

ACS Theory Assignment 3

This assignment is due via Absalon on Jan 4, 23:59. While individual hand-ins will be accepted, we strongly recommend that this assignment be solved in groups of maximum two students. NOTE: The KU IDs of ALL group members MUST be stated on a separate `group.txt` file to ensure all group members get feedback and get the assignment accounted for in Absalon.

A well-formed solution to this assignment should include a PDF file with answers to all theory questions. Note that all homework assignments have to be submitted via Absalon in electronic format. It is your responsibility to make sure in time that the upload of your files succeeds. While it is allowed to submit scanned PDF files of your homework assignments, we strongly suggest composing the assignment using a text editor or LaTeX and creating a PDF file for submission. Email or paper submissions will not be accepted.

Learning Goals

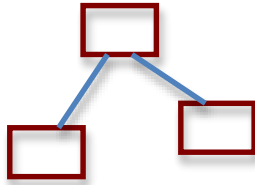
This assignment targets the following learning goals:

- Perform simple system reliability calculations, while clearly stating underlying assumptions.
- Reason about operations in object replications with vector clocks.
- Explain the functioning of commit protocols for distributed transactions.
- Reason about local and global deadlocks.

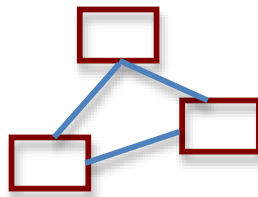
Question 1: Reliability

The town council wants to implement a municipal network to connect the local area networks in the library, the town hall, and the school. They want to minimize the chance that any building is completely disconnected from the others. They are considering two network topologies:

1. Daisy Chain:



2. Fully connected:



Each link in the network has a failure probability of p . Now, answer the following questions:

1. What is the probability that the daisy chain network is connecting all the buildings?
2. What is the probability that the fully connected network is connecting all the buildings?
3. The town council has a limited budget, with which it can buy either a daisy chain network with two high reliability links ($p = .000001$), or a fully connected network with three low-reliability links ($p = .0001$). Which should they purchase?

Remember to explicitly state the assumptions you make for your reliability calculations.

Question 2: Vector Clock

Three processes, P1, P2, and P3, replicate a common object. The object is updated asynchronously by three different clients, and each client happens to dispatch its update to a different process out of P1, P2, and P3. After receiving each update, the respective processes incorporate the update in its local replica and then propagates the new value to the other processes. The precise exchange of messages is shown in the following figure:

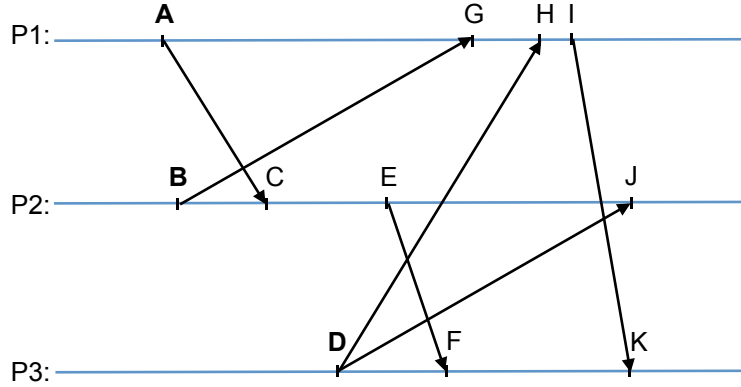


Figure 1: Exchange of messages among processes

Note that P1 got a client update at A, P2 at B, and P3 at D; the clients are omitted from the figure to avoid cluttering. The system uses vector clocks to keep track of the object version internally. You may assume that all local clock values start at zero at the beginning of the interaction shown. In addition, assume that only updates from clients cause an increment of a local clock, i.e., not every event causes an increment of the local clock except for original client updates. Therefore, $(1,1,1)$ is the maximum vector clock possible in the given interaction. The latter policy ensures that, if update propagation reaches all processes, then we expect the objects version to converge as either overwriting or merging through a commutative associative function takes place.

Every time an update is received by a process P , the process chooses one out of the following actions to incorporate the update:

- (A1) Keep P 's own value;
- (A2) Overwrite P 's value with the latest value from the message;
- (A3) Call an application-specific merge function to combine both values consistently.

For each point in the execution from A to K shown in Figure 1, give the value of the vector clock at the corresponding process before the update event is processed, the value of the vector clock received in the incoming message or sent in the outgoing message, the action taken by the process to incorporate the update, and the value of the vector clock after the update is processed. Use a table with the following format to report your answer:

Event	Vector Clock at Process Before	Message Vector Clock	Action Taken by Process	Vector Clock at Process After
A
B
...

In addition to the table, provide a paragraph with a brief justification for your reasoning for constructing the table, i.e., discuss when each action should be taken depending on the vector clock values at the given message and event.

Question 3: Distributed Coordination

A three-phase commit protocol has the following parts:

- Phase 1: is the same as for two-phase commit.
- Phase 2: the coordinator collects the votes and makes a decision; if it is No, it aborts and informs participants that voted Yes; if the decision is Yes, it sends a *preCommit* request to all the participants. Participants that voted Yes wait for a *preCommit* or *doAbort* request. They acknowledge *preCommit* requests and carry out *doAbort* requests.
- Phase 3: the coordinator collects the acknowledgments. When all are received, it *Commits* and sends *doCommit* to the participants. Participants wait for a *doCommit* request. When it arrives they *Commit*.

Explain how this protocol avoids delay to participants during their 'uncertain' period due to the failure of the coordinator or other participants. Assume that communication does not fail.

Question 4: Distributed Transactions

The schedule for three distributed transactions (T1, T2, T3) spanning three nodes has been outlined below. Node 1 stores data elements X and Y, node 2 stores data elements A and B while node 3 stores data elements C and D. Strict two phase locking is used as the concurrency control mechanism.

N O D E 1	T1	R(X)			
	T2		R(X)	W(X)	
	T3		R(Y)	W(Y)	
2	T1		R(A)	W(A)	
	T2		R(B)		
	T3			R(B)	W(B)
3	T1			R(C)	W(C)
	T2	R(D)		W(D)	
	T3		R(C)		

1. Construct the local waits-for graphs on each node. Is there a local deadlock on any of the nodes? Exhibit the graphs and explain why there are deadlocks or why there are no deadlocks.
2. Construct a global waits-for graph. Is there a global deadlock? Exhibit the graph and explain why there is a global deadlock or why there is no global deadlock.
 **Recall that the waits-for graph is a directed graph in which nodes represent transactions, and edges represent that an object is held by a transaction, and another transaction is waiting for the object.
3. Is there any scenario starting from the situation above in which T1 is allowed to commit? Explain. If it is possible for T1 to commit, show the steps and messages needed by T1 to commit, assuming a two-phase commit protocol. Make sure to point out when information for a step should be recorded durably in the log of any of the nodes involved in the commit protocol.