DataSys Coin(DSC) Blockchain

December 2, 2023

Abstract

This paper introduces a centralized simplified blockchain program that demonstrates the fundamental concepts and implementation of blockchain technology. The program simulates block creation and transaction processing, and comprises five main components: a blockchain acts as a ledger that stores the blocks, a metronome that regulates block growth intervals, a pool that receives transactions from wallets, multiple validators that perform crypto mining and provide proof for block creation, and multiple wallets that generate transactions to the pool. This paper provides a beginner's guide to blockchain technology and its essential components, offering insights into the inner workings of a blockchain system. The DSC blockchain program serves as an educational tool, allowing individuals to gain hands-on experience with core blockchain concepts.

1 Introduction

A blockchain is a distributed ledger with a growing list of blocks that are securely linked with cryptographic hashes. Each block contains a cryptographic hash of the previous block and transactions. Together the blocks effectively formed a irreversible chain. [Wikipedia contributors(2023a)]

1.1 Bitcoin

The concept of blockchain was first introduced with the launch of bitcoin. In October 31, 2008, Satoshi Nakamoto released the groundbreaking paper entitled "Bitcoin: A Peer-to-Peer Electronic Cash System" [Nakamoto(2008)]. In this paper, bitcoin is described as a purely peer-to-peer electronic cash system that would allow online payments to be sent directly from one party to another without going through a financial institution. Satoshi Nakamoto introduced a solution to address the double-spending problem [Wikipedia contributors(2023b)] by leveraging a peer-to-peer network and implementing a hash-based proof-of-work mechanism.

Since the launch of Bitcoin, blockchain technology has exponentially gained popularity across various industries, especially in finance. A notable illustration

of Bitcoin's remarkable success is the famous bitcoin pizza story. On May 22, 2010, Laszlo Hanyecz exchanged 10,000 bitcoins (BTC) with Jeremy Sturdivant for two Papa John's pizzas. As of today, the value of a single bitcoin stands at $38,764.

1.2 Motivation

The impact of blockchain technology extends far beyond its initial introduction with bitcoin. As industries recognize the potential of decentralized and transparent systems, blockchain has become a catalyst for innovation and transformation. Over the years, blockchain technology has witnessed a proliferation of innovative applications. Prominent examples such as Ethereum and NFTs have gained significant popularity. Irrespective of their profession or industry, a foundational understanding of blockchain empowers individuals to confidently navigate the future.

While the growth of blockchain applications is undoubtedly promising, the intricacies of blockchain technology pose a significant hurdle for beginners. Understanding the underlying concepts of blockchain, such as consensus mechanisms, cryptographic mechanisms, requires a solid foundation in computer science and cryptography.

To address this challenge, this paper introduces a centralized simplified blockchain program that serves as an educational tool, enabling individuals to gain hands-on experience and a deeper understanding of the fundamental concepts and implementation of blockchain technology.

2 Design and Implementation

2.1 Overview

Figure 1 illustrates the architecture of the DSC blockchain. This paper covers five key components: the *blockchain*, *metronome*, *validator*, *pool*, and *wallet*.

The entire process of the DSC blockchain system involves the following steps:

- The *wallet* sends transactions to the *pool*.
- The *metronome* sends a heartbeat message to *blockchain* periodically.
- The *blockchain* receives the heartbeat message, add a new block to the list of blocks.
- The *blockchain* broadcast the next target hash to *metronome* and all the *validators*
- The *validators* initiate the process of mining a cryptographic nonce capable of generating a hash with matching leading bits to the target hash. They then send the computed nonce to the *metronome* for validation.

- The *metronome* validates the nonce, sends an approval message back to the *validator*.
- Once a *validator* receives approval, it proceeds to request and retrieve unconfirmed transactions from the *pool*. These transactions are then written into a new block, which is subsequently sent to the *blockchain*.

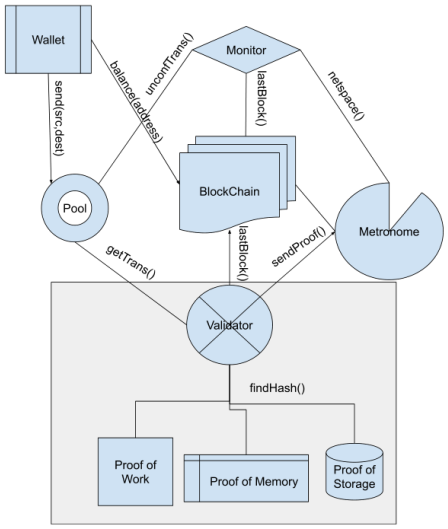


Figure 1: DataSys Coin Blockchain Centralized Architecture

2.2 Component introduction

2.2.1 Blockchain

The *blockchain* component plays a vital role in storing all the blocks and facilitating interaction with other system components. To ensure simplicity, the blocks and wallet balances are stored and managed in memory.

The *blockchain* establishes a connection with the *metronome* and begins waiting for connections from *validators* and the *pool*.

When the *blockchain* receives the block growth heartbeat message from the *metronome*, it appends either a previously received block from a *validator* or an empty block from the *metronome* to the list of blocks. Subsequently, the hash of the last block is broadcasted as the next target hash to both the *metronome* and all the *validators*.

2.2.2 Metronome

The metronome component fulfills three responsibilities: maintaining the block growth interval by periodically sending heartbeat messages to the *blockchain*,

adjusting the difficulty level dynamically for *validators*, and validating the nonce received from *validators* and issuing block creation approval to the winning *validator*.

Block growth heartbeat Once the *blockchain* is connected, the *metronome* periodically sends heartbeat messages to the *blockchain*. In the event that no *validator* discovers the nonce, the heartbeat message will include an empty new block to be inserted into the *blockchain*.

Dynamic difficulty If, within a single heartbeat time window, fewer than half of the *validators* discover the nonce, the *metronome* decreases the current difficulty level by one. Conversely, if every *validator* finds a nonce, the *metronome* increases the current difficulty level by one. Subsequently, the *metronome* broadcasts the updated difficulty level to all *validators*.

Validating the nonce Upon receiving a nonce, the *metronome* also obtains the fingerprint and public key of the corresponding *validator*. It utilizes these information to generate a result hash. If the leading bytes of this result hash, equivalent in size to the difficulty level, match the target hash broadcasted by the *blockchain*, the nonce is considered valid. Conversely, if the leading bytes do not align, the nonce is deemed invalid.

2.2.3 Validator

The primary role of the *validator* is to discover the cryptographic nonce for a given target hash. Various approaches exist for finding the nonce, and in this paper, we will delve into two specific strategies: proof of work and proof of memory.

Algorithm 1 Proof of work algorithm

$nonce \leftarrow 0$
$foundNonce \leftarrow False$
while $currentTime - startTime \leq INTERVAL$ **do**
  $keyBytes \leftarrow (fingerprint + publicKey + toBytes(nonce))$
  $newHash \leftarrow blake3(keyBytes)$
  if $CompareFirstNBits(newHash, targetHash, difficulty)$ **then**
    $foundNonce \leftarrow True$
    break
  end if
  $nonce \mathrel{+}= 1$
end while
if $foundNonce$ **then**
  $SendNonceToMetronome(fingerprint, publicKey, nonce)$
end if

Proof of work The proof-of-work algorithm pseudocode (Algorithm 1) demonstrates that the *validator* iteratively increments the nonce and endeavors to produce a hash that matches the first N bits of the target hash. Once the nonce is found, the *validator* transmits it, along with the fingerprint and public key, to the *metronome*.

Algorithm 2 Proof of memory: generating hashes

$hashPairNum \leftarrow \lfloor hashMemSize/(hashSize + intSize) \rfloor$
$hashList \leftarrow NewList$
for $nonce \leftarrow 1$ to $hashPairNum$ **do**
    $keyBytes \leftarrow (fingerprint + publicKey + toBytes(nonce))$
    $newHash \leftarrow blake3(keyBytes)$
    $hashList.append((newHash, nonce))$
end for
$SortByHashBits(hashList)$

Algorithm 3 Proof of memory: finding the nonce

$validateBits \leftarrow GetFirstNBits(targetHash)$
$resultIdx = BinarySearchHash(validateBits)$
if $resultIdx \neq -1$ **then**
    $nonce = hashList[resultIdx][1]$
    $SendNonceToMetronome(fingerprint, publicKey, nonce)$
end if

Proof of memory The Proof of Memory strategy consists of two distinct parts. The first part, described in Algorithm 2, involves generating hashes in memory. This step is executed only once during the initialization of the validator.

The second part, described in Algorithm 3, focuses on finding the nonce. When the *validator* receives the target hash from the *blockchain*, it performs a binary search to locate a hash in memory that aligns with the first N bits of the target hash. If a matching hash is successfully found, the corresponding nonce is then sent to the *metronome*.

2.2.4 Pool

The *pool* is responsible for receiving and storing transactions from *wallets*, as well as sending these transactions to the *validator* upon request. Whenever a *validator* receives approval from the *metronome*, it sends a request to the *pool* to obtain transactions. These transactions are then written to a new block, which is subsequently sent to the *blockchain*. In the DSC blockchain system, the block size is restricted to 1MB. Considering that each transaction occupies 128 bytes, the *validator* can accommodate 8,090 transactions per block.

2.2.5 Wallet

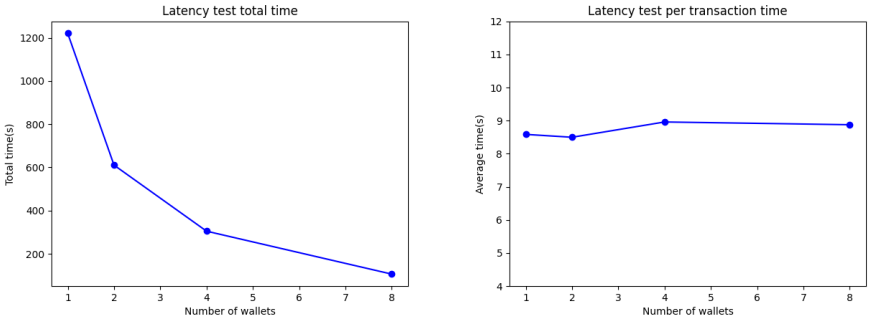
A *wallet* serves as a user interface, the following operations are provided by the *wallet*:

- *wallet create*: creates a private key and public key pair.
- *wallet balance*: query the balance of a *wallet*.
- *wallet send value public_key*: create a transaction.
- *wallet transaction transaction_id*: query the status of a transaction.
- *wallet transactions*: query all the transactions related to current *wallet*.

3 Evaluation

3.1 Latency test

I performed a strong scaling latency test to analyze the latency of sequentially sending a total of 128 transactions in four different scenarios, considering varying numbers of *wallets*: 1, 2, 4, and 8. The *wallet* waits for confirmation from the *blockchain* before sending the next transaction. The heartbeat interval in the *metronome* is configured to 6 seconds.



(a) Total time to finish all transactions (b) Average time to finish one transaction

Figure 2: Latency test

From Figure 2a, we can see that reducing the number of transactions sent per *wallet* leads to a decrease in the overall processing time.

Based on the results of Figure 2b, it is evident that the latency remains unaffected by the number of *wallets*.

The latency of the system is exclusively determined by the heartbeat interval of the *metronome*. These results align precisely with my initial expectations. On average, a transaction requires approximately 1.5 times the duration of the

heartbeat interval to be confirmed. When the transaction pool is not overwhelmed by a large influx of transactions and the *validators* are functioning as intended, a transaction can be reliably confirmed by the *blockchain* within the designated heartbeat time window. Consequently, the latency of an individual transaction within the blockchain system is predominantly influenced by the heartbeat interval established by the *metronome*.

3.2 Throughput test

I performed a strong scaling throughput test to evaluate the throughput of sequentially sending a total of 128,000 transactions in four scenarios, with different numbers of *wallets*: 1, 2, 4, and 8. In this throughput test, unlike the latency test, I did not wait for the confirmation of individual transactions. Instead, I submitted all transactions simultaneously and waited for the confirmation of the entire batch.

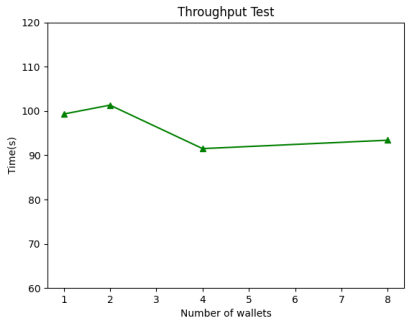


Figure 3: Sample Image

Based on the findings depicted in Figure 3, it is evident that the throughput remains unaffected by the number of *wallets*. The primary factor influencing the throughput is the maximum number of transactions the *validator* can process within a heartbeat time window, which is constrained to 8,090 transactions.

The reason why looking up transaction state does not overwhelm the system is due to the current implementation of the DSC *blockchain* component, where all transaction information is stored in a map. This design choice ensures that the time complexity of querying a transaction remains constant. However, it should be noted that in a real-world blockchain, this approach would not be feasible. In such cases, transaction information would need to be extracted by traversing the blocks, which can significantly slow down the process.

4 Conclusions

In conclusion, this paper has presented a centralized simplified blockchain program designed as an educational tool to facilitate the comprehension and practi-

cal exploration of blockchain technology. Individuals, even those without extensive computer science or cryptography backgrounds, can gain hands-on experience and a deeper understanding of the underlying concepts and implementation of blockchain.

Throughout this paper, I have provided in-depth explanations of the key components and functionalities of the blockchain system. By carefully dissecting various aspects, ranging from transaction handling to block creation and validation, I have endeavored to demystify the intricacies surrounding blockchain technology, ultimately making it more approachable for beginners.

Moving forward, we can continue refining and expanding this educational tool to encompass more advanced concepts and real-world scenarios. By incorporating additional features and exploring the integration of decentralized elements, future iterations of this program can provide an even more comprehensive learning experience.

References

[Nakamoto(2008)] Satoshi Nakamoto. 2008. Bitcoin whitepaper. *URL: https://bitcoin. org/bitcoin. pdf-(: 17.07. 2019)* (2008).

[Wikipedia contributors(2023a)] Wikipedia contributors. 2023a. Blockchain — Wikipedia, The Free Encyclopedia. `https://en.wikipedia.org/w/index.php?title=Blockchain&oldid=1187753169`. [Online; accessed 1-December-2023].

[Wikipedia contributors(2023b)] Wikipedia contributors. 2023b. Double-spending — Wikipedia, The Free Encyclopedia. `https://en.wikipedia.org/w/index.php?title=Double-spending&oldid=1178219920`. [Online; accessed 1-December-2023].