OmniLedger: A Secure, Scale-Out, Decentralized Ledger

Eleftherios Kokoris Kogias†, **Philipp Jovanovic†**, Linus Gasser†, Nicolas Gailly†, Ewa Syta*, Bryan Ford†

†EPFL, Switzerland
*Trinity College, USA

Workshop on Blockchain Technology and Theory
2017-10-16, Vienna

Talk Outline

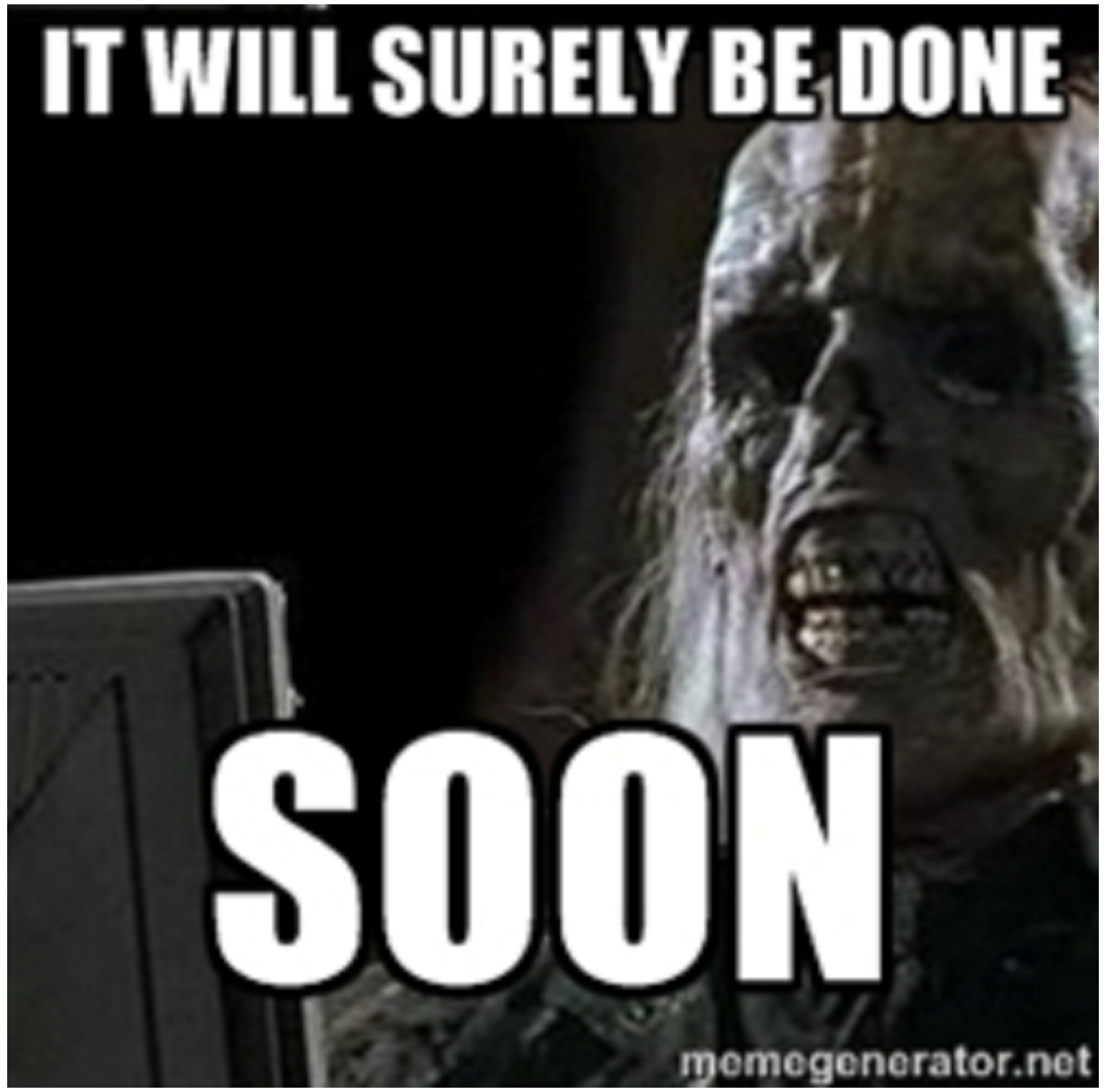
- Motivation
- OmniLedger
- Evaluation
- Conclusion

Talk Outline

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

Drawbacks of Nakamoto Consensus

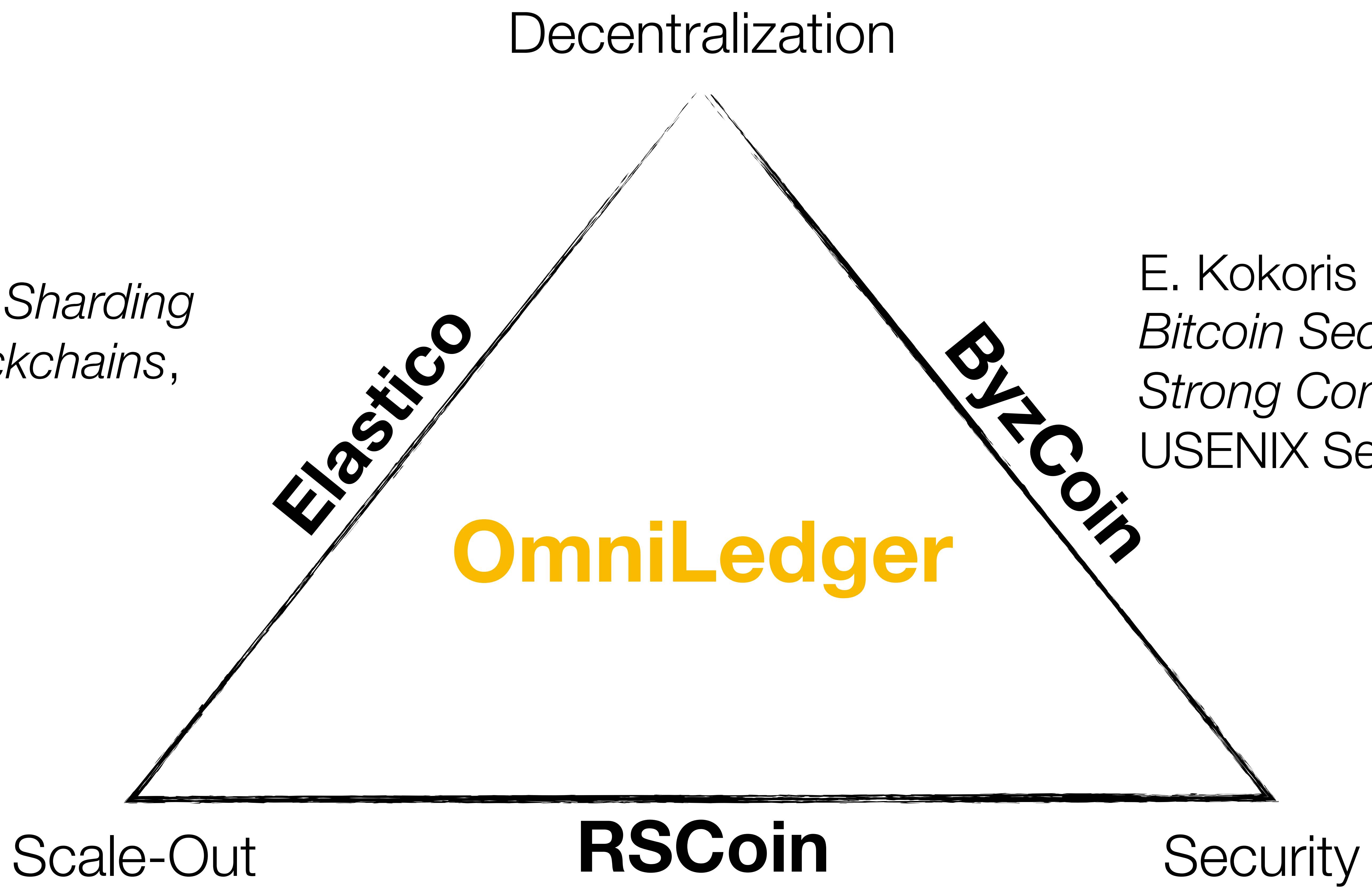
- Transaction confirmation delay
  - Any transaction takes *at least* 10 mins until being confirmed
- Weak consistency
  - You are not really certain your transaction is committed until you wait 1 hour or more
- Low throughput
  - Bitcoin: ~7 tx/sec
- Proof-of-work mining
  - Wastes huge amount of energy



Scaling Bitcoin 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

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

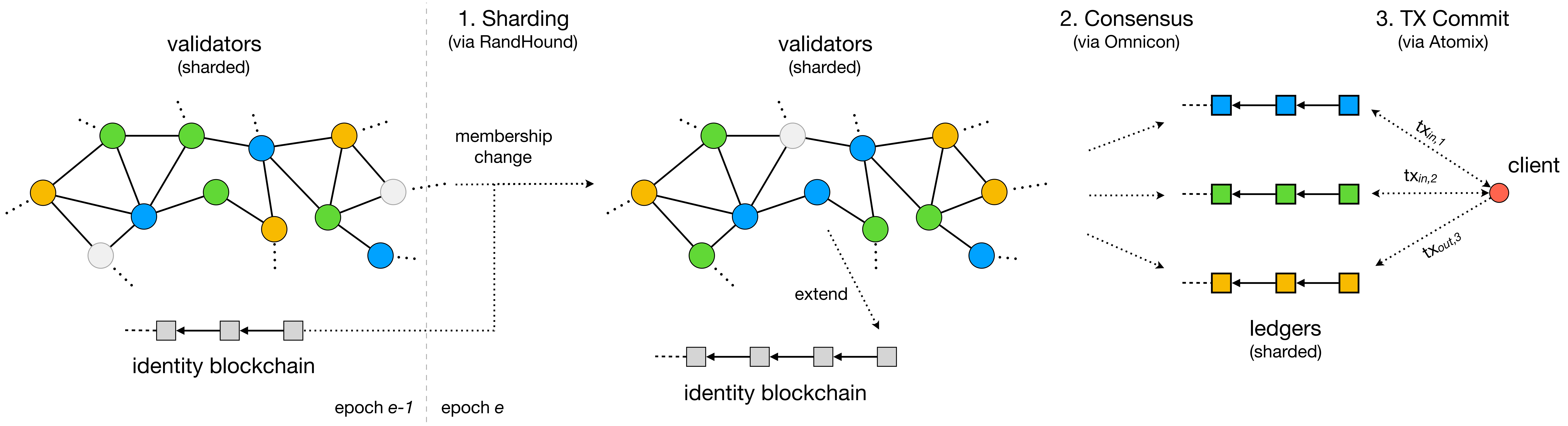
4. Scale-out
Throughput increases linearly in the number of active validators

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

6. Low Latency
TX are confirmed quickly

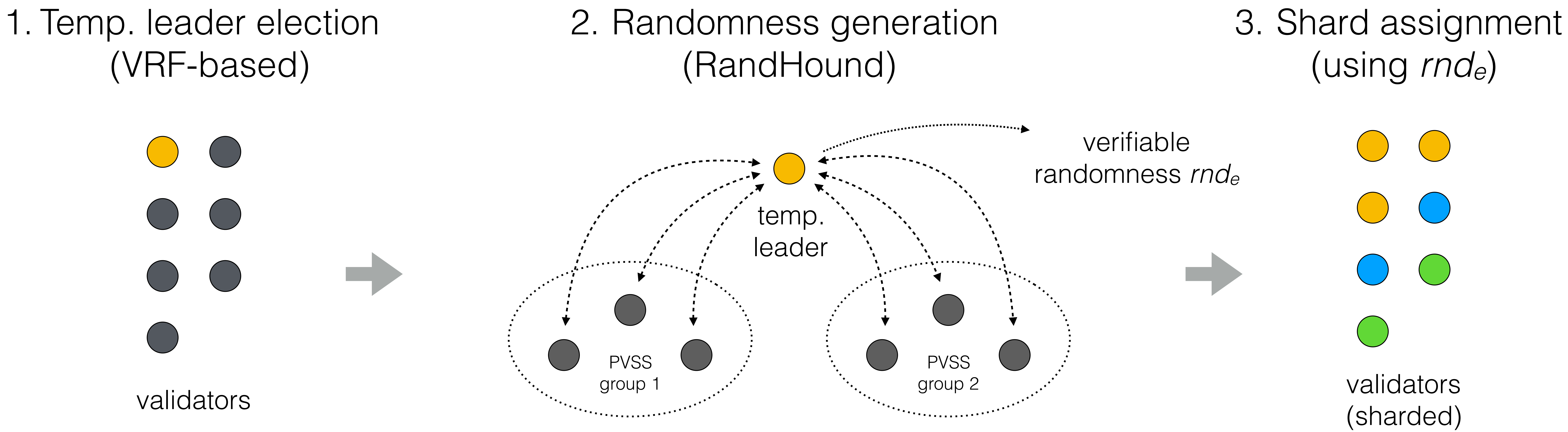
Assumptions: <= 25% mildly adaptive Byzantine adversary, (partially) synchronous network, UTXO model

OmniLedger – Overview



1. **Sharding:** Periodically re-assign validators to shards in a randomized manner (using RandHound)
2. **Consensus:** Validators ensure consistency of shard states (using Omnicon)
3. **TX Commit:** Clients ensure consistency of cross-shard transactions (using Atomix)

Sharding



Goal:

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

Solution:

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

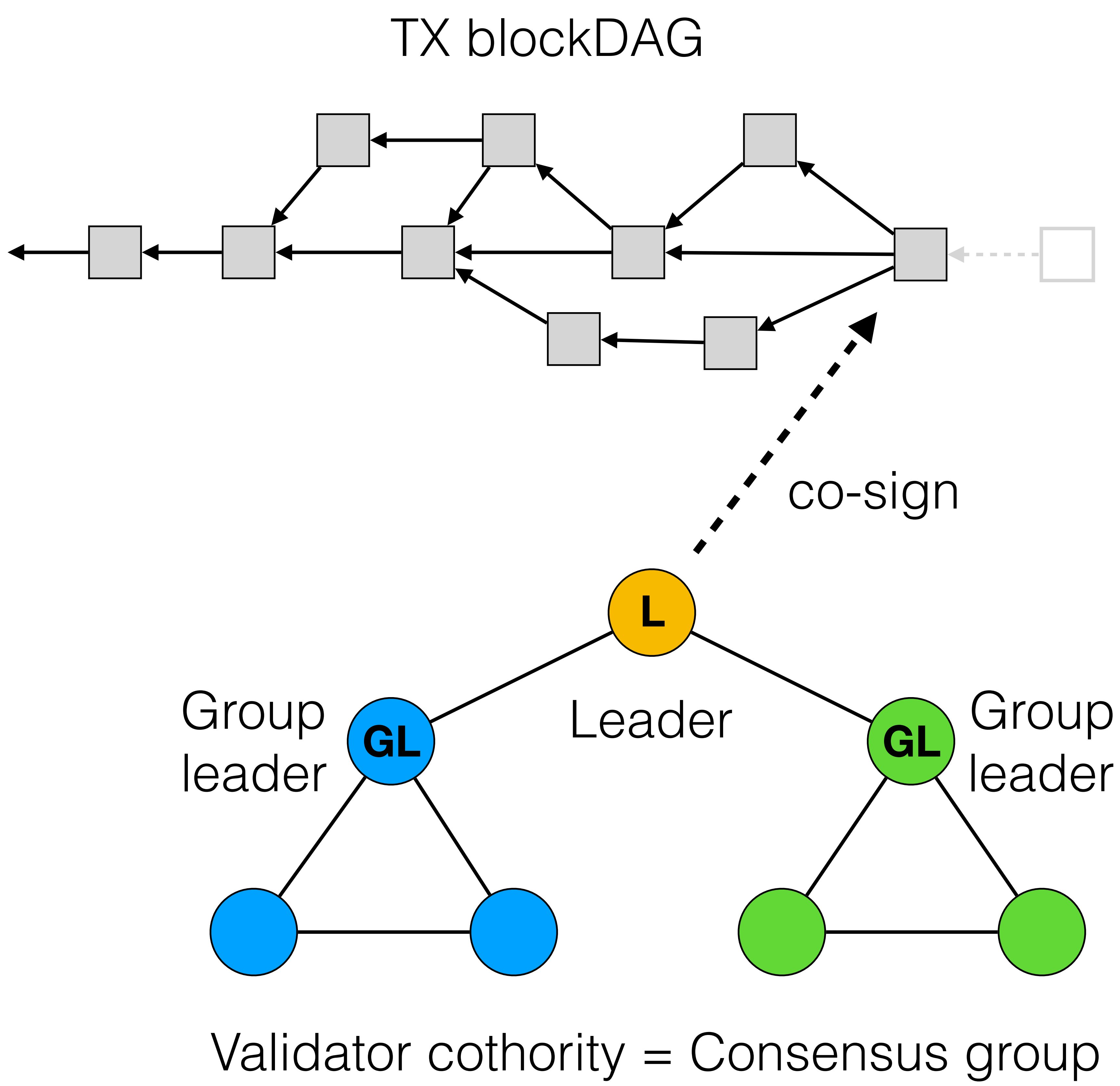
Consensus

Goal:

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

Solution: Omnicon

- 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



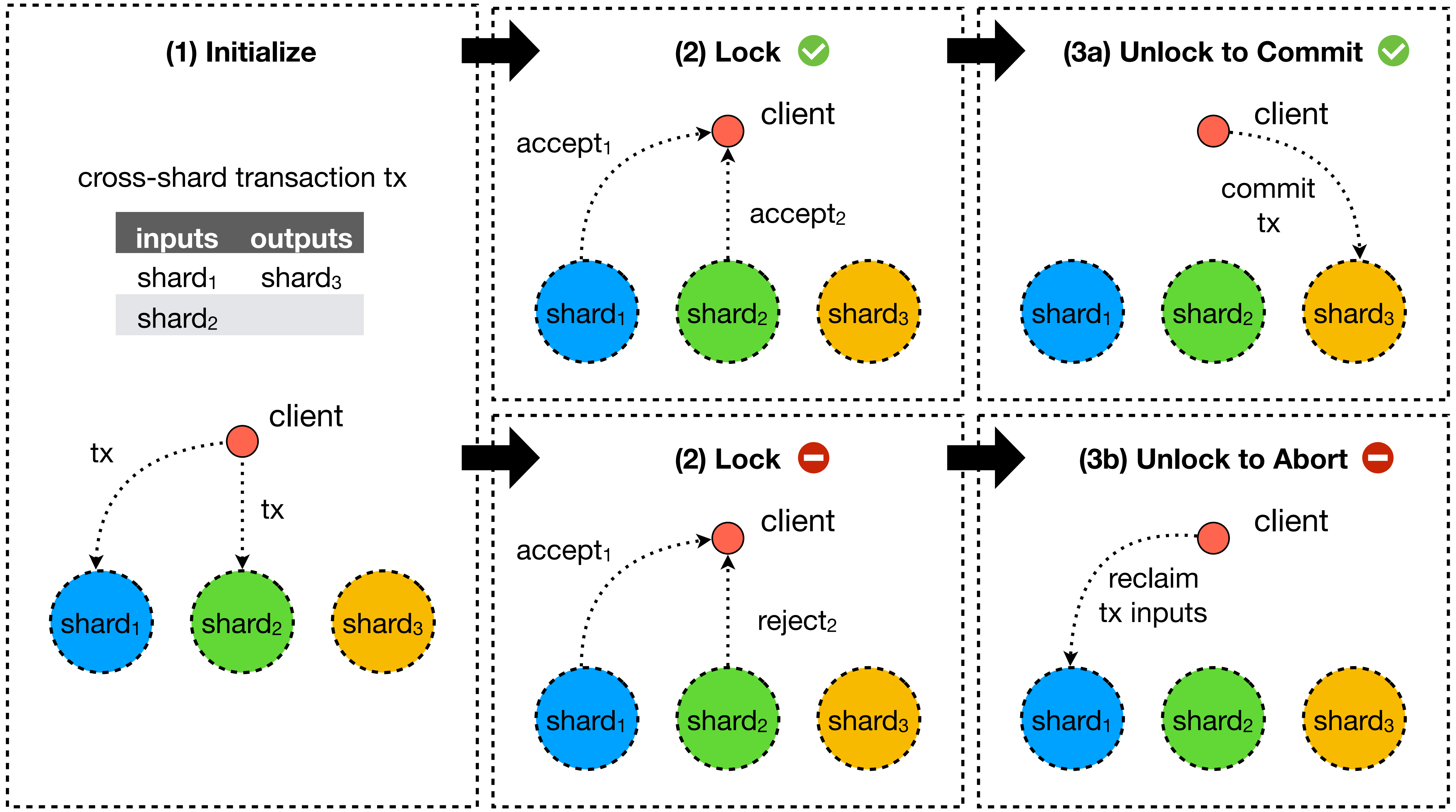
Transaction Commit

Goal:

- Cross-shard TX commit atomically or abort eventually

Solution: Atomix

- Client-managed protocol
  1. Client sends cross-shard TX to input shards
  2. Collect acceptance/rejection proofs from input shards
  3. (a) If all input shards accepted, commit to output shard, otherwise (b) abort and reclaim input funds
- Optimistically trust client for liveness
  - Anyone can take over the clients job if he times out
- Collective signing (CoSi) ensures compact proofs



The **Atomix** protocol for secure cross-shard transactions

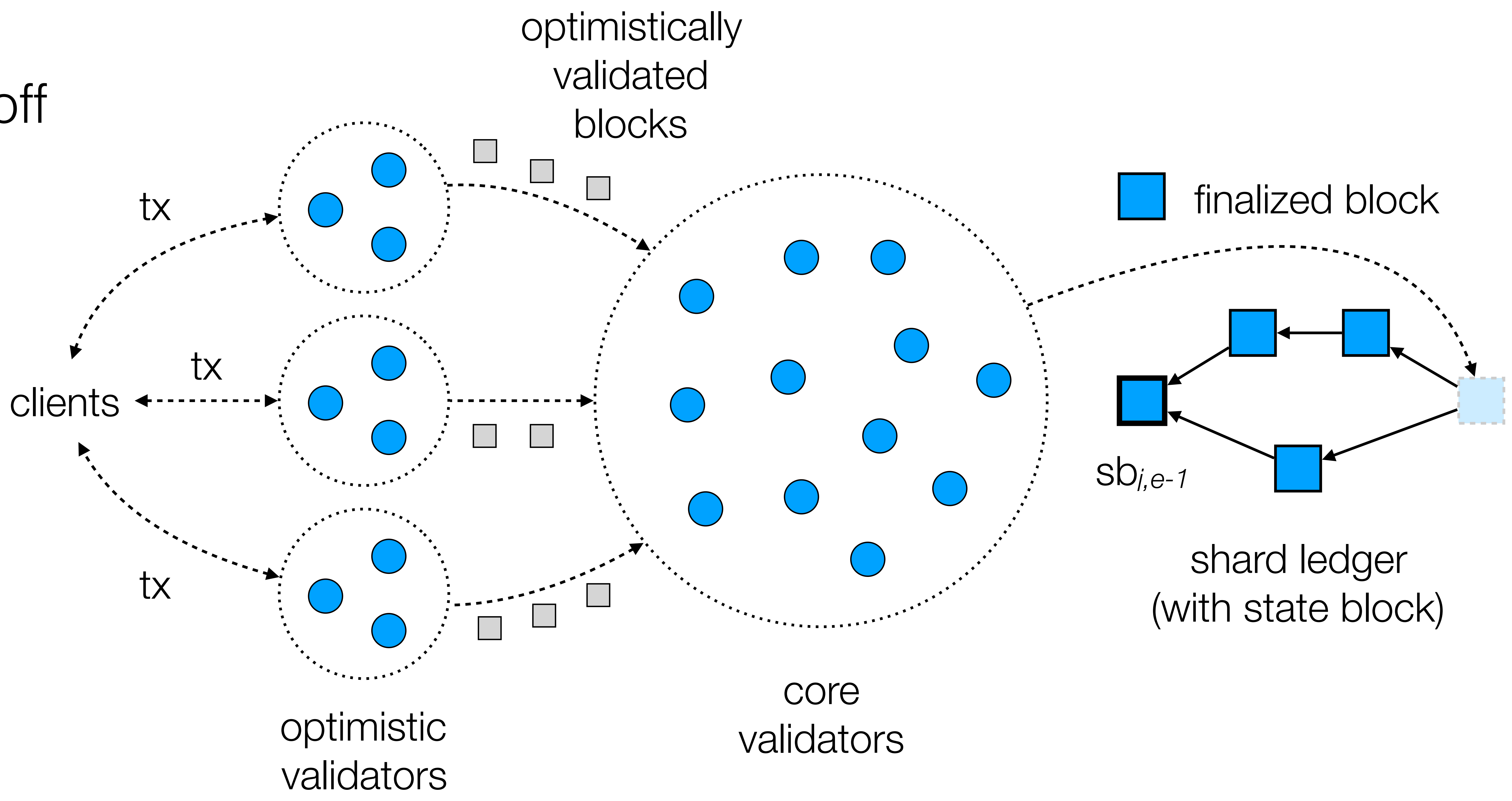
Trust-but-Verify Transaction Validation

Goal:

- Avoid latency vs. throughput trade-off

Solution:

- Use 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

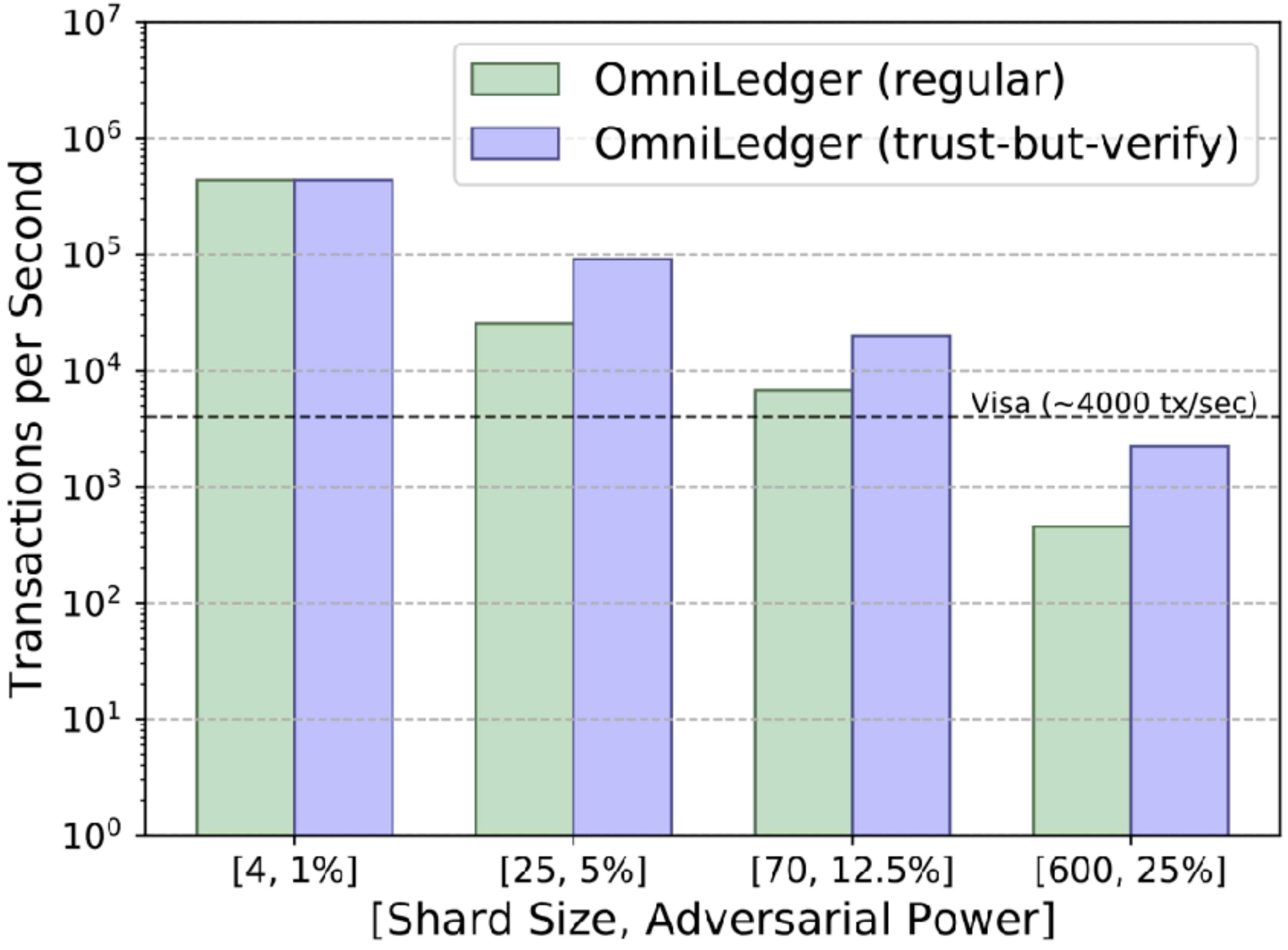
Implementation

- Go versions of OmniLedger and its subprotocols (Omnicon, Atomix, etc.)
- Based on DEDIS code
  - Kyber crypto library
  - 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



Scale-out throughput for 12.5%-adversary and shard size 70

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

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%	
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
  - Sharding through publicly-verifiable unbiasable randomness (via RandHound)
  - Intra-shard BFT consensus (via Omnicon)
  - Client-managed cross-shard TX (via Atomix)
  - Avoids latency vs. throughput tradeoff (via trust-but-verify TX validation)
  - Visa-level throughput and beyond
- Full paper: ia.cr/2017/406

Thank you!

Questions?

Philipp Jovanovic – philipp.jovanovic@epfl.ch – @Daeinar