CS 3251 – Programming Assignment 2

Abhishek Shroff

November 30, 2010

1 Submission

This tar.gz file contains 4 directories – *bin*, *doc*, *src*, and *test*.

The *doc* directory contains this readme, as well as the output file that shows the replies received from the TCP and UDP servers.

The *bin* directory contains binary files created as a result of executing 'make' in the *src* directory.

The *src* directory contains all the source files that are required for this project including a Makefile that contains a target to build the binaries and place them in the appropriate folder.

The *test* directory contains test files of various sizes ranging from 100kb - 100mb, and some test scripts that record values after running the required transfer tests.

2 Protocol

The protocol follows an extremely simple design with minimal header information. Most of this information is encoded into the first byte, called the status byte. All information sent from the server end contains a 4 byte header. The first byte is the status byte, and the next three bytes represent the length of the packet. Following this are $length - 4$ bytes of data. If the length is 4, it simply means there is no other data to send. Apart from the initial transaction request, each packet sent by the client is a single status byte that encodes everything. The format of the initial request is specified below.

A transaction is started by a client by sending a packet whose status byte is 0, followed by 3 bytes of maximum packet length, least significant byte first. Following these 4 bytes are the characters of the file name up until the end of the packet. Note that since the maximum packet size has to be sent in 3 bytes, a packet size cannot have an unsigned representation greater than 24 bits. This means that the maximum supported packet size is 4MB.

On the server end, it waits for a connection from a client following this format, *i.e.* the status byte is 0. In this case, the server knows to parse a length and filename. If the server is unable to find the requested file, then it replies with a status byte of 3 without any data, and waits for a response. If the client receives a status byte of 3, replies with a status of 3 and terminates without waiting for a response. Note that the server may not receive this packet; this case will be dealt with.

In the normal case, the server replies with an alternating bit as the status byte, followed by the the next chunk of data. This data is of length max length - 4 bytes, to accoutn for the 4 byte header. When the client receives this, it replies with a 1 or'd with the alternating bit left shifted by 7, in order to increase efficiency and be able to pack the data into one byte. If the server receives a packet with a bit that does not match, it resends the old packet assuming that this packet never got sent. Once it receives a matching bit from the client, it then proceedes to send the next file chunk with the alternating bit flipped.

Once the server has nothing more to send in that transaction, it sends a packet with no data and the status byte as 2, which indicates the end of the file. When the client receives this packet, it terminates after sending a header with the status byte 2. Again, there is a chance that this packet is not received by the server. This case is dealt below.

It is very important to note that the server cannot handle multiple clients at the same time, because it was specified that it need not do that. It can handle them sequentially, but simultaneously will not work. If a server receives a packet with a status byte of 0, it closes the old connection and initializes a new connection with the new client. This design fits in with the fact that the client does not wait for a response upon termination. If a connection is terminated with either code 2 (success) or code 3 (file not found), or any other possible extension, and the response from the client to the server is dropped, the server simply keeps trying to send the packet until the next new connection when it receives a status byte of 0.

3 Limitations

The servers are fairly robust, and will not terminate on any input from the clients. That said, there are some limitations, which, if surpassed, will break this program.

1. Theoretically, the server supports a maximum packet size of 2MB because this is the largest signed integer that can be represented with 24 bits. However, the system might not be able to allocate that much memory, in which case it will try to reduce the packet size.