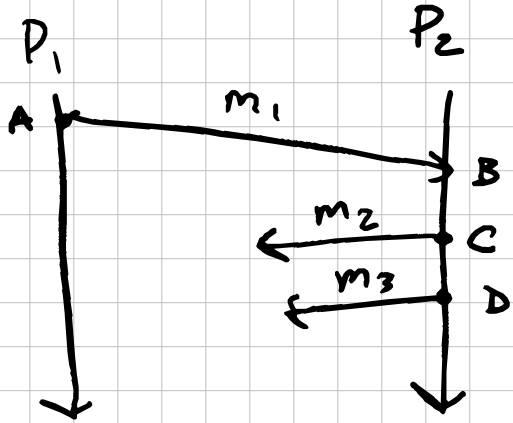
CSE138 Lecture 6

this time:

- Chandy-Lamport snapshots

Channel - a connection from one process to another

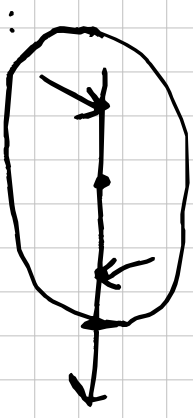


C_{12} - Channel From P_1 to P_2 - empty

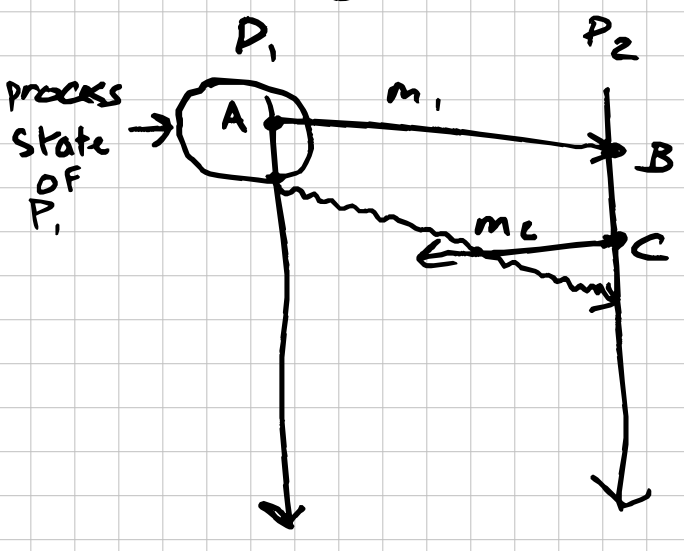
C_{21} - Channel From P_2 to P_1 - $[m_2]$

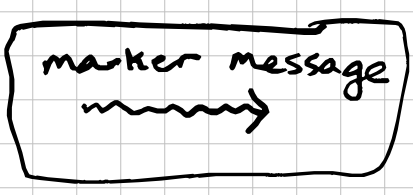
Assume that messages in a channel are delivered reliably, and in FIFO order.

initiating a snapshot - one (or more!) processes does the following:



- Record its <u>own</u> state.
- Send a <u>marker message</u> out on all its <u>outgoing</u> channels.
- Start recording the messages it receives on all its <u>incoming</u> channels.





<u>Receiving a marker message</u>

Two cases to be concerned with: either it's the first marker message we've ever seen, or ... it's not.

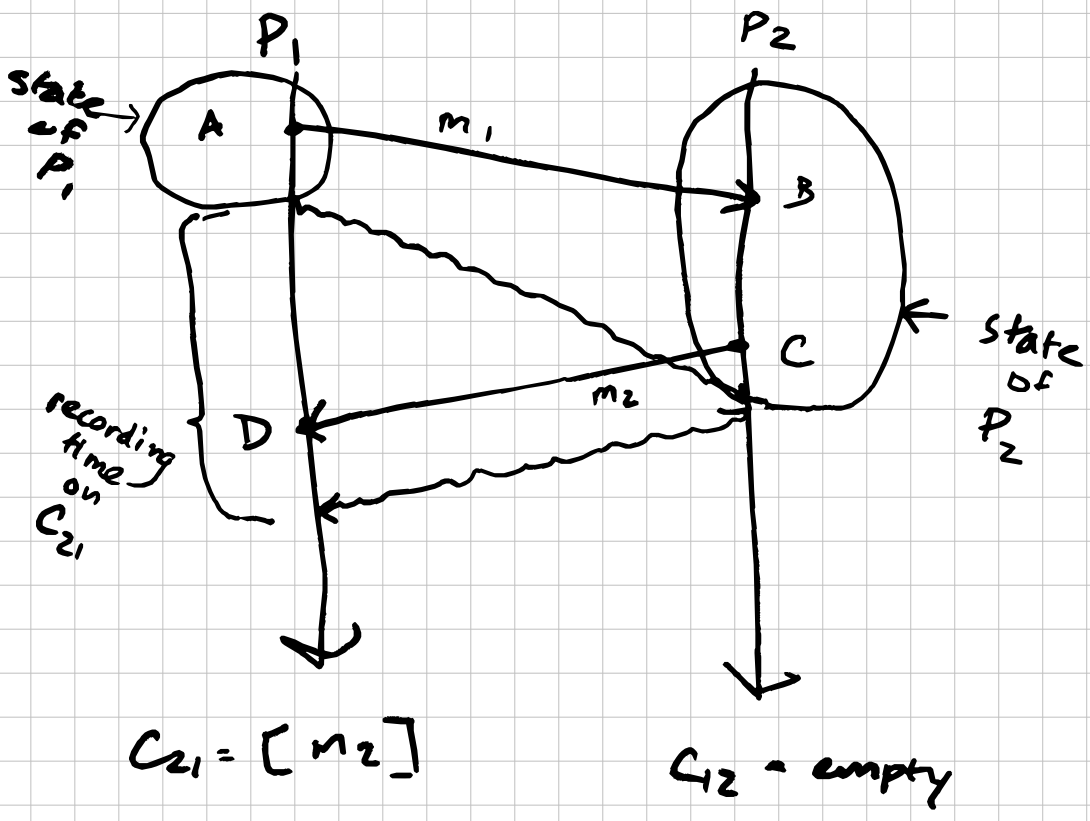
When process P_i receives a marker on C_{ki}:

IF it's the <u>first</u> marker P_i has seen:

- P_i records its state.
- P_i marks channel C_{ki} as "empty". (i.e., don't pay attention to C_{ki} anymore).
- P_i sends marker messages out on all its outgoing channels C_{ij}.
- P_i starts recording incoming messages on all its incoming channels except C_{ki}.

(C_{ki}: channel from process P_k to process P_i)

(TL;DR: Record your state and start recording incoming messages.)



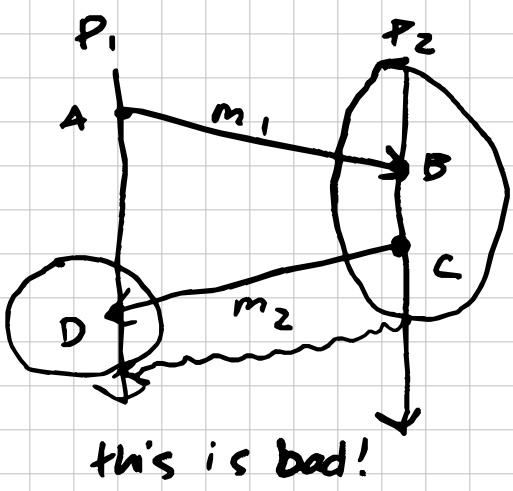
$C_{21} = [m_2]$ $\quad$ C_{12} = empty

back to describing the algorithm:

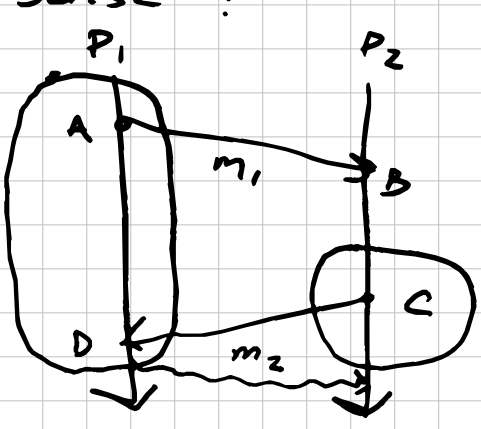
- If P_i receives a marker message on channel C_{ki} and P_i has <u>already seen</u> a marker:
  (↑ either sent or received!)
  - P_i stops recording on channel C_{ki}, and sets C_{ki}'s final state as as the sequence of all the messages that arrived on C_{ki} since recording began.

---

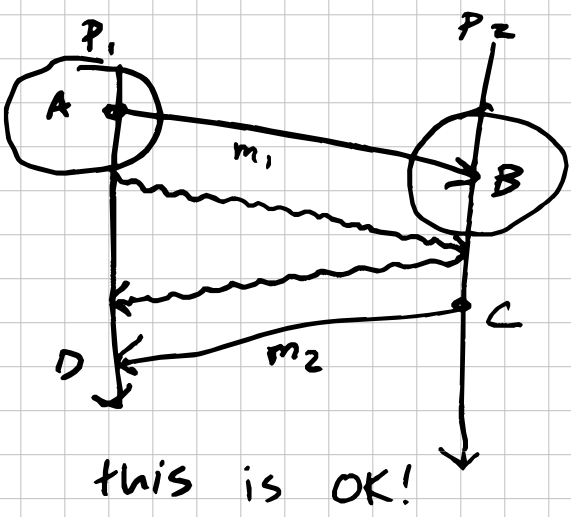
were done taking our snapshot. Does it "make sense"?



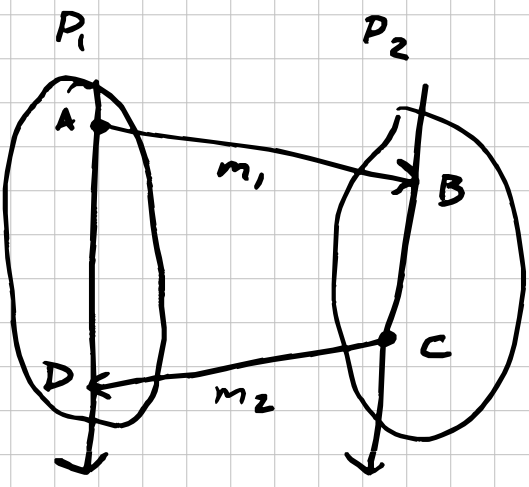
this is bad!



this is also bad!



this is OK!



this is also OK!

In general, the Chandy-Lamport algorithm only takes <u>consistent</u> snapshots.

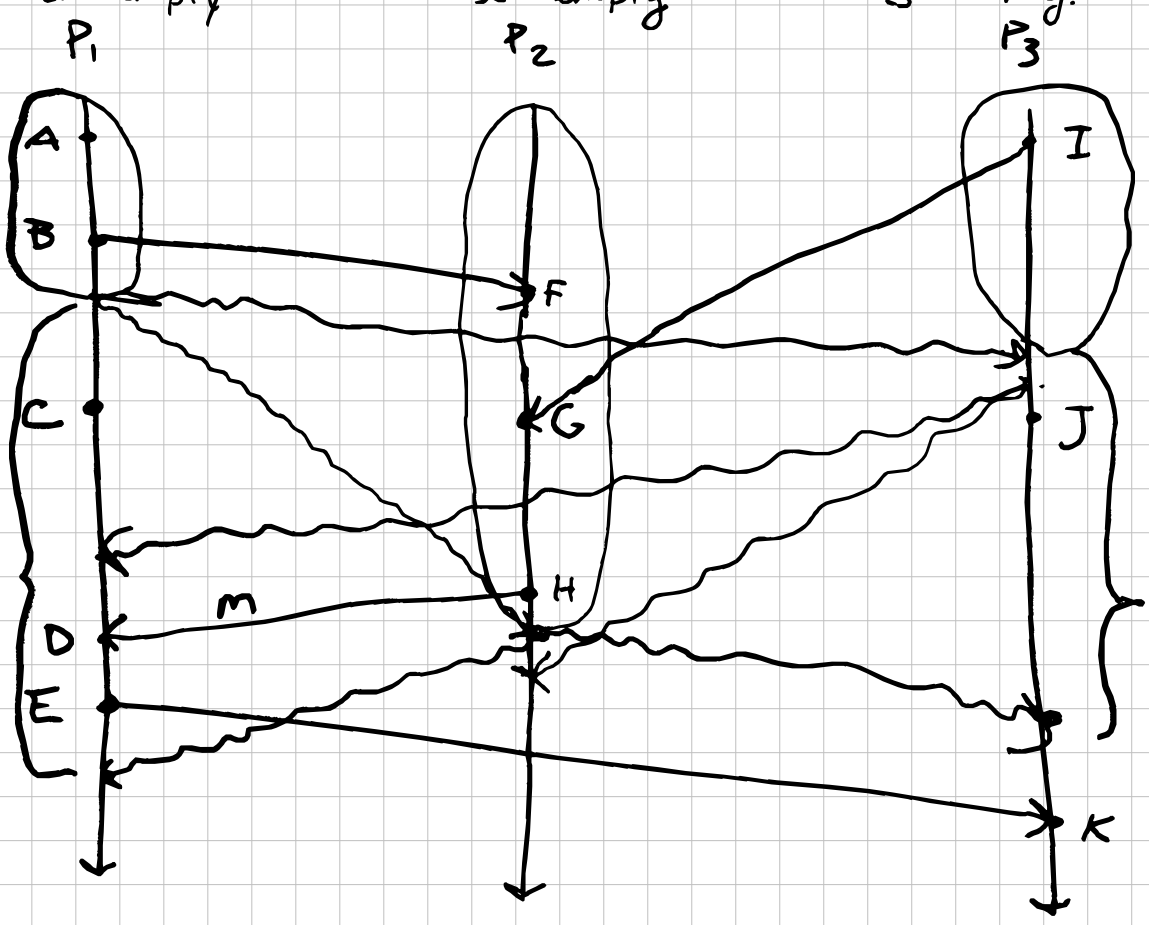
A consistent snapshot is one where, if an event e is in the snapshot, all events that <u>happened before</u> e are in the snapshot too.

Lamport's happens-before relation (not physical time)

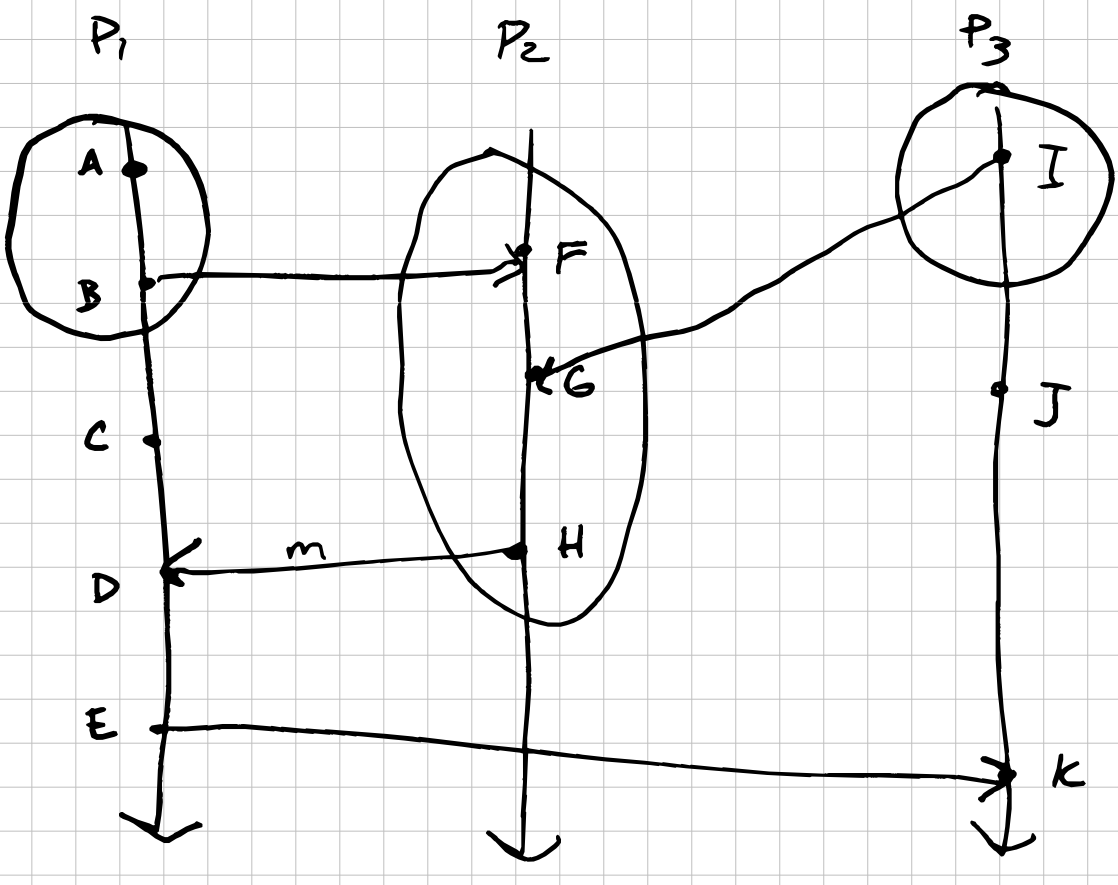
$C_{21} = [m]$
$C_{31} = \text{empty}$

$C_{12} = \text{empty}$
$C_{32} = \text{empty}$

$C_{13} = \text{empty}$
$C_{23} = \text{empty}$.



BTW, we have to send $O(N^2)$ marker messages when snapshotting a system with N processes.



What if P_2 had said, "I don't want to be the token holder anymore. P_1, you take over." in message m.

In the C-L algorithm, channel recordings help with this situation. In particular, we record the fact that m was sent on C_{21}, so m isn't forgotten about!

Weird and wonderful thing about the C-L algorithm: it works fine with multiple initiators!