



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:*
 - N processes in the system
 - There are two uni-directional communication channels between each ordered process pair : $P_j \rightarrow P_i$ and $P_i \rightarrow P_j$
 - 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)

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}**

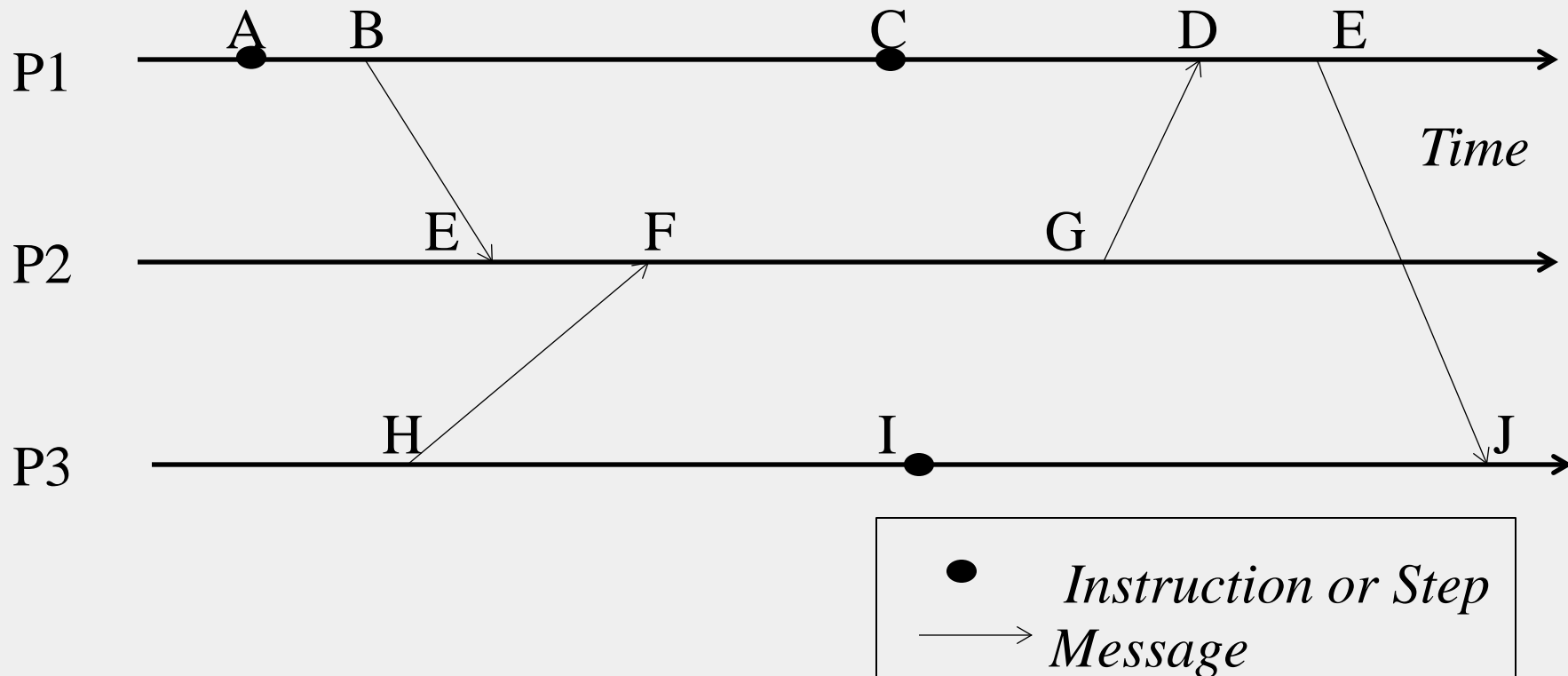
CHANDY-LAMPORT GLOBAL SNAPSHOT ALGORITHM (3)

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

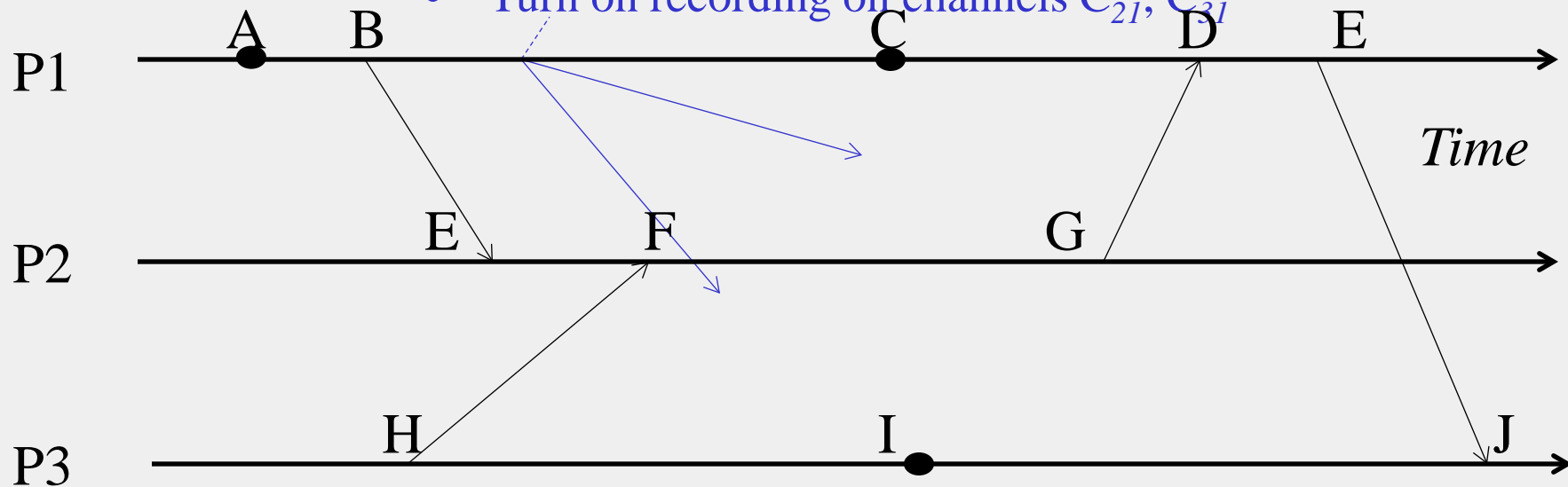
Then, (if needed), a central server collects all these partial state pieces to obtain the full global snapshot

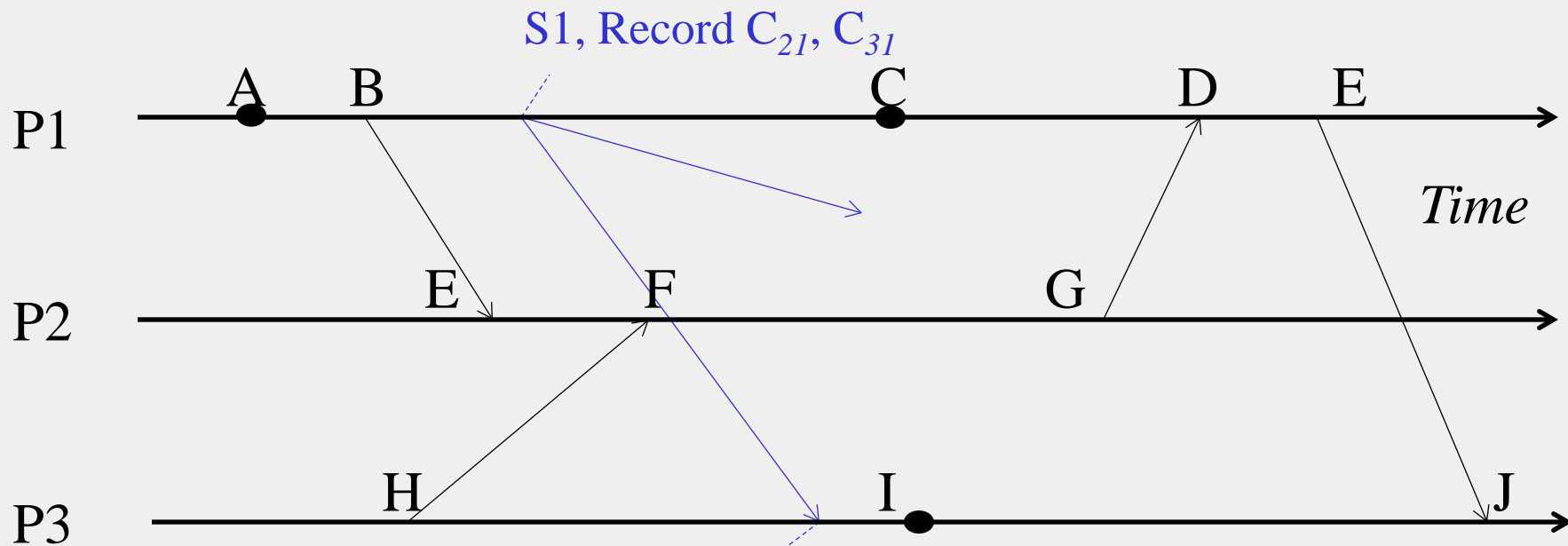
EXAMPLE



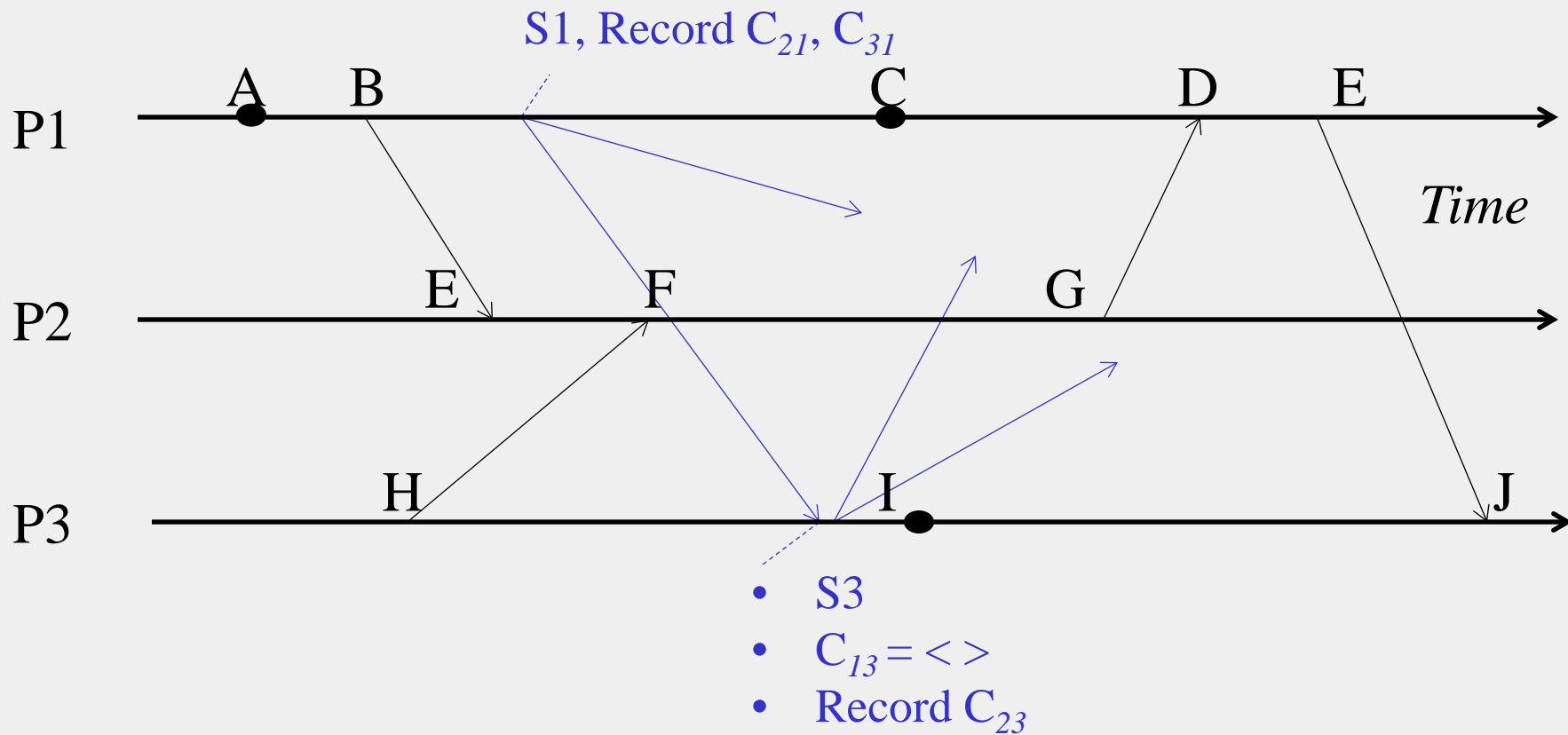
P1 is Initiator:

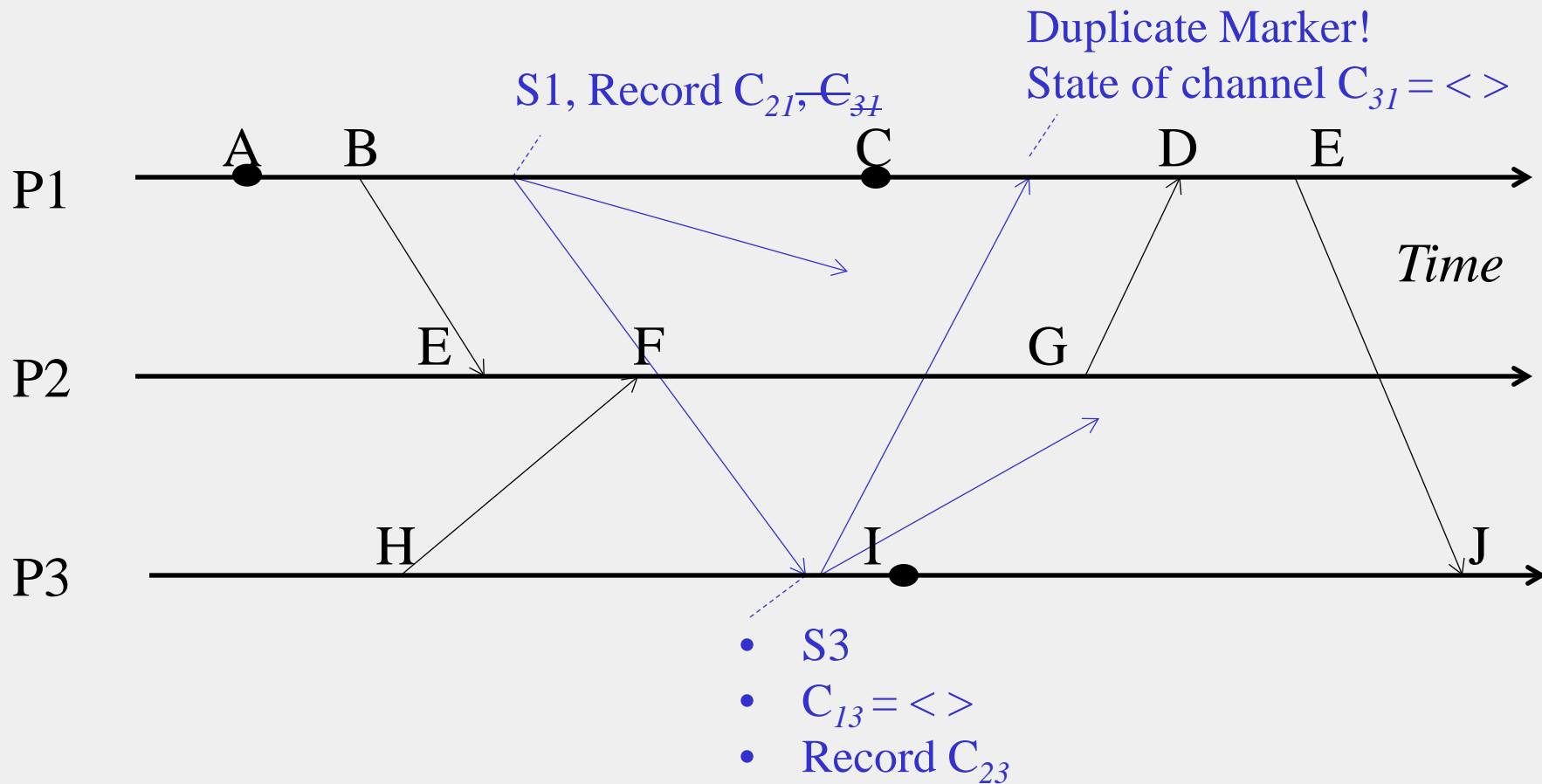
- Record local state S1
- Send out markers
- Turn on recording on channels C_{21} , C_{31}

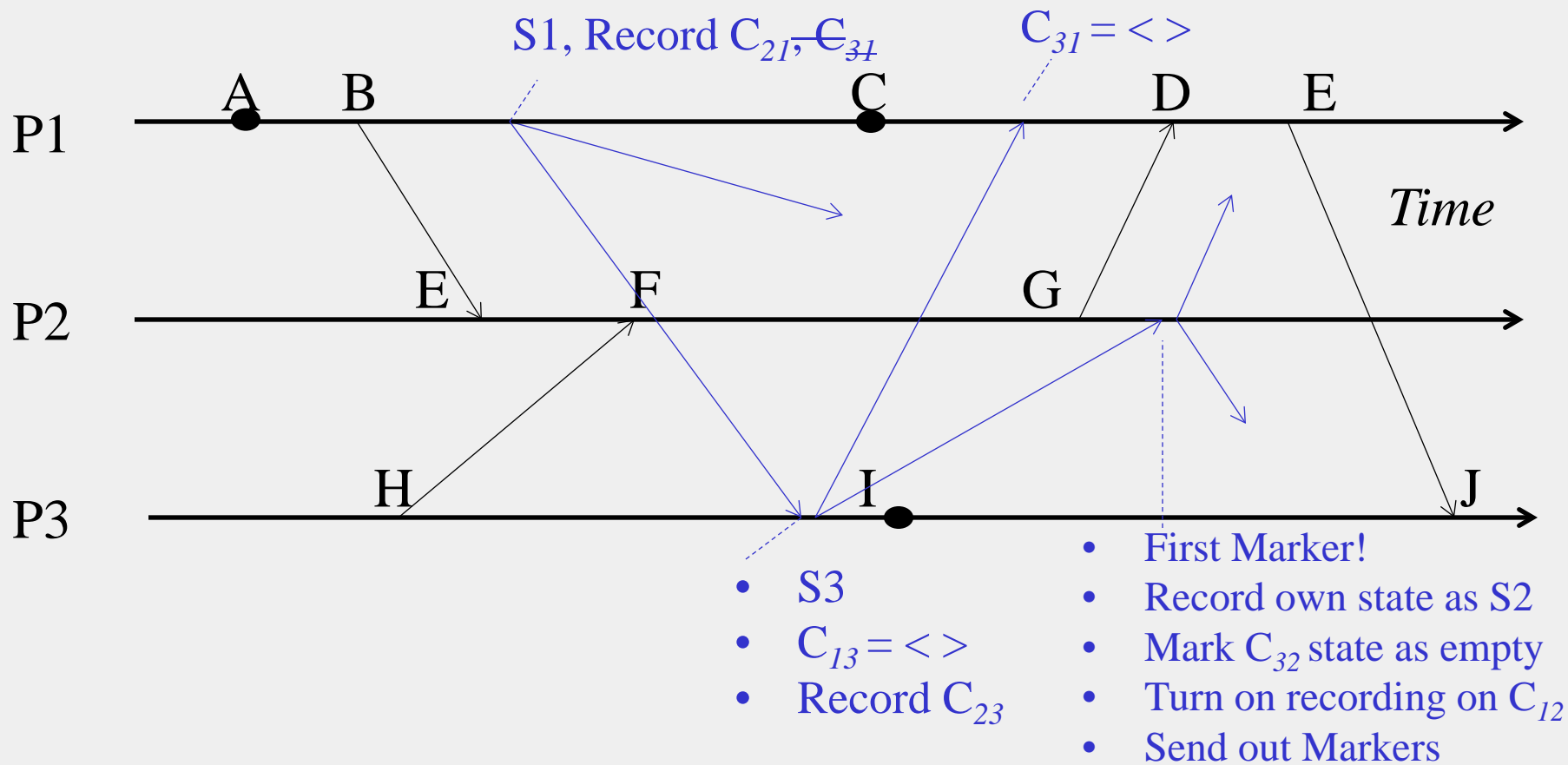


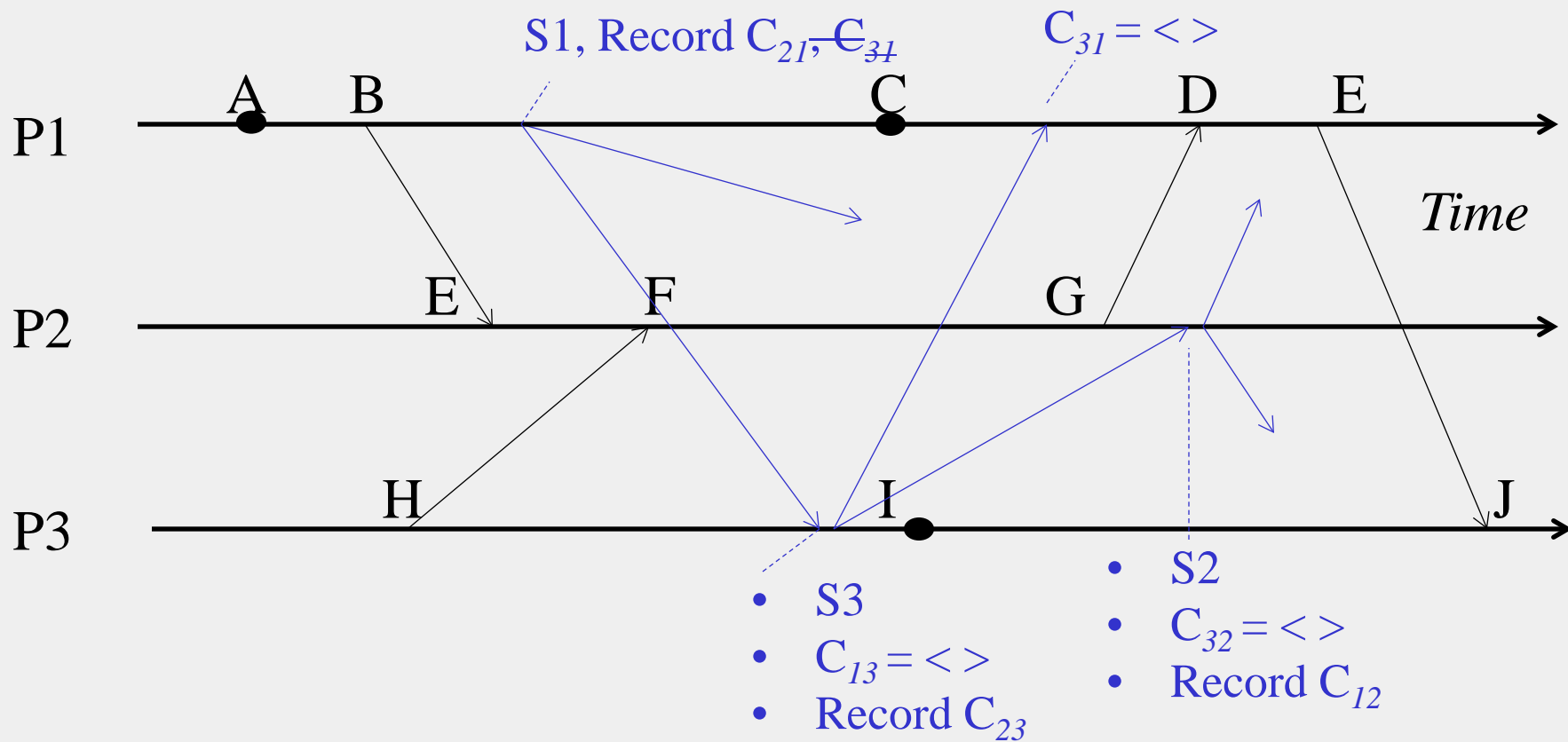


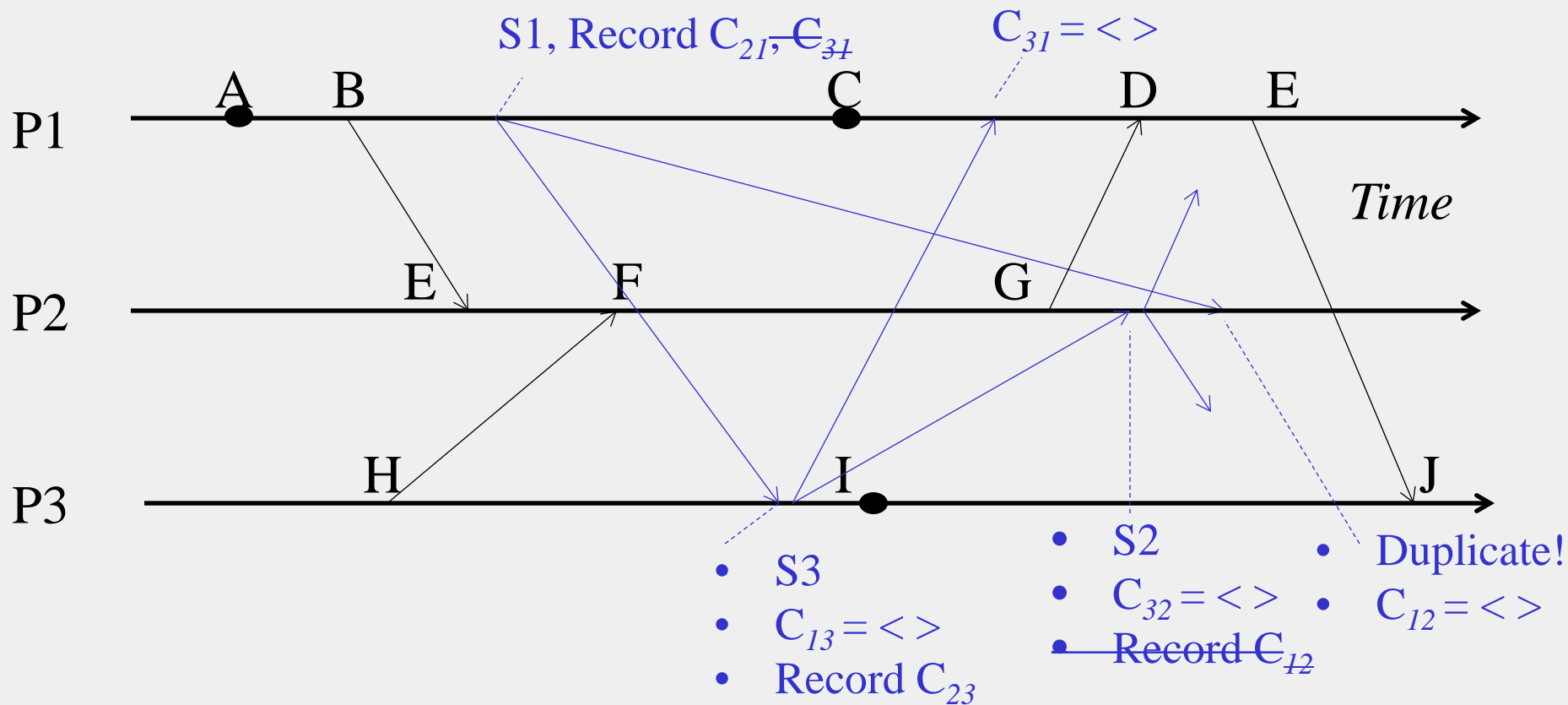
- First Marker!
- Record own state as S3
- Mark C_{13} state as empty
- Turn on recording on other incoming C_{23}
- Send out Markers

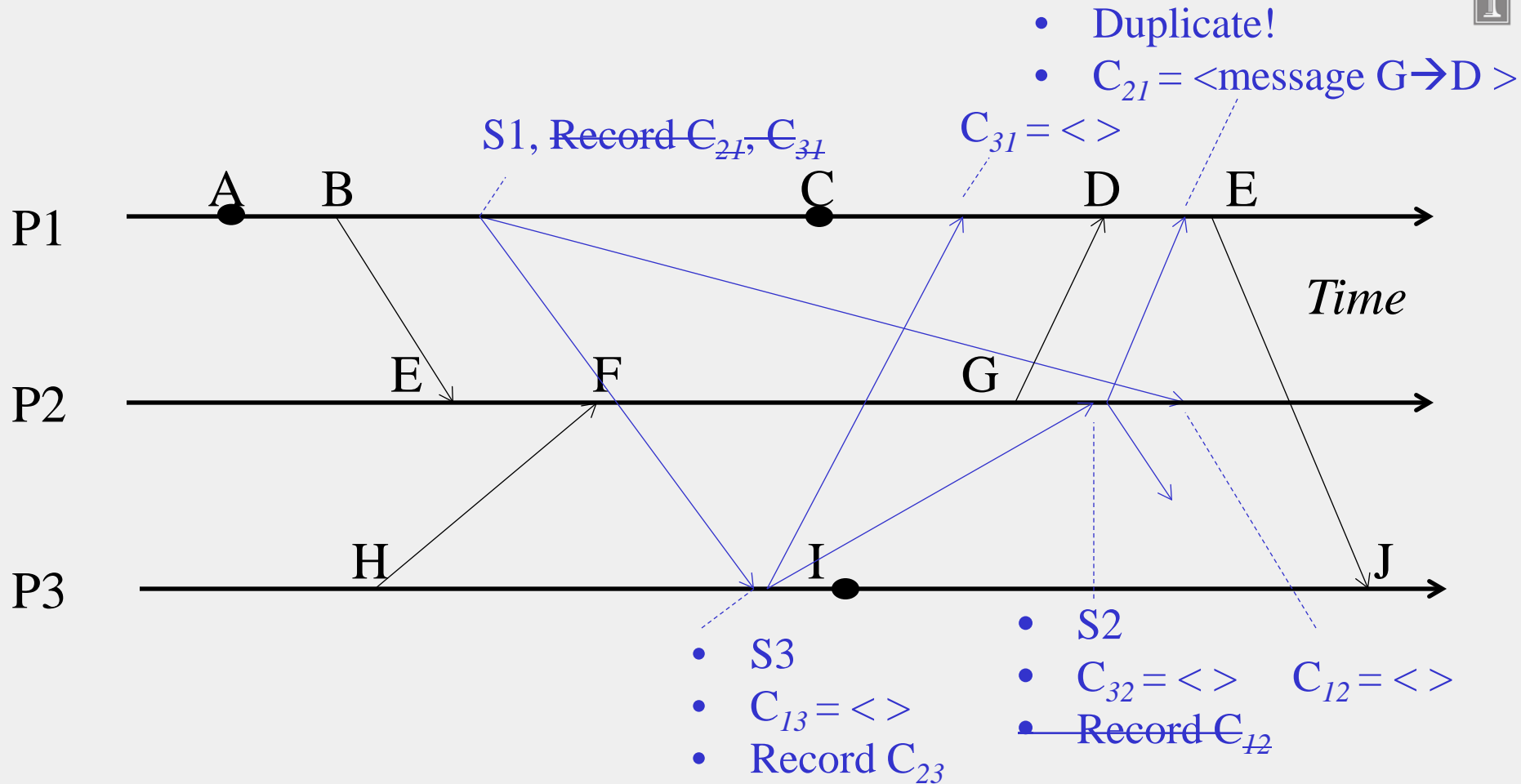


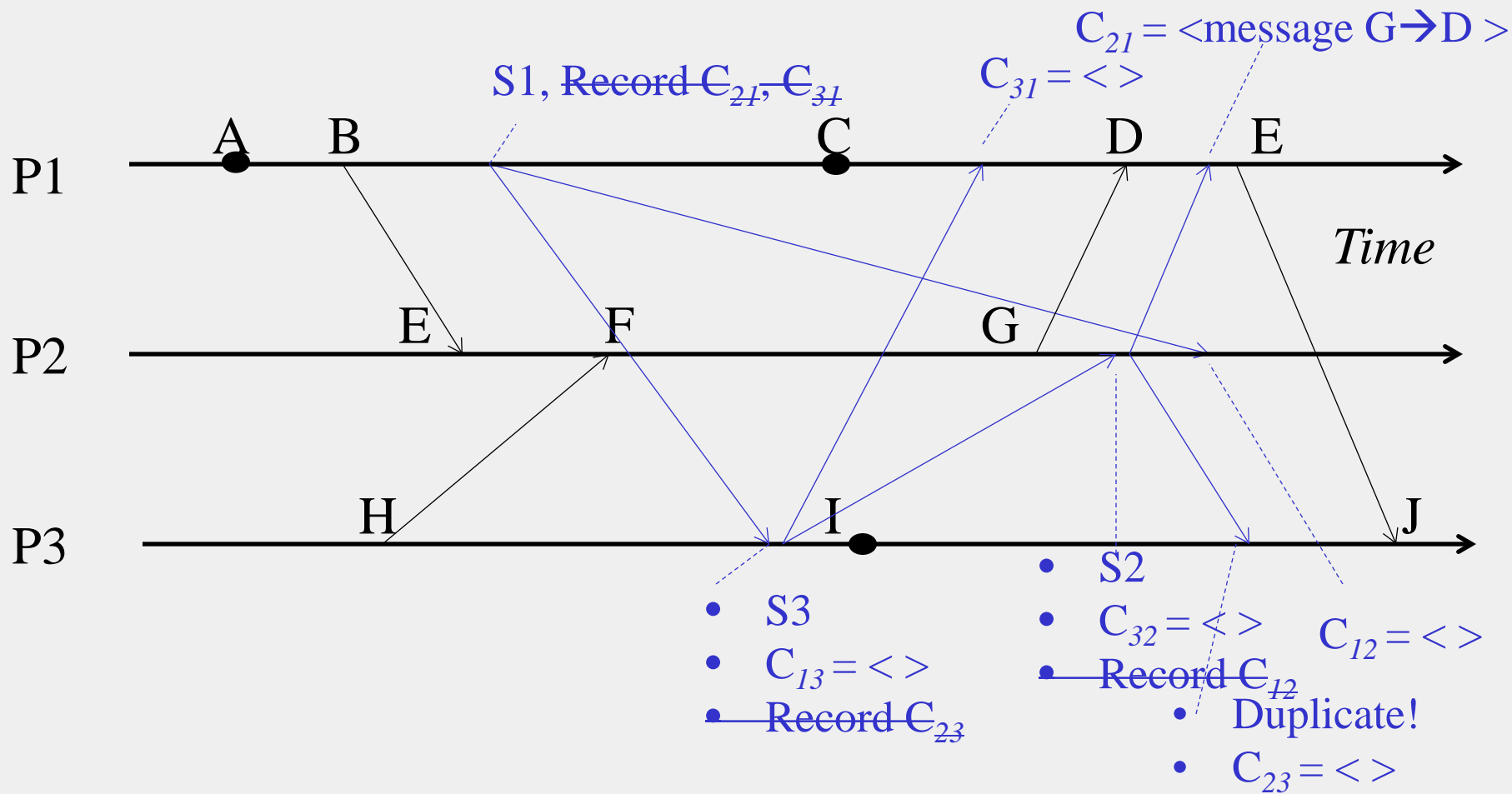




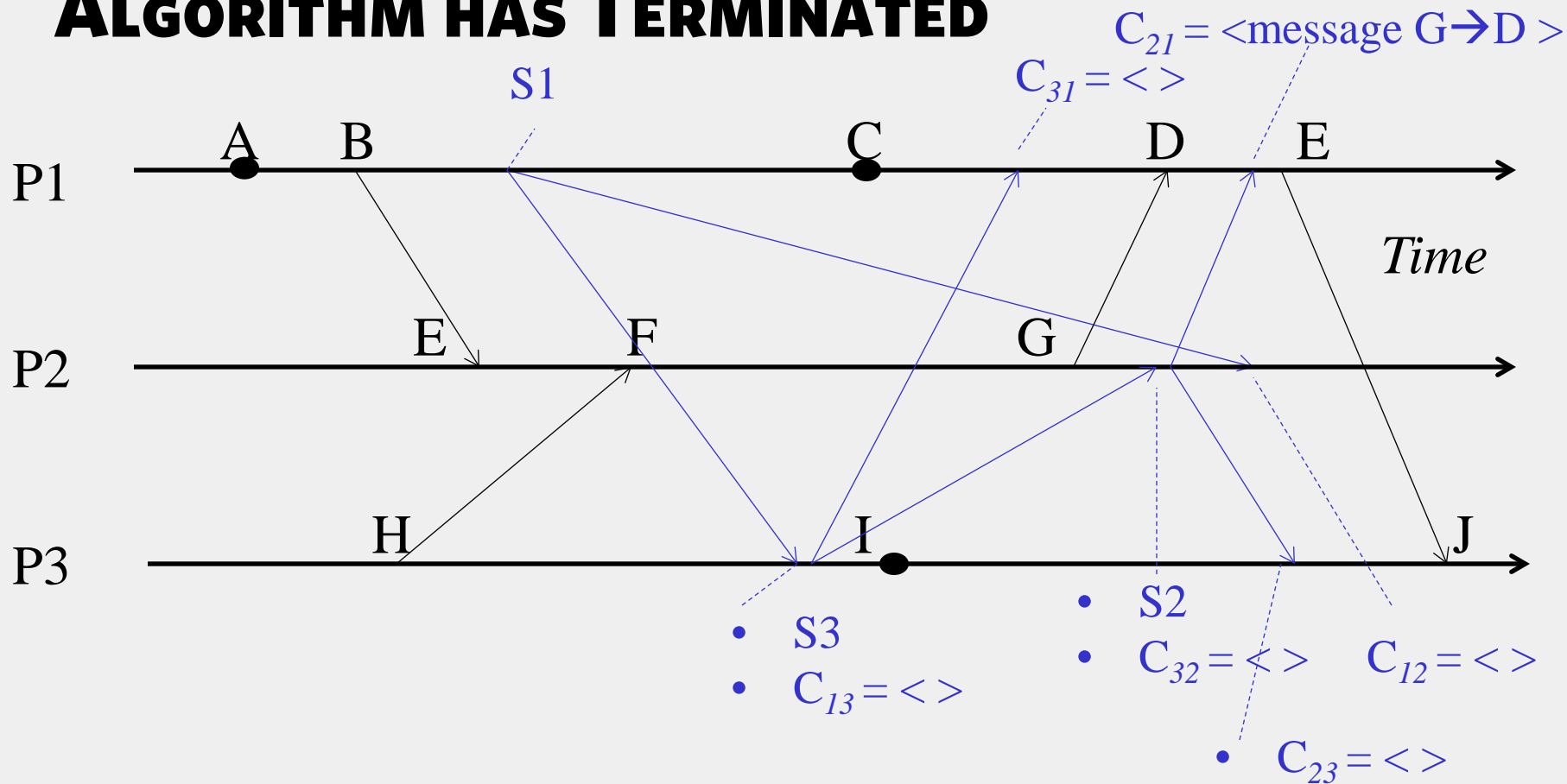




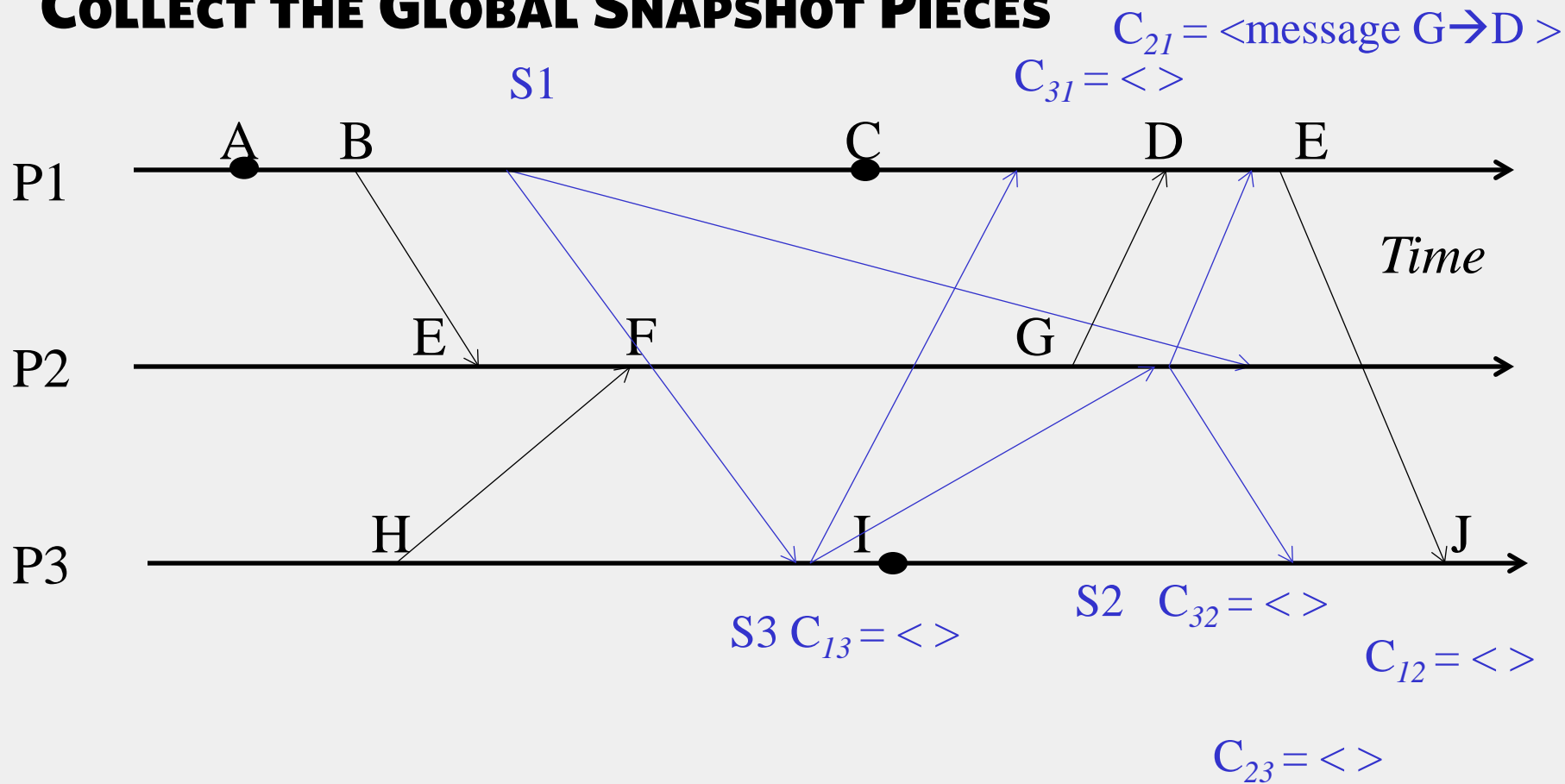




ALGORITHM HAS TERMINATED



COLLECT THE GLOBAL SNAPSHOT PIECES



NEXT

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