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

Miscellaneous Practice questions

Notes

These exercises are intended to aid in your preparation for the final exam. Note the distribution of topics has no indication of the final exam. The difficulty is likewise varied, with some questions being easier whilst others near the limit of what might be expected.

No solutions are provided however you are welcome to crowd-source (discuss) them on the discussion forum with your peers.

Exercises

The following algorithm description relates to the next two questions:

In a asynchronous bidirectional message-passing network that is a complete graph. Initially there are n nodes, of which m have a token. It is necessary to move tokens, each node can only have a single token or none.

2. Can you guarantee that there are exactly m tokens in any reachable configuration? Show why or why not.

3. Provide an algorithm that satifies (a) safety: in any reachable configuration, there are at most m tokens. (b) liveness: from any reachable configuration C_0, for any subset S of the processes who currently hold a token such that $|S| = m$, there exists an execution starting in C_0 that ends in a configuration with which every node in S has a token.

4. A hypercube (somtimes called n-cube) is a common network topology due to its desirable diameter to node ratio. It is a graph in which each node has degree d and there are $n = 2^d$ nodes. Suppose you wish to adapt an asynchronous ring based algorithm onto the hupercube toplogy. How can you do this without an increase in message complexity? (it may help to define an encoding that represents a node in the hypercube, e.g. bits)

5. Suppose you now wish to generalize your ring algorithm to any arbitrary graph. What issues might you face?

The following description relates to question 6:

In "comp99999: Very distributed algorithms", there are n students arranged in a ring $student_0, ..., student_{n-1}$. In the initial configuration, student 0 has n copies of the solution to the final exam (who knows how they got it...some suggest the coordinators incorrectly thought their communication was free of byzantine actors). Additionally, the copies cannot be duplicated and there remains $\leq n$ copies in any reachable configuration. The students take steps asynchronously. Whenever $student_i$ takes a step in a configuration where they have a copy of the solutions to the exam but their successor $student_{i+1 mod n}$ does not, the student will give them a copy. Else nothing happens. We can assume that fairness ensures that all students take a step infinitely often.

6. Show that after some finite number of steps, every student will have a copy of the exams.

7. Suppose there is an asynchronous network of nodes organized in a connected graph where all nodes are nearly homogenous and cannot fail. They run the same algorithm, however every node starts with an id and knowledge of its neighbours ids. Each of these ids is unique except that for the minimum id, which could be assigned to (configuration C_a) one or (configuration C_b) two nodes. Provide an algorithm that determines whether the system is initially in C_a or C_b, or show that no such algorithm can exist.

8. Suppose there is an asynchronous network of nodes organized in a fully connected graph. There exists 1 byzantine node. A non-faulty node should be able to execute an algorithm to obtain a monotonically increasing timestamp. Such that, it should be guarantee that when a node is non-faulty, it eventually obtains a timestamp ts_i that is larger than any previously returned $i > j \implies ts_i > ts_j$ (where $ts_1, ..., ts_m$ and m is finite) by a non-faulty process that finished before the execution of the algorithm started. Assume n is the number of nodes in the network, for what values of n is the problem not able to be solved.

9. Vector clocks are useful for identifying concurrent and causally ordered events. However the algorithm provided in class grows linearly with the number of processes n. Is it possible to detect causality using vector clocks of a smaller size? Justify

10. Assume there is an anonymous network, where processes do not have identifiers. Is it possible to define a total ordering among events?

11. Extend chandy-lamport algorithm so that it can operate on a network topology that is a DAG. What limitations exist?

12. Does chandy-lamport always work correctly when the channels it is using are not CO (causally) ordered?

13. Assume that there are physical clocks that are synchronized (bound known) and the delays between channels are known. Propose a distributed snapshot algorithm that works under these assumptions. Assume the network is fully connected.

14. Assume there is an asynchronous network of nodes organized in a completely connected graph. Every node has a perfect failure detector. Describe how you can elect a leader under these assumptions.

15. Consider two groups A and B each containing a set of members. Within each group, total order (TO) multicast and causal order (CO) multicast have already been implemented. Assume now that some members of group A became members of group B as well. Will CO multicast work? Will TO multicast work?

16. The logical clock values used in Ricart-Agrawala algorithm are unbounded. Propose a modification to the algorithm such that the range of these values becomes bounded. Discuss correctness.

17. Assume there is an anonymous network of nodes organized in an acylic graph. Is it possible to determine the size (number of nodes)?

18. Argue that a consistent snapshot rules out orphan messages.

19. Consider a symmetric network of nodes which starts in an initial configuration in which equivalent nodes have the same state. Argue that if the system is deterministic then there is a syncrhonous execution in which equivalent nodes continue to have the same state.

For the following questions we will adopt a slightly different notation for transactions:

Let $q_i(x), q \in w, r$ denote a write or read operation by transaction t_i on object x. Let c_i represent a commit operation done by transaction $t_i, 1 \leq i \leq n$ Let $T = \{t_1, ..., t_n\}$ be a finite set of transactions, $1 \leq i \leq n$

Additionally, we will only consider conflict serializability (which we did in class).

Summarily, conflict serializability prevents lost updates and inconsistent reads but does not prevent dirty reads or premature writes, we need commit (strict) serializability for that:

We derive $D_2(s) := (V, E)$ where $V = op(s)$ and $E = conf(s)$. e.g. each operation is represented by a node, where edges between nodes represent a conflict. This is called a conflict step graph.

$s \approx_c s' \Leftrightarrow D_2(s) = D_2(s')$

A history s is conflict serializable if there exists a serial history s' such that $s \approx_c s'$.

20. For each of the following schedules determine whether the history is serializable.

(a) $s_1 = w_1(x)w_2(x)w_2(y)c_2w_1(y)c_1$

(b) $s_2 = w_1(x)r_2(x)w_2(y)c_2r_1(y)w_1(y)c_1w_3(x)w_3(y)c_3$

(c) $s_3 = w_1(x)r_2(x)w_2(y)w_1(y)c_1c_2$

(d) $s_4 = w_1(x)w_2(x)w_2(y)c_2w_1(y)c_1w_3(x)w_3(y)c_3$

(e) $s_5 = w_1(x)r_2(x)w_2(y)w_1(y)c_1c_2w_3(x)w_3(y)c_3$

(f) $s_6 = w_1(x)w_2(x)w_2(y)c_2w_1(y)w_3(x)w_3(y)c_3w_1(z)c_1$

(g) $s_7 = w_1(x)w_2(x)w_2(y)c_2w_1(z)c_1$

(h) $s_8 = w_3(y)c_3w_1(x)r_2(x)c_2w_1(y)c_1$

(i) $s_9 = w_3(y)c_3w_1(x)r_2(x)w_1(y)c_1c_2$

(j) $s_{10} = w_1(x)w_1(y)c_1w_2(x)w_2(y)c_2$

21. Enumerate the lock compatibility table for $rl(x)$ and $wl(x)$ for two contending transactions.

22. Can 2PL produce all schedules that are serial?

23. Assume the lock scheduler works from left to right, show the outpput a 2Pl locking schedule for each of the following input schedules. If unable, note why.

(a) $s_1 = w_1(x)r_2(y)r_1(x)c_1r_2(x)w_2(y)c_2$

(b) $s_2 = r_1(x)r_2(x)w_3(x)w_4(x)w_1(x)c_1, w_2(x), c_2, c_3, c_4$

24. A lock point of a transaction t denotes the point in time at which t has obtained all locks it needs,but has not yet released any. Show that for each history s produced by 2PL there exists a serial history s' in which all transactions occur in the same order of lock points as in s.

25. Descirbe the waits-for graph resulting from the use of 2PL for each of the histories in 23.

26. Consider the following local histories of transactions in a distributed system, state for each whether it is locally serializable at all sites and globally serializable. Note here that s_1 denotes the site local history of 1.

(a)

- $s_1 = r_1(a)w_2(a)$
- $s_2 = r_3(b)w_1(b)r_2(c)w_3(c)$

(b)

- $s_1 = r_1(a)r_3(a)r_3(b)w_3(a)w_3(b)r_2(b)$
- $s_2 = r_2(c)r_4(c)r_4(d)w_4(c)w_4(d)r_1(d)$

(c)

- $s_1 = w_1(a)r_3(a)w_3(b)r_2(b)$
- $s_2 = r_2(c)w_4(d)r_1(d)w_2(e)r_4(e)$

(d)

- $s_1 = w_1(a)r_3(a)r_3(b)w_2(b)$
- $s_2 = w_2(c)r_4(c)r_4(d)w_1(d)$

27. Can a general distributed commit protocol ensure independent process recovery in the presence of multiple faults? ($f > 1$)