BYZANTINE

FAULT

TOLERANCE

ANNOUNCEMENTS

* No office hours on 11/15, 11/22, 11/24

I am not in New York

Thanksgiving

* Will send out notes on project proposals by Sanders. Assume that they are fine.

What are Byzantine Failures?
Malicious messages sent by a process

- Behave arbitrarily

What causes Byzantine Failures

- De rigueur: Malicious users in large applications

- Bugs when implementing safety critical systems

Brief Excursion into Cryptography

- Digital Signatures
\[ y = \text{Sig}_{\text{key}}(\text{message}) \]

- Key: Secret known to only one node
- Given \( y \) anyone can check
  \[ \rightarrow \text{Generated from message} \]
  \[ \rightarrow \text{Generated by someone who has key} \]

- Usually implemented using public key cryptography

\[ \exists a, b \text{ s.t.} \]

\[ \text{dec}_b(\text{enc}_a(M)) = M \neq M \cdot \]

\[ \rightarrow \text{Known to all nodes} \]

key from above

Assume \( \cdot \) Given \( b \) finding \( a \) is

computationally infeasible

- Given \( M \neq b \) computing \( (M)_b \) is not possible
\( \text{enc}_a(M) \) is computationally infeasible

\( \langle \text{hash}(M), \text{enc}_a(M) \rangle \) acts as a signature.

- Hash-based Message Authentication Key
  
  \[ \text{HMAC}_{\text{KEY}}(M) \]

  - Key known to a set of nodes
  - Cannot compute \( \text{HMAC}_{\text{KEY}}(M) \) without key.
  - Generally faster to compute than digital signatures

How used?
Byzantine Failures, Cryptography, Bugs...
What does this have to do with last class?
- Want to protect against Byzantine failures
- Thus far: State machine replication for surviving fail-stop failures
- Today: RSM for Byzantine?

Remember: RSMs Depend on Consensus.
Start with Consensus in Byzantine Environments
Agreement: All correct nodes agree on the selected value (IC1)

Validity: If the leader is correct, then correct nodes accept values proposed by the leader.

Liveness: Byzantine nodes cannot prevent progress.

Lamport's Impossibility:

- Without authenticated channels, need 3f + 1 replicas to survive f failures.

- Authenticated Channel:

- Q: Asynchronous / Partially synchronous / Synchronous?
OM: Example of a BFT consensus protocol

Q1: Assumed processing model?
   - Unauthenticated

Requirement:

\[ \text{OM}(f) \text{ Up to } f \text{ faulty nodes, } 3f+1 \text{ total nodes.} \]

- At each process, \( \text{OM}(f) \) produces vector \( (v_0, \ldots, v_n) \) where \( n=2f+1 \)
  such that any two correct processes i, j agree on a majority of values.
OM(0)

\[ f = 0 \Rightarrow \text{No faulty node} \]
\[ \Rightarrow \text{Everyone can trust all messages.} \]

OM(1)

1. \[ R_0: V_L, V_{R_0}, V_{R_2} \] (At most one of \( L, R_0, R_1, R_2 \))
   \[ R_1: V_L, V_{R_0}, V_{R_2} \]
   \[ R_2: V_L, V_{R_0}, V_{R_1} \] (faulty)

2.
Case split:

* 1 faulty

\[ R_0: x \quad y \quad z \]
\[ R_1: y \quad x \quad z \]
\[ R_2: z \quad x \quad y \]

* \( R_0 \) (or \( R_1 \) or \( R_2 \)) faulty

\( R_0 \) send \( y \) to \( R_1 \); \( z \) to \( R_2 \)

\[ R_1 \quad x \quad y \quad x \]
\[ R_2 \quad x \quad z \quad x \]

* So we know:

\( \text{OM}(0) \) works \ [\text{Trivially}] \n
\( \text{OM}(1) \) works by demo above

Should show \( \text{OM}(f) \) works.

How? Induction: Assume \( \text{OM}(f-1) \) works

\[ \Gamma \text{satisfies agreement/IC1} \quad \Gamma \text{valid/IC2} \]
First \( \text{OM}(f) \)

1. Leader sends value to all followers.

\[ \text{OM}_i(f-1) \]

2. Follower \( i \) invokes \( \text{OM}(f-1) \) where \( k \) is leader & there are \( n-2 \) followers.

3. Each follower collects values from \( n-2 \):
\[ \text{OM}_i: (v_k, v_l, \ldots, v_n) \]
and takes majority.

Validity: Requires correct leader.

\[ \text{Lemma 1: If at most } f \text{ failures, leader is correct & } n > 2f + m \text{ then } \text{OM}(m) \text{ meets validity.} \]

\[ \text{Proof: } \text{OM}(0) \text{ trivially true.} \]

Channels are reliable.

- Assume true for \( \text{OM}(m-1) \& \)

leader sends value \( v \).

\[ \Rightarrow \text{At least } n-1-f \geq f + (m-1) \]

since \( \text{OM}(1) \) are run
copies of \( \Omega(m-1) \) with \((n-1)\) processes to send \( v \).

- But \((n-1) > 2f + (m-1)\)

  \( \Rightarrow \) By inductive hypothesis each process gets at least \( f + (m-1) \) copies of \( v \).

- But \( f + (m-1) \) is a majority

  \( \Rightarrow \) Choose \( v \).

\[ \text{Lemma 1 with } m = f \Rightarrow \text{Validity.} \]

**Agreement:**

- If leader is correct, this follows from Lemma 1.

- Only need to consider faulty leader

  Leader is faulty \( \Rightarrow \) at most \( f-1 \) faulty followers
Correct followers run $OM(f-1)$ with $n-1$ participants of which $f-1$ are faulty.

We know $n > 3f$

$\Rightarrow n-1 > 3(f-1)$

$\Rightarrow$ At least $(n-1)-(f-1)$ correct versions of $OM(f-1)$ are run & reach agreement.

$(n-1)-(f-1)$ is a majority of values $\Rightarrow OM(f)$ reaches agreement.

Bottom Line

Agreement & validity without authentication requires $3f+1$ participants.
Why Authentication Helps?

PBFI

- Uses authentication
  (Digital signatures OR HMACs)

- Still requires 3f+1 processes.

- Why?
Why we need at least $2f+1$?

$2f+1$  $f+1$

Why we need $3f+1$?

Protocol: "Paros" Like
Differences

- Every process counts decisions
- Signatures/HMAC to validate messages
- Every one responds to the client

Message Authentication

- Client requests are signed
  - Malicious leader cannot impersonate clients
- Pre-Prepare carries command but command & pre-prepare message signed separately

\[
\left[ \langle \text{PRE-PREPARE, VIEW, SLOT, HASH(c)}\rangle, c \right]
\]

Why?

- Client must wait for \( f+1 \) responses
Liveness?

Remember: Faulty nodes should not impede the algorithm's progress.

But, FLP => No termination.

What is going on here?

Achieving Liveness

Two problems

* Faulty leader might not send pre-prepare

* Faulty followers bunches view change frequently
Faulty Leaders

* Clients can timeout & broadcast

Signals that leader might be faulty

Q: Failure assumption about clients?

* Need to make sure non-faulty leaders can be elected.

Raft & Multi-Paxos

PBFT*: View number dictate leader
No more than $f$ view changes before correct leader

**Faulty Followers**

*Require $2f+1$ view change messages before changing*

$\rightarrow$ Faulty followers cannot unilaterally trigger change
PBFI in practice