OmniLedger: A Secure, Scale-Out, Decentralized Ledger

Philipp Jovanovic

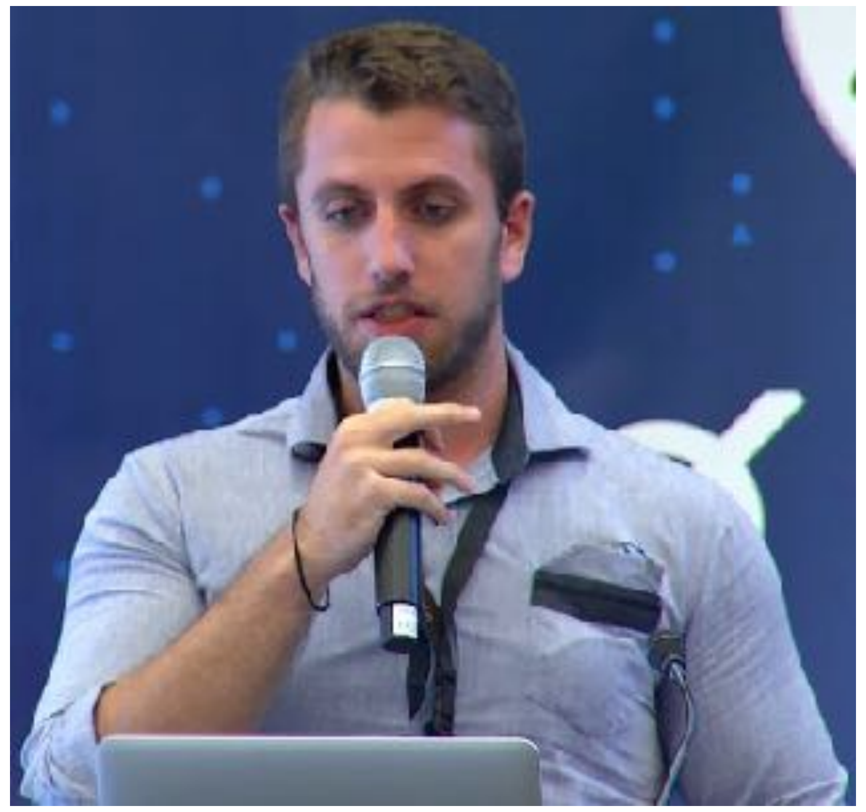
Decentralized and Distributed Systems Lab (DEDIS)

Swiss Federal Institute of Technology Lausanne (EPFL)

Binary District

2018-02-15, London

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

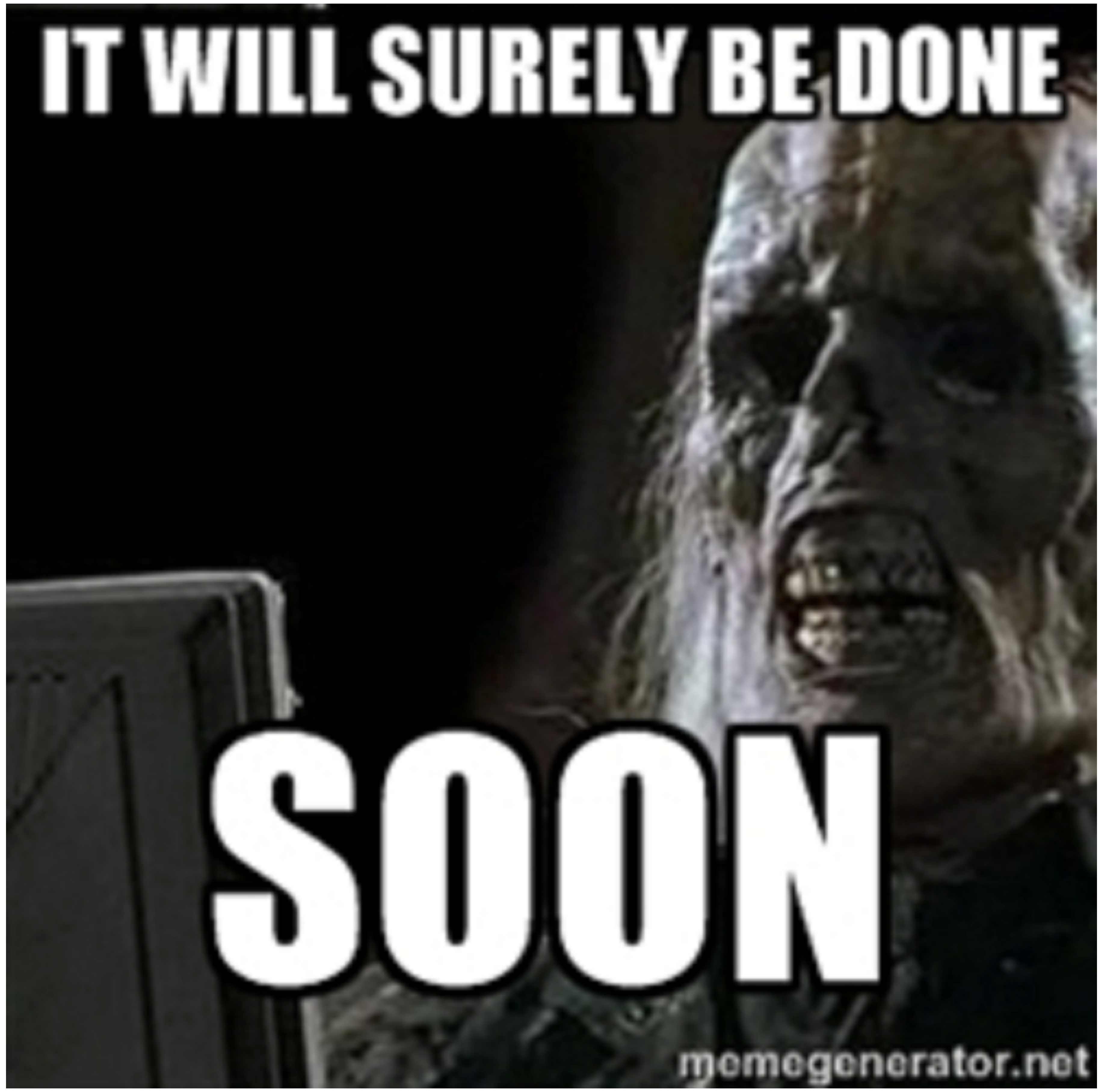
- Motivation
- OmniLedger
- Evaluation
- Conclusion

Talk Outline

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

Drawbacks of Nakamoto Consensus

- Transaction confirmation delay
  - Bitcoin: Any tx takes >10 mins until being confirmed
- Weak consistency
  - Bitcoin: You are not really certain your tx is committed until you wait >1 hour
- Low throughput
  - Bitcoin: ~7 tx/sec
- Proof-of-work mining
  - Wastes huge amount of energy



Scaling Blockchains is More Important Than Ever …

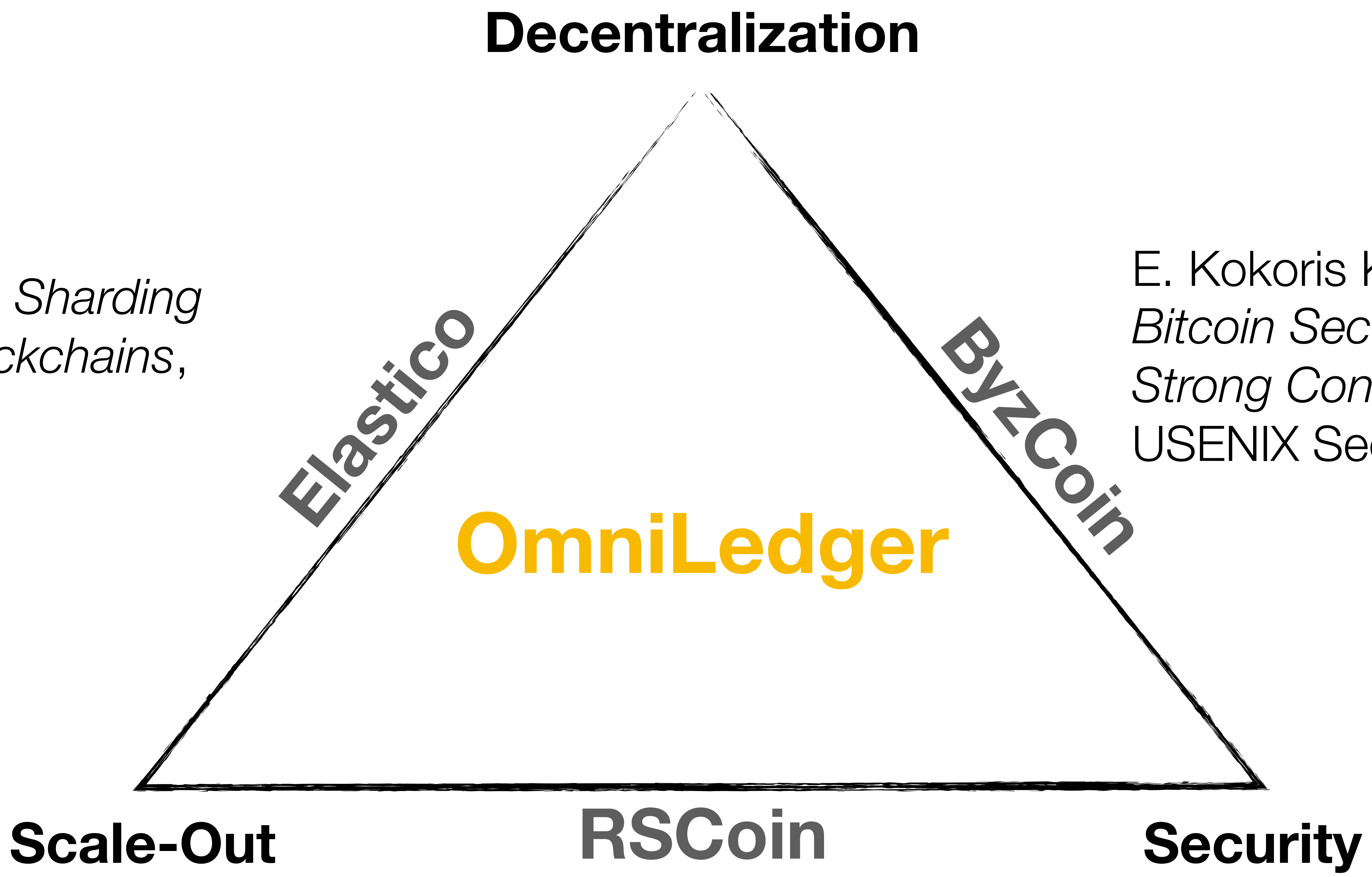
CATS RULE THE BLOCKCHAIN, TOO

The ethereum network is getting jammed up because people are rushing to buy cartoon cats on its blockchain

… But Scaling Blockchains is Not Easy

Distributed Ledger Landscape

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



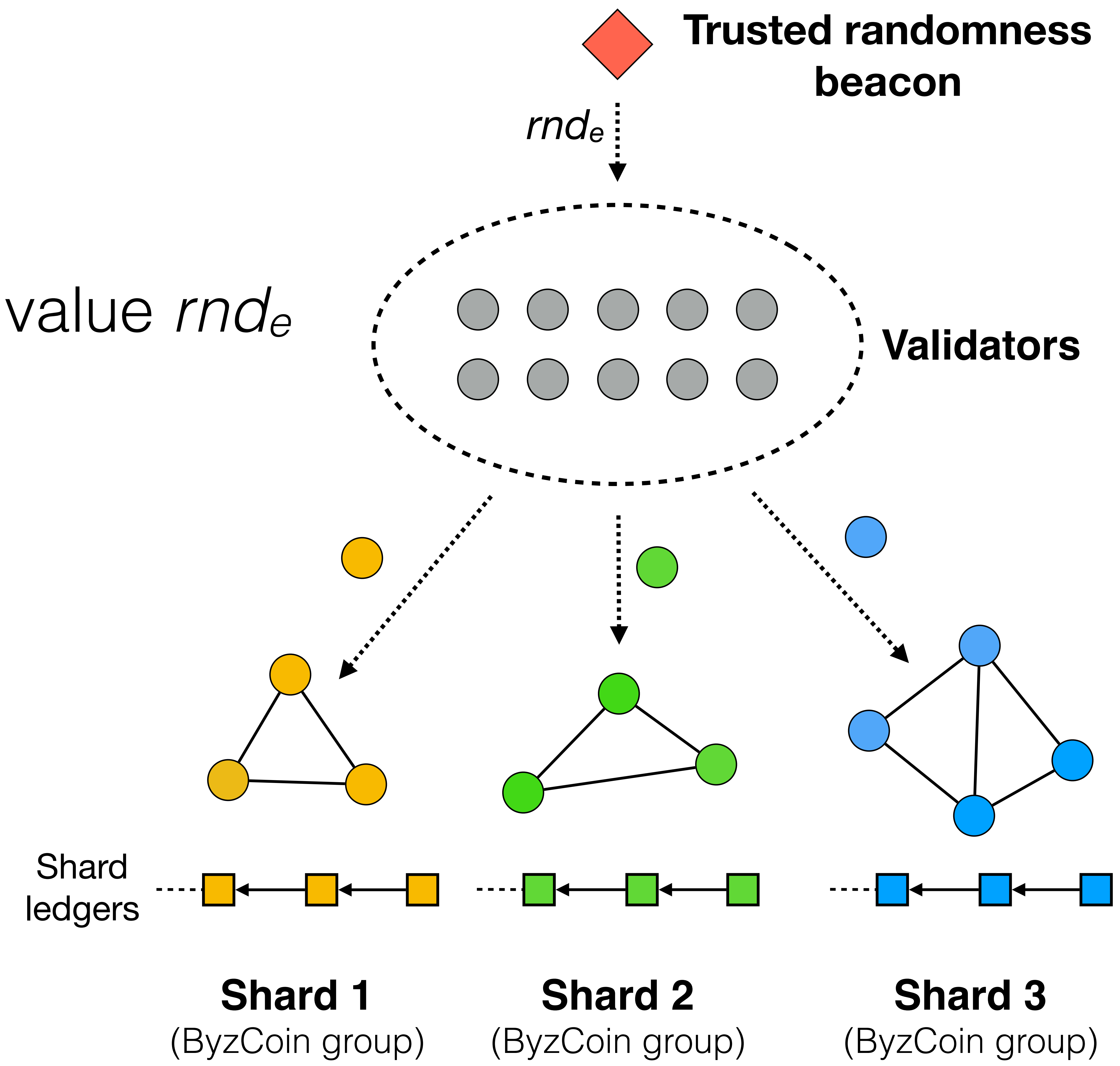
E. Kokoris Kogias et al., *Enhancing Bitcoin Security and Performance with Strong Consistency via Collective Signing*, USENIX Security 2016

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

Strawman: SimpleLedger

Overview

- Evolves in epochs *e*
- Trusted randomness beacon emits random value rnd_e
- Validators:
  - Use rnd_e to compute shard assignment (ensures shard security)
  - Bootstrap from the shard ledger
  - Process tx using consensus (e.g., ByzCoin)



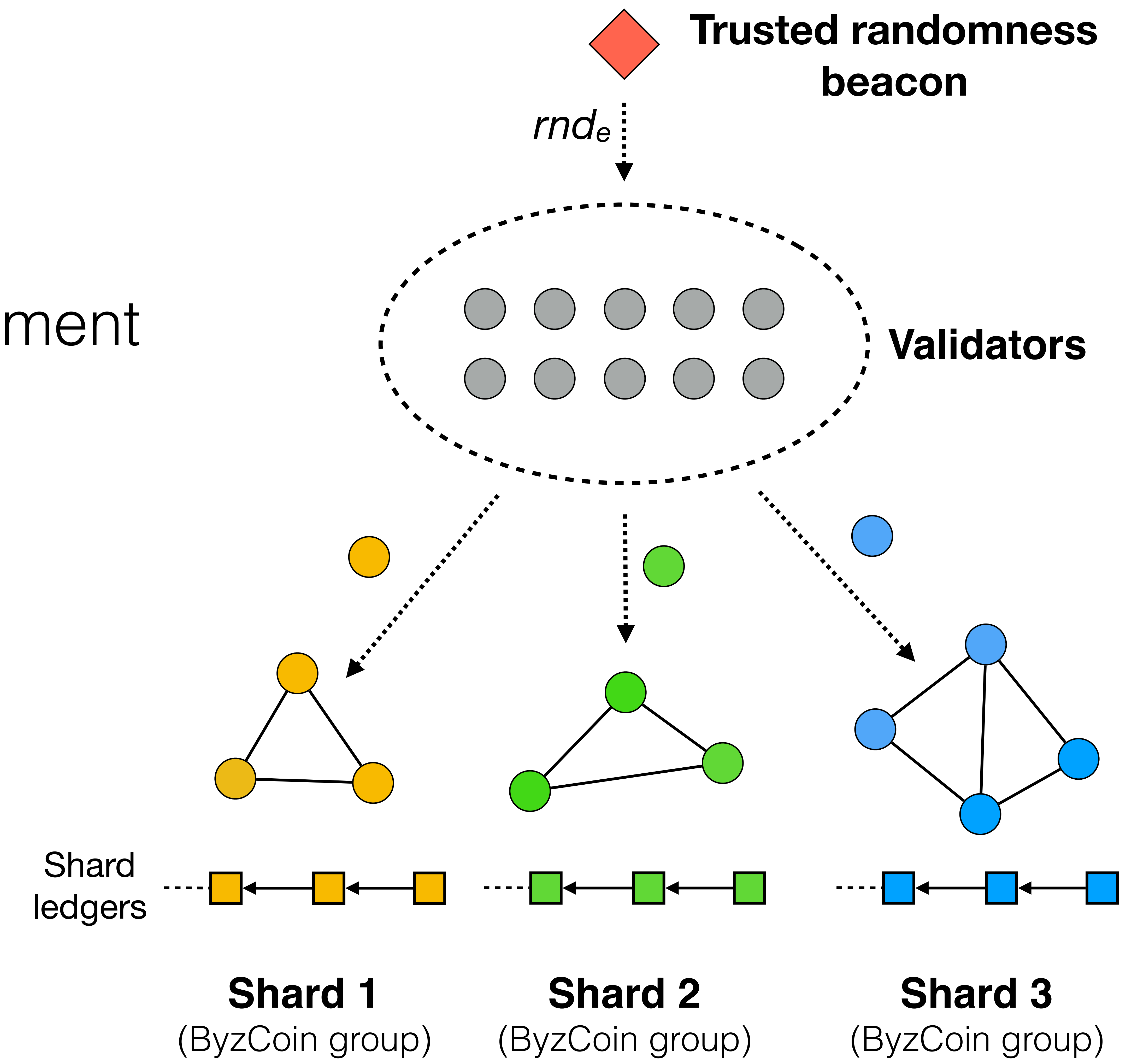
Strawman: SimpleLedger

Security Drawbacks

- Randomness beacon: 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



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

Roadmap

SimpleLedger
Sharding via distributed randomness
Selective validator re-assignment: Robust epoch transitions
Security
Atomix: Client-managed atomic cross-shard txs
ByzCoinX: Robust BFT consensus
Performance
Shard ledger pruning: Reduce storage & bootstrapping cost
Trust-but-verify validation: No throughput vs latency trade-off
OmniLedger

Roadmap

SimpleLedger

Sharding via distributed randomness

Selective validator re-assignment: Robust epoch transitions

Atomix: Client-managed atomic cross-shard txs

Security

ByzCoinX: Robust BFT consensus

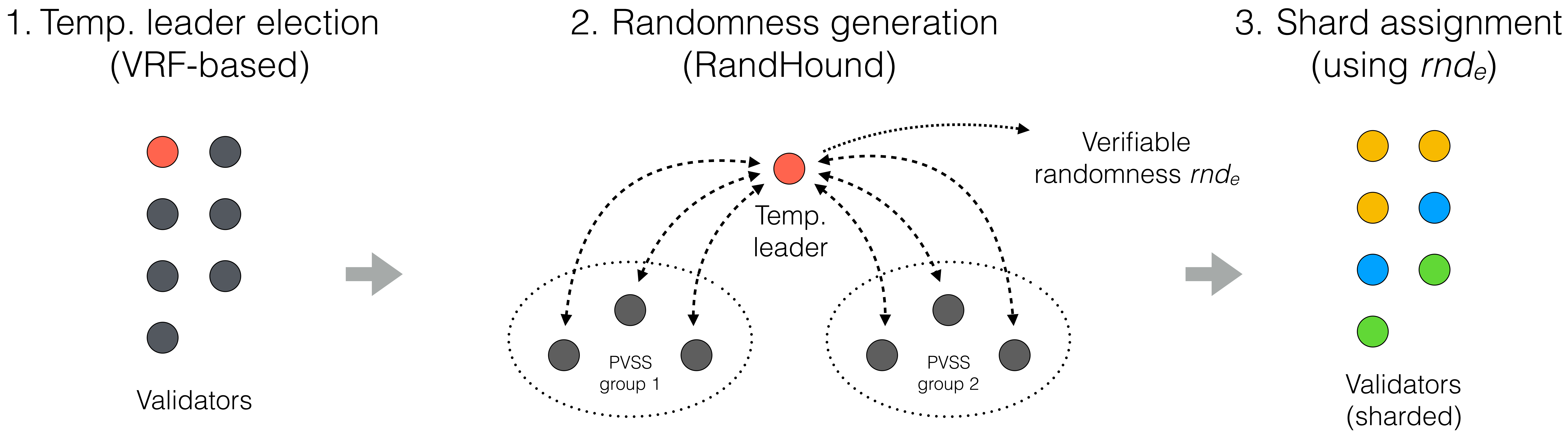
Shard ledger pruning: Reduce storage & bootstrapping cost

Trust-but-verify validation: No throughput vs latency trade-off

Performance

OmniLedger

Shard Validator Assignment



Challenge:

- Prevent (adaptive) adversary from subverting an entire shard with high probability

Solution:

- Periodically re-assign validators to shards using unbiasable, publicly-verifiable randomness

Robust Epoch Transitions

Challenge:

- Full validator re-assignment & bootstrapping enforces system halt during epoch transitions

Solution:

- For n validators & shard number m fix swap-out batch size $k < 1/3 \times n/m$ (e.g., $k = \log(n/m)$)
- Compute random permutation for j-th shard seeded by $H(j \,||\, rnd_e)$
- Re-assign lowest k validators evenly across m shards
- Similar approach for new validators using seed $H(0 \,||\, rnd_e)$
- Ensures BFT consensus security/liveness since $> 2/3 \times n/m$ honest validators per shard

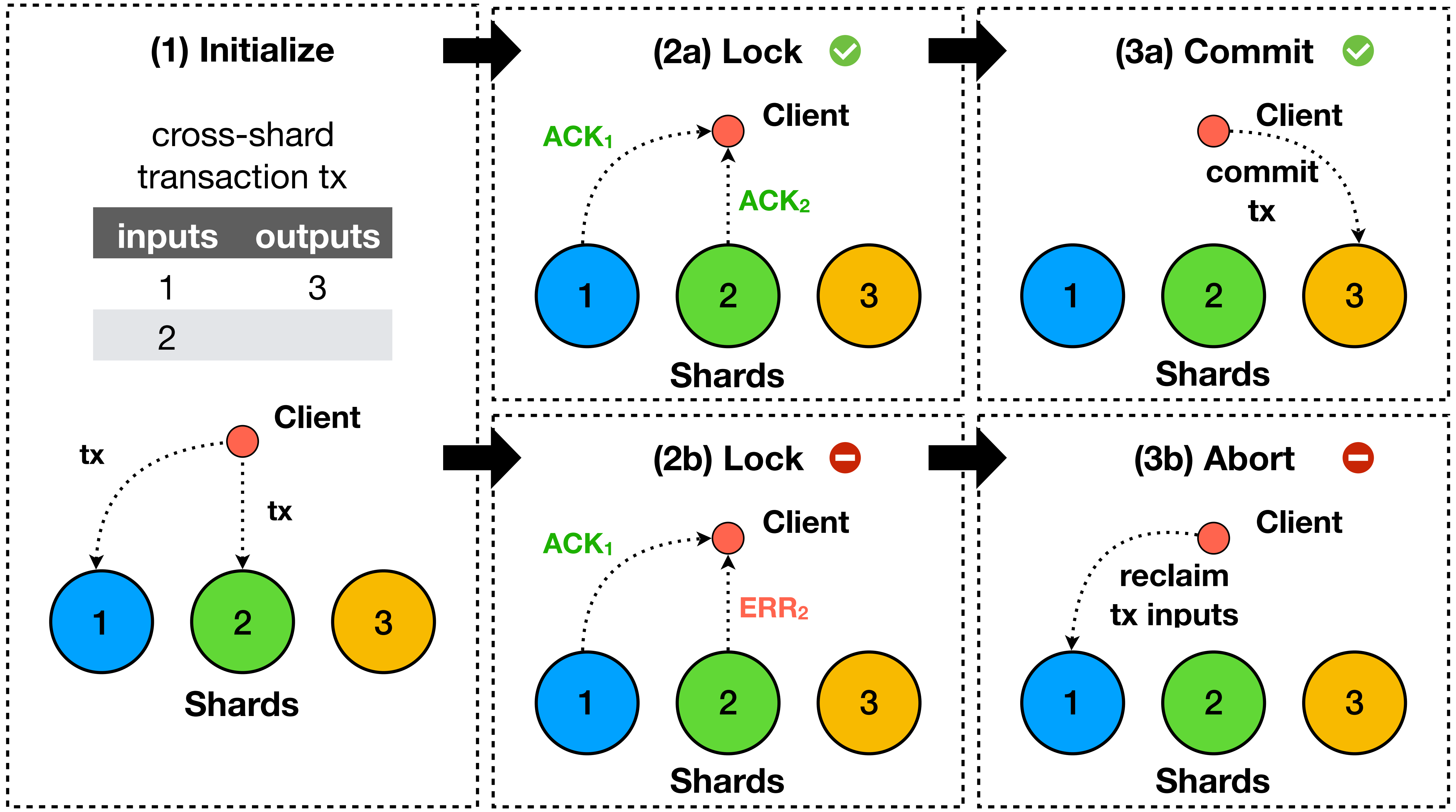
Atomix: Cross-Shard Transactions

Challenge:

- Cross-shard tx commit atomically or abort eventually

Solution: Atomix

- Client-managed protocol
  1. Client sends cross-shard tx to input shards
  2. Collect ACK/ERR proofs from input shards
  3. (a) If all input shards accept, commit to output shard, otherwise (b) abort and reclaim input funds
- Optimistically trust client for liveness
- Collective signing (CoSi) ensures compact proofs



The Atomix protocol for secure cross-shard transactions

Roadmap

SimpleLedger

Security

- Sharding via distributed randomness
- Selective validator re-assignment: Robust epoch transitions
- Atomix: Client-managed atomic cross-shard txs

Performance

- ByzCoinX: Robust BFT consensus
- Shard ledger pruning: Reduce storage & bootstrapping cost
- Trust-but-verify validation: No throughput vs latency trade-off

OmniLedger

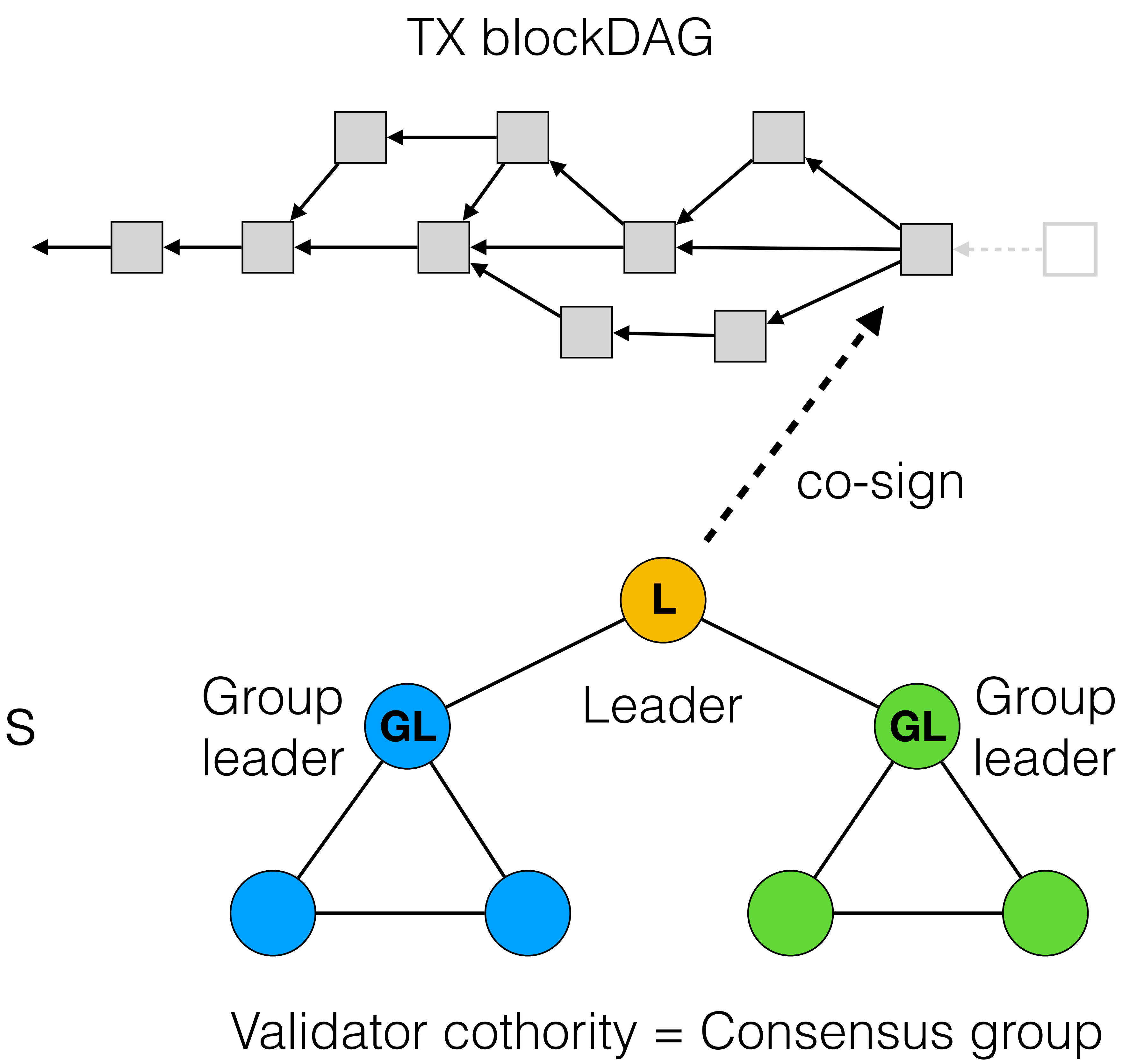
ByzCoinX: Consensus

Challenge:

- Ensure shard state consistency (process tx, etc.)

Solution: ByzCoinX

- Variant of ByzCoin
- Group- instead of tree-based communication
    - Trade-off some scalability for higher fault tolerance
    - Performs better for practically relevant configurations
- BlockDAG instead of blockchain
    - Capture dependencies between txs
    - Better performance due to better resource utilization



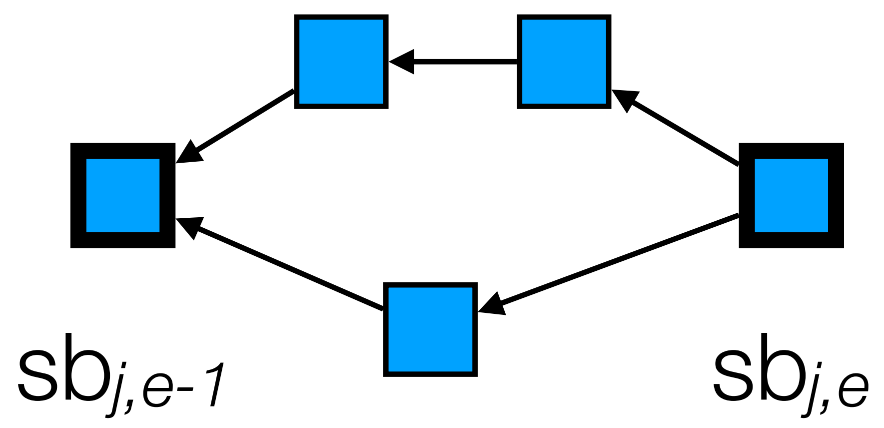
Shard Ledger Pruning

Challenge:

- High storage & bootstrapping cost for validators in high-throughput systems

Solution:

- State block $sb_{j,e}$ summarizes state of shard j at the end of epoch e
- $sb_{j,e}$ stores UTXOs in an order Merkle tree
- Validators joining shard j in epoch e bootstrap from $sb_{j,e-1}$
- Drastically reduces storage and bootstrap cost



Shard ledger
with state blocks

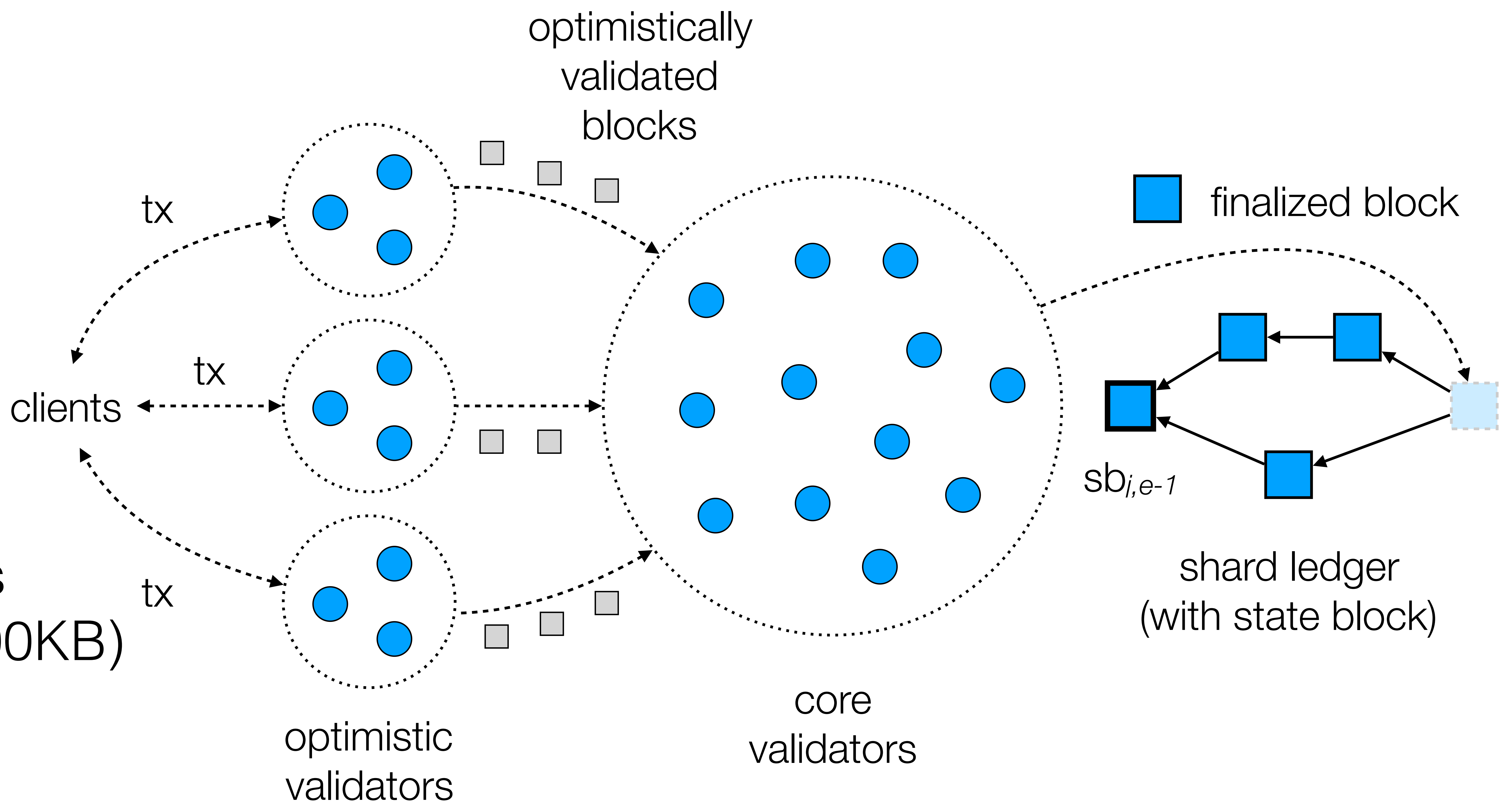
Trust-but-Verify Transaction Validation

Challenge:

- Latency vs. throughput trade-off

Solution:

- Two-level “trust-but-verify” validation
- Low latency:
  - Optimistically validate transactions batched into small blocks (*e.g.*, 500KB)
- High throughput:
  - Batch optimistically validated blocks into bigger blocks (*e.g.*, 16MB) and re-validate



Talk Outline

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

Implementation & Experimental Setup

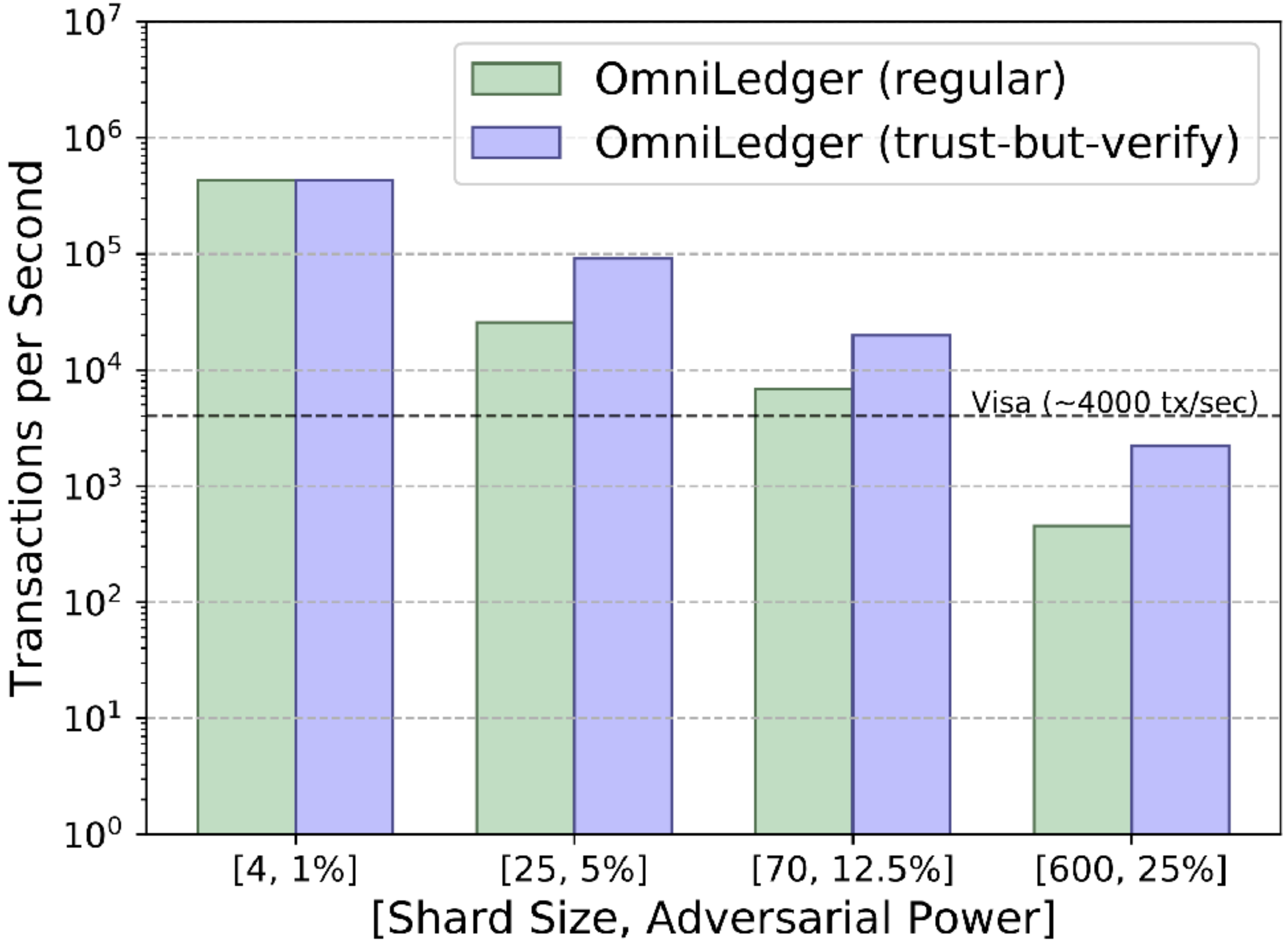
Implementation

- 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
- Network restrictions
  - 20 Mbps bandwidth
  - 200 ms round-trip latency

Evaluation: Throughput



Results for 1800 validators

Evaluation: Throughput

#shards	1	2	4	8	16
tx/sec	439	869	1674	3240	5850

Scale-out throughput for 12.5%-adversary
and shard size 70 and 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%	
regular validation	1.38	5.99	8.04	14.52	1 MB blocks
1st lvl. validation	1.38	1.38	1.38	4.48	500 KB blocks
2nd lvl. validation	1.38	55.89	41.89	62.96	16 MB blocks

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**
  - RandHound: Secure shard-validator assignment via publicly-verifiable unbiasable randomness
  - Atomix: Client-managed cross-shard tx
  - ByzCoinX: Robust intra-shard BFT consensus
  - Sharding: Visa-level throughput and beyond
  - Trust-but-verify validation: No latency vs. throughput tradeoff
  - For PoW, PoS, permissioned, etc.
- **Paper:** ia.cr/2017/406 (to be published at IEEE S&P'18)
- **Code:** https://github.com/dedis