

Blockchain System

Mia Weaver | Ruthvika Reddy Loka | Mahitha Pillodi

Introduction

- Blockchain: A distributed ledger technology that enables secure and transparent transactions without the need for intermediaries.
- Key Features: Decentralization, Transparency, Immutability, Security
- Examples: Bitcoin, Ethereum, Ripple



System Architecture

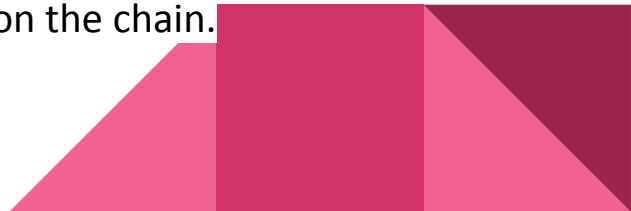
Overall Design

- Blocks: Containers that hold a set of transactions or data. Each block contains a unique hash that identifies it and links it to the previous block.
- Chain: A sequence of blocks linked together using cryptographic hashes. The chain is stored and maintained by multiple nodes in a distributed network.
- Distributed Network: Multiple nodes that maintain and validate the blockchain. Nodes can be run by anyone and participate in the validation process.



Proposed Blockchain Architecture

- Clients issue TXs to any node.
 - We use an automated process to send a predetermined amount of randomly generated transactions between random pairs of nodes.
- Nodes batch TXs locally and propagate them in blocks
- Block order established via Nakamoto consensus
 - We implement proof-of-work. Each node, upon timeout, attempts to mine its own block. If it receives a new block during this process, the mined block will be discarded.
- 4th block back committed; 3 most recent nodes pending/subject to change.
 - In case the most recent block is discarded, the transactions in it are not yet committed. TXs are committed once they are at least four blocks back on the chain.



Nakamoto Consensus

- Definition: A consensus mechanism used in blockchain networks to ensure agreement on the state of the blockchain.
- Named after Satoshi Nakamoto, the pseudonymous creator of Bitcoin.
- How it works: Nodes in the network compete to solve a mathematical puzzle (proof-of-work) to validate a new block and add it to the chain. The first node to solve the puzzle broadcasts the new block to the network, which then verifies and accepts it.
- Advantages: Decentralized, secure, resistant to tampering.



Proof of Work Mechanism

- Definition: A computational puzzle that miners (nodes) must solve to validate new blocks in a blockchain network.
- How it works: Miners use computing power to perform calculations and find a solution to the puzzle. The solution is then included in the new block as proof of work.
- Purpose: The proof-of-work algorithm ensures that the process of adding new blocks to the blockchain is resource-intensive and time-consuming, making it difficult for malicious actors to manipulate the blockchain.
- Bitcoin's Use: Bitcoin, the first blockchain-based cryptocurrency, utilizes the proof-of-work algorithm (SHA-256) to validate transactions and secure the network.



Why?

Benefits of Nakamoto Consensus and Proof-of-Work

- Decentralization: Nakamoto consensus allows for a decentralized network where no single entity controls the blockchain.
- Security: The proof-of-work algorithm ensures that adding new blocks requires significant computational power, making the blockchain resistant to attacks.
- Trust: The consensus mechanism and proof-of-work algorithm build trust among participants by providing transparency and immutability of transactions.





Implementation - Key Functionalities

Transactions

- Clients can issue any transactions to any node.
- A transaction will remain in a node's local pending pool until it is batched into a block and propagated into the network.
- If a block is for some reason discarded, its transactions will return to the nodes pending pool of transactions to be batched again later.
- If a node crashes, the pending transactions on that node are lost.



Transactions

- Transactions are communicated through the network as strings.
- Nodes parse these strings to extract the source node, destination node, amount, and reward associated with transactions. The timestamp of the transaction is included as well, so that equivalent transactions will have different hashes.
 - The reason for this design decision is that it is easier to send strings via GRPC than user-defined types.
- Single transactions are sent from client to server. For server-to-server interactions, block transactions are concatenated into a longer string and are sent in messages containing all other block information.



Block Creation

- A node can generate a block if it has any local pending transactions.
- A miner batches some or all of its transactions, and (pair-wise) hashes them together into a “merkle root.”
- The merkle root is hashed with the miner node’s identifier and with the previous hash.
- The previous hash allows for it to be fit on the chain.
- The hashing of the miner identifier prevents other nodes from claiming the rewards of the block.



Block Appending

A node accepts a received block and appends it to its local blockchain if:

- The received block's previous hash is equivalent to the block hash of the last block in the local chain.
- The node can recreate the received block's block hash by hashing the transactions stored in the received block with the miner ID and previous hash values.
 - Valid transactions and miner
- The last six characters of the resulting string when the received block nonce is hashed with the received block hash is \geq "fffffd"
 - Valid proof-of-work



Consensus Mechanism

- Whoever mines and propagates the block the quickest determines the transactions and order of transactions for that “step” or position on the ledger.
 - If a node receives a block while it is in the mining process, the mining process will continue but the block it is mining will be discarded, as the length of the chain has changed during mining.
- Nodes alert each other of their chain length and of the block hashes of their current chain, which can be used to reach consensus when there is a discrepancy among nodes.
 - If a block append fails and the replica is shorter than the miner’s chain, then the miner finds the last common block hash and sends a series of blocks until the replica is updated.



Hashing Mechanism

- We use the simple built in hash command provided by Python to hash data in our blockchain
- A block hash is comprised of:
 - Merkle root: Items in the list of transactions are recursively hashed together until one item, called the merkle root, remains.
 - Miner Node ID: The merkle root is hashed with the ID of the block miner, so that no later node may try to claim credit for the block.
 - Previous Hash: The previous hash value is hashed with the merkle root and miner node ID, so that it fits on to the chain.
- Nonce is hashed with the block hash to show proof-of-work.



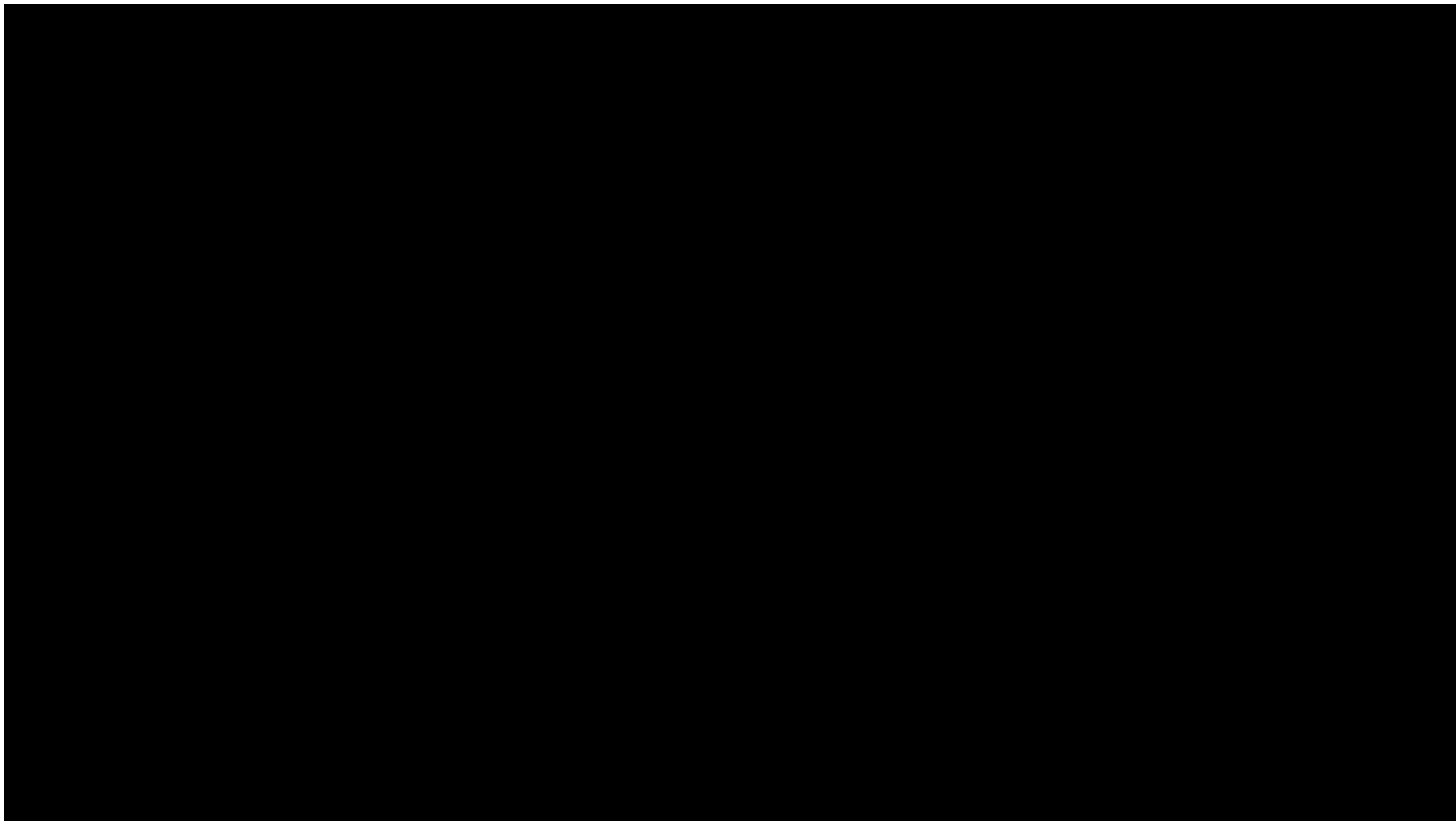
GRPC Implementation

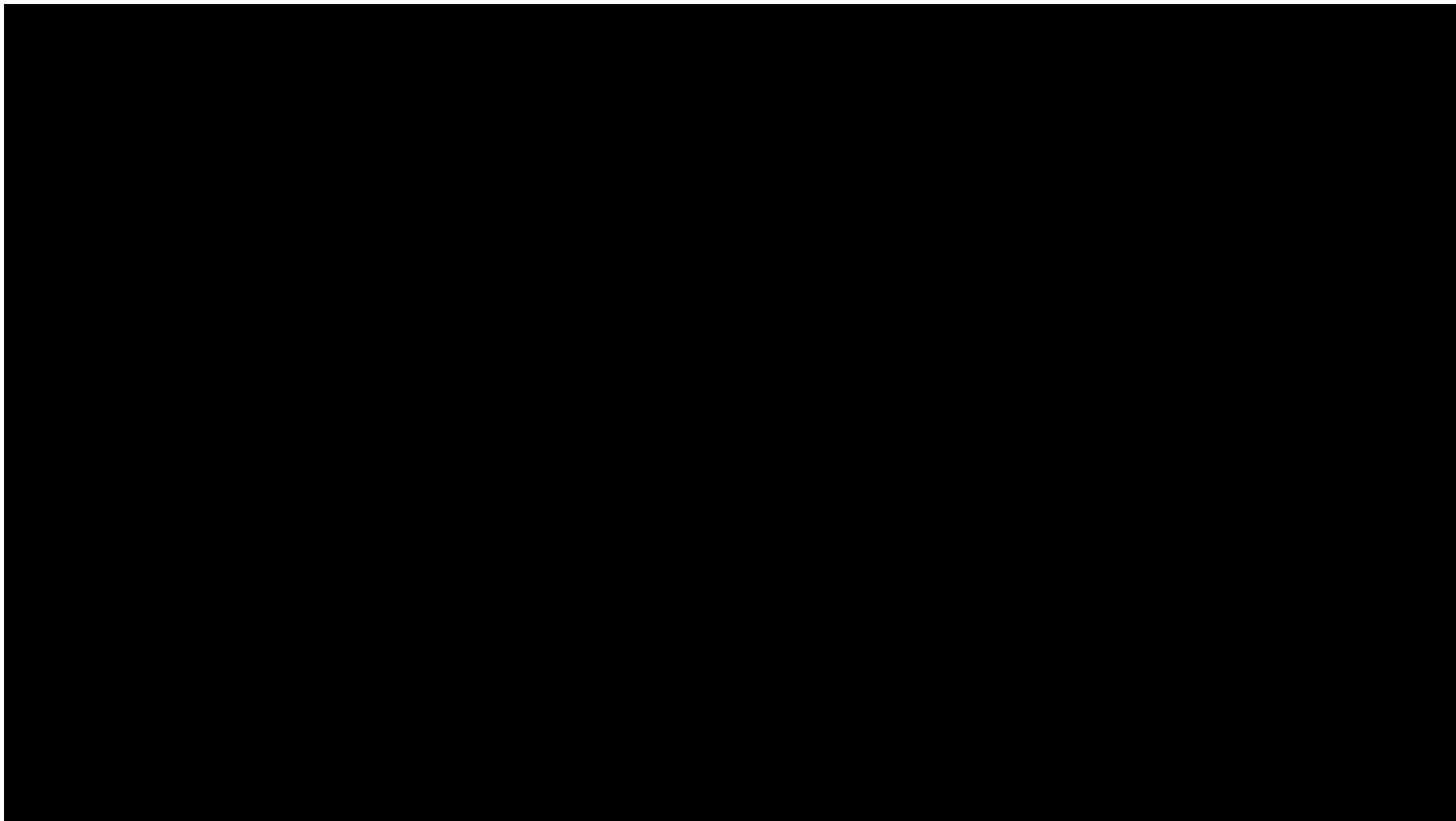
- We use Python gRPC to coordinate transaction issuing and block ordering.
- Nodes communicate **only** to propagate blocks. On the server side, each RPC invocation is either sending a block or responding to a block that was sent.
- When a miner sends a block, a replica responds with the status of the block (whether it was appended to the chain or not), the length of its current chain, and a list of block hashes in its current chain.





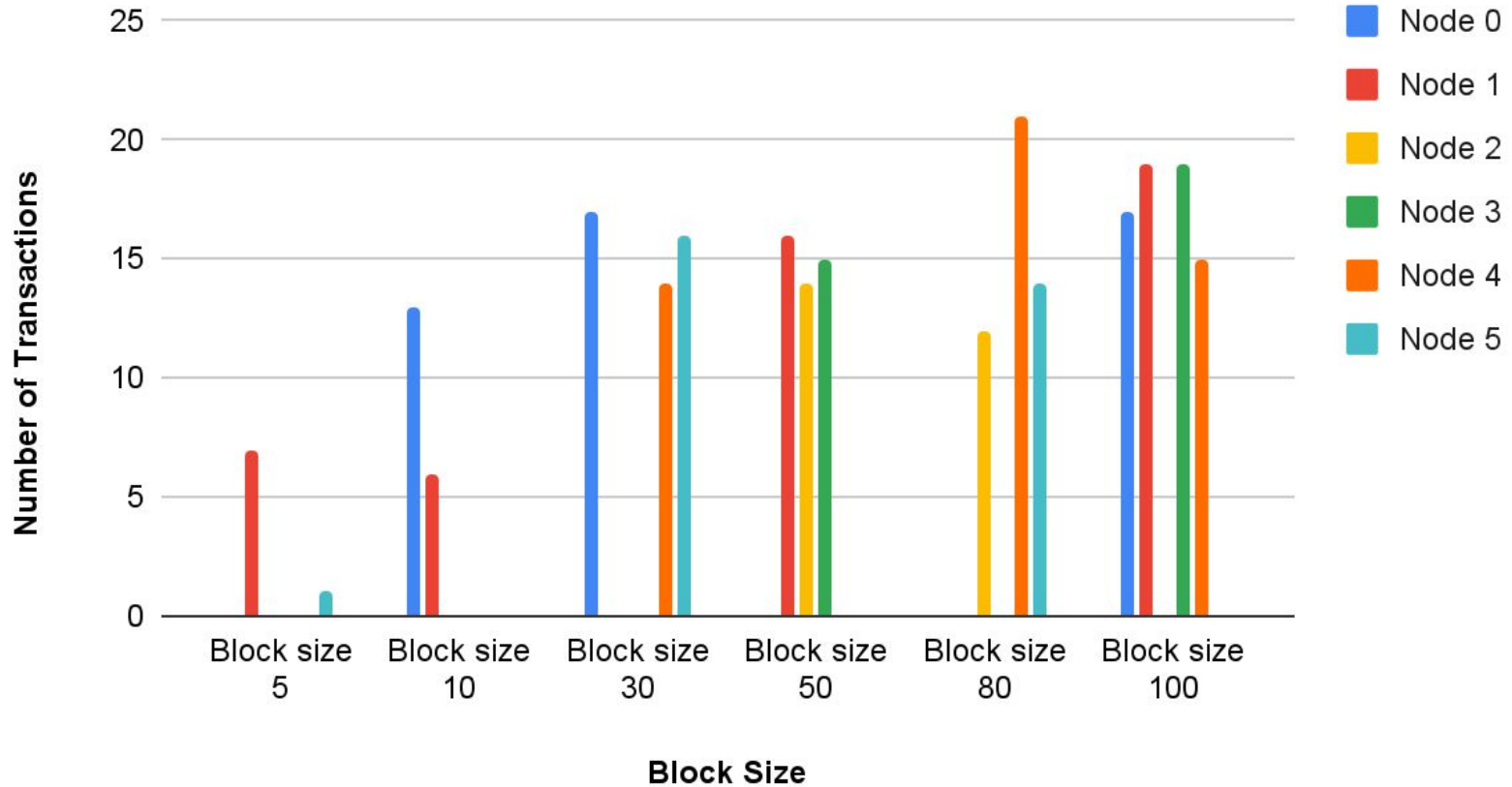
Demos



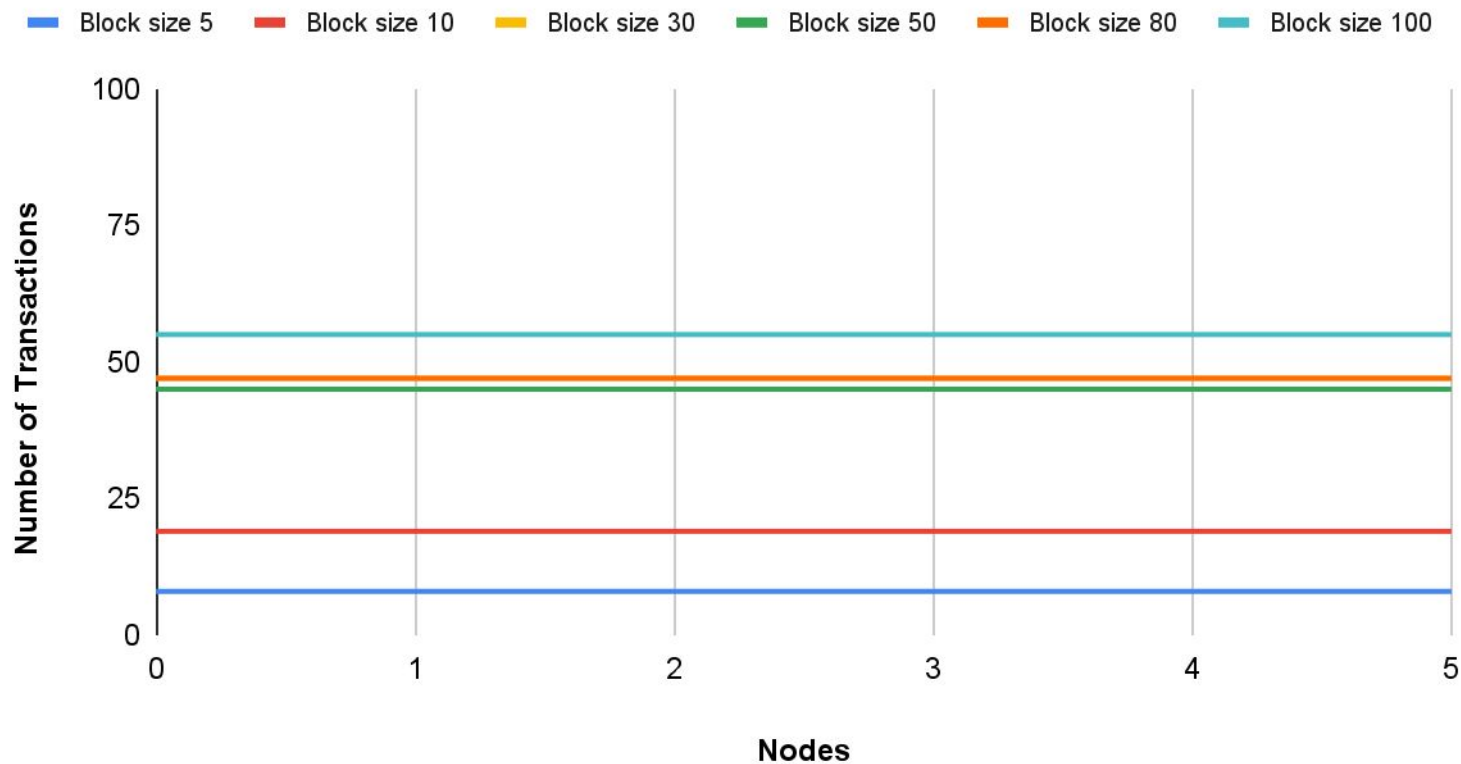


Performance Tests

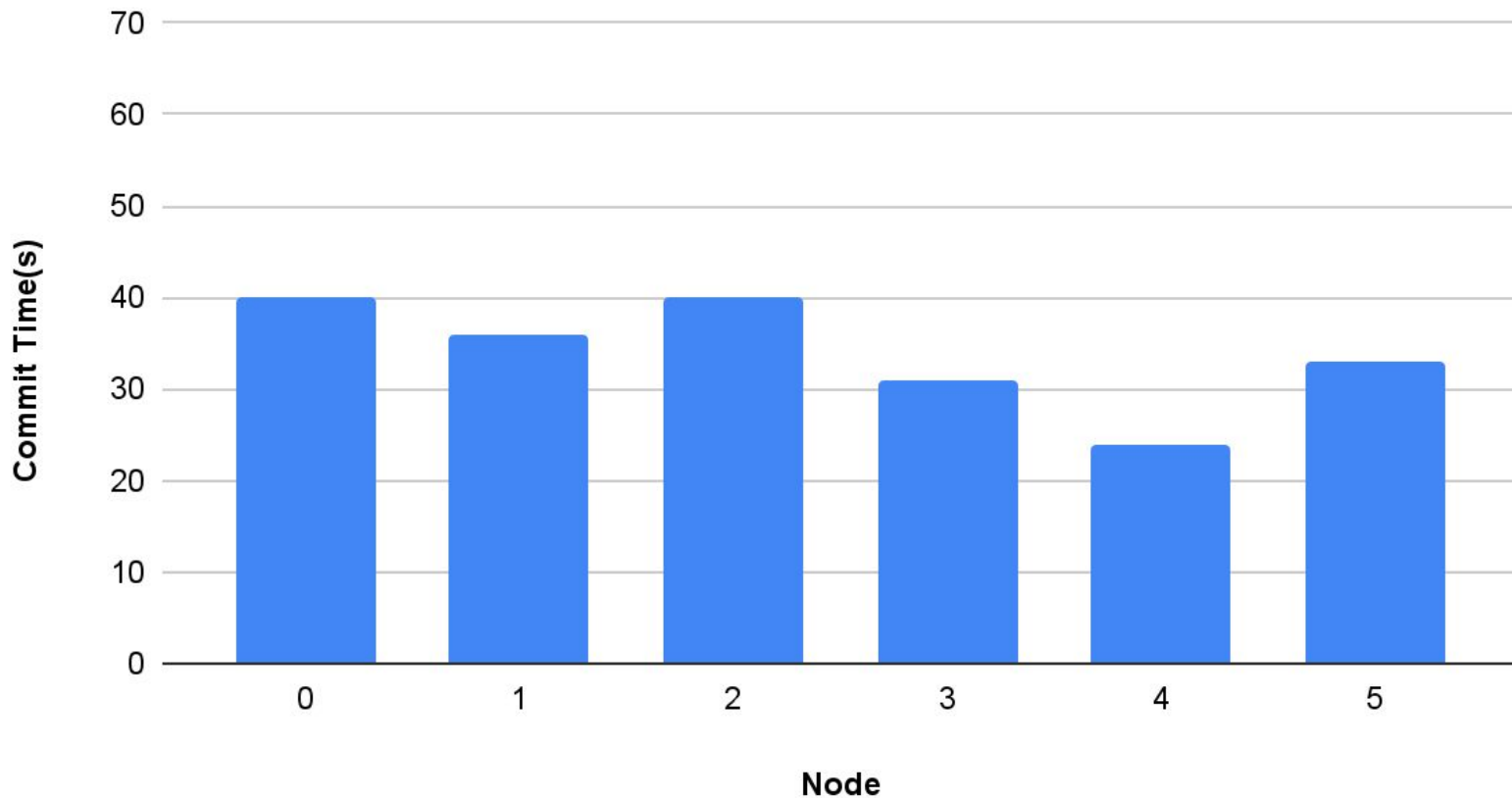
Pending Transactions in various block sizes



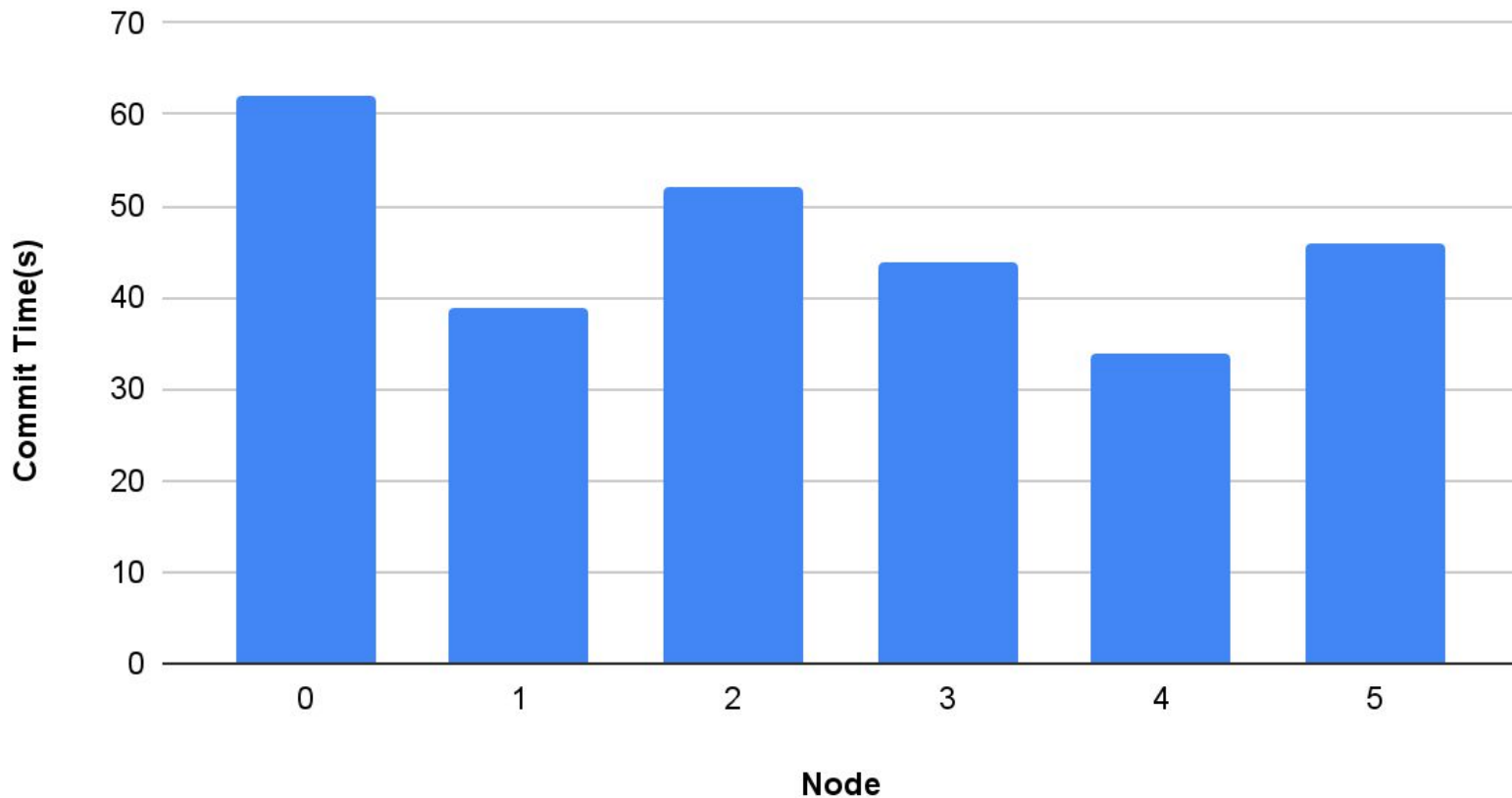
Uncommitted Transactions in various block sizes



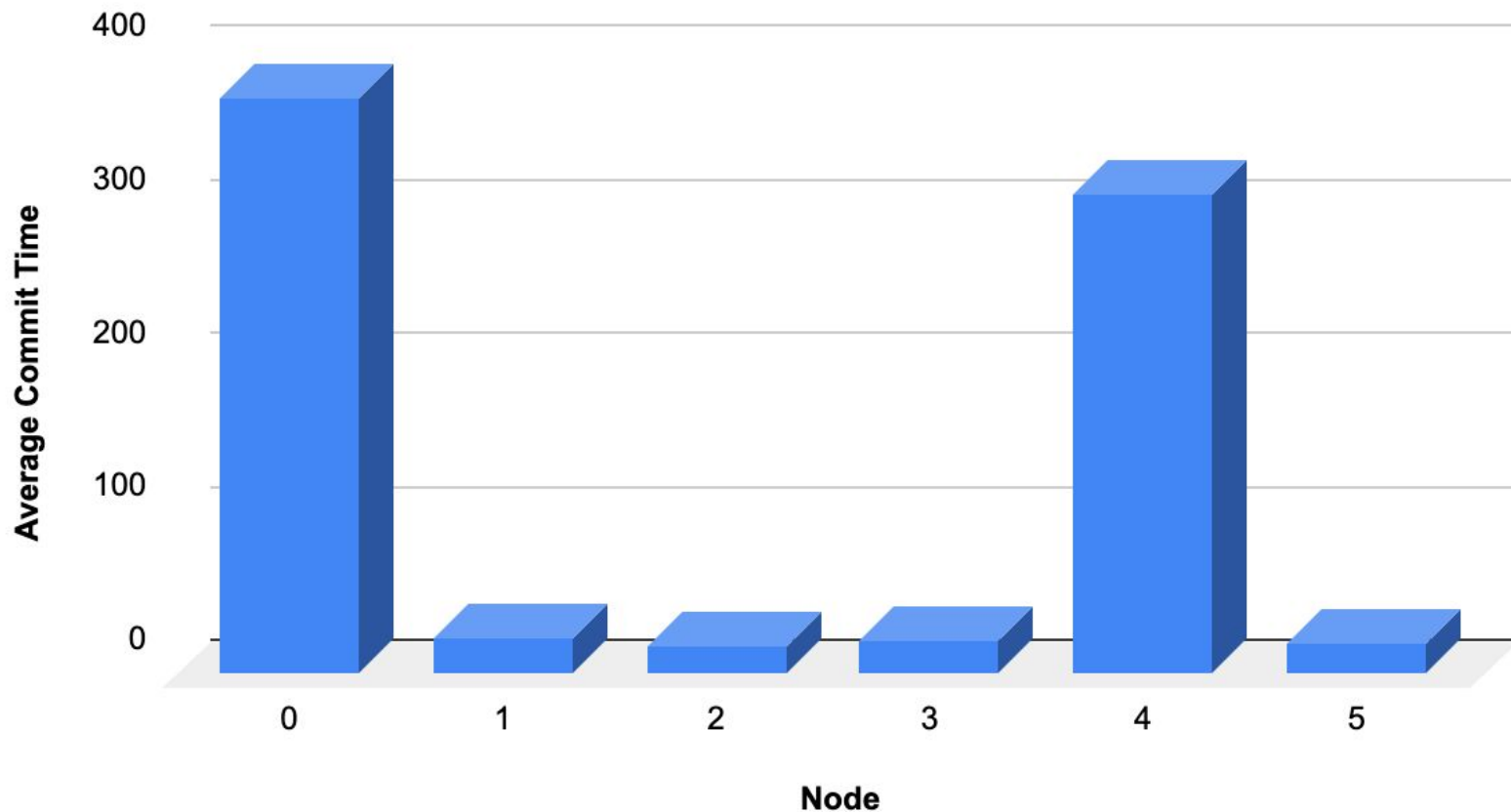
Average Commit Time for 100 sequential transactions



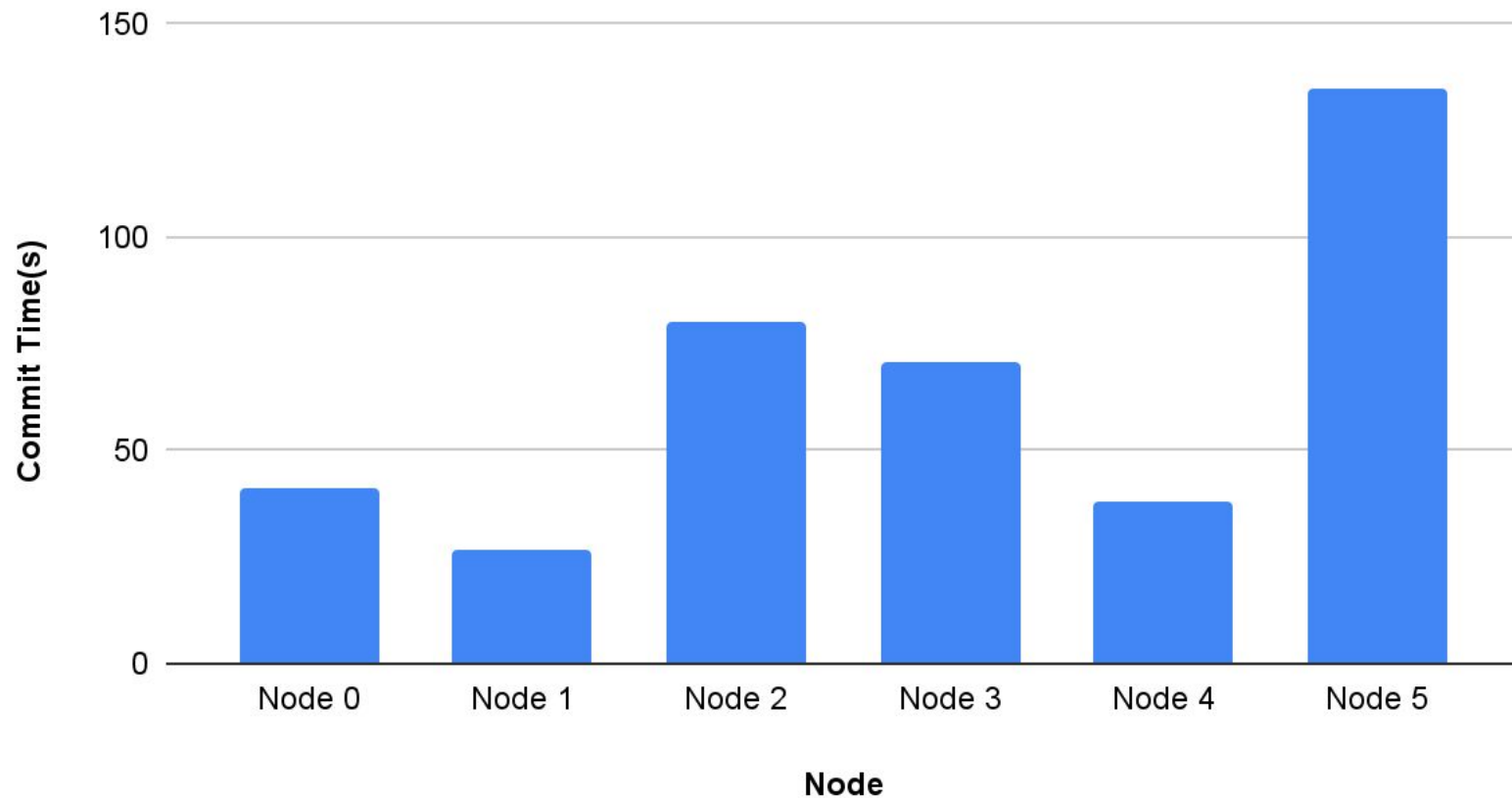
Average Commit Time for 150 sequential transactions



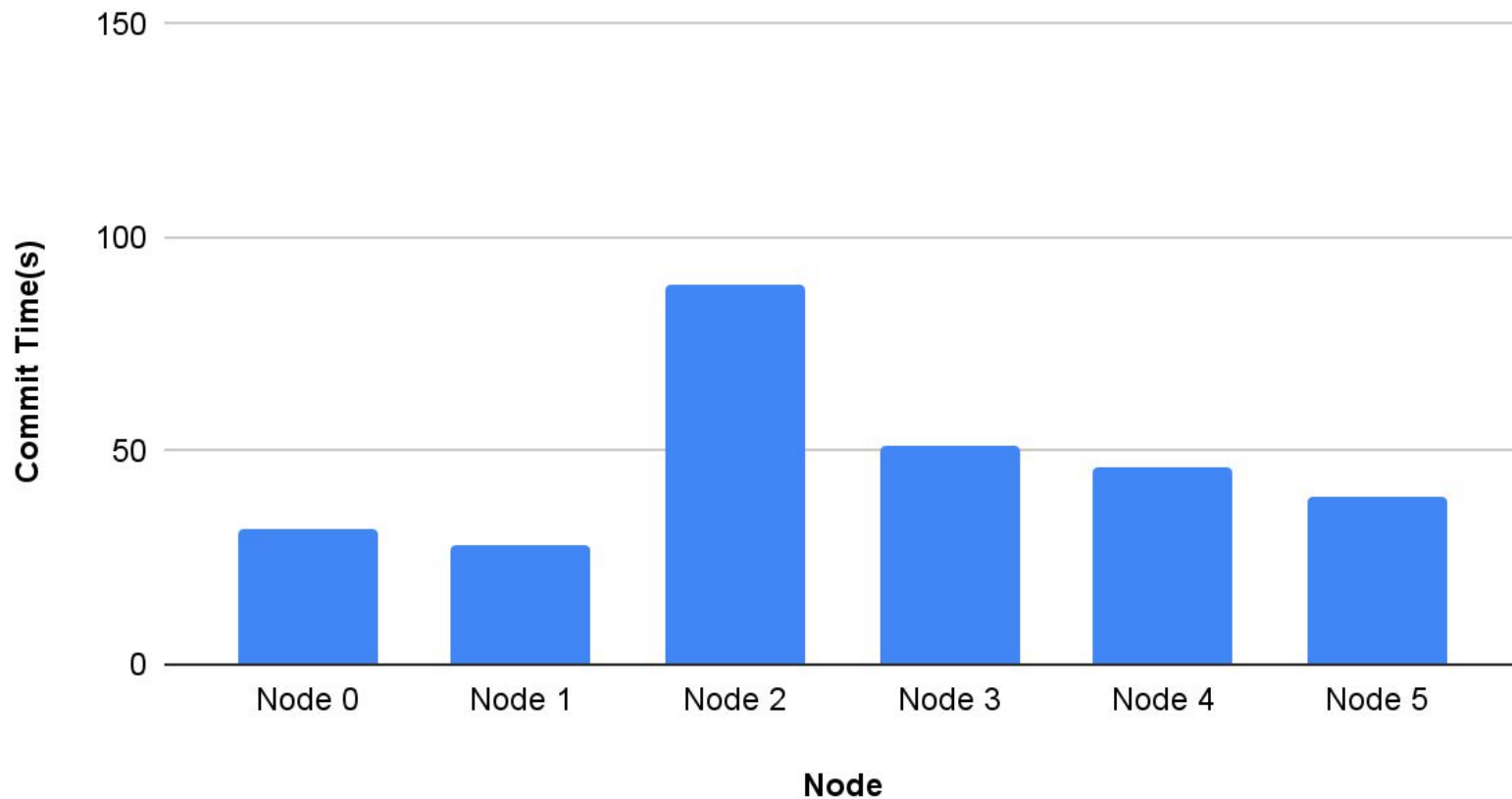
Average Commit Time for 100 concurrent transactions



Average Commit Time for 100 concurrent transactions



Average Commit Time for 150 concurrent transactions



Conclusions from Implementation

- Blockchain consensus is relatively simple to implement compared to something like RAFT. Fewer RPC must be passed through the network. However, PoW and Nakamoto consensus require a significant delay between when a TX is proposed and when it is committed.
 - This may also be an artifact of our implementation choice to have TXs batched locally before being sent into the network. Nodes don't have to agree on the selection of TXs to batch from.
 - This implementation decision may mean TXs get dropped if the node crashes, or could be “stuck” pending for a while.

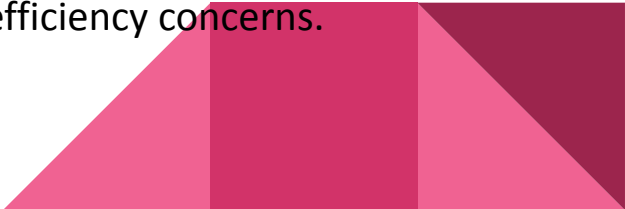


Future Work

- We could improve our blockchain system by having nodes communicate issued TXs rather than batching locally. We chose to focus on TX ordering as our main implementation goal, with less priority given to anonymity and security, the traditional goals of a cryptocurrency scheme. Our design leaves open the opportunity for a malicious node to create and batch illegitimate transactions into the network. This is unverifiable, since there is no shared, agreed upon pool of TXs to select from.
- Transactions are sent transparently through the network. This was so that all nodes can update the wallet values they're tracking and can be verified as equivalent at the end of the simulation.
- These design choices would certainly not work in a real blockchain environment and could stand to be fixed in future work.



Final Note

- Blockchain architecture with Nakamoto consensus and proof-of-work algorithms provides a secure and decentralized approach to transaction validation.
 - We've gained more insights into the novelty of blockchains as a solution to the consensus problem and have a better appreciation for the cleverness of these solutions.
 - These mechanisms have been successfully implemented in various blockchain networks, enabling trust, transparency, and immutability.
 - We now have a greater understanding of how it is blockchain achieves security and establishes transaction order in a decentralized fashion.
 - As blockchain technology continues to evolve, alternative consensus mechanisms and algorithms are being explored to address scalability and energy efficiency concerns.
- 



Thanks