION LOG ==============
Client: (127.0.0.1, 8002)
Number of RTP packets sent: 337

Data received:
SETUP movie1.Mjpeg RTSP/1.0
CSeq: 0
Transport: RTP/UDP; client_port= 8002

processing SETUP

OK 200

Data received:
PLAY movie1.Mjpeg RTSP/1.0
CSeq: 1
Session: 938117

processing PLAY

OK 200

Data received:
PAUSE movie1.Mjpeg RTSP/1.0
CSeq: 2
Session: 938117

processing PAUSE

OK 200

Data received:
PLAY movie1.Mjpeg RTSP/1.0
CSeq: 3
Session: 938117

processing PLAY

OK 200

Data received:
FORWARD movie1.Mjpeg RTSP/1.0
CSeq: 4
Session: 938117
Frame: 265

processing FORWARD

OK 200

Data received:
FORWARD movie1.Mjpeg RTSP/1.0
CSeq: 5
Session: 938117
```

```
Frame: 95

processing FORWARD

OK 200

Data received:
TEARDOWN movie1.Mjpeg RTSP/1.0
CSeq: 6
Session: 938117

processing TEARDOWN

OK 200

Session with client (127.0.0.1, 8002) closed.

============== SESSION LOG ==============
Client: (127.0.0.1, 8002)
Number of RTP packets sent: 252

Data received:
SETUP movie1.Mjpeg RTSP/1.0
CSeq: 0
Transport: RTP/UDP; client_port= 8002

processing SETUP

OK 200

Data received:
PLAY movie1.Mjpeg RTSP/1.0
CSeq: 1
Session: 588016

processing PLAY

OK 200

Data received:
LIST RTSP/1.0
CSeq: 2
Session: 588016

processing LIST

OK 200

Data received:
SWITCH movie2.Mjpeg RTSP/1.0
CSeq: 3
Session: 588016

processing SWITCH

OK 200
```

```
 Data received:
PLAY movie2.Mjpeg RTSP/1.0
 CSeq: 4
 Session: 588016

 processing PLAY

OK 200

 Data received:
TEARDOWN movie2.Mjpeg RTSP/1.0
 CSeq: 5
 Session: 588016

 processing TEARDOWN

OK 200

 Session with client (127.0.0.1, 8002) closed.

=============== SESSION LOG ===============
 Client: (127.0.0.1, 8002)
 Number of RTP packets sent: 108
```

In all three sessions that have been logged, the number of RTP packets received by the Client equals to the number of packets sent by Server. While we were testing, we sometimes encountered cases in which 1 or 2 RTP packets are lost. These cases mostly happen when we abruptly close the Client by closing the GUI window or pressing Ctrl-C to shut the application down.

Thus, we can conclude that the packet loss rate is approximately 0.

- Frames per second: We also calculated the frame rate of a video by letting it run uninterrupted to completion and logging the elapsed time on client-side.

  We measured the duration of two Mjpeg videos - namely, movie1.Mjpeg and movie2.Mjpeg.

Listing 3: Client's log when streaming movie1.Mjpeg

```
 Data received:
RTSP/1.0 200 OK
 CSeq: 0
 Session: 454512
 {"nframe": 500, "duration": 32}

 Data received:
RTSP/1.0 200 OK
 CSeq: 1
 Session: 454512

 Video duration: 31.216107845306396 (seconds)
 Data received:
RTSP/1.0 200 OK
 CSeq: 2
 Session: 454512

=============== SESSION LOG ===============
Session ID: 454512
Number of RTP packets received: 500
```

Listing 4: Client's log when streaming movie2.Mjpeg

```
Data received:
RTSP/1.0 200 OK
CSeq: 3
Session: 959613

Video duration: 14.977200269699097 (seconds)
Data received:
RTSP/1.0 200 OK
CSeq: 4
Session: 959613

=============== SESSION LOG ===============
Session ID: 959613
Number of RTP packets received: 240
```

The movie1.Mjpeg video contains 500 frames and movie2.Mjpeg contains 240 frames. Thus, the number of frames per second for each video is

– movie1.Mjpeg:
$$\frac{31.2161078453063 96}{500} \approx 0.0624 \text{ (frames/second)}$$

– movie2.Mjpeg:
$$\frac{14.977200269699097}{240} \approx 0.0624 \text{ (frames/second)}$$

According to the calculation results, the number of frames per second is approximately 0.0624. The video duration includes the time to fetch and render video frames one by one. Thus, this number is affected by connection bandwidth and many other factors.

7 User manual

In order to test the system, user must have the required a Python interpreter and all required Python packages installed. The packages needed for this application has been listed in the *requirements.txt* file. User can install these packages using the following command line.

```
pip install -r requirements.txt
```

After this, user needs to start the server by running the *Server.py* script. The command line for this task looks like this.

```
python Server.py <Server's RTSP port>
```

As recommended in the problem text, user should choose a port number greater 1024 for Server's RTSP port. The command line for launching Client application is

```
python ClientLauncher.py <Server's address> \
<Server's RTSP port> \
<Client's RTP port> \
<name of video>
```

User must provide all 4 required arguments in order to launch the Client. After successfully launching the Client application, a GUI window will be displayed.

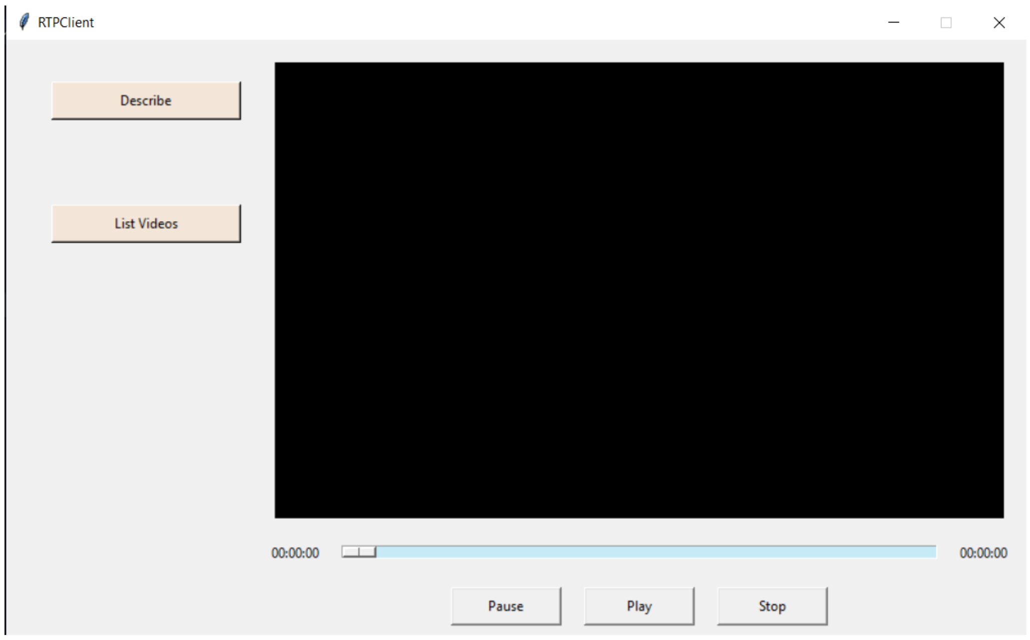
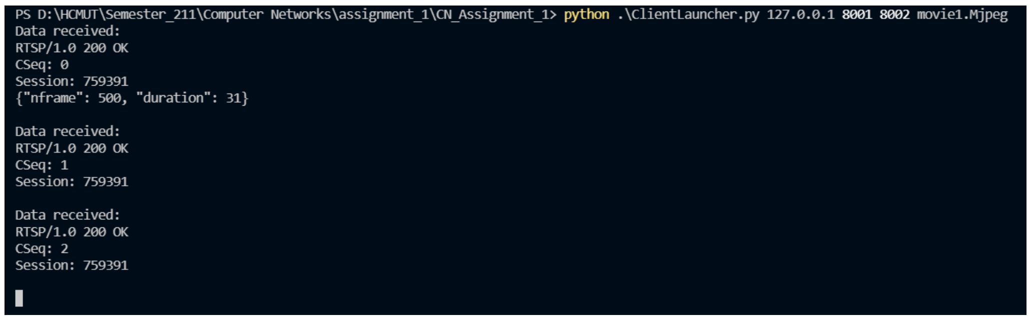


Figure 6: The GUI window at startup

The user can now interact with the GUI and explore all the functions of the application. Logging information is displayed in the terminals of Client and Server, so user can see how they are communicating over the RTSP protocol.

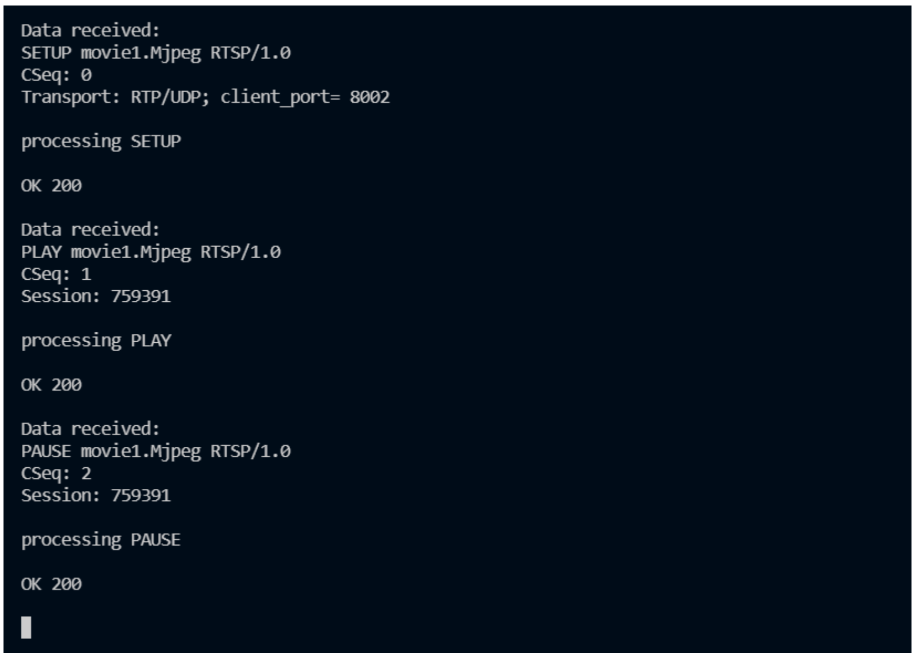


```
PS D:\HCMUT\Semester_211\Computer Networks\assignment_1\CN_Assignment_1> python .\ClientLauncher.py 127.0.0.1 8001 8002 movie1.Mjpeg
Data received:
RTSP/1.0 200 OK
CSeq: 0
Session: 759391
{"nframe": 500, "duration": 31}

Data received:
RTSP/1.0 200 OK
CSeq: 1
Session: 759391

Data received:
RTSP/1.0 200 OK
CSeq: 2
Session: 759391
```

Figure 7: Client's terminal



```
Data received:
SETUP movie1.Mjpeg RTSP/1.0
CSeq: 0
Transport: RTP/UDP; client_port= 8002

processing SETUP

OK 200

Data received:
PLAY movie1.Mjpeg RTSP/1.0
CSeq: 1
Session: 759391

processing PLAY

OK 200

Data received:
PAUSE movie1.Mjpeg RTSP/1.0
CSeq: 2
Session: 759391

processing PAUSE

OK 200
```

Figure 8: Server's terminal

References

[Begen et al., 2021] Begen, A., Kyzivat, P., Perkins, C., and Handley, M. (2021). SDP: Session description protocol. Technical report.