Checkpointing & Rollback Recovery

Chapter 13

Anh Huy Bui
Jason Wiggs
Hyun Seok Roh

Introduction

- Rollback recovery protocols
  - restore the system back to a consistent state after a failure
  - achieve fault tolerance by periodically saving the state of a process during the failure-free execution
  - treats a distributed system application as a collection of processes that communicate over a network
- Checkpoints
  - the saved states of a process
- Why is rollback recovery of distributed systems complicated?
  - messages induce inter-process dependencies during failure-free operation
- Rollback propagation
  - the dependencies may force some of the processes that did not fail to roll back
  - This phenomenon is called "*domino effect*"

Introduction

- If each process takes its checkpoints independently, then the system can not avoid the domino effect
  - this scheme is called independent or uncoordinated checkpointing
- Techniques that avoid domino effect
  - Coordinated checkpointing rollback recovery
    - processes coordinate their checkpoints to form a system-wide consistent state
  - Communication-induced checkpointing rollback recovery
    - forces each process to take checkpoints based on information piggybacked on the application
  - Log-based rollback recovery
    - combines checkpointing with logging of non-deterministic events
    - relies on piecewise deterministic (PWD) assumption

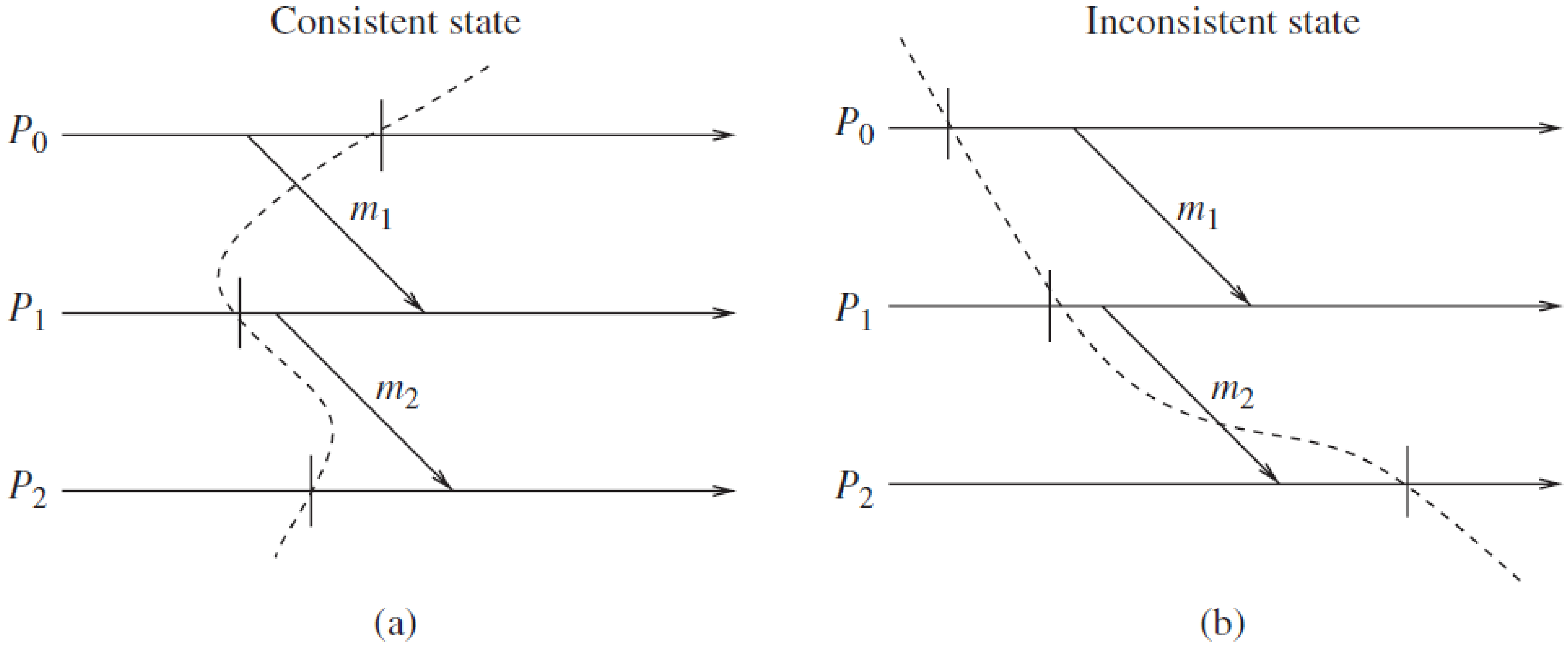
A local checkpoint

- All processes save their local states at certain instants of time
- A local check point is a snapshot of the state of the process at a given instance
- Assumption
  - A process stores all local checkpoints on the stable storage
  - A process is able to roll back to any of its existing local checkpoints
- $C_{i,k}$
  - The *k*th local checkpoint at process P_i
- $C_{i,0}$
  - A process P_i takes a checkpoint $C_{i,0}$ before it starts execution

Consistent states

- A global state of a distributed system
  - a collection of the individual states of all participating processes and the states of the communication channels
- Consistent global state
  - a global state that may occur during a failure-free execution of distribution of distributed computation
  - if a process's state reflects a message receipt, then the state of the corresponding sender must reflect the sending of the message
- A global checkpoint
  - a set of local checkpoints, one from each process
- A consistent global checkpoint
  - a global checkpoint such that no message is sent by a process after taking its local point that is received by another process before taking its checkpoint

Consistent states - examples



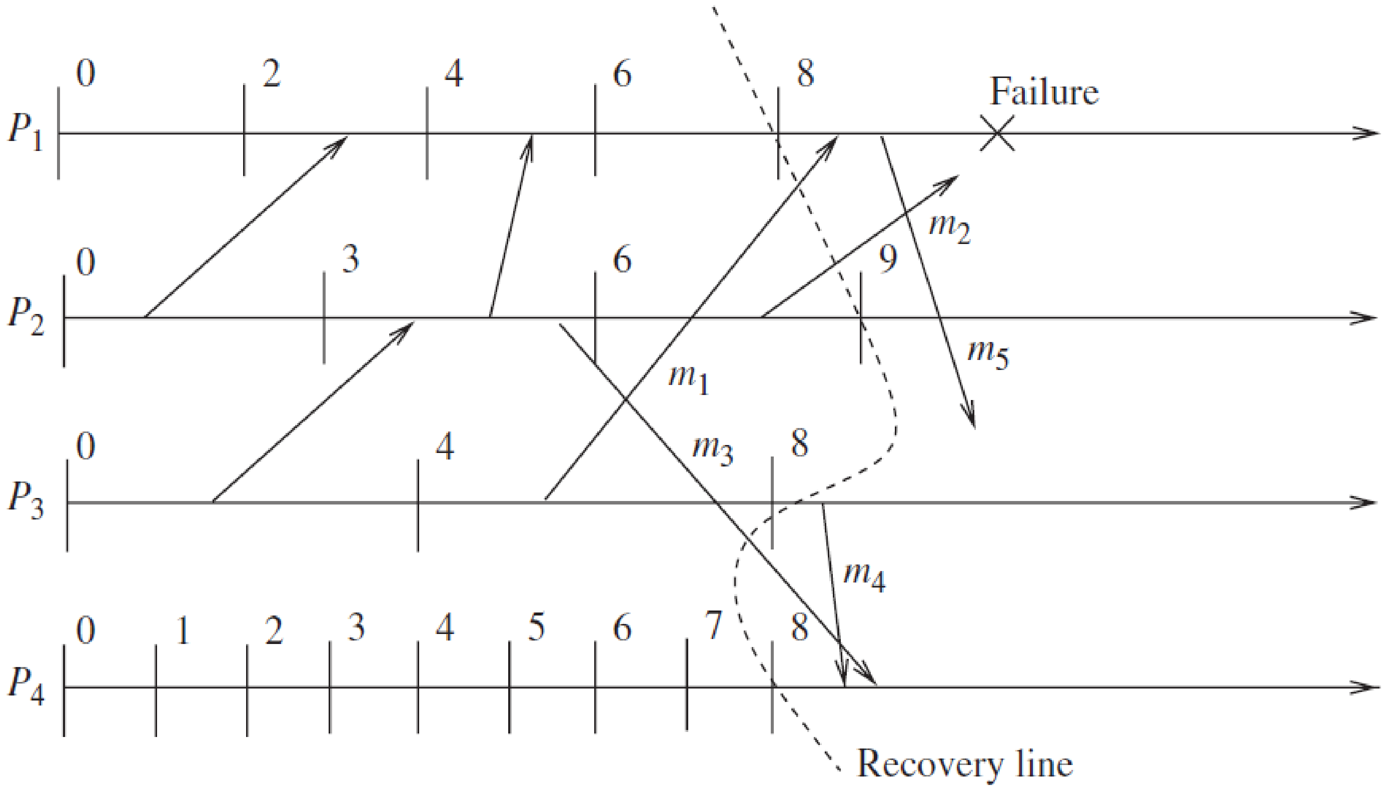
Interactions with outside world

- A distributed system often interacts with the outside world to receive input data or deliver the outcome of a computation
- Outside World Process (OWP)
  - a special process that interacts with the rest of the system through message passing
- A common approach
  - save each input message on the stable storage before allowing the application program to process it
- Symbol “||”
  - An interaction with the outside world to deliver the outcome of a computation

Messages

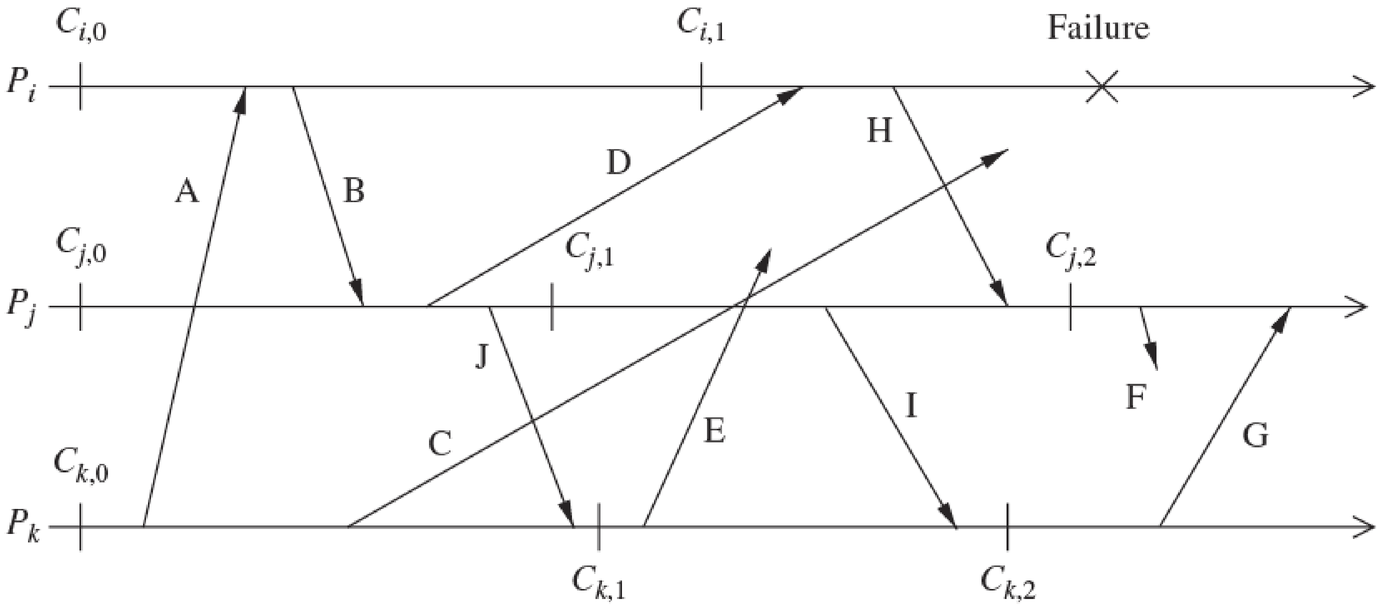
- In-transit message
  - messages that have been sent but not yet received
- Lost messages
  - messages whose ‘send’ is done but ‘receive’ is undone due to rollback
- Delayed messages
  - messages whose ‘receive’ is not recorded because the receiving process was either down or the message arrived after rollback
- Orphan messages
  - messages with ‘receive’ recorded but message ‘send’ not recorded
  - do not arise if processes roll back to a consistent global state
- Duplicate messages
  - arise due to message logging and replaying during process recovery

Messages – example



- In-transit
  - m_1, m_2
- Lost
  - m_1
- Delayed
  - m_1, m_5
- Orphan
  - none
- Duplicated
  - m_4, m_5

Issues in failure recovery



- Checkpoints : $\{C_{i,0}, C_{i,1}\}$, $\{C_{j,0}, C_{j,1}, C_{j,2}\}$, and $\{C_{k,0}, C_{k,1}, C_{k,2}\}$
- Messages : A - J
- The restored global consistent state : $\{C_{i,1}, C_{j,1}, C_{k,1}\}$

Issues in failure recovery

- The rollback of process P_i to checkpoint $C_{i,1}$ created an orphan message H
- Orphan message I is created due to the roll back of process P_j to checkpoint $C_{j,1}$
- Messages C, D, E, and F are potentially problematic
  - Message C: a delayed message
  - Message D: a lost message since the send event for D is recorded in the restored state for P_j, but the receive event has been undone at process P_i.
  - Lost messages can be handled by having processes keep a message log of all the sent messages
  - Messages E, F: delayed orphan messages. After resuming execution from their checkpoints, processes will generate both of these messages