# CLOUD COMPUTING CONCEPTS

with Indranil Gupta (Indy)

## SNAPSHOTS

Lecture B

## GLOBAL SNAPSHOT ALGORITHM

# System Model

- **Problem: Record a global snapshot (state for each process, and state for each channel)**
- ***System Model*:** example DS assumptions
  - *N* processes in the system
  - There are two uni-directional communication channels between each ordered process pair : P*j* → P*i* and P*i* → P*j*
  - Communication channels are FIFO-ordered
    - First in, first out
  - No failure
  - All messages arrive intact and are not duplicated
    - Other papers later relaxed some of these assumptions

# Requirements

- **Snapshot should not interfere with normal application actions, and it should not require application to stop sending messages**
- **Each process is able to record its own state**
  - Process state: Application-defined state or, in the worst case:
  - Its heap, registers, program counter, code, etc. (essentially the coredump)
- **Global state is collected in a distributed manner**
- **Any process may initiate the snapshot**
  - We'll assume just one snapshot run for now

# Chandy-Lamport Global Snapshot Algorithm

- **First, Initiator P*i* records its own state**
- **Initiator process creates special messages called "Marker" messages**
  - Not an application message, does not interfere with application messages
- **for *j=1 to N* except *i***

  P*i* sends out a Marker message on outgoing channel $C_{ij}$
  - (*N-1*) channels
- Starts recording the incoming messages on each of the incoming channels at P*i*: $C_{ji}$ (for *j=1 to N* except *i*)

application/coredump state == major struct

marker msg == homebrew packet, sender tells receiver ps starts "global snapshot"

initiator == client ps that calls global snapshot

recording == starts u() struct + flush to disk ?

channels == list of maps{ps : any packets}

global snapshot == flush disk all struct above

# Chandy-Lamport Global Snapshot Algorithm (2)

**Whenever a process P*i* receives a Marker message on an incoming channel $C_{ki}$**

- **if** (this is the first Marker P*i* is seeing)
  - P*i* records its own state first
  - Marks the state of channel $C_{ki}$ as "empty"
  - For *j=1 to N* except *i*
    - P*i* sends out a Marker message on outgoing channel $C_{ij}$
  - Starts recording the incoming messages on each of the incoming channels at P*i*: $C_{ji}$ (for *j=1 to N* except *i* and *k*)
- **else // already seen a Marker message**
  - Mark the state of channel $C_{ki}$ as all the messages that have arrived on it since recording was turned on for $C_{ki}$

# Chandy-Lamport Global Snapshot Algorithm (3)

**The algorithm terminates when**

- All processes have received a Marker
  - To record their own state
- All processes have received a Marker on all the (*N*-*1*) incoming channels at each
  - To record the state of all channels

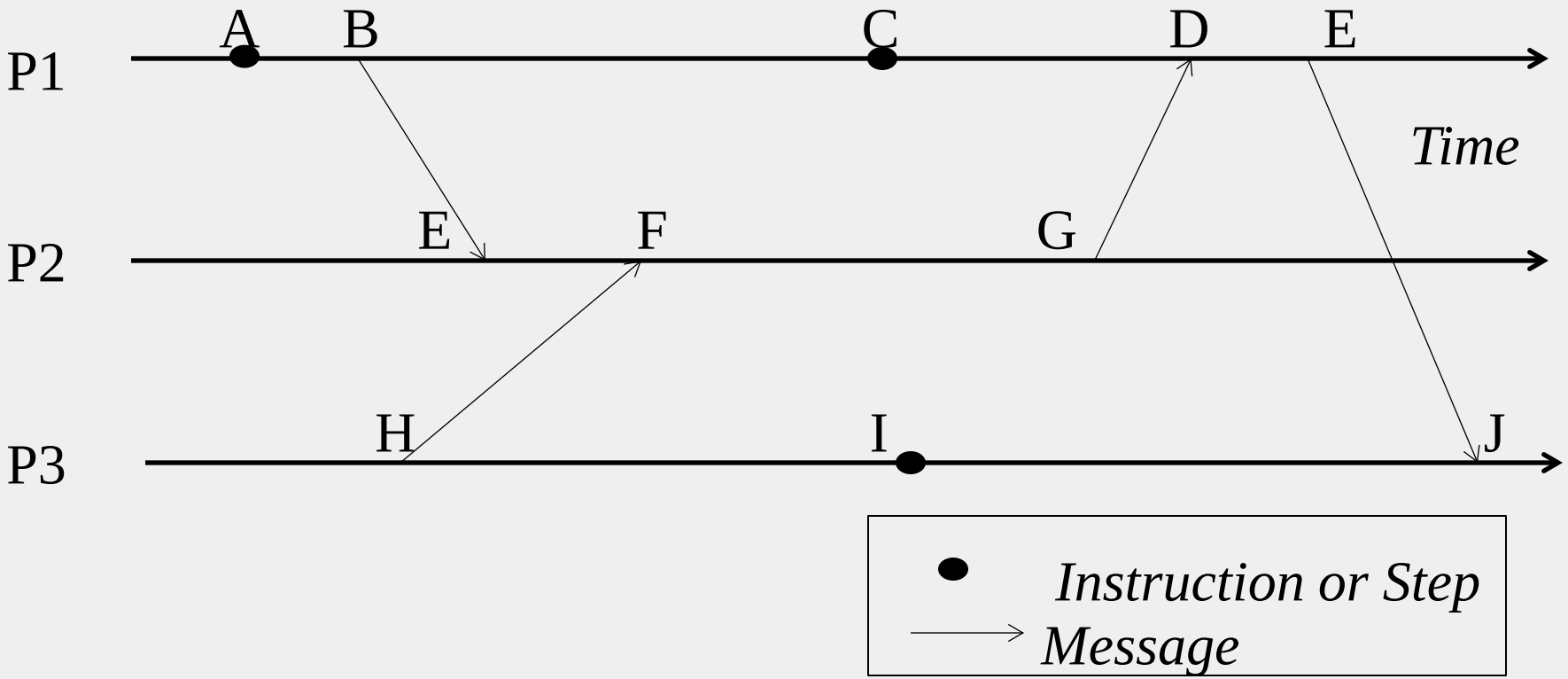
**Then, (if needed), a central server collects all these partial state pieces to obtain the full global snapshot**

intuition:

each ps needs 2 markers to terminate:
1st marker is to record state since that wont change
2nd marker is to record msg sent during global snapshot since likely to receive msgs from other ps during latencies

saved channel:
ever since P1 starts taking snapshots, it is going to take some time before that reaches every ps, so every msg sent during that latency is recorded as part of the save state

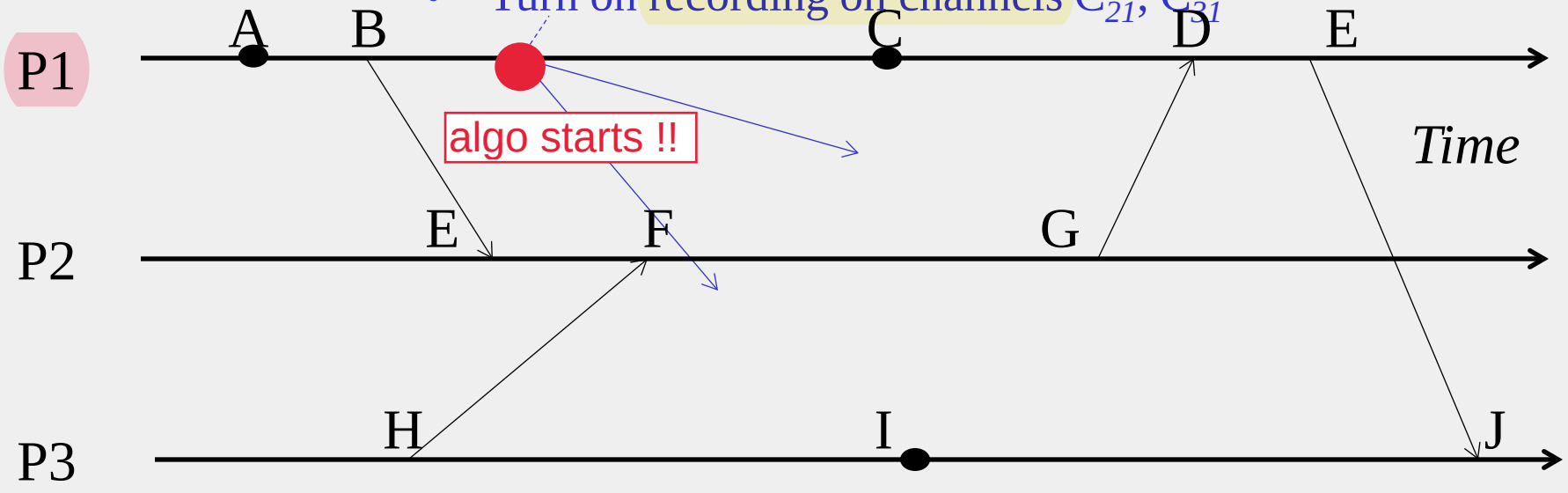
# Example

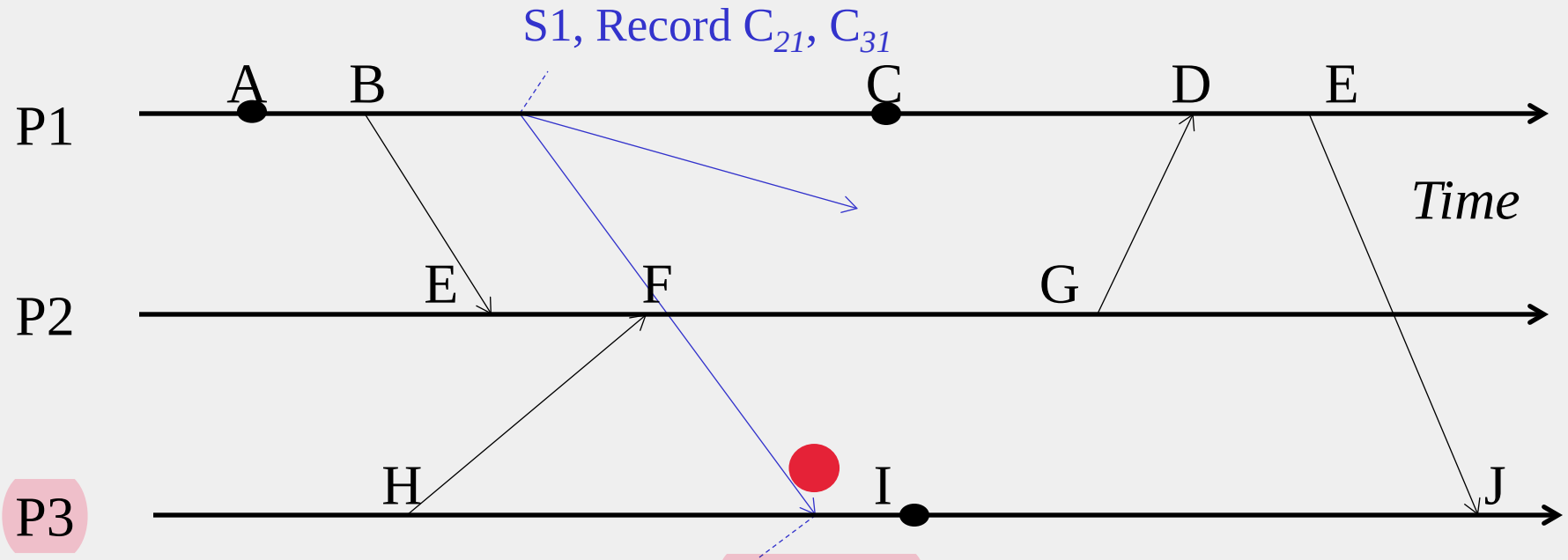


P1 is Initiator:

- Record local state S1
- Send out markers
- Turn on recording on channels $C_{21}$, $C_{31}$

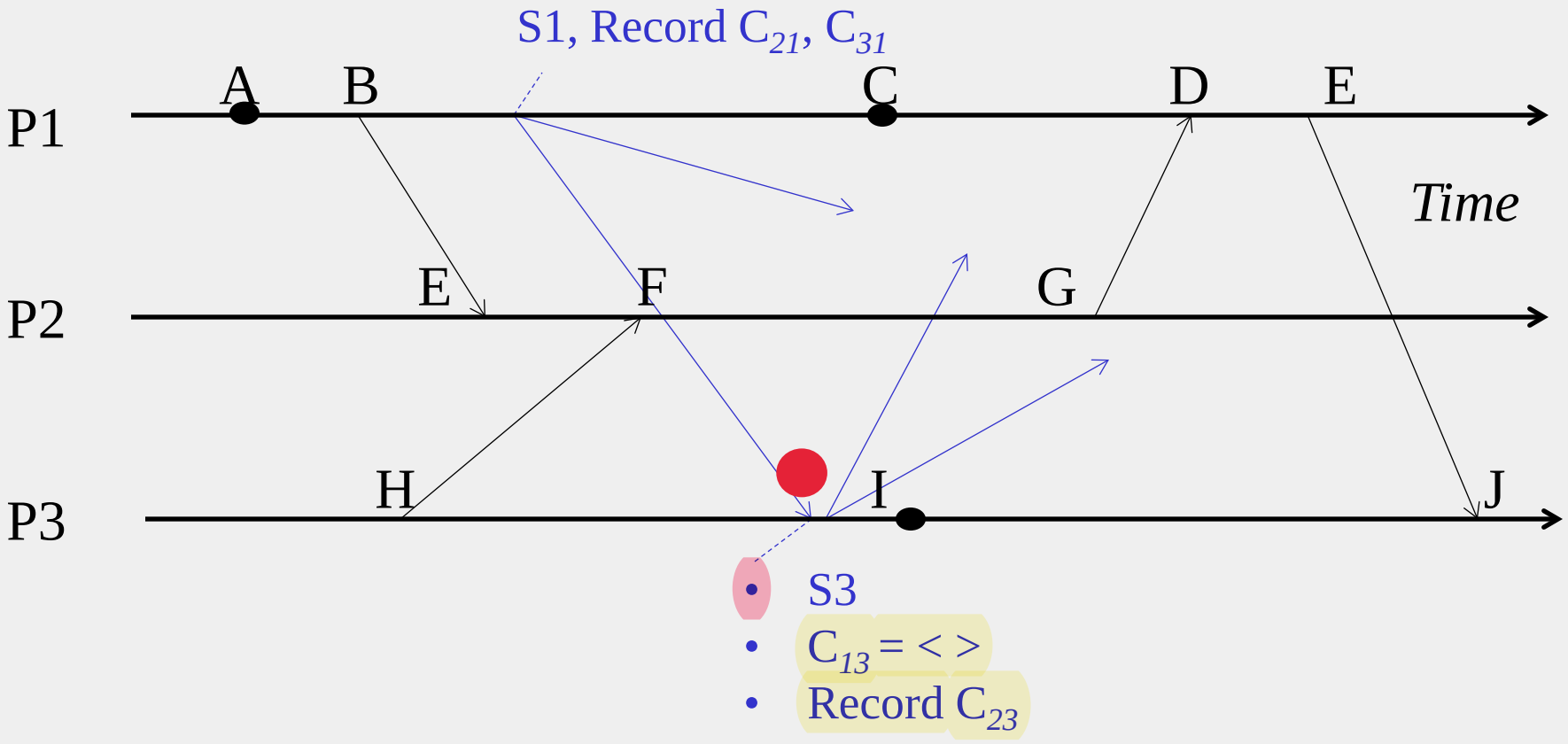
C21: P1's message queue to buffer P2's msgs






receiving marker BEFORE sending marker == "save" state

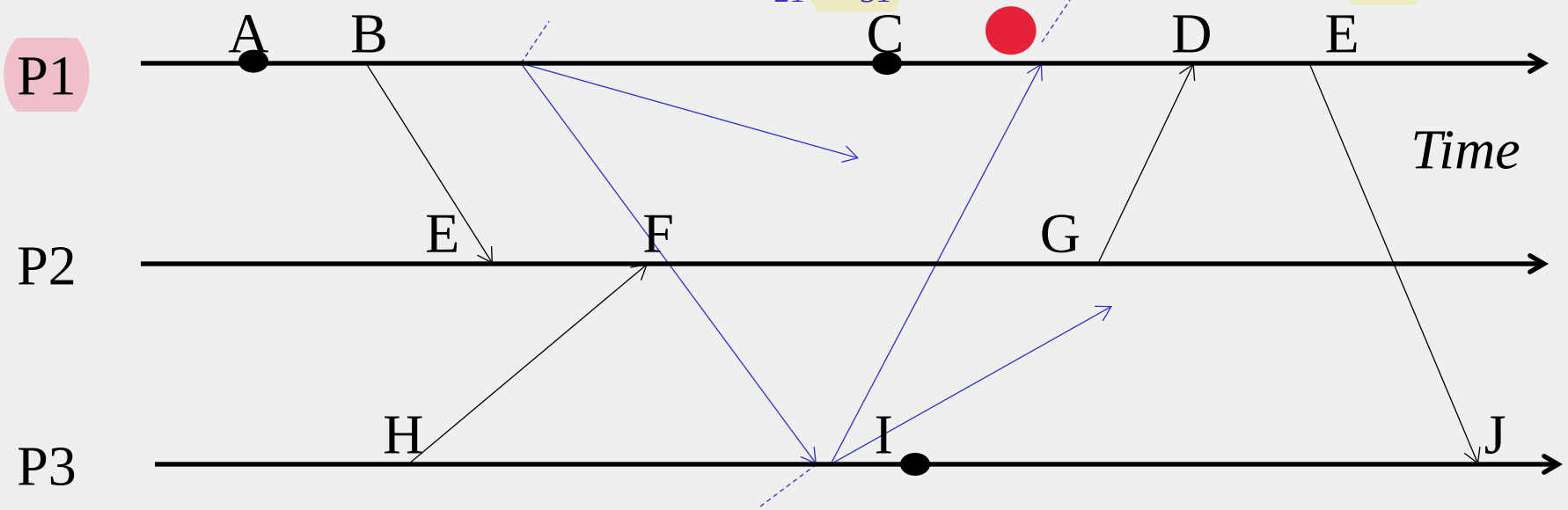
- First Marker!
- Record own state as S3
- Mark $C_{13}$ state as empty
- Turn on recording on other incoming $C_{23}$
- Send out Markers


S1, Record $C_{21}$, $C_{31}$
A
B
C
D
E
P1
Time
E
F
G
P2
H
I
J
P3
S3
$C_{13}$ = < >
Record $C_{23}$

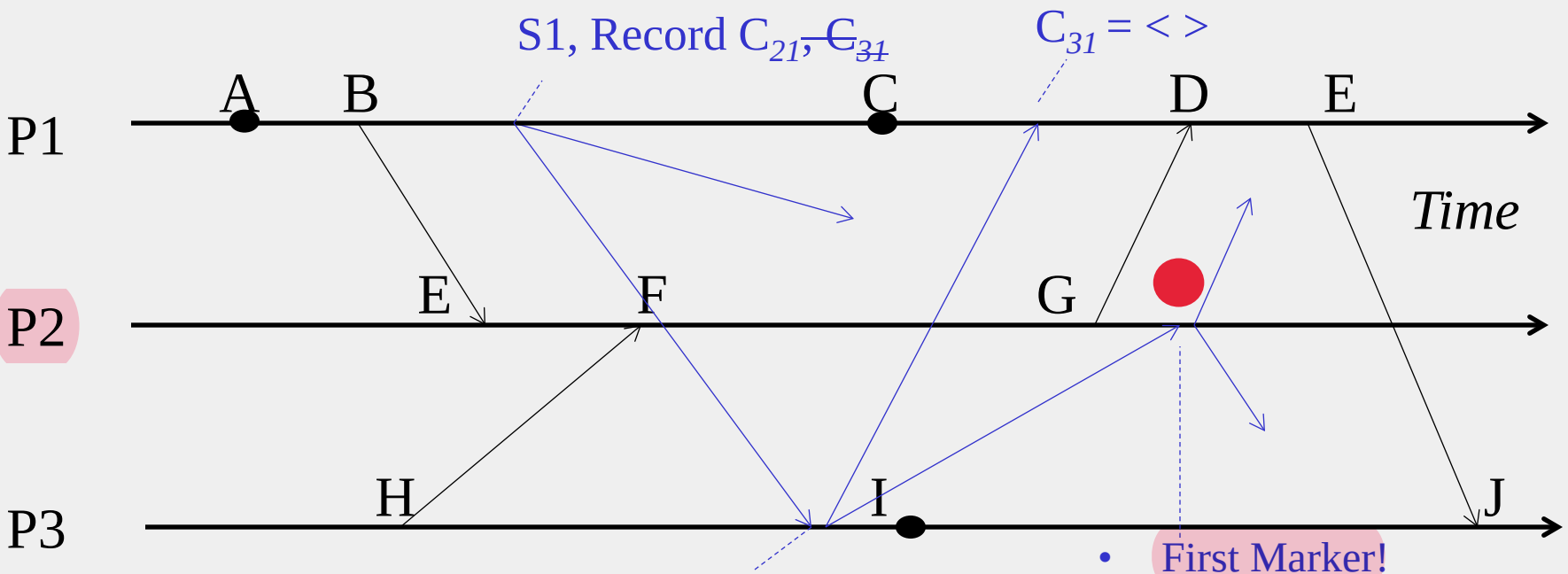
receiving marker AFTER sending
marker == "save" channel
i.e. recording state -> saved state

Duplicate Marker!

S1, Record $C_{21}$, ~~$C_{31}$~~ ----> State of channel $C_{31}$ = < >

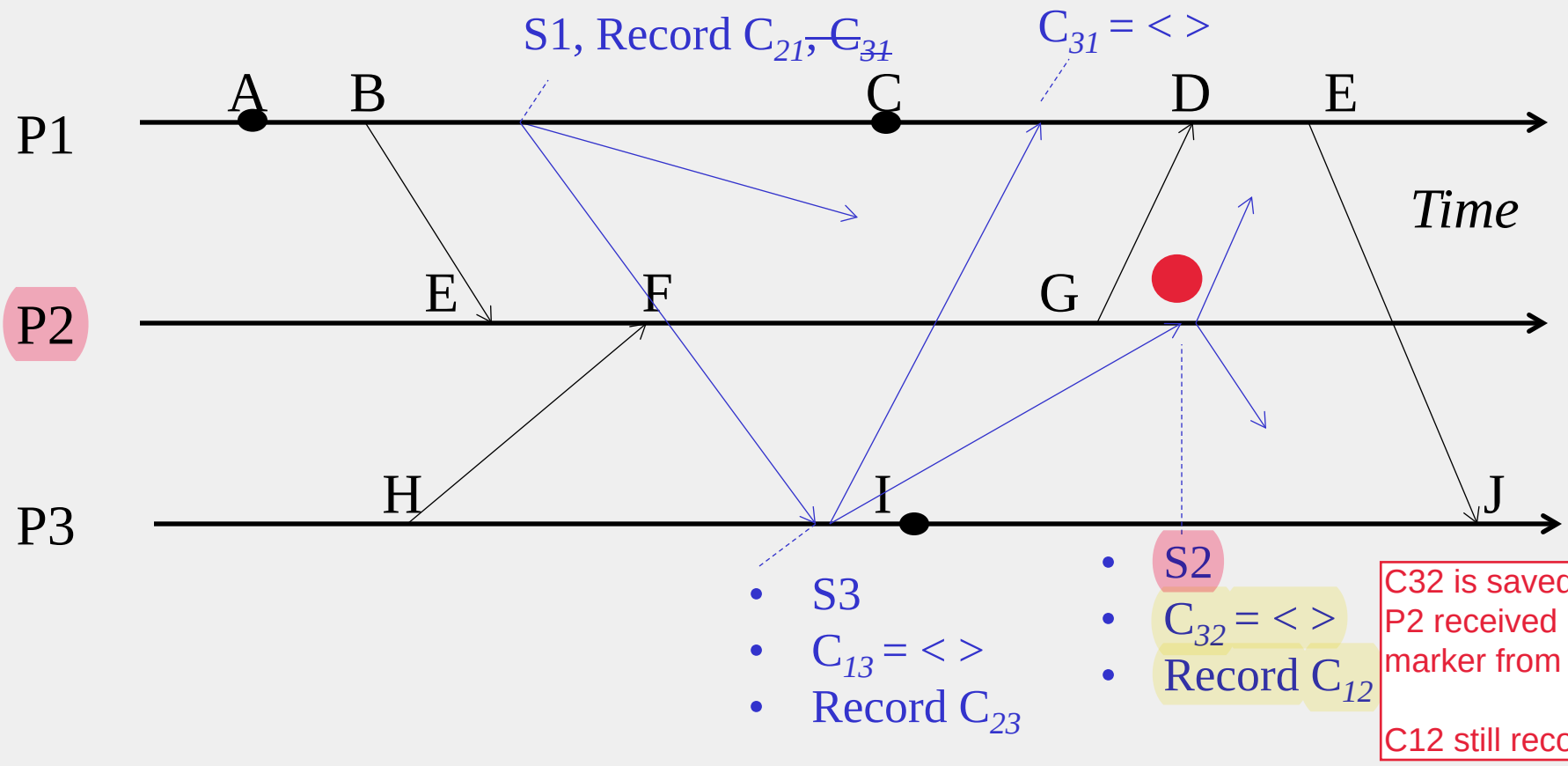


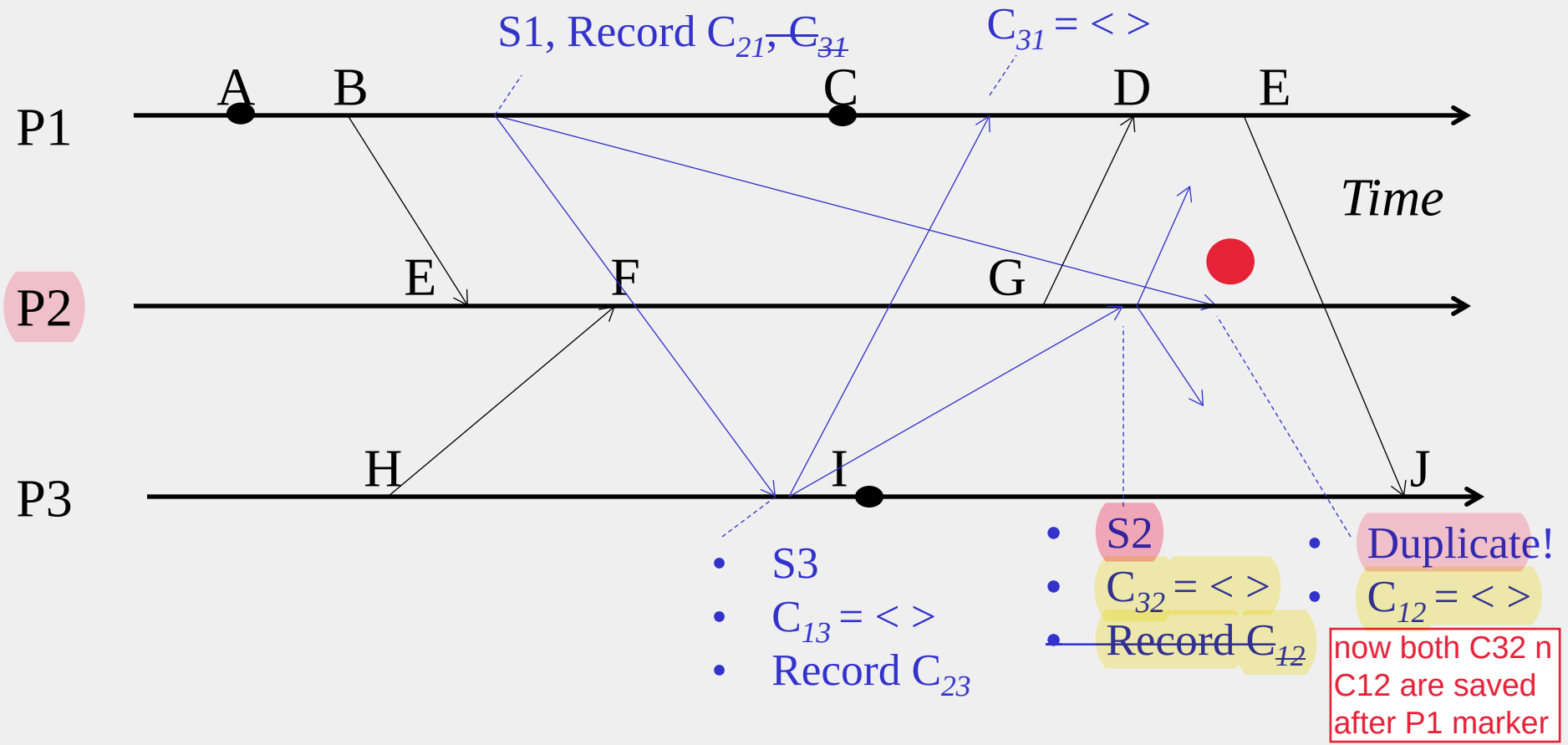

- S3
- $C_{13}$ = < >
- Record $C_{23}$



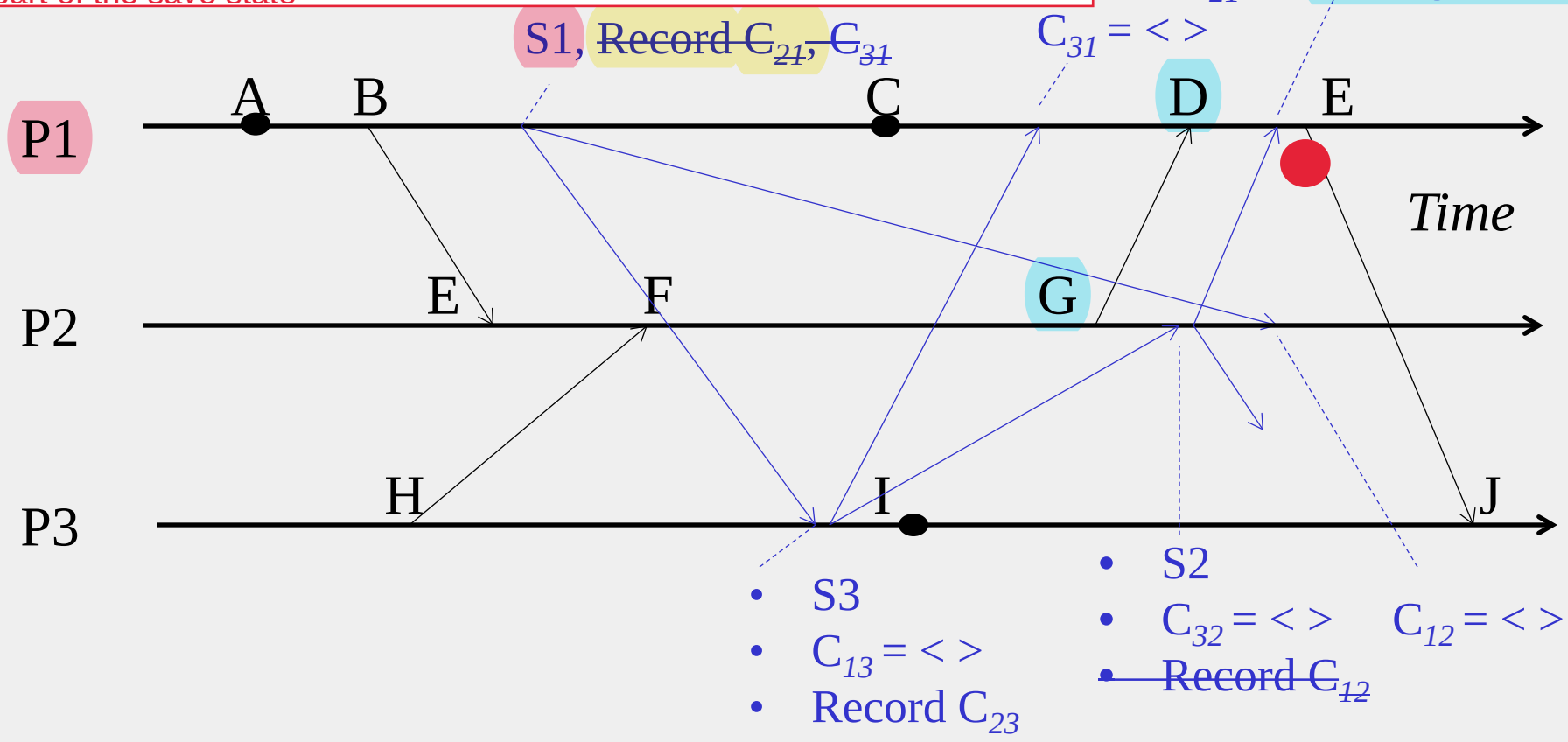

- S3
- $C_{13}$ = < >
- Record $C_{23}$

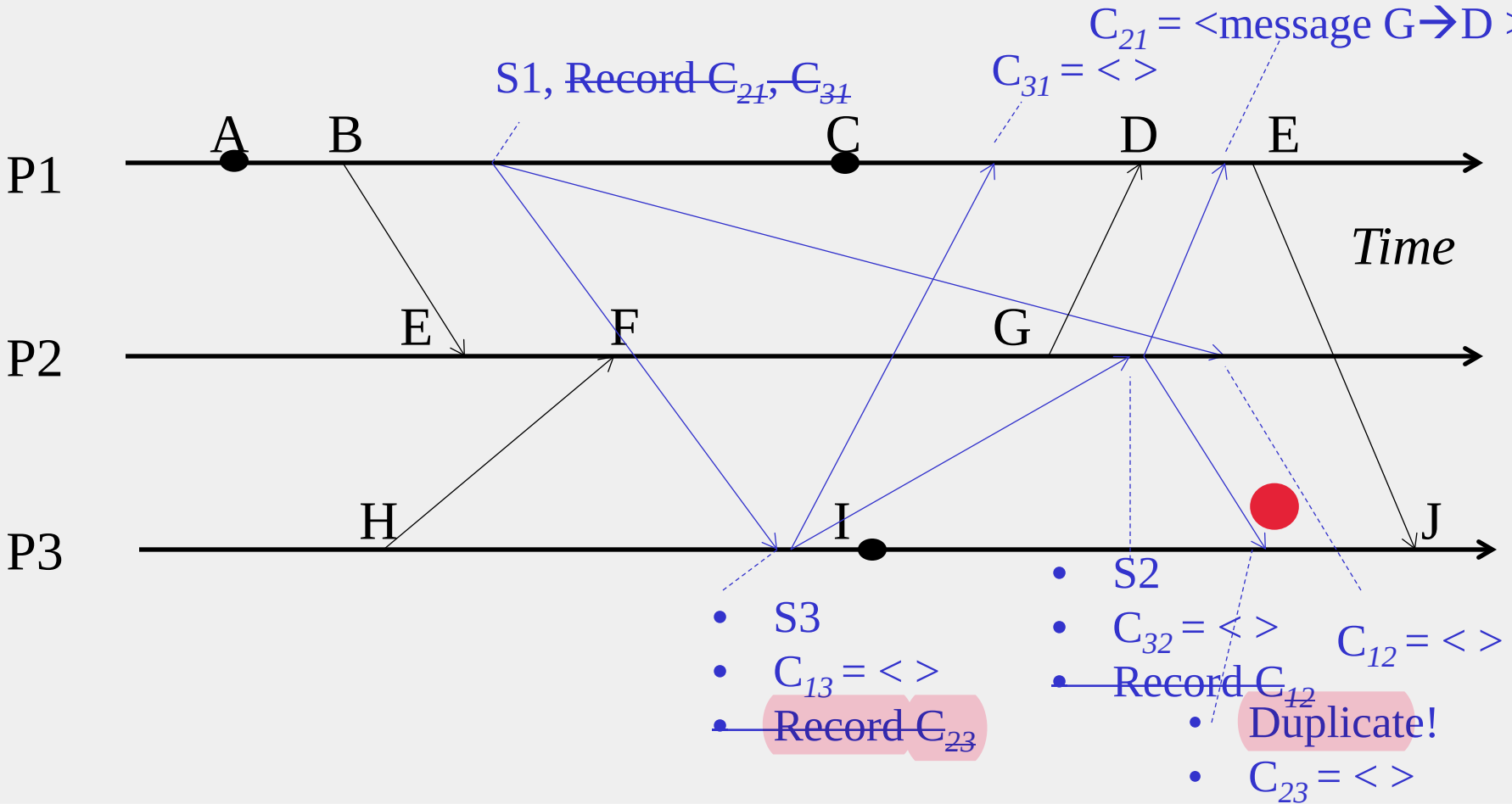
- First Marker!
- Record own state as S2
- Mark $C_{32}$ state as empty
- Turn on recording on $C_{12}$
- Send out Markers


S1, Record $C_{21}$, ~~$C_{31}$~~
$C_{31}$ = < >
A
B
C
D
E
P1
Time
E
F
G
P2
H
I
J
P3
• S3
• $C_{13}$ = < >
• Record $C_{23}$
• S2
• $C_{32}$ = < >
• Record $C_{12}$
C32 is saved as P2 received marker from P3
C12 still recording

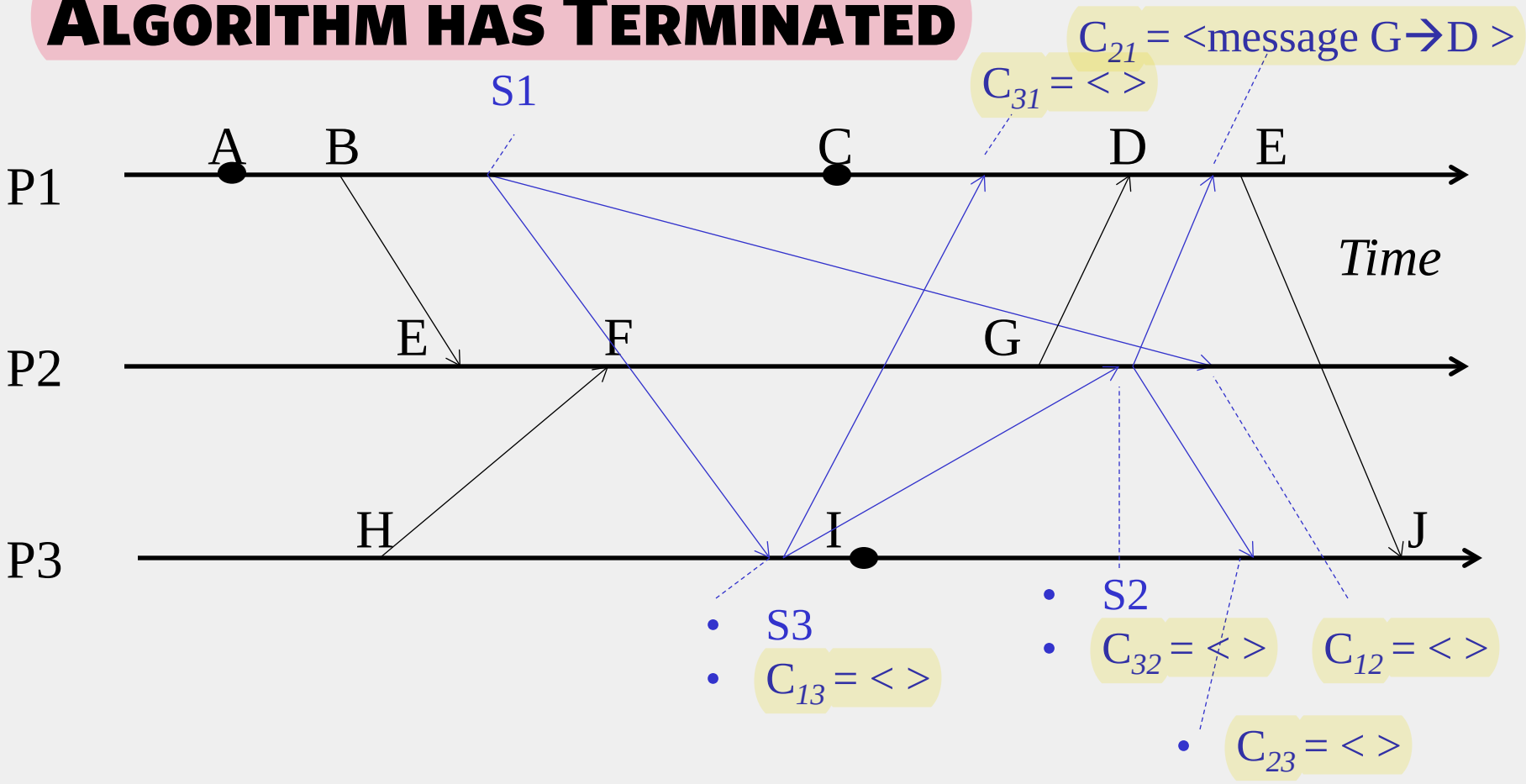

S1, Record $C_{21}$, ~~$C_{31}$~~
$C_{31}$ = < >
A
B
C
D
E
P1
Time
E
F
G
P2
H
I
J
P3
• S3
• $C_{13}$ = < >
• Record $C_{23}$
• S2
• $C_{32}$ = < >
• ~~Record $C_{12}$~~
• Duplicate!
• $C_{12}$ = < >
now both C32 n C12 are saved after P1 marker

intuition:
ever since P1 starts taking snapshots, it is going to take some time before that reaches every ps, so every msg sent during that latency is recorded as part of the save state

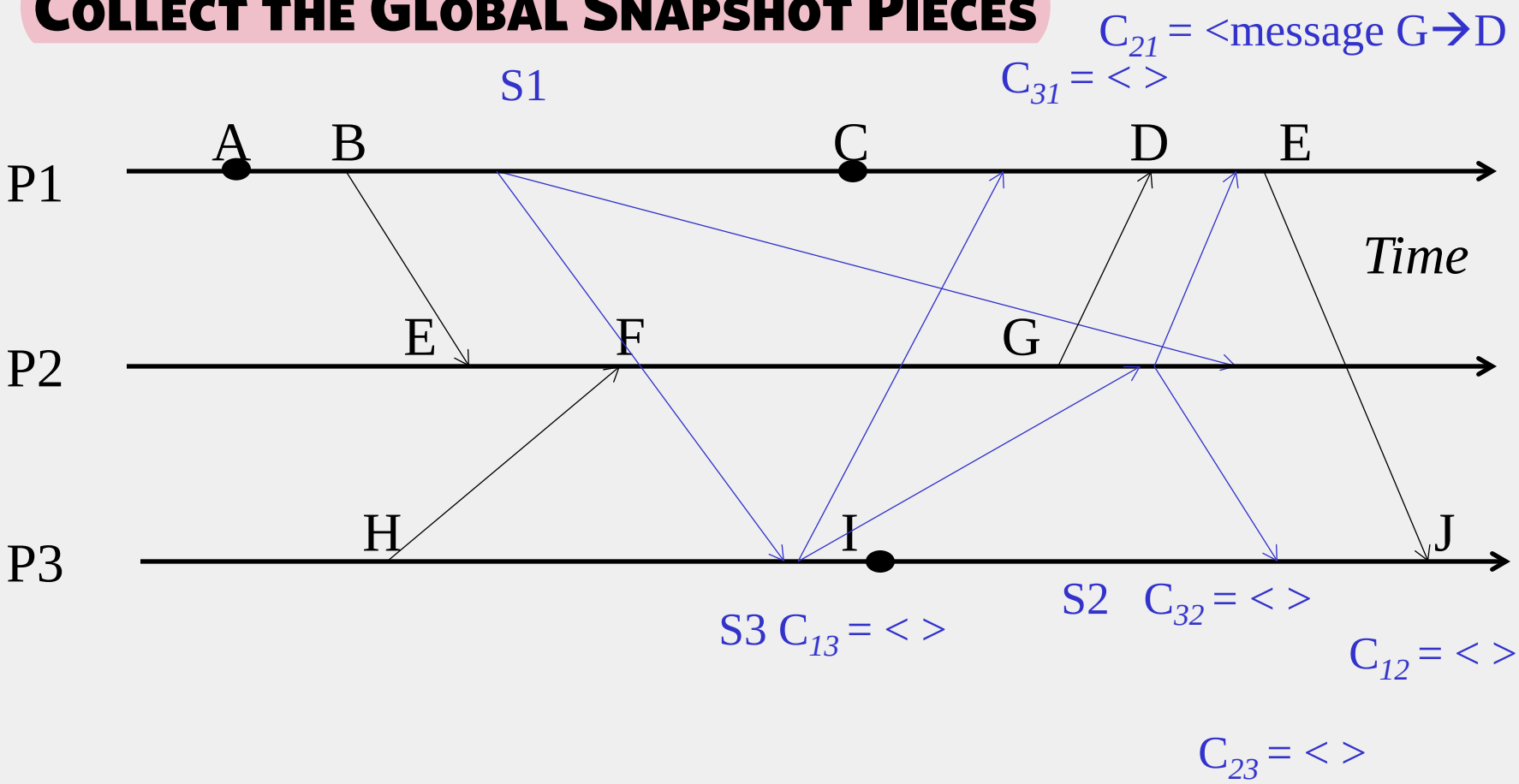



$C_{21}$ = <message G→D >
S1, ~~Record $C_{21}$, $C_{31}$~~
$C_{31}$ = < >
A
B
C
D
E
P1
Time
E
F
G
P2
H
I
J
P3
• S3
• $C_{13}$ = < >
• ~~Record $C_{23}$~~
• S2
• $C_{32}$ = < >
• ~~Record $C_{12}$~~
$C_{12}$ = < >
• Duplicate!
• $C_{23}$ = < >

# Algorithm has Terminated



# Collect the Global Snapshot Pieces



# Next

- **Global Snapshot calculated by Chandy-Lamport algorithm is causally correct**
  - What?