CSE 586: DISTRIBUTED SYSTEMS
COURSE PROJECT 1

UB ID: abachani

TASK 1:

Scenario A: TMMAYFAIL == FALSE

- Case 1: Completed|Consistency|Deadlock
  No errors
- Case 2: Completed|Consistency|No Deadlock
  No errors
- Case 3: Consistency| Not Committed|No Deadlock
  No errors
- Case 4: Consistency|Not Committed|Deadlock
  No errors

Scenario B: TMMAYFAIL == TRUE

- Case 1: Completed|Consistency|No Deadlock
  Completed Property violated.
  States:
  1. /\ rmState = <<"aborted","aborted","aborted","prepared">> /\ tmState = "abort"
  2. /\ rmState = <<"aborted","aborted","aborted","prepared">> /\ tmState = "unavailable"
  3. /\ rmState = <<"aborted","aborted","aborted","prepared">> /\ tmState = "unavailable"

- Case 2: Not Committed|Consistency|Deadlock
  Deadlock Reached
  States:

1. /\ rmState = <<”prepared”,”prepared”,”prepared”,”prepared”>> /\ tmState = “init”
2. /\ rmState = <<”prepared”,”prepared”,”prepared”,”prepared”>> /\ tmState = “commit”
3. /\ rmState = <<”prepared”,”prepared”,”prepared”,”prepared”>> /\ tmState = “unavailable”

- Case 3: Consistency|Completed|Deadlock
  Deadlock Reached
  States:
  1. /\rmState = <<”prepared”,”prepared”,”prepared”,”prepared”>> /\ tmState = “init”
  2. /\ rmState = <<”prepared”,”prepared”,”prepared”,”prepared”>> /\ tmState = “commit”
  3. /\ rmState = <<”prepared”,”prepared”,”prepared”,”prepared”>> /\ tmState = “unavailable”

TASK 2:
Scenario A: TMMAYFAIL == FALSE, RMMAYFAIL ==TRUE

- Case 1: Completed|Consistency|Deadlock
  No errors
- Case 2: Consistency|Completed|No Deadlock
  No errors
- Case 3: Consistency|Not Committed|No Deadlock
  No errors
- Case 4: Not Committed|Consistency|Deadlock
  No errors

Scenario B: TMMAYFAIL == TRUE, RMMAYFAIL == TRUE

- Case 1: Consistency| Not Committed| Deadlock == Deadlock Reached
  States:
  1. /\ rmState = <<”prepared”,””prepared”,”prepared”,”prepared”>> /\ tmState = “commit”
  2. /\ rmState = <<”prepared”,””prepared”,”prepared”,”prepared”>> /\ tmState = “unavailable”
  3. /\ rmState = <<”prepared”,””prepared”,”prepared”,”unavailable”>> /\ tmState = “unavailable”
  4. /\ rmState = <<”prepared”,””prepared”,”prepared”,”prepared”>> /\ tmState = “unavailable”

- Case 2: Consistency|Not Committed|No Deadlock
  No errors
- Case 3: Consistency|Completed| No Deadlock
  Consistency Property Violated
  States:
  1. /\rmState = <<"committed","committed","unavailable","prepared">> /\ tmState = "unavailable"
  2. /\rmState = <<"committed","committed","prepared","prepared">> /\ tmState = "unavailable"
  3. /\rmState = <<"committed","committed","prepared","unavailable">> /\ tmState = "unavailable"

- Case 4: Consistency|Completed|Deadlock == Deadlock Reached
  States:
  1. /\ pre =<<" prepared","prepared","prepared","prepared">> /\rmState = <<"prepared","prepared","prepared","prepared">> /\ tmState = "unavailable"
  2. /\ pre =<<" prepared","prepared","prepared","prepared">> /\rmState = <<"prepared","prepared","prepared","unavailable">> /\ tmState = "unavailable"
  3. /\ pre =<<" prepared","prepared","prepared","prepared">> /\rmState = <<"prepared","prepared","prepared","prepared">> /\ tmState = "unavailable"

TASK 3:
Scenario A: TMMAYFAIL == TRUE, RMMAYFAIL == FALSE

- Case 1: Consistency|Not Committed|Deadlock
  Deadlock Reached
  States:
  1. /\rmState = <<"aborted","working","aborted","aborted">> /\ tmState = "abort"
  2. /\rmState = <<"aborted","aborted","aborted","aborted">> /\ tmState = "abort"
- Case 2: Consistency|Not Committed|No Deadlock
  No errors
- Case 3: Consistency|Completed|No Deadlock
  No errors

- Case 4: Consistency|Completed|Deadlock
  Deadlock Reached
  States:
  1. /\rmState = <<”working”,”aborted”,”working”,”aborted”>> /\ tmState = “abort”
  2. /\rmState = <<”aborted”,”aborted”,”working”,”aborted”>> /\ tmState = “abort”
  3. /\rmState = <<”aborted”,”aborted”,”working”,”aborted”>> /\ tmState = “abort”

Scenario B: TMMAYFAIL == TRUE, RMMAYFAIL == TRUE

- Case 1: Consistency|Not Committed|No Deadlock
  Consistency Violated
  States:
  1. /\rmState = <<”prepared”,”prepared”,”prepared”,” unavailable”>> /\ tmState = “unavailable”
  2. /\rmState = <<”prepared”,”prepared”,”prepared”,”unavailable”>> /\ tmState = “abort”
  3. /\rmState = <<”committed”,”prepared”,”prepared”,”unavailable”>> /\ tmState = “abort”
  4.
- Case 2: Consistency|Not Committed|Deadlock
  Invariant Consistency Violated
  States:
  1. /\rmState = <<”prepared”,”unavailable”,”prepared”,” prepared”>> /\ tmState = “abort”
  2. /\rmState = <<”committed”,”unavailable”,”prepared”,” preapred”>> /\ tmState = “abort”
  3. /\rmState = <<”committed”,”unavailable”,”prepared”,” preapred”>> /\ tmState = “abort”

- Case 3: Consistency|Completed|No Deadlock
  Consistency Violated
  States:
  1. /\rmState = <<”prepared”,”prepared”,”prepared”,” preapred”>> /\ tmState = “commit”
  2. /\rmState = <<”prepared”,”prepared”,”prepared”,” preapred”>> /\ tmState = “unavailable”

3. /\rmState = <<”prepared”,”unavailable”,”prepared”,” preapred”>> /\ tmState = “unavailable”
4. /\rmState = <<”prepared”,”unavailable”,”prepared”,” preapred”>> /\ tmState = “unavailable”

- Case 4: Consistency|Completed|Deadlock
  Consistency Violated
  States:
  1. /\rmState = <<”prepared”,”prepared”,”working”,” preapared”>> /\ tmState = “init”
  2. /\rmState = <<”prepared”,”prepared”,”prepared”,” preapred”>> /\ tmState = “init”
  3. /\rmState = <<”prepared”,”prepared”,”prepared”,” preapred”>> /\ tmState = “commit”

Explanations for the cases in the scenarios listed above:

Completed Property Violation, while most of the RMs are in abort state, TM becomes unavailable, without Transaction Manager, RMs cannot receive any decision to what the next step if for them, so the system enters the non-termination state as RMs keeps waiting for TM for decision. This leads to Completed Property being violated.

Deadlock Reasoning: Deadlock is a condition where resources are limited which is required by more events at a time, which leads to a deadlock situation. In a scenario, where RMs are waiting for a response from TM, but TM becomes “unavailable” before the transaction is processed then RMs are in a state where they wait indefinitely and system enters a non-terminating state. This creates a deadlock and this is when a deadlock is reached in the system when RMs are waiting indefinitely for TM.

Consistency Violated Reasoning: Let’s take an example where we have 4 resource managers, which are in prepared state initially. Transaction Manager sends “commit” to RMs. RM1 when receives the “commit” from TM, in the next action TM becomes unavailable and so does RM2. Because TM is unavailable Back-up Transaction Manager (BTM) takes over. When BTM analyzes the states of the 4 RMs which are: “prepared”,”unavailable”,”prepared”,”prepared”. So, BTM decides to abort as one of the RM is unavailable. Then RM1 acts on the

“commit” action given to it by Original TM. But RM3 sees “abort” from BTM and acts on it. This is how consistency of the system is violated.
When one of the RMs acts on the action received by TM but BTM has no way of knowing which RM were already queued to be committed, then Consistency is likely to be violated which has happened in all the cases of Scenario B.

Conclusion
2 Phase commit Protocol is a transaction management system which has multiple Resource Managers and one transaction manager where RMs are databases and TM is the coordinator which decides commit or abort. It makes sure that all the RMs are in the same state, either all are in commit state or all are in abort state so that the system maintains consistency in the transaction. It makes sure that the RMs don’t have different invariant and give us desirable outcome and consistency to our system. This protocol follows “co-ordinator”, which keeps a connection with all the events which are in resource manager, so that is is able to make a connection with any of the RMs. There are many possible scenarios where the 2PC protocol fails. Scenarios where RMs can become unavailable or TM becomes unavailable at any time in between a transaction and there is no way of knowing when it will come back. 2PC does provide a Back-up Transaction Manager that handles the transactions once the TM becomes “unavailable”, but the inconsistency caused while the BTM takes over after TM becomes unavailable is something that 2PC cannot sync.
Two Phase uses TM component, which maintains connections with resource managers and communicates with them and plays a role of leader to them or sometimes as a participant depending on where the transaction was initiated.
There are two phases that cover the Two phase commit, in the first phase which is the proposal phase, it proposed a value to every participant in the system and gather responses of the RMs. In the next phase, commit-or-abort phase, the TM sends the “commit” or ”abort” decision to RMs based on the votes that were received by the TM in phase 1. If any of the RM was not ready to commit then it sends abort to all the RMs. Otherwise it sends “commit”. If any of the events wants to be co-ordinator that it can propose a value and initiate a round.
Sometimes in the transactions, the coordinator fails permanently which leads to a case where some participants can never resolve their transactions, suppose an RM has sent a approval signal to TM, it will be blocked till it receives a response from the TM.
In a scenario, where 2PC occurs after a transaction is processed so the latency is increased as the protocol takes a significant amount of time to run, so latency is increased in such scenarios. This can be huge disadvantage in scenarios like

Banking System, where one blockage can lead to blocking other transactions as well. Overall the latency keeps increasing and accuracy decreases.