File Transfer over TCP/IP using Sockets

Lab Report

Nguyen Manh Khoi
22BI13219

November 29, 2024

1 Introduction

File transfer protocols are important in modern networking. The goal of this lab is to create a simple file transfer system using the Transmission Control Protocol (TCP). Using Python's socket library, we built a client-server application where the client sends a file to the server over a TCP/IP connection.

2 Objectives

The objectives of this lab were:

- Implement a file transfer system using TCP sockets.
- Understand client-server communication using socket programming.
- Explore how data is transferred in chunks over a network.

3 Methodology

In this lab, the client-server architecture was implemented for file transfer over TCP/IP. The methodology was as follows:

1. The **server** was created to listen for incoming client connections on a specific port.
2. Once a client connected, the **server** received the file sent by the client in chunks, ensuring that large files could be handled efficiently.
3. The **client** connected to the server, sent the filename, and then transferred the file data in 1024-byte chunks.
4. The **server** saved the file to disk once the entire file had been received.
5. Both the client and server closed the connection once the transfer was completed.

4 System Design

The system has two main components: the **server** and the **client**. The server listens on a predefined port, accepts incoming connections, and receives files from the client. The client connects to the server, sends the file, and closes the connection once the transfer is complete.

4.1 File Transfer Protocol

The file transfer protocol follows a simple sequence:

1. The client sends the filename to the server.
2. The client then sends the file in 1024-byte chunks.
3. The server receives each chunk and writes it to a file on the disk.
4. Once the file transfer is complete, the server confirms the transfer, and both the client and server close the connection.

5 Implementation

The implementation of the file transfer system was done using Python's 'socket' library. The code for the server and client is provided below.

5.1 Server Code

Listing 1: Server Code

```
import socket

HOST = '172.19.219.4'  # Localhost
PORT = 12344           # Port to listen on

# Create a socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind((HOST, PORT))

# Start listening
server_socket.listen(1)
print(f"Server listening on {HOST}:{PORT}...")

# Connection from client
conn, addr = server_socket.accept()
print(f"Connected by {addr}")

# Receive the file
file_name = conn.recv(1024).decode('utf-8')  # Receive file name
print(f"Receiving file: {file_name}")
```

```
with open(file_name, 'wb') as f:
    while True:

        data = conn.recv(1024)
        if not data:
            break
        f.write(data)

print(f"File {file_name} received successfully.")
conn.close()  # Close
server_socket.close()
```

5.2 Client Code

Listing 2: Client Code

```
import socket

def client_program():
    # server and client must have the same IP and port
    host = '172.19.219.4'
    port = 12344  # Port

    # TCP socket
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # Connect to server
    client_socket.connect((host, port))

    # File
    filename = 'received_transfer_file.txt'  # Name the the file

    # Try to receive the file from the server
    with open(filename, 'wb') as f:
        print(f"Receiving file from the server...")

        while True:
            data = client_socket.recv(1024)
            if not data:
                print("No more data received, closing connection.")
                break  # No more data received, end of file transfer
            f.write(data)  # Write data to the file

    print(f"File received successfully! Saved as {filename}")

    # Close socket connection
    client_socket.close()
```

```
if __name__ == '__main__':
    client_program()
```

5.3 Sample Output from the Client

```
Server listening on 172.19.219.4:12344...
Connected by ('172.19.208.1', 59923)
Receiving file: example.txt
File example.txt received successfully.
```

The file ‘example.txt‘ was successfully transferred from the client to the server, and the server saved it in the current directory.

6 Discussion

This lab demonstrated the basic principles of file transfer over a TCP/IP connection using socket programming. The system successfully transferred files by breaking them into smaller chunks to handle larger files.

Additionally, the client-server system could be upgraded to handle multiple file transfers simultaneously.

7 Conclusion

This lab shown how to use TCP/IP sockets to transfer files between a client and a server. By following the basic principles of socket programming, we were able to successfully implement a file transfer system that efficiently sends and receives files.