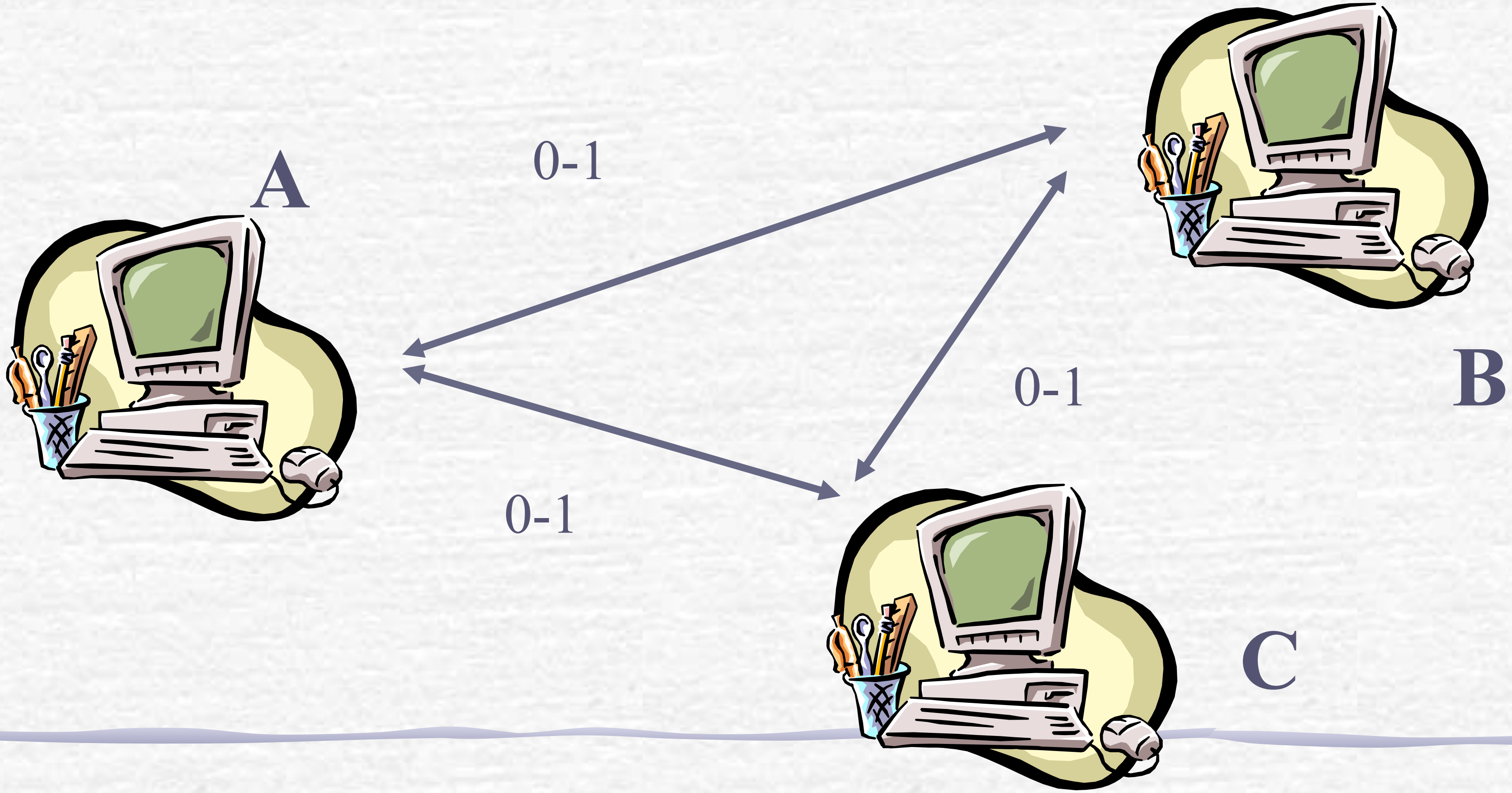
Distributed Systems
Non-Blocking Atomic Commit

Prof R. Guerraoui
Distributed Programming Laboratory

Non-Blocking Atomic Commit: An Agreement Problem



Transactions (Gray)

- A transaction is an atomic program describing a sequence of accesses to shared and distributed information

- A transaction can be terminated either by *committing* or *aborting*

Transactions

- beginTransaction
  - Pierre.credit(1.000.000)
  - Paul.debit(1.000.000)
- outcome := commitTransaction
- if (outcome = abort) then …

ACID properties

Atomicity: a transaction either performs entirely or none at all

Consistency: a transaction transforms a consistent state into another consistent state

Isolation: a transaction appears to be executed in isolation

Durability: the effects of a transaction that commits are permanent

The Consistency Contract

(system)

Atomicity

Isolation

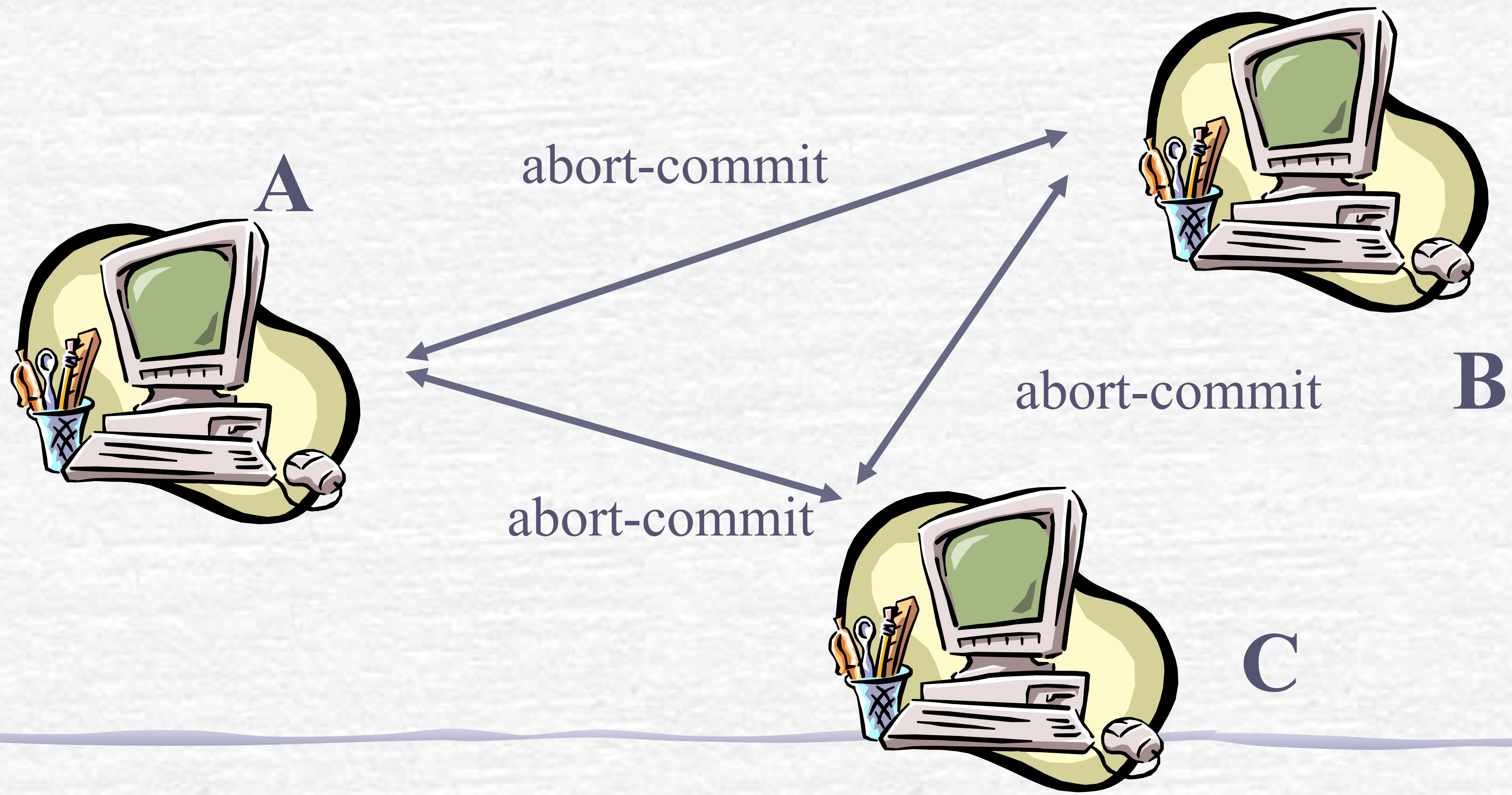
Durability

(programmer)

Consistency (local)

Consistency (global)

Distributed Transaction



Non-Blocking Atomic Commit

- 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
- The decision reflects the contract with the user
- Unlike consensus, the processes here seek to decide 1 but every process has a veto right

Non-Blocking Atomic Commit

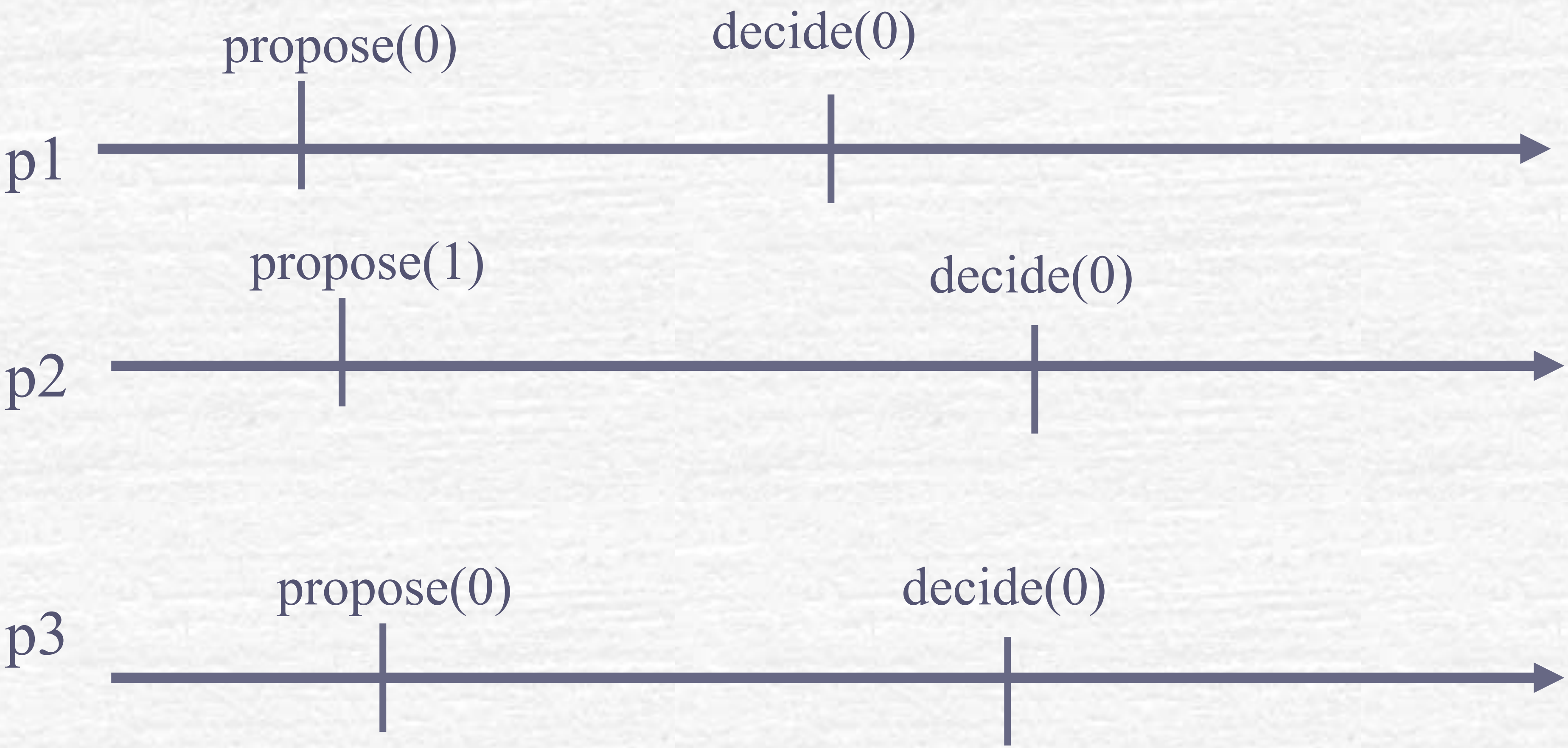
NBAC1. Agreement: No two processes decide differently

NBAC2. Termination: Every correct process eventually decides

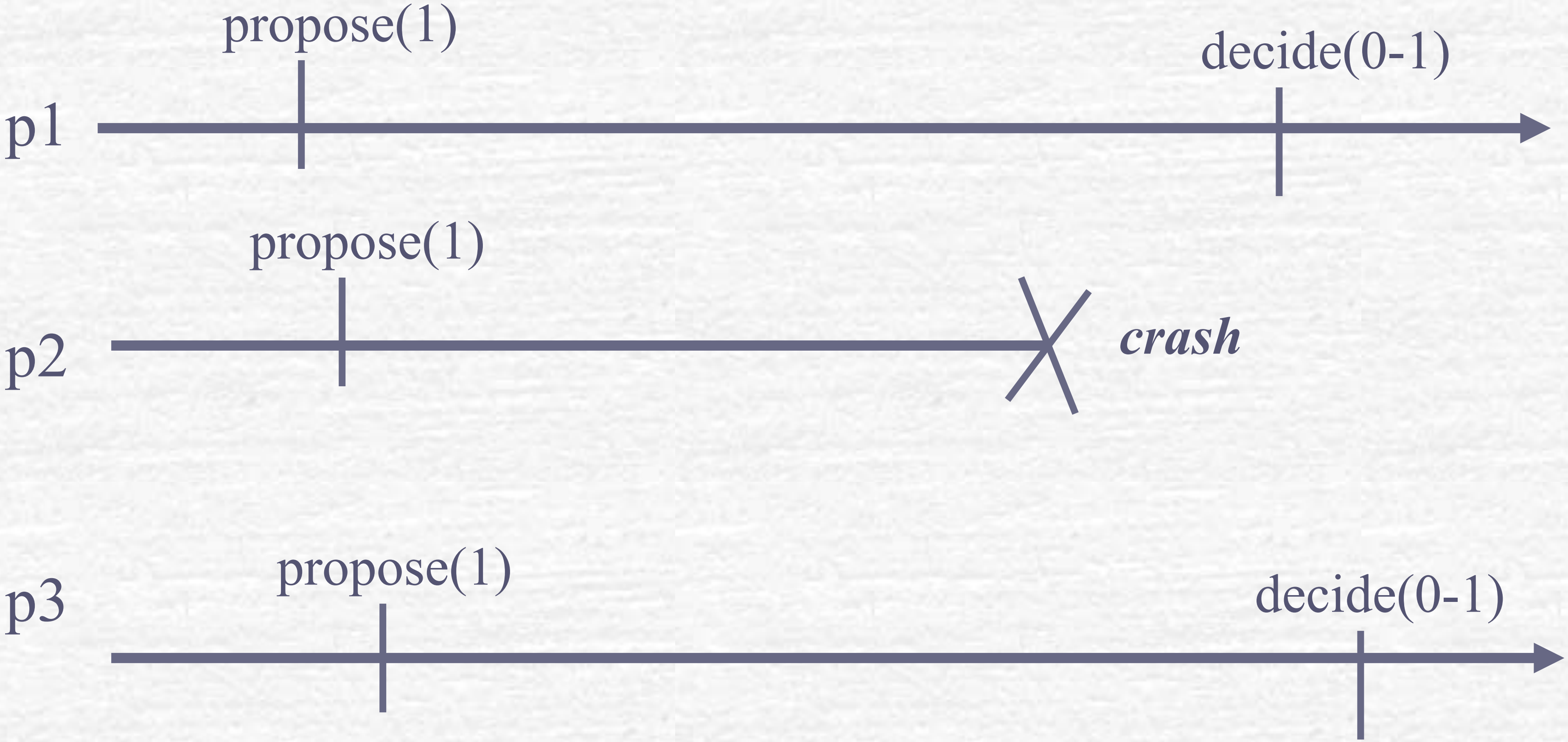
NBAC3. Commit-Validity: 1 can only be decided if all processes propose 1

NBAC4. Abort-Validity: 0 can only be decided if some process crashes or votes 0

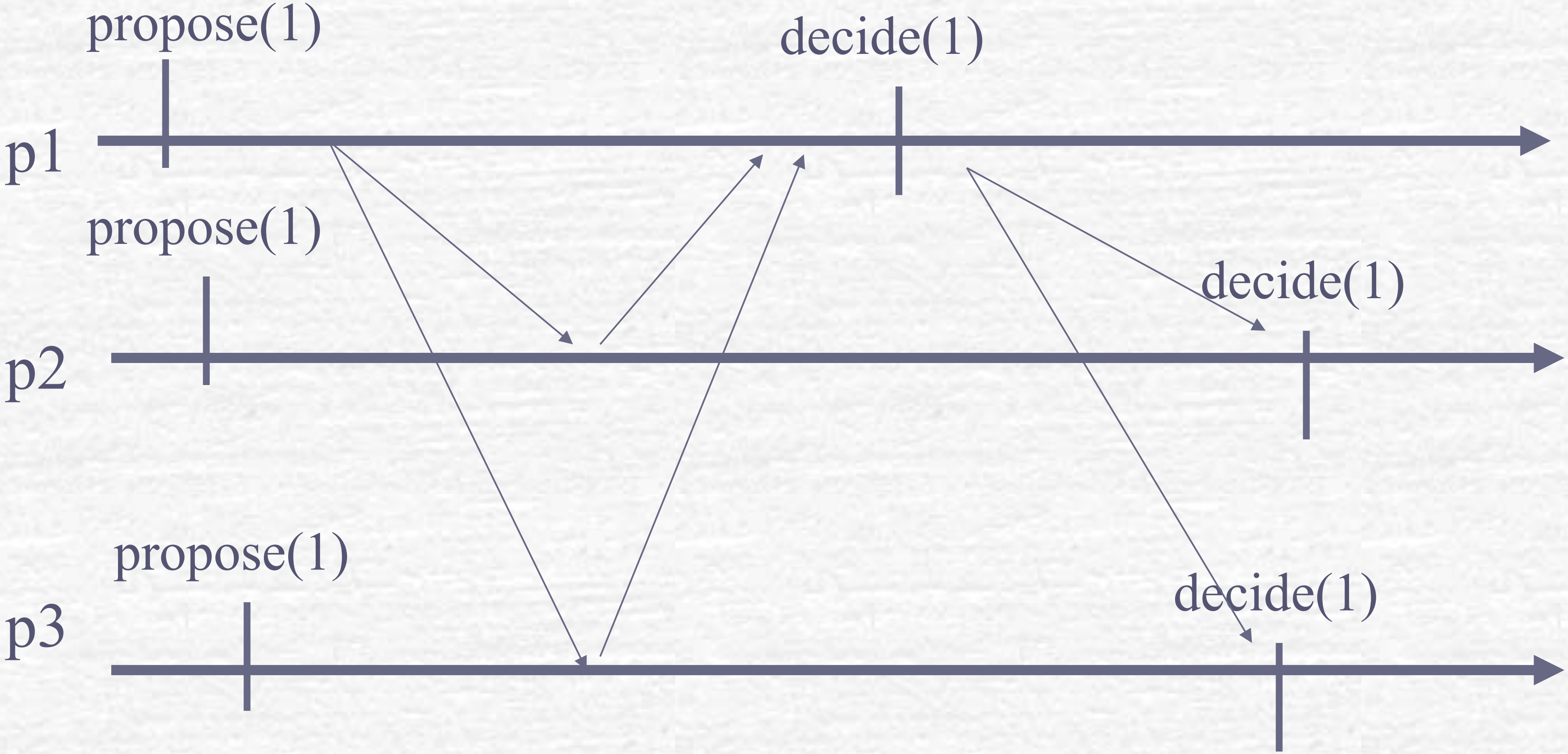
Non-Blocking Atomic Commit



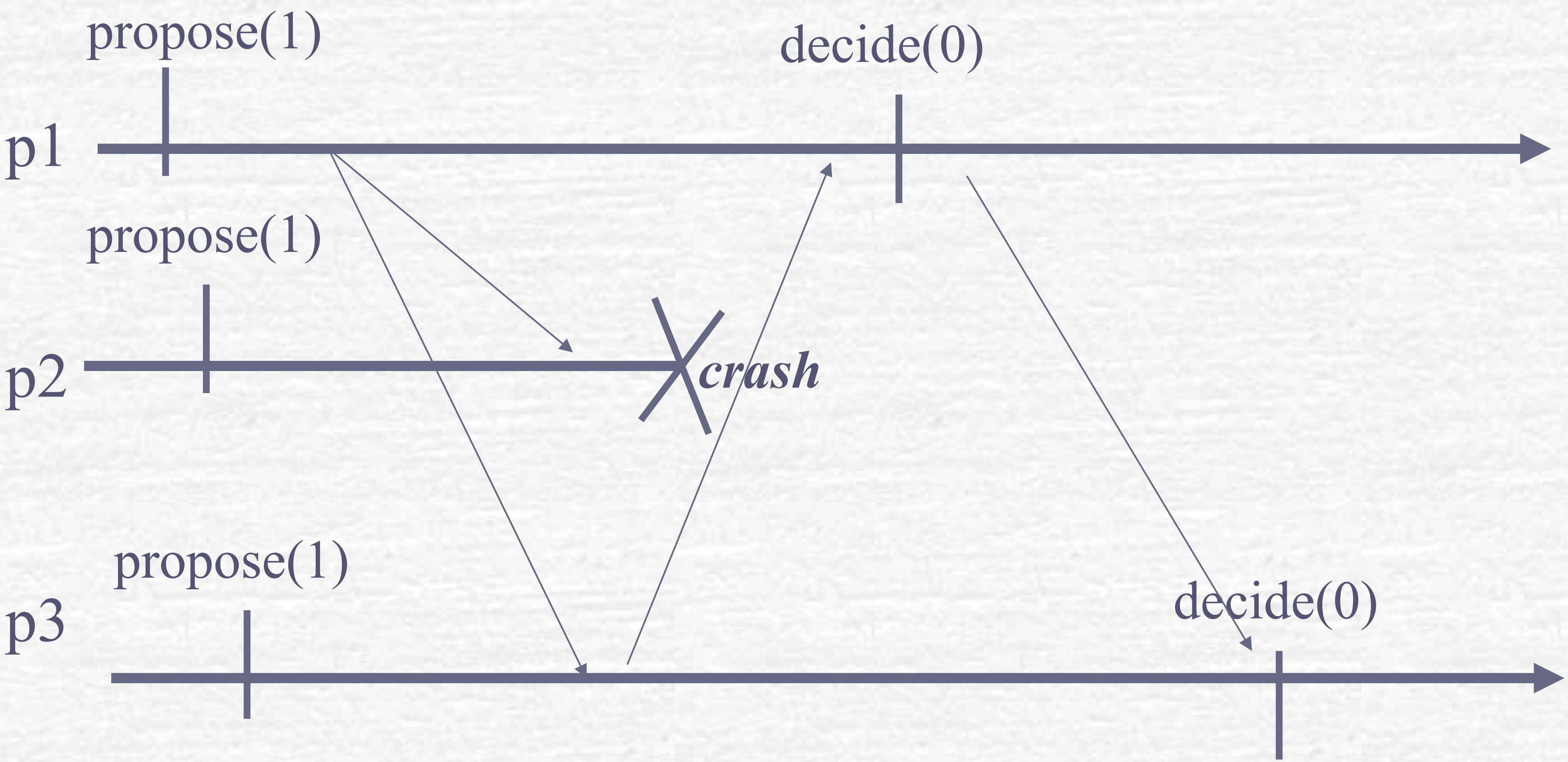
Non-Blocking Atomic Commit



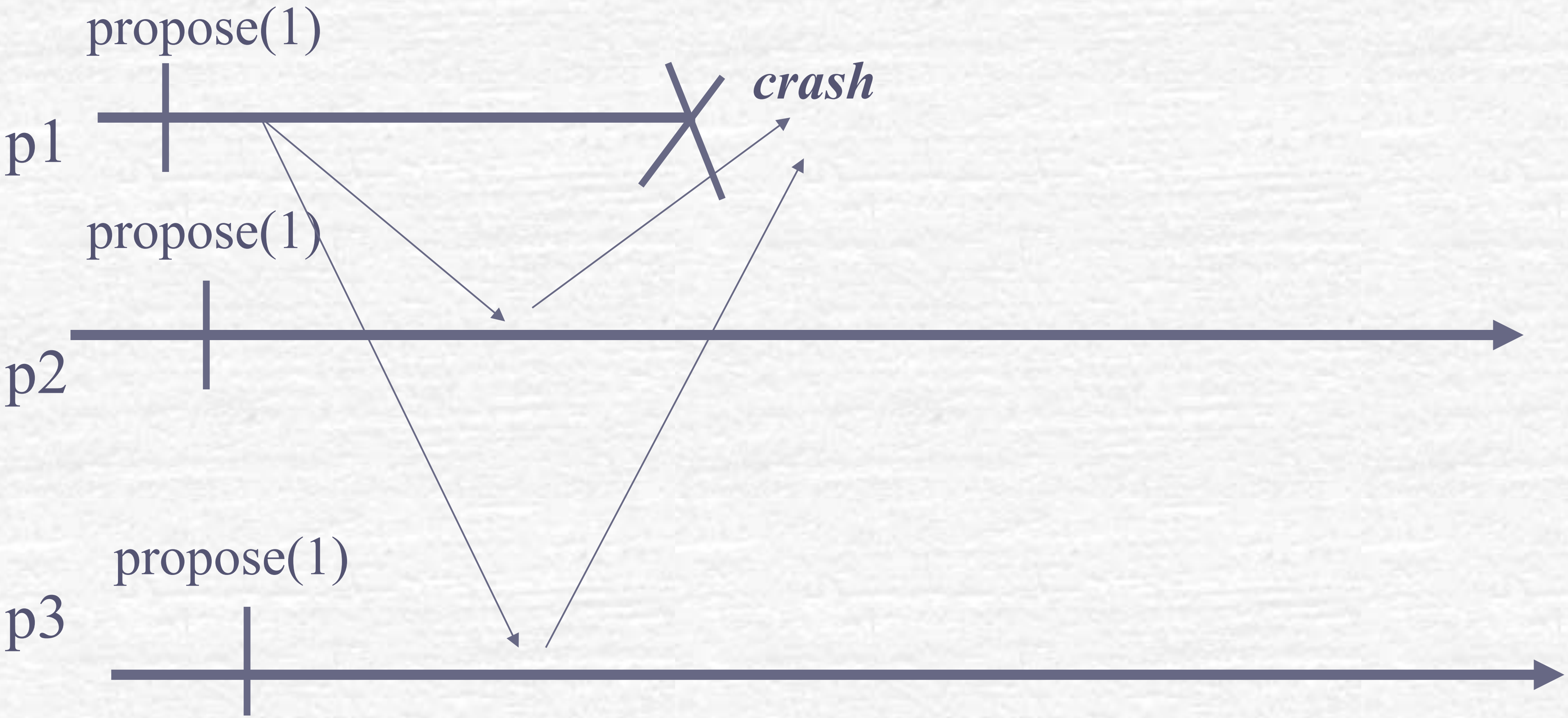
2-Phase Commit



2-Phase Commit



2-Phase Commit



Non-Blocking Atomic Commit

- ***Events***
  - Request: <Propose, v>
  - Indication: <Decide, v'>
- ***Properties:***
  - ***NBAC1, NBAC2, NBAC3, NBAC4***

Algorithm (nbac)

- **Implements:** nonBlockingAtomicCommit (nbac).
- **Uses:**
  - BestEffortBroadcast (beb).
  - PerfectFailureDetector (P).
  - UniformConsensus (uniCons).
- **upon event** < Init > **do**
  - prop := 1;
  - delivered := $\emptyset$; correct := Π;

Algorithm (nbac – cont'd)

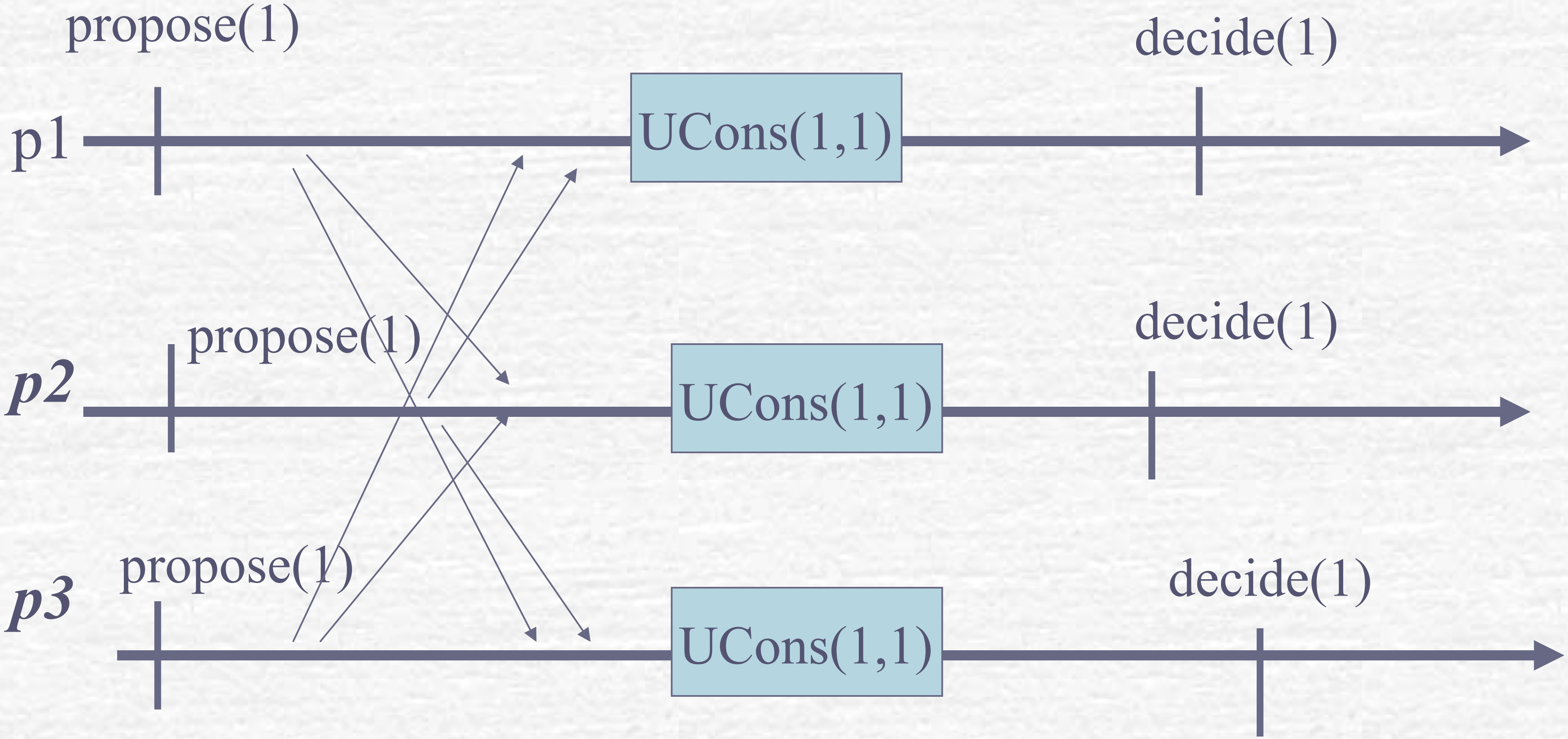
- **upon event** < crash, pi > **do**
  - correct := correct \ {pi}
- **upon event** < Propose, v > **do**
  - **trigger** < bebBroadcast, v>;
- **upon event** <bebDeliver, pi, v> **do**
  - delivered := delivered U {pi};
  - prop := prop * v;

Algorithm (nbac – cont'd)

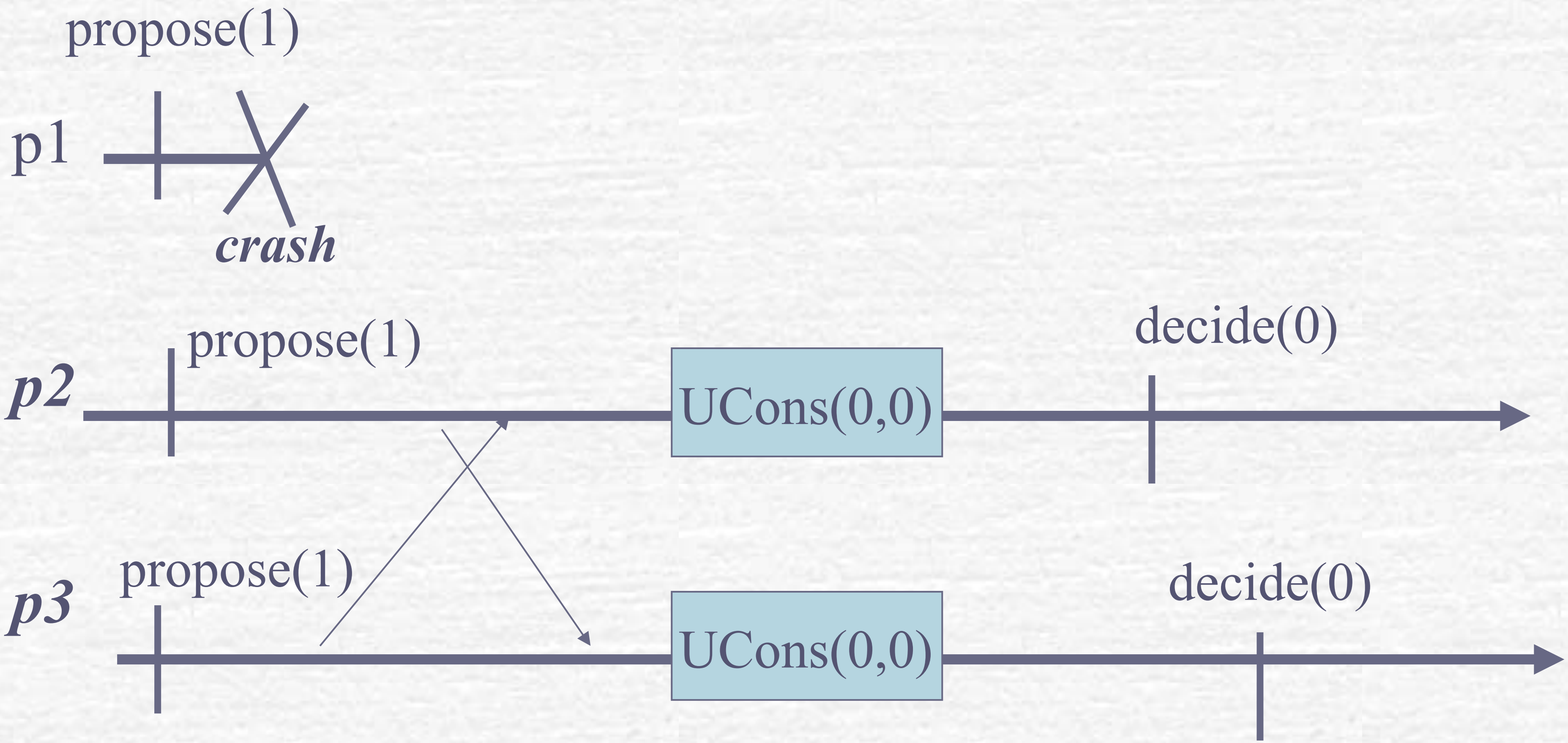
- **upon event** correct \ delivered = empty **do**
  - **if** correct $\neq \Pi$
    - prop := 0;
  - **trigger** < uncPropose, prop>;

- **upon event** < uncDecide, decision> **do**
  - **trigger** < Decide, decision>;

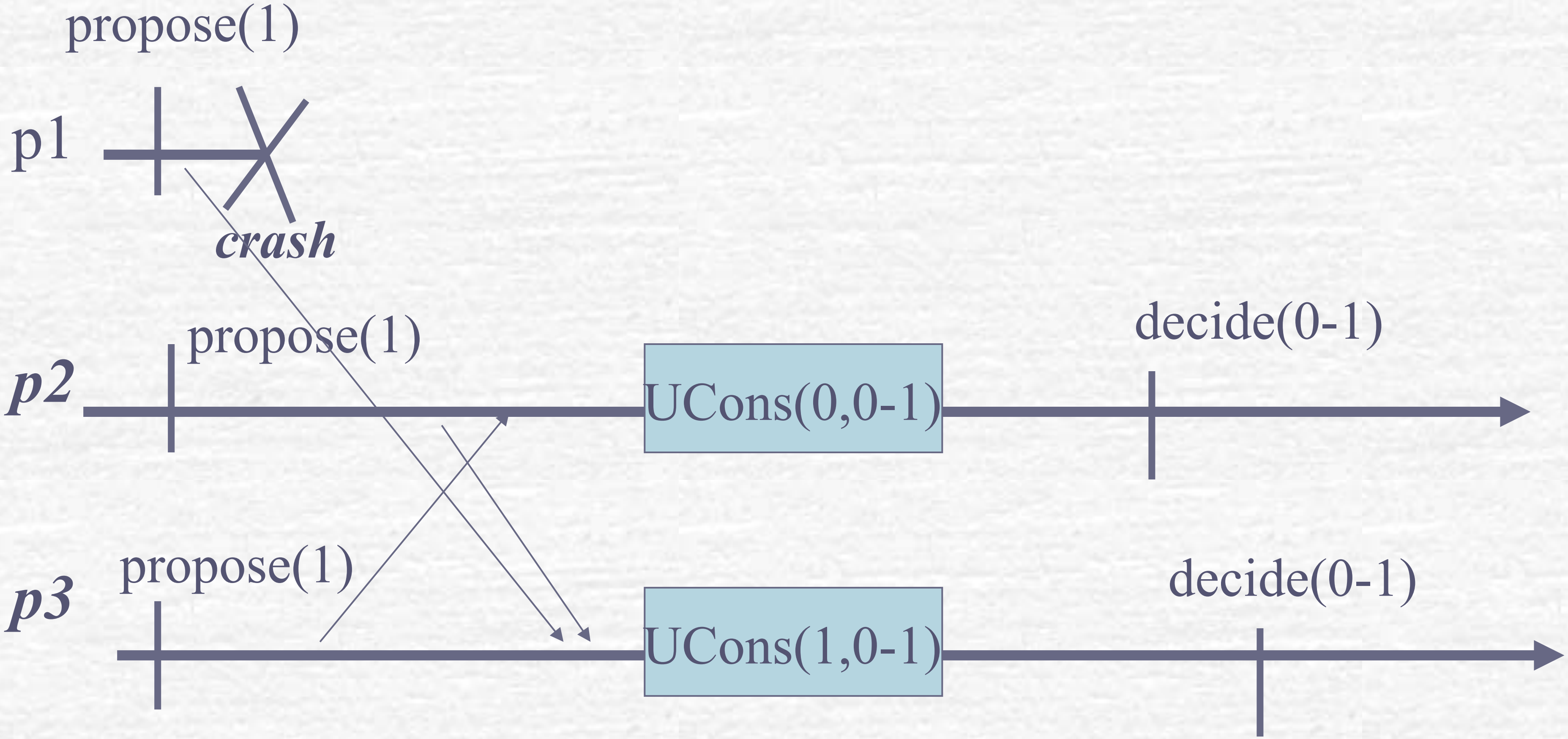
nbac with ucons



nbac with ucons



nbac with ucons



Non-Blocking Atomic Commit

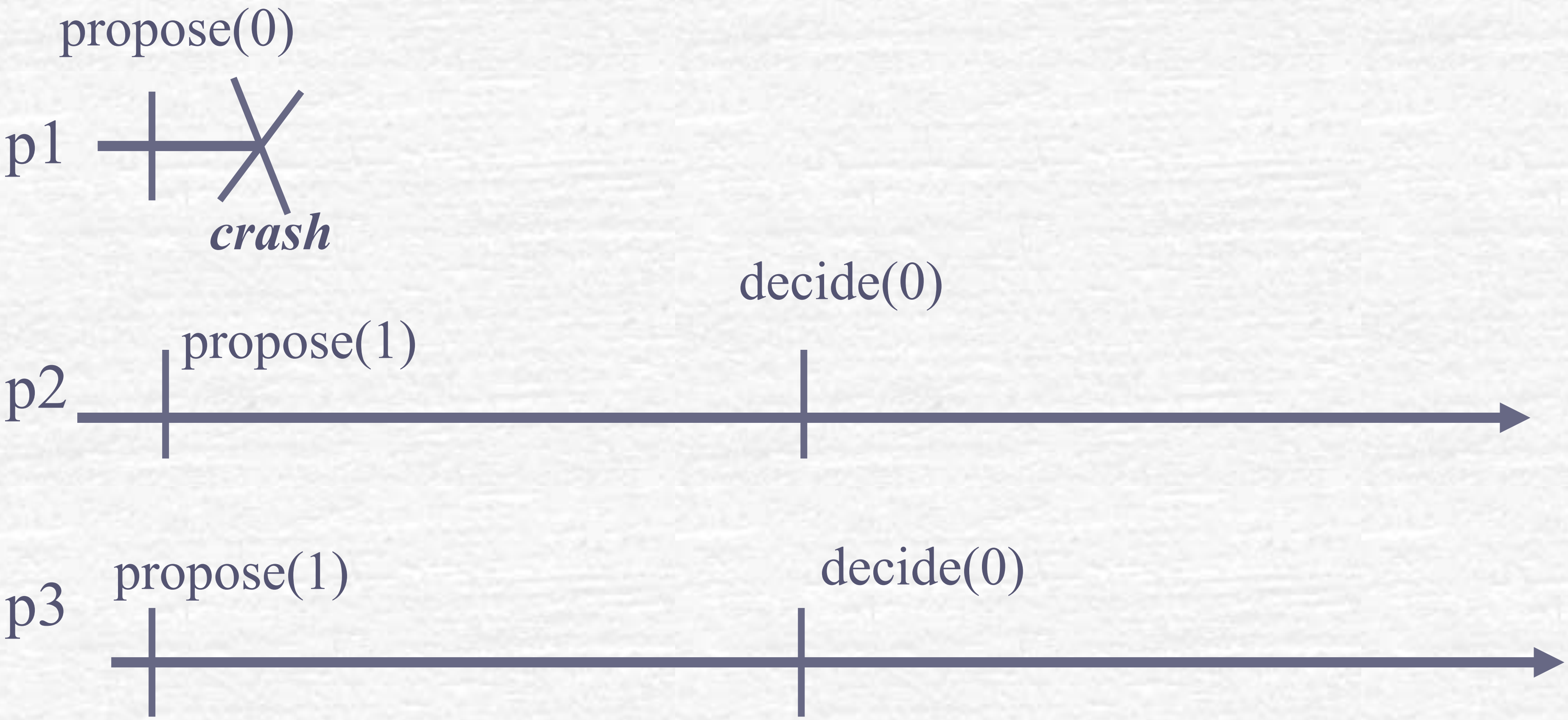
- Do we need perfect failure detector P?
  - 1. <>P is not enough
  - 2. P is needed if one process can crash

Non-Blocking Atomic Commit

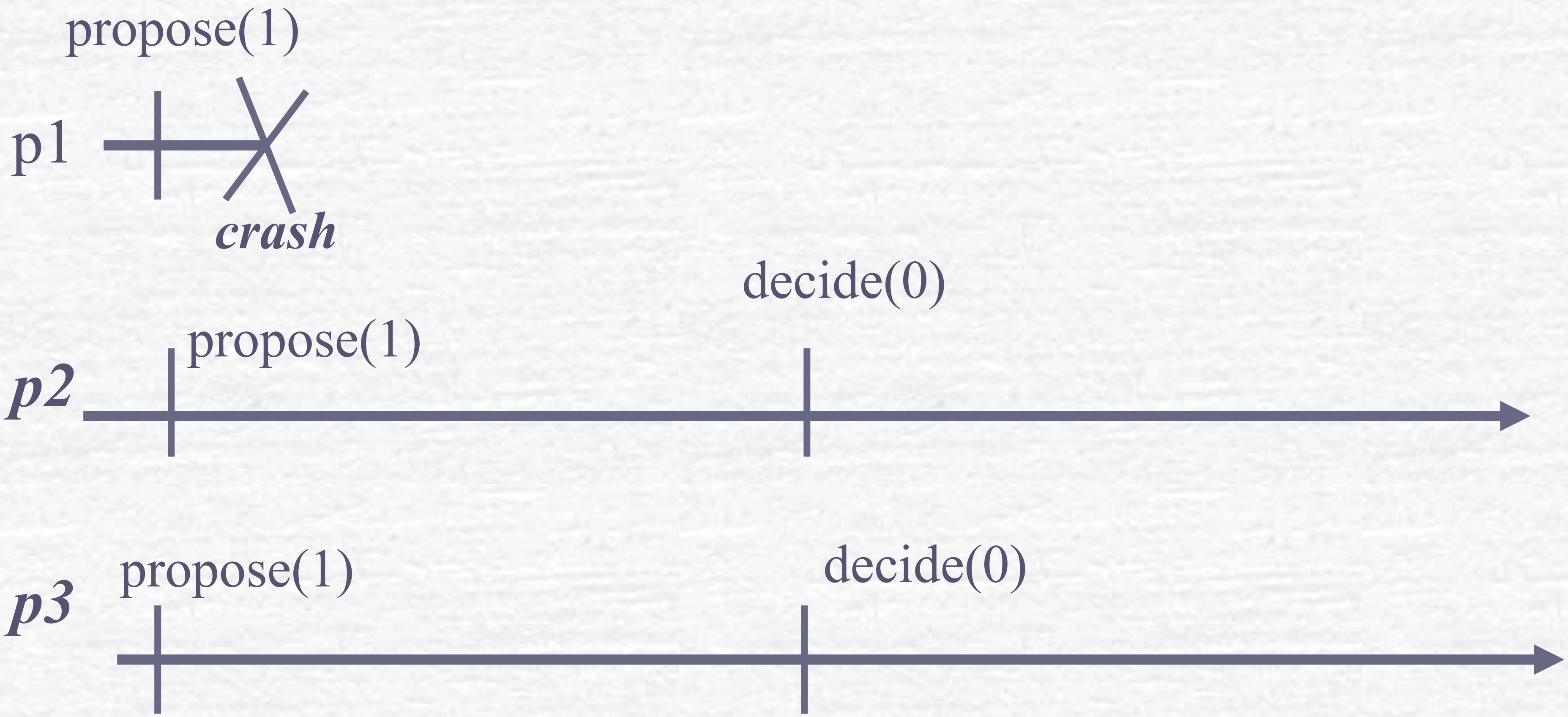
- Do we need perfect failure detector P?

  - ***1. <>P is not enough***
  - 2. P is needed if one process can crash

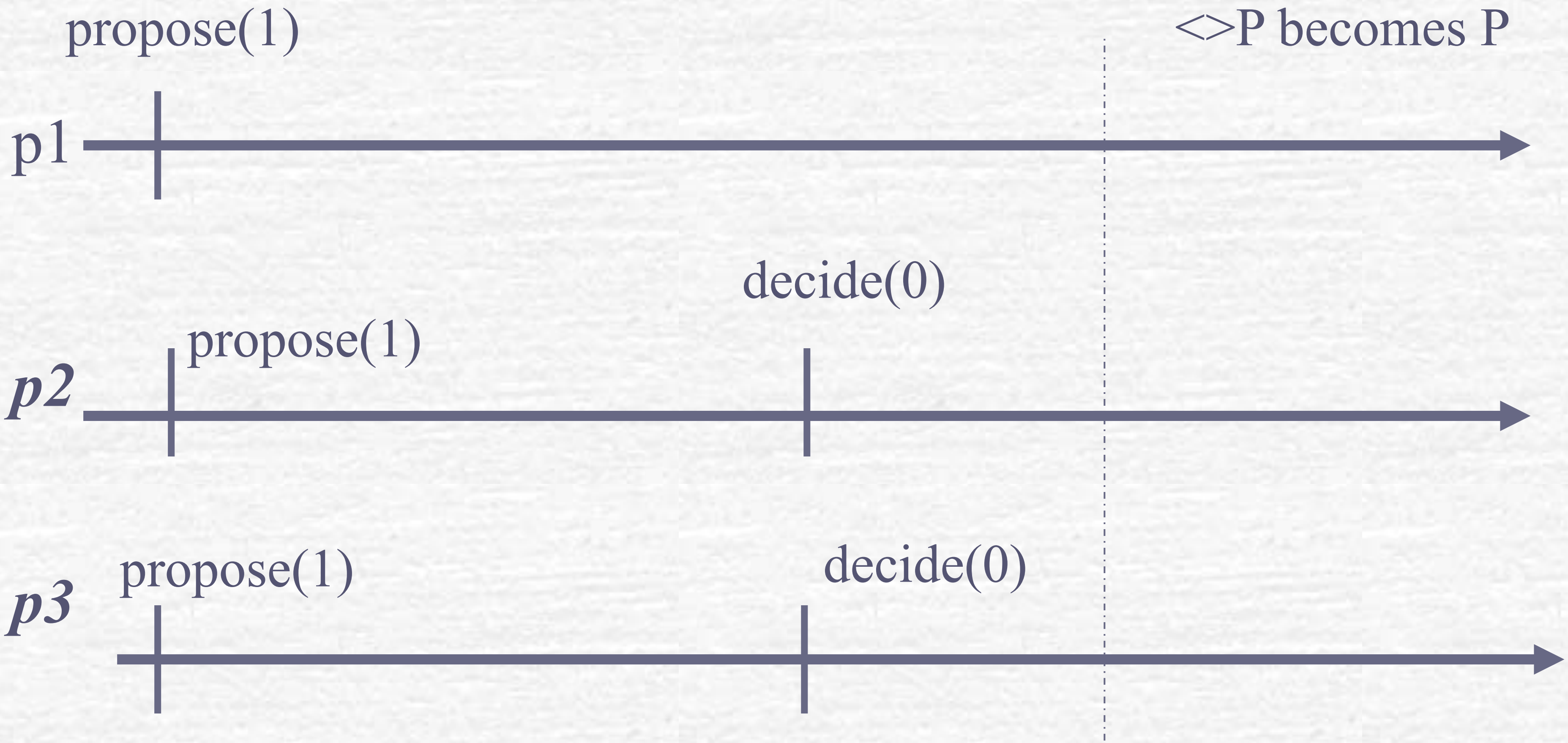
1. Run 1



1. Run 2



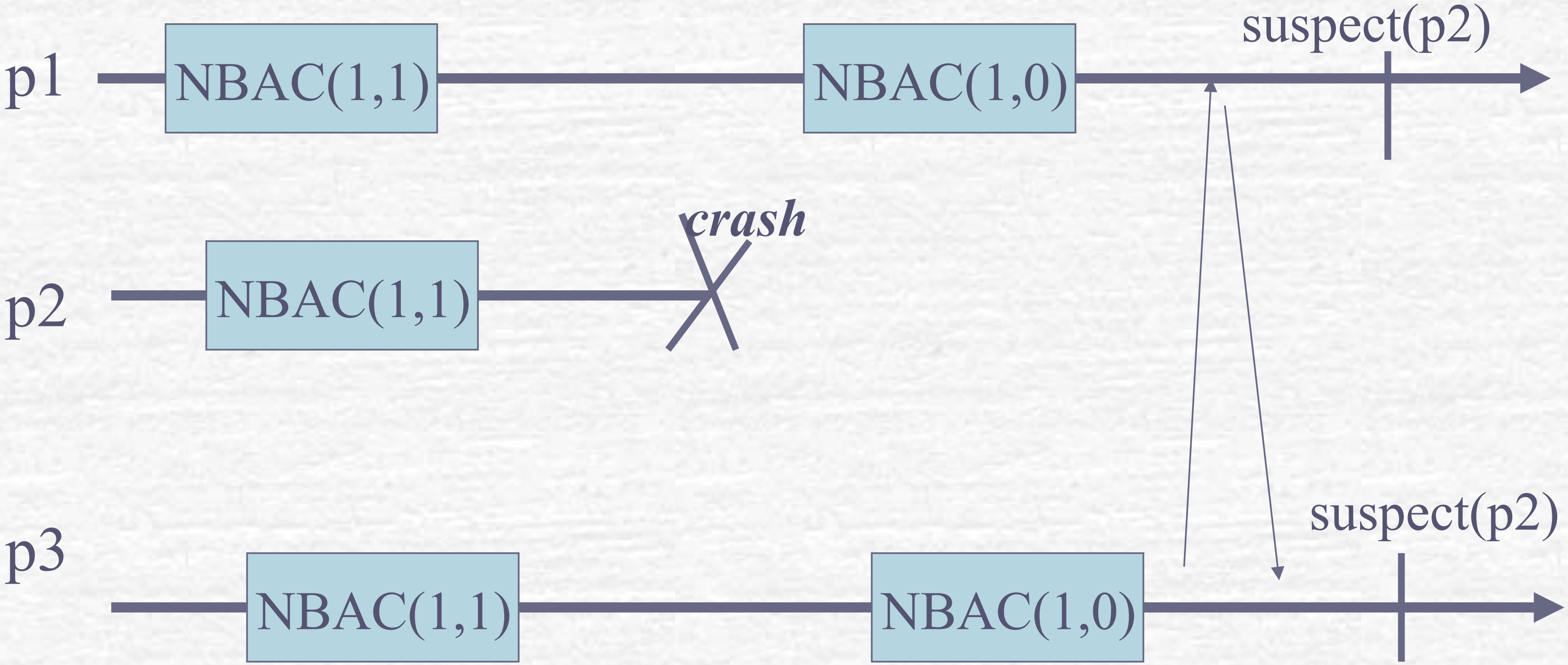
1. Run 3


propose(1)
<>P becomes P
p1
decide(0)
propose(1)
p2
propose(1)
decide(0)
p3

Non-Blocking Atomic Commit

- Do we need perfect failure detector P?
  - 1. <>P is not enough
  - ***2. P is needed if one process can crash***

2. P is needed with one crash



Non-Blocking Atomic Commit

- The weakest failure detector for NBAC

  Read DFGHTK04 for the general case