17 - Synchronisation 2

Tutor version

Dr Leonardo Mostarda,
School of Science and Technology,
Camerino University, Italy
Version 1.0, 10 Mar 2013

Question 1

Describe the Bully Algorithm in details.

Solution 1

The Bully Algorithm P sends an ELECTION message to all processes with higher numbers. If no one responds, P wins the election and becomes coordinator. If one of the higher-ups answers, it takes over. P's job is done.

Question 2

Describe Ricart and Agrawala's algorithm.

Solution 2

When a process wants to access a shared resource, it builds a message containing the name of the resource, its process number, and the current (logical) time. It then sends the message to all other processes, conceptually including itself. The sending of messages is assumed to be reliable; that is, no message is lost. When a process receives a request message from another process, the action it takes depends on its own state with respect to the resource named in the message. Three different cases have to be clearly distinguished:

1. If the receiver is not accessing the resource and does not want to access it, it sends back an OK message to the sender.

2. If the receiver already has access to the resource, it simply does not reply. Instead, it queues the request.

3. If the receiver wants to access the resource as well but has not yet done so, it compares the timestamp of the incoming message with me. one contained in the message that it has sent everyone. The lowest one wins. If the incoming message has a lower timestamp, the receiver sends back an OK message. If its own message has a lower timestamp, the receiver queues the incoming request and sends nothing.

After sending out requests asking permission, a process sits back and waits until everyone else has given permission. As soon as all the permissions

are in, it may go ahead. When it is finished, it sends OK messages to all processes on its queue and deletes them all from the queue.

Question 3

In the centralised approach to mutual exclusion, upon receiving a message from a process releasing its exclusive access to the resources it was using, the coordinator normally grants permission to the first process on the queue. Give another possible algorithm for the coordinator.

Solution 3

Requests could be associated with priority levels, depending on their importance. The coordinator could then grant the highest priority request first.

Question 4

Consider the centralised approach to mutual exclusion again. Suppose that the coordinator crashes. Does this always bring the system down? If not, under what circumstances does this happen? Is there any way to avoid the problem and make the system able to tolerate coordinator crashes?

Solution 4

Suppose that the algorithm is such that every request is answered immediately, either with permission or with denial. If there are no processes accessing resources and no processes queued, then a crash is not fatal. The next process to request permission will fail to get any reply at all, and can initiate the election of a new coordinator. The system can be made even more robust by having the coordinator store every incoming request on disk before sending back a reply. In this way, in the event of a crash, the new coordinator can reconstruct the list of accessed resources and the queue by reading the file from the disk.

Question 5

Ricart and Agrawala's algorithm has the problem that if a process has crashed and does not reply to a request from another process to access a resources, the lack of response will be interpreted as denial of permission. We suggested that all requests be answered immediately to make it easy to detect crashed processes. Are there any circumstances where even this method is insufficient? Discuss.

Solution 5

Suppose that a process denies permission and then crashes. The requesting process thinks that it is alive, but permission will never come. One way out is to have the requester not actually block, but rather go to sleep for a fixed period of time, after which it polls all processes that have denied permission to see if they are still running.