

# PROJECT 3 REPORT

**Yashwanth Rao Varala Balaji** – 1001916698

**Rakshith Singh Thakur** – 1002037934

I have neither given nor received unauthorized assistance on this work. I will not post the project description and the solution online.

Sign: Yashwanth Rao Varala Balaji, Rakshith Singh Thakur Date: 04/23/2023

**Prompt**:

The program entails the implementation of a two-phased distributed commit protocol, consisting of two distinct phases. The program involves two types of clients: coordinators, which function as leaders, and participants.

**Phase 1:**

Step 1 - The coordinator sends a message to all participants requesting their commitment to the transaction.

Step 2 - Participants respond with either a Yes or No vote.

**Phase 2:**

Based on the responses received, the coordinator either commits or cancels the transaction.

Failures are an inherent consideration in the process, and multiple failure cases are simulated during program implementation.

Case 1:

All participants vote in favor of committing the transaction, and the coordinator proceeds with committing the transaction, which represents the optimal scenario.

Case 2 & Sub-case (as per implementation):

One of the participants fails, either by termination or delayed response, before acknowledging the commit message from the coordinator, or after acknowledging the commit message. In both cases, once the participant recovers, it should resynchronize with the other participants and coordinators.

Case 3: If the coordinator sends the "can commit" message but fails afterward, or if it receives acknowledgements from participants and then fails, it can recover by obtaining the last log of the participant and synchronizing with other participants in the network.

Case 4: If the coordinator fails after sending the "commit" transaction message, it can recover by obtaining the last log from the participants and updating its values.

The aim of the program in all the failure cases mentioned above is to maintain consistency of data across all participants and the coordinator.

**Transaction:**

Throughout the program, for failure simulation and transaction simulation, each time the coordinator sends the "can commit" message, participants perform an addition operation. Specifically, each participant calculates a future value by adding 1 to the current value:

future\_value = current\_value + 1

If the transaction is committed, the current value is updated to the future value and committed. If the transaction is aborted, the current value is committed without any modifications.

**Scenarios Conducted:**

• For this programming assignment, a total of 5 simulations were conducted, including 4 main simulations and 1 sub-simulation, covering all the discussed failure cases.

• Python was chosen as the implementation language.

• In all simulations, the process number and initial value were passed consistently across all participants and the coordinator.

**Participant Failure Case:**

• A simulation was conducted where participant -1 was intentionally failed.

• Upon recovery from failure, participant one requested previous logs from other clients in the network and updated its current value based on the received logs.

**Coordinator Failure Case:**

• Like the participant failure case, a simulation was conducted where the coordinator failed, and upon recovery, obtained the last log from the participant to update its value accordingly.

Different Message Types and States Used:

**Message Types:**

Q: Question - Coordinator initially sends a "can commit" message

A: Acknowledge - Participants acknowledge messages from the coordinator

Ab: Abort - Coordinator sends an abort transaction instruction

C: Commit - Coordinator sends a commit transaction instruction

R: Request for previous log - When Coordinator/Participant fails, it requests logs from others

Rb: Reply with backed-up data - When the message type is "R", other participants reply with logs

**State:**

're': Restarted the server - Indicates recovery of either participant or coordinator from failure

Major Challenges: Injecting failures with delays in message passing and termination of processes was challenging due to the simulation nature. However, after multiple trials of testing at different time intervals (using sleep for processes), the goal was achieved.