

# Shoal: Improving DAG-BFT Latency and Robustness

FC'24

Alexander Spiegelman, Rati Gelashvili, Balaji Arun, Zekun Li. Aptos

1 Context: DAG-based BFT Consensus . . . . .	3
2 Problem . . . . .	14
3 Solution . . . . .	17
4 Evaluation . . . . .	27

# 1 Context: DAG-based BFT Consensus

- $N = 3f+1$  validators in total
- At most  $f$  validators are faulty

## Goal

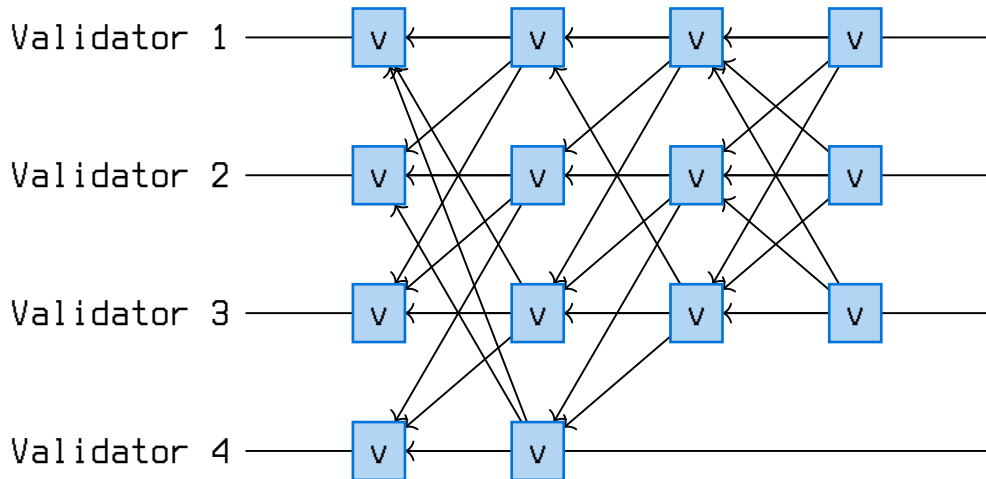
Global agreement on an infinitely growing sequence of some values.

- Historically we have a bunch of protocols which were optimized in the way of reducing communication complexity.
  - ▶ PBFT
  - ▶ Jolteon
  - ▶ ...
  - ▶ Hotstuff - 3500 TPS
- Now we have new generation of protocols
  - ▶ 160kTPS - 600kTPS

### Idea

Separate the network communication layer from the consensus logic.

- Each message contains a set of transactions, and a set of references to previous messages.
- Together, all the messages form a DAG that keeps growing - a message is a vertex and its references are edges.



## Common abstraction

Reliable BFT broadcast (Narwhal based protocols)

Result:

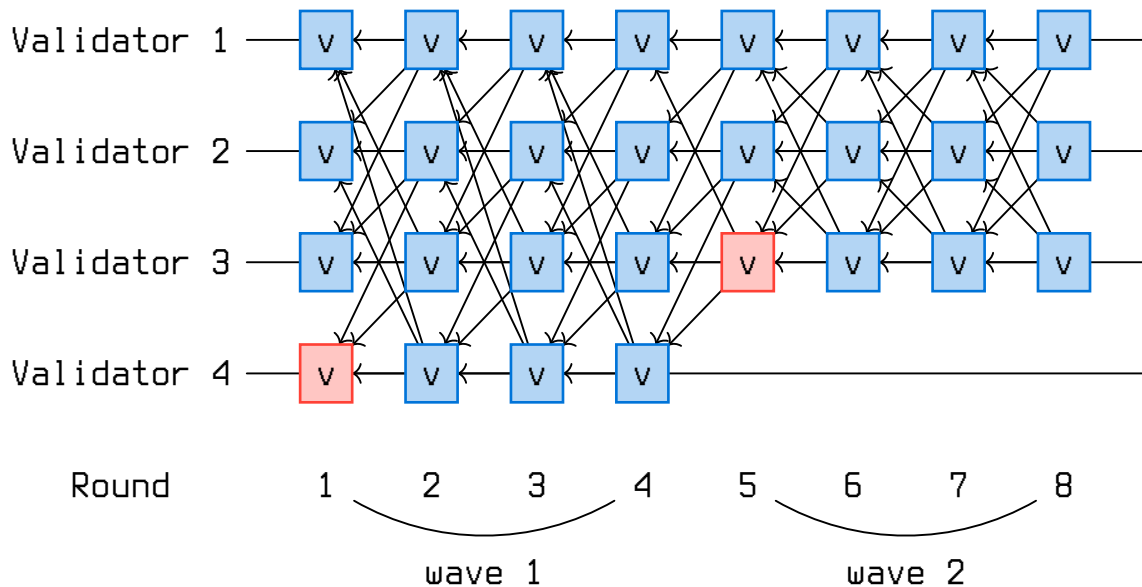
- All honest validators eventually deliver the same vertices and all vertices by honest validators are eventually delivered.
- Causal history of any vertex in both local views is exactly the same.
- All validators eventually see the same DAG

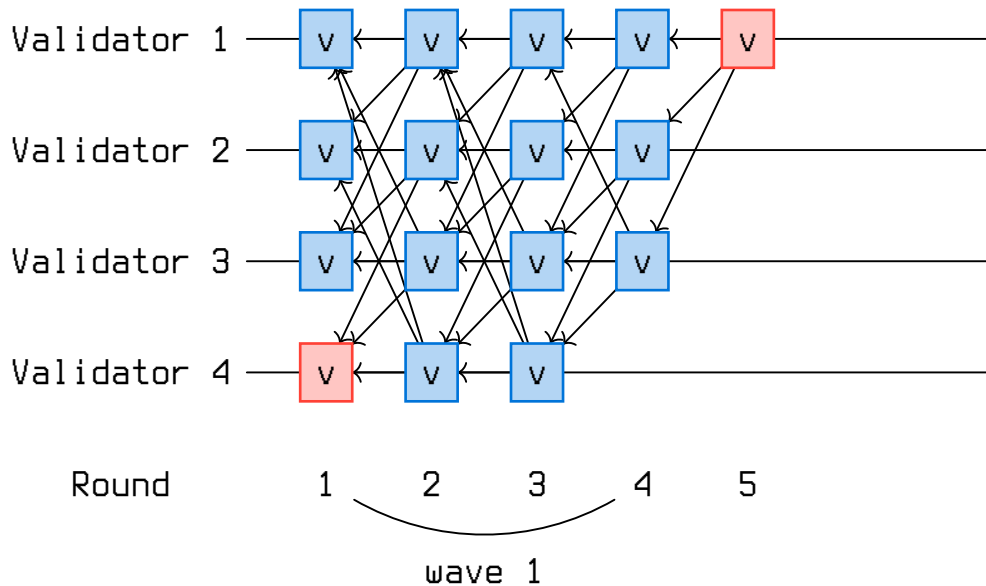


- Interpreting DAG structure as the consensus logic
  - ▶ Outcome - local solving
  - ▶ No need of any extra communication.
- Consensus divided into rounds
- Rounds groups waves
- Each wave contains predefined leader  
(Simplified)

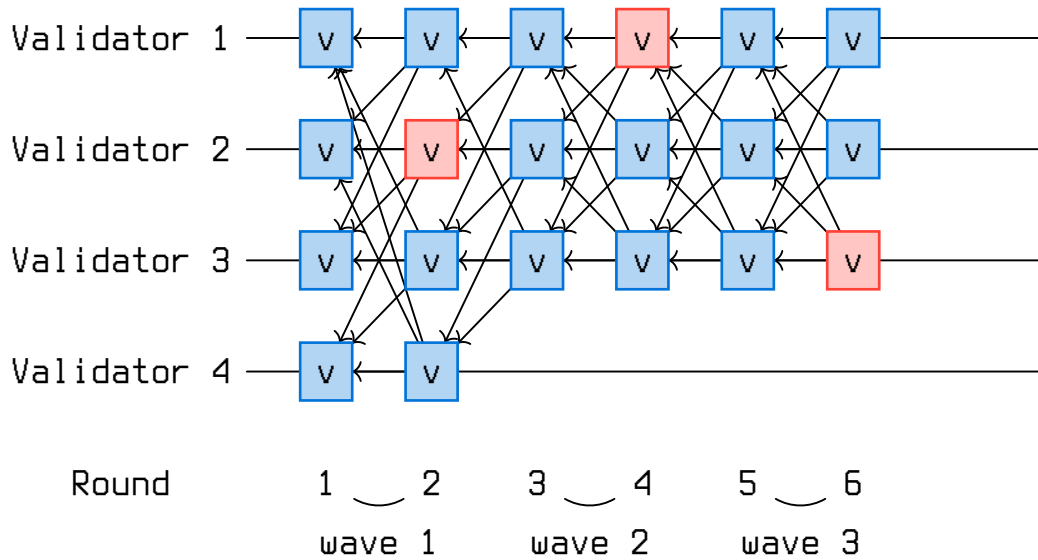
# DAG Waves example [DAG-Rider]

10 / 34





- Consists of two phases:
  1. Each validator determines which leader's vertices to order.
  2. Sequentially traverse these vertices backwards ordering the rest of vertices.
- Ordering happens with roughly linear dependency on wave size
- Larger waves size -> larger latency



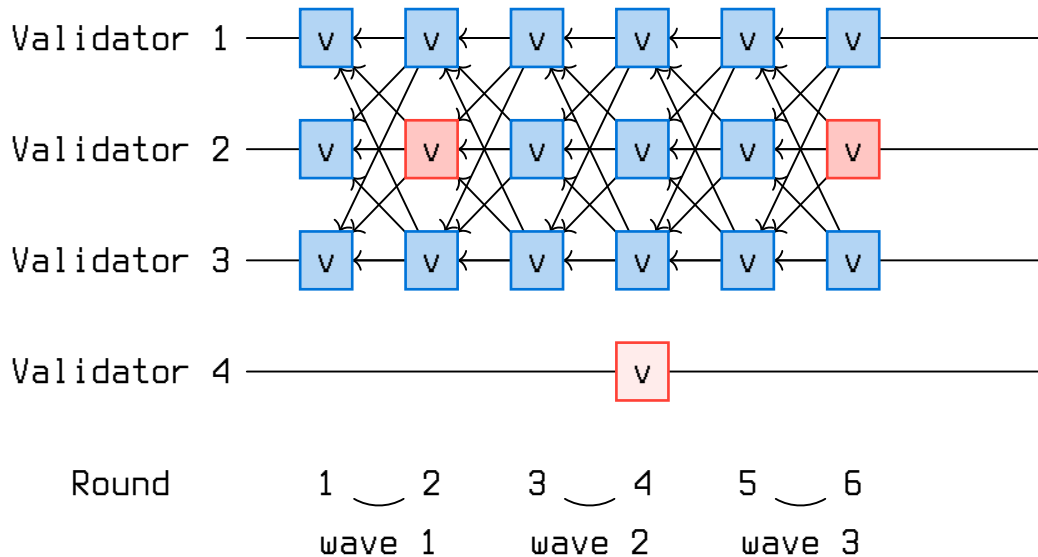
## 2 Problem

## Problem №1

Sparse leader's vertices.

Protocol	Common case round latency	Async round latency
DAG-Rider	4	$E(6)$
Tusk	3	$E(7)$
Bullshark	2	$E(6)$

- Ideally we want to commit something each round.





## 3 Solution

1. Pre-determined leaders each  $k$  rounds.
2. Order leaders. Same local ordering on honest validators.
3. Order casual histories.

## Abstract property

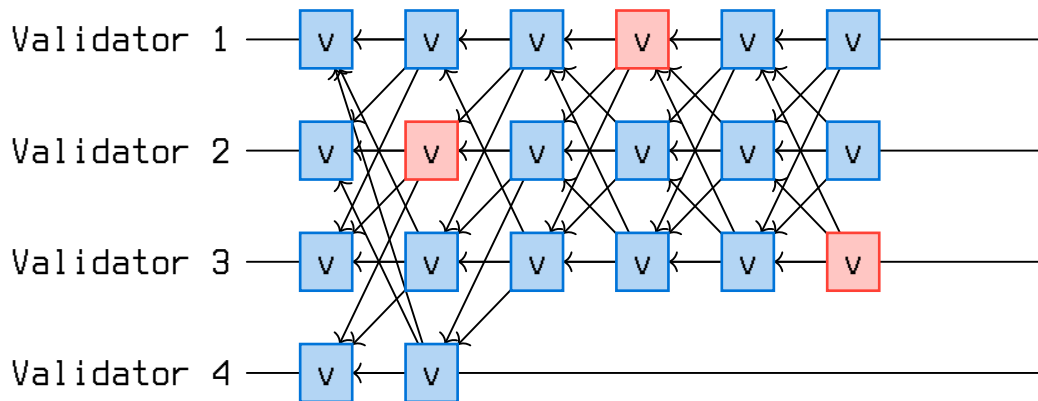
Given a Narwhal-based protocol  $\mathbb{P}$ , if all honest validators agree on the mapping from rounds to leaders before the beginning of instance  $\mathbb{P}$ , then they will agree on the first leader each of them orders during execution of  $\mathbb{P}$ .

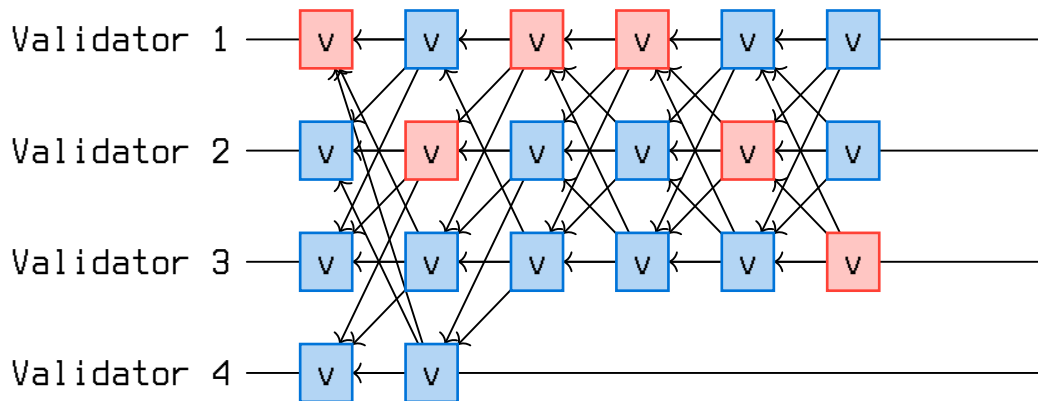
- Protocol agnostic framework
- Suitable for all Narwhal-based protocols

## Idea

Combine batch of protocols instance in black-box manner.

```
1: current_round  $\leftarrow 0$ 
2:  $F: \mathbb{R} \rightarrow \mathbb{L}$ 
3: while true do
4:     Execute  $\mathbb{P}$ , select leaders by  $F$ , starting from
       current_round until the first ordered (not skipped)
       leader is determined.
5:     let  $L$  be the first ordered leader in round  $r$ 
6:     order  $L$ 's casual history according to  $\mathbb{P}$ 
7:     current_round  $\leftarrow r+1$ 
```

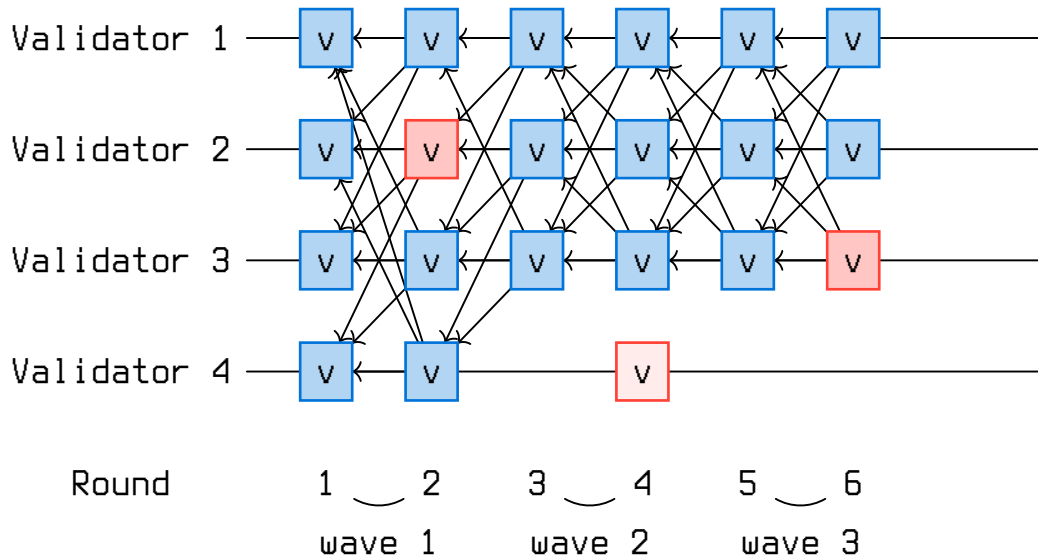




- Byzantine systems are design to tolerate worst-case guarantees.
- However most common problem is slow leaders.

# Example of missing leader

24 / 34

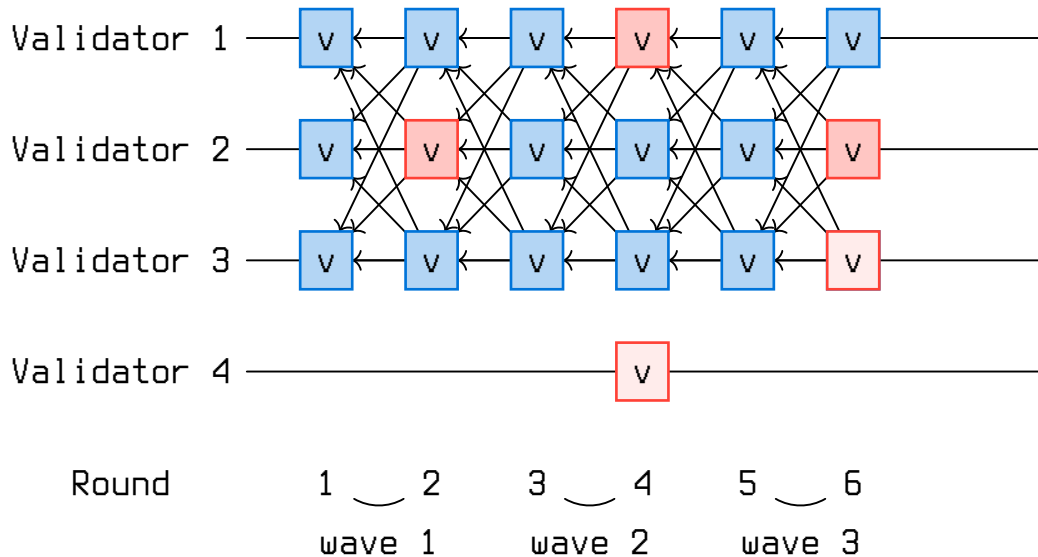




```
1: current_round  $\leftarrow 0$ 
2:  $F: \mathbb{R} \rightarrow \mathbb{L}$ 
3: while true do
4:     Execute  $\mathbb{P}$ , select leaders by  $F$ , starting from
       current_round until the first ordered (not skipped)
       leader is determined.
5:     let  $L$  be the first ordered leader in round  $r$ 
6:     order  $L$ 's casual history according to  $\mathbb{P}$ 
7:     current_round  $\leftarrow r+1$ 
8:     Update  $F$  according to  $L$ 's causal story
```

# Leader change in action

26 / 34



## 4 Evaluation

- Machines:
  - ▶ t2d-standard-32 type virtual machine
  - ▶ 32 vCPUs, 128GB of memory, up to 10Gbps of network bandwidth.
- Cluster:
  - ▶ Google Cloud
  - ▶ Machines spread equally across regions: us-west1, europe-west4, asia-east1.
  - ▶ Latencies: us-west1 asia-east1 [118ms]; europe-west4 asia-east1 [251ms]; us-west1 europe-west4 [133ms]
  - ▶ Cluster size (N): 10 ( $f \leq 3$ ); 20 ( $f \leq 6$ ); 50 ( $f \leq 16$ )
- Data:

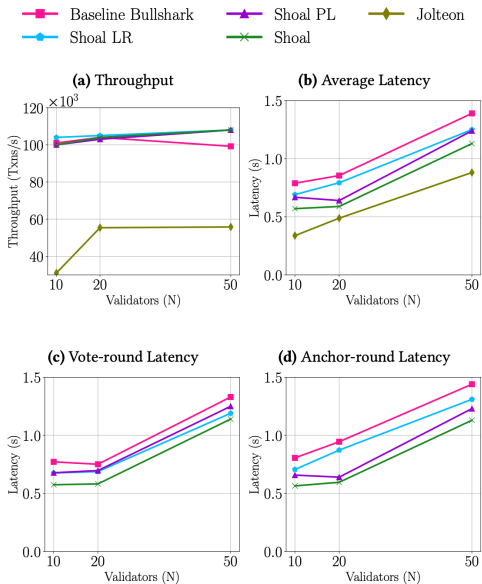
- ▶ Transactions ~270B in size
- ▶ Maximum batch size of 5000 transactions

## Latency

Time elapsed from when a vertex is created from a batch of client transactions to when it is ordered by a validator

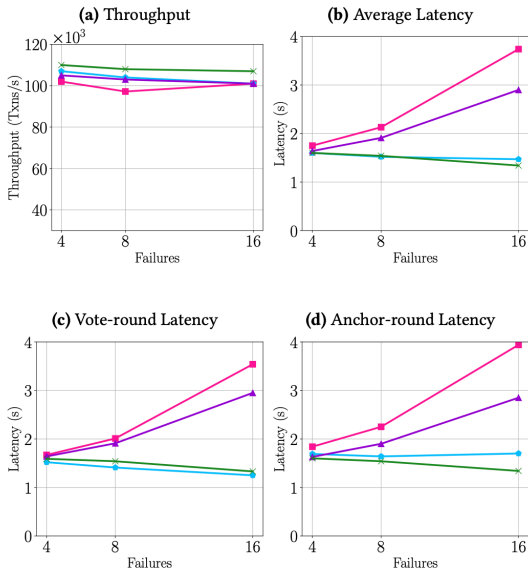
# Results: No failures

31 / 34

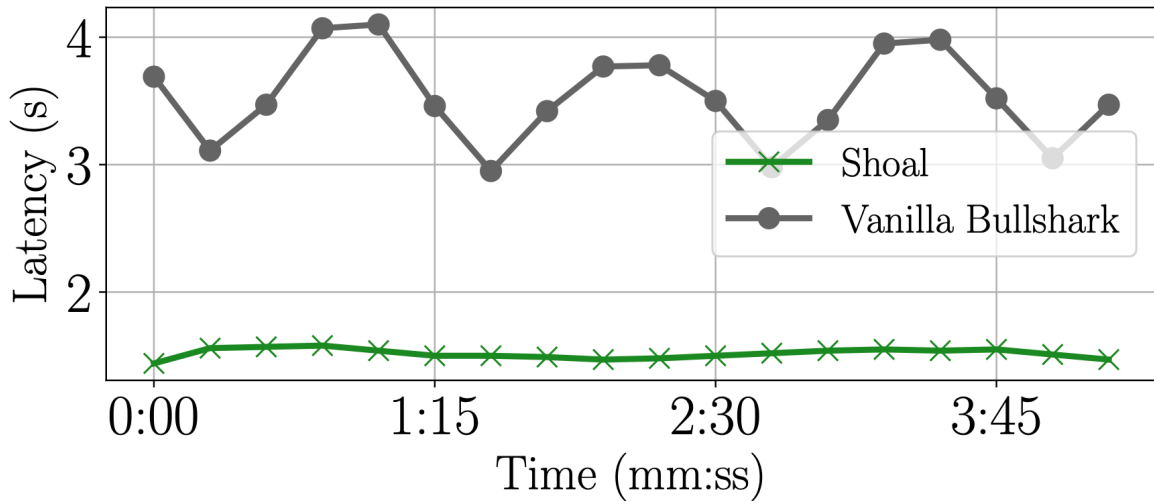


# Results: With failures

32 / 34







- Up to 40% latency reduction in failure-free executions
- Up to 80% reduction in executions with failures against vanilla Bullshark