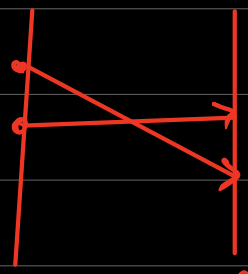
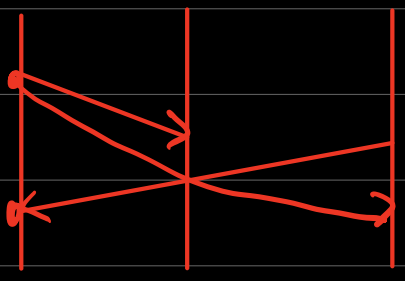
Agenda

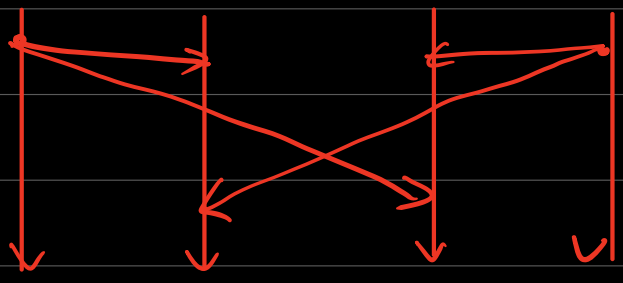
- Recap of delivery properties
- Implementing causal broadcast
- Use of causality
- Chandy-Lamport snapshot algorithm



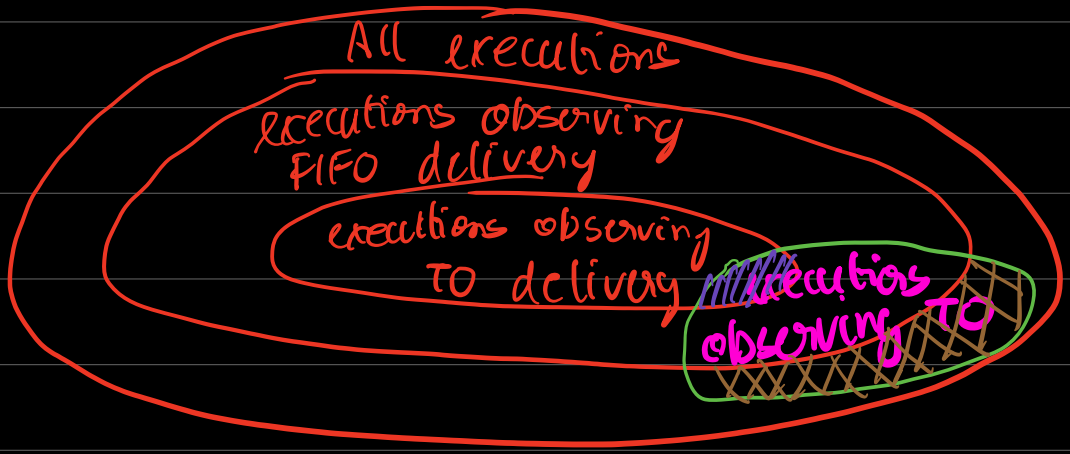
Violation of FIFO delivery



Violation of causal delivery



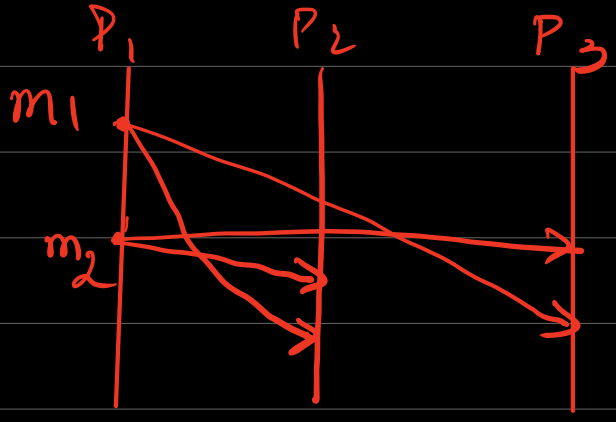
Violation of TO delivery



Executions with TO + causal delivery

Executions with TO BUT NOT FIFO

An execution in the brown zone:



P_2 & P_3 got m_2 then m_1

However, the order of messages in both P_2 & P_3 was the SAME! Hence, it obeys TO delivery.

Implementing causal broadcast

Unicast messages (point-to-point): 1 sender 1 receiver

Multicast messages: 1 sender many receivers

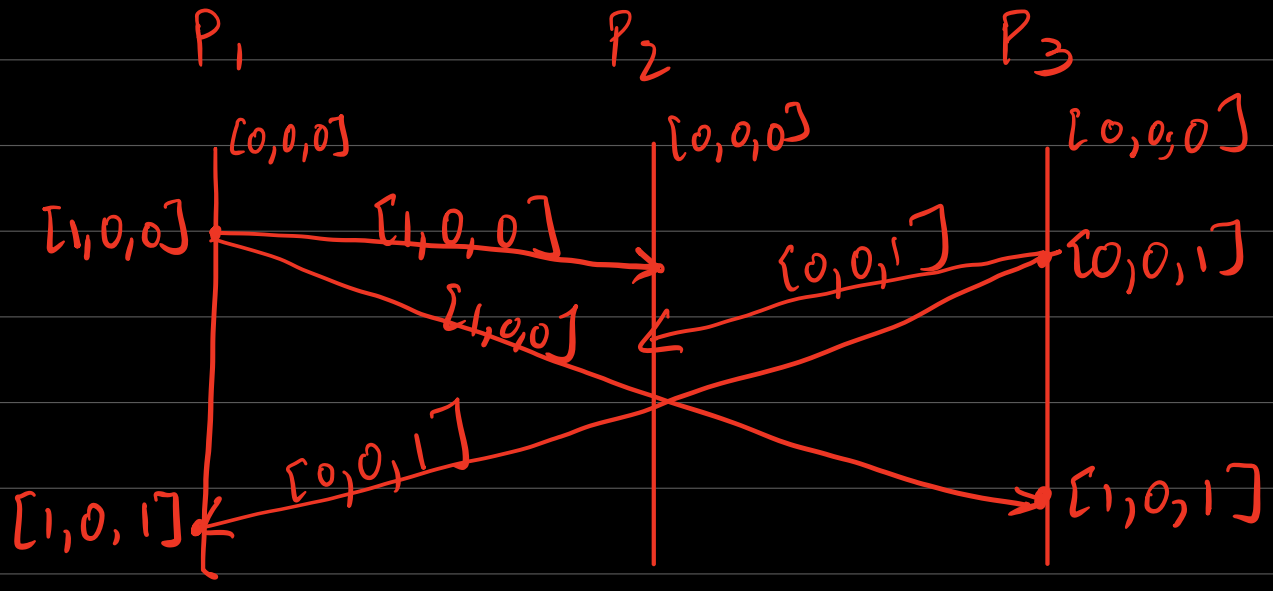
Broadcast: special case of multicast msgs
1 sender everyone receives
↓
including sender itself!

Vector clocks algorithm (with a twist?)

don't count msg receives as events!

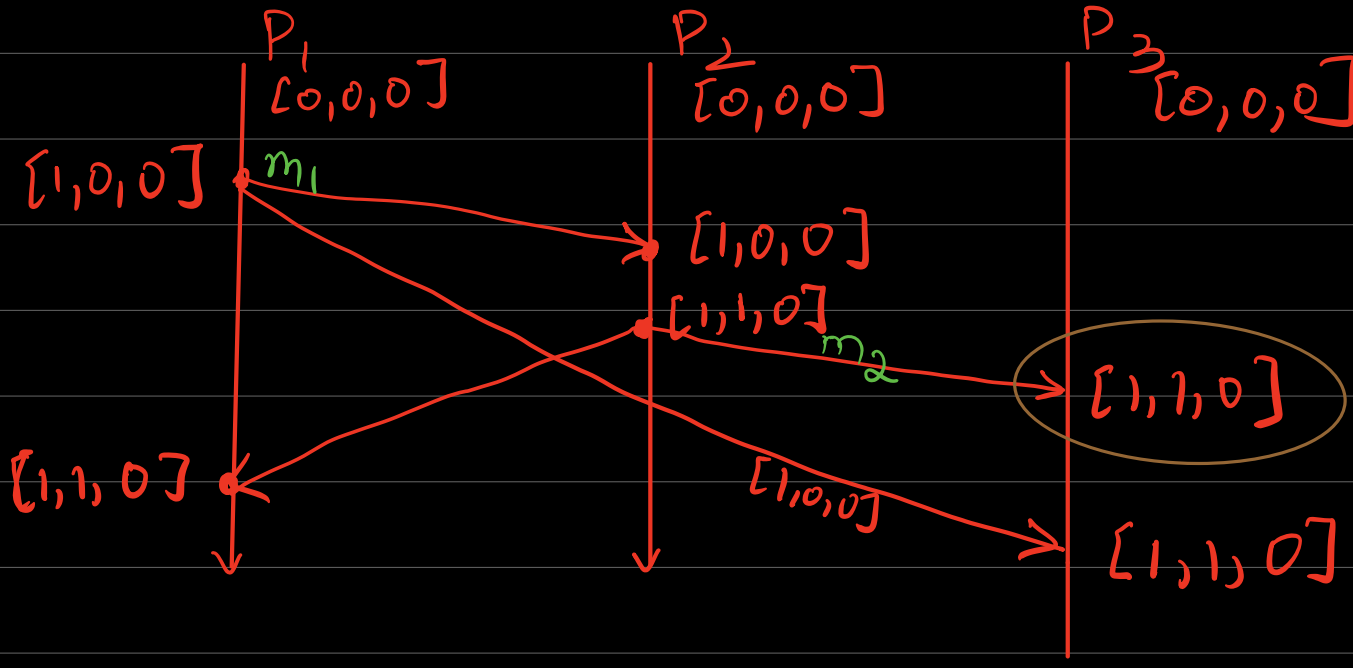
Don't count message receives as events!

- Every process keeps at VC, init at 0
- When a process sends a message, it increments it's own position in its VC, and it's going to include updated VC with the message
- When a process delivers a message, it updates its VC to the pointwise max of its local VC & the received VC on the message



Causal delivery — Property of executions that we care about today

Causal broadcast — Algo that gives causal delivery in a setting where all messages are broadcast messages



Brown event is the problematic message!
m_2 is getting delivered before m_1

How to use the VC to determine which message can be delivered now & which message should be reserved & delivered later?

Causal broadcast

Define a deliverability condition that tells us if a received message is or is not OK to deliver

This deliverability condition will use the vector clock on the message.

Deliverability:

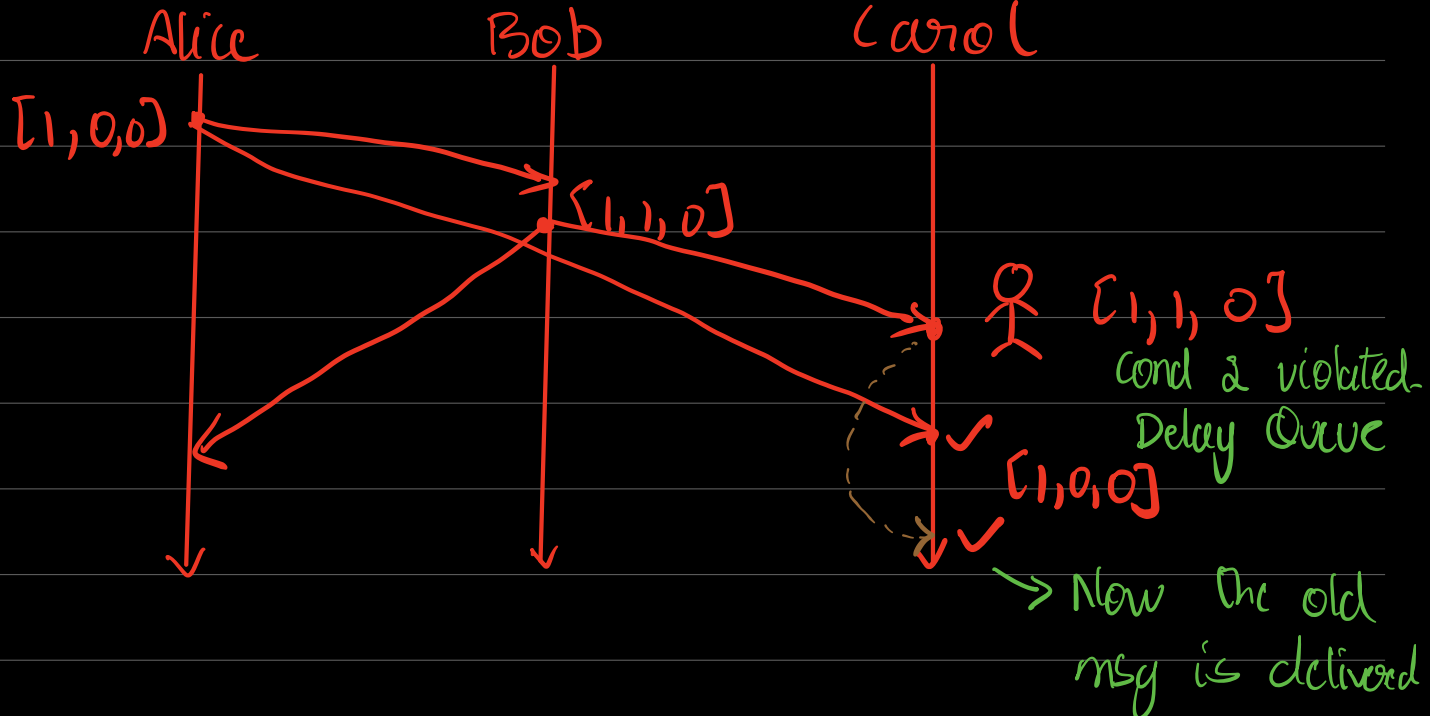
A message m is deliverable at process P if, for $k: 1 \ldots n$ where VC's have n entries

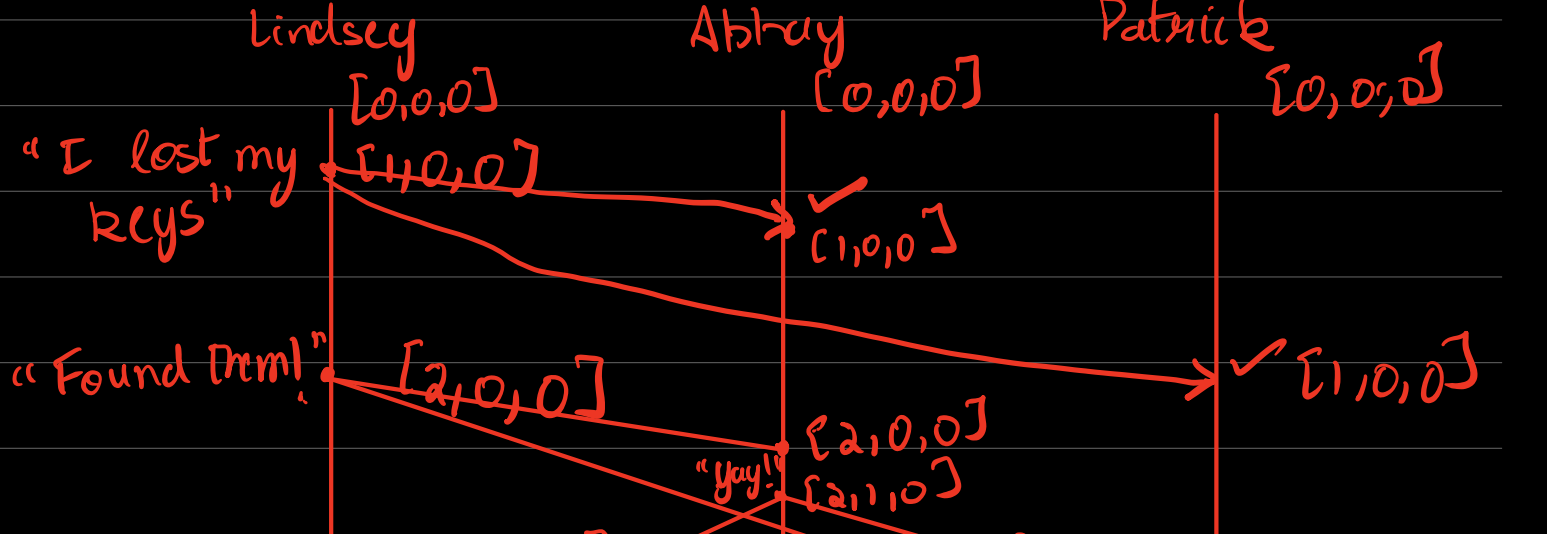
$$VC(m)[k] = VC(p)[k] + 1$$

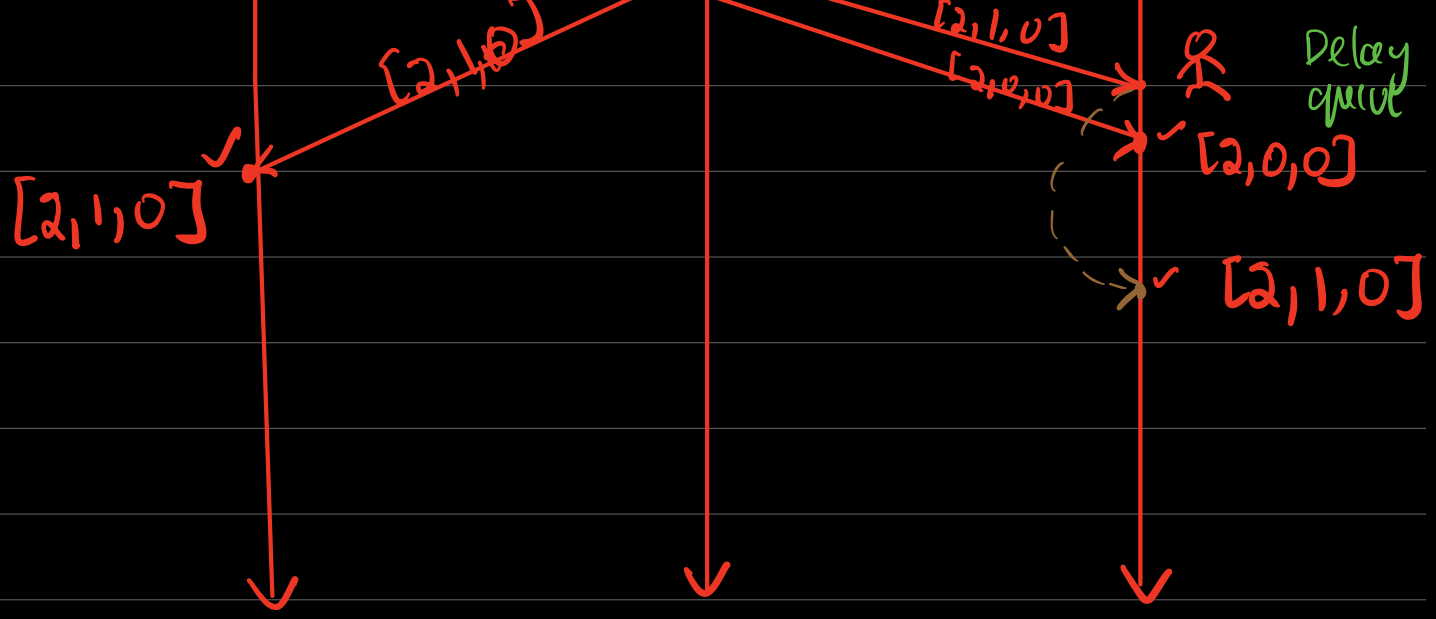
k is the sender

$$VC(m)[k] \leq VC(p)[k]$$

otherwise







Queue size? Not too large

Snapshots of distributed systems

Global state of the system: How?
- Using globally synced clock can't use

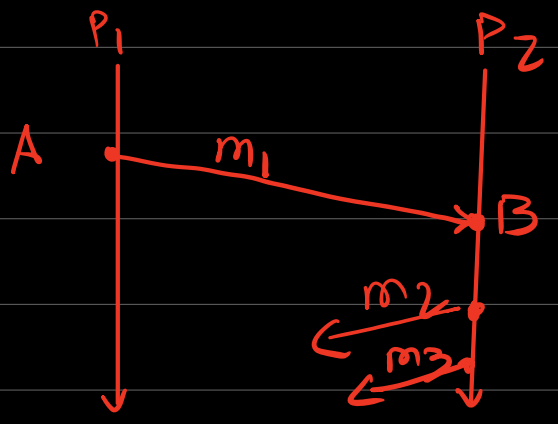
Property that we want:
If we have events A & B where A → B & B is in the snapshot, A is in the snapshot too!

Chandy - Lamport Algorithm

1985 algorithm

Channel: Connection from one process to another

with FIFO ordering!



C_{12} - channel from P_1 to P_2

id of sender

id of receiver

C_{21} - channel from P_2 to P_1

Inflight msg (m_2) are in the channel

$$C_{21} = [m_2, m_3]$$