COMPENG 4DN4 Lab 3 Report

Aaron Pinto
pintoa9
400190637

Raeed Hassan
hassam41
400188200

March 26, 2023

Server Class

The server class code is run from the terminal and will listen for client requests made with TCP connections and respond with student grade information. When the server class initializes itself, it will open and parse the `course_grades_2023.csv` file containing the student grade information using the `csv` python package. The student grade information will be printed on the command line for the server host.

The server client will then create an IPv4 TCP socket, and bind the socket to `localhost` (as the client will be running on the same host) on port 50007. The server socket is set to the listen state, and will be listening for any attempted TCP connections from the client. The server will block while waiting for accepting incoming connections, and will pass the socket to a connection handler function when a connection is accepted.

The connection handler will attempt to receive bytes over the TCP connection. If no bytes are received from the connection, then we close the connection. Otherwise, we receive a command input from the client that we can use to parse the appropriate student grade information. If the student number received does not match to any in the CSV file, we will close the connection.

If the student number received does match any entry from the CSV file, we will match the student with the appropriate information requested by the command and encrypt the information. The encryption is done using the `cryptography` package and its `Fernet` class, using the encryption key found in the CSV file matching each student. The encrypted message is sent to client through the TCP connection, and then we close the connection.

When we close the connection after responding to the student, or if we close the connection due to a non-matching student or not receiving a message through the connection, the server will return to listening for any new incoming connections and will repeat the process described above until the program is stopped.

Client Class

The client class code is run from the terminal and will have a local file sharing directory, and be able to scan for and connect to file sharing services to transfer files.

The client will allow the user to scan for `SERVICE DISCOVERY` broadcasts. When the user requests a `scan`, the client will create a UDP socket, and broadcast a service discovery packet. The client will listen for service responses, and will return a list of the available file sharing services. The client will then be able to `connect` to a file sharing service by establishing a TDP connection.

The client will have file sharing commands available. It will be able to output the contents of the local file sharing directory with the `llist` command. When the client has established a connection with a file sharing server, it will also have access to additional commands, `rlist`, `put`, and `get`. The three file sharing commands will send a packet with the format of the packet determined by the command.

When a `rlist` command is issued, the client will send a single byte integer designated for the `list` command, and receive eight bytes from the server which encode the response size, and then will receive the number of bytes required for the listing response.

When a `get` command is issued, the client will send a single byte integer designated for the `get` command, followed by one bytes that encodes the filename size, followed by the filename. The client will receive eight bytes from the server which encode the file size, and then will receive the number of bytes required for the file. The file will then be written to the local file sharing directory.

When a `put` command is issued, the client will send a single byte integer designated for the `put` command, followed by one byte that encodes the filename size, followed by the filename, followed by eight bytes that encode the file size, followed by the file being sent to the server.

The client will close the connection on the socket when an error occurs, or when no response is received from the server. The client will also close the connection when the `bye` command is issued.

Member Contributions

Aaron contributed to the server class, Raeed contributed to the client class.