EE 382N Assignment 1 Fall 2004

Due September 12

You are required to implement a fault-tolerant ticket reservation system for a movie. Assume that the movie theatre has c total seats with seat number 0 as the most desirable seat and the seat number $c-1$ as the least desirable seat. There are n servers that keep the current allocation of the seats. All servers know ip addresses of other servers. Your system should give the user illusion of a single immortal server (so far as serving the requests are concerned).

A client may connect to any of the servers. A server accepts three kinds of calls from a client:

1. `reserve name` – inputs the name of a person and assigns the most desirable seat available to that person. You may assume that the name is a single word. The command returns the seat number assigned to the person. If a seat is already assigned to that name, the command returns -1. if the theater is full, the command returns -2. No waiting list is maintained.

2. `search name` – returns the seat number assigned to him or her (returns -1 if the if the name is not found).

3. `delete name` – frees up the seat allocated to that person. The command returns the seat number that has become free now (and -1 if the name is not found). It is a no-op if the person was not assigned any seat. It does not reallocate seats to clients that have already been assigned seats even though a more desirable seat may have opened up.

Note that you also have to write a program that takes input from the user and then communicates with a server using sockets. Assume that a client chooses a server at random. If that server is down (i.e. after the timeout), the client contacts some other server. You may assume that there is at least one server that is always running. Do not assume that it is the same server that is always up.

When a server comes up again, it would need to synchronize with existing servers to ensure consistency. You can also assume "perfect" failure detection.

Your program should behave correctly in presence of multiple concurrent clients.