CS118 Project 2 Report

Presented by:

Jack Gong 005025415

Jason Mao 205027376

June 5th 2018

TA: Muhammad Taqi Raza

Department of Computer Science

UCLA

Project description

This project involves the implementation of both client and server. In between, the client and the server will communicate with a window based TCP protocol. This protocol is established upon UDP socket.

The server listens to any incoming http request and attempts to establish a three way handshake with the incoming client first. Once the hand shakes are successful, the client will send a request (piggy back to the third handshake) to the server, requesting a file described from the command line input. Afterwards, the server will reply with the corresponding file, or data segments, if the file is bigger then the maximum packet size limit.

Server and Client design

The server was designed to create a socket, then listen to any incoming client's socket, bind to it, then begin file transfer. I was similar to project 1, except the protocol we're using now is UDP(SOCK_DGRAM). Once the three way handshake process is complete, the client will send a file request, and the server will look for the file and send corresponding request. Once the file transfer is complete, the server will send a FIN to notify the client, and thus begin the four way handshake process. Finally, the client socket will close, while the server remains open, resetting all the default configuration and listen to other incoming sockets.

Implementation Description

- ### Header format

  The header is stored in a character array and is comprised of the following:

  [source IP, source port number, destination IP, destination number, type, sequence number, ack number, body]

- Packet format

The struct packet contains all the information.Its member variable contains all parameters inside the header above, as well as retransmission timeout for each packet. It's member functions include parsing the receiving buffer into packet object, setting the time out, constructing a packet with correct parameters, etc.

- Message Queue and Window

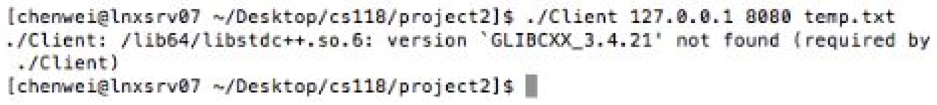
When the file is greater than the window size, the file is separated into packet segment and will be pushed into a message queue in order. If the window still has available size, then a new packet from the message queue will be popped into the window. The window is a list, each time the server gets an "ACK", the window will remove the corresponding packet and load more packets from the message queue. If both window and queue are empty, then a "FIN" will be sent to the client, indicating the file transfer is complete.

Obstacles Encountered

Numerous obstacles were encountered in this project unfortunately. We set the retransmission time out as a const int value = 0.5, without noticing it has to be a double type. We spent a long time debugging and wondering why the client is spamming retransmission dispute there's a 0.5 second delay.

Jack spent a long time understanding the packet struct and its parameter definition, which was designed by Jason for the server. Specifically the sequence number, ack number, expected sequence number, sequence progress and ack progress, how the entire logic flows. The pair programming can sometimes be difficult. The client was implemented with the same logic thereafter.

The Linux server provided by the SEASNET does not support our client and server.



```
[chenwei@lnxsrv07 ~/Desktop/cs118/project2]$ ./Client 127.0.0.1 8080 temp.txt
./Client: /lib64/libstdc++.so.6: version `GLIBCXX_3.4.21' not found (required by
 ./Client)
[chenwei@lnxsrv07 ~/Desktop/cs118/project2]$
```

We then decided to use virtual machine to run our project.