OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding

Philipp Jovanovic (@daeinar)

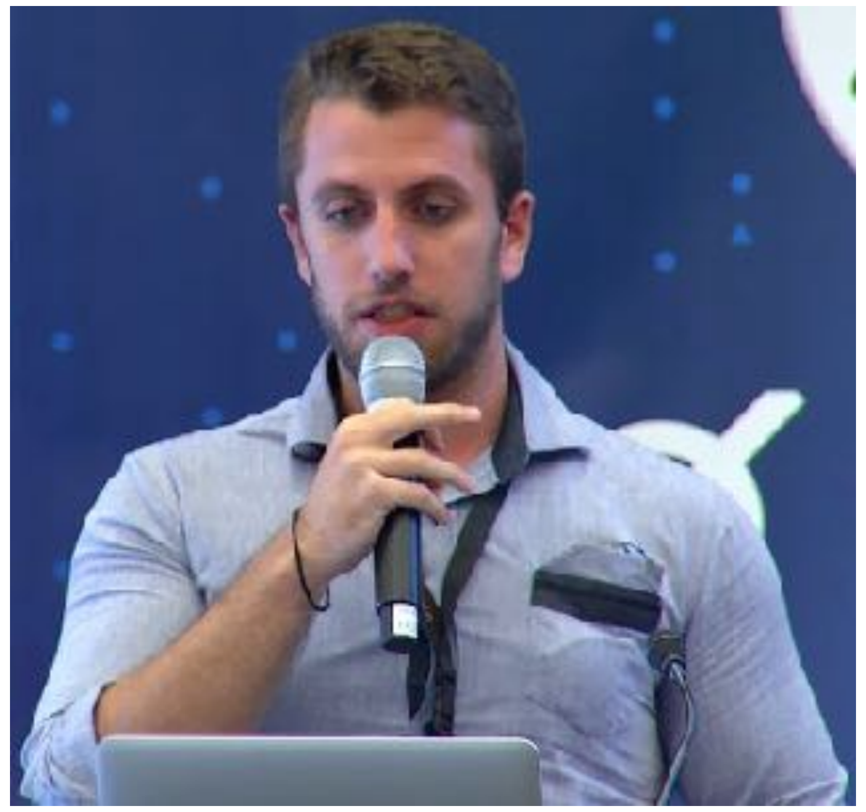
Distributed and Decentralized Systems Lab (DEDIS)

École polytechnique fédérale de Lausanne (EPFL)

IEEE International Verification and Security Workshop

2018-07-04, Platja D'Aro

Acknowledgements



Eleftherios Kokoris Kogias
(EPFL, CH)

Nicolas Gailly
(EPFL, CH)

Linus Gasser
(EPFL, CH)

Ewa Syta
(Trinity College, USA)

Bryan Ford
(EPFL, CH)

Talk Outline

- Motivation
- OmniLedger
- Evaluation
- Conclusion

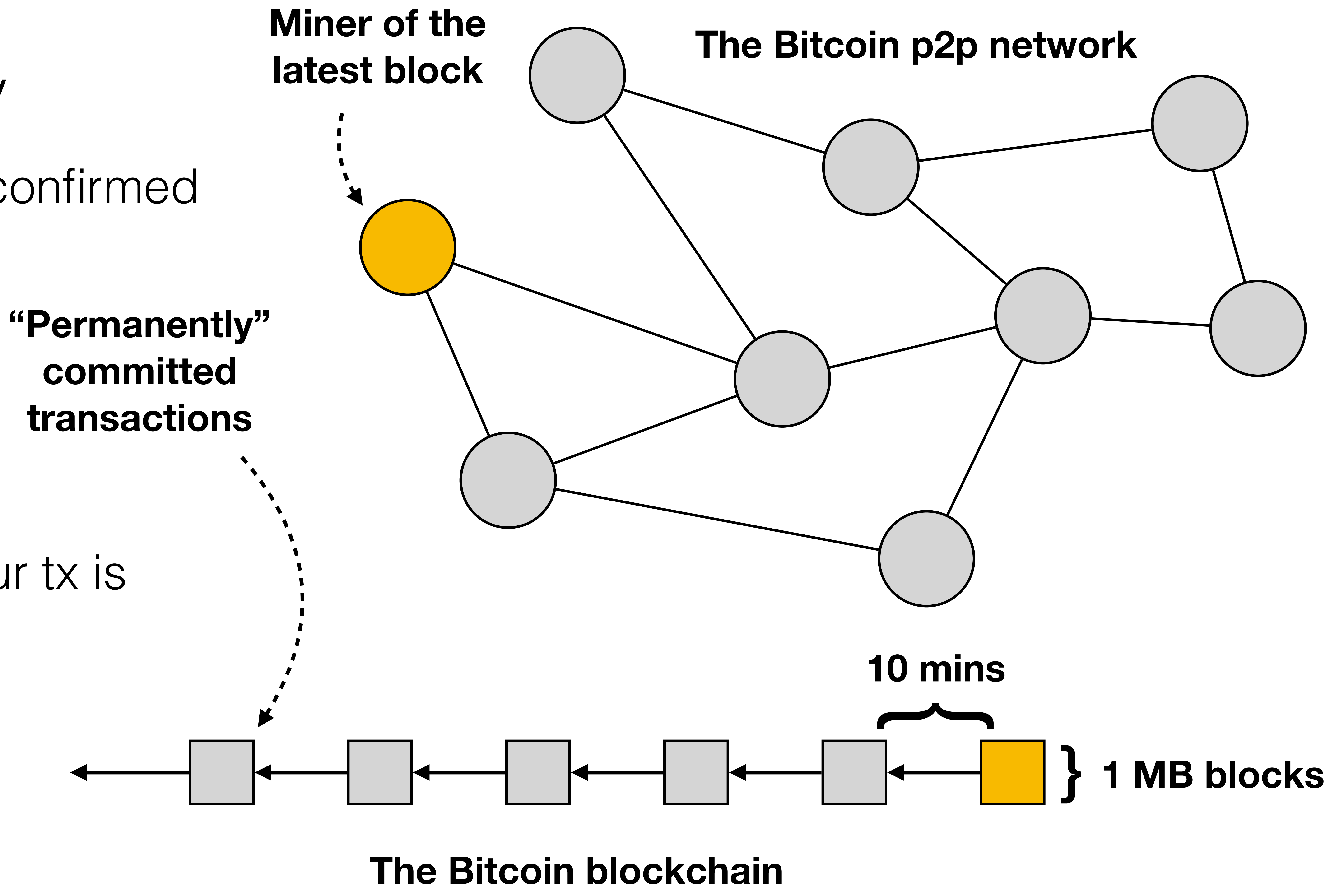
Talk Outline

- **Motivation**
- OmniLedger
- Evaluation
- Conclusion

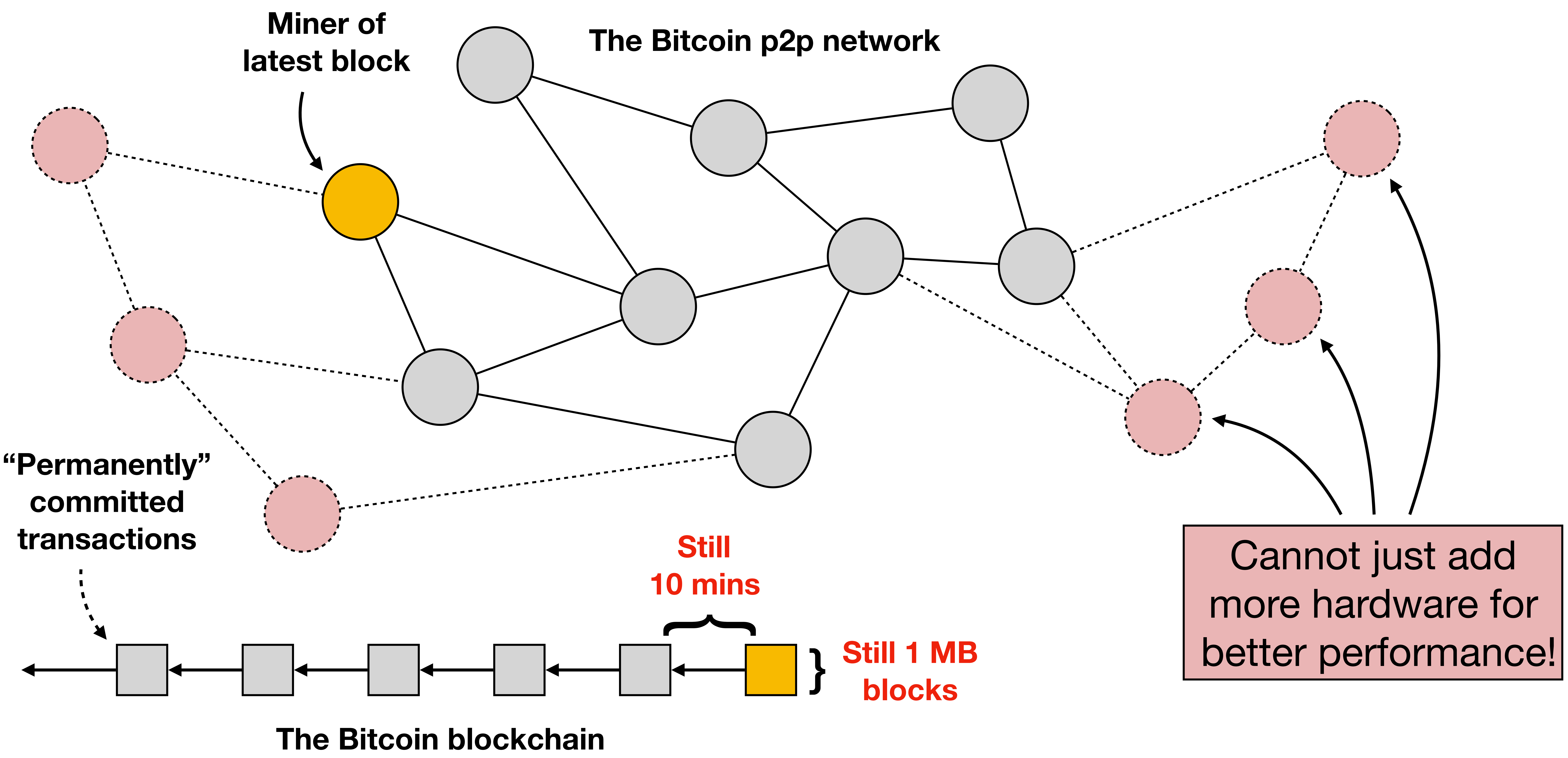
The Core of Bitcoin: Nakamoto Consensus

Drawbacks

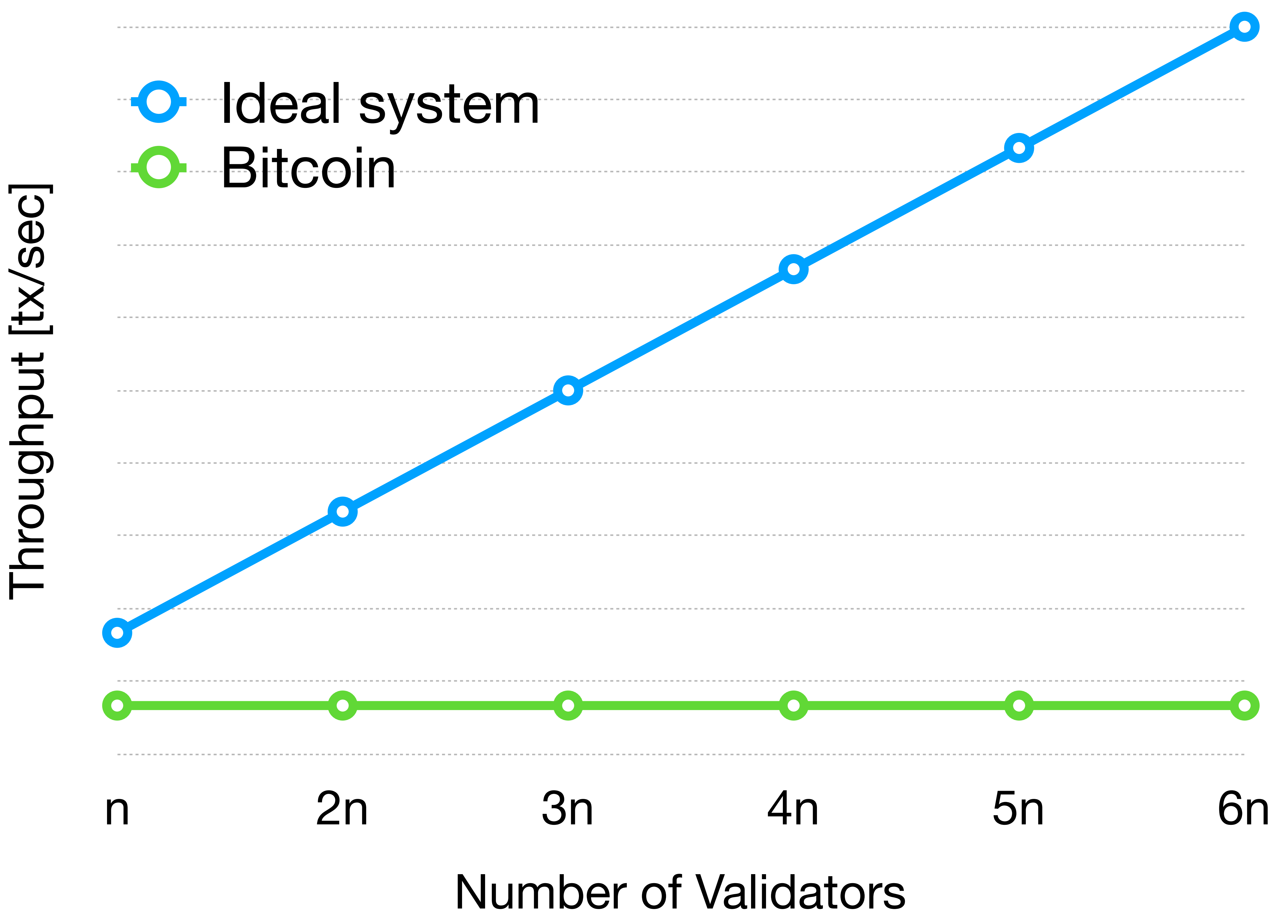
- Transaction confirmation delay
  - Bitcoin: Any tx takes >10 mins until confirmed
- Low throughput
  - Bitcoin: ~4 tx/sec
- Weak consistency
  - Bitcoin: You are not really certain your tx is committed until you wait >1 hour
- Proof-of-work mining
  - Wastes huge amount of energy



… But Scaling Blockchains is Not Easy



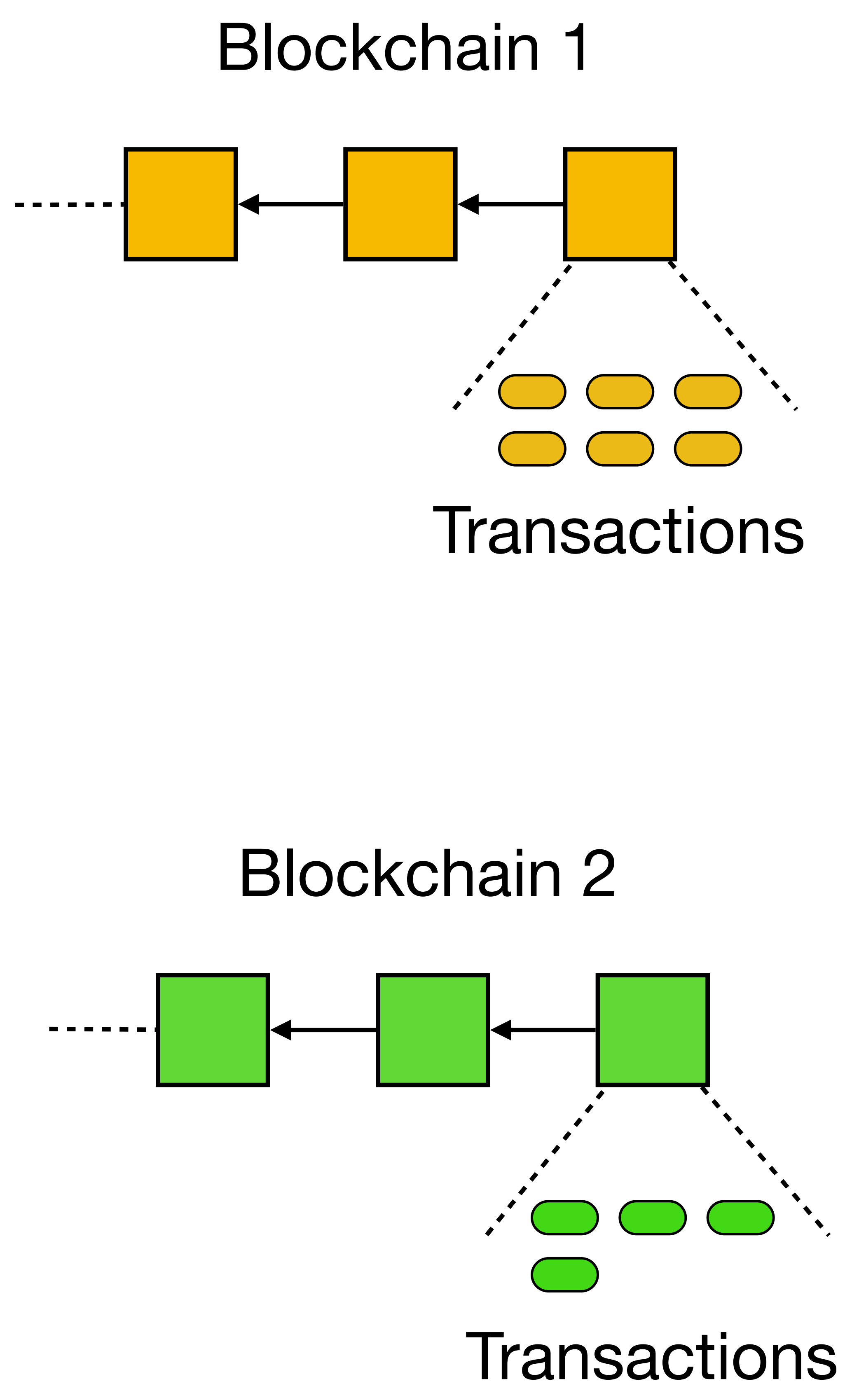
What we Want: Scale-Out Performance



Scale-out: Throughput increases *linearly* with the available resources.

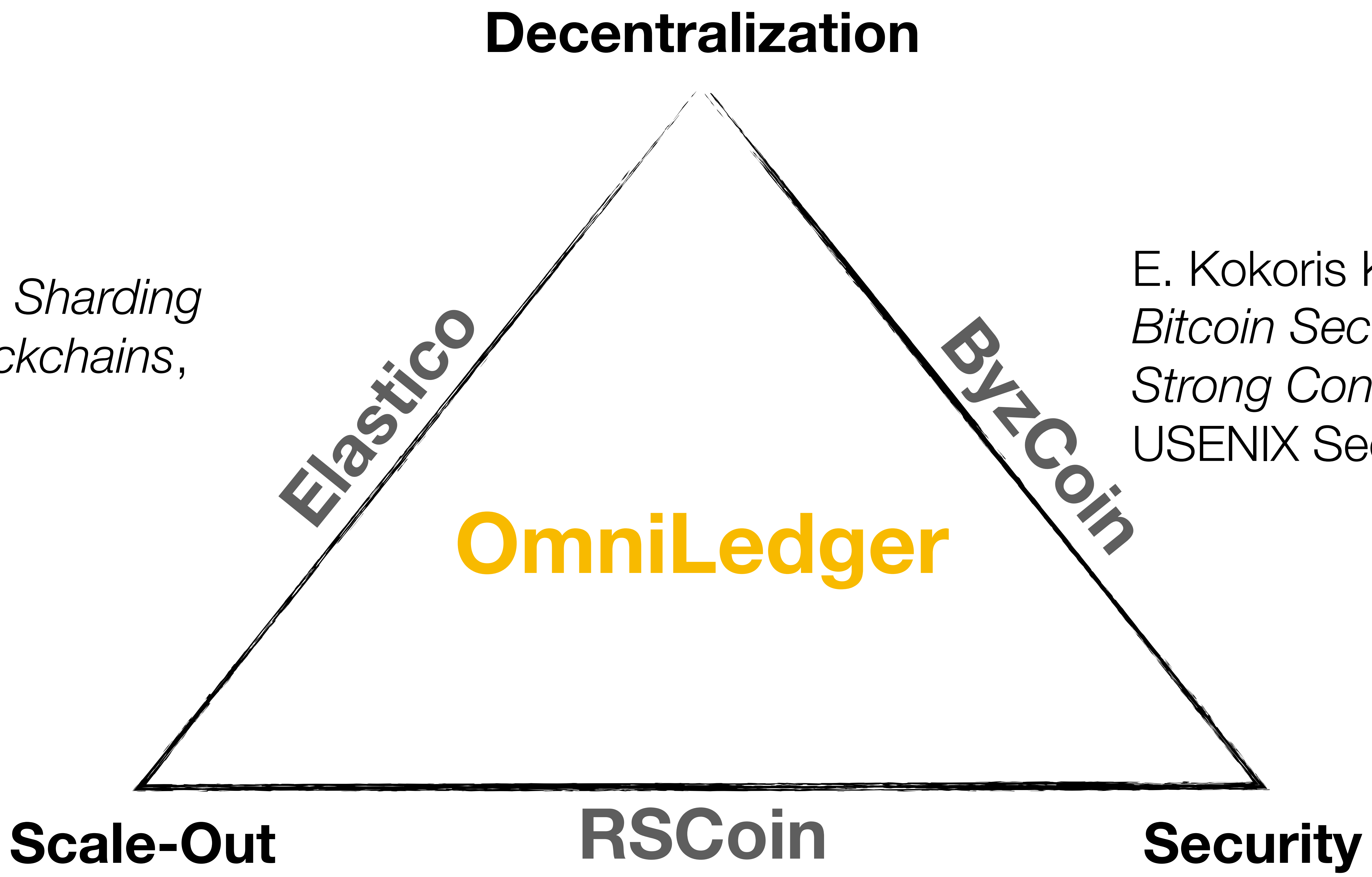
Towards Scale-Out Performance via Sharding

- **Concept:**
  - Validators are grouped into distinct subsets
  - Each subset processes different transactions
  - Achieves parallelization and therefore scale-out
- **But:**
  - How to assign validators to shards?
  - How to send transactions across shards?



Distributed Ledger Landscape

L. Luu et al., *A Secure Sharding Protocol for Open Blockchains*, CCS 2016



E. Kokoris Kogias et al., *Enhancing Bitcoin Security and Performance with Strong Consistency via Collective Signing*, USENIX Security 2016

G. Danezis and S. Meiklejohn, *Centrally Banked Cryptocurrencies*, NDSS 2016

Talk Outline

- Motivation
- **OmniLedger**
- Evaluation
- Conclusion

OmniLedger – Design Goals

Security Goals

1. Full Decentralization
No trusted third parties or single points of failure

2. Shard Robustness
Shards process txs correctly and continuously

3. Secure Transactions
Txs commit atomically or abort eventually

Performance Goals

4. Scale-out
Throughput increases linearly in the number of active validators

5. Low Storage
Validators do not need to store the entire shard tx history

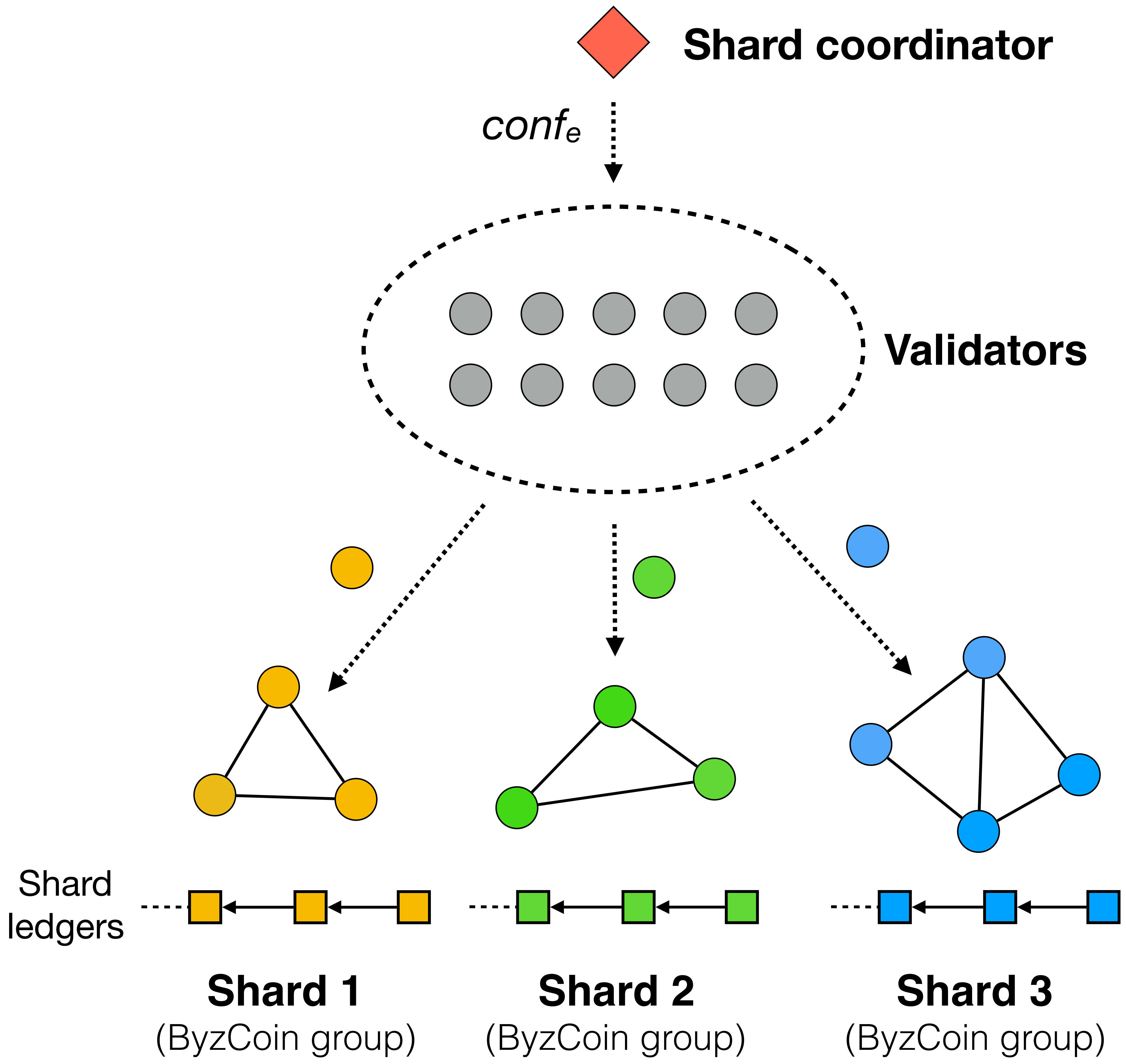
6. Low Latency
Tx are confirmed quickly

Assumptions: <= 25% mildly adaptive Byzantine adversary, (partially) synchronous network, UTXO model

Strawman: SimpleLedger

Overview

- Evolves in epochs *e*
- Trusted source releases shard configuration $conf_e$
- Validators:
    - Bootstrap from the shard ledger according to $conf_e$
    - Process transactions in parallel using per-shard consensus



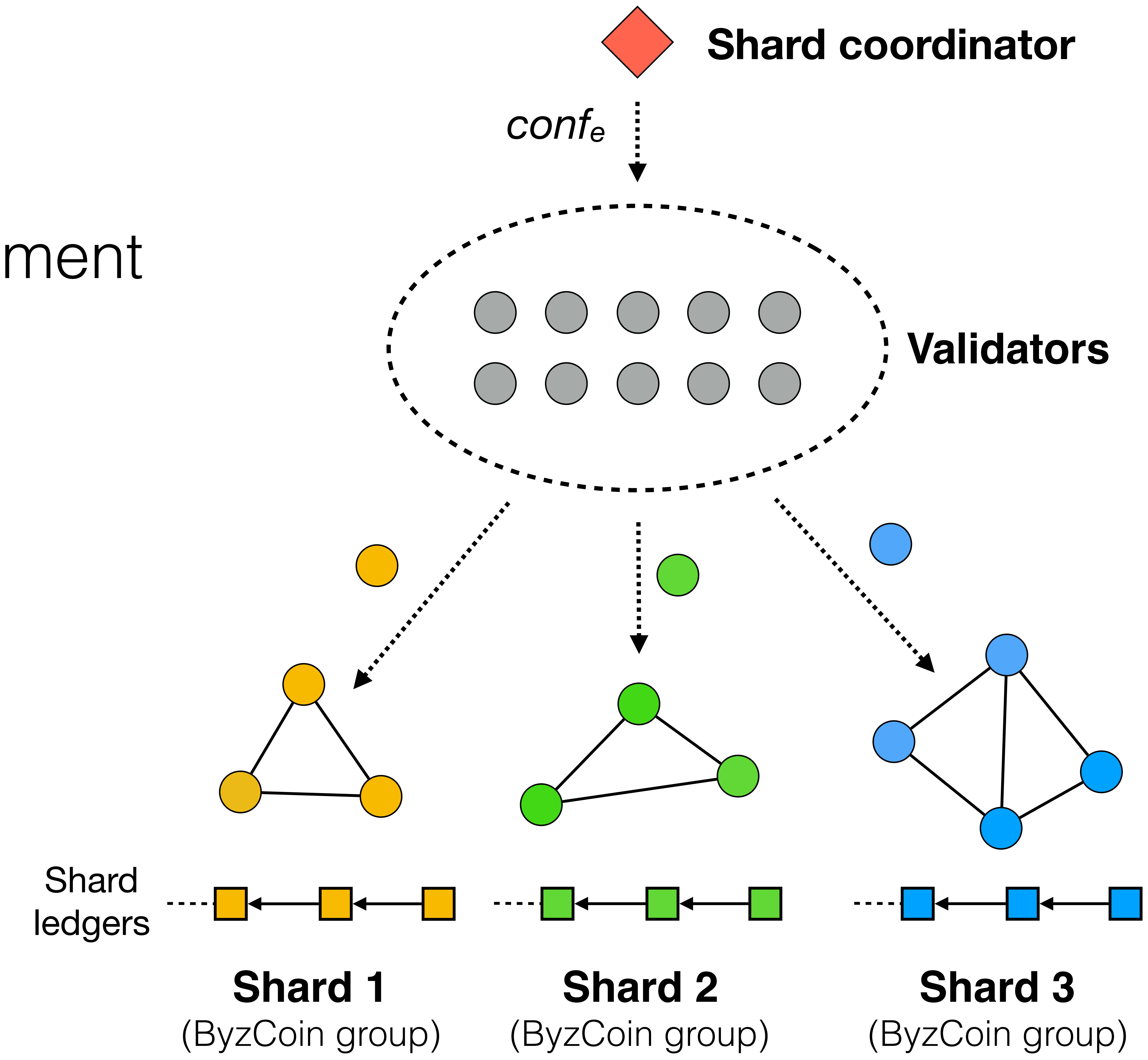
Strawman: SimpleLedger

Security Drawbacks

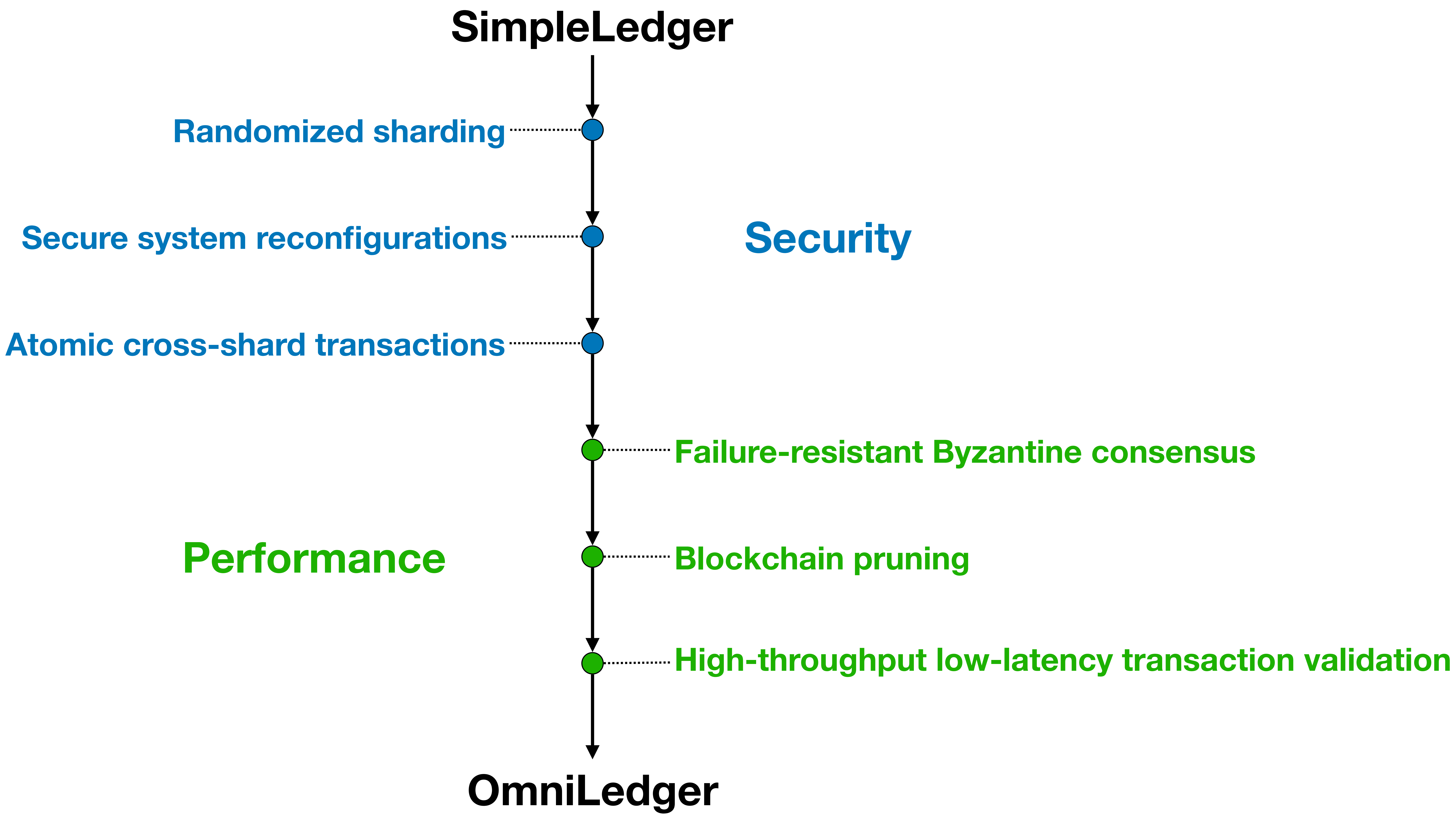
- Shard coordinator: trusted third party
- No tx processing during validator re-assignment
- No cross-shard tx support

Performance Drawbacks

- ByzCoin failure mode
- High storage and bootstrapping cost
- Throughput vs. latency trade-off



Roadmap


SimpleLedger
Randomized sharding
Secure system reconfigurations
Security
Atomic cross-shard transactions
Failure-resistant Byzantine consensus
Performance
Blockchain pruning
High-throughput low-latency transaction validation
OmniLedger

Roadmap

SimpleLedger

Randomized sharding

Secure system reconfigurations

Atomic cross-shard transactions

Security

Failure-resistant Byzantine consensus

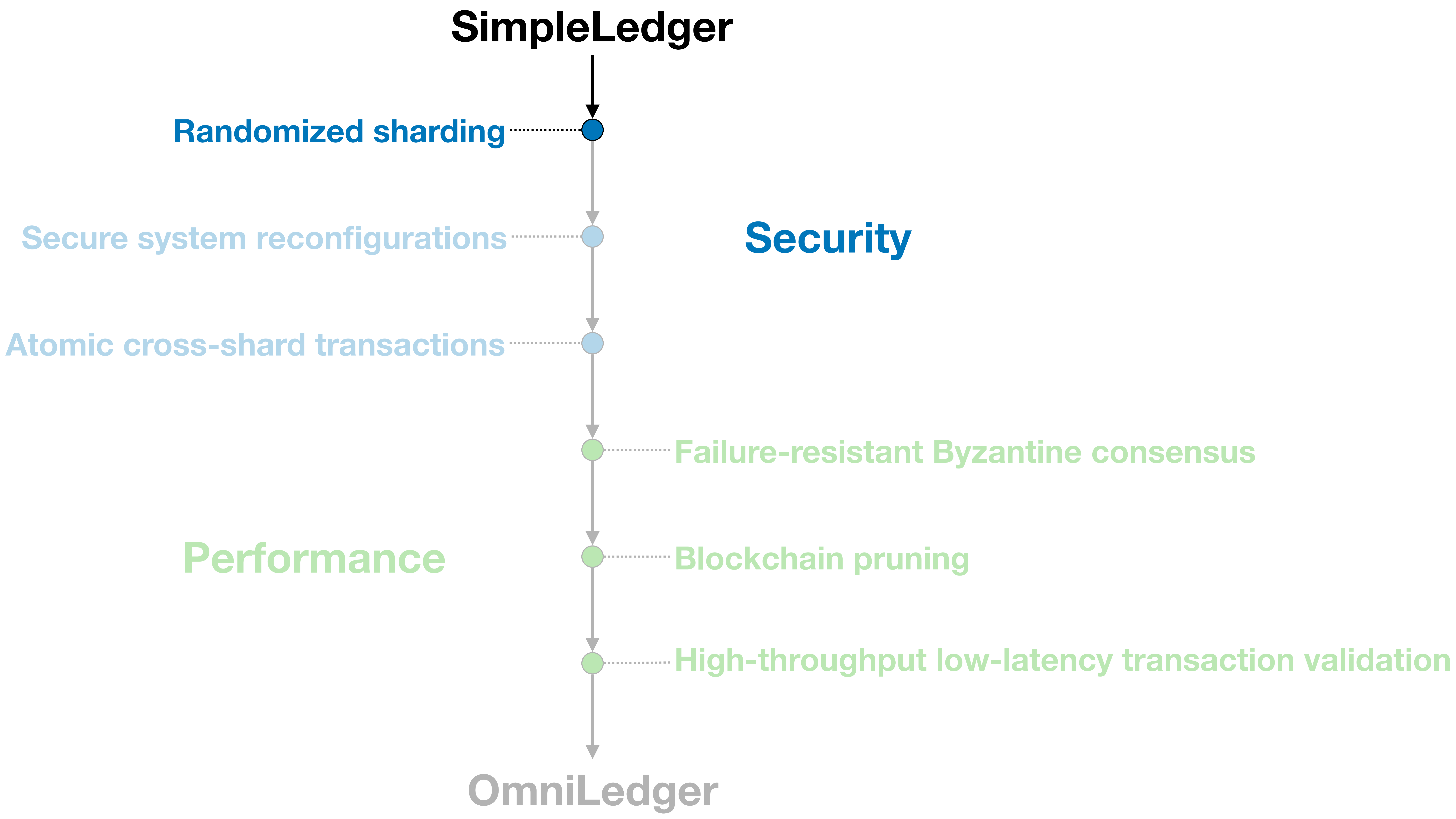
Blockchain pruning

High-throughput low-latency transaction validation

Performance

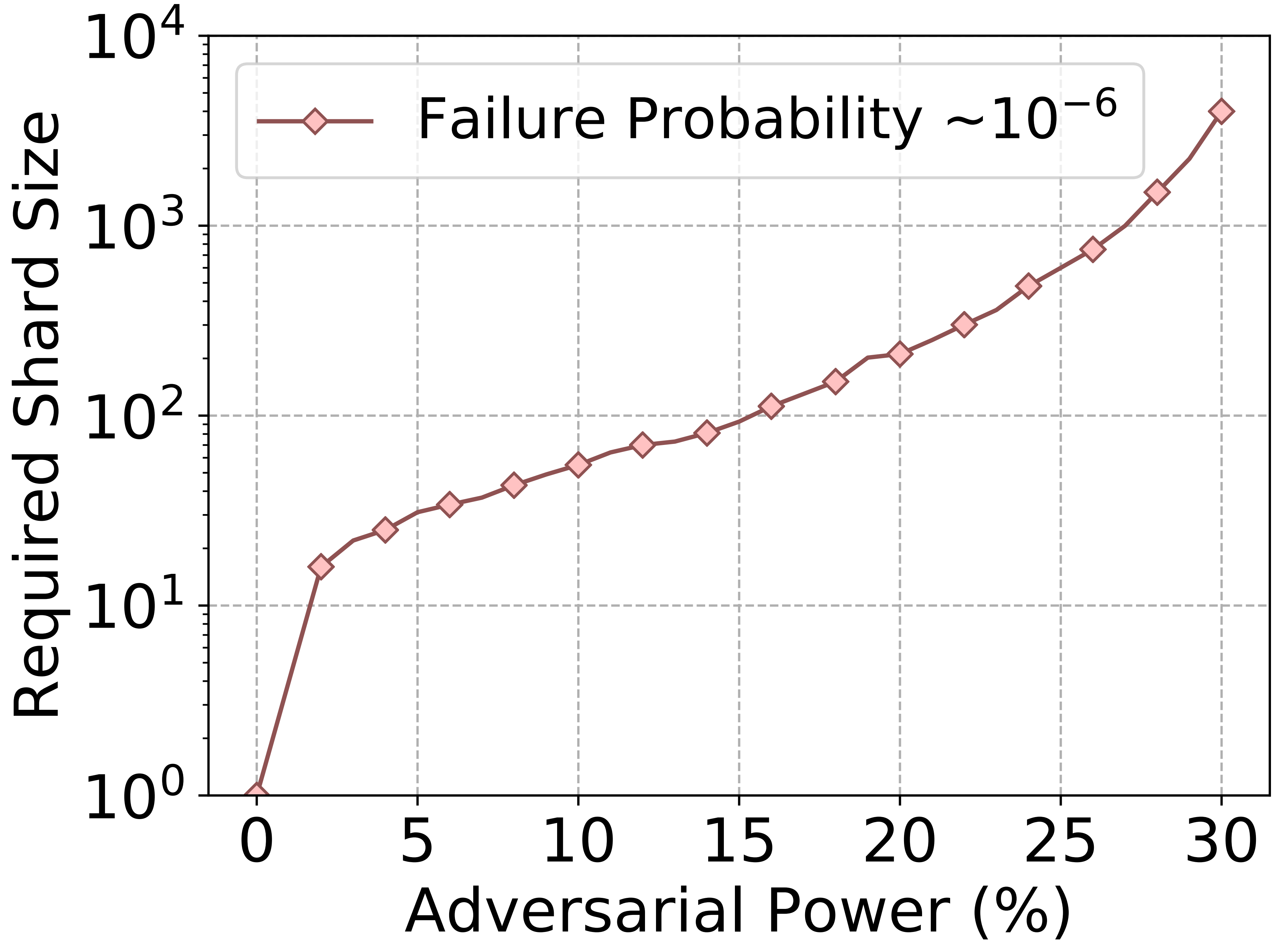
OmniLedger

Roadmap


SimpleLedger
Randomized sharding
Secure system reconfigurations
Security
Atomic cross-shard transactions
Failure-resistant Byzantine consensus
Performance
Blockchain pruning
High-throughput low-latency transaction validation
OmniLedger

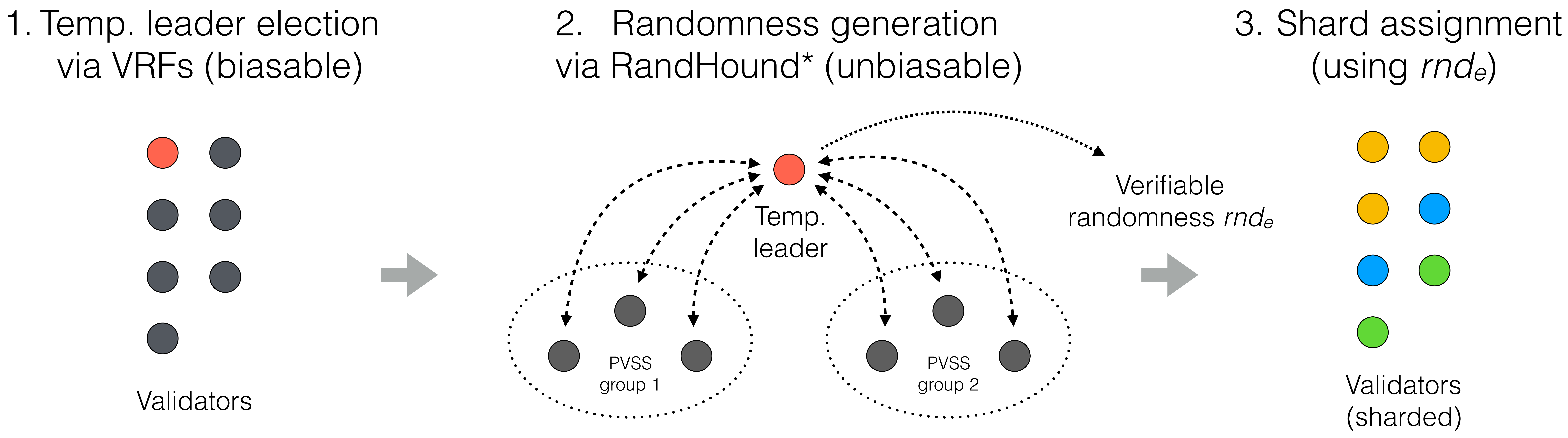
Shard Validator Assignment

- **How to assign validators to shards?**
  - Deterministically: Adversary can use predictable assignments to his advantage
  - Randomly: Adversary cannot control or predict assignment
- **How to ensure long-term shard security against an adaptive adversary?**
  - Make shards large enough
  - Periodically re-assign validators to shards



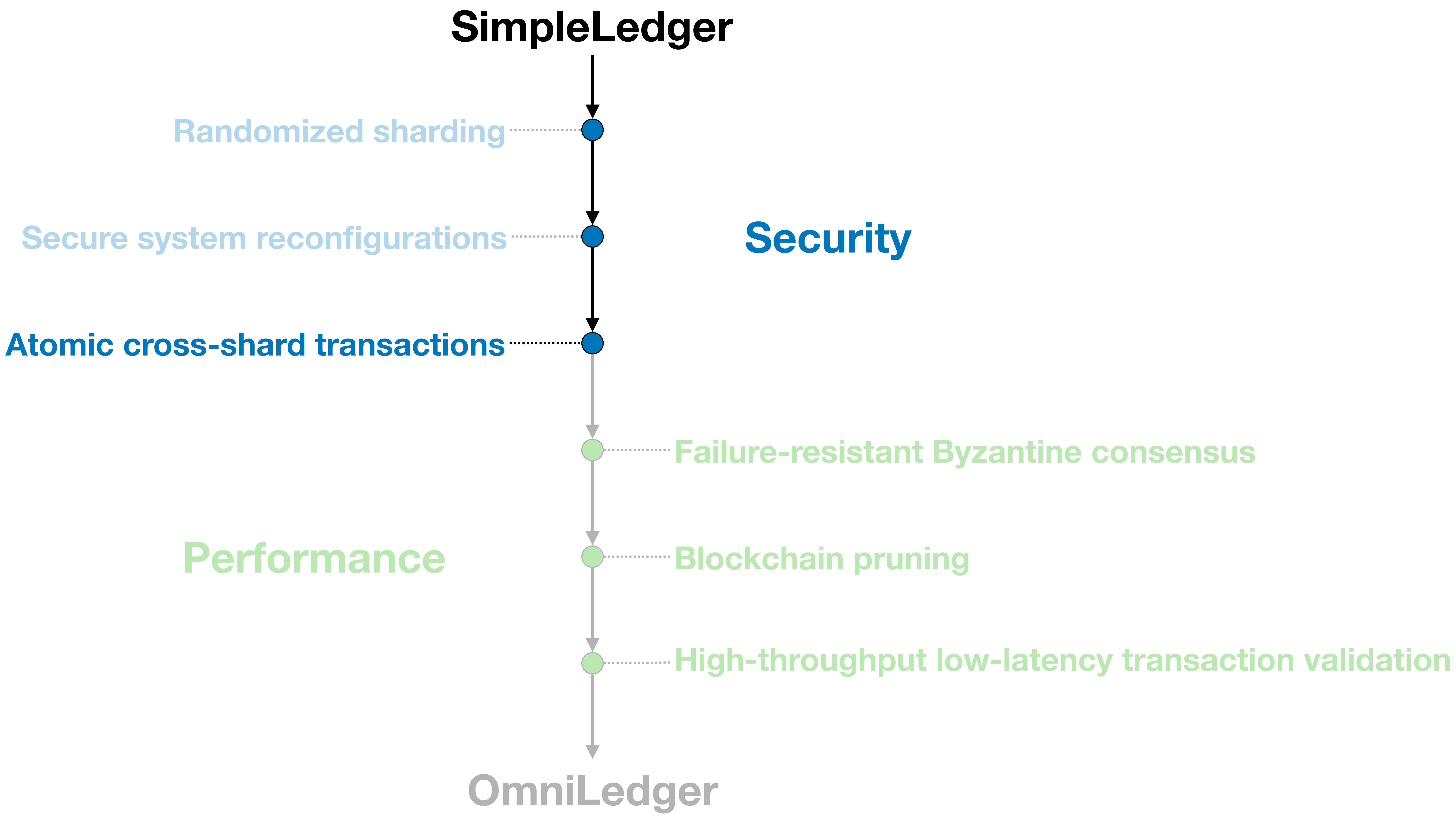
Shard Validator Assignment

- **Challenge:** Unbiasable, unpredictable and scalable shard validator assignment
- **Solution:** Combine VRF-based lottery and unbiasable randomness protocol for sharding

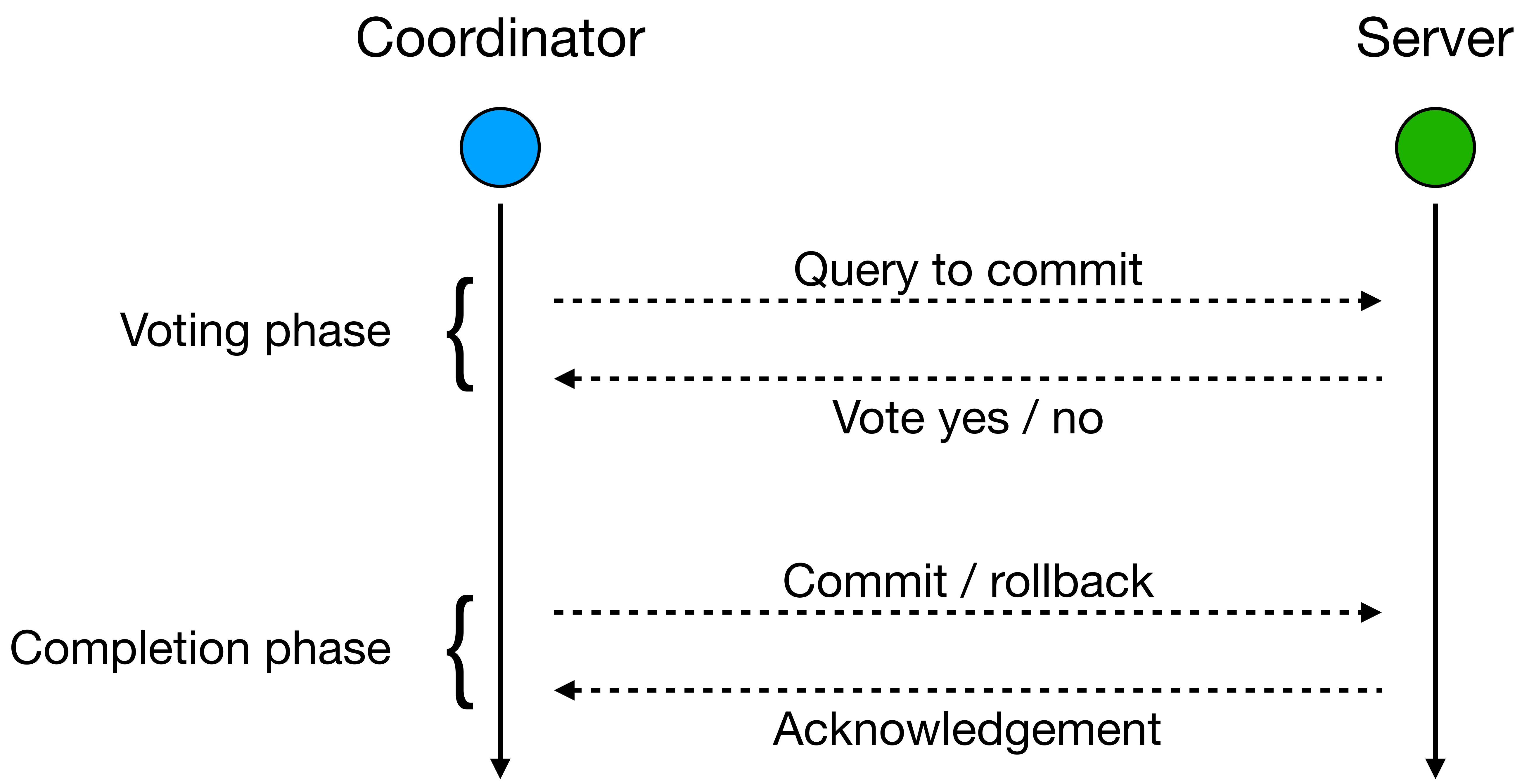


*Scalable Bias-resistant Distributed Randomness, E. Syta et al., IEEE S&P'17

Roadmap


SimpleLedger
Randomized sharding
Secure system reconfigurations
Security
Atomic cross-shard transactions
Failure-resistant Byzantine consensus
Performance
Blockchain pruning
High-throughput low-latency transaction validation
OmniLedger

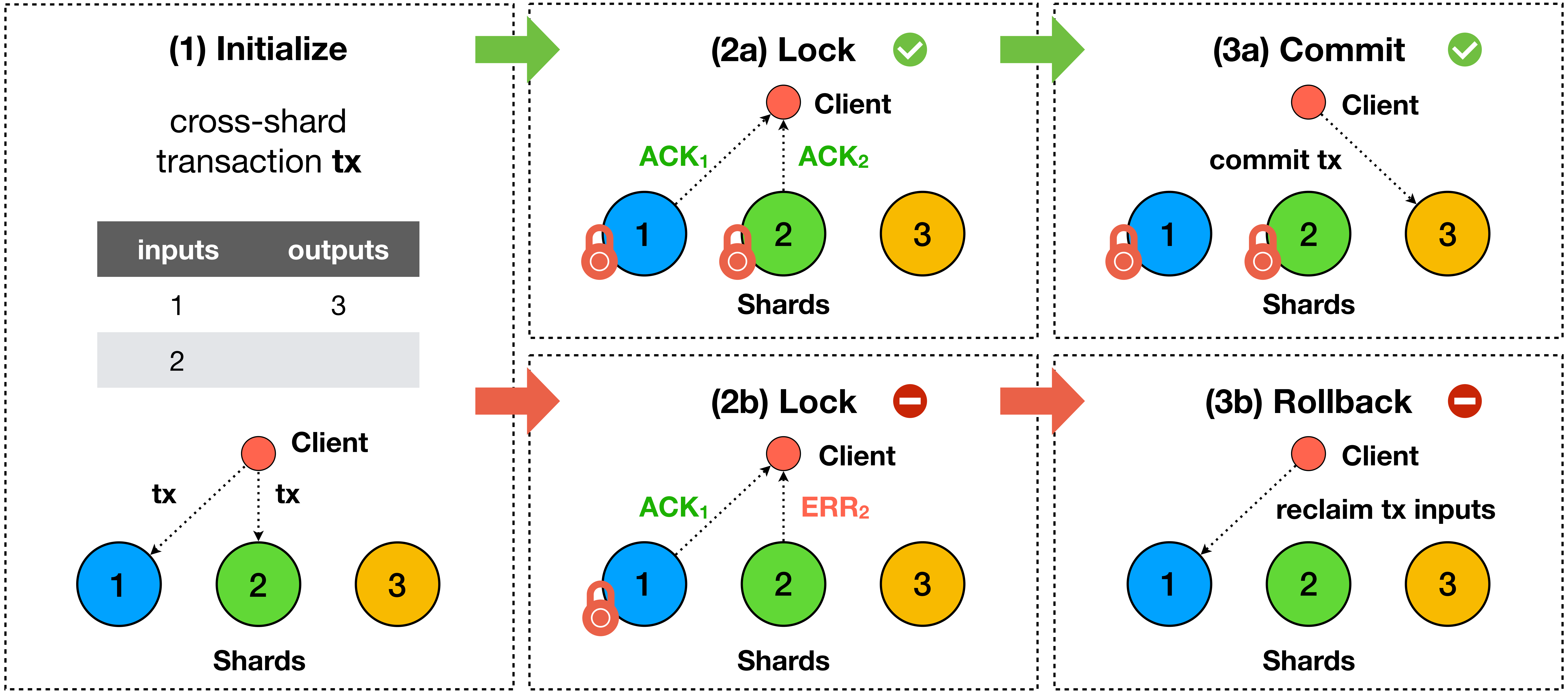
Two-Phase Commits



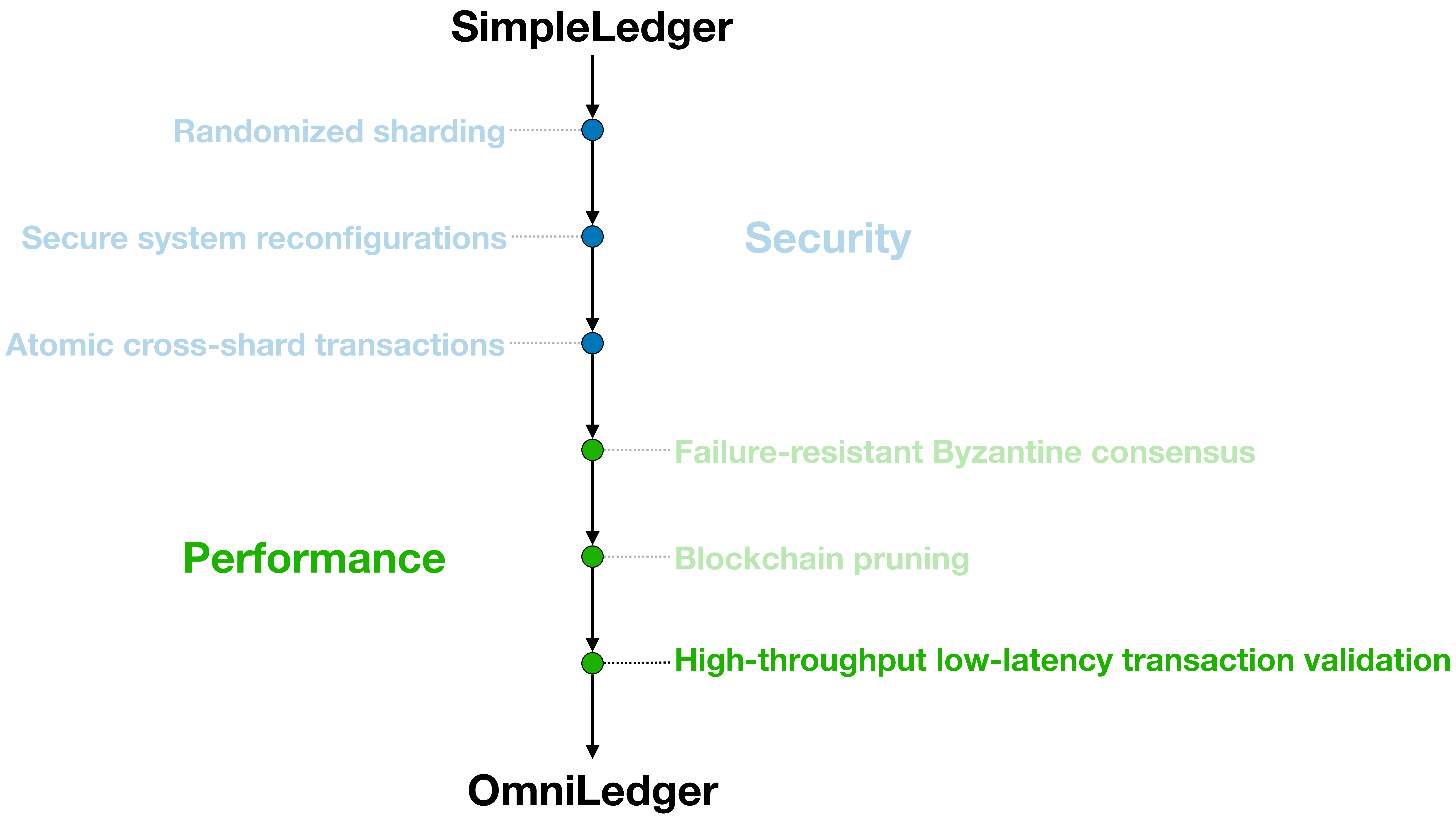
Problem: Does not work in a Byzantine setting as malicious nodes can always abort.

Atomix: Secure Cross-Shard Transactions

- **Challenge:** Cross-shard transactions commit atomically or abort eventually
- **Solution:** Atomix, a secure cross-shard transaction protocol (utilizing secure BFT shards)

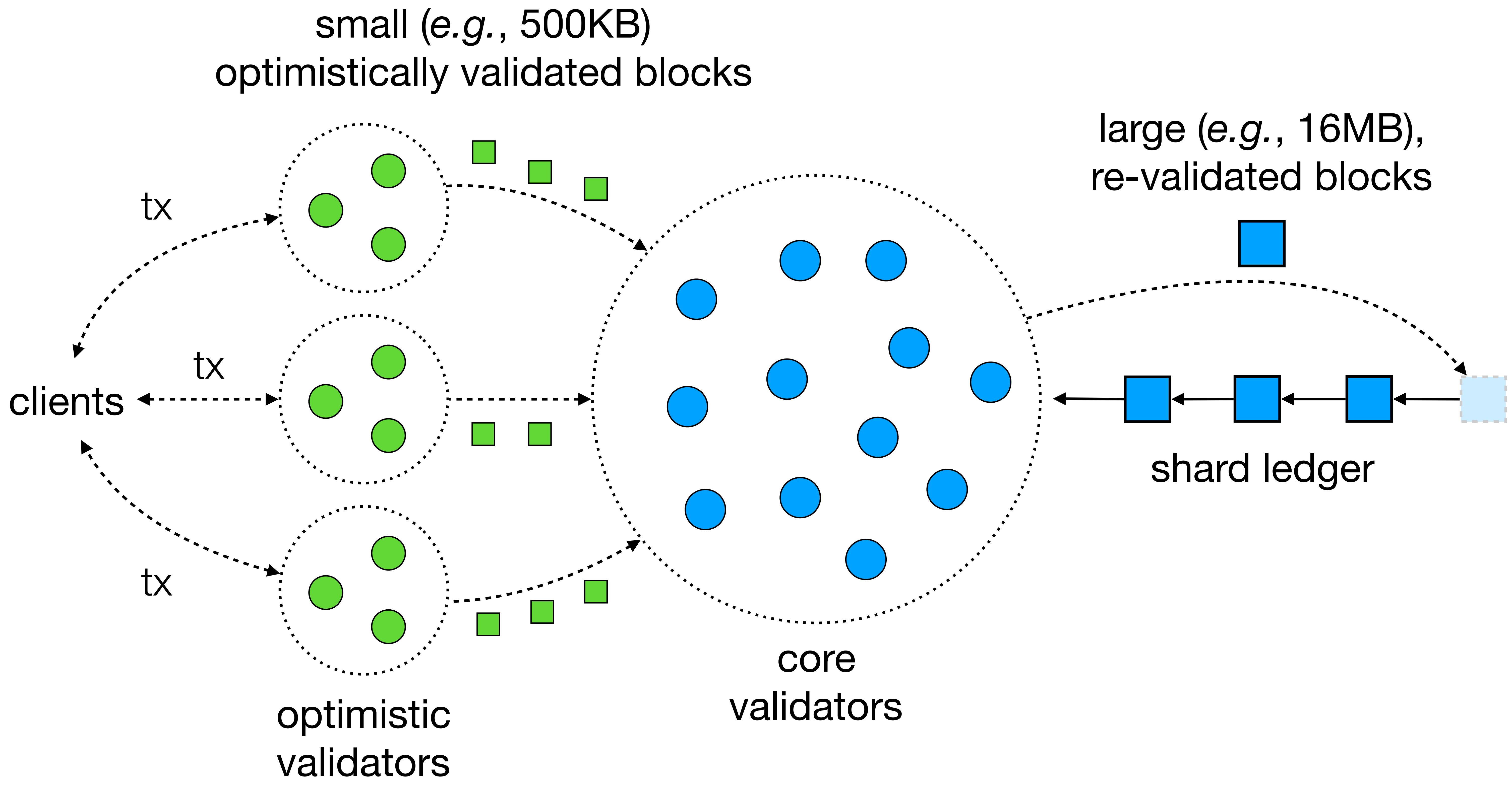


Roadmap


SimpleLedger
Randomized sharding
Secure system reconfigurations
Atomic cross-shard transactions
Security
Failure-resistant Byzantine consensus
Performance
Blockchain pruning
High-throughput low-latency transaction validation
OmniLedger

Trust-but-Verify Transaction Validation

- **Challenge:** Latency vs. throughput trade-off
- **Solution:** Two-level "trust-but-verify" validation to get low latency *and* high throughput



Talk Outline

- Motivation
- OmniLedger
- **Evaluation**
- Conclusion

Implementation & Experimental Setup

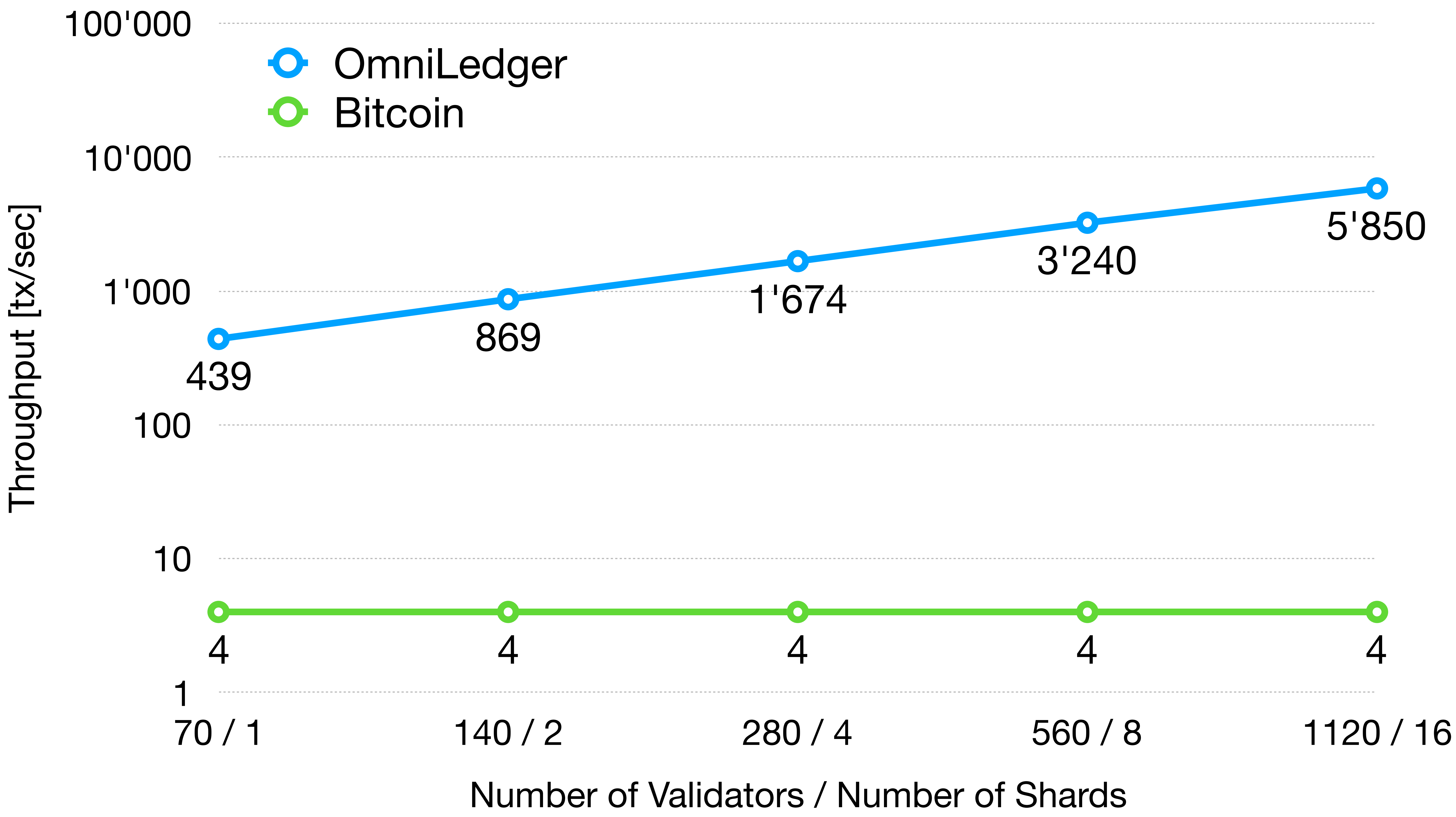
Implementation

- Go versions of OmniLedger and its subprotocols (ByzCoinX, Atomix, etc.)
- Based on DEDIS code
  - Kyber crypto library
  - Onet network library
  - Cothority framework
- https://github.com/dedis

DeterLab Setup

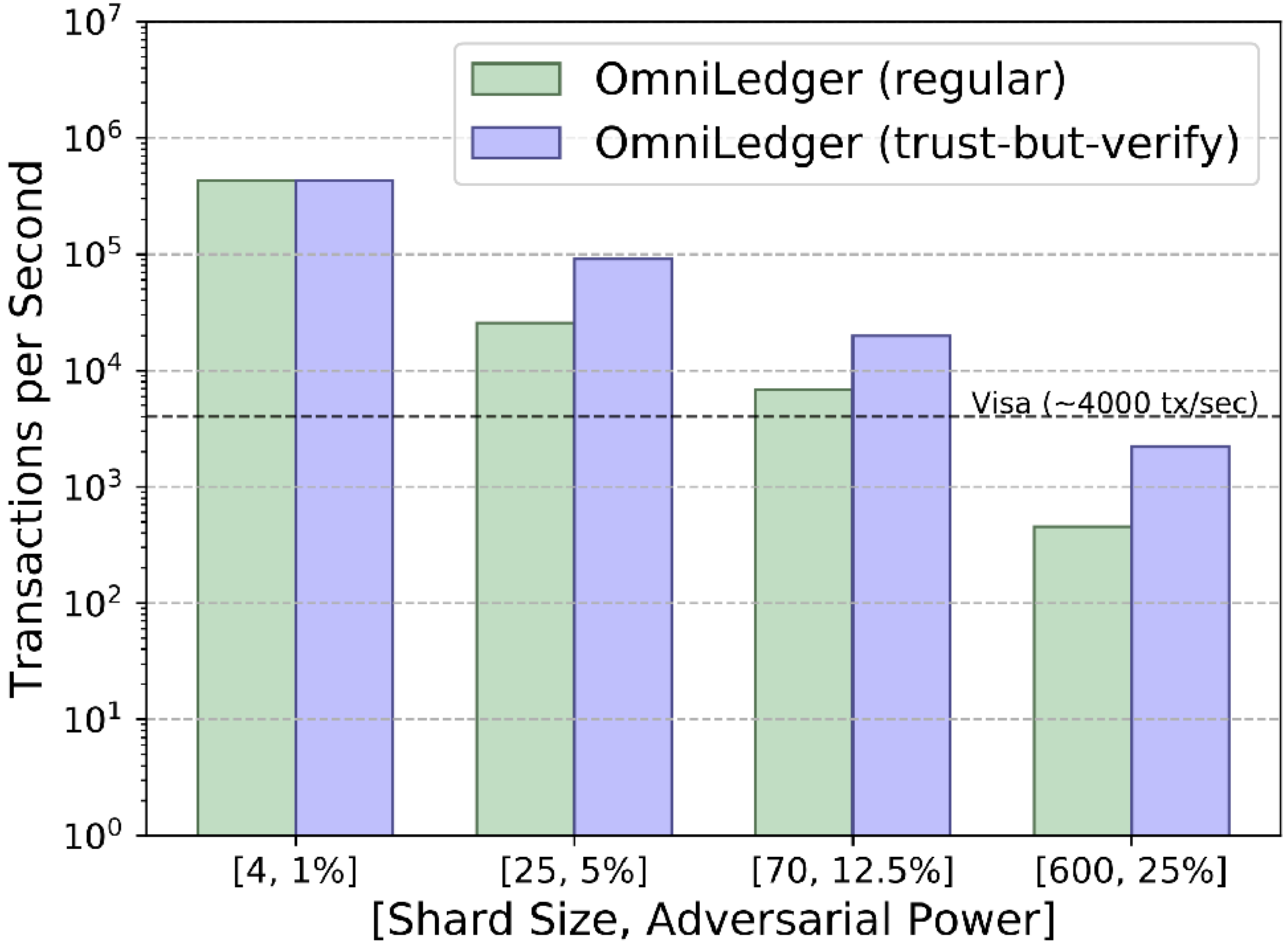
- 48 physical machines
  - Intel Xeon E5-2420 v2 (6 cores @ 2.2 GHz)
  - 24 GB RAM
  - 10 Gbps network link
- Realistic network configurations
  - 20 Mbps bandwidth
  - 200 ms round-trip latency

Evaluation: Scale-Out



For a 12.5%-adversary

Evaluation: Maximum Throughput



Results for 1800 validators

Evaluation: Latency

Transaction confirmation latency in *seconds* for regular and mutli-level validation

#shards, adversary	4, 1%	25, 5%	70, 12.5%	600, 25%	
OmniLedger regular	1.38	5.99	8.04	14.52	1 MB blocks
OmniLedger confirmation	1.38	1.38	1.38	4.48	500 KB blocks
OmniLedger consistency	1.38	55.89	41.89	62.96	16 MB blocks
Bitcoin confirmation	600	600	600	600	1 MB blocks
Bitcoin consistency	3600	3600	3600	3600	

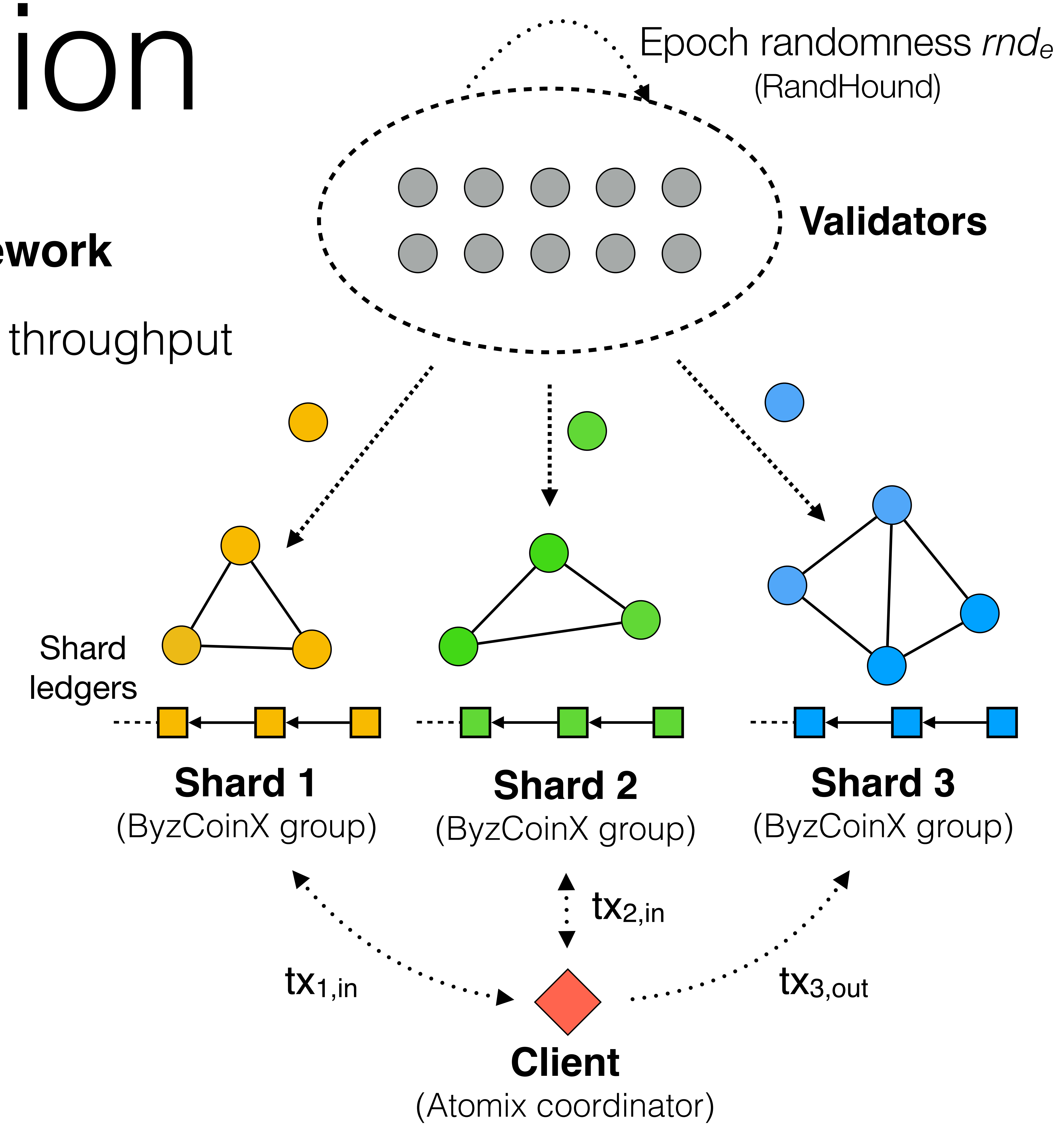
latency increase since optimistically validated blocks are batched into larger blocks for final validation to get better throughput

Talk Outline

- Motivation
- OmniLedger
- Experimental Results
- **Conclusion**

Conclusion

- **OmniLedger – Secure scale-out distributed ledger framework**
  - Sharding via unbiasable randomness for linearly-scaling throughput
  - Atomix: Client-managed cross-shard transactions
  - ByzCoinX: Robust intra-shard BFT consensus
  - Trust-but-verify validation for low latency *and* high throughput
  - For PoW, PoS, permissioned, etc.
- **Paper:** ia.cr/2017/406 (published at IEEE S&P'18)
- **Code:** https://github.com/dedis



Thanks!

philipp.jovanovic@epfl.ch – @daeinar

Network Coding for Distributed Consensus*

(Beongjun Choi, Jy-yong Sohn, Dong-Jun Han and Prof. Jaekyun Moon)

Achievements & Future Plan (Summary)

- Achievements
  - Suggested **Network-coded PBFT** algorithm [CSHM,ISIT19]
  - Obtained **Fundamental Bounds** for Network-coded PBFT algorithms
  - Constructed **Optimal Codes** using Constant-weight Codes

- Future Works
  - Generalize to Systems using Message Digests, the output of a Hash function
  - Apply the Suggested Codes in Blockchain Systems

[CSHM, ISIT'19] B. Choi, J. Sohn, D. Han and J. Moon, "Scalable Network-Coded PBFT Consensus Algorithm", accepted at *IEEE International Symposium on Information Theory* (ISIT) 2019.

Making a Consensus in Distributed Networks

- Various Applications
  - Database, File Systems
  - Tamper-resistant Distributed Ledger: Blockchain
  - Making a Decision without a Central Authority

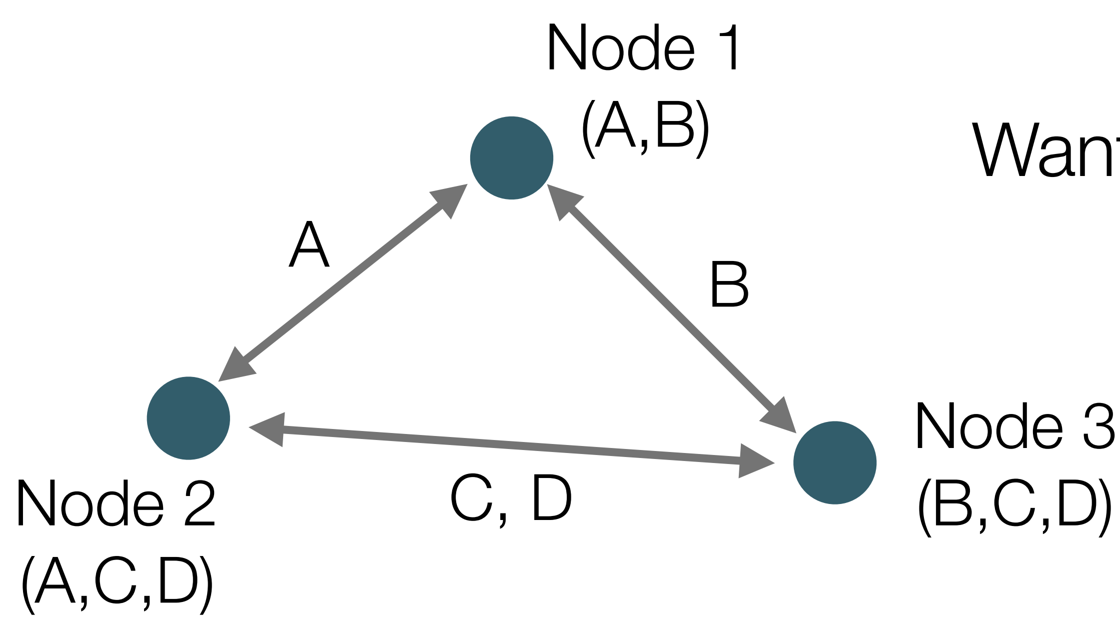
- **Byzantine** nodes
  - Transmit false data to other nodes
  - Sufficiently large number of Byzantine nodes **mislead the consensus**

Practical Byzantine Fault Tolerance (PBFT) **[OSDI'99]**

- Consider f nodes are Byzantine out of m nodes. If $m \geq 3f + 1$, then the PBFT algorithm ensures the agreement of data in finite steps

- Various Blockchain Systems use PBFT-based consensus protocols
  - Ripple, Tendermint, Zilliqa

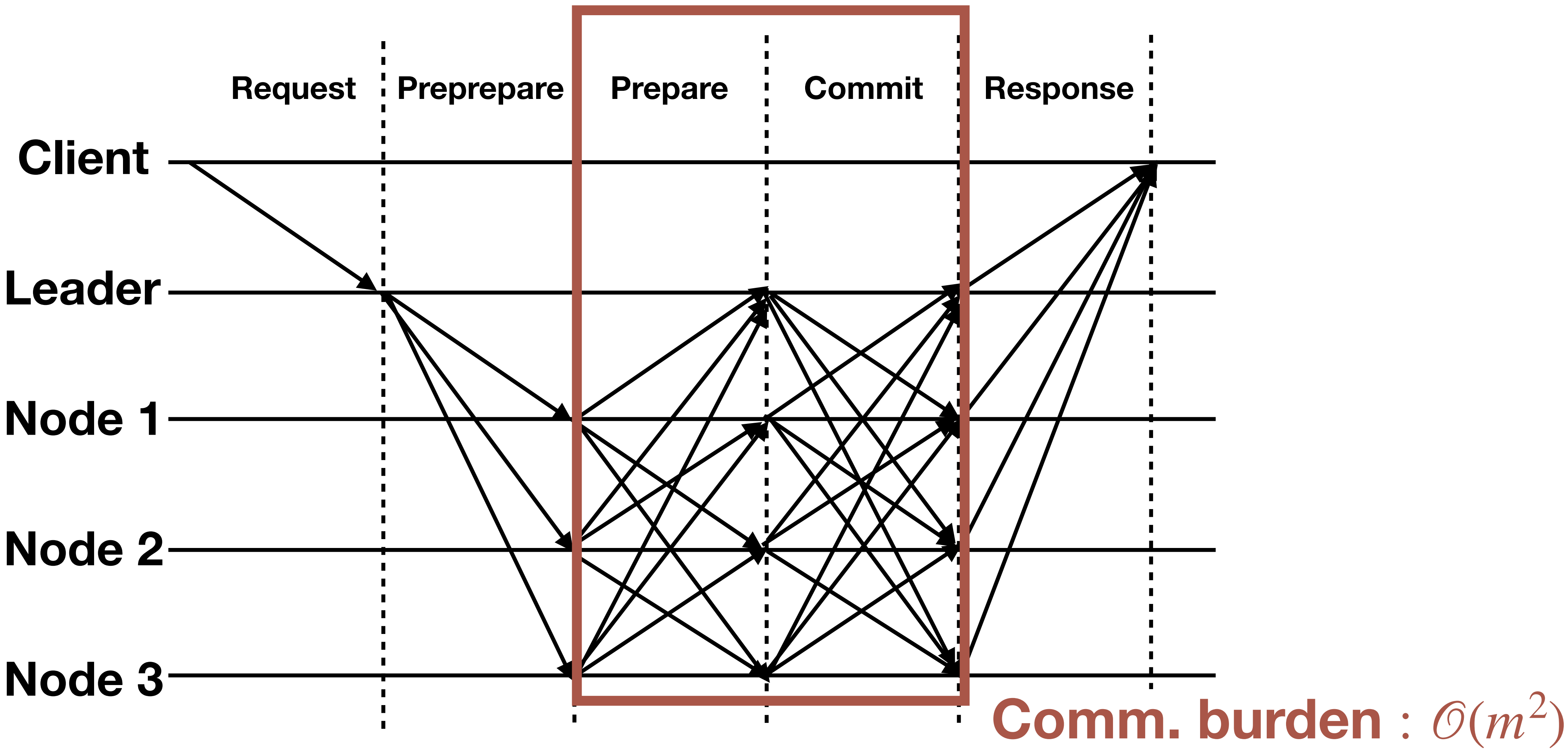
- **Core**: nodes sharing a common data transmit what they store



Want to make a consensus on data blocks A,B,C,D

[OSDI'99] M. Castro, B. Liskov et al., "Practical byzantine fault tolerance," in OSDI, vol. 99, 1999, pp. 173–186.

Issue: Communication burden of PBFT Algorithm

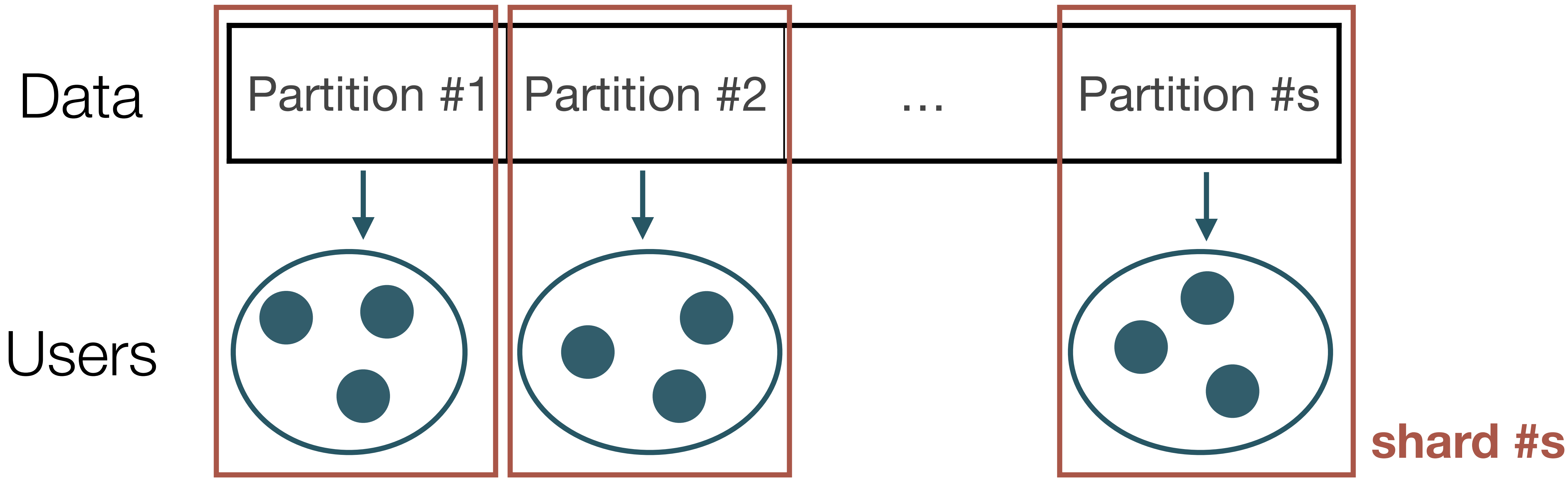


Comm. burden is one of the major drawbacks which limits the **scalability** of PBFT algortihm

(i.e., cannot increase m)

Alternative: PBFT + Sharding

- Network (Data & Users) is divided into s disjoint shards
- Users in each shard is responsible for each data partition



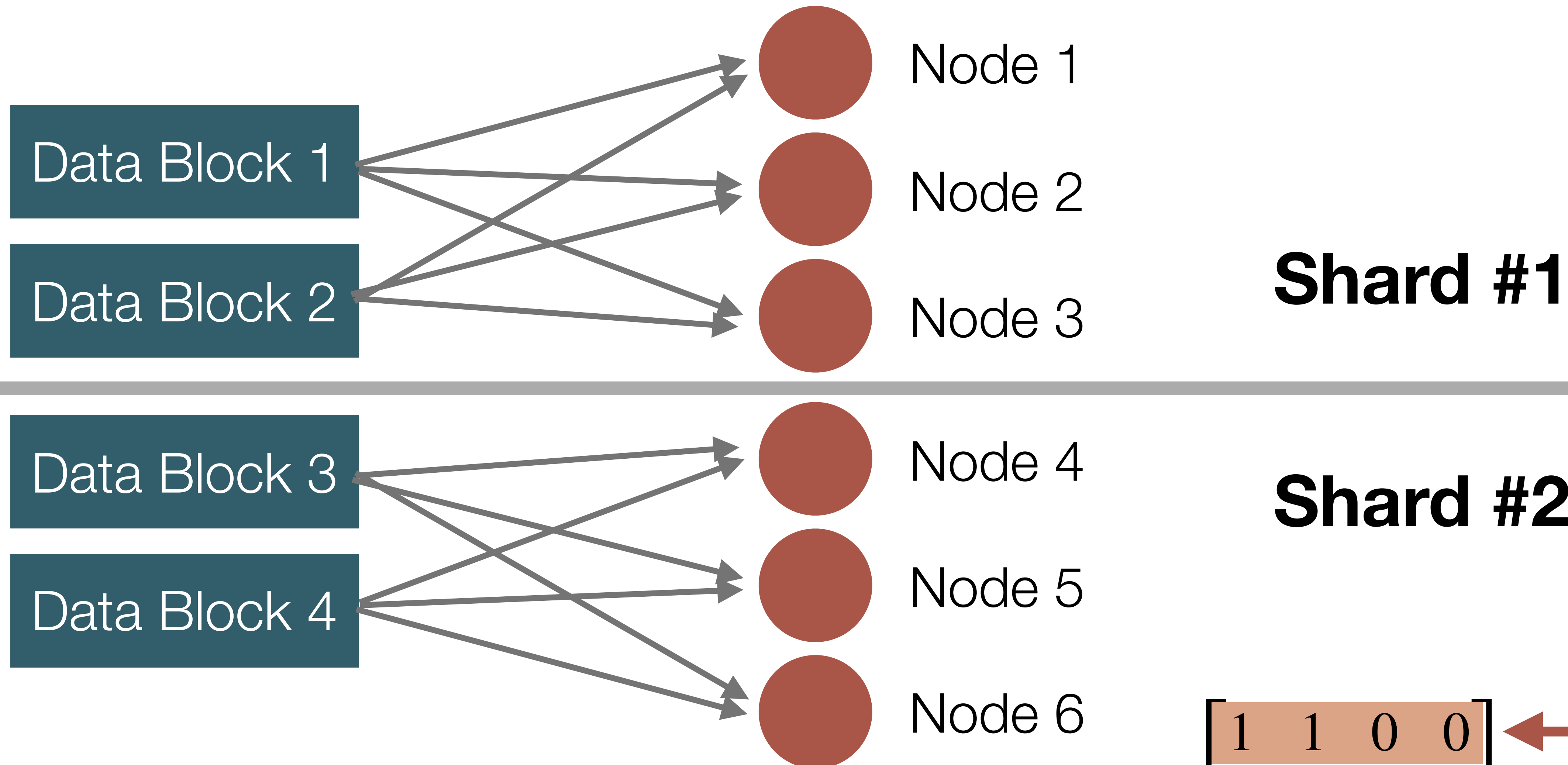
- Various blockchain systems [SIGSAG'18] use this scheme

PBFT → PBFT+Sharding

Comm. Burden : $\mathcal{O}(m^2)$ → **Comm. Burden** : $\mathcal{O}(m^2/s^2)$

[SIGSAC'18] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: scaling blockchain via full sharding," in Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 931–948.

Sharding: a Special Data Allocation Rule

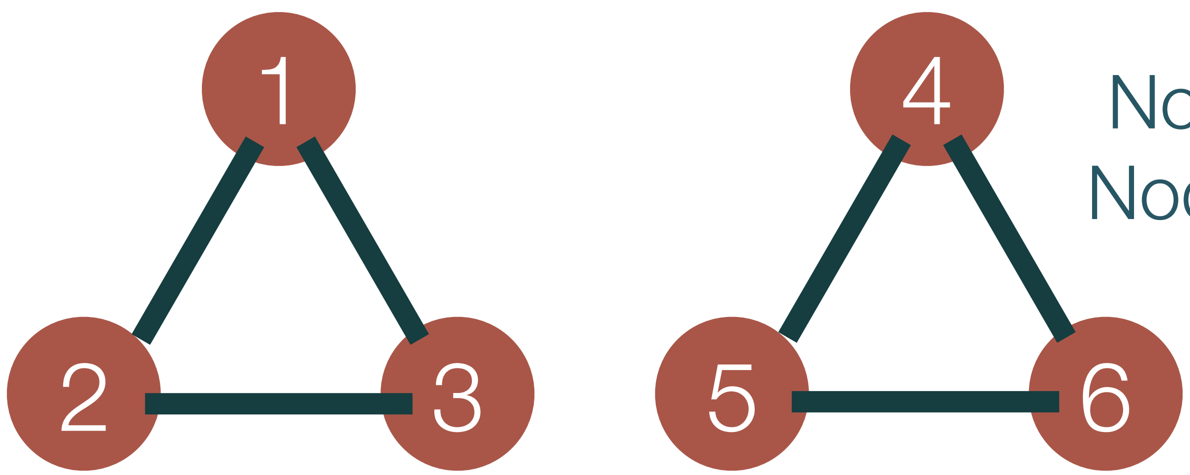


This can be expressed as a matrix $G = \begin{array}{c} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \end{array} \begin{array}{l} \leftarrow \textbf{Node 1} \\ \\ \\ \\ \\ \\ \end{array}$

ex. **Node 1** is responsible for **Data Blocks #1,#2**

Issue: Maximum Link Bandwidth

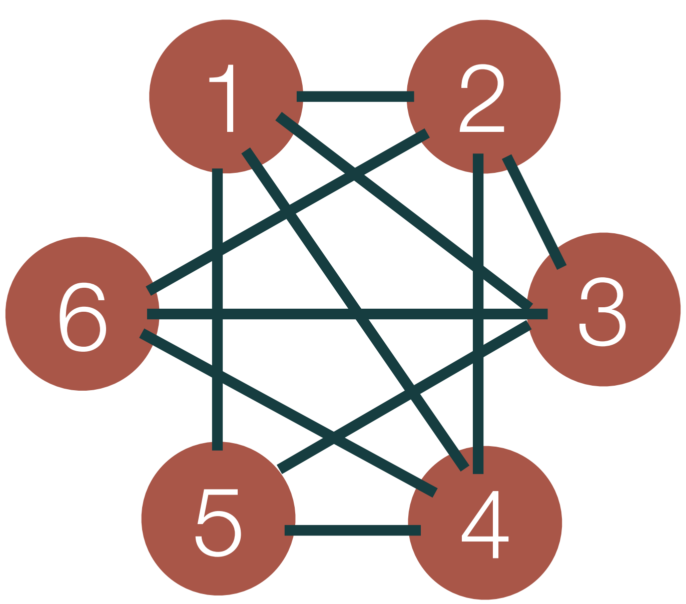
- Sharding: some limited links cover all the communication burden



Nodes 1,2,3 transmit Data blocks #1,#2
Nodes 4,5,6 transmit Data blocks #3, #4

Total Bandwidth = 12
Maximum Link Bandwidth = 2

- Can we spread out this communication, and reduce the **maximum link bandwidth,** while maintaining the same total bandwidth?



Total Bandwidth = 12
Maximum Link Bandwidth = 1

Suggested: PBFT + Network Coding (General Data Allocation Framework)

- Parameters

m: # of nodes
n: # of data blocks
f: # of Byzantine nodes

$$G = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Node 1 is responsible for **Data Blocks #1,#2**

$G \in \{0,1\}^{m \times n}$: Data Allocation Matrix

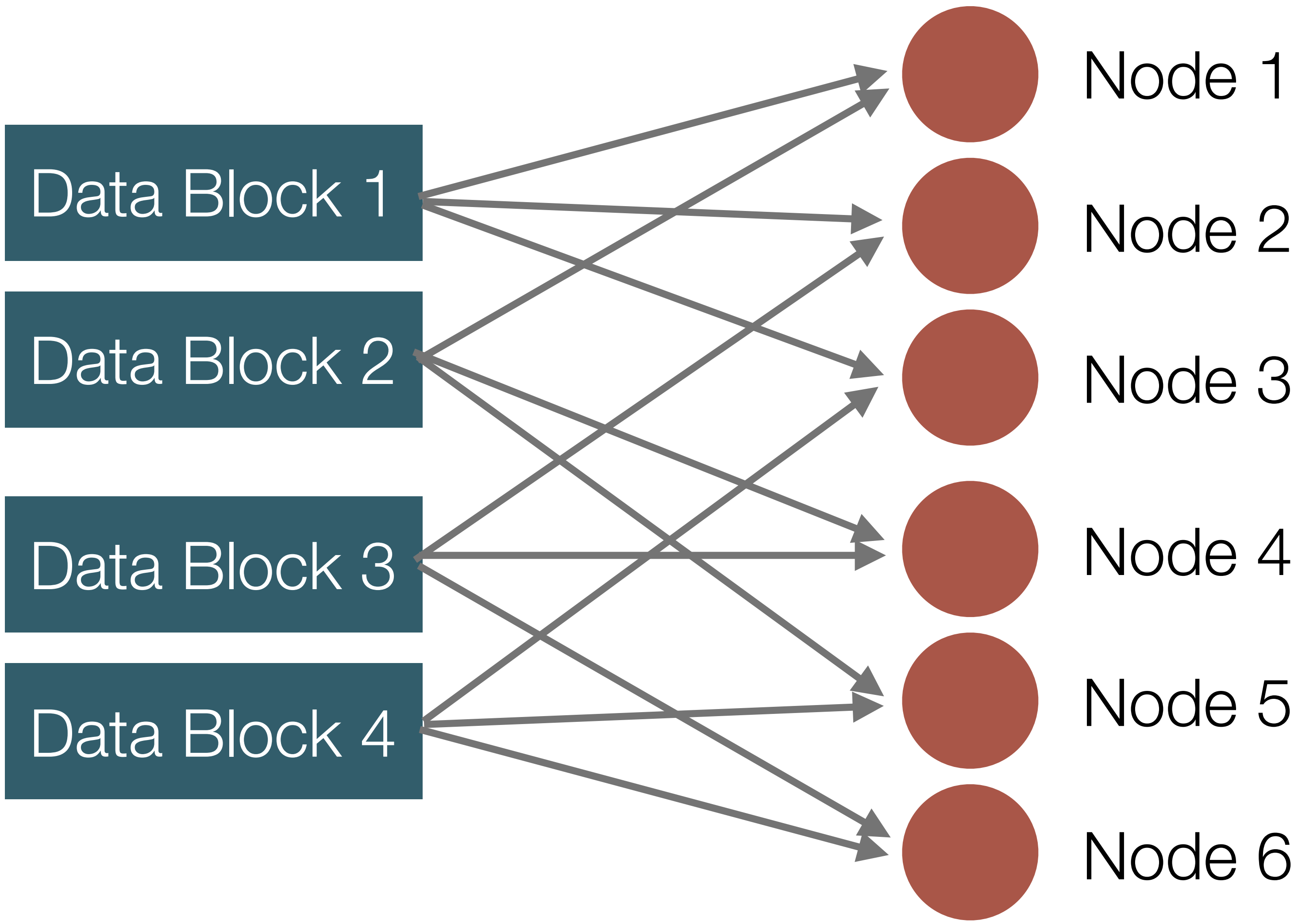
$$G_{ij} = \begin{cases} 1, & \text{if node } i \text{ is responsible for data block } j \\ 0, & \text{otherwise} \end{cases}$$

$$\rho_i = \frac{\sum_{j=1}^{n} G_{ij}}{n}$$: Storage overhead of node i

$$\gamma_{a,b} = \frac{\sum_{k=1}^{n} G_{a,k} G_{b,k}}{n}$$: Bandwidth between nodes a and b

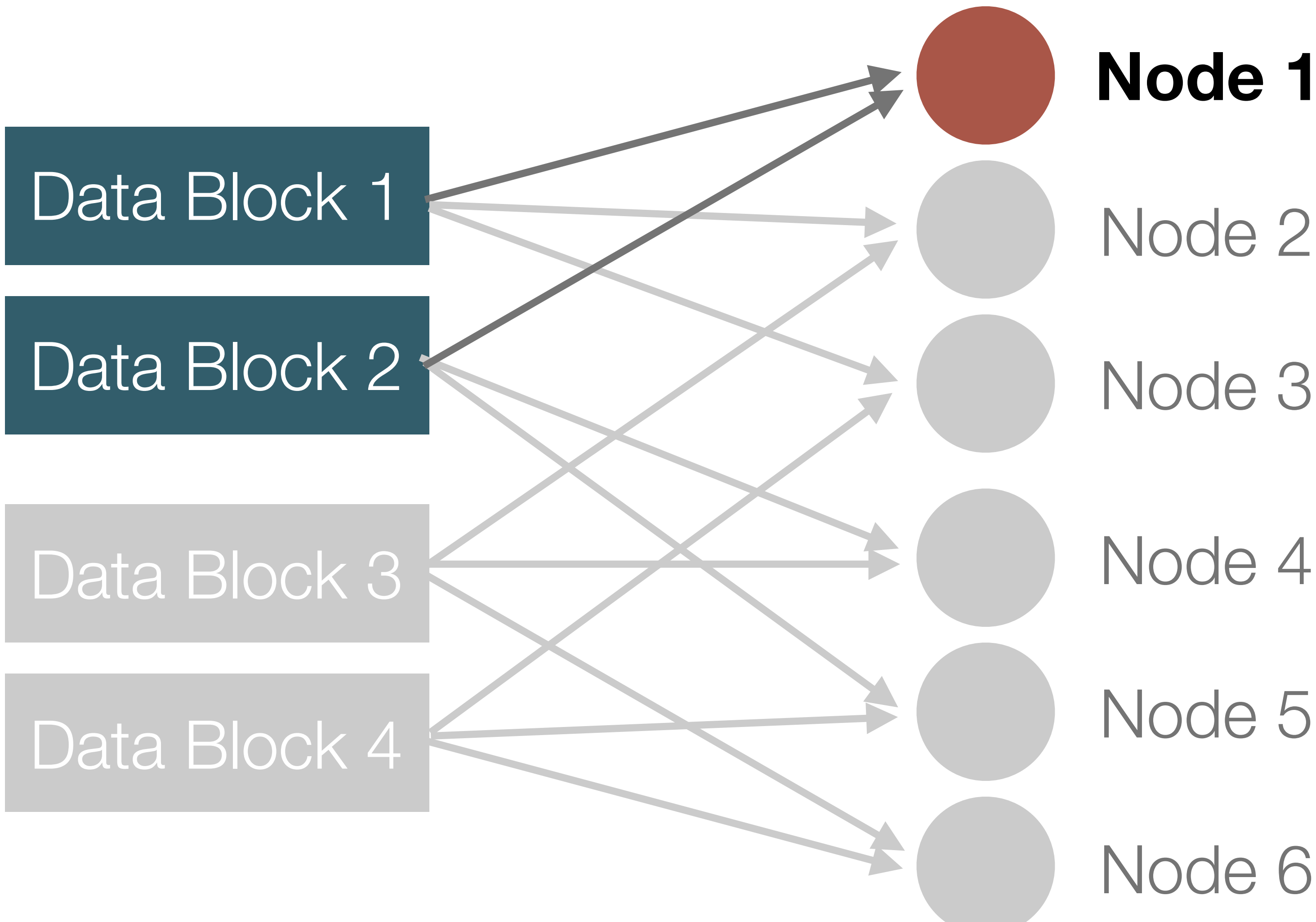
$$\gamma_{max} = \max_{a,b \in [m], a \neq b} \gamma_{a,b}$$: **Maximum Link Bandwidth**

Example: m=6 nodes share n=4 data blocks



$$G = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

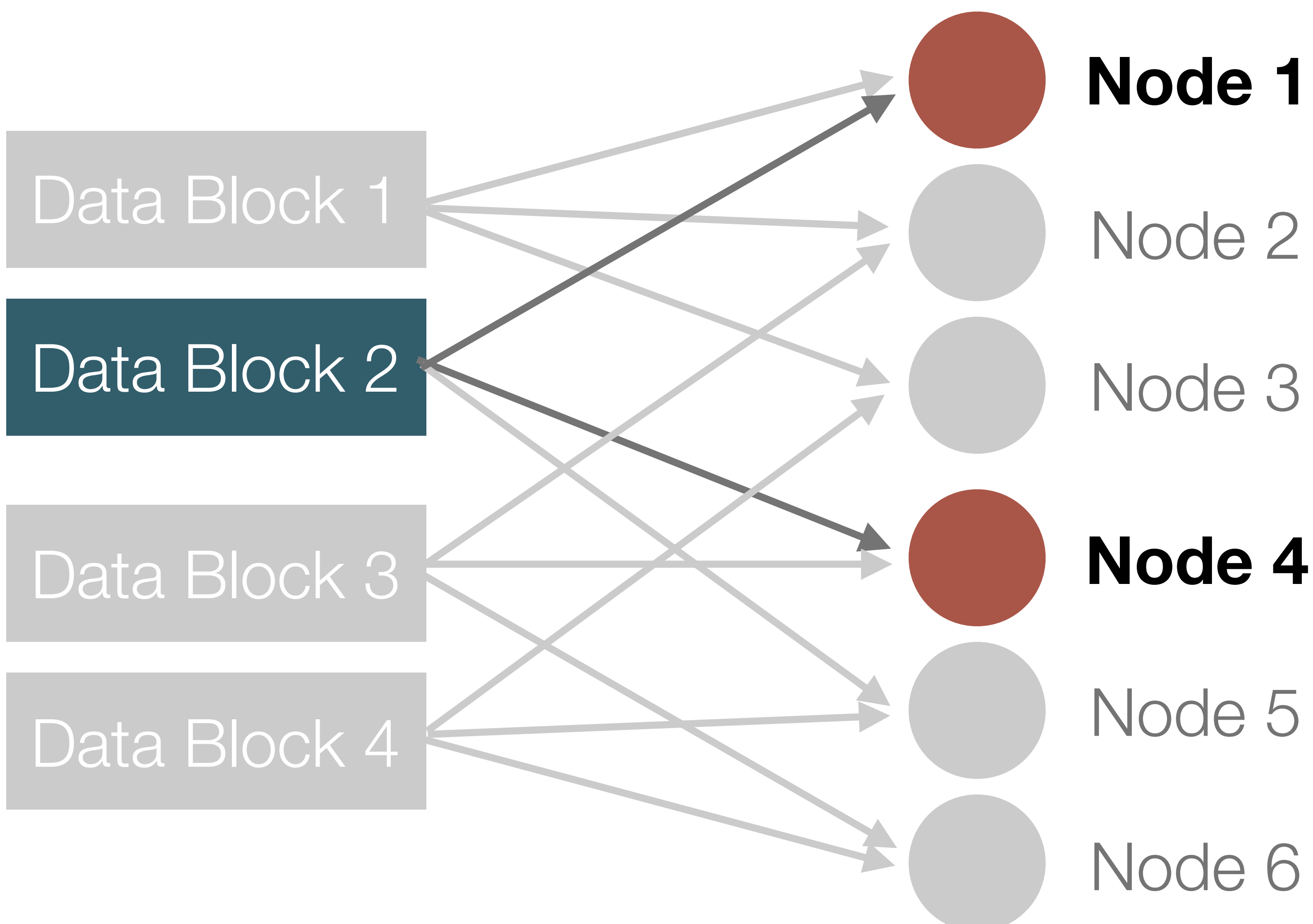
Example: m=6 nodes share n=4 data blocks



$$G = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Storage overhead of node i : $\rho_i = \rho = 0.5$ for $i = 1, \cdots, 6.$

Example: m=6 nodes share n=4 data blocks

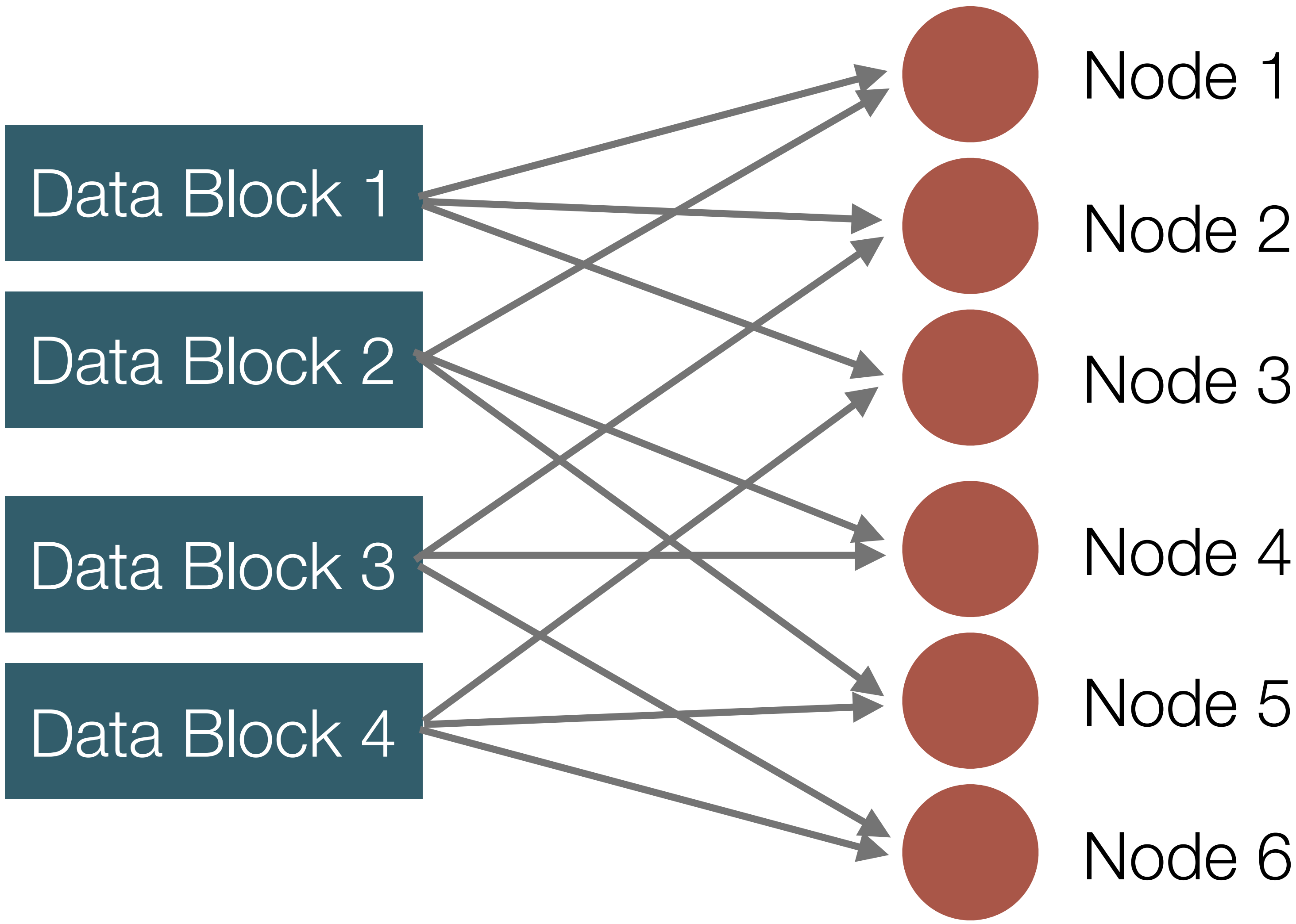


$$G = \begin{array}{cl} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} & \begin{array}{l} \textbf{Node 1} \\ \\ \\ \textbf{Node 4} \\ \\ \\ \end{array} \end{array}$$

Storage overhead of node i : $\rho_i = \rho = 0.5$ for $i = 1, \cdots, 6.$

Bandwidth between nodes 1 and 4: $\gamma_{1,4} = 0.25$

Example: n=4 data blocks allocated to m=6 nodes



$$G = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Storage overhead of node i : $\rho_i = \rho = 0.5$ for $i = 1, \cdots, 6.$

Bandwidth between nodes 1 and 4: $\gamma_{1,4} = 0.25$

Maximum Link Bandwidth: $\gamma_{\text{max}} = 0.25$ (since $\gamma_{a,b} \leq 0.25 \ \forall a, b$)

Suggested: Reducing the maximum link bandwidth

- Key Questions [CSHM, ISIT'19]
  - How can we design G to reduce the **maximum link bandwidth** γ_{max} compared to the sharding protocol?
  - Q1: Is there any lower bound on γ_{max}?
  - Q2: How to design the optimal G which achieves the lower bound?

[CSHM, ISIT'19] B. Choi, J. Sohn, D. Han and J. Moon, "Scalable Network-Coded PBFT Consensus Algorithm", accepted at *IEEE International Symposium on Information Theory* (ISIT) 2019.

Suggested: Reducing the maximum link bandwidth

- Key Questions [CSHM, ISIT'19]
  - How can we design G to reduce the **maximum link bandwidth** $\gamma_{\max}$ compared to the sharding protocol?
  - Q1: Is there any lower bound on $\gamma_{\max}$?

    >> Result #1. Maximum link bandwidth satisfies $\gamma_{\max} \geq \gamma^{\star}(\rho)$.

  - Q2: How to design the optimal G which achieves the lower bound?

    >> Result #2. Provided optimal G which satisfies $\gamma_{\max} = \gamma^{\star}(\rho)$ using Constant Weight Codes [TIT'90]

[CSHM, ISIT'19] B. Choi, J. Sohn, D. Han and J. Moon, "Scalable Network-Coded PBFT Consensus Algorithm", accepted at *IEEE International Symposium on Information Theory* (ISIT) 2019.
[TIT'90] A. E. Brouwer, L. B. Shearer, N. Sloane et al., "A new table of constant weight codes," in IEEE Transactions on Information Theory, 1990.

Necessary Condition for Tolerating f Byzantines

- Consider *n* data blocks are allocated to *m* nodes, where each node contains $n\rho$ data blocks. Then,

A consensus algorithm can tolerate f Byzantines

$\Downarrow$

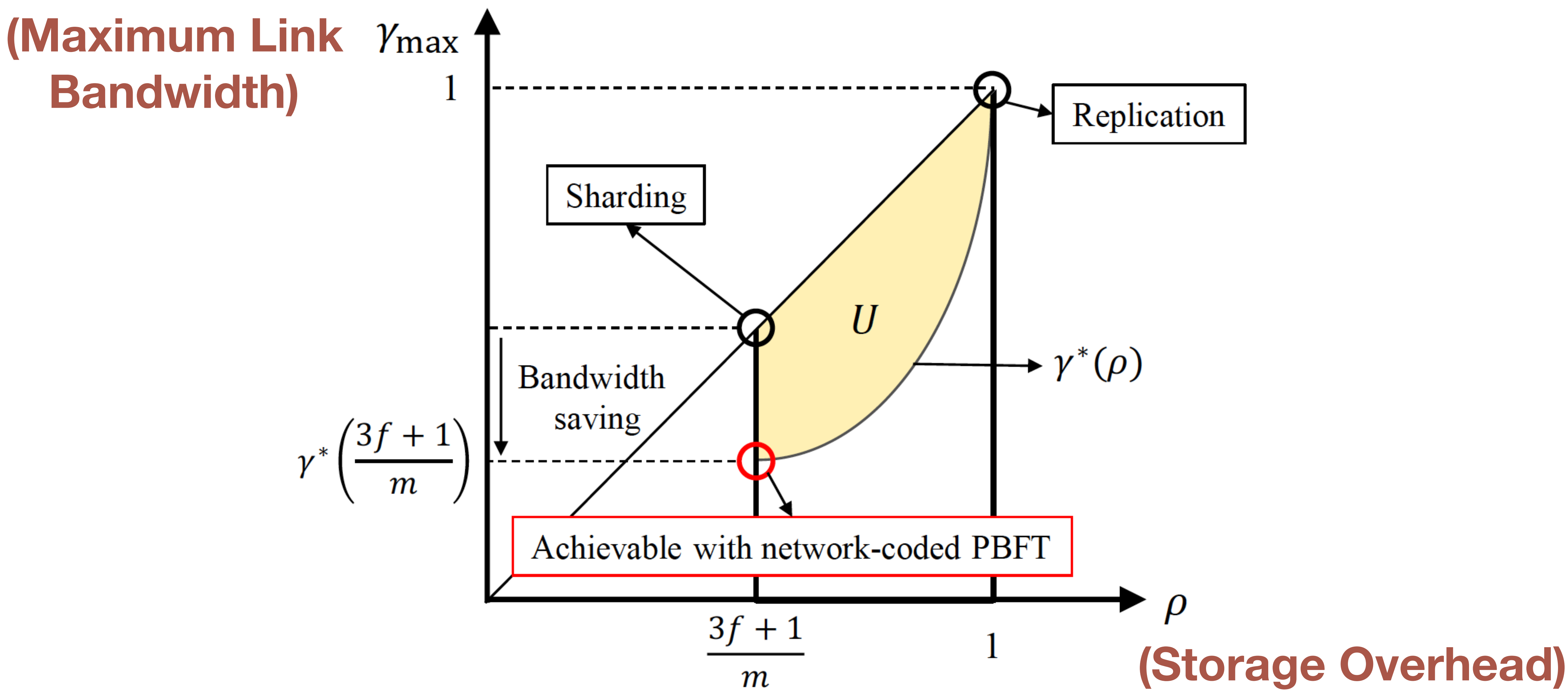
$$(\rho, \gamma_{\text{max}}) \in U$$

$$U = \left\{ (\rho, \gamma_{\text{max}}) : \frac{3f+1}{m} \le \rho \le 1, \right.$$
$$\left. \gamma^{\star}(\rho) \le \gamma_{\text{max}} \le \rho \right\}$$

ρ **: Storage Overhead**

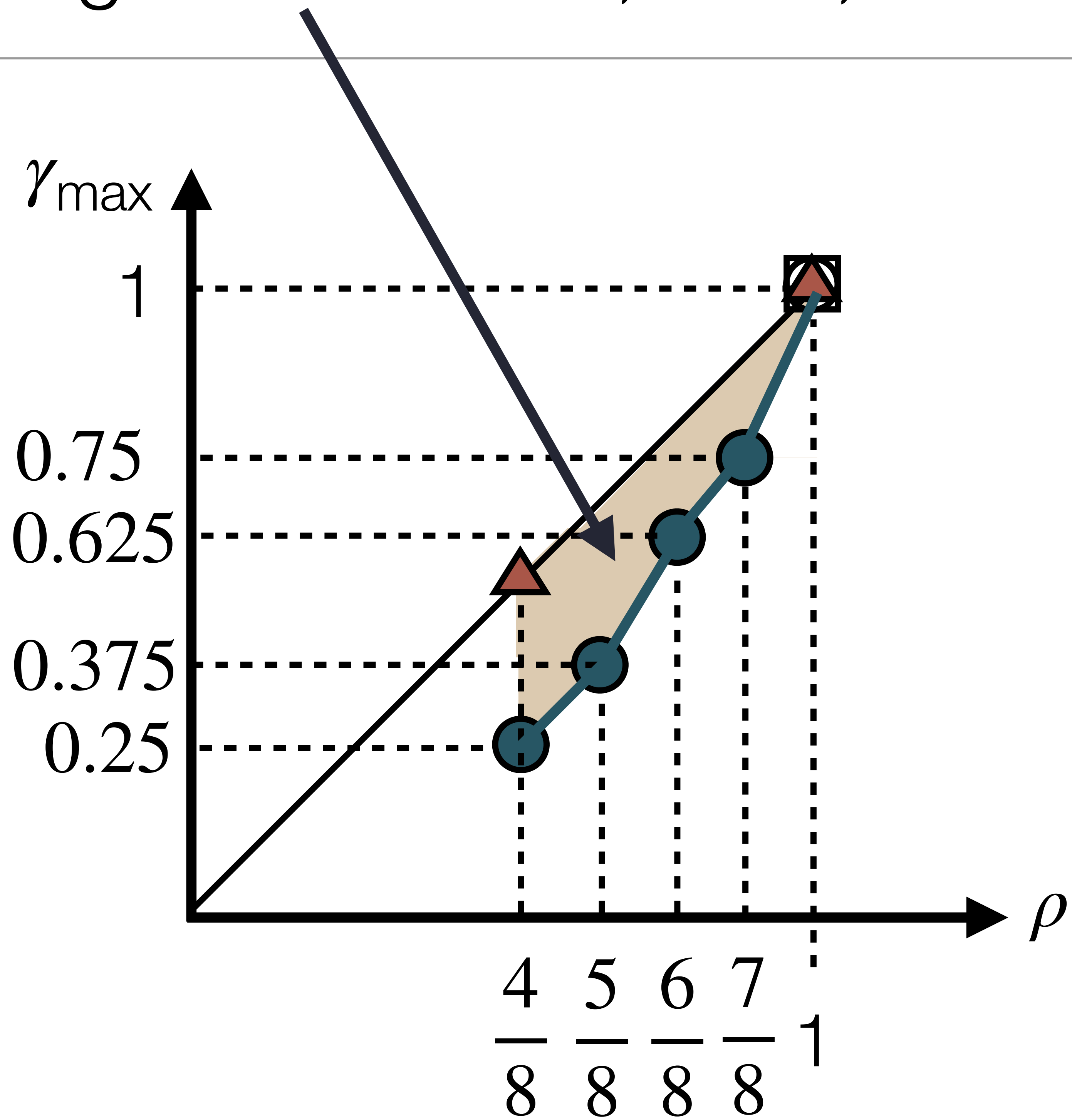
γ_{max} **: Max. Link Bandwidth**

Compare with Sharding: Reduced $\gamma_{\max}$



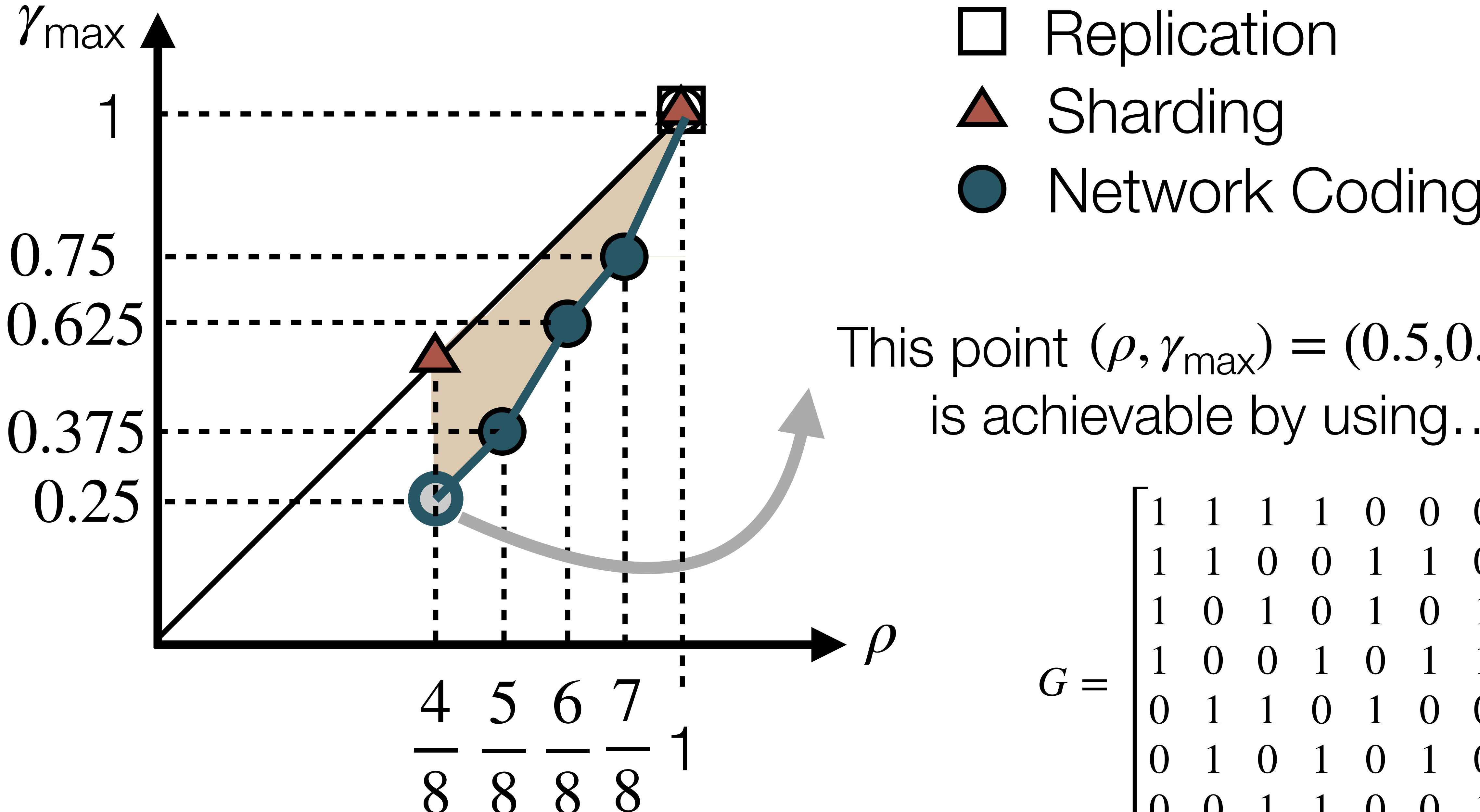
Suggested scheme (network-coded PBFT) can reduce $\gamma_{\max}$ compared to the conventional sharding protocol

Region **U** for n=8, m=8, f=1



ρ	$\gamma^{\star}(\rho)$
0.5	0.25
0.625	0.375
0.75	0.625
0.875	0.75
1	1

Feasible Region **U** for n=8, m=8, f=1



This point $(\rho, \gamma_{max}) = (0.5, 0.25)$ is achievable by using…

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Code for n=8, m=8, f=1, $\rho = 0.5$, $\gamma_{\max} = 0.25$

n=(# of data blocks)

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

m=(# of nodes)

- Each row has weight $n\rho = 4$.
- Any two rows share at most $n\gamma_{\max} = 2$ columns where both have the element 1.

Code for n=8, m=8, f=1, $\rho = 0.5$, $\gamma_{\text{max}} = 0.25$

n=(# of data blocks)

$$\begin{array}{r} \text{Node 1} \rightarrow \\ \\ \\ G = \\ \\ \\ \text{Node 7} \rightarrow \\ \\ \end{array} \begin{bmatrix} 1 & 1 & \boxed{1} & \boxed{1} & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & \boxed{1} & \boxed{1} & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

m=(# of nodes)

- Each row has weight $n\rho = 4$.
- Any two rows share at most $n\gamma_{\text{max}} = 2$ columns where both have the element 1.

Code for n=8, m=8, f=1, $\rho = 0.5$, $\gamma_{\max} = 0.25$

n=(# of data blocks)

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

m=(# of nodes)

- **Note**: Every row is a codeword of **constant weight codes** **[TIT'90]** with parameters $(n, d, w) = (n, 2\lceil n(\rho - \gamma_{\max})\rceil, n\rho) = (8,4,4)$

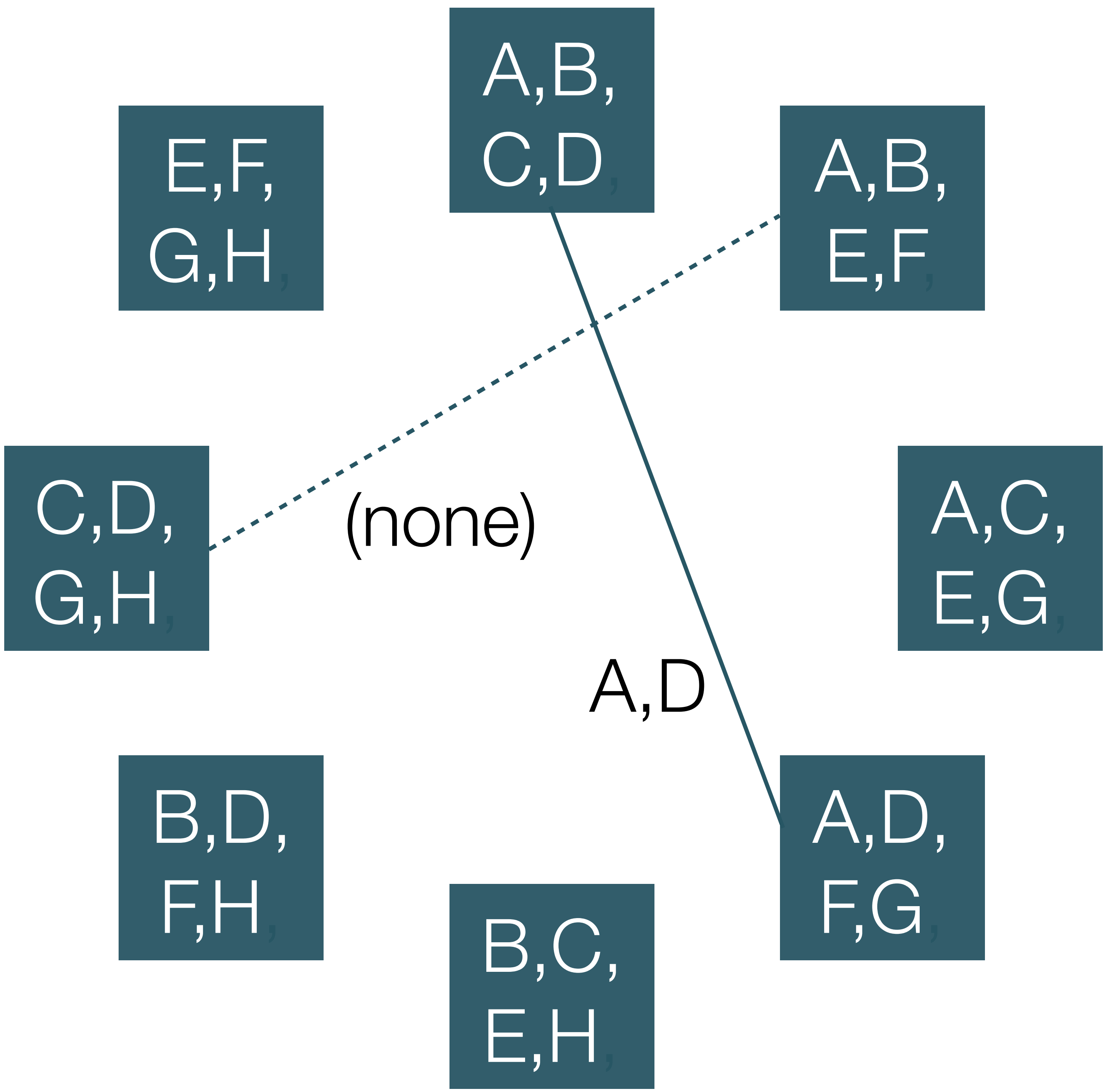
Code Length

Minimum Distance

Weight of each codeword

[TIT'90] A. E. Brouwer, L. B. Shearer, N. Sloane et al., "A new table of constant weight codes," in IEEE Transactions on Information Theory, 1990.

Code for n=8, m=8, f=1, $\rho = 0.5$, $\gamma_{\max} = 0.25$



Data Blocks
A, B, …, H

Each node contains
$n\rho = 4$ blocks

Any two nodes share
at most $n\gamma_{\max} = 2$ blocks

Future Plan

- Generalize to systems using message digests
  - The suggested scheme assumes that **plaintext** is transmitted across different nodes.
  - In practical blockchain systems (e.g. Bitcoin, Ethereum, Zilliqa), data blocks are **compressed by a hash function** before transmission, due to the large size of data blocks.

  >> The advantage of the suggested scheme dwarfs when message digests are used. **Appropriate alternatives are required** for such systems.