Distributed Consensus

GW Advanced Networking and Distributed Systems
Tim Wood and Lucas Chaufournier

How is it going?

How are you all doing?

Deadlines have been relaxed

- Submit by “Suggested Deadline” to get +5 point bonus
- Submit after “Final Deadline” and you lose 5 points per day
- If you have special circumstances that deserve an extension, contact me!
- Check Blackboard for empty grades and compare to Tasks page!

Is there anything else we can do to make the rest of our online semester more productive or effective for you?

- Quick poll

Outline

Last time:

- State machine replication
- f+1 replicas isn't enough if nodes can fail at any time
- Need 2f+1 to ensure there is always at least one node who was alive in a prior majority (consider case where f nodes fail, then another f nodes immediately fail… we need one more!)

This time:

- Overview of why reaching consensus (agreement on a value or action to perform) is difficult in distributed systems

Deadlines:

- ETL first deadline has passed, final deadline is April 26th
- Reading quiz on Consistency Models, due May 3rd
- Leader Election assignment will be posted soon

Two Generals Problem

Two generals are preparing to attack a city

- They will only succeed if both attack simultaneously

How can they coordinate their attack?

- Any messengers sent out might get captured!

General Sun Tzu

Target

General Washington

Two Generals Problem

Impossible to guarantee agreement in lossy network!

- So usually we will need to assume that network will eventually transmit, or loss can be detected

General Sun Tzu

Target

???

General Washington

Properties

Asynchrony: networks can have unbounded delay

Safety: all nodes agree on the state of the system

- nothing bad should happen

Liveness: progress is made on incoming requests

- something good should happen

Fault Tolerance: at least one node can fail

Properties

Asynchrony: networks can have unbounded delay

Safety: all nodes agree on the state of the system

- nothing bad should happen

Liveness: progress is made on incoming requests

- something good should happen

Fault Tolerance: at least one node can fail

FLP Impossibility Theorem: in an asynchronous network, you can only get 2 out of 3 properties

Sleepy Generals Problem*

Our general are tired, but messengers can't die!

Need **2** generals to be awake and attack for success

- If at most **f** generals can fall asleep at a time, how many general do we need?

* I made up this name

Bureaucratic Generals Problem*

Our general are tired, but messengers can't die!

Need **1** general to be awake and attack for success

Need to ensure that all paperwork is filled correctly!

- Need complete history of commands to attack (stateful system)

* I made up this name too

Traitorous Generals Problem

One of our generals is a traitor!

How to make majority of generals agree to attack?

* I made up this name too

Traitorous Generals Problem

One of our generals is a traitor!

How to make majority of generals agree to attack?

Need more than f+1 replicas!

Can't have a trusted primary anymore!

Replicas need to talk to each other to reach agreement on the decision

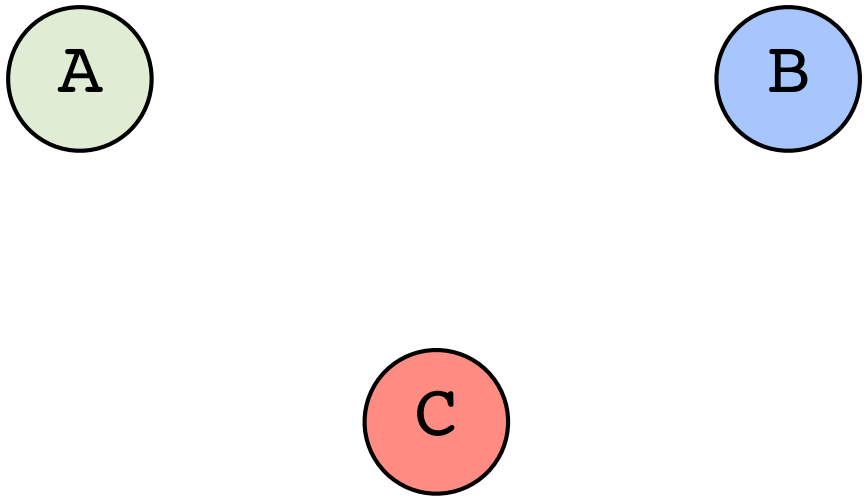
Vote and take the majority?

* I made up this name too

Reaching Agreement

The assault will only succeed if at least 2 armies attack at the same time

- I vote we should… 1 = attack, 0 = retreat!

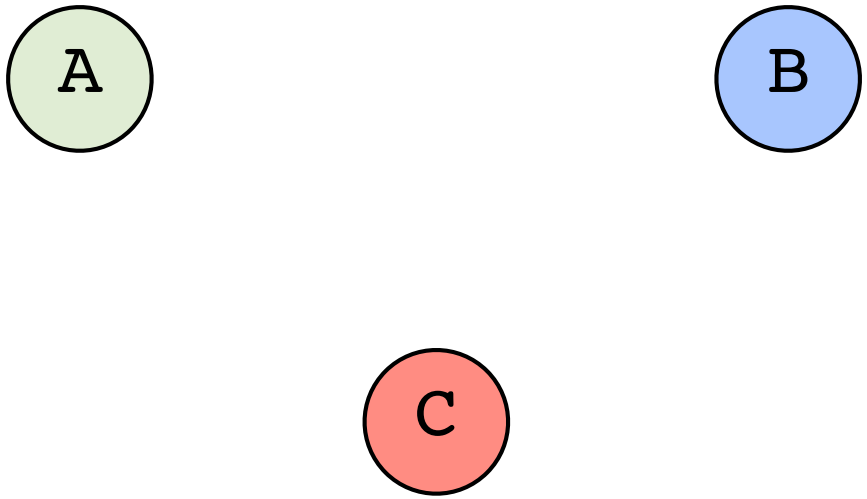


Replica	Receives	Action
A: 1		
B: 0		
C: 1		

Reaching Agreement

The assault will only succeed if at least 2 armies attack at the same time

- I vote we should… 1 = attack, 0 = retreat!

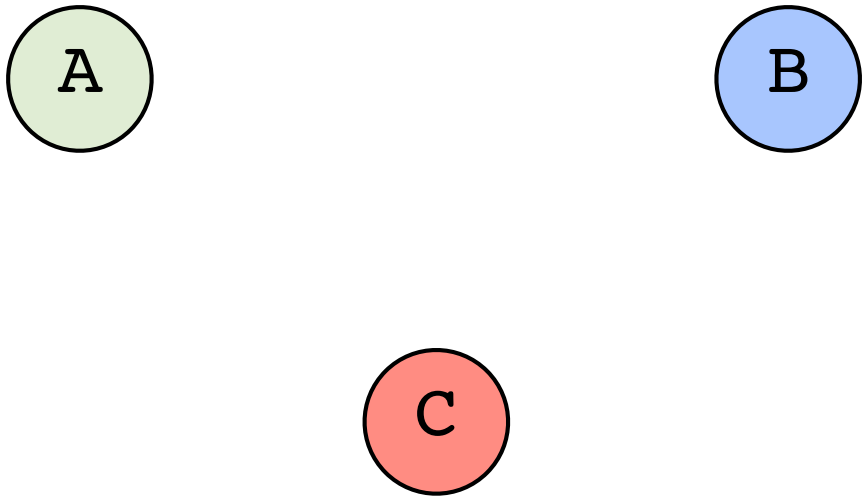


Replica	Receives	Action
A: 1		
B: 1		
C: 0		

Reaching Agreement

The assault will only succeed if at least 2 armies attack at the same time

- I vote we should… 1 = attack, 0 = retreat!



Replica	Receives	Action
A: 1		
B: 0		
C: ???		

[Lamport '82]

Byzantine Generals Solved*!

Need more replicas to reach consensus

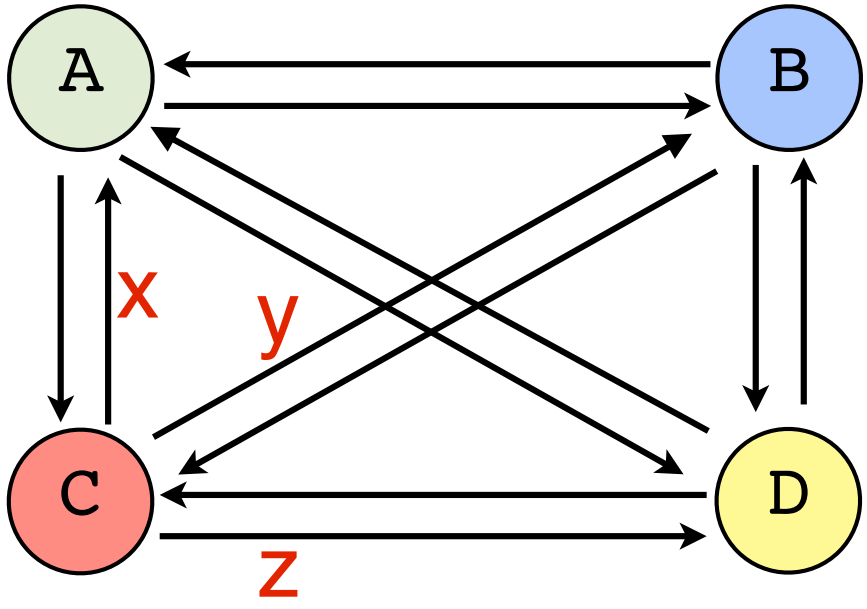
- Requires ***3f+1*** replicas to tolerate ***f*** byzantine faults

Step 1: Send your plan to everyone

Step 2: Send learned plans to everyone

Step 3: Detect conflicts and use majority

Replica	Receives	Majority
A	A: (1,0,1,1) B: (1,0,0,1) C: (1,1,1,1) D: (1,0,1,1)	A: 1 B: 1 C: 1 D: 1
B	A: (1,0,1,1) B: (1,0,0,1) C: (0,0,0,0) D: (1,0,1,1)	A: 1 B: 1 C: 0 D: 1



Problem Summary

Two Generals Problem

- If network can arbitrarily lose messages, then it is impossible to guarantee two (or more) nodes can reach agreement

Sleepy Generals Problem

- If **f** nodes can fail, you need _____ replicas to guarantee **x** correct responses from a **stateless** system (typically x=1)

Bureaucratic Generals Problem

- If **f** nodes can fail, you need _____ replicas to guarantee a correct response from a **stateful** system

Byzantine Generals Problem

- If **f** nodes can be arbitrarily malicious, you need _____ replicas to guarantee a correct response (stateful or stateless)

Problem Summary

Two Generals Problem

- If network can arbitrarily lose messages, then it is impossible to guarantee two (or more) nodes can reach agreement

Sleepy Generals Problem

- If **f** nodes can fail, you need **f+x** replicas to guarantee x correct responses from a **stateless** system (typically x=1)

Bureaucratic Generals Problem

Paxos, Raft

- If **f** nodes can fail, you need **2f+1** replicas to guarantee a correct response from a **stateful** system

Byzantine Generals Problem

PBFT, Zyzzyva, Blockchain

- If **f** nodes can be arbitrarily malicious, you need **3f+1** replicas to guarantee a correct response (stateful or stateless)

Paxos and Raft

Goal: Achieve state machine replication for crash fault tolerance (non-byzantine, stateful, reliable network)

Paxos: Lamport ’90, published ’98 (interesting history)

- Consensus algorithm presented in a paper pretending to describe how a fictitious ancient greek civilization wrote laws
- Used by Google Chubby, Apache Zookeeper, etc

Raft: Ongaro and Ousterhout ’14

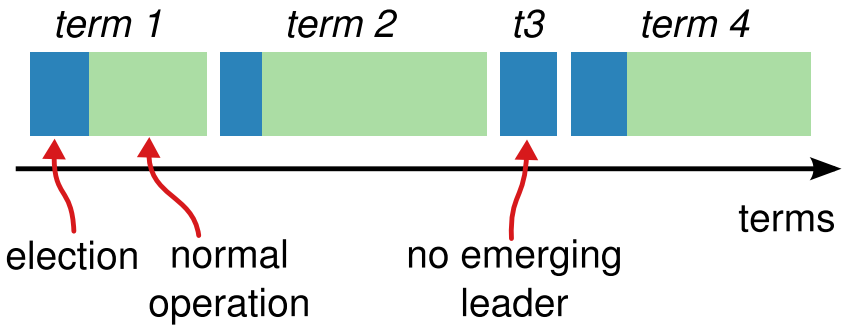
- An “Understandable Consensus Algorithm”. Described as a set of Remote Procedure Calls (RPCs) that need to be implemented, but still provides strong guarantees
- Dozens of implementations, used in many real products

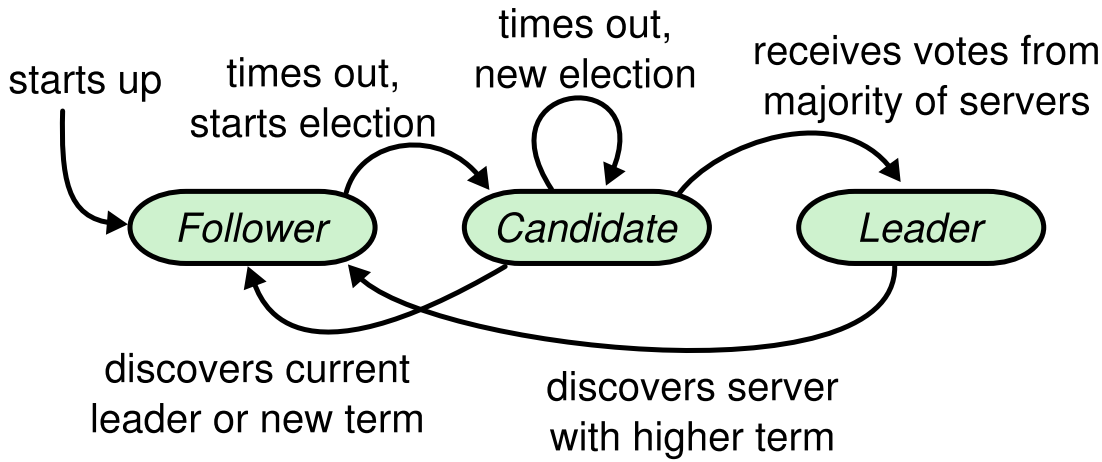
Both provide fault tolerance and safety, but are not guaranteed to terminate (no liveness)

Raft

Strong leader: All incoming requests pass through leader to be ordered

Leader election: Elections periodically occur in case the primary fails





Raft

Logs: Store a change to the system state, a term number and a log index. A log is committed once a majority of the nodes in the system have stored a copy of the entry.

Recovery and changes: Failed nodes or newly joining nodes can be brought up to date by synchronizing log