Distributed Algorithms (COMP90020_2020_SM1)

Started: Jul 2 at 9:01

Quiz Instructions

Academic Integrity Declaration

By commencing and/or submitting this assessment I agree that I have read and understood the University's policy on academic integrity.

I also agree that:

1. Unless paragraph 2 applies, the work I submit will be original and solely my own work (cheating);
2. I will not seek or receive any assistance from any other person (collusion) except where the work is for a designated collaborative task, in which case the individual contributions will be indicated; and,
3. I will not use any sources without proper acknowledgment or referencing (plagiarism).
4. Where the work I submit is a computer program or code, I will ensure that:
   a. any code I have copied is clearly noted by identifying the source of that code at the start of the program or in a header file or, that comments inline identify the start and end of the copied code; and
   b. any modifications to code sourced from elsewhere will be commented upon to show the nature of the modification.

This exam begins at 9:00 AM Australian Eastern Standard Time (AEST) on Thursday 02/07/2020 in Canvas (lms.unimelb.edu.au). The exam must be completed by 12.30 PM AEST on Thursday 02/07/2020. This exam has 180 minutes of writing time and 30 minutes of technical buffer.

Exam Instructions

Writing time: 3 hours

Technical buffer time: 30 minutes

Total marks for this exam: 60

Authorized materials:

This is an **open book** assessment.

You are **allowed to:**

- Use the textbook, lecture slides, and any other reading materials used or referenced throughout the course.
- Use explanations or examples given in the lectures.
- Use the solutions given for all the tutorial questions.
- Use your own notes.

While you are undertaking this assessment you **must not:**

- Use any messaging or communications technology.
- Act in any manner that could be regarded as providing assistance to another student who is undertaking this assessment, or will in the future be undertaking this assessment.

The work you submit must be based on **your own knowledge and skills**, without assistance from any other person.

Instructions to Students:

- There are **10 questions** in this exam, **attempt all** of them.
- All questions must be answered in their corresponding text box.
- You can enter text, images, and upload documents as part of your answer.
- Ensure handwritten answers are **legible**.
- For further information on Canvas Quizzes please see: https://students.unimelb.edu.au/your-course/manage-your-course/exams-assessments-and-results/exams/exam-types.

Questions

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Question 8

Time Running: Hide
Attempt due: Jul 2 at 12:30
11 Minutes, 32 Seconds

Question 1

6 pts

Cristian's clock synchronization algorithm assumes symmetric message transmission delays; that is, the transmission delay of the request message (m_r) is expected to be the same as the transmission delay of the response message (m_s). This is not true in all networks. Assume a system where the transmission delay of the request message is 3 times longer than the transmission delay of the response message.

A. How would you modify the formula for setting the clock of a process (C_p)?
B. What would the bound on clock skew be (i.e., the maximum skew)? Assume the minimum transmission delay of the request message is 3 times longer than the minimum transmission delay of the response message.

Show your calculations and notations clearly.

12pt Paragraph

Part A

Old formula is to use half the round trip time since it assumes $m_r = m_s$.

But we only care about how long it took the message to get from the server back (not how long our request took to get there).

If we assume the delay of the request is 3 times the delay of the response, the response delay *should* only be a quarter of the round trip time.

New formula:

$C_p = S + \frac{1}{4}T_{round}$

where S is the time in the message from the server.

Part B

Accuracy will be given by:

$\pm\left(\frac{1}{2}T_{round} - min\right)$

where min is the minimum transmission delay of the response.

p

98 words

Question 2

6 pts

In the Chandy-Lamport snapshot algorithm, the marker sending rule states that after recording its state, a process must send a marker on every outgoing channel before sending other messages. Explain the importance of this rule with regards to the consistency theorem. To help your answer, explain what would happen if after recording its state, a process sends an application message followed by a marker.

12pt Paragraph

Consider a simple example with two processes P and Q. P sends "hello" messages and Q replies with "world" messages. Each keeps track of the number of messages it has sent.

If we don't follow the rule the marker should be sent before application messages, we can get the below situation:

The snapshot is highlighted. Notice that it does not correspond to a consistent cut.

The snapshot cuts execution of process P before it sets h = 1 and sends m0. However, we record message m1. Notice that m0 happened-before m1 so this violates the consistency rule that any event in the cut must have all the events that happened-before the event also in the cut.

As can be seen in the example, it is important to send the marker **immediately** after recording state, before sending application messages to ensure the snapshot corresponds with a consistent cut. Otherwise, we can get snapshots containing e' but not e where $e \rightarrow e'$.

p

156 words

Question 3

6 pts

In the Ricart and Agrawala's algorithm, requests to access the critical section are labeled with (T_i, p_i), where T_i is the Lamport timestamp of the request and p_i is the unique identifier of the process issuing the request. When determining whether $(T_j, p_j) < (T_i, p_i)$, the algorithm compares the Lamport timestamps first and uses the process identifiers to break ties if necessary. Suppose the algorithm is implemented incorrectly: instead of labeling requests with (T_i, p_i) it labels them with (p_i, T_i). This means the algorithm will compare process identifiers first and use the Lamport timestamps to break ties if necessary. The rest of the algorithm is unchanged. Does this algorithm satisfy safety (ME1), liveness (ME2), and/or ordering (ME3)? Justify your answer by explaining why a condition is met or by presenting a counterexample.

12pt Paragraph

Safety (ME1) is satisfied.

Ricart and Agrawala's algorithm (R&A) uses the (T_i, p_i) labelling to order requests which includes p_i to break ties on T_i. Hence, every request has a **unique** (T_i, p_i). Therefore, if we swap the order of the tuple to (p_i, T_i), it will **still** be unique for each request and the tuples will have a total order.

We can see that at most one process would be able to access the critical section (CS) at a time. Only thing is that lower process IDs will have priority now.

Liveness (ME2) may not be satisfied.

R&A involves setting state → WANTED and multicasting a request if a process wants to access the CS. It waits until N-1 replies are received (at which point it can access the CS).

The order of access is changed. If a process wants to access the CS (state = WANTED) and receives a request, it uses the tuple to decide whether to queue the request.

Since we are now using (p_i, T_i), a process with a lower ID will be granted access before a process with a higher ID. This could potentially lead to **starvation** of access for a process with a high ID, in the case where processes with lower IDs keep requesting access to the CS.

If there is no period in time when all processes with lower IDs do not want to access the CS, a process will experience starvation.

Ordering (ME3) is not satisfied.

As explained above, changing the order of the tuple changes the order in which the processes access the CS. When two processes want access to the CS at the same time, access is given to the lower ID process first.

However, it may be the case that the request from the higher ID process happened before the request from the lower ID. Thus, ordering is violated.

p

305 words

Question 4

3 pts

Consider a synchronous distributed system with 5 processes, p_1, p_2, p_3, p_4, and p_5. p_1 is currently the coordinator and when p_5 crashes, it is p_2 that notices the coordinator is down. Suppose the Bully algorithm is used to elect a new coordinator by choosing the process with the largest identifier (i.e., $p_1 < p_2 < p_3 < p_4 < p_5$). Assume there are no failures in the system during the election run. Show the set of all messages that are sent during the election run. Clearly indicate for each message the sending process, the receiving process, and the type of message (i.e., election, coordinator, or answer).

12pt Paragraph

p2 detects failure and sends the following messages and waits for time T:

[election] p2 -> p3
[election] p2 -> p4
[election] p2 -> p5

It receives these replies so waits for another time T:

[answer] p3 -> p2
[answer] p4 -> p4

p3 is doing an election (and waits time T) since it received an election message from p2:

[election] p3 -> p4
[election] p3 -> p5

It receives this reply so waits for another time T:

[answer] p4 -> p3

p4 is doing an election (and waits time T) since it received an election message from p2:

[election] p4 -> p5

But it doesn't receive an answer in the time T. It knows it is coordinator now:

[coordinator] p4 -> p3
[coordinator] p4 -> p2
[coordinator] p4 -> p1

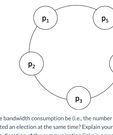
p • strong

131 words

Question 5

9 pts

Consider a distributed system with 5 processes. Assume the processes are arranged in a logical ring with unidirectional clockwise communication links as shown in the figure below. Suppose the ring-based leader election algorithm we discussed in the lectures is used to elect a new coordinator by choosing the process with the largest identifier ($p_1 < p_2 < p_3 < p_4 < p_5$).



A. What would the bandwidth consumption be (i.e., the number of exchanged messages) if all 5 processes initiated an election at the same time? Explain your answer.
B. Now assume the direction of the communication links is reversed so that messages travel in an anticlockwise direction (e.g., p_1 communicates with p_5, p_5 with p_4, and so on). Would the bandwidth consumption change? Explain your answer.

12pt Paragraph

Part A

First, since everyone starts an election at the same time, everyone is a participant.

Consider p5: It sends an election message to p4. Since the ID is greater, it forwards it to the next process p3. Then, p3 does the same and so on. 5 election messages for p5 are exchanged before p5 receives an election message with its ID. Now, it knows it has been elected and will send an elected message to p4 who will forward it to p3 (and so on) until p5 gets an elected message with its own ID. This is another 5 messages.

Now, consider p1: It sends an election message to p5 who does not forward the message since it is a participant and has a higher ID. This is 1 message.

Consider p2: It sends an election message to p1 who forwards it to p5 since the ID is greater. Again, p5 does not forward it since it is a participant and has a higher ID. This is 2 messages.

Similarly, for p3 and p4, 3 messages to elect p3 and 4 messages to elect p4 are exchanged.

The total number of messages exchanged is 5 + 5 + 1 + 2 + 3 + 4 = 20 messages.

Part B

First, since everyone starts an election at the same time, everyone is a participant.

Consider p1: It sends an election message to p2 who does not forward the message since it is a participant and has a higher ID. This is 1 message.

Similarly, for p2, p3 and p4, they each send an election message to a process with a higher ID (who does not forward the message since everyone is a participant). This is 1 message each = 3 messages.

Now, consider p5: It sends an election message to p1 who forwards it to p2 because the ID is higher. This makes it all the way back to p5, involving 5 messages. It now knows it is the coordinator and sends an elected message with its own ID. This is another 5 messages.

The total number of exchanged messages is 1 + 1 + 1 + 1 + 5 + 5 = 14 messages.

p • strong

367 words

Question 6

6 pts

Consider a distributed system that connects every pair of processes through a communication channel. Channels are reliable and guarantee that messages arrive in FIFO order. Assume processes do not fail and that a process delivers a message to itself before sending it out to any other process. If B-multicast was used in this system, would FIFO, causal, and/or total ordering be guaranteed? Justify your answer by explaining why an ordering is guaranteed, or by presenting an example that demonstrates why it is not.

12pt Paragraph

FIFO Ordering would be guaranteed.

Since FIFO ordering is only concerned about the order of multicasts issued by the same process, it would be guaranteed. This is because channels are reliable and guarantee a FIFO order. Thus, if a process issues a multicast m then m', every outbound channel will preserve the m then m' order.

Causal Ordering would

Total Ordering would not be guaranteed.

Consider two processes p0 and p1 in a group.

Suppose p0 does a multicast m0. p0 B-delivers the message m0 straight away.

Soon after, p1 also does a multicast m1 **before** receiving m0. p1 B-delivers m1 straight away and afterwards B-delivers m0 straight away when it receives it.

p0 (who has already delivered m0) now receives m1 and B-delivers it straight away.

Here, p0 delivered m0 then m1 but p1 delivered m1 then m0 so total ordering is not preserved.

p

145 words

Question 7

6 pts

Consider transactions T and U as shown in the figure below. Assume they are running concurrently and that the system uses a strict two-phase hierarchical locking mechanism. Can a deadlock happen in a concurrent execution of these two transactions? Explain your answer.

T	U
read(A)	write(B)
read(B)	read(A)
write(A)	

12pt Paragraph

p

0 words

Question 8

6 pts

A solution to the Byzantine generals (BG) problem is to use a consensus (C) algorithm as follows:

1. The commander P_j sends its proposed value V to itself and each of the remaining processes.
2. All processes run C with the values $V_1, V_2, \ldots, V_N$ that they receive (P_j may be faulty)
3. They derive $BG_i(j, V) = C_i(V_1, V_2, \ldots, V_N)$ $(i = 1, 2, \ldots, N)$

Are the termination, agreement, and integrity conditions of BG preserved in the above algorithm? Justify your answer.

12pt Paragraph

Termination is satisfied.

Since the consensus algorithm guarantees termination (every correct process reaches the decided state), the BG algorithm given will also terminate.

All correct processes will eventually set their decision value.

Agreement is satisfied.

Since the consensus algorithm guarantees that the decision value for all correct processes is the same, it is satisfied also for the BG algorithm.

All correct processes agree on the same decision value

Integrity is satisfied.

For BG, integrity means that if the commander is correct, all correct processes decide on the value proposed by the commander.

If a process is correct in the BG problem, it is not treacherous and proposes the same value it received from the commander.

It follows from the consensus definition of integrity (if all correct processes propose the same value, any correct process has chosen that value) that every correct process in the BG problem must reach the same decision value.

When the commander is correct, all lieutenants propose the same value. Hence, they arrive at the same decision value.

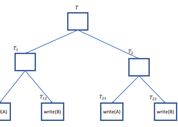
p

171 words

Question 9

6 pts

The graph depicted in the figure below shows two nested transactions (T_1 and T_2) running concurrently. The final operation of each child is written in the leaves. The system uses a strict two-phase hierarchical locking mechanism to regulate concurrency. Assume a parent node releases the I-read or I-write locks after the associated child transaction releases its lock.



A. Give an example of a concurrent execution of T_1 and T_2 that has a dirty read.
B. Give an example of a concurrent execution of T_1 and T_2 that has a premature write.

12pt Paragraph

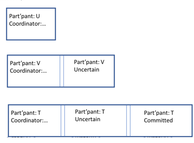
p

0 words

Question 10

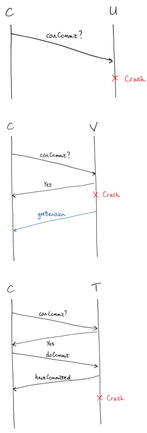
6 pts

Each log file below shows the state of a transaction before it crashed during a two-phase commit protocol. Using a communication diagram (discussed in the lectures), illustrate at what stage of the protocol each transaction has crashed. Show which message participants will send to the coordinator after recovery.



12pt Paragraph

Where message sent after crash, message is in blue.



p

9 words

Saved at 12:18

Submit Quiz