LTAT.06.007 Distributed Systems
Seminar 8 - Coordination II (Mutual exclusion)

Mohan Liyanage, PhD
Tartu, Estonia April 8, 2020
Quiz 8

Questions are based on the Coordination II (Mutual exclusion) lecture

Short answers

Link to quiz – https://tinyurl.com/ua05jw6

Quiz will be available until April-06-2020:23.59
1. Explain how the token-based mutual exclusion algorithm works.

In token-based solutions mutual exclusion is achieved by passing a special message between the processes, known as a token. There is only one token available and whoever has that token is allowed to access the shared resource. When finished, the token is passed on to a next process. If a process having the token is not interested in accessing the resource, it passes it on.
2. What are the limitations of the token-based mutual exclusion algorithm?

The main drawback of token-based solutions is a rather serious one: when the token is lost (e.g., because the process holding it crashed), an intricate distributed procedure needs to be started to ensure that a new token is created, but above all, that it is also the only token.
3. This figure shows how the permission-based, centralized mutual exclusion algorithm works. Briefly explain the main steps relevant to a, b, and c.

Figure 6.15: (a) Process $P_1$ asks for permission to access a shared resource. Permission is granted. (b) Process $P_2$ asks permission to access the same resource, but receives no reply. (c) When $P_1$ releases the resource, the coordinator replies to $P_2$. 
4. What are the main drawbacks of the permission-based, centralized algorithm?

The coordinator is a single point of failure, so if it crashes, the entire system may go down. If processes normally block after making a request, they cannot distinguish a dead coordinator from “permission denied” since in both cases no message comes back. In addition, in a large system, a single coordinator can become a performance bottleneck.
5. What is the main requirement for Ricart and Agrawala [1981], the permission-based distributed algorithm to works?

This algorithm requires a total ordering of all events in the system. That is, for any pair of events, such as messages, it must be unambiguous which one actually happened first.
6. Concerning the Ricart and Agrawala algorithm, what are actions that a process can perform if it receives a request message from another process to access a shared resource (three actions)?

- If the receiver is not accessing the resource and does not want to access it, it sends back an OK message to the sender.
- If the receiver already has access to the resource, it simply does not reply. Instead, it queues the request.
- 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 the 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.
7. Briefly explain how two processes simultaneously access the same resource by referring to this figure.

Figure 6.16: (a) Two processes want to access a shared resource at the same moment. (b) $P_0$ has the lowest timestamp, so it wins. (c) When process $P_0$ is done, it sends an OK also, so $P_2$ can now go ahead.
8. What are the Pros, Cons of Ricart & Agrawala algorithm?

• Pros
  • Mutual exclusion is guaranteed without deadlock or starvation
  • No single point of failure

• Cons
  • N points of failure (crashed node interpreted as denial of access)
  • Requires more communication
  • Low efficiency, as all processes are involved in all decisions (n bottlenecks)
9. What is the main purpose of mutual exclusion election algorithms?

In general, election algorithms attempt to locate the process with the highest identifier and designate it as coordinator.
10. This figure shows the bully algorithm which is used as an election algorithm. Please explain the main steps (a, b, c, d)

The bully algorithm

A well-known solution for electing a coordinator is the bully algorithm devised by Garcia-Molina [1982]. In the following, we consider $N$ processes $\{P_0, \ldots, P_{N-1}\}$ and let $id(P_k) = k$. When any process notices that the coordinator is no longer responding to requests, it initiates an election. A process $P_k$, holds an election as follows:

1. $P_k$ sends an ELECTION message to all processes with higher identifiers: $P_{k+1}, P_{k+2}, \ldots, P_{N-1}$.
2. If no one responds, $P_k$ wins the election and becomes coordinator.
3. If one of the higher-ups answers, it takes over and $P_k$’s job is done.

Figure 6.20: The bully election algorithm. (a) Process 4 holds an election. (b) Processes 5 and 6 respond, telling 4 to stop. (c) Now 5 and 6 each hold an election. (d) Process 6 tells 5 to stop. (e) Process 6 wins and tells everyone.
11. This figure shows the main steps of the election algorithm in a ring. The solid line shows the election messages initiated by P6; the dashed one those by P3. Please explain the main steps accordingly.

![Election algorithm using a ring. The solid line shows the election messages initiated by P6; the dashed one those by P3.](image)

**Figure 6.21**: Election algorithm using a ring. The solid line shows the election messages initiated by $P_A$; the dashed one those by $P_2$.

In Figure 6.21 we see what happens if two processes, $P_3$ and $P_6$, discover simultaneously that the previous coordinator, process $P_7$, has crashed. Each of these builds an ELECTION message and each of them starts circulating its message, independent of the other one. Eventually, both messages will go all the way around, and both $P_3$ and $P_6$ will convert them into COORDINATOR messages, with exactly the same members and in the same order. When both have gone around again, both will be removed. It does no harm to have extra messages circulating; at worst it consumes a little bandwidth, but this is not considered wasteful.
Thank You !!