Distributed Systems Lab - Assignment # 3

Peer-to-Peer (P2P) Chat Application:

Peer-to-peer (P2P) applications are a common feature of computing today. Services from BitTorrent to Skype all rely on direct client to client contact. In this programming assignment you will construct a simple P2P based relay application. This application will demonstrate the basic operation of P2P communication.

For this lab, you need to write a client and a server utilizing TCP socket connections. The client must be able to connect to the server and obtain a list of available clients. After obtaining this list, the client must be able to connect to one other client on the list. The clients must then be capable of relaying text messages directly to one another without further help from the server.

You can define whatever convention you would like to indicate that a text message is complete (e.g. hitting [Enter], clicking send). Each client must send an acknowledgement to the other client that a text message has been received and displayed. You must also provide some way to disconnect and end the conversation. After disconnecting, the user should be able to retrieve a new copy of the server's client list and connect to a new client. The connection between the two clients must be duplex, so both clients are capable of transmitting and receiving at the same time.

This can be envisioned as a distributed P2P chat application, where you can figure out who is online from the server and then talk to those other users directly.

Assignment Goal:

The goal of this assignment is to perform socket programming in a peer-to-peer context. You can use any IP addresses you would like for communication.

Hints:

Here are a few hints that may help you as you write the program.

1. You have to choose a server port to connect to. Ports from 1-1023 are mostly used for certain services and require administrative privileges. Use port numbers greater than at least 1023.
2. Close your sockets cleanly before exiting the program. If you abort the program, the port may not be freed.
3. You can run all of the processes on the same machine. For your machine, just use localhost. You can use ifconfig (unix) or ipconfig (windows) to determine the IP address for testing across multiple machines
4. Be wary of overzealous firewalls stopping your connections - try temporarily disabling firewalls if you find your connections timeout or are denied.
5. Don't forget that Wireshark can watch what your program is transmitting, possibly helping you during debugging.