Project 1: Distributed Mutual Exclusion

deadline: *March 4, noon*

Introduction

In this project there will be two implementations a distributed mutual exclusion algorithm. Each version can be either Ricart&Agrawala or Maekawa for tokenless protocols, and any token passing protocol (e.g., Raymond's tree-based algorithm). This distributed algorithm will get the necessary permissions to update a central database (actually, just a file on /afs space, maybe a website). Several processes will communicate with each other in different machines, and this will be done through servers running on these machines. Remember that the alternative is to have a sequencer (centralized approach), which may partially be used here.

The potential applications for distributed ME are numerous and include: sign-up sheet for the project, distributed control of scheduling appointments, distributed control of DBs (e.g., airline reservation system), distributed administration of mailing lists. We will implement a facility that updates a database of lectures in the CS dept, which can be seen by anyone running a specific client program: `today`. This facility can also be used to provide room reservations.

The software you will build updates the database in two ways: (a) using an interactive procedure to request information from the user, collect the information, and then update the database when all information is available, or (b) reading the information from a file, and then updating the database. You are to implement both ways.

This assignment is to be done in pairs: one student will implement the tokenless protocol while the other will implement the token-based protocol. These algorithms will be compared with each other in terms of performance (e.g., number of messages and synchronization delay). You must send mosse@cs.pitt.edu an email by Feb 20 with your group members, who will implement each algorithm and the metrics you will use.

Technical Details

There is a central database file residing in your public space (you should test your program with your own version of the file). We will consider an unbounded number of clients per machine in this project.

Several clients can connect to one of many possible servers, and servers communicate among themselves. You may assume that there is a well-known name server and there are up to n servers and that the server machines are well known to all servers and clients in the group (read the list of servers from afs space). Depending on the algorithm implemented, the interconnection among servers will be different.

Each server that needs to update or read the database, must go into a negotiation phase, during which the distributed ME algorithm will be executed. After getting permission to enter the Critical Session (CS), the server can authorize a client to update the database. Upon update completion, the server is notified. Alternatively the server can get requests from the clients and update the database.

Since many clients may be simply reading the database, your protocol should be able to (consistently) resolve the conflicts of the read/write possibilities. It should not block the readers if there are

only readers accessing the database (any solution for the readers/writers problem is acceptable).

Here's an example of the output to be displayed. Clearly, you are free to customize your own interface, if you don't like the one below.

```
Monday, February 19, 2002
 10:00 AM       K. K. Ramakrishnan
 EB 332         AT&T Bell Laboratories
                ``Operating System Support for Multimedia Servers''
                CS Seminar Series
                For more information, contact Daniel Mosse
                at 624-2772

 3:00 PM        T. V. Lakshman
 EB 332         AT&T Bell Laboratories
                ``Modeling Variable Bit Rate Video Traffic Sources''
                Geek Club Seminar Series
                The talk will be followed by a reception
                For more information, contact Daniel Mosse
                at 624-2772 or mosse@cs.pitt.edu

Tuesday, February 20, 1996
...
```

Therefore, 11 database fields: day, date, time, location, speaker, speaker's institution, title, additional info, contact name, contact e-mail, and contact phone number. The message format has four components: client id, server id, time of request, and type of request. You also need to display this information back on each client's screen, to verify correctness.

The user interface is up to you (web, plain text, etc). The best design and implementation will be considered (by Daniel Mossé!) to be installed in the Department for announcing new lectures and reservation of rooms.

Approval pending, must be registered to participate, all proceeds benefit my child's college education if she decides to go to a private university.