EECS 591
Distributed Systems

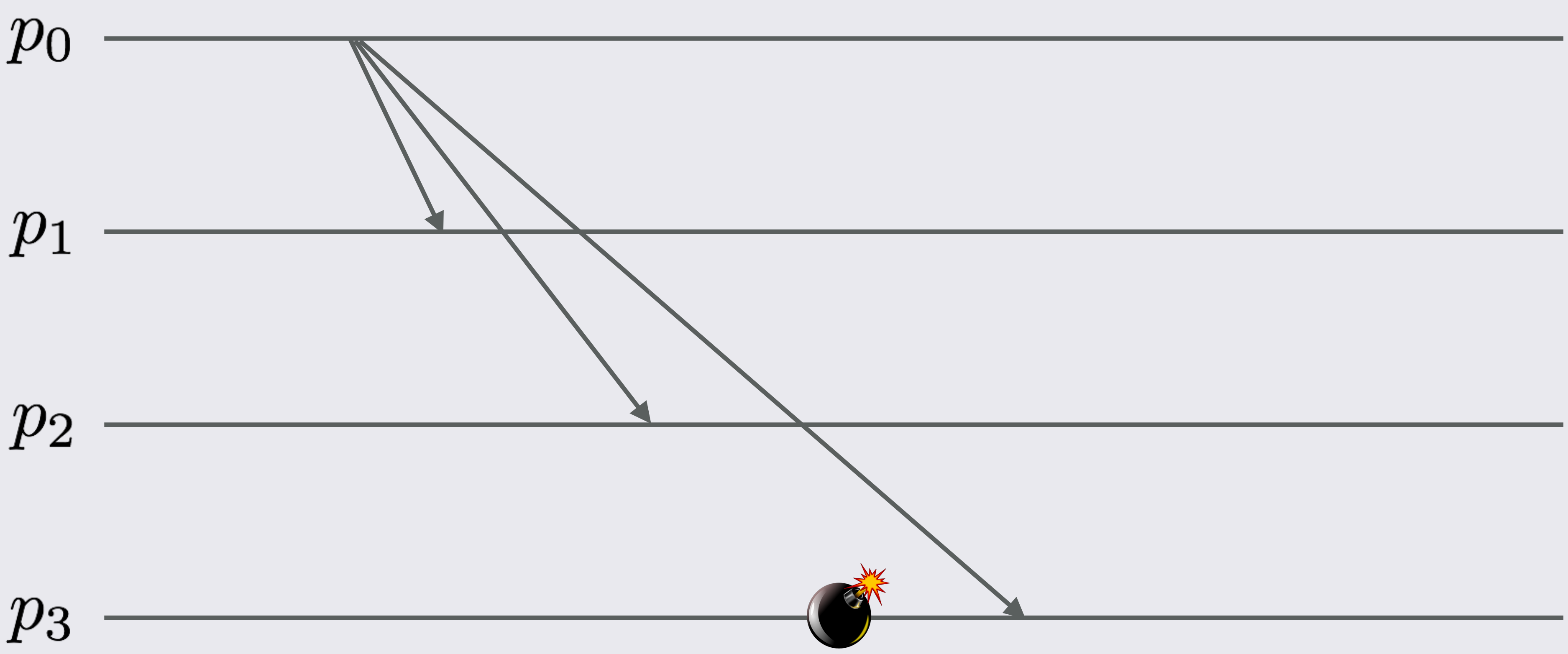
Manos Kapritsos
Fall 2021

Slides by: Lorenzo Alvisi

Consensus and Reliable Broadcast

BROADCAST

If a process sends a message m, then every process eventually delivers m



How can we adapt the spec for an environment where processes may fail?

Reliable broadcast

Validity If the sender is correct and broadcasts a message m, then all correct processes eventually deliver m

Agreement If a correct process delivers a message m, then all correct processes eventually deliver m

Integrity Every correct process delivers at most one message, and if it delivers $m \neq SF$, then some process must have broadcast m

Terminating Reliable broadcast

Validity If the sender is correct and broadcasts a message m, then all correct processes eventually deliver m

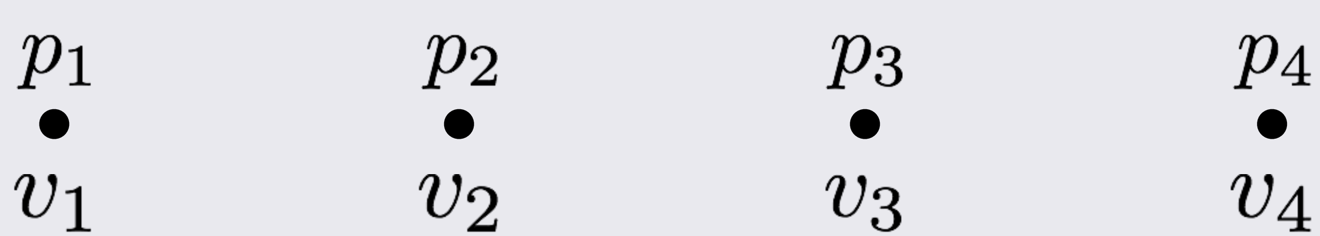
Agreement If a correct process delivers a message m, then all correct processes eventually deliver m

Integrity Every correct process delivers at most one message, and if it delivers $m \neq SF$, then some process must have broadcast m

Termination Every correct process eventually delivers some message

CONSENSUS

Every process has a value v_i to propose. After running a consensus algorithm, all processes should deliver the same value.



Consensus

Validity	If all processes that propose a value propose v, then all correct processes eventually decide v
Agreement	If a correct process decides v, then all correct processes eventually decide v
Integrity	Every correct process decides at most one value, and if it decides v, then some process must have proposed v
Termination	Every correct process eventually decides some value

PROPERTIES OF **send(m)** AND **receive(m)**

Benign failures:

Validity If p sends m to q, and p, q and the link between them are correct, then q eventually receives m

Uniform* integrity For every message m, q receives m at most once from p, and only if p sent m to q

* A property is called uniform if it applies to both correct and faulty processes

Model

- **Synchronous** message passing
  - Execution is a sequence of rounds
  - In each round every process takes a step
    - sends messages to neighbors
    - receives messages send in that round
    - changes its state
- Network is fully connected
- **No communication failures**

A SIMPLE CONSENSUS ALGORITHM

Process p_i:

Initially $V = \{v_i\}$

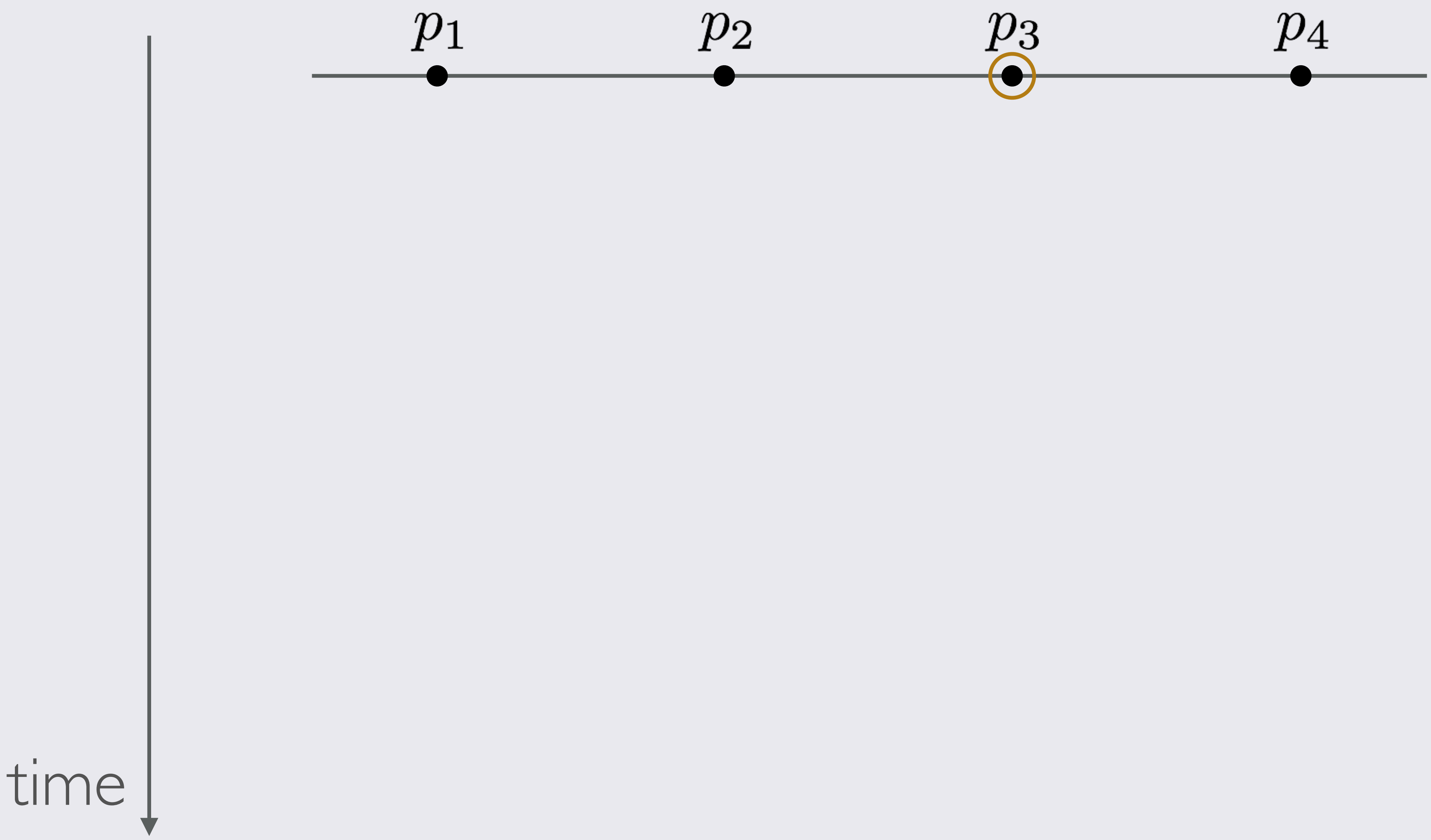
To execute ***propose(*** v_i ***)***:

1. Send $\{v_i\}$ to all

decide( ) occurs as follows:

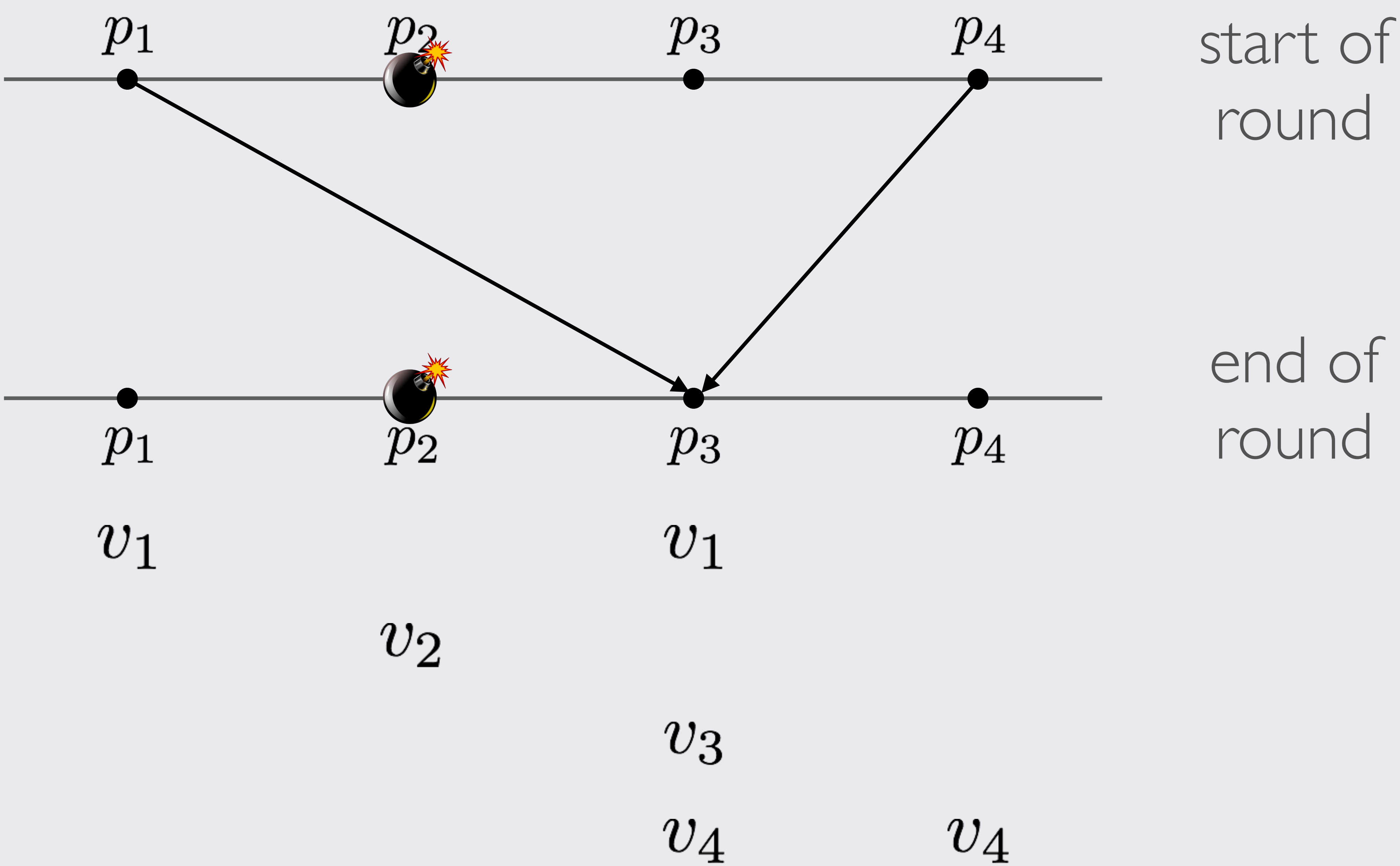
2. for all $j, 0 \leq j \leq n+1, j \neq i$, do
3.   receive S_j from p_j
4. $V := V \cup S_j$
5. decide min(V)

An execution



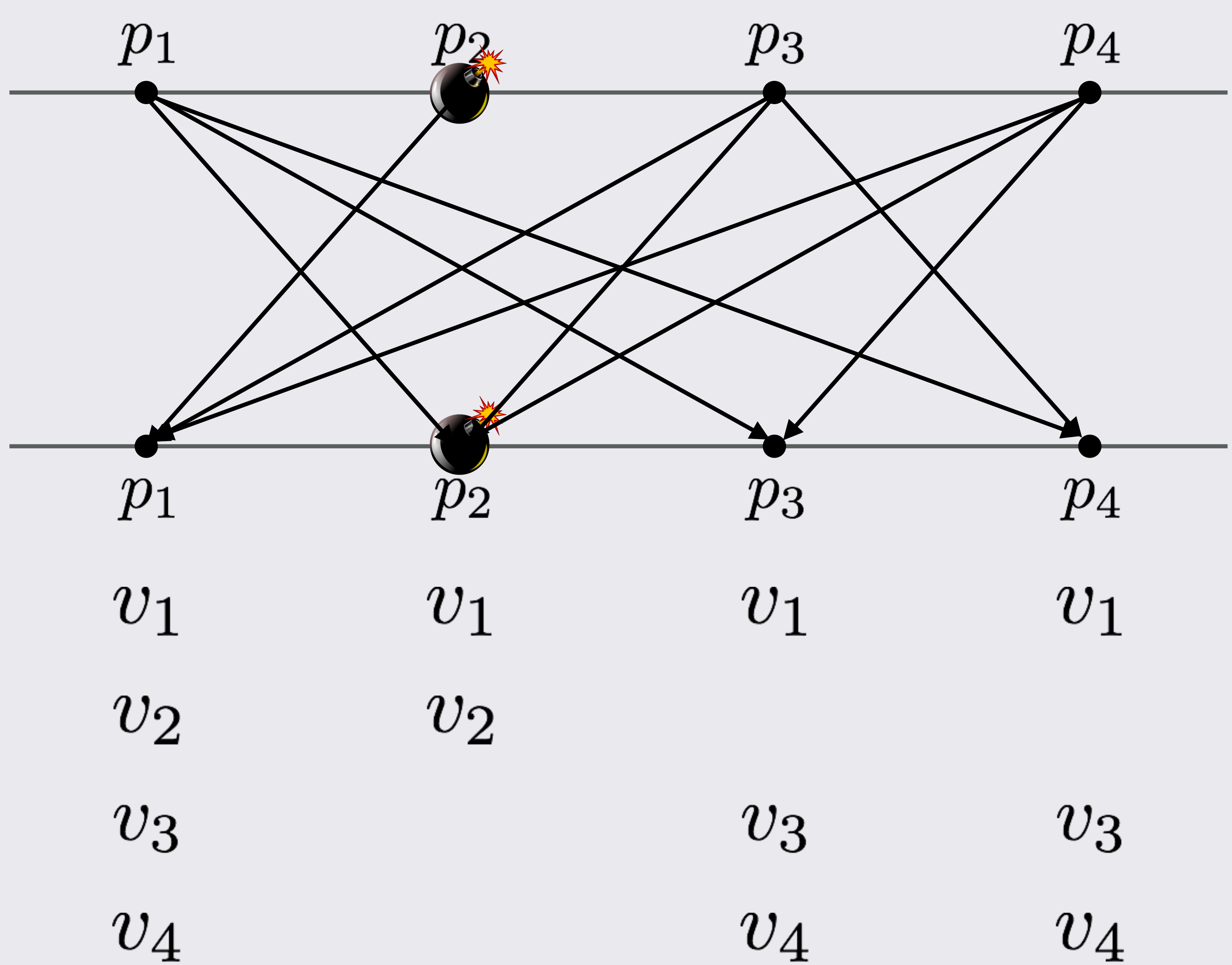
AN EXECUTION

What should p_3 decide at the end of the round?



AN EXECUTION

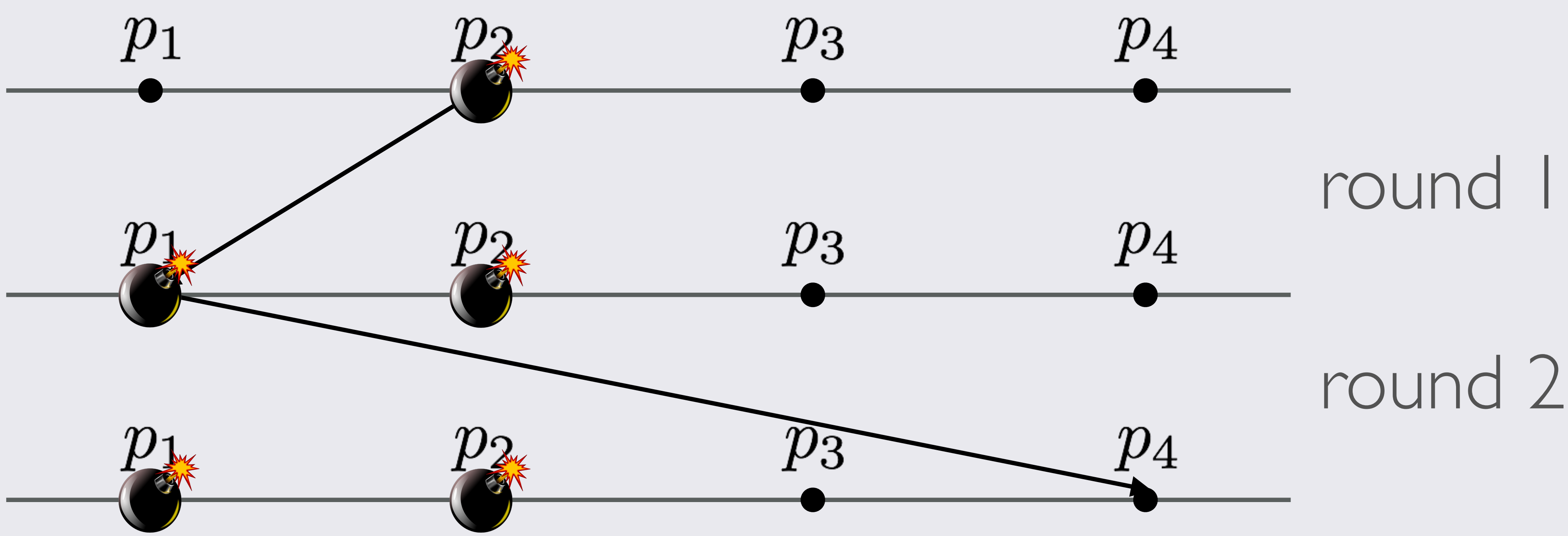
What should p_3 decide at the end of the round?



ECHOING VALUES

A process that receives a proposal in round 1, relays it to others during round 2

Suppose p_3 hasn't heard from p_2 at the end of round 2. Can p_3 decide?



WHAT IS GOING ON

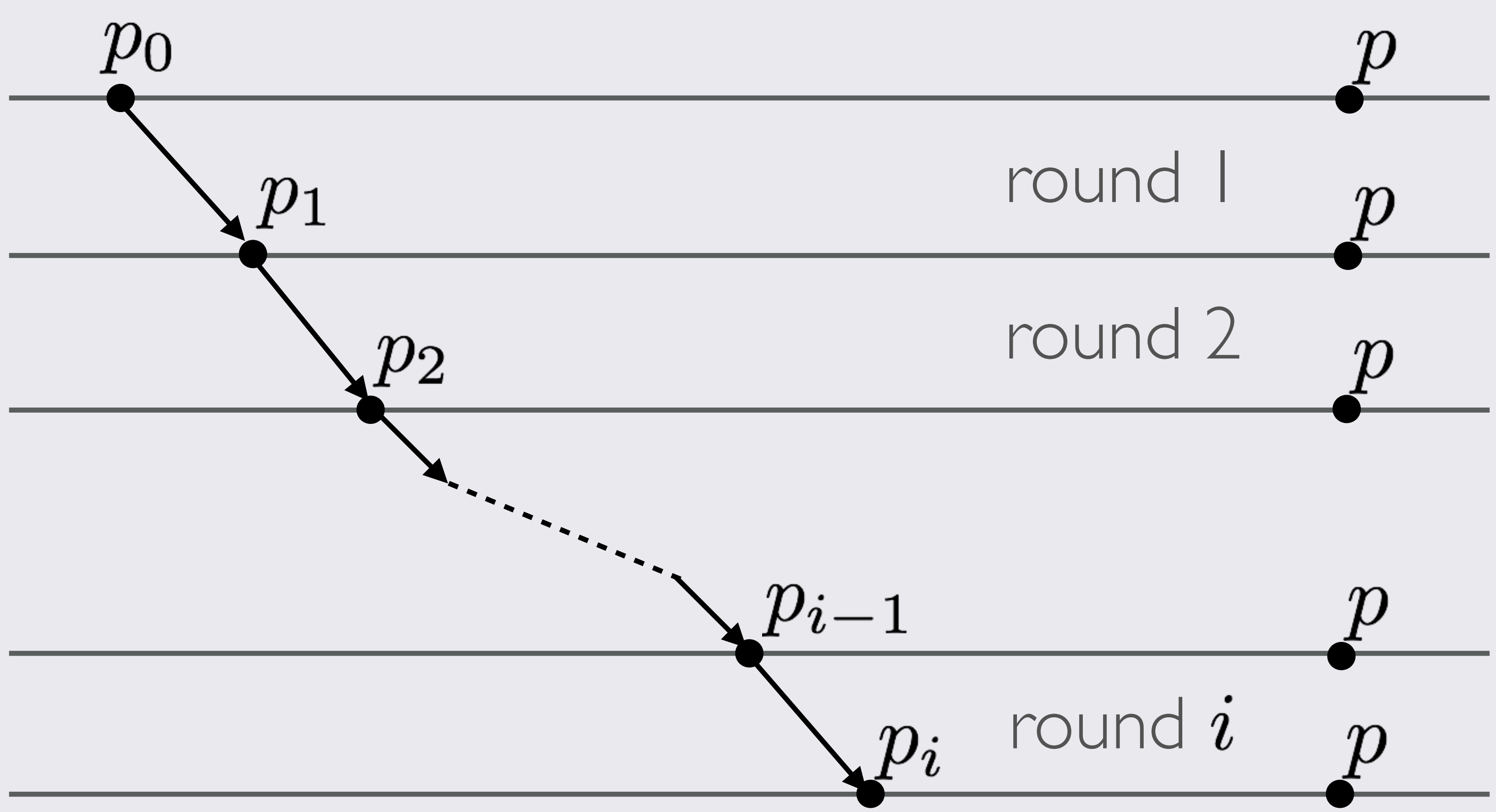
A correct process p has not received all proposals by the end of round i. Can p decide?

Another process may have received the missing proposal at the end of round i and be ready to relay it in round $i+1$

Dangerous chains

Dangerous chain

The last process in the chain is correct, all others faulty



LIVING DANGEROUSLY

How many rounds can a dangerous chain span?

- f faulty processes
- At most $f+1$ nodes in the chain
- Spans at most f rounds

It is safe to decide by the end of round $f+1$!

Administrivia

Problem set #1 due September 27

See Piazza post for a list of deadlines

Preparing for the "research" part of the course

Look at the papers listed on the course webpage

You each pick one to present
(email me 4 preferences by Monday night)

I'll assign you to a paper and post the schedule

- ~25-30 minutes presentation
- Send me the slides by Nov 2
  - (unless you are presenting earlier)

THE RESEARCH PROJECT

Sample topics:

Concrete

- Combining Fast Paxos and Flexible Paxos to reduce latency in a geo-replicated storage system
- Proving the correctness of BitCoin

Motivational

- Why the world needs real-time proofs of distributed systems
- Supporting the equivalent instruction hypothesis
- All the things you can do with Flexible Paxos

Survey

- Applying Byzantine Fault Tolerance to blockchains: theory and practice

The algorithm

Process p_i:

Initially $V = \{v_i\}$

To execute ***propose(*** v_i ***)***:

round $k, 1 \leq k \leq f+1$

1. Send $\{v \in V : p_i$ has not already sent $v\}$ to all
2. for all $j, 0 \leq j \leq n+1, j \neq i$, do
3. receive S_j from p_j
4. $V := V \cup S_j$

---

decide( ) occurs as follows:

5. if $k = f+1$
6. decide min(V)

PROVING TERMINATION

To execute ***propose(v_i)***:

round $k, 1 \leq k \leq f+1$

1. Send $\{v \in V : p_i$ has not already sent $v\}$ to all
2. for all $j, 0 \leq j \leq n+1, j \neq i$, do
3. receive S_j from p_j
4. $V := V \cup S_j$

decide( ) occurs as follows:

5. if $k = f+1$
6. decide min(V)

Every correct process

- Reaches round $f+1$
- Decides min(V), which is well defined

Proving Integrity

To execute ***propose(v_i)***:

round $k, 1 \leq k \leq f+1$

1. Send $\{v \in V: p_i$ has not already sent $v\}$ to all
2. for all $j, 0 \leq j \leq n+1, j \neq i$, do
3. receive S_j from p_j
4. $V := V \cup S_j$

decide( ) occurs as follows:

5. if $k = f+1$
6. decide min(V)

At most one value:

One decide() and min(V) is unique

Only if it was proposed:

- To be decided, must be in V in round $f+1$
- If value $= v_i$, then it is proposed in round 1
- else, suppose it was received in round k

By induction:

- $k = 1$
  - By Uniform Integrity of underlying send and receive, it must have been sent in round 1
  - By the protocol, and because we only have benign failures, it must have been proposed
- Induction hypothesis: all values received up to round $k = j$ have been proposed
- $k = j+1$
  - Sent in round $j+1$ (Uniform Integrity of send and synchronous model)
  - Must have been part of V of sender at end of round j
  - By the protocol, must have been received by sender by the end of round j
  - By induction hypothesis, must have been proposed

PROVING VALIDITY

To execute ***propose(v_i)***:

round $k, 1 \leq k \leq f + 1$

1. Send $\{v \in V$: p_i has not already sent $v\}$ to all
2. for all $j, 0 \leq j \leq n + 1, j \neq i$, do
3. receive S_j from p_j
4. $V := V \cup S_j$

decide( ) occurs as follows:

5. if $k = f + 1$
6. decide min(V)

Suppose every process proposes v^*

Since we only deal with crash failures, only v^* can be sent

By Uniform Integrity of send and receive, only v^* can be received

By the protocol, $V = \{v^*\}$

min(V) = v^*

decide(v^*)

PROVING AGREEMENT

To execute ***propose(v_i)***:

round $k, 1 \leq k \leq f+1$

1. Send $\{v \in V$: p_i has not already sent $v\}$ to all
2. for all $j, 0 \leq j \leq n+1, j \neq i$, do
3. receive S_j from p_j
4. $V := V \cup S_j$

decide( ) occurs as follows:

5. if $k = f+1$
6. decide min(V)

Lemma 1

For any $r \geq 1$, if a process p receives a value v in round r, there exists a sequence of distinct processes $p_0, p_1, \ldots, p_r$ such that $p_r = p$, p_0 is v's proponent and in each round p_{k-1} sends v and p_k receives it.

Proof

By induction on the length of the sequence

PROVING AGREEMENT

To execute ***propose(v_i)***:

round $k, 1 \leq k \leq f+1$

1. Send $\{v \in V$: p_i has not already sent $v\}$ to all
2. for all $j, 0 \leq j \leq n+1, j \neq i$, do
3. receive S_j from p_j
4. $V := V \cup S_j$

decide( ) occurs as follows:

5. if $k = f+1$
6. decide min(V)

Lemma 2

In every execution, at the end of round $f+1$, $V_i = V_j$ for every correct process p_i and p_j

Agreement follows from Lemma 2, since *min* is a deterministic function

Proof

- Show that if a correct p has x in its V at the end of round $f+1$ then every correct process has x in its V at the end of round $f+1$
- Let r be the earliest round x is added to the V set of a correct process. Let that process be p^*
- If $r \leq f$,then p^*sends x in round $r+1 \leq f+1$ Every correct process receives x and adds it to its V in round $r+1$
- **What if $r = f+1$?**
  - By Lemma 1, there exists a sequence of distinct processes $p_0, ..., p_{f+1} = p^*$
  - Consider processes $p_0, ..., p_f$
  - $f+1$ processes; only f can be faulty
  - One of $p_0, ..., p_f$ is correct and adds x to its V before p^* does it in round r

Contradiction!