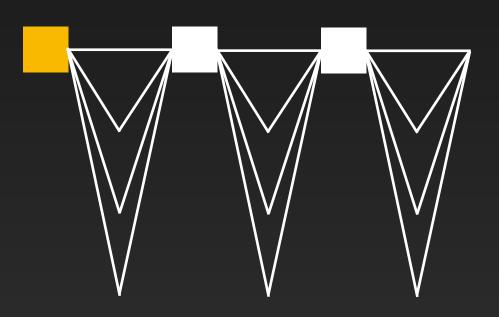
# The Evolution of Sui From Academic Paper to Mainnet



# Sibro



## HotStuff

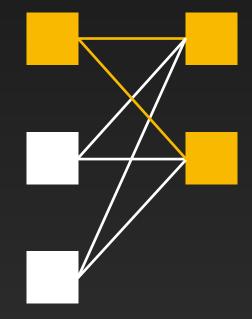








## HotStuff + Mempool



# Bullshark, Mysticeti

# Libra, 2019

## HotStuff

## 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-

## 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 clientinitiated 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,  $n \ge 3f + 1$  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

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

2019 Jul 23 DC] CS 9 .05069 803 Xiv

ar

## HashGraph

Verifying the Hashgraph Consensus Algorithm

Karl Crary Carnegie Mellon University

2021 Abstrac

eb

O

0

2

X

The Hashgraph consensus algorithm is an algorithm for asynchronous Byzantine fault tolerance intended for disributed 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 omplete discussion of the algorithm and its correctness argument in English.

## Introductio

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, 67 such as unreliable communications. The problem is at the center 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 assigned a place 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 2 position

A distributed shared ledger makes it possible for all participants to agree, at any point in the order, on the cur-rent 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 transac-. (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.

Algorithms for Byzantine consensus (under various assumptions) have existed for some time, indeed longer than the problem 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 faulttolerant state machine replication [13] and probably would

not perform well under the workload of a distributed shared

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 asynchr

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 first solve a mathematical problem that takes the list's old head as one of its inputs. 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 also 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 ransactions 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 olock-creation efforts that nearly always fail. Finally—more to the point of this paper—the theoretical properties of proof  $% \left( {{{\mathbf{F}}_{{\mathbf{F}}}} \right)$ 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 manypeers, 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

# 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 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 clientinitiated 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 eader to relay information from (n-f) replicas, each reporting its own highest known QC. Even counting just





## HotStuff

- Linear
- Clearly isolated components

## HashGraph

- K Hard to garbage collect
- X Unclear block synchroniser

# The first 6 months...

## 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

## **SMR in the Libra Blockchain**

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

<sup>\*</sup> 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

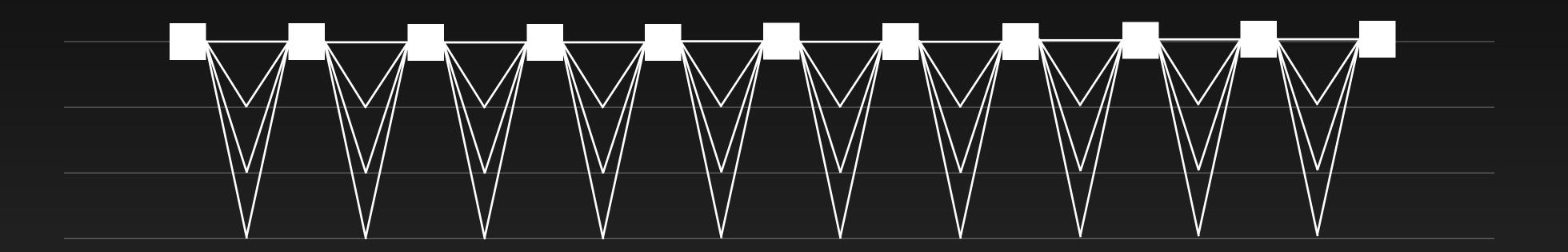
## Network model? 1.

# Lessons Learned

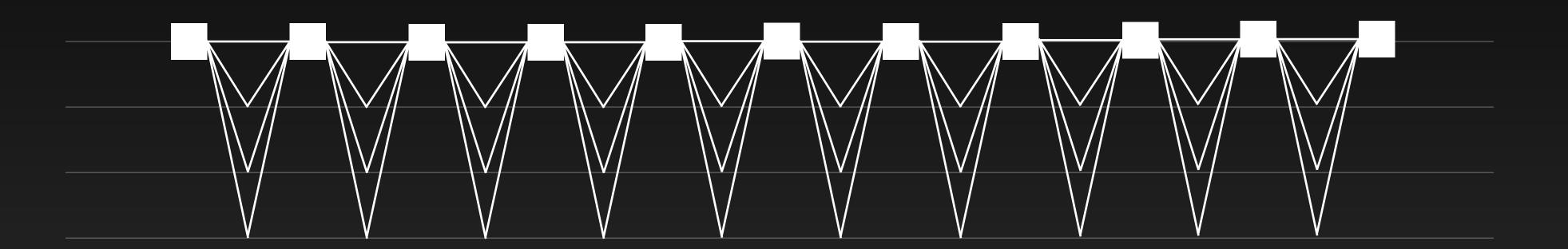
# 1. Modularisation is a design strategy



# **HotStuff** Typical leader-based protocols



# Naive Implementation Uneven resource utilisation





# Research Questions

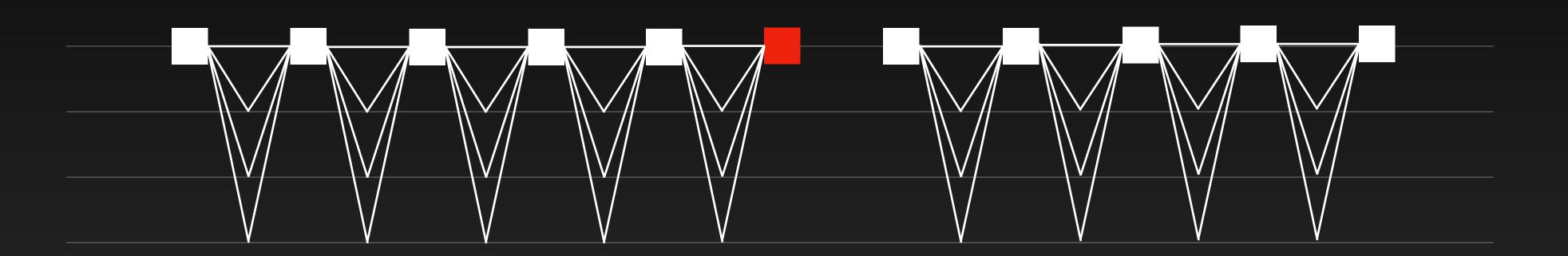
## Network model? 1.

# Lessons Learnec

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



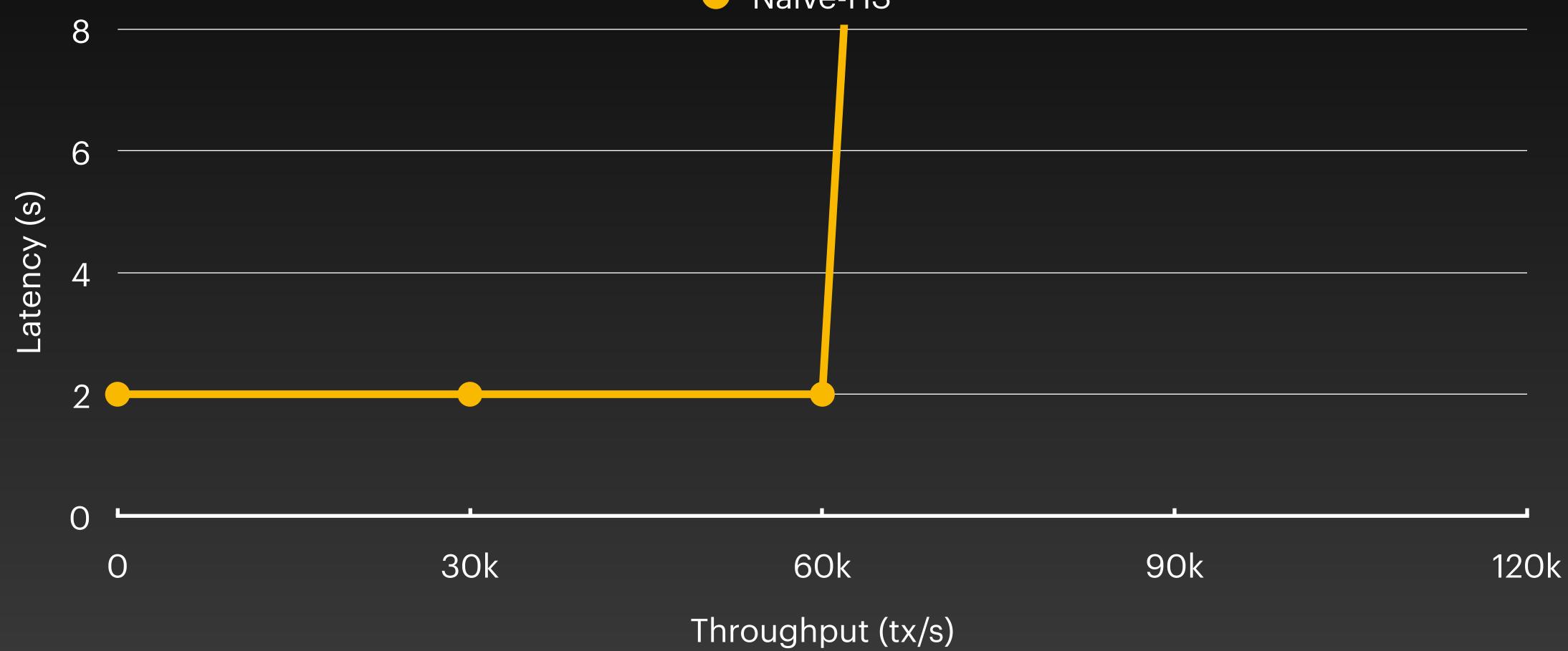
# Leader-Driven Consensus Fragility to faults and asynchrony



# Leader-Driven Consensus Fragility to faults and asynchrony

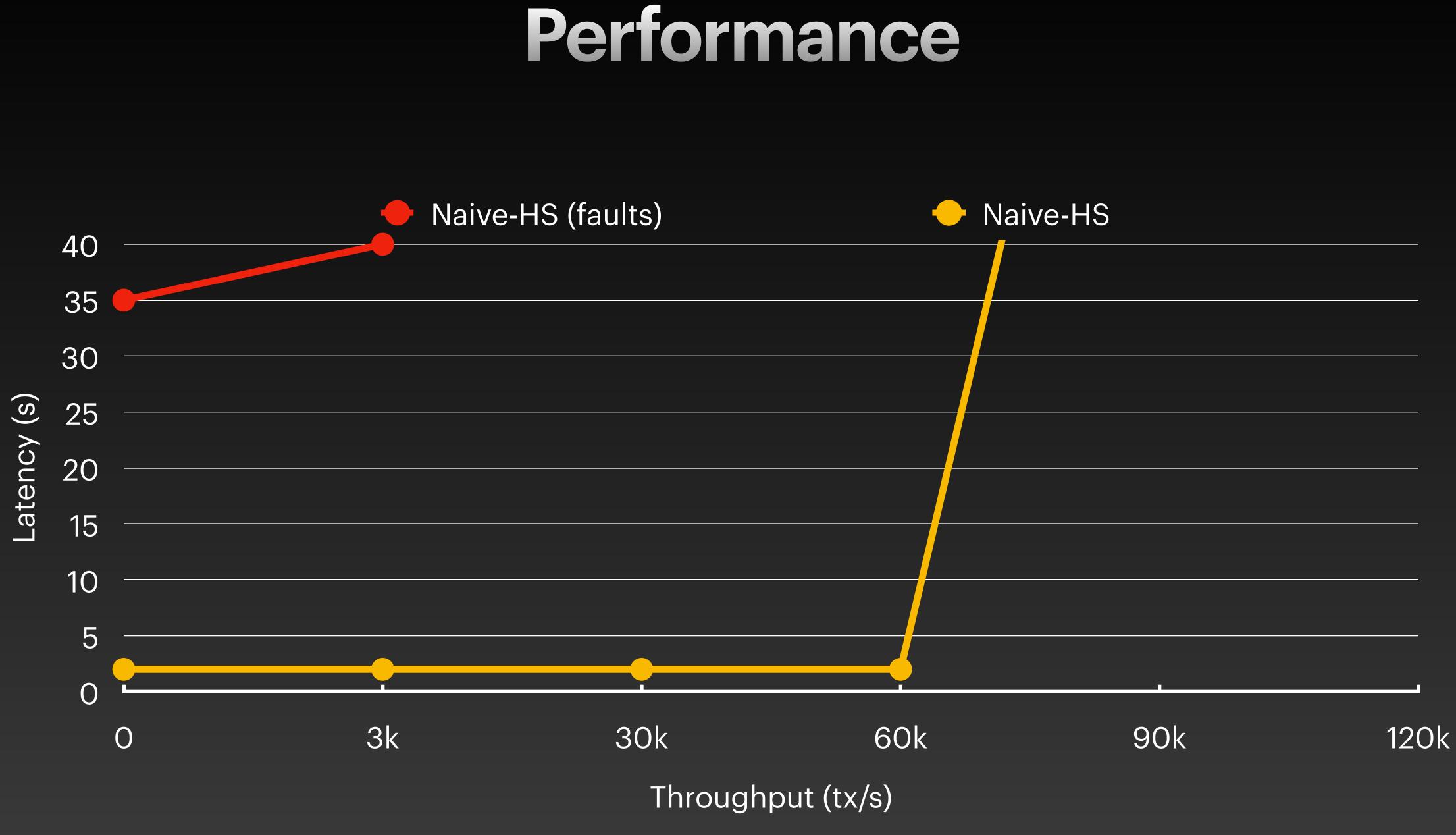






# Performance

## Naive-HS



# Research Questions

## Network model? 1.

# Lessons Learnec

- Modularisation is a design strategy 1.
- 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 failover (vs. cubic with BFT-SMaRt).

## 1 Introduction

By zantine fault tolerance (BFT) refers to the ability of a computing system to endure arbitrary (i.e., By zantine) 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 clientinitiated service commands, despite the efforts of f By zantine 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



- HotStuff (naive mempool)
  - Linear

ullet

- 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  $\rightarrow$  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 r 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. 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/10.1145/3492321.351959

Lefteris Kokoris-Kogias IST Austria

Alexander Spiegelman Aptos

## ACM Reference Format:

George Danezis, Lefteris Kokoris-Kogias, Alberto Sonnino, and Alexan der 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.

## Introduction

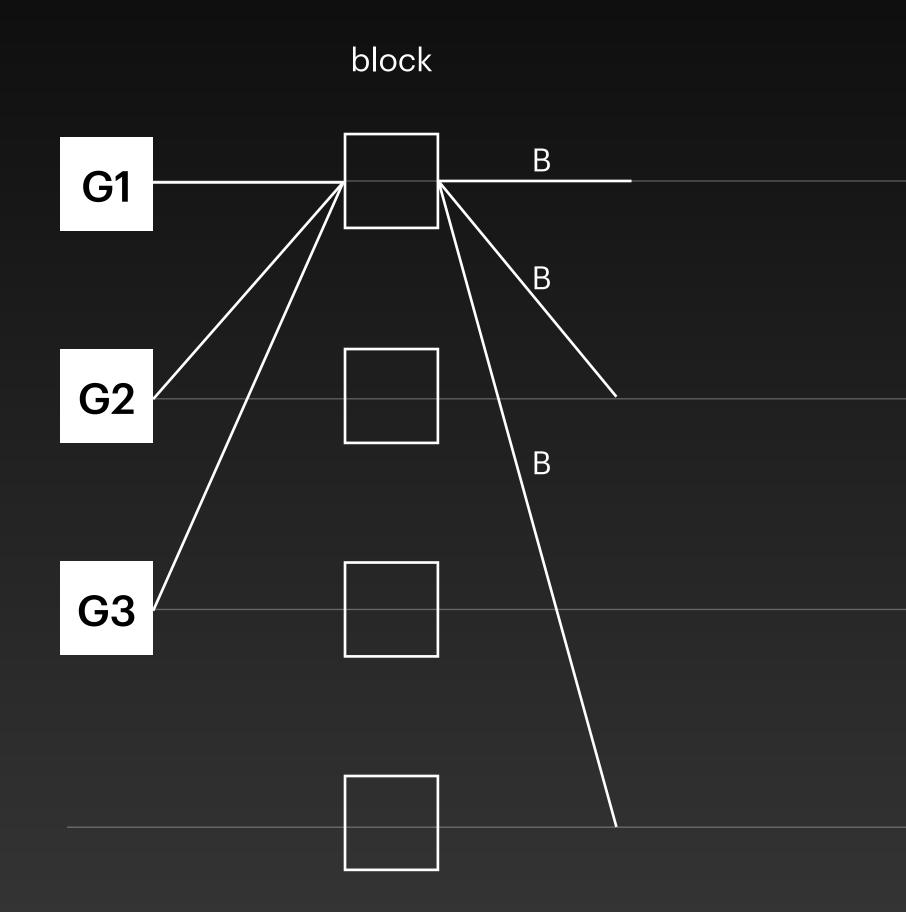
Byzantine consensus protocols [15, 19, 21] and the state machine replication paradigm [13] for building reliable distributed systems have been studied for over 40 years. However, with the rise in popularity of blockchains there has been a renewed interest in engineering high-performance consensus protocols. 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.

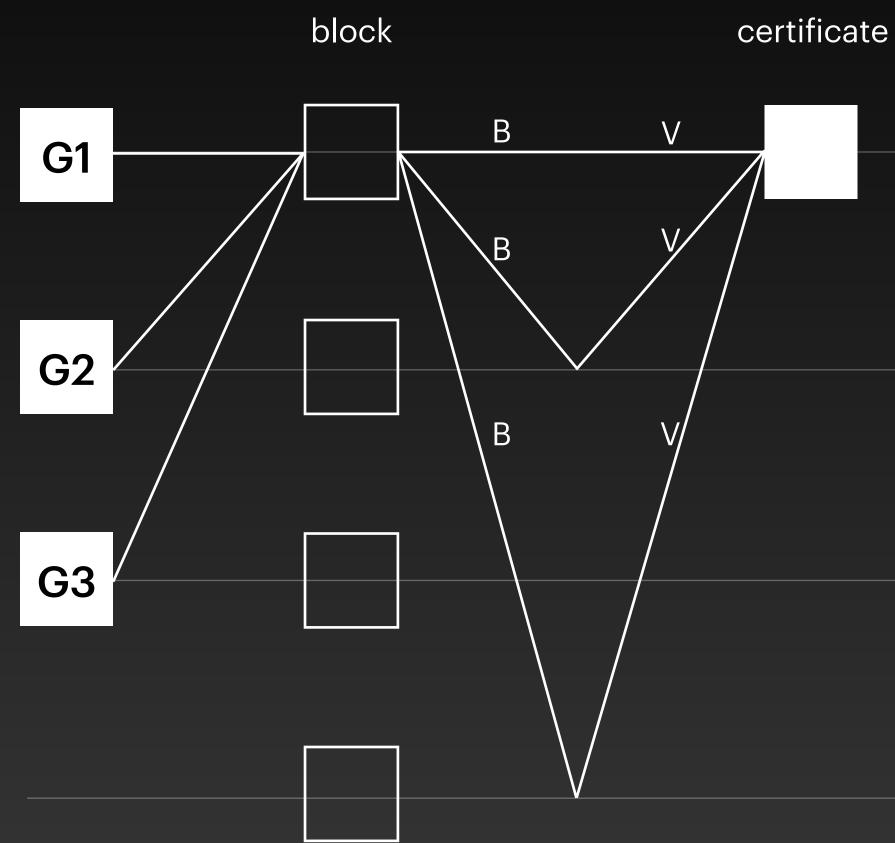
Existing approaches to increasing the performance of distributed ledgers focus on creating lower-cost consensus algorithms culminating with Hotstuff [38], which achieves linear message complexity in the partially synchronous setting. To achieve this, Hotstuff leverages a leader who collects, aggregates,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

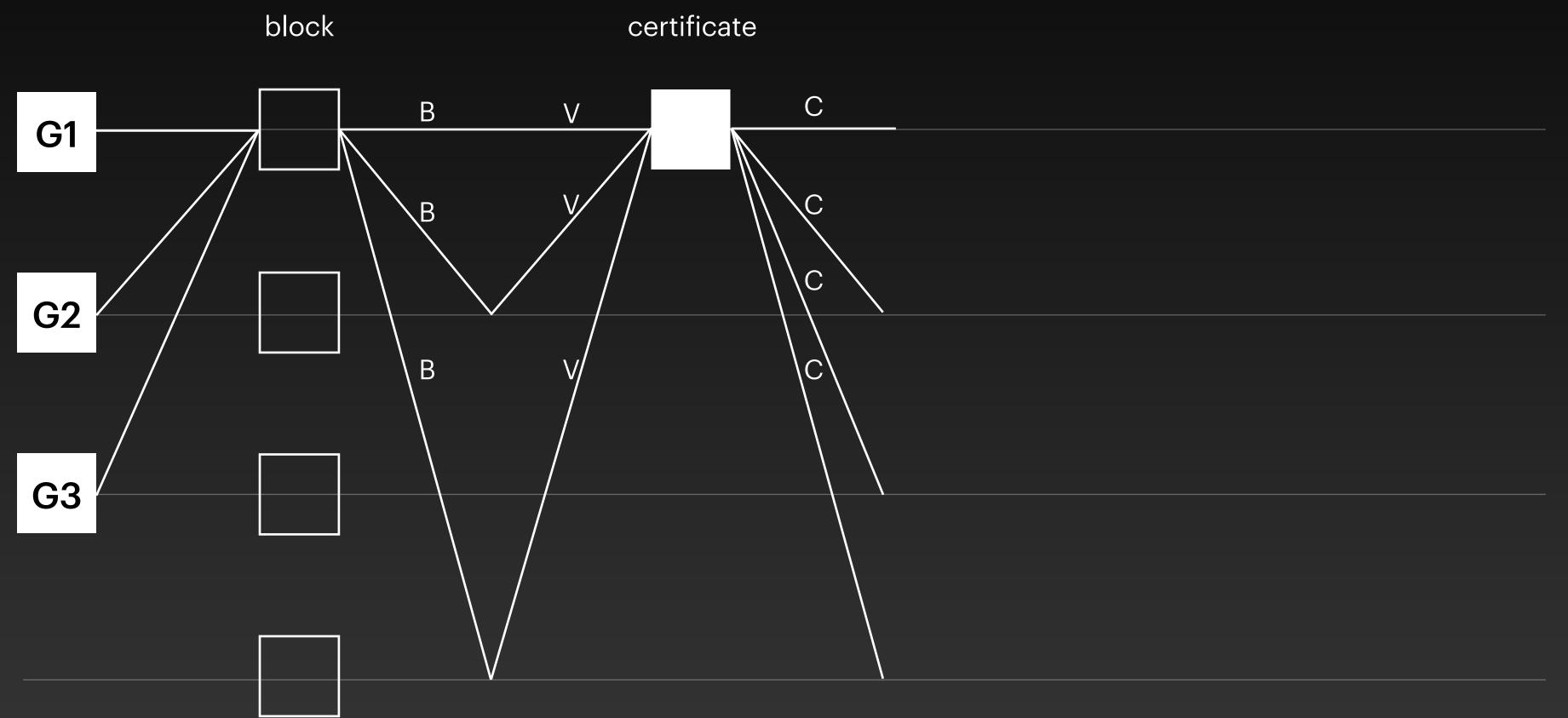
- Any (partially-synchronous) protocol that minimizes overall message number, but relies on a leader to produce proposals and coordinate consensus, fails to capture the high load this imposes on the leader who inevitably becomes a bottleneck.
- Message complexity counts the number of metadata messages (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  $\sim$  50 nodes).

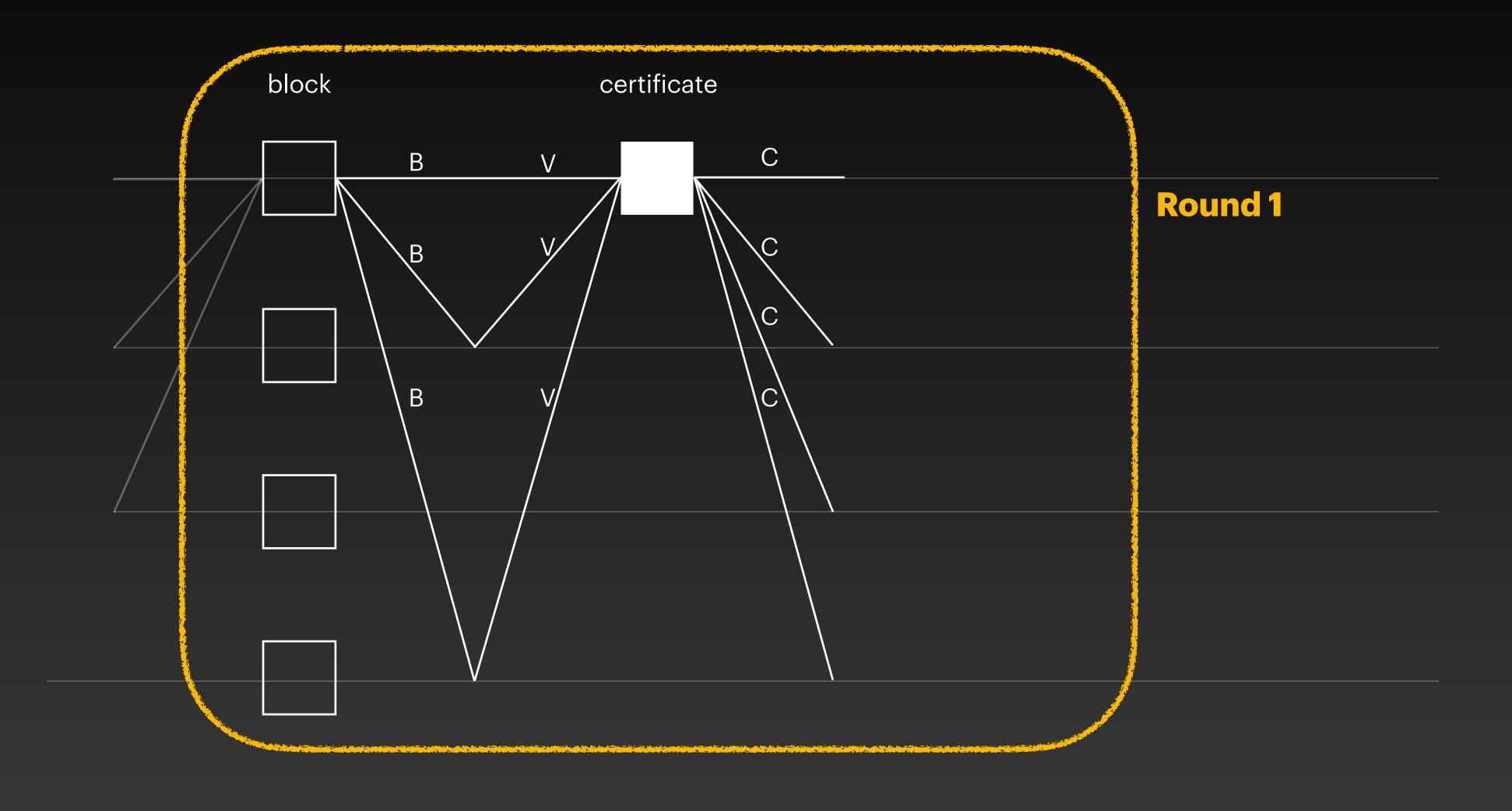
Additionally, consensus protocols have grouped a lot of functions into a monolithic protocol. In a typical distributed

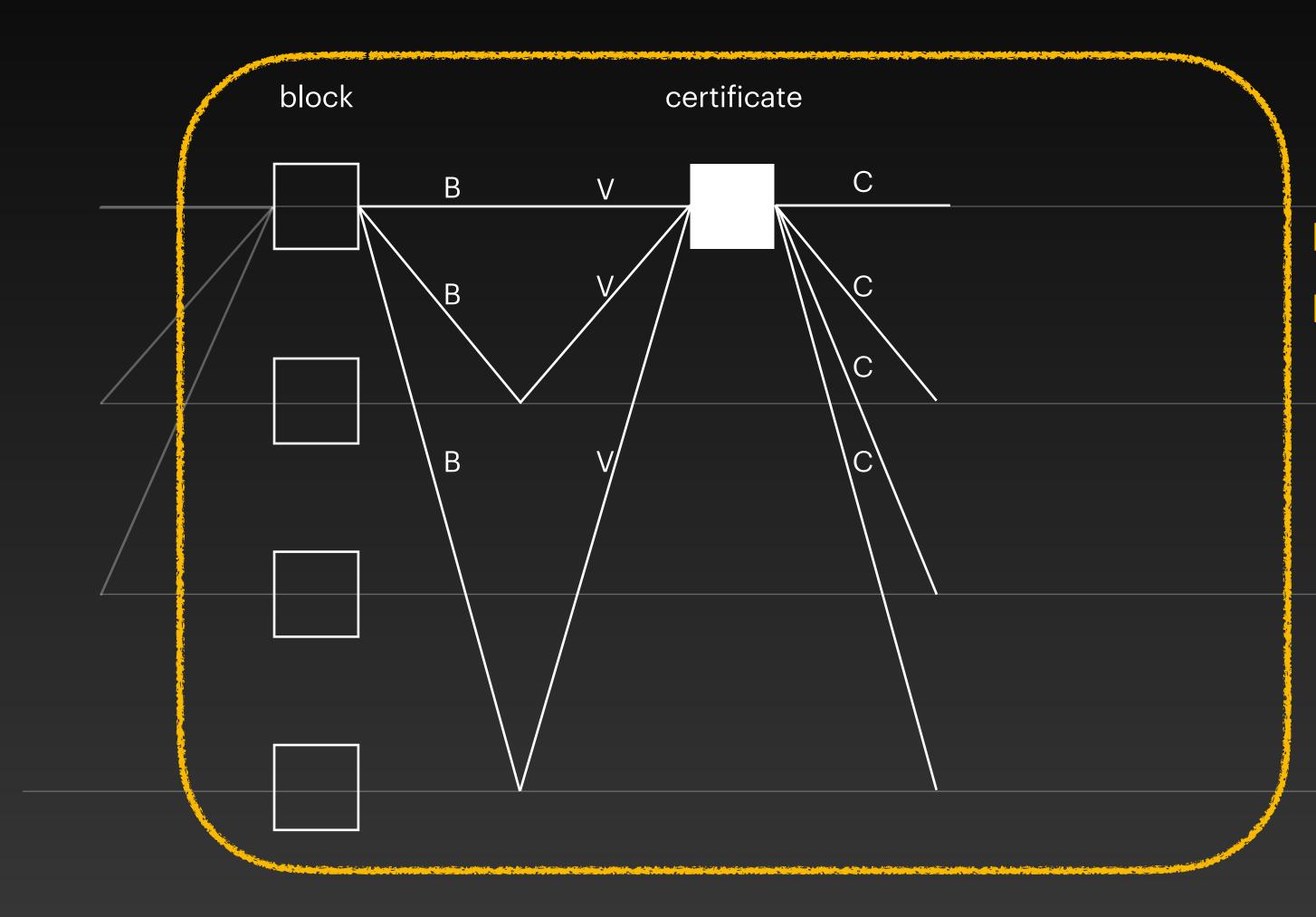
- Quadratic but even resource utilisation
- Separation between consensus and data dissemination







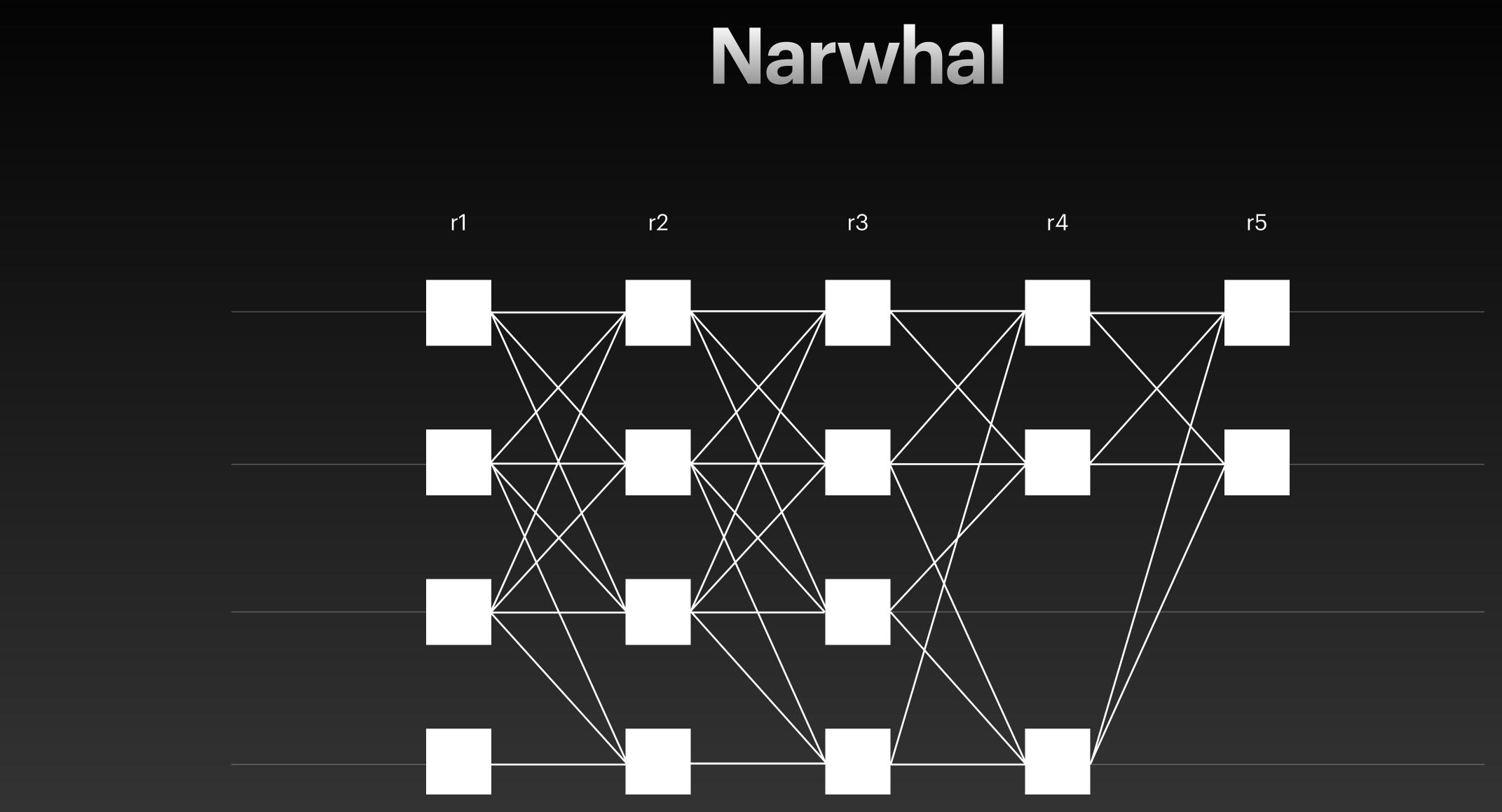




Round 1

**Byzantine 'Reliable' Broadcast** 



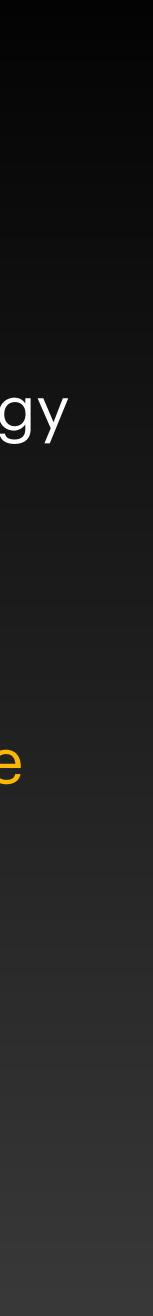


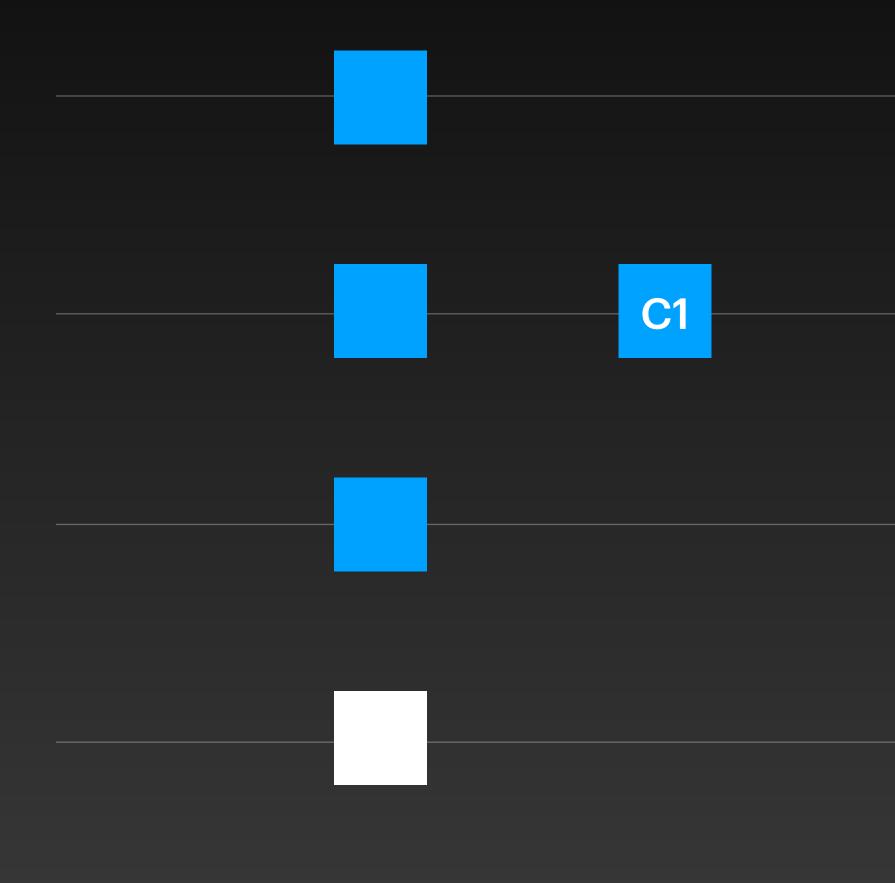
# Research Questions

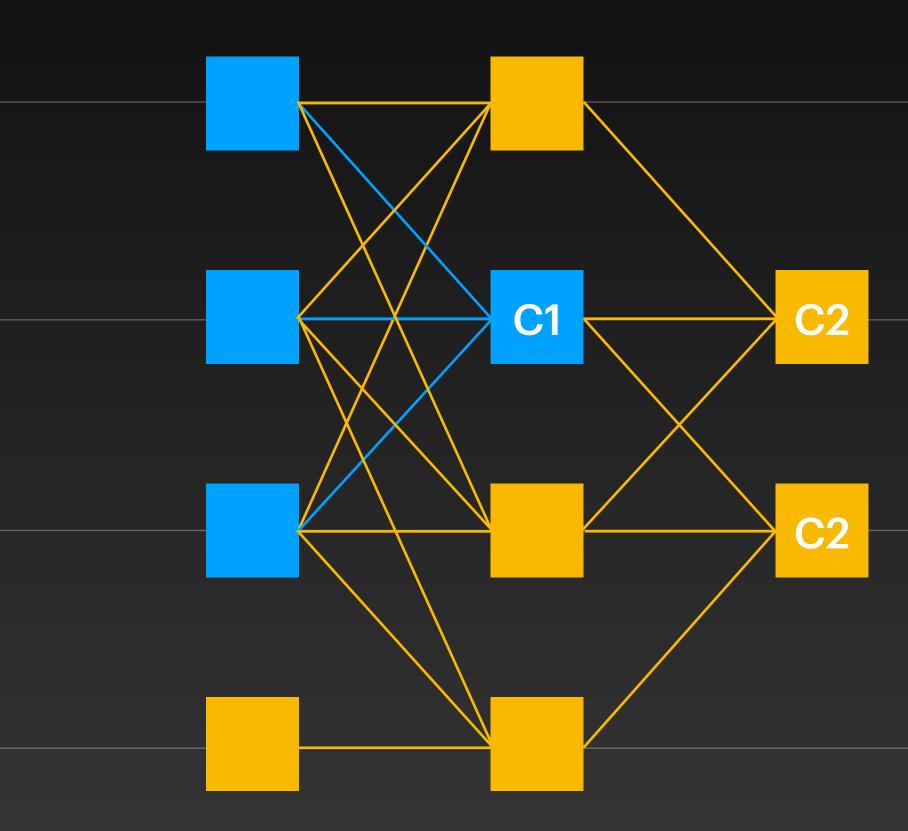
# 1. Network model?

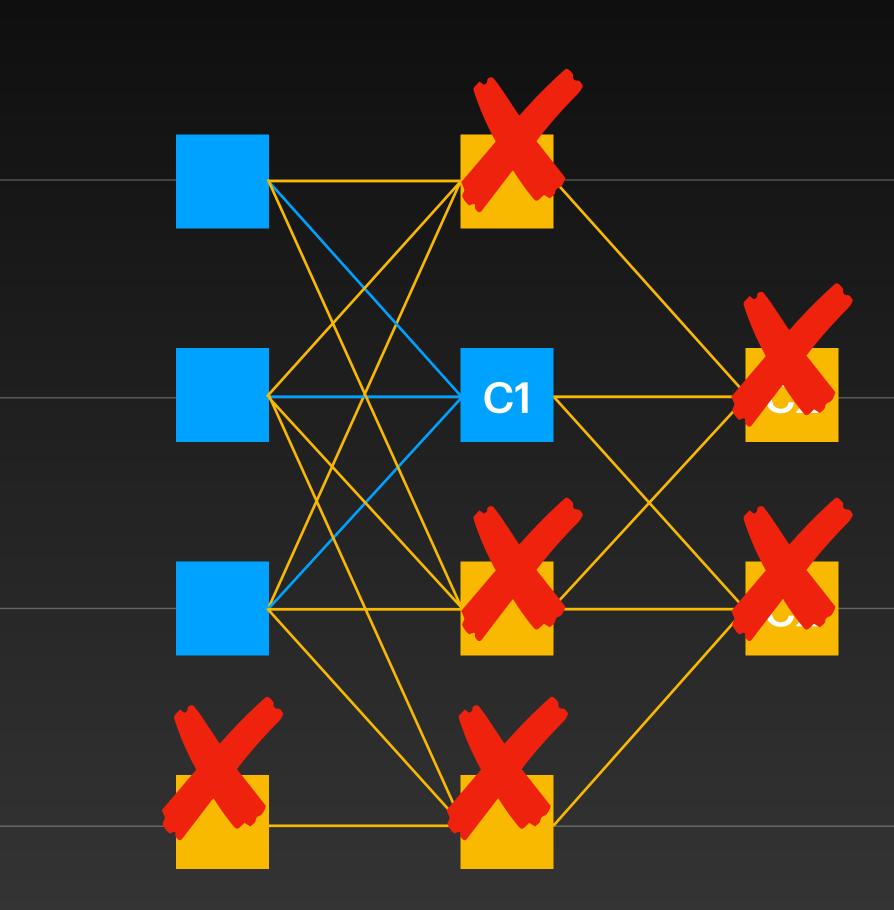
# Lessons Learned

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



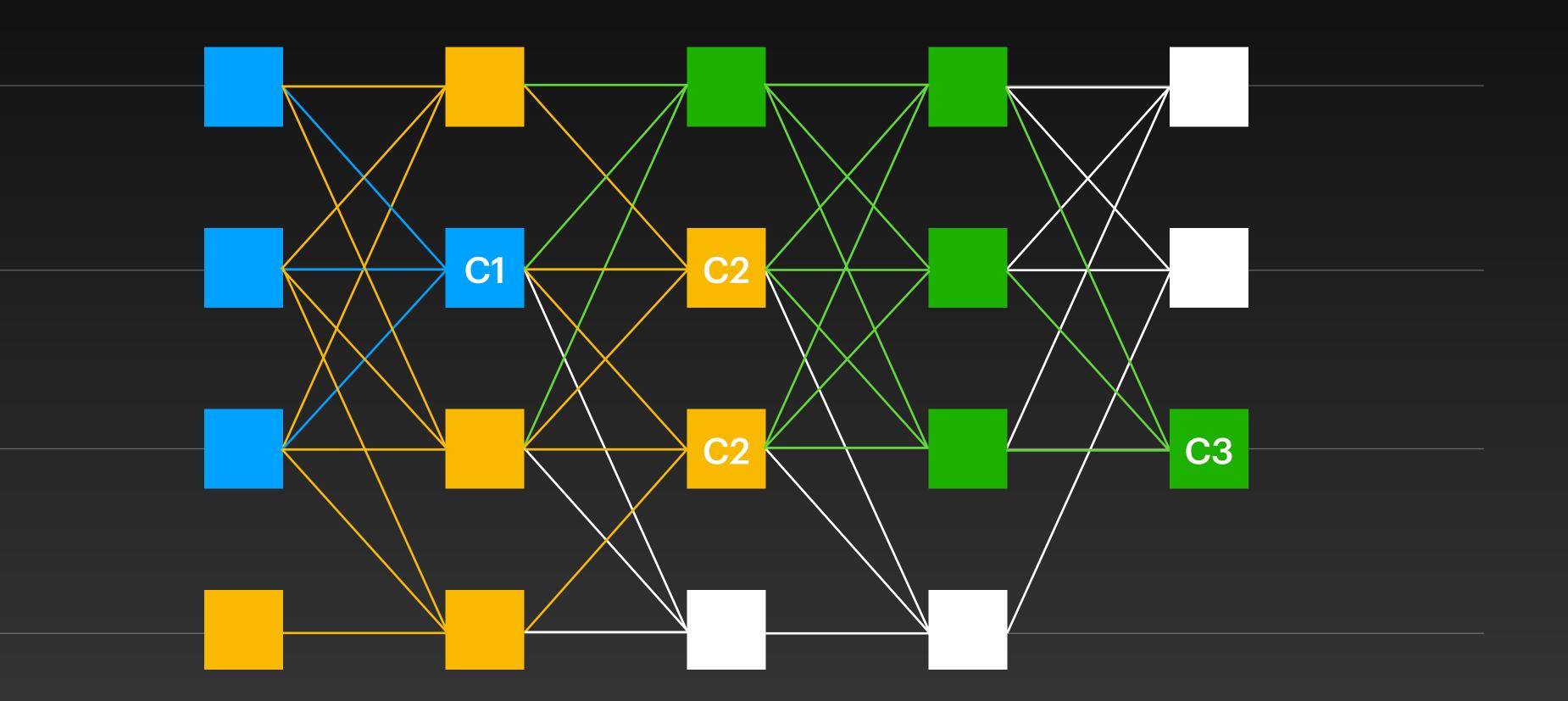


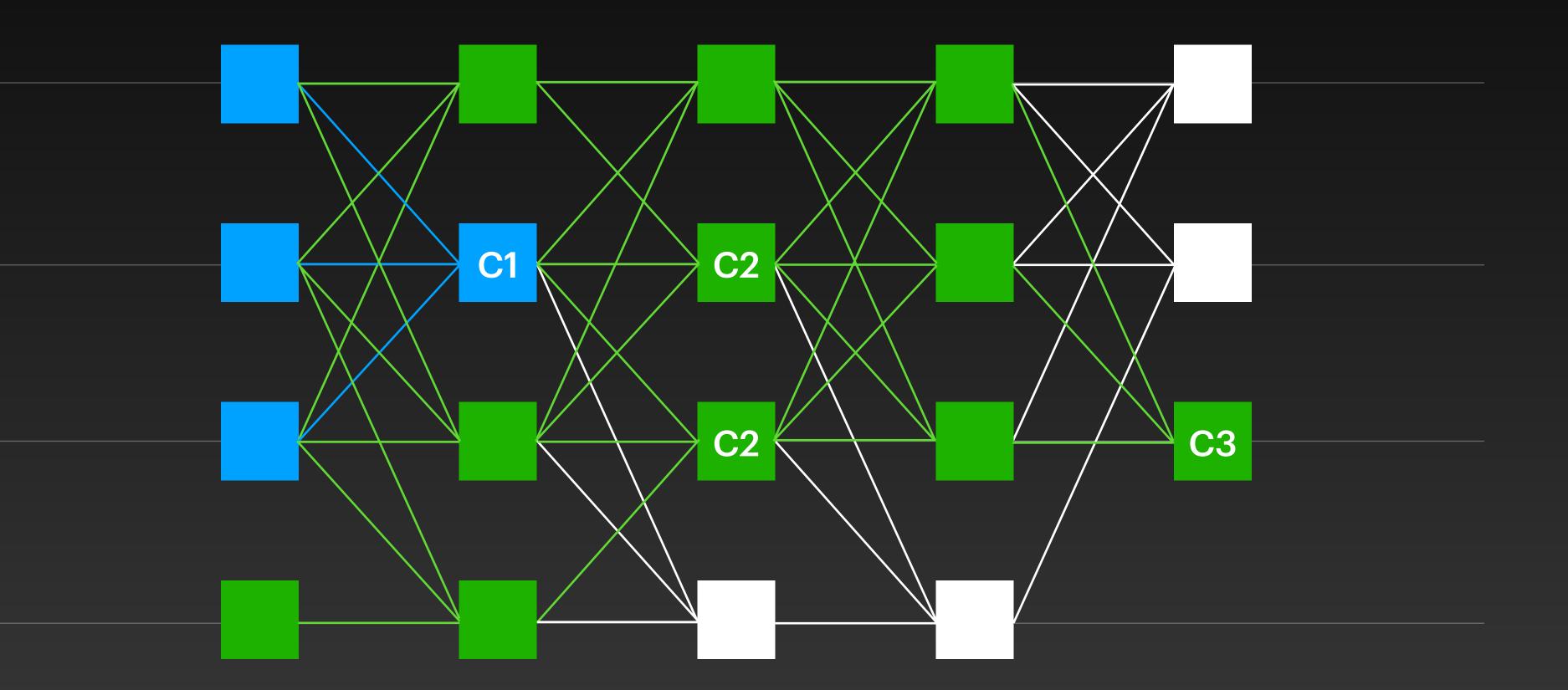


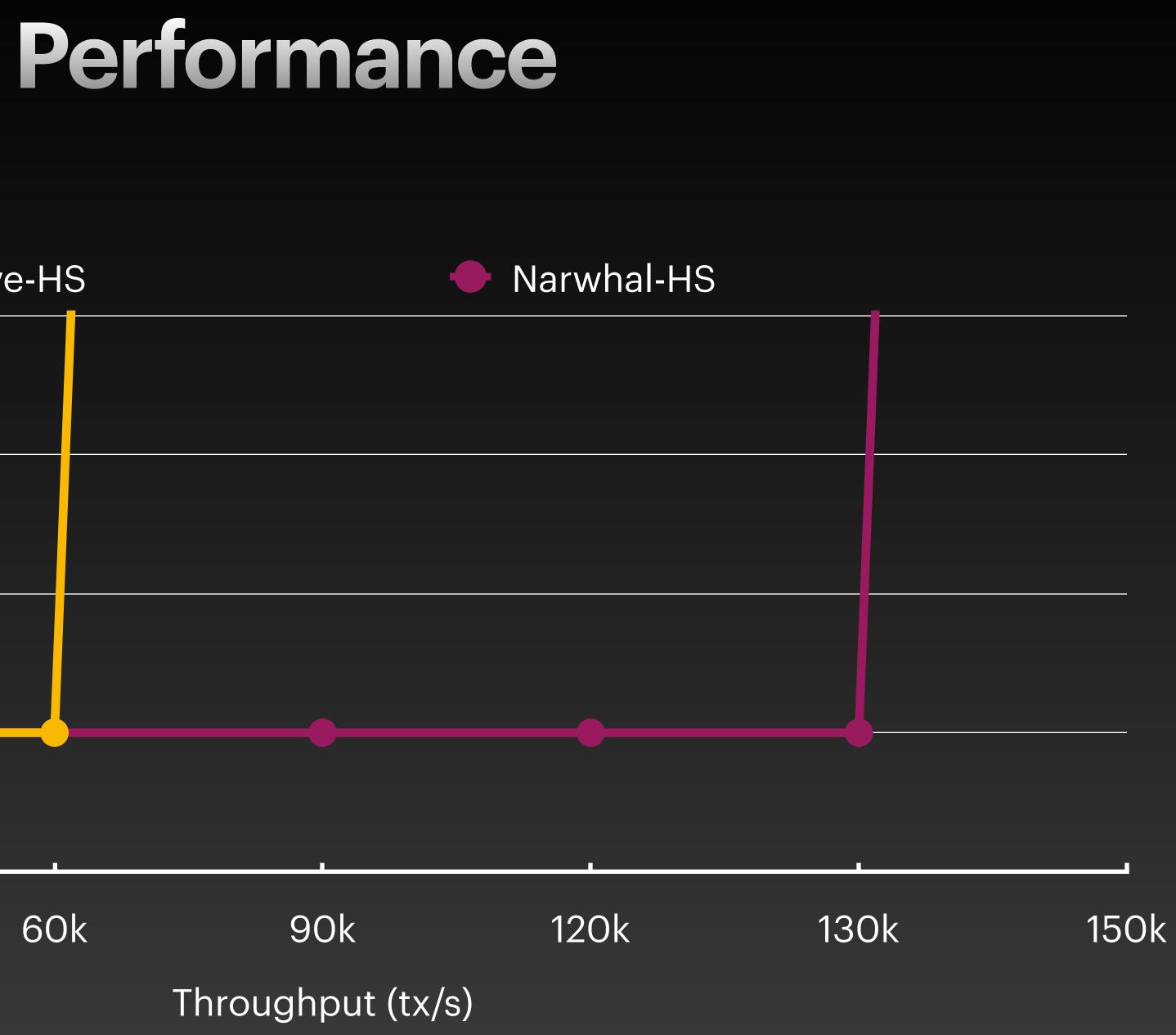


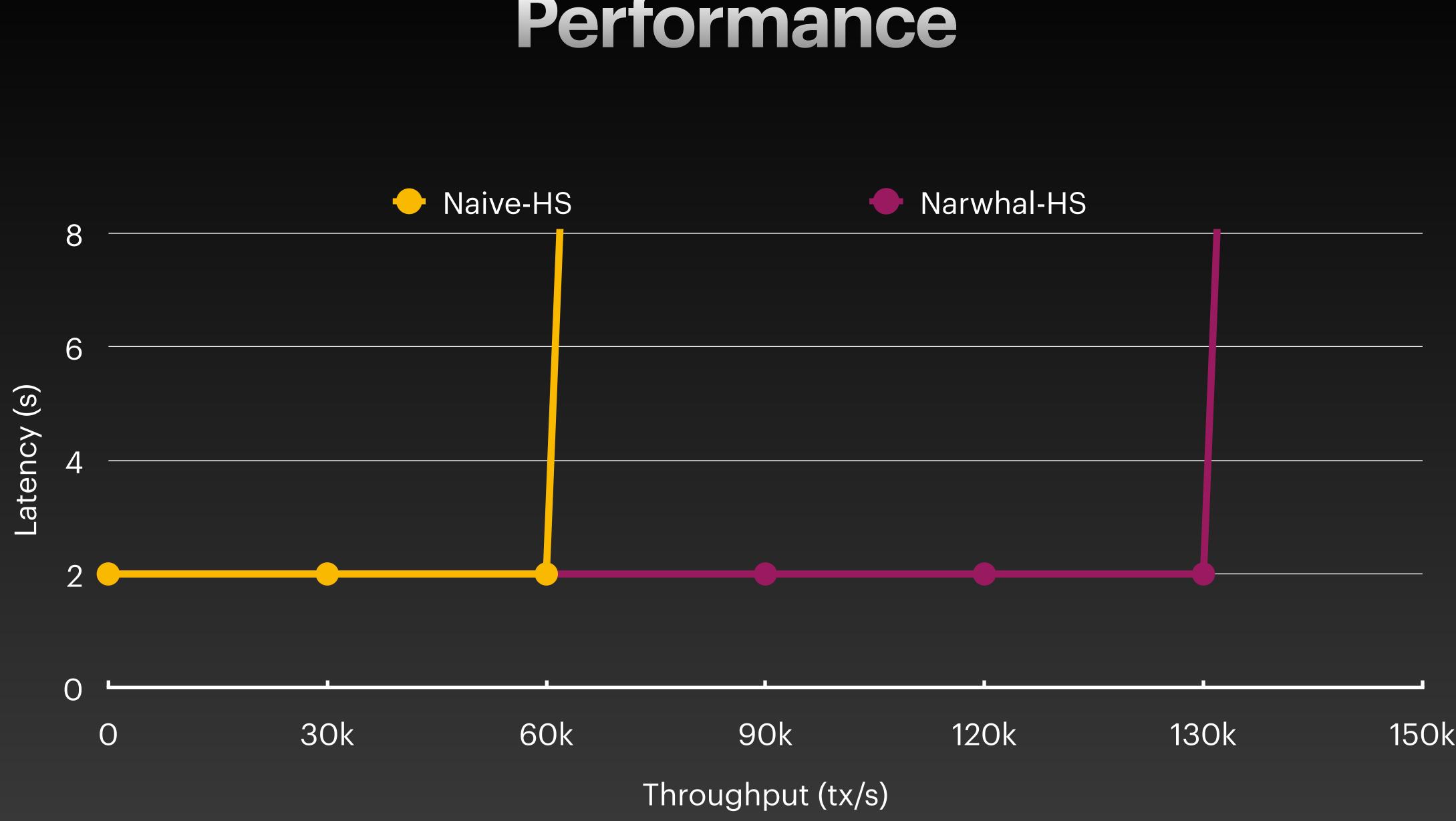
Faulty HotStuff Leader!

**Blocks may still be 'saved'** 

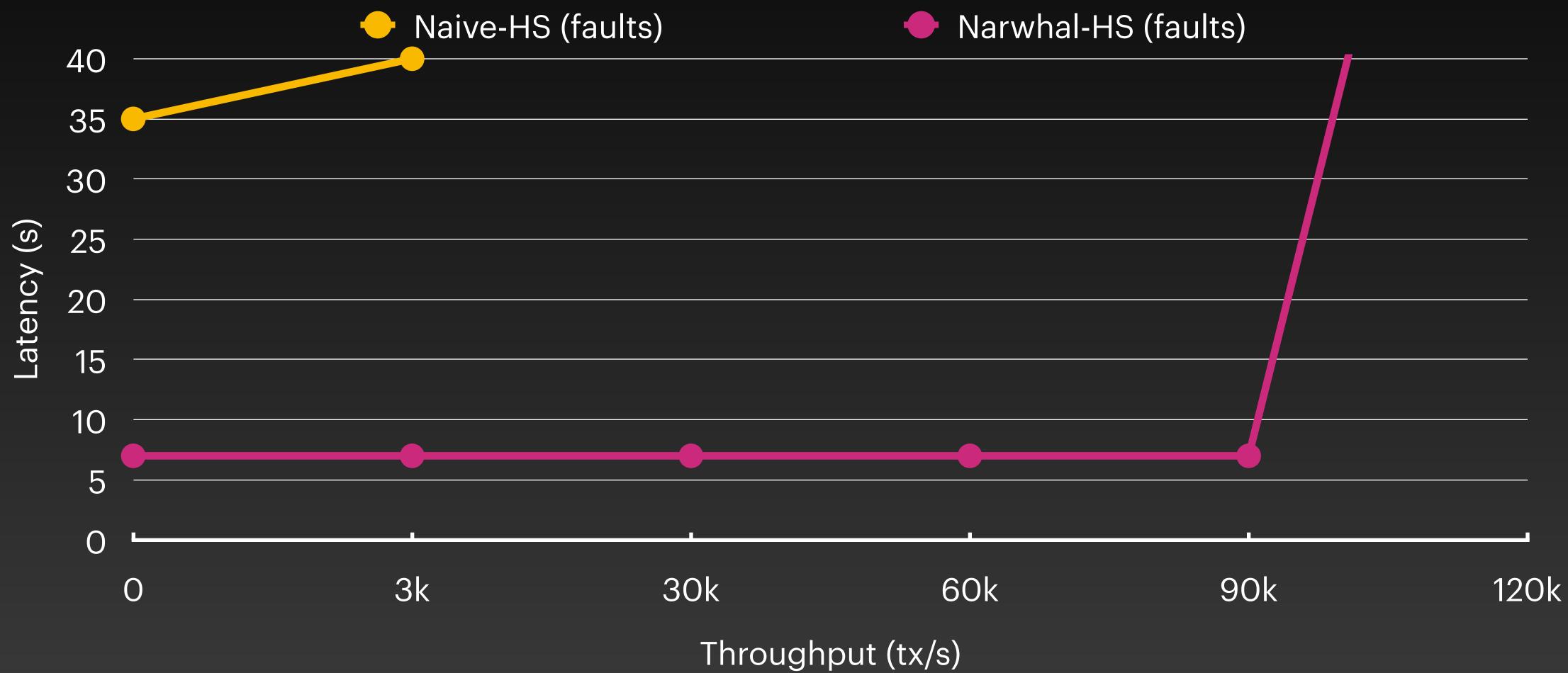




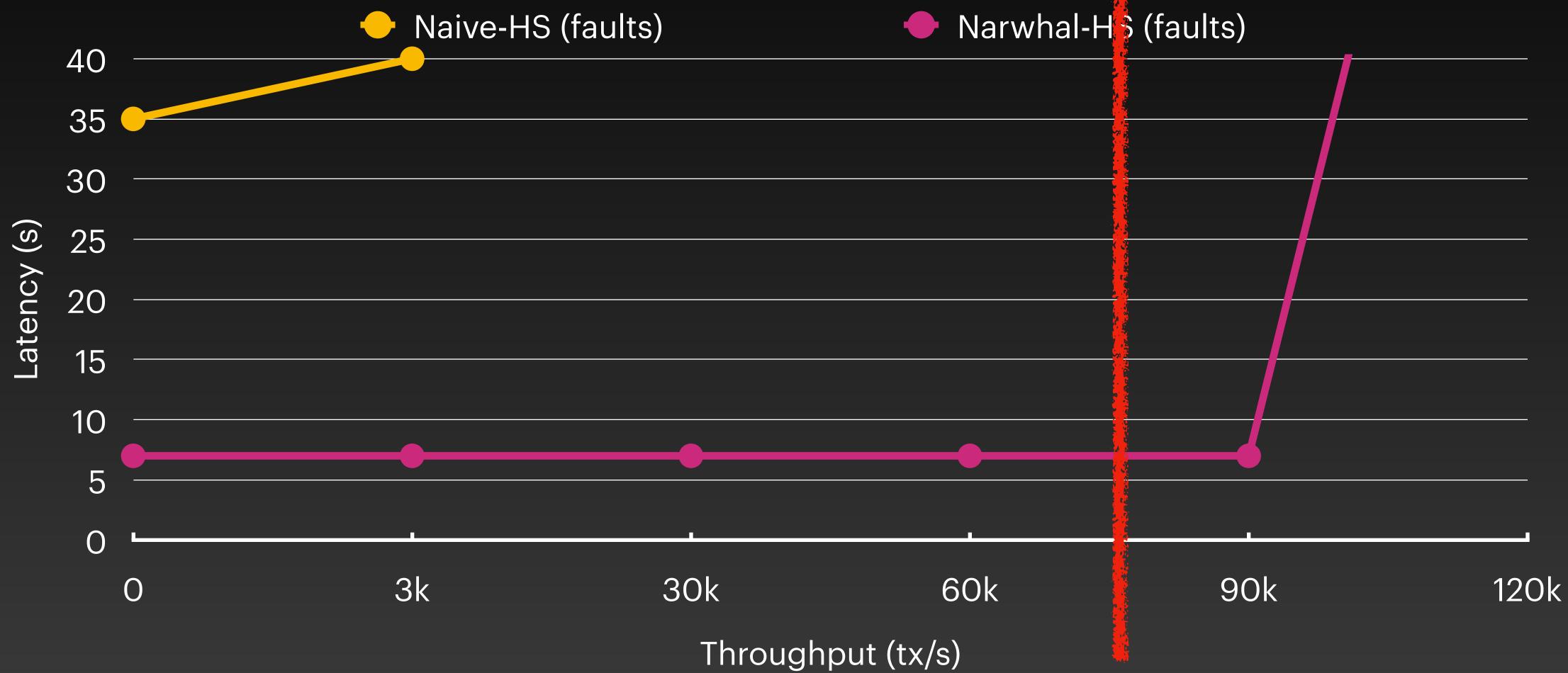




# Performance



# 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  $\rightarrow$  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 r 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. 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/10.1145/3492321.351959

Lefteris Kokoris-Kogias IST Austria

Alexander Spiegelman Aptos

## ACM Reference Format:

George Danezis, Lefteris Kokoris-Kogias, Alberto Sonnino, and Alexan der 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.

## 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  $\sim$  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

- Network model? 1.
- 2. BFT testing?

# Lessons Learnec

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



## DagRider

## All You Need is DAG

## Idit Keidar Oded Naon

## ABSTRACT

We present DAG-Rider, the first asynchronous Byzantine Atomic Broadcast protocol that achieves optimal resilience, optimal amortized communication complexity, and optimal time complexity. DAG-Rider is post-quantum safe and ensures that all values pro-posed by correct processes eventually get delivered. We construct DAG-Rider in two layers: In the first layer, processes reliably broad-east their proposals and build a structured Directed Acyclic Graph (DAG) of the communication among them. In the second layer, pro-cesses locally observe their DAGs and totally order all proposals with no extra communication.

Technion

ACM Reference Format: Idit Keidar, Eleftherios Kokoris-Kogias, Oded Naor, and Alexander Spiegel-man. 2021. All You Need is DAG. In Proceedings of the 2021 ACM Symposium on Detracibac of Directivent Communic (2007) 211. July 62: 0021 Visual man. 2021. All You Need is DAG. In Proceedings of the 2021 ACM Symposium 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/

years (4a, 52, 4a) to capture the needs or molectionalit systems. To ad-dress the fairness issues that naturally arise in interorganizational deployments, we focus on the classic long-lived Byzantine Atomic Foradcast (BAB) problem [12, 19], which in a datificiton to total order and progress also guarantees that *all* proposals by correct processes are eventually included. Up until recently, asynchronous protocols for the Byzantine con-

Up until recently, asynchronous protocols for the Byzantine con-sensus problem [12, 16, 26] have been considered too costly or complicated to be used in practical SMR solutions. However, two recent single-shot Byzantine consensus papers, VADA [1] and later Dumbo [35], presented asynchronous solutions with (1) optimal re-silience, (2) expected constant time complexity, and (3) optimal qua-dratic expresention and neutral examples. complexity (for the latter). In this paper, we follow this recent line Oded Naor is grateful to the Technion Hiroshi Fujiwara Cyber-Security Research enter for providing a research grant. Part of Oded's work was done while at Novi

classroom use is granted without lee 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 republish, to post on servers or to redistribute to lists, resources notice normaling the permission and/or a

Alexander Spiegelman Novi Research of work and present DAG-Rider: the first asynchronous BAB proto col with optimal resilience, optimal round complexity, and optimal amortized communication complexity. In addition, given a perfect shared coin abstraction, our protocol does not use signatures and does not rely on asymmetric cryptographic assumptions. Therefore, when using a deterministic threshold-based coin implementation

Eleftherios Kokoris-Kogias IST Austria and Novi Research

with an information theoretical agreement guarantee [13, 34], the safety properties of our BAB protocol are post-quantum secure Overview. We construct DAG-Rider in two layers: a communica-tion layer and a zero-overhead ordering layer. In the communication layer, processes reliably broadcast their proposals with some meta-data that help them form a Directed Acyclic Graph (DAG) of the messages they deliver. That is, the DAG consists of rounds s.t. every process broadcasts at most one message in every round and every man. 2021. All You Need is DAG. In *Proceedings of the 2021 ACM Symposium* on *Principles of Distributed Computing (PDC 21), July 24-30, 2021, Wight 24-30, 2021, Wig* 

simply a single reliable broadcast. The agreement property of the reliable broadcast ensures that all correct processes eventually see the same DAG. Moreover, the validity property of the reliable broad-cast guarantees that all broadcast messages by correct processes are cast guarantees that all broadcast messages by correct processes are eventually included in the DAG. As a result, in contrast to the VABA and Dumbo protocols that retroactively ignore half the protocol messages and commit one value out of O(n) proposals, DAG-Rider does not waste any of the messages and all proposed values by correct processes are eventually ordered (i.e., there is no need to re-propose).

Complexity. We measure time complexity as the asynchronous time [16] required to commit O(n) values proposed by different correct processes, and we measure communication complexity by the number of bits processes send to commit a single value. To compare DAG-Rider to the state-of-the-art asynchronous Byza tine agreement protocols, we consider SMR implementations that run an unbounded sequence of the VABA or Dumbo protocols to independently agree on every slot. To compare apples to apples in respect to our time complexity definition, we allow VABA and Dumbo based SMRs to run up to n slots concurrently. Note, how-Dumbo based SMRs to run up to *n* stots concurrently. Note, how-ever, that for execution processes must output the slot decisions in a sequential order (no gaps). Therefore, based on the proof in [6], the time complexity of VABA and Dumbo based SMRs is O(log(*n*)). Table I compares DAQ-Rider to VABA and Dumbo based SMRs. Since our protocol uses a reliable broadcast abstraction as a basic building block, different instantiations yield different complexity.

or example, if we use the classic Bracha broadcast [11] to propose

## Tusk

## 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 deign 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 despit failures. Narwhal is designed to easily scale-out using multi-ple 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) vields significantly better sus protocol (Narwhal-HotStuff) yields significantly better throughput ven 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-com-

As a summary of results, on a WAN, Narwhal-Hotstuff achieves over 130,000 t/xse at less than 2-sec latency com-pared with 1,800 t/xsec at 1-sec latency for Hotstuff. Ad-ditional 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 proto-cols maintain high throughput, but Narwhal-HotStuff suffers from increased latency.

CCS Concepts:  $\bullet$  Security and privacy  $\rightarrow$  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(i) must be honored. Abstracting with credit is permitted. To copy otherwise, or be honored. Abstracting with credit is permitted. Io copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fce. Request permissions from permissions@acm.org. EuroSyn 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

## IST Austria Alexander Spiegelman Aptos

Lefteris Kokoris-Kogias

ACM Reference Format: der 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.

## 1 Introduction

1 Introduction Byzantine consensus protocols [15, 19, 21] and the state machine replication paradigm [13] for building reliable dis-tributed systems have been studied for over 40 years. How-ever, with the rise in popularity of blockchains there has been a renewed interest in engineering high-performance con-sensus protocols. Specifically, to improve on Bitcoin's [33] throughput of only 4 txise carly works [29] suggested com-mittee 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. and are now becoming the norm in proof-of-stake designs Existing approaches to increasing the performance of dis-tributed ledgers focus on creating lower-cost consensus algo-rithms 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-synchic ) protocol that minimizes verall message number, but relies on a leader to pro duce proposals and coordinate consensus, fails to cap ture 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 order 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 distri

## **Data Dissemination**

- Hard to make efficient
- 99% of the code

## Bullshark

## **Dumbo-NG**

## **Bullshark: DAG BFT Protocols Made Practical**

## Alexander Spiegelman sasha@aptosia... Aptos

Alberto Sonnino alberto@mysternlabs.com Mysten Labs

ABSTRACT

We present BullShark, the first directed acyclic graph (DAG) based asynchronous Byzantine Atomic Broadcast protocol that is opti-mized for the common synchronous case. Like previous DAC-based BFT protocols [19, 25], BullShark requires no extra communication to achieve consensus on top of building the DAG. That is, parties can totally order the vertices of the DAG by interpreting their local view of the DAG edges. Unlike other asynchronous DAG-based protocols, BullShark provides a practical low latency fast-path that exploits synchronous periods and depreciates the need for notori-ously complex view-change mechanisms. BullShark achieves this while maintaining all the distribut properties of its predecessor DAG-Rider [25]. Namely, it has optimal amortized communication com-plexity, it provides fairness and asynchronous liveness, and asfety is guaranteed even under a quantum adversary. In order to show the practicality and simplicity of our approach, We present BullShark, the first directed acyclic graph (DAG) based

In order to show the practicality and simplicity of our approach, we also introduce a standalone partially synchronous version of BullSark which we evaluate against the state of the art. The im-plemented protocol is embarrassingly simple (200 LOC on top of an existing DAG-based mempool implementation [19]). It is highly efficient, achieving for example, 125,000 transaction per second with a 2 seconds latency for a deployment of 50 parties. In the same setting the state of the art pays a steep 50% latency increase as it optimizes for asynchrony. ACM Reference Format:

Alexander Spiegelman, Neil Giridharan, Alberto Sonnino, and Lefteris Kokoris-Kogias. 2022. Bullshark: DAG BFT Protocols Made Practical. In Proceedings of ACM Conference, Los Angeles, CA, USA, November 2022 (Con-ference '22), 17 pages. https://doi.org/10.1145/n

1 INTRODUCTION rdering transactions in a distributed Byzantine nechanism has become one of the most timely research areas in recent years due to the blooming Blockchain use-case.

to post on servers or to redistri fee. Request permissions from

A recent line of work [8, 19, 21, 25, 33, 40] proposed an elegan

Permission to make digital or hard copies of all or part of this work for personal or

classroom use is granteed without are portant or copies bear this notice and the full citation 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 republish to the strategies of the strategies of the strategies for specific permission and/or a

way to separate between the distribution of transactions and the

giridhn@berkeley.edu UC Berkeley Lefteris Kokoris-Kogias ekokoris@ist.ac.at IST Austria logic required to safely order them. The idea is simple. To propos transactions, parties send them in a way that forms a casual orde transactions, parties send them in a way that forms a ca among them. That is, messages contain blocks of trans well as references to previously received messages with the sly received messages, which togeth form a directed acyclic graph (DAG). Interestingly, the structure o he DAG encodes information that allow parties to totally orde the DAG by locally interpreting their view of it without sending any extra messages. That is, once we build the DAG, implementing consensus on top of it requires zero-overhead of communication. The pioneering work of Hashgraph [8] constructed an unstructured DAG, where each message refers to two previous ones, ar used hashes of messages as local coin flips to totally order the DAG in asynchronous settings. Aleph [21] later introduced a structured round-based DAG and encoded a shared randomness in each round via a threshold signature scheme to achieve constant latency in expectation. The state of the art is DAG-Rider [25], which is built on expectation. The state of the art is DAG previous ideas. Every round in its DAG for each party), each of which contain

well as references (edges) to at least 2f + 1 vertices in the previous round. Blocks are disseminated via reliable broadcast [11] to avoid equivocation and an honest party advances to the next round once it reliably delivers 2f + 1 vertices in the current 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 encodes the "voting" information that is sufficient to totally order all the DAG's vertices. So in this sense it is not different from order all the DAG & vertices. So in this sense it is not autrent from other BFT protocols in which parties send explicit vote messages, which contain no transactions as well. Remarkably, by using the DAG to abstract away the communication layer, the entire edges interpretation logic of DAG-Rider to totally order the DAG spans over less than 30 lines of pseudocode. DAG-Rider is an asynchronous Byzantine atomic broadcast (BAB),

Neil Giridharan

which achieves optimal amor per transaction), post quantum safety, and some notion of fairnes per transaction, joint quantum assisty and some notion to mances (called Validity) that guarantees that every transaction proposed by an honest party is eventually delivered (ordered). To achieve optimal amortized communication DAG-Rider combines batching techniques with an efficient asynchronous verifiable information dispersal protocol [14] for the reliable broadcast bub ling block. The protocol is post quantum safe because it does not rely on primitives that a quantum computer can brake for the safety properties. That is, a quantum adversary can prevent the protocol progress, but it cannot violate safety guarantees. However, although DAC-based protocols have a solid theoreti-cal foundation, they have multiple gaps before being realistically deployable in practise. First, they all optimize for the worst case

## Dumbo-NG: Fast Asynchronous BFT Consensus with

Throughput-Oblivious Latency							
	Yingzi Gao* ISCAS & UCAS yingzi2019@iscas.ac.cn	Yuan Lu ISCAS luyuan@isca		Zhenliang Lu* USYD zhlu9620@uni.sydney.edu.au			
arXiv:2209.00750v3 [cs.CR] 1 Feb 2024	yingzi2019@iscas.ac.cn Qiang Tang USYD qiang.tang@yolmey.edu.au ABSTEACT Despite recent progresses of practical asynchr fault tolerant (BFT) consensus, the state-of-th suffer from suboptimal performance. Particular muthroughput, most existing protocols with amortized communication complexity require node to broadcast a huge batch of transactions, w sacrifices latency. Worze still, the f slowest nodes never be agreed to output and thus can be cen the number of faults. Implementable miligatio ther uses computationally costly threshold ene communication blow-up by letting the honest n redundant transactions, thus causing further eff We present Dumbo-NC, a novel asynchronor (atomic broadcast) to solve the remaining practica (atomic broadcast) to solve the researces the WBA of est nodes with 1/2 probability). Most interestingly structure empowers completely concurrent execting	luyuan@isca Jing Xu ISCAS xujing@iscas onous Byzantine- e-art designs still y to obtain maxi- guaranteed linear ach participanti hich dramatically 'broadcasts might i 'broadcasts might i 'broadcasts might i 'broadcasts one of the off the threat ei- n to the threat ei- til subscience of the threat greement (MVBA) the new protocol tion of transaction	s.ac.cn 	zhlu9620@uni.sydney.edu.au Zhenfeng Zhang' ISCAS zhenfeng@iscas.ac.n msensus, Byzantine-fault tolerance, blockchain sensus, Byzantine-fault tolerance, blockchain UCTION of Bitcoin [63] and blockchain [19, 24] leads to an ny to lay down the infrastructure of distributed ver thical gobal infrastructure maintained by a set of ful and geologically distributed nodes [11], and sensus protocols that are both secure and efficient wer the Internet. BFT for indispensable robustness. The con- ralized infrastructure has to thrive in a highly soment. In particular, when the applications atop cial and banking services, some nodes can be well ad and launch malicious attacks. Even worse, the mjäth become part of the attack surface due to net- misconfigurations and even network attacks. To verstaid delpolyment environment. <i>a, synchronous</i>			
arXiv:2209.00	dissemination and asynchronous agreement. This brings about two benefits: (i) the throughput-latency tension is resolved to approach peak throughput with minimal increase in latency; (ii) the trans- actions broadcasted by any honest node can be agreed to output, thus conquering the censorship threat with no extra cost. We implement Dumbo-NG with using the current fastest GLL+22 WVBA with quality (NDSS'22) and compare it to the state-of-the-art asynchronous BFT with guaranteed eensorship resilience including Dumbo (CCS'20) and Speeding-Dumbo (NDSS'22). Along the way, we apply the techniques from Speeding-Dumbo to Dispersedled.edger		Byzantine-fault tolerant (BFT) consensuses [4, 20, 53, 47, 58, 60] are arguably the most suitable candidates. They can realize high security-assurance to ensure liveness (as well as safety) despite an asynchronous adversary that can arbitrarily delay messages. In contrast, many (partial) synchronous consensus protocols [5, 6, 8, 15, 27, 44, 45, 64, 73] such as PBFT [26] and HofStuff [75] might sustain the inherent loss of liveness (i.e., generate unbounded com- munications without making any progress) [36, 60] when unluckly encountering an asynchronous network adversary. 1.1 Practical obstacles of adopting				

SDumbo-DL for comprehensive comparison. Extensive experiments (over up to 64 AWS EC2 nodes across 16 AWS regions) reveal: Dumbo-NG realizes a peak throughput 4-8x over Dumbo, 2-4x over Dumbo-NG realizes a peak throughput 4-8x over Dumbo, 2-4x over Speeding-Dumbo, and 2-3x over sDumbo-DL (for varying scales); More importantly, Dumbo-NG's latency, which is lowest among all tested protocols, can almost remain stable when throughput grows CCS CONCEPTS

• Security and privacy  $\rightarrow$  Systems security; Distributed systems security; • Computer systems organization  $\rightarrow$  Reliability.

## 1.1 Practical obstacles of adopting

asynchronous BFT consensus Unfortunately, it is fundamentally challenging to realize practical asynchronous BFT consensus, and none of such protocols was widely adopted due to serious efficiency concerns. The seminal FLP "impossibility" [36] proves that no determinist exists in the asynchronous netwo [1, 12, 13, 21, 25, 65, 67] aimed at to ous network. Since the 1980s, many attempts enting the "imposs (1) in 3.5 response of Lance we consider the second sec Authors are listed alphabetically. Yingzi, Yuan & Zhenliang made equal contributions. An abridged version of the paper will appear in ACM CCS 2022.

## Consensus

- Error prone
- Isolated, easy to maintain



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

<section-header><section-header><section-header><section-header><section-header><section-header><section-header><section-header><text><text><text><text><text><text><text><text><text>

## Bullshark

## **Bullshark: DAG BFT Protocols Made Practical**

## Alexander Spiegelman sasha@aptoslabs.com Aptos

Alberto Sonnino alberto@mysternlabs.com Mysten Labs

<section-header><section-header><section-header><section-header><section-header><section-header><section-header><section-header><section-header><section-header><section-header><section-header><section-header><section-header>

Neil Giridharan

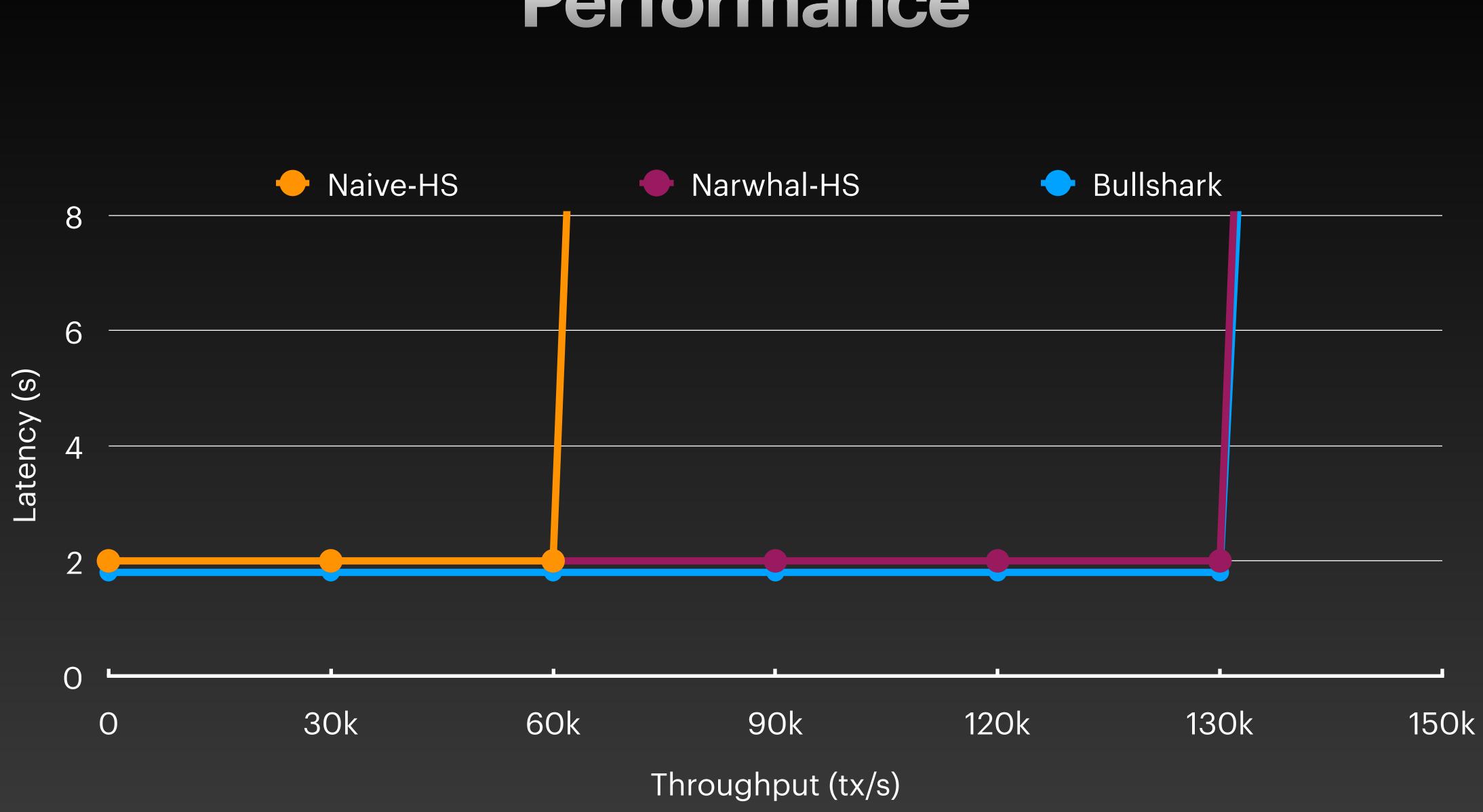
giridhn@berkeley.edu UC Berkeley

Lefteris Kokoris-Kogias

ekokoris@ist.ac.at IST Austria

## Dumbo-NG: Fast Asynchronous BFT Consensus with

	0 1			
Yingzi Gao* ISCAS & UCAS vingzi2019@iscas.ac.cn		Yuan Lu* ISCAS luvuan@iscas.ac.cn		Zhenliang Lu* USYD zhlu9620@uni.sydney.edu.au
	Qiang Tang*	Jing J		
	USYD	ISCA		ISCAS
	qiang.tang@sydney.edu.au	xujing@isc	cas.ac.cn	zhenfeng@iscas.ac.cn
5	ABSTRACT		KEYWORDS	
arXiv:2209.00750v3 [cs.CR] 1 Feb 2024	Despite recent progresses of practical asynchronous 1 fault tolerant (BFT) consensus, the state-of-the-art de saffer from suboptimal performance. Particularly, to ob mun throughput, most existing protocols with guarant motical communication complexity require each par node to broadcast a huge batch of transactions, which dr particles latency. Worse still, the <i>f</i> slowest nodes' broadc never be agreed to output and thus can be censored (i the number of faults). Implementable mitgation to the there uses computationally costly threshold encryption communication low-up by letting the hones in the fault of strong strong strong strong strong strong strong strong rouge to a strong strong strong strong strong strong strong (strong strong strong strong strong strong strong strong strong strong rouge strong strong strong strong strong strong strong strong strong rouge strong strong strong strong strong strong strong strong strong rouge strong strong strong strong strong strong strong strong rouge strong strong strong strong strong strong strong strong strong strong rouge strong strong strong strong strong strong strong strong strong rouge strong strong strong strong strong strong strong strong strong rouge strong strong strong strong strong strong strong strong strong rouge strong str	signe still tain maxi- ceed linear ticipating manically unmitcally stars might threat ei- or incurs sources and the star broadcast ssues. consensus its techni- us atomic t (AVBA) from hon- p protocol ansaction about two approach the trans- to output, t. st GLL-22 of the-art including g the way, wedLedger agrecalled periments .2-4x over among all put grows, d systems dity.	1 INTRODUCTION The huge success of Bitch foregars for insignment of ledgers for mission-eritical is envisioned as critical generation of deployment over the Asynchronous BET for sums of decentralized diversarial environment is are critical financial and motivated to collade and motivated to far the superstructure of the superstructure of the superstructure of the motivated to the superstructure of the superstructure of the superstructure of the superstructu	in [63] and blockchain [19, 24] leads to y down the infrastructure of distribu- ingolications. Such decentralized busi- lobal infrastructure maintanced by as geologically distributed nodes [11], rotocols that are both secure and effic Internet. <b>vr indispensable robustness</b> . The <i>i</i> - infrastructure has to thrive in a high lanch malicous attacks. Even worse, ecome part of the attack surface due to lanch malicous attacks. Even worse, accome part of the attack surface due to hank malicous attacks. Even worse, accome part of the attack surface due to hank malicous attacks. Even worse, accome part of the attack surface due to (BT) consensussa [14, 20, 35, 47, 58, hatble candidates. They can realize 1 sure liveness (as well as safety) des york-nonus consensus protocols [5], and PIN [20] and HolStuff [75] m of liveness (as, generate unbounde d ing any progress) [36, 60] when unluc ronous network adversary. <b>tacles of adopting</b> <b>s BFT consensus</b> mentially challenging to realize pract- nerious efficiency concerns. The sem proves that to <i>deterministic</i> conser- s network. Since the 1980s, many atter- ations of michenystopic substaces and the interposition proves that to <i>deterministic</i> conser- s network. Since the 1980s, many atter- gives emption and the interposition and and the interposition and the interposition and the interposition and and and the interposition and and and and a



# Performance

### Research Questions

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

### Lessons Learnec

- 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...





Reply -

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...





### Aptos

### Linera

 $\bullet \bullet \bullet$ 

### **Over a year for mainnet**

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



## Research Questions

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

### Lessons Learnec

- 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



- 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

### Rati Gelashvili

Abstract

The Narwhal system is a state-of-the-art Byzantine faultnt scalable architecture that involves constructing a ed acyclic graph (DAG) of messages among a set of val-s in a Blockchain network. Bullshark is a zero-overhead

idators in a Blockchain network. Bullshark is a zero-overhead consensus protocol on top of the Narwhal's DAG that can order over 100k transactions per second. Unfortunately, the high throughput of Bullshark comes with a latency price due to the DAG construction, increasing the latency compared to the state-of-the-art leader-based BFT consensus protocols. We introduce Shoal, a protocol-agnostic framework for en-hancing Narwhal-based consensus. By incorporating leader reputation and pipelining support for the first time, Shoal significantly reduces latency. Moreover, the combination of properties of the DAG construction and the leader reputa-tion mechanism enables the elimination of timeouts in all but extremely uncommon scenarios in practice, a property out extremely uncommon scenarios in practice, a property we name "prevalent responsiveness" (it strictly subsumes the established and often desired "optimistic responsiveness" property for BFT protocols). We integrated 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 executions, and up-to 80% reduction in executions with failures against the vanilla Bullshark implementation. CCS Concepts: - Security and privacy  $\rightarrow$  Distributed systems security. existing Narwhal-based consensus protocol, in an open-source

systems security Keywords: Consensus Protocol, Byzantine Fault Tolerance i

ACM Reference Format: 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 Byzantine fault tolerant (BFT) systems, including consensus protocols [13, 23, 24, 29] and state machine replication [7, 10, 26, 42, 46], have been a topic of research for over four decades as a means of constructing reliable distributed sys-tems. Recently, the advent of Blockchains 18 as underscored the significance of high performance. While Bitcoin handles approximately 10 transactions per second (TF8), the proof-of-stake committee-based blockchains [38–41, 43, 44] are now engaged in a race to deliver a scalable BFT system with the utmost throughput and minimal latency.

Historically, the prevailing belief has been that reducing communication complexity was the key to unlocking high performance, leading to the pursuit of protocols with li ear communication. However, this did not result in drastic enough improvements in the throughput, falling significantly short of the current blockchain network targets. For example the state-of-the-art Hotstuff [46] protocol in this line of work only achieves a throughput of 3500 TPS [3]. A recent breakthrough, however, stemmed from the real-

Balaji Arun <sub>Aptos</sub>

Zekun Li

ization that data dissemination is the primary bottleneck for leader-based protocols, and it can benefit from parallelization [4, 17, 37, 45]. The Narwhal system [17] separated data dissemination from the core consensus logic and propose an architecture where all validators simultaneously dissem nate data, while the consensus component orders a smaller amount of metadata. A notable advantage of this archite ture is that not only it delivers impressive throughput of a single machine, but also naturally supports scaling out each blockchain validator by adding more machines. The Narwhal paper [17] evaluated the system in a geo-replicated environment with 50 validators and reported a throughpu of 160,000 TPS with one machine per validator, which furthe

Developing a production ready reliable distributed system is challenging, and integrating intricate consensus protocols only adds to the difficulty. Narwhal addresses this issue by abstracting away networking from the consensus protocol. It constructs a non-equivocating round-based directed acyclic graph (DAG), a concept initially introduced by Aleph [21]. In this design, each validator contributes one vertex per round, and each vertex links to n - f vertices in the preceding round. Each vertex is disseminated via an efficient reliable broadcast implementation, ensuring that malicious validators cannot . ready reliable distributed system implementation, ensuring that malicious validators cannot distribute different vertices to different validators within the same round. With networking abstraction separated from the details of consensus, the DAG can be constructed without contending with complex mechanisms like view-change or

view-synchronization. During periods of network asynchrony, each validator may observe a slightly different portion of the DAG at any

### Sailfish

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

n.shrestha@supraoracles.com rohan	aniket	iket Kate Kartik Nayak purdue.com kartik@cs.duke.edu sity / Supra Research Duke University
-----------------------------------	--------	--

Abstract—Directed Acyclic Graph (DAG) based BF1 protocols balance consensus efforts across different parties and maintain high throughput even when some designated parties fail. How-ever, existing DAG-based BFT protocols exhibit long latency to mmit decisions, primarily because they have a leader every 2 or more "rounds". Recent works, such as Shoal (FC'23) and and their network are not used, leading to uneven resource Mysticeti, have deemed supporting a leader vertex in each round particularly difficult, if not impossible. Consequently even under honest leaders, these protocols require high latency (or communication complexity) to commit the proposal ubmitted by the leader (leader vertex) and additional latency

submitted by the leader (leader vertex) and additional latency to commit other proposals (non-leader vertices). In this work, we present Salifish, the first DAG-based BFT that supports a leader vertex in each round. Under honest leaders, Salifish maintains a commit latency of one reliable broadcast (RBC) round plus 16 to commit the leader vertex (where  $\delta$  is the actual transmission latency of a message) and only an additional RBC round to commit non-leader vertices. We also extend Salifish to Multi-leader Salifish, which facili-tates multiple leaders within a single round and commits all leader vertices in a round with a latency of one RBC round plus 1 $\delta$ . Our experimental evaluation demonstrates that our proto-cols introduce significantly lower latency overhead compared cols introduce significantly lower latency overhead comparto existing DAG-based protocols, with similar throughput.

### 1. Introduction

Byzantine fault-tolerant state machine replication (BFT SMR) protocols form the core underpinning for blockatas. 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 up to f of these parties is Byzantine (arbitrarily malicious). Owing to the need for efficient blockchains in practice, there has been a lot of recent progress in improving the key ficinety metrics namely, latency, communication complexity, and throughput under various network conditions. Assume the trader vertex and order other non-leader vertices are disgnated and how fast the leader vertices are committed directly influences the commit latency. **Supporting a leader vertex in each round.** State-of-the-art protocols designate leaders once every two or more rounds, and in fact, deem supporting a leader vertex in each round particularly difficult. In their works, Shoal [45] who is the narve secondition.

htract-Directed Acyclic Graph (DAG) based BFT protocols agree on the proposed values and ensure that the leade keeps making progress. From an efficiency standpoint, this approach results in two key drawbacks. First, there is a uneven scheduling of work among the parties. While the leader is sending a proposal, the other parties' processors and their network at the totake, it cannot be directively resoluted usage across partices. Second, in typical leader-based pro-tocols progress stops if the leader fails and until it is replaced. Several techniques proposed in the literature can potentially mitigate these concerns. These include the use of erasure coding techniques [2], [41] or the data availability committees [26], [27], [49] to disseminate the data more efficiently. efficientl

Recently, a novel approach known as DAG-based BFT has emerged [5], [18], [28], [33], [34], [46], [47]. These protocols enable all participating parties to propose in paralprotocols enable any participating parties to propose in paral-lel, maximizing bandwidth utilization and ensuring equitable distribution of workload. Additionally, because each party is responsible for disseminating its own transactions, the protocol continues to progress in constructing the DAG even if a party fails during a round. Consequently, these protocols have demonstrated improved throughput compared to their leader-based counterparts under moderate network sizes [19], [46]. However, existing DAG-based protocols incur a high latency compared to their "leader-heavy" coun-terparts [12], [22], [30], [37], [51]. Is high latency inherent for such DAG-based protocols? Addressing this question is

timistic conditions (such as an honest leader). Most of these protocol designs rely on a designated the inner workings of the protocol and exploring complex leader who is the party responsible for proposing transac-tions and driving the protocol forward while other parties Intuitively, this is because ... ". Similarly, Mysticeti 4

### **Techniques**

- Many leaders per round
- Leaders every round
- Uncertified DAG

### CM

### Cordial Miners: Fast and Efficient Consensus for

### **Every Eventuality**

Idit Keidar

Oded Naor

- Technion and StarkWar
- Ouri Poupko

Ehud Shapiro Weizmann Institute of Science

### — Abstract

Cordial Miners are a family of efficient Byzantine Atomic Broadcast protocols, with instance for a synchrony and eventual synchrony. They improve the latency of state-of-the-art DAG-based protocols by almost  $2\times$  and achieve optimal good-case complexity of O(n) by forgoing Reliable Broadcast as a building block. Rather, Cordial Miners use the blocklace--a partially-ordered counterpart of the totally-ordered blockchain data structure-to implement the three algorithmic

2012 ACM Subject Classification Computing methodologies  $\rightarrow$  Distributed algorithm

Keywords and phrases Byzantine Fault Tolerance, State Machine Replication, DAG, Consensus,

Related Version Cordial Miners: Fast and Efficient Consensus for Every Eventuality Full Version: https://arxiv.org/abs/2205.09174

components of consensus: Dissemination, equivocation-exclusion, and ordering.

Acknowledgements Oded Naor is grateful to the Azrieli Foundation for the award of an Azrieli Fellowship, and to the Technion Hiroshi Fujiwara Cyber-Security Research Center for providir renovany, and so the retinnois representation of the second secon

### 1 Introduction

The problem of ordering transactions in a permissioned Byzantine distributed system, also known as Byzantine Atomic Broadcast (BAB), has been investigated for four decades [30], and in the last decade, has attracted renewed attention due to the emergence of cryptocur Recently, a line of works [4, 14, 20, 33, 21, 27] suggests ordering transactions using a distributed Directed Acyclic Graph (DAG) structure, in which each vertex contains a block of transactions as well as references to previously sent vertices. The DAG is distributively or transactions as we as the critics to providely can relate in the Drive distinguirty constructed from messages of miners running the consensus protocol. While building the DAG structure, each miner also totally orders the vertices in its DAG locally. That is, as the DAG is being constructed, a consensus on its ordering emerges without additional nication among the miners.

The two state-of-the-art protocols in this context are DAG-Rider [21] and Bullshark [33]. DAG-Rider works in the asynchronous setting, in which the adversary controls the finite delay on message delivery between miners, and Bullshark works in the Eventual Synchrony (ES) model, in which eventually all messages between correct miners are delivered within a know

### Mysticeti

### MYSTICETI: Reaching the Latency Limits with Uncertified DAGs

### Kushal Babel\*<sup>†</sup>, Andrey Chursin<sup>‡</sup>, George Danezis<sup>‡§</sup>, Anastasios Kichidis<sup>‡</sup>, Lefteris Kokoris-Kogias<sup>‡¶</sup>, Arun Koshy<sup>‡</sup>, Alberto Sonnino<sup>‡§</sup>, Mingwei Tian<sup>‡</sup>

### \*Cornell Tech, $^{\dagger}IC3,\,^{\ddagger}Mysten$ Labs, $^{\$}University$ College London (UCL), $^{\P}IST$ Austria

Abstract—We introduce MYSTICETI-C, the first DAG-based Byzantine consensus protocol to achieve the lower bounds of latency of 3 message rounds. Since MYSTICETI-C is built over DAGs it also achieves high resource efficiency and censorship resistance. MYSTICETI-C achieves this latency improvement by avoiding explicit certification of the DAG blocks and by proposing a novel commit rule such that every block can be committed without delays, resulting in optimal latency in the steady state and under crash failures. We further extend MYSTICETI-C to MYSTICETI-FPC, which incorporates a fast commit path that achieves even lower latency for transferring assets. Unlike prior fast commit path protocols, MYSTICETI-FPC minimizes the number of signatures and messages by weaving the fast path transactions into the DAG. This frees up resources, which subsequently result in better performance. We prove the safety I and liveness in a Byzantine context. We evaluate both MYSTICETI furprotocols and compare them with state-of-the-art consensus and fast path protocols to demonstrate their low latency and resource efficiency, as well as helf more graceful degradation under creash failures. MYSTICETI-C is the first Byzantine consensus protocol to achieve WAN latency of 0.55 for consensus commit while simultaneously maintaining state-of-the-art throughput of over 2006 TBS. Finally, we report on integrating MYSTICETI-C as the consensus protocol into the Sui blockchain [67], resulting in over 4x latency reduction. Abstract-We introduce MYSTICETI-C, the first DAG-based

reasons: (1) the certification process requires multiple round-trips to broadcast each block between validators, get signa-ures, and re-broadcast each block between validators, get signa-than traditional consensus protocols [31], [64], [15]; (2) blocks normit on a "per-wave" basis, which means that only one operation, achieving the best of both worlds. MYSTICETI-C is



Fig. 1: P50 latency of a major blockchain switching from Bullshark (1900ms to MYSTICETI-C (390ms) consensus on 106 independently run validators

and verification consume a large amount of CPU on each validator, which grows with the number of validators [42], [16]. This burden is particularly heavy for a crash-recovered validator that typically needs to verify thousands of signatures when trying to catch up with the rest. Although at a first glance, certification seems to have the benefit that in adversarial case nodes can advance the DAG without needing to synchr I. INTRODUCTION Several recent blockchains, such as Sui [67], [12], have looted consensus protocols based on certified directed acvelue to the full-history, production experience of deploying Bullshark shows that this benefit is negated when needing to execute the committed transactions. As a result, the certification benefits only Byzantine Atomic Broadacst protocols but not if used for

1. INTRODUCTION Several recent blockchains, such as Sui [67], [12], hai adopted consensus protocols based on certified directed acyclic graphs (DAG) of blocks [25], [55], [56], [34], [30], [70], [52], [8], [44]. By design, these consensus protocols scale well in terms of throughput, with a performance of 100k tx's anyther against faults and network approximate are robust against faults and network approximate are robust against faults and network approximate are robust against faults and network in terms of uncertified DAGs Certified "- 4-livered through"

tures, and re-broadcast certificates. This leads to higher latency than traditional consensus protocols [31], (64), (15]; (2) bore commit on a "per-wave" basis, which means that only once every two rounds (for Bullshark [55]) there is a chance to finish increasing the latency of transactions proposed by the block. This phenomenon is similar to committing big batters of 2f + 1 blocks. Finally, (3) since all certified blocks need to

### Discussion

### **Certified DAG**

### Shoal/shoal++

### Sailfish/BBCA

- Low latency
- Easier synchroniser
- Leverage existing code • Flexible

### **Uncertified DAG**

- Lower latency
- Easy synchroniser

### **CM/Mysticeti**

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



## Research Questions

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

### Lessons Learnec

- 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





baimsn:	Iowards	Improving	g the Latency	of DAG-based	DF I

Nibesh Shrestha	Rohan Shrothrium	Aniket Kate	Kartik Nayak
n.shrestha@supraoracles.com	rohan@kurulabs.xyz	aniket@purdue.com	kartik@cs.duke.edu
Supra Research	Kuru Labs	Purdue University / Supra Research	Duke University
Abstract—Directed Acyclic Graph (D. balance consensus efforts across diffe high throughput even when some des	rent parties and maintain	keeps making progress. From an e	efficiency standpoint, t

<text><section-header><text><text><text><text>

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

Idit Keidar

- Ouri Poupko

### - Abstract

### 1 Introduction

### Mysticeti

### MYSTICETI: Reaching the Latency Limits with Uncertified DAGs

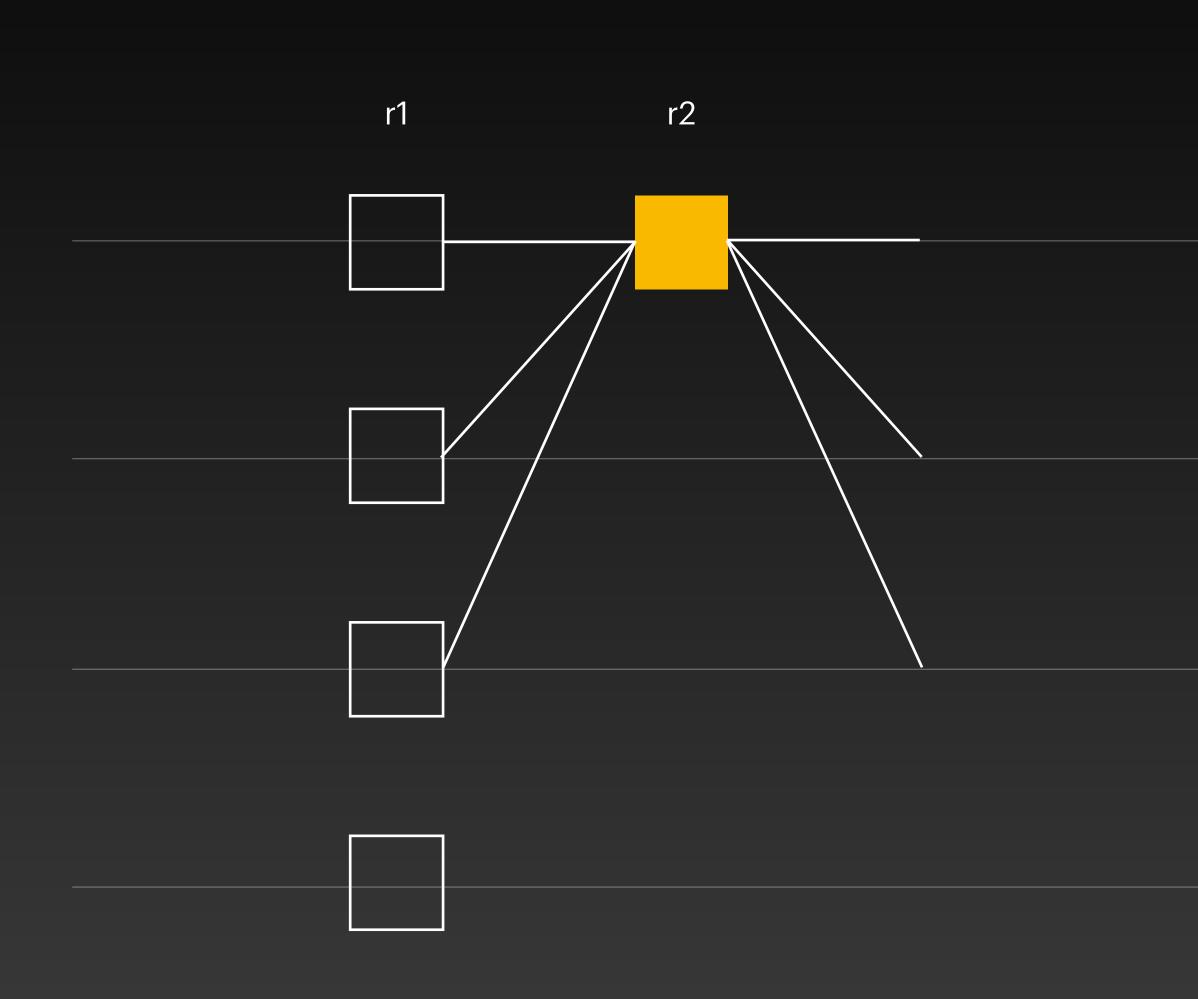
### Kushal Babel\*<sup>†</sup>, Andrey Chursin<sup>‡</sup>, George Danezis<sup>‡§</sup>, Anastasios Kichidis<sup>‡</sup>, Lefteris Kokoris-Kogias<sup>‡¶</sup>, Arun Koshy<sup>‡</sup>, Alberto Sonnino<sup>‡§</sup>, Mingwei Tian<sup>‡</sup>

### \*Cornell Tech, $^{\dagger}IC3,~^{\ddagger}Mysten$ Labs, $^{\$}University$ College London (UCL), $^{\P}IST$ Austria

<text><text><section-header><section-header><text><text><text><text><text>

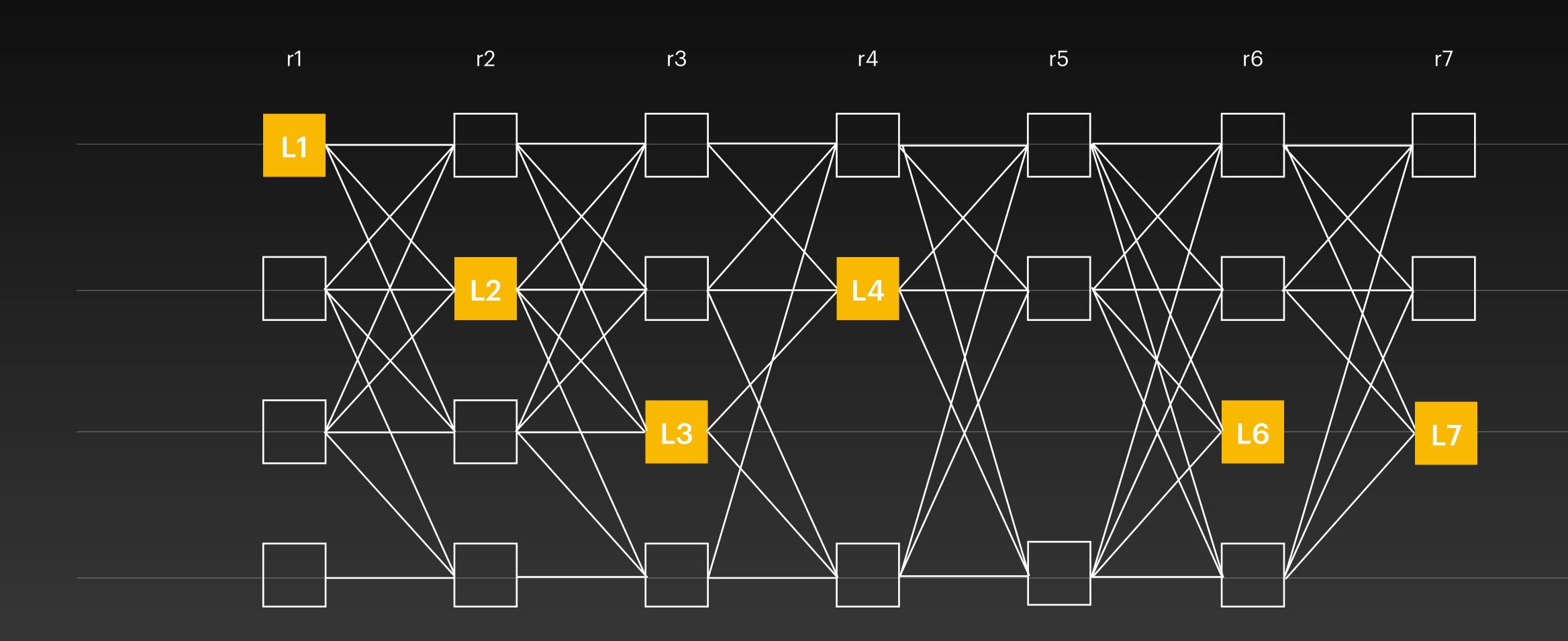


## **Uncertified DAG**

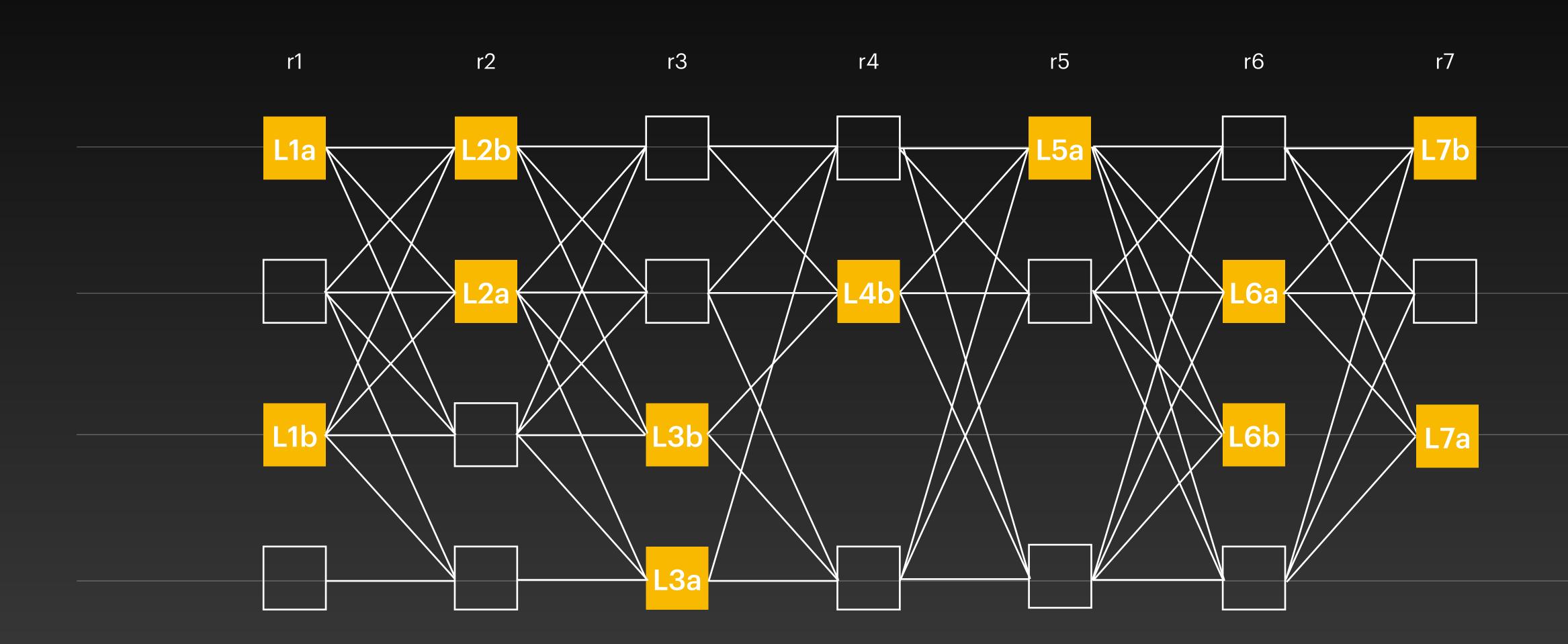


- Round number
- Author
- Payload (transactions)
- Signature

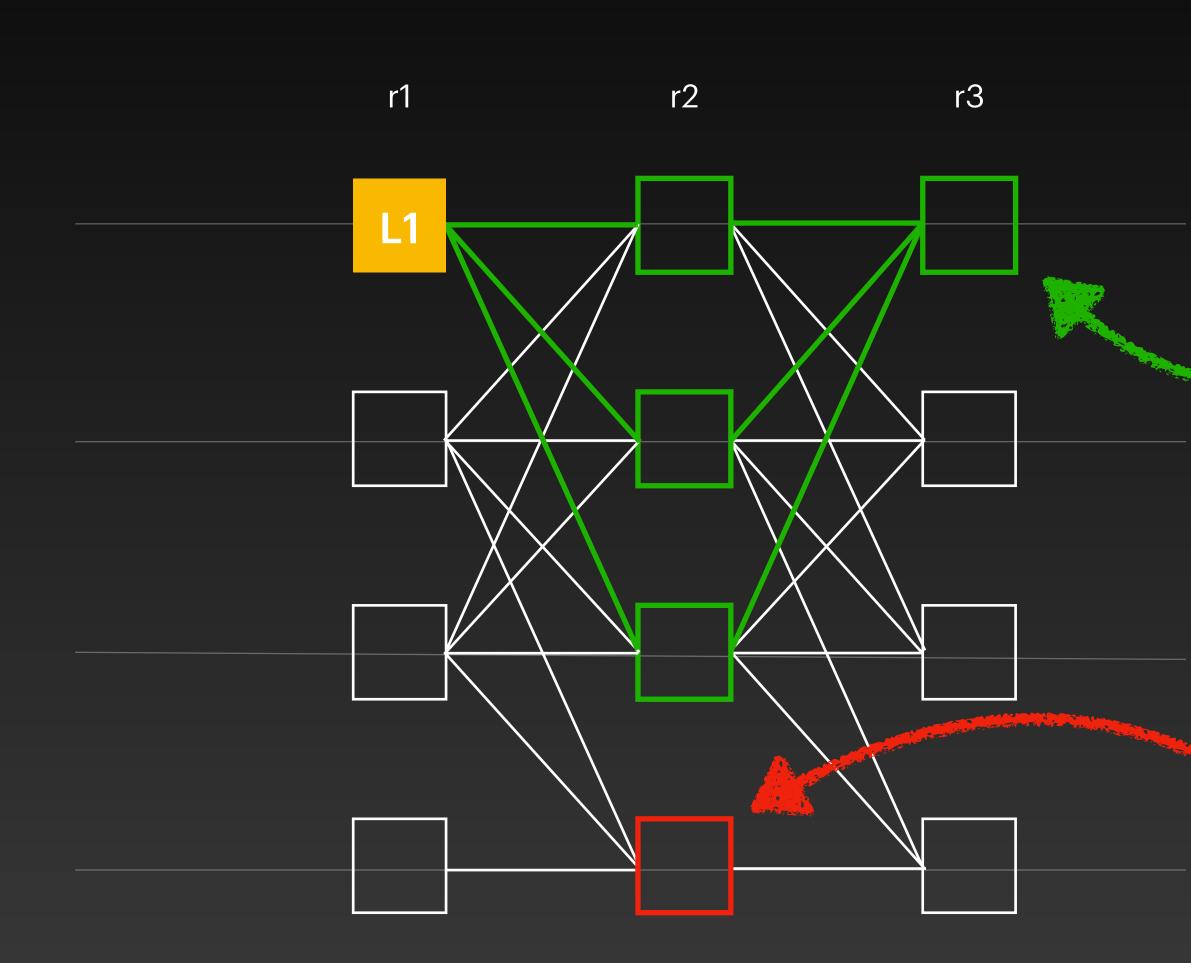
## **Uncertified DAG**



## **Uncertified DAG**

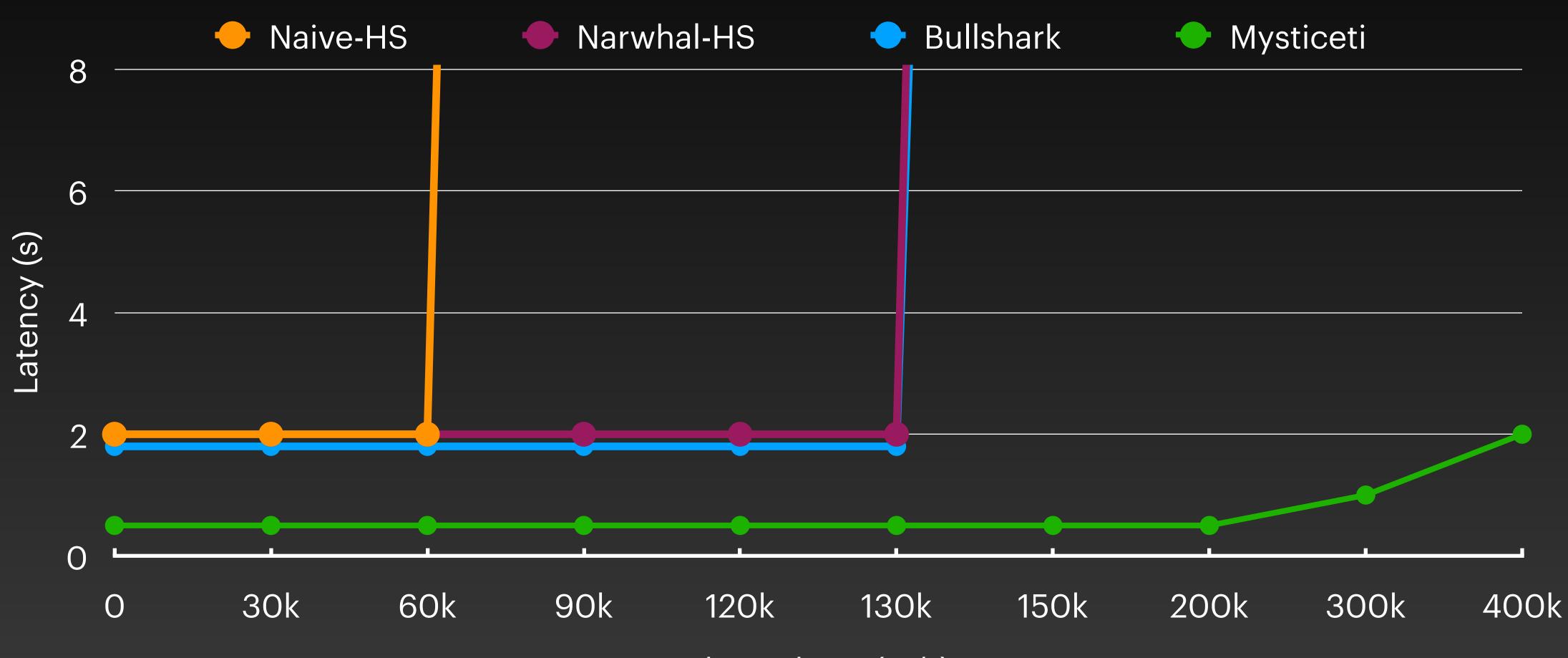


## Interpreting DAG Patterns



### Certificate





Throughput (tx/s)

### Performance

## Research Questions

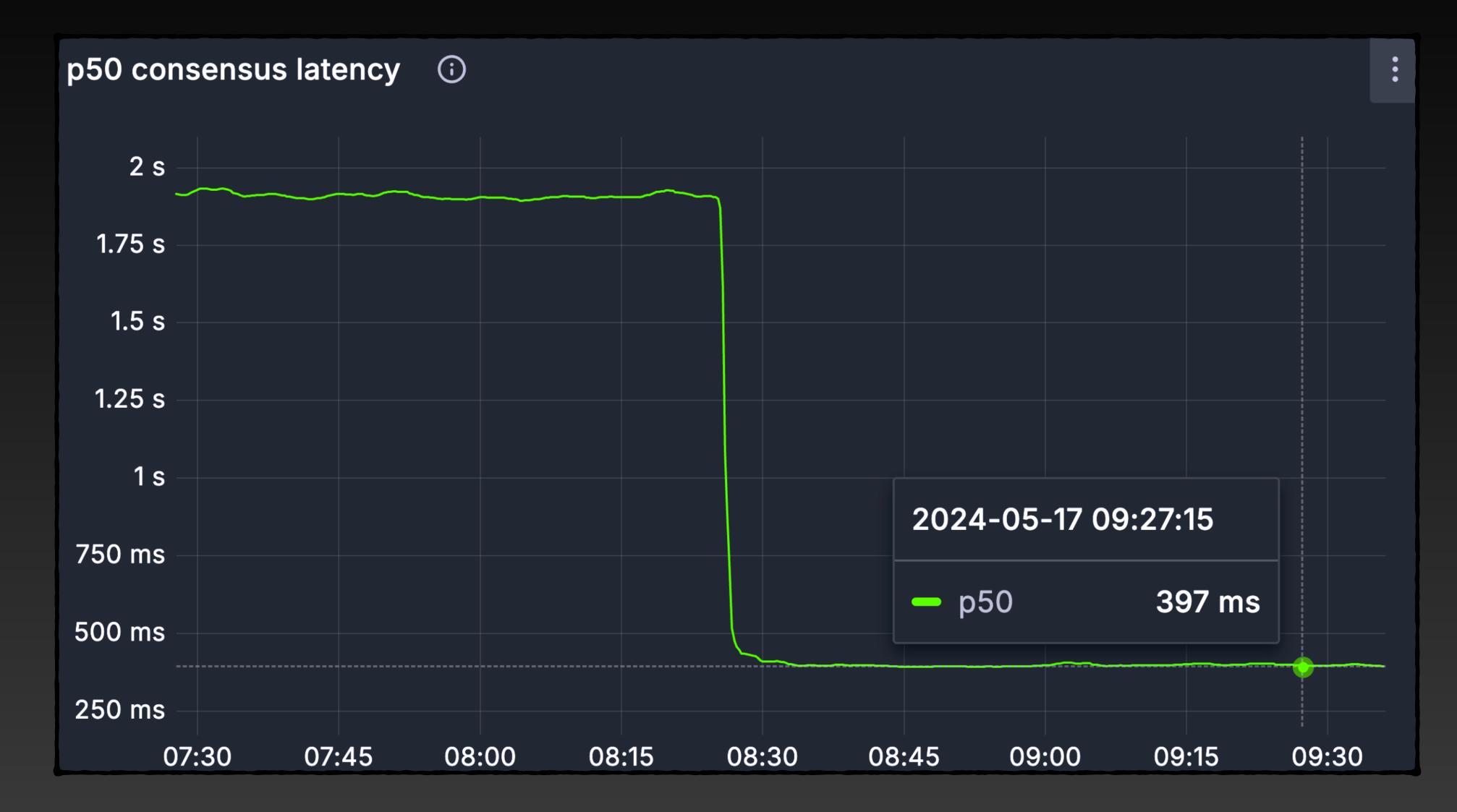
- Network model? 1.
- 2. BFT testing?
- 3. Consensus-exec interface?
- 4. Storage architecture?
- 5. Block synchroniser?
- 6. Realistic benchmarks?
- 7. Efficient reads?

### Lessons Learnec

- 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

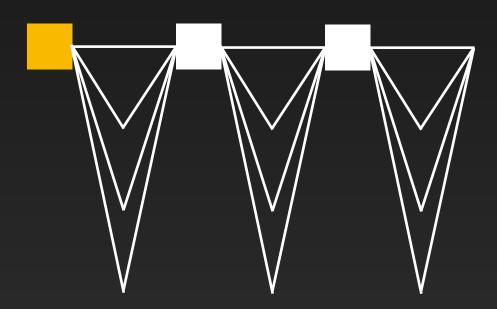


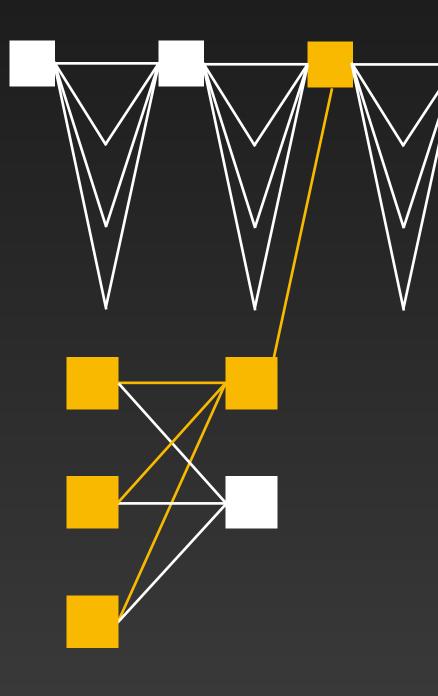
## The Sui Mainnet



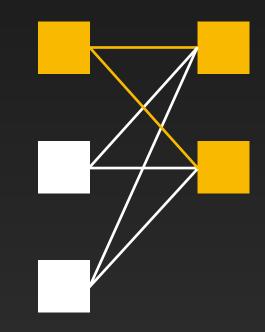
## The Roadmap

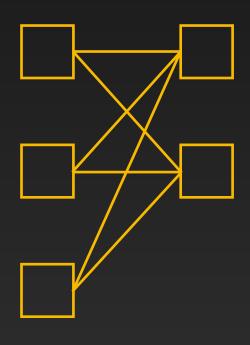
# 2019 2020-2021 naive consensus mempool @consensus











# **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 🚧



## **Projects Roadmap**



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



2 replies





**Thread** *<sub># sui-core-internal*</sub>



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 🚧



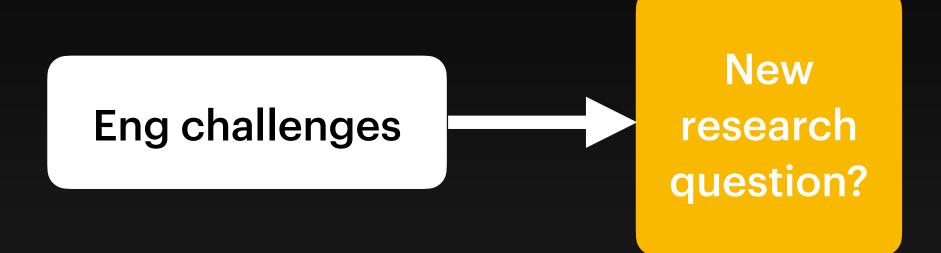


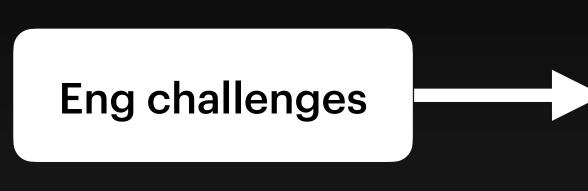
John Martin Oct 18th at 6:16 AM Can I get an invite to this 🙏



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

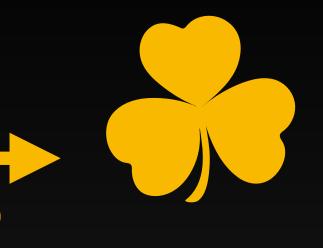
<del>TÝ</del> 1 **(;;**)



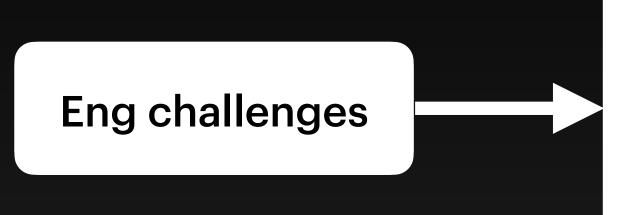


New research question?

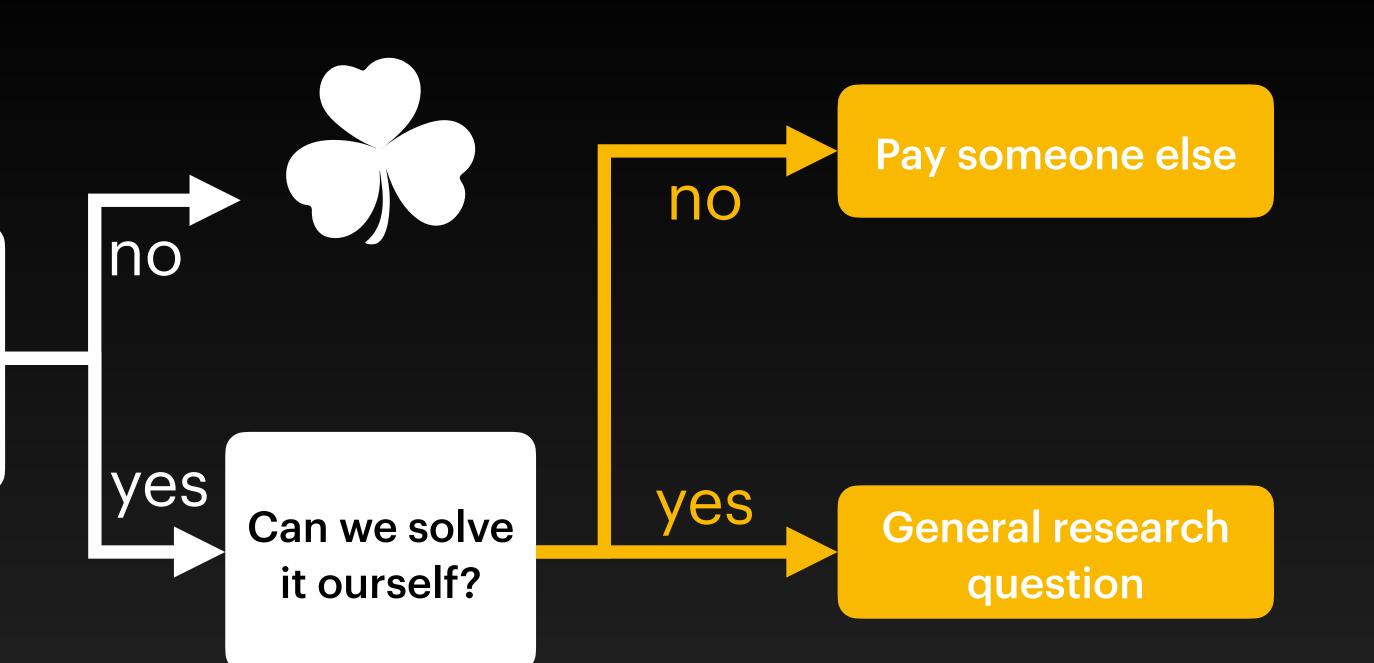




Can we solve it ourself?



New research question?







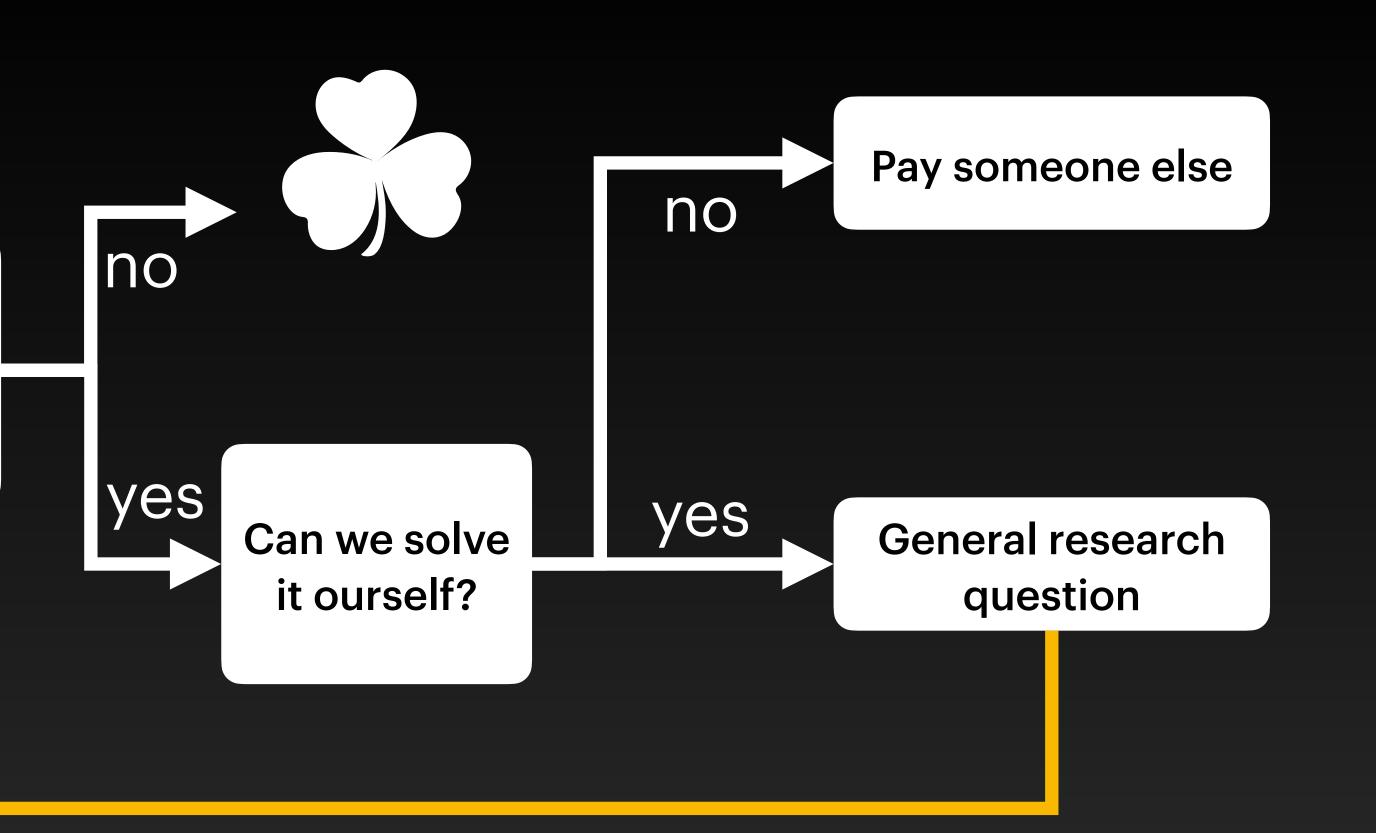


## **Research Gifts**

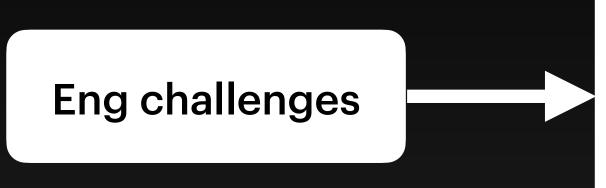
### (please keep it short)



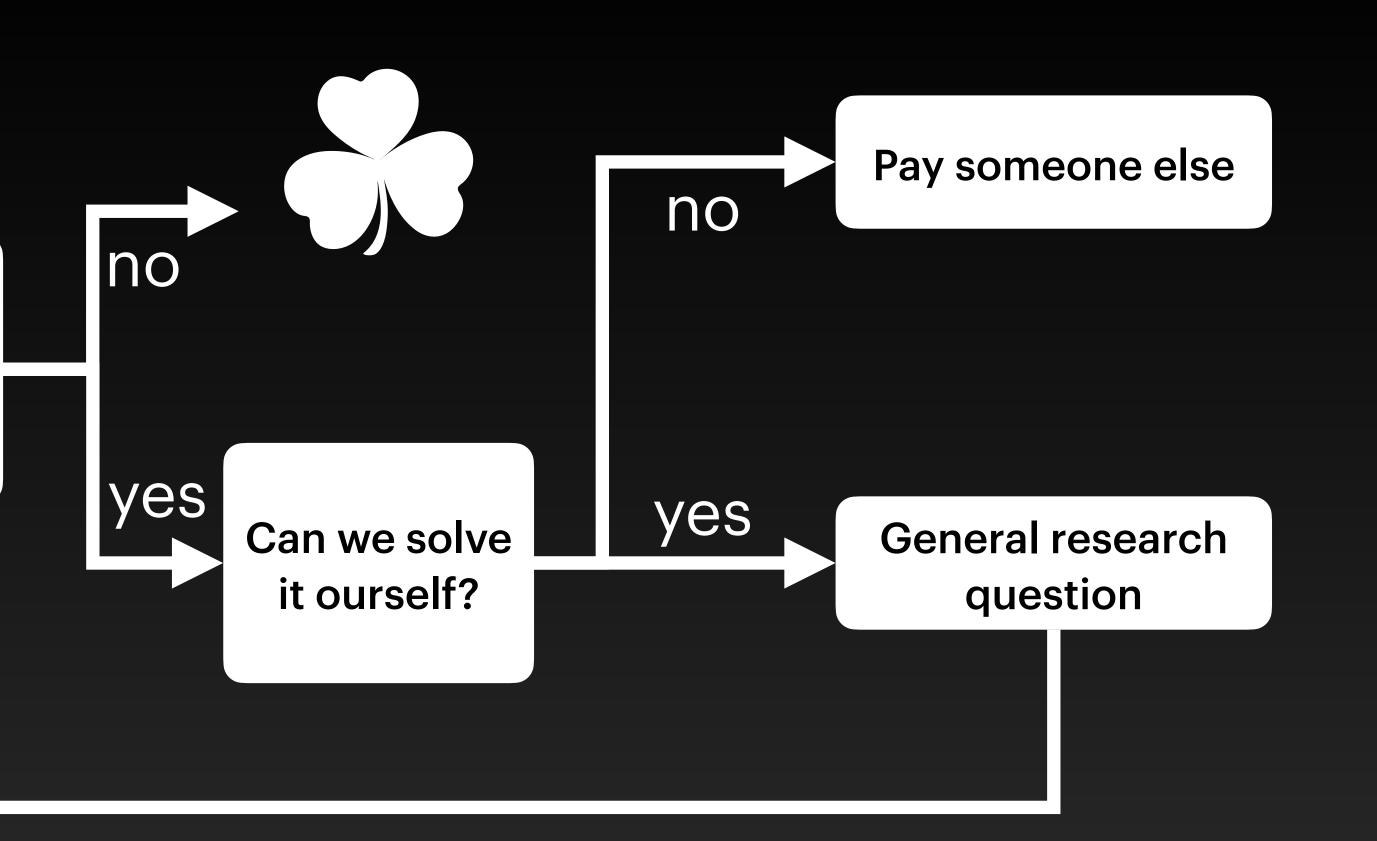
New research question?



**Research paper** 



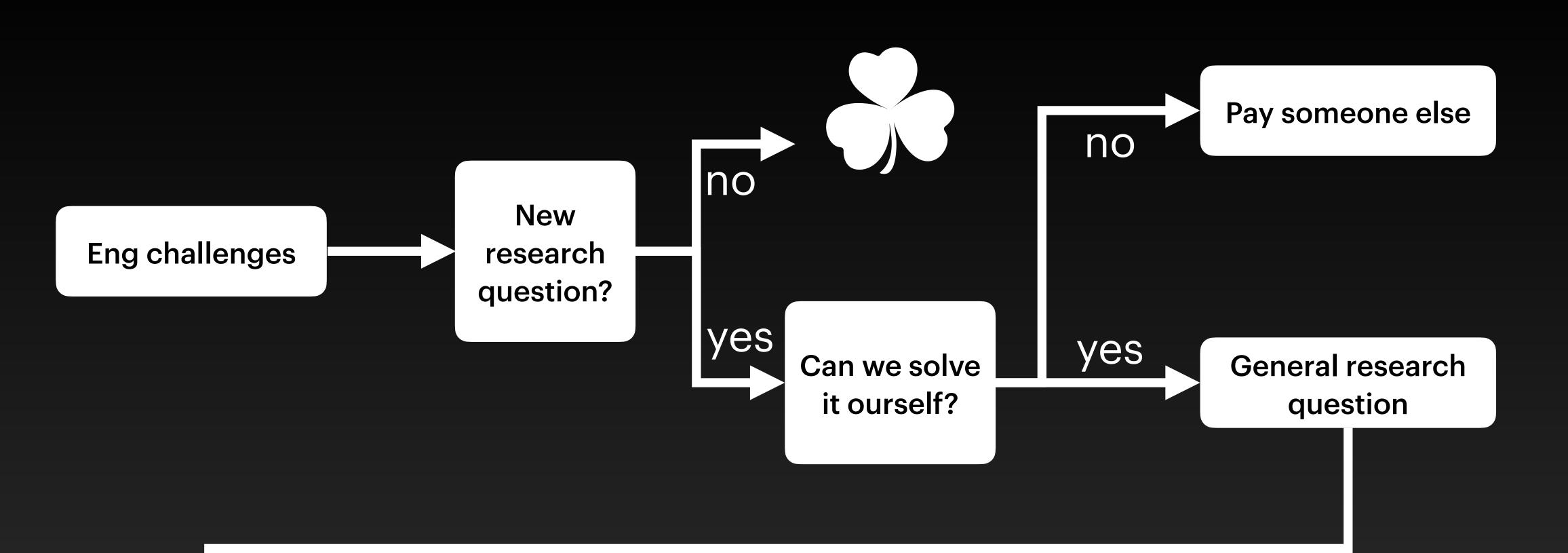
New research question?

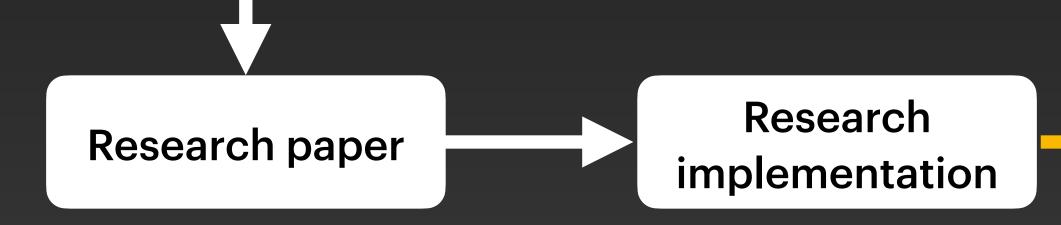




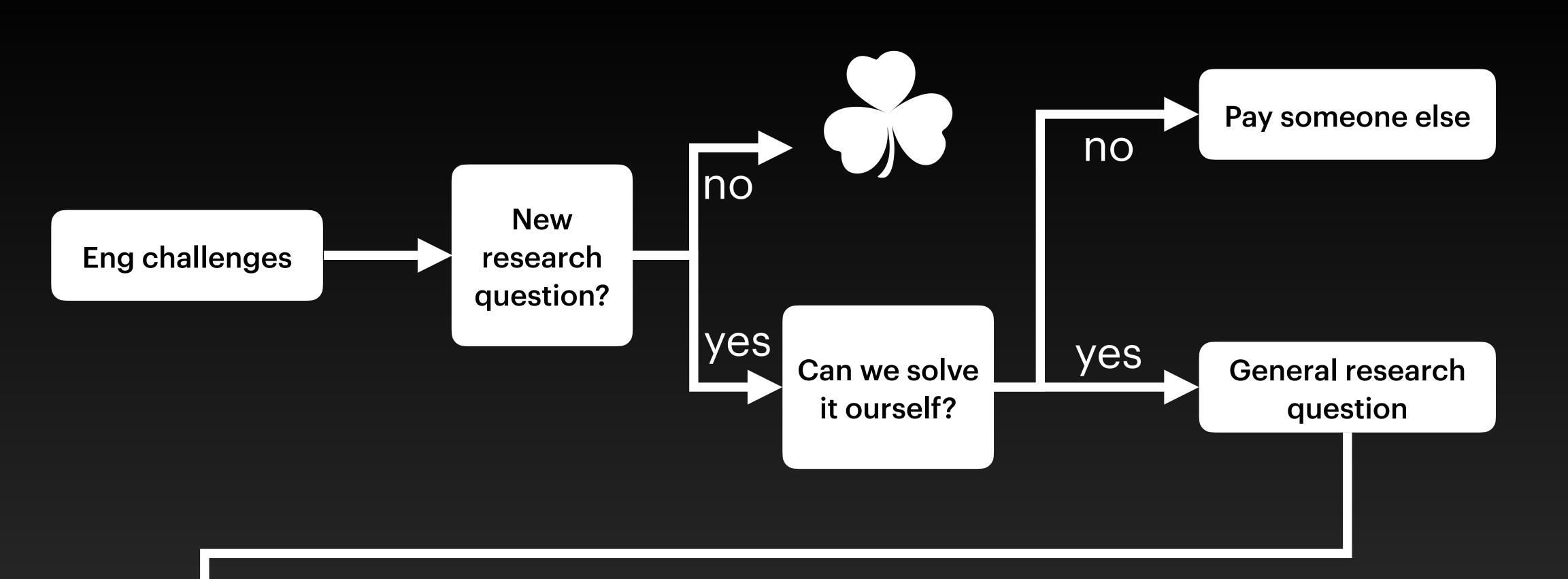


Research implementation

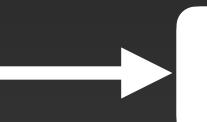








### **Research paper**



### Research implementation



## Research Questions

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

### Lessons Learnec

- 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

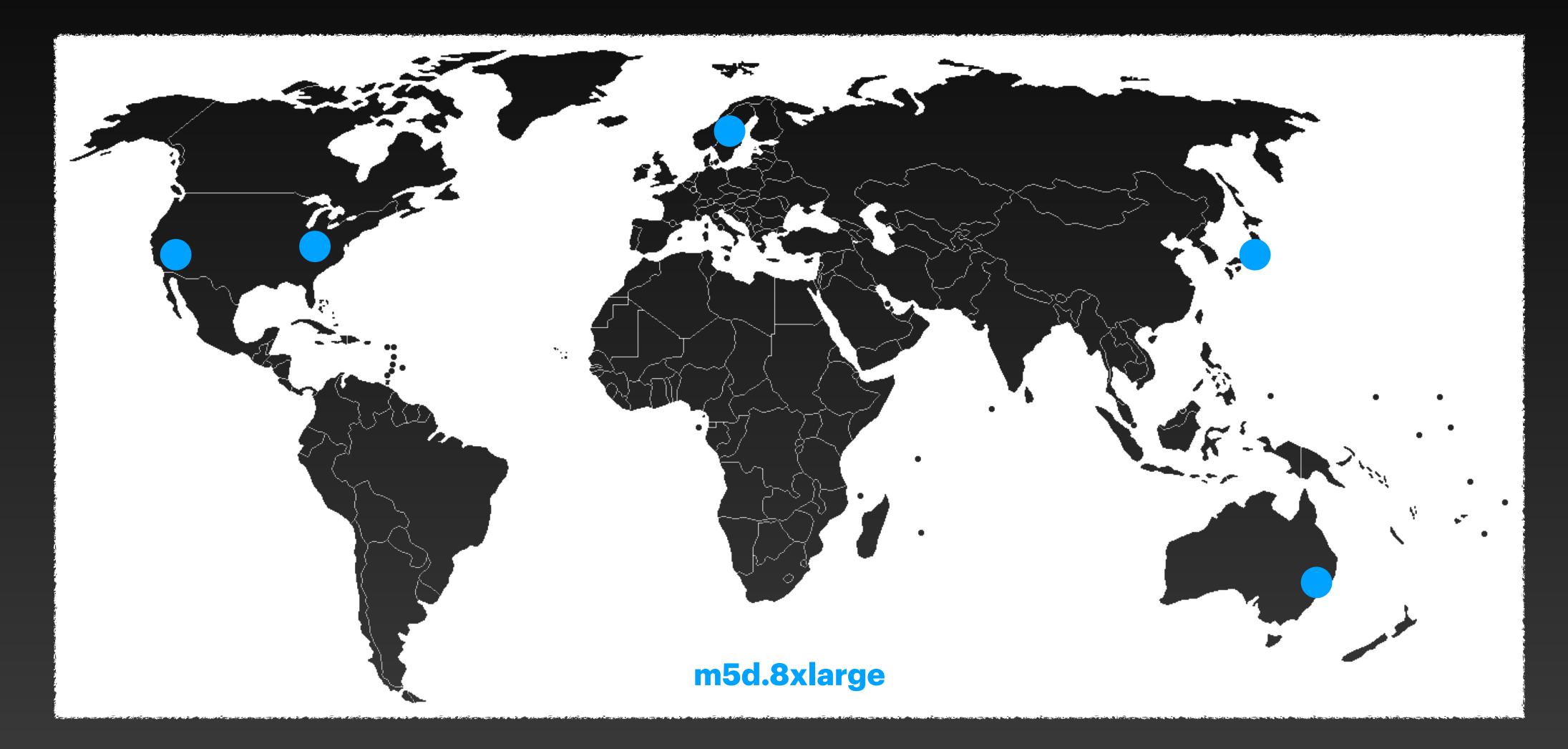
## https://github.com/mystenlabs/mysticeti

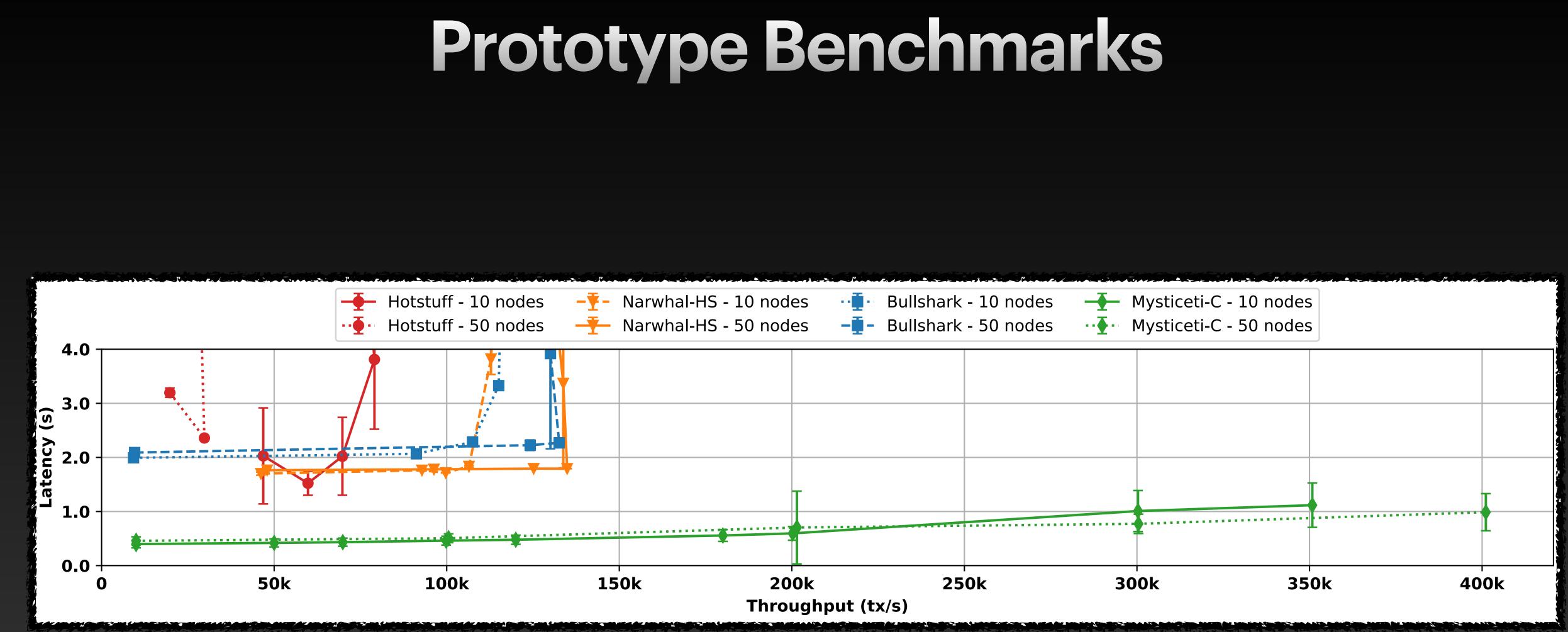
## Implementation

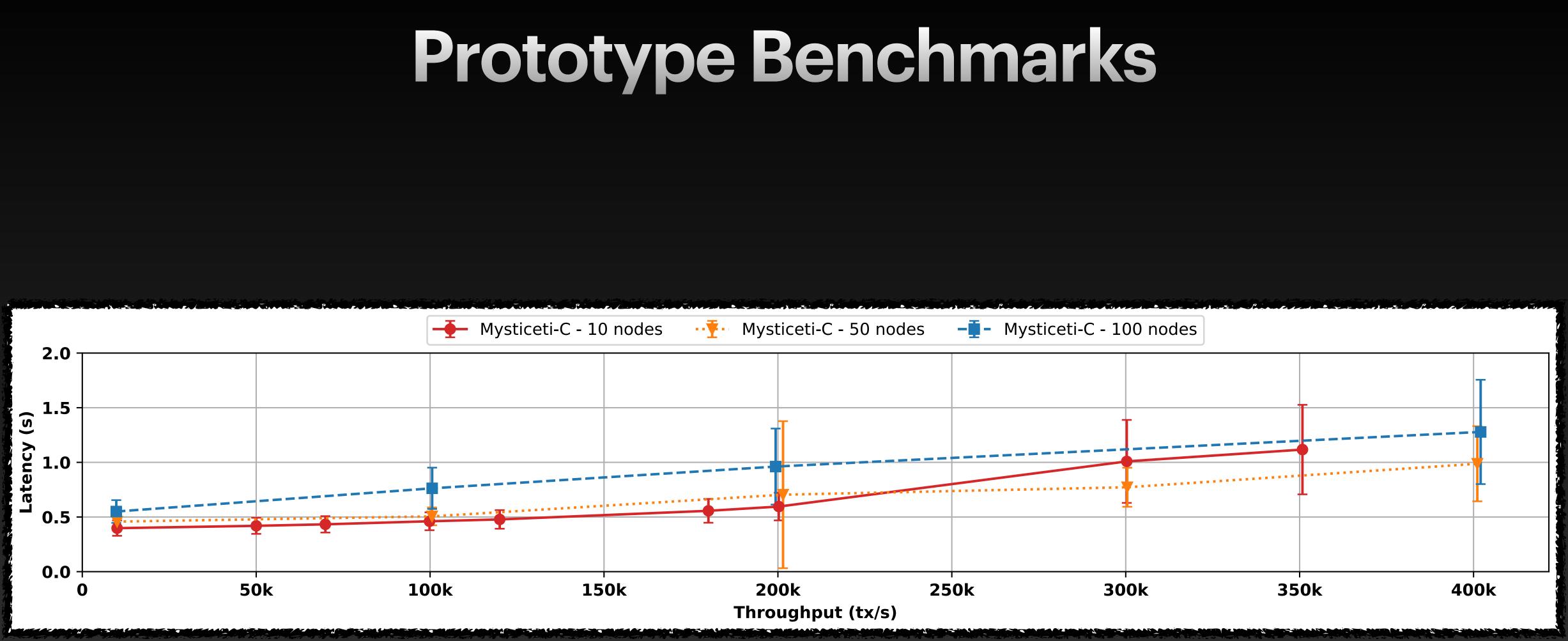
- Synchronous core
- One Tokio task per peer (limiting resource usage)
- DTE simulator

## https://github.com/mystenlabs/mysticeti

### **Evaluation** Experimental setup on AWS







### Mysticeti **Key Limitations**

- Block Synchroniser
- Parallelise block creation and synchronisation
- Rigid DAG structure?