Poster Abstract: OrderlessChain: A CRDT-Enabled Blockchain Without Total Global Order of Transactions

Pezhman Nasirifard
Technical University of Munich
Munich, Germany
p.nasirifard@tum.de

Ruben Mayer
Technical University of Munich
Munich, Germany
ruben.mayer@tum.de

Hans-Arno Jacobsen
University of Toronto
Toronto, Canada
jacobsen@eecg.toronto.edu

ABSTRACT

Blockchains often use coordination-based consensus protocols to offer trust in a Byzantine environment and to serialize transactions to preserve the application's invariants. However, coordination can be a performance bottleneck. There exist application-level invariants known as *Invariant-Confluence (I-confluence)*, which can be preserved without coordination and benefit from improved performance. We introduce OrderlessChain, a coordination-free permissioned blockchain for the safe execution of I-confluent applications.

CCS CONCEPTS

• **Computer systems organization** → **Distributed architectures**.

KEYWORDS

Blockchain, I-Confluence, Conflict-free Replicated Data Type

ACM Reference Format:
Pezhman Nasirifard, Ruben Mayer, and Hans-Arno Jacobsen. 2022. Poster Abstract: OrderlessChain: A CRDT-Enabled Blockchain Without Total Global Order of Transactions. In *23rd International Middleware Conference Demos and Posters (Middleware '22), November 7–11, 2022, Quebec, QC, Canada.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3565386.3565486

1 INTRODUCTION

Blockchains rely on consensus protocols to offer trust in a trustless environment. Another property of consensus protocols is serializing transactions into a total global order, which is required to preserve the correctness of the application's state. For example, serialization prevents a user's negative account balance on Ethereum. However, coordinating to reach consensus in several public and permissioned blockchains is a bottleneck [7]. Decreasing coordination and increasing concurrent execution of transactions are essential factors in improving the throughput and latency [2]. However, eliminating the coordination may violate the application-level correctness invariants. For example, Ethereum cannot prevent double-spending attacks without coordination and negative account balances [5, 6].

There exist *Invariant-Confluent (I-confluent)* invariants [2], which are application-level correctness invariants that can be preserved without coordination and can occur in any order. For example, consider several transactions that deposit funds to users' accounts. The final state of the account is independent of the transactions' order. One technique for creating I-confluent applications is *Conflict-free Replicated Data Types (CRDTs)* [11]. Bailis et al. [2] demonstrated the safety and liveness of I-confluent applications in non-Byzantine and eventually consistent environments. However, the safety and liveness of I-confluent applications in a Byzantine environment without paying the high coordination cost is challenging [7]. This paper presents OrderlessChain, a coordination-free permissioned blockchain for creating safe and live applications in a Byzantine environment.

2 ARCHITECTURE AND PROTOCOL

OrderlessChain is an asynchronous permissioned blockchain and consists of organizations and clients. Organizations are responsible for hosting smart contracts, executing transactions, and managing the application's ledger consisting of an append-only hash-chain log and a database representing the current application state. Developers create smart contracts containing the application's logic and define the *Endorsement Policy (EP)* of the application. EP specifies which organizations must sign and commit the transactions and has the format $EP: \{q \, of \, n\}$, where n is the number of organizations, and q is the minimum number of endorsing and committing organizations. OrderlessChain follows a two-phase protocol, as demonstrated in Figure 1. We also published an extended version of the paper [8].

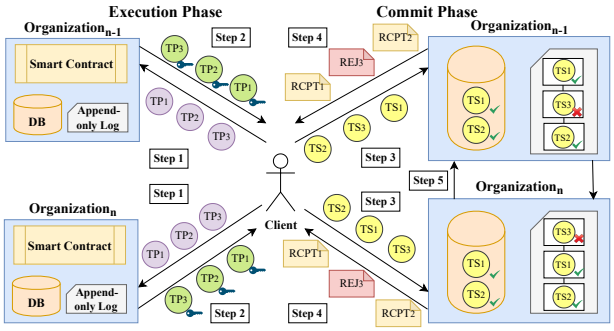


Figure 1: Transaction lifecycle on OrderlessChain.

Execution Phase – The client creates proposal TP_i containing the input parameters for the smart contract and broadcasts it to at least q organizations according to EP (Step 1 in Figure 1). The organizations execute the smart contract with the provided parameters and create a write-set containing I-confluent operations for modifying the application's state based on the CRDT methodology. The organizations hash and sign the write-set with their private key based on public-key cryptography and deliver the write-sets and the signatures as endorsements to the client (Step 2). This ensures that the write-set cannot be tampered with without invalidating the signature.

Middleware '22, November 7–11, 2022, Quebec, QC, Canada

ACM ISBN 978-1-4503-9931-9/22/11…$15.00
https://doi.org/10.1145/3565386.3565486

Commit Phase – The client waits to receive the minimum number of endorsements required by EP. If the write-sets of all endorsements are identical, it assembles a transaction TS_i based on the endorsements. The client adds the endorsement's write-set to the TS_i's write-set and hashes and signs the transaction's write-set with its private key to create a signature and includes it in TS_i. The client sends back the transactions to at least q organizations based on EP (Step 3). TS_i is validated and committed if an organization has not previously committed it. First, organizations verify whether the transaction's endorsements and the client's signature are valid and whether endorsements satisfy the EP. This validation shows that the organizations created identical write-sets, and the client did not tamper with the endorsements. If TS_i passes the validation, each organization updates its database with its write-set, whereas all valid and invalid transactions are appended to the log. The organization creates a block $Block_h$, which contains TS_i and the hash of the previous block, and appends the created block to the log. A signed receipt containing the block's hash for valid transactions is sent to the client (Step 4). Otherwise, a rejection is sent. As the receipt contains the block's hash, which is dependent on the previous blocks, a Byzantine organization cannot modify TS_i without invalidating the TS_i receipt and the other transactions. Finally, the organization periodically gossips the transactions to other organizations (Step 5).

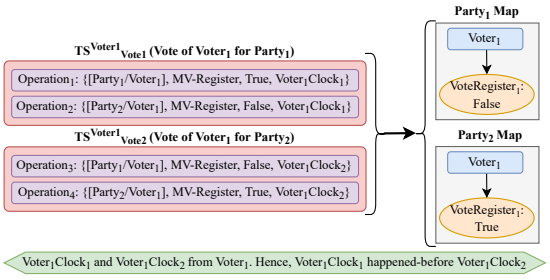


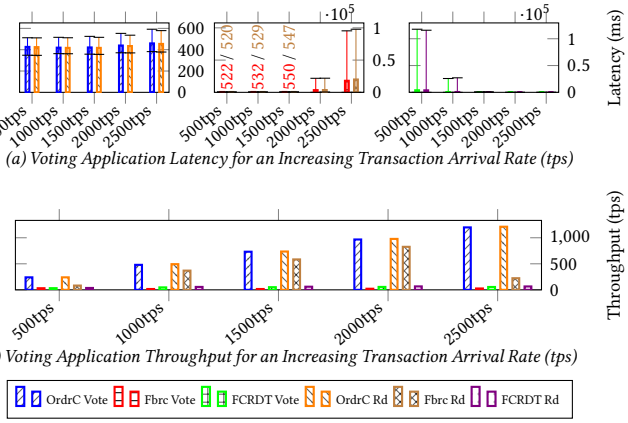
Figure 2: Preserving the invariant for the voting application.

3 ORDERLESSCHAIN APPLICATION

We create a voting application (other applications are discussed in other studies [8–10]). Each voter votes for one party among the participating parties. Every party is assigned to an organization. The application is correct if each voter votes for at most one party. Hence, we recognize one invariant: *maximally one vote per voter*. For modeling this application in the smart contract, each party is modeled as a CRDT Map [4] containing key-value pairs. The key is the voter's ID, and the value is a Multi-Value Register (MV-Register) [4] that stores the voter's vote. Each client keeps track of a logical clock. The clock is incremented with every new proposal and is used to infer the *happened-before* relations. The developer implements the smart contract to create the operations for modifying the party's CRDT Map for the cast vote. Consider an election with two parties, where the voter votes for P_1. The smart contract creates two operations. One operation sets the voter's MV-Register in P_1 to *true*, and the other operation sets it in P_2 to *false*. These two operations are included in the write-set. Since organizations do not coordinate, a voter can submit several votes. To show that the transactions are I-confluent concerning *maximally one vote per voter*, we reason as follows. Consider the transaction set $\{TS^{Voter1}_{Vote1}, TS^{Voter1}_{Vote2}\}$ of a voter voting for two different parties, as shown in Figure 2. Based on the client's logical clock, a happened-before relation exists between transactions' operations. Therefore, independent of their processing order, operations in TS^{Voter1}_{Vote2} overwrite the effects of operations in TS^{Voter1}_{Vote1}. Hence, we count only one of the submitted votes.

4 EVALUATION

We compare our system to the prototypes of coordination-based blockchains of *Fabric* [1] and *FabricCRDT* [7] based on the voting applications. The application's smart contracts have two functions of *Vote* and *ReadVoteCount*. The experiments are conducted with eight organizations, the $EP: \{4\,of\,8\}$, and the uniform workload. The results are shown in Figure 3. We observe that OrderlessChain demonstrates a higher throughput with constant latency.



(a) Voting Application Latency for an Increasing Transaction Arrival Rate (tps)

(b) Voting Application Throughput for an Increasing Transaction Arrival Rate (tps)

Figure 3: Experiments with voting application.

5 RELATED WORK

Permissioned blockchains such as Fabric [1] use coordination-based protocols that can be a bottleneck. Kleppmann et al. [5] introduced a BFT replicated database for processing I-confluent transactions. Studies on CRDTs in blockchains are limited. Vegvisir [3] studied integrating CRDTs with a blockchain without support for smart contracts.

REFERENCES

[1] E. Androulaki, A. Barger, V. Bortnikov, and *et al.* 2018. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. In *EuroSys.*

[2] P. Bailis, A. Fekete, M. J. Franklin, and *et al.* 2014. Coordination Avoidance in Database Systems. In *VLDB.*

[3] K. Karlsson, W. Jiang, S. Wicker, and *et al.* 2018. Vegvisir: A Partition-Tolerant Blockchain for the Internet-of-Things. In *ICDCS.*

[4] M. Kleppmann and A. R. Beresford. 2017. A Conflict-free Replicated JSON Datatype. *IEEE TPDS* (2017).

[5] M. Kleppmann and H. Howard. 2020. Byzantine Eventual Consistency and the Fundamental Limits of Peer-to-Peer Databases. *CoRR* (2020). arXiv:2012.00472

[6] F. Kohlbrenner, P. Nasirifard, C. Löbel, and H.-A. Jacobsen. 2019. A Blockchain-Based Payment and Validity Check System for Vehicle Services. In *Middleware Demos and Posters.*

[7] P. Nasirifard, R. Mayer, and H.-A. Jacobsen. 2019. FabricCRDT: A Conflict-free Replicated Datatypes Approach to Permissioned Blockchains. In *Middleware.*

[8] P. Nasirifard, R. Mayer, and H.-A. Jacobsen. 2022. OrderlessChain: Do Permissioned Blockchains Need Total Global Order of Transactions? *CoRR* (2022). https://doi.org/10.48550/arXiv.2210.01477 arXiv:2210.01477

[9] P. Nasirifard, R. Mayer, and H.-A. Jacobsen. 2022. OrderlessFile: A CRDT-Enabled Permissioned Blockchain for File Storage. In *Middleware Demos and Posters.*

[10] P. Nasirifard, R. Mayer, and H.-A. Jacobsen. 2022. OrderlessFL: A CRDT-Enabled Permissioned Blockchain for Federated Learning. In *Middleware Demos and Posters.*

[11] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski. 2011. Conflict-free Replicated Data Types. In *SSS.*