

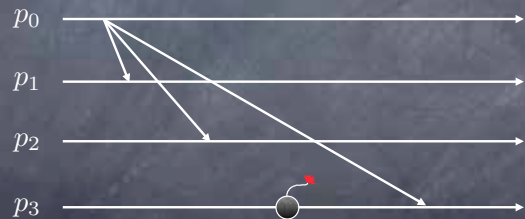
Consensus and Reliable Broadcast

Broadcast

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

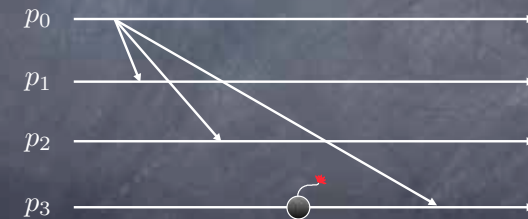
Broadcast

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



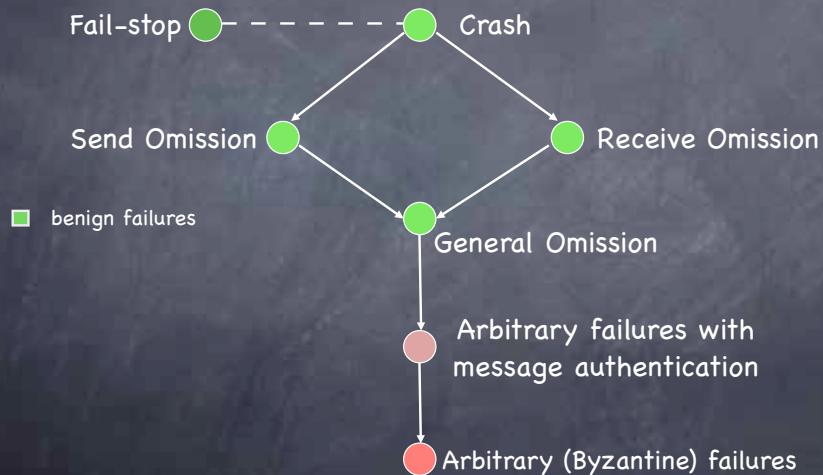
Broadcast

- 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 send(m) and 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...

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

Plan

- Benign Synchronous Systems
 - Consensus for synchronous systems with crash failures
 - Lower bound on the number of rounds
- Benign Asynchronous Systems
 - Impossibility of Consensus for crash failures
 - Failure detectors
 - PAXOS
- Byzantine (Synchronous and Asynchronous)
 - Reliable Broadcast for arbitrary failures
 - PBFT, Zyzzyva

Model

- 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
- Network is fully connected (an n -clique)
- 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(x) occurs as follows:

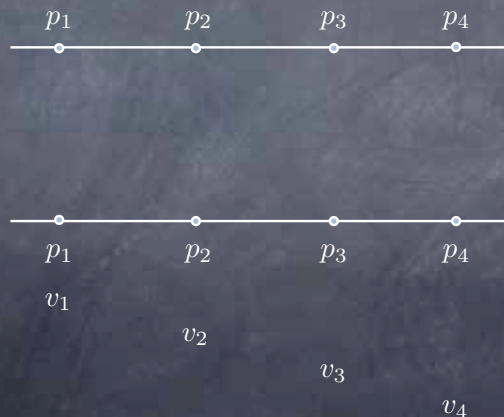
2: for all $j, 0 \leq j \leq n-1, j \neq i$ do

3: receive S_j from p_j

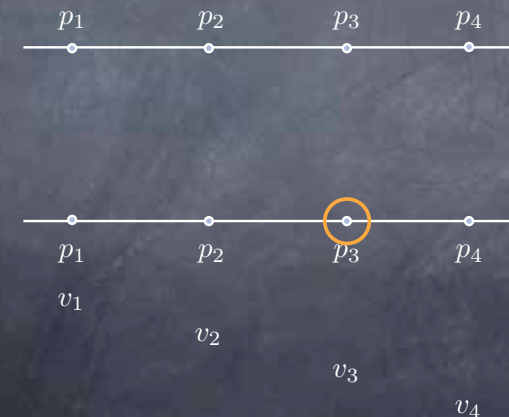
4: $V := V \cup S_j$

5: decide $\min(V)$

An execution



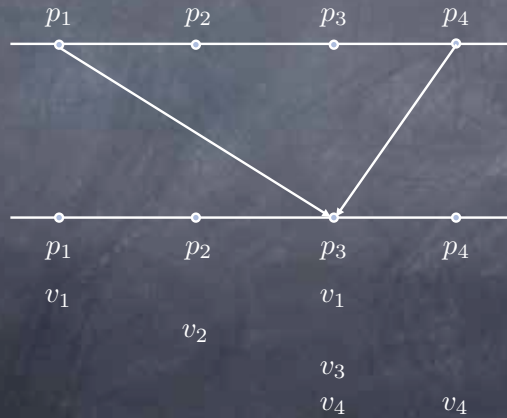
An execution



An execution

Suppose $v_1 = v_3 = v_4$ at the end of round 1

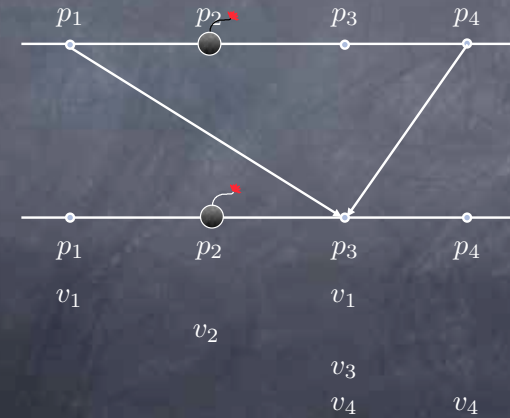
Can p_2 decide?



An execution

Suppose $v_1 = v_3 = v_4$ at the end of round 1

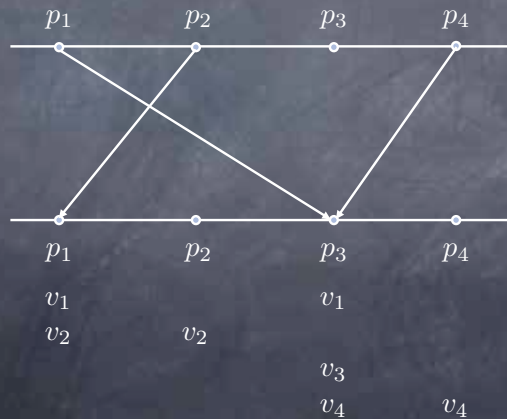
Can p_2 decide?



An execution

Suppose $v_1 = v_3 = v_4$ at the end of round 1

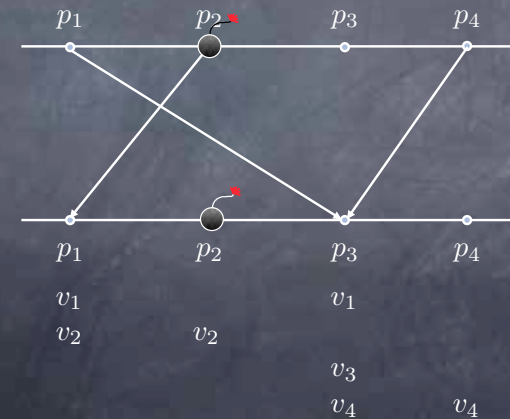
Can p_2 decide?



An execution

Suppose $v_1 = v_3 = v_4$ at the end of round 1

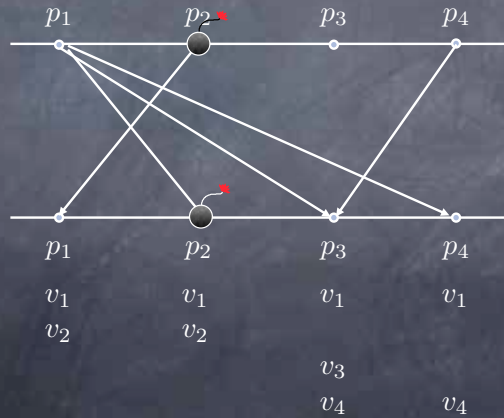
Can p_2 decide?



An execution

Suppose $v_1 = v_3 = v_4$ at the end of round 1

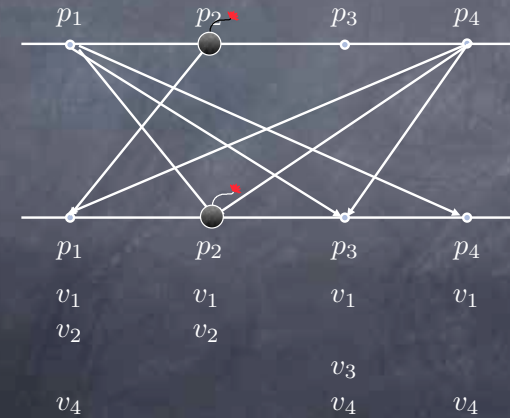
Can p_2 decide?



An execution

Suppose $v_1 = v_3 = v_4$ at the end of round 1

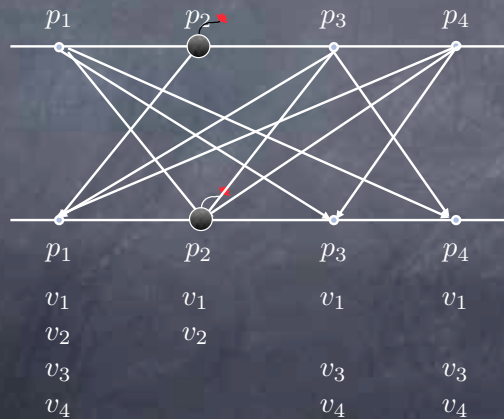
Can p_2 decide?



An execution

Suppose $v_1 = v_3 = v_4$ at the end of round 1

Can p_2 decide?



Echoing values

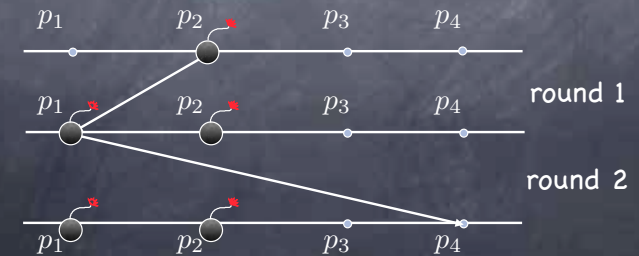
- ⦿ A process that receives a proposal in round 1, relays it to others during round 2.

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?

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?



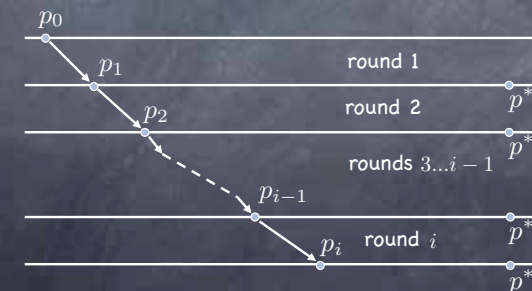
What is going on

- ⦿ 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 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 `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: $V := V \cup S_j$

`decide(x)` occurs as follows:

5: if $k = f+1$ then

6: decide $\min(V)$

Termination and Integrity

Initially $V = \{v_i\}$

To execute `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: $V := V \cup S_j$

`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 `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: $V := V \cup S_j$

`decide(x)` occurs as follows:

5: if $k = f+1$ then

6: decide $\min(V)$

Termination

Every correct process

③ reaches round $f + 1$

③ Decides on $\min(V)$ --- which is well defined

Termination and Integrity

Initially $V = \{v_i\}$

To execute 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: $V := V \cup S_j$

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

- ③ reaches round $f+1$
- ③ Decides on $\min(V)$ --- which is well defined

Termination and Integrity

Initially $V = \{v_i\}$

To execute 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: $V := V \cup S_j$

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

- ③ reaches round $f+1$
- ③ Decides on $\min(V)$ --- which is well defined

Termination and Integrity

Initially $V = \{v_i\}$

To execute 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: $V := V \cup S_j$

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

- ③ reaches round $f+1$
- ③ Decides on $\min(V)$ --- which is well defined

Validity

Initially $V = \{v_i\}$

To execute 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: $V := V \cup S_j$

decide(x) occurs as follows:

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

Validity

Initially $V = \{v_i\}$

To execute 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:    $V := V \cup S_j$ 
```

decide(x) occurs as follows:

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

- 1. Suppose every process proposes v^*
- 2. Since only crash model, only v^* can be sent
- 3. By Validity of send and receive, v^* sent by correct processes will be received by correct processes
- 4. By Uniform Integrity of send and receive, only v^* can be received
- 5. By protocol, $V = \{v^*\}$
- 6. $\min(V) = v^*$
- 7. decide(v^*)

Agreement

Initially $V = \{v_i\}$

To execute 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:    $V := V \cup S_j$ 
```

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 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:    $V := V \cup S_j$ 
```

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 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:    $V := V \cup S_j$ 
```

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 $f+1$

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 and adds x to its V in round $r+1$

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 and adds x to its V in round $r+1$
- **What if $r = f+1$?**

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 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 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 $m \neq \text{SF}$, then some process must have broadcast m

Termination

Every correct process eventually delivers some message

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 $p \neq \text{sender}$ 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 $p \neq \text{sender}$ 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 $faulty(p, i)$
- If $|faulty(p, i)| < i$ there can be no active dangerous chains, and p can safely deliver SF

Early Stopping: The Protocol

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

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

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 $|faulty(p, k)| < k$ then

11: value := SF

12: deliver value

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

Termination

Let $faulty(p, k)$ 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:  $faulty(p, k) := faulty(p, k-1) \cup \{q \mid p \text{ received no value from } q \text{ in round } k\}$ 
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  $|faulty(p, k)| < k$  then
11:    $\text{value} := \text{SF}$ 
12:   deliver value
13:   if  $k = f+1$  then halt
```

Termination

Let $faulty(p, k)$ 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:  $faulty(p, k) := faulty(p, k-1) \cup \{q \mid p \text{ received no value from } q \text{ in round } k\}$ 
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  $|faulty(p, k)| < k$  then
11:    $\text{value} := \text{SF}$ 
12:   deliver value
13:   if  $k = f+1$  then halt
```

- 1. If in any round a process receives a value, then it delivers the value in that round
- 2. If a process has received only "?" for $f+1$ rounds, then it delivers SF in round $f+1$

Validity

Let $faulty(p, k)$ 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:  $faulty(p, k) := faulty(p, k-1) \cup \{q \mid p \text{ received no value from } q \text{ in round } k\}$ 
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  $|faulty(p, k)| < k$  then
11:    $\text{value} := \text{SF}$ 
12:   deliver value
13:   if  $k = f+1$  then halt
```

Validity

Let $faulty(p, k)$ 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:  $faulty(p, k) := faulty(p, k-1) \cup \{q \mid p \text{ received no value from } q \text{ in round } k\}$ 
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  $|faulty(p, k)| < k$  then
11:    $\text{value} := \text{SF}$ 
12:   deliver value
13:   if  $k = f+1$  then halt
```

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

Agreement - 1

Let $faulty(p, k)$ 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:  $faulty(p, k) := faulty(p, k-1) \cup \{q \mid p \text{ received no value from } q \text{ in round } k\}$ 
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  $|faulty(p, k)| < k$  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 $k, 1 \leq k \leq r$, p_{k-1} sent m and p_k 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 , $|faulty(q_j, j)| < j$, 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 $faulty(p, k)$ 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:  $faulty(p, k) := faulty(p, k-1) \cup \{q \mid p \text{ received no value from } q \text{ in round } k\}$ 
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  $|faulty(p, k)| < k$  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 $faulty(p, k)$ 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:  $faulty(p, k) := faulty(p, k-1) \cup \{q \mid p \text{ received no value from } q \text{ in round } k\}$ 
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  $|faulty(p, k)| < k$  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

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} $1 \leq k \leq j$ in round k

q_j must conclude that p_0, \dots, p_{j-1} are all faulty processes

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

CONTRADICTION

Agreement - 3

Let $faulty(p, k)$ 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:  $faulty(p, k) := faulty(p, k-1) \cup \{q \mid p \text{ received no value from } q \text{ in round } k\}$ 
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  $|faulty(p, k)| < k$  then
11:    $\text{value} := \text{SF}$ 
12:   deliver value
13:   if  $k = f+1$  then halt
```

Agreement - 3

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

• $f+1$ processes; only f faulty

• one of p_0, \dots, p_f is correct-- let it be p_c

• 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 $faulty(p, k)$ 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:  $faulty(p, k) := faulty(p, k-1) \cup \{q \mid p \text{ received no value from } q \text{ in round } k\}$ 
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  $|faulty(p, k)| < k$  then
11:   value := SF
12:   deliver value
13:   if  $k = f+1$  then halt
    
```

Integrity

Let $faulty(p, k)$ 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:  $faulty(p, k) := faulty(p, k-1) \cup \{q \mid p \text{ received no value from } q \text{ in round } k\}$ 
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  $|faulty(p, k)| < k$  then
11:   value := SF
12:   deliver value
13:   if  $k = f+1$  then halt
    
```

⑥ At most one m

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

⑥ If $m \neq \text{SF}$, only if m was broadcast

□ From Lemma 1 in the proof of Agreement