TCP File Transfer Practical Work 1

Le Tuan Anh

1 Introduction

The purpose of this practical is to implement a simple file transfer system using TCP/IP. The work involves socket programming in Python, with one laptop acting as a server and the other as a client.

2 Protocol Design

The file transfer protocol involves:

- Establishing a connection between the client and the server.
- Sending the file name from the client to the server.
- Transferring the file in chunks of data.
- The server saving the file after all data is received.

3 System Interaction

The interaction involves two laptops, as shown below:

4 Implementation

The implementation involves two Python scripts: one for the server and one for the client.

4.1 Server Code

The server listens for incoming connections and saves the received file:

```
import socket

def start_server():
    server_ip = '192.168.162.200'  # Listen on all available
        interfaces
```

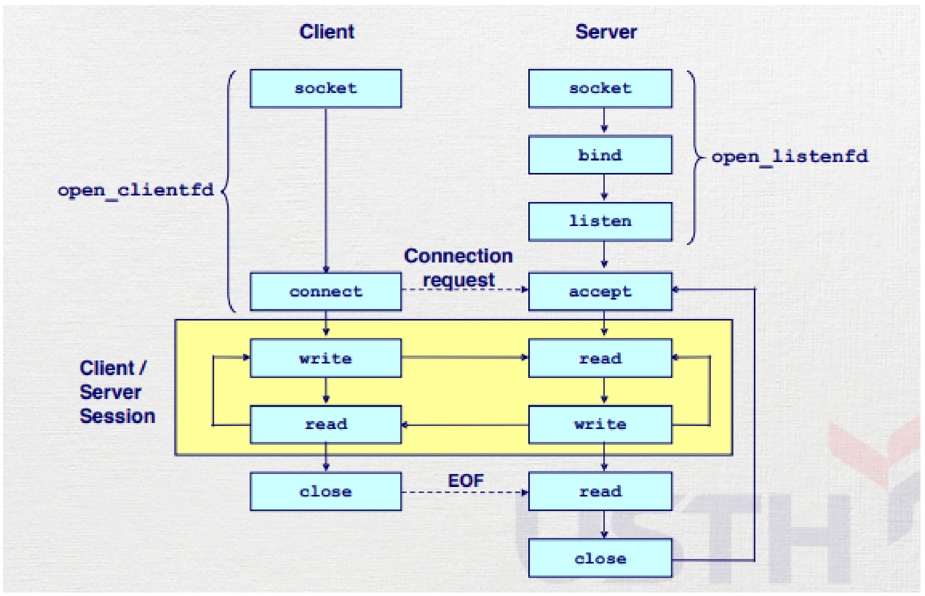

Figure 1: File Transfer Protocol Design

```python
    server_port = 33333     # Port to listen on

    # Create a socket object
    server_socket = socket.socket(socket.AF_INET, socket.
        SOCK_STREAM)
    server_socket.bind((server_ip, server_port))  # Bind to
        the specified IP and port
    server_socket.listen(1)  # Allow one client connection
        at a time
    print(f"Server is listening on {server_ip}:{server_port
        }...")

    # Accept a connection from the client
    conn, addr = server_socket.accept()
    print(f"Connection accepted from {addr}")

    # Receive the filename from the client
    filename = conn.recv(1024).decode()
    print(f"Receiving file: {filename}")

    # Receive the file content and save it
    with open(filename, 'wb') as file:
        while True:
            data = conn.recv(1024)  # Receive data in chunks
            if not data:
                break
```

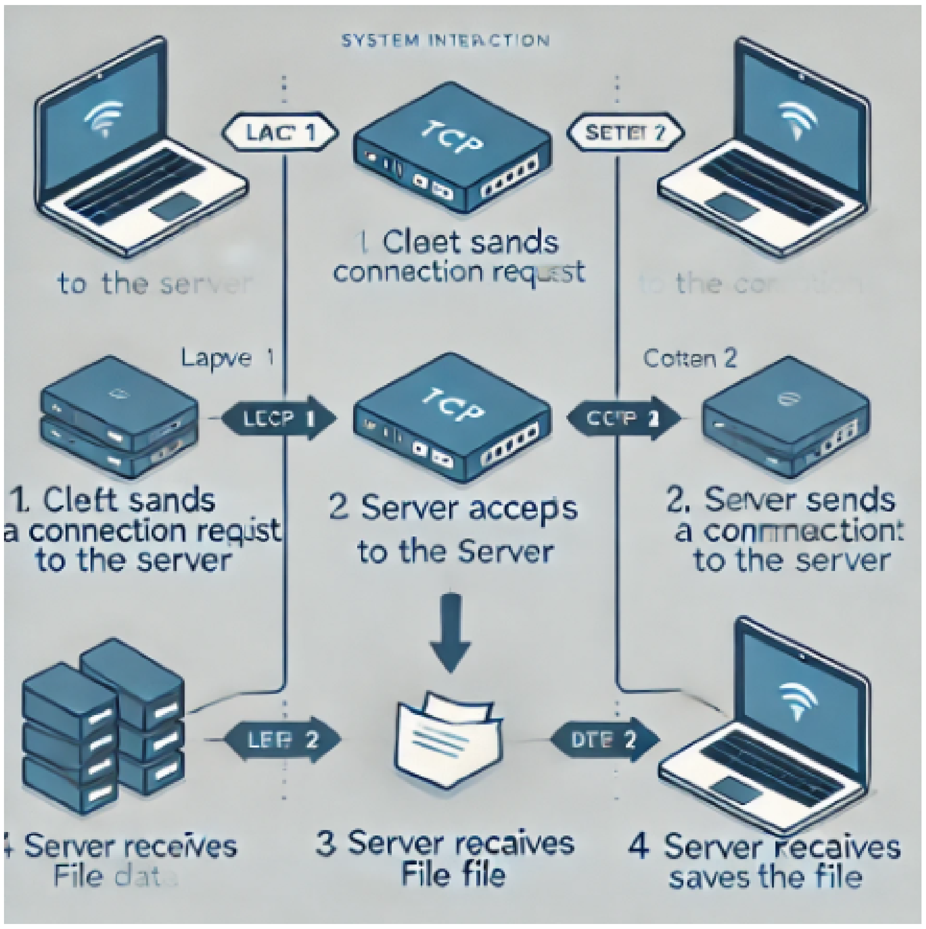
SYSTEM INTERACTION
LAC 1
TCP
SETEI 2
1 Cleet sands connection request
to the server
Lapve 1
Cotten 2
LECP 1
TCP
CCP 2
1. Cleft sands a connection reqist to the server
2 Server acceps to the Server
2. Server sends a conmnectiont to the server
LEF 2
DTE 2
4 Server receives File dat
3 Server recaives File file
4 Server recaives saves the file

Figure 2: System Interaction Diagram

```
            file.write(data)

    print(f"File {filename} received successfully.")
    conn.close()  # Close the connection
    server_socket.close()  # Close the server socket

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

4.2 Client Code

The client connects to the server and sends the file:

```
import socket
```

```

def send_file(server_ip, server_port, filename):
    # Create a socket object
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((server_ip, server_port))  # Connect to the server

    # Send the filename to the server
    client_socket.send(filename.encode())

    # Send the file content
    with open(filename, 'rb') as file:
        while (data := file.read(1024)):  # Read the file in chunks
            client_socket.send(data)

    print(f"File {filename} sent successfully.")
    client_socket.close()  # Close the client socket

if __name__ == "__main__":
    # Replace with the actual file name in the same directory
    server_ip = '192.168.162.200'  # Server's IP address
    server_port = 33333           # Server's port
    filename = 'example.txt'      # File to send

    send_file(server_ip, server_port, filename)
```

5 Results

The file transfer system was successfully tested. The following results were obtained:

- File transferred: `example.txt`
- File size: 58 B
- Transfer time: 1 second

6 Roles

Contributed to this project:

- Le Tuan Anh: Developed the server script (laptop 1).
- Le Tuan Anh: Developed the client script (laptop 2).
- Le Tuan Anh: Prepared the report in LaTeX.