An error is a manifestation of a fault that can lead to a failure.

Failure Recovery:
- backward recovery
- operation-based (do-undo-redo logs)
- state-based (checkpoints)
- forward recovery

Domino Effect

Cases:
- X fails after $x_1$
- Y fails after sending message $m$
- Z fails after sending message $n$

Orphan Message

The set of checkpoints $\{x_1, y_1, z_1\}$ are strongly consistent - no recovery ever need rollback past this set. The set stops the domino effect. $\{x_2, y_2, z_2\}$ is a consistent set.

Check Point Algorithm

(1) When must I take a checkpoint?
(2) Who else has to take a checkpoint when I do?

Check Point Algorithm

Initiator Process $P_i$:

for all $p \notin \text{cohorts}(P_i)$
    send TakeTentativeCheckpoint($P_i$, LastLabelFrom($p$));

if all cohorts replied "yes" then
    for all $p \notin \text{cohorts}(P_i)$
        send MakeCheckpointPermanent;
else
    for all $p \notin \text{cohorts}(P_i)$
        send UndoTentativeCheckpoint;
**Checkpoint Algorithm**

A Cohort process, p:

- On receiving `TakeTentativeCheckpoint(q, LastLabelFrom(p))`:
  - if `AgreeToCheckpoint` and `NeedToCheckpoint` then
    - `TakeTentativeCheckpoint`;
    - for all `r ∈ cohorts(p)`
      - `send TakeTentativeCheckpoint(p, LastLabelFrom(r))`;
    - if all cohorts replied "yes" then
      - `AgreeToCheckpoint = yes`;
    - else
      - `AgreeToCheckpoint = no`;
    - `send(p, AgreeToCheckpoint)` to `q`;
  - On receiving `MakeCheckpointPermanent`
    - `make checkpoint permanent`;
    - for all `r ∈ cohorts(p)`
      - `send MakeCheckpointPermanent`;
  - On receiving `UndoTentativeCheckpoint`
    - `undo tentative checkpoint`;
    - for all `r ∈ cohorts(p)`
      - `send UndoTentativeCheckpoint`;

**Rollback Algorithm**

(1) When must I rollback?
(2) Who else might have to rollback when I do?

![Diagram](diagram.png)

- If (me) have received a message from the checkpointing process, p, since p’s last checkpoint.
  - `LastLabelFrom(p) >= LastLabelTo, (me)`
- Any other process to whom I can send messages.

**Rollback Algorithm**

Initiator Process `P_i`:

- for all `p ∈ cohorts(P_i)`
  - `send PrepareToRollback(P_i, LastLabelTo(p))`;
- if all cohorts replied "yes" then
  - for all `p ∈ cohorts(P_i)`
    - `send Rollback`;
- else
  - for all `p ∈ cohorts(P_i)`
    - `send DoNotRollBack`;

```
if AgreeToRollback and NeedToRollback and NotInProgress then
  NotInProgress = false;
  for all `r ∈ cohorts(p)`
    send PrepareToRollback(p, LastLabelTo(r));
if all cohorts replied "yes" then
  AgreeToRollback = yes;
else
  AgreeToRollback = no;
  send(p, AgreeToRollback) to `q`;
```

On receiving `Rollback` AND if `NotInProgress` is false:

- *restart from p’s permanent checkpoint*;
- for all `r ∈ cohorts(p)`
  - `send Rollback`;

On receiving `DoNotRollBack`:

- *resume execution*;
- for all `r ∈ cohorts(p)`
  - `send DoNotRollBack`;

```