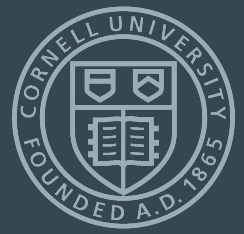
The Next-Gen Blockchain Infrastructure



Ted Yin // Cornell University & Ava Labs, Inc.
tederminant@gmail.com

The Infrastructure of Practical Blockchain Systems

Blockchain?

> What makes a system “blockchain” ?
>
> Unique challenges...

What is NOT a blockchain?

“Block” ?	We have batching...
“Chain” ?	Isn’t that linked list, or a log?
“Consensus” ?	We have Paxos/Raft/...
“Smart Contracts” ?	DSL like Lua/Lisp/Google Apps Script/...
“P2P” ?	Well...BitTorrent, Tor, DHTs...

What makes a blockchain “blockchain” ?

“Decentralization” vs. “Distributed”

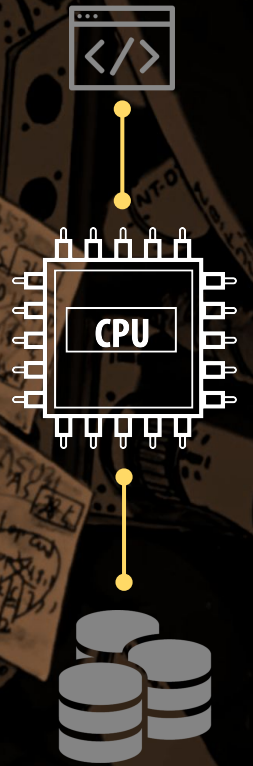
Uncooperative participants (“adversaries”)

→ “Byzantine” behavior

Decentralized Trust!

Part 0. Infrastructure

- Usability: Transactions, smart contracts, ... (program)
- Feasibility: Byzantine Fault Tolerant consensus (processor)
- Durability: Local storage system (memory)

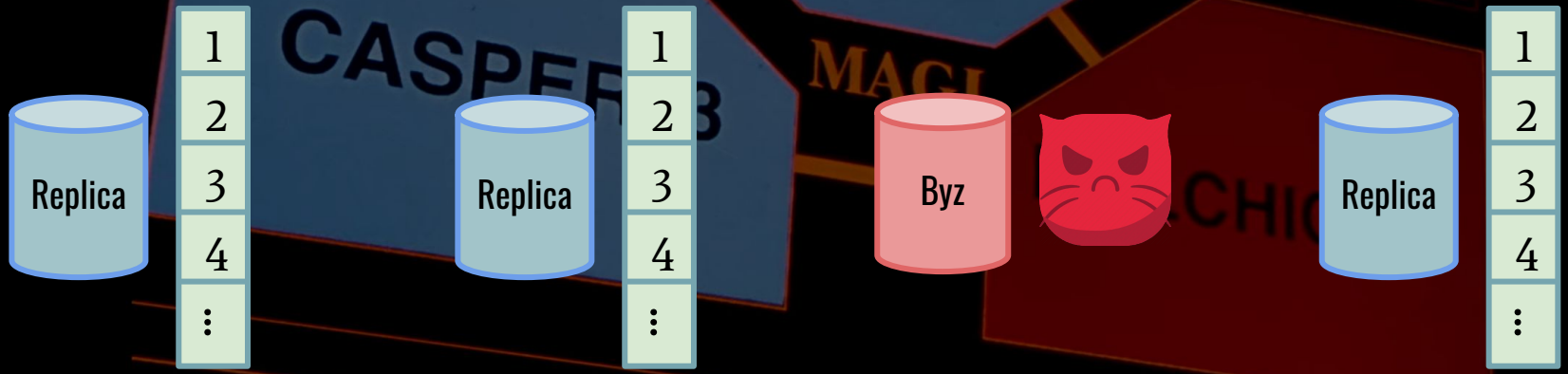


BFT Consensus

- *(Agreement)* All correct nodes must agree on the same value.
  “Safety”
- *(Termination)* All nodes must eventually decide on an output value.
  “Liveness”
- *(Validity)* If all correct nodes receive the same input value, then they must all output that value. → “Non-triviality”

BFT Consensus → BFT State Machine Replication

- f out of all n nodes could exhibit arbitrarily faulty behavior
- *(Replica Coordination)* other n-f correct replicas process the same sequence of requests
  - Two replicated sequence $s_1 \subseteq s_2 \vee s_2 \subseteq s_1$



Model & Known Solutions

- Impossibility (FLP '83)

"In this paper, it is shown that every protocol for this problem has the possibility of non-termination, even with only one faulty process."

Model & Known Solutions

- Termination → “Probability of 1”

  → Asynchronous model (Ben-Or ’83)

- Alway safe no matter what, and terminate when network is synchronized

  → Partially synchronous model (DLS ’88)

- Use synchronous assumption

  → Synchronous model (LSP ’82)

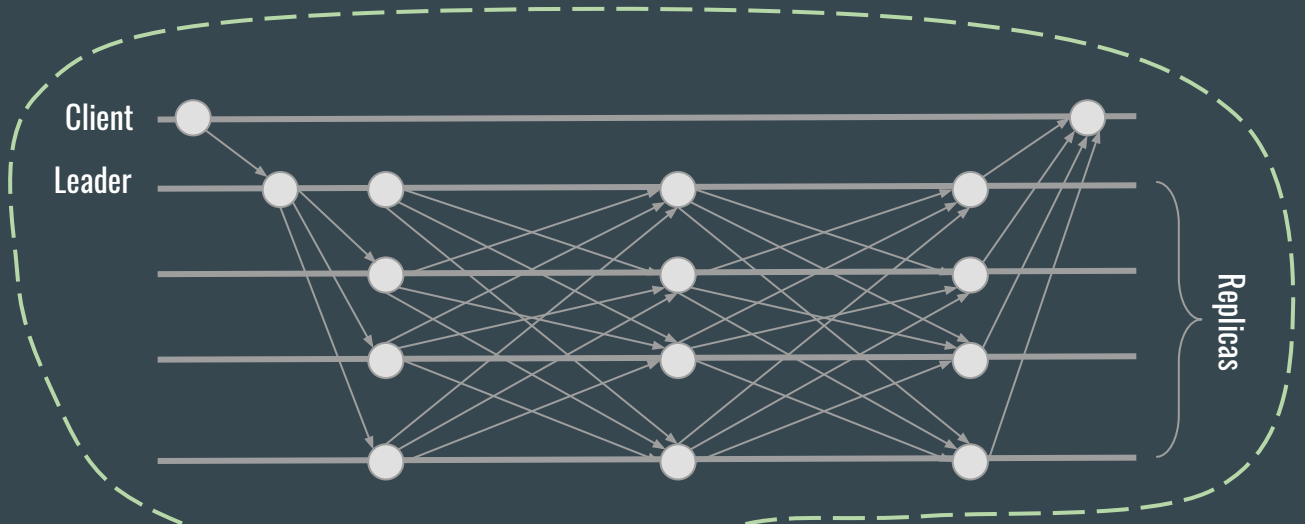
Model & Known Solutions

- Asynchronous model
  - Randomization to hope for a skew
  - Example: Ben-Or ('83): exponential termination time; HoneyBadgerBFT ('16): $\Omega(n^3)$
- Partially synchronous model
  - Leader-based
  - Example: PBFT ('99): $O(n^2)$ on a "good" day
- Synchronous model
  - Can tolerate up to n/2 faulty nodes
  - Example: XFT (OSDI '16) is fast when f = O(1) for tiny n, very inefficient in general
  - Nakamoto Consensus (synchronous)

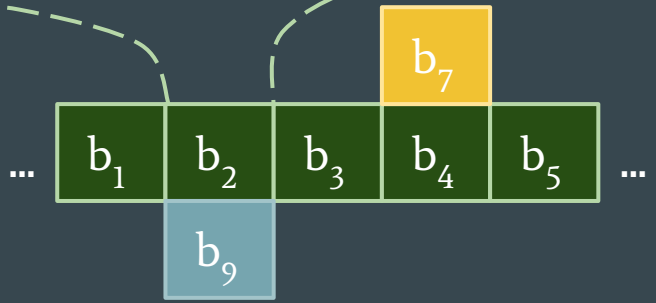
Theory vs Practice

- Asynchronous model
  - Without a leader — everyone proposes
  - “Expected convergence” — non-deterministic number of rounds
- Partially synchronous model (practical):
  - $O(n^2)$ on a good day is still far from the benign counterpart
  - Complicated and subtle operational logic
  - The leader is the bottleneck
- Synchronous model:
  - Lock-step execution — throughput bottleneck
  - Strong assumption of delivery timeout — dilemma in choosing Δ
  - Nakamoto Consensus: PoW is prohibitively expensive

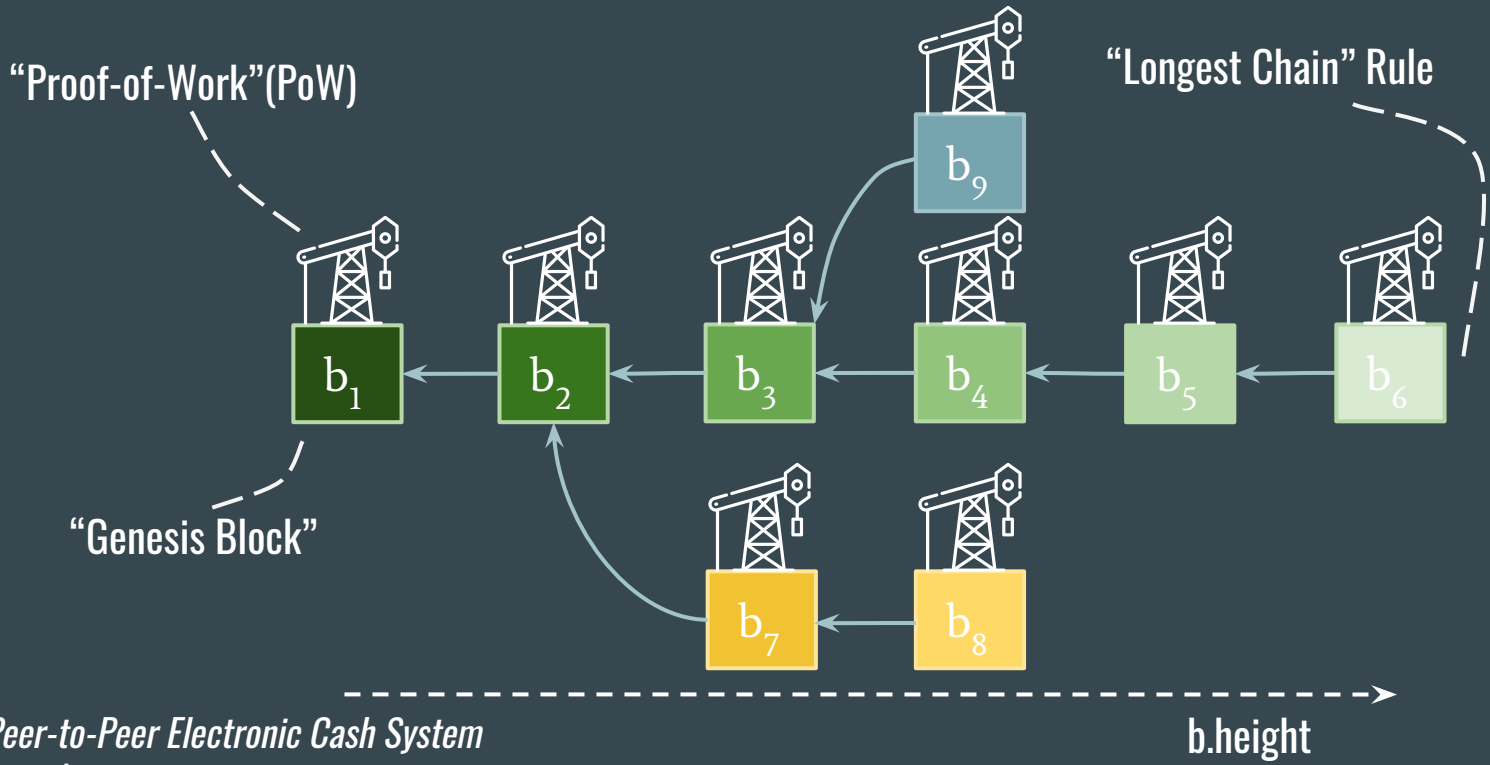
Paradigm: Quorum/Vote-Based



Example: PBFT



Paradigm: Nakamoto Consensus

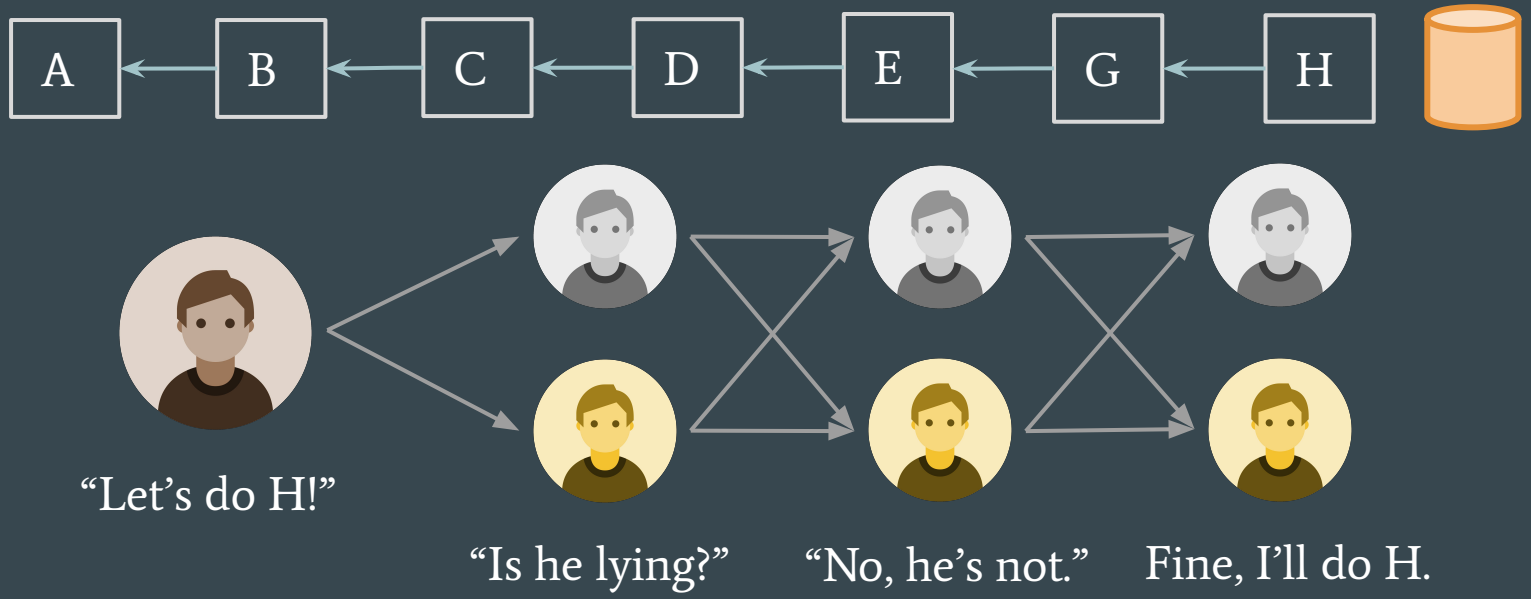


Bitcoin: A Peer-to-Peer Electronic Cash System
Satoshi Nakamoto

State Machine Replication

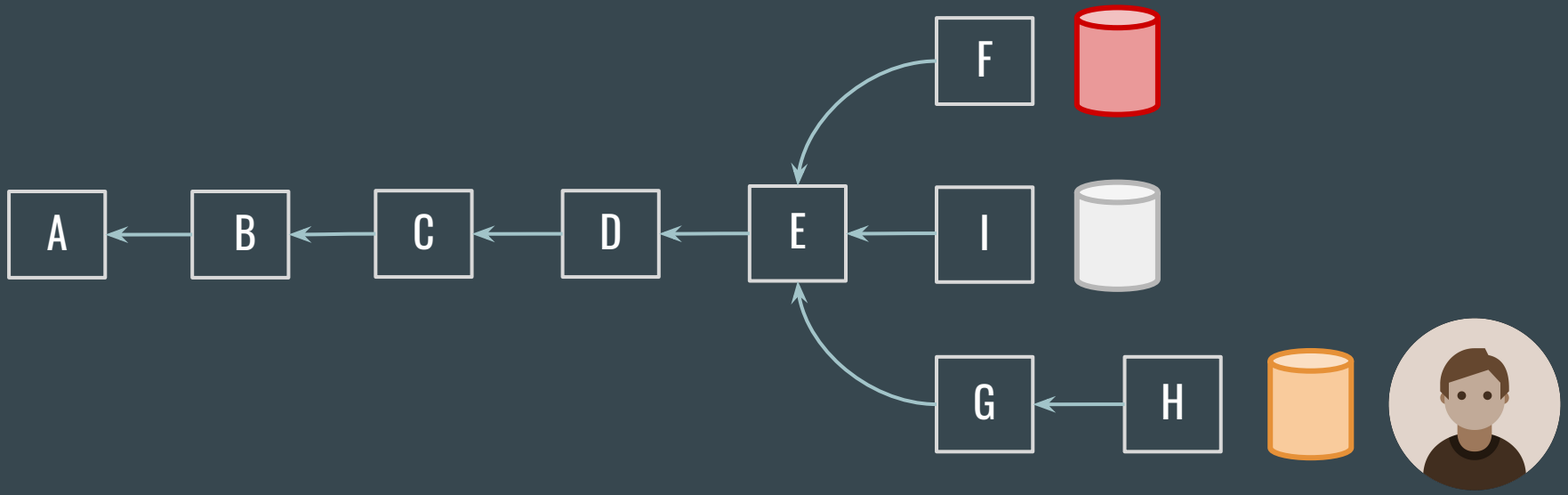
Practical Byzantine Fault Tolerance

Castro and Liskov 1999

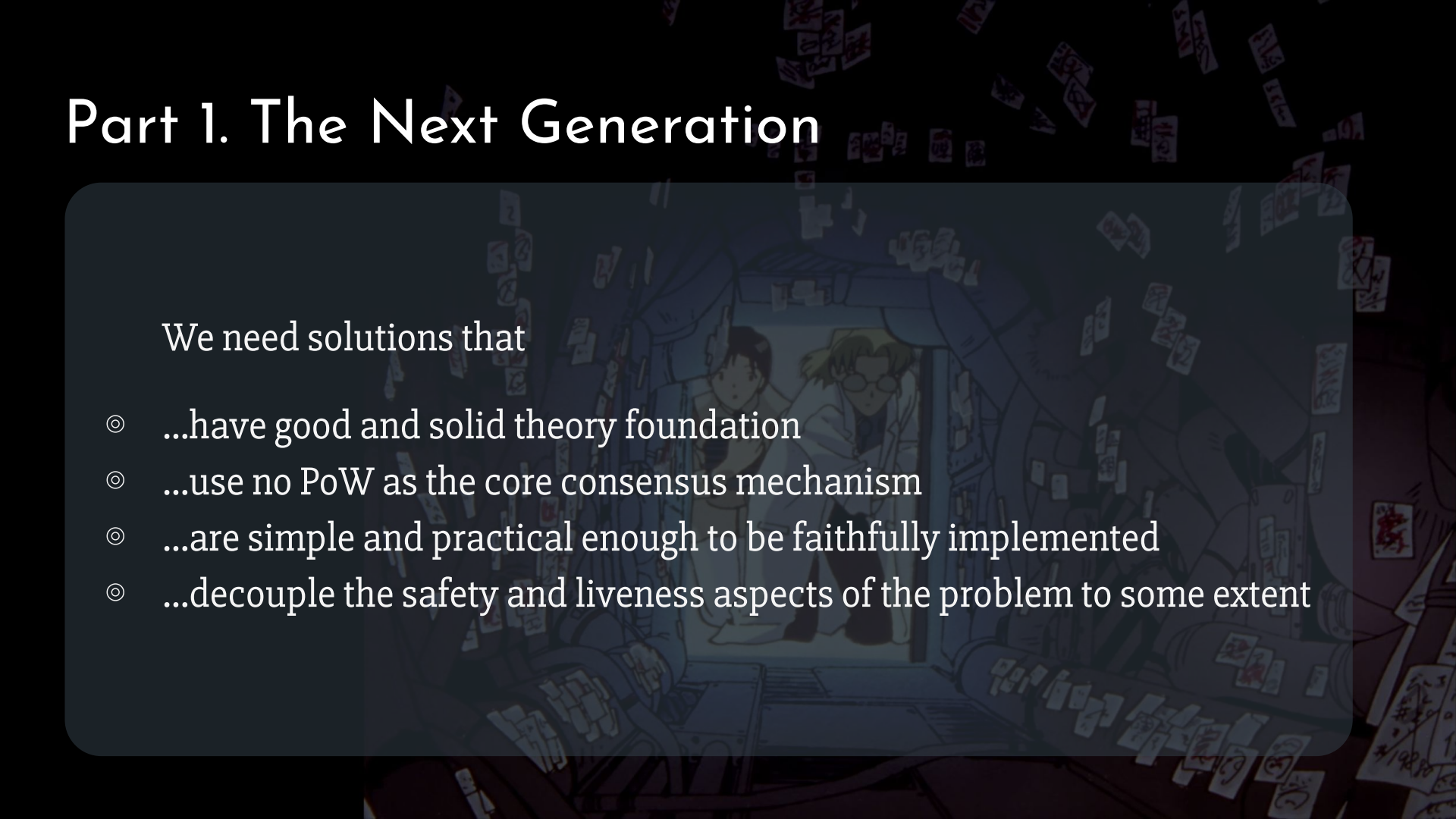


Bitcoin: A Peer-to-Peer Electronic Cash System

Nakamoto 2008



Part 1. The Next Generation



We need solutions that

- ...have good and solid theory foundation
- ...use no PoW as the core consensus mechanism
- ...are simple and practical enough to be faithfully implemented
- ...decouple the safety and liveness aspects of the problem to some extent

The Next-Gen BFT Protocols

- “HotStuff” Project:
  - standard *partial synchrony* model
  - the new king of the old paradigm realm
- “Snow/Avalanche” Project:
  - looks more like randomized *async protocols*
  - new realm: random subsampling for the first time

The Next-Gen: HotStuff

- A Paradigm distilled from quorum-based consensus — “Quorum Certificate”
- Revisit the “old wine”: DLS (and Tendermint/Casper) and locking mechanism
- “Blockchain-style”, no special treatment of view change

libra

facebook

Linearity

The total number of exchanged authenticators is O(n).

"The communication cost is linear"

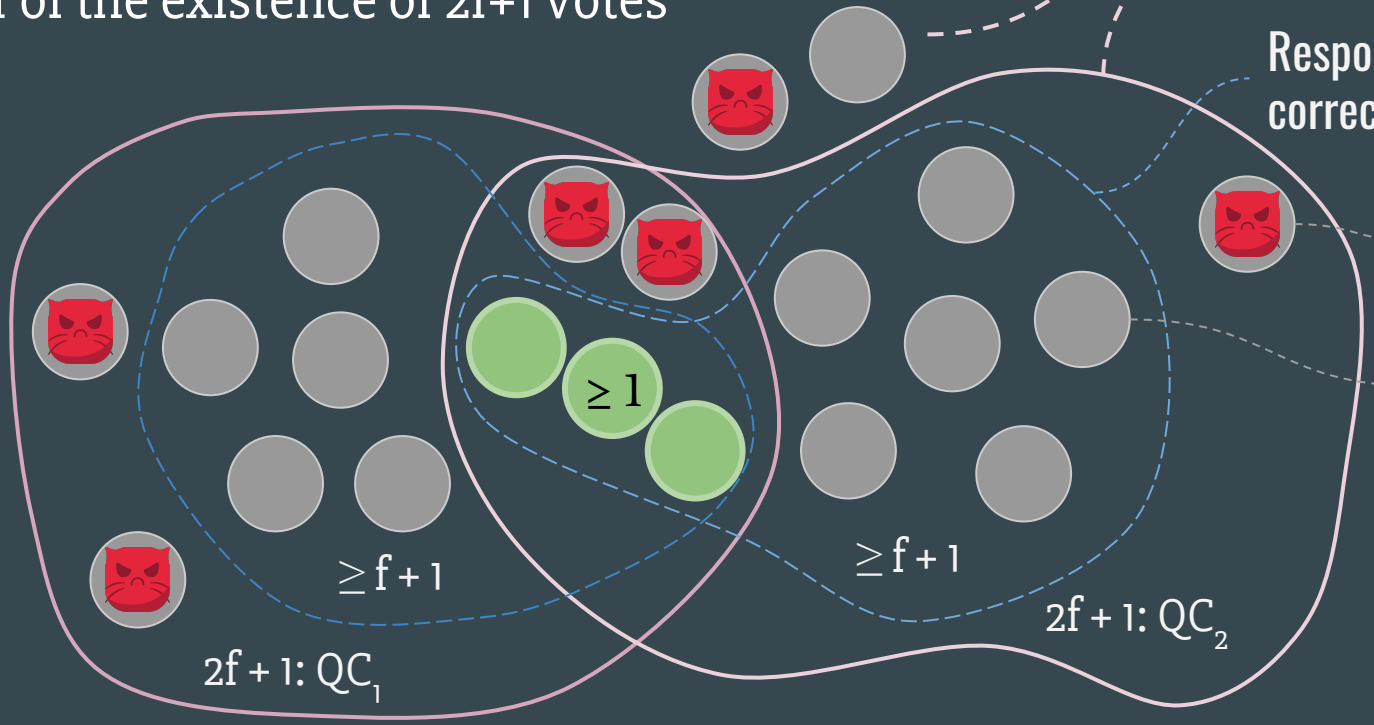
(Optimistic) Responsiveness

After GST, any correct leader, once designated, needs to wait just for the first *(n–f)* responses to guarantee that it can create a proposal that will make progress.

"As fast as the network propagates, on a good day"

Quorum Certificate (QC)

Proof of the existence of 2f+1 votes

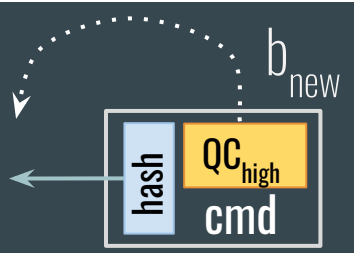


HotStuff: Data Structure

Messages

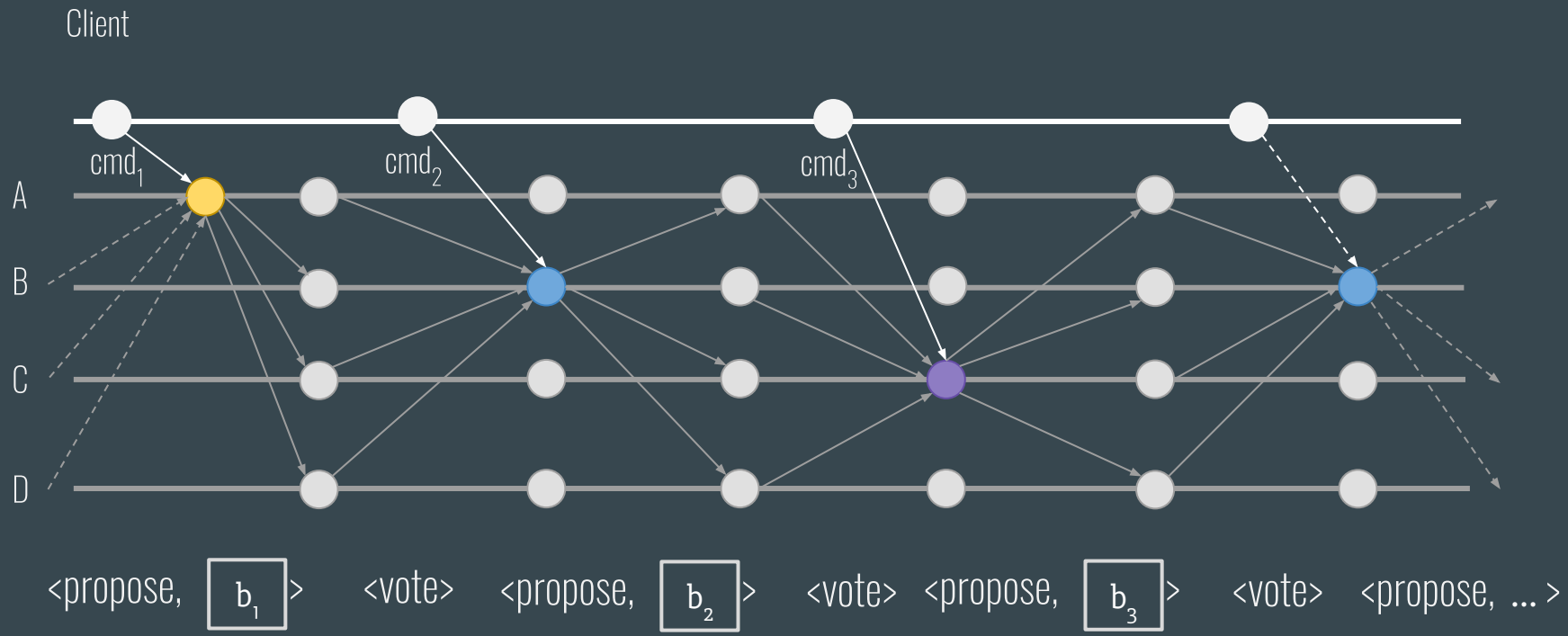
<propose, b_{new}>

<vote, <u, b_{new}>$_{signed\ by\ u}$>



- Leader broadcasts the propose message carrying block b_{new}
- Voters give back their opinions to the next leader via votes
- Only one type of messages for voting/view change, etc.

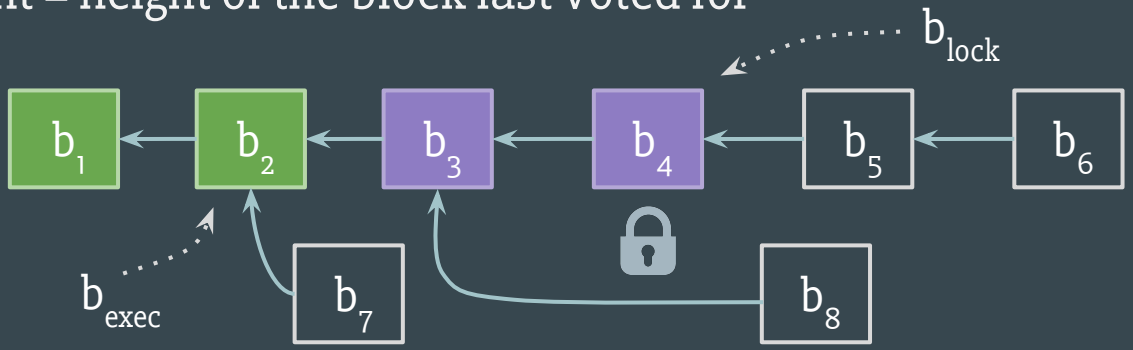
HotStuff: Communication Pattern



HotStuff: The Protocol

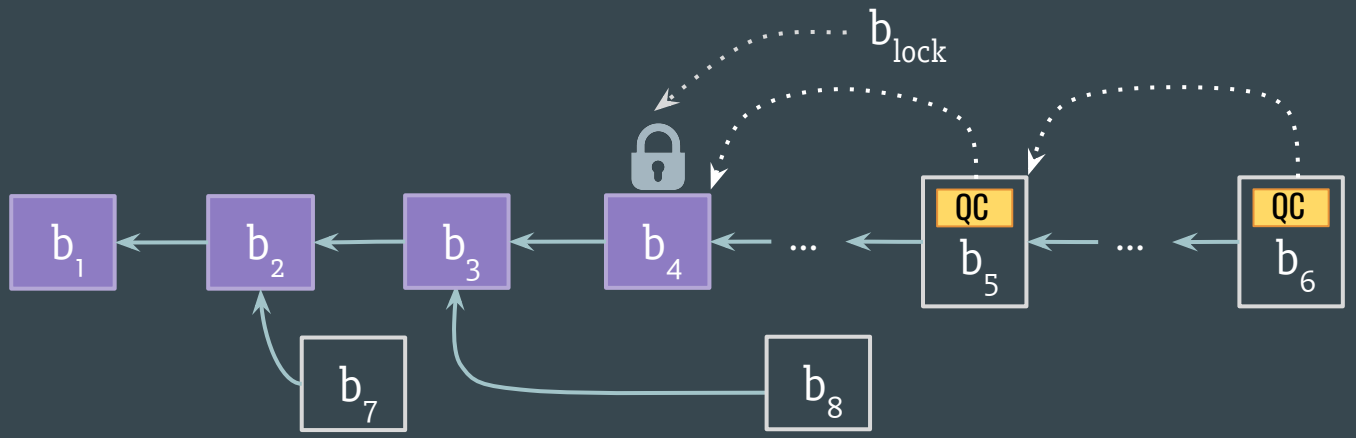
Protocol State Variables

- b_{lock} = block leading the preferred branch
- b_{exec} = last committed block
- vheight = height of the block last voted for



“Longest” Chain Rule: Branch Preference

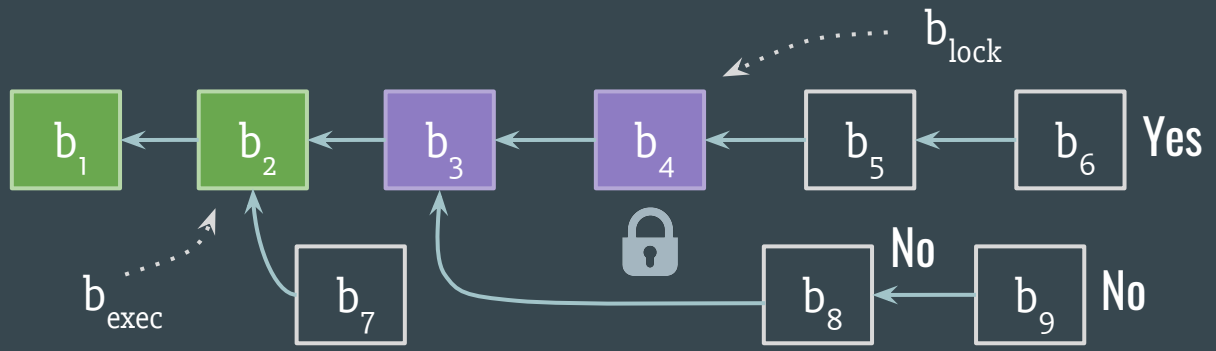
- b_{lock}: the “locked block” that leads the preferred branch
- Locking mechanism: a replica only votes for the block on the preferred branch, unless...



HotStuff: Voting

How to Vote? (Safety Rule)

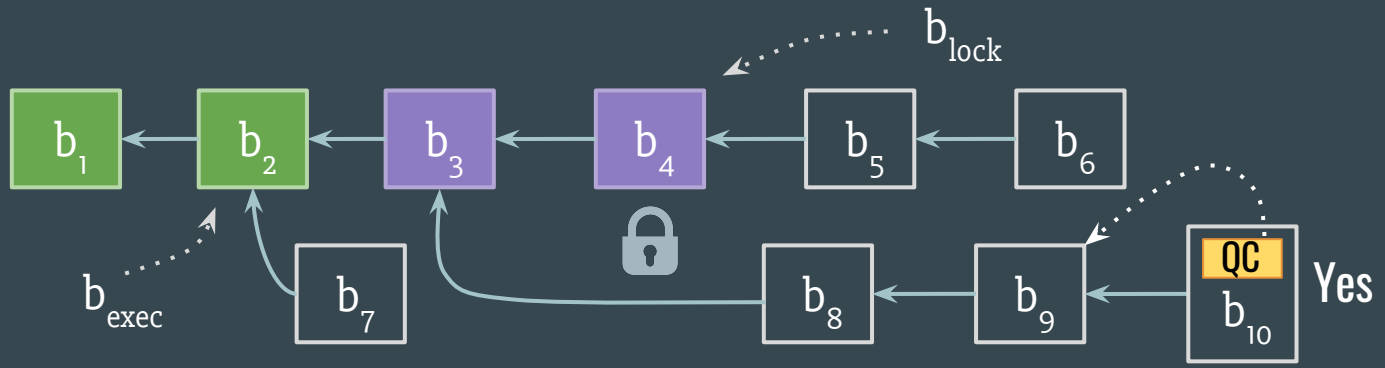
- Only vote for b_{new} if the following constraints hold:
  - b_{new}.height > vheight
  - (b_{new} is on the same branch as b_{lock}) or (b_{new} .justify .node.height > b_{lock}.height)



HotStuff: Voting

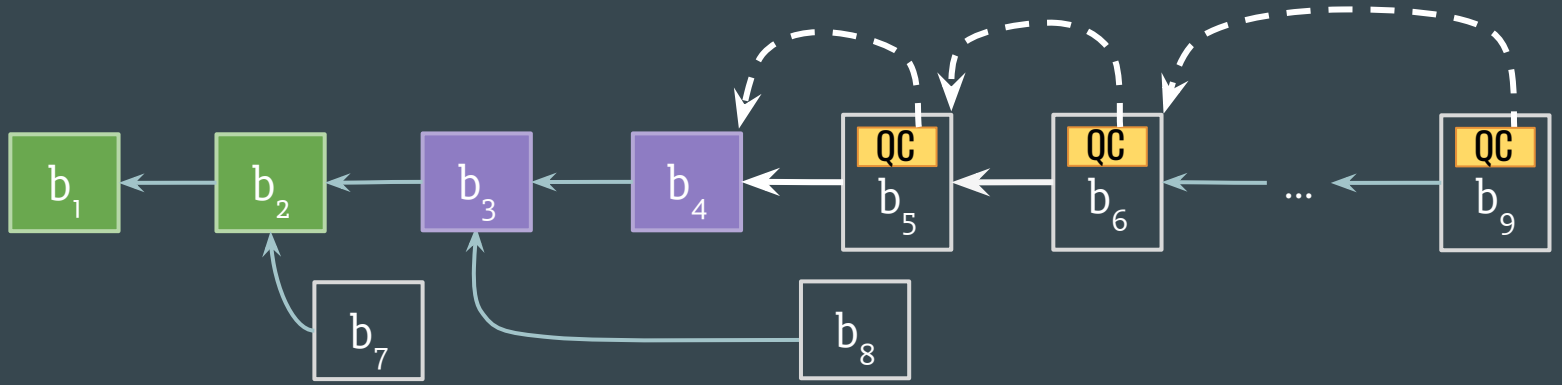
How to Vote? (Liveness Rule)

- Only vote for b_{new} if the following constraints hold:
  - b_{new}.height > vheight
  - (b_{new} is on the same branch as b_{lock}) or (b_{new} .justify .node.height > b_{lock}.height)



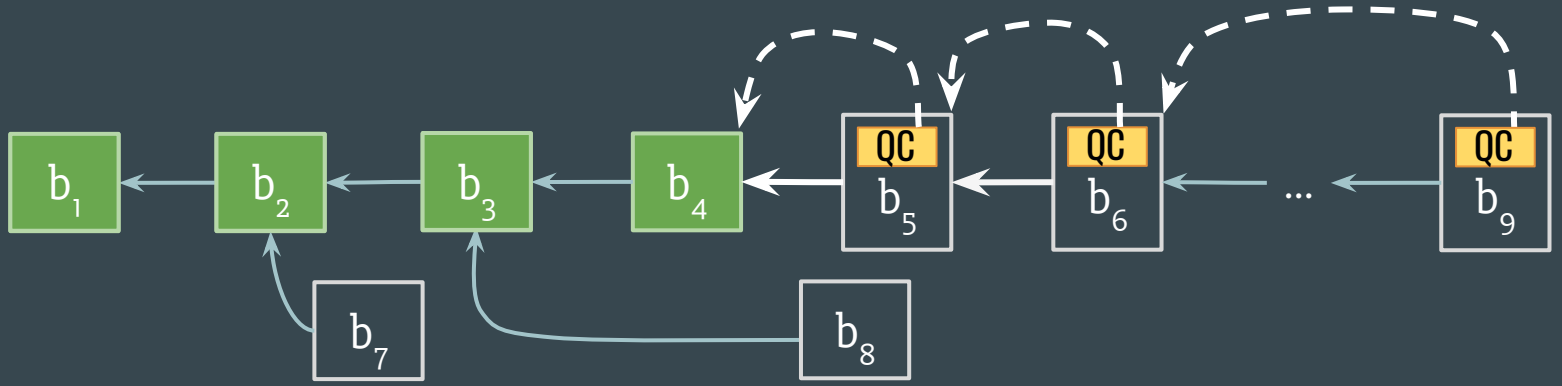
HotStuff: Decision

When to Commit?

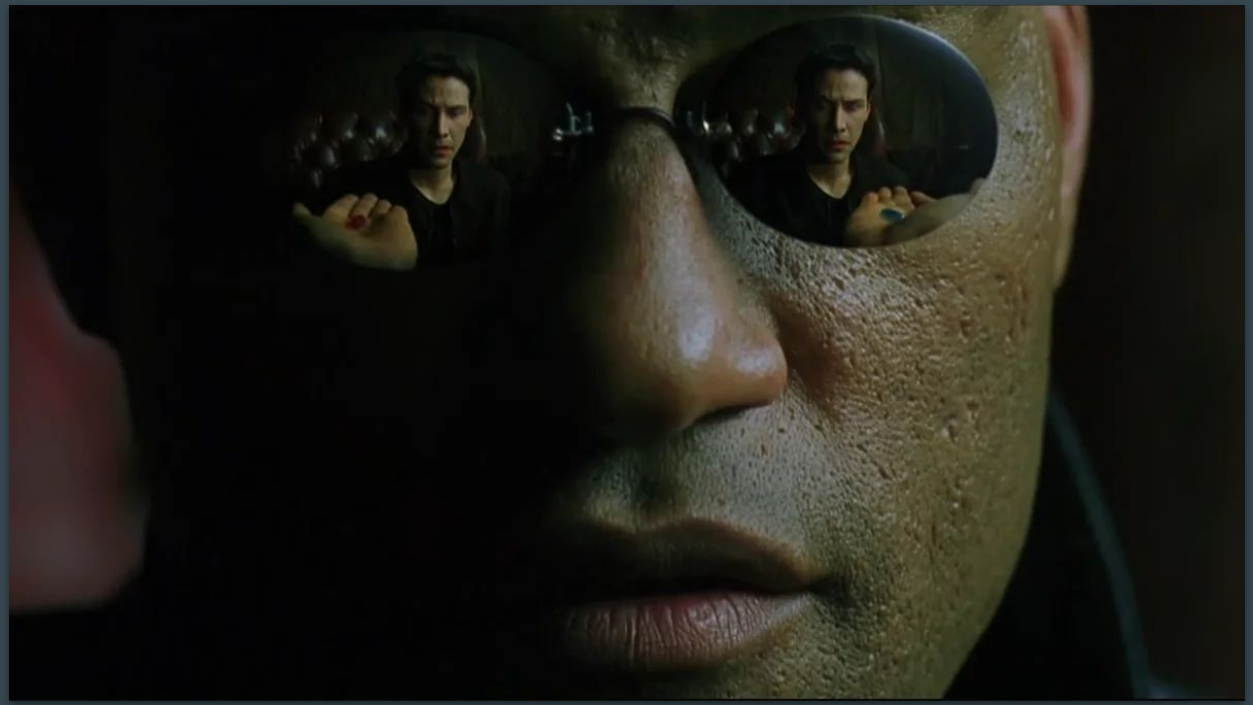


HotStuff: Decision

When to Commit?



Is it the end? Take the red pill...

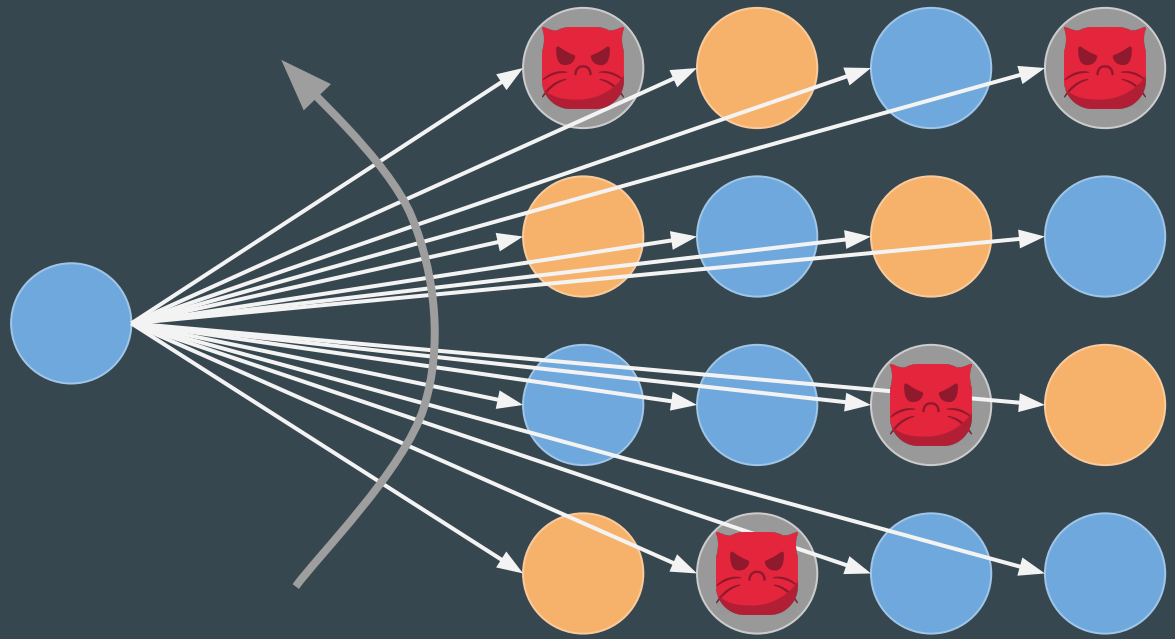


The Next-Gen: Snow/Avalanche Project

- New paradigm: P2P gossip can be directly used as consensus
- Model is “in-between” the conventional sync. and async.
- Uses a unique stochastic process rather than quorum reasoning
- Loose membership
- Weakened but practical (safety) guarantees in exchange for...
- ...significant better scalability

Full Broadcast to Partial Sampling.

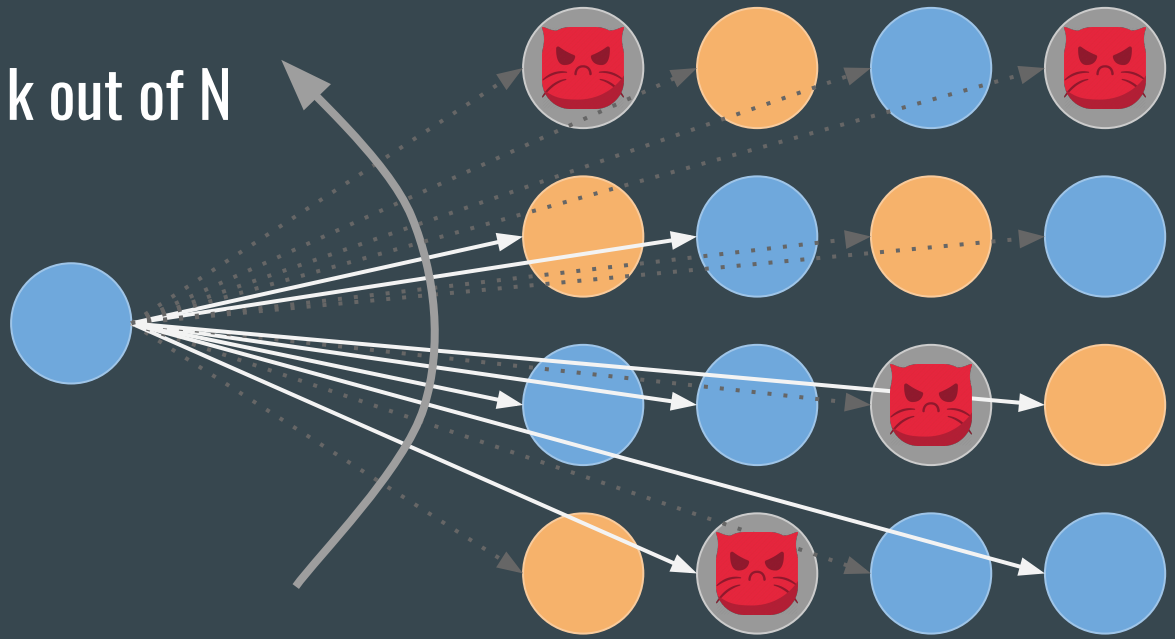
Query all

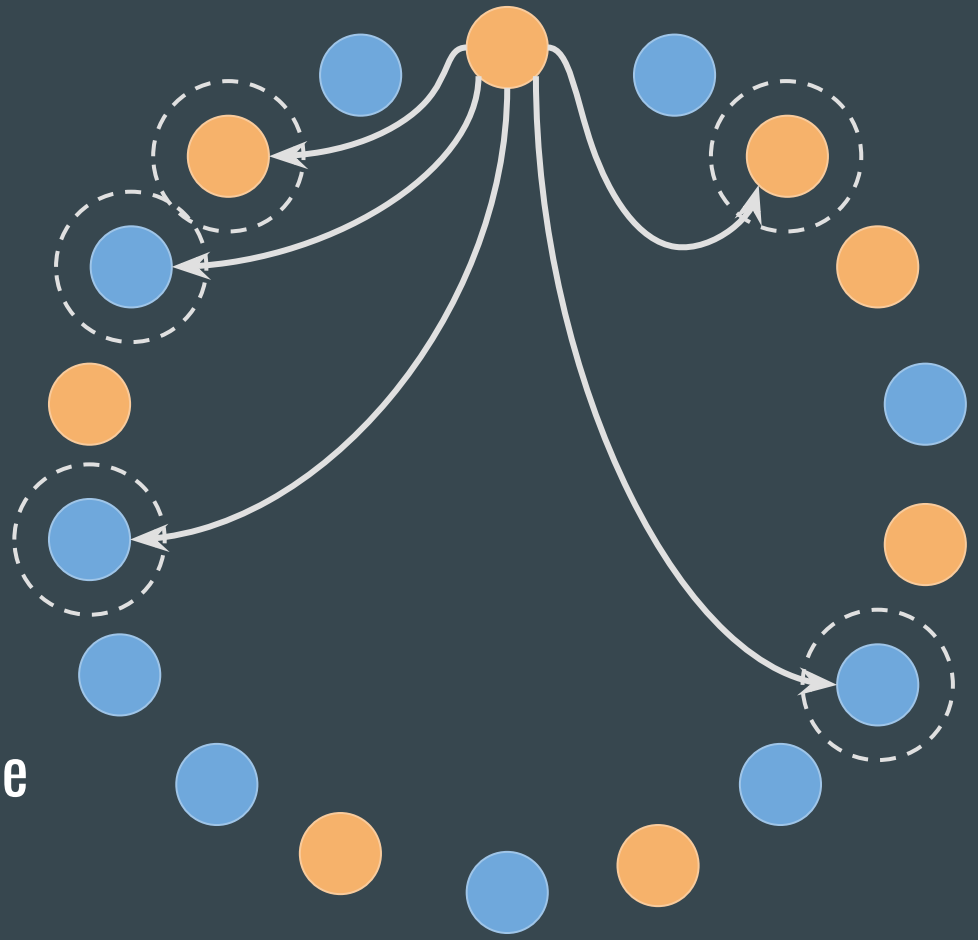


Full Broadcast to Partial Sampling.

~~Query all~~

Only Sample k out of N





Alice asks
5 other people
randomly

Alice: “Blue is the majority answer!”

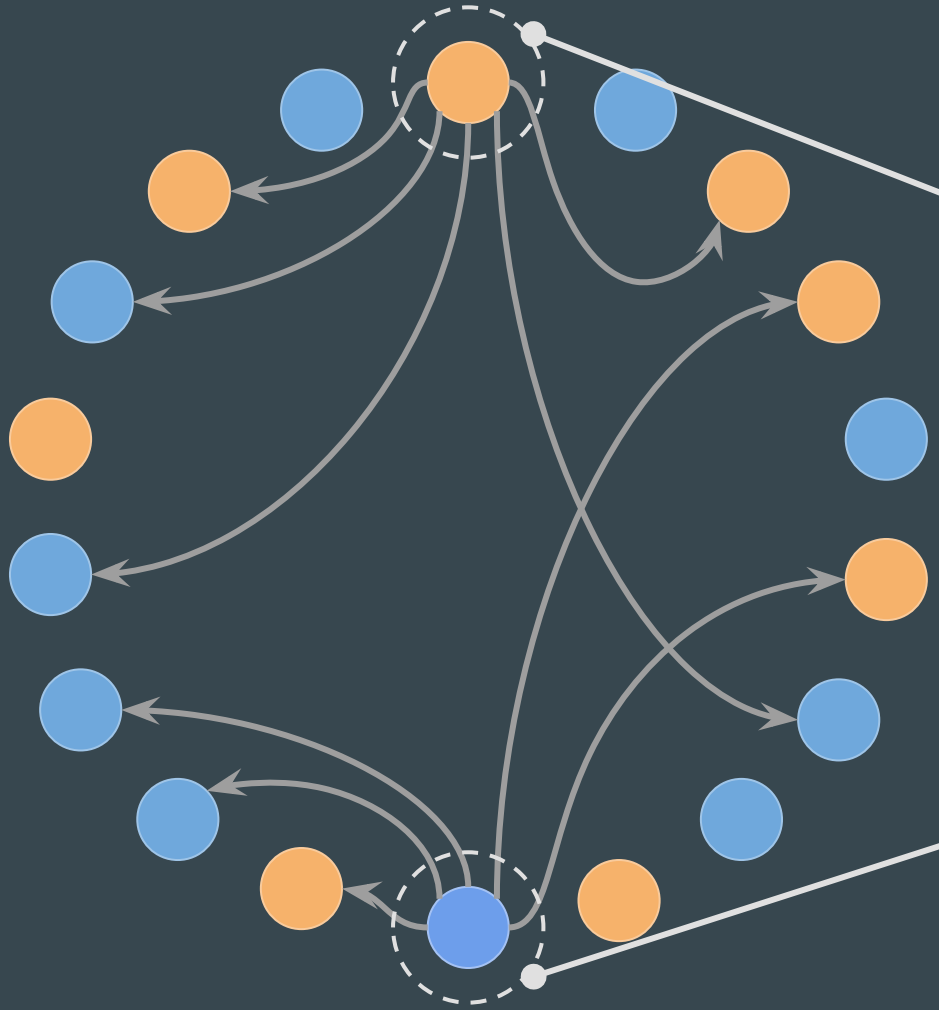
Bob asks
5 other people
randomly

Alice: “Blue is
the majority
answer!”

Alice asks
5 other people
randomly

Bob: “Yellow is
the majority
answer!”

Bob asks
5 other people
randomly

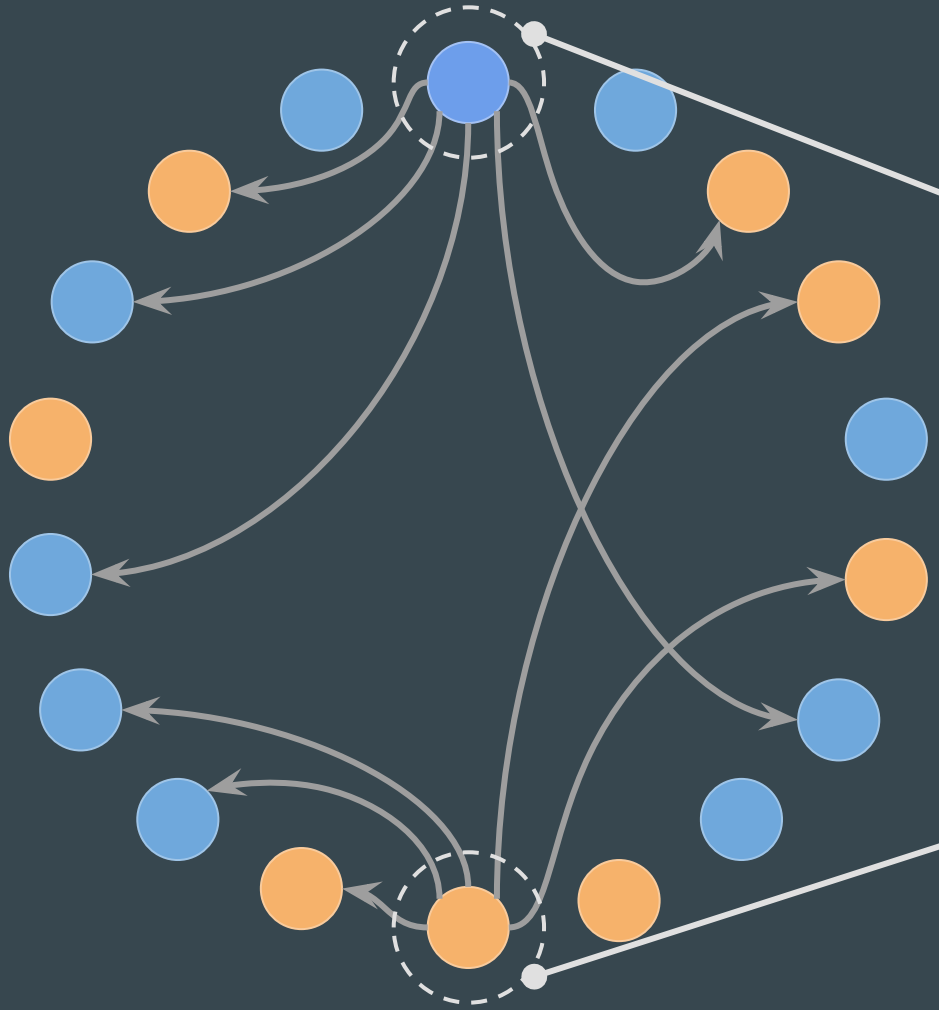


Alice: “Blue is the majority answer!”

Alice asks
5 other people
randomly

Bob: “Yellow is the majority answer!”

Bob asks
5 other people
randomly



Alice: “Blue is the majority answer!”

Alice asks
5 other people
randomly

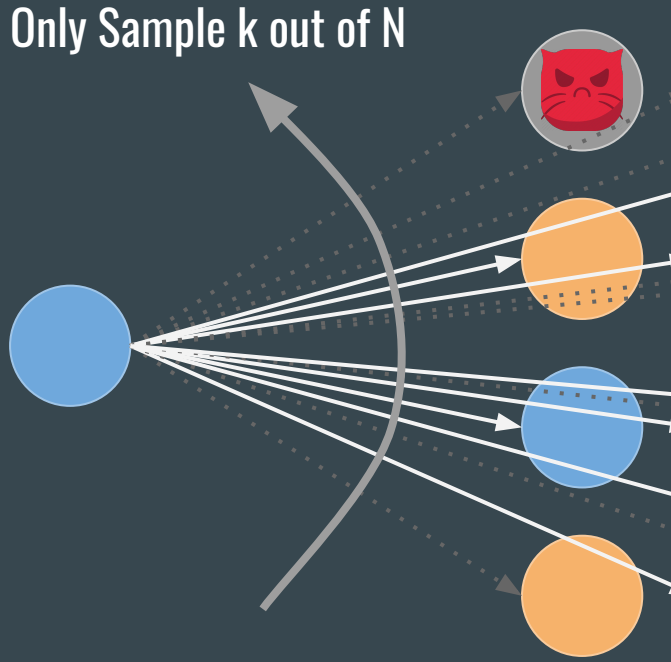
Bob: “Yellow is the majority answer!”

Snowflake: Epidemic Loop

1. Sample k peers uniformly at random
2. If αk agrees on a color c':
   a. If c' is the same as the current c:
      i. Increase the *counter*
   b. Else:
      i. c := c'
      ii. Reset the *counter*

(2*. Also reset the counter if no majority)

3. Go to line 1



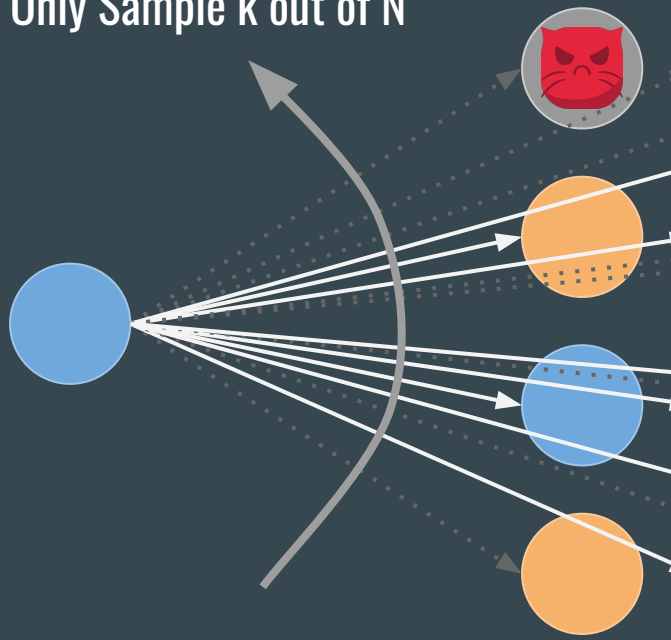
Snowball: Adding Confidence

1. Sample k peers uniformly at random
2. If αk agrees on a color c':
   a. Increase the *confidence* d[c'] for c'
   b. If d[c'] > d[c]
      i. c := c'
   c. If c' is not the last color gets αk
      i. Reset the counter
   d. Else:
      i. Increase the counter

(2*. Also reset the counter if no majority)

3. Go to line 1

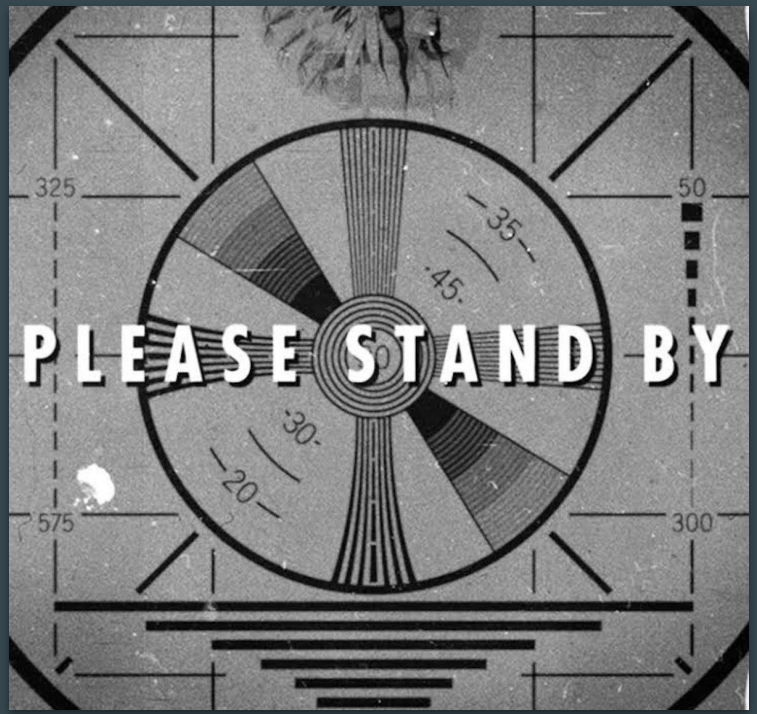
Only Sample k out of N



Converge in O(logN) time
Better complexity?

Snowball: Demo

https://tedyin.com/archive/snow-bft-demo



Part 2. Bridging The Gap.

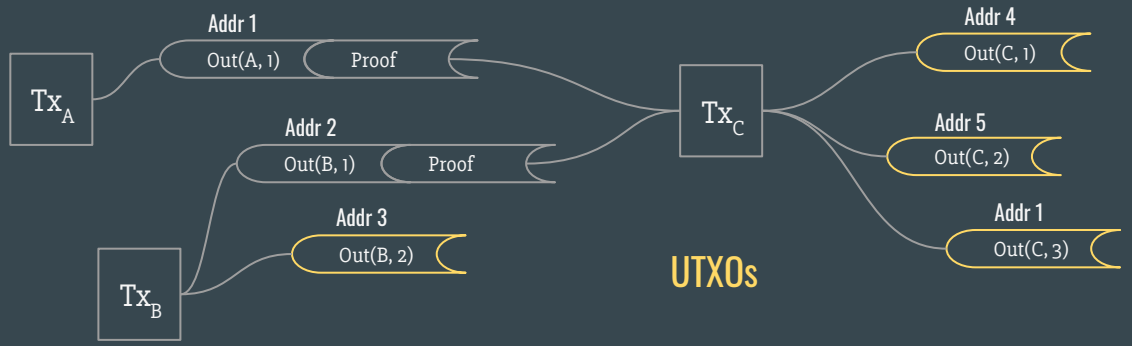
- Binary vs. multi-value consensus
  - “Common proposal problem” in leaderless solutions
  - “Log-encoding”: a feasible reduction
- Liveness, liveness, liveness!
  - “while true {}” is always safe
  - Safety and liveness entanglement
- Do we really want standard consensus?
  - Achieving a total order is fundamentally a bottleneck by itself
  - Byzantine nodes: Byzantine clients that don’t deserve guarantees

Example 1: HotStuff and Its Pacemaker

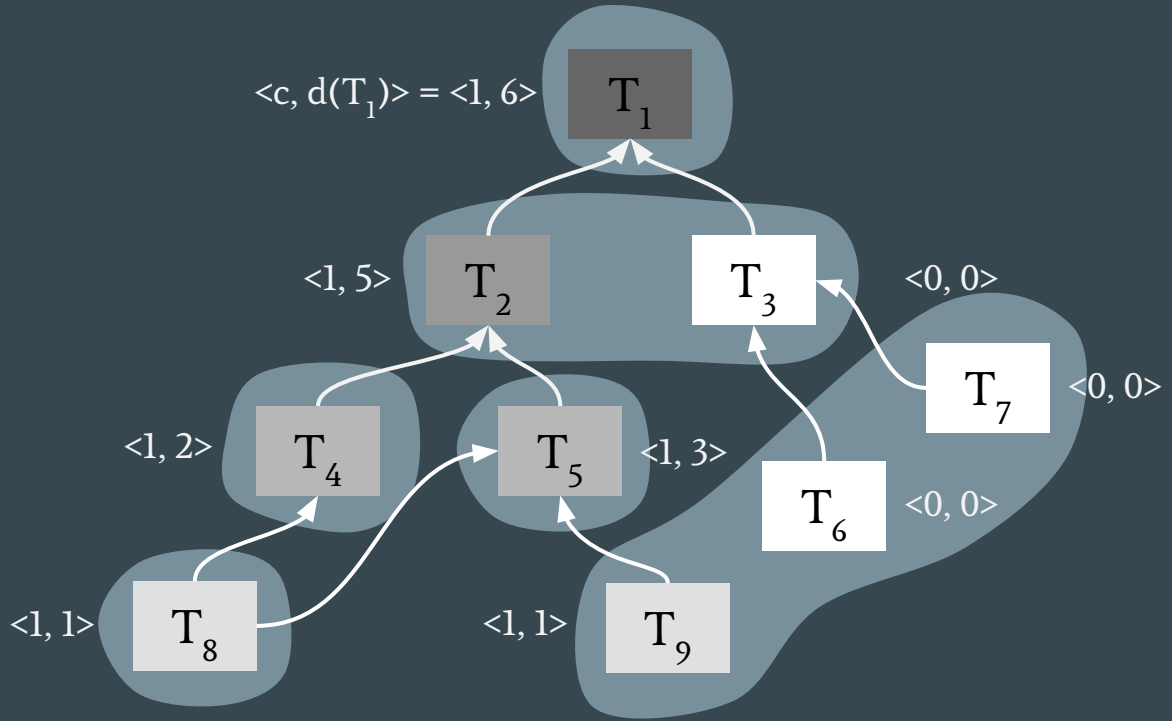
- The original PBFT code spans >10k C code
- Notoriously difficult to get it right: the “fast/slow” path, view change protocol
- Liveness guarantee creeps into the logic that guarantees safety
- HotStuff “Pacemaker”
  - Safety part < 200 C code
  - Customized liveness gadget
  - LibraBFT is an instantiation of such Pacemaker concept!

Example 2a: Snow → Avalanche

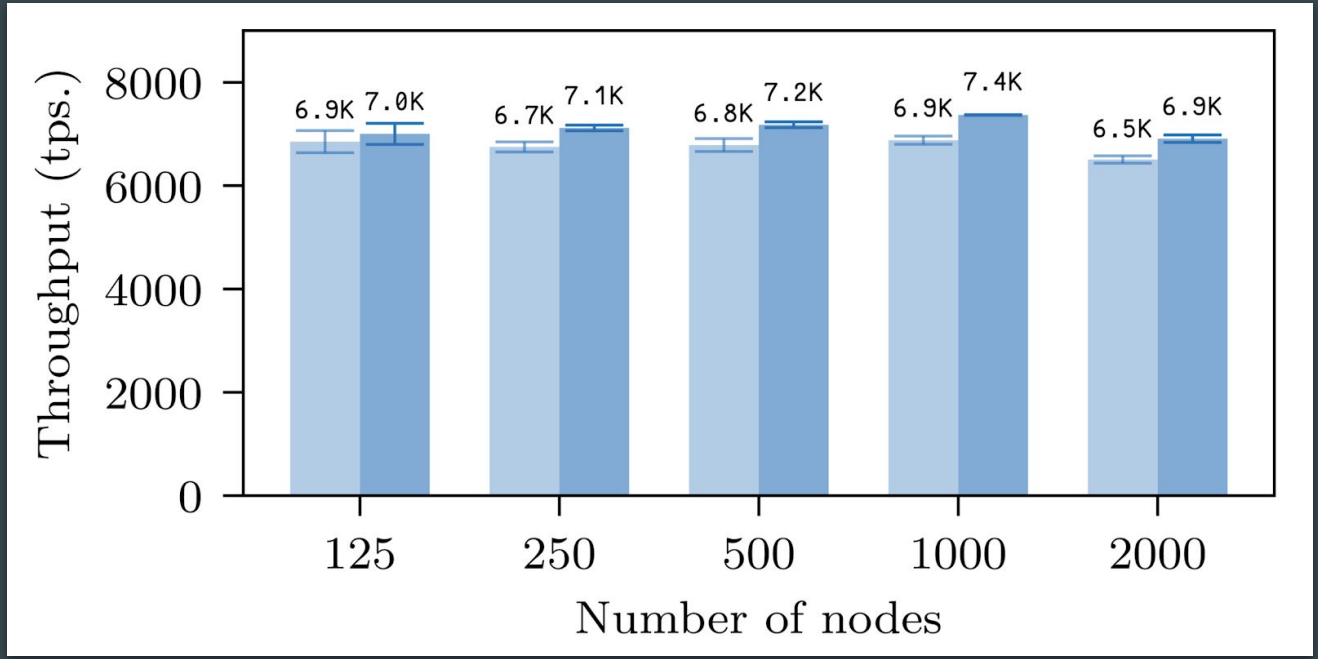
- Key observation: standard “consensus” is not a necessary requirement for making a payment system
- A payment system:
  - UTXO ensures a verifiable flow of spending
  - The only missing piece is “conflict resolution” — weaker than “consensus”
- Weakening the liveness:
  - Honest spenders → “triviality case” in consensus
  - Malicious spenders → may get stuck forever!



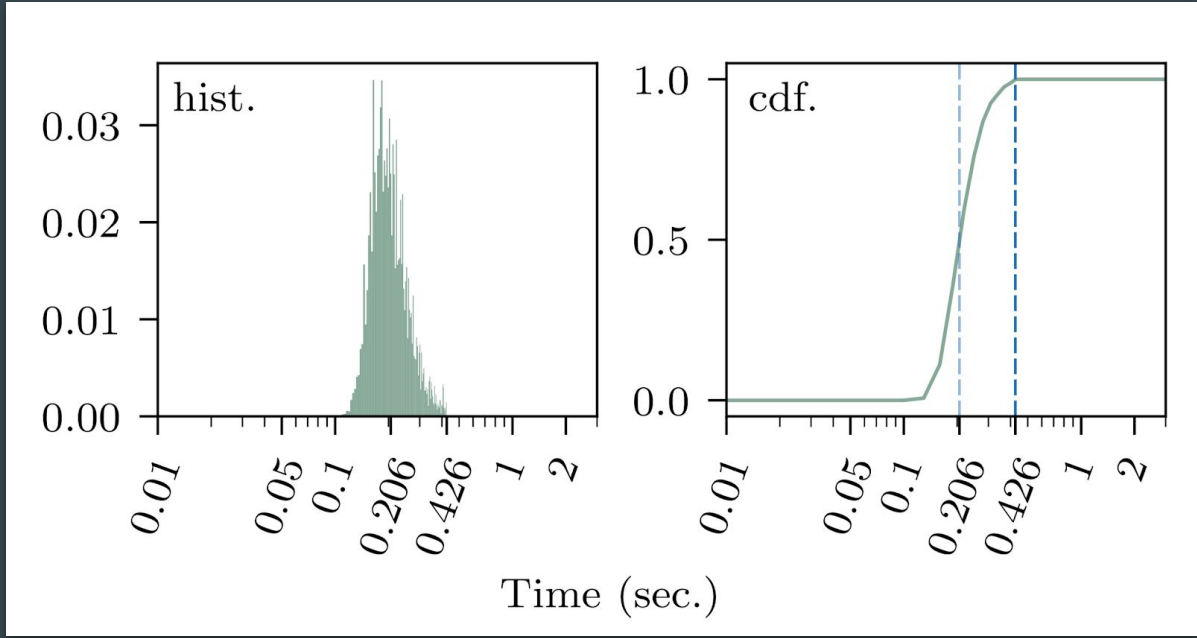
Avalanche: Cascading the Sampling Process



Avalanche Throughput.



Avalanche Latency.



Geo-replication

In an even more realistic setting:

- 2000 nodes in 20 cities across the globe
- All nodes directly participate in consensus
- Full signature verification

Our evaluation results:

- ~3400 tps
- ~1.35 sec

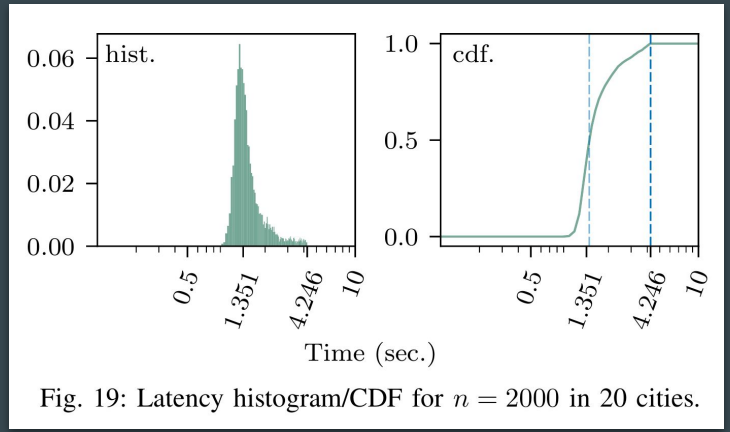
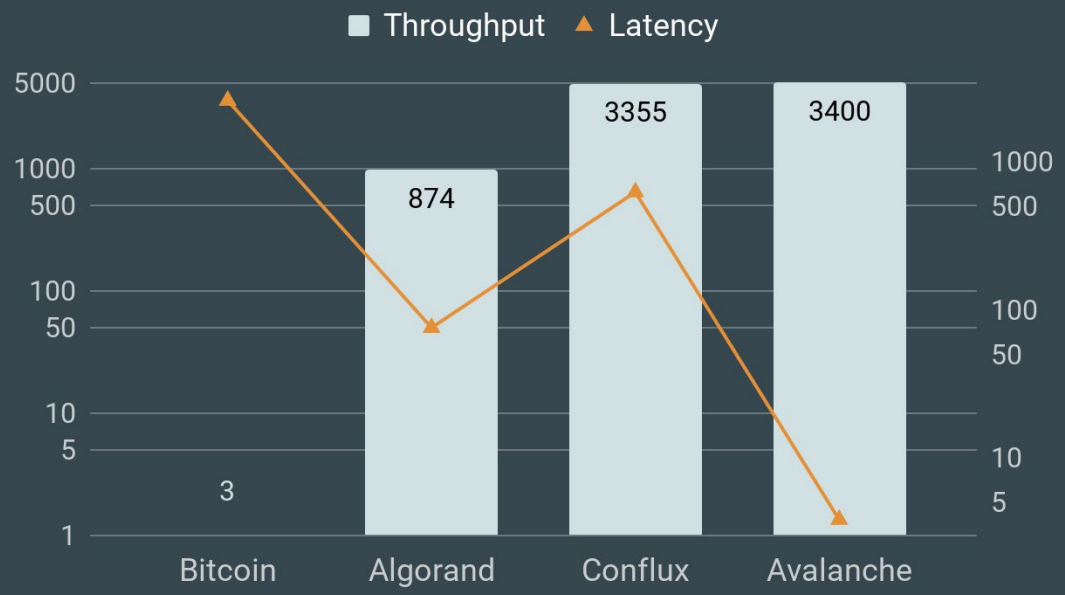


Fig. 19: Latency histogram/CDF for $n = 2000$ in 20 cities.

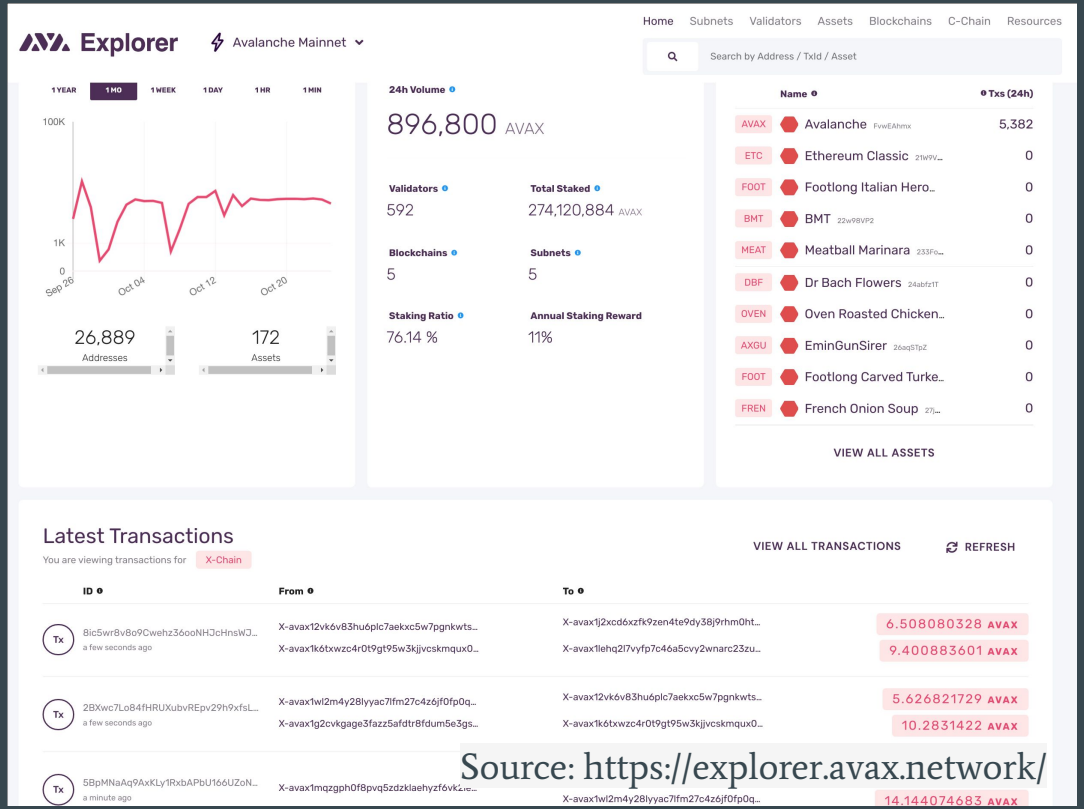
Evaluation: Comparison to Other Systems



Example 2b: Snow → Snowman

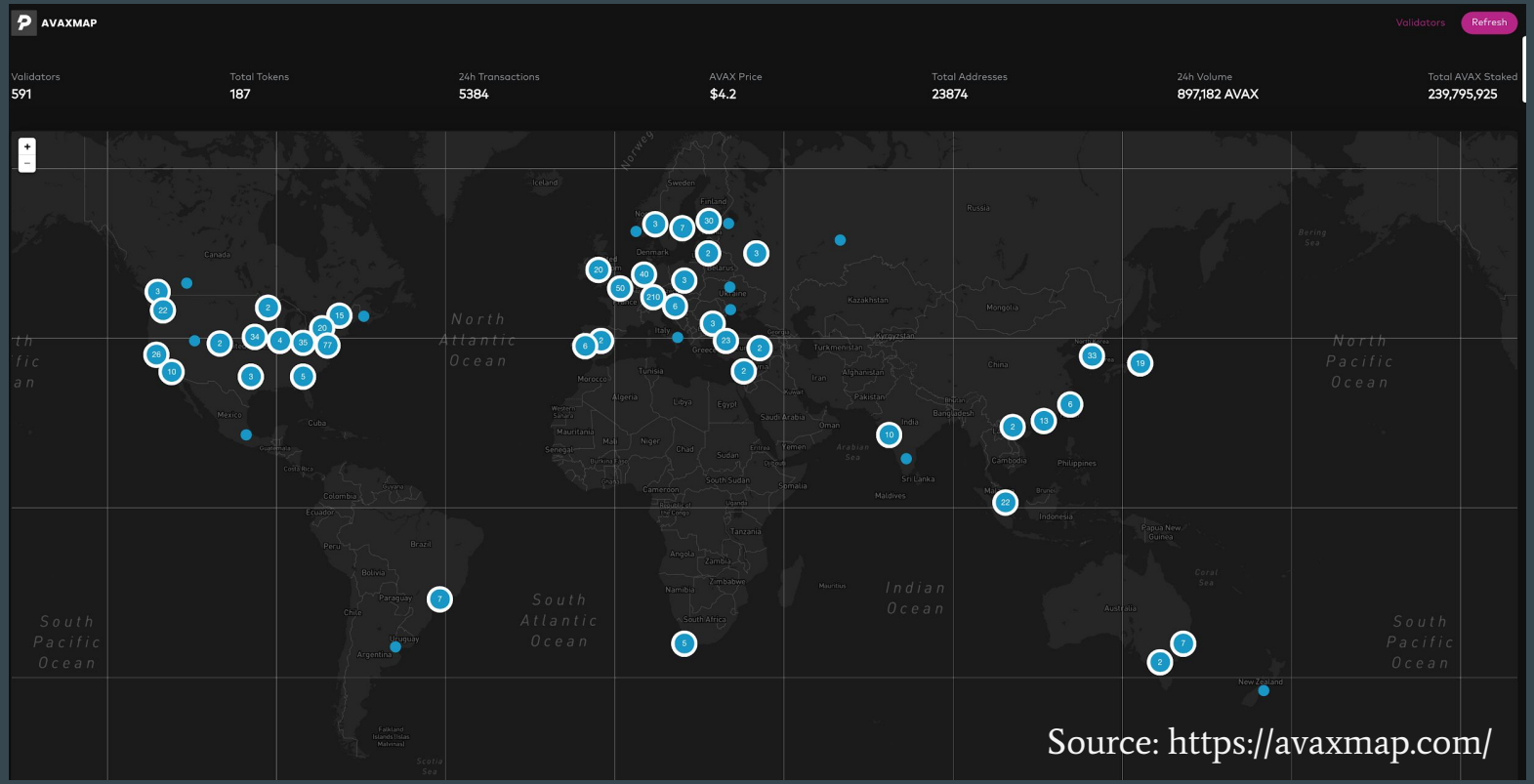
- What about computation in general?
  - Log-encoding from binary consensus to multi-value consensus
  - EVM support

Deployment



Source: https://explorer.avax.network/

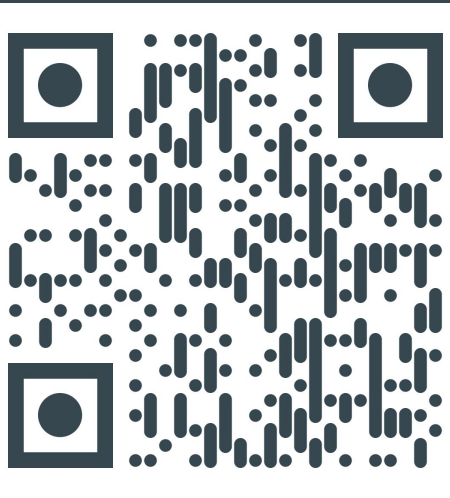
Deployment



Source: https://avaxmap.com/

Thanks for Listening!

←Avalanche Paper



HotStuff Paper→