Introduction to Distributed Systems
WT 19/20

Assignment 5 – Part I

Submission Deadline: Monday, 13.01.2020, 08:00

- *Submit the solution in PDF via Ilias (only one solution per group).*
- *Respect the submission guidelines (see Ilias).*

1 Two-Phase Locking [10 points]

Figure 1 shows a serializeable history generated by basic (non-strict) Two-Phase Locking for read and write operations originating from four transactions (T_1, T_2, T_3, T_4). In the figure, transaction T_1 is aborted while other transactions T_2, T_3, and T_4 are committed.

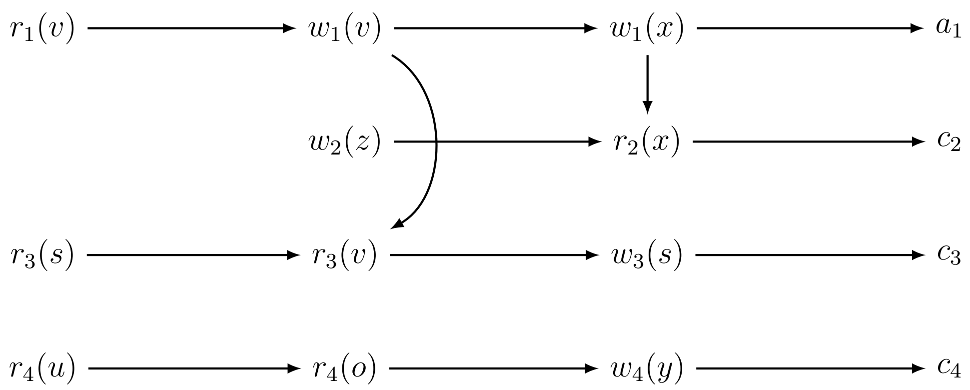

Figure 1: History H_1

a) [5 points] Complete the following space-time diagram (Figure 2) with all read and write operations shown in Figure 1 while ensuring the order of operations from/between different transactions. Also, add to the space-time diagram all required lock and unlock operations. Ensure that all locks are released as early as possible and the operations of each transaction start at the earliest possible time in the space-time diagram, i.e., an operation must not be delayed unless it is blocked due to being unable to acquire locks.

b) [2 points] In the history $H1$ shown in Figure 1, some transactions might suffer from cascading abort since the transaction T_1 aborts and as a result of using basic Two-Phase Locking. Name the transactions that are aborted as a result of aborting T_1. Justify your answer.

c) [3 points] Now, Strict Two-Phase Locking is used for synchronization. Draw the corresponding space-time diagram (Figure 3). Ensure that the operations of each transaction start at the earliest possible time in the space-time diagram, i.e. an operation must not be delayed unless it is blocked due to being unable to acquire locks. Explain why Strict 2PL prevents cascading aborts?

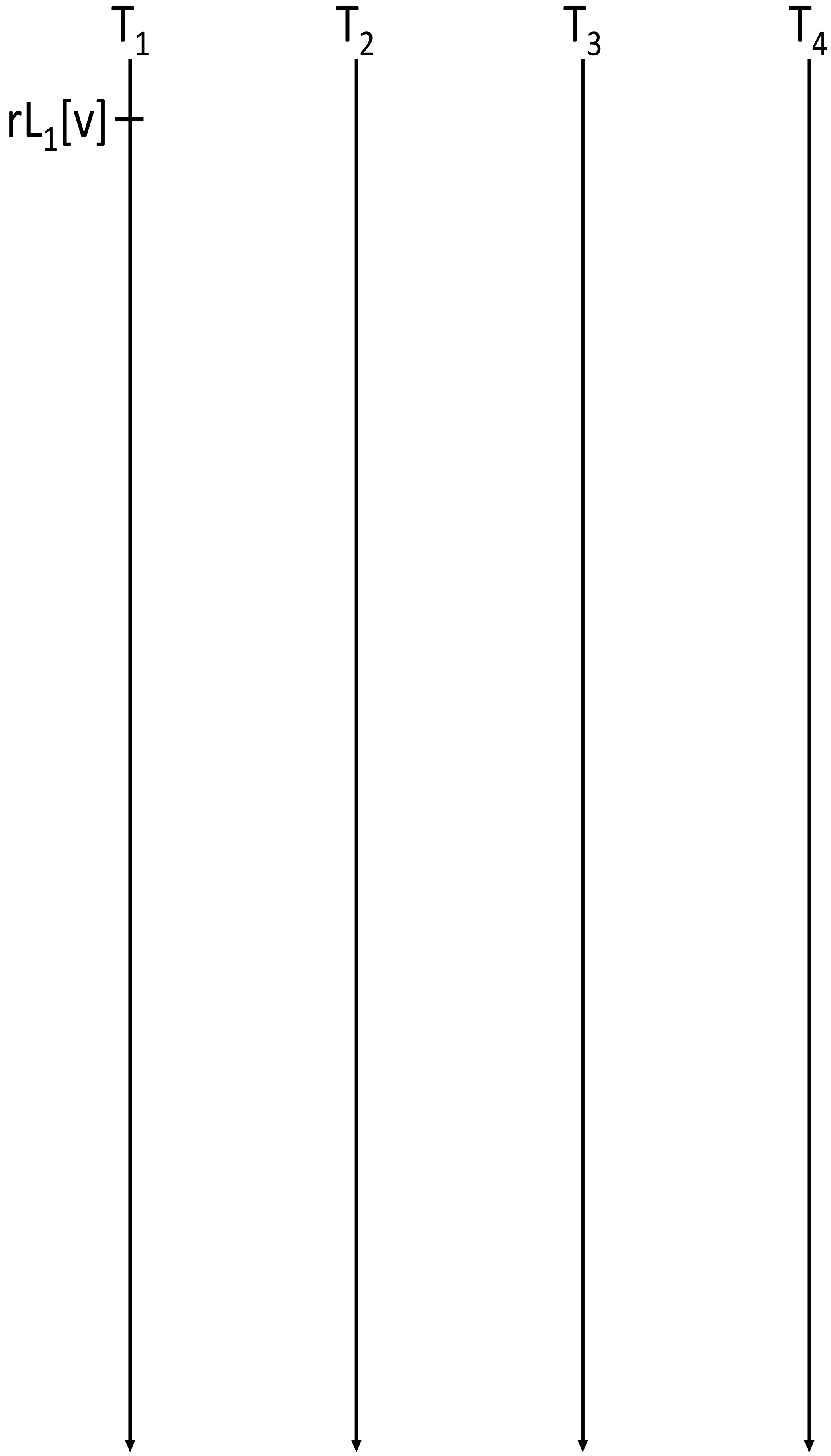

Figure 2: Basic Two-Phase Locking: Execution of Transactions T_1 to T_4

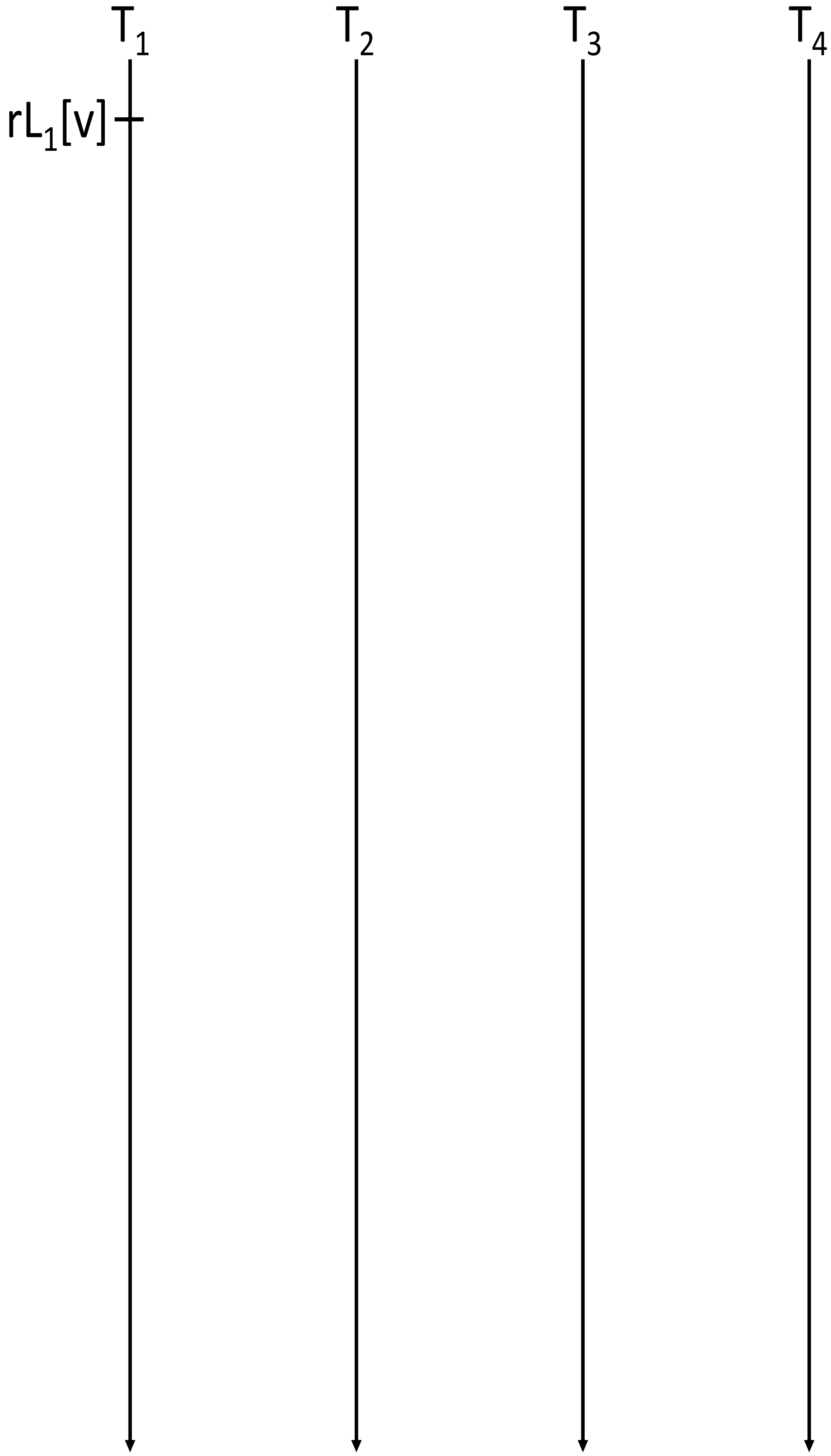

Figure 3: Strict Two-Phase Locking: Execution of Transactions T_1 to T_4

2 Two-Phase Commit [9 points]

Figure 4 shows five database servers (nodes) N_1, N_2, N_3, N_4 and N_5. Nodes N_1 and N_2 are both located in North America and connected by a wired link. Similarly, nodes N_4 and N_5 reside in Asia and are also connected to each other through a wired link. Node N_3 is located in Europe and communicates with other nodes (N_1, N_2, N_4 and N_5) via two separate satellites.

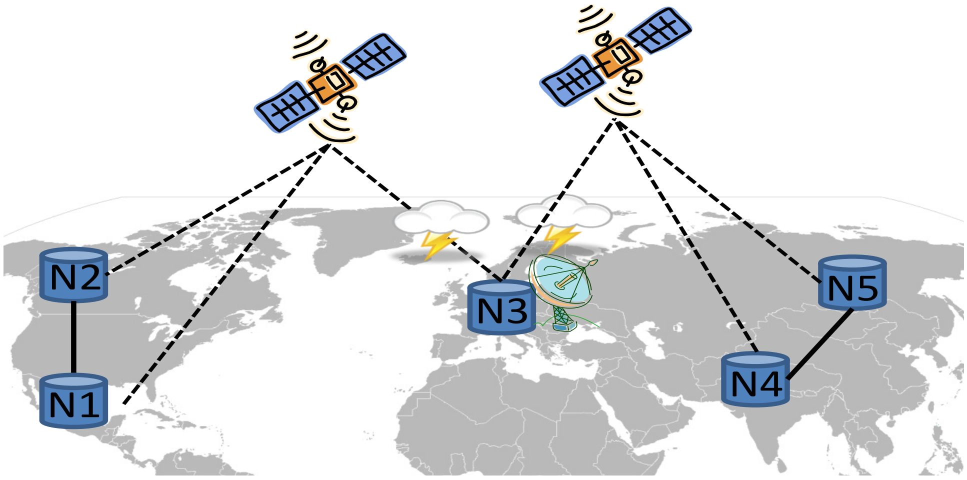

Figure 4: Connectivity between database servers

The five nodes perform a distributed transaction initiated by N_3, i.e., N_3 is the coordinator. To commit the transaction, N_3 starts the 2-Phase Commit Protocol (2PC), as shown in Figure 5. (Note that the space–time diagram in Figure 5 shows only the message flow and not the logging to stable storage.)

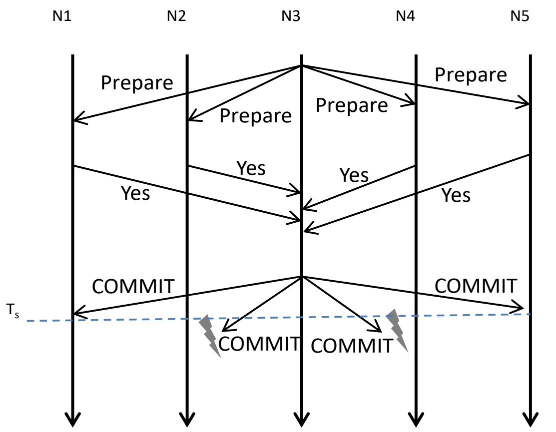

Figure 5: Space–time diagram for the 2-Phase Commit Protocol.

At time T_s, a heavy thunderstorm above N3 interrupts the bidirectional link to both satellites where N3 is not reachable for several minutes until T_e. Therefore, the `COMMIT` messages from N3 to N2 and N4 are lost.

a) [3 points] What happens to N_1, N_2, N_3, N_4 and N_5 if the *simple termination protocol* is used? Explain the protocol executed by each node until termination of the transaction. Specifically, explain the execution both before and after the time t_e. Assume a system, where node failures never occur.

b) [4 points] What happens to nodes N_1, N_2, N_3, N_4 and N_5 if the *cooperative termination protocol* is used? Sketch the execution of the protocol until termination of the transaction by completing the space–time diagram from Figure 5, showing all message exchanges. Specifically, explain the execution both before and after the time t_e. Also, indicate for nodes N_1, N_2, N_4 and N_5 whether they remain blocked or not until the connectivity to node N_3 is re-established (i.e., time t_e is reached). Again, assume that no node failures occur.

Note: we consider a node being "blocked" as long as it is unable to unilaterally abort a running transaction and it is waiting for a message to proceed, i.e. it has sent a `YES` vote but is still waiting for a termination decision (`COMMIT` or `ABORT`).

c) [2 points] Now assume that node failures may occur, and that N_2 suffers a crash failure immediately after sending its `YES` message to N_3 and recovers after t_e. Modify your diagram from part (b) accordingly, under the assumption of node failures.

3 Data Replication [11 points]

Given are four nodes *A, B, C, D* as shown in Figure 6. An object X is replicated to these nodes using the “weighted voting” replication protocol. Each node n is available with a certain probability p_n, also shown in the figure, and maintains a replica X_n of the logical object X. Assume that no communication failures occur (i.e., network partitioning is not possible).

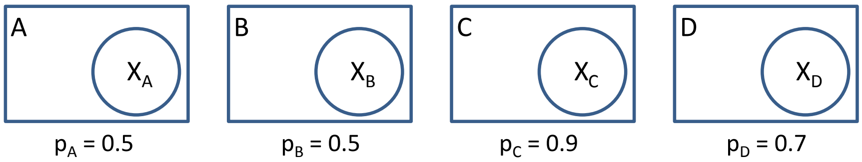

Figure 6: Configuration for Weighted Voting

a) [6 points] Assume that read and write thresholds of $q_r[X] = 3$ and $q_w[X] = 4$ are provided. Find a valid assignment of values for $q[X_A]$, $q[X_B]$, $q[X_C]$, and $q[X_D]$ that

1. minimizes the number of nodes that have to be locked for writing X
2. maximizes the availability of X for writing

Also, give all possible read and write quorums for your solution.
Note: each node must have at least one vote, i.e. $q[X_n] \geq 1$. *Consider all quorums, not only minimal ones.*

b) [3 points] Now, assume that $q[X_A] = 1, q[X_B] = 1, q[X_C] = 2, q[X_D] = 3$. Find a write threshold $q_w[X]$ that enables exactly three different write quorums. Give the

threshold value $q_w[X]$ and the resulting write quorums.
Note: Consider all quorums, not only minimal quorums.

c) [2 points] *Majority Consensus* is a special case of Weighted Voting replication management protocol. Now, assume that the *Majority Consensus* is used to replicate an object Y in a system with three nodes *K, L, M* where each node has exactly *1* vote. As before, each node n is available with a certain probability p_n. Let $p_r(Y)$ be the probability that the logical object Y is available for *reading.* What is $p_r(Y)$ in terms of p_K, p_L, and p_M? Justify your answer.