Ryan Bauert
Ben Hunt
Austin Kelly
Tony Ho

# Final report for CPSC3780 Project

Our project is a simple UDP protocol that utilizes networking features to connect a client to a server through a proxy. The proxy will be used to simulate an actual network, which has the probability to delay or drop frames/packets. We decided to use a 85/10/5 probability ratio for our project for successful, delayed, and dropped frames respectfully.

Our sliding window protocol uses a size of 3. This means that 3 frames are sent at a time and the acknowledgment for all 3 must be returned back successfully within the time window for the next set of frames to be sent. A cumulative acknowledgement is used, meaning one acknowledgement is sent back to signify all 3 frames have been successfully received, rather 3 separate ones. This is advantageous due to there being less traffic having to be sent, but not advantageous if a frame is dropped as they will all have to be re-sent. We have approximately an 85% success rate, so it is reasonable that only one acknowledgment would be sent back. Increasing the sliding window value anymore than 3 though, a cumulative protocol would be a poor choice due to the increasing probability that frames will be dropped. We would adopt an independant acknowledgment protocol if this was the case.

See Appendix A for a visualization of the probability that all frames will be sent successfully, and see Appendix B for a spreadsheet of the probability. There are a total of 27 cases that may occur. There are 3 frames being sent, each with 3 possible outcomes, for a total of 3 * 3 * 3 = 27.

The probability of success for delayed frames is conditional that the delay is less than the timeout value. If the random delay time is low, the frame will succeed within the timeout window. However, if it is long, it may fail and the process will need to restart. This was not taken into account for the calculations for simplicity, but it is being noted here for completeness.

Ryan Bauert
Ben Hunt
Austin Kelly
Tony Ho

# The Protocol

Our protocol is described as follows:

A three-step handshake will occur to verify a connection has been established. After the connection is verified, the server will begin to send data packets to the client. Packets will consist of a sequence of random numbers between 0 and 1000. The server will send these numbers in groups of 3, and they must all be acknowledged successfully before the next set is sent.

Our program has various error handling procedures, such as timeouts for non-response, non-connection etc. and theoretically will not crash.

The server will randomly generate a number between 0 and 99. Based on the number, it will drop the frame, delay the frame, or send it successfully. If there are delays, the delays must not be longer than the timeout value of 5 seconds. The server will randomly generate a delay between 1 and 5 seconds, so if more than 2 frames are delayed, it's possible the process will timeout, and have to restart.

# Server:

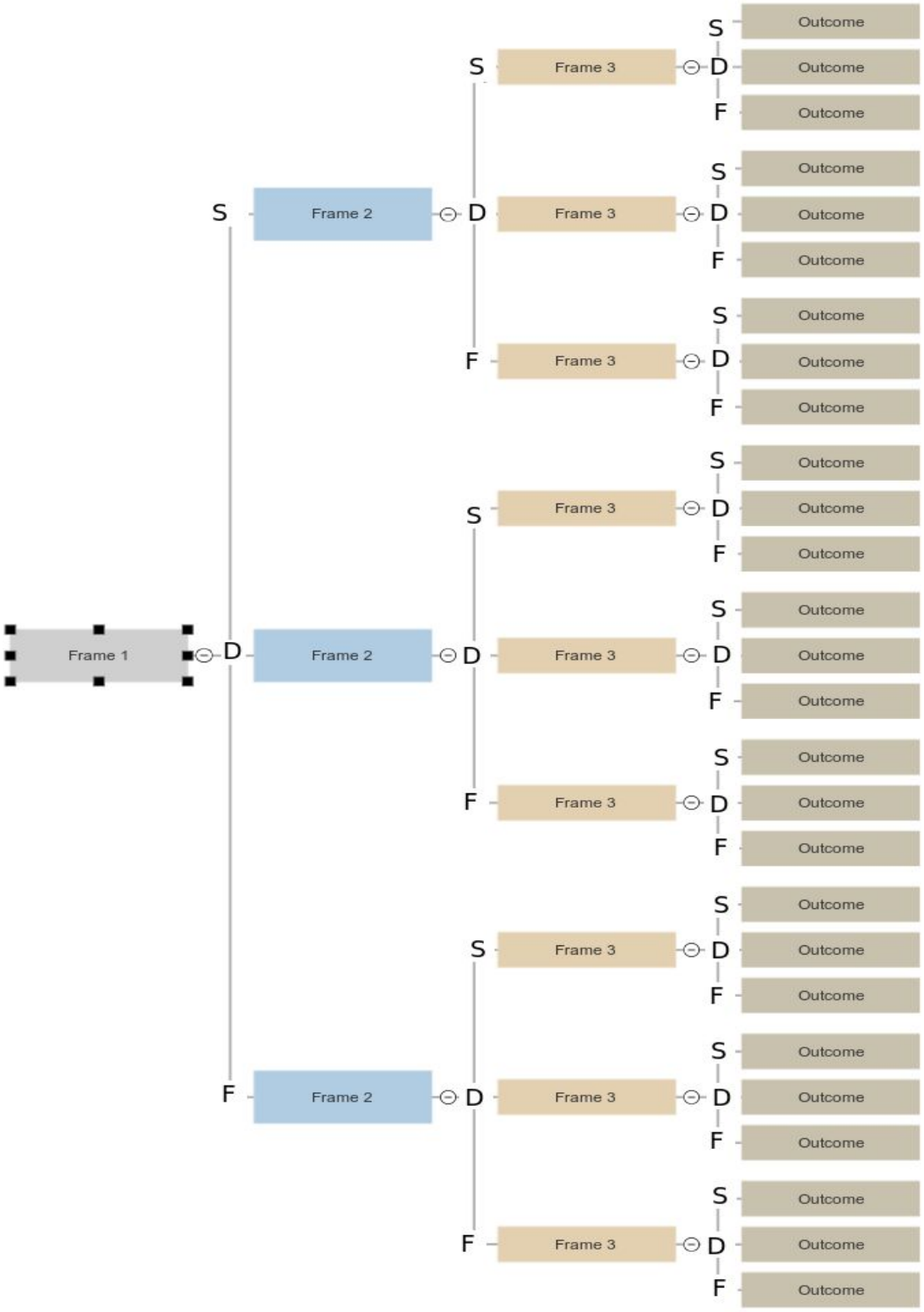
We found after testing the connection a multitude of times, that the success rate for sending frames was as expected in the spreadsheet. See the attached log in Appendix C for further information and details on the actual transmissions. See the output files in the repo as well to see that the outputs in the console match the outputs in the file.

# Conclusion:

By creating this program, we have demonstrated how to use UDP to create a sliding window protocol to send data from a server to a client. By creating a dropping algorithm, we simulated an actual network by randomly dropping packets or delaying them. j
This project has served as a great way for our group members to collaborate on a hands-on actual experience of utilizing networks and learn Python in the process.

Ryan Bauert
Ben Hunt
Austin Kelly
Tony Ho

# Appendix A



Ryan Bauert
Ben Hunt
Austin Kelly
Tony Ho

# Appendix B

| | Frame 1 | Frame 2 | Frame 3 | Outcome | Successful frames |
|---|---|---|---|---|---|
| Case 1 | 0.85 | 0.85 | 0.85 | 0.614125 | 0.614125 |
| Case 2 | 0.85 | 0.85 | 0.1 | 0.07225 | 0.07225 |
| Case 3 | 0.85 | 0.85 | 0.05 | 0.036125 | |
| Case 4 | 0.85 | 0.1 | 0.85 | 0.07225 | 0.07225 |
| Case 5 | 0.85 | 0.1 | 0.1 | 0.0085 | 0.0085 |
| Case 6 | 0.85 | 0.1 | 0.05 | 0.00425 | |
| Case 7 | 0.85 | 0.05 | 0.85 | 0.036125 | |
| Case 8 | 0.85 | 0.05 | 0.1 | 0.00425 | |
| Case 9 | 0.85 | 0.05 | 0.05 | 0.002125 | |
| Case 10 | 0.1 | 0.85 | 0.85 | 0.07225 | 0.07225 |
| Case 11 | 0.1 | 0.85 | 0.1 | 0.0085 | 0.0085 |
| Case 12 | 0.1 | 0.85 | 0.05 | 0.00425 | |
| Case 13 | 0.1 | 0.1 | 0.85 | 0.0085 | 0.0085 |
| Case 14 | 0.1 | 0.1 | 0.1 | 0.001 | 0.001 |
| Case 15 | 0.1 | 0.1 | 0.05 | 0.0005 | |
| Case 16 | 0.1 | 0.05 | 0.85 | 0.00425 | |
| Case 17 | 0.1 | 0.05 | 0.1 | 0.0005 | |
| Case 18 | 0.1 | 0.05 | 0.05 | 0.00025 | |
| Case 19 | 0.05 | 0.85 | 0.85 | 0.036125 | |
| Case 20 | 0.05 | 0.85 | 0.1 | 0.00425 | |
| Case 21 | 0.05 | 0.85 | 0.05 | 0.002125 | |
| Case 22 | 0.05 | 0.1 | 0.85 | 0.00425 | |
| Case 23 | 0.05 | 0.1 | 0.1 | 0.0005 | |
| Case 24 | 0.05 | 0.1 | 0.05 | 0.00025 | |
| Case 25 | 0.05 | 0.05 | 0.85 | 0.002125 | |
| Case 26 | 0.05 | 0.05 | 0.1 | 0.00025 | |
| Case 27 | 0.05 | 0.05 | 0.05 | 0.000125 | |
| | | | | | |

Ryan Bauert
Ben Hunt
Austin Kelly
Tony Ho

| | | | | | |
|---|---|---|---|---|---|
| Total success | | | | | 0.857375 |

# Appendix C

***Server console ouput***

Awaiting connection...
Received b'10' from ('142.66.140.39', 41013)
Sending b'10' to ('142.66.140.39', 41013)
Connection Established
successful frame 0
successful frame 1
successful frame 2
received ack:  b'ack'
successful frame 3
successful frame 4
delayed frame 5   2.1
received ack:  b'ack'
successful frame 6
delayed frame 7   3.3
successful frame 8
received ack:  b'ack'
successful frame 9
successful frame 10
successful frame 11
received ack:  b'ack'
b'terminated'
Received close acknowledgement from ('142.66.140.39', 41013)
Connection Closed

Awaiting connection...
Received b'10' from ('142.66.140.39', 57743)
Sending b'10' to ('142.66.140.39', 57743)
Connection Established
successful frame 0
successful frame 1
successful frame 2
received ack:  b'ack'
successful frame 3

Ryan Bauert
Ben Hunt
Austin Kelly
Tony Ho
successful frame 4
successful frame 5
received ack:  b'ack'
successful frame 6
successful frame 7
successful frame 8
received ack:  b'ack'
successful frame 9
successful frame 10
successful frame 11
received ack:  b'ack'
b'terminated'
Received close acknowledgement from ('142.66.140.39', 57743)
Connection Closed

Awaiting connection...
Received b'10' from ('142.66.140.39', 55765)
Sending b'10' to ('142.66.140.39', 55765)
Connection Established
successful frame 0
successful frame 1
successful frame 2
received ack:  b'ack'
successful frame 3
successful frame 4
successful frame 5
received ack:  b'ack'
successful frame 6
successful frame 7
successful frame 8
received ack:  b'ack'
delayed frame 9   3.9
successful frame 10
successful frame 11
received ack:  b'ack'
b'terminated'
Received close acknowledgement from ('142.66.140.39', 55765)

Ryan Bauert
Ben Hunt
Austin Kelly
Tony Ho
Connection Closed

***Client console output***

```
~/Desktop/NetworkingSocketDemo$ python3 client.py output
Enter the IP address of the server to connect to: 142.66.140.38
sending 2 bytes to ('142.66.140.38', 8000)
data received:  b'10'
sending 3 to ('142.66.140.38', 8000)
connection established
Received data:  b'520'
Received data:  b'984'
Received data:  b'430'
Received data:  b'729'
Received data:  b'39'
Received data:  b'682'
Received data:  b'957'
Received data:  b'869'
Received data:  b'450'
Received data:  b'424'
Received data:  b'395'
Received data:  b'893'
b'close'
terminated
Connection gracefully terminated

~/Desktop/NetworkingSocketDemo$ python3 client.py output1
Enter the IP address of the server to connect to: 142.66.140.38
sending 2 bytes to ('142.66.140.38', 8000)
data received:  b'10'
sending 3 to ('142.66.140.38', 8000)
connection established
Received data:  b'130'
Received data:  b'403'
Received data:  b'430'
```

Ryan Bauert
Ben Hunt
Austin Kelly
Tony Ho
Received data:  b'26'
Received data:  b'109'
Received data:  b'956'
Received data:  b'876'
Received data:  b'295'
Received data:  b'229'
Received data:  b'771'
Received data:  b'751'
Received data:  b'25'
b'close'
terminated
Connection gracefully terminated
~/Desktop/NetworkingSocketDemo$ python3 client.py output2
Enter the IP address of the server to connect to: 142.66.140.38
sending 2 bytes to ('142.66.140.38', 8000)
data received:  b'10'
sending 3 to ('142.66.140.38', 8000)
connection established
Received data:  b'614'
Received data:  b'162'
Received data:  b'263'
Received data:  b'470'
Received data:  b'61'
Received data:  b'453'
Received data:  b'329'
Received data:  b'639'
Received data:  b'149'
Received data:  b'194'
Received data:  b'639'
Received data:  b'976'
b'close'
terminated
Connection gracefully terminated