Patrick O'Shea
CS550
PA1
2/28/2011

Program Manual

Initial setup:

Edit the java security policy to temporarily allow all permissions.
(UNIX and Windows)
in your user's home directory edit the file .java.policy and add the following lines:

```
grant {
  permission java.security.AllPermission;
};
```

This file can be changed back after evaluation of the program

Start the RMI Registry
(UNIX) Run the following command on the command line:
rmiregistry &

(Windows) Run the following command on the command line:
start rmiregistry

Change the working directory to the /OShea_Patrick_PA1 included in the submitted archive.

(**All java commands must be issued from the root /Oshea_Patrick_PA1 folder or the program will not run correctly)**

Start the p2pServer with the following command:
java -Djava.rmi.server.codebase="file:///-workingdirectory-/p2pServer/bin/" -cp p2pServer/bin p2pServer.p2pServer

where – working directory – is the fully qualified current path that can be found with pwd.

Here is an example:
java -Djava.rmi.server.codebase="file:///home/pat/Documents/Masters/CS550/p2pServer/bin/" -cp p2pServer/bin p2pServer.p2pServer

Once the p2pServer program is running the user should be presented with the following:
Server running...

Enter 0 to exit:

Open a new terminal window and change the working directory to /Oshea_Patrick_PA1

Start the p2pClient with the following command:
java -Djava.rmi.server.codebase="file:///-workingdirectory-/p2pClient/bin/" -cp p2pClient/bin:p2pServer/bin client.Client k

where -workingdirectory- is as above, and k is a unique integer for the peer id, such as 1. (Assuming user will make sure to use unique ids) The user can launch multiple nodes in multiple terminal windows as long as they use unique ids.

Here is an example:
java -Djava.rmi.server.codebase="file:///home/pat/Documents/Masters/CS550/p2pClient/bin/" -cp p2pClient/bin:p2pServer/bin client.Client 1

Once the Client program is running, it will test to see if a shared directory exists, and if not will create one. Then it will automatically register all files in the shared directory with the server.

The user will then be presented with something similar to this menu:

ClientServer running...PeerID = 1
# of files registered: 5

PeerID: 1
Options:
1 - Search for filename
2 - Obtain filename from peer
3 - List files in shared directory
4 - Exit

:

Option 1 will return all peers sharing the filename the user specifies.

Example output:
Enter filename: text1000.txt

The following peers have the file (text1000.txt) :
1

Lookup Response time: 14904362 ns

Option 2 will attempt to connect to the specified peer and download the specified file into the client's shared folder. Upon successful download, the client will automatically register the file with the index server. (There will be different folders for each client).

Example output:
Enter filename: text1000.txt

Enter peer: 2

display file text1000.txt
Download Response time: 64311672ns

Option 3 will list all the files in the current client's shared directory.

Example output:
Files in shared directory:
text8000.txt
text5000.txt
text2000.txt
text7000.txt
text6000.txt
text10000.txt
text3000.txt
text4000.txt
text9000.txt
text1000.txt

Option 4 will list the average index server response time for the entire session, unbind the client from the registry and exit the program.

Example output:
Average Lookup Response time for this session: 9126756

The program also will present the download response time, lookup response time and a running average lookup response time (which is displayed upon exiting the program).

The server program may be exited by entering 0, which will also unbind it from the registry.