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

Philipp Jovanovic (@daeinar)

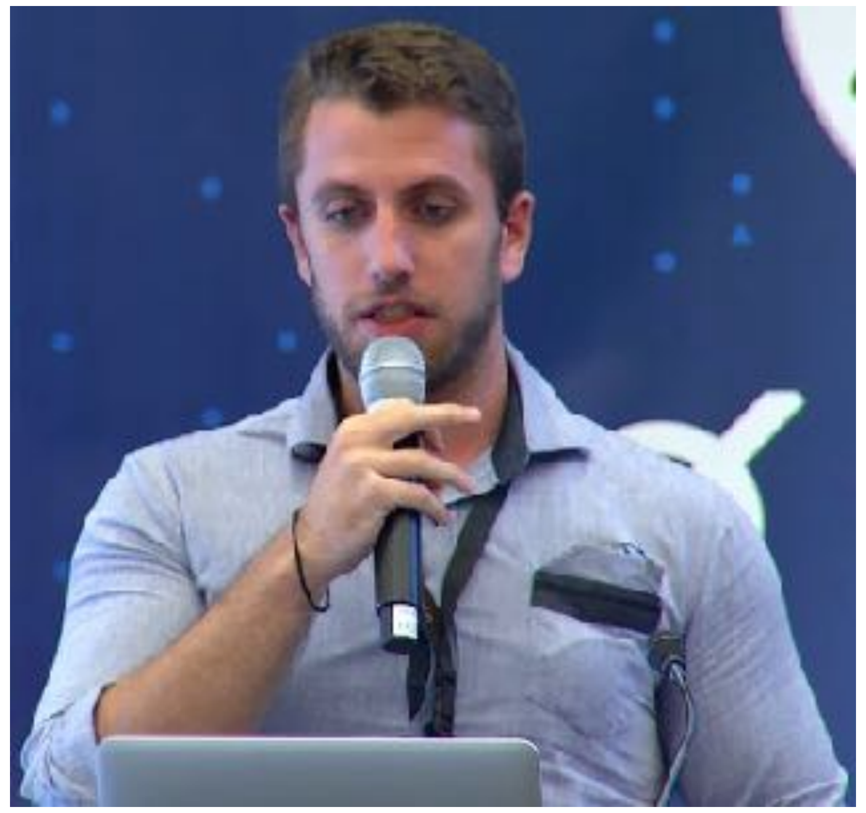
Distributed and Decentralized Systems Lab (DEDIS)

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

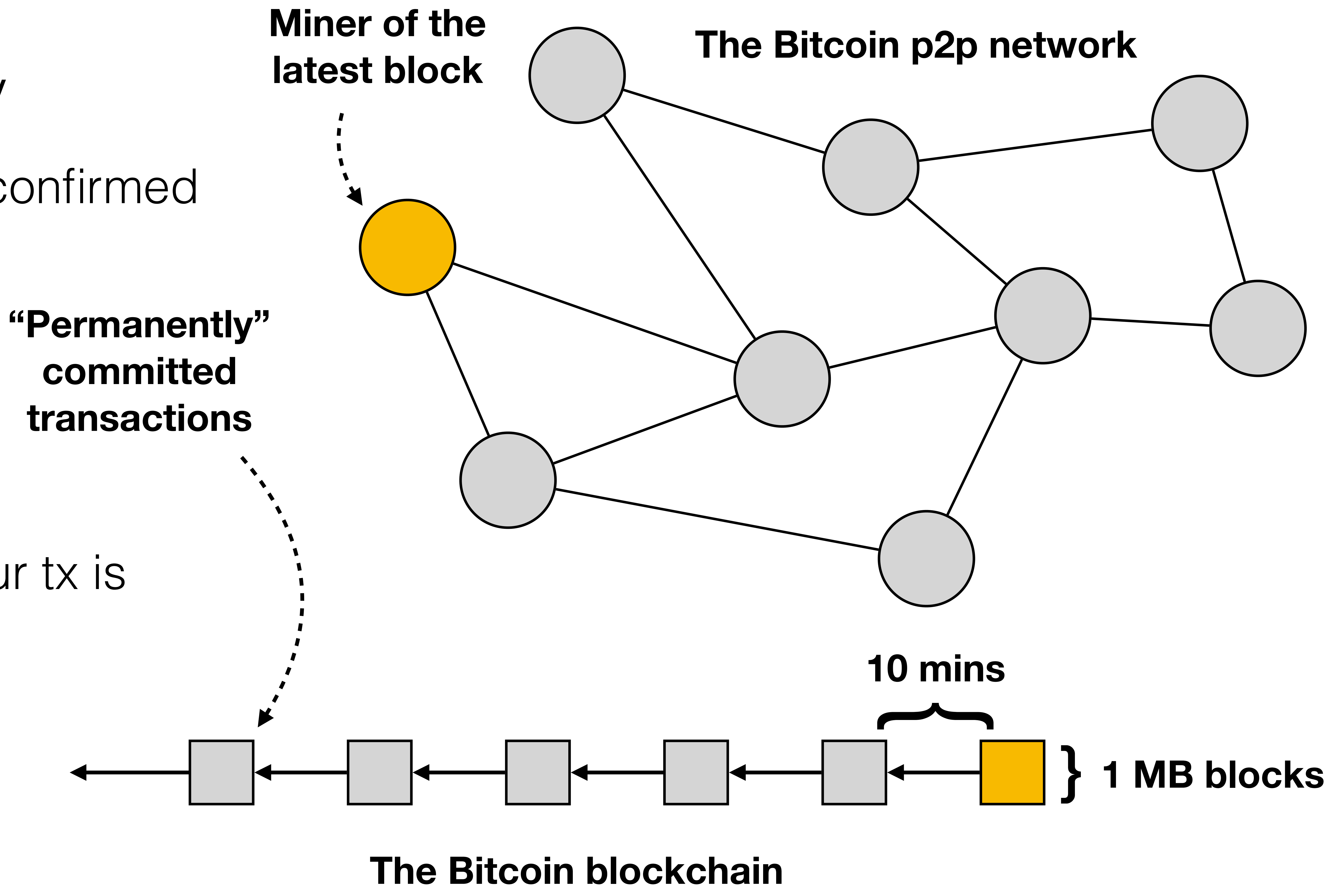
Talk Outline

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

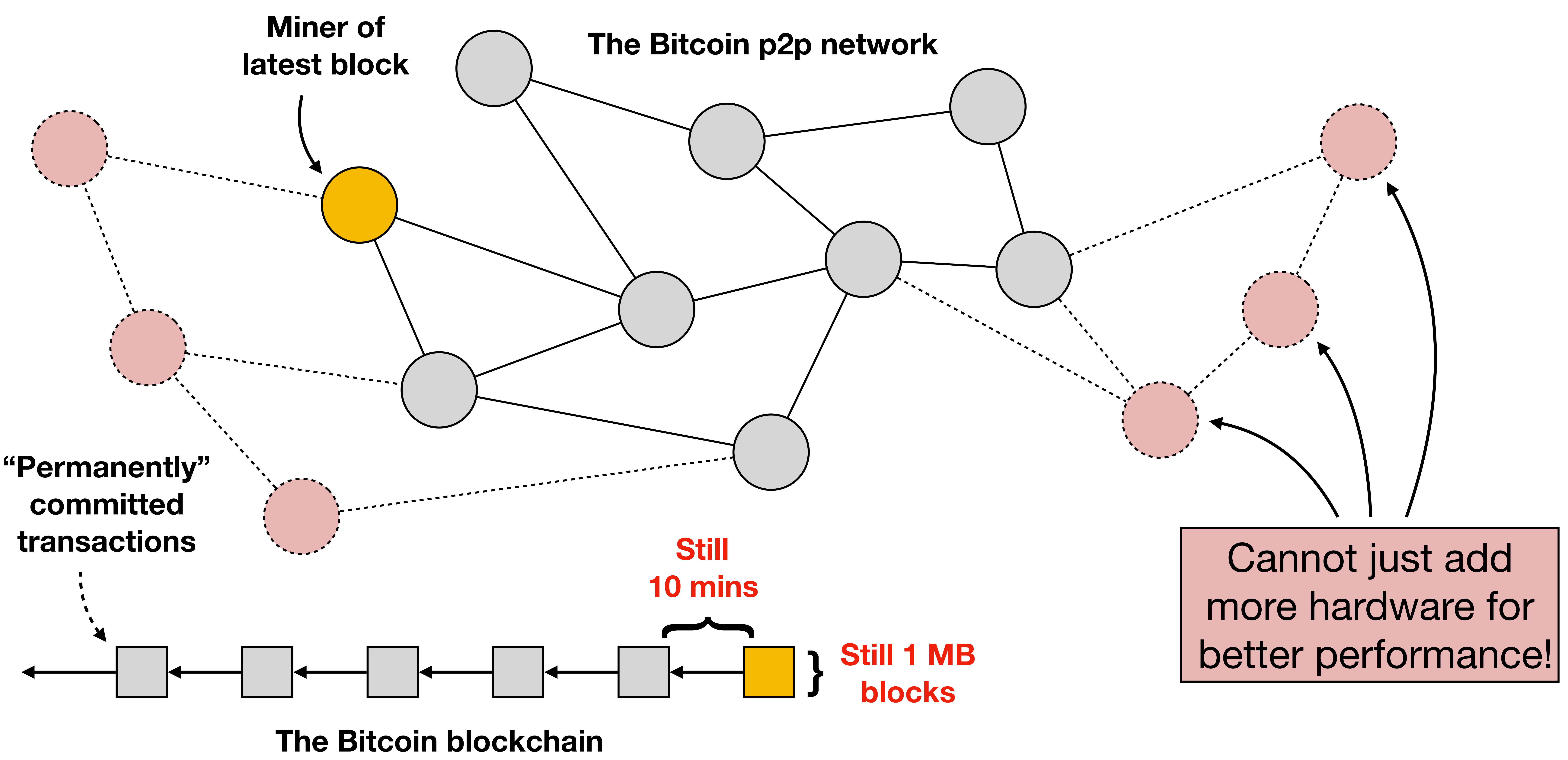
The Core of Bitcoin: Nakamoto Consensus

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

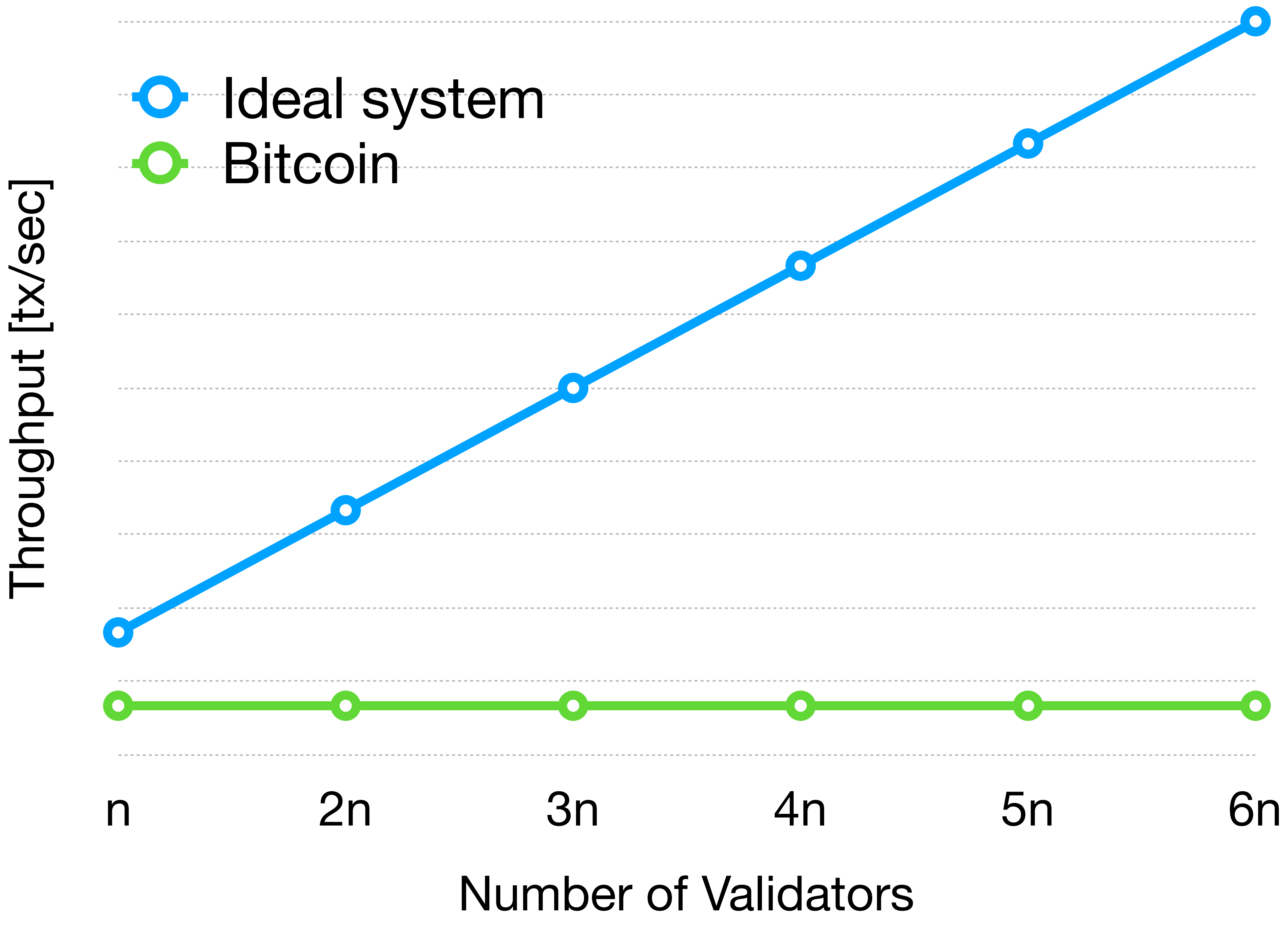


… But Scaling Blockchains is Not Easy



Cannot just add more hardware for better performance!

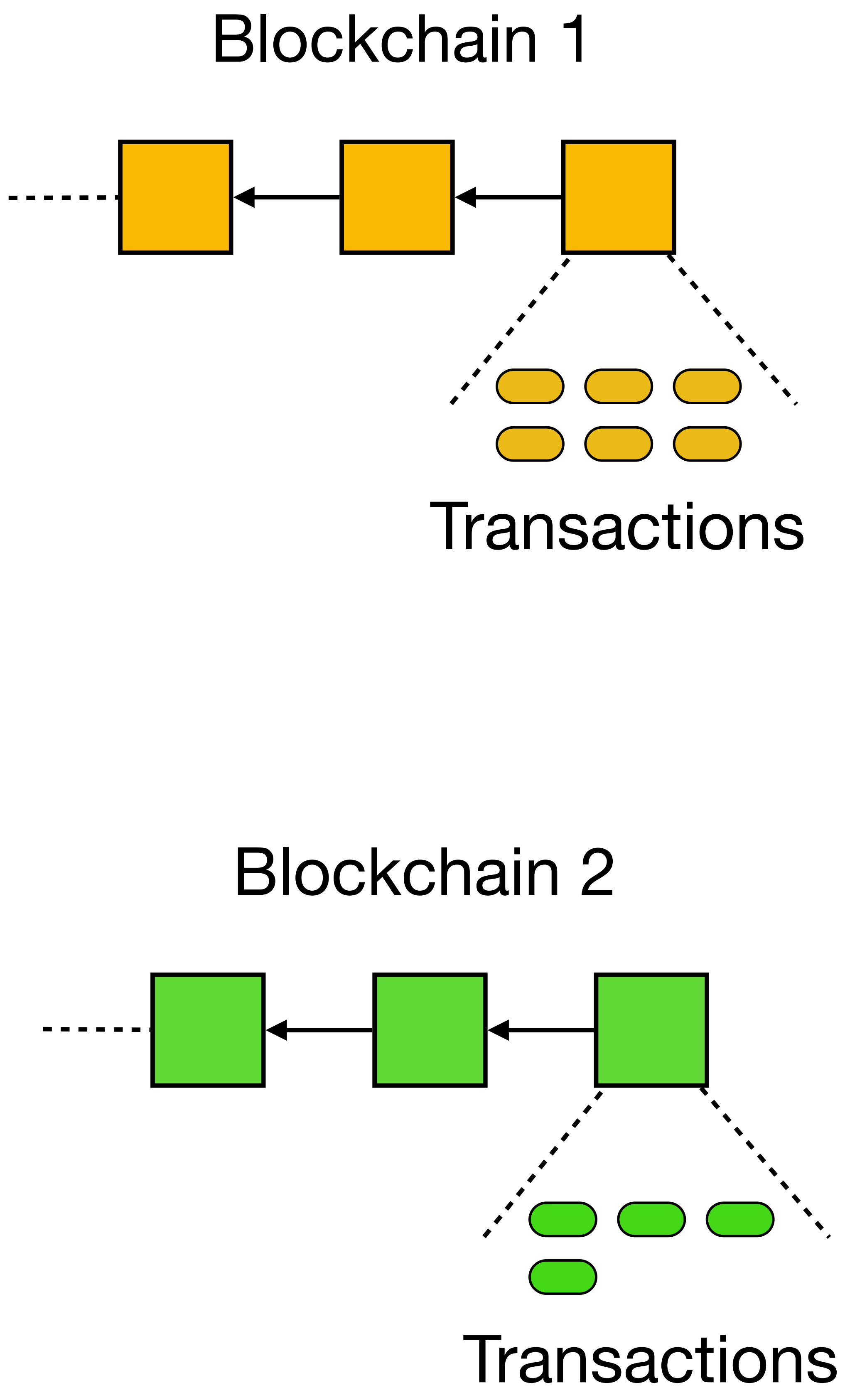
What we Want: Scale-Out Performance



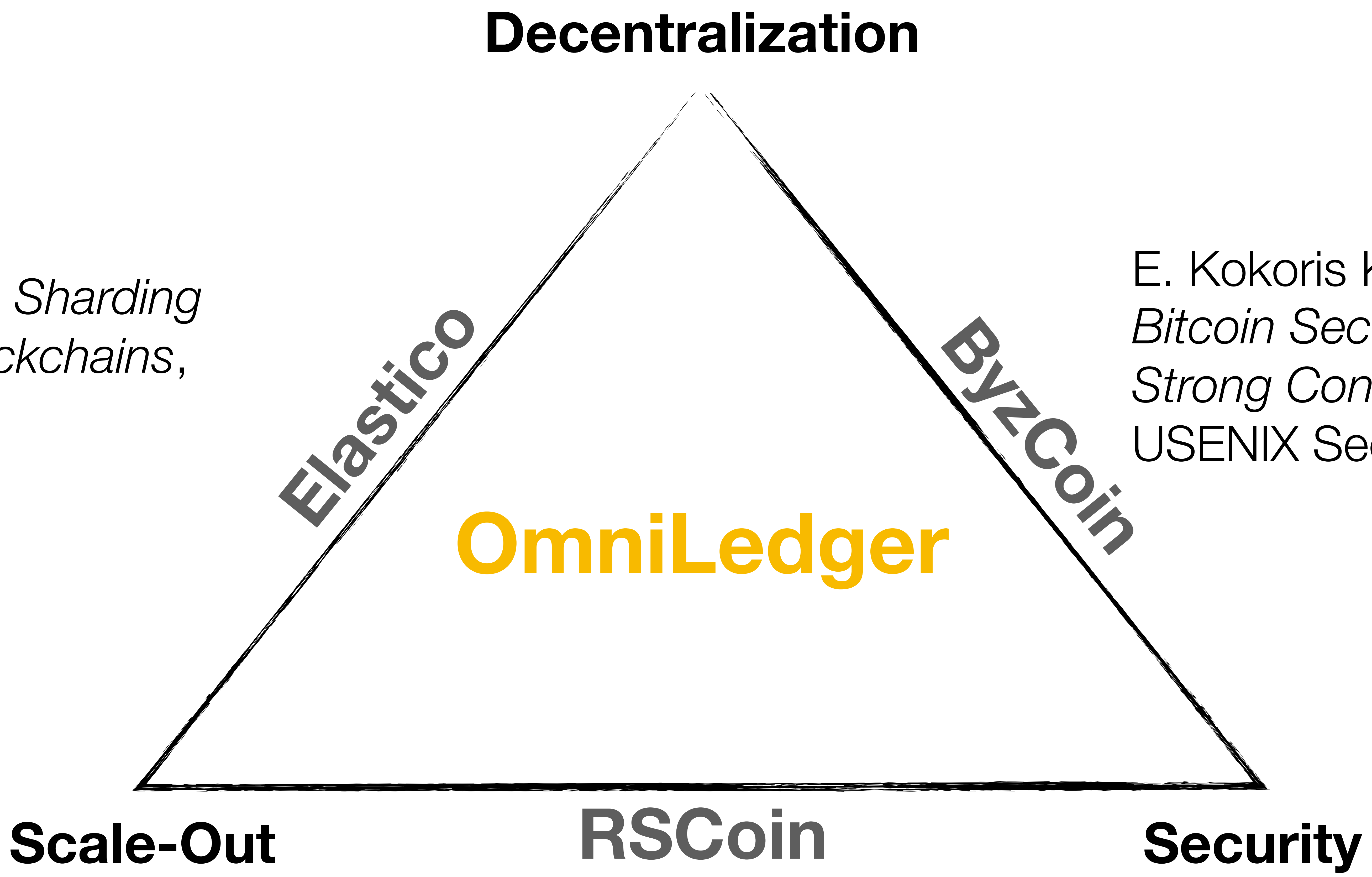
Scale-out: Throughput increases *linearly* with the available resources.

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



Decentralization

Elastico

ByzCoin

OmniLedger

Scale-Out

RSCoin

Security

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

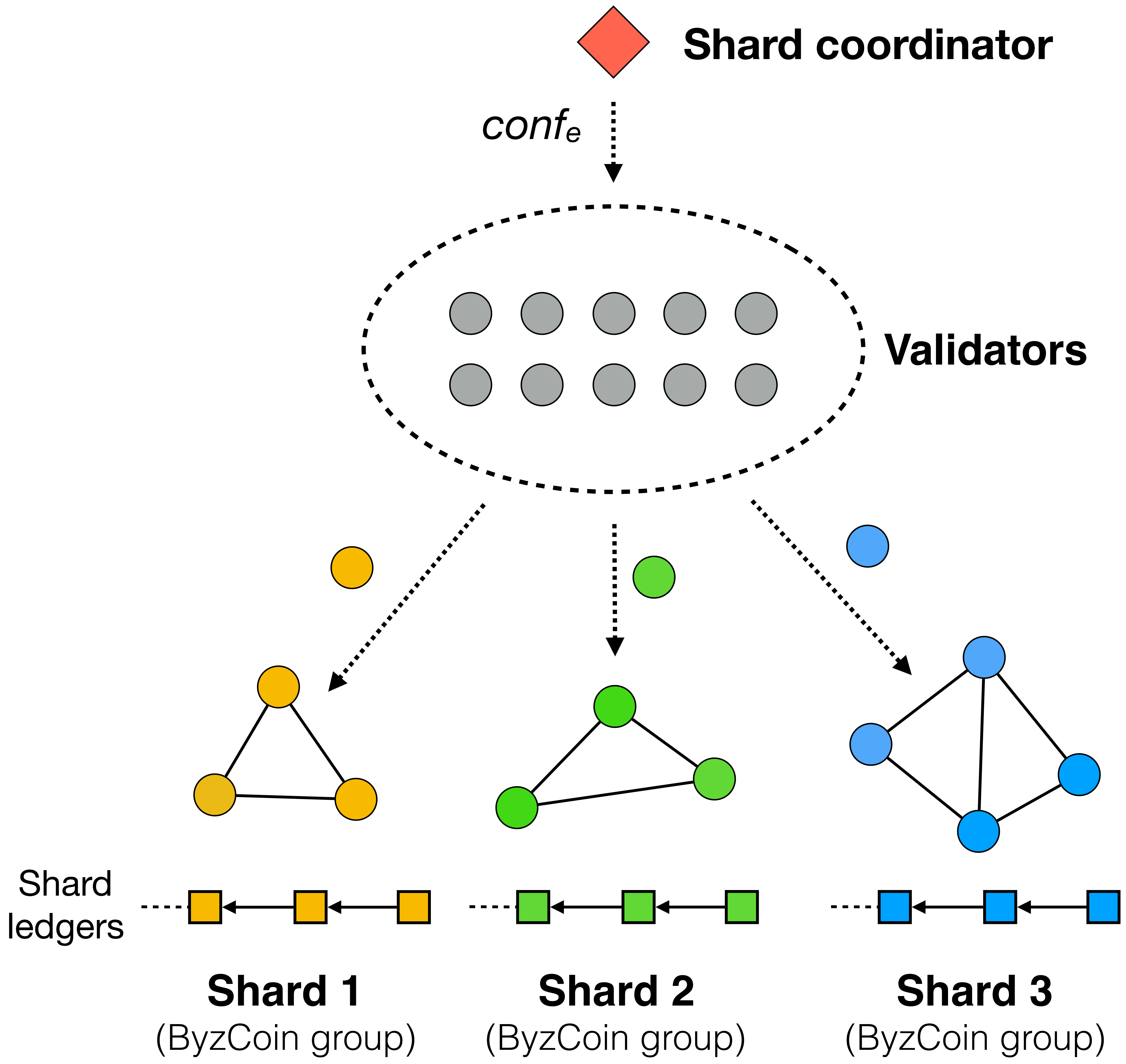
6. Low Latency
Tx are confirmed quickly

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



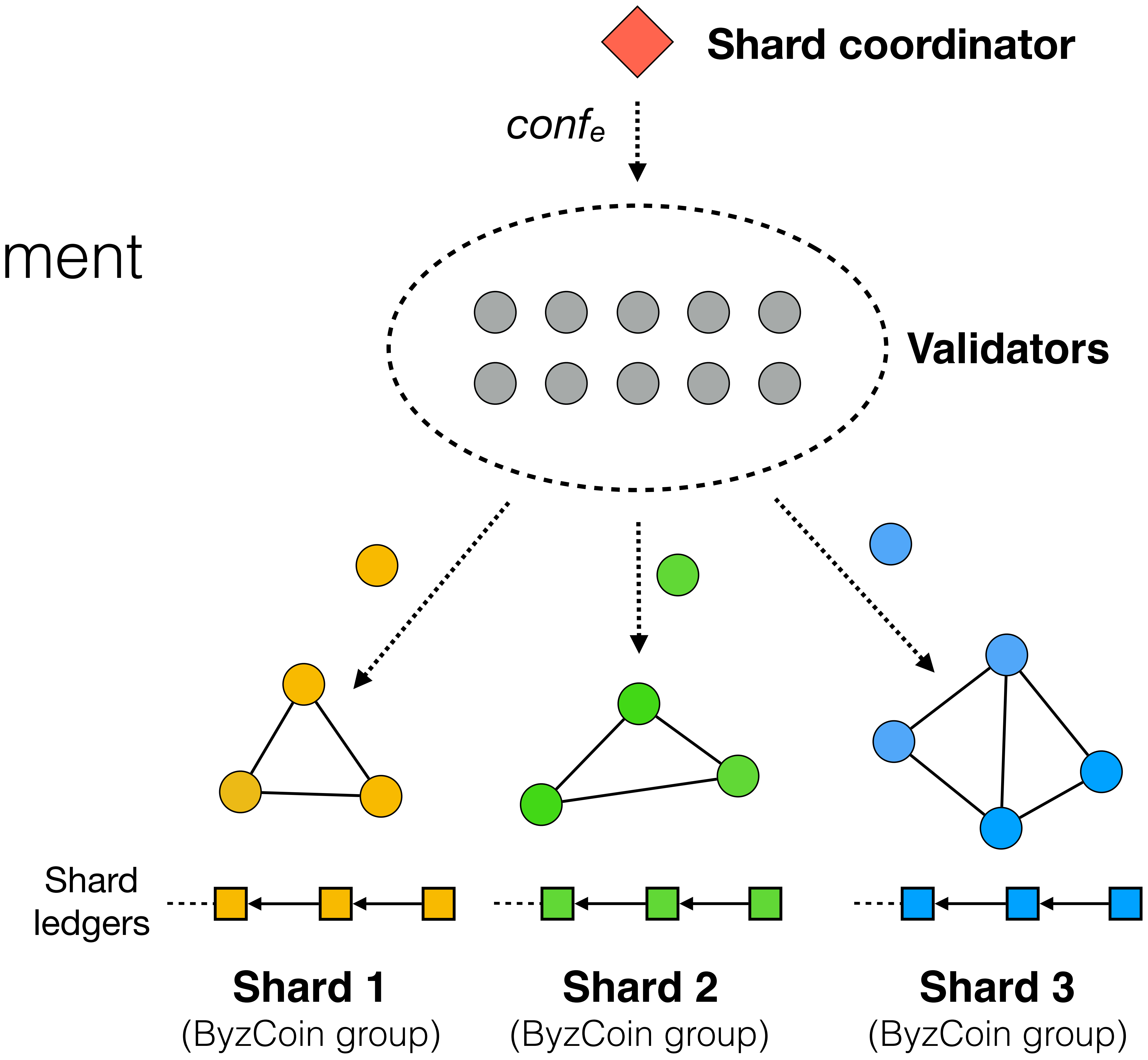
Strawman: SimpleLedger

Security Drawbacks

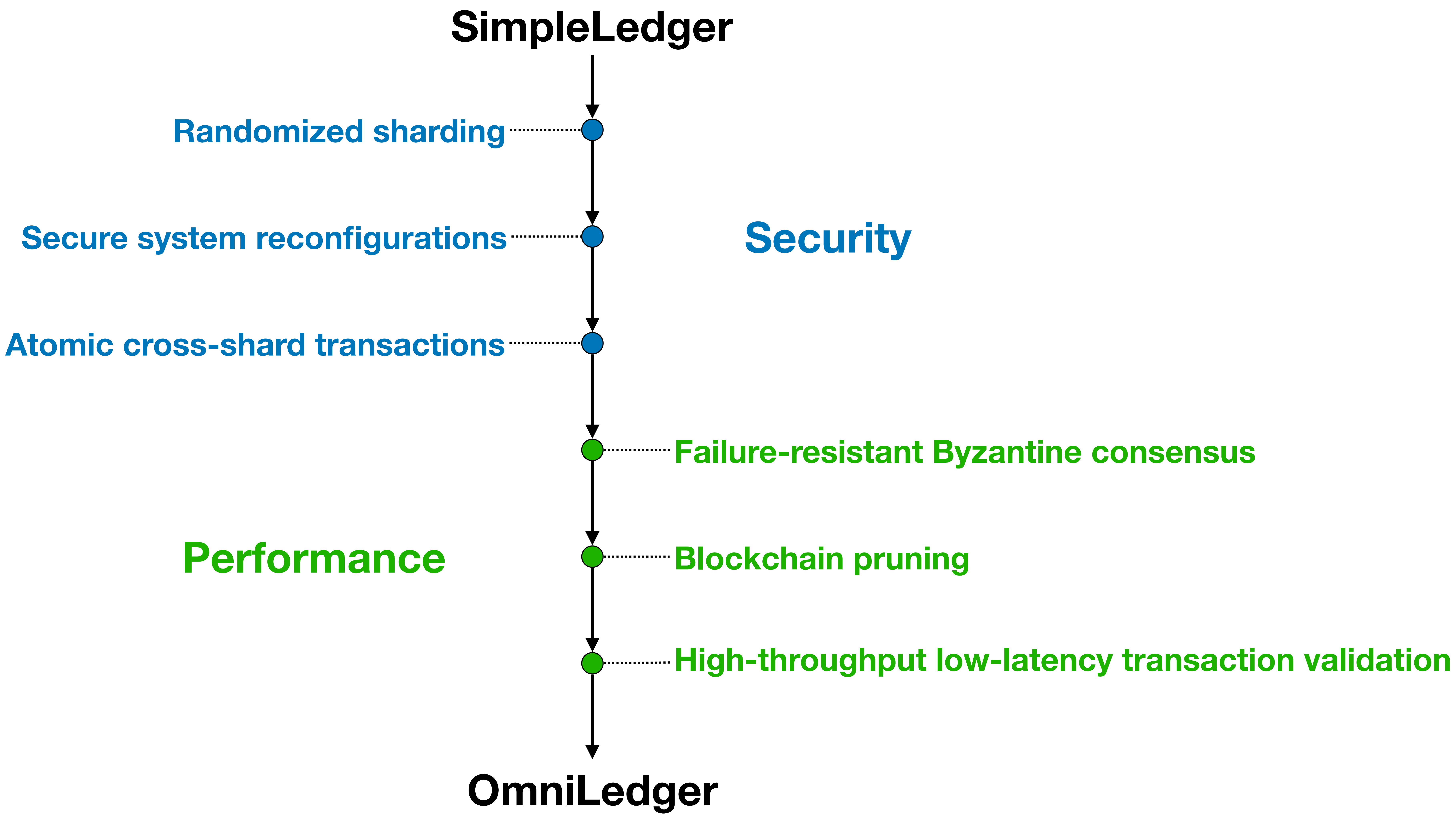
- Shard coordinator: trusted third party
- No tx processing during validator re-assignment
- No cross-shard tx support

Performance Drawbacks

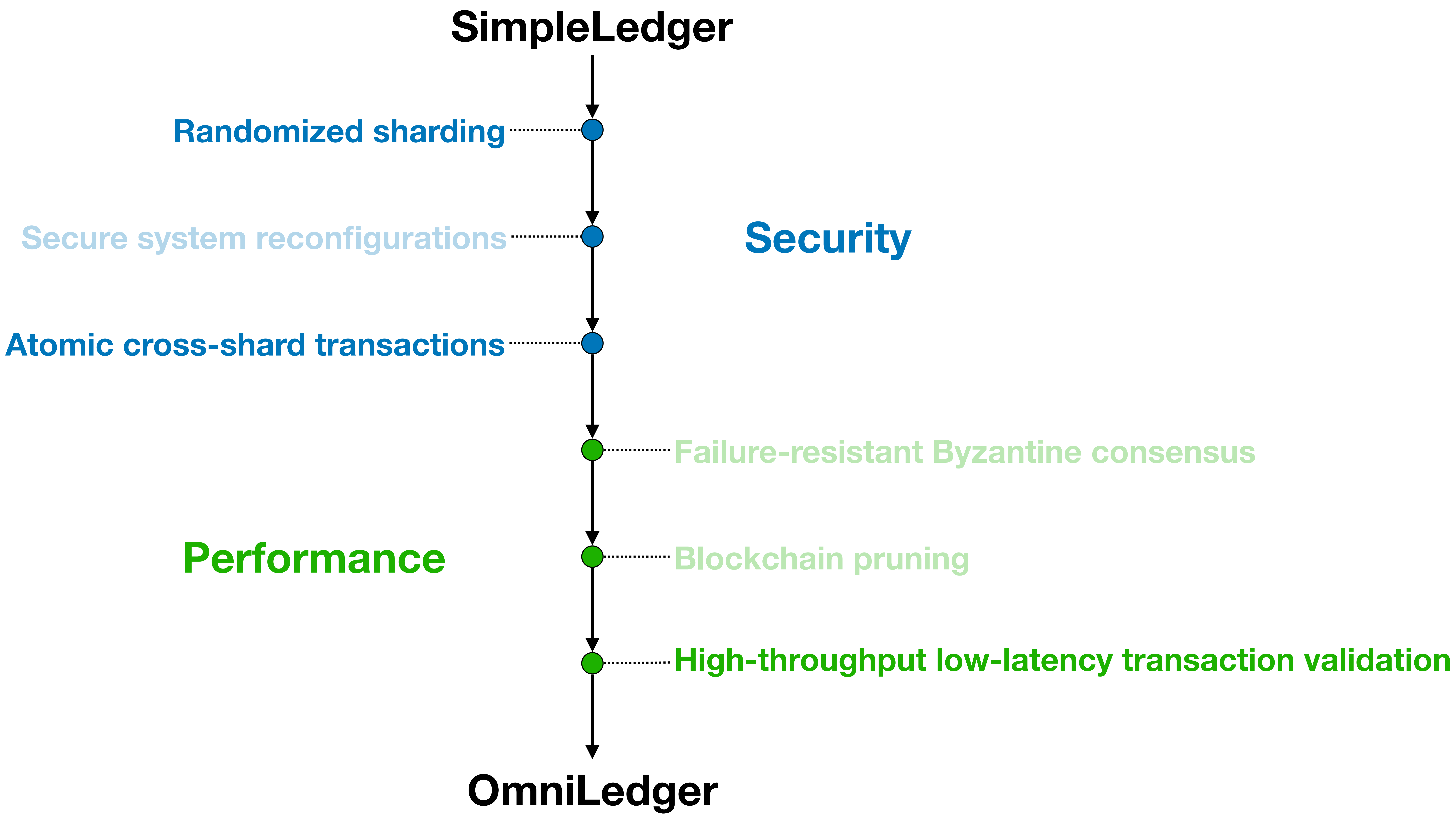
- ByzCoin failure mode
- High storage and bootstrapping cost
- Throughput vs. latency trade-off



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

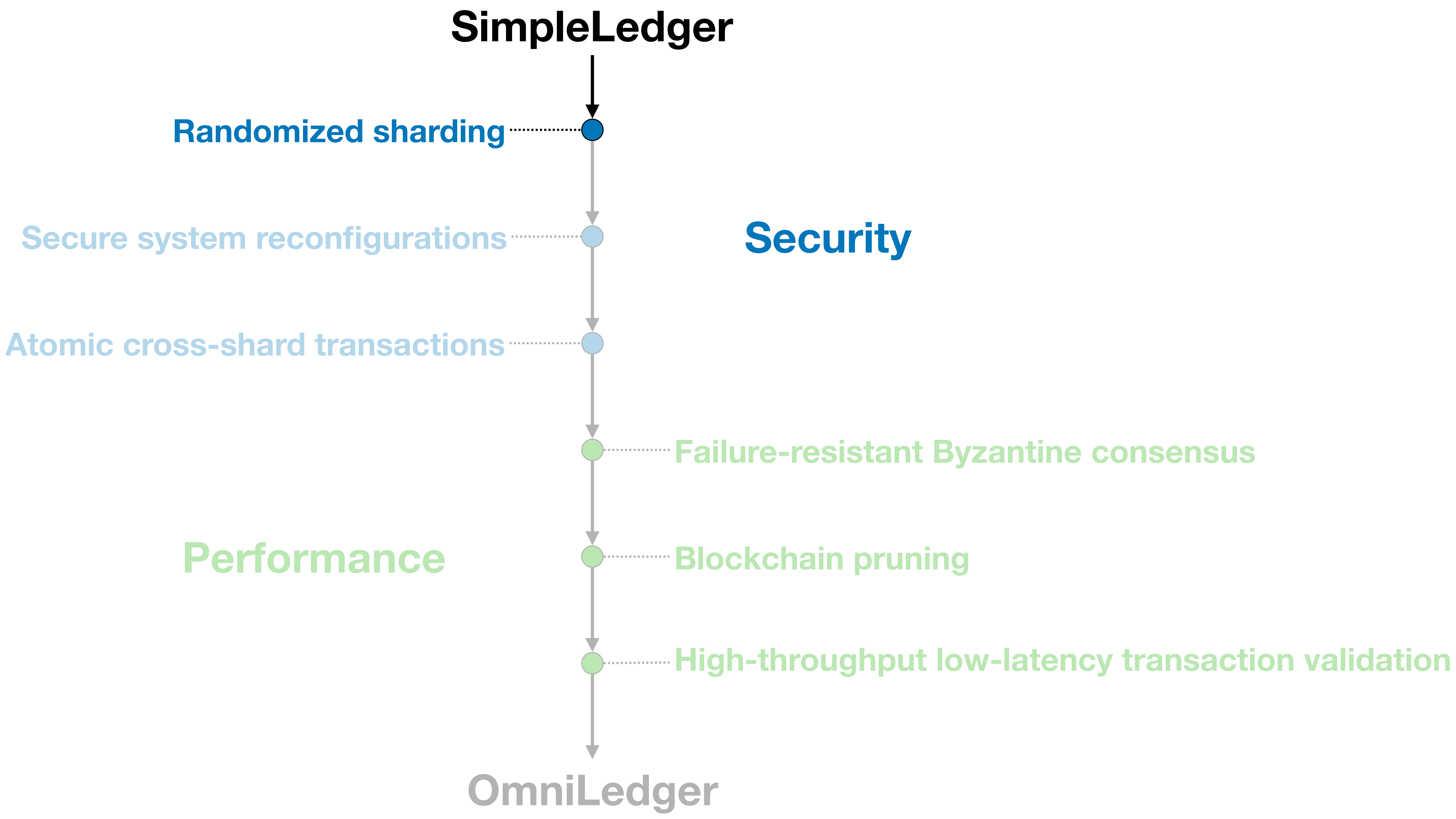
Blockchain pruning

High-throughput low-latency transaction validation

Performance

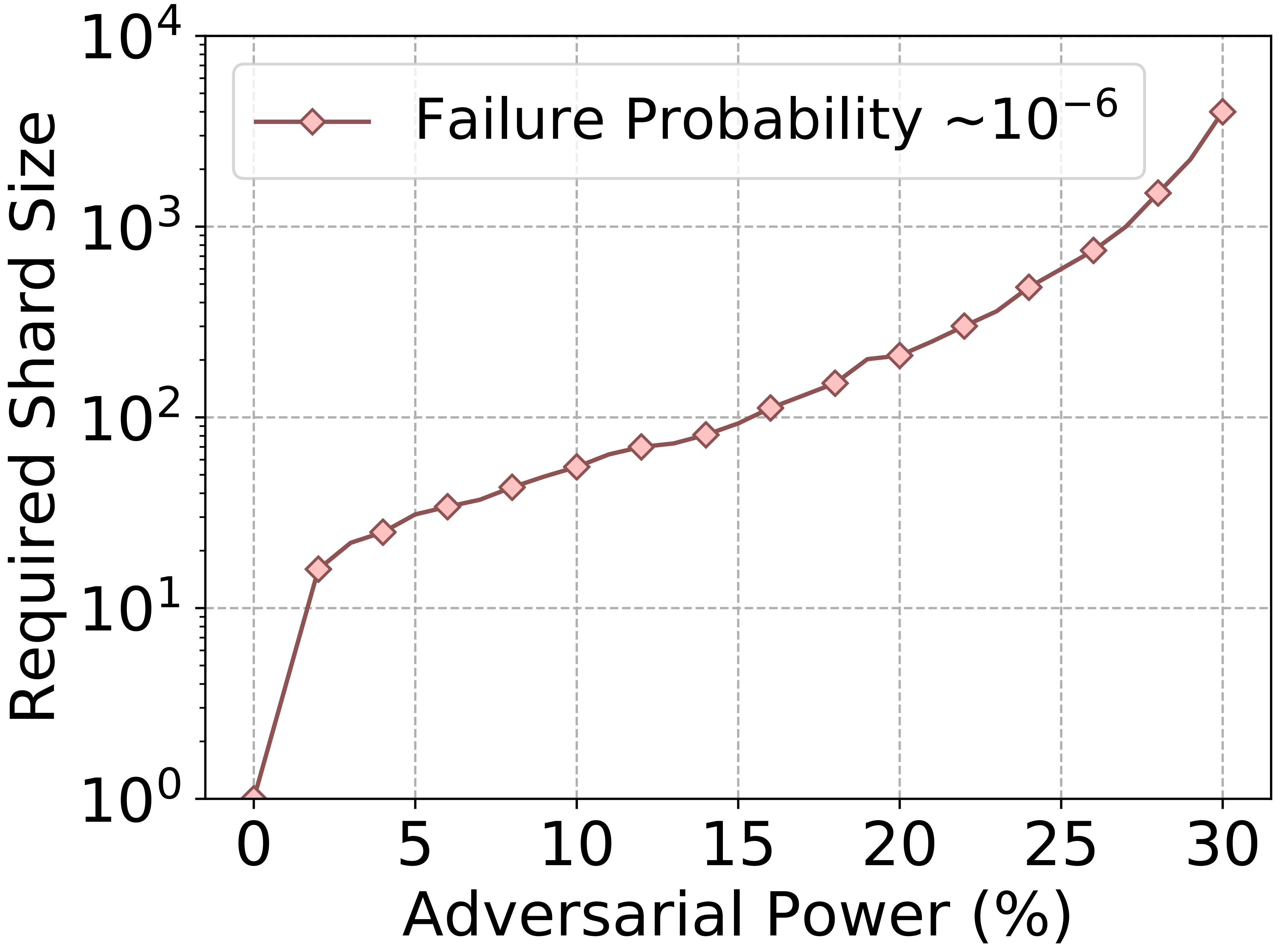
OmniLedger

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



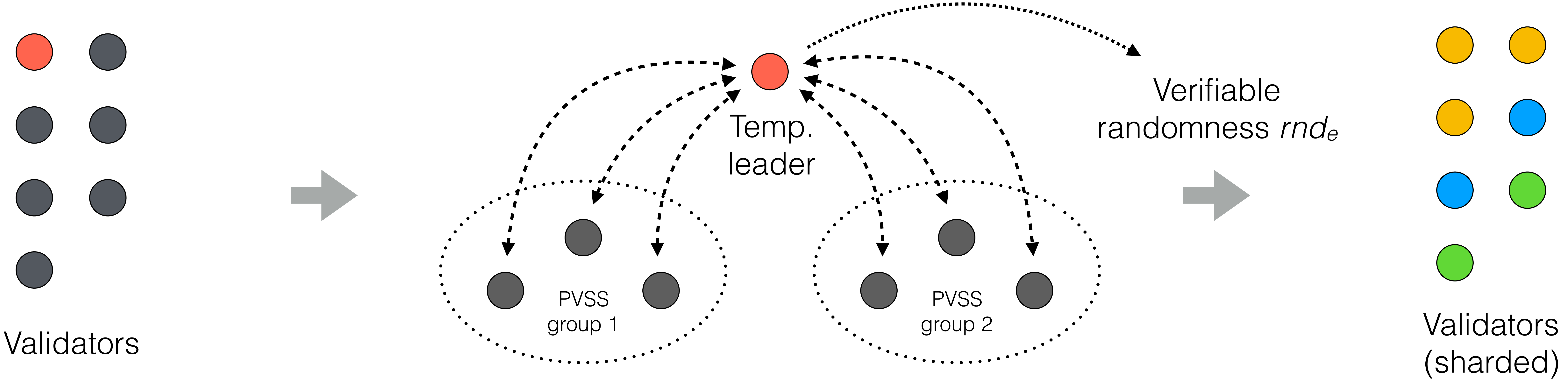
Shard Validator Assignment

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

1. Temp. leader election via VRFs (biasable)

2. Randomness generation via RandHound* (unbiasable)

3. Shard assignment (using rnd_e)



Validators

Temp. leader

PVSS group 1

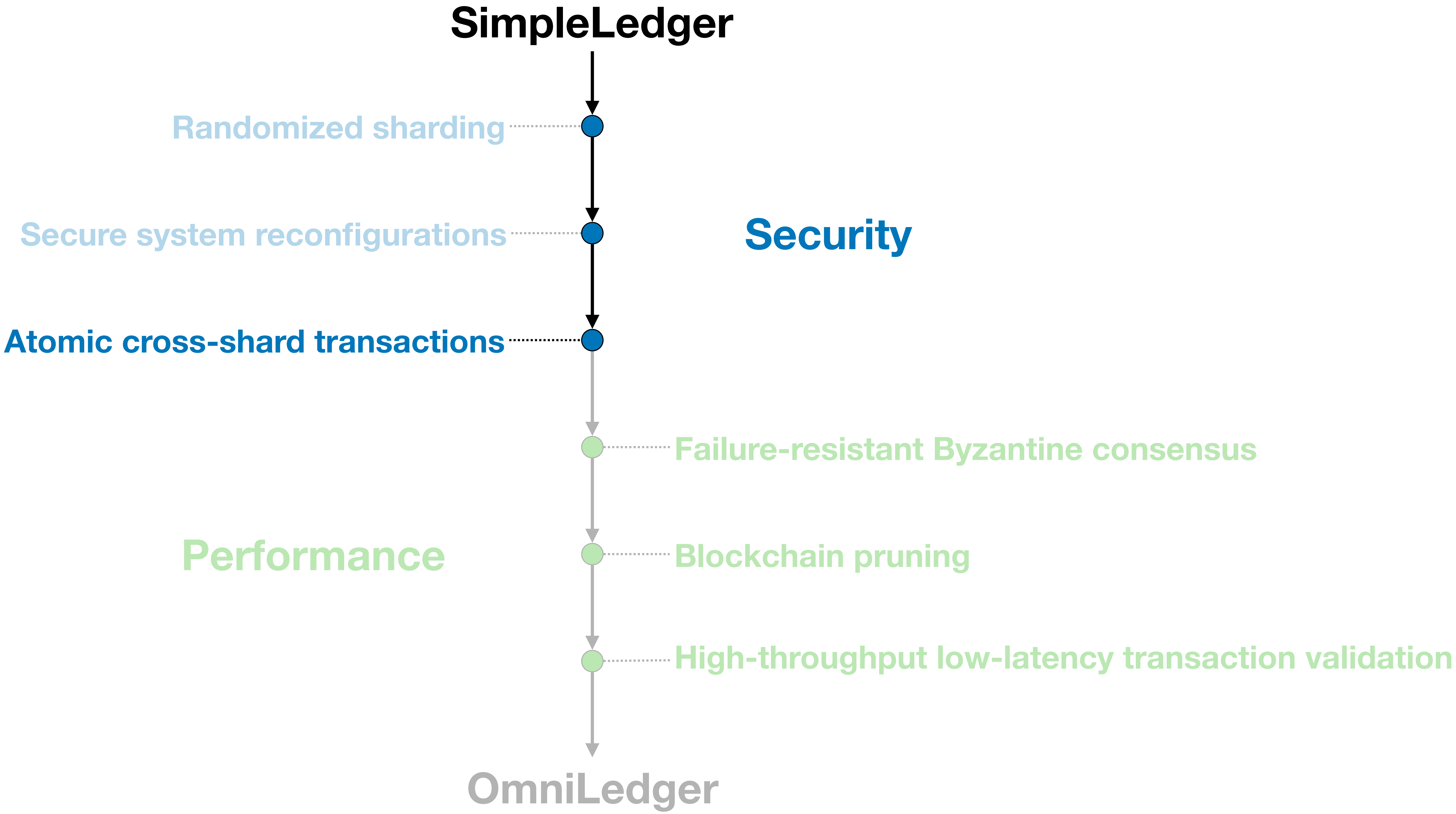
PVSS group 2

Verifiable randomness rnd_e

Validators (sharded)

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

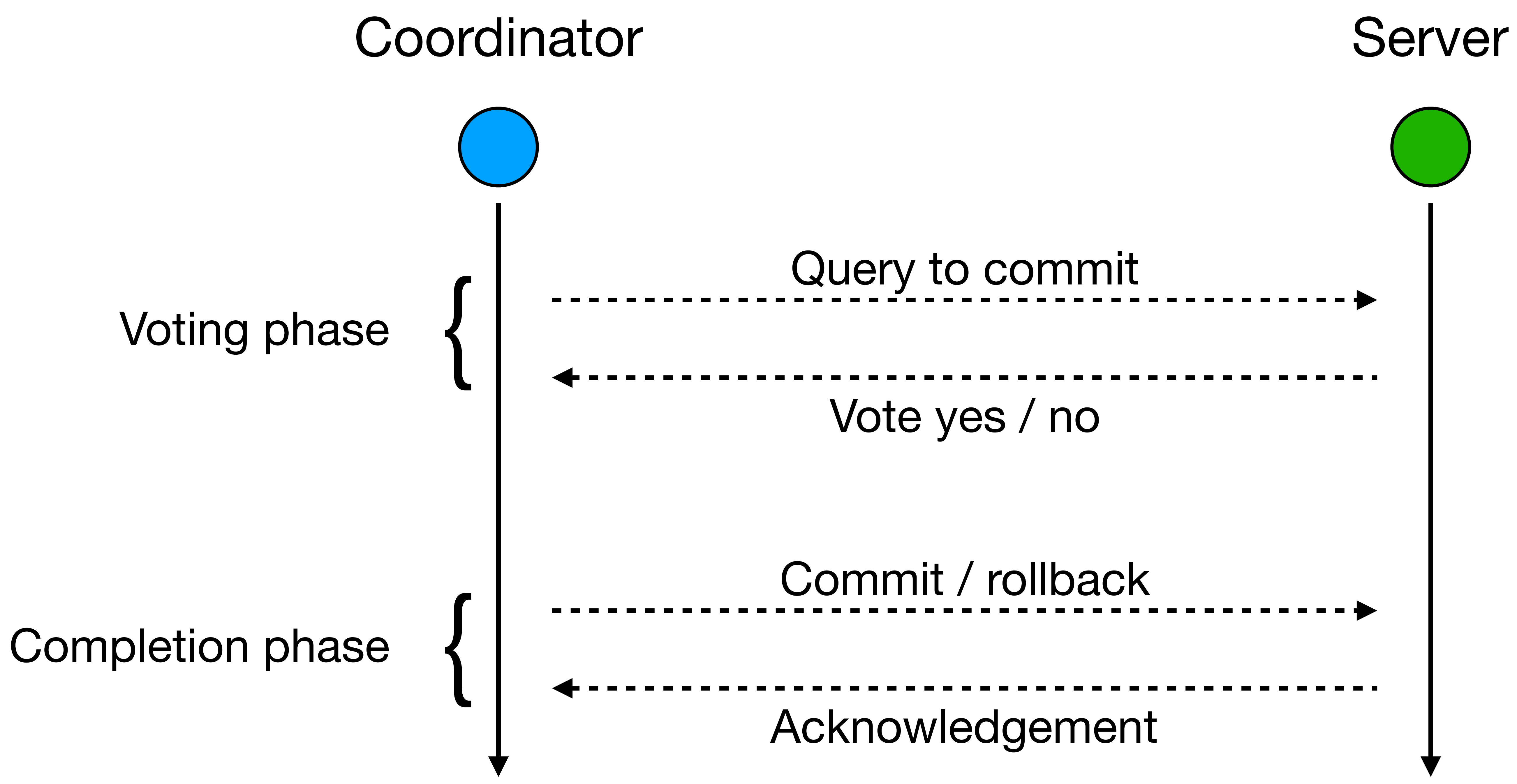
Performance

Blockchain pruning

High-throughput low-latency transaction validation

OmniLedger

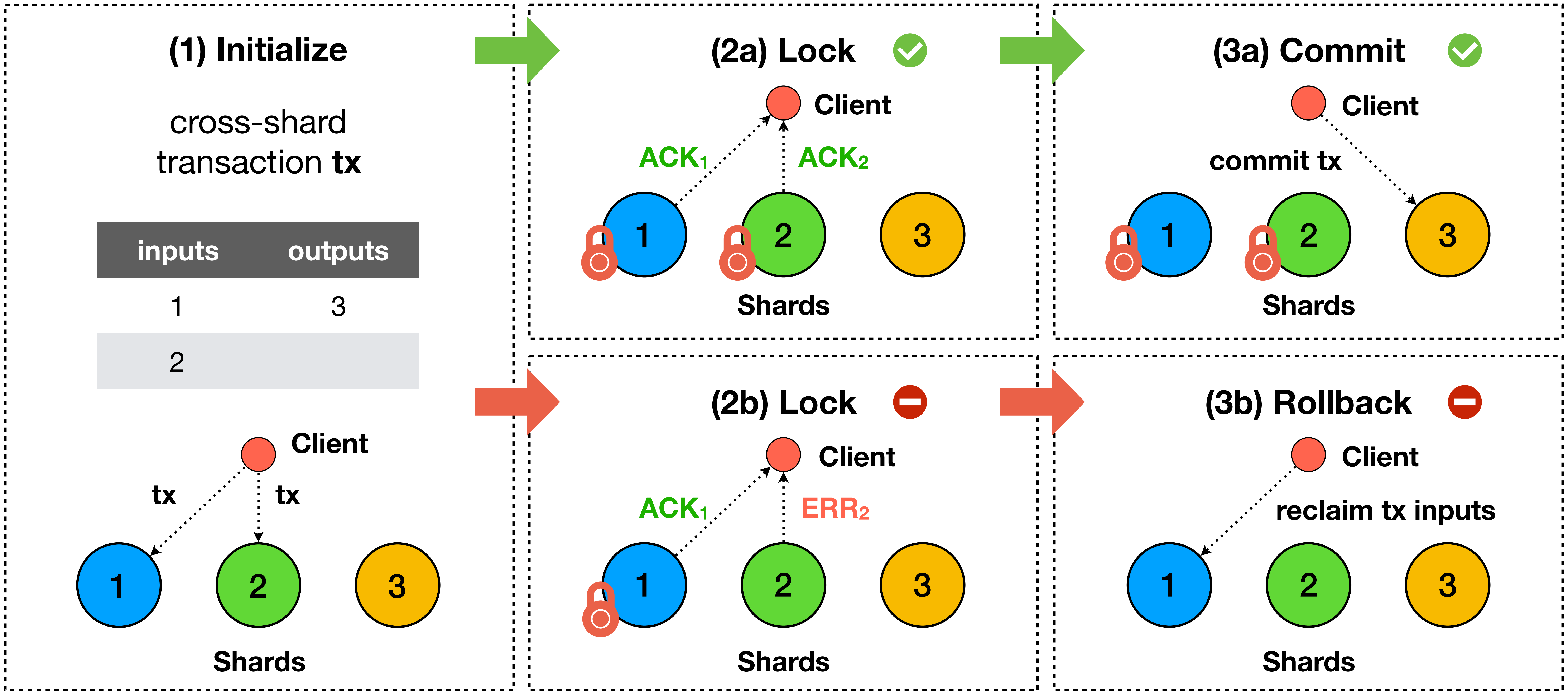
Two-Phase Commits



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

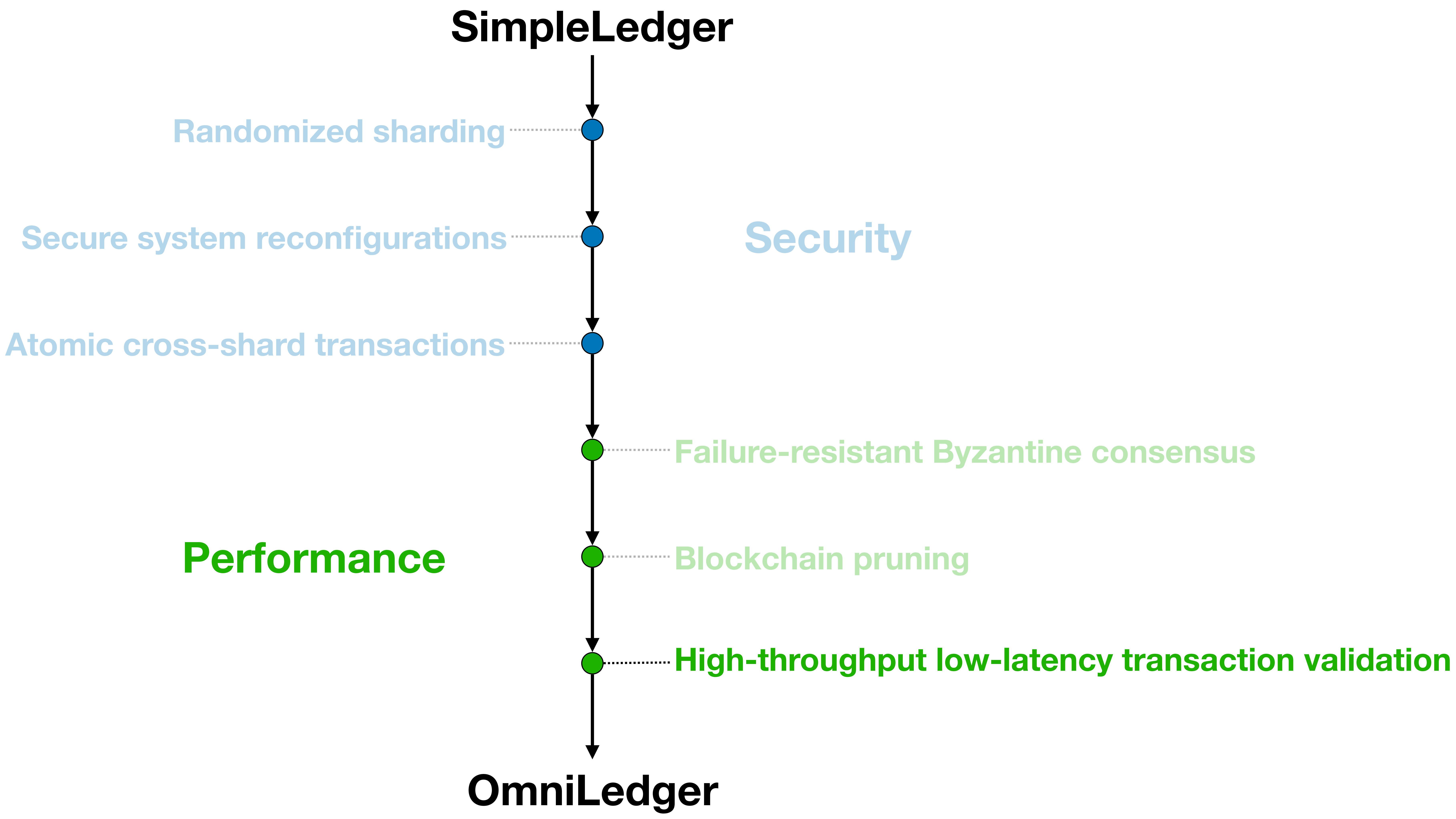
Atomix: Secure Cross-Shard Transactions

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



inputs	outputs
1	3
2	

Roadmap



SimpleLedger

Randomized sharding

Secure system reconfigurations

Atomic cross-shard transactions

Security

Failure-resistant Byzantine consensus

Performance

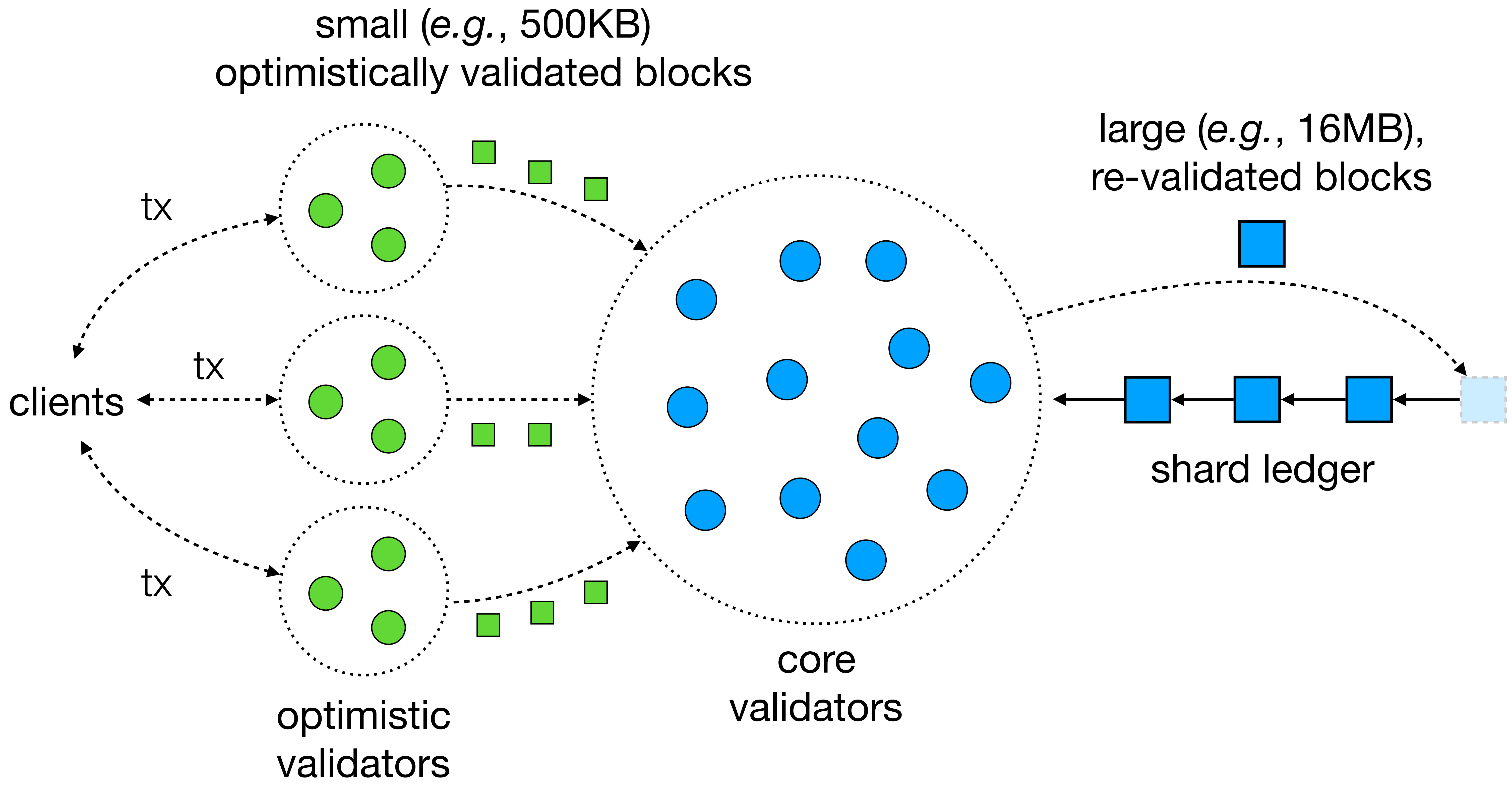
Blockchain pruning

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

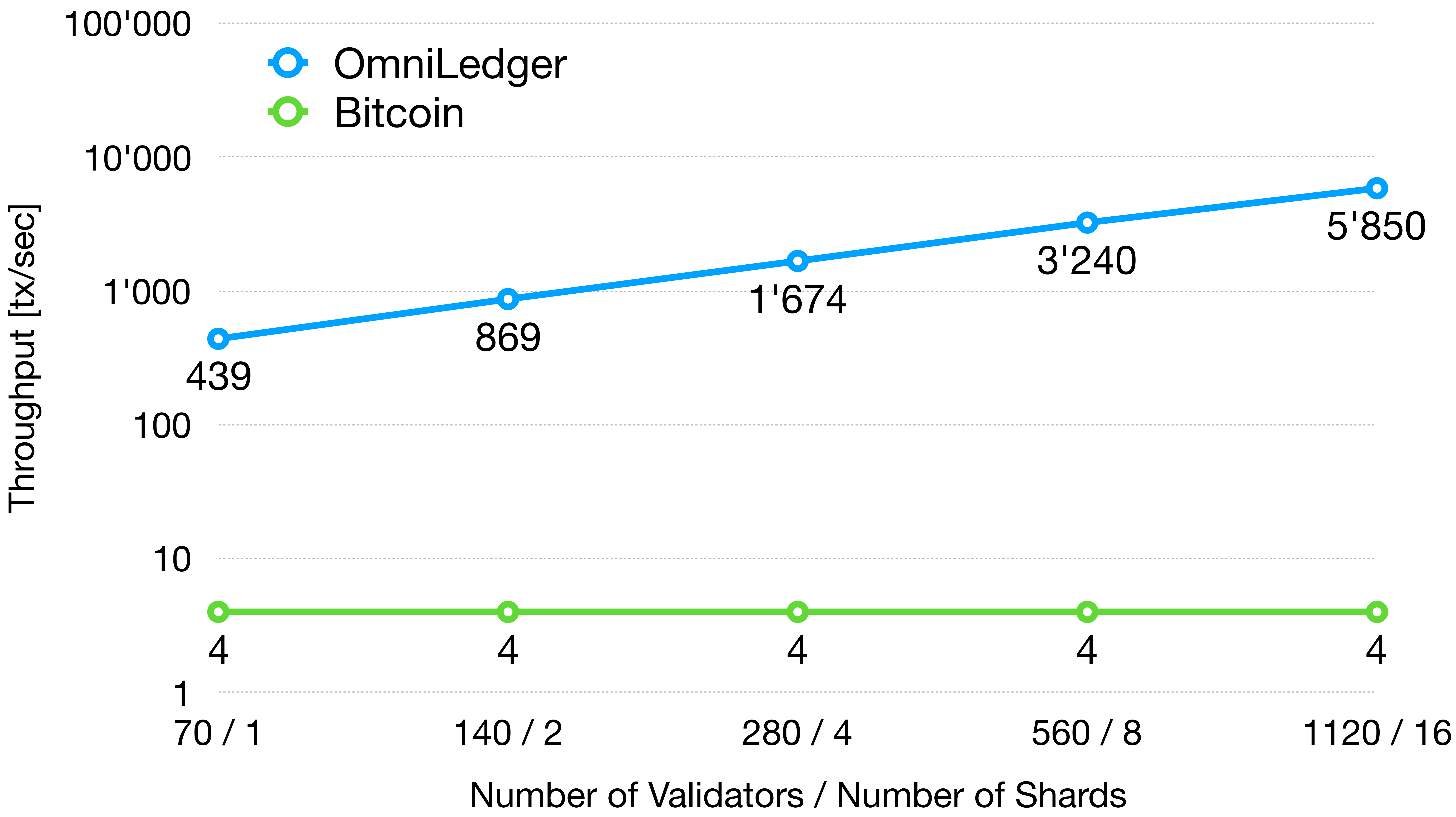
Implementation

- 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

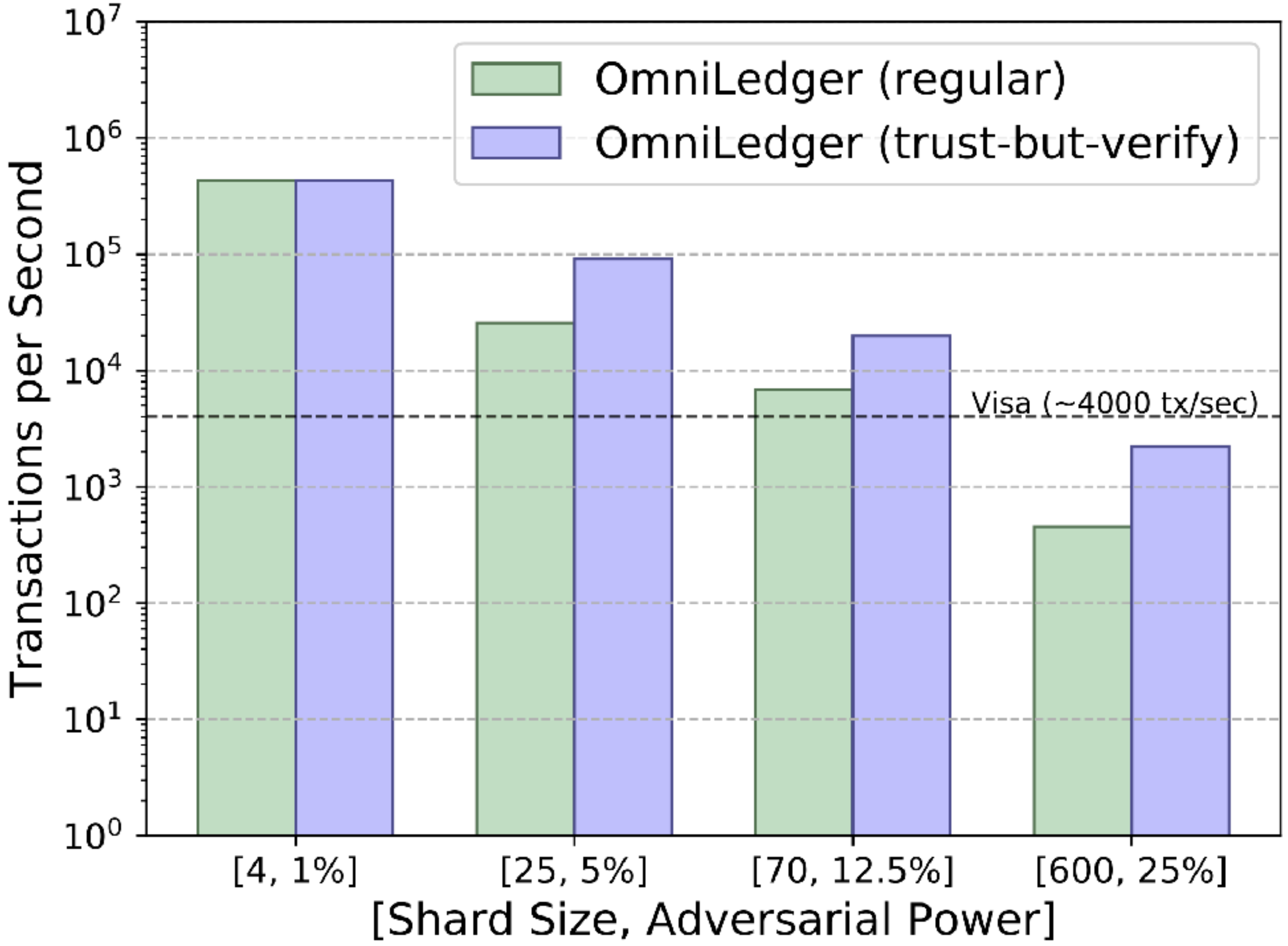
- 48 physical machines
  - Intel Xeon E5-2420 v2 (6 cores @ 2.2 GHz)
  - 24 GB RAM
  - 10 Gbps network link
- Realistic network configurations
  - 20 Mbps bandwidth
  - 200 ms round-trip latency

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	

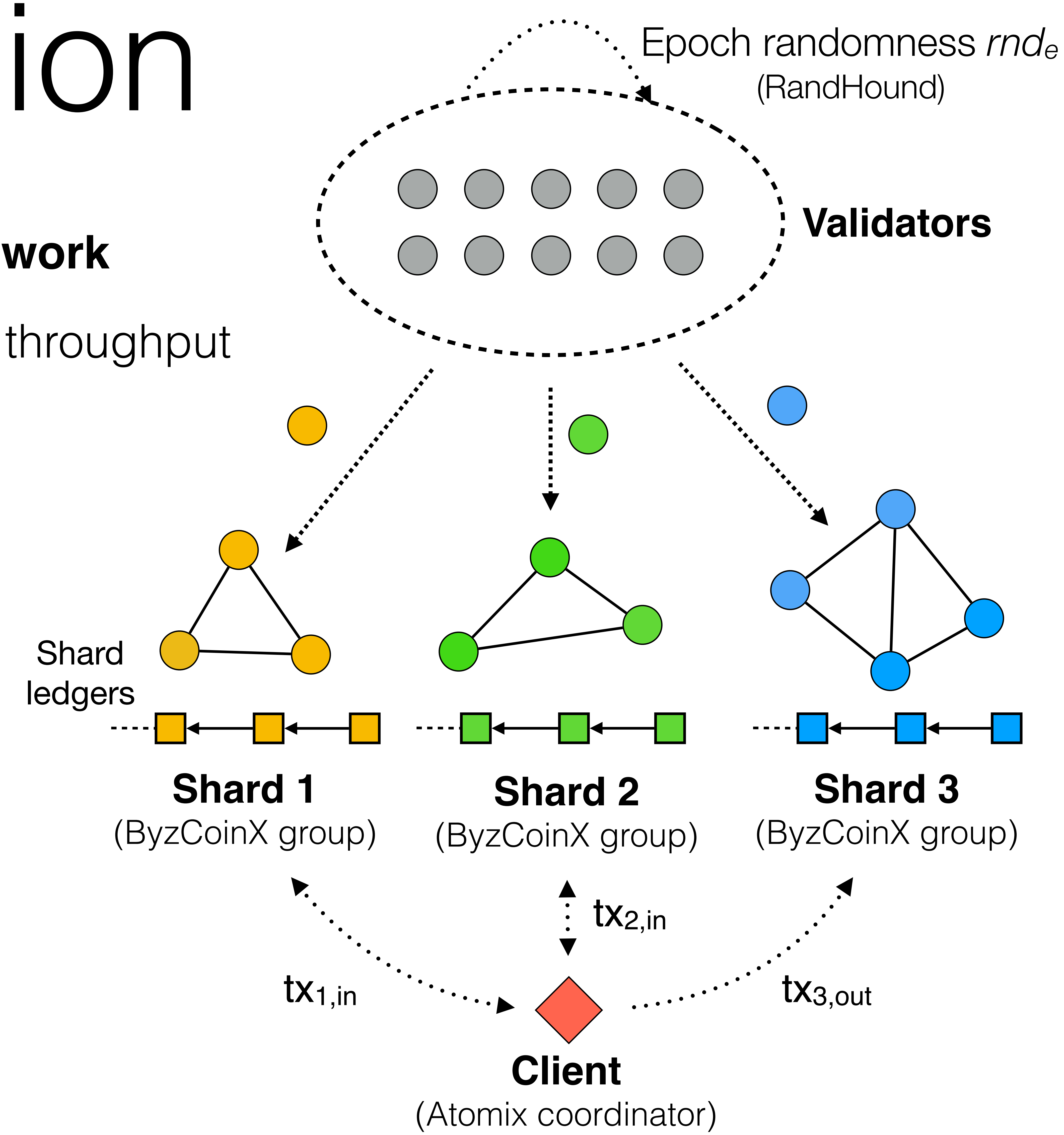
latency increase since optimistically validated blocks are batched into larger blocks for final validation to get better throughput

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