UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI

DISTRIBUTED SYSTEMS

PRACTICAL WORK 1 TCP FILE TRANSFER

TRAN DUC HUY
BA12-085
huytd.ba12-085@st.usth,edu.vn

CYBER SECURITY

Lecturer:	MS. Le Nhu Chu Hiep
Department:	Information and Technology
University:	Science and Technology of Hanoi

Hanoi, 11/2024

Contents

1 Introduction

In this practical work, the goal was to implement a file transfer system using TCP/IP based on a client-server architecture. The system consists of one server and one client, which communicate via sockets.

2 Protocol design

2.1 Overview of the protocol

The File Transfer Protocol (FTP) is designed to ensure a reliable, efficient, and secure transfer of files between a client and a server over a TCP/IP network. The protocol uses two separate connections: one for control commands (control connection) and one for transferring file data (data connection).

2.2 Steps in protocol design

The protocol for file transfer is structured as follows:

1. **Connection establishment:** The client connects to the server through a predefined IP and port; the server listens for and accepts the connection.
2. **File metadata:** The client sends the file name and size to the server; the server acknowledges that it is ready to receive the file.
3. **File transfer:** The client sends the file in chunks; the server receives and writes the data, sending an acknowledgment ("ACK") for each chunk received.
4. **Acknowledgment:** After each chunk is received, the server sends an acknowledgment back to the client to confirm successful receipt.
5. **Completion:** Once all chunks are transferred, the server signals the completion to the client.
6. **Connection closure:** Both the client and the server close the control and data connections after the transfer is complete.

2.3 Figure design protocol

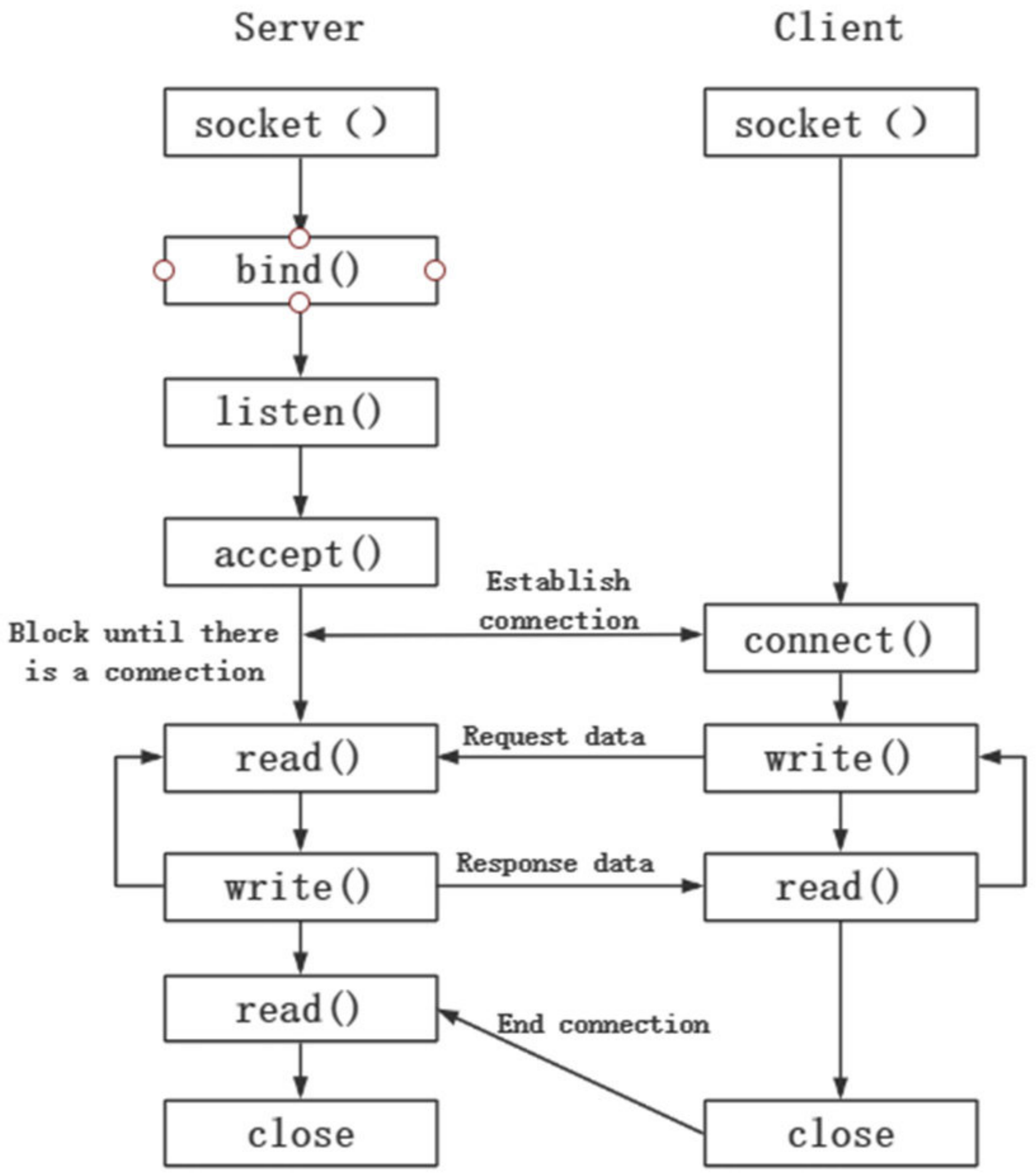


Figure 1: FTP design

3 Systems organization

3.1 System architecture

The system is built around a "Client-Server" architecture, where the "Client" initiates a connection and transfers a file to the "Server". The two main components of the system are the "Client" and the "Server", which communicate through a TCP connection. The following is a detailed breakdown of the system architecture:

- Client:
  - The client initiates the file transfer by establishing a TCP connection to the server.
  - It first sends the file metadata (name and size), followed by the file data in chunks.
  - The client waits for an acknowledgment (ACK) from the server after each chunk to ensure reliable data transfer.
  - Once the entire file has been sent successfully, the client closes the connection.
- Server:
  - The server listens for incoming connections on a pre-defined port.
  - Once the connection is established, it receives the file metadata and prepares to receive the file.
  - The server receives the data from the file in chunks and writes the data to a file.
  - After receiving each chunk, the server sends an acknowledgment (ACK) back to the client.
  - When the file is fully received, the server signals the completion of the transfer and closes the connection.

3.2 Figure system organization

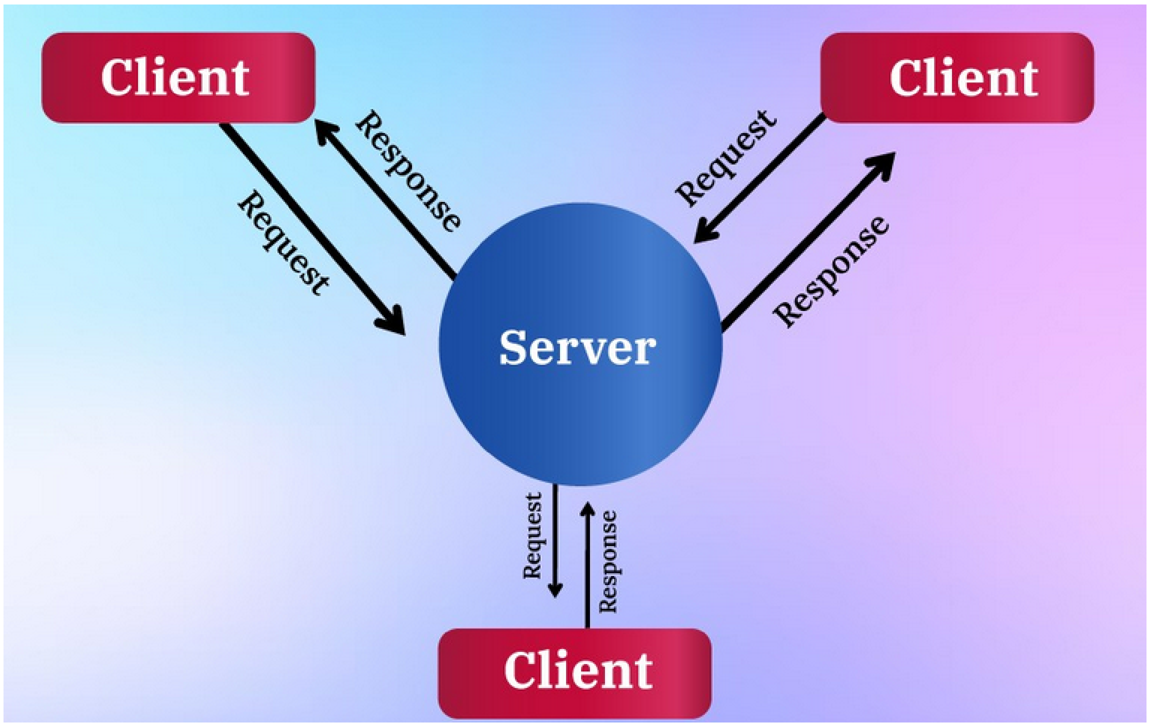


Figure 2: System organization

4 File transfer implementation

4.1 Client code

The following code snippet shows the implementation of the client in Python using TCP/IP for file transfer. The client connects to the server, sends the file metadata (name and size), and then transfers the file data in chunks. After sending each chunk, the client waits for an acknowledgment from the server before sending the next chunk.

```
import socket
import os

def send_file():
    server_ip = '127.0.0.1'
    server_port = 21003
    filename = 'send.txt'

    try:
        with open(filename, 'rb') as file:
            client_socket = socket.socket(socket.AF_INET, socket.SO
            client_socket.connect((server_ip, server_port))

            file_name = os.path.basename(filename)
            file_size = os.path.getsize(filename)
            client_socket.send(f"{file_name},{file_size}".encode())

            ack = client_socket.recv(1024)
            print(f"Server acknowledgment: {ack.decode()}")

            while chunk := file.read(1024):
                client_socket.send(chunk)
                ack = client_socket.recv(1024)
                print(f"Acknowledgment received: {ack.decode()}")

            print(f"File '{filename}' has been sent successfully")

    except Exception as e:
        print(f"Error: {e}")

    finally:
        client_socket.close()

if __name__ == "__main__":
    send_file()
```

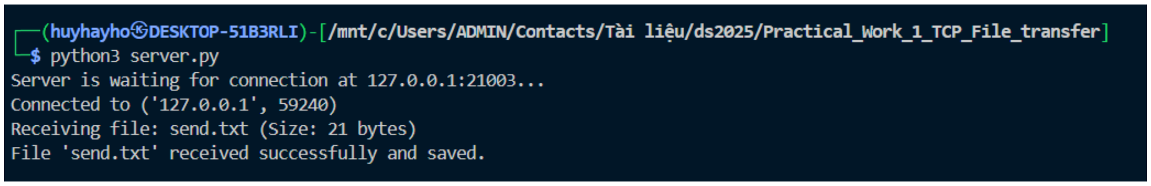
4.2 Server code

The following code snippet shows the implementation of the server in Python using TCP/IP for file transfer. The server listens for incoming connections from the client, receives the file metadata (name and size), and then writes the received file data to disk in chunks. After receiving each chunk, the server sends an acknowledgment to the client.

Listing 1: Server code for file transfer using TCP/IP

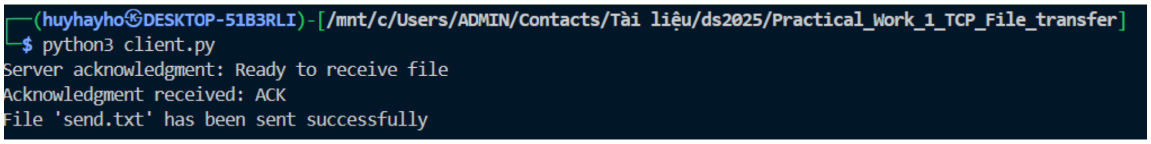
```
import socket

def receive_file():
    server_ip = '0.0.0.0'
    server_port = 21003
    allowed_ips = ['127.0.0.1', '192.168.48.140', '172.30.176.1']

    try:
        server_socket = socket.socket(socket.AF_INET, socket.SOCK_S
        server_socket.bind((server_ip, server_port))
        server_socket.listen(5)

        print(f"Server is waiting for connection at {server_ip}:{se

        while True:
            client_socket, client_address = server_socket.accept()
            print(f"Connected to {client_address}")

            if client_address[0] not in allowed_ips:
                print(f"Connection from {client_address[0]} is not
                client_socket.close()
                continue

            file_info = client_socket.recv(1024).decode()
            filename, file_size = file_info.split(',')
            file_size = int(file_size)
            print(f"Receiving file: {filename} (Size: {file_size} b

            client_socket.send("Ready to receive file".encode())

            with open(f'received_{filename}', 'wb') as file:
                bytes_received = 0
                while bytes_received < file_size:
                    data = client_socket.recv(1024)
                    file.write(data)
                    bytes_received += len(data)

                    client_socket.send("ACK".encode())

            print(f"File '{filename}' received successfully and sav

            client_socket.close()

    except Exception as e:
        print(f"Error: {e}")

    finally:
        server_socket.close()

if __name__ == "__main__":
    receive_file()
```

4.3 Testing code snippet with myself



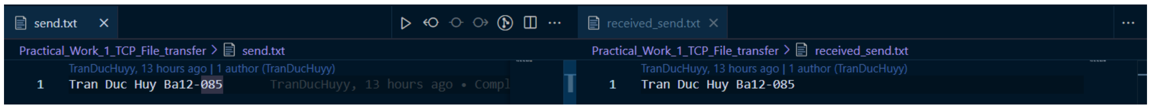
```
┌──(huyhayho㉿DESKTOP-51B3RLI)-[/mnt/c/Users/ADMIN/Contacts/Tài liệu/ds2025/Practical_Work_1_TCP_File_transfer]
└─$ python3 server.py
Server is waiting for connection at 127.0.0.1:21003...
Connected to ('127.0.0.1', 59240)
Receiving file: send.txt (Size: 21 bytes)
File 'send.txt' received successfully and saved.
```

Figure 3: Server command line



```
┌──(huyhayho㉿DESKTOP-51B3RLI)-[/mnt/c/Users/ADMIN/Contacts/Tài liệu/ds2025/Practical_Work_1_TCP_File_transfer]
└─$ python3 client.py
Server acknowledgment: Ready to receive file
Acknowledgment received: ACK
File 'send.txt' has been sent successfully
```

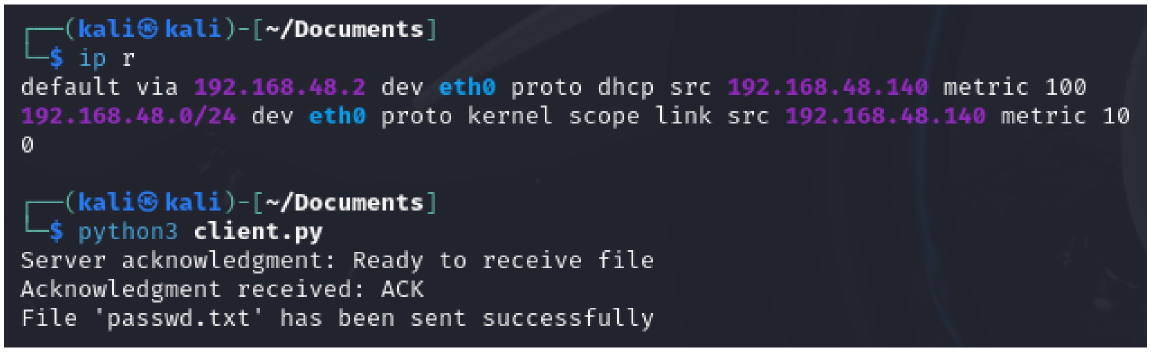
Figure 4: Client command line



```
send.txt
Practical_Work_1_TCP_File_transfer > send.txt
TranDucHuyy, 13 hours ago | 1 author (TranDucHuyy)
Tran Duc Huy Ba12-085        TranDucHuyy, 13 hours ago • Compl

received_send.txt
Practical_Work_1_TCP_File_transfer > received_send.txt
TranDucHuyy, 13 hours ago | 1 author (TranDucHuyy)
Tran Duc Huy Ba12-085
```

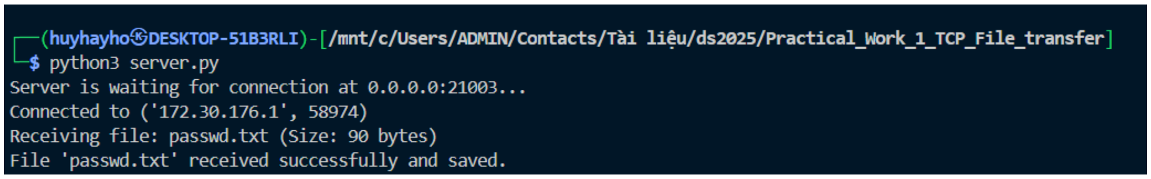
Figure 5: Compare send.txt and received$_send.txt$

4.4 Testing code with groupmate



```
┌──(kali㉿kali)-[~/Documents]
└─$ ip r
default via 192.168.48.2 dev eth0 proto dhcp src 192.168.48.140 metric 100
192.168.48.0/24 dev eth0 proto kernel scope link src 192.168.48.140 metric 10
0

┌──(kali㉿kali)-[~/Documents]
└─$ python3 client.py
Server acknowledgment: Ready to receive file
Acknowledgment received: ACK
File 'passwd.txt' has been sent successfully
```

Figure 6: Client command line - crab241 computer



```
┌──(huyhayho㉿DESKTOP-51B3RLI)-[/mnt/c/Users/ADMIN/Contacts/Tài liệu/ds2025/Practical_Work_1_TCP_File_transfer]
└─$ python3 server.py
Server is waiting for connection at 0.0.0.0:21003...
Connected to ('172.30.176.1', 58974)
Receiving file: passwd.txt (Size: 90 bytes)
File 'passwd.txt' received successfully and saved.
```

Figure 7: Server command line - my computer

5 Conclusion

In this project, we successfully implemented a file transfer system using the TCP/IP protocol, following a client-server architecture. The "Client" and "Server" components communicated efficiently through a socket connection, where the client sent a file in chunks and the server received the chunks and stored the file.

The key features of the system include:

- Reliable file transfer with acknowledgment after each chunk.
- Handling of file metadata, such as the file's name and size, before the transfer begins.
- A clear separation of responsibilities between the client (sending the file) and the server (receiving and saving the file).
- Simple error handling to handle connection issues and ensure robust transfer.

In conclusion, this project provided a solid foundation for understanding how to implement a basic file transfer system using TCP/IP sockets, with practical insights into how such systems function and can be enhanced for real-world use.

6 References

References

[1] Mắt Bão. (n.d.). *Socket là gì? Khái niệm cần biết về giao thức TCP/IP và UDP*. Retrieved from `https://wiki.matbao.net/socket-la-gi-khai-niem-can-biet-ve-giao-thuc-tcp-ip-va-udp/`