Bullshark

DAG BFT Protocols Made Practical

Alberto Sonnino

Acknowledgements

Alexander Spiegelman

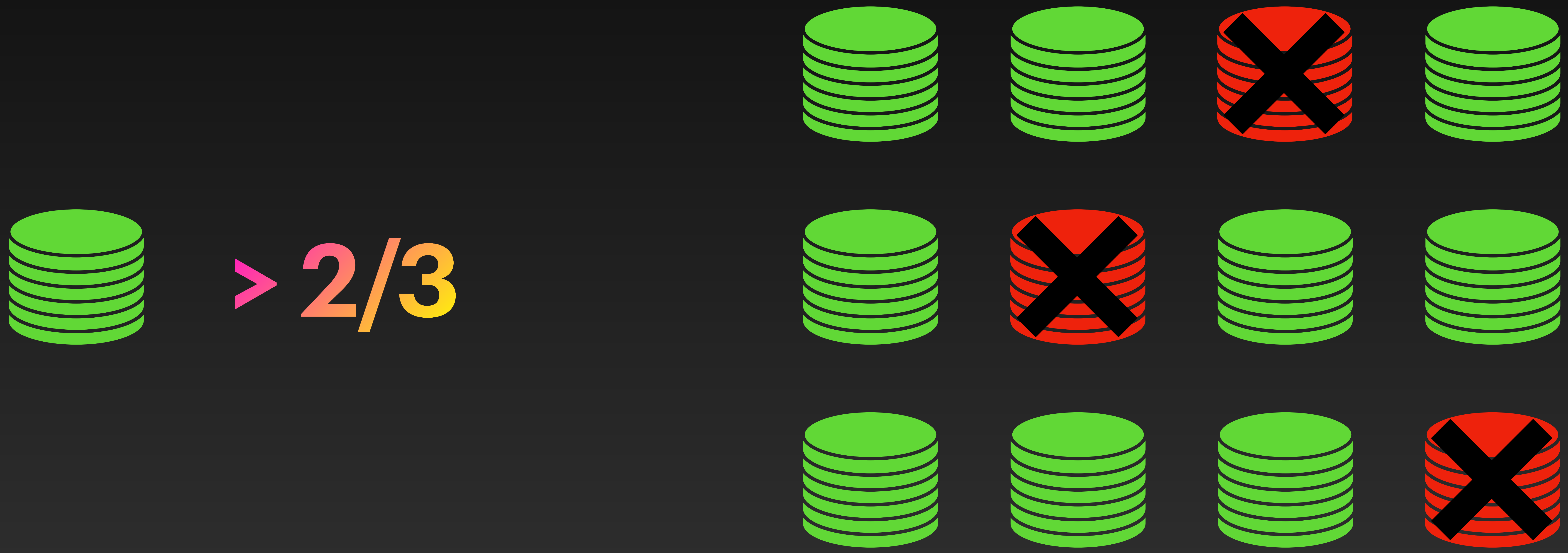
Neil Giridharan

Alberto Sonnino

Lefteris Kokoris-Kogias

Work done at Facebook Novi

Byzantine Fault Tolerance



> 2/3

Consensus on top of Narwhal

Goal of this project

Simple

- Zero-message overhead
- No view-change
- No common-coin

Performant

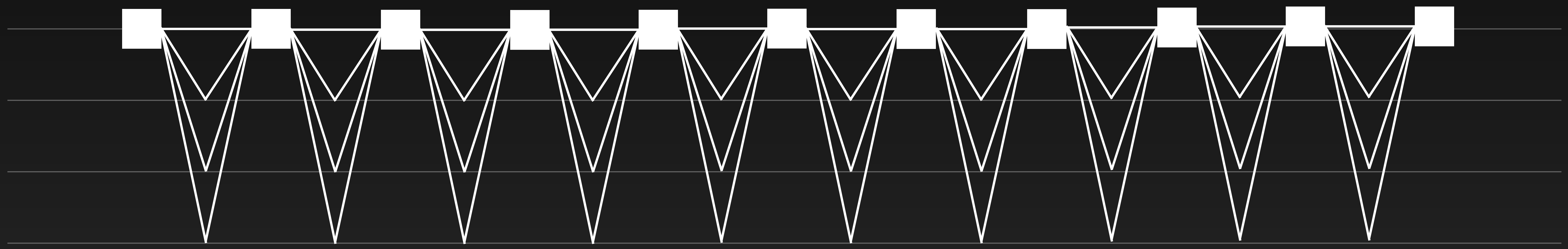
- Take advantage of Narwhal
- Exploit periods of synchrony

Current Designs

- Monolithic protocol sharing transaction data as part of the consensus
- Optimize overall message complexity of the consensus protocol
- Complex & Error-prone view-change protocol

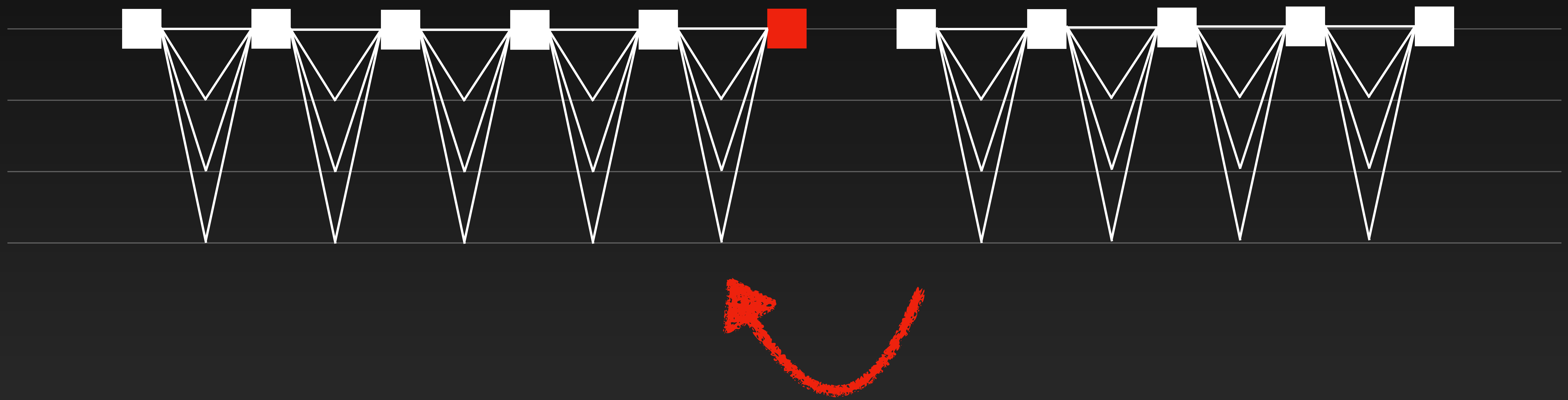
Current Designs

Typical leader-based protocols



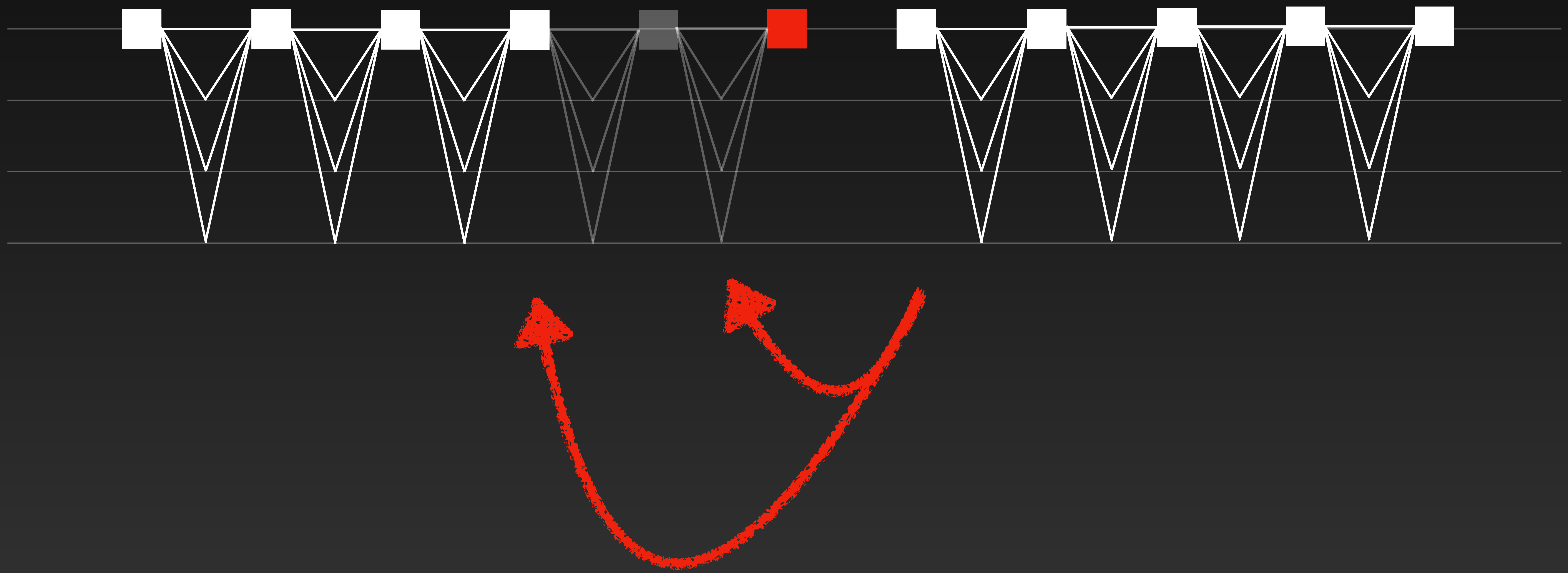
Current Designs

Typical leader-based protocols



Current Designs

Typical leader-based protocols



Narwhal

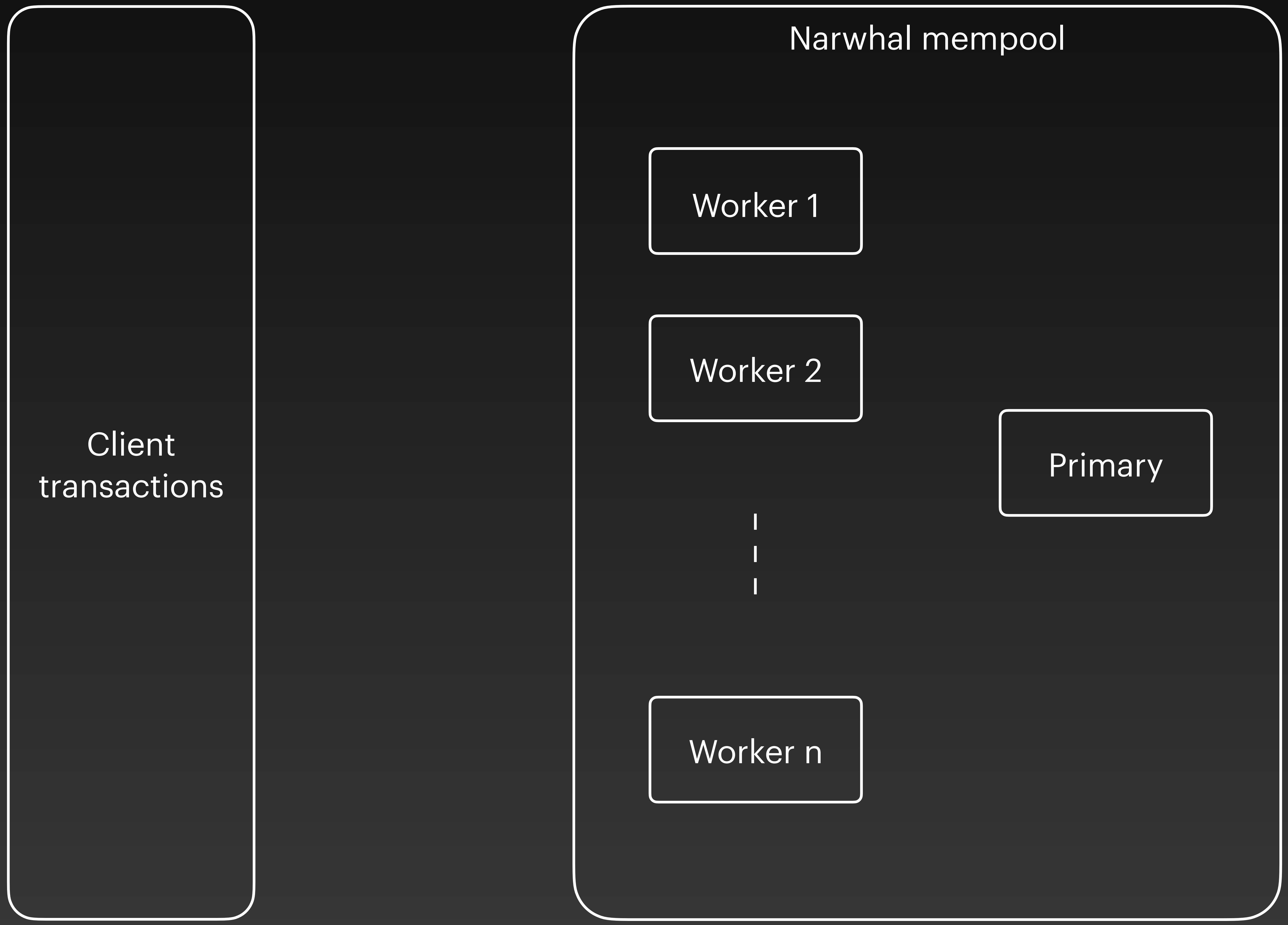
Dag-based mempool

The mempool is the key

Reaching consensus on metadata is cheap

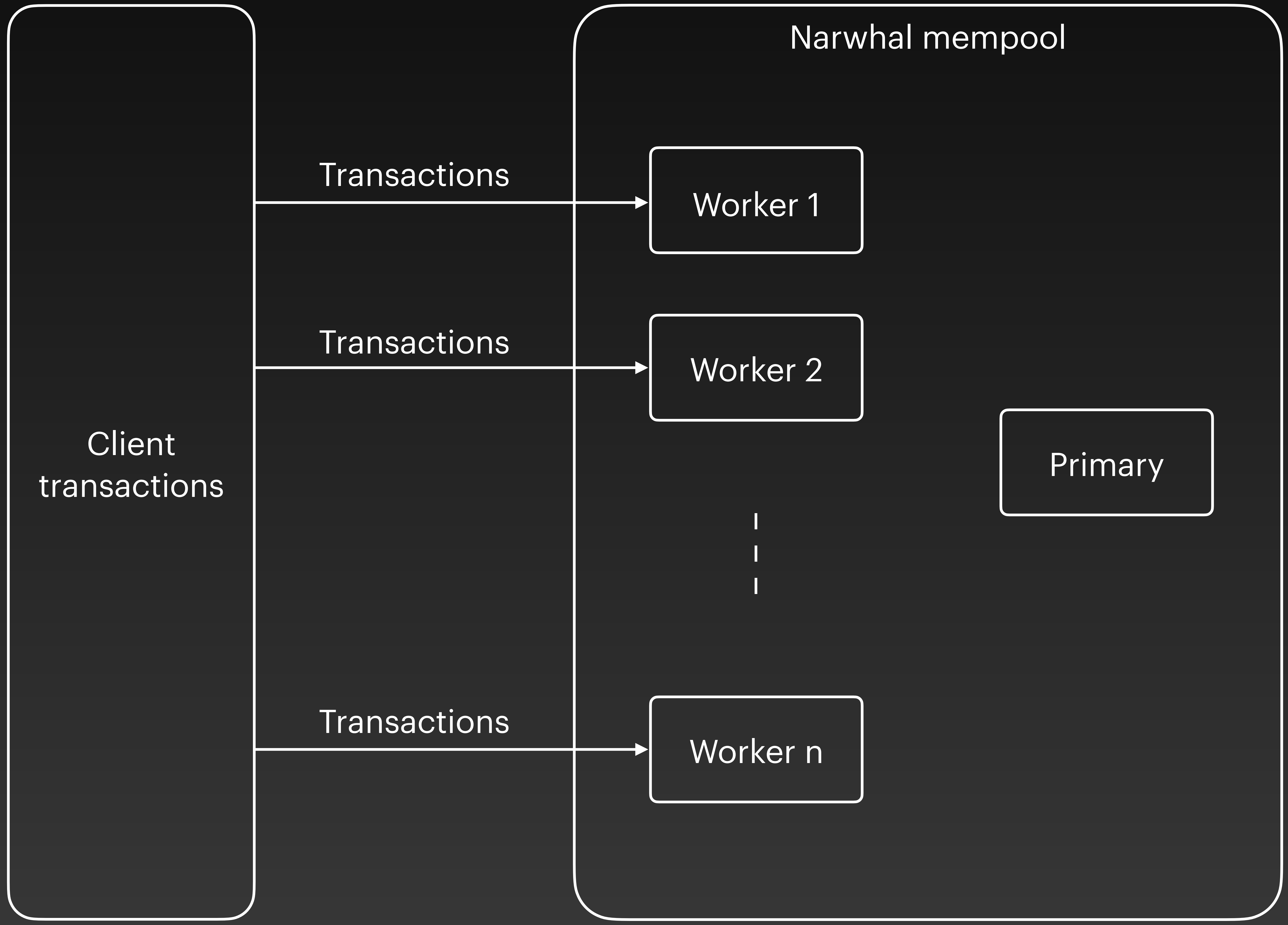
Narwhal

The workers and the primary



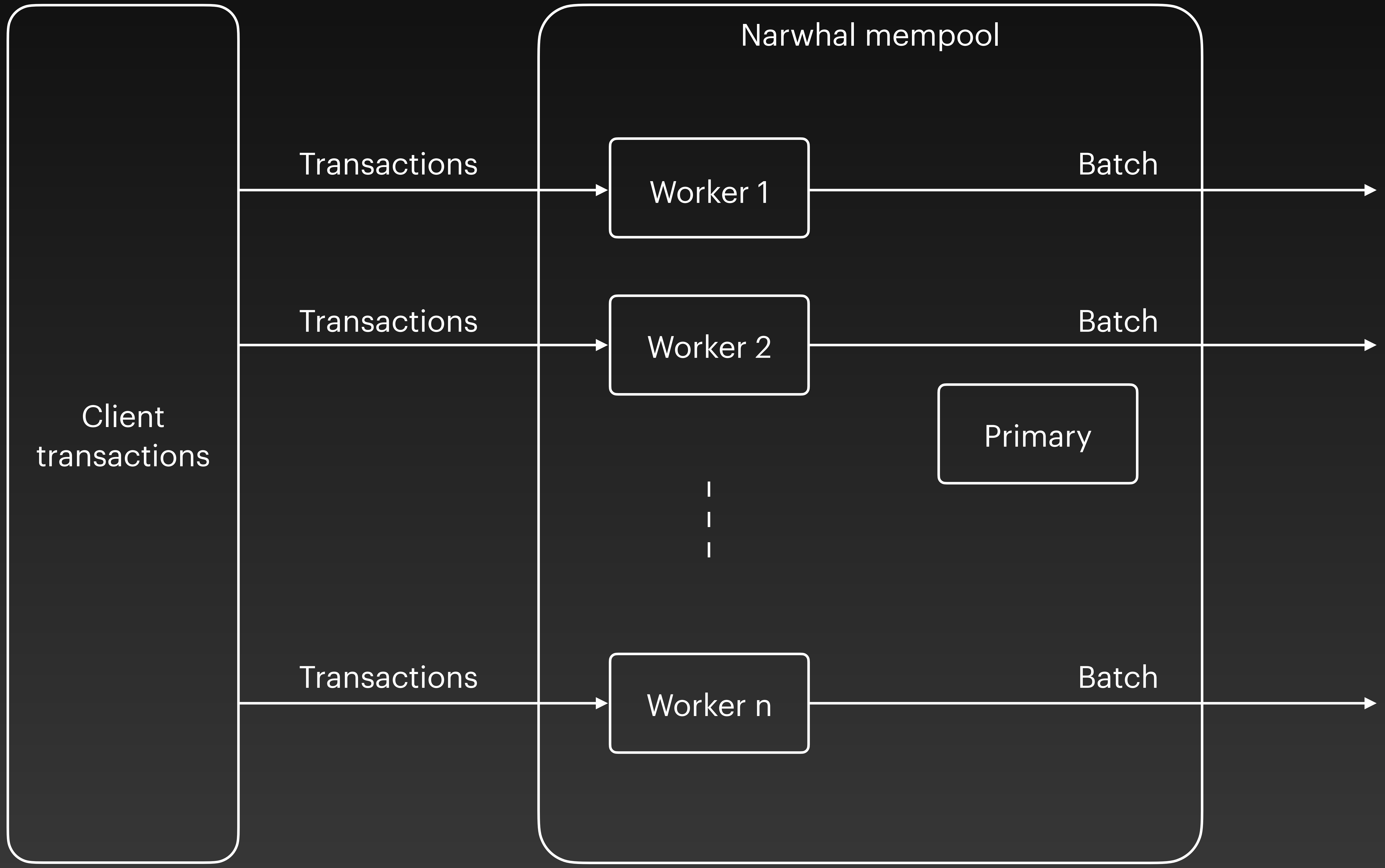
Narwhal

The workers and the primary



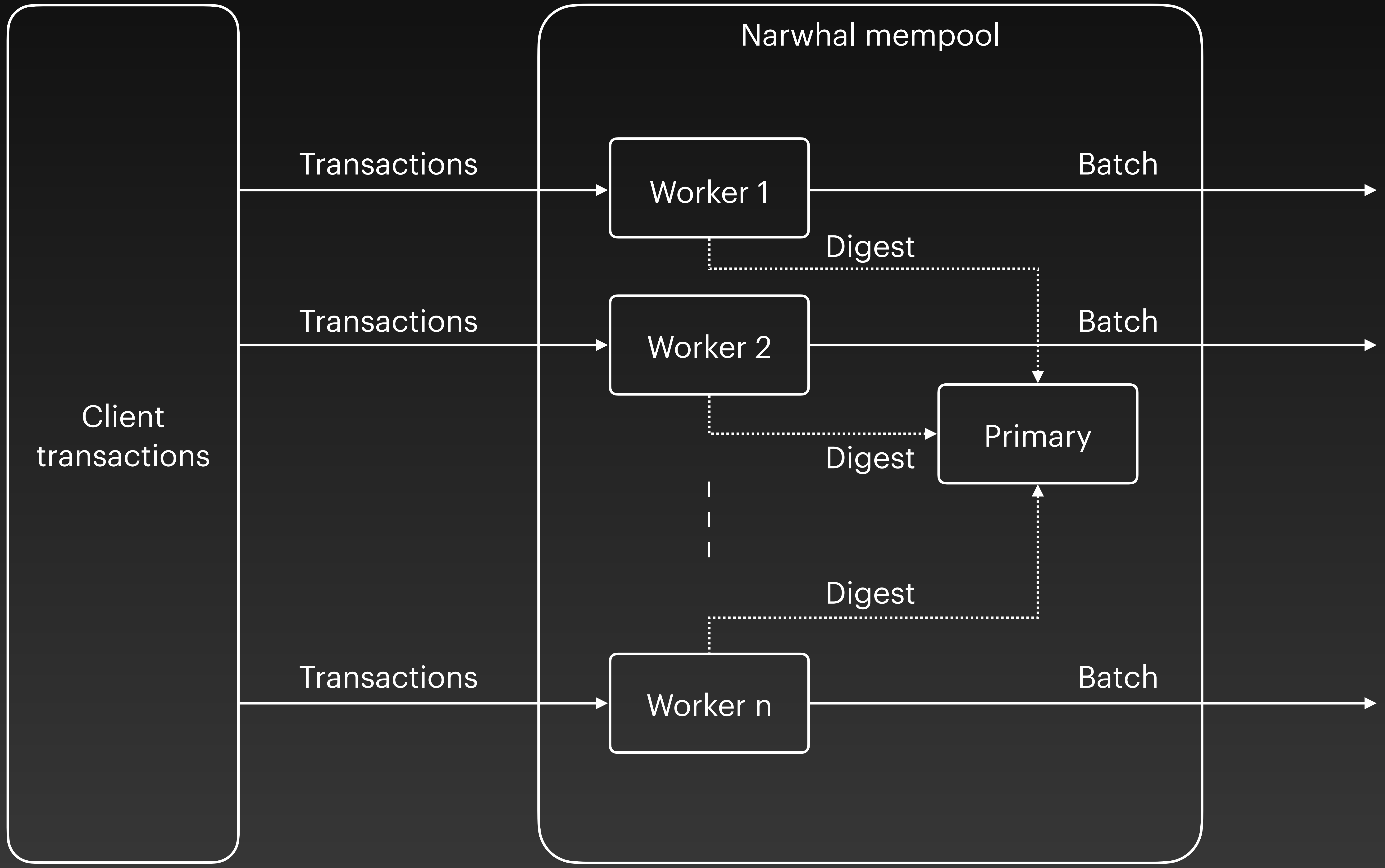
Narwhal

The workers and the primary



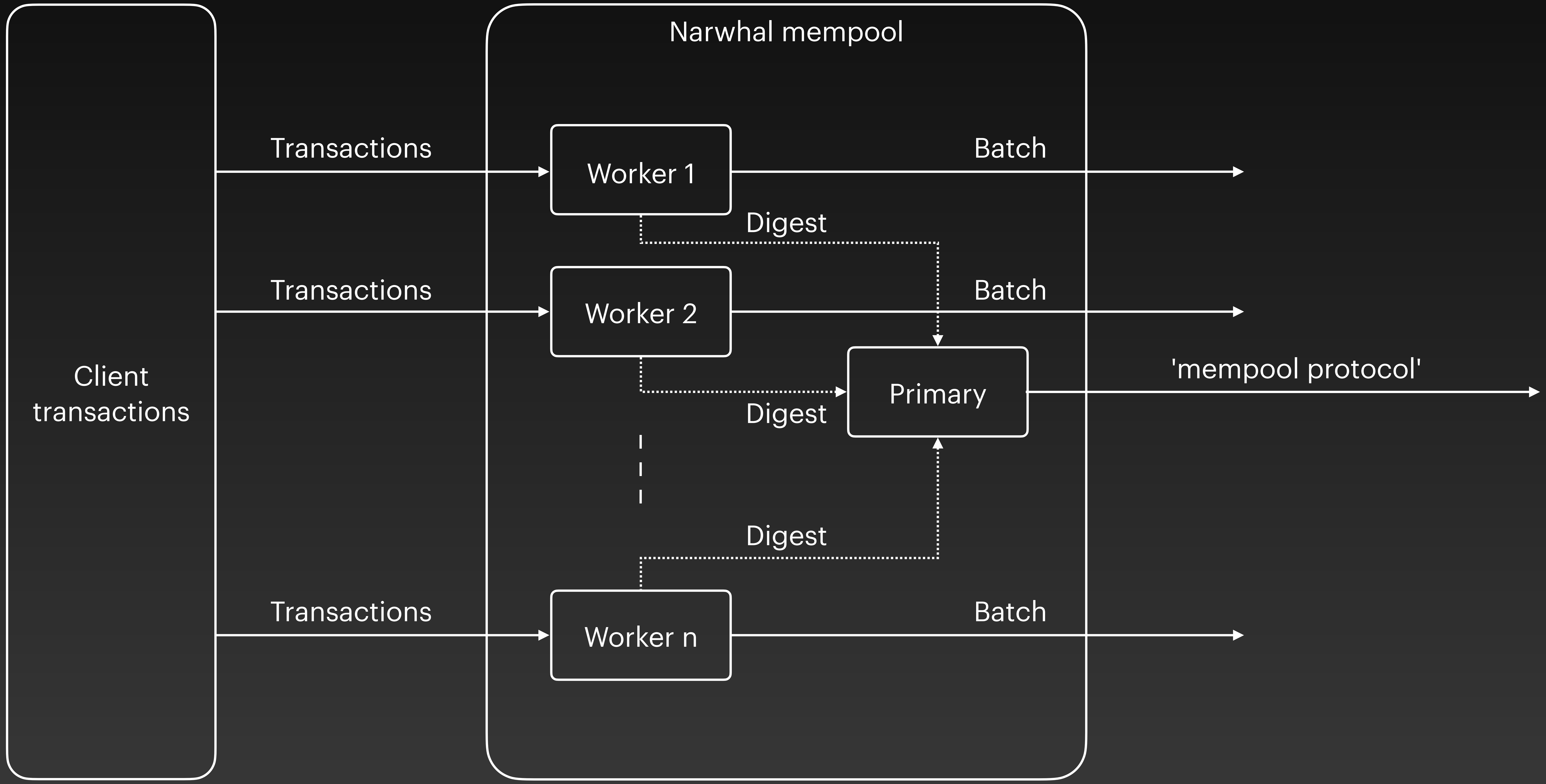
Narwhal

The workers and the primary



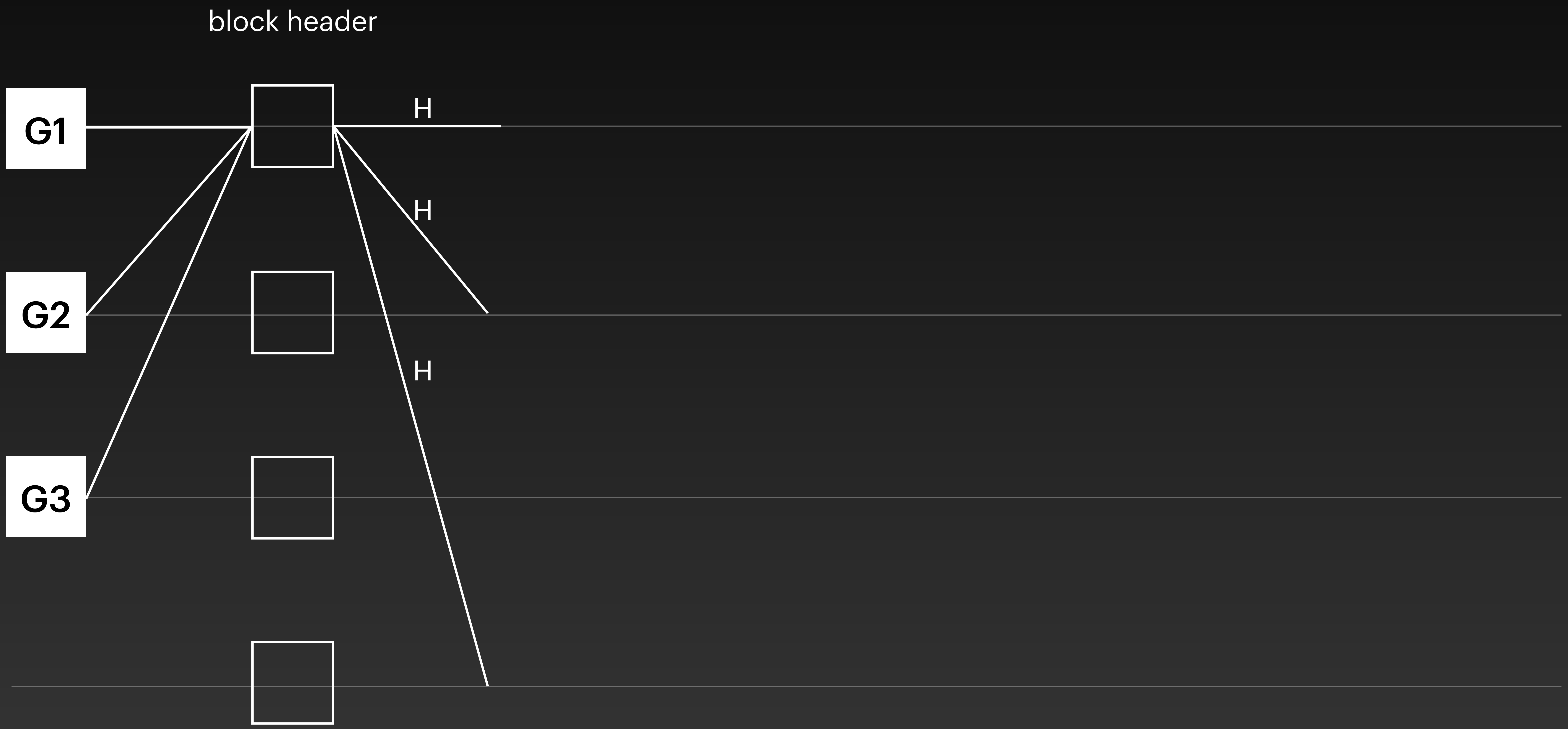
Narwhal

The workers and the primary



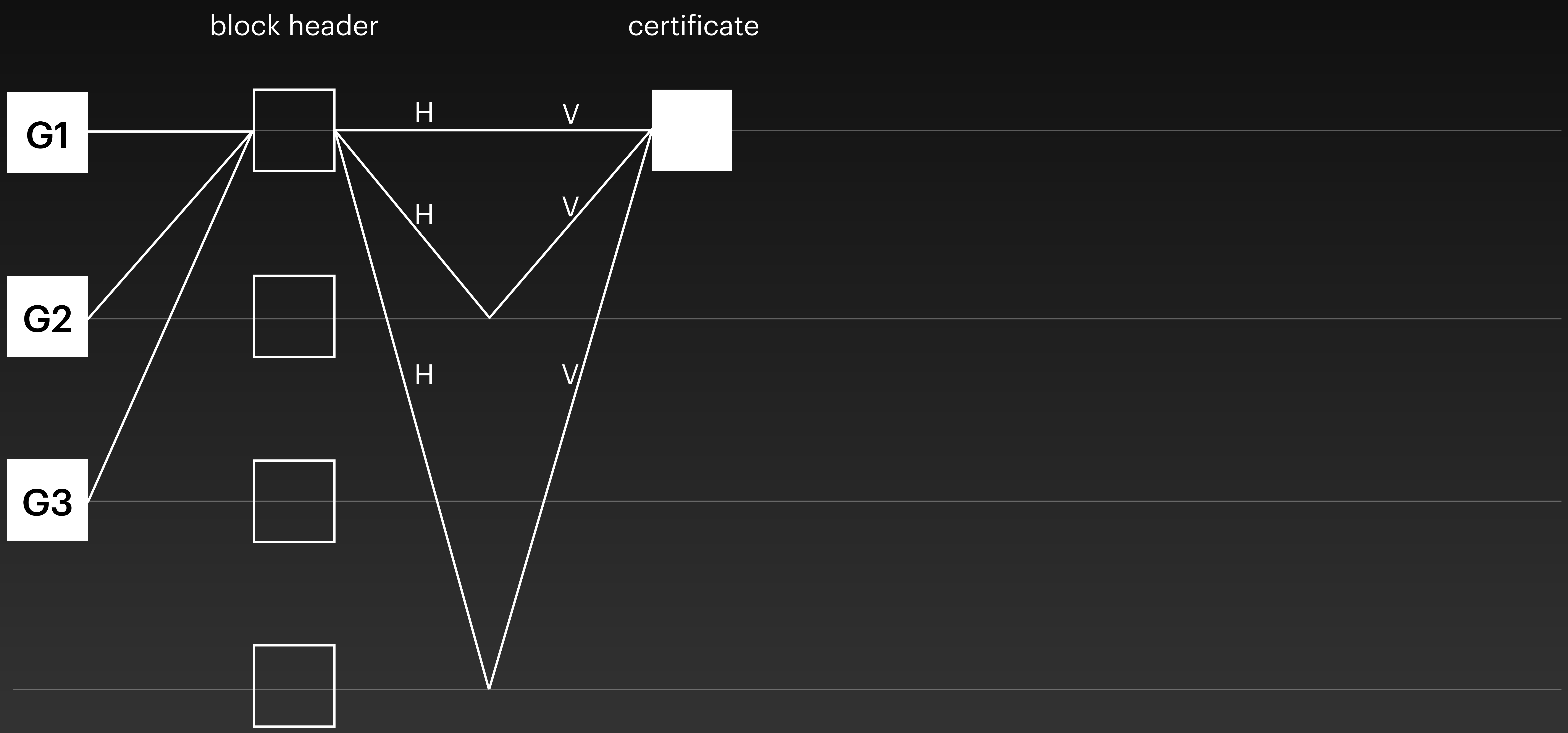
Narwhal

The primary machine



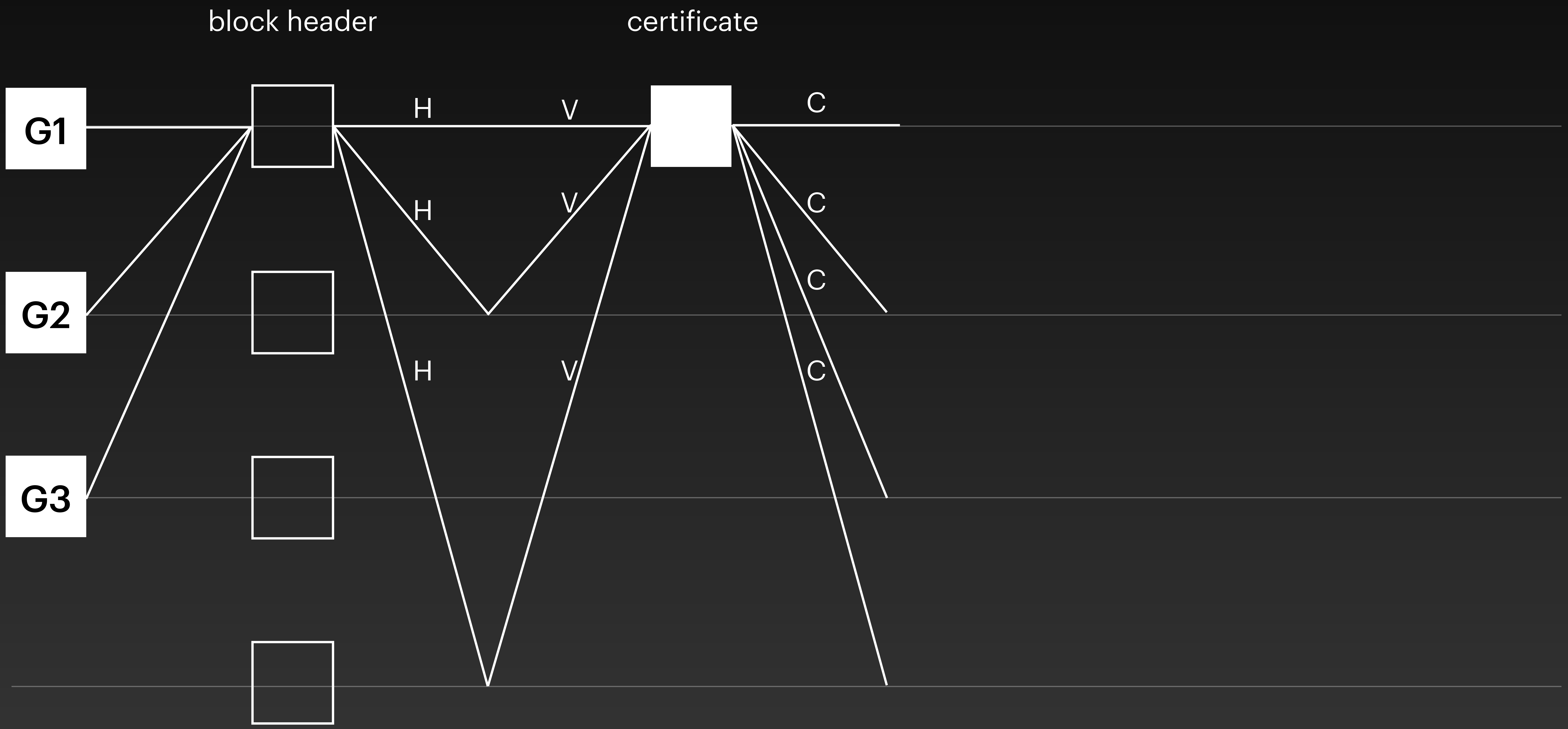
Narwhal

The primary machine



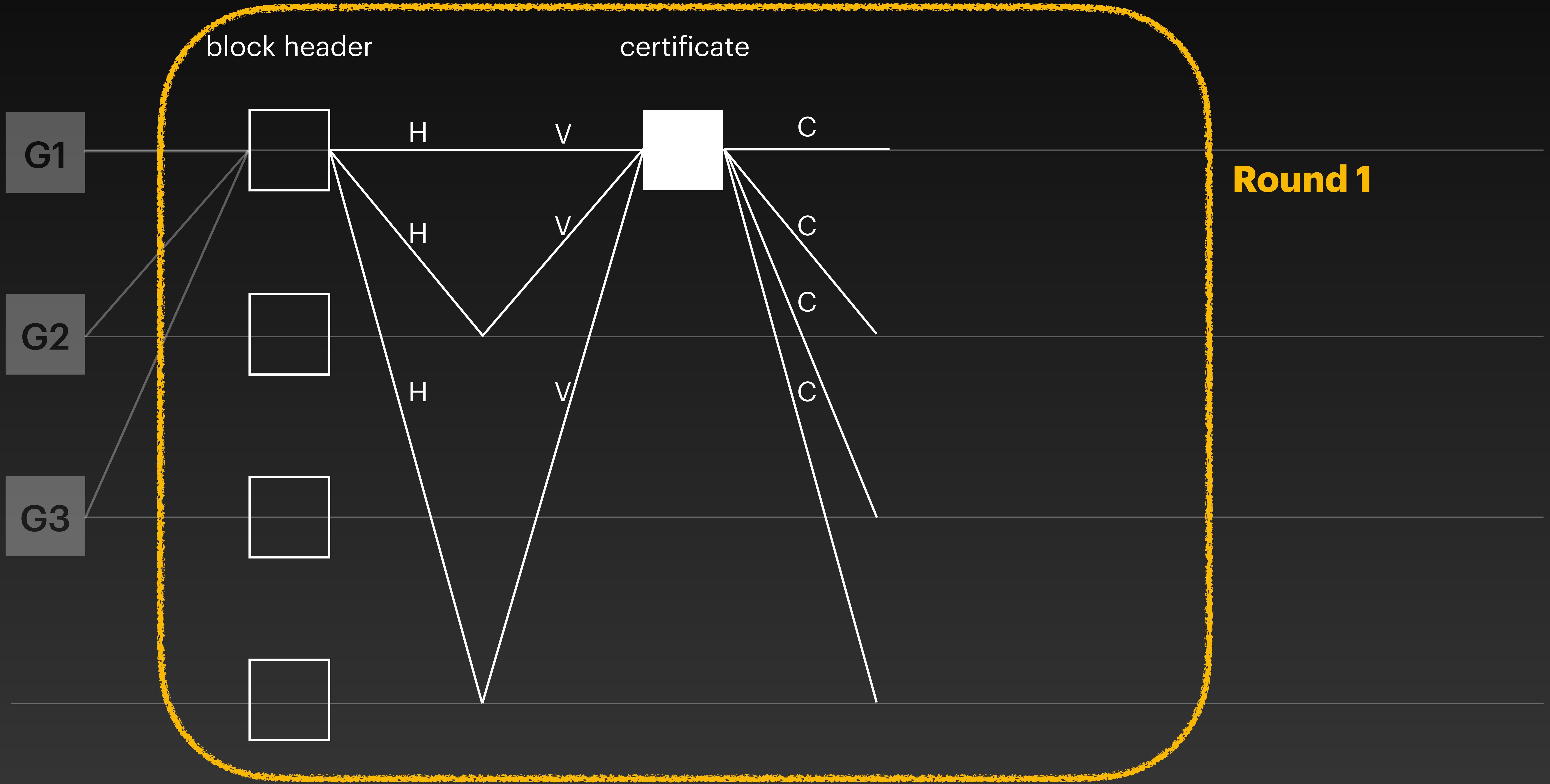
Narwhal

The primary machine



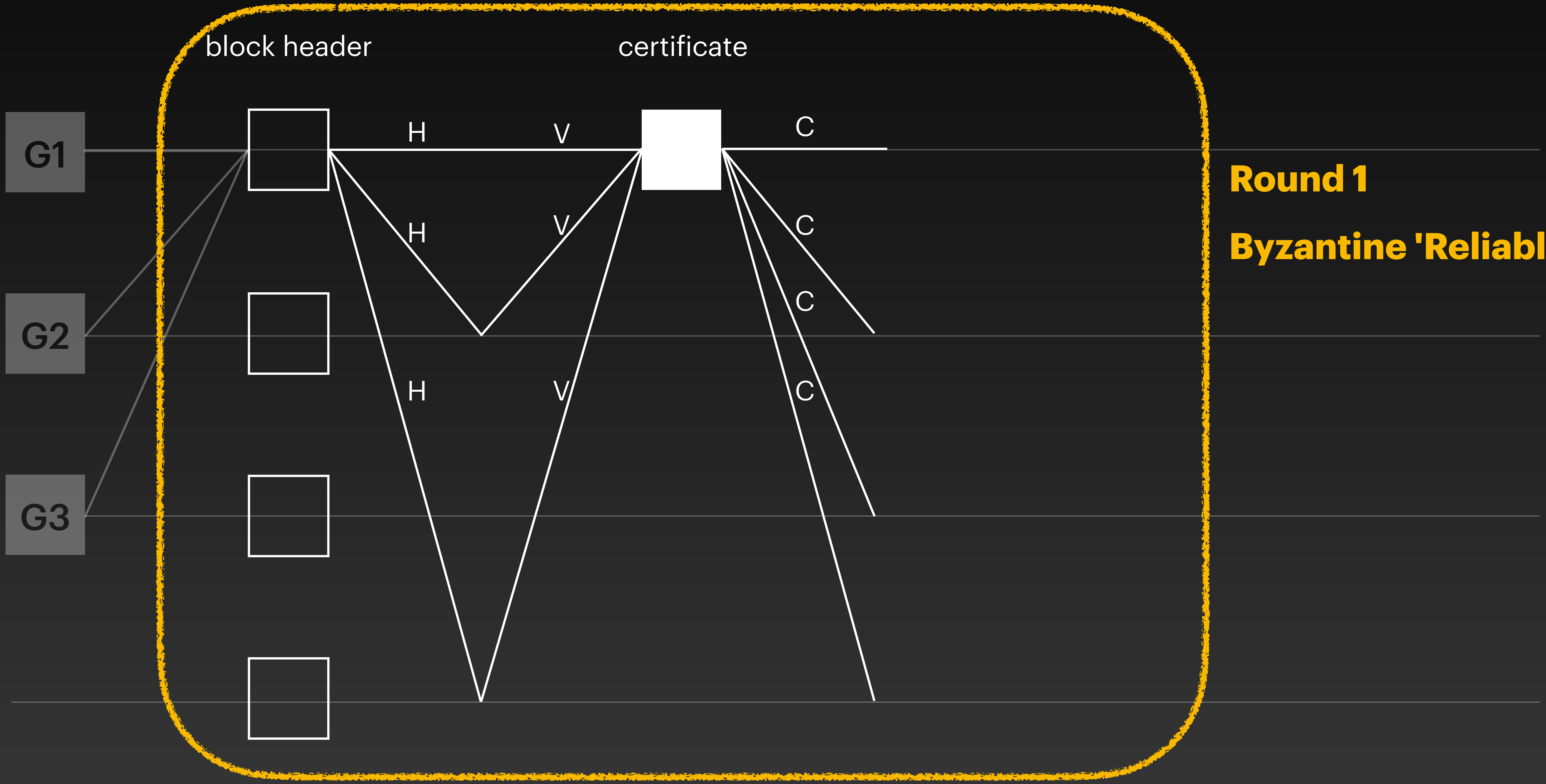
Narwhal

The primary machine



Narwhal

The primary machine

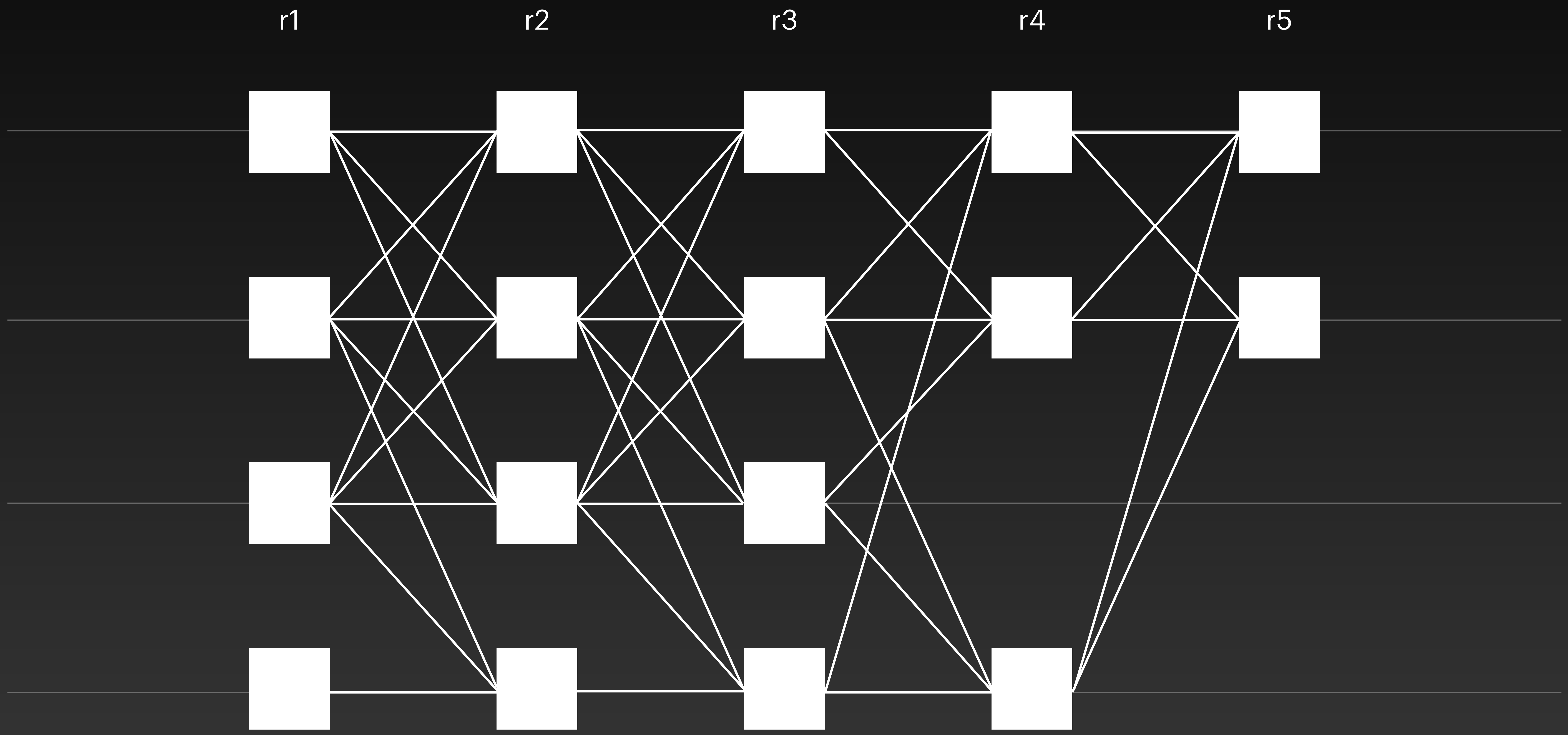


Round 1

Byzantine 'Reliable' Broadcast

Narwhal

The primary machine

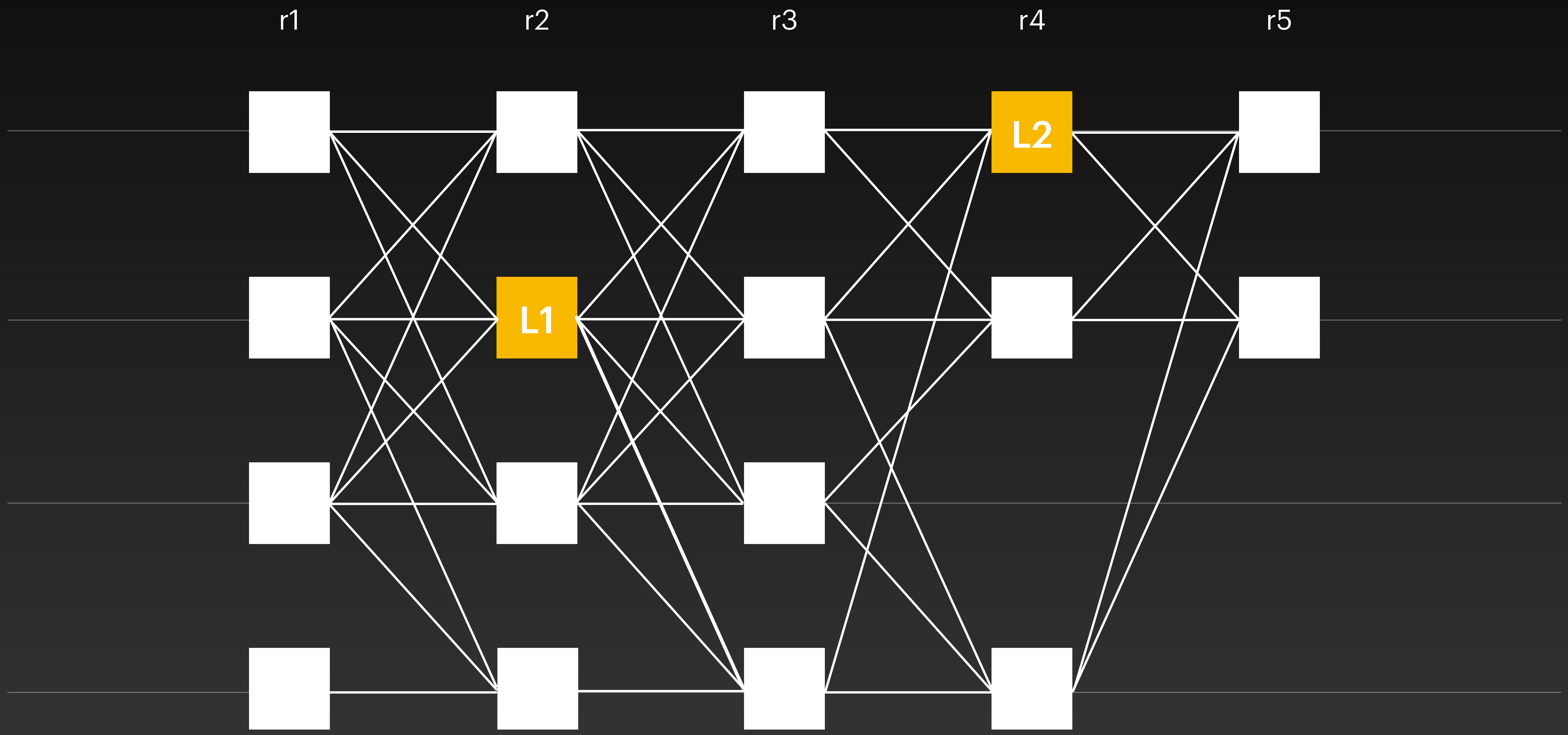


Modified Narwhal

Adapt Narwhal for partial-synchronous networks

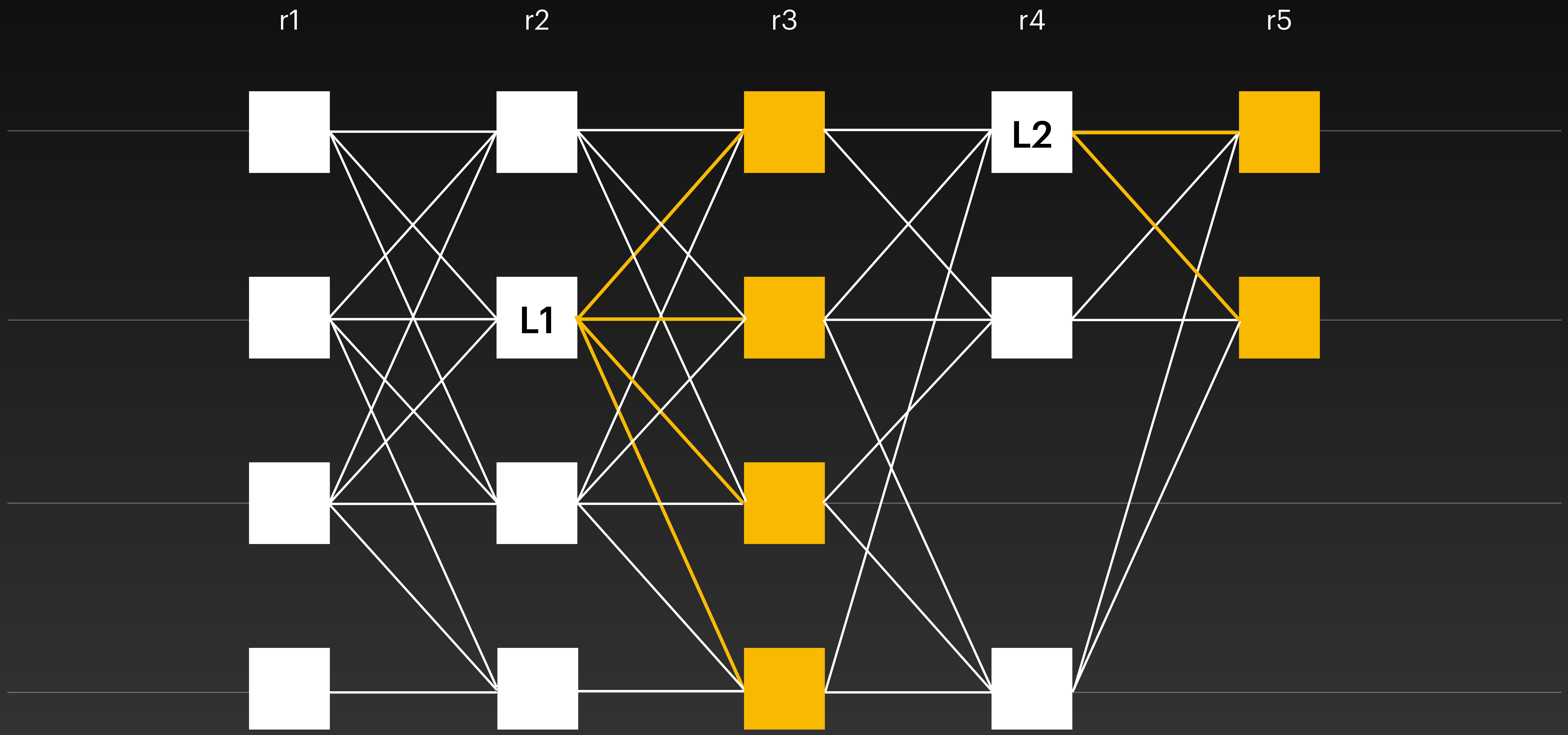
Modified Narwhal

Even rounds: wait for the leader (or to timeout)



Modified Narwhal

Odd rounds: wait for enough votes (or to timeout)



Bullshark

Zero-message partially-synchronous consensus

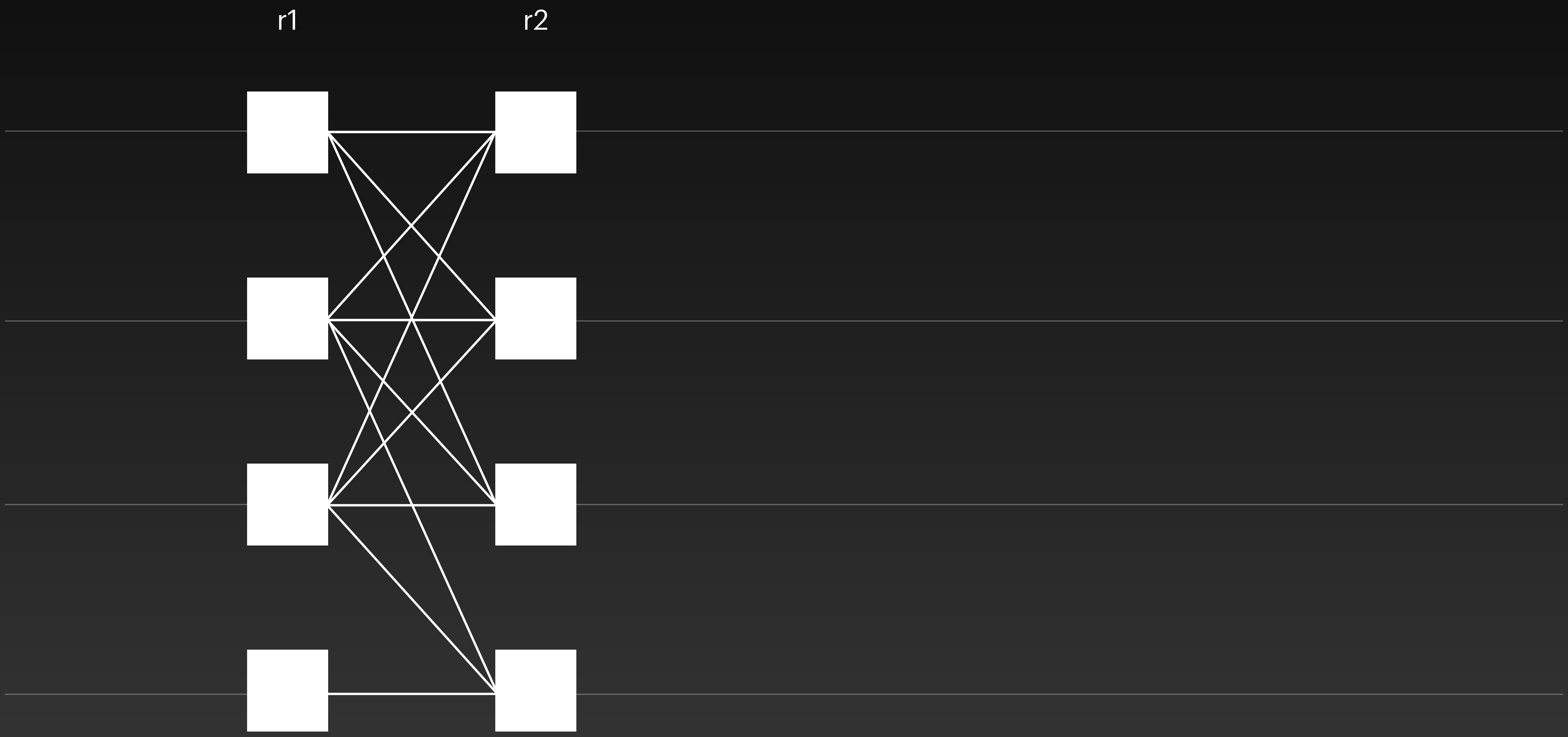
Bullshark

Zero-message partially-synchronous consensus

* without asynchronous fallback

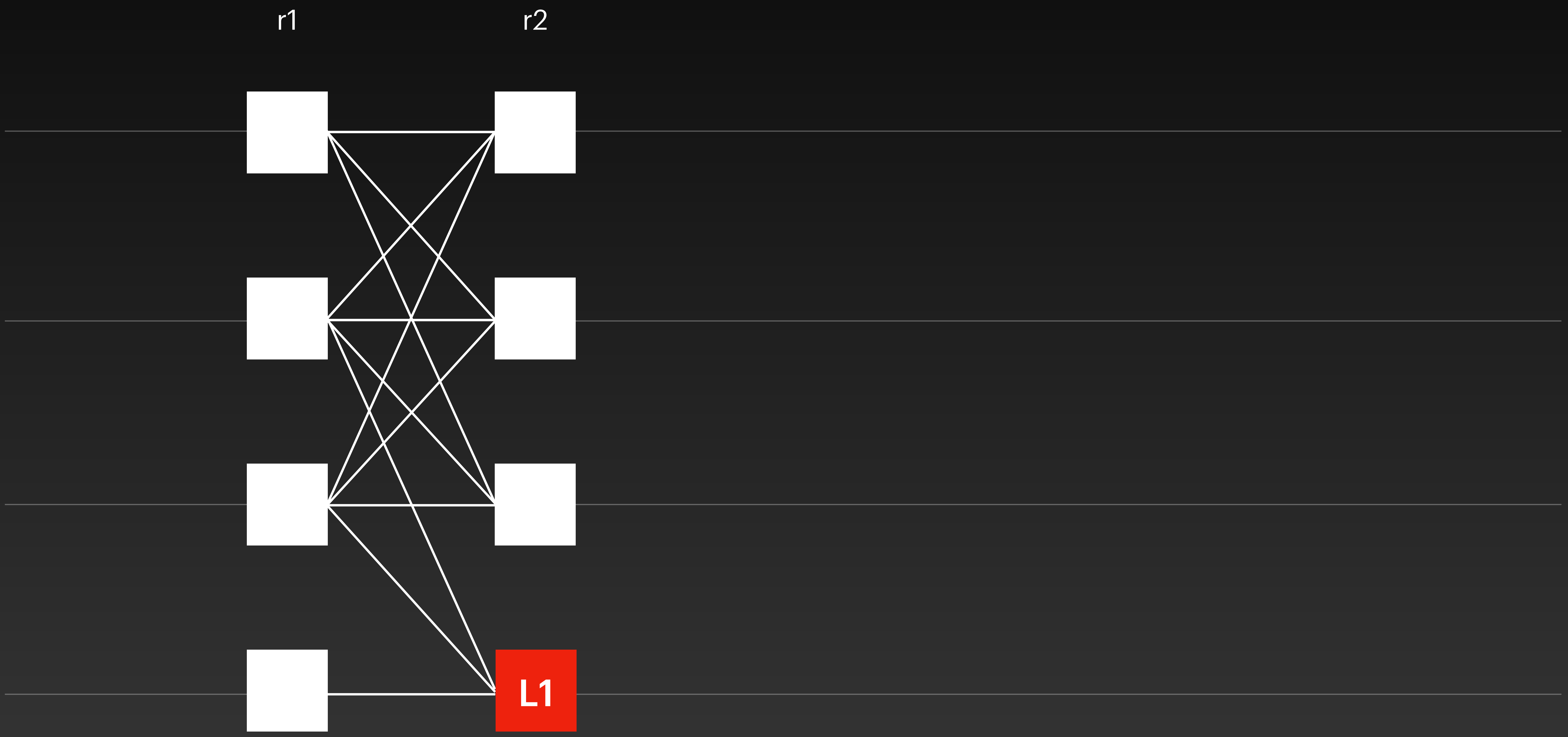
Bullshark

Just interpret the DAG



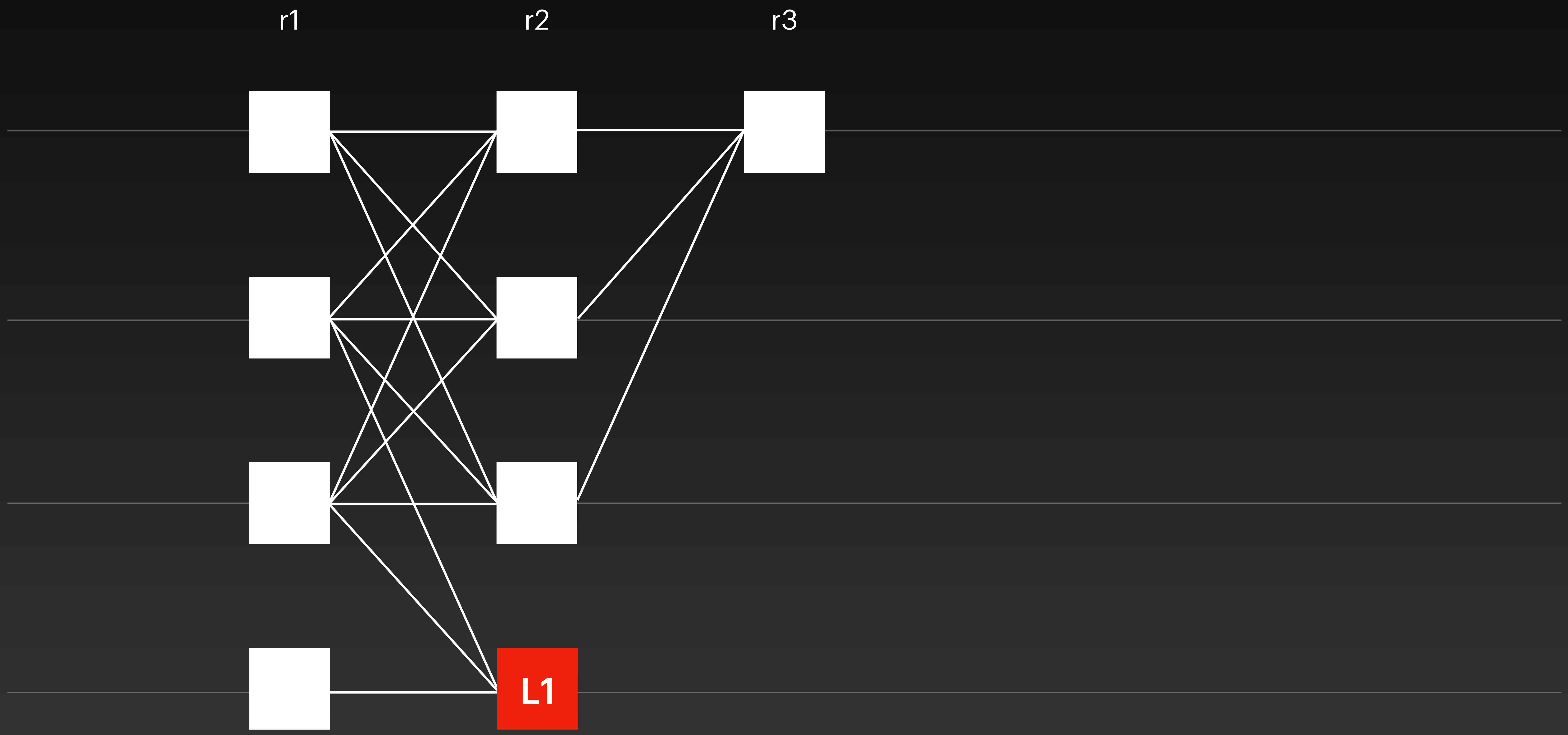
Bullshark

Deterministic leader every 2 rounds



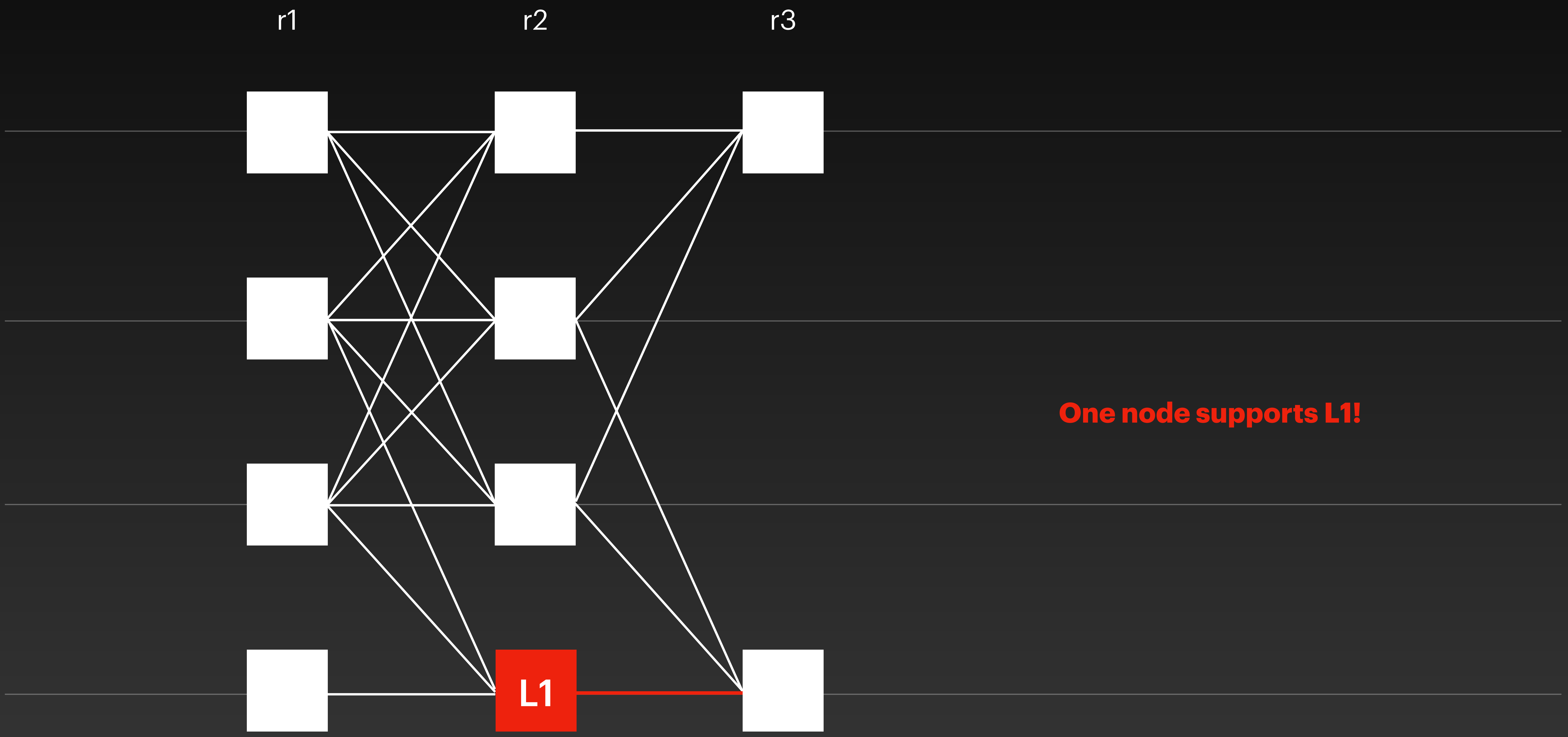
Bullshark

The leader needs f+1 links from round r



Bullshark

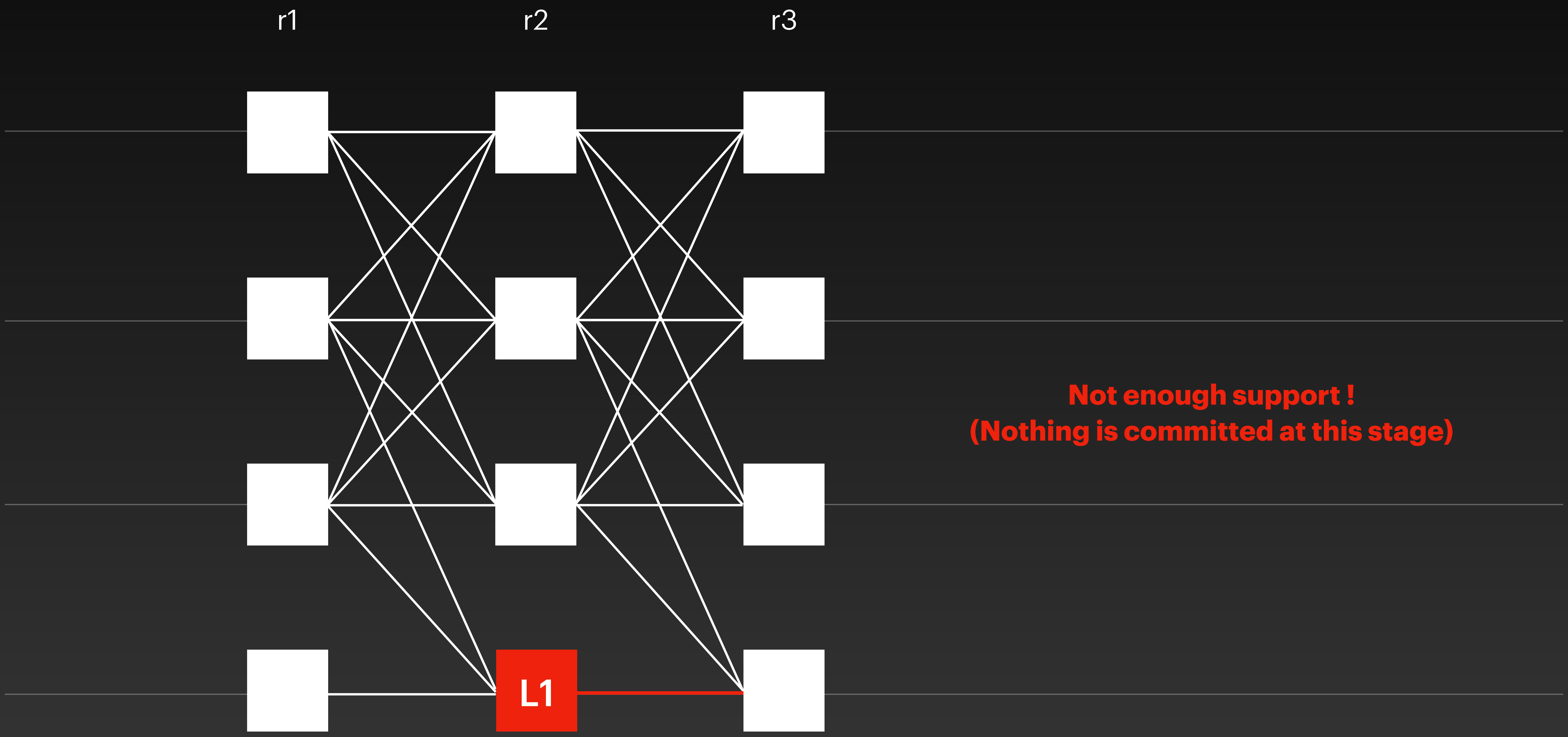
The leader needs f+1 links from round r



One node supports L1!

Bullshark

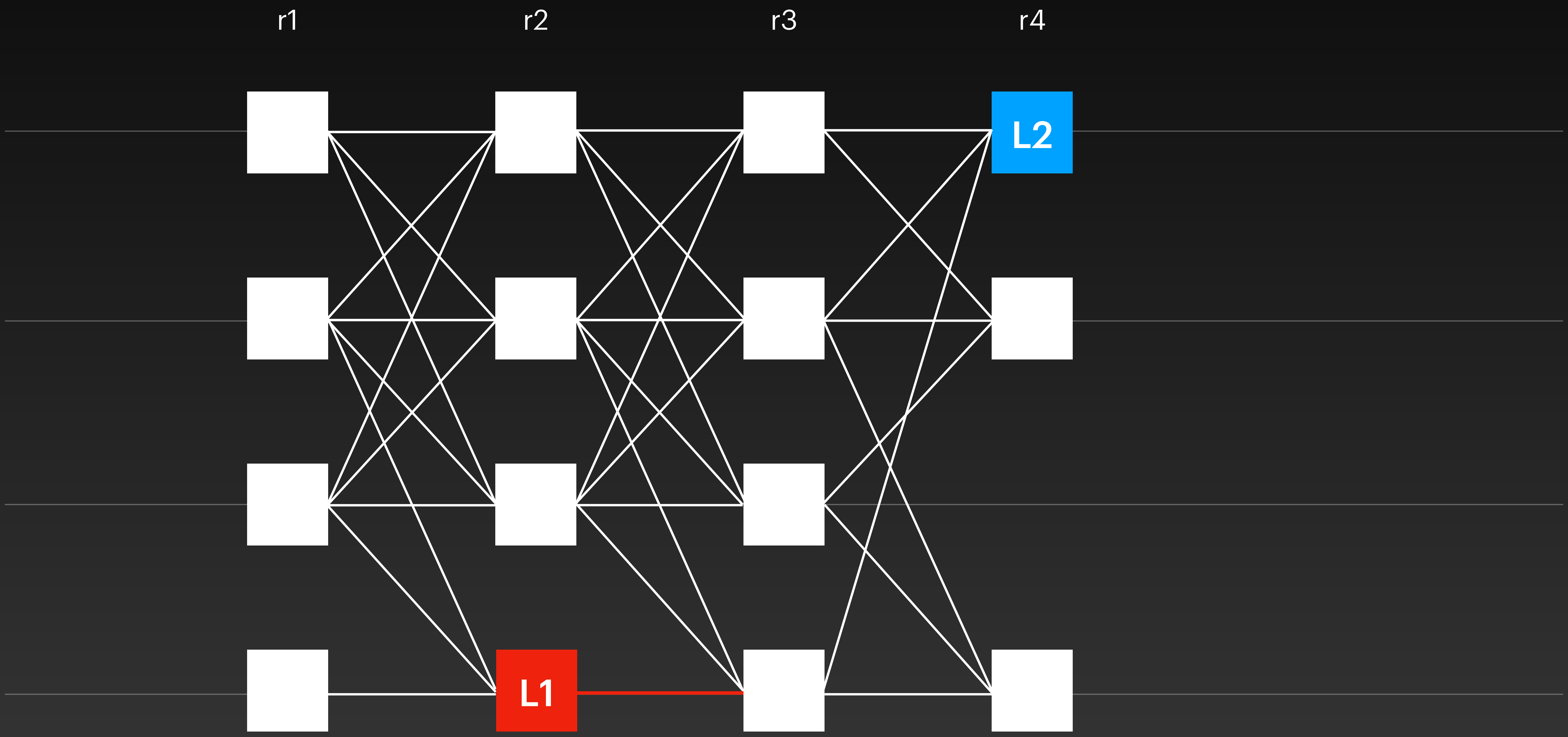
The leader needs f+1 links from round r



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

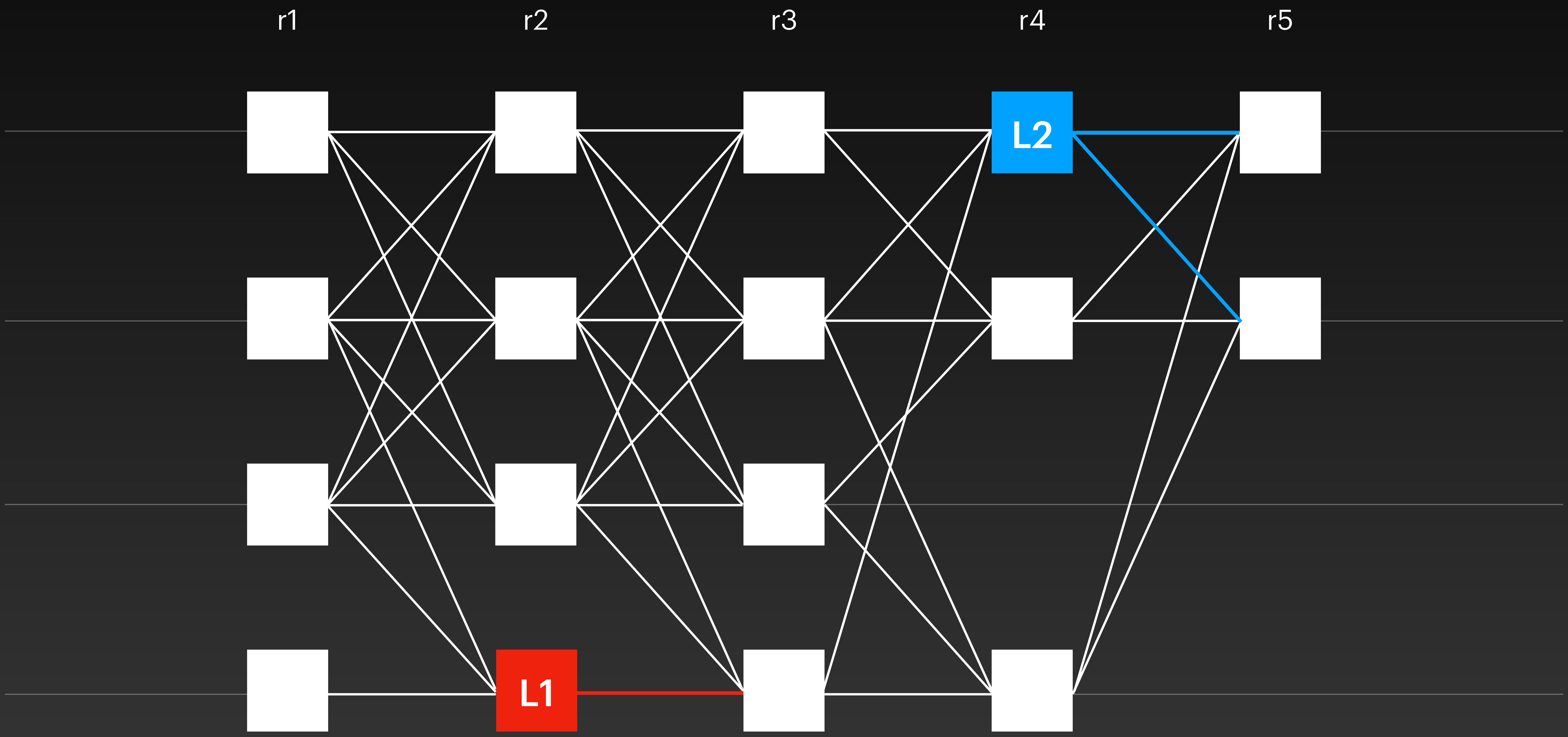
Bullshark

Elect the leader of r4



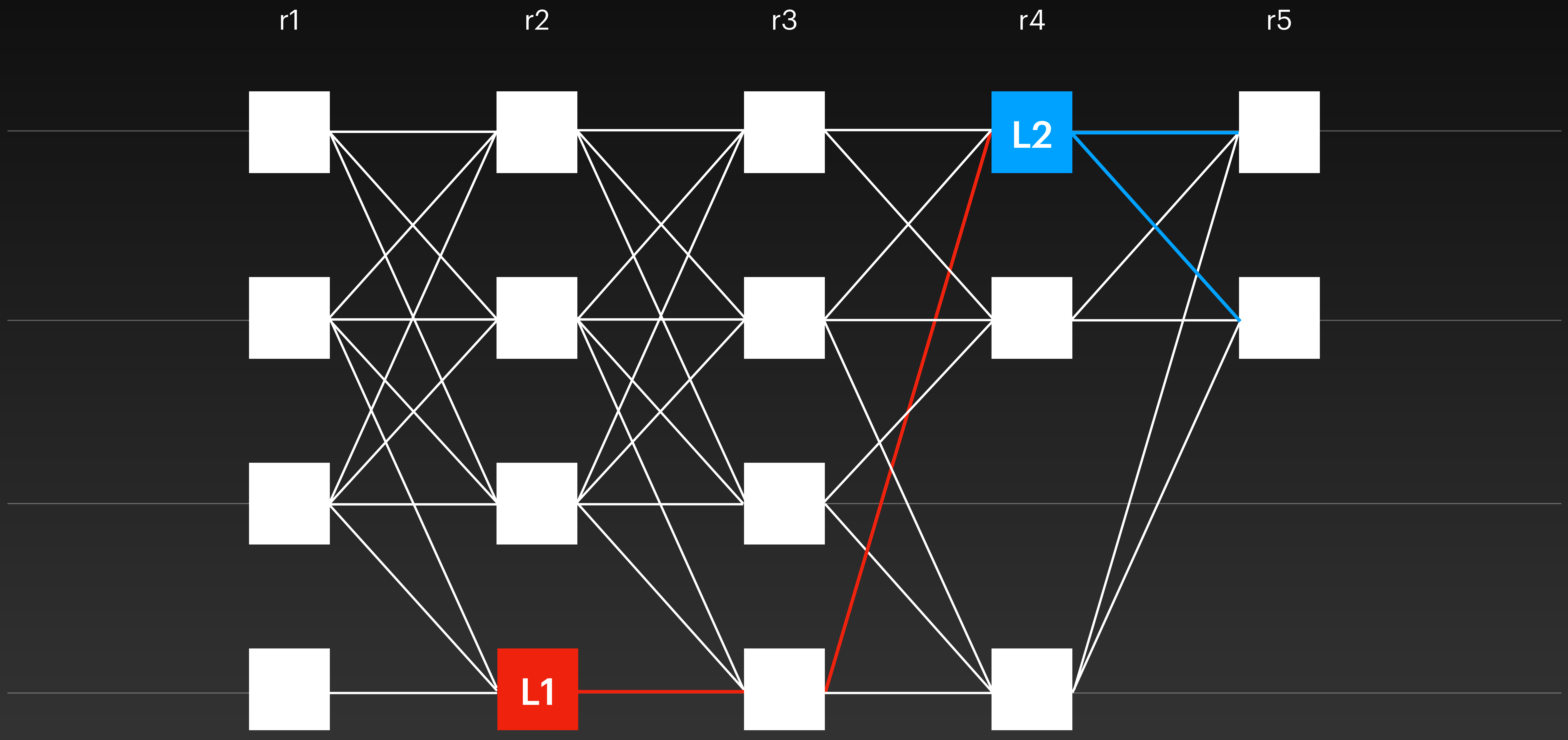
Bullshark

Leader L2 has enough support



Bullshark

Leader L2 has links to leader L1

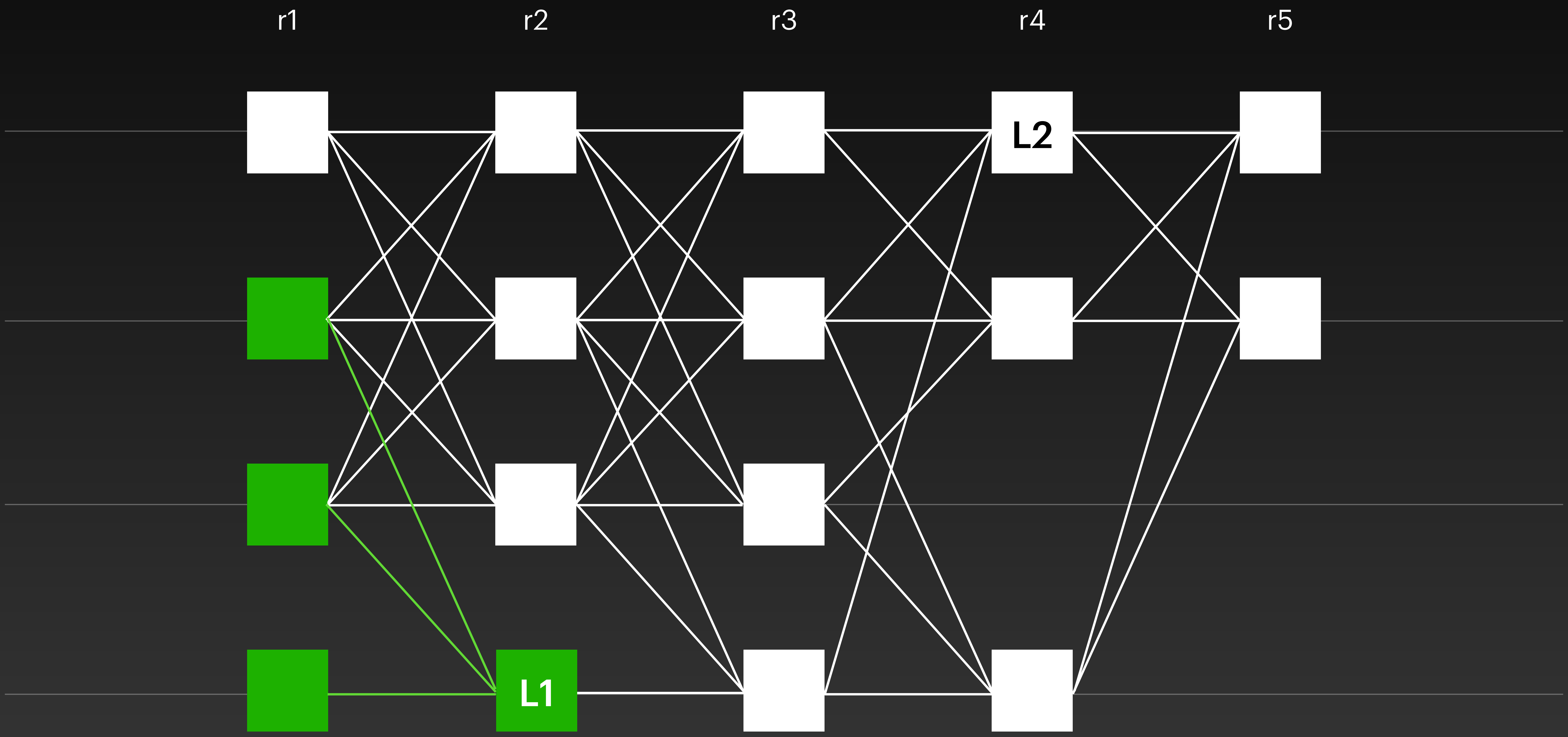


First commit L1

Then commit L2

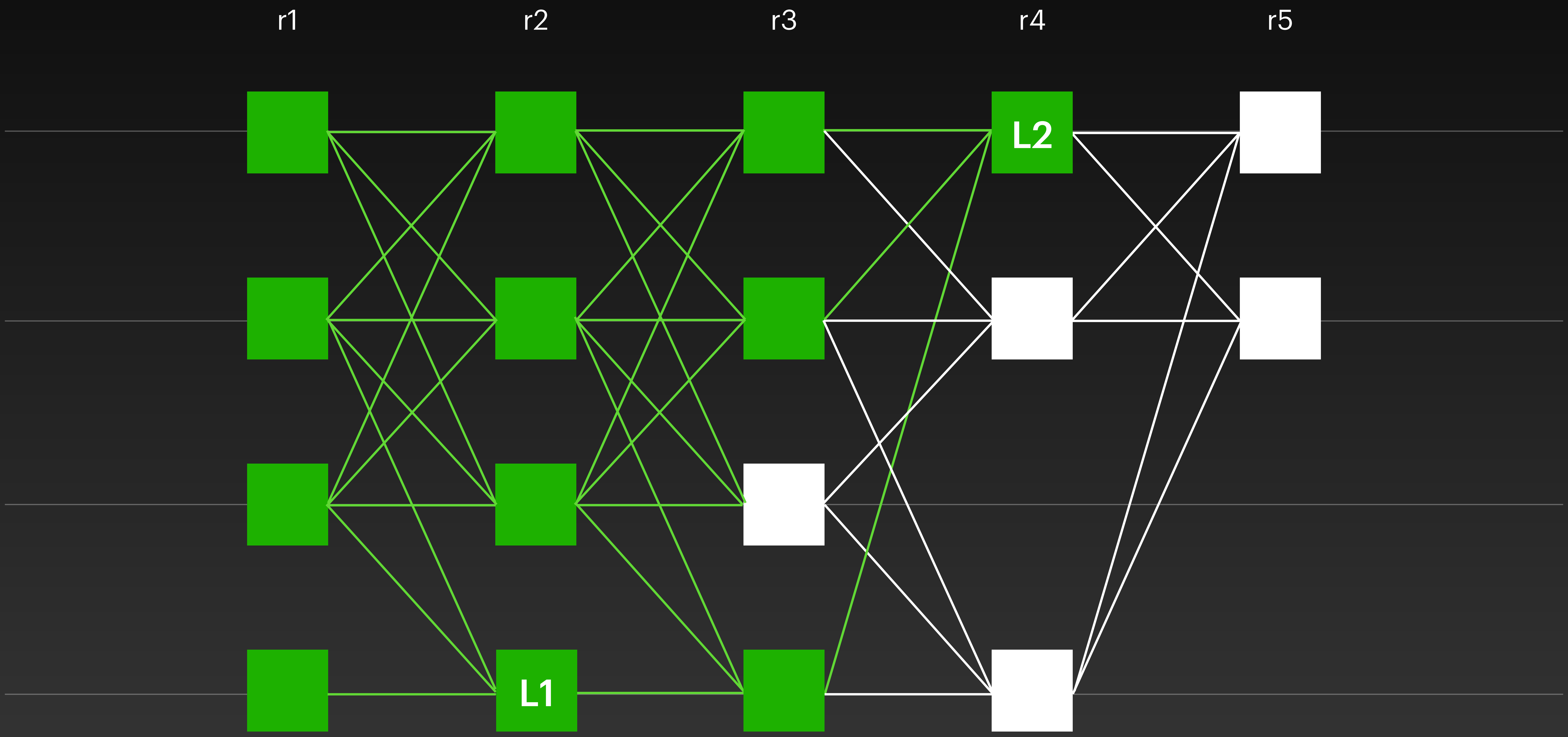
Bullshark

Commit all the sub-DAG of the leader



Bullshark

Commit all the sub-DAG of the leader



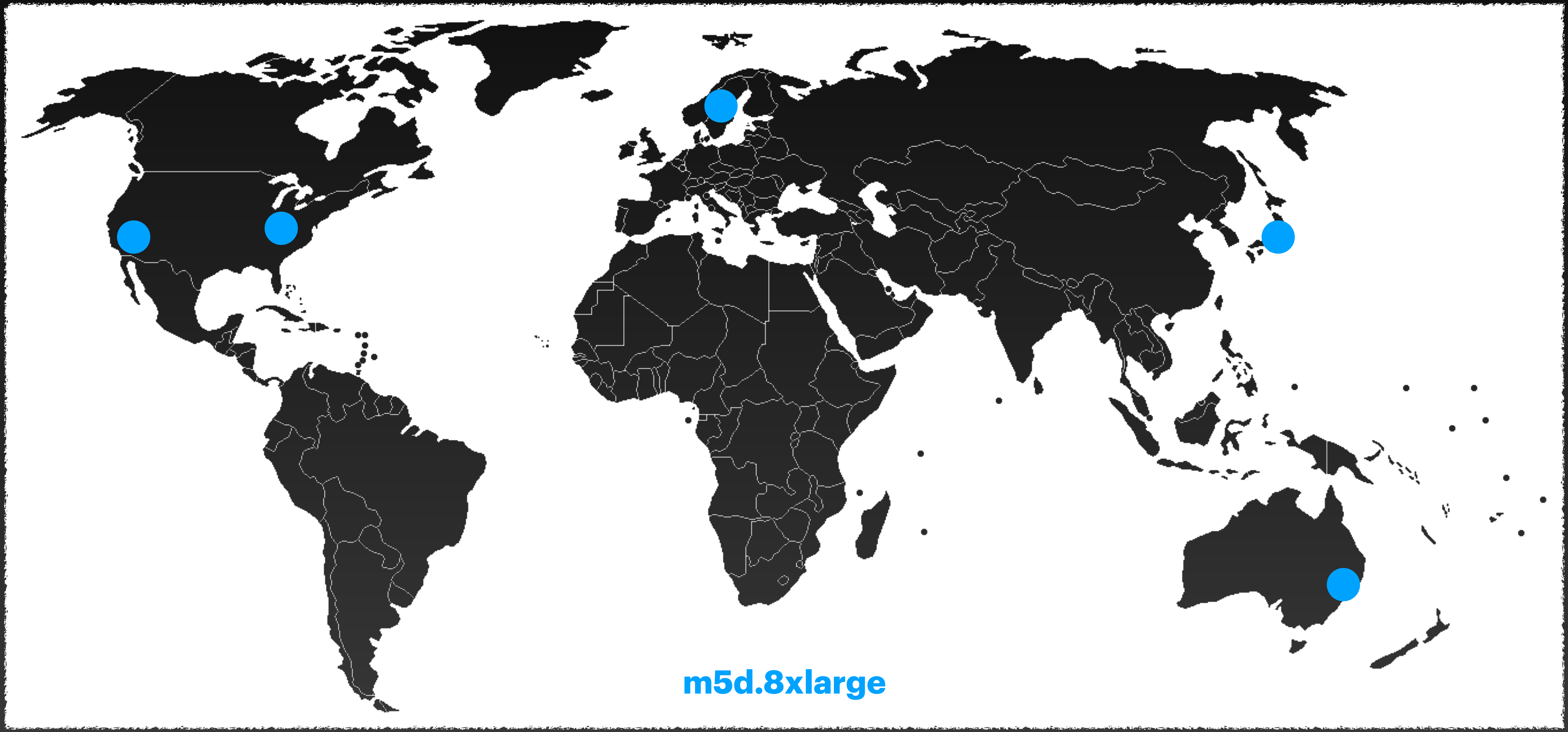
Implementation

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

https://github.com/asonnino/narwhal

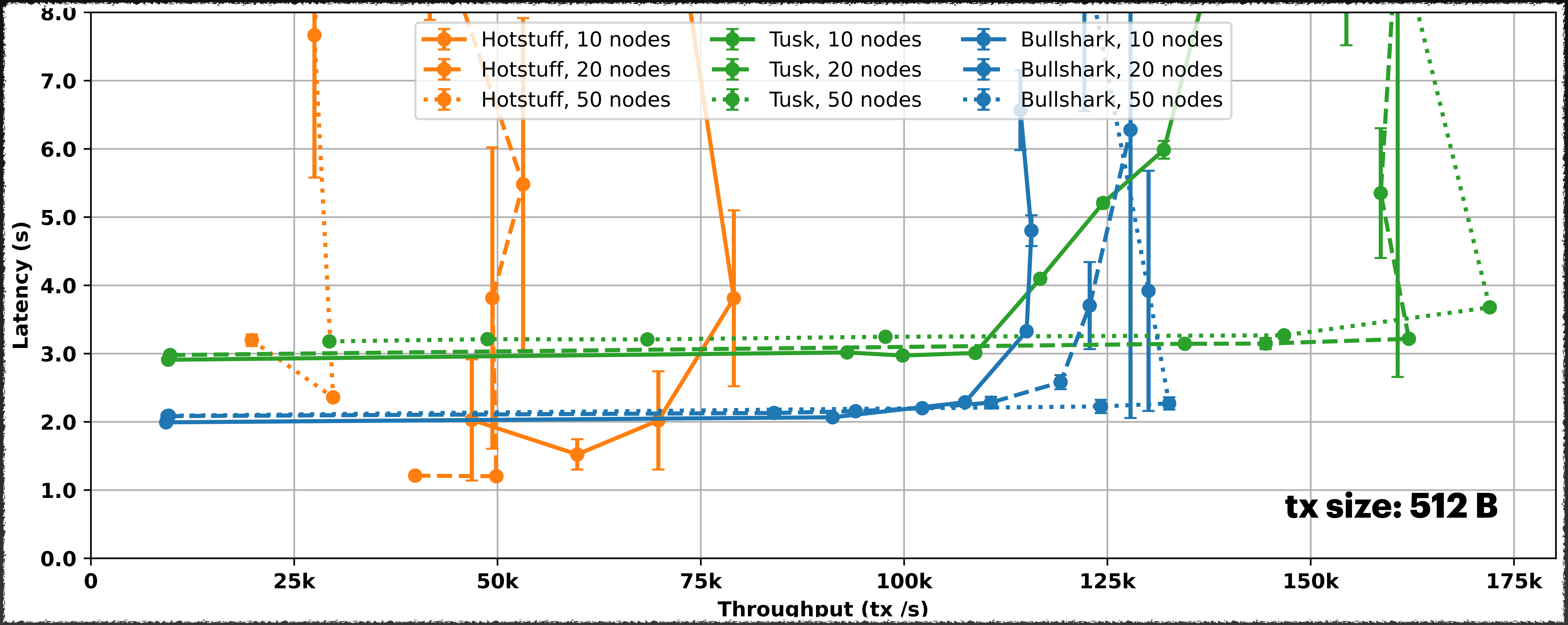
Evaluation

Experimental setup on AWS



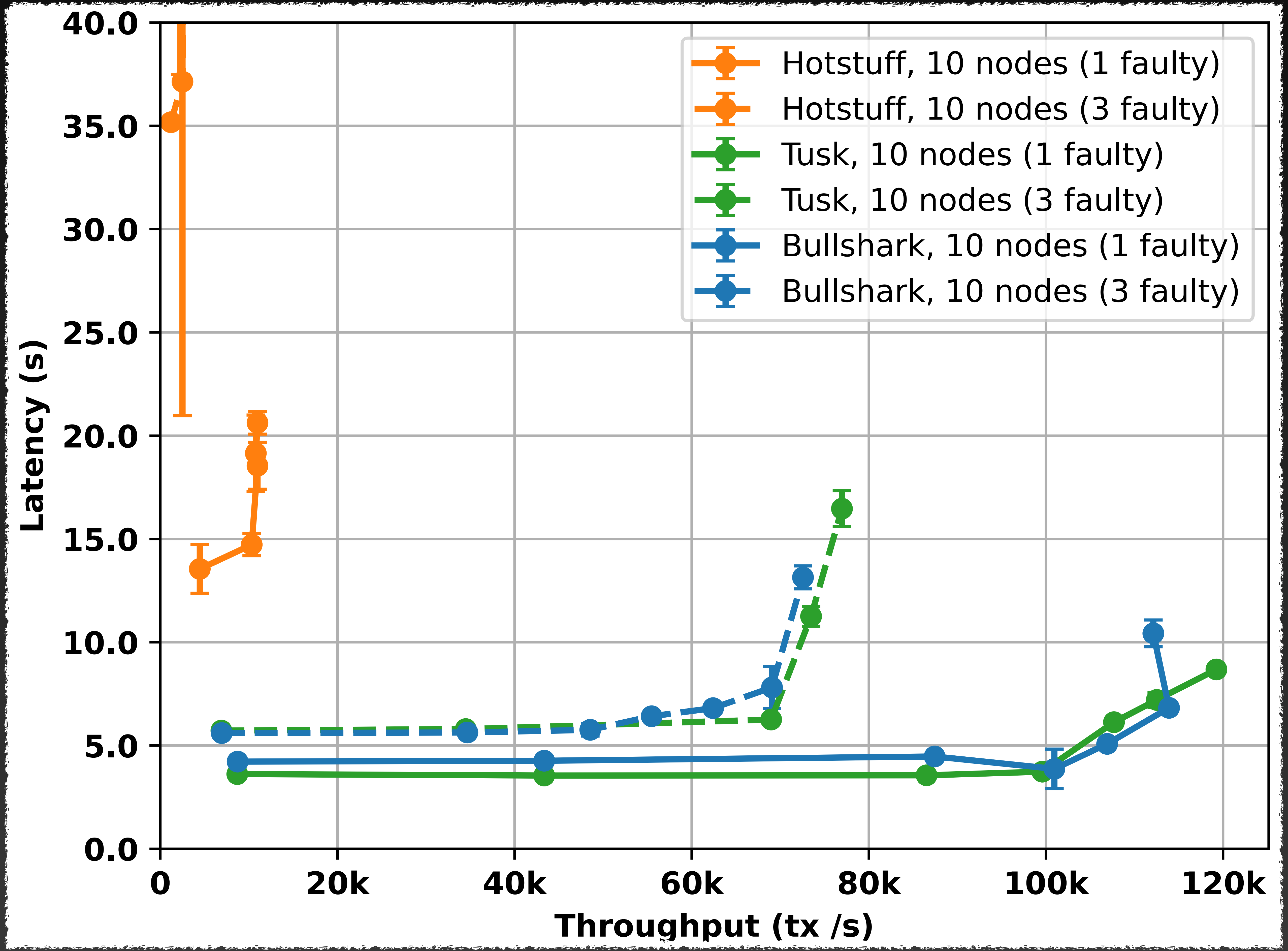
Evaluation

Throughput latency graph



Evaluation

Performance under faults



Conclusion

Bullshark

- Zero-message overhead, no view-change, no common-coin
- Disseminate data with Narwhal, exploits periods of synchrony

- **Paper:** https://sonnino.com/papers/bullshark.pdf
- **Code:** https://github.com/asonnino/narwhal

alberto@mystenlabs.com

Alberto Sonnino