Department of Information and Communication Technology

Data Science

Distribution System

Practical Work 1: TCPFiletransfer

Members:
22BI13472 - Nguyen Ba Vinh
22BI13220 - Nguyen Minh Khoi
22BI13227 - Tran Trung Kien
22BI13229 - Dinh Tuan Kiet
BA12-033 - Nguyen Khac Cong

Supervisor:
Huynh Vinh Nam

Academic year: 2022 - 2025
Hanoi, January 2024

Table of Content

1 Introduction

In the modern era of digital communication, file transfer systems play a critical role in enabling seamless sharing of data between devices and networks. These systems underpin numerous applications, ranging from cloud storage services to enterprise collaboration tools, making their design and implementation a fundamental area of study in computer science.

This report presents the development of a **Client-Server File Transfer System**, designed to facilitate the reliable upload and download of files over a network. Using Python and socket programming, the system establishes a TCP-based connection between the client and server, allowing real-time file exchange while ensuring data integrity.

The project focuses on key aspects of file transfer systems:

- **Protocol Design**: A simple yet effective communication protocol is developed to handle user requests, such as uploading and downloading files, ensuring clarity and ease of use.
- **System Architecture**: The system is structured with distinct roles for the client (initiating requests) and the server (processing and responding to requests), enabling scalability and modularity.
- **Data Transmission**: Files are transferred in chunks to accommodate large file sizes and ensure efficient resource utilization.
- **Error Handling**: Mechanisms are integrated to manage potential issues, such as missing files or network interruptions, providing a robust user experience.

The system's implementation is evaluated through practical testing, demonstrating its functionality and performance under various scenarios. This report details the design, organization, and coding aspects of the project, highlighting challenges encountered and solutions devised during development.

By building this system, the project showcases the practical application of network programming concepts and offers insights into the design principles of reliable file transfer protocols. This work serves as a foundation for future enhancements, such as incorporating encryption, authentication, or advanced concurrency management.

2 How You Design Your Protocol

Explanation
The protocol defines how the client and server communicate. It typically includes the following components:

- **Request Format**: Specifies the format of requests sent by the client to the server. For example, `UPLOAD filename` or `DOWNLOAD filename`.
- **Error Handling**: Ensures the system can handle potential issues, such as missing files (`File not found`) or invalid commands.
- **Data Transfer**: Defines the method for transferring files, such as splitting files into chunks for efficient transmission and managing acknowledgment of received data.

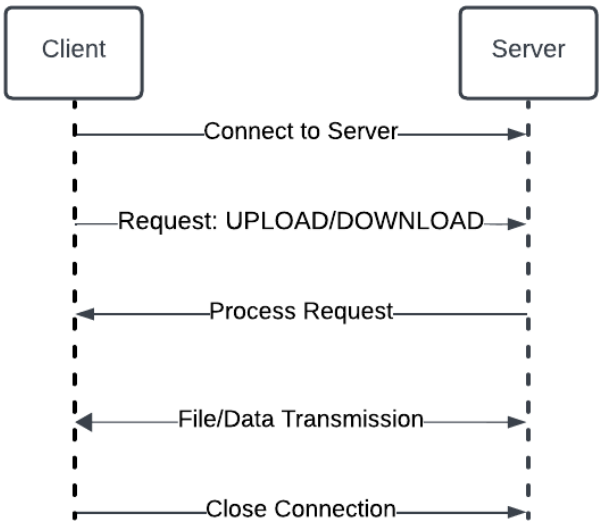

Figura 1: Protocol Flow Diagram

3 System Organization

3.1 System Components:

Server

- Listens on a socket.
- Handles `UPLOAD` and `DOWNLOAD` commands.
- Reads/Writes files using binary mode for efficient transfers.

Client

- Sends commands to the server.
- Uploads or downloads files based on the server's response.
- Handles file reading/writing locally.

3.2 System Organization

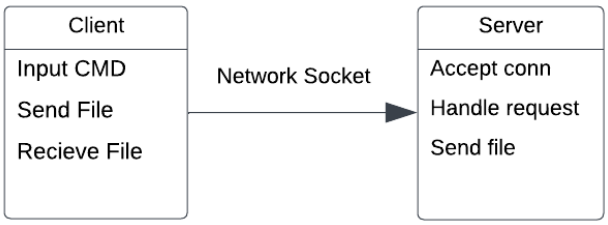

Figura 2: Protocol Flow Diagram

4 File Transfer Implementation

Client Side (UPLOAD File)

```
if req.startswith("UPLOAD"):
    try:
        with open(filename, "rb") as file:
            print(f"Client Uploading {filename}")
            while chunk := file.read(1024):  # Read file in
                chunks
                clientSocket.send(chunk)    # Send chunk to
                    server
            print(f"{filename} is uploaded by client")
    except FileNotFoundError:
        print("File not found")
```

```
if command.startswith("UPLOAD"):
    filename = command.split()[1]
    with open(filename, "wb") as file:
        while True:
            data = conn.recv(1024)          # Receive data in
               chunks
            if not data: break             # Stop if no data
            file.write(data)               # Write chunk to
               file
        print(f"{filename} is received by server!")
```

Server Side (DOWNLOAD File)

```
if req.startswith("DOWNLOAD"):
    with open(filename, "wb") as file:
        print(f"Client Downloading {filename}")
        while True:
            data = clientSocket.recv(1024) # Receive chunks
            if not data: break             # Stop if no data
            file.write(data)               # Write to file
        print(f"Client successfully downloaded {filename}")
```

```
if command.startswith("DOWNLOAD"):
    filename = command.split()[1]
    try:
        with open(filename, "rb") as file:
            print(f"Server sending {filename}")
            while chunk := file.read(1024): # Read chunks
                conn.send(chunk)            # Send chunk to
                   client
            print(f"Server successfully sent {filename}")
    except FileNotFoundError:
        conn.send(b"Error: File not found")
```