Joshua Eldridge

Cameron Sprowls

CIS 457

10/15/2017

UDP Reliable File Transfer Project Documentation

Overview

This project uses Java to transfer files over the UDP protocol and implements checksum for reliability and sliding window for efficiency.

Client-Side Decisions

The client starts out by prompting the user for a client port number, server port number, and an IP address. If any of these are invalid values then it will re-prompt the user until all of the values are valid. It then sets up the client’s socket with a timeout of 5000 milliseconds, this higher timeout allows for any slow packets to get there. It then tries to connect to the server and get a list of available files, it will try to get a valid file list packet with 10 attempts, if it doesn’t receive a valid packet after that many attempts then we just determined that it is done wasting its time trying to get the list information from the server. If it does get a valid file list packet from the server we loop through and build an available files string from the characters in the packet data and present them to the user. We then tell the user to select a file out of the list and we send a request packet to request that selected file from the server and like before, we try 10 times to get a valid response back. The server will respond first with the number of packets we expect to receive and the size of the file. Finally, we start reading packets using a sliding window of 5 and keep reading packets until we have received the expected number of packets.

Server-Side Decisions

The server is similar to the client in the order of actions it takes. It follows the same kind of initial value prompting and sets up its server socket with a timeout value of 4000 milliseconds. It gets the client file list request and then calculates all of the available files and sends it back to the client. Once it receives another request for the actual file the client wants, the server will see if that file indeed exists and calculates the number of packets that it will send and sends this information to the client with a status packet. The server follows the sliding window pattern and will send a window of 5 packets to the client and keep doing so until it receives an acknowledgement and then sets the beginning of the sliding window to that packet sequence number because the client wouldn’t send an acknowledgement for this lastest packet if it didn’t receive the earlier packets. It also makes sure not to read the entire file into a single byte array, but instead uses the RandomAccessFile class to read from various parts of the file and stores that into a byte array the same size as the sliding window packets so that it only uses what packets are currently waiting to be acknowledged.

Utility Methods and Classes

There were two utility classes that we used to help in the file transfer process, one is the ChecksumException class which extends the standard Exception. This class just has some small override that tells you what it expected as a checksum and what the checksum actually was if they were different. The other utility class is the Header class, this is the powerhouse of the file transfer process and helps with the reliability aspect. It puts all of the flags into the correct places of the header in the packet to be sent and calculates the checksum of the packet before sending and put it into the packet for the receiving end to calculate again. One of the last important methods it uses is the intToByteArray which will convert an integer into a given number of bytes with bit shifting.