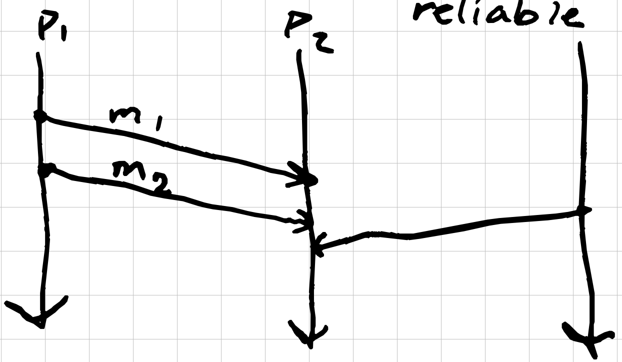
CSE138 Lecture 7

<u>this time</u>:

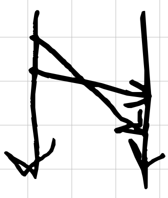
- ✓ Chandy-Lamport wrap-up
- ✓ uses of snapshots
- ✓ centralized vs. decentralized algorithms
- ✓ cuts and consistent cuts
- ✓ safety and liveness

if time {
- reliable delivery
- fault models

<u>channels</u> – connect one process to another, FIFO behavior, reliable behavior



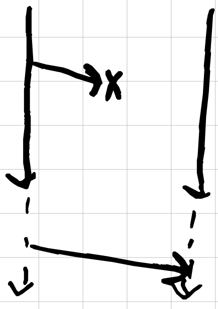
FIFO violation



won't happen in channels

example of a <u>safety</u> property.

(sort of) reliable delivery violation



example of a <u>liveness</u> property.

what does it mean for a snapshot algorithm to be "correct"?

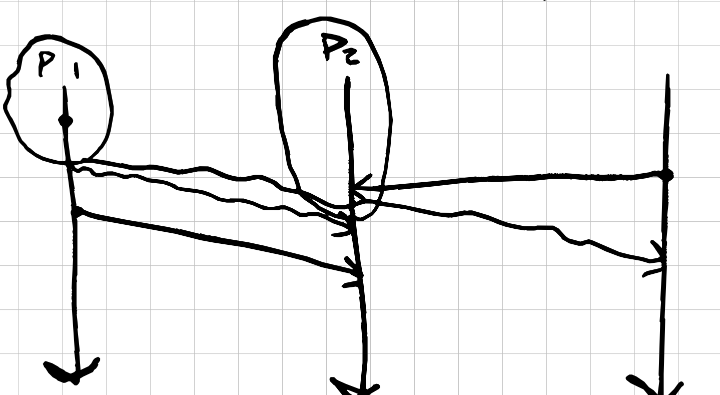
<u>safety</u> {
- snapshots it takes are consistent snapshots.
  (if an event e is in the snapshot, then all events e′ such that e′ → e are also in the snapshot.)

liveness {
- termination! I want to eventually get done taking a snapshot.

Does the C-L algorithm terminate?

What assumptions do we have to make?

- Fixed number of processes



if every process does this stuff, we terminate!

a process has to record:

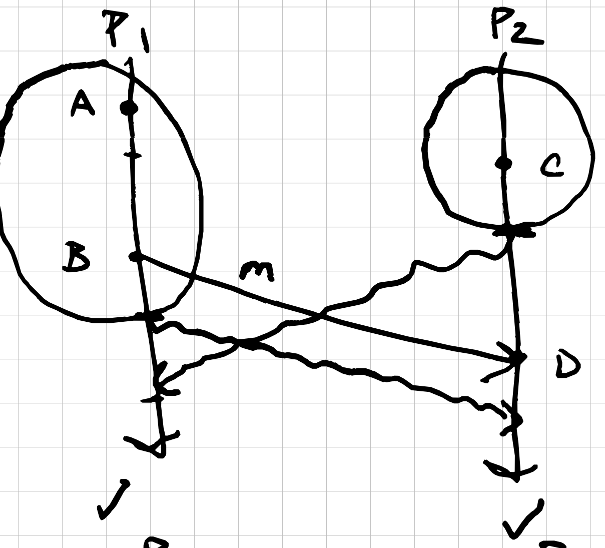
- its own process state ✓
- the states of its incoming channels. ✓

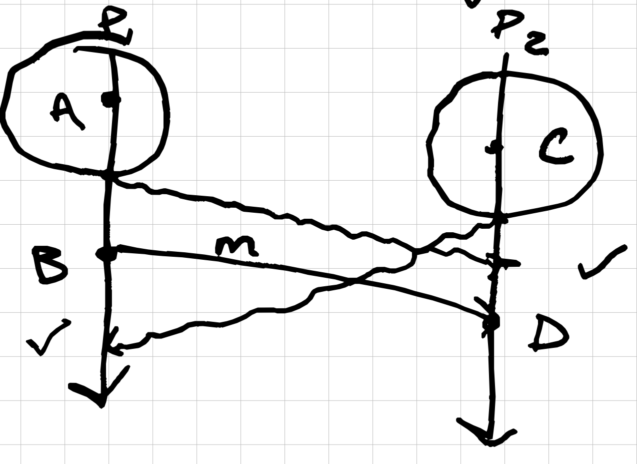
once everyone receives a marker message on all of their incoming channels, this is done.

---

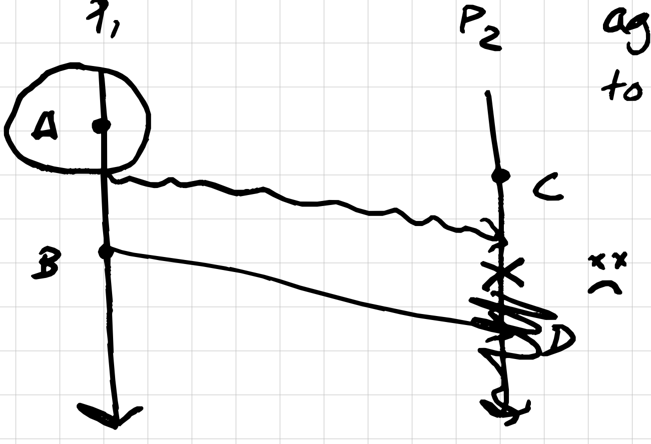
in the C-L algorithm, at least one process has to initiate.

but more than one is OK!





This is actually a huge deal that multiple processes can initiate the algorithm independently. (because if this weren't true, we would have to solve a hard agreement problem to decide who gets to initiate).



if processes can crash, termination is not guaranteed. (e.g., here P1 would be waiting forever to get a marker message that's never being sent.)

Because this algorithm works even with multiple initators, it is an example of a decentralized algorithm.

A decentralized algorithm is one that can be independently initiated by different processes without them needing to coordinate with each other.

(By contrast, a centralized algorithm must have exactly one initiator.)

What are snapshots for?

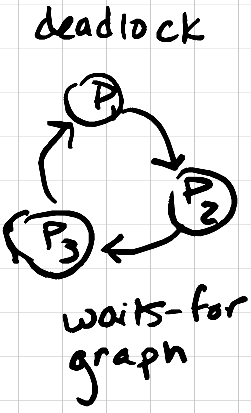
- Checkpointing - take periodic snapshots ~~of~~ the state of a running application, so that in case of failure, they have a consistent state to start from

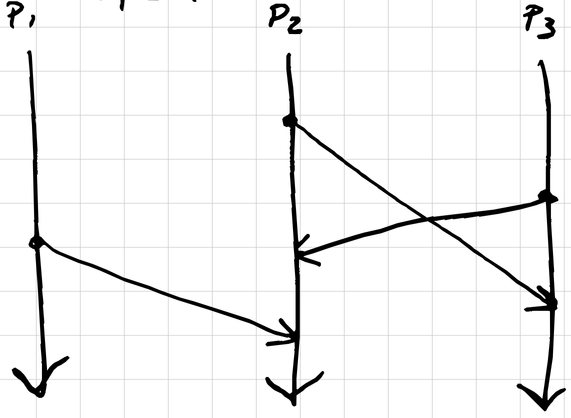
VLDB 2017 Flink paper → "as if nothing had gone wrong",

↑ according to the Apache Flink docs, anyway.

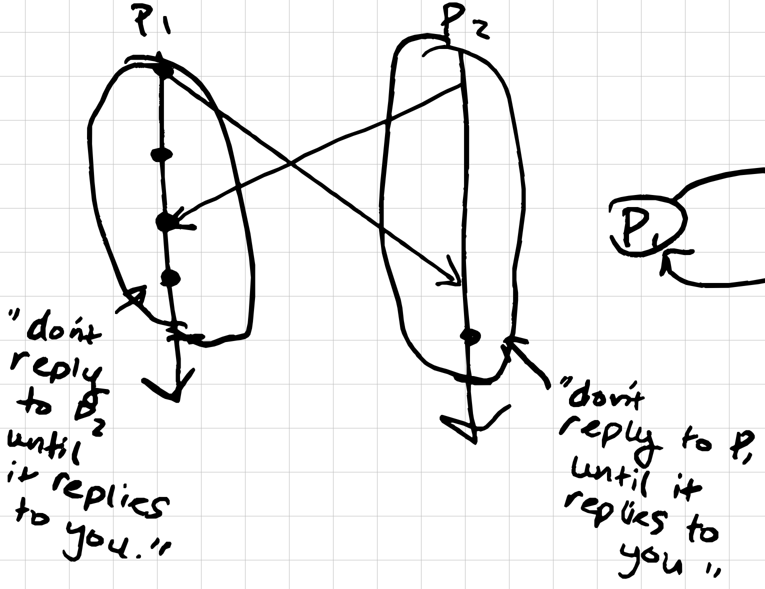
incl. deadlock detection ↓

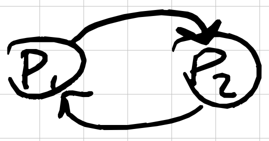
- Stable property detection - to detect if a stable property is true of a system, take a snapshot and inspect it.



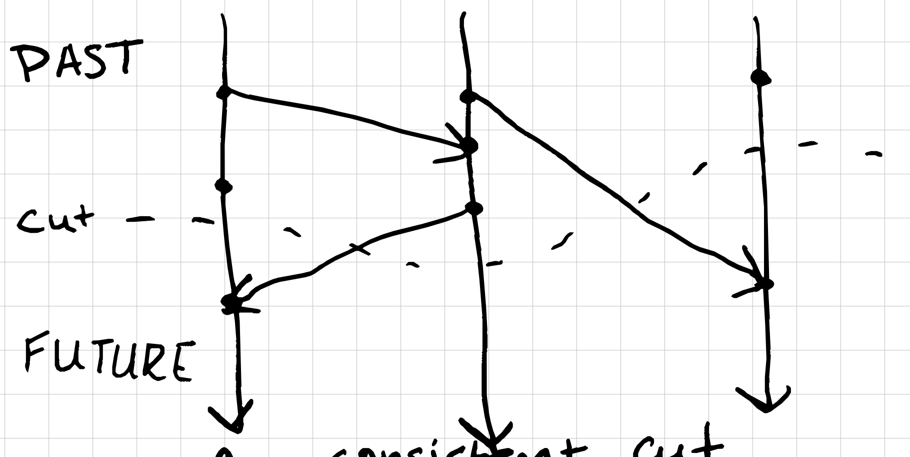


deadlock is an example of a stable property: a property that, once true, remains true forever.


P1
P2
"don't reply to P2 until it replies to you."
"don't reply to P1 until it replies to you"

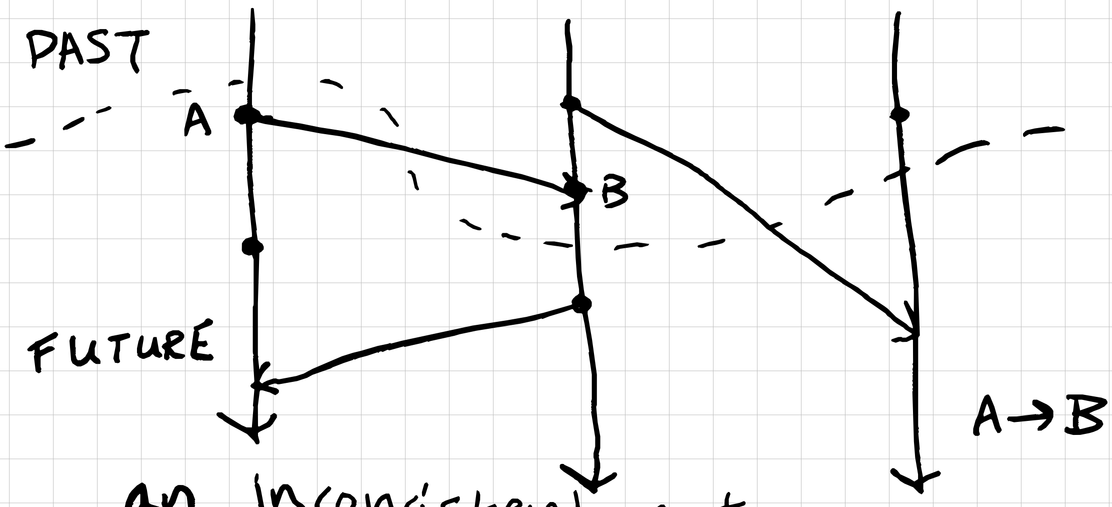


cuts and consistent cuts



a consistent cut.

A **cut** is an imaginary line through an execution dividing its events into "past" and "future"



an inconsistent cut.

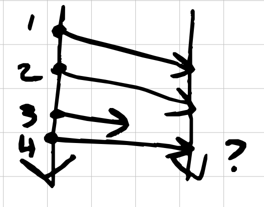
A "good" snapshot algorithm, such as the C-L algorithm, will only take snapshots that correspond to consistent cuts.

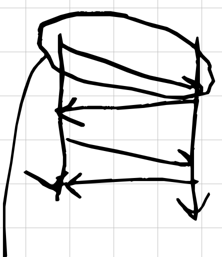
An inconsistent cut is one in which there exist events e and e′ such that e ⟶ e′, but e′ is in the "past" set of events and e is in the "future" set of events.

A little more about reliable delivery.

a liveness property

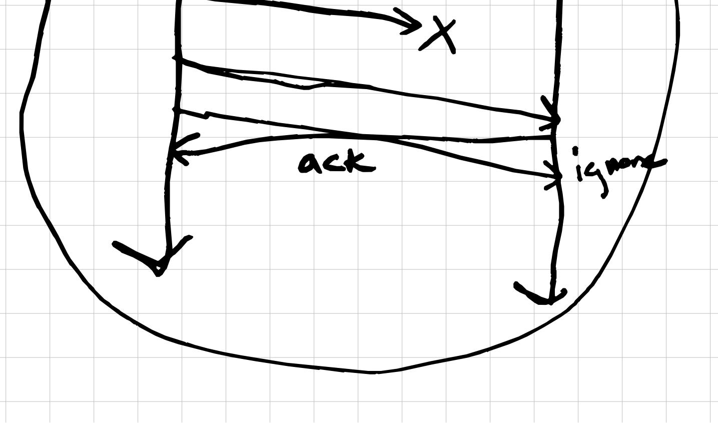
Reliable delivery goes hand in hand with safety properties such as FIFO delivery.





How do we actually get reliable delivery?

idea: Keep sending until you get an ack.



reliable delivery protocol