Chandy-Lamport Snapshotting

COS 418: *Distributed Systems*
Precept 8

Themis Melissaris and Daniel Suo

[Content adapted from I. Gupta]

Agenda

- What are global snapshots?
- The Chandy-Lamport algorithm
- Why does Chandy-Lamport work?

Global snapshots

Example of a global snapshot

But that was easy

- In our system of world leaders, we were able to capture their ‘state’ (i.e., likeness) easily
  - Synchronized in space
  - Synchronized in time
- How would we take a global snapshot if the leaders were all at home?
- What if Obama told Trudeau that he should really put on a shirt?
- This message is part of our system state!

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

Why do we need snapshots?

- **Checkpointing**: 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 than printf…

We could just synchronize clocks

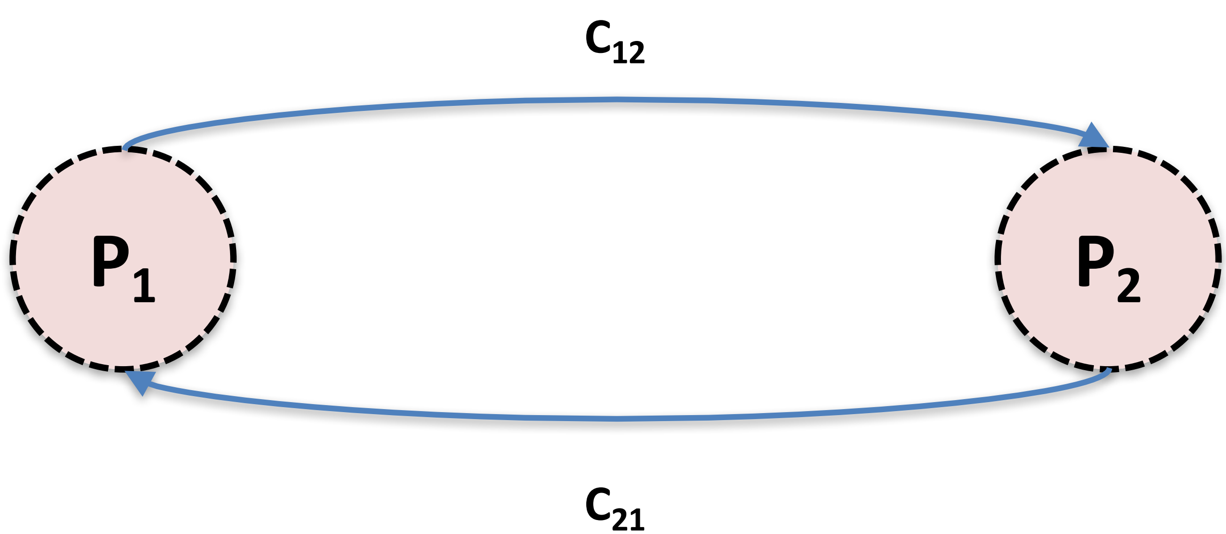
- Each process records state at time some agreed upon *t*
  - But clocks skew
  - And we wouldn't record messages
- Do we need synchronization?
- What did Lamport realize about ordering events?

Example of global snapshots v2

- Two processes: P_1 and P_2

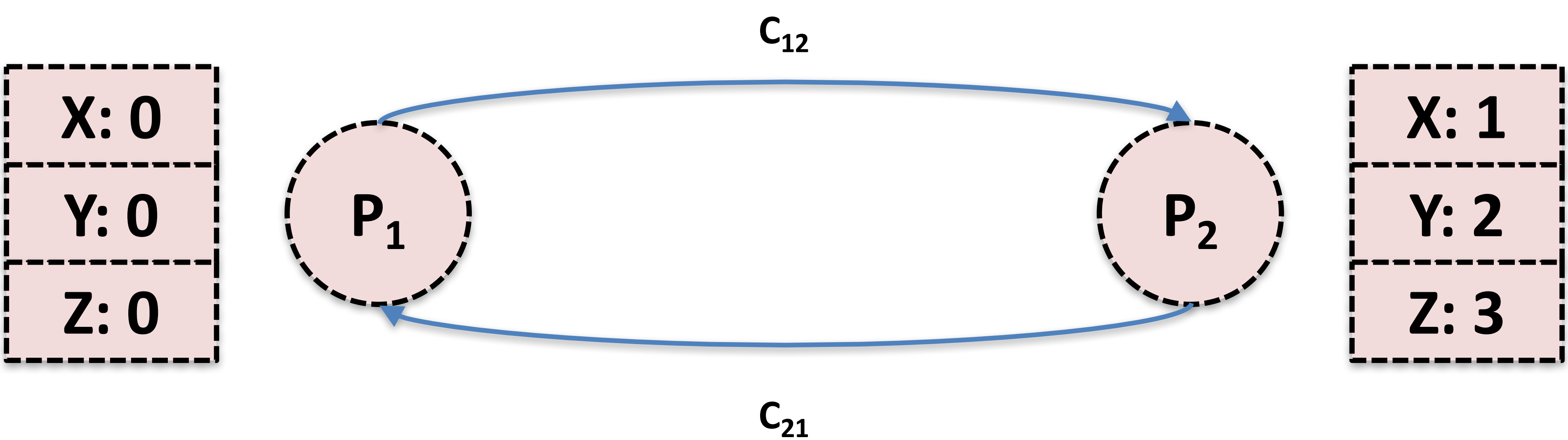
Example of global snapshots v2

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



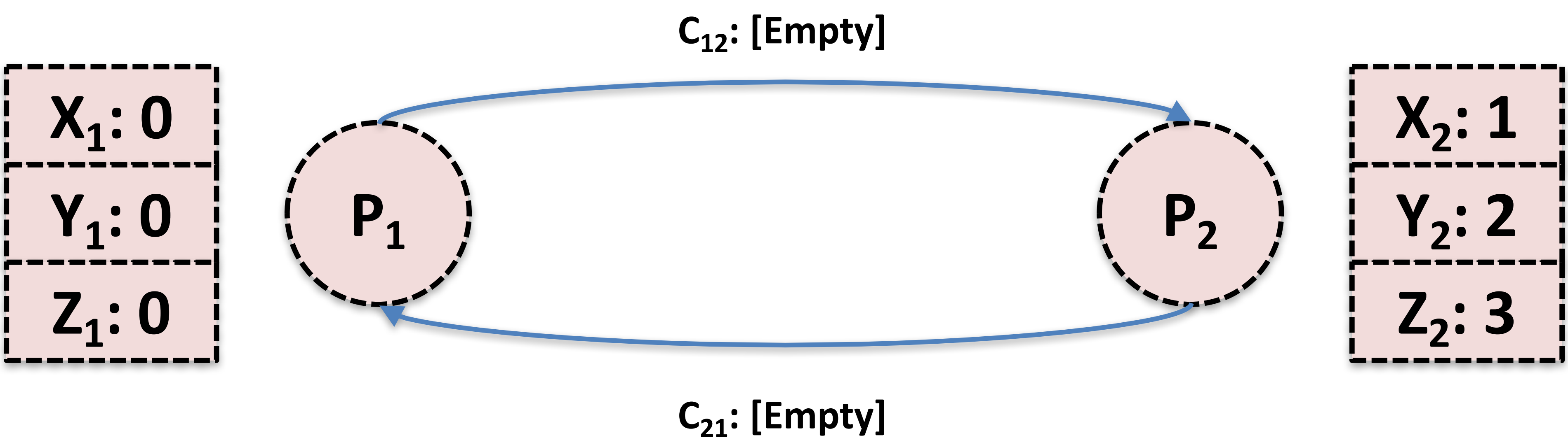
Example of global snapshots v2

- Process states for P_1 and P_2



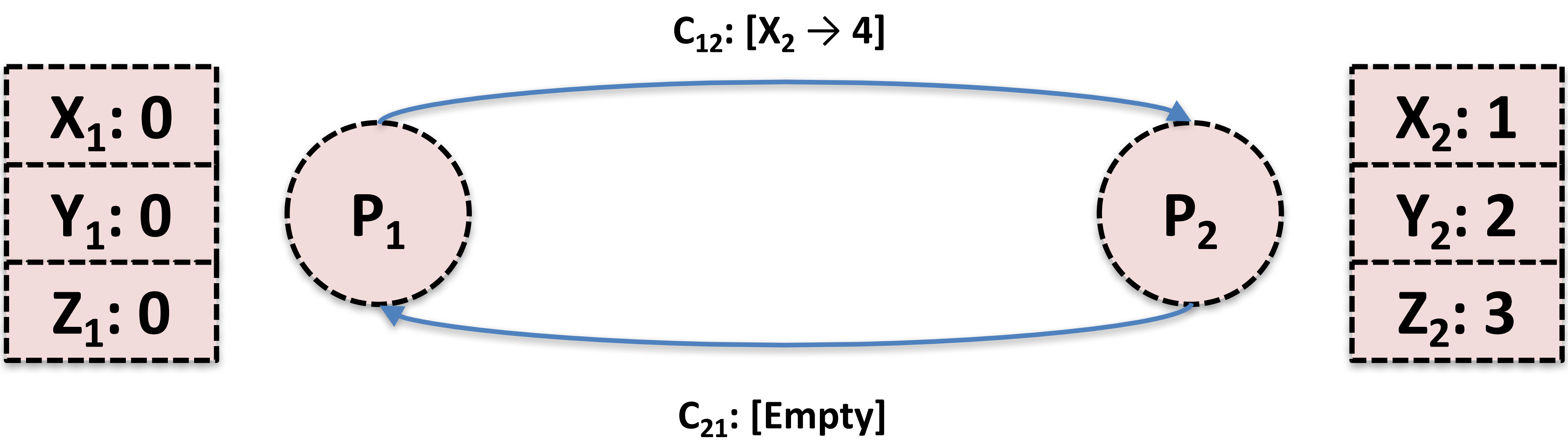
Example of global snapshots v2

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



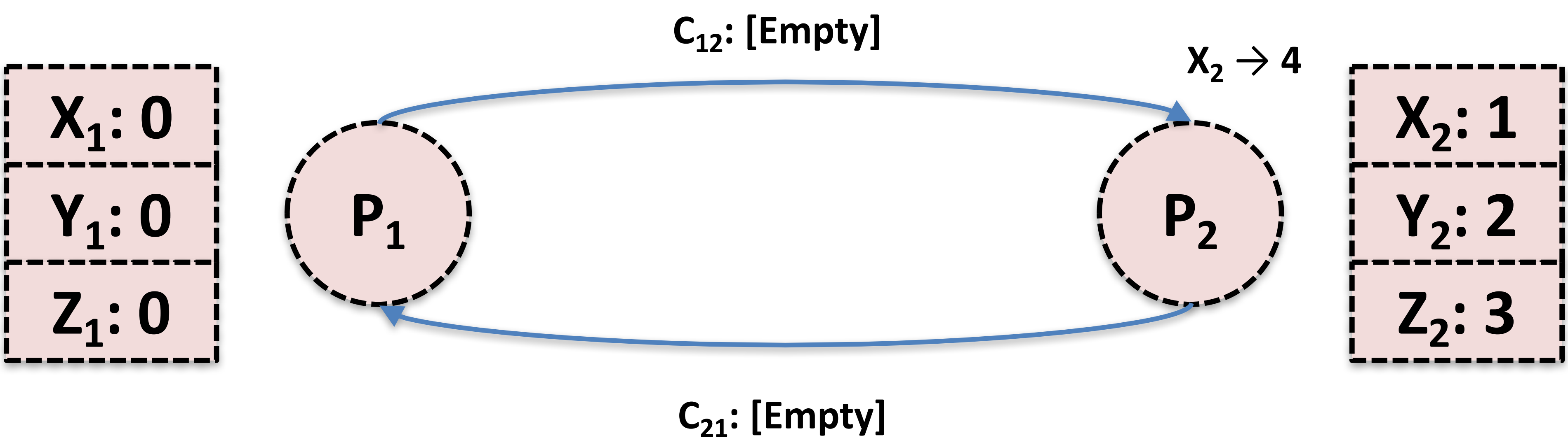
Example of global snapshots v2

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



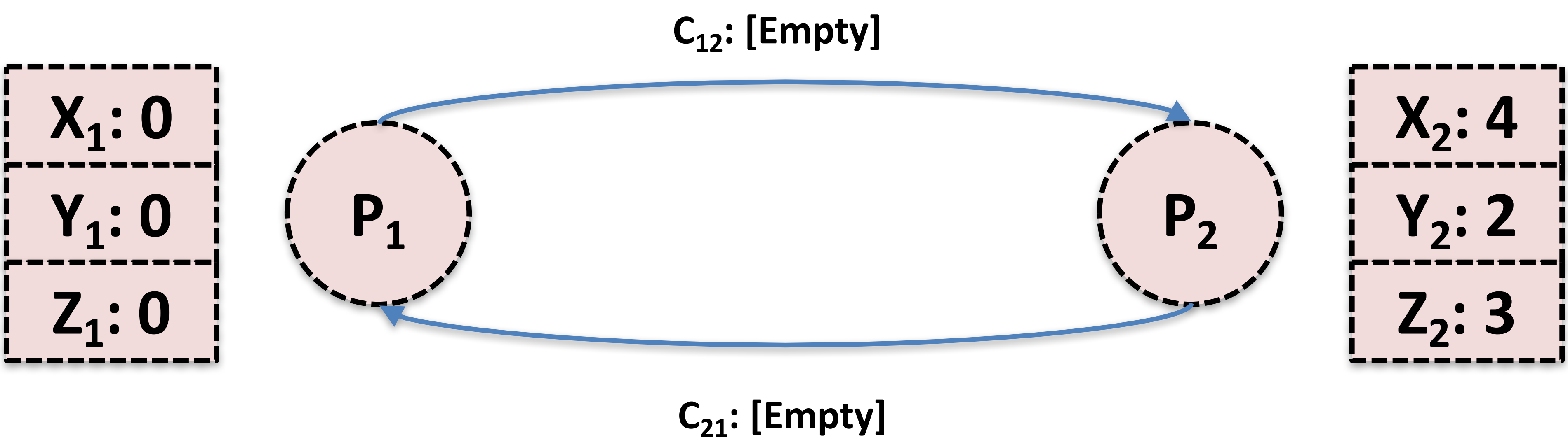
Example of global snapshots v2

- P_2 receives the message from P_1
- Another global snapshot



Example of global snapshots v2

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

Chandy-Lamport algorithm

System model

- **Problem**: record a global snapshot (state for each process and channel)
- **Model**
  - *N* processes in the system with no failures
  - There are two FIFO unidirectional channels between every process pair ($P_i \rightarrow P_j$ and $P_j \rightarrow P_i$)
  - All messages arrive, intact, not duplicated
- Future work relaxes these assumptions

System requirements

- Taking a snapshot shouldn’t interfere with normal application behavior
  - Don’t stop sending messages
  - Don’t stop the application!
- Each process can record its own state
- Collect state in a distributed manner
- Any process can initiate a snapshot

Initiating a snapshot

- Let's say process P_i initiates the snapshot
- P_i records its own state and prepares a special marker message (distinct from application messages)
- Send the marker message to all other processes (using *N-1* outbound channels)
- Start recording all incoming messages from channels C_{ji} for j not equal to i

Propagating a snapshot

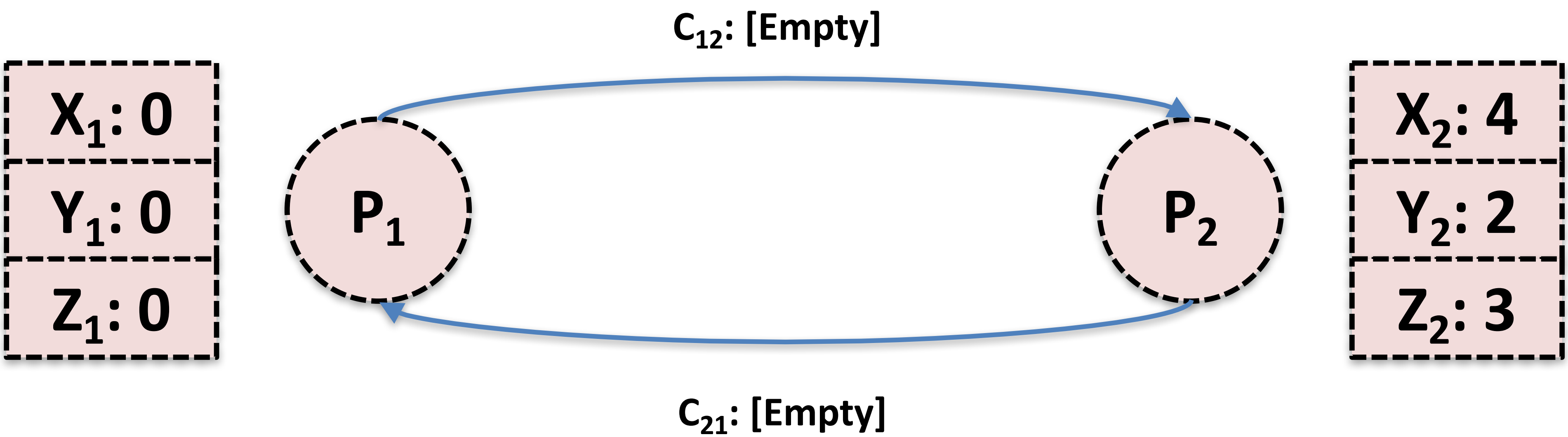
- For all processes P_j (including the initiator), consider a message on channel C_{kj}
- If we see marker message for the first time
  - P_j records own state and marks C_{kj} as empty
  - Send the marker message to all other processes (using *N-1* outbound channels)
  - Start recording all incoming messages from channels C_{lj} for *l* not equal to *j* or *k*
- Else add all messages from inbound channels since we began recording to their states

Terminating a snapshot

- All processes have received a marker (and recorded their own state)
- All processes have received a marker on all the *N-1* incoming channels (and recorded their states)
- Later, a central server can gather the partial state to build a global snapshot

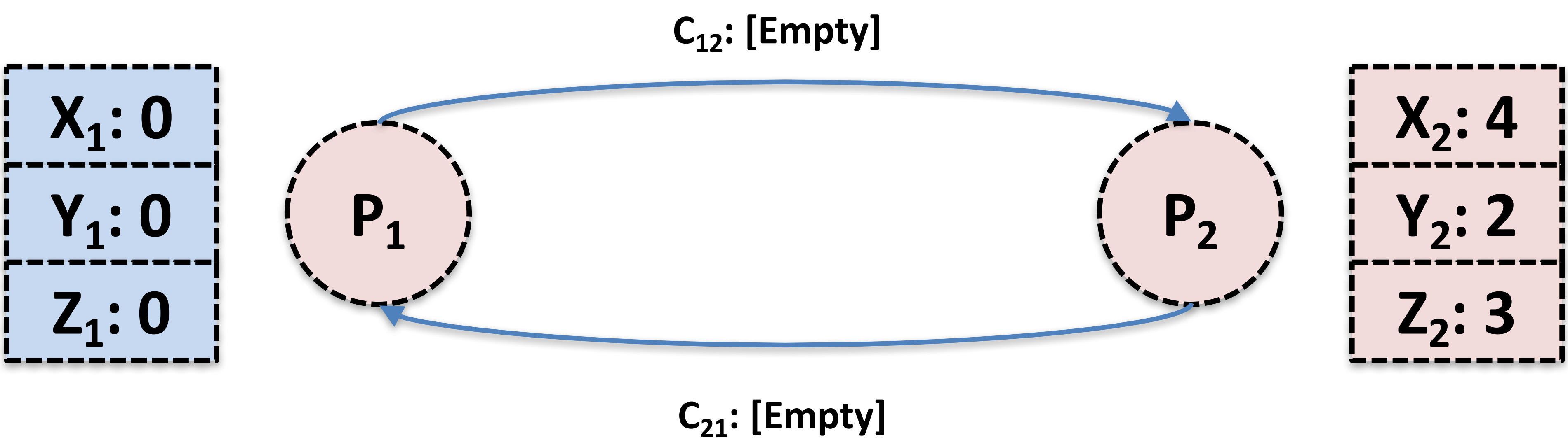
Example

- P_1 initiates a snapshot



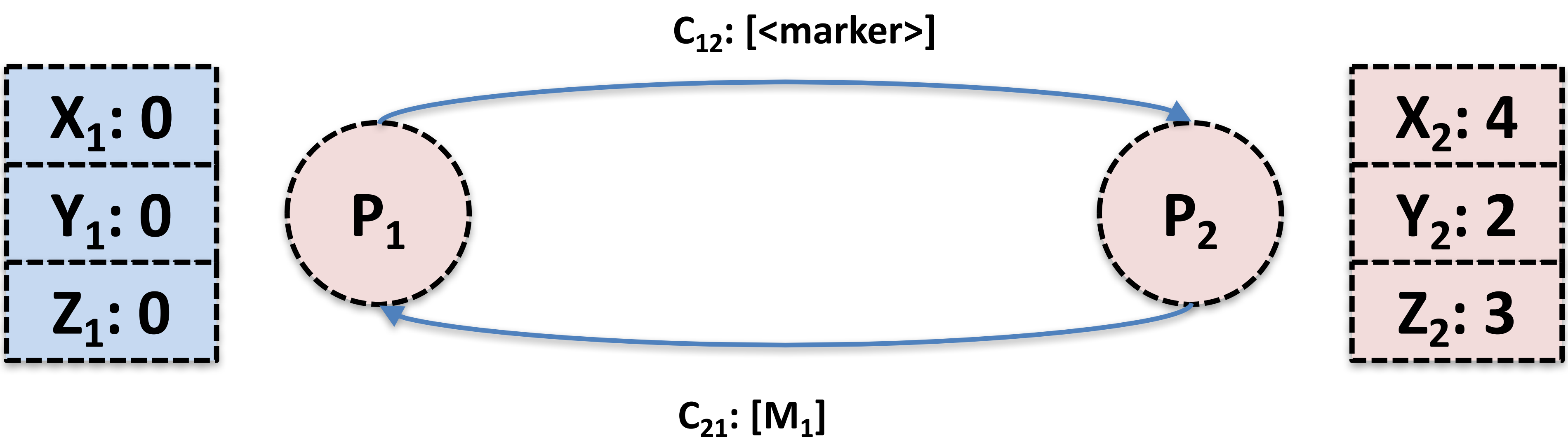
Example

- First, P_1 records its state



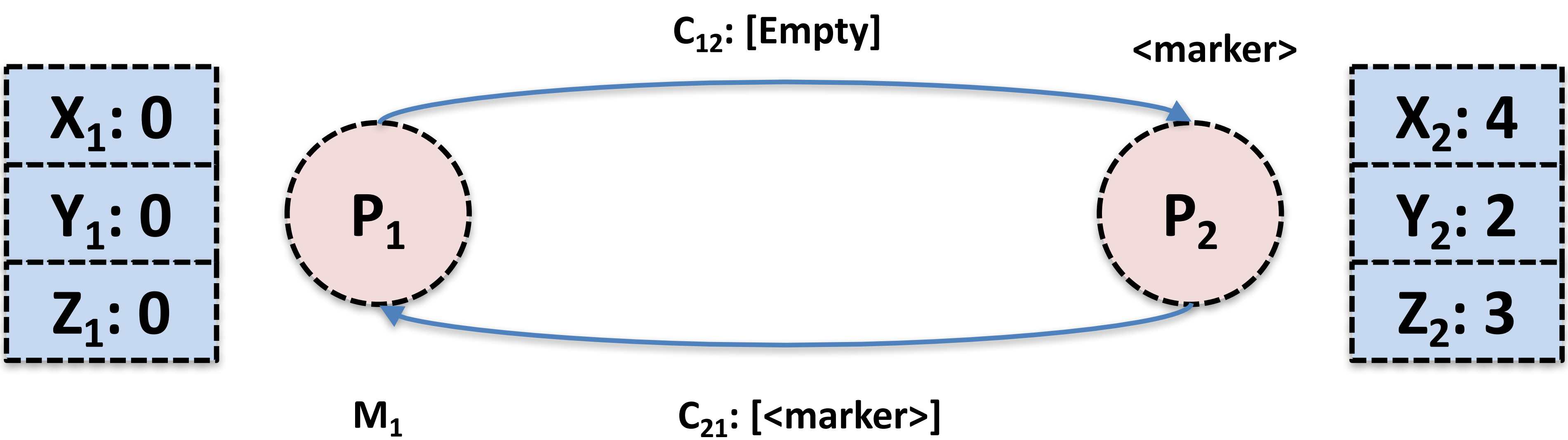
Example

- Then, P_1 sends a marker message to P_2 and begins recording all messages on inbound channels
- Meanwhile, P_2 sent a message to P_1



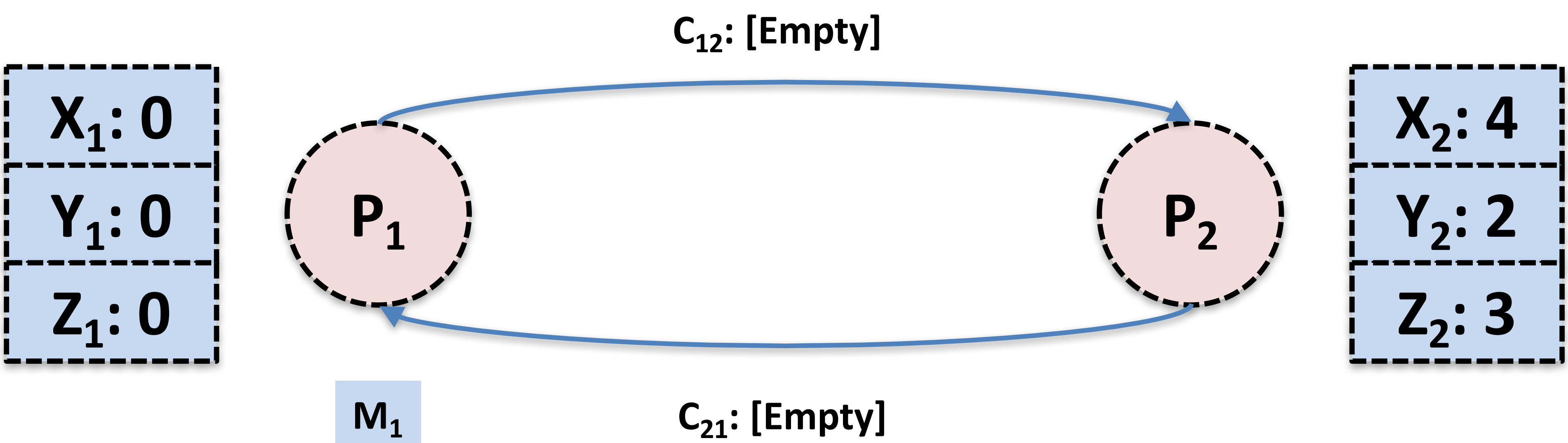
Example

- P_2 receives a marker message for the first time, so records its state
- P_2 then sends a marker message to P_1



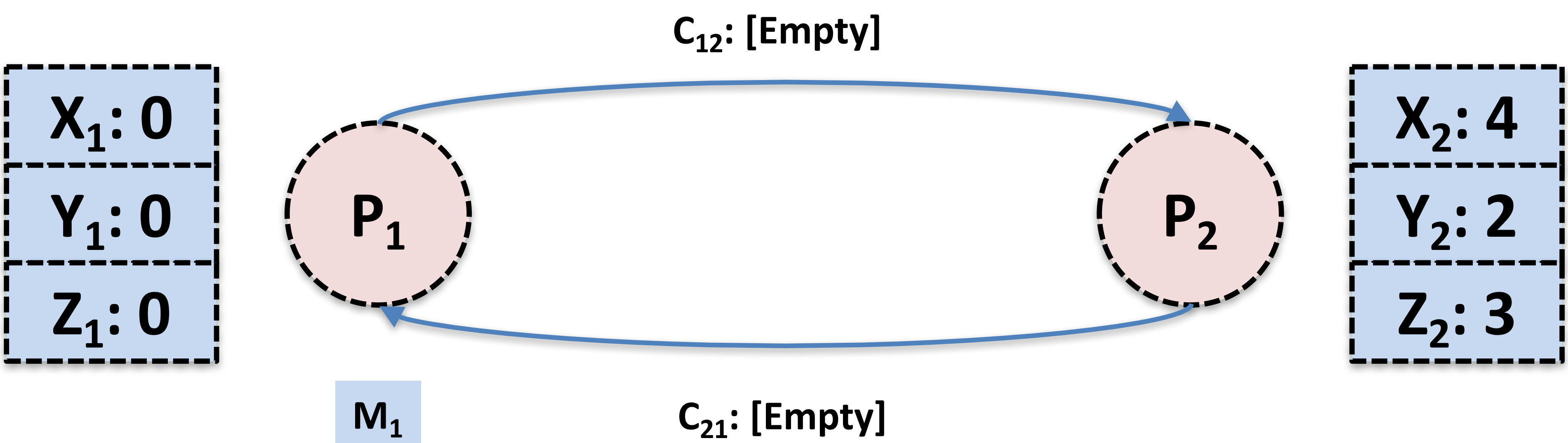
Example

- P_1 has already sent a marker message, so it records all messages it received on inbound channels to the appropriate channel's state



Example

- Both processes have recorded their state and all the state of all incoming channels
- Our snapshotted state is highlighted in blue



Reasoning about the Chandy-Lamport algorithm

Causal consistency

- Related to the Lamport clock partial ordering
- An event is presnapshot if it occurs before the local snapshot on a process
- Postsnapshot if afterwards
- If event *A* happens causally before event *B*, and *B* is presnapshot, then *A* is too

Proof

- If *A* and *B* happen on the same process, then this is trivially true
- Consider when *A* is the send and *B* is the corresponding receive event on processes *p* and *q,* respectively
  - Since *B* is presnapshot, *q* can't have received a marker and *p* can't have sent a marker
  - *A* must also happen presnapshot
- Similar logic for *A* happening postsnapshot

Poking the proof: Part I

- In order for an application message *m* in the channel from process *p* to process *q* to be in the snapshot
  - Must happen after *q* has received its first marker
  - Before *p* has sent its marker to *q*
- A message *m* will only be in the snapshot if the sending process was presnapshot and the receiving process was postsnapshot

Poking the proof: Part II

- How do we order concurrent events?
  - Remember, all processes communicate
- What if a process receives a marker in between sending a marker and some event?
  - These should happen atomically
- What if something happens on a process independently of messages after the wall-clock time of when the snapshot starts?
  - Snapshots are causally consistent

Monday topic:

Streaming Data Processing