[1] Shweta Babu Prasad
[2] Ashok Kumar A. R
[3] Rajini V Honnungar

Scalable IoT Data Access with Block Chain and Bloom Filters

Abstract: - The Internet of Things (IoT) connects numerous devices that perform unique tasks and gather various types of data, requiring scalable, efficient, and secure management. Traditional centralized solutions have limitations concerning security and scalability. Blockchain, a decentralized system, can enhances data integrity and security but poses challenges because of the IoT devices' resource limitations. A multi-layer blockchain framework is proposed in this study to improve scalability and access control for IoT data management by using multiple permissioned blockchains and enabling parallel processing. As an extension to this, this architecture is also integrated with decentralized storage, InterPlanetary File System or IPFS to facilitate scalability of the framework. Bloom filters have been used in IPFS to optimize data retrieval by filtering out non-existent content before initiating full data retrieval processes. The investigation is separated into four sections: the examination of latency, throughput, and IPFS query and response time and time taken for data retrieval using Bloom's filter. The experimental results show that proposed framework has a low response time for detection of presence of data and measures 1.21345 s when data is present and 0.02476 s when data is not present. The suggested method works better than most of today's cutting-edge consensus methods. Additionally, it is shown that the suggested method works well in IoT applications that demand low latency or resource efficiency.

Keywords: Bloom Filter, Blockchain, IPFS, Data Storage, Data Retrieval

I. INTRODUCTION

By 2035, there could be as many as 1 trillion interconnected IoT devices globally. This projection is largely driven by advancements in IoT platforms and technologies, such as ARM's Pelion IoT platform, which is designed to manage a vast number of devices at scale across diverse environments and networks [1]. IoT networks have become prevalent for a multitude of applications in the fields of business, healthcare, security, tracking, smart homes, and smart grids, among others. This can be attributed to the availability of resource-constrained IoT devices that are cost-effective and practical [2,3]. IoT networks have often been created with centralized technologies and structure. This indicates that information from IoT devices is gathered and processed by a centralized server. But this strategy leaves IoT networks open to privacy and security flaws caused by both physical and cyberattacks [4,5]. Blockchain technology, due to its inherent nature is a strong contender to provide safe IoT network implementations [6]. Blockchain-based Internet of Things (IoT) networks provide an efficient and consistent means of connecting and exchanging data between both physical and virtual devices that have sensors and actuators via the internet [7]

Blockchain is a distributed ledger technology that tracks, supervises, and manages IoT devices through the use of cryptography and decentralization [8,9]. Blockchain transactions are highly dependable since they don't require intermediaries. Its many intrinsic qualities—such as decentralization, immutability, and transparency—offer major advantages in terms of heightened security, safeguarding data from unwanted access, and full process traceability. By combining these cutting-edge technologies, secure and scalable IoT networks may be created, facilitating data sharing amongst stakeholders, systems, and devices.

While blockchain offers robust access control, the storage and retrieval of the vast amounts of data built up by IoT devices present another critical challenge. The Inter Planetary File System (IPFS), a peer-to-peer distributed file system, has been proposed as an ideal solution for decentralized data repository in IoT networks. IPFS enables efficient and secure storage by distributing data across multiple nodes, thus eliminating the need for centralized data servers and reducing the likelihood of data loss [10]. However, the decentralized nature of IPFS also introduces complexities in data retrieval, particularly in large-scale networks where data may be scattered across numerous nodes.

To address these challenges, Bloom filters—a space-efficient probabilistic data structure—can be integrated into the IPFS framework to enhance data retrieval efficiency [11]. Bloom filters allow for fast and efficient existence checks of data within the network, significantly reducing the time and computational resources required to locate

[1] Dept. of Computer Science and Engineering RV College of Engineering, Bengaluru, India. Email: shwetababup@rvce.edu.in, Orcid ID: 0009-0004-1194-1559
[2] Dept. of Computer Science and Engineering RV College of Engineering, Bengaluru, India. Email: ashokkumarar@rvce.edu.in, Orcid ID: 0000-0003-3574-8919
[3] Dept. of Electronics and Communication Engineering RNS Institute of Technology, Bengaluru, India. Email: rajinivh@gmail.com, Orcid ID: 0000-0002-8365-7360

and retrieve specific pieces of data. This is particularly important in IoT environments, where quick access to data is crucial for real-time decision-making and operation.

This paper presents a comprehensive multi-layered architectural framework for IoT data exchange that is both secure and transparent. The design is centered around a dual blockchain topology that includes both a lightweight local blockchain and a management blockchain. Additionally, the framework integrates the IPFS to facilitate the storage of large volumes of data in a distributed, peer-to-peer network. Bloom filters help in indexing and data retrieval from blockchain to facilitate access control in the system.

II. Related Work

The integrating of blockchain, IPFS, and Bloom filters in IoT networks is a cutting-edge area of research that addresses the critical challenges of security, scalability, and efficient data retrieval in distributed environments. This literature survey provides an overview of the latest advancements and contributions in these areas.

[12] demonstrates how the decentralized nature of blockchain can mitigate common security threats, such as unauthorized access and data tampering, and improve the overall reliability of IoT systems.

The concept of Spatial blockchain [13] involves dividing the blockchain into distinct spatial units, while the migration manager function [14] facilitates time-based migration. In this model, each blockchain manages data according to its spatial unit, and over time, the data from all blockchains are consolidated into a single block. This process effectively reduces the size of the blockchain by starting anew from the summarized block.

RapidChain [15] enhances throughput considerably by implementing sharding in public blockchains and can withstand Byzantine faults affecting almost one-third of participants. Each shard is created by arbitrarily assigning participants to committees, ensuring that the proportion of compromised members in any committee remains below one-half. Following this, a synchronous Byzantine consensus protocol is used within each committee to achieve consensus.

[16] introduces a streamlined hierarchical access control system built on blockchain and multi-chaincode for IoT networks, employing a clustering strategy with blockchain managers to boost scalability.

The network of miners in Elastico [17] is organised into several groups, allowing the transaction throughput to rise linearly with the total amount of processing power given to mining. By increasing the number of nodes from 100 to 1600, it is possible to improve the number of blocks generated every epoch from 1 to 16. And epoch period increases from 600 to 711 seconds due to this increase of nodes.

Through parallel transaction processing in an intra-shard manner, Omniledger [18]—which is based on two PoS blockchain technologies termed Ouroboros and Algorand—improves the blockchain's scalability and bias-resistant validators.

RandHound [19] is employed to manage the secondary key security channel. To improve efficiency, a checkpoint-based method is utilized, allowing miners to produce new blocks without needing to download the entire blockchain history. Here, a throughput of 13,000 transactions per second (TPS) can be achieved, even with an adversary rate of 12.5% and 1,800 hosts spread across 25 shards. However, the system requires a significant time (in the order of minutes) to bootstrap each epoch.

Split-scale [20] divides the entire distributed ledger and the UTXO region in an effort to increase throughput without sacrificing decentralization. Whenever the ledger is split into a tree, numerous sub-chains that can function independently are created at each split event. Every sub-chain mines a block in each block interval, which increases the transactional throughput tremendously.

A dynamic multi-chain network was proposed in [21] to facilitate inter-blockchain communication. This system's throughput is based on how many chains are operating concurrently. However, the cross-chain transaction ratio—that is, the proportion of the total number of transactions to cross-chain transactions—affects this throughput. As the ratio increases, throughput rapidly declines. Moreover, this protocol does not contain or make use of encryption or an Access Control component.

A permissioned and private blockchain architecture that uses satellite-chains and is appropriate for industrial organisations is described in [22], wherein several methods of consensus can operate in private simultaneously. These satellite chains keep their own private ledger, which is unavailable to other chain members, yet are simultaneously linked and autonomous. Using a smart contract, a regulator oversees the entire network and enforces guidelines.

[23] introduces a multi-layered Bloom filter technique aimed at enhancing precision by minimizing false positives. As an alternative to using a single Bloom filter, the proposed method employs multiple (N) Bloom filters to support efficient keyword insertion and search operations. This approach achieves an improvement in

performance ranging from 15-30% by significantly reducing false positives compared to the traditional single-filter approach.

In [24], the authors prove that Bloom filters are better than Cuckoo filters in high-throughput scenarios because of their lower lookup costs. Specifically, when dealing with tasks that require rapid filtering to avoid costly operations like CPU work, cache misses, or network messages, Bloom filters—especially the new variants like register-blocked and cache-sectorized Bloom filters—offer a performance advantage. These variants optimize the use of hardware resources, such as wide-SIMD instructions, reducing the time required for lookups. This reduction in lookup costs allows Bloom filters to outperform Cuckoo filters in environments where speed and high throughput are crucial.

An innovative solution to the blockchain scalability challenge is proposed in [25] by adaptively restructuring Merkle and Verkle trees, to ensure data accuracy and effective validation. By employing binary and non-binary tree configurations, the adaptive model, in contrast to static tree structures, dynamically rearranges these trees in response to usage patterns, minimizing computing cost and the average path length for verification. The proposed method offers a scalable, reliable, and effective method for blockchain data verification, potentially accelerating the deployment of technology in industries including logistics and banking.

Count-Min Sketch and HyperLogLog are both probabilistic data structures used for approximating counts and cardinalities but have different strengths [26], [27]. Count-Min Sketch approximates frequency counts of items and is particularly effective in cases with high-dimensional data, providing compact space for approximate frequency queries. Whereas, HyperLogLog, is optimized for approximating the cardinality or unique element count in large datasets, offering very efficient space usage for set size estimation but not designed for frequency counts. While Count-Min Sketch provides estimates with guarantees on error bounds, HyperLogLog excels in scenarios requiring accurate estimates of distinct elements.

III. METHODOLOGY:

As seen in Fig. 1, the suggested framework is built on multi-layer scalable blockchain-based access control for IoT. Several IoT networks were established structured to enhance security and manageability. Devices seeking membership in these networks must first undergo a registration process, which involves authenticating their identity and ensuring compliance with network protocols. Upon successful registration, each device is granted access to its respective network. Each network is equipped with a gateway that acts as a central hub for communication, data aggregation, and routing. The gateway also plays a crucial role in enforcing network policies, monitoring device activities, and facilitating secure data exchanges between devices and external systems. This structured approach guarantees that only authorized devices are permitted to participate in the network, while the gateway maintains robust control over network operations and data integrity.

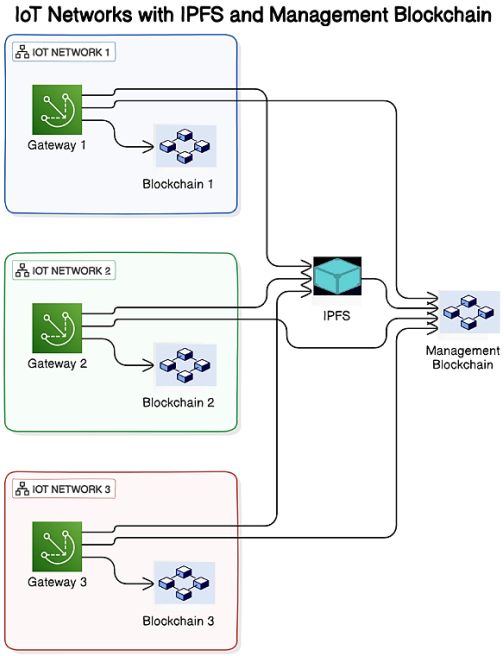

Fig 1: Scalable IoT networks integrated with IPFS

In the proposed blockchain-based system, the network or cluster leader is elected based on the device with the highest remaining energy to establish consensus as described in algorithm 1. Each network operates its own blockchain, and the leader plays a crucial role in validating transactions and maintaining consensus. By selecting the device with the highest energy, it is ensured that the leader has the necessary resources to perform these tasks efficiently, reducing the likelihood of interruptions due to energy depletion. This method promotes network stability and reliability, as the energy-rich leader can handle the computational demands of blockchain operations and manage network consensus without frequent changes. Consequently, this technique improves performance and durability of the blockchain network, ensuring robust and uninterrupted consensus processes. This algorithm ensures that the leader, chosen for its optimal energy resources, can effectively handle the demands of blockchain consensus and network management.

Algorithm 1: Leader Election Based on Energy
Ensure: LeaderAssigned

Let N be the set of devices in the network, with N= $\{D_1, D_2, \ldots, D_N\}$.
Let E(D_i) denote the energy of device D_i.

1. **Collect Energy Data**
   For each device D_i in N, collect its energy level:
   E(D_i) for each i=1,2,..,n.
2. **Identify Device with Highest Energy**
   Determine the device with the highest energy level:
   Define: $D_{max} = \arg\max_{D_i \in N} E(D_i)$
   Where: E(D_{max})= $\arg\max_{D_i \in N} E(D_i)$
3. **Elect Leader**
   The device with the highest energy D_{max} is elected as the leader:
   LeaderID=D_{max}
4. **Leader Responsibilities**
   Assign responsibilities to the leader device D_{max}.
5. **Update Network**
   Notify all devices and update the network configuration with the new leader:
   NotifyAllDevices(N,LeaderID) UpdateNetworkConfiguration(N,LeaderID)
   LeaderID== $\arg\max_{D_i \in N} E(D_i)$

At the end of its operational cycle, the elected network or cluster leader communicates the current state of the blockchain to the gateway to ensure that all transactions and consensus data are accurately recorded and updated as detailed in algorithm 2. This communication facilitates the synchronization of blockchain records across the network and the central management system. Following this handover, a new leader is elected based on the highest remaining energy among the devices. This rotation ensures that the leader with the most energy resources takes over, maintaining optimal performance and stability for blockchain operations. The transition process minimizes disruptions and ensures continuous, effective management of network consensus and data integrity throughout the system. This algorithm ensures a smooth transition of leadership while keeping the blockchain up-to-date and maintaining network stability.

Algorithm 2: Leader Transition and Blockchain Update
Ensure: NewLeaderElected, BlockchainUpdated

1. **End of Cycle:**
   Check if the current leader's cycle has ended and prepare for leadership transition if needed:
   If C.cycleEnd()=true, then prepare for leadership transition
2. **Communicate Blockchain State**
   (C)=getBlockchainState(C), where S(C): Blockchain state of current leader
   C: current leader device
3. **Update Blockchain**
   Update the blockchain via the gateway:
   updateStatus=gateway.updateBlockchain(S(C))
   If the update is successful:
   If updateStatus=success, then BlockchainUpdated=true

4. **Collect Energy Data**
   Collect the energy levels of all devices in the network:
   E(D_i) for each $D_i \in N$
5. **Elect New Leader**
   Identify the device with the highest energy to elect as the new leader: $D_{max} = \arg\max_{D_i \in N} E(D_i)$
   E(D_{max})= $\arg\max_{D_i \in N} E(D_i)$
   Assign the new leader:
   newLeader=Dmax
   NewLeaderElected=true
6. **Transition Leadership**
   Transition leadership to the new leader:
   C=newLeaderC
7. **Inform the network**
   Notify the network of the new leader and update the configuration:
   NotifyAllDevices(N,newLeader)
   UpdateNetworkConfiguration(N,newLeader)

At the end of each cycle, the gateway receives the updated blockchain data, which contains all the transactions and activities that occurred during that period. The gateway then transmits this data to the IPFS, where it is stored as a Content Identifier (CID) in a decentralized manner. By utilizing IPFS, the blockchain data is distributed across multiple nodes, enhancing both the scalability and resilience of the network. The data from IPFS, including CIDs or hashes, are collected by the gateway and stored on the management blockchain. To uniquely identify the data belonging to a specific network, a *NetworkID* is utilized, while a *GatewayID* is used to identify the particular gateway that processed and stored the data. By associating each CID or hash with its respective *NetworkID* and *GatewayID*, the management blockchain maintains a clear and organized record of the distributed data. The gateway also holds a mapping of the *TransactionID* to *BlockID*, While the nodes store their *TransactionIDs*. This is detailed in algorithm 3.

Algorithm 3: Blockchain Data Storage via Gateway and IPFS
Ensure: Blockchain storage

Let B represent the blockchain data.
Let G represent the gateway.
Let I represent the InterPlanetary File System (IPFS).
Let CID denote the Content Identifier generated by IPFS for the stored data.

1. **End of each cycle**
   At the end of each cycle, check if the cycle has ended and update the blockchain data:
   If cycleEnds()=true, then
   $B_{updated}$=getUpdatedBlockchainData()
   G.receive($B_{updated}$)
2. **Transmit to IPFS**
   Send the updated blockchain data from the gateway to IPFS:
   G.sendToIPFS($B_{updated}$)
3. **Store data on IPFS**
   Store the updated blockchain data on IPFS and obtain a CID:
   CID=I.storeData($B_{updated}$)
4. **Retrieve CID**
   Receive the CID from IPFS via the gateway:
   receivedCID=G.receiveCIDFromIPFS(CID)
5. **Store CID on Management Blockchain**
   Store the received CID on the management blockchain via the gateway:
   G.storeCIDOnManagementBlockchain(receivedCID)
6. **Repeat process**
   The process repeats indefinitely:
   Repeat the process for each cycle: wait for next cycle to end→go to step 1.

For access management in IoT networks, when a particular IoT device with a specific *'DeviceID'* from a *'NetworkID'* requests data, the Gateway uses a Bloom filter as an indexing method to efficiently retrieve the necessary information from the management blockchain. The procedure is given below and is detailed in algorithm 4.

When a device with a specific *DeviceID* from a particular *NetworkID* requests data, the Gateway initiates the data retrieval process. It uses a Bloom filter, a space-efficient probabilistic data structure that acts as an index to quickly determine if the data is present and also by using the *TransactionID* to *BlockID* mapping. The Bloom filter is preloaded with hashes or identifiers of stored CIDs, *NetworkIDs*, and *GatewayIDs* from the management blockchain. The Gateway hashes the *NetworkID* and *GatewayID* along with the requested data identifiers to query the Bloom filter, which indicates whether the data is likely present on the management blockchain. If the filter suggests the data exists, the Gateway retrieves the exact CIDs or hashes from the management blockchain using the *NetworkID* to identify the specific network and *GatewayID* to locate the relevant gateway data. This approach allows the Gateway to efficiently find the data without scanning the entire blockchain, making the search process faster and more resource-efficient. Once the data is confirmed, the Gateway retrieves the records and delivers the requested information to the device.

Algorithm 4: Data Retrieval Using Bloom Filter in IoT Network

Let BM represent the management blockchain.
Let BF denote the Bloom filter

1. **Receive data request**
   Receive a data request from a device in the network:
   request=receiveRequestFromDevice(D_i,N), where D_i is the requesting device, and N is the network identifier.
2. **Initialize Bloom filter**
   Initialize a Bloom filter and load it with data from the management blockchain:
   BF=initializeBloomFilter()
   loadBloomFilter(BF,BM)
3. **Check with Bloom filter**
   Compute the query hash and check the Bloom filter for the presence of the requested data:
   queryHash=hash(N,G,requestedDataIdentifiers)
   isPresent=BF.query(queryHash)
4. **Verify data presence**
   Verify whether the data is present based on the Bloom filter check:
   If isPresent=true, proceed to retrieve data (Step 5) If isPresent=false, return "Error: Data not available".
5. **Retrieve data from management blockchain**
   Locate and retrieve the required data from the management blockchain:
   networkData=locateNetworkData(BM,N)
   gatewayRecords=findGatewayRecords(networkData,G) =
   retrievedData=retrieveCIDsOrHashes(gatewayRecords,requestedDataIdentifiers)
6. **Provide access**
   Return the retrieved data to the requesting device:
   returnDataToDevice(D_i,retrievedData)
7. **Repeat process**
   The process repeats indefinitely:
   Repeat: While true, wait for next data request, then go to Step 1

IV. IMPLEMENTATION/PERFORMANCE EVALUATION

The experiment is conducted on a desktop PC having Intel Core i5-1235U, 1300 Mhz, and 8 GB RAM. This work predominantly uses Go, additionally referred to as Golang, which is Google-designed open-source coding language, to address issues of concurrency, scalability, and maintainability in large-scale software development.

Each node or the IoT device present in the network, is implemented as a Docker container. These nodes are connected through a Docker network.

The networks are externally connected to the decentralized storage IPFS via the gateway. Here, Kubo (formerly known as go-ipfs) is being used, developed primarily in the Go programming language.

The nodes use gRPC protocol for all their communication and it is a Google built a high-performance, freely available RPC (Remote Procedure Call) framework enabling the creation of efficient, cross-platform, client-server communication systems. gRPC uses HTTP/2 for transport, Protocol Buffers (protobuf) for serialization, and provides a robust mechanism for defining services and methods in a clear, language-neutral way. The gateway to IPFS communication takes place using the HTTP protocol. As part of the setup, Docker (https://www.docker.com/) was utilized to initialize local network nodes.

The performance evaluation findings are presented in this part along with a comparison of the PoW and PoS consensus techniques. Determining the scalability possibilities inside the IoT network model is the key objective of this evaluation. Here, the system's ability to link common devices to the Internet is being evaluated in order to determine how well it will be likely to cope with an increasing number of devices over time, even ones with limited processing power. A gateway usually needs to be connected to a network, which can be done wirelessly or through a wired configuration. However, simulation is carried out in the context of this paper using the gRPC protocol. It is crucial to emphasize that the framework under study is targeted at scalable systems that use blockchain technology to guarantee security irrespective of the underlying communication protocol deployed. The performance of the framework is assessed in the ensuing tests through the analysis of parameters such as throughput, latency, CPU time and NET bandwidth use, and the storage needs.

V. LATENCY AND THROUGHPUT

In order to compare the IoT framework's performance to the proposed consensus process, latency and throughput must be evaluated. There are multiple steps in this process, including validation, data addition to a block, and transaction throughput measurement. Latency is the time taken for a data packet to get through the gateway and append the blockchain. The number of completed transactions from the initial to the final transaction is another indicator used to quantify throughput. As the number of blockchain IoT nodes per gateway rises, the changes are recorded. The test setup assists in the evaluation of the performance.

The total count of blockchain IoT nodes ranged from 60 to roughly 500. The block size, which accommodates data packets, was set at 1 MB, and the payload size remained at 50 bytes.

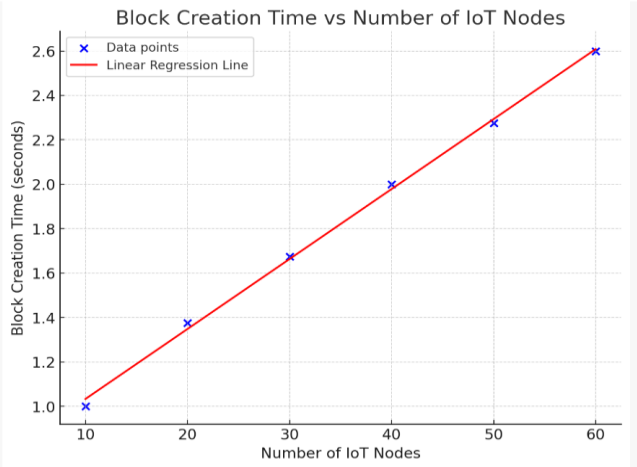

Fig 2: Comparison of Block creation time for different number of IoT nodes

The graph above illustrates the relationship between the number of IoT nodes and block creation time, along with the fitted linear regression line. The linear regression results show that the slope of the regression line is approximately 0.0315, and the intercept is around 0.718. This means that for every additional IoT node, the block creation time increases by about 0.0315 seconds, and the base block creation time (with zero nodes) would be approximately 0.718 seconds.

From the table 1 below it is evident that the proposed consensus mechanism outperforms traditional methods such as Proof of Work (PoW) and Proof of Stake (PoS) in processing transactions by eliminating the need for computational work and complex staking systems. PoW relies on solving energy-intensive cryptographic puzzles, leading to high energy consumption, long processing times, and scalability issues as the network grows. PoS, while more energy-efficient, depends on participants locking up significant amounts of cryptocurrency to validate transactions, which can introduce centralization risks and still requires complex protocols to prevent attacks. In contrast, the proposed mechanism streamlines the consensus process by bypassing both intensive computation and

staking requirements, resulting in faster, more scalable transaction validation. PoW shows very limited scaling, with an increase of only 1 TPS per additional 10 IoT nodes. This highlights its inherent inefficiency due to the high computational cost. PoS performs better than PoW, but its growth remains linear and slow, showing an increase of 5 TPS per 10 additional nodes, which reflects the potential limitations due to staking protocols and transaction validation bottlenecks. The proposed work, on the other hand, scales significantly with the number of IoT nodes. It exhibits rapid TPS growth, increasing by approximately 650 TPS per 10 additional nodes, demonstrating far superior scalability and processing capacity.

Table 1: Comparison of throughput for PoW and PoS for varying number of IoT nodes

Number of IoT Nodes	Transactions per Second (TPS) - PoW	Transactions per Second (TPS) - PoS	Proposed Work (TPS)
10	5	15	2000
20	6	20	2750
30	7	25	3350
40	8	30	4000
50	9	35	4550
60	10	40	5200

The time taken for a transaction to be included in a block, often referred to as "confirmation time" or "block time," varies depending on the blockchain protocol and network conditions. In PoW, this time is influenced by factors such as block size, network congestion, and mining difficulty. For Bitcoin, the average block time is approximately 10 minutes, while Ethereum, which uses a PoS mechanism, typically has a block time of around 12-15 seconds. High transaction volumes can lead to network congestion, increasing the waiting time for transactions to be confirmed. In the proposed work, the time taken for a transaction is measured and compared with PBFT and PoET.

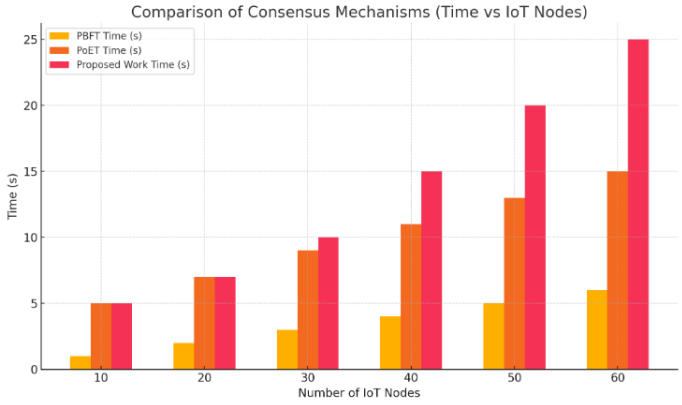

Fig 3: Comparison of Block Confirmation time for PBFT vs PoET vs Proposed work for varying number of IoT nodes

Scalability

The outcomes are noticeable through the analysis of Figs. 2,3 and table 1, it becomes evident that the throughput and transaction speed observed within the current approach speed has good linear scalability when the number of nodes increases. This demonstrates that proposed approach outperforms the PoW and PoS approach and performs well when the number of blockchain IoT nodes increases.

IPFS Query and Response Time

The time taken for the gateway node to query the IPFS and response given by it to the gateway supplying it with the required data is measured for multiple networks varying in the number of IoT nodes. Figure shows the graph of the relationship between the number of IoT nodes and the IPFS query and response time, with a fitted linear trend line. The linear regression analysis shows that for each additional IoT node, the response time increases by about 0.83 ms. This indicates a mostly linear increase in response time as the number of IoT nodes grows. These findings strongly support the widespread adoption and promotion of IPFS in the field of distributed storage applications.

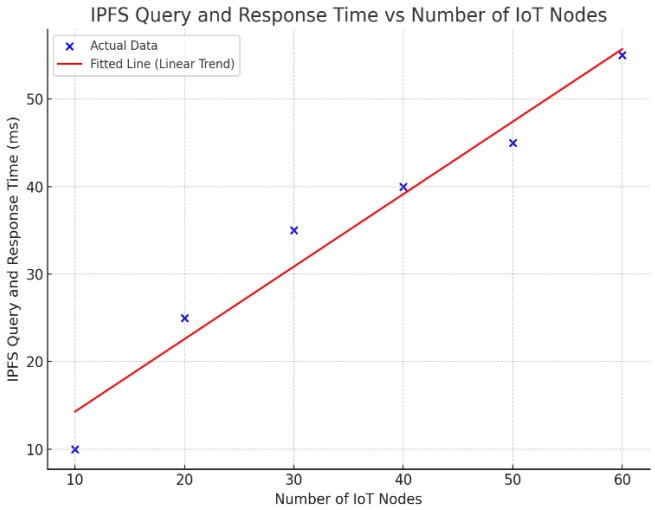

Fig 4: IPFS query and response time for the proposed approach for varying number of IoT nodes.

Data retrieval using Bloom's filter

The gateway receives a transaction data request from an individual node in the IoT network and queries the management blockchain, which stores CIDs acquired from IPFS. This process involves populating a Bloom filter that holds bit positions representing the presence of data. If all the queried positions are set to 1, it indicates that the data is likely present. To achieve this, SHA-256 and MURMUR3 hashing functions are utilized. The time measured to confirm the presence of a block, based on its Block ID, is 1.21345 seconds as shown in Fig 5, while the time taken to indicate the absence of a block is 0.02476 seconds as shown in Fig 6.

```
└─ cat bloom_filter_log.txt

       File: bloom_filter_log.txt

   1   management blockchain: checking for Block ID 226 ( Request from subnet 4)
   2   management blockchain: sending bloom filter to subnet 4
   3   subnet 4: checking for Block ID 226 in bloom filter
   4   subnet 4: found SHA256 hash matching subnet 4
   5   subnet 4: found MURMUR3 hash matching subnet 4
   6   subnet 4: retriving IPFS Hash for Block ID 226
   7   subnet 4: elapsed time 0.06742 second
   8   subnet 4: retriving data with hash QmNk6vLxKwy6nNZ8JQ43JyjdkvsCbfuoyHKbjuG7KDpASk from IPFS
   9   subnet 4: retrived Block ID 226 from IPFS
  10   subnet 4: Total elapsed time 1.21345 second
```

Fig 5: Time measured for block retrieval using Bloom's filter

```
└─ cat bloom_filter_log_2.txt

       File: bloom_filter_log_2.txt

   1   management blockchain: checking for Block ID 216 ( Request from subnet 3)
   2   management blockchain: sending bloom filter to subnet 3
   3   subnet 3: checking for Block ID 216 in bloom filter
   4   subnet 3: not found SHA256 hash matching subnet 3
   5   subnet 3: access for Block ID 216 denied
   6   subnet 3: elapsed time 0.02476 second
```

Fig 6: Time taken to notify that block is absent

Fig 7 shows the comparison between Linear search and Bloom filter. Due to its O(n) complexity, the time required for linear search increases linearly with the number of nodes, whereas the time required for Bloom filter search stays almost constant at O(1). Despite their probabilistic nature and possibility for false positives, Bloom filters are much faster and beneficial for quick membership tests in large datasets because of this efficiency gap, which becomes more noticeable with larger datasets.

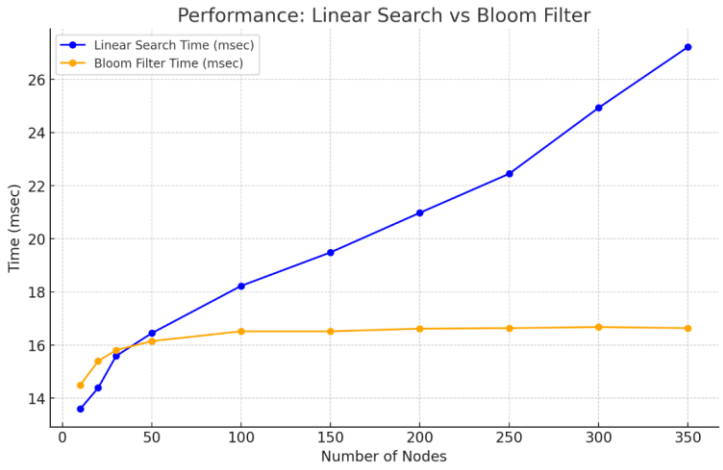

Fig 7: Comparison between Linear search and Bloom's filter for varying number of nodes

VI. System Comparison

There has been substantial investigation into using blockchain to distribute IoT data. However, specific options focus on the massive amount of data generated by IoT devices, which are a vital component of our daily lives. This paper presents an architecture for distributed IoT data storage and scalability. This solution addresses not only the efficacy and performance of an extensive number of IoT devices, but also the considerable difficulty of processing enormous volumes of information from IoT devices in a distributed fashion, as opposed to traditional blockchain approaches that focus solely on system security. IoT devices are popular making them vulnerable to attacks. Furthermore, they do not comprise necessary security management.

Lastly, compared to the conventional blockchain approach, a better throughput can be achieved by employing a lightweight consensus algorithm. The results of the performance study indicate the solution is efficient in terms of throughput and also scalability. This addresses the necessary application needs by processing the massive volume of transactions that are anticipated to be generated from IoT devices at a faster rate while taking into account the limited resources of IoT devices.

VII. Conclusion and Future Work

In the proposed work, a methodology for data access and management in resource-constrained IoT networks is presented. Through verification and validation processes involving elected leader and gateway end-to-end security is achieved while mitigating device performance degradation risks. Metrics including latency, throughput, and resource usage are taken into account for networks with between 60 and 500 devices. Docker is used to measure the network's performance related to throughput, latency and distributed storage provided by IPFS. The investigation is separated into four sections: the examination of distributed storage latency, throughput, and IPFS query and response time.

The proposed methodology can be helpful in IoT applications for resource efficiency and low latency and suitable for real-time uses in the finance and medical sectors due to its lower latency and increased throughput. The extensive use of blockchain technology for safe and secure data managing and storing in medium-sized and big enterprises also depends on their low cost.

References

[1] Mafei, A. A., Lavric, A., Petrariu, A. I. & Popa, V. Massive data storage solution for IoT devices using blockchain technologies. Sensors 23(3), 1570 (2023).

[2] A Holst. Iot Connected Devices Worldwide 2019–2030— Statista. https://www.statista.com/statistics/1183457/iot-connected-devic es-worldwide/ Accessed on 21-October-2022.

[3] Selvarajan, S. & Mouratidis, H. A quantum trust and consultative transaction-based blockchain cybersecurity model for healthcare systems. Sci. Rep. 13(1), 7107 (2023).

[4] Hossein Shafagh, Lukas Burkhalter, Anwar Hithnawi, and Simon Duquennoy. Towards Blockchain-Based Auditable Storage and Sharing of IoT Data. In Proceedings of the 2017 on Cloud Computing Security Workshop, CCSW '17, page 45–50, New York, NY, USA, 2017. Association for Computing Machinery.

[5] Vinothkumar, T., Sivaraju, S. S., Tangavelu, A. & Srithar, S. An energy efcient and reliable data gathering infrastructure using the Internet of Tings and smart grids. Automatika 64(4), 720–732 (2023).Ul Haque, E. et al. Cyber forensic investigation infrastructure of Pakistan: an analysis of the cyber threat landscape and readiness. IEEE Access 11, 40049–40063 (2023).

[6] Novo, O. Blockchain meets IoT: an architecture for scalable access management in IoT. IEEE Internet Tings J. 5(2), 1184–1195 (2018).
[7] Khan, A. A., Laghari, A. A., Li, P., Dootio, M. A. & Karim, S. The collaborative role of blockchain, artifcial intelligence, and industrial internet of things in digitalization of small and medium-size enterprises. Sci. Rep. 13(1), 1656 (2023).
[8] Kutub, T., Al-Sakib, K. P. & Sadia, I. Internet of things (IoT). In Emerging ICT Technologies and Cybersecurity: From AI and ML to Other Futuristic Technologies 165–183 (Springer, 2023).
[9] Kunhahamed, P. K. & Rajak, S. Application of blockchain in mining 4.0. In Blockchain and its Applications in Industry 4.0 (eds Suyel, N. & Kemal, A.) 123–137 (Springer, Singapore, 2023).
[10] Naz, M., Al-zahrani, F. A., Khalid, R., Javaid, N., Qamar, A. M., Afzal, M. K., & Shafiq, M. (2019). A secure data sharing platform using blockchain and InterPlanetary file system. *Sustainability*, *11*(24), 7054.
[11] Sathiya Devi, S., & Bhuvaneswari, A. (2023). Design of efficient storage and retrieval of medical records in blockchain based on InterPlanetary File System and modified bloom tree. *Security and Privacy*, *6*(5), e301.
[12] Gong, Qinghua, et al. "SDACS: Blockchain-Based Secure and Dynamic Access Control Scheme for Internet of Things." *Sensors* 24.7 (2024): 2267.
[13] Alsalih, W.; Islam, K.; Rodríguez, Y.N.; Xiao, H. Distributed voronoi diagram computation in wireless sensor networks. In Proceedings of the SPAA, Munich, Germany, 14–16 June 2008; p. 364.
[14] Dennis, R.; Owenson, G.; Aziz, B. A temporal blockchain: A formal analysis. In Proceedings of the 2016 International Conference on Collaboration Technologies and Systems (CTS), Orlando, FL, USA, 31 October–4 November 2016; pp. 430–437.
[15] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: Scaling blockchain via full sharding," in Proc. ACM CCS'18, pp. 931–948, 2018.
[16] Abdi, Adam Ibrahim, et al. "Hierarchical blockchain-based multi-chaincode access control for securing IoT systems. *Electronics* 11.5 (2022): 711.
[17] Luu, L., et al.: A secure sharding protocol for open blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 17–30 (2016)
[18] KokorisKogias, E., et al.: A secure, scale-out, decentralized ledger via sharding. In: 2018 IEEE Symposium on Security and Privacy (SP), pp. 583–598. IEEE (2018)
[19] Syta, E., et al.: Scalable bias-resistant distributed randomness. In: 2017 IEEE Symposium on Security and Privacy (SP). Ieee, pp. 444–460 (2017)
[20] Özyılmaz, K.R., Patel, H., Malik, A.: Split-scale: scaling bitcoin by partitioning the utxo space. In: 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), pp. 41–45. IEEE (2018)
[21] Kan, L., et al.: A multiple blockchains architecture on inter-blockchain communication. In: 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), pp. 139–145. IEEE (2018)
[22] Li, W., et al.: Towards scalable and private industrial blockchains. In: Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts, pp. 9–14 (2017)
[23] S. Saha and D. V. N. Siva Kumar, "Enhancement of Precision on Cloud Data using Multi-Layered Bloom Filter," *2021 IEEE 18th India Council International Conference (INDICON)*, Guwahati, India, 2021, pp. 1-7, doi: 10.1109/INDICON52576.2021.9691668.
[24] Lang, Harald, et al. "Performance-optimal filtering: Bloom overtakes cuckoo at high throughput." *Proceedings of the VLDB Endowment* 12.5 (2019): 502-515.
[25] Kuznetsov, Oleksandr, et al. "Adaptive Restructuring of Merkle and Verkle Trees for Enhanced Blockchain Scalability."*arXiv preprint arXiv:2403.00406* (2024).
[26] Kang, Sinjung, and DaeHun Nyang. "Count-Min HyperLogLog: Cardinality Estimation Algorithm for Big Network Data." *Journal of the Korea Institute of Information Security & Cryptology* 33.3 (2023): 427-435.
[27] V. Clemens, L. -C. Schulz, M. Gartner and D. Hausheer, "DDoS Detection in P4 Using HYPERLOGLOG and COUNTMIN Sketches," *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, Miami, FL, USA, 2023, pp. 1-6, doi: 10.1109/NOMS56928.2023.10154315.