

COMP 4704 – Spring 2009

Class 7 Exercises

Instructor:

Prof. Matt Rutherford – mjr@cs.du.edu

May 14, 2009

Totally Ordered Multicast for Leader Election

In this exercise, we will be working with the Bully leader election algorithm (pp. 264 – 265). This algorithm works as follows:

1. When a process P notices that the current leader is down, it sends an ELECTION message to all processes with higher identifiers.
2. If no one responds to the ELECTION message, P becomes the leader and sends out a COORDINATOR message.
3. Any processes that receive an ELECTION message return OK to the sender and then initiate their own ELECTION round.
4. **Special Case** – When the process with the highest possible identifier initiates, it always sends COORDINATOR to take over the leader role.

Consider the situation where there are 4 processes labeled 0 through 3. P_3 is initially down, and the other processes have run an election up to the point that P_2 is just sending COORDINATOR. At that very instant, P_3 starts back up and sends COORDINATOR.

In this situation it is critically important that all nodes process the COORDINATOR messages in precisely the same order. **Assuming the processes are coded such that they automatically accept the sender of the most recent COORDINATOR as the leader, it is not critical which message is processed first, only that everyone processes them in the same order.** If they process them differently, there is the possibility that there are 0 or 2 leaders, and that the non-leaders are mixed in who they treat as leader.

Exercise

We are going to use Lamport's Logical Clocks to implement Totally Ordered Multicast so that all processes handle the messages in the same order. Use the following initial Logical Clock Values for the processes:

- P_0 – 11
- P_1 – 22
- P_2 – 33
- P_3 – 44

Start off the exercise by having P_2 and P_3 send COORDINATOR messages concurrently. Keep track of the message queue at each process in a table with columns for the sending timestamp, the sending process, and the message body. Each process is using the logical clock algorithm to maintain their own logical clock. The queue should be sorted according to the message timestamp with the process ID used as a tiebreaker. Messages are only processed when they are at the front of the queue **AND** they have been acknowledged by all processes.