Homework 3

Nov 9, 2018 (Fri)

Eunggu Yun

1. Introduction

There are three programs: RTSPServer, RTSPClient and RTSPReceiver. These three programs are a server program and client programs which are using UDP protocol and (UDP based) RTSP protocol to communicate each other. RTSPServer program is a server program that waits on the port 8000 for a client and streams the requested video file to the client. RTSPClient program is a client program that requests to server using the specified IP address and the port and receives the video stream using RTSPReceiver. When these two programs get contact to each other, they transfer **the hello messages**, **the video list,** and **the video file name**. After that, RTSPServer transfers **the video stream** and RTSPReceiver receives it. After video streaming is ended, all programs are terminated.

Attention 1: RTSPServer only supports **Matroska Multimedia Container (*.mkv)** videos.

Attention 2: RTSPClient sometimes failes to start receiving a video stream (It terminates as soon as streaming starts). If that happens, please just try again.

Attention 3: Streaming takes a little while, so it seems to be stopped, but it actually works. Please wait a minute.

2. Development Environment

1) Platform (OS): Ubuntu 16.04.1 LTS (GNU/Linux 4.4.0-31-generic x86_64)

2) Compiler: gcc 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.10)

This is the SKKU server's development environment. (swye.skku.edu)

3. How to build and run

1) Build

To build the server and client program:

```
$ make all
```

To build the server program only:

```
$ make server
```

To build the client programs only:

```
$ make client
```

To clean up output files:

```
$ make clean
```

2) Run

To run the server program,

```
$ ./RTSPServer
```

To run the client program,

```
$ ./RTSPClient [server_ip] [port_number]
```

For example,

```
$ ./RTSPClient 127.0.0.1 8000
```

4. Project Structure

```
Project Folder
 ├ live
 ├ Makefile
 ├ RTSPClient.cpp
 ├ RTSPReceiver.cpp
 ├ RTSPServer.cpp
 └ video
```

1) live

The "LIVE555 Streaming Media" library folder. It provides many RTSP based functionalities related to video streaming.

2) Makefile

File for GNU Make. It gives the knowledge of how to build RTSPServer program, RTSPClient program, RTSPReceiver program, and the "LIVE555 Streaming Media" library to GCC compiler.

3) RTSPClient.cpp

A source code of RTSPClient. RTSPClient program is a client program that requests video file name to be streamed and execute RTSPReceiver program as a child process to receive the stream.

4) RTSPReceiver.cpp

A source code of RTSPReceiver. RTSPReceiver program is not designed to be run on its own. It is subprogram of RTSPClient.

5) RTSPServer.cpp

A source code of RTSPServer. RTSPServer program is a server program that sends video file list and receives the request. Then, it streams the requested video file to client.

6) video

A folder to store videos to be streamed by RTSPServer. Only **Matroska Multimedia Container (*.mkv)** videos are valid. Other file extensions are not supported by RTSPServer.

5. Implementation Details

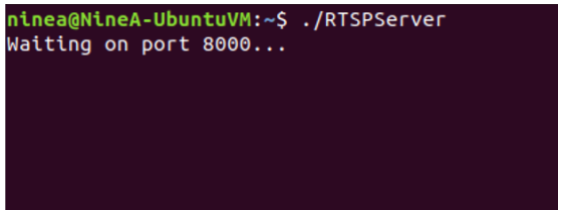
When RTSPServer program starts, it listens for client on ports 8000 using UDP socket. After it receives the hello message from a client, it also sends the hello message to the client and waits for a video list request. When it receives a request for a video list from client, it reads video folder to find the **Matroska Multimedia Containder (*.mkv)** videos. And then, it makes the video list and sends it to client. After it receives a request for video streaming, it sends a response (OK or not) and starts video streaming using RTSP protocol. When the streaming ends, it terminates.

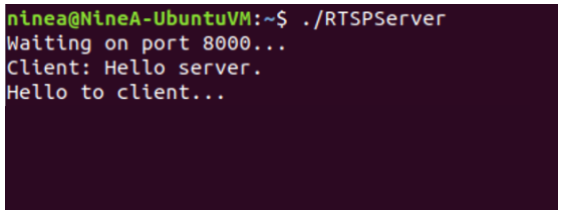
RTSPClient gets two arguments: server IP address and port number. When it starts, it creates a UDP socket and sends a hello message to the server. When it receives the hello message from server, it requests a video list to server and receives it. And then, it shows the video list to the user and gets the video file name desired by the user. After the user choice, it requests the video streaming and receives a response (OK or not). If the response is OK, RTSPClient executes RTSPReceiver to receive the video stream using RTSP protocol. When the streaming ends, both RTSPClient and RTSPReceiver terminate.

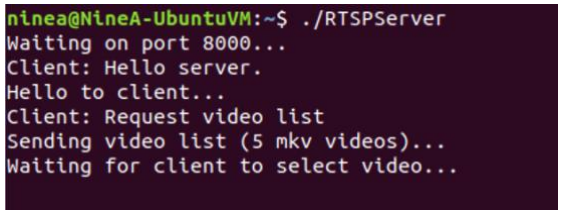
6. Screen Shots

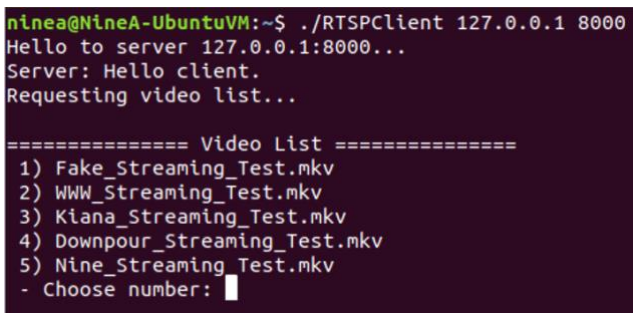
Client

Server









```
ninea@NineA-UbuntuVM:~$ ./RTSPClient 127.0.0.1 8000
Hello to server 127.0.0.1:8000...
Server: Hello client.
Requesting video list...

============== Video List ==============
 1) Fake_Streaming_Test.mkv
 2) WWW_Streaming_Test.mkv
 3) Kiana_Streaming_Test.mkv
 4) Downpour_Streaming_Test.mkv
 5) Nine_Streaming_Test.mkv
 - Choose number: 4
========================================

Requesting video Downpour_Streaming_Test.mkv...
```

```
ninea@NineA-UbuntuVM:~$ ./RTSPServer
Waiting on port 8000...
Client: Hello server.
Hello to client...
Client: Request video list
Sending video list (5 mkv videos)...
Waiting for client to select video...
Client: Request Downpour_Streaming_Test.mkv
Streaming video...
```

```
ninea@NineA-UbuntuVM:~$ ./RTSPClient 127.0.0.1 8000
Hello to server 127.0.0.1:8000...
Server: Hello client.
Requesting video list...

============== Video List ==============
 1) Fake_Streaming_Test.mkv
 2) WWW_Streaming_Test.mkv
 3) Kiana_Streaming_Test.mkv
 4) Downpour_Streaming_Test.mkv
 5) Nine_Streaming_Test.mkv
 - Choose number: 4
========================================

Requesting video Downpour_Streaming_Test.mkv...
Streaming video...
```

```
ninea@NineA-UbuntuVM:~$ ./RTSPServer
Waiting on port 8000...
Client: Hello server.
Hello to client...
Client: Request video list
Sending video list (5 mkv videos)...
Waiting for client to select video...
Client: Request Downpour_Streaming_Test.mkv
Streaming video...
Streaming complete.
ninea@NineA-UbuntuVM:~$
```

```
ninea@NineA-UbuntuVM:~$ ./RTSPClient 127.0.0.1 8000
Hello to server 127.0.0.1:8000...
Server: Hello client.
Requesting video list...

============== Video List ==============
 1) Fake_Streaming_Test.mkv
 2) WWW_Streaming_Test.mkv
 3) Kiana_Streaming_Test.mkv
 4) Downpour_Streaming_Test.mkv
 5) Nine_Streaming_Test.mkv
 - Choose number: 4
========================================

Requesting video Downpour_Streaming_Test.mkv...
Streaming video...
Streaming complete.
ninea@NineA-UbuntuVM:~$
```

7. Wireshark Checking

Because the server and the client are running on the same computer using loopback (127.0.0.1), packets are captured only **once**.

1) Hello messages (Client -> Server, Server -> Client)

```
▸ Frame 1: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ User Datagram Protocol, Src Port: 48016, Dst Port: 8000
▸ Data (14 bytes)

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E·
0010  00 2a f4 c8 40 00 40 11  47 f8 7f 00 00 01 7f 00   ·*··@·@· G·......
0020  00 01 bb 90 1f 40 00 16  fe 29 48 65 6c 6c 6f 20   ·····@·· ·)Hello
0030  73 65 72 76 65 72 2e 00                           server.·
```

```
▸ Frame 2: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ User Datagram Protocol, Src Port: 8000, Dst Port: 48016
▸ Data (14 bytes)

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E·
0010  00 2a f4 c9 40 00 40 11  47 f7 7f 00 00 01 7f 00   ·*··@·@· G·......
0020  00 01 1f 40 bb 90 00 16  fe 29 48 65 6c 6c 6f 20   ···@···· ·)Hello
0030  63 6c 69 65 6e 74 2e 00                           client.·
```

2) List request message (Client -> Server)

```
▸ Frame 3: 61 bytes on wire (488 bits), 61 bytes captured (488 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ User Datagram Protocol, Src Port: 48016, Dst Port: 8000
▸ Data (19 bytes)

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E·
0010  00 2f f4 ca 40 00 40 11  47 f1 7f 00 00 01 7f 00   ·/··@·@· G·......
0020  00 01 bb 90 1f 40 00 1b  fe 2e 52 65 71 75 65 73   ·····@·· ··Reques
0030  74 20 76 69 64 65 6f 20  6c 69 73 74 00            t video  list·
```

3) List response message (Server -> Client)

```
▸ Frame 4: 167 bytes on wire (1336 bits), 167 bytes captured (1336 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ User Datagram Protocol, Src Port: 8000, Dst Port: 48016
▸ Data (125 bytes)

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E·
0010  00 99 f4 cb 40 00 40 11  47 86 7f 00 00 01 7f 00   ····@·@· G·......
0020  00 01 1f 40 bb 90 00 85  fe 98 46 61 6b 65 5f 53   ···@···· ··Fake_S
0030  74 72 65 61 6d 69 6e 67  5f 54 65 73 74 2e 6d 6b   treaming _Test.mk
0040  76 0a 57 57 57 5f 53 74  72 65 61 6d 69 6e 67 5f   v·WWW_St reaming_
0050  54 65 73 74 2e 6d 6b 76  0a 4b 69 61 6e 61 5f 53   Test.mkv ·Kiana_S
0060  74 72 65 61 6d 69 6e 67  5f 54 65 73 74 2e 6d 6b   treaming _Test.mk
0070  76 0a 44 6f 77 6e 70 6f  75 72 5f 53 74 72 65 61   v·Downpo ur_Strea
0080  6d 69 6e 67 5f 54 65 73  74 2e 6d 6b 76 0a 4e 69   ming_Tes t.mkv·Ni
0090  6e 65 5f 53 74 72 65 61  6d 69 6e 67 5f 54 65 73   ne_Strea ming_Tes
00a0  74 2e 6d 6b 76 0a 00                              t.mkv··
```

4) Video stream request message (Client -> Server)

```
▸ Frame 5: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ User Datagram Protocol, Src Port: 48016, Dst Port: 8000
▸ Data (36 bytes)

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E·
0010  00 40 f5 9c 40 00 40 11  47 0e 7f 00 00 01 7f 00   ·@··@·@· G·......
0020  00 01 bb 90 1f 40 00 2c  fe 3f 52 65 71 75 65 73   ·····@·, ·?Reques
0030  74 20 44 6f 77 6e 70 6f  75 72 5f 53 74 72 65 61   t Downpo ur_Strea
0040  6d 69 6e 67 5f 54 65 73  74 2e 6d 6b 76 00         ming_Tes t.mkv·
```

5) Video stream (Server -> Client): Total sent/received packets = 18892 packets