Practical Work 1: TCP File Transfer Report

Nguyen Thanh Dat
Student ID: 23BI14091

November 27, 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.

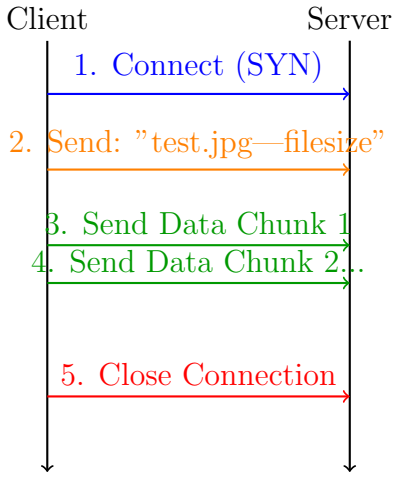
2 Protocol Design

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.

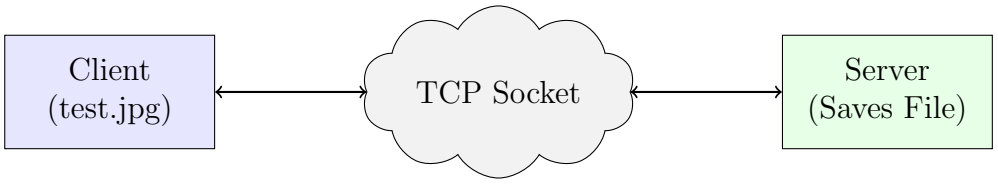
2.2 Protocol Sequence Diagram



figureSequence 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.



figureSystem 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.

```
filename = "test.jpg"
filesize = os.path.getsize(filename)

s.connect((HOST, PORT))

# 1. Send Metadata
metadata = f"{filename}|{filesize}"
s.sendall(metadata.encode())

# 2. Send File Content
with open(filename, 'rb') as f:
    while True:
        bytes_read = f.read(1024)
        if not bytes_read:
            break
        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. Receive Metadata
data = conn.recv(1024).decode()
filename, filesize = data.split('|')
filesize = int(filesize)

# 2. Receive File Data
received_bytes = 0
with open(filename, 'wb') as f:
    while received_bytes < filesize:
        chunk = conn.recv(1024)
```

```
11          if not chunk:
12              break
13          f.write(chunk)
14          received_bytes += len(chunk)
```