Chalmers Techniska Högskola
Data- och Informationsteknik

CTH TDA297
GU DIT290

Distribuerade system fk

Lösningar - Tentamen 2022-03-14

Dag, *Tid, Sal*: March 14th 2022, 08:30-12:30, SB

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.

LYCKA TILL !!!!

1. (10 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.

   Solution: *Integrity, Validity should be easy to prove since the communication between processes is based on a reliable, point-to-point, and FIFO-ordered abstraction, processes have unique ids and broadcast the updates that they receive to all.*

   The algorithm uses Lamport´s logical clocks. Lamport´s logical clocks are not consecutive. When a process i has received an update via a broadcast with logical clock value $(3, 5)$, even when this update is the update with the smallest timestamp in the local queue of i, it does not know if there exists an update with clock value $(3, 4)$ that needs to be delivered before. The algorithm uses the acknowledgments phase that uses a FIFO point-to-point communication to all to solve this problem. If i has received ack. from process 3 for $(3, 5)$ it can conclude that there is no update with timestamp $(3, 4)$, if there was such an update it would have received an ack. based on the FIFO point-to-point communication before from 3. This nice property of the algorithm can be extended to cover all causally related events to $(3, 5)$ e.g. $(2, 4)$ will be acknowledged and "pushed" this way by processes 3, via the FIFO point-to-point communication, before it broadcasts the causally related $(3, 5)$. The above is a proof scetch that proves causality, and agreement.

   In order to prove total ordering, one way is to prove it via contradiction: Let us assume that process i delivers (x, y) before (k, z) and $(x, y) < (k, z)$ and j delivers them the other way around. In this case j would have received an ack. from all the processes including process x for (k, z) before receiving an ack for (x, y) from x which is a contradiction because $(x, y) < (k, z)$ and because a the FIFO, point-to-point communication is used in the acks phase.

2. (10 points) Explain:

(a) State the CAP theorem.

Solution: *Definitions in https://hal.inria.fr/inria-00609399v1/document*

(b) Define Eventual Consistency.

Solution: *Definitions in https://hal.inria.fr/inria-00609399v1/document and in slides (8th Lecture).*

(c) Define Strong Eventual Consistency.

Solution: *Definitions in https://hal.inria.fr/inria-00609399v1/document and in slides (8th Lecture).*

(d) Compare Eventual Consistency and Strong Eventual Consistency. What are the pros and cons of each one of them and how do they relate?

Solution: *Motivations of CRDTs in https://hal.inria.fr/inria-00609399v1/document*

(e) Is a linearizable replication solution eventual consistent? Please explain.

Solution: *Yes. All operations are executed in all replicas. Take any execution of a lineraizable replication, map it to a respective sequential legal execution that respects the real time order of the operations. The state of all replicas at the end of the execution is the same with the final state of the sequential execution.*

(f) Is a strong eventual consistent solution sequential consistent? Please explain.

Solution: *SEC is incomparable to sequential consistency. Proof in https://hal.inria.fr/inria-00609399v1/document*

(g) Is bitcoin strong eventual consistent?

Solution: *No it is Eventual Consistent. Discussed in Lecture 14 (Slides) and https://chalmers.instructure.com/courses/17257/files/2055317?wrap=1*

3. (10 points) Consider an asynchronous ring with deterministic anonymous processes that run the same code and do not know the size of the ring. Moa got as an assignment from her developers team to design an algorithm, to be given to the processes, that computes a leader (leader election). Each process may select a leader only once, all the processes have to select the same leader that needs to be a process of the system and must do so after a finite number of system steps.

   Is it possible to find such an algorithm for this system model?

- If yes provide an algorithm that Moa can give back to her team, together with a proof of correctness and time complexity analysis.
- If not provide Moa with an impossibility proof.

Solution: *It is impossible. Discussed in Lectures 10 and 11 (proof is on the respective slides).*

4. (10 points) Suppose that you have a bidirectional but not necessarily complete asynchronous message-passing network represented by a graph $G = (V, E)$ where each node in $v_i \in V$ represents a process with id v_i and each edge in E connects two processes that can send messages to each other.

   Give a protocol that computes a spanning tree that spans all processes and has depth $O(D)$, where D is the diameter of G.

   Prove that your algorithm contracts i) a spanning tree that ii) has depth D. Calculate the time and message complexity of your algorithm.

   Solution: *There are many algorithms that can be proposed here.*

   The simplest is to start asynchronously broadcasts with acknowledgements. Each broadcaster collect the network information. The broadcaster with the smallest id computes sequentially the spanning tree using a sequential breath first search (BFS) algorithm (prove that this gives $O(D)$ depth). Broadcasts the spanning tree to all.

   Another solution is to design a distributed BFS algorithm: you start and do a broadcast with acks for the nodes that are 1 far away from the initiator, then 2 far away, ...

5. (10 points) Moa is building a database that is replicated on N machines with unique ids. To access the database, a client accesses any of the replicas that plays the role of a front-end (FA). The communication between clients and the replicas is reliable, FIFO- ordered point-to-point. The communication between the replicas themselves is FIFO-ordered, reliable multicast (FIFO-multicast).

   The replication mechanism works like this:

   - The client sends a request to one of the replicas that acts as a FA
   - The FA FIFO-multicasts the request to all of the RMs
   - The RMs execute the respective request immediately after they FIFO-deliver it
   - The FA that received the request from the client replies back after it executes the respective request

   1. Does this replication scheme guarantee linearizbility? Provide a proof or a counterexample.

      Solution: *No it is not. Replicate a FIFO queue. Consider two concurrent clients enqueuing two different elements e_1 and e_2 at the same time using two different front-ends. Two replicas can enque these elements in different order.*

   2. Does this replication scheme guarantee sequential consistency? Provide a proof or a counterexample.

Solution: *No it is not. Replicate a FIFO queue. Consider a client enqueuing two different elements e_1 and e_2 one after the other using two different front-ends. Two replicas can enque these elements in different order. FIFO is preserved per front-end.*

3. Define Lineraizability and Sequential Consistency.

   Solution: *The definitions are on pages 792 of the course book and discussed also during lecture 6 and 7 (see slides).*

6. (10 points) The algorithm by Choy and Singh for resource allocation that we discussed in the class, starts with the idea of using the Δ-coloring conflict solution for keeping low the maximum access time. As described in Chandy and Mistra's solution a Δ-coloring will result in an acyclic precedence graph. In contrast to the Chandy and Mistra algorithm, in the Choy and Singh algorithm the precedences will remain static and hence keep the longest chain of processes waiting for resources to be proportional to Δ. The unfairness problem of the static precedence solution is avoided by a mechanism called double doorway.

   - Provide and describe the double doorway mechanism.

     Solution: *It is descibed in the paper: https://dl.acm.org/doi/abs/10.1145/203095.203101, in the thesis: https://chalmers.instructure.com/courses/17257/files/1971824?wrap=1 and in the slides of the 12th and 13th lecture.*

   - Now consider the algorithm that is using only the asynchronous doorway together with the Δ-coloring conflict resolution scheme to solve the general resource allocation problem, i.e. the synchronous doorway is not used.

     1. Does this algorithm solve the general resource allocation problem? Provide a proof of the properties or a counterexample.

        Solution: *Yes it is. It is descibed in the paper: https://dl.acm.org/doi/abs/10.1145/203095.203101, in the thesis: https://chalmers.instructure.com/courses/17257/files/1971824?wrap=1 and in the slides of the 12th and 13th lecture.*

     2. If the algorithm is correct, what is the time complexity of this algorithm in an asynchronous system?

Solution: $O(\Delta^{\Delta})$ *it is descibed in the paper: https://dl.acm.org/doi/abs/10.1145/203095.203101, in the thesis: https://chalmers.instructure.com/courses/17257/files/1971824?wrap=1 and in the slides of the 12th and 13th lecture.*