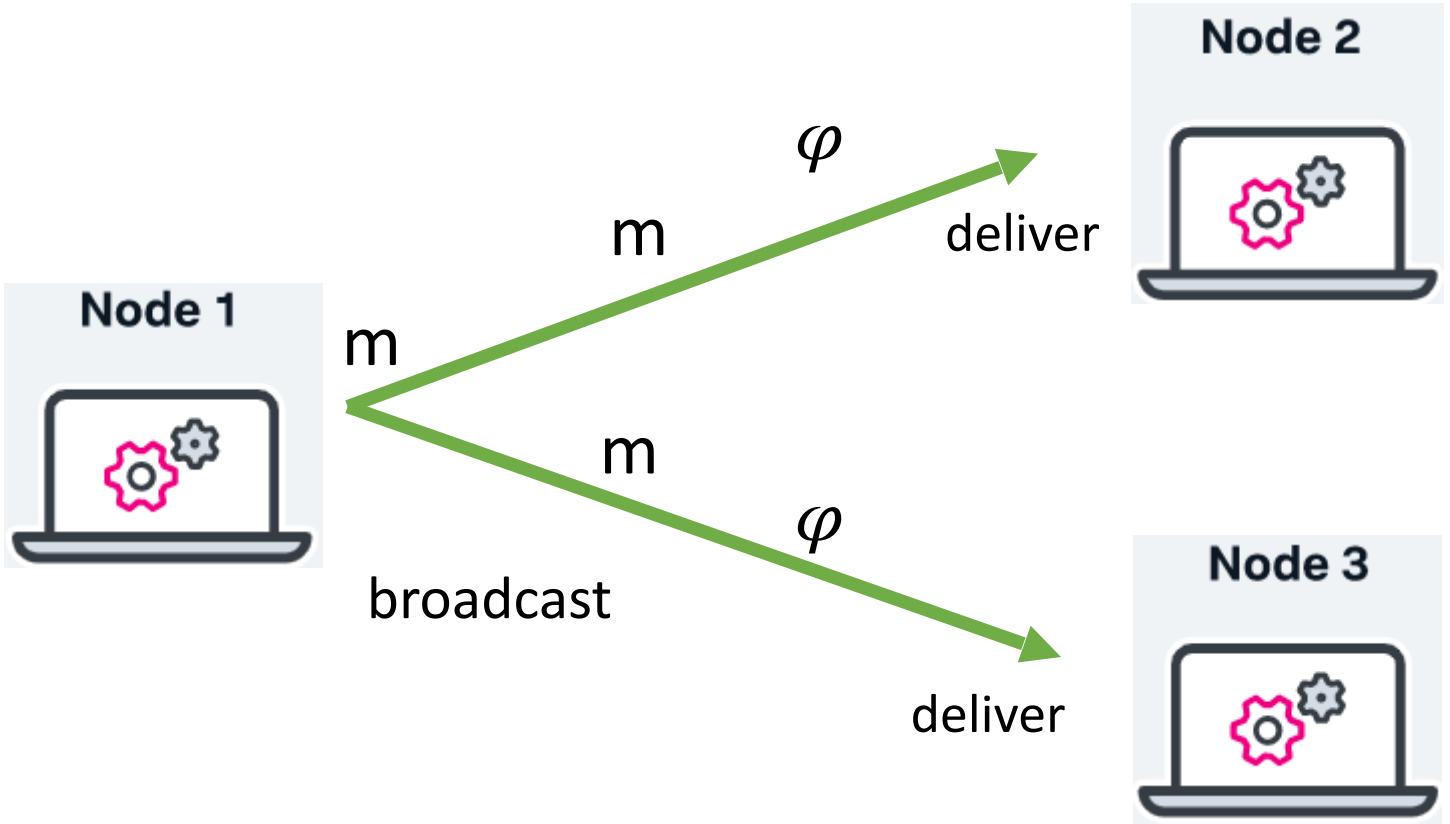
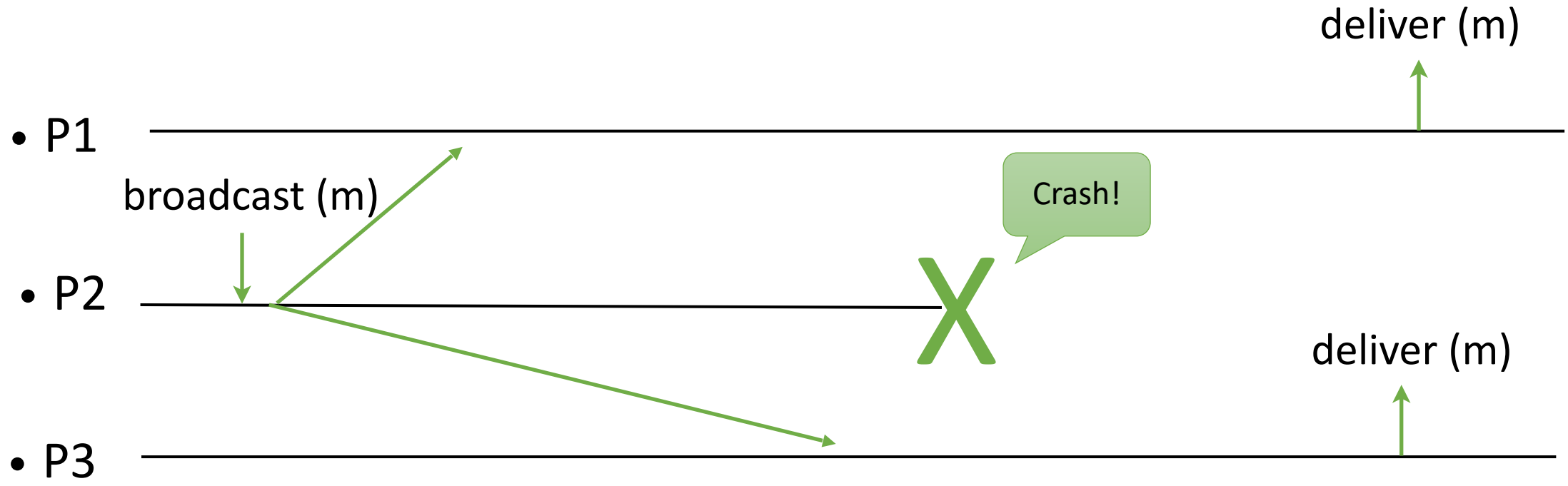
Terminating Reliable Broadcast

Mohsen Lesani

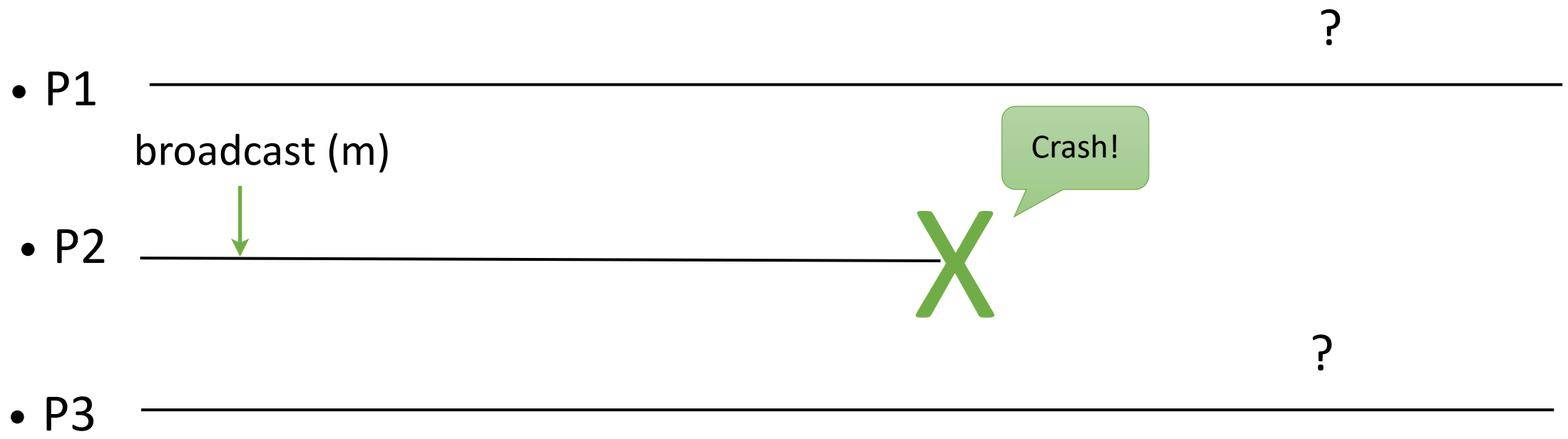
Terminating Reliable Broadcast



(Uniform) Reliable Broadcast



(Uniform) Reliable Broadcast



The processes p1 and p3 are never sure whether a message will arrive.

Terminating Reliable Broadcast

- In RB, a process p has no means to distinguish the case where some process q has delivered m (and by agreement p can indeed wait for m), from the case where no process will ever deliver m (and p should definitely not keep waiting for m).

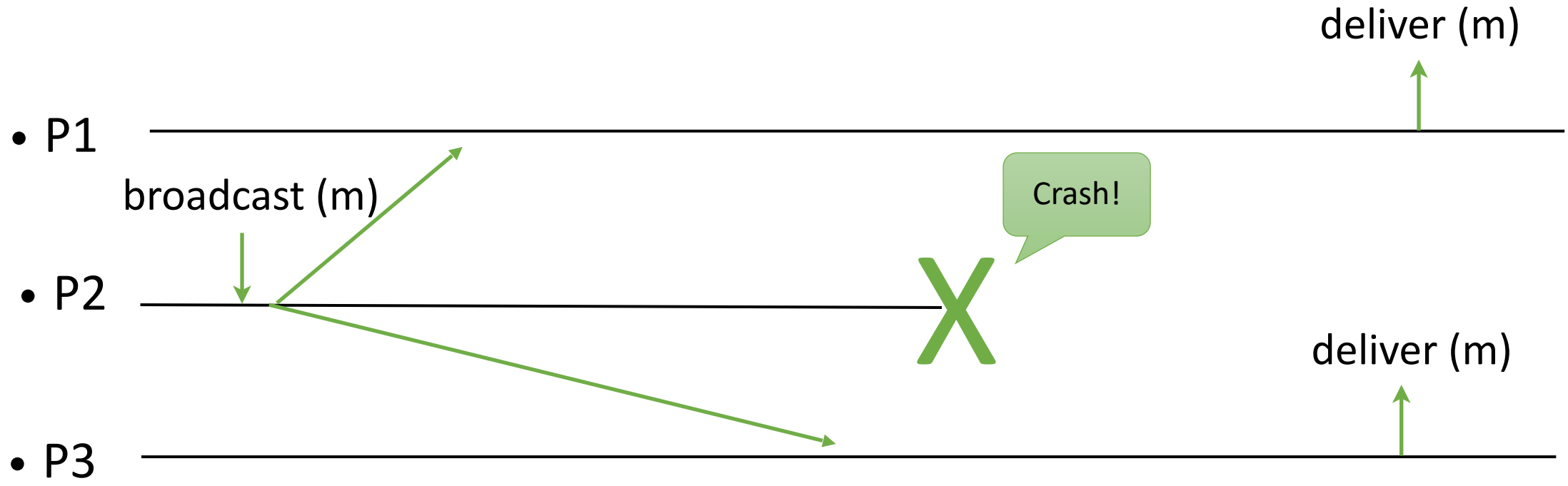
- TRB ensures that every process p either delivers the message m from the sender or some failure indication, denoting that m will never be delivered (by any process).

- TRB is however strictly stronger than (U)RB.

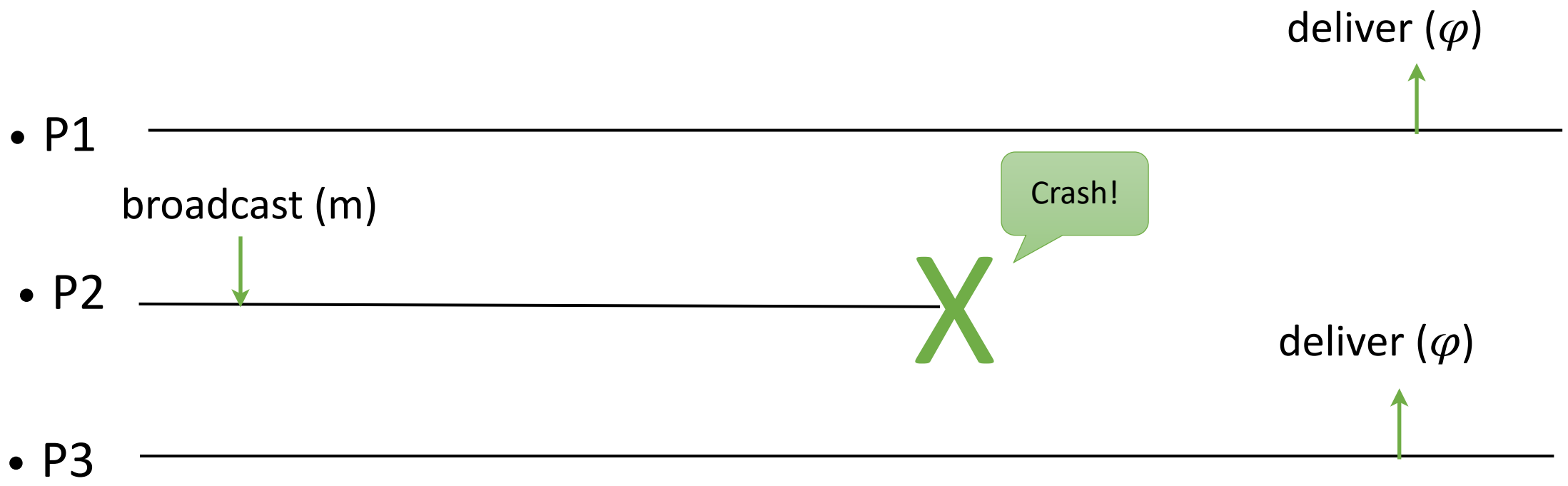
Terminating Reliable Broadcast

- **Like** reliable broadcast, agreement: correct processes in TRB agree on the set of messages they deliver.
- **Like** (uniform) reliable broadcast, uniform agreement: if a (correct or incorrect) process delivers a message, then every correct process delivers it.
- **Unlike** reliable broadcast, every correct process delivers a message, even if the broadcaster crashes.

(Uniform) Reliable Broadcast



Terminating Reliable Broadcast



Terminating Reliable Broadcast

- The problem is defined for a specific broadcaster process 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

- **Events**
  - Request: <broadcast, m>
  - Indication: <deliver, p, m>
- **Properties:**
  - **TRB1, TRB2, TRB3, TRB4**

Terminating Reliable Broadcast

- **TRB1. Validity**: If the sender src is correct and broadcasts a message m, then every correct process eventually delivers m.
- **TRB2. Termination**: Every correct process eventually delivers exactly one message.
- **TRB3. Integrity**: If a process delivers a message m, then either m is φ, or m was broadcast by src.
- **TRB4. (Uniform) Agreement**: For any message m, if a correct (any) process delivers m, then every correct process delivers m.

Integrity is similar to RB No creation, and Termination includes RB No duplication.

Terminating Reliable Broadcast

Idea:

- Wait to either receive the message, or hear that the src has crashed. Accordingly, propose either the message m or none φ to consensus.

TRB

Implements: broadcast (trb).

Uses:

BestEffortBroadcast (beb)

PerfectFailureDetector (P)

Consensus (cons)

upon event < Init > **do**

prop := ⊥

correct := S

upon event < broadcast(m) >

trigger < beb, broadcast(m) >

TRB

upon event < P, crash(src) > $\wedge$ (prop = $\perp$) **do**

    prop := φ

upon event < beb, deliver(src, m) > **do**

    prop := m

To propose either m or φ and prefer m over φ.

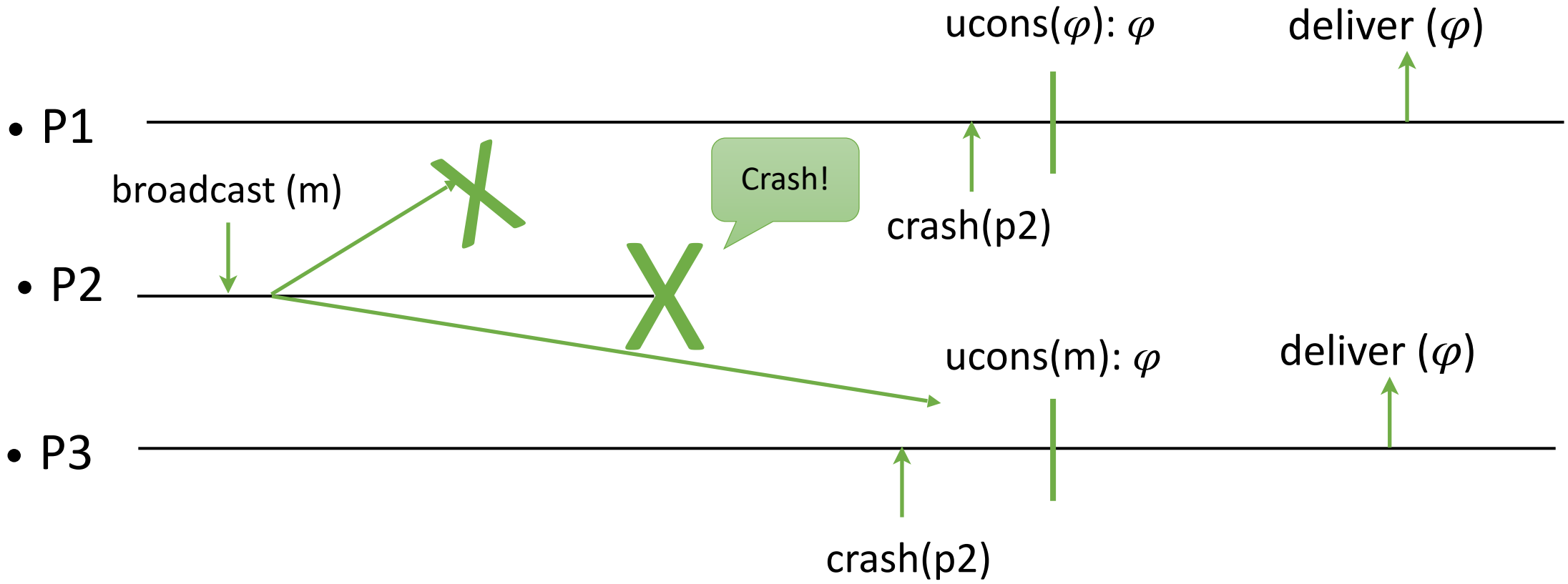
upon event (prop $\neq \perp$) **do**

    trigger < cons, propose(prop) >

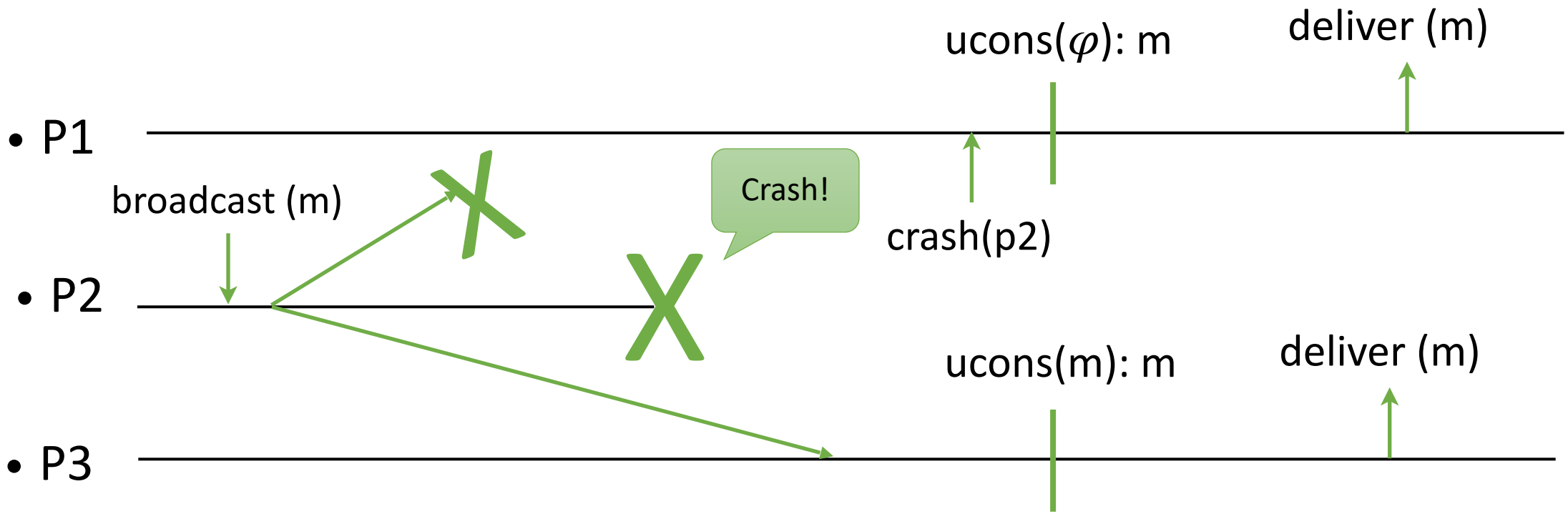
upon event < decide (decision) > **do**

    trigger < deliver (src, decision) >

Algorithm (trb); src = p2



Algorithm (trb); src = p2



Synchrony

- 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)

Synchrony

TRB cannot be implemented by <>P.

Proof by contradiction: <>P → TRB

From the next slide, we have TRB → P.

Thus, we have <>P → P, that is a contradiction.

Synchrony

We give an algorithm that implements **P** using **TRB.**

- We let every process pi use an infinite number of instances of TRB where pi is the sender.
- Every process pi keeps on trb broadcasting messages mi1, mi2, etc
- If a process pk delivers φ instead of an mi, pk suspects pi.
- The algorithm needs only non-uniform TRB.

Synchrony

Completeness

Termination and Integrity of TRB imply completeness of P:

We show that if src crashes, it is eventually suspected. By termination, each TRB in the sequence will eventually deliver a message. The message will be eventually φ because otherwise by integrity, src has been broadcasting messages and has not crashed.

When φ is received, src is suspected.

Accuracy

Validity and Termination of TRB implies Accuracy of P:

We show that if φ is delivered, src has crashed. By contradiction, if src is correct, by validity, a message $m \neq \varphi$ will be delivered, and by termination, at most one message is delivered.

Original slides adopted from R. Guerraoui