Chalmers Techniska Högskola
Data- och Informationsteknik

CTH TDA297
GU DIT290

Distribuerade system fk

Tentamen 2020-03-18

Dag, *Tid, Sal*: March 18th 2020, 08:30-12:30, Home sweet home

***Kursansvarig*:** Philippas Tsigas (Tel: 772 5409)

Totalt *Poängtal*: 60

***Betygsgränser*:**

CTH: 3:a 30 p, 4:a 38 p, 5:a 48 p

GU: Godkänd 30p, Väl godkänd 48 p

Instructions

- Please answer in English, if possible.
  If you have very big difficulty with that, though, you may answer in Swedish.
- **Do not forget to write your personal number and if you are a GU or CTH student and at which "linje".**
- Please start answering each assignment on a new page; number the pages and use only one side of each sheet of paper.
- Please write in a tidy manner and explain (Clearly) your answers.
- Follow the instruction posted in Canvas: https://chalmers.instructure.com/courses/8965/pages/important-information-regarding-home-exam

LYCKA TILL !!!!

1. (11 points) Consider a synchronous message-passing system. The topology of the system is a ring topology. All processes are anonymous, they have no ids and they run the same code that is independent/agnostic of the size of the ring, i.e. your systems is symmetric.

    1. Give an algorithm for determining if the size of the ring is odd or even, or show that no such algorithm is possible. If you give such an algorithm compute the time complexity of the algorithm.
    2. Suppose now that some of the processes, but not all, are given inputs, at the initialization marking them as "specials". The number of marked processes is not given to any process.

       Give an algorithm for determining if the size of the ring is odd or even, or show that no such algorithm is possible. If you give such an algorithm compute the time complexity of the algorithm. Compare this result to the previous result presented above.
    3. Suppose now that the processes that are marked as specials, are also given inputs at initialization informing them the exact number of marked "special" processes in the system.

       Give an algorithm for determining if the size of the ring is odd or even, or show that no such algorithm is possible. If you give such an algorithm compute the time complexity of the algorithm. Compare this result to the previous results presented above.

    Clarification: Assume a bidirectional, oriented ring and a deterministic algorithm.

2. (11 points) Data replication is a very important area in distributed systems. There are two common schemes for data replication: *active* and *passive*.

    1. Describe both the active and the passive replication scheme, including the differences between them.
    2. Describe in detail a primary-back up, linearizable replication scheme with one Primary and One Replica.
    3. Prove that your algorithm is linearizable.

3. (11 points) Conflict-free Replicated Data Types, eventual consistency and strong eventual consistency:

    1. Define eventual consistency and strong eventual consistency in the context of Replicated Abstract Data Types.
    2. What was the motivation that lead to the introduction of Conflict-free Replicated Data Types?
    3. What is the consistency that they guarantee.
    4. Describe, by providing pseudocode and an informal description, an example of a CRDT implementation. Provide also a proof of its correctness.

4. (11 points) Consider the dinning philosophers problem, n philosophers $P_1, \ldots, P_n$ in a bidirectional ring. Moa wants to help them eat and not starve. She assigns n priorities to them in the following way: P_1 has the lowest priority, P_2, has the second lowest, and so on up to P_n who has the highest priority. Moa instructs each philosopher to seek forks in parallel and to use their priorities to resolve conflicts, i.e. the philosopher with the highest priority gets the respective fork when hungry and competing with a neighbor. Moa is not sure whether her protocol is correct and cannot find a way to calculate its complexity.

1. Can you help Moa to prove or disprove the correctness of her protocol by providing a proof or a counterexample?
2. If you prove that the protocol is correct please provide a complexity analysis of the protocol.
3. If you prove that the protocol is not correct, can you extend the protocol, by wrapping it around with another protocol leyer, without modifying Moa's part (you do not want to hurt Moa's feelings :)), to make it correct? What is the complexity of the algorithm?
4. Can you design another algorithm from scratch that solves the dinning philosophers problem that has better time complexity?

5. (5 points) Moa thinks that she has come up with a breakthrough. She claims that she has managed to design an algorithm A that solves the Byzantine Agreement problem for a system of 6 processes where up to 2 processes are Byzantine. She knows that the problem is impossible for a system comprised by 3 processes where 1 of them can be byzantine. Can you prove to her that her claim contradicts the impossibility result that she is aware of?

6. (11 points) Moa is building a database that is replicated on N machines. To access the database, a client accesses any of the replicas. The communication between clients and the replicas and between the replicas themselves is reliable, point-to-point, and FIFO-ordered. However, the communication delays can vary significantly. Moa has designed the following variant of a totally-ordered multicast algorithm to be used on her system:

   > Each replica maintains Lamport's logical clock, and every inter-replica message is stamped with the unique id of the sender upon transmission. Whenever a replica receives a database update message from a client, it brodcasts the update in a message to all replicas (including itself). Whenever a replica receives an update message from another replica (or from itself) it puts the message into a local queue, and acknowledges the reception of the update by sending an acknowledgment message to all other replicas (including itself). The replica applies an update from its local queue to its local database if and only if:
   >
   > i) the update message has the lowest timestamp among the messages in the replica's local queue, and
   >
   > ii) the update message has been acknowledged by a quorum of N
   >
   > After a replica has applied the update to the local database, the update message and its acknowledgments are removed from the local queue. If later another acknowledgment arrives for the message, such a late acknowledgment is simply dropped from the queue.

   Prove that the above algorithm implements totally-ordered causal multicast (i.e. satisfies the following requirements: 1. Reliability: Integrity, Validity, Agreement 2. Ordering: Causal, Total-order) or produce a counter example.