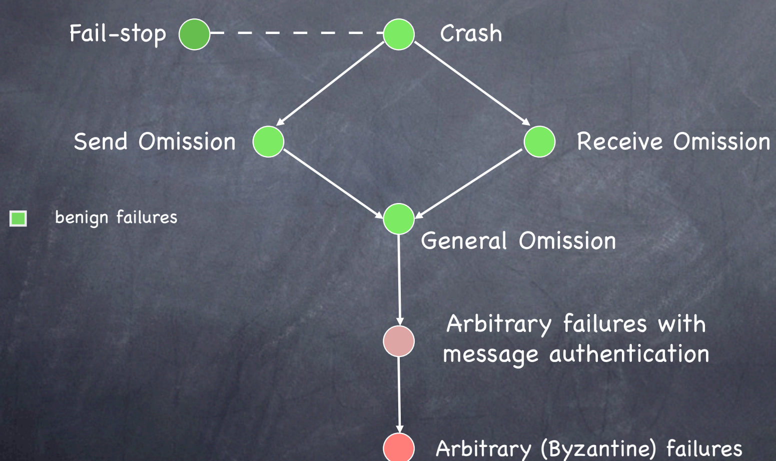
The Long March of BFT

Lorenzo Alvisi
UT Austin

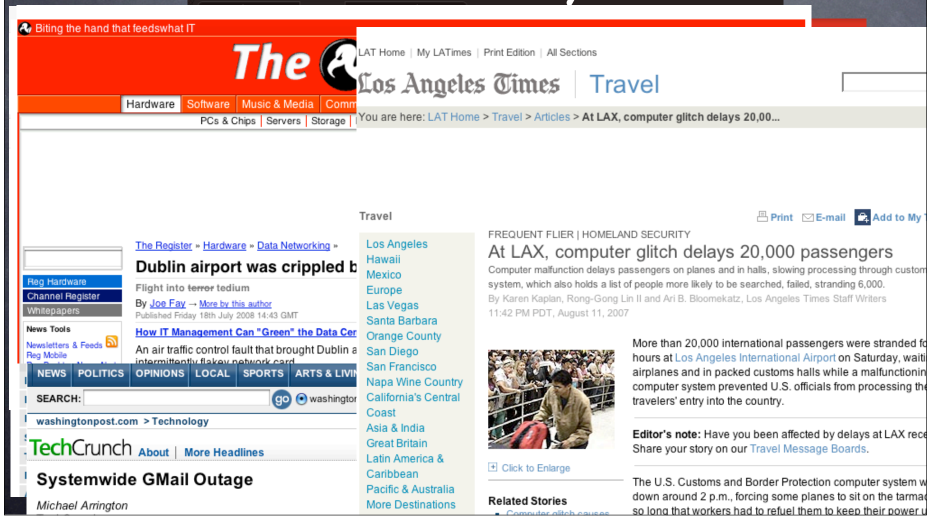
A specter is haunting the system's community...

A hierarchy of failure models



Weird Things Happen in Distributed Systems

Weird Things Happen in Distributed Systems



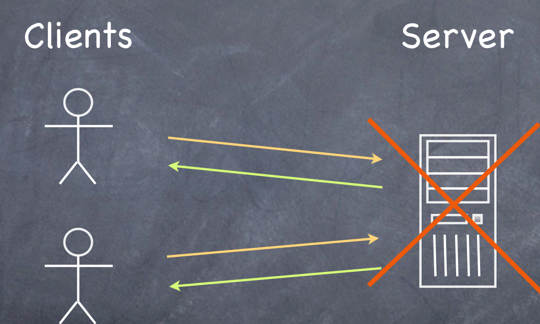
Why BFT?

- Systems are not fail-stop
  - insider attacks, soft errors, bugs...
- Assumptions are vulnerabilities!
- Hardware gets cheaper/data gets more valuable
  - Google FS already uses 3-way replication
- Lean and mean BFT systems have been built

Outline

- How it all began: BFT in synchronous systems
- FLP: Elaborating the grief
- A new dawn: Practical Byzantine Fault Tolerance
- Citius, Altius, Fortius

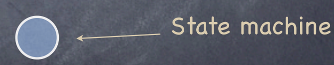
The Problem



Solution: replicate server!

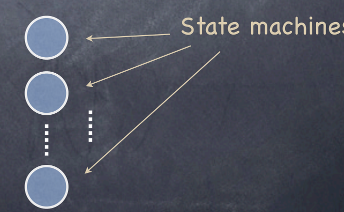
State Machine Replication

1. Make server deterministic (state machine)



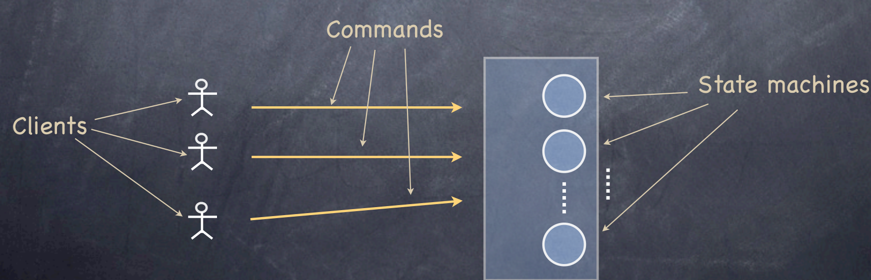
State Machine Replication

1. Make server deterministic (state machine)
2. Replicate server



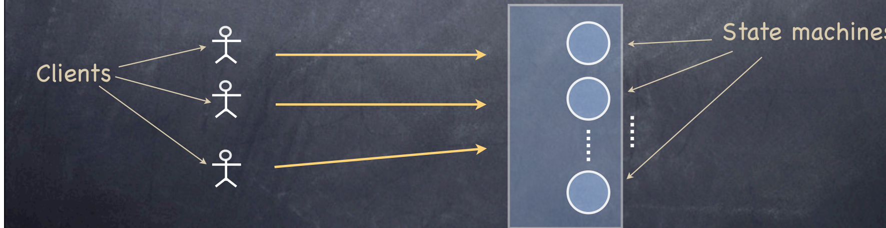
State Machine Replication

1. Make server deterministic (state machine)
2. Replicate server
3. Ensure correct replicas step through the same sequence of state transitions



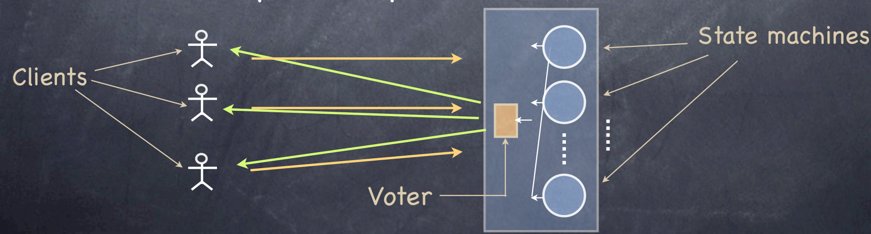
State Machine Replication

1. Make server deterministic (state machine)
2. Replicate server
3. Ensure correct replicas step through the same sequence of state transitions
4. Vote on replica outputs for fault-tolerance

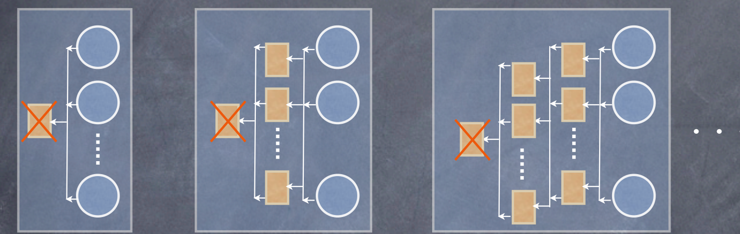


State Machine Replication

1. Make server deterministic (state machine)
2. Replicate server
3. Ensure correct replicas step through the same sequence of state transitions
4. Vote on replica outputs for fault-tolerance

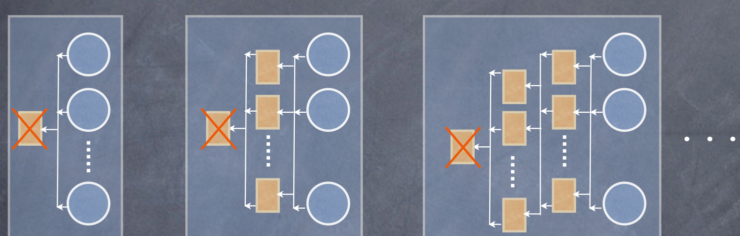


A conundrum

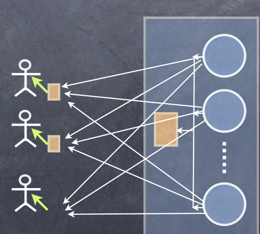


A: voter
and client
share fate!

A conundrum



A: voter
and client
share fate!



Replica Coordination

All non-faulty state machines receive all commands in the same order

- AGREEMENT: Every non-faulty state machine receives every command
- ORDER: Every non-faulty state machine processes the commands it receives in the same order

Terminating Reliable Broadcast

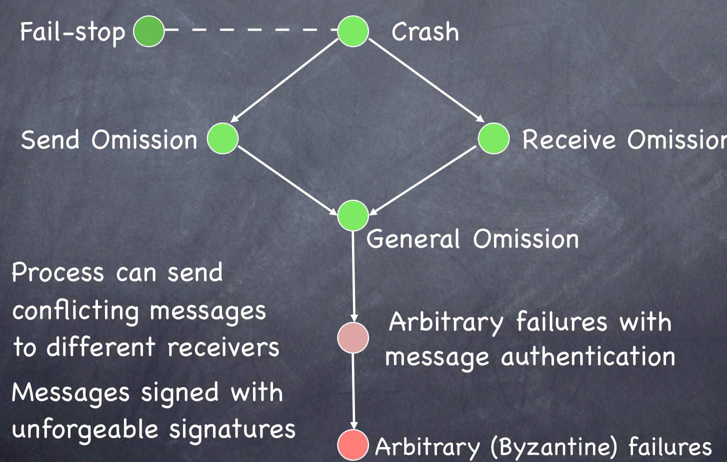
Validity If the sender is correct and broadcasts a message m, then all correct processes eventually deliver m

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

Arbitrary failures with message authentication



- 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:

$m : s_{id}$ (m 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 = 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 = 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 := $\emptyset$
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 = $\{m\}$ then
deliver m
else deliver SF

Termination

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 := $\emptyset$
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 = $\{m\}$ then
deliver m
else deliver SF

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

Agreement

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 := $\emptyset$
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 = $\{m\}$ then
deliver m
else deliver SF

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

- 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 $r = f+1$, p has received in round $f+1$ a message

$$m : p_1 : p_2 : \ldots : p_{f+1}$$

- Each $p_j, 1 \leq j \leq f+1$ has signed and relayed a message in round $j-1 < f+1$
- At most f faulty processes - one p_j is correct and has extracted m before p

CONTRADICTION

Agreement follows directly, since all correct process extract the same set of messages

Validity

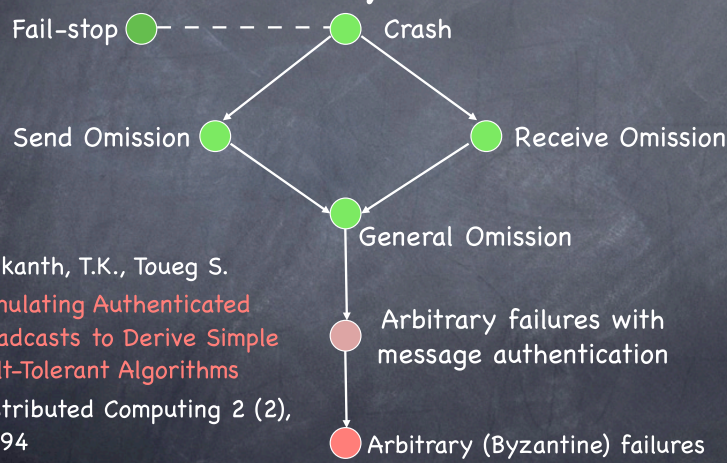
Initialization for process p :
if p = sender and p wishes to broadcast m then
extracted := relay := $\{m\}$

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 := $\emptyset$
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 = $\{m\}$ then
deliver m
else deliver SF

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

- 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

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_i, m, j_i) $1 \leq i \leq k$ in rounds 1 through k
(where (i) q_i distinct from each other and from p, (ii) one q_i is s, and
(iii) $1 \leq j_i \leq k$) 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

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_i, m, j_i) $1 \leq i \leq k$ in rounds 1 through k
(where (i) q_i distinct from each other and from p, (ii) one q_i is s, and (iii) $1 \leq j_i \leq k$)
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

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 $k, 1 \leq k \leq f+1$
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_i, m, j_i) $1 \leq i \leq k$ in rounds 1 through k
(where (i) q_i distinct from each other and from p, (ii) one q_i is s, and (iii) $1 \leq j_i \leq k$)
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

Lemma
If a correct process extracts m, then every correct process eventually extracts m

Agreement - 1

sender s in round 0:
0: extract m
sender s in round 1:
1: broadcast$(s, m, 1)$

Process p in round $k, 1 \leq k \leq f+1$
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_i, m, j_i) $1 \leq i \leq k$ in rounds 1 through k
(where (i) q_i distinct from each other and from p, (ii) one q_i is s, and (iii) $1 \leq j_i \leq k$)
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

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
- 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
  - $r \leq f$ 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 $p, q_1, q_2, \ldots, q_r$ will have accepted $r+1$ triples $(q_k, m, j_k), 1 \leq j_k \leq r+1$, by round $r+1$
  - $q_1, q_2, \ldots, q_r, p$ are all distinct
  - every correct process other than $q_1, q_2, \ldots, q_r, p$ will extract m
  - p already extracted m; what about $q_1, q_2, \ldots, q_r$?

Agreement - 2

sender s in round 0:
0: extract m
sender s in round 1:
1: broadcast$(s, m, 1)$

Process p in round $k, 1 \leq k \leq f+1$
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_i, m, j_i) $1 \leq i \leq k$ in rounds 1 through k
(where (i) q_i distinct from each other and from p, (ii) one q_i is s, and (iii) $1 \leq j_i \leq k$)
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

Claim: $q_1, q_2, \ldots, q_r$ are all faulty

- Suppose q_k were correct
- p has accepted(q_k, m, j_k) in round $j_k \leq r$
- By UNFORGEABILITY, q_k executed broadcast (q_k, m, j_k) in round j_k
- q_k extracted m in round $j_{k-1} < r$

CONTRADICTION

- Case 2: $r = f+1$
  - Since there are at most f faulty processes, some process q_l in $q_1, q_2, \ldots, q_{f+1}$ is correct
  - By UNFORGEABILITY, q_l executed broadcast(q_l, m, j_l) in round $j_l \leq r$
  - q_l has extracted m in round $j_{l-1} < f+1$

CONTRADICTION

Validity

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_i, m, j_i) $1 \leq i \leq k$ in rounds 1 through k
(where (i) q_i distinct from each other and from p, (ii) one q_i is s, and (iii) $1 \leq j_i \leq k$)
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

- 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

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

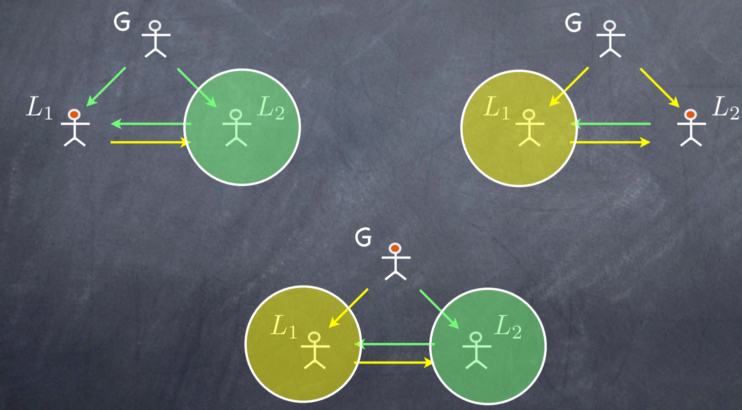
* * *

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 $n \leq 3f$

(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 $(init, p, m, r)$ to all
- A confirmation has the form $(echo, p, m, r)$
- A witness sends $(echo, p, m, r)$ if either:
  - ☐ it receives $(init, p, m, r)$ 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 $2r-1$

p sends $(init, p, m, r)$ to all

Phase $2r$

if q received $(init, p, m, r)$ in phase $2r-1$ then

    q sends $(echo, p, m, r)$ to all /* q becomes a witness */

if q receives $(echo, p, m, r)$ from at least $n-f$ distinct processes in phase $2r$ then

    q accepts (p, m, r)

Phase $j > 2r$

if q has received $(echo, p, m, r)$ from at least $f+1$ distinct processes in phases $(2r, 2r+1, \ldots, j-1)$ then

    q sends $(echo, p, m, r)$ to all processes /* q becomes a witness */

if q has received $(echo, p, m, r)$ from at least $n-f$ processes in phases $(2r, 2r+1, \ldots, j)$ then

    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 reliable (between correct processes) and 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

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 p is correct then

- p sends $(init, p, m, r)$ to all in round r (phase $2r-1$)
- by Validity of the underlying send and receive, every correct process receives $(init, p, m, r)$ in phase $2r-1$
- every correct process becomes a witness
- every correct process sends $(echo, p, m, r)$ in phase $2r$
- since there are at least $n-f$ correct processes, every correct process receives at least $n-f$ echoes in phase $2r$
- every correct process executes accept(p, m, r) in phase $2r$ (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

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

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

Case 1: $k' = 2r - 1$
- q' received $(init, p, m, r)$ from p
- since p is correct, it follows that p did execute broadcast(p, m, r) in round r

Case 2: $k' > 2r - 1$
- q' has become a witness by receiving $(echo, p, m, r)$ from $f+1$ distinct processes
- at most f are faulty; one is correct
- this process was a witness to (p, m, r) before phase k'

CONTRADICTION

The first correct process receives $(init, p, m, r)$ from p!

Summing up...

- For q to accept, some correct process must become witness.
- Earliest correct witness q' becomes so in phase $2r - 1$, and only if p did indeed executed broadcast(p, m, r)
- Any correct process that becomes a witness later can only do so if a correct process is already a witness.
- 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 $j \geq r$, then all correct processes will execute accept(p, m, r) by round $j + 1$

Relay

If a correct process q executes accept(p,m,r) in round $j \geq r$, then all correct processes will execute accept(p,m,r) by round $j+1$

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