Dependency-Aware Execution Mechanism in Hyperledger Fabric Architecture

Sanyam Kaul, Manaswini Piduguralla, Gayathri Shreeya Patnala, and **Sathya Peri**

27^{th} International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2025)

11 October 2025

भारतीय प्रौद्योगिकी संस्थान
हैदराबाद
Indian Institute of Technology
Hyderabad

Table of Contents

Table of Contents

Research Objective

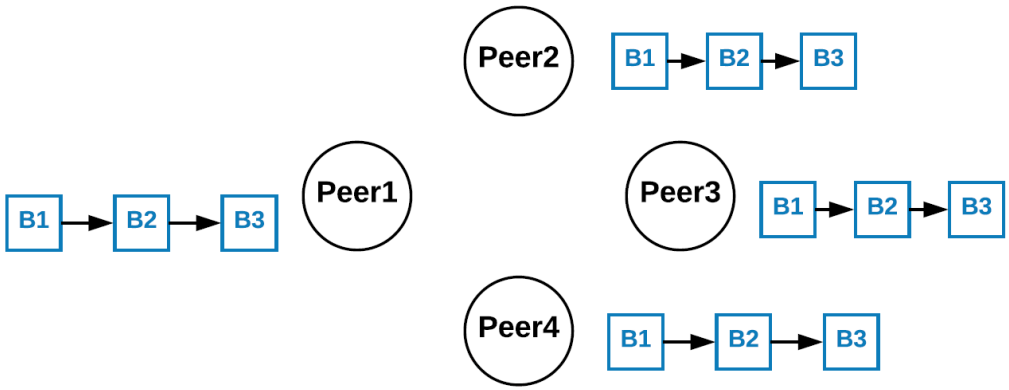
Research objective is to improve throughput, and increase commit efficiency under high concurrency in Hyperledger Fabric Blockchain.

- We Introduced a dependency-aware execution model for efficient transaction execution.

Table of Contents

Blockchain

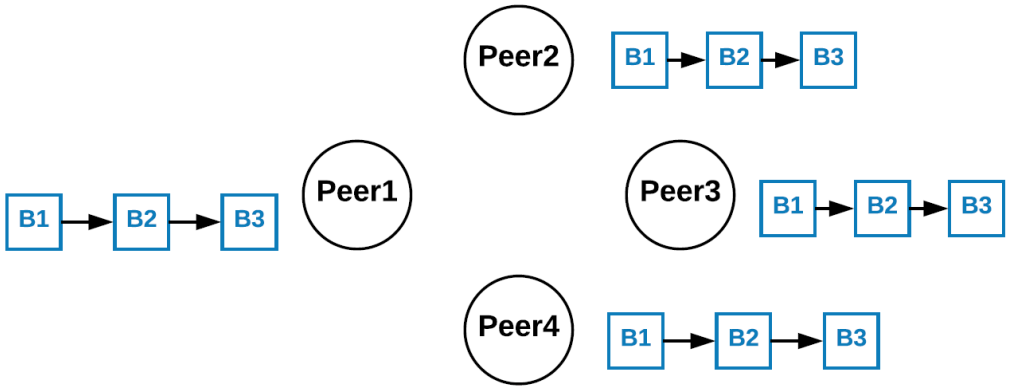
- Blockchain is a decentralized distributed immutable ledger shared among untrusted parties[1].



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

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.

Hyperledger Fabric Architecture

Hyperledger Fabric is a **permissioned blockchain platform** widely used in enterprise applications[2].

- Its **modular design**, with pluggable consensus and privacy features, enables **industrial-grade flexibility**.

[2] Elli Androulaki et al. "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains". In: *Proceedings of the Thirteenth EuroSys Conference*. EuroSys '18. ACM, 2018, 30:1–30:15. DOI: 10.1145/3190508.3190538.

Hyperledger Fabric Architecture

Hyperledger Fabric is a **permissioned blockchain platform** widely used in enterprise applications[2].

- Its **modular design**, with pluggable consensus and privacy features, enables **industrial-grade flexibility**.
- Fabric operates via a **three-phase transaction pipeline:**
  1. **Endorsement:** Clients submit proposals to endorsers, who simulate transactions and return read-write sets.
  2. **Ordering:** The ordering service collects these into blocks and broadcasts them to committing peers.
  3. **Validation:** Peers validate and apply the blocks to the ledger.

[2] Elli Androulaki et al. "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains". In: *Proceedings of the Thirteenth EuroSys Conference*. EuroSys '18. ACM, 2018, 30:1–30:15. DOI: 10.1145/3190508.3190538.

Fabric Transaction Flow

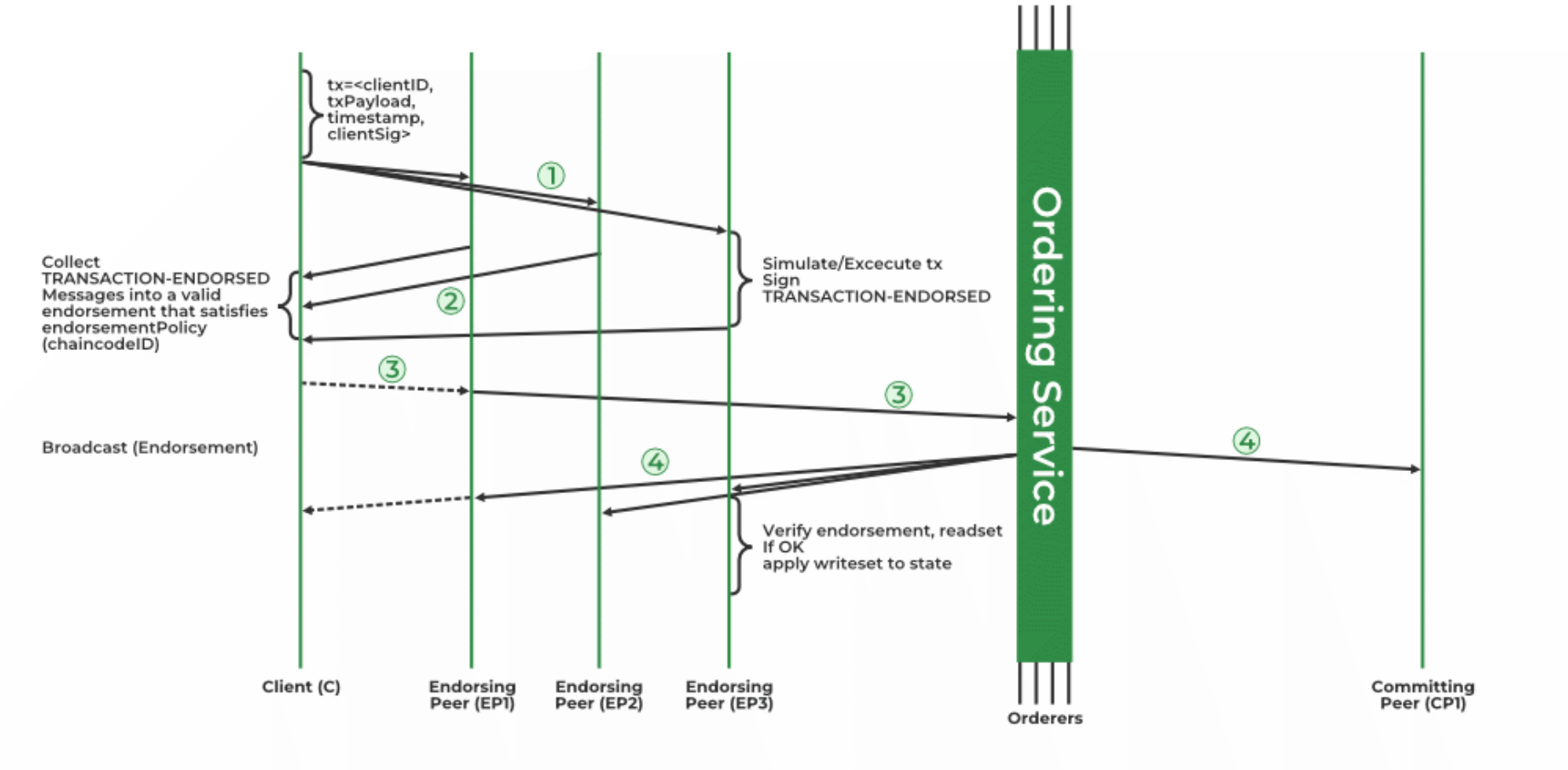


Figure: Fabric transaction flow by[3]

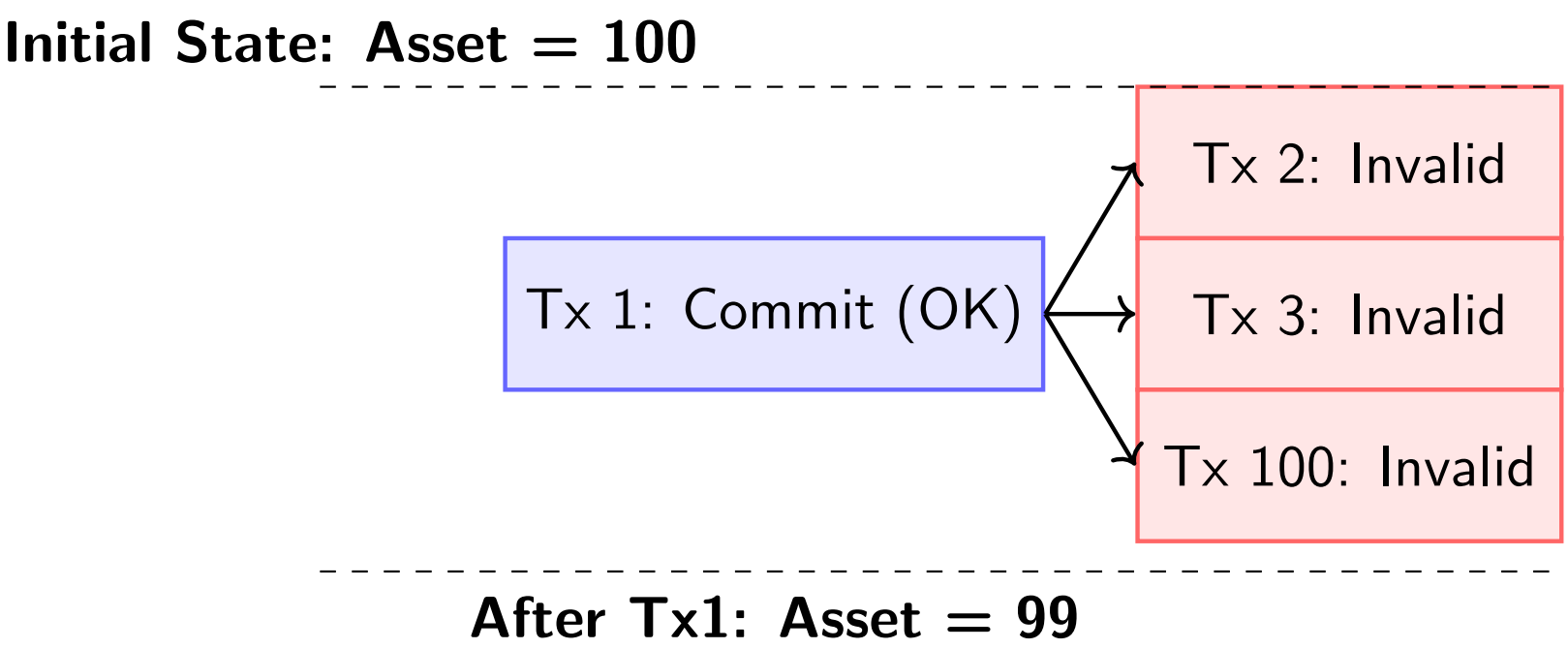
[3] Elli Androulaki et al. "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains". In: *Proceedings of the Thirteenth EuroSys Conference*. EuroSys '18. ACM, 2018, 30:1–30:15. DOI: 10.1145/3190508.3190538.

Key Issues in Current Fabric Architecture

Key Issues Limiting Performance

- **Sequential Commit:** All transactions are committed one-by-one, even if they access independent keys.
  - Multi-core capabilities are not leveraged during the commit phase.
- **No Concurrency Awareness:** Fabric does not distinguish between dependent and independent transactions during commit.
- **False Rejections:** Transactions with valid logic may be rejected due to stale versions.

Conflict Scenario: Version Mismatch



Explanation

Tx1 is committed because the version of the asset it read during endorsement matches the latest version in the world state at the time of validation.
Tx2, Tx3, and Tx100 are rejected because **Tx1's commit updates the version number** of the asset.

Table of Contents

Proposed DAG-aware Fabric Architecture

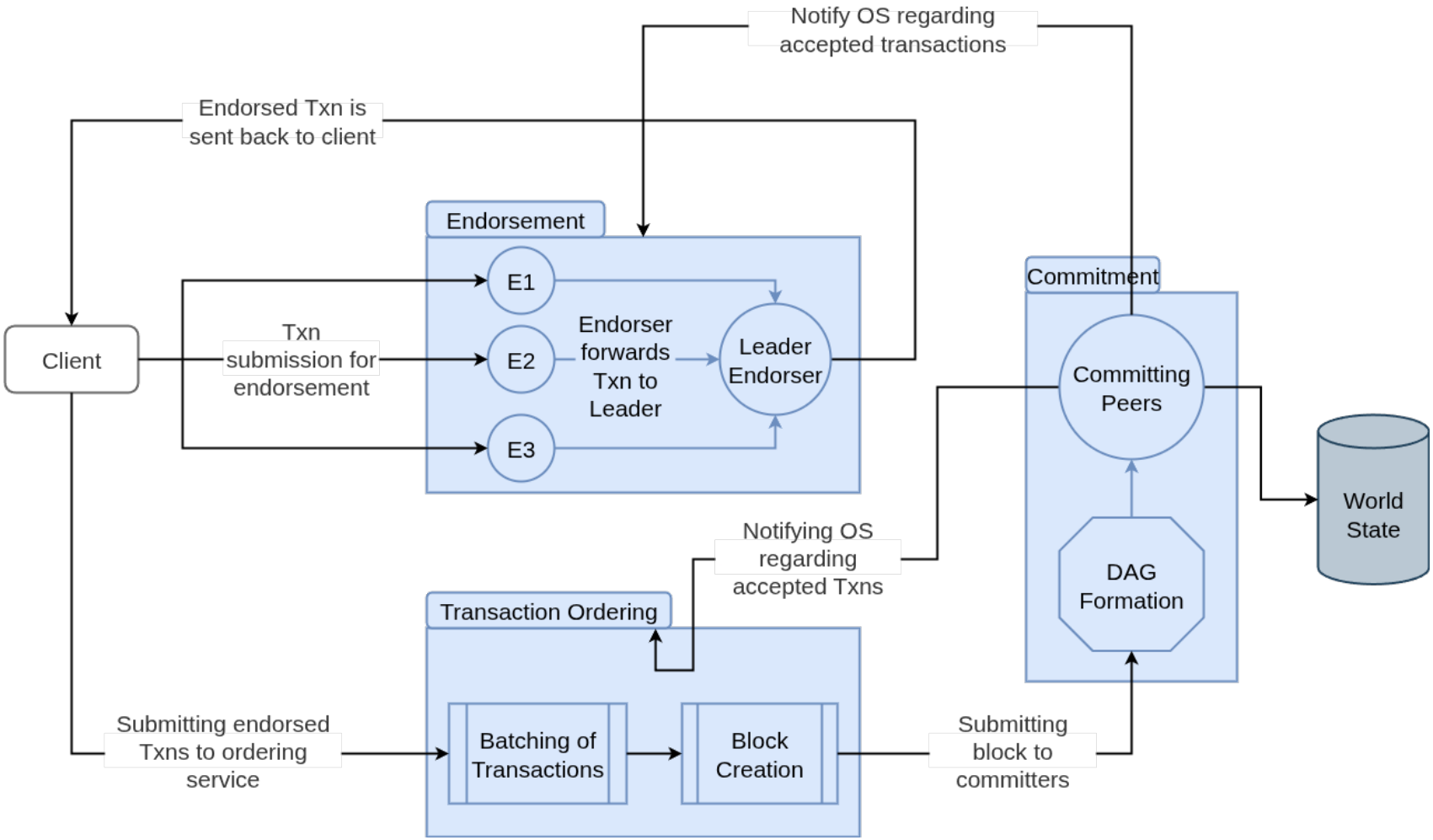


Figure: Proposed DAG-aware Fabric Architecture

Dependency Flagging at Endorsement

- Leader endorser simulates transaction
- Reads/writes tracked using a hashmap
- Dependency determined based on:
  - Recent key modifications
  - Active endorsement conflicts
- Flag set: `flag = 0 (independent)`, `flag = 1 (dependent)`

Example:

- Tx1: modifies key K $\rightarrow$ `flag = 0`
- Tx2–Tx100: access same K $\rightarrow$ `flag = 1`

Proposed DAG-aware Fabric Architecture

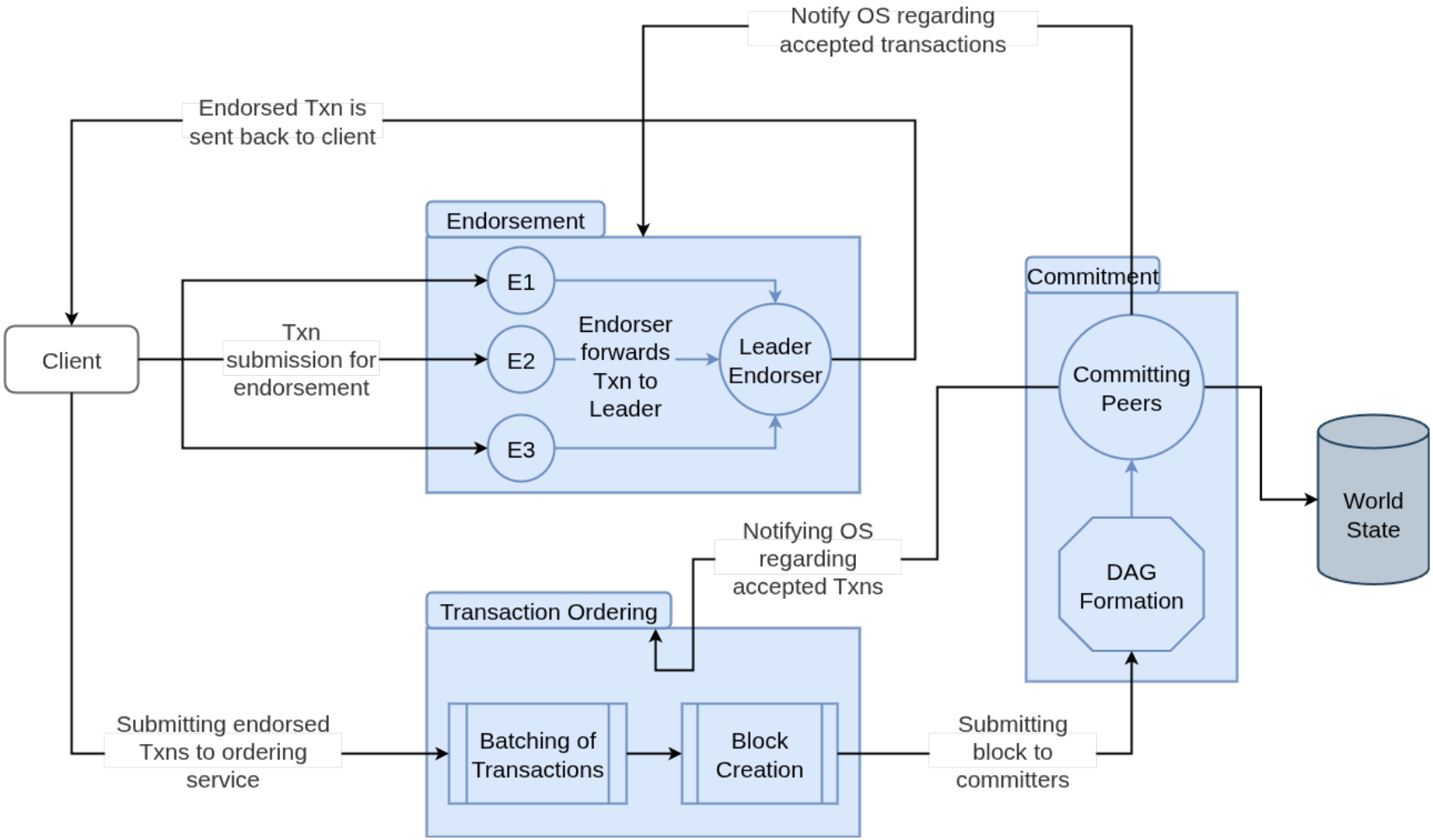
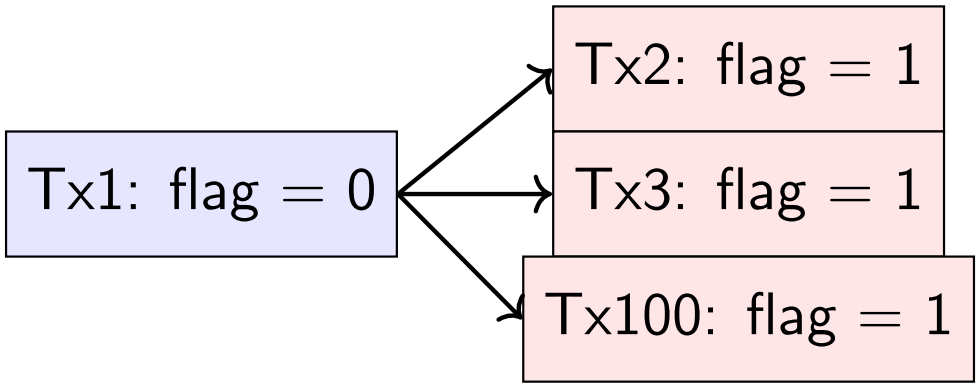


Figure: Proposed DAG-aware Fabric Architecture

Illustration: Dependency Flagging



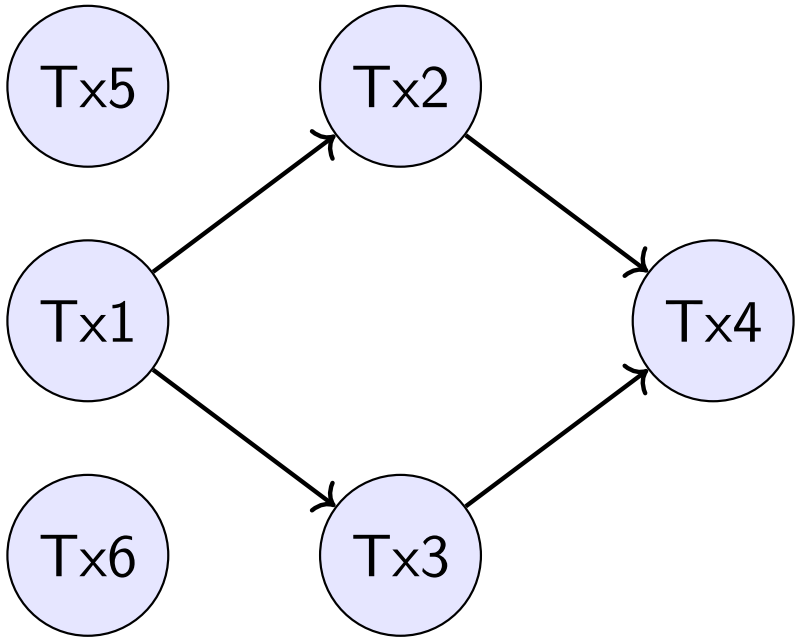
Key accessed: Asset = 100

Tx1 modifies it → Tx2-Tx100 become dependent

DAG Construction at Commit

- Committer constructs Directed Acyclic Graph (DAG)
- Each transaction = node
- Edge A $\rightarrow$ B = Tx B depends on Tx A
- Topological sort organizes DAG into levels
  - Level 0: Independent Txs
  - Higher levels: Respect dependencies

Illustration: DAG-based Commit Execution



Independent transactions (Tx1, Tx5, Tx6) run first. Tx2 and Tx3 depend on Tx1. Tx4 depends on Tx2 and Tx3.

Proposed DAG-aware Fabric Architecture

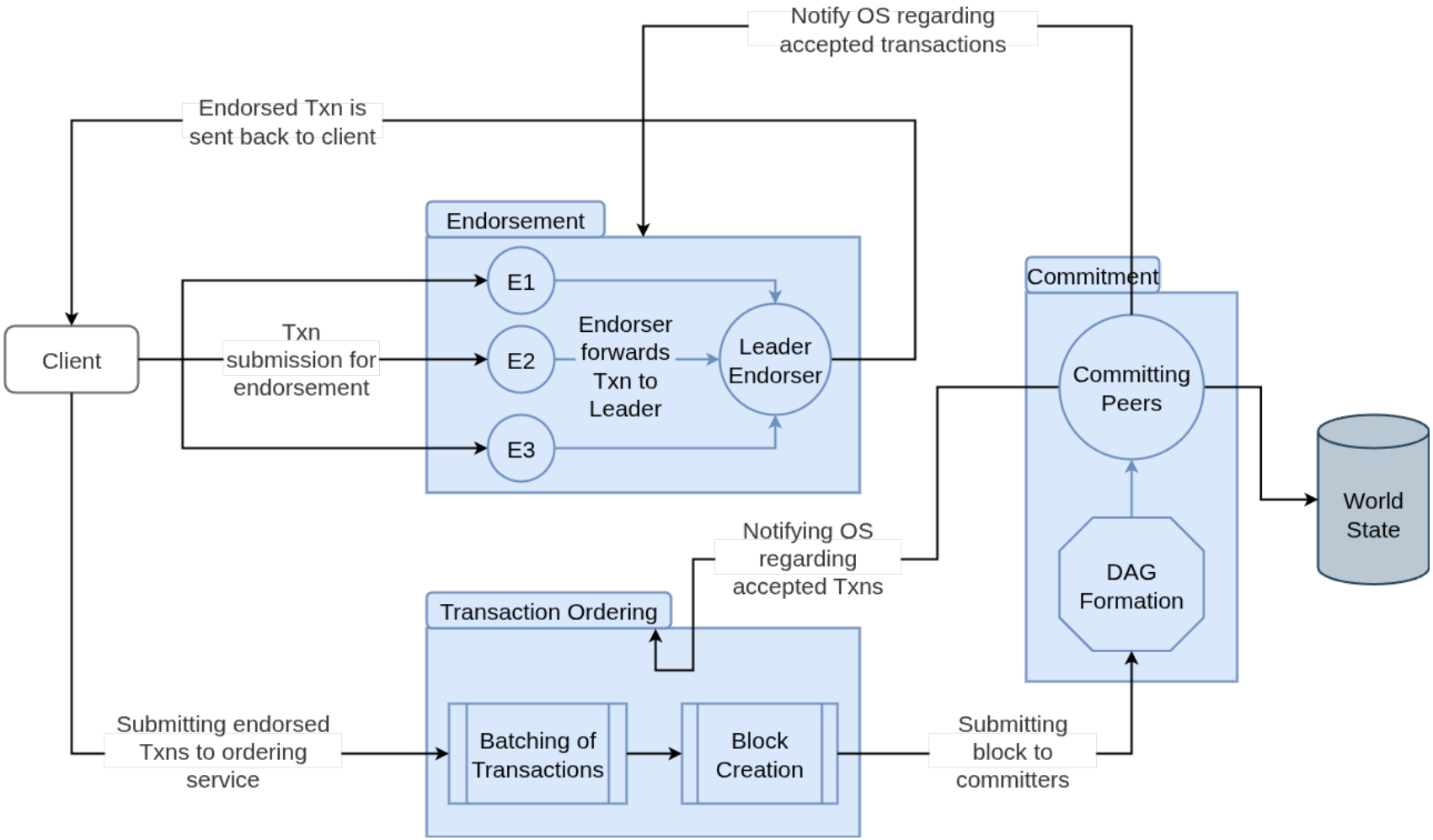
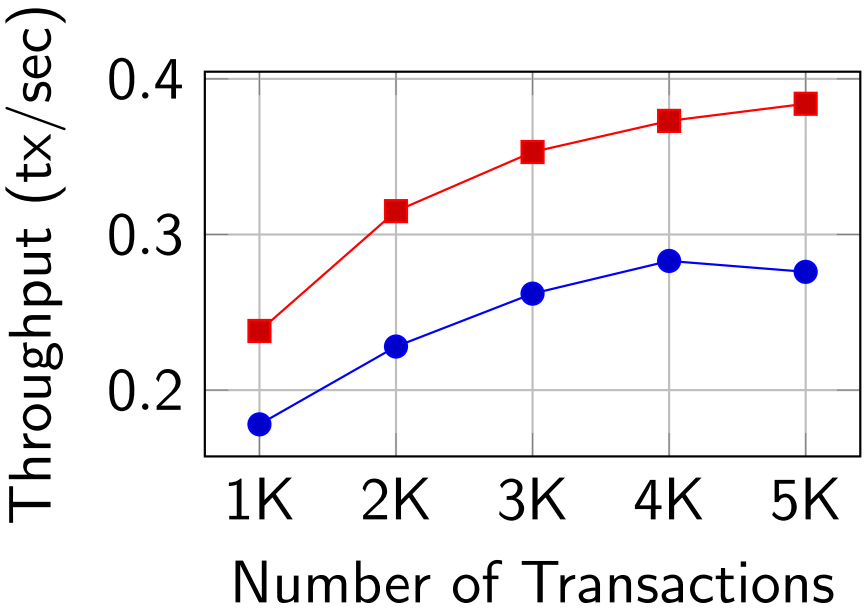


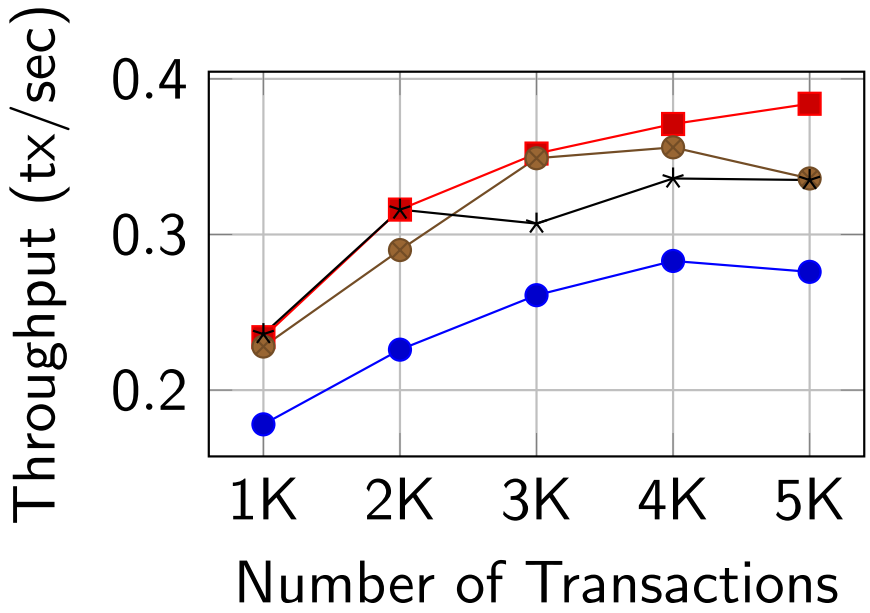
Figure: Proposed DAG-aware Fabric Architecture

Table of Contents

Experiment - Throughput (transactions per second) VS Transactions



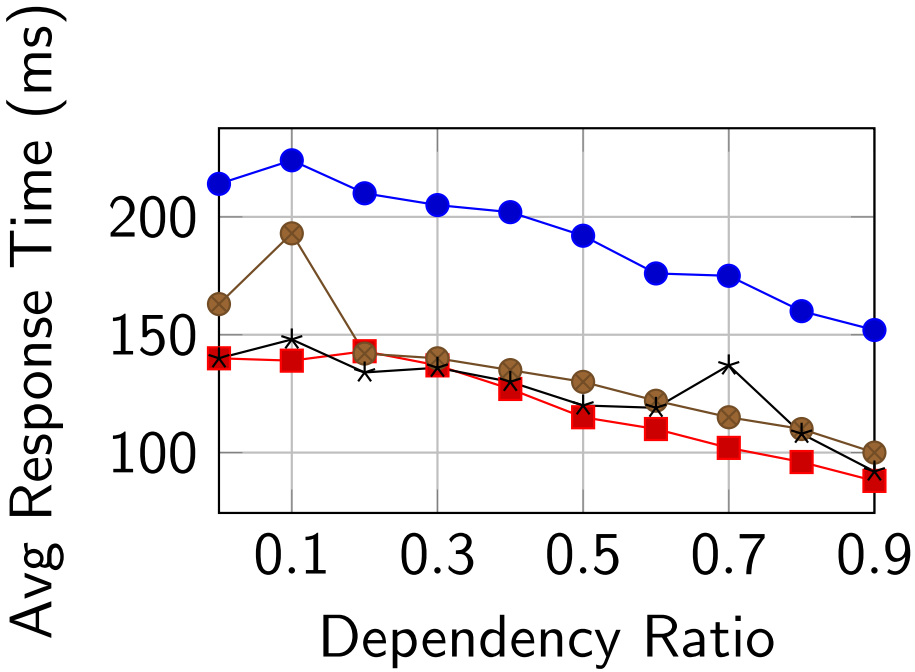
(a) Original vs Modified Fabric



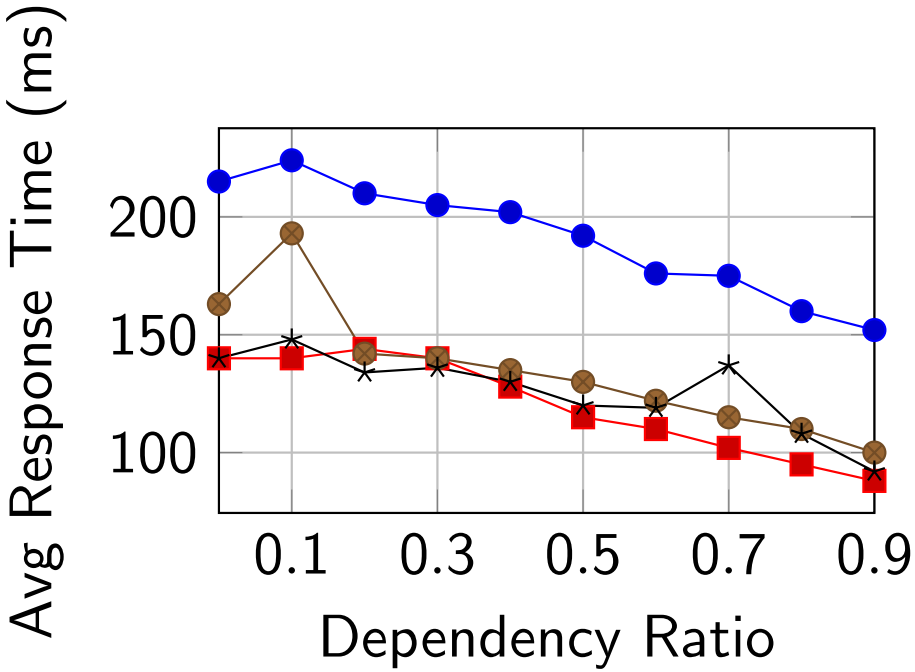
(b) Parallelism in Modified Fabric

Figure: Experiment 1: Impact of Number of Transactions on Throughput

Experiment - Avg Response Time (ms) VS Dependency Ratio



(a) Original vs Modified Fabric



(b) Parallelism in Modified Fabric

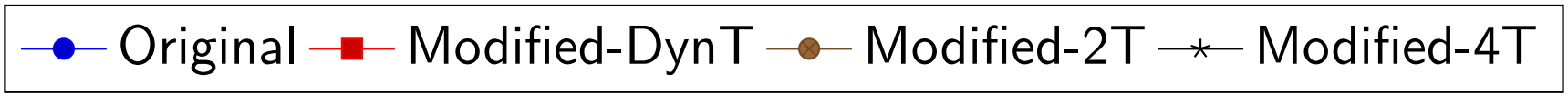


Figure: Experiment 2: Impact of Dependency Ratio on Average Response Time

Table of Contents

Impact and Benefits

- ↓ **Rejection Rate:** Early detection prevents commit-time failures
- ↑ **Resource Utilization:** Independent txs run concurrently
- ↑ **Throughput,** ↓ **Latency:** DAG execution boosts performance (up to 40%)
- **Minimal Fabric Disruption:** Chaincode, ordering, clients untouched

Conclusion

- Hyperledger Fabric's transaction model lacks concurrency awareness, which limits its performance under high transaction loads.
- We proposed a dependency-aware execution scheme that identifies transaction dependencies, and executes them using a parallel scheduler.
- The implementation extends the existing Fabric codebase with minimal modifications.
- In future work, we plan to explore decentralized endorsing system while tracking dependencies