ASSIGNMENT – 1
PROGRAMMING WITH SOCKETS

This assignment introduces you to basic socket programming. The goal is to get you comfortable with (a) Unix socket programming and (b) simple client-server interaction.

Goal: Implement a peer-to-peer file sharing service like Napster (refer to the notes on p2p networks on course website) using sockets. The interface should be a simple command line interface. However you can be creative and add features that are not specifically asked for. Following components need to be implemented:

1. **A central server:**
    1. All peers who wish to use this service register with the server
    2. The server maintains an index keeping track of what files belong to which peer/peers
    3. All search queries are directed to the server which returns information on peers holding the file searched for.

2. **A peer:** This is the client which connects to the server

Features to Implement:

1. The server will start in *passive* mode listening on a specified port and display the following menu:
    1. Register with Napster
    2. Share files
    3. Search for a file

2. A client starts separately. If it's interested in using Napster, it will contact the server on it's IP address and port number

```
./client server-IP-address port-number
```

3. The client firstly selects option 1 to register itself. The server correspondingly updates its list of active peers.

4. The client can select option 2 to send the list of files it wishes to share to the server which updates its index accordingly.

5. A client can further send a search query to the server containing a filename, using option 3. The server should lookup for the file in its index and send the querying client a list of clients having the file.

Note:

1. The client messages to the server can have headers so that the server understands what is it that the client wants to do. For example to share a list of files along with their locations on the client's local disk, the client can send the following message to the server:

```
SHARE_FILES
<filename1>     </path/to/file/filename1>
<filename2>     </path/to/file/filename2>
```

The server on reading the first line of this message understands that the client wishes to share a file list and acts accordingly.

2. There can be many peers in the system, therefore make sure that your server can hadle multiple clients.

3. There can be issues such as mutiple clients sending in search queries simultaneously or another instance where a client sends a request to share a list of files and another one a search query simultaneously. In this situation the server should prioritize which type of request to handle first and which client's request to handle first.

4. Use a waiting queue at the server to buffer the client requests. Think upon the various scheduling algorithms with their advantages and disadvantages and then use the most appropriate algorithm to schedule these requests.

Extra Credit:

The querying client on getting a list of candidate clients, selects any one client, connects to it and downloads the file. To make it simpler, use simple text files instead of mp3 files (as were originally there in Napster)

Stuff to assist you:

1. Notes on Socket Programming on course website: http://web.cse.iitk.ac.in/users/cs425/2013/lecnotes/sockets.pdf

2. Notes on Napster can be found in the detailed notes on p2p networks: http://web.cse.iitk.ac.in/users/cs425/2013/lecnotes/p2p_networks.pdf

3. Further there's a lot of material on web on socket programming. Following is a very good tutorial: http://beej.us/guide/bgnet/output/html/singlepage/bgnet.html

4. Sockets are very easy to use in Python. However you are free to use any programming language.

Deadline:

This assignment is due at 11:59 PM Sunday i.e. 8-09-2013. Please submit on time, late submissions shall be penalized. Kindly mail a zipped folder of your assignment to the TAs.