CSC203L Computer Networks Lab

Submitted by:

Umair Arshad 2024-SE-38

Submitted to:

Prof. Noman Munir

Dated: November 21, 2025

Department of Computer Science

University of Engineering and Technology, New Campus

SOCKET PROGRAMMING HOME TASKS UDP File Sharing

1. Home Task Problem for UDP in file sharing:

You are required to **implement a UDP-based file transfer application** that includes a **ping-pong mechanism** and **Round-Trip Time (RTT)** calculation. The **client** will first send a “ping” message to the **server** to check the connection status. The **server** will reply with a “pong” message, confirming readiness to receive data. The **client** will then calculate the **RTT** based on the time taken between sending “ping” and receiving “pong.” After confirming connectivity, the client will start sending the file to the server in **UDP packets**. The **server** will receive the file, store it locally, and send a **confirmation message** back to the client upon successful transfer. Both the client and server will include **basic error handling** to detect and handle lost packets or transmission errors, ensuring reliable file transfer even though UDP is a connectionless protocol.

2. SOLUTION:

2.1 SERVERSIDE SETUP:

Step 1:

Import socket and OS libraries essential for file sharing.

Step 2:

Assign server with IP address and Port number with buffer capacity.

Step 3:

Make the server-side socket.

Step 4:

bind the server-side socket.

Step 5:

Receive the client’s address with message.

Step 6:

If client sends “ping” server automatically send it “pong” and if client says START <filename>.txt extract filename otherwise print message “unknown message from client”.

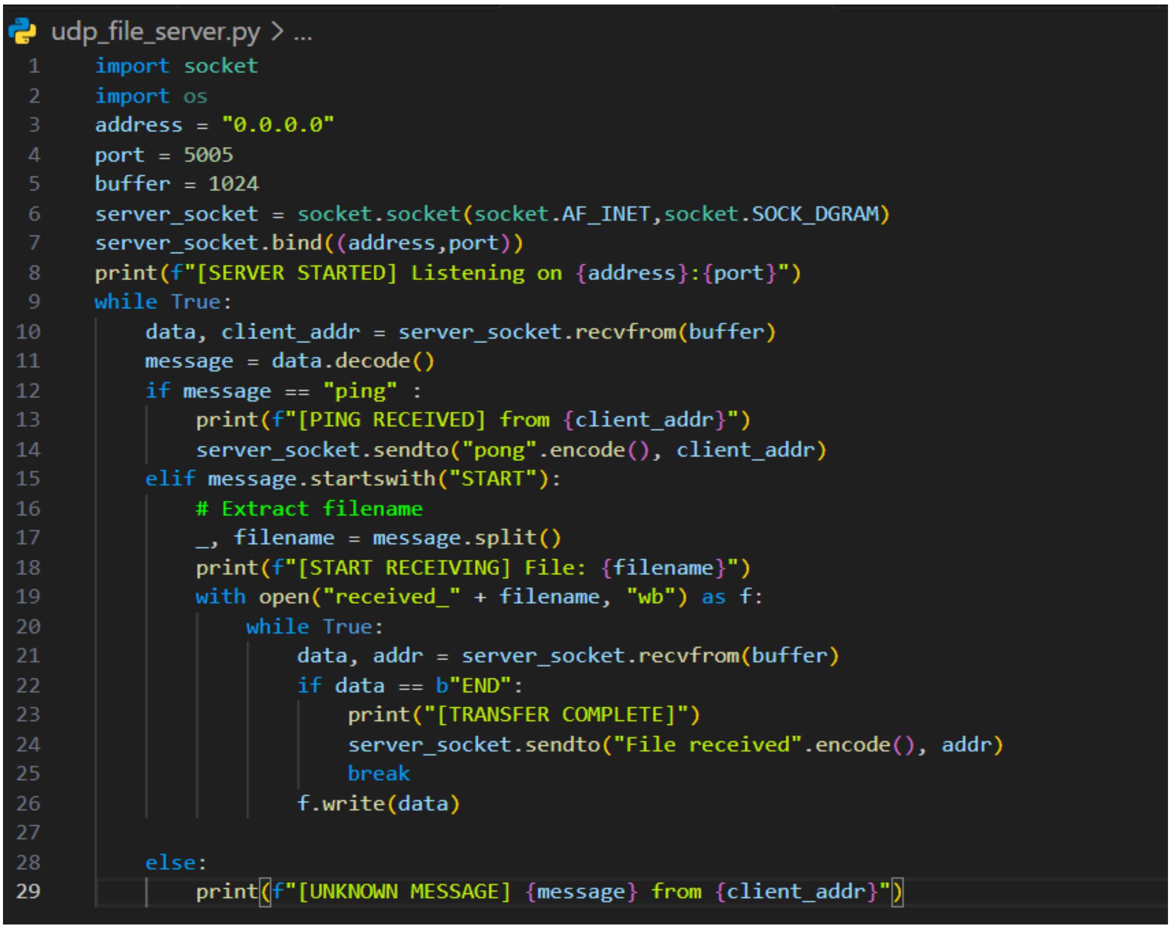
Step 7:

Server creates new file named Received_<filename>.txt where it opens the file and receives data inside the file shared by client in chunks and writes it into the new file in server side.

Step 8:

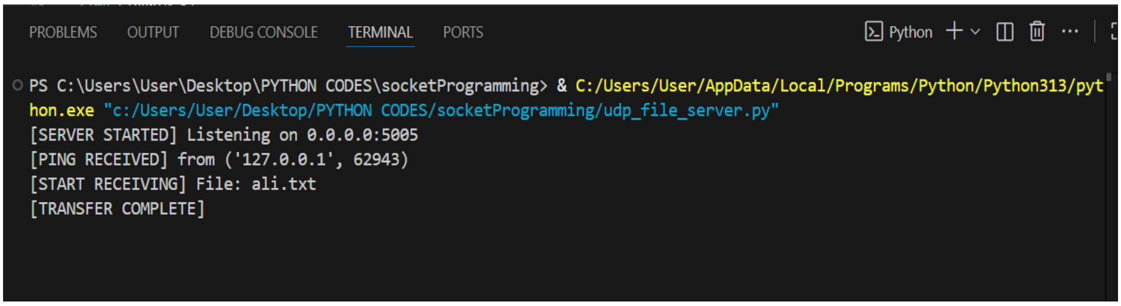
After that tell the client that file content is transferred successfully!

2.2 PYTHON CODE SNIPPET:



```python
import socket
import os
address = "0.0.0.0"
port = 5005
buffer = 1024
server_socket = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
server_socket.bind((address,port))
print(f"[SERVER STARTED] Listening on {address}:{port}")
while True:
    data, client_addr = server_socket.recvfrom(buffer)
    message = data.decode()
    if message == "ping" :
        print(f"[PING RECEIVED] from {client_addr}")
        server_socket.sendto("pong".encode(), client_addr)
    elif message.startswith("START"):
        # Extract filename
        _, filename = message.split()
        print(f"[START RECEIVING] File: {filename}")
        with open("received_" + filename, "wb") as f:
            while True:
                data, addr = server_socket.recvfrom(buffer)
                if data == b"END":
                    print("[TRANSFER COMPLETE]")
                    server_socket.sendto("File received".encode(), addr)
                    break
                f.write(data)

    else:
        print(f"[UNKNOWN MESSAGE] {message} from {client_addr}")
```

2.3 OUTPUT



```
PS C:\Users\User\Desktop\PYTHON CODES\socketProgramming> & C:/Users/User/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/User/Desktop/PYTHON CODES/socketProgramming/udp_file_server.py"
[SERVER STARTED] Listening on 0.0.0.0:5005
[PING RECEIVED] from ('127.0.0.1', 62943)
[START RECEIVING] File: ali.txt
[TRANSFER COMPLETE]
```

3. CLIENTSIDE SETUP:

Step 1:

Import socket, time and OS libraries essential for file sharing.

Step 2:

Use server IP address and port number as it is.

Step 3:

Make the client-side socket.

Step 4:

Start the time and send “ping” message to server.

Step 5:

Receive the server message and checks whether it says “pong” or not. If says “pong”, the client calculates RTT and prints it. Then client is asked to enter file name to share its content.

Step 6:

Client will write file name and checks if file on client side exists.

Step 7:

If exists, the data inside the file is read and sent in the form of chunks to server side until it reaches to “END”.

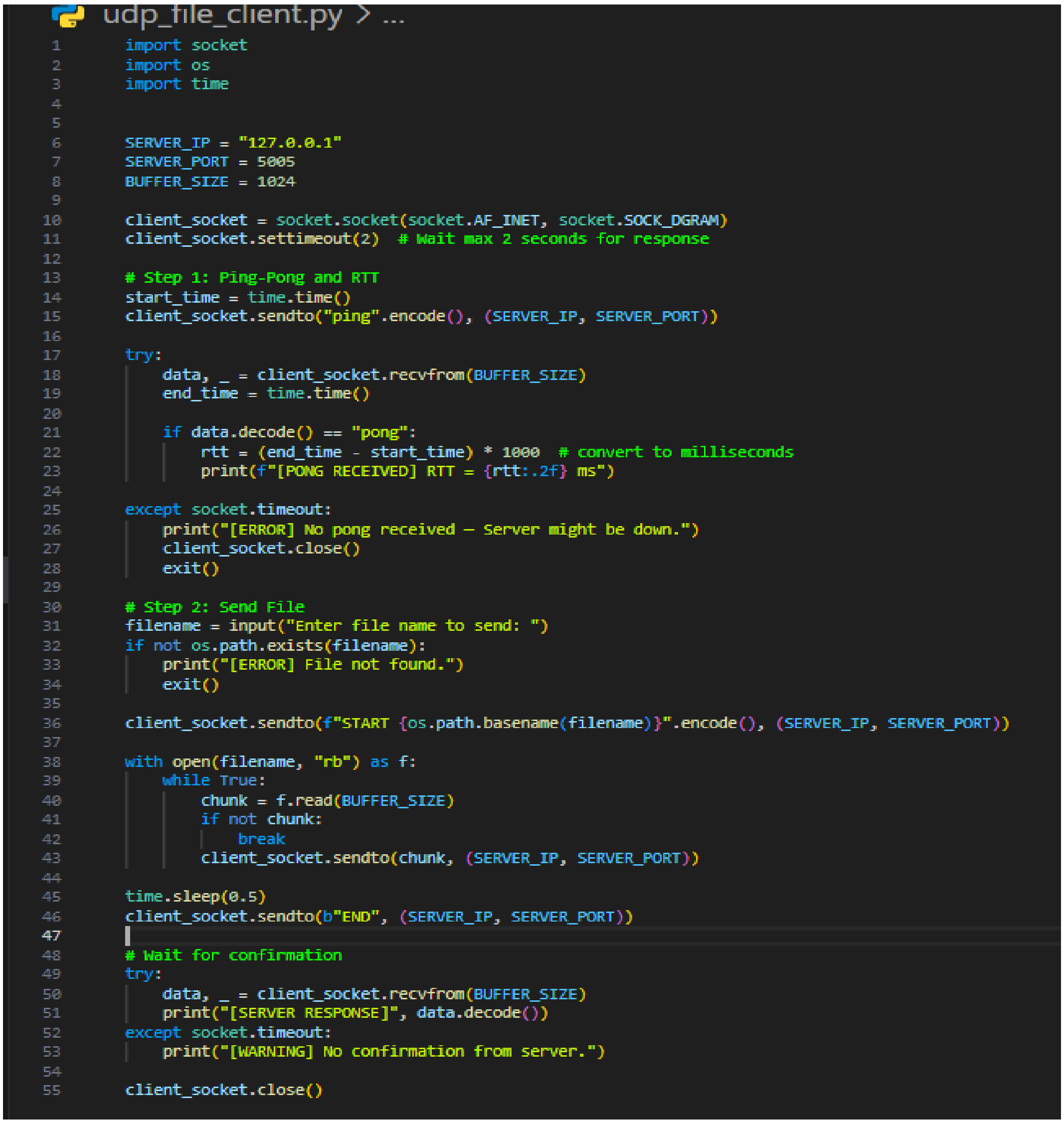
Step 8:

After that tell the client that file content is transferred successfully!

Step 9:

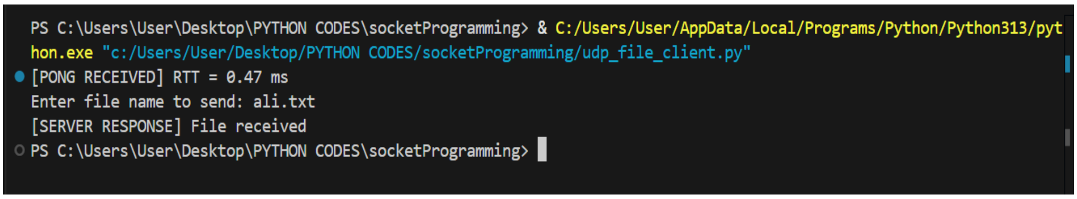
Close the client-side socket.

3.1 PYTHON CODE SNIPPET:



```python
import socket
import os
import time


SERVER_IP = "127.0.0.1"
SERVER_PORT = 5005
BUFFER_SIZE = 1024

client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
client_socket.settimeout(2)  # Wait max 2 seconds for response

# Step 1: Ping-Pong and RTT
start_time = time.time()
client_socket.sendto("ping".encode(), (SERVER_IP, SERVER_PORT))

try:
    data, _ = client_socket.recvfrom(BUFFER_SIZE)
    end_time = time.time()

    if data.decode() == "pong":
        rtt = (end_time - start_time) * 1000  # convert to milliseconds
        print(f"[PONG RECEIVED] RTT = {rtt:.2f} ms")

except socket.timeout:
    print("[ERROR] No pong received — Server might be down.")
    client_socket.close()
    exit()

# Step 2: Send File
filename = input("Enter file name to send: ")
if not os.path.exists(filename):
    print("[ERROR] File not found.")
    exit()

client_socket.sendto(f"START {os.path.basename(filename)}".encode(), (SERVER_IP, SERVER_PORT))

with open(filename, "rb") as f:
    while True:
        chunk = f.read(BUFFER_SIZE)
        if not chunk:
            break
        client_socket.sendto(chunk, (SERVER_IP, SERVER_PORT))

time.sleep(0.5)
client_socket.sendto(b"END", (SERVER_IP, SERVER_PORT))

# Wait for confirmation
try:
    data, _ = client_socket.recvfrom(BUFFER_SIZE)
    print("[SERVER RESPONSE]", data.decode())
except socket.timeout:
    print("[WARNING] No confirmation from server.")

client_socket.close()
```

3.2 OUTPUT:



```
PS C:\Users\User\Desktop\PYTHON CODES\socketProgramming> & C:/Users/User/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/User/Desktop/PYTHON CODES/socketProgramming/udp_file_client.py"
[PONG RECEIVED] RTT = 0.47 ms
Enter file name to send: ali.txt
[SERVER RESPONSE] File received
PS C:\Users\User\Desktop\PYTHON CODES\socketProgramming>
```