

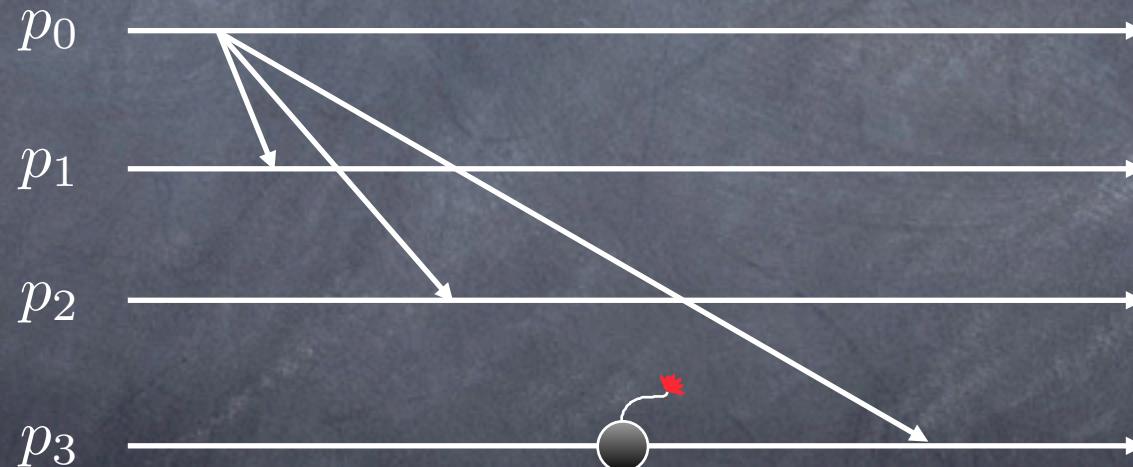
Consensus and Reliable Broadcast

Broadcast

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

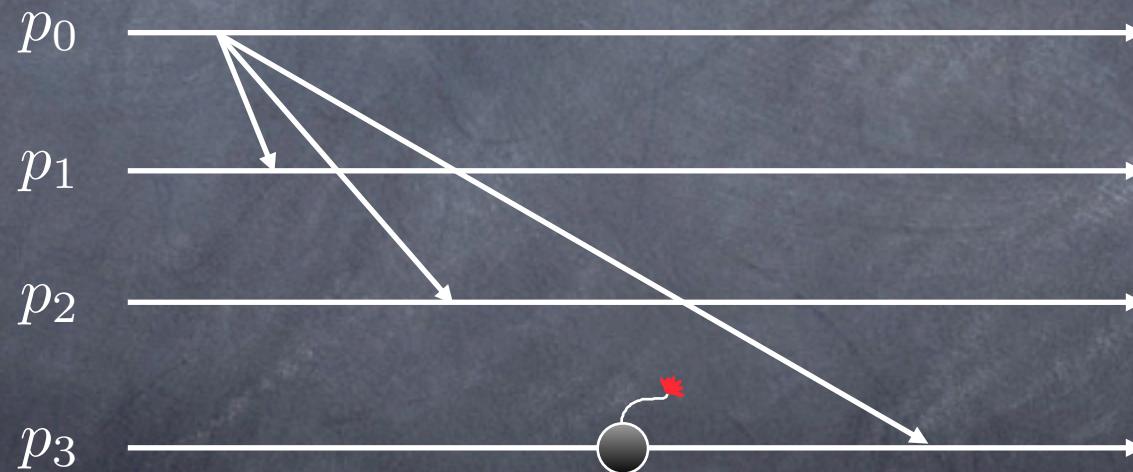
Broadcast

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



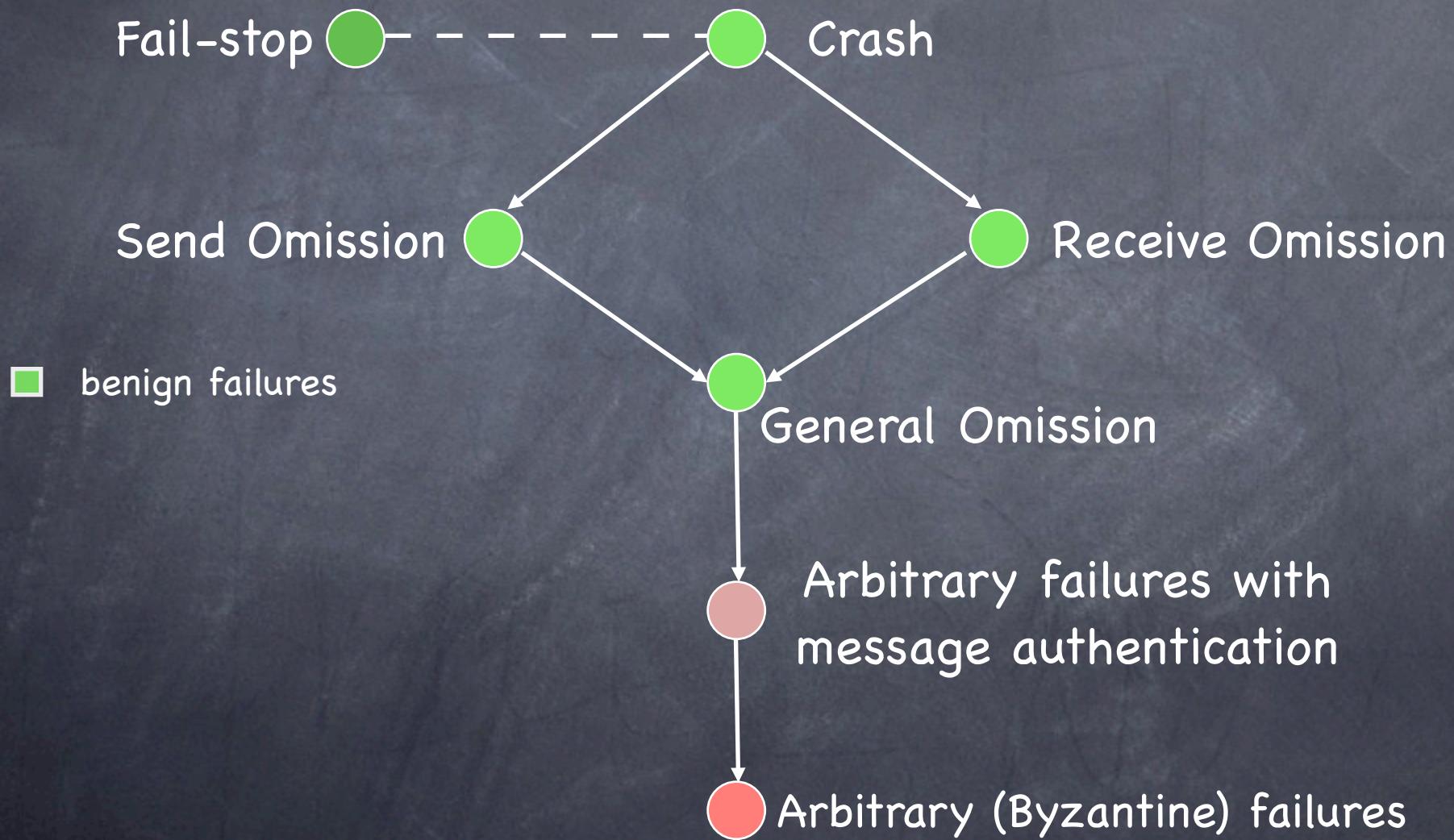
Broadcast

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



- How can we adapt the spec for an environment where processes can fail? And what does “fail” mean?

A hierarchy of failure models



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 , 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

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 any message m , q receives m at most once from p , and only if p sent m to q

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

Properties of $\text{send}(m)$ and $\text{receive}(m)$

Arbitrary failures:

Integrity For any message m , if p and q are correct then q receives m at most once from p , and only if p sent m to q

Questions, Questions...

- ➊ Are these problems solvable at all?
- ➋ Can they be solved independent of the failure model?
- ➌ Does solvability depend on the ratio between faulty and correct processes?
- ➍ Does solvability depend on assumptions about the reliability of the network?
- ➎ Are the problems solvable in both synchronous and asynchronous systems?
- ➏ If a solution exists, how expensive is it?

Plan

② Benign Synchronous Systems

- Consensus for synchronous systems with crash failures
- Lower bound on the number of rounds

③ Benign Asynchronous Systems

- Impossibility of Consensus for crash failures
- Failure detectors
- PAXOS

④ Byzantine (Synchronous and Asynchronous)

- Reliable Broadcast for arbitrary failures
- PBFT, Zyzzyva

Model

- ➊ Synchronous Message Passing
 - Execution is a sequence of rounds
 - In each round every process takes a step
 - sends messages to neighbors
 - receives messages sent in that round
 - changes its state
- ➋ Network is fully connected (an n -clique)
- ➌ No communication failures

A simple Consensus algorithm

Process p_i :

Initially $V = \{v_i\}$

To execute $\text{propose}(v_i)$

1: send $\{v_i\}$ to all

$\text{decide}(x)$ occurs as follows:

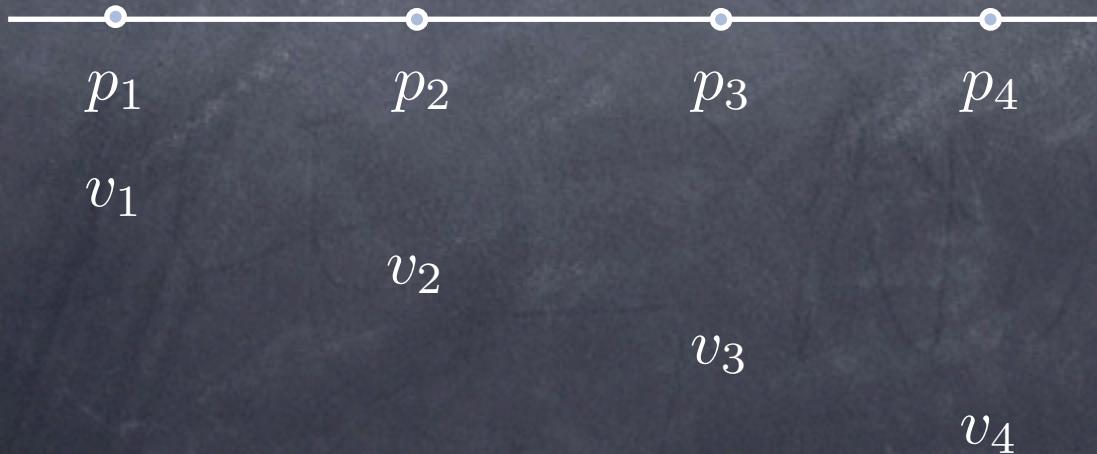
2: for all j , $0 \leq j \leq n-1$, $j \neq i$ do

3: receive S_j from p_j

4: $V := V \cup S_j$

5: decide $\min(V)$

An execution



An execution



v_1

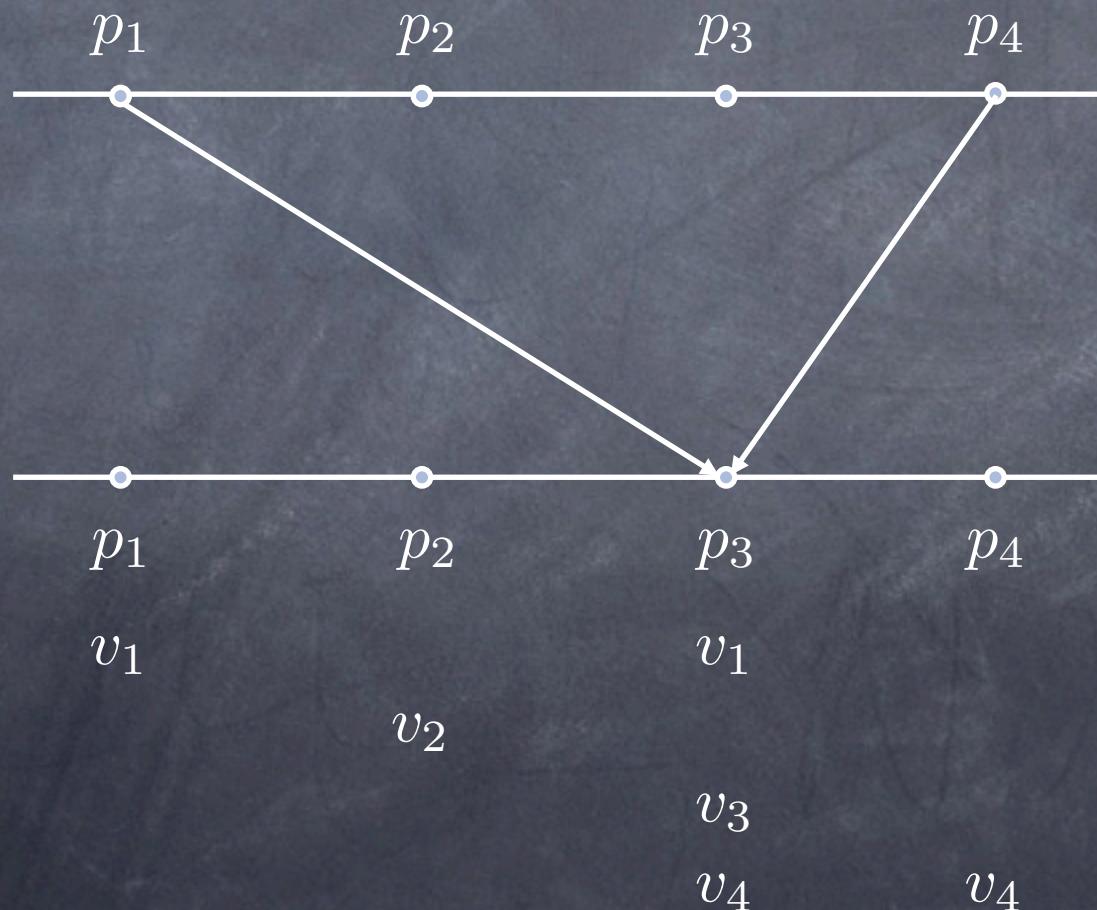
v_2

v_3

v_4

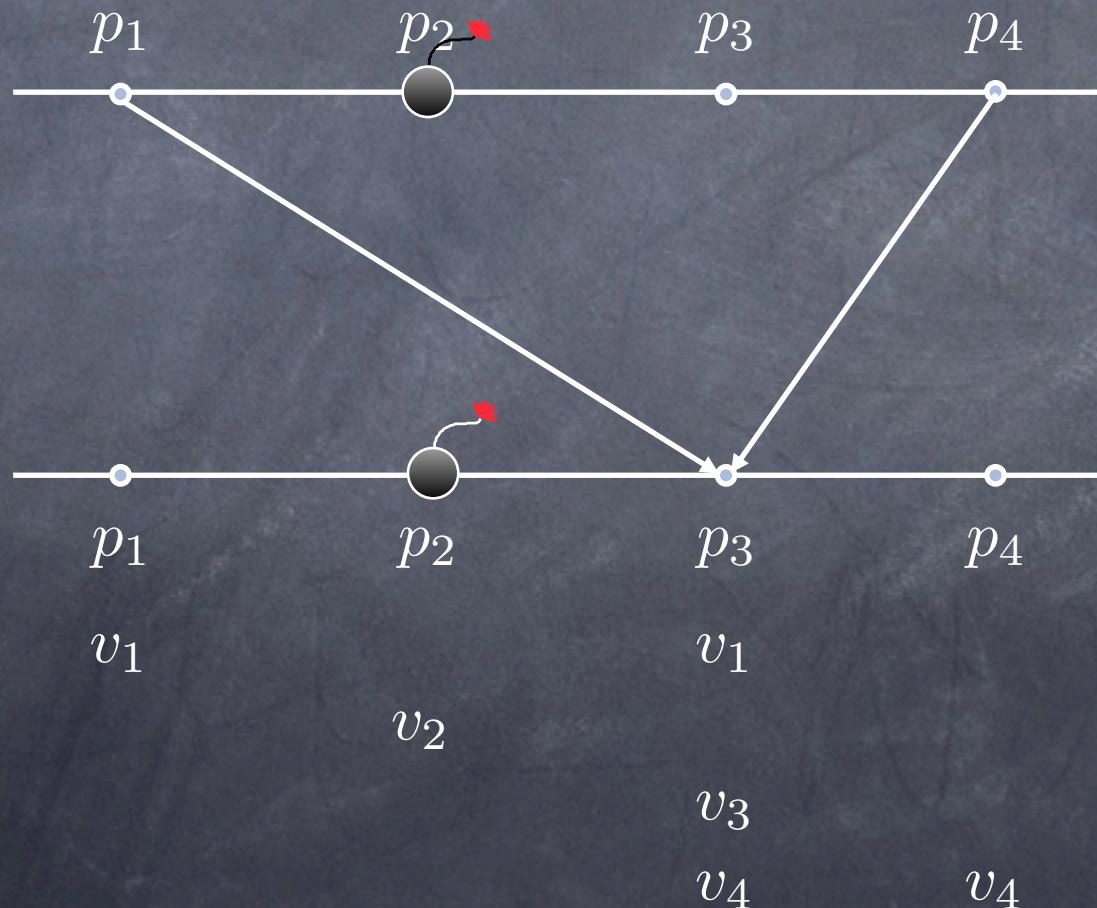
An execution

Suppose $v_1 = v_3 = v_4$ at the end of round 1
Can p_3 decide?



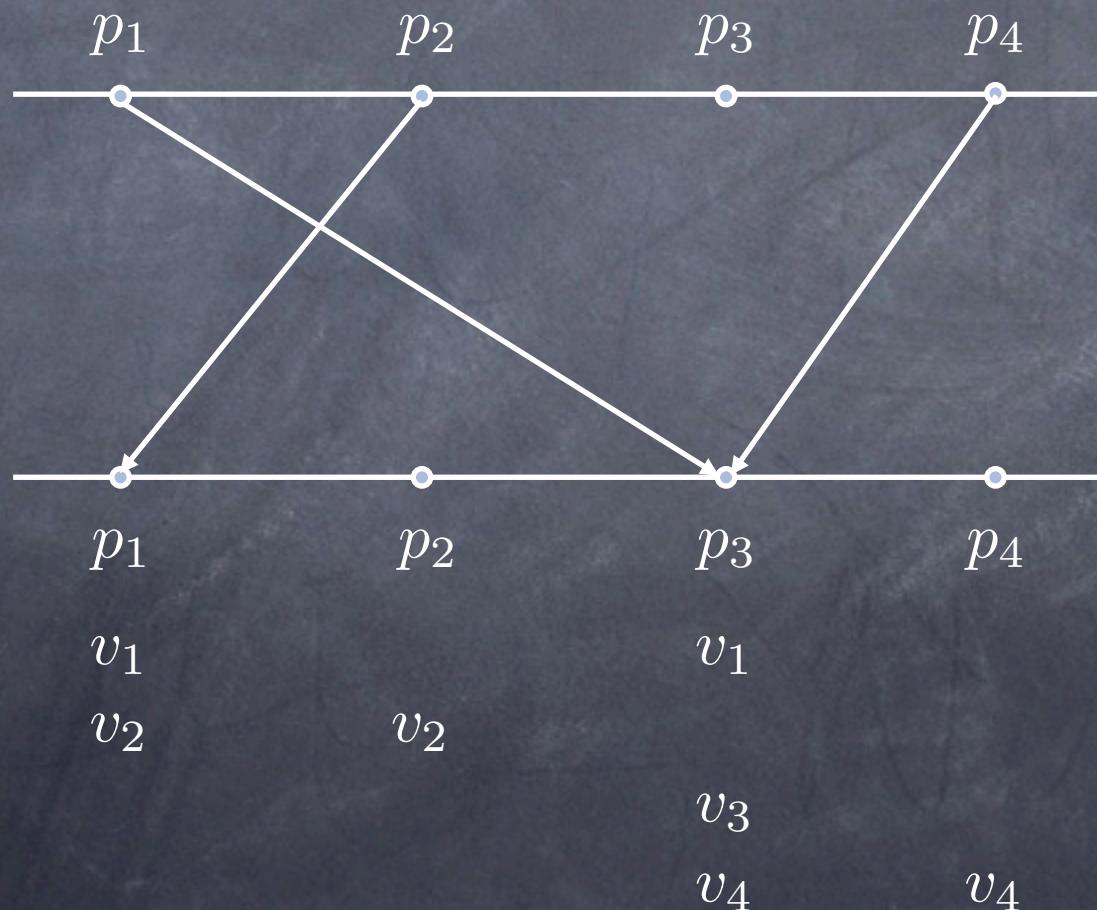
An execution

Suppose $v_1 = v_3 = v_4$ at the end of round 1
Can p_3 decide?



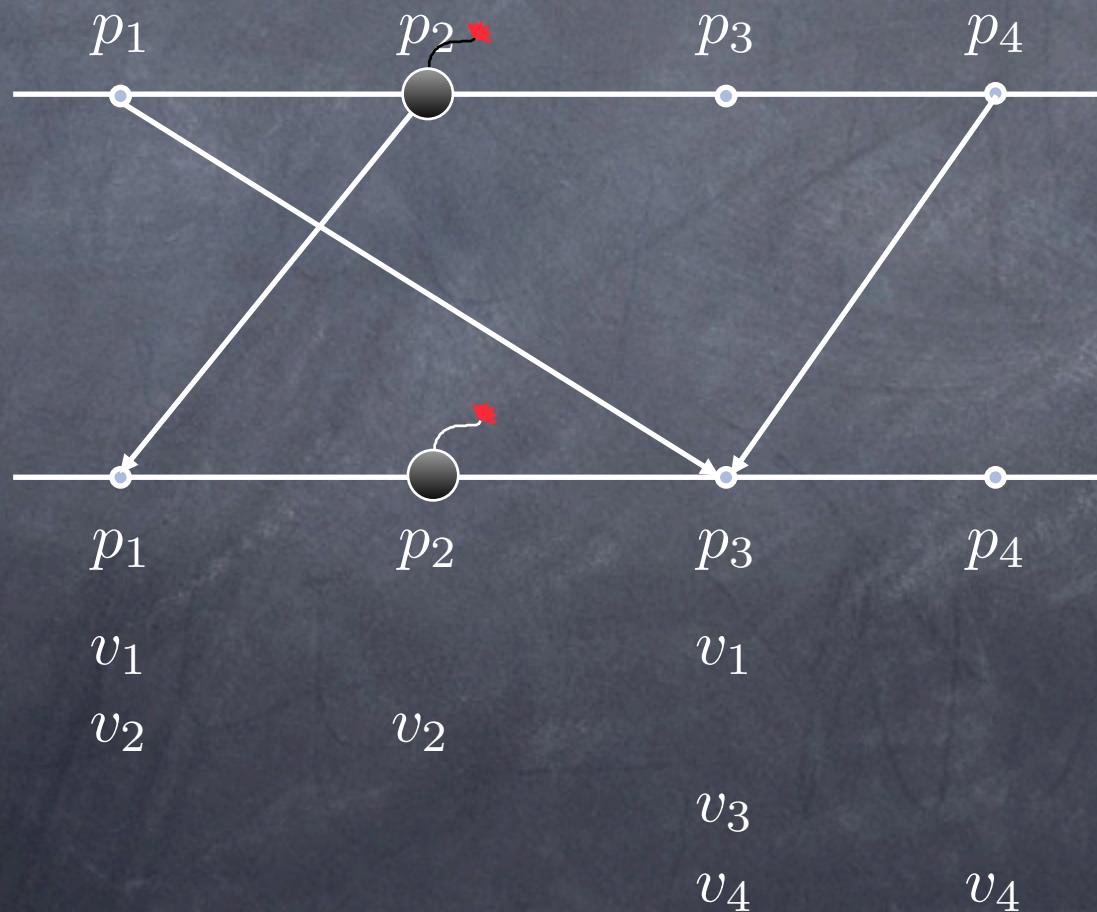
An execution

Suppose $v_1 = v_3 = v_4$ at the end of round 1
Can p_3 decide?



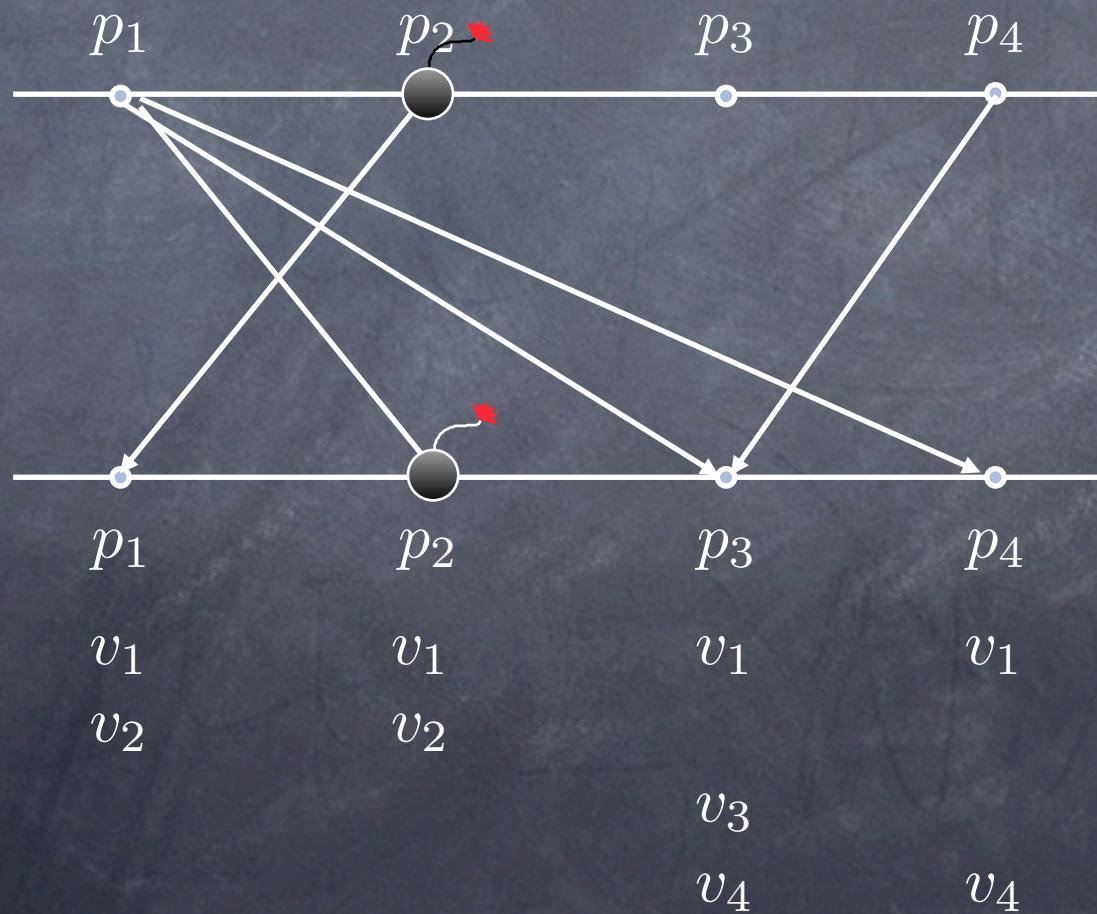
An execution

Suppose $v_1 = v_3 = v_4$ at the end of round 1
Can p_3 decide?



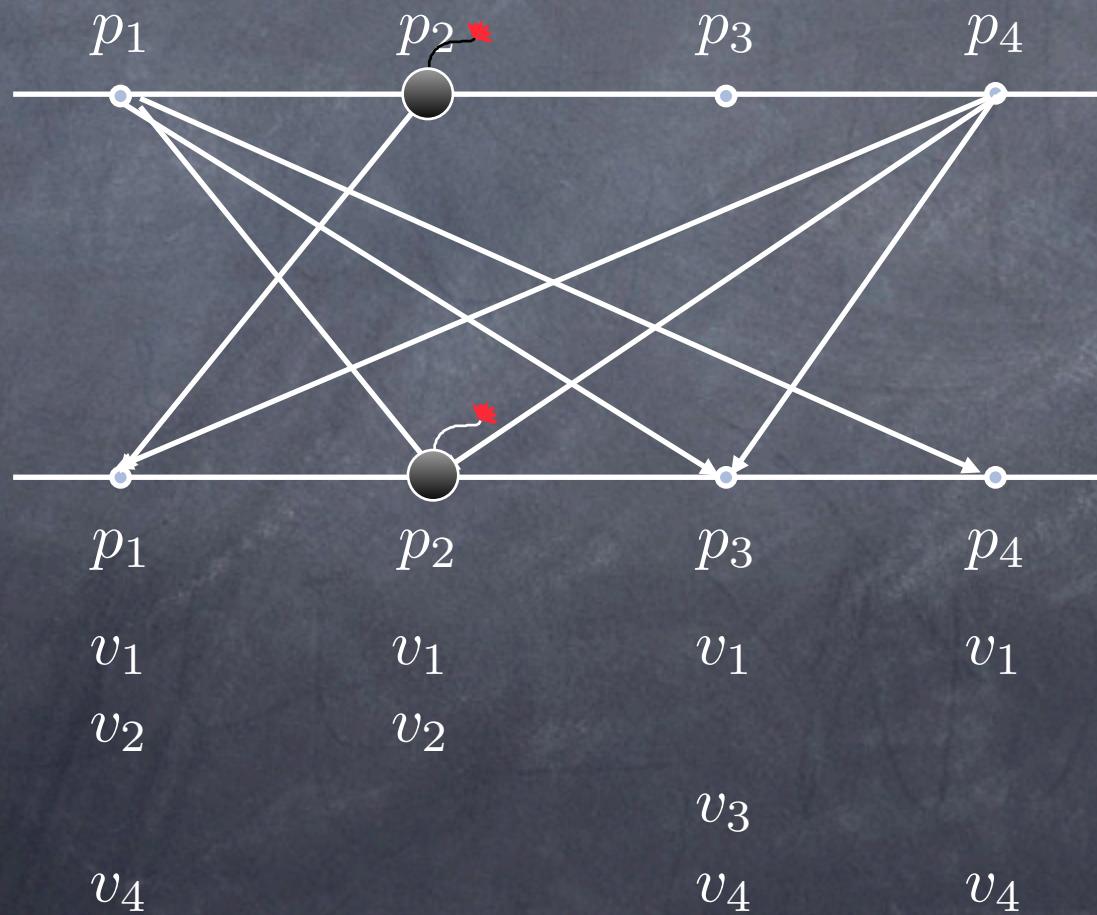
An execution

Suppose $v_1 = v_3 = v_4$ at the end of round 1
Can p_3 decide?



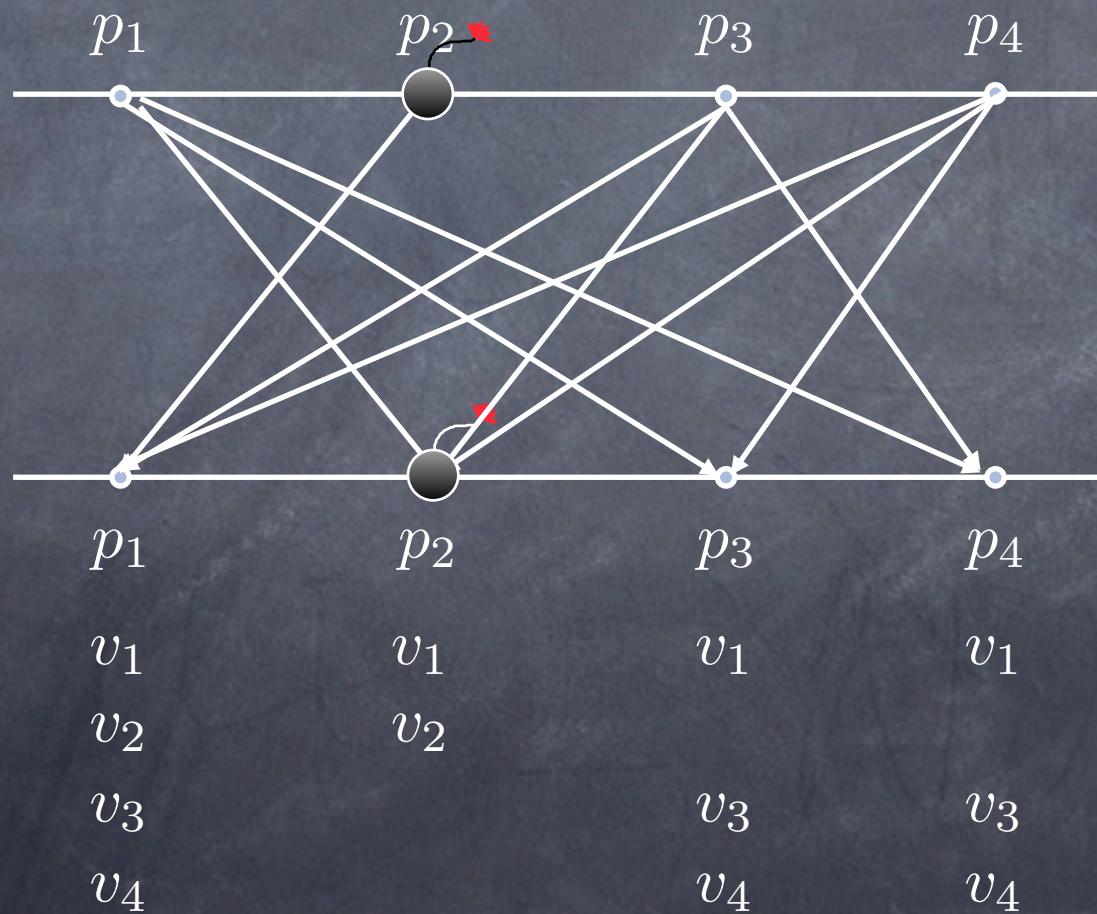
An execution

Suppose $v_1 = v_3 = v_4$ at the end of round 1
Can p_3 decide?



An execution

Suppose $v_1 = v_3 = v_4$ at the end of round 1
Can p_3 decide?



Echoing values

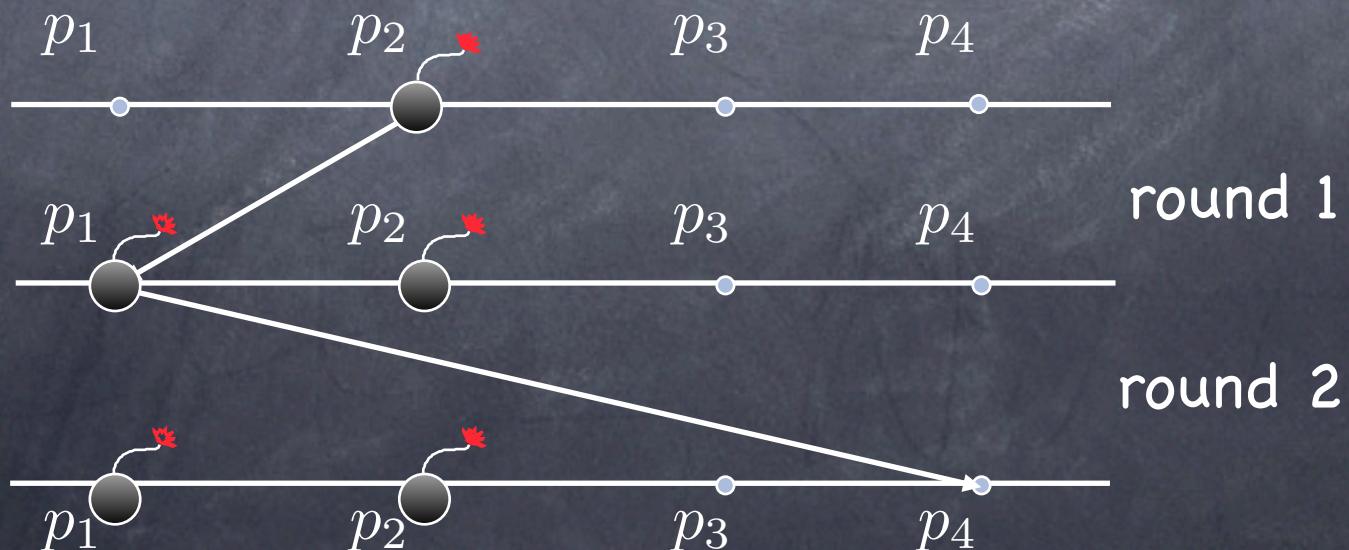
- ④ A process that receives a proposal in round 1, relays it to others during round 2.

Echoing values

- ➊ A process that receives a proposal in round 1, relays it to others during round 2.
- ➋ Suppose p_3 hasn't heard from p_2 at the end of round 2. Can p_3 decide?

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?



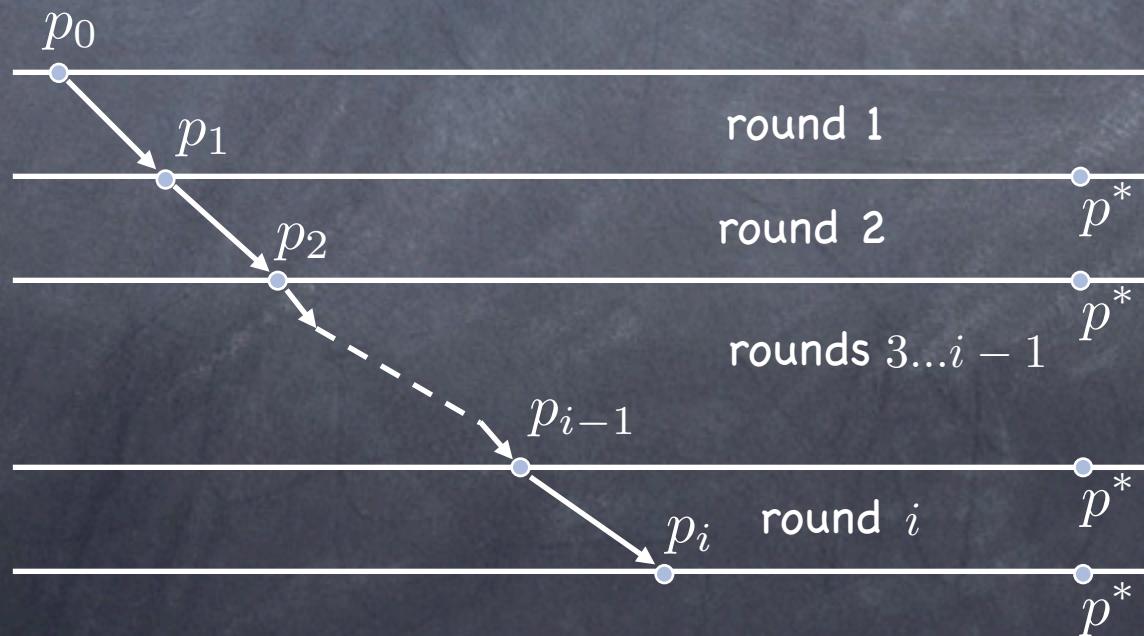
What is going on

- ➊ A correct process p^* has not received all proposals by the end of round i . Can p^* decide?
- ➋ 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 are 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 $f+1$!

The Algorithm

Code for process p_i :

Initially $V = \{v_i\}$

To execute $\text{propose}(v_i)$

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

1: send $\{v \in V : p_i \text{ has not already sent } v\}$ to all

2: for all j , $0 \leq j \leq n-1$, $j \neq i$ do

3: receive S_j from p_j

4: $V := V \cup S_j$

$\text{decide}(x)$ occurs as follows:

5: if $k = f+1$ then

6: decide $\min(V)$

Termination and Integrity

Initially $V = \{v_i\}$

To execute $\text{propose}(v_i)$

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

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

$\text{decide}(x)$ occurs as follows:

- 5: if $k = f+1$ then
- 6: decide $\min(V)$

Termination

Termination and Integrity

Initially $V = \{v_i\}$

To execute $\text{propose}(v_i)$

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

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

$\text{decide}(x)$ occurs as follows:

- 5: if $k = f+1$ then
- 6: decide $\min(V)$

Termination

Every correct process

reaches round $f + 1$

Decides on $\min(V)$ --- which is well defined

Termination and Integrity

Initially $V = \{v_i\}$

To execute $\text{propose}(v_i)$

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

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

$\text{decide}(x)$ occurs as follows:

- 5: if $k = f + 1$ then
- 6: decide $\min(V)$

Integrity

At most one value:

Only if it was proposed:

Termination

Every correct process

reaches round $f + 1$

Decides on $\min(V)$ --- which is well defined

Termination and Integrity

Initially $V = \{v_i\}$

To execute $\text{propose}(v_i)$

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

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

$\text{decide}(x)$ occurs as follows:

- 5: if $k = f+1$ then
- 6: decide $\min(V)$

Integrity

At most one value:

- one decide, and $\min(V)$ is unique

Only if it was proposed:

Termination

Every correct process

reaches round $f + 1$

Decides on $\min(V)$ --- which is well defined

Termination and Integrity

Initially $V = \{v_i\}$

To execute $\text{propose}(v_i)$

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

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

$\text{decide}(x)$ occurs as follows:

- 5: if $k = f+1$ then
- 6: decide $\min(V)$

Integrity

At most one value:

- one decide, and $\min(V)$ is unique

Only if it was proposed:

- To be decided upon, must be in V at round $f+1$
- if value = v_i , then it is proposed in round 1
- else, suppose 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 only crash 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 protocol, must have been received by sender by end of round j
 - by induction hypothesis, must have been proposed

Termination

Every correct process

- ② reaches round $f+1$
- ② Decides on $\min(V)$ --- which is well defined

Validity

Initially $V = \{v_i\}$

To execute $\text{propose}(v_i)$

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

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

$\text{decide}(x)$ occurs as follows:

- 5: if $k = f+1$ then
- 6: decide $\min(V)$

Validity

Initially $V = \{v_i\}$

To execute $\text{propose}(v_i)$

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

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

$\text{decide}(x)$ occurs as follows:

- 5: if $k = f+1$ then
- 6: decide $\min(V)$

- Suppose every process proposes v^*
- Since only crash model, only v^* can be sent
 - By Uniform Integrity of send and receive, only v^* can be received
- By protocol, $V = \{v^*\}$
- $\min(V) = v^*$
- $\text{decide}(v^*)$

Agreement

Initially $V = \{v_i\}$

To execute $\text{propose}(v_i)$

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

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

$\text{decide}(x)$ occurs as follows:

- 5: if $k = f+1$ then
- 6: decide $\min(V)$

Lemma 1

For any $r \geq 1$, if a process p receives a value v in round r , then there exists a sequence of processes p_0, p_1, \dots, 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. Furthermore, all processes in the sequence are distinct.

Proof

By induction on the length of the sequence

Agreement

Initially $V = \{v_i\}$

To execute $\text{propose}(v_i)$

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

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

$\text{decide}(x)$ occurs as follows:

- 5: if $k = f+1$ then
- 6: decide $\min(V)$

Lemma 2:

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

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

Agreement

Initially $V = \{v_i\}$

To execute $\text{propose}(v_i)$

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

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

$\text{decide}(x)$ occurs as follows:

- 5: if $k = f+1$ then
- 6: decide $\min(V)$

Proof:

- Show that if a correct p has x in its V at the end of round $f+1$, then every correct p has x in its V at the end of round $f+1$

Lemma 2:

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

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

Agreement

Initially $V = \{v_i\}$

To execute $\text{propose}(v_i)$

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

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

$\text{decide}(x)$ occurs as follows:

- 5: if $k = f+1$ then
- 6: decide $\min(V)$

Proof:

- Show that if a correct p has x in its V at the end of round $f+1$, then every correct p has x in its V at the end of round $f+1$
- Let r be earliest round x is added to the V of a correct p . 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 x to its V in round $r+1$

Lemma 2:

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

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

Agreement

Initially $V = \{v_i\}$

To execute $\text{propose}(v_i)$

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

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

$\text{decide}(x)$ occurs as follows:

- 5: if $k = f+1$ then
- 6: decide $\min(V)$

Proof:

- Show that if a correct p has x in its V at the end of round $f+1$, then every correct p has x in its V at the end of round $f+1$
- Let r be earliest round x is added to the V of a correct p . 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 x to its V in round $r+1$
- What if $r = f+1$?

Lemma 2:

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

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

Agreement

Initially $V = \{v_i\}$

To execute $\text{propose}(v_i)$

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

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

$\text{decide}(x)$ occurs as follows:

- 5: if $k = f+1$ then
- 6: decide $\min(V)$

Lemma 2:

In every execution, at the end of round $f+1$, $V_i = V_j$ for every correct processes 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 p has x in its V at the end of round $f+1$
- Let r be earliest round x is added to the V of a correct p . 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 x to its V in round $r+1$
- What if $r = f+1$?
- By Lemma 1, there exists a sequence of distinct processes $p_0, \dots, p_{f+1} = p^*$
- Consider processes p_0, \dots, p_f
- $f+1$ processes; only f faulty
- one of p_0, \dots, p_f is correct, and adds x to its V before p^* does it in round r

CONTRADICTION!

Terminating Reliable Broadcast

- Validity** If the sender is correct and broadcasts a message $_n$, 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 $\neq SF$, then some process must have broadcast
- Termination** Every correct process eventually delivers

TRB for benign failures

Sender in round 1:

1: send m to all

Process p in round $k, 1 \leq k \leq f+1$

1: if delivered m in round $k-1$ then

2: if $p \neq$ sender then

3: send m to all

4: halt

5: receive round k messages

6: if received m then

7: deliver(m)

8: if $k = f+1$ then halt

9: else if $k = f+1$

10: deliver(SF)

11: halt

TRB for benign failures

Sender in round 1:

1: send m to all

Process p in round k , $1 \leq k \leq f+1$

```
1: if delivered m in round k-1 then
2:   if p ≠ sender then
3:     send m to all
4:   halt
5: receive round k messages
6: if received m then
7:   deliver(m)
8:   if k = f+1 then halt
9: else if k = f+1
10:  deliver(SF)
11:  halt
```

Terminates in $f+1$ rounds

How can we do better?

Find a protocol whose round complexity is proportional to t - the number of failures that actually occurred - rather than to f - the max number of failures that may occur

Early stopping: the idea

- Suppose processes can detect the set of processes that have failed by the end of round i
- Call that set $\text{faulty}(p, i)$
- If $|\text{faulty}(p, i)| < i$ there can be no active dangerous chains, and p can safely deliver SF

Early Stopping: The Protocol

$\text{faulty}(p, k) \equiv \text{set of processes that failed to send a message to } p \text{ in some round}$

1: if $p = \text{sender}$ then $\text{value} := m$ else $\text{value} := ?$

Process p in round $k, 1 \leq k \leq f+1$

2: send value to all

3: if delivered in round $k-1$ then halt

4: receive round k values from all

5: $\text{faulty}(p, k) := \text{faulty}(p, k-1) \cup \{q \mid p \text{ received no value from } q \text{ in round } k\}$

6: if received value $v \neq ?$ then

7: $\text{value} := v$

8: deliver value

9: if $p = \text{sender}$ then $\text{value} := ?$

10: else if $k = f+1$ or $|\text{faulty}(p, k)| < k$ then

11: $\text{value} := \text{SF}$

12: deliver value

13: if $k = f+1$ then halt

Termination

Let $\text{faulty}(p, k)$ be the set of processes that have failed to send a message to p in any round $1, \dots, k$

1: if $p = \text{sender}$ then $\text{value} := m$ else $\text{value} := ?$

Process p in round $k, 1 \leq k \leq f+1$

2: send value to all
3: if delivered in round $k-1$ then halt
4: receive round k values from all
5: $\text{faulty}(p, k) := \text{faulty}(p, k-1) \cup \{q \mid p$
received no value from q in round $k\}$
6: if received value $v \neq ?$ then
7: $\text{value} := v$
8: deliver value
9: if $p = \text{sender}$ then $\text{value} := ?$
10: else if $k = f+1$ or $|\text{faulty}(p, k)| < k$ then
11: $\text{value} := \text{SF}$
12: deliver value
13: if $k = f+1$ then halt

Termination

Let $\text{faulty}(p, k)$ be the set of processes that have failed to send a message to p in any round $1, \dots, k$

1: if $p = \text{sender}$ then $\text{value} := m$ else $\text{value} := ?$

Process p in round $k, 1 \leq k \leq f+1$

2: send value to all
3: if delivered in round $k-1$ then halt
4: receive round k values from all
5: $\text{faulty}(p, k) := \text{faulty}(p, k-1) \cup \{q \mid p$
received no value from q in round $k\}$
6: if received value $v \neq ?$ then
7: $\text{value} := v$
8: deliver value
9: if $p = \text{sender}$ then $\text{value} := ?$
10: else if $k = f+1$ or $|\text{faulty}(p, k)| < k$ then
11: $\text{value} := \text{SF}$
12: deliver value
13: if $k = f+1$ then halt

② If in any round a process receives a value, then it delivers the value in that round

③ If a process has received only "?" for $f+1$ rounds, then it delivers SF in round $f+1$

Validity

Let $\text{faulty}(p, k)$ be the set of processes that have failed to send a message to p in any round $1, \dots, k$

1: if $p = \text{sender}$ then $\text{value} := m$ else $\text{value} := ?$

Process p in round $k, 1 \leq k \leq f+1$

2: send value to all
3: if delivered in round $k-1$ then halt
4: receive round k values from all
5: $\text{faulty}(p, k) := \text{faulty}(p, k-1) \cup \{q \mid p$
received no value from q in round $k\}$
6: if received value $v \neq ?$ then
7: $\text{value} := v$
8: deliver value
9: if $p = \text{sender}$ then $\text{value} := ?$
10: else if $k = f+1$ or $|\text{faulty}(p, k)| < k$ then
11: $\text{value} := \text{SF}$
12: deliver value
13: if $k = f+1$ then halt

Validity

Let $\text{faulty}(p, k)$ be the set of processes that have failed to send a message to p in any round $1, \dots, k$

1: if $p = \text{sender}$ then $\text{value} := m$ else $\text{value} := ?$

Process p in round $k, 1 \leq k \leq f+1$

2: send value to all
3: if delivered in round $k-1$ then halt
4: receive round k values from all
5: $\text{faulty}(p, k) := \text{faulty}(p, k-1) \cup \{q \mid p$
received no value from q in round $k\}$
6: if received value $v \neq ?$ then
7: $\text{value} := v$
8: deliver value
9: if $p = \text{sender}$ then $\text{value} := ?$
10: else if $k = f+1$ or $|\text{faulty}(p, k)| < k$ then
11: $\text{value} := \text{SF}$
12: deliver value
13: if $k = f+1$ then halt

- ➊ If the sender is correct then it sends m to all in round 1
- ➋ By Validity of the underlying send and receive, every correct process will receive m by the end of round 1
- ➌ By the protocol, every correct process will deliver m by the end of round 1

Agreement – 1

Let $\text{faulty}(p, k)$ be the set of processes that have failed to send a message to p in any round $1, \dots, k$

1: if $p = \text{sender}$ then $\text{value} := m$ else $\text{value} := ?$

Process p in round $k, 1 \leq k \leq f+1$

2: send value to all
3: if delivered in round $k-1$ then halt
4: receive round k values from all
5: $\text{faulty}(p, k) := \text{faulty}(p, k-1) \cup \{q \mid p$
received no value from q in round $k\}$
6: if received value $v \neq ?$ then
7: $\text{value} := v$
8: deliver value
9: if $p = \text{sender}$ then $\text{value} := ?$
10: else if $k = f+1$ or $|\text{faulty}(p, k)| < k$ then
11: $\text{value} := \text{SF}$
12: deliver value
13: if $k = f+1$ then halt

Lemma 1:

For any $r \geq 1$, if a process p delivers $m \neq \text{SF}$ in round r , then there exists a sequence of processes p_0, p_1, \dots, p_r such that $p_0 = \text{sender}$, $p_r = p$, and in each round $j, 1 \leq j \leq r$, p_{j-1} sent p_j and received it. Furthermore, all processes in the sequence are distinct, unless $r = 1$ and $p_0 = p_1 = \text{sender}$

Lemma 2:

For any $r \geq 1$, if a process p sets value to SF in round r , then there exist some $j \leq r$ and a sequence of distinct processes $q_j, q_{j+1}, \dots, q_r = p$ such that q_j only receives "?" in rounds 1 to j , $|\text{faulty}(q_j, j)| < j$, and in each round $k, j+1 \leq k \leq r$, q_{k-1} sends SF to q_k and q_k receives SF

Agreement – 2

Let $\text{faulty}(p, k)$ be the set of processes that have failed to send a message to p in any round $1, \dots, k$

1: if $p = \text{sender}$ then $\text{value} := m$ else $\text{value} := ?$

Process p in round $k, 1 \leq k \leq f+1$

2: send value to all
3: if delivered in round $k-1$ then halt
4: receive round k values from all
5: $\text{faulty}(p, k) := \text{faulty}(p, k-1) \cup \{q \mid p$
received no value from q in round $k\}$
6: if received value $v \neq ?$ then
7: $\text{value} := v$
8: deliver value
9: if $p = \text{sender}$ then $\text{value} := ?$
10: else if $k = f+1$ or $|\text{faulty}(p, k)| < k$ then
11: $\text{value} := \text{SF}$
12: deliver value
13: if $k = f+1$ then halt

Lemma 3:

It is impossible for p and q , not necessarily correct or distinct, to set value in the same round r to m and SF , respectively

Agreement – 2

Let $\text{faulty}(p, k)$ be the set of processes that have failed to send a message to p in any round $1, \dots, k$

1: if $p = \text{sender}$ then value := m else value:= ?

Process p in round $k, 1 \leq k \leq f+1$

2: send value to all

3: if delivered in round $k-1$ then halt

4: receive round k values from all

5: $\text{faulty}(p, k) := \text{faulty}(p, k-1) \cup \{q \mid p \text{ received no value from } q \text{ in round } k\}$

6: if received value $v \neq ?$ then

7: value := v

8: deliver value

9: if $p = \text{sender}$ then value := ?

10: else if $k = f+1$ or $|\text{faulty}(p, k)| < k$ then

11: value := SF

12: deliver value

13: if $k = f+1$ then halt

Lemma 3:

It is impossible for p and q , not necessarily correct or distinct, to set value in the same round r to m and SF, respectively

Proof

By contradiction

Suppose p sets value = m and q sets value = SF

By Lemmas 1 and 2 there exist

p_0, \dots, p_r

q_j, \dots, q_r

with the appropriate characteristics

Since q_j did not receive m from process $p_{k-1} \quad 1 \leq k \leq j$ in round k q_j must conclude that p_0, \dots, p_{j-1} are all faulty processes

But then, $|\text{faulty}(q_j, j)| \geq j$

CONTRADICTION

Agreement – 3

Let $\text{faulty}(p, k)$ be the set of processes that have failed to send a message to p in any round $1, \dots, k$

1: if $p = \text{sender}$ then $\text{value} := m$ else $\text{value} := ?$

Process p in round $k, 1 \leq k \leq f+1$

2: send value to all
3: if delivered in round $k-1$ then halt
4: receive round k values from all
5: $\text{faulty}(p, k) := \text{faulty}(p, k-1) \cup \{q \mid p$
received no value from q in round $k\}$
6: if received value $v \neq ?$ then
7: $\text{value} := v$
8: deliver value
9: if $p = \text{sender}$ then $\text{value} := ?$
10: else if $k = f+1$ or $|\text{faulty}(p, k)| < k$ then
11: $\text{value} := \text{SF}$
12: deliver value
13: if $k = f+1$ then halt

Agreement – 3

Let $\text{faulty}(p, k)$ be the set of processes that have failed to send a message to p in any round $1, \dots, k$

1: if $p = \text{sender}$ then value := m else value:= ?

Process p in round $k, 1 \leq k \leq f+1$

2: send value to all
3: if delivered in round $k-1$ then halt
4: receive round k values from all
5: $\text{faulty}(p, k) := \text{faulty}(p, k-1) \cup \{q \mid p \text{ received no value from } q \text{ in round } k\}$
6: if received value $v \neq ?$ then
7: value := v
8: deliver value
9: if $p = \text{sender}$ then value := ?
10: else if $k = f+1$ or $|\text{faulty}(p, k)| < k$ then
11: value := SF
12: deliver value
13: if $k = f+1$ then halt

Proof

If no correct process ever receives m , then every correct process delivers SF in round $f+1$

Let r be the earliest round in which a correct process delivers value $\neq \text{SF}$

$r \leq f$

- By Lemma 3, no (correct) process can set value differently in round r
- In round $r+1 \leq f+1$, that correct process sends its value to all
- Every correct process receives and delivers the value in round $r+1 \leq f+1$

$r = f+1$

- By Lemma 1, there exists a sequence $p_0, \dots, p_{f+1} = p_r$ of distinct processes
- Consider processes p_0, \dots, p_f
 - ② $f+1$ processes; only f faulty
 - ② one of p_0, \dots, p_f is correct-- let it be p_c
 - ② To send v in round $c+1$, p_c must have set its value to v and delivered v in round $c < r$

CONTRADICTION

Integrity

Let $\text{faulty}(p, k)$ be the set of processes that have failed to send a message to p in any round $1, \dots, k$

1: if $p = \text{sender}$ then $\text{value} := m$ else $\text{value} := ?$

Process p in round $k, 1 \leq k \leq f+1$

2: send value to all
3: if delivered in round $k-1$ then halt
4: receive round k values from all
5: $\text{faulty}(p, k) := \text{faulty}(p, k-1) \cup \{q \mid p$
received no value from q in round $k\}$
6: if received value $v \neq ?$ then
7: $\text{value} := v$
8: deliver value
9: if $p = \text{sender}$ then $\text{value} := ?$
10: else if $k = f+1$ or $|\text{faulty}(p, k)| < k$ then
11: $\text{value} := \text{SF}$
12: deliver value
13: if $k = f+1$ then halt

Integrity

Let $\text{faulty}(p, k)$ be the set of processes that have failed to send a message to p in any round $1, \dots, k$

1: if $p = \text{sender}$ then $\text{value} := m$ else $\text{value} := ?$

Process p in round $k, 1 \leq k \leq f+1$

2: send value to all
3: if delivered in round $k-1$ then halt
4: receive round k values from all
5: $\text{faulty}(p, k) := \text{faulty}(p, k-1) \cup \{q \mid p$
received no value from q in round $k\}$
6: if received value $v \neq ?$ then
7: $\text{value} := v$
8: deliver value
9: if $p = \text{sender}$ then $\text{value} := ?$
10: else if $k = f+1$ or $|\text{faulty}(p, k)| < k$ then
11: $\text{value} := \text{SF}$
12: deliver value
13: if $k = f+1$ then halt

② At most one m

□ Failures are benign, and a process executes at most one deliver event before halting

③ If $m \neq \text{SF}$, only if m was broadcast

□ From Lemma 1 in the proof of Agreement

A Lower Bound

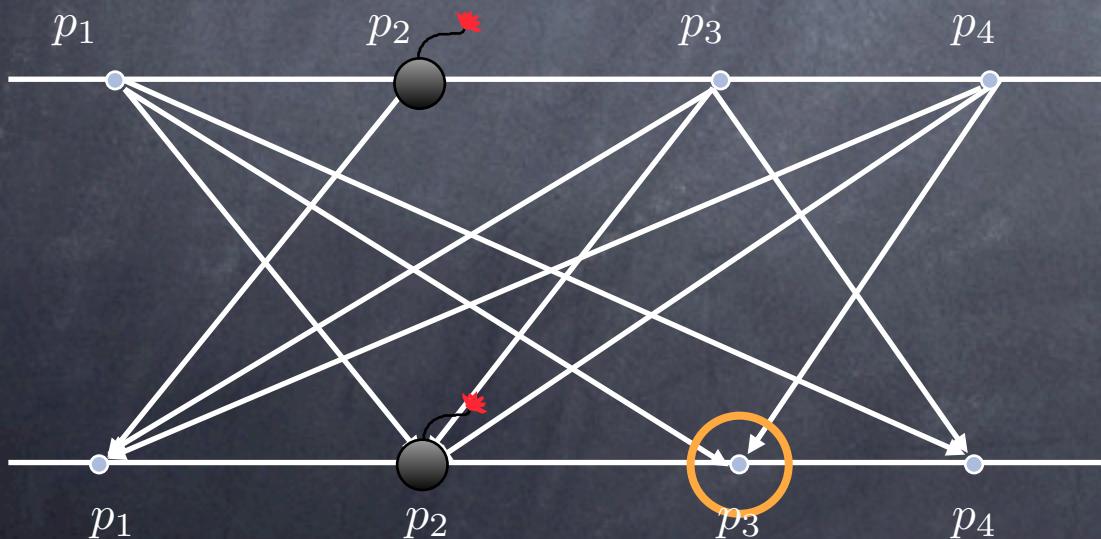
Theorem

There is no algorithm that solves the consensus problem in fewer than $f+1$ rounds in the presence of f crash failures, if $n \geq f+2$

We consider a special case ($f=1$) to study the proof technique

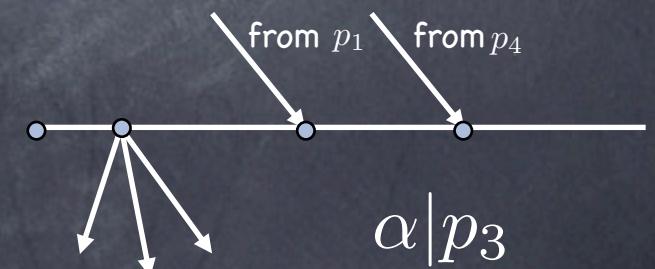
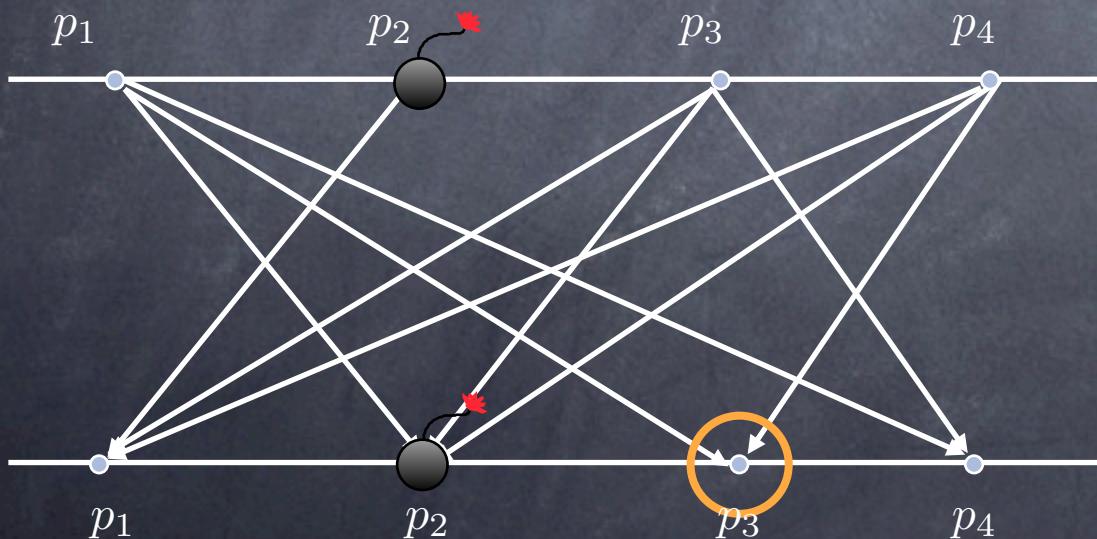
Views

Let α be an execution. The **view** of process p_i in α , denoted by $\alpha|_{p_i}$, is the subsequence of computation and message receive events that occur in p_i together with the state of p_i in the initial configuration of α .



Views

Let α be an execution. The **view** of process p_i in α , denoted by $\alpha|_{p_i}$ is the subsequence of computation and message receive events that occur in p_i together with the state of p_i in the initial configuration of α .



Similarity

Definition Let α_1 and α_2 be two executions of consensus and let p_i be a correct process in both α_1 and α_2 .

α_1 is **similar** to α_2 with respect to p_i , denoted $\alpha_1 \sim_{p_i} \alpha_2$ if

$$\alpha_1|_{p_i} = \alpha_2|_{p_i}$$

Similarity

Definition Let α_1 and α_2 be two executions of consensus and let p_i be a correct process in both α_1 and α_2 .

α_1 is **similar** to α_2 with respect to p_i , denoted $\alpha_1 \sim_{p_i} \alpha_2$ if

$$\alpha_1|_{p_i} = \alpha_2|_{p_i}$$

Note If $\alpha_1 \sim_{p_i} \alpha_2$ then p_i decides the same value in both executions

Similarity

Definition Let α_1 and α_2 be two executions of consensus and let p_i be a correct process in both α_1 and α_2 .

α_1 is **similar** to α_2 with respect to p_i , denoted $\alpha_1 \sim_{p_i} \alpha_2$ if

$$\alpha_1|_{p_i} = \alpha_2|_{p_i}$$

Note If $\alpha_1 \sim_{p_i} \alpha_2$ then p_i decides the same value in both executions

Lemma If $\alpha_1 \sim_{p_i} \alpha_2$ and p_i is correct, then $\text{dec}(\alpha_1) = \text{dec}(\alpha_2)$

Similarity

Definition Let α_1 and α_2 be two executions of consensus and let p_i be a correct process in both α_1 and α_2 .

α_1 is **similar** to α_2 with respect to p_i , denoted $\alpha_1 \sim_{p_i} \alpha_2$ if

$$\alpha_1|p_i = \alpha_2|p_i$$

Note If $\alpha_1 \sim_{p_i} \alpha_2$ then p_i decides the same value in both executions

Lemma If $\alpha_1 \sim_{p_i} \alpha_2$ and p_i is correct, then $\text{dec}(\alpha_1) = \text{dec}(\alpha_2)$

The transitive closure of $\alpha_1 \sim_{p_i} \alpha_2$ is denoted $\alpha_1 \approx \alpha_2$.

We say that $\alpha_1 \approx \alpha_2$ if there exist executions $\beta_1, \beta_2, \dots, \beta_{k+1}$ such that

$$\alpha_1 = \beta_1 \sim_{p_{i_1}} \beta_2 \sim_{p_{i_2}} \dots \sim_{p_{i_k}} \beta_{k+1} = \alpha_2$$

Similarity

Definition Let α_1 and α_2 be two executions of consensus and let p_i be a correct process in both α_1 and α_2 .

α_1 is **similar** to α_2 with respect to p_i , denoted $\alpha_1 \sim_{p_i} \alpha_2$ if

$$\alpha_1|p_i = \alpha_2|p_i$$

Note If $\alpha_1 \sim_{p_i} \alpha_2$ then p_i decides the same value in both executions

Lemma If $\alpha_1 \sim_{p_i} \alpha_2$ and p_i is correct, then $\text{dec}(\alpha_1) = \text{dec}(\alpha_2)$

The transitive closure of $\alpha_1 \sim_{p_i} \alpha_2$ is denoted $\alpha_1 \approx \alpha_2$.

We say that $\alpha_1 \approx \alpha_2$ if there exist executions $\beta_1, \beta_2, \dots, \beta_{k+1}$ such that

$$\alpha_1 = \beta_1 \sim_{p_{i_1}} \beta_2 \sim_{p_{i_2}} \dots \sim_{p_{i_k}} \beta_{k+1} = \alpha_2$$

Lemma If $\alpha_1 \approx \alpha_2$ then
 $\text{dec}(\alpha_1) = \text{dec}(\alpha_2)$

Single-Failure Case

There is no algorithm that solves consensus in fewer than two rounds in the presence of one crash failure, if $n \geq 3$

The Idea

By contradiction

- ➊ Consider a one-round execution in which each process proposes 0. What is the decision value?
- ➋ Consider another one-round execution in which each process proposes 1. What is the decision value?
- ➌ Show that there is a chain of similar executions that relate the two executions.

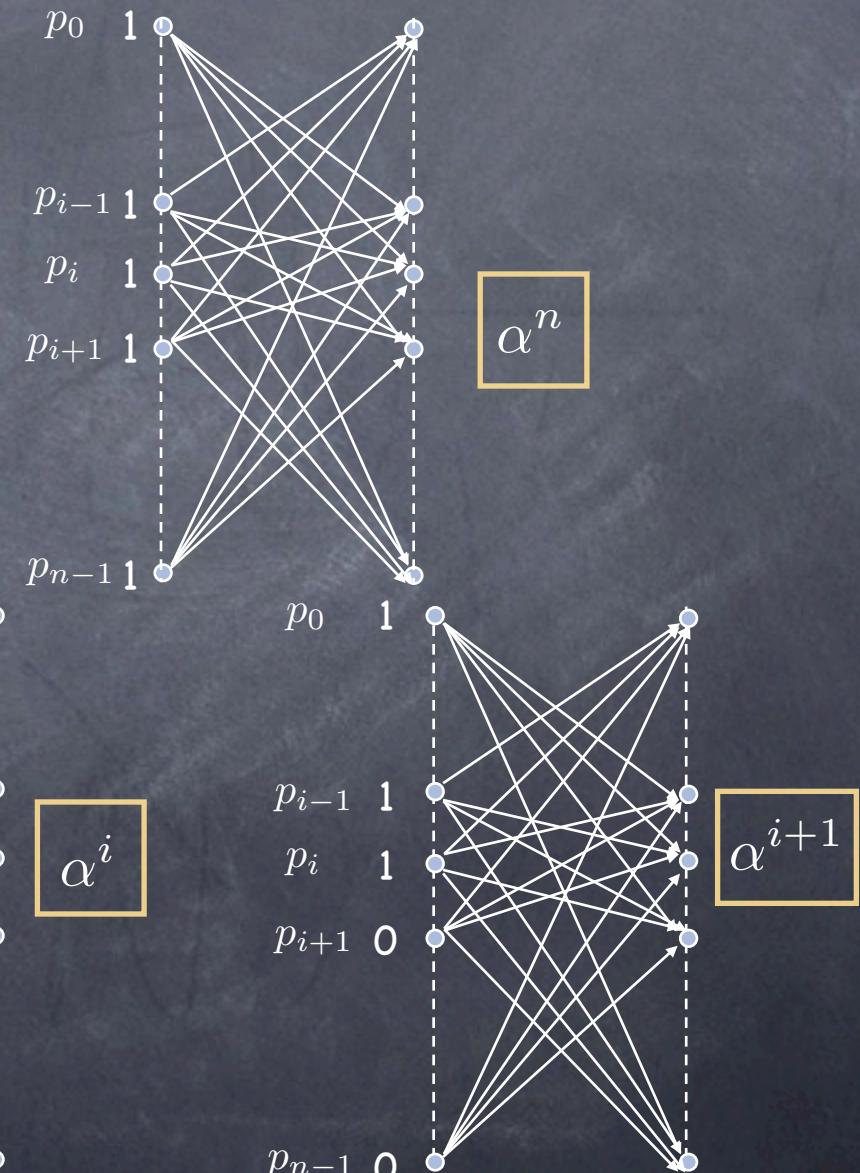
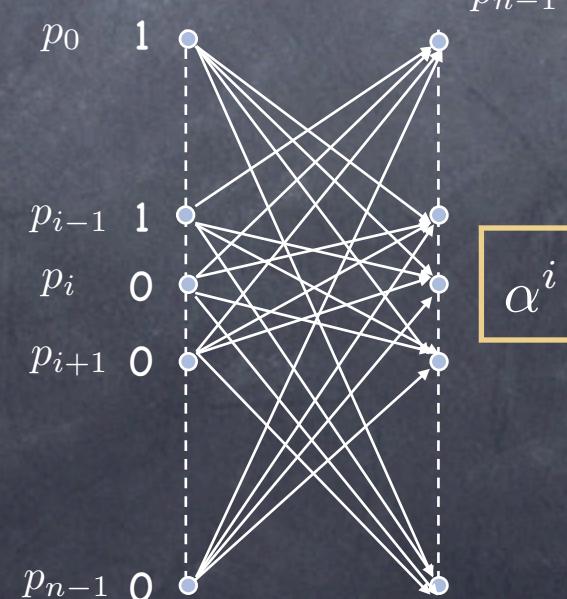
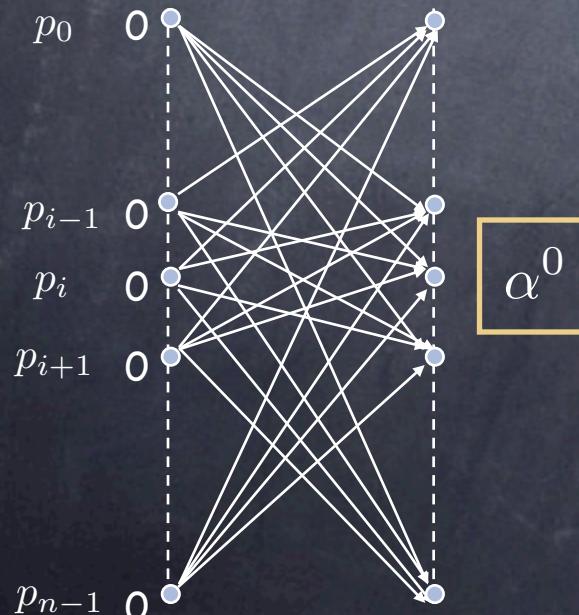
So what?

$\alpha^i s$

Definition

α^i is the execution of the algorithm in which

- no failures occur
- only processes p_0, \dots, p_{i-1} propose 1

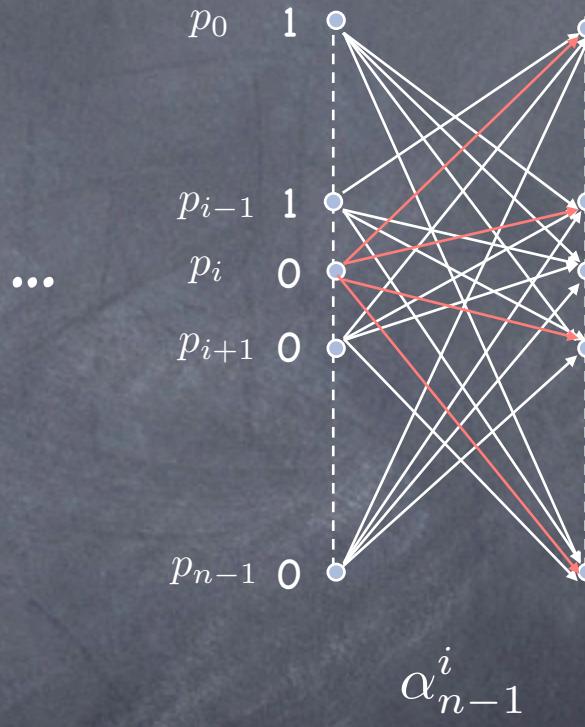
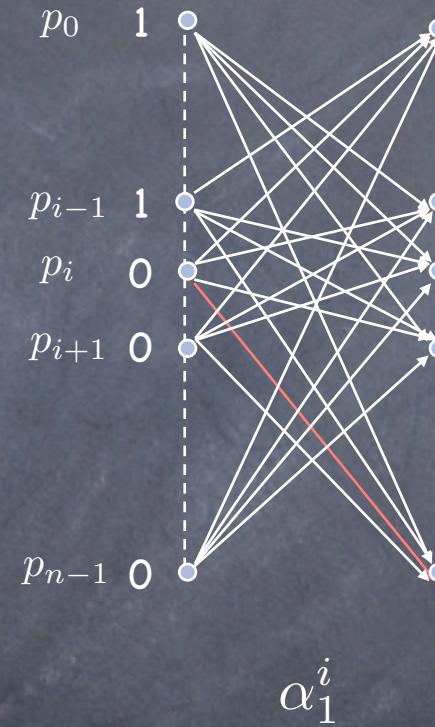
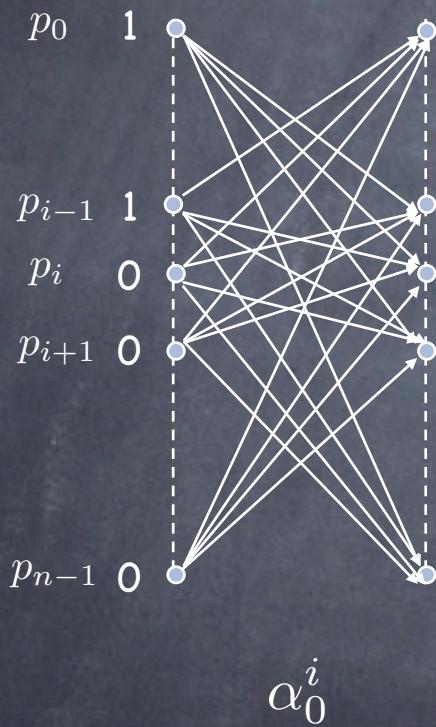


Adjacent α^i 's are similar!

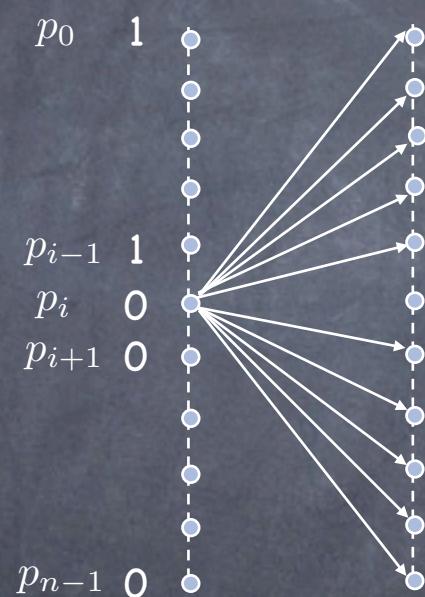
Starting from α^i , we build a set of executions
 α_j^i where $0 \leq j \leq n-1$ as follows:

α_j^i is obtained from α^i after removing the messages that p_i sends to the j -th highest numbered processors (excluding itself)

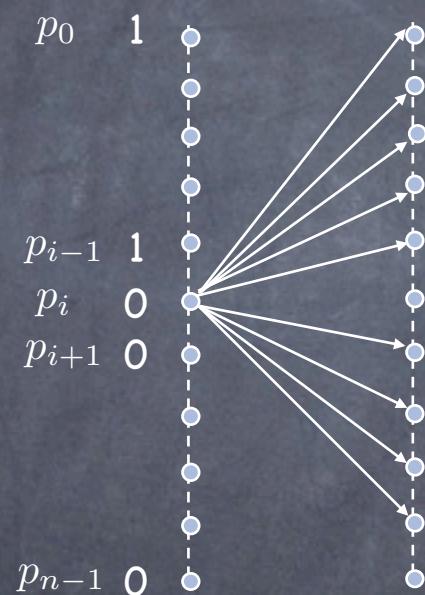
The executions



Indistinguishability

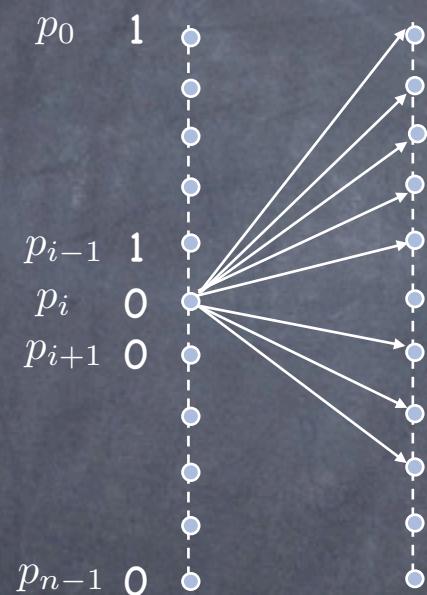
 α^i α_0^i

Indistinguishability



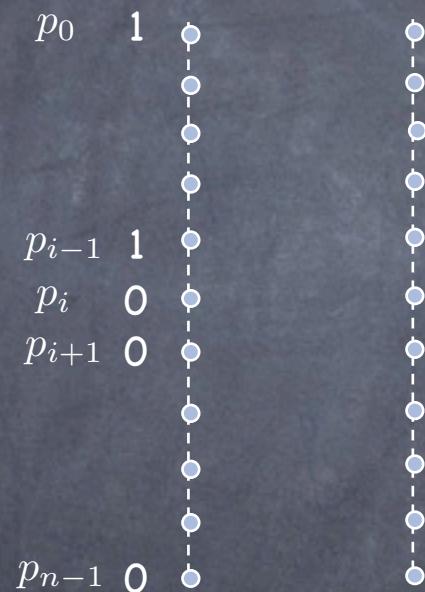
α^i
 α_1^i

Indistinguishability



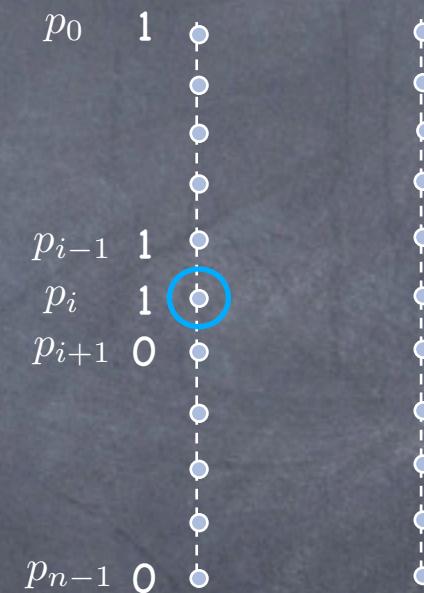
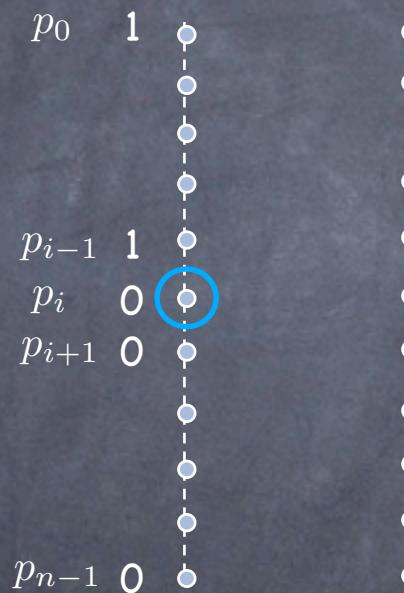
$$\begin{matrix} \alpha^i \\ \Downarrow \\ \alpha_2^i \end{matrix}$$

Indistinguishability



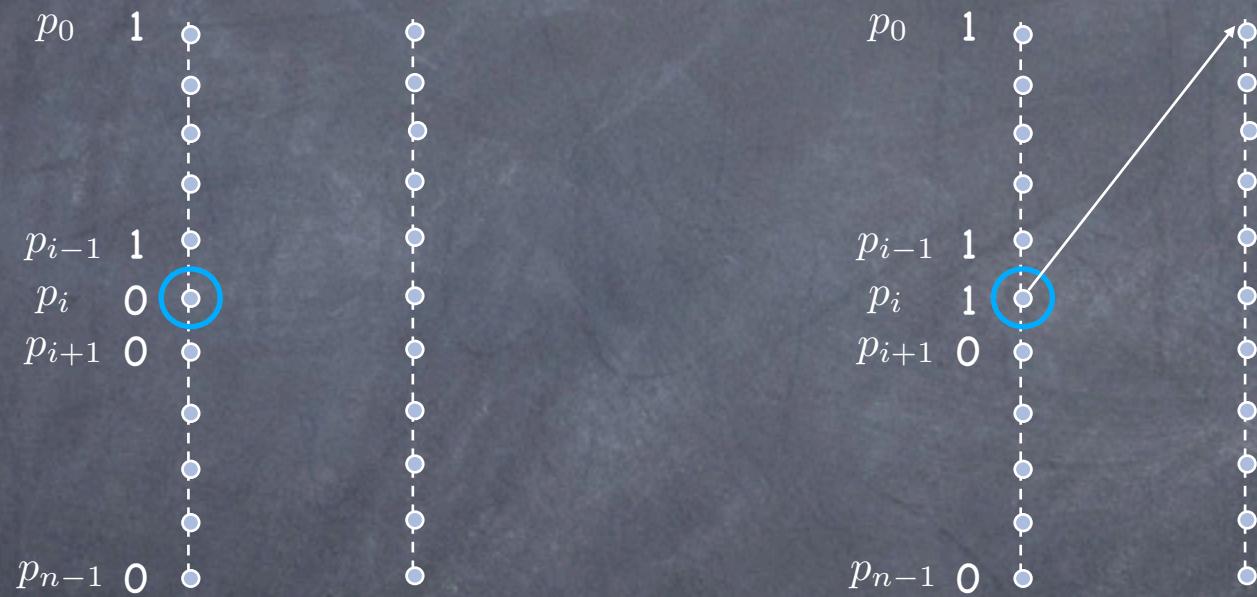
$$\begin{matrix} \alpha^i \\ \Downarrow \\ \alpha_{n-1}^i \end{matrix}$$

Indistinguishability



$$\alpha^i \approx \alpha_{n-1}^i \quad \beta_{n-1}^i$$

Indistinguishability

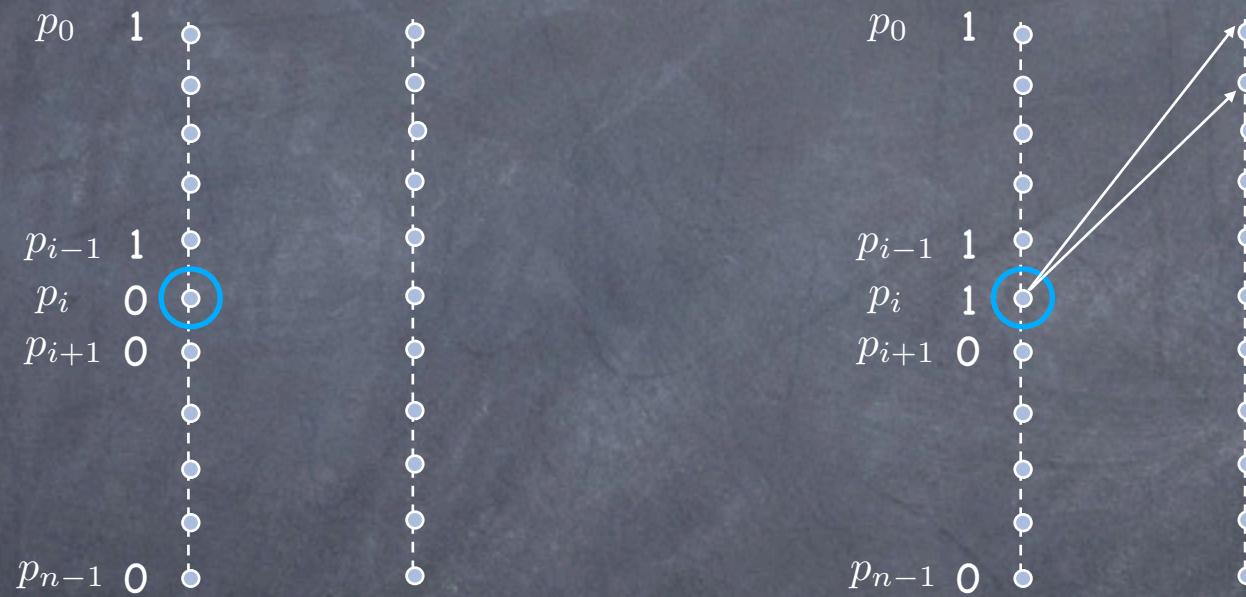


$$\alpha^i \\ \Downarrow \\ \alpha_{n-1}^i$$

\approx

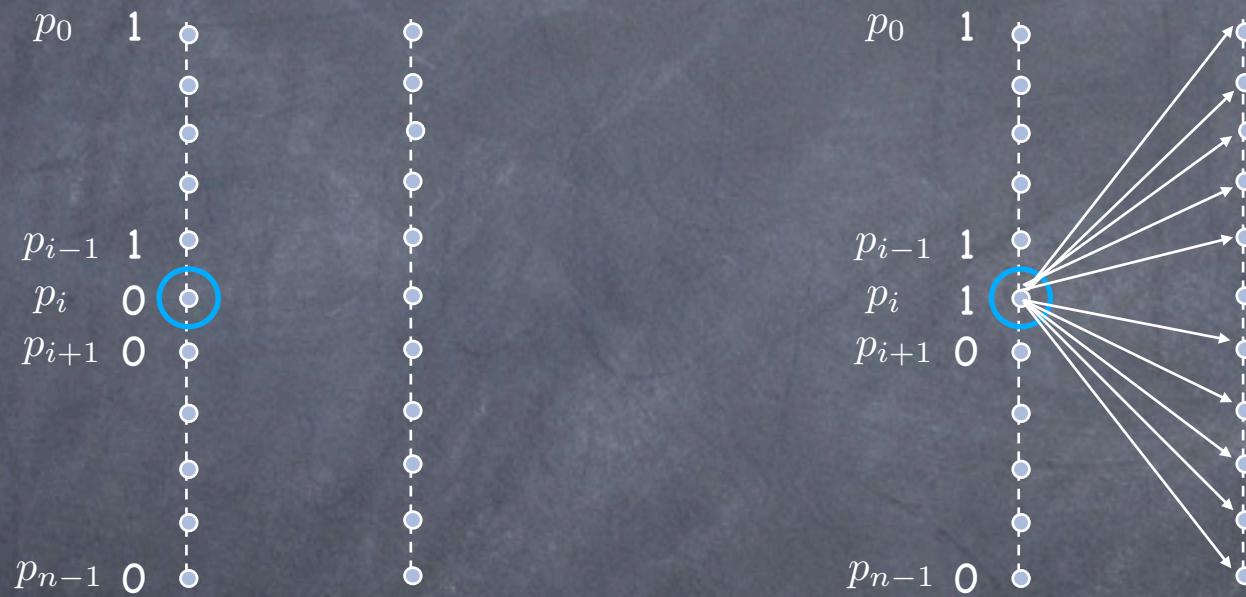
$$\beta_{n-2}^i$$

Indistinguishability



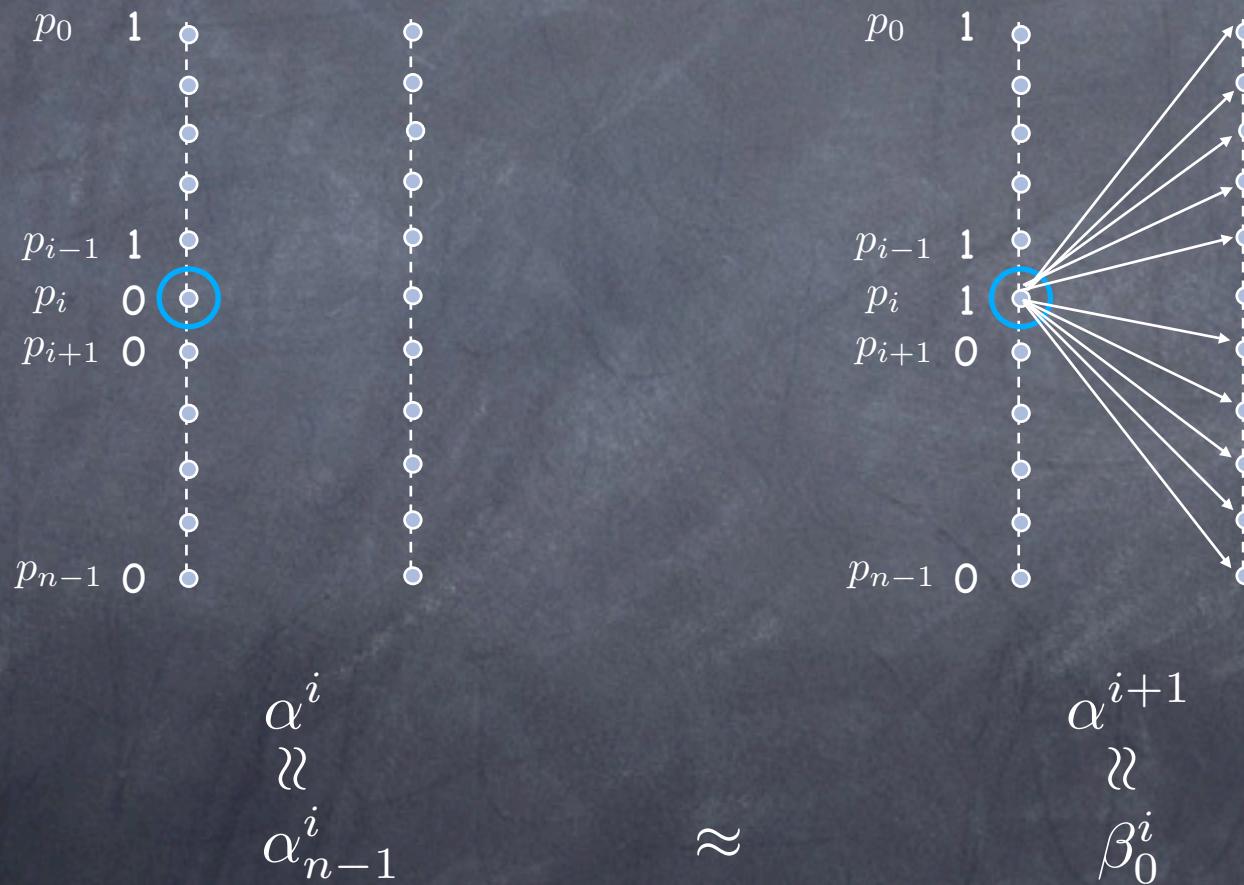
$$\begin{matrix} \alpha^i \\ \Downarrow \\ \alpha^i_{n-1} \end{matrix} \approx \begin{matrix} \beta^i_{n-3} \end{matrix}$$

Indistinguishability

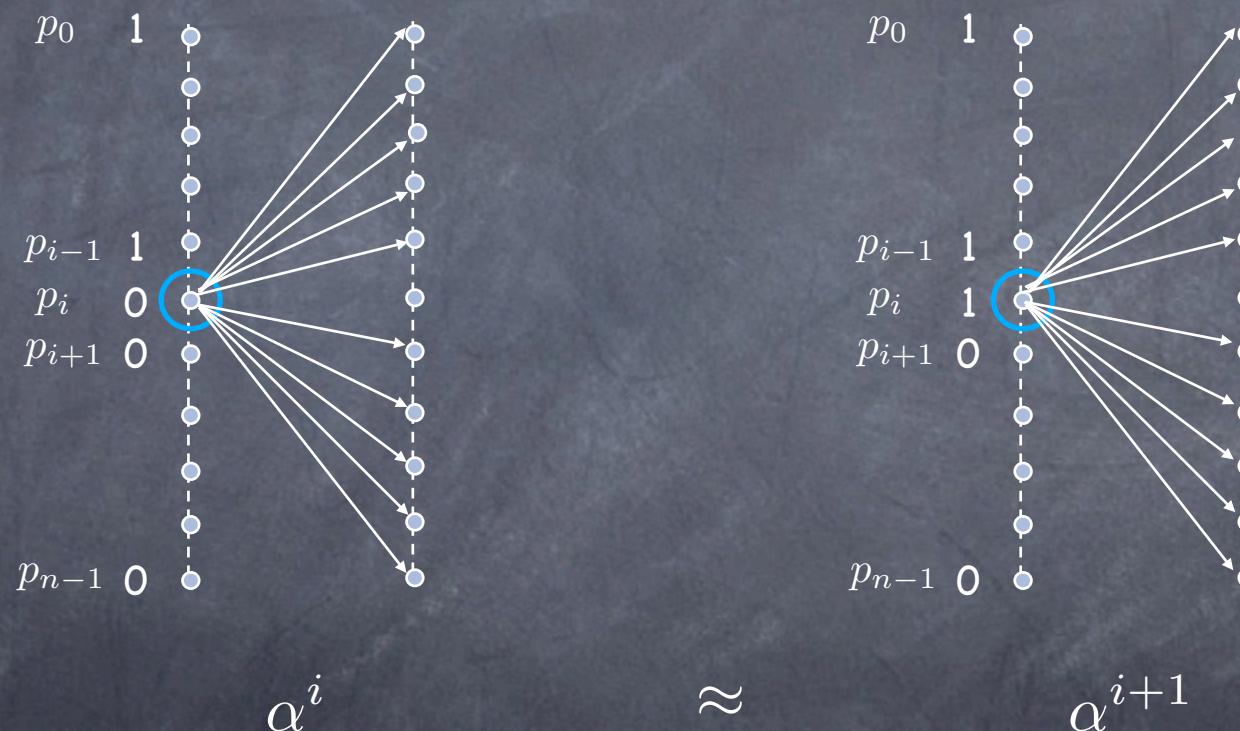


$$\frac{\alpha^i}{\alpha_{n-1}^i} \approx \beta_0^i$$

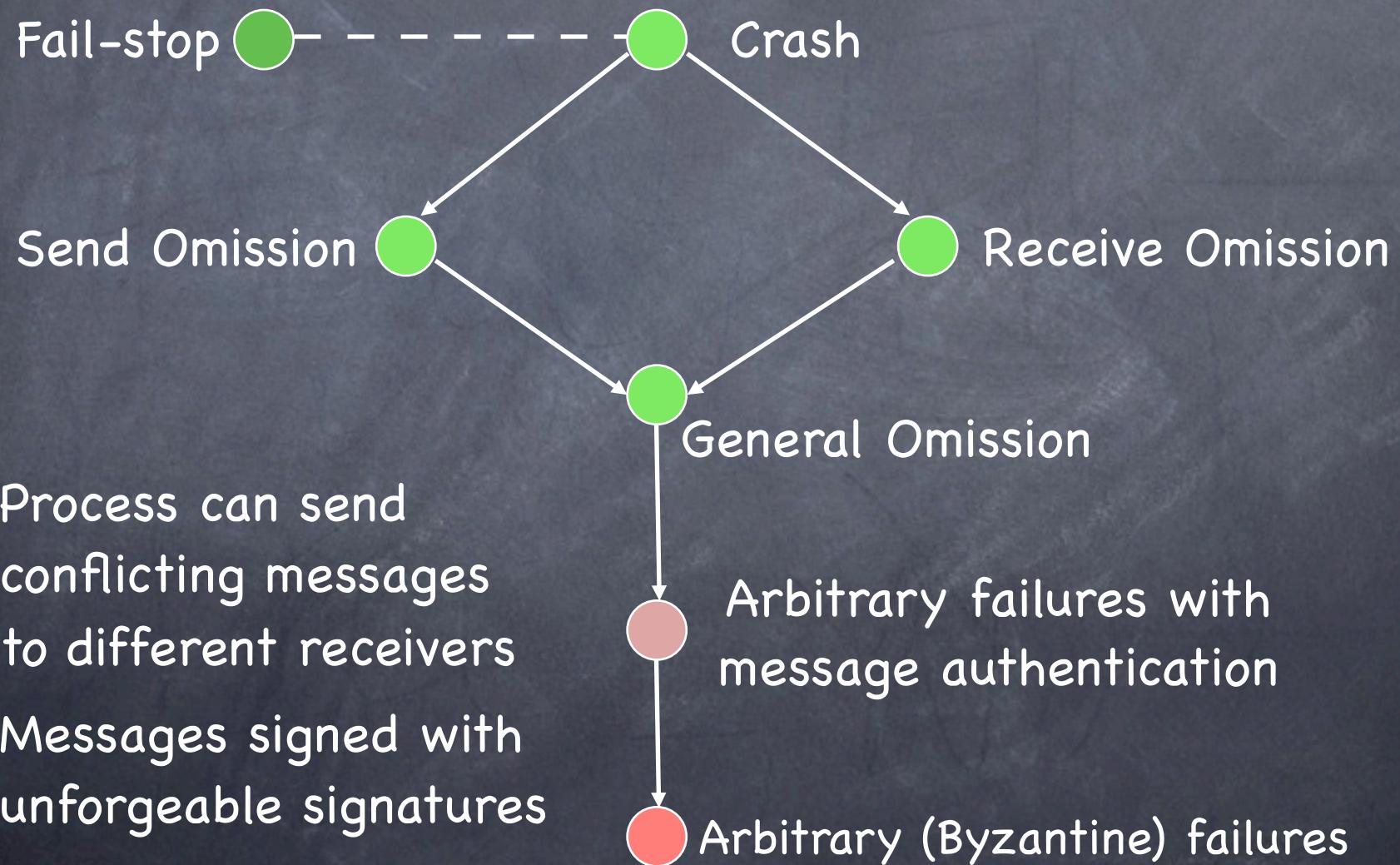
Indistinguishability



Indistinguishability



Arbitrary failures with message authentication



Valid messages

A **valid message** m has the following form:

in round 1:

$m : s_{id}$ (m is signed by the sender)

in round $r > 1$, if received by p from q :

$m : p_1 : p_2 : \dots : p_r$ where

- ➊ $p_1 = \text{sender}; p_r = q$
- ➋ p_1, \dots, p_r are distinct from each other and from p
- ➌ message has not been tampered with

AFMA: The Idea

- ② A correct process p discards all non-valid messages it receives
- ② If a message is valid,
 - it “extracts” the value from the message
 - it relays the message, with its own signature appended
- ② At round $f+1$:
 - if it extracted exactly one message, p delivers it
 - otherwise, p delivers SF

AFMA: The Protocol

Initialization for process p :

```
if  $p$  = sender and  $p$  wishes to broadcast  $m$  then  
    extracted := relay :=  $\{m\}$ 
```

Process p in round $k, 1 \leq k \leq f+1$

for each $s \in \text{relay}$

send $s : p$ to all

receive round k messages from all processes

$\text{relay} := \emptyset$

for each valid message received $s = m : p_1 : p_2 : \dots : p_k$

if $m \notin \text{extracted}$ then

$\text{extracted} := \text{extracted} \cup \{m\}$

$\text{relay} := \text{relay} \cup \{s\}$

At the end of round $f+1$

if $\exists m$ such that $\text{extracted} = \{m\}$ then

deliver m

else deliver SF

Termination

Initialization for process p :

```
if  $p$  = sender and  $p$  wishes to broadcast  $m$  then  
    extracted := relay :=  $\{m\}$ 
```

Process p in round k , $1 \leq k \leq f+1$

```
for each  $s \in \text{relay}$   
    send  $s : p$  to all  
receive round  $k$  messages from all processes  
 $\text{relay} := \emptyset$   
for each valid message received  $s = m : p_1 : p_2 : \dots : p_k$   
if  $m \notin \text{extracted}$  then  
    extracted := extracted  $\cup \{m\}$   
    relay := relay  $\cup \{s\}$ 
```

At the end of round $f+1$

```
if  $\exists m$  such that  $\text{extracted} = \{m\}$  then  
    deliver  $m$   
else deliver SF
```

In round $f+1$, every correct process delivers either m or SF and then halts

Agreement

Initialization for process p :

```
if  $p$  = sender and  $p$  wishes to broadcast  $m$  then  
    extracted := relay :=  $\{m\}$ 
```

Process p in round k , $1 \leq k \leq f+1$

```
for each  $s \in \text{relay}$   
    send  $s : p$  to all  
receive round  $k$  messages from all processes  
 $\text{relay} := \emptyset$   
for each valid message received  $s = m : p_1 : p_2 : \dots : p_k$   
if  $m \notin \text{extracted}$  then  
    extracted := extracted  $\cup \{m\}$   
    relay := relay  $\cup \{s\}$ 
```

At the end of round $f+1$

```
if  $\exists m$  such that  $\text{extracted} = \{m\}$  then  
    deliver  $m$   
else deliver SF
```

Lemma. If a correct process extracts m , then every correct process eventually extracts m

Proof

Let r be the earliest round in which some correct process extracts m . Let that process be p .

- p has received in round r a message

$$m : p_1 : p_2 : \dots : p_r$$

- If $r \leq f$, p will send a valid message

$$m : p_1 : p_2 : \dots : p_r : p$$

in round $r+1 \leq f+1$ and every correct process will extract it in round $r+1 \leq f+1$

- What if $r = f+1$?

- Claim: p_1, p_2, \dots, p_{f+1} are all faulty

- true for $p_1 = s$

- Suppose $p_j, 1 < j \leq f+1$, were correct

- p_j signed and relayed message in round j
- p_j extracted message in round $j-1 < f+1$
- $f+1$ was supposed to be earliest round where a correct process extracted m

CONTRADICTION

- At most f faulty processes
- CONTRADICTION

Validity

Initialization for process p :

```
if  $p$  = sender and  $p$  wishes to broadcast  $m$  then  
    extracted := relay :=  $\{m\}$ 
```

Process p in round k , $1 \leq k \leq f+1$

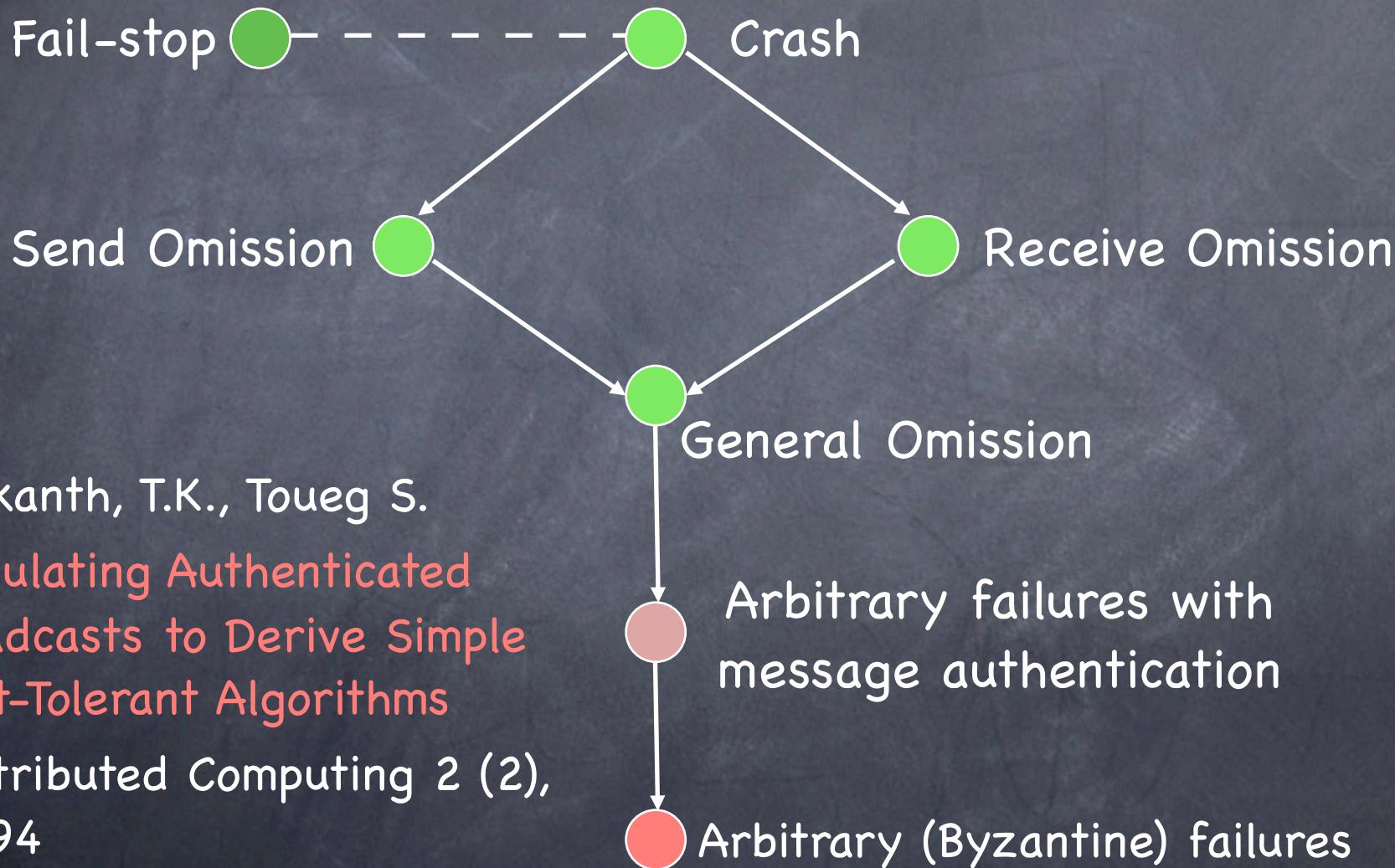
```
for each  $s \in \text{relay}$   
    send  $s : p$  to all  
receive round  $k$  messages from all processes  
relay :=  $\emptyset$   
for each valid message received  $s = m : p_1 : p_2 : \dots : p_k$   
if  $m \notin \text{extracted}$  then  
    extracted := extracted  $\cup \{m\}$   
    relay := relay  $\cup \{s\}$ 
```

At the end of round $f+1$

```
if  $\exists m$  such that  $\text{extracted} = \{m\}$  then  
    deliver  $m$   
else deliver SF
```

From Agreement and the observation that the sender, if correct, delivers its own message.

TRB for arbitrary failures



Srikanth, T.K., Toueg S.

Simulating Authenticated
Broadcasts to Derive Simple
Fault-Tolerant Algorithms

Distributed Computing 2 (2),
80-94

AF: The Idea

- ② Identify the essential properties of message authentication that made AFMA work
- ③ Implement these properties without using message authentication

AF: The Approach

- ➊ Introduce two primitives
 - $\text{broadcast}(p, m, i)$ (executed by p in round i)
 - $\text{accept}(p, m, i)$ (executed by q in round $j \geq i$)
- ➋ Give axiomatic definitions of broadcast and accept
- ➌ Derive an algorithm that solves TRB for AF using these primitives
- ➍ Show an implementation of these primitives that does not use message authentication

Properties of broadcast and accept

- ② **Correctness** If a correct process p executes $\text{broadcast}(p, m, i)$ in round i , then all correct processes will execute $\text{accept}(p, m, i)$ in round i
- ③ **Unforgeability** If a correct process q executes $\text{accept}(p, m, i)$ in round $j \geq i$, and p is correct, then p did in fact execute $\text{broadcast}(p, m, i)$ in round i
- ④ **Relay** If a correct process q executes $\text{accept}(p, m, i)$ in round $j \geq i$, then all correct processes will execute $\text{accept}(p, m, i)$ by round $j+1$

AF: The Protocol - 1

sender s in round 0:

0: extract m

sender s in round 1:

1: broadcast($s, m, 1$)

Process p in round $k, 1 \leq k \leq f+1$

2: if p extracted m in round $k-1$ and $p \neq$ sender then

4: broadcast(p, m, k)

5: if p has executed at least k accept(q_i, m, j_i) $1 \leq i \leq k$ in rounds 1 through k
(where (i) q_i distinct from each other and from p , (ii) one q_i is s , and
(iii) $1 \leq j_i \leq k$) and p has not previously extracted m then

6: extract m

7: if $k = f+1$ then

8: if in the entire execution p has extracted exactly one m then

9: deliver m

10: else deliver SF

11: halt

Termination

sender s in round 0:

0: extract m

sender s in round 1:

1: broadcast($s, m, 1$)

Process p in round $k, 1 \leq k \leq f+1$

- 2: if p extracted m in round $k-1$ and $p \neq$ sender then
- 4: broadcast(p, m, k)
- 5: if p has executed at least k accept(q_i, m, j_i) $1 \leq i \leq k$ in rounds 1 through k
 (where (i) q_i distinct from each other and from
 , (ii) one q_i , and (iii) $1 \leq j_i \leq k$
 and p has not previously extracted m then)
- 6: extract m
- 7: if $k = f+1$ then
- 8: if in the entire execution p has extracted exactly
 one m then
- 9: deliver m
- 10: else deliver SF
- 11: halt

In round $f+1$, every correct process delivers either m or SF and then halts

Agreement – 1

sender s in round 0:

0: extract m

sender s in round 1:

1: broadcast($s, m, 1$)

Process p in round $k, 1 \leq k \leq f+1$

2: if p extracted m in round $k-1$ and $p \neq$ sender then

4: broadcast(p, m, k)

5: if p has executed at least k accept(q_i, m, j_i) $1 \leq i \leq k$ in
rounds 1 through k

(where (i) q_i distinct from each other and from
, (ii) one is q_i , and (iii) $1 \leq j_i \leq k$

and p has not previously extracted m then

6: extract m

7: if $k = f+1$ then

8: if in the entire execution p has extracted exactly
one then

9: deliver m

10: else deliver SF

11: halt

Lemma

If a correct process extracts m , then
every correct process eventually extracts m

Agreement - 1

sender s in round 0:

0: extract m

sender s in round 1:

1: broadcast($s, m, 1$)

Process p in round k , $1 \leq k \leq f+1$

2: if p extracted m in round $k-1$ and $p \neq$ sender then
4: broadcast(p, m, k)
5: if p has executed at least k accept(q_i, m, j_i) $1 \leq i \leq k$ in
 rounds 1 through k
 (where (i) q_i distinct from each other and from
 , (ii) one q_i is s , and (iii) $1 \leq j_i \leq k$
 and p has not previously extracted m then
6: extract m
7: if $k = f+1$ then
8: if in the entire execution p has extracted exactly
 one m then
9: deliver m
10: else deliver SF
11: halt

Proof

Let r be the earliest round in which some correct process extracts m . Let that process be p .

• if $r=0$, then $p=s$ and p will execute broadcast($s, m, 1$) in round 1. By CORRECTNESS, all correct processes will execute accept($s, m, 1$) in round 1 and extract m

Lemma

If a correct process extracts m , then
every correct process eventually extracts m

Agreement - 1

sender s in round 0:

0: extract m

sender s in round 1:

1: broadcast($s, m, 1$)

Process p in round k , $1 \leq k \leq f+1$

- 2: if p extracted m in round $k-1$ and $p \neq$ sender then
- 4: broadcast(p, m, k)
- 5: if p has executed at least k accept(q_i, m, j_i) $1 \leq i \leq k$ in rounds 1 through k
 - (where (i) q_i distinct from each other and from , (ii) one is q_i , and (iii) $1 \leq j_i \leq k$
- and p has not previously extracted m then
- 6: extract m
- 7: if $k = f+1$ then
- 8: if in the entire execution p has extracted exactly one then
- 9: deliver m
- 10: else deliver SF
- 11: halt

Proof

Let r be the earliest round in which some correct process extracts m . Let that process be p .

- ① if $r=0$, then $p=s$ and p will execute broadcast($s, m, 1$) in round 1. By CORRECTNESS, all correct processes will execute accept($s, m, 1$) in round 1 and extract m
- ② if $r > 0$, then the sender is faulty. Since r has extracted in round r , has accepted at least triples with properties (i), (ii), and (iii) by round $r \leq f$
 - By RELAY, all correct processes will have accepted those triples by round $r+1$
 - will execute broadcast($p, m, r+1$) in round $r+1$
 - By CORRECTNESS, any correct process other than (q_k, m, j_k), $1 \leq j_k \leq r+1$ will have accepted triples , by round $r+1$

Lemma

If a correct process extracts m , then every correct process eventually extracts m

Agreement - 1

sender s in round 0:

0: extract m

sender s in round 1:

1: broadcast($s, m, 1$)

Process p in round k , $1 \leq k \leq f+1$

- 2: if p extracted m in round $k-1$ and $p \neq$ sender then
- 4: broadcast(p, m, k)
- 5: if p has executed at least k accept(q_i, m, j_i) $1 \leq i \leq k$ in rounds 1 through k
 (where (i) q_i distinct from each other and from
 , (ii) one q_i is q_i , and (iii) $1 \leq j_i \leq k$
 and p has not previously extracted m then
- 6: extract m
- 7: if $k = f+1$ then
- 8: if in the entire execution p has extracted exactly
 one m then
- 9: deliver m
- 10: else deliver SF
- 11: halt

Proof

Let r be the earliest round in which some correct process extracts m . Let that process be p .

- ① if $r=0$, then $p=s$ and p will execute broadcast($s, m, 1$) in round 1. By CORRECTNESS, all correct processes will execute accept($s, m, 1$) in round 1 and extract m
- ② if $r > 0$, then the sender is faulty. Since r has extracted m in round r , p has accepted at least r triples with properties (i), (ii), and (iii) by round $r \leq f$
 - By RELAY, all correct processes will have accepted those triples by round $r+1$
 - p will execute broadcast($p, m, r+1$) in round $r+1$
 - By CORRECTNESS, any correct process other than (q_k, m, j_k) , $1 \leq j_k \leq r+1$ will have accepted m triples by round $r+1$

Lemma

If a correct process extracts m , then every correct process eventually extracts m

Agreement - 1

sender s in round 0:

0: extract m

sender s in round 1:

1: broadcast($s, m, 1$)

Process p in round k , $1 \leq k \leq f+1$

- 2: if p extracted m in round $k-1$ and $p \neq$ sender then
- 4: broadcast(p, m, k)
- 5: if p has executed at least k accept(q_i, m, j_i) $1 \leq i \leq k$ in rounds 1 through k
 - (where (i) q_i distinct from each other and from , (ii) one is q_i , and (iii) $1 \leq j_i \leq k$
- and p has not previously extracted m then
- 6: extract m
- 7: if $k = f+1$ then
- 8: if in the entire execution p has extracted exactly one then
- 9: deliver m
- 10: else deliver SF
- 11: halt

Proof

Let r be the earliest round in which some correct process extracts m . Let that process be p .

- if $r=0$, then $p=s$ and p will execute broadcast($s, m, 1$) in round 1. By CORRECTNESS, all correct processes will execute accept($s, m, 1$) in round 1 and extract m by round $r > 0$
- if $r > 0$, then the sender is faulty. Since r has extracted m in round r , p has accepted at least r triples with properties (i), (ii), and (iii) by round $r \leq f$
 - By RELAY, all correct processes will have accepted those triples by round $r+1$
 - p will execute broadcast($p, m, r+1$) in round $r+1$
 - By CORRECTNESS, any correct process other than (q_k, m, j_k), $1 \leq j_k \leq r+1$ will have accepted m triples q_1, q_2, \dots, q_r, p by round $r+1$

Lemma

If a correct process extracts m , then every correct process eventually extracts m

Agreement - 1

sender s in round 0:

0: extract m

sender s in round 1:

1: broadcast($s, m, 1$)

Process p in round k , $1 \leq k \leq f+1$

- 2: if p extracted m in round $k-1$ and $p \neq$ sender then
- 4: broadcast(p, m, k)
- 5: if p has executed at least k accept(q_i, m, j_i) $1 \leq i \leq k$ in rounds 1 through k
 - (where (i) q_i distinct from each other and from , (ii) one is q_i , and (iii) $1 \leq j_i \leq k$
- and p has not previously extracted m then
- 6: extract m
- 7: if $k = f+1$ then
- 8: if in the entire execution p has extracted exactly one then
- 9: deliver m
- 10: else deliver SF
- 11: halt

Proof

Let r be the earliest round in which some correct process extracts m . Let that process be p .

- ① if $r=0$, then $p=s$ and p will execute broadcast($s, m, 1$) in round 1. By CORRECTNESS, all correct processes will execute accept($s, m, 1$) in round 1 and extract m by round $r > 0$
- ② if $r > 0$, then the sender is faulty. Since r has extracted m in round r , p has accepted at least r triples with properties (i), (ii), and (iii) by round $r \leq f$
 - By RELAY; all correct processes will have accepted those triples by round 1
 - will execute broadcast($p, m, r+1$) in round $r+1$
 - By CORRECTNESS, any correct process other than (q_k, m, j_k), $1 \leq j_k \leq r+1$ will have accepted triples q_1, q_2, \dots, q_r, p by round m
 - every correct process other than q_1, q_2, \dots, q_r will extract m
 - already extracted m ; what about ?

Lemma

If a correct process extracts m , then every correct process eventually extracts m

Agreement – 2

sender s in round 0:

0: extract m

sender s in round 1:

1: broadcast($s, m, 1$)

Process p in round k , $1 \leq k \leq f+1$

2: if p extracted m in round $k-1$ and $p \neq$ sender then

4: broadcast(p, m, k)

5: if p has executed at least k accept(q_i, m, j_i) $1 \leq i \leq k$ in rounds 1 through k

(where (i) q_i distinct from each other and from
, (ii) one is q_i , and (iii) $1 \leq j_i \leq k$)

and p has not previously extracted m then

6: extract m

7: if $k = f+1$ then

8: if in the entire execution p has extracted exactly one then

9: deliver m

10: else deliver SF

11: halt

Claim: q_1, q_2, \dots, q_r are all faulty

> Suppose q_k were correct

> p has accepted (q_k, m, j_k) in round $j_k \leq r$

> By UNFORGEABILITY, q_k executed broadcast (q_k, m, j_k) in round j_k

> correct q_k extracted m in round $j_{k-1} < r$

CONTRADICTION

□ **Case 2:** $r = f+1$

□ Since there are at most f faulty processes, some process q_l in q_1, q_2, \dots, q_{f+1} is correct

□ By UNFORGEABILITY, q_l executed broadcast (q_l, m, j_l) in round $j_l \leq r$

□ q_l has extracted m in round $j_{l-1} < f+1$

CONTRADICTION

Validity

sender s in round 0:

0: extract m

sender s in round 1:

1: broadcast($s, m, 1$)

Process p in round k , $1 \leq k \leq f+1$

2: if p extracted m in round $k-1$ and $p \neq$ sender then
4: broadcast(p, m, k)
5: if p has executed at least k accept(q_i, m, j_i) $1 \leq i \leq k$ in
 rounds 1 through k
 (where (i) q_i distinct from each other and from
 , (ii) one q_i and (iii) $1 \leq j_i \leq k$
 and p has not previously extracted m then
6: extract m
7: if $k = f+1$ then
8: if in the entire execution p has extracted exactly
 one m then
9: deliver m
10: else deliver SF
11: halt

- ④ A correct sender executes broadcast($s, m, 1$) in round 1
- ④ By CORRECTNESS, all correct processes execute accept($s, m, 1$) in round 1 and extract m
- ④ In order to extract a different message m' , a process must execute accept() in some round $i \leq f + 1$
- ④ By UNFORGEABILITY, and because s is correct, no correct process can extract $m' \neq m$
- ④ All correct processes will deliver m

Implementing broadcast and accept

- ➊ A process that wants to broadcast m , does so through a series of **witnesses**
 - Sends m to all
 - Each correct process becomes a witness by relaying m to all
- ➋ If a process receives enough witness confirmations, it accepts m

Can we rely on witnesses?

- ➊ Only if not too many faulty processes!
- ➋ Otherwise, a set of faulty processes could fool a correct process by acting as witnesses of a message that was never broadcast
- ➌ How large can be f with respect to n ?

Byzantine Generals

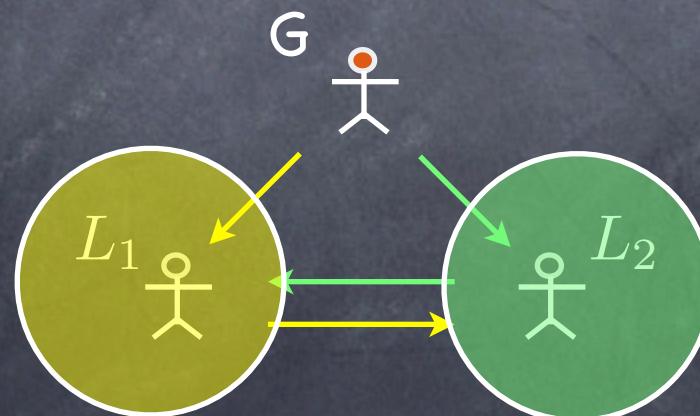
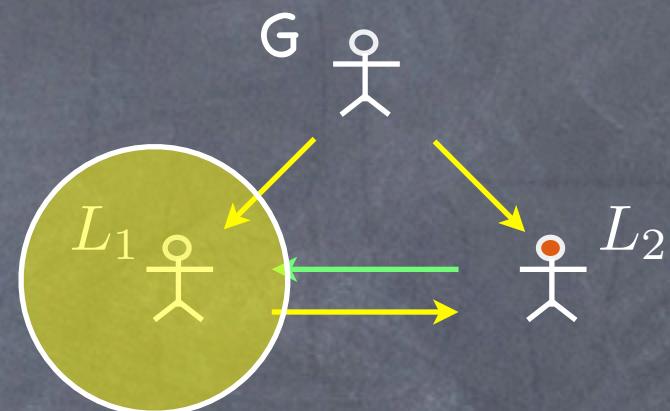
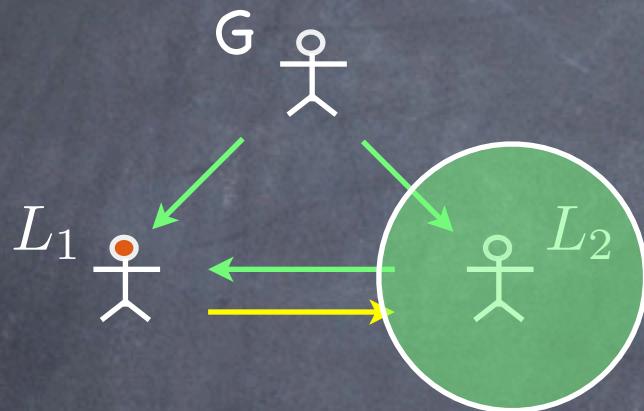
- One General G , a set of Lieutenants L_i
- General can order Attack (**A**) or Retreat (**R**)
- General may be a traitor; so may be some of the Lieutenants

* * *

- I. If G is trustworthy, every trustworthy L_i must follow G 's orders
- II. Every trustworthy L_i must follow same battleplan

The plot thickens...

One traitor



A Lower Bound

Theorem

There is no algorithm that solves TRB for Byzantine failures if $n \leq 3f$

(Lamport, Shostak, and Pease, The Byzantine Generals Problem, ACM TOPLAS, 4 (3), 382-401, 1982)

Back to the protocol...

- ➊ To broadcast a message in round r , p sends $(init, p, m, r)$ to all
- ➋ A confirmation has the form $(echo, p, m, r)$
- ➌ A witness sends $(echo, p, m, r)$ if either:
 - it receives $(init, p, m, r)$ from p directly or
 - it receives confirmations for (p, m, r) from at least $f + 1$ processes (at least one correct witness)
- ➍ A process accepts (p, m, r) if it has received $n - f$ confirmations (as many as possible...)
- ➎ Protocol proceeds in rounds. Each round has 2 phases

Implementation of broadcast and accept

Phase $2r-1$

1: p sends $(init, p, m, r)$ to all

Phase $2r$

2: if q received $(init, p, m, r)$ in phase $2r-1$ then

3: q sends $(echo, p, m, r)$ to all /* q becomes a witness */

4: if q receives $(echo, p, m, r)$ from at least $n-f$ distinct processes in phase $2r$ then

5: q accepts (p, m, r)

Phase $j > 2r$

6: if q has received $(echo, p, m, r)$ from at least $f+1$ distinct processes in phases $(2r+1, \dots, j-1)$ then

7: q sends $(echo, p, m, r)$ to all processes /* q becomes a witness */

8: if q has received $(echo, p, m, r)$ from at least $n-f$ processes in phases $(2r, 2r+1, \dots, j)$ then

9: q accepts (p, m, r)

Is termination a problem?

The implementation is correct

Theorem

If $n > 3f$, the given implementation of $\text{broadcast}(p, m, r)$ and $\text{accept}(p, m, r)$ satisfies Unforgeability, Correctness, and Relay

Assumption

Channels are authenticated

Correctness

If a correct process p
executes $\text{broadcast}(p, m, r)$
in round r , then all
correct processes will
execute $\text{accept}(p, m, r)$ in
round r

Correctness

If a correct process p executes $\text{broadcast}(p, m, r)$ in round r , then all correct processes will execute $\text{accept}(p, m, r)$ in round r

If p is correct then

- p sends (init, p, m, r) to all in round r (phase $2r-1$)
- by Validity of the underlying send and receive, every correct process receives (init, p, m, r) in phase $2r-1$
- every correct process becomes a witness
- every correct process sends (echo, p, m, r) in phase $2r$
- since there are at least $n-f$ correct processes, every correct process receives at least $n-f$ echoes in phase $2r$
- every correct process executes $\text{accept}(p, m, r)$ in phase $2r$ (in round r)

Unforgeability – 1

If a correct process q executes $\text{accept}(p, m, r)$ in round $j \geq r$, and p is correct, then p did in fact execute $\text{broadcast}(p, m, r)$ in round r

- Suppose q executes $\text{accept}(p, m, r)$ in round j
- q received (echo, p, m, r) from at least $n - f$ distinct processes by phase k , where $k = 2j - 1$ or $k = 2j$
- Let k' be the earliest phase in which some correct process q' becomes a witness to (p, m, r)

Unforgeability – 1

If a correct process q executes $\text{accept}(p, m, r)$ in round $j \geq r$, and p is correct, then p did in fact execute $\text{broadcast}(p, m, r)$ in round r

- Suppose q executes $\text{accept}(p, m, r)$ in round j
- q received (echo, p, m, r) from at least $n - f$ distinct processes by phase k , where $k = 2j - 1$ or $k = 2j$
- Let k' be the earliest phase in which some correct process q' becomes a witness to (p, m, r)

Case 1: $k' = 2r - 1$

- q' received (init, p, m, r) from p
- since p is correct, it follows that p did execute $\text{broadcast}(p, m, r)$ in round r

Case 2: $k' > 2r - 1$

- q' has become a witness by receiving (echo, p, m, r) from $f + 1$ distinct processes
- at most f are faulty; one is correct
- this process was a witness to (p, m, r) before phase

CONTRADICTION

The first correct process receives (init, p, m, r) from p !

Unforgeability -2

- For q to accept, some correct process must become witness.
- Earliest correct witness q' becomes so in phase $2r - 1$, and only if p did indeed executed $\text{broadcast}(p, m, r)$
- Any correct process that becomes a witness later can only do so if a correct process is already a witness.
- For any correct process to become a witness, p must have executed $\text{broadcast}(p, m, r)$

Relay

If a correct process q
executes $\text{accept}(p, m, r)$ in
round $j \geq r$, then all
correct processes will
execute $\text{accept}(p, m, r)$ by
round $j + 1$

Relay

If a correct process q executes $\text{accept}(p, m, r)$ in round $j \geq r$, then all correct processes will execute $\text{accept}(p, m, r)$ by round $j + 1$

- Suppose correct q executes $\text{accept}(p, m, r)$ in round j (phase $k = 2j - 1$ or $k = 2j$)
- q received at least $n - f$ (echo, p, m, r) from distinct processes by phase k
- At least $n - 2f$ of them are correct.
- All correct procs received (echo, p, m, r) from at least $n - 2f$ correct processes by phase k
- From $n > 3f$, it follows that $n - 2f \geq f + 1$. Then, all correct processes become witnesses by phase k
- All correct processes send (echo, p, m, r) by phase $k + 1$
- Since there are at least $n - f$ correct processes, all correct processes will accept (p, m, r) by phase $k + 1$ (round $2j$ or $2j + 1$)

What about asynchronous systems?

- ➊ FLP says consensus cannot be solved...
- ➋ For benign failures, Paxos provides next best thing
 - always safe
 - live during periods of sufficient synchrony
- ➌ And for Byzantine failures?