22BCE0476

AMAN CHAUHAN

BCSE308P - Computer Networks Lab
LAB REPORT

DA-4 Socket Programming

3. Socket
1. UDP socket
2. TCP socket
3. HTTP TCP
4. Echo TCP
5. ARP TCP
6. RARP TCP
7. Chat TCP
8. File Transfer TCP
9. DNS TCP
10. Multiuser chat TCP
11. Math server TCP

1. UDP socket programs:

Server:

```
import socket

def udp_server():
    # Create a UDP socket and bind it to port 9876
```

```
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server_socket.bind(('localhost', 9876))
    print("Server is up and running")

    while True:
        # Receive data from the client
        receive_data, client_address = server_socket.recvfrom(1024)
        sentence = receive_data.decode('utf-8').strip()
        print(f"Received from client: {sentence}")

        # Process the data (convert to uppercase)
        capitalized_sentence = sentence.upper()
        send_data = capitalized_sentence.encode('utf-8')

        # Send the response back to the client
        server_socket.sendto(send_data, client_address)
        print(f"Sent to client: {capitalized_sentence}")

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

Client:

```
import socket

def udp_client():
    # Create a UDP socket
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server_address = ('localhost', 9876)

    # Get input from the user
    sentence = input("Enter a string to convert to uppercase: ")
    send_data = sentence.encode('utf-8')

    # Send data to the server
    client_socket.sendto(send_data, server_address)

    # Receive response from the server
    receive_data, _ = client_socket.recvfrom(1024)
    capitalized_sentence = receive_data.decode('utf-8').strip()
    print(f"From server: {capitalized_sentence}")

    # Close the client socket
    client_socket.close()

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

Output:

Server output:

```
"D:\newlatestchar\CN NETWORKS PROJECT\.venv\Scripts\python.exe" "D:\newlatestchar\CN NETWORKS PROJEC
Server is up and running
Received from client: ert
Sent to client: ERT
```

Client output:

```
"D:\newlatestchar\CN NETWORKS PROJECT\.venv\Scripts\python.exe" "D:\newlatestchar\CN NETWORKS PRO
Enter a string to convert to uppercase: ert
From server: ERT

Process finished with exit code 0
```

2. TCP socket:

Server:

```python
import socket
import os

def tcp_server():
    # Set up the server socket to listen on port 4000
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(('localhost', 4000))
    server_socket.listen(1)
    print("Server is ready for connection")

    # Accept a connection from a client
    client_socket, client_address = server_socket.accept()
    print("Connection established with:", client_address)

    # Receive the filename from the client
    file_name = client_socket.recv(1024).decode('utf-8').strip()
    print(f"Client requested file: {file_name}")

    try:
        # Check if the file exists and read its contents
        if os.path.isfile(file_name):
            with open(file_name, 'r') as file:
                for line in file:
                    client_socket.sendall(line.encode('utf-8'))
                    print(f"Sent line to client: {line.strip()}")
        else:
            client_socket.sendall(b"File not found.")
            print("File not found.")

    except Exception as e:
        print(f"An error occurred: {e}")

    finally:
        # Close the connections
        client_socket.close()
        server_socket.close()
        print("Server closed the connection.")

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

Client:

```python
import socket

def tcp_client():
    # Connect to the server running on localhost at port 4000
    server_address = ('localhost', 4000)
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect(server_address)

    # Get the filename from the user
    file_name = input("Enter the filename: ")
    client_socket.sendall(file_name.encode('utf-8'))

    # Receive the file content from the server and display it
    try:
        while True:
            data = client_socket.recv(1024)
            if not data:
                break
            print(data.decode('utf-8'), end="")

    except Exception as e:
        print(f"An error occurred: {e}")

    finally:
        # Close the client socket
        client_socket.close()
        print("\nClient closed the connection.")

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

Output:

Server output:

```
"D:\newlatestchar\CN NETWORKS PROJECT\.venv\Scripts\python.exe" "D:\newlatestchar\CN NET
 .py"
Server is ready for connection
Connection established with: ('127.0.0.1', 55009)
Client requested file: hello.txt
Sent line to client: hello Aman
Server closed the connection.

Process finished with exit code 0
```

Client output:

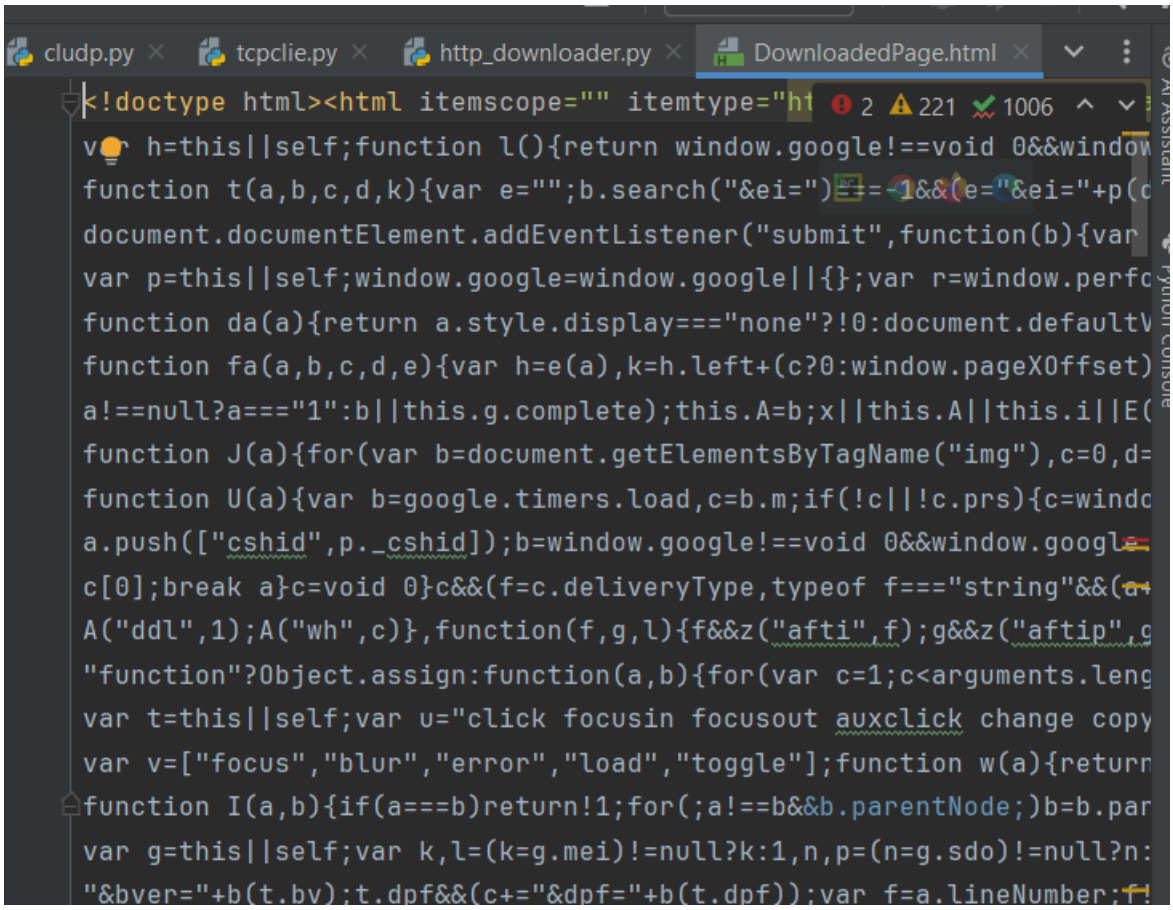
```
"D:\newlatestchar\CN NETWORKS PROJECT\.venv\Scripts\python.exe" "D:\newlatestchar\CN NETWORKS P
 .py"
Enter the filename: hello.txt
hello Aman
Client closed the connection.

Process finished with exit code 0
```

3. HTTP TCP:

```python
import requests
import os

def download_web_page(url, file_path="D:/newlatestchar/CN NETWORKS
PROJECT/DownloadedPage.html"):
    try:
        # Ensure the directory exists
        os.makedirs(os.path.dirname(file_path), exist_ok=True)

        # Send a GET request to the URL with a user-agent header
        headers = {
            "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.183 Safari/537.36"
        }
        response = requests.get(url, headers=headers, timeout=10)

        # Check if the request was successful (status code 200)
        if response.status_code == 200:
            # Write the page content to a file with the specified encoding
            with open(file_path, "w", encoding="utf-8") as file:
                file.write(response.text)
            print(f"Successfully downloaded the webpage to {file_path}")
        else:
            print(f"Failed to retrieve webpage. Status code:
{response.status_code}")

    except requests.exceptions.MissingSchema:
        print("Invalid URL format. Please check the URL.")
    except requests.exceptions.ConnectionError:
        print("Failed to establish a connection. Please check the URL or
your internet connection.")
    except requests.exceptions.Timeout:
        print("The request timed out. Please try again later.")
    except requests.exceptions.RequestException as e:
        print(f"An error occurred: {e}")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")

if __name__ == "__main__":
    url = "https://www.google.com/"
    download_web_page(url)
```

Output

```
"D:\newlatestchar\CN NETWORKS PROJECT\.venv\Scripts\python.exe" "D:\newlatestchar\CN NETWORKS
 PROJECT\http_downloader.py"
Successfully downloaded the webpage to D:/newlatestchar/CN NETWORKS PROJECT/DownloadedPage.html

Process finished with exit code 0
```



4. Echo TCP:

Server:

```
import socket


class EchoServer:
    def __init__(self, host='127.0.0.1', port=9999):
        self.server_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
        self.server_socket.bind((host, port))
        self.server_socket.listen(1)
        print(f"Server started. Listening on port {port}")

    def serve(self):
        try:
            while True:
```

```
                client_socket, client_address = self.server_socket.accept()
                print(f"Client connected from {client_address}")

                client_socket.sendall("Welcome to the Python EchoServer.
Type 'bye' to close.\n".encode())
                while True:
                    message = client_socket.recv(1024).decode()
                    if message.strip().lower() == 'bye':
                        print("Client disconnected.")
                        break
                    print(f"Received from client: {message}")
                    client_socket.sendall(f"Got: {message}".encode())

                client_socket.close()
            except Exception as e:
                print(f"Error: {e}")
            finally:
                self.server_socket.close()
                print("Server closed.")


if __name__ == "__main__":
    server = EchoServer()
    server.serve()
```

Client:

```
import socket


class EchoClient:
    def __init__(self, host='127.0.0.1', port=9999):
        self.client_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
        self.client_socket.connect((host, port))
        print("Connected to the server. Type 'bye' to end the connection.")

    def chat(self):
        try:
            # Receive the initial welcome message from the server
            welcome_message = self.client_socket.recv(1024).decode()
            print("Server:", welcome_message)

            while True:
                message = input("You: ")
                self.client_socket.sendall(message.encode())

                if message.strip().lower() == 'bye':
                    print("Disconnected from server.")
                    break

                response = self.client_socket.recv(1024).decode()
                print("Server:", response)
        except Exception as e:
            print(f"Error: {e}")
        finally:
            self.client_socket.close()


if __name__ == "__main__":
```

```
    client = EchoClient()
    client.chat()
```

Output:

```
"D:\newlatestchar\CN NETWORKS PROJECT\.venv\Scripts\python.exe" "D:\newlatestchar\CN NETWORKS PR
 .py"
Connected to the server. Type 'bye' to end the connection.
Server: Welcome to the Python EchoServer. Type 'bye' to close.

You: Hye
Server: Got: Hye
You: how are you
Server: Got: how are you
You: bye
Disconnected from server.

Process finished with exit code 0
```

Server output:

Client output:

```
echoclient  echoserver
"D:\newlatestchar\CN NETWORKS PROJECT\.venv\Scripts\python.exe" "D:\newlatestchar\CN NETWORKS PROJECT\echose
 .py"
Server started. Listening on port 9999
Client connected from ('127.0.0.1', 55463)
Received from client: Hye
Received from client: how are you
Client disconnected.
```

5. ARP TCP:

Server:

```
import socket

class ARPServer:
    def __init__(self, host='127.0.0.1', port=5604):
        self.server_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
        self.server_socket.bind((host, port))
        self.server_socket.listen(1)
        print(f"ARP Server started and listening on port {port}")
```

```
    def serve(self):
        # Predefined IP to MAC address mapping
        ip_to_mac = {
            "165.165.80.80": "6A:08:AA:C2",
            "165.165.79.1": "8A:BC:E3:FA"
        }

        try:
            while True:
                client_socket, client_address = self.server_socket.accept()
                print(f"Client connected from {client_address}")

                # Receive IP address from client
                ip_address = client_socket.recv(1024).decode().strip()
                print(f"Received IP Address: {ip_address}")

                # Send corresponding MAC address if found
                mac_address = ip_to_mac.get(ip_address, "MAC address not
found")
                client_socket.sendall(mac_address.encode())
                print(f"Sent MAC Address: {mac_address}")

                client_socket.close()
                print("Client disconnected.")
        except Exception as e:
            print(f"Error: {e}")
        finally:
            self.server_socket.close()
            print("Server closed.")

if __name__ == "__main__":
    server = ARPServer()
    server.serve()
```

Client:

```
import socket


class ARPClient:
    def __init__(self, host='127.0.0.1', port=5604):
        self.client_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
        self.client_socket.connect((host, port))
        print("Connected to ARP server.")

    def request_mac_address(self, ip_address):
        try:
            # Send IP address to the server
            self.client_socket.sendall(ip_address.encode())

            # Receive MAC address from the server
            mac_address = self.client_socket.recv(1024).decode().strip()
            print(f"The Physical Address (MAC) for {ip_address} is:
{mac_address}")
        except Exception as e:
            print(f"Error: {e}")
        finally:
            self.client_socket.close()
```

```
            print("Disconnected from server.")


if __name__ == "__main__":
    client = ARPClient()
    ip_address = input("Enter the Logical address (IP): ")
    client.request_mac_address(ip_address)
```

Output:

Server output:

```
"D:\newlatestchar\CN NETWORKS PROJECT\.venv\Scripts\python.exe" "D:\newlatestc
 .py"
Server started. Listening on port 9999
Client connected from ('127.0.0.1', 55463)
Received from client: Hye
Received from client: how are you
Client disconnected.
```

Client output:

```
arpserver × echoserver × arpclient ×
"D:\newlatestchar\CN NETWORKS PROJECT\.venv\Scripts\python.exe" "D:\newlatestchar\CN N
 .py"
Connected to ARP server.
Enter the Logical address (IP): 165.165.80.80
The Physical Address (MAC) for 165.165.80.80 is: 6A:08:AA:C2
Disconnected from server.

Process finished with exit code 0
```

6. RARP TCP:

Server:

```
import socket

def main():
    try:
        # Create a UDP socket
        server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        server_socket.bind(('localhost', 1309))
        print("Server is listening...")

        # Define the MAC to IP mapping
```

```
        mac_to_ip = {
            "6A:08:AA:C2": "165.165.80.80",
            "8A:BC:E3:FA": "165.165.79.1"
        }

        while True:
            # Receive the MAC address from client
            data, client_address = server_socket.recvfrom(1024)
            mac_address = data.decode().strip()
            print(f"Received MAC address: {mac_address}")

            # Find the corresponding IP address
            ip_address = mac_to_ip.get(mac_address, "Not found")
            print(f"Sending IP address: {ip_address}")

            # Send the IP address back to client
            server_socket.sendto(ip_address.encode(), client_address)
    except Exception as e:
        print(f"Server Error: {e}")
    finally:
        server_socket.close()

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

Client:

```
import socket

def main():
    try:
        # Create a UDP socket
        client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

        # Server address and port
        server_address = ('localhost', 1309)

        # Get MAC address input from the user
        mac_address = input("Enter the Physical address (MAC): ").strip()
        print(f"Sending MAC address: {mac_address}")

        # Send MAC address to the server
        client_socket.sendto(mac_address.encode(), server_address)

        # Receive IP address from the server
        data, _ = client_socket.recvfrom(1024)
        ip_address = data.decode().strip()
        print(f"The Logical Address is (IP): {ip_address}")
    except Exception as e:
        print(f"Client Error: {e}")
    finally:
        client_socket.close()

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

Output:

Server output:

```
"D:\newlatestchar\CN NETWORKS PROJECT\.venv\Scripts\python.exe" "D:\n
 .py"
Server is listening...
Received MAC address: 189.123.45.66
Sending IP address: Not found
Received MAC address: 6A:08:AA:C2
Sending IP address: 165.165.80.80
```

Client output:

```
"D:\newlatestchar\CN NETWORKS PROJECT\.venv\Scripts\python.exe" "D:\newlatestchar\CN NETW
 .py"
Enter the Physical address (MAC): 6A:08:AA:C2
Sending MAC address: 6A:08:AA:C2
The Logical Address is (IP): 165.165.80.80

Process finished with exit code 0
```

7. Chat TCP:

Server:

```python
import socket


def start_server():
    # Create a server socket, bind to port 2000, and listen for incoming
connections
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(('0.0.0.0', 2000))
    server_socket.listen(1)
    print("Server is listening on port 2000...")

    # Accept a connection from a client
    client_socket, client_address = server_socket.accept()
    print(f"Connected to {client_address}")

    # Create input and output streams for communication
    client_input = client_socket.makefile('r')
    client_output = client_socket.makefile('w')

    while True:
        # Read message from client
        client_message = client_input.readline().strip()
        if client_message.lower() == "end":
            client_output.write("BYE\n")
```

```
            client_output.flush()
            break
        print(f"Client: {client_message}")

        # Respond to client
        server_message = input("Server: ")
        client_output.write(server_message + "\n")
        client_output.flush()

    # Close the connections
    client_input.close()
    client_output.close()
    client_socket.close()
    server_socket.close()


# Start the server
start_server()
```

Client:

```
import socket

def start_client():
    # Create a client socket and connect to the server at localhost on port
2000
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect(('127.0.0.1', 2000))

    # Create input and output streams for communication
    server_input = client_socket.makefile('r')
    server_output = client_socket.makefile('w')

    while True:
        # Get message from user to send to server
        client_message = input("Client: ")
        server_output.write(client_message + "\n")
        server_output.flush()

        # Read response from server
        server_message = server_input.readline().strip()
        print(f"Server: {server_message}")

        if server_message.lower() == "bye":
            break

    # Close the connection
    server_input.close()
    server_output.close()
    client_socket.close()

# Start the client
start_client()
```

Output:

Server output:

```
"D:\newlatestchar\CN NETWORKS PROJECT\.venv\Scripts\python.exe" "D:
 .py"
Server is listening on port 2000...
Connected to ('127.0.0.1', 55875)
Client: HI
Server: Bye
Client:
Server: What are you
```

Client output:

```
"D:\newlatestchar\CN NETWORKS PROJECT\.venv\Scripts\python.exe" "D:\newlate
 .py"
Client: HI
Server: Bye

Process finished with exit code 0
```

8. File Transfer TCP:

Server:

```
import socket
import os

def start_server():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(('localhost', 5000))
    server_socket.listen(1)
    print("Server listening on port 5000...")

    client_socket, client_address = server_socket.accept()
    print(f"Connection established with {client_address}")

    file_path = "hello.txt"
    file_size = os.path.getsize(file_path)
    with open(file_path, 'rb') as file:
        bytes_sent = 0
        while bytes_sent < file_size:
            # Read in chunks of 10000 bytes
            chunk = file.read(10000)
            client_socket.send(chunk)
            bytes_sent += len(chunk)
            print(f"Sending file... {(bytes_sent * 100) // file_size}%
complete")

    client_socket.close()
    server_socket.close()
    print("File sent successfully!")
```

```
# Start the server
start_server()
```

Client:

```
import socket

def start_client():
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect(('localhost', 5000))

    file_path = "received_Text.txt"
    with open(file_path, 'wb') as file:
        while True:
            data = client_socket.recv(10000)
            if not data:
                break
            file.write(data)

    client_socket.close()
    print("File saved successfully!")

# Start the client
start_client()
```

Output:

Server output:

```
"D:\newlatestchar\CN NETWORKS PROJECT\.venv\Scripts\python.exe" "D
 .py"
Server listening on port 5000...
Connection established with ('127.0.0.1', 55932)
Sending file... 100% complete
File sent successfully!

Process finished with exit code 0
```

Client output:

```
"D:\newlatestchar\CN NETWORKS PROJECT\.venv\Scripts\python.exe" "D:\newlatestch
 .py"
File saved successfully!

Process finished with exit code 0
```

9. DNS TCP:

Server:

```
import socket


def index_of(array, str):
    str = str.strip()
    for i in range(len(array)):
        if array[i] == str:
            return i
    return -1


def start_server():
    hosts = ["zoho.com", "gmail.com", "google.com", "facebook.com"]
    ip = ["172.28.251.59", "172.217.11.5", "172.217.11.14", "31.13.71.36"]

    server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server_socket.bind(('localhost', 1362))
    print("DNS Server is running. Press Ctrl+C to quit.")

    while True:
        # Receive data from client
        data, client_address = server_socket.recvfrom(1024)
        request_host = data.decode('utf-8')
        print(f"Request for host: {request_host}")

        # Find the IP for the requested host
        if index_of(hosts, request_host) != -1:
            response = ip[index_of(hosts, request_host)]
        else:
            response = "Host Not Found"

        # Send the response back to the client
        server_socket.sendto(response.encode('utf-8'), client_address)


# Start the server
start_server()
```

Client:

```
import socket

def start_client():
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server_address = ('localhost', 1362)

    host_name = input("Enter the hostname: ")
    client_socket.sendto(host_name.encode('utf-8'), server_address)

    data, server = client_socket.recvfrom(1024)
    print(f"IP Address: {data.decode('utf-8')}")

    client_socket.close()
```

```
# Start the client
start_client()
```

Output:

Server output:

```
"D:\newlatestchar\CN NETWORKS PROJECT\.venv\Scripts\python.exe" "D:\newlatestchar\CN
 .py"
DNS Server is running. Press Ctrl+C to quit.
Request for host: google.com
```

Client output:

```
"D:\newlatestchar\CN NETWORKS PROJECT\.venv\Scripts\python.exe" "D:\newlatestc
 .py"
Enter the hostname: google.com
IP Address: 172.217.11.14

Process finished with exit code 0
```

10. Multiuser chat TCP:

```
import socket
import threading

# List to hold all connected clients
clients = []


# Function to broadcast messages to all clients except the sender
def broadcast(message, client_socket):
    for client in clients:
        if client != client_socket:
            try:
                client.send(message.encode())
            except:
                clients.remove(client)


# Function to handle communication with each client
def handle_client(client_socket, client_address):
    # Generate default names for users
    client_name = f"User{len(clients) + 1}"
    print(f"{client_name} has joined the chat.")

    # Add client to the list of clients
```

```
    clients.append(client_socket)

    # Broadcast the join message to all other clients
    broadcast(f"{client_name} has joined the chat!", client_socket)

    while True:
        try:
            message = client_socket.recv(1024).decode()
            if message.lower() == "exit":
                break
            broadcast(f"{client_name}: {message}", client_socket)
        except:
            break

    # Remove client from list and close connection
    clients.remove(client_socket)
    client_socket.close()
    broadcast(f"{client_name} has left the chat.", client_socket)
    print(f"{client_name} disconnected.")


# Function to start the server
def start_server():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(('127.0.0.1', 12345))
    server_socket.listen(5)
    print("Server started on port 12345")

    while True:
        client_socket, client_address = server_socket.accept()
        print(f"Connection from {client_address} established.")

        # Handle each client in a new thread
        client_thread = threading.Thread(target=handle_client,
args=(client_socket, client_address))
        client_thread.start()


# Start the server
if __name__ == "__main__":
    start_server()
```

```
import socket
import threading

# Function to receive messages from the server
def receive_messages(client_socket):
    while True:
        try:
            message = client_socket.recv(1024).decode()
            if message:
                print(message)
        except:
            print("Connection lost.")
            break

# Function to start the client
def start_client(client_id):
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect(('127.0.0.1', 12345))
```

```
    # Start a thread to listen for incoming messages from the server
    receive_thread = threading.Thread(target=receive_messages,
args=(client_socket,))
    receive_thread.daemon = True  # Allow thread to exit when the main
program exits
    receive_thread.start()

    # Set the default name for the client
    name = f"User{client_id}"
    client_socket.send(name.encode())

    # Start sending messages to the server
    while True:
        message = input(f"{name}: ")
        if message.lower() == "exit":
            client_socket.send(message.encode())
            break
        client_socket.send(message.encode())

    client_socket.close()

# Start the client
if __name__ == "__main__":
    # Assume the client_id is passed as an argument or can be assigned
manually
    client_id = 1  # For example, this client will be "User1"
    start_client(client_id)
```

Output:

```
"D:\newlatestchar\CN NETWORKS PROJECT\.venv\Scripts\python.exe" "D:
 PROJECT\multiuserclient.py"
User1: hello
User1: cx
User1: sfsgs
User1: sdfg
User1: sf
User1: rfhy
```

```
"D:\newlatestchar\CN NETWORKS PROJECT\.venv\Scripts\python.exe" "D:\newlatestchar\C
 PROJECT\multiuserchatserver.py"
Server started on port 12345
Connection from ('127.0.0.1', 60649) established.
User1 has joined the chat.
User1 disconnected.
```

Multi user:

11. Math server TCP:

Server:

```
import socket
import threading


def handle_client(client_socket):
    while True:
        try:
            # Receive equation from client
            equation = client_socket.recv(1024).decode()
            if equation.lower() == "bye":
                print("Client disconnected.")
                break

            print(f"Equation received: {equation}")

            # Parse the equation
            tokens = equation.split()
            if len(tokens) != 3:
                result = "Error: Invalid equation format"
            else:
                try:
                    operand1 = int(tokens[0])
                    operator = tokens[1]
                    operand2 = int(tokens[2])

                    # Perform the operation
                    if operator == "+":
                        result = operand1 + operand2
                    elif operator == "-":
                        result = operand1 - operand2
                    elif operator == "*":
                        result = operand1 * operand2
                    elif operator == "/":
                        if operand2 != 0:
                            result = operand1 / operand2
                        else:
                            result = "Error: Division by zero"
                    else:
                        result = "Error: Invalid operator"
                except ValueError:
                    result = "Error: Invalid operand(s)"

            print(f"Sending result: {result}")
            client_socket.send(str(result).encode())  # Ensure result is
sent as a string

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

    client_socket.close()


def start_server():
```

```
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(('127.0.0.1', 4444))
    server_socket.listen(5)
    print("Server started on port 4444")

    while True:
        client_socket, client_address = server_socket.accept()
        print(f"Connection from {client_address} established.")

        # Handle each client in a new thread
        client_thread = threading.Thread(target=handle_client,
args=(client_socket,))
        client_thread.start()


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

Client:

```
import socket


def start_client():
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect(('127.0.0.1', 4444))

    while True:
        # Get user input for the equation
        equation = input("Enter the equation in the form 'operand operator
operand' (e.g., 2 + 3): ")
        if equation.lower() == "bye":
            client_socket.send(equation.encode())
            break

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

        # Receive the result from the server
        result = client_socket.recv(1024).decode()
        print(f"Answer = {result}")

    client_socket.close()


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

Output:

Server output:

```
"D:\newlatestchar\CN NETWORKS PROJECT\.venv\Scripts\python.exe" "D:\newlatestchar\CN NETWORKS
 PROJECT\multiuserchatserver.py"
Server started on port 12345
Connection from ('127.0.0.1', 60649) established.
User1 has joined the chat.
User1 disconnected.

Process finished with exit code -1
```

Client output:

```
"D:\newlatestchar\CN NETWORKS PROJECT\.venv\Scripts\python.exe" "D:\newlatestchar\
 PROJECT\multiuserclient.py"
Enter the equation in the form 'operand operator operand' (e.g., 2 + 3): 2+5
Answer = Error: Invalid equation format
Enter the equation in the form 'operand operator operand' (e.g., 2 + 3): 4 + 6
Answer = 10
Enter the equation in the form 'operand operator operand' (e.g., 2 + 3): 3 +7
Answer = Error: Invalid equation format
Enter the equation in the form 'operand operator operand' (e.g., 2 + 3): 2 + 9
Answer = 11
Enter the equation in the form 'operand operator operand' (e.g., 2 + 3): bye

Process finished with exit code 0
```