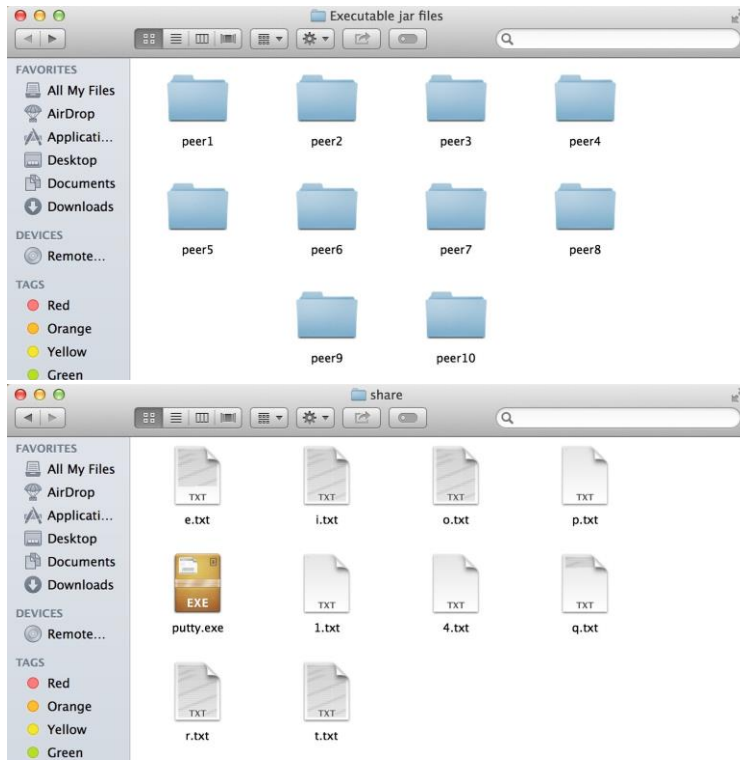


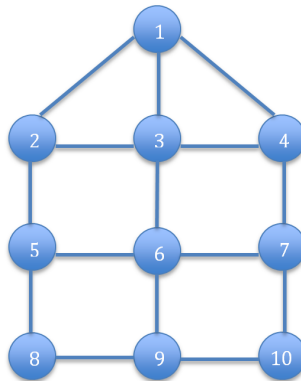
Verification

1. Test Environment and Nodes Deployment

Under this test, 10 peers were used for testing on MAC operating system.



And its topology structure is showed in the following figure.



The configure file config.txt is like this:



2. Set up peer name

First, we need to set the peer name, for example, for peer 1,

```
Yis-MacBook-Pro:peer1 sys$ java -jar peer.jar
```

```
1 Set up peer name
2 Register a file
3 Search a file
4 Exit
```

Type '1' to set the peer name "p1":

```
Yis-MacBook-Pro:peer1 sys$ java -jar peer.jar
```

```
1 Set up peer name
2 Register a file
3 Search a file
4 Exit
1
Enter the peer name:
p1
Local peer information:
p1 127.0.0.1 8010
Neighbor peers information:
p2 127.0.0.1 8020
p3 127.0.0.1 8030
p4 127.0.0.1 8040
```

Server started!

The program will read the configure file and get the peer information.

All the nodes must be set up names.

```
Yis-MacBook-Pro:peer1 sys$ java -jar peer.jar
1 Set up peer name
2 Register a file
3 Search a file
4 Exit
1
Enter the peer name:
p1
Local peer information:
p1 127.0.0.1 8010
Neighbor peers information:
p2 127.0.0.1 8020
p3 127.0.0.1 8030
p4 127.0.0.1 8040
Server started!

Yis-MacBook-Pro:peer2 sys$ java -jar peer.jar
1 Set up peer name
2 Register a file
3 Search a file
4 Exit
1
Enter the peer name:
p2
Local peer information:
p2 127.0.0.1 8020
Neighbor peers information:
p1 127.0.0.1 8010
p3 127.0.0.1 8030
p4 127.0.0.1 8040
Server started!

Yis-MacBook-Pro:peer3 sys$ java -jar peer.jar
1 Set up peer name
2 Register a file
3 Search a file
4 Exit
1
Enter the peer name:
p3
Local peer information:
p3 127.0.0.1 8030
Neighbor peers information:
p1 127.0.0.1 8010
p2 127.0.0.1 8020
p4 127.0.0.1 8040
p5 127.0.0.1 8050
Server started!

Yis-MacBook-Pro:peer4 sys$ java -jar peer.jar
1 Set up peer name
2 Register a file
3 Search a file
4 Exit
1
Enter the peer name:
p4
Local peer information:
p4 127.0.0.1 8040
Neighbor peers information:
p1 127.0.0.1 8010
p2 127.0.0.1 8020
p3 127.0.0.1 8030
p5 127.0.0.1 8050
p6 127.0.0.1 8060
Server started!

Yis-MacBook-Pro:peer5 sys$ java -jar peer.jar
1 Set up peer name
2 Register a file
3 Search a file
4 Exit
1
Enter the peer name:
p5
Local peer information:
p5 127.0.0.1 8050
Neighbor peers information:
p1 127.0.0.1 8010
p2 127.0.0.1 8020
p3 127.0.0.1 8030
p4 127.0.0.1 8040
p6 127.0.0.1 8060
p7 127.0.0.1 8070
Server started!

Yis-MacBook-Pro:peer6 sys$ java -jar peer.jar
1 Set up peer name
2 Register a file
3 Search a file
4 Exit
1
Enter the peer name:
p6
Local peer information:
p6 127.0.0.1 8060
Neighbor peers information:
p1 127.0.0.1 8010
p2 127.0.0.1 8020
p3 127.0.0.1 8030
p4 127.0.0.1 8040
p5 127.0.0.1 8050
p7 127.0.0.1 8070
p8 127.0.0.1 8080
Server started!

Yis-MacBook-Pro:peer7 sys$ java -jar peer.jar
1 Set up peer name
2 Register a file
3 Search a file
4 Exit
1
Enter the peer name:
p7
Local peer information:
p7 127.0.0.1 8070
Neighbor peers information:
p1 127.0.0.1 8010
p2 127.0.0.1 8020
p3 127.0.0.1 8030
p4 127.0.0.1 8040
p5 127.0.0.1 8050
p6 127.0.0.1 8060
p8 127.0.0.1 8080
p9 127.0.0.1 8090
Server started!

Yis-MacBook-Pro:peer8 sys$ java -jar peer.jar
1 Set up peer name
2 Register a file
3 Search a file
4 Exit
1
Enter the peer name:
p8
Local peer information:
p8 127.0.0.1 8080
Neighbor peers information:
p1 127.0.0.1 8010
p2 127.0.0.1 8020
p3 127.0.0.1 8030
p4 127.0.0.1 8040
p5 127.0.0.1 8050
p6 127.0.0.1 8060
p7 127.0.0.1 8070
p9 127.0.0.1 8090
Server started!

Yis-MacBook-Pro:peer9 sys$ java -jar peer.jar
1 Set up peer name
2 Register a file
3 Search a file
4 Exit
1
Enter the peer name:
p9
Local peer information:
p9 127.0.0.1 8090
Neighbor peers information:
p1 127.0.0.1 8010
p2 127.0.0.1 8020
p3 127.0.0.1 8030
p4 127.0.0.1 8040
p5 127.0.0.1 8050
p6 127.0.0.1 8060
p7 127.0.0.1 8070
p8 127.0.0.1 8080
p10 127.0.0.1 8100
Server started!

Yis-MacBook-Pro:peer10 sys$ java -jar peer.jar
1 Set up peer name
2 Register a file
3 Search a file
4 Exit
1
Enter the peer name:
p10
Local peer information:
p10 127.0.0.1 8100
Neighbor peers information:
p1 127.0.0.1 8010
p2 127.0.0.1 8020
p3 127.0.0.1 8030
p4 127.0.0.1 8040
p5 127.0.0.1 8050
p6 127.0.0.1 8060
p7 127.0.0.1 8070
p8 127.0.0.1 8080
p9 127.0.0.1 8090
Server started!
```

3. Register

In this part, we will test the register function.

Use peer1 to register one file o.txt:

```
1 Set up peer name
2 Register a file
3 Search a file
4 Exit
2
Enter the file name:
o.txt
File o.txt is registered!
```

The file o.txt is successfully registered.

If we want to register a nonexistent file:

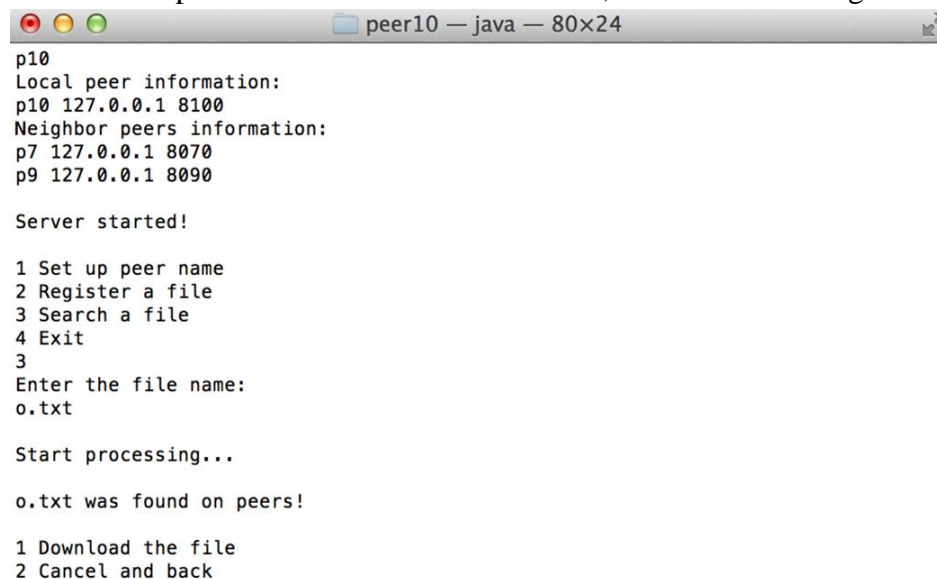
```
1 Set up peer name
2 Register a file
3 Search a file
4 Exit
2
Enter the file name:
sdfe
sdfe is not exist!
```

The program finds that file “sdfe” does not exist.

4. Search

In this part, we will test the search function.

Now we use peer 10 to search for the file o.txt, which has been registered in peer 1.



```
p10
Local peer information:
p10 127.0.0.1 8100
Neighbor peers information:
p7 127.0.0.1 8070
p9 127.0.0.1 8090

Server started!

1 Set up peer name
2 Register a file
3 Search a file
4 Exit
3
Enter the file name:
o.txt

Start processing...

o.txt was found on peers!

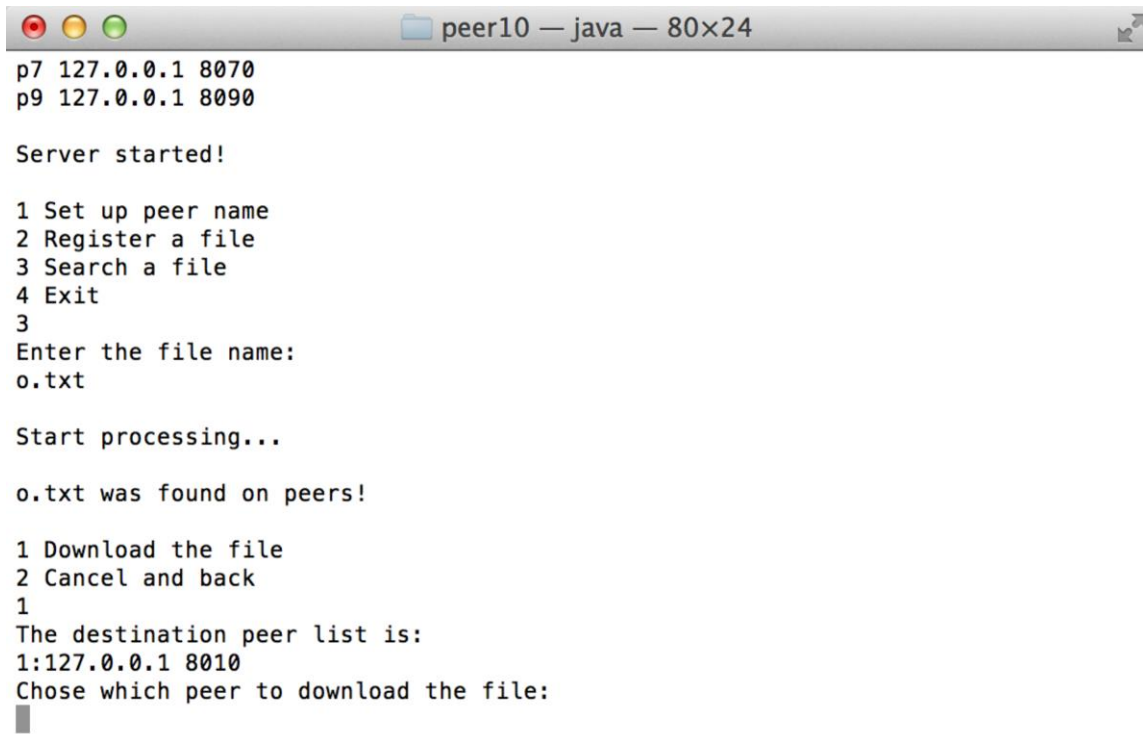
1 Download the file
2 Cancel and back
```

It shows that the file has been found, so the search function works.

5. Download

In the previous part, file has been found, now we can download it.

Type '1' to choose downloading.



```
p7 127.0.0.1 8070
p9 127.0.0.1 8090

Server started!

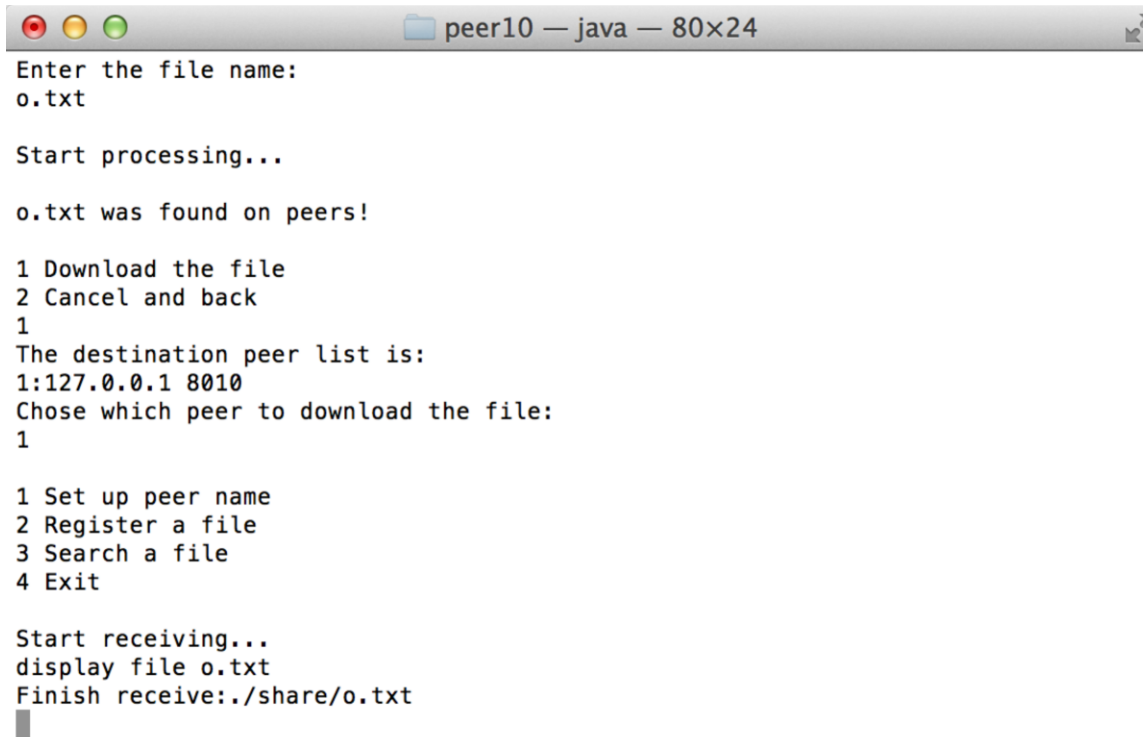
1 Set up peer name
2 Register a file
3 Search a file
4 Exit
3
Enter the file name:
o.txt

Start processing...

o.txt was found on peers!

1 Download the file
2 Cancel and back
1
The destination peer list is:
1:127.0.0.1 8010
Chose which peer to download the file:
1
```

Choose the source to start download.



```
Enter the file name:
o.txt

Start processing...

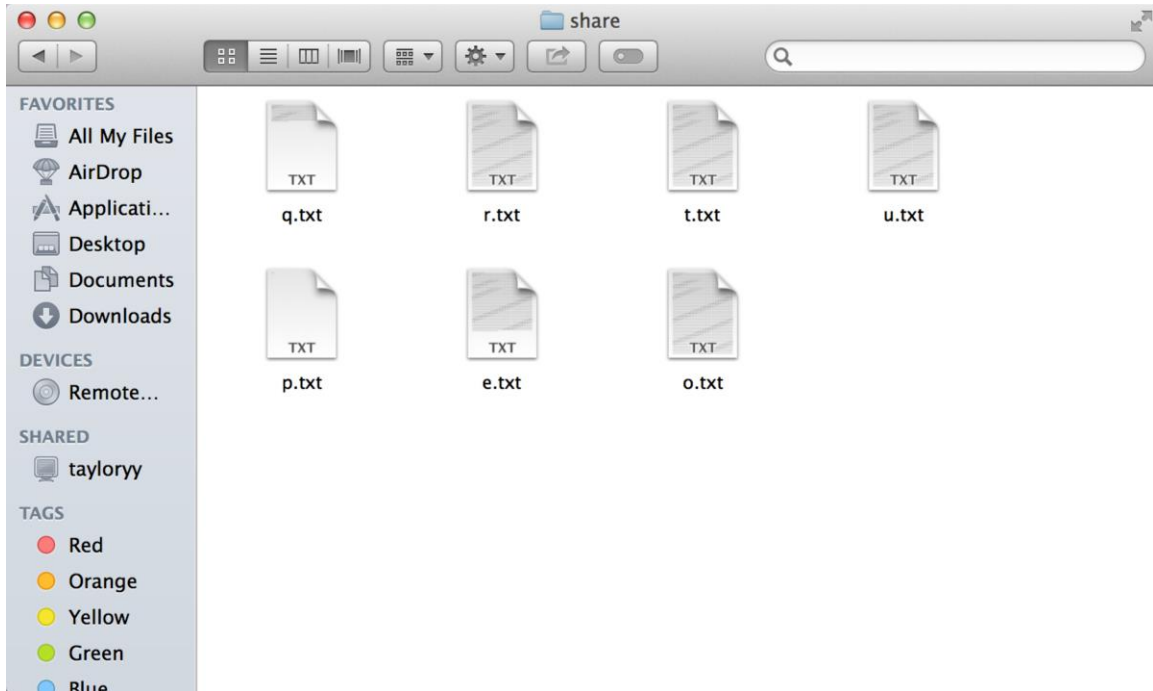
o.txt was found on peers!

1 Download the file
2 Cancel and back
1
The destination peer list is:
1:127.0.0.1 8010
Chose which peer to download the file:
1

1 Set up peer name
2 Register a file
3 Search a file
4 Exit

Start receiving...
display file o.txt
Finish receive:./share/o.txt
```

Then the file is received.



File o.txt has been downloaded.

Let peer 10 register file o.txt:

```
peer10 — java — 80x24
1
The destination peer list is:
1:127.0.0.1 8010
Chose which peer to download the file:
1

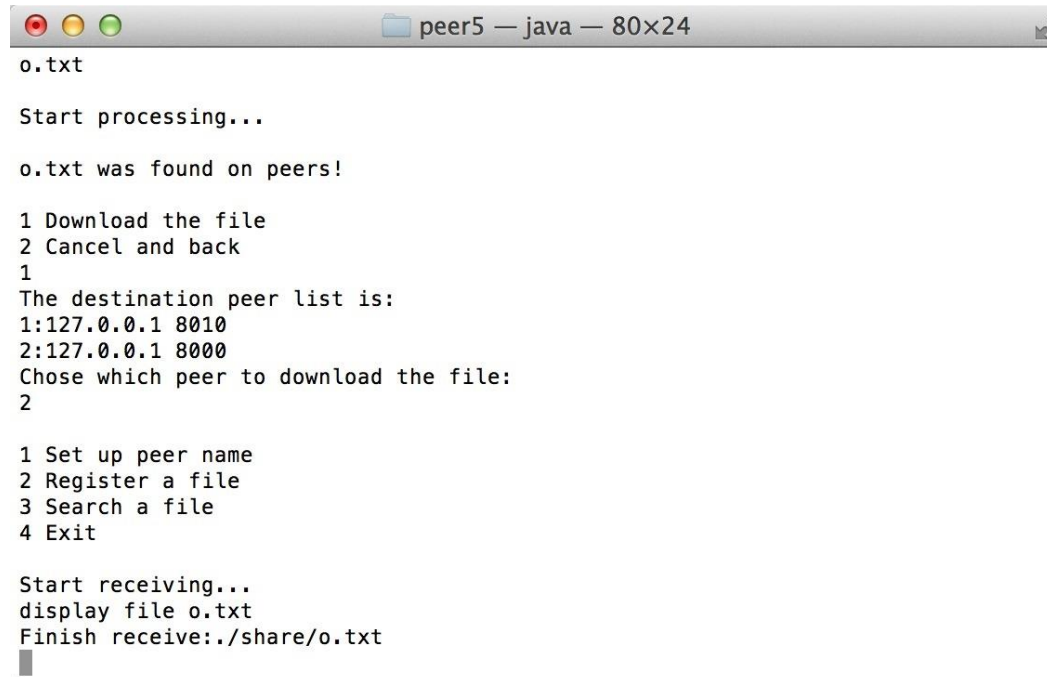
1 Set up peer name
2 Register a file
3 Search a file
4 Exit

Start receiving...
display file o.txt
Finish receive:./share/o.txt
2
Enter the file name:
o.txt
File o.txt is registered!

1 Set up peer name
2 Register a file
3 Search a file
4 Exit
```

Then use peer 5 to search for file o.txt (now o.txt has been registered by peer 1 and peer 10). It can find two peers that have the file. When downloading the file, there are two places can be chosen to download:

Receive End:



```
o.txt

Start processing...

o.txt was found on peers!

1 Download the file
2 Cancel and back
1
The destination peer list is:
1:127.0.0.1 8010
2:127.0.0.1 8000
Chose which peer to download the file:
2

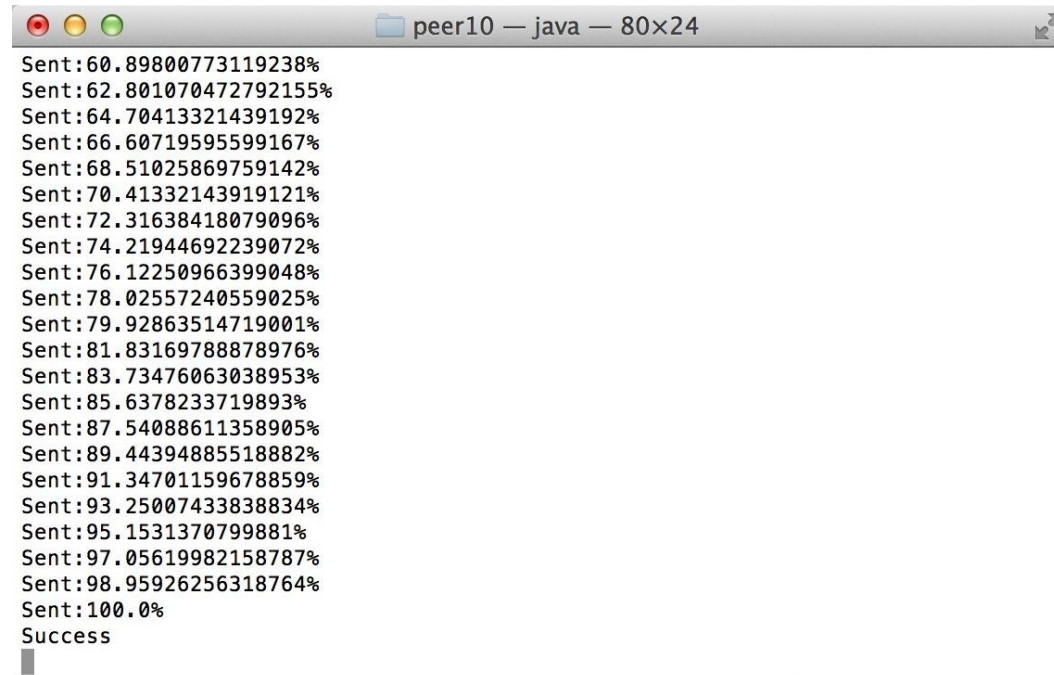
1 Set up peer name
2 Register a file
3 Search a file
4 Exit

Start receiving...
display file o.txt
Finish receive:./share/o.txt
```

Choose '2' to download, peer 5 will receive file from peer 10.

If the screen displays like the above picture, the file has been received.

Send End:



```
Sent:60.89800773119238%
Sent:62.801070472792155%
Sent:64.70413321439192%
Sent:66.60719595599167%
Sent:68.51025869759142%
Sent:70.41332143919121%
Sent:72.31638418079096%
Sent:74.21944692239072%
Sent:76.12250966399048%
Sent:78.02557240559025%
Sent:79.92863514719001%
Sent:81.83169788878976%
Sent:83.73476063038953%
Sent:85.6378233719893%
Sent:87.54088611358905%
Sent:89.44394885518882%
Sent:91.34701159678859%
Sent:93.25007433838834%
Sent:95.1531370799881%
Sent:97.05619982158787%
Sent:98.95926256318764%
Sent:100.0%
Success
```

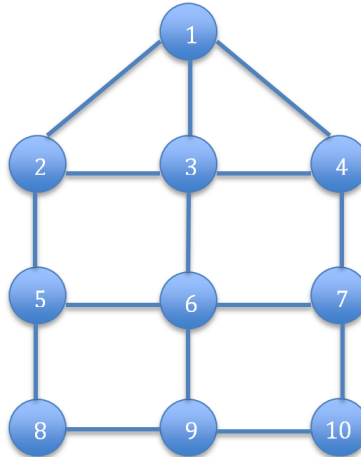
6. TTL test

In the previous part, we can see that peer 5 can search file from peer 1 and peer 10, because we set the TTL = 3. The distance between peer 1 and peer 5 is smaller than 3,

and the distance between peer 5 and peer 10 equals to 3. So both peer 1 and peer 10 can be searched when peer 5 runs the searching process.

Since the TTL = 3, then the messages should only be sent three layers, now we will test whether the TTL works.

In our topology:



The distance between peer 2 and peer 10 is larger than 3, so we use peer 2 to search for the file o.txt:

The result is shown below:

```
peer2 — java — 80x24
p3 127.0.0.1 8030
p5 127.0.0.1 8050

Server started!

1 Set up peer name
2 Register a file
3 Search a file
4 Exit
3
Enter the file name:
o.txt

Start processing...

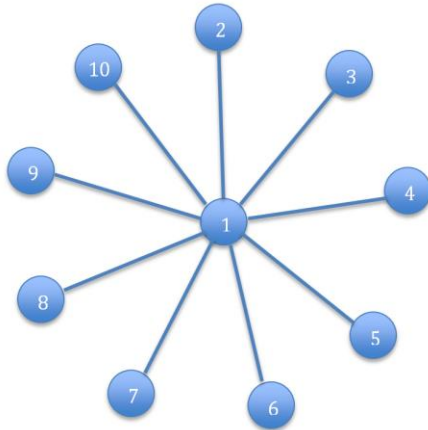
o.txt was found on peers!

1 Download the file
2 Cancel and back
1
The destination peer list is:
1:127.0.0.1 8010
Chose which peer to download the file:
█
```

It shows that peer 2 can only find the file in peer 1, can not find the file in peer 10, so the TTL works.

7. Average query time test

1) Star-topology



The configure file is like this:

```
config.txt
p1 127.0.0.1 8010 p2 p3 p4 p5 p6 p7 p8 p9 p10
p2 127.0.0.1 8020 p1
p3 127.0.0.1 8030 p1
p4 127.0.0.1 8040 p1
p5 127.0.0.1 8050 p1
p6 127.0.0.1 8060 p1
p7 127.0.0.1 8070 p1
p8 127.0.0.1 8080 p1
p9 127.0.0.1 8090 p1
p10 127.0.0.1 8000 p1
```

We modified the program, let one peer query 200 times, and measure the average time.

The result is shown below:

```
peer1 — java — 80x24

1 Set up peer name
2 Register a file
3 Search a file
4 Exit
3
Enter the file name:
t.txt

Start Time:1413598443252

1 Set up peer name
2 Register a file
3 Search a file
4 Exit

End Time:1413598446552

Total Time: 144ms

Total Query times:200

Average Query Time:0.7218212104386452ms
```


When two peers query at the same time, the average time is as following:

End Time:1413602617447

Total Time:223ms

Total Query times:200

Average Query Time:1.1149361465852303ms

When three peers query at the same time, the average time is got:

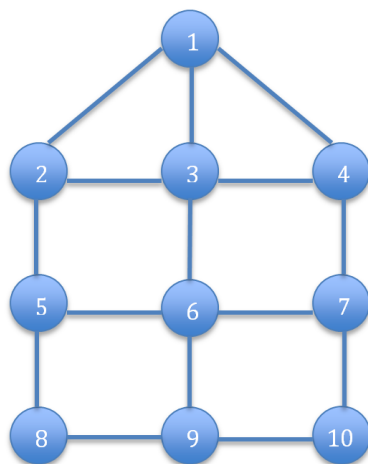
End Time:1413602617447

Total Time:549ms

Total Query times:200

Average Query Time:2.7481481481481481ms

2) 2D-mesh topology



The configure file is like this:

```
config
p1 127.0.0.1 8010 p2 p3 p4
p2 127.0.0.1 8020 p1 p3 p5
p3 127.0.0.1 8030 p1 p2 p4 p6
p4 127.0.0.1 8040 p1 p3 p7
p5 127.0.0.1 8050 p2 p6 p8
p6 127.0.0.1 8060 p3 p5 p7 p9
p7 127.0.0.1 8070 p4 p6 p10
p8 127.0.0.1 8080 p5 p9
p9 127.0.0.1 8090 p6 p8 p10
p10 127.0.0.1 8100 p7 p9
```

Let one peer query 200 times, get the average query time:

End Time:1413607289686

Total Time:914ms

Total Query times:200

Average Query Time:4.574074074074074ms

Then let two peers query at the same time. The results show below:

End Time:1413607478330

Total Time:1047ms

Total Query times:200

Average Query Time:5.235583223029639ms

Three peers to query at the same time:

End Time:1413610168327

Total Time:1142ms

Total Query times:200

Average Query Time:5.711111111111111ms