CS 118 Project 2 Report

Fall 2016

Brian Li (504442822)

Daniel O’Laughlin (304467494)

Jack Storch (304483390)

High Level Design of Server/Client:

We placed the TCP Message (header + payload) implementation in a separate file called helper.cpp and helper.h. In these files, we also put helper functions, such as getIP() which takes in a port number and returns an IP address.

In both server and client, we first store argv[1] and argv[2] into local variables. We create a UDP socket and bind to it.

Ran out of time to finish the project. Works for all file sizes with no packet loss. Only works for small files when packet loss is introduced. The error came into play because of cumulative acks and there annoyance. It was very close to being done but the ack numbers between the two caused a lot of error because it was hard for one to know what packets were lost in order to send the correct packet ack and sequence number. Code became to convoluted to solve at a certain point but I know I could have gotten It If I had just a bit more time. I am determined to get this to work.

Problems/Difficulties:

One of the first difficulties we faced was making sure that we sent the correct TCP message depending on which (A,S,F) flag was set. This was confusing to debug. To overcome this problem, we spent a lot of time discussing the correctness conceptually and making sure that it matched up in the code.

We also ran into problems with the last packet of the file. The packet size was fine (less than 1024 as expected) but would there would be 9 extra random bytes in the client’s output file. After many hours of trying to find this bug, we finally found that the error was due to the memcpy function. Essentially, we never cleared the buffer after it became full, so if and when the last packet only needed less than 1024 bytes, memcpy would copy over extraneous leftover bytes.

Additional Instructions:

No additional instructions.

Testing Our Project:

We use the command ‘wget’ to obtain html files from the internet. We use these files as the test files that we put into the command line as the argument for ./server. We then tested our client and server using an unused port number. After the client receives all the bytes sent from server, it puts the data into an output file. We diff the output file with the original html file to check that the two files are equivalent.

We obviously also tested for dealing with packet loss using the given network emulation.

Team Member Contributions:

Instead of splitting up the work as we did the last project, we all worked on the code simultaneously, with one person as the scribe (doing the actual typing). We found that this method of collaboration reduced bugs (don’t have three separate sets of code merging into one) without slowing down the process too much. John did the typing, but the contribution of every team member was equivalent.