# TCP File Transfer Practical Work 1

Ha Tan Minh

## 1 Introduction

The purpose of this practical is to implement a simple file transfer system using TCP/IP. The work involves socket programming in Python, with one laptop acting as a server and the other as a client.

## 2 Protocol Design

The file transfer protocol involves:

- Establishing a connection between the client and the server.
- Sending the file name from the client to the server.
- Transferring the file in chunks of data.
- The server saving the file after all data is received.

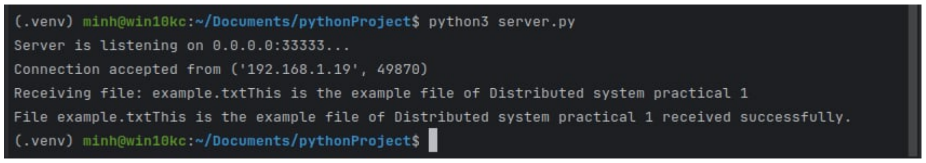
## 3 System Interaction

Laptop 1 (Server)

Laptop 2 (Client)

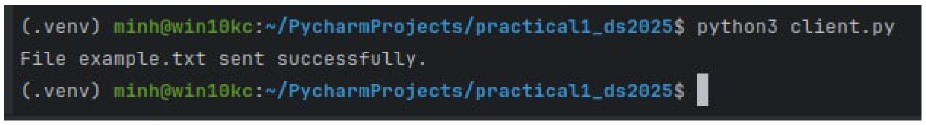
## 4 Implementation

The implementation involves two Python scripts:



```
(.venv) minh@win10kc:~/Documents/pythonProject$ python3 server.py
Server is listening on 0.0.0.0:33333...
Connection accepted from ('192.168.1.19', 49870)
Receiving file: example.txtThis is the example file of Distributed system practical 1
File example.txtThis is the example file of Distributed system practical 1 received successfully.
(.venv) minh@win10kc:~/Documents/pythonProject$
```

Figure 1: Enter Caption



```
(.venv) minh@win10kc:~/PycharmProjects/practical1_ds2025$ python3 client.py
File example.txt sent successfully.
(.venv) minh@win10kc:~/PycharmProjects/practical1_ds2025$
```

Figure 2: Enter Caption

## 4.1 Server Code

The server listens for incoming connections and saves the file:

```
import socket

def start_server():
    server_ip = '0.0.0.0'  # Listen on all available interfaces
    server_port = 33333    # Port to listen on

    # Create a socket object
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((server_ip, server_port))  # Bind to the specified IP and port
    server_socket.listen(1)  # Allow one client connection at a time
    print(f"Server is listening on {server_ip}:{server_port}...")

    # Accept a connection from the client
    conn, addr = server_socket.accept()
    print(f"Connection accepted from {addr}")

    # Receive the filename from the client
    filename = conn.recv(1024).decode()
    print(f"Receiving file: {filename}")

    # Receive the file content and save it
    with open(filename, 'wb') as file:
        while True:
            data = conn.recv(1024)  # Receive data in chunks
            if not data:
                break
            file.write(data)

    print(f"File {filename} received successfully.")
    conn.close()  # Close the connection
    server_socket.close()  # Close the server socket

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

## 4.2 Client Code

The client connects to the server and sends the file:

```
import socket

def send_file(server_ip, server_port, filename):
    # Create a socket object
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((server_ip, server_port))  # Connect to the server

    # Send the filename to the server
    client_socket.send(filename.encode())

    # Send the file content
    with open(filename, 'rb') as file:
        while (data := file.read(1024)):  # Read the file in chunks
            client_socket.send(data)

    print(f"File {filename} sent successfully.")
    client_socket.close()  # Close the client socket

if __name__ == "__main__":
    # Replace with the actual file name in the same directory
    server_ip = '192.168.1.20'  # Server's IP address
    server_port = 33333           # Server's port
    filename = 'example.txt'      # File to send

    send_file(server_ip, server_port, filename)
```

## 5 Results

The file transfer system was successfully tested. The following results were obtained:

- File transferred: ‘example.txt‘
- File size: 58 B
- Transfer time: 1 second

## 6 Roles

Contributed to this project:

- Ha Tan Minh: Developed the server script (laptop 1).
- Ha Tan Minh: Developed the client script (laptop 2).
- Ha Tan Minh: Prepared the report in LaTeX.