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
- General Omission
- Receive Omission
- Arbitrary failures with message authentication
- Arbitrary (Byzantine) failures

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

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

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
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
- 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 $p_i$:

1. Initially $V = \{v_i\}$
2. To execute $\text{propose}(v_i)$
3. decide($x$) occurs as follows:
   1. send $\{v_i\}$ to all
   2. for all $j$, $0 \leq j \leq n-1$, $j \neq i$ do
   3. receive $S_j$ from $p_j$
   4. $V := V \cup S_j$
   5. decide $\min(V)$

An execution

- An execution
An execution

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

An execution

Suppose $v_1 = v_3 = v_4$ at the end of round 1
Can $p_3$ decide?
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 `propose(v)`

- 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: $V := V \cup S_j$

$\text{decide}(v)$ 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
Termination and Integrity

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

To execute propose(\( x \))

1. \( k = f + 1 \)
2. for all \( j, 0 \leq j < n - 1, j \neq i \)
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. \( \text{decide}(x) \)

Termination

Every correct process

\( \text{reaches round } f + 1 \)

\( \text{Decides on } \min(V) \) --- which is well defined

Integrity

At most one value:

- Only if \( x \) was proposed:
  - To be decided upon, must be in \( V \) at round \( f + 1 \)
  - If \( x = v \) then it is proposed in round \( f + 1 \)
  - Else, suppose received in round \( k \), by induction
    - \( k = f + 1 \)
    - By Uniform Integrity of underlying send and receive
    - Must have been sent in round \( f + 1 \)
    - By 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(\( x \))

1. \( k = f + 1 \)
2. for all \( j, 0 \leq j < n - 1, j \neq i \)
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. \( \text{decide}(x) \)

Termination

Every correct process

\( \text{reaches round } f + 1 \)

\( \text{Decides on } \min(V) \) --- which is well defined

Integrity

At most one value:

- Only if it was proposed:
  - One decide, and \( \min(V) \) is unique

Validity

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

To execute propose(\( x \))

1. \( k = f + 1 \)
2. for all \( j, 0 \leq j < n - 1, j \neq i \)
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. \( \text{decide}(x) \)
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, $V = \{ v^* \}$
- $\min(V) = v^*$
- $\text{decide}(v^*)$

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.
**Agreement**

**Proof:**

- Show that if a correct process has a value in its local view at the end of round \( f + 1 \), then every correct process has the same value in its local view at the end of round \( f + 1 \).

- Let \( r \) be the earliest round where \( x \) is added to the local view of a correct process \( p \). 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 it to its local view in round \( r + 1 \).

**Initial Conditions:**

Initially, \( V = \{ x \} \).

To execute propose(\( x \)) with round \( 6 \) \( 1 \leq k \leq f + 1 \):

1. Send \( \{ i = k : \) process \( p \) has not already sent \( x \} \) to all processes.
2. For all \( j \), if \( 0 \leq j < n - 1 \), do:
   a. Receive \( V \) from process \( p \).
3. Receive \( V \) from process \( p \).
4. \( V = V \cup S \).

**Decision:**

- If \( k = f + 1 \) then:
  1. Decide \( x \).

**Lemma 2:**

In every execution, at the end of round \( f + 1 \), the local view \( V \) of every correct process \( p \), and \( p' \) is the same.

Agreement follows from Lemma 2, since \( \text{min} \) 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.

**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 $\text{SF}$

**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 $1, \ldots, k$.

1. if $p$ is sender then value := $m$ else value := $\oplus$

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

6. if received value $v$, $v \neq ?$ then

7. value := $v$

8. deliver value

9. if $p$ is sender then value := $\oplus$

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

**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 := $\oplus$

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

6. if received value $v$, $v \neq ?$ then

7. value := $v$

8. deliver value

9. if $p$ is sender then value := $\oplus$

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
Termination

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

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 $\text{SF}$ 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 $1, \ldots, k$.

If $p$ sender then value $= m$, else value $= ?$.

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.

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

Agreement – 1

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

Lemma 2:

For any $r \geq 1$, if a process $p$ sets value to SF in round $r$, then there exists some $j < r$ and a sequence of distinct processes $q_0, q_1, \ldots, q_{r-1}$ such that $q_j$ only receives "?" in rounds $1$ to $j$, $|\text{faulty}(q_j, k)| < j$, and in each round $k$, $j+1 \leq k \leq r$, $q_{k-1}$ sends SF to $q_k$ and $q_k$ receives SF.
Let $\text{fault}(p, k)$ 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 delivered in round $k-1$ then halt
4. receive round $k$ values from all
5. $\text{fault}(p, k) = \text{fault}(p, k-1) \cup \{p\}$
6. if received value $= ?$ then
7. value $= m$
8. deliver value
9. if $p = \text{sender}$ then value $= m$
10. else if $k-1/f+1$ or $|\text{fault}(p, k)| < k$ then
11. 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 $\text{SF}$, respectively.

---

**Agreement - 2**

Let $\text{fault}(p, k)$ 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 delivered in round $k-1$ then halt
4. receive round $k$ values from all
5. $\text{fault}(p, k) = \text{fault}(p, k-1) \cup \{p\}$
6. if received value $= ?$ then
7. value $= m$
8. deliver value
9. if $p = \text{sender}$ then value $= m$
10. else if $k-1/f+1$ or $|\text{fault}(p, k)| < k$ then
11. value $= \text{SF}$
12. deliver value
13. 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 $p_0, \ldots, p_f$, $q_0, \ldots, q_f$ with the appropriate characteristics.

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

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

**CONTRACTION**

---

**Agreement - 3**

Let $\text{fault}(p, k)$ 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 delivered in round $k-1$ then halt
4. receive round $k$ values from all
5. $\text{fault}(p, k) = \text{fault}(p, k-1) \cup \{p\}$
6. if received value $= ?$ then
7. value $= m$
8. deliver value
9. if $p = \text{sender}$ then value $= m$
10. else if $k-1/f+1$ or $|\text{fault}(p, k)| < k$ then
11. value $= \text{SF}$
12. deliver value
13. if $k = f+1$ then halt

**Proof**

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

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

1. If $p$ is the sender then value := $m$, else value := $\infty$.

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. $\text{fault}(p, k) = \text{fault}(p, k-1) \cup \{p\}$, received no value from $p$ in round $k$.
6. if received value $\neq \infty$ then
7. value := $m$
8. deliver value
9. if $p$ is the sender then value := $\infty$.
10. else if $k = f+1$ or $|\text{fault}(p, k)| < k$ then
11. value := SF
12. deliver value
13. 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

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 $\alpha$ 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 $\alpha$.

\[ p_1 \xrightarrow{m} p_2 \xrightarrow{m} \ldots p_4 \]
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 $\alpha_2$. $\alpha_1$ is similar to $\alpha_2$ 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 $\alpha_2$. $\alpha_1$ is similar to $\alpha_2$ 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 consensus in fewer than two rounds in the presence of one crash failure, if $n \geq 3$

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

$$\alpha_1 = \beta_1 \sim_{p_1} \beta_2 \sim_{p_2} \ldots \sim_{p_k} \beta_{k+1} = \alpha_2$$

**Note** If $\alpha_1 \approx \alpha_2$ then $p_i$ decides the same value in both executions

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

The transitive closure of $\alpha_1 \approx \alpha_2$ 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

$$\alpha_1 = \beta_1 \sim_{p_1} \beta_2 \sim_{p_2} \ldots \sim_{p_k} \beta_{k+1} = \alpha_2$$

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

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 $\alpha^j$ where $0 \leq j \leq n-1$ as follows:

$\alpha^j$ 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
Indistinguishability
Indistinguishability

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

\[ \alpha_i \approx \beta_{n-2} \]

\[ \alpha_{n-1} \approx \beta_{n-3} \]

\[ \alpha^i \approx \beta^i \]

\[ \alpha_{n-1} \approx \beta_{n-1} \]

\[ \alpha_i \approx \beta^i_{i+1} \]

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

$$\alpha^i \approx \alpha^{i+1}$$

Valid messages

A valid message $m$ has the following form:

- in round 1:
  $$m : s_{id}$$ (is signed by the sender)
- in round $r > 1$, if received by $p$ from $q$:
  $$m : p_1 : p_2 : \ldots : p_r$$ where
  - $p_1 = \text{sender}; p_r = q$
  - $p_1, \ldots, p_r$ are distinct from each other and from $p$
  - message has not been tampered with

AFMA: The Idea

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

Arbitrary failures with message authentication

- Fail-stop
- Crash
- Send Omission
- Receive Omission
- General Omission
- Arbitrary failures with message authentication
- Arbitrary (Byzantine) failures
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 $s \in relay$
  send $s$ := $p$ to all
- receive round $k$ messages from all processes
  relay := \$
- for each valid message received $s = m : p_1 : p_2 : \ldots : p_k$
  if $m \notin$ extracted then
    extracted := extracted $\cup \{m\}$
    relay := relay $\cup \{s\}$

At the end of round $f+1$
- if $\exists m$ such that extracted $\notin \{m\}$ then
  deliver $m$
- else deliver SF

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 $m : p_1 : p_2 : \ldots : p_k$
- if $r \leq f$, $p$ will send a valid message $m : p_1 : p_2 : \ldots : p_k$ in round $r+1$ and every correct process will extract it in round $r+1$
- if $r = f+1$ then
  - Choose $p_1, p_2, \ldots, p_k$ are all faulty
  - True for $p_k = x$
  - 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

Validation

Termination

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

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

Srikanth, T.K., Toueg S.
Simulating Authenticated Broadcasts to Derive Simple Fault-Tolerant Algorithms
Distributed Computing 2 (2), 80-94

AF: The Idea

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

AF: The Approach

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

Properties of broadcast and accept

- Correctness If a correct process \(p\) executes broadcast\((p, m, i)\) in round \(i\), then all correct processes will execute accept\((p, m, i)\) in round \(i\)
- Unforgeability If a correct process \(q\) executes accept\((p, m, i)\) in round \(j \geq i\), and \(p\) is correct, then \(p\) did in fact execute broadcast\((p, m, i)\) in round \(i\)
- Relay If a correct process \(q\) executes accept\((p, m, i)\) in round \(j \geq i\), then all correct processes will execute accept\((p, m, i)\) by round \(j+1\)
**AF: The Protocol - 1**

sender \( s \) in round 0:
0. extract \( m \)
1. broadcast \((s, m, 1)\)
2. if \( p \) extracted \( m \) in round \( k - 1 \) and \( p \neq \) sender then
   4. broadcast \((p, m, k)\)
3. if \( p \) has executed at least \( k \) accepts \((q, m, j)\) \( 1 \leq j \leq k \) in rounds 1 through \( k \):
   - (where \( i \) is distinct from each other and from \( j \)) one \( q \) is \( s \) and
   - (ii) \( 1 \leq j \leq k \) 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**

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

**Agreement - 1**

sender \( s \) in round 0:
0. extract \( m \)
1. sender \( s \) in round 1:
   1. broadcast \((s, m, 1)\)
   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 \( k \) accepts \((q, m, j)\) \( 1 \leq j \leq k \) in rounds 1 through \( k \):
      - (where \( i \) is distinct from each other and from \( j \)) one \( q \) is \( s \) and
      - (ii) \( 1 \leq j \leq k \) 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

**Agreement - 1**

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 \( p \).

- 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 11:
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 \( p \).

\( \exists r \geq 0 \) such that \( p \) will extract broadcast\((m, r+1)\) in round \( r+1 \). By CORRECTNESS, all correct processes will execute broadcast\((m, r+1)\) in round \( r+1 \) and extract \( m \).

If \( r = 0 \), then \( p \) and \( p \) will execute broadcast\((m, 0)\) in round \( 0 \). By CORRECTNESS, all correct processes will execute broadcast\((m, 0)\) in round \( 0 \) and extract \( m \).

If \( r > 0 \), the sender is faulty. Since \( p \) has extracted \( m \), \( r \) has executed at least \( r \) triples with properties (i), (ii), and (iii) but not previously extracted \( m \).

\( \exists \in \mathbb{R} \) such that \( p \) will extract broadcast\((m, r+1)\) in round \( r+1 \).

By CORRECTNESS, any correct process other than \( p \) will have accepted \( r+1 \) triples \((q, m, j), 1 \leq j \leq r+1\) by round \( r+1 \).

Lemma 12:
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 \( p \).

\( \exists r \geq 0 \) such that \( p \) will extract broadcast\((m, r+1)\) in round \( r+1 \). By CORRECTNESS, all correct processes will execute broadcast\((m, r+1)\) in round \( r+1 \) and extract \( m \).

If \( r = 0 \), then \( p \) and \( p \) will execute broadcast\((m, 0)\) in round \( 0 \). By CORRECTNESS, all correct processes will execute broadcast\((m, 0)\) in round \( 0 \) and extract \( m \).

If \( r > 0 \), the sender is faulty. Since \( p \) has extracted \( m \), \( r \) has executed at least \( r \) triples with properties (i), (ii), and (iii) but round \( r \).

\( \exists \in \mathbb{R} \) such that \( p \) will extract broadcast\((m, r+1)\) in round \( r+1 \).

By CORRECTNESS, any correct process other than \( p \) will have accepted \( r+1 \) triples \((q, m, j), 1 \leq j \leq r+1\) by round \( r+1 \).
Agreement - 2

Claim: $q_1, q_2, \ldots, q_l$ are all faulty.

Suppose $m$ were correct.

By UNFORGEABILITY, $q_1$ executed broadcast $(s, m, j_1)$ in round $j_1 < r$

$\Rightarrow$ $\exists q_2, \ldots, q_l$ extracted $m$ in round $j_{2, \ldots, l} < r$

CONTRADICTION

Case 2: $r = f+1$

Since there are at most $f$ faulty processes, some process $q_1$ in $(q_1, q_2, \ldots, q_{f+1})$ is correct.

By UNFORGEABILITY, $q_1$ executed broadcast $(s, m, j_1)$ in round $j_1 < r$

$\Rightarrow$ $m$ has extracted $m$ in round $j_{2, \ldots, l} < f+1$

CONTRADICTION

Implementing broadcast and accept

- 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.
  - If a process receives enough witness confirmations, it accepts $m$.

Validity

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

Can we rely on witnesses?

- Only if not too many faulty processes!
- Otherwise, a set of faulty processes could fool a correct process by acting as witnesses of a message that was never broadcast.
- 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...

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

Phase 2\(r-1\)
1: \(p\) sends \((\text{init}, p, m, r)\) to all
Phase 2\(r\)
2: if \(q\) received \((\text{init}, p, m, r)\) in phase 2\(r-1\) then
3: \(q\) sends \((\text{echo}, p, m, r)\) to all /* \(q\) becomes a witness */
4: if \(q\) receives \((\text{echo}, p, m, r)\) from at least \(n-f\) distinct processes in phase 2\(r\) then
5: \(q\) accepts \((p, m, r)\)
Phase \(f > 2r\)
6: if \(q\) has received \((\text{echo}, p, m, r)\) from at least \(f+1\) distinct processes in phases \(2r, 2r+1, \ldots, f-1\) then
7: \(q\) sends \((\text{echo}, p, m, r)\) to all processes /* \(q\) becomes a witness */
8: if \(q\) has received \((\text{echo}, p, m, r)\) from at least \(n-f\) processes in phases \(2r, 2r+1, \ldots, f\) then
9: \(q\) accepts \((p, m, r)\)

Is termination a problem?

The implementation is correct

Theorem

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

Assumption
Channels are authenticated

Correctness

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

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

Unforgeability - 1

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

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

Unforgeability - 2

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

Relay

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

Case 1: \( k' = 2r - 1 \)

1. \( q' \) received \((\text{init}, p, m, r)\) from \( p \).
2. Since \( p \) is correct, it follows that \( p \) did indeed execute broadcast\((p, m, r)\) in round \( r \).

Case 2: \( k' > 2r - 1 \)

1. \( q' \) has become a witness by receiving \((\text{echo}, p, m, r)\) from \( f + 1 \) distinct processes.
2. At most \( f \) are faulty; one is correct.
3. This process was a witness to \((p, m, r)\) before phase \( k' \).

Contradiction

The first correct process receives \((\text{init}, p, m, r)\) from \( p \)!
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 \( j + 1 \).

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

Taking a step back...

Specified Consensus and TRB

In the synchronous model:
- solved Consensus and TRB for General Omission failures
- proved lower bound on rounds required by TRB
- solved TRB for AFMA
- proved lower bound on replication for solving TRB with AF
- solved TRB with AF