**University Of Science And Technology Of Hanoi**


# Distributed Systems

# Practical Work 1: TCP File transfer

Luong Quynh Nhi, 23BI14356, Cyber Security

Lecturer: Ms. Le Nhu Chu Hiep

## 1. Introduction

This practical work focuses on implementing a simple TCP-based file transfer system using a client–server architecture. The system allows a client to send a file to the server over a reliable TCP/IP connection.

## 2. Objectives

The main objectives of this practical are:

- Implement a working client–server architecture using TCP.
- Use fundamental socket functions for network communication.
- Organize system components clearly.
- Successfully transfer a text file from the client to the server.

## 3. System Architecture

The system uses a 1-to-1 TCP connection between a single server and a single client.

### 3.1 Server Responsibilities

- Start a TCP socket
- Configure socket using setsockopt()
- Bind to a specific IP and port
- Listen for incoming connections
- Accept a client
- Receiving file name and file data
- Writing received data to a local file.
- Sending acknowledgement messages.
- Closing the file and the connection.

### 3.2 Client Responsibilities

- Resolve server hostname using gethostbyname()
- Connect to server using TCP
- Send commands to the server
- Reading the file to be uploaded.
- Sending file name and file data
- Receiving acknowledgement messages.

- Closing the connection.

TCP File Transfer Protocol

Client Server

---------- Connect (TCP Handshake) ------->

---------- Request (UPLOAD/DOWNLOAD) ----->

---------- Filename ---------->

<---------- Status (OK / NOT FOUND) --------

(UPLOAD)

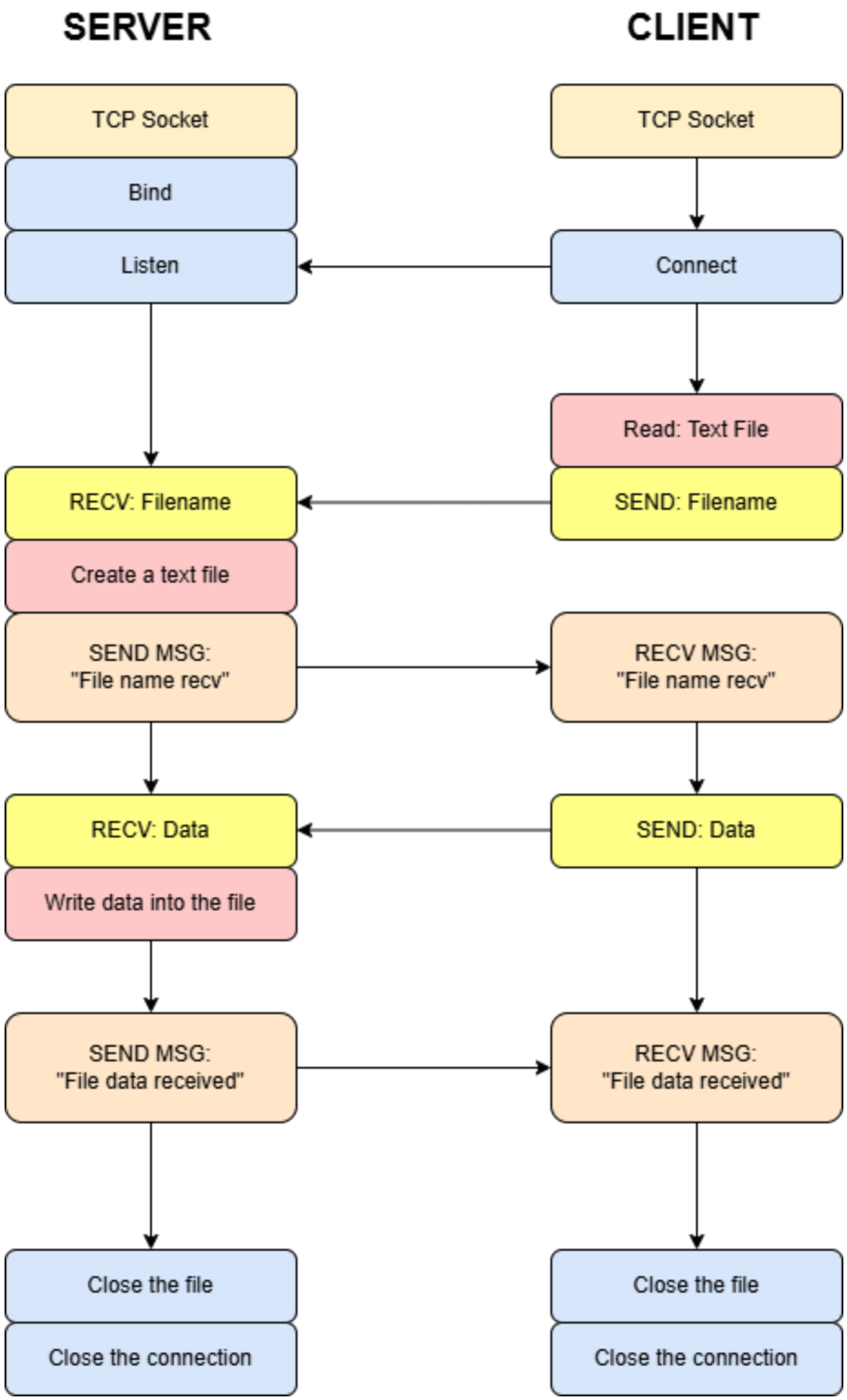
---------- File Data ---------->

(DOWNLOAD)

<---------- File Data ----------

---------- Close Connection ---------->

Client–Server Communication Diagram File Transfer




The diagram shows a linear protocol where:

- The client sends filename and file data

- The server responds with ACK1 ("File name recv") and ACK2 ("File data received")
- Both sides close the connection safely

This simple protocol ensures:

- The server knows the exact filename before receiving data.
- The client waits for confirmation before sending the next stage.
- File data is written only after proper synchronization.
- TCP guarantees reliable transmission.

## 4. Execution

Below is the code snippet for adding file function. I will execute the function of creating a New File on Server (addfile) as a demo for this practical.

- Client

```
send_text(sock, "addfile")

send_text(sock, filename)

sock.send(data.encode())
```

- Server

```
filename = recv_text(conn)

server_filename = "server_" + filename

open(server_filename, "wb").close()

conn.send(b"file recv\n")


data = conn.recv(4096)

with open(server_filename, "ab") as f:
```

```
    f.write(data)
conn.send(b"file data recv\n")
```

- Result:

```
Server listening on 127.0.0.1 port 80
Client connected: ('127.0.0.1', 52025)
Client command: addfile
Creating new file: 1412412
Sent: file recv
Receiving file data for: 1412412
Sent: file data recv
```

## 5. Conclusion

This practical demonstrates the core principles of TCP socket programming through a simple file transfer system. By designing a custom protocol, implementing a client - server architecture, and managing reliable data transmission, the project illustrates: How network connections are established; How data flows between client and server; How files can be transferred using a TCP stream.