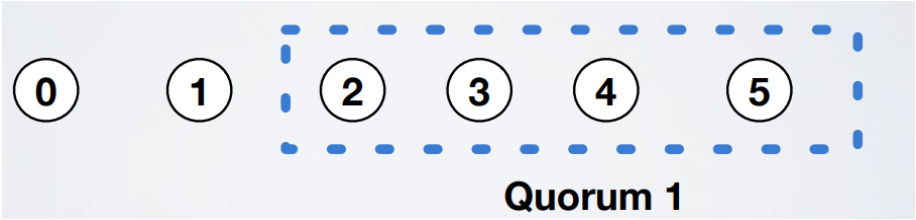
# 7. QUORUM

## Quorum Definition and Properties

We have $n$ processes in our set of total processes $P$, **a quorum is any subset of $P$ of size at least**:

$$\frac{n}{2} + 1 = \text{Majority}$$




**property**: any two quorums intersect in at least one process (suppose two quorums do not intersect, then they have all distinct processes. This implies $|P| > n$)

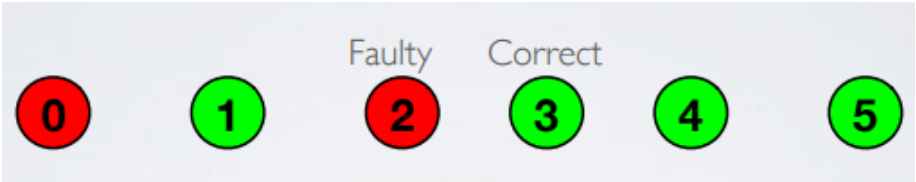
If we have:

- $C$: set of correct processes.
- $F$: set of faulty processes.

We don't know who is in $C$ and who is in $F$ but if we assume $f < \frac{n}{>}2$ (Majority of the processes are in $C$), then we can say that:

**Any quorum $Q$ of $P$ contain at least one correct process** ($C$ is a quorum, $Q$ is a quorum tooo. Two quorums intesect in at least one process).

**Example**: takes 4 processes at random, at least one of them will be green. Protip: it works even if you first take your set and then you decide which is green and which is red.




## Implementation in Fail-Silent

Implements **URB** and uses **BEB**.
**We need to assume a majority of correct processes**.

**Majoriti-ACK**:

**Algorithm 3.5:** Majority-Ack Uniform Reliable Broadcast

**Implements:**
UniformReliableBroadcast, **instance** *urb*.

**Uses:**
BestEffortBroadcast, **instance** *beb*.

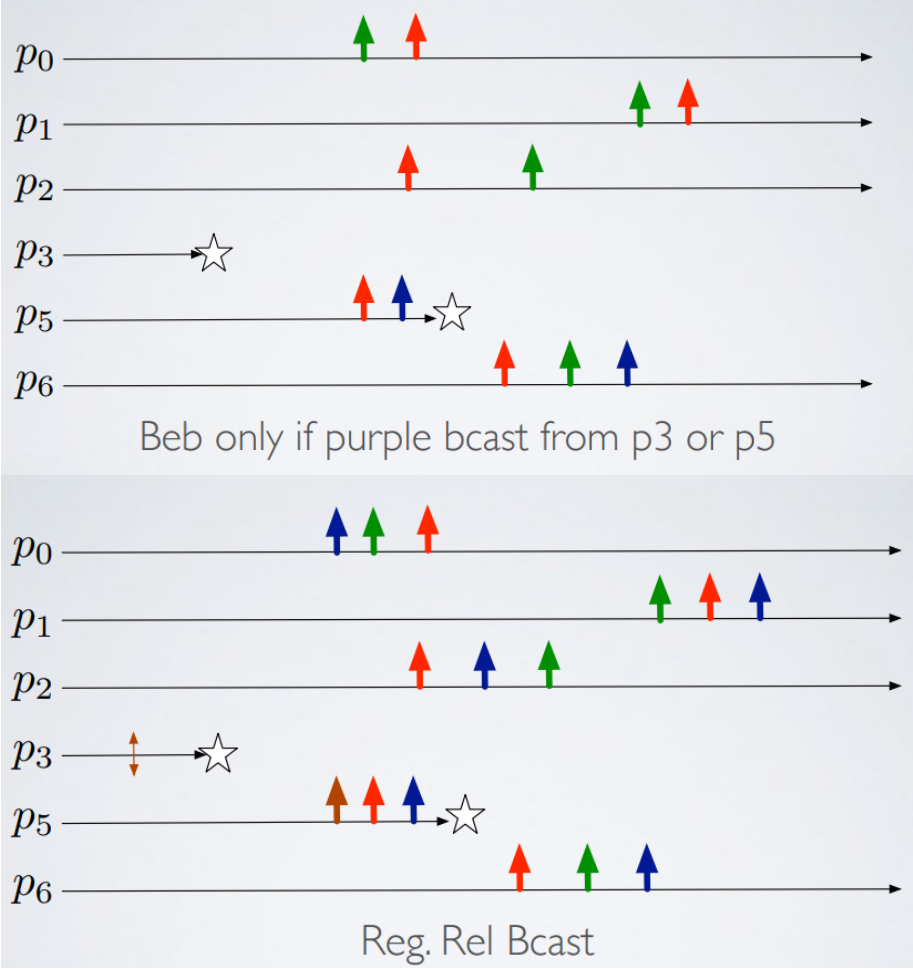
// Except for the function *candeliver*(·) below and for the absence of ⟨ *Crash* ⟩ events
// triggered by the perfect failure detector, it is the same as Algorithm 3.4.

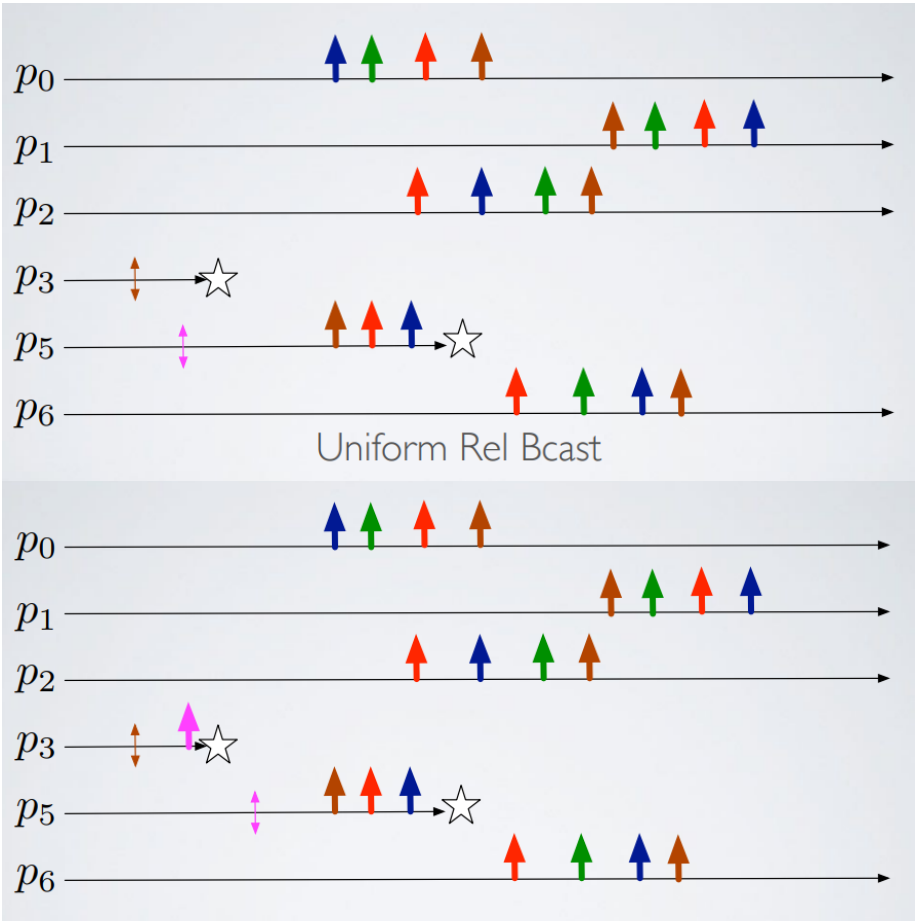
**function** *candeliver*(*m*) **returns** Boolean **is**
**return** #(ack[m])>|F|

if I reach the treshold is fine but can happen that the number of ACK will never be reached (if |F| > n/2 and I wait for |F|+1 I will wait forever once everyone in F crashed. This means that the validity will never be satisfied)

We can use perfetct failure detector:

## Examples






## Ordered Communications

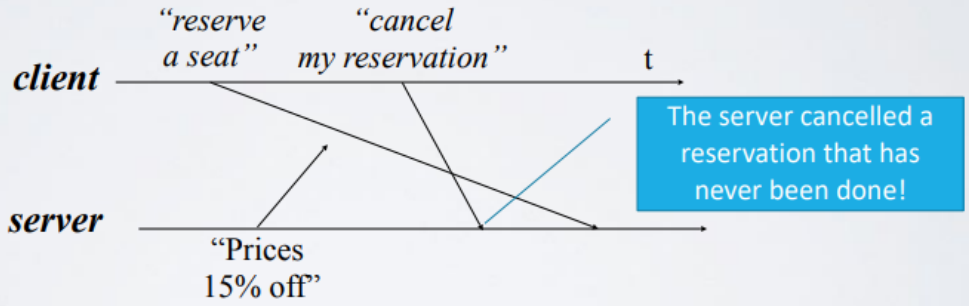
- Define guarantees about the order of deliveries inside group of processes
- Type of ordering:
  - Deliveries respect the FIFO ordering of the corresponding send
  - Deliveries respect the Causal ordering of the corresponding send
  - Delivery respects a total ordering of deliveries (atomic communication, we will see them after consensus)

## ADVANTAGES OF ORDERED COMMUNICATION

Orthogonal TO reliable communication.

- Reliable broadcast does not have any property on ordered delivery of messages

This can cause anomalies in many applicative contexts

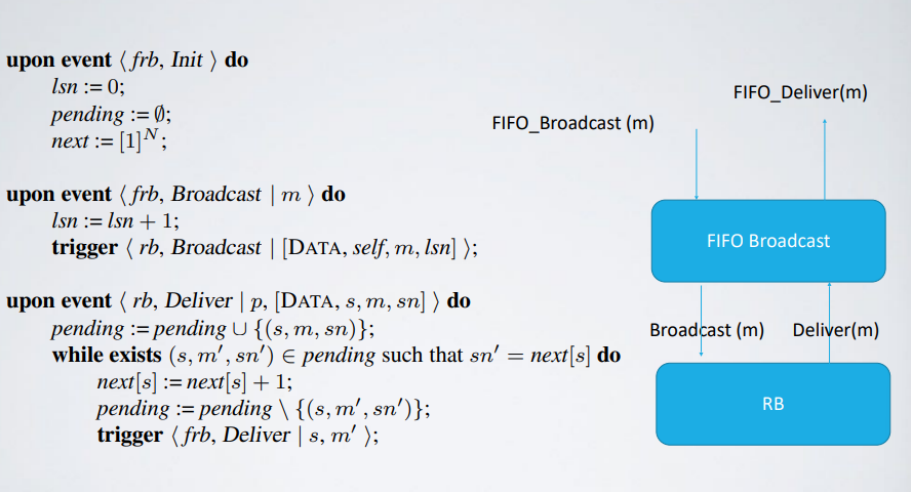



"Reliable ordered communication" are obtained adding one or more ordering properties to reliable communication

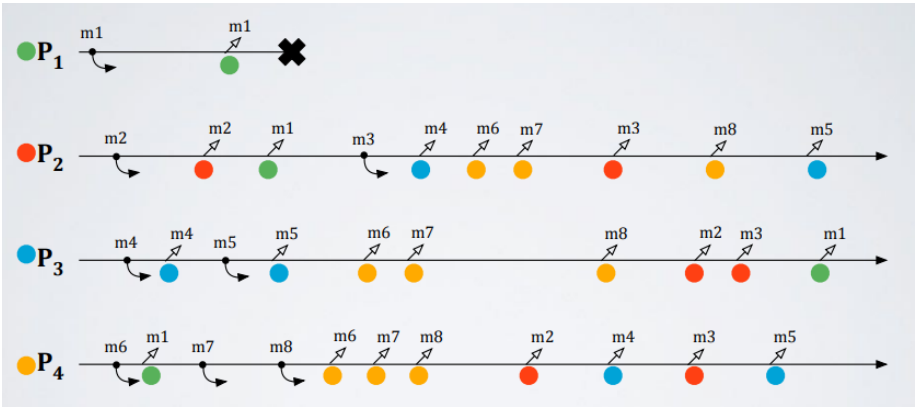
## FIFO BROADCAST

can be regular or uniform

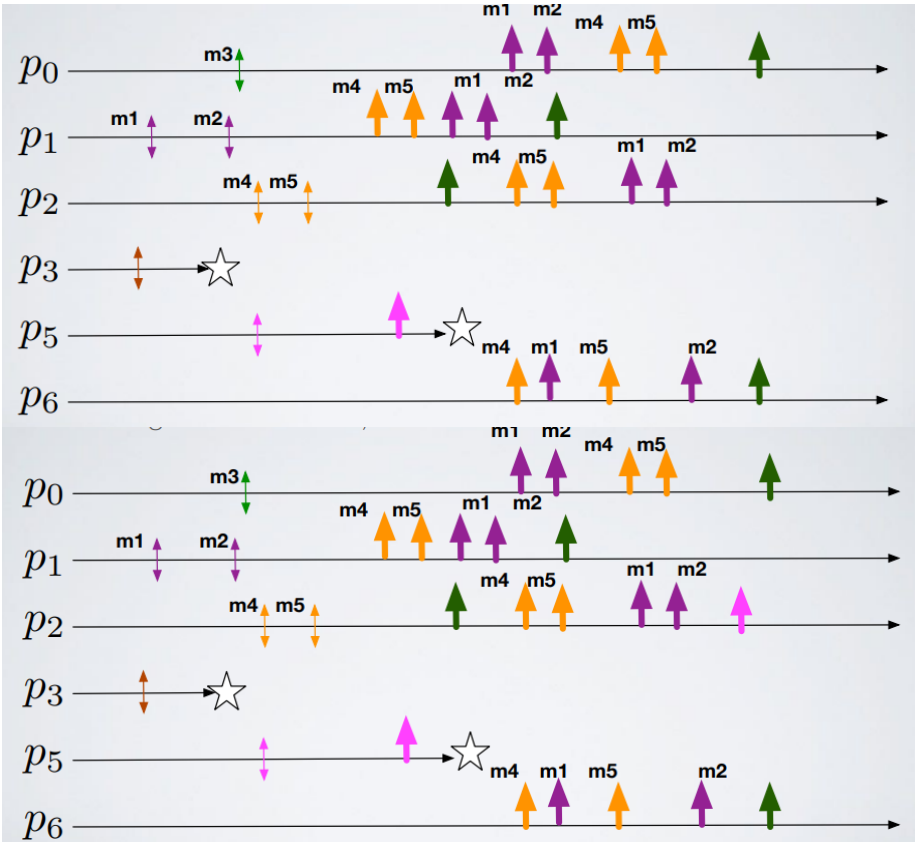
**properties**
first 4 properties = RB
**FRB5** (FIFO Delivery): if some process broadcast message $m_1$ before it broadcasts message $m_2$, then no correct process delivers $m_2$ unless it has already delivered $m_1$.

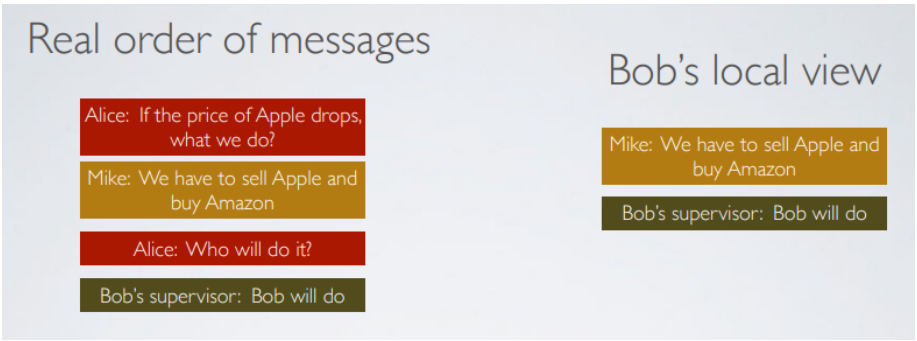



**Example**:




**FIFO RELIABLE BROADCAST**:
FIFO property is orthogonal with reliability guarantees (you can be FIFO but not regular nor uniform):



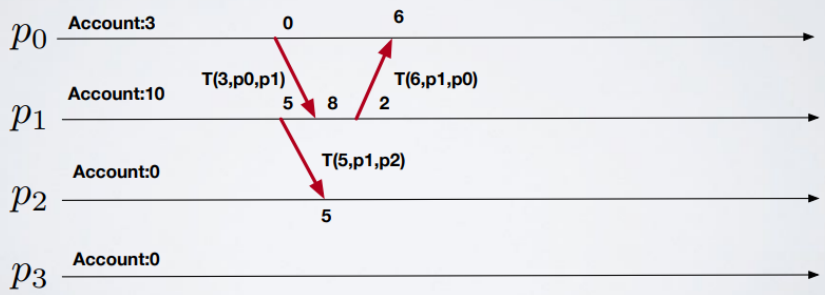

**Example**:

Money Transaction:

T(MoneyQuantity, SourceAccount, DstAccount).

Simple Check:

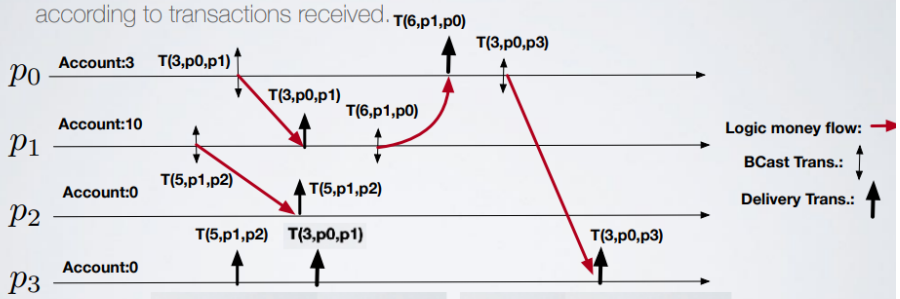
T is valid, at logical time t, if SourceAccount-MoneyQuantity >= 0.



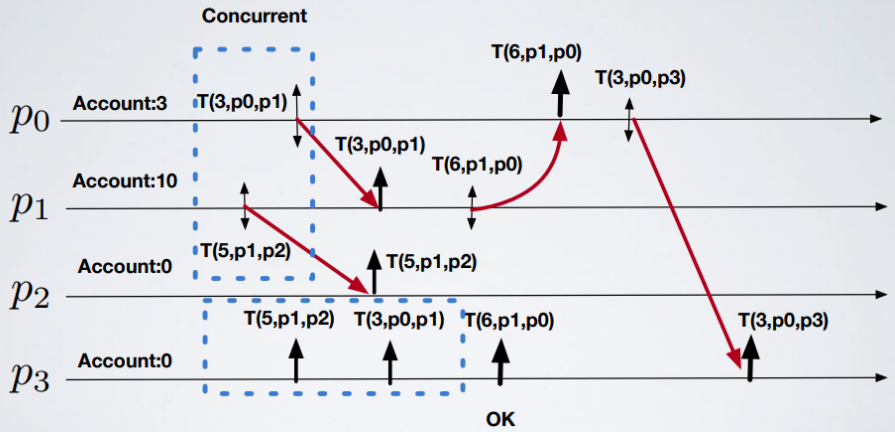

Simple Check:T is valid, at logical time t, if SourceAccount-MoneyQuantity >= 0.

Distributed implementation:

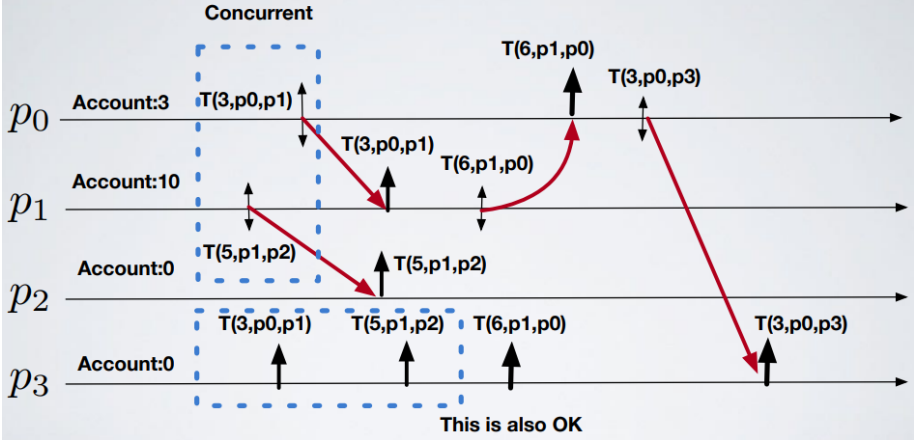
- At the beginning each one knows the balance of others.
- Uniform FIFO broadcast each transaction. Update your view of the world according to transactions received.




Happened-before



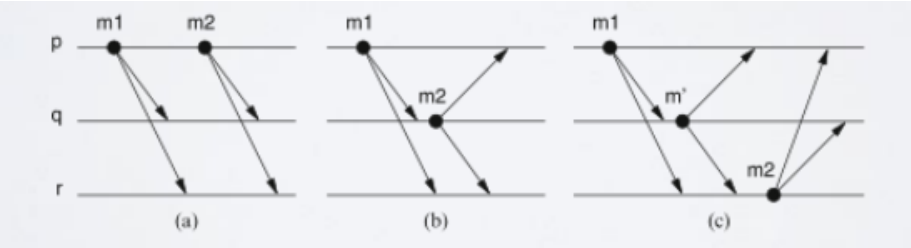

Happened-before



# CAUSAL ORDER BROADCAST

**Guarantees that messages are delivered such that they respect all cause–effect relations**.
Causal order is an **extension of the happened-before relation**.
A message $m_1$ may have potentially caused another message $m_2$ (denoted as $m_1 \rightarrow m_2$) if any of the following holds:

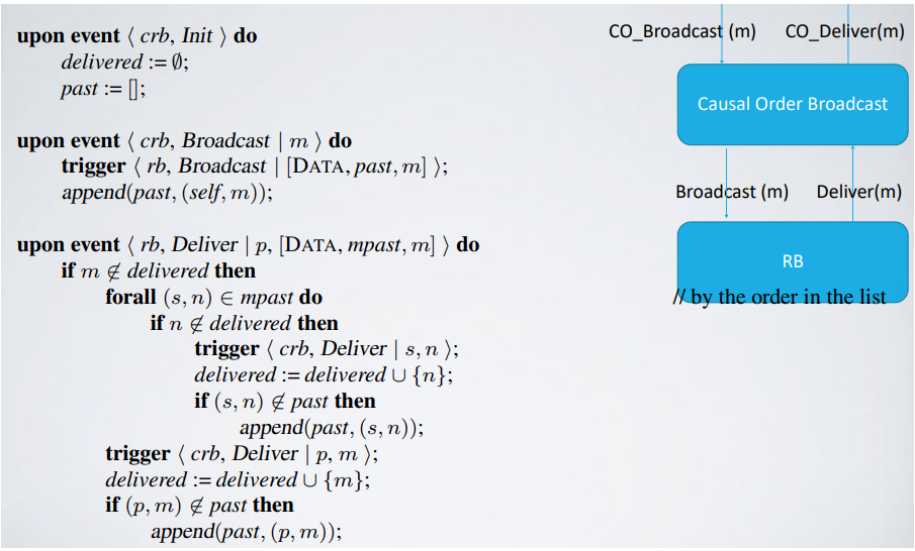
- some process $p$ broadcasts $m_1$ before it broadcasts $m_2$.
- some process $p$ delivers $m_1$ and subsequently broadcasts $m_2$.
- there exists some message $m\prime$ such that $m_1 \rightarrow m\prime$ and $m\prime \rightarrow m_2$



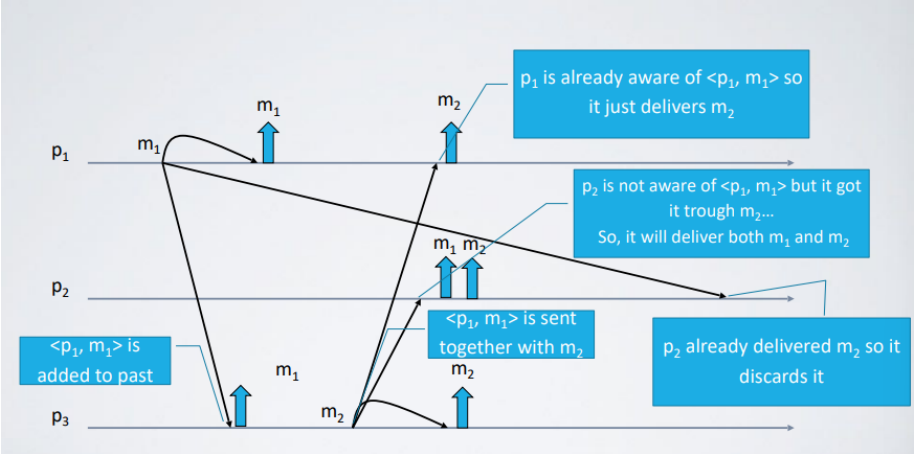

## NO-WAIT Casual Reliable Broadcast (NO-Wait CRB)

**Properties**:

- **CRB1-CRB4** = RB1-RB4
- **CRB5** (Causal Delivery): : For any message $m_1$ that potentially caused a message $m_2$, i.e., $m_1 \rightarrow m_2$, no process delivers $m_2$ unless it has already delivered $m$.




**example**:



**+++ PAG 29 DS-2324Ordered_communications_1**