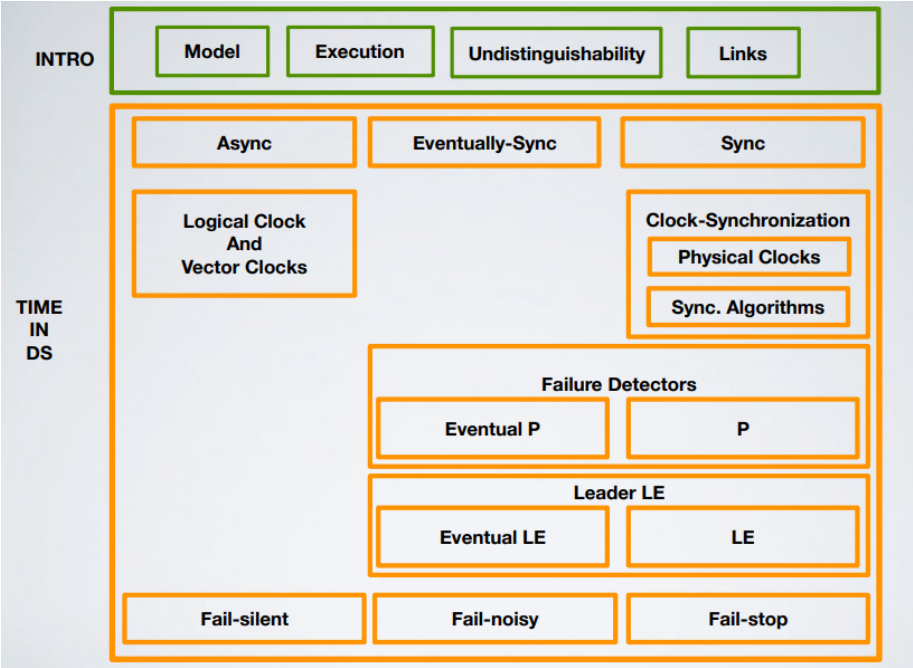
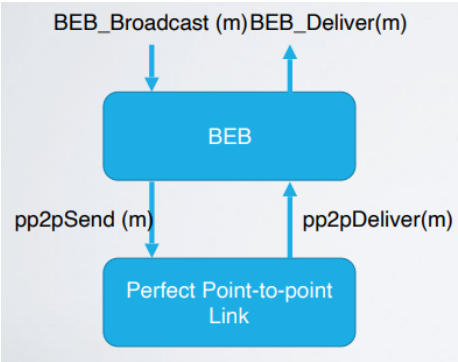
6. BROADCASTING



BEST EFFORT BROADCAST (BEB)

System model:

- Asynchronous system
- Perfect links
- Crash failures



Algorithm

Implements:
BestEffortBroadcast, **instance** *beb*.

Uses:
PerfectPointToPointLinks, **instance** *pl*.

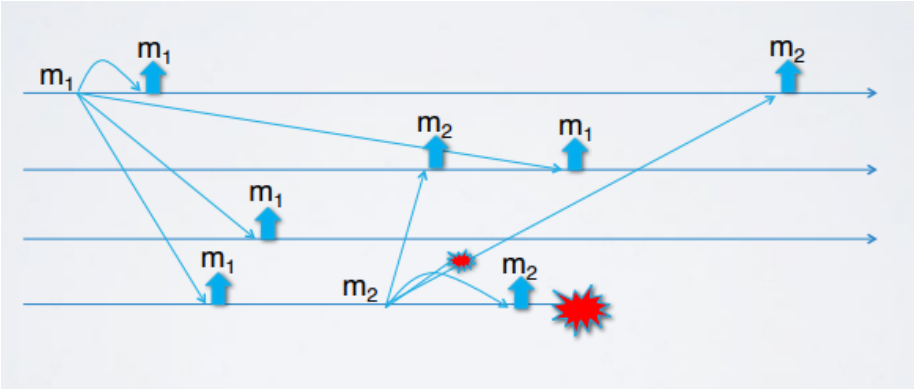
upon event $\langle$ *beb*, *Broadcast* | m $\rangle$ **do**
forall $q \in \Pi$ **do**
trigger $\langle$ *pl*, *Send* | q, m $\rangle$;

upon event $\langle$ *pl*, *Deliver* | p, m $\rangle$ **do**
trigger $\langle$ *beb*, *Deliver* | p, m $\rangle$;

Correctness:

- *Validity*: it comes from the reliable delivery property of perfect links and the fact that the sender sends the message to every other process in the system.
- *No Duplication*: it directly follows from the No Duplication of perfect links.
- *No Creation*: it directly follows from the corresponding property of perfect links.

BEB ensures the delivery of messages as long as the sender does not fail.
If the sender fails processess may disagree on whether or not deliver the message.



Problems

Processes do not have a common view of messages sent by the non correct.

- This can have a negative impact in some applications:
  - Think about a distributed chat, a message from a client that crashes could reach only a subset of processes.
- We would like to have an agreement on the set of messages to be delivered, at least among the correct processes (all-correct or nothing).

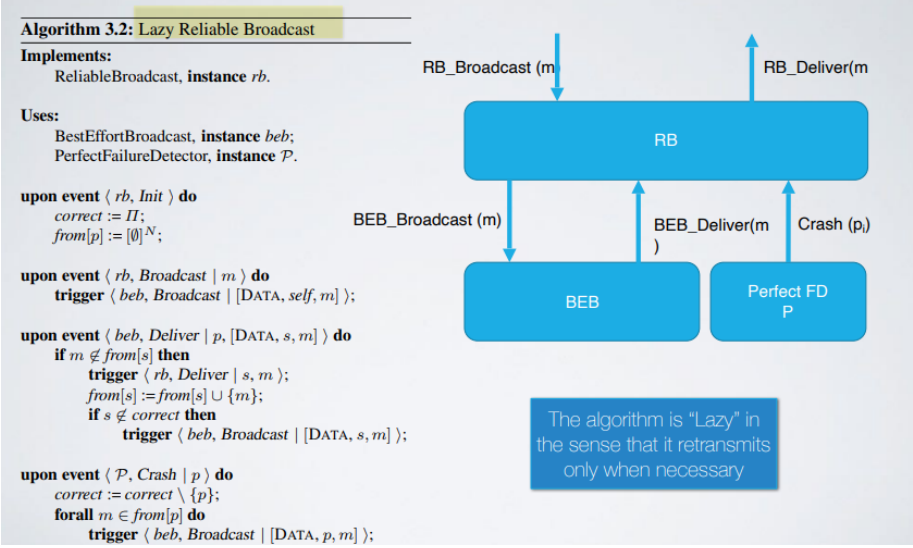
REGULAR RELIABLE BROADCAST (RB)

Module:

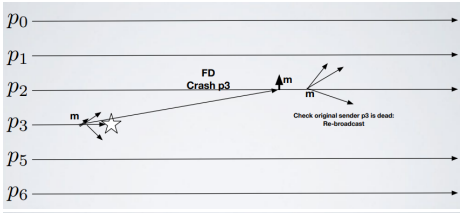
- **Name**: Reliable Broadcast, **istance**: rb.
  Events:
- **Request**: $\langle rb, \text{Broadcast} \mid m \rangle$ (Broadcasts a message m to all processes).
- **Indication**: $\langle rb, \text{Deliver} \mid p, m \rangle$ (Delivers a message m broadcast by process p)
  Properties:
- **RB1** (validity): if a correct process p broadcasts a message m, then p eventually delivers m.
- **RB2** (no duplication): no message is delivered more than once.
- **RB3** (no creation): if a process delivers a message m with sender s, then m was previously broadcast by process s.
- **RB4** (agreement): if a message m is delivered by some correct process, then m is eventually delivered by every correct process.

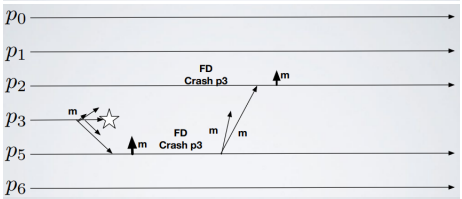
BEB vs RB:

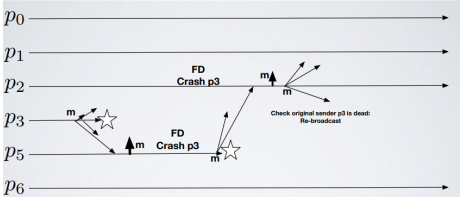
Implementation in Fail-Stop (Algorithm)



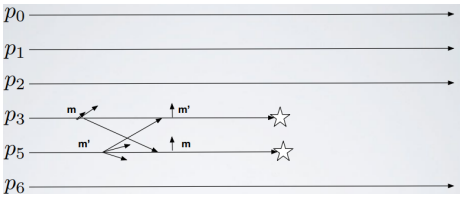
Examples of executions:

1. 

2. 



Difference between 1 and 2: in the first the message is delivered after the FD Crash, in the second case the message is delivered before the crash. In both cases it works

3. 

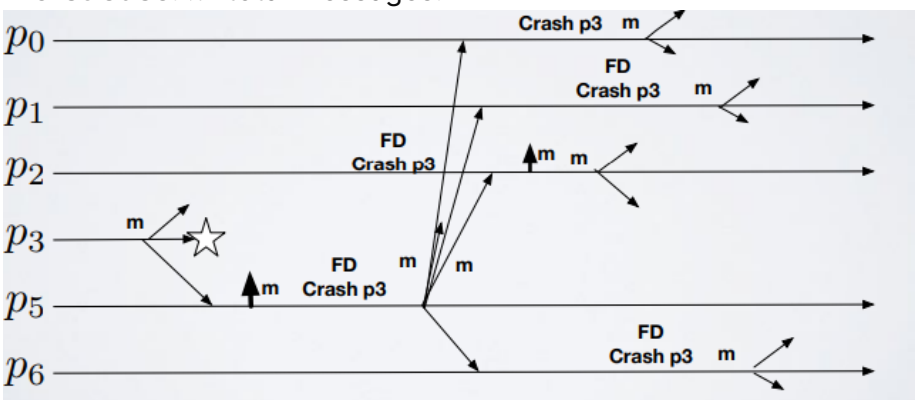
Also this execution is correct

Proof

- **Validity and No Creation** are ensured by th **BEB Broadcast**.
- **No Duplication**: by the check - *if from[p]* - in the Delivery handler from BEB.
- **Agreement**: suppose by contradiction that process p is correct and delivers message m with original sender q, while another correct p' doest not deliver m. What happens?
  1. if q does not crash then, by BEB also p' delivers m.
  2. if q crashes and q detects the crash before receiving m, then p relays message m. By BEB p' also delivers m.
  3. if q crashed and q detects the crash after delivering m, then p also relays message m. By BEB p' also delivers m.

Complexity

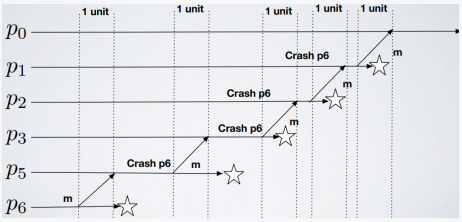
- **Best Case** n point-to-point messages.
- **Worst Case**: n^2 total messages.



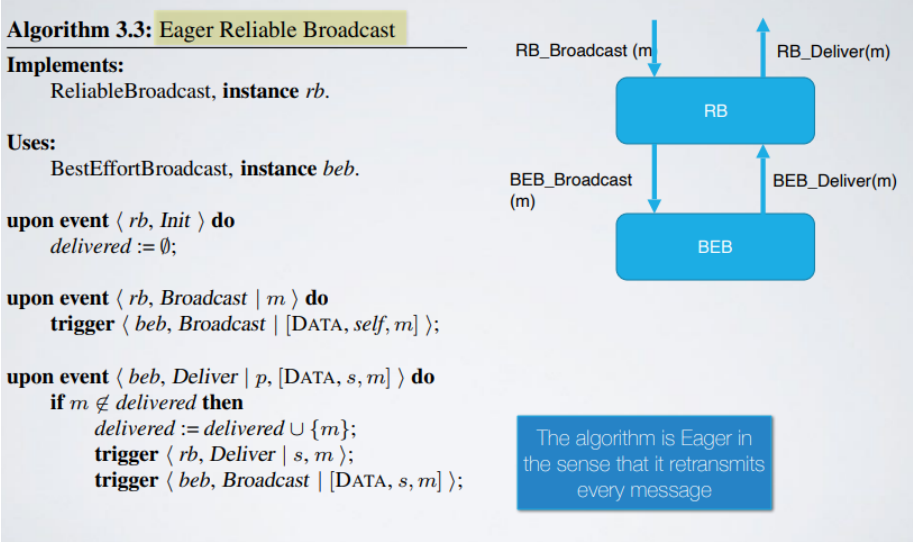
Example:

Best Case: 1 step (0 failures):

Worst Case: n steps



Implementation in Fail-Silent (Algorithm)



Algorithm 3.3: Eager Reliable Broadcast

Implements:
ReliableBroadcast, **instance** *rb*.

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

upon event ⟨ *rb*, *Init* ⟩ **do**
delivered := ∅;

upon event ⟨ *rb*, *Broadcast* | m ⟩ **do**
trigger ⟨ *beb*, *Broadcast* | [DATA, *self*, m] ⟩;

upon event ⟨ *beb*, *Deliver* | p, [DATA, s, m] ⟩ **do**
if $m \notin$ *delivered* **then**
delivered := *delivered* ∪ {m};
trigger ⟨ *rb*, *Deliver* | s, m ⟩;
trigger ⟨ *beb*, *Broadcast* | [DATA, s, m] ⟩;

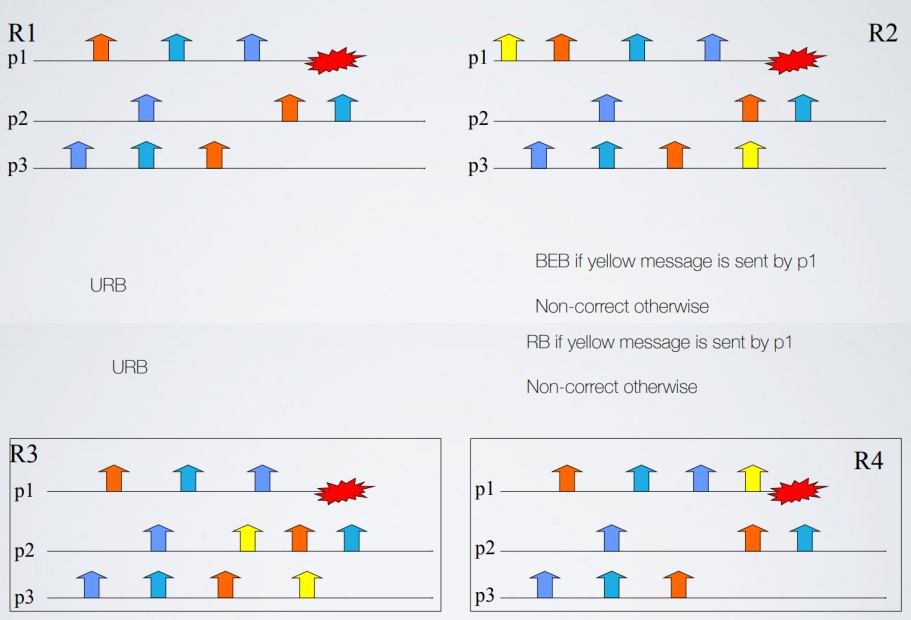
BEST CASE = WORST CASE - N BEB messages per one RB (N^2 point to point messages)
Deelay:

- besta case: 1 step
- worst case: $O(n)$ steps

Uniform Reliable Broadcast (URB)

properties:

- URB1-URB3 = RB1-RB3
- **URB4** (Uniform Agreement): if a message m is delivered by some process (whether correct or faulty), then m is eventually delivered by every correct process.



Algorithm

Algorithm 3.4: All-Ack Uniform Reliable Broadcast

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

Uses:
BestEffortBroadcast, **instance** *beb*.
PerfectFailureDetector, **instance** $\mathcal{P}$.

upon event $\langle$ *urb*, *Init* $\rangle$ **do**
delivered := $\emptyset$;
pending := $\emptyset$;
correct := Π;
forall m **do** $ack[m] := \emptyset$;

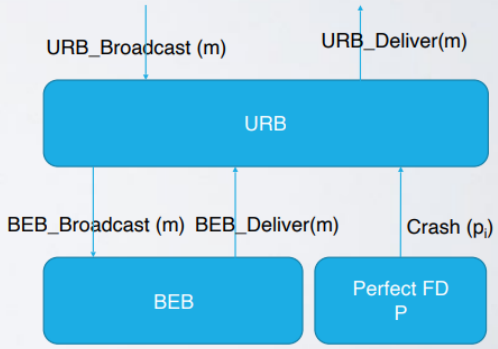
upon event $\langle$ *urb*, *Broadcast* | m $\rangle$ **do**
pending := *pending* $\cup \{(self, m)\}$;
trigger $\langle$ *beb*, *Broadcast* | [DATA, *self*, m] $\rangle$;

upon event $\langle$ *beb*, *Deliver* | p, [DATA, s, m] $\rangle$ **do**
$ack[m] := ack[m] \cup \{p\}$;
if $(s, m) \notin$ *pending* **then**
pending := *pending* $\cup \{(s, m)\}$;
trigger $\langle$ *beb*, *Broadcast* | [DATA, s, m] $\rangle$;

upon event $\langle \mathcal{P}$, *Crash* | $p \rangle$ **do**
correct := *correct* $\setminus \{p\}$;

function *candeliver*(m) **returns** Boolean **is**
return (*correct* $\subseteq$ $ack[m]$);

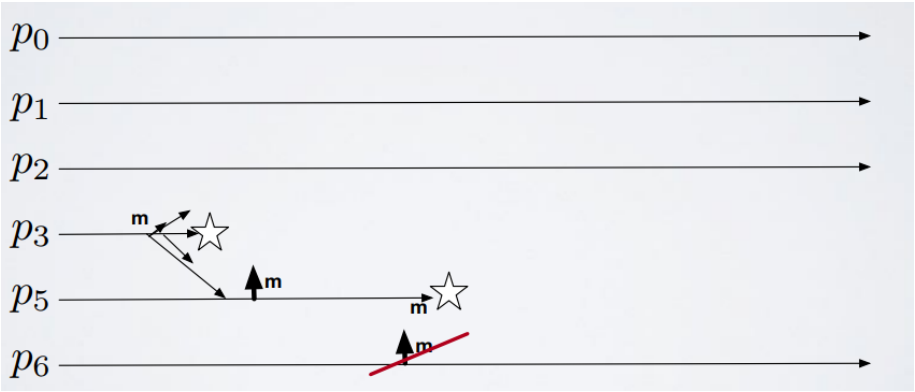
upon exists $(s, m) \in$ *pending* such that *candeliver*(m) $\wedge$ $m \notin$ *delivered* **do**
delivered := *delivered* $\cup \{m\}$;
trigger $\langle$ *urb*, *Deliver* | s, m $\rangle$;



Statement: if a correct process p, sees a message m it will eventually deliver it.ù
proof: by assumption p is correct, thus its message will reach every other correct process (see BEB). By the strong completeness of FD, P eventually process p detects as crashed all the crashed processes, thus it will not waits forever for an ack of a crashed process. Therefore, candeliver(m) on p will be true.

Agreement proof (by contradiction): suppose $p5$ (faulty) delivers m and $p6$ correct does not. The only possibility is that $p6$ does not see the message (**statement**). This implies that $p6$ is detected faulty by $p5$, $p5$ delivers without receiving the ack from $p6$. This **contradicts the strong**

accuracy of the failure detector P.



Costs:

- BEST CASE = WORST CASE - N BEB messages per one RB (N^2 point to point messages).
- Deelays:
  - **Best case**: 2 step (0 failures) (1 for disseminate 1 for ACKs).
  - **Worst case**: $O(n)$ steps (chain of failures as in the lazy RB).