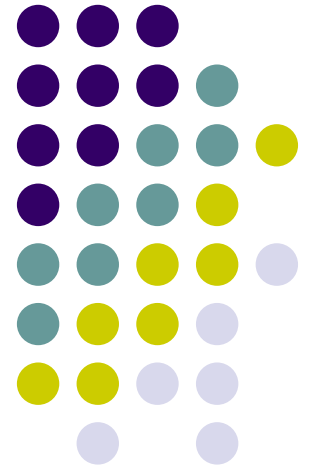


Distributed Algorithms for Building Reliable Systems

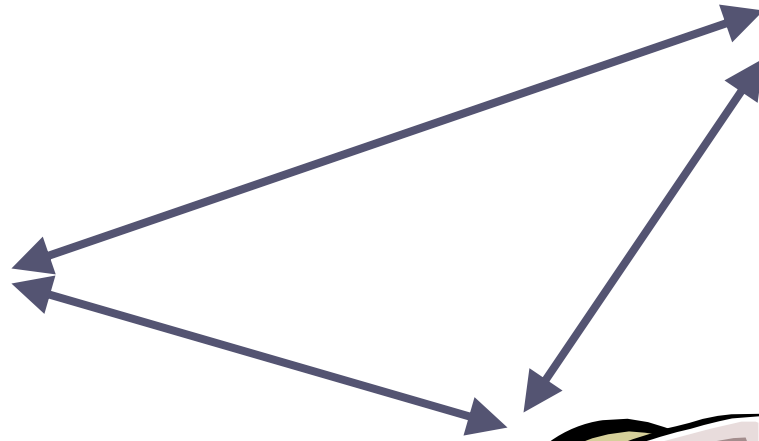
Seif Haridi
Consensus





Consensus

B



C

Consensus



- In the consensus problem, the processes propose values and have to agree on one among these values
- Solving consensus is key to solving many problems in distributed computing
 - Total order broadcast
 - Atomic commit
 - Terminating reliable broadcast

Consensus



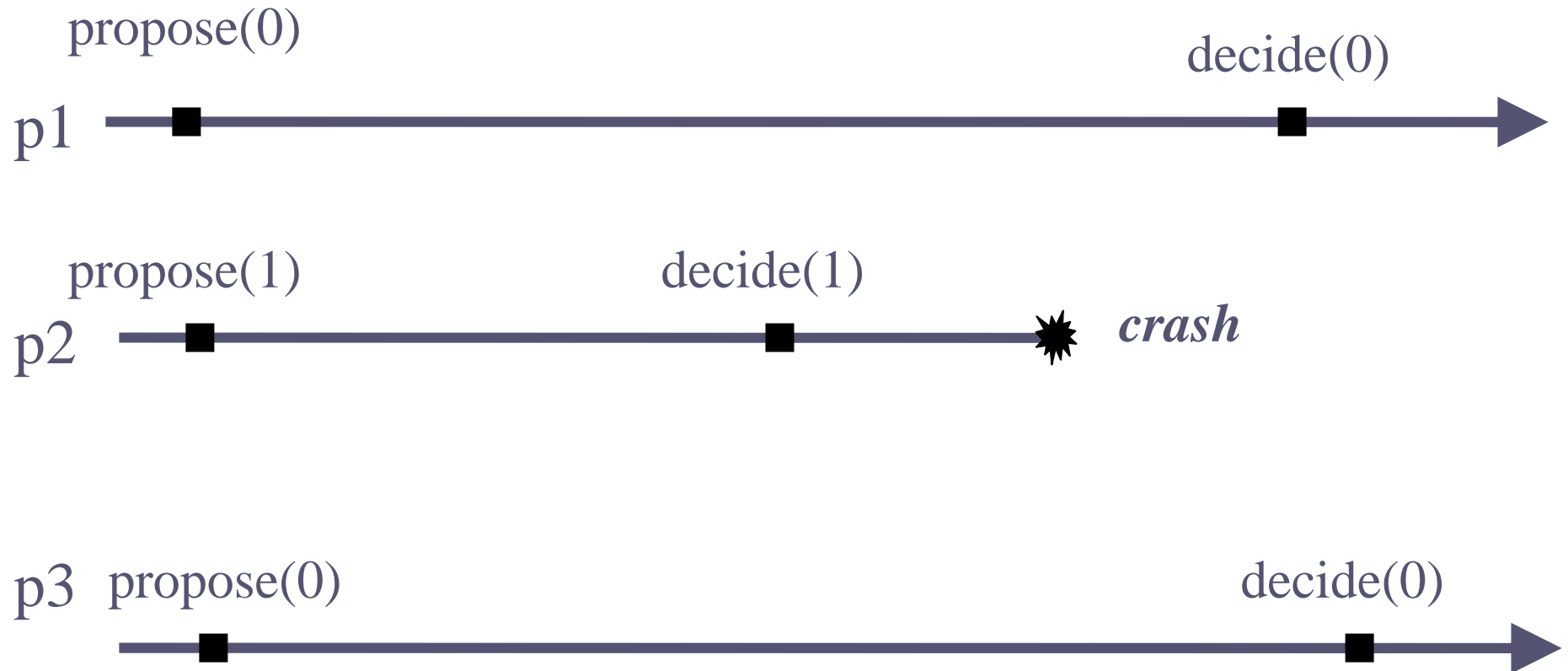
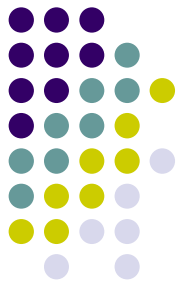
C1. Validity: Any value decided is a value proposed

C2. Agreement: No two correct processes decide differently

C3. Termination: Every correct process eventually decides

C4. Integrity: No process decides twice

Consensus



Uniform consensus



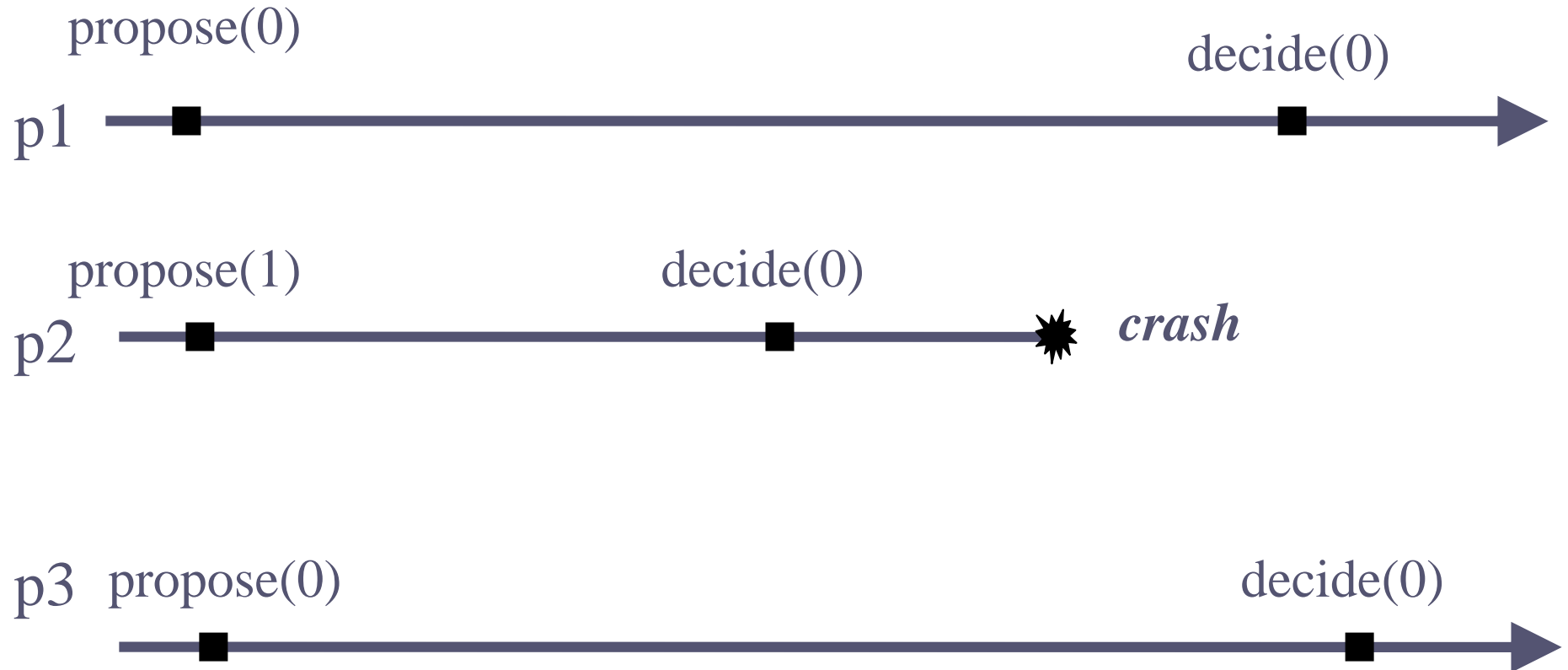
C1. Validity: Any value decided is a value proposed

C2'. Uniform Agreement: No two processes decide differently

C3. Termination: Every correct process eventually decides

C4. Integrity: No process decides twice

Uniform Consensus



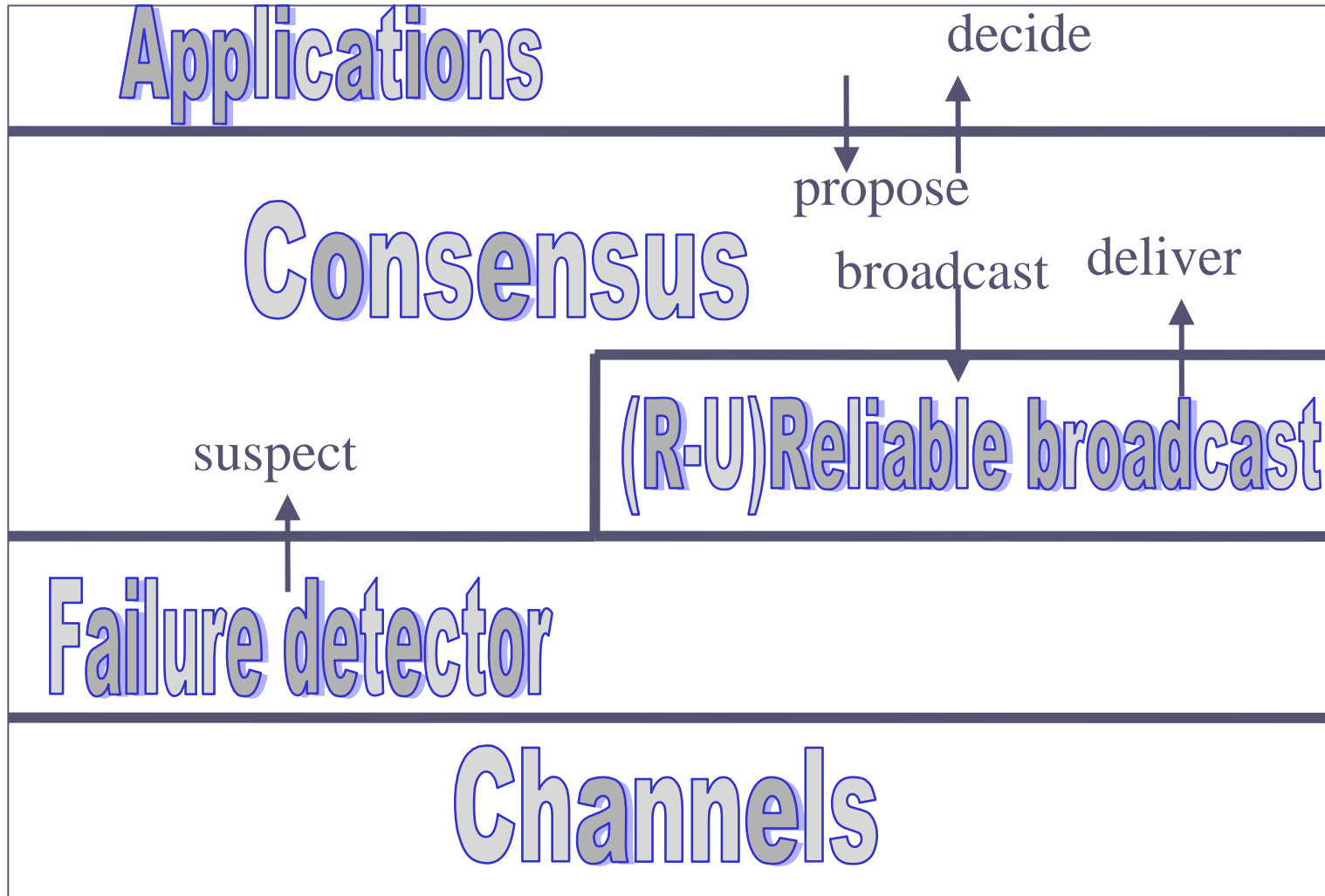
Consensus



Events

- **Request:** $\langle c\text{Propose } v \rangle$
- **Indication:** $\langle c\text{Decide } v \rangle$
- **Properties:**
 - ***C1, C2, C3, C4***

Modules of a process



Consensus algorithm I



- A fail-stop flooding consensus algorithm
- The processes exchange and update proposals in rounds and decide on the value of the non-suspected process with the smallest id [Gue95]



Consensus algorithm II

- A P-based (i.e., fail-stop) uniform consensus algorithm
- The processes exchange and update proposal in rounds, and after n rounds decide on the current proposal value [Lyn96]



Consensus algorithm III

- A $\diamond P$ -based uniform consensus algorithm assuming a correct majority
- The processes alternate in the role of a coordinator until one of them succeeds in imposing a decision [DLS88,CT91]

Consensus algorithm I



- The processes go through rounds incrementally (1 to n): in each round, the process with the id corresponding to that round is the leader of the round
- The leader of a round decides its current proposal and broadcasts it to all
- A process that is not leader in a round waits
 - (a) to deliver the proposal of the leader in that round to adopt it, or
 - (b) to detect crash of the leader
- The algorithm is not uniform since a decision is revised to the lowest rank correct process

Consensus algorithm I



☞ **Implements:** Consensus (c)

☞ **Uses:**

☞ BestEffortBroadcast (beb)

☞ PerfectFailureDetector (P)

☞ **upon event** $\langle \text{Init} \rangle$ **do**

- $\text{detected} := \emptyset$ $\text{round} := 1$;
- $\text{proposal} := \perp$; $\text{proposer} := 0$
- **for** $i = 1$ **to** N **do**
 - $\text{broadcast}[i] := \text{delivered}[i] := \text{false}$

Consensus algorithm I



- **upon event** $\langle \text{crash } p_i \rangle$ **do**
 - **detected** $:= \text{detected} \cup \{ \text{rank}(p_i) \}$

- **upon event** $\langle \text{cPropose } v \rangle$ **do**
 - **if** $\text{proposal} = \perp$ **then**
 - $\text{proposal} := v$

Consensus algorithm I



- ☞ **upon** round = rank(self) **and**
 - ☞ broadcast[round] = false **and**
 - ☞ proposal $\neq \perp$ **do**
 - ☞ **trigger** \langle cDecide proposal \rangle
 - ☞ **trigger** \langle bebBroadcast (DECIDED, round, proposal) \rangle
 - ☞ broadcast[round] := true

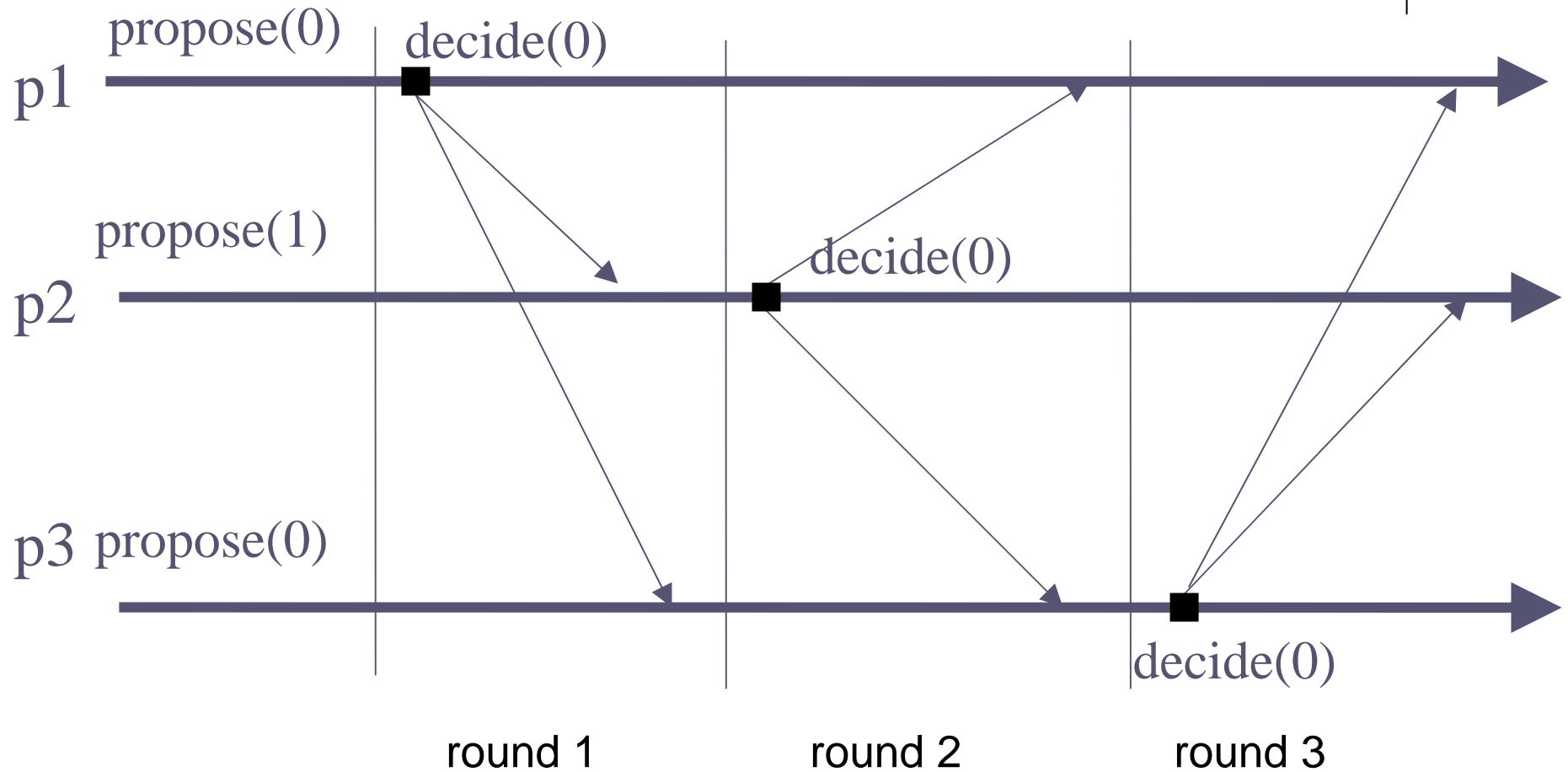
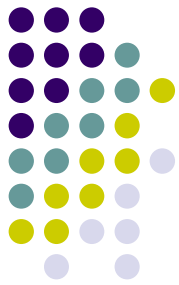
Consensus algorithm I



- ☞ **upon event** $\langle \text{bebDeliver } p_i, (\text{DECIDED}, r, v) \rangle$ **do**
 - ☞ **if** $r < \text{rank}(\text{self})$ **and** $r > \text{proposer}$ **then**
 - ☞ $\text{proposal} := v; \text{proposer} := r$
 - ☞ $\text{delivered}[r] := \text{true}$
- ☞ **upon** $\text{delivered}[\text{round}]$ **or** $\text{round} \in \text{detected}$ **do**
 - ☞ $\text{round} := \text{round} + 1$

Consensus algorithm I

No failure



Consensus



C1. Validity: Any value decided is a value proposed

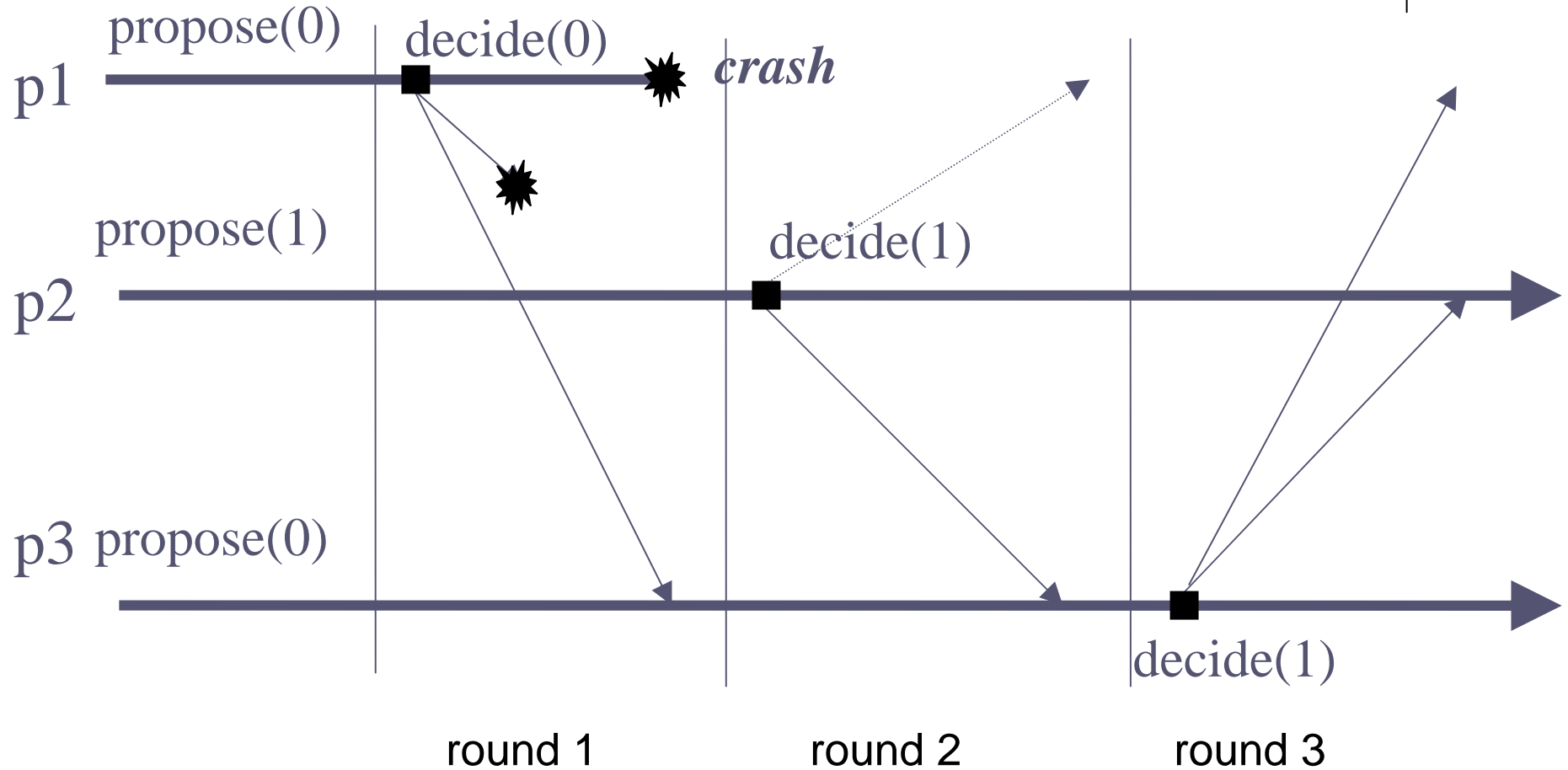
C2. Agreement: No two correct processes decide differently

C3. Termination: Every correct process eventually decides

C4. Integrity: No process decides twice

Consensus algorithm I

Failure, not uniform



Correctness argument



- Let p_i be the correct process with the smallest id in a run R
- Assume p_i decides v
 - If $i = n$, then p_n is the only correct process.
 - Otherwise, in round i , all correct processes receive v and will not decide anything different from v .

Fail-Stop Algorithm I'

Flooding Regular Consensus



- Terminates quickly when there is no failure
- Each process broadcast its set of proposal, initially containing only its own proposal
- Each process collects the set of proposals, and decides on one value deterministically, e.g. the lowest number
- Decision happens when a process knows it gathered all proposals that will ever be seen by all other correct processes
- This implies that the algorithm works in rounds

Fail-Stop Algorithm I'

Flooding Regular Consensus



- When a round terminates at p_i ?
 - When p_i receives a message tagged with the round number from every process that p_i has not detected as crashed
- When p_i knows it is safe to decide ?
 - When a round terminates with no new failure detected in this round
- Which value is selected ?
 - A value based on agreed-upon deterministic function

Flooding Consensus



- **Implements:**
 - Consensus
- **Uses:**
 - BestEffortBroadcast (beb)
 - PerfectFailureDetector (P)
- **upon event** $\langle \text{Init} \rangle$
 - $\text{correct} := \text{correct-this-round}[0] := \Pi$
 - $\text{decided} := \perp; \text{round} := 1$
 - **for** $i = 1$ **to** N **do**
 - $\text{correct-this-round}[i] := \text{proposal-set}[i] := \emptyset$

Flooding Consensus



- **upon event** $\langle \text{crash } pi \rangle$ **do**
 - $\text{correct} := \text{correct} \setminus \{ pi \}$
- **upon event** $\langle \text{cPropose } v \rangle$ **do**
 - $\text{proposal-set}[1] := \text{proposal-set}[1] \cup \{ v \}$
 - **trigger** $\langle \text{bebBroadcast} (\text{MYSET}, 1, \text{proposal-set}[1]) \rangle$
- **upon event** $\langle \text{bebDeliver } pi, (\text{MYSET}, r, \text{pset}) \rangle$ **do**
 - $\text{correct-this-round}[r] := \text{correct-this-round}[r] \cup \{ pi \}$
 - $\text{proposal-set}[r] := \text{proposal-set}[r] \cup \text{pset}$

Flooding Consensus



- **upon** $\text{correct} \subseteq \text{correct-this-round}[\text{round}]$ **and**
 - $\text{decided} = \perp$ **do**
 - **if** $\text{correct-this-round}[\text{round}] = \text{correct-this-round}[\text{round} - 1]$
 - **then**
 - $\text{decided} := \min(\text{proposal-set}[\text{round}])$
 - **trigger** $\langle \text{cDecided } \text{decided} \rangle$
 - **trigger** $\langle \text{bebBroadcast}(\text{DECIDED}, \text{decided}) \rangle$
 - **else**
 - $\text{round} := \text{round} + 1$
 - **trigger** $\langle \text{bebBroadcast}(\text{MYSET}, \text{round}, \text{proposal-set}[\text{round} - 1]) \rangle$

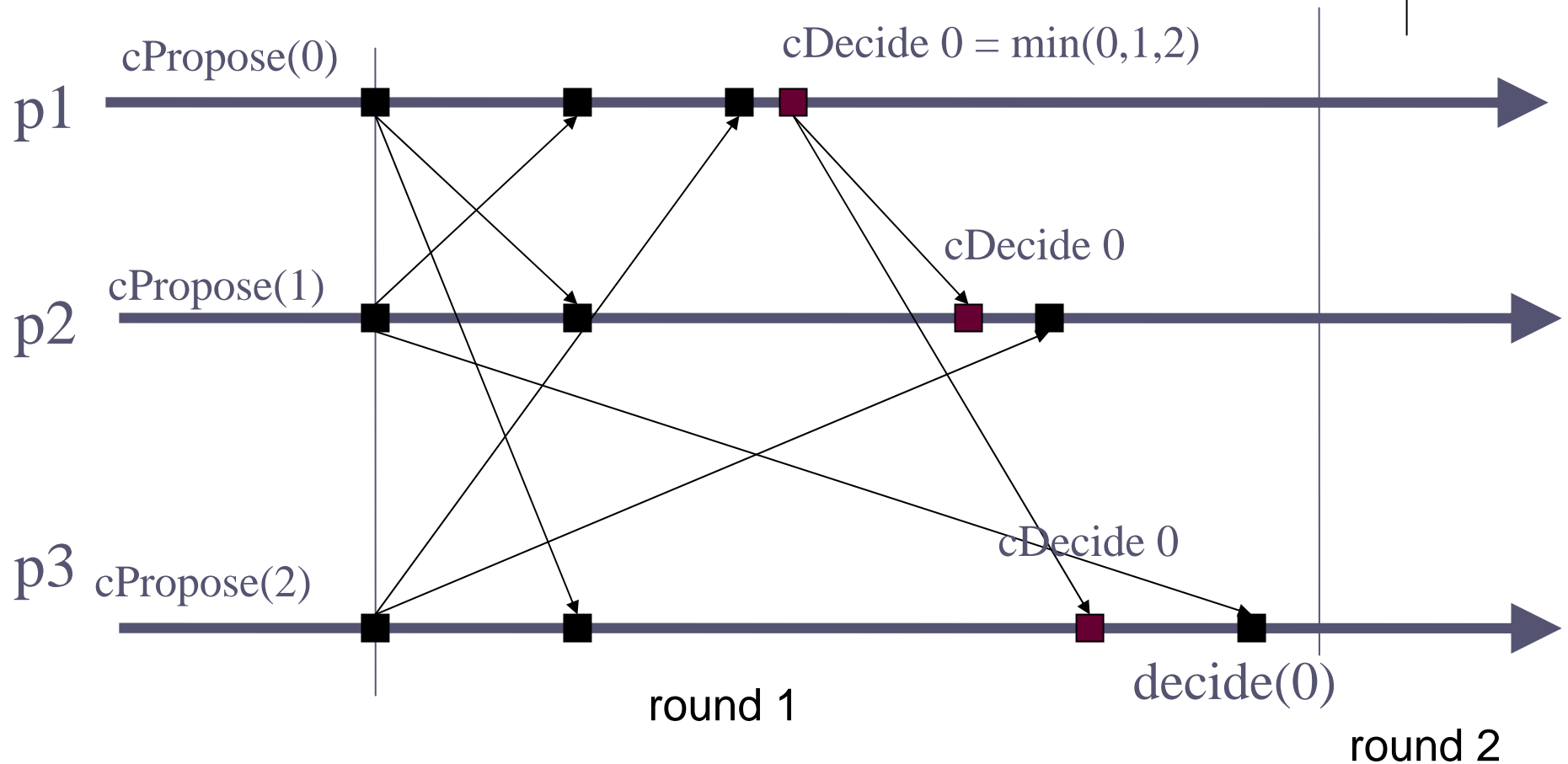
Flooding Consensus



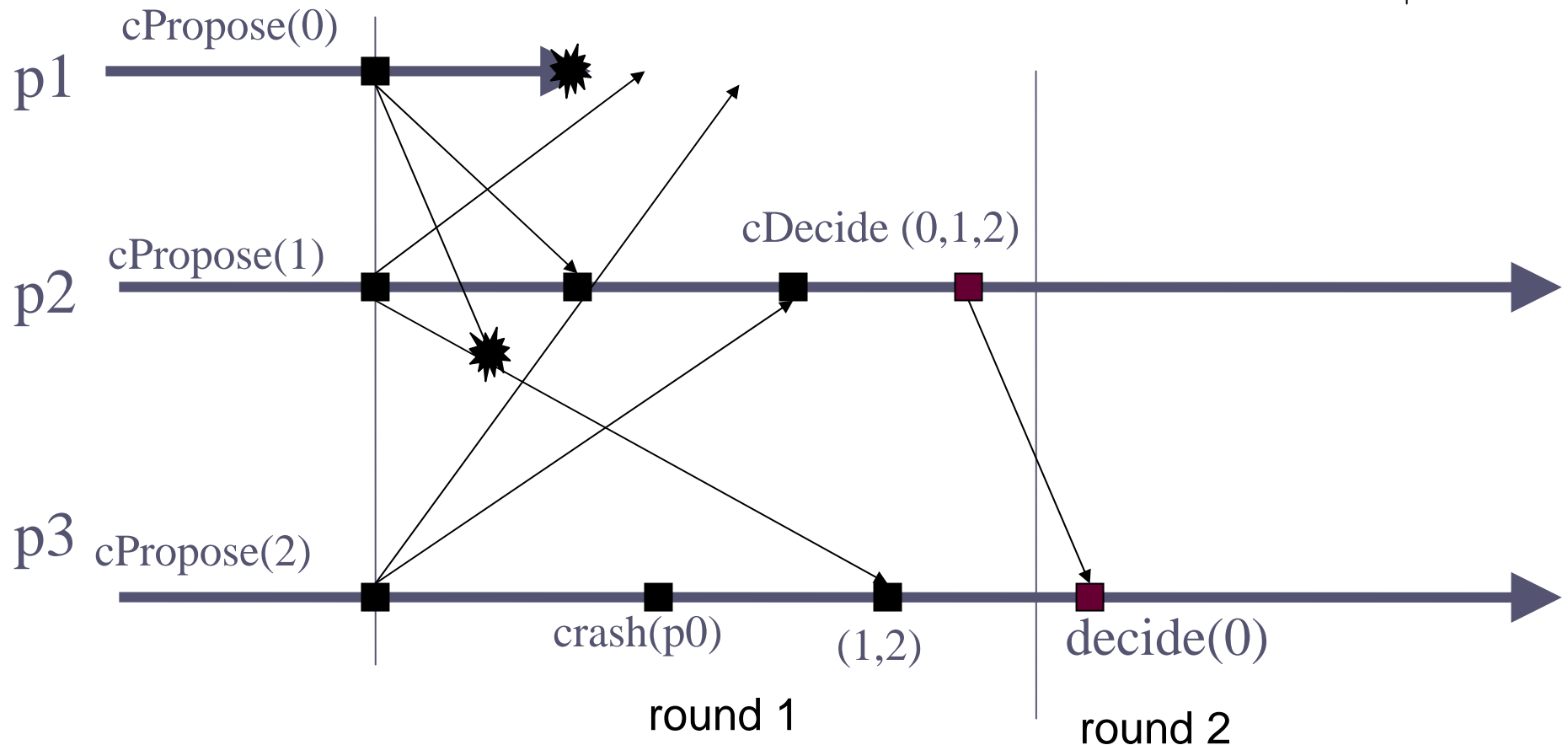
- **upon event** $\langle \text{bebDeliver } p_i, (\text{DECIDED}, v) \rangle$ **do**
 - **if** $p_i \in \text{correct}$ **and** $\text{decided} = \perp$ **then**
 - $\text{decided} := v$
 - **trigger** $\langle \text{cDecided } \text{decided} \rangle$
 - **trigger** $\langle \text{bebBroadcast } (\text{DECIDED}, v) \rangle$

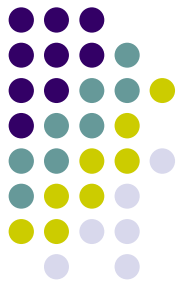
Consensus algorithm I'

No failure



Consensus algorithm I' failure

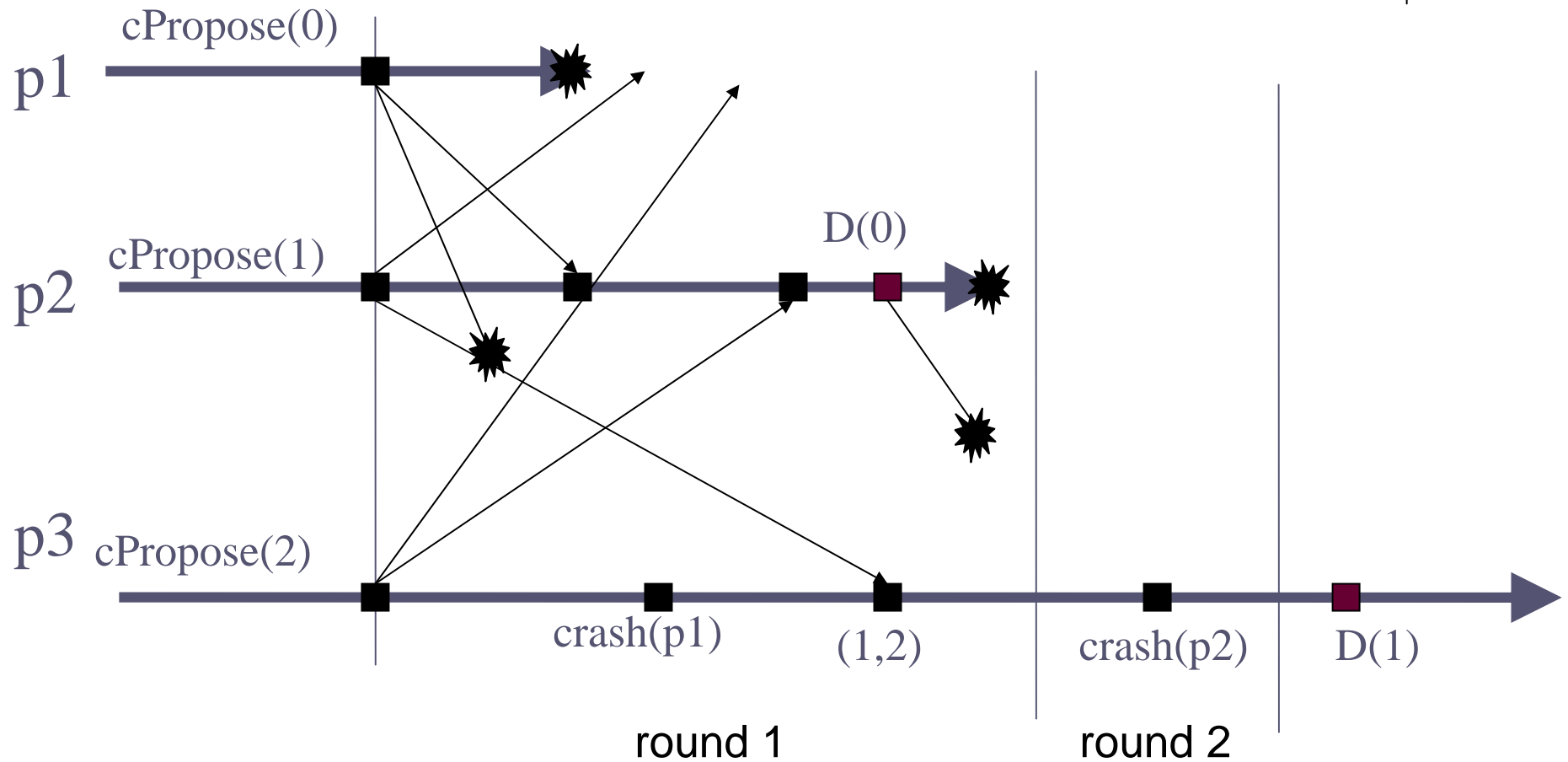
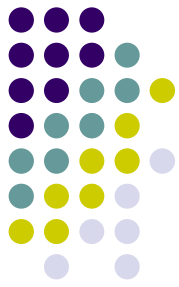




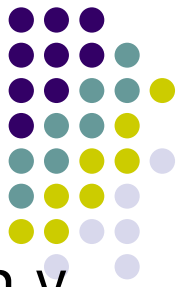
Consensus I'

- Why the algorithm (flooding consensus) does not ensure uniform agreement?

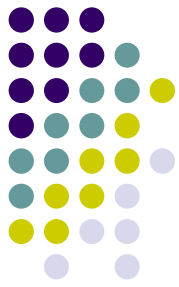
Consensus algorithm I' nonuniform



Correctness Agreement



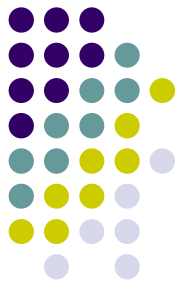
- Let r be the smallest round in which some p_i decides on v , we have two cases
- (1) p_i decides after observing two similar sets of correct processes at rounds $r-1$, and r . By accuracy property no process sees a smaller value than v by end of r
- Let p_j be such a process. Either p_j detects no failure on round r , and also decides on v , or detects some process p_k crashes and decides on round $r+1$ or later after delivering a DECIDED message from p_i
- (2) p_i decides after delivering a DECIDED message from p_k and p_k crashes in round r . Other correct process decide after delivering a DECIDED message from p_i in $r+1$



Consensus algorithm II

- Algorithm II implements uniform consensus
- The processes go through rounds incrementally, from 1 and at most until n
- The process with lowest rank that decides (before eventual crash) imposes its decision on other processes
- Every round has a leader: process p_i at round i

Consensus algorithm II



- A round consists of two communication steps
 - The leader broadcasts a message trying to impose its value
 - Processes that gets a proposal from the leader of the round adopts the proposal, and send ACK back
- If the leader gets all ACKs back, it commits to its proposal, and disseminate its decision by a reliable broadcast
- Once the decision reaches a correct process its guaranteed to be accepted even if the leader fails

Consensus algorithm II



- ☞ **Implements:** Uniform Consensus (uc)
- ☞ **Uses:** BestEffortBroadcast (beb)
 - ☞ PerfectFailureDetector (P)
 - ☞ ReliableBroadcast (rb)
 - ☞ PerfectP2PLink (pp2p)
- **upon event** $\langle \text{Init} \rangle$ **do**
 - $\text{detected} := \text{ackSet} := \emptyset$
 - $\text{round} := 1$; **broadcast** := false
 - $\text{proposal} := \text{decided} := \perp$
 - **for** $i = 1$ **to** N **do** $\text{proposed}[i] := \perp$



Consensus algorithm II

- **upon event** $\langle \text{crash } p_i \rangle$ **do**
 - $\text{detected} := \text{detected} \cup \{ \text{rank}(p_i) \}$
- **upon event** $\langle \text{ucPropose } v \rangle$ **do**
 - if** $\text{proposal} = \perp$ **then**
 - $\text{proposal} := v$

Consensus algorithm II



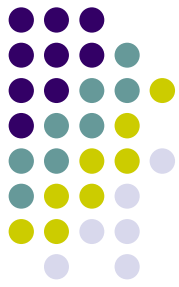
- upon round = rank(self) and
 - decided = \perp and not broadcast
 - proposal $\neq \perp$ do //prevents multiple proposals
 - trigger \langle bebBroadcast (PROPOSE, round, proposal) \rangle
 - broadcast := true

Consensus algorithm II



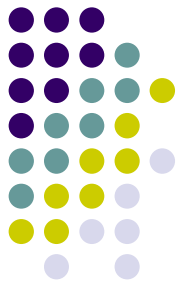
```
upon event ⟨bebDeliver pi (PROPOSAL, r, v)⟩ do
  proposed[ r ] := v
  if r ≥ round then
    trigger ⟨pp2pSend pi (ACK r)⟩
  upon event ⟨p2pDeliver pi (ACK r)⟩ do
    ackSet := ackSet ∪ { rank(pi) }
```

Consensus algorithm II



```
upon round ∈ detected do
  if proposed[ round ] ≠ ⊥ then
    proposal := proposed[ round ]
  round := round + 1
```

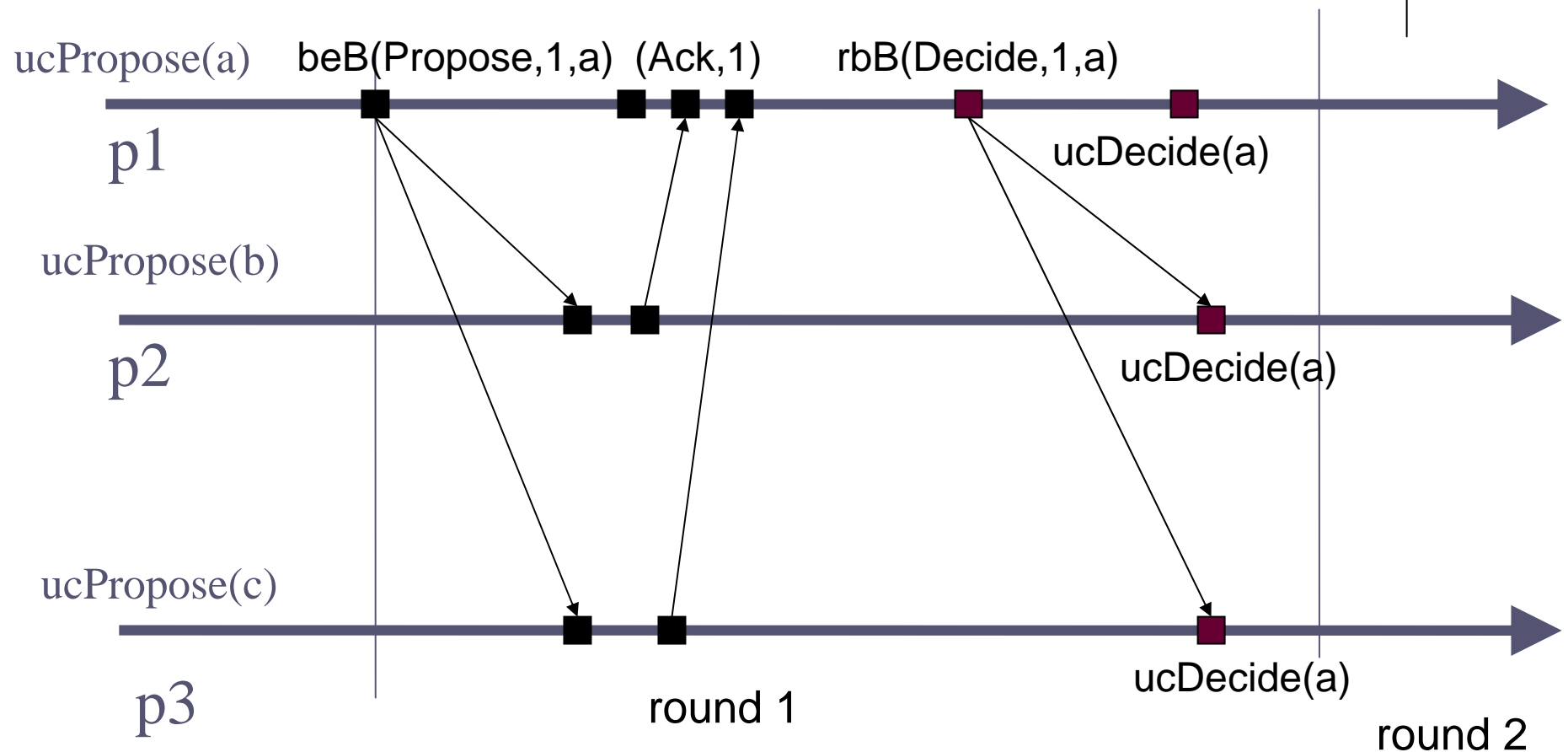
Consensus algorithm II



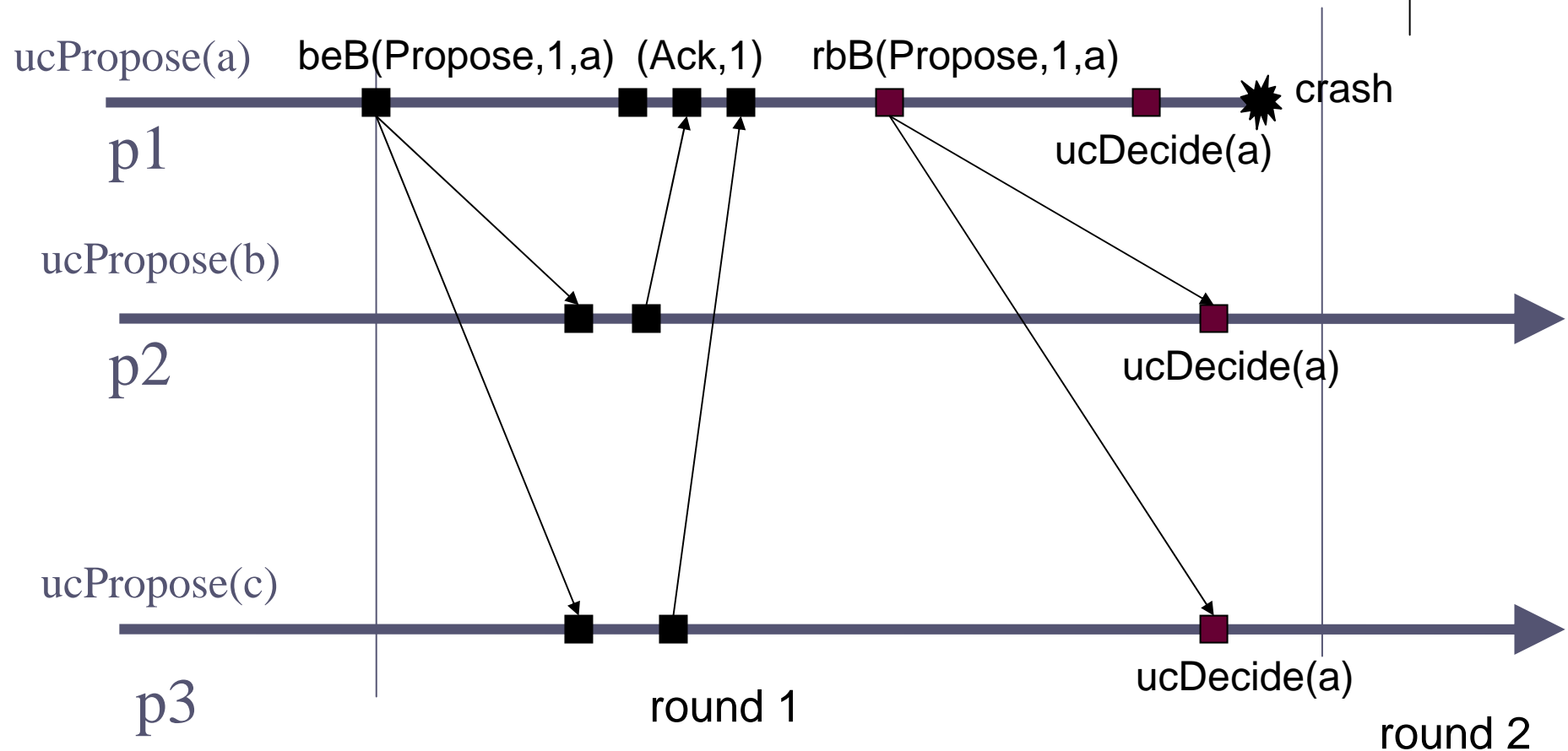
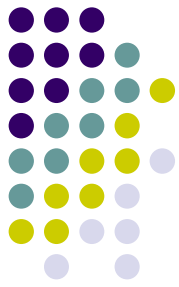
- ☛ **upon** $| \text{ackSet} \cup \text{detected} | = N$ **do**
 - ☛ **trigger** $\langle \text{rbBroadcast} (\text{DECIDED}, \text{proposal}) \rangle$
 - ☛ $\text{ackSet} := \emptyset$
- ☛ **upon event** $\langle \text{rbDeliver } p_i, (\text{DECIDED}, \text{proposal}) \rangle$ **do**
 - ☛ **if** $\text{decided} = \perp$ **then**
 - ☛ $\text{decided} := v$
 - ☛ **trigger** $\langle \text{ucDecide } v \rangle$

Consensus algorithm II

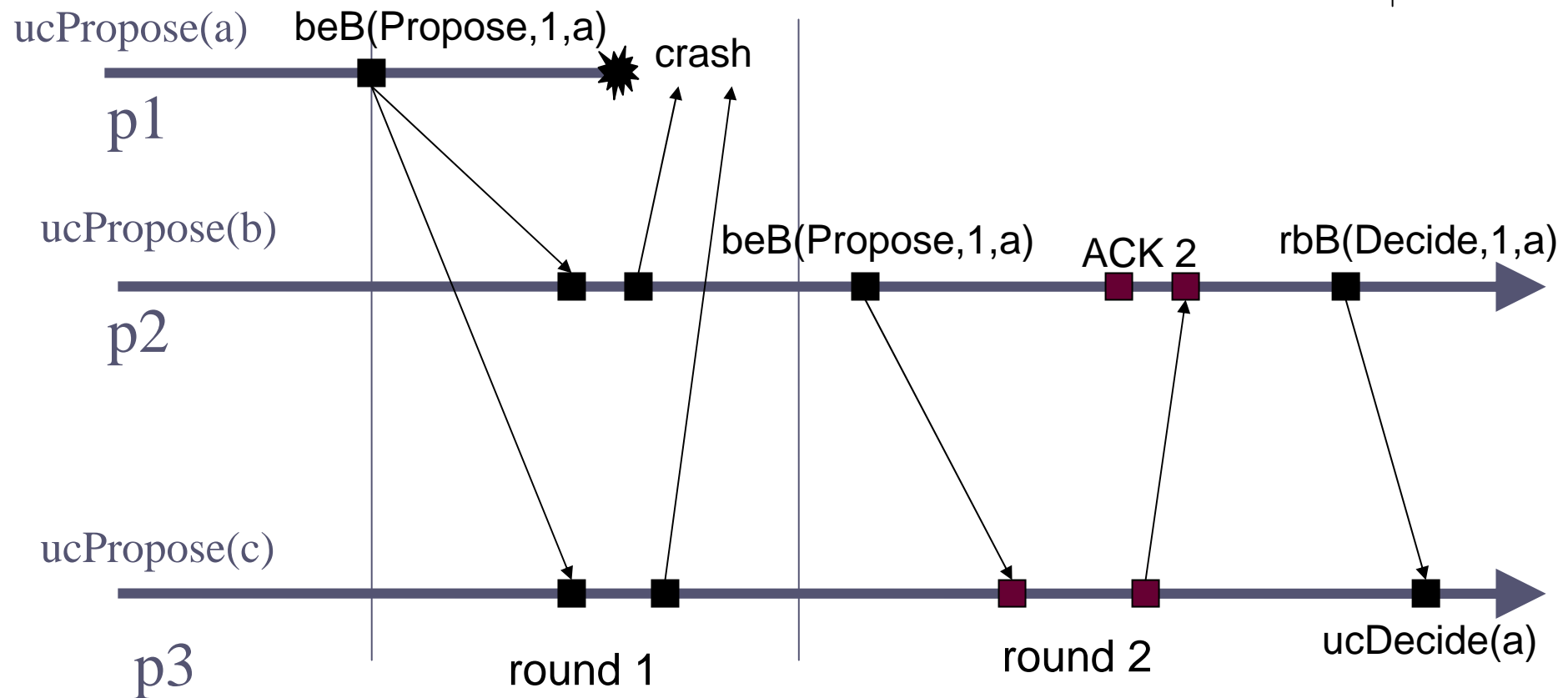
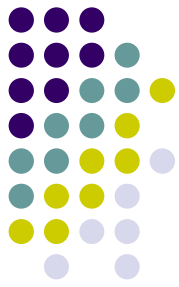
No failure

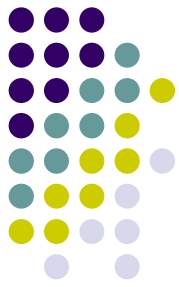


Consensus algorithm II failure



Consensus algorithm II failure

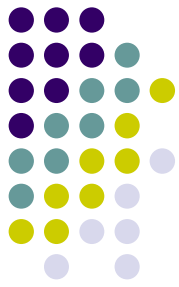




Correctness argument

- Validity and Integrity
 - follows from the properties of the underlying communication, and the algorithm
- Agreement
 - Assume two processes decide differently, this can happen if two decisions were `rbBroadcast`
 - Assume p_i and p_j , $j > i$, `rbBroadcast` two decisions v_i and v_j , because of accuracy of P , p_j must have adopted the value v_i

Consensus algorithm III



- A uniform consensus algorithm assuming:
 - a correct majority
 - a $\diamond P$ failure detector
- Basic idea: the processes alternate in the role of a phase coordinator until one of them succeeds in imposing a decision



Consensus algorithm III

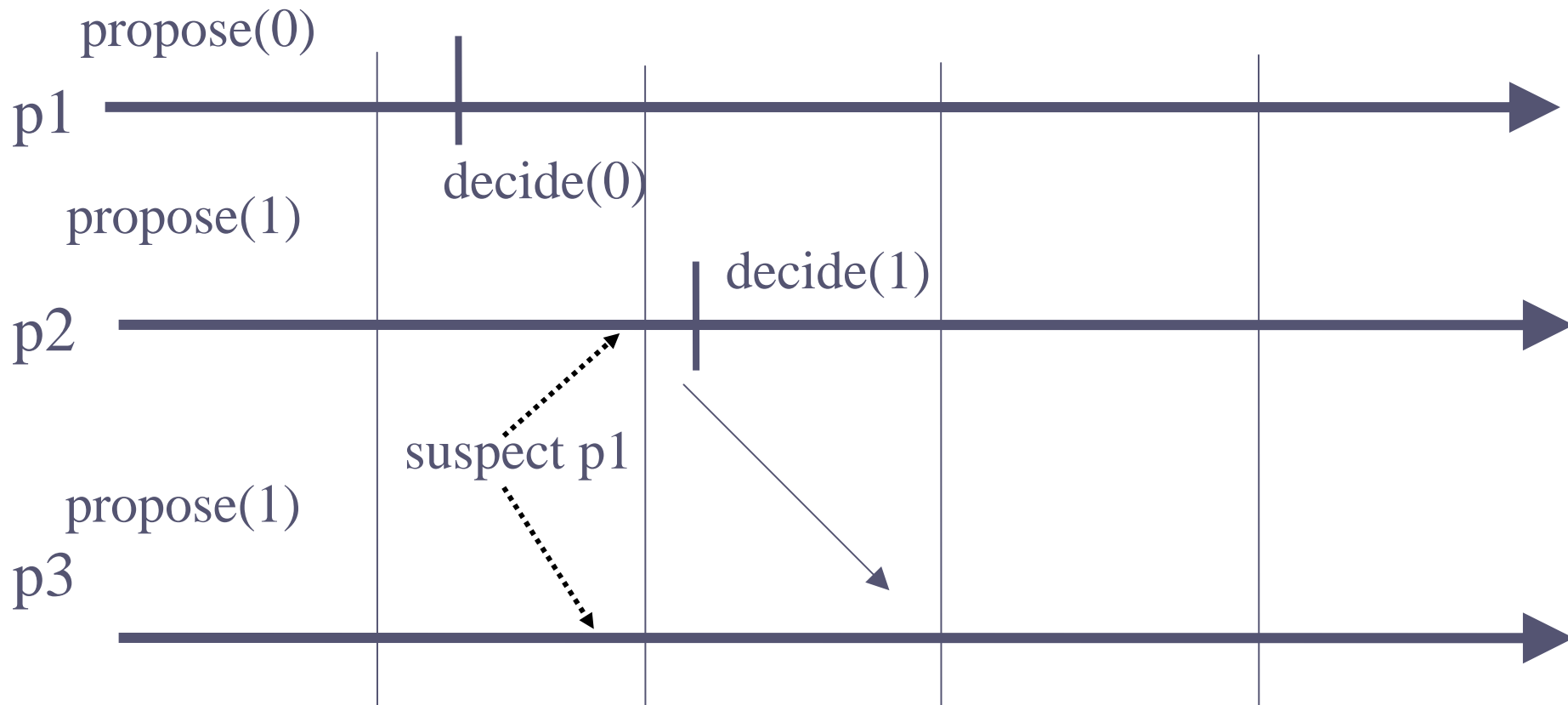
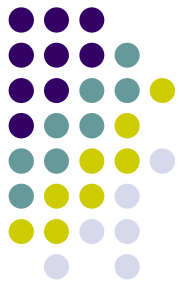
- $\diamond P$ ensures:
 - ***Strong completeness:*** eventually every process that crashes is permanently suspected by all correct processes
 - ***Eventual strong accuracy:*** eventually no correct process is suspected by any process

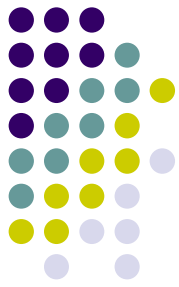


"◇" makes a difference

- ***Eventual strong accuracy***: strong accuracy holds only *after finite time*.
- Correct processes may be *falsely suspected* a finite number of times.
- This breaks consensus algorithms I and II
 - Counterexamples for algorithm I and II (see next slide)

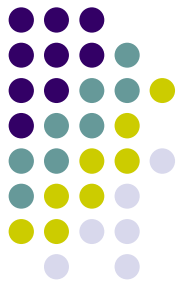
Agreement violated with $\diamond P$ in algorithm I





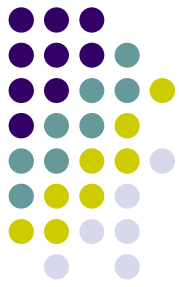
Consensus algorithm III

- The algorithm is also round-based: processes move incrementally from one round to the other
- Process p_i is *leader* in every round r such that $(r-1) \bmod N + 1 = i$
- In such a round, p_i *tries to decide* (next 2 slides)



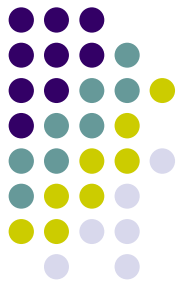
Consensus algorithm III

- p_i succeeds if it is not suspected (processes that suspect p_i inform p_i and move to the next round; p_i does so as well)
- If p_i succeeds, p_i uses a reliable broadcast to send the decision to all (the reliability of the broadcast is important here to preclude the case where p_i crashes, some other processes delivers the message and stop and the rest keeps going without the majority)



Consensus algorithm III

- To decide, p_i executes steps 1-2-3
 - 1. p_i selects among a majority the latest adopted value (latest with respect to the round in which the value is adopted – see step 2)
 - 2. p_i imposes that value at a majority: any process in that majority adopts that value – p_i fails if it is suspected
 - 3. p_i decides and broadcasts the decision to all



Algorithm 5.11 Rotating Coordinator: leader role

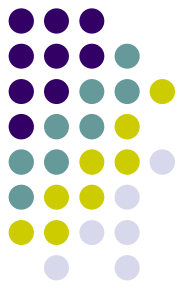
Uses:

PerfectPointToPointLinks (pp2p);

ReliableBroadcast (rb);

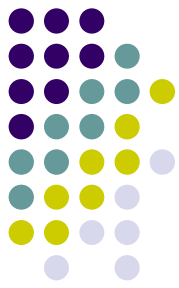
BestEffortBroadcast (beb);

EventuallyPerfectFailureDetector ($\diamond\mathcal{P}$);



function leader (r) **returns** processid **is**
 return $p_i: (\text{rank}(p_i) = (r \bmod N + 1))$;

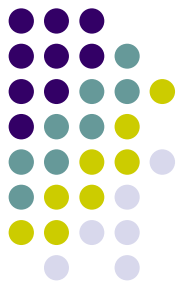
upon event $\langle \text{Init} \rangle$ **do**
 round := 1; proposal := decided := \perp ;
 suspected := estimate-set[] := ack-set[] := nack-set[] := \emptyset ;
 forall r **do** estimate[r] := ack[r] := false; proposed[r] := \perp



upon event $\langle ucPropose \mid v \rangle \wedge proposal \neq \perp$ **do**
 $proposal := (v, 0);$

upon event $\langle pp2pDeliver \mid p_i, [ESTIMATE, r, e] \rangle$ **do**
 $estimate-set[r] := estimate-set[r] \cup \{e\};$

upon $(leader(round)=self) \wedge (|estimate-set[round]| > N/2)$ **do**
 $proposal := highest(estimate-set[round]);$
 trigger $\langle bebBroadcast \mid [PROPOSE, round, proposal] \rangle;$

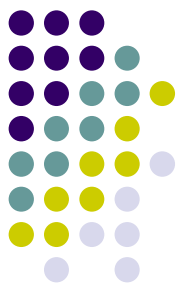


upon event $\langle pp2pDeliver \mid p_i, [ACK, r] \rangle$ **do**
 $ack\text{-}set[r] := ack\text{-}set[r] \cup \{p_i\};$

upon event $\langle pp2pDeliver \mid p_i, [NACK, r] \rangle$ **do**
 $nack\text{-}set[r] := nack\text{-}set[r] \cup \{p_i\};$

upon $(leader(round)=self) \wedge nack\text{-}set[round] \neq \emptyset$ **do**
 $round := round + 1;$

upon $(leader(round)=self) \wedge (|ack\text{-}set[round]| > N/2)$ **do**
 trigger $\langle rbBroadcast \mid [DECIDE, proposal] \rangle;$

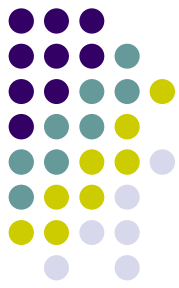


Algorithm 5.12 Rotating Voordinator: witness role

upon event $(\text{proposal} \neq \perp) \wedge (\text{estimate}[\text{round}] = \text{false})$ **do**
 $\text{estimate}[\text{round}] := \text{true};$
 trigger $\langle pp2pSend \mid \text{leader}(\text{round}), [\text{ESTIMATE}, \text{round}, \text{proposal}] \rangle$

upon event $\langle bebDeliver \mid p_i, [\text{PROPOSE}, r, v] \rangle$ **do**
 $\text{proposed}[r] := v;$

upon event $(\text{proposed}[\text{round}] \neq \perp) \wedge (\text{ack}[\text{round}] = \text{false})$ **do**
 $\text{proposal} := (\text{proposed}[\text{round}], \text{round});$
 $\text{ack}[\text{round}] := \text{true};$
 trigger $\langle pp2pSend \mid \text{leader}(\text{round}), [\text{ACK}, \text{round}] \rangle;$
 $\text{round} := \text{round} + 1;$

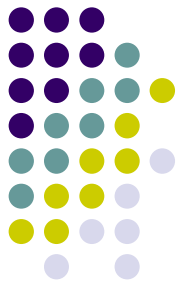


upon event $(\text{leader}(\text{round}) \in \text{suspected}) \wedge (\text{ack}[\text{round}] = \text{false})$ **do**
 $\text{ack}[\text{round}] := \text{true};$
 trigger $\langle \text{pp2pSend} \mid \text{leader}(\text{round}), [\text{NACK}, \text{round}] \rangle;$
 $\text{round} := \text{round} + 1;$

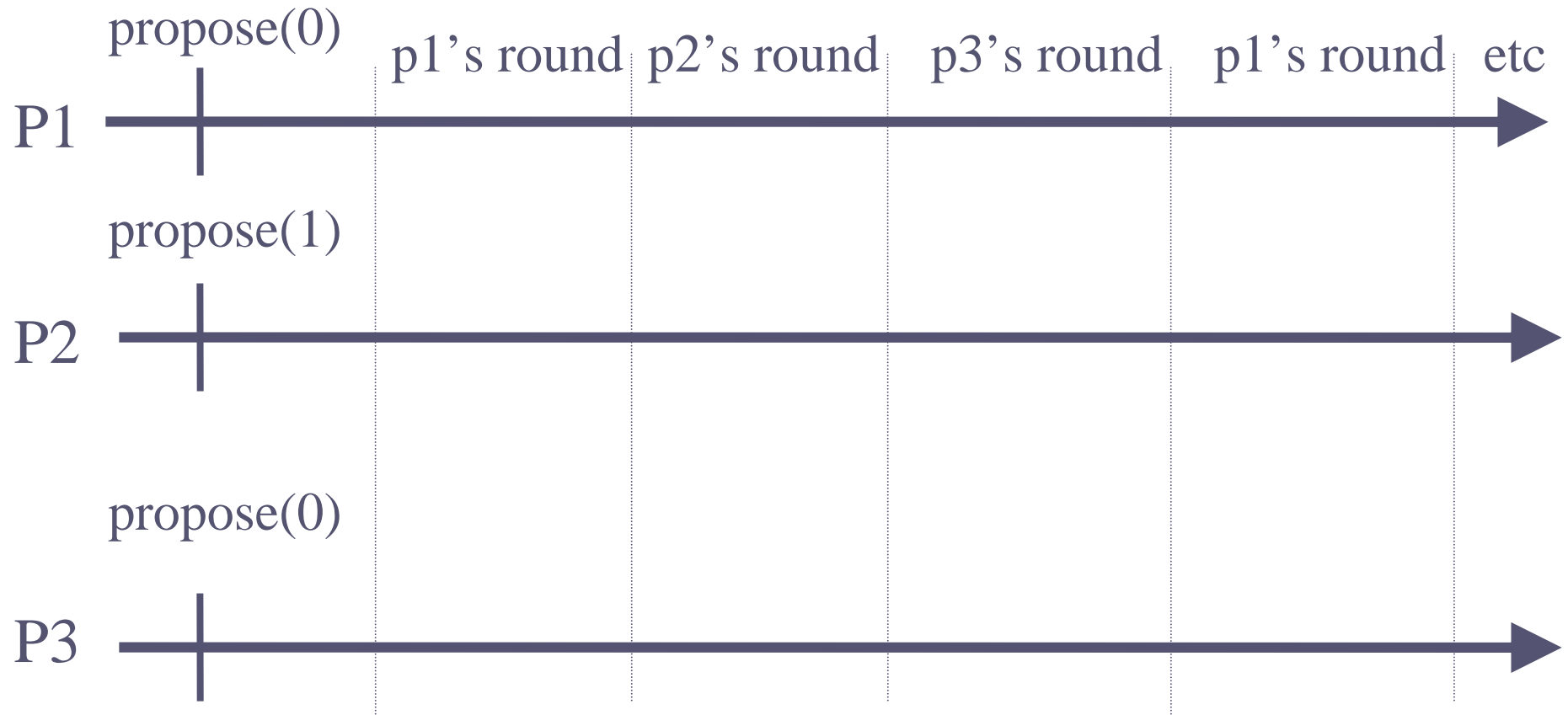
upon event $\langle \text{rbDeliver} \mid p_i, [\text{DECIDED}, v] \rangle \wedge (\text{decided} = \perp)$ **do**
 $\text{decided} := v;$
 trigger $\langle \text{ucDecided} \mid v \rangle;$

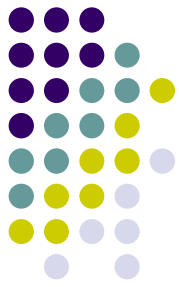
upon event $\langle \text{suspect} \mid p_i \rangle$ **do**
 $\text{suspected} := \text{suspected} \cup \{p_i\};$

upon event $\langle \text{restore} \mid p_i \rangle$ **do**
 $\text{suspected} := \text{suspected} \setminus \{p_i\};$

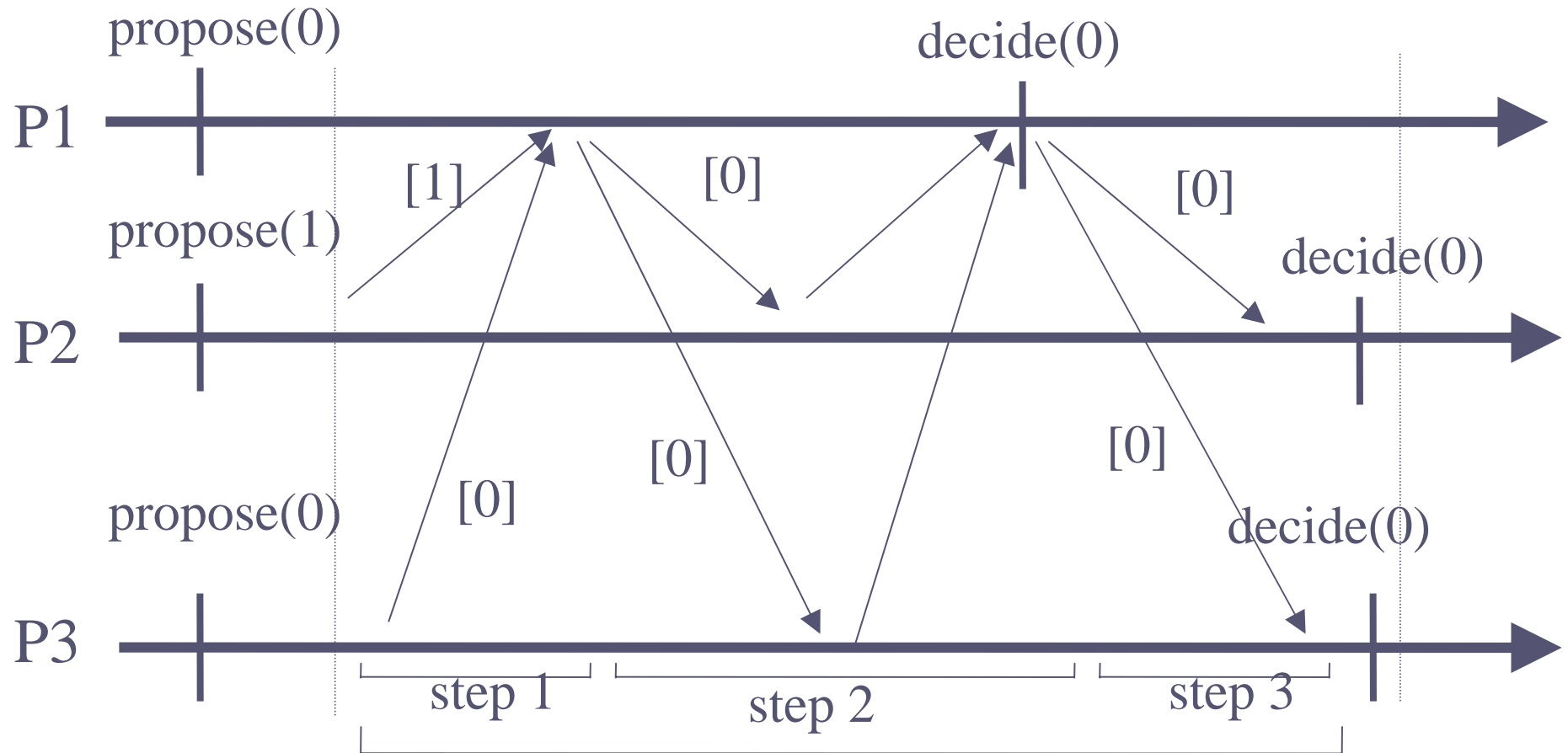


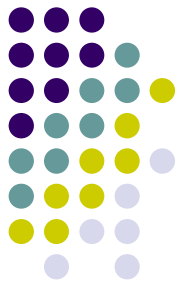
Consensus algorithm III



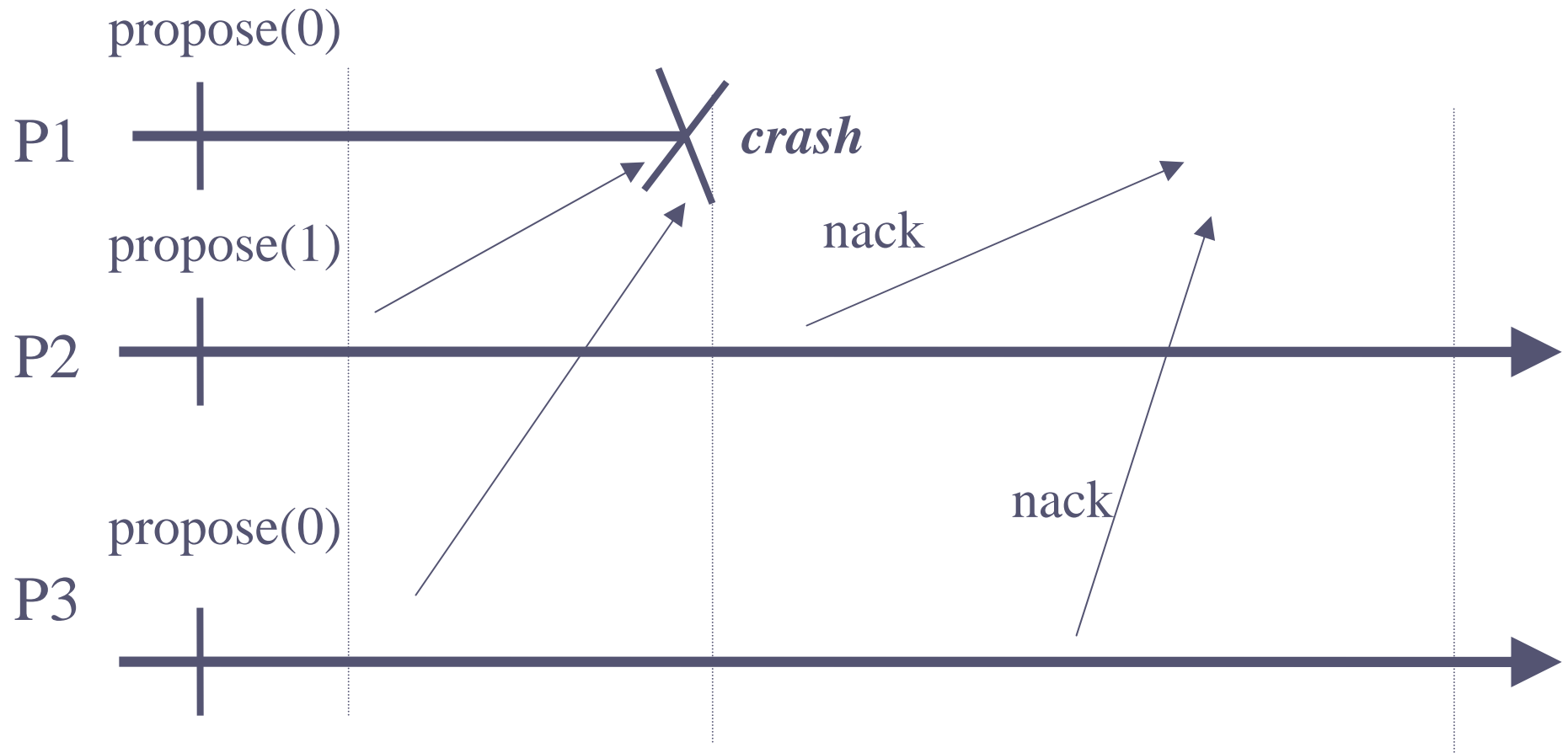


Consensus algorithm III



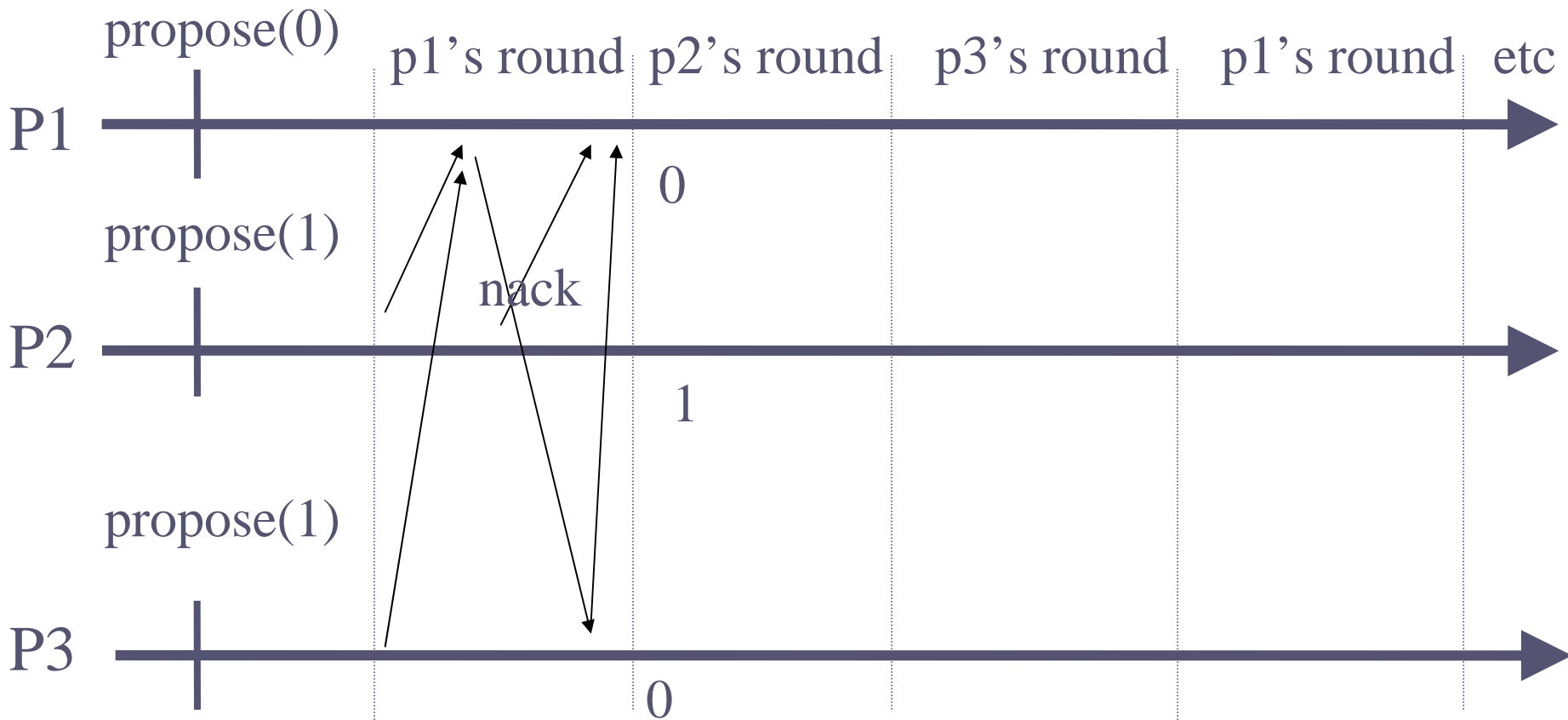


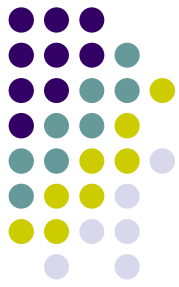
Consensus algorithm III





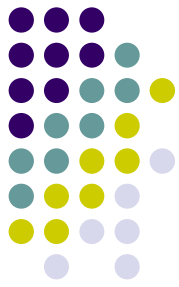
Consensus algorithm III





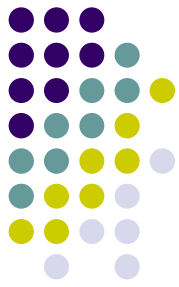
Correctness argument A

- *Validity* and *integrity* are trivial
- Consider *termination*: if a correct process decides, it uses reliable broadcast to send the decision to all: every correct process decides
- Assume by contradiction that some process is correct and no correct process decides. We argue that this is impossible.



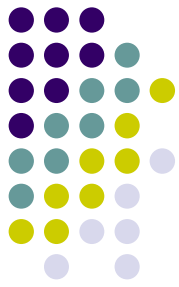
Correctness argument A'

- By the correct *majority* assumption and the *completeness* property of the failure detector, no correct process remains blocked forever in some phase.
- By the *accuracy* property of the failure detector, some correct process reaches a phase where it is leader and it is not suspected and reaches a decision in that phase: a contradiction



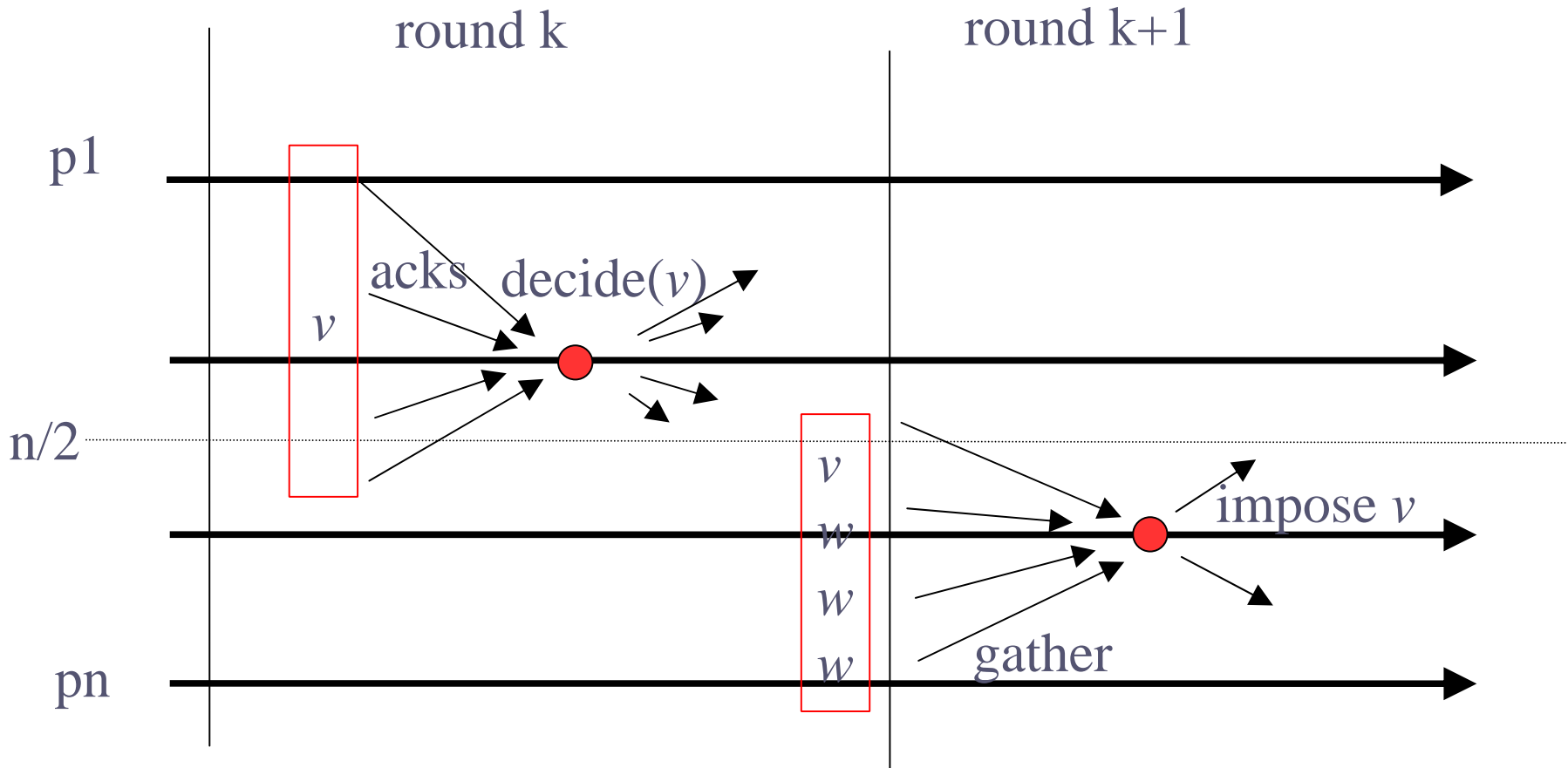
Correctness argument B

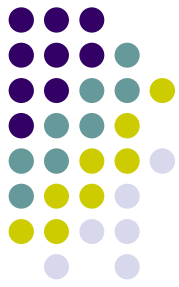
- Consider now *agreement*
- Let k be the first round in which some process p_i decides some value v , i.e., p_i is the leader of round k and p_i decides v in k
- This means that, in round k , a majority of processes have adopted v
- By the algorithm, no value else than v will be proposed (and hence decided) by any process in a round higher than k



new

Correctness argument B

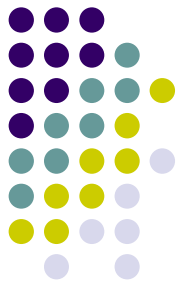




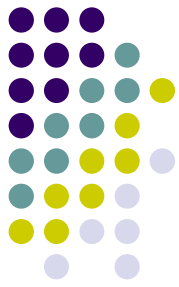
Agreement is never violated

- Look at "totally unreliable" failure detector (provides no guarantees)
 - may always suspect everybody
 - may never suspect anybody
- Agreement is never violated
 - Can use the same correctness argument as before
 - Termination not ensured (everybody may be suspected infinitely often)

Randomized Consensus Fail-Silent Model



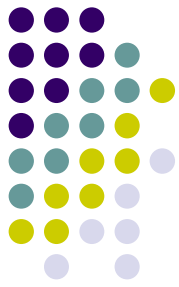
- **Name:**
 - RandomizedConsensus (rc)
- **Events:**
 - **Request:** $\langle \text{rcPropose } v \rangle$
 - **Indication:** $\langle \text{rcDecide} \rangle$
- **Properties**
 - RC1, RC2, RC3, RC4



Properties

- **RC1: Termination**
 - With probability 1, every correct process decides some value
- **RC2: Validity**
 - If a process decides a value v , then v was proposed by some process
- **RC3: Integrity**
 - No process decides twice
- **RC3: Agreement**
 - No two correct processes decide differently

Algorithm IV



- **Implements:**
 - Randomized Consensus (rc)
- **Uses:**
 - ReliableBroadcast (rb)
 - BestEffortBroadcast (beb)
- **upon event** $\langle \text{Init} \rangle$ **do**
 - $\text{decided} := \text{estimate} := \perp$
 - $\text{round} := 0; \text{val} := \emptyset$
 - **forall** r **do**
 - $\text{phase1}[r] := \emptyset$
 - $\text{phase2}[r] := \emptyset$



Algorithm IV (phase 1)

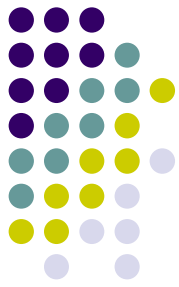
- **upon event** $\langle \text{rcPropose } v \rangle$ **do**
 - **trigger** $\langle \text{bebBroadcast (INIVAL } v) \rangle$
 - $\text{estimate} := v$
 - $\text{round} := \text{round} + 1$
 - $\text{val} := \text{val} \cup \{ v \}$
 - **trigger** $\langle \text{bebBroadcast (PHASE1, round, val)} \rangle$
- **upon event** $\langle \text{bebDeliver } pi, (\text{INIVAL } v) \rangle$ **do**
 - $\text{val} := \text{val} \cup \{ v \}$
- **upon event** $\langle \text{bebDeliver } pi, (\text{ PHASE1, } r, v) \rangle$ **do**
 - $\text{phase1}[r] := \text{phase1}[r] \cup \{ v \}$

Algorithm IV (phase 2)



- **upon** $\text{decided} = \perp$ **and** $|\text{phase1}[\text{round}]| > N/2$ **do**
 - **if exists** $v: \forall x \in \text{phase1}[\text{round}]: x=v$ **then**
 - $\text{estimate} := v$
 - **else**
 - $\text{estimate} := \perp$
 - **trigger** $\langle \text{bebBroadcast}(\text{PHASE2}, \text{round}, \text{estimate}) \rangle$
- **upon event** $\langle \text{bebDeliver } \pi_i, (\text{PHASE2}, r, v) \rangle$ **do**
 - $\text{phase2}[r] := \text{phase2}[r] \cup \{v\}$

Algorithm IV (phase 2)

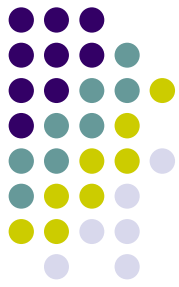


- **upon** $\text{decided} = \perp$ **and** $|\text{phase2}[\text{round}]| > N/2$ **do**
 - **if exists** $v \neq \perp: \forall x \in \text{phase2}[\text{round}]: x=v$ **then**
 - $\text{decided} := v$
 - **trigger** $\langle \text{rbBroadcast}(\text{DECIDED}, \text{round}, \text{decided}) \rangle$
 - **else**
 - **if exists** $v \in \text{phase2}[\text{round}]: v \neq \perp$ **then**
 - $\text{estimate} := v$
 - **else**
 - $\text{estimate} := \text{random}(\text{val})$
 - $\text{round} := \text{round} + 1$ // start one more round
 - **trigger** $\langle \text{bebDeliver}(\text{PHASE1}, \text{round}, \text{estimate}) \rangle$

Algorithm IV (phase 2)

- **upon event** $\langle \text{rbDeliver}(\text{DECIDED}, r, v) \rangle$ **do**
 - $\text{decided} := v$
 - **trigger** $\langle \text{rcDecide } v \rangle$





Summary

- (Uniform) Consensus problem is an important problem to maintain consistency
- Three algorithms:
 - I: consensus using P
 - II: uniform consensus using P
 - III: uniform consensus using $\langle \rangle P$ and a correct majority