The University of Melbourne
School of Computing and Information Systems
COMP90020 Distributed Algorithms

Tutorial Week 5: Mutual Exclusion and Failure Detection

Notes

In this tutorial we will look at mutual exclusion and briefly at failure detectors. Both of these topics have much more to offer. In particular, we will only examine mutual exclusion using message passing and won't look at algorithms that use shared memory.

If you're interested to learn more broadly on these topics, a great place to start is Lamport's bakery algorithm and further continue onto queue, CLH and MCS locks.

Exercises

18. Is it possible to implement either a reliable or an unreliable (process) failure detector using an unreliable communication channel?

19. Say for both mutual exclusion and startvation-freeness whether it is a safety or liveness property.

20. In the central server algorithm for mutual exclusion, describe a situation in which two requests are not processed in happened-before order.

21. Give an example execution of the ring-based algorithm to show that processes are not necessarily granted entry to the critical section in happened-before order

22. Consider running Maekawa's algorithm on a system of 13 processes. Figure out the composition of the 13 subsets S_0 - S_{12}, so that (1) each subset includes four processes, (2) there are exactly four subsets, and (3) process $i \in S_i$.

23. The L-exclusion problem is a generalized version of the mutual exclusion problem in which up to L processes $L \geq 1$ are allowed to be in their critical sections simultaneously. Precisely, if fewer than L processes are in the CS at any time and one more process wants to enter its critical section, then it must be allowed to do so. Modify Ricart–Agrawala's algorithm to solve the L-exclusion problem.

24. In a network of processes, the *local mutual exclusion* problem guarantees that no two neighbors execute a critical action at the same time. Extend Ricart–Agrawala's to solve the local mutual exclusion problem.