RapidChain:
Scaling Blockchain *via* Full Sharding

Mahdi Zamani
Visa Research

Join work with
Mahnush Movahedi, *Dfinity*
Mariana Raykova, *Yale University*

Stanford Blockchain Seminar
August 2018

Modified by Xinyuan Sun (sxysun@ucdavis.edu)

VISA

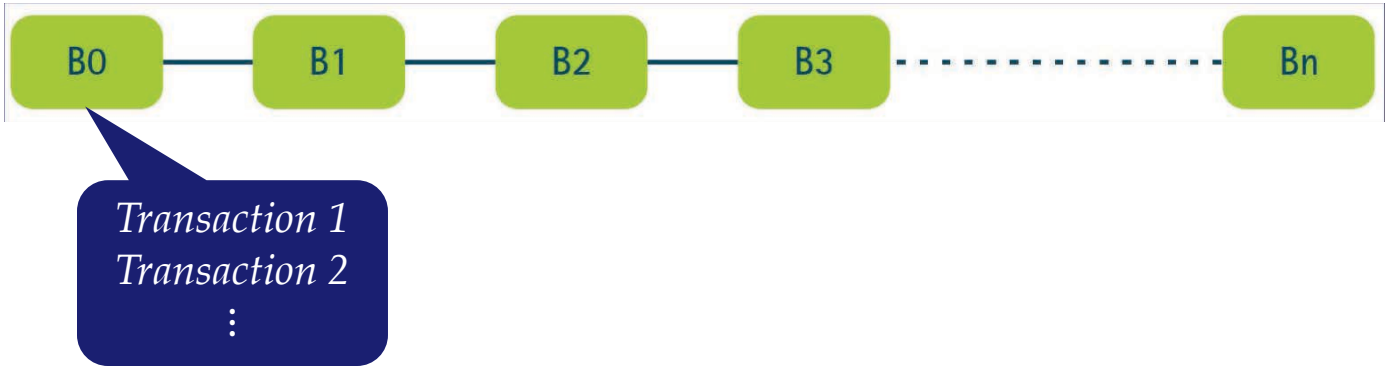
Agenda

- *Part I: Consensus Protocols*
  - Traditional mechanisms
  - Blockchain consensus
- *Part II: RapidChain*[CCS 2018]
  - Sharding-based consensus
  - Protocol overview
  - Results

Part I: Consensus Protocols

Blockchain Consensus (since 2008)

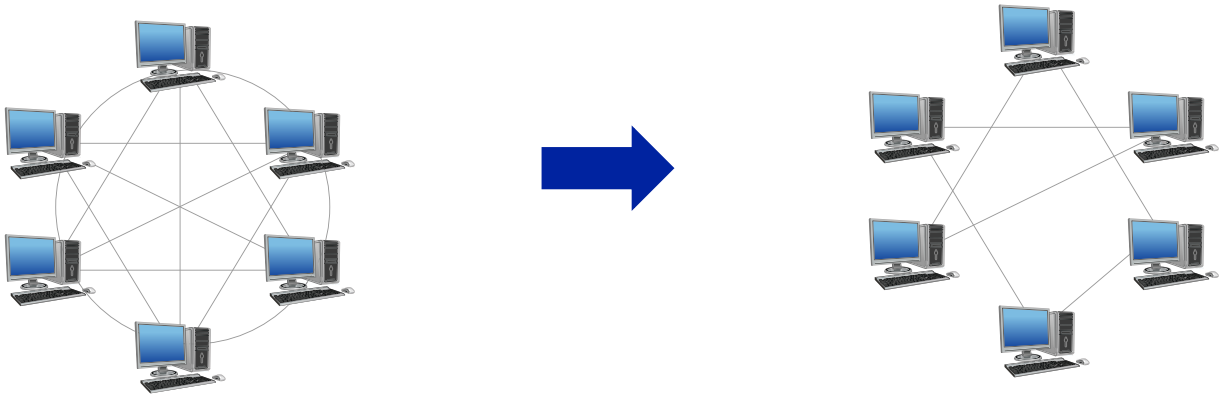
- Atomic broadcast [HT93]
  - *Total Order*
    - An honest node receives m_1 before m_2 iff sender sends m_1 before m_2
- Open membership



VISA

Open Membership Setting

- Sparsely-connected network (*e.g.*, peer-to-peer)
- New nodes can join/leave (churn)
- Sybil rate limited via PoW

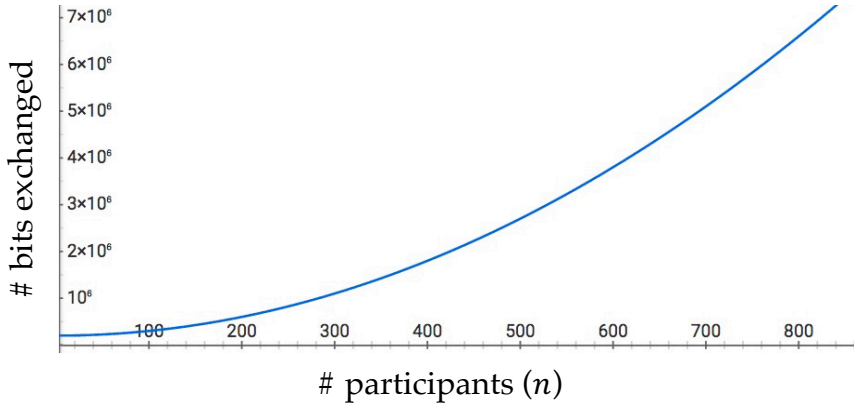


Bitcoin: Too Much of a Good Thing

- Sacrifices *scalability* for *decentralization*
- *Permissioned* blockchain?
  - A closed group of servers run the consensus protocol
  - Membership isn't the bottleneck
- *Full replication* scales out in a cluster but not between datacenters/Internet nodes
  - High round-trip time (~200ms vs <1ms)

Full Replication Doesn't Scale

- Inherently inefficient and unscalable
  - $\Omega(n^2)$ communication and computation needed
  - More nodes, higher message latency
  - Gossiping a 2 MB message to 90% of 4,000 nodes takes ~50 sec



How much redundancy is really *needed*?

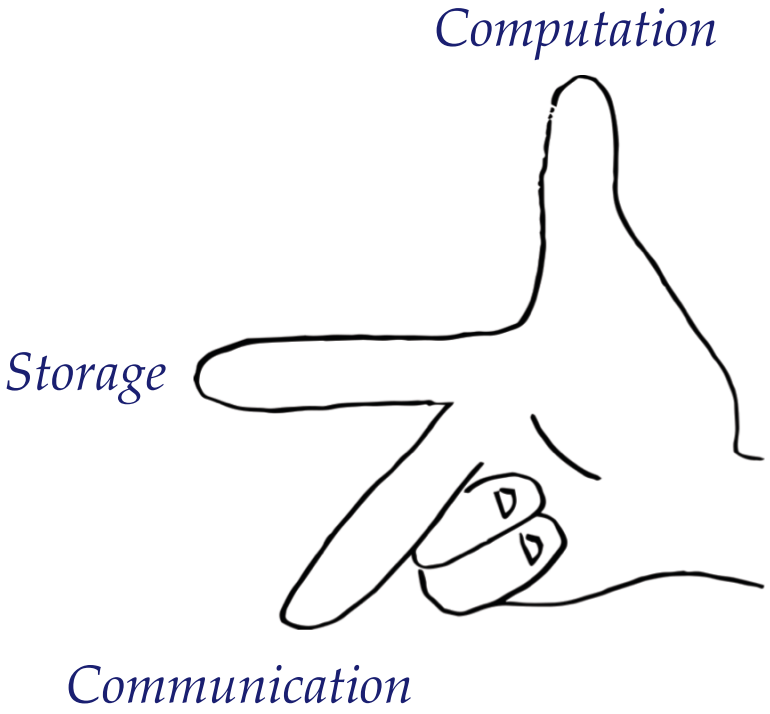
- To tolerate faults and attacks?
  - Bitcoin and Ethereum create 4,000+ replicas

Part II: RapidChain

Sharding-Based Consensus

- Elect a set of ***random committee*** of nodes
- Each committee runs consensus on behalf of all
- And, maintains a ***disjoint ledger***
- Throughput increases linearly with # nodes

Full Sharding



Sharding Challenges

- **Electing *small* committees** *via probabilistic sampling process*
- **Reconfiguration** *to avoid Sybil attack*
- **Cross-shard transactions**
  - Verify transactions located in other committees
- **Decentralized bootstrapping***
  - Establish PKI without initial randomness, CRS, etc.

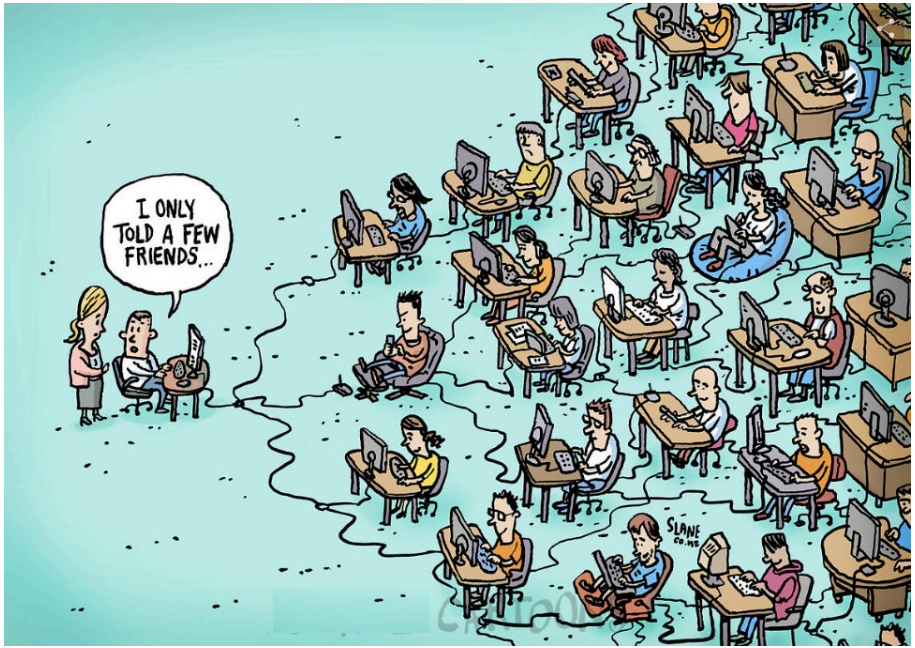
Our Goals/Achievements

- **Higher throughput (7300 tx/sec)**
  - Sublinear communication per tx
  - 6-10x faster fast committee consensus
- **Higher resiliency**
  - RapidChain : $t < n/3$, previous work [LNZ+16, KJG+18]: $t < n/4$
- **Provable reconfiguration**
  - Based on Cuckoo rule [AS06]
- **Decentralized bootstrapping**

  RapidChain: $O(n\sqrt{n})$, previous work: $\Omega(n^2)$

Network Model

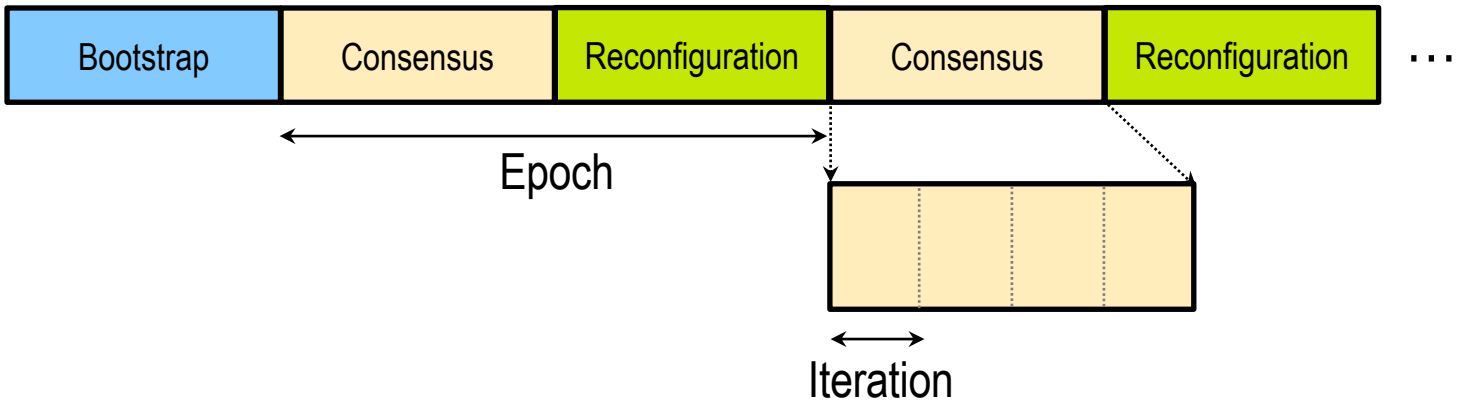
- n nodes each with a key pair (pk_i, sk_i)
- m committees (*a.k.a., shards)*
- P2P network with gossiping
- **Bounded message delay**
  - Known fixed delay, Δ
  - Similar to Bitcoin



Threat Model

- $t < n/3$ **Byzantine nodes** at any moment
- **Slowly-adaptive** adversary [PS16, KJG+18] 5% churn rate

Epochs and Iterations



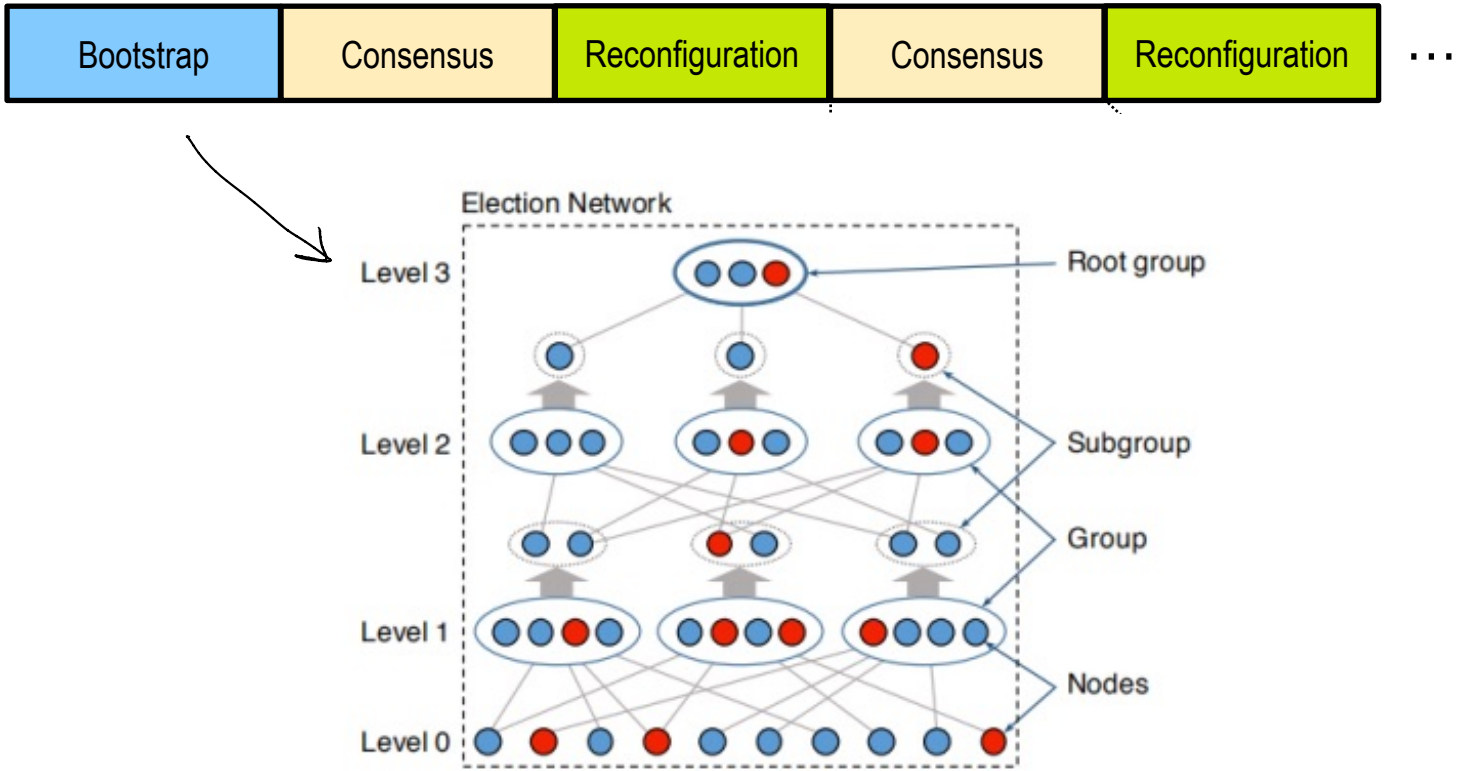
RapidChain Overview

- Proceed in *epochs*
- **First epoch:** Bootstrap a *reference committee*
  - Creates *epoch randomness*
  - Creates a *reconfiguration block* in every epoch
- **Epoch randomness**
  - Samples *sharding committees*
  - New nodes join the system
  - Re-organize existing committees

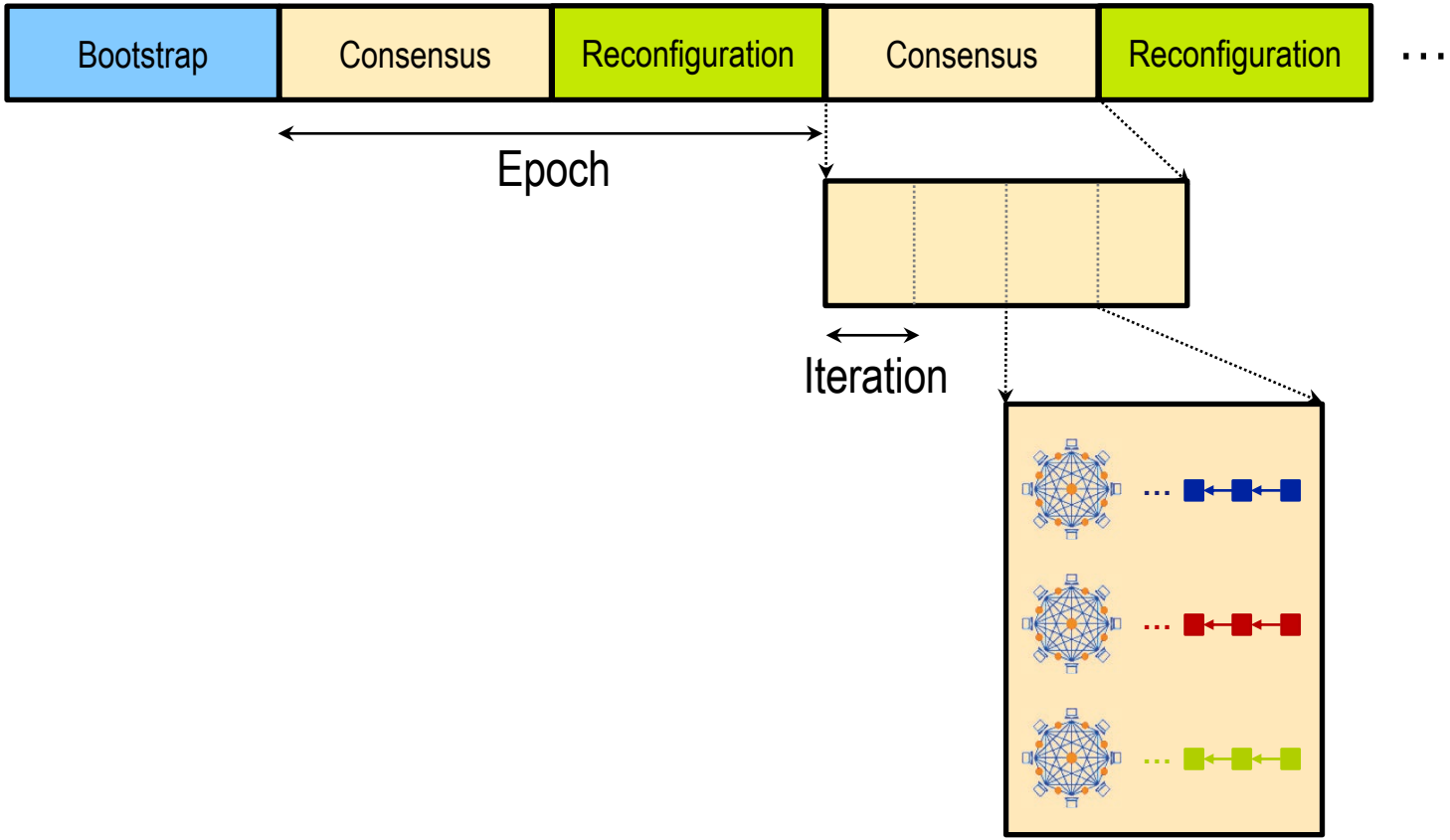
RapidChain Overview (cont'd)

- Each users sends its tx to some arbitrary node
- tx is routed to the *output committee,* C_{out}
- Members of C_{out} create a block
- **Cross-shard verification**
  - Verify input transactions of tx
  - Batch requests based on *input committees*
  - Forward them via an *inter-committee routing* protocol

Sharded Consensus



Sharded Consensus

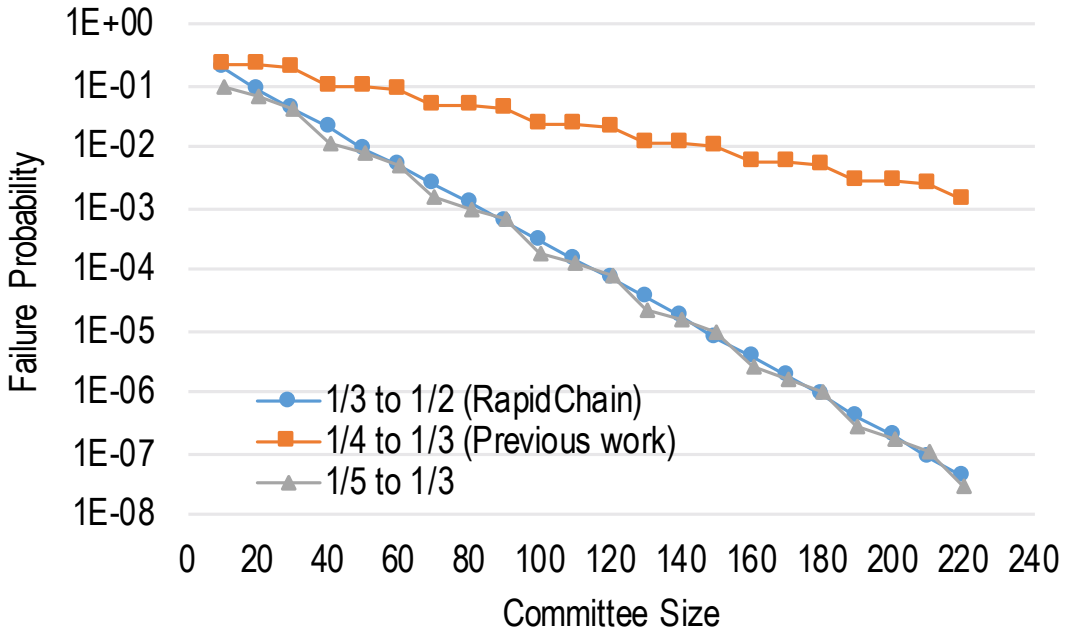


Committee Size

- How large should the committees be?
- Depends on the *intra-committee consensus protocol*
- Higher committee resiliency → Smaller committee
- 4,000 nodes
  - 16 committees of size 250
  - 100 committees of size 40

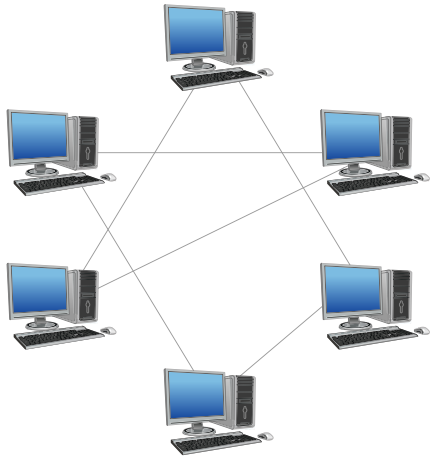
Sampling Error

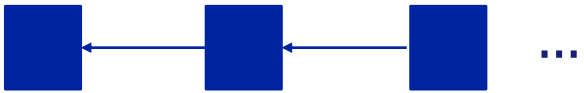
- Using *hypergeometric distribution*



Consensus in Committees

- Gossiping a 2-MB block in a 250-node committee takes ≈ **12 sec**
- *Idea:* Gossip the block, then agree on the hash of the block



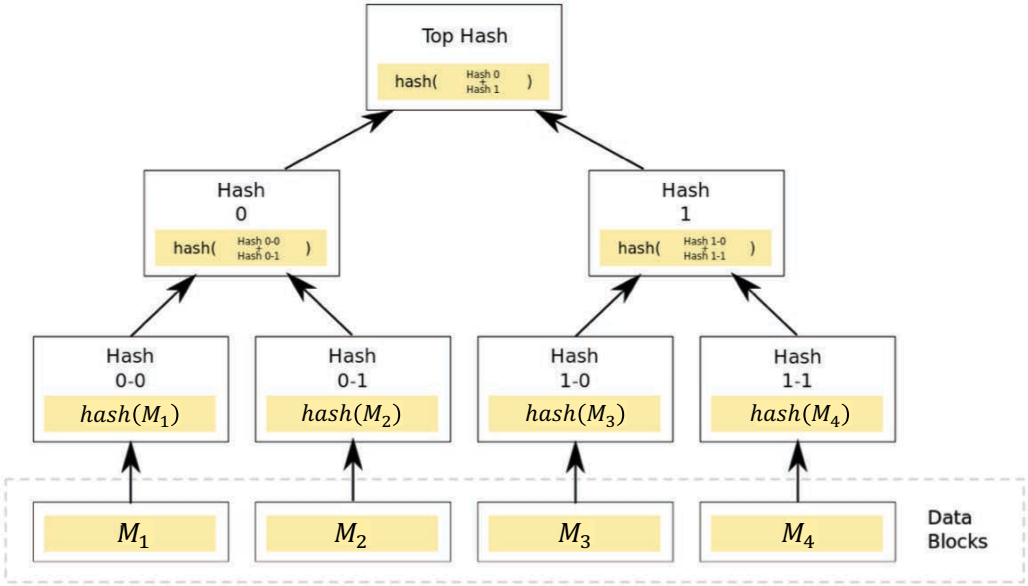


Gossiping Large Blocks

- *Information dispersal algorithm (IDA)* [R89]
  - Encode M into κ chunks $M_1, \ldots, M_\kappa$ using an **erasure coding mechanism**
  - Give each neighbor κ/d chunks (d is the node degree)
  - M can be reconstructed from any set of $(1 - f)\kappa$ chunks
- ECC blowup ≈ **2.7x**
- New gossip time: **2.6 sec**
- Gossip latency reduction ≈ **5.3x**

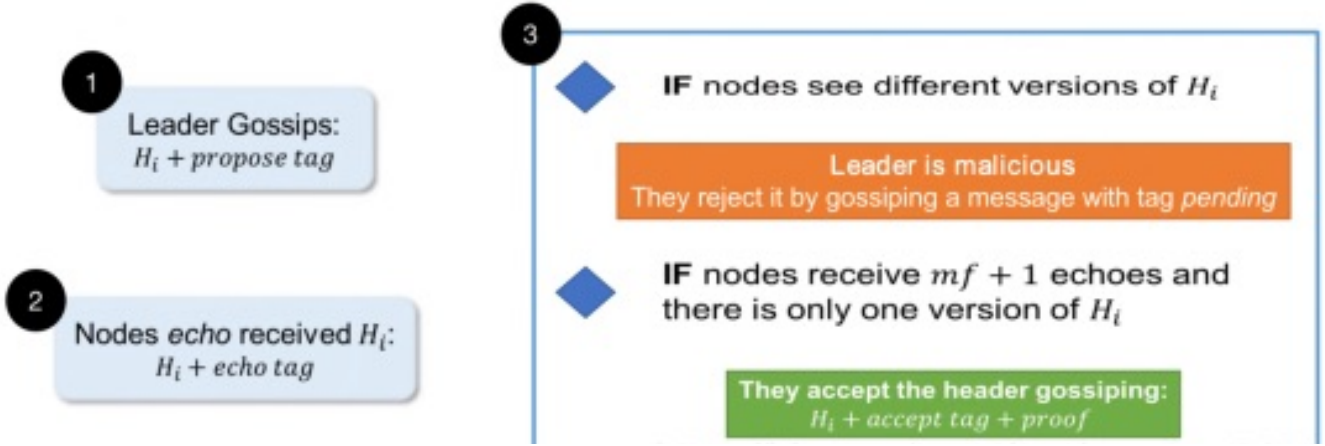
Gossiping Merkle Hash

- Compute a Merkle tree over chunks $M_1, \ldots, M_k$
- Send chunks along with Merkle proofs



Gossiping Merkle Hash

- Compute a Merkle tree over chunks $M_1, \ldots, M_k$
- Send chunks along with Merkle proofs



Cross Shard Transactions

- UTXO and transaction splitting

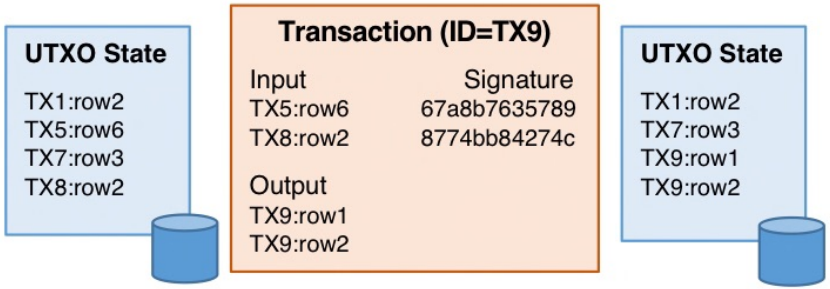
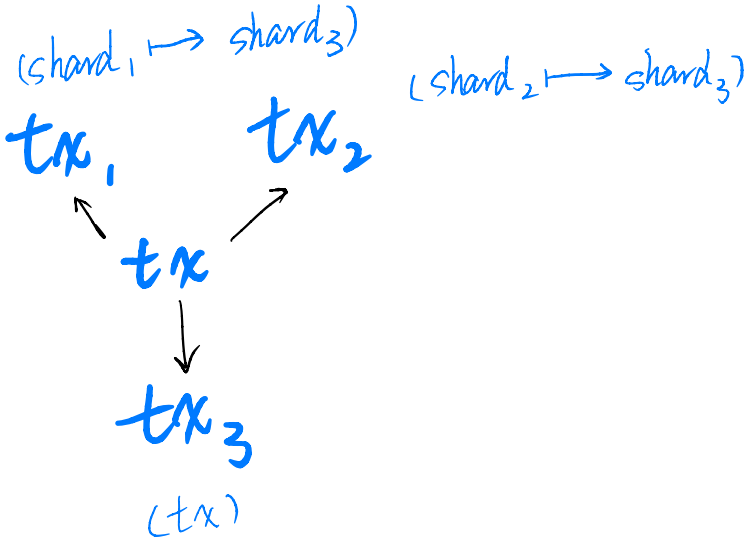
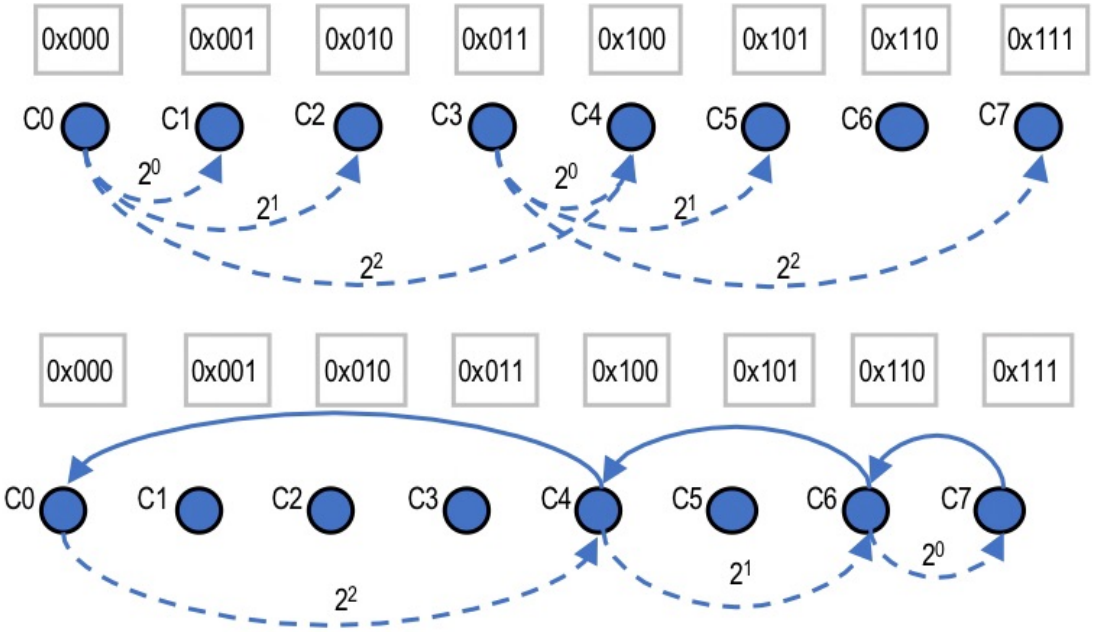


Figure 2: UTXO states before and after a transaction



Inter-committee Routing



Log n committee

Log log n nodes

Handling Join-Leave Attacks

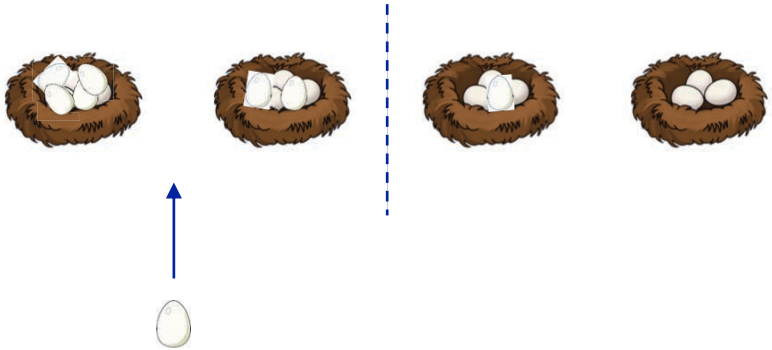
- *Cuckoo rule* [AS06]
  - Assign the new node to a random shard
  - Evict k nodes from the shard
- *Bounded Cuckoo rule*

Handling Join-Leave Attacks

- *Cuckoo rule* [AS06]
  - Assign the new node to a random shard
  - Evict k nodes from the shard
- *Bounded Cuckoo rule*

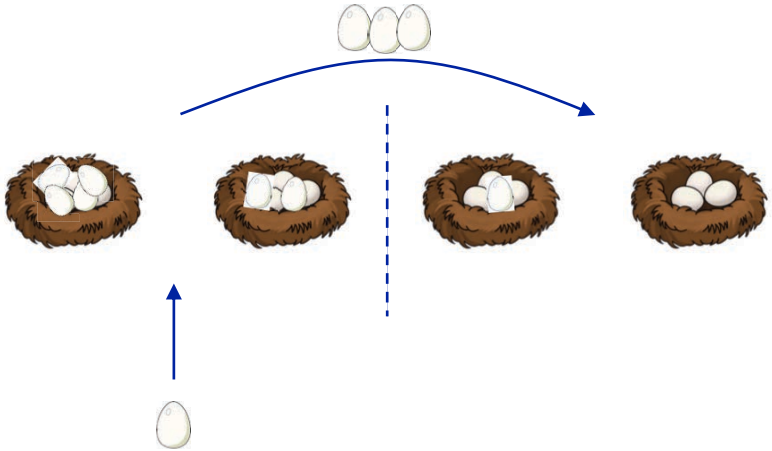
Handling Join-Leave Attacks

- *Cuckoo rule* [AS06]
  - Assign the new node to a random shard
  - Evict k nodes from the shard
- *Bounded Cuckoo rule*



Handling Join-Leave Attacks

- *Cuckoo rule* [AS06]
  - Assign the new node to a random shard
  - Evict k nodes from the shard
- *Bounded Cuckoo rule*

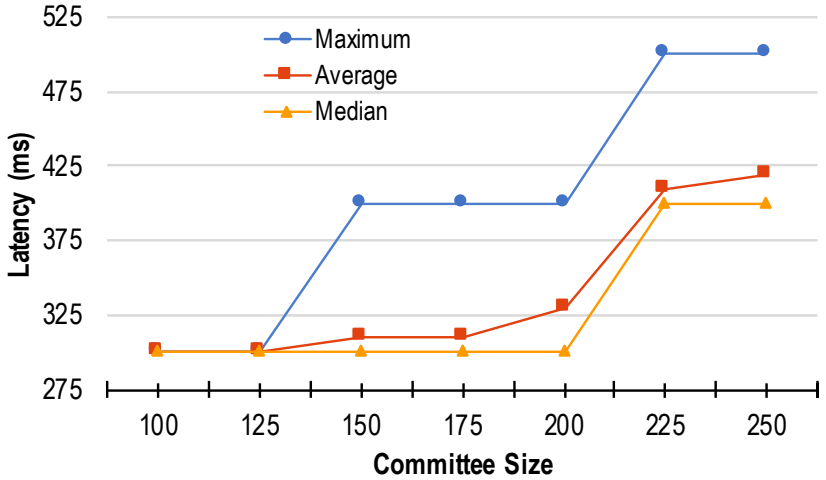


Experimental Setup

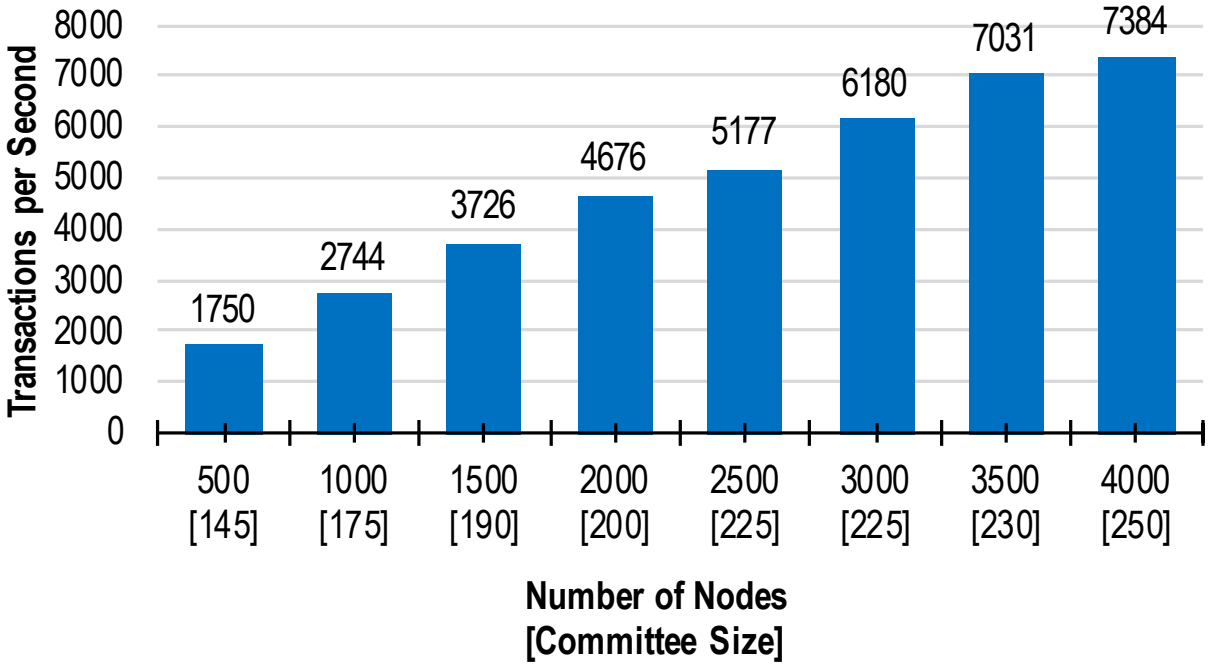
- Prototype implemented in Go
- Link latency of **100 *ms***
- Node bandwidth of **20 Mbps**
- **2 MB** blocks (**4096 tx/block**, **512 B/tx**)
- Corruption model
  - 49% of leaders equivocate in the same iteration
  - 49% of nodes do not participate in gossips (*i.e.*, remain silent)

Synchronous Round Time-Out

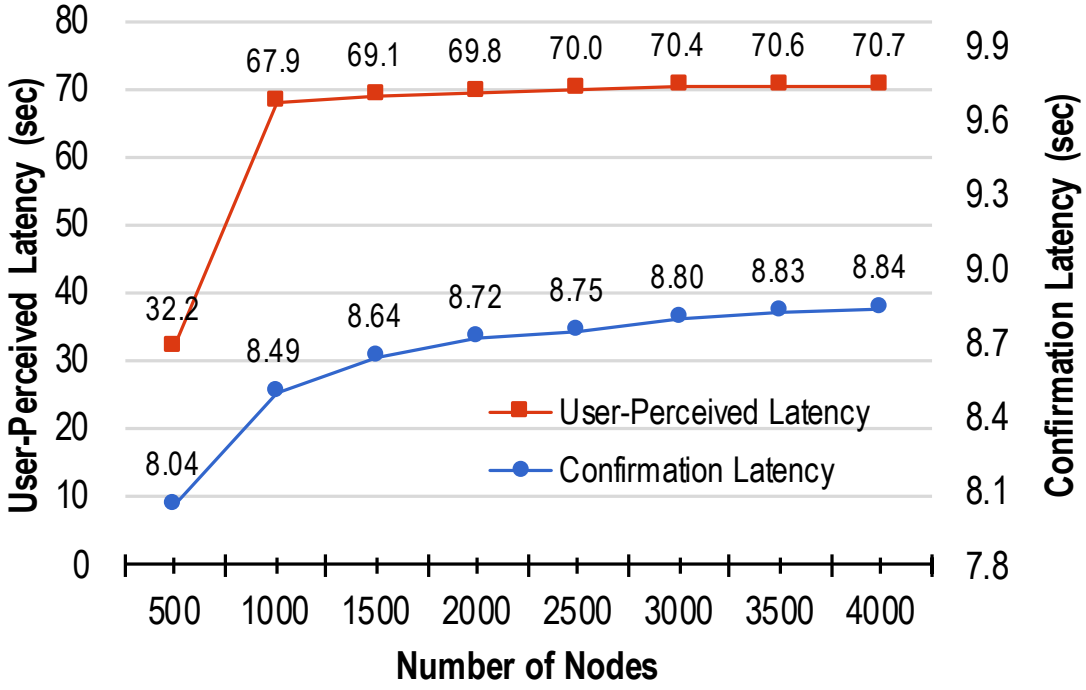
- $\Delta = 600\ ms$
- Latency of gossiping an 80-byte message in a committee of size 250



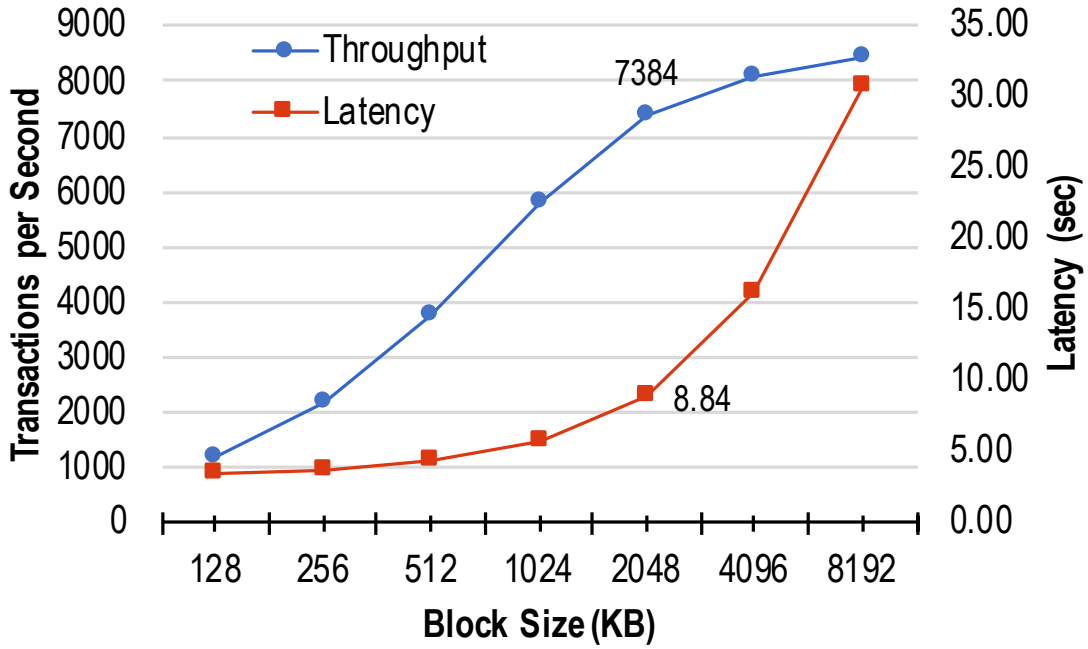
Throughput vs *n*



Latency vs *n*


User-Perceived Latency (sec)
Confirmation Latency (sec)
Number of Nodes
0
10
20
30
40
50
60
70
80
7.8
8.1
8.4
8.7
9.0
9.3
9.6
9.9
500
1000
1500
2000
2500
3000
3500
4000
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

Impact of Block Size



Where do we stand?

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*

References

- [PSL80] Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. In JACM, 1980.
- [BT83] Gabriel Bracha and Sam Toueg. Resilient consensus protocols. In PODC, 1983.
- [HT93] Vassos Hadzilacos and Sam Toueg. Distributed systems. Chapter on Fault-tolerant Broadcasts and Related Problems, ACM Press, 1993.
- [PSS17] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In Advances in Cryptology – EUROCRYPT 2017, pages 643–673. Springer International Publishing, 2017.
- [R86] Michael O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. In JACM, 1989.
- [KJG+18] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. OmniLedger: A secure, scale-out, decentralized ledger via sharding. In S&P, 2018.
- [LNZ+16] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. A secure sharding protocol for open blockchains. In CCS, 2016.
- [PS16] Rafael Pass and Elaine Shi. Hybrid consensus: Efficient consensus in the permissionless model. Cryptology ePrint Archive, Report 2016/917, 2016.
- [RNA+17] Ling Ren, Kartik Nayak, Ittai Abraham, and Srinivas Devadas. Practical synchronous byzantine consensus. CoRR , abs/1704.02397, 2017.
- [AS06] Baruch Awerbuch and Christian Scheideler. Towards a scalable and robust DHT. In SPAA, 2006.