Consensus and Reliable Broadcast
Broadcast

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

**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 $m$, then some process must have broadcast $m$
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 $m$
# Consensus

<table>
<thead>
<tr>
<th>Termination</th>
<th>Every correct process eventually decides some value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validity</td>
<td>If all processes that propose a value propose $v$, then all correct processes eventually decide $v$</td>
</tr>
<tr>
<td>Agreement</td>
<td>If a correct process decides $v$, then all correct processes eventually decide $v$</td>
</tr>
<tr>
<td>Integrity</td>
<td>Every correct process decides at most one value, and if it decides $v \neq \text{NU}$, then some process must have proposed $v$</td>
</tr>
</tbody>
</table>
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

- Synchronous Systems
  - Consensus for synchronous systems with crash failures
  - Lower bound on the number of rounds
  - Early stopping protocols for Reliable Broadcast
  - Reliable Broadcast for arbitrary failures with message authentication
  - Lower bound on the ratio of faulty processes for Consensus with arbitrary failures
  - Reliable Broadcast for arbitrary failures

- Asynchronous Systems
  - Impossibility of Consensus for crash failures
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 $\text{propose}(v_i)$

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: $\text{decide} \ \min(V)$
An execution
An execution
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?

\[ v_1 = v_3 = v_4 \]
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?
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

\[
\begin{align*}
p_0 & \quad \text{round 1} \\
p_1 & \\
p_2 & \quad \text{round 2} \\
p_{i-1} & \quad \text{rounds 3...} \ i - 1 \\
p_i & \quad \text{round } \ i \\
p_0 & \quad \text{round 1} \\
\end{align*}
\]
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 $\text{min}(V)$
Termination and Integrity

Initially $V=\{v_i\}$

To execute $\text{propose}(v_i)$
  \begin{enumerate}
  \item round $k$, $1 \leq k \leq f+1$
  \item send $\{v \in V : p_j \text{ has not already sent } v\}$ to all
  \item for all $j$, $0 \leq j \leq n-1$, $j \neq i$ do
  \item receive $S_j$ from $p_j$
  \item $V := V \cup S_j$
  \end{enumerate}

decide$(x)$ occurs as follows:
  \begin{enumerate}
  \item if $k = f+1$ then
  \item decide $\min(V)$
  \end{enumerate}

Termination

Every correct process
  \begin{enumerate}
  \item reaches round $f + 1$
  \item Decides on $\min(V)$ --- which is well defined
  \end{enumerate}

Integrity

At most one value:
  \begin{itemize}
  \item one decide, and $\min(V)$ is unique
  \end{itemize}

Only if it was proposed:
  \begin{itemize}
  \item To be decided upon, must be in $V$ at round $f+1$
  \item if value = $v_i$, then it is proposed in round 1
  \item else, suppose received in round $k$. By induction:
    \begin{itemize}
    \item $k = 1$:
      \begin{itemize}
      \item by Uniform Integrity of underlying send
      \item and receive, it must have been sent in round 1
      \item by the protocol and because only crash
      \item failures, it must have been proposed
      \end{itemize}
    \item Induction Hypothesis: all values received up to
    \item round $k = j$ have been proposed
    \item $k = j+1$
      \begin{itemize}
      \item sent in round $j+1$ (Uniform Integrity of send
      \item and synchronous model)
      \item must have been part of $V$ of sender at end
      \item of round $j$
      \item by protocol, must have been received by sender
      \item by end of round $j$
      \item by induction hypothesis, must have been proposed
      \end{itemize}
    \end{itemize}
  \end{itemize}
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: $V := V \cup S_j$

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: $\text{decide} \ \text{min}(V)$

> 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, $V = \{v^*\}$

$\text{min}(V) = v^*$

$\text{decide}(v^*)$
Agreement

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 \( p_0, p_1, \ldots, p_r \) such that \( p_0 = v \)'s proponent, \( p_r = p \) and in each round \( k, 1 \leq k \leq r \), \( p_{k-1} \) sends \( v \) and \( p_k \) receives it. Furthermore, all processes in the sequence are distinct.

Proof
By induction on the length of the sequence.

Initially \( V = \{v_i\} \)

To execute \( \text{propose}(v_i) \)
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 \( \min(V) \)
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: $\text{decide} \text{ min}(V)$
Agreement

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

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 $\text{min}$ is a deterministic function.

Proof:
- Show that if a correct process has $x$ in its $V$ at the end of round $f + 1$, then every correct process has $x$ in its $V$ at the end of round $f + 1$.
- Let $r$ be earliest round $x$ is added to the $V$ of a correct process. Let that process be $p$.
- If $r \leq f$, then $p$ sends $x$ in round $r + 1 \leq f + 1$; every correct process receives $x$ and adds $x$ to its $V$ in round $r + 1$.
- What if $r = f + 1$?
- By Lemma 1, there exists a sequence $p_0, \ldots, p_{f+1} = p$ of distinct processes.
- Consider processes $p_0, \ldots, p_f$.
- $f + 1$ processes; only $f$ faulty.
- one of $p_0, \ldots, p_f$ is correct, and adds $x$ to its $V$ before $p$ does it in round $r$.
CONTRACTION!
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 $\alpha$. 
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 $\alpha_1|p_i = \alpha_2|p_i$
Simil\aaty

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 $\text{dec}(\alpha_1) = \text{dec}(\alpha_2)$

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 $\beta_1, \beta_2, \ldots, \beta_{k+1}$ such that

$$\alpha_1 = \beta_1 \sim p_{i_1} \beta_2 \sim p_{i_2} \ldots \sim p_{i_k} \beta_{k+1} = \alpha_2$$
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 $\text{dec}(\alpha_1) = \text{dec}(\alpha_2)$

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 $\beta_1, \beta_2, \ldots, \beta_{k+1}$ such that $\alpha_1 = \beta_1 \sim_{p_{i_1}} \beta_2 \sim_{p_{i_2}} \ldots \sim_{p_{i_k}} \beta_{k+1} = \alpha_2$

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 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?
Definition

\( \alpha^i \) is the execution of the algorithm in which

\( \bigotimes \) no failures occur

\( \bigotimes \) processes \( p_0, \ldots, p_{i-1} \) propose 1
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

\[ p_0 \]
\[ p_{i-1} \]
\[ p_i \]
\[ p_{i+1} \]
\[ p_{n-1} \]
\[ \alpha^0 \]
\[ \alpha^i \]
Indistinguishability
Indistinguishability

\[ p_0, 1 \]
\[ p_{i-1}, 1 \]
\[ p_i, 0 \]
\[ p_{i+1}, 0 \]
\[ p_{n-1}, 0 \]

\[ \alpha_i \]
\[ \alpha \]
\[ \alpha^i \]
\[ \alpha_2 \]
Indistinguishability

\[ p_0 \quad 1 \quad \cdots \quad p_i \quad 0 \quad p_{i+1} \quad 0 \quad \cdots \quad p_{n-1} \quad 0 \]

\[ \alpha_i \quad \cdots \quad \alpha_{n-1} \]
Indistinguishability

\[ p_0 1 \]
\[ p_{i-1} 1 \]
\[ p_i 0 \]
\[ p_{i+1} 0 \]
\[ p_{n-1} 0 \]

\[ \approx \]

\[ \alpha_i^n \]
\[ \approx \]

\[ \alpha_{n-1}^i \]

\[ p_0 1 \]
\[ p_{i-1} 1 \]
\[ p_i 1 \]
\[ p_{i+1} 0 \]
\[ p_{n-1} 0 \]

\[ \beta_{n-1}^i \]
Indistinguishability
Indistinguishability

\[ p_0 1 \]
\[ p_{i-1} 1 \]
\[ p_i 0 \]
\[ p_{i+1} 0 \]
\[ p_{n-1} 0 \]

\[ \alpha^i \]
\[ \alpha_{n-1} \]

\[ \approx \]

\[ p_0 1 \]
\[ p_{i-1} 1 \]
\[ p_i 1 \]
\[ p_{i+1} 0 \]
\[ p_{n-1} 0 \]

\[ \beta_{n-3}^i \]
Indistinguishability
Indistinguishability

\[ p_0, \quad 1 \]

\[ p_{i-1}, \quad 1 \]
\[ p_i, \quad 0 \]
\[ p_{i+1}, \quad 0 \]
\[ p_{n-1}, \quad 0 \]

\[ \alpha^i \]
\[ \alpha^i \]
\[ \alpha^{i-1} \]

\approx

\[ p_0, \quad 1 \]

\[ p_{i-1}, \quad 1 \]
\[ p_i, \quad 1 \]
\[ p_{i+1}, \quad 0 \]
\[ p_{n-1}, \quad 0 \]

\[ \alpha^{i+1} \]
\[ \beta_0^i \]
Indistinguishability

\[ p_0 \approx p_i^0 \approx p_{i+1} \]

\[ p_n \approx p_{i+1} \approx p_{i-1} \]

\[ \alpha^i \approx \alpha^{i+1} \]
Terminating
Reliable Broadcast

<table>
<thead>
<tr>
<th>Termination</th>
<th>Every correct process eventually delivers some message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validity</td>
<td>If the sender is correct and broadcasts a message $m$, then all correct processes eventually deliver $m$</td>
</tr>
<tr>
<td>Agreement</td>
<td>If a correct process delivers a message $m$, then all correct processes eventually deliver $m$</td>
</tr>
<tr>
<td>Integrity</td>
<td>Every correct process delivers at most one message, and if it delivers $m \neq SF$, then some process must have broadcast $m$</td>
</tr>
</tbody>
</table>
TRB for benign failures

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

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

1: if $p =$ 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 $\neq ?$ 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: 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 $1...k$

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 $\neq ?$ 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: 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 $1...k$

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 $\text{value}$ to all
3: if $\text{value} \neq ?$ 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: 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

- 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 $f+1$
Validity

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

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 $\neq ?$ 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: value := $v$
8: deliver(value)
9: else if $k = f+1$ or $|\text{faulty}(p,k)| < k$ then
10: value := $\text{SF}$
11: deliver(value)
12: 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 $1 \ldots k$

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 $\neq ?$ 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: 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

- 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
- By the protocol, every correct process will deliver $m$ by the end of round 1
Agreement - 1

Lemma 1:
For any r ≥ 1, if a process p delivers m ≠ SF in round r, then there exists a sequence of processes p₀, p₁, ..., pᵣ such that p₀ = sender, pᵣ = p, and in each round k, 1 ≤ k ≤ r, pᵦ₋₁ sent m and pᵦ received it. Furthermore, all processes in the sequence are distinct, unless r = 1 and p₀ = p₁ = sender.

Lemma 2:
For any r ≥ 1, if a process p sets value to SF in round r, then there exist some j ≤ r and a sequence of distinct processes q_j, qⱼ₊₁, ..., qᵣ = p such that qⱼ only receives “?” in rounds 1 to j, |faulty(qⱼ, j)| < j, and in each round k, j + 1 ≤ k ≤ r, qᵦ₋₁ sends SF to qᵦ and qᵦ receives SF.

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

1: if p = sender then value := m else value := ?

Process p in round k, 1 ≤ k ≤ f+1

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

**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 \(1...k\)

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 \(\neq ?\) 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: \(\text{value} := v\)
8: \(\text{deliver}(\text{value})\)
9: else if \(k = f+1\) or \(|\text{faulty}(p,k)| < k\) then
10: \(\text{value} := \text{SF}\)
11: \(\text{deliver}(\text{value})\)
12: if \(k = f+1\) then halt

**Proof**

By contradiction

Suppose \(p\) sets value = \(m\) and \(q\) sets value = \(\text{SF}\)

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

**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

**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...k

1: if p = sender then value := m else value := ?

Process p in round  k, 1 ≤ k ≤ f+1

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

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

2: send $\text{value}$ to all
3: if $\text{value} \neq ?$ 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($\text{value}$)
9: else if $k = f+1$ or $|\text{faulty}(p,k)| < k$ then
10: $\text{value} := \text{SF}$
11: deliver($\text{value}$)
12: if $k = f+1$ then halt

Proof

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

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

$r \leq f$

- 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 $p_0, \ldots, p_{f+1} = p_r$ of distinct processes
- Consider processes $p_0, \ldots, p_f$
  - $f + 1$ processes; only $f$ faulty
  - one of $p_0, \ldots, p_f$ 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 $|\text{faulty}(p,k)|$ be the set of processes that have failed to send a message to $p$ in any round $1 \ldots k$

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

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

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

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 value \(\neq ?\) 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: \(\text{deliver}(\text{value})\)
9: else if \(k = f+1\) or \(|\text{faulty}(p,k)| < k\) then
10: \(\text{value} := \text{SF}\)
11: \(\text{deliver}(\text{value})\)
12: if \(k = f+1\) then halt

- At most one \(m\)
  - Failures are benign, and a process executes at most one deliver event before halting
  - If \(m \neq \text{SF}\), only if \(m\) was broadcast
  - From Lemma 1 in the proof of Agreement