CS 550 Programming Assignment1
P2P File Sharing System
Yue Wu A20359521
Xin Liu A20353208
Group# 17

Manual

The document shows to user how to use this program, and attach the example of test file.
Running the program:
This program is developed in Mac OS by java programming language. In order to run the program, change the directory to the sourcecode folder.

1. Use java *.java to compile the program. If there are no .class file in the source code file, you should type this command first.
2. Use java cServer to run the central indexing server
3. Use java Peer to run the peer
4. Use java Test1 to run the test file. However, before running the test file, you should run a indexing server and a peer (name test, port 6678) at first.
5. Use rm *.class to delete all the files which type is .class.

Indexing Server
In this program, user should run the indexing server first with typing command “java cServer ” in terminal. Then the server runs on port 20000. No more input messages require from indexing server.

Peer
As the same as indexing server, user can run the peer with command “java Peer ”. Then the peer program will run two subthreads called peer client and peer server.

Run Peer step-by-step:
This part will show how to run the peer part step by step. Also the line start by ">>" in this part are input messages typed by user.

a. a new peer, user will be asked to input the name and port of the peer. Just input a reasonable name and legal port number, then press enter. A peer will be set up for you. Following is the output in the terminal.

```
set up a new peer, please input the name of the peer(press enter to end):
>> Peer
please input the port number of peer:
>> 6787
```

b. Once the peer started, it will automatically register all the files in this peer. Following is the output in the terminal. The output is the information send by the peer which will be processed by the indexing server.

```
Start the client.....
The name of the client is: Peer
start
REGISTRY@_@test0.txt@_@Peer@_@localhost@_@6787
REGISTRY@_@test1.txt@_@Peer@_@localhost@_@6787
REGISTRY@_@test10.txt@_@Peer@_@localhost@_@6787
REGISTRY@_@test2.txt@_@Peer@_@localhost@_@6787
REGISTRY@_@test3.txt@_@Peer@_@localhost@_@6787
REGISTRY@_@test4.txt@_@Peer@_@localhost@_@6787
REGISTRY@_@test5.txt@_@Peer@_@localhost@_@6787
REGISTRY@_@test6.txt@_@Peer@_@localhost@_@6787
```

REGISTRY@_@test7.txt@_@Peer@_@localhost@_@6787
REGISTRY@_@test8.txt@_@Peer@_@localhost@_@6787
REGISTRY@_@test9.txt@_@Peer@_@localhost@_@6787
register success!

c. User will be asked to choose from one of the following:
1. look up files
2. download files from other peer
3. delete one file in the client

Option 1 is function to search one file in the indexing server, it will ask user to input a file's name, then it will return the searching result to the user.
Option 2 is function to download files from other peer. However, this option will ask user to input a file's name, then searching this name in the indexing server, then return the searching result to the user. User will choose which peer to download the file.
Option 3 is to delete one file of this peer. User will be asked to input a file's name. Then the file will be deleted in local folder and the indexing server.

Following is an example of using this program:

1. Lookup one file(name: test1.txt). Then the indexing server will send back the searching result to the peer which is shown below.

Please choose between the following:
1. look up files
2. download files from other peer
3. delete one file in the client
>>1
please input the file's name you want to search
>> test1.txt
Gonna find the file test1.txt
Following is the result of lookup
test1.txt>_<Peer>_<localhost>_<6787

2. Download one file from other peer, in this condition, there are two peers contain this file. Peer program will output the information of this two peer, and ask user to choose one. Then the file needed will be saved in local folder.

Please choose between the following:
1. look up files
2. download files from other peer
3. delete one file in the client
>>2
input name of files, you want to download
>>test6.txt
Gonna find the file test6.txt
found this file you want to download
There are 2 peers contain this file
Following is the name and port of the peer
0 : test6.txt>_<Peer>_<localhost>_<6787
1 : test6.txt>_<A>_<localhost>_<6798
Please input the index of the peer you want to connect
>>1
The file is download, and saved in the local folder, name of folder is Peer_download

3. Delete one file in local folder and also the central indexing server.

Please choose between the following:
1. look up files
2. download files from other peer
3. delete one file in the client

>>3
input the name of file you want to delete
>>test6.txt
test6.txt is deleted!

Running test case

User can use command " java Test1" to run test file(name: Test1.java) of this program. This test file will test the look up and download function, then output the time cost by these two function.
In order to run this test file, user need to run the central server at first, and then run the test peer(name: test, port:6678).
The download file will be saved in test2_download folder.

Following is the step of running test case:

1. type java cServer in terminal
2. type java Peer in a new terminal, and the name of the peer should be test, the port of the peer should be 6678
3. type java Test1 in a new terminal

Following is an example of the output of the test case:
At first, test the look up function.
test the download function
Total time cost by this lookup thread is :4332
Total time cost by this lookup thread is :4333
Total time cost by this lookup thread is :4458
Total time cost by this lookup thread is :4471
Total time cost by this lookup thread is :4472
Total time costs by this download Thread is: 4493ms
Total time costs by this download Thread is: 4495ms
Total time costs by this download Thread is: 4496ms
Total time costs by this download Thread is: 4496ms
Total time costs by this download Thread is: 4502ms

This test case will keep looking up one file for 1000 times, and there will be five thread which represent three client. Also, this test case will keep download one file from test peer for 1000 times. This test file will ensure that the peer and central can deal with multiply requests.