Project 2

Assigned Oct. 30, 2015
Work in groups of two.

Description
In project 1, you implemented a distributed calendar using the Wu-Bernstein replicated log and dictionary algorithm. This algorithm is asynchronous, and log information need only be shared after an event is scheduled. This algorithm has the benefits of simplicity and low message overhead, but it also allowed conflicting events to be scheduled. Thus, you needed to implement a conflict resolution algorithm that must be resolved via an additional mechanism.

In this project, you will implement a similar distributed calendar application. The specification of how to schedule and cancel events is the same as for project 1.

The system consists of N nodes, each node corresponding to a single user. Each node stores its own local calendar data structure, and you must implement a distributed algorithm to schedule events and share calendar information among users using a replicated log. The log entry creation and replication will be done using the Paxos algorithm in a way that prevents the scheduling of conflicting events.

More details will be given in class.

Paxos Details and Partial Credit
The Paxos algorithm consists of two main phases, a leader election phase and an atomic broadcast phase, initiated by the leader. The complete Paxos implementation must include both phases. You should use TCP for the leader election phase and UDP for the atomic broadcast phase. A suggestion is to first implement the atomic broadcast phase, assuming the leader never fails. 70% credit will be given for a correct implementation of this. You can then implement the leader election phase, which is 30% of the project grade.

Implementation Details
You must implement a system with five nodes. Each node will be run on a different machine. The log should be stored on disk so that it will survive node crashes. The calendar should be kept in memory and reconstructed from the log on crash recovery. All other state information should be stored as required by the Paxos specification.

You need to provide, at least, a minimal UI to view, insert, and delete appointments. A text-based UI is fine. The contents of the log must also be viewable. This can be done by reading a file, so long as the file is human-readable.

You will demo your project on Amazon EC2, on five machines in five different regions. You will use micro-instances. You can use any programming language supported by this platform that supports the message-passing model and does not mask failures (e.g., C, C++, Java, Python).
Deliverables and Due Dates

Project Due Sunday, Dec. 6, 2015 at 11:00pm.

Turn in your source code and a 2-page project report. The report should provide details of your design and implementation. The code and report will be submitted using the RPI Homework Server. Details will be provided closer to the due date. No late turn-ins will be accepted.

We will schedule project demonstrations for the week of Dec. 7 – Dec. 11, 2015

Based on a project by S. Das and A. El Abbadi