Narwhal and Tusk

A DAG-based Mempool and Efficient BFT Consensus

Alberto Sonnino

Acknowledgements

George Danezis

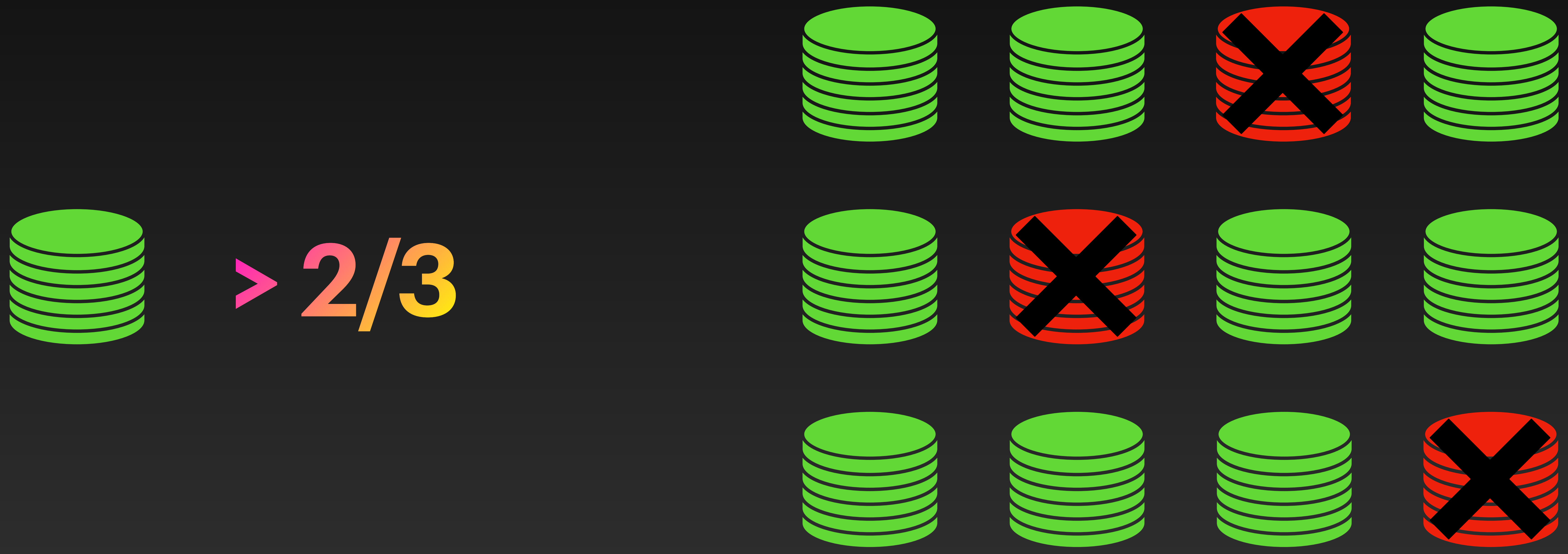
Lefteris Kokoris-Kogias

Alexander Spiegelman

Alberto Sonnino

Facebook Novi

Byzantine Fault Tolerance



> 2/3

How to build (really) high performance blockchains

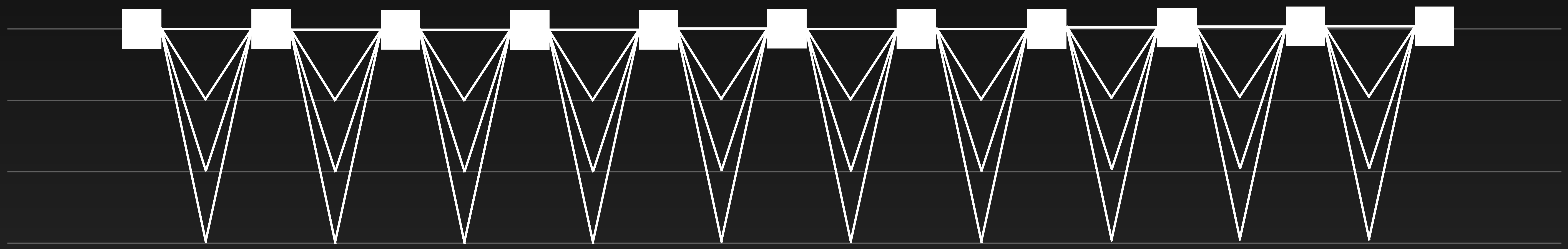
The goal of this project

Current Designs

- Monolithic protocol sharing transaction data as part of the consensus
- Optimize overall message complexity of the consensus protocol

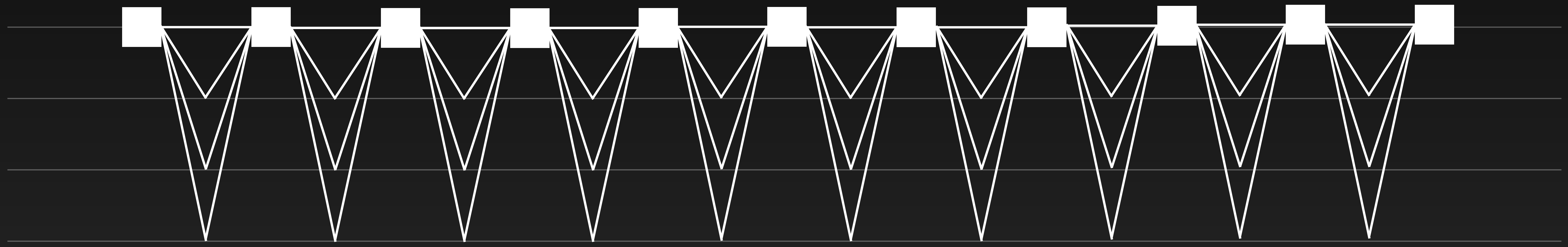
Current Designs

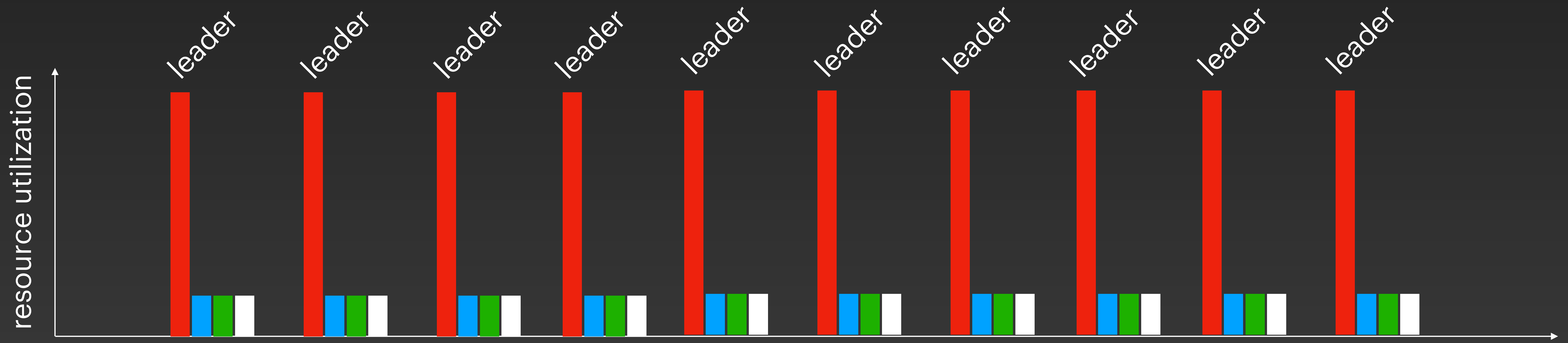
Typical leader-based protocols



Current Designs

Typical leader-based solutions





The mempool is the key

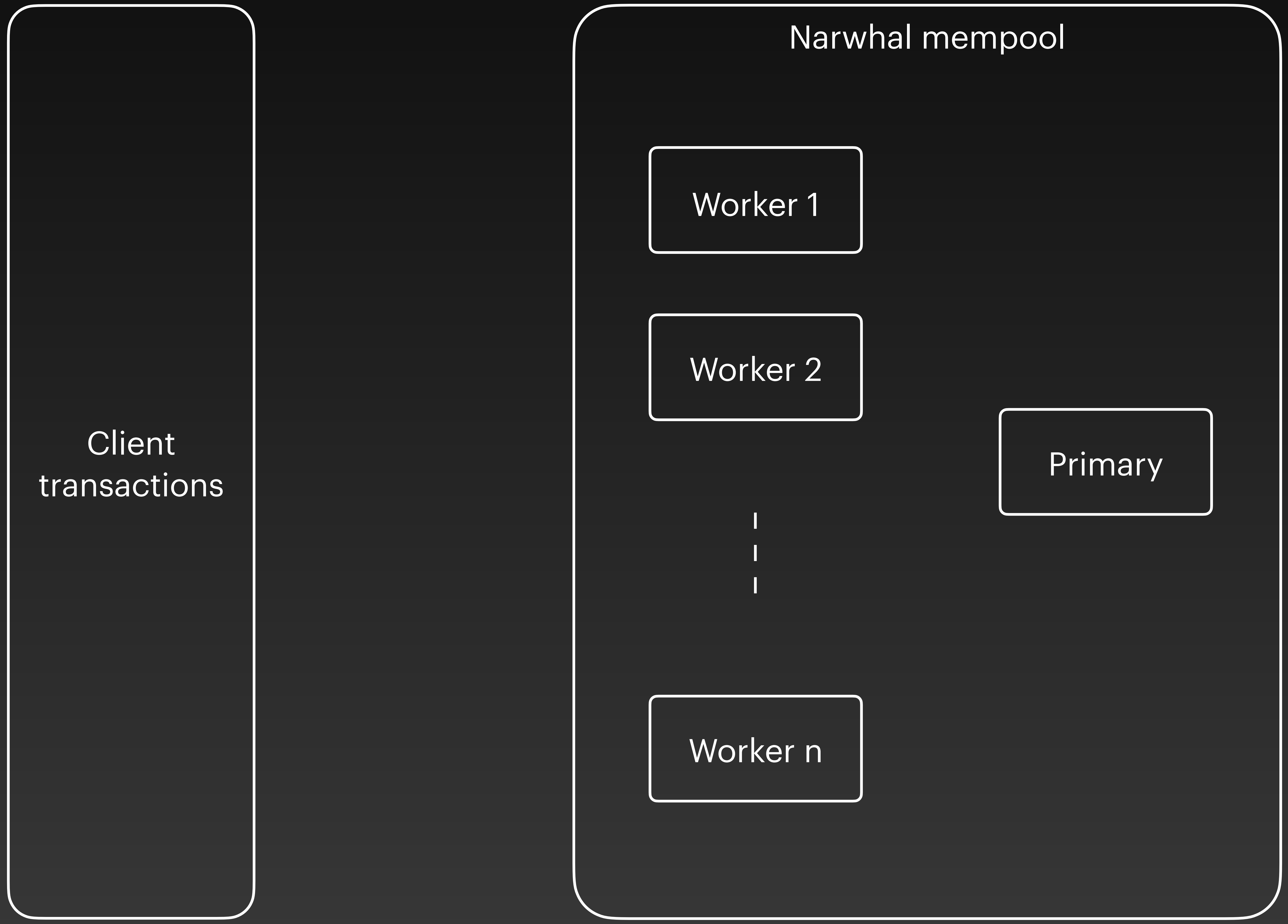
Reaching consensus on metadata is cheap

Narwhal

Dag-based mempool

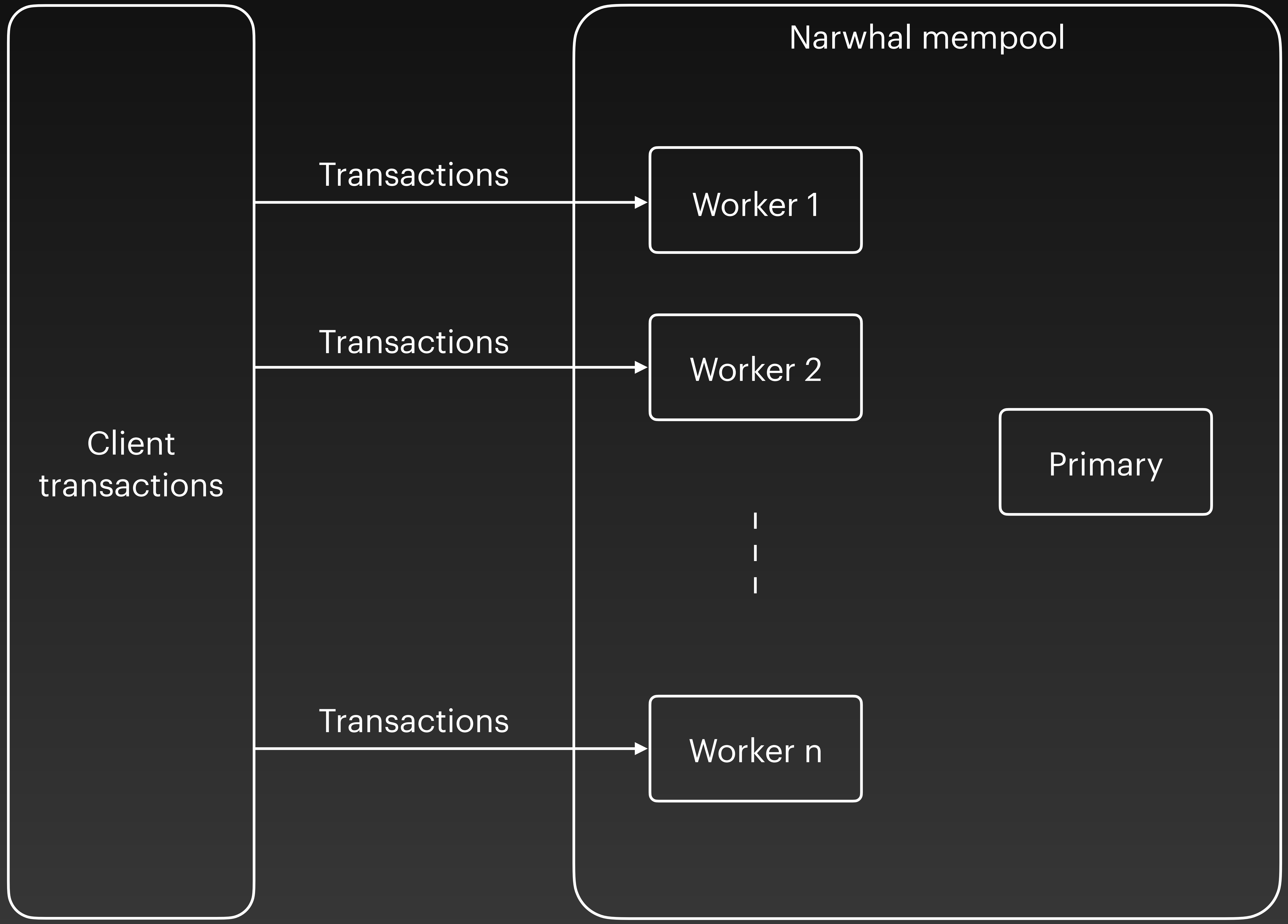
Narwhal

The workers and the primary



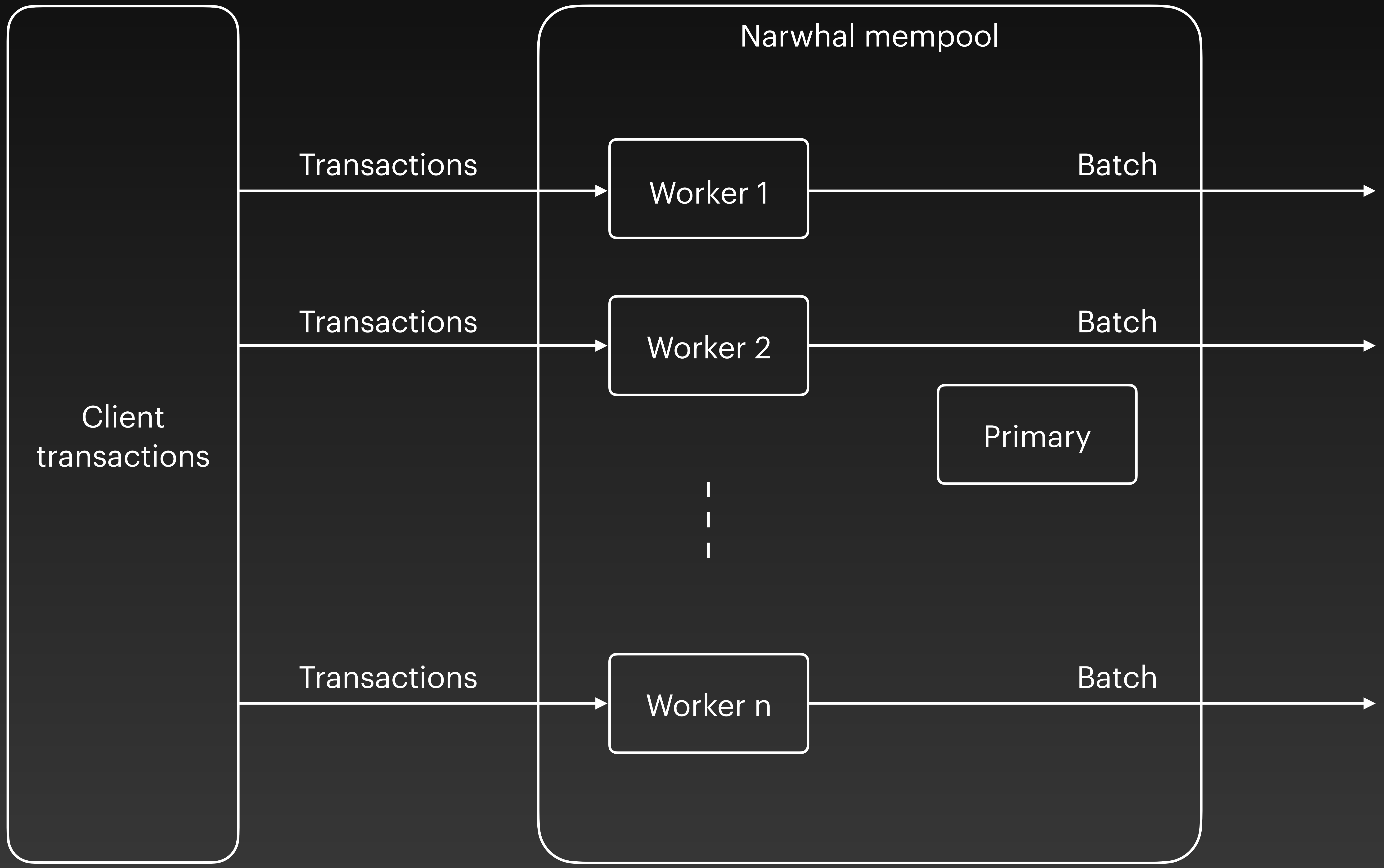
Narwhal

The workers and the primary



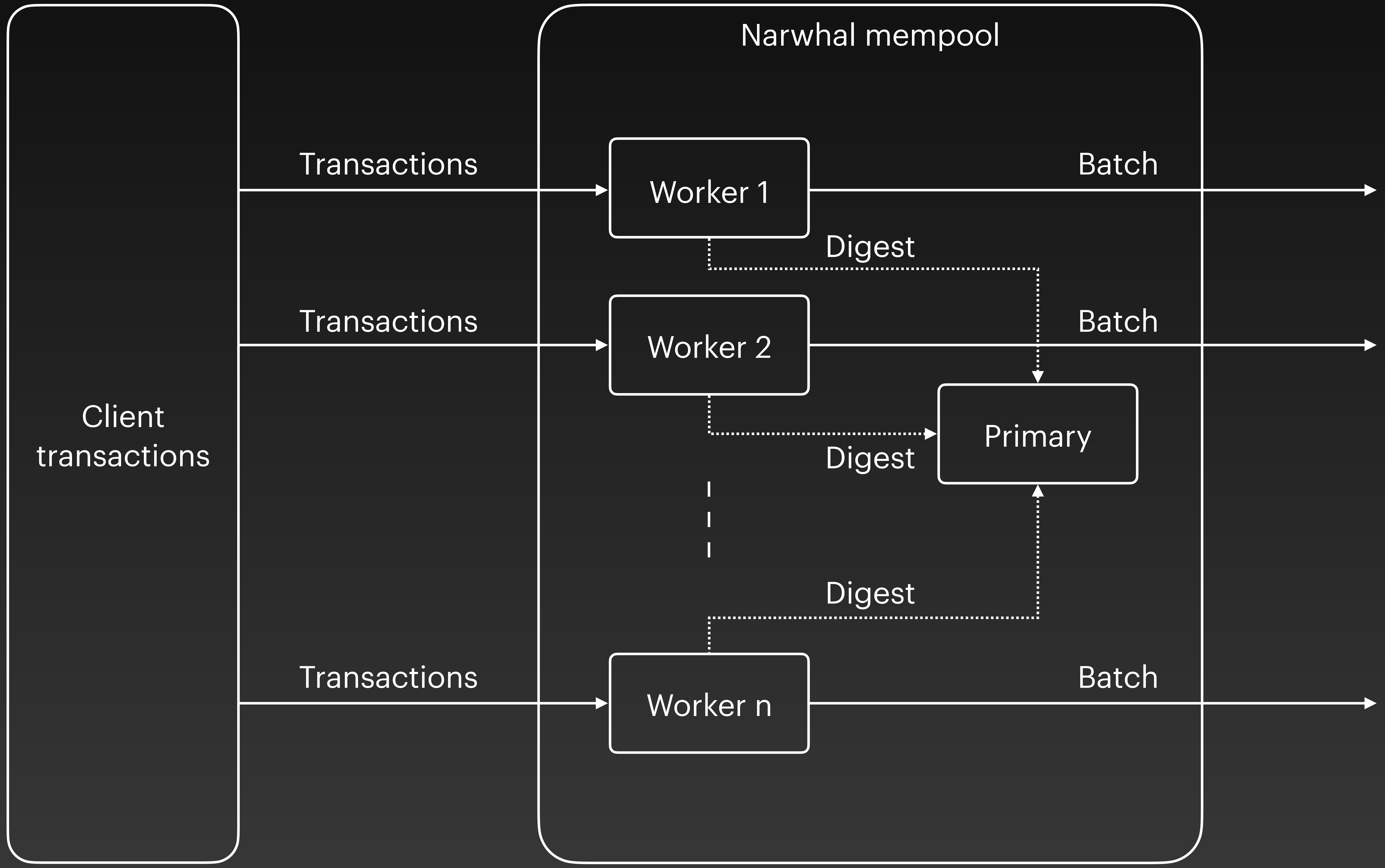
Narwhal

The workers and the primary



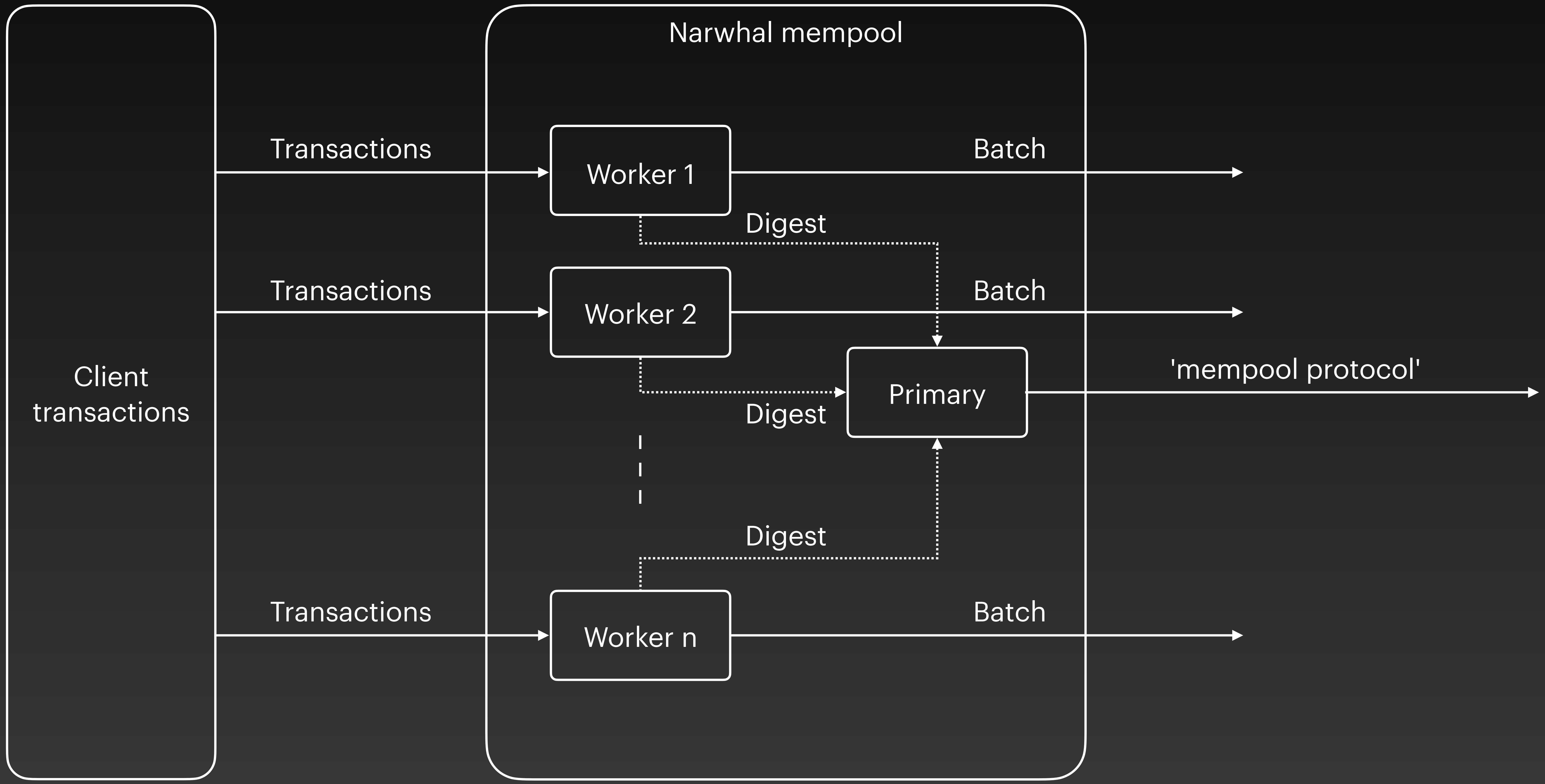
Narwhal

The workers and the primary



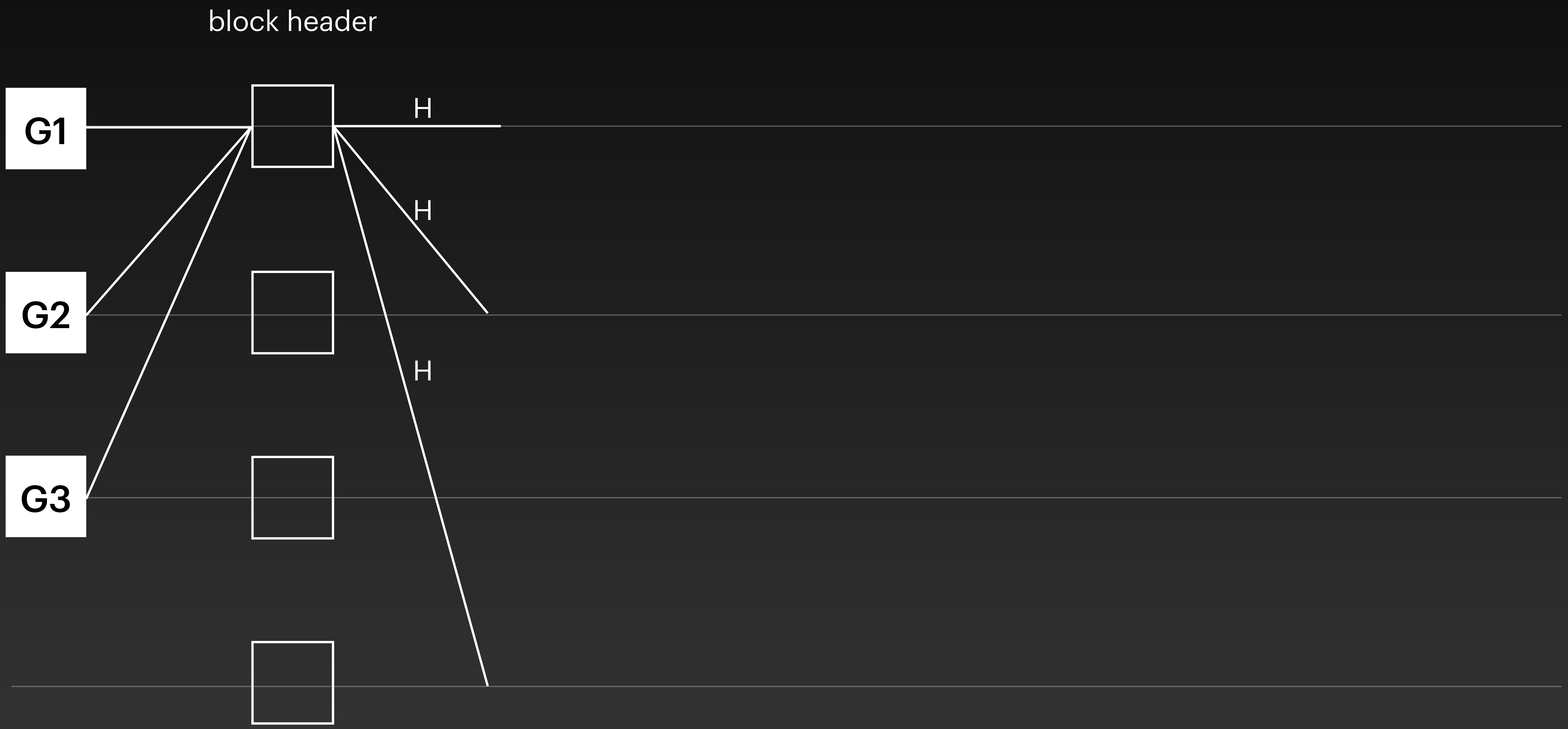
Narwhal

The workers and the primary



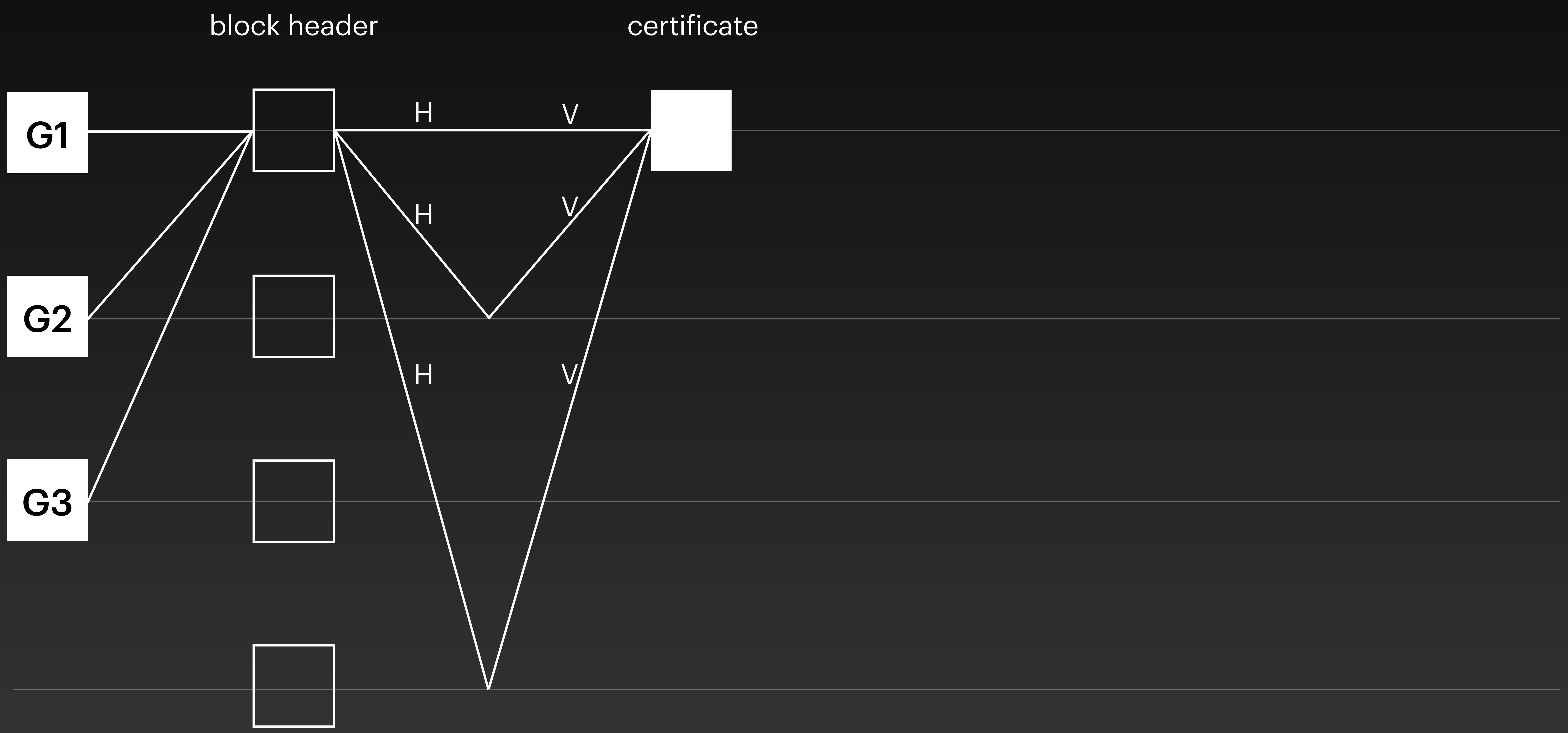
Narwhal

The primary machine



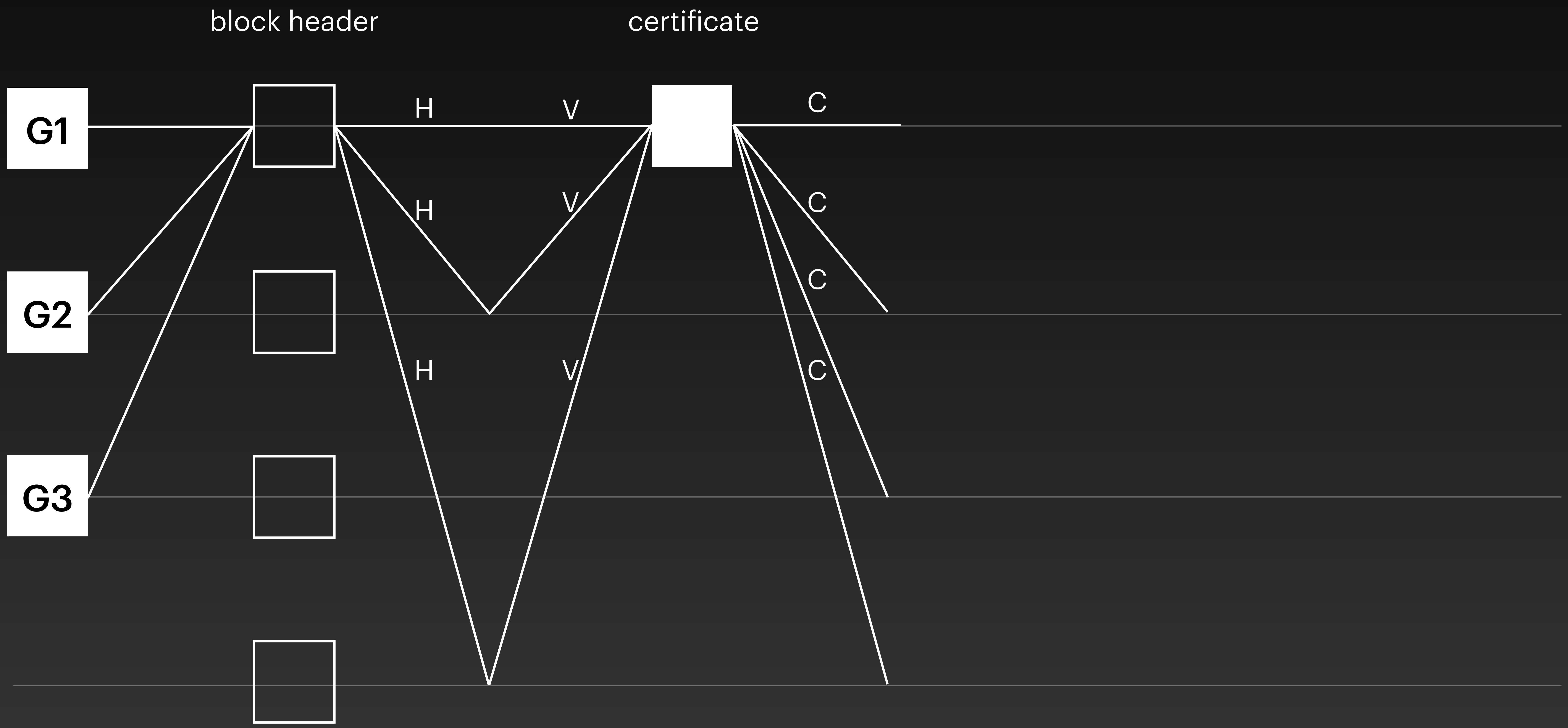
Narwhal

The primary machine



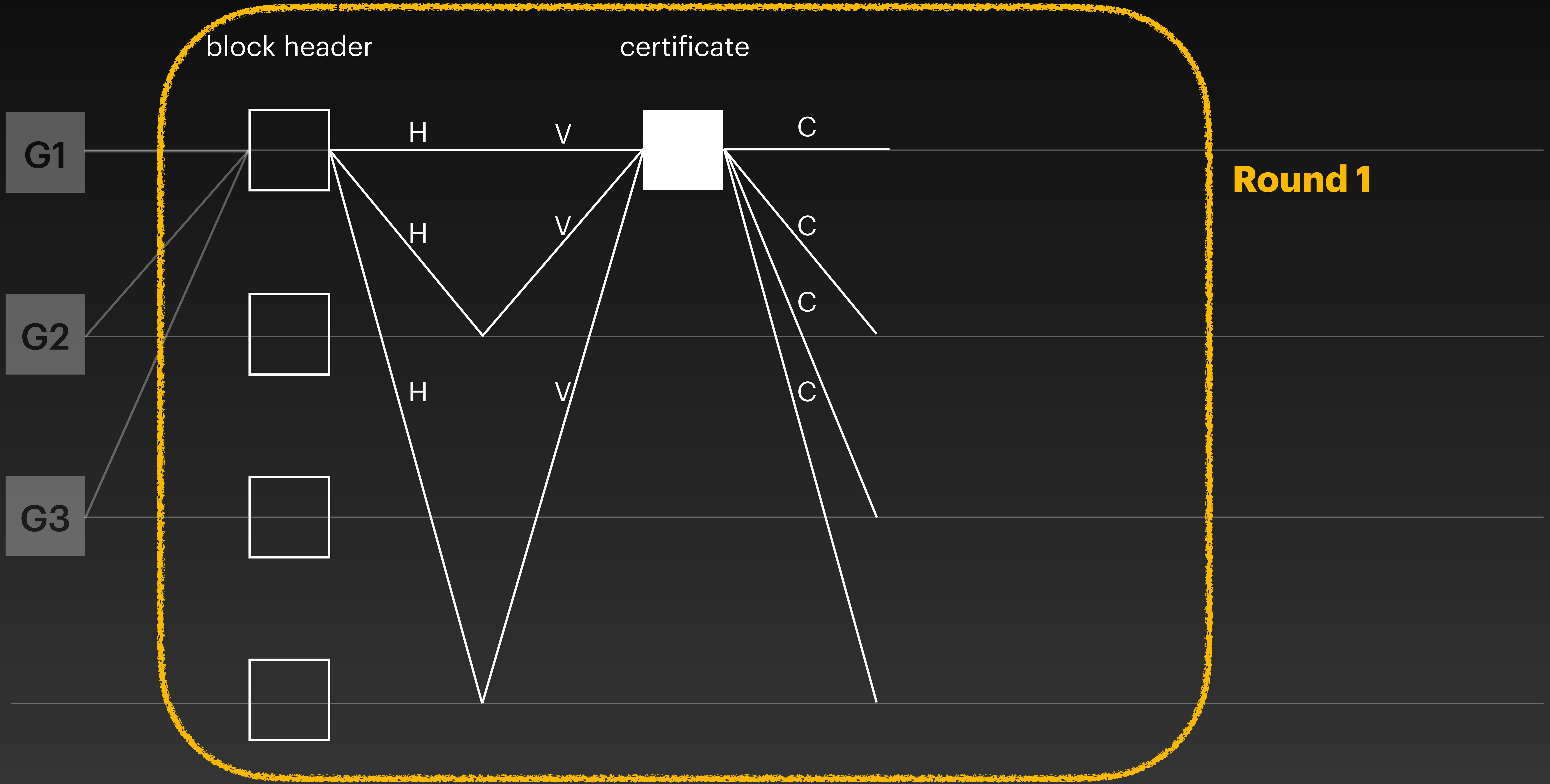
Narwhal

The primary machine



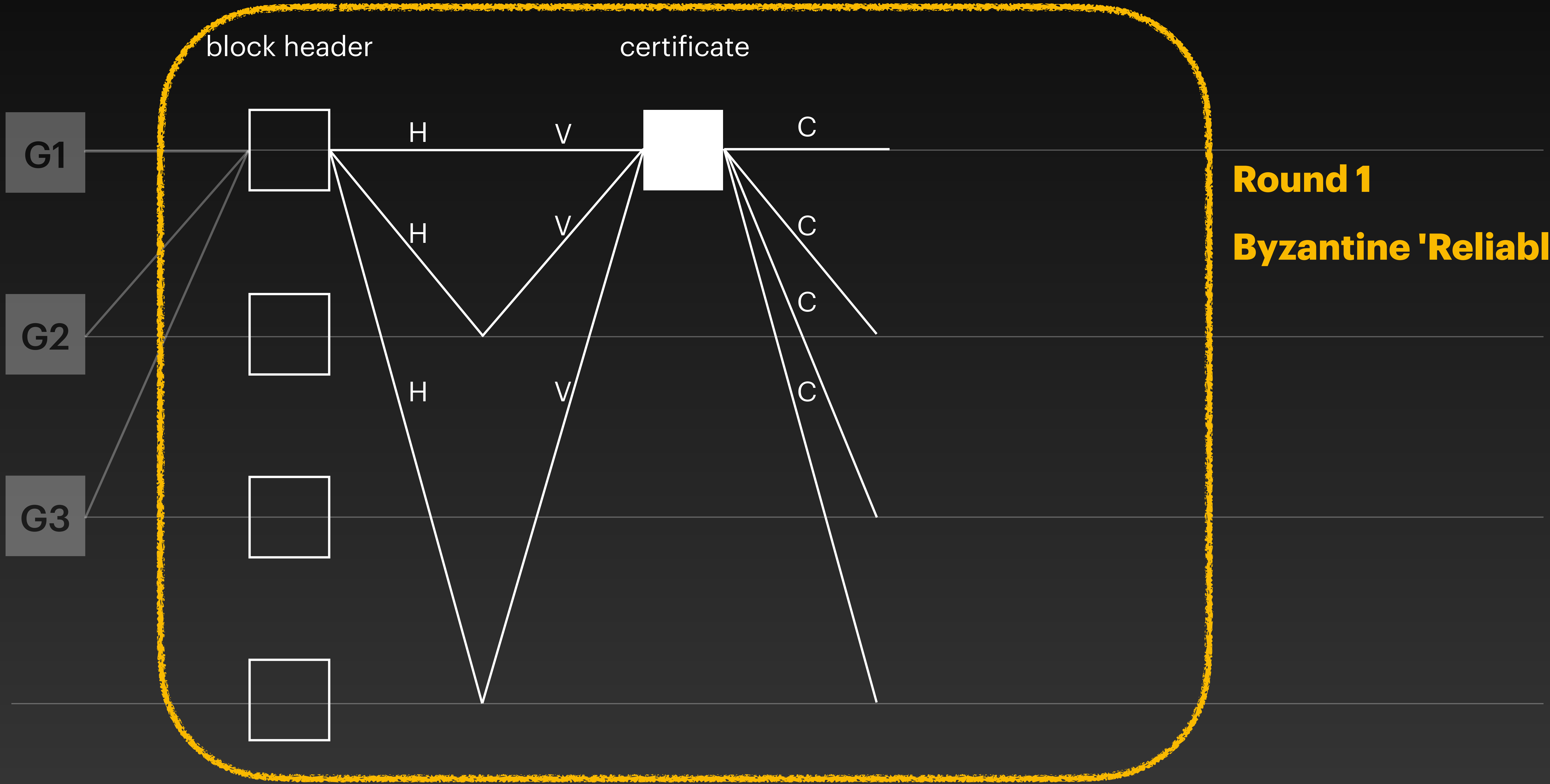
Narwhal

The primary machine



Narwhal

The primary machine

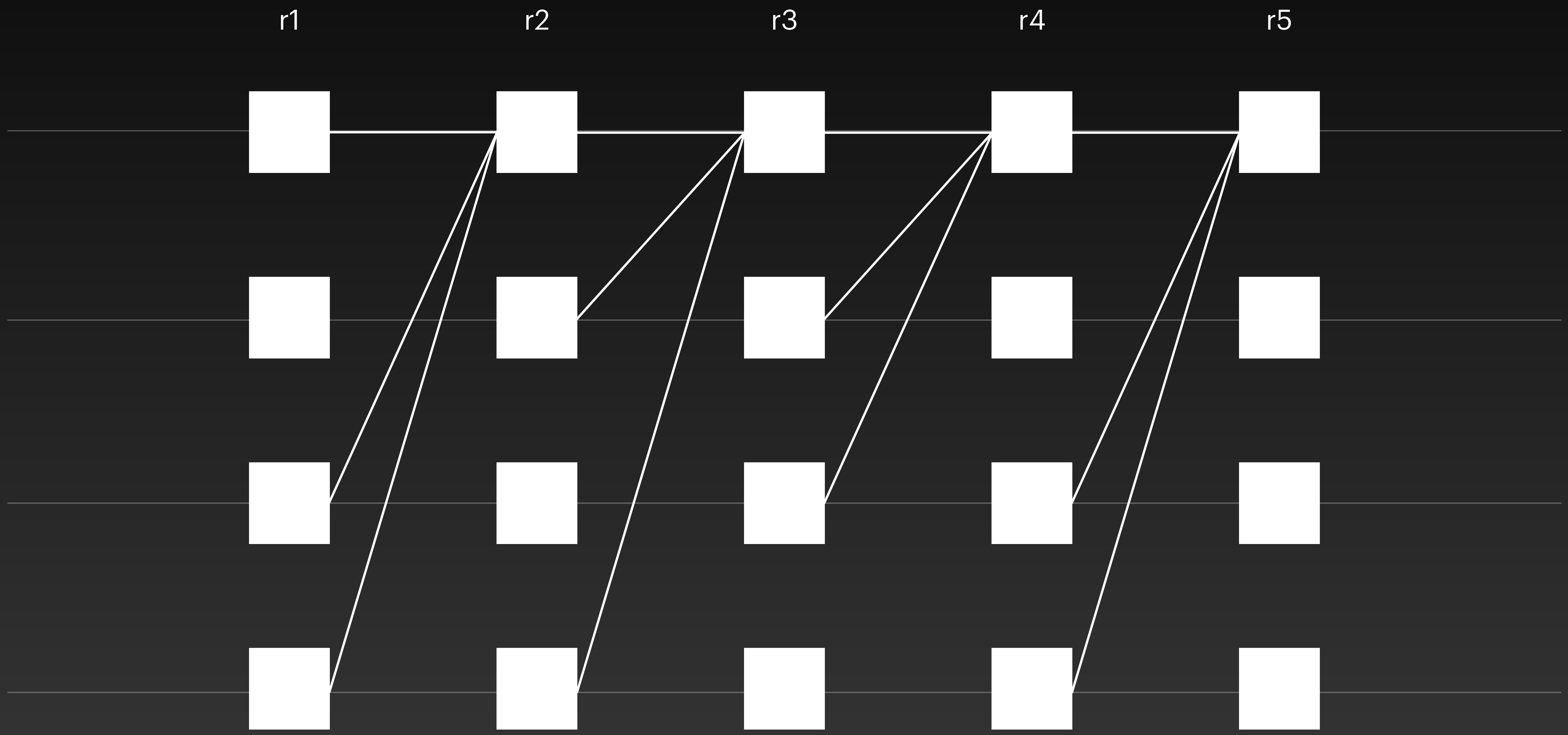


Round 1

Byzantine 'Reliable' Broadcast

Narwhal

The primary machine

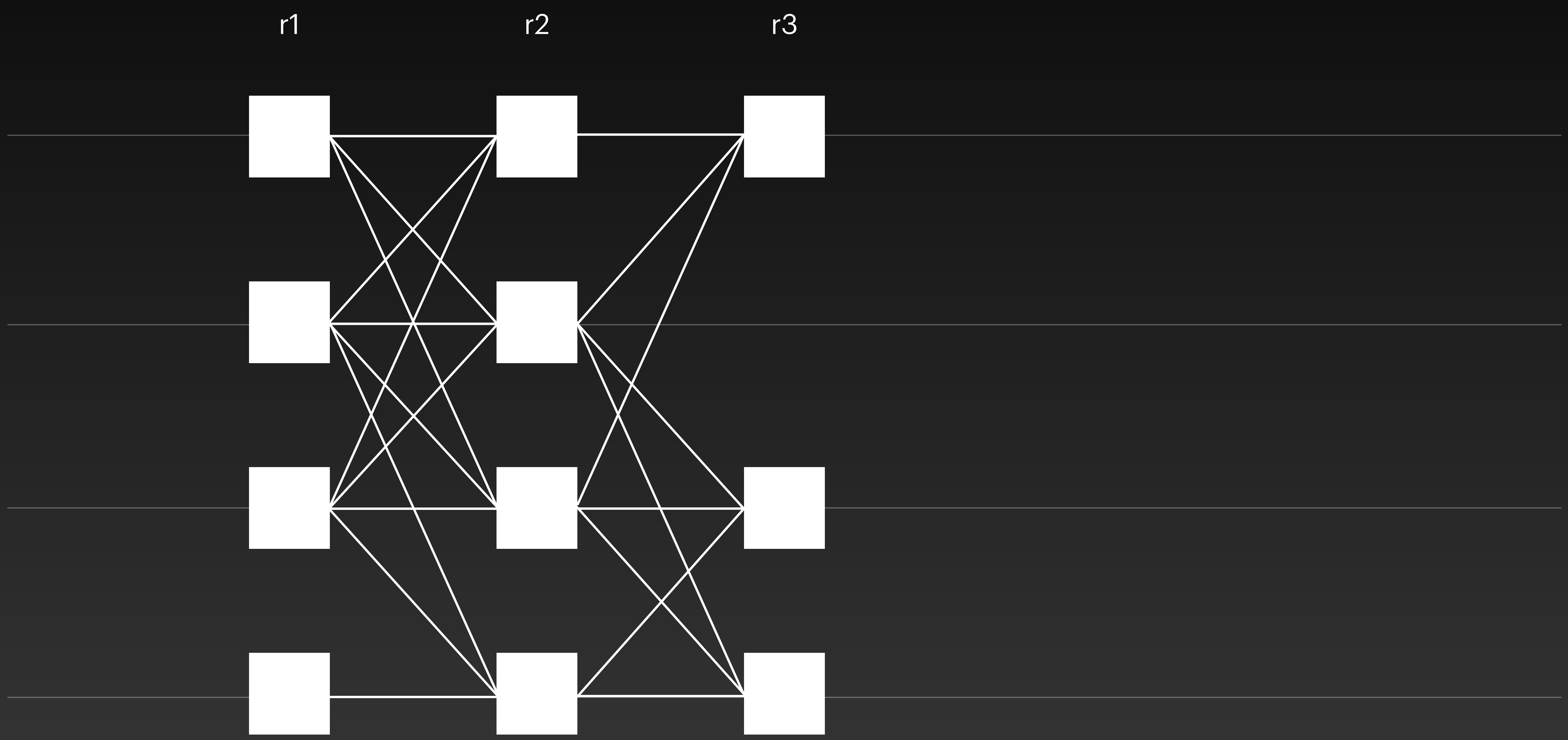


Tusk

Zero-message asynchronous consensus

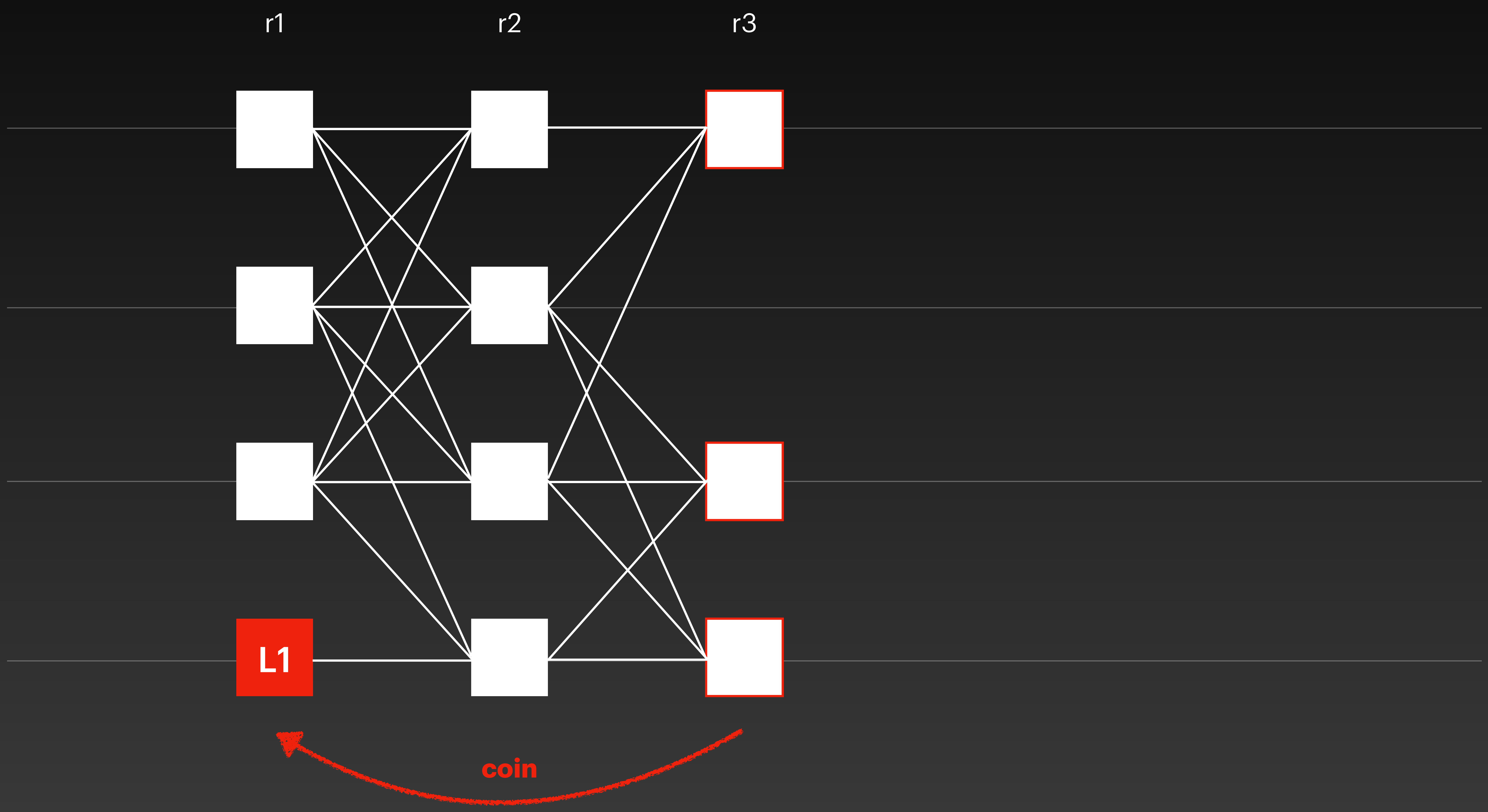
Tusk

Just interpret the DAG



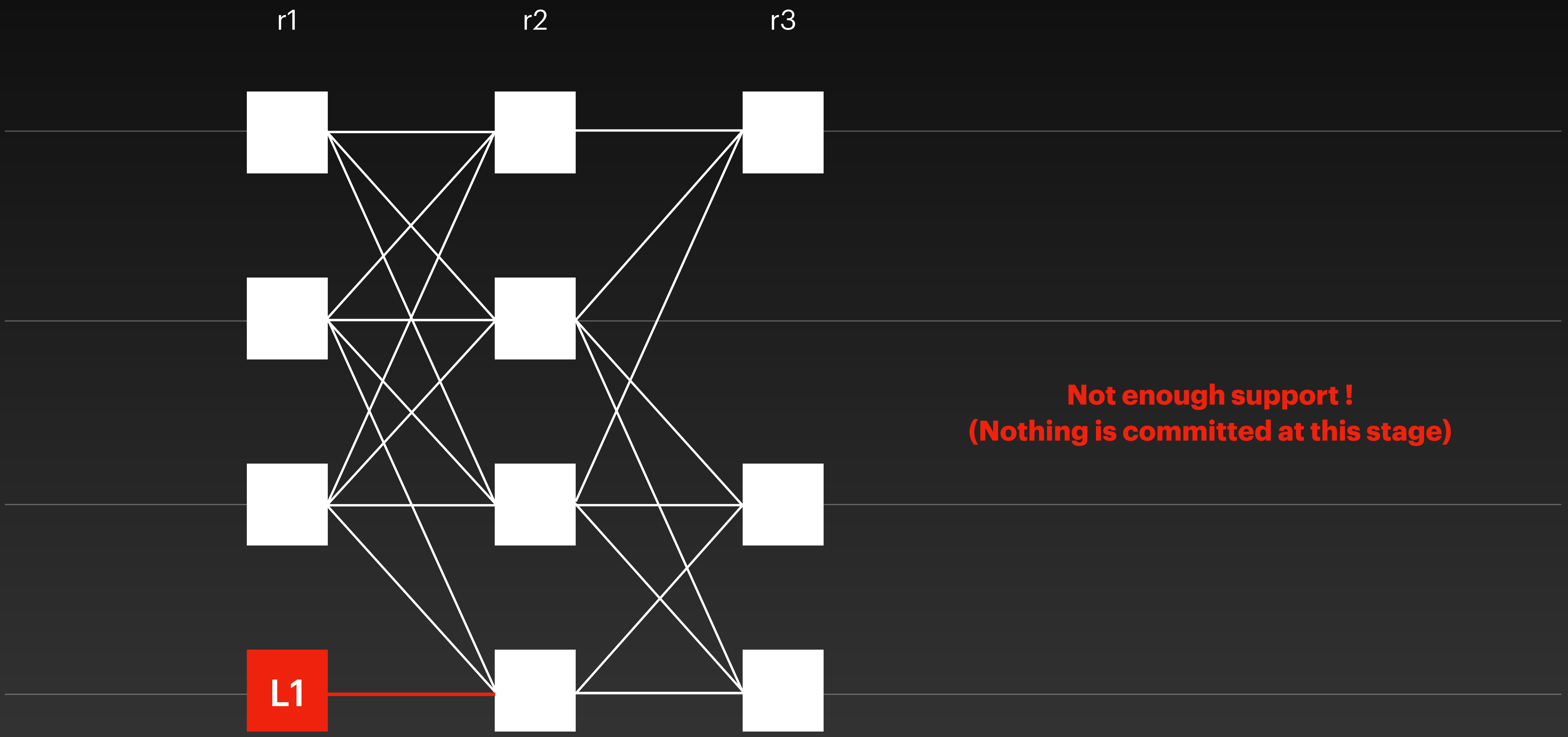
Tusk

The random coin elects the leader of r-2



Tusk

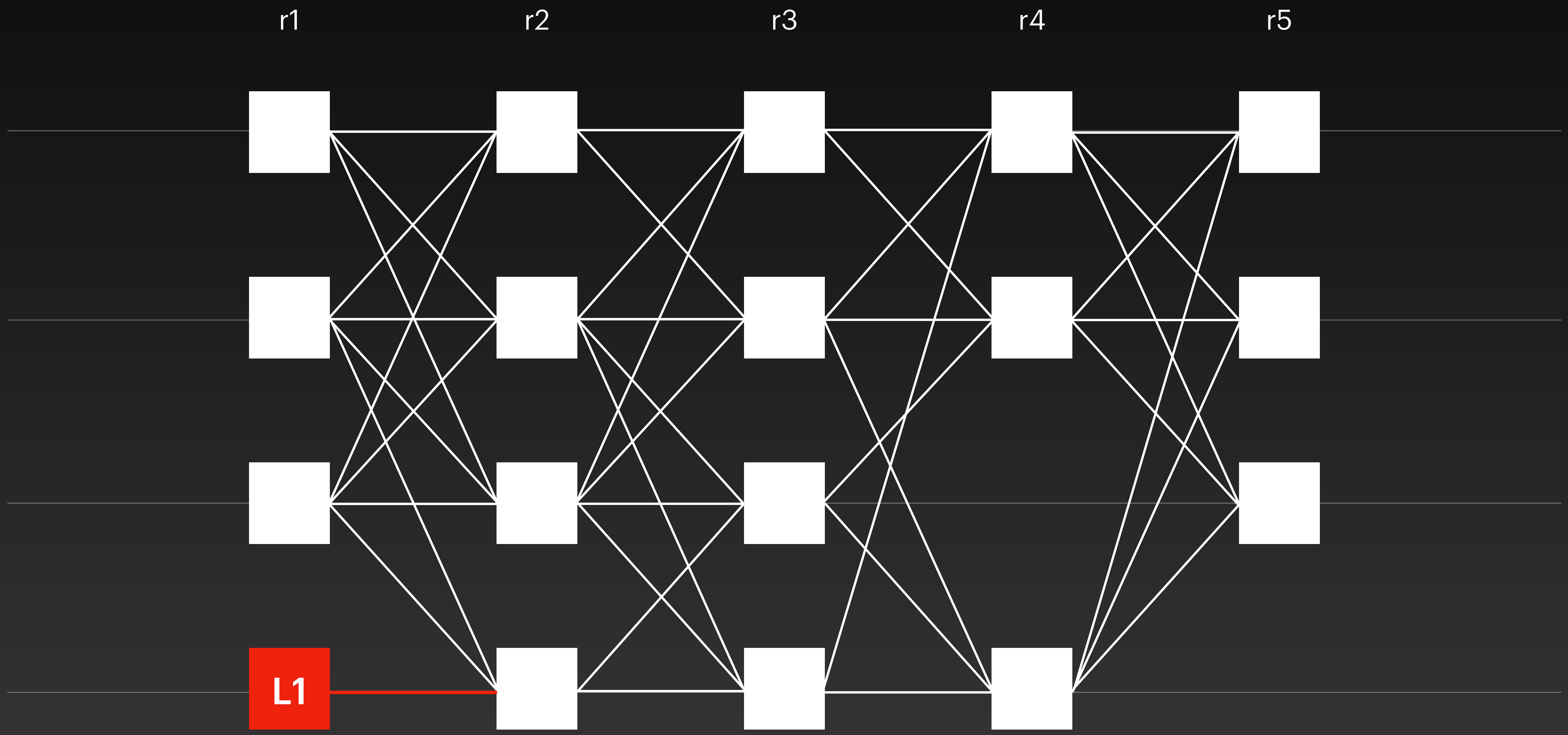
The leader needs f+1 links from round r-1



Not enough support !
(Nothing is committed at this stage)

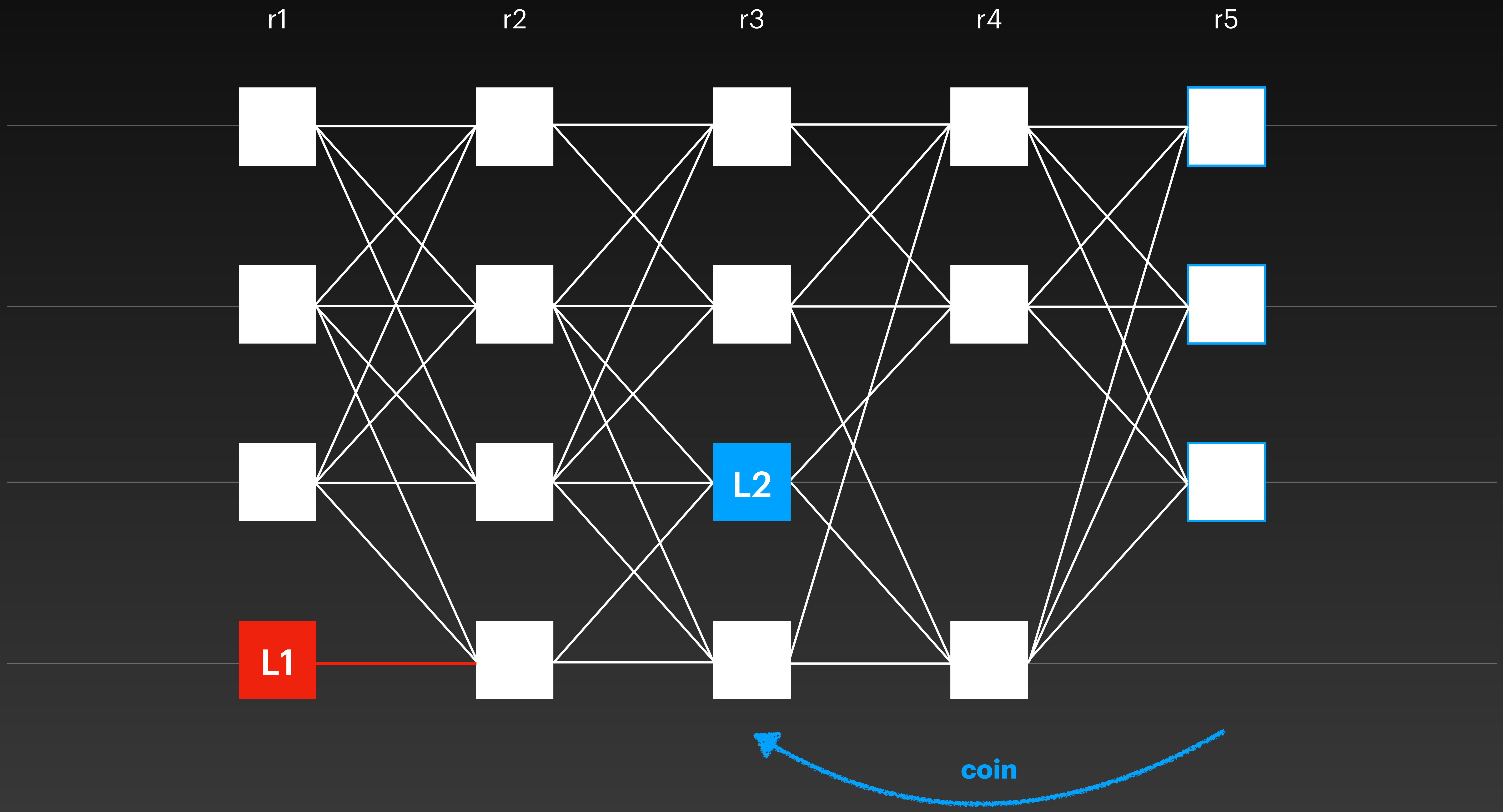
Tusk

Nothing is committed and we keep build the DAG



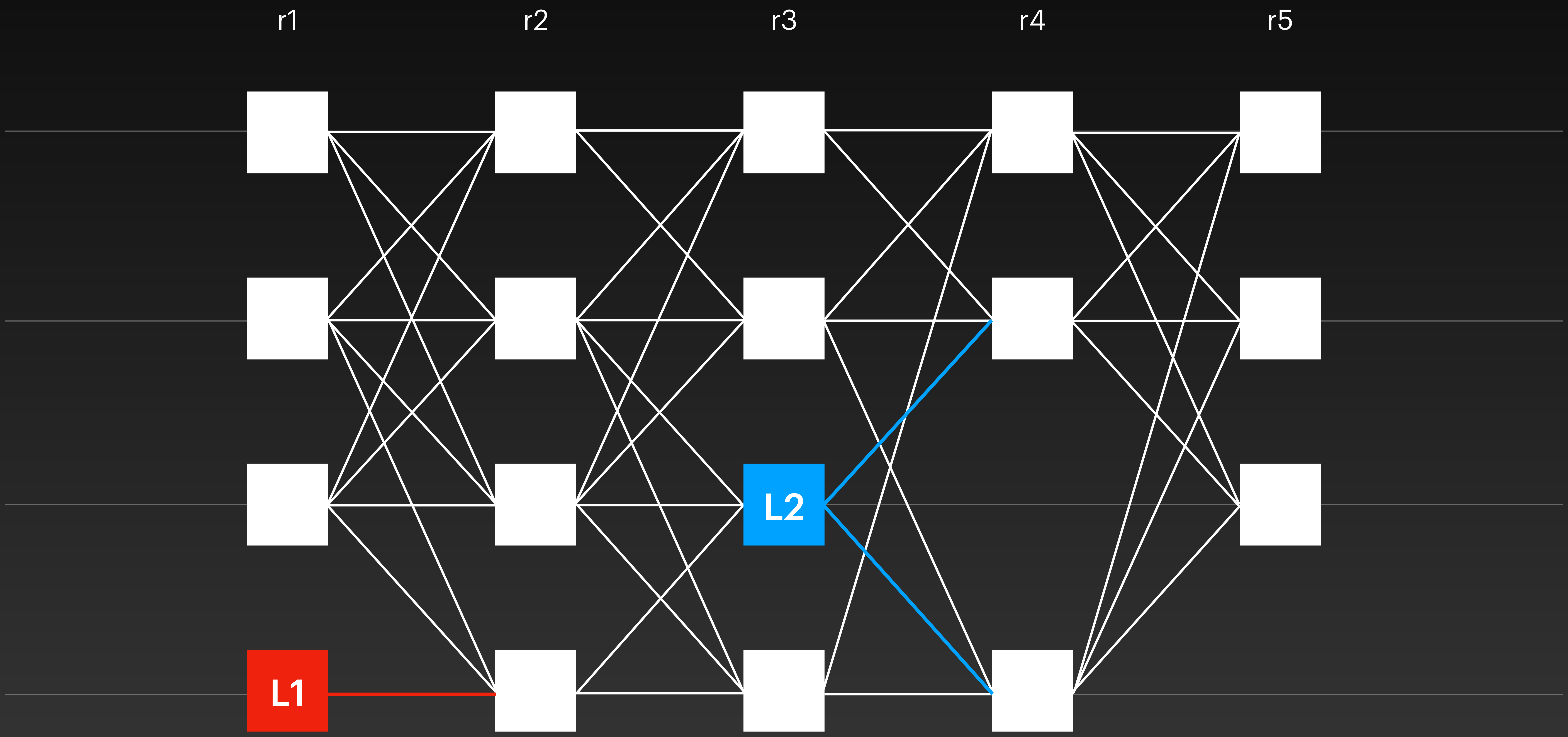
Tusk

Elect the leader of r3



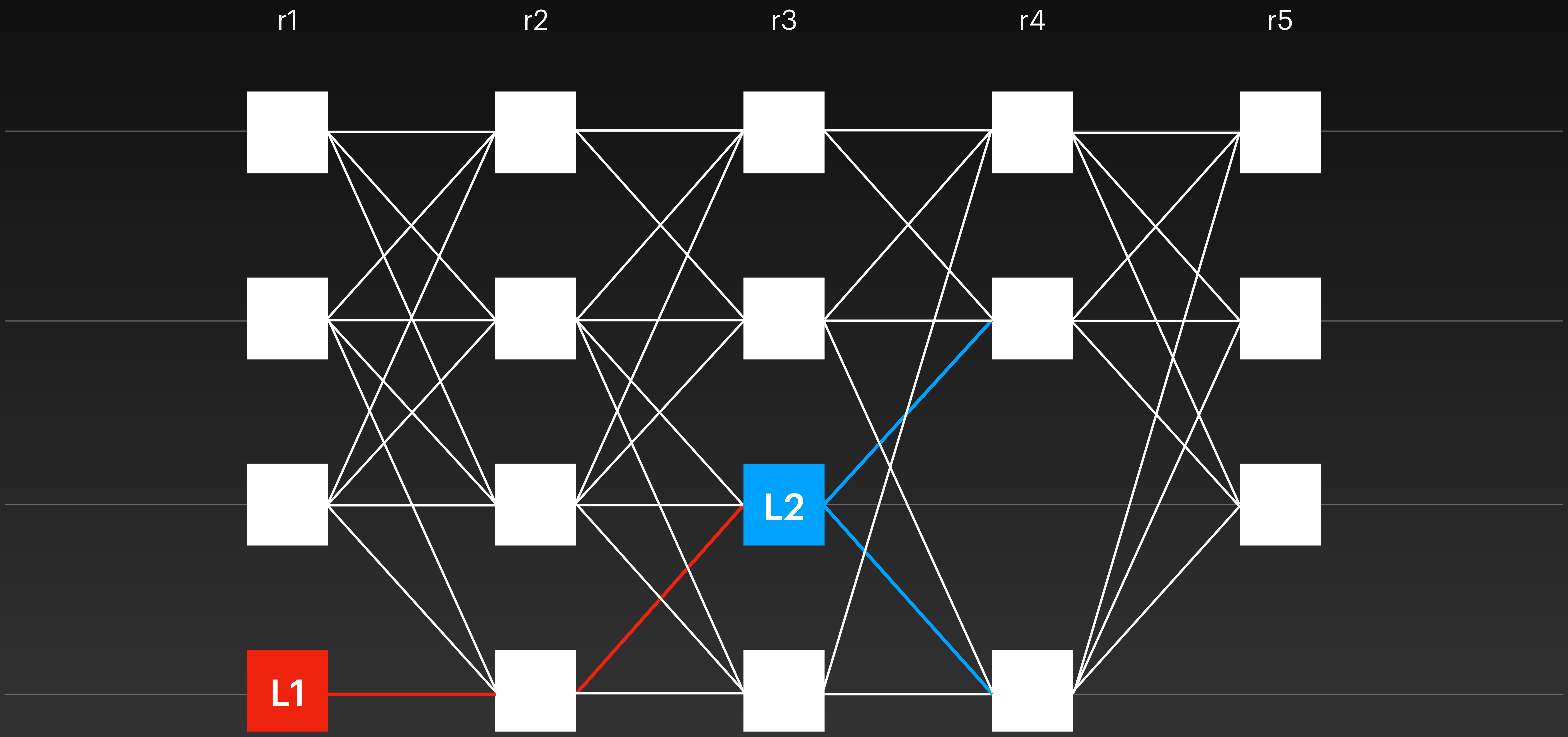
Tusk

Leader L2 has enough support



Tusk

Leader L2 has links to leader L1

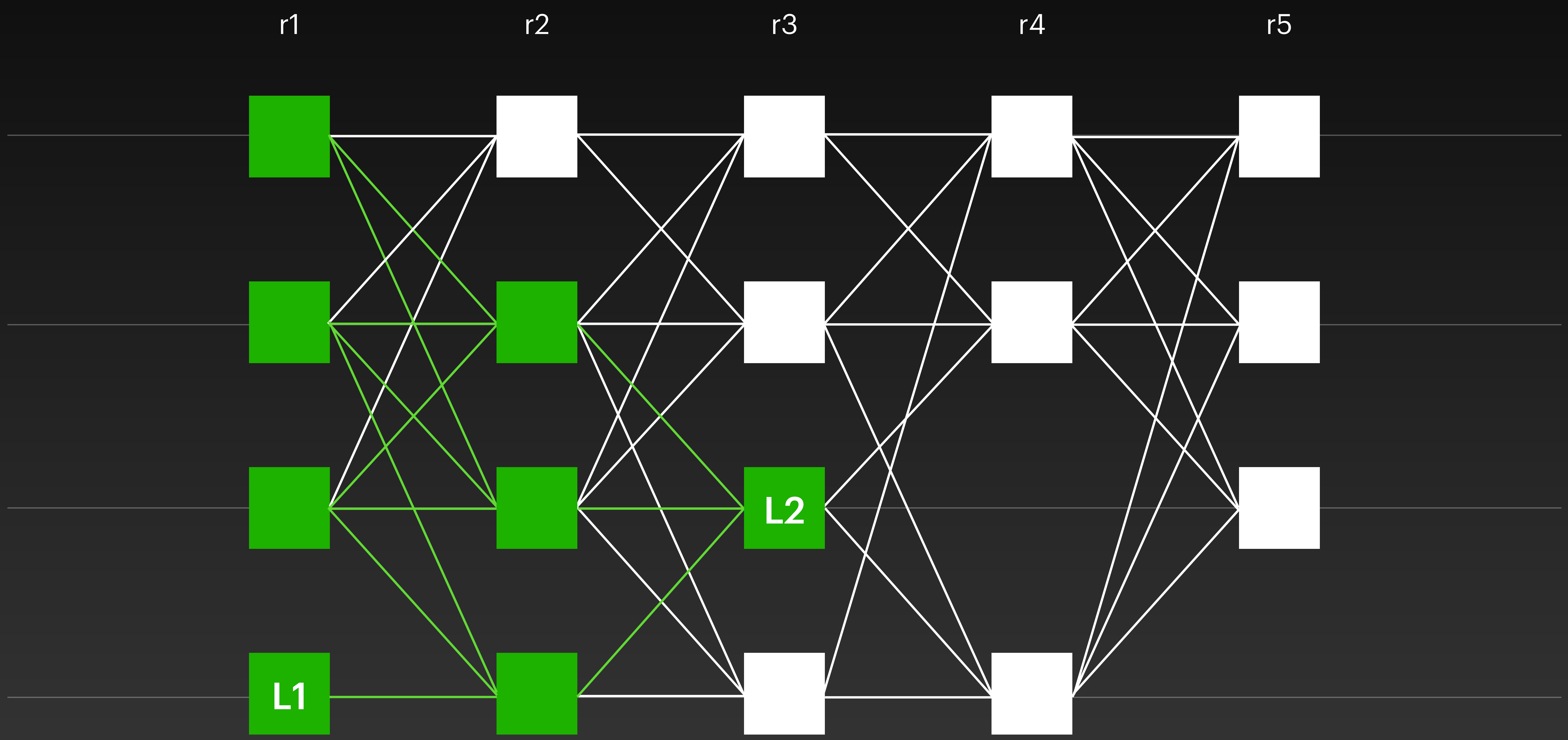


First commit L1

Then commit L2

Tusk

Commit all the sub-DAG of the leader

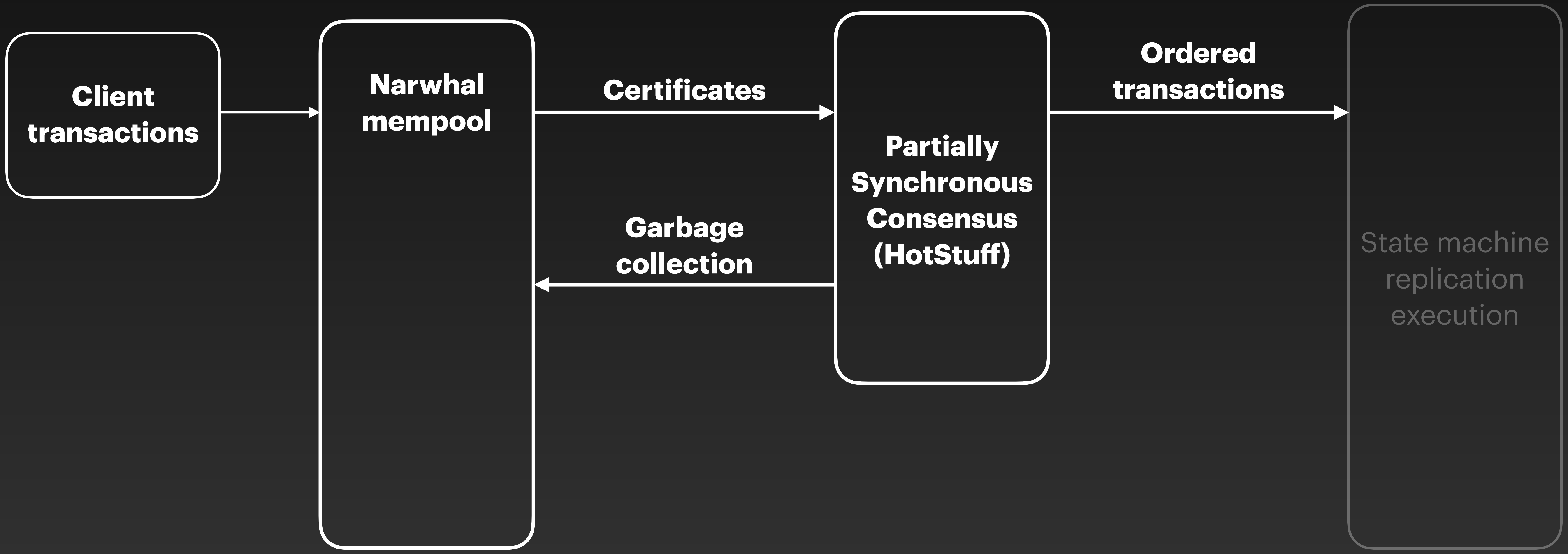


HotStuff on Steroids

Just by replacing the mempool

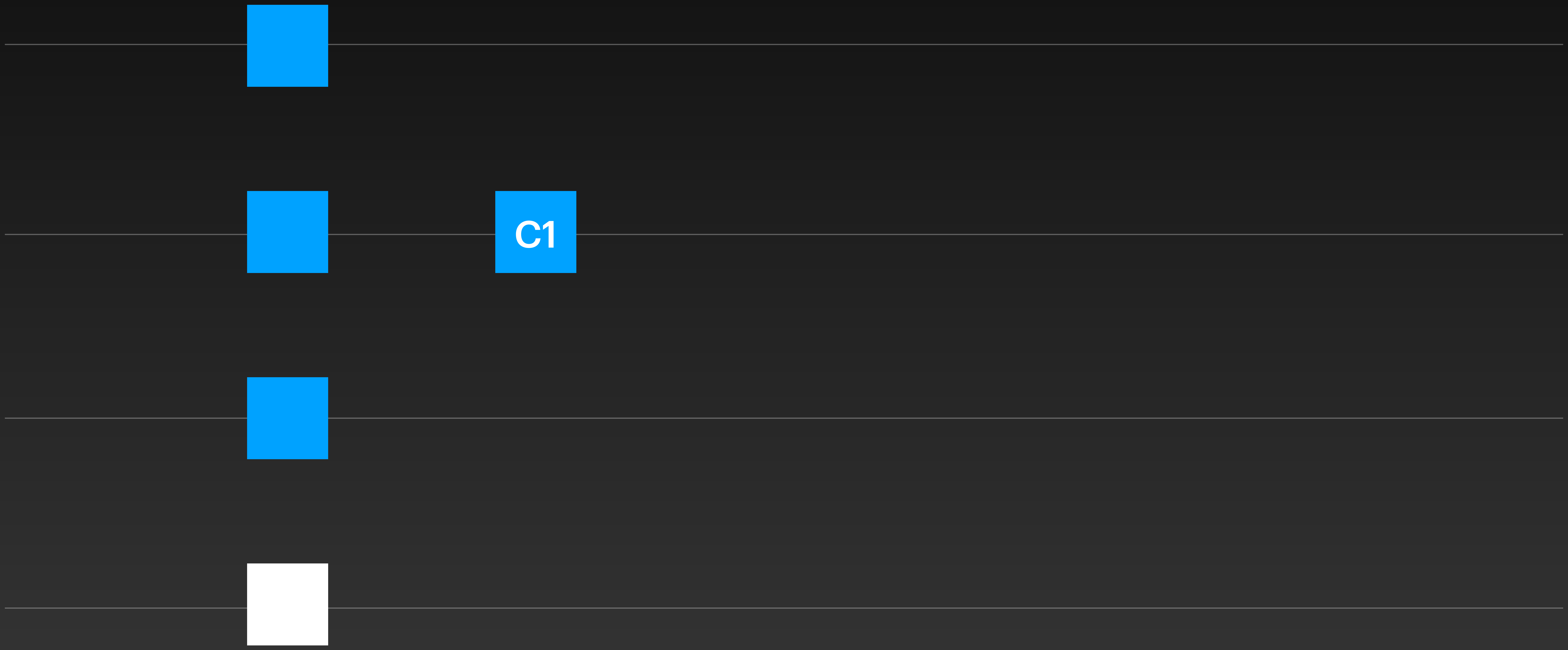
HotStuff on Narwhal

Overview



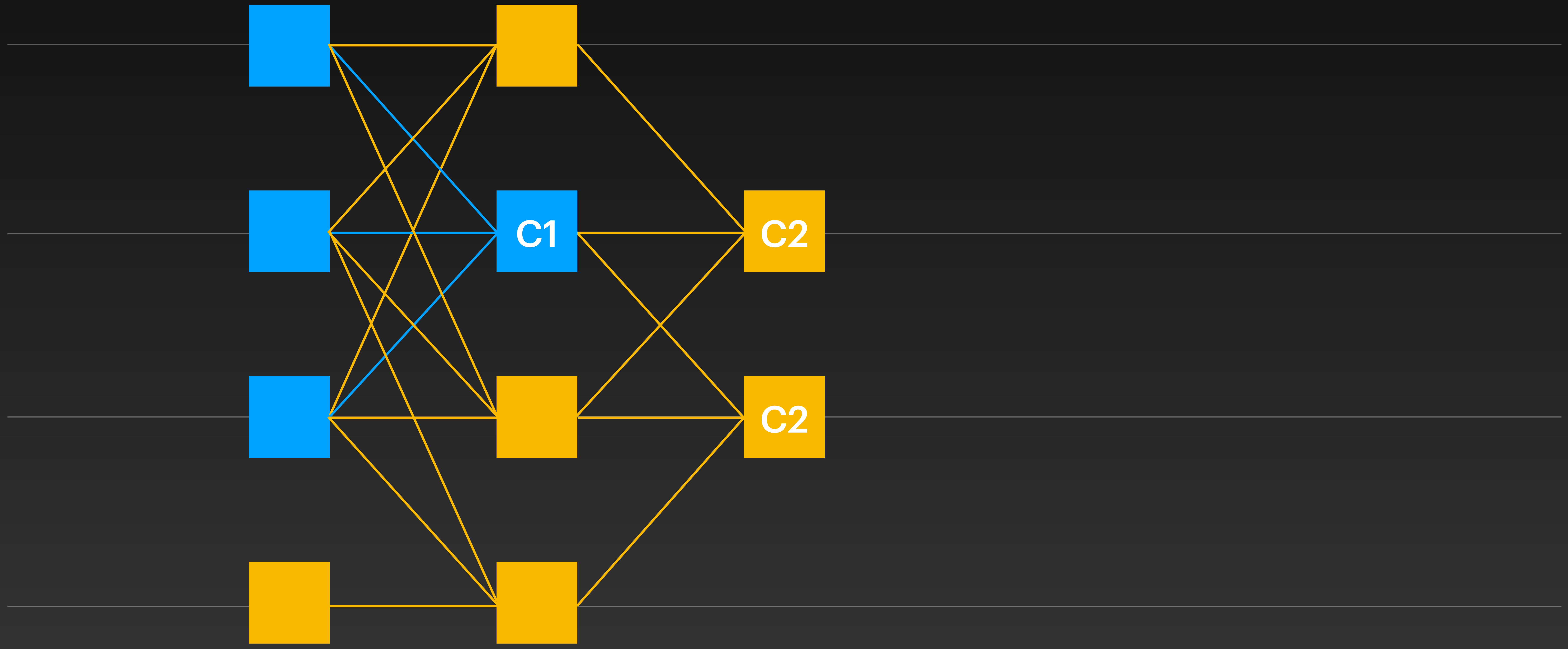
HotStuff on Narwhal

Enhanced commit rule



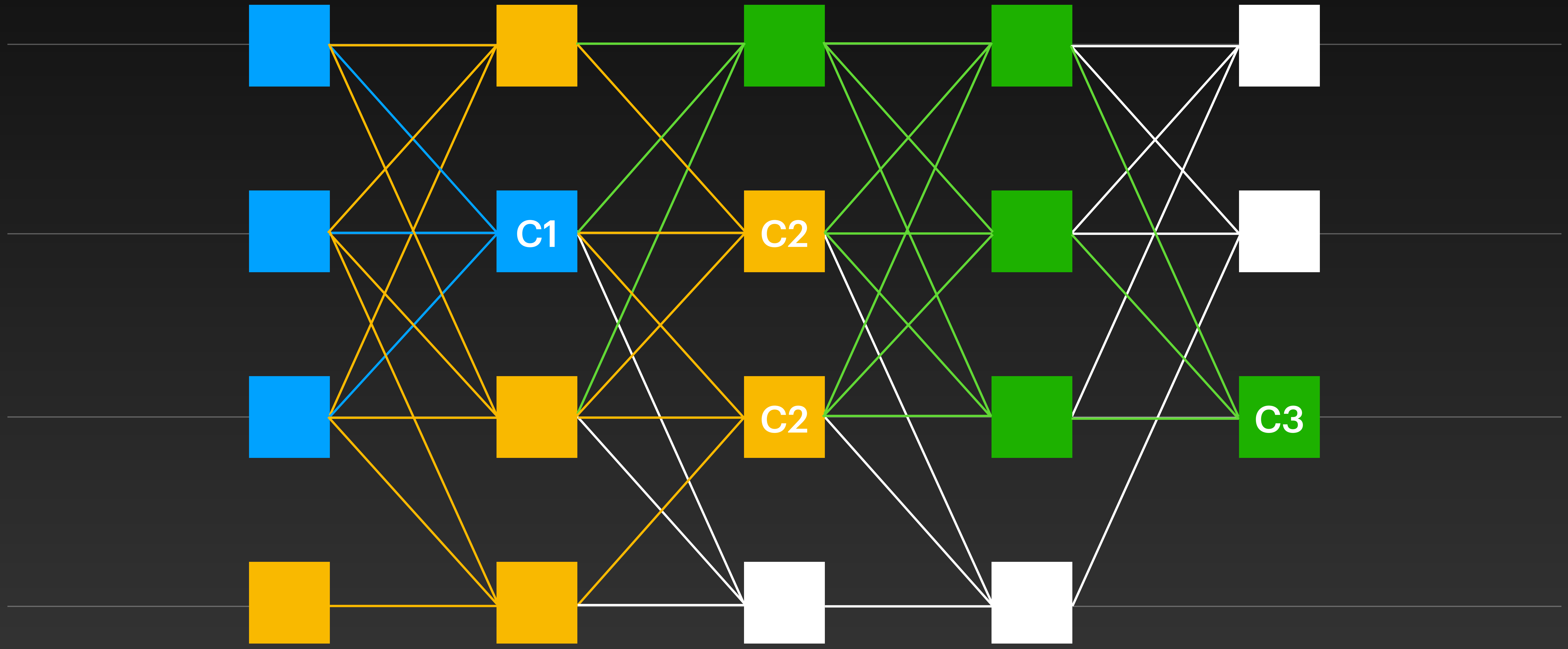
HotStuff on Narwhal

Enhanced commit rule



HotStuff on Narwhal

Enhanced commit rule



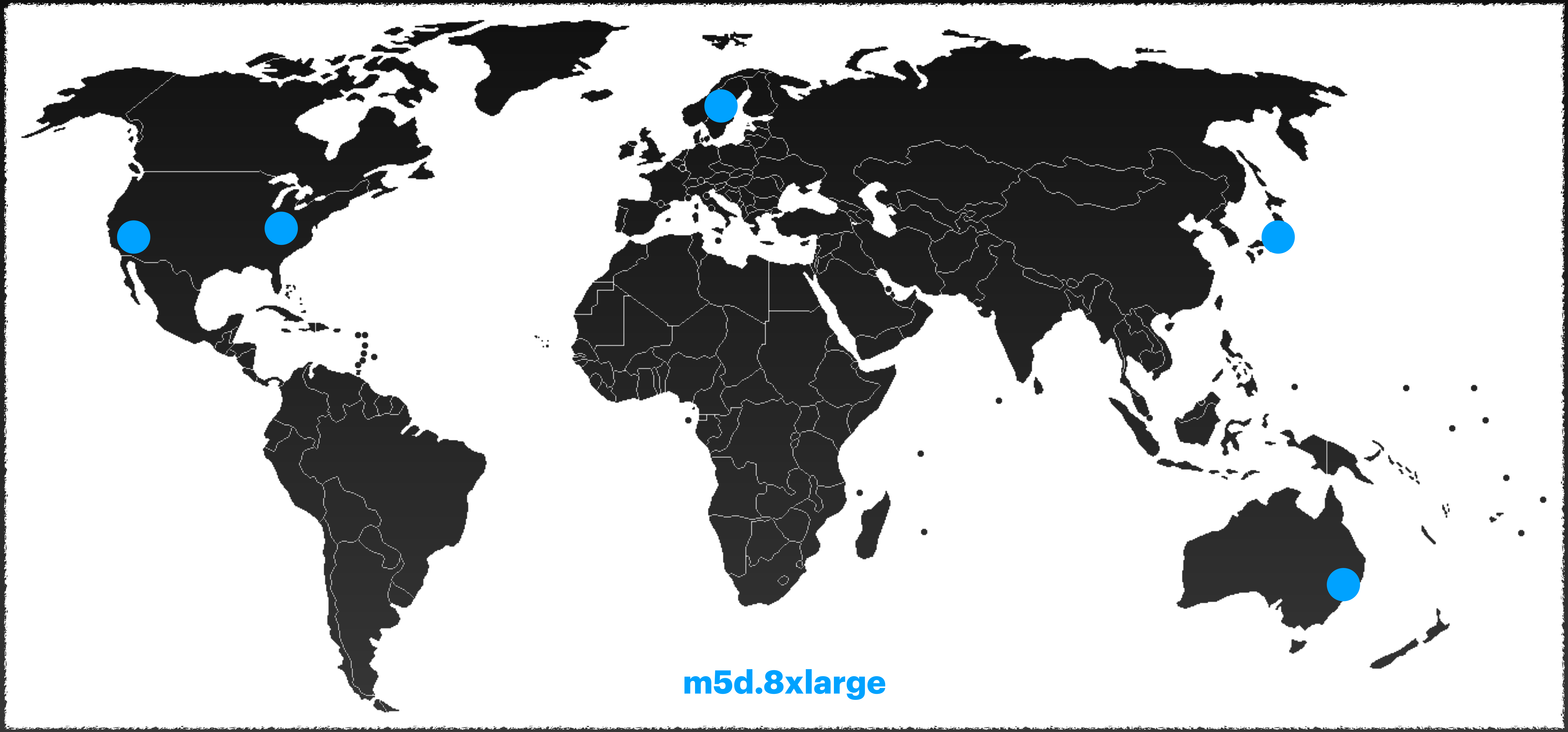
Implementation

- Written in Rust
- Networking: Tokio (TCP)
- Storage: RocksDB
- Cryptography: ed25519-dalek

https://github.com/asonnino/narwhal

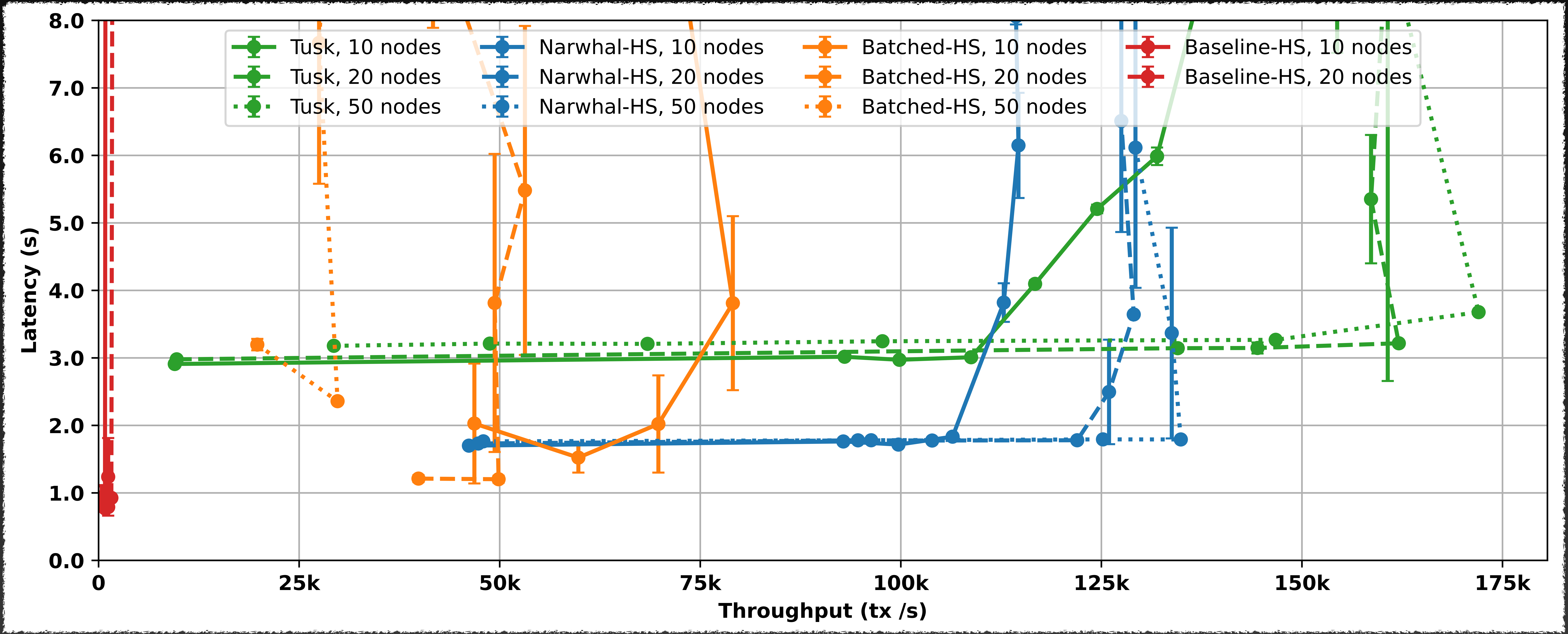
Evaluation

Experimental setup on AWS



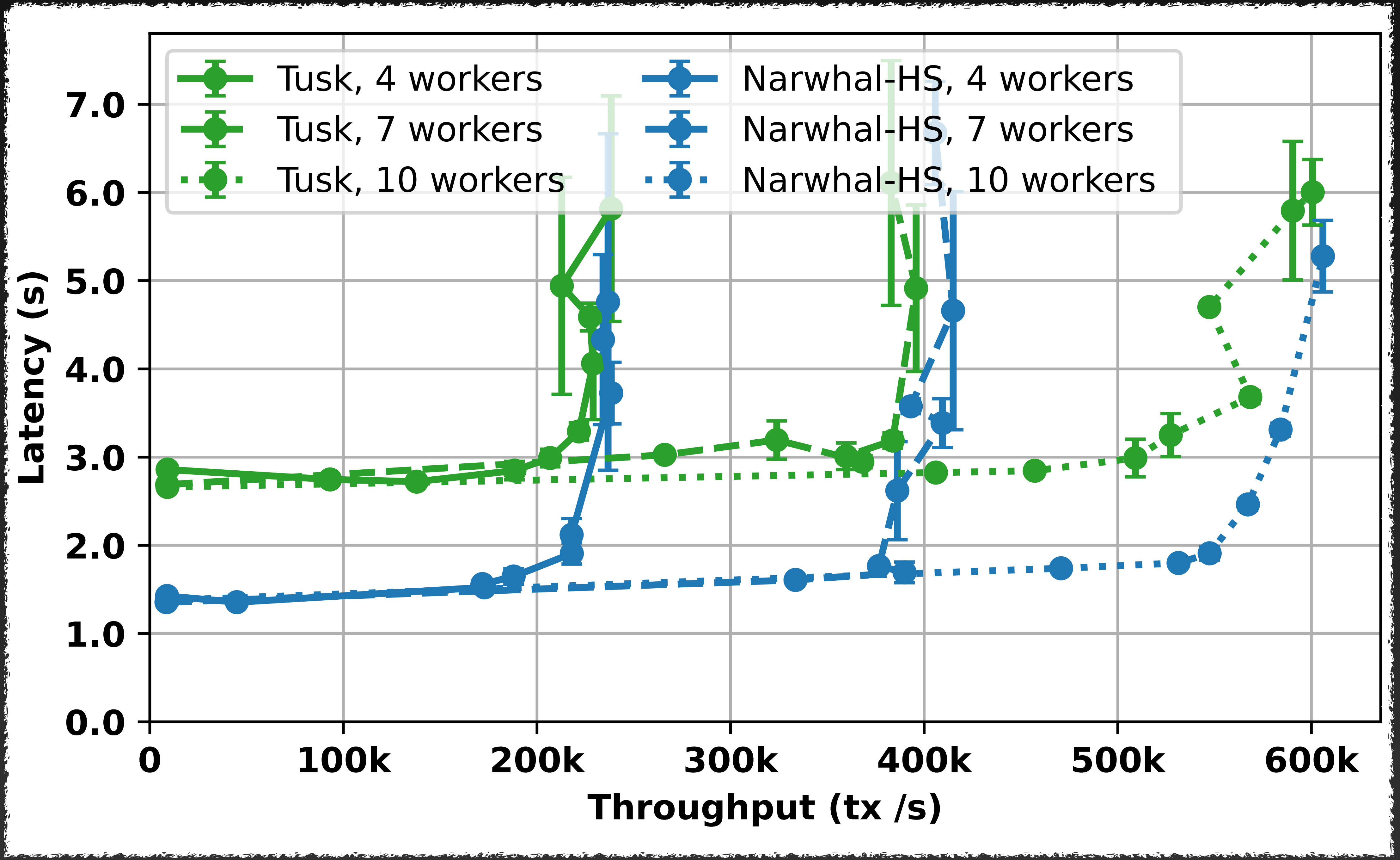
Evaluation

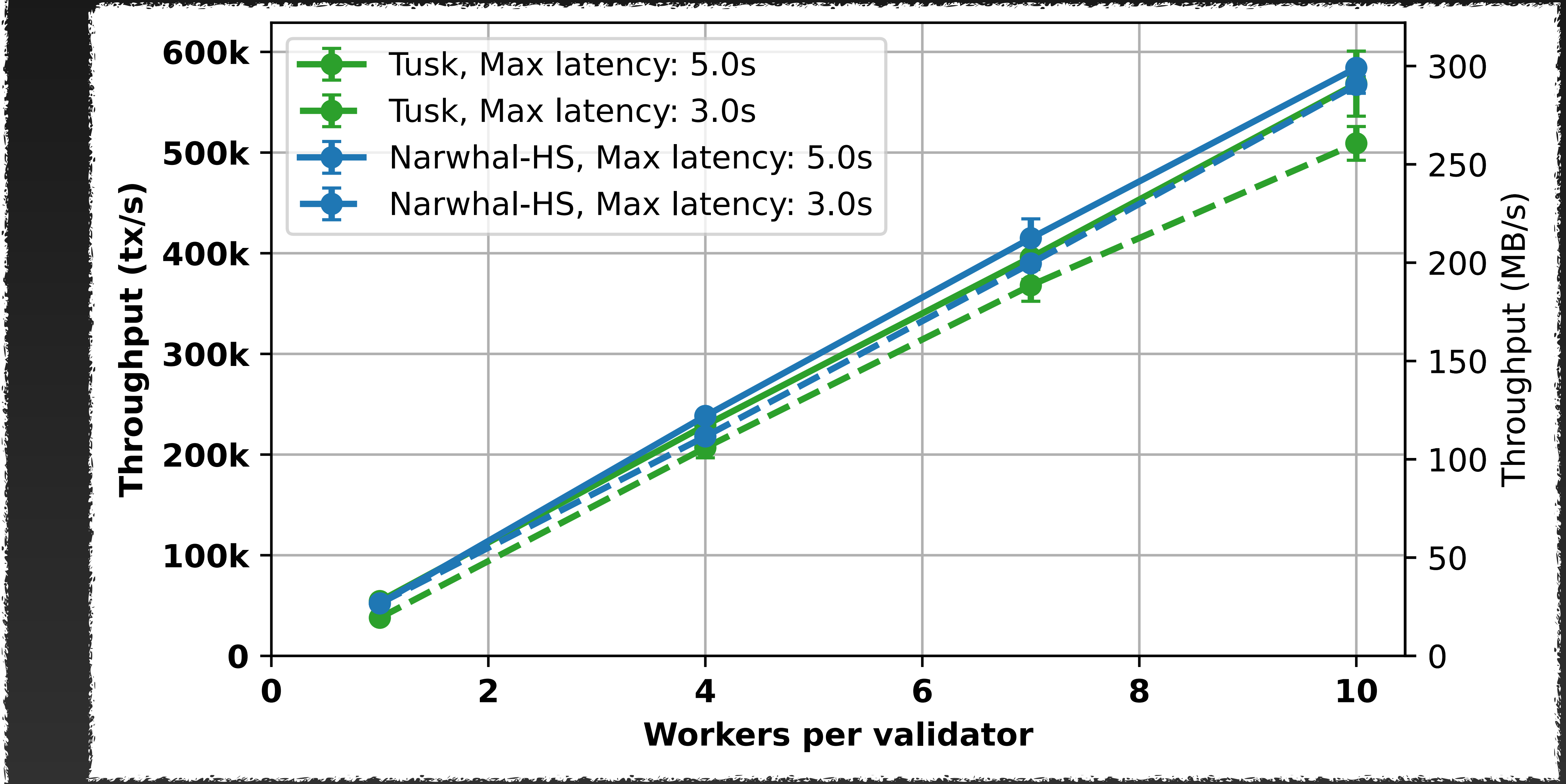
Throughput latency graph



Evaluation

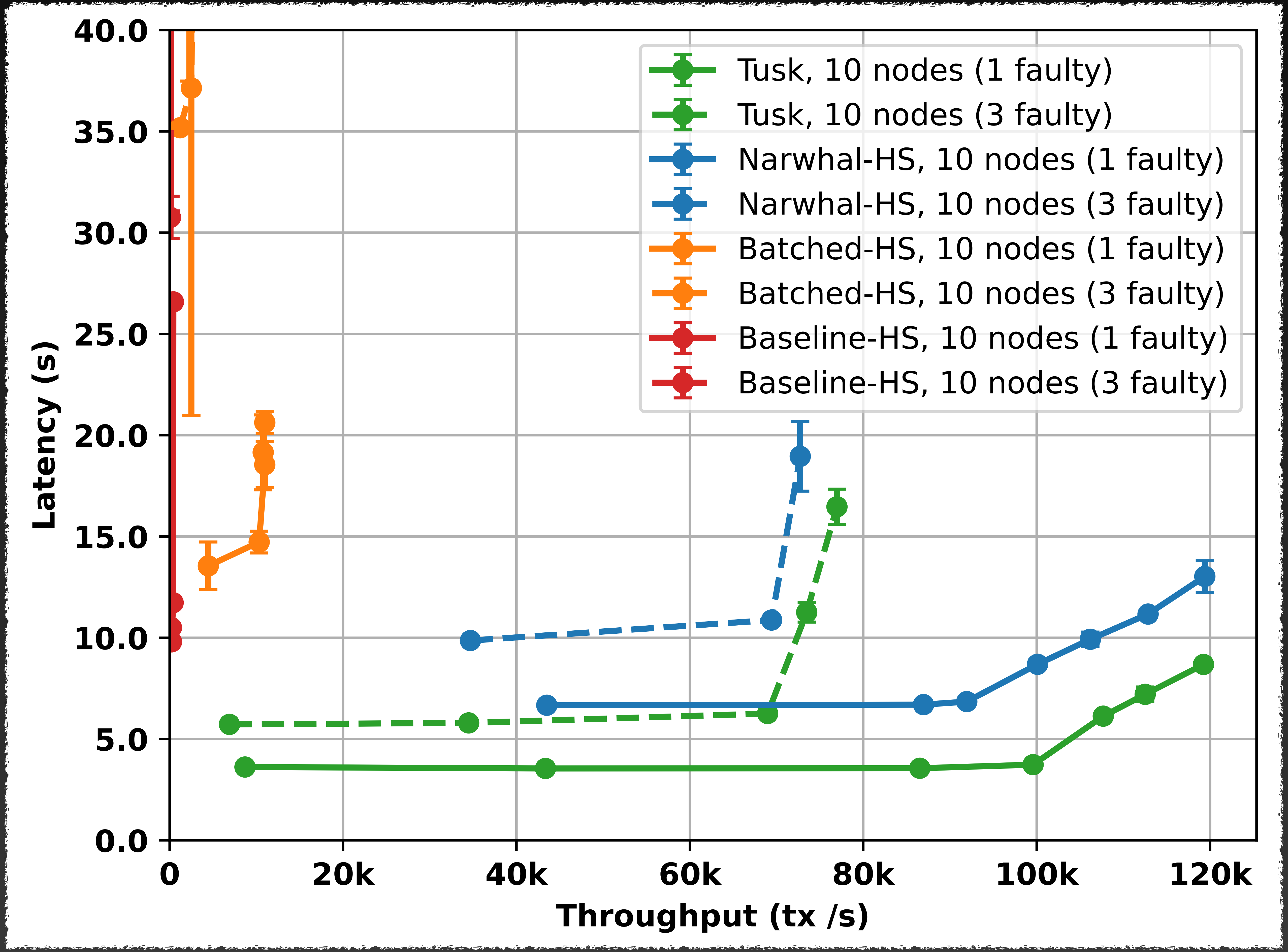
Scalability





Evaluation

Performance under faults



Conclusion

Narwhal & Tusk

- Separate consensus and data dissemination for high performance
- Scalable design, egalitarian resource utilizations

- **Paper:** https://arxiv.org/pdf/2105.11827.pdf
- **Code:** https://github.com/facebookresearch/narwhal

Future Works

Interested?

- Performance under DDoS attack?
- Can we embed a partially synchronous consensus into the DAG?
- How to implement scalable execution?

alberto.sonnino@ucl.ac.uk

Alberto Sonnino