Networking Through Python and Multiplayer Game

Project Presentation

Ku. Khushi (2108390100030)

Mentor: Mr. Shashank Yadav

Rajkiya Engineering College, Kannauj

Department of Computer Science and Engineering

November 26, 2024

Content

Objective

The objective of this project is to first implement key networking protocols—TCP/IP, FTP, and UDP—by developing server-client systems that demonstrate their functionality and real-world applications. The focus is on data transmission, error handling, and ensuring reliable communication.
Subsequently, the project transitions into the creation of a multiplayer game that uses socket programming and multithreading.

The game will:

- Allow multiple players to connect and interact with the server in real-time.
- Handle concurrent player actions and communication through multithreading.
- Ensure smooth and dynamic gameplay with efficient server-client communication.

The goal of this project is to combine networking theory with practical implementation, showcasing the use of protocols and multithreading in a real-world multiplayer environment.

Motivation

In today's digital world, multiplayer games and real-time communication systems are essential in various sectors like gaming, remote work, and collaboration. This project aims to provide hands-on experience with networking protocols (TCP/IP, FTP, UDP) and their real-world applications, such as file transfers and messaging. The project also focuses on creating a multiplayer game that ensures efficient client-server communication using socket programming and multithreading for concurrency management. By implementing these protocols and building a real-world game, this project combines theoretical knowledge with practical skills, offering a deeper understanding of networking concepts in an interactive and engaging way.

Problem Statement

This project addresses the challenges of implementing networking protocols and real-time communication between multiple clients. It first focuses on hands-on implementation of TCP/IP, FTP, and UDP to understand data transmission and error handling. The second phase involves creating a multiplayer game where players interact over a network, using socket programming and multithreading to manage concurrent actions. Key challenges include protocol implementation, handling multiple connections, managing errors, and ensuring real-time communication. This project aims to provide both a theoretical foundation and practical skills in network programming for multiplayer environments.

Implementation

This project implements a server-client system using Python's socket programming and multithreading capabilities. The server is designed to handle multiple client connections, support file transfers, and echo chat messages back to the clients. The system allows clients to send and receive files while also providing a messaging interface for real-time communication.

Server-Side Implementation

The server is responsible for:

- **Accepting Client Connections:** The server continuously listens for incoming connections from clients. Upon accepting a connection, the server assigns a separate thread to handle communication with each client.

- **Handling Client Communication:** The server uses a dedicated thread to manage each client. It listens for commands from the client (such as sending or receiving files, or messaging) and processes them accordingly.
- **File Transfer:** The server handles file transfers, either sending files to clients or receiving files from them. It reads and writes the files in chunks to ensure efficient data handling.
- **Error Handling:** The server gracefully manages connection issues and file transmission errors, ensuring that resources are properly cleaned up when a client disconnects.

Client-Side Implementation

The client application facilitates the following functionalities:

- **Establishing Connection:** The client connects to the server using the server's IP address and port number.
- **Sending Messages:** Clients can send text messages to the server. The server responds by echoing the received message back to the client.
- **Sending and Receiving Files:** The client can choose to send or receive files. It first sends a request to the server, specifying the file's name and size, and then transmits or receives the file in chunks.
- **Error Handling:** The client handles various errors, such as file not found or connection issues, and provides feedback to the user.

Flowchart

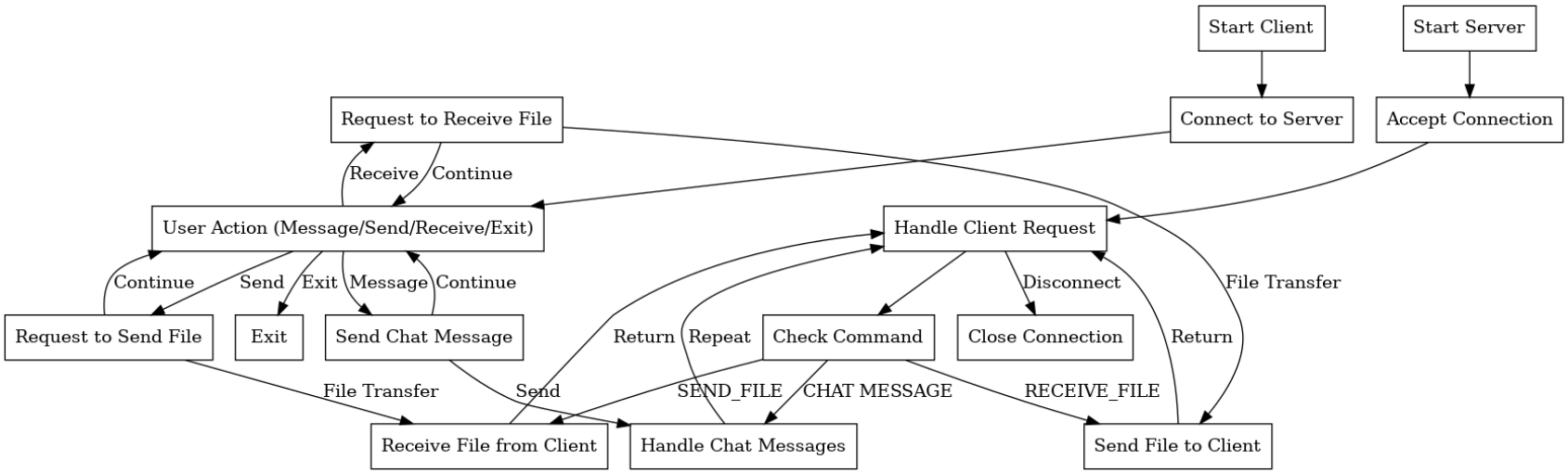

Figure: Flowchart

Results

```
PS C:\Users\mishr\Downloads\ftp> & C:/Users/mishr/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/mishr/Downlo
ads/ftp/Sever Socket.py"
[INFO] Server is listening on port 5555...
[INFO] Connection established with ('192.168.232.25', 64081)
[CHAT] ('192.168.232.25', 64081): hello server
[INFO] Connection established with ('192.168.232.25', 64100)
[CHAT] ('192.168.232.25', 64100): hii server
[INFO] Receiving file 'adhar card.pdf' of size 322121 bytes...
[SUCCESS] File 'adhar card.pdf' received successfully.
[INFO] Receiving file 'CV.pdf' of size 30482 bytes...
[SUCCESS] File 'CV.pdf' received successfully.
```

Figure: Server

```
PS C:\Users\mishr\Downloads\ftp> & C:/Users/mishr/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/mishr/Downlo
ads/ftp/client Socket.py"
[INFO] Connected to the server.
Enter 'message', 'send', 'receive', or 'exit': message
Enter your message: hello server
Server: Server Echo: hello server
Enter 'message', 'send', 'receive', or 'exit': send
Enter the filename to send: C:\Users\mishr\Downloads\adhar card.pdf
[INFO] Sending file 'C:\Users\mishr\Downloads\adhar card.pdf' of size 322121 bytes...
[SUCCESS] File received successfully.
Enter 'message', 'send', 'receive', or 'exit':
```

Figure: Client 1

Results

```
PS C:\Users\mishr\Downloads\ftp> & C:/Users/mishr/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/mishr/Downlo
ads/ftp/client 2.py"
[INFO] Connected to the server.
Enter 'message', 'send', 'receive', or 'exit': message
Enter your message: hii server
Server: Server Echo: hii server
Enter 'message', 'send', 'receive', or 'exit': send
Enter the filename to send: C:\Users\mishr\Downloads\CV.pdf
[INFO] Sending file 'C:\Users\mishr\Downloads\CV.pdf' of size 30482 bytes...
[SUCCESS] File received successfully.
Enter 'message', 'send', 'receive', or 'exit':
```

Figure: Client 2


Thank You !