CONSENSUS

DISTRIBUTED SYSTEMS

Master of Science in Cyber Security

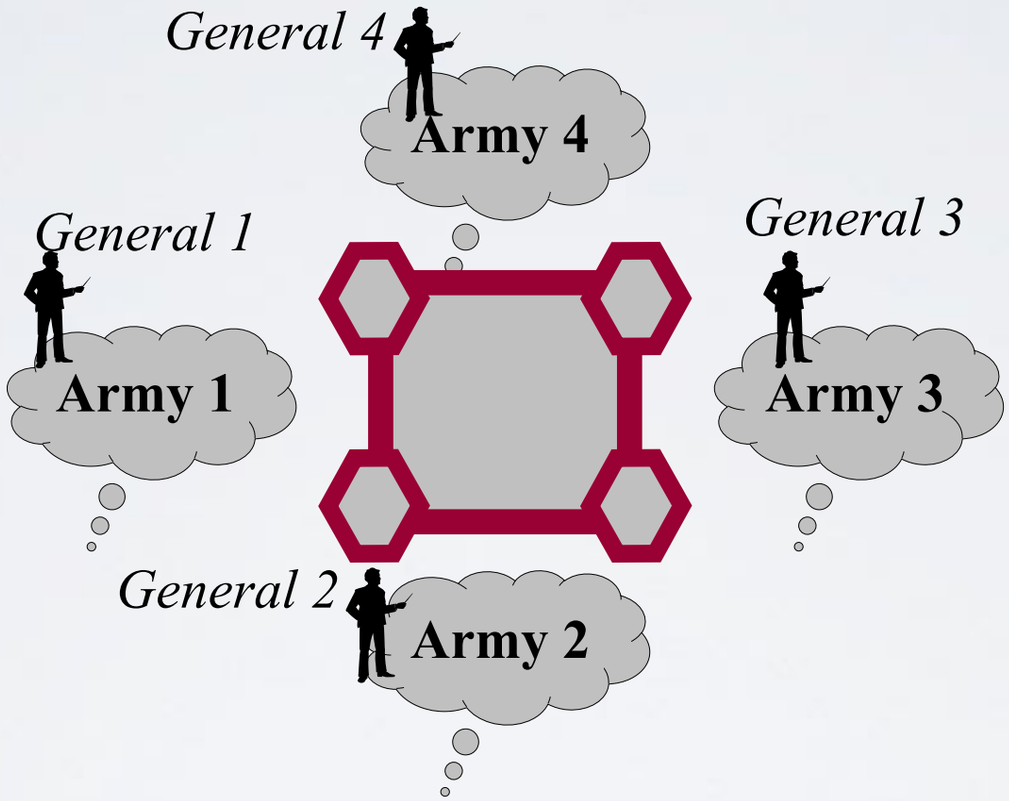
SAPIENZA
UNIVERSITÀ DI ROMA

CIS SAPIENZA
CYBER INTELLIGENCE AND INFORMATION SECURITY

CONSENSUS HISTORY

- 1982 Byzantine General problem
- 1985 FLP
- 1989 Paxos
- 1999 PBFT
- 2001 Paxos made simple
- 2006 Chubby
- 2009 Zookeeper
- ....

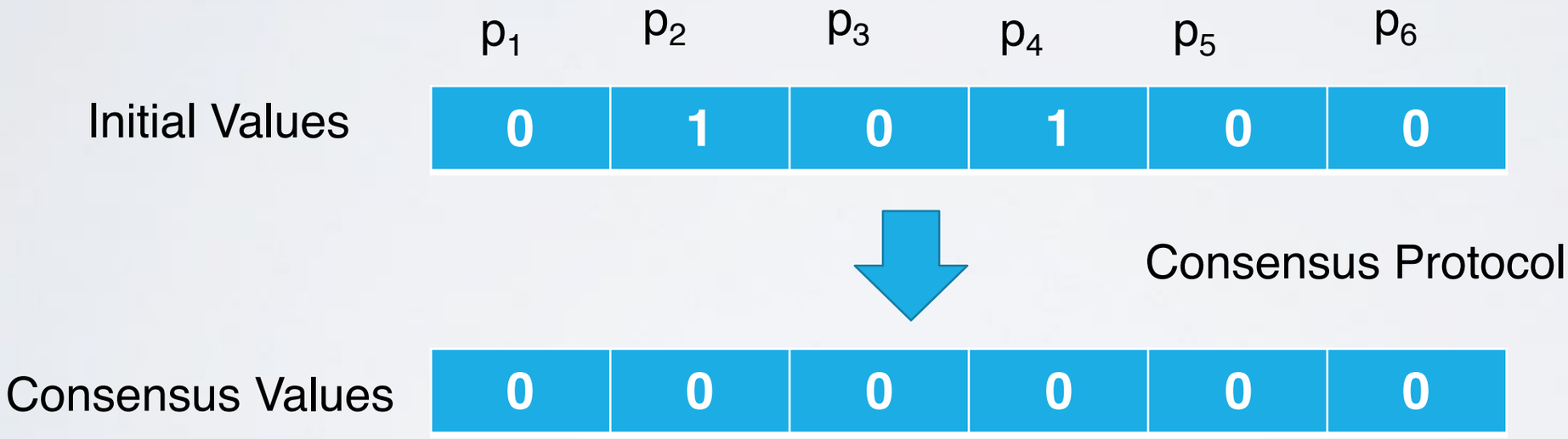
CONSENSUS EXAMPLE



Generals must reach consensus: attack or retreat?

SINGLE-VALUE CONSENSUS DEFINITION

- Set of initial values ∈ {0,1}.
- All process shall decide the same value ∈ {0,1} based on the initial proposals.



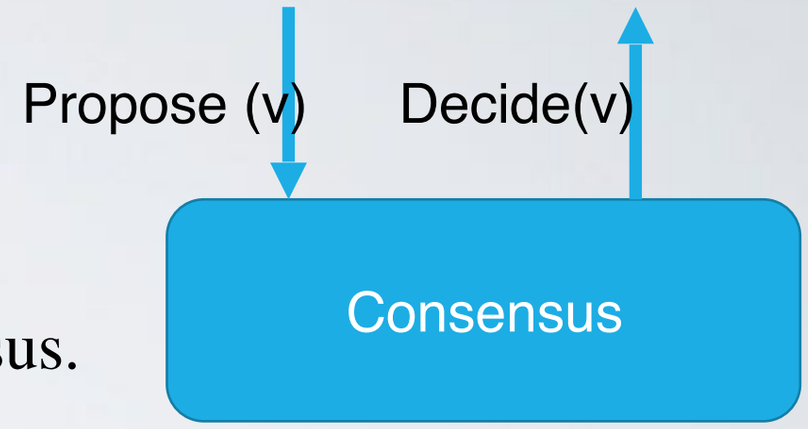
CONSENSUS: IN SYNCHRONOUS SYSTEMS, ALGORITHMS

CONSENSUS SPECIFICATION

Module 5.1: Interface and properties of (regular) consensus

Module:

Name: Consensus, **instance** c.



Events:

Request: $\langle\ c, \mathit{Propose} \mid v\ \rangle$: Proposes value v for consensus.

Indication: $\langle\ c, \mathit{Decide} \mid v\ \rangle$: Outputs a decided value v of consensus.

Properties:

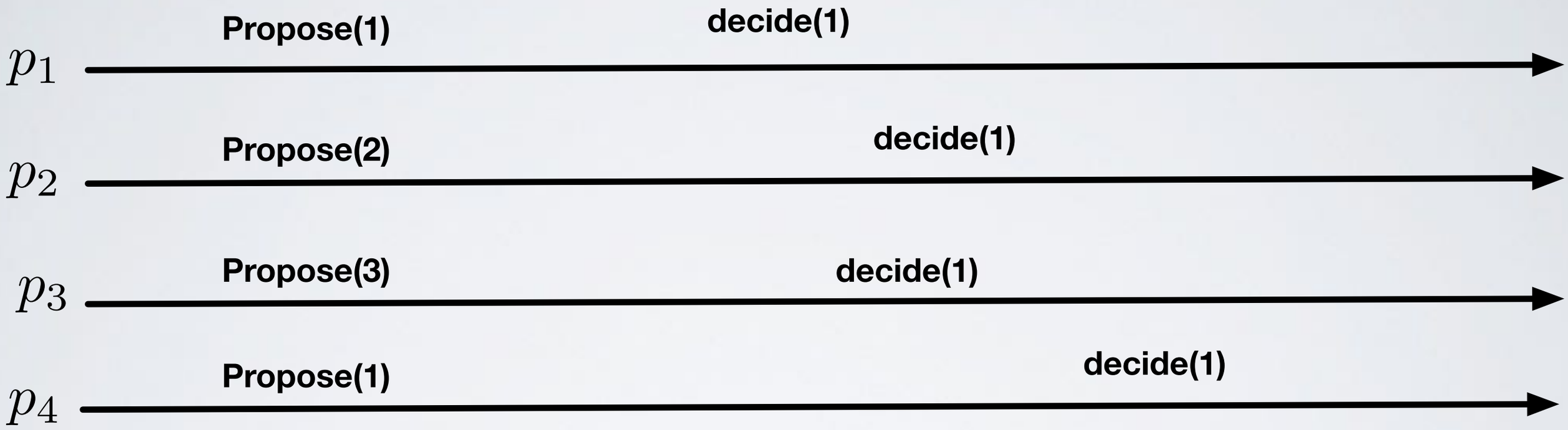
C1: *Termination:* Every correct process eventually decides some value.

C2: *Validity:* If a process decides v, then v was proposed by some process.

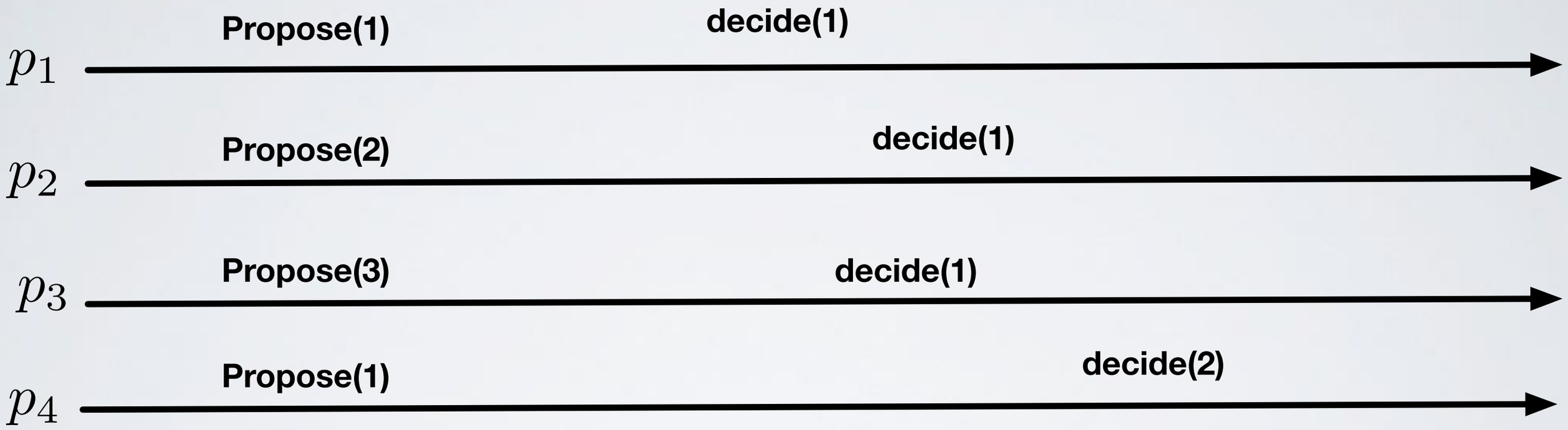
C3: *Integrity:* No process decides twice.

C4: *Agreement:* No two correct processes decide differently.

RUN

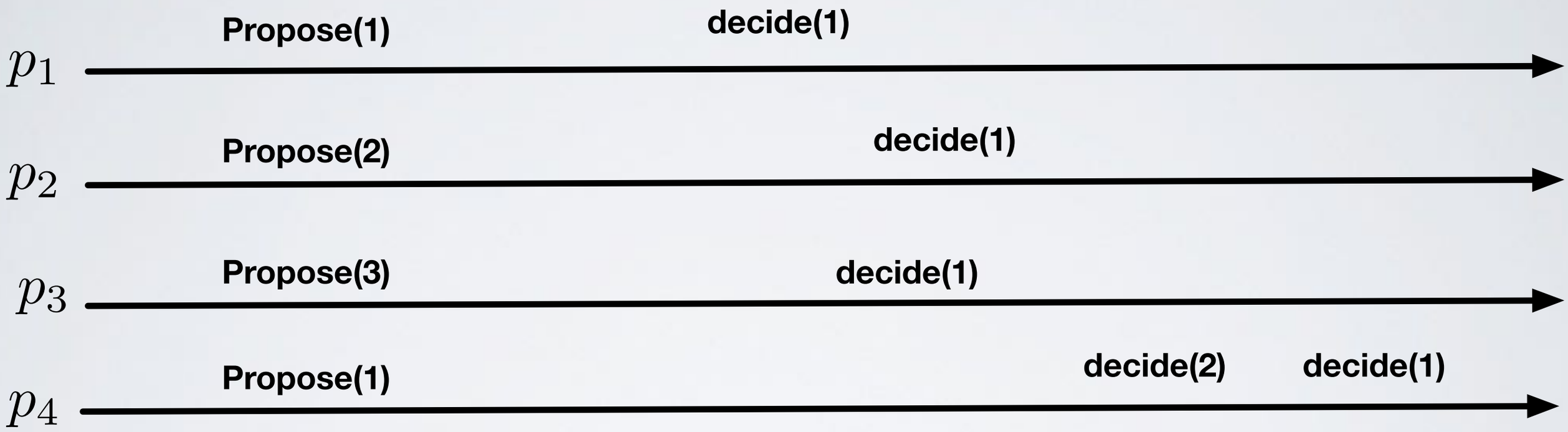

Propose(1)
decide(1)
p_1
Propose(2)
decide(1)
p_2
Propose(3)
decide(1)
p_3
Propose(1)
decide(1)
p_4

VIOLATED RUN 1



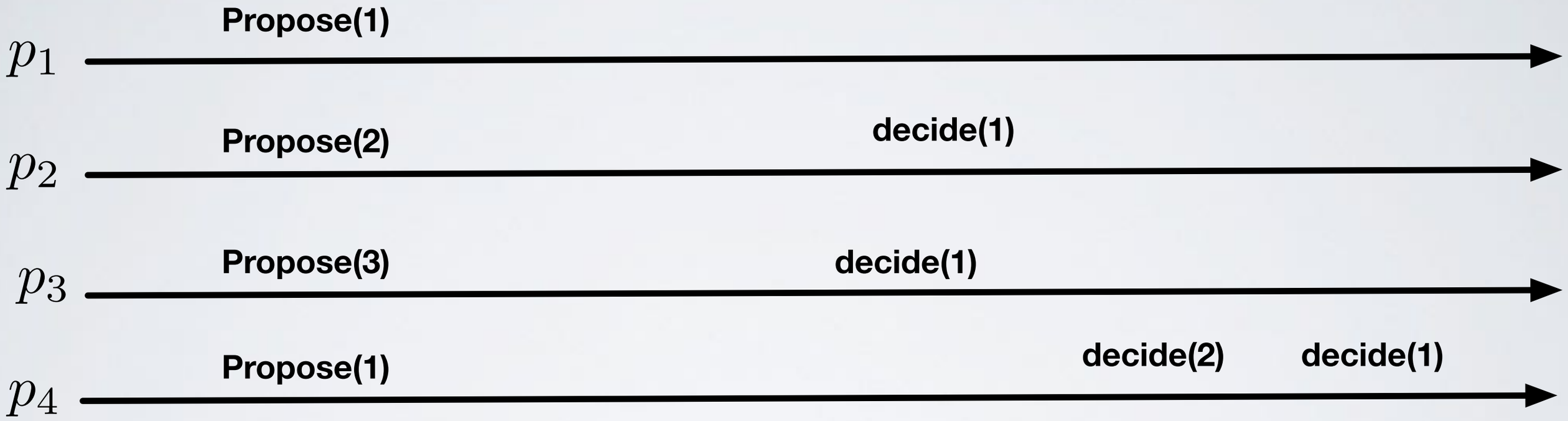
Agreement violated

VIOLATED RUN 2



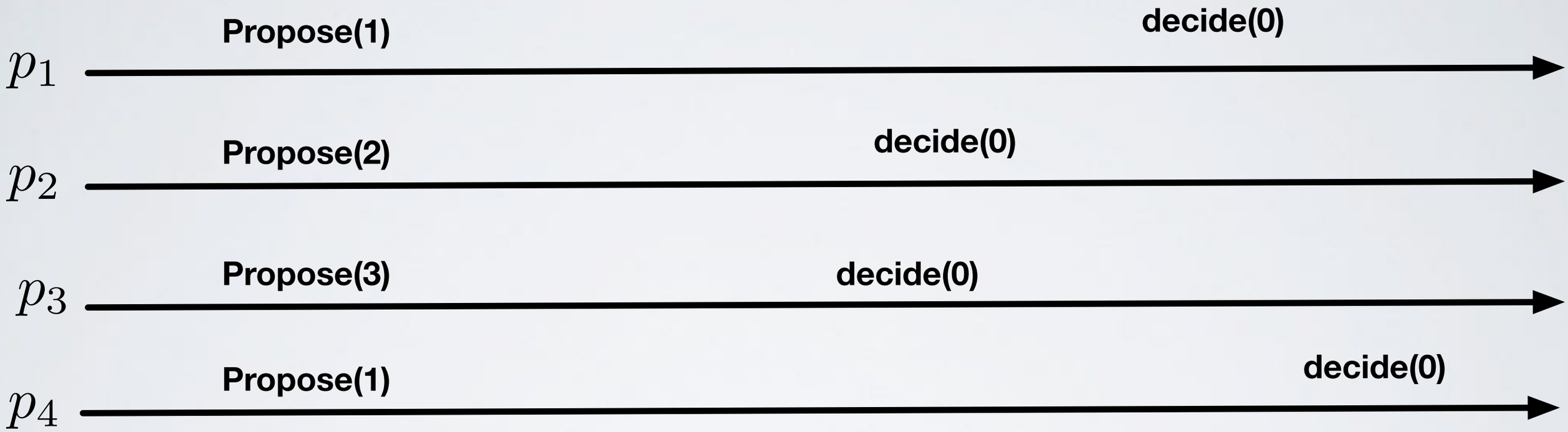
Integrity violated

VIOLATED RUN 3



Termination violated

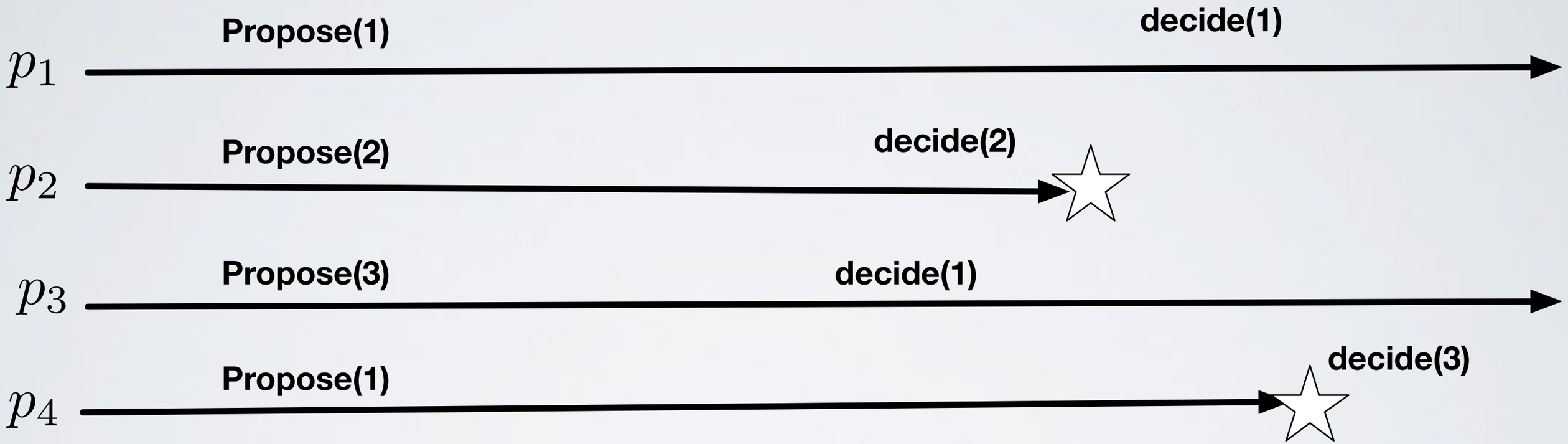
VIOLATED RUN 4



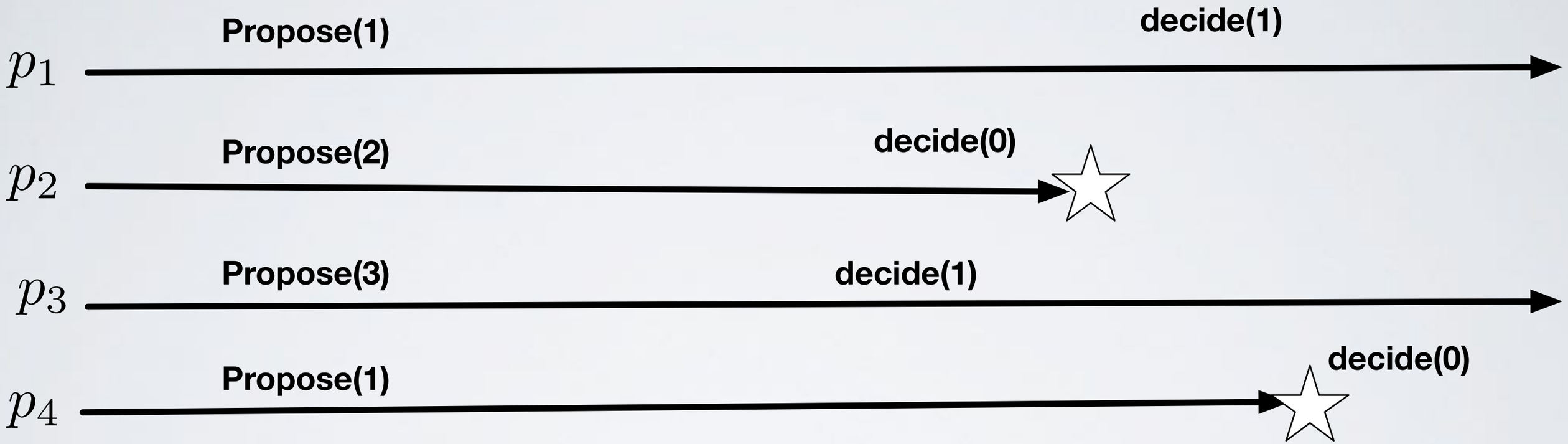
Validity violated.

Validity is also called non-triviality (could you see why?)

VIOLATED RUN?



VIOLATED RUN?

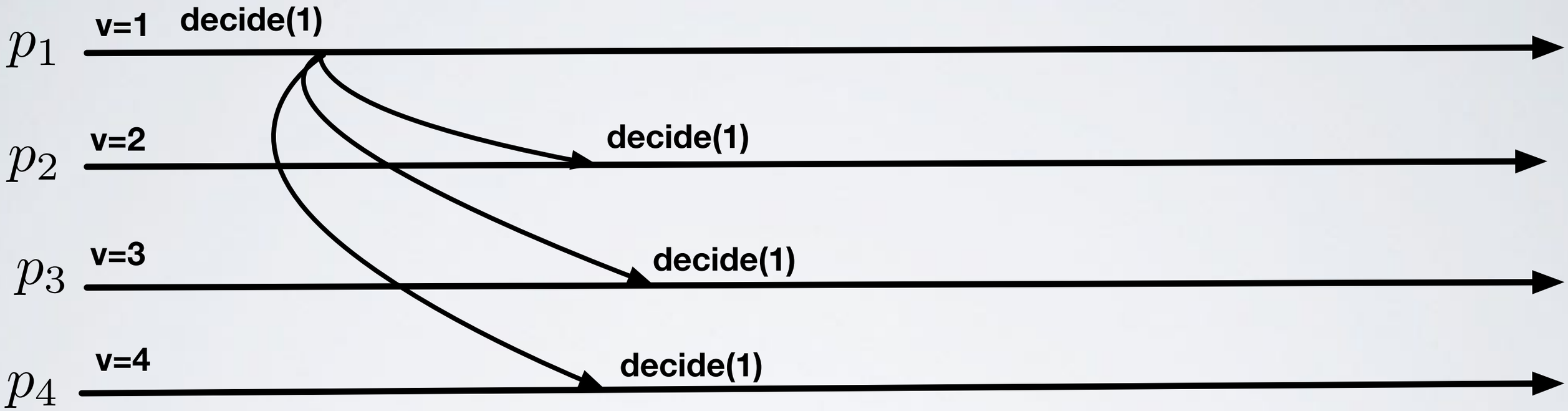


WHICH TECHNIQUE CAN WE USE?

LEADER BASED STRATEGY

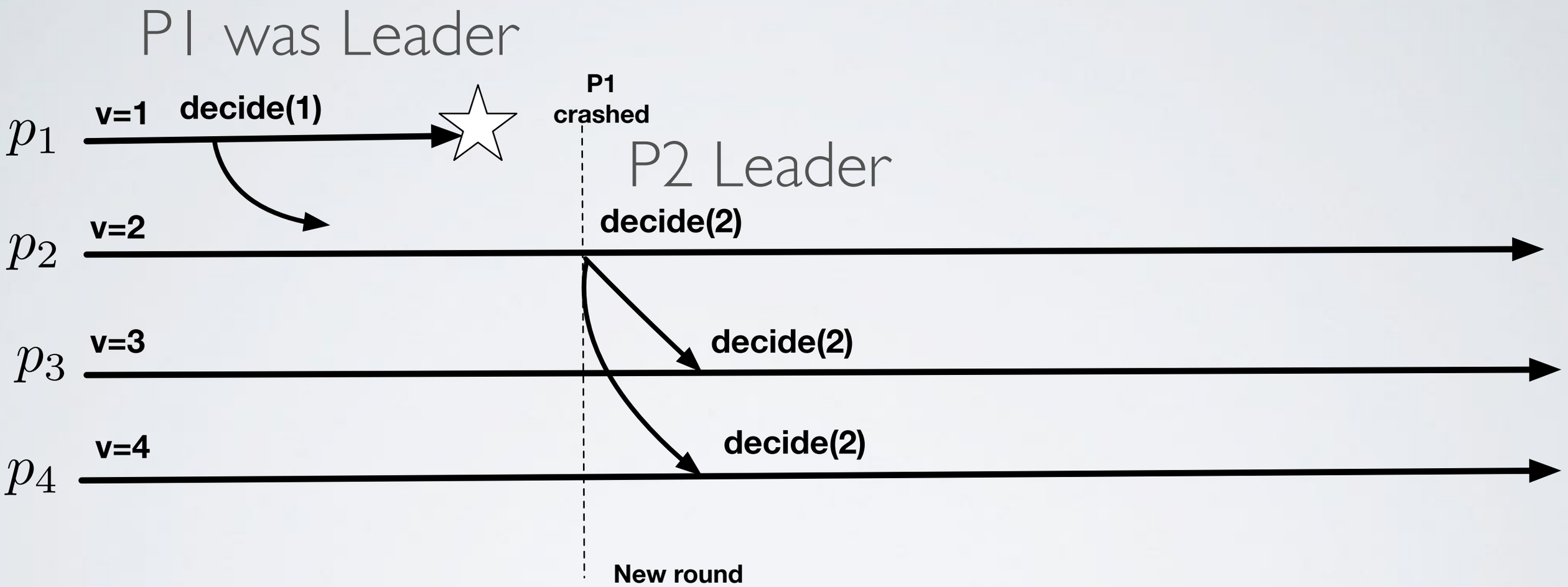
Leader imposes consensus

P1 Leader



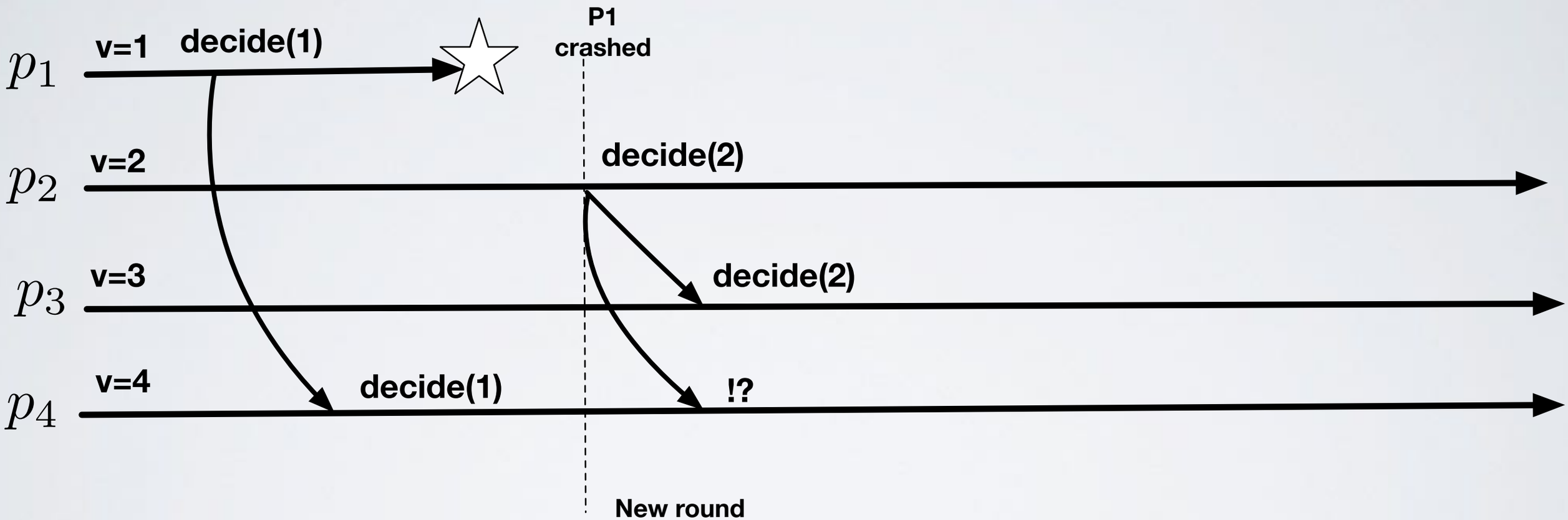
LEADER BASED STRATEGY

Upon crash, new leader new impose



LEADER BASED STRATEGY

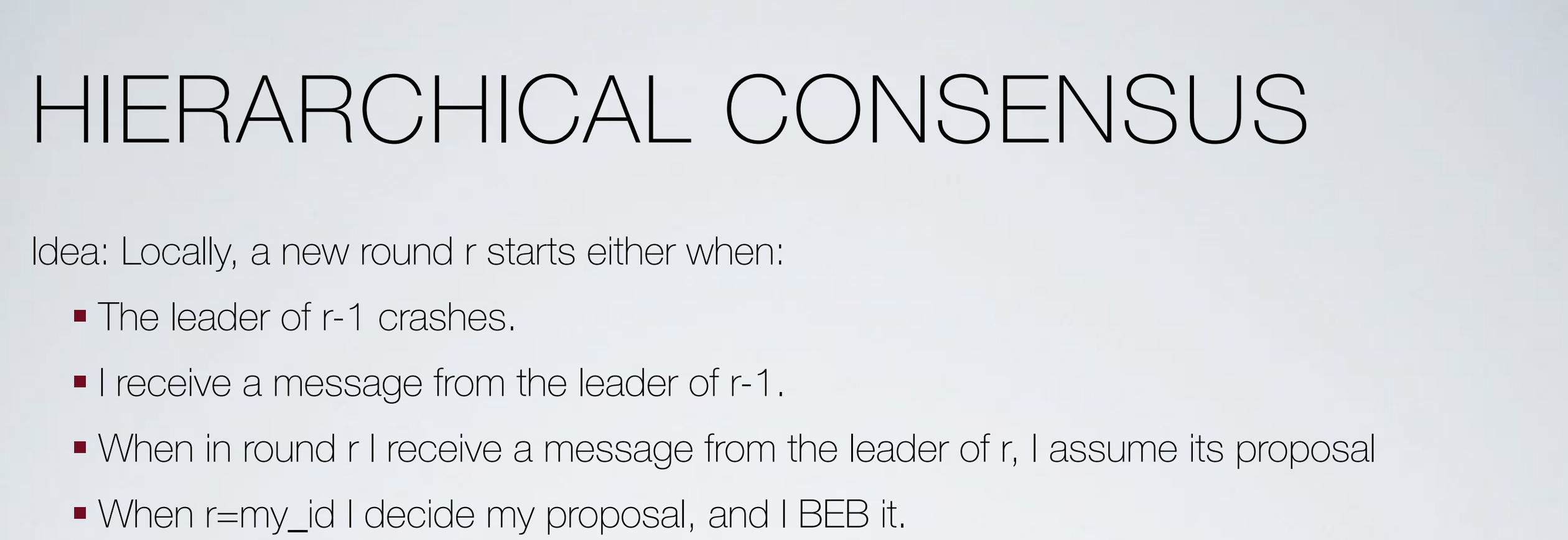
Problem: the leader 1 imposed 1 on p4, then it crashes and p2 imposes 2 on others.



HIERARCHICAL CONSENSUS

Idea: Locally, a new round r starts either when:

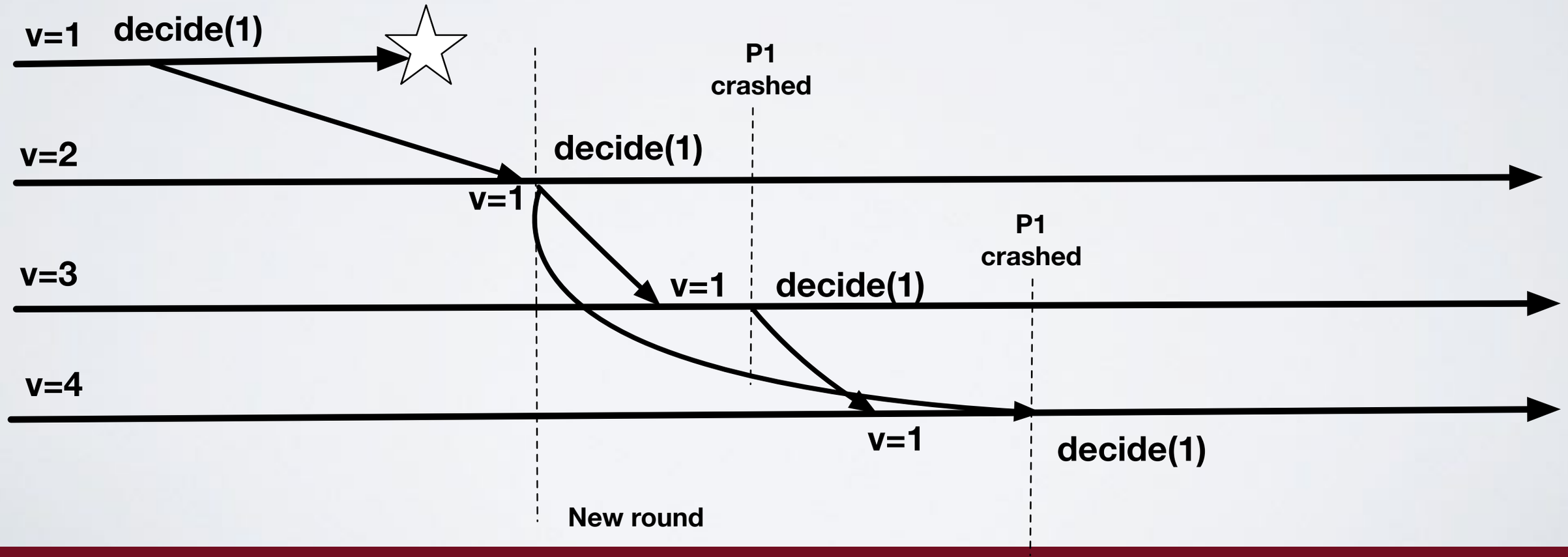
- The leader of r-1 crashes.
- I receive a message from the leader of r-1.
- When in round r I receive a message from the leader of r, I assume its proposal
- When r=my_id I decide my proposal, and I BEB it.



HIERARCHICAL CONSENSUS

Idea: Locally, a new round r starts either when:

- The leader of r-1 crashes.
- I receive a message from the leader of r-1.
- When in round r I receive a message from the leader of r, I assume its proposal
- When r=my_id I decide my proposal, and I BEB it.



HIERARCHICAL CONSENSUS

Algorithm 5.2: Hierarchical Consensus

Implements:
Consensus, **instance** c.

Uses:
BestEffortBroadcast, **instance** *beb*;
PerfectFailureDetector, **instance** $\mathcal{P}$.

upon event $\langle\ c, \mathit{Init}\ \rangle$ **do**
detectedranks := $\emptyset$;
round := 1;
proposal := $\perp$; *proposer* := 0;
delivered := $[\text{FALSE}]^N$;
broadcast := FALSE;

upon event $\langle\ \mathcal{P}, \mathit{Crash} \mid p\ \rangle$ **do**
detectedranks := *detectedranks* $\cup\ \{\mathit{rank}(p)\}$;

upon event $\langle\ c, \mathit{Propose} \mid v\ \rangle$ **such that** *proposal* $= \perp$ **do**
proposal := v;

upon *round* $=$ *rank*(*self*) $\wedge$ *proposal* $\neq \perp \wedge$ *broadcast* $=$ FALSE **do**
broadcast := TRUE;
trigger $\langle\ \mathit{beb}, \mathit{Broadcast} \mid [\text{DECIDED}, \mathit{proposal}]\ \rangle$;
trigger $\langle\ c, \mathit{Decide} \mid \mathit{proposal}\ \rangle$;

upon *round* $\in$ *detectedranks* $\vee$ *delivered*[*round*] $=$ TRUE **do**
round := *round* $+ 1$;

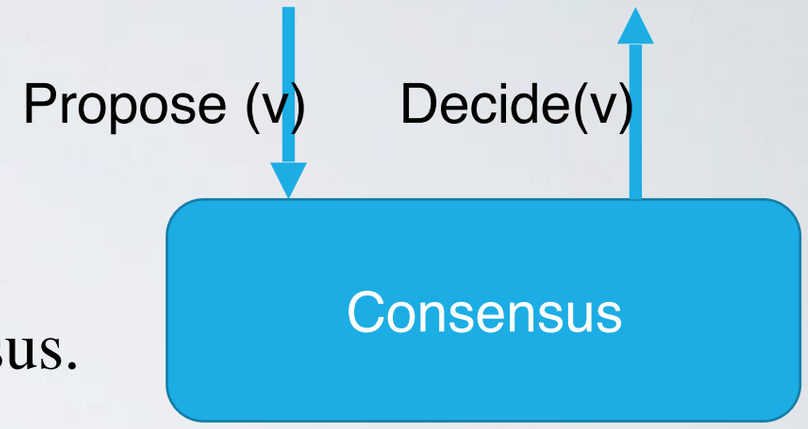
upon event $\langle\ \mathit{beb}, \mathit{Deliver} \mid p, [\text{DECIDED}, v]\ \rangle$ **do**
r := *rank*(p);
if $r <$ *rank*(*self*) $\wedge\ r >$ *proposer* **then**
proposal := v;
proposer := r;
delivered[r] := TRUE;

CORRECTNESS

Module 5.1: Interface and properties of (regular) consensus

Module:

Name: Consensus, **instance** c.



Events:

Request: $\langle\ c, \mathit{Propose} \mid v\ \rangle$: Proposes value v for consensus.

Indication: $\langle\ c, \mathit{Decide} \mid v\ \rangle$: Outputs a decided value v of consensus.

Properties:

C1: *Termination:* Every correct process eventually decides some value.

C2: *Validity:* If a process decides v, then v was proposed by some process.

C3: *Integrity:* No process decides twice.

C4: *Agreement:* No two correct processes decide differently.

HIERARCHICAL CONSENSUS

Correctness

- Integrity: There is only one line in which you decide. The line is protected by a boolean variable Broadcast==False that is set to True inside the handler. Therefore, it can be executed at most once.
- Validity: You can only decide the value of your proposal variable. Such a variable can be set on two lines:
  - In the propose handler -> this means that what proposed by yourself
  - In the BebDelivery handler -> in this case I receive a proposal variable of somonelse, so it was proposed by someone (see previous point).
- Termination: By induction on the process id.
  - Base case: id=1 decides or crashes -> round 1= id=1 thus the handler used to decide is eventually executed or the process crashes.
  - Inductive hypothesis: process with id=k-1 either crashes or terminate
  - Inductive case id=k. By inductive hypothesis all processes with ids in 1..k-1 either terminates or crash. This implies that the handler that increases the round will be true for all rounds in 1…k-1. Therefore, eventually round will be k. At this point, as in the base case, either I decide or I crash.

HIERARCHICAL CONSENSUS

Correctness

- Agreement:
  - Let p_i be the correct process with minimum ID.
  - No process will go in round i+1 without receiving the decided message by p_i (recall p_i is correct).
  - This means that for a process to update its round to i+1 it has to put proposal to the same proposal of p_i.
- Therefore each process with ID greater than p_i decides the same proposal of p_i. Recall, that you will not accept in round i+1 messages from old leaders
- The ID of p_i is the minimum so this set is the set contains all correct processes

HIERARCHICAL CONSENSUS

Performance messages

- O(N^2) messages:
  - We have n rounds, in each round the leader Beb-Broadcast: n messages x n rounds

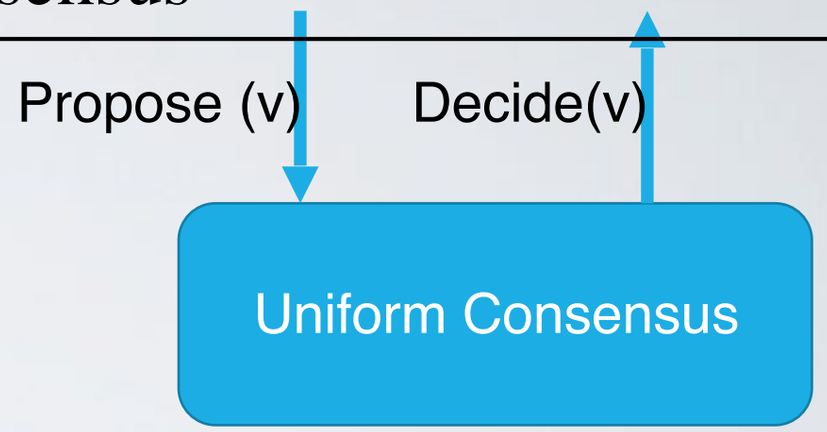
Performances communication steps:

- O(N) fixed, does not depend on **f**.

UNIFORM CONSENSUS SPECIFICATION

Module 5.2: Interface and properties of uniform consensus

Module:



Name: UniformConsensus, **instance** *uc*.

Events:

Request: ⟨ *uc*, *Propose* | v ⟩: Proposes value v for consensus.

Indication: ⟨ *uc*, *Decide* | v ⟩: Outputs a decided value v of consensus.

Properties:

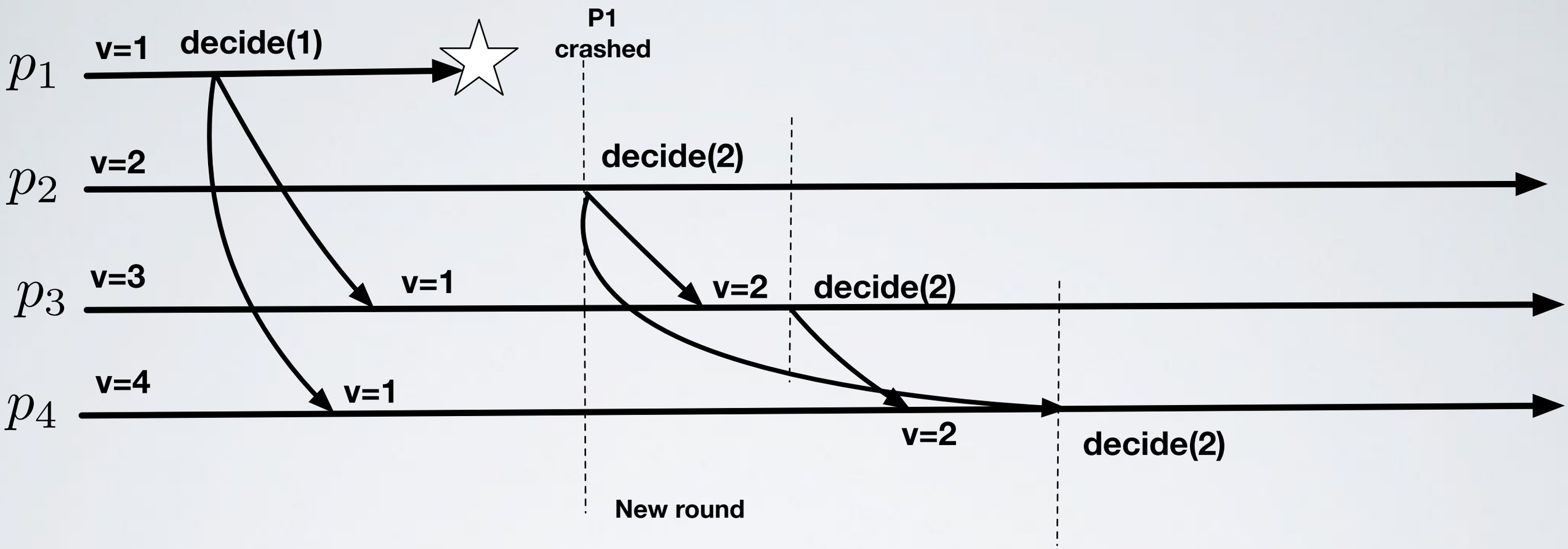
UC1–UC3: Same as properties C1–C3 in (regular) consensus (Module 5.1).

UC4: *Uniform agreement:* No two processes decide differently.

UNIFORM CONSENSUS AND FLOODING CONSENSUS

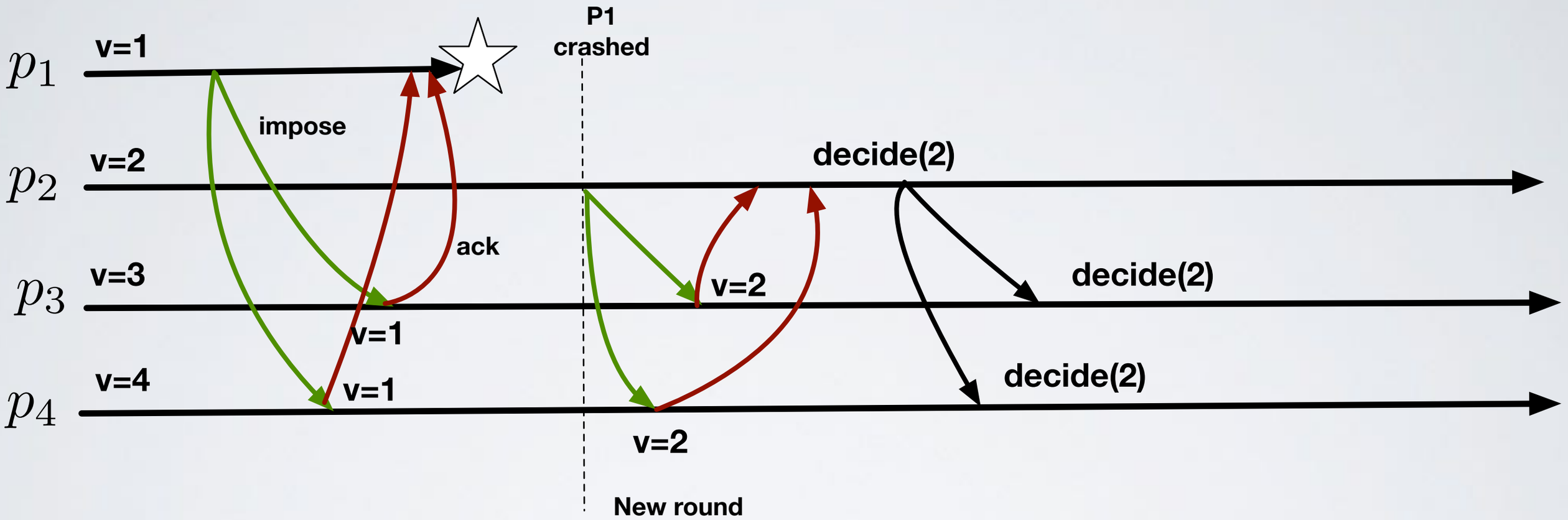
- Does the previous algorithms satisfy the Uniform Consensus specification?

UNIFORM HIERARCHICAL CONSENSUS



The problem is that the leader decides
Before imposing others

UNIFORM HIERARCHICAL CONSENSUS



solution: acks

UNIFORM HIERARCHICAL CONSENSUS

Uses:
PerfectPointToPointLinks, **instance** *pl*;
BestEffortBroadcast, **instance** *beb*;
ReliableBroadcast, **instance** *rb*;
PerfectFailureDetector, **instance** $\mathcal{P}$.

upon event $\langle$ *uc*, *Init* $\rangle$ **do**
detectedranks := $\emptyset$;
ackranks := $\emptyset$;
round := 1;
proposal := $\perp$; *decision* := $\perp$;
proposed := $[\perp]^N$;

upon event $\langle$ $\mathcal{P}$, *Crash* | p $\rangle$ **do**
detectedranks := *detectedranks* $\cup$ $\{rank(p)\}$;

upon event $\langle$ *uc*, *Propose* | v $\rangle$ **such that** *proposal* $= \perp$ **do**
proposal := v;

upon *round* $=$ *rank*(*self*) $\wedge$ *proposal* $\neq \perp \wedge$ *decision* $= \perp$ **do**
trigger $\langle$ *beb*, *Broadcast* | [PROPOSAL, *proposal*] $\rangle$;

upon event $\langle$ *beb*, *Deliver* | p, [PROPOSAL, v] $\rangle$ **do**
proposed[*rank*(p)] := v;
if *rank*(p) $\geq$ *round* **then**
trigger $\langle$ *pl*, *Send* | p, [ACK] $\rangle$;

upon *round* $\in$ *detectedranks* **do**
if *proposed*[*round*] $\neq \perp$ **then**
proposal := *proposed*[*round*];
round := *round* + 1;

upon event $\langle$ *pl*, *Deliver* | q, [ACK] $\rangle$ **do**
ackranks := *ackranks* $\cup$ $\{rank(q)\}$;

upon *detectedranks* $\cup$ *ackranks* $= \{1, \ldots, N\}$ **do**
trigger $\langle$ *rb*, *Broadcast* | [DECIDED, *proposal*] $\rangle$;

upon event $\langle$ *rb*, *Deliver* | p, [DECIDED, v] $\rangle$ **such that** *decision* $= \perp$ **do**
decision := v;
trigger $\langle$ *uc*, *Decide* | *decision* $\rangle$;

UNIFORM HIERARCHICAL CONSENSUS

Correctness

- Integrity and Validity: The same proof of the non-uniform hierarchical
- Termination: By induction on the IDs, similar to the previous algorithm:
  - Base Case ID=1:, for the BebCast and the perfect failure detector if p1 is correct it eventually executes the reliableBroadcast of Decided and delivers it terminating.
  - Inductive Hypothesis, ID=k-1: terminates or crashes
  - Inductive step ID=k: If a process with id less or equal to k-1 it is correct then it terminates by a ReliableDelivered of a Decided messages, for the property of the reliable broadcast if I am correct I will also receive a Decided message terminating. If all processes with ID less or equal to k-1 crash, then by the property of P I go to round r=k, since my ID is k I crash or terminate with the same prof of the base case.

UNIFORM HIERARCHICAL CONSENSUS

Correctness

- Uniform agreement: Let pi be the first process that decides, then it received the acks from all correct process (each process has proposed[i]=v, where v is the value decided by pi). We have two cases:
  - Pi is correct -> the other processes receive the decided message from Pi, they have to decide the same value
  - Pi crashes -> all the processes that go to round i+1 set proposal=v, the processes that decide in round I will decide by receiving the decided message of Pi that also contains v.

UNIFORM HIERARCHICAL CONSENSUS

Message complexity

- (1Beb + 1 Rb)*(f+1)

Step complexity

- 2 delays for each leader failed, and 3 for each leader that succeeds in sending and delivering one decided = O(f).

EXERCISE

Algorithm 5.2: Hierarchical Consensus

Implements:
Consensus, **instance** c.

Uses:
BestEffortBroadcast, **instance** *beb*;
PerfectFailureDetector, **instance** $\mathcal{P}$.

upon event $\langle$ c, *Init* $\rangle$ **do**
detectedranks := $\emptyset$;
round := 1;
proposal := $\perp$; *proposer* := 0;
delivered := $[\text{FALSE}]^N$;
broadcast := FALSE;

upon event $\langle$ $\mathcal{P}$, *Crash* | p $\rangle$ **do**
detectedranks := *detectedranks* $\cup$ $\{rank(p)\}$;

upon event $\langle$ c, *Propose* | v $\rangle$ **such that** *proposal* $= \perp$ **do**
proposal := v;

upon *round* $=$ *rank*(*self*) $\wedge$ *proposal* $\neq \perp \wedge$ *broadcast* $=$ FALSE **do**
broadcast := TRUE;
trigger $\langle$ *beb*, *Broadcast* | [DECIDED, *proposal*] $\rangle$;
trigger $\langle$ c, *Decide* | *proposal* $\rangle$;

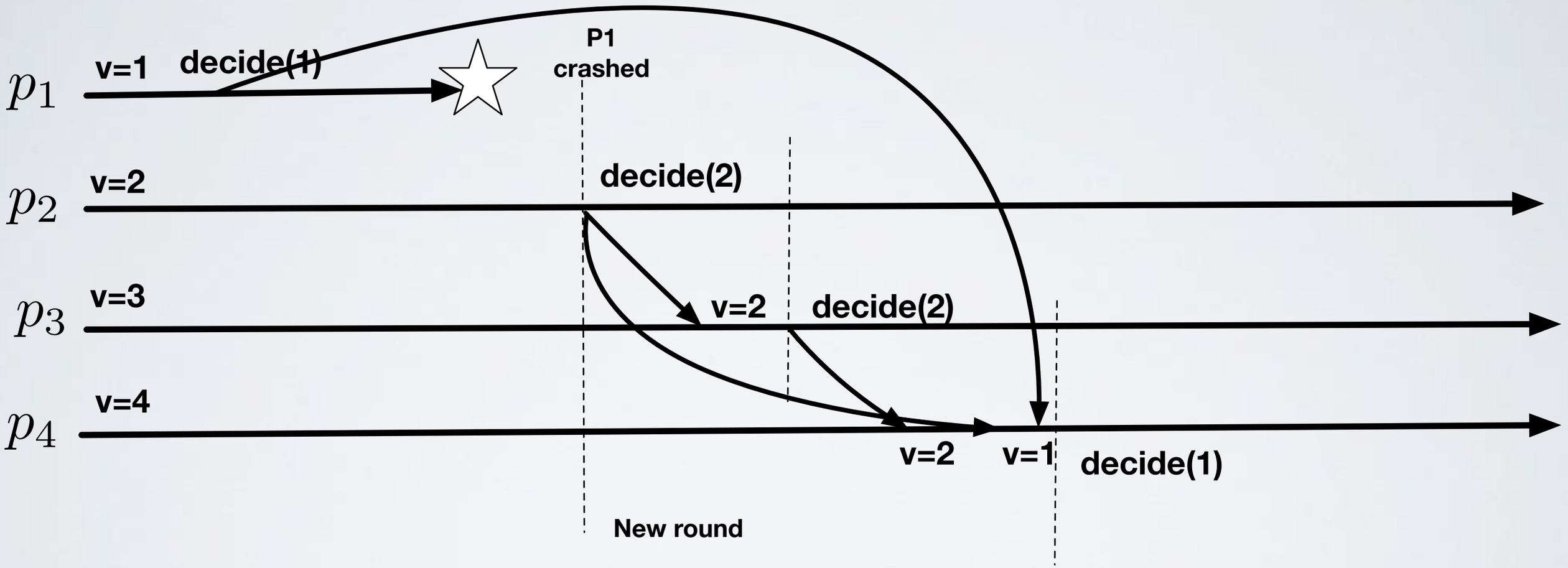
upon *round* $\in$ *detectedranks* $\vee$ *delivered*[*round*] $=$ TRUE **do**
round := *round* + 1;

upon event $\langle$ *beb*, *Deliver* | p,[DECIDED, v] $\rangle$ **do**
r := *rank*(p);
if $r <$ *rank*(*self*) $\wedge$ $r >$ *proposer* **then**
proposal := v;
proposer := r;
delivered[r] := TRUE;

What happens If we Remove this?

SOLUTION

What happens If we Remove this?



EXERCISE

Uses:
PerfectPointToPointLinks, **instance** *pl*;
BestEffortBroadcast, **instance** *beb*;
ReliableBroadcast, **instance** *rb*;
PerfectFailureDetector, **instance** $\mathcal{P}$.

upon event $\langle$ *uc*, *Init* $\rangle$ **do**
detectedranks := $\emptyset$;
ackranks := $\emptyset$;
round := 1;
proposal := $\perp$; *decision* := $\perp$;
proposed := $[\perp]^N$;

upon event $\langle$ $\mathcal{P}$, *Crash* | p $\rangle$ **do**
detectedranks := *detectedranks* $\cup$ $\{rank(p)\}$;

upon event $\langle$ *uc*, *Propose* | v $\rangle$ **such that** *proposal* $= \perp$ **do**
proposal := v;

upon *round* $=$ *rank*(*self*) $\wedge$ *proposal* $\neq \perp \wedge$ *decision* $= \perp$ **do**
trigger $\langle$ *beb*, *Broadcast* | [PROPOSAL, *proposal*] $\rangle$;

upon event $\langle$ *beb*, *Deliver* | p, [PROPOSAL, v] $\rangle$ **do**
proposed[*rank*(p)] := v;
if *rank*(p) $\geq$ *round* **then**
trigger $\langle$ *pl*, *Send* | p, [ACK] $\rangle$;

upon *round* $\in$ *detectedranks* **do**
if *proposed*[*round*] $\neq \perp$ **then**
proposal := *proposed*[*round*];
round := *round* + 1;

upon event $\langle$ *pl*, *Deliver* | q, [ACK] $\rangle$ **do**
ackranks := *ackranks* $\cup$ $\{rank(q)\}$;

upon *detectedranks* $\cup$ *ackranks* $= \{1, \ldots, N\}$ **do**
trigger $\langle$ *rb*, *Broadcast* | [DECIDED, *proposal*] $\rangle$;

upon event $\langle$ *rb*, *Deliver* | p, [DECIDED, v] $\rangle$ **such that** *decision* $= \perp$ **do**
decision := v;
trigger $\langle$ *uc*, *Decide* | *decision* $\rangle$;

What happens if we use Beb instead of Rb?

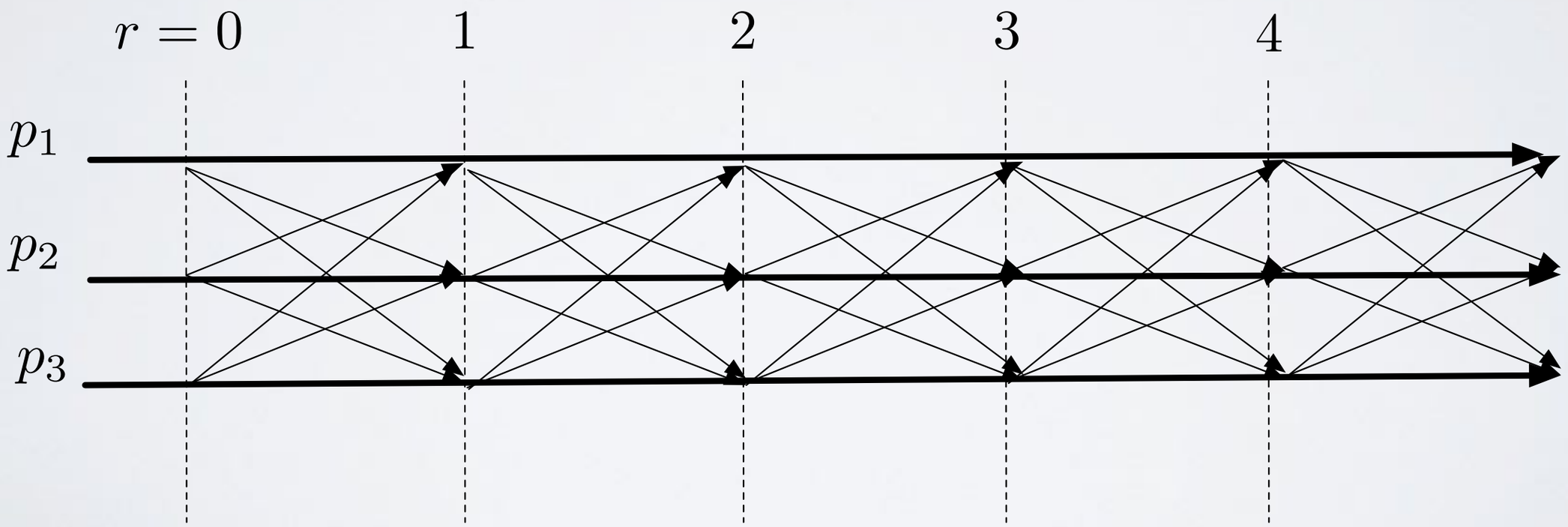
CONSENSUS:
IN SYNCHRONOUS SYSTEMS,
LOWER BOUND ON TIME

Remember the equivalence between round-based model and synchronous system (Failure-Detector Lecture).

NB: this round is different from the concept of round used in the previous algorithm.

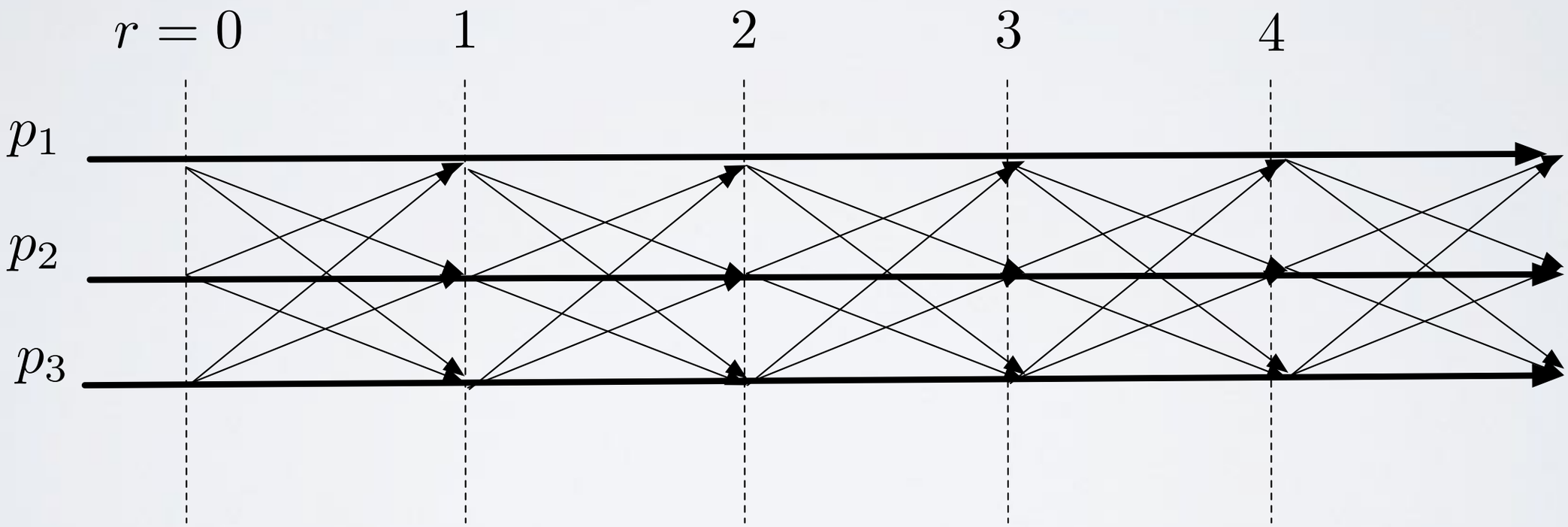
Synchrony assumptions are abstracted by assuming:

- Processes switch rounds exactly at the same time.
- If I a correct process send a message to a set of processes at the beginning of round r, the messages will reach all correct processes in the set by the end of round r.



FULL-INFORMATION PROTOCOL

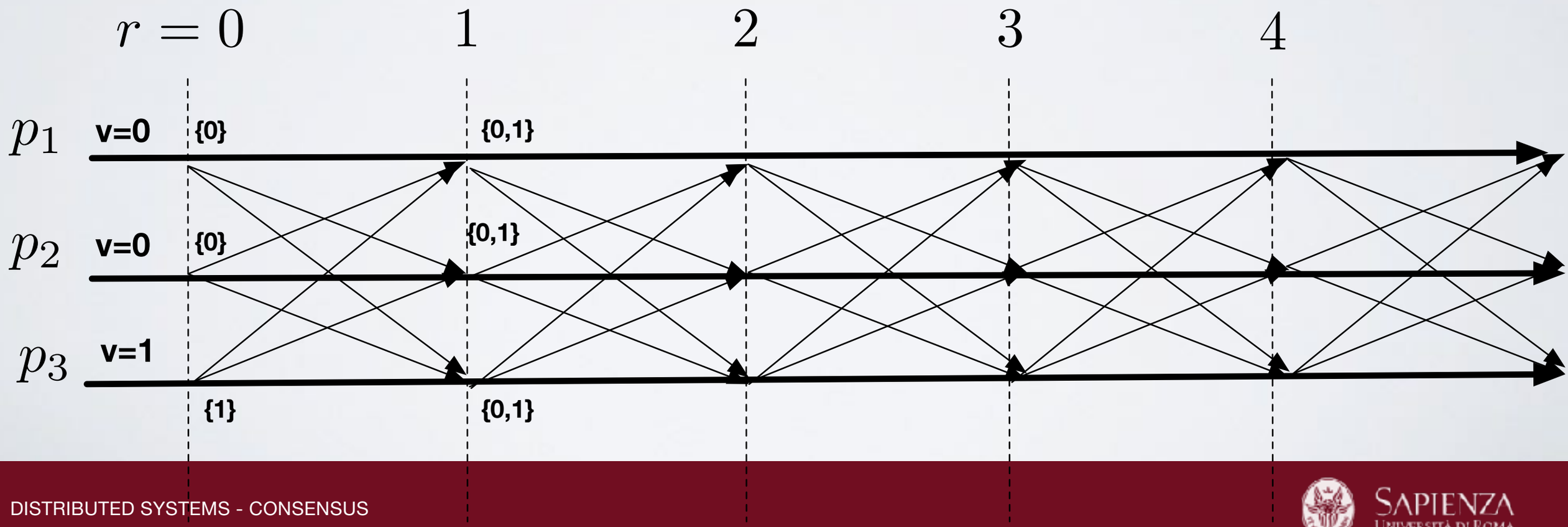
In a Full-Information Protocol (FIP) at each round, each process sends its entire states to others (broadcast).
Each synchronous protocol (or algorithm) can be expressed as a FIP. Intuition: With FIP each process collect all possible information on the system, it builds a view of the system and then decides locally.



FULL-INFORMATION PROTOCOL

Consensus FIP - ASSUME you cannot used processes ID in the algorithm:

- Broadcast my set of proposed values (initially, just my value) to all at the beginning of round r
- Collect all messages at the end of round r and updates proposed value as union of the received message.
- At round r=k take the **maximum** value in the proposed value set and decides.



FULL-INFORMATION PROTOCOL

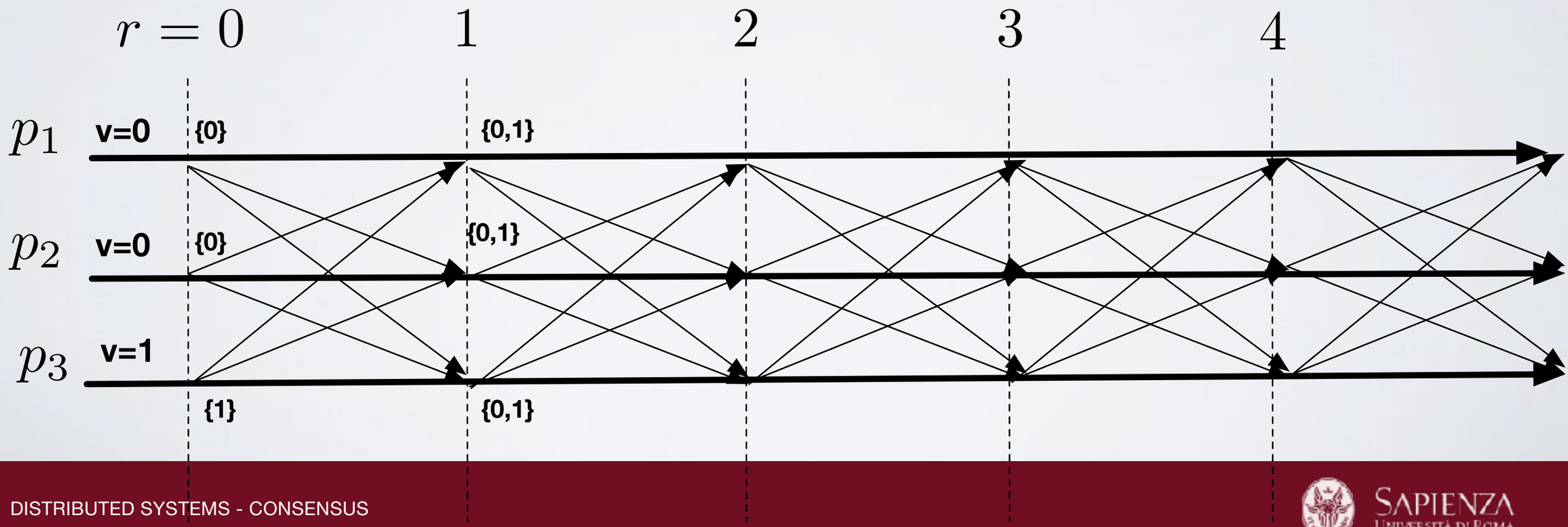
What is the good value for k in a system with f failures

Consensus FIP:

- Broadcast my set of proposed values (initially, just my value) to all at the beginning of round r

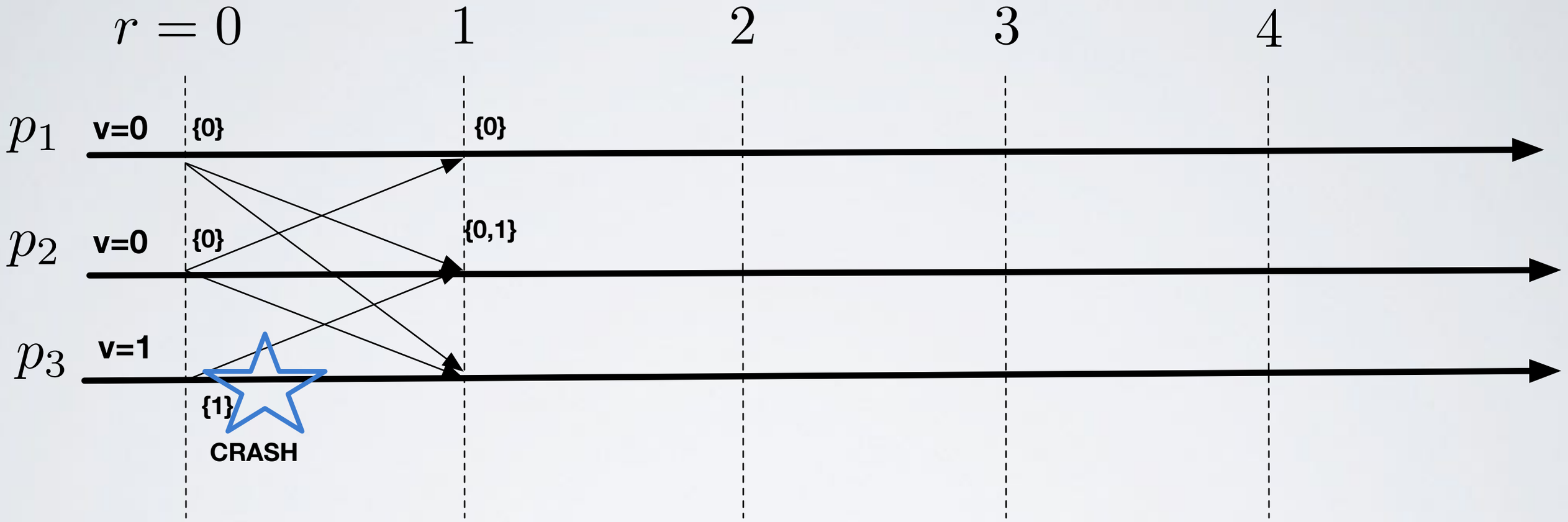
To solve uniform consensus?

- Collect all messages at the end of round r and updates proposed value as union of the received message.
- At round r=k take the **maximum** value in the proposed value set and decides.



FULL-INFORMATION PROTOCOL

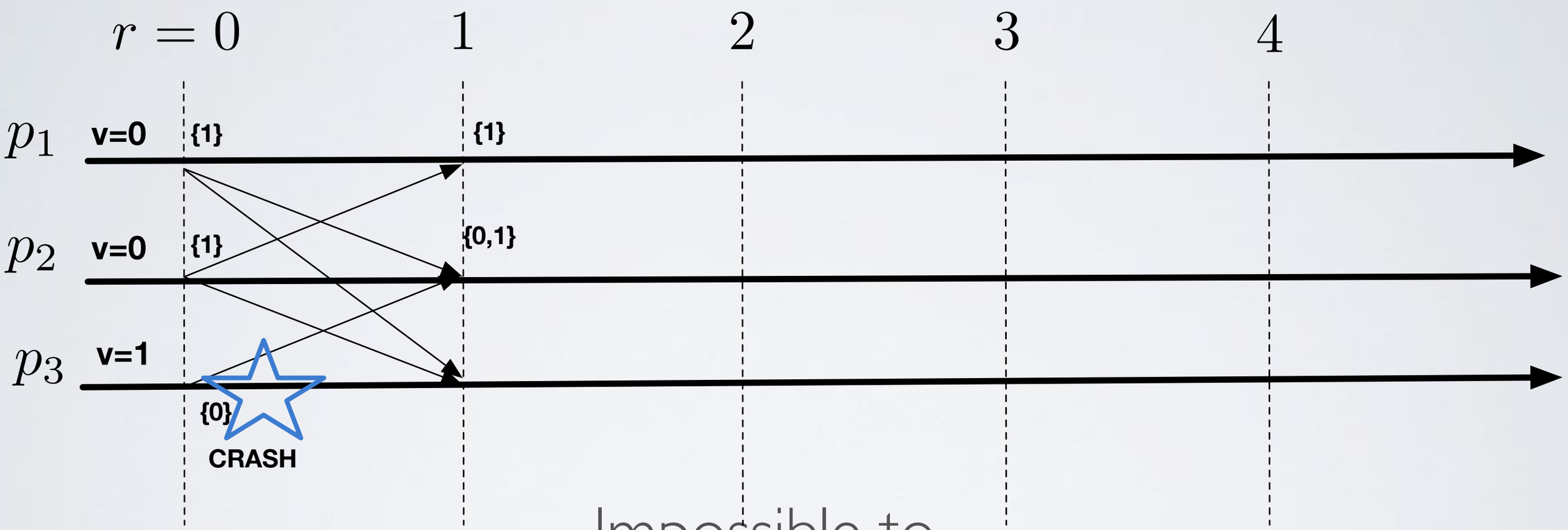
What happens with 1 failure? Note that using min instead of max does not save you



Impossible to
Decide at the end of round 1. P1 decides 0
P2 decides 1.

FULL-INFORMATION PROTOCOL

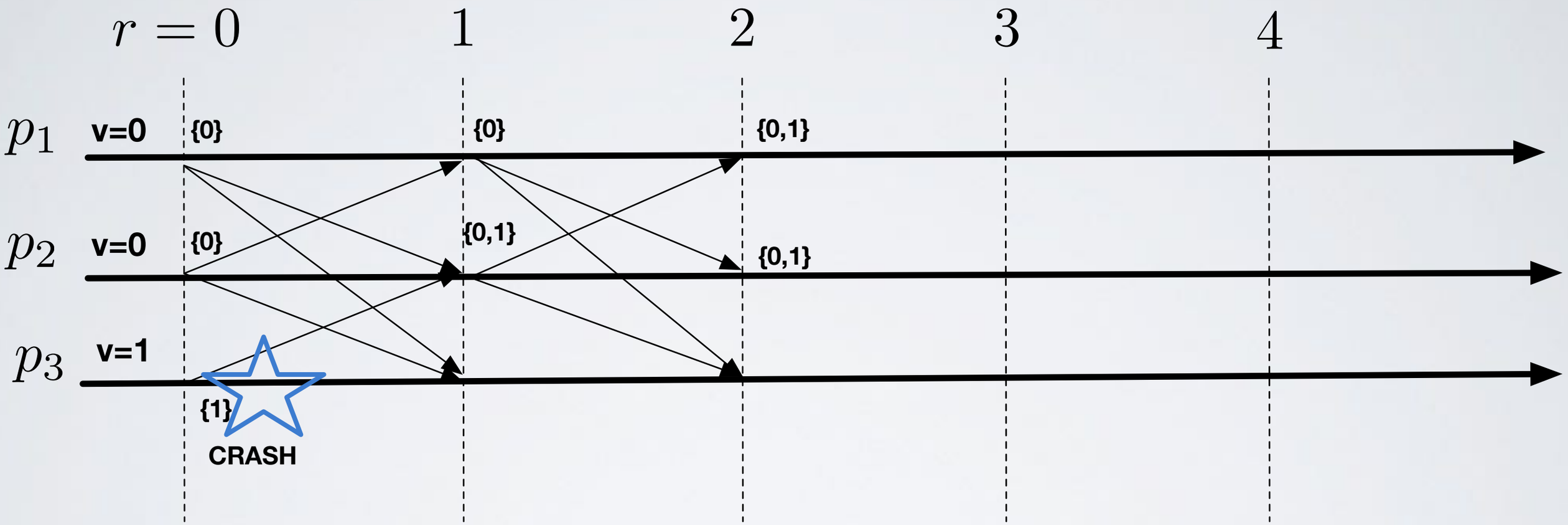
What happens with 1 failure? Note that using **min instead of max does not save you**



Impossible to
Decide at the end of round 1. P1 decides 1
P2 decides 0.

FULL-INFORMATION PROTOCOL

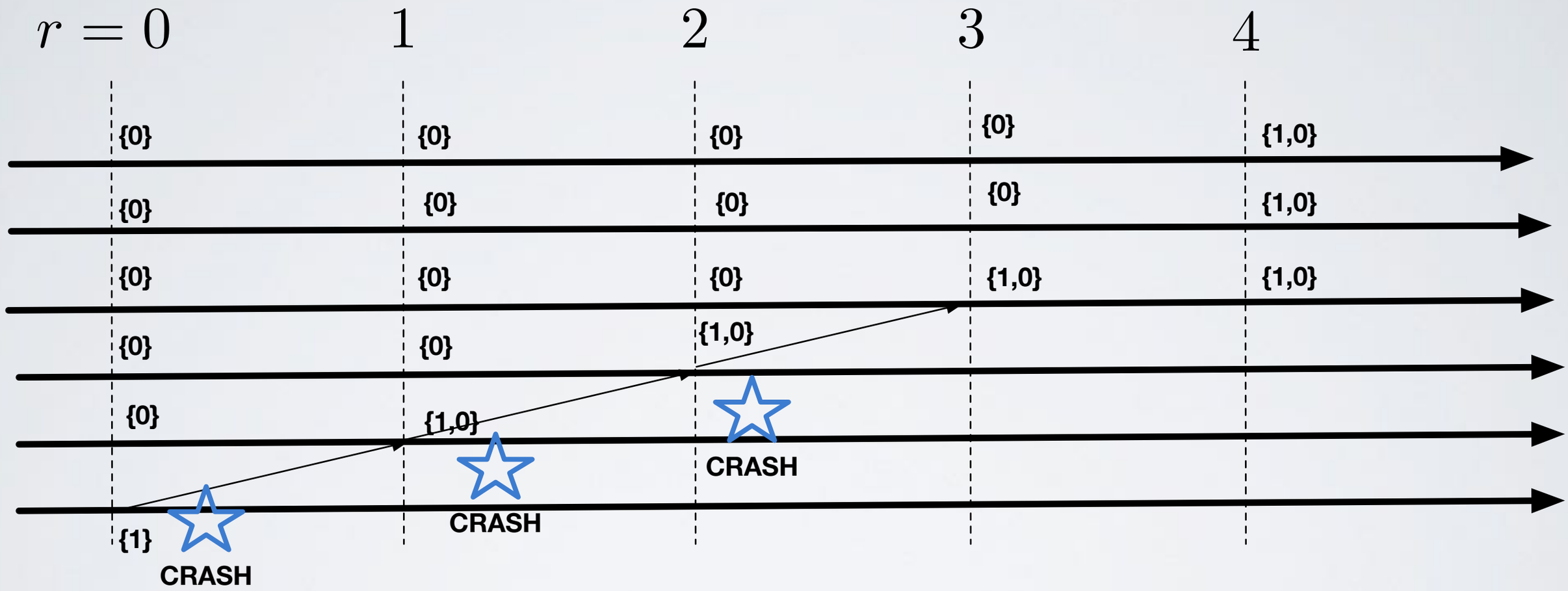
But if f=1 at the end of round 2, **I know that everyone alive has my same set**



At the end of round 2 I can decide

FULL-INFORMATION PROTOCOL

So for a generic **f** I need **f+1** rounds….



F=3 round k=f+1

FULL-INFORMATION PROTOCOL

Theorem. Given a synchronous system with f crash failures and n processes. There exists no algorithm that terminates in less than f+1 rounds, and so there exists no algorithm that has a step complexity of f or less.