4.6 KOO AND TOUEG COORDINATED CHECKPOINTING AND RECOVERY TECHNIQUE:

- Koo and Toueg coordinated check pointing and recovery technique takes a consistent set of checkpoints and avoids the domino effect and livelock problems during the recovery.
- Includes 2 parts: the check pointing algorithm and the recovery algorithm

A. The Checkpointing Algorithm

The checkpoint algorithm makes the following assumptions about the distributed system:

- Processes communicate by exchanging messages through communication channels.
- Communication channels are FIFO.
- Assume that end-to-end protocols (the sliding window protocol) exist to handle with message loss due to rollback recovery and communication failure.
- Communication failures do not divide the network.

The checkpoint algorithm takes two kinds of checkpoints on the stable storage: Permanent and Tentative.

A *permanent checkpoint* is a local checkpoint at a process and is a part of a consistent global checkpoint.

A *tentative checkpoint* is a temporary checkpoint that is made a permanent checkpoint on the successful termination of the checkpoint algorithm.

The algorithm consists of two phases.

First Phase

1. An initiating process Pi takes a tentative checkpoint and requests all other processes to take tentative checkpoints. Each process informs Pi whether it succeeded in taking a tentative checkpoint.
2. A process says “no” to a request if it fails to take a tentative checkpoint
3. If Pi learns that all the processes have successfully taken tentative checkpoints, Pi decides that all tentative checkpoints should be made permanent; otherwise, Pi decides that all the tentative checkpoints should be thrown-away.

Second Phase

1. Pi informs all the processes of the decision it reached at the end of the first phase.
2. A process, on receiving the message from Pi will act accordingly.

3. Either all or none of the processes advance the checkpoint by taking permanent checkpoints.
4. The algorithm requires that after a process has taken a tentative checkpoint, it cannot send messages related to the basic computation until it is informed of Pi's decision.

Correctness: for two reasons

i. Either all or none of the processes take permanent checkpoint
ii. No process sends message after taking permanent checkpoint

An Optimization

The above protocol may cause a process to take a checkpoint even when it is not necessary for consistency. Since taking a checkpoint is an expensive operation, we avoid taking checkpoints.

B. The Rollback Recovery Algorithm

The rollback recovery algorithm restores the system state to a consistent state after a failure. The rollback recovery algorithm assumes that a single process invokes the algorithm. It assumes that the checkpoint and the rollback recovery algorithms are not invoked concurrently. The rollback recovery algorithm has two phases.

First Phase

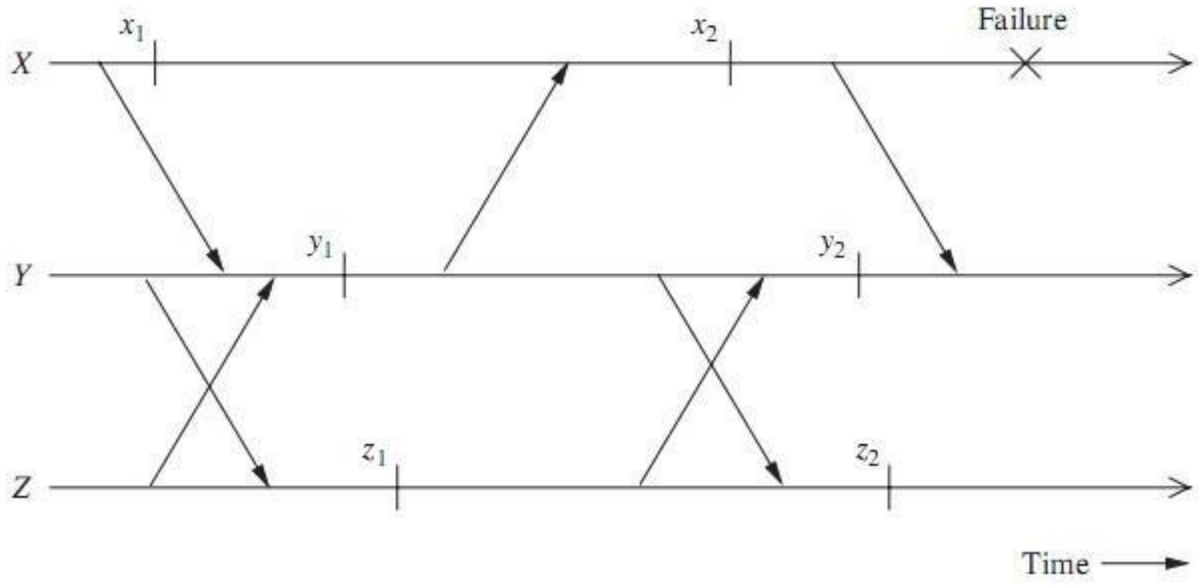
1. An initiating process Pi sends a message to all other processes to check if they all are willing to restart from their previous checkpoints.
2. A process may reply "no" to a restart request due to any reason (e.g., it is already participating in a check pointing or a recovery process initiated by some other process).
3. If Pi learns that all processes are willing to restart from their previous checkpoints, Pi decides that all processes should roll back to their previous checkpoints. Otherwise,
4. Pi aborts the roll back attempt and it may attempt a recovery at a later time.

Second Phase

1. Pi propagates its decision to all the processes.
2. On receiving Pi's decision, a process acts accordingly.
3. During the execution of the recovery algorithm, a process cannot send messages related to the underlying computation while it is waiting for Pi's decision.

Correctness: Resume from a consistent state

Optimization: May not to recover all, since some of the processes did not change anything



The above protocol, in the event of failure of process X, the above protocol will require processes X, Y, and Z to restart from checkpoints x2, y2, and z2, respectively.

Process Z need not roll back because there has been no interaction between process Z and the other two processes since the last checkpoint at Z.