CS 6/73201

Homework #3

Advanced OS

Due in class on Wednesday 11 April 2001

1. **(Exercise 6.13 from *Distributed Operating Systems*) The first general algorithm for implementing mutual exclusion in a distributed environment was developed by Lamport [1978]. Find the details of this algorithm and compare its performance and reliability with that of Ricart and Agrawala's [1981] algorithm.**

   Performance: Lamport's algorithm requires 3(N–1) messages to enter the critical section, while R&A's algorithm requires only 2(N–1) messages. It has also been proven that R&A's algorithm is deadlock-free.

   Reliability: Both algorithms have a distributed performance bottleneck, and multiple points of failure.

2. **(Exercise 6.35 from *Distributed Operating Systems*) Why are election algorithms normally needed in a distributed system? A LAN based distributed algorithm has broadcast facility. Suggest a simple election algorithm for use in this system.**

   Election algorithms are needed to elect a leader in algorithms that require a central coordinator, or to elect someone to generate a token in algorithms that require a token. This election may be required when the algorithm begins normally, or if the coordinator happens to fail or the token gets lost.

   The Bully algorithm can easily be modified to broadcast an election message to everyone, and to broadcast the coordinator message to everyone. Further improvements involving snooping on elections in progress could reduce the number of elections in progress at the same time.

3. **(Exercise 6.26 from *Distributed Operating Systems*) Discuss why advance knowledge of the resource use of processes is essential to avoid deadlocks. Why is the deadlock avoidance strategy never used in distributed systems for handling deadlocks?**

   Deadlock avoidance algorithms use knowledge of the current state of resource assignment, plus anticipated future usage, to decide whether or not to honor a request for additional resources — only resource requests that can be guaranteed not to cause deadlock are honored.

   These algorithms are not used in distributed systems for the same reason they are not used in centralized systems — they are too computationally-intensive and add too much overhead to the system, as they have to be run for every resource request. Furthermore, gathering this advance knowledge is difficult if not impossible. Finally, most of these algorithms require a static set of processes and resources, which is also unrealistic.

4. **Use the technique described in Lecture 27 for serializability testing to show that the schedule on the left side of Figure 9.10 from *Distributed Operating Systems* is not serializable.**

   Diagram should show: arrow pointing right from a2 to b4, arrow pointing left from b2 to a4. Since the arrows point in opposite directions, the schedule is not serializable.

5. **(Exercise 9.30 from *Distributed Operating Systems*) What advantages does nested transactions facility have over traditional transaction facility in a distributed system? Answer the following questions for a nested transaction family:**

   They can provide nice modularity and transparency, with the client only having to contact one server (which contacts other servers itself) instead of having to contact multiple servers. Of course, both nested and distributed transactions can provide more concurrency than a transaction executed on a single server, and provides better protection against failures in that if only part of the transaction fails, that part may be re-executed.

   a. **Can a transaction commit independent of other transactions in the family?**

      No, the whole family of transactions must commit as an atomic unit.

   b. **Can a transaction abort independent of other transactions in the family?**

      Yes, but if the parent aborts, the children must also abort, whereas if a child aborts, the parent can then decide whether to abort or continue (possibly retrying that aborted transaction).

   c. **When do the changes made to data items by a transaction become visible to other transactions in the family?**

      Ideally the children commit at the same time, upon being told by the parent to commit.

   d. **What happens if a transaction in the family fails?**

      If the parent fails, the children should abort. If a child aborts, the parent can then decide whether to abort or continue (possibly retrying that aborted transaction)

   e. **When are the updates performed by the transaction made permanent?**

      Once the parent (and thus the transaction as a whole) commits.

   f. **When does the entire transaction family commit and success is reported to the client?**

      When the parent commits.