RapidChain: Scaling Blockchain via Full Sharding

Mahdi Zamani, Mahnush Movahedi, Mariana Raykova

Presented by Avish Menon and Neil Arakkal

Agenda

- Limitations of Current Consensus Protocols
- Overview of RapidChain
- Bootstrapping
- Consensus
- Reconfiguration
- Evaluation

Traditional Byzantine Consensus

- Vulnerable to Sybil attacks if used in an open-membership setting
- Protection against Sybil is inefficient

Bitcoin

- Uses Nakamoto Consensus
- Inhibits Sybil using PoW
- Full Replication: Low transaction throughput, high latency, poor scaling
- Tradeoff between scalability and decentralization

Sharding-Based Consensus

- Electing random small committees
  - Disjoint Ledger

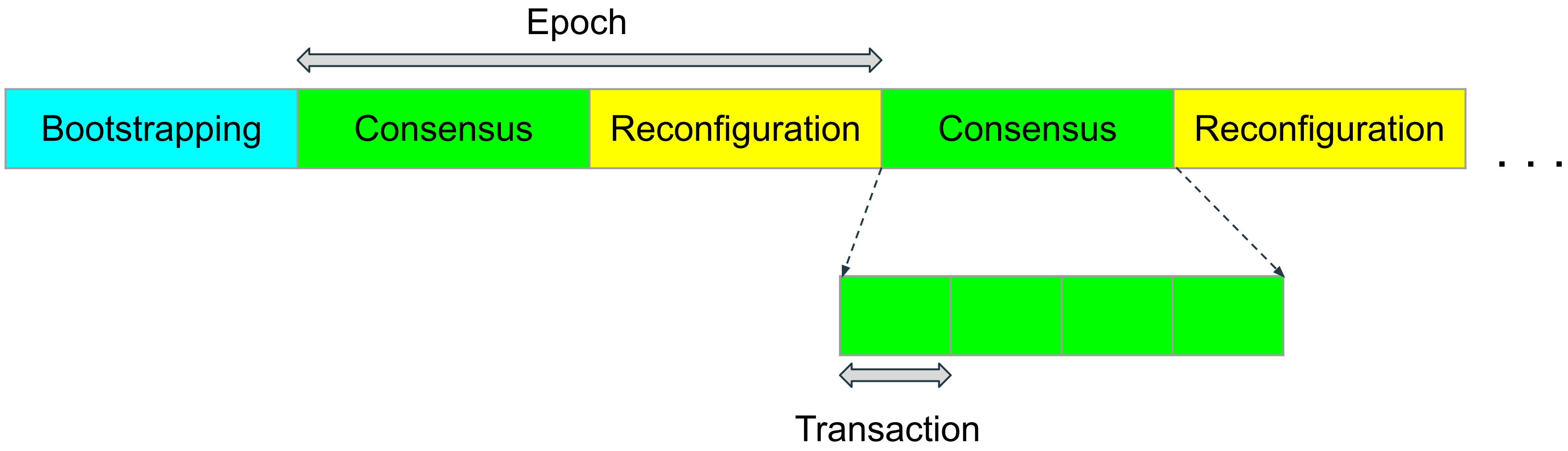
RapidChain Aims

- Electing via probabilistic sampling process
- Reconfiguration to avoid Sybil Attacks
- Cross-shard transactions
  - Verification of transactions involving other committees
- Decentralized bootstrapping
  - Creating initial random committees

Model Overview

- n nodes with public (pk_i) and secret key(sk_i)
- m committees
- $t < n/3$ Byzantine nodes
- Less than 5% churn in every epoch

Top Level Diagram



RapidChain Top-Level

- Bootstrapping Phase - Establishing a reference committee
  - Epoch Randomness
    - First sharding committees
    - Challenges for new nodes
    - Reorganization of existing committees
  - Reconfiguration Block

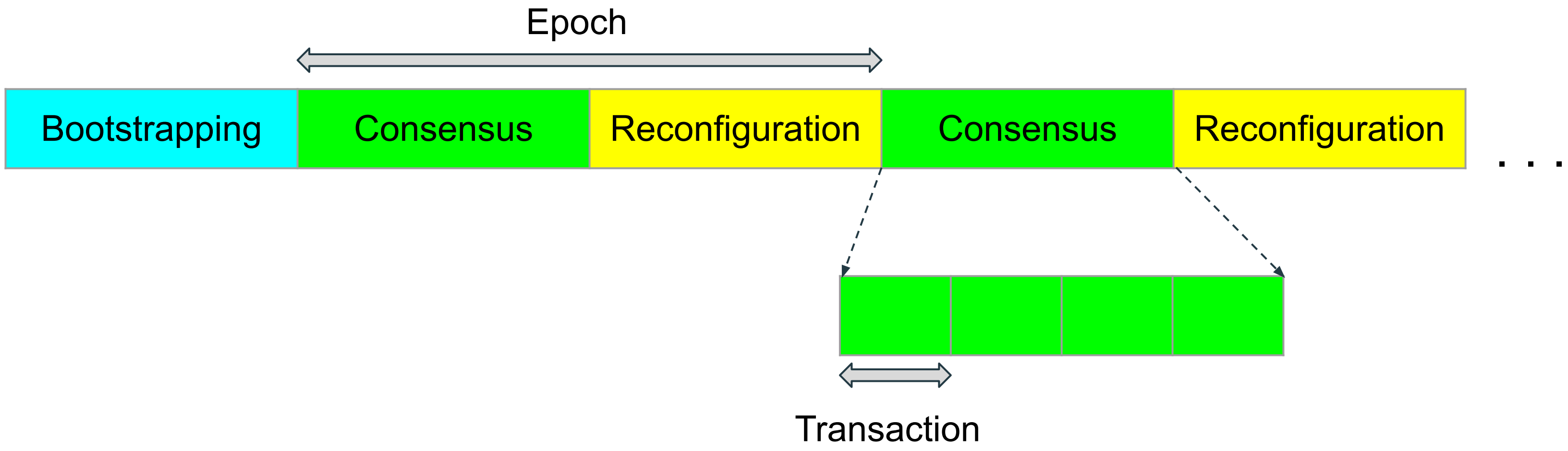
RapidChain Top-Level (cont’d)

- Consensus Phase
  - Each tx is sent to a random node
  - Tx sent to the output committee
  - Committee verifies and adds tx to block

RapidChain Top-Level (cont’d)

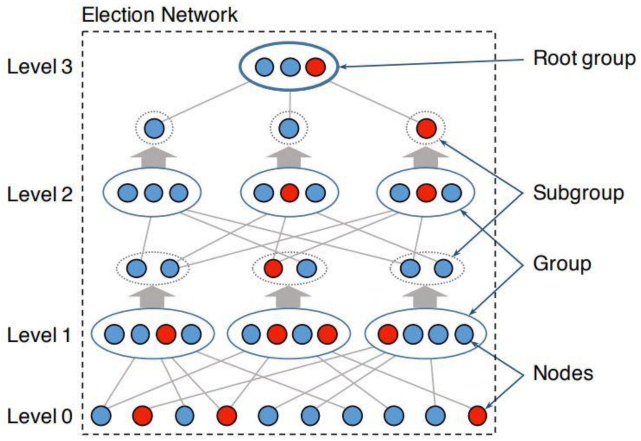
- Reconfiguration Phase
    - Reconfiguration block generated
        - Fresh epoch randomness
        - New list of participants
    - Cuckoo rule used to reconfigure existing committees

Stages of RapidChain



Bootstrapping

- Runs an election committee protocol
- Divides nodes into committees
- Only runs at the initialization of RapidChain



Consensus in Committees

Consists of two parts:

- A gossiping protocol for intershard communication
- A synchronous consensus protocol to agree on the header of the block

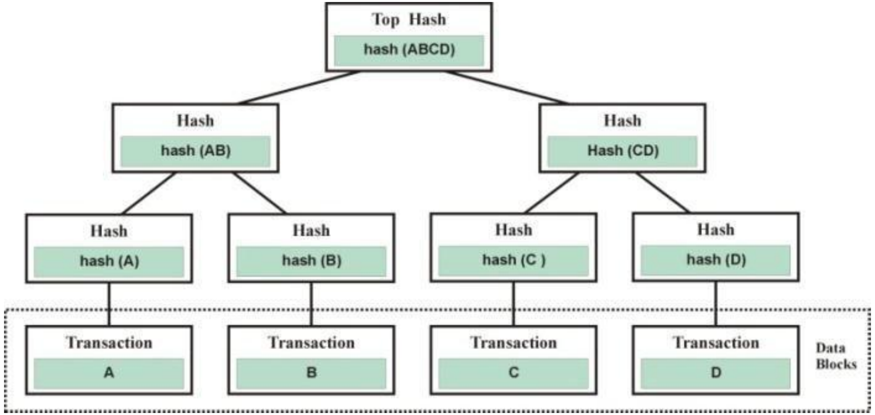
Gossiping Large Blocks

Information dispersal algorithm (IDA):

- Encode message into k chunks (M1,...,Mk) using an erasure coding mechanism.
- Give each neighbor k/d chunks (d is the number of neighbors)
- The message can be reconstructed from any set of (1-f)k chunks

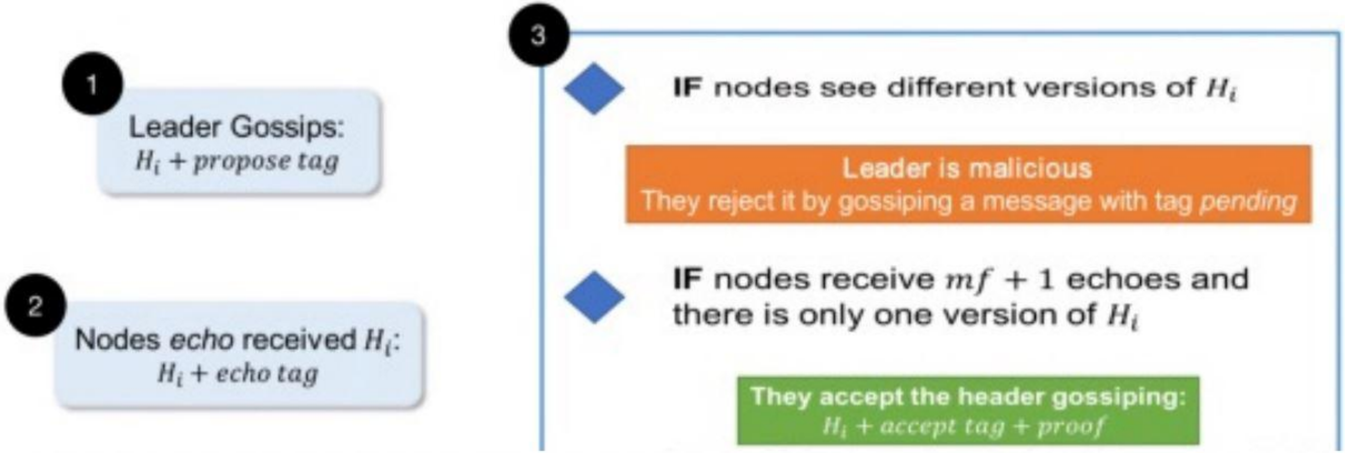
Gossiping Large Blocks

- Compute a Merkle hash tree over message chunks M1, …, Mk
- Send Merkle proof along with message chunk to neighbors
- Each node verifies the message using the Merkle proof and the Merkle root.



Consensus Protocol Details

- A new leader is picked at the start of each iteration.



1 Leader Gossips:
$H_i + propose\ tag$

2 Nodes *echo* received H_i:
$H_i + echo\ tag$

3 **IF** nodes see different versions of H_i

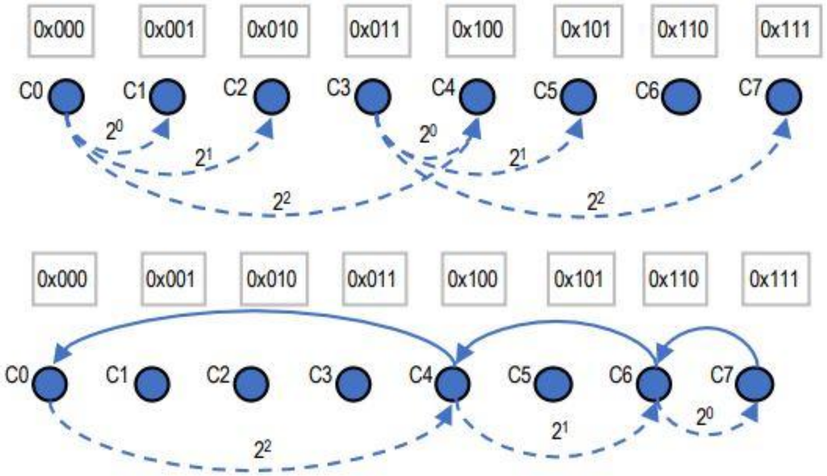
Leader is malicious
They reject it by gossiping a message with tag *pending*

IF nodes receive $mf + 1$ echoes and there is only one version of H_i

They accept the header gossiping:
$H_i + accept\ tag + proof$

Inter-Committee Routing

Each committee maintains a routing table of log n records that point to log n different committees



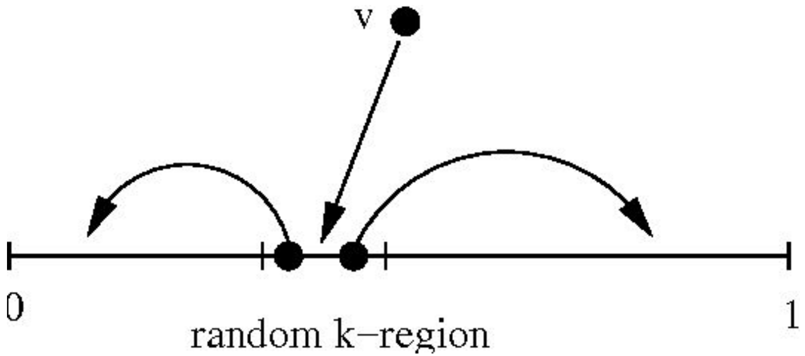
Problem with Committees

Leave/join attacks: Corrupt nodes could strategically rejoin the network to take control of a committee.

Reconfiguration (Resharding)

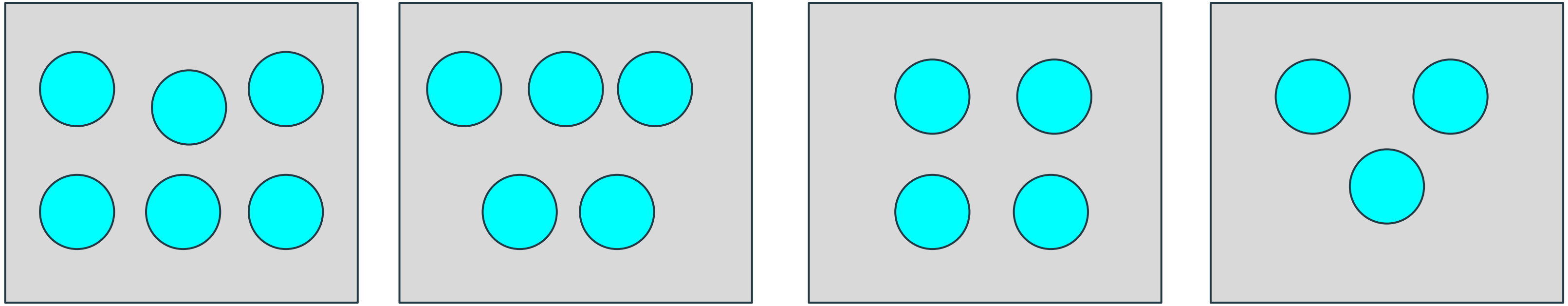
Cuckoo rule:

- New node assigned a random shard
- Evict k nodes from the shard, not including the new node



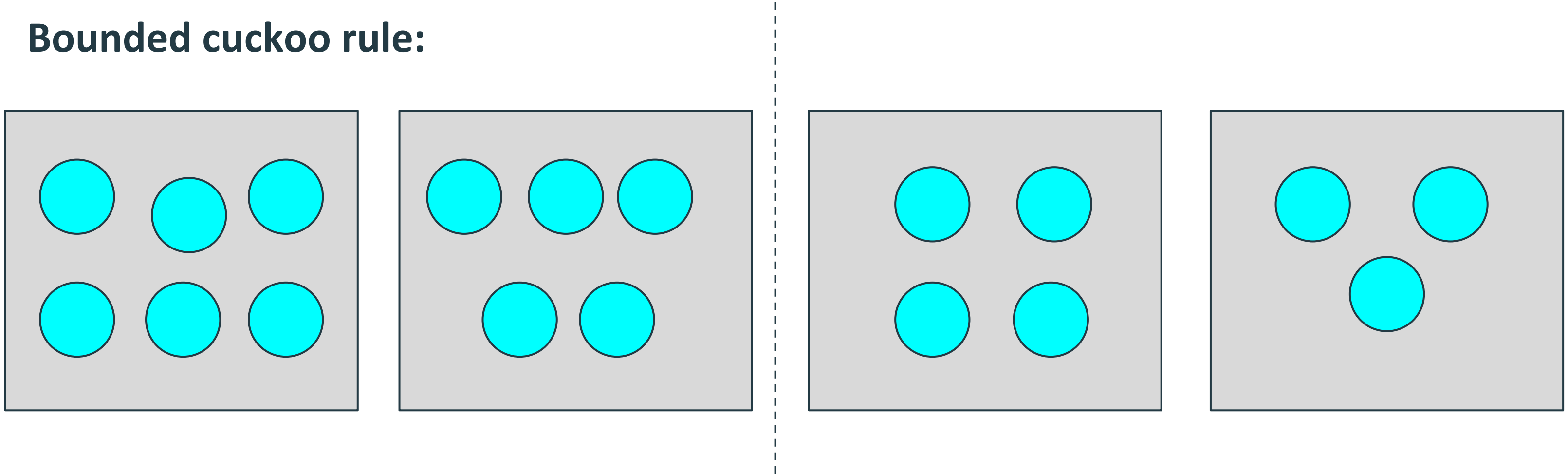
Reconfiguration (Resharding)

Bounded cuckoo rule:



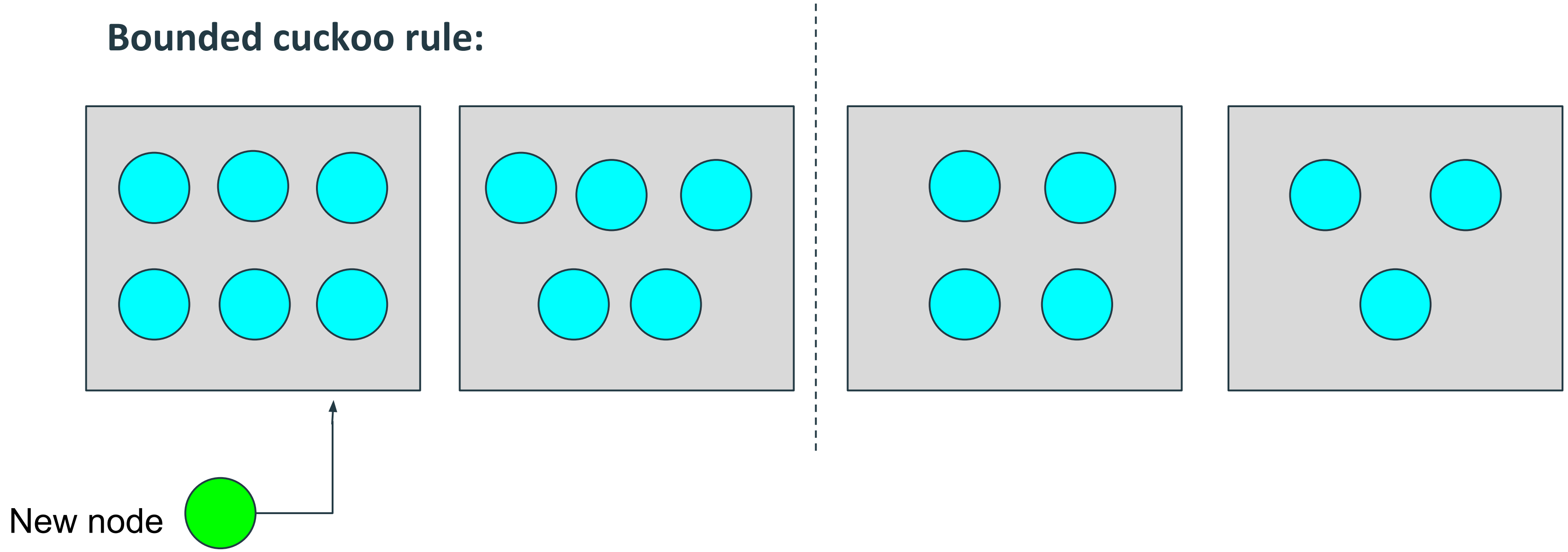
Reconfiguration (Resharding)

Bounded cuckoo rule:

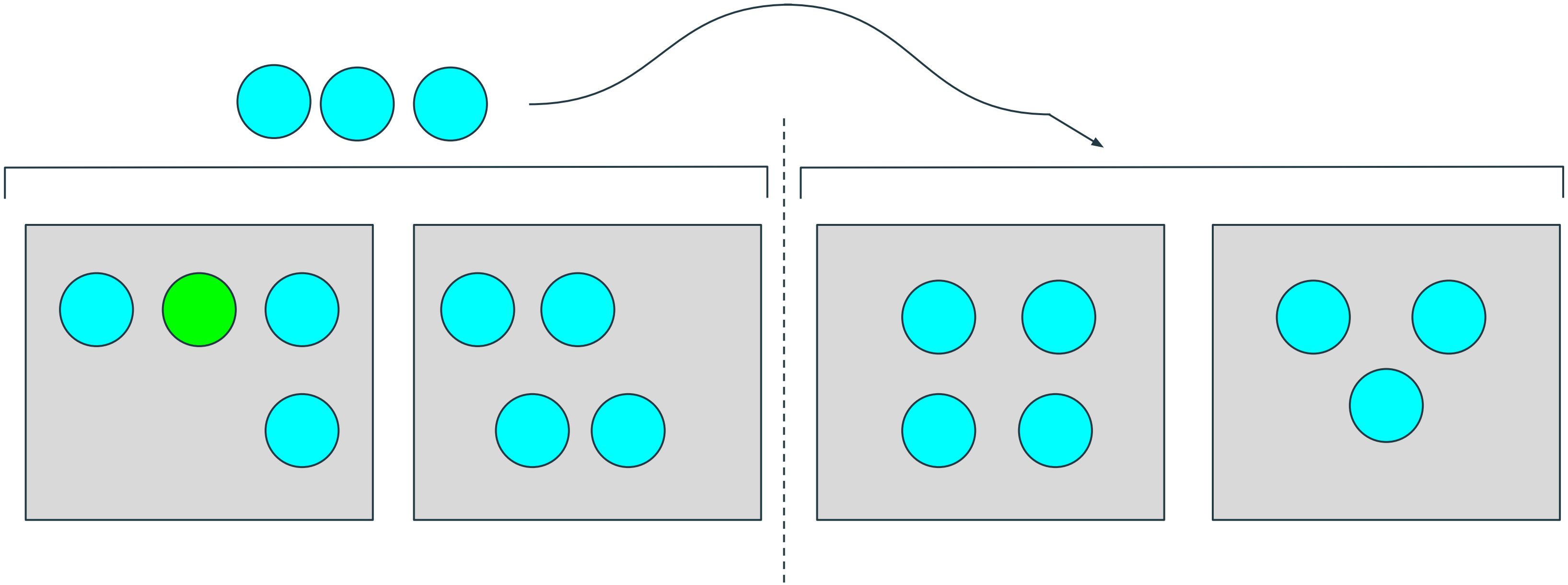


Reconfiguration (Resharding)

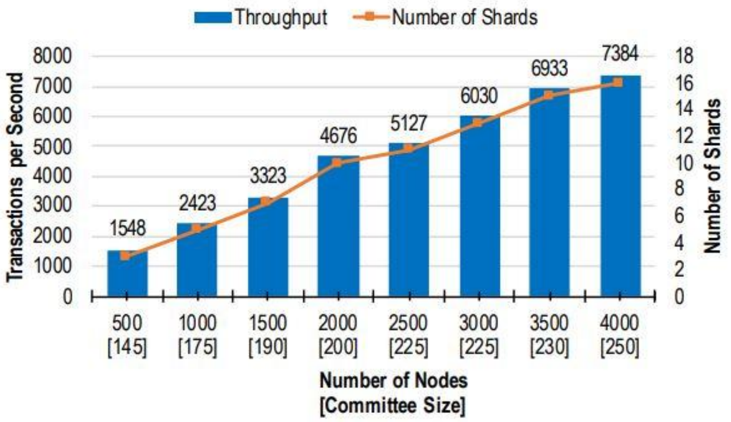
Bounded cuckoo rule:

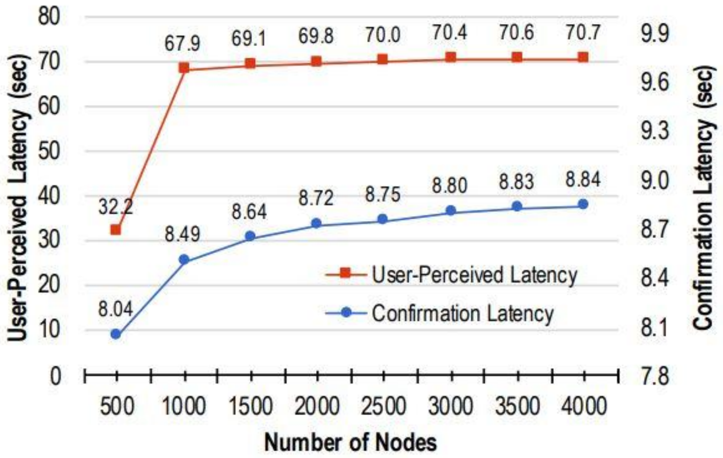


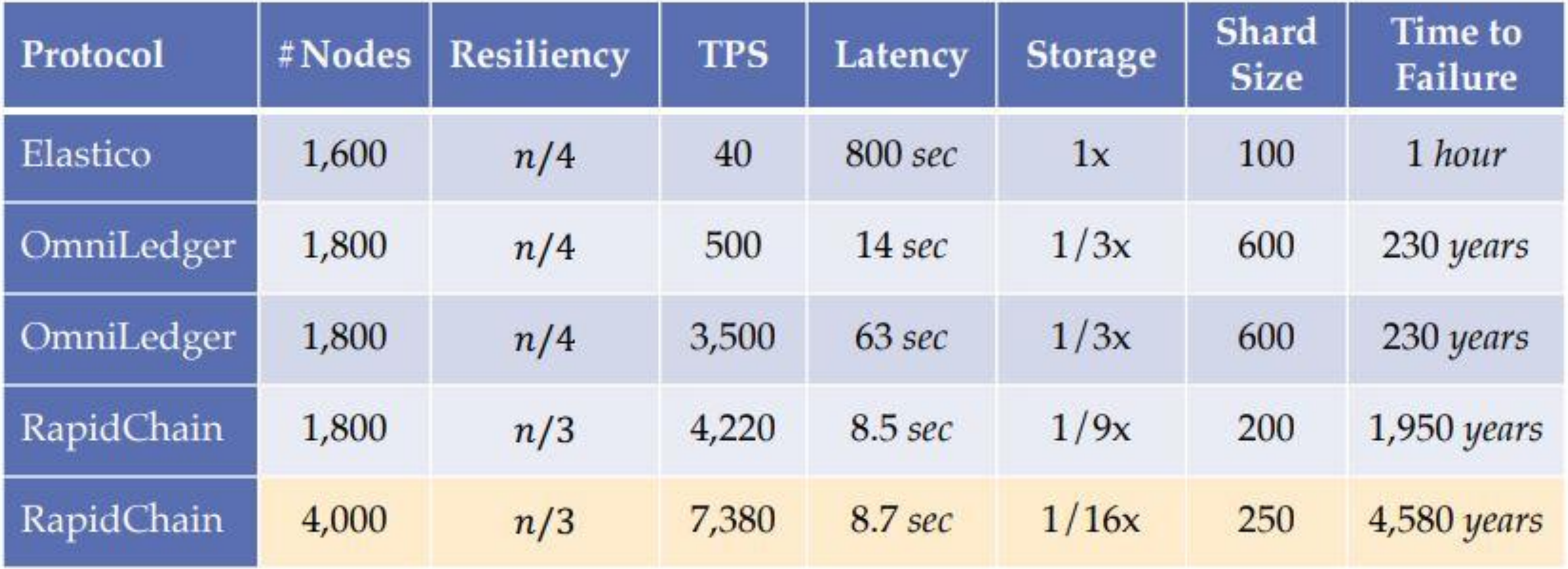
Reconfiguration (Resharding)



Evaluation


Throughput
Number of Shards
Transactions per Second
8000
7000
6000
5000
4000
3000
2000
1000
0
1548
2423
3323
4676
5127
6030
6933
7384
18
16
14
12
10
8
6
4
2
0
Number of Shards
500 [145]
1000 [175]
1500 [190]
2000 [200]
2500 [225]
3000 [225]
3500 [230]
4000 [250]
Number of Nodes
[Committee Size]


User-Perceived Latency (sec)
80
70
60
50
40
30
20
10
0
32.2
67.9
69.1
69.8
70.0
70.4
70.6
70.7
8.04
8.49
8.64
8.72
8.75
8.80
8.83
8.84
User-Perceived Latency
Confirmation Latency
9.9
9.6
9.3
9.0
8.7
8.4
8.1
7.8
Confirmation Latency (sec)
500
1000
1500
2000
2500
3000
3500
4000
Number of Nodes



Protocol	#Nodes	Resiliency	TPS	Latency	Storage	Shard Size	Time to Failure
Elastico	1,600	$n/4$	40	800 *sec*	1x	100	1 *hour*
OmniLedger	1,800	$n/4$	500	14 *sec*	1/3x	600	230 *years*
OmniLedger	1,800	$n/4$	3,500	63 *sec*	1/3x	600	230 *years*
RapidChain	1,800	$n/3$	4,220	8.5 *sec*	1/9x	200	1,950 *years*
RapidChain	4,000	$n/3$	7,380	8.7 *sec*	1/16x	250	4,580 *years*

Thank you!