TCP File Transfer System: Architecture and Implementation

Technical Documentation

November 25, 2024

1 Introduction

The system implements a client-server architecture for file transfer over TCP/IP. It consists of three main components:

$$\text{Components} = \{\text{Client}, \text{Server}, \text{Script Interface}\}$$

2 System Architecture

The system follows a traditional client-server architecture with the following mathematical representation:

$$\begin{aligned}
S &= \text{Server Process} \\
C &= \text{Client Process} \\
F &= \text{File Data} \\
P &= \text{Protocol Operations} = \{\text{SEND}, \text{REQUEST}, \text{LIST}\}
\end{aligned}$$

2.1 Communication Protocol

The protocol follows a state machine model where each operation $p \in P$ transitions through states:

$$\begin{aligned}
&\text{State}_0 \xrightarrow{\text{Connection}} \text{State}_1 \\
&\text{State}_1 \xrightarrow{\text{Operation Selection}} \text{State}_2 \\
&\text{State}_2 \xrightarrow{\text{Data Transfer}} \text{State}_3 \\
&\text{State}_3 \xrightarrow{\text{Completion}} \text{State}_0
\end{aligned}$$

3 Key Components

3.1 FileTransferServer

The server implements a listening socket with the following configuration:

$$\text{Server Socket} = \begin{cases} \text{Host:} & 0.0.0.0 \\ \text{Port:} & 12345 \\ \text{Backlog:} & 1 \\ \text{Protocol:} & \text{TCP} \end{cases}$$

3.2 FileTransferClient

The client implements three main operations:

$$\text{Client Operations} = \begin{cases} \text{send\_file}(f) & : \text{Upload file } f \text{ to server} \\ \text{request\_file}(f) & : \text{Download file } f \text{ from server} \\ \text{list\_files}() & : \text{Get server file listing} \end{cases}$$

3.3 Data Transfer Protocol

For file transfers, the protocol follows this sequence:

$$\begin{aligned} \text{Step 1} &: \text{ Initialize connection} \\ \text{Step 2} &: \text{ Send operation type } p \in P \\ \text{Step 3} &: \text{ Exchange metadata (filename, size)} \\ \text{Step 4} &: \text{ Transfer data in chunks of 4096 bytes} \\ \text{Step 5} &: \text{ Verify completion} \end{aligned}$$

The progress of file transfer is calculated as:

$$\text{Progress}(\%) = \frac{\text{Bytes Transferred}}{\text{Total Bytes}} \times 100$$

4 Implementation Details

4.1 Buffer Sizes

The system uses optimized buffer sizes:

$$\begin{aligned} \text{Control Messages} &: 1024 \text{ bytes} \\ \text{Data Transfer} &: 4096 \text{ bytes} \end{aligned}$$

4.2 Error Handling

The system implements comprehensive error handling with the following categories:

$$\text{Errors} = \begin{cases} \text{Connection Failures} \\ \text{File Not Found} \\ \text{Transfer Interruption} \\ \text{Invalid Operations} \end{cases}$$

5 Usage Examples

5.1 Server Mode

To start the server:

$$\text{python script.py –mode server –port 12345}$$

5.2 Client Mode

For client operations:

$$\begin{aligned} \text{Send}: &\ \text{python script.py –mode client –sor send –fn file.txt} \\ \text{Receive}: &\ \text{python script.py –mode client –sor receive –fn file.txt} \\ \text{List}: &\ \text{python script.py –mode client –sor list} \end{aligned}$$

6 Performance Characteristics

The system's performance can be characterized by:

$$\text{Transfer Time} = \frac{\text{File Size}}{\text{Network Bandwidth}} + \text{Protocol Overhead}$$

Where protocol overhead includes:

$$\text{Overhead} = \text{Connection Setup} + \text{Metadata Exchange} + \text{Acknowledgments}$$

7 Code Structure

7.1 Class Hierarchy

The system is organized into two main classes with their interfaces:

$$\text{FileTransferServer} \rightarrow \{\text{run}, \text{handle\_file\_transfer}, \text{send\_file}, \text{receive\_file}\}$$
$$\text{FileTransferClient} \rightarrow \{\text{send\_file}, \text{request\_file}, \text{list\_files}\}$$

7.2 Protocol Flow

The protocol flow can be represented as a sequence:

$$\begin{pmatrix} \text{Client} & \xrightarrow{\text{Connect}} & \text{Server} \\ \text{Client} & \xrightarrow{\text{Operation}} & \text{Server} \\ \text{Server} & \xrightarrow{\text{ACK}} & \text{Client} \\ \text{Client/Server} & \xrightarrow{\text{Data}} & \text{Server/Client} \end{pmatrix}$$