Agreement – 3

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

Let $failed(p)$ be the set of processes that have failed to send a message to $p$ in any round.

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

1. if $p = sender$ then value := $m$, else value := $?$

2. send value to all

3. if value $\neq ?$ and delivered $m$ in round $k-1$ then halt

4. receive round $k$ values from all

5. faulty(p, k) := faulty(p, k-1) $\cup \{p\}$

received no value from $p$ in round $k$

6. if received value $\neq ?$ 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 – 3

Let $r$ be the earliest round in which a correct process delivers value := SF.

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

1. if $p = sender$ then value := $m$, else value := $?$

2. send value to all

3. if value $\neq ?$ and delivered $m$ in round $k-1$ then halt

4. receive round $k$ values from all

5. faulty(p, k) := faulty(p, k-1) $\cup \{p\}$

received no value from $p$ in round $k$

6. if received value $\neq ?$ 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

Proof

If no correct process ever receives $m$, then every correct process delivers SF in round $f+1$.

Integrity

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

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

1. if $p = sender$ then value := $m$, else value := $?$

2. send value to all

3. if value $\neq ?$ and delivered $m$ in round $k-1$ then halt

4. receive round $k$ values from all

5. faulty(p, k) := faulty(p, k-1) $\cup \{p\}$

received no value from $p$ in round $k$

6. if received value $\neq ?$ 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

Integrity

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

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

1. if $p = sender$ then value := $m$, else value := $?$

2. send value to all

3. if value $\neq ?$ and delivered $m$ in round $k-1$ then halt

4. receive round $k$ values from all

5. faulty(p, k) := faulty(p, k-1) $\cup \{p\}$

received no value from $p$ in round $k$

6. if received value $\neq ?$ 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

Proof

If no correct process ever receives $m$, then every correct process delivers SF in round $f+1$.

At most one $m$

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

- If $m \neq SF$, 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 \( \alpha \).

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 \( \alpha_1|p_i = \alpha_2|p_i \).
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 $\text{dec}(\alpha_1) = \text{dec}(\alpha_2)$

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 $\alpha_1 = \beta_1 \sim_{p_i} \beta_2 \sim_{p_i} \ldots \sim_{p_i} \beta_{k+1} = \alpha_2$.

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 $\text{dec}(\alpha_1) = \text{dec}(\alpha_2)$.
There is no algorithm that solves consensus in fewer 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.

**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

\[ \ldots \]

\[ \alpha_0 \]

\[ \alpha_1 \]

\[ \alpha_{n-1} \]

Indistinguishability

\[ \ldots \]

\[ \alpha_0 \]

\[ \alpha_1 \]

\[ \alpha_{n-1} \]
Indistinguishability

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

\[ \alpha_i \approx \beta_{n-i} \approx \alpha_{n-i} \]

\[ \alpha_i \approx \beta_{n-i+1} \approx \alpha_{n-i} \]

\[ \alpha_i \approx \beta_{n-i+2} \approx \alpha_{n-i} \]

\[ \alpha_i \approx \beta_{n-i+3} \approx \alpha_{n-i} \]
Indistinguishability

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

\[ \alpha_i \approx \beta_0 \]

Indistinguishability

Arbitrary failures with message authentication

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

- Process can send conflicting messages to different receivers
- Messages signed with unforgeable signatures
Valid messages

A valid message $m$ has the following form:

in round 1:

$$m : s_{id} \quad (m \text{ is signed by the sender})$$

in round $r > 1$, if received by $p$ from $q$:

$$m : p_1 : p_2 : \ldots : p_r \quad \text{where}$$

- $p_1$ = 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,
  - it “extracts” the value from the message
  - it 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

AFMA: The Protocol

Initialization for process $p$:

- if $p = \text{sender and } p \text{ wishes to broadcast } m$ then
  - extracted := relay := \{} m \}\$

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

- for each $s \in \text{relay}$
  - 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 $m \notin \text{extracted}$ then
    - extracted := extracted $\cup \{ m \}$
    - relay := relay $\cup \{ s \}$

At the end of round $f+1$

- if $\exists m \text{ such that } \text{extracted} = \{ m \}$ then
  - deliver $m$
- else deliver SF

Termination

In round $f+1$, every correct process delivers either $m$ or SF and then halts
Agreement

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 \( p \) is the sender, then in round \( 1 \), \( p \) sends a valid message to all.

All correct processes extract that message in round \( 1 \)

• otherwise, \( p \) has received in round \( r \) a message

Claim: \( m, p_1, p_2, \ldots, p_r \) are all faulty

– true for \( m \)

– Suppose \( m, p_1, p_2, \ldots, p_r \) were correct

• \( p_1 \) signed and relayed message in round \( j \)

• \( p_i \) extracted message in round \( j-1 \)

CONTRACTION

• If \( r \leq f \), \( p \) will send a valid message \( m, p_1, p_2, \ldots, p_r, p \)

in round \( r+1 \leq f+1 \) and every correct process will extract it in round \( r+1 \leq f+1 \)

• If \( r = f+1 \), by Claim above, \( p_1, p_2, \ldots, p_r \) faulty

• At most \( f \) faulty processes

• CONTRACTION

Validity

From Agreement and the observation that the sender, if correct, delivers its own message.

TRB for arbitrary failures

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

AF: The Idea

.identifier

- 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

```plaintext
sender s in round 0:
0: extract m

sender s in round 1:
1: broadcast(s, m, 1)
Process p in round k, 1 \leq k \leq f+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 accept(q, m, j) 1 \leq j \leq k in rounds 1 through k
   (where (i) \(q\) distinct from each other and from \(y\), (ii) one \(q\) is \(s\), and
   (iii) 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

Lemma
If a correct process extracts \( m \), then every correct process eventually extracts \( m \).

Agreement - 2

Claim: \( q_1, q_2, \ldots, q_f \) are all faulty
- Suppose \( q_i \) were correct
- \( q_i \) has accepted \((s, m, k)\) in round \( j \) \( \leq f \)
- By UNFORGEABILITY, \( q_i \) executed broadcast \((s, m, k)\) in round \( j \)
- \( q_i \) extracted \( m \) in round \( j \), \( j < r \)

CONTRACTION

Case 2: \( r = f + 1 \)
- Since there are at most \( f \) faulty processes, some process \( q_i \) in \( q_1, q_2, \ldots, q_f \) is correct
- By UNFORGEABILITY, \( q_i \) executed broadcast \((s, m, k)\) in round \( j \) \( \leq r \)
- \( q_i \) has extracted \( m \) in round \( j \), \( j < f + 1 \)

CONTRACTION

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 \( r \leq f + 1 \)
- By UNFORGEABILITY, and because \( s \) is correct, no correct process can extract \( m' \neq m \)

All correct processes will deliver \( m \)
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 $m$

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 \( n \leq 3f \)