

COMP 4704 – Spring 2009

Class 6 Exercises

Instructor:

Prof. Matt Rutherford – mjr@cs.du.edu

May 7, 2009

In the following exercises you will simulate, on paper, the operation of two mutual exclusion algorithms: centralized and distributed. You will simulate the behavior of 5 processes, labeled 0 through 4 for the scenarios described below.

Centralized Algorithm

In the centralized mutual exclusion algorithm, one process is designated as the **COORDINATOR**; for this exercise assume that this is process 0. When a process wants access to the resource being coordinated by 0, it issues a **REQUEST** message to the coordinator. When the coordinator is sure no other processes are using the resource, it sends an **OK** message to the process requesting the resource. When a process is done using a resource it sends a **RELEASE** message to the coordinator. The coordinator queues requests that arrive when the resource is in use.

Please write down the messages sent in the following scenarios. For each message, write down the sending process, the receiving process, and the message content.

Also write down the state of any queues maintained by the processes during the scenario.

Each algorithm takes place in “message rounds” during which processes send the next message they are supposed to, as determined by the algorithm.

Scenario 1

1. Process 1 requests access to the resource.
2. Process 2 requests access to the resource while process 1 is busy using it.
3. Process 3 requests access to the resource while process 1 is busy using it.
4. Process 1 eventually finishes using the resource.
5. Process 2 eventually finishes using the resource.
6. Process 3 eventually finishes using the resource.

Scenario 2

1. Process 1 requests access to the resource.
2. Process 2 requests access to the resource while process 1 is busy using it.
3. Process 1 crashes while accessing the resource.

Distributed Algorithm

In the distributed algorithm, all processes contribute to the mutual exclusion algorithm. Assume there is a resource named **R1**. When a process wants to access a resource, it sends out a request with the resource name, its ID, and its timestamp; the request goes to all other processes.

When a process receives a request, it does one of the following:

- If the receiver is not accessing the resource and does not have a pending request, it sends back an **OK** message.
- If the receiver is currently accessing the resource, it queues the request.
- If the receiver has a pending request it makes the following determination:
 - If the timestamp of its request is lower than the received one, it queues the request
 - If the incoming request has a lower timestamp it immediately responds **OK**.

Scenario 1

1. Processes 1 and 2 request access to the resource. Process 1 has the lower timestamp.
2. Process 1 eventually finishes using the resource.
3. Process 2 eventually finishes using the resource.

Scenario 2

1. Process 3 crashes
2. Process 1 requests access to the resource.