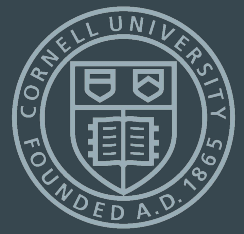
The Infrastructure for Practical Blockchain Systems



Ted Yin // Cornell University & Ava Labs, Inc.

Twitter @Tederminant

微博 @Determinanted

知乎 @determinant

The Infrastructure 区块链 of Practical Blockchain 基础架构 Systems

Blockchain?

What makes a system “blockchain” ?
究竟什么是区块链？

Unique challenges...
区块链独有的挑战.....

What is NOT a blockchain?

"Block" ?	We have batching...
"区块？"	系统领域早就有了Batching思想...
"Chain" ?	Isn't that linked list, or a log?
"链？"	那不就是链表或者记录吗？
"Consensus" ?	We have Paxos/Raft/...
"共识协议？"	Paxos/Raft早已经出现并被广泛使用...
"Smart Contracts" ?	DSL like Lua/Lisp/Google Apps Script/...
"智能合约"	领域特定语言/框架也不少...
"P2P" ?	Well...BitTorrent, Tor, DHTs...
"点对点（对等）协议？"	基于点对点协议更是数不胜数...

What makes a blockchain “blockchain” ?

“Decentralization” vs. “Distributed”
“去中心化” vs. “分布式”

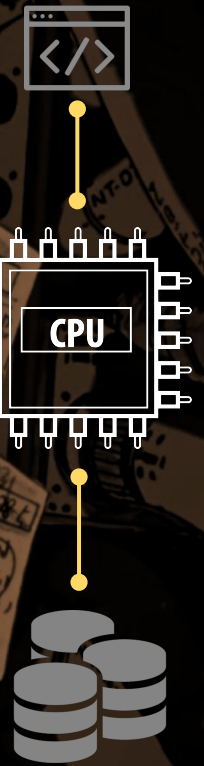
Uncooperative participants (“adversaries”)
心里打着“小九九”的参与者

→ “Byzantine” behavior
“拜占庭”行为

Decentralized Trust! “分布式”的信任！

Part 0. Infrastructure

- Usability: Transactions, smart contracts, ... (program)
  可用性：交易、智能合约（程序逻辑）
- Feasibility: Byzantine Fault Tolerant consensus (processor)
  可行性：拜占庭容错共识（执行中枢）
- Durability: Local storage system (memory)
  持久性：本地存储系统（状态记忆）

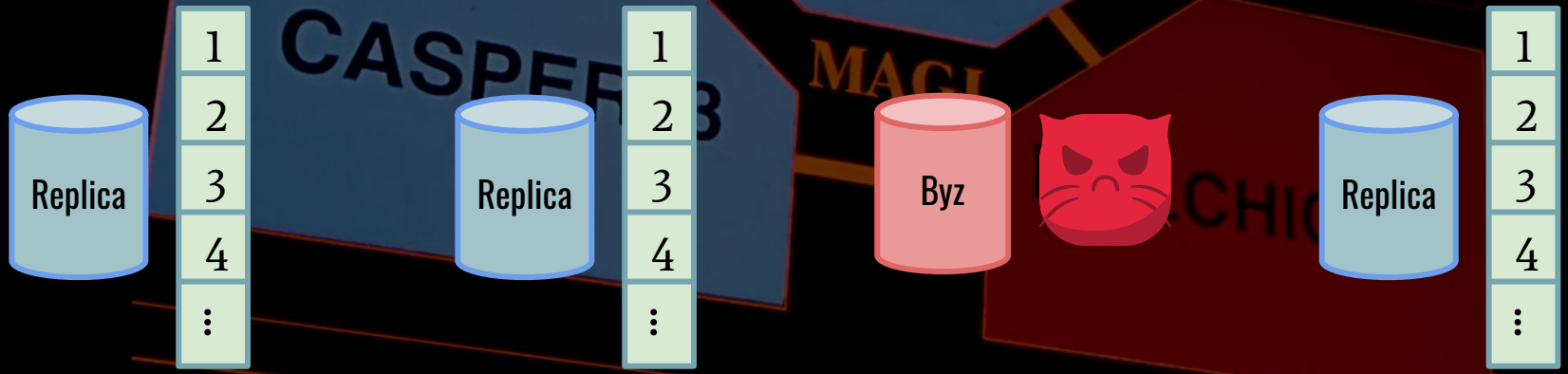


BFT Consensus

- *(Agreement)* All correct nodes must agree on the same value.
  “Safety 安全性”
- *(Termination)* All nodes must eventually decide on an output value.
  “Liveness 活性”
- *(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
  n个结点中有至多有f个 “坏结点”
- *(Replica Coordination)* other n-f correct replicas process the same sequence of requests
  剩下n-f个好结点须对执行序列达成一致
  - 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."

"论文中我们展示了即便在只有一个结点宕机的情况下，任何尝试解决该问题的协议都可能无法终止。"

Model & Known Solutions

- Termination → “Probability of 1” 确定性终止到概率终止

  → Asynchronous model (Ben-Or ’83) （完全）异步模型

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

  → Partially synchronous model (DLS ’88) 部分异步模型

- Use synchronous assumption

  → Synchronous model (LSP ’82) 同步模型

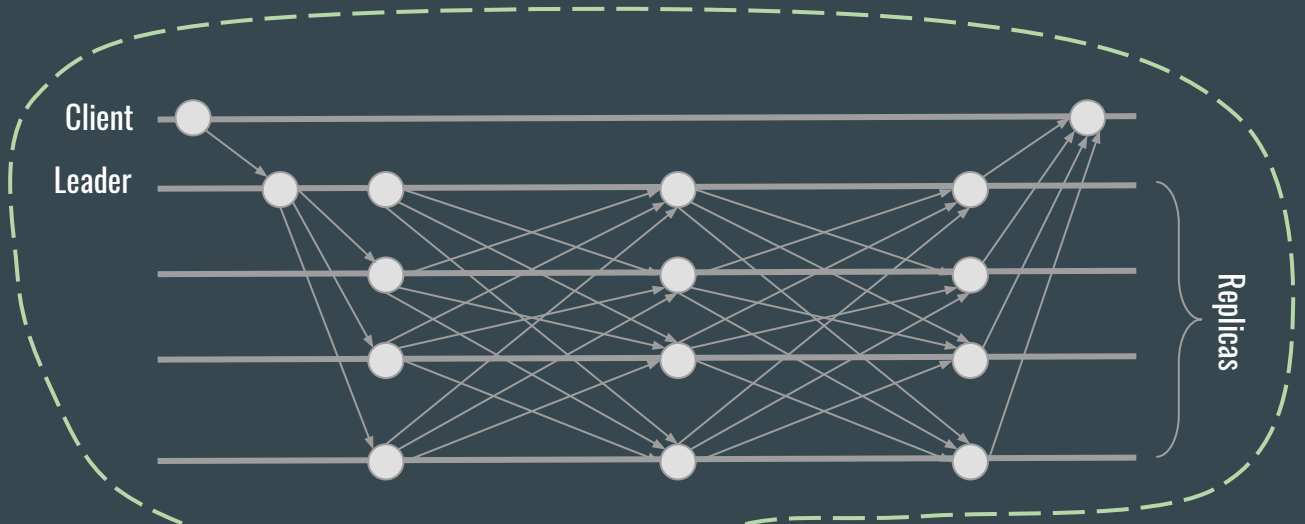
Model & Known Solutions

- Asynchronous model 异步模型
  - Randomization to hope for a skew 随机化并期待偏向某个选择
  - Example: Ben-Or ('83): exponential termination time; HoneyBadgerBFT ('16): $\Omega(n^3)$
- Partially synchronous model 部分异步模型
  - Leader-based 需要领袖
  - Example: PBFT ('99): $O(n^2)$ on a "good" day
- Synchronous model 同步模型
  - Can tolerate up to n/2 faulty nodes 可以容忍不超过一半的坏结点
  - Example: XFT (OSDI '16) is fast when f = O(1) for tiny n, very inefficient in general
  - Nakamoto Consensus (synchronous) 中本聪共识亦是同步模型

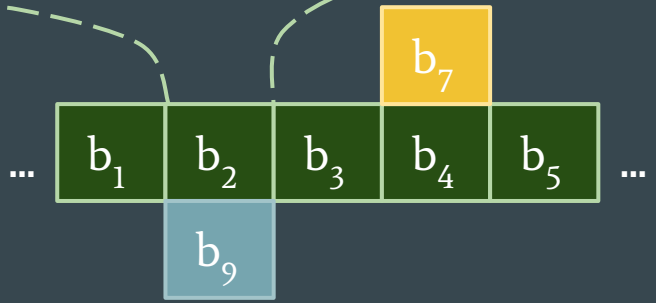
Theory vs Practice

- Asynchronous model 异步模型
  - Without a leader — everyone proposes 没领袖其实更低效
  - “Expected convergence” — non-deterministic number of rounds 期望收敛
- Partially synchronous model (practical): 部分异步模型（更实际）
  - $O(n^2)$ on a good day is still far from the benign counterpart 平方复杂度
  - Complicated and subtle operational logic 算法本身复杂，包含微妙的逻辑
  - The leader is the bottleneck 领袖即性能瓶颈
- Synchronous model: 同步模型
  - Lock-step execution — throughput bottleneck 每轮都需要同步
  - Strong assumption of delivery timeout — dilemma in choosing Δ Δ取值两难论
  - Nakamoto Consensus: PoW is prohibitively expensive PoW挖矿极低效

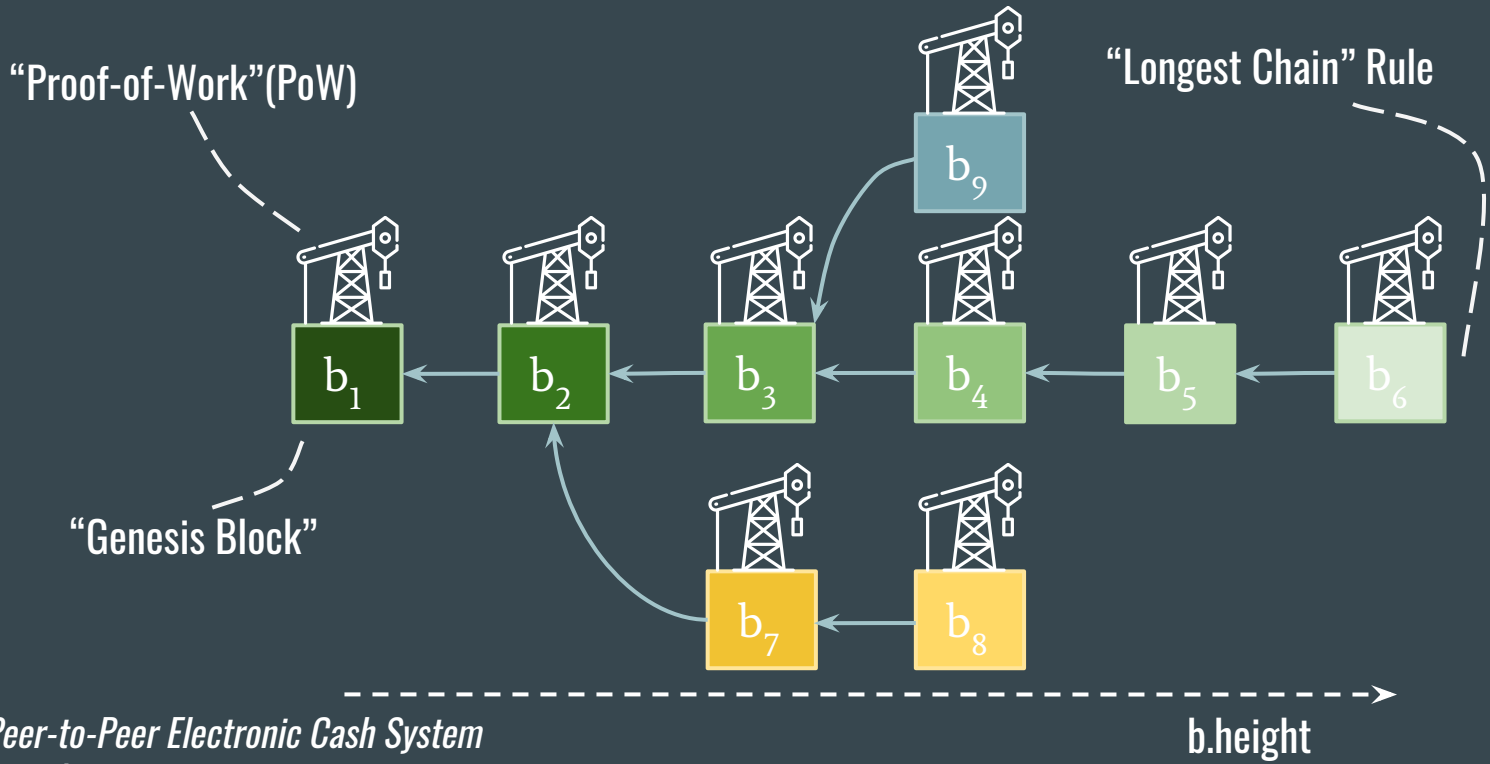
Paradigm: Quorum/Vote-Based



Example: PBFT



Paradigm: Nakamoto Consensus

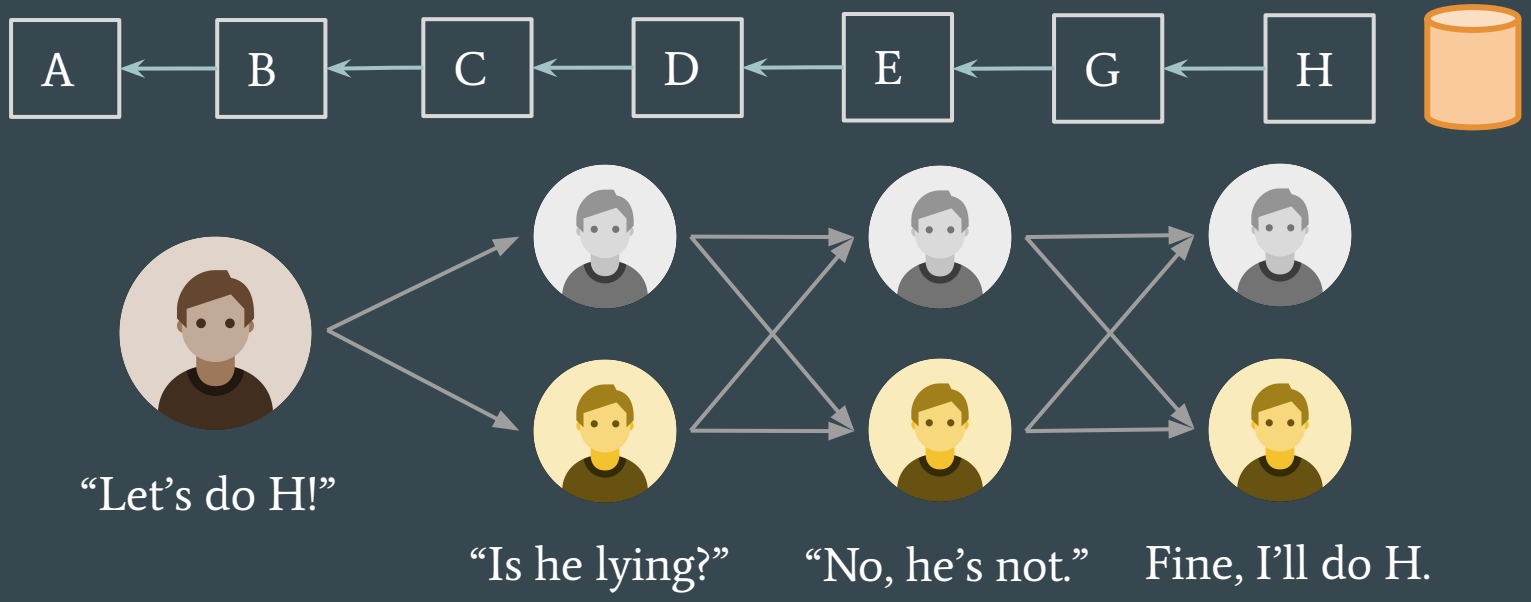


Bitcoin: A Peer-to-Peer Electronic Cash System
Satoshi Nakamoto

State Machine Replication

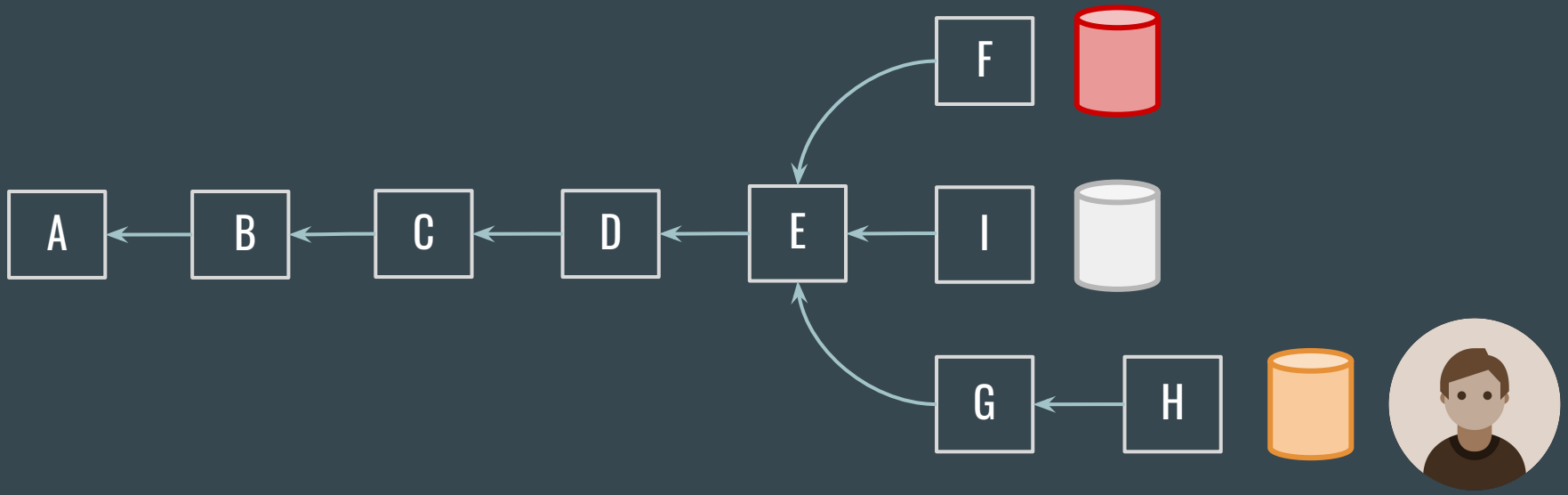
Practical Byzantine Fault Tolerance

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
  ...有着扎实的理论基础
- ...use no PoW as the core consensus mechanism
  ...无需费力挖矿
- ...are simple and practical enough to be faithfully implemented
  ...足够简洁实际，以便开发实现
- ...decouple the safety and liveness aspects of the problem to some extent
  ...一定程度上将安全性和活性解耦

The Next-Gen BFT Protocols

- “HotStuff” Project:
  - standard *partial synchrony* model
    采用标准的部分异步模型
  - the new king of the old paradigm realm
    已有共识范式下性能之王
- “Snow/Avalanche” Project:
  - looks more like randomized *async protocols*
    看上去更像是随机化异步协议
  - new realm: random subsampling for the first time
    首次尝试以统计学采样实现共识

The Next-Gen: HotStuff

- A Paradigm distilled from quorum-based consensus — “Quorum Certificate”
  从使用Quorum的共识协议提炼的范式： “Quorum证书”
- Revisit the “old wine”: DLS (and Tendermint/Casper) and locking mechanism
  再品陈年老酿之DLS锁定机制
- “Blockchain-style”, no special treatment of view change “区块链”风格：换届无特殊处理

libra

facebook

Linearity 线性复杂度

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

"The communication cost is linear"

(Optimistic) Responsiveness 响应度

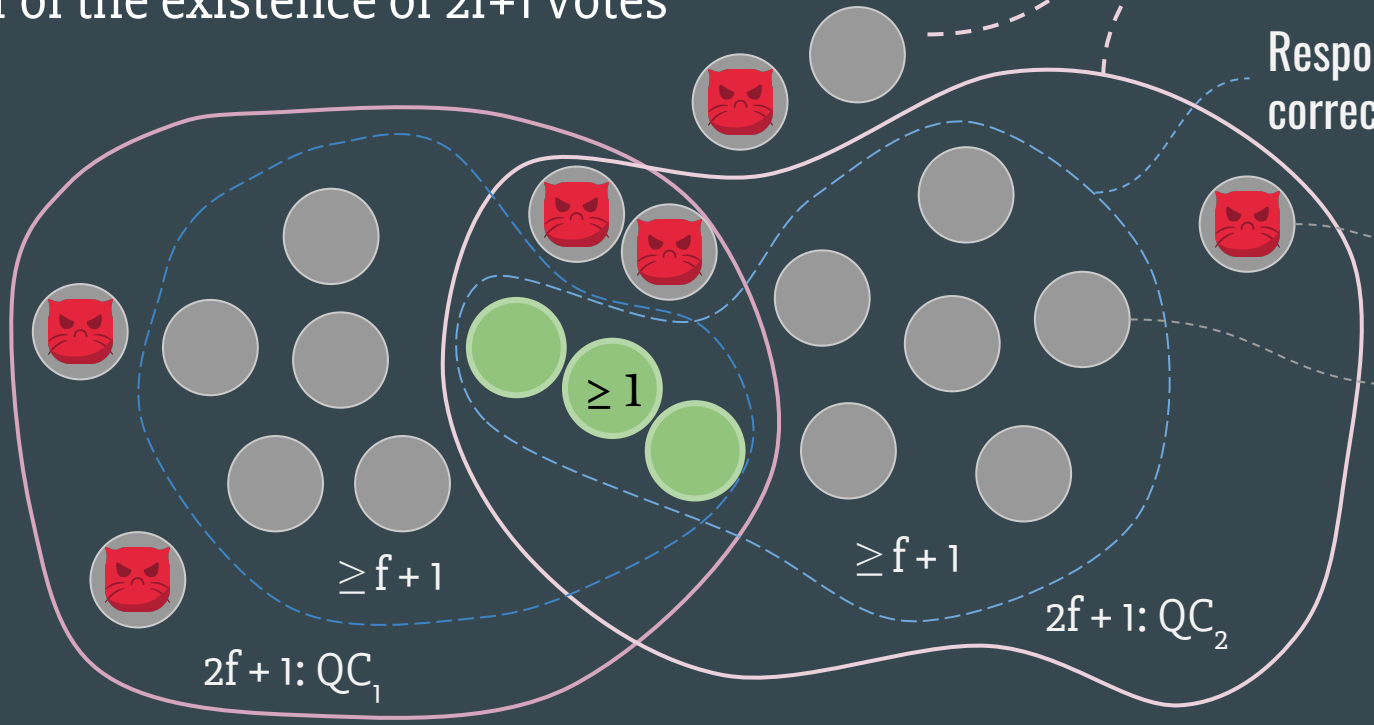
After GST, any correct leader, once designated, needs to wait just for the first *(n–f)* responses to guarantee that it can create a proposal that will make progress.

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

Protocol	Correct Leader	Leader Failure	f Leader Failures	Responsive
DLS	$O(n^4)$	$O(n^4)$	$O(n^4)$	
PBFT	$O(n^2)$	$O(n^3)$	$O(fn^3)$	✓
SBFT	$O(n)$	$O(n^2)$	$O(fn^2)$	✓
Tendermint/Casper	$O(n^2)$	$O(n^2)$	$O(fn^2)$	
Tendermint/Casper*	$O(n)$	$O(n)$	$O(fn)$	
HotStuff	$O(n)$	$O(n)$	$O(fn)$	✓

Quorum Certificate (QC)

Proof of the existence of 2f+1 votes

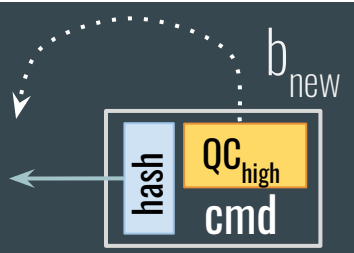


HotStuff: Data Structure

Messages

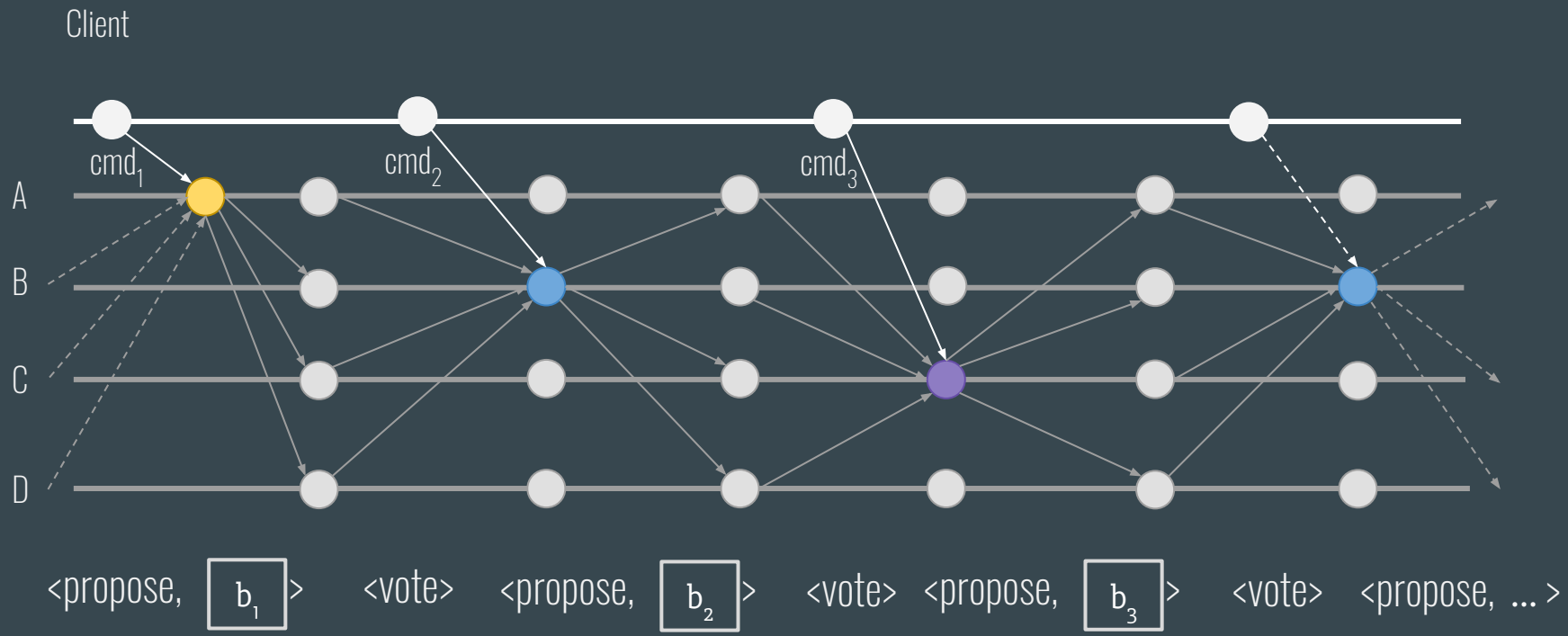
<propose, b_{new}>

<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.

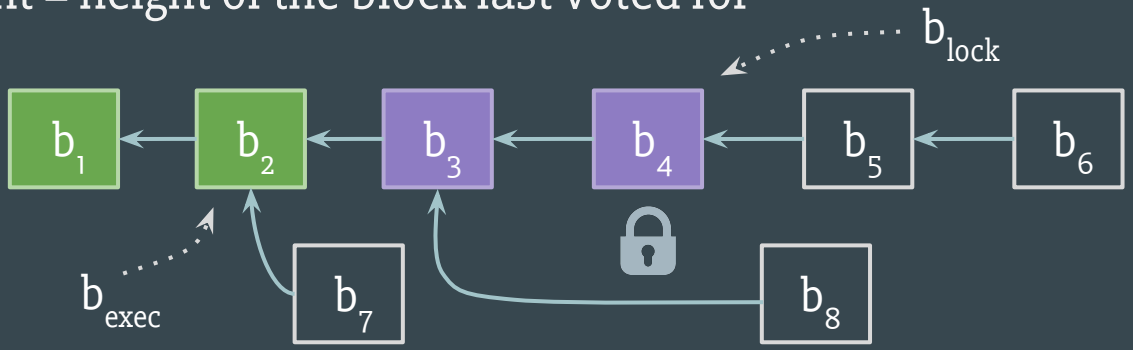
HotStuff: Communication Pattern



HotStuff: The Protocol

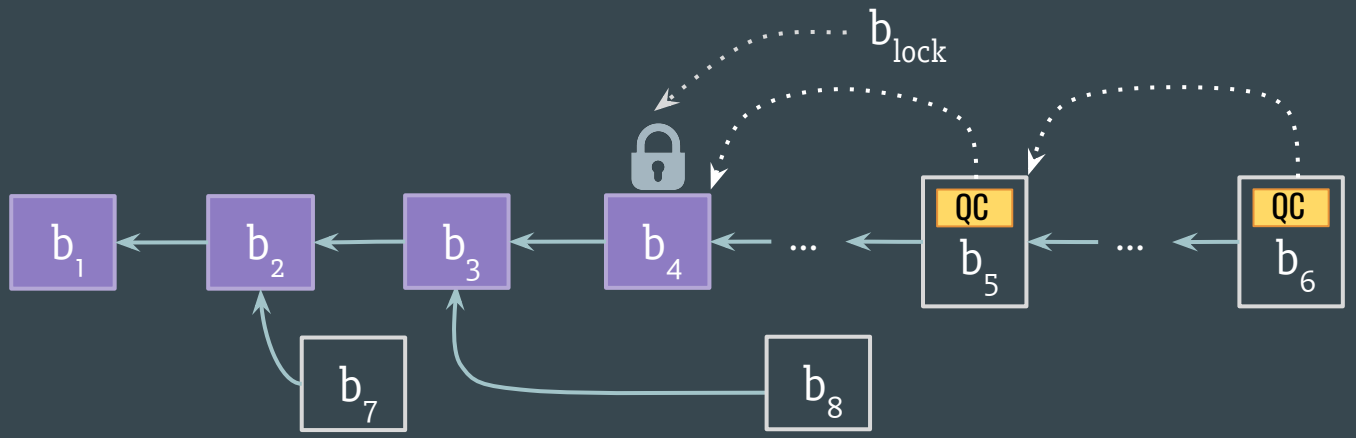
Protocol State Variables

- 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

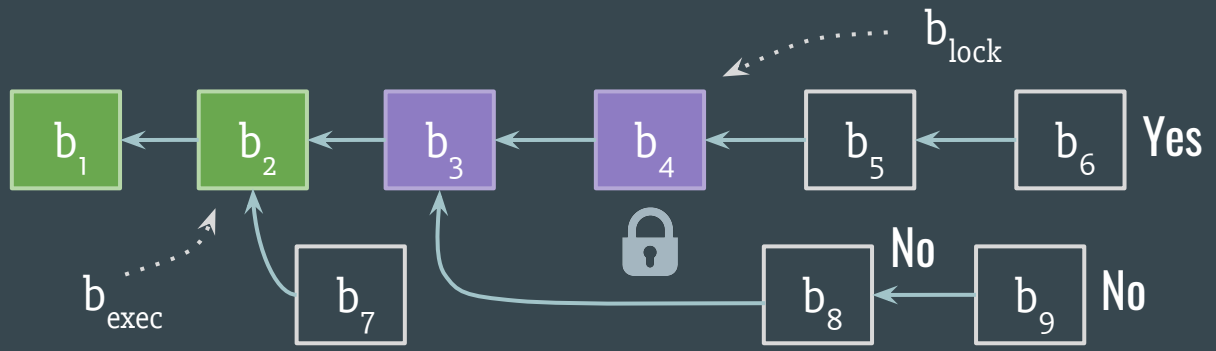
- b_{lock}: the “locked block” that leads the preferred branch
- Locking mechanism: a replica only votes for the block on the preferred branch, unless...



HotStuff: Voting

How to Vote? (Safety Rule)

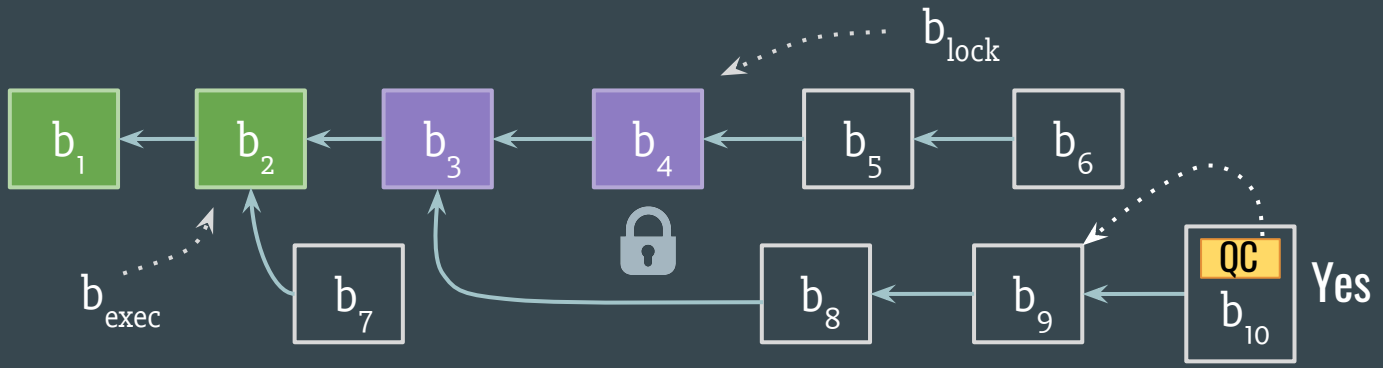
- Only vote for b_{new} if the following constraints hold:
  - b_{new}.height > vheight
  - (b_{new} is on the same branch as b_{lock}) or (b_{new} .justify .node.height > b_{lock}.height)



HotStuff: Voting

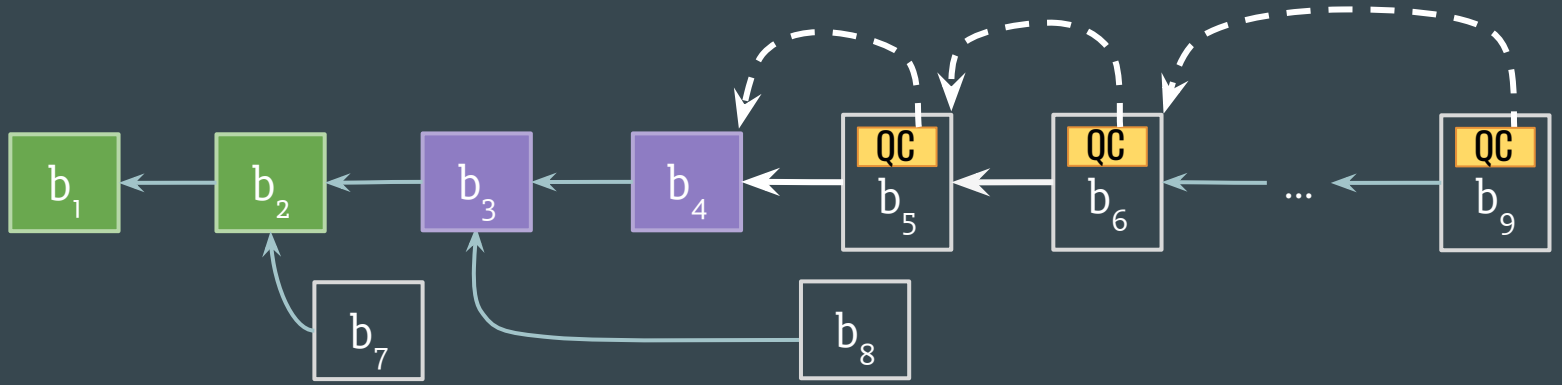
How to Vote? (Liveness Rule)

- 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)



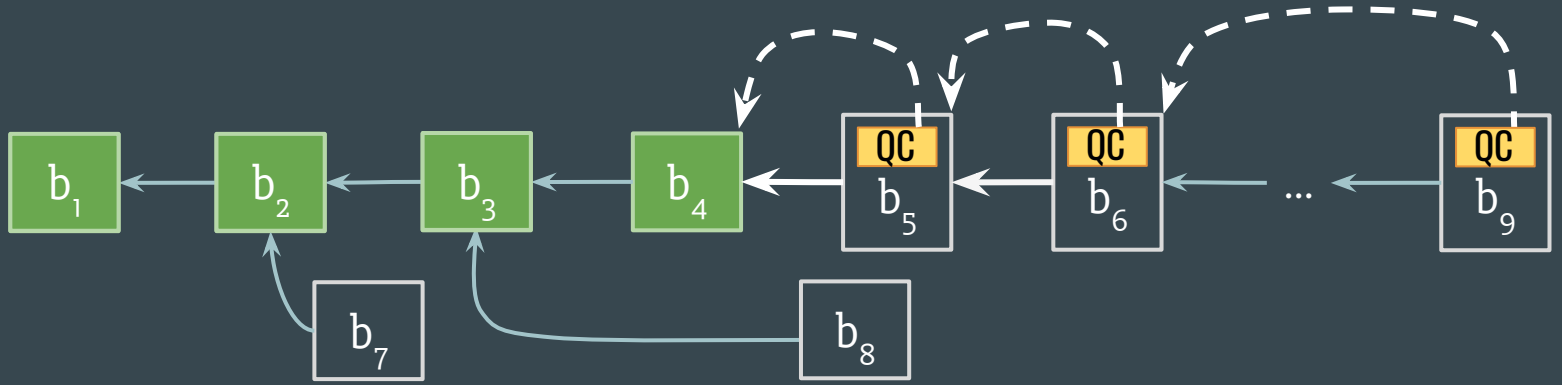
HotStuff: Decision

When to Commit?

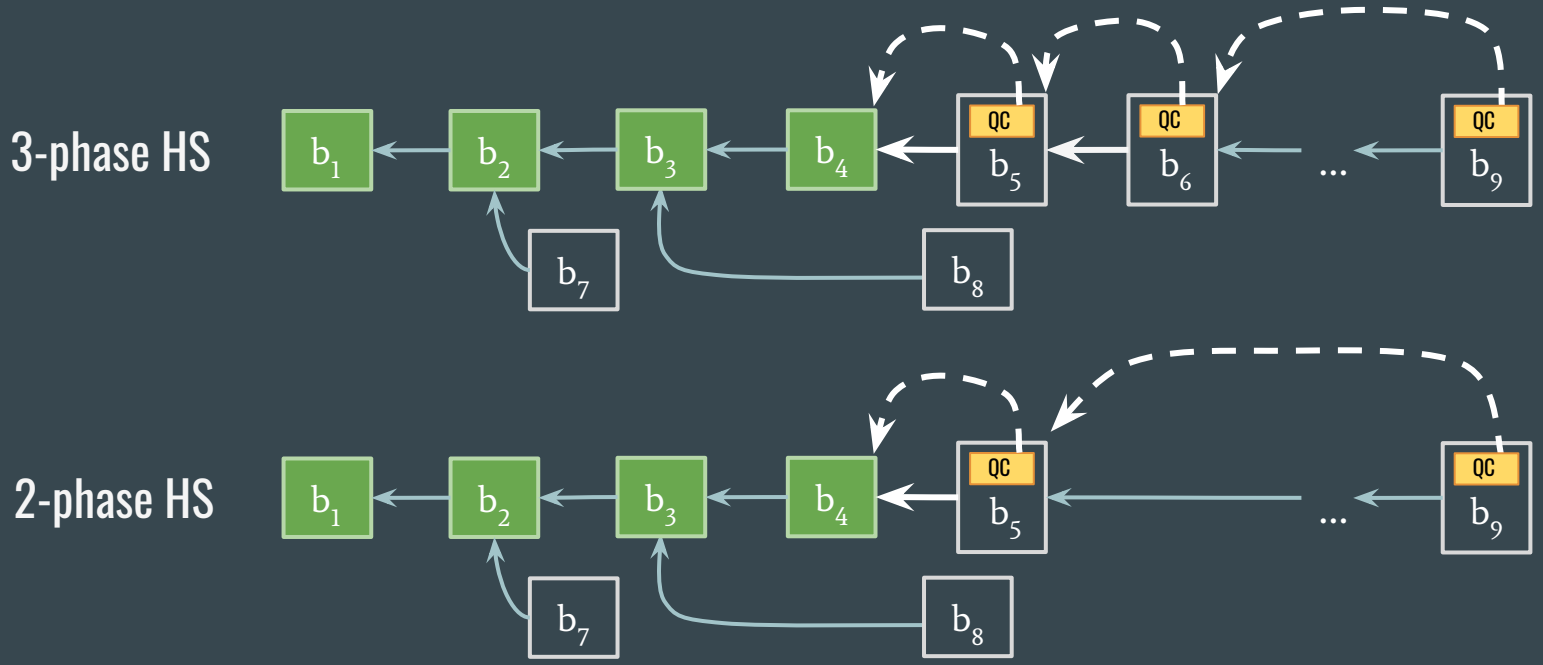


HotStuff: Decision

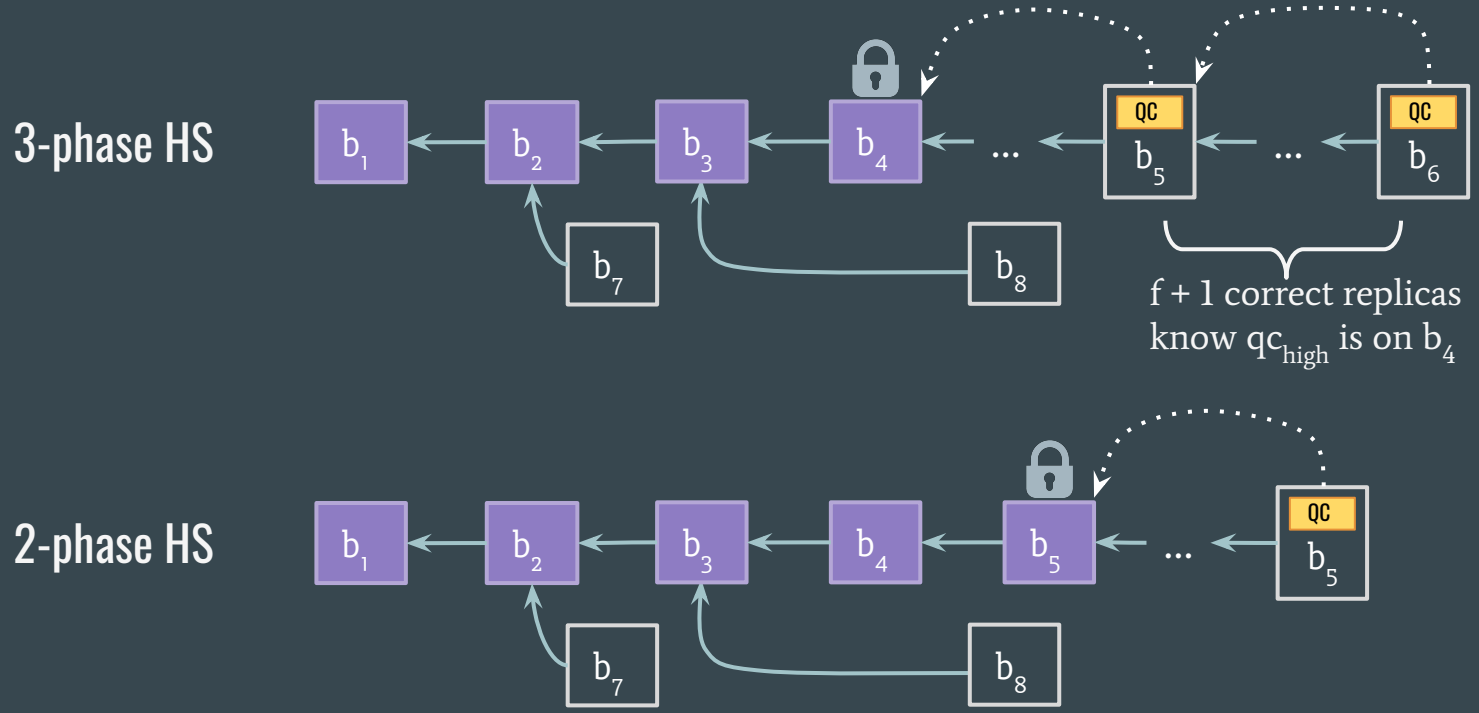
When to Commit?



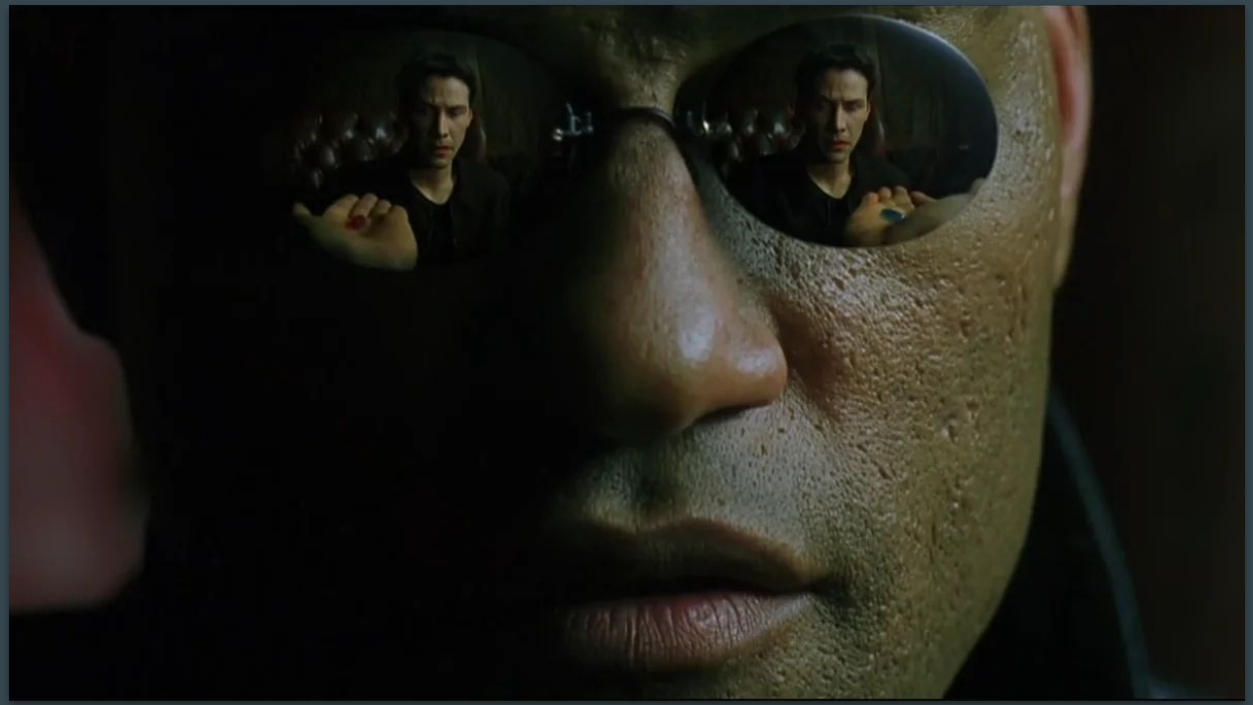
HotStuff Framework: Commit Rule



HotStuff Framework: Branch Preference



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

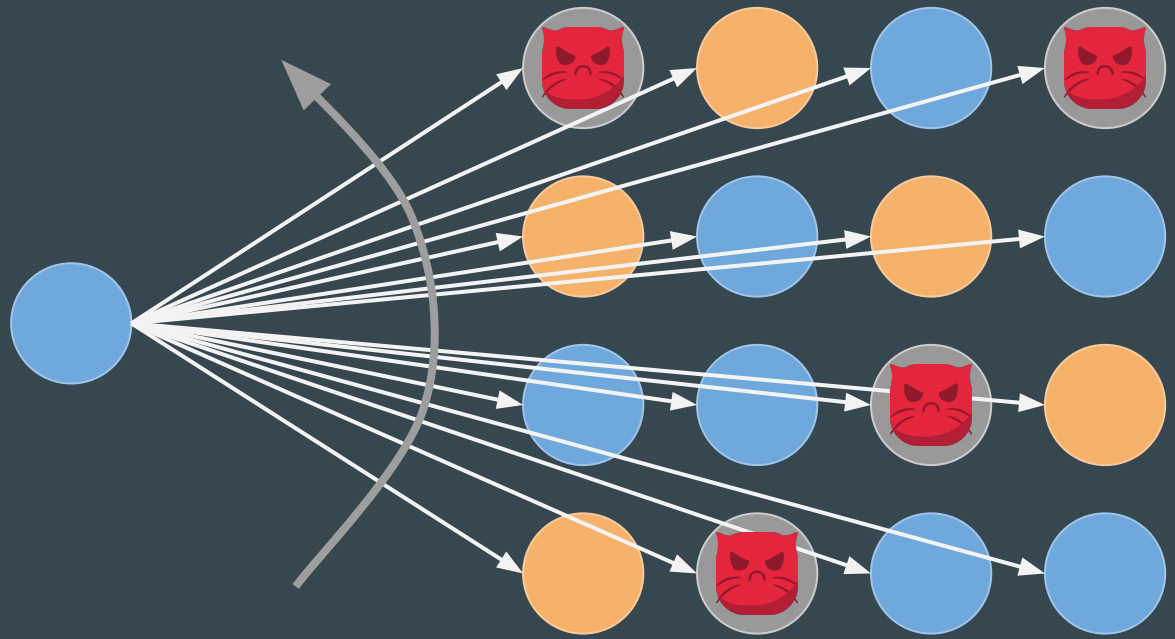


The Next-Gen: Snow/Avalanche Project

- New paradigm: P2P gossip can be directly used as consensus
  全新的范式：点对点流言传播算法可以直接用作共识
- Model is “in-between” the conventional sync. and async.
  网络模型介于传统的同步与异步之间
- Uses a unique stochastic process rather than quorum reasoning
  使用独特的随机过程而不是Quorum抽屉原理
- Loose membership
  成员可以更加散漫
- Weakened but practical (safety) guarantees in exchange for...
- ...significant better scalability
  以合理弱化的安全性换取极高的扩容性

Full Broadcast to Partial Sampling.

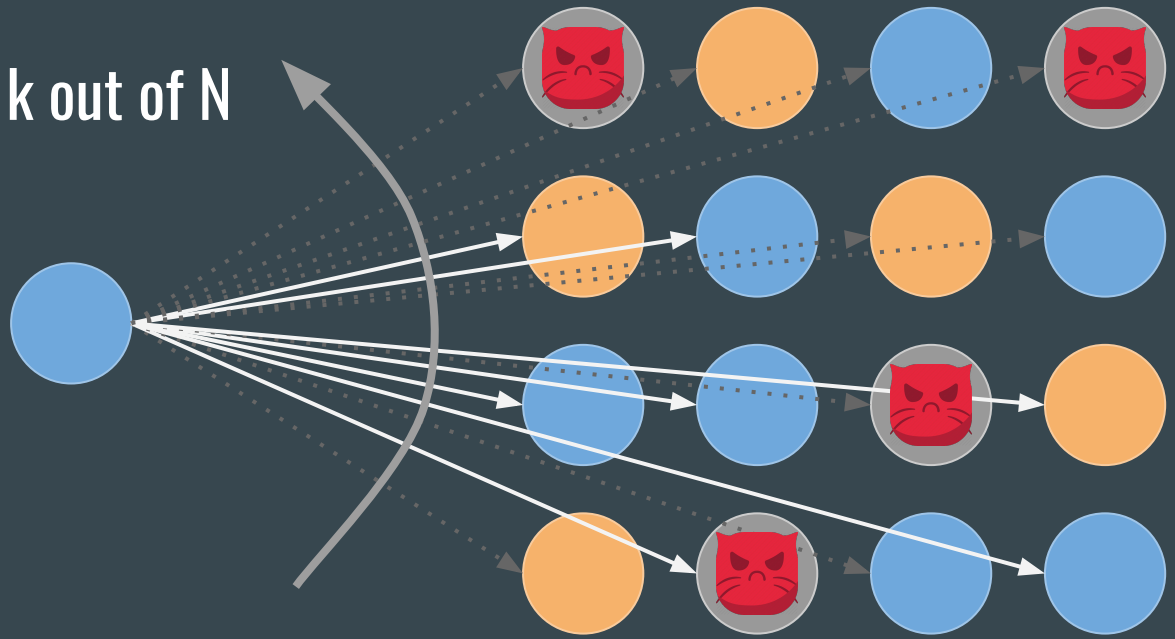
Query all

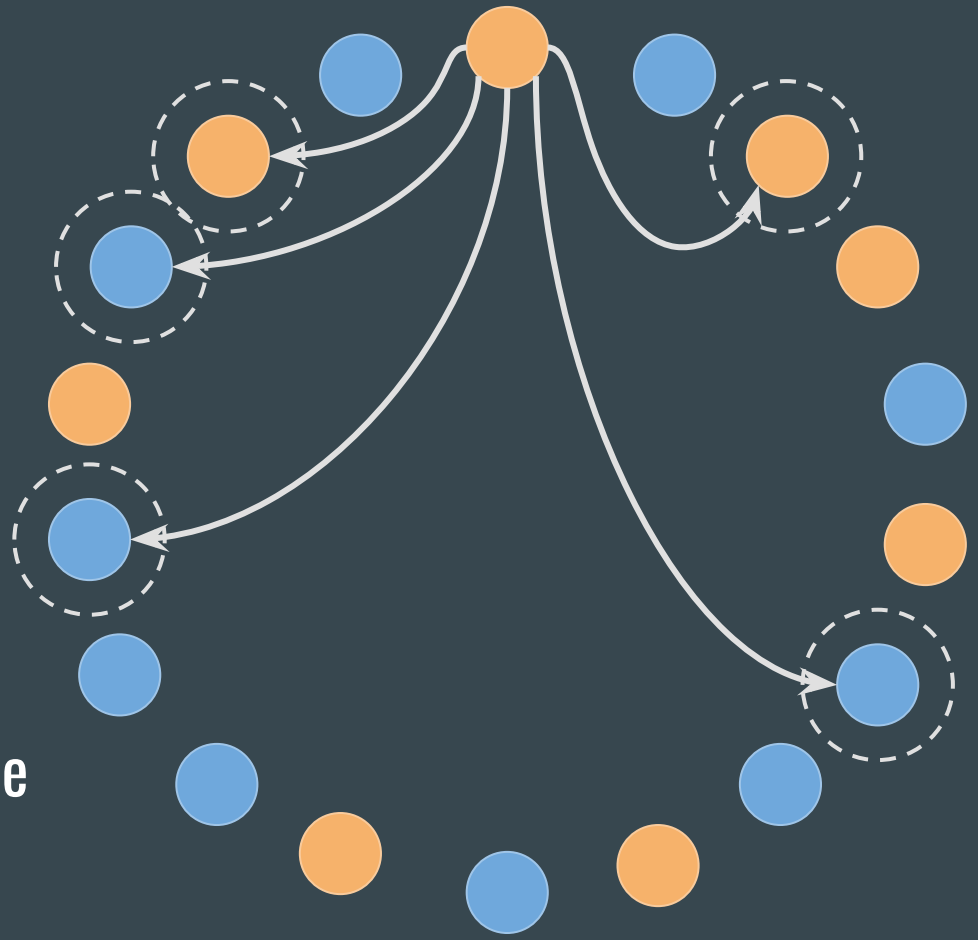


Full Broadcast to Partial Sampling.

~~Query all~~

Only Sample k out of N





Alice asks
5 other people
randomly

Alice: “Blue is the majority answer!”

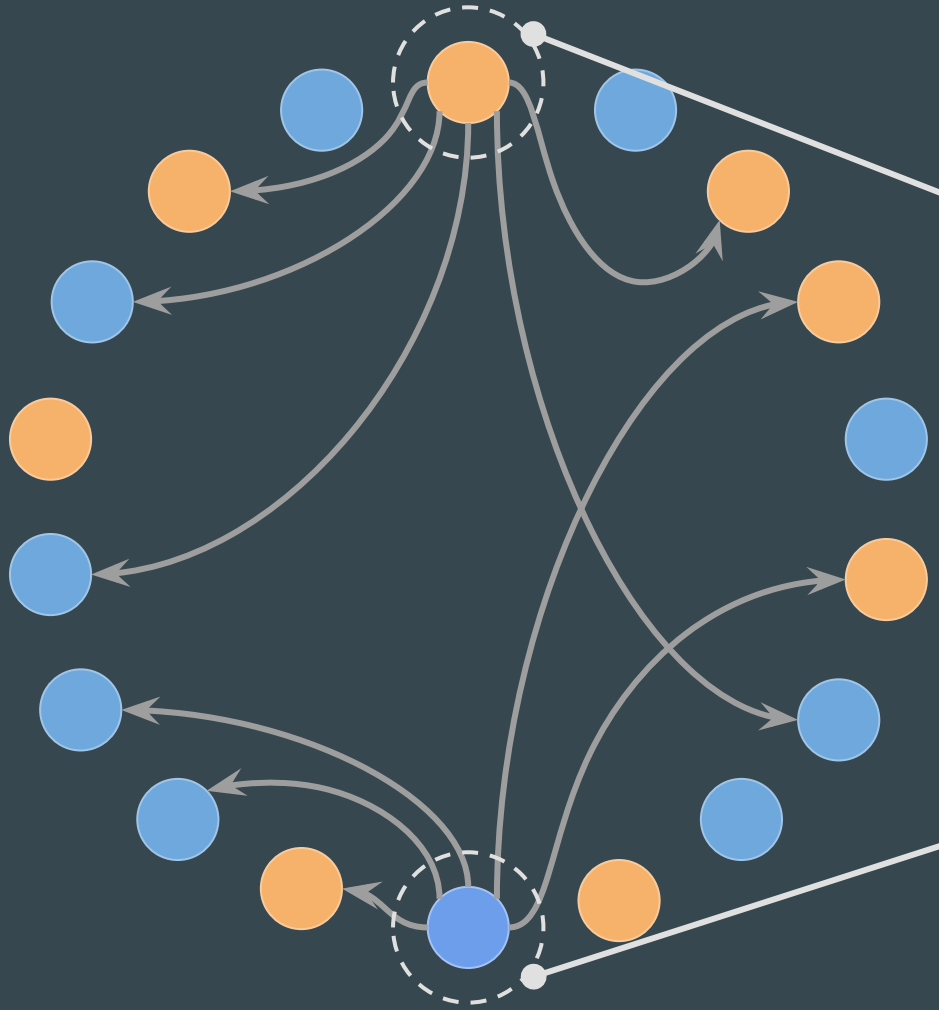
Bob asks
5 other people
randomly

Alice: “Blue is
the majority
answer!”

Alice asks
5 other people
randomly

Bob: “Yellow is
the majority
answer!”

Bob asks
5 other people
randomly

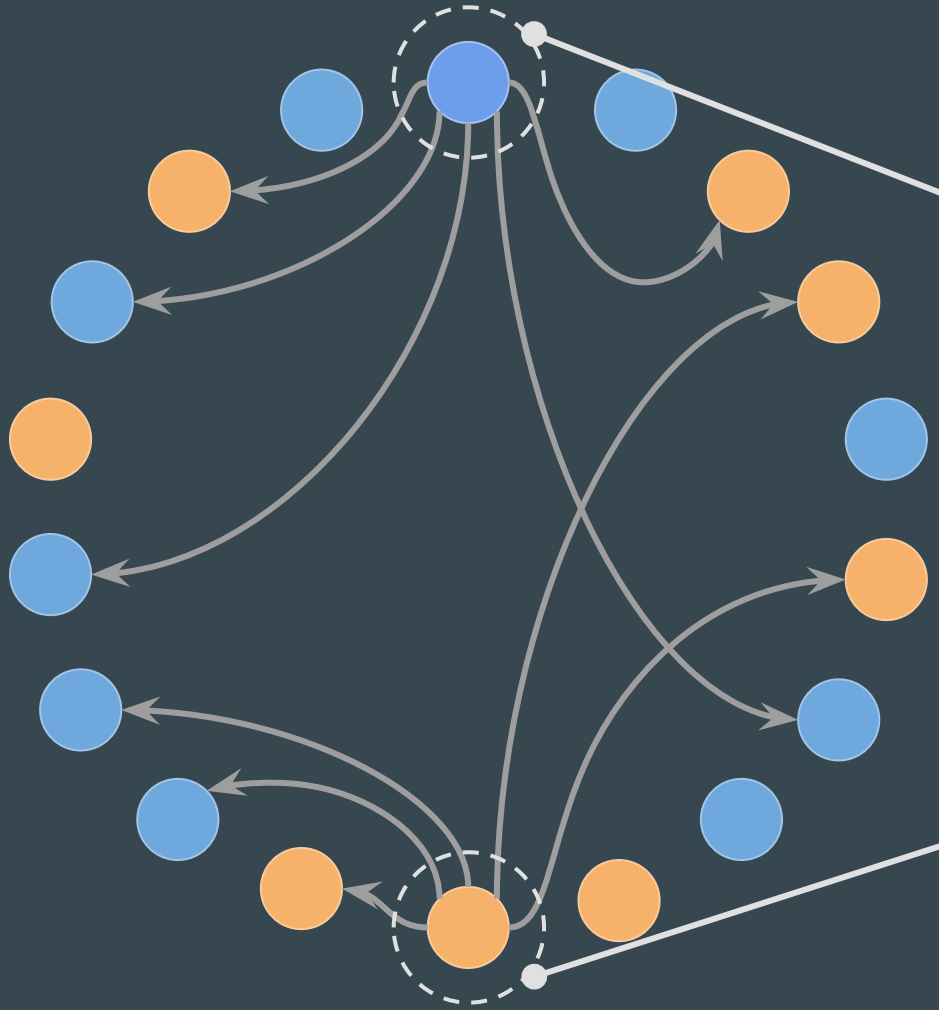


Alice: “Blue is the majority answer!”

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

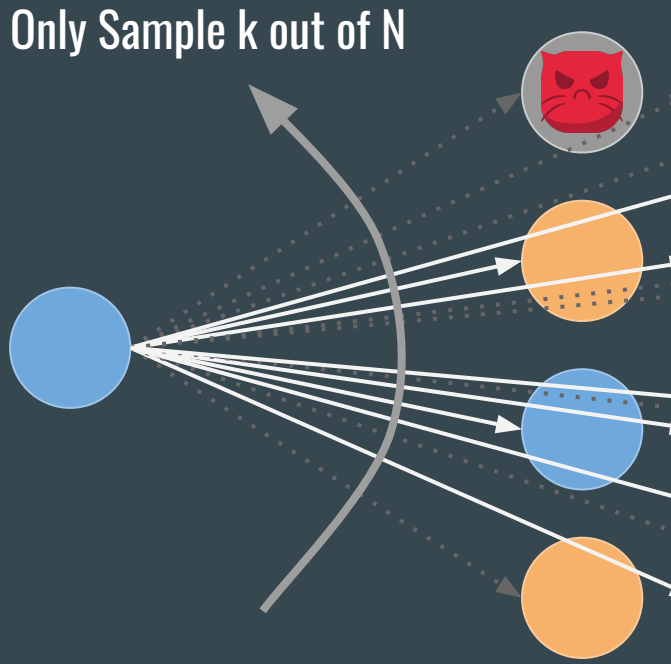
Bob: “Yellow is the majority answer!”

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



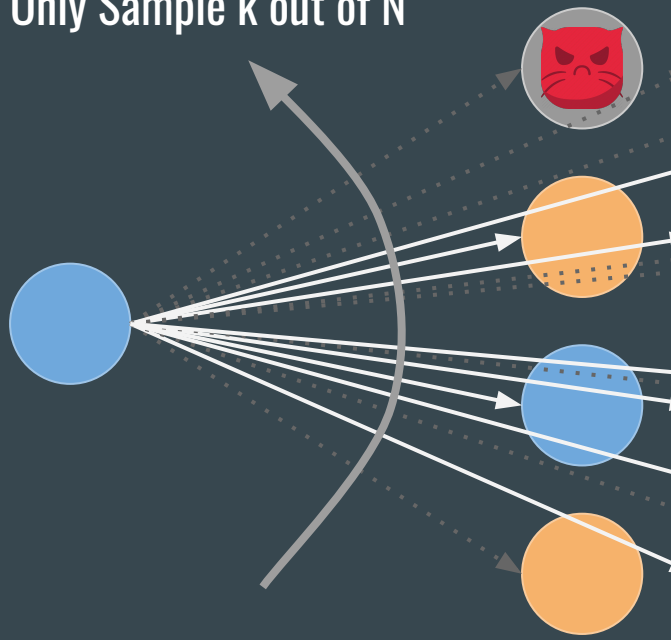
Snowball: Adding Confidence

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

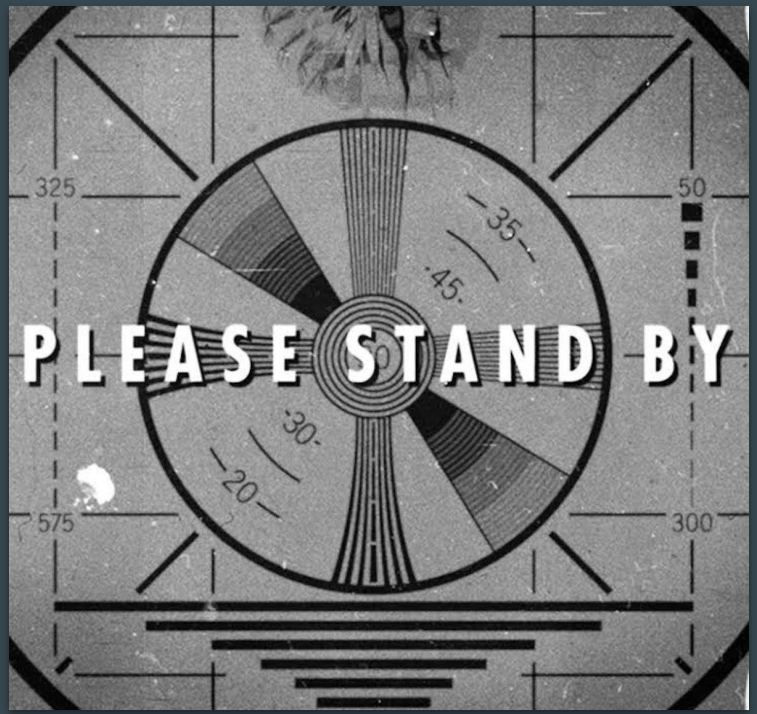
Only Sample k out of N



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　二元/多值共识
  - “Common proposal problem” in leaderless solutions
  - “Log-encoding”: a feasible reduction
- Liveness, liveness, liveness!　活性！
  - “while true {}” is always safe
  - Safety and liveness entanglement　安全性与活性 “纠缠态”
- Do we really want standard consensus?　一定要标准共识吗？
  - Achieving a total order is fundamentally a bottleneck by itself
  - Byzantine nodes: Byzantine clients that don’t deserve guarantees

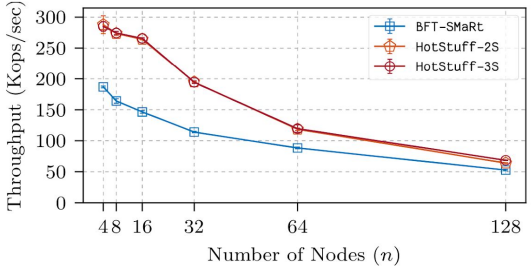
Example 1a: HotStuff and Its Pacemaker

- The original PBFT code spans >10k C code
  PBFT的科研原始C代码逾万行
- Notoriously difficult to get it right: the “fast/slow” path, view change protocol
  晦涩难以将其正确实现：正常流程和换届协议
- Liveness guarantee creeps into the logic that guarantees safety
  活性与安全性的逻辑纠缠不清
- HotStuff “Pacemaker” “起搏器”
  - safety part < 200 C code 安全性保证C代码约不超过200行
  - Customized liveness gadget 定制化活性部件
  - LibraBFT is an instantiation of such Pacemaker concept!
    LibraBFT就是起搏器概念的具体实例

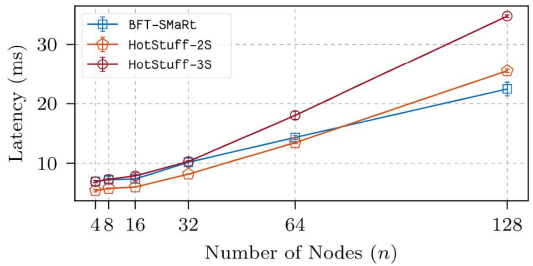
libra

facebook

Example 1a: HotStuff and Its Pacemaker

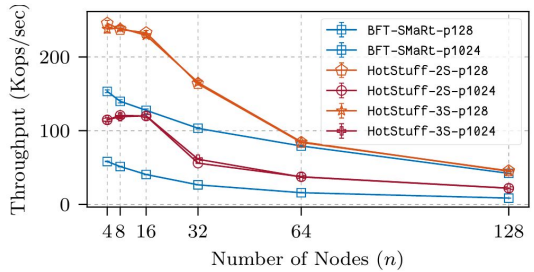


(a) Throughput

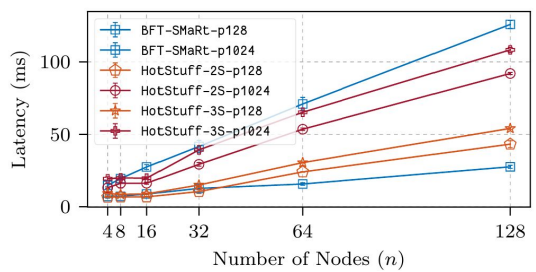


(b) Latency

Figure 6: Scalability with 0/0 payload, batch size of 400.



(a) Throughput



(b) Latency

Figure 7: Scalability for 128/128 payload or 1024/1024 payload, with batch size of 400.

Example 1a: HotStuff and Its Pacemaker

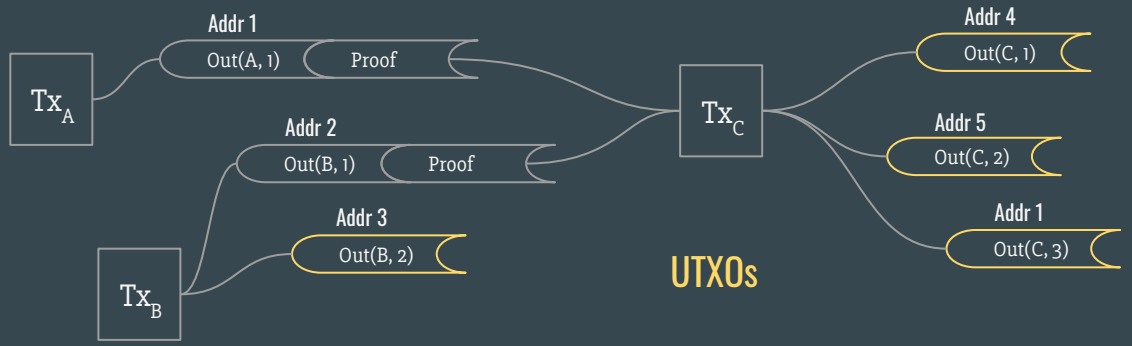
```
93
94  void HotStuffCore::update(const block_t &nblk) {
95      /* nblk = b*, blk2 = b'', blk1 = b', blk = b */
96  #ifndef HOTSTUFF_TWO_STEP
97      /* three-step HotStuff */
98      const block_t &blk2 = nblk->qc_ref;
99      if (blk2 == nullptr) return;
100     /* decided blk could possible be incomplete due to pruning */
101     if (blk2->decision) return;
102     update_hqc(blk2, nblk->qc);
103
104     const block_t &blk1 = blk2->qc_ref;
105     if (blk1 == nullptr) return;
106     if (blk1->decision) return;
107     if (blk1->height > b_lock->height) b_lock = blk1;
108
109     const block_t &blk = blk1->qc_ref;
110     if (blk == nullptr) return;
111     if (blk->decision) return;
112
113     /* commit requires direct parent */
114     if (blk2->parents[0] != blk1 || blk1->parents[0] != blk) return;
115 #else
116     /* two-step HotStuff */
117     const block_t &blk1 = nblk->qc_ref;
118     if (blk1 == nullptr) return;
119     if (blk1->decision) return;
120     update_hqc(blk1, nblk->qc);
121     if (blk1->height > b_lock->height) b_lock = blk1;
122
123     const block_t &blk = blk1->qc_ref;
124     if (blk == nullptr) return;
125     if (blk->decision) return;
126
127     /* commit requires direct parent */
128     if (blk1->parents[0] != blk) return;
129 #endif
130     /* otherwise commit */
```

Example 1b: Sync HotStuff

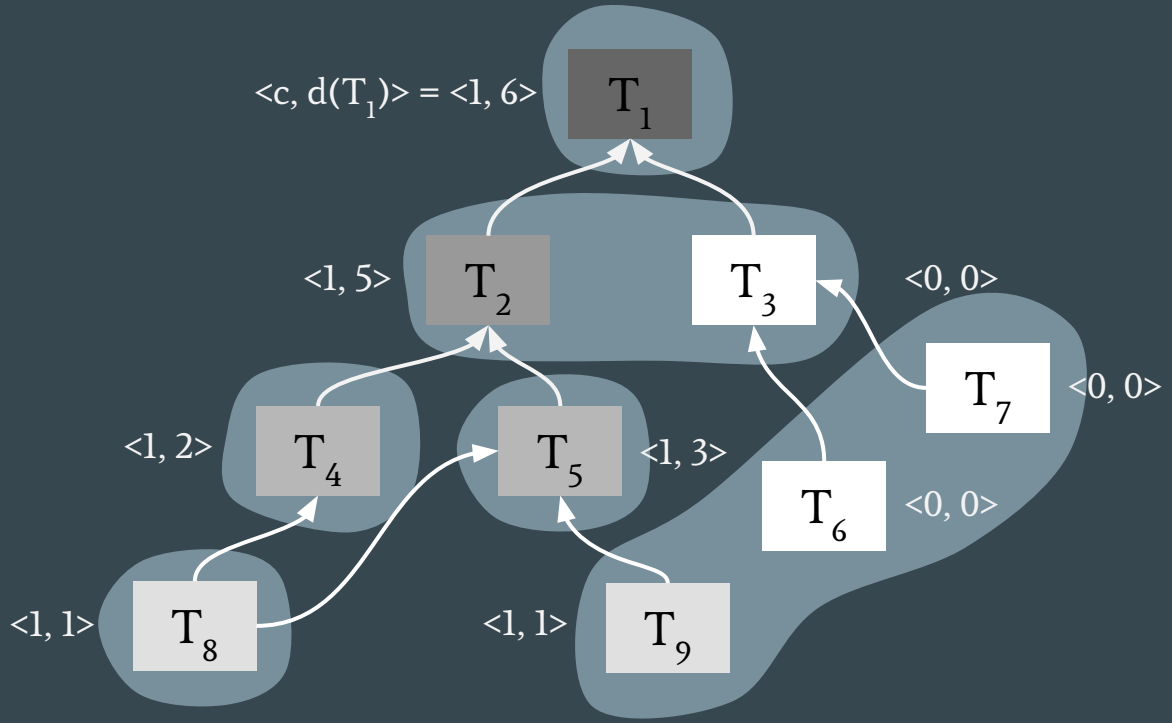
- Not a lock-step protocol: a leap to making a more practical sync BFT protocol
  非每轮全体同步的协议：迈向实用同步BFT协议的一大步
- Low (2Δ) latency for single commitment and optimistic path allows network-speed finality
  每个决策延迟低(2Δ)，“乐观”执行路径上允许网络实际速度级别的确认时间
- Uses latest sluggish model: nodes can temporarily “break” the Δ assumption
  使用“迟缓模型”，结点能够临时“打破”Δ假设

Example 2a: Snow → Avalanche

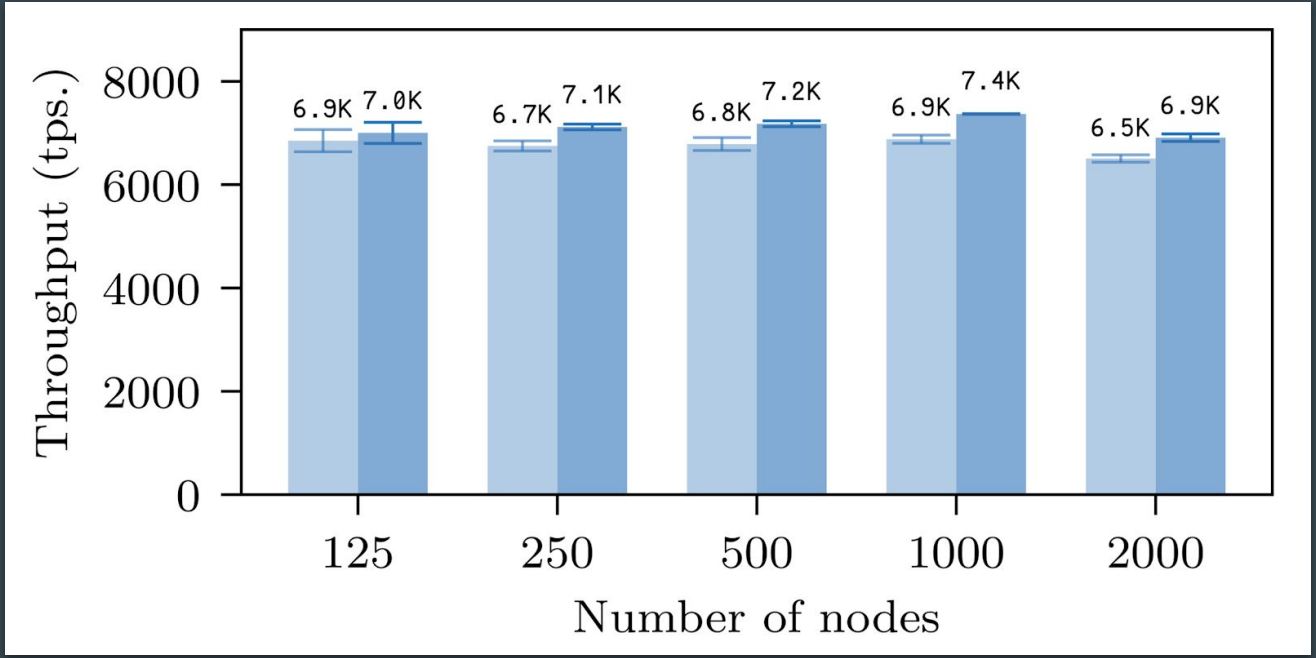
- Key observation: standard “consensus” is not a necessary requirement for making a payment system
- A payment system:
  - UTXO ensures a verifiable flow of spending
  - The only missing piece is “conflict resolution” — weaker than “consensus”
- Weakening the liveness:
  - Honest spenders → “triviality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



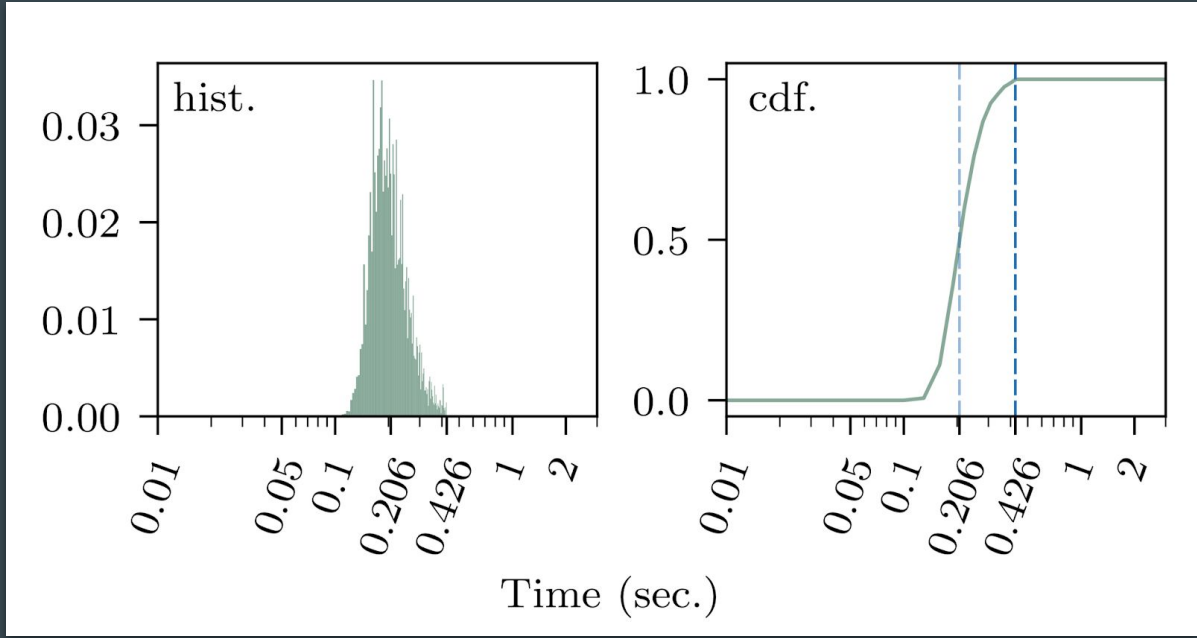
Avalanche: Cascading the Sampling Process



Avalanche Throughput.



Avalanche Latency.



Geo-replication

In an even more realistic setting:

- 2000 nodes in 20 cities across the globe
  分布在20个全球主要城市的2000个结点
- All nodes directly participate in consensus
  所有结点直接参与共识
- Full signature verification
  完整的数字签名验证

Our evaluation results:

- ~3400 tps
- ~1.35 sec

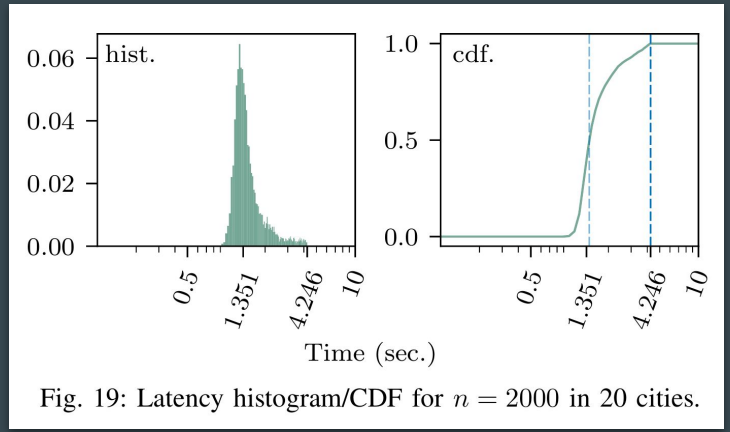
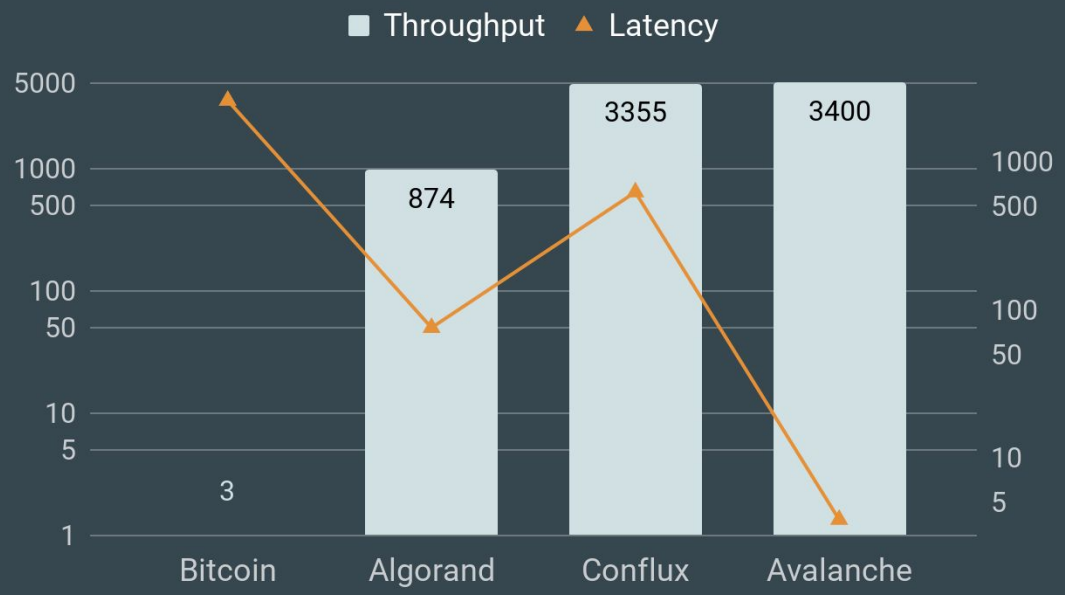


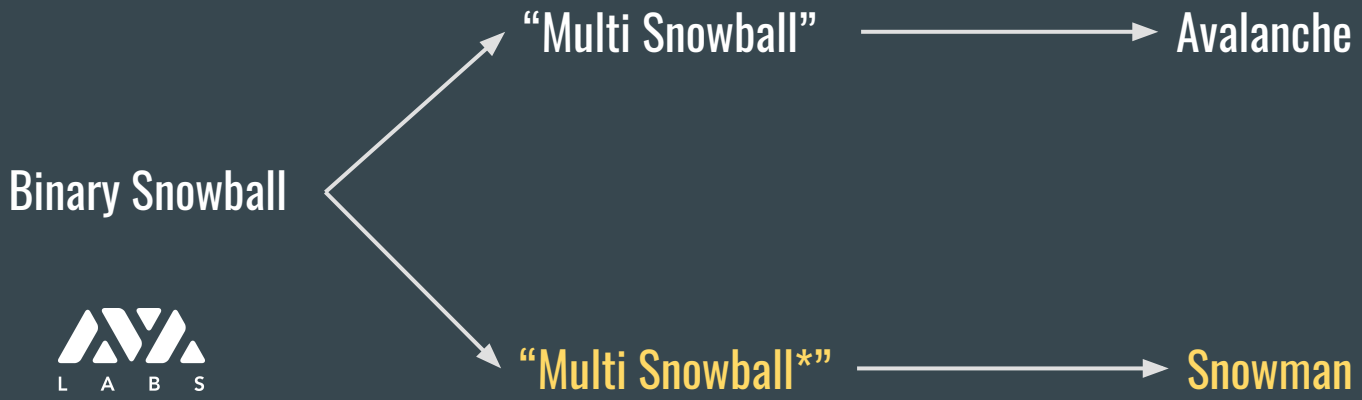
Fig. 19: Latency histogram/CDF for $n = 2000$ in 20 cities.

Evaluation: Comparison to Other Systems



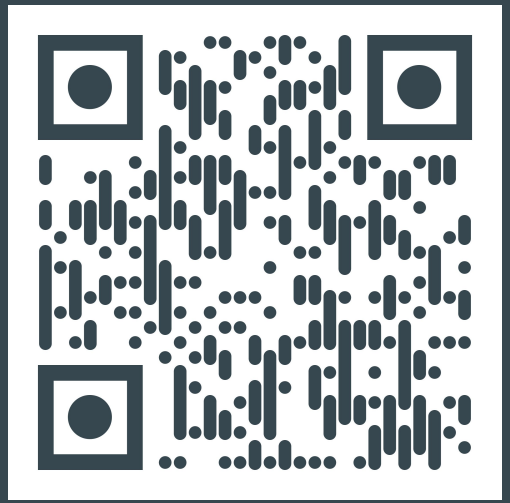
Example 2b: Snow → Snowman

- What about computation in general? 那么关于一般的计算呢？
  - Log-encoding from binary consensus to multi-value consensus
  - EVM support



我的邮箱：tederminant@gmail.com
Ava Labs 网站：https://www.avalabs.org/

Thanks
感谢倾听



← HotStuff Paper

Avalanche Paper →