PROJECT 2 REPORT

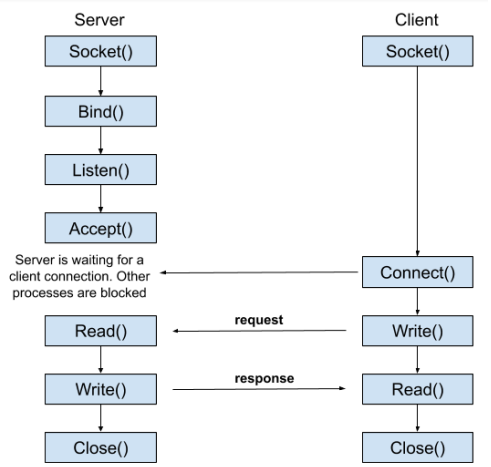
1.INTRODUCTION

In this project, we set out to explore the TCP protocol and socket programming. We've built a system where the client can send student information to another computer server via the TCP protocol. The server stores this data, and the client can later check if everything was delivered correctly. Both client and server acts as a sender and receiver in our task.

2. MATERIALS AND METHODS

Server works in a way that creating a TCP socket (serverSocket) and binding it to the IP address and port 5050. This means the server will listen on all available network interfaces and on port 5050. Created socket will be used for byte streaming as “SOCK_STREAM” indicates and it will also uses IPv4 as “AF_INET” indicates. Then, it enters a listening state, waiting for incoming client connections. It can handle only one connection at a time due to the “1” argument.

When a client connects to the server, the code accepts the connection, creating a new socket (connectionSocket) for communication with the client. It then receives an operation choice from the client, which should be either "u" or "d" (upload or download). If the received operation choice is "u", it receives the filename from the client, opens the file with the same name on the server, and writes the content received from the client into the file. Finally, it closes the file, indicating the successful upload. If the operation choice is "d," it reads the content of a file named "modified_file.txt" and sends both the filename and the content back to the client over the network. This allows the client to download the file. After performing the requested operation or indicating that the operation is undefined, the server closes the connection with the client using connectionSocket.close() and returns to listening for incoming connections.



The client code starts by defining the server's address as "localhost" and the port as 5050. This information is used to establish a connection with the server. The client creates a TCP socket (clientSocket) and connects to the server. The user is prompted to input either "u" for upload or "d" for download. If the user chooses to upload, the client reads the content of a local file named "file.txt". It sends the server the operation choice "u", the name of the file to be created on the server ("modified_file.txt"), and the content of the file. If the user chooses to download, the client

sends the server the operation choice "d". It receives the filename and content from the server, writes the content to a file named after the received filename ("modified_file.txt"), and then compares the transmitted file's content with the original "file.txt". If the contents match, it prints "Successfully transmitted."; otherwise, it prints "Unsuccessfully transmitted." After performing the upload/download, the client closes the connection with the server using clientSocket.close().

3.RESULTS

The successful file transfer results indicate that no data loss occurred during the process. This may be due to the use of the TCP protocol, which is designed to prevent data loss and ensure reliable transmission. Additionally, the local environment of the system, where the transfer occurs, minimizes external factors that could disrupt the process. However, it's important to note that the code is basic, handling only one connection at a time and using simple text-based communication for file transfer.

4. SUMMARY

In this project, I've learned the inner workings of sockets and how to implement socket operations using Python. Through the socket module, I gained knowledge about creating and configuring sockets, which is crucial for establishing connections and enabling communication between clients and servers over a network.

I also acquired hands-on experience in setting up a server by creating a listening socket, which patiently awaits incoming client connections. This foundation allowed me to comprehend the dynamics of server-client interactions in networked applications.

Additionally, I explored the practical aspect of file transfer, understanding how clients can send data to servers and vice versa. I saw the process of uploading files from a client to a server and downloading files from the server to the client, illustrating the two-way nature of these operations.