Ameer Khan
ID: 1476214
2/19/19

CMPE 156: Lab 4 Documentation

In order to compile the program, first go to src folder which has the makefile, client.c and server.c. First run make clean, then run make which will create an executable called client and server in the bin folder. In order to have files transferred in the dest folder, go to this folder and then run the files. For example, after making the file in src. Then go to the dest folder and run ../bin/client <server-info.txt> <numchunks> <filename> and run ../bin/server <portnumber>. The root folder also has the server-info.txt file which has the ip addresses and port numbers. There is the also the files folder which has six text files.

Now I will explain the protocol. First the server side, I initialize the server, the server connects, the server receives file name requested, the server sends file size, then receives the thread information, then sends the file content back to the server. On the client side, I initialize the client, the client reads the server information, then request the size of the file, then creates a number of threads which depends on the number of connections, then the threads connect to the server, then the threads receive the file contents. The client sends a filename to request the file size, the server sends back the file size and then connection ends. The client also sends a filename, offset and chunk size to request the file contents, the server sends back the file contents and then the connection ends after the client disconnects. The server knows whether to get the size or contents based on a flag I set. I replaced all reads and write with recvfrom and sendto function in order to do UDP file transfer.

Now I will explain the message formats. On the server side, the user has to insert one argument which is the port number. On the client side, the user has to insert three arguments.

The first argument is the server-info.txt file which has the ip and port number, the second argument is the number of chunks/connections the user wants, and the third argument is the file the client wants to be transferred from the files folder. Once the arguments have been entered on the client side and server side, I have a few print statements that help me know when the file size is requested and when the file contents is requested. I also have a print statement on the server side that shows how many bytes the file is and how many bytes each thread/server sends. I also have error handling messages printed to stderr.

Now I will talk about error handling done in the client and server programs. The error checking done on the client side is to check the correct number of arguments input by the user, error checking for opening the socket, error checking for checking if the ip address is valid, error checking for connecting to the server, error checking for writing/reading from the server and error checking for opening a file. The error checking done on the server side is to check the correct number of arguments input by the user, error checking for opening the socket, error checking for binding the socket to the port number, error checking to see if connection is valid with the client, error checking for reading in the socket from the client and error checking for reading in the file.