

lect

It don't help to eliminate Total anomaly,
it's just an algo for screenshot ~~to~~ at any
moment that can help to debug etc.


Chandy Lamport Algorithm

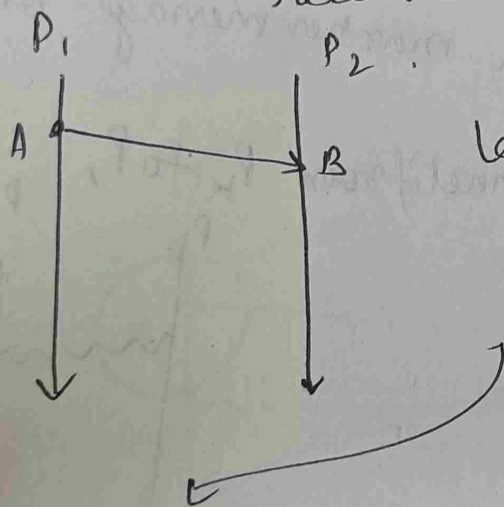
- It is an algorithm for taking a Snapshot of the state of a distributed system
- and it works by having every process snapshot of its own states in a way that can be put together in a way that makes sense.
- Interesting thing about the Chandy Lamport Algorithm is that any process can start it, it doesn't have to be like a one designated process that can only trigger.
- Any process can begin it and that process does not have to announce to everybody, "okay, I am going to start the Snapshot", it can just go ahead and start because it turns out that there's no problem if two processes happen to start taking a snapshot at the same time without coordinating with each other, that turns out not to cause a problem for the algorithm.

Chandy Lamport Algorithm

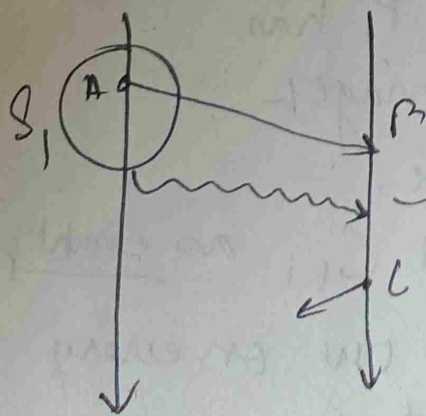
→ we are going to say that one particular process kicks things off and we will call it the "initiator process"

The Initiator process:

- Records its own state.
(right after that)
- Send a marked message  out on all its outgoing channels.
- Starts recording the messages it receives on all its incoming channels.



lets say P_1 is the initiator and it takes the Snapshot after event A.



Sends markers and starts recording on all its incoming channels.
(for this scenario P_1 has only one incoming channel that is from process P_2)
(SC₂₁)

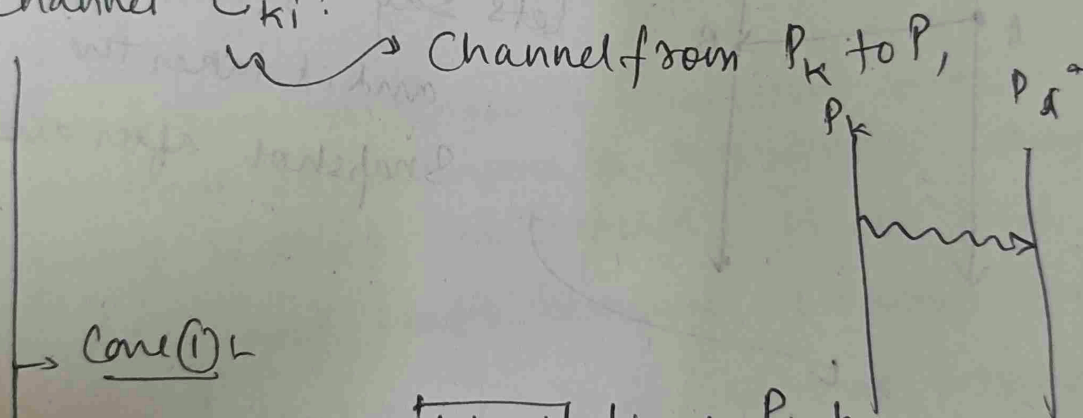
So, process 1 (P_1) is recording on C_{21}

Note:- Snapshot captures all the events that have happened on that processes so far ever.
(In other words, it's the current state of the process)

Q) What happens when someone receives a "marker message"?

→ Receiving a marker message:-

• When a process P_i get a marker message on channel C_{ki} :



If it's the first time P_i has seen a marker message:-

- P_i records its state.
- P_i marks channel C_{ki} as empty.
- P_i sends a marker out on every outgoing channel C_{ij} .
- P_i start recording incoming messages on all its incoming channels except C_{ki}

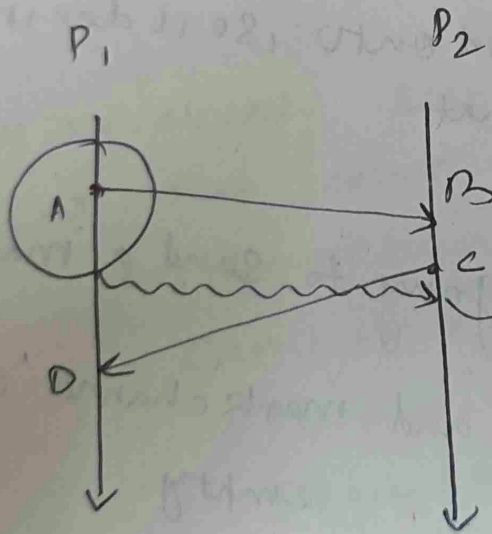
what counts as seeing a marker message?

→ If you have anything to a marker message then you have seen a marker message.

→ So even sending a marker msg counts as seeing a marker message.

→ Case (11) : If P_i has already seen a marker:

- P_i stops recording on C_{ki} ,
- and it sets C_{ki} 's final state as the sequence of all the incoming messages that arrived on C_{ki} since recording began.

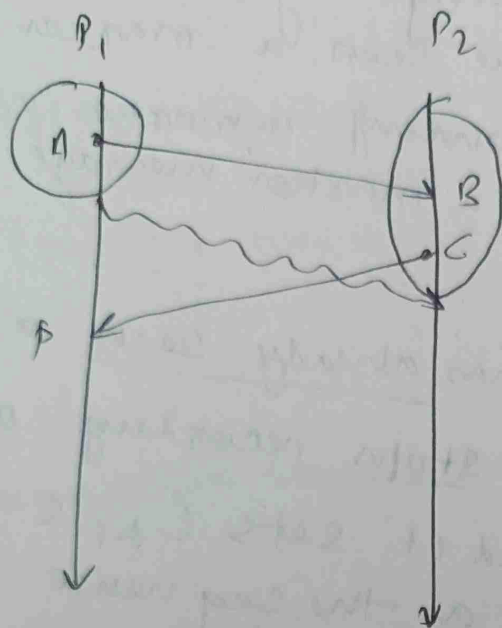


when P_2 receives the marker msg

P_1 records its state → PTO

→ P2

It records its state.



recorded its state.

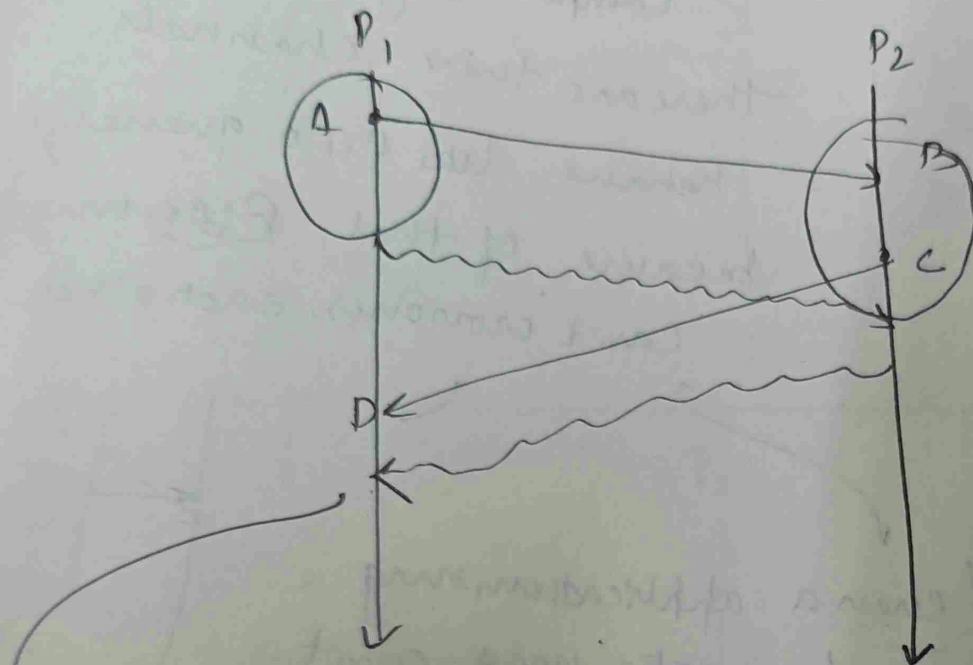
and the
marks the channel,
from which
marker msg came
as empty.

and as P2 has no incoming channels.
to record onto, so it doesn't have to
do that

[So, its goes going to send a marker back
to P1, and mark channel C_{12} as
empty.]

Now as P_1 was recording on its incoming channels $\rightarrow C_{21}$

and P_2 is going to set C_{12} as empty.

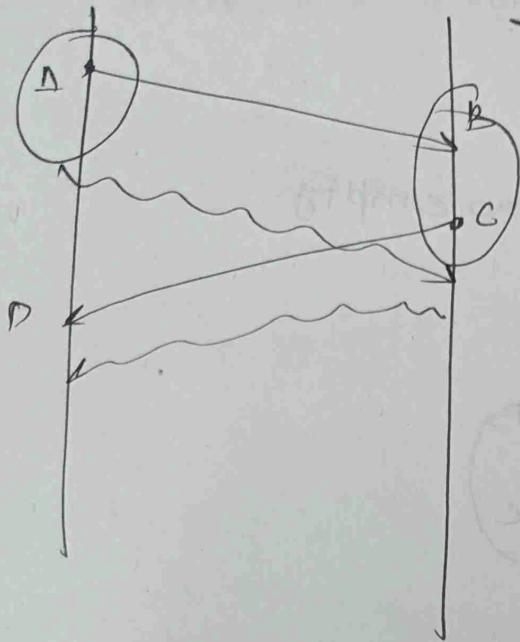


This is the "already seen" marker case for P_1 , at that point P_1 stops recording on the channel that marker ms came on on and it sets that channel final state as the sequence of all the incoming messages that arrived there since recording began.

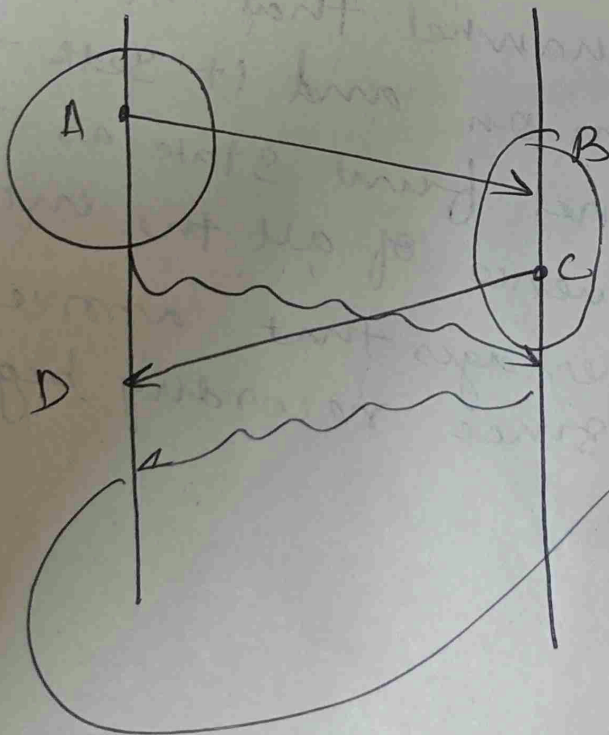
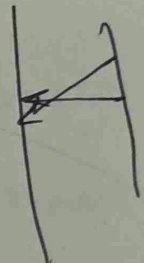
Kya D marker msg ke baad aa sakta hai

→ One interesting thing to point out is that remember how we said that in the Chandy Lamport algorithm,

there are two channels behave like FIFO queues; because of that ~~Msgs~~ msgs can't cross over each other.

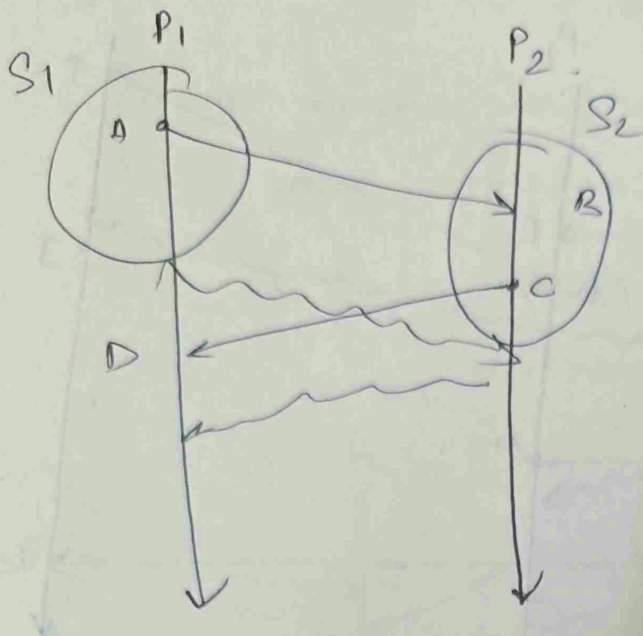


(even a application msg & marker msg can't cross in a channel)



Ye msg aate hi channel C_{21} ko P_1 recording rok deta hai aur,

C_{21} ke corresponding msg record hua $C_{21}D$

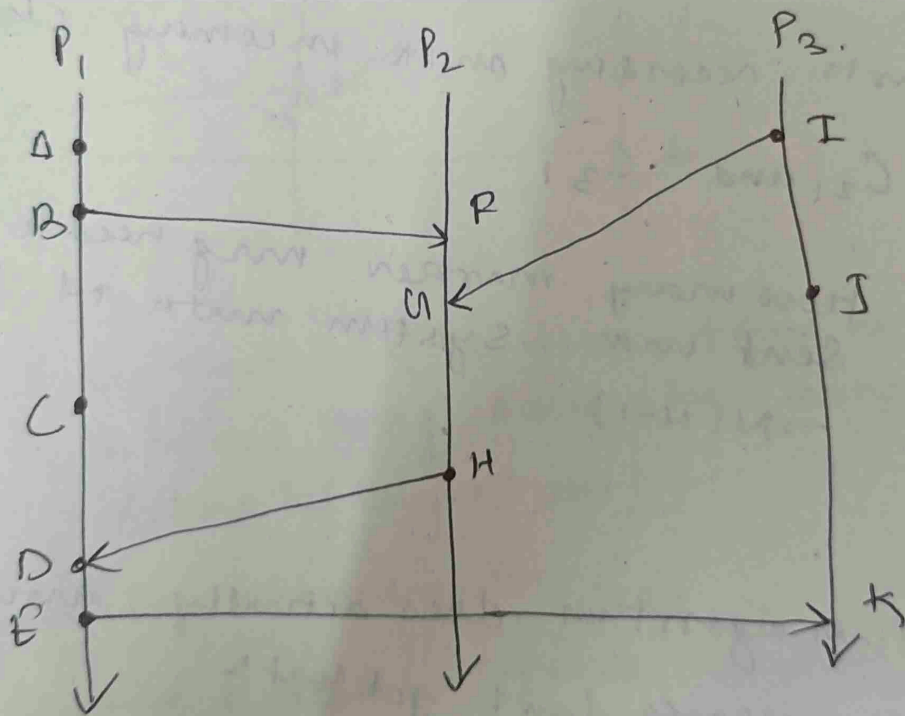


$$C_{21} \rightarrow \text{C2P}$$

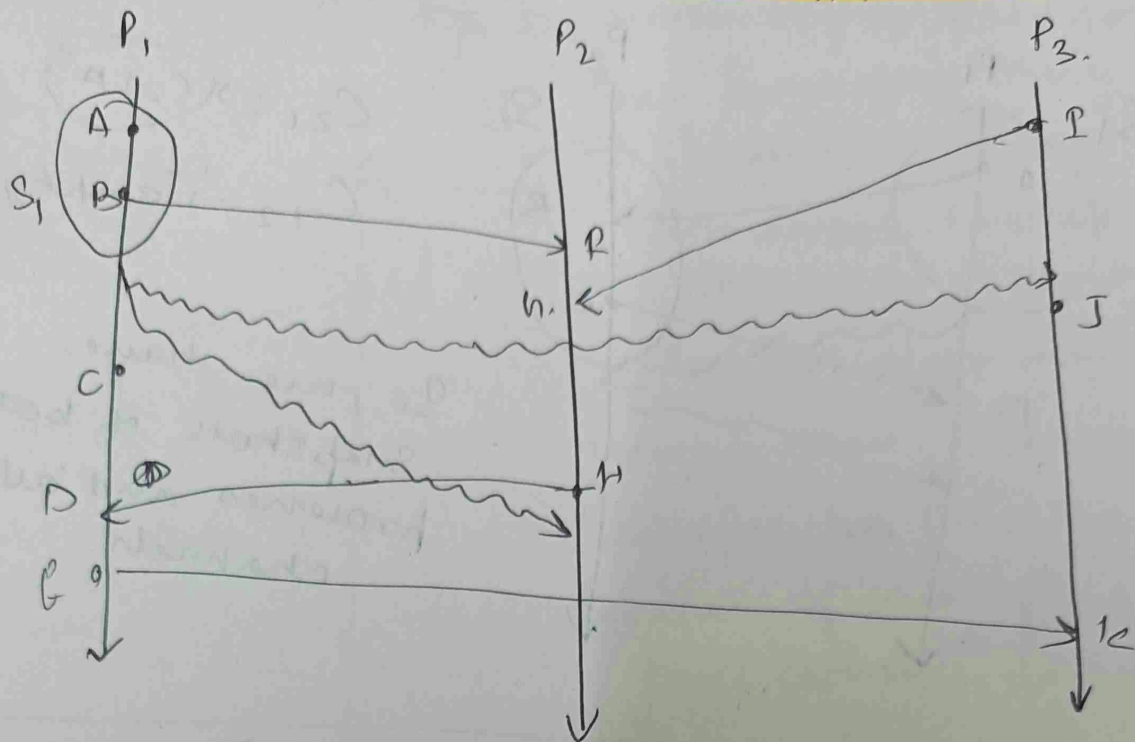
$$C_{12} \rightarrow \text{empty}$$

So, we have.
Snapshots of both
processes and all
channels.

(eg 2)



Let's say problem is the initiator.



P_1 starts recording on its incoming channels.

C_{21} and C_{31} .

How many markers may need to be sent in a system with n processes
 $\rightarrow n(n-1)$

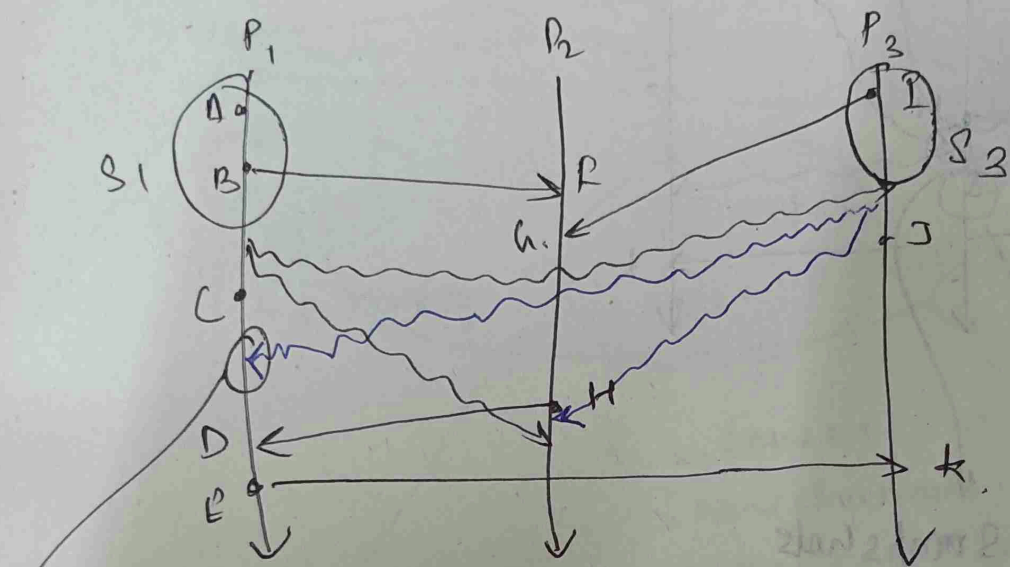
"This algorithm does actually assume that messages don't get lost"

we have reliable delivery, but they can't be slow

P_3 ke jab marker msg aya to,
~~uske~~ uske apna Snapshot liya aur.

$C_{13} \rightarrow$ empty

C_{23} - recording on



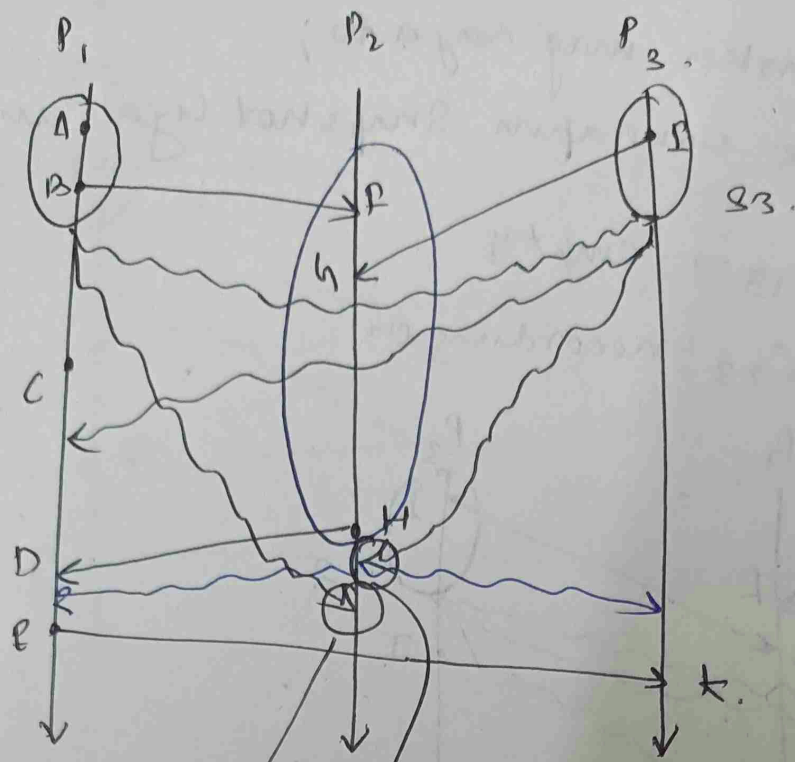
Let's see this, P_1 ka first time hi h
 marker msg dekha h, uske phere
 dekha h (bhejne time)

P_1
 \rightarrow stop recording on C_{31}

\rightarrow and P_1 didn't receive anything on
 channel C_{31} while it was recording

so

$C_{31} \rightarrow$ empty.



P_2 Snapshots

$C_{32} \rightarrow \text{empty}$
and $C_{12} \rightarrow \text{recording}$.

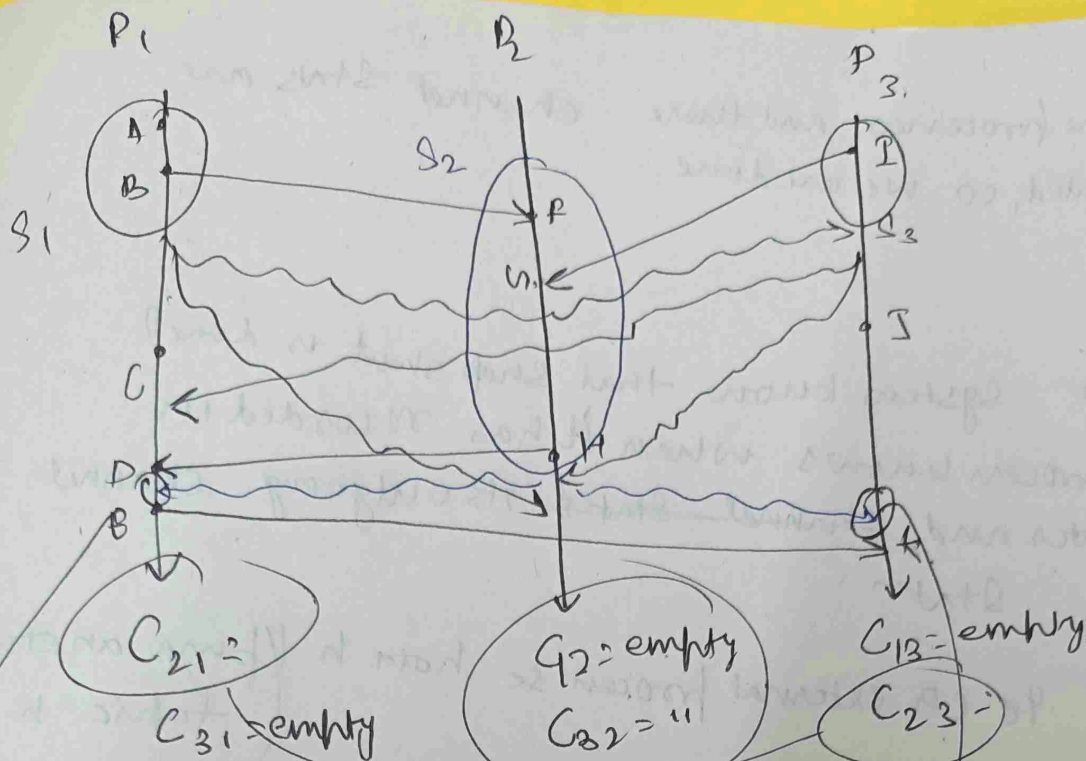
and outgoing he marker msg bhejega

P_2 ne phere dekh liya ki marker manage

C_{12} ki recording off an.

Since no msg arrived.
after ~~rec~~ Snapshot

So $C_{12} \rightarrow \text{empty}$



P_2 is done as.
Same channels,
uske marks
ho gaye h

par C_{21} aur C_{23} pe rec on hi h.

→ P_2 ne already dekh liya h marker msg to.

C_{12} ki rec off.

aur C_{12} pe msg →

(H → D)

$C_{12} = (H \rightarrow D)$

msg from H to D

No msg came while it was
rec so, C_{23} is empty

So, all the processes and their channel states are recorded, so we are done.

How does a system know that snapshot is done?

→ A process knows when it has recorded its states and, ~~channel states~~ its outgoing channel states.

→ Ye ek external process se hota h (pura another topic h)