File Transfer Over TCP/IP Using Sockets

Nguyen Thanh Nam - 22BI13325

1 Introduction

This report is about a file transfer system over TCP/IP using sockets. the system involves two components: a client and a server. The client sends a file in chunks to the server, which receives and stores it on disk.

2 Protocol Design

The communication between the client and server is based on the following protocol:

- **Client Side:**
  - Reads the file to be transferred.
  - Establishes a TCP connection to the server.
  - Sends the file in chunks (1024 bytes).
  - Closes the connection once the file is fully sent.
- **Server Side:**
  - Listens for incoming client connections on a specified port.
  - Receives the file in chunks.
  - Writes the received data to a file on disk.
  - Closes the connection once the file is fully received.

3 System Organization

```
+--------------------+      (connect)         +--------------------+
|                    | <----------------------> |                    |
|    Client          |      File Transfer       |    Server          |
|                    |   (send file chunks)     |                    |
+--------------------+ <----------------------> +--------------------+
       |                                                   |
       | (opens file, sends data in chunks)                |
       |                                                   |
       v                                                   v
+---------------------+                          +-----------------------------+
|  File: example.jpg  |                          |  File: received_example.jpg |
+---------------------+                          +-----------------------------+
```

3.1 Detailed Client-Server Interaction

The following is a step-by-step description of how the client and server interact during the file transfer:

```
Client                                   Server
-----------------------------------------------
open_clientfd -> connect                wait for connection
send file in chunks                    accept connection
-----------------------------------------------
write chunk -> send                    read chunk -> receive
-----------------------------------------------
write chunk -> send                    write to file
-----------------------------------------------
close connection                       close connection
```

4 Code Implementation

4.1 Client Code

Below is the Python code for the client that reads a file and sends it to the server.

```
import socket

SERVER_HOST = '127.0.0.1'
SERVER_PORT = 5001
BUFFER_SIZE = 1024
INPUT_FILE = "example.jpg"

def send_file():
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((SERVER_HOST, SERVER_PORT))
    print(f"[*] Connected to {SERVER_HOST}:{SERVER_PORT}")

    with open(INPUT_FILE, "rb") as f:
        print(f"[*] Sending {INPUT_FILE}...")
        while (data := f.read(BUFFER_SIZE)):
            client_socket.sendall(data)

    print("[+] File sent successfully.")

    client_socket.close()
    print("[*] Connection closed.")

if __name__ == "__main__":
    send_file()
```

4.2 Server Code

Below is the Python code for the server that listens for incoming connections and saves the received file.

```
import socket

SERVER_HOST = '0.0.0.0'
SERVER_PORT = 5001
BUFFER_SIZE = 1024
OUTPUT_FILE = "received_example.jpg"

def start_server():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    server_socket.bind((SERVER_HOST, SERVER_PORT))
    server_socket.listen(1)
    print(f"[*] Listening on {SERVER_HOST}:{SERVER_PORT}...")

    client_socket, client_address = server_socket.accept()
    print(f"[+] Accepted connection from {client_address}")

    with open(OUTPUT_FILE, "wb") as f:
        print("[*] Receiving file...")
        while True:
            data = client_socket.recv(BUFFER_SIZE)
            if not data:
                break
            f.write(data)

    print(f"[+] File received and saved as {OUTPUT_FILE}")

    client_socket.close()
    server_socket.close()
    print("[*] Connection closed.")

if __name__ == "__main__":
    start_server()
```

5 Who Does What

- **Client:**
  - Opens the file to be sent.
  - Establishes a TCP connection to the server.
  - Sends the file in chunks (1024 bytes).
  - Closes the connection after sending the file.
- **Server:**
  - Listens for incoming client connections.
  - Receives the file in chunks.
  - Writes the received chunks to the disk.
  - Closes the connection once the file transfer is complete.

6 Conclusion

This implements a simple file transfer protocol over TCP/IP using sockets. The client sends the file in chunks, and the server receives the file and stores it on disk. This basic implementation can be extended with additional features such as file validation, error handling, and encryption to secure file transfers.