Practical Work 1: TCP File Transfer

Group ID: 18

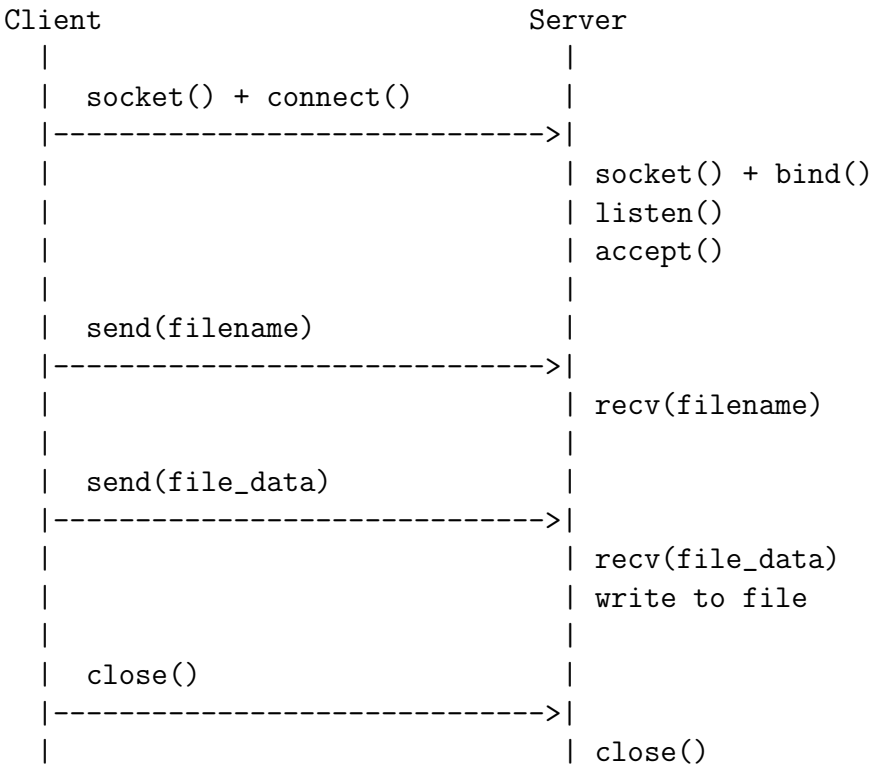
1 Protocol Design

The file transfer protocol is designed with the following simple steps:

1. **Client** creates a TCP connection to the **Server**
2. **Client** sends the filename (as UTF-8 string)
3. **Client** sends file content in chunks of 4096 bytes
4. **Server** receives and saves data to a new file
5. **Client** closes the connection after sending is complete

1.1 Sequence Diagram

System workflow:



2 System Organization

The system consists of 2 main components:

2.1 Server

- Listens for connections on address `127.0.0.1:65432`
- Accepts connections from clients
- Receives filename and data
- Saves file with `received_` prefix

2.2 Client

- Connects to the server
- Reads file from disk
- Sends filename and content
- Closes connection after completion

3 Implementation

3.1 Server Implementation

Key steps in the server:

```
# Tao socket TCP/IP
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Gan socket voi dia chi va port
server_socket.bind((HOST, PORT))

# Lang nghe ket noi
server_socket.listen(1)
```

Listing 1: Server - Create socket and bind

```
# Chap nhan ket noi
client_socket, client_address = server_socket.accept()

# Nhan ten file
filename = client_socket.recv(1024).decode('utf-8')

# Nhan du lieu file
with open(received_filename, 'wb') as f:
    while True:
        data = client_socket.recv(4096)
        if not data:
```

```
            break
        f.write(data)
```

Listing 2: Server - Accept and receive data

3.2 Client Implementation

Key steps in the client:

```
# Tao socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Ket noi den server
client_socket.connect((HOST, PORT))

# Gui ten file
client_socket.send(filename.encode('utf-8'))

# Gui noi dung file
with open(filename, 'rb') as f:
    while True:
        data = f.read(4096)
        if not data:
            break
        client_socket.send(data)
```

Listing 3: Client - Connect and send file

4 Who Does What

4.1 Server Responsibilities

- `socket()`: Creates a socket endpoint
- `bind()`: Associates the socket with a specific address
- `listen()`: Listens for incoming connections
- `accept()`: Accepts a connection from a client
- `recv()`: Receives data from the client
- `close()`: Closes the connection

4.2 Client Responsibilities

- `socket()`: Creates a socket endpoint
- `connect()`: Establishes a connection with the server
- `send()`: Sends data to the server
- `close()`: Closes the connection

5 Testing

To test the program:

1. Open the first terminal, run: `python server.py`
2. Open the second terminal, run: `python client.py`
3. Check that file `received_test_file.txt` has been created
4. Compare content with the original file `test_file.txt`