TL;DR

ProofChain introduces a novel transaction ordering mechanism called Client-Side Ordinal Transaction Ordering (COTO) that achieves deterministic and uniform ordering of transactions across shards without requiring shard synchronization, enhancing the scalability of the system.

Background

Existing blockchain systems face scalability challenges due to limitations in their transaction ordering approaches. Global consensus ordering suffers from limited throughput, while shard-level consensus ordering requires cross-shard synchronization. DAG-based approaches face challenges in transaction finality and handling conflicting transactions.

Proposal

COTO assigns a unique ordinal rank to each transaction based on the sender's shard ID, logical clock value, timestamp, and hash of the serialized transaction. The ordinal rank determines the order in which transactions are processed. Each shard validates and processes transactions independently using the assigned ordinal ranks. Processed transactions are propagated to other shards through either Client View (signed transactions) or Global View (Global State Proofs) propagation modes.

Let \mathcal{T}

denote the set of all transactions in the ProofChain network. Each transaction tx \in \mathcal{T}

is represented as a tuple:

tx = (s, r, a, n, t)

where:

s

is the sender's address

r

is the recipient's address

a

is the transaction amount

n

is the transaction nonce

t

is the transaction timestamp

The ordinal rank of a transaction tx

is calculated as follows:

OrdinalRank(tx) = (Shard(tx), LogicalClock(tx), Timestamp(tx), Hash(Serialize(tx)))

where:

Shard(tx) = hash(tx.s) \bmod m

, with hash

being a cryptographic hash function and m

the total number of shards

LogicalClock(tx)

is the current logical clock value of the client

Timestamp(tx)

is the current timestamp at the time of transaction submission

Hash(Serialize(tx))

is the cryptographic hash of the serialized transaction

Illustration

Consider two transactions, tx_1

and tx_2

, submitted to the ProofChain network. The ordinal ranks of these transactions are calculated as follows:

OrdinalRank(tx_1) = (Shard(tx_1), LogicalClock(tx_1), Timestamp(tx_1), Hash(Serialize(tx_1)))

OrdinalRank(tx_2) = (Shard(tx_2), LogicalClock(tx_2), Timestamp(tx_2), Hash(Serialize(tx_2)))

Assuming tx_1

and tx_2

belong to different shards and have unique ordinal ranks, they can be processed independently by their respective shards without requiring cross-shard synchronization.

Advantages

COTO offers several advantages over alternative transaction ordering approaches:

1. Scalability: COTO allows parallel transaction processing across shards, eliminating the need for global consensus or cross-shard synchronization.
2. Deterministic Ordering: COTO ensures a deterministic ordering of transactions based on their unique ordinal ranks, providing consistency across shards.
3. Efficient Propagation: COTO supports efficient propagation of shard states through succinct Global State Proofs (GSPs) or signed transactions.

Applications

COTO can be applied in various scenarios where scalability and deterministic ordering of transactions are crucial:

1. High-Throughput Payment Systems: COTO enables fast and parallel processing of transactions, making it suitable for large-scale payment networks.
2. Decentralized Exchanges: COTO ensures a consistent ordering of trades across shards, facilitating efficient and fair execution of orders.
3. Supply Chain Management: COTO can be used to track and order events in supply chain networks, ensuring data integrity and consistency across participants.

Conclusion

Client-Side Ordinal Transaction Ordering (COTO) introduces a scalable and deterministic approach to transaction ordering in sharded blockchain networks. By assigning unique ordinal ranks to transactions and enabling parallel processing across shards, COTO addresses the scalability challenges faced by existing transaction ordering approaches. The mathematical formalisms, algorithms, and proofs presented demonstrate the correctness and scalability properties of COTO, making it a promising solution for various applications requiring high throughput and consistent transaction ordering.