Practical 1: TCP File transfer

Truong Hoang Khanh Linh - 22BI13254

1 Protocol design

The system is built on a simple model, including a client and a server communicating over TCP/IP.

- The server acts as a file receiver and listens for client connections.
- The client establishs a connection to the server and sends files.
- Connection is done over TCP/IP connection.

Below is a sequence diagram for protocol flow:

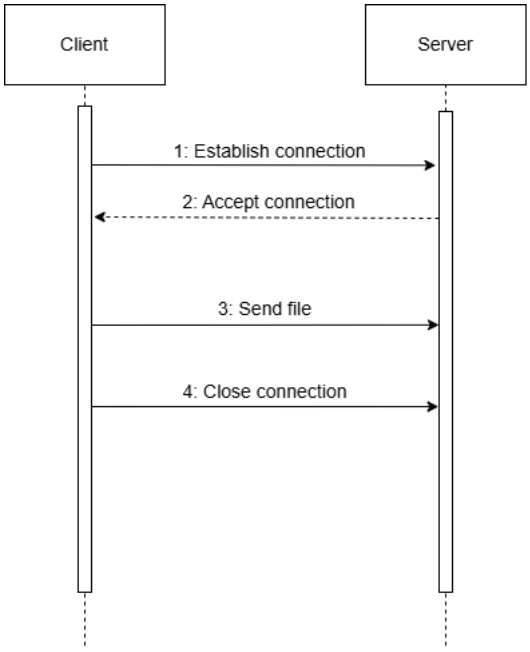

Figure 1: Protocol Flow

2 System organization

The system is organized with two oponents: a client and a server. Communication begins with the client establishing a connection to the server. The server, which is listening for incoming connections on a specified port, accepts the connection. After the connection is established, the client will send the file to the server in binary mode. The file is read in chunks of 1024 bytes and transmitted over the connection. The

server receives these and write them to a file. Once the transferation is completed, the connection will be closed.

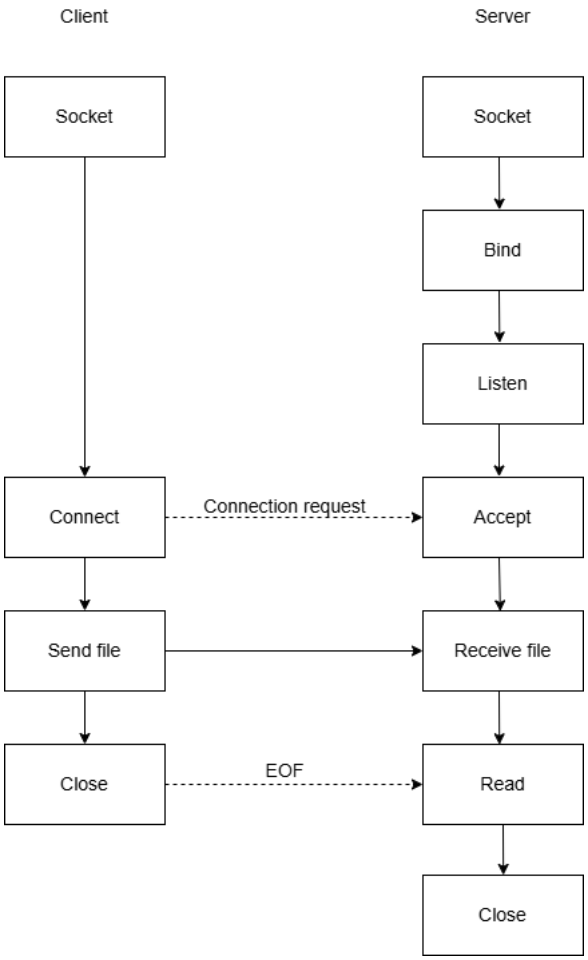

Figure 2: Sytem Architecture

3 Implementation

The system is implemented in Python using socket. The client and the server transfer file over TCP/IP.

3.1 The server

The server is binded to listen for incoming connections on a specified port using **bind()** and **listen()**. After receiving the connection request, the server accepts the connection with **accept()** method. The sent file will be received with **recv()**.

Below is the code for the server:

```
import socket

def start_server(host='0.0.0.0', port=888):
    # Create and configure server socket
    server_socket = socket.socket(socket.AF_INET, socket.
        SOCK_STREAM)
    server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR
        , 1)

    server_socket.bind((host, port))

    server_socket.listen(1)
    print(f"Server listening at {host} on port {port}...")

    client_socket, client_address = server_socket.accept()
    print(f"Connection established.")

    # Receive file
    with open('received_file', 'wb') as file:
        while True:
            data = client_socket.recv(1024)
            if not data:  # EOF
                break
            file.write(data)

    print("File received and saved as 'received_file'")
    client_socket.close()
    server_socket.close()

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

3.2 The client

The server first establishes a connection to the server using the **connect()** method. Once the server and client are connected, the client starts transmitting the file in chunks of 1024 bytes with the **send()** method. After the file is transfered successfully, the connection is closed with **close()**.

Below is the code for the client:

```python
import socket

def send_file(host, port, file_path):
    # Create client socket and connect to server
    client_socket = socket.socket(socket.AF_INET, socket.
        SOCK_STREAM)

    client_socket.connect((host, port))
    print(f"Connected to server {host} on port {port}.")

    # Send file
    with open(file_path, 'rb') as file:
        while True:
            data = file.read(1024)
            if not data:
                break
            client_socket.send(data)

    client_socket.close()
    print("File sent successfully!")
    print("Connection closed.")

if __name__ == "__main__":
    send_file("192.168.86.128", 888, "transfer_file.txt")
```

4 Testing

First, run **server.py** to set up the server, let it listen to connections on port 888.

```
D:\Study\USTH\B3\DistributedSystem\Practical1>python server.py
Server listening at 0.0.0.0 on port 888...
```

After that, run **client.py** to establish the connection and transfer the file to the server. The file is now transfered successfully as shown in the images below.

```
D:\Study\USTH\B3\DistributedSystem\Practical1>python client.py
Connected to server 192.168.86.128 on port 888.
File sent successfully!
Connection closed.
```

```
D:\Study\USTH\B3\DistributedSystem\Practical1>python server.py
Server listening at 0.0.0.0 on port 888...
Connection established.
File received and saved as 'received_file'.

D:\Study\USTH\B3\DistributedSystem\Practical1>type received_file
<This is transfer file>
```