

(1) Your program uses IP Addresses to sort files from clients. What are the limitations and potential problems with this system? How would you improve upon it?

ans)

Using IP addresses to sort files from clients is problematic because an IP address does not reliably identify a single user or device. Multiple users can share one IP address, or a single user can have many different IP addresses over time. These inconsistencies and security issues make IP addresses an unsuitable metric for sorting files accurately

Potential Limitations include,

Inaccurate user identification: An IP address is a geographical and network identifier, not a personal one.

- **Shared connections:** All devices on a corporate network, university campus, or public Wi-Fi hotspot often appear to use the same public IP address. As a result, files from dozens or hundreds of different people could be incorrectly sorted into the same client's folder.
- **Proxy servers and VPNs:** A client can easily mask their true IP address by using a Virtual Private Network (VPN) or proxy, which would make it appear as if their file came from a different location.
- **Dynamic IP addresses:** Most residential and mobile internet connections use dynamic IP addresses, which are temporary and change regularly


To improve this , we can add a different way to authenticate users.

1. Require user accounts: Create a system where every client has their own user account, authenticated with a username and password. This is a standard and secure way to identify individuals.
2. Generate a user ID: Assign a unique, persistent User ID to each account. This ID is permanent and does not change, making it a reliable key for sorting files

And we will then sort the files based on these UserID assigned to every user.

2) Asking for the port number on the server side , on which the server will run on.

And, asking for the server IP , and server Port on client side to connect client to server.



```
ClientTCP.py X ServerTCP.py M
Client > ClientTCP.py > ...

22
23     fileName = message
24     # Send file name to server
25     clientSocket.sendall(fileName.encode('utf-8') + b'\n')
26     fileName = message[4:]
27     # Uploading the file to server
28     with open(fileName, 'rb') as f:
29         while True:
30             bytesRead = f.read(4096)
31             if not bytesRead:
32                 f.close()
33                 break
34             clientSocket.sendall(bytesRead) # Send file data to server
35     print(f"File {fileName} sent to server.")
36
37 # handling GET

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Owner\Desktop\416_project1\Server> python3.12.exe .\ServerTCP.py
Enter port number to listen on: 12000
The server is ready to receive
Connection from 169.226.237.49 : 54486 established.

PS C:\Users\Owner\Desktop\416_project1\Client> python3.12.exe .\ClientTCP.py
Enter Server IP : 169.226.237.49
Enter Port Number : 12000
```

Being able to upload the file on the server using “PUT<filename>” command. Able to handle both, filename and full path given to filename.

The screenshot shows a VS Code editor with two tabs: 'ClientTCP.py M' and 'ServerTCP.py M'. The 'ClientTCP.py' tab is active, displaying a Python script for a client. The script handles a GET request by sending the filename to the server and then uploading the file content in 4096-byte chunks. The terminal at the bottom shows the execution of the client script, which successfully connects to the server at 169.226.237.49 and uploads two files: 'file1.txt' and 'file2.txt'.

```

22 |
23 |     fileName = message
24 |     # Send file name to server
25 |     clientSocket.sendall(fileName.encode('utf-8') + b'\n')
26 |     fileName = message[4:]
27 |     # Uploading the file to server
28 |     with open(fileName, 'rb') as f:
29 |         while True:
30 |             bytesRead = f.read(4096)
31 |             if not bytesRead:
32 |                 f.close()
33 |                 break
34 |             clientSocket.sendall(bytesRead) # Send file data to server
35 |         print(f"File {fileName} sent to server.")
36 |
37 | # handling GET

```

TERMINAL

```

PS C:\Users\Owner\Desktop\416_project1\Server> python3.12.exe .\ServerTCP.py
Enter port number to listen on: 12000
The server is ready to receive
Connection from 169.226.237.49 : 54486 established.
Receiving file: file1.txt
File file1.txt received from client.
Connection from 169.226.237.49 : 50880 established.
Receiving file: file2.txt
File file2.txt received from client.
Connection from 169.226.237.49 : 50882 established.

```

CLIENT TERMINAL

```

PS C:\Users\Owner\Desktop\416_project1\client> python3.12.exe .\ClientTCP.py
Enter Server IP : 169.226.237.49
Enter Port Number : 12000
PUT file1.txt
File file1.txt sent to server.
PUT file2.txt
File file2.txt sent to server.

```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

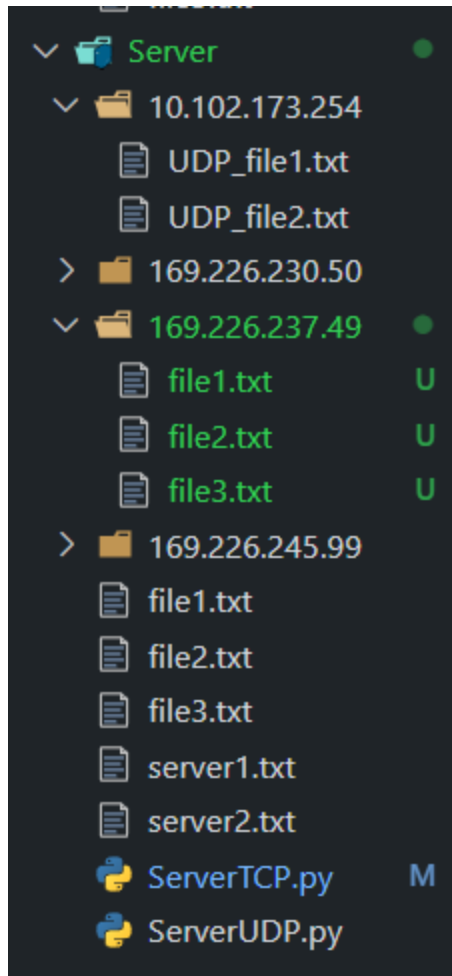
python3.12 + ▾ 🗑️ ⋮ 🔄 ✕

PS C:\Users\Owner\Desktop\416_project1\Server> python3.12.exe .\ServerTCP.py
Enter port number to listen on: 12000
The server is ready to receive
Connection from 169.226.237.49 : 54486 established.
Receiving file: file1.txt
File file1.txt received from client.
Connection from 169.226.237.49 : 50880 established.
Receiving file: file2.txt
File file2.txt received from client.
Connection from 169.226.237.49 : 50882 established.
Receiving file: file3.txt
File file3.txt received from client.
Connection from 169.226.237.49 : 49672 established.

python3.12

python3.12

Sorting all the files based on the Client IP address. As we can see ,just created a new directory for “169.226.237.49” and saved the given 3 files in that directory.



Also GET works correctly, I am able to get the desired files, by giving the full path as well as the file name.

416_PROJECT1

Client

downloads

file2.txt

server1.txt

server2.txt

ClientTCP.py

ClientUDP.py

file1.txt

file2.txt

file3.txt

Server

10.102.173.254

UDP_file1.txt

UDP_file2.txt

169.226.230.50

169.226.237.49

file1.txt

file2.txt

file3.txt

169.226.245.99

file1.txt

file2.txt

file3.txt

server1.txt

server2.txt

ServerTCP.py

ServerUDP.py

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS python3.12 + ... | [?] x
PS C:\Users\Owner\Desktop\416_project1\Server> python3.12.exe .\ServerTCP.py
Enter port number to listen on: 12000
The server is ready to receive
Connection from 169.226.237.49 : 60167 established.
Sending file: server1.txt
File server1.txt sent to client.
Connection from 169.226.237.49 : 60168 established.
Sending file: C:\Users\Owner\Desktop\416_project1\Server\server2.txt
File C:\Users\Owner\Desktop\416_project1\Server\server2.txt sent to client.
Connection from 169.226.237.49 : 60170 established.
Connection from 169.226.237.49 : 63631 established.
Sending file: C:\Users\Owner\Desktop\416_project1\Server\file2.txt
File C:\Users\Owner\Desktop\416_project1\Server\file2.txt sent to client.
Connection from 169.226.237.49 : 63633 established.

PS C:\Users\Owner\Desktop\416_project1\Client> python3.12.exe .\ClientTCP.py
Enter Server IP : 169.226.237.49
Enter Port Number : 12000
GET server1.txt
File GET server1.txt received from server.
GET C:\Users\Owner\Desktop\416_project1\Server\server2.txt
File GET C:\Users\Owner\Desktop\416_project1\Server\server2.txt received from server.
get C:\Users\Owner\Desktop\416_project1\Server\file2.txt
GET C:\Users\Owner\Desktop\416_project1\Server\file2.txt
File GET C:\Users\Owner\Desktop\416_project1\Server\file2.txt received from server.
[]
```