Consensus and Reliable Broadcast

Broadcast

If a process sends a message \( m \), then every process eventually delivers \( m \)

How can we adapt the spec for an environment where processes can fail? And what does “fail” mean?
A hierarchy of failure models

- **Fail-stop**
- **Crash**
- **Send Omission**
- **Receive Omission**
- **General Omission**
- **Arbitrary failures with message authentication**
- **Arbitrary (Byzantine) failures**

**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 $m$, 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.

**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 $m$. 

"Consensus"
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

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?

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 $q$

Plan

Synchronous Systems
Consensus for synchronous systems with crash failures
Lower bound on the number of rounds
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
Failure detectors
PAXOS
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: decide $\min(V)$

An execution

$\ldots p_1 \ldots p_2 \ldots p_3 \ldots p_4 \ldots$

An execution

$\ldots p_1 \ldots p_2 \ldots p_3 \ldots p_4 \ldots$

$\ldots v_1 \ldots v_2 \ldots v_3 \ldots v_4 \ldots$

$\ldots p_1 \ldots p_2 \ldots v_3 \ldots p_4 \ldots$

$\ldots v_1 \ldots v_2 \ldots v_3 \ldots v_4 \ldots$
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?

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 $\text{propose}(v_i)$
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 $\text{min}(V)$

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: $V := V \cup S_j$
decide($x$) occurs as follows:
5: if $k = f + 1$ then
6: decide $\text{min}(V)$

Termination
Every correct process
- reaches round $f + 1$
- decides on $\text{min}(V)$ --- which is well defined
Termination and Integrity

**Termination**
Every correct process
\(\text{reaches round } f + 1\)
\(\text{Decides on } \min(V) \text{ --- which is well defined}\)

**Integrity**
At most one value:

- Only if it was proposed:

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

To execute propose(\(v_i\))

\(\text{round } k, 1 \leq k \leq f + 1\)

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

decide(\(x\)) occurs as follows:

5: \(\text{if } k = f + 1 \text{ then}\)
6: \(\text{decide } \min(V)\)

Validity

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

To execute propose(\(v_i\))

\(\text{round } k, 1 \leq k \leq f + 1\)

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

decide(\(x\)) occurs as follows:

5: \(\text{if } k = f + 1 \text{ then}\)
6: \(\text{decide } \min(V)\)

Every correct process
\(\text{reaches round } f + 1\)
\(\text{Decides on } \min(V) \text{ --- which is well defined}\)
Validity

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

\( \text{decide}(v) \) occurs as follows:
5. if \( k = f + 1 \) then
6. \( \text{decide}(v^*) \)

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

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 \( p_r = p \) \( p_0 \) 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

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

\( \text{decide}(v) \) occurs as follows:
5. if \( k = f + 1 \) then
6. \( \text{decide}(v^*) \)

Lemma 2:
In every execution, at the end of round \( f + 1 \), for every correct processes \( p_i \) and \( p_j \), \( V_i = V_j \) if \( p_i \) receives \( v \) and \( p_j \) receives \( v \).
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 \( f + 1 \).
In every execution, at the end of round \( f + 1 \), then every correct process \( p \) has \( x \) in its \( V \) at the end of round \( f + 1 \).

Lemma 2:
In every execution, at the end of round \( f + 1 \), then every correct process \( p \) has \( x \) in its \( V \) at the end of round \( f + 1 \).

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 \( \overline{p} \).

If \( r \leq f \), then \( \overline{p} \) 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 \( f + 1 \), \( V_i = V_f \) for every correct processes \( p_i \) and \( p_f \).

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 \( r = f + 1 \)?

Agreement follows from Lemma 2, since \( min \) is a deterministic function.

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.
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.

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

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