CS 138: Byzantine Consensus
Consensus

• Raft election
  – Candidate: request vote
  – Follower: at most one vote

• Distribute transactions
  – Coordinator: request transaction
  – Nodes: Commit/Abort

• Current assumptions
  – Follower/Nodes don’t lie

• What happens if nodes can lie?
Consensus Background

• Nodes start in **undecided state**
• Nodes propose states to each other
• Nodes converge on the same value
• All node exchange their decisions with each other
  – The nodes in turn inform each of all the decisions they know
• All node exchange their decisions with each other
  – The nodes in turn inform each of all the decisions they know
Convergence/Termination

• How many rounds without failures?

• How many rounds with failures?
Evaluate Consensus

• Termination: nodes eventually converge on a decision

• Agreement: the decisions is the same across correct nodes

• Integrity: if all correct processes propose the same value, then any correct process decides the same value
How are properties achieved

- Reliable multicast $\rightarrow$ Termination
  - Eventually, all nodes will get the messages

- Agreement $\rightarrow$ count majority votes
  - Majority is the same function

- Integrity
  - Reliable Multicast to the rescue maintain integrity
    - TCP has Checksums to ensure messages are not tampered with
# Evaluating Protocols

<table>
<thead>
<tr>
<th></th>
<th>Termination</th>
<th>Integrity</th>
<th>Agreement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raft (leader Election)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distributed Transactions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total ordered Multicast</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Byzantine Generals Problem

Commanding General

Attack!

Lieutenant General

Lieutenant General

Attack!
Byzantine Generals Problem

Commanding General

Retreat!

Lieutenant General

Retreat!

Lieutenant General
Byzantine Generals Problem

• C1: All loyal lieutenant generals obey the same order

• C2: If the commanding general is loyal, then every loyal lieutenant general obeys the order she sends

• Note: if the commanding general is lying then Byzantine Generals doesn’t help reach consensus
<table>
<thead>
<tr>
<th></th>
<th>Termination</th>
<th>Integrity</th>
<th>Agreement</th>
</tr>
</thead>
<tbody>
<tr>
<td>consensus</td>
<td>Eventually set decision</td>
<td>All correct processes will have the same value</td>
<td>All correct process have the same decision</td>
</tr>
<tr>
<td>Byzantine general</td>
<td></td>
<td>If leading general is correct, then correct processes will have the same value</td>
<td></td>
</tr>
</tbody>
</table>
Byzantine Agreement Problem

• All generals co-equal
  – each general $i$ has a value $v(i)$ she sends to the others

1) Every loyal general must obtain the same information $v(1), \ldots, v(n)$

2) If the $i$th general is loyal, then the value she sends must be used by every loyal general as the value of $v(i)$
Byzantine Generals Problem

Attack!

Retreat!
Byzantine Generals Problem

Attack!

Retreat!

She said Attack!

She said Retreat!
Byzantine Generals Problem

Attack!

She said Attack!

She said Retreat!
When $f = 1$, and $N = 3$ the correct processes can’t differentiate between a good or bad general.
Summing Up

• Byzantine Generals Problem with 3 Generals, at most one of whom is a traitor ([3,1]BGP)
  – no solution satisfying C1 and C2
Message-Passing Assumptions

• Nodes only communicate through message-passing
  – Every message sent is delivered correctly
  – The receiver of a message knows who sent it
  – The absence of a message can be detected

• Hidden assumptions
  – No message signing (encryption)
  – Nodes can only listen to messages to/from them
  – Nodes can only modify messages they send
[4, 0] Byzantine Generals Problem

Attack!

Attack!

Attack!
[4,1] Byzantine Generals Problem

She said Attack!

Retreat!

She said Attack!

She said Attack!

She said Retreat!

She said Attack!

She said Retreat!

She said Attack!
[4,1] Byzantine Generals Problem

She said Attack!

She said Retreat!

She said Attack!

She said Retreat!

She said Attack!

She said Retreat!
Some Details

• Each general receives messages u, v, and w from the others
  – if no message is received, interpret its lack as “retreat”

• Loyal general takes its order to be $\text{majority}(u, v, w)$
  – if no majority: retreat
Summing Up

• Byzantine Generals Problem with 4 Generals, one of whom is a traitor ([4,1]BGP)
  – Solvable

• As long as the traitor is not the leader!
Theorem

- If \( N \) is the number of generals and \( T \) is the number of traitors, then there is a solution to the Byzantine Generals Problem iff

\[
N > 3T
\]
Byzantine V. Consensus

• Consensus
  – Survive $f$ failures
  – $N > 2f$

• Byzantine general
  – Survive $f$ failures
  – $N > 3f$
Signed Messages

Attack!

Attack!

Attack!

Attack!

Attack!
Signed Messages

Attack!

Retreat!

Attack!

Retreat!
Failures: Synch. V. Asynch.

He said "Attack!"

He said "Attack!"

Attack!

Attack!

Attack!

He said "Attack!"

He said "Attack!"

He said "Attack!"

He said "Attack!"

?
Asynchronous Communication

• Processes may respond to messages at arbitrary times
  – can’t use timeouts to determine failures

• BGP has no solution
  – non-responding general might respond at any time with whatever response counters the decision made assuming it was missing
  – in practice this is surmountable
Surmounting Failure

- Recover quickly
  - state kept in non-volatile memory

- Detect failure
  - enforced timeouts

- Be unpredictable
  - randomized algorithm
Byzantine Lecture Recap

• Discussed consensus properties
  – Termination, integrity, agreement
  – Analyzed Raft, Distributed transactions, and total ordered multicast within this context

• Discussed message passing assumptions
  – Impact of hidden assumptions on byzantine problem

• Byzantine failures
  – Lack of consensus in 2f+1 scenarios
  – Demonstrate of consensus in 3f+1 scenarios
  – Impact of message passing assumptions on byzantine failures
CS 138: Practical Byzantine Consensus
Scenario

- Asynchronous system
- Signed messages
- Servers are state machines
- It has to be practical
Ground Rules

• All messages, including client requests, are signed
  – messages cannot be modified
• Primary determines order: assigns sequence numbers
  – traitorous primary server could give different total orders to different backup servers
• A server could be unresponsive
  – delay does not affect safety
  – can affect liveness
    - exponential bound on how long delays may be
Faulty Servers

• $n > 3f$
  – assume $n = 3f + 1$

• Backup servers must have identical responses from $n-f$ servers (including primary) before they will trust primary

• New primary is chosen if current one is faulty (or non-responsive)
The Request

Client

Server 0

pre-prepare

Server 1

pre-prepare

Server 2

pre-prepare

Server 3

pre-prepare

Client

request
Non-Primaries Respond (1)
Non-Primaries Respond (2)

Client

Server 0

Server 1

prepare

prepare

Server 2

prepare

Server 3

Client
Non-Primaries Respond (3)
Servers Commit to Request (1)
Servers Commit to Request (2)