Resit Distributed Algorithms

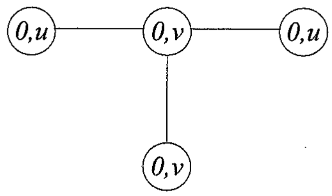
Free University Amsterdam, 28 August 2009, 8:45-11:30

(At this exam, you may use the book Introduction to Distributed Algorithms by Gerard Tel, the handouts from Attiya & Welch and Liu, and copies of the slides without handwritten comments. Answers can be given in English or Dutch. Use of a laptop is not allowed.)

(The exercises in this exam sum up to 90 points; each student gets 10 points bonus.)

1. Give an example to show that the Dijkstra-Feijen-van Gasteren algorithm for termination detection in a unidirectional network does not work in case of asynchronous communication. (6 pts)

2. Apply the echo algorithm with extinction to elect a leader in the following anonymous undirected network, where initial random identities have been chosen (and each process is at level 0). All processes are initiators, and know that the network size is 4.



   Let $u < v$. Give a scenario in which the process at the bottom becomes the leader at level 1. Explain why, in such a scenario, at most one of the nodes $(0, u)$ will progress to level 1. (10 pts)

3. Consider the Dolev-Klawe-Rodeh algorithm for leader election in directed rings. Give a probabilistic version of this algorithm to get a Las Vegas algorithm for leader election in *anonymous* directed rings (of known size). It should terminate with probability 1; explain why this is the case. (10 pts)

4. We adapt the Bracha-Toueg algorithm for t-Byzantine consensus by allowing a correct process to decide b if it accepts *at least* (instead of more than) $\frac{N+t}{2}$ b-votes in one update round.

Consider a *complete* network G (i.e., there is a channel between each pair of different processes) of five processes. Let three processes hold the value 0, while two processes hold the value 1. Apply the adapted version of the Bracha-Toueg algorithm for 1-Byzantine consensus to G, and show that it can lead to inconsistent decisions. (12 pts)

5. In the t-Byzantine robust synchronizer of Lamport and Melliar-Smith, a correct process p accepts a local clock value of another process q if it differs no more than δ from its own clock value, at the moment of synchronization. Explain in detail why that synchronizer has precision $\frac{3t}{N}\delta$ (versus precision $\frac{2t}{N}\delta$ of the Mahaney-Schneider synchronizer). (14 pts)

6. Consider the Mellor-Crummey-Scott lock. Suppose that when a process q exits the critical section and finds that $\mathit{next}_q = \perp$, it only performs $\mathit{compare\text{-}and\text{-}swap}(q, \perp)$ on *last*, and undertakes no further action when $\mathit{last} \neq q$. Give a scenario to show that in that case the algorithm could deadlock. (10 pts)

7. Let preemptive jobs J_1, J_2 and J_3 arrive at times 2, 1 and 0, respectively, with execution time 2. Let the priorities be $J_1 > J_2 > J_3$. Let J_1 and J_3 use resource R for their entire execution. The jobs are executed using priority ceiling.

   How are the three jobs executed if the arrival of J_1 is known from the start? And how are they executed if the arrival of J_1 is not known before time 2? (9 pts)

8. One part of the Arora-Gouda self-stabilizing leader election algorithm says that if $\mathit{leader}_i < \mathit{leader}_j$ where $j \in \mathit{Neigh}_i$ and $\mathit{dist}_j < K$, then

$$\mathit{leader}_i := \mathit{leader}_j \qquad \mathit{father}_i := j \qquad \mathit{dist}_i := \mathit{dist}_j + 1$$

   Show that if the condition "and $\mathit{dist}_j < K$" were omitted here, then the algorithm might not stabilize. (12 pts)

9. In weighted reference counting, why is the possibility of an underflow (weight 1) a much more serious problem than the possibility of an overflow of a reference counter? (7 pts)