BlockRaFT: A Distributed Framework for Fault-Tolerant and Scalable Blockchain Nodes

Manaswini Piduguralla Souvik Sarkar Arunmoezhi Ramachandran
Sathya Peri

ICSA 2026

24^{th} June 2026

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Table of Contents

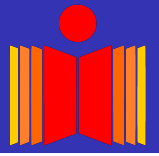
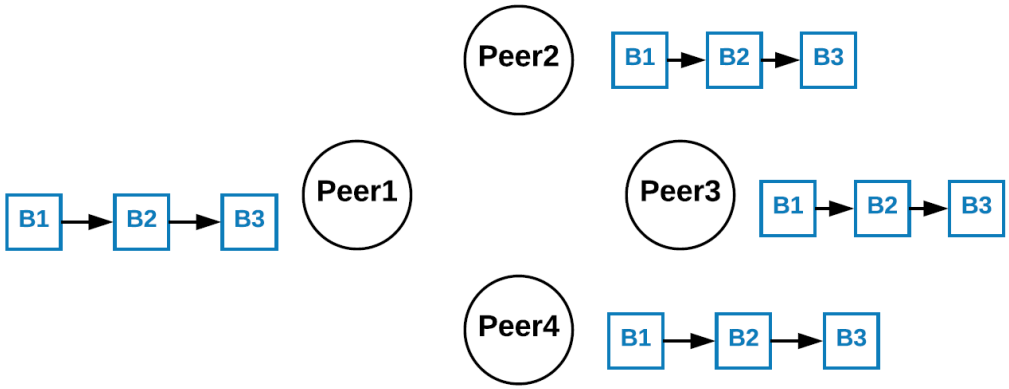


Table of Contents

Introduction to Blockchain

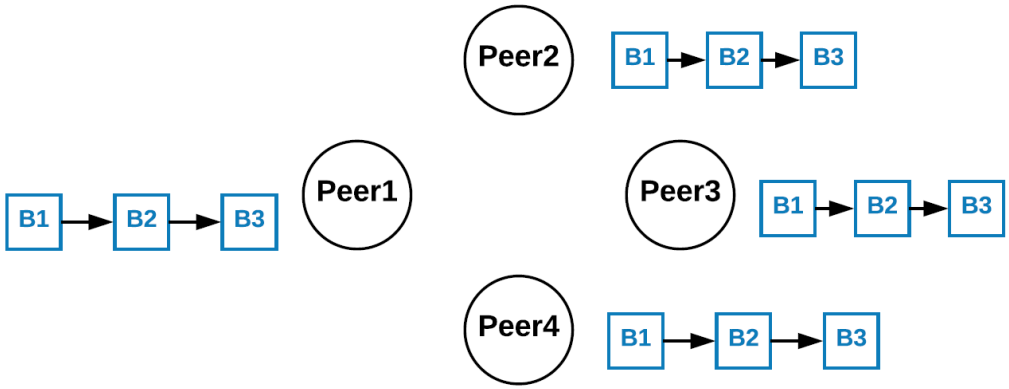
- Blockchain is a decentralized distributed immutable ledger shared among untrusted parties[1].



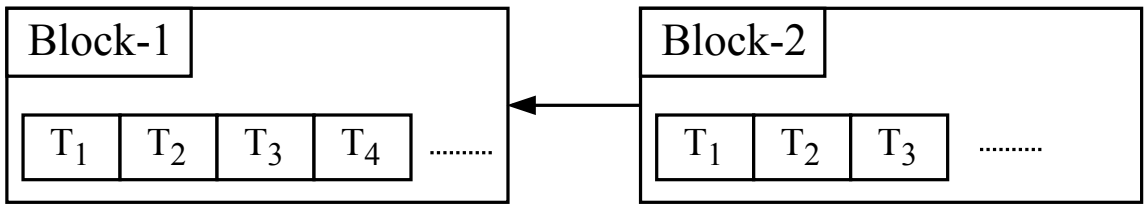
[1] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. https://bitcoin.org/bitcoin.pdf. 2008.

Introduction to Blockchain

- Blockchain is a decentralized distributed immutable ledger shared among untrusted parties[1].



- Each block contains a set of transactions.



[1] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. https://bitcoin.org/bitcoin.pdf. 2008.

Introduction: Blockchain High Level Design

- Block Producer is responsible for creating new blocks by grouping transaction.

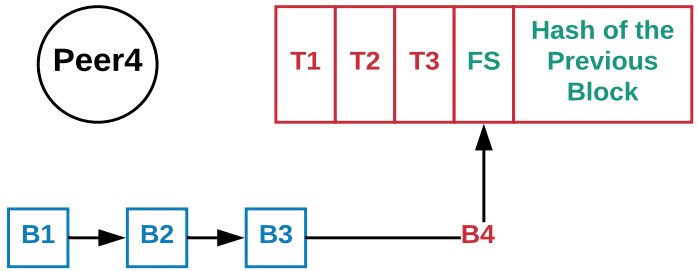


Figure: Block producer forms a block B4 and computes final state (FS) sequentially

Introduction: Blockchain High Level Design

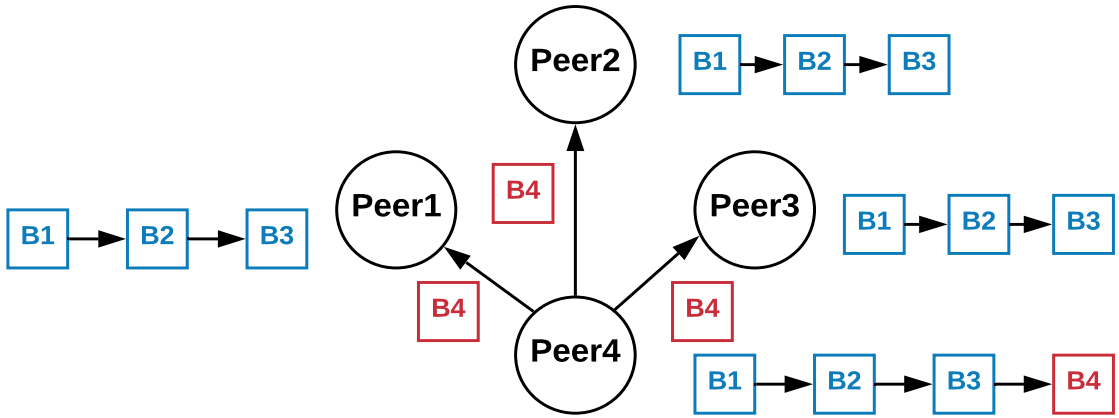


Figure: Block producer broadcasts the block B4

Introduction: Blockchain High Level Design

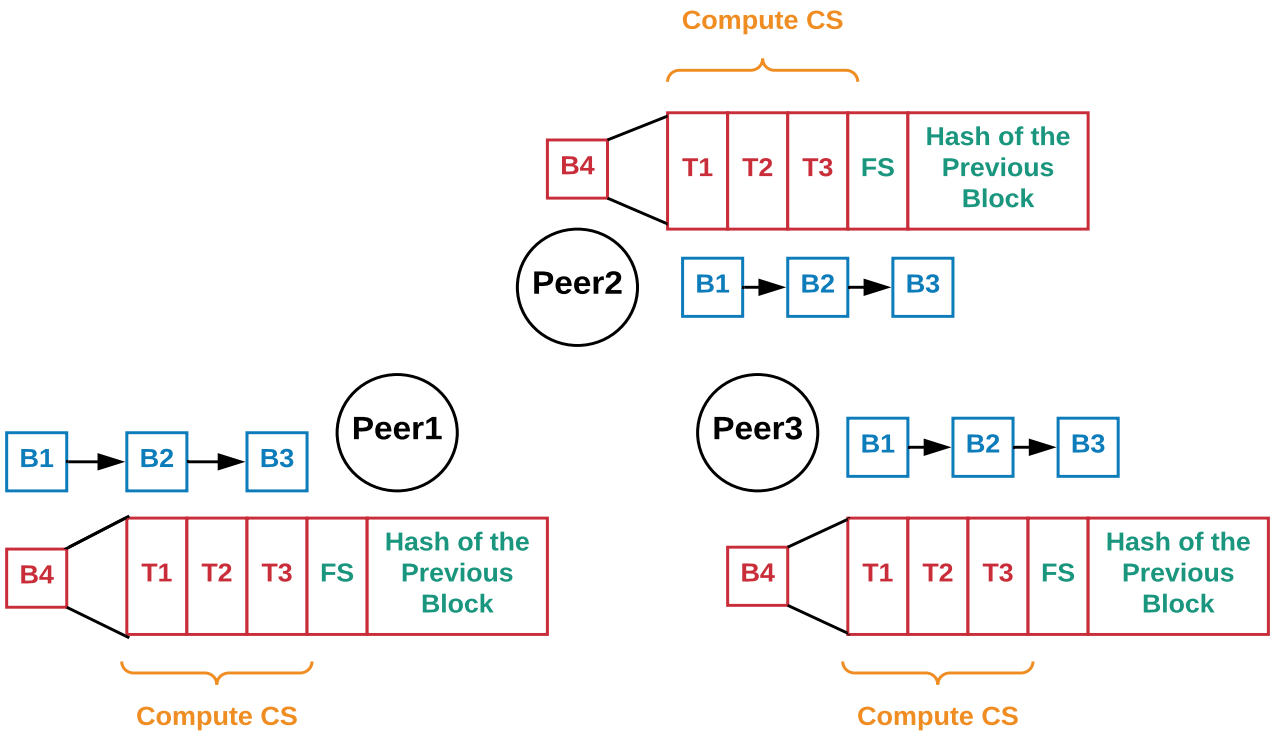


Figure: Validators (Peer 1, 2, and 3) compute current state (CS) sequentially

Introduction: Blockchain High Level Design

- Peers acting as Block Validators are responsible for verifying transactions.

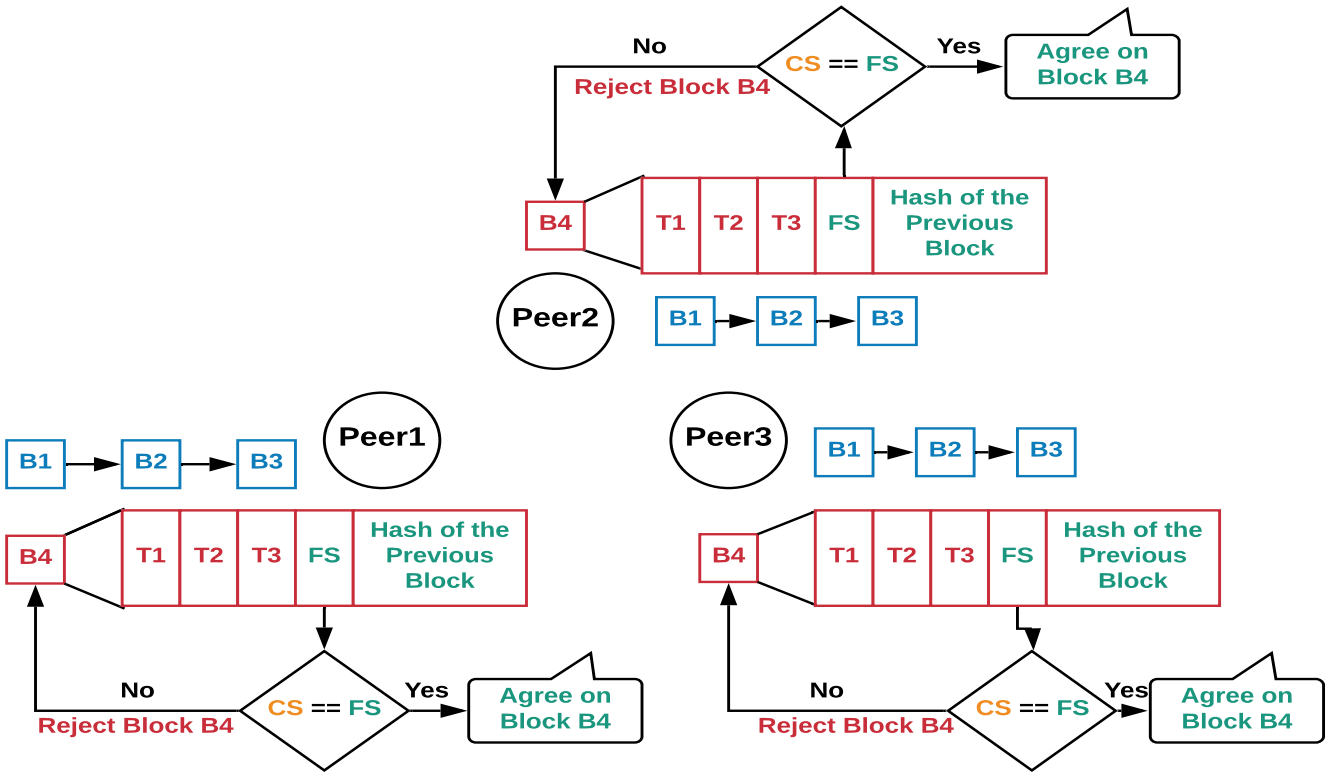


Figure: Block Validators verify the FS and add the block

Introduction: Blockchain High Level Design

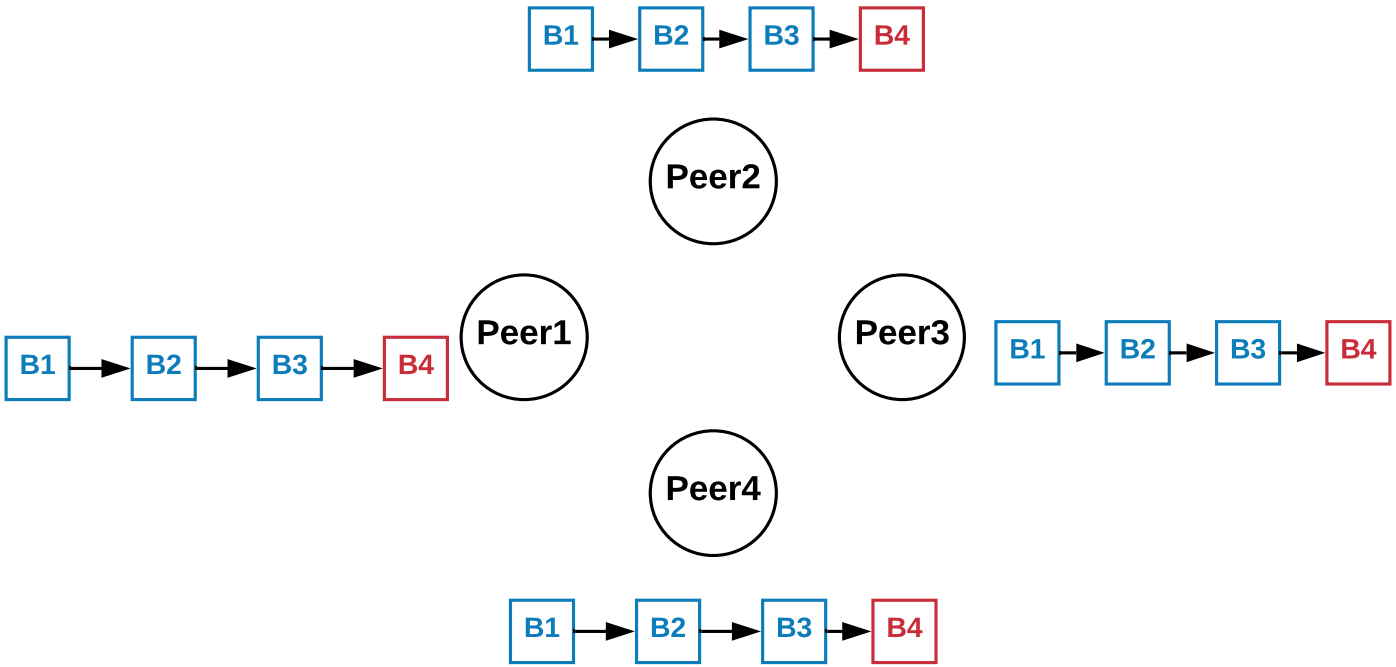


Figure: Block B4 successfully added to the blockchain

Table of Contents

Motivation: Fault Tolerance

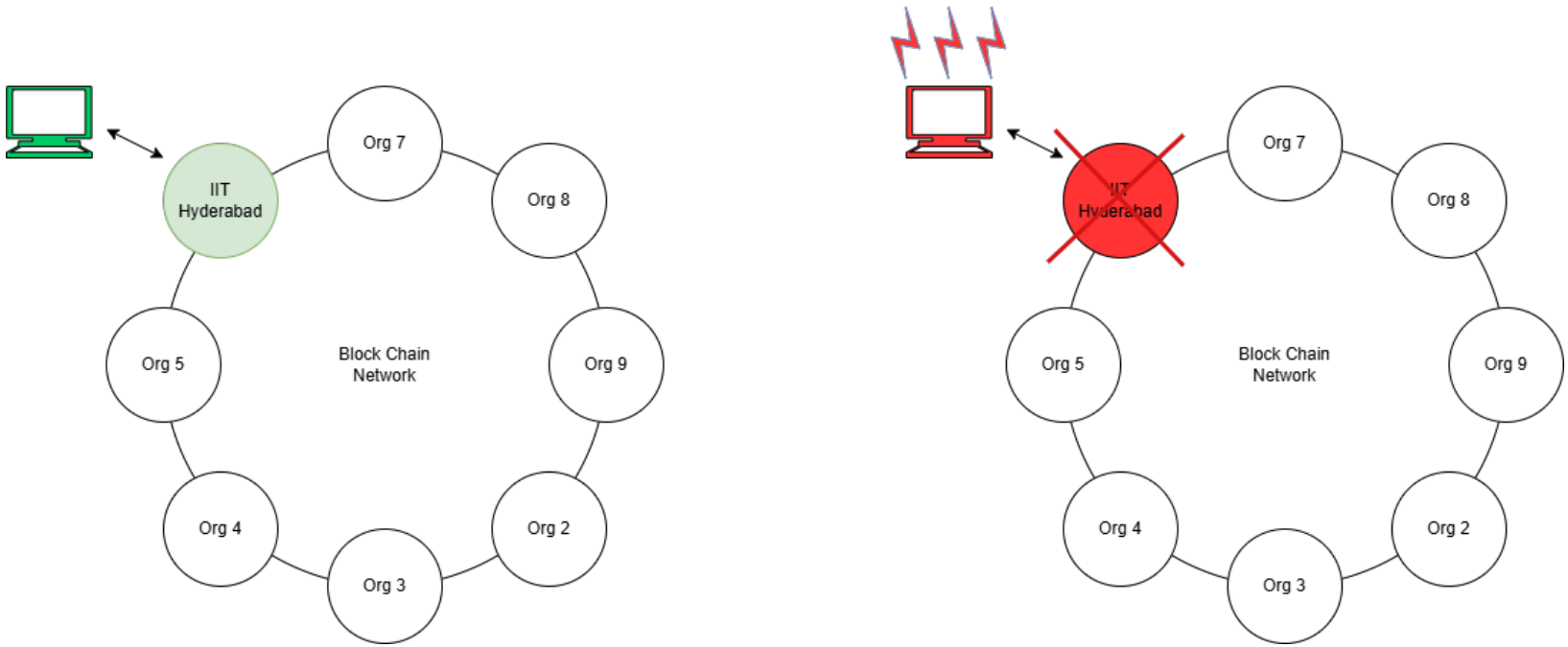


Figure: Active node vs Crashed node

- Blockchain networks are inherently **crash tolerant**.
- However, if an organization's blockchain node crashes, that organization loses access to the blockchain services.

Motivation

Permissioned blockchains face two major challenges:

- **Scalability**
  - Low throughput
  - Increasing smart-contract complexity
- **Availability and Fault Tolerance**
  - Each organization typically runs only a few nodes
  - Node crashes deny service to that organization

Solutions : Approaches

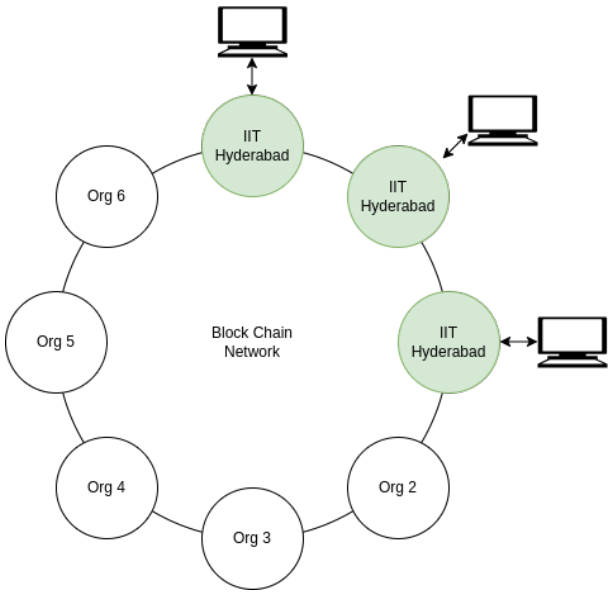


Figure: Replicated Node

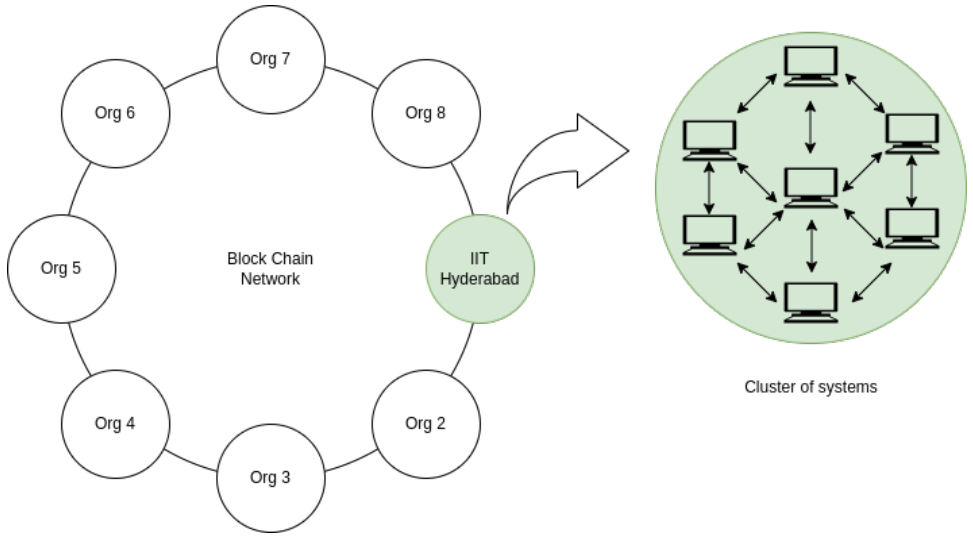


Figure: **Our Approach:** Distributed Node

Table of Contents

Related Work on Distributed Architecture

There are two frameworks on Distributed Architecture:

[2] Shrey Baheti et al. "DiPETrans: A framework for Distributed Parallel Execution of Transactions of Blocks in Blockchains". In: *Concurrency and Computation: Practice and Experience* 34.10 (2022), e6804.

[3] Quentin Kniep et al. "Pilotfish: Distributed Execution for Scalable Blockchains". In: *Financial Cryptography and Data Security (FC)*. Miyakojima, Japan, Apr. 2025.

Related Work on Distributed Architecture

There are two frameworks on Distributed Architecture:

- DiPETrans[2]: Distributed framework mainly for performance.

[2] Shrey Baheti et al. "DiPETrans: A framework for Distributed Parallel Execution of Transactions of Blocks in Blockchains". In: *Concurrency and Computation: Practice and Experience* 34.10 (2022), e6804.

[3] Quentin Kniep et al. "Pilotfish: Distributed Execution for Scalable Blockchains". In: *Financial Cryptography and Data Security (FC)*. Miyakojima, Japan, Apr. 2025.

Related Work on Distributed Architecture

There are two frameworks on Distributed Architecture:

- DiPETrans[2]: Distributed framework mainly for performance.
- PilotFish[3]: Distributed architecture with master & slave approach. It can tolerate failure of slave nodes but not the master.

[2] Shrey Baheti et al. "DiPETrans: A framework for Distributed Parallel Execution of Transactions of Blocks in Blockchains". In: *Concurrency and Computation: Practice and Experience* 34.10 (2022), e6804.

[3] Quentin Kniep et al. "Pilotfish: Distributed Execution for Scalable Blockchains". In: *Financial Cryptography and Data Security (FC)*. Miyakojima, Japan, Apr. 2025.

Our Solution: Distributed Architecture

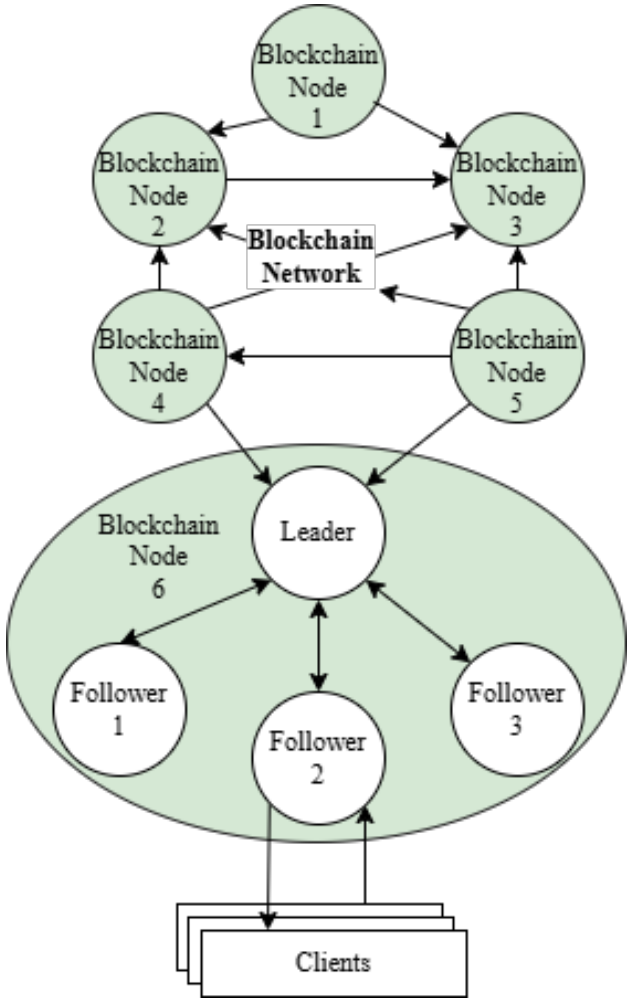


Figure: Distributed Framework Design for the Blockchain Node.

Distributed Framework

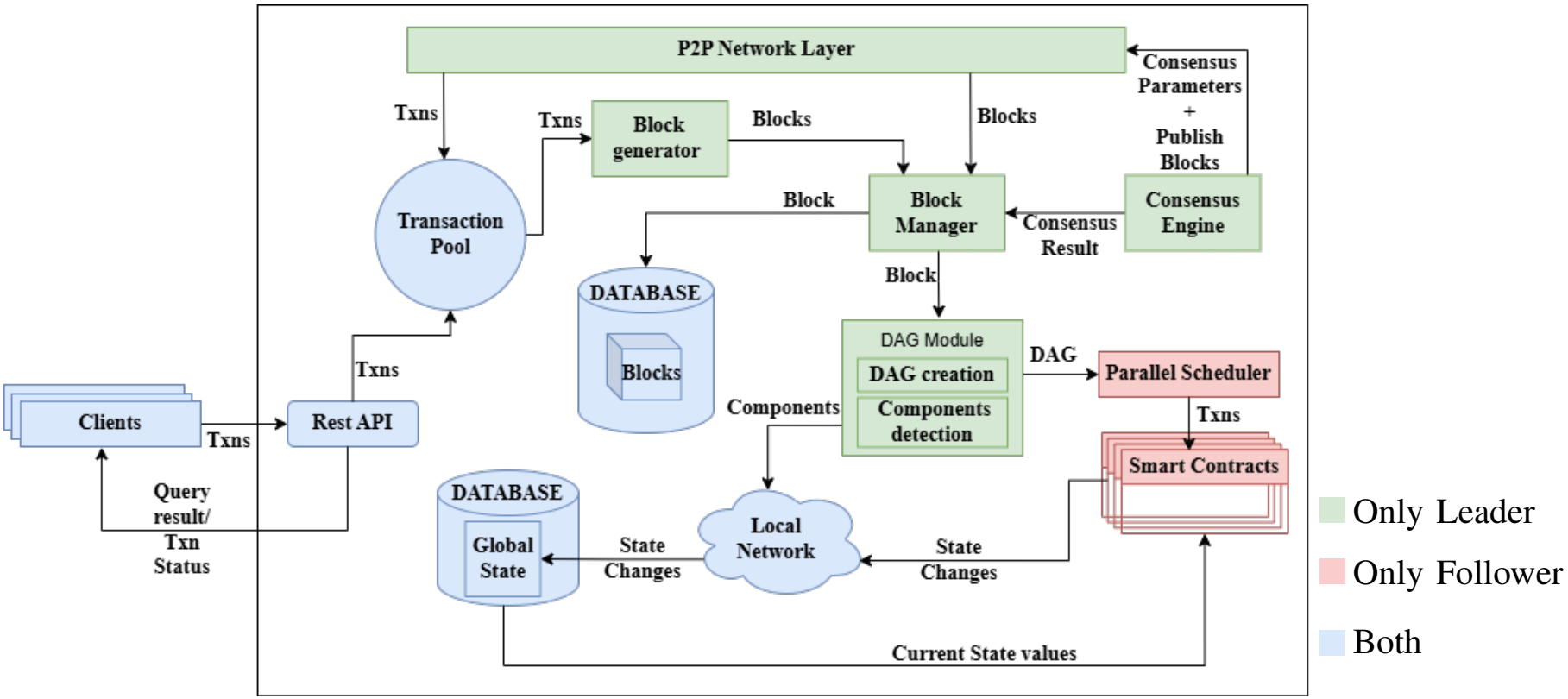


Figure: Node Architecture in the Proposed Framework

Distributed Framework

- Only the leader interacts with the blockchain network, allowing the entire BlockRaFT cluster to appear as a single blockchain node.
- All nodes exchange periodic heartbeat messages to detect failures.

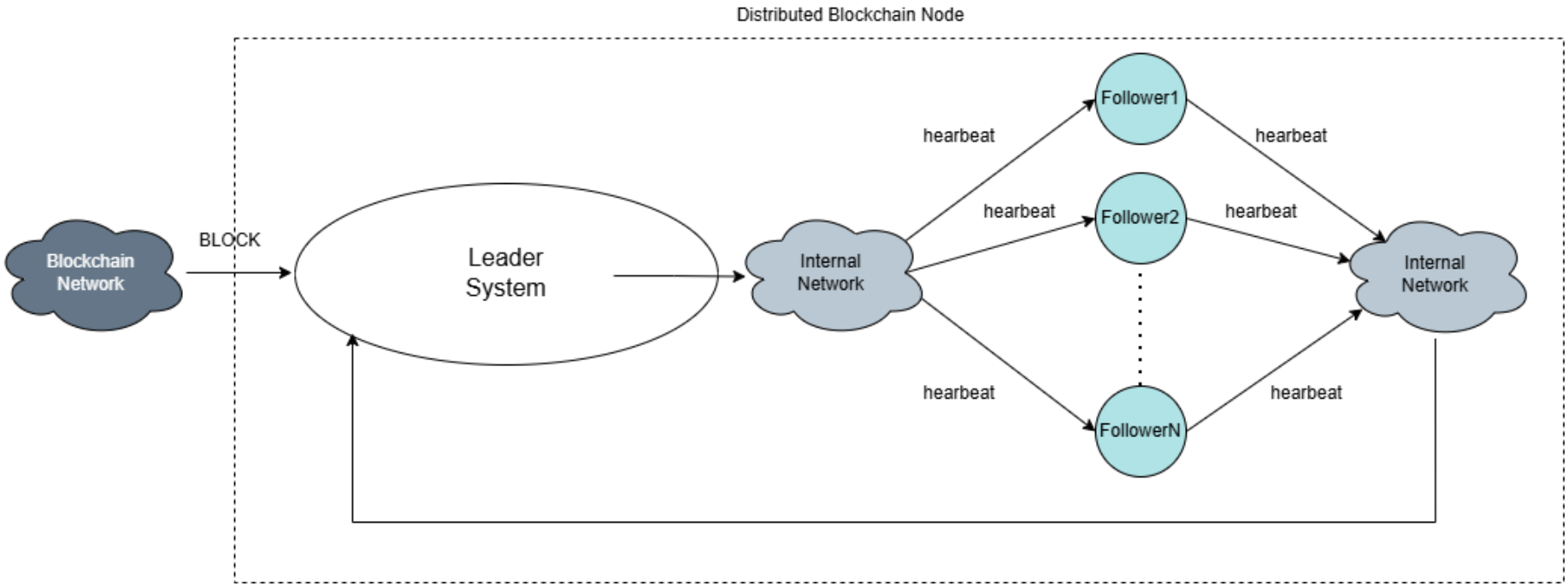


Figure: Simplified Leader-Follower Execution Flow

Distributed Framework

- A workload balancer at the leader partitions and distributes execution tasks across active followers.
- ETCD[4] provides a distributed shared-memory abstraction for cluster-wide coordination and task communication.

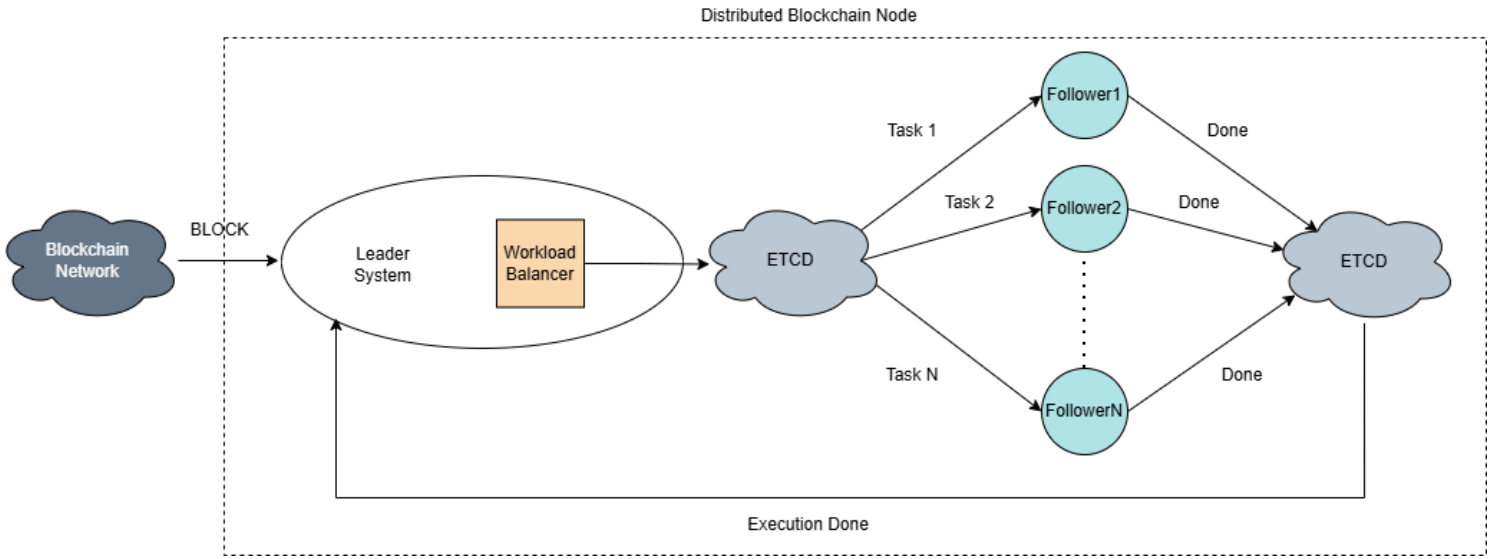


Figure: Simplified Leader-Follower Execution Flow

[4] Open Source Community. *etcd: A distributed, reliable key-value store for critical data*. GitHub repository. Accessed: 2025-02-19. URL: https://github.com/etcd-io/etcd

Block Validator Work Flow

- The leader receives a block for validation from the blockchain network
- A dependency DAG is constructed to capture conflicts among transactions.
- Independent DAG components are assigned to followers.

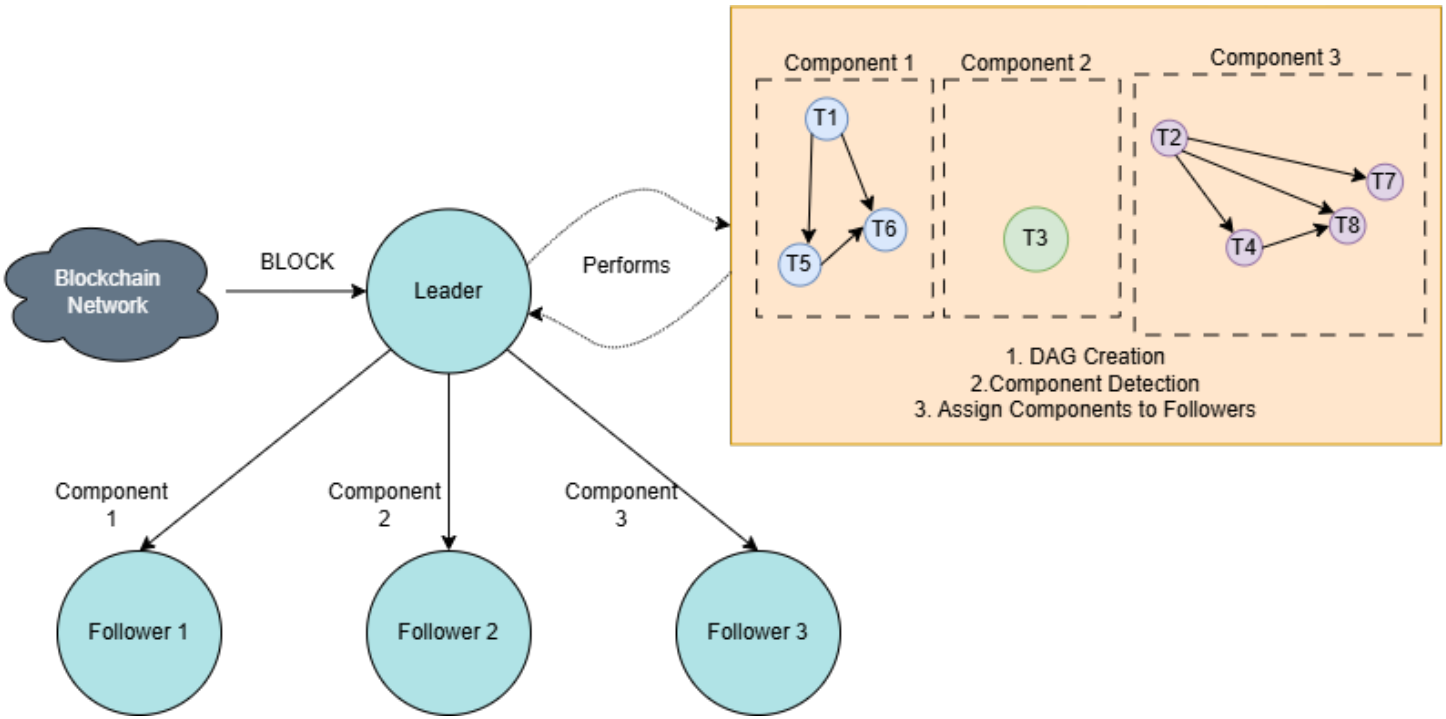


Figure: Block Validation Work Flow

Table of Contents

Crash Scenarios: Leader Crash

- Leader failure is detected through missing heartbeats.
- Followers initiate a new RAFT protocol for leader election.
- We use on ETCD's RAFT[5] leader election to elect a new leader.

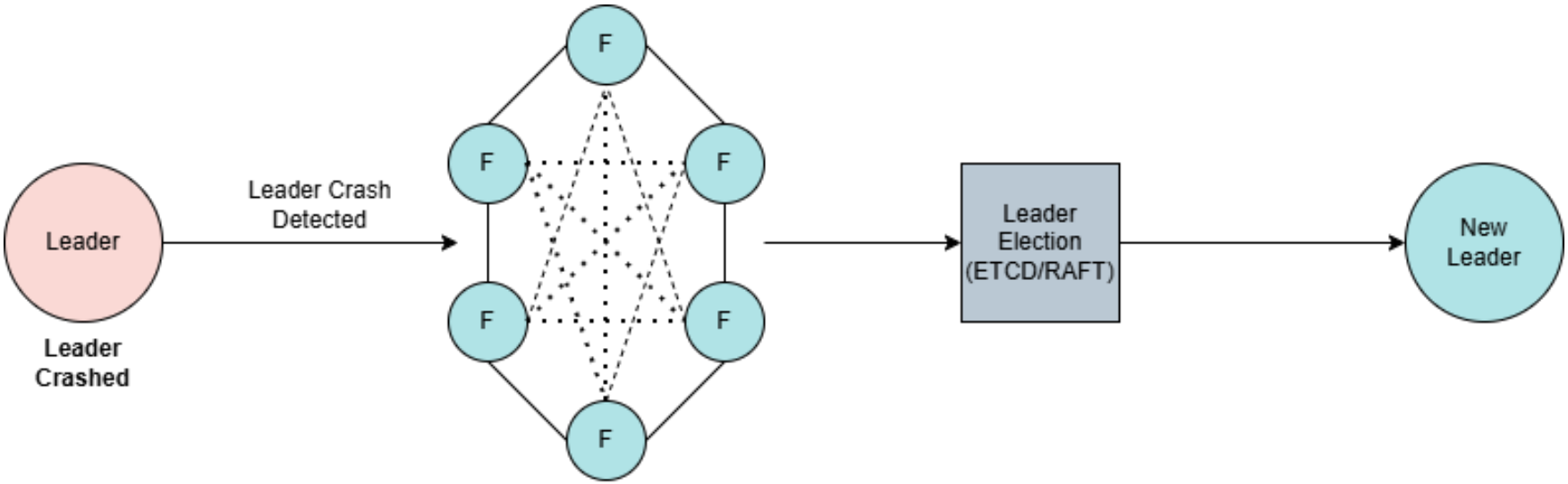


Figure: Procedure for Leader Crash Handling

[5] Diego Ongaro and John Ousterhout. "In search of an understandable consensus algorithm". In: *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*. USA, 2014, pp. 305–320.

Crash Scenarios: Follower Crash

- The leader detects follower failures using heartbeat monitoring.
- Work assigned to the failed follower is redistributed to active followers.
- Execution continues as long as the cluster maintains quorum ($n/3$ as required by RAFT).

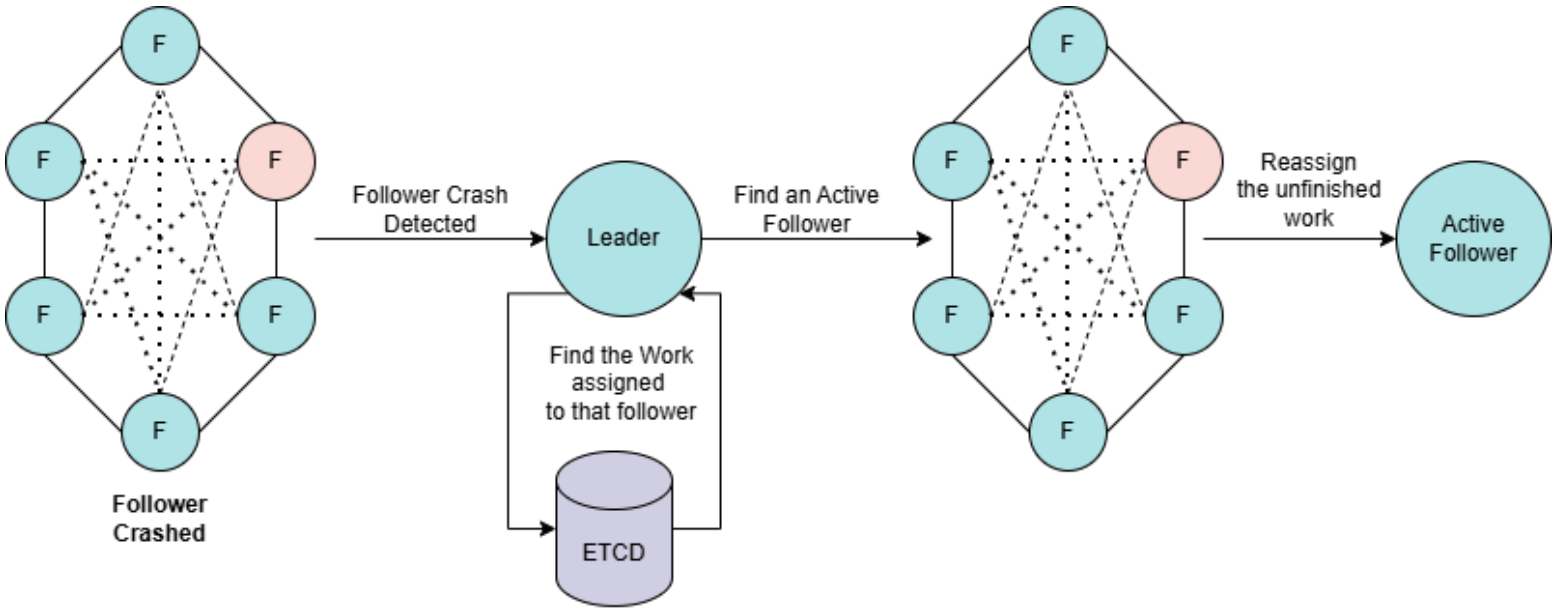


Figure: Procedure for Follower Crash Handling

Table of Contents

Experimental Setup

Operating System: Ubuntu (64-bit)
Processor:

- Model: AMD EPYC 7452 32-Core Processor
- Architecture: x86_64
- CPU Count: 128 logical CPUs (2 sockets $\times$ 32 cores $\times$ 2 threads per core)
- Memory: 251 GB
- Buffer/Cache: 7.2 GB
- Base Frequency: 1.5 GHz

Individual node configuration
Operating System: Ubuntu (64-bit)
Processor:

- CPU Count: 16 logical CPUs (1 sockets $\times$ 8 cores $\times$ 2 threads per core)
- Memory: 9.7 GB
- Buffer/Cache: 3.4 GB

Results: Varying Transactions per Block

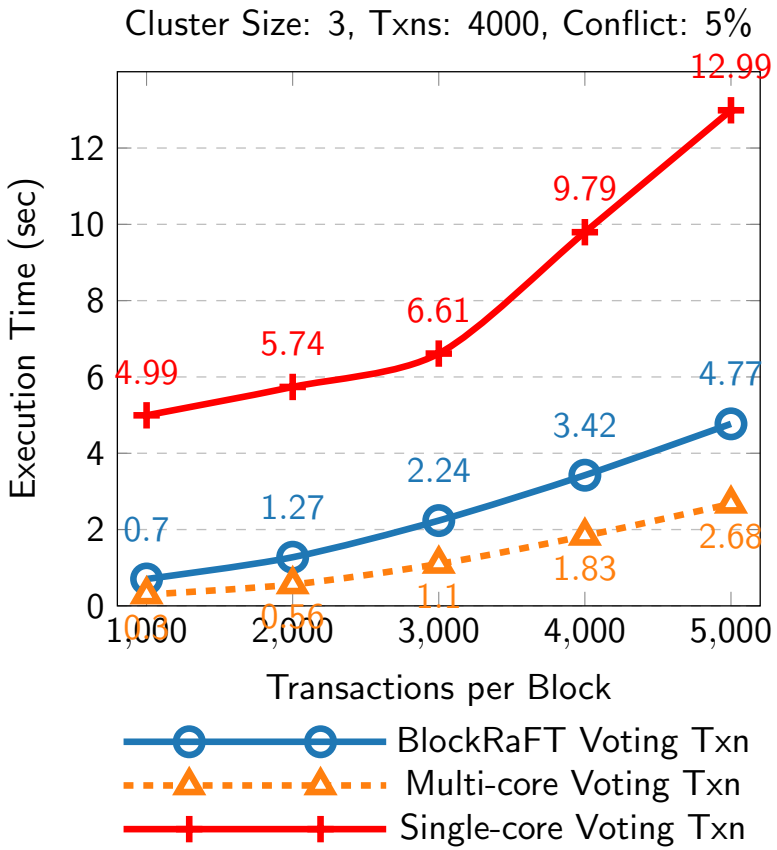


Figure: Performance Comparison across Transactions per Block

Observation:

- Execution time grows linearly with transaction volume.
- Multi-core performance is best and single-core is worst.

Results: Varying Thread count

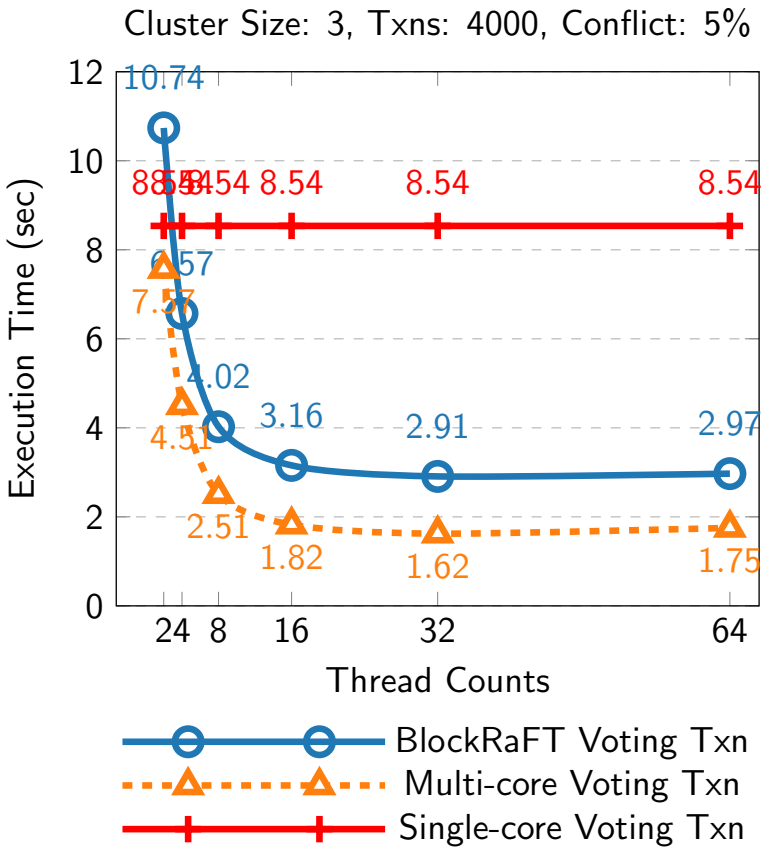


Figure: Performance Comparison across varying Thread count

Observation:

- Execution time decreases as thread count increases from 2 to 16.
- Optimal performance is achieved at 32 threads, aligning with twice the number of node cores.

Results:Varying cluster size

Observation :

- A cluster of size n, can tolerate $n/3$ crashes.
- Execution time increased with failures but system stayed functional.
- After the first crash, further crashes have minimal performance impact.

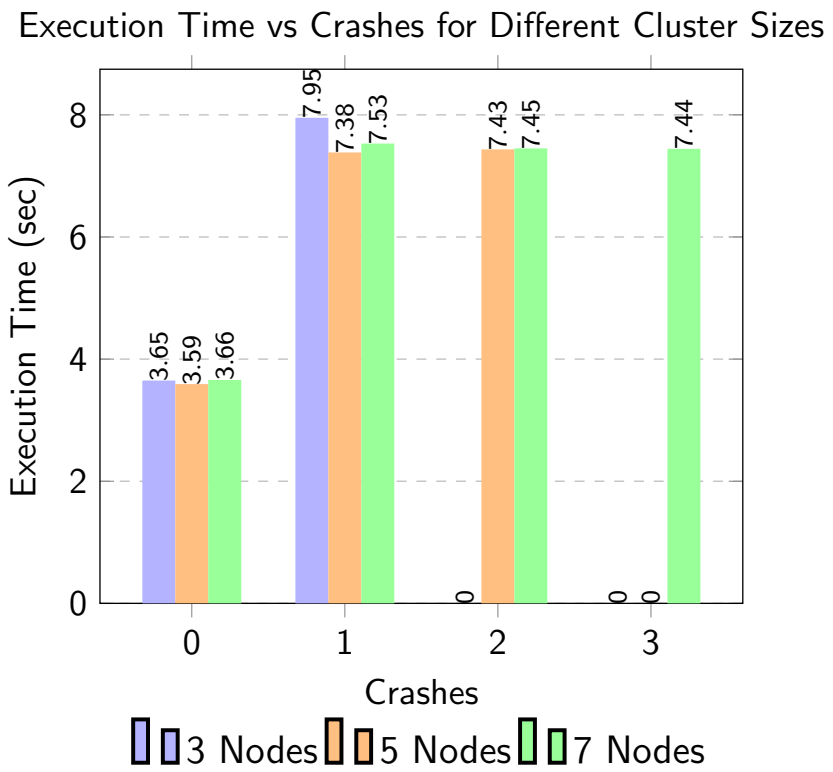
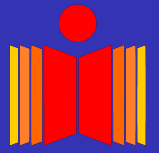


Figure: Execution Time Varying cluster size with crashes

Table of Contents



Conclusion

- **Distributed node architecture** Achieves an average **3× speedup** over traditional single-core blockchain systems.
- Incurs a **1.8× overhead** compared to multi-core systems a trade-off justified by:
  - **Crash tolerance**
  - **Fault resilience**
  - **Improved availability**
- Overall, Distributed node presents a practical and robust alternative for scalable, reliable blockchain execution.

Future Work

- **Reduce communication overhead**
  - Reduce overhead introduced by data sharing, particularly at 0% conflict percentage
- **Move toward decentralized coordination**
  - Beyond leader-centric cluster management (peer-to-peer arch)
- **Evaluate at larger scale**
  - Perform analysis with large scale deployment.

Thank you

Figure: Link to Code Repository