Assignment 1

CP372 - Assignment 1 Report

Authors

- **Vikas Movva**
- **Sohan Hossain**

Course Information

- **Course:** CP372 – Computer Networks
- **Assignment Title:** Socket Programming - Client-Server Communication Application
- **Submission Date:** Feb 10th, 2025

---

1. Introduction

This report documents the implementation of a client-server communication application using **TCP sockets** in Python. The server manages multiple concurrent clients, allowing them to send messages, request files, and retrieve connection statuses. The client is designed to interact with the server by sending and receiving messages, listing available files, and downloading files.

This implementation adheres to the project specifications, ensuring multi-threaded client handling, message acknowledgment, client identification, and controlled connection limits. The project was developed using Python and tested to ensure functional correctness.

---

2. Code Overview

2.1 Server (server.py)

The server is responsible for managing multiple client connections, enforcing connection limits, and responding to client requests. It is implemented using multi-threading to handle concurrent clients efficiently.

Key Functionalities

1. **Client Connection Management**
   - Accepts client connections and assigns them unique names in the format `"ClientXX"`, where XX is an incremental number.
   - Enforces a **maximum concurrent client limit** (set to 3). If the limit is reached, additional clients receive a message indicating that the server is full.
2. **Client Request Handling**

- Supports communication through text messages, with the server appending `"ACK"` to confirm receipt.
- Implements a `"status"` command to provide clients with a list of all connected and previously disconnected clients, including their connection timestamps.
- Responds to a `"list"` command by displaying available files in a predefined repository.
- Implements a `"get <filename>"` command, allowing clients to request files for download.

3. **Multi-Threaded Client Management**
   - Each client is handled by a separate thread, enabling concurrent communication.
   - Uses **thread synchronization mechanisms** (threading locks) to ensure consistency when modifying shared data structures.
4. **File Transfer Functionality**
   - The server maintains a repository of files that clients can request.
   - If a client requests a file, the server sends the file size, followed by the file contents in chunks.
   - Implements a start-end signal mechanism (`"START <size>"` and `"END"`) to ensure complete file transmission.
5. **Graceful Disconnection**
   - Clients can send `"exit"` to terminate the session.
   - When a client disconnects, the server updates the connection log and removes the client from active connections.

2.2 Client (client.py)

The client connects to the server and allows the user to send messages, check connection status, list available files, and download files.

Key Functionalities

1. **Connection Management**
   - Establishes a connection with the server and receives a unique client name.
   - If the server is at maximum capacity, the client receives a `"Server full, try later"` message and terminates the connection.
2. **Message Communication**
   - Users can input text messages, which are sent to the server and echoed back with an `"ACK"` acknowledgment.
   - The `"status"` command requests the list of connected and previously connected clients.
3. **File Handling**
   - The `"list"` command retrieves a list of available files on the server.
   - The `"get <filename>"` command downloads a specified file. The client uses **chunked file reception** to ensure integrity.
   - A **progress bar** (using the `tqdm` library) visually indicates the download progress.
4. **Graceful Disconnection**
   - The `"exit"` command cleanly terminates the session and notifies the server.

5. **Error Handling**
    - Implements exception handling to manage connection failures and unexpected disconnections.

---

3. Challenges and Solutions

Challenge 1: Managing Multiple Clients

- Handling concurrent connections required a robust implementation to avoid conflicts in shared resources.
- **Solution:** Multi-threading was implemented using the `threading` module. A threading lock was used to ensure consistency when modifying the shared client list.

Challenge 2: Ensuring Reliable File Transfer

- Ensuring complete file transmission required handling cases where data packets might be lost or incomplete.
- **Solution:** A **start-end signaling mechanism** was implemented to mark the beginning and end of a file transmission, ensuring completeness.

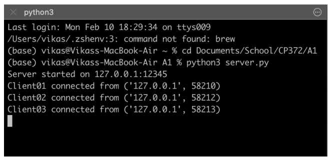
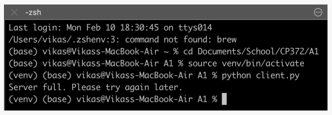
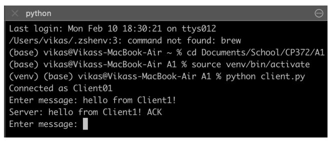
Challenge 3: Handling Client Disconnections

- Clients may disconnect unexpectedly, potentially leaving the system in an inconsistent state.
- **Solution:** The system was designed to handle abrupt disconnections gracefully by updating the client log and freeing up resources when a client disconnects.

---

4. Testing and Results

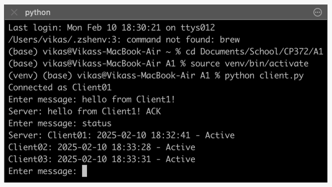
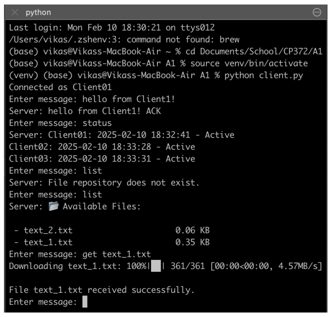
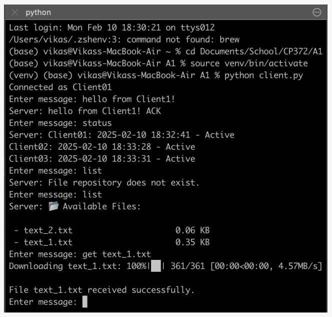
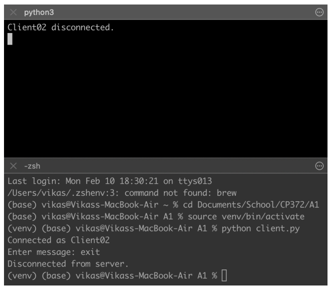
Several test cases were conducted to ensure the system met the assignment requirements. The results are summarized below.

Test Case	Expected Output	Actual Output	Result	Screenshot
Client Name Assignment	Each client receives a unique name (`Client01`, `Client02`, etc.)	Clients were assigned correct names	✅ Pass	
Max Client Limit Enforcement	A fourth client attempting to connect is rejected	Fourth client received "Server full" message	✅ Pass	
Message Communication	Clients send messages and receive `<message> ACK` response	Messages echoed with "ACK"	✅ Pass	

```
Last login: Mon Feb 10 18:29:34 on ttys009
/Users/vikas/.zshenv:3: command not found: brew
(base) vikas@Vikass-MacBook-Air ~ % cd Documents/School/CP372/A1
(base) vikas@Vikass-MacBook-Air A1 % python3 server.py
Server started on 127.0.0.1:12345
Client01 connected from ('127.0.0.1', 58210)
Client02 connected from ('127.0.0.1', 58212)
Client03 connected from ('127.0.0.1', 58213)
```

```
Last login: Mon Feb 10 18:30:45 on ttys014
/Users/vikas/.zshenv:3: command not found: brew
(base) vikas@Vikass-MacBook-Air ~ % cd Documents/School/CP372/A1
(base) vikas@Vikass-MacBook-Air A1 % source venv/bin/activate
(venv) (base) vikas@Vikass-MacBook-Air A1 % python client.py
Server full. Please try again later.
(venv) (base) vikas@Vikass-MacBook-Air A1 %
```

```
Last login: Mon Feb 10 18:30:21 on ttys012
/Users/vikas/.zshenv:3: command not found: brew
(base) vikas@Vikass-MacBook-Air ~ % cd Documents/School/CP372/A1
(base) vikas@Vikass-MacBook-Air A1 % source venv/bin/activate
(venv) (base) vikas@Vikass-MacBook-Air A1 % python client.py
Connected as Client01
Enter message: hello from Client1!
Server: hello from Client1! ACK
Enter message:
```

Test Case	Expected Output	Actual Output	Result	Screenshot
Client Status Query	Clients request connection status and receive a list of active/disconnected clients	Correct status returned	✅ Pass	
File Listing	Clients send `"list"` and receive a list of files in the repository	File list displayed correctly	✅ Pass	
File Download	Clients request files using `"get <filename>"`, receiving file with a progress bar	Files downloaded successfully	✅ Pass	
Graceful Disconnection	Clients send `"exit"` and disconnect cleanly	Clients removed from active list	✅ Pass	

```
python
Last login: Mon Feb 10 18:30:21 on ttys012
/Users/vikas/.zshenv:3: command not found: brew
(base) vikas@Vikass-MacBook-Air ~ % cd Documents/School/CP372/A1
(base) vikas@Vikass-MacBook-Air A1 % source venv/bin/activate
(venv) (base) vikas@Vikass-MacBook-Air A1 % python client.py
Connected as Client01
Enter message: hello from Client1!
Server: hello from Client1! ACK
Enter message: status
Server: Client01: 2025-02-10 18:32:41 - Active
Client02: 2025-02-10 18:33:28 - Active
Client03: 2025-02-10 18:33:31 - Active
Enter message: 
```

```
python
Last login: Mon Feb 10 18:30:21 on ttys012
/Users/vikas/.zshenv:3: command not found: brew
(base) vikas@Vikass-MacBook-Air ~ % cd Documents/School/CP372/A1
(base) vikas@Vikass-MacBook-Air A1 % source venv/bin/activate
(venv) (base) vikas@Vikass-MacBook-Air A1 % python client.py
Connected as Client01
Enter message: hello from Client1!
Server: hello from Client1! ACK
Enter message: status
Server: Client01: 2025-02-10 18:32:41 - Active
Client02: 2025-02-10 18:33:28 - Active
Client03: 2025-02-10 18:33:31 - Active
Enter message: list
Server: File repository does not exist.
Enter message: list
Server: 📂 Available Files:

 - text_2.txt                     0.06 KB
 - text_1.txt                     0.35 KB
Enter message: get text_1.txt
Downloading text_1.txt: 100%|█| 361/361 [00:00<00:00, 4.57MB/s]

File text_1.txt received successfully.
Enter message: 
```

```
python
Last login: Mon Feb 10 18:30:21 on ttys012
/Users/vikas/.zshenv:3: command not found: brew
(base) vikas@Vikass-MacBook-Air ~ % cd Documents/School/CP372/A1
(base) vikas@Vikass-MacBook-Air A1 % source venv/bin/activate
(venv) (base) vikas@Vikass-MacBook-Air A1 % python client.py
Connected as Client01
Enter message: hello from Client1!
Server: hello from Client1! ACK
Enter message: status
Server: Client01: 2025-02-10 18:32:41 - Active
Client02: 2025-02-10 18:33:28 - Active
Client03: 2025-02-10 18:33:31 - Active
Enter message: list
Server: File repository does not exist.
Enter message: list
Server: 📂 Available Files:

 - text_2.txt                     0.06 KB
 - text_1.txt                     0.35 KB
Enter message: get text_1.txt
Downloading text_1.txt: 100%|█| 361/361 [00:00<00:00, 4.57MB/s]

File text_1.txt received successfully.
Enter message: 
```

```
python3
Client02 disconnected.

-zsh
Last login: Mon Feb 10 18:30:21 on ttys013
/Users/vikas/.zshenv:3: command not found: brew
(base) vikas@Vikass-MacBook-Air ~ % cd Documents/School/CP372/A1
(base) vikas@Vikass-MacBook-Air A1 % source venv/bin/activate
(venv) (base) vikas@Vikass-MacBook-Air A1 % python client.py
Connected as Client02
Enter message: exit
Disconnected from server.
(venv) (base) vikas@Vikass-MacBook-Air A1 % 
```

---

5. Possible Improvements

While the implementation meets the assignment requirements, several enhancements could be considered with additional development time:

1. **Security Enhancements**
   - Currently, there is no authentication mechanism for clients. Implementing authentication would improve security.
2. **File Upload Functionality**
   - The system only supports file downloads. Adding an upload feature would allow clients to send files to the server.
3. **Persistent Storage for Client Logs**
   - Client connection logs are stored in memory and lost upon server shutdown. Implementing a **database or file-based logging** system would allow for persistent record-keeping.
4. **Improved Scalability**

- The current implementation limits the server to **three concurrent clients**. A more scalable architecture with **load balancing** could be explored.

5. **Enhanced User Interface**
   - The current interaction is **command-line based**. A graphical user interface (GUI) or web-based interface could improve usability.

---

6. Conclusion

This assignment provided valuable experience in **socket programming**, **multi-threaded server management**, and **network communication protocols**. The implementation successfully supports **concurrent clients**, **message exchange**, and **file transfer**, meeting all specified requirements. Testing confirmed that the system functions correctly under various scenarios, including handling multiple clients, enforcing connection limits, and managing file requests.

Future improvements could focus on enhancing security, scalability, and usability to make the application more robust and versatile.