Failure Recovery in Distributed System

Gaurav Parashar

November 9, 2020

Table of Contents I

Table of Contents II

Introduction I

Recovery in computer systems refers to restoring a system to its normal operational state. Recovery may be as simple as restarting a failed computer or failed process. Recovery is a complicated process.

- In general, resources are allocated to executing processes in a computer.
- Ex: a process has memory allocated to it and a process may have locked shared resources, such as files and memory.
- It is imperative that the resources allocated to the failed process be reclaimed so that it can be allocated to other processes.
- If the failed process has modified the database, then it is important that all the modifications made to the database by the failed process are undone.
- if a process has executed for some time before failure, it would be preferable to restart the process from the point of its failure & resume its execution

Basic Concepts I

- A system constitutes of a set of hardware and software components & designed to provide a specific service
- The components of a system may themselves be systems together with interrelationships. There is a relationship between fault, error and failure which is shown in Figure 1

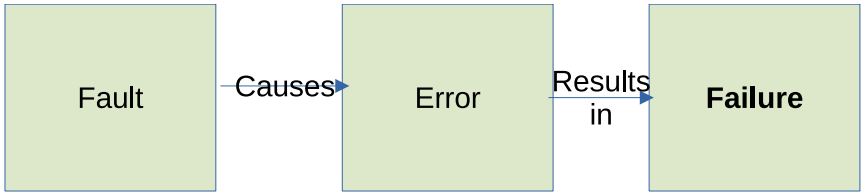

Figure 1: Relationship fault-error-failure

Definition

Fault: It is a defect within a system.

Error: It is observed by a deviation from the expected behaviour of the system

Failure: occurs when the system can no longer perform as required

Fault Tolerance: Is the ability of the system to provide a service, even in the presence of errors.

From the above definitions, it can be seen that an error is a manifestation of a fault in the system figure 2, which could lead to system failure, Failure recovery is a process that involves restoring an erroneous state to an error-free state. Classification of failure:

Process failure: The computation results in an incorrect outcome. The process causes the system state to deviate from the specification. The process may fail to progress.

System failure: It occurs when processors fail to execute.

Secondary storage failure: It happens when secondary storage fails to function like during archiving or logging activity.

Communication medium failure: It occurs when messages fail to receive or send.

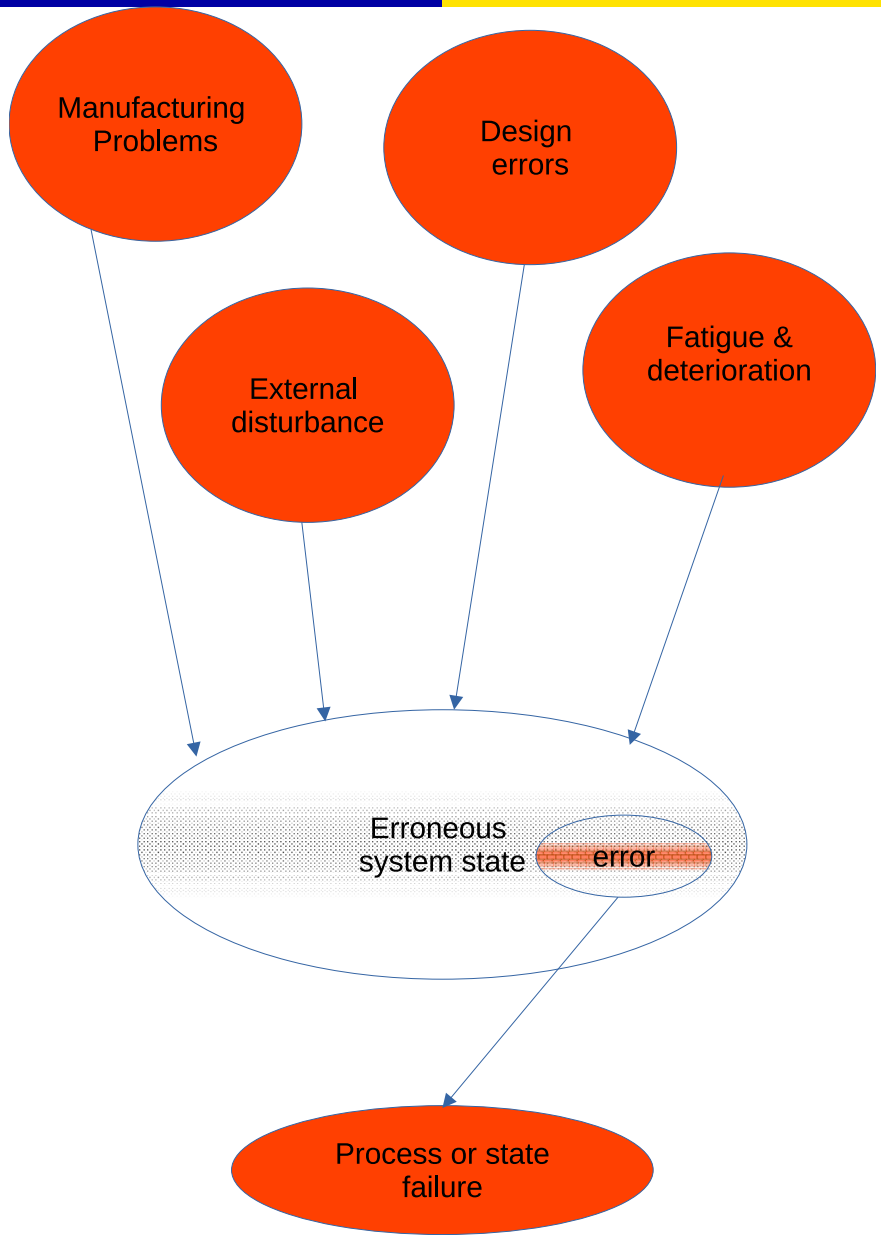

Figure 2: An error is a manifestation of a fault in the system

System failure can be further classified into:

Amnesia failure: occurs when a system restarts in a predefined state that does not depend upon the state of system before failure

A partial amnesia: occurs when a system restarts in a state wherein a part of state is the same as the state before the failure & the rest of the state is predefined.

A pause failure: occurs when a system restarts in the same state it was before the failure

A halting failure: occurs when a crashed system never restarts

Backward and Forward error recovery I

- Error is that part of the state that differs from its intended value & can load to a system failure, & failure recovery is a process that involves restoring an erroneous state to an error-free state
- There are two approaches for restoring an erroneous state to an error free state
  - If the nature of errors & damages caused by faults can be completely & accurately accessed, then it is possible to remove these errors in the process's state & enable to more forward. This technique is called as *forward-error recovery*.
  - If it is not possible to foresee the nature of faults & to remove all the errors in the processes state, then the process's state can be restored to previous error-free state. This technique is known as *backward-error recovery*.

Backward and Forward error recovery I

- The major problem associated with the backward error recovery approach are:
  - *Performance penalty*-the overhead to restore a process state to a prior state can be quite high
  - there is no guarantee that faults will not occur again when processing begins from prior state
  - some component of the system state may be unrecoverable (Cash dispensed from the ATM).

Backward Error recovery I

- In this recovery approach, a process is restored to a prior state in the hope that the prior state is free from errors.
- the points in the execution of a process to which the process can later be restored is known as *recovery points*.
- There are two ways to implement
  - operation-based approach
  - state-based approach

System Model I

- the system is assumed to be composed of single machine

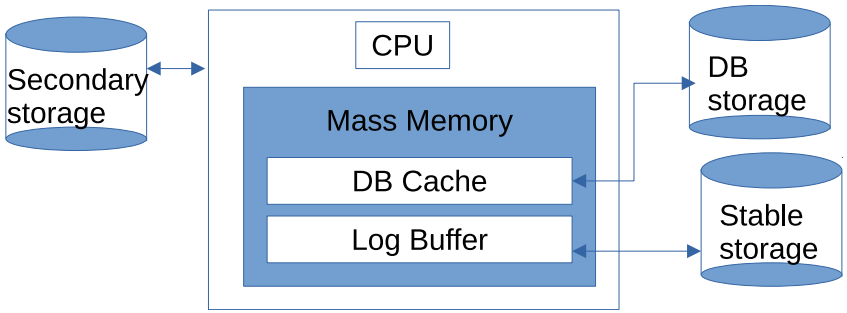

Figure 3: System Model

- Stable storage do not loose information in the event of system failure
- is used to keep logs & recovery points

Operation-based Approach I

- All modifications that are made to the state of a process are recorded in sufficient detail so that a previous state of the process can be restored by reversing all the changes made to the state. The record of the system activity is known as *audit trail or a log*.
- there are two scheme used in this approach:
  - Updating In-place
  - Write-ahead-log protocol

Updating In-place Scheme I

Under this scheme every update operation to an object updates, the object & results is a log to be recorded in a stable storage which has enough information to completely undo or redo the operation. The information recorded includes:-

- The name of the object (Do)
- The old state of the object(Used for UNDO)
- The new state of the object(Used for REDO)

Write-Ahead-log Protocol I

In this approach, recoverable update operation is implemented by the following operations:-

- update an object only after the UNDO log is recorded
- before commit, redo and undo logs are recorded

State-based Approach I

- record a snapshot of the state
- restore state by reloading snapshot
- use of shadow pages can reduce size of checkpoints

Shadow Page

- whenever a process wants to modify an object the page containing the object is duplicated & is maintained on a stable storage.
- the process updates only one copy. The unmodified copy is called shadow page.
- if the process fails, the modified copy is discarded & the shadow page is used.
- if the process commits the shadow page then it is discarded.

Recovery in Concurrent Systems I

- In concurrent systems, several processes cooperate by exchanging information to accomplish a task. The information exchange can be through a shared memory.
- In such systems, if one of the cooperating processes fails & resumes execution from a recovery point, then the effects it has caused at other processes due to the information it has exchanged with them after establishing the recovery point will have to be undone
- to undo the effects caused by a failed process at an active process the active process must also rollback to an earlier state
- thus in concurrent systems all cooperating processes need to establish recovery points.
- The following shows the rolling back of processes can cause further problems:-
  - Orphan Messages & the Domino Effect

Recovery in Concurrent Systems II

- Lost Messages
- Problem of livelocks

Orphan Messages & the Domino Effect I

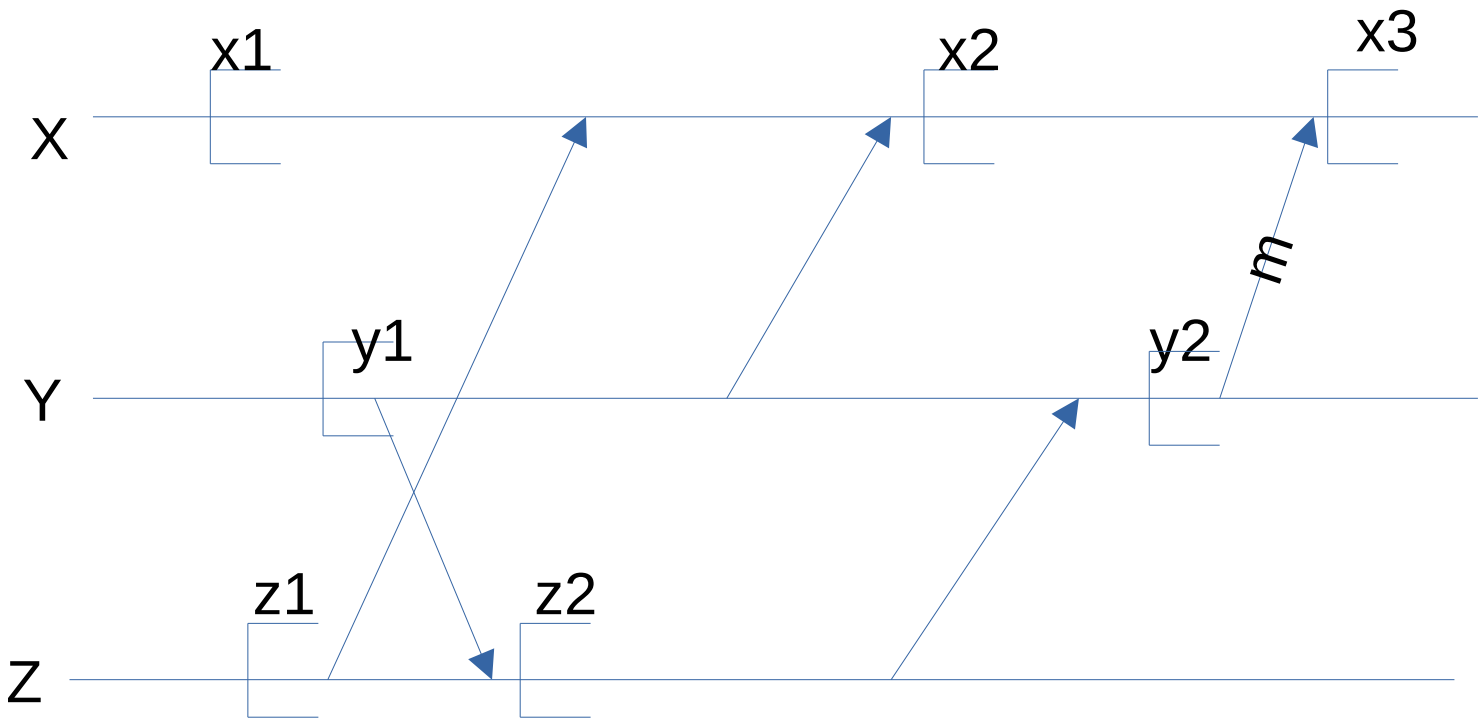

Figure 4: Domino Effect

Lost Messages I

A message whose sending event is recorded, but its receiving event is not recorded (see Figure 5).

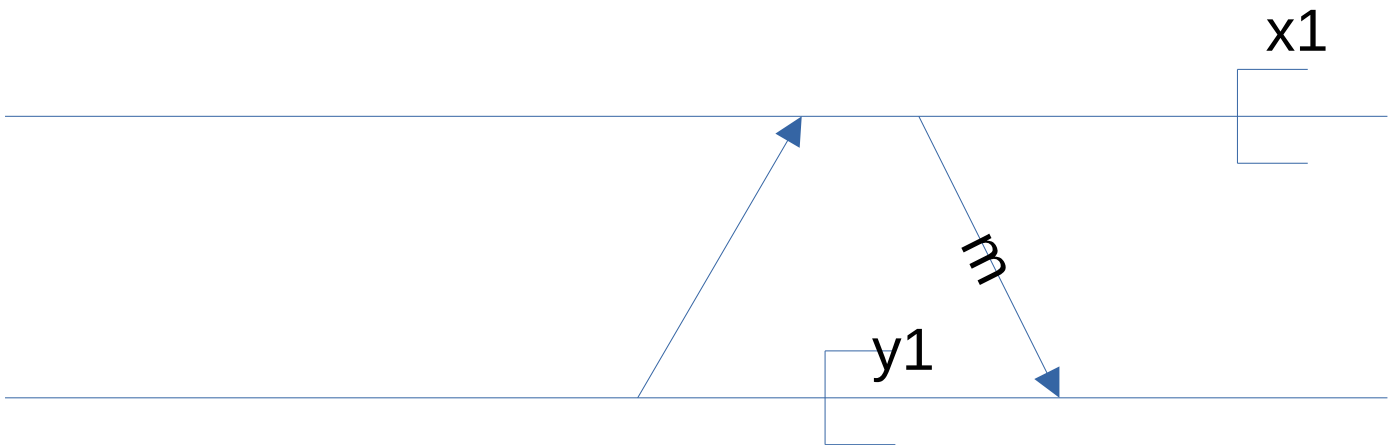

Figure 5: Lost Messages

Livelock I

It is a situation in which a single failure can cause an infinite number of rollbacks, preventing the system from making progress. These situations are shown in Figure 6 and Figure 7

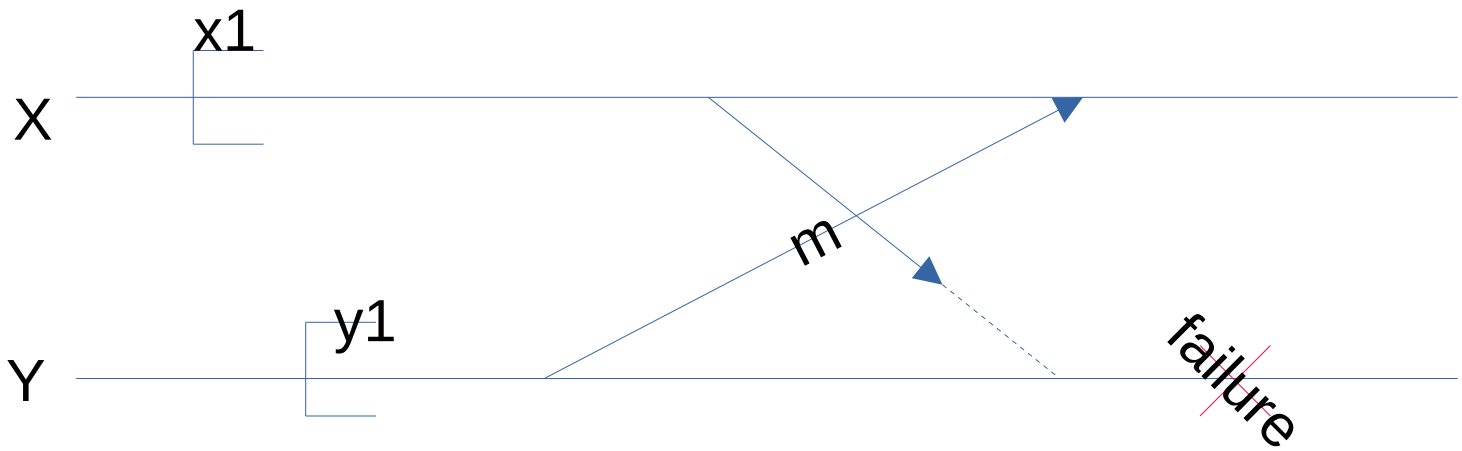

Figure 6: Livelocks

Livelock II

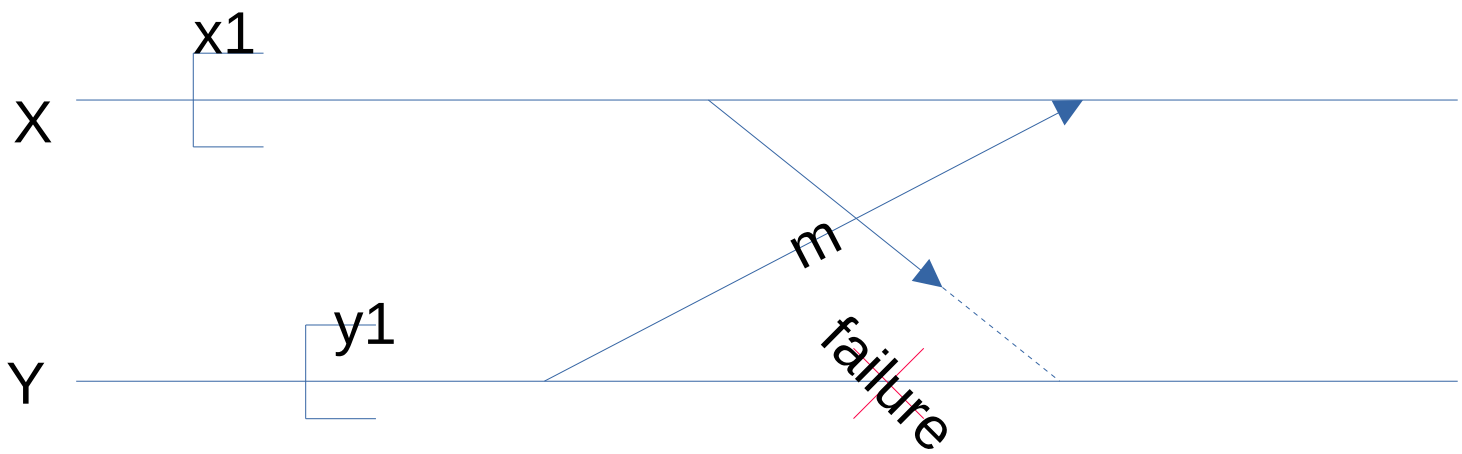

Figure 7: Livelocks

Consistent Checkpoints I

- A process saves its local state on the stable storage, which called as local checkpoint
- the process of saving checkpoints are called as checkpointing
- All the local checkpoints, one from each site collectively form a Global checkpoint.
  - Strongly consistent set of checkpoints
    - The domino effect is caused by orphan messages, which themselves are due to rollbacks
    - To overcome the domino effect, a set of local checkpoints is needed such that no information flow takes place between any set of processes without saved by checkpoints
    - A set of checkpoints is known as recovery line or strongly consistent set of checkpoints
    - A Global checkpoint is strongly consistent set of checkpoints if there is no orphan message & no lost message. (x1,y1 and z1) are strongly consistent checkpoints as shown in Figure 8.

Consistent Checkpoints II

- Consistent set of checkpoints
  - A consistent set of checkpoints is in which each message recorded as received in a checkpoint should also be recorded as sent in another checkpoint. (x2,y2,z2) as shown in Figure 8
- A simple method of taking a consistent set of checkpoints.
  - if every process takes a checkpoint after sending every message, the set of most recent checkpoints are always consistent.
  - however, it has a high overhead
  - so one may reduce the overhead, by taking checkpoint after every k, where k$>$1.

Consistent Checkpoints III

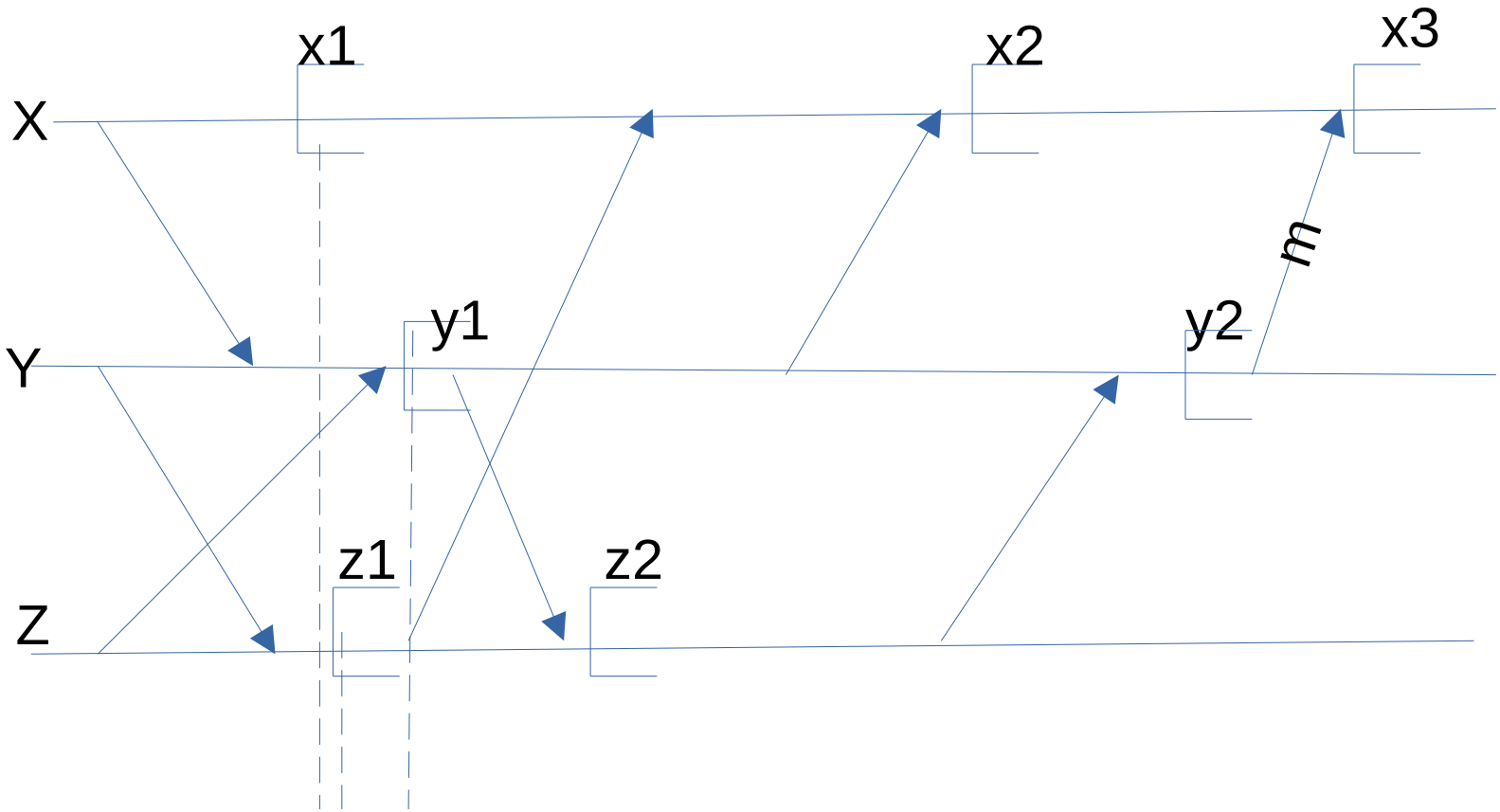

Figure 8: Livelocks

Synchronous Checkpoint & Recovery I

Checkpointing & error recovery technique by Koo & Toueg takes a consistent set of checkpoints & avoids lovelock problems during the recovery.
The Checkpoint Algorithm (Koo & Toueg).
The algorithm assumes the following characteristics for the DS:

- Process communicate by exchanging messages through channels.
- Channels are FIFO in nature.
- End-to-End protocols cope with message loss due to rollback recovery
- Communication failures do not partition the network.
- Algorithm uses two kinds of checkpoints on stable storage.
  - Permanent: A local checkpoint at a process & is a part of consistent Global checkpoint.
  - Tentative: A temporary checkpoint that is made permanent checkpoint on successful termination of the checkpoint algorithm.

Synchronous Checkpoint & Recovery II

This is a 2 Phase Algorithm:

- Synchronous checkpoint Phase 1:
  - An initiating process takes a tentative checkpoint & requests all the processes to take tentative checkpoints.
  - other processes: can respond 'yes' or 'no'
  - Initiator: Decides to make checkpoints permanent if everyone has responded 'yes'. Otherwise all the tentative checkpoints are discarded.
- Synchronous Checkpoint Phase 2:
  - Initiator Process: Informs all the processes of Phase 1to commit or abort.
  - other processes act accordingly
  - between tentative checkpoint & commit/abort of checkpoint process must hold back messages.

Synchronous Checkpoint & Recovery III

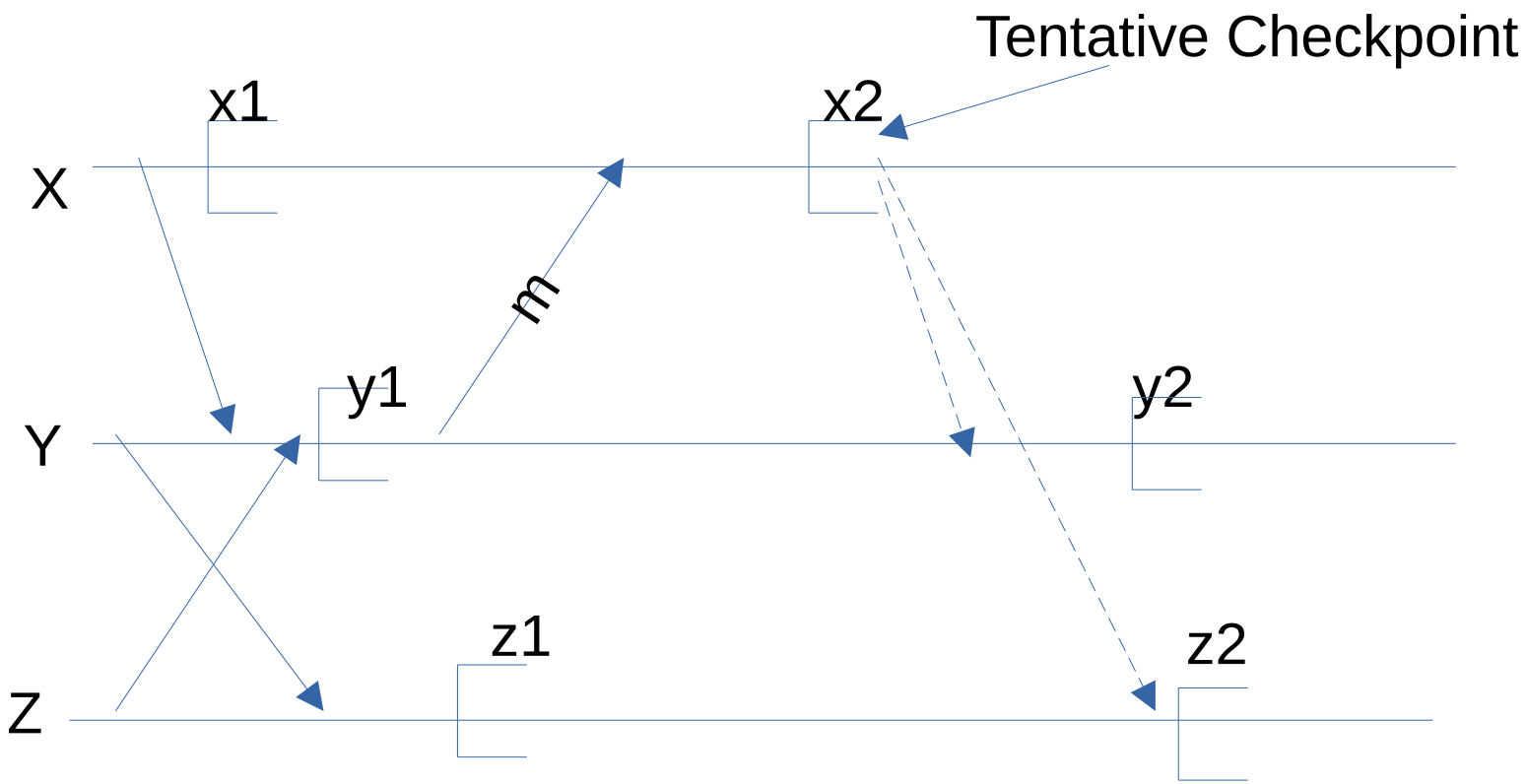

Figure 9: Checkpoints taken unnecessarily

Synchronous Checkpoint & Recovery I

Checkpoints can be taken unnecessarily:

- (x1,y1,z1) are consistent set of checkpoints
- X decides to initiate the checkpoint after receiving message, m.
- (x2,y2,z2) form consistent set of checkpoints
- (x2,y2,z1) can also form consistent set of checkpoint.

Synchronous Checkpoint & Recovery I

We now have to avoid these unnecessary checkpoints. Refer Figure 9

- Record all messages sent & received after the last checkpoint
- When X request Y to take a tentative checkpoint, X sends the last message, received from Y, with the request.
- Y takes tentative checkpoint only if the last message received by X from Y was sent after Y had set the last checkpoint.

The Rollback Recovery Algorithm I

- It assumes that a single process invokes the algorithm, as opposed to several processes concurrently invoking it to rollback & recover.
- It also assumes that checkpoint & rollback recovery algorithm are not currently invoked.
- The rollback algorithm has two phase:- Rollback Recovery phase 1:
  - Initiator: Check whether all processes are willing to restart from the previous checkpoints.
  - A process may reply "no" to a restart request if it is already participating in a checkpointing or a recovering process initiated by some other process.
  - If "yes" is received by all the processes, then it decides to restart otherwise Pi will decide that the process should continue with their normal operations.
- Rollback Recovery phase 2:
  - Initiator: propagate go/nogo decision to all processes.

The Rollback Recovery Algorithm II

- others: Carryout the decision of the initiator.
- between request to rollback & decision, no one sends other messages.

- Rollback recovery properties:
  - All or none of the processes restart from checkpoints
  - After rollback all processes resume a consistent state
  - In the event of failure of process X, would require processes X,Y ,and Z to restart from checkpoints (x2,y2,z2), respectively.
  - However that process Z need not have to be rollback as there was no interaction between Z & other two processes.

The Rollback Recovery Algorithm III

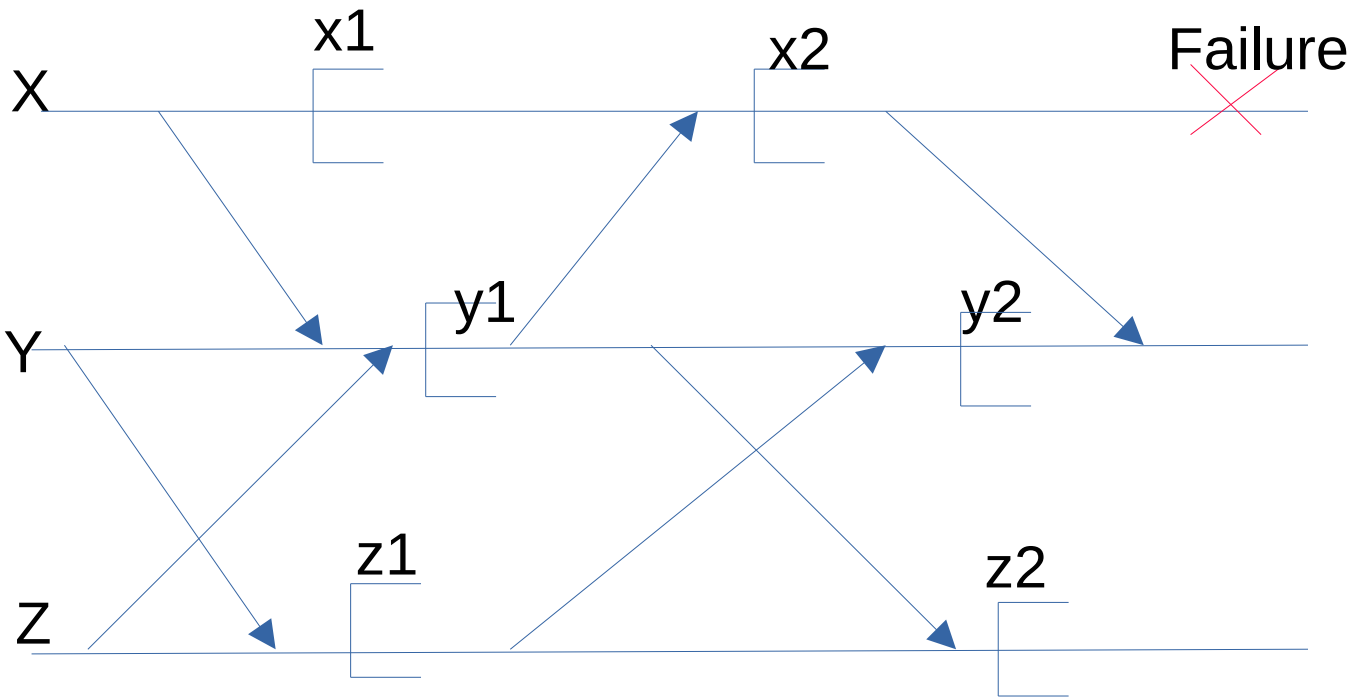

Figure 10: Unnecessary Rollbacks

Asynchronous Checkpointing and Recovery I

While synchronous checkpointing simplifies recovery, it has following drawbacks:-

- Additional messages are exchanged by the checkpoint algorithm when it takes each checkpoint.
- Synchronization delays are involved during normal operations.
- If failure occurs between successive checkpoints, then the synchronous approach places unnecessary burden on the system.

Asynchronous Checkpointing and Recovery I

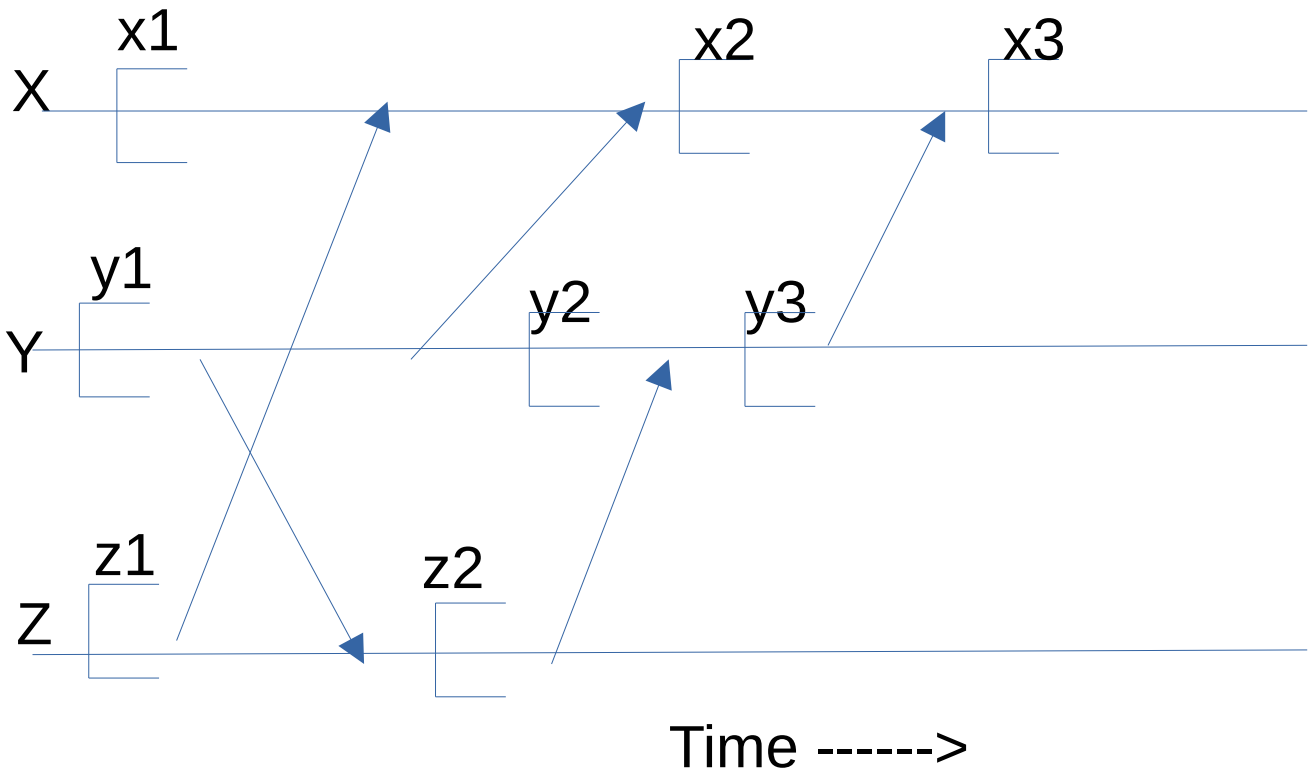

Figure 11: Asynchronous checkpointing may not result in a consistent set of checkpoints

Asynchronous Checkpointing and Recovery II

The consistent set of checkpoints (x2,y2,z2), but most recent is (x3,y3,z2)

Asynchronous Checkpointing and Recovery I

To initialise the amount of computation undone during the rollback, all incoming messages are logged at each processor. The messages are logged in two ways:-

pessimistic: log each message before processing

optimistic: buffer messages & log in batches

A Scheme for Asynchronous Checkpointing and Recovery I

Assumptions:-

- The communication channels are reliable
- The communication channels deliver the messages in the order they were sent.
- The communication channels are assumed to have infinite buffer.
- The message transmission delay if arbitrary, but finite.
- The underlying communication is assumed to be event-driven

Algorithm I

Steps:

- At each event a triplet s,m,msgs_sent is put in the log where s is state, m is messing causing the event and msgs_sent is the set of messages sent
- Two data structures used are:-
  - RCVD(i,j,checkpoint) the number of message received from process$_i$ from process$_j$ at checkpoint
  - SENT(i,j,checkpoint) the number of message sent from process$_i$ from process$_j$ at checkpoint
- use the message send / receive counts to determine the point to rollback

Algorithm II

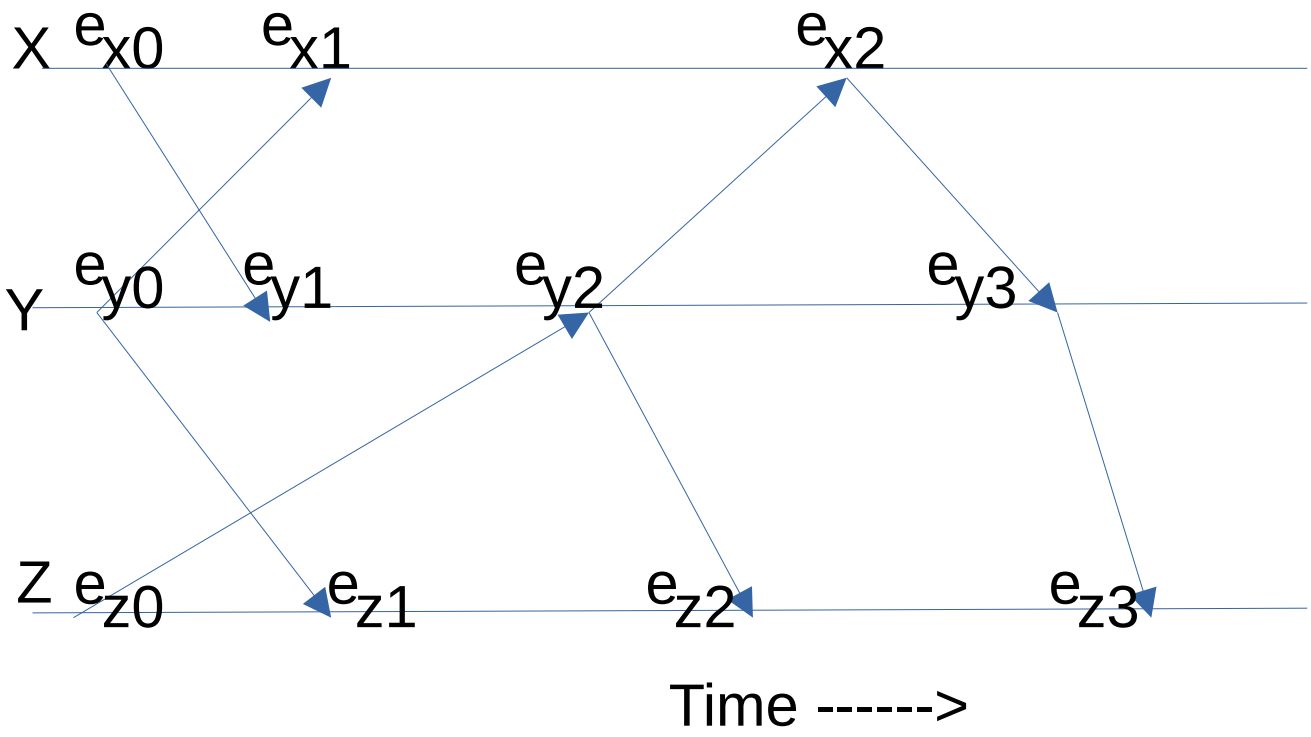

Figure 12: event drive computation