Blake Rusteberg

Programming Assignment #02: RTSP Protocol

CS 447: Networking and Data Communication

Introduction

This project was intended for us to get a better understanding of streaming audio files using UDP over a network. I started this project in Python and learned the basic functionality of sending RTP packets over a network. I started by creating a test UDP server and UDP client to learn how to send WAV files. I found out that reading them in bytes was the easiest way I was going to transfer them over a network. It turns out the first 44 bytes was the header of the WAV file and this determines how the rest of the file is going to sound. I later realized that it wasn't needed because aplay takes care of the encoding of the header for each payload. I eventually got the client to communicate back to the server by sending the entire file in bytes to a new .wav file in the server. The most difficult part of this project was finding the right buffer for sending data over the network with minimal packet loss and some Compared to the last program this one took 3 times as long but there was a lot less error checking and more understanding of the networking side of things.

Objectives

- Establishing a connection between the server, controller, and receiver using socket programming
- Go through each command one by one to test all errors and functionality of each command. Ex. Make sure SETUP worked error free before moving on to PLAY
- Set up a basic UDP client and server to test how I am going to send the data over
- Test to see if the receiver was receiving data
- Make a thread for the receiver and server side of handling the data
- Ran PLAY all the way through and listened to the .wav completely
- Spent 30+ hours determining a buffer size and finding packet loss when sending over UDP
- Fix TEARDOWN
- Multithread the server
- Clean up Code
- Final Error Checking
- Turn in

Design Choices

I stuck to a very simple design choice to make sure my code was going to be understandable to anyone who is reading it. I error checked the entire program so the controller could only enter 1 of 4 commands, SETUP, PLAY, PAUSE, and TEARDOWN. When the controller typed TEARDOWN, they could make a new connection request with the server. I did this by multithreading the server.

Program Run-Through

Controller.py

Figure 1.1: a run-through for each command the server has to offer

Server.py

```
blruste@blruste:~/Documents/CS447_Projects/Project2/Project2_Server$ python3 server.py 30001
127.0.1.1
Port Number 30001 is valid.
Listening for networks.

Client connected...
IP: 127.0.0.1
Port: 54074

C --> S: SETUP rtsp://blruste/default.wav RTSP/2.0
         CSeq: 1
         Transport: UDP;unicast;dest_addr":25001"

C --> S: PLAY rtsp://blruste RTSP/2.0
         CSeq: 2
b'1000.wav\xff\xfd\xff\x05\x00\x04\x00\xfc\xff\xfd\xff\x03\x00\x01\x00\xfe\xff\x01\x00\x02\x00\xfe\xff\xfd\xff\
x02\x00'
b'2000.wav\x00\xff\xff\x00\x00\x02\x00\x00\x00\xfd\xff\x00\x00\x02\x00\x00\x00\xff\xff\x00\x00\x01\x00\x00\x00\
x00\x00'
b'3000.wav\x02/\x01;\x01\xa4\x00\xa6\xff\xc3\xff\xf5\xfet\xff,\xfe\xea\xfe\xec\xfe\xea\xff\xa3\x01\xa7\x01'
b'4000.wav\x06\xb1\x04\xf9\x05\x08\x05\xf2\x063\x08\xb1\x07\x03\t\xa7\x07\x8c\x07t\x06l\x07\x01\x06\r\x08'
b'5000.wav\xfa\xf0\xf8\xee\xfc\x8a\xfc\xf6\xfeG\xfd-\x00O\xfc\xb5\x02h\x00;\x04d\x03-\x03\x17\x01'
b'6000.wav\x03R\xfd(\x02\xa2\xfc\x8f\x02\x1a\xfc\x1e\x04\x88\xfd\xe7\x04\xbf\xff\x85\x04\xf5\xff\xae\x03\xa7\xf
d'
b'7000.wav\x0c\xcb\x0c0\re\rV\x0ea\x0e\xa6\x0ew\x0e6\x0eF\r\x14\x0f\xc0\x0e\xfa\x10\x12\x12'
b'8000.wav\x01\x9a\x05V\x01\xcb\x06\xd2\x00\xa1\x07F\x00u\x07\xe2\xff\xcb\x06\x06\xff\xf6\x05\x8d\xfd\xb7\x04'
b'9000.wav\xf8\xe5\xfc\x90\xf7k\xfbH\xf7p\xf9V\xf7\xd6\xf8\x07\xf8\x8c\xfa\xfd\xf8u\xfc\xab\xf96\xfd'
b'10000.wav\x05\x97\t\x88\x04\x04\x06\xb1\x02\xe9\x01\xe0\x02\x14\x05\x97\x02\xce\x04 \x00A\xff!\xff\x1d\x00'
b'11000.wav\xfb\xff\xf5f\xfb\xe0\xf6\x88\xfb9\xf8\x9a\xfb<\xfa\xbd\xfb\xc9\xfb\x8b\xfbM\xfc\xea\xfa2\xfc'
b'12000.wav\x03\x82\x02\xd5\x02\x14\x02J\x02G\x01\xdf\x01\x8f\x00\xb2\x01\xe3\x000\x02\xd6\x01\xf5\x02\x12\x02'
b'13000.wav\xfc\xdd\xf5\xeb\xfa\x91\xf7\x17\xf4\x88\xf3\xfd\xf3}\xf6\x07\xf9 \xfa\xcc\xf8\x17\xf4A\xf7\xb0\xf2'
b'14000.wav\xfd\xa6\xfb-\xfd\xe1\xfc\xc7\xfeV\xfem\xfer\xfc\x1e\xfdu\xfb\xc4\xfd&\xfd\x93\xfe\xaa\xfd'
b'15000.wav\xe9{\xeb\x8e\xeae\xec\x93\xeb|\xed\xc3\xeb\xd8\xed\x15\xec\x17\xee\x03\xed\x95\xee_\xed\x88\xee'
b"16000.wav\x03^\x04y\x03\xbb\x03\x1e\x04\x88\x04\x9c\x04'\x05\xf4\x04\xdc\x04\xf1\x04Z\x04\x18\x05\x8c\x04"
b'17000.wav\x03<\xff;\x01\x0e\xff\\\x00\x87\xfe1\x01\xde\xfdD\x01\x9d\xfc\xfd\x00S\xfc\x14\x02J\xfe'

C --> S: PAUSE rtsp://blruste RTSP/2.0
         CSeq: 3

C --> S: PLAY rtsp://blruste RTSP/2.0
         CSeq: 4
b'18000.wav\x0b\x80\x0b\x90\x0cR\r \r\x89\ru\r\x15\r\xf4\x0c\xa9\x0c\xff\x0bQ\x0c\xf7\ta\n'
b'19000.wav\x03\xc5\x05\xad\x02\xb7\x05p\x02\x94\x05\x19\x02U\x05\xc6\x01\xf5\x04\x81\x01\x94\x04}\x01w\x04'
b'20000.wav\xfdQ\xf7u\xfc\x06\xf9\xd1\xfc\xa0\xf8q\xfc\x91\xf4\xe4\xfa\xdd\xf2(\xfa\x0b\xf5Y\xfa\x8d\xf5'
b'21000.wav\x03\xdd\x04\xfa\x025\x05^\x01z\x04\x1c\x00\x08\x04s\xff\x1a\x04\x84\xff\x93\x03\x01\x00\xb9\x02'
b'22000.wav\x02\xe5\x03\x06\x02/\x03\x84\x01\xd1\x01B\x01\xcc\x00\x92\x01\xb6\x00^\x02\xd5\x00<\x036\x00'
b'23000.wav\x000\xfd\xac\x00\x86\xfc\xc5\x00\xa0\xfbf\x00\xb7\xfc\x97\xff\xc9\xfdq\xffT\xfb\x1f\x00p\xf9'
```

Figure 2.1: Every time 1000 bytes were sent to the receiver; I would print them to make sure they were being sent. From the figure you can also see the PAUSE and PLAY sequence work gracefully picking up from where it last left off.

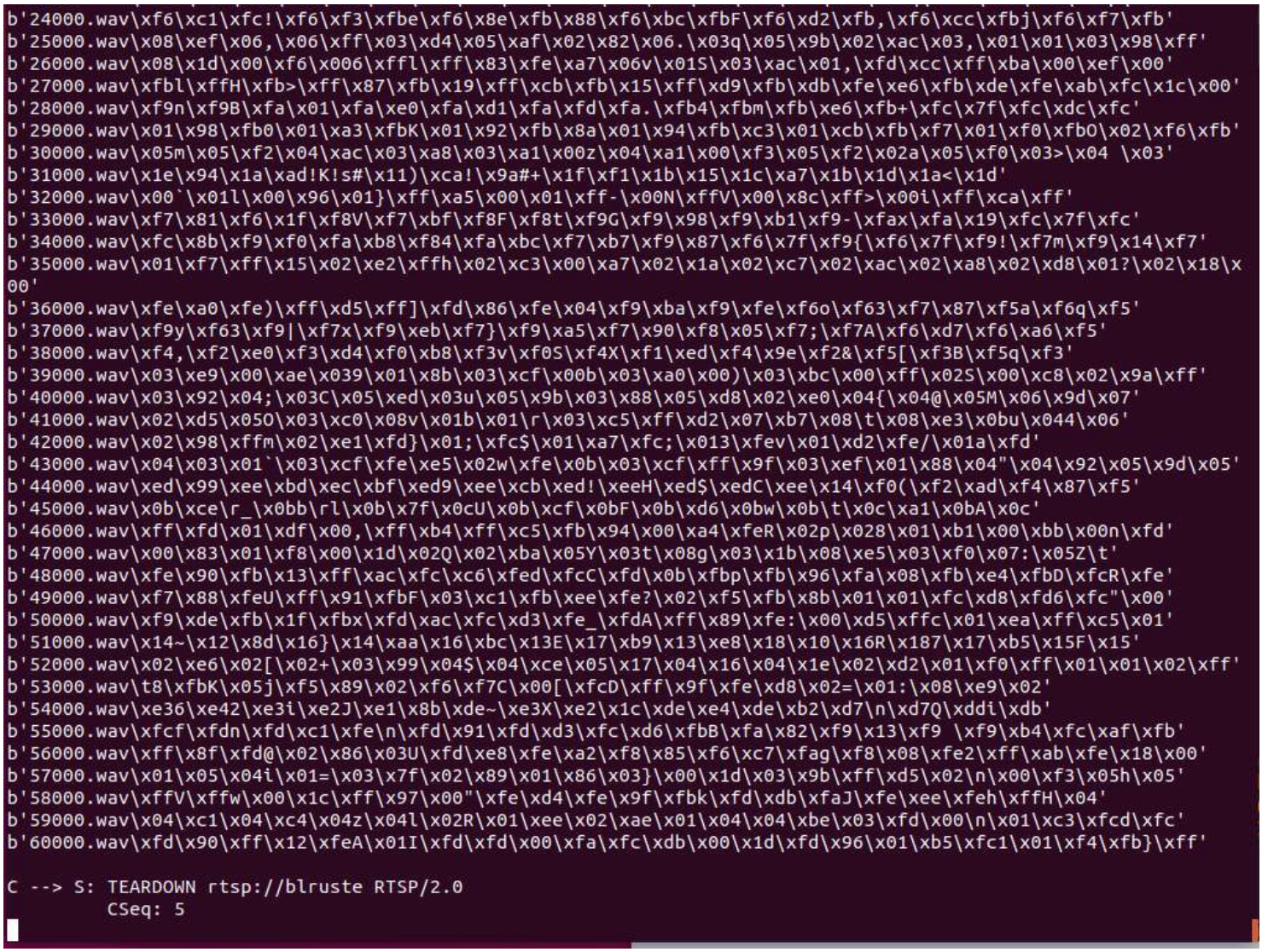

Figure 2.2: This shows some data still being sent and the TEARDOWN request being activated on the server, but the server is still running because of its thread.

Receiver.py

```
blruste@blruste:~/Documents/CS447_Projects/Project2/Project2_Client$ python3 receiver.py 25001
127.0.1.1
The Server is Ready to Receive!!

Message Received from Server: b'1000.wav\xff\xfd\xff\x05\x00\x04\x00\xfc\xff\xfd\xff\x03\x00\x01\x00\xfe\xff\x01\x00
\x02\x00\xfe\xff\xfd\xff\x02\x00'

Message Received from Server: b'2000.wav\x00\xff\xff\x00\x00\x02\x00\x00\x00\xfd\xff\x00\x00\x02\x00\x00\x00\xff\xff
\x00\x00\x01\x00\x00\x00\x00\x00'

Message Received from Server: b'3000.wav\x02/\x01;\x01\xa4\x00\xa6\xff\xc3\xff\xf5\xfet\xff,\xfe\xea\xfe\xec\xfe\xea
\xff\xa3\x01\xa7\x01'

Message Received from Server: b'4000.wav\x06\xb1\x04\xf9\x05\x08\x05\xf2\x063\x08\xb1\x07\x03\t\xa7\x07\x8c\x07t\x06
l\x07\x01\x06\r\x08'

Message Received from Server: b'5000.wav\xfa\xf0\xf8\xee\xfc\x8a\xfc\xf6\xfeG\xfd-\x00O\xfc\xb5\x02h\x00;\x04d\x03-\
x03\x17\x01'

Message Received from Server: b'6000.wav\x03R\xfd(\x02\xa2\xfc\x8f\x02\x1a\xfc\x1e\x04\x88\xfd\xe7\x04\xbf\xff\x85\x
04\xf5\xff\xae\x03\xa7\xfd'

Message Received from Server: b'7000.wav\x0c\xcb\x0c0\re\rV\x0ea\x0e\xa6\x0ew\x0e6\x0eF\r\x14\x0f\xc0\x0e\xfa\x10\x1
2\x12'

Message Received from Server: b'8000.wav\x01\x9a\x05V\x01\xcb\x06\xd2\x00\xa1\x07F\x00u\x07\xe2\xff\xcb\x06\x06\xff\
xf6\x05\x8d\xfd\xb7\x04'

Message Received from Server: b'9000.wav\xf8\xe5\xfc\x90\xf7k\xfbH\xf7p\xf9V\xf7\xd6\xf8\x07\xf8\x8c\xfa\xfd\xf8u\xf
c\xab\xf96\xfd'

Message Received from Server: b'10000.wav\x05\x97\t\x88\x04\x04\x06\xb1\x02\xe9\x01\xe0\x02\x14\x05\x97\x02\xce\x04
\x00A\xff!\xff\x1d\x00'

Message Received from Server: b'11000.wav\xfb\xff\xf5f\xfb\xe0\xf6\x88\xfb9\xf8\x9a\xfb<\xfa\xbd\xfb\xc9\xfb\x8b\xfb
M\xfc\xea\xfa2\xfc'

Message Received from Server: b'12000.wav\x03\x82\x02\xd5\x02\x14\x02J\x02G\x01\xdf\x01\x8f\x00\xb2\x01\xe3\x000\x02
\xd6\x01\xf5\x02\x12\x02'

Message Received from Server: b'13000.wav\xfc\xdd\xf5\xeb\xfa\x91\xf7\x17\xf4\x88\xf3\xfd\xf3}\xf6\x07\xf9 \xfa\xcc\
xf8\x17\xf4A\xf7\xb0\xf2'

Message Received from Server: b'14000.wav\xfd\xa6\xfb-\xfd\xe1\xfc\xc7\xfeV\xfem\xfer\xfc\x1e\xfdu\xfb\xc4\xfd&\xfd\
x93\xfe\xaa\xfd'

Message Received from Server: b'15000.wav\xe9{\xeb\x8e\xeae\xec\x93\xeb|\xed\xc3\xeb\xd8\xed\x15\xec\x17\xee\x03\xed
\x95\xee_\xed\x88\xee'

Message Received from Server: b"16000.wav\x03^\x04y\x03\xbb\x03\x1e\x04\x88\x04\x9c\x04'\x05\xf4\x04\xdc\x04\xf1\x04
Z\x04\x18\x05\x8c\x04"

Message Received from Server: b'17000.wav\x03<\xff;\x01\x0e\xff\\\x00\x87\xfe1\x01\xde\xfdD\x01\x9d\xfc\xfd\x00S\xfc
\x14\x02J\xfe'

Message Received from Server: b'18000.wav\x0b\x80\x0b\x90\x0cR\r \r\x89\ru\r\x15\r\xf4\x0c\xa9\x0c\xff\x0bQ\x0c\xf7\
ta\n'
Playing WAVE 'new.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
```

Figure 3.1: This shows the receiver.py receiving the data and playing the first chunk after packet 18000.

```
Message Received from Server: b'53000.wav\t8\xfbK\x05j\xf5\x89\x02\xf6\xf7C\x00[\xfcD\xff\x9f\xfe\xd8\x02=\x01:\x08\
xe9\x02'

Message Received from Server: b'54000.wav\xe36\xe42\xe3i\xe2J\xe1\x8b\xde~\xe3X\xe2\x1c\xde\xe4\xde\xb2\xd7\n\xd7Q\x
ddi\xdb'

Message Received from Server: b'55000.wav\xfcf\xfdn\xfd\xc1\xfe\n\xfd\x91\xfd\xd3\xfc\xd6\xfbB\xfa\x82\xf9\x13\xf9 \
xf9\xb4\xfc\xaf\xfb'

Message Received from Server: b'56000.wav\xff\x8f\xfd@\x02\x86\x03U\xfd\xe8\xfe\xa2\xf8\x85\xf6\xc7\xfag\xf8\x08\xfe
2\xff\xab\xfe\x18\x00'

Message Received from Server: b'57000.wav\x01\x05\x04i\x01=\x03\x7f\x02\x89\x01\x86\x03}\x00\x1d\x03\x9b\xff\xd5\x02
\n\x00\xf3\x05h\x05'

Message Received from Server: b'58000.wav\xffV\xffw\x00\x1c\xff\x97\x00"\xfe\xd4\xfe\x9f\xfbk\xfd\xdb\xfaJ\xfe\xee\x
feh\xffH\x04'

Message Received from Server: b'59000.wav\x04\xc1\x04\xc4\x04z\x04l\x02R\x01\xee\x02\xae\x01\x04\x04\xbe\x03\xfd\x00
\n\x01\xc3\xfcd\xfc'

Message Received from Server: b'60000.wav\xfd\x90\xff\x12\xfeA\x01I\xfd\xfd\x00\xfa\xfc\xdb\x00\x1d\xfd\x96\x01\xb5\
xfc1\x01\xf4\xfb}\xff'
Receiver is exiting gracefully!
return_value: 0
blruste@blruste:~/Documents/CS447_Projects/Project2/Project2_Client$
```

Figure 3.2: This shows that the server has ran the TEARDOWN request and shows the receiver.py exiting the program gracefully!

Wireshark

127	09:43:33.859196	192.168.56.1	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
128	09:43:34.859748	192.168.56.1	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
129	09:44:10.451411	fe80::4c90:8f28:d82d:c124	ff02::c	UDP	718	56915 → 3702 Len=656
130	09:44:10.451766	192.168.56.1	239.255.255.250	UDP	698	56914 → 3702 Len=656
131	09:44:10.572634	fe80::4c90:8f28:d82d:c124	ff02::c	UDP	718	56915 → 3702 Len=656
132	09:44:10.642978	192.168.56.1	239.255.255.250	UDP	698	56914 → 3702 Len=656
133	09:44:10.814861	fe80::4c90:8f28:d82d:c124	ff02::c	UDP	718	56915 → 3702 Len=656
134	09:44:11.025696	192.168.56.1	239.255.255.250	UDP	698	56914 → 3702 Len=656
135	09:44:11.298903	fe80::4c90:8f28:d82d:c124	ff02::c	UDP	718	56915 → 3702 Len=656
136	09:44:11.789969	192.168.56.1	239.255.255.250	UDP	698	56914 → 3702 Len=656
137	09:44:12.267718	fe80::4c90:8f28:d82d:c124	ff02::c	UDP	718	56915 → 3702 Len=656
138	09:44:13.318935	192.168.56.1	239.255.255.250	UDP	698	56914 → 3702 Len=656
139	09:44:14.204439	fe80::4c90:8f28:d82d:c124	ff02::c	UDP	718	56915 → 3702 Len=656
140	09:44:15.319808	192.168.56.1	239.255.255.250	UDP	698	56914 → 3702 Len=656
141	09:44:16.205153	fe80::4c90:8f28:d82d:c124	ff02::c	UDP	718	56915 → 3702 Len=656
142	09:44:17.320095	192.168.56.1	239.255.255.250	UDP	698	56914 → 3702 Len=656

Figure 4.1: These are some UDP packets that are being sent over the network to the receiver

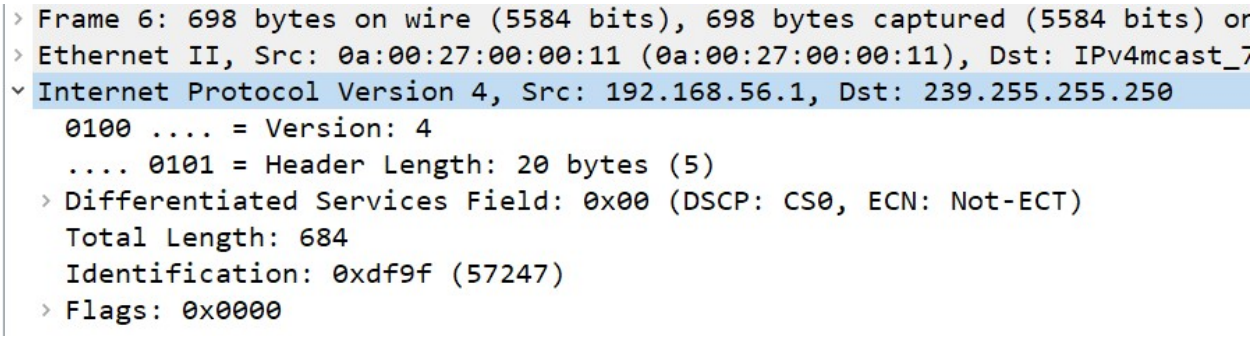

Figure 4.2: This shows the header length of each packet it 5 bytes.

Conclusion:

I learned a lot in this project and have found a newfound respect for people who are good at socket programming, because for me it was quite difficult. This project itself took about 70 hours to complete and for only 2 weeks to complete it that is most of the week. I enjoyed the ups and downs of this project, but I wish there was more time to complete it. The most difficult part for me in this project was figuring out the buffer size of sending my RTP packets. Once I figured it out there was another hurdle of pausing the audio and then playing it back again. It came back to what we did in our operating systems class, and I basically used a semaphore stopping, waiting and starting threads, although this was much more difficult. All an all this project took a lot of time out of my week, but it just goes to show how dedicated and interested I am in researching programming and networking. Not a lot of people will spend 70 hours in any given week to complete a project.