Alexander Long, Michael Nelson

CS 4390.002

Due: 29 April 2017

Project Documentation

The programming of this project was split into two primary components: TCP-oriented, handled by Mr. Long, and UDP-oriented, handled by Mr. Nelson. Each member was responsible for the creation and implementation of a sender and receiver utilizing sockets and UDP datagrams where needed. The sender for each was compiled into one user-friendly class by Mr. Nelson, while the receivers for the UDP and TCP remain in separate classes.

TCP is designed to guarantee the delivery of packets as well as ensure they are delivered in the order they were sent. It uses error detection to ensure that any problems with packets result in retransmission, thus allowing for guaranteed transmission. TCP also allows for both flow and congestion control. TCP regulates flow in an end-to-end manner. It regulates the rate at which the sender sends data, to avoid overwhelming the receiver. It uses sliding-window to ensure it only sends a certain amount of data before the receiver acknowledges successful data delivery. It prevents a deadlock in flow with the inclusion of a persist timer, which, when run out, results in the sender attempting to recover from deadlock by sending a small packet in an attempt to get a new window size from the receiver. TCP uses varying retransmit times based on the time between packet send/acknowledgement times (round trip time estimates). There are various algorithms used in the implementation of TCP congestion control using this information to avoid congested networks or relieve congested receivers.

The initial data to be sent via TCP is divided into packets of bytes, which are encapsulated into an IP datagram and sent through the media to be reassembled at the destination. Each packet is a specific size, decided when the connection between the sender and receiver is first made. The smaller the packet size, the less likely there is for an error to occur, but the more packets must be sent. It first waits for a request from a TCP and port, after which a connection between the receiver and sender will be initialized. Then, after each of the packets are sent to the receiver, the connection will close. As opposed to UDP, the delivery of packets is in the order sent and the delivery is guaranteed. As such TCP is not as fast as its UDP counterpart, but has the benefit of error-free transmission. The fact that the packet size is agreed upon when the connection between the sender and receiver is first made helps to minimize excessive packet retransmission, further mitigating the effects of slower transmission speed.

The class Receiver\_TCP.java and the TCP components of Sender.java perform the same basic functions as TCP. Sender splits the specified file into packets, where the number of bytes is defaulted to 1kb. The file is put into a buffer, which is then segmented into said 1kb packets and written into the receiver in order as each packet is sent. The receiver has a similar setup, preparing to receive each of these 1kb packets as they come and outputting a message confirmation for each packet that is received as well as which packet number it is and which sequence of bytes it contains. After sending a message that the file transfer was completed, the sockets are closed, the buffers are cleared, and the program ends. If the user desires to run the protocol based on TCP, the Sender must be run first, and then the TCP\_Reciever after CONNECT\_TCP has been selected. This must be done again if the user wishes to send another file.