Lab 3: UDP File Transer

Client Side:
The client uses the window slide protocol to send out the packets to the server.

Functions including:
int create_socket()
- Socket creation.

*void connect_to_server(int sock, struct sockaddr_in * server_addr, const char * ip, int port)*
- Connect to the server

*void parse_input(int argc, char *argv[], char *ip, int *port, int *mss, int *win_size, char *in_file, char *out_file)*
- Parse the input and give out the variables to different functions

*void get_rfc_time(char * buffer, size_t len)*
- Get the timestamp for the stdout log

void send_file(int sockfd, struct sockaddr_in server_addr, int mss, int win_size, const char* in_file_path, const char* out_file_path)*
- Implement the sliding window protocol for the program
- Send out the filename(outfile path) as the first packet
- If the server got the filename and created the outfile then sent the rest of the packets
- If the packets in the window did not receive an acknowledgment, try again until meeting the maximum retries of 5
- Problem complete

Server Side:
Receive and store the packets from the client locally.

Functions including:
*ClientData * get_client(struct sockaddr_in * client_addr)*
- Created a struct called *ClientData* for events when multiple clients are connecting to the server.
- This function helps get the information about client

*void get_rfc_time(char * buffer, size_t len)*
- Get timestamp

int create_socket()
- Create the socket

int start_server(int port_number)
- Start the server and bind to the port number

*void create_output_directory(const char * filepath)*
- If the output directory doesn’t exist, create one

*void send_ack(int sockfd, struct sockaddr_in * client_addr, int ack_num, int drop_rate)*
- Taking the timestamp to send the log

void receive_packets(int sockfd, int drop_rate)
- If the packet’s sequence number is 0, it is the packet containing the outfile_path
- The rest is the body, write them to the file opened earlier

Testcases:

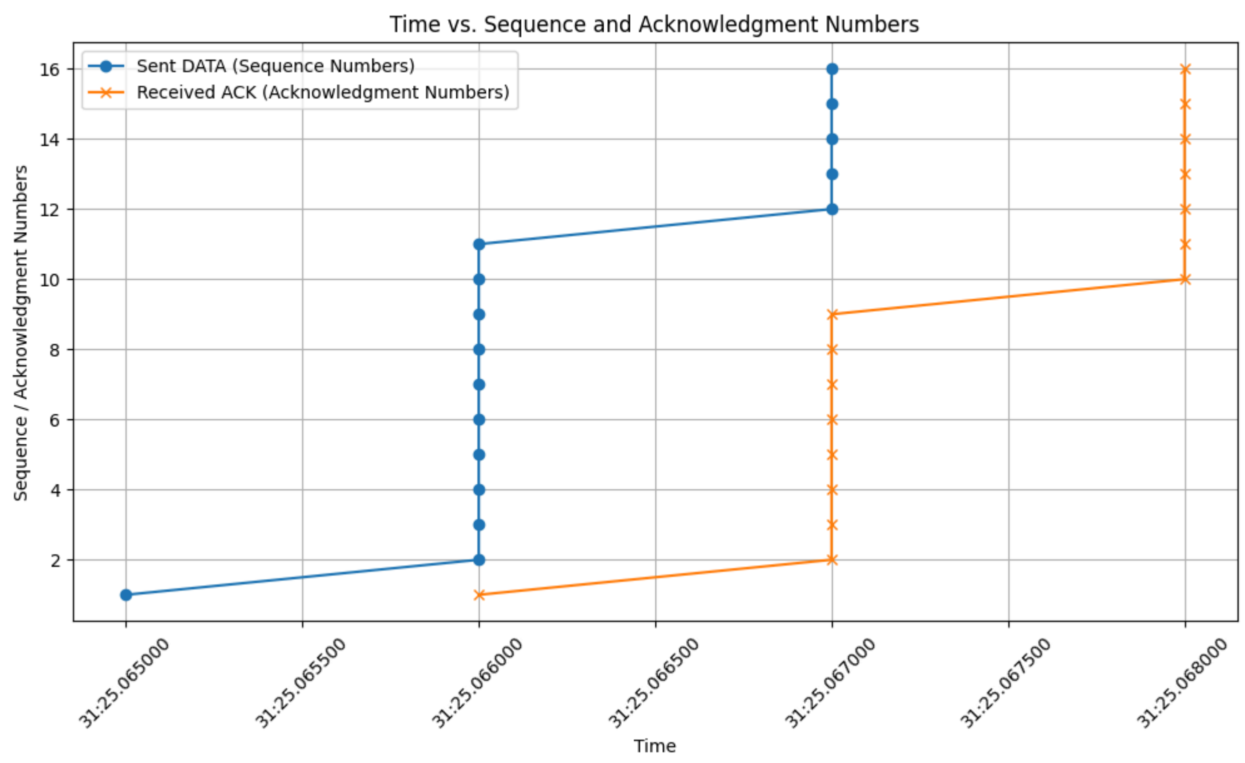
1. Large File Transfer with No Loss
   Test the basic functionality for file transfer with no packet drops

   Commands:
   ```
   ./bin/myserver 9090 0
   ./bin/myclient 127.0.0.1 9090 512 10 largefile.txt server_largefile.txt
   ```

2. High Packet Loss with 50 %
   Commands:
   ```
   ./bin/myserver 9090 50
   ./bin/myclient 127.0.0.1 9090 512 10 testfile.txt server_testfile.txt
   ```
   Result: it takes a while
3. Server Crash
   Commands:
   ```
   ./bin/myserver 9090 50
   ./bin/myclient 127.0.0.1 9090 512 10 testfile.txt server_testfile.txt
   pkill -f myserver
   ```

4. Invalid MSS Size
   Commands:
   ```
   ./bin/myserver 9090 50
   ./bin/myclient 127.0.0.1 9090 10 10 testfile.txt server_testfile.txt
   ```

5. Zero Length File
   Commands:
   ```
   ./bin/myserver 9090 50
   touch emptyfile.txt
   ./bin/myclient 127.0.0.1 9090 512 10 emptyfile.txt server_emptyfile.txt
   ```


Time vs. Sequence and Acknowledgment Numbers
Sent DATA (Sequence Numbers)
Received ACK (Acknowledgment Numbers)
Sequence / Acknowledgment Numbers
2
4
6
8
10
12
14
16
31:25.065000
31:25.065500
31:25.066000
31:25.066500
31:25.067000
31:25.067500
31:25.068000
Time