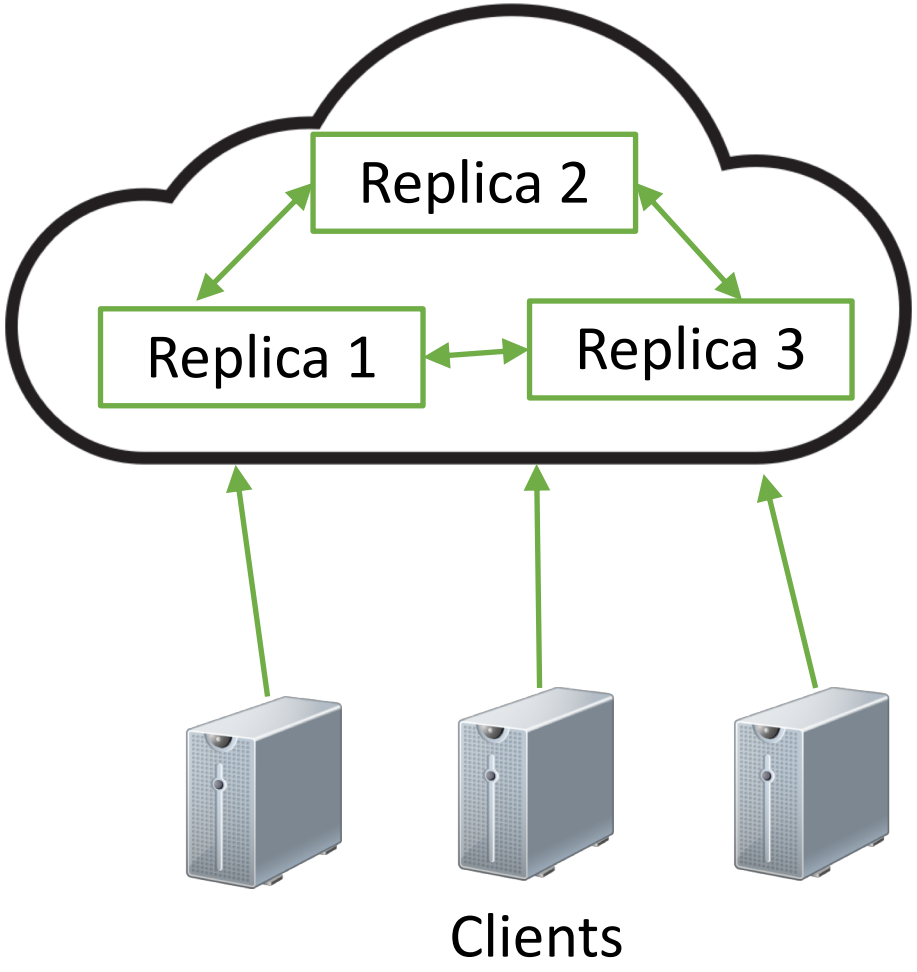
Week 2
Spyros Mastorakis

Outline

- Clocks
- Snapshots & Checkpointing

Why we need clocks?

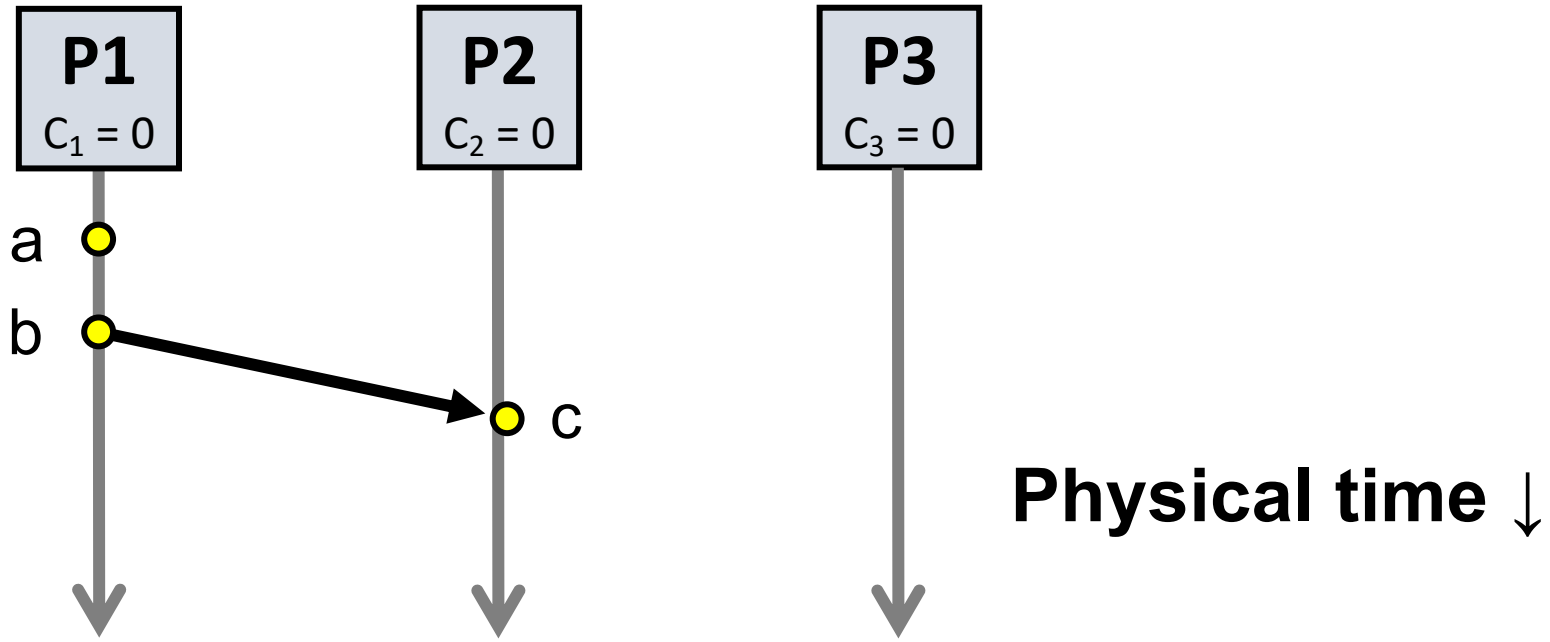


- Multiple replicas running a service
  - Give ”illusion” of consistent state to clients
- All replicas need to apply updates in the same order
  - Replicated state machine
- Need to come up with a relative order of events

Logical Clocks

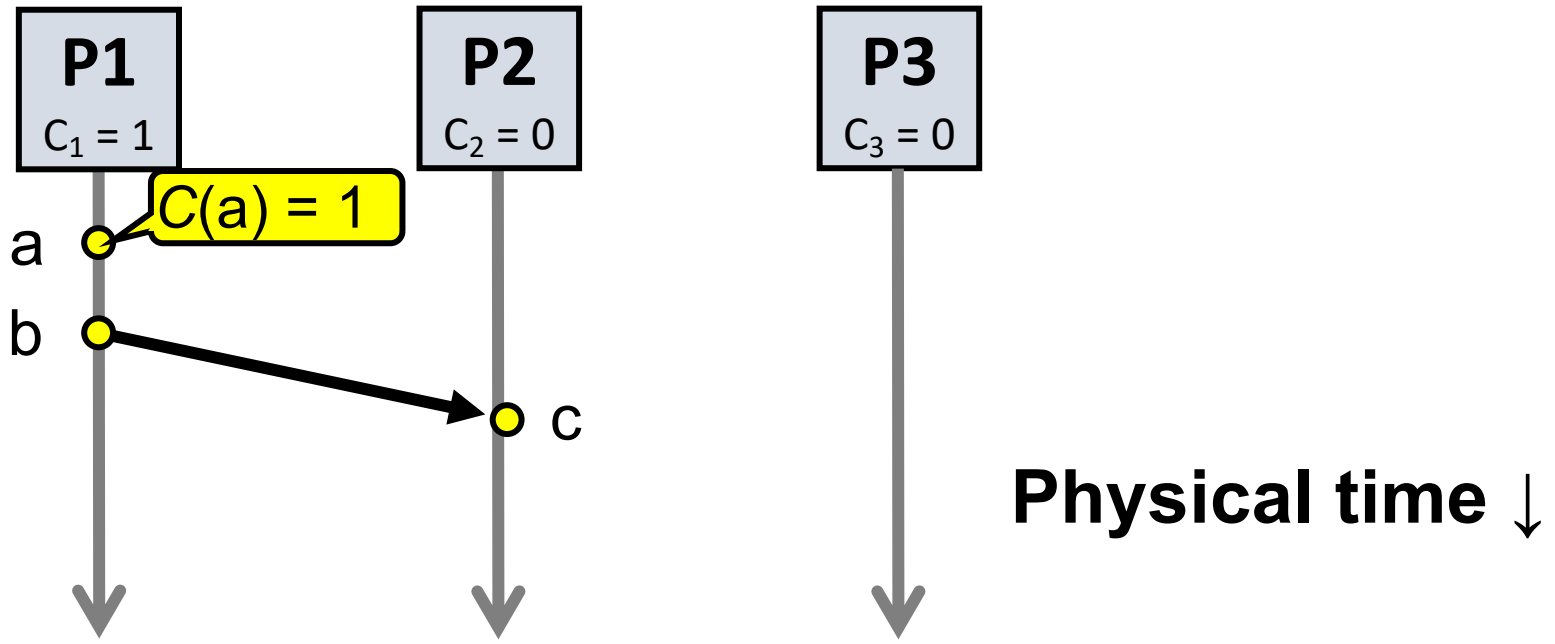
- Associate any event a with a clock time C(a)
- Clock condition: if a → b, then C(a) < C(b)
- Order events at all nodes based on clock time

The Lamport Clock algorithm



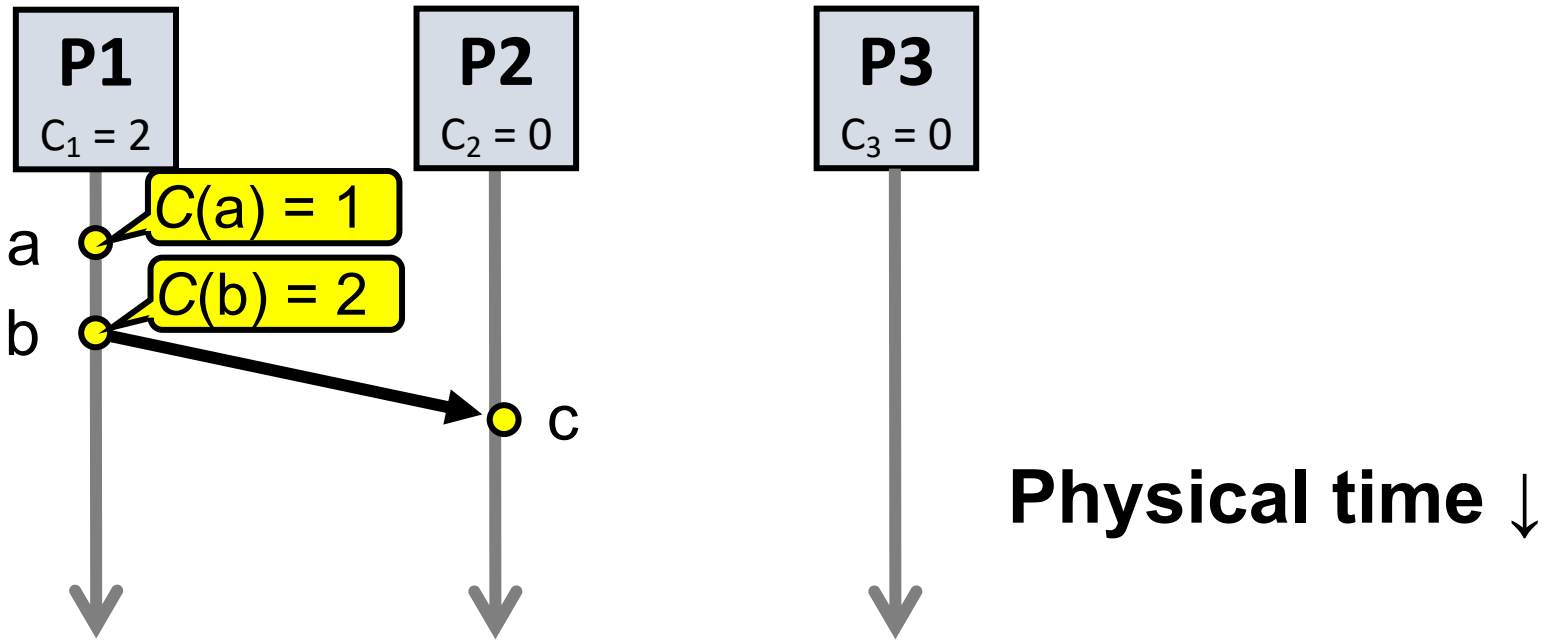
- Each process P_i maintains a local clock C_i
- All process clocks start at time 0

The Lamport Clock algorithm



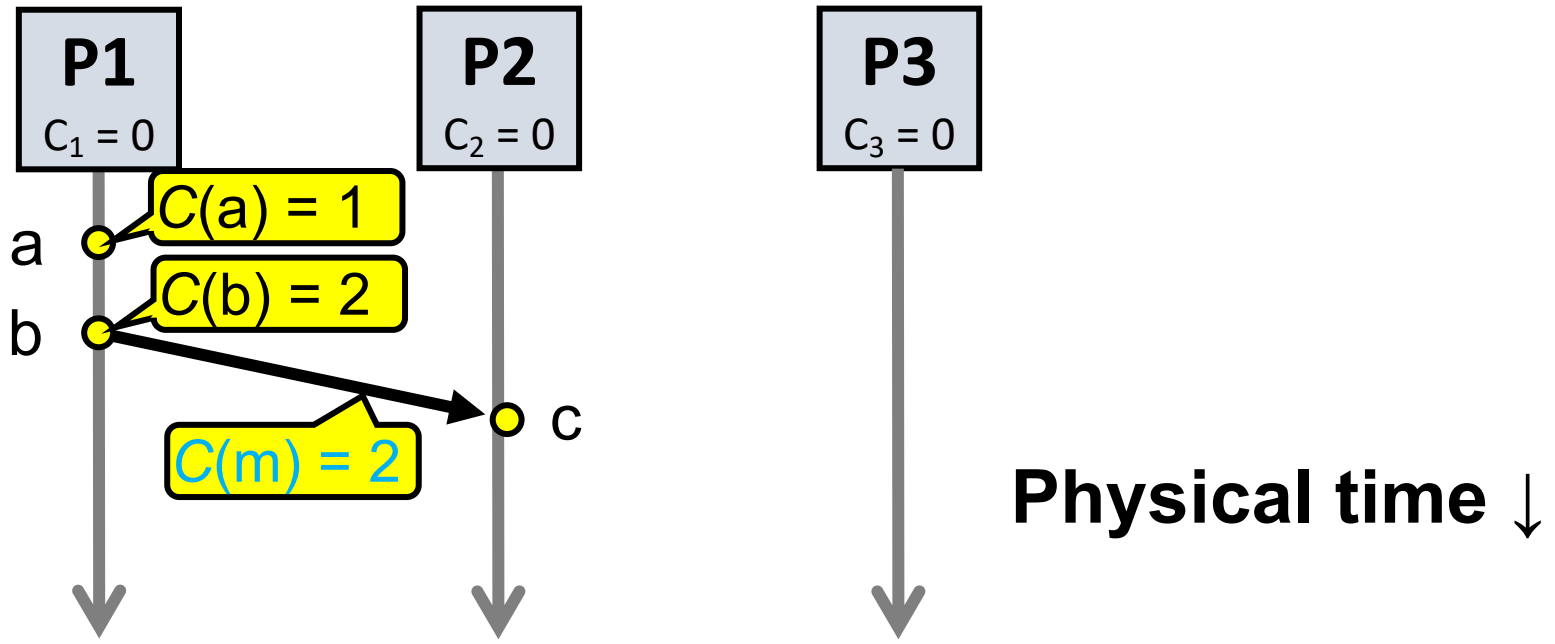
1. Before executing an event, $C_i \leftarrow C_i + 1$

The Lamport Clock algorithm



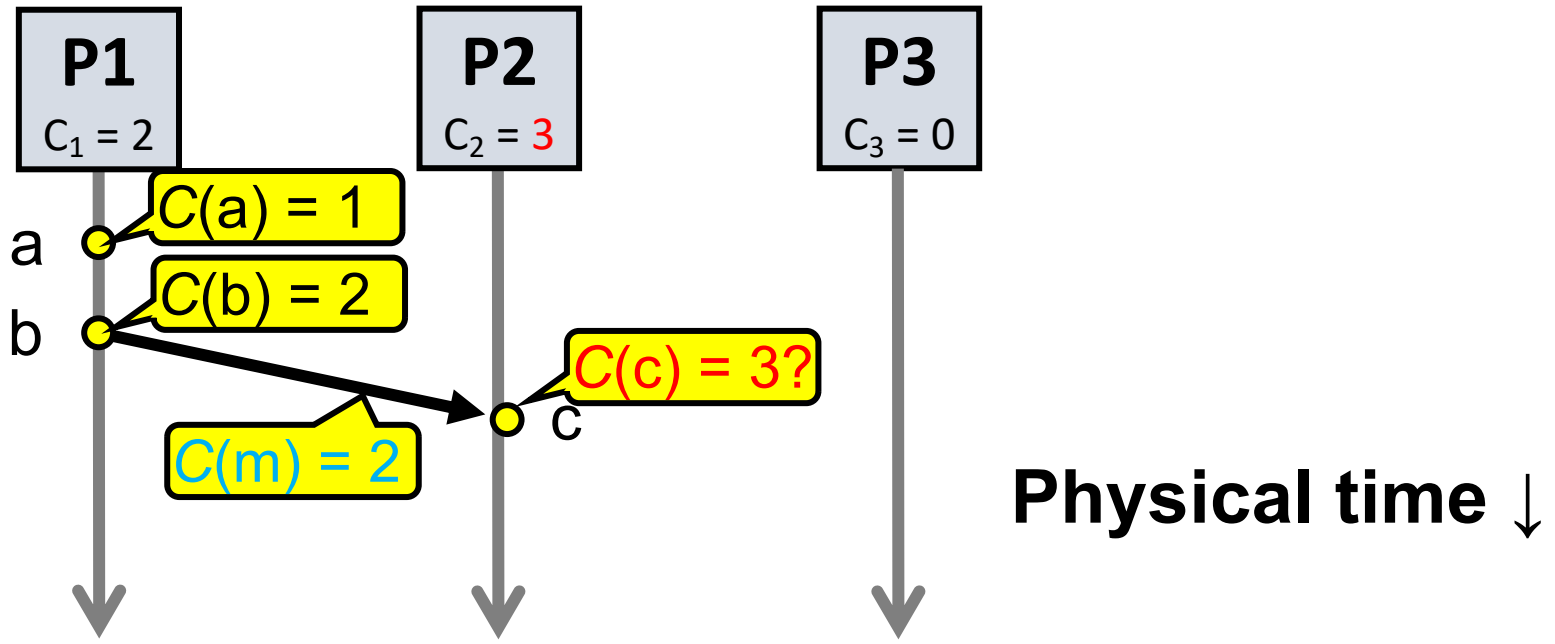
1. Before executing an event, $C_i \leftarrow C_i + 1$

The Lamport Clock algorithm



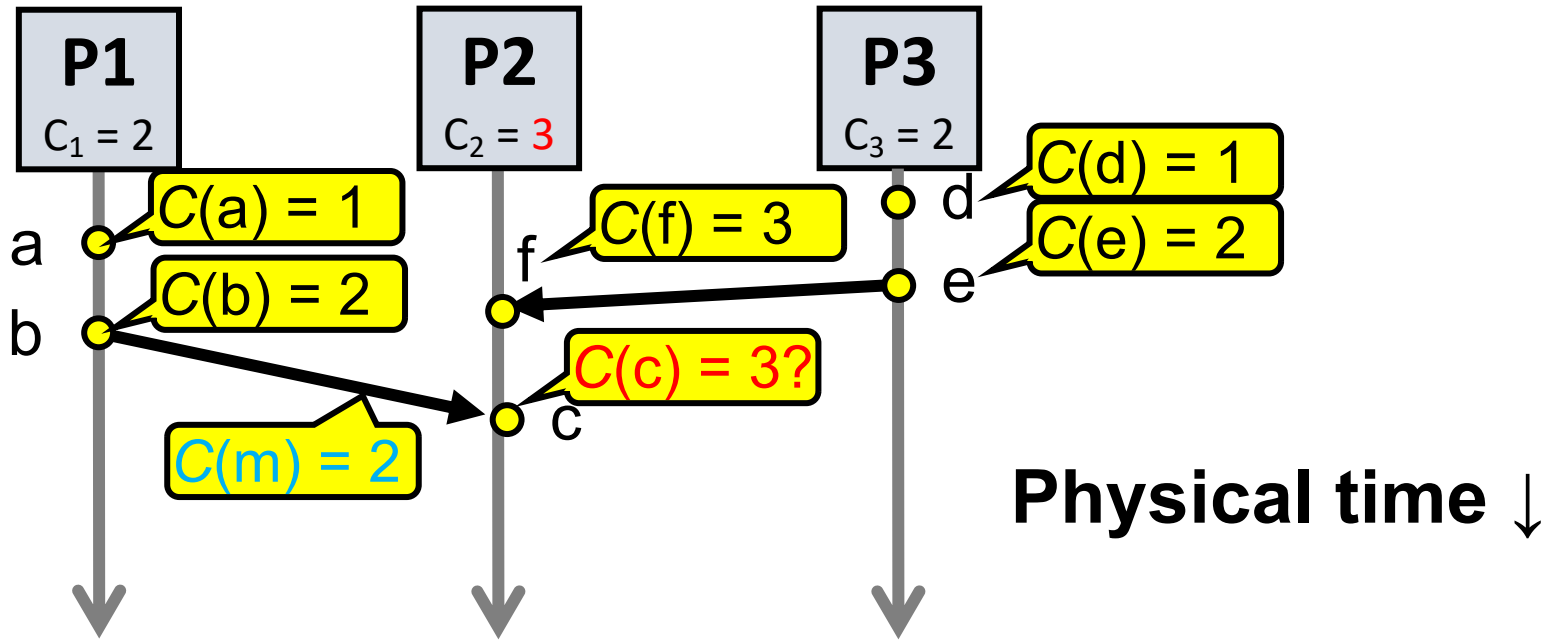
1. Before executing an event, $C_i \leftarrow C_i + 1$
2. Send the local clock in the message m

The Lamport Clock algorithm



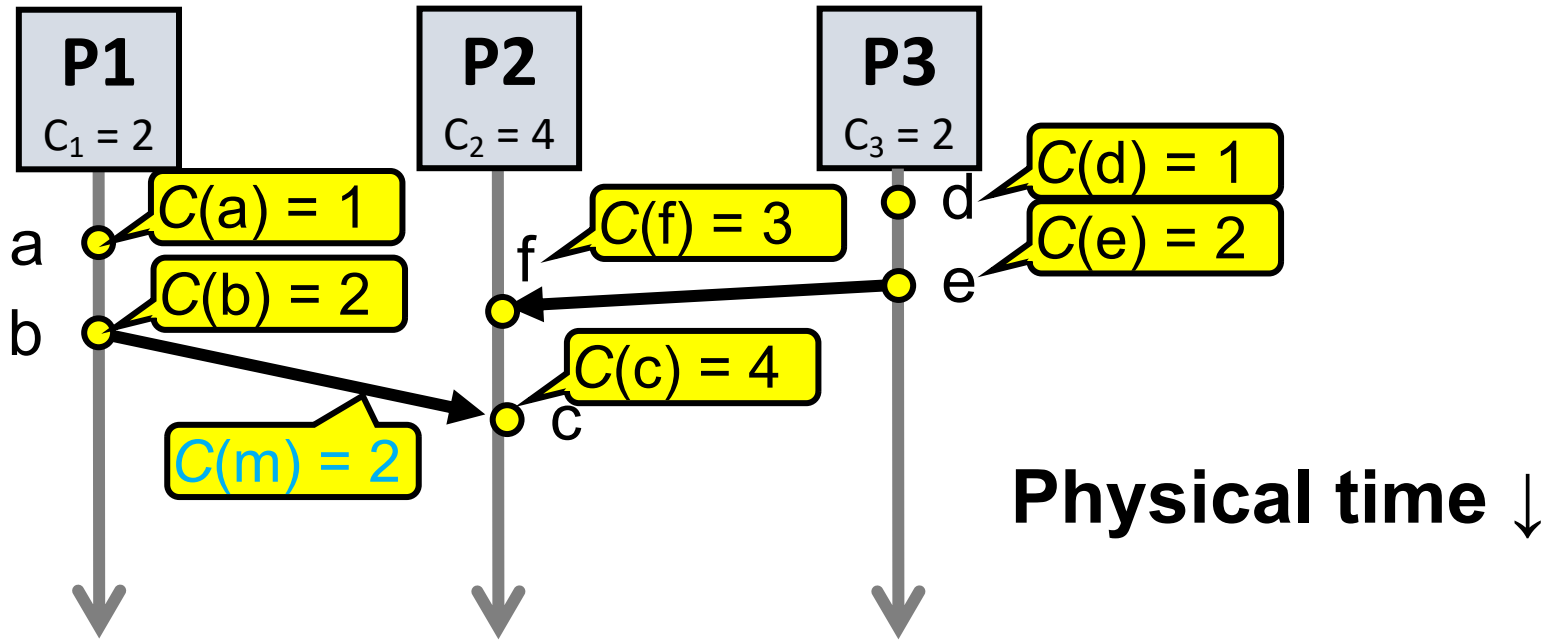
1. Before executing an event, $C_i \leftarrow C_i + 1$
2. Send the local clock in the message m
3. On process P_j receiving message m: what time should the receive event (C(m) + 1) use?

The Lamport Clock algorithm



1. Before executing an event, $C_i \leftarrow C_i + 1$
2. Send the local clock in the message m
3. On process P_j receiving message m: what time should the receive event (C(m) + 1) use?

The Lamport Clock algorithm

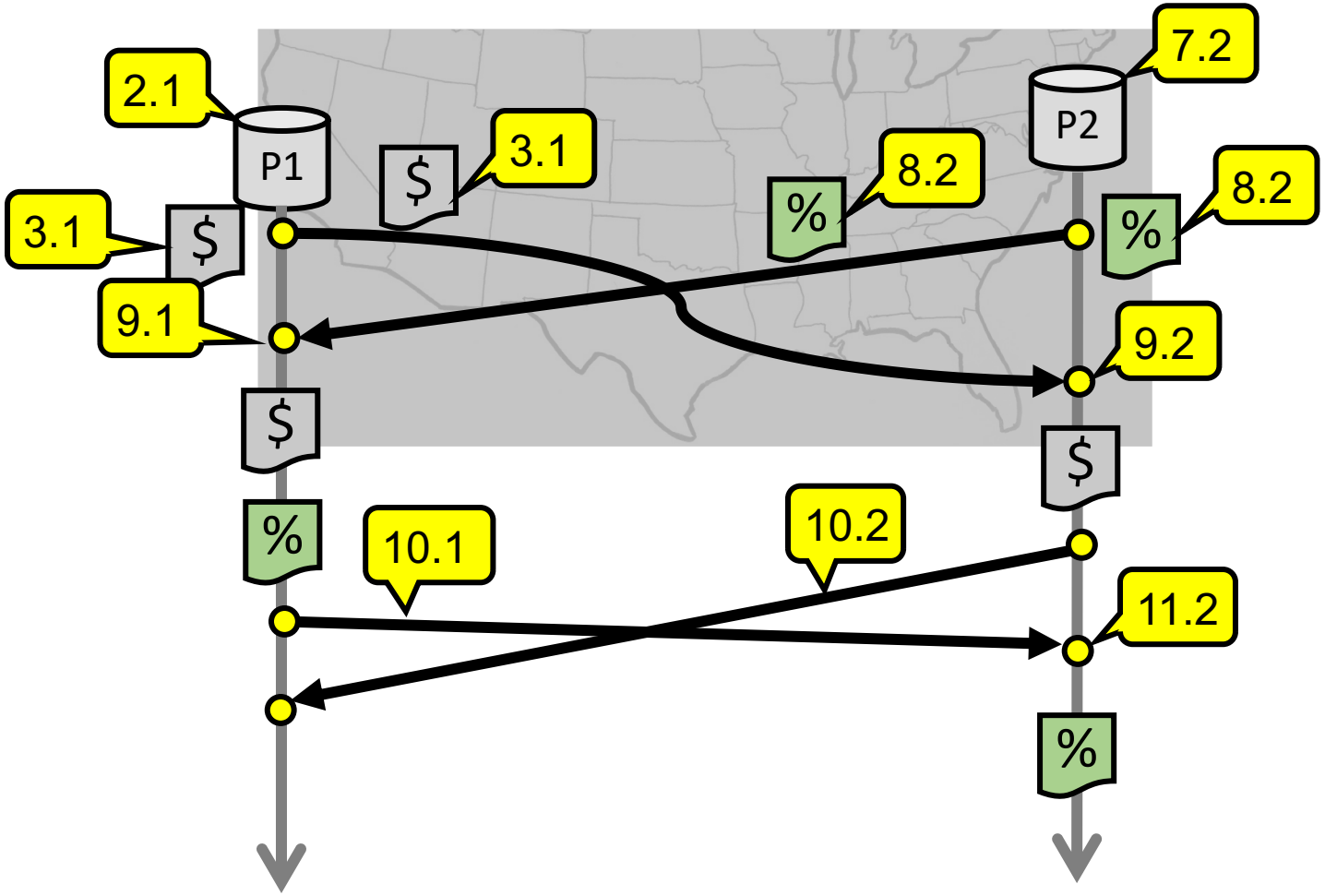


1. Before executing an event, $C_i \leftarrow C_i + 1$
2. Send the local clock in the message m
3. On process P_j receiving message m: set C_j and time of receive event to $1 + \max\{C_j, C(m)\}$

RSMs with Lamport Clocks

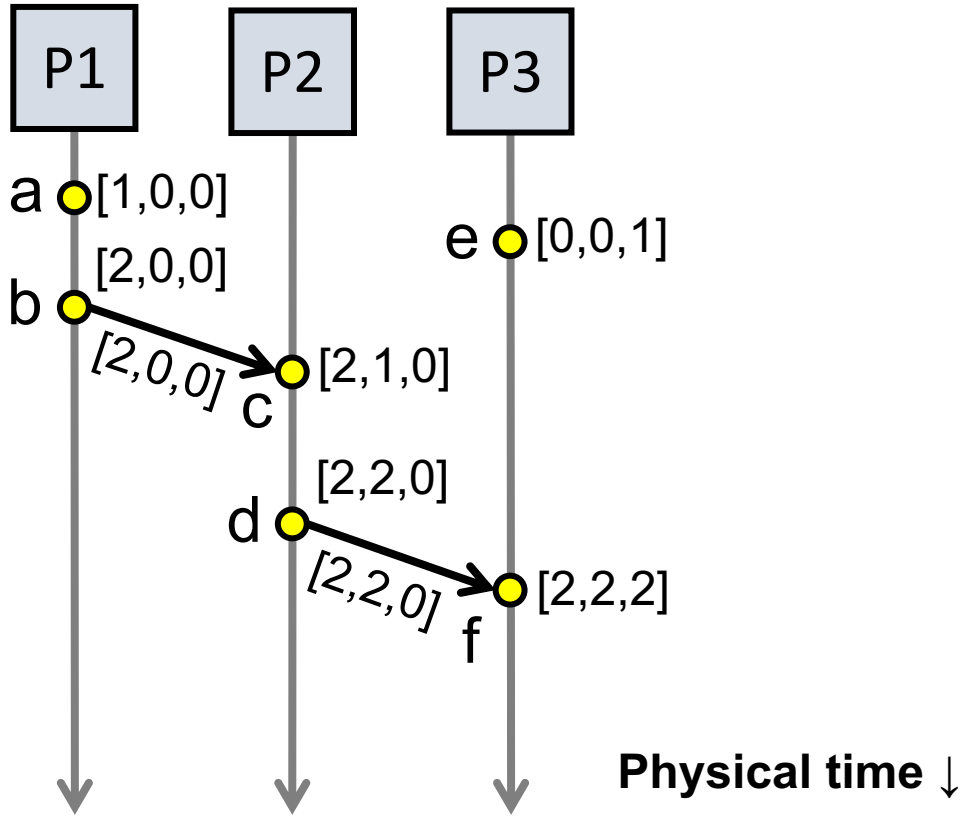
RSMs with Lamport Clocks

Assumption: network guarantees FIFO delivery



Vector clocks (VC)

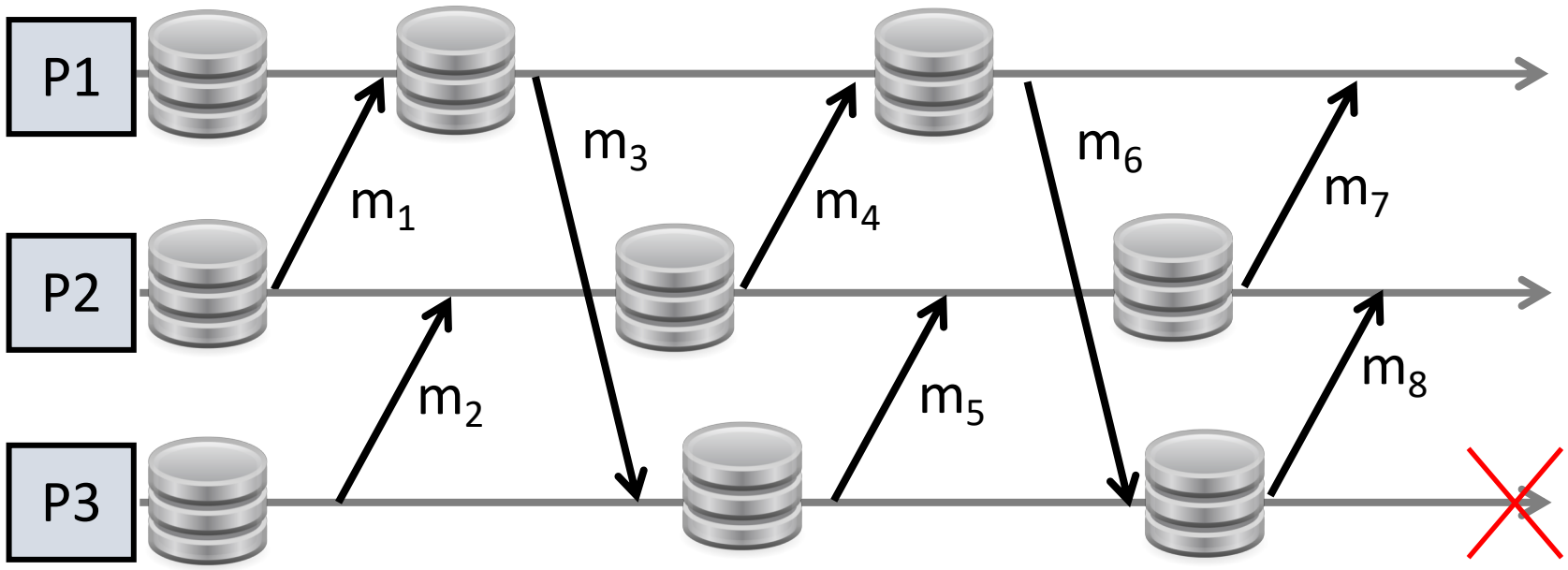
- Label each event e with a vector, V(e) = [c_1, c_2,…, c_n]
  - Number of components = number of processes



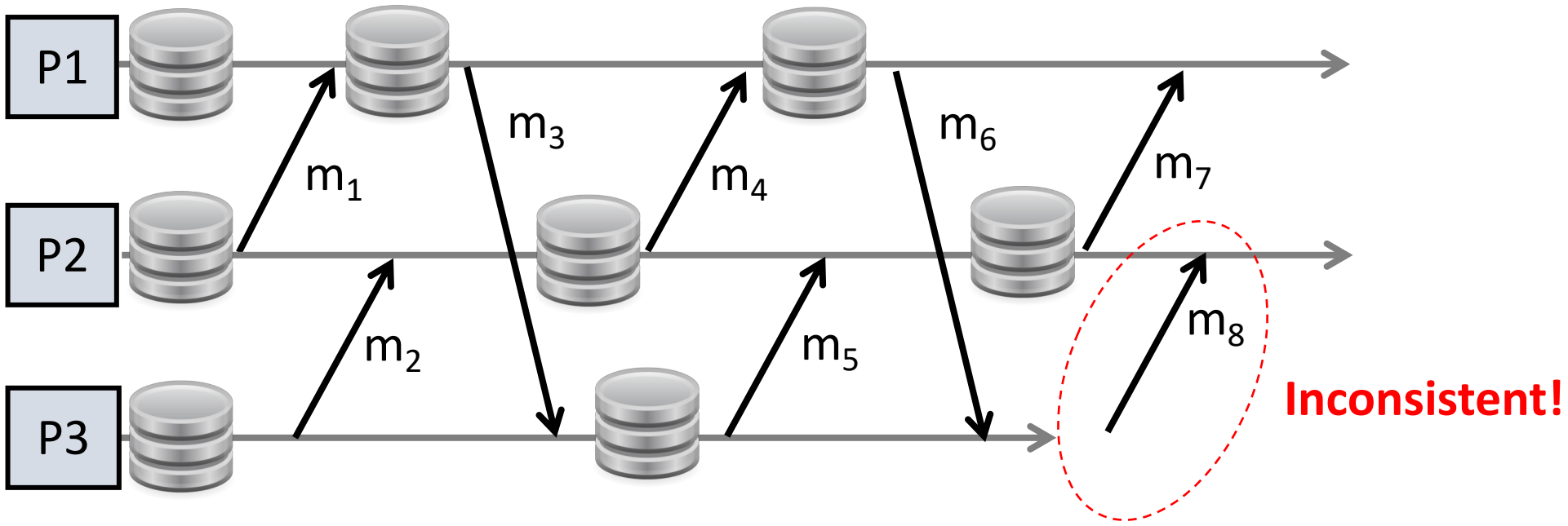
Replicated State Machines with Logical Clocks

- Any replica can execute an update only after confirming the clock on all other replicas is higher than the update's clock
- Implication: if any one replica is down, all other replicas are unable to make progress
  - Our RSM implementation (with logical clocks) only gives us a way to apply updates at replicas
- We assume crash failures: replicas can resume execution with saved state

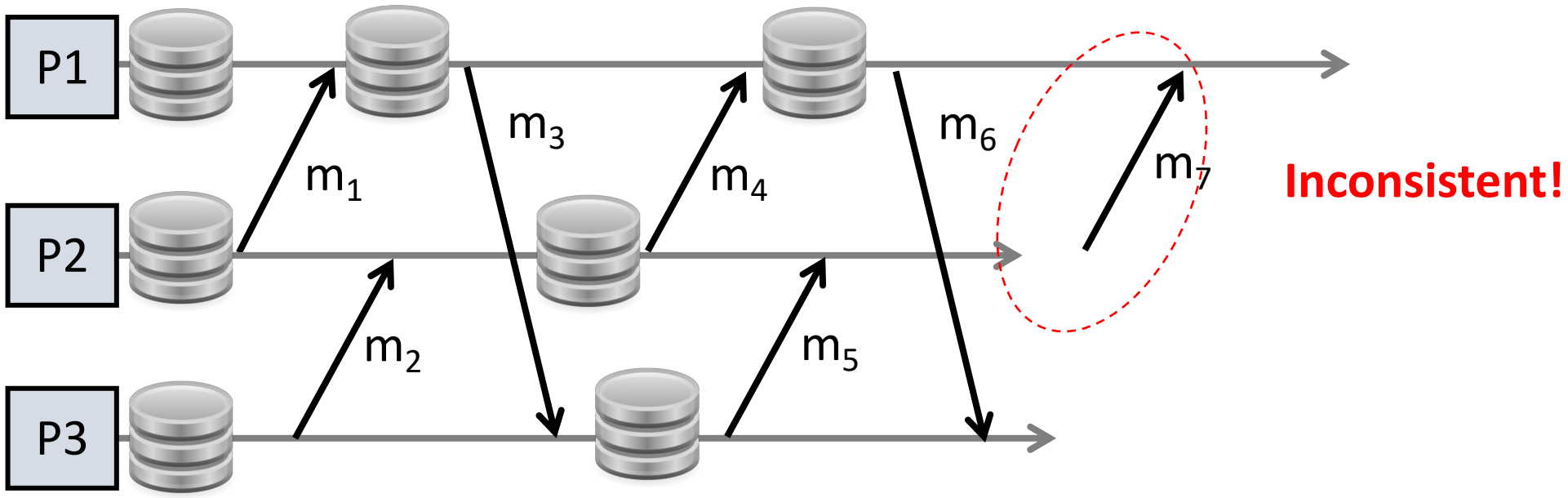
Challenge in checkpointing



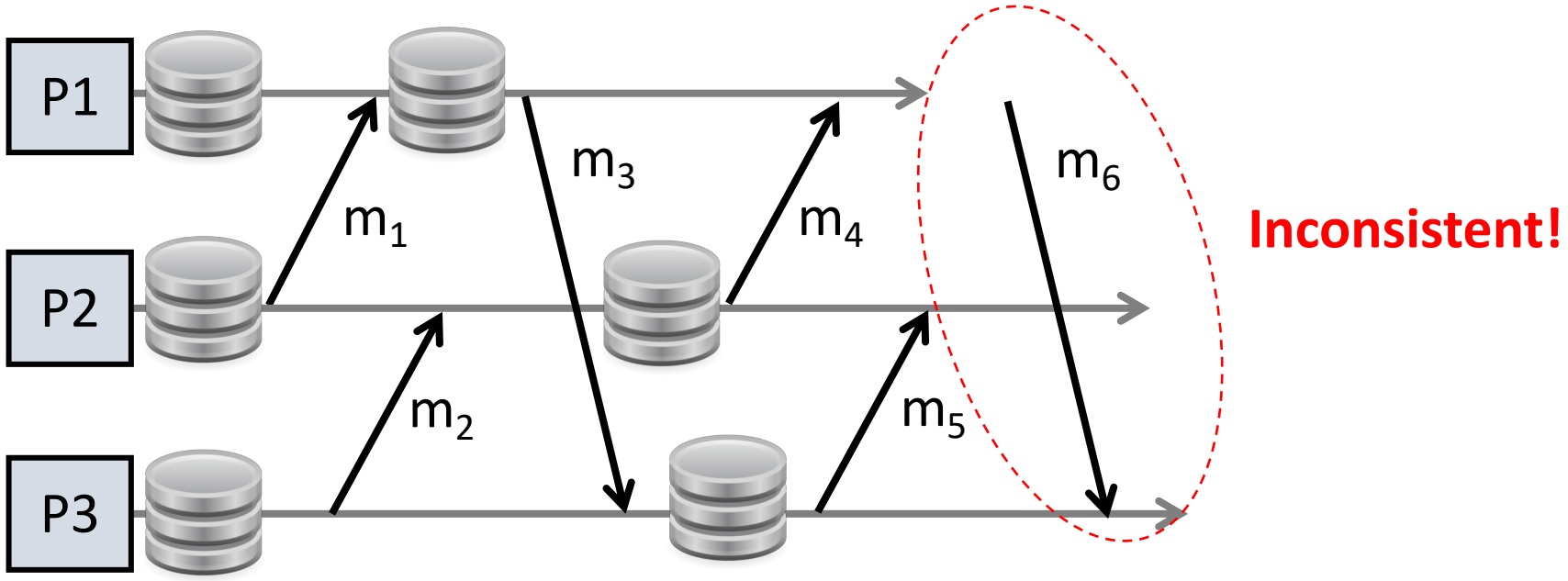
Challenge in checkpointing



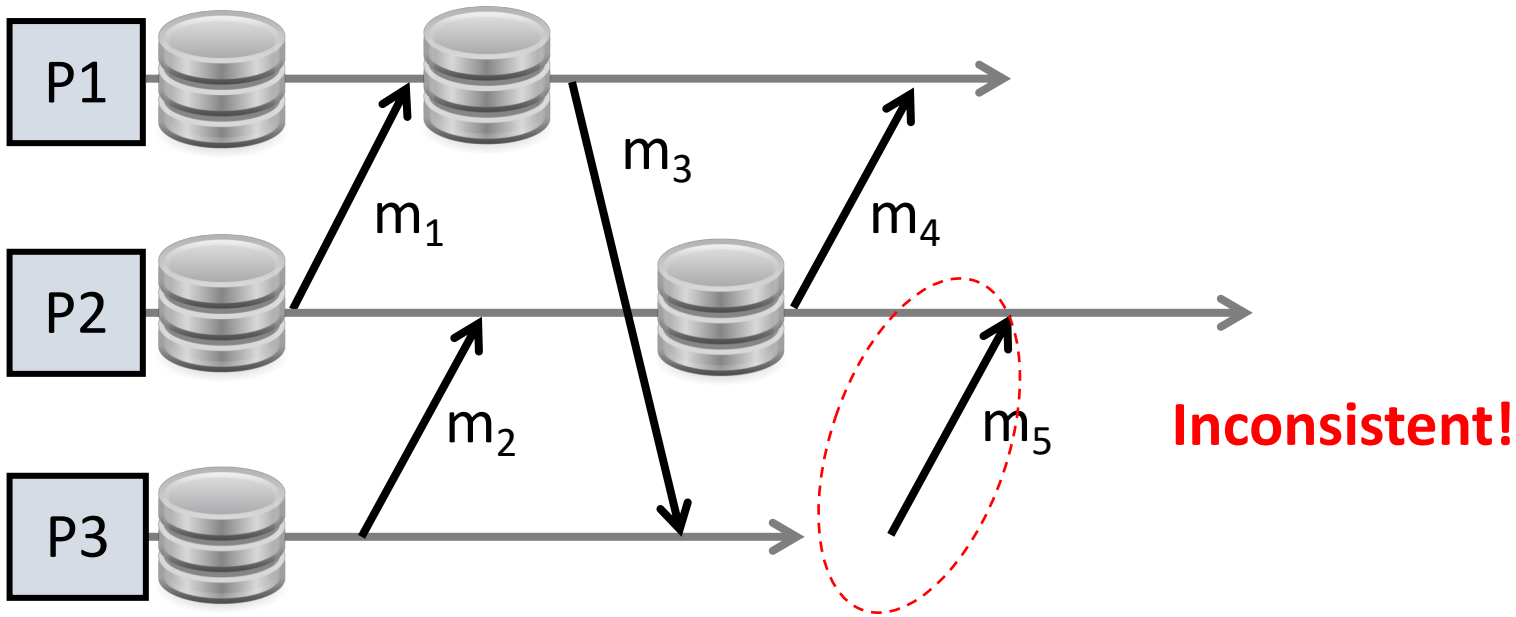
Challenge in checkpointing



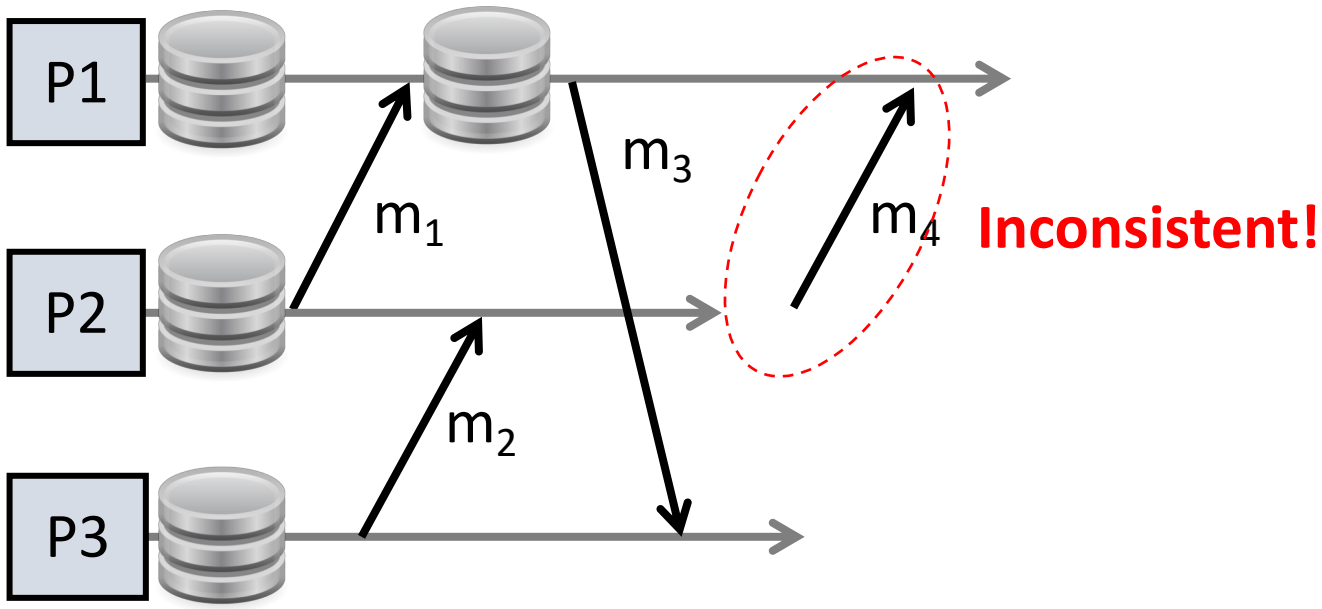
Challenge in checkpointing



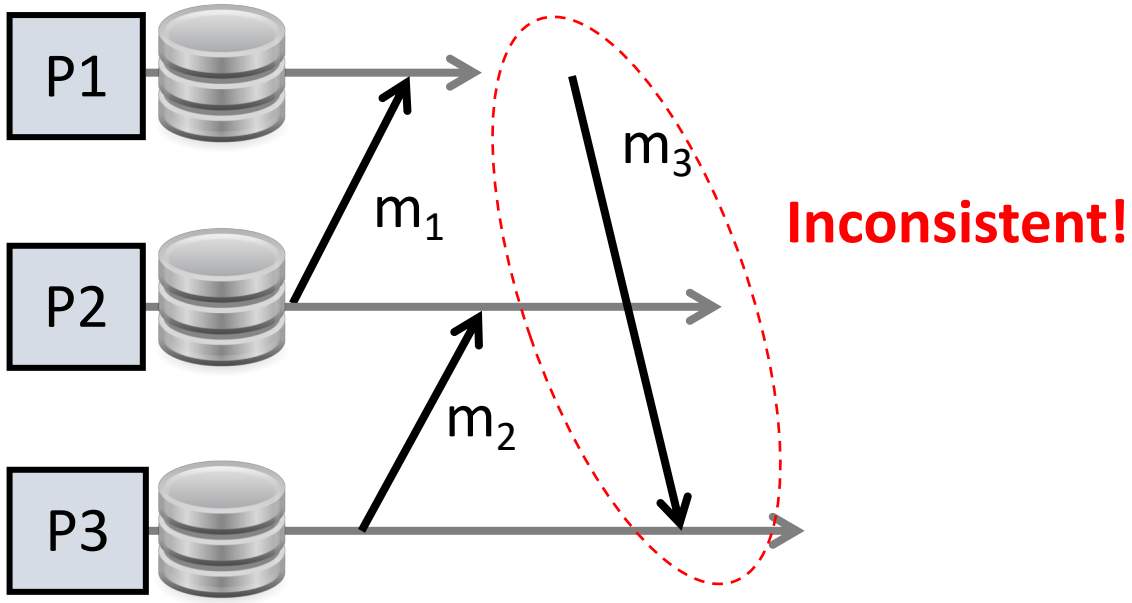
Challenge in checkpointing



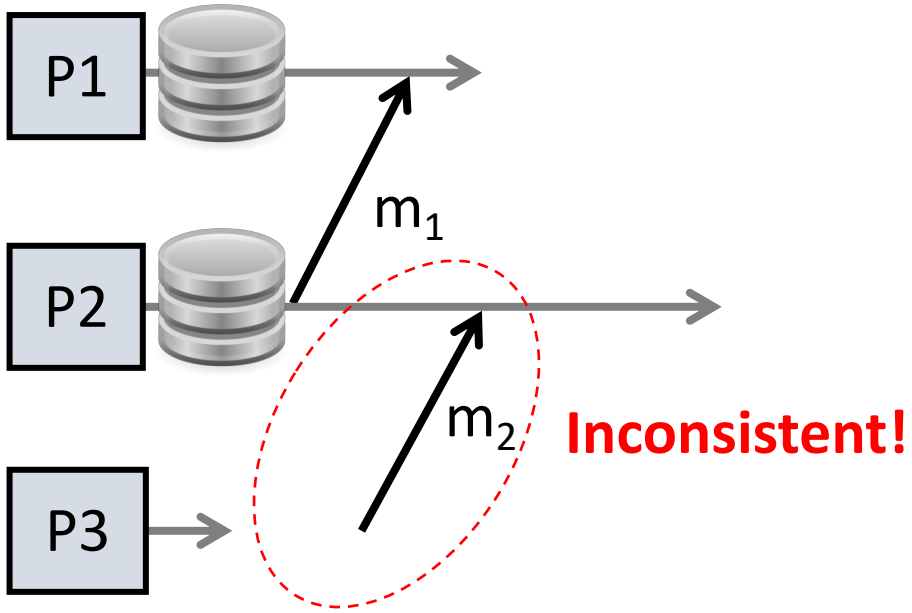
Challenge in checkpointing



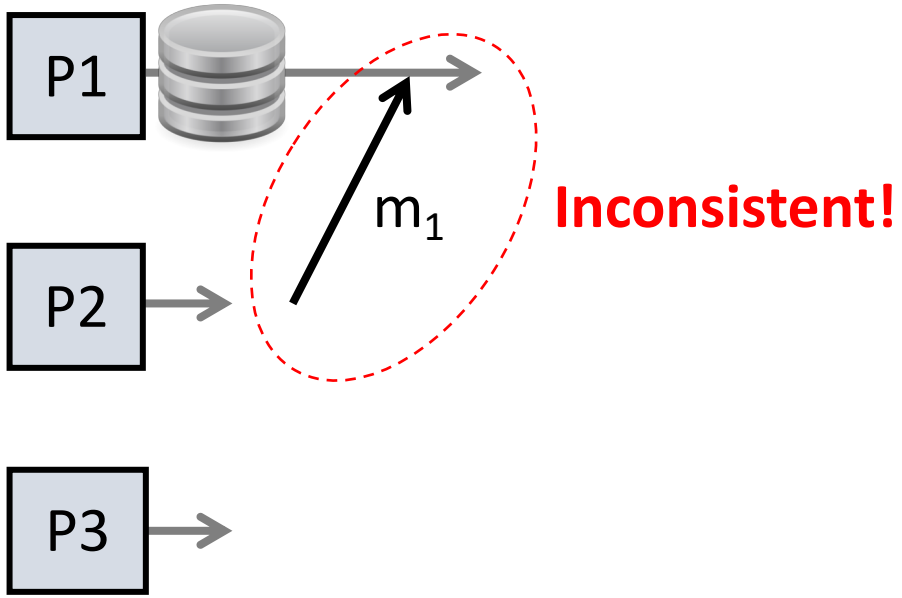
Challenge in checkpointing



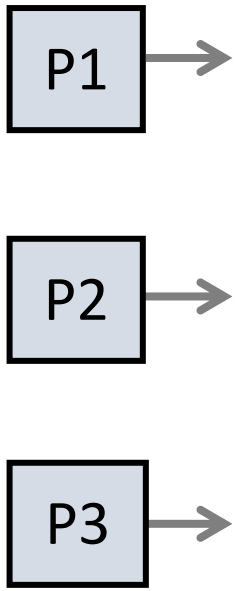
Challenge in checkpointing



Challenge in checkpointing



Challenge in checkpointing

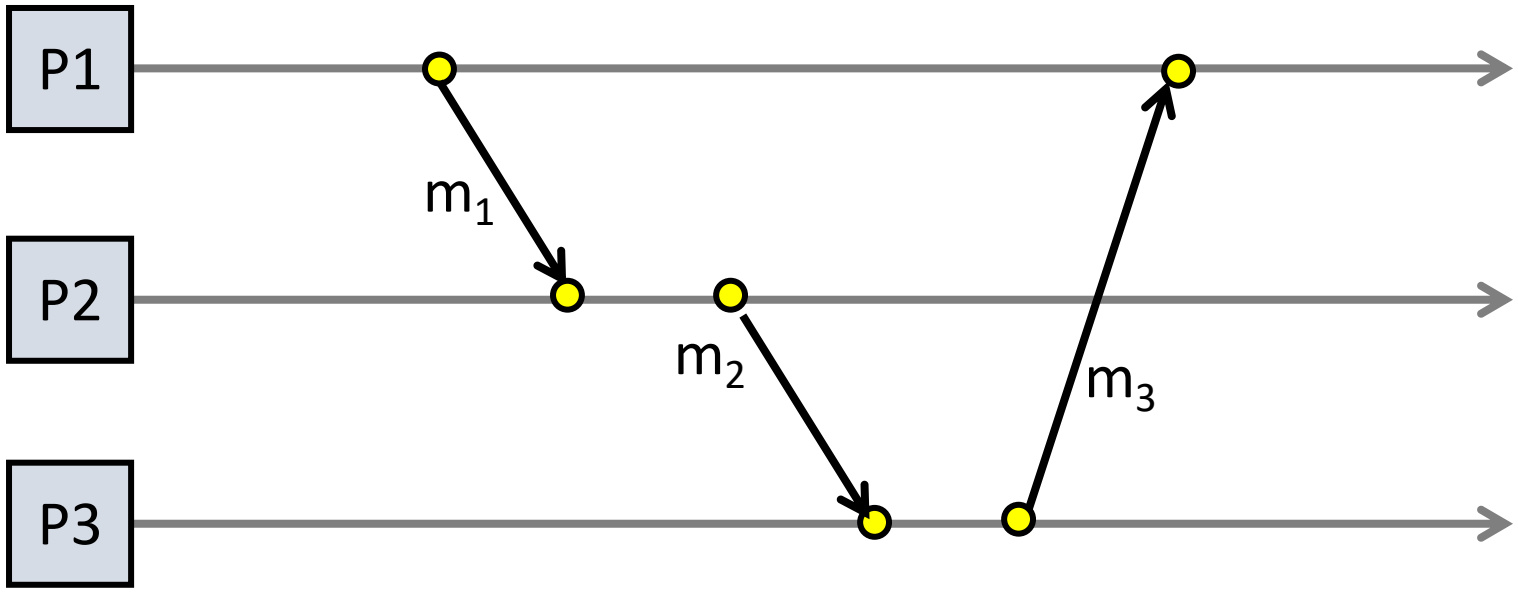


Domino effect!

Chandy-Lamport algorithm

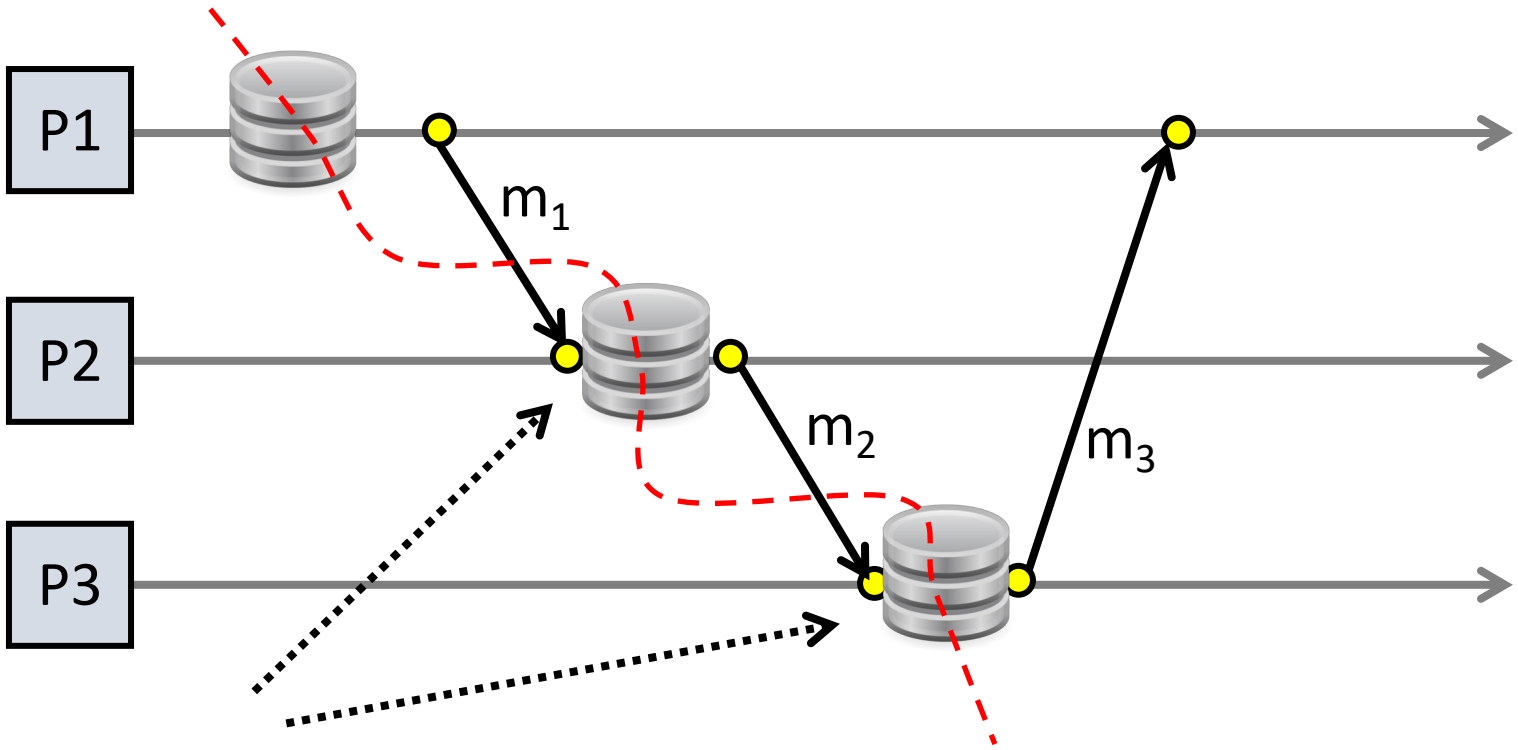
- Take a snapshot of a distributed system
- N processes in the system
  - Processes don’t fail while taking snapshot
  - Any process may initiate collection of snapshot
- One unidirectional channel/network in either direction between each pair of processes
  - All channels ensure FIFO delivery
  - Lossless and no duplication

Example: token ring



Rotate lock possession across processes in circle/ring order

Example: token ring (everybody has a token!)

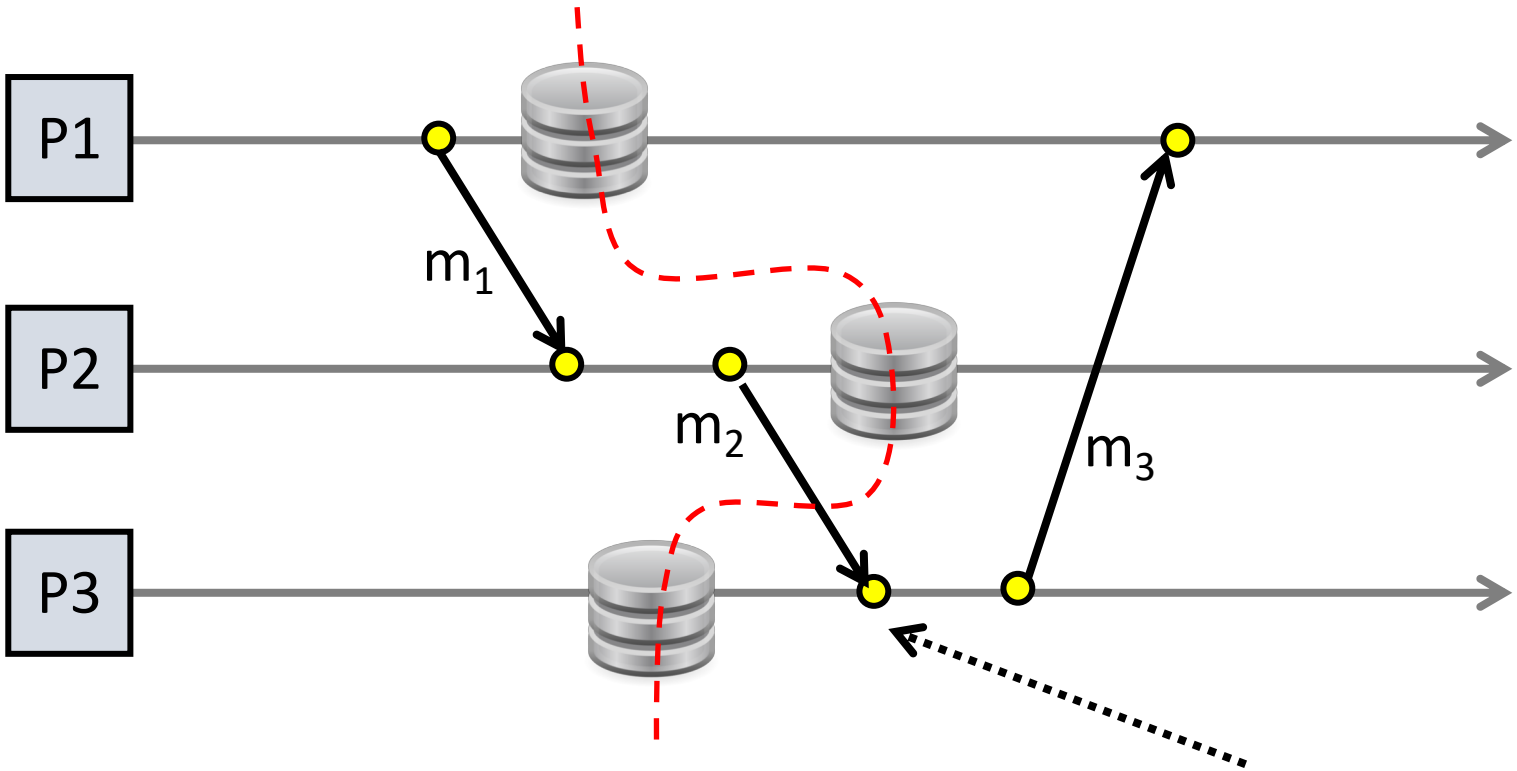


Global Snapshot

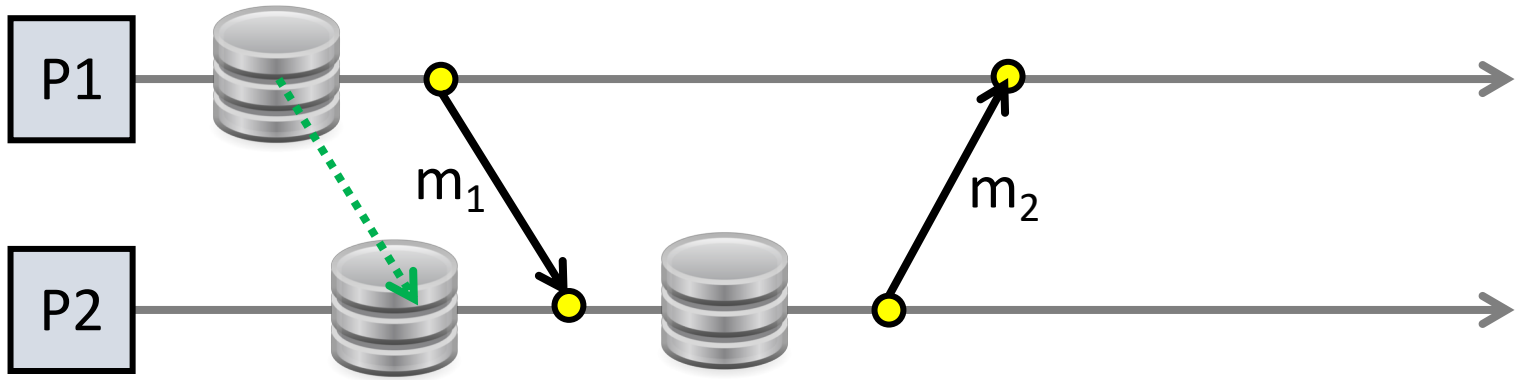
- Captured state must satisfy “happens before” principle
  - If event b is in the snapshot and a → b, then event a must also be in the snapshot

- ***Main challenge***: If a snapshot includes message receipt, it should also include sending it

Example: token ring (desired snapshot)



Chandy-Lamport snapshot



Solution: after taking checkpoint, send message to others informing them to also take checkpoints

- This message will arrive before any subsequent ones (FIFO)