Int. j. inf. tecnol. (April 2024) 16(4):2379–2394
https://doi.org/10.1007/s41870-023-01685-x

ORIGINAL RESEARCH

A novel approach to optimizing transaction processing rate and space requirement of blockchain via off-chain architecture

Saha Reno[1] · Sadia Hossain Priya[1] · G. M. Abdullah Al-Kafi[1] · Sheikh Tasfia[2] · Marzia Khan Turna[3]

Received: 19 July 2023 / Accepted: 4 December 2023 / Published online: 6 January 2024

Abstract In the realm of blockchain technology, public ledgers such as Bitcoin face significant challenges in terms of throughput enhancement and addressing the massive space requirements. This paper presents a novel approach to tackle these issues by employing a decentralized peer-to-peer (P2P) file distribution and a newly introduced strategy known as the Twin-Ledger architecture. By utilizing the P2P-based file management and minimal storage needs of the content reference, along with the proposed Twin-Ledger framework described in this research, we efficiently solve the scalability problems. Our system allows for the integration of any consensus mechanism, but we specifically utilize the Proof-of-Work consensus to preserve both scalability and decentralization without compromising security. This innovative approach paves the way for broader adoption of blockchain technology across various industries, as it efficiently tackles the throughput and storage challenges associated with public blockchains. In our experimental results, we demonstrate that our system can store nearly 22,000 transactions in a 3.4 MB-sized block, achieving a TPS of 32 on average, which can be significantly increased by extending the original block size without imposing any storage burden.

Keywords Blockchain · Scalability · IPFS · Off-Chain · Twin-Ledger

✉ Saha Reno
reno.saha39@gmail.com

Sadia Hossain Priya
sadiahossainpriya126@gmail.com

G. M. Abdullah Al-Kafi
kaficsebaiust02@gmail.com

Sheikh Tasfia
sheikhtasfia0602@gmail.com

Marzia Khan Turna
marziakhanturna@gmail.com

[1] Department of Computer Science and Engineering, Bangladesh Army International University of Science and Technology (BAIUST), Cumilla, Bangladesh

[2] Department of Computer Science and Engineering, Northern University Bangladesh (NUB), Dhaka, Bangladesh

[3] Department of Computer Science and Engineering, Indiana University Perdue University Indianapolis (IUPUI), Indianapolis, IN, USA

1 Introduction

Blockchain is a decentralized, distributed digital ledger used to record assets or transactions in an irrevocable manner. It is designed for real-world applications where conventional systems fail to provide quick security, as transactions are cryptographically secured and robust security is supplied by its powerful consensus protocol. Compared to existing methods, blockchain promises higher efficiency, accuracy, and safety for governmental, public, and social services. Blockchain can be divided into three generations depending on its applicability and mode of use: Blockchain 1.0, 2.0, and 3.0 [19]. The first generation, including cryptocurrencies such as Bitcoin and Litecoin, is the most well-known and widespread [14]. The second generation uses smart contracts in various decentralized applications (DApps), including crowdfunding, autonomous organizations, land management, philanthropy, and more [17]. Third-generation applications involve the interaction of blockchain with the physical world, using the Internet of Things (IoT) for deployment in public sectors and industry [18].

To harness the potential of blockchain for second and third-generation applications, throughput enhancement and addressing massive storage requirements are crucial. Scalability, in

the context of distributed networks like blockchain, refers to the ability to handle a large number of transactions in a short period [11]. To increase the throughput of any blockchain-based system, more transactions must be stored in each block, requiring an increase in block size [10]. However, this causes storage bloating and slows down the processing rate over time as it takes longer to broadcast a specific block on the network for verification and storage [21]. Furthermore, the blockchain's total size grows to the point where a newly joined node struggles to replicate the entire ledger into its system [22]. For instance, the size of the Bitcoin blockchain, at the end of 2021, was about 386 GB, and any user interested in joining this network must download the entire ledger, which took nearly 240 h on average [6].

In this paper, we propose an off-chain solution to enhance throughput and address storage requirements in public blockchains by utilizing the InterPlanetary File System (IPFS), a distributed, peer-to-peer network-based data sharing and storage service. The key contributions of our work are as follows:

- All validated transactions are stored in IPFS (off-chain), and a Content Identifier (CID) of only 46 Bytes is generated. Since this CID is much smaller in size than the actual transactions, storing CIDs in the ledger (on-chain) instead of the raw transactions results in a significantly higher number of transactions per block.
- We introduce a new Twin-ledger strategy that maintains two distinct ledgers, both off-chain and on-chain. The Actual Block, which contains all raw transactions, is stored off-chain, and its size varies according to the number of transactions stored within it. After uploading this Actual Block to IPFS and receiving a CID, a new block called final block is generated, with a size of approximately 290 KB. The final block, which is much smaller than the Actual Block, only stores the 46-Byte CID of the Actual Block. The network holds and distributes this final block, eliminating the storage bloating issue.
- To ensure security without compromising throughput and storage capabilities, we develop our system using a public and permissionless blockchain. A large number of nodes, including miners, contribute to the decentralization and security features of our system. We employ a combination of Proof-of-Work and Nakamoto Consensus Rules to increase the network's security against the concerns to which private blockchains are prone, even though our system supports any consensus technique.

2 Literature review

Various strategies have been employed by researchers to address the issues of scalability, security, and decentralization in blockchain technology. These include techniques such as sharding, parallel processing, off-chain solutions, hardware-supported mechanisms, micro-payment channels, and novel consensus approaches. As a result, blockchain protocols can be classified into the aforementioned categories. While these methods strive to strike a balance between throughput, storage bloat, security, and decentralization, each approach comes with its constraints in effectively addressing all these challenges simultaneously. Table 1 delivers a comprehensive review of established blockchain systems, emphasizing their contributions, outcomes, and associated limitations.

In summary, the present work diverges significantly from the aforementioned literature by addressing two cardinal challenges in public blockchain systems: throughput and storage. Unlike Payment Channel Networks, Micropayment Channels and other traditional systems reviewed in this section which largely concentrate on increasing transaction speed but offer little in the way of reducing storage requirements, our proposed model enhances the throughput and substantially reduces on-chain data storage needs through the use of a lightweight 46-Byte Content Identifier (CID) at the same time. It is also noticeable that most of the reviewed works fail to deliver a robust security due to the enhancement of scalability. As IPFS works in a similar approach like blockchain and torrent, where the transactions remain secured due to its decentralized nature, and the utilization of proof-of-work does not hamper the throughput advancement, our system also guarantees a strong defense against adversarial attacks; which is not achievable in the existing works discussed above. Furthermore, the architecture is constructed on a public, permissionless blockchain, thus providing a high degree of decentralization and security. Notably, our system is designed to work with any consensus method, making it flexible for future changes. Therefore, it is inevitable that our approach offers a holistic solution to the limitations commonly cited in existing blockchain technologies and contributes a significant advancement to the field.

3 System overview

In this section, we briefly introduce our system before diving into a detailed explanation for each of the sub-steps. To make it easier to understand, we present the proposed system's overview with a focus on its impact on enhancing throughput and minimizing space requirements.

3.1 Utilizing CID for enhancing throughput

In cryptocurrencies like Bitcoin, blocks composed of validated transactions are approximately 1 MB in size. The number of transactions that can be added to a block is limited by

Table 1 Reviews of existing approaches to enhance blockchain scalability

Authors and Their Works	Contribution and Methodology	Results	Limitations
Poon and Dryja [15]	Payment Channel Networks (PCN) use Micropayment Channels to enhance throughput and address storage limitations	These channels enable numerous transactions within a specific time frame without connecting to the Bitcoin network ledger	Funds may be trapped indefinitely or for the channel's duration
Malavolta et al. [13]	Introduced a new cryptographic protocol called Anonymous Multi-Hop Lock (AMHL) to protect PCNs against Wormhole Attacks	AMHL requires an additional communication round and reduces transaction size and storage requirements	Does not provide significant metrics from current PCNs to evaluate the running time and communication overhead required by each role in a multi-hop lock protocol
Luu et al. [12]	Elastico partitions the network of miners into multiple committees, allowing the transaction rate to scale linearly with the computational capacity for mining	This results in increased blocks per epoch as the number of nodes grows, though epoch time also increases	The protocol must tolerate variable rates of identity creation and inconsistency in views of committee members due to both Byzantine failures and network delays in real networks
Kokoris-Kogias et al. [8]	Omniledger leverages two proof-of-stake-based blockchain solutions to enhance performance by processing transactions in parallel using an intra-shard approach	Achieves a throughput of 13000 TPS with specific parameters	The evaluation of the experimental prototype is limited to demonstrating the scalability and throughput of OmniLedger, without providing a comprehensive analysis of its performance under various network conditions or attack scenarios,
Zamani et al. [20]	Rapidchain addresses some of the limitations of other sharding-based blockchains by implementing the Cuckoo rule, which can withstand Byzantine faults up to 1/3 of participants	It achieves high throughput and low confirmation latency through block pipelining, gossiping protocol, and intra-committee consensus	Existing sharding-based blockchain protocols still require a linear amount of communication per transaction, which introduces a major bottleneck to the throughput and latency of these protocols
Dang et al. [1]	In this paper, the authors enhance the Byzantine Fault Tolerant Consensus to increase the throughput of each shard. The system relies on the Trusted Execution Environment (TEE) provided by Intel SGX hardware	Eliminate the deceptions of the Byzantine Failure model and improve fault tolerance	(i) Improving Byzantine Fault Tolerant (BFT) protocols alone does not address the scalability issue due to the quadratic communication cost (ii) Existing solutions for shard formation in blockchain systems are expensive and result in large shards
Kuzmanovic [9]	BDN is a cloud-based distribution network which leverages the cloud for distributing transactions and using indexes rather than original transactions when transmitting blocks across the network	BDN, achieving a throughput of 1000 transactions per second	The paper does not provide a detailed explanation of how trust can be established in the underlying networking infrastructure to achieve a provably neutral network design
He et al. [5]	Chameleon is a BDN-based permissioned blockchain protocol that adopts a non-forking principle, enabling independent transaction processing and cooperation between regions	Consensus nodes store a single epoch of blocks, with earlier epochs stored in the cloud, reducing storage needs. Achieves an average of 600 transactions per second	It remains a Proof-of-Concept with unaddressed challenges, such as handling overload situations
Hazari and Mahmoud [4]	Parallel processing involves executing consensus, mining, or smart contract operations concurrently to enhance scalability and transaction speed	Shows an improvement in the scalability of Proof of Work up to 34%	A single point of failure may occur if the block manager fails or becomes disconnected, and the fate of a specific nonce range when its miner becomes unavailable is not addressed
Ehmke et al. [2]	Proof-of-Property improves blockchain scalability by incorporating the system's state in the most recent block and including the relevant state portion in new transactions	This reduces storage requirements and verification complexity	It requires a significant number of nodes for path validation, and forks can occur

Table 1 (continued)

Authors and Their Works	Contribution and Methodology	Results	Limitations
Fan and Chai [3]	Roll-DPoS enhances Delegated Proof-of-Stake for IoT-based applications, addressing the challenges of large-scale IoT device deployment and substantial data volume	It allows more blockchain nodes to produce blocks and earn rewards compared to classical DPoS	The paper does not discuss the potential security vulnerabilities or attack vectors that could be exploited in the Roll-DPoS consensus algorithm
Kogias et al. [7]	ByzCoin optimizes transaction commitment and verification through communication trees and collective signing	It ensures Microblock commitment irreversibility, preventing double-spending attacks	It is vulnerable to DoS attacks and slowdowns, and its security doesn't outperform Proof-of-Work. ByzCoin's security breaks at 30% attacker control
Teutsch and Reitwießner [16]	TrueBit manages a decentralized group of miners and applications through a smart contract	It reduces Ethereum's computational overload, uses the Verification Game for accuracy, simplifies and scales mining	Performs poorly in Big Data Applications due to the decline in performance when handling large-scale computations

the fixed and finite amount of space per block. Regrettably, merely increasing the block size is not a suitable solution for enhancing the throughput of a blockchain system. The proposed method allows for the inclusion of more transactions per block without expanding the block size. Users first complete a transaction and upload it to IPFS, which returns a CID of around 46 Bytes in length. As depicted in Fig. 1, users then send this CID to a nearby miner who retrieves the actual transaction from IPFS using the CID. If the transaction is successfully validated, the miner adds the CID to their mempool and shares it with other miners. Invalid transactions are discarded. An original block is created using the CIDs of all validated transactions after a certain number of transactions have been validated. The size of this original block can be adjusted according to system requirements, but in our approach, we maintain it at the same size as Bitcoin's block size, approximately 1 MB. By storing CIDs in the original block instead of actual transactions, a massive number of transactions can be included. Consequently, the number of transactions per block and ultimately the throughput is increased without enlarging the block size. Figure 1 illustrates the overall process of decreasing the transaction size using IPFS and distribute it to the nearby miners.

3.2 Twin-ledger mechanism to reduce space requirement

Bitcoin's 1 MB-sized blocks are distributed among all network nodes. The total size of this ledger has exceeded 473 GB and continues to grow as around 156 blocks are generated daily. Consequently, storage bloating becomes a concern, as new users must replicate these 473 GB of data in their systems. In our suggested system, however, users are not required to store the 1 MB original blocks. Instead, miners upload the original block to IPFS, which generates a corresponding CID for the block. A new block, called the final block, is created using only the CID of the original block following the PoW consensus. The smaller size of the final block allows for faster distribution among miners, consuming less bandwidth. The term "Twin-ledger" refers to the presence of two ledgers in our proposed system: one for original blocks and one for final blocks. Miners can validate transactions by accessing the original block ledger via the Actual blockchain and IPFS. This approach achieves significant throughput improvements without increasing the block size. The overall working procedure of our system is depicted using Fig. 2.

4 Methodology

In this section, we explain how our proposed blockchain uses two different ledgers and IPFS to enhance overall scalability.

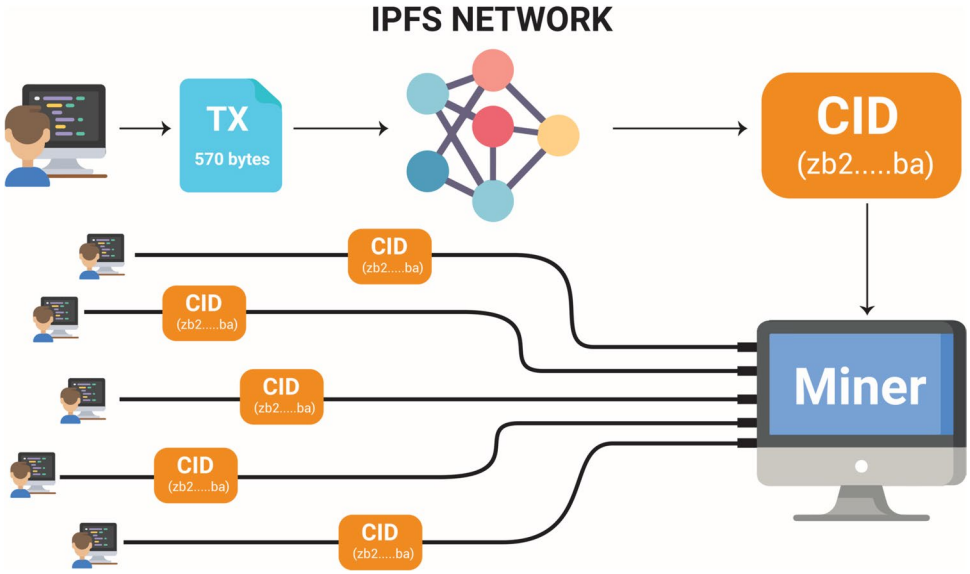


Fig. 1 Obtaining CID-Formatted Transactions from Clients via IPFS at the Mining Endpoint

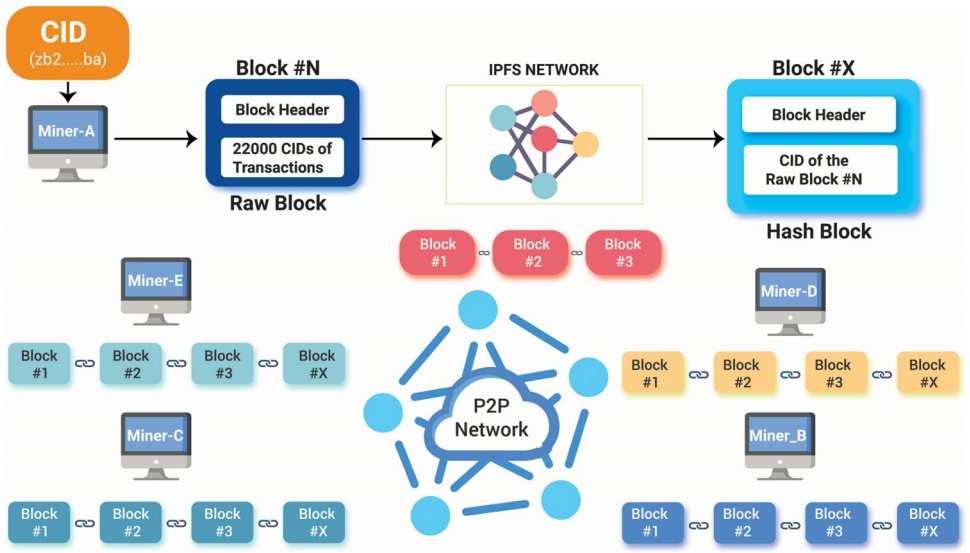


Fig. 2 Schematic Representation of the Suggested System's Process Flow

We begin by detailing the transaction generation process, its interaction with IPFS, and conclude with the creation of two distinct block types. Each of these steps is elaborately outlined in the subsequent subsections. Furthermore, we discuss the transaction verification protocols, followed by an in-depth examination of the verification processes for both the original block and the final block.

4.1 Transaction generation

Similar to most blockchain-based systems, transactions in our system consist of two components: (i) Message and (ii) Digital Signature. The message includes information about the sender and recipient as well as the recipient's public key. Sender information encompasses the sender's address and block number, while recipient information comprises the amount of the asset they are entitled to receive and their address. The block number identifies the block that serves as proof of the sender's ownership of the asset they wish to transfer to the recipient's address. The block number is provided to avoid accessing the entire blockchain, thus implicitly improving the system's throughput. The entire message is encrypted with the sender's public key, generating the transaction's digital signature, which is used to verify the transaction's legitimacy.

4.2 Uploading transactions to IPFS

After creating a transaction, the user uploads it to IPFS. The Merkle Directed Acyclic Graph is utilized to divide

the transaction into multiple 256 KB segments. These segments are then distributed to other network users via IPFS. A distributed hash table maintains data retrieval information such as the peer's location holding a specific segment and the routes to each peer (DHT). A SHA-256 hash of each segment's contents serves as a unique identifier for each one. A Content Identifier (CID) is created by feeding all the segments of a specific transaction into a SHA-256 hashing algorithm. Anyone can retrieve transactions from IPFS using this CID. The checksum verifies every part of a transaction, allowing easy detection of any instance of transaction tampering. Furthermore, IPFS prevents and discards the storage of duplicate transaction copies. After receiving it, the user provides the CID of the transaction to the miner for validation. If successful, the transaction is recorded in the ledger.

4.3 Transfer of transactions to nearby miners using peer-to-peer network

A peer-to-peer network is essential for a blockchain-based system because it does not rely on a central server, and each node functions as both a client and server. Utilizing a P2P network offers numerous advantages, such as improved performance, faster file-sharing speed, cost-effectiveness, and scalability since system resources are distributed among the decentralized network's peers, eliminating the need for a single point of failure. To verify transactions, CIDs obtained from IPFS must be sent to nearby miners. The transfer of these CIDs occurs through a P2P network, ensuring the aforementioned benefits.

4.4 Retrieving transactions from IPFS

After receiving CIDs from users through the P2P network, miners fetch the actual transactions from IPFS. The miner's IPFS node employs the Merkle DAG to locate all the 256 KB-sized segments that constitute the desired transaction to be fetched for a single CID. The miner then consults the DHT to identify the peer nodes that store those specific segments. The miner obtains routing information to connect to the IPFS nodes holding the desired segments. Finally, IPFS assembles all the segments to reconstruct the original transaction, which miners use for validation. Since tampering with transactions in IPFS entirely alters the corresponding CID, it is guaranteed that tampering with transactional data is infeasible.

4.5 Verification of transactions

To ensure the validity and integrity of transactions within our system, each transaction undergoes four distinct verification checks. A transaction is deemed invalid if it fails any of these checks. The essential checks are as follows:

- *Identity Verification Check:* This check establishes the sender's authenticity by validating their identity using the sender's public key, transactional data, and digital signature. All transactions are encrypted with the private key, which generates a digital signature.
- *Block Verification and Address Matching:* During a transaction, the sender is required to provide a block number when transferring assets to an address. This block number contains the transaction that serves as proof of the sender's ownership of the assets being transferred. The sender acted as the recipient in the specific block's transaction while claiming the assets. To pass this check, the recipient address of the previously validated transaction must match the sender's address in the transaction being verified. If the addresses match, the transaction proceeds to the next verification step.
- *Asset Quantity Verification:* This check prevents the sender from transferring more assets than they possess. The Authentication Check supplies the block number, which serves as proof of the sender's ownership of the assets. Using this number, the miner verifies if the quantity of assets in that specific block is greater than, equal to, or less than the amount the sender intends to transfer. The transaction will pass this check if the sender has enough assets to complete the transfer to the intended address.
- *Detecting and Preventing Double Spending Attack:* This check ensures that no user exploits an asset more than once. Our system maintains a database containing only the addresses of individuals with a specific amount of assets. Before transferring assets to another user, the sender's address must be listed in the database as the recipient, since the assets were previously verified as belonging to the recipient. Suppose the sender's address is found in the database and passes all other verification checks. In this case, the transaction is considered genuine, and the sender's address is removed from the database. Simultaneously, the recipient address of the newly verified transaction is added to the database. The database is updated each time a transaction is confirmed. It is worth noting that a sender may use multiple receiving addresses they own when transferring assets in a single transaction, as a single receiving address may not have sufficient assets to transfer. If a user uses several receiving addresses, one of which is not present in the database, all legitimate receiving addresses used in the transaction being checked will be removed from the database as a penalty. This check ensures that an address that has been used to send a specific quantity of assets cannot be reused for subsequent transactions. This approach prevents double-spending in our system, as illustrated in Fig. 3.

Algorithm 1 Identity Verification Check + Verifying Blocks + Amount Checking

```
counter ← 0
messageHash ← sha3_256Hash(ipfsRawTransactionMessage)
validation ← verify(ipfsRawTransactionPublicKey, messageHash,
ipfsRawTransactionSignature, generator_secp256k1)
if (validation = true) then
    counter ← counter + 1
end if
totalSenderAmount ← 0
for each senderAddr in ipfsRawTransactionSenderAddressList do
    blockNumber ← senderAddr[−1]
    senderAddress ← senderAddr[0]
    ultimateBlock ← os.path.join(ultimateBlockDirectory,(blockNumber+
    ".json"))
    rawBlockContent ← subprocess.check_output(f"ipfs cat
    ultimateBlockContent["CID"]", shell=true, text=true)
    for each transaction in rawBlockContent["Transactions"] do
        transactionReceiverList ← transaction["Receiver Address"]
        for each receiverAddr in transactionReceiverList do
            if (receiverAddr = senderAddress) then
                counter ← counter + 1
                rawTransactionContent ← subprocess.check_output(f"ipfs cat receiverAddr["CID"]",
shell=true, text=true)
                rawTransactionReceiverList ←
rawTransactionContent["Message"]["Receiver Address"]
                for each recipientAddr in rawTransactionReceiverList do
                    for each recipientAddress m in recipientAddr.keys() do
                        if (recipientAddress = senderAddress) then
                            totalSenderAmount ←
                            totalSenderAmount + recipientAddr[recipientAddress]
                        end if
                    end for
                end for
            end if
        end for
    end for
    receiverAddressTotalAmount ← 0
    for each recipientAddr in ipfsRawTransactionReceiverAddressList do
        for key, val in recipientAddr.items() do
            receiverAddressTotalAmount ← receiverAddressTotalAmount + value
        end for
    end for
    if (totalSenderAmount ≥ receiverAddressTotalAmount) then
        counter ← counter + 1
    end if
end for
```

Algorithm 2 Checking and Defending Double-Spending Attack

```
validatedSenderList ← []
for each senderAddr in ipfsRawTransactionSenderAddresses do
    senderAddress ← senderAddr[0]
    dbConnection ← sqlite3.connect(databasePath)
    cursor ← dbConnection.cursor()
    cursor.execute("SELECT * from table_1")
    retrievedRows ← cursor.fetchall()
    recipientList ← []
    for each retrievedRow in retrievedRows do
        for each tupleItem in retrievedRow do
            recipientList.append(tupleItem)
        end for
    end for
    if (retrievedRow IN recipientList) then
        validatedSenderList.append(retrievedRow)
        recipientList.remove(retrievedRow)
        cursor.executemany("DELETE from table_1 where recipient_address = ?",
        retrievedRow)
        dbConnection.commit()
    end if
end for
if (validatedSenderList = ipfsRawTransactionSenderAddresses) then
    counter ← counter + 1
    for each recipientAddr in ipfsRawTransactionRecipientAddresses do
        for key, value of recipientAddr do
            recipientAddress ← key
            cursor.executemany("INSERT into table_1 VALUES (?)", recipientAddress)
        end for
    end for
end if
if (counter = 4) then
    mempoolFilePath ← os.path.join (mempoolDirectory,
    serialNumberOfRawTransactionFromIpfs)
    with open(mempoolFilePath, "w") as outputFile:
    outputFile.write(receivedTransactionFromP2P)
end if
```

When a transaction passes all the checks mentioned above, its CID is added to the mempool, which stores confirmed but unconfirmed and pending transactions. Algorithm 1 presents the pseudocodes for verification and validation of identity, blocks, and asset amount, while Algorithm 2 shows the pseudocode for identifying and defending double-spending attack.

4.6 Creation of original blocks

After validating and confirming the transactions from the mempool, miners assemble a block containing the CIDs of all the approved and confirmed raw transactions. This block is referred to as an original block, and the ledger containing original blocks is called IPFS blockchain. The IPFS blockchain is stored on IPFS instead of the miner's computer. Our approach utilizes Proof-of-Work (PoW) consensus for generating original blocks since this consensus mechanism provides strong security and decentralization. The process of original block creation is illustrated in Fig. 4. Due to PoW's limitations, including high energy consumption, slow processing, and high costs, our system can accommodate any consensus mechanism. As a result, our proposed system allows for the adoption of any consensus technique depending on the application's needs and requirements.

4.7 Creation of final blocks

The miner responsible for generating the original block is the sole authorized party to execute the following procedure.

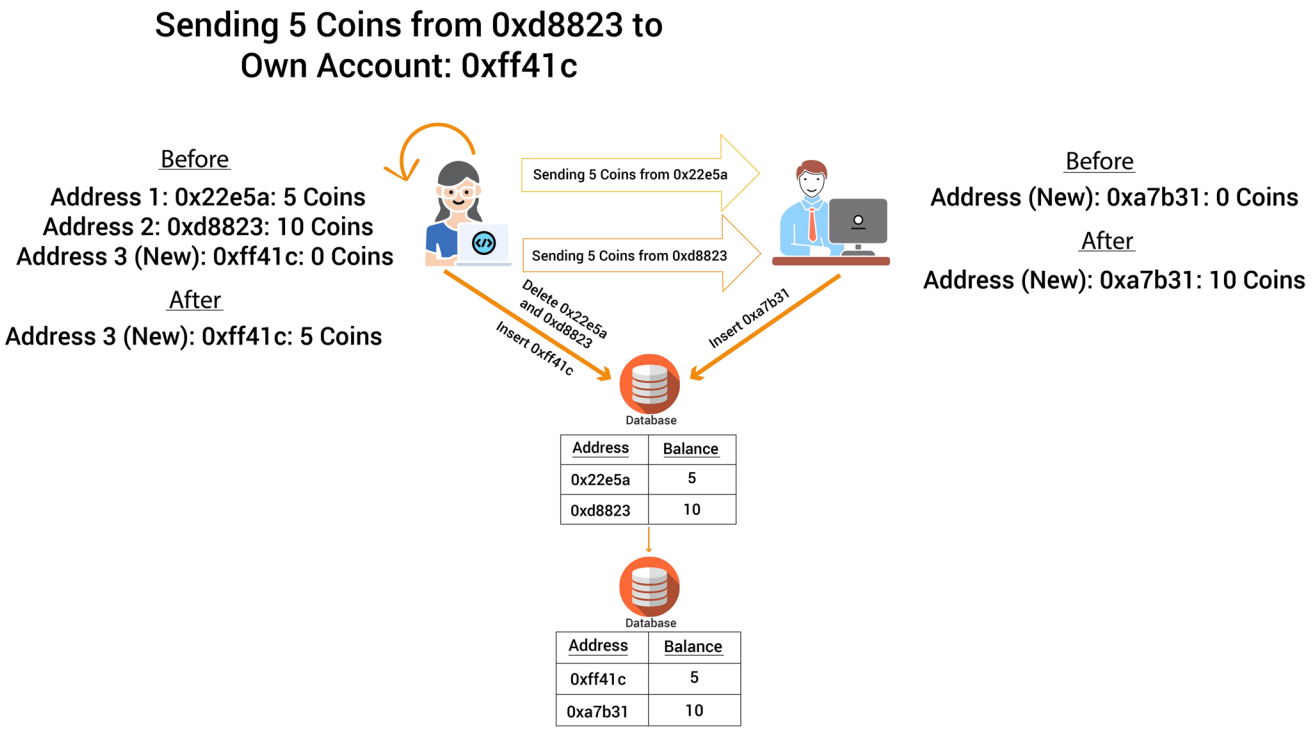


Fig. 3 Employing Distributed Database Solutions to Defend Double Spending Attack

First, the miner uploads the original block to IPFS, which generates a unique.

CID for it. Then, using PoW, the miner creates a block containing only the CID of the original block, known as a final block. This final block is distributed to nearby miners through a P2P network, and all miners store it locally, maintaining a new ledger called the Actual blockchain. Figure 5 visually depicts the entire process.

4.8 Validation of the final blocks

Upon receiving the final block, each miner determines whether to incorporate it into their local Actual blockchain based on the Reverse Block Verification test results. This verification consists of four checks: Index Check, Current Final Block Check, Previous Block Hash Check, and Verified Transaction Check. First, the miner verifies that the current block index is greater than the previous one. Then, the hash of the block is examined to ensure it has a specified number of leading zeros. Next, the block hash is calculated to verify if it matches the previously computed hash included in the block header. The previous block's hash is also checked to confirm if it matches the value of the 'Previous Hash' attribute in the current block's header. If the final block passes the first three tests, the miner retrieves the corresponding original block from IPFS using the CID stored within the final block. The original block undergoes the same tests as the final block, along with an additional

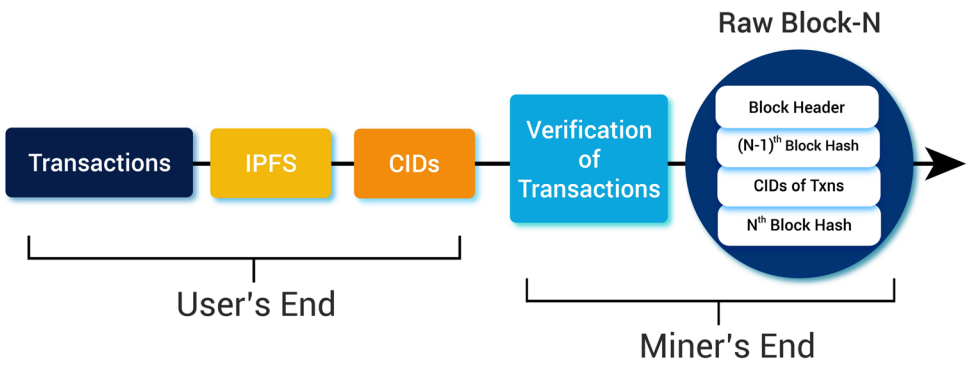


Fig. 4 Creation of Original blocks Utilizing Transaction CIDs

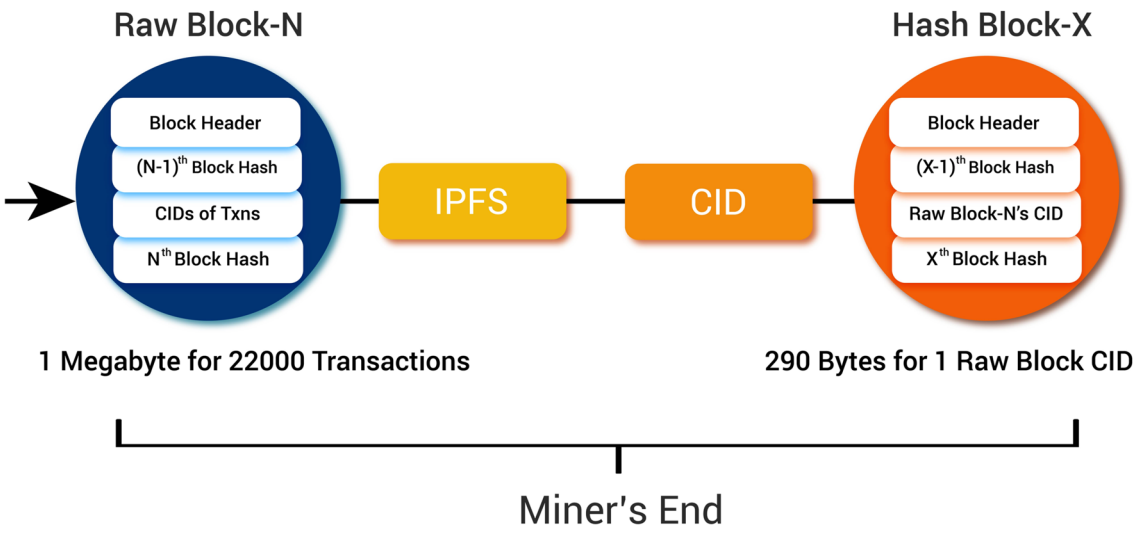


Fig. 5 Deriving Final Block by Leveraging the Original Block's CID

test called Verified Transaction Check, which ensures that all CIDs of transactions within this original block are present in the verifier's mempool. These tests are performed sequentially, and Algorithm 3 presents the reverse block verification pseudocode for the final block.

If the block fails any of the verification tests, it is discarded and not added to the ledger. If it passes, the CIDs of the transactions are removed from the miner's mempool, and the verified final block is incorporated into the miner's local Actual blockchain.

5 Result analysis

Our proposed system enhances scalability concerning storage requirements and throughput issues while preserving the fundamental features of blockchain technology, such as decentralization and security. First, we provide a theoretical analysis of our system's performance. Next, a practical evaluation is discussed and examined. Finally, we compare the theoretical and practical analyses, as well as other relevant blockchain frameworks. The dataset utilized in this study was exclusively formulated by our research team, which includes random and dummy transactional data. This dataset is automatically generated using a Python script, which takes a number denoting the amount of transactions as input.

6 Theoretical analysis

6.1 Evaluating storage efficiency

Our system consists of two separate types of ledgers: IPFS blockchain and Actual blockchain. The IPFS blockchain holds multiple verified transaction CIDs in original blocks, which are stored on IPFS rather than the miner's device. Conversely, the final blocks contain only a single CID of a specific original block, with an approximate size of 46 bytes. Including metadata, the size per final block amounts to around 290 bytes. Since the Actual blockchain and final blocks are considerably smaller compared to the IPFS blockchain, miners only need to store a copy of the Actual blockchain instead of the original block ledger. As a result, our system has a significantly reduced storage requirement.

A Bitcoin blockchain block averages around 1 MB in size. As of 1st January, 2023, the bitcoin blockchain height is 769,903 blocks. Thus, the total size of the Bitcoin ledger is approximately 770 GB. In contrast, our system's block size is 290 bytes and remains constant. When compared to the Bitcoin blockchain, our.

system only requires 0.231 GB for the same block height. Therefore, our system necessitates almost 3332 times less storage than the Bitcoin blockchain. The detailed data of theoretical comparison of storage between Bitcoin and our blockchain is presented in Table 2.

Table 2 Comparing the Total Size of Ledger between Bitcoin and the Proposed System

Bitcoin's Block Height (Total Count of Blocks)	769,903 Blocks
Bitcoin's Total Count of Transactions Till Now	792,482,423 Txns
Bitcoin	
Size Per Block	1 MB (average)
Overall Required Storage	770 Gigabytes
Our Experimental Blockchain	
Size Per Block	290 Bytes (Approximately)
Overall Required Storage	0.231 Gigabytes
Our System's Multiplicative Factor	3332 Times Less

Table 3 Comparing transactions per block between bitcoin and the proposed system

Bitcoin Blockchain	
Size Required for Header	80 Bytes
Storage Required for Each Block (Approximately)	1 MB
Size of Each Transaction (Approximately)	249 Bytes
Count of Transactions in Each Block	998 Transactions
Our Experimental Blockchain	
Size Required for Header	80 Bytes
Storage Required for Each Original Block (Approximately)	1 MB
Size of Each CID (Transaction)	46 Bytes
Count of Transactions in Each Original Block	22,000 Transactions (Approximately)
Our System's Multiplicative Factor	23 Times More

6.1.1 Evaluating throughput enhancement

The CIDs of verified transactions are located in original blocks. Each CID has a constant size of 46 bytes, independent of the transaction size. Consequently, CID is significantly smaller than the actual transaction size. As a result, we can include more CIDs than raw transactions in a 1 MB block, which is the standard block size for the Bitcoin blockchain. The current height of the Bitcoin blockchain is 769,903 blocks, containing around 792,482,423 transactions. Hence, each block has an average of 998 transactions. According to Eq. (1), the Bitcoin blockchain's throughput is approximately two transactions per second.

$$\text{Throughput} = \frac{\text{Transaction Amount in Each block}}{\text{Block Time (in seconds)}} \tag{1}$$

This transaction rate per second is significantly lower than centralized and popular financial services such as VISA, PayPal, and others. In contrast, our proposed blockchain can accommodate around 22,000 transactions per block, based on the Eq. (2), with each CID being 46 bytes in size, the block size being the same as Bitcoin's (1 MB on average), and the block header being 80 bytes in size. Table 3 demonstrates that our system's blocks can hold 23 times more transactions than the Bitcoin network. This multiplication factor can be further increased if the block size is expanded. Since the final block size remains constant despite increasing the Original block size, increasing the block size does not result in storage bloating in our system, unlike Bitcoin.

$$\text{Proposed Blockchains throughput} = \frac{\text{Per Block Size} - \text{Size of Block Header}}{\text{Size of Each CID}} \tag{2}$$

7 Practical analysis

7.1 Assessing storage efficiency

In practice, an original block size of 3.4 MB is needed for 22,000 transactions. The block header and hash require 87 and 64 bytes of storage, respectively. Each recipient's address takes up 46 bytes, and each CID needs 64 bytes. The indexing terms for the header, block hash, CIDs, recipient addresses, and all 22,000 transactions occupy approximately 1,012,000 Bytes of original block space.

Table 4 Practical details of original block

Meta-data	Required storage
Header	87 Bytes (Approx.)
Hash	64 Bytes
Total CIDs	1,012,000 Bytes (22,000 Txns X 46 Bytes)
Address of the Receiver	1,408,000 Byes (22,000 Txns X 64 Bytes)
Indices to Indicate Aforementioned Terms	991,867 Bytes (Approx.)
Required Size for Each Original Block	3.4 Megabytes

Table 5 Practical Details of Final Block

Meta-Data	Required Storage
Header	86 Bytes (Approx.)
Hash	64 Bytes
1 CID of Original Block	46 Bytes
Indices to Indicate Aforementioned Terms	92 Bytes (Approx.)
Required Size for Each Final Block	288 Bytes

These practical details of original block are summarized in Table 4.

For final blocks, the header, block hash, and original block CID each need 86, 64, and 46 bytes. The indexing terms for these fields occupy about 92 bytes. Consequently, each final block requires 288 bytes of storage. Table 5 includes the practical details of final block.

Comparing our proposed system to the Bitcoin blockchain in terms of blockchain height, as shown in Fig. 6, our system requires only 222 MB for a height of 785,113 blocks, since the final block requires just 288 bytes of storage. In contrast, the Bitcoin blockchain requires approximately 472.59 GB or 2,129 times more storage than our system.

Figure 7 compares several alternative blockchain protocols to our proposed system in terms of transaction size and transaction count per block. Lpeer has the largest transaction size, at about 4,594 bytes, allowing a 1 MB Lpeer block to hold around 221 transactions. In contrast, our system has the smallest transaction size requirement, at only 288 Bytes, enabling a 1 MB block to store around 22,000 transactions.

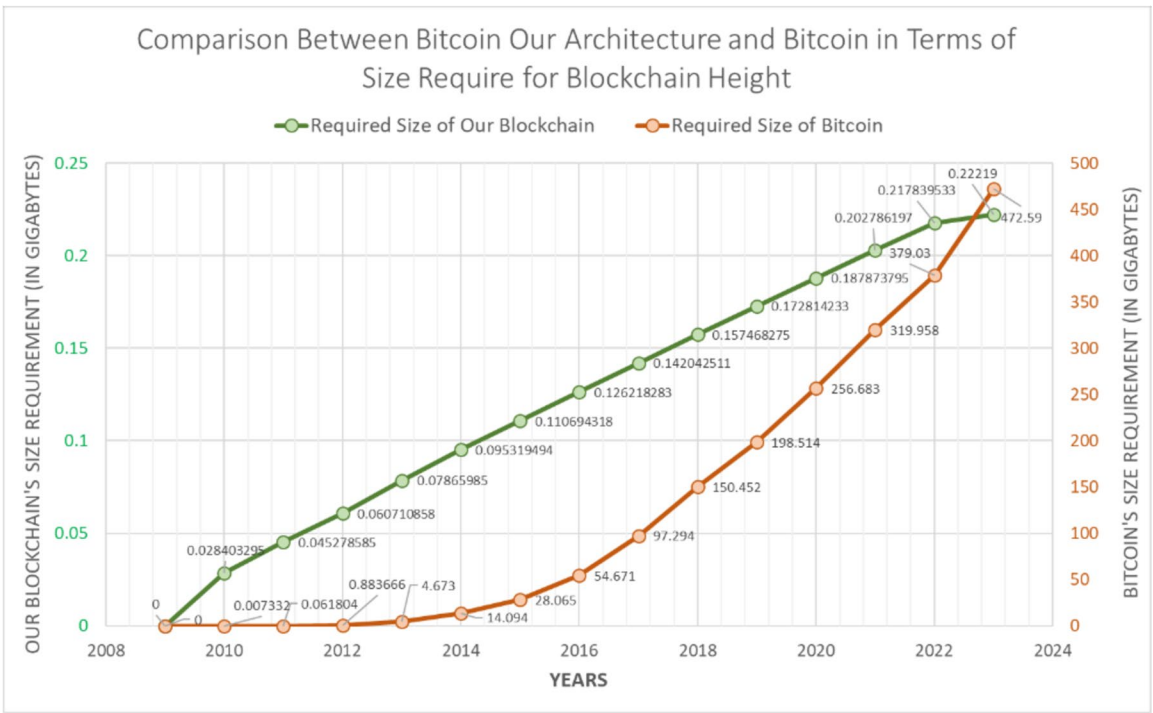


Fig. 6 Comparing blockchain heights: bitcoin versus the presented system

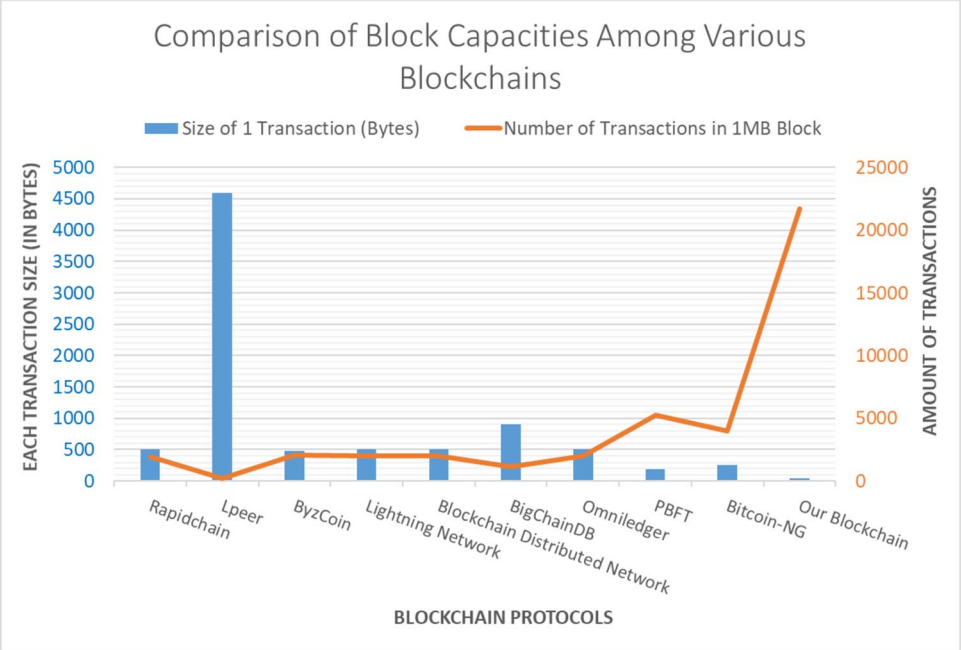


Fig. 7 Assessing transaction sizes and quantity within a 1 MB block

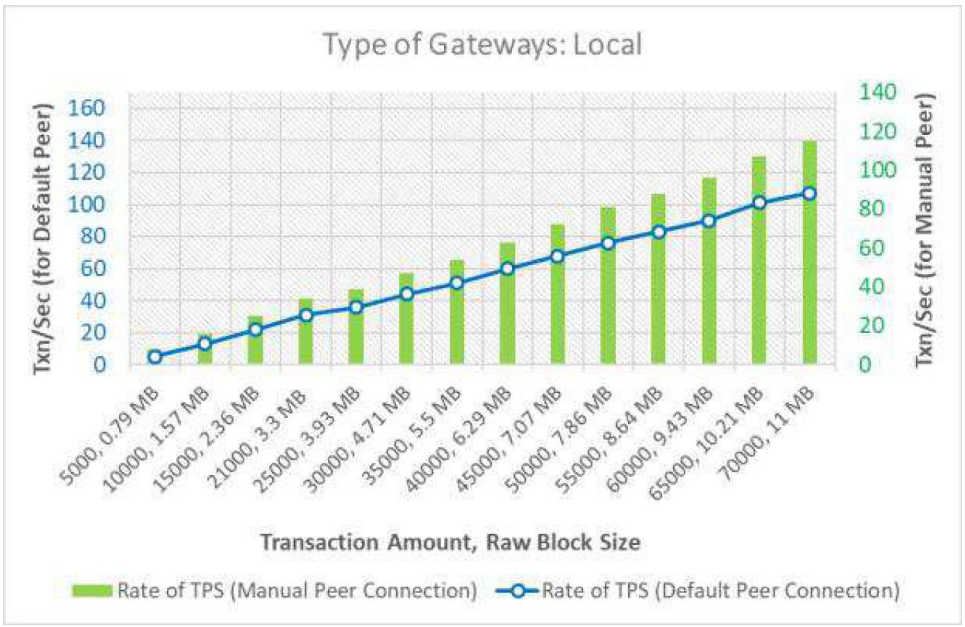
7.1.1 Assessing scalability

Figure 8 displays a line chart illustrating our system's transactions per second (TPS) rate when each miner operates an IPFS node on their devices. The creation of a local gateway at the miner's end eliminates latency in connecting to a public gateway, accelerating the transaction fetching process. However, if a miner's gateway is not directly linked to the nodes storing transaction chunks, the miner may face delays and need to wait to retrieve transactions, reducing our system's TPS rate. Additionally, IPFS mining nodes can lose connectivity, causing content routing to become challenging as it takes time to identify nodes serving transaction chunks using the DHT, leading to a decline in the TPS rate.

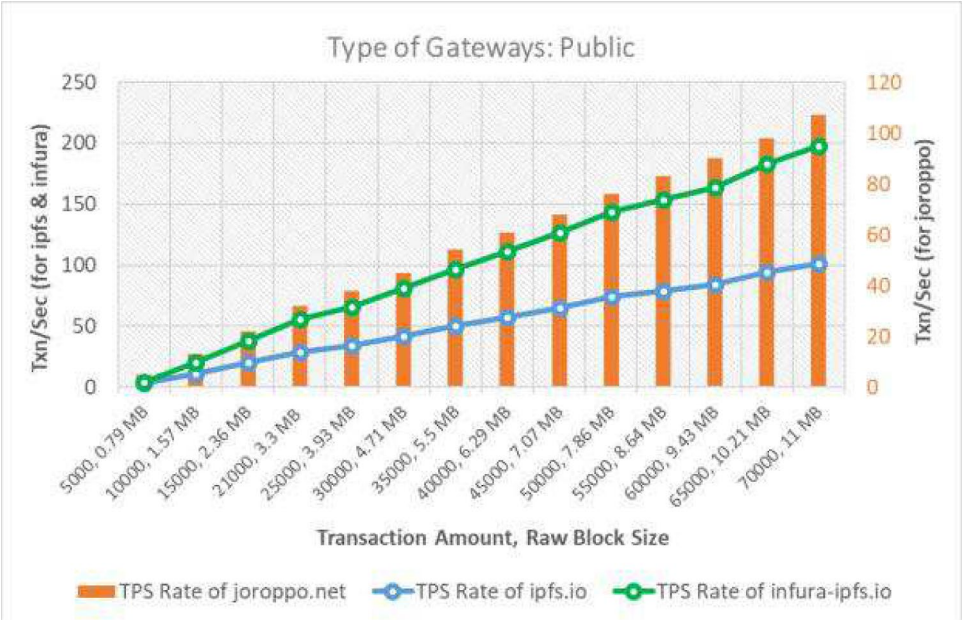
To tackle this problem, we developed a script that manually connects content-providing nodes with miners. This approach enables transactions to be retrieved almost instantly after requesting the IPFS. For instance, the TPS is 43 for 29,949 transactions when using the default connection; however, with a manual peer connection, the TPS increases to 48. The original block size has a minor impact on TPS, as larger original blocks take longer to distribute among miners than smaller ones. Consequently, for large blocks, the TPS rate is slightly lower than expected. The chart indicates that the TPS is 61 when the original block size containing 40,577 transactions is 6.35 MB. The anticipated rate is 67, but distributing a 6.35 MB original block takes marginally longer than a lightweight block, causing the rate to decrease.

Miners can use open IPFS gateways to access transactions without setting up IPFS on their end or installing an IPFS client. Several readily available public gateways can be used to download files from IPFS using a web browser. To evaluate our system's TPS rate, we tested three different public gateways: ipfs.io, joropo.net, and infura-ipfs.io. Joropo.net and infura-ipfs.io have the lowest and highest response times, respectively, while ipfs.io has a moderate response time compared to the other two gateways. As a result, using joropo.net provides a higher TPS rate than using infura-ipfs.io and ipfs.io. For example, we can achieve a TPS rate of 39 with joropo.net for a 3.91 MB block containing 25,164 transactions, while achieving 35 and 31 TPS rates with ipfs.io and infura-ipfs.io, respectively. Comparing Fig. 8a and b reveals that using a local gateway instead of a public one results in faster transaction processing.

Figure 9's bar graph demonstrates the necessary block size for our system to surpass the TPS of various other blockchain systems. By merely increasing the block size, our system can outperform other blockchains' TPS rates, assuming a block interval time of 10 min. Our system can attain a TPS of 32 for the standard block size of 1 MB, higher than both Bitcoin and Bitcoin-NG. To exceed the TPS of Paypal and VISA, which are 542 and 2057, respectively, the block size needs to be increased to 14 MB and 57 MB. Among the blockchains presented in the figure, Rapidchain has the highest TPS, processing nearly 7,002 transactions per second. If the block size is set to 187 MB, our blockchain can achieve 7029 transactions per second, surpassing Rapidchain.



(a)



(b)

Fig. 8 Transaction Processing Speed: **a** Local Gateway, **b** Public Access Points

8 Conclusion

The proposed system attains an average throughput rate of 32 transactions per second for 21,000 transactions within a 3.4 MB block, and this rate can be substantially improved by enlarging the original block size. Additionally, our approach necessitates minimal storage, as each block only requires 288 bytes, regardless of the transaction count. These features enhance the system's scalability, while the high level of decentralization and the inclusion of numerous participants are facilitated by its low storage dependency, public accessibility, affordable mining nodes, and off-chain governance.

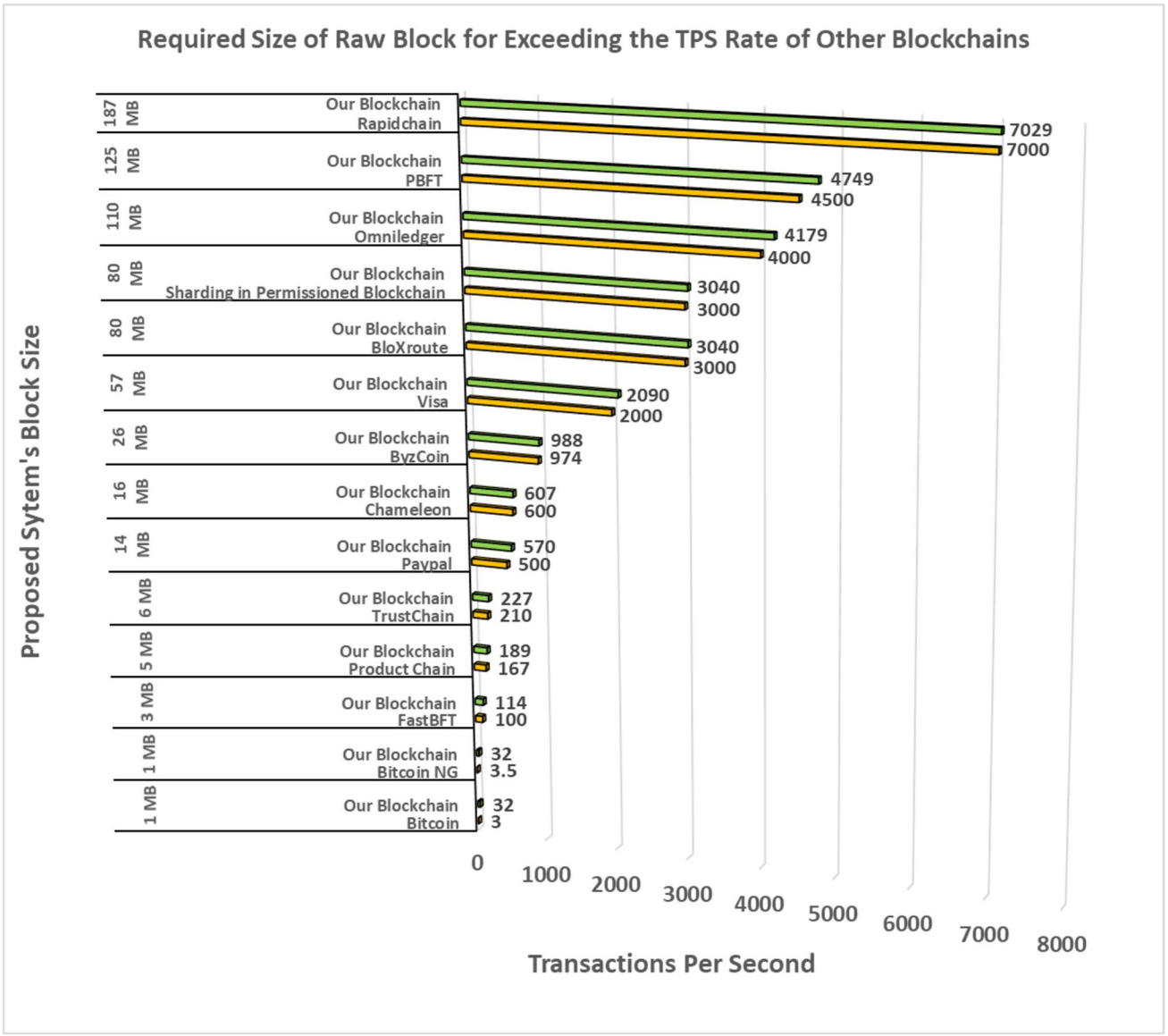


Fig. 9 Outperforming Competing Blockchain Systems in Terms of TPS

The system's characteristics, designed to overcome the limitations of well-known public blockchains like Bitcoin, incentivize more users to join the network, making it easier to resist Sybil and 51% attacks. An innovative countermeasure to build defense against double-spending is also introduced.

Throughput can be further increased by implementing sharding or payment channels on top of our protocol, representing a promising area for future research. Moreover, any scalable consensus mechanism beyond Proof-of-Work can be integrated into our system to enhance scalability, albeit potentially at the cost of security and decentralization.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper. There was no sponsorship or funding agency involved in this study. The views and opinions expressed in this article are those of the authors and do not necessarily reflect the official policy or position of any affiliated agencies of the authors. This research was conducted independently and did not involve any scenarios that could give rise to a conflict of interest.

References

1. Dang H, Dinh TTA, Loghin D, Chang EC, Lin Q, Ooi BC (2019) Towards scaling blockchain systems via sharding. Proceedings of the 2019 International Conference on Management of Data, pp. 123–140. https://doi.org/10.1145/3299869.3319886
2. Ehmke C, Wessling F, Friedrich CM (2018) Proof-of-property: a lightweight and scalable blockchain protocol. Proceedings of the 1st International Workshop on Emerging Trends in Software Engineering for Blockchain, pp. 48–51. https://doi.org/10.1145/3194113.3194119
3. Fan X, Chai Q (2018) Roll-dpos: a randomized delegated proof of stake scheme for scalable blockchain-based internet of things systems. Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, pp. 482–484
4. Hazari SS, Mahmoud QH (2019) A parallel proof of work to improve transaction speed and scalability in blockchain systems. 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), pp. 0916–0921. https://doi.org/10.1109/CCWC.2019.8666593
5. He G, Su W, Gao S (2018) Chameleon: a scalable and adaptive permissioned blockchain architecture. 2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN), 2018, pp. 87–93. https://doi.org/10.1109/HOTICN.2018.8605944
6. John K, O'Hara M, Saleh F (2021) Bitcoin and beyond. Ann Rev Financ Econ. https://doi.org/10.1146/annurev-financial-110720-111326
7. Kogias EK, Jovanovic P, Gailly N, Khoffi I, Gasser L, Ford B (2016) Enhancing bitcoin security and performance with strong consistency via collective signing. 25th USENIX security symposium (USENIX Security 16), pp. 279–296
8. Kokoris-Kogias E, Jovanovic P, Gasser L, Gailly N, Syta E, Ford B (2018) Omniledger: a secure, scale-out, decentralized ledger via sharding. 2018 IEEE Symposium on Security and Privacy (SP), IEEE, pp. 583-598. https://doi.org/10.1109/SP.2018.00029
9. Kuzmanovic A (2019) Net neutrality: unexpected solution to blockchain scaling. Commun ACM 62(5):50–55. https://doi.org/10.1145/3292034
10. Tiwari A, Batra U (2021) Ipfs enabled blockchain for smart cities. Int J Inf Technol 13(1):201–211
11. Liu J, Li W, Karame GO, Asokan N (2018) Scalable byzantine consensus via hardware-assisted secret sharing. IEEE Trans Comput 68(1):139–151. https://doi.org/10.1109/TC.2018.2859961
12. Luu L, Narayanan V, Zheng C, Baweja K, Gilbert S, Saxena P (2016) A secure sharding protocol for open blockchains. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2016, pp. 17–30. https://doi.org/10.1145/2976749.2978389
13. Malavolta G, Moreno-Sanchez P, Schneidewind C, Kate A, Maffei M (2018) Anonymous multi-hop locks for blockchain scalability and interoperability. Cryptology ePrint Archive. https://eprint.iacr.org/2018/472
14. Quamara S, Singh AK (2022) Schain: towards the quest for redesigning supply-chain by augmenting blockchain for end-toend management. Int J Inf Technol 14(5):2343–2354
15. Poon J, Dryja T (2016) The bitcoin lightning network: scalable off-chain instant payments. https://lightning.network/lightning-network-paper.pdf
16. Teutsch J, Reitwießner C (2019) A Scalable Verification Solution for Blockchains. arXiv preprint arXiv:1908.04756
17. Pabitha P, Priya JC, Praveen R, Jagatheswari S (2023) Modchain: a hybridized secure and scaling blockchain framework for iot environment. Int J Inf Technol 15(3):1741–1754
18. Balamurugan S, Ayyasamy A, Joseph KS (2021) Iot-blockchain driven traceability techniques for improved safety measures in food supply chain. Int J Inf Technol. https://doi.org/10.1007/s41870-020-00581-y
19. Hossain CA, Mohamed MA, Zishan MSR, Ahasan R, Sharun SM (2022) Enhancing the security of e-health services in bangladesh using blockchain technology. Int J Inf Technol 14(3):1179–1185
20. Zamani M, Movahedi M, Raykova M (2018) Rapidchain: scaling blockchain via full sharding. Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 931–948. https://doi.org/10.1145/3243734.3243841
21. Vacca A, Di Sorbo A, Visaggio CA, Canfora G (2021) A systematic literature review of blockchain and smart contract development: techniques, tools, and open challenges. J Syst Softw 174:110891. https://doi.org/10.1016/j.jss.2021.110891
22. Kwon Y, Kim H, Shin J, Kim, Y (2019) Bitcoin vs. Bitcoin cash: coexistence or Downfall of Bitcoin Cash? 2019 IEEE Symposium on Security and Privacy (SP), IEEE, pp. 935-951. https://doi.org/10.1109/SP.2019.00059