

# BFT Consensus

From Academic Paper to Mainnet

Alberto Sonnino

# Alberto Sonnino

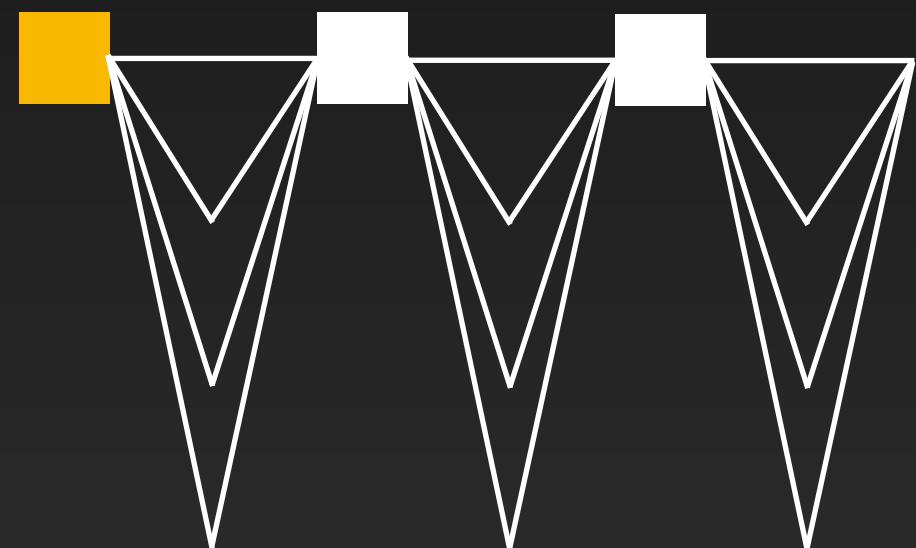
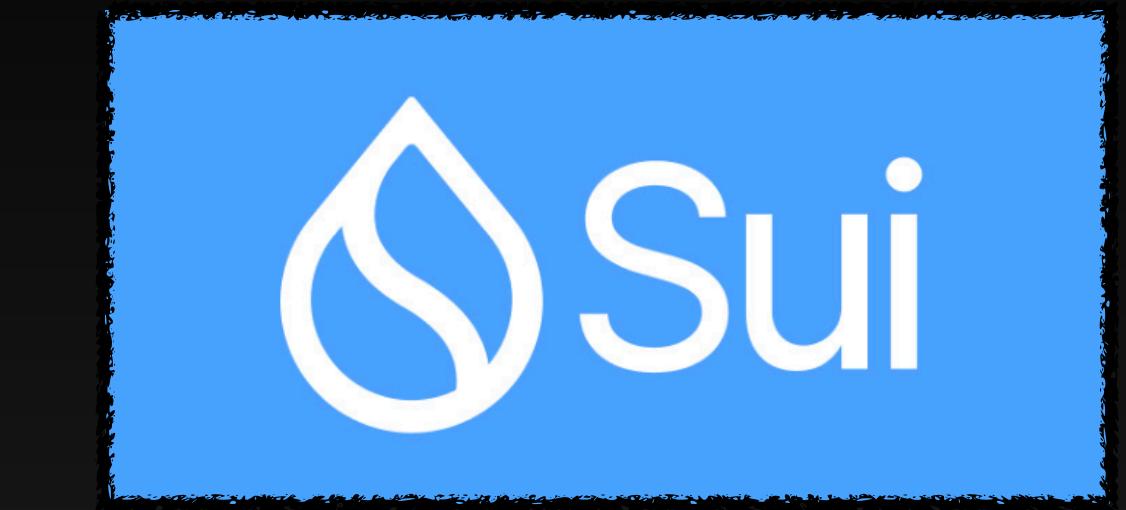
## Research Scientist

- PhD from UCL (George Danezis & Jens Groth)
- Co-founded Chainspace
- At Libra / Diem from day 1
- Now building the Sui blockchain

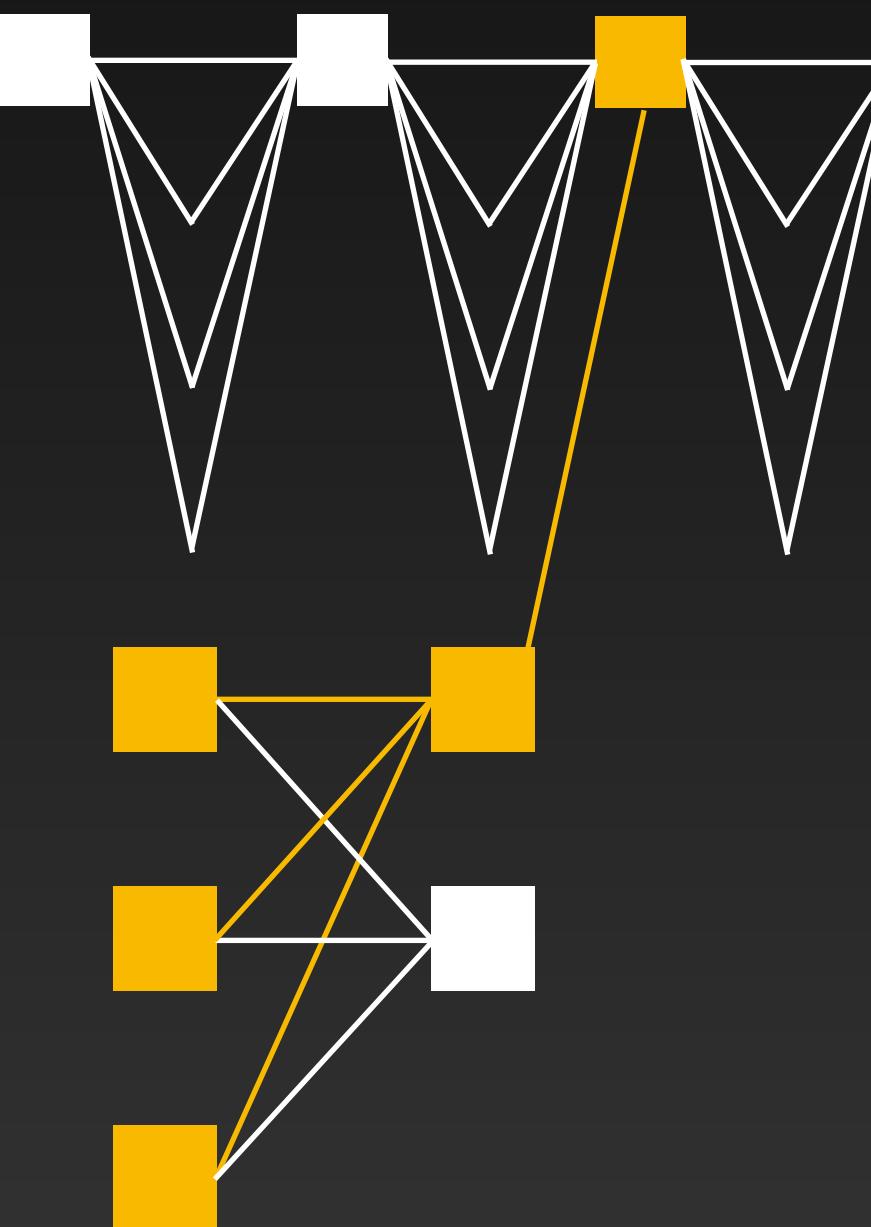
2019



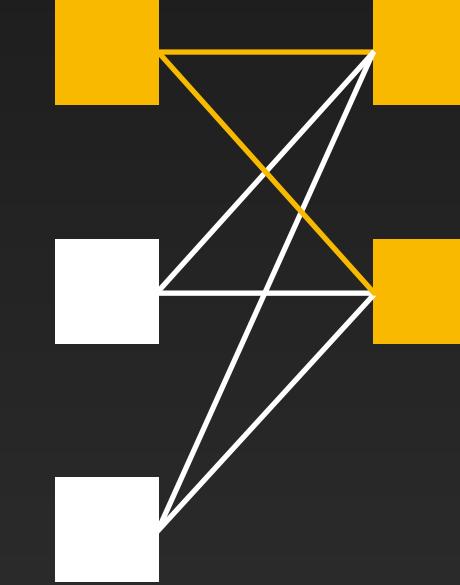
2024



**HotStuff**



**HotStuff + Mempool**

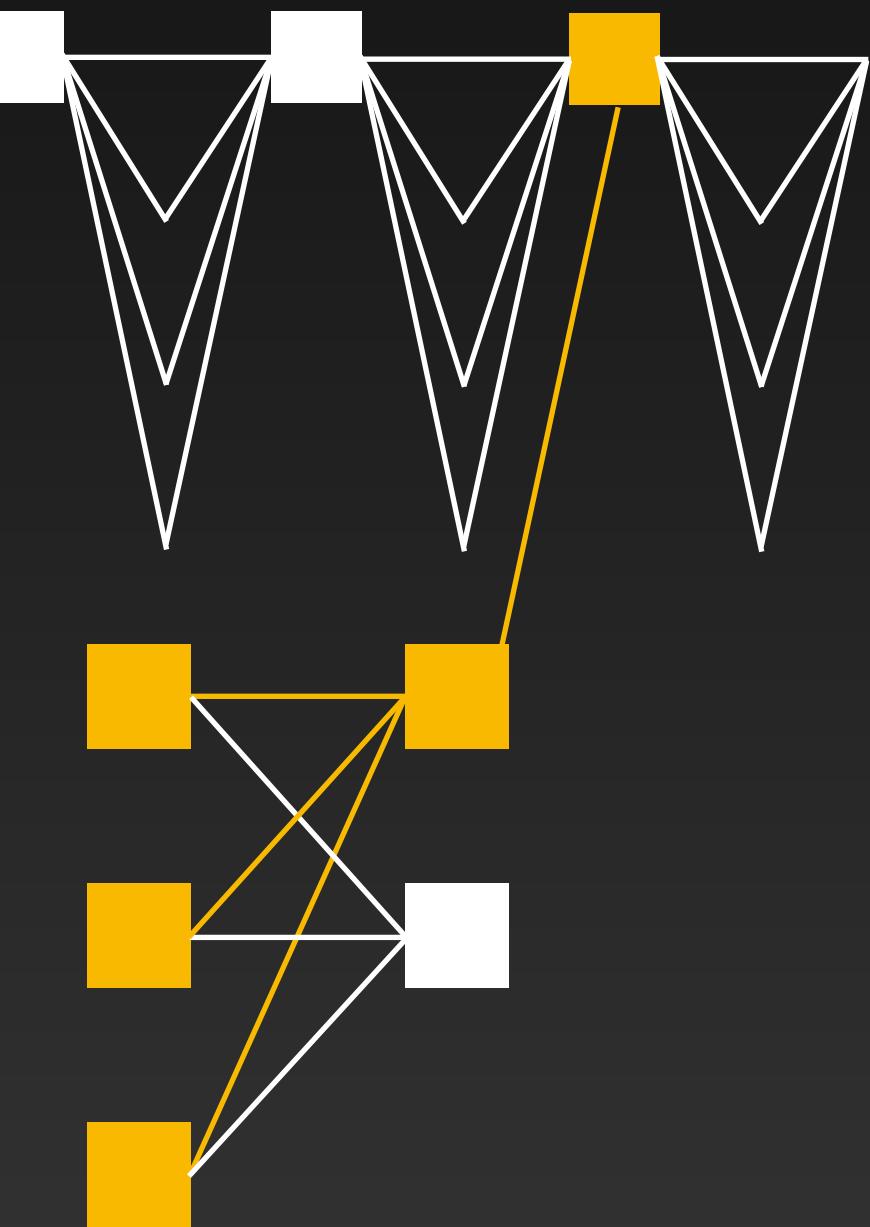


**Bullshark,  
Mysticeti**

2019



2024



- Lessons learned
- Open research challenges

# Research Gifts



(please keep it short)

# Byzantine Fault Tolerance



# Byzantine Fault Tolerance



$> 2/3$



# Partial Synchrony

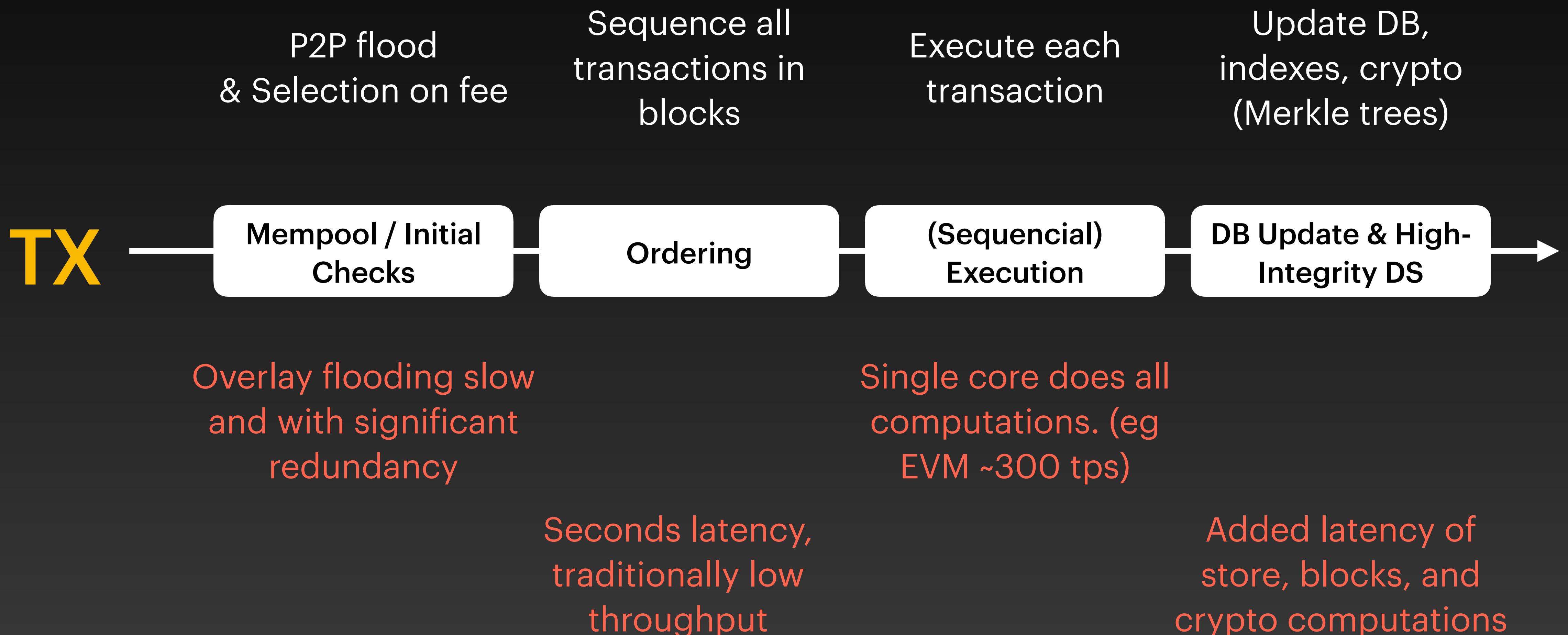


# Research Questions

1. Network model?

# Lessons Learned

# Typical Blockchain



# Typical Blockchain

**P2P flood  
& Selection on fee**

**Sequence all  
transactions in  
blocks**

**Mempool / Initial  
Checks**

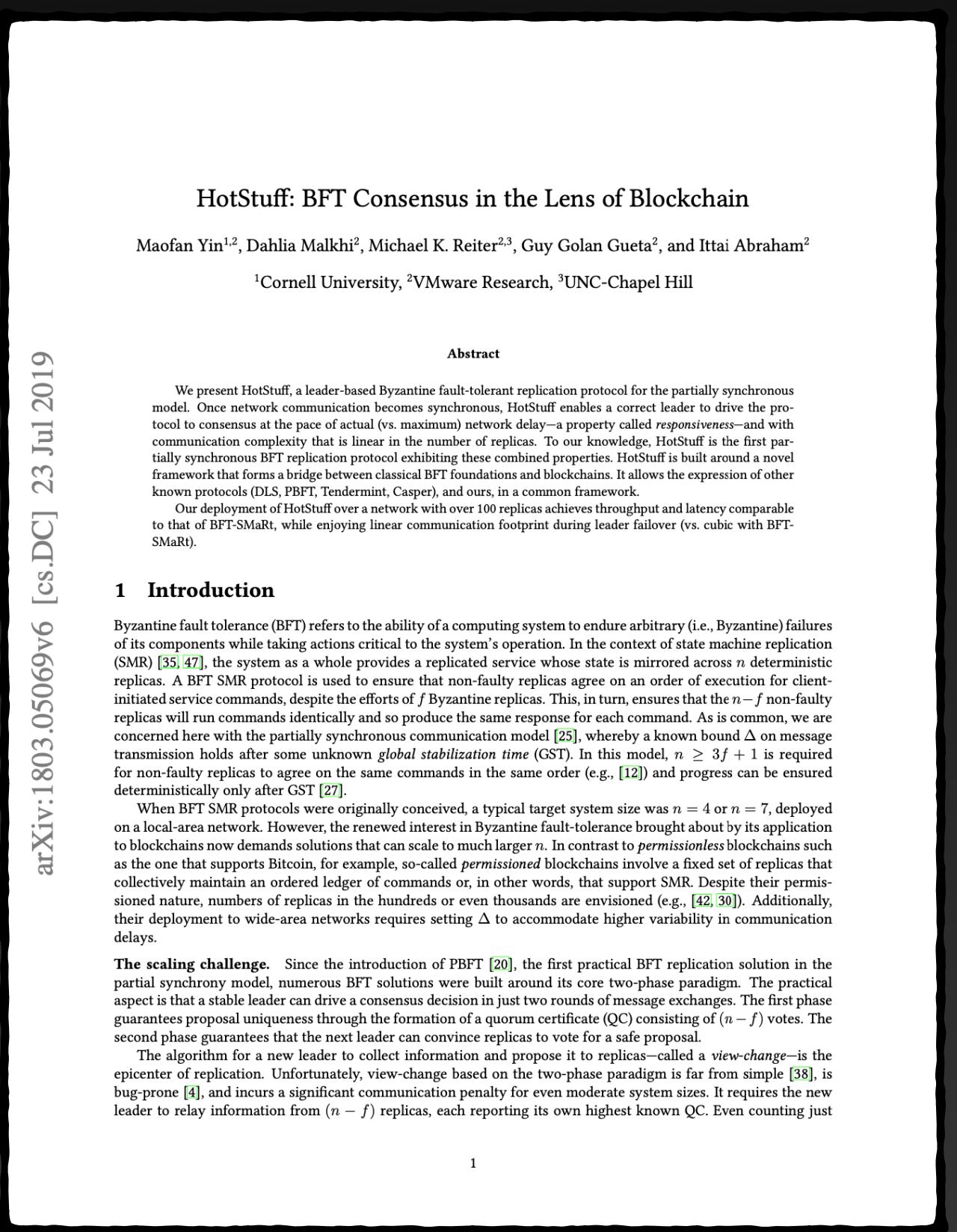
**Ordering**

**Overlay flooding  
slow and with  
significant  
redundancy**

**Seconds latency,  
traditionally low  
throughput**

# Libra, 2019

## HotStuff



arXiv:1803.05069v6 [cs.DC] 23 Jul 2019

**HotStuff: BFT Consensus in the Lens of Blockchain**

Maofan Yin<sup>1,2</sup>, Dahlia Malkhi<sup>2</sup>, Michael K. Reiter<sup>2,3</sup>, Guy Golan Gueta<sup>2</sup>, and Ittai Abraham<sup>2</sup>

<sup>1</sup>Cornell University, <sup>2</sup>VMware Research, <sup>3</sup>UNC-Chapel Hill

**Abstract**

We present HotStuff, a leader-based Byzantine fault-tolerant replication protocol for the partially synchronous model. Once network communication becomes synchronous, HotStuff enables a correct leader to drive the protocol to consensus at the pace of actual (vs. maximum) network delay—a property called *responsiveness*—and with communication complexity that is linear in the number of replicas. To our knowledge, HotStuff is the first partially synchronous BFT replication protocol exhibiting these combined properties. HotStuff is built around a novel framework that forms a bridge between classical BFT foundations and blockchains. It allows the expression of other known protocols (DLS, PBFT, Tendermint, Casper), and ours, in a common framework.

Our deployment of HotStuff over a network with over 100 replicas achieves throughput and latency comparable to that of BFT-SMaRt, while enjoying linear communication footprint during leader failover (vs. cubic with BFT-SMaRt).

**1 Introduction**

Byzantine fault tolerance (BFT) refers to the ability of a computing system to endure arbitrary (i.e., Byzantine) failures of its components while taking actions critical to the system's operation. In the context of state machine replication (SMR) [35, 47], the system as a whole provides a replicated service whose state is mirrored across  $n$  deterministic replicas. A BFT SMR protocol is used to ensure that non-faulty replicas agree on an order of execution for client-initiated service commands, despite the efforts of  $f$  Byzantine replicas. This, in turn, ensures that the  $n - f$  non-faulty replicas will run commands identically and so produce the same response for each command. As is common, we are concerned here with the partially synchronous communication model [25], whereby a known bound  $\Delta$  on message transmission holds after some unknown *global stabilization time* (GST). In this model,  $n \geq 3f + 1$  is required for non-faulty replicas to agree on the same commands in the same order (e.g., [12]) and progress can be ensured deterministically only after GST [27].

When BFT SMR protocols were originally conceived, a typical target system size was  $n = 4$  or  $n = 7$ , deployed on a local-area network. However, the renewed interest in Byzantine fault-tolerance brought about by its application to blockchains now demands solutions that can scale to much larger  $n$ . In contrast to *permissionless* blockchains such as the one that supports Bitcoin, for example, so-called *permissioned* blockchains involve a fixed set of replicas that collectively maintain an ordered ledger of commands or, in other words, that support SMR. Despite their permissioned nature, numbers of replicas in the hundreds or even thousands are envisioned (e.g., [42, 30]). Additionally, their deployment to wide-area networks requires setting  $\Delta$  to accommodate higher variability in communication delays.

**The scaling challenge.** Since the introduction of PBFT [20], the first practical BFT replication solution in the partial synchrony model, numerous BFT solutions were built around its core two-phase paradigm. The practical aspect is that a stable leader can drive a consensus decision in just two rounds of message exchanges. The first phase guarantees proposal uniqueness through the formation of a quorum certificate (QC) consisting of  $(n - f)$  votes. The second phase guarantees that the new leader can convince replicas to vote for a safe proposal.

The algorithm for a new leader to collect information and propose it to replicas—called a *view-change*—is the epicenter of replication. Unfortunately, view-change based on the two-phase paradigm is far from simple [38], is bug-prone [4], and incurs a significant communication penalty for even moderate system sizes. It requires the new leader to relay information from  $(n - f)$  replicas, each reporting its own highest known QC. Even counting just

1



## HashGraph



arXiv:2102.01167v1 [cs.LO] 1 Feb 2021

**Verifying the Hashgraph Consensus Algorithm**

Karl Crary  
Carnegie Mellon University

**Abstract**

The Hashgraph consensus algorithm is an algorithm for asynchronous Byzantine fault tolerance intended for distributed shared ledgers. Its main distinguishing characteristic is it achieves consensus without exchanging any extra messages; each participant's votes can be determined from public information, so votes need not be transmitted.

In this paper, we discuss our experience formalizing the Hashgraph algorithm and its correctness proof using the Coq proof assistant. The paper is self-contained; it includes a complete discussion of the algorithm and its correctness argument in English.

**1 Introduction**

Byzantine fault-tolerance is the problem of coordinating a distributed system while some participants may maliciously break the rules. Often other challenges are also present, such as a lack of communication. The challenge is the creation of a variety of new applications, such as cryptocurrencies. Such applications rely on *distributed shared ledgers*, a form of Byzantine fault-tolerance in which a set of transactions are placed in a globally-agreed total order that is *immutable*. The latter means that once a transaction enters the order, no new transaction can enter at an earlier position.

A distributed shared ledger makes it possible for all participants to agree, at any point in the order, on the current owner of a digital commodity such as a unit of cryptocurrency. A transaction transferring ownership is valid if the commodity's current owner authorizes the transaction. (The authorization mechanism—presumably using a digital signature—is beyond the scope of the ledger itself.) Because the order is total, one transaction out of any pair has priority. Thus we can show that a commodity's chain of ownership is uniquely determined. Finally, because the order is immutable, the chain of ownership cannot change except by adding new transactions at the end.

Algorithmic Byzantine consensus (under various assumptions) have existed for some time, indeed longer than the protocol has been named [12, 9]. Practical algorithms are more recent; in 1999, Castro and Liskov [6] gave an algorithm that when installed into the NFS file system slowed it only 3%. As Byzantine consensus algorithms have become more practical, they have been tailored to specific applications. Castro and Liskov's algorithm was designed for fault-tolerant state machine replication [13] and probably would not perform well under the workload of a distributed shared ledger.

However, in the last few years there have arisen Byzantine fault-tolerance algorithms suitable for distributed shared ledgers, notably HoneyBadgerBFT [10], BEAT [7], and—the subject of this paper—Hashgraph [2]. Moreover, the former two each claim to be the first practical *asynchronous* BFT algorithm (with different standards of practicality). Hashgraph does not claim to be first, but is also practical and asynchronous.

In parallel with that line of work has been the development of distributed shared ledgers based on *proof of work*, beginning with Bitcoin [11]. The idea behind proof of work is to maintain agreement on the ledger by maintaining a list of blocks of transactions, and to ensure that the list does not become a tree. To ensure this, the rules state that (1) the longest branch defines the list, and (2) to create a new block, one must solve a difficult computational problem that takes the last old header as part of its input. The problem's solution is much easier to verify than to obtain, so when one learns of a new block, one's incentive is to restart work from the new head rather than continue work from the old head.

Bitcoin and some of its cousins are widely used, so in a certain sense they are indisputably practical. They are truly permissionless, in a way that the BFT algorithms, including Hashgraph, cannot quite claim. Nevertheless, they offer severely limited throughput. Bitcoin is limited to seven transactions per second and has a latency of one hour, while its BFT competitors all do several orders of magnitude better. Proof-of-work systems are also criticized for being wasteful: an enormous amount of electricity is expended on block-creation efforts that nearly always fail. Finally—more to the point of this paper—the theoretical properties of proof of work are not well understood.

The Hashgraph consensus algorithm is designed to support high-performance applications of a distributed shared ledger. Like the other BFT systems, it is several orders of magnitude faster than proof of work. Actual performance depends very much on configuration choices (e.g., how many peers, geographic distribution, tradeoff between latency and throughput, etc.), but in all configurations published in Miller, et. al [10] (for HoneyBadgerBFT) and Duan, et al. [7] (for BEAT), the Hashgraph algorithm equals or exceeds the published performance figures [4]. A frequently cited throughput goal is to equal the Visa credit-card network. According to Visa's published figures, Hashgraph can

# Too much impact?

- Patent your work
- Send the patent around
- Ask companies to cite your patented work (ideally in public)

# Libra, 2019

arXiv:1803.05069v6 [cs.DC] 23 Jul 2019

HotStuff: BFT Consensus in the Lens of Blockchain

Maofan Yin<sup>1,2</sup>, Dahlia Malkhi<sup>2</sup>, Michael K. Reiter<sup>2,3</sup>, Guy Golan Gueta<sup>2</sup>, and Ittai Abraham<sup>2</sup>

<sup>1</sup>Cornell University, <sup>2</sup>VMware Research, <sup>3</sup>UNC-Chapel Hill

**Abstract**

We present HotStuff, a leader-based Byzantine fault-tolerant replication protocol for the partially synchronous model. Once network communication becomes synchronous, HotStuff enables a correct leader to drive the protocol to consensus at the pace of actual (vs. maximum) network delay—a property called *responsiveness*—and with communication complexity that is linear in the number of replicas. To our knowledge, HotStuff is the first partially synchronous BFT replication protocol exhibiting these combined properties. HotStuff is built around a novel framework that forms a bridge between classical BFT foundations and blockchains. It allows the expression of other known protocols (DLS, PBFT, Tendermint, Casper), and ours, in a common framework.

Our deployment of HotStuff over a network with over 100 replicas achieves throughput and latency comparable to that of BFT-SMaRt, while enjoying linear communication footprint during leader failover (vs. cubic with BFT-SMaRt).

**1 Introduction**

Byzantine fault tolerance (BFT) refers to the ability of a computing system to endure arbitrary (i.e., Byzantine) failures of its components while taking actions critical to the system's operation. In the context of state machine replication (SMR) [35, 47], the system as a whole provides a replicated service whose state is mirrored across  $n$  deterministic replicas. A BFT SMR protocol is used to ensure that non-faulty replicas agree on an order of execution for client-initiated service commands, despite the efforts of  $f$  Byzantine replicas. This, in turn, ensures that the  $n - f$  non-faulty replicas will run commands identically and so produce the same response for each command. As is common, we are concerned here with the partially synchronous communication model [25], whereby a known bound  $\Delta$  on message transmission holds after some unknown *global stabilization time* (GST). In this model,  $n \geq 3f + 1$  is required for non-faulty replicas to agree on the same commands in the same order (e.g., [12]) and progress can be ensured deterministically only after GST [27].

When BFT SMR protocols were originally conceived, a typical target system size was  $n = 4$  or  $n = 7$ , deployed on a local-area network. However, the renewed interest in Byzantine fault-tolerance brought about by its application to blockchains now demands solutions that can scale to much larger  $n$ . In contrast to *permissionless* blockchains such as the one that supports Bitcoin, for example, so-called *permissioned* blockchains involve a fixed set of replicas that collectively maintain an ordered ledger of commands or, in other words, that support SMR. Despite their permissioned nature, numbers of replicas in the hundreds or even thousands are envisioned (e.g., [42, 30]). Additionally, their deployment to wide-area networks requires setting  $\Delta$  to accommodate higher variability in communication delays.

**The scaling challenge.** Since the introduction of PBFT [20], the first practical BFT replication solution in the partial synchrony model, numerous BFT solutions were built around its core two-phase paradigm. The practical aspect is that a stable leader can drive a consensus decision in just two rounds of message exchanges. The first phase guarantees proposal uniqueness through the formation of a quorum certificate (QC) consisting of  $(n - f)$  votes. The second phase guarantees that the next leader can convince replicas to vote for a safe proposal.

The algorithm for a new leader to collect information and propose it to replicas—called a *view-change*—is the epicenter of replication. Unfortunately, view-change based on the two-phase paradigm is far from simple [38], is bug-prone [4], and incurs a significant communication penalty for even moderate system sizes. It requires the new leader to relay information from  $(n - f)$  replicas, each reporting its own highest known QC. Even counting just

1

## HotStuff

✓ Linear

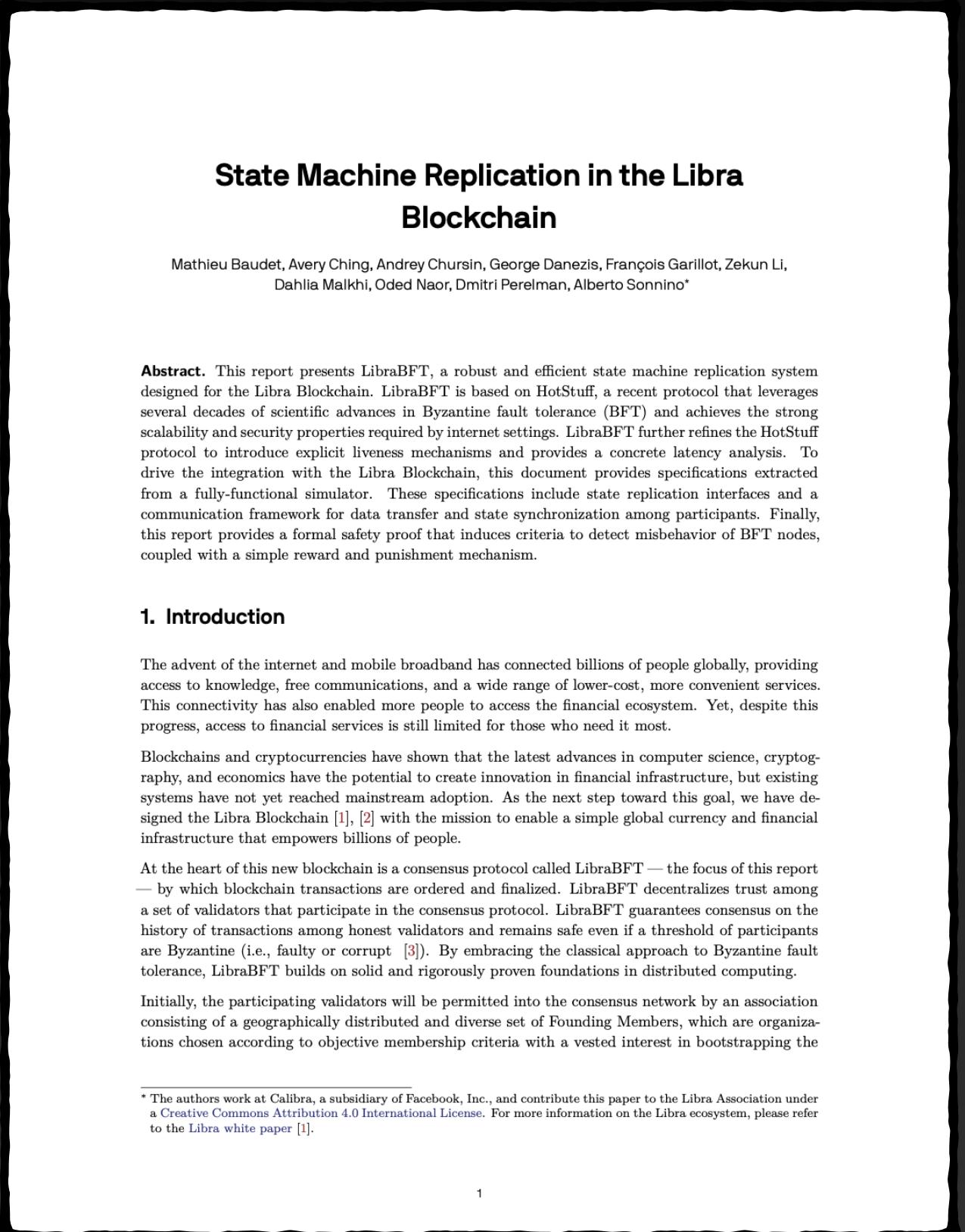
✓ Clearly isolated components

## HashGraph

✗ Impossible to garbage collect

✗ Unclear block synchroniser

# The first 6 months...



## SMR in the Libra Blockchain

- The LibraBFT/DiemBFT pacemaker
- Codesign the pacemaker with the rest

### State Machine Replication in the Libra Blockchain

Mathieu Baudet, Avery Ching, Andrey Chursin, George Danezis, François Garillot, Zekun Li, Dahlia Malkhi, Oded Naor, Dmitri Perelman, Alberto Sonnino\*

**Abstract.** This report presents LibraBFT, a robust and efficient state machine replication system designed for the Libra Blockchain. LibraBFT is based on HotStuff, a recent protocol that leverages several decades of scientific advances in Byzantine fault tolerance (BFT) and achieves the strong scalability and security properties required by internet settings. LibraBFT further refines the HotStuff protocol to introduce explicit liveness mechanisms and provides a concrete latency analysis. To drive the integration with the Libra Blockchain, this document provides specifications extracted from a fully-functional simulator. These specifications include state replication interfaces and a communication framework for data transfer and state synchronization among participants. Finally, this report provides a formal safety proof that induces criteria to detect misbehavior of BFT nodes, coupled with a simple reward and punishment mechanism.

#### 1. Introduction

The advent of the internet and mobile broadband has connected billions of people globally, providing access to knowledge, free communications, and a wide range of lower-cost, more convenient services. This connectivity has also enabled more people to access the financial ecosystem. Yet, despite this progress, access to financial services is still limited for those who need it most.

Blockchains and cryptocurrencies have shown that the latest advances in computer science, cryptography, and economics have the potential to create innovation in financial infrastructure, but existing systems have not yet reached mainstream adoption. As the next step toward this goal, we have designed the Libra Blockchain [1], [2] with the mission to enable a simple global currency and financial infrastructure that empowers billions of people.

At the heart of this new blockchain is a consensus protocol called LibraBFT — the focus of this report — by which blockchain transactions are ordered and finalized. LibraBFT decentralizes trust among a set of validators that participate in the consensus protocol. LibraBFT guarantees consensus on the history of transactions among honest validators and remains safe even if a threshold of participants are Byzantine (i.e., faulty or corrupt [3]). By embracing the classical approach to Byzantine fault tolerance, LibraBFT builds on solid and rigorously proven foundations in distributed computing.

Initially, the participating validators will be permitted into the consensus network by an association consisting of a geographically distributed and diverse set of Founding Members, which are organizations chosen according to objective membership criteria with a vested interest in bootstrapping the

\* The authors work at Calibra, a subsidiary of Facebook, Inc., and contribute this paper to the Libra Association under a Creative Commons Attribution 4.0 International License. For more information on the Libra ecosystem, please refer to the Libra white paper [1].

# Research Questions

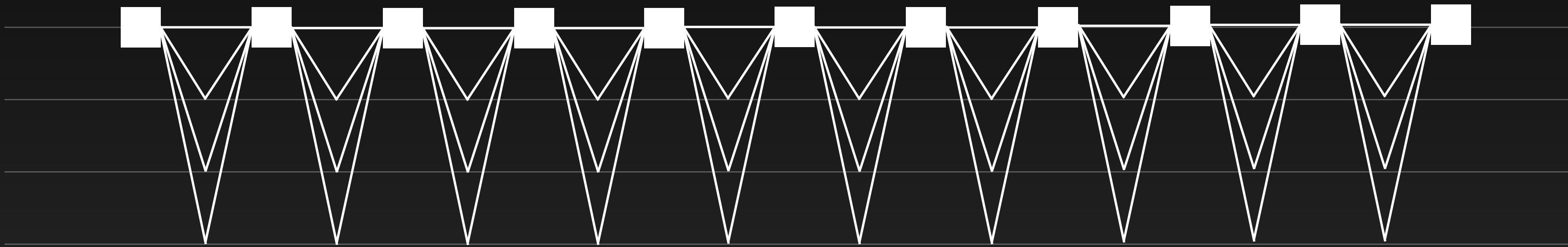
1. Network model?

# Lessons Learned

1. Modularisation is a design strategy

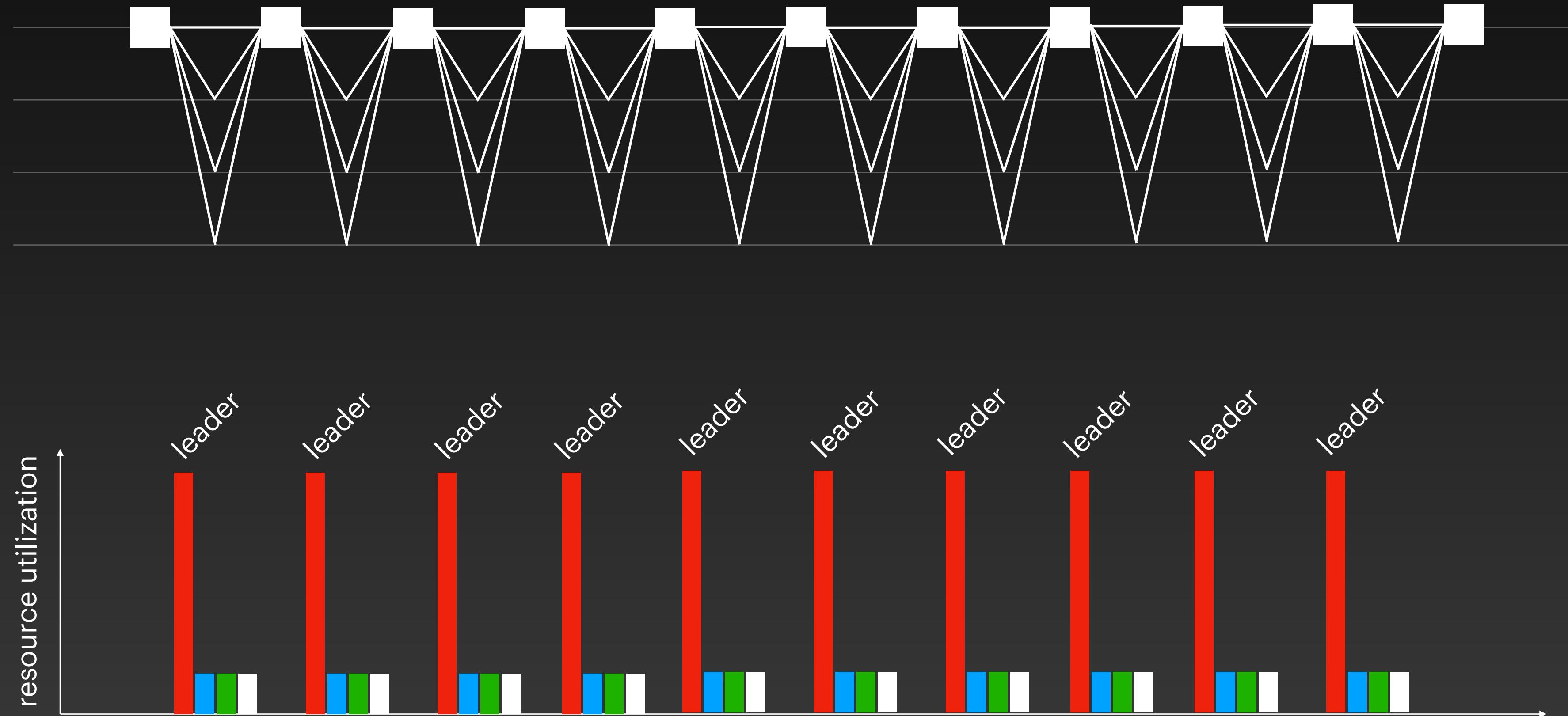
# HotStuff

## Typical leader-based protocols



# Naive Implementation

## Uneven resource utilisation



# Research Questions

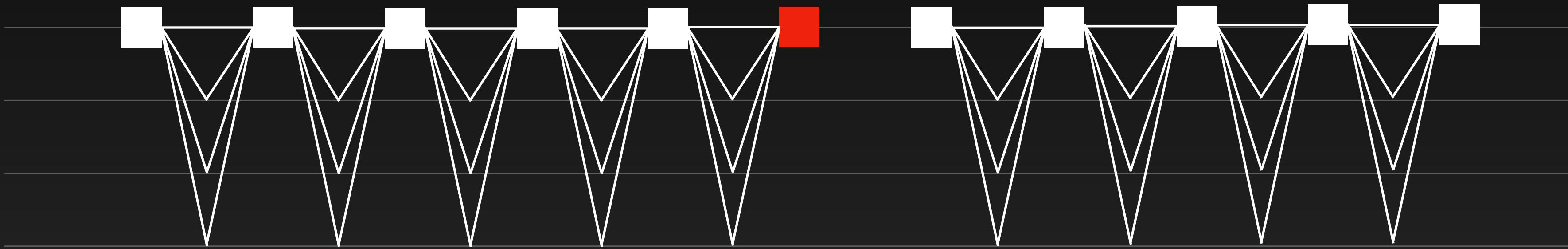
1. Network model?

# Lessons Learned

1. Modularisation is a design strategy
2. Tasks-threads allocation

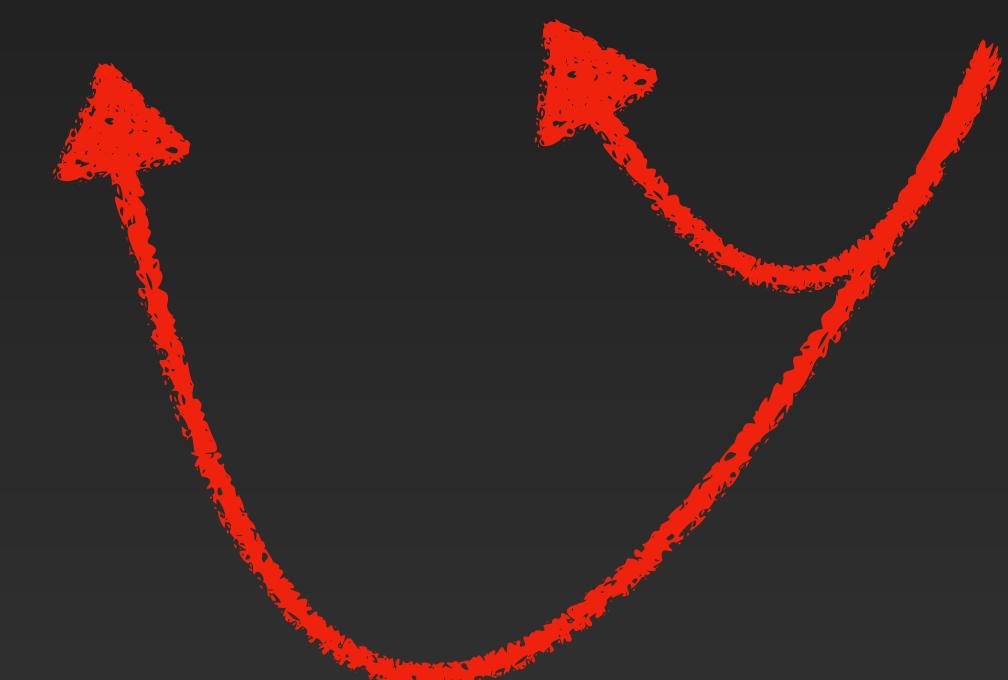
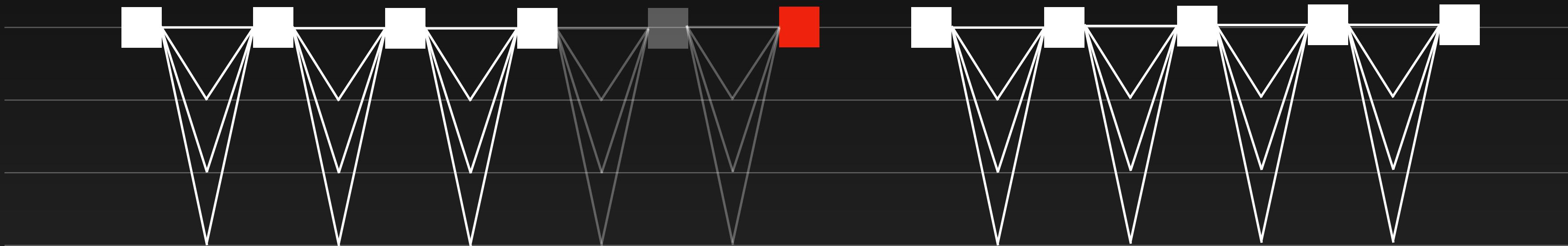
# Leader-Driven Consensus

## Fragility to faults and asynchrony

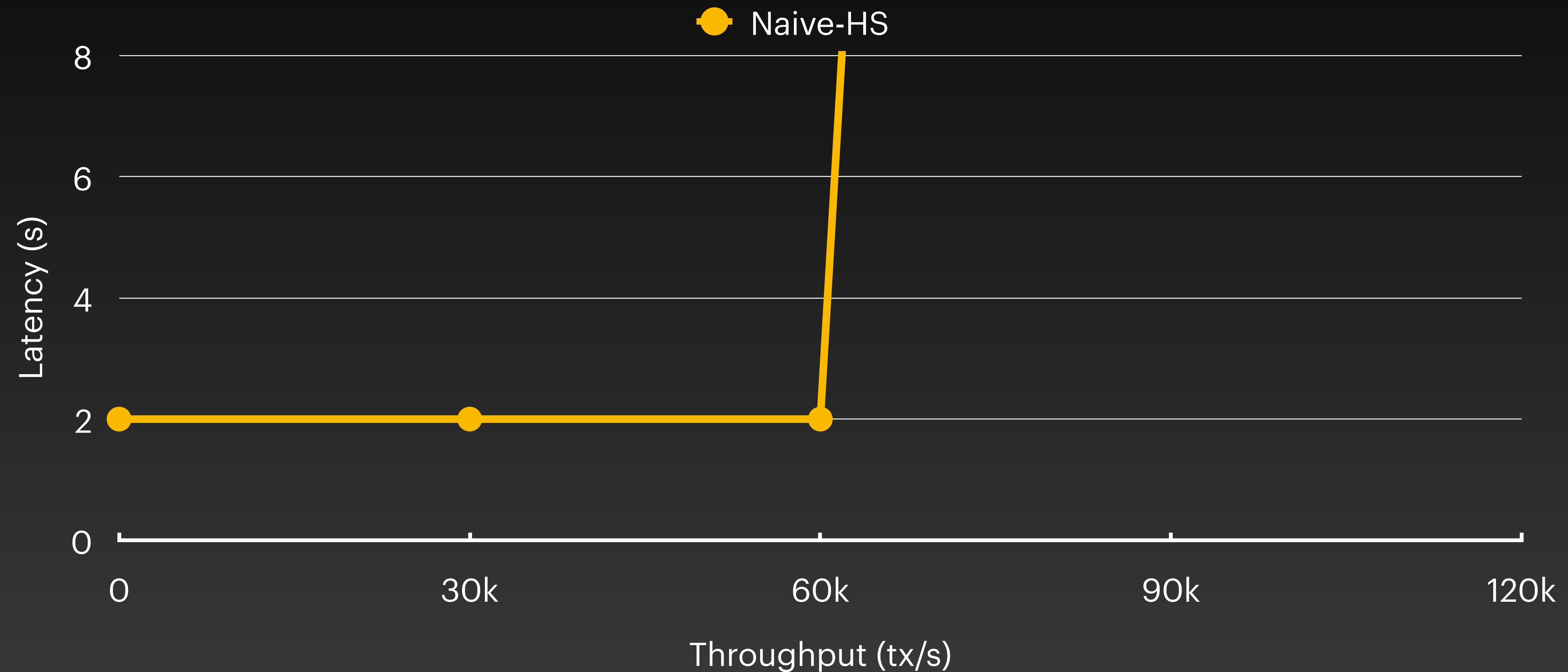


# Leader-Driven Consensus

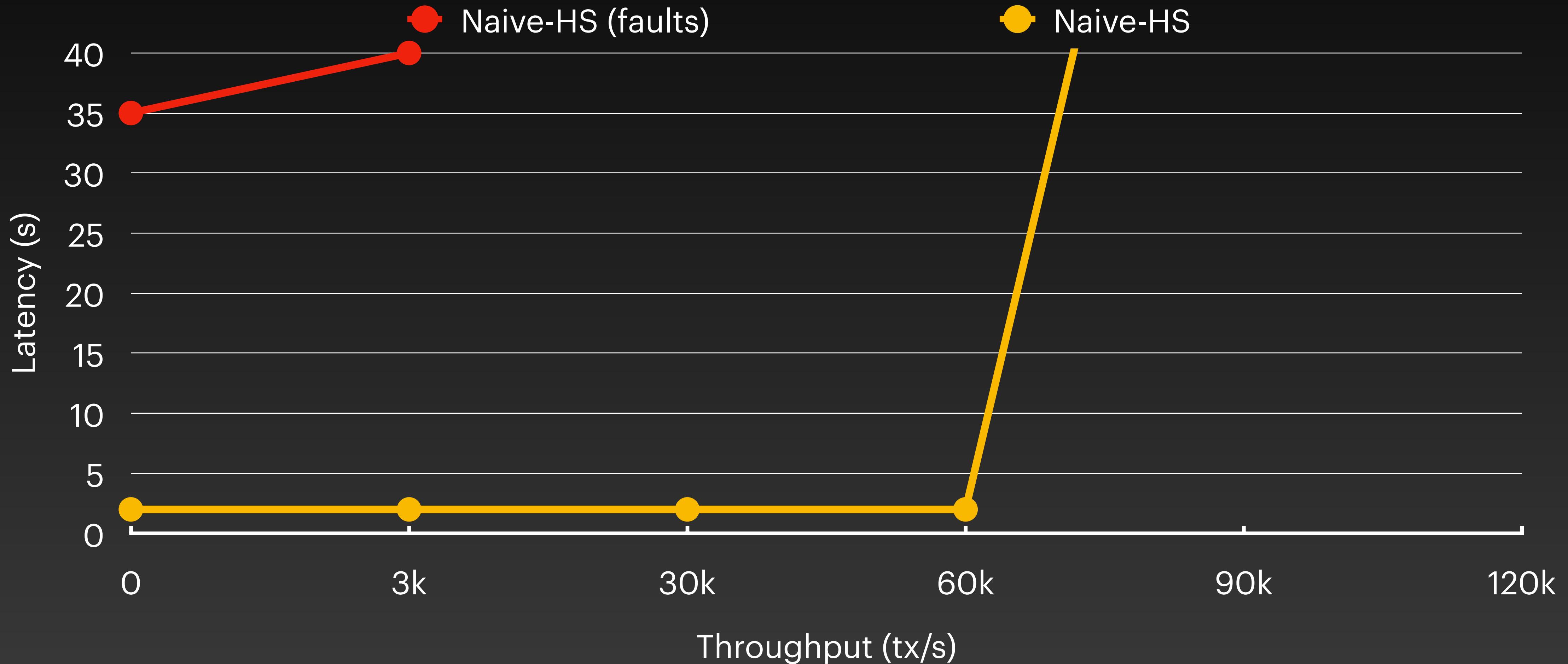
## Fragility to faults and asynchrony



# Performance



# Performance



# Research Questions

1. Network model?

# Lessons Learned

1. Modularisation is a design strategy
2. Tasks-threads allocation
3. Benchmark early

# Libra, 2019

HotStuff: BFT Consensus in the Lens of Blockchain

Maofan Yin<sup>1,2</sup>, Dahlia Malkhi<sup>2</sup>, Michael K. Reiter<sup>2,3</sup>, Guy Golan Gueta<sup>2</sup>, and Ittai Abraham<sup>2</sup>

<sup>1</sup>Cornell University, <sup>2</sup>VMware Research, <sup>3</sup>UNC-Chapel Hill

**Abstract**

We present HotStuff, a leader-based Byzantine fault-tolerant replication protocol for the partially synchronous model. Once network communication becomes synchronous, HotStuff enables a correct leader to drive the protocol to consensus at the pace of actual (vs. maximum) network delay—a property called *responsiveness*—and with communication complexity that is linear in the number of replicas. To our knowledge, HotStuff is the first partially synchronous BFT replication protocol exhibiting these combined properties. HotStuff is built around a novel framework that forms a bridge between classical BFT foundations and blockchains. It allows the expression of other known protocols (DLS, PBFT, Tendermint, Casper), and ours, in a common framework.

Our deployment of HotStuff over a network with over 100 replicas achieves throughput and latency comparable to that of BFT-SMaRt, while enjoying linear communication footprint during leader failure (vs. cubic with BFT-SMaRt).

**1 Introduction**

Byzantine fault tolerance (BFT) refers to the ability of a computing system to endure arbitrary (i.e., Byzantine) failures of its components while taking actions critical to the system's operation. In the context of state machine replication (SMR) [35, 47], the system as a whole provides a replicated service whose state is mirrored across  $n$  deterministic replicas. A BFT SMR protocol is used to ensure that non-faulty replicas agree on an order of execution for client-initiated service commands, despite the efforts of  $f$  Byzantine replicas. This, in turn, ensures that the  $n - f$  non-faulty replicas will run commands identically and so produce the same response for each command. As is common, we are concerned here with the partially synchronous communication model [25], whereby a known bound  $\Delta$  on message transmission holds after some unknown *global stabilization time* (GST). In this model,  $n \geq 3f + 1$  is required for non-faulty replicas to agree on the same commands in the same order (e.g., [12]) and progress can be ensured deterministically only after GST [27].

When BFT SMR protocols were originally conceived, a typical target system size was  $n = 4$  or  $n = 7$ , deployed on a local-area network. However, the renewed interest in Byzantine fault-tolerance brought about by its application to blockchains now demands solutions that can scale to much larger  $n$ . In contrast to *permissionless* blockchains such as the one that supports Bitcoin, for example, so-called *permissioned* blockchains involve a fixed set of replicas that collectively maintain an ordered ledger of commands or, in other words, that support SMR. Despite their permissioned nature, numbers of replicas in the hundreds or even thousands are envisioned (e.g., [42, 30]). Additionally, their deployment to wide-area networks requires setting  $\Delta$  to accommodate higher variability in communication delays.

**The scaling challenge.** Since the introduction of PBFT [20], the first practical BFT replication solution in the partial synchrony model, numerous BFT solutions were built around its core two-phase paradigm. The practical aspect is that a stable leader can drive a consensus decision in just two rounds of message exchanges. The first phase guarantees proposal uniqueness through the formation of a quorum certificate (QC) consisting of  $(n - f)$  votes. The second phase guarantees that the next leader can convince replicas to vote for a safe proposal.

The algorithm for a new leader to collect information and propose it to replicas—called a *view-change*—is the epicenter of replication. Unfortunately, view-change based on the two-phase paradigm is far from simple [38], is bug-prone [4], and incurs a significant communication penalty for even moderate system sizes. It requires the new leader to relay information from  $(n - f)$  replicas, each reporting its own highest known QC. Even counting just

arXiv:1803.05069v6 [cs.DC] 23 Jul 2019

1

## HotStuff (naive mempool)

- Linear
- Clearly isolated components
- Uneven resource utilisation
- Fragile to faults and asynchrony
- Unspecified components (pacemaker)

# Libra, 2021

**Narwhal and Tusk: A DAG-based Mempool and Efficient BFT Consensus**

George Danezis  
Mysten Labs & UCL

Alberto Sonnino  
Mysten Labs

**Abstract**  
We propose separating the task of reliable transaction dissemination from transaction ordering, to enable high-performance Byzantine fault-tolerant quorum-based consensus. We design and evaluate a mempool protocol, Narwhal, specializing in high-throughput reliable dissemination and storage of causal histories of transactions. Narwhal tolerates an asynchronous network and maintains high performance despite failures. Narwhal is designed to easily scale-out using multiple workers at each validator, and we demonstrate that there is no foreseeable limit to the throughput we can achieve.

Composing Narwhal with a partially synchronous consensus protocol (Narwhal-HotStuff) yields significantly better throughput even in the presence of faults or intermittent loss of liveness due to asynchrony. However, loss of liveness can result in higher latency. To achieve overall good performance when faults occur we design Tusk, a zero-message overhead asynchronous consensus protocol, to work with Narwhal. We demonstrate its high performance under a variety of configurations and faults.

As a summary of results, on a WAN, Narwhal-HotStuff achieves over 130,000 tx/sec at less than 2-sec latency compared with 1,800 tx/sec at 1-sec latency for HotStuff. Additional workers increase throughput linearly to 600,000 tx/sec without any latency increase. Tusk achieves 160,000 tx/sec with about 3 seconds latency. Under faults, both protocols maintain high throughput, but Narwhal-HotStuff suffers from increased latency.

**CCS Concepts:** Security and privacy → Distributed systems security.

**Keywords:** Consensus protocol, Byzantine Fault Tolerant

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).  
*EuroSys '22, April 5–8, 2022, Rennes, France*  
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.  
ACM ISBN 978-1-4503-9162-7/22/04... \$15.00  
<https://doi.org/10.1145/3492321.3519594>

**ACM Reference Format:**  
George Danezis, Lefteris Kokoris-Kogias, Alberto Sonnino, and Alexander Spiegelman. 2022. Narwhal and Tusk: A DAG-based Mempool and Efficient BFT Consensus . In *Seventeenth European Conference on Computer Systems (EuroSys '22), April 5–8, 2022, Rennes, France*. ACM, New York, NY, USA, 17 pages. <https://doi.org/10.1145/3492321.3519594>

**1 Introduction**  
Byzantine consensus protocols [15, 19, 21] and the state machine replication paradigm [13] for building reliable distributed systems have been studied for over 40 years. However, with the rise in popularity of blockchains there has been a renewed interest in engineering high-performance consensus protocols. Specifically, to improve on Bitcoin's [33] throughput of only 4 tx/sec early works [29] suggested committee based consensus protocols. For higher throughput and lower latency committee-based protocols are required, and are now becoming the norm in proof-of-stake designs.

Existing approaches to increasing the performance of distributed ledgers focus on creating lower-cost consensus algorithms culminating with HotStuff [38], which achieves linear message complexity in the partially synchronous setting. To achieve this, HotStuff leverages a leader who collects, aggregates, and broadcasts the messages of other validators. However, theoretical message complexity should not be the only optimization target. More specifically:

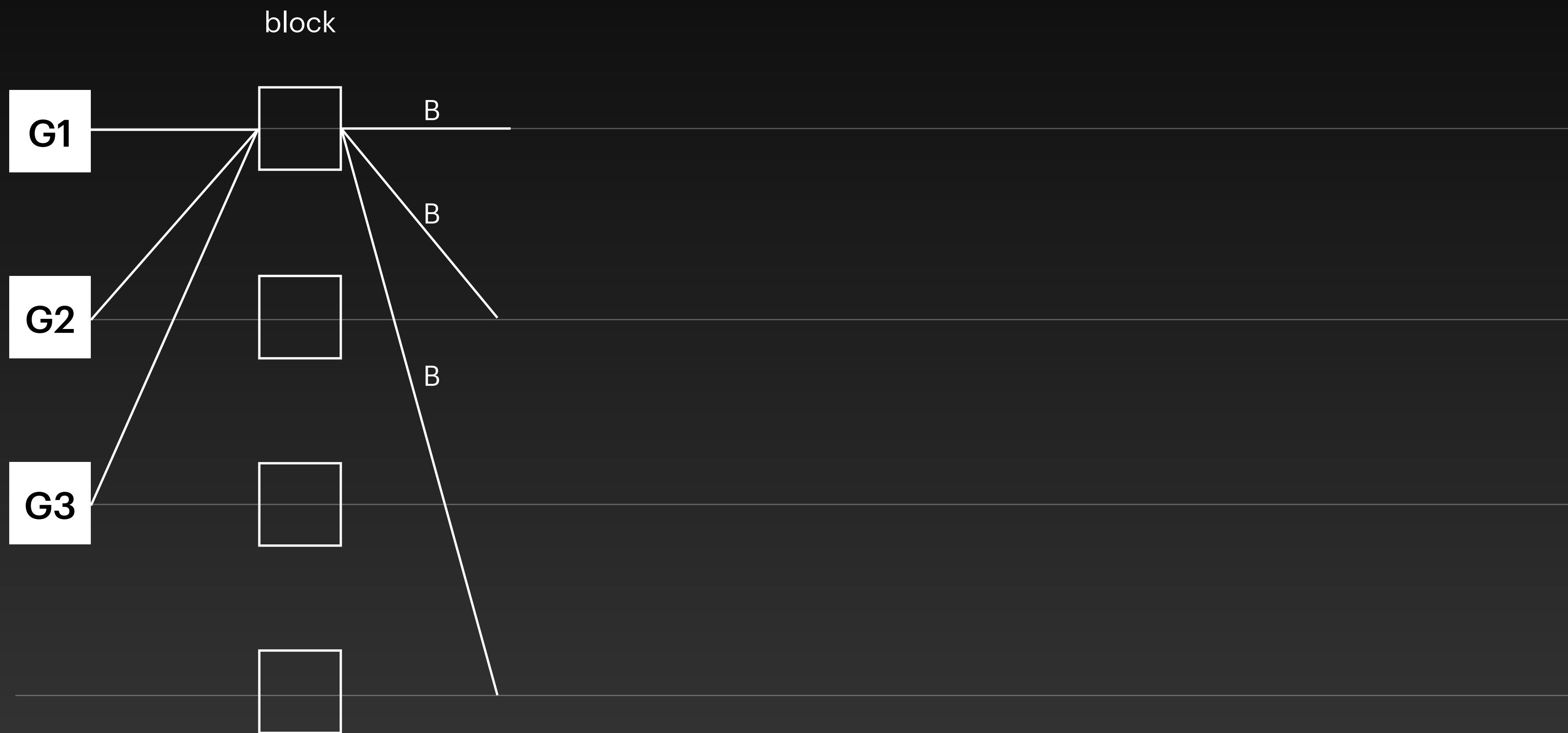
- Any (partially-synchronous) protocol that minimizes overall message number, but relies on a leader to produce proposals and coordinate consensus, fails to capture the high load this imposes on the leader who inevitably becomes a bottleneck.
- Message complexity counts the number of *metadata* messages (e.g., votes, signatures, hashes) which take minimal bandwidth compared to the dissemination of bulk transaction data (blocks). Since blocks are orders of magnitude larger (10MB) than a typical consensus message (100B), the asymptotic message complexity is practically amortized for fixed mid-size committees (up to ~ 50 nodes).

Additionally, consensus protocols have grouped a lot of functions into a monolithic protocol. In a typical distributed

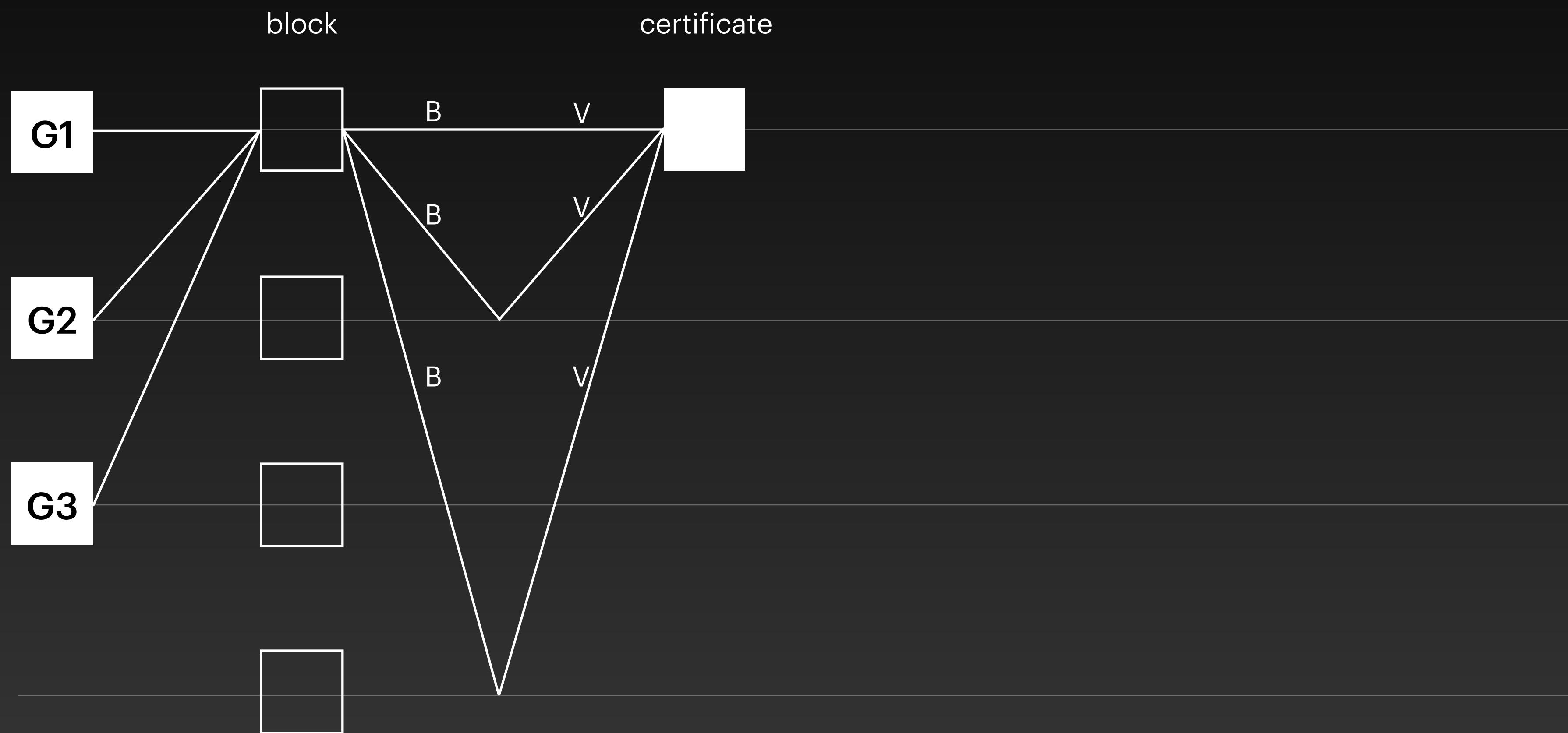
## Narwhal

- Quadratic but even resource utilisation
- Separation between consensus and data dissemination

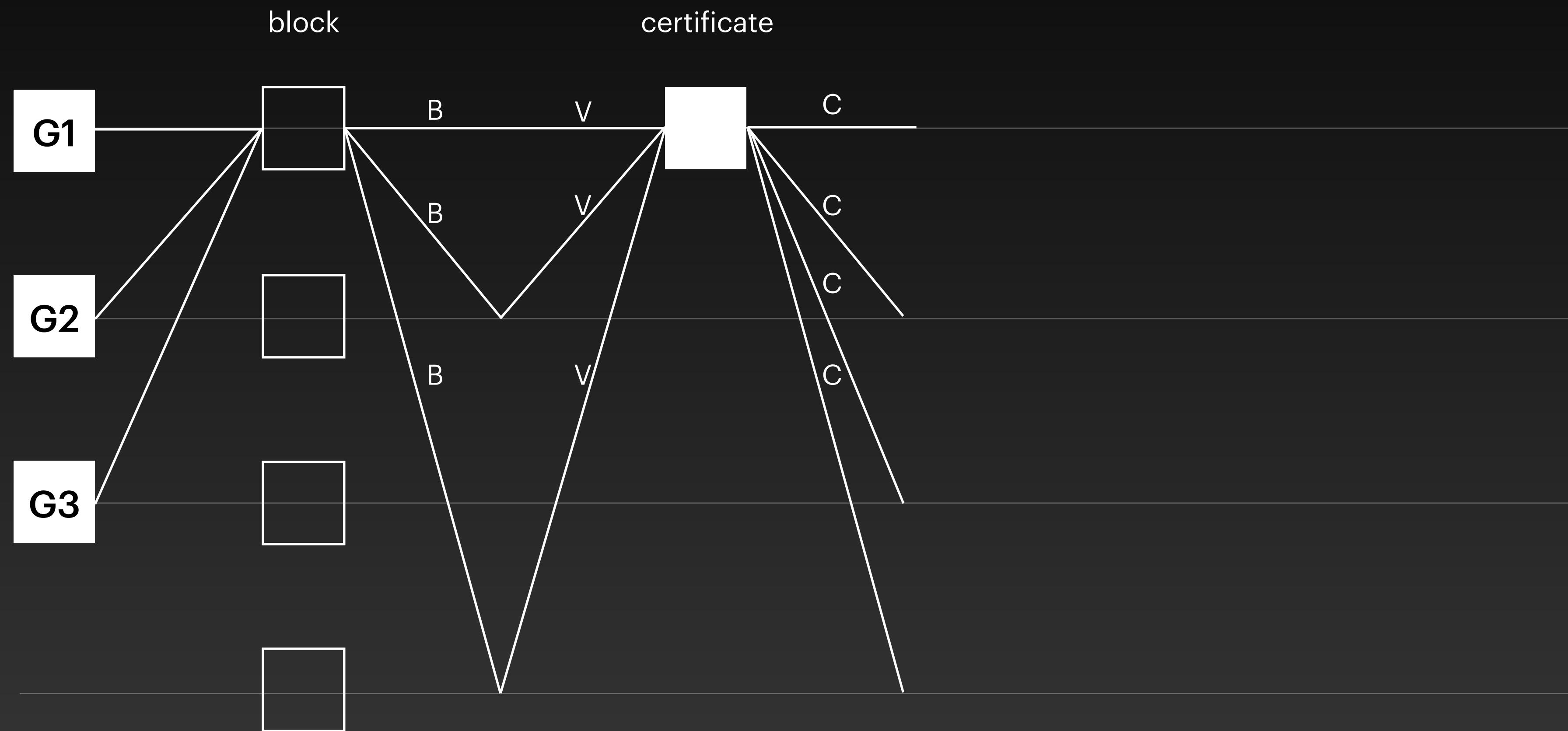
# Narwhal



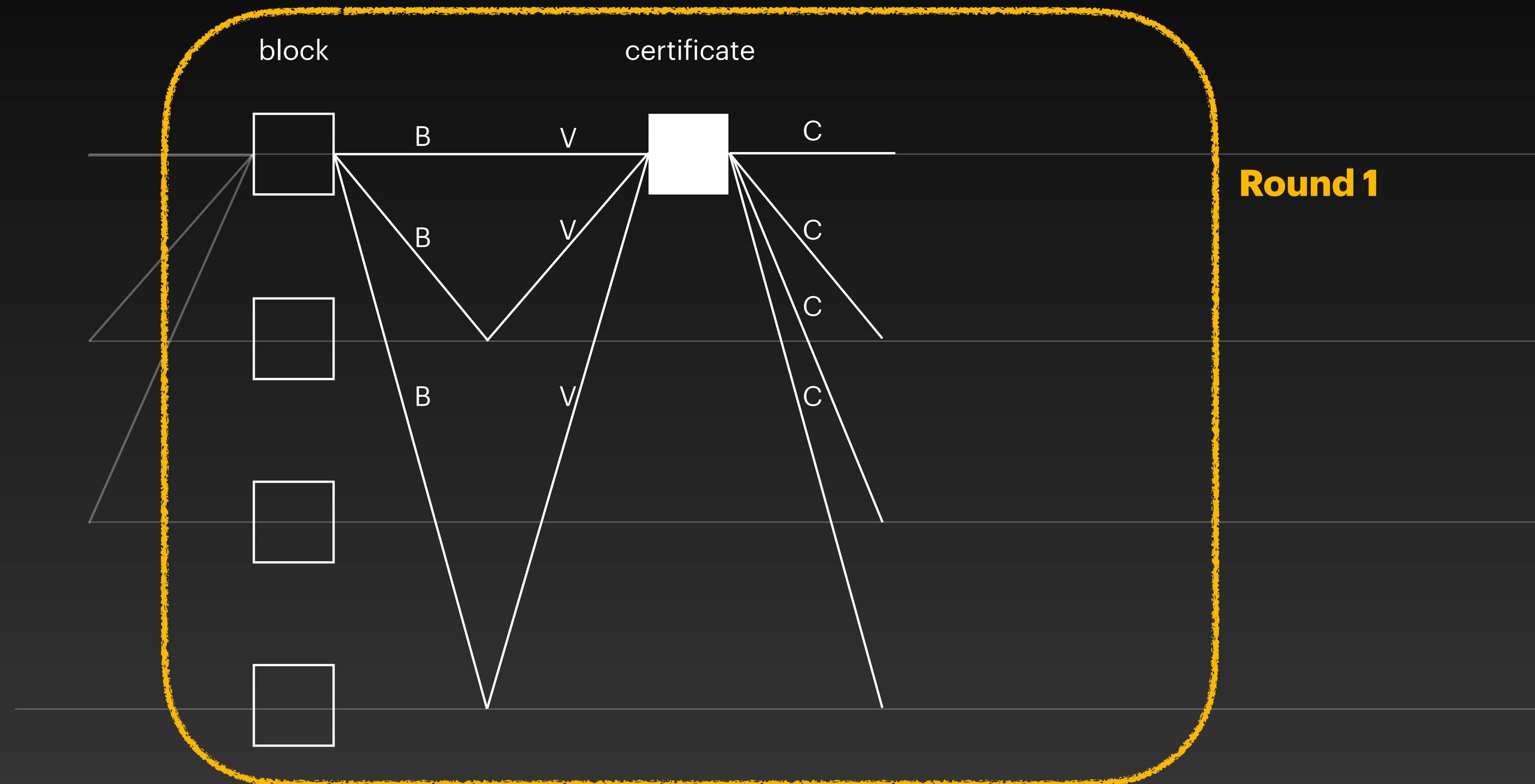
# Narwhal



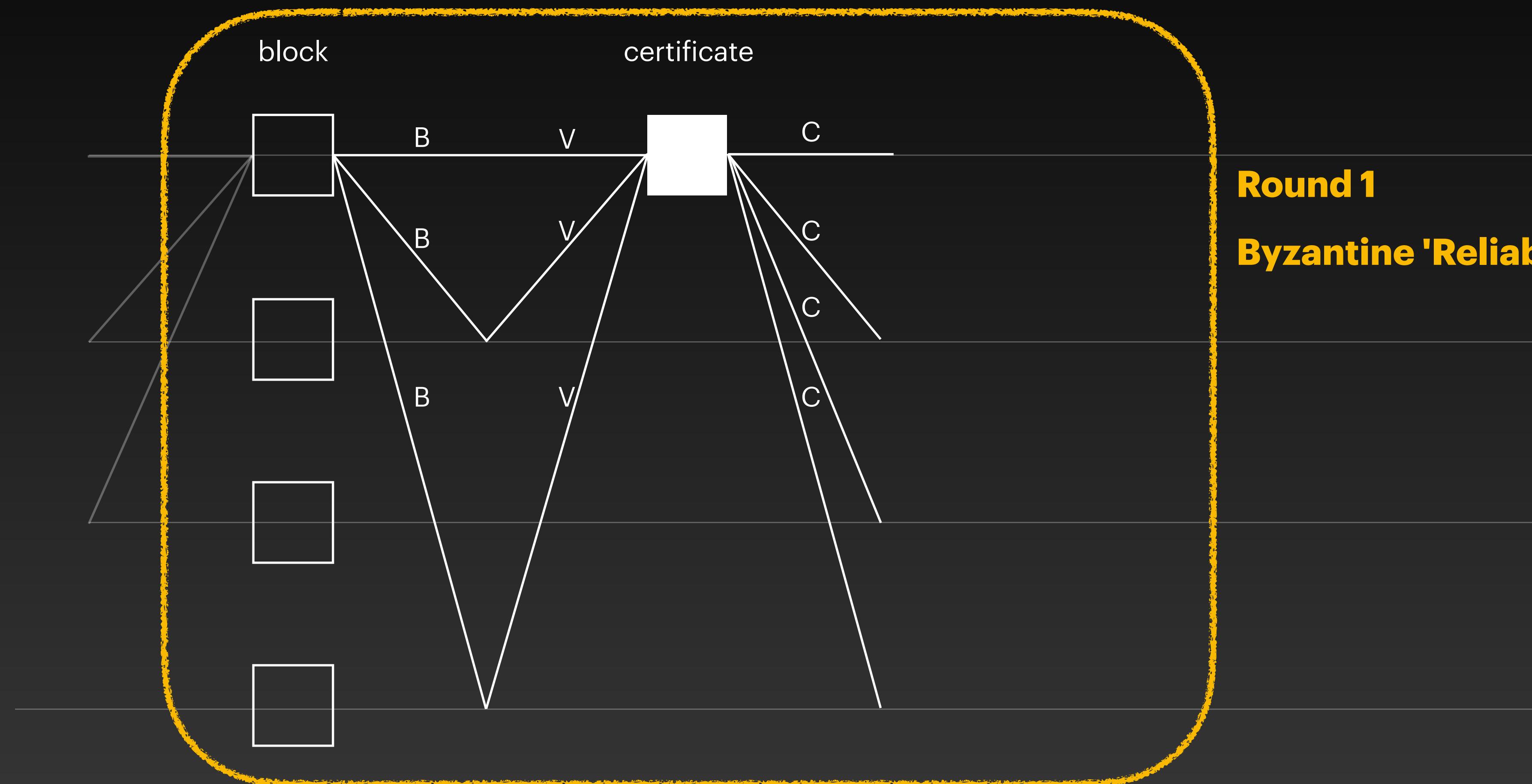
# Narwhal



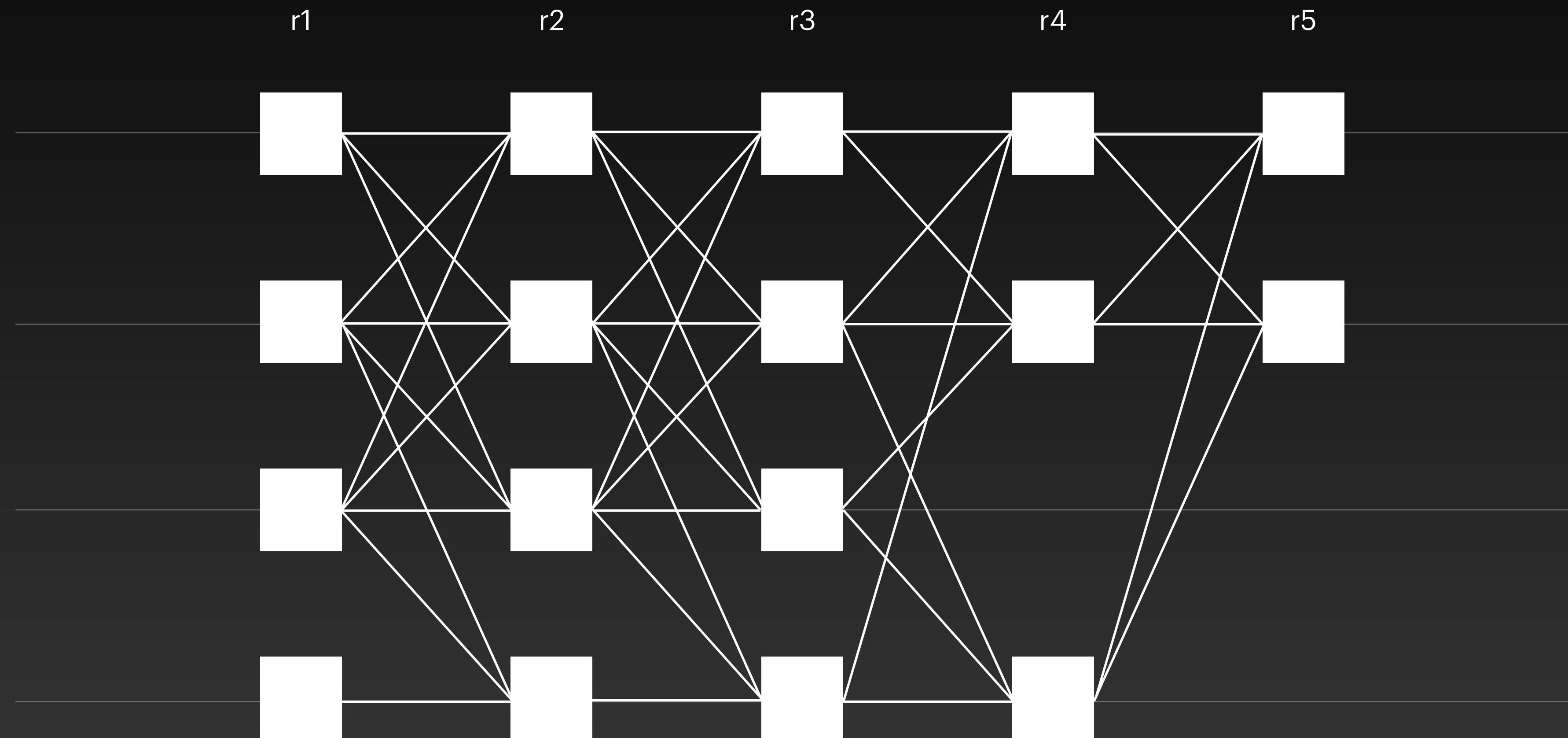
# Narwhal



# Narwhal



# Narwhal



# Research Questions

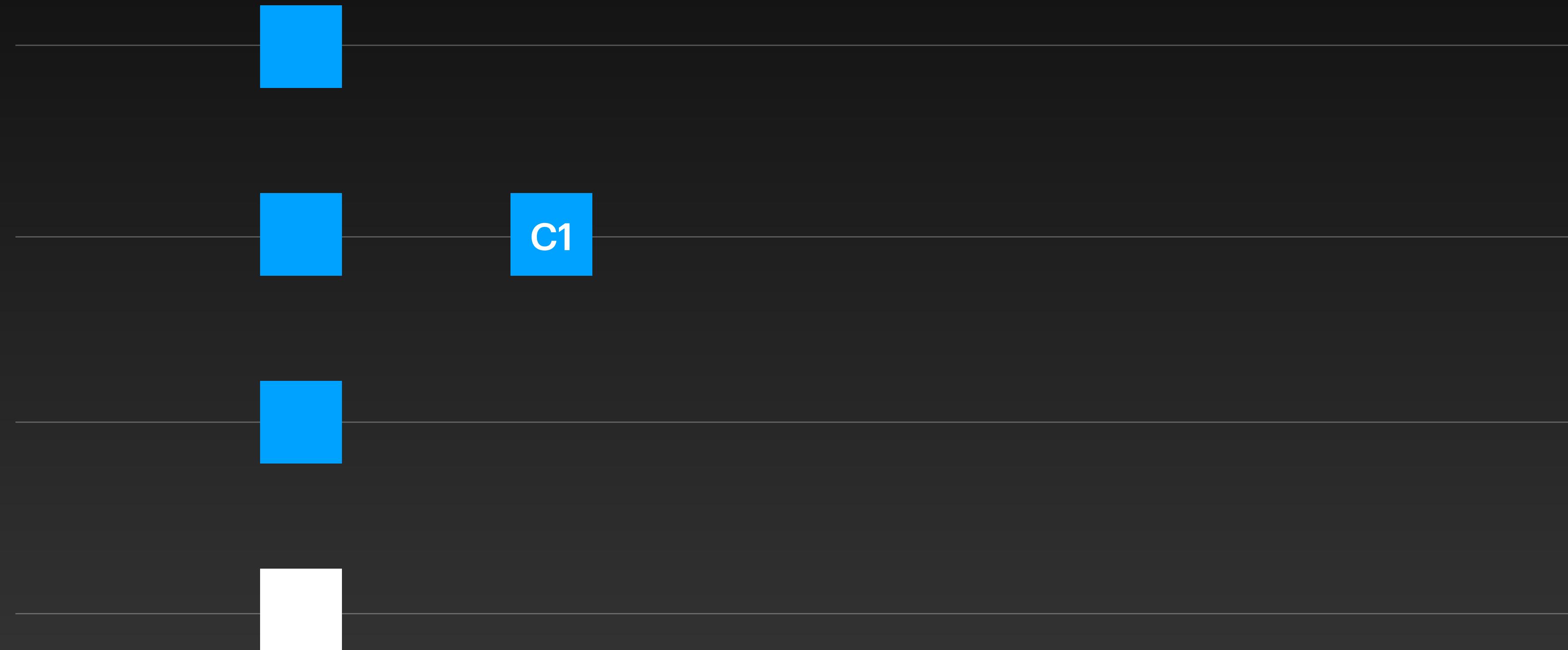
1. Network model?

# Lessons Learned

1. Modularisation is a design strategy
2. Tasks-threads allocation
3. Benchmark early
4. Codesign with mem. and storage

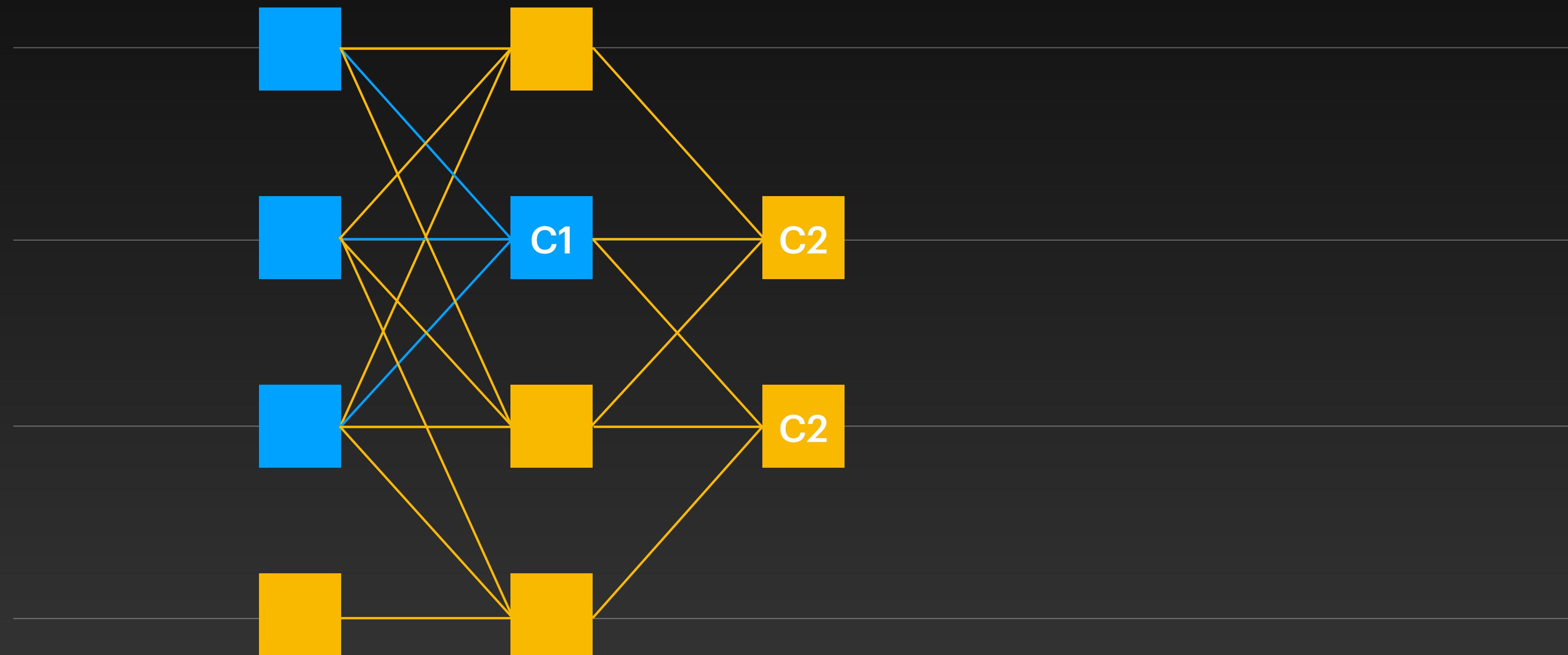
# HotStuff on Narwhal

## Enhanced commit rule



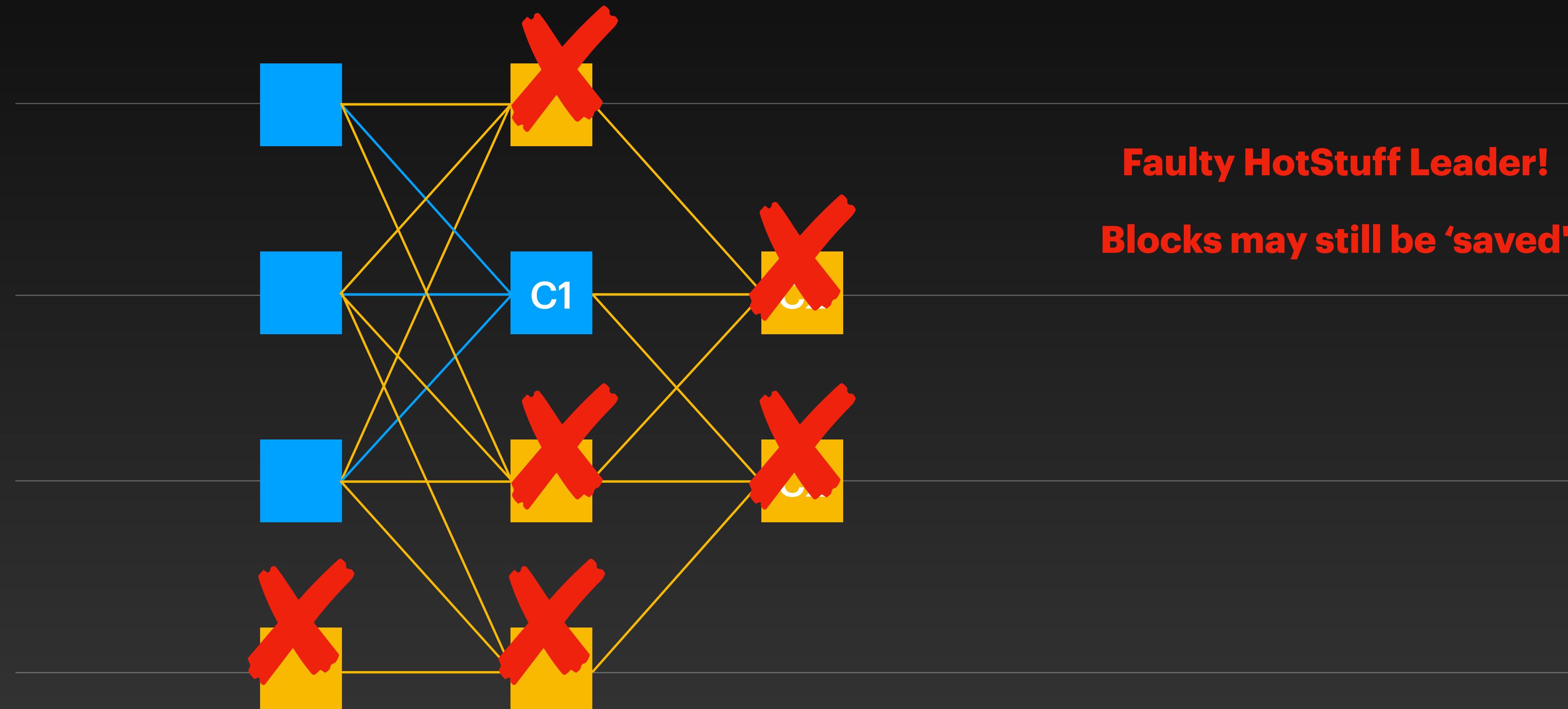
# HotStuff on Narwhal

## Enhanced commit rule



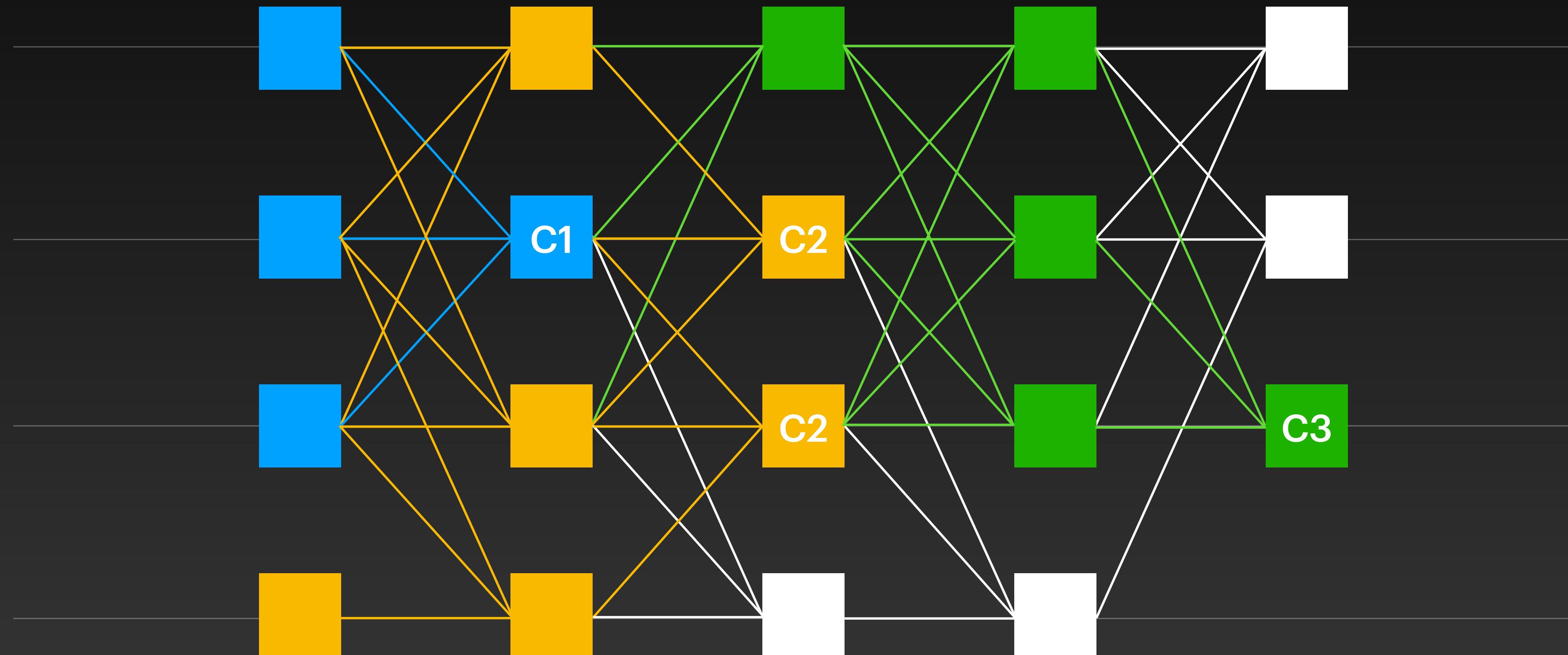
# HotStuff on Narwhal

## Enhanced commit rule



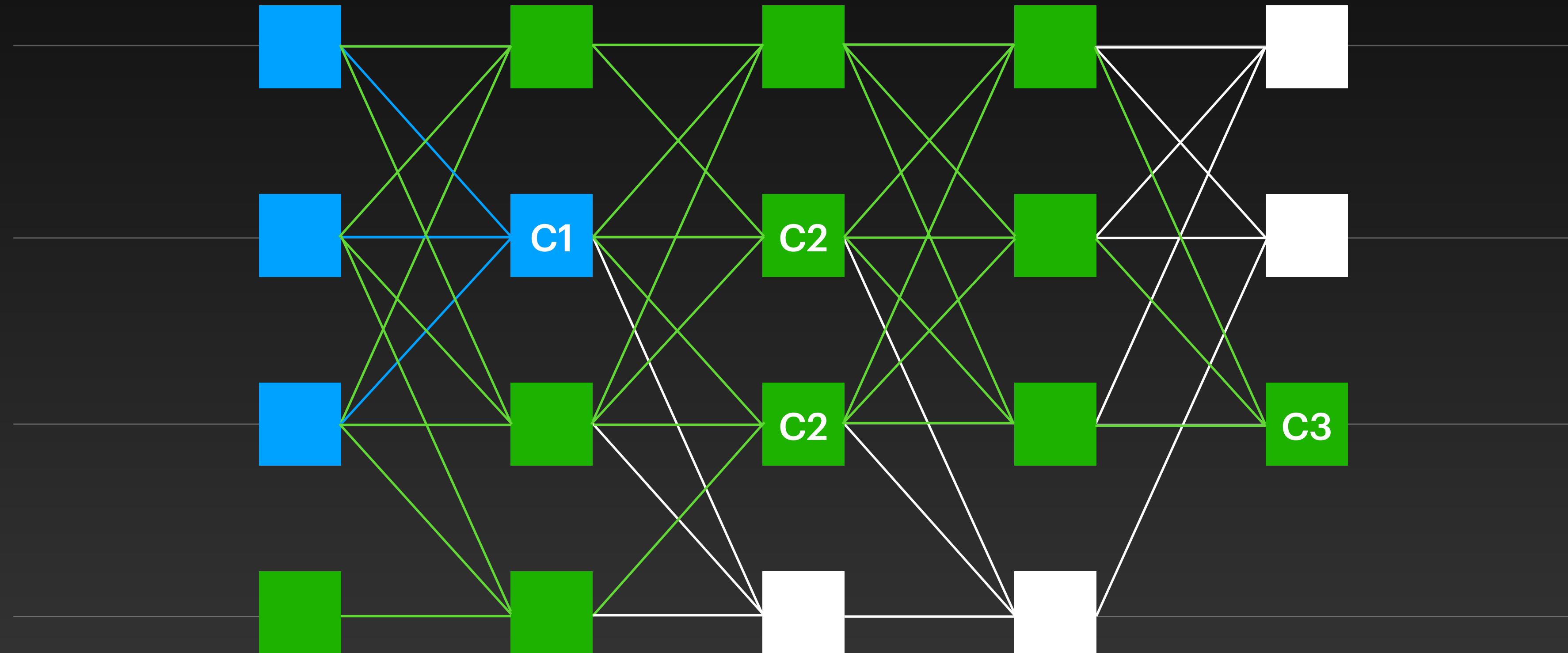
# HotStuff on Narwhal

## Enhanced commit rule

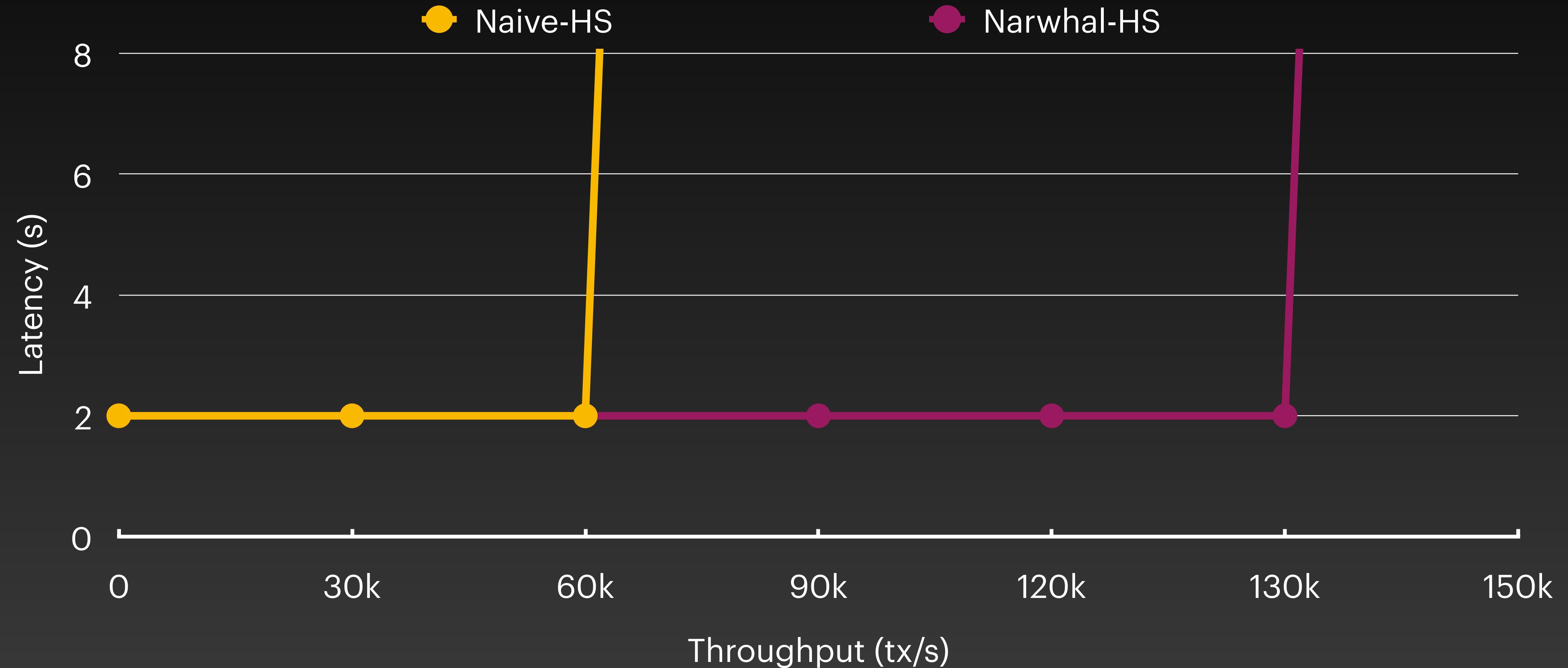


# HotStuff on Narwhal

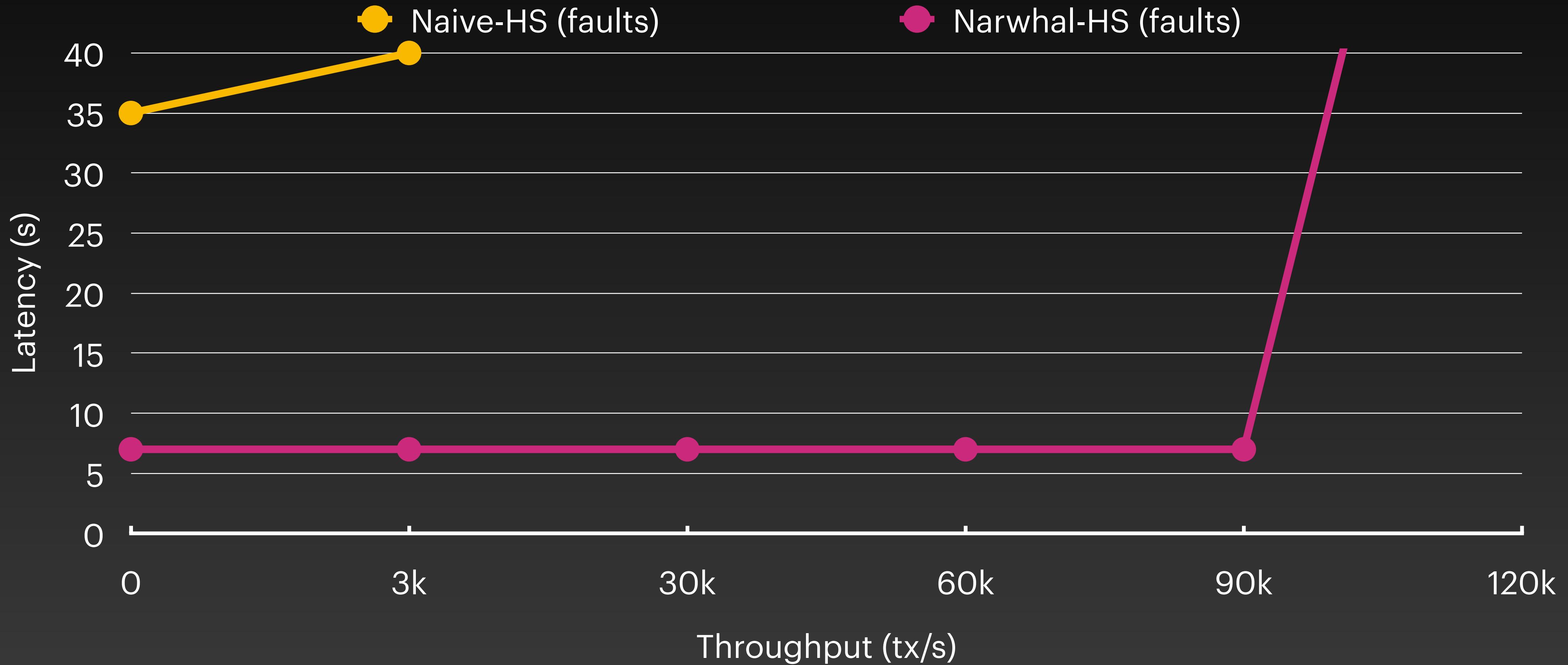
## Enhanced commit rule



# Performance

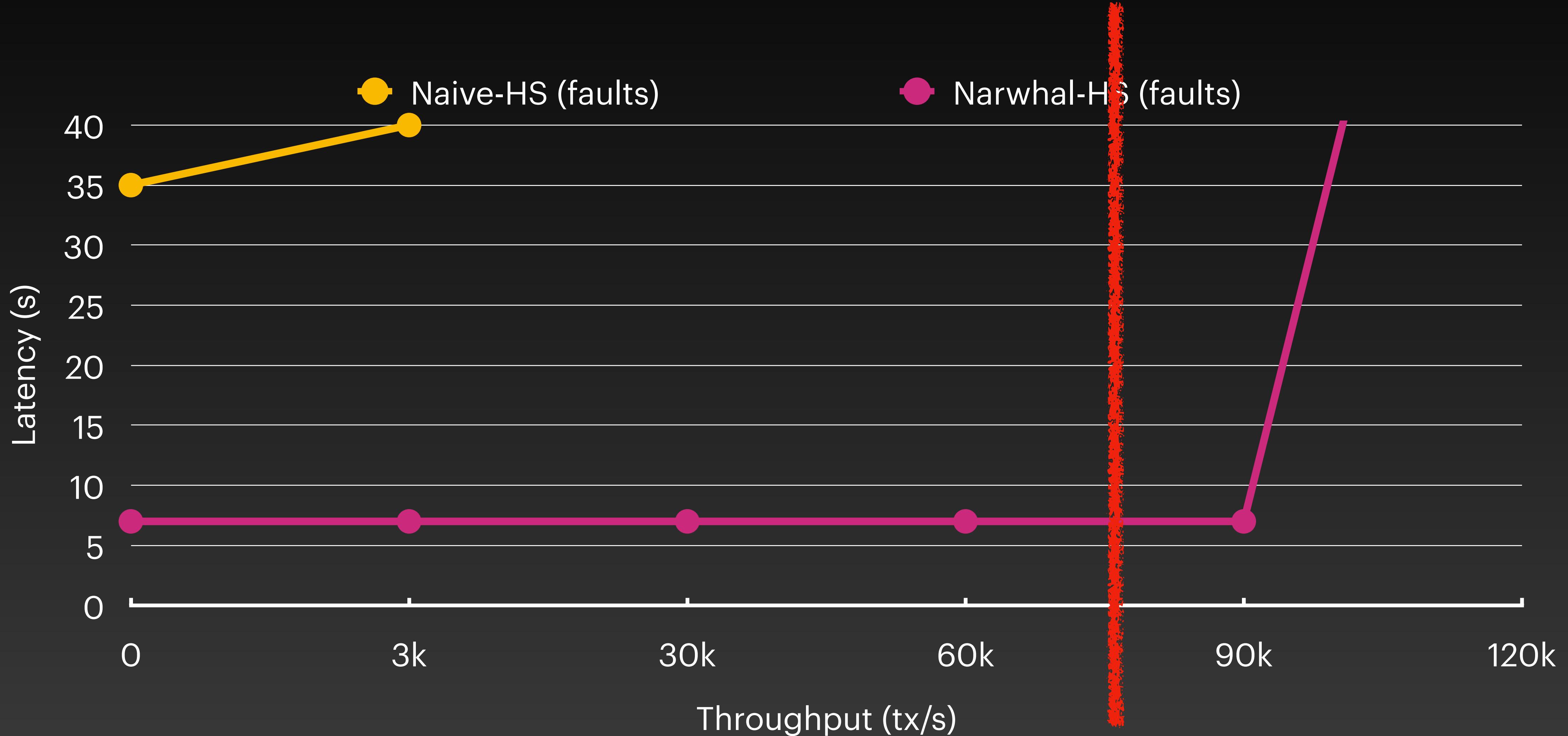


# Performance



# Performance

visa+mastercard



# Libra, 2021

**Narwhal and Tusk: A DAG-based Mempool and Efficient BFT Consensus**

George Danezis  
Mysten Labs & UCL

Alberto Sonnino  
Mysten Labs

**Abstract**  
We propose separating the task of reliable transaction dissemination from transaction ordering, to enable high-performance Byzantine fault-tolerant quorum-based consensus. We design and evaluate a mempool protocol, Narwhal, specializing in high-throughput reliable dissemination and storage of causal histories of transactions. Narwhal tolerates an asynchronous network and maintains high performance despite failures. Narwhal is designed to easily scale-out using multiple workers at each validator, and we demonstrate that there is no foreseeable limit to the throughput we can achieve.

Composing Narwhal with a partially synchronous consensus protocol (Narwhal-HotStuff) yields significantly better throughput even in the presence of faults or intermittent loss of liveness due to asynchrony. However, loss of liveness can result in higher latency. To achieve overall good performance when faults occur we design Tusk, a zero-message overhead asynchronous consensus protocol, to work with Narwhal. We demonstrate its high performance under a variety of configurations and faults.

As a summary of results, on a WAN, Narwhal-HotStuff achieves over 130,000 tx/sec at less than 2-sec latency compared with 1,800 tx/sec at 1-sec latency for HotStuff. Additional workers increase throughput linearly to 600,000 tx/sec without any latency increase. Tusk achieves 160,000 tx/sec with about 3 seconds latency. Under faults, both protocols maintain high throughput, but Narwhal-HotStuff suffers from increased latency.

**CCS Concepts:** Security and privacy → Distributed systems security.

**Keywords:** Consensus protocol, Byzantine Fault Tolerant

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).  
*EuroSys '22, April 5–8, 2022, RENNES, France*  
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.  
ACM ISBN 978-1-4503-9162-7/22/04... \$15.00  
<https://doi.org/10.1145/3492321.3519594>

**ACM Reference Format:**  
George Danezis, Lefteris Kokoris-Kogias, Alberto Sonnino, and Alexander Spiegelman. 2022. Narwhal and Tusk: A DAG-based Mempool and Efficient BFT Consensus . In *Seventeenth European Conference on Computer Systems (EuroSys '22), April 5–8, 2022, RENNES, France*. ACM, New York, NY, USA, 17 pages. <https://doi.org/10.1145/3492321.3519594>

**1 Introduction**  
Byzantine consensus protocols [15, 19, 21] and the state machine replication paradigm [13] for building reliable distributed systems have been studied for over 40 years. However, with the rise in popularity of blockchains there has been a renewed interest in engineering high-performance consensus protocols. Specifically, to improve on Bitcoin's [33] throughput of only 4 tx/sec early works [29] suggested committee based consensus protocols. For higher throughput and lower latency committee-based protocols are required, and are now becoming the norm in proof-of-stake designs.

Existing approaches to increasing the performance of distributed ledgers focus on creating lower-cost consensus algorithms culminating with HotStuff [38], which achieves linear message complexity in the partially synchronous setting. To achieve this, HotStuff leverages a leader who collects, aggregates, and broadcasts the messages of other validators. However, theoretical message complexity should not be the only optimization target. More specifically:

- Any (partially-synchronous) protocol that minimizes overall message number, but relies on a leader to produce proposals and coordinate consensus, fails to capture the high load this imposes on the leader who inevitably becomes a bottleneck.
- Message complexity counts the number of *metadata* messages (e.g., votes, signatures, hashes) which take minimal bandwidth compared to the dissemination of bulk transaction data (blocks). Since blocks are orders of magnitude larger (10MB) than a typical consensus message (100B), the asymptotic message complexity is practically amortized for fixed mid-size committees (up to ~ 50 nodes).

Additionally, consensus protocols have grouped a lot of functions into a monolithic protocol. In a typical distributed

## Narwhal

- Quadratic but even resource utilisation
- Separation between consensus and data dissemination
- High engineering complexity

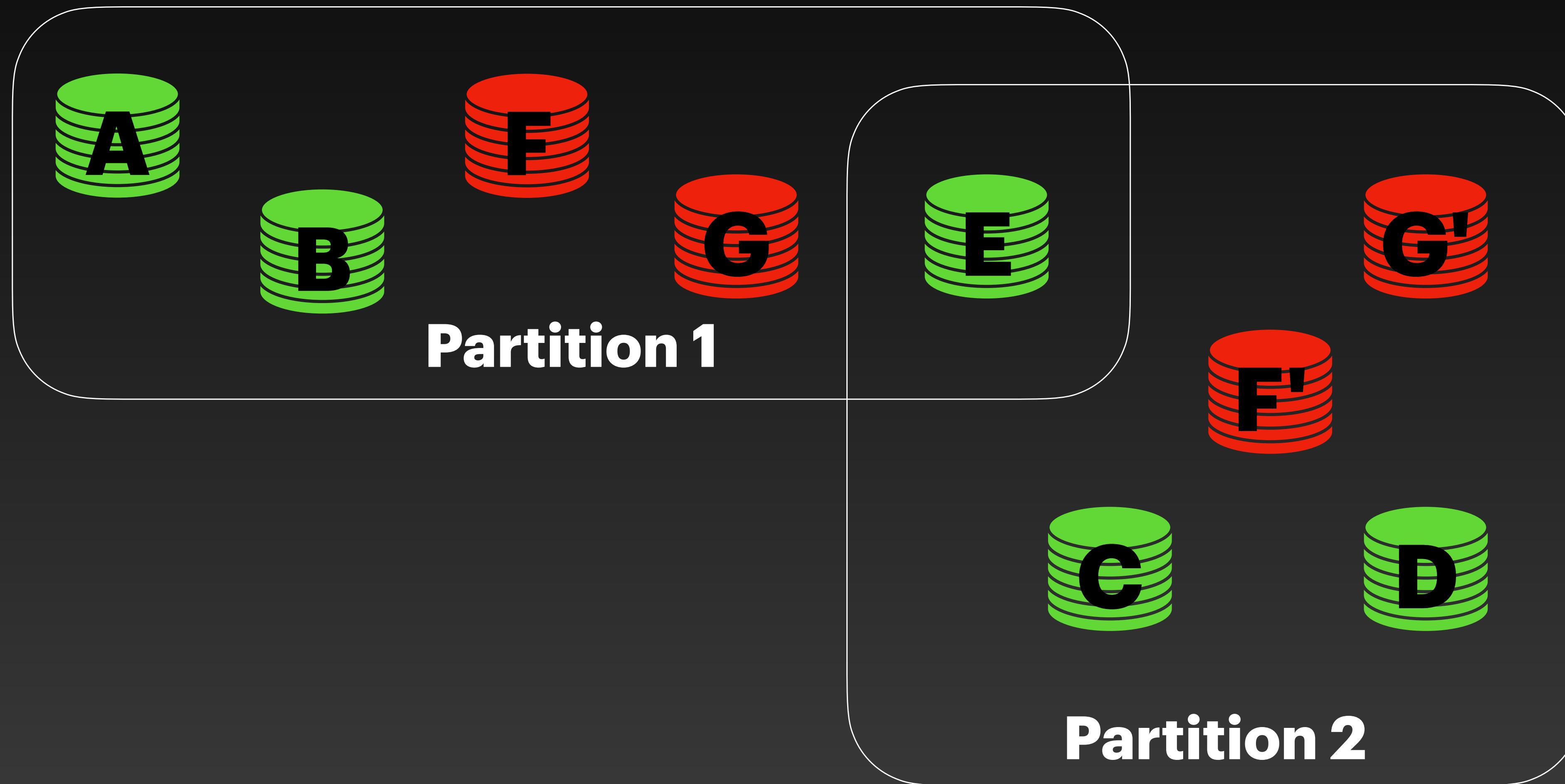
# Research Questions

1. Network model?
2. BFT testing?

# Lessons Learned

1. Modularisation is a design strategy
2. Tasks-threads allocation
3. Benchmark early
4. Codesign with mem. and storage

# Twins



# DagRider

**All You Need is DAG**

Idit Keidar  
Technion

Oded Naor\*  
Technion

**ABSTRACT**  
We present *DagRider*, the first asynchronous Byzantine Atomic Broadcast protocol that is post-quantum secure, optimal communication complexity, and optimal time complexity. DagRider is correct processes eventually get delivered. We construct two layers. In the lower layer, *Narwhal* tolerates a single faulty node and does not rely on asymmetric cryptographic assumption. Therefore, when using a deterministic threshold-based coin implementation, the safety properties of our DAG protocol are post-quantum secure.

**ACM Reference Format:**  
Idit Keidar, Eleftherios Kokoris-Kogias, Oded Naor, and Alexander Spiegelman. 2021. All You Need is DAG. In *Proceedings of the 2021 ACM SIGPLAN Conference on Principles of Distributed Computing (PODC '21)*, July 26–30, 2021, Virtual Event, Italy. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3460044.3479221>

# Tusk

**Narwhal and Tusk: A DAG-based Mempool and Efficient BFT Consensus**

George Danezis  
Mythen Labs & UCL

Alexander Spiegelman  
Mythen Labs

**Abstract**  
We propose separating the task of reliable transaction dissemination from transaction ordering, to enable high-performance Byzantine fault-tolerant quorum-based consensus. We design Narwhal as a mempool-based protocol, Narwhal tolerates at most two faulty nodes in the network and stores a copy of each transaction in its mempool. Tusk is a stateless consensus protocol that casts their processes and has a timestamped Directed Acyclic Graph (DAG) of the communication among them. In the second layer, Tusk preserves the safety properties of their DAGs and totally order all processes with no extra communication.

**ACM Reference Format:**  
George Danezis, Eleftherios Kokoris-Kogias, Alberto Sonnino, and Alexander Spiegelman. 2021. Narwhal and Tusk: A DAG-based Mempool and Efficient BFT Consensus. In *Seventeenth European Conference on Computer Systems (EuroSys '21)*, April 5–6, 2021, Rennes, France. ACM, New York, NY, USA, 17 pages. <https://doi.org/10.1145/3451959.3459594>

# Bullshark

**Bullshark: DAG BFT Protocols Made Practical**

Alexander Spiegelman  
sasha@septangle.com  
Aptos

George Danezis  
IST Austria

Alexander Spiegelman  
Aptos

**Abstract**  
We present BullShark, the first directed acyclic graph (DAG) based asynchronous Byzantine atomic broadcast protocol that is optimized for the common synchronous case. Like previous DAG-based BFT protocols [19, 25], BullShark requires no extra communication to achieve consensus at the top of the DAG. That is, parties can safely order messages sent by other parties in the bottom half of the DAG by locally interpreting their view of it without sending any extra messages. This is, once we build the DAG, implementing consensus is as simple as reading off the top of the DAG.

The pioneering work of Hashgraph [4] constructed an unstructured DAG, where each message refers to previous ones, and used batches of messages as local coin flips to totally order the DAG in expectation. BullShark follows a similar approach, but uses a structured round-based DAG and encodes a shared randomness in each round via a threshold signature scheme to achieve constant latency in expectation. BullShark is the first DAG-based BFT protocol built on previous ideas. Every round in the DAG has at most  $n$  vertices (one for each party), each of which contains a block of transactions as well as a timestamped log of the previous round's messages in a previous round. Blocks are disseminated via reliable broadcast [11] to avoid equivocation and an honest party advances to the next round once it reliably sees  $\lceil \frac{n}{2} + 1 \rceil$  messages from the previous round. Note that building the DAG requires honest parties to broadcast vertices even if they have no transactions to propose. However, the edges of the DAG are not necessarily transitive, so it is not safe to totally order all the DAG's vertices. So in this sense it is not different from other BFT protocols in which parties send explicit vote messages, which contain no transactions as well. Remarkably, by using the DAG's structure, BullShark only pays a steep 50% latency increase as it optimizes for asynchrony.

**ACM Reference Format:**  
Alexander Spiegelman, George Danezis, Alberto Sonnino, and Eleftherios Kokoris-Kogias. 2022. Bullshark: DAG BFT Protocols Made Practical. In *Proceedings of ACM Conference, Los Angeles, CA, USA, November 2022 (Compoce 22)*, 17 pages. <https://doi.org/10.1145/3495000.3495003>

# Dumbo-NG

**Dumbo-NG: Fast Asynchronous BFT Consensus with Throughput-Oblivious Latency**

Yingqi Guo\*  
ISCA & UCAS

Yuan Lu\*  
ISCA

Zherong Lu\*  
USID

Neil Girdharan  
girdharan@berkeley.edu  
UC Berkeley

Jing Xu\*  
ISCA

Zhenfeng Zhang\*  
ISCA

Qiang Tang  
UCSC  
qiang.tang@syu.edu.au

huayan@iscas.ac.cn

zhulu@ust.hk

zhenfeng@iscas.ac.cn

**Abstract**  
Despite recent progresses of practical asynchronous Byzantine-fault tolerant (BFT) consensus, the state-of-the-art designs still suffer from suboptimal performance. Particularly, to obtain maximum throughput, one needs to rely on the infrastructure of distributed ledger for mission-critical applications. Such decentralized business is envisioned as critical global infrastructure maintained by a set of multi-party consensus protocols. However, the consensus protocols are agreed to be exposed to the threat of a single point of failure. The large success of Hashgraph [4] and Blockchain [19, 24] leads to an interesting research direction to lay down the infrastructures of distributed ledger for mission-critical applications. Such decentralized business is envisioned as critical global infrastructure maintained by a set of multi-party consensus protocols. However, the consensus protocols are agreed to be exposed to the threat of a single point of failure.

We present Dumbo-NG, a novel asynchronous BFT consensus (throughput-oblivious latency) that is designed for mission-critical technical core is a non-trivial direct derivation from asynchronous atomic broadcast to multi-valued validated Byzantine agreement (MVA). Dumbo-NG is the first BFT consensus that achieves linear throughput of  $O(n)$  (twice as much as Hashgraph [29]) suggested consensus-based consensus protocols. For higher throughput and lower latency committee-based consensus, more nodes are required, and we make a significant breakthrough in proof of impossibility. Dumbo-NG is the first BFT consensus that minimizes the number of nodes required to achieve linear throughput. Note that the number of nodes required to achieve linear throughput is only slightly larger than the number of nodes required to achieve linear throughput in Hashgraph [4].

We implement Dumbo-NG with the current fastest GL-22 MVA with quorum (N2D2) and compare it to the state-of-the-art throughput-oblivious BFT consensus, Speeding-Dumbo and dumbo-DL (for varying scales). More interestingly, Dumbo-NG's latency, which is lowest among all tested protocols, can almost remain stable when throughput grows, thus conquering the censorship threat with no extra cost.

We implement Dumbo-NG with the current fastest GL-22 MVA with quorum (N2D2) and compare it to the state-of-the-art throughput-oblivious BFT consensus, Speeding-Dumbo and dumbo-DL for comprehensive comparison. Extensive experiments (over up to 64 AWS EC2 nodes across 16 AWS regions) demonstrate that Dumbo-NG is the most efficient BFT consensus, especially for Speeding-Dumbo, and 3x over dumbo-DL (for varying scales). More importantly, Dumbo-NG's latency, which is lowest among all tested protocols, can almost remain stable when throughput grows, thus conquering the censorship threat with no extra cost.

**CCS CONCEPTS:**  
• Any (partial) consensus protocol that minimizes overall message number, but relies on a leader to propose proposals and coordinate consensus fails to capture the high load this imposes on the leader who inevitably becomes a bottleneck.

**Keywords:** Consensus protocol, Byzantine Fault Tolerant

**Permissions:** To make digital or hard copies of all or part of this work for personal use or the internal or local network of your organization, you may only copy or re-use what is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](https://permissions ACM.org). © 2022 Copyright held by the author(s)/owner(s). Publication rights licensed to ACM. 978-1-4503-8549-0/22/07...\$15.00  
https://doi.org/10.1145/3495231.3519594

# Data Dissemination

- Hard to make efficient
- 99% of the code

# Consensus

- Error prone
- Isolated, easy to maintain

# Bullshark

## All You Need is DAG

Idit Keidar  
Technion

Oded Naor  
Technion

**ABSTRACT**  
We present *DAG-Rider*, the first asynchronous Byzantine Atomic Broadcast protocol achieving optimal resilience. It is the first DAG-Rider is post-quantum safe and ensures that all values proposed by correct processes eventually get delivered. We construct *DAG-Rider* in two layers. In the inner layer, processes broadcast their proposed values in a timestamped Directed Acyclic Graph (DAG) of the communication among them. In the second layer processes locally observe their DAGs and totally order all proposals with a timestamp.

**ACM Reference Format:**  
Idit Keidar, Eleftherios Kokoris-Kogias, Oded Naor, and Alexander Spiegelman. 2021. All You Need is DAG. In *Proceedings of Distributed Computing (PODC '21), July 28–30, 2021, Virtual Event, Italy*. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3453132.3463252>

## 1 INTRODUCTION

The rapid growth in scalable ge-replicated Byzantine fault-tolerant systems has motivated an enormous amount of study on the Byzantine State Machine Replication (SMR) problem [17, 18, 20, 21, 22]. Many variants of the problem were defined in recent years [17, 18, 20, 21, 22], and many solutions have been proposed [23–27]. To address the fairness issues that naturally arise in interorganizational deployments, we focus on the classic long-lived Byzantine Atomic Broadcast (BAB) problem [28].

Asynchronous consensus protocols for the Byzantine consensus problem [22, 16, 26] have been considered too costly or complicated to be used in practical SMR solutions. However, two recent papers [29, 30] presented asynchronous consensus protocols with (1) optimal resilience, (2) constant time complexity, and (3) optimal communication and optimal amortized communication complexity.

For the first time in the literature, we follow a similar line

of work and present *DAG-Rider*, the first asynchronous BAB protocol with optimal resilience, optimal round complexity, and optimal amortized communication complexity. In addition, given a perfect adversary that does not corrupt more than half of the nodes, and assuming that correct processes eventually get delivered, we construct *DAG-Rider* in two layers. In the inner layer, processes broadcast their proposed values in a timestamped Directed Acyclic Graph (DAG) of the communication among them. In the second layer processes locally observe their DAGs and totally order all proposals with a timestamp.

## Narwhal and Tusk: A DAG-based Mempool and Efficient BFT Consensus

Eleftherios Kokoris-Kogias  
IST Austria and Novi Research

Alexander Spiegelman  
Novi Research

**ABSTRACT**

We propose separating the task of reliable transaction dissemination from transaction ordering. In the ordering layer, we send some messages that help them form a *Directed Acyclic Graph* (DAG) of the messages they deliver. That is, the DAG consists of rounds  $r$ : every process broadcasts at most one message in every round and every message is timestamped with its round number. The total number of rounds is the total number of processes. The ordering layer does not require any extra communication. Instead, processes observe their DAGs and totally order all the messages they receive. This allows us to drop (or re-order) decisions on values proposed by different processes.

Overviews. We construct *DAG-Rider* in two layers: a communication layer and a zero-overhead ordering layer. In the communication layer, processes broadcast their values with some timestamp data that help them form a *Directed Acyclic Graph* (DAG) of the messages they deliver. That is, the DAG consists of rounds  $r$ : every process broadcasts at most one message in every round and every message is timestamped with its round number. The total number of rounds is the total number of processes. The ordering layer does not require any extra communication. Instead, processes observe their DAGs and totally order all the messages they receive. This allows us to drop (or re-order) decisions on values proposed by different processes.

ACM Reference Format:

George Danesis, Lefteris Kokoris-Kogias, Alberto Sonnino, and Alexander Spiegelman. 2022. Narwhal and Tusk: A DAG-based Mempool and Efficient BFT Consensus. In *Seventeenth European Conference on Computer Systems (EuroSys '22), April 4–6, 2022, Rennes, France*. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3509594>

## 1 INTRODUCTION

Byzantine consensus protocols [15, 19, 21] and the state of the art [29, 30] have been studied for over 40 years. However, with the rise in popularity of blockchains there has been a renewed interest in engineering high-performance consensus protocols. Specifically, the implementation of BitShares's [31] consensus protocol is a prime example of a highly performant consensus protocol, to work with Narwhal. We demonstrate its high performance under a variety of configurations and faults.

As a summary of results, on a WAN, Narwhal-Hotstuff achieves over 100 tx/sec with less than 2-second latency per message over 1,800 tx/sec at 1-second latency for Hotstuff. Additional processes increase throughput linearly to 600,000 tx/sec without any latency increase. Thus achieves 160,000 tx/sec with an average 3 seconds latency. Under faults, both protocols show high resilience, but Narwhal-Hotstuff suffers from increased latency.

Complexity. We measure time complexity as the asynchronous consensus problem [22, 16, 26] have been considered too costly or complicated to be used in practical SMR solutions. However, two recent papers [29, 30] presented asynchronous consensus protocols with (1) optimal resilience, (2) constant time complexity, and (3) optimal communication and optimal amortized communication complexity.

For the first time in the literature, we follow a similar line

of work and present *DAG-Rider*, the first asynchronous BAB protocol with optimal resilience, optimal round complexity, and optimal amortized communication complexity.

**Keywords:** Consensus protocol, Byzantine Fault Tolerant

Permit to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyright for this work belongs to others than ACM or editor. Authorization to make copies must be obtained from the owner(s) or the author(s). For more information about the journal that published this article, please go to the journal's website or on the publisher's website. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permission from permissions@acm.org. Last updated: 2022-07-22. © 2022 Copyright held by the owner(s). All rights reserved. ACM ISBN 978-1-4503-1617-7/22/07...\$15.00  
<https://doi.org/10.1145/3453132.3463252>

*\*Oded Naor is grateful to the Turkish Ministry of Science and Technology for providing a research grant. Part of Oded's work was done while he was a visiting professor at Novi Research.*

*Permit to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyright for this work belongs to others than ACM or editor. Authorization to make copies must be obtained from the owner(s) or the author(s). For more information about the journal that published this article, please go to the journal's website or on the publisher's website. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permission from permissions@acm.org. Last updated: 2022-07-22. © 2022 Copyright held by the owner(s). All rights reserved. ACM ISBN 978-1-4503-1617-7/22/07...\$15.00  
<https://doi.org/10.1145/3453132.3463252>*

## Narwhal and Tusk: A DAG-based Mempool and Efficient BFT Consensus

George Danesis  
Mysten Labs & UCL

Alberto Sonnino  
Mysten Labs

**ABSTRACT**

We propose separating the task of reliable transaction dissemination from transaction ordering, to enable high-performance Byzantine fault-tolerant quorum-based consensus. We design and evaluate a memory protocol, Narwhal, specifically in the context of a reliable distributed mempool. Narwhal leverages causal histories of transactions to maintain high performance despite failures. Narwhal is designed to easily scale-out using multiple workers at each node. Narwhal tolerates asynchronous communication and is fully aware of their DAGs and totally order all proposals with a timestamp.

ACM Reference Format:

George Danesis, Lefteris Kokoris-Kogias, Alberto Sonnino, and Alexander Spiegelman. 2022. Narwhal and Tusk: A DAG-based Mempool and Efficient BFT Consensus. In *Seventeenth European Conference on Computer Systems (EuroSys '22), April 4–6, 2022, Rennes, France*. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3509594>

## 1 INTRODUCTION

Byzantine consensus protocols [15, 19, 21] and the state of the art [29, 30] have been studied for over 40 years. However, with the rise in popularity of blockchains there has been a renewed interest in engineering high-performance consensus protocols. Specifically, the implementation of BitShares's [31] consensus protocol is a prime example of a highly performant consensus protocol, to work with Narwhal. We demonstrate its high performance under a variety of configurations and faults.

As a summary of results, on a WAN, Narwhal-Hotstuff achieves over 100 tx/sec with less than 2-second latency per message over 1,800 tx/sec at 1-second latency for Hotstuff. Additional processes increase throughput linearly to 600,000 tx/sec without any latency increase. Thus achieves 160,000 tx/sec with an average 3 seconds latency. Under faults, both protocols show high resilience, but Narwhal-Hotstuff suffers from increased latency.

Complexity. We measure time complexity as the asynchronous consensus problem [22, 16, 26] have been considered too costly or complicated to be used in practical SMR solutions. However, two recent papers [29, 30] presented asynchronous consensus protocols with (1) optimal resilience, (2) constant time complexity, and (3) optimal communication and optimal amortized communication complexity.

For the first time in the literature, we follow a similar line

of work and present *DAG-Rider*, the first asynchronous BAB protocol with optimal resilience, optimal round complexity, and optimal amortized communication complexity.

**Keywords:** Consensus protocol, Byzantine Fault Tolerant

Permit to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyright for this work belongs to others than ACM or editor. Authorization to make copies must be obtained from the owner(s) or the author(s). For more information about the journal that published this article, please go to the journal's website or on the publisher's website. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permission from permissions@acm.org. Last updated: 2022-07-22. © 2022 Copyright held by the owner(s). All rights reserved. ACM ISBN 978-1-4503-1617-7/22/07...\$15.00  
<https://doi.org/10.1145/3453132.3463252>

*\*Oded Naor is grateful to the Turkish Ministry of Science and Technology for providing a research grant. Part of Oded's work was done while he was a visiting professor at Novi Research.*

*Permit to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyright for this work belongs to others than ACM or editor. Authorization to make copies must be obtained from the owner(s) or the author(s). For more information about the journal that published this article, please go to the journal's website or on the publisher's website. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permission from permissions@acm.org. Last updated: 2022-07-22. © 2022 Copyright held by the owner(s). All rights reserved. ACM ISBN 978-1-4503-1617-7/22/07...\$15.00  
<https://doi.org/10.1145/3453132.3463252>*

## 1 INTRODUCTION

Byzantine consensus protocols [15, 19, 21] and the state of the art [29, 30] have been studied for over 40 years. However, with the rise in popularity of blockchains there has been a renewed interest in engineering high-performance consensus protocols. Specifically, the implementation of BitShares's [31] consensus protocol is a prime example of a highly performant consensus protocol, to work with Narwhal. We demonstrate its high performance under a variety of configurations and faults.

As a summary of results, on a WAN, Narwhal-Hotstuff achieves over 100 tx/sec with less than 2-second latency per message over 1,800 tx/sec at 1-second latency for Hotstuff. Additional processes increase throughput linearly to 600,000 tx/sec without any latency increase. Thus achieves 160,000 tx/sec with an average 3 seconds latency. Under faults, both protocols show high resilience, but Narwhal-Hotstuff suffers from increased latency.

Complexity. We measure time complexity as the asynchronous consensus problem [22, 16, 26] have been considered too costly or complicated to be used in practical SMR solutions. However, two recent papers [29, 30] presented asynchronous consensus protocols with (1) optimal resilience, (2) constant time complexity, and (3) optimal communication and optimal amortized communication complexity.

For the first time in the literature, we follow a similar line

of work and present *DAG-Rider*, the first asynchronous BAB protocol with optimal resilience, optimal round complexity, and optimal amortized communication complexity.

**Keywords:** Consensus protocol, Byzantine Fault Tolerant

Permit to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyright for this work belongs to others than ACM or editor. Authorization to make copies must be obtained from the owner(s) or the author(s). For more information about the journal that published this article, please go to the journal's website or on the publisher's website. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permission from permissions@acm.org. Last updated: 2022-07-22. © 2022 Copyright held by the owner(s). All rights reserved. ACM ISBN 978-1-4503-1617-7/22/07...\$15.00  
<https://doi.org/10.1145/3453132.3463252>

*\*Oded Naor is grateful to the Turkish Ministry of Science and Technology for providing a research grant. Part of Oded's work was done while he was a visiting professor at Novi Research.*

*Permit to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyright for this work belongs to others than ACM or editor. Authorization to make copies must be obtained from the owner(s) or the author(s). For more information about the journal that published this article, please go to the journal's website or on the publisher's website. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permission from permissions@acm.org. Last updated: 2022-07-22. © 2022 Copyright held by the owner(s). All rights reserved. ACM ISBN 978-1-4503-1617-7/22/07...\$15.00  
<https://doi.org/10.1145/3453132.3463252>*

## 1 INTRODUCTION

Byzantine consensus protocols [15, 19, 21] and the state of the art [29, 30] have been studied for over 40 years. However, with the rise in popularity of blockchains there has been a renewed interest in engineering high-performance consensus protocols. Specifically, the implementation of BitShares's [31] consensus protocol is a prime example of a highly performant consensus protocol, to work with Narwhal. We demonstrate its high performance under a variety of configurations and faults.

As a summary of results, on a WAN, Narwhal-Hotstuff achieves over 100 tx/sec with less than 2-second latency per message over 1,800 tx/sec at 1-second latency for Hotstuff. Additional processes increase throughput linearly to 600,000 tx/sec without any latency increase. Thus achieves 160,000 tx/sec with an average 3 seconds latency. Under faults, both protocols show high resilience, but Narwhal-Hotstuff suffers from increased latency.

Complexity. We measure time complexity as the asynchronous consensus problem [22, 16, 26] have been considered too costly or complicated to be used in practical SMR solutions. However, two recent papers [29, 30] presented asynchronous consensus protocols with (1) optimal resilience, (2) constant time complexity, and (3) optimal communication and optimal amortized communication complexity.

For the first time in the literature, we follow a similar line

of work and present *DAG-Rider*, the first asynchronous BAB protocol with optimal resilience, optimal round complexity, and optimal amortized communication complexity.

**Keywords:** Consensus protocol, Byzantine Fault Tolerant

Permit to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyright for this work belongs to others than ACM or editor. Authorization to make copies must be obtained from the owner(s) or the author(s). For more information about the journal that published this article, please go to the journal's website or on the publisher's website. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permission from permissions@acm.org. Last updated: 2022-07-22. © 2022 Copyright held by the owner(s). All rights reserved. ACM ISBN 978-1-4503-1617-7/22/07...\$15.00  
<https://doi.org/10.1145/3453132.3463252>

*\*Oded Naor is grateful to the Turkish Ministry of Science and Technology for providing a research grant. Part of Oded's work was done while he was a visiting professor at Novi Research.*

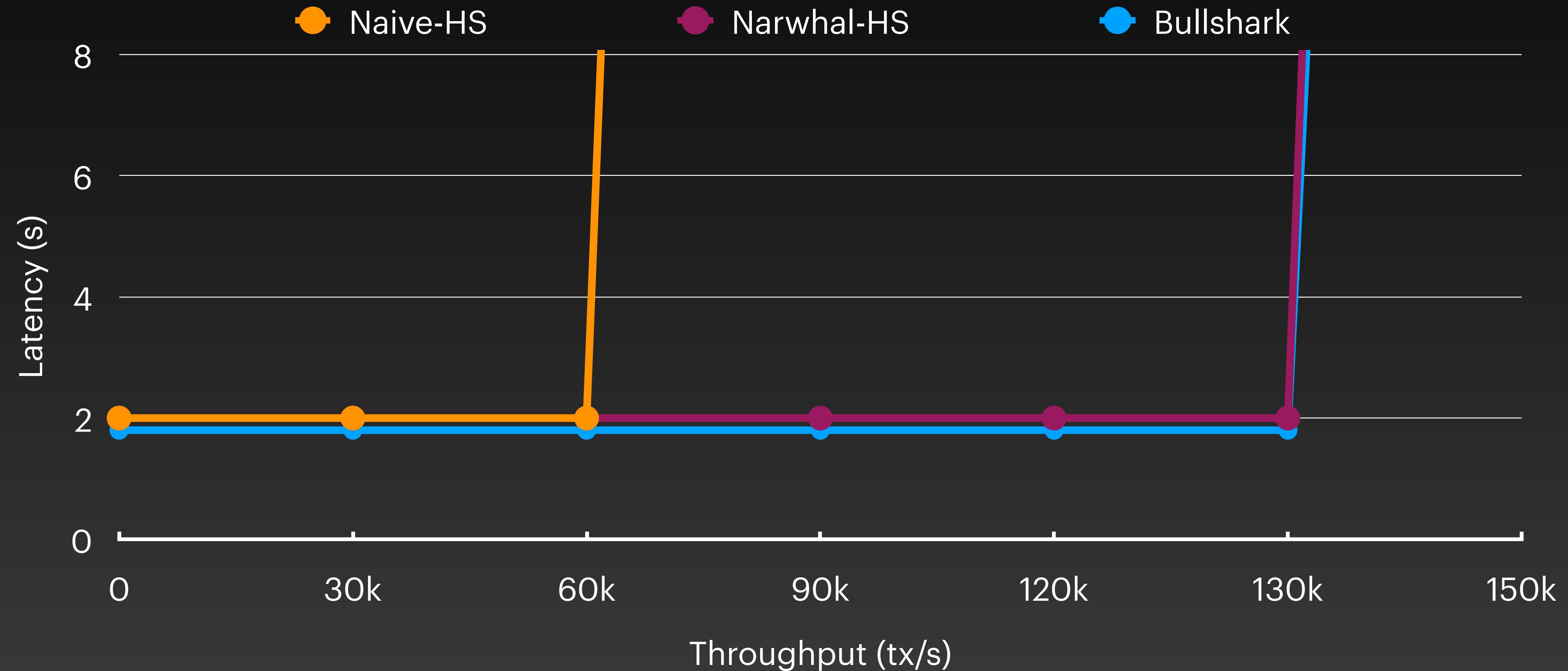
*Permit to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyright for this work belongs to others than ACM or editor. Authorization to make copies must be obtained from the owner(s) or the author(s). For more information about the journal that published this article, please go to the journal's website or on the publisher's website. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permission from permissions@acm.org. Last updated: 2022-07-22. © 2022 Copyright held by the owner(s). All rights reserved. ACM ISBN 978-1-4503-1617-7/22/07...\$15.00  
<https://doi.org/10.1145/3453132.3463252>*

## 1 INTRODUCTION

Byzantine consensus protocols [15, 19, 21] and the state of the art [29, 30] have been studied for over 40 years. However, with the rise in popularity of blockchains there has been a renewed interest in engineering high-performance consensus protocols. Specifically, the implementation of BitShares's [31] consensus protocol is a prime example of a highly performant consensus protocol, to work with Narwhal. We demonstrate its high performance under a variety of configurations and faults.

As a summary of results, on a WAN, Narwhal-Hotstuff achieves over 100 tx/sec with less than 2-second latency per message over 1,800 tx/sec at 1-second latency for Hotstuff. Additional processes increase throughput linearly to 600,000 tx/sec without any latency increase. Thus achieves 160,000 tx/sec with an average 3 seconds latency. Under faults, both protocols show high resilience, but Narwhal

# Performance



# Research Questions

1. Network model?
2. BFT testing?
3. Consensus-exec interface?

# Lessons Learned

1. Modularisation is a design strategy
2. Tasks-threads allocation
3. Benchmark early
4. Codesign with mem. and storage
5. Core is hard, consensus is easy

# By that time...



← Post

Reply

Pinned



David Marcus

@davidmarcus



...

How Libra Was Killed.

I never shared this publicly before, but since [@pmarca](#) opened the floodgates on [@joerogan](#)'s pod, it feels appropriate to shed more light on this.

As a reminder, Libra (then Diem) was an advanced, high-performance, payments-centric blockchain paired with a stablecoin that we built with my team at [@Meta](#). It would've solved global payments at scale. Prior to announcing the project, we spent months briefing key regulators in DC and abroad. We then announced the project in June 2019 alongside 28 companies. Two weeks later, I was called to testify in front of both the Senate Banking Committee and the House Financial Services Committee, which was the starting point of two years of nonstop work and changes to appease lawmakers and regulators.

By spring of 2021 (yes they slow played us at every step), we had addressed every last possible regulatory concern across financial crime, money laundering, consumer protection, reserve management, buffers,

# By that time...



**Sui**

**Aptos**

**Linera**

...

Fundraising with papers  
seems to work

# Sui, 2022

## **Over a year for mainnet**

- Lack of checkpoints
- Lack of epoch-change
- Lack of crash-recovery

# Research Questions

1. Network model?
2. BFT testing?
3. Consensus-exec interface?
4. Storage architecture?

# Lessons Learned

1. Modularisation is a design strategy
2. Tasks-threads allocation
3. Benchmark early
4. Codesign with mem. and storage
5. Core is hard, consensus is easy
6. Epoch change is not an add-on

# Sui, 2023

- Latency was too high
- Crash faults were the predominant faults
- Building Bullshark was still too complex

# Shoal

## Shoal: Improving DAG-BFT Latency And Robustness

Alexander Spiegelman  
Aptos  
Rati Gelashvili  
Aptos

### Abstract

The Narwhal system is a state-of-the-art Byzantine fault-tolerant scalable architecture that involves constructing a directed acyclic graph (DAG) of messages among a set of validators (or Blockleaders). Recently, it was proposed as a consensus protocol on top of the Narwhal's DAG that can support over 100k transactions per second. Unfortunately, the high throughput of Bullshark comes with a latency price due to the DAG-based consensus mechanism, which compares to the state-of-the-art leader-based BFT consensus protocols.

We introduce Shoal, a protocol-agnostic framework for enhancing Narwhal-based consensus. By incorporating leader replacement, Shoal achieves a round time that is significantly reduced. Moreover, the combination of properties of the DAG construction and the leader replacement mechanism enables the protocol to scale in all but extremely adversarial scenarios in practice, a property we name "prevalent responsiveness". It strictly subsumes the established and often desired "optimistic responsiveness" property for BFT consensus.

We evaluate Shoal instantiated with Bullshark, the fastest existing Narwhal-based consensus protocol, in an open-source Blockchain project and provide experimental evaluations demonstrating up to 40% latency reduction in the failure-free execution. We also evaluate the execution with failures against the vanilla Bullshark implementation.

**CCS Concepts** - Security and privacy → Distributed systems security

**Keywords**: Consensus Protocol, Byzantine Fault Tolerance, ACM Reference Format

Alexander Spiegelman, Rati Gelashvili, and Zekun Li  
2023. Shoal: Improving DAG-BFT Latency And Robustness

### 1 Introduction

Byzantine fault tolerant (BFT) systems, including consensus protocols [13, 23, 24, 29] and state machine replication [7, 10, 22–24], have been designed to tolerate up to  $t$  faulty nodes as a means of constructing reliable distributed systems. Recently, the advent of Blockchains has underscored the significance of high performance. While Bitcoin handles approximately 10 transactions per second (TPS), the state-of-the-art consensus-based blockchains [30, 31, 43, 44] are now engaged in a race to deliver a scalable BFT system with the utmost throughput and minimal latency.

# Sailfish

## Sailfish: Towards Improving the Latency of DAG-based BFT

Nibesh Shrestha  
[n.shrestha@supraclouds.com](mailto:n.shrestha@supraclouds.com)  
Supra Research  
Balaji Arun  
Aptos  
Rati Gelashvili  
Aptos  
Zekun Li  
Aptos

Historically, the prevailing belief has been that reducing communication complexity was the key to unlocking high performance, low-latency consensus. However, this did not result in dramatic improvements in the throughput. For example, the state-of-the-art Hotstuff [46] protocol in this line of work supports over 100k transactions per second. Unfortunately, the high throughput of Bullshark comes with a latency price due to the DAG-based consensus mechanism, which compares to the state-of-the-art leader-based BFT consensus protocols.

We introduce Shoal, a protocol-agnostic framework for enhancing Narwhal-based consensus. By incorporating leader replacement, Shoal achieves a round time that is significantly reduced. Moreover, the combination of properties of the DAG construction and the leader replacement mechanism enables the protocol to scale in all but extremely adversarial scenarios in practice, a property we name "prevalent responsiveness". It strictly subsumes the established and often desired "optimistic responsiveness" property for BFT consensus.

We evaluate Shoal instantiated with Bullshark, the fastest existing Narwhal-based consensus protocol, in an open-source Blockchain project and provide experimental evaluations demonstrating up to 40% latency reduction in the failure-free execution. We also evaluate the execution with failures against the vanilla Bullshark implementation.

**CCS Concepts** - Security and privacy → Distributed systems security

**Keywords**: Consensus Protocol, Byzantine Fault Tolerance, ACM Reference Format

Alexander Spiegelman, Balaji Arun, Rati Gelashvili, and Zekun Li  
2023. Shoal: Improving DAG-BFT Latency And Robustness

### 1. Introduction

Byzantine fault tolerant (BFT) systems, including consensus protocols, form the core underpinning for blockchains. At a high level, a BFT-SMR enables a group of  $n$  parties to agree on a sequence of values, even if a bound of  $t$  up to  $\frac{n}{2}$  of these parties are Byzantine (arbitrarily malicious). Over the last decade, significant progress has been made in improving the efficiency of these consensus protocols. In particular, non-equivocable round-based directed acyclic graph (DAG), a concept initially introduced by Aleph [21]. In this model, each validator runs its own DAG locally, and each vertex is assigned to  $n-1$  vertices in the preceding round. Each vertex is disseminated via an efficient reliable broadcast implementation, ensuring that malicious validators cannot distinguish different vertices to different validators within the same round. This allows the DAG to be constructed without the need for consensus between validators. In their words, "Shoal is designed to be a DAG-based consensus protocol that can commit with a latency overhead of 3δ (where δ represents the round time)" [12]. They also claim that "Shoal can achieve linear communication complexity" [12].

Most of these protocols design a DAG such that a designated leader vertex is mainly responsible for proposing transactions and driving the protocol forward while other parties

# CM

## Cordial Miners: Fast and Efficient Consensus for Every Eventuality

Idit Keidar  
Technion  
Oded Naor  
Technion and StarkWare  
Ouri Poupko  
Ben-Gurion University  
Ehud Shapiro  
Weizmann Institute of Science

Agree on the proposed values and ensure that the leader keeps miners busy. Finally, determine the next round. This approach results in two drawbacks. First, there is an uneven scheduling of work among the parties. While the leader is sending a proposal, the other parties' processors and their miners are used to process and propagate more round particularly difficult, if not impossible. Consequently, even under honest leaders, these protocols require high latency (or communication complexity) to commit the proposed values and to keep the miners busy. Therefore, the use of erasure coding techniques [2], [42] or the data availability component of consensus [26], [27], [29] disseminating the data more efficiently.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader. Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of 1000 TPS. Additionally, Cordial Miners is the first DAG-based consensus protocol that achieves a fast commit path for the total-ordering of blockchain data structure—without the need for a leader.

Recently, a novel approach known as DAG-based BFT protocols has emerged [5], [18], [28], [33], [34], [40], [46]. These protocols enable a partial ordering of parties in safety and fast consensus. We evaluate the low latency and resource efficiency of Cordial Miners, a DAG-based consensus protocol for asynchronous and eventual synchronous environments. Cordial Miners is the first Byzantine consensus protocol to achieve WAN latency of 4.5 s for consensus commit while simultaneously maintaining a throughput of

# Discussion

**Certified DAG**

**Uncertified DAG**



**Shoal/shoal++**

- Low latency
- Easier synchroniser
- Leverage existing code

**Sailfish/BBCA**

- Lower latency
- Easy synchroniser
- Flexible

**CM/Mysticeti**

- Lowest latency
- Graceful crash faults
- Simpler, less CPU

# Research Questions

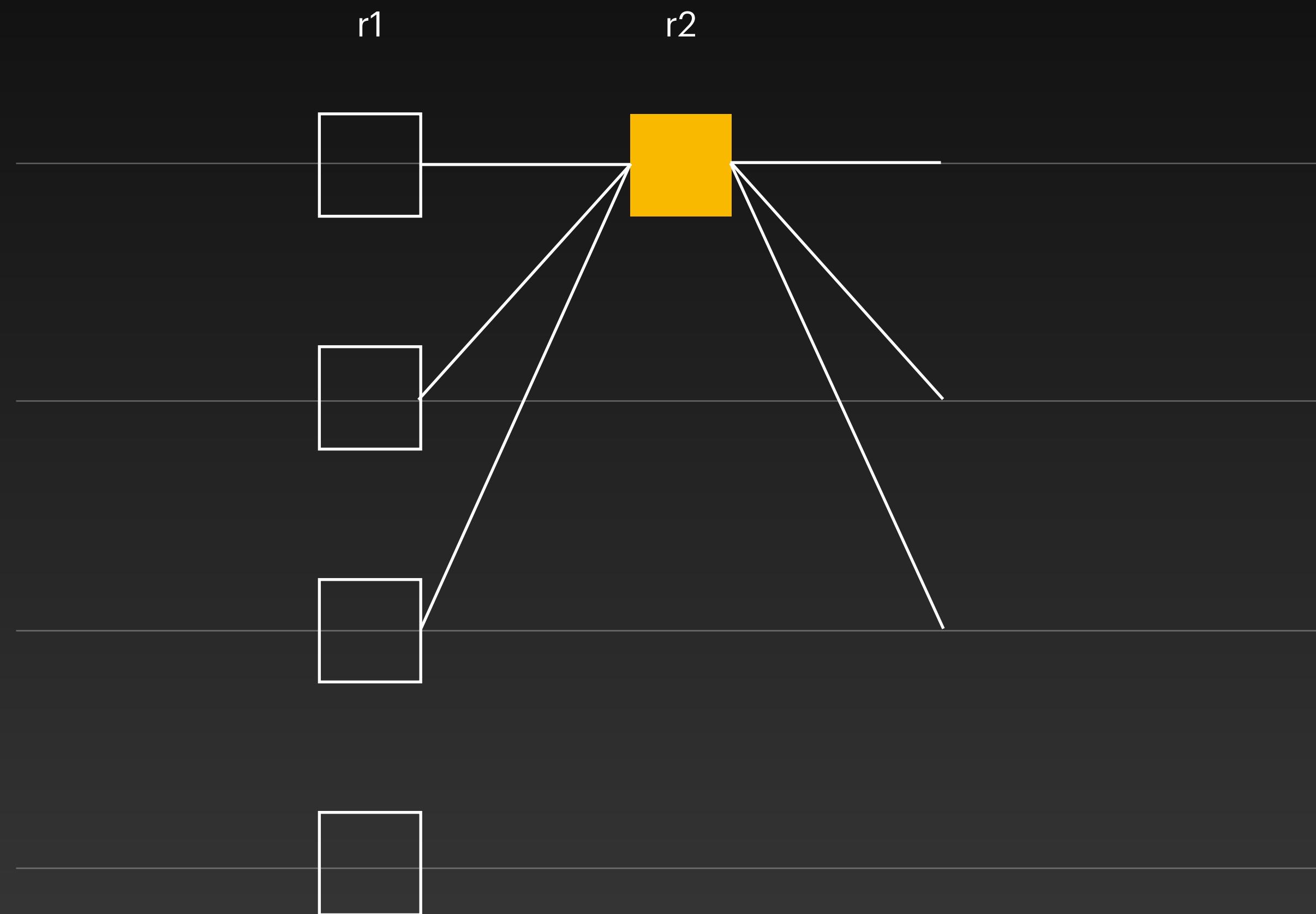
1. Network model?
2. BFT testing?
3. Consensus-exec interface?
4. Storage architecture?
5. Block synchroniser?

# Lessons Learned

1. Modularisation is a design strategy
2. Tasks-threads allocation
3. Benchmark early
4. Codesign with mem. and storage
5. Core is hard, consensus is easy
6. Epoch change is not an add-on

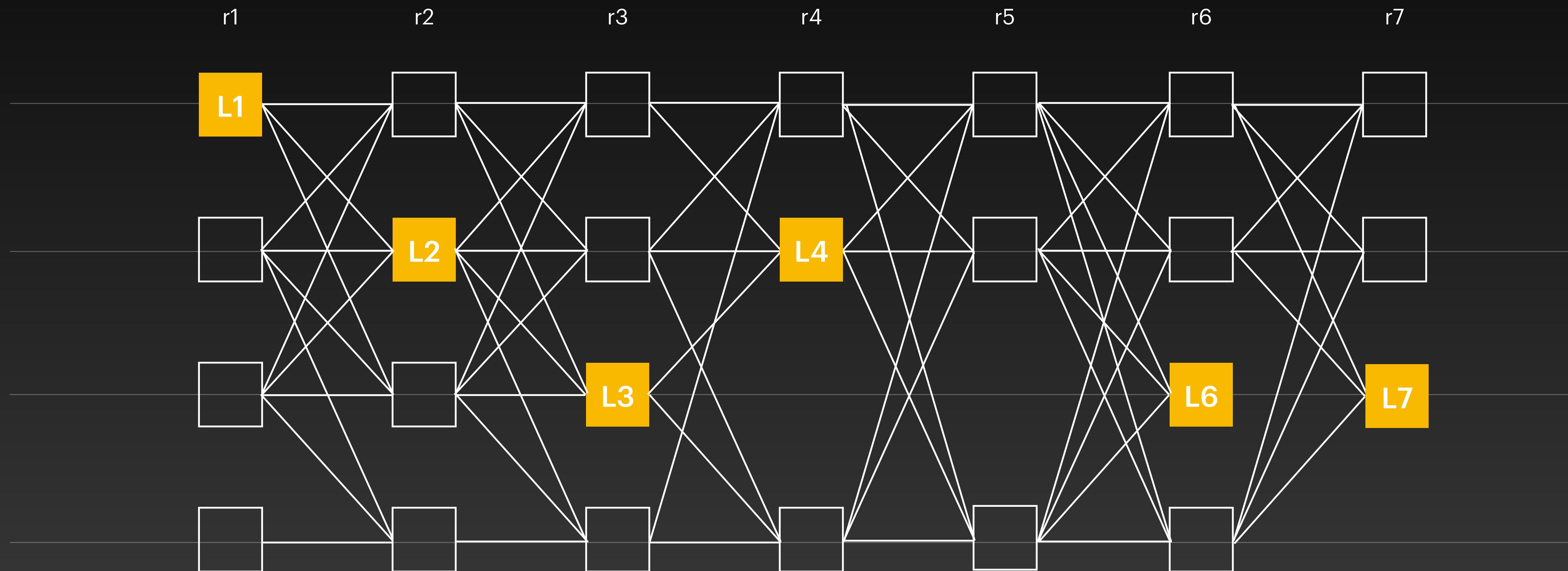


# Uncertified DAG

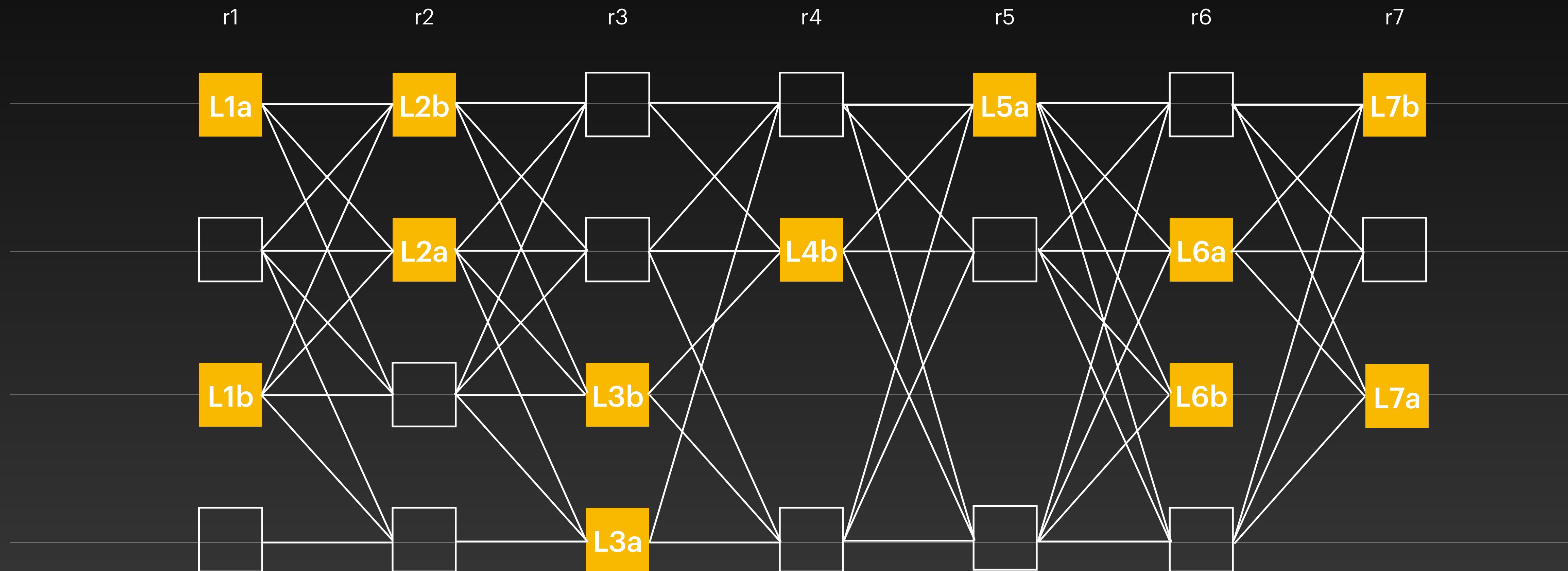


- Round number
- Author
- Payload (transactions)
- Signature

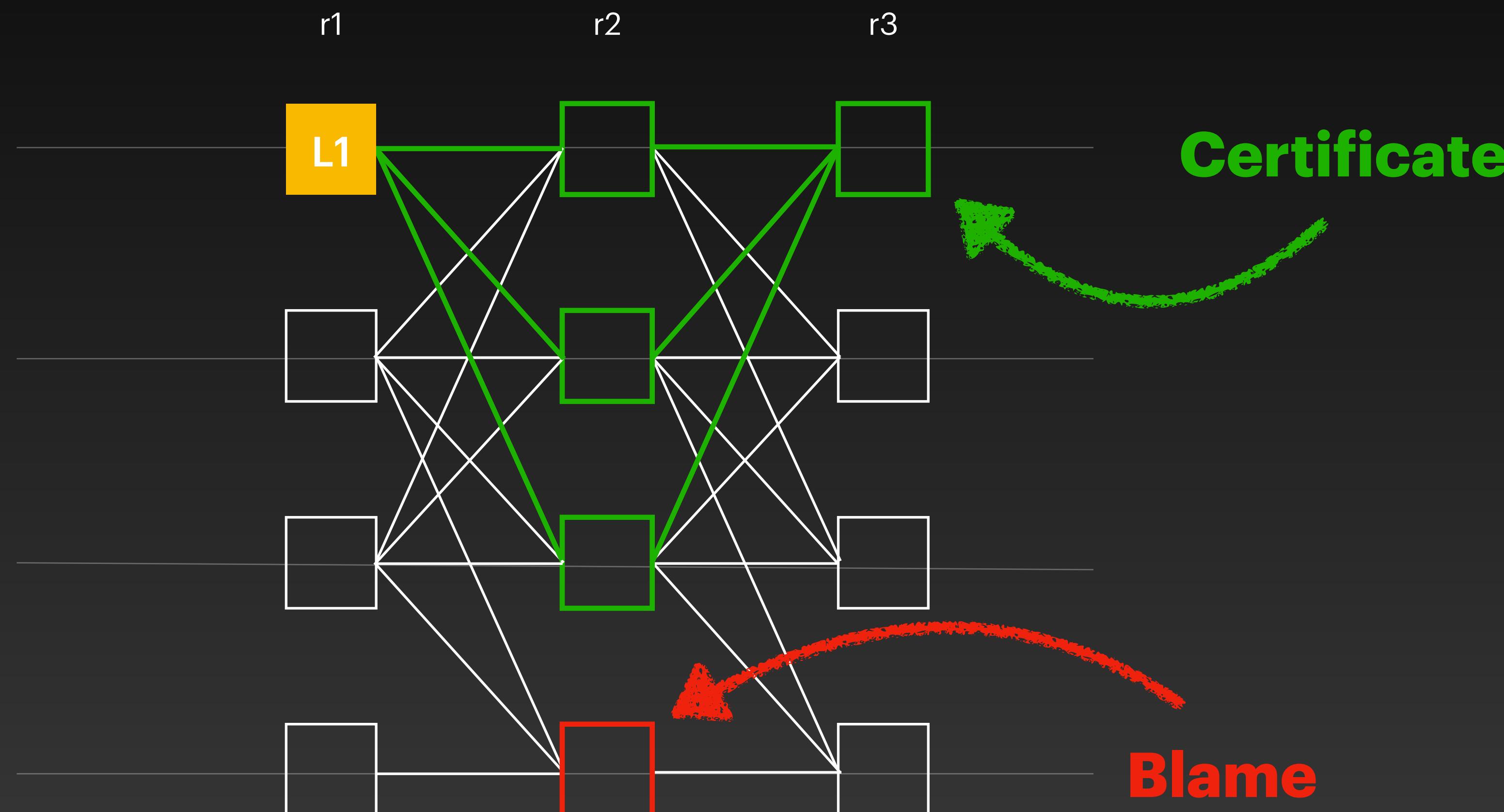
# Uncertified DAG



# Uncertified DAG



# Interpreting DAG Patterns



# Direct Decision Rule

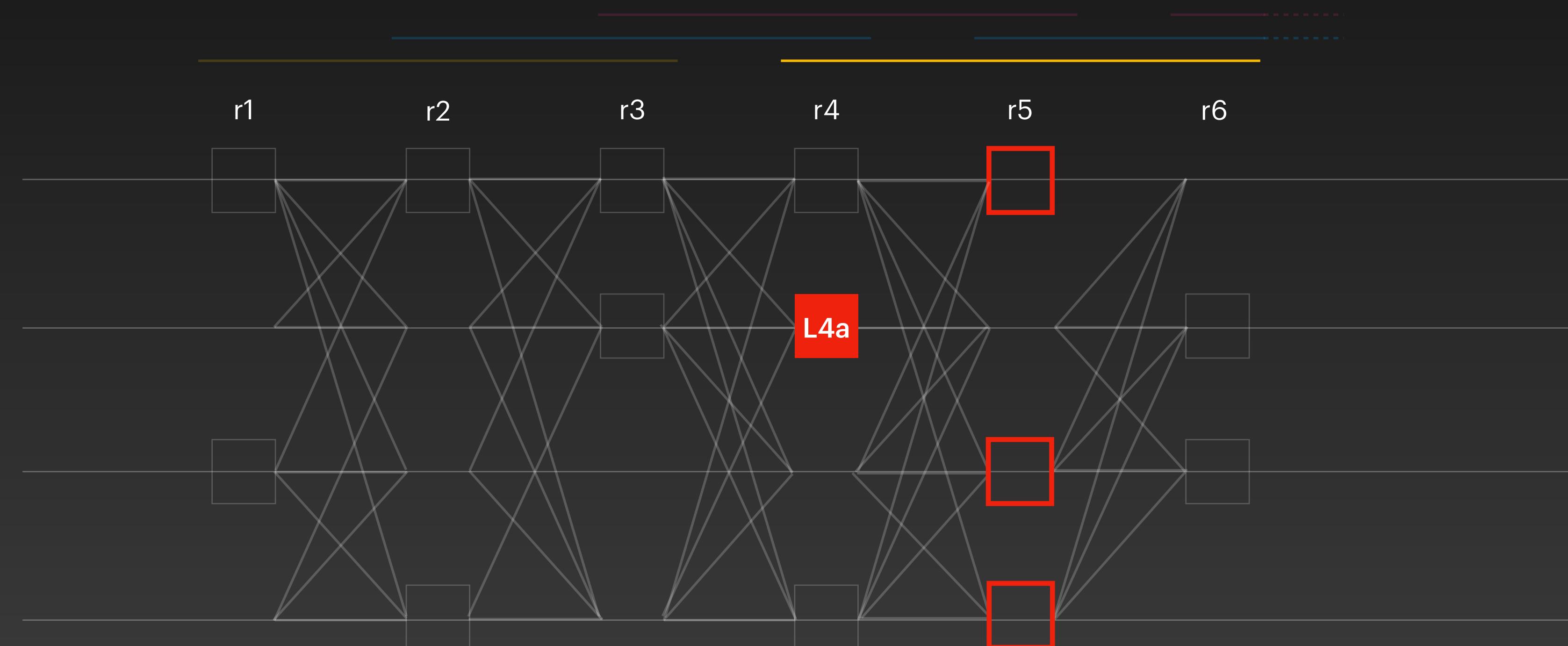
On each leader starting from highest round:

- **Skip** if  $2f+1$  blames
- **Commit** if  $2f+1$  certificates
- **Undecided** otherwise

# Direct Decision Rule

On each leader starting from highest round:

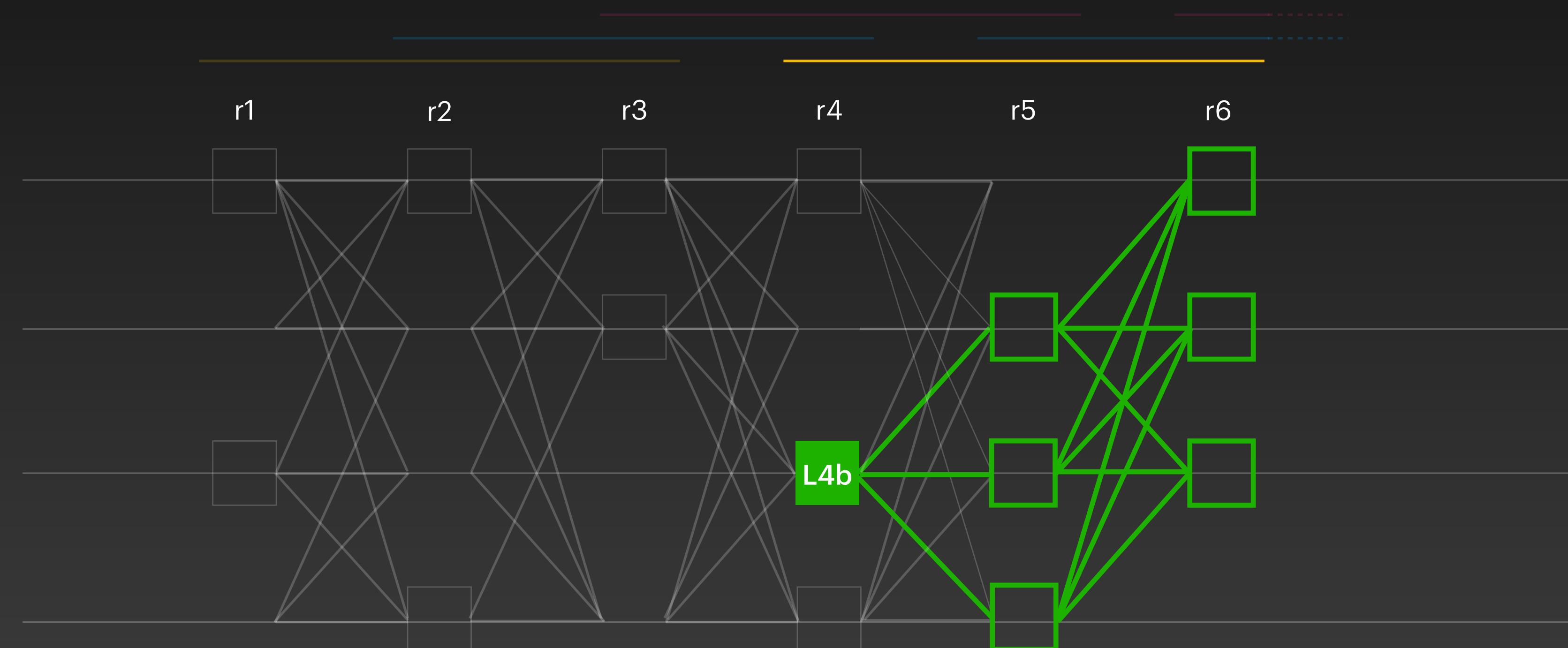
- **Skip** if  $2f+1$  blames
- **Commit** if  $2f+1$  certificates
- **Undecided** otherwise



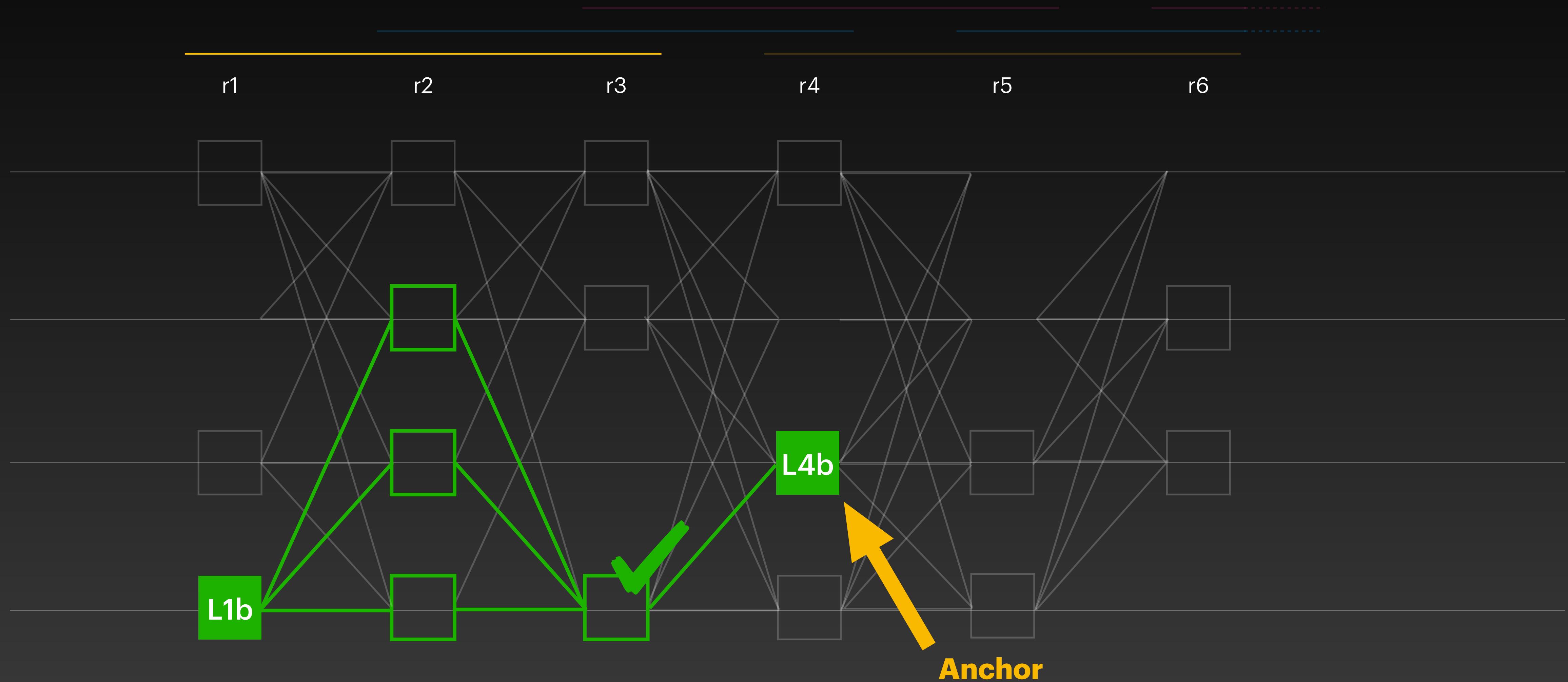
# Direct Decision Rule

On each leader starting from highest round:

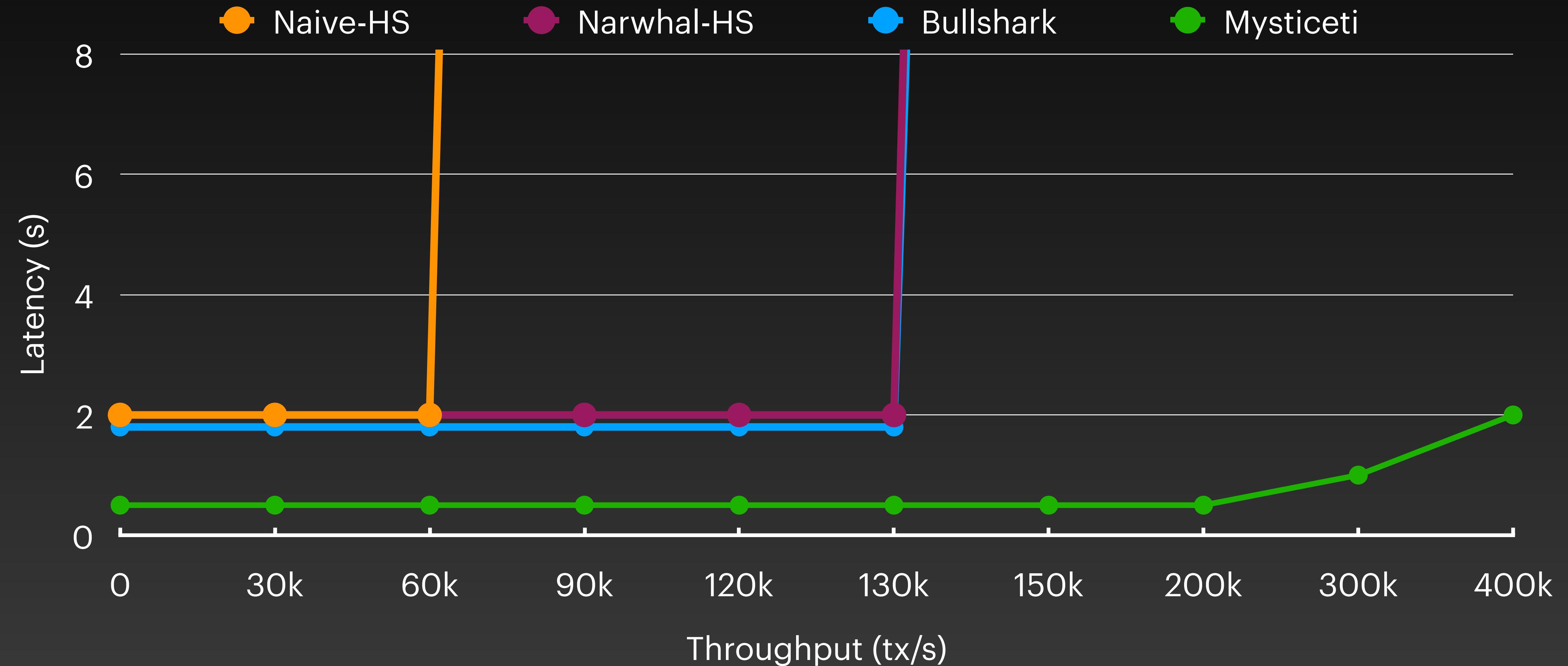
- **Skip** if  $2f+1$  blames
- **Commit** if  $2f+1$  certificates
- **Undecided** otherwise



# Indirect Decision Rule



# Performance



# Engineering Benchmarks

Protocol	Committee	Load/TPS	P50	P95
Bullshark	137	5k	2.89 s	4.60 s
Mysticeti	137	5k	397 ms	690 ms

We ran it for 24h and it looks good 

# Research Questions

1. Network model?
2. BFT testing?
3. Consensus-exec interface?
4. Storage architecture?
5. Block synchroniser?
6. Realistic benchmarks?
7. Efficient reads?

# Lessons Learned

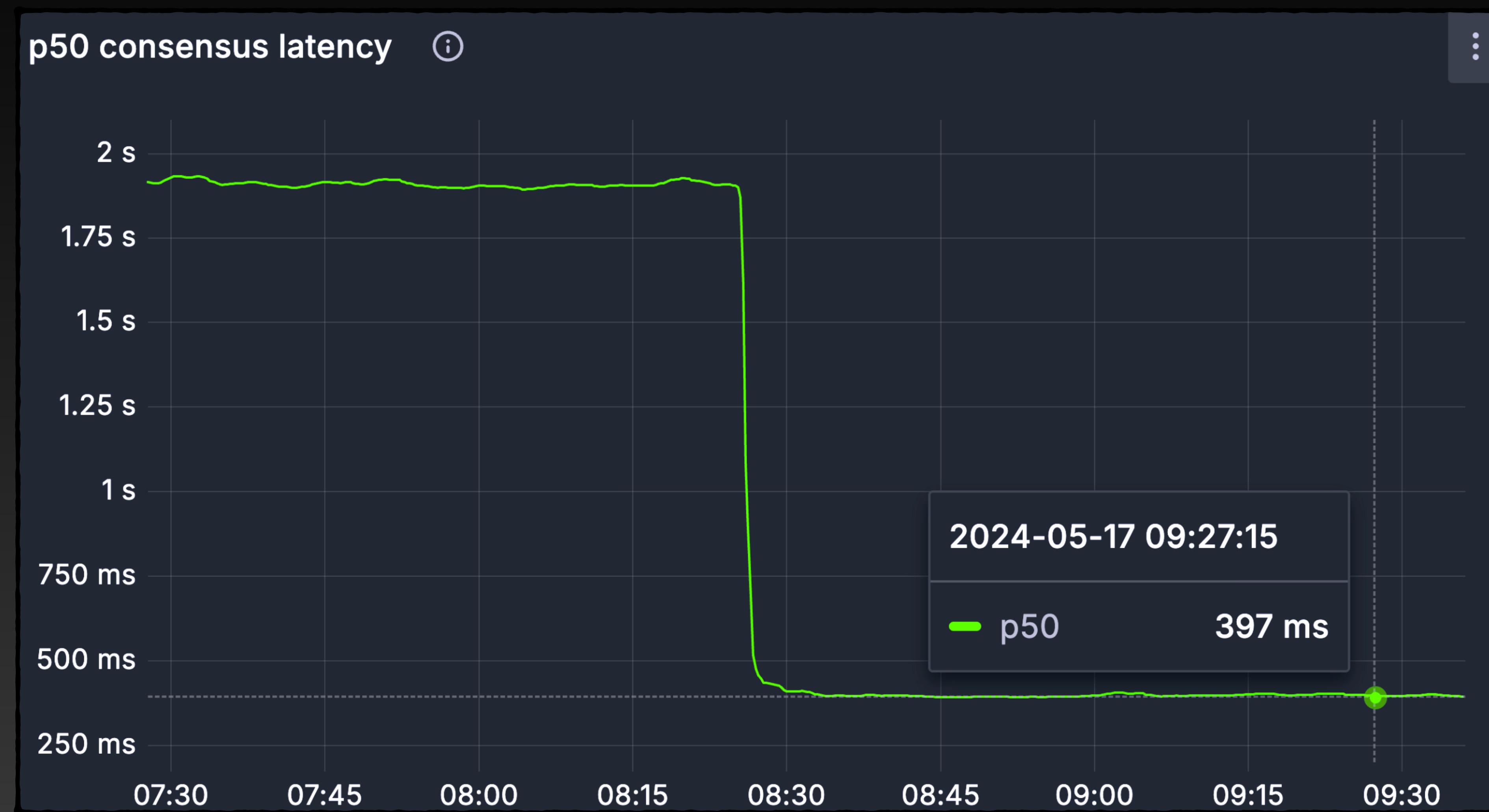
1. Modularisation is a design strategy
2. Tasks-threads allocation
3. Benchmark early
4. Codesign with mem. and storage
5. Core is hard, consensus is easy
6. Epoch change is not an add-on

# Testing Strategy

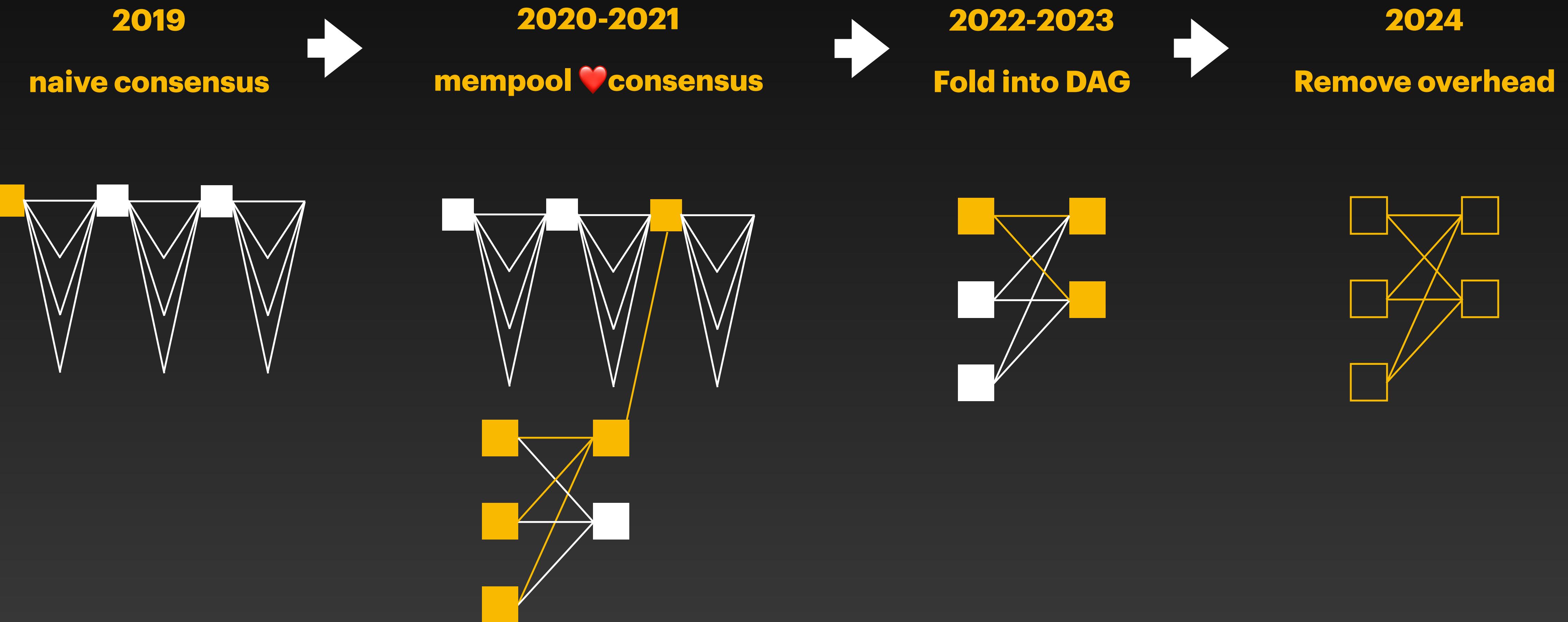


- Compare performance & robustness
- Test mainnet change bullshark -> mysticeti
- Prepare for the worst mysticeti -> bullshark

# The Sui Mainnet



# Conclusion



alberto@mystenlabs.com

# **EXTRA:**

# **Research in Industry**

# Projects Roadmap



**Dmitri Perelman** Oct 18th at 5:55 AM

In tomorrow's Research <> Core Eng syncup, [@Mark Logan](#) is going to share top of mind of Core Eng pain points and current struggles. See you 



2



# Projects Roadmap

 **Dmitri Perelman** Oct 18th at 5  
In tomorrow's Research <> C  
going to share top of mind of  
struggles. See you 

 2 

< **Thread** # sui-core-internal

 **Dmitri Perelman** Oct 18th at 5:55 AM  
In tomorrow's Research <> Core Eng syncup, [@Mark Logan](#) is  
going to share top of mind of Core Eng pain points and current  
struggles. See you 

 2 

2 replies

 **John Martin** Oct 18th at 6:16 AM  
Can I get an invite to this 

 2 

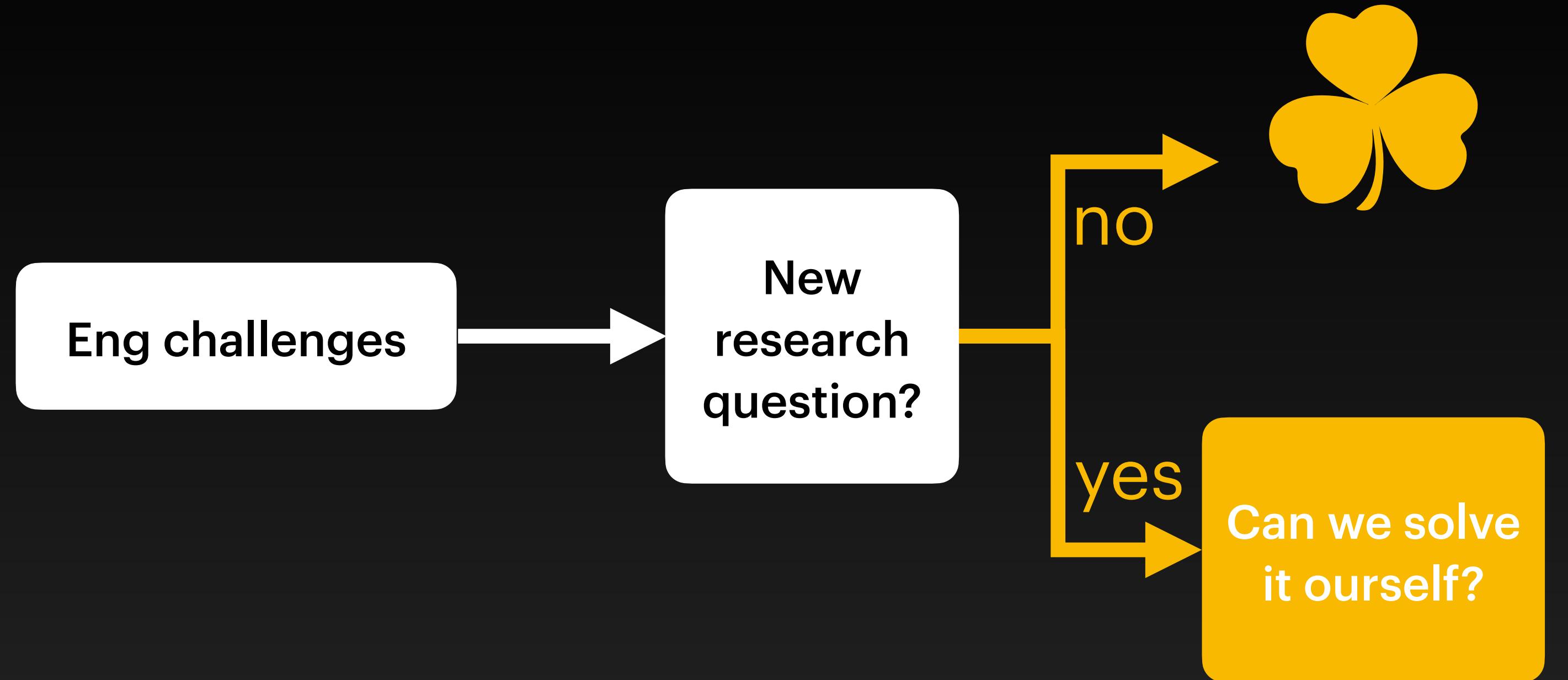
 **Dmitri Perelman** Oct 18th at 7:36 AM  
You're in the invite list!

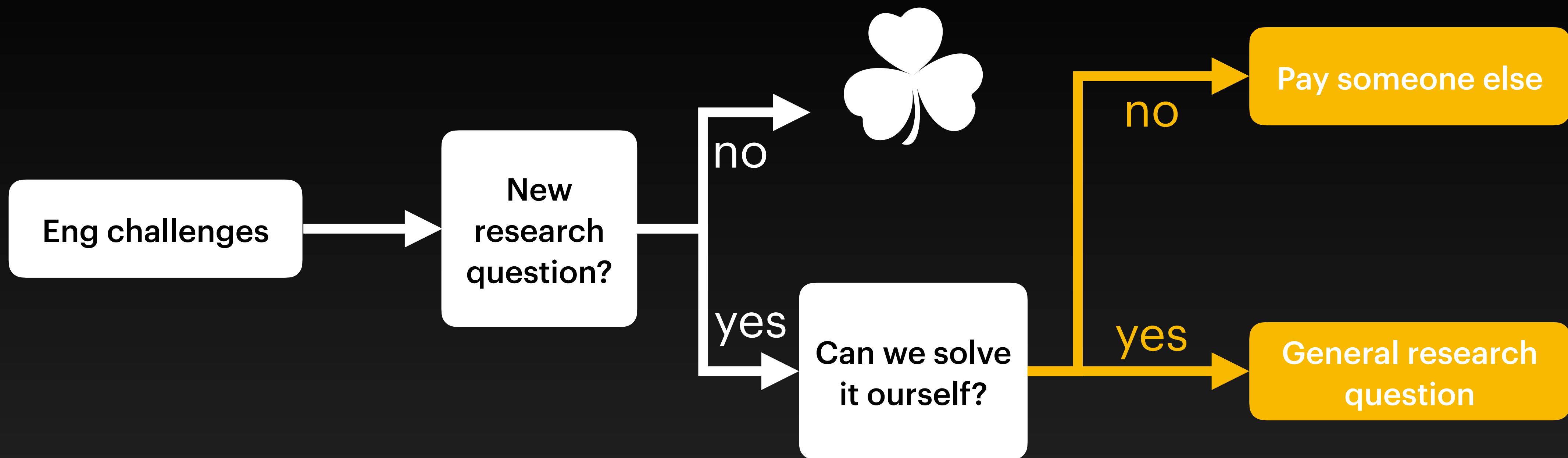
 1 

Eng challenges



New  
research  
question?

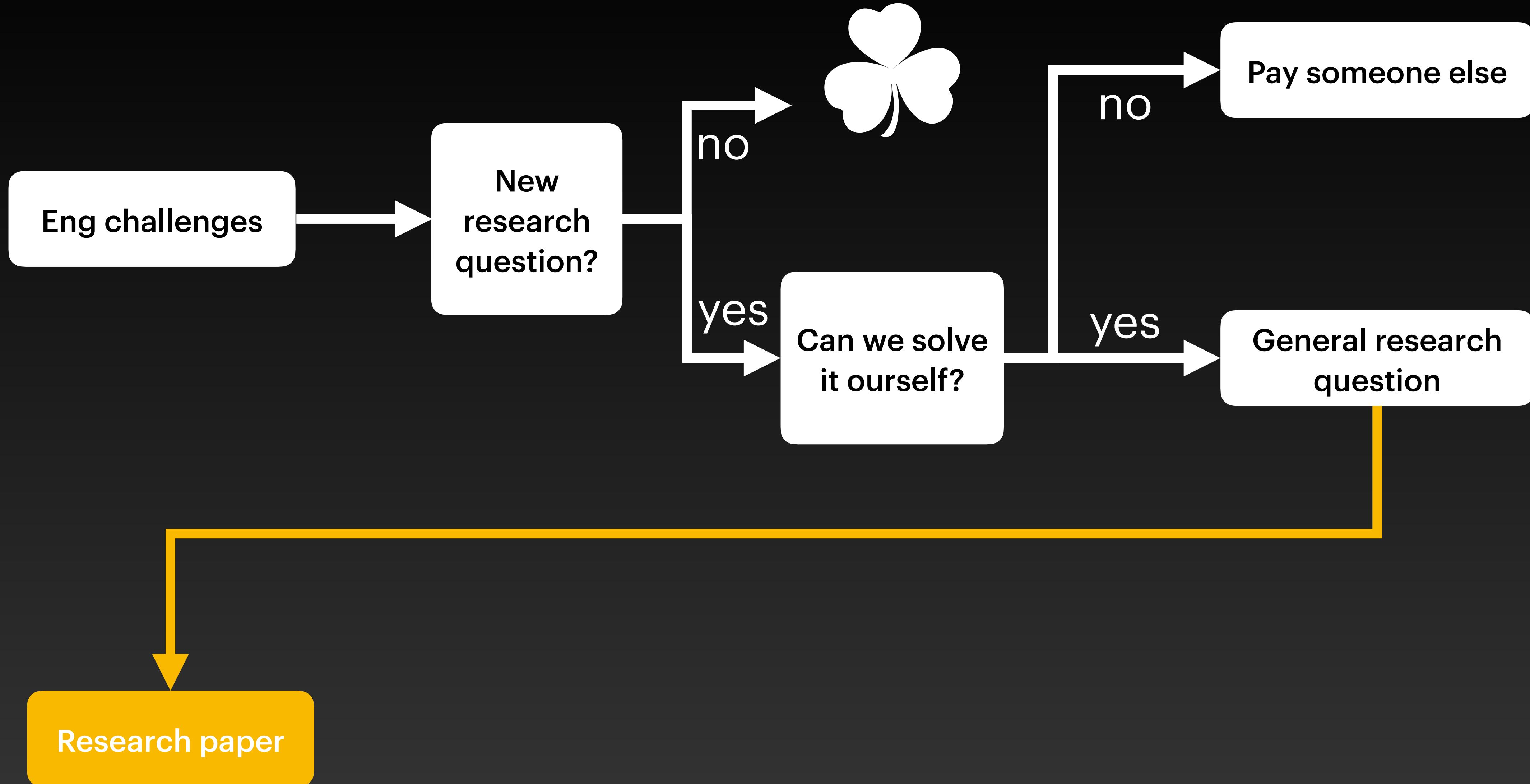


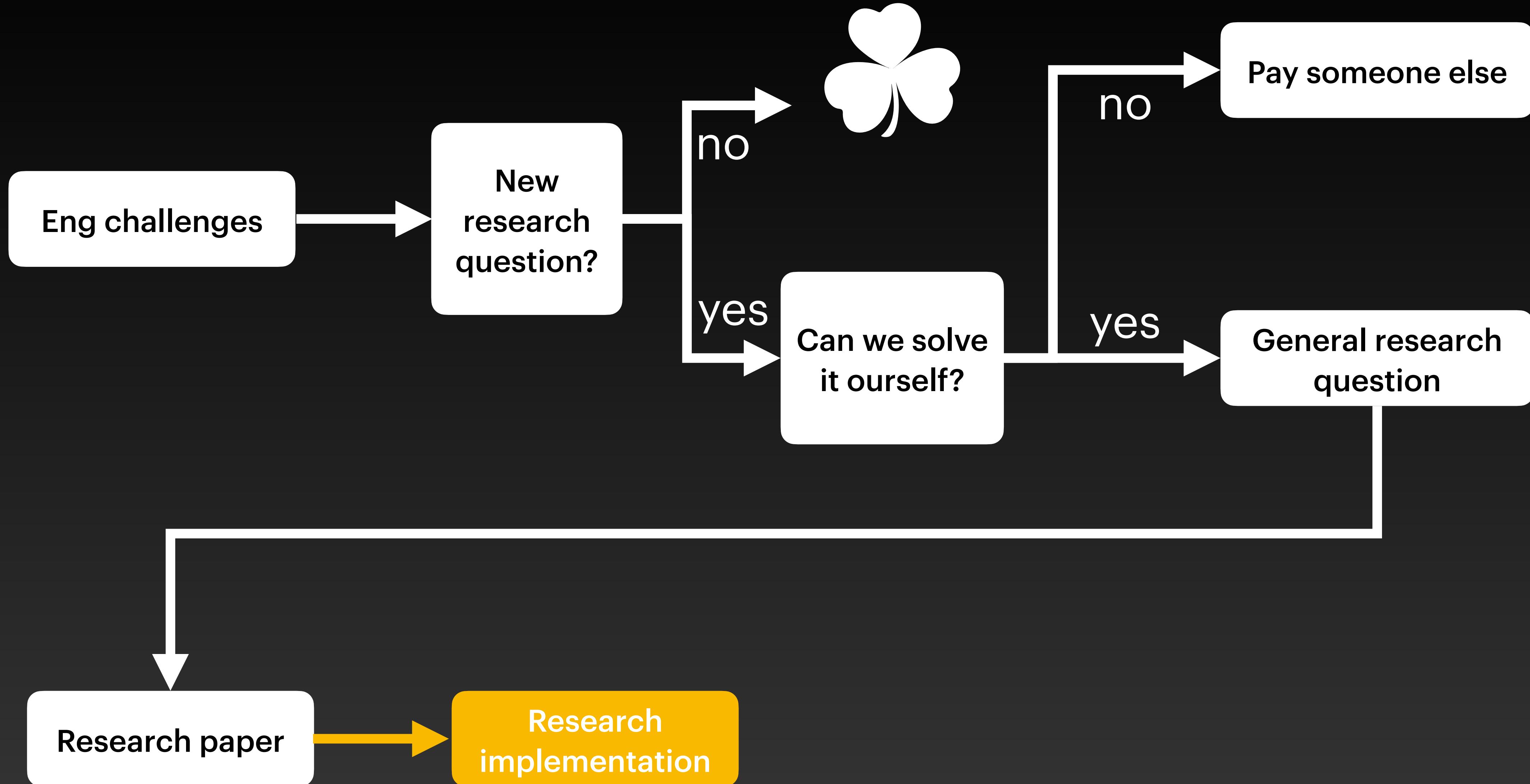


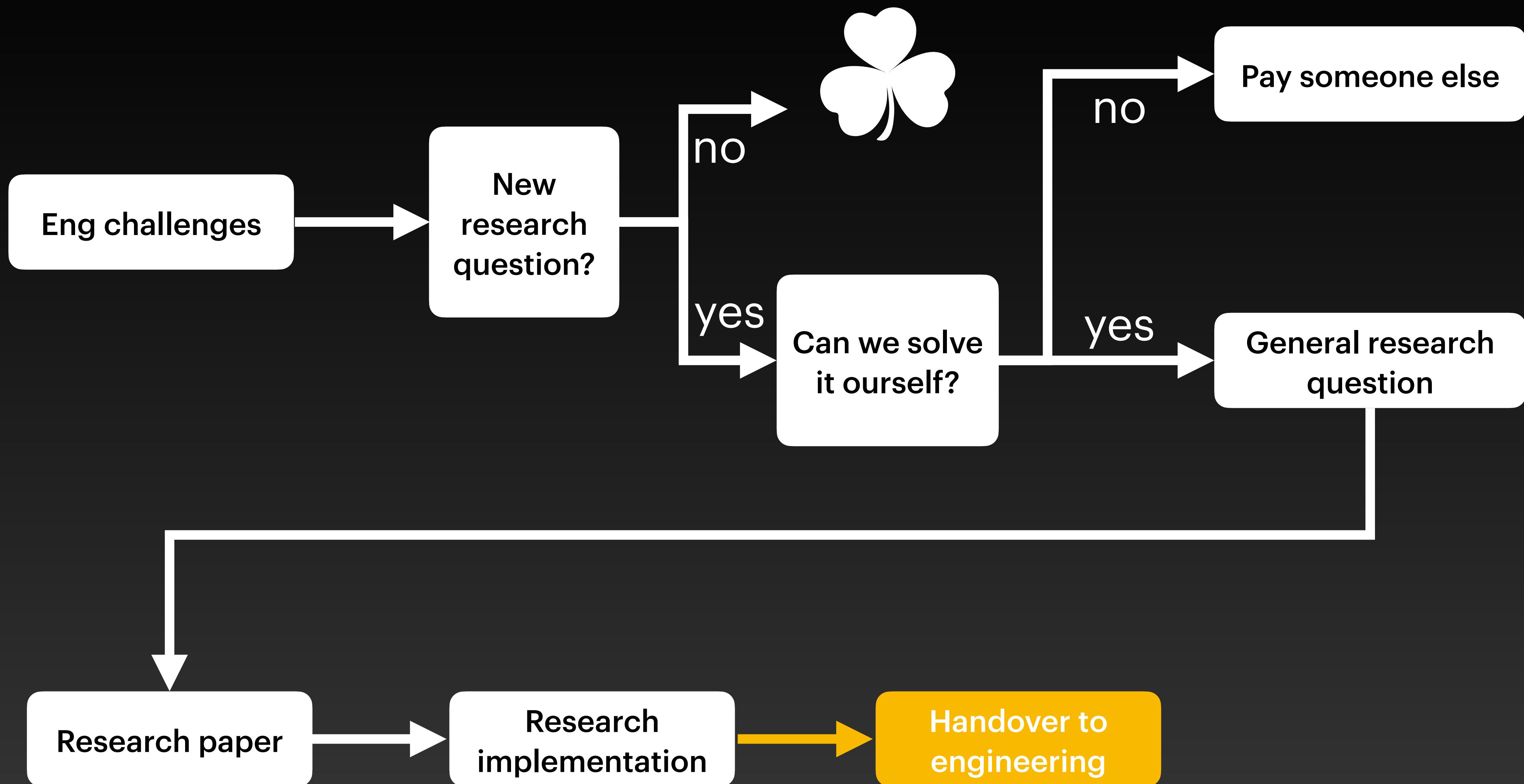
# Research Gifts

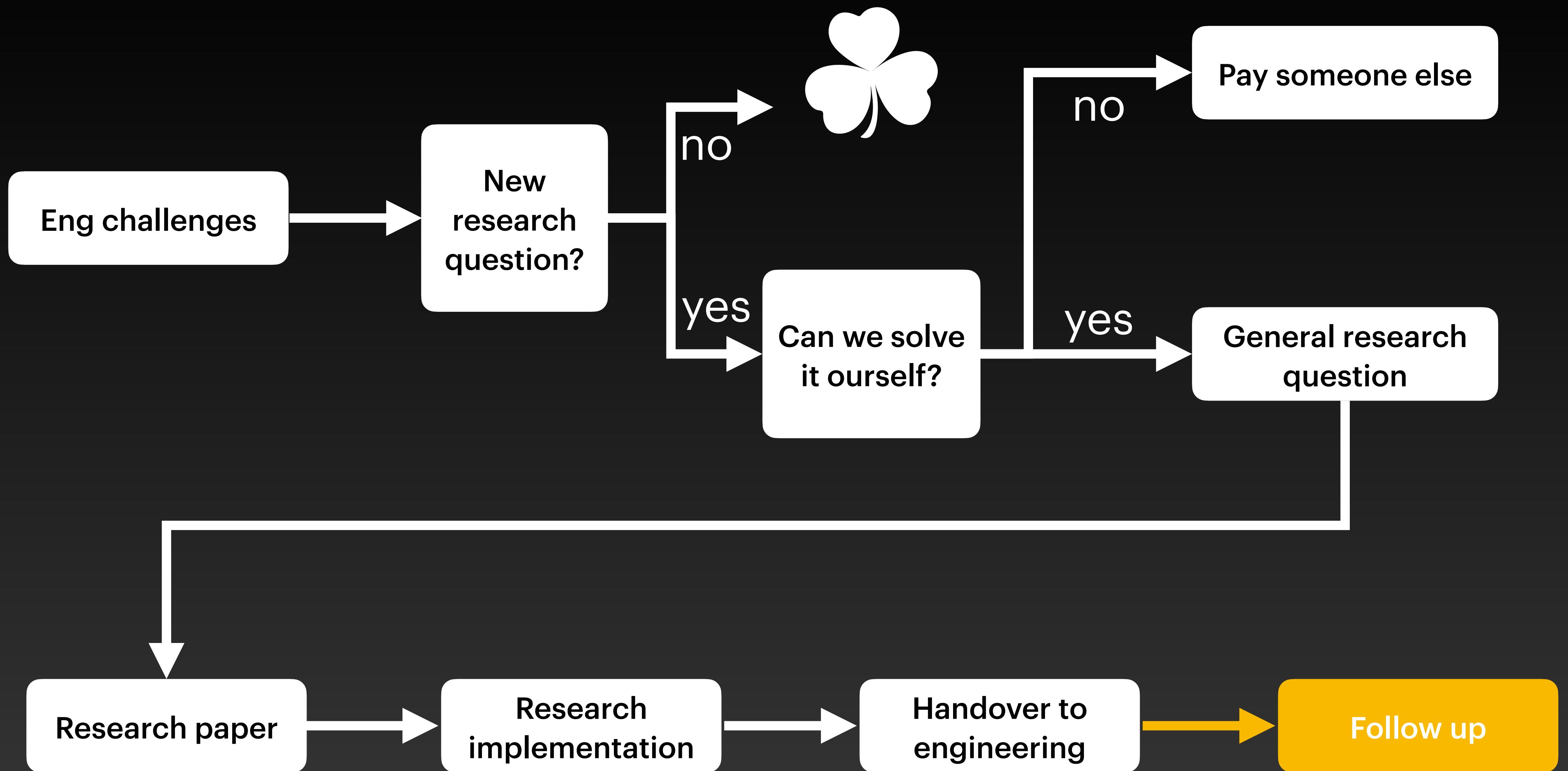


(please keep it short)









[Chainspace: A Sharded Smart Contracts Platform](#)

NDSS • Adopted by chainspace.io

[Coconut: Threshold Issuance Selective Disclosure ...](#)

NDSS • Adopted by chainspace.io, Ketl, Nym, ...

[Replay Attacks and Defenses against Cross-shard ...](#)

EuroS&P • Adopted by chainspace.io

[FastPay: High-Performance Byzantine Fault Tolerant ...](#)

AFT • Adopted by Sui, Linera

[Twins: BFT Systems Made Robust](#)

OPODIS • Adopted by Diem, Aptos, Chainlink

[Fraud Proofs: Maximising Light Client Security and ...](#)

FC • Adopted by Ethereum, Celestia

[Jolteon and Ditto: Network-Adaptive Efficient Consensus ...](#)

FC • Adopted by Flow, Diem, Aptos, Monad

[Be Aware of Your Leaders](#)

FC • Adopted by Diem, Aptos

[Subset-optimized BLS Multi-signature with Key Aggregation](#)

FC • Adopted by Fastcrypto

[Narwhal and Tusk: A DAG-based Mempool and Efficient ...](#)

EuroSys • Adopted by Sui, Aptos, Fleek, Aleo

Best paper award

[Bullshark: DAG BFT Protocols Made Practical](#)

CCS • Adopted by Sui, Aleo, Fleek

[Zef: Low-latency, Scalable, Private Payments](#)

WPES • Adopted by Linera

[Parakeet: Practical Key Transparency for End-to-End ...](#)

NDSS • Adopted by WhatsApp

IETF Applied Networking Research Prize

[HammerHead: Leader Reputation for Dynamic Scheduling](#)

ICDCS • Adopted by Sui

[Fastcrypto: Pioneering Cryptography via Continuous ...](#)

LTB • Adopted by Fastcrypto

[Sui Lutris: A Blockchain Combining Broadcast and ...](#)

CCS • Adopted by Sui

Distinguished paper award

[Mysticeti: Reaching the Limits of Latency with Uncertified ...](#)

NDSS • Adopted by Sui

# Research Questions

1. Network model?
2. BFT testing?
3. Consensus-exec interface?
4. Storage architecture?
5. Block synchroniser?
6. Realistic benchmarks?
7. Efficient reads?

# Lessons Learned

1. Modularisation is a design strategy
2. Tasks-threads relationship
3. Benchmark early
4. Codesign with mem. and storage
5. Core is hard, consensus is easy
6. Epoch change is not an add-on
7. Writing papers to explore designs

# **EXTRA:**

# Benchmarks

# Implementation

- Written in Rust
- Networking: Tokio (TCP)
- Storage: custom WAL
- Cryptography: ed25519-consensus

<https://github.com/mystenlabs/mysticeti>

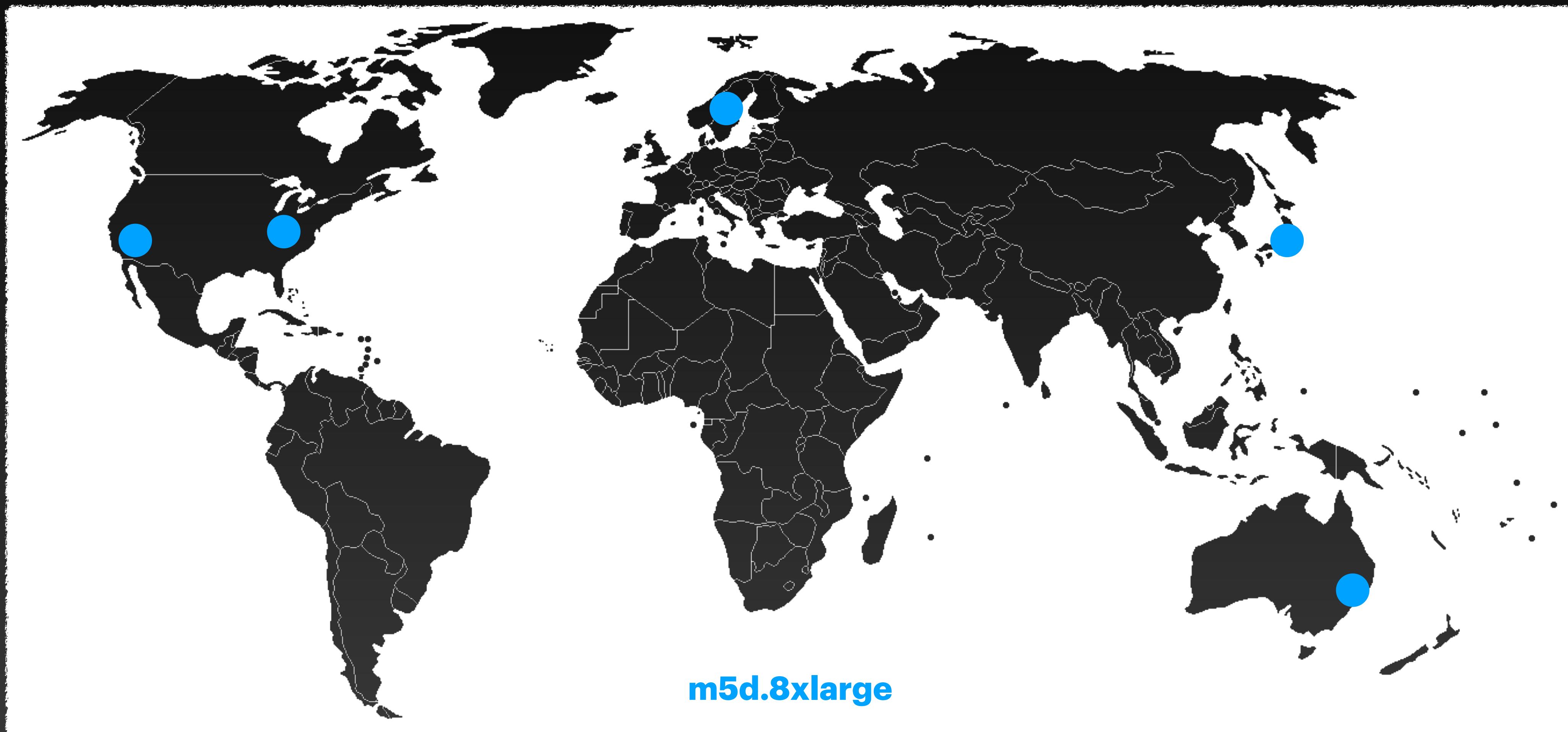
# Implementation

- Synchronous core
- One Tokio task per peer (limiting resource usage)
- DTE simulator

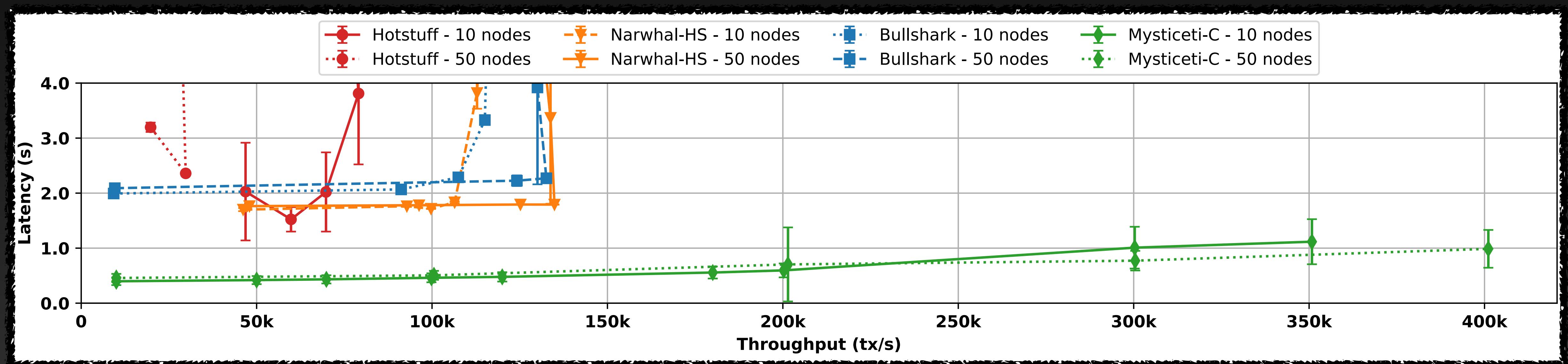
<https://github.com/mystenlabs/mysticeti>

# Evaluation

## Experimental setup on AWS



# Prototype Benchmarks



# Prototype Benchmarks

