`

AMRITA
VISHWA VIDYAPEETHAM

AHEAD Online

Distributed Systems

Dr. Swaminathan J
Department of Computer Science
Amrita Vishwa Vidyapeetham

Objectives

- To introduce the concept global snapshot, the conditions for consistency.

- To discuss algorithms for taking snapshots in the case of
    - FIFO (Chandy-Lamport)
    - Async (Lai-Yang)
    - CO (Acharya-Badrinath)

Place your
Webcam Video here
Size 100%

What is a global snapshot?

Place your
Webcam Video here
Size 38%

- The state of each process + state of each channel at an instant makes the global snapshot of the DS.
- Consider a 4-Process distributed system.

Local States

- LS_1
- LS_2
- LS_3
- LS_4

Channel States

- SC_{12}
- SC_{21}
- SC_{13}
- SC_{31}
- SC_{14}
- SC_{41}
- SC_{23}
- SC_{32}
- SC_{24}
- SC_{42}
- SC_{34}
- SC_{43}

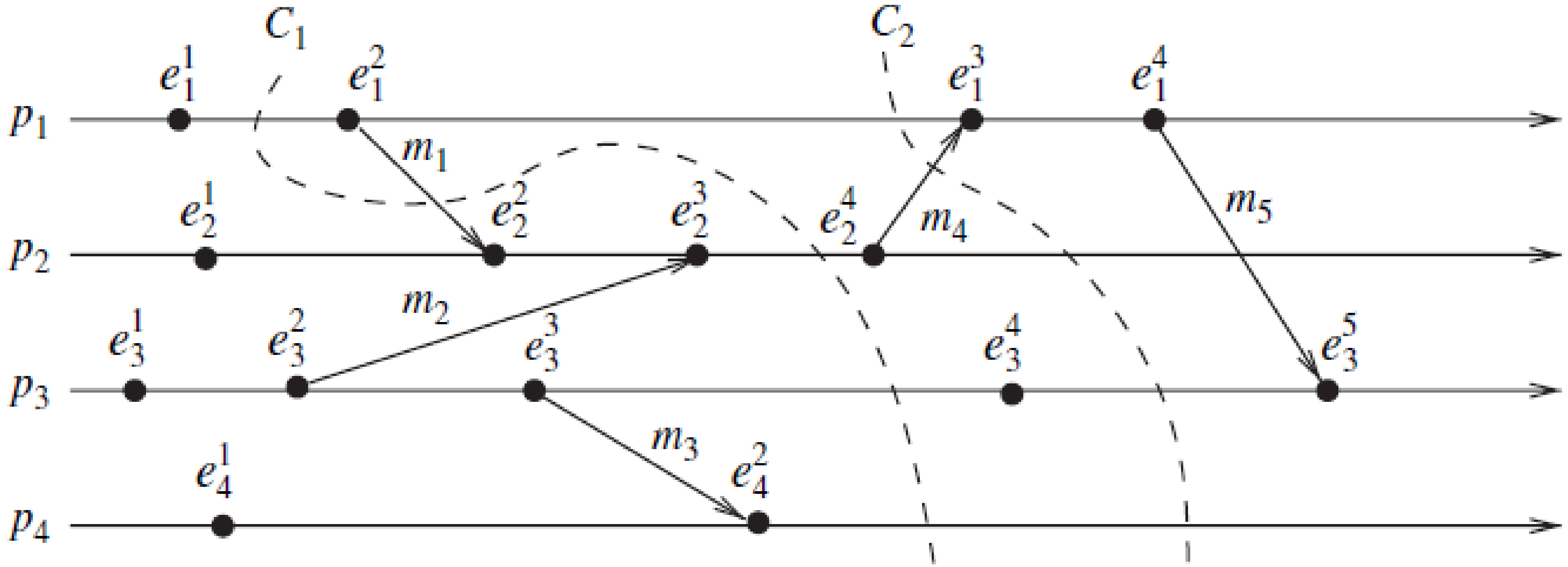
$$\text{GS} = \{ \cup_i LS_i \ , \cup_{ij} SC_{ij} \}$$

Issues with taking snapshot

Place your
Webcam Video here
Size 38%

- You cannot take a global snapshot in an instant.
- You may end up in snapshot with anomalies.

C2 is fine



C1 does not have the record of sending m1 but has the record of receiving m1.

Consistent Global Snapshot

Place your
Webcam Video here
Size 38%

- Let's define what is consistent global snapshot.

(1) $send(m_{ij}) \in LS_i \Longrightarrow m_{ij} \in SC_{ij} \oplus recv(m_{ij}) \in LS_j$

(2) $send(m_{ij}) \notin LS_i \Longrightarrow m_{ij} \notin SC_{ij} \wedge recv(m_{ij}) \notin LS_j$

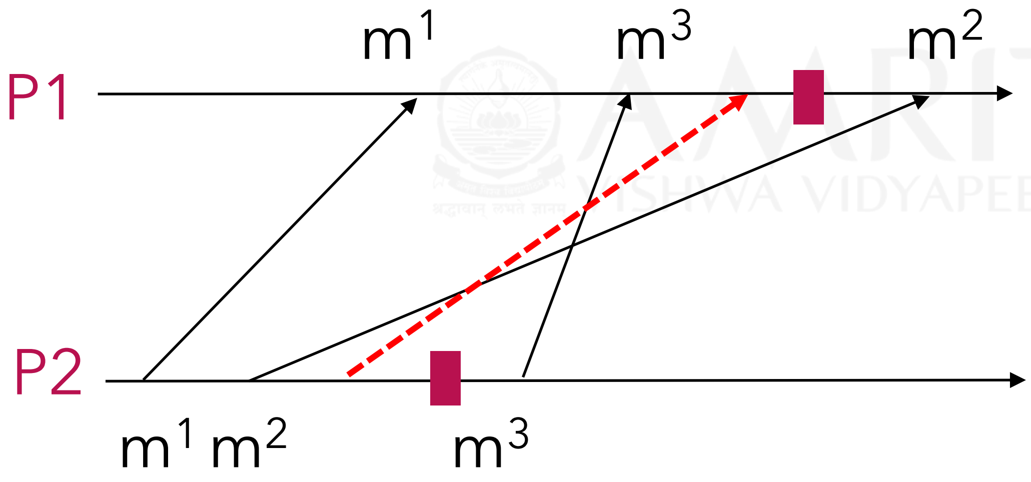
Why would inconsistency happen?

Place your
Webcam Video here
Size 38%

- Taking snapshot requires (control) message passing and is usually a multi-step coordination process.

Why removing it from LS_1 is not a solution?

Consider Async (Non-FIFO) channel with 2 processes.



$LS_1 = \{ recv(m^1), recv(m^3) \}$

$LS_2 = \{ send(m^1), send(m^2) \}$

$SC_{12} = \{ \}$

$SC_{21} = \{ m^2 \}$

Channel snapshots can be constructed by sending local snapshots to a single process.

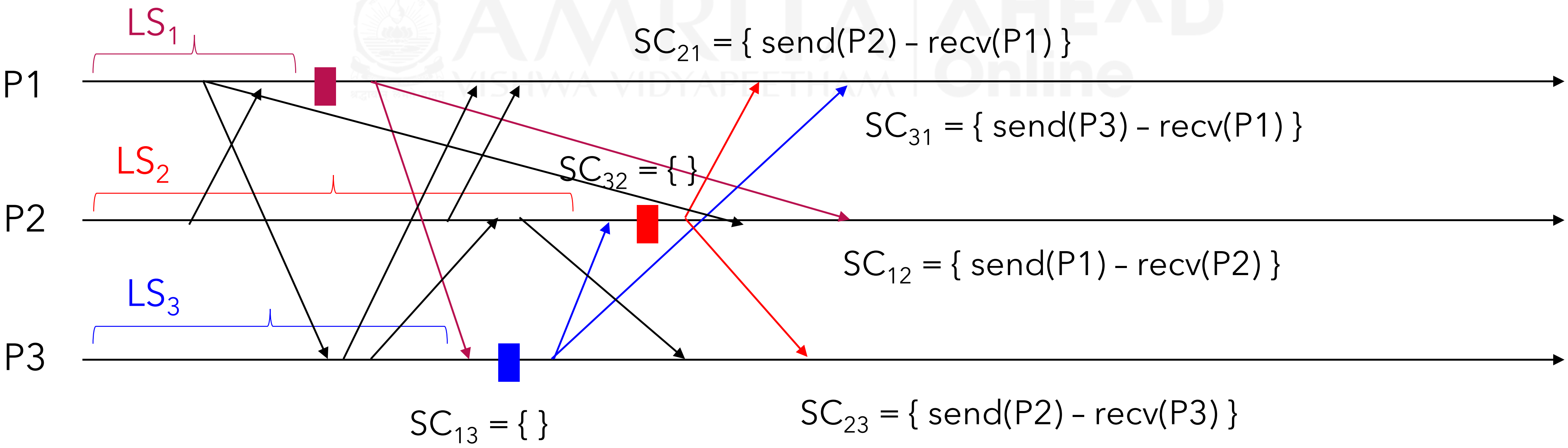
(1) $send(m_{ij}) \in LS_i \Longrightarrow m_{ij} \in SC_{ij} \oplus recv(m_{ij}) \in LS_j$

(2) $send(m_{ij}) \notin LS_i \Longrightarrow m_{ij} \notin SC_{ij} \wedge recv(m_{ij}) \notin LS_j$

Taking snapshot on FIFO ordered system

Place your
Webcam Video here
Size 38%

- Chandy-Lamport Algorithm
  - P1 takes a snapshot and sends a marker along all its channels (C_{12}, C_{13})
  - P3 gets marker, takes a snapshot, sends marker along C_{31}, C_{32}.
  - P2 gets marker from P3 first, takes snapshot, sends marker along C_{31}, C_{32}. (Note: we only claim this is a FIFO ordered not Causal ordered system)

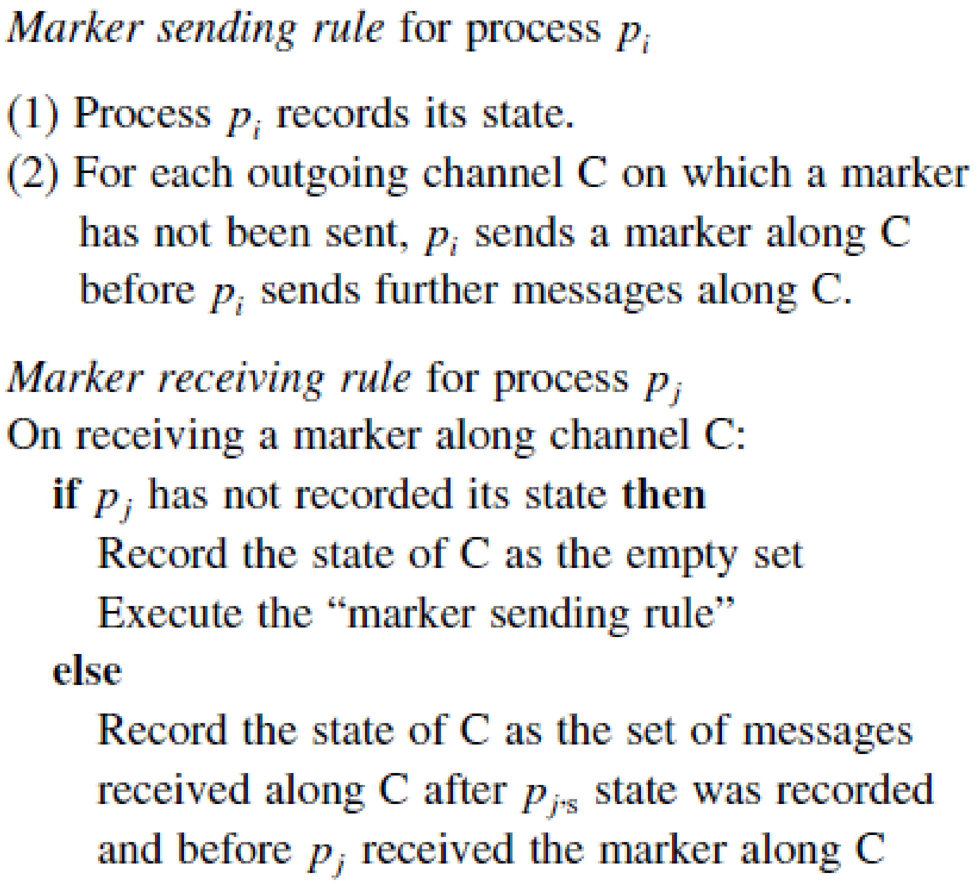


LS_1
LS_2
LS_3
P1
P2
P3
SC_{21} = { send(P2) – recv(P1) }
SC_{31} = { send(P3) – recv(P1) }
SC_{32} = { }
SC_{12} = { send(P1) – recv(P2) }
SC_{13} = { }
SC_{23} = { send(P2) – recv(P3) }

Chandy-Lamport Algorithm

Place your
Webcam Video here
Size 38%

- The algorithm from Mukesh & Ajay (Page 94)



```
*Marker sending rule* for process $p_i$

(1) Process $p_i$ records its state.
(2) For each outgoing channel C on which a marker
    has not been sent, $p_i$ sends a marker along C
    before $p_i$ sends further messages along C.

*Marker receiving rule* for process $p_j$
On receiving a marker along channel C:
  if $p_j$ has not recorded its state then
     Record the state of C as the empty set
     Execute the "marker sending rule"
  else
     Record the state of C as the set of messages
     received along C after $p_{j's}$ state was recorded
     and before $p_j$ received the marker along C
```

Which process consolidates the global snapshot and how?

1. All process can send their local snapshot to one process – initiator or leader – which can do it.
2. Each process can send the local snapshot along with the marker. So, eventually, every process can do it.

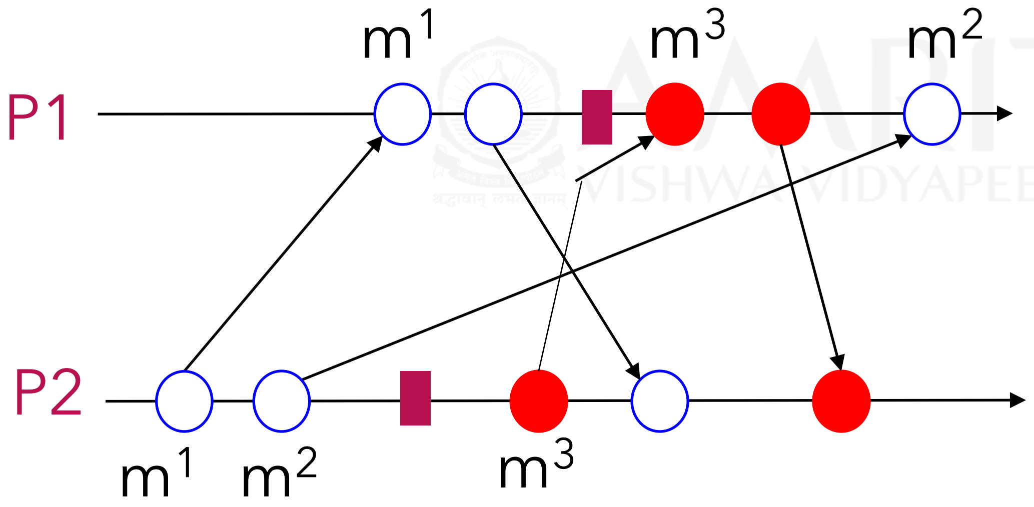
What if two processes initiate snapshot recording concurrently?

- Distinguish based on the marker.
- For each marker, take a snapshot.

Taking snapshot on Async ordered system

Place your
Webcam Video here
Size 38%

- Lai-Yang algorithm
  - Uses white and red color messages. White before snapshot & red after.
  - When a red message arrives for the first time, it is parked and received after snapshot. This crucial step ensures consistency is maintained.



$LS_1 = \{ recv(m^1) \}$

$LS_2 = \{ send(m^1), send(m^2) \}$

$SC_{12} = \{ \}$

$SC_{21} = \{ m^2 \}$

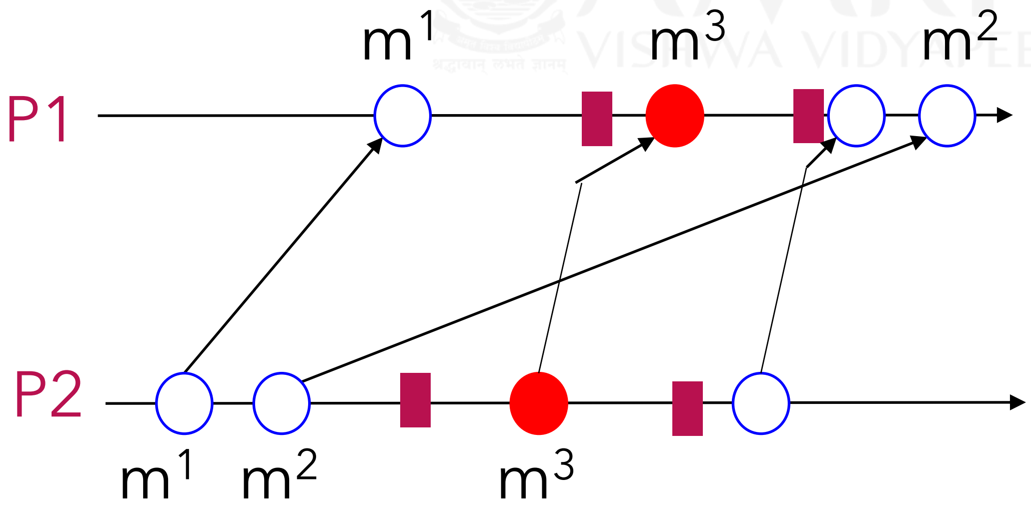
(1) $send(m_{ij}) \in LS_i \Longrightarrow m_{ij} \in SC_{ij} \oplus recv(m_{ij}) \in LS_j$

(2) $send(m_{ij}) \notin LS_i \Longrightarrow m_{ij} \notin SC_{ij} \wedge recv(m_{ij}) \notin LS_j$

Properties of Lai-Yang Algorithm

Place your
Webcam Video here
Size 38%

- No marker is needed. The message colour is enough.
- It is a single process initiator algorithm. One designated initiator process always takes the snapshot.
- Colour needs to be toggled after every snapshot.
  - It is little dangerous to initiate snapshot in quick succession.



Taking snapshot on CO system

- Acharya-Badrinath algorithm
- It gets easier since the system is causally ordered.
- Initiate snapshot at any time and message others.
- Upon incoming snapshot taking request, every process takes the snapshot and informs others.
- Channel message = Sent by source – received by destination process.
- Please look into the book for the details of the algorithm.

Summary

- We introduce the concept of global snapshots.

- We discussed two algorithms.
- Chandy-Lamport algorithm for FIFO ordered
- Lai-Yang algorithms for Async/non-FIFO channel.

Place your
Webcam Video here
Size 100%