ALTERNATIVE CONSENSUS MECHANISMS

Nadir Akhtar
Brian Ho

LECTURE OVERVIEW

1. DISTRIBUTED SYSTEMS
2. PROOF-OF-SOMETHING-ELSE
3. PROOF-OF-STAKE
4. VOTING-BASED CONSENSUS ALGORITHMS
5. FEDERATED CONSENSUS

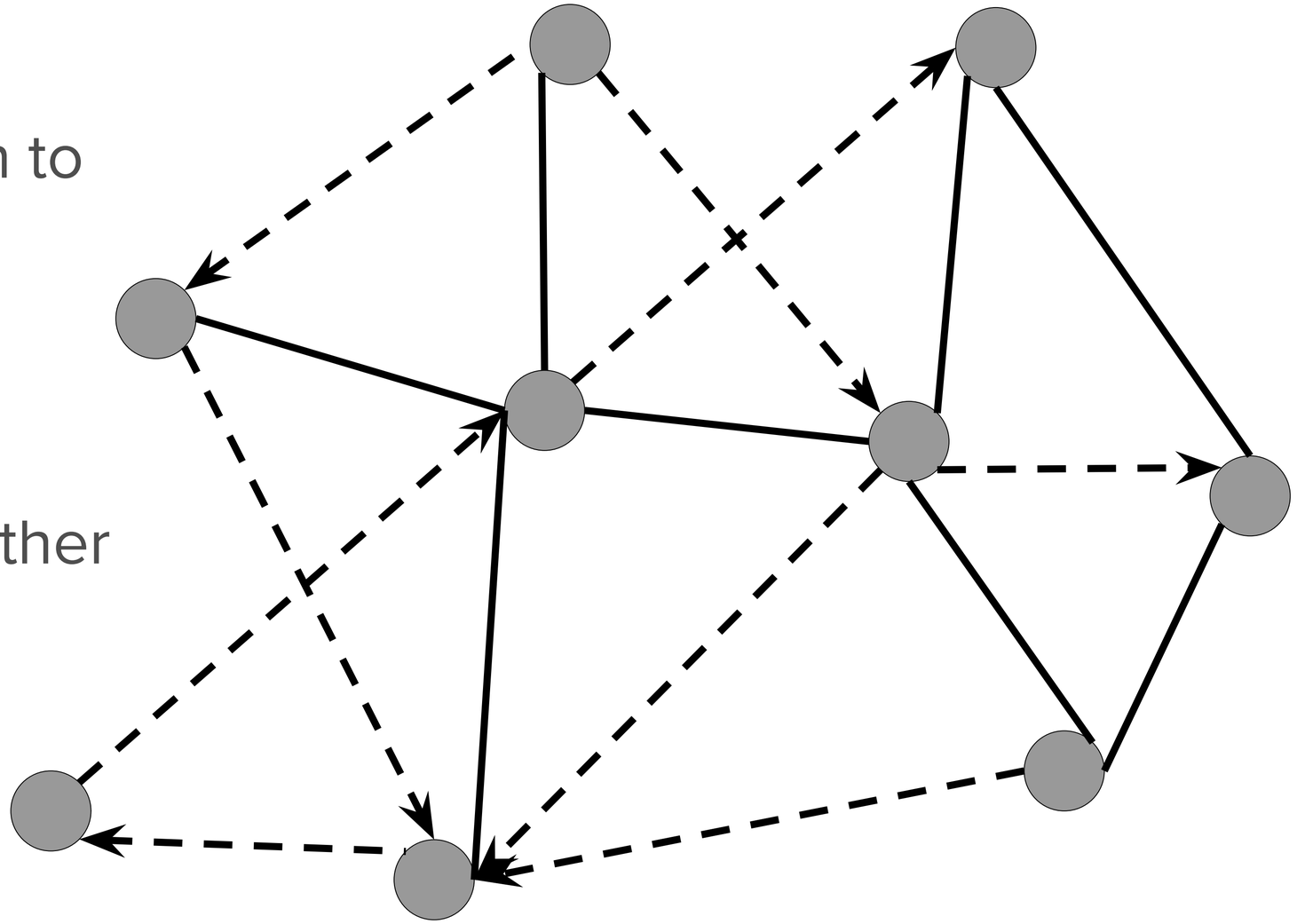
1 DISTRIBUTED SYSTEMS

1.1 DIST. SYSTEMS FUNDAMENTALS

DISTRIBUTED SYSTEMS

INTRODUCTION

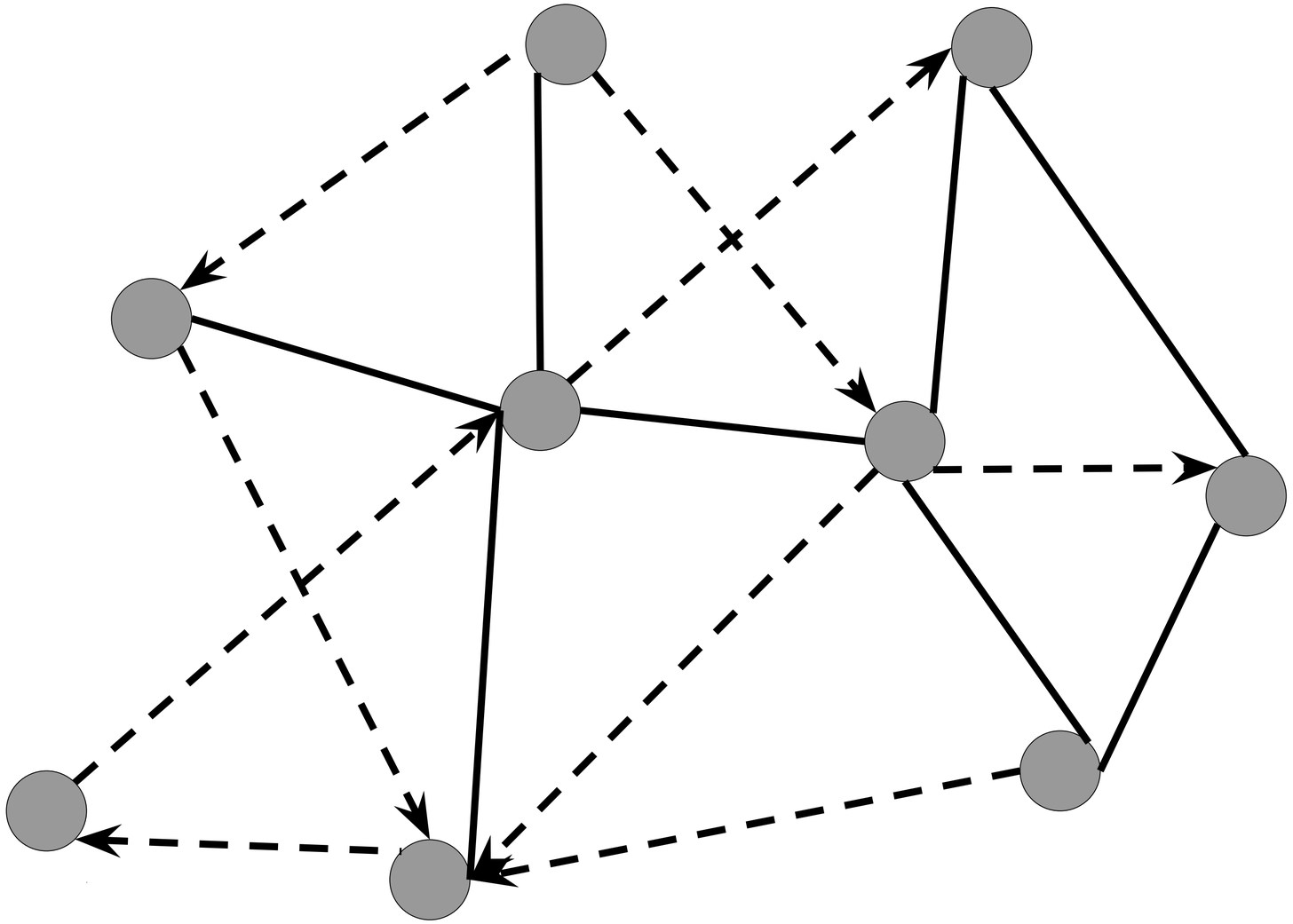
- Definition (varies from person to person):
  - A network of independent nodes, each representing a “process,” talking to each other via messages
- Purpose:
  - Accomplish a common goal



DISTRIBUTED SYSTEMS

INTRODUCTION

- Properties of distributed systems:
  - Concurrent components
  - Message sharing
  - No global clock
  - Potential failure of individual components



QUESTIONS?

1.2 DIST. SYSTEMS PROPERTIES

DISTRIBUTED SYSTEMS

PROPERTIES OF MULTIPROCESS PROGRAMS

Category of Property	Safety	Liveness
Definition	This will *not* happen	This *must* happen

Source:
https://www.microsoft.com/en-us/research/uploads/prod/2016/12/Proving-the-Correctness-of-Multiprocess-Programs.pdf

DISTRIBUTED CONSENSUS

CORRECTNESS

Correctness of a distributed system: achieving its intended goal

To ensure correctness, one uses a **consensus algorithm** achieving the following:

- **Validity:** any value decided upon must be proposed by one of the processes
- **Agreement:** all non-faulty processes must agree on the same value
  - Agreement and Validity are *safety* properties: Honest nodes will never decide on trivial, random, or different values
- **Termination:** all non-faulty nodes eventually decide.
  - Termination is a *liveness* property: All nodes eventually decide on a value

QUESTIONS?

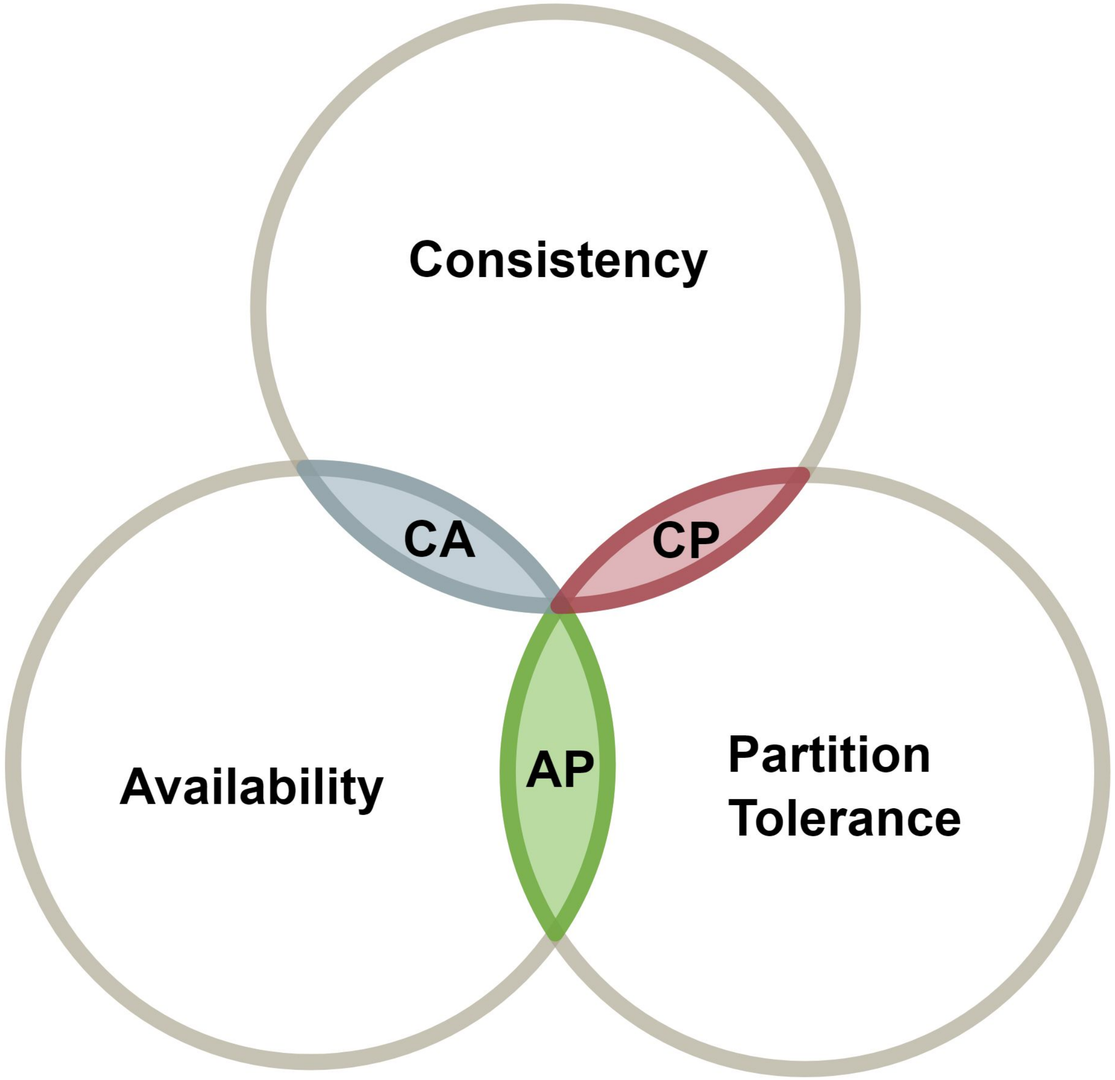
1.3 CAP THEOREM

CAP THEOREM

INTRODUCTION

Image source:
http://berb.github.io/diploma-thesis/original/resources/cap.svg

CAP Theorem:
Fundamental theorem for any distributed system pertaining to achievable properties



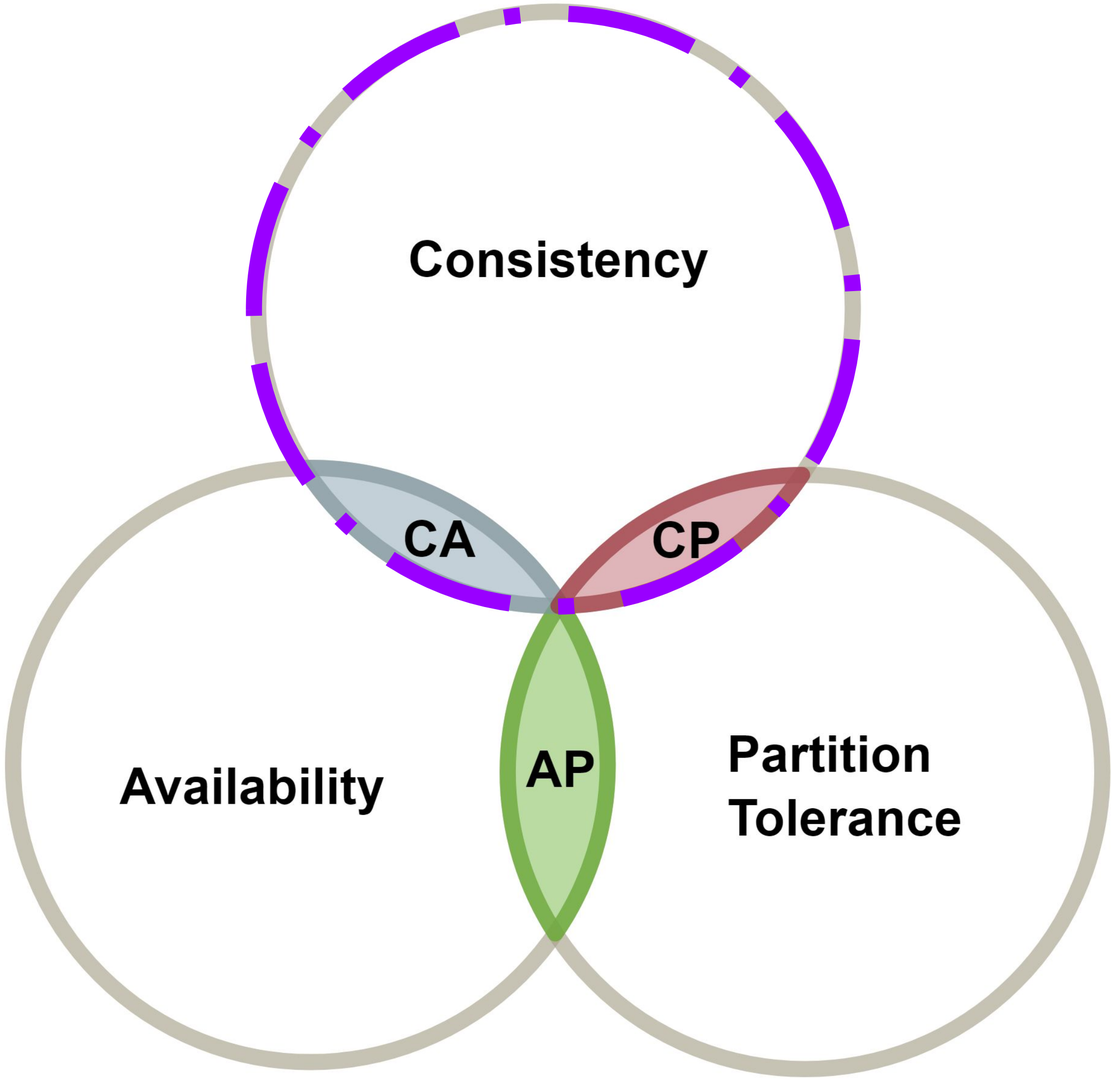
Source:
https://www.youtube.com/watch?v=Jw1iFr4v58M

CAP THEOREM

CONSISTENCY

Image source:
http://berb.github.io/diploma-thesis/original/resources/cap.svg

Consistency:
Every node provides the most recent state



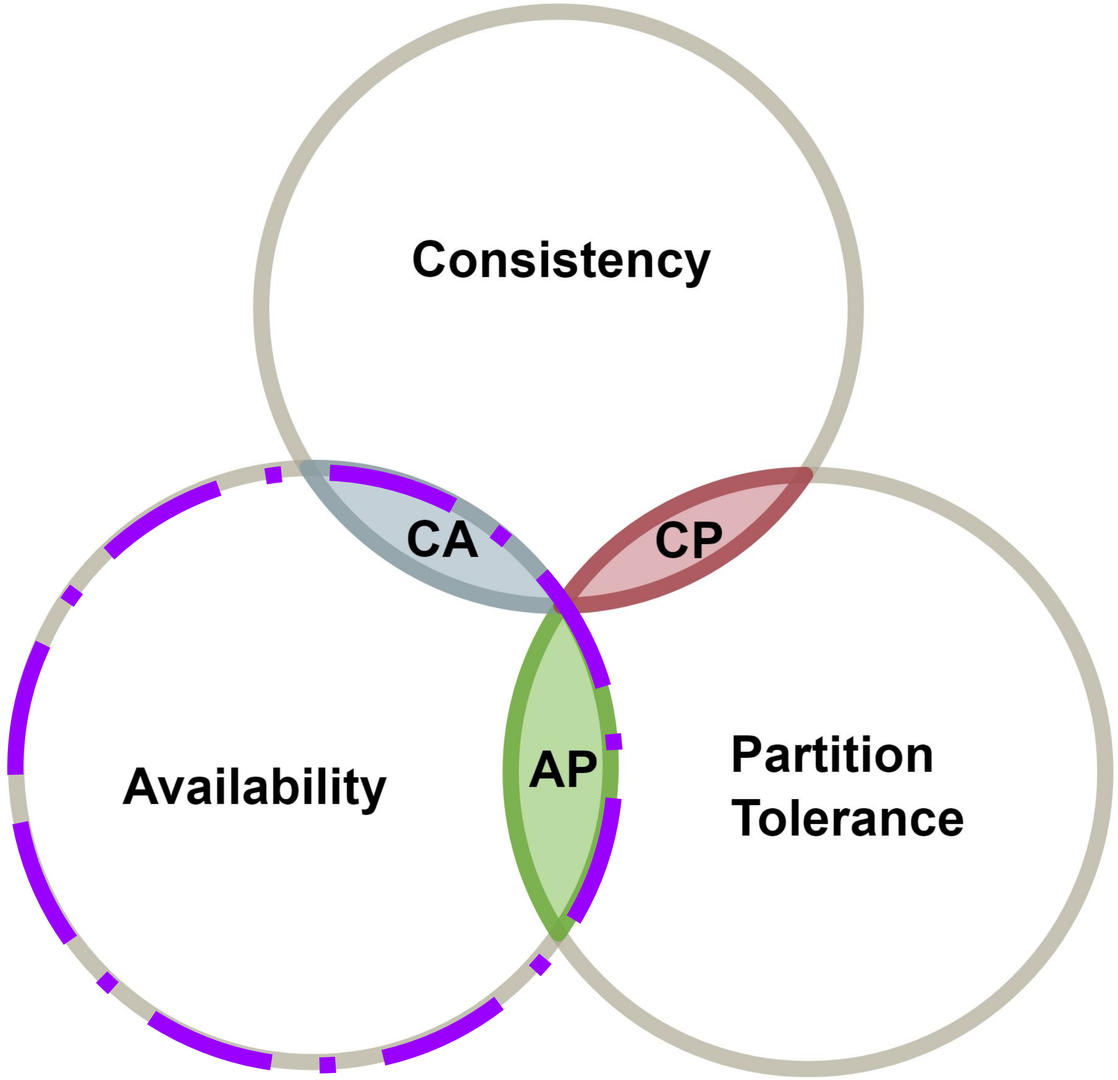
Source:
https://www.youtube.com/watch?v=Jw1iFr4v58M

CAP THEOREM

AVAILABILITY

Image source:
http://berb.github.io/diploma-thesis/original/resources/cap.svg

Availability:
Every node has consistent read and write access



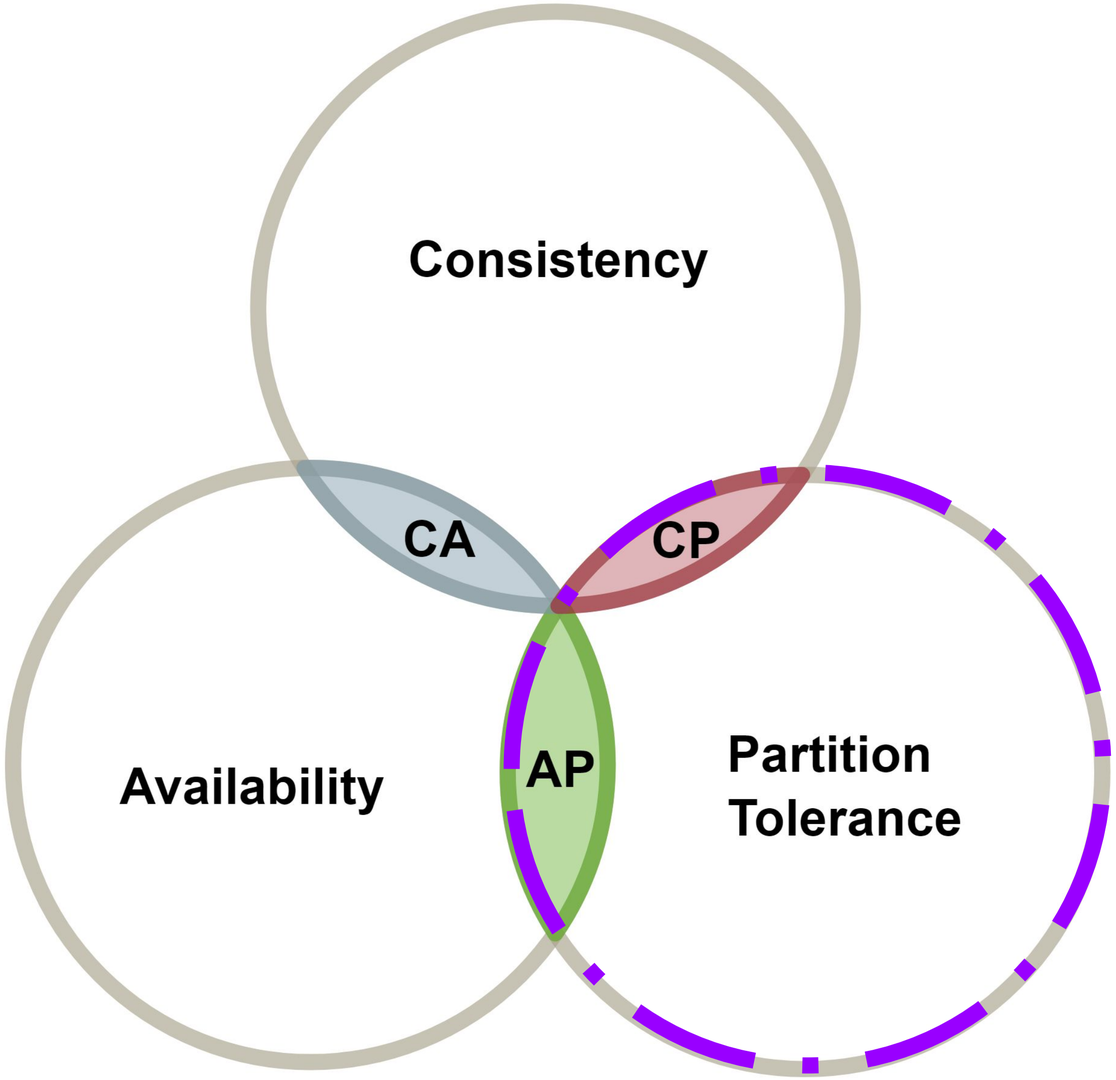
Source:
https://www.youtube.com/watch?v=Jw1iFr4v58M

CAP THEOREM

PARTITION TOLERANCE

Image source:
http://berb.github.io/diploma-thesis/original/resources/cap.svg

Partition Tolerance:
The system works despite partitions in the network



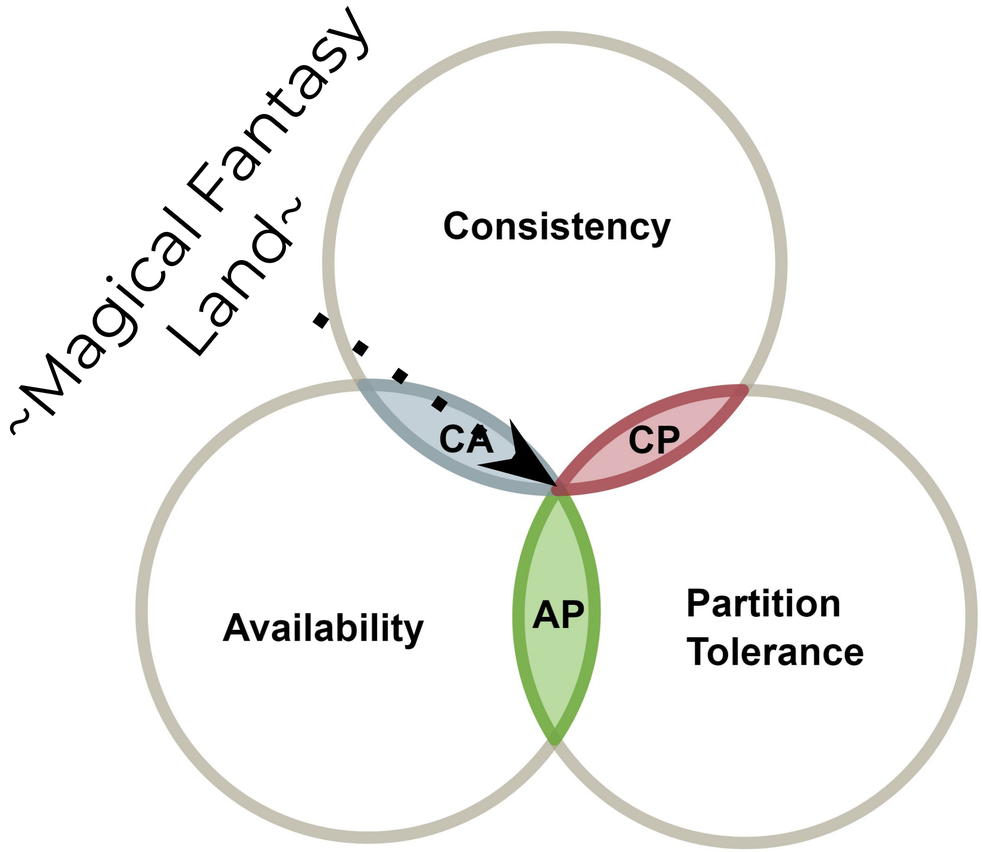
Source:
https://www.youtube.com/watch?v=Jw1iFr4v58M

CAP THEOREM

LIMITATIONS

Image source:
http://berb.github.io/diploma-thesis/original/resources/cap.svg

Can only have **two of three**

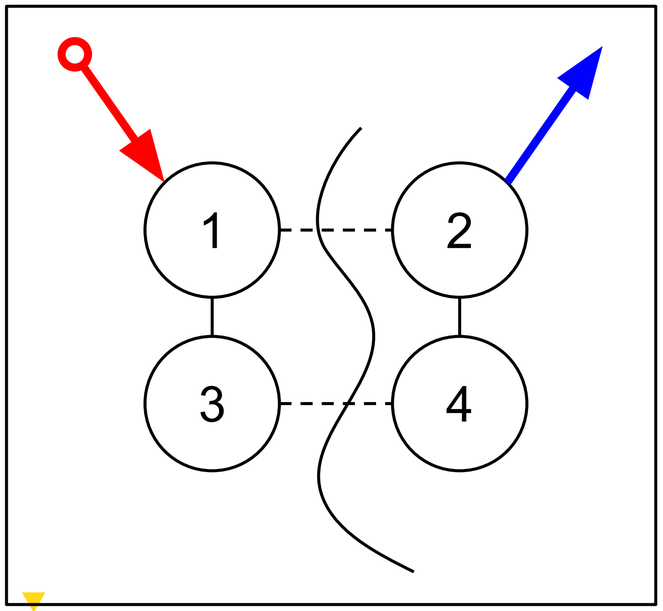


Source:
https://www.youtube.com/watch?v=Jw1iFr4v58M

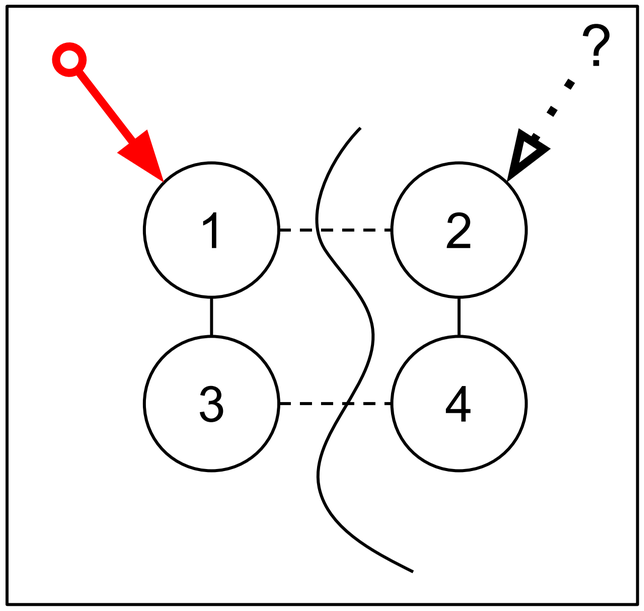
CAP THEOREM

PROOF

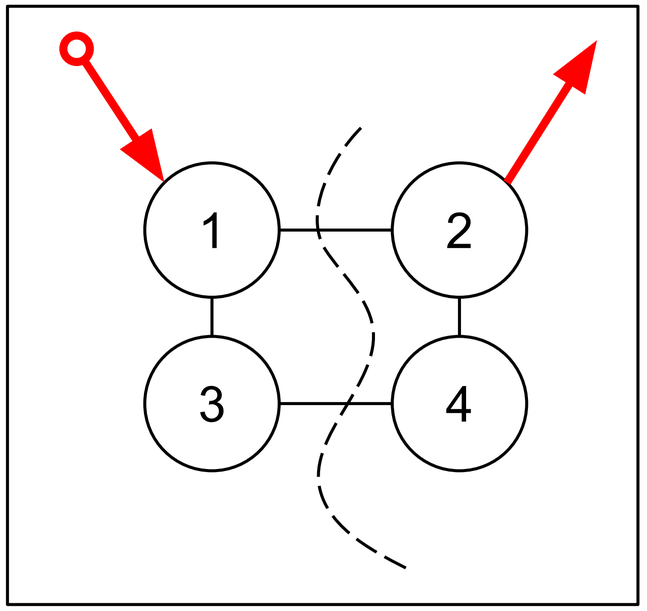
Partition Tolerant + Available = **Not Consistent**



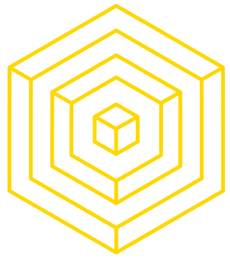
Partition Tolerant + Consistent = **Not Available**



Consistent + Available = **Not Partition Tolerant**



QUESTIONS?



1.4 BYZANTINE GENERALS' PROBLEM

BYZANTINE FAULT TOLERANCE

BYZANTINE GENERALS' PROBLEM

Byzantine Generals' Problem:

- How do we ensure that all honest generals make the same decision?
  - Spoiler alert: No solution for ⅓ or greater corruption
- Solution to all remaining cases: **Practical Byzantine Fault Tolerance**

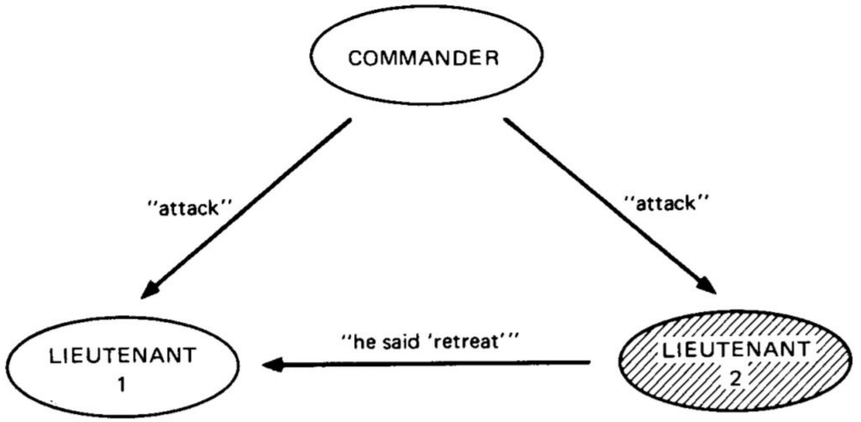


Fig. 1. Lieutenant 2 a traitor.

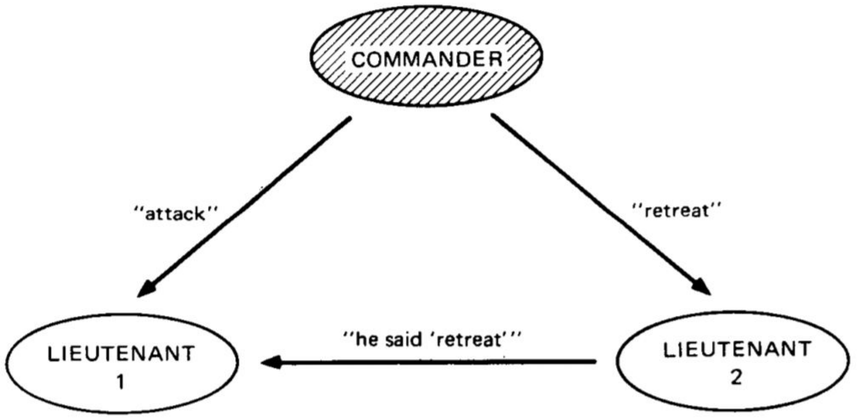


Fig. 2. The commander a traitor.

Source:
http://www-inst.eecs.berkeley.edu/~cs162/fa12/hand-outs/Original_Byzantine.pdf

BYZANTINE FAULT TOLERANCE

DISTINGUISHING FAULT CATEGORIES

Assumptions of fault tolerant systems vs Byzantine fault tolerant systems:

- **Fail-stop fault:** Nodes can crash, not return values, crash detectable by other nodes
- **Byzantine fault:** Nodes can do all of the above *and* send incorrect/corrupted values, corruption or manipulation harder to detect

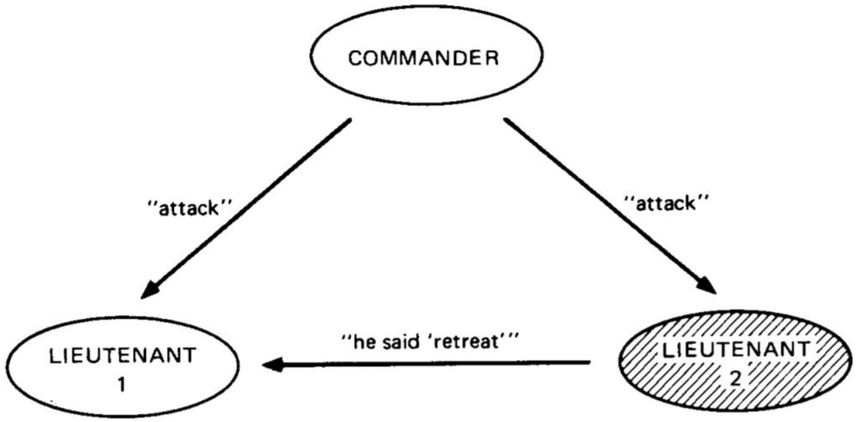


Fig. 1. Lieutenant 2 a traitor.

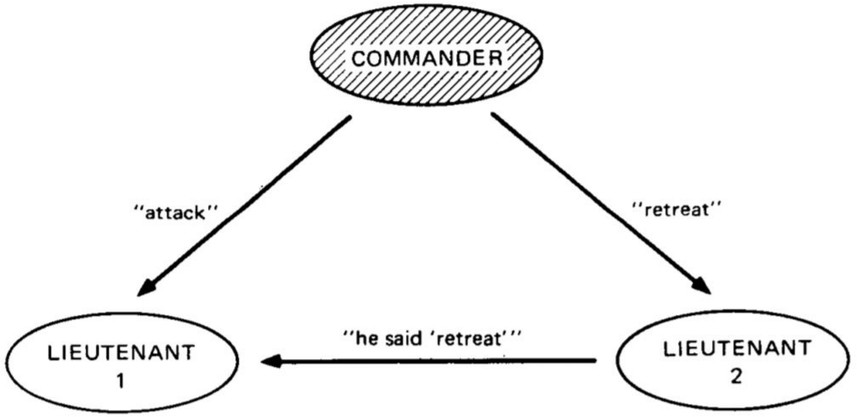
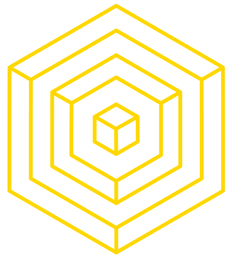


Fig. 2. The commander a traitor.

Source:
http://www-inst.eecs.berkeley.edu/~cs162/fa12/hand-outs/Original_Byzantine.pdf

QUESTIONS?



2 PROOF-OF-SOMETHING-ELSE

PROOF-OF-SOMETHING-ELSE

PROOF-OF-STAKE

Quick review:

- 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/

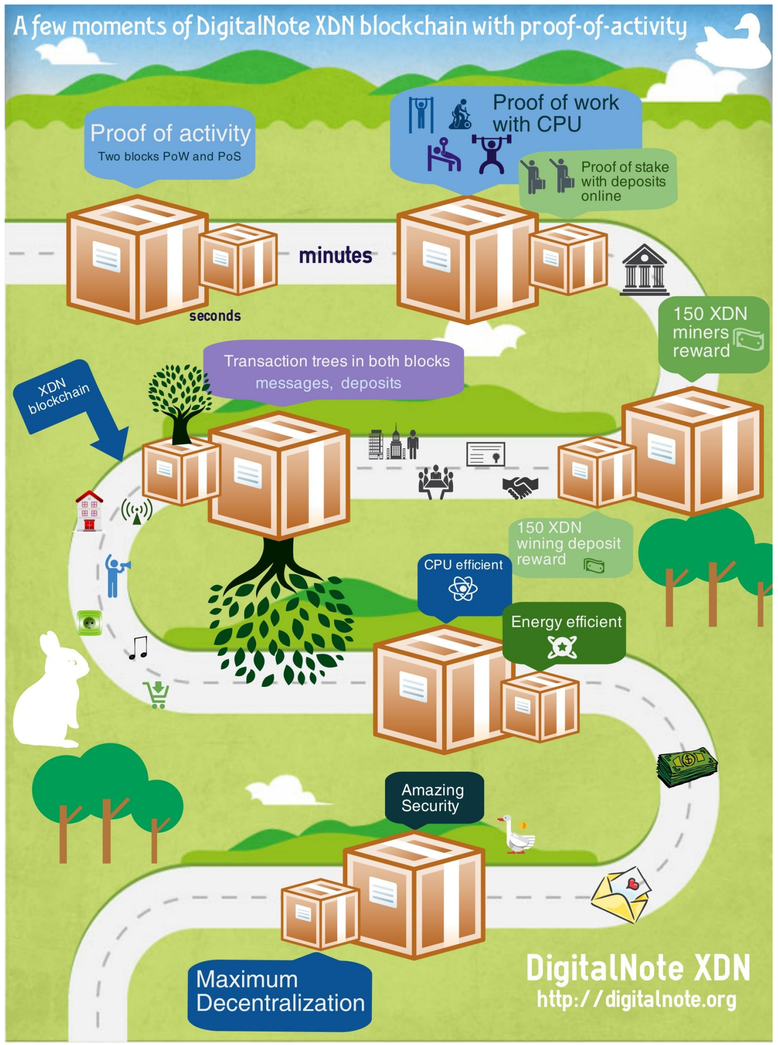
PROOF-OF-SOMETHING-ELSE

PROOF-OF-ACTIVITY

PoW + PoS hybrid:

- PoW: Miners solve and submit block headers to the system (with or without transactions, depending on implementation)
- PoS: Validators vote on valid blocks, receiving awards for signing

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



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

PROOF-OF-SOMETHING-ELSE

PROOF-OF-BURN

Overview:

- Send coins to irretrievable address
  - More coins burned, higher likelihood of voting
- Like Proof-of-Stake, but edgier

Bootstrapping mechanism:

- Tie a coin's value to some other coin (e.g. Bitcoin) by demonstrating that users willingly burn Bitcoin to receive this other coin

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/

PROOF-OF-SOMETHING-ELSE

PROOF-OF-SPACE

Overview:

- Use disk space to solve challenge
- Nowadays used for file storage
- Variations on implementations:
  - Let users receive coins after voting on blocks (Also known as Proof-of-Capacity)
  - Give rewards directly for storing information (Filecoin, Storj, Sia)

Resource(s) consumed: storage space

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/

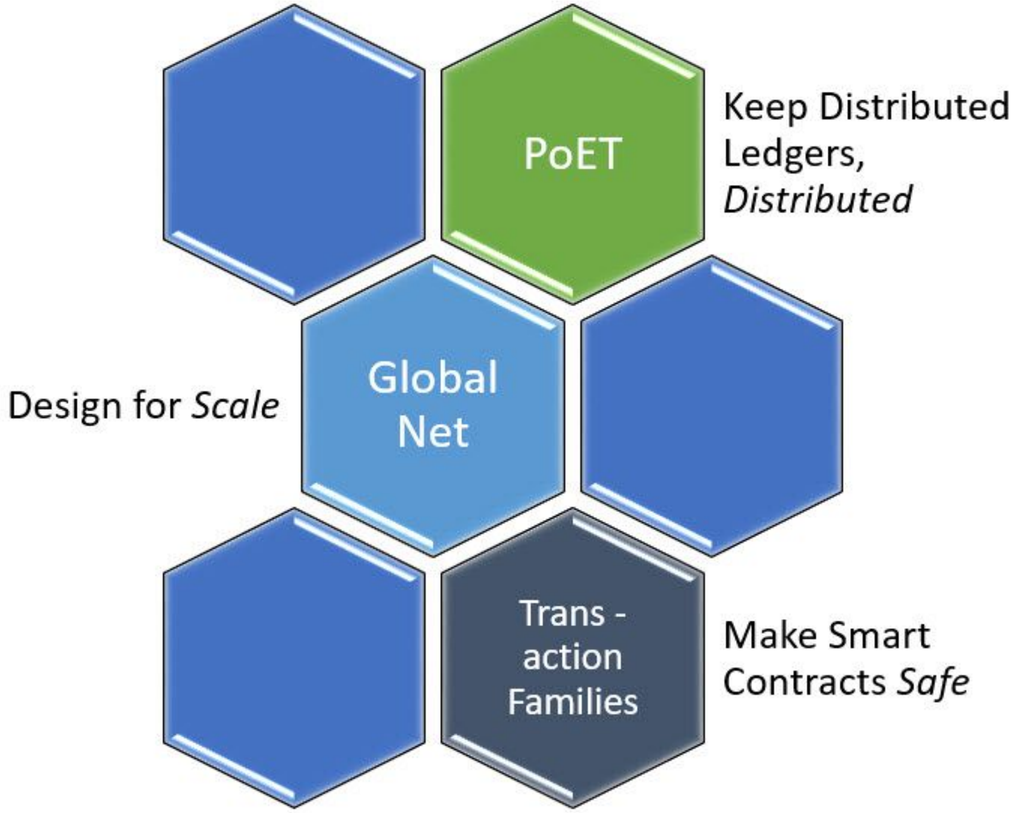
PROOF-OF-SOMETHING-ELSE

PROOF-OF-ELAPSED-TIME

Overview:

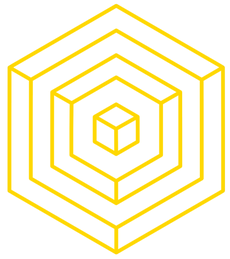
- Spend time instead of mining power
  - Uses Trusted Execution Environments (TEEs), particularly Intel SGXs
- Ask your SGX to wait some amount of time decided by the machine, wait for an attribution that it has indeed waited that much time
  - Let the first person to complete this random wait win
- Based on assumptions of randomness and trust in manufacturer (Intel)

Resource(s) consumed: time



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

QUESTIONS?



3

PROOF OF STAKE

PROOF-OF-STAKE

IMPLEMENTATIONS

- **Tendermint:** mostly asynchronous, BFT consensus protocol with voting power denominated in validator stake; two-stage voting process; favors safety (consistency). Used in blockchain network Cosmos.
- **Casper:** Two versions:
  - *Casper the Friendly GHOST*: Correct-by-Construction (Vlad), which designs the protocol from scratch
  - *Casper the Friendly Finality Gadget* (Vitalik), a PoW/PoS hybrid that uses checkpoints

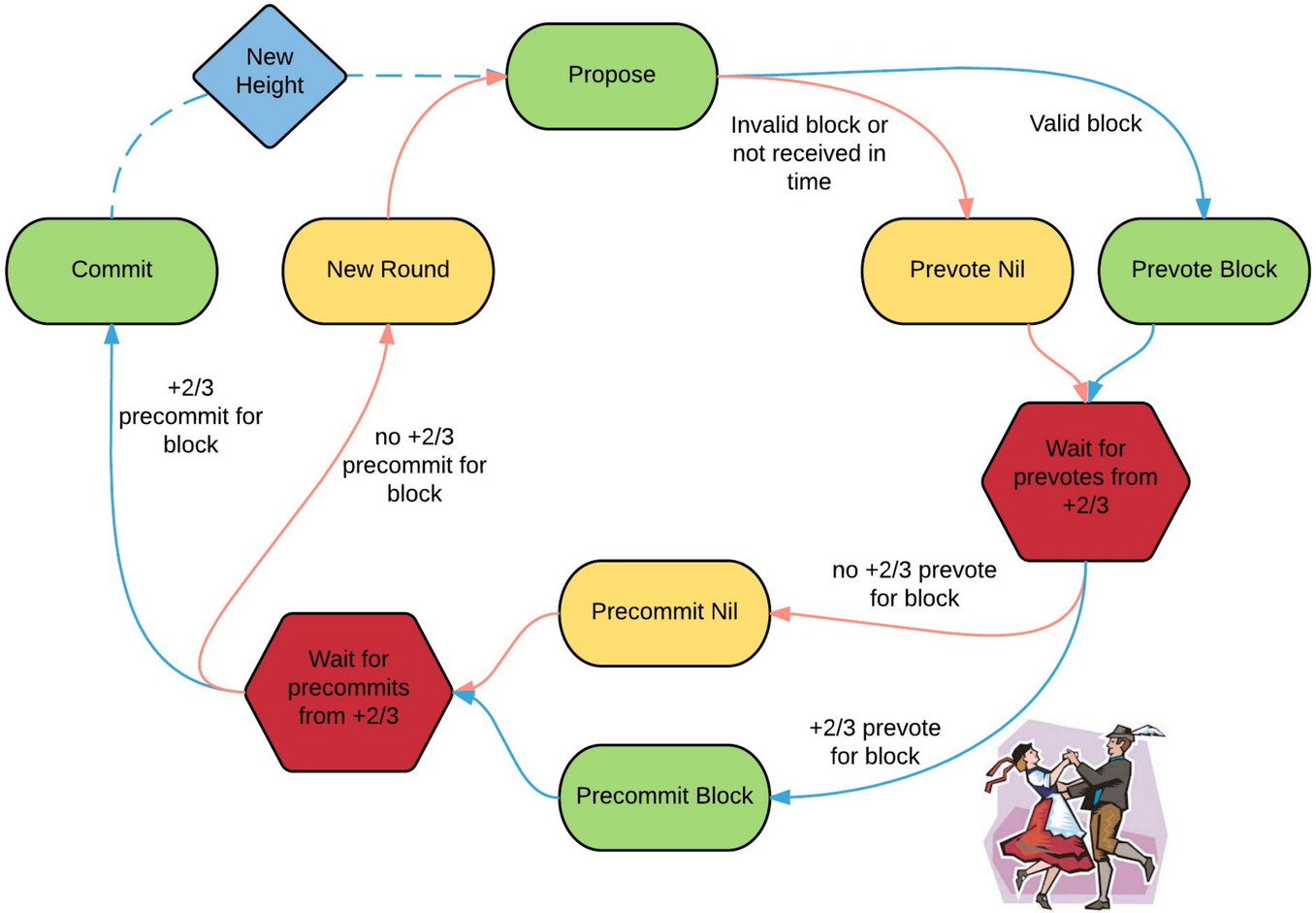
COSMOS

INTERNET OF BLOCKCHAINS

PROOF-OF-STAKE

TENDERMINT

- BFT-based PoS:
  - A chosen set of validators of a specific size at any given point with bonded stake
  - Weakly asynchronous
  - Favors consistency over availability
  - ~~Requires dancing experience~~
  - Energy efficient & fast
    - Excellent for enterprise blockchains



Source: http://tendermint.readthedocs.io/projects/tools/en/master/introduction.html

PROOF-OF-STAKE

CASPER

Two versions, one by Vitalik and one by Vlad Zamfir

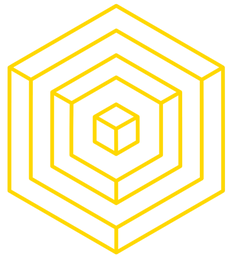
- Casper the Friendly Finality Gadget is a chain/BFT-style hybrid
  - Anyone can bond tokens
  - Initially PoW/PoS hybrid, a slow transition to pure PoS
- Casper the Friendly Ghost is a pure PoS protocol
  - Focuses on safety
  - Maintains validator set

casper

Source: https://media.consensys.net/casper-smart-contract-consensus-7be6cfa6f7ec

QUESTIONS?


BREAK
SECTION
BLOCKCHAIN
AT BERKELEY



4 VOTING-BASED CONSENSUS ALGORITHMS

Source: http://greece10best.com/wp-content/uploads/2017/05/Paxos.jpg

PAXOS

OVERVIEW

Source: http://paxos-map.com/paxos-island-general-information/

PAXOS

OVERVIEW

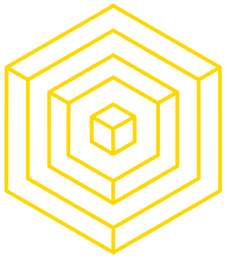
PAXOS
OVERVIEW

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/

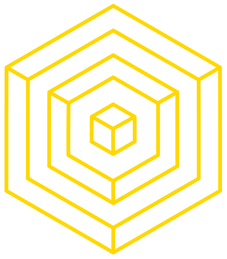


PAXOS

TERMINOLOGY

Proposer: advocates a client request, moves protocol forward
Acceptor: a voter
Learner: remembers result, send message to the client

Quorum:
- any majority of Acceptors
- subset of Acceptors, such that any two Quorums share at least one member (they must overlap)



PAXOS

MAIN IDEA

Protocol proceeds over several rounds
where each successful round has 2 phases

1. Client talks to Paxos
2. Within Paxos, Proposer, Acceptor, and Learner discuss
   a. Pass “ballots”
   b. Ballot has number and value
3. Learner talks to Client

PAXOS IN PRACTICE

ASSUMPTIONS AND REAL WORLD USE

- Nodes do not try to subvert the protocol (messages delivered without corruption)
- Only works for fail-stop (no Byzantine failures) faults
- Failed nodes at any point won't affect the entire network
- Good performance (fast)
- Generally used to replicate large sets of data

RAFT
OVERVIEW

Raft is another consensus algorithm 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

PROOF-OF-STAKE

PROBLEM

In Proof-of-Stake, we have the problem of “the rich get richer”

- Large stakeholders stake their balances
- Smaller stakeholders see fewer rewards and lower returns
- Centralizes staking rewards

SOURCE:
https://cointelegraph.com/news/the-inevitable-failure-of-proof-of-stake-blockchains-and-why-a-new-algorithm-is-needed

DELEGATED PROOF-OF-STAKE

SOLUTION

Idea: Appoint **delegates** to secure the network on your behalf

- Everyone has a proportional voice in security of the network
- Delegates vote on “witnesses” to produce blocks
- If they you don’t like their behavior, “witnesses” can get voted out

SOURCE:
https://cointelegraph.com/news/the-inevitable-failure-of-proof-of-stake-blockchains-and-why-a-new-algorithm-is-needed

BITSHARES

OVERVIEW

Bitshares is a decentralized asset exchange started by Daniel Larimer in 2014

- First DPOS system
- Assets backed by core currency (BTS)

bitshares

QUESTIONS?

5 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 the we still achieve consensus?

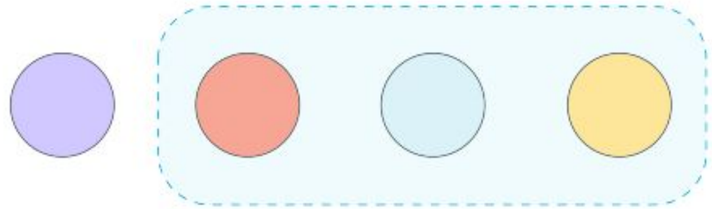
FEDERATED BYZANTINE AGREEMENT

OVERVIEW

Problem: How do we determine **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



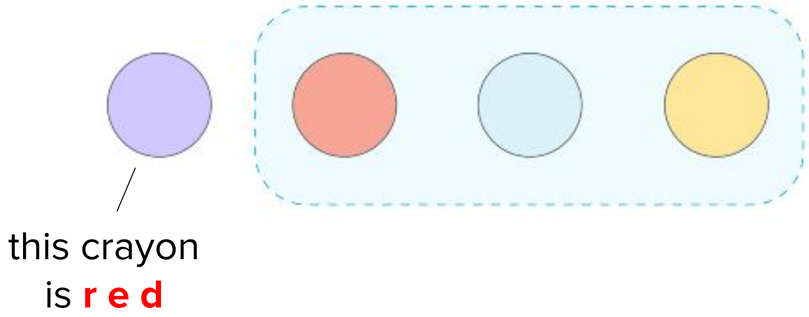
FEDERATED BYZANTINE AGREEMENT

OVERVIEW

Problem: How do we determine **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



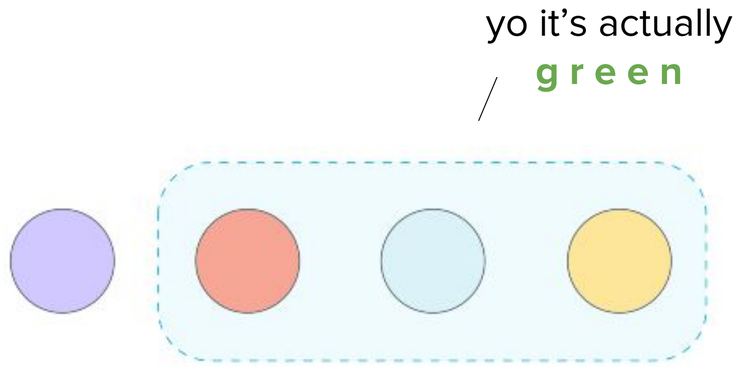
FEDERATED BYZANTINE AGREEMENT

OVERVIEW

Problem: How do we determine **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



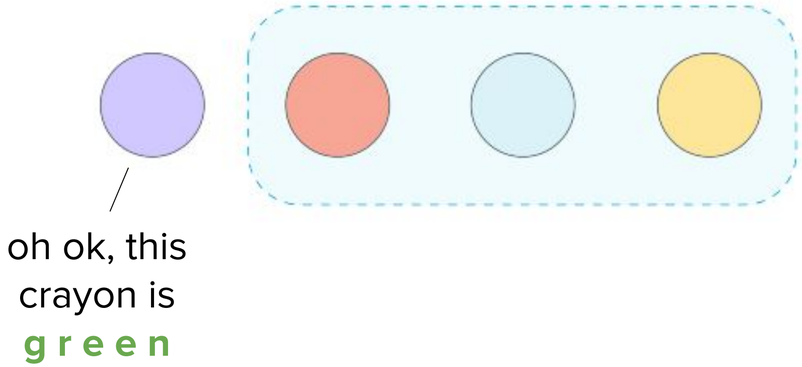
FEDERATED BYZANTINE AGREEMENT

OVERVIEW

Problem: How do we determine **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



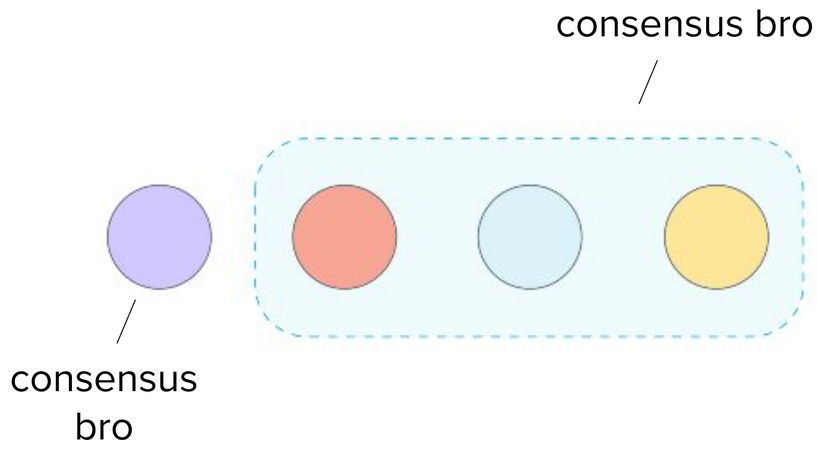
FEDERATED BYZANTINE AGREEMENT

OVERVIEW

Problem: How do we determine **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

Idea: What happens when multiple quorum slices join together?

We get a **quorum intersection**!

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

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

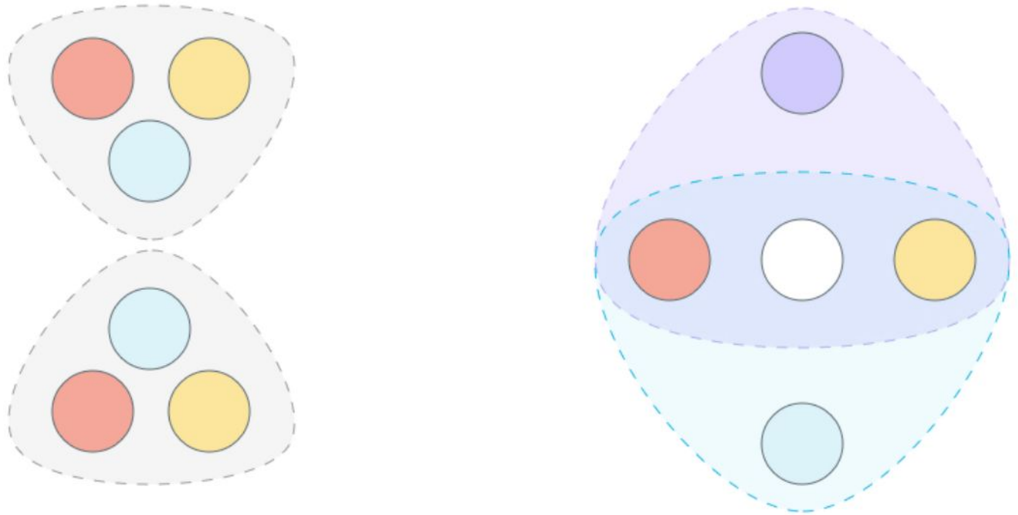
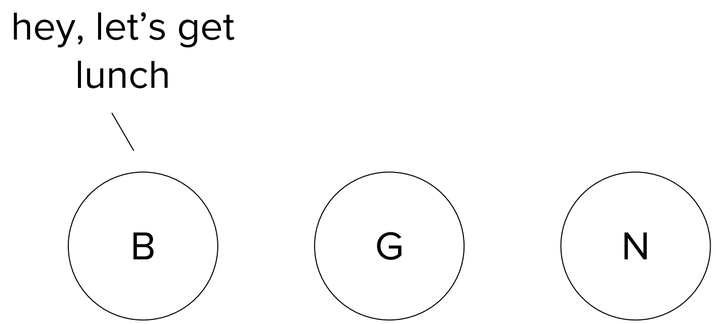


Image source: https://cdn-images-1.medium.com/max/1600/0*msL7MVVEy4p2VzhP

FEDERATED BYZANTINE AGREEMENT

EXAMPLE

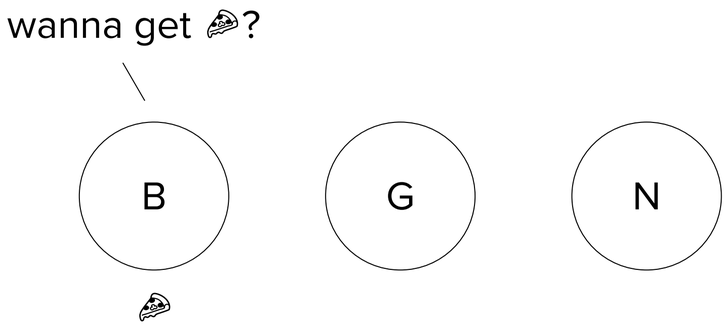
Lunchtime Consensus!



FEDERATED BYZANTINE AGREEMENT

EXAMPLE

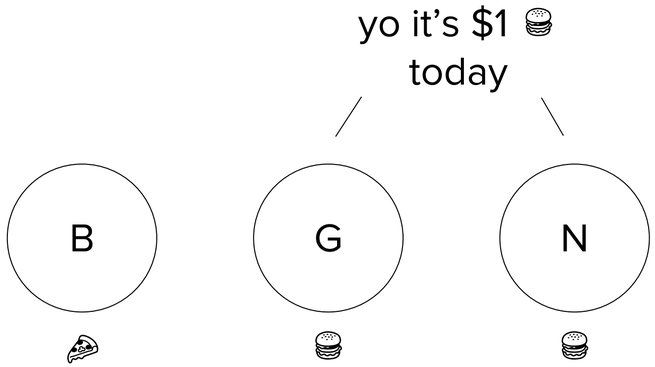
Lunchtime Consensus!



FEDERATED BYZANTINE AGREEMENT

EXAMPLE

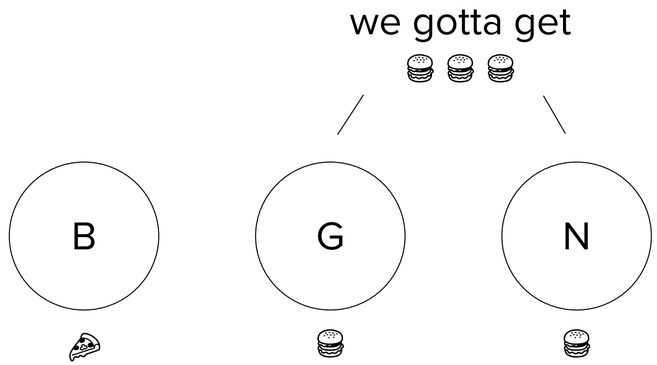
Lunchtime Consensus!



FEDERATED BYZANTINE AGREEMENT

EXAMPLE

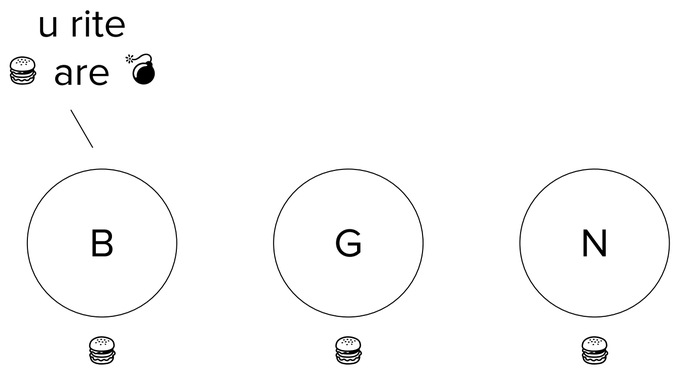
Lunchtime Consensus!



FEDERATED BYZANTINE AGREEMENT

EXAMPLE

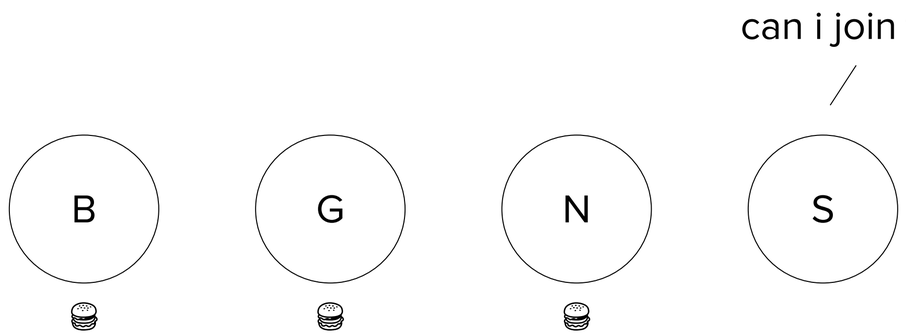
Lunchtime Consensus!



FEDERATED BYZANTINE AGREEMENT

EXAMPLE

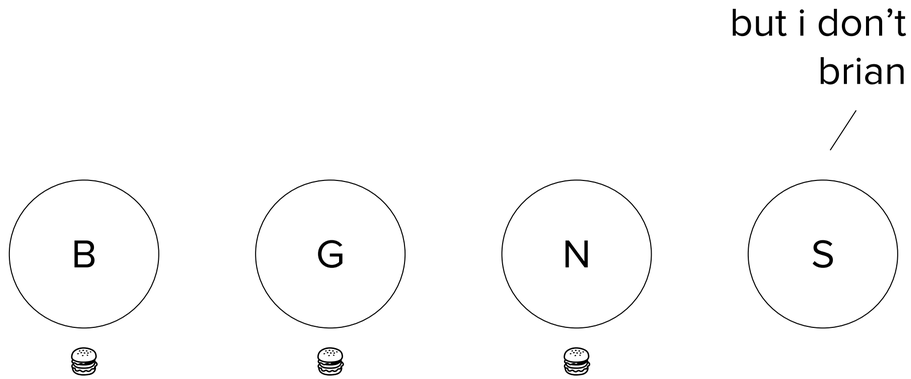
Lunchtime Consensus!



FEDERATED BYZANTINE AGREEMENT

EXAMPLE

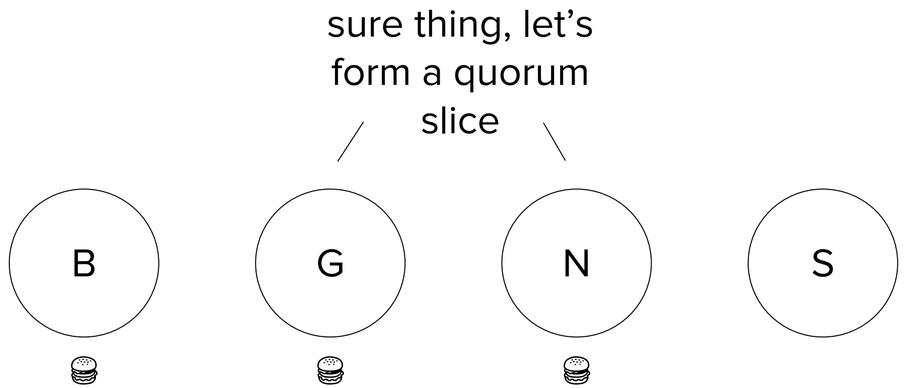
Lunchtime Consensus!



FEDERATED BYZANTINE AGREEMENT

EXAMPLE

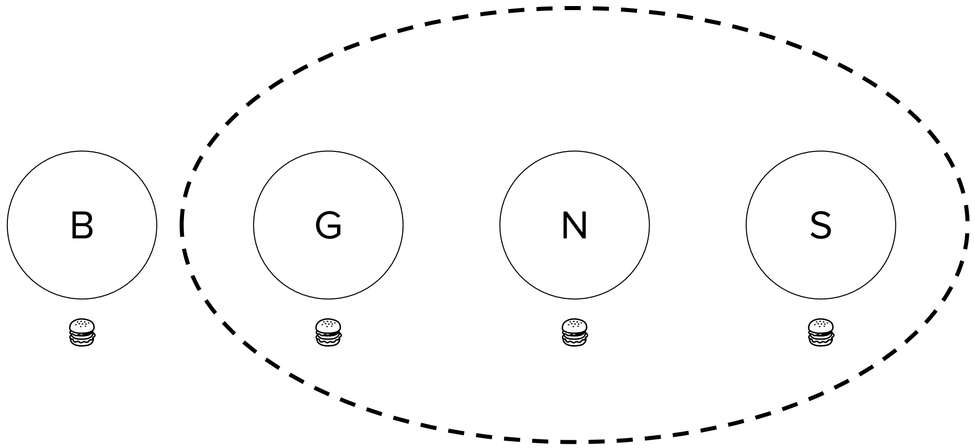
Lunchtime Consensus!



FEDERATED BYZANTINE AGREEMENT

EXAMPLE

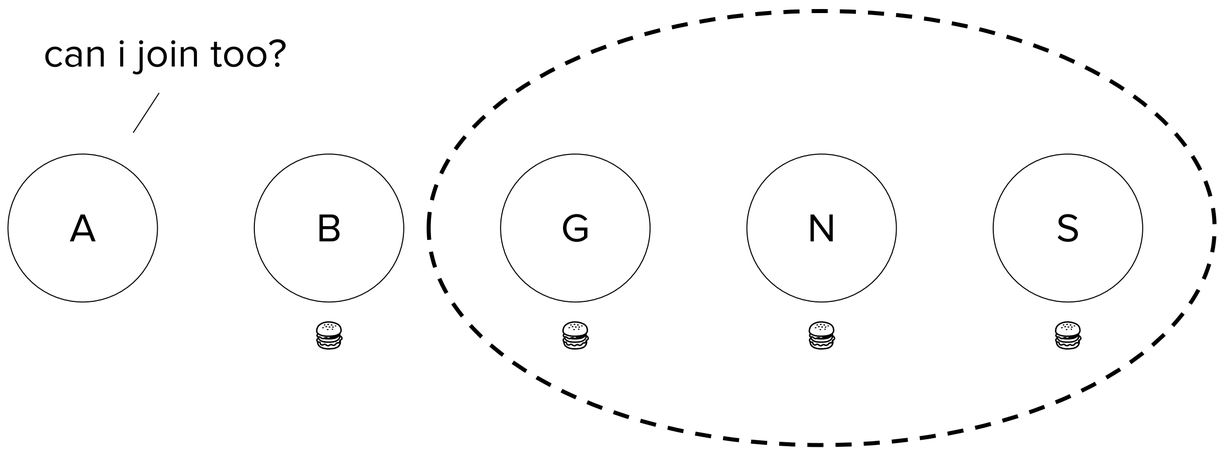
Lunchtime Consensus!



FEDERATED BYZANTINE AGREEMENT

EXAMPLE

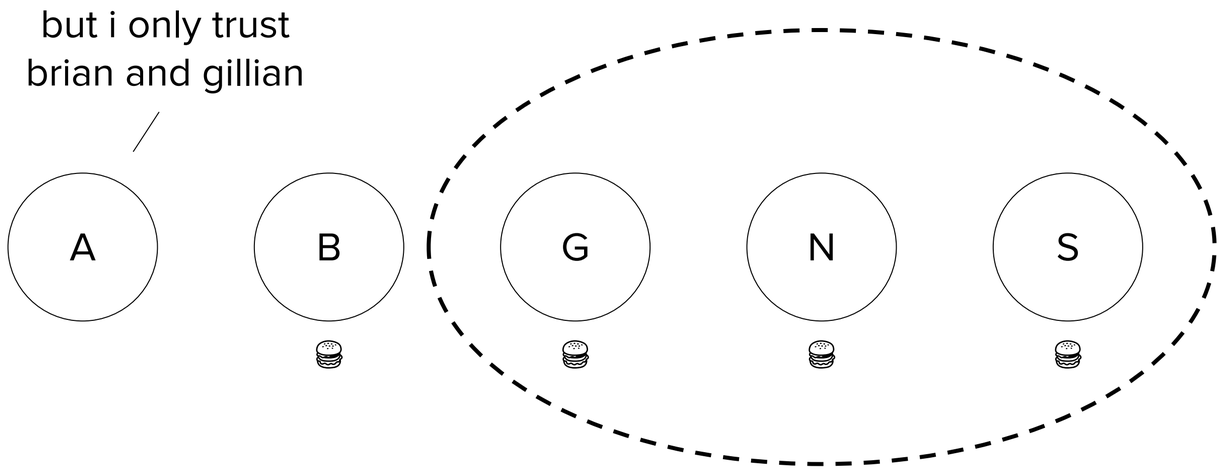
Lunchtime Consensus!



FEDERATED BYZANTINE AGREEMENT

EXAMPLE

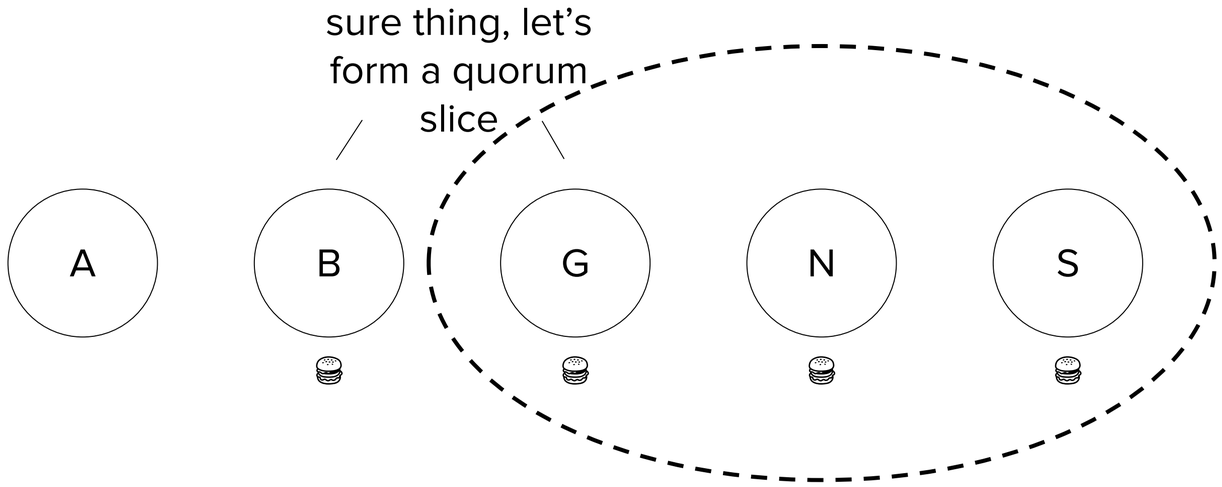
Lunchtime Consensus!



FEDERATED BYZANTINE AGREEMENT

EXAMPLE

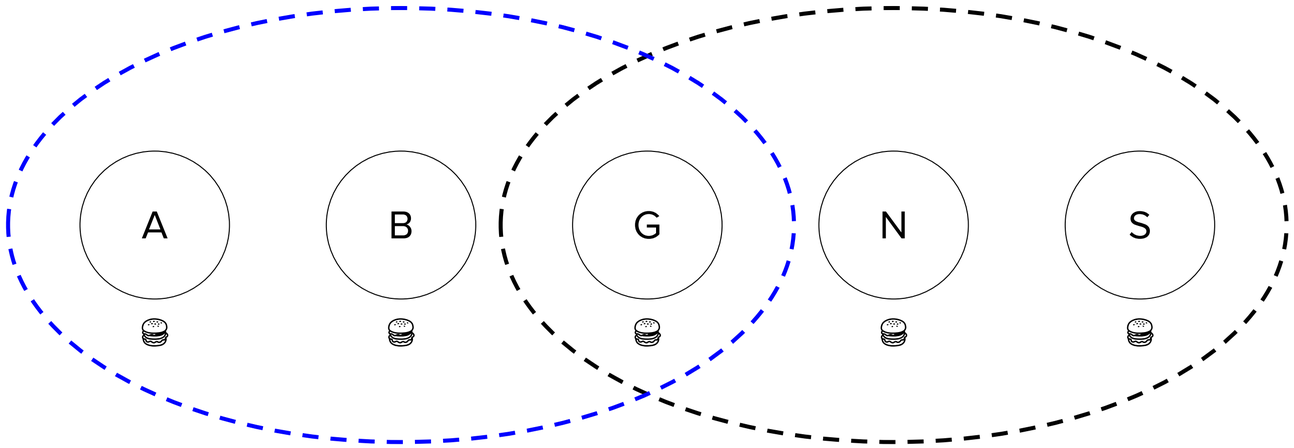
Lunchtime Consensus!

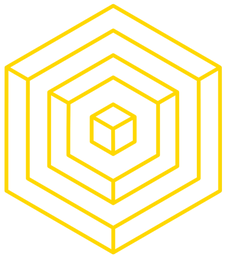


FEDERATED BYZANTINE AGREEMENT

EXAMPLE

Lunchtime Consensus!





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

RIPPLE

OVERVIEW

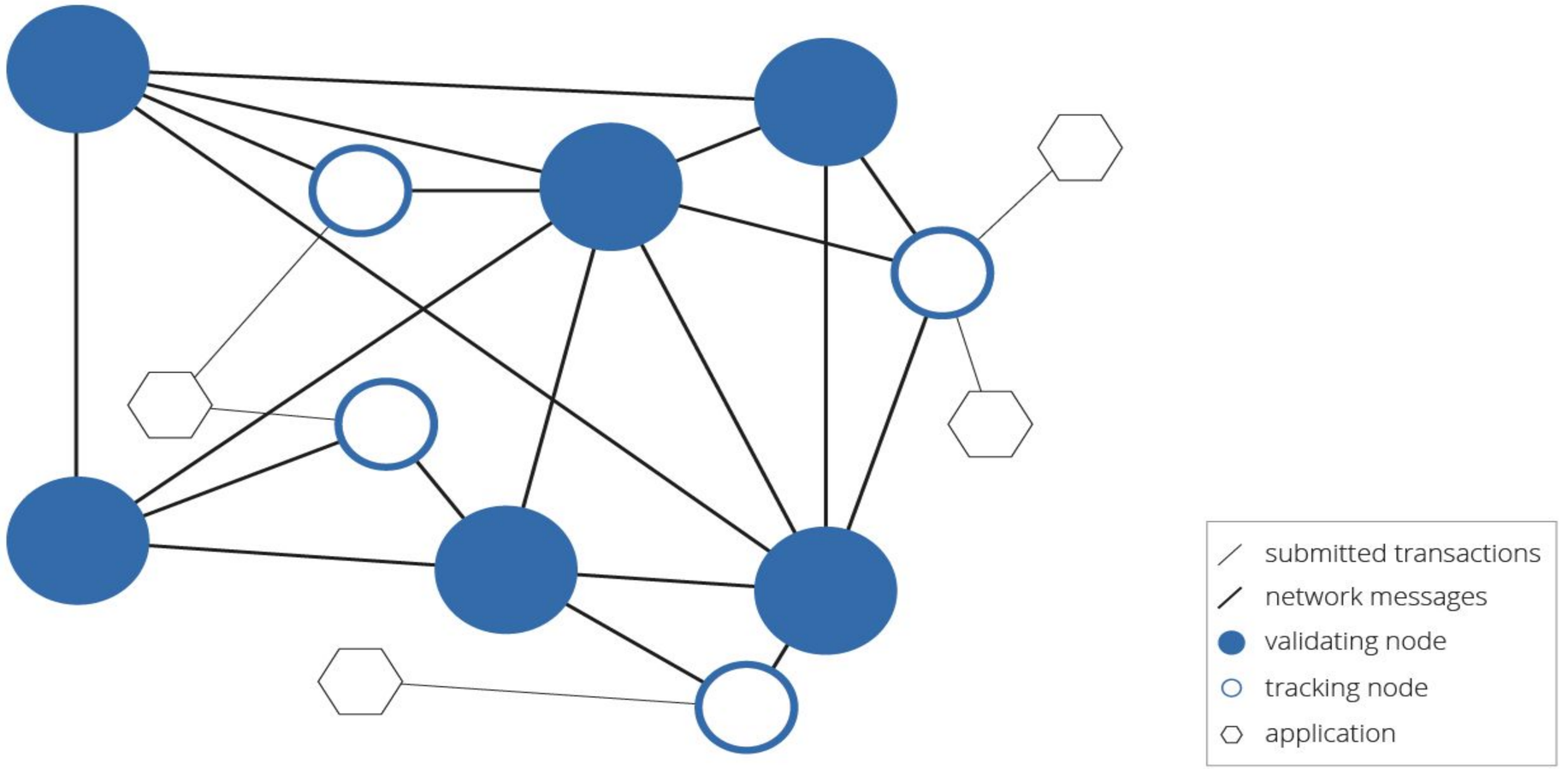
Ripple is a real-time gross settlement system (RTGS), currency exchange, and remittance network

- Distributed ledger across financial institutions
- Allows banks to send money to overseas branches in real time with very low fees via Ripple's blockchain system
- Permissioned ledger

Focus: big financial institutions

RIPPLE

OVERVIEW



SOURCE: https://ripple.com/build/xrp-ledger-consensus-process/

STELLAR

OVERVIEW

Stellar is an open-source protocol for exchanging money

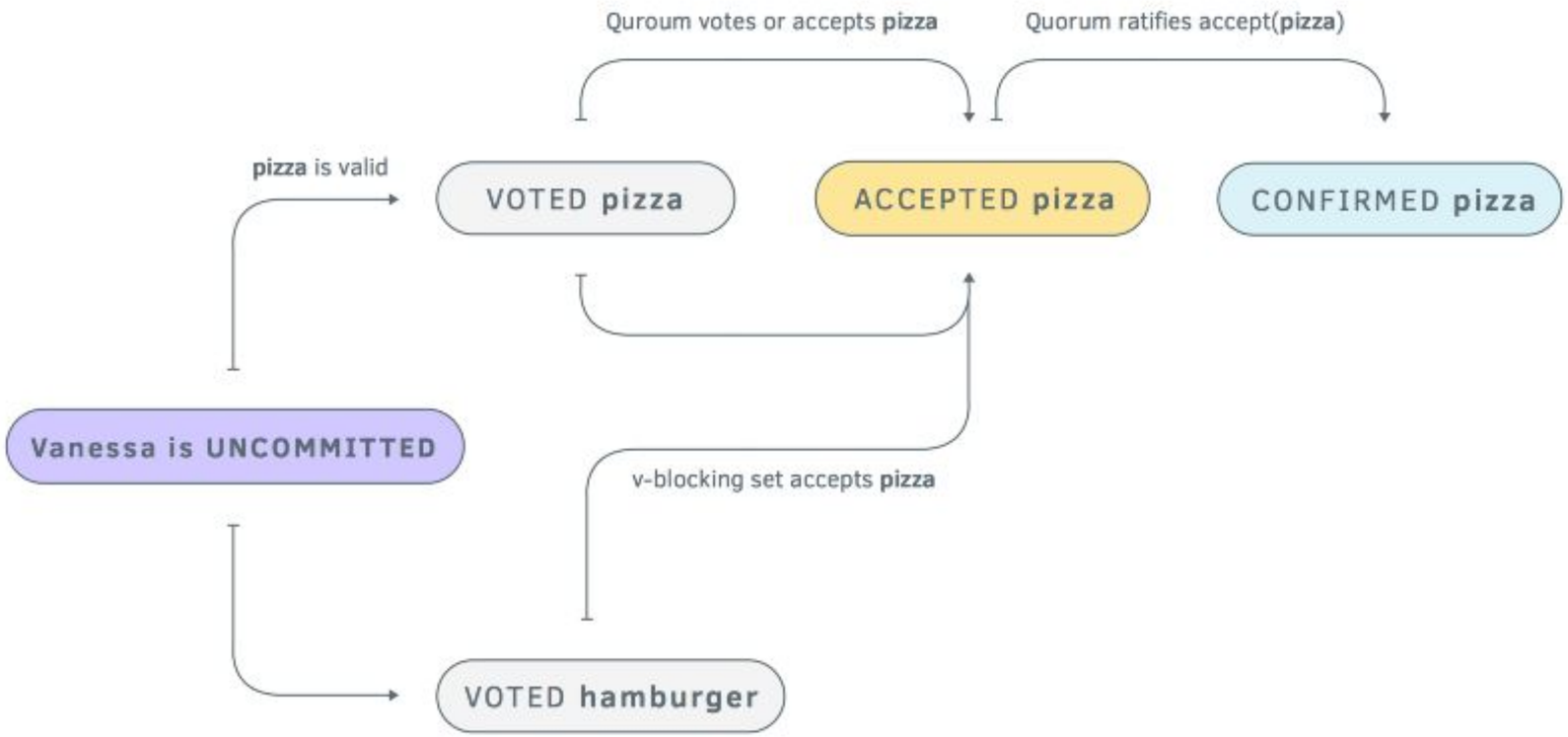
- Platform that connects banks, payments systems, and people
- Open ledger

Focus: non-profit distribution of wealth

STELLAR

STELLAR

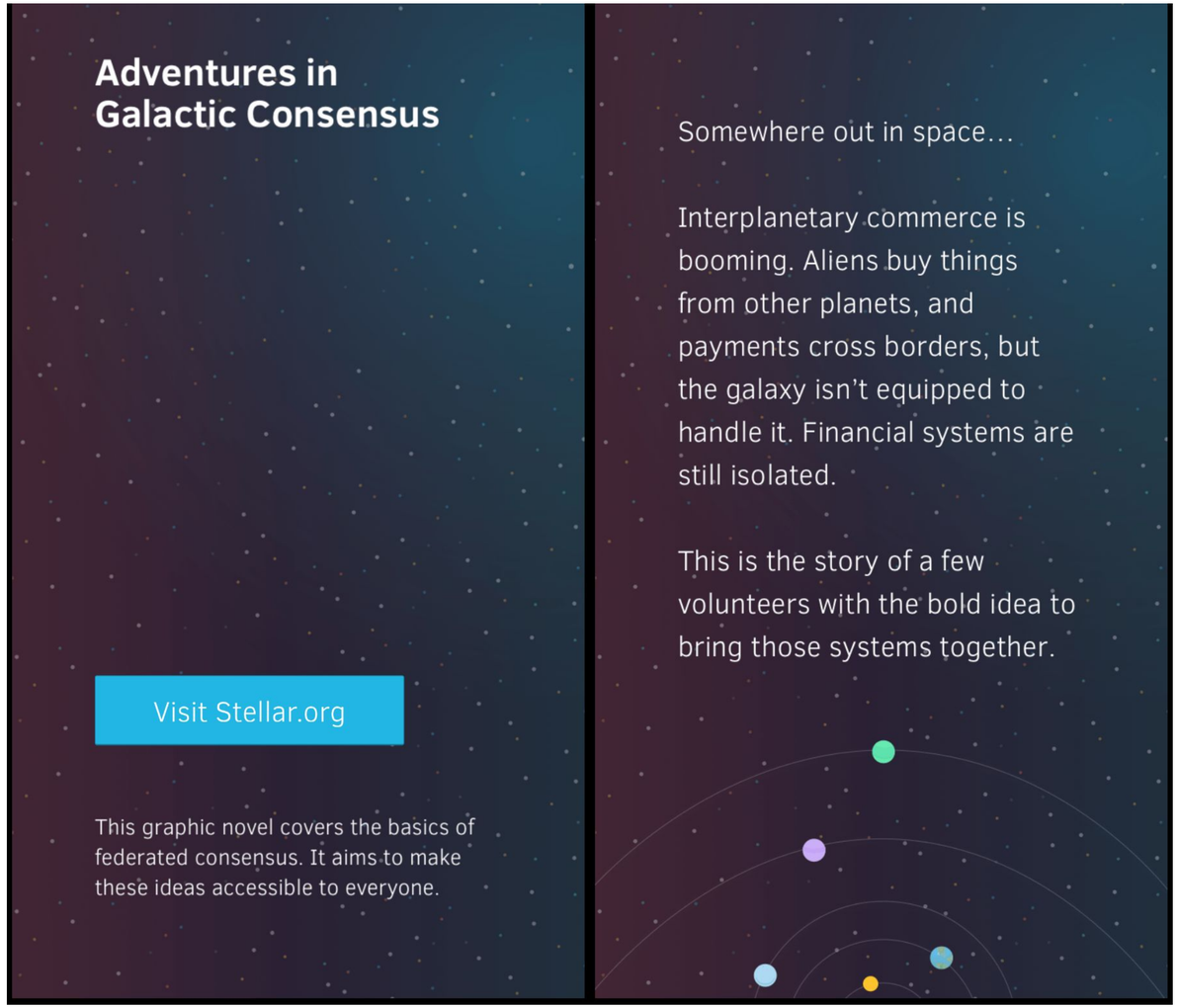
OVERVIEW



SOURCE: https://medium.com/a-stellar-journey/on-worldwide-consensus-359e9eb3e949

STELLAR

OVERVIEW



SOURCE: https://www.stellar.org/stories/adventures-in-galactic-consensus-chapter-1/

QUESTIONS?

READINGS

- Short overview of alternatives to PoW: https://www.coindesk.com/short-guide-blockchain-consensus-protocols/
- Stellar Consensus Comic: https://www.stellar.org/stories/adventures-in-galactic-consensus-chapter-1/
- Stellar Consensus Protocol Overview: https://medium.com/a-stellar-journey/on-worldwide-consensus-359e9eb3e949
- CAP Theorem Overview: https://www.youtube.com/watch?v=Jw1iFr4v58M
- Raft Overview: http://container-solutions.com/raft-explained-part-1-the-consenus-problem/

EXTRA: BYZANTINE GENERALS IRL

DIGITAL AVIONICS

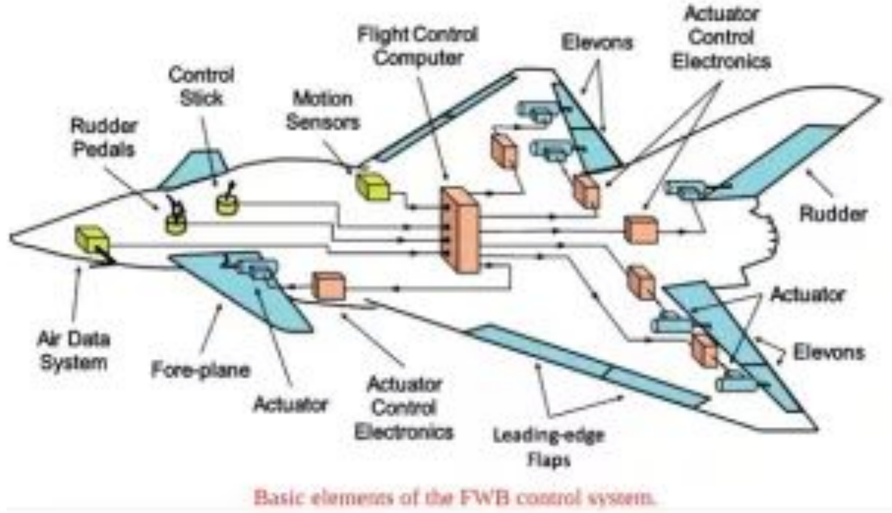
- Super dependable computers pioneered by aircraft manufacturers
- Safety first, especially…
  - when you’re flying 60-120 million USD aircraft
  - when there’s 100s of passengers

- Fault Tolerant Avionics: http://www.davi.ws/avionics/TheAvionicsHandbook_Cap_28.pdf

AUTHOR: RUSTIE LIN

EXTRA: BYZANTINE GENERALS IRL

DIGITAL AVIONICS



Basic elements of the FWB control system.

- e.g. 550 passenger Boeing 777

- ARINC 659 SAFEbus network
  - nodes: computers, sensors,
    - duplicate transmitters that control nodes
    - silence node if transmitters informed of too many errors
  - recipient nodes receive 4 copies of message
  - accept/record only if 4 identical messages

AUTHOR: RUSTIE LIN

EXTRA: BYZANTINE GENERALS IRL

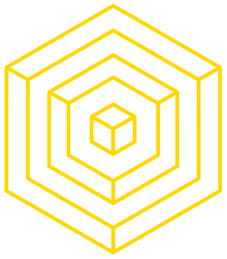
DIGITAL AVIONICS

- e.g. SpaceX Dragon

- NASA requirements for ISS
- Triply redundant computers
- Radiation events changing memory or register values

SOURCE:
https://www.universetoday.com/112272/meet-spacexs-new-manned-dragon-cool-animation-shows-how-it-works/

AUTHOR: RUSTIE LIN



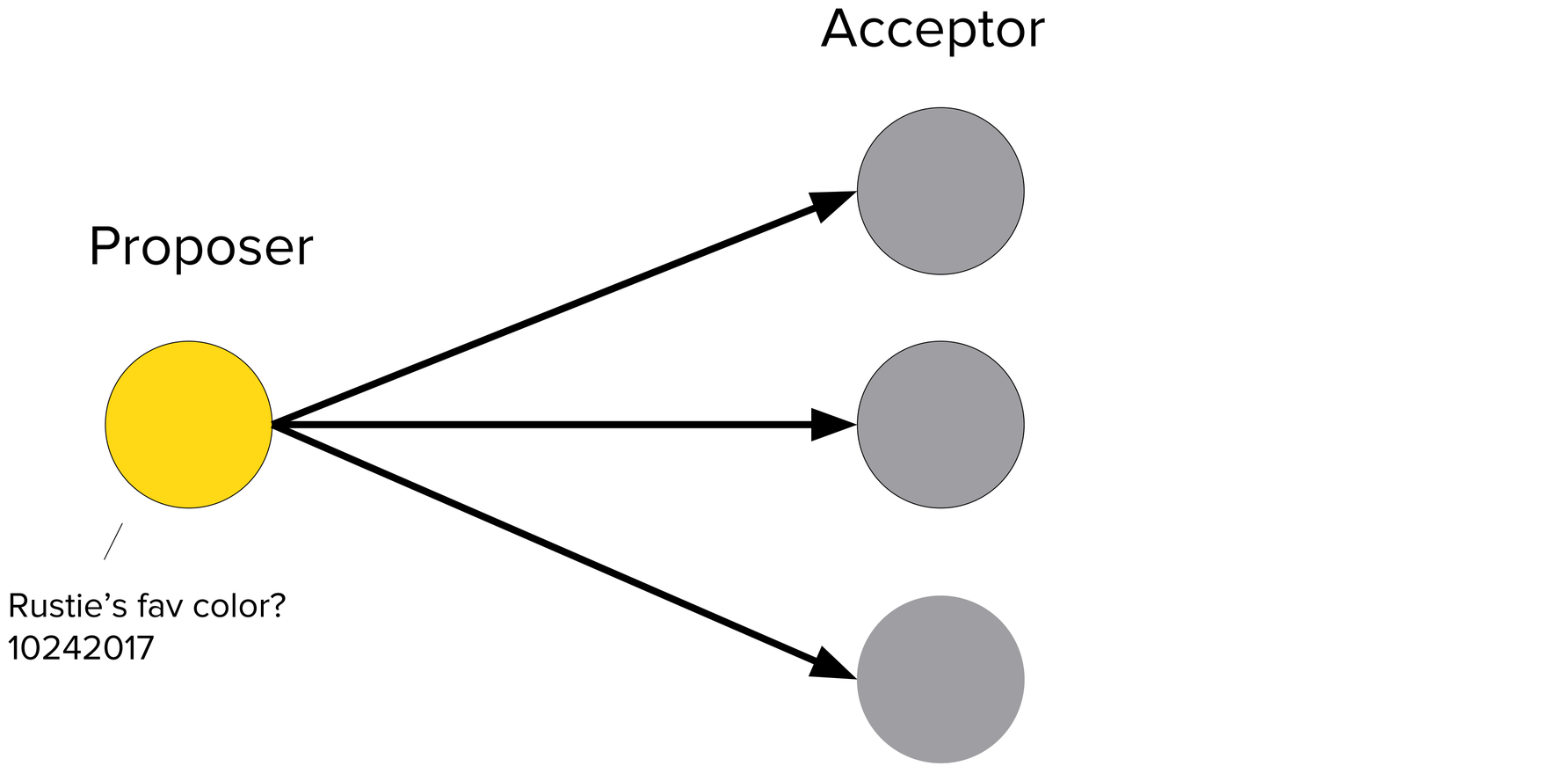
EXTRA: BASIC PAXOS

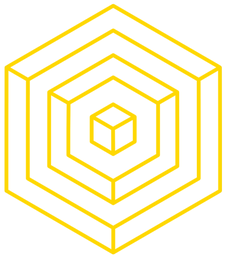
PHASE 1A: PREPARE

Proposer creates proposal N, sends to quorum of acceptors

EXTRA: BASIC PAXOS

PHASE 1A: PREPARE





EXTRA: BASIC PAXOS

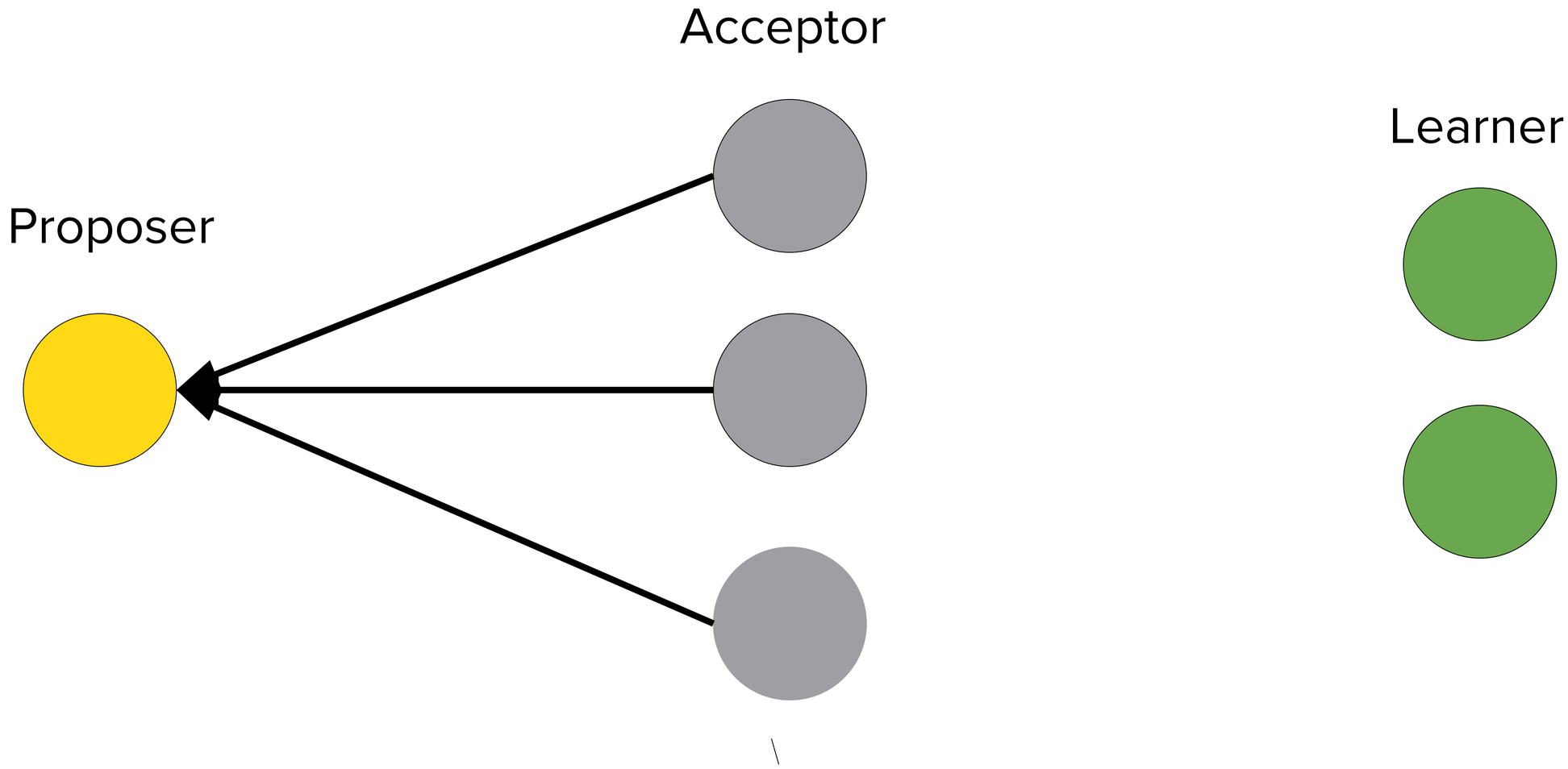
PHASE 1B: PROMISE

After getting proposal N, Acceptors promise to ignore proposals less than N, response to Proposer

If Acceptor had accepted proposal in the past, send previous proposal number and value back to Proposer, along with Promise

EXTRA: BASIC PAXOS

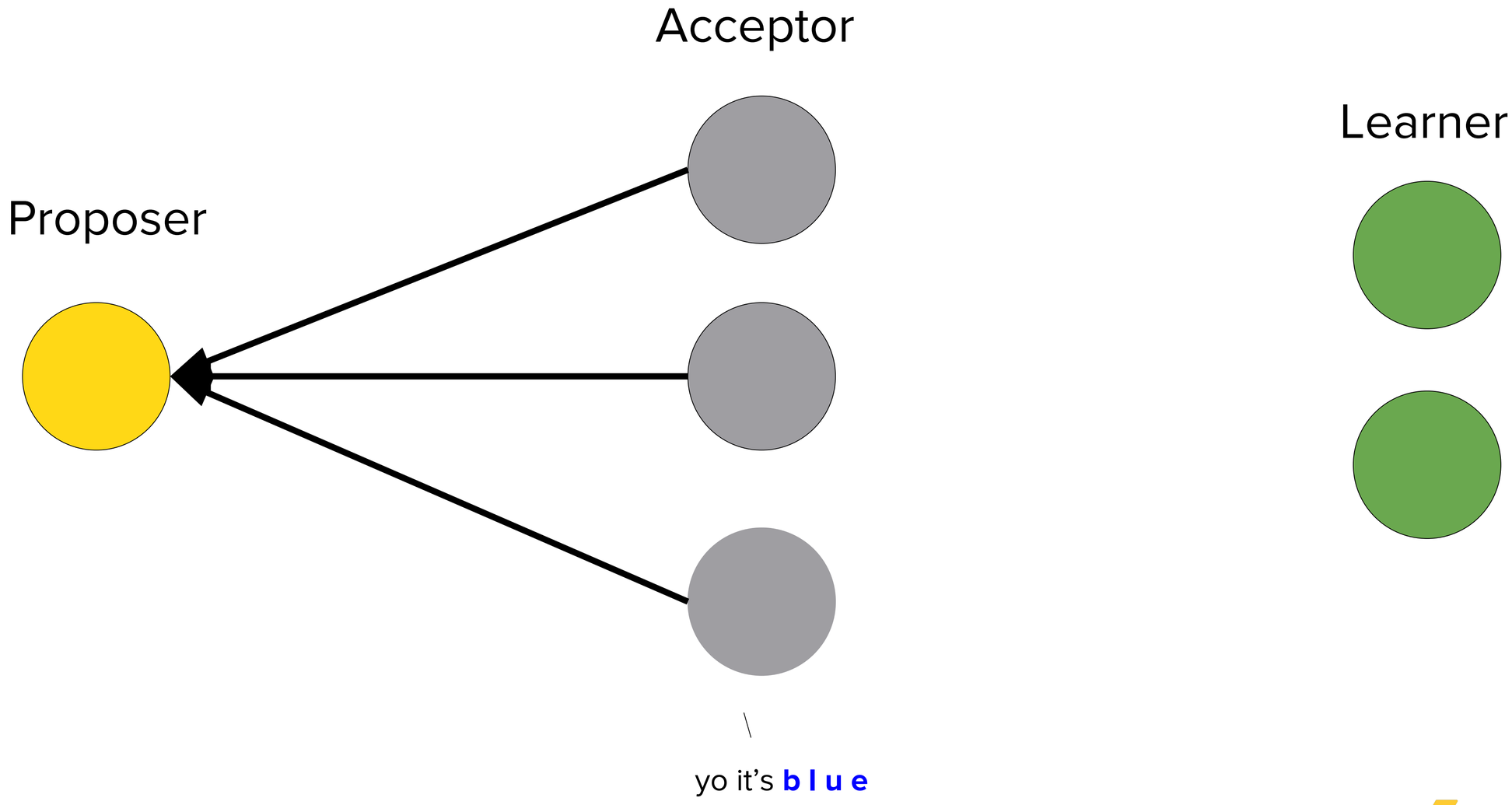
PHASE 1B: PROMISE

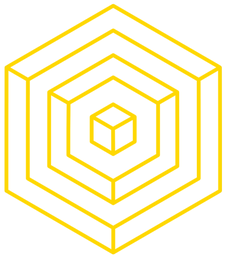


wait idk, lmk what it is pls

EXTRA: BASIC PAXOS

PHASE 1B: PROMISE





EXTRA: BASIC PAXOS

PHASE 2A: ACCEPT REQUEST

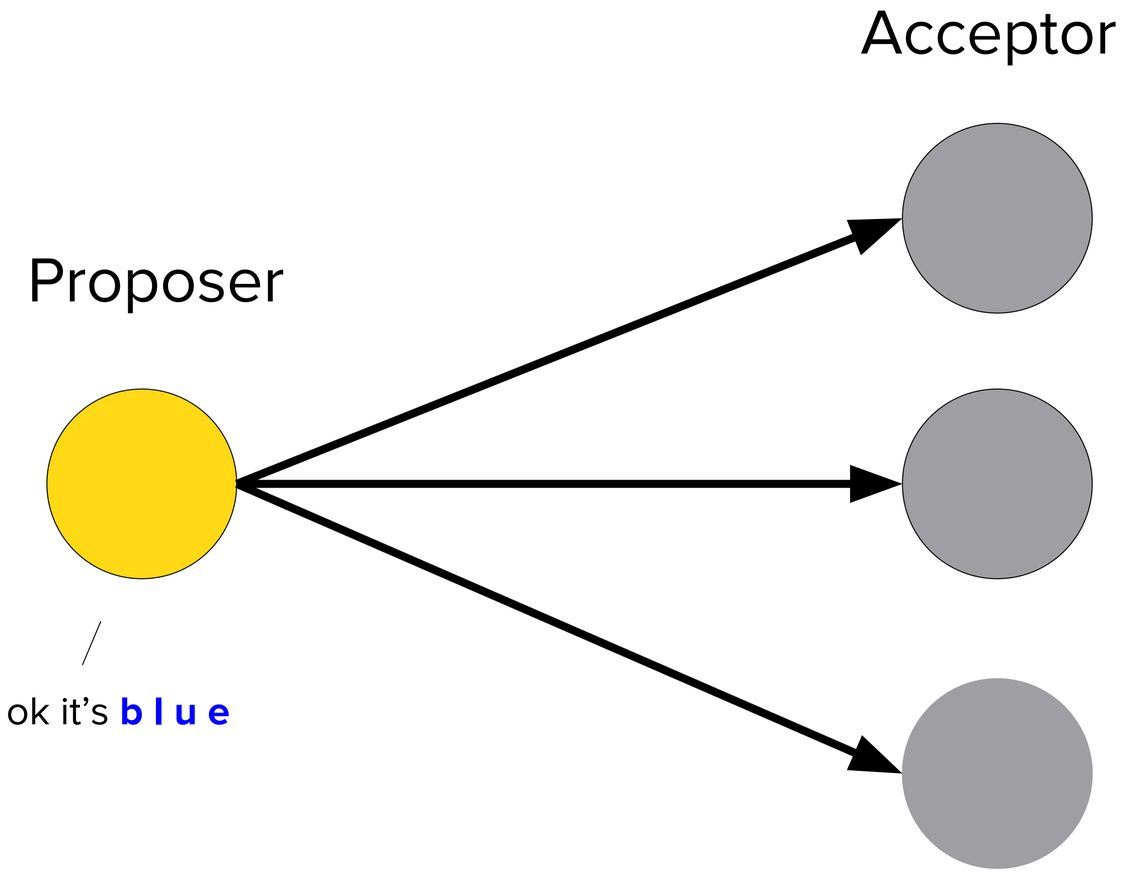
If Proposer gets enough promises from a Quorum of Acceptors, need to set a proposal value

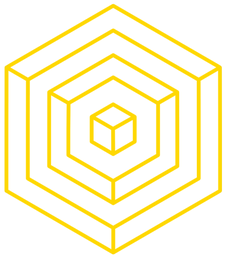
```
set proposal value:
  if Acceptors had accept proposal before:
    proposal value := value of proposal with the highest proposal number
  else:
    proposal value := any value determined by Proposer
```

Proposer then sends proposal value to Quorum of Acceptors

EXTRA: BASIC PAXOS

PHASE 2A: ACCEPT REQUEST





EXTRA: BASIC PAXOS

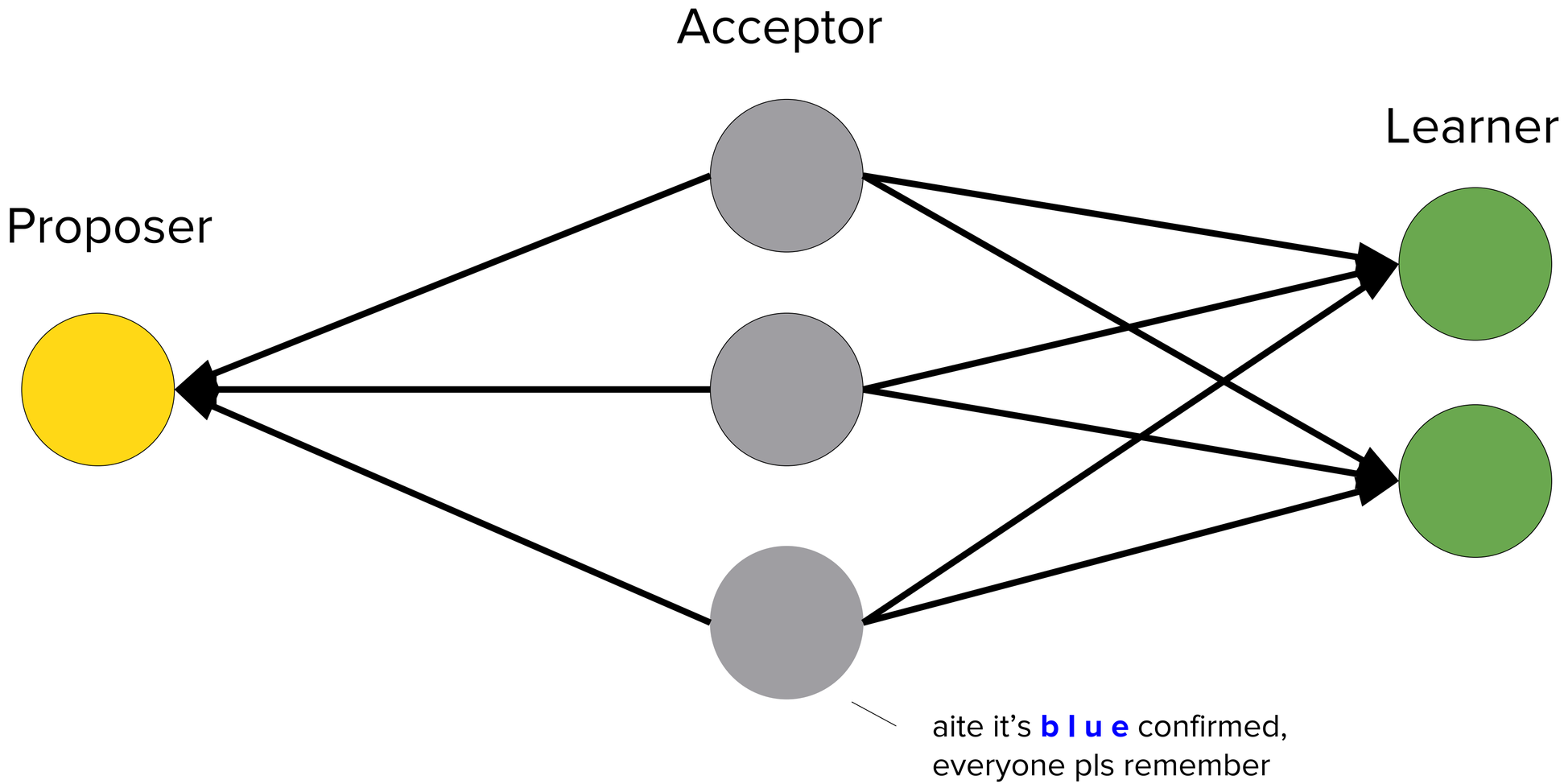
PHASE 2B: ACCEPTED

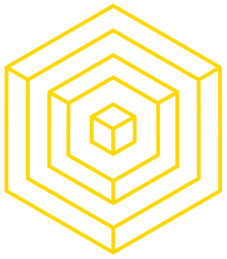
Acceptor accepts if and only if it has not already promised to only consider proposals greater than N

Register corresponding proposal value, and send message to Proposer and every Learner

EXTRA: BASIC PAXOS

PHASE 1B: PROMISE



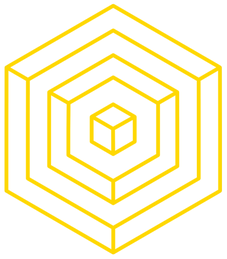


EXTRA: PAXOS

ASSUMPTIONS

- Nodes do not try to subvert the protocol
- Only works for fail-stop (no Byzantine failures)
- Messages delivered without corruption

- Failed nodes at any point won’t affect the entire network



EXTRA: PAXOS

SAFETY AND LIVENESS PROPERTIES

- Learners can only learn values that have been proposed
- If there’s a proposal, a Learner will eventually learn some value
- At most one value can be learned (different learners cannot learn different values)