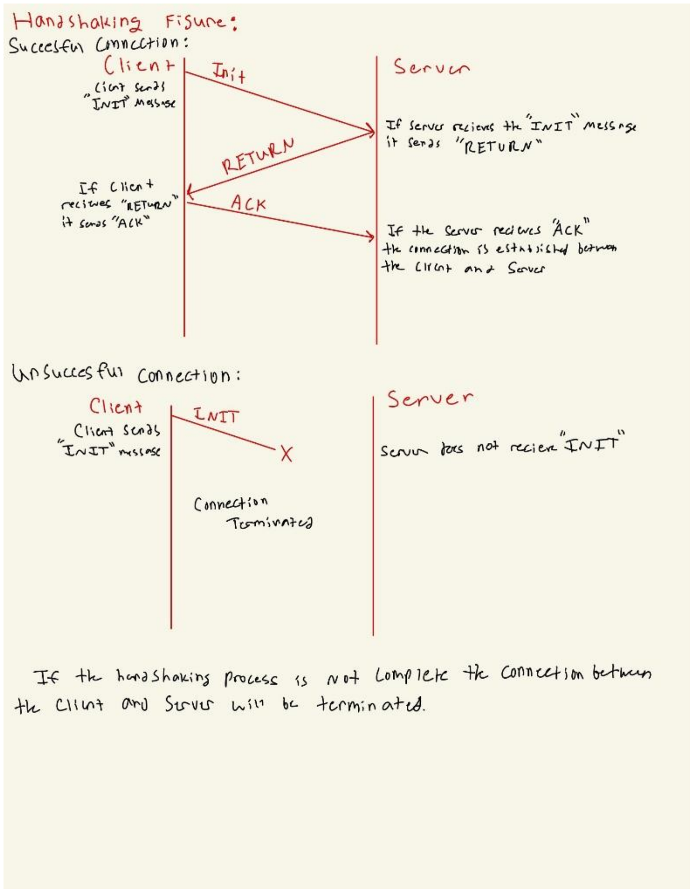
Introduction:

We created our application layer protocol with a couple things in mind, speed and privacy between the server and client. We wanted to create a way to transmit messages to a server from the client over a UDP connection quickly to the server for message storage. The server will store the history of every message from the client. UDP was the right fit for our protocol because it is a lightweight datagram transportation layer protocol meaning quicker speeds for message transfer. Also, since we developed our handshaking method for our protocol (described in a later section), we didn't believe TCPs handshaking was necessary for our messages from the client to the server. No status codes are outputted to the terminal unless and error or connection termination occurs.

Handshaking Process:

The handshaking process (Figure 1) is similar to TCPs but uses "INIT", "RETURN" and "ACK" packets. The client reaches out for connection sending an INIT packet and verifies it is an INIT packet, to which the server responds with a RETURN packet. The client then receives the RETURN packet and verifies it as a RETURN packet. If so, an ACK packet is sent to the server and verified. If all packets are verified, then a connection is established. The purpose of this handshaking process is to ensure a connection. If the connection is successful, both the server and client print "HANDSHAKE COMPLETE". If the connection is unsuccessful the connection is terminated. Once connected, the client is prompted to enter a password developed by a team of skilled developers. After the password is sent, the server and client will output they connected successfully, if the password is correct. If after 3 attempts, the client has still not entered the correct password, the server closes the connection.


Handshaking Figure:
Succesful Connection:
Client
Init
Server
Client sends "INIT" message
If Server recieves the "INIT" message it sends "RETURN"
RETURN
If Client recieves "RETURN" it sends "ACK"
ACK
If the Server recieves "ACK" the connection is established between the Client and Server
Unsuccesful Connection:
Client
INIT
X
Server
Client sends "INIT" message
Server does not recieve "INIT"
Connection Terminated

If the handshaking process is not complete the connection between the Client and Server will be terminated.

Figure 1: The Handshaking Process

While Connected:

After connecting to the server, the client will be prompted to send messages to the server (Figure 2). These messages will be sent in packets to the server. The server does not send packets back to the client. The server stores every part of the message including the header information and the payload (client message).

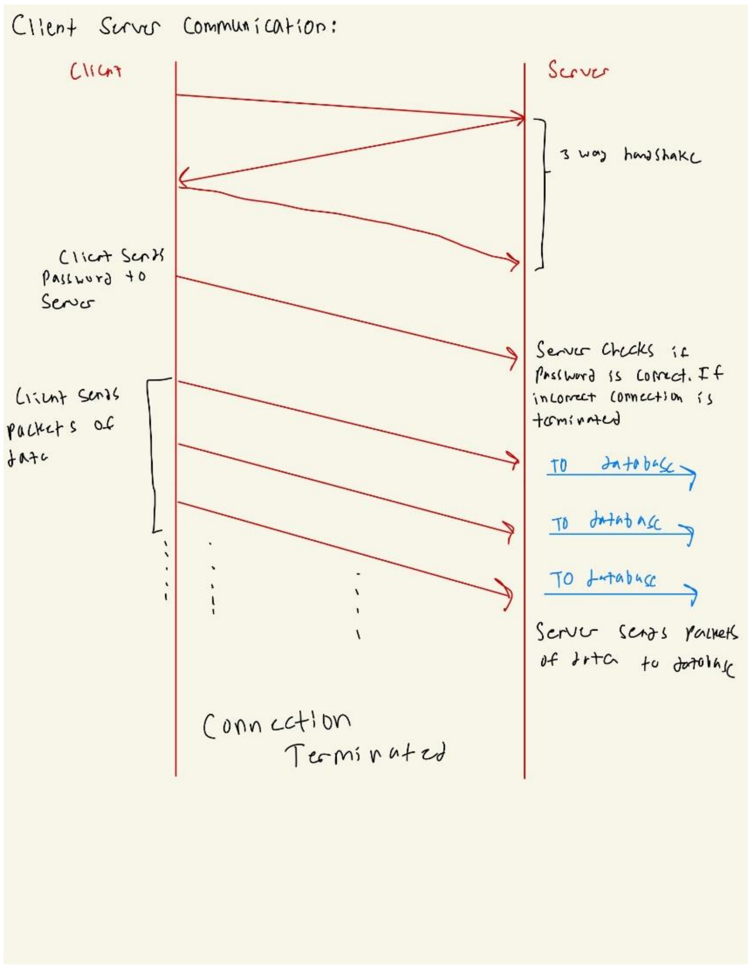

Client Server Communication:
Client
Server
3 way handshake
Client Sends Password to Server
Server Checks if Password is Correct. If incorrect connection is terminated
Client Sends packets of data
To database
To database
To database
Server Sends Packets of data to database
Connection Terminated

Figure 2: Client and Server Communication

Closing Connection:

To close connection, the client should send “CLOSE CONNECTION” to the server. “DISCONNECTED FROM SERVER” is printed to the client's terminal and “CLIENT DISCONNECTED” is printed to the server's terminal.

Formatting of the Protocol Messages:

Each datagram is sent over a UDP connection between the client and the server. Datagrams sent by our protocol can have at the most 1024 bytes of information which includes the 14-byte header (Figure 3). The header consists of 5 parts:

4-byte packet ID, to differentiate the packet, cannot be negative

4-byte source IP address, the address of the incoming message, cannot be negative

4-byte destination IP address, the address that receives the message, cannot be negative

2-bytes to represent the length of the data payload, cannot be negative

1010-max bytes payload, the actual content of the message the client wants to send to the server

If any of the header fields do not follow the correct format or if the packet is too large, the packet will be dropped. We capped the size of the packet to 1024 bytes because we were keen on speed, so we did not want users to send big messages that can possibly slow down and have a higher chance of being corrupted.

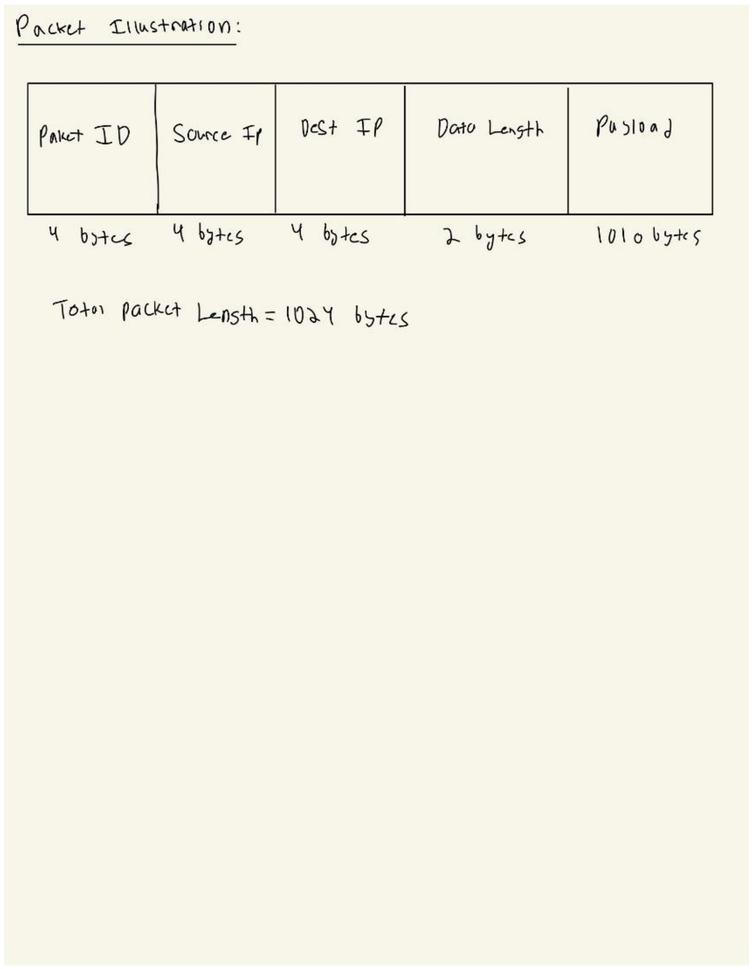


Figure 3: The Packet