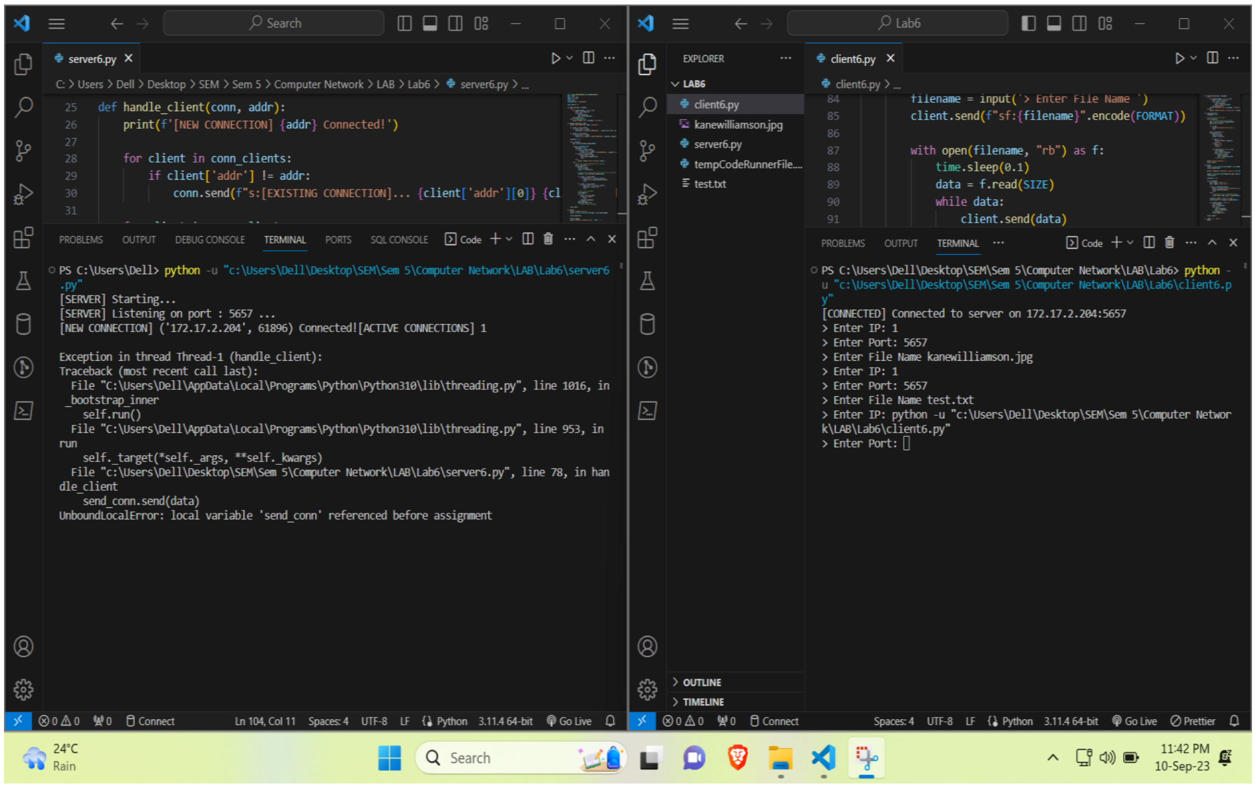
NITIN REDDY K

CS21B2028

LAB 6:

Terminal After Execution:



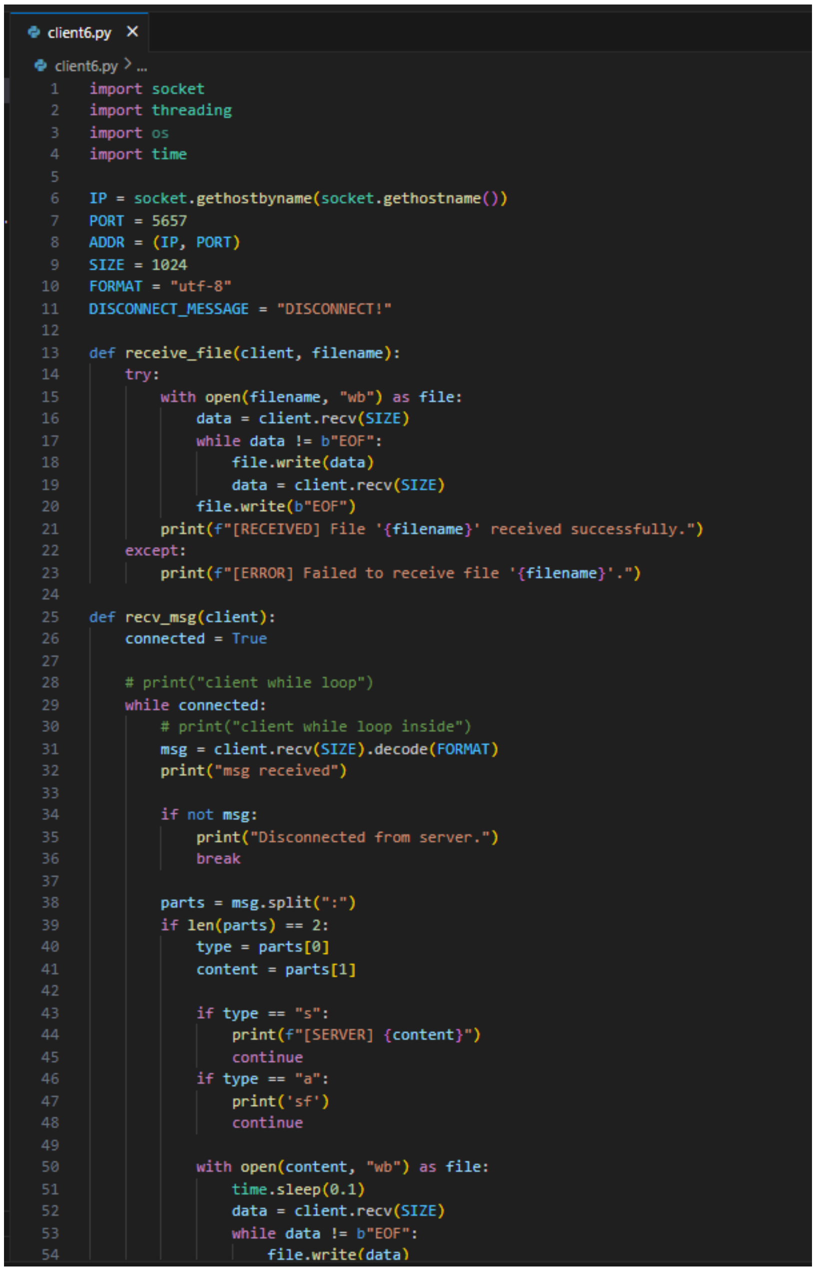
```
def handle_client(conn, addr):
    print(f'[NEW CONNECTION] {addr} Connected!')

    for client in conn_clients:
        if client['addr'] != addr:
            conn.send(f"s:[EXISTING CONNECTION]... {client['addr'][0]} {cl
```

```
PS C:\Users\Dell> python -u "c:\Users\Dell\Desktop\SEM\Sem 5\Computer Network\LAB\Lab6\server6
.py"
[SERVER] Starting...
[SERVER] Listening on port : 5657 ...
[NEW CONNECTION] ('172.17.2.204', 61896) Connected![ACTIVE CONNECTIONS] 1

Exception in thread Thread-1 (handle_client):
Traceback (most recent call last):
  File "C:\Users\Dell\AppData\Local\Programs\Python\Python310\lib\threading.py", line 1016, in
 _bootstrap_inner
    self.run()
  File "C:\Users\Dell\AppData\Local\Programs\Python\Python310\lib\threading.py", line 953, in
run
    self._target(*self._args, **self._kwargs)
  File "c:\Users\Dell\Desktop\SEM\Sem 5\Computer Network\LAB\Lab6\server6.py", line 78, in han
dle_client
    send_conn.send(data)
UnboundLocalError: local variable 'send_conn' referenced before assignment
```

```
            filename = input('> Enter File Name ')
            client.send(f"sf:{filename}".encode(FORMAT))

            with open(filename, "rb") as f:
                time.sleep(0.1)
                data = f.read(SIZE)
                while data:
                    client.send(data)
```

```
PS C:\Users\Dell\Desktop\SEM\Sem 5\Computer Network\LAB\Lab6> python -
u "c:\Users\Dell\Desktop\SEM\Sem 5\Computer Network\LAB\Lab6\client6.p
y"
[CONNECTED] Connected to server on 172.17.2.204:5657
> Enter IP: 1
> Enter Port: 5657
> Enter File Name kanewilliamson.jpg
> Enter IP: 1
> Enter Port: 5657
> Enter File Name test.txt
> Enter IP: python -u "c:\Users\Dell\Desktop\SEM\Sem 5\Computer Networ
k\LAB\Lab6\client6.py"
> Enter Port: 
```

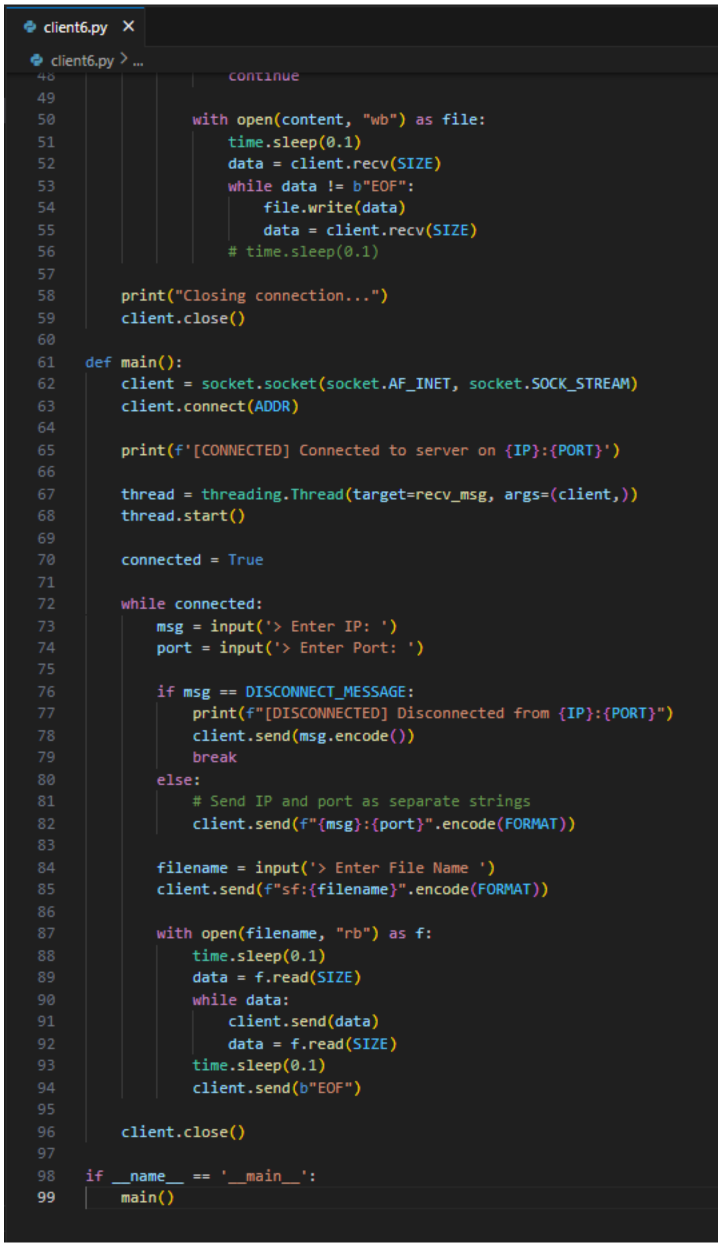
Client Code:



client6.py ×

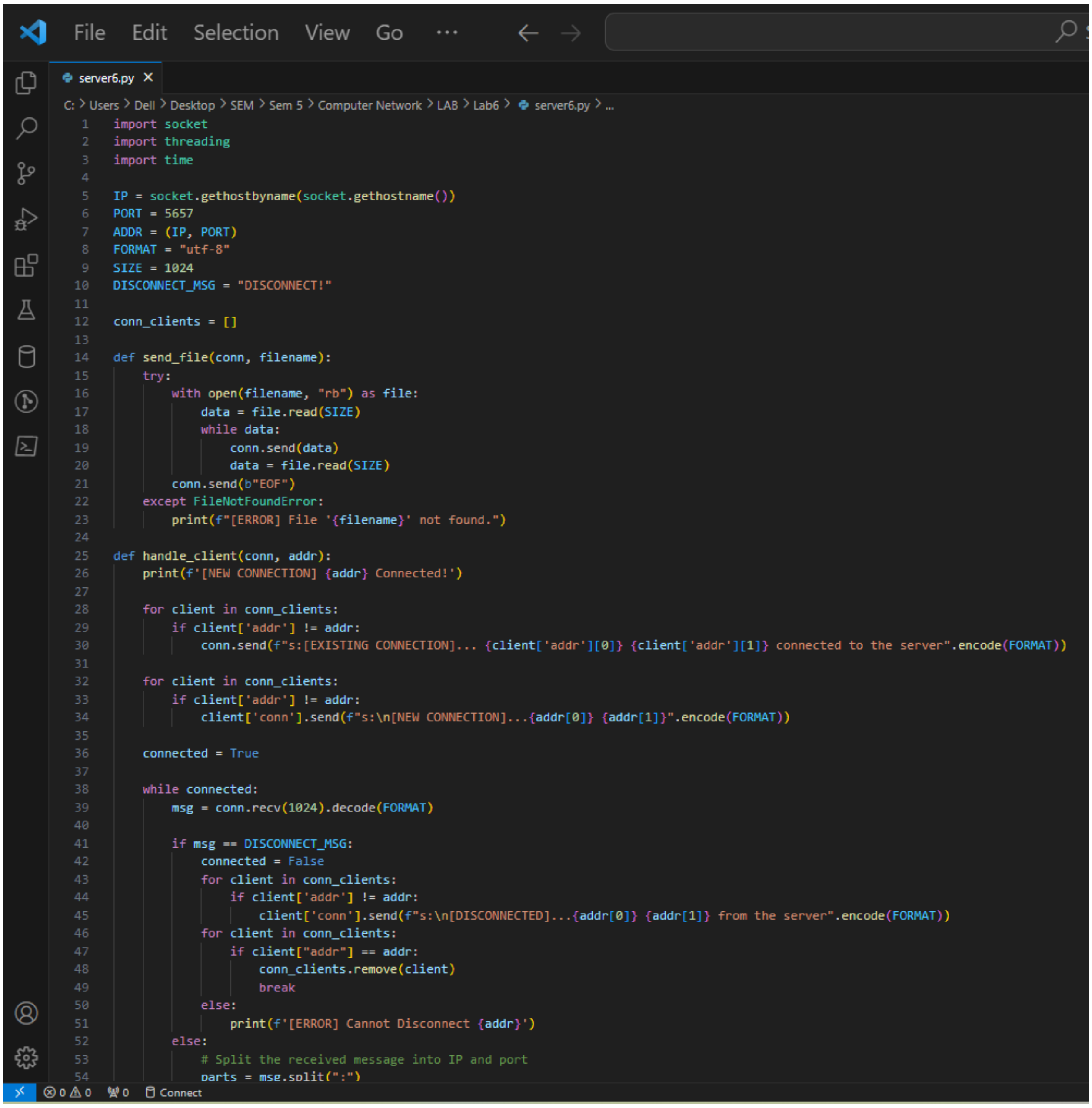
client6.py > ...

```python
import socket
import threading
import os
import time

IP = socket.gethostbyname(socket.gethostname())
PORT = 5657
ADDR = (IP, PORT)
SIZE = 1024
FORMAT = "utf-8"
DISCONNECT_MESSAGE = "DISCONNECT!"

def receive_file(client, filename):
    try:
        with open(filename, "wb") as file:
            data = client.recv(SIZE)
            while data != b"EOF":
                file.write(data)
                data = client.recv(SIZE)
            file.write(b"EOF")
        print(f"[RECEIVED] File '{filename}' received successfully.")
    except:
        print(f"[ERROR] Failed to receive file '{filename}'.")

def recv_msg(client):
    connected = True

    # print("client while loop")
    while connected:
        # print("client while loop inside")
        msg = client.recv(SIZE).decode(FORMAT)
        print("msg received")

        if not msg:
            print("Disconnected from server.")
            break

        parts = msg.split(":")
        if len(parts) == 2:
            type = parts[0]
            content = parts[1]

            if type == "s":
                print(f"[SERVER] {content}")
                continue
            if type == "a":
                print('sf')
                continue

            with open(content, "wb") as file:
                time.sleep(0.1)
                data = client.recv(SIZE)
                while data != b"EOF":
                    file.write(data)
```

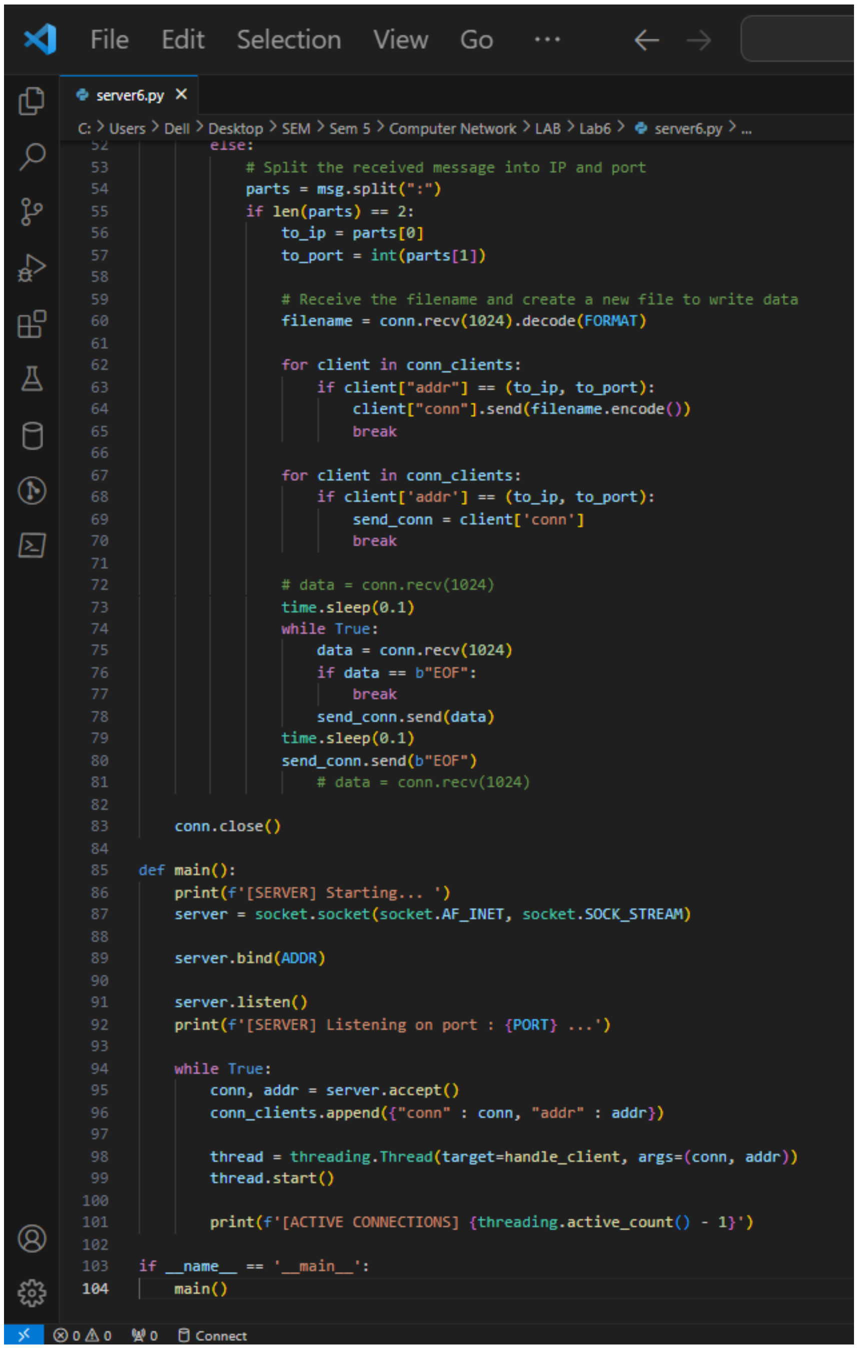


```python
                continue

            with open(content, "wb") as file:
                time.sleep(0.1)
                data = client.recv(SIZE)
                while data != b"EOF":
                    file.write(data)
                    data = client.recv(SIZE)
                # time.sleep(0.1)

    print("Closing connection...")
    client.close()

def main():
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect(ADDR)

    print(f'[CONNECTED] Connected to server on {IP}:{PORT}')

    thread = threading.Thread(target=recv_msg, args=(client,))
    thread.start()

    connected = True

    while connected:
        msg = input('> Enter IP: ')
        port = input('> Enter Port: ')

        if msg == DISCONNECT_MESSAGE:
            print(f"[DISCONNECTED] Disconnected from {IP}:{PORT}")
            client.send(msg.encode())
            break
        else:
            # Send IP and port as separate strings
            client.send(f"{msg}:{port}".encode(FORMAT))

        filename = input('> Enter File Name ')
        client.send(f"sf:{filename}".encode(FORMAT))

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

    client.close()

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

Server Code:


File
Edit
Selection
View
Go
server6.py
C: > Users > Dell > Desktop > SEM > Sem 5 > Computer Network > LAB > Lab6 > server6.py > ...
⊗ 0 ⚠ 0
0
Connect

```python
import socket
import threading
import time

IP = socket.gethostbyname(socket.gethostname())
PORT = 5657
ADDR = (IP, PORT)
FORMAT = "utf-8"
SIZE = 1024
DISCONNECT_MSG = "DISCONNECT!"

conn_clients = []

def send_file(conn, filename):
    try:
        with open(filename, "rb") as file:
            data = file.read(SIZE)
            while data:
                conn.send(data)
                data = file.read(SIZE)
        conn.send(b"EOF")
    except FileNotFoundError:
        print(f"[ERROR] File '{filename}' not found.")

def handle_client(conn, addr):
    print(f'[NEW CONNECTION] {addr} Connected!')

    for client in conn_clients:
        if client['addr'] != addr:
            conn.send(f"s:[EXISTING CONNECTION]... {client['addr'][0]} {client['addr'][1]} connected to the server".encode(FORMAT))

    for client in conn_clients:
        if client['addr'] != addr:
            client['conn'].send(f"s:\n[NEW CONNECTION]...{addr[0]} {addr[1]}".encode(FORMAT))

    connected = True

    while connected:
        msg = conn.recv(1024).decode(FORMAT)

        if msg == DISCONNECT_MSG:
            connected = False
            for client in conn_clients:
                if client['addr'] != addr:
                    client['conn'].send(f"s:\n[DISCONNECTED]...{addr[0]} {addr[1]} from the server".encode(FORMAT))
            for client in conn_clients:
                if client["addr"] == addr:
                    conn_clients.remove(client)
                    break
            else:
                print(f'[ERROR] Cannot Disconnect {addr}')
        else:
            # Split the received message into IP and port
            parts = msg.split(":")
```


File
Edit
Selection
View
Go
server6.py
C: > Users > Dell > Desktop > SEM > Sem 5 > Computer Network > LAB > Lab6 > server6.py > ...
Connect

```python
        else:
            # Split the received message into IP and port
            parts = msg.split(":")
            if len(parts) == 2:
                to_ip = parts[0]
                to_port = int(parts[1])

                # Receive the filename and create a new file to write data
                filename = conn.recv(1024).decode(FORMAT)

                for client in conn_clients:
                    if client["addr"] == (to_ip, to_port):
                        client["conn"].send(filename.encode())
                        break

                for client in conn_clients:
                    if client['addr'] == (to_ip, to_port):
                        send_conn = client['conn']
                        break

                # data = conn.recv(1024)
                time.sleep(0.1)
                while True:
                    data = conn.recv(1024)
                    if data == b"EOF":
                        break
                    send_conn.send(data)
                time.sleep(0.1)
                send_conn.send(b"EOF")
                    # data = conn.recv(1024)

    conn.close()

def main():
    print(f'[SERVER] Starting... ')
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    server.bind(ADDR)

    server.listen()
    print(f'[SERVER] Listening on port : {PORT} ...')

    while True:
        conn, addr = server.accept()
        conn_clients.append({"conn" : conn, "addr" : addr})

        thread = threading.Thread(target=handle_client, args=(conn, addr))
        thread.start()

        print(f'[ACTIVE CONNECTIONS] {threading.active_count() - 1}')

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