ACS Theory Assignment 3

This assignment is due via Absalon on Jan 7, 23:59. While individual hand-ins will be accepted, we strongly recommend that this assignment be solved in groups of maximum two students. A well-formed solution to this assignment should include a PDF file with answers to all theory questions.

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

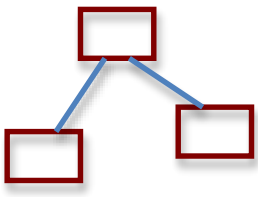
This assignment targets the following learning goals:

- 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.

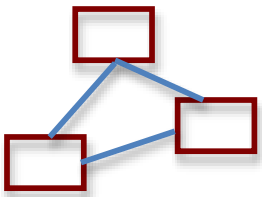
Question 1: Reliability

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 incorporates the update in its local replica and then propagates the new value to the other processes. In other words, the system adopts the peer-to-peer asynchronous replication paradigm.The precise exchange of messages is shown in the following figure:

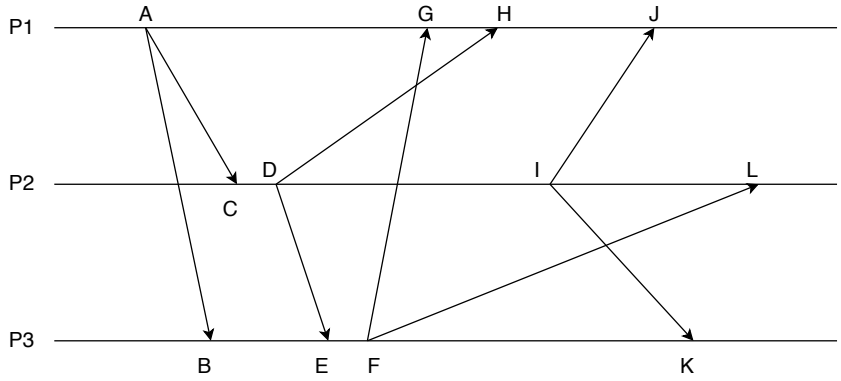


Figure 1: Exchange of messages among processes

Note that P1 got a client update at A, P2 at D, P3 at F, and P2 at I; 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., message events do not increment the clock. Therefore, (1,2,1) 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 object's version to converge as either overwriting or merging through a commutative associative function takes place. That is to say, the system ensures eventual consistency.

Note that the policy of incrementing vector clocks deviates from what was presented in the lecture, where message events also increment vector clocks. This is because, in this

application, we only need to use vector clocks to keep track of the version of the object, and message events do not affect the version of the object.

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 value from the message or the client update request;

(A3) Call an application-specific merge function to combine both values consistently.

For each point in the execution from A to L 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 the Event	Vector Clock in the Message (either received or sent)	Action Taken by Process	Vector Clock at Process After the Event
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 in order to achieve eventual consistency. Hint: action A3 should be invoked only if you cannot choose A1 or A2 based on the vector clocks.

Question 3: Distributed Transactions

The schedule for three distributed transactions (T1, T2, T3) spanning three nodes and one non-distributed transaction (T4) running only on node 2 has been outlined below. Node 1 stores data elements A and B, node 2 stores data elements C and D while node 3 stores data elements E, F and G. Strict two phase locking is used as the concurrency control mechanism.

Node 1	T1	R(A)				W(A)				
	T2		R(B)					W(A)		
	T3				R(B)					W(B)
Node 2	T1		R(C)			R(D)				
	T2	R(D)						W(C)		
	T4				W(D)					
Node 3	T1					R(G)	W(E)			
	T2		R(F)						W(G)	
	T3	R(E)			R(F)					

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. Suppose the distributed transactions use the two-phase commit protocol. Explain a scenario starting from the situation above in which T3 is allowed to commit. List the steps and messages needed by T3 to commit. 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.