RAMAIAH
Institute of Technology

M.S. Ramaiah Institute of Technology
(Autonomous Institute, Affiliated to VTU)
Department of Computer Science and Engineering

Course Name: Distributed Systems
Course Code: CSE20
Credits: 3:0:0:1

Term: September – December 2020

Faculty:
Sini Anna Alex

Global snapshot is global state

- Each distributed application has a number of processes (leaders) running on a number of physical servers.
- These processes communicate with each other via channels (text messaging).
- A **snapshot** captures the local states of each process (e.g., program variables) along with the state of each communication channel.

RAMAIAH
Institute of Technology

Why do we need snapshots?

Check pointing: restart if the application fails

- **Collecting garbage**: remove objects that don’t have any references
- **Detecting deadlocks**: can examine the current application state
- **Other debugging**: a little easier to work with

We could just synchronize clocks

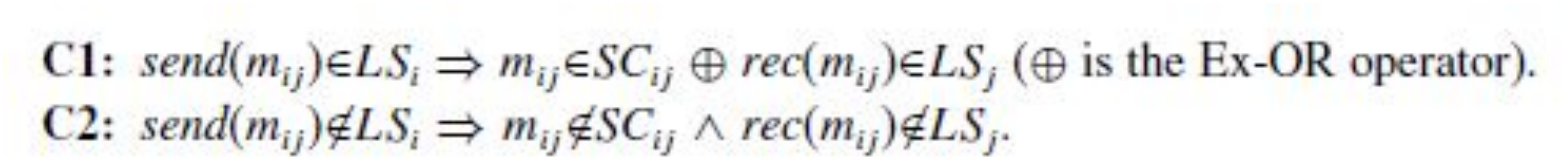
Each process records state at time some agreed upon *t*

– But clocks skew

– And we wouldn’t record messages

- Do we need synchronization?
- What did Lamport realize about ordering events?

Consistent Global state


C1: $send(m_{ij}) \in LS_i \Rightarrow m_{ij} \in SC_{ij} \oplus rec(m_{ij}) \in LS_j$ ($\oplus$ is the Ex-OR operator).

C2: $send(m_{ij}) \notin LS_i \Rightarrow m_{ij} \notin SC_{ij} \wedge rec(m_{ij}) \notin LS_j$.

- Strongly Consistent Global State
- Inconsistent Global State

Issues in recording a global state

If a global physical clock were available, the following simple procedure could be used to record a consistent global snapshot of a distributed system.

In this, the initiator of the snapshot collection decides a future time at which the snapshot is to be taken and broadcasts this time to every process. All processes take their local snapshots at that instant in the global time.

The snapshot of channel Cij includes all the messages that process pj receives after taking the snapshot and whose timestamp is smaller than the time of the snapshot. If channels are not FIFO, a termination detection scheme will be needed to determine when to stop waiting for messages on channels.

Issues

Global physical clock is not available , the following two issues need to be addressed in recording of a consistent global snapshot of a distributed system

- **I1:** How to distinguish between the messages to be recorded in the snapshot (either in a channel state or a process state) from those not to be recorded.

The answer to this comes from conditions **C1** and **C2** as follows:

Any message that is sent by a process before recording its snapshot, must be recorded in the global snapshot (from **C1**).

Any message that is sent by a process after recording its snapshot, must not be recorded in the global snapshot (from **C2**).

Issues

- **I2:** How to determine the instant when a process takes its snapshot.

  The answer to this comes from condition **C2** as follows:

  A process pj must record its snapshot before processing a message mij, that was sent by process pi after recording its snapshot.

Lamport clock

Lamport clocks and vector clocks are replacements for physical clocks which rely on counters and communication to determine the order of events across a distributed system. These clocks provide a counter that is comparable across different nodes.

A Lamport clock is simple. Each process maintains a counter using the following rules:

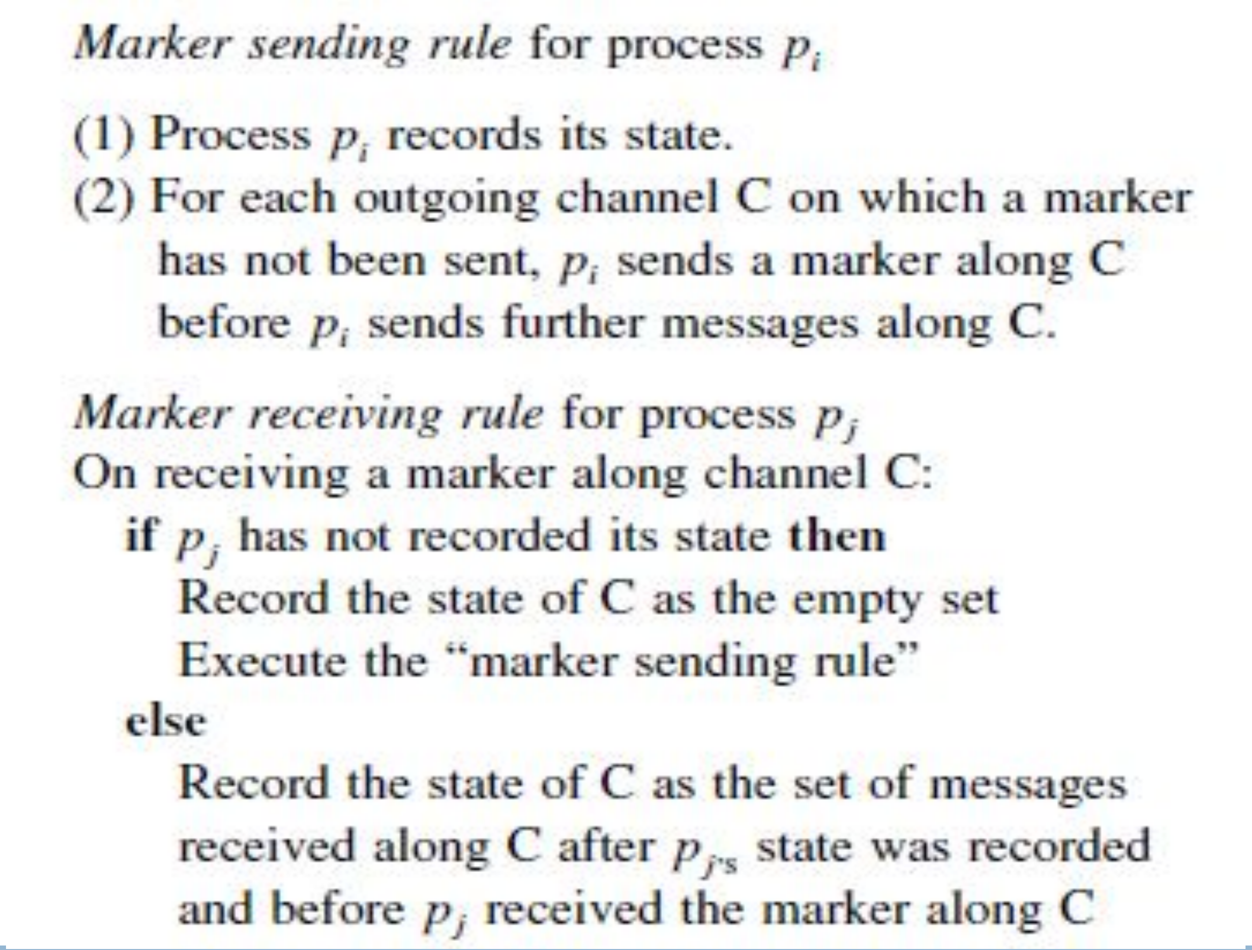
- Whenever a process does work, increment the counter
- Whenever a process sends a message, include the counter
- When a message is received, set the counter to max(local_counter, received_counter) + 1

Snapshot algorithms for FIFO channels- first algorithm to record the global snapshot

Chandy–Lamport algorithm:

- The **Chandy–Lamport algorithm** is a snapshot **algorithm** that is used in distributed systems for recording a consistent global state of an asynchronous system. It was developed by and named after Leslie **Lamport** and K. Mani **Chandy**.
- The Chandy-Lamport algorithm uses a control message, called a *marker*. After a site has recorded its snapshot, it sends a *marker* along all of its outgoing channels before sending out any more messages.
- Since channels are FIFO, a marker separates the messages in the channel into those to be included in the snapshot (i.e., channel state or process state) from those not to be recorded in the snapshot. This addresses issue **I1**. The role of markers in a FIFO system is to act as delimiters for the messages in the channels so that the channel state recorded by the process at the receiving end of the channel satisfies the condition **C2**.

Algorithm



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
```

Working

- Initiates snapshot collection by executing the marker sending rule by which it records its local state and sends a marker on each outgoing channel.
- The algorithm can be initiated by any process by executing the marker sending rule.
- The algorithm terminates after each process has received a marker on all of its incoming channels.

Chandy–Lamport snapshot algorithm

The assumptions of the algorithm are as follows:

- Neither channels nor processes fail, *communication is reliable*
- Channels are unidirectional and provide *FIFO-ordered delivery*
- There is a communication path between any two processes in the system
- Any process may initiate the snapshot algorithm
- The snapshot algorithm does not interfere with the normal execution of the processes
- Each process in the system records its local state and the state of its incoming channels
- Global state is collected in a distributed manner

RAMAIAH
Institute of Technology

Chandy–Lamport snapshot algorithm

The snapshot algorithm works using marker messages. The marker message is a special control message and not an application message. It does not interfere with application messages. The algorithm has three phases: Initiation, Propagation and Termination and works as follows:

1. Initiating a Snapshot

Process Pi initiates the snapshot.

The initiator, Pi records its own state and creates the marker message.

Pi then forwards the marker message to all other processes using the using N-1 outbound channels Cij(j = 1 to N & j != i).

Pi starts recording all incoming messages from channels Cji (j = 1 to N & j != i).

RAMAIAH Institute of Technology

Chandy–Lamport snapshot algorithm

2. Propagating a Snapshot

If a process Pj receives a Marker message on an incoming channel Ckj, and if this is the first Marker that Pj is seeing:

Pj records its own state and mark Ckj as empty.

Pj then forwards the marker message to all other processes using the N-1 outbound channels.

Pj starts recording all incoming messages from channels Clj for l not equal to j or k.

If the process Pj has already seen a Marker message:

Mark the state of channel Ckj as all the messages that have arrived on it since the recording was turned on for Ckj.

Chandy–Lamport snapshot algorithm

3. Terminating a Snapshot

The algorithm terminates when:

All processes have received a marker (which implies that process state has been recorded)

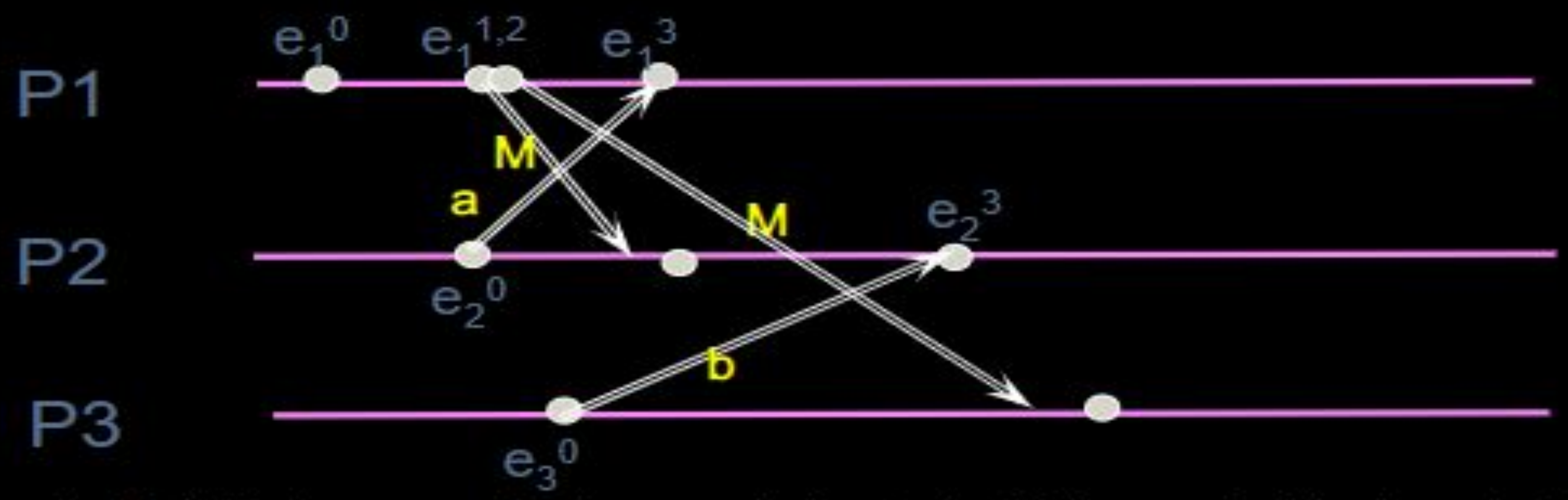
All processes have received a marker on all the N-1 incoming channels (which implies that state of all channels has been recorded)

Later, if needed, the central server collects all these partial states to build the global snapshot.

RAMAIAH
Institute of Technology

Implementation

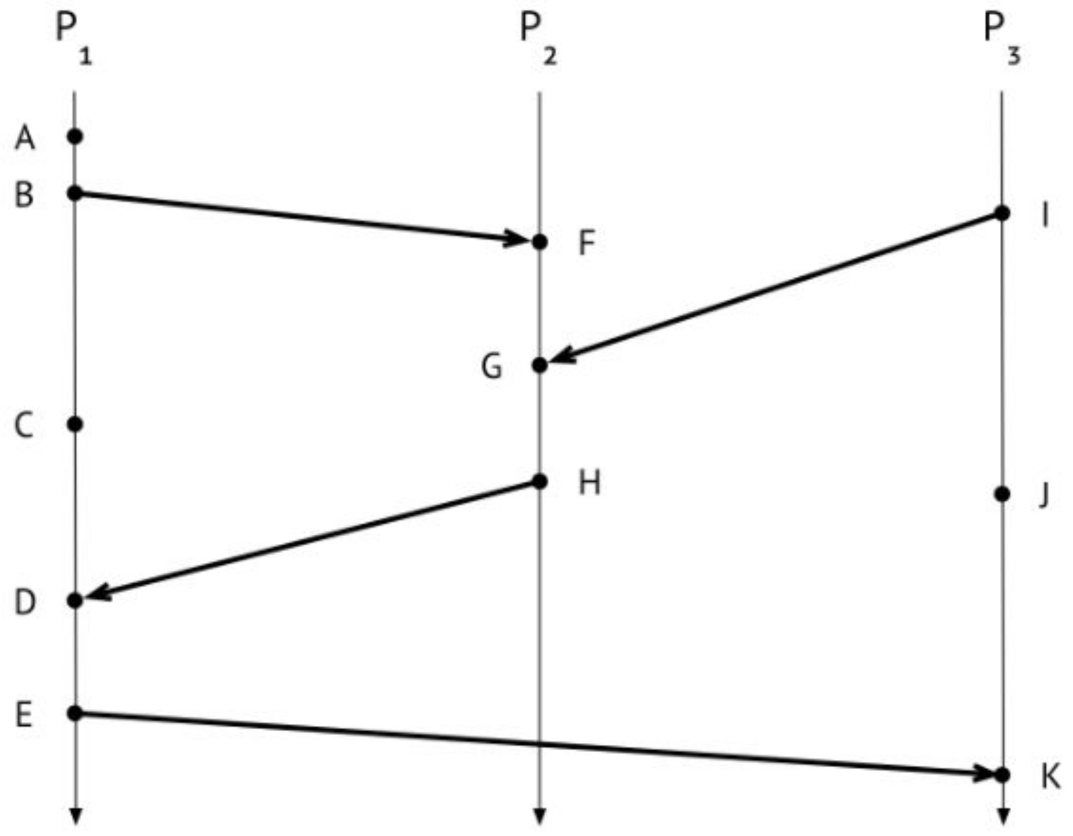
Chandy Lamport Algorithm

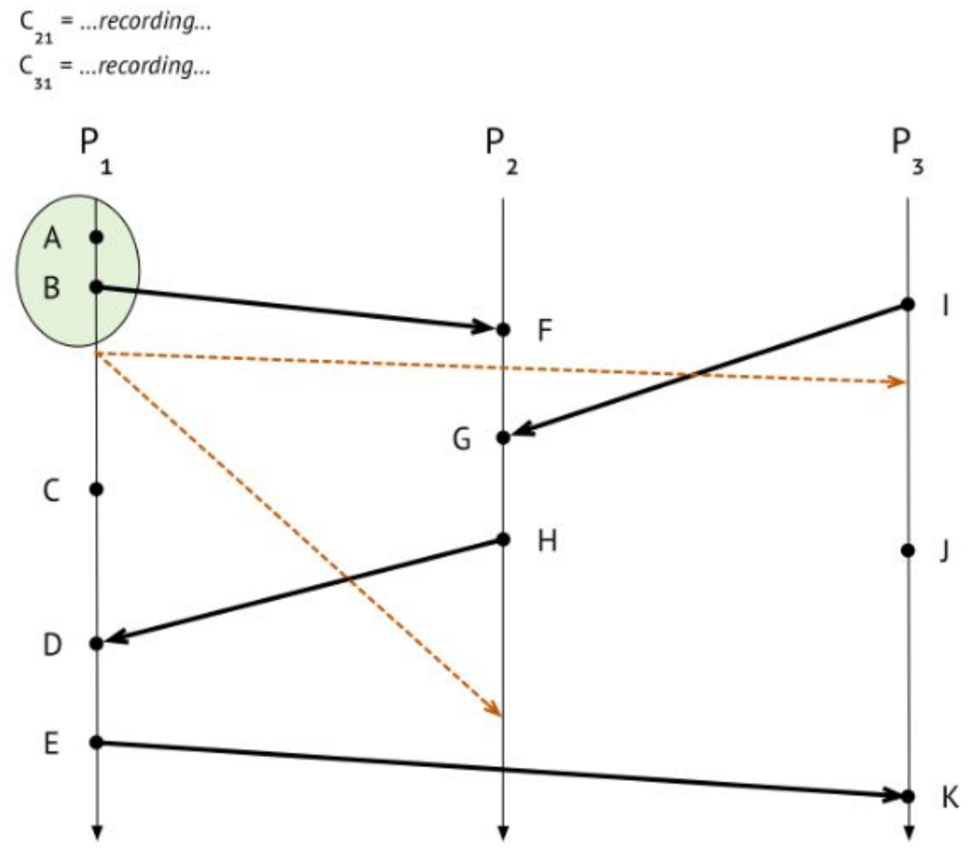


1- P1 initiates snapshot: records its state (S1); sends Markers to P2 & P3; turns on recording for channels Ch_{21} and Ch_{31}

Chandy–Lamport algorithm

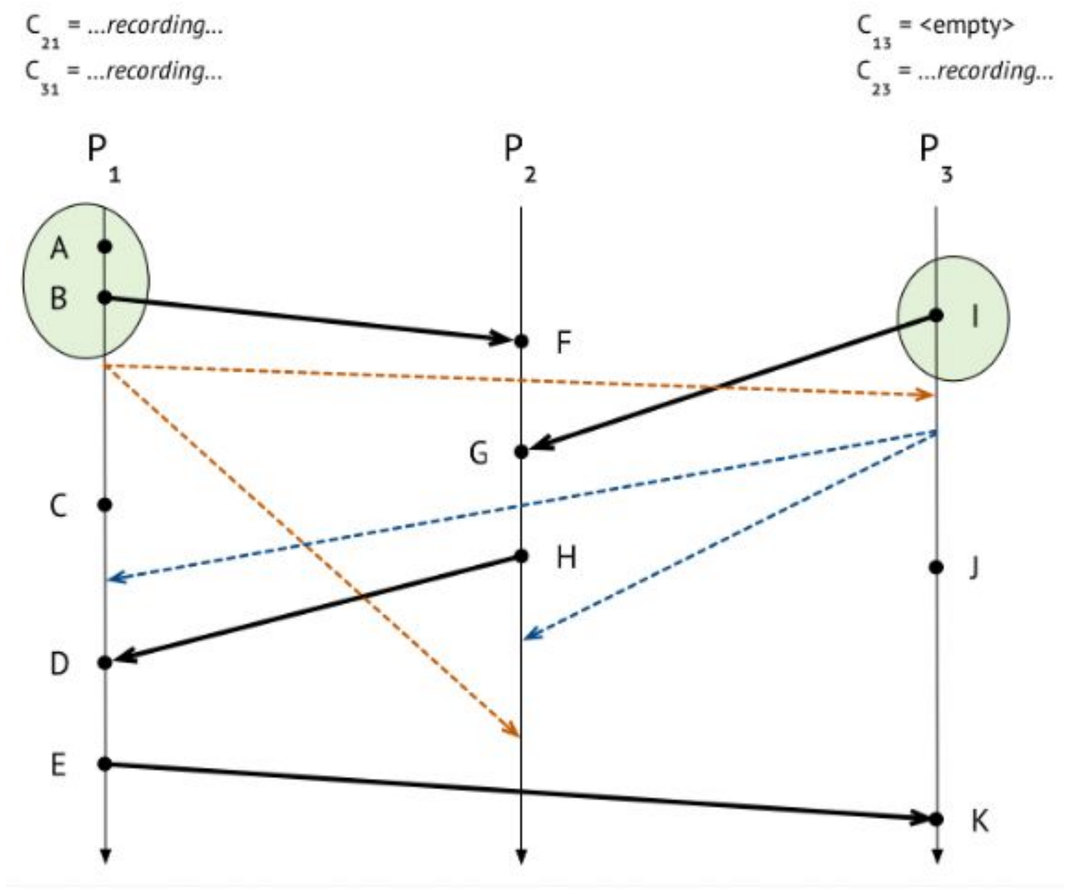
There are 3 process, with several events. Dots on process lines with no incoming or outgoing are internal events

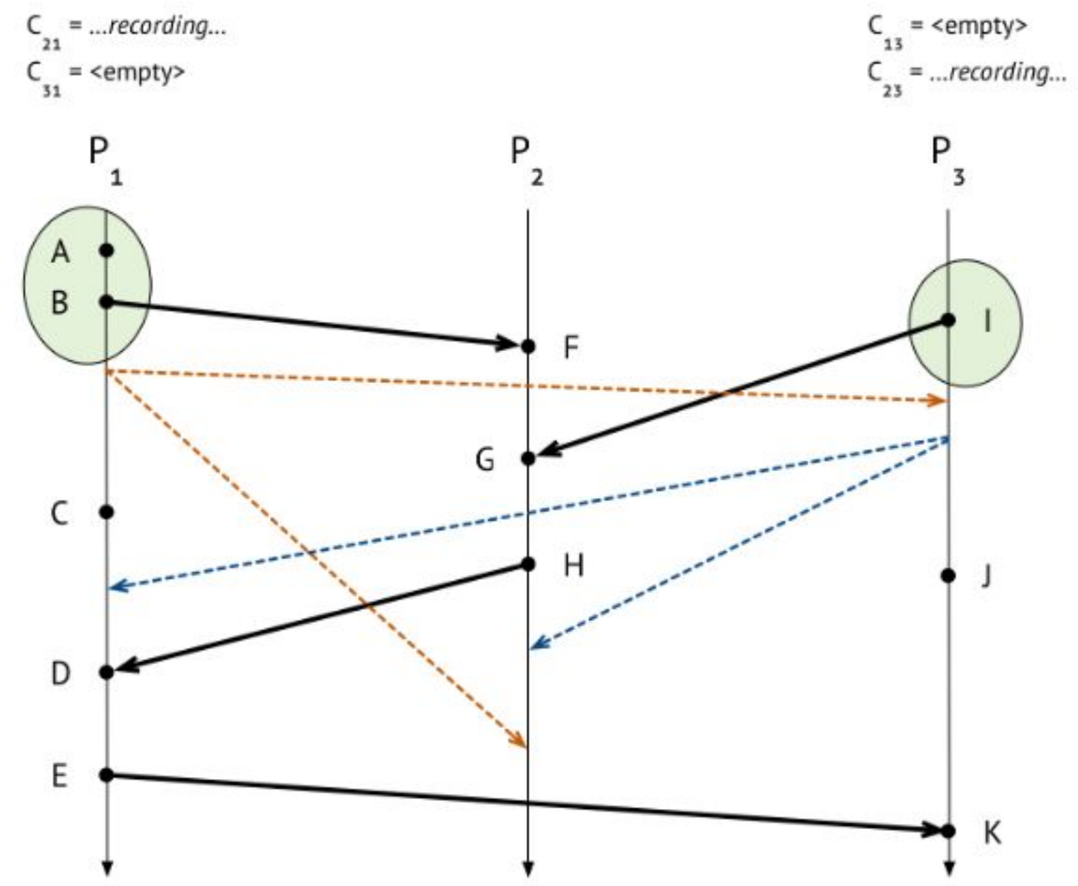




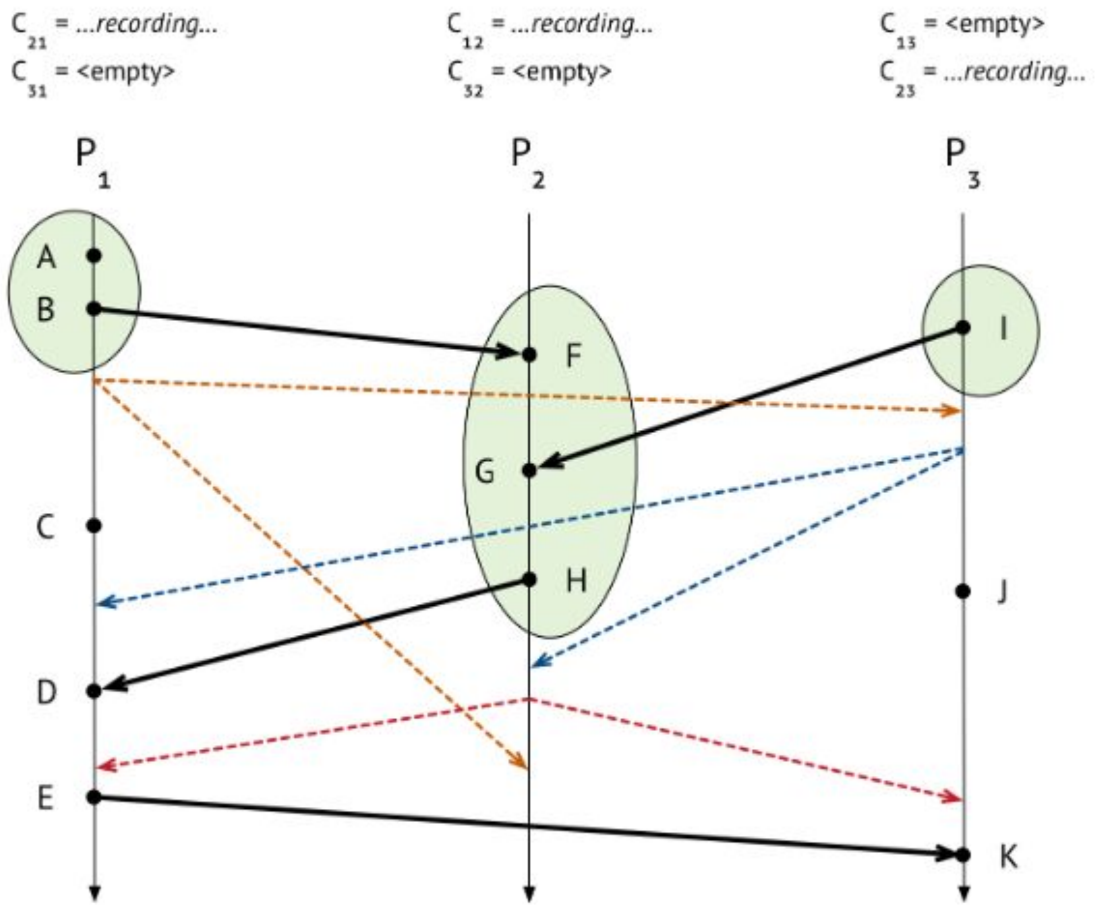
RAMAIAH
Institute of Technology

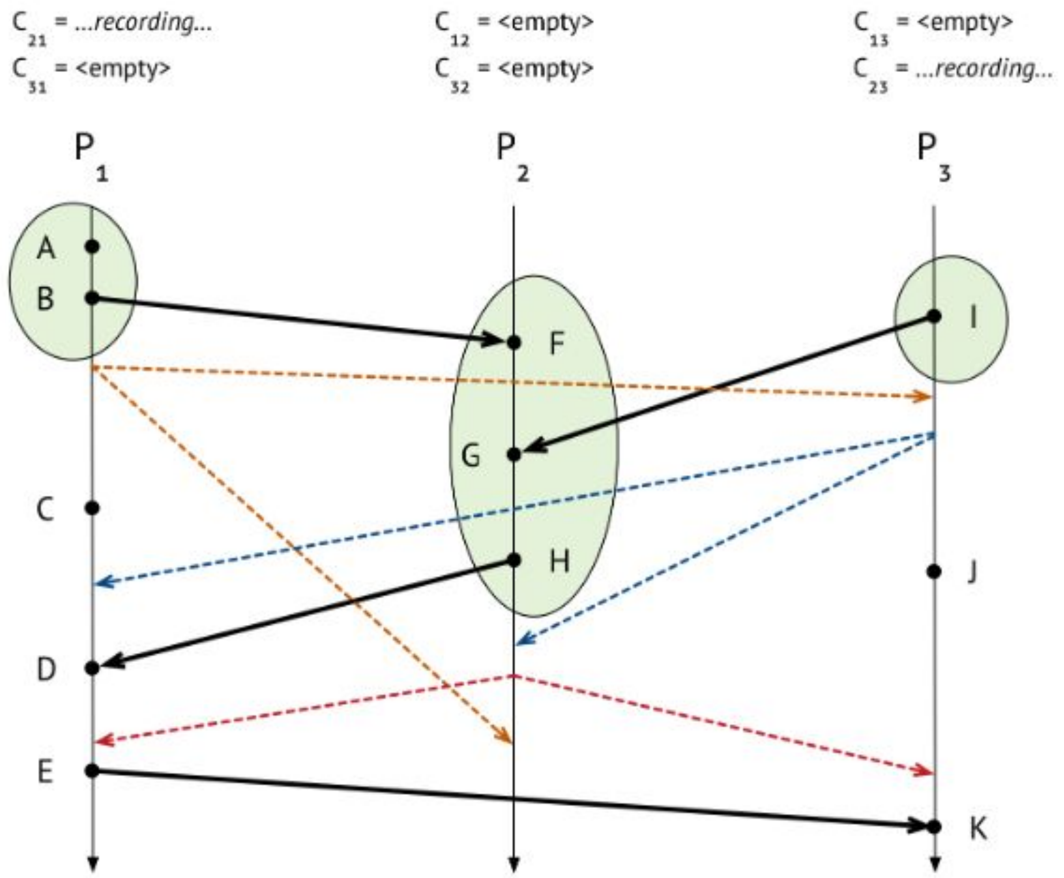
Chandy–Lamport algorithm


C_{21} = ...recording...
C_{31} = ...recording...
C_{13} = <empty>
C_{23} = ...recording...
P_1
P_2
P_3
A
B
C
D
E
F
G
H
I
J
K

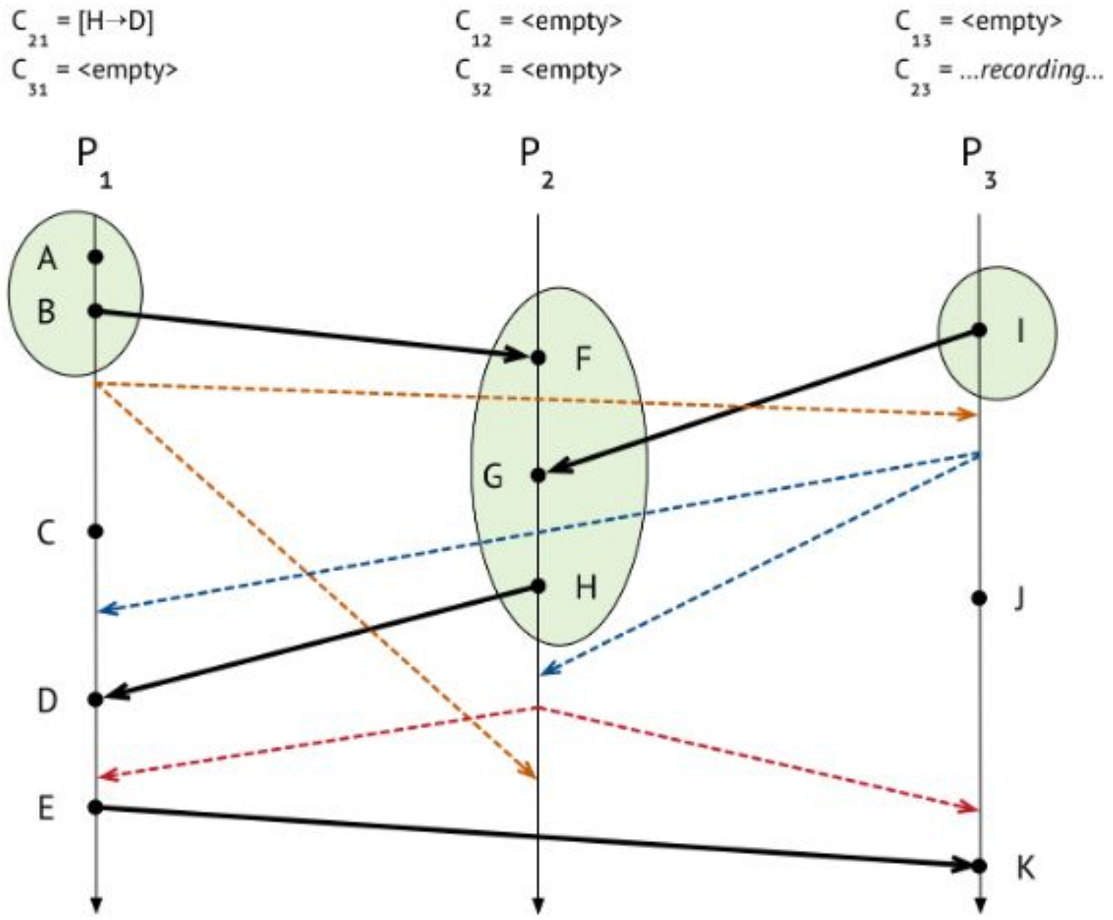


Chandy–Lamport algorithm

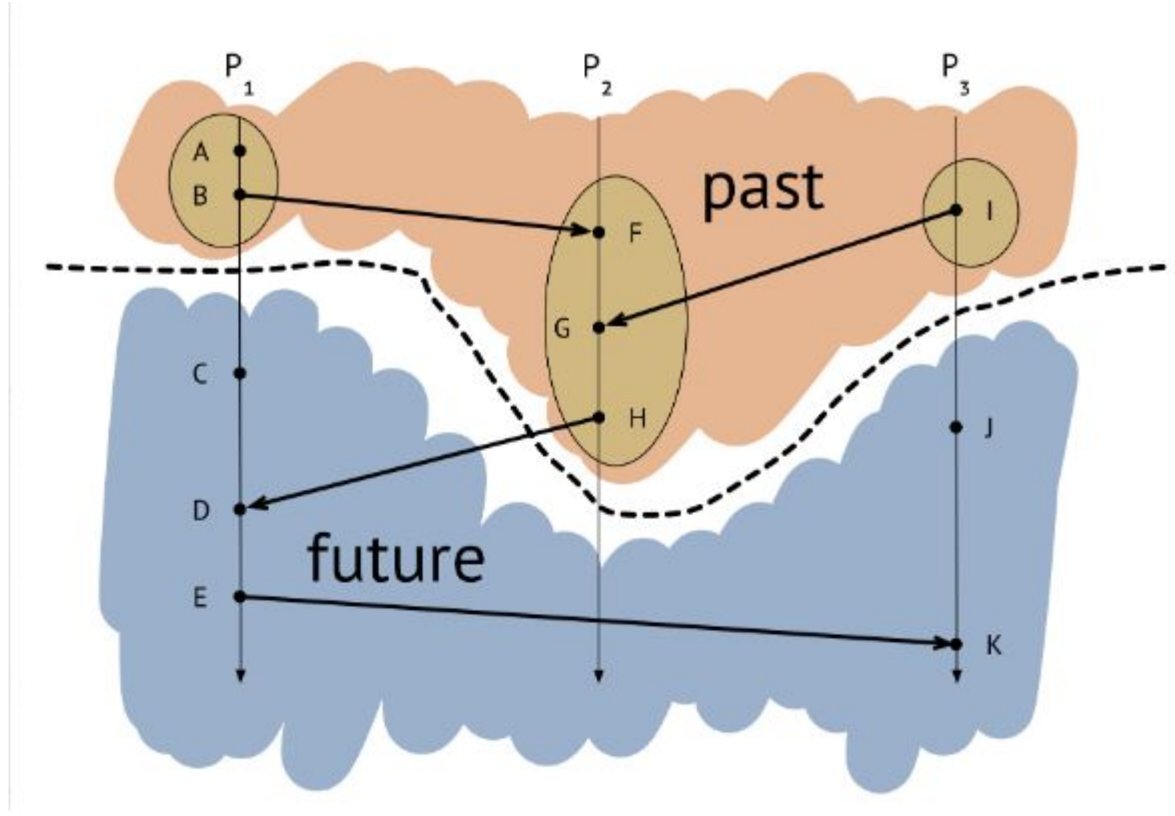




Chandy–Lamport algorithm



Chandy–Lamport algorithm



Global Snapshot

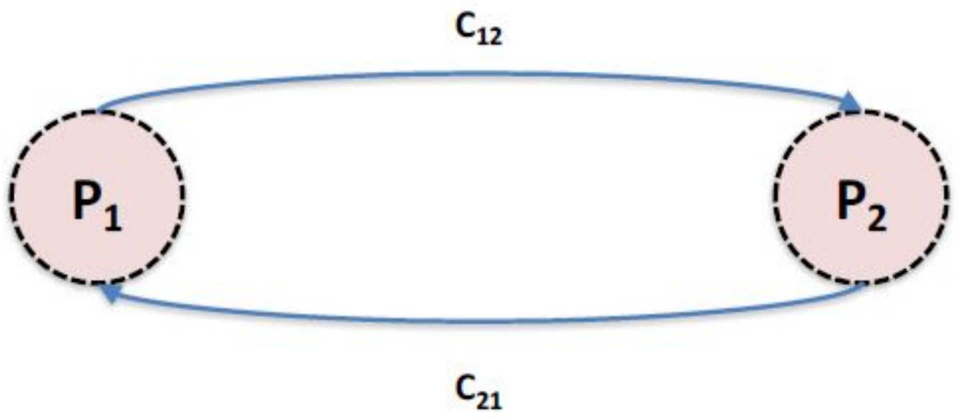
- The recorded local snapshots can be put together to create the global snapshot in several ways.
  - One policy is to have each process send its local snapshot to the initiator of the algorithm.
  - Another policy is to have each process send the information it records along all outgoing channels, and to have each process receiving such information for the first time propagate it along its outgoing channels. All the local snapshots get disseminated to all other processes and all the processes can determine the global state.

Example of global snapshots

- Two processes: P_1 and P_2

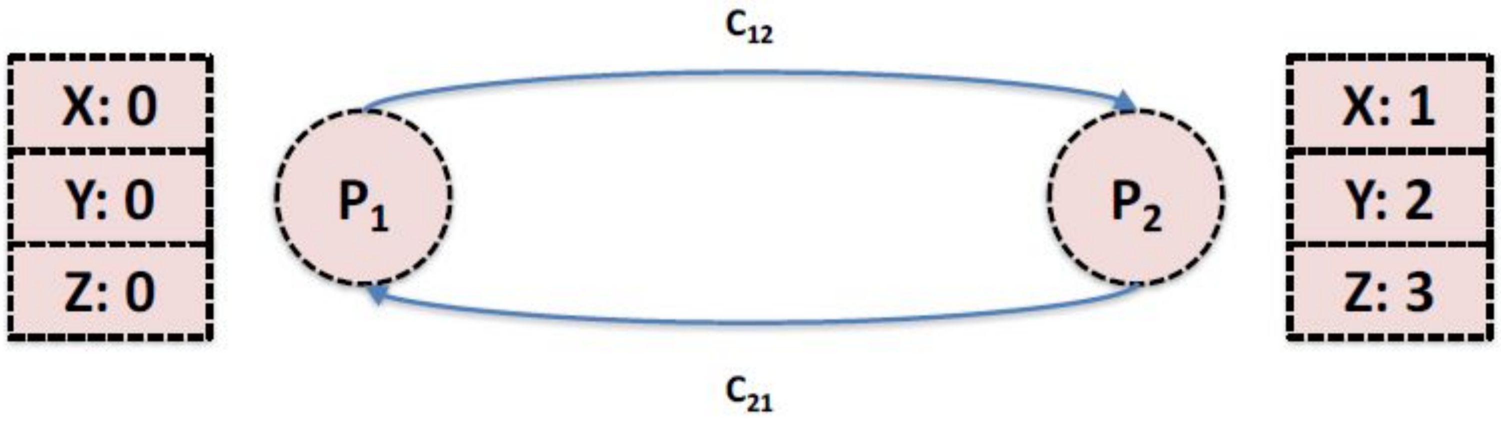
Example of global snapshots

- Channel C_{12} from P_1 to P_2
- Channel C_{21} from P_2 to P_1



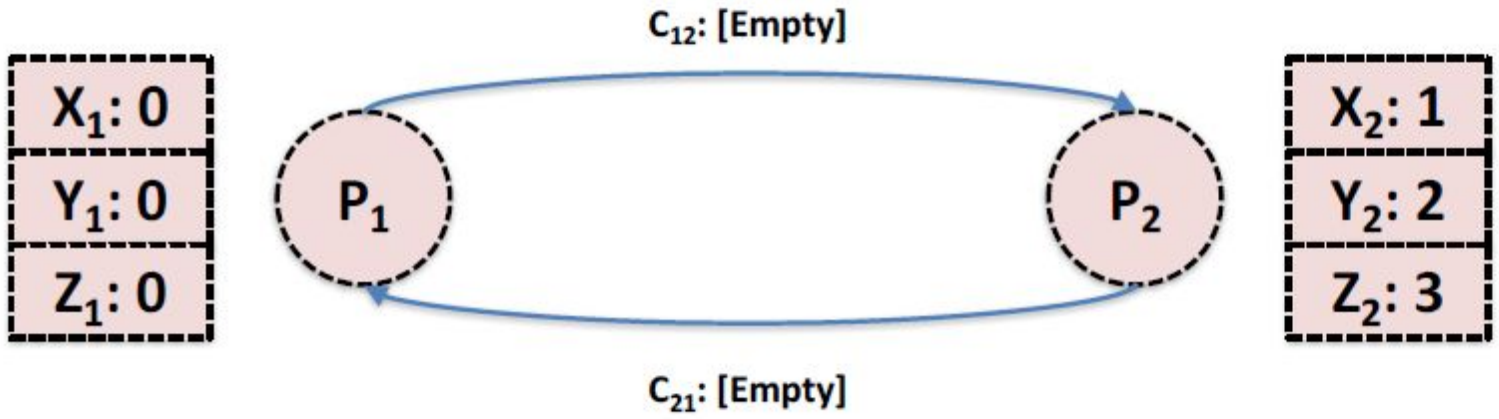
Example of global snapshots

- Process states for P_1 and P_2



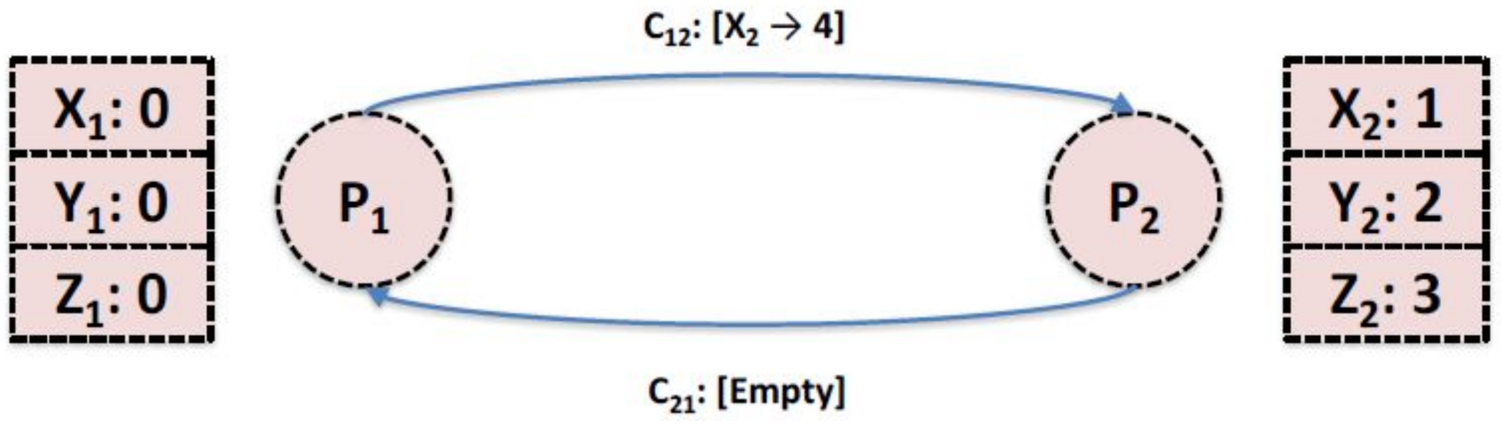
Example of global snapshots

- Channel states (i.e., messages) for C_{12} and C_{21}
- This is our initial global state
- Also a global snapshot



Example of global snapshots

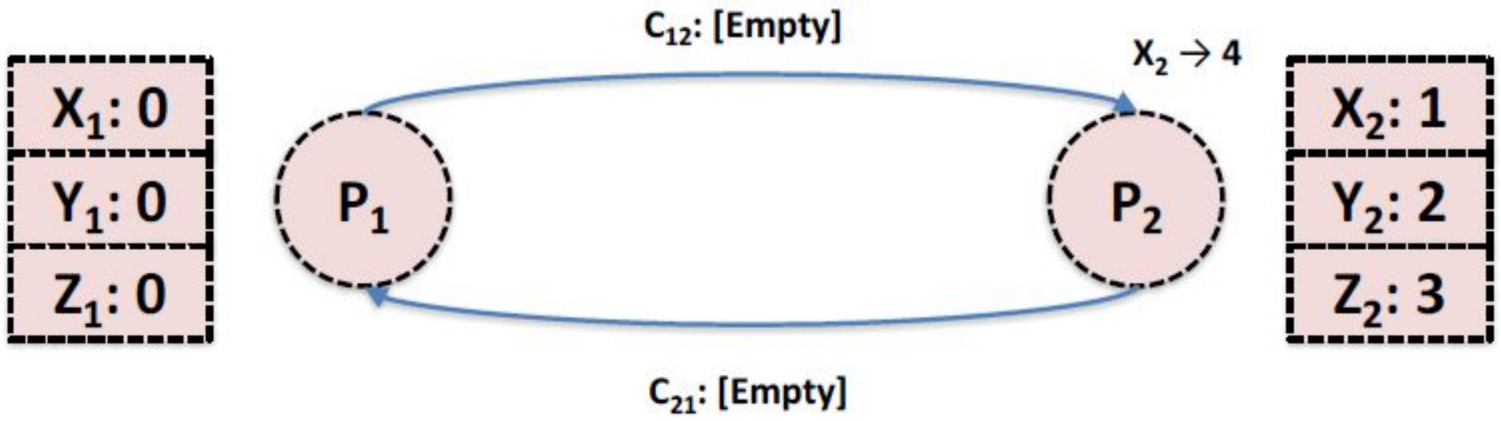
- P_1 tells P_2 to change its state variable, X_2, from 1 to 4
- This is another global snapshot



RAMAIAH Institute of Technology

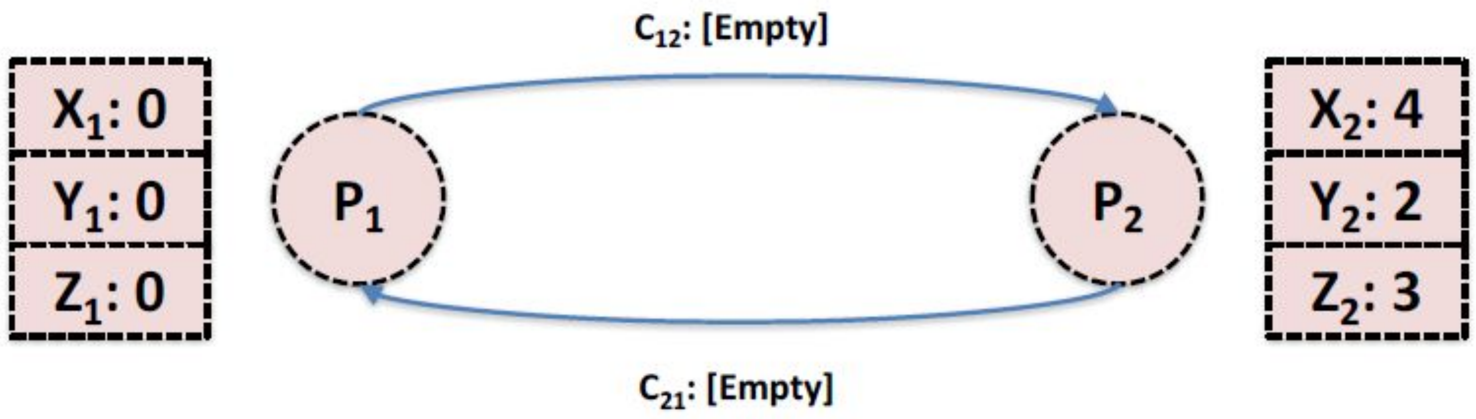
Example of global snapshots

- P_2 receives the message from P_1
- Another global snapshot



Example of global snapshots

- P_2 changes its state variable, X_2, from 1 to 4
- And another global snapshot



Correctness

To prove the correctness of the algorithm, we show that a recorded snapshot satisfies conditions C1 and C2.

- Since a process records its snapshot when it receives the first marker on any incoming channel, no messages that follow markers on the channels incoming to it are recorded in the process's snapshot.

Summary

• The global state changes whenever an event

happens

– Process sends message

– Process receives message

– Process takes a step

• Moving from state to state **obeys causality**

RAMAIAH
Institute of Technology

Thank you