

Practical Work 1: TCP File Transfer Report

Nguyen Thanh Dat

Student ID: 23BI14091

December 20, 2025

1 Introduction

The goal of this practical work is to implement a one-to-one file transfer system using TCP/IP sockets within a Command Line Interface (CLI). This report details the design and implementation of the protocol used to reliably transfer files (specifically tested with `test.jpg`) between a client and a server.

2 Protocol Design

To ensure that the server knows exactly what file is arriving and how large it is, I designed a simple application-layer protocol.

2.1 Protocol Logic

1. **Metadata Exchange:** Before sending the actual file content, the client sends a metadata header containing the `filename` and the `filesize` separated by a delimiter (`|`).
2. **Data Stream:** Once the server parses the size, the client streams the bytes. The server reads exactly the number of bytes specified in the header.

2.2 Protocol Sequence Diagram

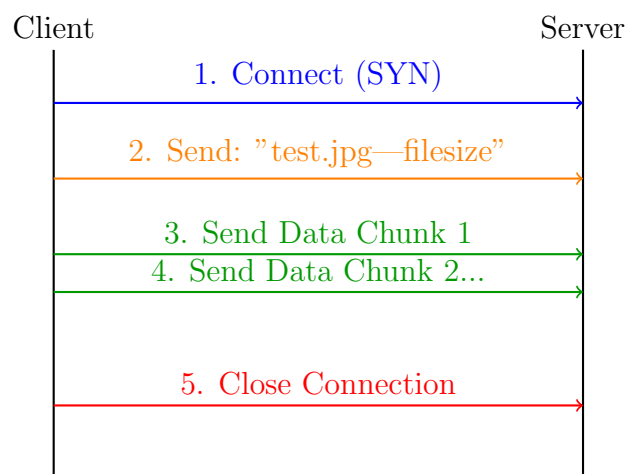


Figure 1: Sequence Diagram of the File Transfer Protocol

3 System Organization

The system consists of a single server and a single client. The server binds to a specific IP and Port and listens for incoming TCP requests. The client initiates the connection using a blocking socket.

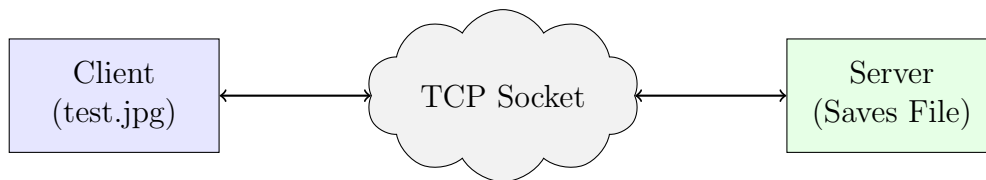


Figure 2: System Architecture

4 Implementation

The following is the core logic used to implement the file transfer.

4.1 Client Side

The client locates `test.jpg`, calculates its size, sends the header, and then streams the file content.

```
1 filename = "test.jpg"
2 filesize = os.path.getsize(filename)
3
4 s.connect((HOST, PORT))
5
6 # 1. Send Metadata
7 metadata = f"{filename}|{filesize}"
8 s.sendall(metadata.encode())
9
10 # 2. Send File Content
11 with open(filename, 'rb') as f:
12     while True:
13         bytes_read = f.read(1024)
14         if not bytes_read:
15             break
16         s.sendall(bytes_read)
```

4.2 Server Side

The server receives the metadata, splits them to get the file size, and loops until it has received the total expected bytes.

```
1 # 1. Receive Metadata
2 data = conn.recv(1024).decode()
3 filename, filesize = data.split('|')
4 filesize = int(filesize)
5
6 # 2. Receive File Data
```

```
7 received_bytes = 0
8 with open(filename, 'wb') as f:
9     while received_bytes < filesize:
10         chunk = conn.recv(1024)
11         if not chunk:
12             break
13         f.write(chunk)
14         received_bytes += len(chunk)
```