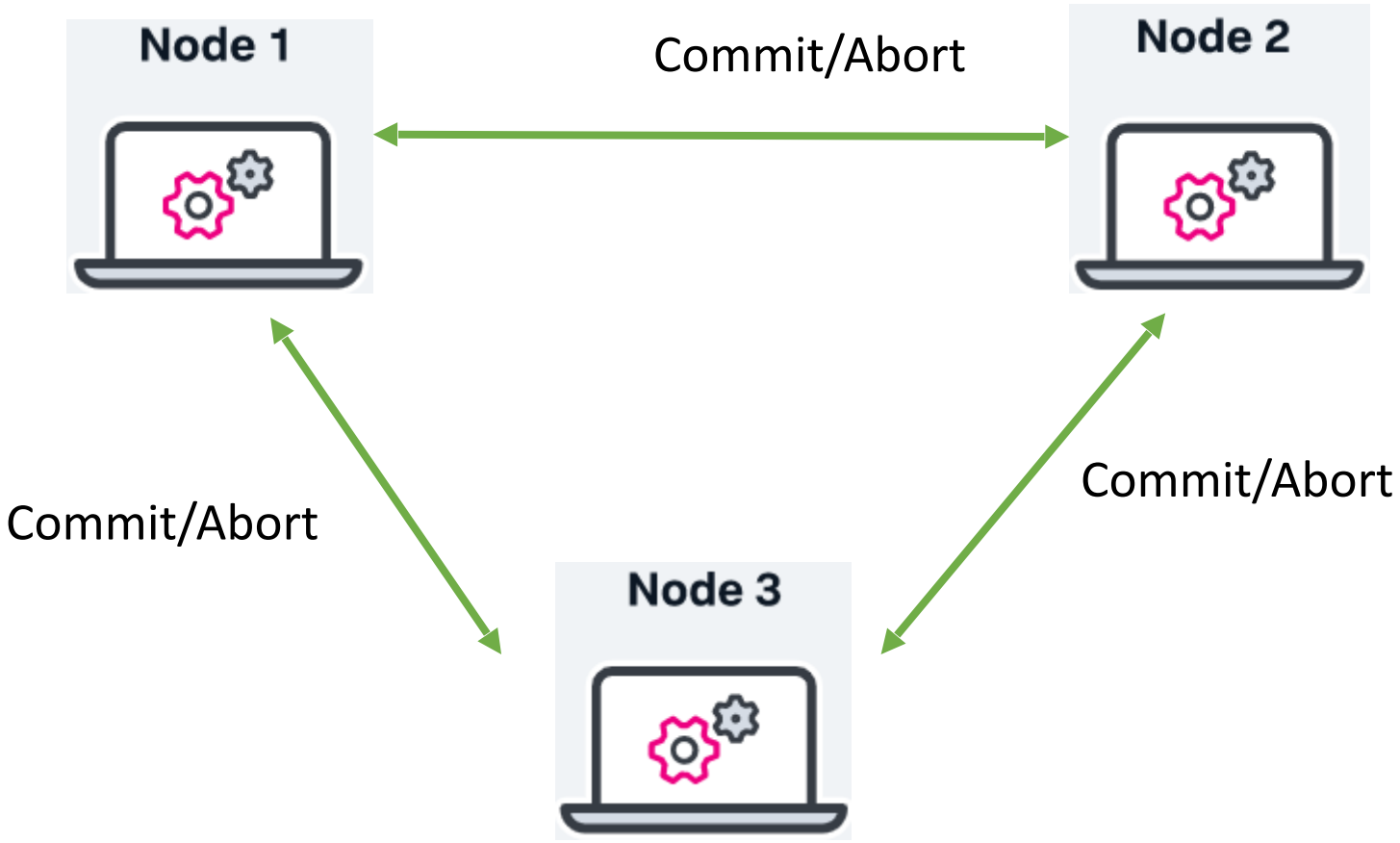
Atomic Commit

Mohsen Lesani

Atomic Commit: An Agreement Problem



Transactions (Gray)

- A transaction is a program describing a sequence of accesses to shared and distributed information
- A transaction should be executed atomically. It can be terminated either by committing or aborting.

Transactions

```
beginTransaction
alice.withdraw(1.000.000)
bob.deposit(1.000.000)
outcome := commitTransaction
if (outcome = abort) then …
```

ACID properties

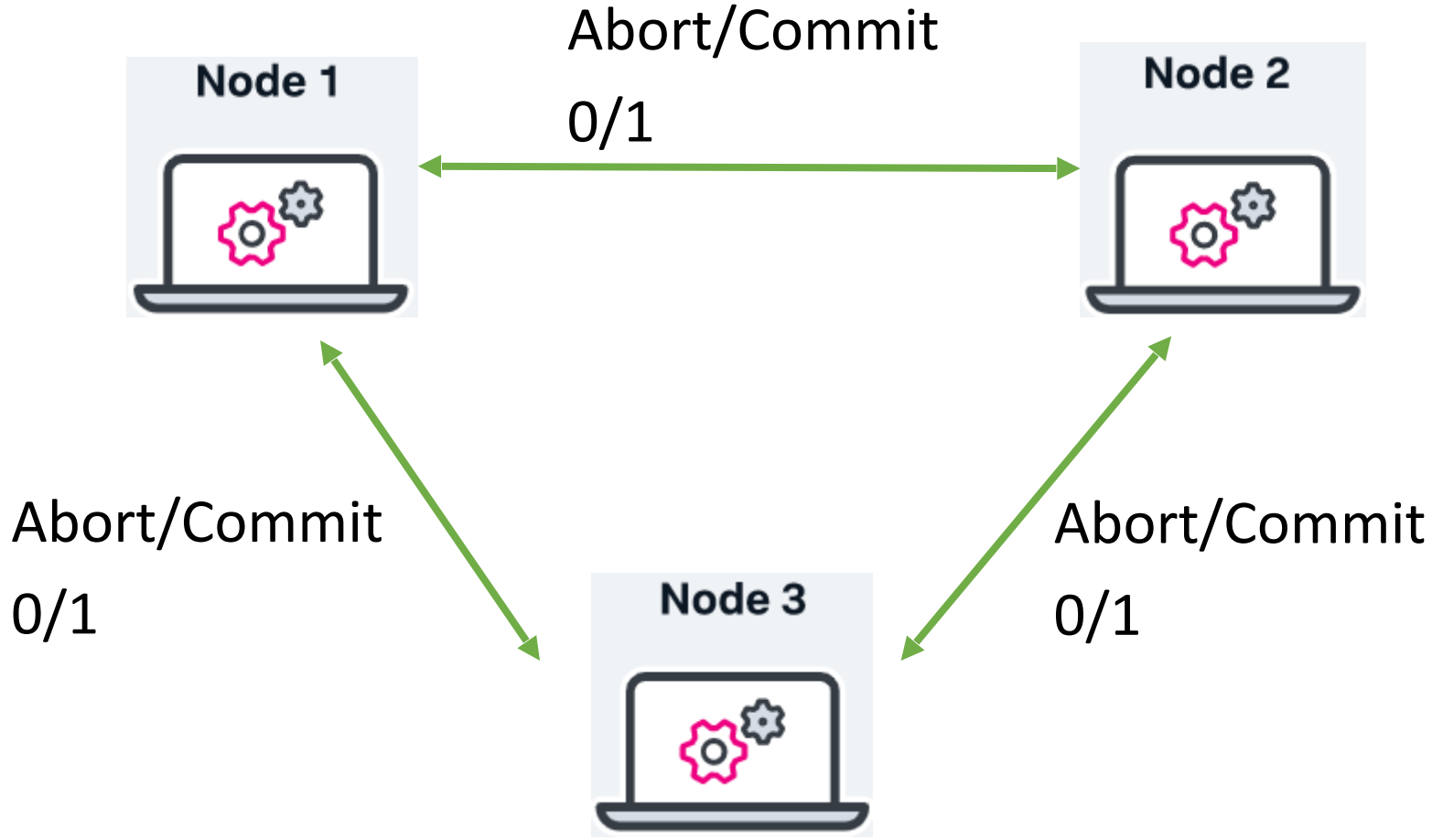
- **Atomicity**: a transaction either performs entirely or not at all.
- **Consistency**: a transaction transforms a consistent state into another consistent state.
- **Isolation**: a transaction appears to be executed in isolation, not exposing intermediate states.
- **Durability**: the effects of a transaction that commits are permanent.

The programmer should write transactions that assuming consistent state, return consistent state.

The Consistency Contract

If the programmer preserves the consistency locally, then the system deals with concurrency to preserve the consistency globally.

Distributed Transaction



Atomic Commit Events

- As in consensus, every process has an initial value 0 (no) or 1 (yes) and must decide on a final value 0 (abort) or 1 (commit).
- The proposition means the ability to commit the transaction.
- Unlike consensus, processes seek to decide 1 but every process has a veto right.

Atomic Commit

- **Events**
  - Request: <propose, v>
  - Indication: <pecide, v’>
  - v, v’ ∈ {Commit, Abort}

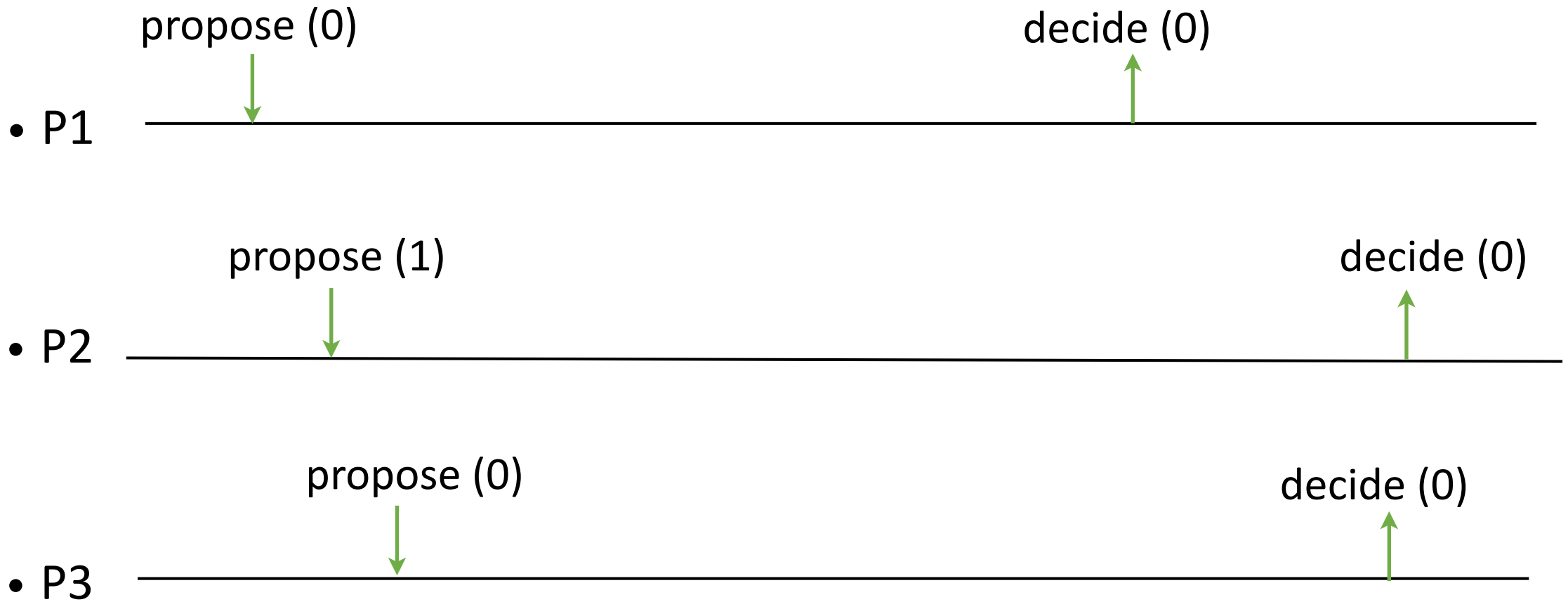
- **Properties:**
  - **NBAC1, NBAC2, NBAC3, NBAC4**

Atomic Commit Specification

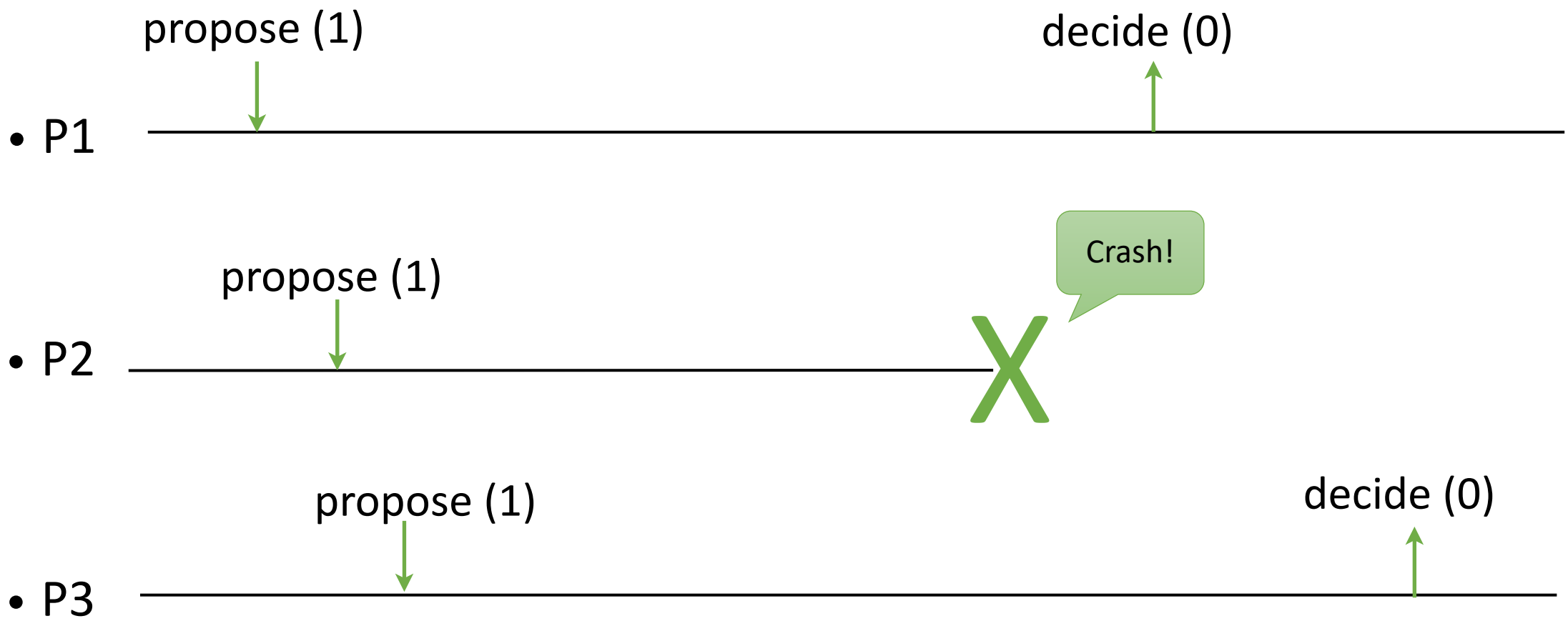
- **NBAC1. Termination**: Every correct process eventually decides.
- **NBAC2. Abort-Validity**: 0 can only be decided if some process crashes or votes 0.
- **NBAC3. Commit-Validity**: 1 can only be decided if all processes propose 1.
- **NBAC4. Integrity**: No process decides twice.
- **NBAC5. Uniform Agreement**: No two processes decide differently.

Abort-validity is the non-triviality property. Otherwise the decision could be always 0.

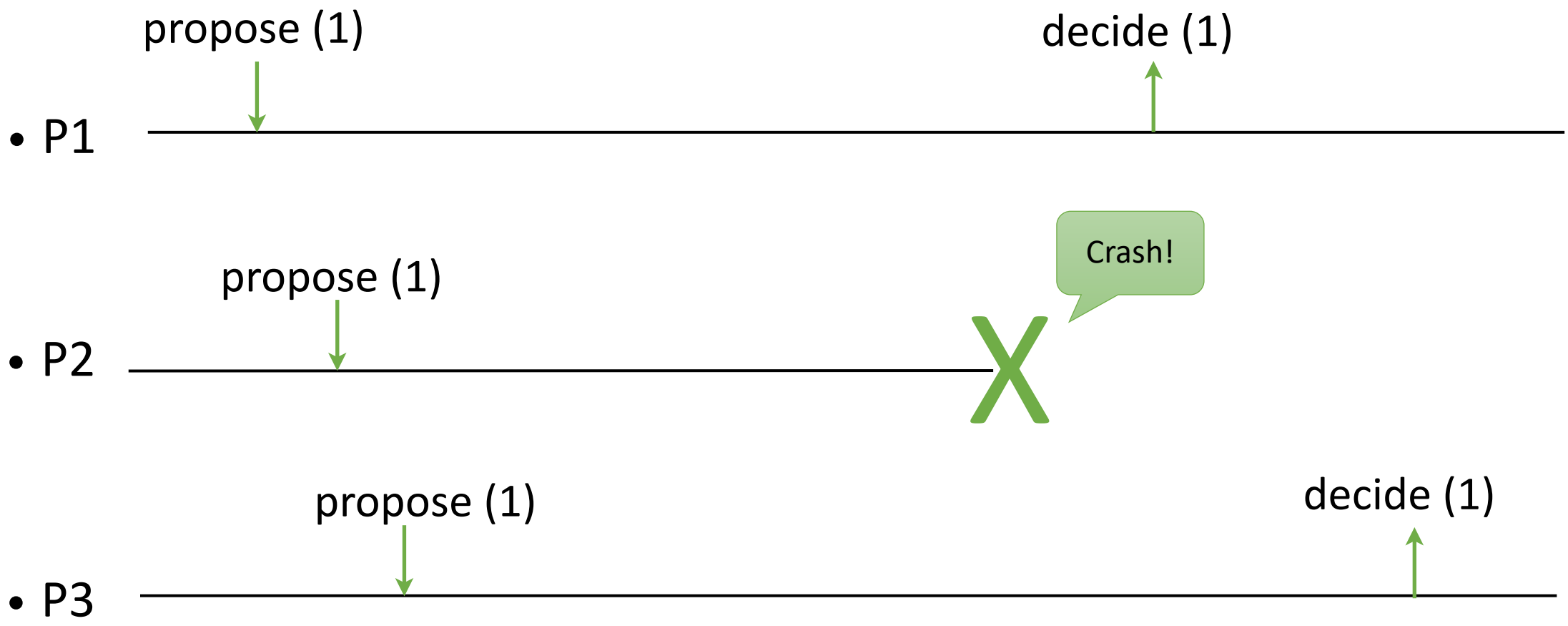
Atomic Commit Example 1



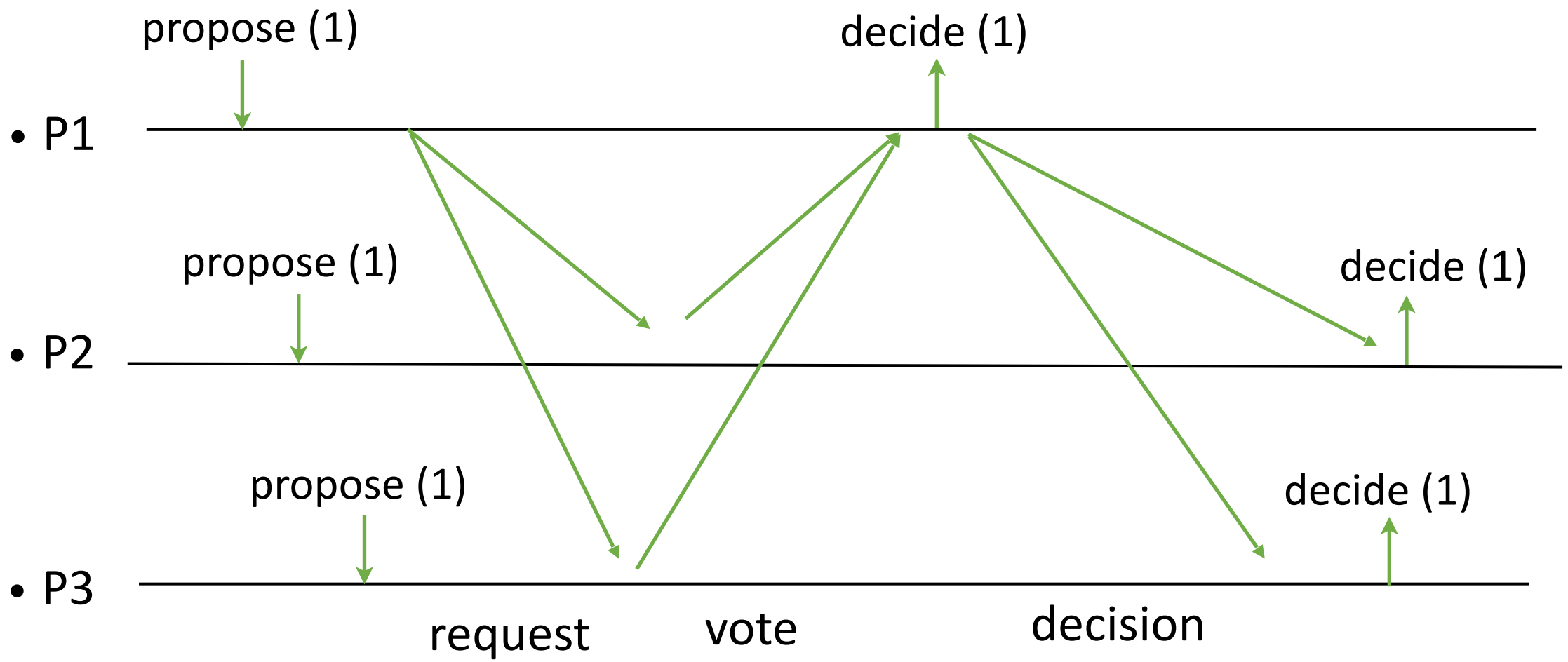
Atomic Commit Example 2



Atomic Commit Example 3

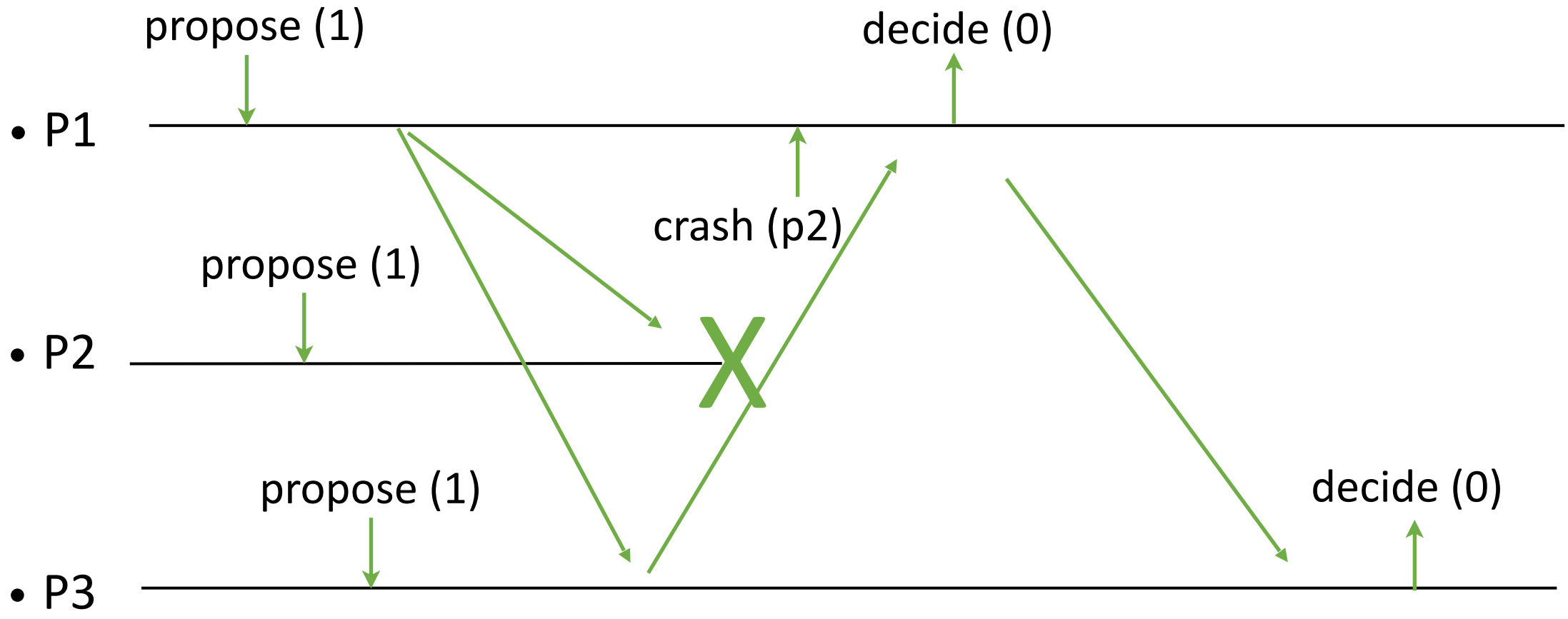


2-Phase Commit Protocol



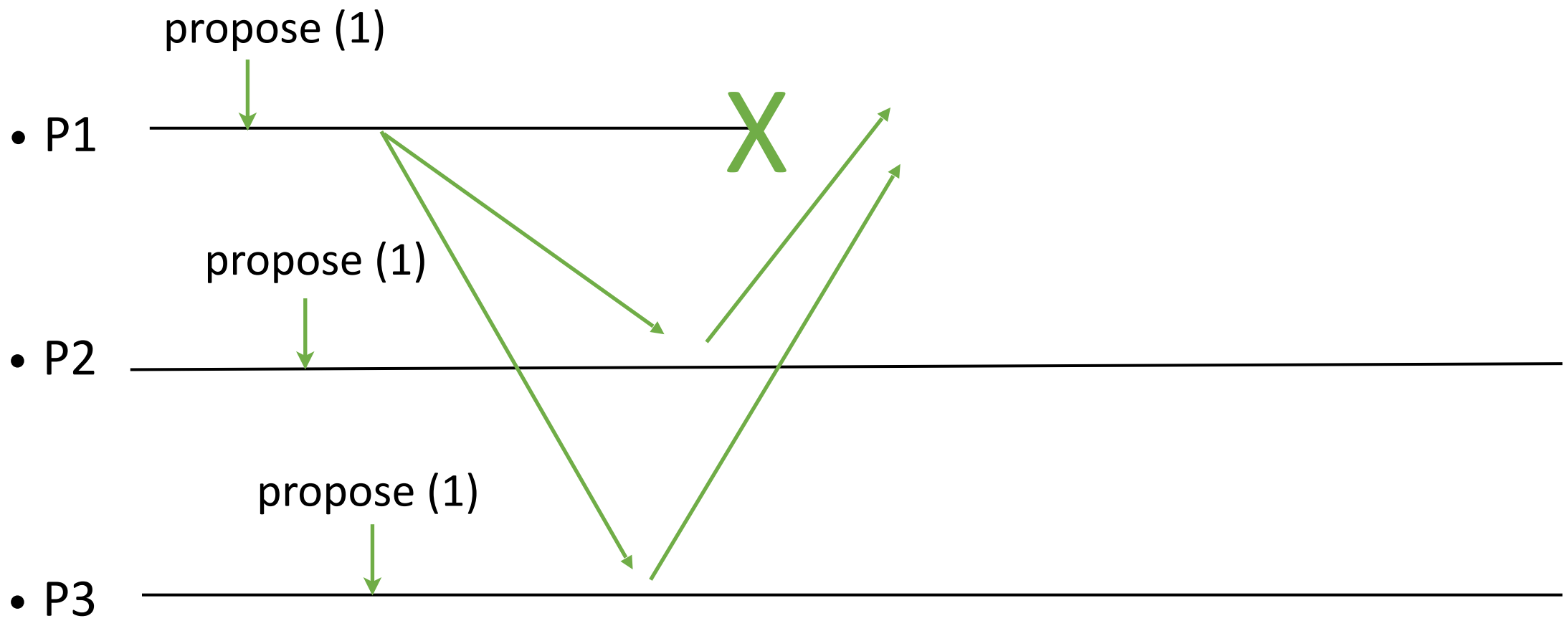
The request phase can be skipped if the leader is previously known.

2-Phase Commit Protocol



2PC uses the perfect failure detector P.

2-Phase Commit Protocol



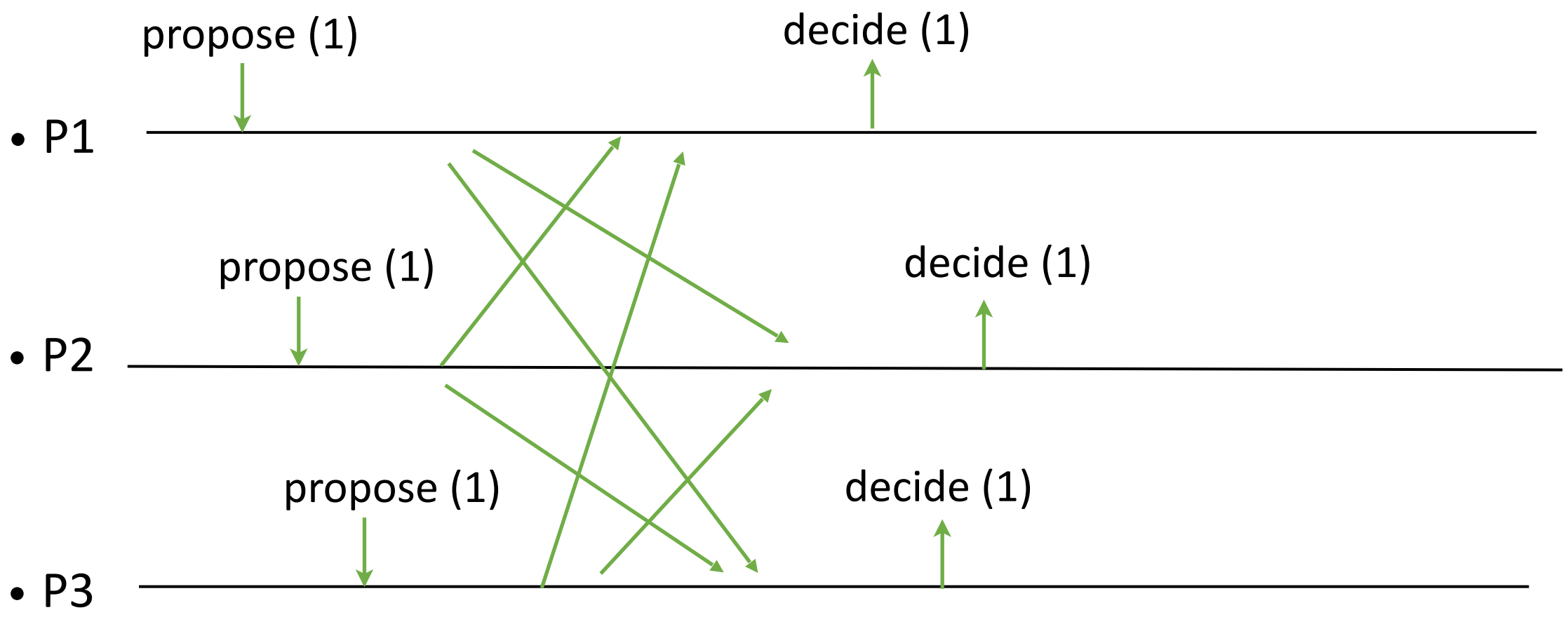
Week termination: If the leader crashes then the processes are blocked.

Non-Blocking Atomic Commit

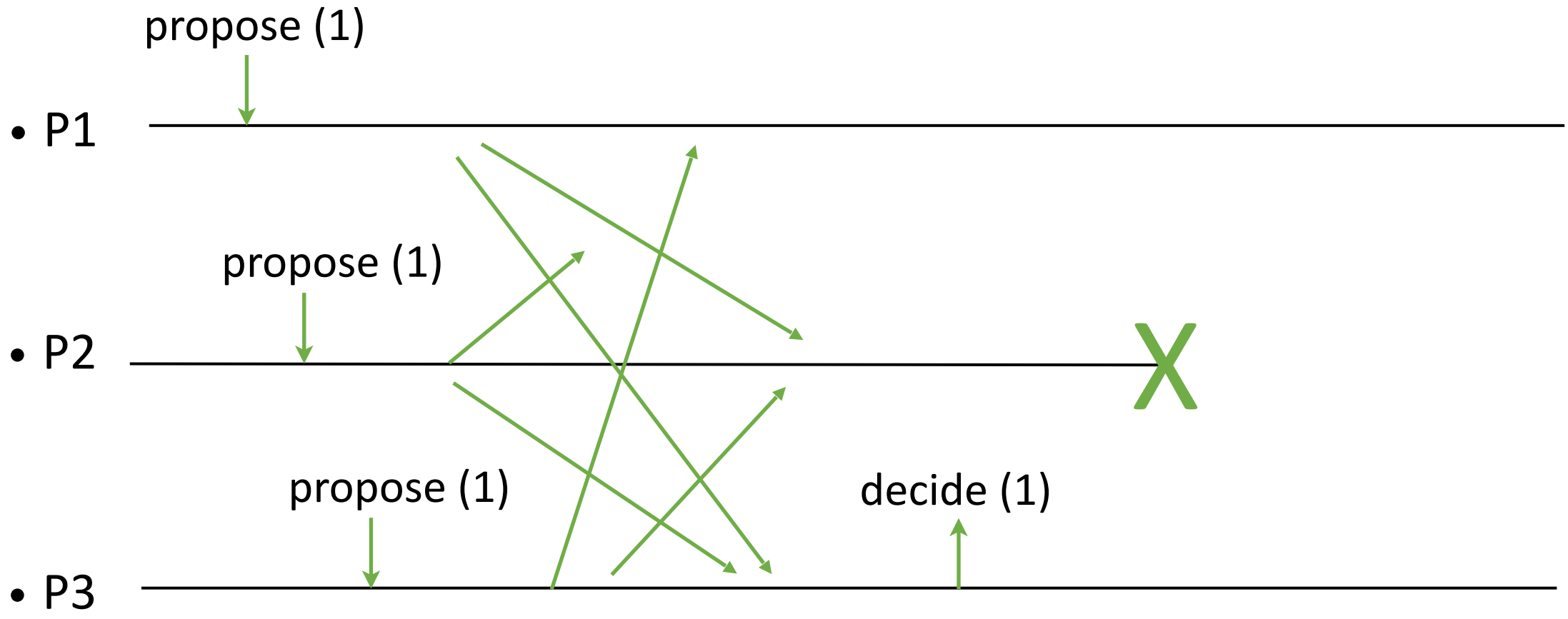
Idea:

- There cannot be just one leader. The leader should be distributed. Each process should act as a leader.

Broadcast?

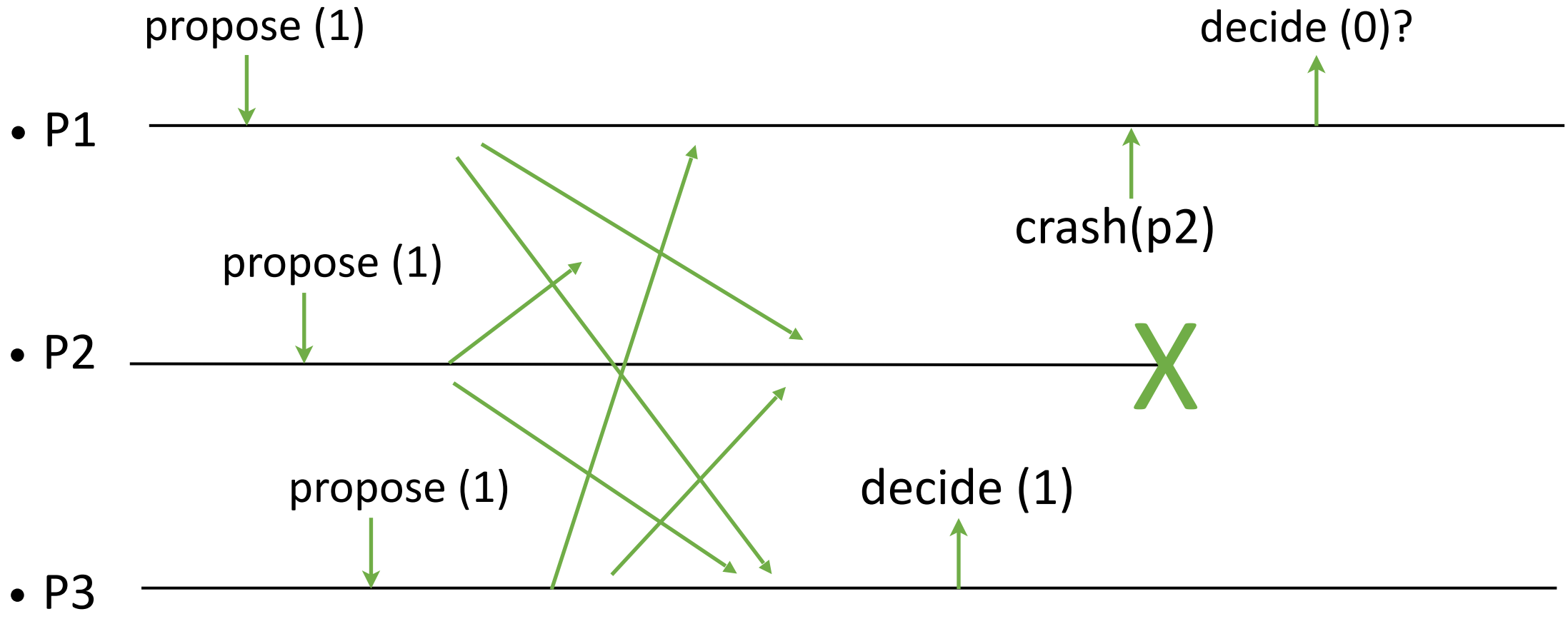

propose (1)
decide (1)
P1
propose (1)
decide (1)
P2
propose (1)
decide (1)
P3

Broadcast?



Very weak termination: Even if one process crashes, others can be blocked.

Broadcast?



If a failure detector is used, the issue is that agreement can be violated.

Non-Blocking Atomic Commit

Idea:

- There cannot be just one leader. The leader should be distributed. Each process should act as a leader.
- Processes broadcast their proposals. Each process arrives at a value based on the delivered proposals or crash of other processes. Processes may disagree because of unreliable broadcast by crashed processes. They use consensus to agree.

- By validity of consensus, the decided value is proposed.
- The value 1 is proposed only when the process receives the proposal 1 from all processes. This leads to commit-validity.
- The value 0 is proposed only when the process receives the proposal 0 from a process or its crash notification. This leads to abort-validity.

NBAC

Implements: NonBlockingAtomicCommit (nbac).

Uses:

BestEffortBroadcast (beb).

PerfectFailureDetector (P).

UniformConsensus (uc).

upon event < Init > **do**

prop = 1;

delivered = ∅; correct = Π

NBAC

upon event <propose(v)> **do**
    trigger <beb, broadcast(v)>

upon event <beb, deliver(pi, v)> **do**
    delivered = delivered U {pi}
    prop = prop * v

upon event <P, crash(pi)> **do**
    correct = correct \ {pi}
    Prop = 0

A crashed process or delivery of a zero proposal vetoes the current proposal.

NBAC

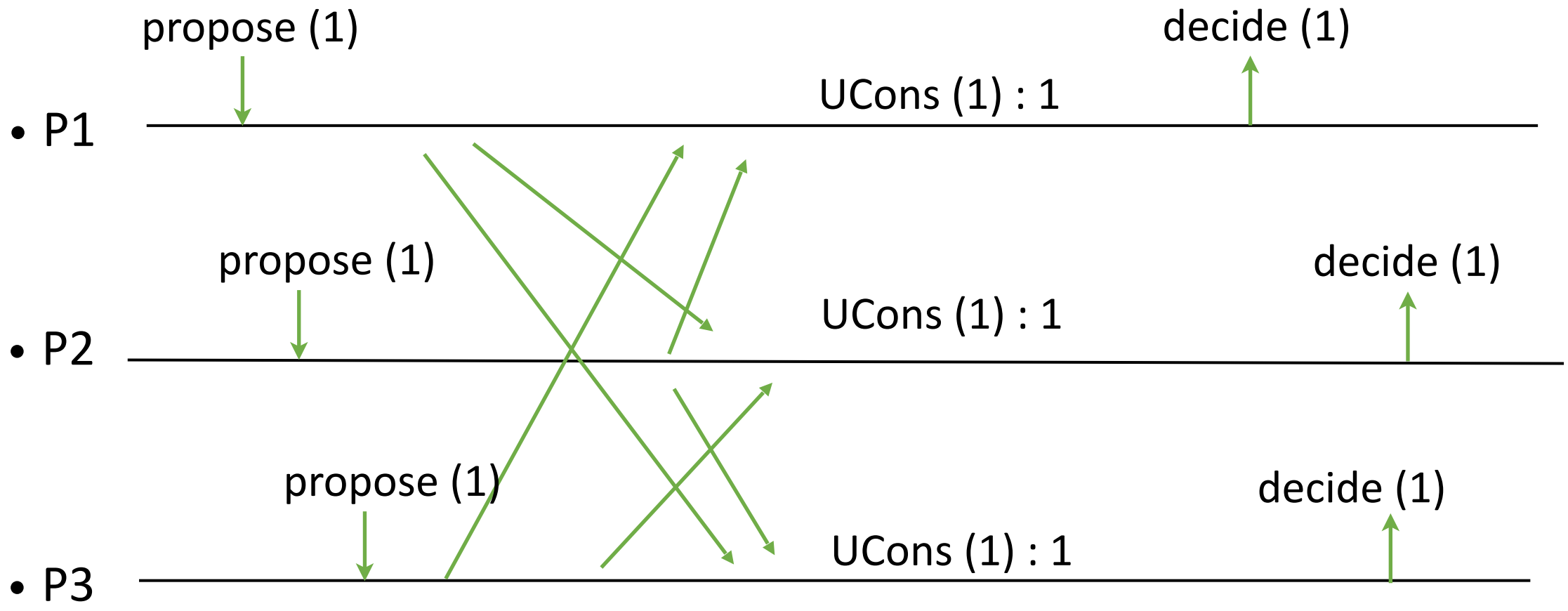
upon event delivered = Π or correct $\neq \Pi$ do

    trigger <uc, propose(prop)>

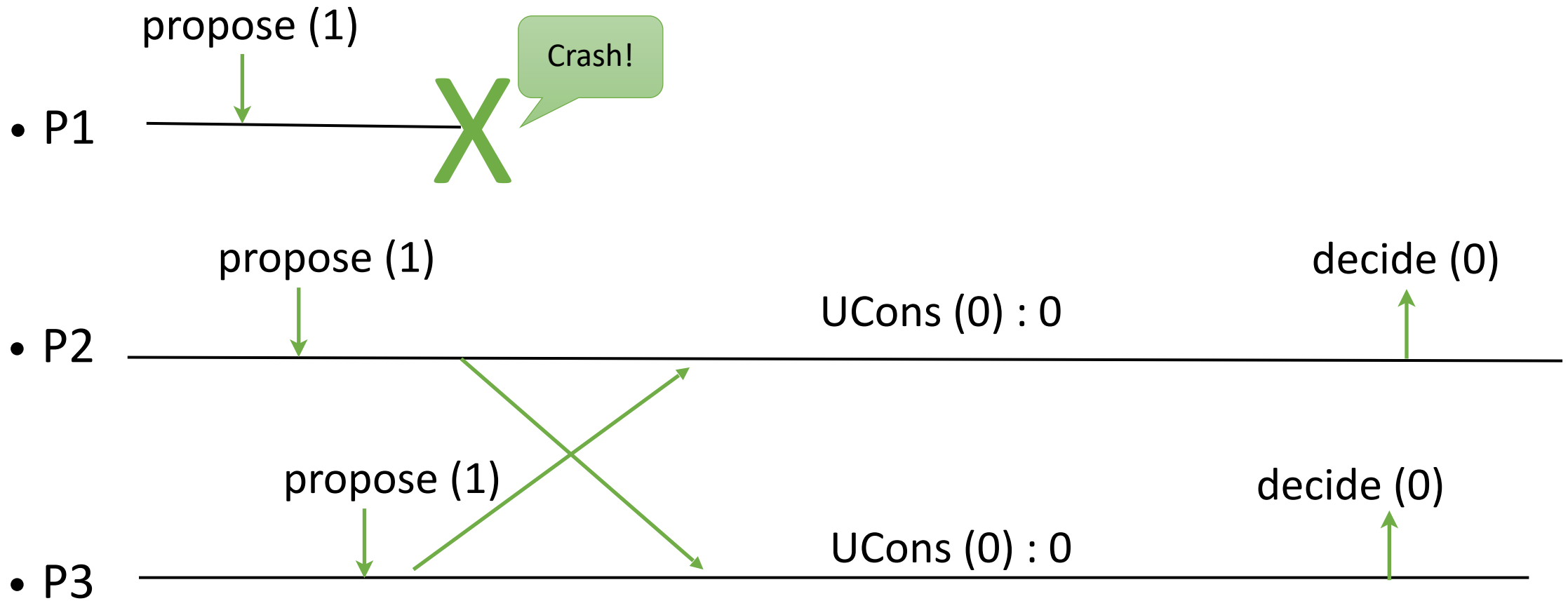
upon event <uc, decide(dec)> do

    trigger <decide (dec)>

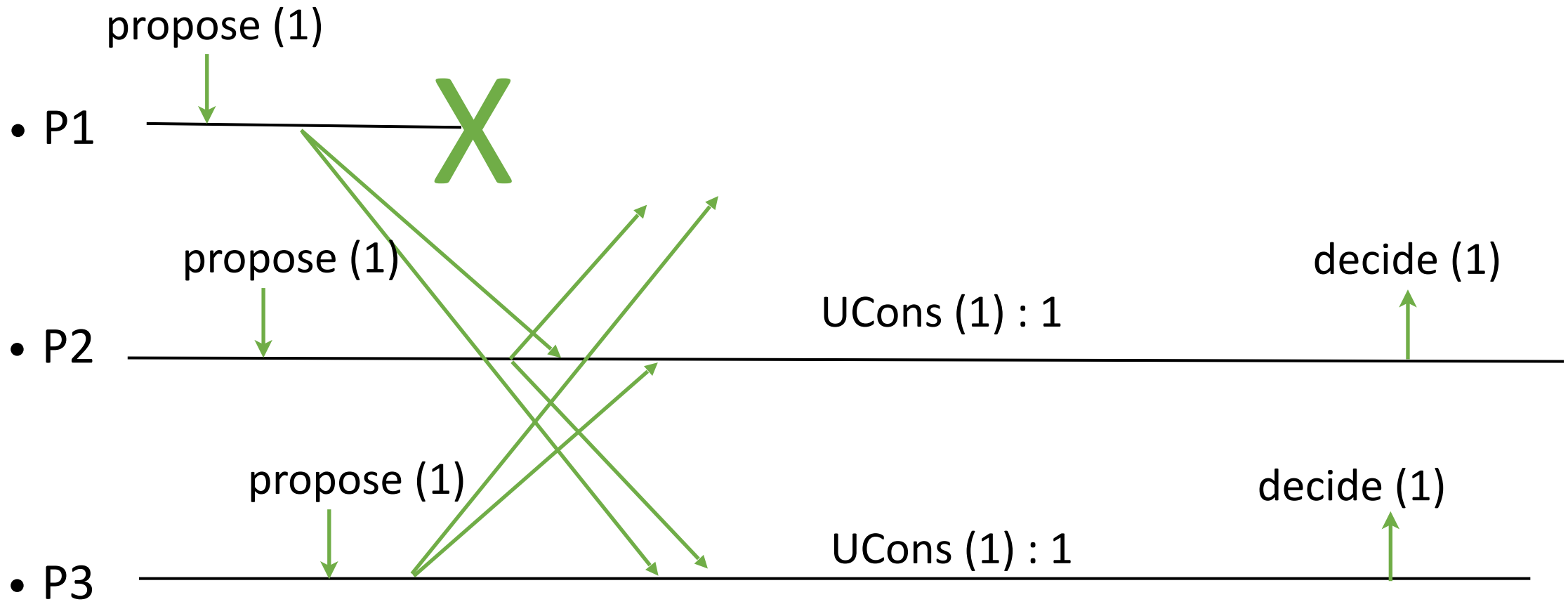
NBAC with UCons



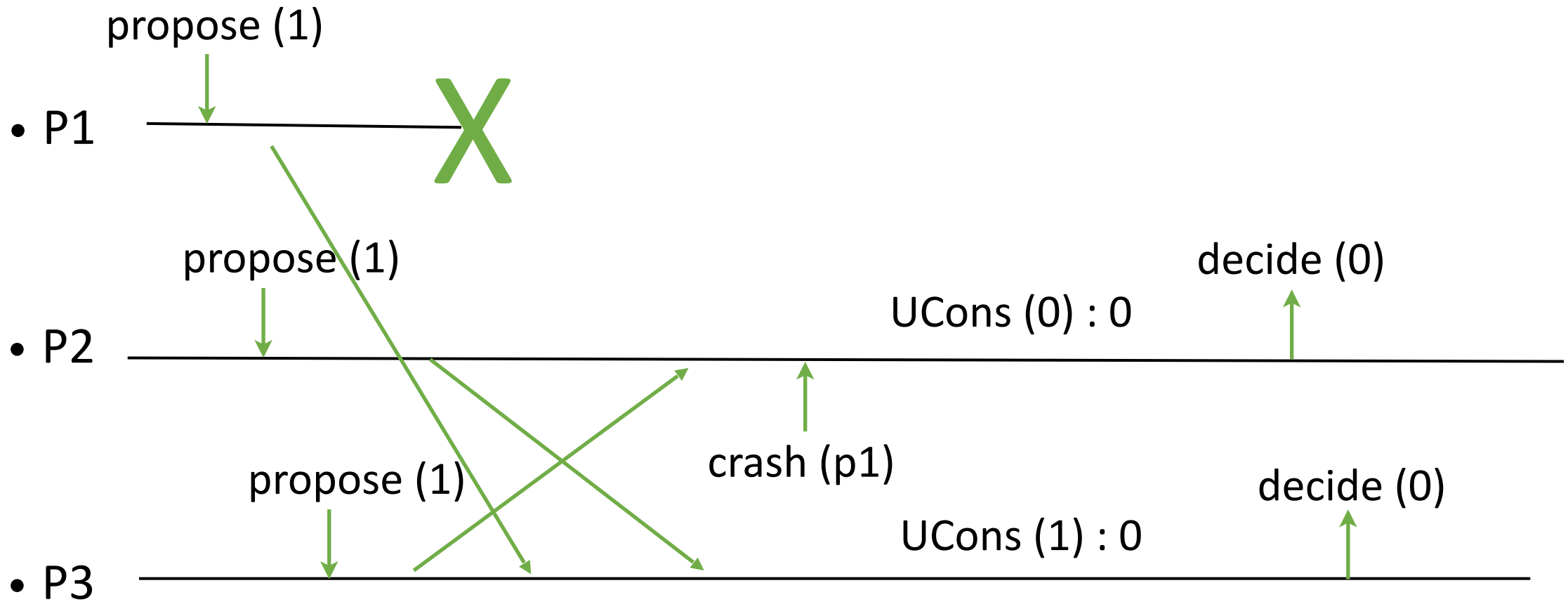
NBAC with UCons



NBAC with UCons



NBAC with UCons

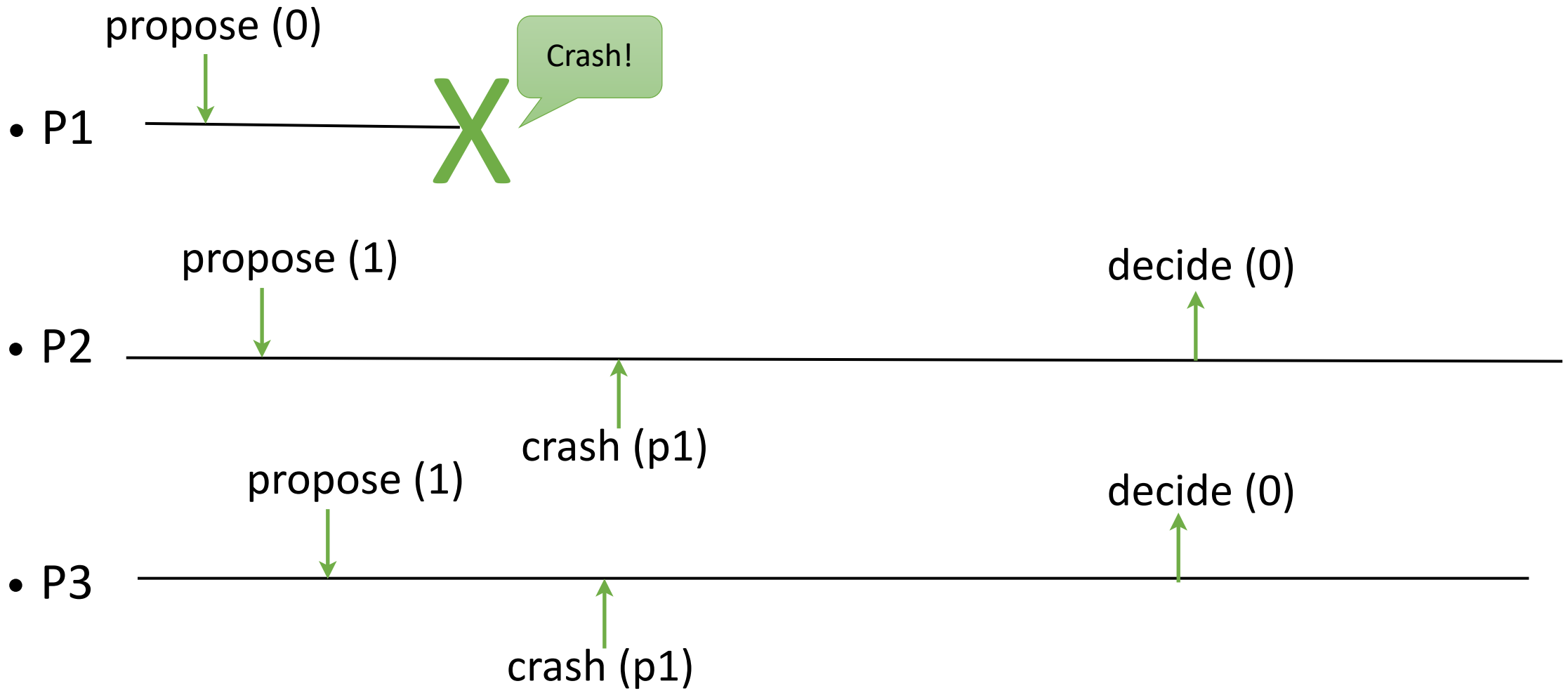


Non-Blocking Atomic Commit

- Do we need the perfect failure detector P?

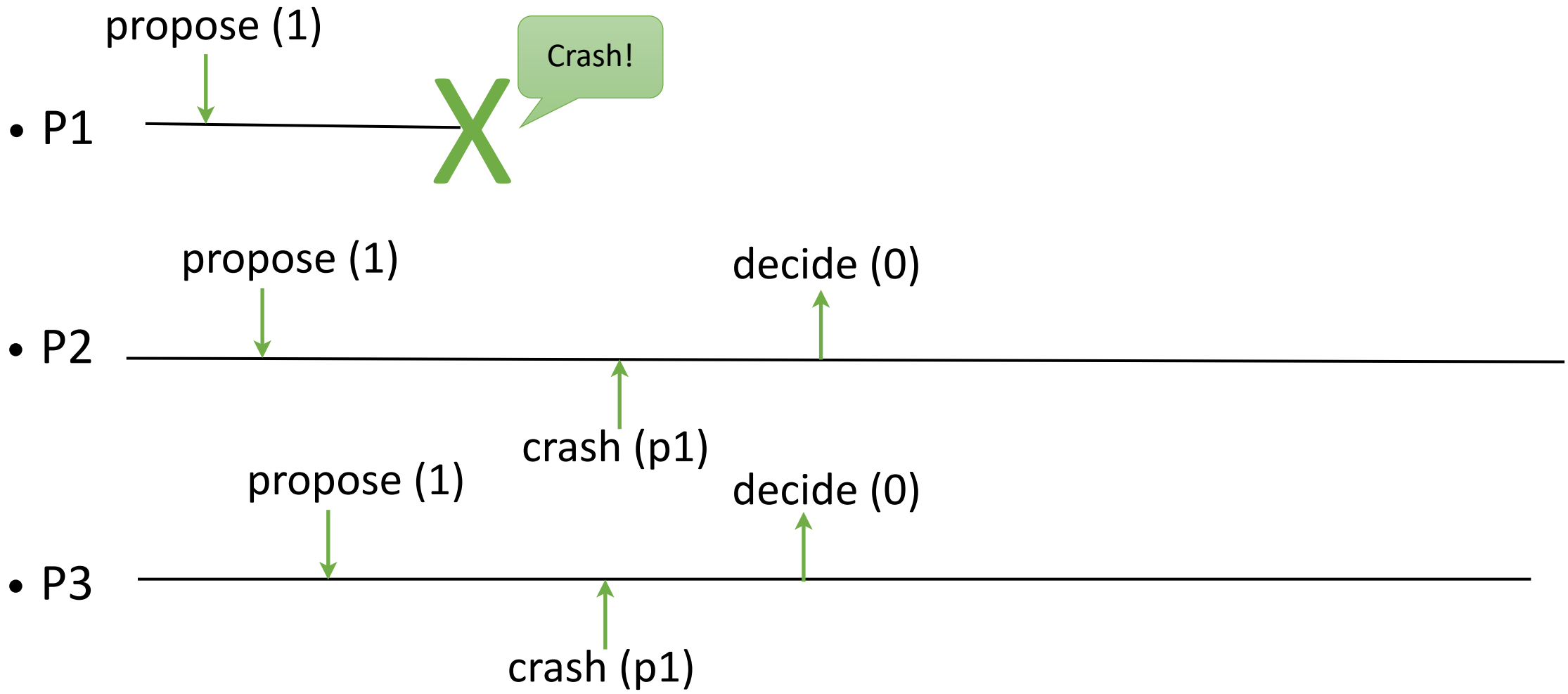
- We show that <>P is not enough.
- Let’s assume there is an atomic commit protocol that uses only <>P.
- We arrive at a contradiction by an indistinguishability argument.

Run 1



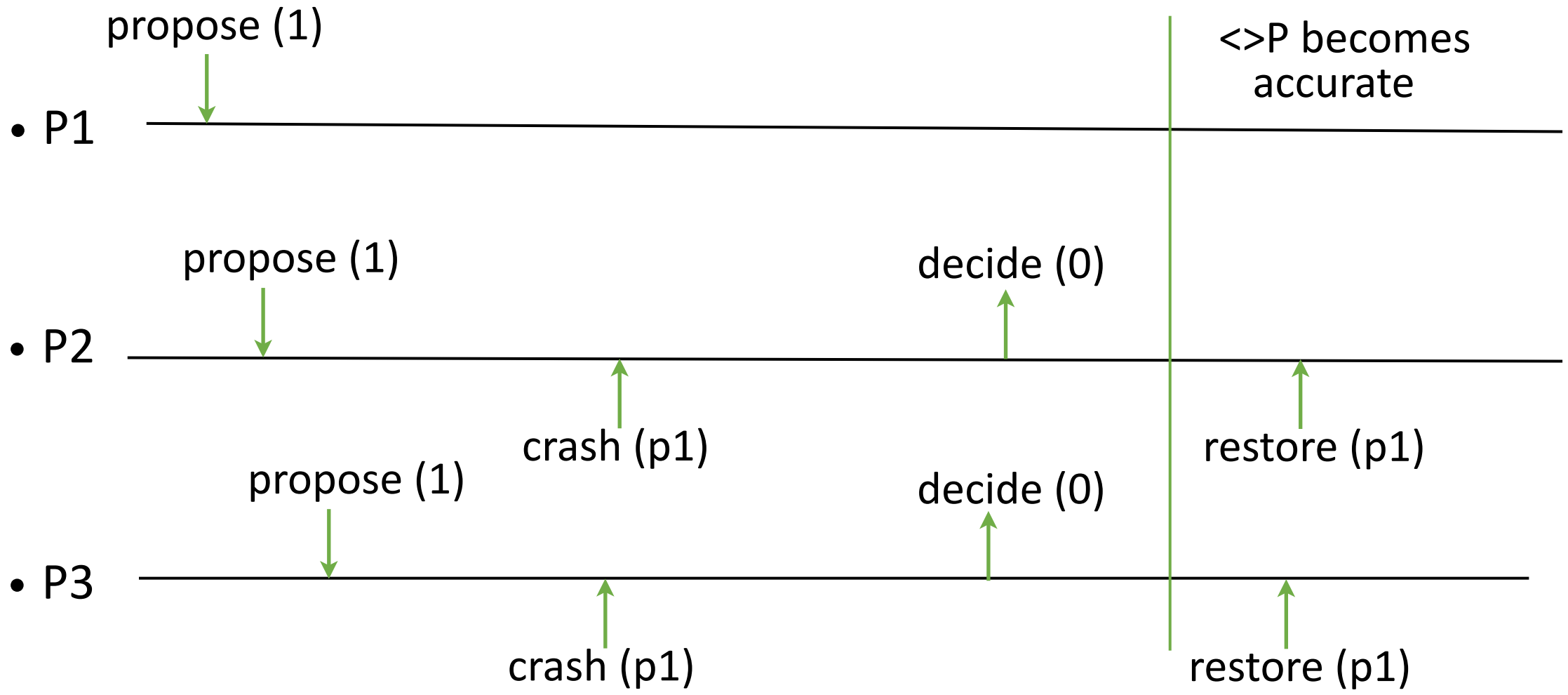
The process p1 is proposing 0. Therefore, according to the commit-validity and termination properties, other processes eventually decide 0.

Run 2



The process p1 is now proposing 1. However, p1 did not send any message in the previous and this execution. Thus, this execution should have the same decision as the previous one.

Run 3



Now, the process p1 does not crash anymore. Only its messages are slow. Thus, it is still suspected. So the algorithm still decides 0 but now that violates abort-validity.

Original slides adopted from R. Guerraoui