Distributed Systems

Day 15: Consensus + Byzantine Fault Tolerance
Mar 19, 2019
Examples of Consensus

• Raft election: agree on new leader
  • Candidate: request vote
  • Follower: at most one vote

• Distribute transactions: agree to commit/abort
  • Coordinator: request transaction
  • Nodes: Commit/Abort

• Current assumptions
  • Follower/Nodes don’t lie

• What happens if nodes can lie?
The Byzantine Generals Problem

INTRODUCTION

A reliable computer system must be able to cope with the failure of one or more of its components. A failed component may exhibit a type of behavior that is often overlooked—namely, sending conflicting information to different parts of the system. The problem of coping with this type of failure is expressed abstractly in the Byzantine Generals Problem. We devote the major part of the paper to a discussion of this abstract problem and conclude by indicating how our solutions can be used in implementing a reliable computer system.

We imagine that several divisions of the Byzantine army are camped outside an enemy city, each division commanded by its own general. The generals can communicate with one another only by messenger. After observing the enemy, they must decide upon a common plan of action. However, some of the generals...

The Byzantine Generals Problem - Microsoft Research
ACM Transactions on Programming Languages and ...

Solving the Byzantine Generals Problem with Proof of Work - Radix DLT
https://www.radixdlt.com/post/what-is-proof-of-work  
Apr 26, 2016 - Proof of Work (PoW) is the original blockchain consensus algorithm, and is used in projects such as Bitcoin and Ethereum.

What is Byzantine Generals Problem and How Blockchain solves it ...  
https://captainaltcoin.com/byzantine-generals-problem/  
Jan 17, 2019 - Anyone interested in cryptography has at some point heard of the “Byzantine Generals’ problem. This problem has been the main stepping ...

Byzantine Fault Tolerance and The Byzantine Generals' Problem ...
https://www.mycryptopedia.com › Byzantine Fault Tolerance  
Byzantine Fault Tolerance is the feature of a system that can tolerate the class of failures belonging to the Byzantine Generals' Problem.
Outline

• Consensus
  • Analyzing consensus protocols

• Byzantine Fault Tolerance problem
  • Byzantine generals problem
  • Solving the BFT problem
  • Solving BFT with security
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
What is a Good Consensus Algorithm?

- All nodes exchange their decisions with each other
  - The nodes in turn inform each of all the decisions they know
Evaluating Consensus Algorithms

- **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

<table>
<thead>
<tr>
<th></th>
<th>Termination</th>
<th>Integrity</th>
<th>Agreement</th>
</tr>
</thead>
<tbody>
<tr>
<td>consensus</td>
<td>Eventually set</td>
<td>All correct processes will have the same value</td>
<td>All correct process have the same decision</td>
</tr>
<tr>
<td></td>
<td>decision</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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>
</tbody>
</table>

Candidate

receive votes from majority of servers

Leader


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>
</tbody>
</table>

Coordinator

- Init
  - app commit/vote req
  - any abort/abort
  - all commit/commit
  - Wait
    - Abort
    - Commit

Participant

- Init
  - vote req/commit
  - vote req/abort
  - Abort
  - Commit
  - Uncertain
    - abort/ack
    - commit/ack
# 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>
</tbody>
</table>

- Recover quickly
  - state kept in non-volatile memory

- Detect failure
  - enforced timeouts

- Be unpredictable
  - randomized algorithm
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>• Terminates with majority</td>
<td>Only vote for one candidate</td>
<td>Candidates Apply Majority on votes</td>
</tr>
<tr>
<td></td>
<td>• Use randomness to ensure progress</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distributed Transactions</td>
<td>Can block with Coordinator failure!</td>
<td>Once vote, do not change vote</td>
<td>Coordinator Counts votes</td>
</tr>
<tr>
<td></td>
<td>Without failure eventual terminates</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Coordinator

- **Init**
  - app commit/vote req

Wait

- any abort/abort
- all commit/commit

- Abort
- Commit

Participant

- **Init**
  - vote req/commit

Uncertain

- vote req/abort
- abort/ack

- Abort
- Commit

- Coordinator
- Participant
Byzantine General Problem
Byzantine Generals Problem

Commanding General

Lieutenant General

Lieutenant General

Attack!
Byzantine Generals Problem

Retreat! Retreat!

Commanding General

Lieutenant General

Lieutenant General
Byzantine Generals Problem

- General sends msg to LT.
  - Each LT. copies and distribute the msg

- Lies:
  - General can lie: send different msg to each LT
  - LTs can lie: send msg different from what General provided them

Ideally, all participants are truth. However, any can lie: a lie means change the msg
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**
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**

In the presence of a lying general → goal is to detect that the general is a liar.
# Definition of Consensus Properties

<table>
<thead>
<tr>
<th></th>
<th>Termination</th>
<th>Integrity</th>
<th>Agreement</th>
</tr>
</thead>
<tbody>
<tr>
<td>consensus</td>
<td>Eventually set</td>
<td>All correct processes will have the same value</td>
<td>All correct process have the same decision</td>
</tr>
<tr>
<td></td>
<td>decision</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byzantine</td>
<td>If leading general is correct, then correct processes will have the same value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>general</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Definition of Consensus Properties

<table>
<thead>
<tr>
<th>Termination</th>
<th>Integrity</th>
<th>Agreement</th>
</tr>
</thead>
<tbody>
<tr>
<td>consensus</td>
<td>Eventually set decision</td>
<td>All correct process have the same value</td>
</tr>
<tr>
<td>Byzantine general</td>
<td>Eventually set decision</td>
<td>All correct process have the same decision</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If leading general is correct, then correct processes will have the same value</td>
</tr>
</tbody>
</table>

### Consensus Properties

- **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.

Definition of integrity is changed for byzantine style failures based on C2 of the problem.
Challenge: When $N=3$ (total # of nodes) and $f=1$ (liars) the correct processes can’t differentiate between a good or bad general.

Attack!

Retreat!

She said Attack!

She said Retreat!

Attack!

Attack!

Attack!

She said Attack!

She said Retreat!
**Consensus Properties**

- **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

When $f = 1$ (liars), and $N = 3$ (total # of nodes)
the correct processes can’t differentiate between a good or bad general

**Good LT** can’t differentiate between two scenarios.
3 options always pick leader, always pick other LT, or randomly
Consensus Properties

- **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.

When \( f = 1 \) (liars), and \( N = 3 \) (total # of nodes) the correct processes can’t differentiate between a good or bad general.

Good LT can’t differentiate between two scenarios.

3 options always pick leader, always pick other LT, or randomly.
Consensus Properties

- **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.

When \( f = 1 \) (liars), and \( N = 3 \) (total # of nodes) the correct processes can’t differentiate between a good or bad general.

Good LT can’t differentiate between two scenarios: 3 options always pick leader, always pick other LT, or randomly.
Byzantine Agreement Problem
**Byzantine Agreement Problem**

- All generals co-equal
  - each general $i$ has a value $v(i)$ she sends to the others
  - Take turns being the General and sending $v(i)$
  - **Run the Byzantine General Problem N times!!!!**

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

2) If the $i$th general is loyal, then the value she sends must be used by every loyal general as the value of $v(i)$
   1) A variation of property C2
Byzantine Agreement Problem

• All generals co-equal
  • each general $i$ has a value $v(i)$ she sends to the others
  • Take turns being the General and sending $v(i)$
  • Run the Byzantine General Problem $N$ times!!!!

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 $v(i)$
   1) A variation of property $C2$
Byzantine Agreement Problem

• All generals co-equal
  • each general \( i \) has a value \( v(i) \) she sends to the others
  • Take turns being the General and sending \( v(i) \)
  • Run the Byzantine General Problem \( N \) times!!!!

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 \( v(i) \)
   1) A variation of property \( C2 \)

Consensus Properties

• \( C1 \): 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

Retreat!
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

When f= 1 (liars), and N= 3 (total # of nodes)
the correct processes can’t differentiate between a good or bad general

This is important because in Raft (passive replication)
• Assume that cluster of 3 is good to recover from failure of one node
• Assume for f failures, 2N+1 is okay
• For example, with 3 nodes if one fails → 2 are alive
  • With two alive → there’s a majority → a new leader can be elected
BFT: Solving Byzantine General Problem
System 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

[4,0]
- 4 nodes
- 0 liars

Attack!

Attack!

Attack!
[4,1] Byzantine Generals Problem

[4,1]
• 4 nodes
• 1 liar
[4,1] Byzantine Generals Problem

[4,1]
- 4 nodes
- 1 liar

Bad LT
[4,1] Byzantine Generals Problem

**Consensus Properties**

- **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.

---

**Good LTs can’t distinguish bad Gen from Bad LT.**

**But Good LTs will still do the same thing. [Satisfy C1]**
- By looking at majority of messages Good LT will do same thing.

**When Gen is truthful “same thing” --> “leader’s order” [Satisfy C2]**
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 majority$(u, v, w)$
  - if no majority: retreat

- 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!
Some Details

- Each general receives messages \( u, v, \) and \( w \) from the others.

**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 \( N > 3T \) so \( N = 3T+1 \).

- As long as the traitor is not the leader!
Fault Recovery Byzantine V. Consensus

- **Consensus**
  - Survive $f$ failures
  - $N > 2f$:
    - so you need at least $N=2f+1$

- **Byzantine general**
  - Survive $f$ failures
  - $N > 3f$
    - so you need at least $N=3f+1$

How do you get ‘$f$’?
- Analyze failure distributions
- ‘what’s the max server failure at any time’

How do you get ‘$f$’?
- Analyze system security
- ‘what’s max server’s compromised at any time?’
BFT: Solving Byzantine General Problem with encryption
Use of Encryption + Signed Messages

- Assumptions\(^1\):
  - Only General can encrypt msg
  - Any one can decrypt msg

  [1] assumptions align with public-key cryptography
Use of Encryption + Signed Messages

• Assumptions\(^1\):
  • Only General can encrypt msg
  • Any one can decrypt msg
• Given assumptions:
  • General encrypts and sends msg to LT.
  • Each LT. copies and distribute the msg
    • Because LT does not have key – LT can decrypt but not alter the message
    • Thus, duplicates are identical to msg received from General
  • After receiving \((N-1)\) messages
    • Each LT decrypts all msgs and compares
    • If not identical --- general lied

\(^1\) assumptions align with public-key cryptography
Why Not Always use Encryption?

- Original BFT invented in ‘82
- No discussion of encryption
- RSA – invented in ‘77
- Probably still too slow
Outline

• Consensus  
  • Analyzing consensus protocols

• Byzantine Fault Tolerance problem
  • Byzantine generals problem  
    • Byzantine Agreement problem
  • Solving the BFT problem
  • Solving BFT with security
Byzantine Lecture Recap

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

• Discussed BGP
  • Consistency properties [C1 and C2 → implications]
  • 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