Quiz 8: Distributed Transactions

Due Jul 13 at 11:59pm **Points** 100 **Questions** 4
Available Jul 7 at 8am - Jul 13 at 11:59pm 7 days **Time Limit** 60 Minutes

This quiz was locked Jul 13 at 11:59pm.

Attempt History

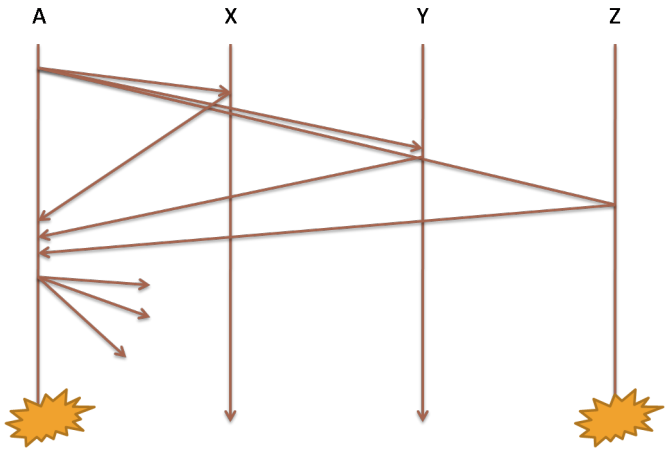
	Attempt	Time	Score
LATEST	Attempt 1	30 minutes	43 out of 100

Score for this quiz: **43** out of 100
Submitted Jul 13 at 1:29pm
This attempt took 30 minutes.

Question 1 — 25 / 30 pts

Consider the process timeline below, where A is the administrator for 2PC. Both A and Z have crashed after the end of the first phase and before X and Y can be notified of the administrator's decision.

1. Why can't X and Y decide to commit?
2. Why can't X and Y decide to abort?



Your Answer:

1. X and Y cannot decide to commit because they have not been told by the coordinator, A, to commit.

2. X and Y cannot decide to abort because they expect the coordinator, A, who failed, to tell them to abort.

1. It is possible that Z voted to abort before it crashed, and the administrator made the decision to abort the transaction. If X and Y commit, they will violate the validity property for atomic commitment: one participant voted to abort, but some of the participants committed. 2. It is possible that Z voted to commit before it crashed, and the administrator told Z to go ahead and commit, but crashed before it could notify X and Y. Again this violates validity: all participants voted to commit, but one participant is committed while two are aborted.

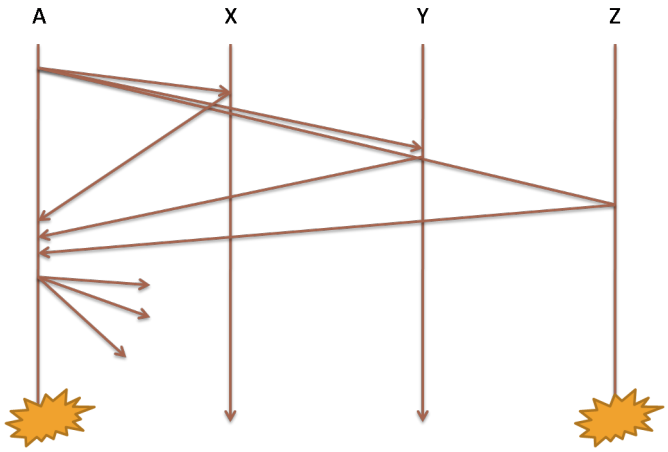
Question 2

5 / 30 pts

This question assumes processes running 3PC. Assume the coordinator has crashed. What do the surviving processes know, and why:

1. If none of the surviving processes has received a Prepare-to-Commit message?
2. If one or more of the surviving processes has received a Prepare-to-Commit message?
3. If one or more of the surviving processes has received a Commit message?

If a process receives a Prepare-to-Commit message, is there any scenario under which it will abort its transaction? Explain.



Your Answer:

1. The surviving processes only know about the OK to commit inquiry

2. The surviving processes inquire if other processes received a prepare-to-commit message before the coordinator crashed.

2. The surviving processes inquire if other processes received a prepare-to-commit message before the coordinator crashed and are all ready to commit. They abort if it is safe.

1. If no surviving process has received a Prepare-to-Commit, then Phase 2 has not completed, and no process has yet entered Phase 3 (where some processes may have committed). 2. If a surviving process has received a Prepare-to-Commit, then all processes in Phase 1 voted to commit, since the administrator is now in Phase 2 telling processes to prepare to commit. 3. If a surviving process has received a Commit message, then all processes have acknowledged the Prepare-to-Commit message in Phase 2, and therefore all processes know that the decision of the protocol is to commit. If the administrator crashes before completing Phase 2, and none of the surviving participants have received Prepare-to-Commit, then (as explained in (1) above), no process has committed yet, and the surviving participants may decide to abort. The crashed participant will learn from the survivors, as part of its recovery protocol, that the decision was made to abort.

Question 3

5 / 20 pts

We can use the Paxos approach to provide a non-blocking atomic commitment protocol, but the obvious approach (using a consensus box to reach consensus among rival 2PC coordinators) introduces two extra message delays. One delay is due to an acceptor having to inform the Transaction Manager (TM) of the decision, but this can be eliminated by simply having the TM be one of the acceptors.

1. What is the other source of this message delay?
2. How does Paxos Commit avoid this other message delay?

Your Answer:

1. The other source of the message delay is within the Resource Manager.

2. Paxos Commit avoids this message delay by collecting proposed values and remembering it forever.

1. What is the source of this message delay? There is an extra round where the TM(s) propose Prepared to the Consensus Box.
2. How does Paxos Commit avoid this message delay? Paxos Commit has the RMs send their proposals directly to their own Consensus Box, which then forwards its choice to the TM.

Question 4

8 / 20 pts

For the nested transaction model:

1. When does failure of one transaction force the failure of other transactions?
2. Why are a child's transactions "anti-inherited" by its parent when it commits?
3. How do multi-level transactions handle this issue?

Your Answer:

1. The failure of one transaction forces the failure of other transactions when the parent transaction aborts causing all of its descendants to abort.

2. Child transactions are tentative to their parents. The Child's locks are released to the parent.

3. Since multi-level transactions are transparent and visible to parents and descendants, child transactions are able to commit even when their parents aborted.

2. Why are a child's transactions "anti-inherited" by its parent when it commits? Since it may be aborted by an ancestor, the updates of a “committed” transaction should not be visible except to its parent.
3. How do multi-level transactions handle this issue? Multi-level transactions allow updates from committed transactions to be made visible immediately, but require compensating transactions if they are subsequently aborted by ancestor transactions.

Quiz Score: **43** out of 100