```python
# Abhishek M J - CS21B2018
# server.py

import socket
import threading
import time
import os
# to url_encode and decode
import urllib.parse
from dataclasses import dataclass

# IP = socket.gethostbyname(socket.gethostname())
IP = ''
PORT = 53533
ADDR = (IP, PORT)
SIZE = 1024
FORMAT = "utf-8"
DISCONNECT_MESSAGE = "QUIT!"

clients = [] # list of clients connected to the server

@dataclass
class Client:
    '''Client class to store client information'''
    conn: socket.socket
    addr: str
    ok: bool


class FileNotSent(Exception):
    '''Exception raised when file is not sent'''
    pass


def find_client(addr: str) -> Client | None:
    '''Find client with given address'''
    for client in clients:
        if client.addr == addr:
            return client
    return None


def msg_encode(msg: str, from_client: Client | None = None) -> bytes:
    '''
    Encode message to be sent to client

    MSG FORMAT: <to_addr>/<msg>
    '''
    msg = urllib.parse.quote(msg)
    if from_client is None: # if message is from server
        to_msg = f"SERVER/{msg}"
    else:
        to_msg = f"{from_client.addr}/{msg}"
    return to_msg.encode(FORMAT)


def forward_file(from_client: Client, to_client: Client, file_name: str):
    '''
    Forward file from one client to another:
    1. Send file name to to_client: <to_addr>/<file_name>
    2. Send file data in raw bytes
    3. Send EOF to indicate end of file
    '''
    to_client.conn.send(msg_encode(file_name, from_client))
    time.sleep(0.1)

    file_data = from_client.conn.recv(SIZE)
    while file_data != b"EOF":
        to_client.conn.send(file_data)
        file_data = from_client.conn.recv(SIZE)
    time.sleep(0.1)
    to_client.conn.send(file_data)


def handle_msg(from_msg: str, from_client: Client):
    '''
    Handle message recieved from client:
    1. File transfer: <to_addr>/<file_name>
    2. Acknowledgement: SERVER/<to_addr>
    '''
    try:
        to_addr, msg = from_msg.strip().split("/")
        msg = urllib.parse.unquote(msg)
    except ValueError:
        raise FileNotSent(f"Invalid message format: {from_msg}")

    # If acknoledgement is recieved
    if to_addr == "SERVER":
        to_addr = msg # to_addr for acknowledgement
        file_name = "File sent"
        from_client = Client(None, "SERVER", True)
    else: # If message is to be sent
        file_name = msg # file_name for file transfer
        print(f"[SENDING] SERVER -> {from_client.addr}: File name recieved")
        from_client.conn.send(msg_encode("File name recieved"))

    to_client = find_client(to_addr)
    if to_client is None:
        raise FileNotSent(f"Client {to_addr} not found.")

    print(f"[SENDING] {from_client.addr} -> {to_client.addr}: {file_name}")
    try:
        if from_client.conn: # if file transfer
            forward_file(from_client, to_client, file_name)
        else: # if acknowledgement
            to_client.conn.send(msg_encode(file_name, from_client))
    except BrokenPipeError:
        disconnect_client(to_client)
        raise FileNotSent(f"Client {to_addr} disconnected.")


def server_broadcast(msg: str):
    '''Send message to all clients'''
    for client in clients:
        try:
            client.conn.send(msg_encode(msg))
        except BrokenPipeError:
            disconnect_client(client)


def disconnect_client(client: Client):
    '''Disconnect client'''
    client.ok = False

    print(f"[DISCONNECT CLIENT] {client.addr} disconnected.")
    print(f"[ACTIVE CLIENTS] {threading.active_count() - 2}")

    # broadcast client disconnect: <client_addr>-
    server_broadcast(f"{client.addr}-")

    clients.remove(client)
    client.conn.close()


def handle_client(client: Client):
    '''
    Handle client connection:
    1. Send all connected clients to new client
    2. Recieve message from client
    3. Handle recieved message
    '''
    addr = client.addr
    conn = client.conn
    print(f"[NEW CLIENT] {addr} connected.")

    for c in clients: # send all connected clients to new client
        if c.addr != addr:
            conn.send(msg_encode(f"{c.addr}+"))
            time.sleep(0.2)

    while client.ok: # recieve message from client
        from_msg = conn.recv(SIZE).decode(FORMAT)
        if from_msg == DISCONNECT_MESSAGE:
            client.ok = False
            break
        elif not from_msg:
            continue

        try: # handle recieved message
            handle_msg(from_msg, client)
        except FileNotSent as e:
            print(f"[ERROR] {e}")
            conn.send(msg_encode(str(e)))

    try:
        disconnect_client(client)
    except Exception as e:
        pass

def main():
    print(f"[STARTING] Server is starting...")

    # create socket, bind to address, and listen
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(ADDR)
    server.listen()
    print(f"[LISTENING] Server is listening on {IP}:{PORT}")

    print(f"[ACTIVE CLIENTS] {threading.active_count() - 1}")

    while True: # accept new connection
        conn, addr = server.accept()
        addr = f"{addr[0]}:{addr[1]}"
        current_client = Client(conn, addr, True)
        clients.append(current_client)

        server_broadcast(f"{current_client.addr}+") # broadcast new client: <client_addr>+

        # start new thread to handle client
        thread = threading.Thread(target=handle_client, args=(current_client,))
        thread.start()

        print(f"[ACTIVE CLIENTS] {threading.active_count() - 1}")

if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt: # handle keyboard interrupt
        print("\n[EXITING] Server is shutting down...")
        for client in clients: # send disconnect message to all clients
            client.conn.send(DISCONNECT_MESSAGE.encode(FORMAT))
        os._exit(0)
```

```python
# Abhishek M J - CS21B2018
# client.py

import socket
import threading
import os
import readline
import time
# to url_encode and decode
import urllib.parse

IP = socket.gethostbyname(socket.gethostname())
# IP = ''
PORT = 53533
ADDR = (IP, PORT)
SIZE = 1024
FORMAT = "utf-8"
DISCONNECT_MESSAGE = "QUIT!"

# create client socket
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

folder_path = "client/" # folder to store files for client

clients = [] # list of clients connected to the server


class FileNotSent(Exception):
    '''Exception raised when file is not sent'''
    pass


current_prompt = "" # current input prompt


def print_msg(msg: str):
    '''Print message without overwriting current input prompt'''
    if not current_prompt:
        print(msg)
    else:
        input_buffer = readline.get_line_buffer()
        print(f"\r{msg}\n{current_prompt}{input_buffer}", end="", flush=True)


def input_msg(string: str):
    '''Same as input() but stores current input prompt while taking input'''
    global current_prompt

    current_prompt = string
    result = input(f"\r{string}")
    current_prompt = ""

    return result


def msg_encode(msg: str, to_addr: str = None) -> bytes:
    '''
    Encode message to be sent to client:

    MSG FORMAT: <to_addr>/<msg>
    '''
    msg = urllib.parse.quote(msg)
    if to_addr is None:
        to_msg = f"SERVER/{msg}"
    else:
        to_msg = f"{to_addr}/{msg}"
    return to_msg.encode(FORMAT)


def send_file(client: socket.socket, to_addr: str, file_path: str):
    '''
    Send file to another client through server:
    1. Send file name to to_client: <to_addr>/<file_name>
    2. Send file data in raw bytes
    3. Send EOF to indicate end of file
    '''
    if not os.path.exists(file_path):
        print_msg(f"[ERROR] File '{file_path}' not found.")
        return

    client.send(msg_encode(os.path.basename(file_path), to_addr))
    time.sleep(0.1)

    with open(file_path, "rb") as f:
        bin_data = f.read(SIZE)
        while bin_data:
            client.send(bin_data)
            bin_data = f.read(SIZE)
        time.sleep(0.1)
        client.send(b"EOF")


def recieve_file(client: socket.socket, file_name: str):
    '''
    Recieve file from another client through server:
    1. Recieve file data in raw bytes
    2. Write file data to file
    3. Recieve EOF to indicate end of file
    '''
    file_path = folder_path + file_name
    with open(file_path, "wb") as f:
        bin_data = client.recv(SIZE)
        while bin_data != b"EOF":
            f.write(bin_data)
            bin_data = client.recv(SIZE)


def handle_msg(from_msg: str, client: socket.socket):
    '''
    Handle message recieved from server:
    1. File transfer: <from_addr>/<file_name>
    2. Acknowledgement: SERVER/<from_addr>
    3. Client online: SERVER/<client_addr>+
    4. Client offline: SERVER/<client_addr>-
    '''
    try:
        from_addr, msg = from_msg.strip().split("/")
        msg = urllib.parse.unquote(msg)
    except ValueError:
        raise FileNotSent(f"Invalid message format: {from_msg}")

    # Check if broadcast
    if from_addr == "SERVER":
        if msg[-1] == "+": # if client online
            if len(clients) == 0: # first client is current client
                print_msg(f"[CURRENT CLIENT] {msg[:-1]}")
                global folder_path
                folder_path = msg[:-1] + "/"
                if not os.path.exists(folder_path): # create folder if not exists
                    os.makedirs(folder_path)
            else:
                print_msg(f"[CLIENT ONLINE] {msg[:-1]} connected.")
            clients.append(msg[:-1])
        elif msg[-1] == "-": # if client offline
            clients.remove(msg[:-1])
            print_msg(f"[CLIENT OFFLINE] {msg[:-1]} disconnected.")
        else: # if acknowledgement
            print_msg(f"[SERVER] {msg}")
    else: # if file transfer
        print_msg(f"[{from_addr}] {msg}")
        recieve_file(client, msg)
        # send acknowledgement
        client.send(msg_encode(from_addr))


def disconnect_server(client: socket.socket, recv_from: str):
    '''
    Disconnect from server, received from "client" or "server":
    '''
    client.send(DISCONNECT_MESSAGE.encode(FORMAT))
    if recv_from == "client":
        print_msg(f"[DISCONNECTED] Client disconnected from {IP}:{PORT}")
    elif recv_from == "server":
        print_msg(f"[DISCONNECTED] Server disconnected from Client.")
    client.close()
    os._exit(0)


def handle_server(client: socket.socket):
    '''
    Handle server:
    1. Recieve message from server
    2. Handle message
    '''
    connected = True
    while connected:
        try:
            msg = client.recv(SIZE).decode(FORMAT)
        except OSError:
            return
        if not msg or msg == DISCONNECT_MESSAGE:
            connected = False
            break

        try:
            handle_msg(msg, client)
        except FileNotSent as e:
            print_msg(f"[ERROR] {e}")

    try:
        disconnect_server(client, "server")
    except Exception:
        pass


def main():
    # connect to server
    global client
    client.connect(ADDR)
    print_msg(f"[CONNECTED] Client connected to {IP}:{PORT}")

    # start server thread to handle messages from server
    server_thread = threading.Thread(target=handle_server, args=(client,))
    server_thread.start()

    connected = True
    while connected: # send file to other clients
        to_addr = input_msg("(ip:port)> ").strip() # get client address

        if to_addr == DISCONNECT_MESSAGE:
            connected = False
            disconnect_server(client, "client")

        elif to_addr.lower() in ["", "l", "list"]: # list all clients
            print_msg(f"[ONLINE CLIENT LIST] {len(clients)} clients connected.")
            print_msg(f"[CURRENT CLIENT] {clients[0]}")
            for c in clients[1:]:
                print_msg(f"[CLIENT] {c}")
            continue

        if to_addr not in clients:
            print_msg(f"[ERROR] Client '{to_addr}' not found.")
            continue

        file_path = input_msg("{file}> ").strip() # get file path
        send_file(client, to_addr, file_path)

    disconnect_server(client, "client")

if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt: # handle keyboard interrupt
        disconnect_server(client, "client")
```

```
…hek@hp in repo: CN/06-Lab on  main [!?] via  v3.11.5 took 4ms
 ╰─ python server.py
[STARTING] Server is starting...
[LISTENING] Server is listening on :53533
[ACTIVE CLIENTS] 0
[NEW CLIENT] 127.0.0.1:37632 connected.
[ACTIVE CLIENTS] 1
[NEW CLIENT] 127.0.0.1:37636 connected.
[ACTIVE CLIENTS] 2
[SENDING] SERVER → 127.0.0.1:37632: File name recieved
[SENDING] 127.0.0.1:37632 → 127.0.0.1:37636: network.png
[SENDING] SERVER → 127.0.0.1:37632: File sent
[SENDING] SERVER → 127.0.0.1:37636: File name recieved
[SENDING] 127.0.0.1:37636 → 127.0.0.1:37632: lab.pdf
[SENDING] SERVER → 127.0.0.1:37636: File sent
[DISCONNECT CLIENT] 127.0.0.1:37632 disconnected.
[ACTIVE CLIENTS] 1
[DISCONNECT CLIENT] 127.0.0.1:37636 disconnected.
[ACTIVE CLIENTS] 0
```

```
…ek@hp in repo: CN/06-Lab on  main [!?] via  v3.11.5 took 4ms
 ╰─ python client.py
[CONNECTED] Client connected to 127.0.0.1:53533
[CURRENT CLIENT] 127.0.0.1:37632
[CLIENT ONLINE] 127.0.0.1:37636 connected.
(ip:port)> 127.0.0.1:37636
(file)> network.png
[SERVER] File name recieved
[SERVER] File sent
[127.0.0.1:37636] lab.pdf
(ip:port)> QUIT!
[DISCONNECTED] Client disconnected from 127.0.0.1:53533

…@hp in repo: CN/06-Lab on  main [!?] via  v3.11.5 took 1m46s
 ╰─
```

```
…ek@hp in repo: CN/06-Lab on  main [!?] via  v3.11.5 took 4ms
 ╰─ python client.py
[CONNECTED] Client connected to 127.0.0.1:53533
[CURRENT CLIENT] 127.0.0.1:37636
[CLIENT ONLINE] 127.0.0.1:37632 connected.
[127.0.0.1:37632] network.png
(ip:port)> 127.0.0.1:37632
(file)> lan.pdf
[ERROR] File 'lan.pdf' not found.
(ip:port)> lab.pdf
[ERROR] Client 'lab.pdf' not found.
(ip:port)> 127.0.0.1:37632
(file)> lab.pdf
[SERVER] File name recieved
[SERVER] File sent
[CLIENT OFFLINE] 127.0.0.1:37632 disconnected.
(ip:port)> QUIT!
[DISCONNECTED] Client disconnected from 127.0.0.1:53533

…@hp in repo: CN/06-Lab on  main [!?] via  v3.11.5 took 1m48s
 ╰─
```

```
…ek@hp in repo: CN/06-Lab on  main [!?] via  v3.11.5 took 13ms
 ╰─ ls
drwxr-xr-x     - abhishek  9 Sep 22:13  127.0.0.1:37632
drwxr-xr-x     - abhishek  9 Sep 22:12  127.0.0.1:37636
drwxr-xr-x     - abhishek  9 Sep 20:53  img
.rw-r--r-- 6.4k abhishek  9 Sep 22:11  client.py
.rw-r--r-- 203k abhishek  9 Sep 22:03  lab.pdf
.rw-r--r--  51k abhishek  9 Sep 22:03  network.png
.rw-r--r-- 5.8k abhishek  9 Sep 22:05  server.py

…ek@hp in repo: CN/06-Lab on  main [!?] via  v3.11.5 took 13ms
 ╰─ # Files sent from both the clients
```

```
…@hp in repo: CN/06-Lab/127.0.0.1:37632 on  main [!?] took 39ms
 ╰─ ls
.rw-r--r-- 203k abhishek  9 Sep 22:13  lab.pdf

…@hp in repo: CN/06-Lab/127.0.0.1:37632 on  main [!?] took 13ms
 ╰─ # Files received in above client: lab.pdf
```

```
…p in repo: CN/06-Lab/127.0.0.1:37636 on  main [!?] took 42ms
 ╰─ ls
.rw-r--r-- 51k abhishek  9 Sep 22:12  network.png

…p in repo: CN/06-Lab/127.0.0.1:37636 on  main [!?] took 12ms
 ╰─ # Files received in above client: network.png
```