Computer Networks
(Course Project)

P2PConnect
Peer to Peer File Sharing System

● **Team Members**

Borkar Ankur Motiram (B19CSE025)
Harpal Sahil Santosh (B19CSE107)
Sakhare Shriyog Dattatray (B19CSE074)

Index

Motivation

The main goal of this project is to create a peer-to-peer system that consists of computers that operate as a distributed file repository and allows computers to seek for and download files from one another.
Using a P2P network we can share different types of files between 2 systems without

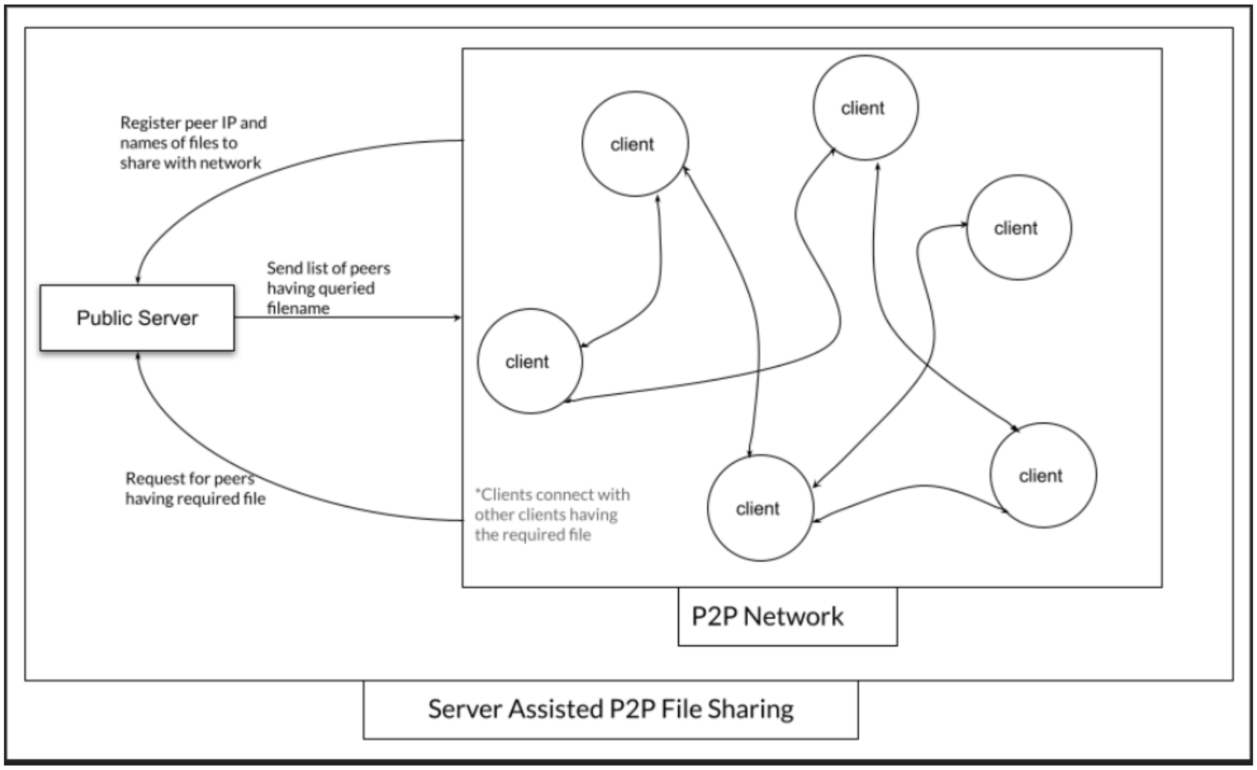
the involvement of a central authority or server. Every user/client in the network is known as a peer. One peer sends requests to others by making TCP or UDP connections.

Server Assisted P2P file sharing

Introduction

It is a client-server architecture in that it keeps a large central server to provide directory service. All of the peers notify the central server of their IP address and the files they are sharing. The server checks in with the peers at regular intervals to see if they are still connected. So, basically, this server keeps a massive database of which files are present at which IP addresses.

Working


Register peer IP and names of files to share with network
Send list of peers having queried filename
Public Server
Request for peers having required file
client
client
client
client
client
client
client
*Clients connect with other clients having the required file
P2P Network
Server Assisted P2P File Sharing

- When a requesting peer connects, it now sends its query to the server.
- Since the server has all of its peers' information, it returns to the peer the IP addresses of all the peers who have the requested file.
- The file transfer is now taking place between these two peers.

Advantages:

- Memory efficient for clients
- Easier communication protocol for peers

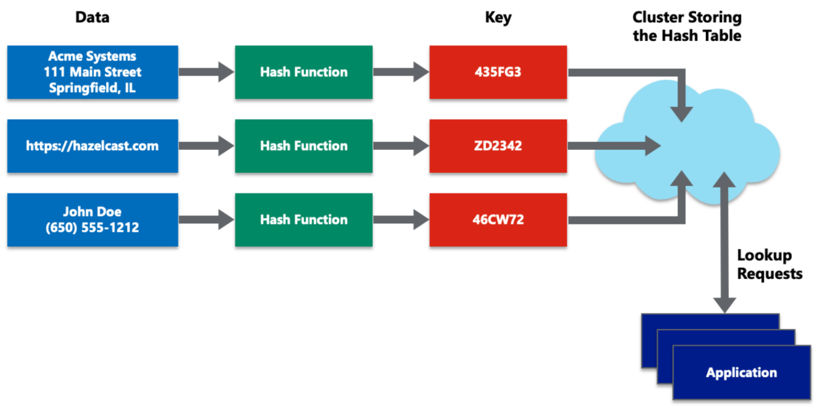
- Easy to set up for Local Area Networks.

Disadvantage:
- Single Point of Failure (SPOF)
- Difficult to handle connection errors due to consistency constraints.
- The server forgets about the file availability status for a disconnected peer.
- Storage space is not scalable.

Possible Optimisation
- Use of persistent data storage to track offline nodes.
- Making nodes responsible for notifying deleted files so as to avoid future problems.

Peer to Peer File Sharing Using Naive DHT Querying

What is DHT?
DHT => Distributed Hash Table
In DHT we store data in the form of Key-Value pairs and it is distributed across the network.
Properties of DHT:
- Decentralized
- Fault-Tolerant
- Scalable



What is getting stored in the database?
Let's take an example of a simple database as shown below.

Original Key	Key	Value
Movie1	8962458	10.255.255.255
Movie2	7800356	100.25.77.20
Movie3	1567109	80.22.25.86
Movie4	2360012	11.89.64.138

This database contains key-value pairs. Here the key is the title of the movie and the value is the IP address of the system where that movie is located. But as we know storing and searching on numerical values are more convenient than on string values. So that's why we have the concept of hash functions. Which takes a string as input and outputs a unique numerical value. We store this key-value pair in the hash tables.

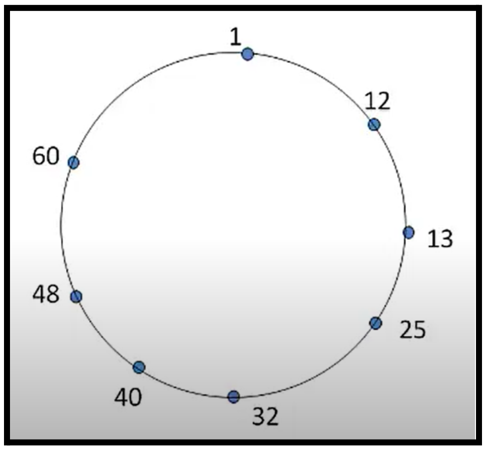
How to assign key-value pairs in DHT?

For this, we need to search for a peer having an ID closest to a value of the key and assign a key-value pair to that peer. Here closest means immediate successor of the key.
For example,
Let, ID space {0,1,2,3,....,63}
This means the hash function outputs values between 0 to 63 only.
And 8 peers in the network with IDs = {1,12,13,25,32,40,48,60}
Now,

- If the key is 51, the peer with just greater than 51 is a peer with ID 60. So, assign the key 51 and its value to a peer with ID 60
- Same for key 60 it will be 60
- If the key is greater than the max value of ID among available peers. Then we take the module by (max value of the ID of available peer+1)
  Hence for key = 61 modulo will give value 0. And closest ID of the peer is 1

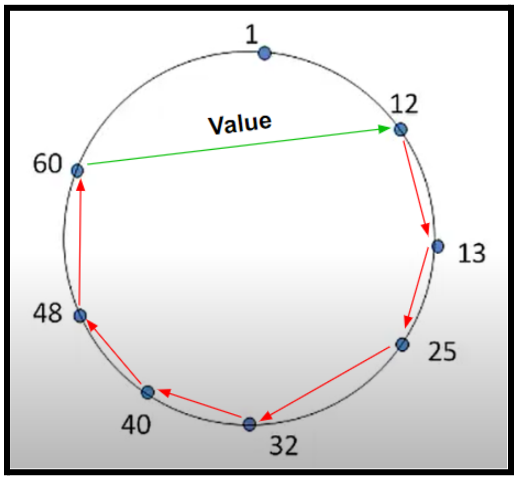
How are peers organized in DHT?

Here we put all peers in a circular manner. Peer 1 follows peer 12. Peer 12 follows peer 13 and so on. Here for node 12, peer 1 is the predecessor, and peer 13 is the successor. Each peer is only aware of the immediate successor and predecessor. This means it knows the IP of both predecessor and successor. Peers are ordered according to indices.



How to resolve the lookup query?

Let peer 12 want to know the value associated with key 53. But this key-value pair is not present at the peer 12 so in naïve DHT it sends a query to the nearest successor. Since peer 12 has an IP address of peer 13 it sends the request to peer 13. Since peer 13 also doesn't have the key 53 so it also sends a request to its successor which is peer 25 and so on. In the end, we reach peer 60 where key 53 is present. Then peer looks into its storage and retrieves the value corresponding to key 53 and sends it to the initiator of the query that is peer 12.



The complexity of this operation is **O(N)**. Where **N** = Number of peers.

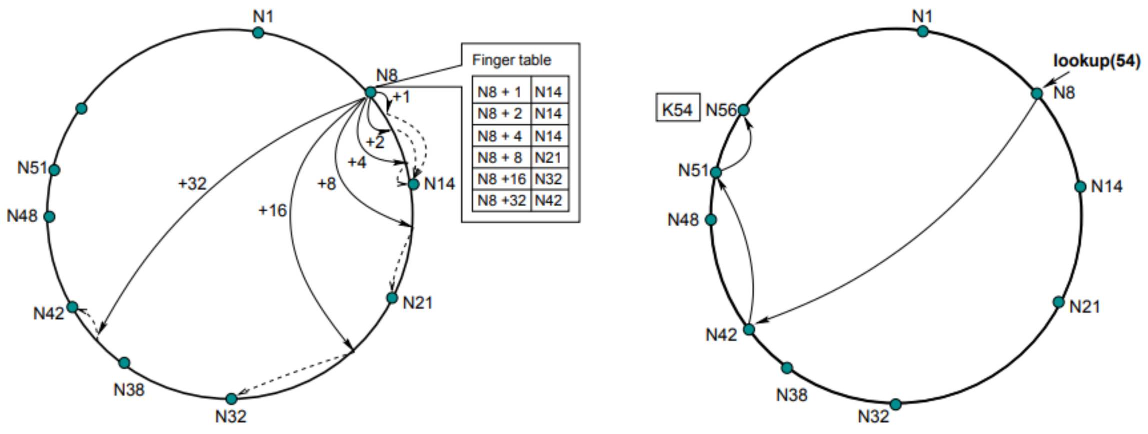
Peer-to-peer file sharing using chord DHT protocol

Introduction:

Chord DHT is a protocol used in peer-to-peer file sharing systems. In a peer-to-peer files sharing system, each node/peer serves the same purpose and there is no centralized control so the load is distributed among all the peers in the system.
Chord DHT allows faster lookups among all the distributed systems. The main specialty of the chord DHT protocol is that it does lookups in O(logn) order and every node/peer in the distributed system stores information about other logn nodes in the system in the form of a finger table. As chord DHT has the load distributed among all the nodes it does not have problems like SPOF(single point of failure) and as it does lookups in O(logn) it is very efficient as compared to the Naive DHT protocol.

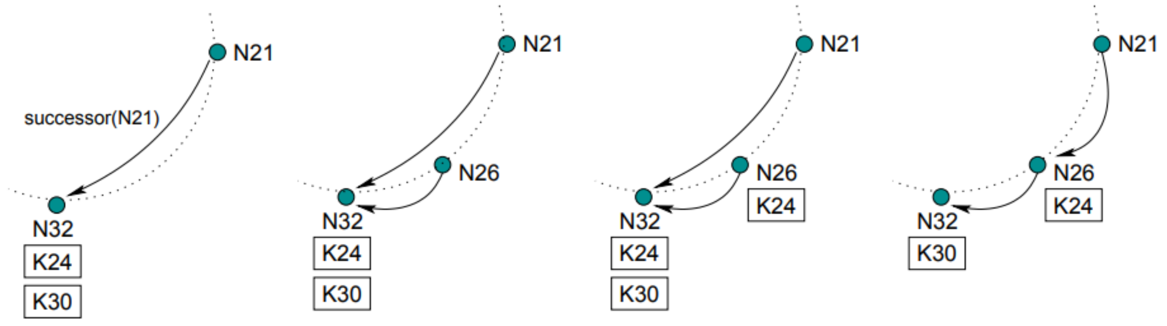
Finger table:

- The finger table is nothing but additional routing information just to carry out faster lookups
- Each finger table contains m entries where m is the no. of bits in the SHA key.
- The ith entry in the finger table of the nth node which is also called the ith finger is$(n+2^{i-1})$.
- finger[i] = successor $(n + 2^{i-1})$



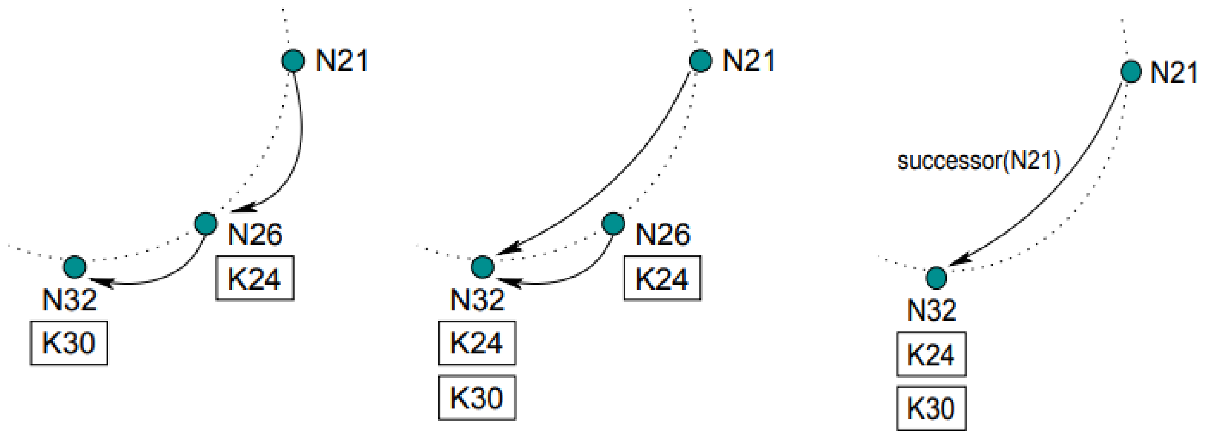
N8 + 1	N14
N8 + 2	N14
N8 + 4	N14
N8 + 8	N21
N8 +16	N32
N8 +32	N42

Stabilization: Handling joins



In the above example when node N26 joins the network then as we can see the N26 finds N32 as its successor then N26 will notify N32 then N32 accepts the node N26 as its predecessor now N26 copies keys which should be in N26.now node N21 ask its successor N32 for its predecessor and N32 will return N26 so now N21 will updates its successor as N26 and N26 updates its predecessor as N21.and this is how nodes are stabilized when a new node join the network.

Stabilization: Handling leaves



When any node voluntarily wants to leave the network then it simply copies all the records contained in the node to its successor node and leaves the network after that its successor node updates its predecessor and its predecessor node updates its successor node. and all the corresponding finger tables also get updated.

Advantages and disadvantages of Chord DHT protocol:

Pros	**Cons**
Scalable without disturbance of joining nodes or leaving nodes	A very rare chance of data inconsistency in case of a large-scale simultaneous failure of nodes.
High decentralization means more autonomy.	The lack of central authority makes it difficult to control leechers and freeloaders.

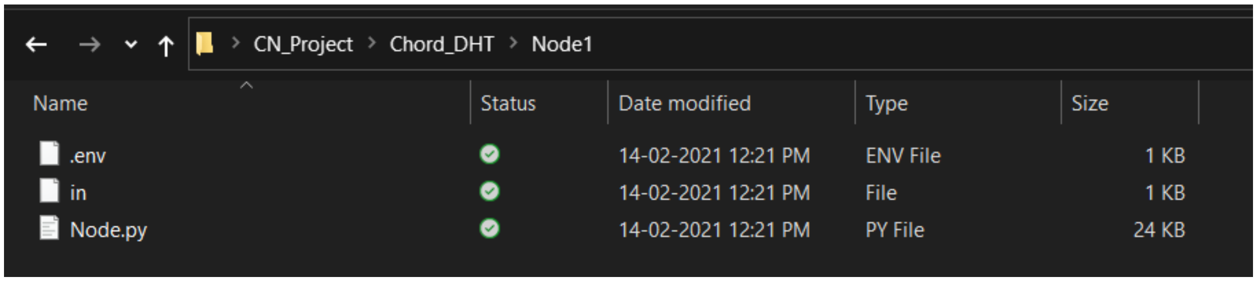
Achieves maximum tolerance to data loss yet has minimal data replications.	

Output

Chord DHT

In our connection, we have three Nodes.

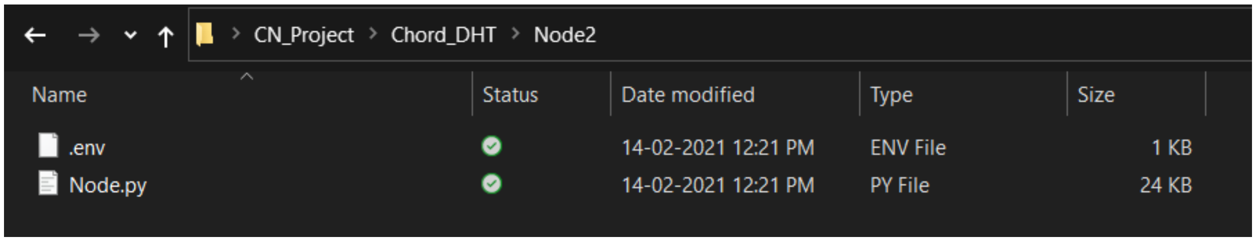
- Node 1



CN_Project › Chord_DHT › Node1

Name	Status	Date modified	Type	Size
.env	✓	14-02-2021 12:21 PM	ENV File	1 KB
in	✓	14-02-2021 12:21 PM	File	1 KB
Node.py	✓	14-02-2021 12:21 PM	PY File	24 KB

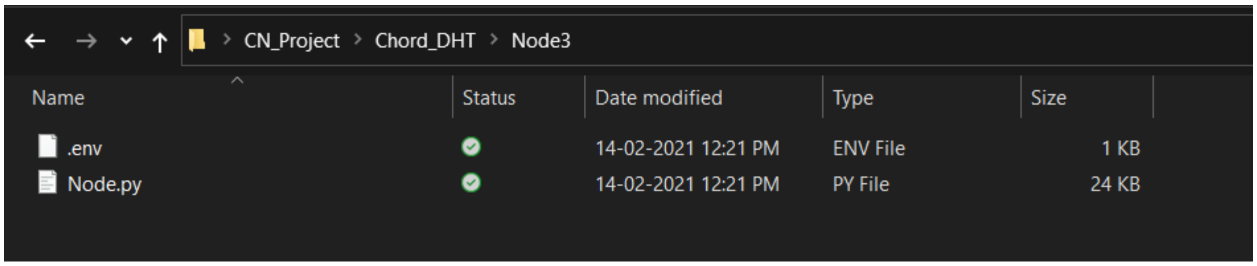
- Node 2



CN_Project › Chord_DHT › Node2

Name	Status	Date modified	Type	Size
.env	✓	14-02-2021 12:21 PM	ENV File	1 KB
Node.py	✓	14-02-2021 12:21 PM	PY File	24 KB

- Node 3

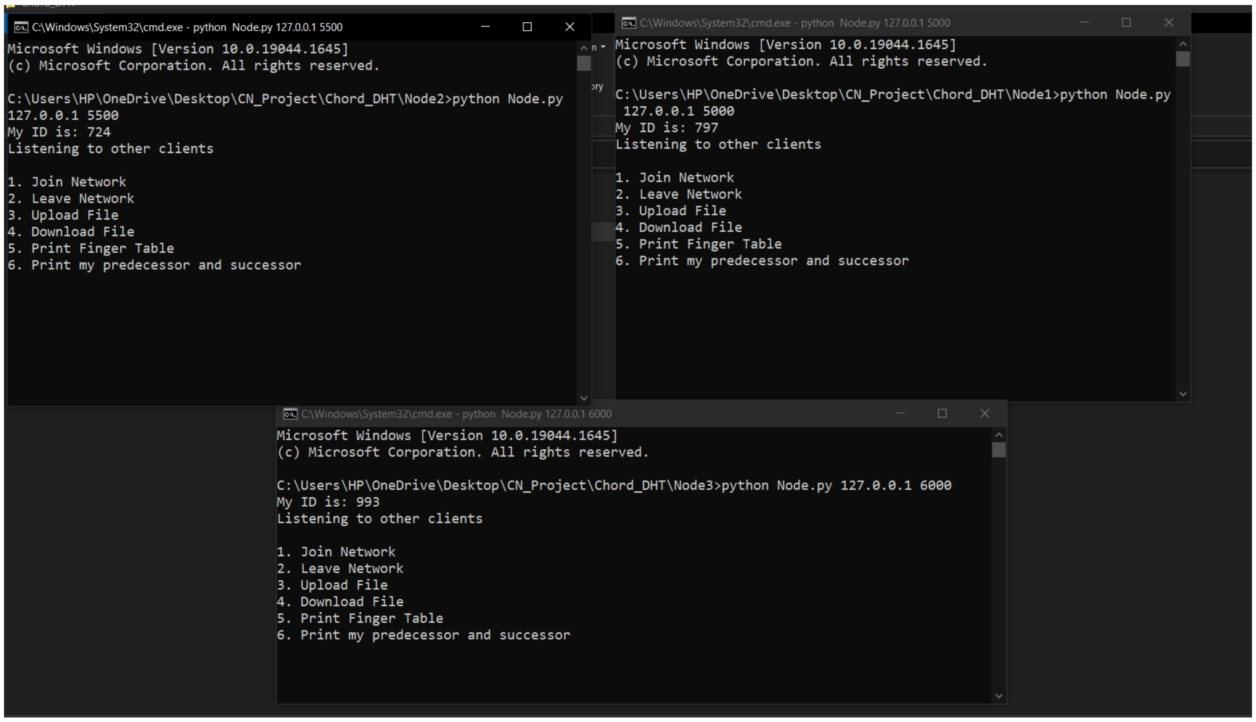


CN_Project › Chord_DHT › Node3

Name	Status	Date modified	Type	Size
.env	✓	14-02-2021 12:21 PM	ENV File	1 KB
Node.py	✓	14-02-2021 12:21 PM	PY File	24 KB

Initialize 3 different peers by running the following commands on 3 different terminals.
For Node 1: python Node.py 127.0.0.1 5000
For Node 2: python Node.py 127.0.0.1 5500

For Node 3: python Node.py 127.0.0.1 6000

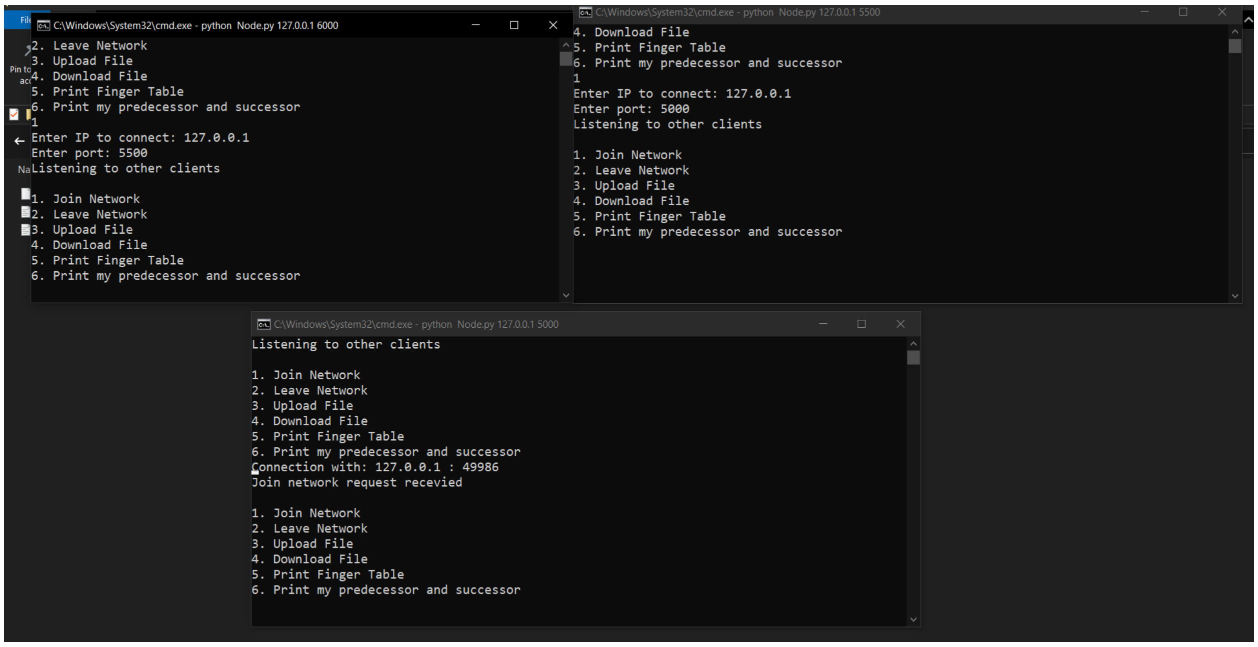


```
C:\Windows\System32\cmd.exe - python Node.py 127.0.0.1 5500
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP\OneDrive\Desktop\CN_Project\Chord_DHT\Node2>python Node.py
127.0.0.1 5500
My ID is: 724
Listening to other clients

1. Join Network
2. Leave Network
3. Upload File
4. Download File
5. Print Finger Table
6. Print my predecessor and successor
```

```
C:\Windows\System32\cmd.exe - python Node.py 127.0.0.1 5000
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP\OneDrive\Desktop\CN_Project\Chord_DHT\Node1>python Node.py
 127.0.0.1 5000
My ID is: 797
Listening to other clients

1. Join Network
2. Leave Network
3. Upload File
4. Download File
5. Print Finger Table
6. Print my predecessor and successor
```

```
C:\Windows\System32\cmd.exe - python Node.py 127.0.0.1 6000
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP\OneDrive\Desktop\CN_Project\Chord_DHT\Node3>python Node.py 127.0.0.1 6000
My ID is: 993
Listening to other clients

1. Join Network
2. Leave Network
3. Upload File
4. Download File
5. Print Finger Table
6. Print my predecessor and successor
```

Node 3 is connected to Node 2.
Node 2 is connected to Node 1.

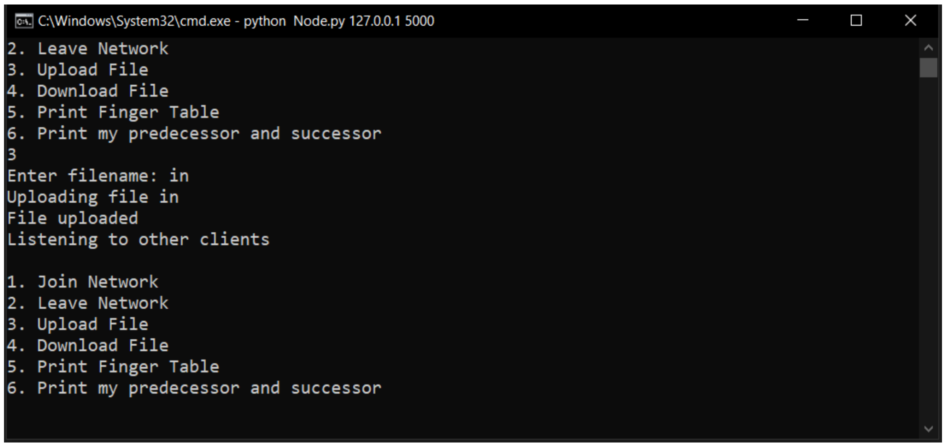


```
C:\Windows\System32\cmd.exe - python Node.py 127.0.0.1 6000
2. Leave Network
3. Upload File
4. Download File
5. Print Finger Table
6. Print my predecessor and successor
1
Enter IP to connect: 127.0.0.1
Enter port: 5500
Listening to other clients

1. Join Network
2. Leave Network
3. Upload File
4. Download File
5. Print Finger Table
6. Print my predecessor and successor
```

```
C:\Windows\System32\cmd.exe - python Node.py 127.0.0.1 5500
4. Download File
5. Print Finger Table
6. Print my predecessor and successor
1
Enter IP to connect: 127.0.0.1
Enter port: 5000
Listening to other clients

1. Join Network
2. Leave Network
3. Upload File
4. Download File
5. Print Finger Table
6. Print my predecessor and successor
```

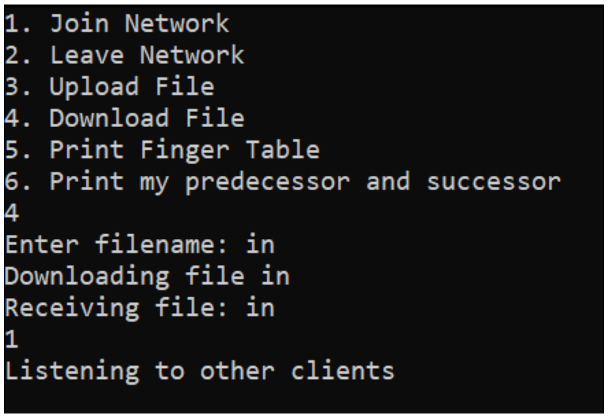
```
C:\Windows\System32\cmd.exe - python Node.py 127.0.0.1 5000
Listening to other clients

1. Join Network
2. Leave Network
3. Upload File
4. Download File
5. Print Finger Table
6. Print my predecessor and successor
Connection with: 127.0.0.1 : 49986
Join network request recevied

1. Join Network
2. Leave Network
3. Upload File
4. Download File
5. Print Finger Table
6. Print my predecessor and successor
```

File uploaded using Node 1
File Name: **in**



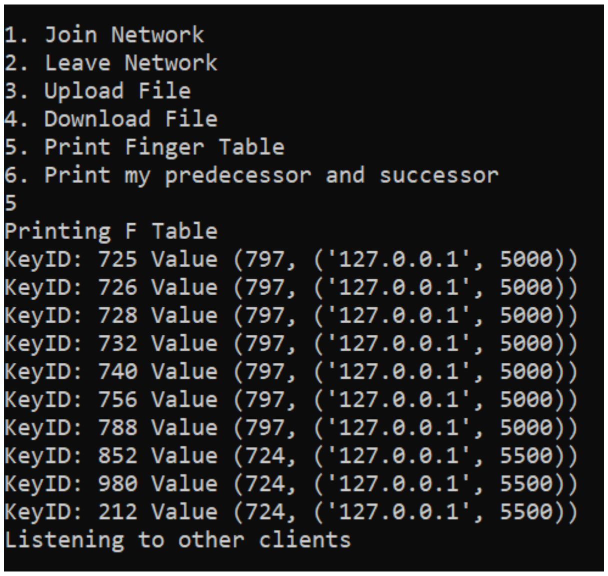
```
C:\Windows\System32\cmd.exe - python  Node.py 127.0.0.1 5000
2. Leave Network
3. Upload File
4. Download File
5. Print Finger Table
6. Print my predecessor and successor
3
Enter filename: in
Uploading file in
File uploaded
Listening to other clients

1. Join Network
2. Leave Network
3. Upload File
4. Download File
5. Print Finger Table
6. Print my predecessor and successor
```

File download by Node 3



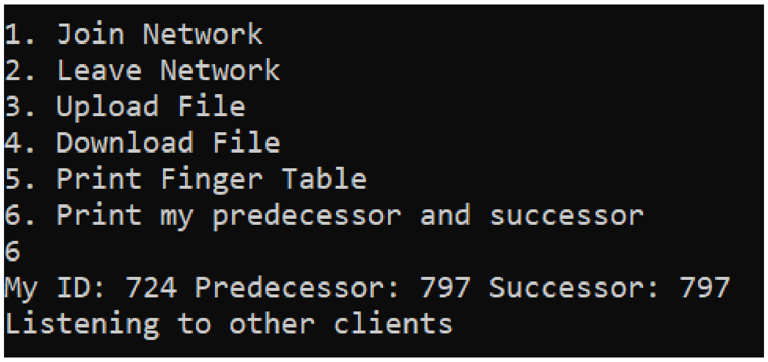
```
1. Join Network
2. Leave Network
3. Upload File
4. Download File
5. Print Finger Table
6. Print my predecessor and successor
4
Enter filename: in
Downloading file in
Receiving file: in
1
Listening to other clients
```

Print finger table of Node 2



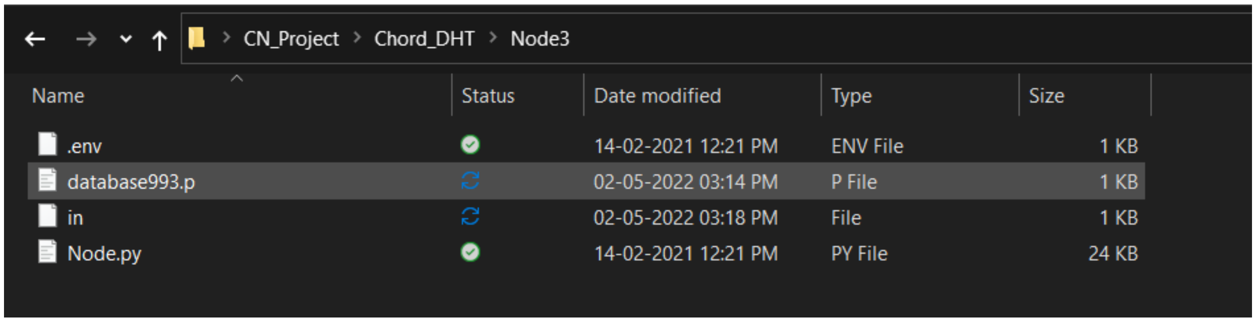
```
1. Join Network
2. Leave Network
3. Upload File
4. Download File
5. Print Finger Table
6. Print my predecessor and successor
5
Printing F Table
KeyID: 725 Value (797, ('127.0.0.1', 5000))
KeyID: 726 Value (797, ('127.0.0.1', 5000))
KeyID: 728 Value (797, ('127.0.0.1', 5000))
KeyID: 732 Value (797, ('127.0.0.1', 5000))
KeyID: 740 Value (797, ('127.0.0.1', 5000))
KeyID: 756 Value (797, ('127.0.0.1', 5000))
KeyID: 788 Value (797, ('127.0.0.1', 5000))
KeyID: 852 Value (724, ('127.0.0.1', 5500))
KeyID: 980 Value (724, ('127.0.0.1', 5500))
KeyID: 212 Value (724, ('127.0.0.1', 5500))
Listening to other clients
```

Print my predecessor and successor



```
1. Join Network
2. Leave Network
3. Upload File
4. Download File
5. Print Finger Table
6. Print my predecessor and successor
6
My ID: 724 Predecessor: 797 Successor: 797
Listening to other clients
```

After successfully downloading the file **in Node** 3, we can see that file in the directory Node3.



Name	Status	Date modified	Type	Size
.env	✓	14-02-2021 12:21 PM	ENV File	1 KB
database993.p	⟳	02-05-2022 03:14 PM	P File	1 KB
in	⟳	02-05-2022 03:18 PM	File	1 KB
Node.py	✓	14-02-2021 12:21 PM	PY File	24 KB

Server assisted

Initialize server by running the following command on terminals.

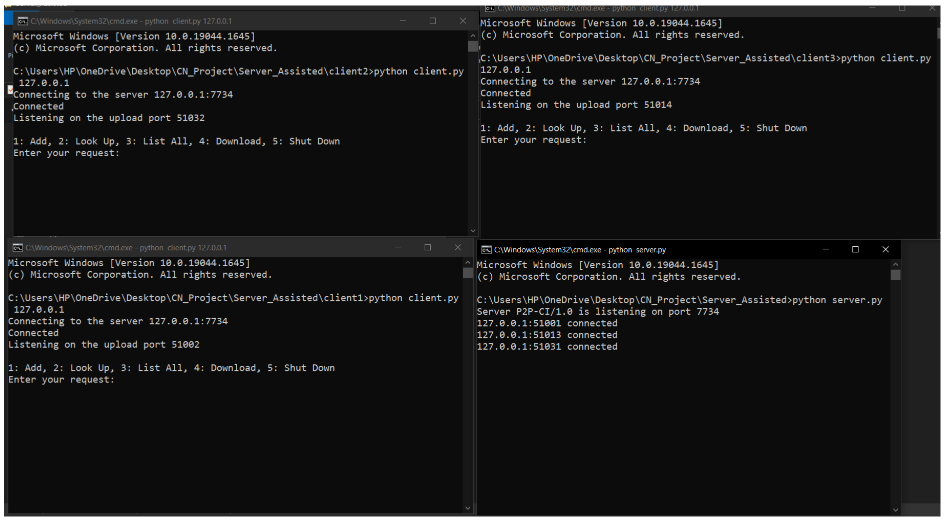
server: python server.py

Initialize 3 different client by running the following commands on 3 different terminals.

client 1: python client.py 127.0.0.1

client 2: python client.py 127.0.0.1

client 3: python client.py 127.0.0.1



```
C:\Windows\System32\cmd.exe - python  client.py 127.0.0.1
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP\OneDrive\Desktop\CN_Project\Server_Assisted\client2>python client.py
 127.0.0.1
Connecting to the server 127.0.0.1:7734
Connected
Listening on the upload port 51032

1: Add, 2: Look Up, 3: List All, 4: Download, 5: Shut Down
Enter your request:
```

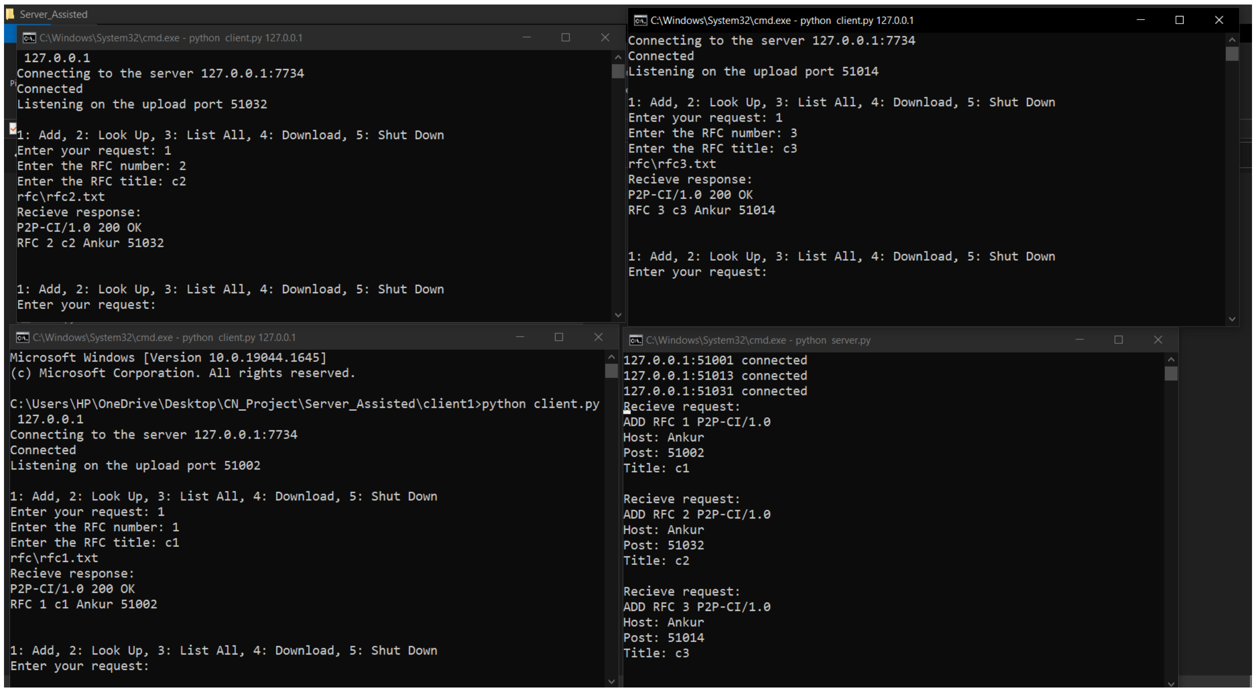
```
C:\Windows\System32\cmd.exe - python  client.py 127.0.0.1
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP\OneDrive\Desktop\CN_Project\Server_Assisted\client3>python client.py
127.0.0.1
Connecting to the server 127.0.0.1:7734
Connected
Listening on the upload port 51014

1: Add, 2: Look Up, 3: List All, 4: Download, 5: Shut Down
Enter your request:
```

```
C:\Windows\System32\cmd.exe - python  client.py 127.0.0.1
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP\OneDrive\Desktop\CN_Project\Server_Assisted\client1>python client.py
 127.0.0.1
Connecting to the server 127.0.0.1:7734
Connected
Listening on the upload port 51002

1: Add, 2: Look Up, 3: List All, 4: Download, 5: Shut Down
Enter your request:
```

```
C:\Windows\System32\cmd.exe - python  server.py
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP\OneDrive\Desktop\CN_Project\Server_Assisted>python server.py
Server P2P-CI/1.0 is listening on port 7734
127.0.0.1:51001 connected
127.0.0.1:51013 connected
127.0.0.1:51031 connected
```

After uploading file:

- rfc1.txt by client 1
- rfc2.txt by client 2
- rfc3.txt by client 3



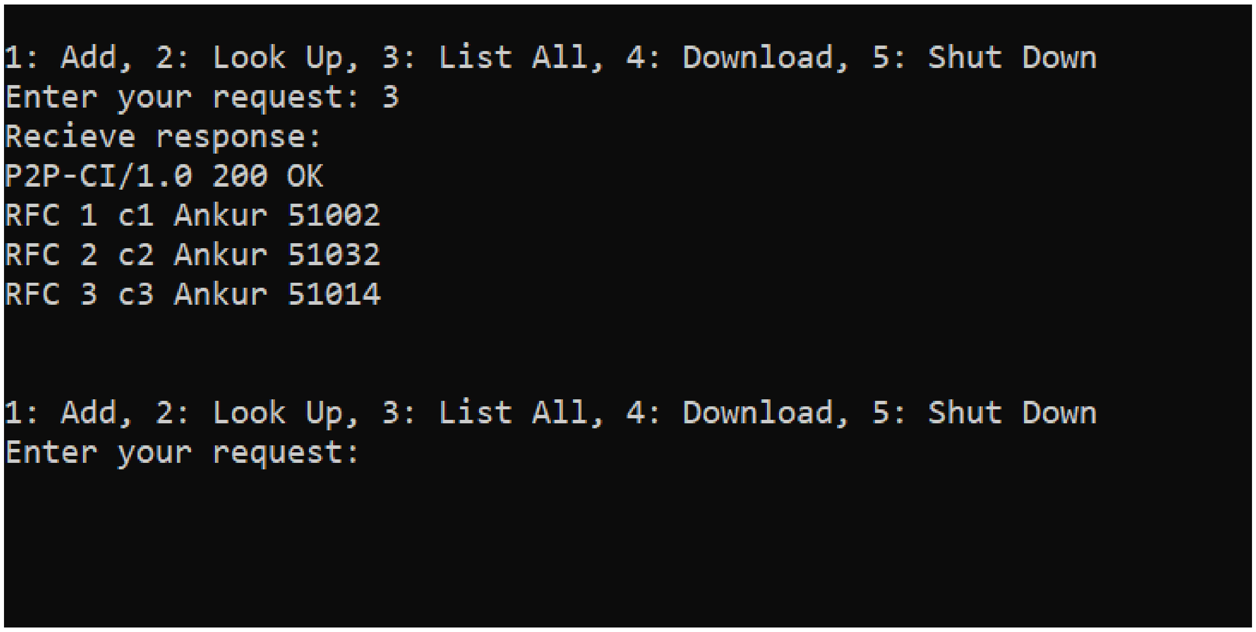
```
 127.0.0.1
Connecting to the server 127.0.0.1:7734
Connected
Listening on the upload port 51032

1: Add, 2: Look Up, 3: List All, 4: Download, 5: Shut Down
Enter your request: 1
Enter the RFC number: 2
Enter the RFC title: c2
rfc\rfc2.txt
Recieve response:
P2P-CI/1.0 200 OK
RFC 2 c2 Ankur 51032


1: Add, 2: Look Up, 3: List All, 4: Download, 5: Shut Down
Enter your request:
```

```
Connecting to the server 127.0.0.1:7734
Connected
Listening on the upload port 51014

1: Add, 2: Look Up, 3: List All, 4: Download, 5: Shut Down
Enter your request: 1
Enter the RFC number: 3
Enter the RFC title: c3
rfc\rfc3.txt
Recieve response:
P2P-CI/1.0 200 OK
RFC 3 c3 Ankur 51014


1: Add, 2: Look Up, 3: List All, 4: Download, 5: Shut Down
Enter your request:
```

```
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP\OneDrive\Desktop\CN_Project\Server_Assisted\client1>python client.py
 127.0.0.1
Connecting to the server 127.0.0.1:7734
Connected
Listening on the upload port 51002

1: Add, 2: Look Up, 3: List All, 4: Download, 5: Shut Down
Enter your request: 1
Enter the RFC number: 1
Enter the RFC title: c1
rfc\rfc1.txt
Recieve response:
P2P-CI/1.0 200 OK
RFC 1 c1 Ankur 51002


1: Add, 2: Look Up, 3: List All, 4: Download, 5: Shut Down
Enter your request:
```

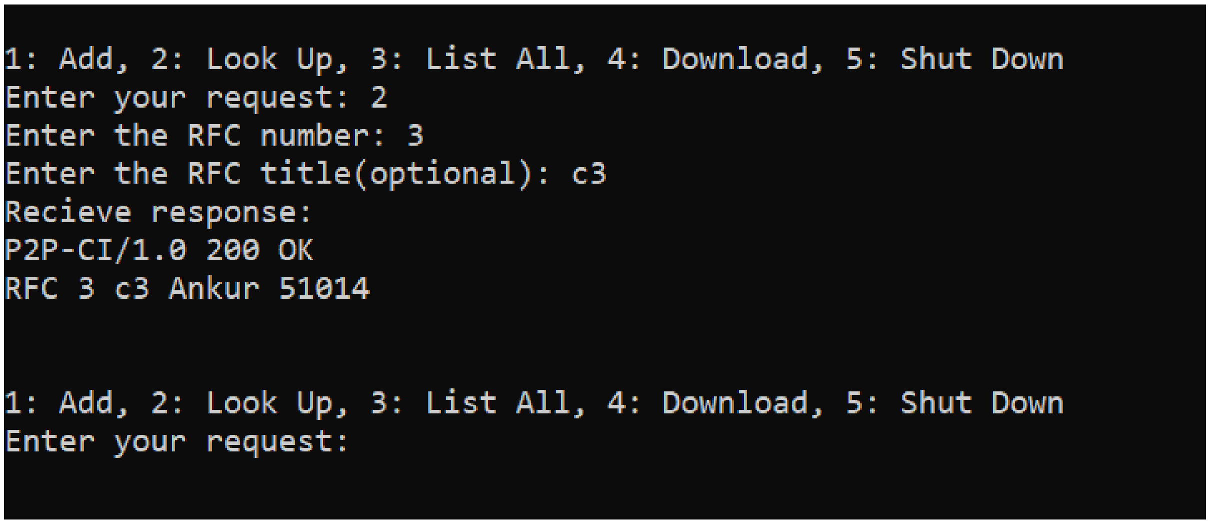
```
127.0.0.1:51001 connected
127.0.0.1:51013 connected
127.0.0.1:51031 connected
Recieve request:
ADD RFC 1 P2P-CI/1.0
Host: Ankur
Post: 51002
Title: c1

Recieve request:
ADD RFC 2 P2P-CI/1.0
Host: Ankur
Post: 51032
Title: c2

Recieve request:
ADD RFC 3 P2P-CI/1.0
Host: Ankur
Post: 51014
Title: c3
```

List all the uploaded file



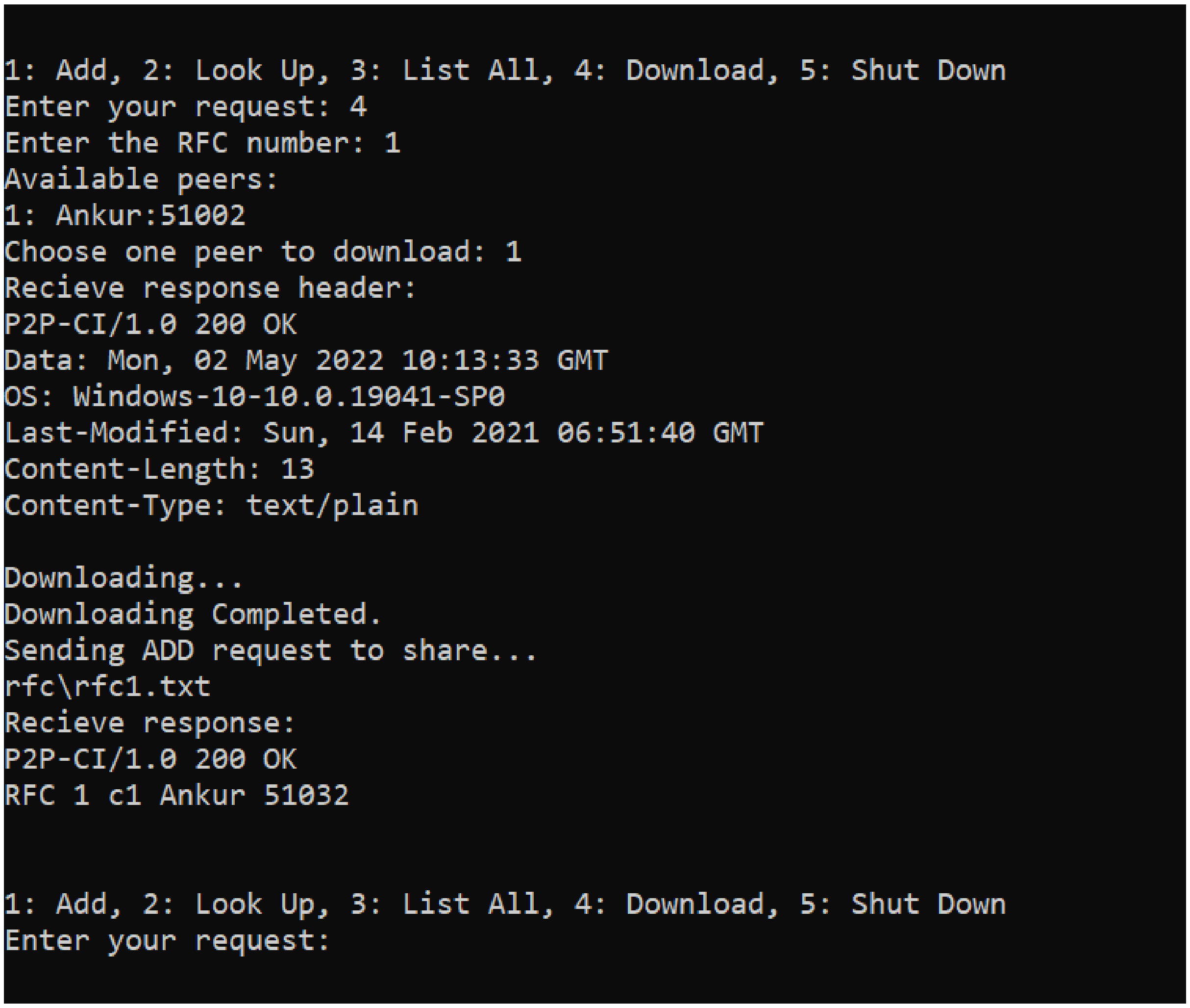
```
1: Add, 2: Look Up, 3: List All, 4: Download, 5: Shut Down
Enter your request: 3
Recieve response:
P2P-CI/1.0 200 OK
RFC 1 c1 Ankur 51002
RFC 2 c2 Ankur 51032
RFC 3 c3 Ankur 51014


1: Add, 2: Look Up, 3: List All, 4: Download, 5: Shut Down
Enter your request:
```

Look for the specific file
Eg: rfc3.txt file uploaded by client 3



```
1: Add, 2: Look Up, 3: List All, 4: Download, 5: Shut Down
Enter your request: 2
Enter the RFC number: 3
Enter the RFC title(optional): c3
Recieve response:
P2P-CI/1.0 200 OK
RFC 3 c3 Ankur 51014


1: Add, 2: Look Up, 3: List All, 4: Download, 5: Shut Down
Enter your request:
```

rfc1.txt file downloaded by client 2



```
1: Add, 2: Look Up, 3: List All, 4: Download, 5: Shut Down
Enter your request: 4
Enter the RFC number: 1
Available peers:
1: Ankur:51002
Choose one peer to download: 1
Recieve response header:
P2P-CI/1.0 200 OK
Data: Mon, 02 May 2022 10:13:33 GMT
OS: Windows-10-10.0.19041-SP0
Last-Modified: Sun, 14 Feb 2021 06:51:40 GMT
Content-Length: 13
Content-Type: text/plain

Downloading...
Downloading Completed.
Sending ADD request to share...
rfc\rfc1.txt
Recieve response:
P2P-CI/1.0 200 OK
RFC 1 c1 Ankur 51032


1: Add, 2: Look Up, 3: List All, 4: Download, 5: Shut Down
Enter your request:
```

Future Work:
As we know there’s no central authority in the peer-to-peer file-sharing system so it's difficult to control freeloaders and leechers. so there’s a need to control such freeloaders and leechers. Cyber-attacks should be controlled in peer-to-file sharing systems.