Fall 2013 Programming Project

CSci 5211

Vivek Ranjan - 4924202

Due on: 10th December, 2013

Contents

1 About

The project is a Peer-to-Peer (P2P) file sharing system, written in Java and implemented by using sockets. This document lists down all the features that have been implemented and how to use the various programs.

2 Archive Contents

The archive contains the following folders/directories:

code Contains all the source files for the project. All source files are heavily commented to ensure high readability and make the code easy to understand for non-java programmers too.

lib Contain a few libraries (mostly related to logging) that have been used in the programs.

docs Contains JavaDocs detailing all the classes and methods that are used. Open index.html in any browser to start.

executables Contains all the executable files that can be used to run the programs. Instructions on how to run the programs are provided below. These are independent (do not require any of the other files in the archive) and portable (can be run on any environment that satisfies the requirements stated below.)

3 Running The Programs

3.1 Requirements

The programs do require java to be installed on the machine prior to running it. Instructions for installing the Java Runtime Environment (JRE) can be found on the following link - `https://www.java.com/en/download/help/download_options.xml`. Do note that the CS grad lab machines (cello, trombone, etc) and the CSE lab machines, both already have Java installed and can be used to run these programs. These programs have been tested to run on the Linux machines on both labs.
Running the programs is simple and follows the requirements stated in the programming project description.

3.2 Central Server

The central server program can be run using the **centralserver.jar** file in the executables directory. The following command should be used:

```
java -jar centralserver.jar
```

The central server program can also take two optional command line arguments. Namely:

-f<path/to/file> Stores the list of peers in the specified file. Default file is peerList.txt in the current working directory. For example:

```
java -jar centralserver.jar -fpeers.txt
```

-p<port> Specifies what port the program should use to listen for incoming messages. Default port used by the program is 5555. For example:

```
java -jar centralserver.jar -p5355
```

3.3 Peer

The peer program can be run using the **peer.jar** file in the executables directory. The following command should be used:

```
java -jar peer.jar <path/to/working-directory>
<central-server-ip>:<central-server-port>
```

Example:

```
java -jar peer.jar peer3 kh2170-01.cselabs.umn.edu:5555
```

The first argument is the *full* path to the working directory that will be used to share files to and from the peers on the network. The second argument is the complete address of the central server, including the port, in the IP:port format. The IP can be the numeric IP or the host address of the central server. Both will work. The peer program does not support any additional arguments.

4 Using the Peer program

The peer program is completely multi-threaded and any number of the mentioned tasks can be run simultaneously. It supports the following functionalities:

get Searches the network for the specified file and downloads it into the working directory.

share Sends the specified file to the specified peer.

list Lists all the files in the working directory.

quit Terminates the program after informing its neighbours and the central server about the same.

The usage of each command is explained below.

4.1 get

The **get** command is used to download a file from the P2P network after searching for it. The file is searched by flooding all the neighbours of the requesting peer with a lookup command, who in turn flood their neighbours with the same. This continues till the file is found and a direct connection is established between the source and destination peer hosts to download the file. The syntax for the command is:

```
get <filename>
```

Example:

```
get mydoc4.pdf
```

The file is downloaded and stored in the working directory specified while initialising the peer program.

If the file is not found within 5 seconds, a timeout error message will be displayed and the peer will stop waiting for the file.

4.2 share

The **share** command is used to send a file to a peer host in the network, that may or may not be the source peer host's neighbour. The syntax for this command is:

```
share <filename> <ip>:<port>
```

Example:

```
share mydoc7.psd trombone.cs.umn.edu:55932
```

If an invalid filename is provided, command will be terminated after displaying an error message. The port number is the port that the destination peer is using to listen for messages.

4.3 list

This command just lists down all the files in the working directory. It does not take or need any arguments

4.4 quit

This command terminates the peer and exits the program. Before completely exiting the program, it informs the neighbours and the central server about the same.

5 Further Information

A quick bird's eye view of the inner working of the programs can be achieved by taking a look at the JavaDocs available in the **docs** directory. It contains all the classes and methods used in the program.

Further exploration can be done by going through the source code in the **code** directory. All source files have been properly and thoroughly commented so that it is easy to understand the code even for a non-java programmer.