Adrian Birylo
Daniel Hitchings
CS 451 Project 3
Due Date: 4-8-2014

# Manual

## Preface

All of the code was tested in Ubuntu 13 with Java 7.
All command are issued from the `Project3/src_[centralized/decentralized]` directory.

## Compiling

The command line argument to compile the server is:

```
$ javac Server/*.java
```

The command line argument to compile the client is:

```
$javac Client/*.java
```

## Running

Before the server or client can be ran rmiregistry needs to be run before hand. To run the command type `$ rmiregistry &` in the terminal. Once the rmiregistry is running then the server and clients can be run. The server needs to run before any of the clients can run for the centralized system. For the decentralized system any client can run in any order.

The command to run the server is:

```
$ java Server.RMIServer -Djava.security.policy=server.policy
```

The command to run the client is:

```
$ java Client.RMIClient (arg) -Djava.security.policy=client.policy
```

The variable (arg) is where the id of the client needs to be past in. For running three client the above command would be run in three terminals with (arg) replaced with 1,2,3 respectfully.

For the decentralized system only the client is needed to run. The clients find it's neighbors by using a config file which is located in `./Config`. For each client 0-9 there is a Client#.config file. On startup the client will look for its specific config file and then try to connect to all of its neighbors. The client will wait until it can connect to all of it's neighbors before running as a peer-to-peer program.

## Server

When the command to run the server is typed the display in the terminal will be:

```
$ java Server.RMIServer -Djava.security.policy=server.policy
Server is running
To exit type 'exit'
```

To exit from the server type `exit` to exit the server program.

## Client

For a simple demo run three clients in three terminals using the command shown above with (args) equal to 1,2,3 respectively. Both for the centralized and decentralized clients have the same user interface and will be have in the same way. When the client is ran the terminal will display:

```
$ java Client.RMIClient 1 -Djava.security.policy=client.policy
Client running... PeerID = 1
Options:
1 - Search for filename
2 - Obtain filename from peer
3 - List files in shared directory
4 - Exit
>
```

Once the Options are displayed a prompted is show that the user can enter 1 through 4 to do the action specified.

If Option 1 is selected then there is another prompt that asks the user for a filename. Then it searches the registry for the file and lists all of the clients that has the file. Below is a sample output.

```
>1
Enter filename: Text1.txt

The clients that have the file Text1.txt are:
Client 1
Client 2
Client 3
```

If Option 2 is selected then there is another prompt that asks the user for a filename and a preferred client to download the file from. If 0 is entered then the first host will be selected. Below is a sample output.

```
>2
Enter filename: Text1.txt
Enter peer. If no preference enter 0.
0
```

If Option 3 is selected then all of the files in the local directory to share are listed. Below is a sample output.

```
>3
Files in the shared directory:
Text1.txt
Text4.txt
Text7.txt
```

If Option 4 is selected then the registry is updated with any modification to which files are in the shared directory thus if files were added then they are added to the registry or if the files are removed then they are removed from the registry.

If option 5 is selected then the program exits.