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 $\{v \in V : p_j \text{ has not already sent } v\}$ 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

Termination

Every correct process reaches round $f+1$
Decides on min($V$) --- which is well defined

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 $= x_0$, then it is proposed in round 1
- else, suppose received in round $k$. By induction:
  - $k = j+1$
  - by Uniform Integrity of underlying send and receive, it must have been sent in round $k!$
  - 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

Validity

Initially $V=\{v_i\}$
To execute propose($v_i$)
round $k$, $1 \leq k \leq f+1$
1: send $\{v \in V : p_j \text{ has not already sent } v\}$ 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

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($V$) = $v^*$
decide($v^*$)
Lemma 2: 
For any r ≥ 1, if a process p receives a value v in round r, then there exists a sequence of processes P₀, P₁, ..., Pᵣ such that P₀ = v's proponent, Pᵣ = p and in each round k, 1 ≤ k ≤ r, Pₖ₋₁ sends v and Pₖ receives it. Furthermore, all processes in the sequence are distinct.

Proof
By induction on the length of the sequence
A Lower Bound

Theorem
There is no algorithm that solves the consensus problem in less 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 proof technique.

Views
Let \( \alpha \) be an execution. The view of process \( p_i \) in \( \alpha \), 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 \( \alpha \).

Similarity
Definition Let \( \alpha_1 \) and \( \alpha_2 \) be two executions of consensus and let \( p_i \) be a correct process in both \( \alpha_1 \) and \( \alpha_2 \). Execution \( \alpha_1 \) is similar to execution \( \alpha_2 \) 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 \( \alpha_1 \) and \( \alpha_2 \) be two executions of consensus and let \( p_i \) be a correct process in both \( \alpha_1 \) and \( \alpha_2 \). Execution \( \alpha_1 \) is similar to execution \( \alpha_2 \) 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 \( \alpha_1 \) and \( \alpha_2 \) be two executions of consensus and let \( p_i \) be a correct process in both \( \alpha_1 \) and \( \alpha_2 \). Execution \( \alpha_1 \) is similar to execution \( \alpha_2 \) 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 \( \text{dec}(\alpha_1) = \text{dec}(\alpha_2) \).

---

**Single-Failure Case**

There is no algorithm that solves the consensus problem in less than two rounds in the presence of one crash failure, if \( n \geq 3 \).
The Idea

By contradiction

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

So what?

Adjacent $\alpha^i$s are similar!

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

$\alpha^i_j$ is obtained from $\alpha^i$ after removing the messages that $p_i$ sends to the $j$-th highest numbered processors (excluding itself)

The executions
Indistinguishability

$p_0$ 1
$p_{i-1}$ 1
$p_i$ 0
$p_{i+1}$ 0
$p_{n-1}$ 0

$\alpha^2$
$\alpha_0$

Indistinguishability

$p_0$ 1
$p_{i-1}$ 1
$p_i$ 0
$p_{i+1}$ 0
$p_{n-1}$ 0

$\alpha^i$
$\alpha_0$

Indistinguishability

$p_0$ 1
$p_{i-1}$ 1
$p_i$ 0
$p_{i+1}$ 0
$p_{n-1}$ 0

$\alpha^2$
$\alpha_0$

Indistinguishability

$p_0$ 1
$p_{i-1}$ 1
$p_i$ 0
$p_{i+1}$ 0
$p_{n-1}$ 0

$\alpha^i$
$\alpha_0$

Indistinguishability

$p_0$ 1
$p_{i-1}$ 1
$p_i$ 0
$p_{i+1}$ 0
$p_{n-1}$ 0

$\alpha^n$
$\alpha_{n-1}$
Indistinguishability

\begin{align*}
\beta_0 \\ \beta_{n-2} \\ \beta_{n-3}
\end{align*}
Indistinguishability

Terminating Reliable Broadcast

**Termination**
Every correct process eventually delivers some message

**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 \( m \)

TRB for benign failures

Terminates in \( f + 1 \) rounds

How can we do better?

- Find a protocol whose round complexity is proportional to \( f \) — 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.

Termination

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

1. If $p$ is sender then value := m else value := $\text{null}$.

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

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

Early Stopping: The Protocol

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

1. If $p$ is sender then value := m else value := $\text{null}$.

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

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

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 $f+1$. 

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

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

Termination
Validity

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

1. If \( p = \text{sender} \) then value := \( m \), else value := ?

Process \( p \) in round \( k, 1 \leq k < f+1 \)
2. Send value to all
3. If value \( = ? \) then halt
4. Receive round \( k \) values from all
5. \( \text{faulty}(p,k) = \text{faulty}(p,k-1) \cup \{ q | p \text{ received no value from } q \text{ in round } k \} \)
6. If received value \( v \neq ? \) then
7. \( \text{value} := v \)
8. Deliver(value)
9. Else if \( k = f+1 \) or \( |\text{faulty}(p,k)| < k \) then
10. \( \text{value} := \text{SF} \)
11. Deliver(value)
12. If \( k = f+1 \) then halt

Validity

If the sender is correct then it sends \( m \) to all in round 1.

By Validity of the underlying send and receive, every correct process will receive \( m \) by the end of round 1.

Agreement - 1

Lemma 1
For any \( r \geq 1 \), if a process \( p \) delivers \( m = \text{SF} \) in round \( r \), then there exists a sequence of processes \( P_0, P_1, \ldots, P_r \) such that \( P_0 = \text{sender} \), \( P_r = p \), and in each round \( k, 1 \leq k < r \), \( P_k \) sent \( m \) and \( P_{k+1} \) received it. Furthermore, all processes in the sequence are distinct, unless \( r = 1 \) and \( P_0 = P_1 = \text{sender} \).

Process \( p \) in round \( k, 1 \leq k < f+1 \)
2. Send value to all
3. If value \( = ? \) then halt
4. Receive round \( k \) values from all
5. \( \text{faulty}(p,k) = \text{faulty}(p,k-1) \cup \{ q | p \text{ received no value from } q \text{ in round } k \} \)
6. If received value \( v \neq ? \) then
7. \( \text{value} := v \)
8. Deliver(value)
9. Else if \( k = f+1 \) or \( |\text{faulty}(p,k)| < k \) then
10. \( \text{value} := \text{SF} \)
11. Deliver(value)
12. If \( k = f+1 \) then halt

Agreement - 2

Lemma 2
For any \( r \geq 1 \), if a process \( p \) sets value to \( \text{SF} \) in round \( r \), then there exist some \( j \leq r \) and a sequence of distinct processes \( q_j, q_{j+1}, \ldots, q_r = p \) such that \( q_j \) only receives \( ? \) in rounds \( 1 \) to \( j \), \( |\text{faulty}(q_j,j)| < j \), and in each round \( k, 1 \leq k < r \), \( q_k \) sends \( \text{SF} \) to \( q_j \) and \( q_j \) receives \( \text{SF} \).

Process \( p \) in round \( k, 1 \leq k < f+1 \)
2. Send value to all
3. If value \( = ? \) then halt
4. Receive round \( k \) values from all
5. \( \text{faulty}(p,k) = \text{faulty}(p,k-1) \cup \{ q | p \text{ received no value from } q \text{ in round } k \} \)
6. If received value \( v \neq ? \) then
7. \( \text{value} := v \)
8. Deliver(value)
9. Else if \( k = f+1 \) or \( |\text{faulty}(p,k)| < k \) then
10. \( \text{value} := \text{SF} \)
11. Deliver(value)
12. 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 \( \text{SF} \), respectively.
Let |faulty(p)| be the set of processes that have failed to send a message to p in any round 1 \ldots k.
1. if p = sender then value := m else value := SF

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

**Proof**

By contradiction
Suppose p and q set value = m and q sets value = SF.
By Lemmas 1 and 2 there exist
\(p_0, \ldots, p_r\)
\(q_j, \ldots, q_r\)
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 \(p_0, \ldots, p_{j-1}\)
are all faulty processes.
But then, |faulty(p_j)| \geq j
CONTRADICTION

**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.

Let |faulty(p)| be the set of processes that have failed to send a message to p in any round 1 \ldots k.
1. if p = sender then value := m else value := SF

Process p in round k, 1 \leq k \leq f + 1:
2. send value to all
3. if value \neq m then halt
4. receive round k values from all
5. |faulty(p)| := |faulty(p) - \{q\} | p received no value from q in round k
6. if received value v \neq m then halt
7. value := v
8. deliver(value)
9. else if k = f + 1 or |faulty(p)| < k then
10. value := SF
11. deliver(value)
12. 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 = SF.
1. By Lemma 3, no correct process can set value differently in round r.
2. In round r + 1 \leq f + 1, that correct process sends its value to all.
3. Every correct process receives and delivers the value in round r + 1 = f + 1.
4. By Lemma 1, there exist a sequence \(p_0, \ldots, p_f\).
5. Consider processes \(p_0, \ldots, p_f\) distinct processes
6. Since p_0 \neq p_f,
7. To send v in round r + 1, p_0 must have set its value to v and delivered v in round r < r.

**Integrity**

Let |faulty(p)| be the set of processes that have failed to send a message to p in any round 1 \ldots k.
1. if p = sender then value := m else value := SF

Process p in round k, 1 \leq k \leq f + 1:
2. send value to all
3. if value \neq m then halt
4. receive round k values from all
5. |faulty(p)| := |faulty(p) - \{q\} | p received no value from q in round k
6. if received value v \neq m then halt
7. value := v
8. deliver(value)
9. else if k = f + 1 or |faulty(p)| < k then
10. value := SF
11. deliver(value)
12. if k = f + 1 then halt
Integrity

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

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 value = ? then halt
4. Receive round \( k \) values from all
5. If \( k = f+1 \) or \( |f_{\text{faulty}}(p,k)| \leq k \) then halt
6. If received value \( v \) = ? then
7. value := v
8. deliver(value)
9. Else if \( k = f+1 \) or \( |f_{\text{faulty}}(p,k)| < k \) then
10. value := SF
11. deliver(value)
12. If \( k = f+1 \) then halt