BLOCKCHAIN AT BERKELEY

TRUST WITHOUT TRUST

DISTRIBUTED SYSTEMS & CONSENSUS

SIMON GUO
AYUSH AGGARWAL

Table of Contents

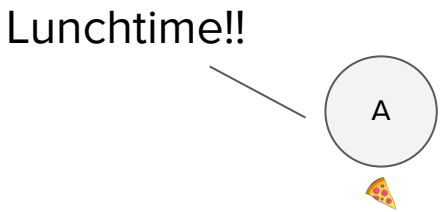
DISTRIBUTED SYSTEMS

Why Study Consensus?

Majority opinion; general agreement

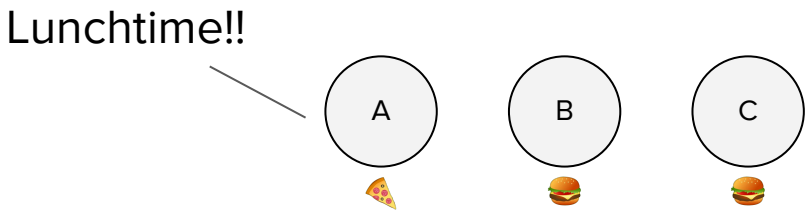
Why Study Consensus?

Majority opinion; general agreement



Why Study Consensus?

Majority opinion; general agreement



???

Origin: Digital Avionics

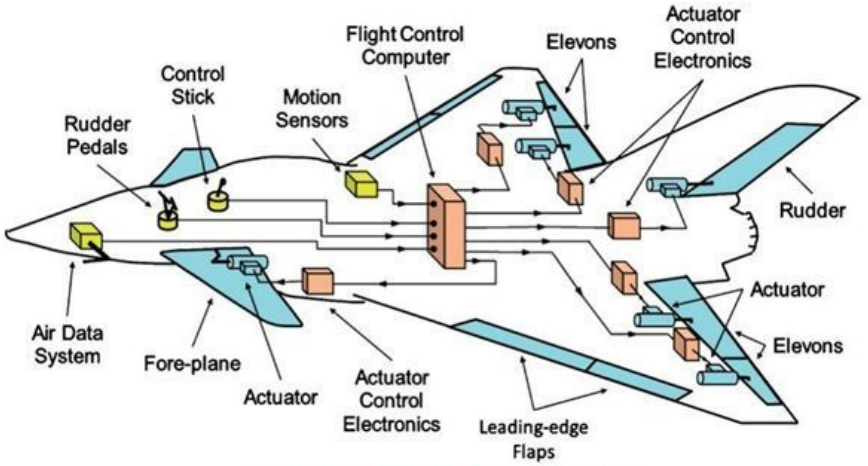
- Super dependable computers pioneered by aircraft manufacturers
- Aircraft $$$
- Passenger's safety

- Altitude, speed, fuel sensors
- Autopilot, fly-by-wire

The Solution

Many problems in distributed computing reduce to consensus

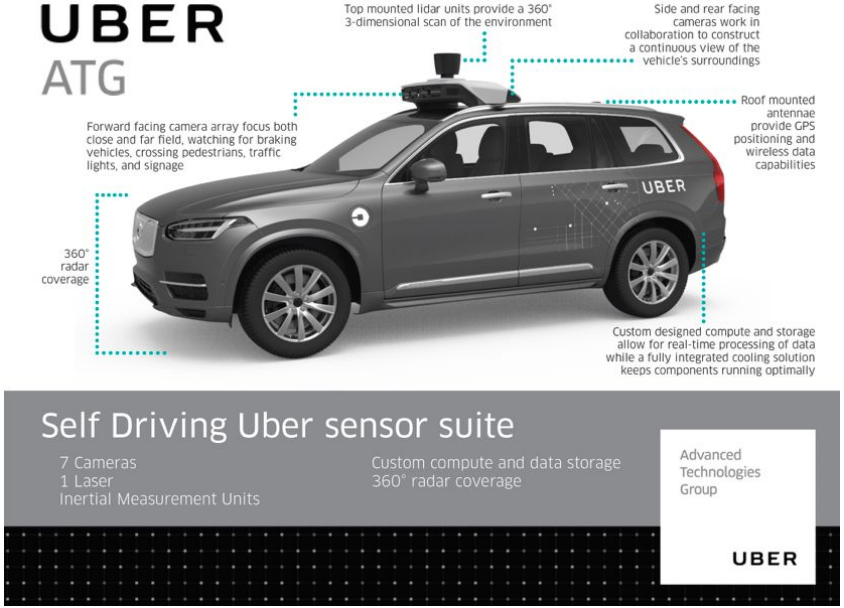
Fundamentally, can we create a reliable system from potentially unreliable parts?



Basic elements of the FWB control system.

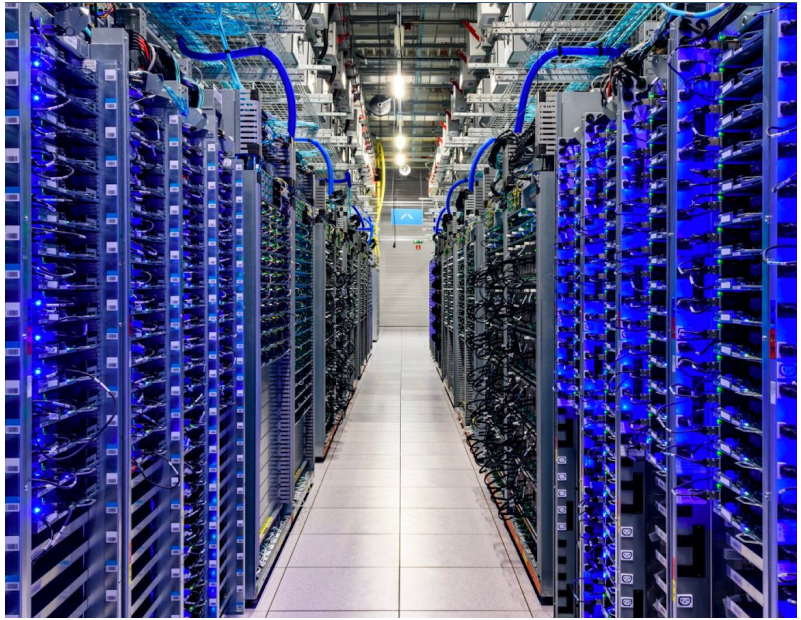
Systems in Adverse Environment

SpaceX Dragon



Self-Driving Cars

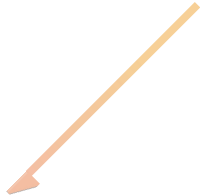
Database Systems



Data Centers

The Blockchain

Trust without ***trust***



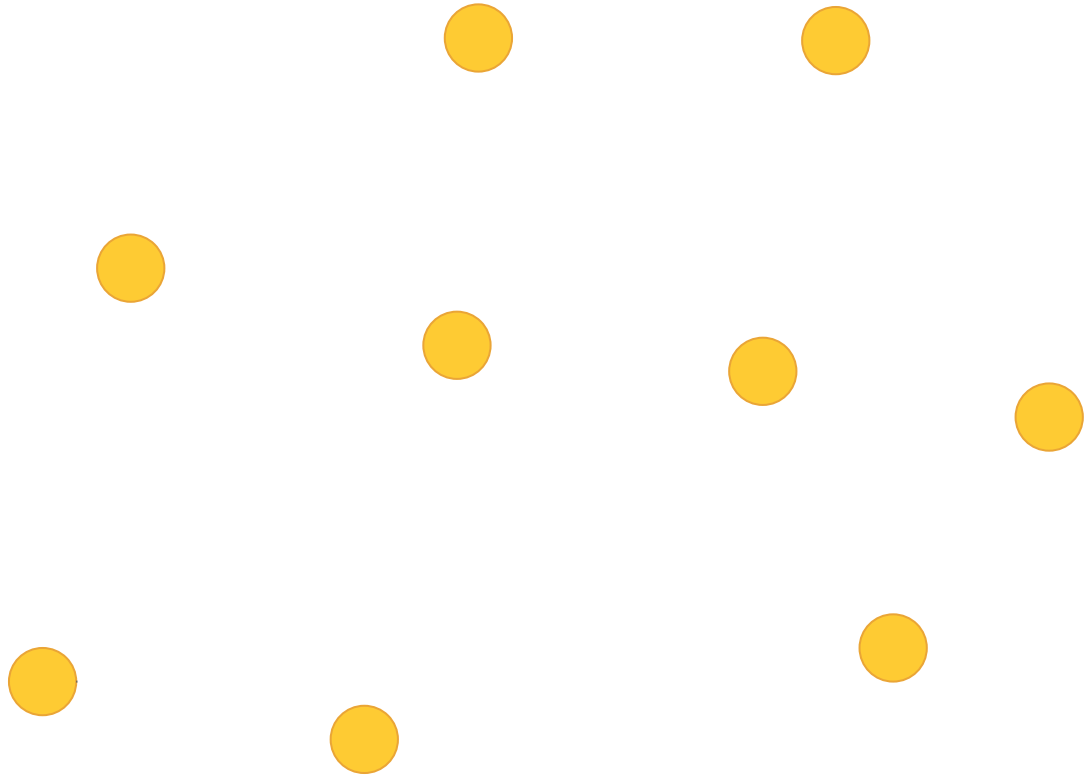
“correct” execution of the distributed system

reliability of individual components

Definition

Nodes

Each a “process”

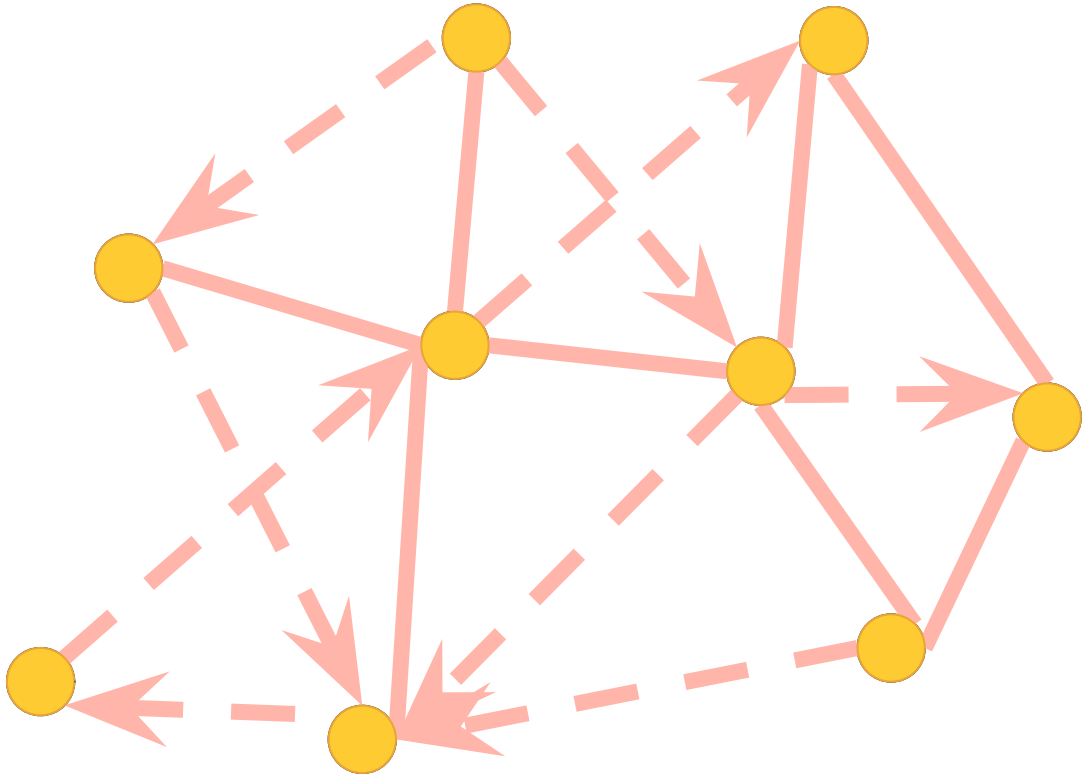


Definition

Nodes
Each a “process”

Message Channels
Move information

Purpose
Accomplish a common goal

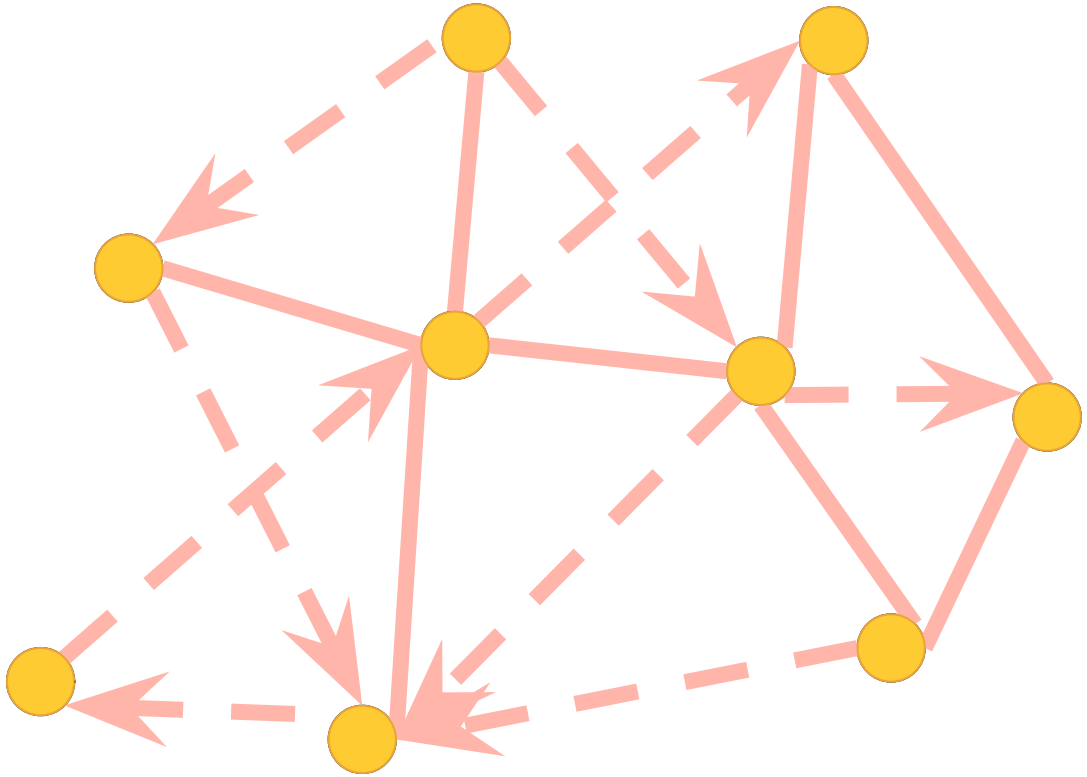


Challenges

Concurrent components

Potential failure of individual components

No global clock

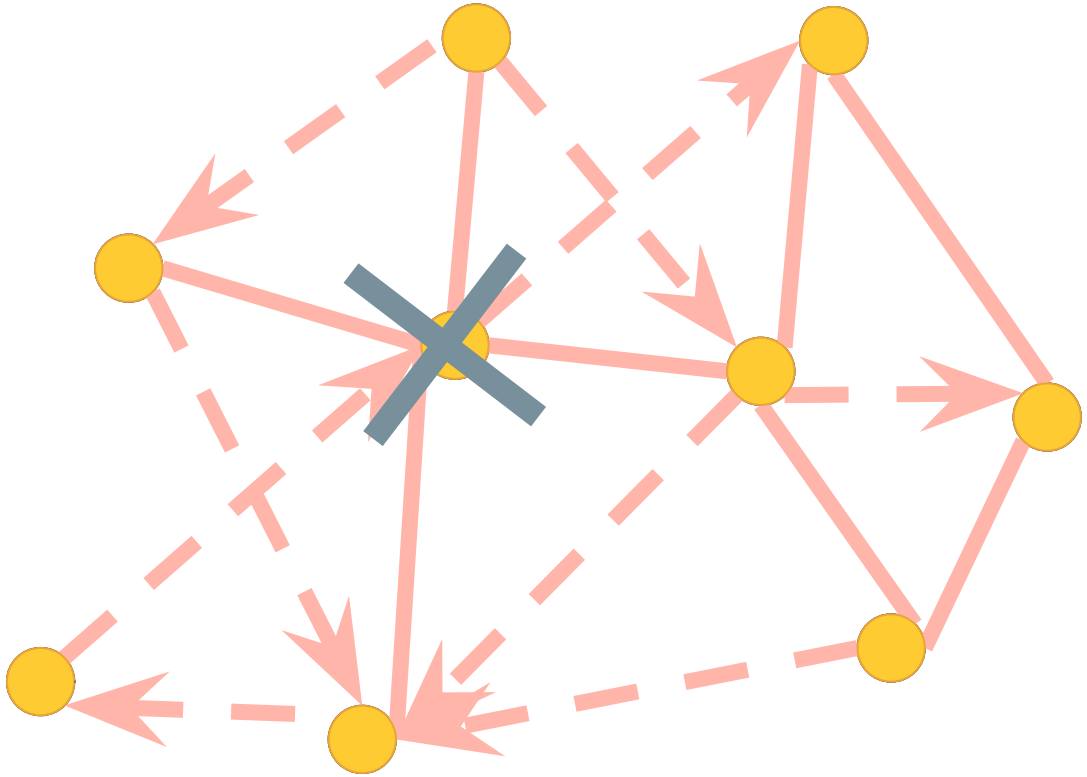


Challenges

Concurrent components

Potential failure of individual components

No global clock

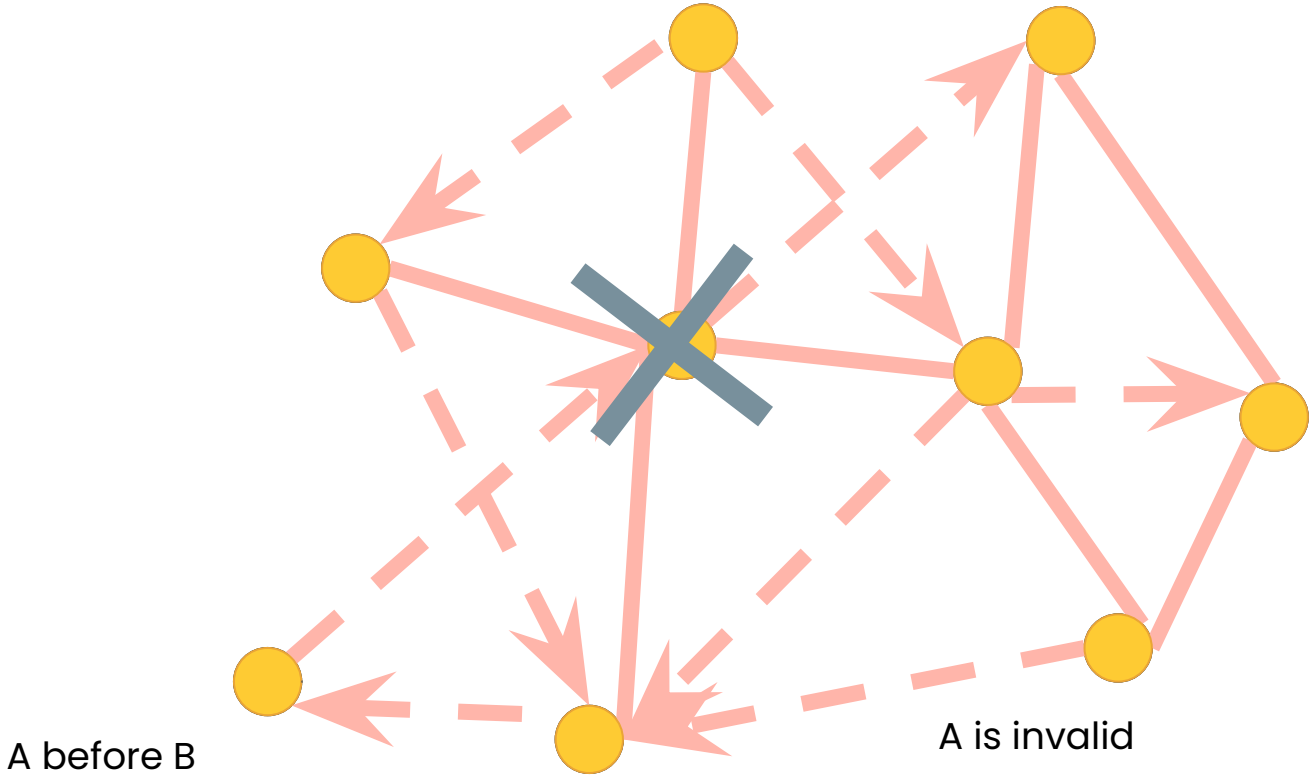


Challenges

Concurrent components

Potential failure of individual components

No global clock



The Big Picture

Do meaningful work while everything is ***on fire***

QUESTIONS?

PROPERTIES & CAP THEORY

Safety vs Liveness: A Trade-Off

SAFETY

This will *not* happen

LIVENESS

This *must* happen

Correctness of Consensus

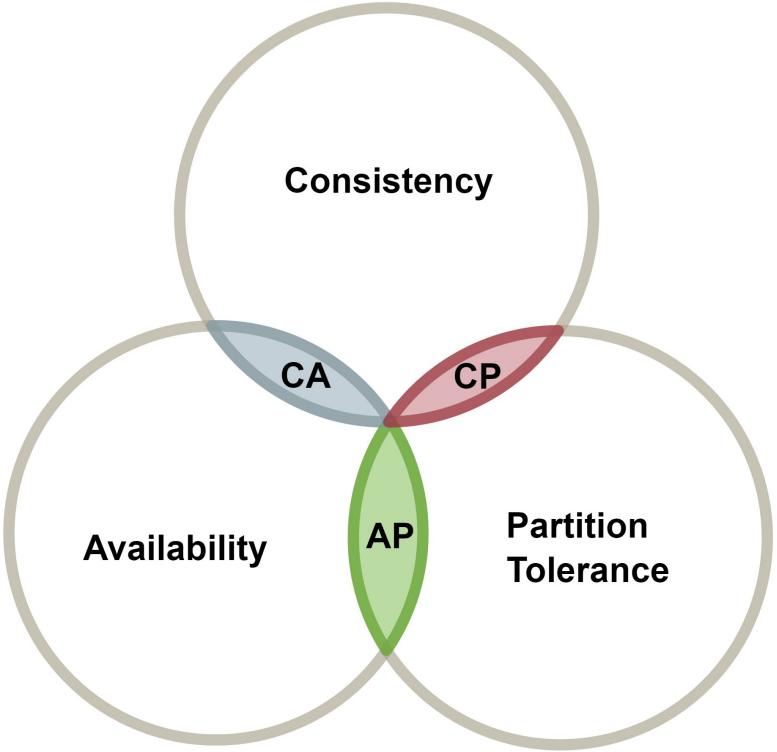
Validity	Agreement	Termination
Any value decided upon must be proposed by one of the processes	All non-faulty processes must agree on the same value	All non-faulty nodes eventually decide

Correctness of a distributed system: Achieving its intended goal

Introduction

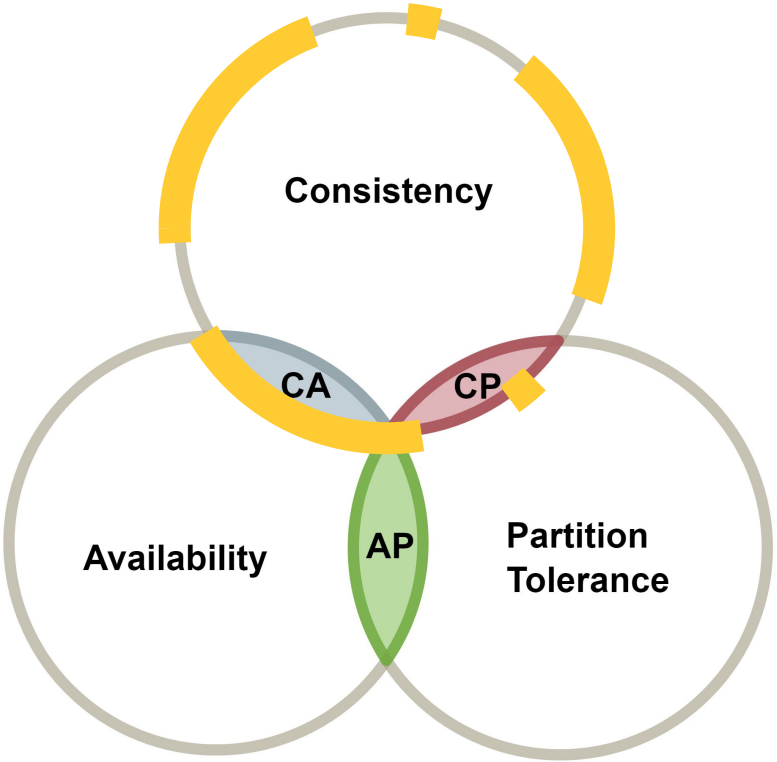
CAP Theorem

Fundamental theorem for any distributed system pertaining to achievable properties



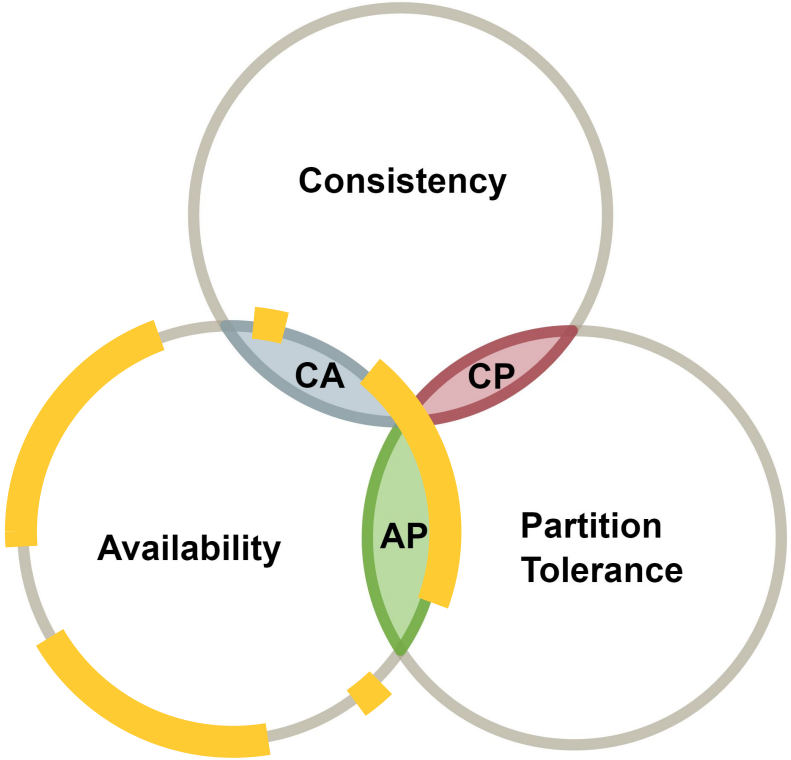
Consistency

Every node provides the most recent state, or does not provide a state at all



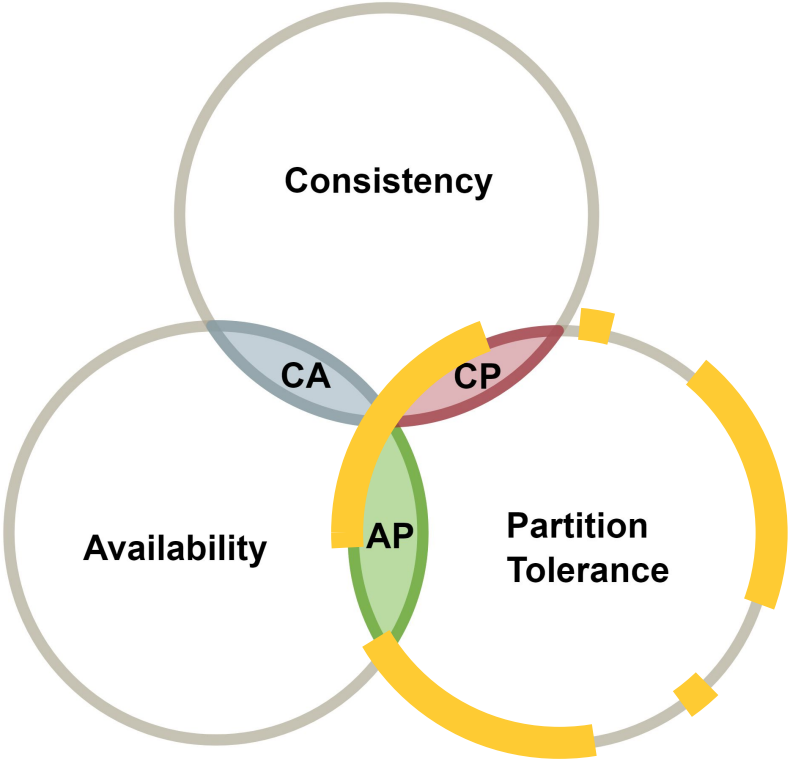
Availability

Every node has constant read and write access



Partition Tolerance:

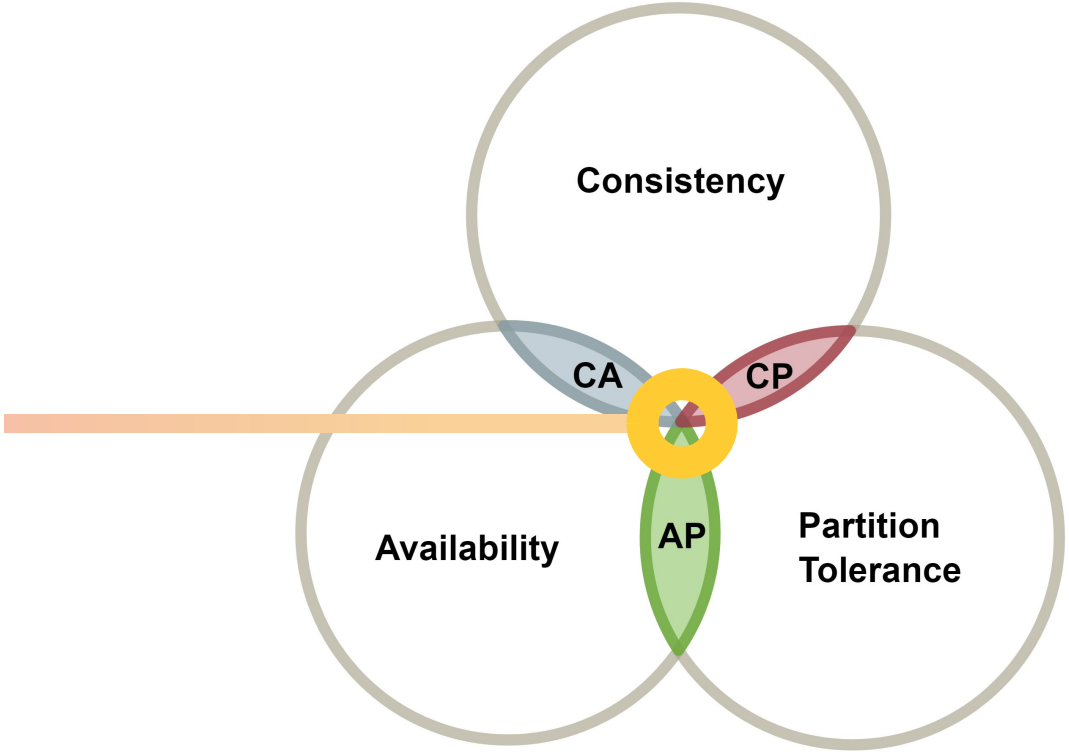
The system works despite partitions in the network



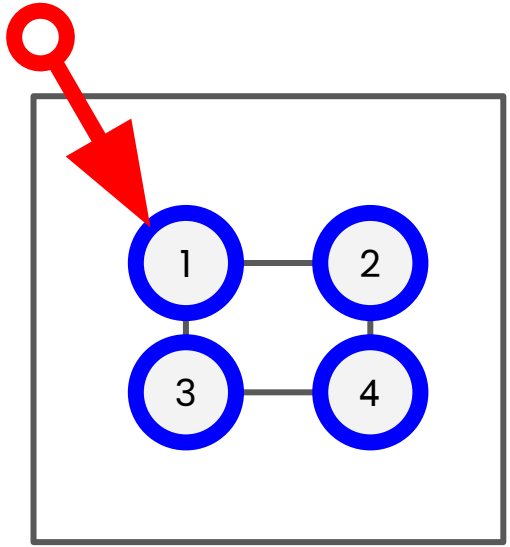
Trilemma

Can only have ***two*** of three

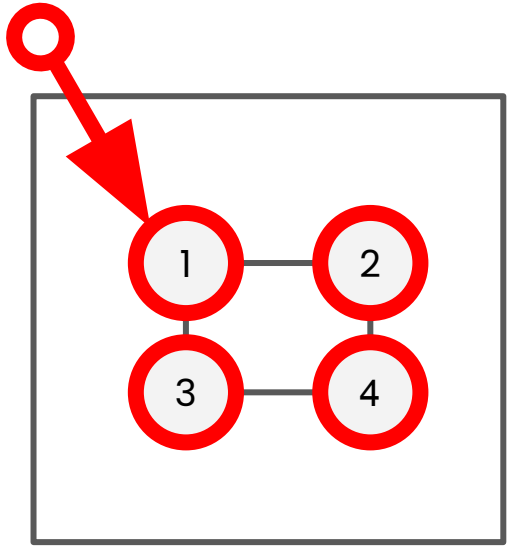
NOTHING IS IMPOSSIBLE...
EXCEPT THIS



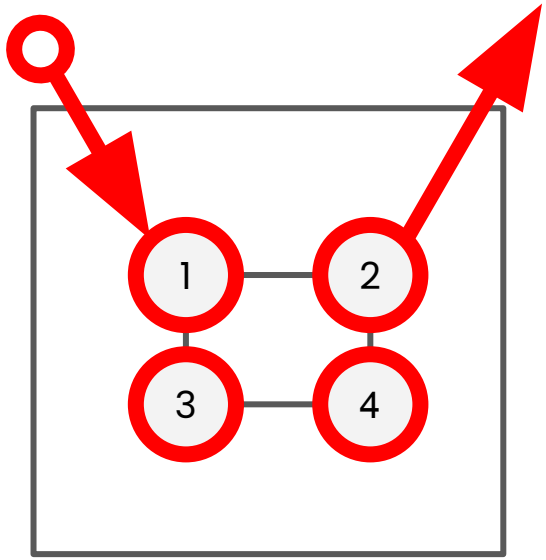
Why can’t we have them all?



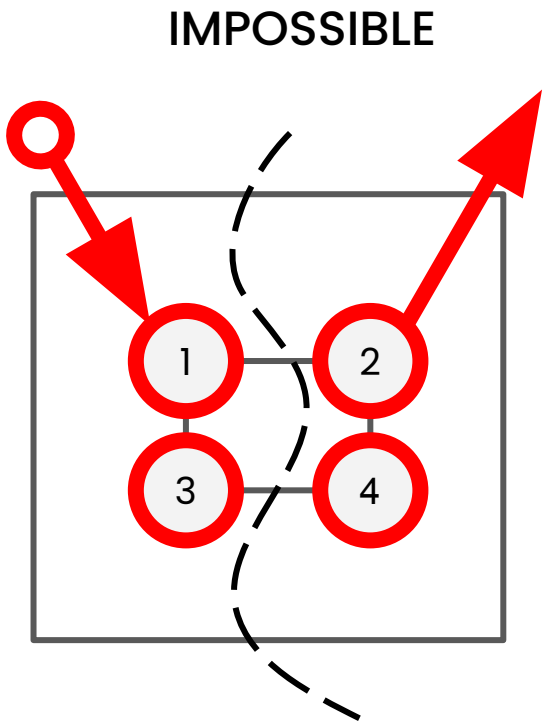
Why can’t we have them all?



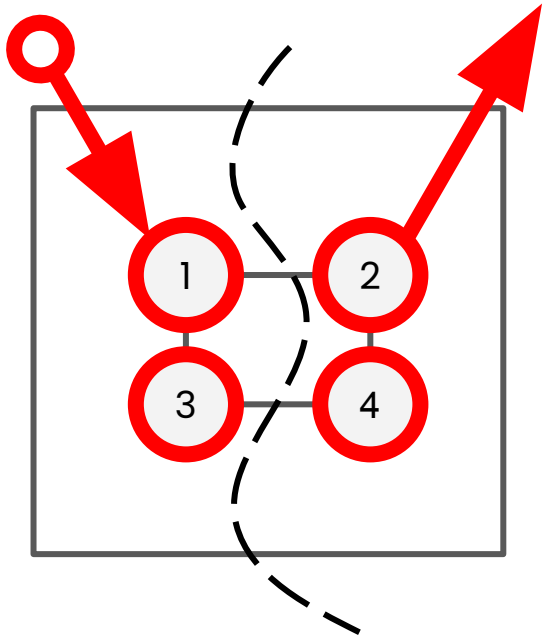
Why can’t we have them all?



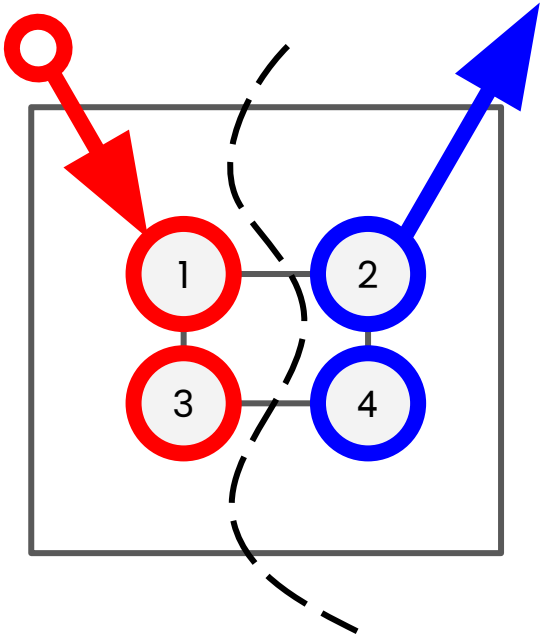
Why can’t we have them all?



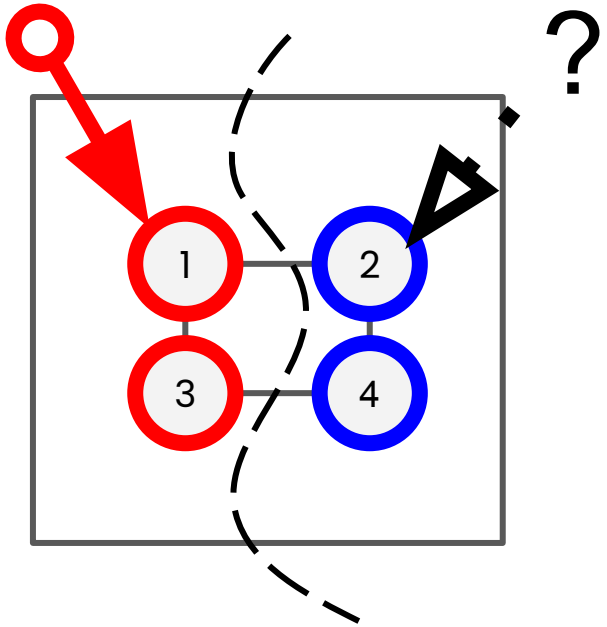
Consistent + Available
= Not Partition Tolerant



Partition Tolerant + Available = Not Consistent

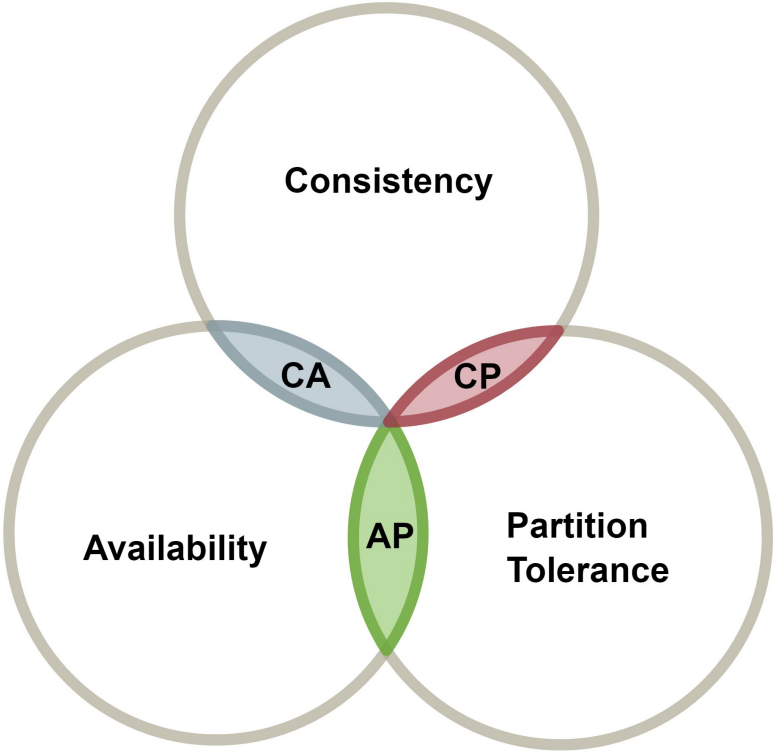


Partition Tolerant + Consistent = Not Available



Notes

- ***P***artition Tolerance is almost a given for any system
- Tradeoff is between ***C*** and ***A***
  - Not black-and-white tradeoffs, but on a spectrum
- CAP Theorem often taken too far, misleading if not well understood



QUESTIONS?

BYZANTINE FAULT TOLERANCE

The Byzantine Generals Problem

LESLIE LAMPORT, ROBERT SHOSTAK, and MARSHALL PEASE
SRI International

Reliable computer systems must handle malfunctioning components that give conflicting information to different parts of the system. This situation can be expressed abstractly in terms of a group of generals of the Byzantine army camped with their troops around an enemy city. Communicating only by messenger, the generals must agree upon a common battle plan. However, one or more of them

The Problem

Practical Byzantine Fault Tolerance, Miguel Castro and Barbara Liskov (1999)

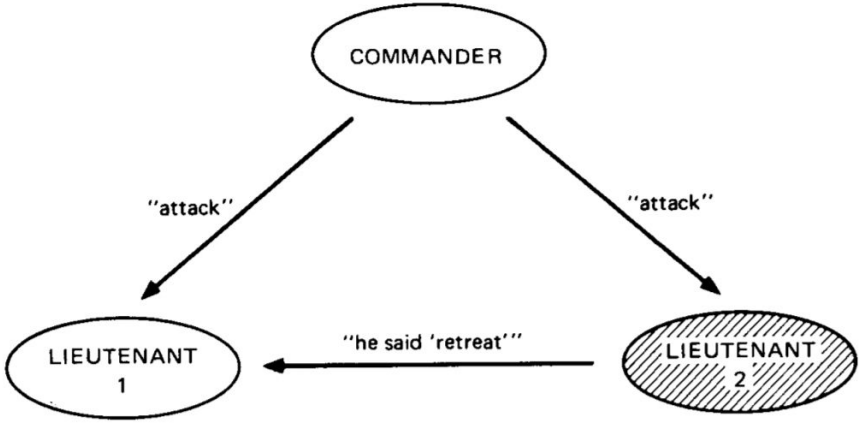


Fig. 1. Lieutenant 2 a traitor.

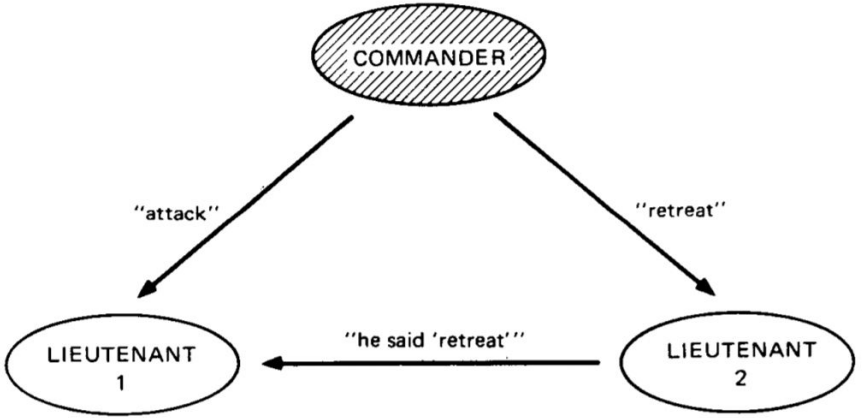


Fig. 2. The commander a traitor.

Possible Faults

Fail-stop

Nodes can crash, not return values, crash detectable by other nodes

Byzantine

Nodes can do all of the above and send incorrect/corrupted values, corruption or manipulation harder to detect

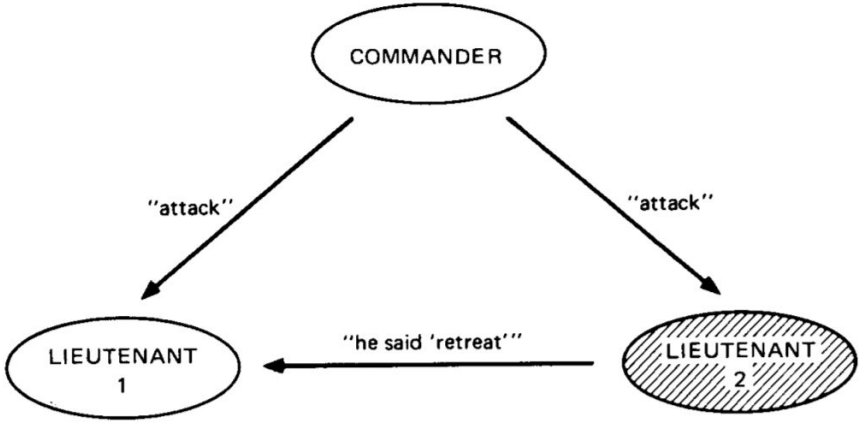


Fig. 1. Lieutenant 2 a traitor.

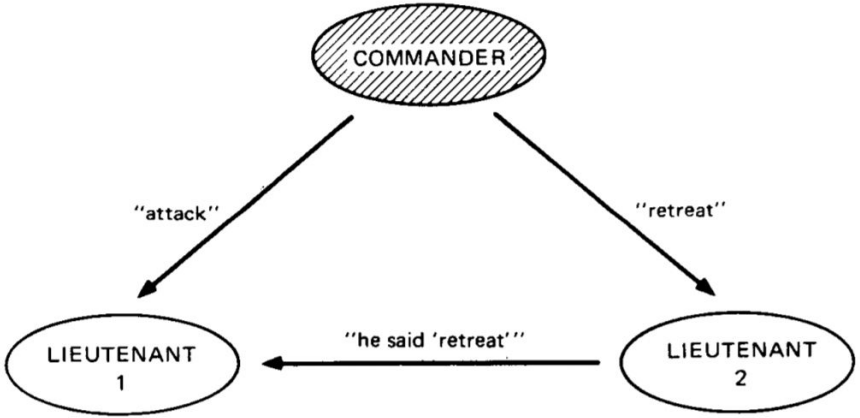


Fig. 2. The commander a traitor.

1/3 Bottleneck

In both cases, Lieutenant 1 gets conflicting messages and there is no way to figure out who is malicious.

We can generalize this:
No solution for >= 1/3 traitors

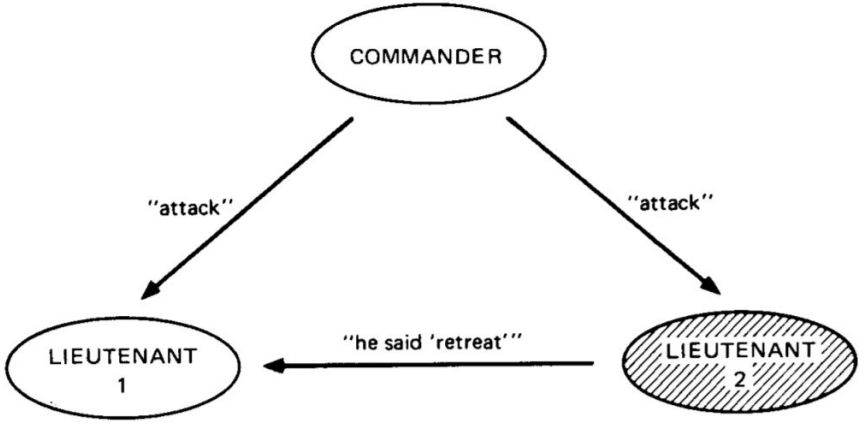


Fig. 1. Lieutenant 2 a traitor.

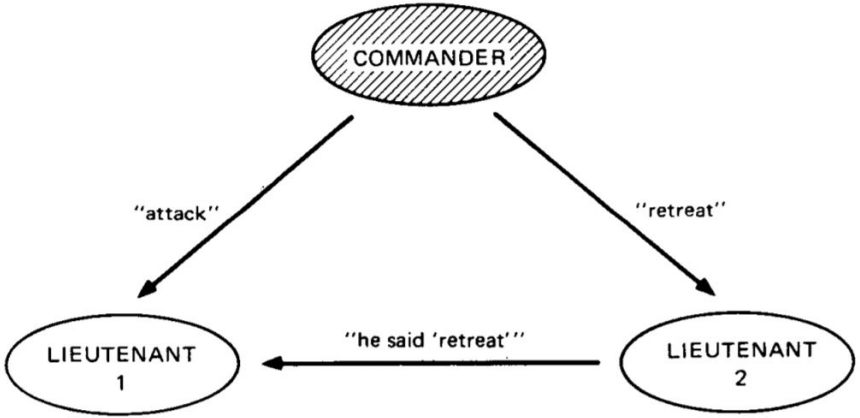


Fig. 2. The commander a traitor.

Connection to Blockchain

Byzantine Generals	The Blockchain
Generals	Nodes
Traitor generals	Faulty or malicious nodes
Geographic distance	Distributed network
Unreliable messengers	Dropped or corrupted messages; unreliable network
Attack or retreat	Consensus on history (e.g. transaction log)

QUESTIONS?

VOTING-BASED CONSENSUS


Paxos

PAXOS

OVERVIEW

Source:
http://www.hikingbikingadventures.com/biking-adventures/bicycling-europe-2/bicycling-greece/bicycling-paxos-island/

AUTHOR: RUSTIE LIN

PAXOS

OVERVIEW

Paxos is a Greek island

Source:
http://www.hikingbikingadventures.com/biking-adventures/bicycling-europe-2/bicycling-greece/bicycling-paxos-island/

AUTHOR: RUSTIE LIN

PAXOS

OVERVIEW

Paxos is a Greek island

but also . . .

Source:
http://www.hikingbikingadventures.com/biking-adventures/bicycling-europe-2/bicycling-greece/bicycling-paxos-island/

AUTHOR: RUSTIE LIN

PAXOS

OVERVIEW

Paxos is a Greek island

but also . . .

A CONSENSUS ALGORITHM

Source:
http://www.hikingbikingadventures.com/biking-adventures/bicycling-europe-2/bicycling-greece/bicycling-paxos-island/

PAXOS

TERMINOLOGY

Within the Paxon Parliament...

Proposer: legislator, advocates a citizen's request, moves protocol forward
Acceptor: legislator, voter
Learner: remembers and carries out result for citizen

Quorum:

- any majority of Acceptors
- any two Quorums must overlap

PAXOS

MAIN IDEA

Protocol proceeds over several rounds
where each successful round has 2 phases: prepare and accept

1. Citizens talk to Proposer
2. Within Paxon parliament: Proposer, Acceptor, and Learner discuss
   a. Pass decrees
   b. A decree has a number and value
3. Learner talks to citizens

1375: *Γωυδα is the new cheese inspector*

277: *The sale of brown goats is permitted*

37: *Painting on temple walls is forbidden*

PAXOS IN PRACTICE

ASSUMPTIONS AND REAL WORLD USE

- Only works for fail-stop (no Byzantine failures) faults
- Many variants of Paxos (e.g. Egalitarian Paxos, Byzantine Paxos, etc.)
- Good performance (fast)
- Generally used to replicate large sets of data

AUTHOR: BRIAN HO
EDITED: RUSTIE LIN

PAXOS

KNOWLEDGE OF THE ANCIENTS

There is an obvious correspondence between this database system and the Paxon Parliament:

Parliament		Distributed Database
legislator	↔	server
citizen	↔	client program
current law	↔	database state

Source: https://lamport.azurewebsites.net/pubs/lamport-paxos.pdf

RAFT

OVERVIEW

Raft is another consensus mechanism designed to be an alternative to Paxos

- Designed to be more understandable than Paxos
- Leader-based approach
- Easier to implement
- JP Morgan's Quorum: Raft-based consensus

RAFT

HOW IT WORKS

A Raft cluster has one and only one elected **leader**

- Communicates with client directly
- Responsible for managing log replication on the other servers of the cluster
- Leads until it fails or disconnects, in which case a new leader is elected

Leader Election

1. Leader sends “heartbeats” to other nodes saying that it is online
2. If other nodes no longer receive “heartbeat,” they start an election cycle (and internal timer)
3. First candidate to timeout becomes new leader

Log Replication

1. Leader accepts client request
2. Leader ensures all other nodes have followed that request

AUTHOR: BRIAN HO

QUESTIONS?

NAKAMOTO CONSENSUS

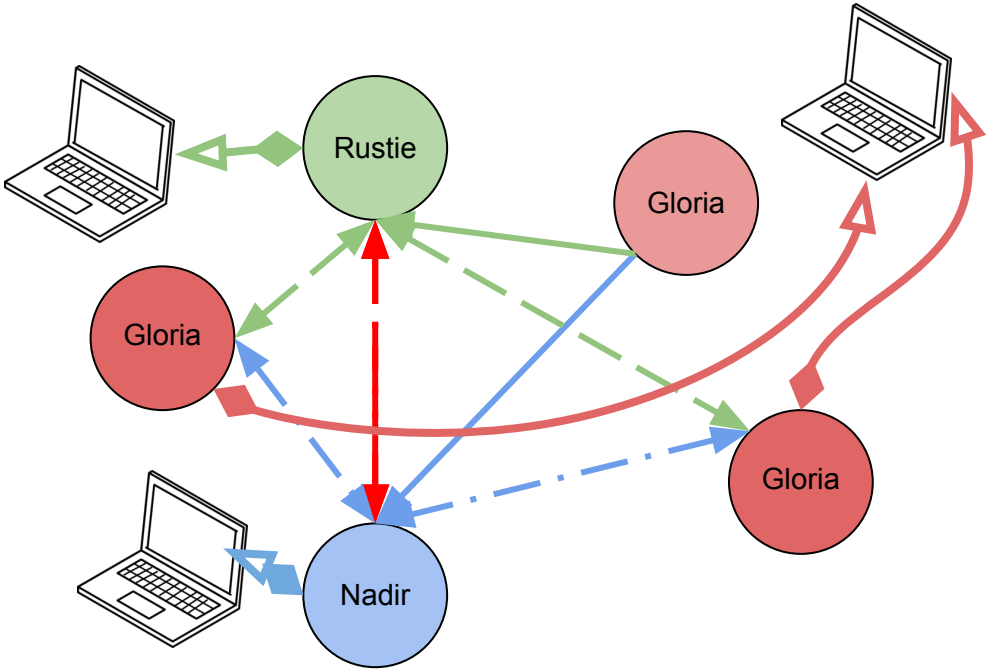
NAKAMOTO CONSENSUS

PROOF-OF-WORK

Quick review:

- Miners solve partial preimage hash puzzle
- Prevent Sybil attacks

Resource(s) consumed:
Computational power



NAKAMOTO CONSENSUS

GENERALIZATION

Quick review:

- Elect leader through some “lottery”
  - Relaxed notion of “membership”
- Leader creates next block
- Others **vote implicitly** by including block in their chain

Specifying the lottery type:

What resource is being consumed?

QUESTION

IDEAS

- What are some other resources that can be spent in consensus?

RESOURCE SPENDING

IDEAS

What are some other viable resources?

- Currency
- Time
- Space
- Reputation

We **spend resources** to partake in the network's progression.

It's **economically infeasible** to outspend all honest nodes

NAKAMOTO CONSENSUS

PROOF-OF-STAKE

Overview:

- Validators instead of “miners”
- Locking up “stake”

Resource(s) consumed:
Native currency

Source: https://btcmanager.com/buterin-confirms-ethereums-proof-of-stake-75-percent-complete/

AUTHOR: NADIR AKHTAR

FLAVORS OF PROOF-OF-STAKE

CHAIN AND BFT BASED

1. Randomly choose a validator based on the proportional stake invested

2. The chosen validator ***creates*** a block
3. Protocol continues forward with no explicit notion of votes

2. The chosen validator ***proposes*** a block
3. Protocol to ensure ⅔+ votes or start over

4. The chosen validator gets the block reward and the transaction fees

AUTHOR: RUSTIE LIN

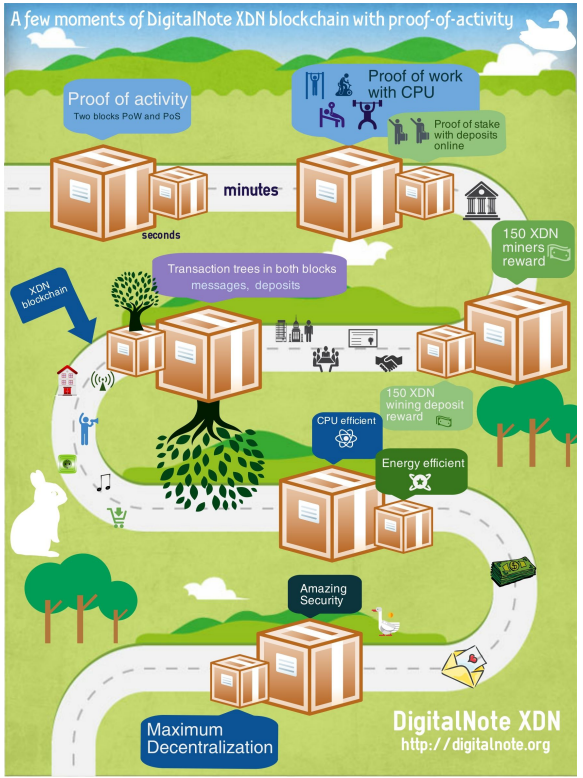
NAKAMOTO CONSENSUS

PROOF-OF-ACTIVITY

PoW + PoS hybrid:

- PoW: Miners create blocks (with or without transactions, depending on implementation); block header contains validator data
- PoS: Validators sign valid blocks

Resource(s) consumed:
PoW *and* PoS resources



Source:
http://digitalnotetalk.org/sites/digitalnotetalk.org/files/media/1459869843.jpg

AUTHOR: NADIR AKHTAR
EDITED BY: RUSTIE LIN

NAKAMOTO CONSENSUS

PROOF-OF-BURN

Overview:

- Send coins to irretrievable address
  - More coins burned, higher likelihood of election
- Like Proof-of-Stake, but ***edgier***
- Bootstrapping mechanism

Resource(s) consumed:

Currency (potentially not native)

Source: https://www.reddit.com/r/dogecoin/comments/2bhiqh/with_all_this_talk_about_proof_of_burn_i_thought/

AUTHOR: NADIR AKHTAR

NAKAMOTO CONSENSUS

PROOF-OF-SPACE

Overview:

- Use disk space to solve challenge
- Can also use for file storage

Source:
https://hackernoon.com/crypto-review-siacoin-sc-b1d0f0a5c78f

Source:
https://medium.com/@tokenlot/filecoin-ico-open-to-accredited-investors-only-b7937f24de44

Source:
https://bitcoinist.com/storj-new-decentralized-storage-solution/

Resource(s) consumed:
Storage space

AUTHOR: NADIR AKHTAR

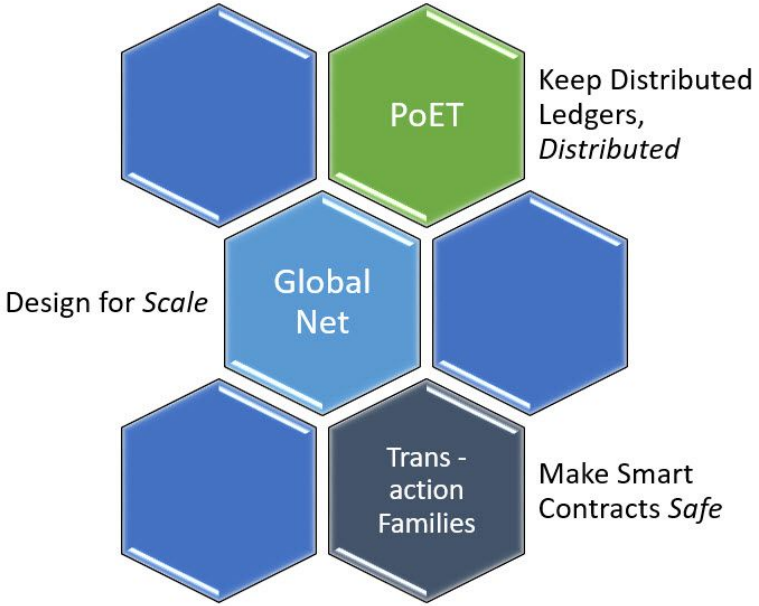
NAKAMOTO CONSENSUS

PROOF-OF-ELAPSED-TIME

Overview:

- Spend time instead of mining power
- Ask your Trusted Execution Environment (TEE) to wait a random amount of time, and attestation that it has indeed waited that much time
- First one who finishes waiting wins!

Resource(s) consumed:
Time



Source: https://www.altoros.com/blog/wp-content/uploads/2017/02/Hyperledger-Intel-Sawtooth-Lake-difference.jpg

PROOF-OF-AUTHORITY

NAKAMOTO, IS THAT YOU?

Overview:

- Permissioned, non-production
- Used in Kovan and Rinkeby Ethereum testnets

Resource(s) consumed(?):
Identity (reputation)

```
// Clique is the proof-of-authority consensus engine proposed to support the
// Ethereum testnet following the Ropsten attacks.
type Clique struct {
	config *params.CliqueConfig // Consensus engine configuration parameters
	db     ethdb.Database       // Database to store and retrieve snapshot checkpoints

	recents    *lru.ARCCache // Snapshots for recent block to speed up reorgs
	signatures *lru.ARCCache // Signatures of recent blocks to speed up mining

	proposals map[common.Address]bool // Current list of proposals we are pushing

	signer common.Address // Ethereum address of the signing key
	signFn SignerFn       // Signer function to authorize hashes with
	lock   sync.RWMutex   // Protects the signer fields
}
```

```
// Authorize injects a private key into the consensus engine to mint new blocks
// with.
func (c *Clique) Authorize(signer common.Address, signFn SignerFn) {
	c.lock.Lock()
	defer c.lock.Unlock()

	c.signer = signer
	c.signFn = signFn
}
```

Source: https://github.com/ethereum/go-ethereum

QUESTIONS?

FEDERATED CONSENSUS

BYZANTINE AGREEMENT

OVERVIEW

In a **distributed system**, a quorum is a set of nodes sufficient to reach agreement.

BYZANTINE AGREEMENT

OVERVIEW

What if you don’t necessarily trust certain nodes in the quorum? How can we still achieve consensus?

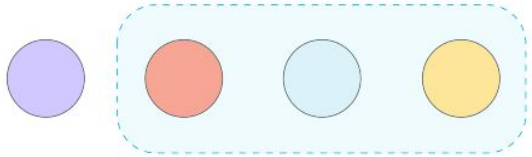
FEDERATED BYZANTINE AGREEMENT

OVERVIEW

Problem: How do we choose **quorums** in a decentralized way?

Solution: introduce **quorum slices**

- Subset of a **quorum** that can convince one particular node of agreement
- Individual nodes decide on other participants they trust for information



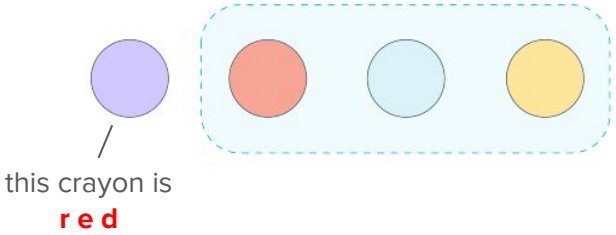
FEDERATED BYZANTINE AGREEMENT

OVERVIEW

Problem: How do we choose **quorums** in a decentralized way?

Solution: introduce **quorum slices**

- Subset of a **quorum** that can convince one particular node of agreement
- Individual nodes decide on other participants they trust for information



AUTHOR: BRIAN HO

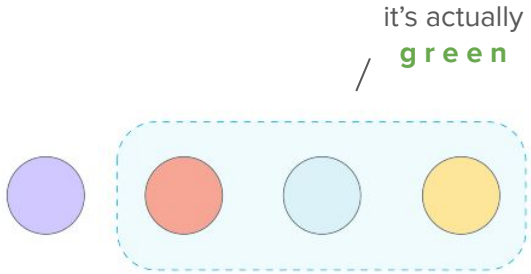
FEDERATED BYZANTINE AGREEMENT

OVERVIEW

Problem: How do we choose **quorums** in a decentralized way?

Solution: introduce **quorum slices**

- Subset of a **quorum** that can convince one particular node of agreement
- Individual nodes decide on other participants they trust for information



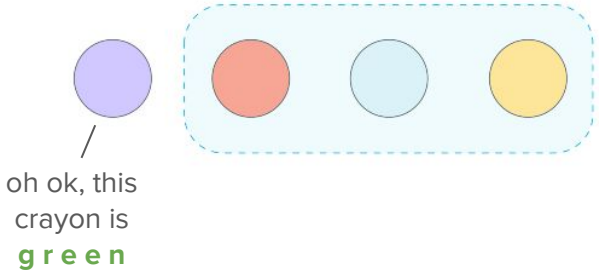
FEDERATED BYZANTINE AGREEMENT

OVERVIEW

Problem: How do we choose **quorums** in a decentralized way?

Solution: introduce **quorum slices**

- Subset of a **quorum** that can convince one particular node of agreement
- Individual nodes decide on other participants they trust for information



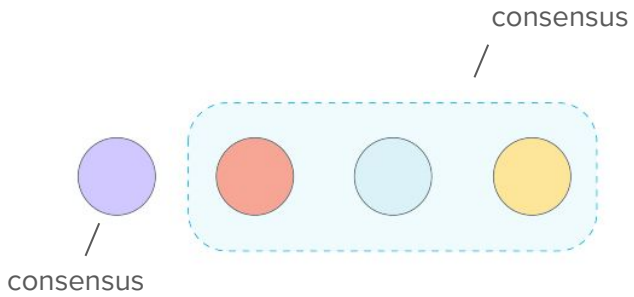
FEDERATED BYZANTINE AGREEMENT

OVERVIEW

Problem: How do we choose **quorums** in a decentralized way?

Solution: introduce **quorum slices**

- Subset of a **quorum** that can convince one particular node of agreement
- Individual nodes decide on other participants they trust for information



AUTHOR: BRIAN HO

FEDERATED BYZANTINE AGREEMENT

OVERVIEW

Idea: What happens when multiple quorum slices join together?

We get a **quorum intersection**!

- Quorum slices that come together will slowly convince other quorums slices and form a “larger” quorum

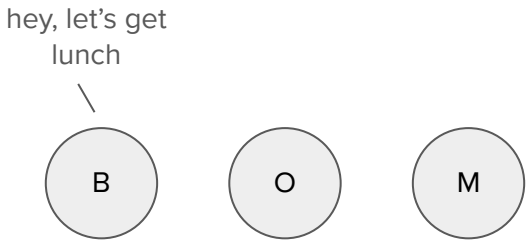
Otherwise we get **disjoint quorums** that agree on different things

Image source: https://cdn-images-1.medium.com/max/1600/0*msL7MVVEy4p2VzhP

FEDERATED BYZANTINE AGREEMENT

EXAMPLE

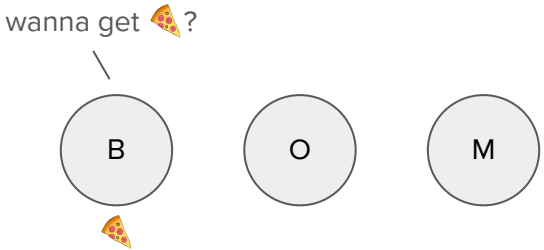
Lunchtime Consensus!



FEDERATED BYZANTINE AGREEMENT

EXAMPLE

Lunchtime Consensus!

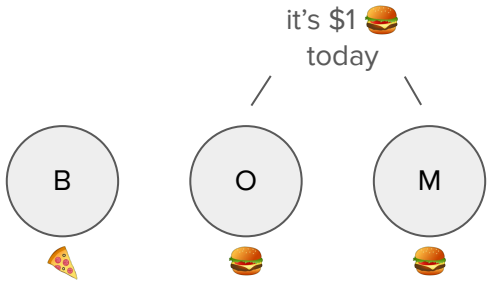


AUTHOR: BRIAN HO

FEDERATED BYZANTINE AGREEMENT

EXAMPLE

Lunchtime Consensus!

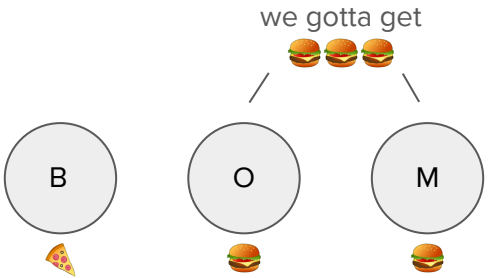


AUTHOR: BRIAN HO

FEDERATED BYZANTINE AGREEMENT

EXAMPLE

Lunchtime Consensus!

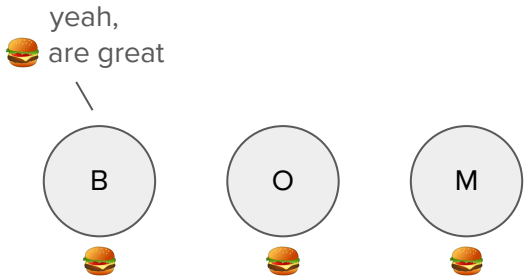


AUTHOR: BRIAN HO

FEDERATED BYZANTINE AGREEMENT

EXAMPLE

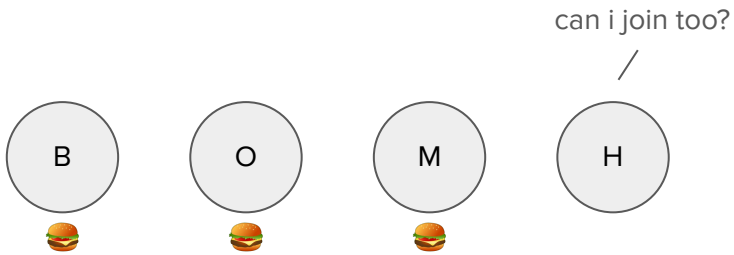
Lunchtime Consensus!



FEDERATED BYZANTINE AGREEMENT

EXAMPLE

Lunchtime Consensus!

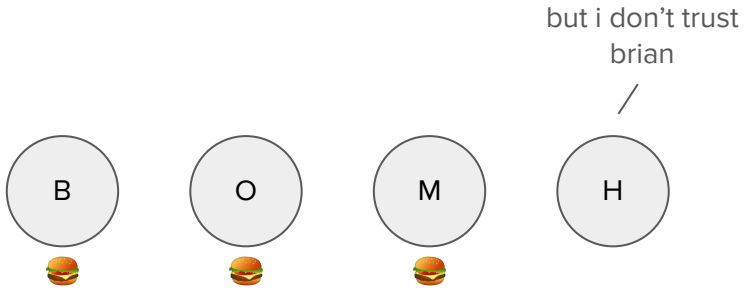


AUTHOR: BRIAN HO

FEDERATED BYZANTINE AGREEMENT

EXAMPLE

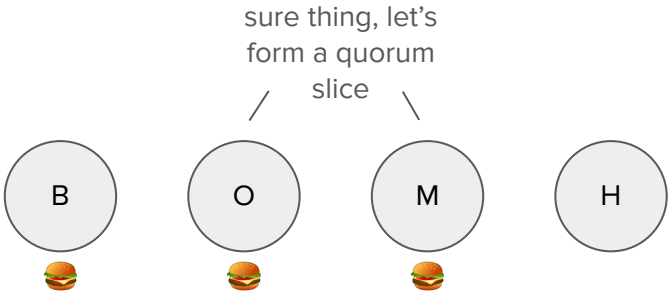
Lunchtime Consensus!



FEDERATED BYZANTINE AGREEMENT

EXAMPLE

Lunchtime Consensus!

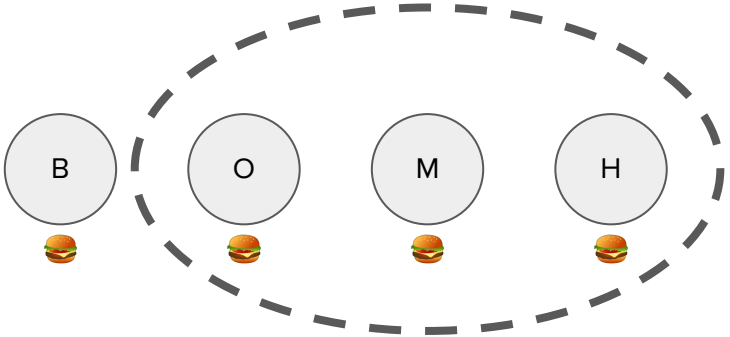


AUTHOR: BRIAN HO

FEDERATED BYZANTINE AGREEMENT

EXAMPLE

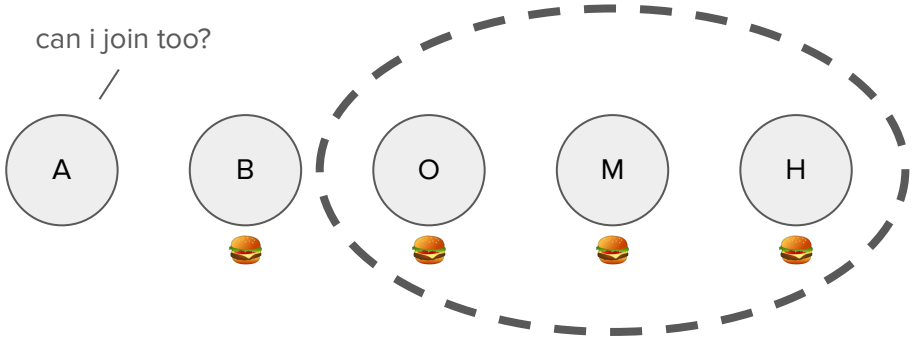
Lunchtime Consensus!



FEDERATED BYZANTINE AGREEMENT

EXAMPLE

Lunchtime Consensus!

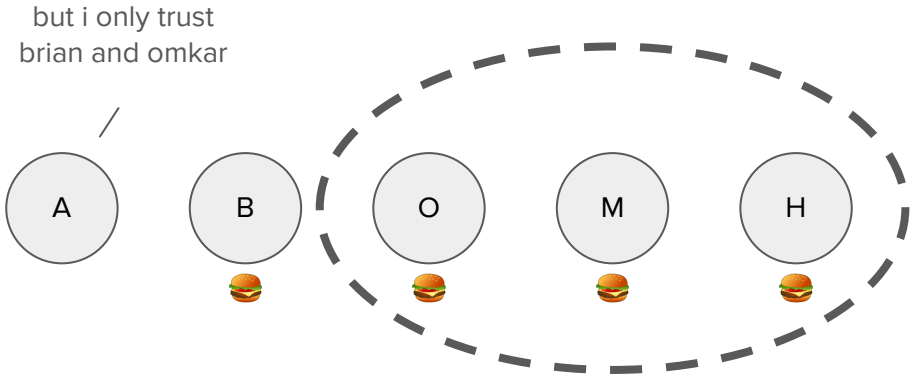


AUTHOR: BRIAN HO

FEDERATED BYZANTINE AGREEMENT

EXAMPLE

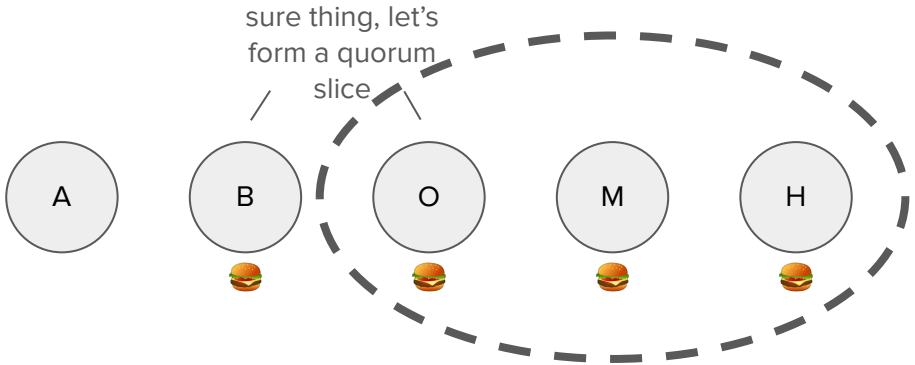
Lunchtime Consensus!



FEDERATED BYZANTINE AGREEMENT

EXAMPLE

Lunchtime Consensus!

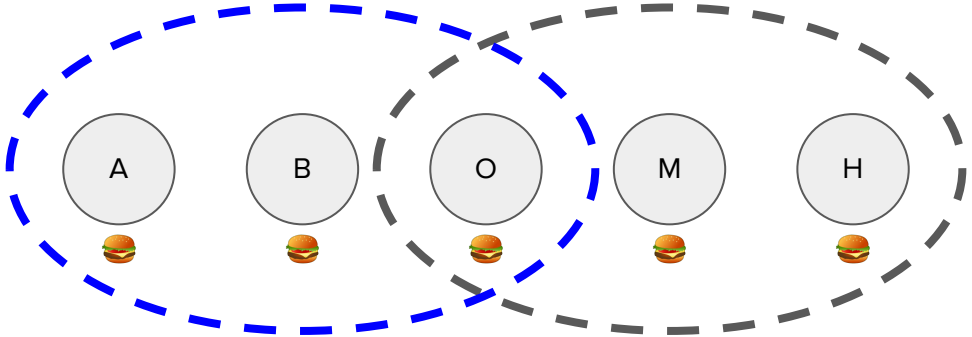


AUTHOR: BRIAN HO

FEDERATED BYZANTINE AGREEMENT

EXAMPLE

Lunchtime Consensus!

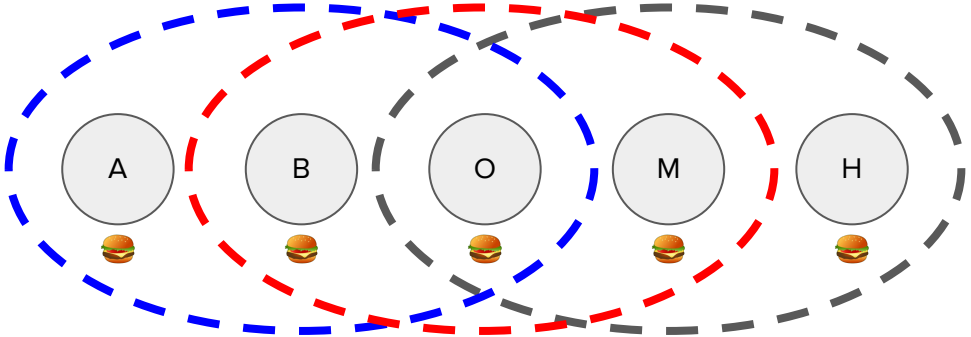


AUTHOR: BRIAN HO

FEDERATED BYZANTINE AGREEMENT

EXAMPLE

Lunchtime Consensus!



AUTHOR: BRIAN HO

FEDERATED BYZANTINE AGREEMENT

OVERVIEW

Decentralized control: no central authority that authorizes consensus

Low latency: consensus achieved in a few seconds

Flexible trust: nodes choose who they trust, don't have to trust the entire network

QUESTIONS?

Summary

01 DISTRIBUTED SYSTEMS

02 PROPERTIES & CAP THEORY

03 BYZANTINE FAULT TOLERANCE

04 VOTING-BASED CONSENSUS

05 NAKAMOTO & RESOURCE-BASED CONSENSUS

06 FEDERATED CONSENSUS

THANK YOU