**[CS341] Introduction to Computer Networks**

**Project 1. Socket Programming**

**20170504 Juan Lee**

**Introduction**

This report is about the first programming assignment of CS341 Introduction to Computer Networks. The first assignment is to implement basic client and server, also server with select if available, with functions of encryption and decryption. Some details about this assignment is described on the instruction file.

The report is talking about the structure of my code, self-test results for my client and servers, and the way to compile them. For some of important codes, I copied it from my source code to explain some details.

**About Packet (Packet.c / Packet.h)**

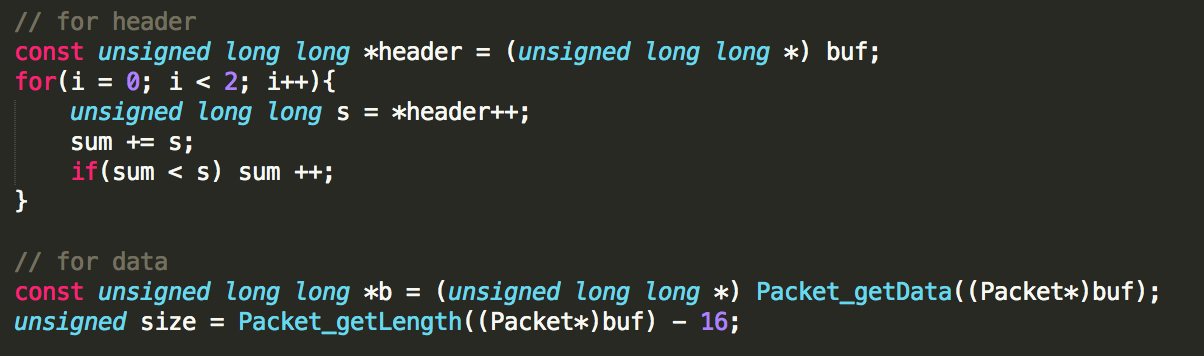
Before I start to explain the structure of client and servers, let me explain the packet module that I made. This assignment has a specific format of packet, and functions for that, so I implemented those functions before writing the main part of the codes. In addition, this header and source have some pre-defined helper functions for socket programming.

Packet Structure is defined like below:



It has five fields, 2-byte op, 2-byte checksum, 4-byte keyword, 8-byte length, and data. I don’t know the length of data before getting the input, so I defined it as a pointer. The data in packet and packet itself will be dynamically allocated by Packet\_create function below. Those Packet\_ functions automatically changed the endian also, so that Op and Length function changed the endian for the data.

For the checksum, I referenced site on KLMS. However, I changed some since I have different structure of packet. Here is where I changed:



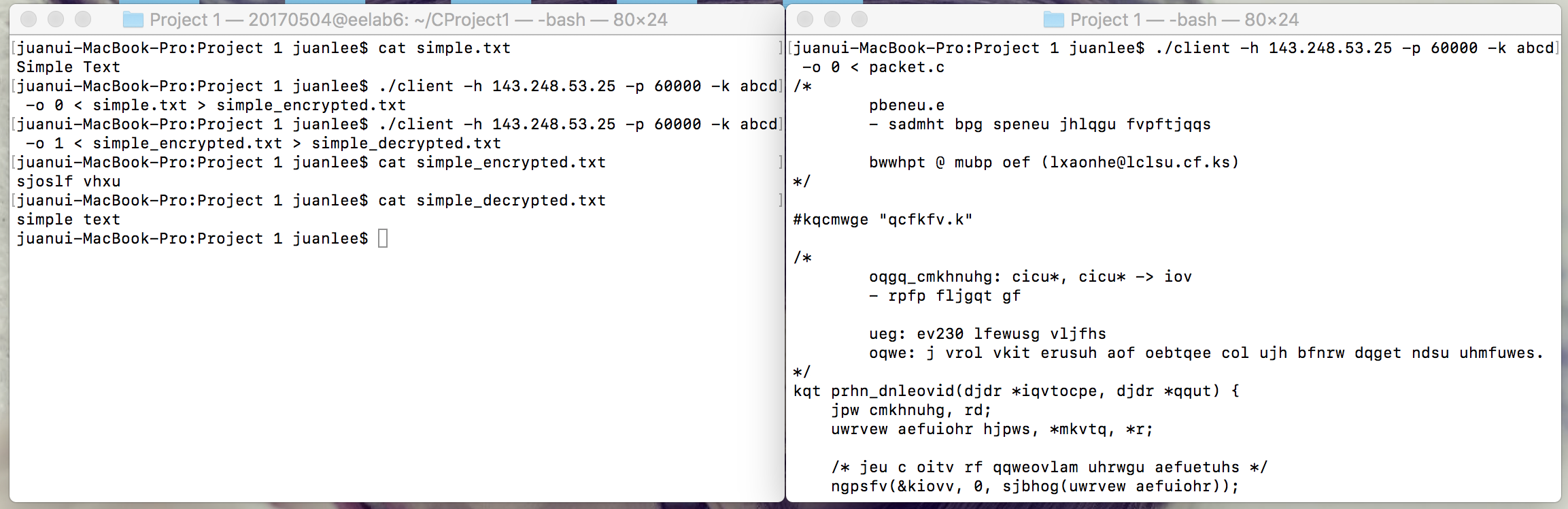
Since the data field is defined as a pointer in the packet, I separate the calculation for the header and that for the data.

**About Client (Client.c)**

There are three parts for client, part for checking parameters, that for getting input from stdin, and that for communicating with server. Basically, I thoroughly follow the instruction, so parameter format of (-p, -o, -k, -h) is totally same as PDF. The order does not matter.

For getting-input part, I dynamically allocate a buffer as 10MB-size array. I get the input with size of 1KB, and copied it at the end of the buffer. I simply used the final buffer for communication with server.

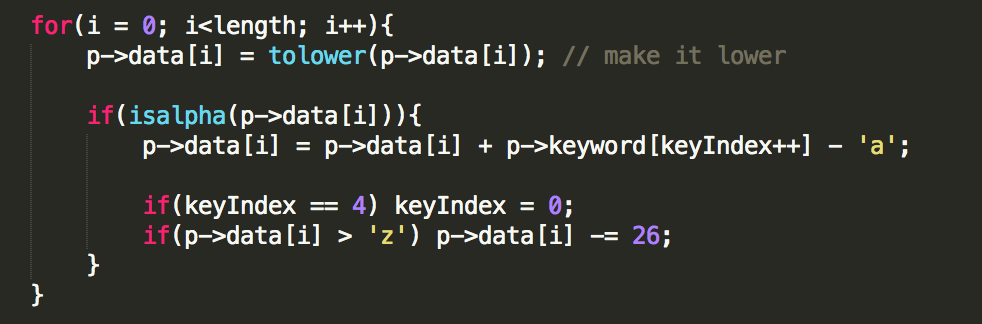
For communication part with server, I used packet functions. I created a packet, set fields except checksum, calculated and set checksum, send packet to server, and destroy it. After sending, I created new packet to receive, receive from server, check checksum, print the data out if it is right, and destroy it. I strictly follow the communication protocol described on PDF.



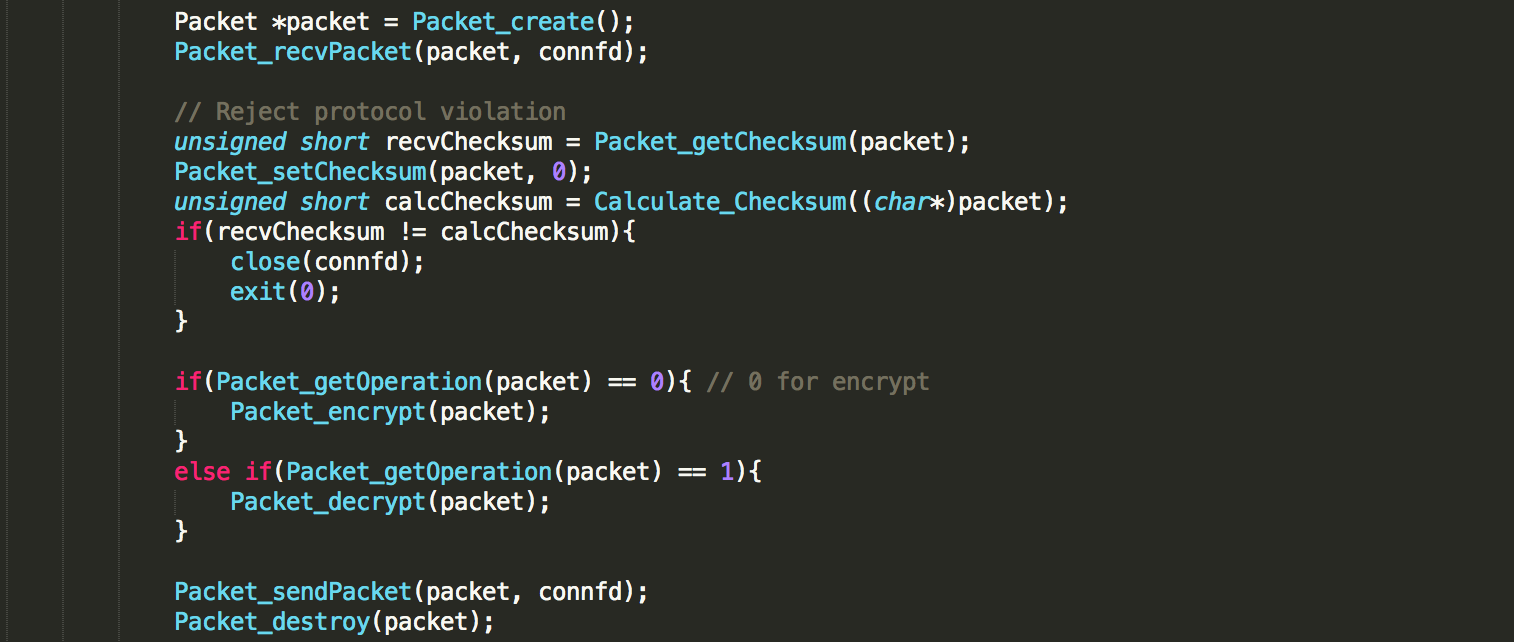
**About Server (server.c / server\_select.c)**

Firstly, I made the server with fork function. Since checking parameter part is very similar to the client, I changed only the number of parameters that server receives. At the communication part with client, I just created a packet, received from client, check the checksum, do operation regarding op-code, send the changed packet, and destroy it. All the packet-related functions are from the packet.h header.

Here is encryption algorithm I wrote, and decryption works similarly.



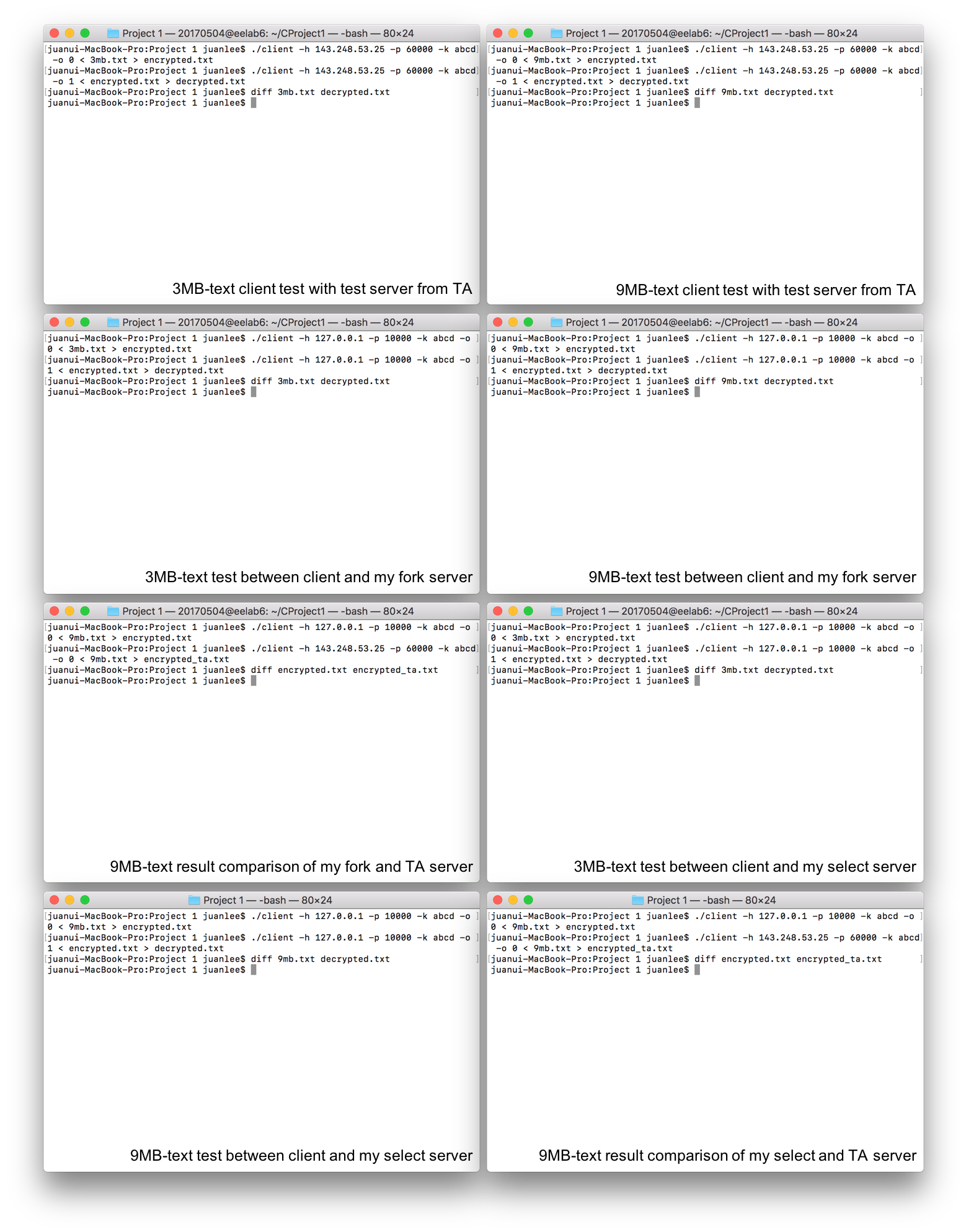
Secondly, I made the server with select function. Select function is a way to implement non-blocking server. I referenced CS230 System Programming lecture slides for the implementation. Select server is very similar to first server, but different in the communication part. It checks all the clients routinely, instead fork and proceed the messages from client. If some client is ready to communicate, the server activates and does same routine of code for packet encryption and decryption.



This is main part for both servers. Sorry for small letters.

**Self-test Results**

All the test cases work well, and below is the result image of each test case. (text is randomly selected alphabet with size of 9MB and 3MB)



It runs on mac os, however, I test same code on Ubuntu environment.

**How to Compile**

I wrote a makefile, which compiles all the client and servers. So, if you simply type ‘make all’ or ‘make’, you will get three executable programs, client, server, and server\_select with some object files. Or, if you want to compile individually, use like below:

* gcc -o client client.c packet.c
* gcc -o server server.c packet.c
* gcc -o server\_select server\_select.c packet.c

**References**

I got helps from many sources, including source code for solving specific problem. I list them and briefly explain what I got from it.

* Instruction pdf – how to encrypt, decrypt and packet format
* CS230 System Programming lecture slides – open listen and client file descriptor, and way to use select functions.
* http://locklessinc.com/articles/tcp\_checksum/ - checkcum
* KLMS, Stackoverflow – some questions about socket functions