

Project Statement: **Hybrid P2P Chatting and File Sharing Application.****Introduction:**

This project is Linux based hybrid peer-to-peer chatting and file sharing application programmed on C++. It involves the use of an Assistor Server, which only helps in keeping records of names of connected peers, their shared directory names, shared file(s) names, IP addresses and port numbers. It helps peers in chatting and downloading files, by providing one peer, the IP address and port number of the other peer, so that they can directly communicate with each other.

Application Design and Workflow:

Server-side:

- Server has a struct Client which stores username, shared directory name, shared file(s) names, IP address, port number and socket integer.
- There is a vector clients of type Struct Client, to save data of all the connected peers/clients.
- Whenever a peer connects to the server, it pass on socket integer to handleclient function in a separate thread, so that server can entertain(or accept connection from) other peers too.
- In handleclient function, server receives all the information of peer (e.g username, shared directory name, shared file(s) names, ip address and port number), and store them in the clients vector.
- After storing the received information, server asks peer 5 choices: 1 for receiving names of all the shared files in the network, 2 for searching any file present in the network, 3 for downloading any file from the network directly from the peer, 4 for start chatting directly with a peer, and 5 for exiting the network.
- On the client/peer side, when peer enters a valid choice number, that choice number is sent to server to entertain that peer accordingly.
- If a peer wants to know about all the shared files in the network, the server sends the names of all the files in the network along with the names of the peers who are holding those files.
- If a peer wants to search a file in the network, the server asks the peer, file name they want to search. When the server gets the file name, it

searches all its database and sends the search results along with the names of peers, who are holding those searched files.

- If a peer wants to chat with any other peer, server asks the name of the target-peer to whom initiator-peer wants to connect, if this name matches any of the names of peers present in server's database, server immediately sends IP address and port number of the target-peer to the initiator-peer, so that initiator-peer can directly connect to target-peer.
- If a peer wants to download any file from the network, the server asks the name of the file. If this name matches any of the names of shared files present in the server's database, it sends back the search results to the peer along with the names of peers who have that file. Then the server asks the peer to enter the username of the peer from who they want to download file. When peer gives that name, server immediately sends the IP address and the port number of the target-peer from who to the initiator-peer wants to download the file. So that initiator-peer can directly connect with target-peer to download the file.
- The Assistor server can entertain any number of peers joining the network.

Client/Peer-side:

- Client/Peer first connects with server and sends its username and shared directory name.
- When peer enters shared directory name there is a code snippet which automatically makes a string containing names of files inside that shared directory and send that string to server after sending shared directory name. Server on the other side automatically parse received string and store shared file(s) names in its database.
- Peer enters its IP address and port number to bind a dedicated socket for accepting connections from other peers.
- There is a separate thread running for accepting connections from other peers.
- Server asks 5 choices to the peer which I had explained above in server-side explanation.

Input validations are also implemented in the code*.

Project Deliverables:

All the required deliverables are full filled in this project:

- ✓ When a peer connects to the network, the details about all the files in the client's shared directory are shared with the community/network. As any member(peer) in the community can request the details of all the files present in the network.
- ✓ Any peer in the network can search and download any file from the network.
- ✓ Any peer in the network can do full-duplex chat with any other peer in the network.
- ✓ There can be any number of peers/members in the network/community.

Results:

Server side

```
● faizan@faizan-VirtualBox:~/Downloads/CNproject$ g++ serverside.cpp -o server
○ faizan@faizan-VirtualBox:~/Downloads/CNproject$ sudo ./server
[sudo] password for faizan:
Faizan's P2P chatting and file sharing Server LIVE. listening for connections on port 1234
faizan
sharedDirectoryOne
Received file name: lab6cn.txt
Received file name: lab4cn.txt
127.0.0.1
10
Sender IP Address: 127.0.0.1
Sender Port: 10
jack
sharedDirectoryTwo
Received file name: lab7cn.txt
Received file name: lab8cn.txt
127.0.0.3
15
Sender IP Address: 127.0.0.3
Sender Port: 15
case : 1
case : 2

query : lab8cn.txt

query (inside function) : lab8cn.txt
sent search results
case : 2

query : lab4cn.txt

query (inside function) : lab4cn.txt
sent search results
case : 1
case : 4
□
```

Peers side

Peer # 1	Peer # 2
<p>(receive all file(s) names and search file)</p> <pre> faizan@faizan-VirtualBox:~/Downloads/CNproject\$ g++ client.cpp -o client faizan@faizan-VirtualBox:~/Downloads/CNproject\$ sudo ./client [sudo] password for faizan: Enter your username: faizan Enter your shared directory: sharedDirectoryOne Enter your Public IP for future P2P connections : 127.0.0.1 Enter the Port Number, on which you will be accepting future P2P connections : 10 ===== 1. List shared files 2. Search for a file 3. Download a file 4. Chat with a user 5. Exit Enter your choice : 1 You entered a number: 1 Shared File: faizan : lab6cn.txt Shared File: faizan : lab4cn.txt Shared File: jack : lab7cn.txt Shared File: jack : lab8cn.txt ===== 1. List shared files 2. Search for a file 3. Download a file 4. Chat with a user 5. Exit Enter your choice : 2 You entered a number: 2 Enter the file you want to search for: lab8cn.txt u want to search : lab8cn.txt searched : jack: lab8cn.txt, Search results: jack: lab8cn.txt ===== 1. List shared files 2. Search for a file 3. Download a file 4. Chat with a user 5. Exit Enter your choice : █ </pre>	<p>(search file and receive all file(s) names)</p> <pre> faizan@faizan-VirtualBox:~/Downloads/CNproject\$ g++ client.cpp -o client faizan@faizan-VirtualBox:~/Downloads/CNproject\$ sudo ./client [sudo] password for faizan: Enter your username: jack Enter your shared directory: sharedDirectoryTwo Enter your Public IP for future P2P connections : 127.0.0.3 Enter the Port Number, on which you will be accepting future P2P connections : 15 ===== 1. List shared files 2. Search for a file 3. Download a file 4. Chat with a user 5. Exit Enter your choice : 2 You entered a number: 2 Enter the file you want to search for: lab4cn.txt u want to search : lab4cn.txt searched : faizan: lab4cn.txt, Search results: faizan: lab4cn.txt ===== 1. List shared files 2. Search for a file 3. Download a file 4. Chat with a user 5. Exit Enter your choice : 1 You entered a number: 1 Shared File: faizan : lab6cn.txt Shared File: faizan : lab4cn.txt Shared File: jack : lab7cn.txt Shared File: jack : lab8cn.txt ===== 1. List shared files 2. Search for a file 3. Download a file 4. Chat with a user 5. Exit Enter your choice : █ </pre>
<p>(chat with peer 2)</p> <pre> Enter your choice : 4 You entered a number: 4 Following is the list of usernames of all the peers in the network : User name : faizan User name : jack Enter the username you want to chat with: jack u entered username : jack Received otherperson's IP : 127.0.0.3 Received otherperson's Port number : 15 Start chatting (type 'exit' to end): Enter Message for 'jack': Enter Message for 'jack': hello jack. how are you? Enter Message for 'jack': Received Message from 'jack': im fine faizan. what are you doing? Enter Message for 'jack': im just busy in making computer networks project. Enter Message for 'jack': Received Message from 'jack': oh nice. have you completed it? Enter Message for 'jack': yes jack. i have completed just now. Enter Message for 'jack': Received Message from 'jack': geat. that was nice talking to you. Enter Message for 'jack': ok bye jack Enter Message for 'jack': Received Message from 'jack': bye faizan Enter Message for 'jack': exit You closed the connection! Socket closed ===== 1. List shared files 2. Search for a file 3. Download a file 4. Chat with a user 5. Exit Enter your choice : █ </pre>	<p>(chat with peer 1)</p> <pre> ===== 1. List shared files 2. Search for a file 3. Download a file 4. Chat with a user 5. Exit Enter your choice : ' faizan ' connected you for chat. Enter 'start' first to activate sending messages : Enter Message for 'faizan': Received Message from 'faizan': hello jack. how are you? Enter Message for 'faizan': start Chat Activated ! You can now send messages. start Enter Message for 'faizan': im fine faizan. what are you doing? Enter Message for 'faizan': Received Message from 'faizan': im just busy in making computer networks project. Enter Message for 'faizan': oh nice. have you completed it? Enter Message for 'faizan': Received Message from 'faizan': yes jack. i have completed just now. Enter Message for 'faizan': geat. that was nice talking to you. Enter Message for 'faizan': Received Message from 'faizan': ok bye jack Enter Message for 'faizan': bye faizan Enter Message for 'faizan': Connection was closed by 'faizan'! Enter 'exit' to close your sending thread too ! Socket closed exit You closed the connection! Error closing socket. Socket already closed. ===== 1. List shared files 2. Search for a file 3. Download a file 4. Chat with a user 5. Exit Enter your choice : █ </pre>

(download file from peer 2)


```
=====
1. List shared files
2. Search for a file
3. Download a file
4. Chat with a user
5. Exit
Enter your choice : 3
You entered a number: 3
Enter the file you want to download: lab7cn.txt

u want to download : lab7cn.txt
searched : jack: lab7cn.txt,
Search results:
jack: lab7cn.txt
Enter the username from who you want to download the required file : jack

u entered username : jack
Received otherperson's IP : 127.0.0.3
Received otherperson's Port number : 15
/home/faizan/Downloads/CNproject/sharedDirectoryOne/lab7cn.txt
File ' lab7cn.txt ' received successfully !
=====
1. List shared files
2. Search for a file
3. Download a file
4. Chat with a user
5. Exit
Enter your choice : 0
```

Peer1's directory (name: sharedDirectoryOne)
(before receiving required file)

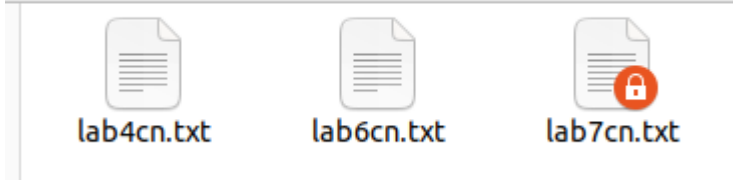
oads / CNproject / sharedDirectoryOne



lab4cn.txt lab6cn.txt

(after receiving required file)

oads / CNproject / sharedDirectoryOne



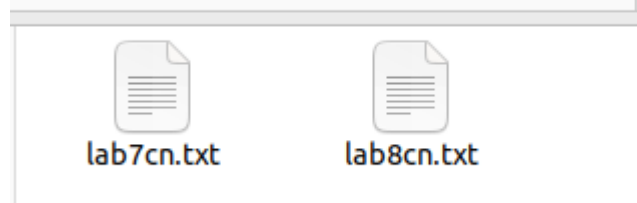
lab4cn.txt lab6cn.txt lab7cn.txt

(send file too peer 1)

```
=====
1. List shared files
2. Search for a file
3. Download a file
4. Chat with a user
5. Exit
Enter your choice : /home/faizan/Downloads/CNproject/sharedDirectoryTwo/lab7cn.txt
File ' lab7cn.txt ' sent successfully !
```

Peer2's directory (name: sharedDirectoryTwo)

oads / CNproject / sharedDirectoryTwo



lab7cn.txt lab8cn.txt

(Peer2's file : 'lab7cn.txt' content)

```

1 computer networks lab number 7
2
3 Creating and Running a Hello Word Thread
4
5 This exercise will guide you about developing a simple hello word thread by using POSIX threads (pthreads) API. To use POSIX
  threads, you must first include the pthread.h header file. Once the headers have been included, you can create a thread by
  calling the pthread_create function. The pthread_create function takes the function name to be run as a separate thread, and
  starts an independent thread. Suppose, we have to run a helloPrint function AF_INET parameter to the socket() call indicates
  that the Internet address family (IPv4) is to be used; AF_INET6 would create an IPv6 socket.
6 SOCK_STREAM parameter indicates that a TCP stream socket is desired.
7 Third parameter (protocol) is not used with TCP or UDP sockets, and is set to zero.
8 Return Value: If an error occurs, -1 is returned and the global variable errno is set to indicate the type of error. If a
  positive number is returned, it acts as descriptor (file handler).
9 Hint: Type man socket on linux shell to check more details about socket
  
```

(Peer1 received a file recently : 'lab7cn.txt' content)

```

1 computer networks lab number 7
2
3 Creating and Running a Hello Word Thread
4
5 This exercise will guide you about developing a simple hello word thread by using POSIX threads (pthreads) API. To use POSIX
  threads, you must first include the pthread.h header file. Once the headers have been included, you can create a thread by
  calling the pthread_create function. The pthread_create function takes the function name to be run as a separate thread, and
  starts an independent thread. Suppose, we have to run a helloPrint function AF_INET parameter to the socket() call indicates
  that the Internet address family (IPv4) is to be used; AF_INET6 would create an IPv6 socket.
6 SOCK_STREAM parameter indicates that a TCP stream socket is desired.
7 Third parameter (protocol) is not used with TCP or UDP sockets, and is set to zero.
8 Return Value: If an error occurs, -1 is returned and the global variable errno is set to indicate the type of error. If a
  positive number is returned, it acts as descriptor (file handler).
9 Hint: Type man socket on linux shell to check more details about socket
  
```

(Peer 1 Disconnecting with the network by pressing 5)

```

/home/faizan/Downloads/CNproject/sharedDirectoryOne/lab7cn.txt
File ' lab7cn.txt ' received successfully !
=====
1. List shared files
2. Search for a file
3. Download a file
4. Chat with a user
5. Exit
Enter your choice : 5
You entered a number: 5
terminate called without an active exception
Aborted
faizan@faizan-VirtualBox:~/Downloads/CNproject$ 
  
```

(Peer 2 Disconnecting with the network by pressing 5)

```

Enter your choice : /home/faizan/Downloads/CNproject/sharedDirectoryTwo/lab7cn.txt
File ' lab7cn.txt ' sent successfully !
1
You entered a number: 1
Shared File: faizan : lab6cn.txt
Shared File: faizan : lab4cn.txt
Shared File: jack : lab7cn.txt
Shared File: jack : lab8cn.txt
=====
1. List shared files
2. Search for a file
3. Download a file
4. Chat with a user
5. Exit
Enter your choice : 5
You entered a number: 5
terminate called without an active exception
Aborted
faizan@faizan-VirtualBox:~/Downloads/CNproject$ 
  
```

Server stays LIVE :

```

sent search results
case : 1
case : 4
case : 3

query (inside function) : lab7cn.txt
sent search results
case : 1
case : 5
Invalid request type.
case : 0
case : 5
Invalid request type.
case : 0
  
```