



Distributed Systems

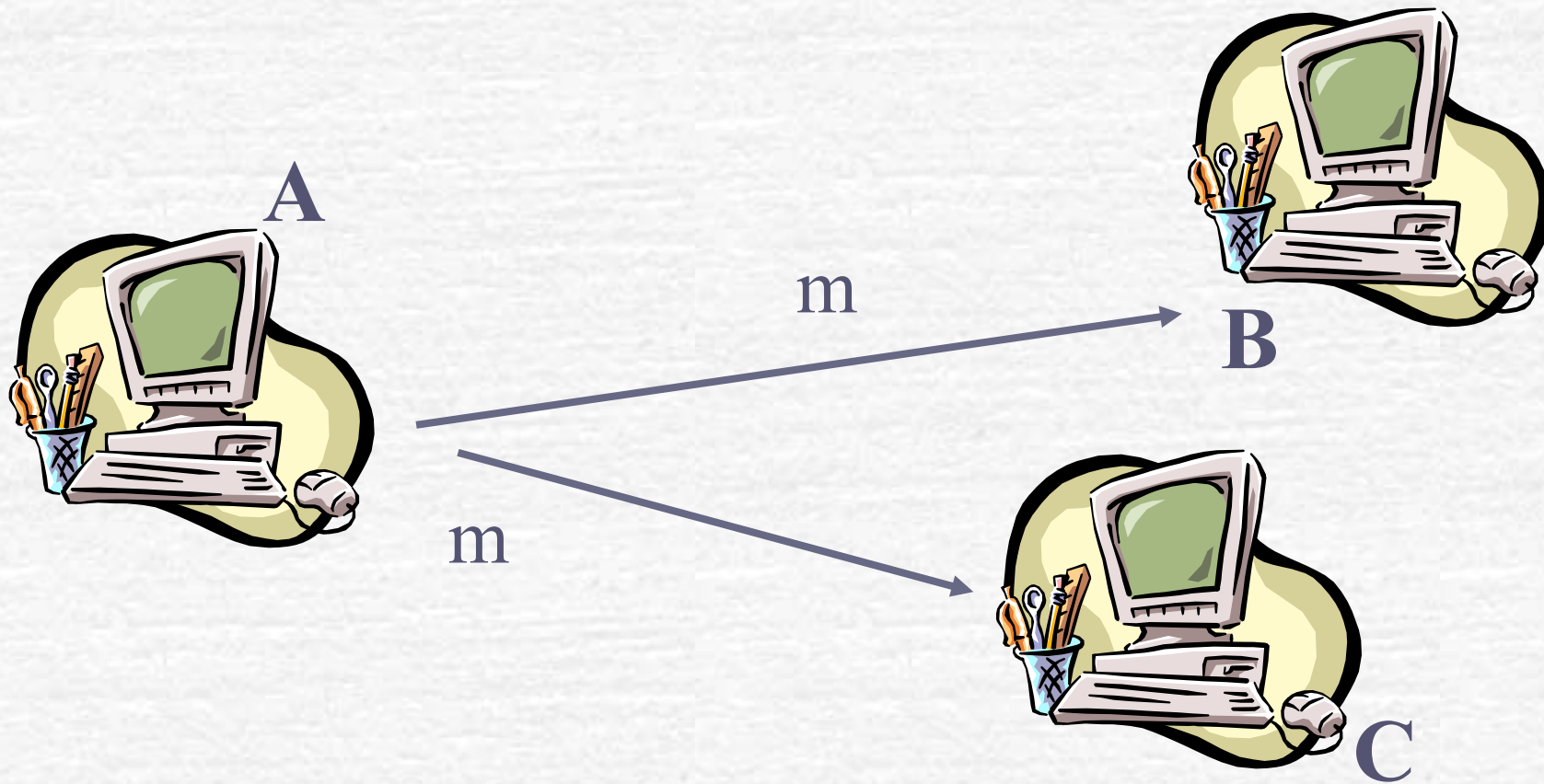
Terminating Reliable Broadcast

Prof R. Guerraoui

Distributed Programming Laboratory



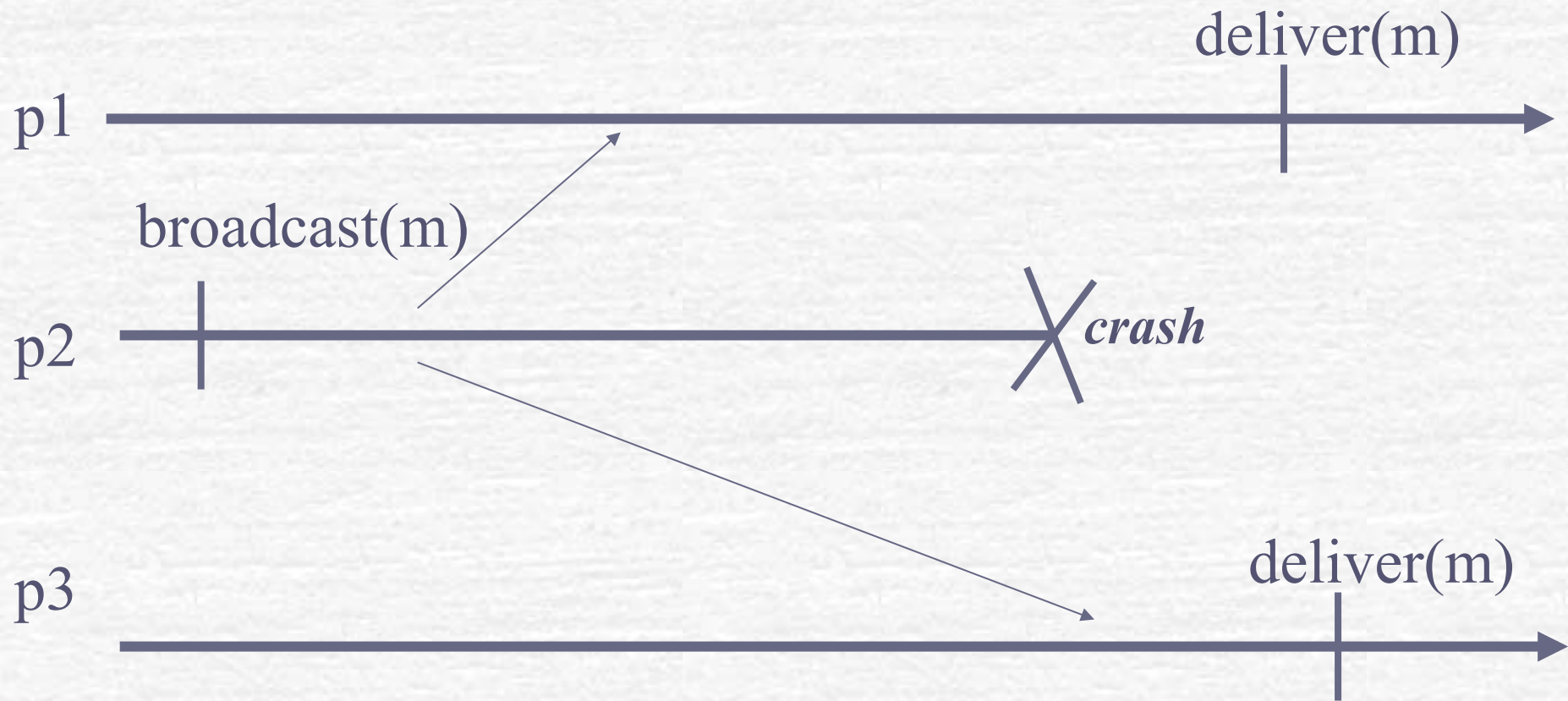
Terminating Reliable Broadcast



Terminating Reliable Broadcast

- Like reliable broadcast, terminating reliable broadcast (TRB) is a communication primitive used to disseminate a message among a set of processes in a reliable way
- TRB is however strictly stronger than (uniform) reliable broadcast

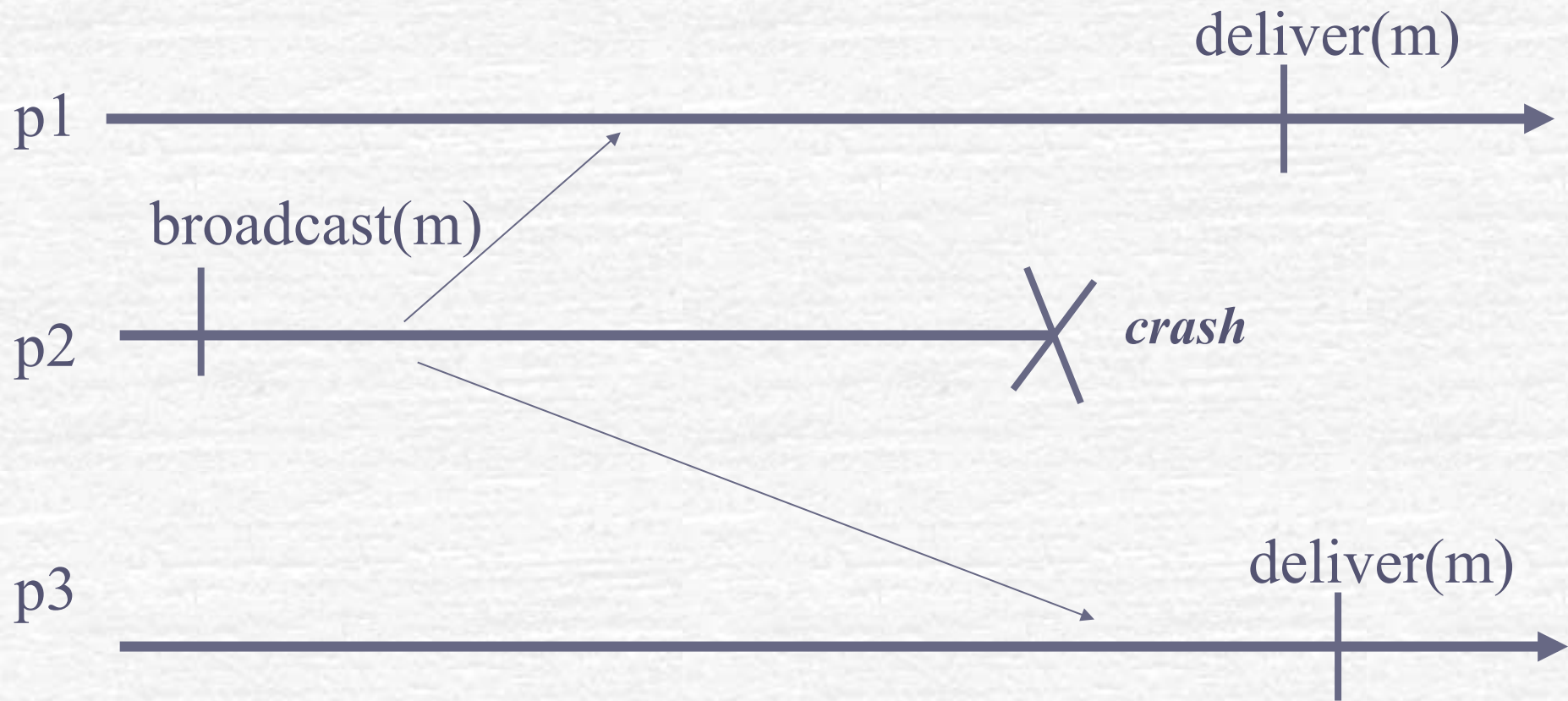
(Uniform) Reliable Broadcast



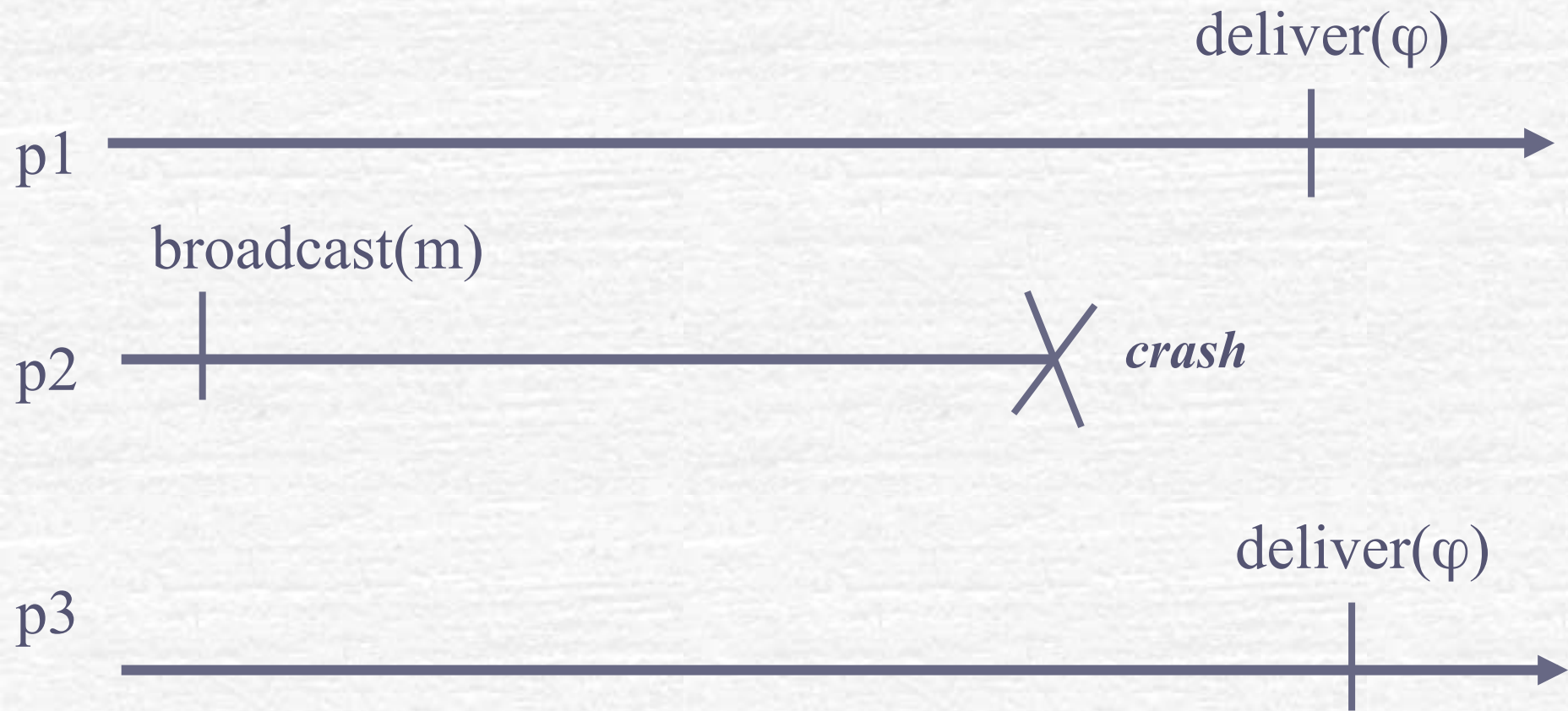
(Uniform) Reliable Broadcast



Terminating Reliable Broadcast



Terminating Reliable Broadcast



Terminating Reliable Broadcast

- **Like** with reliable broadcast, correct processes in TRB agree on the set of messages they deliver
- **Like** with (uniform) reliable broadcast, every correct process in TRB delivers every message delivered by any process
- **Unlike** with reliable broadcast, every correct process delivers a message, even if the broadcaster crashes

Terminating Reliable Broadcast

- The problem is defined for a specific broadcaster process $p_i = \text{src}$ (known by all processes)
- Process src is supposed to broadcast a message m (distinct from φ)
- The other processes need to deliver m if src is correct but may deliver φ if src crashes

Terminating Reliable Broadcast (π)

TRB1. Integrity: If a process delivers a message m , then either m is φ or m was broadcast by src

TRB2. Validity: If the sender src is correct and broadcasts a message m , then src eventually delivers m

TRB3. (Uniform) Agreement: For any message m , if a correct (any) process delivers m , then every correct process delivers m

TRB4. Termination: Every correct process eventually delivers exactly one message

Terminating Reliable Broadcast

• *Events*

• Request: $\langle \text{trbBroadcast}, m \rangle$

• Indication: $\langle \text{trbDeliver}, p, m \rangle$

• *Properties:*

• *TRB1, TRB2, TRB3, TRB4*

Algorithm (trb)

- ✦ **Implements:** trbBroadcast (trb).
- ✦ **Uses:**
 - ✦ BestEffortBroadcast (beb).
 - ✦ PerfectFailureDetector (P).
 - ✦ Consensus(cons).
- ✦ **upon event** < Init > **do**
 - ✦ prop := \perp ;
 - ✦ correct := S;

Algorithm (trb – cont'd)

• **upon event** $\langle \text{trbBroadcast}, m \rangle$ **do**

• **trigger** $\langle \text{bebBroadcast}, m \rangle$;

• **upon event** $\langle \text{crash}, \text{src} \rangle$ and $(\text{prop} = \perp)$ **do**

• $\text{prop} := \varphi$;

Algorithm (trb – cont'd)

• **upon event** <bebDeliver, src, m> and (prop = \perp) **do**

• prop := m;

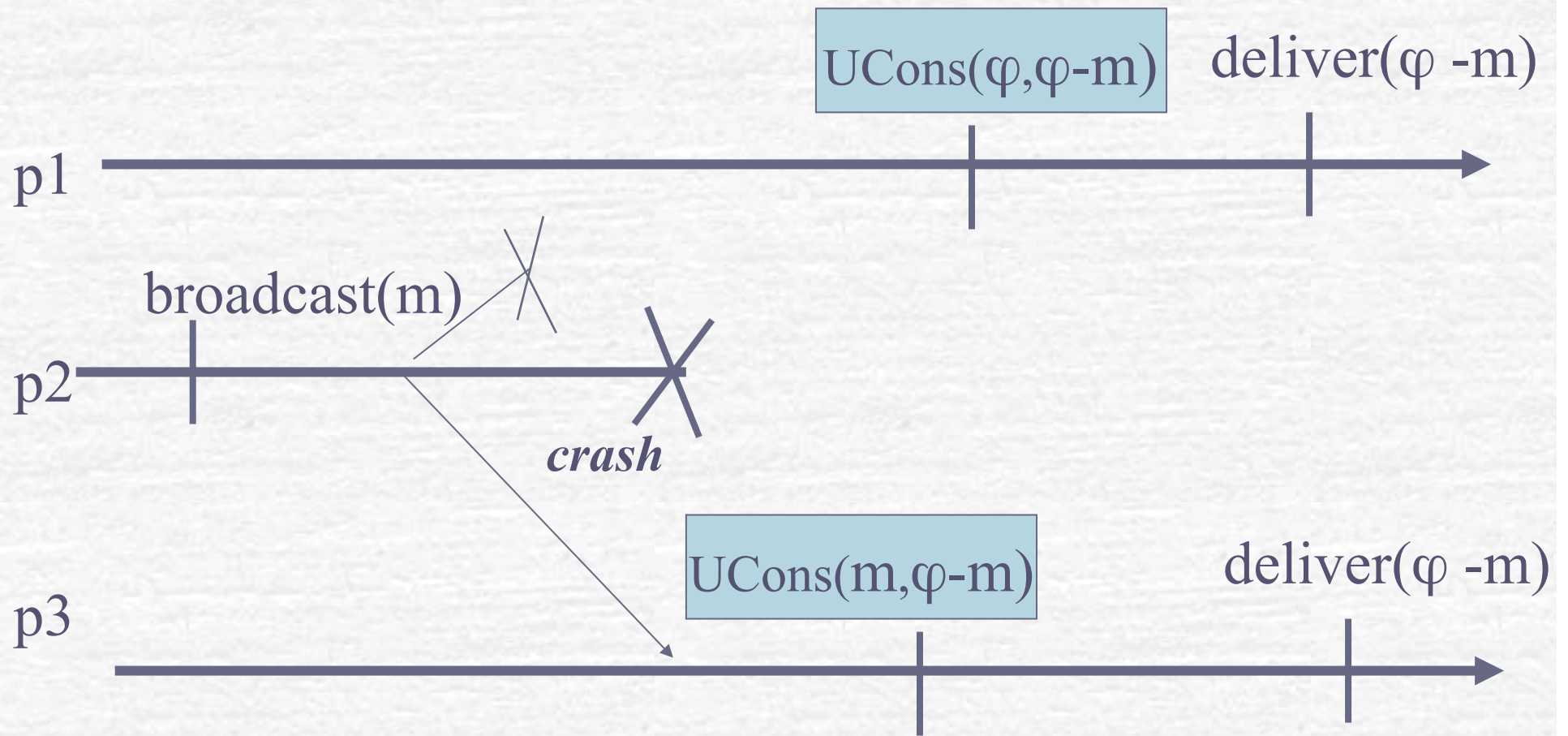
• **upon event** (prop $\neq \perp$) **do**

• **trigger** < Propose, prop>;

• **upon event** < Decide, decision> **do**

• **trigger** < trbDeliver, src, decision>;

Algorithm (trb); src = p2



Terminating Reliable Broadcast

- Our TRB algorithm uses the perfect failure detector P (i.e., P is sufficient)
- Is P also necessary?
 - Is there an algorithm that implements TRB with a failure detector that is strictly weaker than P ? (this would mean that P is not necessary)
 - Is there an algorithm that uses TRB to implement P (this would mean that P is necessary)

Terminating Reliable Broadcast

- We give an algorithm that implements **P** using **TRB**; more precisely, we assume that every process p_i can use an infinite number of instances of TRB where p_i is the sender src
 - 1. Every process p_i keeps on trbBroadcasting messages $m_{i1}, m_{i2}, \text{ etc}$
 - 2. If a process p_k delivers φ_i , p_k suspects p_i
 - NB. The algorithm uses (non-uniform) TRB