A Survey of Rollback-Recovery Protocols in Message-Passing Systems

1. Introduction

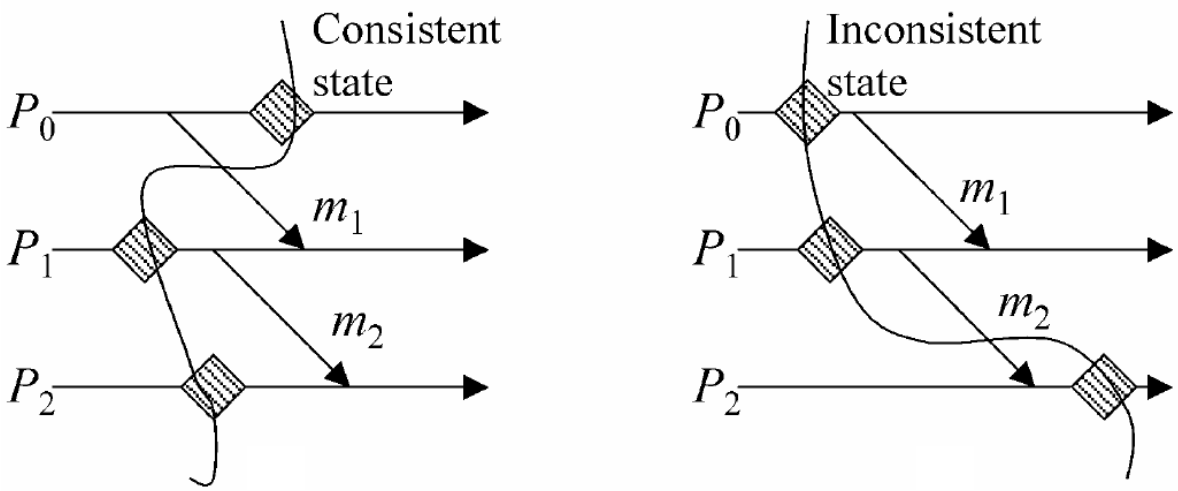
- Terms
  - Transparent rollback recovery
  - Checkpoint
  - Rollback propagation
  - Domino effect
  - Checkpoint-based rollback recovery
  - Log-based rollback recovery
  - Garbage collection
  - Interactions with the outside world

2. *Background and Definitions*

- System Model
  - A message-passing system consists of a fixed # of processes
  - Rollback-recovery protocols assume that
    - Communication network is immune to partitioning
    - Network partition is different from network reliability
  - A process execution
    - Is a sequence of state intervals
  - PWD assumption (log-based rollback)
    - The system can detect and capture sufficient events that initiate the state intervals

2. Background and Definitions (Cont.)

- Consistent System States



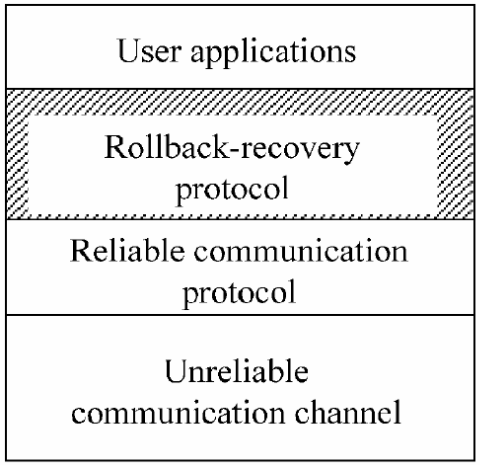
- Goal of rollback-recovery protocol
  : to bring the system into a consistent state when inconsistencies occur because of a failure

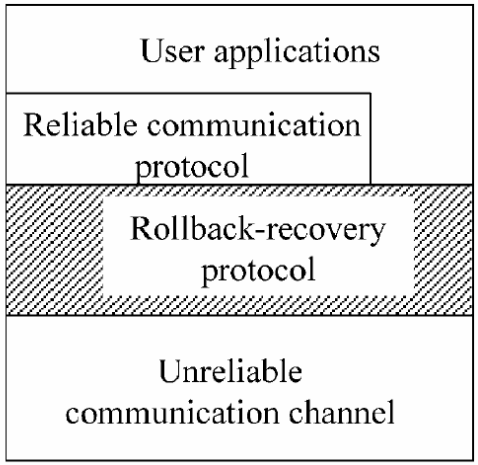
2. *Background and Definitions* (Cont.)

- Interactions with the Outside World
  - The outside world can't roll back
    (ex: printer can't roll back)
  - OWP (Outside World Process)
    - Can't fail
    - Can't maintain state
    - Can't participate in the recovery protocol
  - Output commit problem
    - Before send msg to OWP, system ensure that the state from which the message is sent will be recovered despite any future failure

2. *Background and Definitions* (Cont.)

- In-Transit Messages
  - the message has been sent but not yet received
  - Rollback-recovery protocols must guarantee the delivery of in-transit msg
  - Reliable communication protocol & unreliable communication protocol





2. *Background and Definitions* (Cont.)

- Logging protocols
  - Piecewise Deterministic
    - identify all the nondeterministic events, log a determinant that contains all information necessary to replay the event
    - Don't occur the domino effect
  - Ensure consistent without expensive checkpoints
    - It's good for repeatedly interaction with OWP
- Stable Storage
  - Save checkpoints, event logs, recovery-related information
  - Must ensure that the recovery data persist

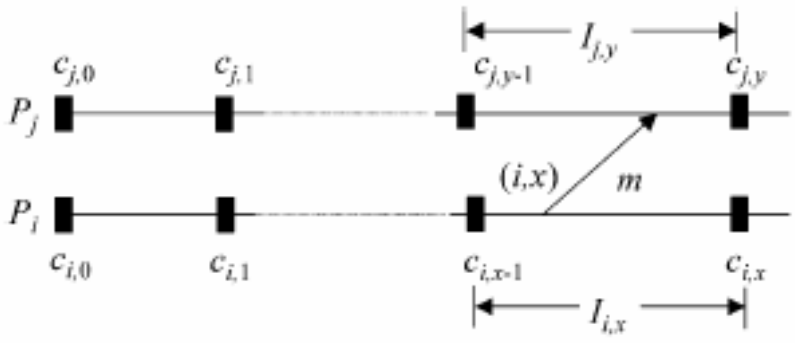
2. *Background and Definitions* (Cont.)

- Garbage Collection
  - the deletion of useless recovery information
    - Checkpoints and event logs consume storage resources
    - Discard useless information incurs overhead
  - Need strategies for GC
    - Complexity
    - Invocation frequency

3. *Checkpoint-based Rollback Recovery*

- Restores the system state to the most recent consistent set of checkpoints
- Does not rely on the PWD assumption
  - → detect, log, replay non-deterministic events.
    : not need
  - → easy to implementation
  - → doesn't guarantee pre-failure execution can be deterministically regenerated after a rollback
- Categories
  : Uncoordinated, Coordinated, CIC

3. Checkpoint-based Rollback Recovery



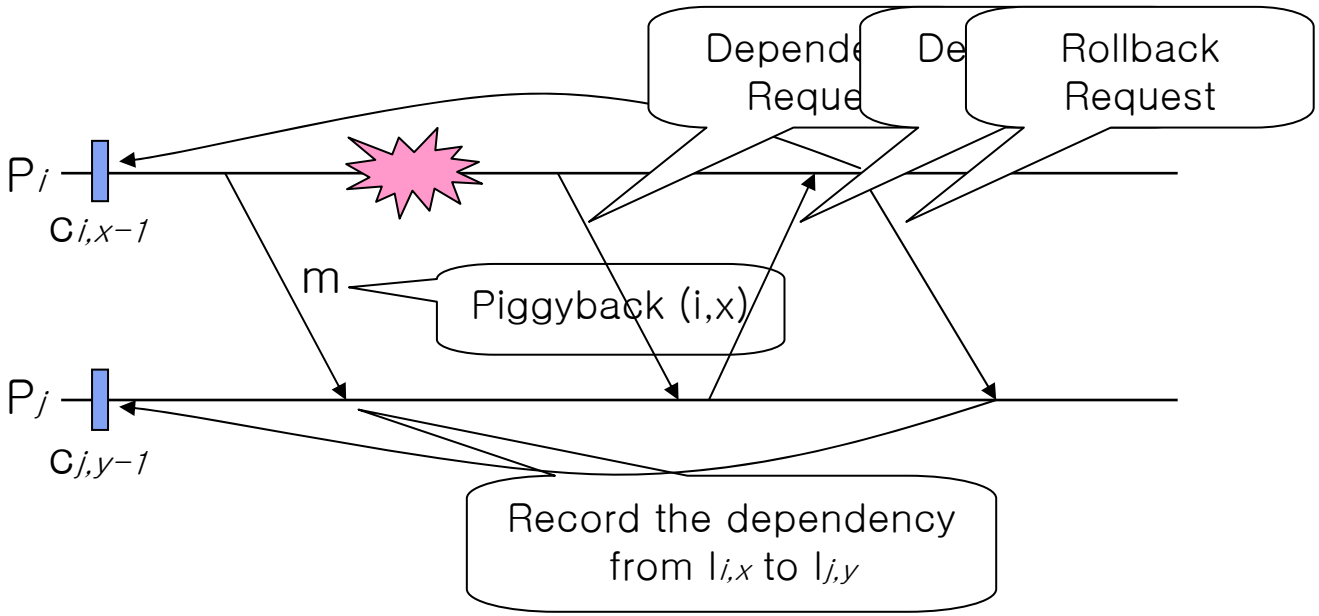
Checkpoint Index and Checkpoint Interval

3.1. Uncoordinated Checkpointing

- Overview
  - Allows each process the maximum autonomy
  - Advantage
    - Each process take a checkpoint when convenient
  - Disadvantages
    - Possibility of the domino effect
    - Produce of useless checkpoint won't be part of a global consistent state
    - Require the garbage collection algorithm
    - Not suitable for applications with frequent output commits

3.1. Uncoordinated Checkpointing (Cont.)

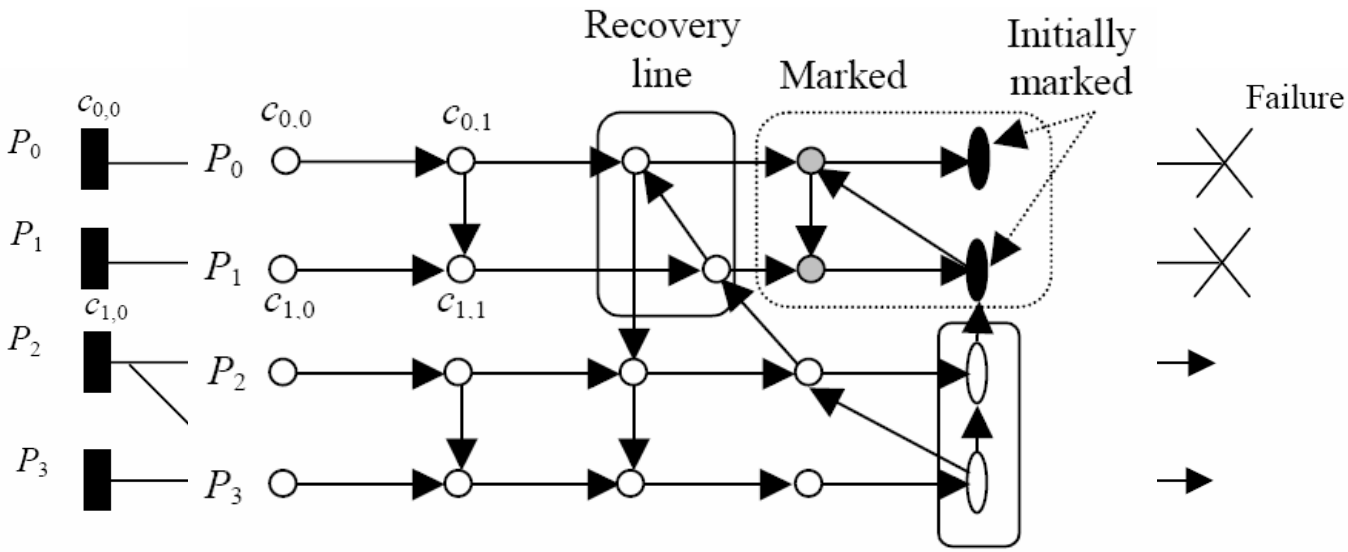
* Simple showcase



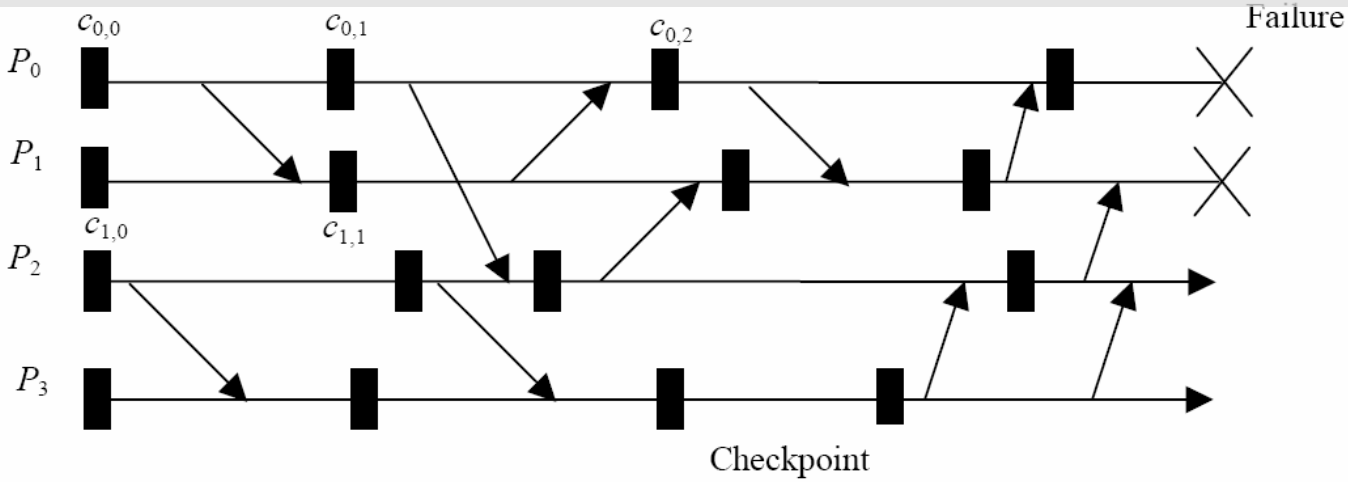
3.1. Uncoordinated Checkpointing (Cont.)

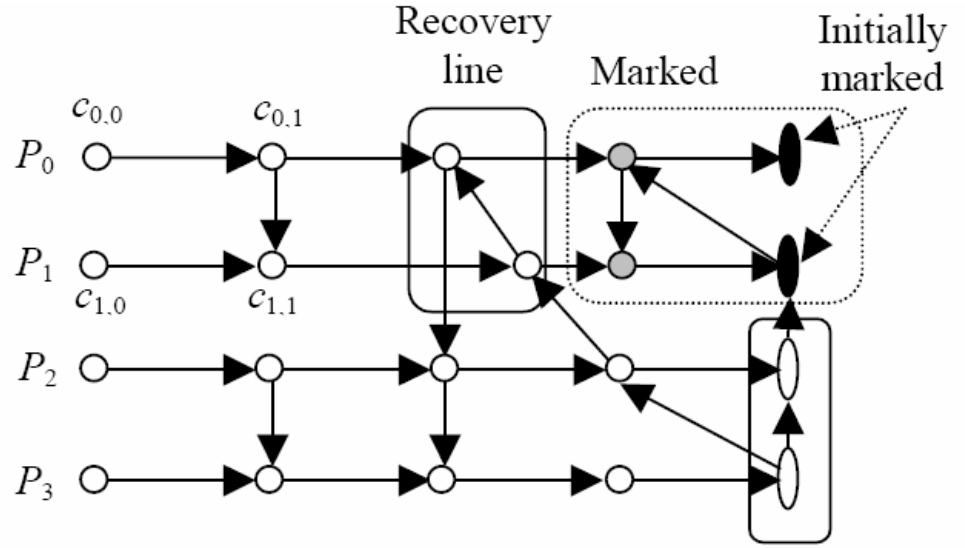
- Recovery Line Calculation
  - Rollback-dependency graph

(1) $i \neq j$, and a message m is sent from $I_{i,x}$ and received in $I_{j,y}$, or
(2) $i = j$ and $y = x + 1$.



(1) $i \neq j$, and a message m is sent from $I_{i,x}$ and received in $I_{j,y}$, or
(2) $i = j$ and $y = x + 1$.

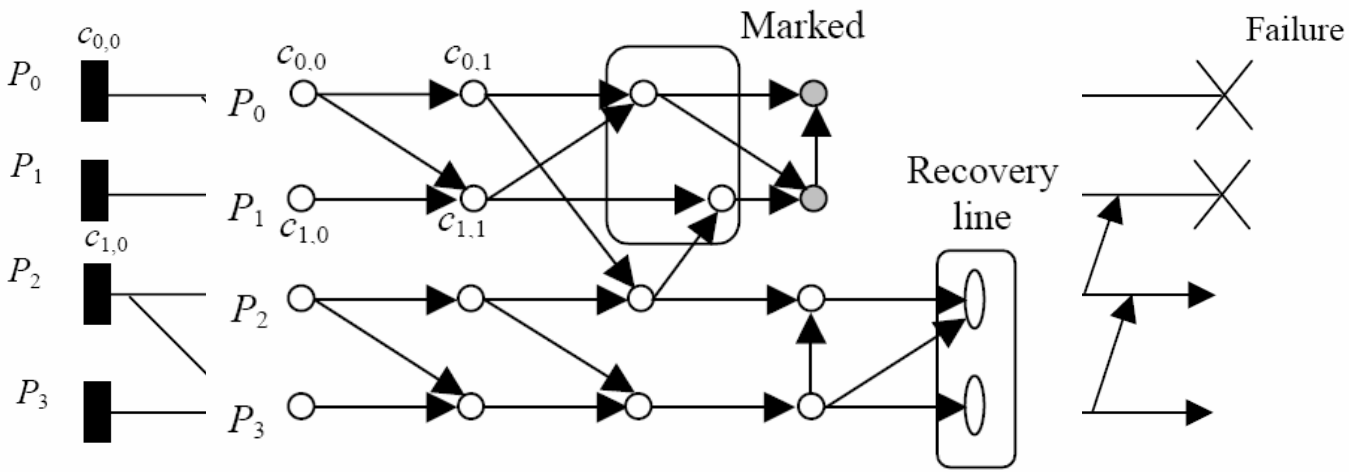


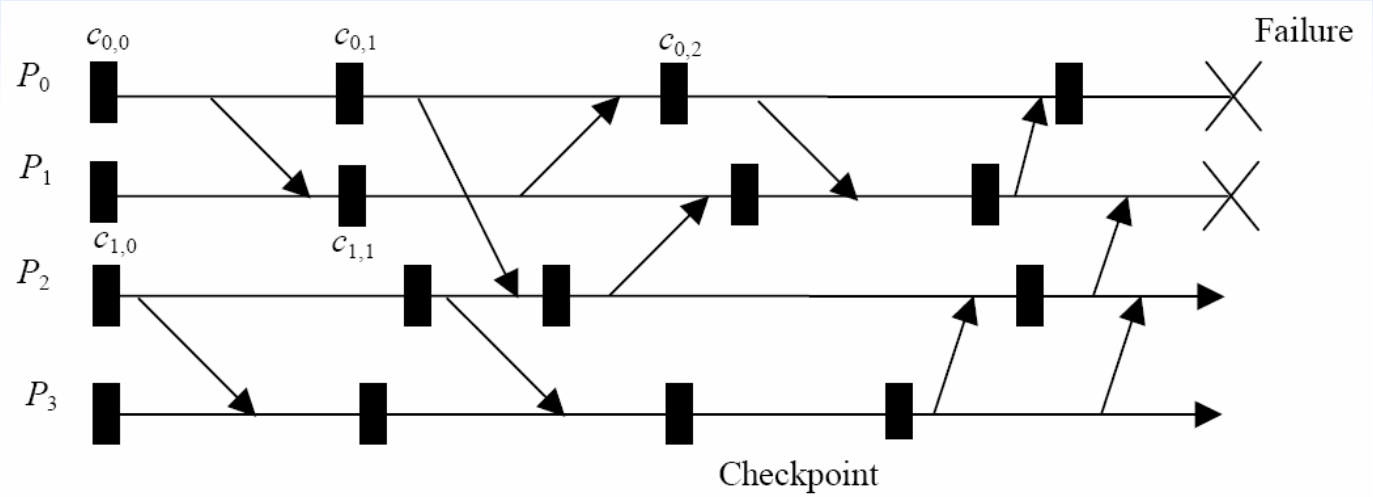


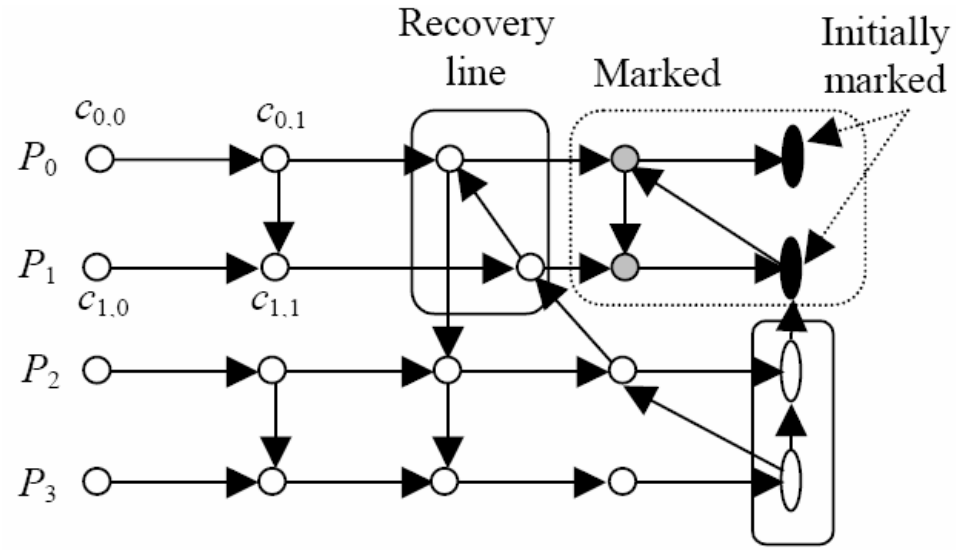
Rollback-dependency graph

3.1. Uncoordinated Checkpointing (Cont.)

- Checkpoint graph
  - Similar to rollback-dependency graph except when a message is sent from $I_{i,x}$ and received in $I_{j,y}$, a directed edge is drawn from $c_{i,x-1}$ to $c_{j,y}$



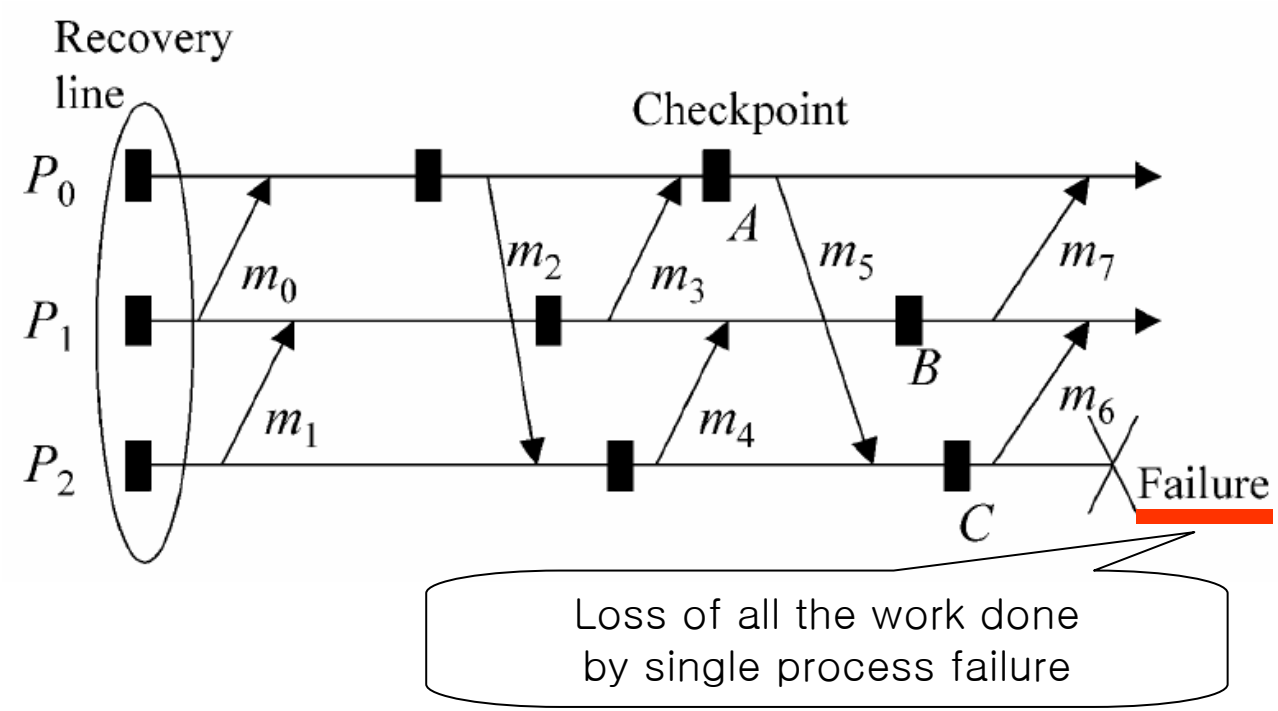




Checkpoint graph

3.1. Uncoordinated Checkpointing (Cont.)

- The Domino Effect
  - Cascaded rollback
    : roll back to beginning of the computation

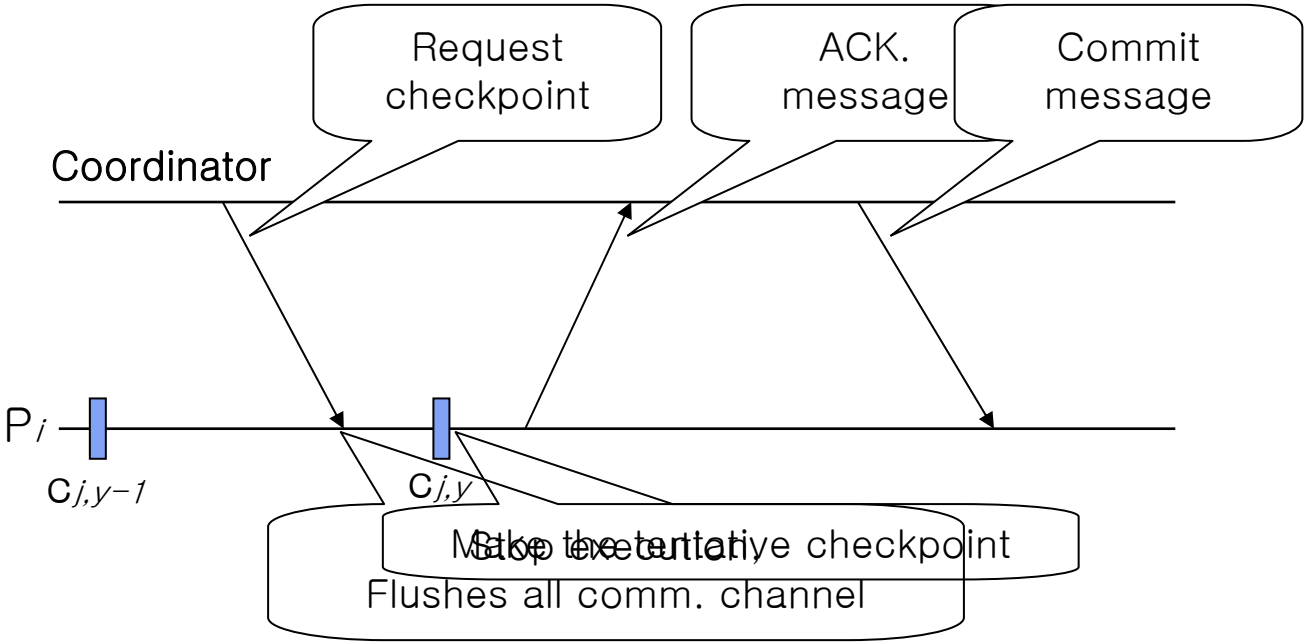


3.2. Coordinated Checkpointing

- Overview
  - Orchestrate all processes checkpoints in order to form a consistent global state
  - Advantages
    - Simply recovery
    - Not susceptible to the domino effect
    - Reduce storage overhead
    - Not need the garbage collection
  - Disadvantage
    - Large latency involved in committing output

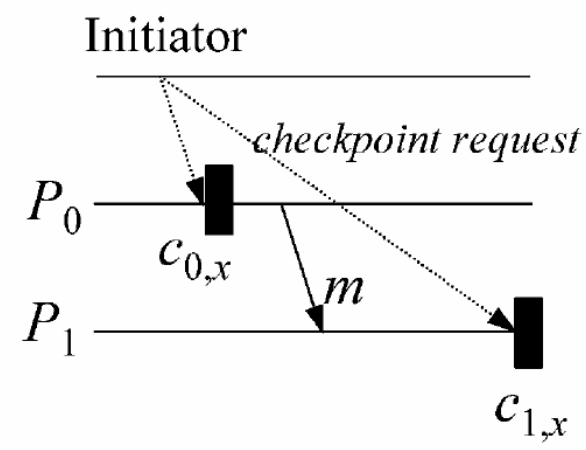
3.2. Coordinated Checkpointing (Cont.)

- Straightforward approach
  - Block communications while the check-pointing protocol executes



3.2. Coordinated Checkpointing (Cont.)

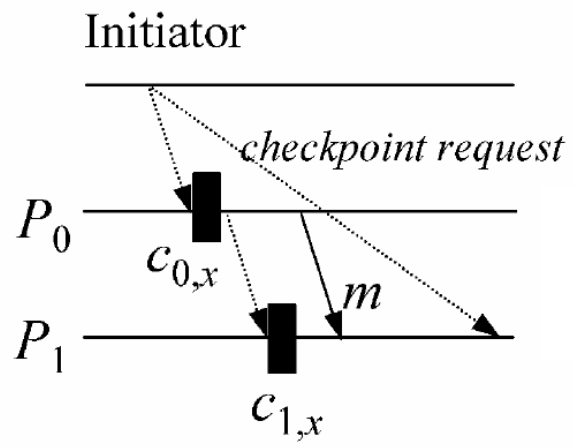
- Non-blocking Checkpoint Coordination
  - Possibility to make the checkpoint inconsistent



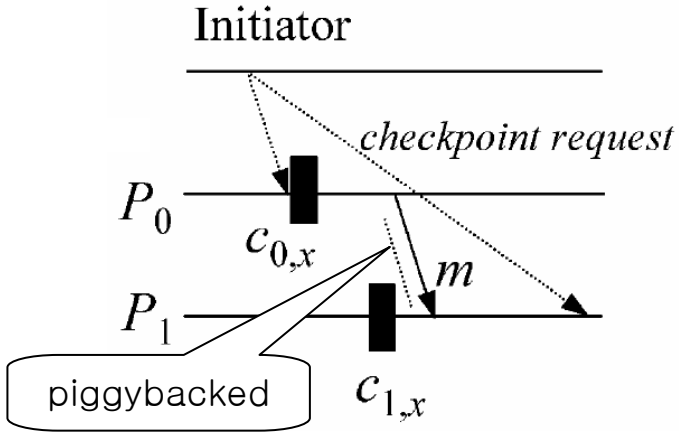
  - Problem : prevent a process that could make the checkpoint inconsistent

3.2. Coordinated Checkpointing (Cont.)

* With FIFO channels



* With non-FIFO channels

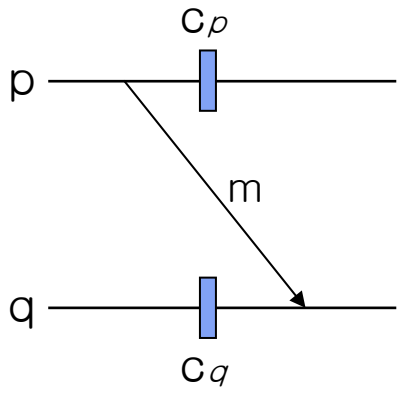


3.2. *Coordinated Checkpointing* (Cont.)

- Checkpointing with Synchronized Clocks
  - Loosely synchronized clocks
  - Trigger all local checkpointing action at approximately the same time without a checkpoint initiator
  - Without the need of exchanging any messages
  - If a failure occurs, it is detected within the specified time and the protocol is aborted

3.2. Coordinated Checkpointing (Cont.)

- Checkpointing and Communication Reliability
  - Problem



  - Solution
    - Reliable channel
      : all in-transit messages be saved by their intended destinations
    - Unreliable channel assumption
      : in-transit messages need not be saved

3.2. *Coordinated Checkpointing* (Cont.)

- Minimal Checkpoint Coordination
  - Coordinated checkpointing requires all processes to participate in every checkpoint
  - So, it is desirable to reduce # of processes
  - Minimal Checkpoint Coordination Protocol
    - First phase
      - Initiator identifies all processes communicated since the last checkpoint
      - Send them a request
      - Each process repeats above until no more processes can be identified
    - Second phase
      - All processes identified take a checkpoint

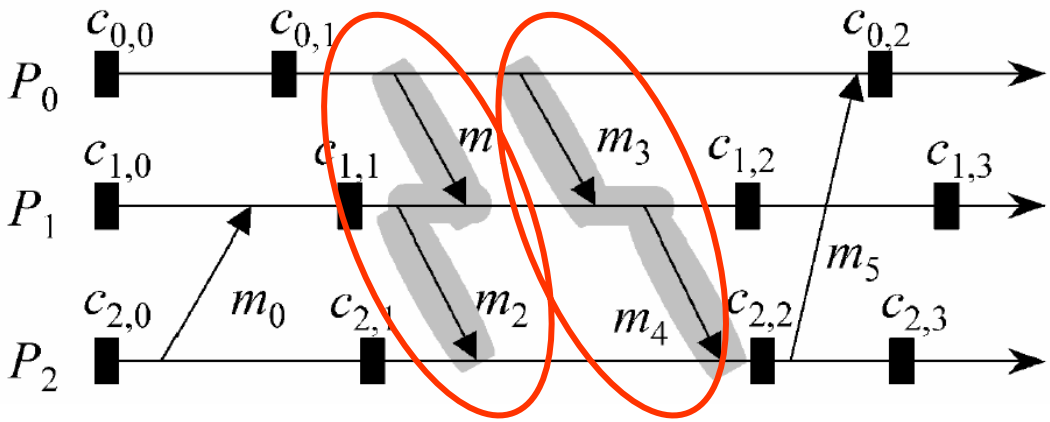
3.3. Communication-induced Checkpointing

- Overview
  - Avoid the domino effect without requiring all checkpoints to be coordinated
  - Local checkpoints : taken independently
  - Forced checkpoints
    : guarantee the eventual progress of the recovery line
  - Prevent the creation of useless checkpoints by these separated checkpoints
  - Not need any special messages
    : instead, use piggyback protocol

3.3. Communication-induced Checkpointing

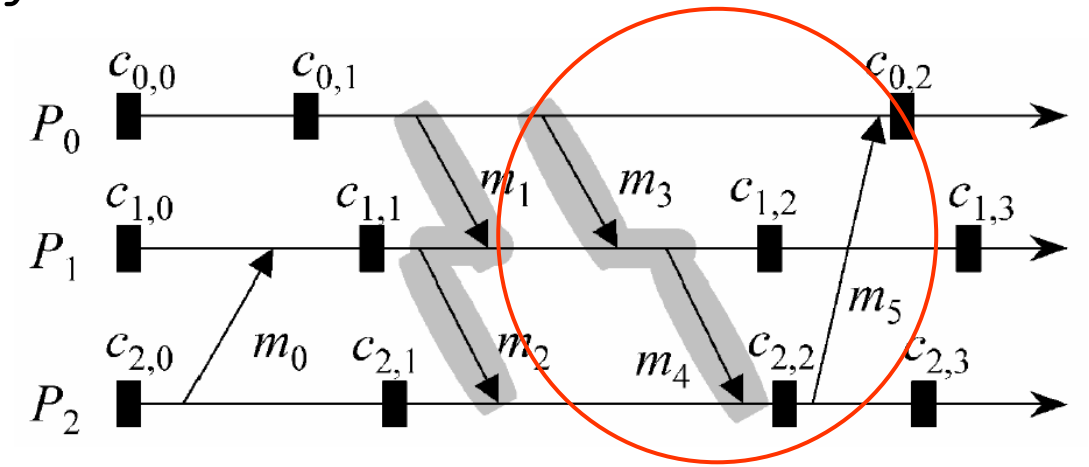
- Z-path

1. $x < y$ and $i = j$; or
2. There exists a sequence of messages $[m_0, m_1, \ldots, m_n]$, $n \geq 0$, such that:
   - $c_{i,x} \mapsto send_i(m_0)$;
   - $\forall\ l < n$, either $deliver_k(m_l)$ and $send_k(m_{l+1})$ are in the same checkpoint interval, or $deliver_k(m_l) \mapsto send_k(m_{l+1})$; and
   - $deliver_j(m_n) \mapsto c_{j,y}$



3.3. Communication-induced Checkpointing

- Z-cycle



  - Is a Z-path that begins and ends with the same checkpoint
  - Z-path [m_5, m_3, m_4] is a Z-cycle
  - Z-cycle can be proved that
    : a checkpoint is useless ↔ part of a Z-cycle

3.3. Communication-induced Checkpointing

- Model-based Protocols
  - Maintain checkpoint and communication structure to prevent useless checkpoint
  - The MRS model
    - By ensure all message-receiving events precede all message-sending events
  - Another way (by Bartlett, 1981)
    - By taking a checkpoint immediately before every message-sending event

3.3. Communication-induced Checkpointing

- Index-based Protocols
  - Guarantee, through forced checkpoint
    - If necessary, $c_{i,m} \mapsto c_{j,n}$, then $ts(c_{j,n}) \geq ts(c_{i,m})$, where ts(c) is the timestamp for checkpoint c and consecutive local checkpoints (increasing)
  - System uses an indexing scheme for the local and forced checkpoints

    → the checkpoints of the same index at all processes form a consistent state
  - performance advantages than others
    - CIC allows considerable autonomy
    - Scale up well with a large # of processes