Lab 4: UDP File Transfer

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

Functions including:
*void log_event(const char *event, int seq_num, int base_sn, int next_sn, int win_size, struct sockaddr_in *server_addr)*

- Log the event with this format
  time, lport, rip, rport, DATA or ACK, pktsn, basesn, nextsn, basesn + winsz

$void\ *\ send\_file\_to\_server(void\ *\ args)$

- Send the packet to the server by using selective repeat window slides
- Send the file name first and then send the rest of the body with window slides
- When a certain ack is not received from the server
- Retry for at least 3 times and record as lost packet when meeting 5 times, and stop

$int\ main(int\ argc,\ char\ *\ argv[])$

- Takes the input command
- Perform send file to server using threads
- Each thread send file to server concurrently
- Create connection to the server using a socket

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 log_event(const char *event, int seq_num, struct sockaddr_in *client_addr)*

- Logging

$void\ create\_directory\_recursive(char\ *\ path)$

- If the output directory doesn't exist, create one

$void\ receive\_packets(int\ sockfd,\ int\ droppc,\ const\ char\ *\ root\_folder)$

- Receive the first packet as the filename
- Collect the rest of the file as the body
- When receiving out of order packet, store it in the buffer
- When the out of order packet got written in the file, remove it from the buffer

$int\ main(int\ argc,\ char\ *\ argv[])$

- Takes the input command
- Checking the validity of the input
- Call receive packets

Test Cases:

1. Small file (50KB) with 0 droppc and 1 server
   Test the basic functionality for file transfer with no packet drops

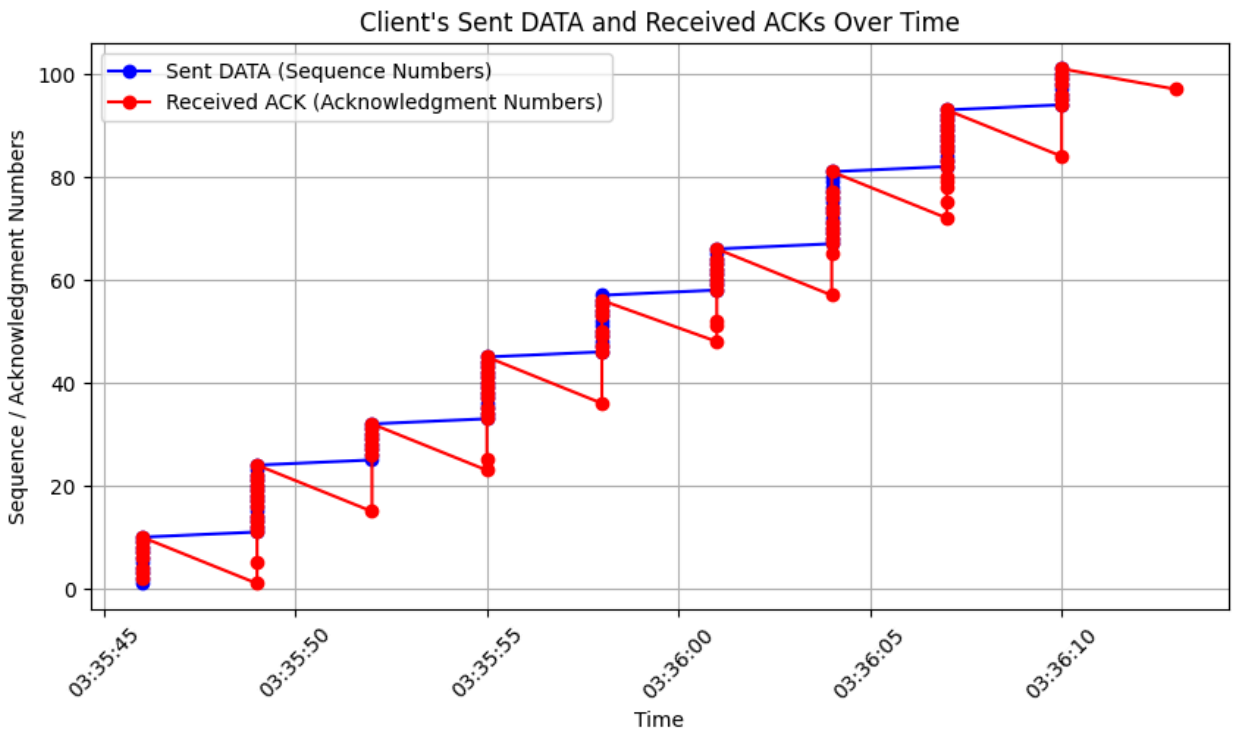
   Commands:
   . $/myserver$ 9001 0 ./server_stoage
   . $/myclient$ 1 $servaddr.conf$ 512 10 $input.txt$ $output.txt$

2. Packet Loss with 10 %
   Test the basic functionality and ability for retransmission
   Commands:
   . $/myserver$ 9001 10 ./server_stoage
   . $/myclient$ 1 $servaddr.conf$ 512 10 $input.txt$ $output.txt$

3. Two servers
   Commands:
   . $/myserver$ 9001 10 ./server_stoage &
   . $/myserver$ 9002 10 ./server1_stoage &
   . $/myclient$ 2 $servaddr.conf$ 512 10 $input.txt$ $output.txt$

4. Invalid MSS Size
   Commands:
   . $/myserver$ 9001 10 ./server_stoage &
   . $/myserver$ 9002 10 ./server1_stoage &
   . $/myclient$ 2 $servaddr.conf$ 510 10 $input.txt$ $output.txt$

5. Large input file
   Commands:
   . $/myserver$ 9001 10 ./server_stoage &
   . $/myserver$ 9002 10 ./server1_stoage &
   . $/myclient$ 2 $servaddr.conf$ 510 10 $input.txt$ $output.txt$


Client's Sent DATA and Received ACKs Over Time
Sent DATA (Sequence Numbers)
Received ACK (Acknowledgment Numbers)
Sequence / Acknowledgment Numbers
0
20
40
60
80
100
03:35:45
03:35:50
03:35:55
03:36:00
03:36:05
03:36:10
Time