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 are signed with unforgeable signatures
Valid messages

A **valid** message $m$ has the following form:

in round 1:

$$< m : s > \quad (m \text{ is signed by the sender})$$

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

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

- $p_1 = \text{sender}; \quad p_r = q$
- $p_1, \ldots, p_r$ are distinct from each other and from $p$
- message has not been tampered with
AFMA: The Idea

- A correct process p discard 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, delivers SF
AFMA: The Protocol

sender s in round 0:
1:  extract m

sender in round 1:
2:  send < m:s > to all

Process p in round k, 1 ≤ k ≤ f+1
3:  if p extracted m from a valid message < m:p1: ... :pk-1> in round k – 1 and p ≠ sender then
4:  send < m:p1: ... :pk-1:p> to all
5:  receive round k messages from all processes
6:  for each valid round k message < m:p1: ... :pk-1:pk> received by p
7:  if p has not previously extracted m then
8:  extract m
9:  if k = f+1 then
10:  if in the entire execution p has extracted exactly one m then
11:    deliver(m)
12:  else deliver(SF)
13:  halt
Termination

sender s in round 0:
1:   extract m
sender in round 1:
2:   send < m:s > to all
Process p in round  k, 1 ≤ k ≤ f+1
3:   if p extracted m from a valid message <m:p_1: ... :p_k-1>
in round k - 1 and p ≠ sender then
    4:       send <m:p_1: ... :p_k-1:p> to all
5:   receive round k messages from all processes
6:   for each valid round k message < m:p_1: ... :p_k-1:p_k>
        received by p
7:     if p has not previously extracted m then
8:       extract m
9: if k = f+1 then
10:   if in the entire execution p has extracted exactly
        one m then
     11:       deliver(m)
12:   else deliver(SF)
13:   halt

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

sender s in round 0:
1: extract m
sender in round 1:
2: send < m:s > to all
Process p in round k, 1 ≤ k ≤ f+1
3: if p extracted m from a valid message <m:p1: ... :pk-1>
in round k - 1 and p ≠ sender then
4: send <m:p1: ... :pk-1:p> to all
5: receive round k messages from all processes
6: for each valid round k message < m:p1: ... :pk-1:pk>
   received by p
7: if p has not previously extracted m then
8: extract m
9: if k = f+1 then
10: if in the entire execution p has extracted exactly
    one m then
11: deliver(m)
12: else deliver(SF)
13: halt

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 message in round 1
• otherwise, p has received in round r a message
   < m:p1:p2: ... :Pr >
• Claim: p1, p2, ..., pr are all faulty
  – true for p1 = s
  – Suppose pj, 1 ≤ j ≤ r, were correct
  • pj signed and relayed message in round j
  • pj extracted message in round j - 1
    CONTRADICTION
• If r ≤ f, p will send a valid message
  < m:p1:p2: ... :Pr>p >
in round r + 1 ≤ f + 1 and every correct process will extract it in round r + 1 ≤ f + 1
• If r = f + 1, by Claim above, p1, p2, ..., pf+1 faulty
  – At most f faulty processes
  – CONTRADITION
Validity

sender s in round 0:
1: extract m
sender in round 1:
2: send <m:s> to all
Process p in round k, 1 \leq k \leq f+1
3: if p extracted m from a valid message <m:p_1: \ldots :p_{k-1}>
in round k - 1 and p \neq sender then
4: send <m:p_1: \ldots :p_{k-1}:p> to all
5: receive round k messages from all processes
6: for each valid round k message <m:p_1: \ldots :p_{k-1}:p_k>
   received by p
7: if p has not previously extracted m then
8: extract m
9: if k = f+1 then
10: if in the entire execution p has extracted exactly
    one m then
11: deliver(m)
12: else deliver(SF)
13: halt

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

sender s in round 1:
1: broadcast (s,m,1)
Process p in round k, 1 ≤ k ≤ f + 1
2: if p extracted m in round k - 1 and p ≠ sender then
4: broadcast (p,m,k)
5: if p has executed at least k accept(qi,m,ji) 1 ≤ i ≤ k in rounds 1 through k
    (where (i) qi distinct from each other and from p, (ii) one qi is s, and
    (iii) 1 ≤ j ≤ 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

sender s in round 0:
0: extract m

sender s in round 1:
1: broadcast (s,m,1)

Process p in round k, 1 ≤ k ≤ f+1
2: if p extracted m in round k - 1 and p ≠ sender then
4: broadcast (p,m,k)
5: if p has executed at least k accept(qi,m,ji) 1 ≤ i ≤ k in rounds 1 through k
   (where (i) qi distinct from each other and from p, (ii) one qi is s, and (iii) 1 ≤ ji ≤ 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

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
sender s in round 1:
1: broadcast (s,m,1)

Process p in round k, 1 ≤ k ≤ f + 1
2: if p extracted m in round k - 1 and p ≠ sender then
4: broadcast (p,m,k)
5: if p has executed at least k accept(q_i,m,j_i) 1 ≤ i ≤ k in rounds 1 through k
   (where (i) q_i distinct from each other and from p, (ii) one q_i is s, and (iii) 1 ≤ j_i ≤ 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

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

☐ if r > 0, the sender is faulty. Since p has extracted m in round r, p has accepted at least r triples with properties (i), (ii), and (iii) by round r

☐ r ≤ f By RELAY, all correct processes will have accepted those r triples by round r + 1

☐ p will execute broadcast(p,m,r + 1) in round r + 1

☐ By CORRECTNESS, any correct process other than p, q_1, q_2,...,q_r will have accepted r + 1 triples (q_k,m,j_k), 1 ≤ j_k ≤ r + 1, by round r + 1

☐ q_1, q_2,...,q_r are all distinct

☐ every correct process other than q_1, q_2,...,q_r will extract m

☐ p has already extracted m; what about q_1, q_2,...,q_r?

Lemma

If a correct process extracts m, then every correct process eventually extracts m
Agreement - 2

sender s in round 0:
0: extract m

sender s in round 1:
1: broadcast (s,m,1)

Process p in round k, 1 ≤ k ≤ f+1
2: if p extracted m in round k - 1 and p ≠ sender then
4: broadcast (p,m,k)
5: if p has executed at least k accept(qi,m,ji) 1 ≤ i ≤ k in rounds 1 through k
    (where (i) qi distinct from each other and from p, (ii) one qi is s, and (iii) 1 ≤ ji ≤ 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

Claim: q1, q2, ..., qr are all faulty

> Suppose qk were correct
> p has accepted (qk, m, jk) in round jk ≤ r
> By UNFORGEABILITY, qk executed broadcast (qk, m, jk) in round jk
> qk extracted m in round jk-1 < r

CONTRACTION

□ Case 2: r = f + 1

□ Since there are at most f faulty processes, some process qk in q1, q2, ..., qf+1 is correct
□ By UNFORGEABILITY, qk executed broadcast (qk, m, jk) in round jk ≤ r
□ qk has extracted m in round jk-1 < 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 \(i \leq f + 1\)

By UNFORGEABILITY, and because \(s\) is correct, no correct process can extract \(m' \neq m\)

All correct processes will deliver \(m\)