Empirically comparing the performance of blockchain’s consensus algorithms

Ashar Ahmad1 | Abdulrahman Alabduljabbar1 | Muhammad Saad1 | DaeHun Nyang2 | Joongheon Kim3 | David Mohaisen1

1 Department of Computer Science, University of Central Florida, Orlando, USA
2 Department of Computer Science and Engineering, Ewha Womans University, Seoul, South Korea
3 School of Electrical Engineering, University of Korea, Seoul, South Korea

Abstract
Blockchain-based audit systems suffer from low scalability and high message complexity. The root cause of these shortcomings is the use of “Practical Byzantine Fault Tolerance” (PBFT) consensus protocol in those systems. Alternatives to PBFT have not been used in blockchain-based audit systems due to the limited knowledge about their functional and operational requirements. Currently, no blockchain testbed supports the execution and benchmarking of different consensus protocols in a unified testing environment. This paper demonstrates building a blockchain testbed that supports the execution of five state-of-the-art consensus protocols in a blockchain system; namely PBFT, Proof-of-Work (PoW), Proof-of-Stake (PoS), Proof-of-Elapsed Time (PoET), and Clique. Performance evaluation of those consensus algorithms is carried out using data from a real-world audit system. These results show that the Clique protocol is best suited for blockchain-based audit systems, based on scalability features.

1 | INTRODUCTION AND PROBLEM STATEMENT

Audit systems use blockchains to harden the security of audit logs and provide a fine-grained provenance model for data flows [1]. From a security standpoint, blockchains prevent the audit logs from two well-known attacks, “the physical access attack” and “the remote vulnerability attack”, which both allow an adversary to corrupt data in the audit logs [17, 18]. For provenance, the blockchain ledger maintains a high-level history of changes made in the audit logs, which can be used for fault detection if needed. However, despite such benefits, blockchain-based audit systems suffer from low scalability and throughput. The root cause of the poor scalability is the use of the “Practical Byzantine Fault Tolerance” (PBFT) consensus protocol, which has a high message complexity and, as a result, low scalability. To reduce the complexity, a design optimization has been recently proposed which fragments blockchains into hierarchical layers to support parallel processing as the system scales up [4]. However, the proposed design also uses the PBFT consensus protocol. Therefore, beyond nominal optimizations, the existing audit systems still suffer from low scalability.

Moreover, in all notable efforts on blockchain-based audit logs, the performance evaluation has been done using an abstract simulation of the proposed frameworks [1, 4]. Intuitively, simulations convey a general idea about the system performance. However, they provide limited information about the design feasibility in the real world. Some frameworks that may appear to be operational in a simulated environment may not be fully relevant in real-world settings. Acknowledging that, in [1], the authors mention that their proposed system BlockAudit is yet to be evaluated in the production environment to study its suitability to the actual audit system.
With these two challenges in mind, an intuitive solution would be to apply alternative consensus protocols in the blockchain-based audit systems for optimization. Some of the notable consensus protocols include Proof-of-Work (PoW), Proof-of-Stake (PoS), Proof-of-Elapsed Time (PoET), and Clique [7]. The benchmarking and evaluation of these protocols require realistic testing environments that can accurately model the varying dynamics of the audit systems and the distributed blockchain. Such a comparative analysis can provide new and more accurate insights to improve the performance of the blockchain-based audit logs.

The intuitive approach to tackle these problems has its own inherent challenges. The existing benchmarks for the aforementioned consensus protocols are usually the latency in end-to-end transaction processing and the transaction throughput. Audit systems cannot tolerate latency in transaction processing beyond a few milliseconds [12] (Section 3.1). In PBFT-based systems, the processing latency increases with the number of network nodes. Therefore, the performance is affected as the system scales up. However, in the PoW-based systems, the latency margins show minimal growth as the number of nodes increases. As a result, they are considered highly scalable and more useful for large distributed networks. The throughput of a blockchain system is measured by the number of transactions processed per second. In the literature, and due to varying system features, the reported throughput of consensus protocols is often not comparable. For example, in [6], the authors compare the throughput of Bitcoin with Visa, stating that Bitcoin has a low throughput of 7 transactions per second, while Visa can process up to 24,000 transactions per second. However, the two systems are not evaluated under the same settings. The transaction size of Bitcoin and Visa vary significantly, and therefore the total number of transactions processed in a second are not easily comparable.

Motivation and Contributions. In this paper, we address the scalability problem in blockchain-based audit systems by varying consensus protocols and comparing their performance based on audit system requirements. We develop a testbed of blockchain nodes and implement five different consensus protocols. We then obtain the audit system requirements such as the transaction size, the network size, and the transaction confirmation time from a real-world audit application. Based on those requirements, we conduct various experiments to identify the most suitable consensus protocol for scalable blockchain-based audit systems. Our key contributions are summarized below.

- We develop a testbed of 250 blockchain nodes hosted across various Autonomous Systems (ASes). We implement five different consensus protocols in our testbed and enable feasible switching between those protocols to expand the scope of our experiments and evaluation.
- We obtain real-world audit logs from [12] to model the audit system requirements such as the transaction size, the network size, and the transaction confirmation time.
- We conduct various experiments to identify the consensus protocols that meet the audit system requirements. Our results show that PoET, PoS, and Clique are capable of meeting the scalability requirements of blockchain-based audit systems. Among them, Clique incurs the minimum transaction confirmation latency, and therefore, the maximum throughput.
- To the best of our knowledge, this is the first work that analyzes the scalability of blockchain-based audit systems by varying the consensus protocols in an actual blockchain testbed.

We want to emphasize that the main objective of this work is to identify suitable consensus protocols that meet the scalability requirements of blockchain-based audit systems. Therefore, we design our experiments towards that objective and draw conclusions that best meet the audit system requirements. Our results can also be generalized to other systems that are similar to blockchain-based audit logs.

Organization. The rest of the paper includes background and preliminaries in Section 2, testbed design and deployment in Section 3, results and evaluation in Section 4, related work and discussion in Section 5, and concluding remarks in Section 6.

2 | BACKGROUND AND PRELIMINARIES

In this section, we provide the background of blockchain-based audit logs, mostly based on the prior work in [4], and discuss their limitations that motivate our work and provide a brief overview of five consensus protocols that we evaluate.

2.1 | Blockchain-based audit logs

Audit logs are a critical component of information systems management, often used to track data changes. Changes in data stores trigger the generation of transactions, which are stored in the audit log. The audit logs are then consulted, where needed, for a historical record of all data changes. As such, the audit logs can be used for rollback and recovery (from intentional and unintentional data corruption), as well as for providing fine-grained information for provenance that can guide attribution of the data changes. In most of today’s audit log systems, centralization of the audit log storage is a common theme, making them prone to a single point of failure, and exposing them to various forms of attacks [17]. To counter these attacks, blockchains have been used to provide a distributed, tamper-proof, and consensus-driven replication of audit logs across multiple replicas [3, 4].

However, today’s blockchain systems for audit logs have various shortcomings, including the following. First and foremost, the state-of-the-art blockchain-based audit systems mostly use the PBFT consensus protocol [3, 4], which suffers from high message complexity and low scalability. Second, the existing systems suffer from a high-latency, even with optimizations. For example, even with optimizations that rely on fragmenting the blockchain system into multiple layers, as in [4], the transaction confirmation time ranges from 2 to 20 s. On the other hand,
audit systems would typically mandate a confirmation time in the order of milliseconds. Third, existing evaluations and assessments for the performance of blockchain systems heavily relied on simulations that abstract (or even ignore) various parameters that are hard to measure, such as the network latency, and compute capabilities, especially on shared infrastructure, making it difficult to reason about the relevance and practicality of those results; for example, [3, 4]. In [4], for example, BlockTrail was tested across 250 nodes with varying transaction rates. However, we note that the same system uses PBFT, which has a high message complexity and is unlikely to scale to 250 nodes. Indeed, using our testbed, we demonstrate in Section 4 that PBFT is impractical beyond 50 nodes. There is a need for performance evaluation of blockchain-based audit systems, taking into account the actual characteristics of networks and various blockchain consensus protocols.

2.2 Consensus protocols

The goal of this study is to evaluate various consensus protocols for audit logs. In distributed systems, consensus protocols ensure that the participating nodes (or processes) agree upon the correct state of data, typically assuming faulty or Byzantine nodes (behaves arbitrarily), which may halt the execution of processes. To address that, consensus protocols have the property of fault tolerance, which means that they can successfully execute the consensus objective notwithstanding a certain number of faulty replicas. There are many protocols developed in the distributed systems community to achieve consensus, including PoW, PoS, PBFT, Clique, PoET etc [22]. The main objective of those protocols is to replicate ledgers (blockchain) across multiple nodes, ensuring consistency.

Proof-of-work. The PoW consensus protocol involves solving a computationally expensive challenge to elect a leader [21]. In PoW-based systems, the leader is also called a miner, who solves the challenge, orders transactions in a block, and broadcasts the block to the network. Upon receiving the block, other network nodes validate the correctness of the PoW solution and append the block to their blockchain. PoW is popularly used in cryptocurrencies such as Bitcoin and Ethereum. In Figure 1, we show an abstraction of PoW protocol. Note that the blockchain data structure shown in Figure 1 is common among many blockchain systems, where a block is linked to its parent block using a one-way hash function, and the hash function provides immutability and collision resistance to the blockchain ledger. Moreover, Figure 1 shows that the hash of the block header is required to be less than or equal to the target value set by the system. The target value can be calibrated to keep the block generation rate under a specified limit. PoW is known to be highly scalable since it sidesteps the multi-round message propagation in PBFT. At the same time, PoW is also considered to be energy-inefficient since it leads to a significant waste of critical computational resources.

Proof-of-stake. PoS protocol addresses the energy inefficiency of PoW and replaces the computational requirements with the notion of stake in the system [16]. In PoS cryptocurrencies, the stake is the number of coins owned by a user, which are then used to make bids during a block auction process. The auction winner is allowed to propose a block. Due to the growing concerns about the energy consumption in PoW, cryptocurrencies including Ethereum are switching to PoS. Despite the obvious benefits, PoS has its limitations, including the problem of “the rich gets richer” [8]. The auction naturally favours rich miners who are rewarded with a transaction fee. As a result, their stake further increases which allows them to win the subsequent auctions as well.

Proof of elapsed time. PoET protocol uses Intel Software Guard Extensions (SGX) to execute the “leader election” code in a secure enclave. Each blockchain node executes the code, generate a random wait time during which the node remains idle. Once the wait time expires, the node is allowed to propose a block [11]. PoET randomizes the leader election process to maintain decentralization. In contrast, PoW and PoS may
### TABLE 1  
Comparison of the consensus protocols considered in this study. The values mentioned in the table are collected from various sources including [20, 27, 29]. From the values reported below, PBFT seems to achieve the highest throughput with over 10,000 transactions per second.

| Properties          | PoW          | PoS          | PBFT        | Clique      | PoET         |
|---------------------|--------------|--------------|-------------|-------------|--------------|
| Blockchain type     | Permissionless| Permissionless| Permissioned| Permissioned| Permissionless|
| Trust model         | Untrusted    | Untrusted    | Semi-trusted| Semi-trusted| Untrusted    |
| Scalability         | High         | High         | Low         | High        | High         |
| Throughput          | <10          | <1,000       | <10,000     | 1,000       |
| Fault tolerance     | 50%          | 50%          | 33%         | 50%         |
| Energy efficient    | No           | Yes          | Yes         | Yes         |

![Figure 2](image)

**Figure 2**  
An abstraction of the Clique protocol’s execution. For simplicity, we show that Clique uses a round-robin scheme to select primary replica. In each round, if the primary fails, the secondary replicas propose a block. The total number of replicas in one round are selected using the formula $N = N/2 + 1$, where $N$ is the total number of nodes. Node 1 is the primary and Node 2 and Node 3 are secondary replicas for the first round; for the second round, Node 2 is the primary and Node 3 and Node 4 the secondary replicas. The selected primary replicas propose blocks at their respective turns. To avoid forks, in each round, only one node is allowed to propose a block. Since the identity of each node is known prior to the execution of the protocol, a violation or deviation can be easily detected. Currently, Clique is being experimented in the Ethereum Geth client. In Figure 2, we provide an abstract execution of the Clique protocol.

Clique. Clique belongs to the family of Proof-of-Authority (PoA) consensus protocols, popularly used in the permissioned blockchains [5]. Clique is executed in epochs, and at the start of a new epoch, a transition block is issued which specifies the order of the next primary replicas (authorities). The selected primary replicas propose blocks at their respective turns. To avoid forks, in each round, only one node is allowed to propose a block. Since the identity of each node is known prior to the execution of the protocol, a violation or deviation can be easily detected. Currently, Clique is being experimented in the Ethereum Geth client. In Figure 2, we provide an abstract execution of the Clique protocol.

Practical Byzantine fault tolerance. PBFT belongs to the family of BFT protocols, popularly used for the state machine replication. Blockchain systems use PBFT to obtain consensus over the ordering of transactions in a block. PBFT is executed in three phases, namely, pre-prepare, prepare, and commit phase. The general workflow of PBFT is summarized below. 1. In the pre-prepare phase, the primary replica receives transactions from a client, orders them in a block, and broadcasts the block to the rest of the replicas in the blockchain network. Upon receiving the block, each replica validates if the transactions are ordered correctly. 2. In the prepare phase, each replica broadcasts its “approval” to all the other replicas. Each replica waits for at least $2f$ responses from other replicas, where $f$ is the number of faulty replicas. 3. When $2f$ responses are collected, replicas enter the “commit” phase where they broadcast their commitment to the client. If the client receives $f + 1$ commitments, it adds the block to the blockchain.

**Comparative analysis.** In Table 1, we show some statistics about the consensus protocols evaluated in this work. It can be observed that PBFT has the lowest fault tolerance (33%) and the highest throughput (10,000 transactions per second). The fault tolerance of each protocol is derived from their theoretical analysis, while the throughput measurements are obtained from the experiments and simulations. As mentioned in Section 1, the experimental results of these protocols are not comparable since they were evaluated in different settings with varying transaction and network sizes. Our testbed resolves this issue by creating a unified testing environment for each consensus protocol.

## 3 TESTBED DESIGN AND DEPLOYMENT

We now present details of our blockchain testbed. First, we outline the functional requirements of our audit system, which provides the baseline thresholds for scalability and throughput.

### 3.1 Audit system requirements

For our testbed instrumentation, we used a real-world audit system provided by a property appraisal enterprise system (application) [12]. The system serves 20–200 clients at any time, depending upon client requirements. Therefore, we expected the blockchain network size to vary between 20–200 nodes at any time.

We then obtained audit log transactions ClearVillage [12] and measured the transaction size. In Figure 3, we show the generic transaction data structure that captures a change in the database value. We observed that the audit log transaction size varied between 1 and 4 MB, with an expected confirmation time of 50 ms. In other words, with a maximum network size of 200 nodes and a maximum transaction size of 4 MB, the application required the audit log transaction to be processed under 50 ms. We used these thresholds as our baseline benchmarks to evaluate the performance of each consensus protocol. The protocols...
that exceeded the baseline thresholds were considered practically feasible for our blockchain-based audit system.

3.2 | Blockchain nodes

We set up the blockchain nodes using Docker containers. Docker uses OS-level virtualization to create lightweight, isolated, and standalone software packagers called containers [19]. A Docker container contains the application code, runtime system tools, and system libraries by default. This allows the container application to seamlessly run in any environment that supports Docker, sidestepping the complexities of the host server configurations. Specific to the requirements of our testbed, Docker enables us to swiftly spawn multiple blockchain nodes across various data centers. Moreover, Docker also supports dynamic adjustment of the network size, which allowed us to test the performance of each consensus protocol at varying network size. However, for this experiment, we maintain a fixed topology in which the network is a completely connected graph.

We deployed the testbed nodes on the digital ocean data center, which hosts its data centers across various ASes. Through the geographic distribution of nodes, we aimed to incorporate realistic transaction propagation delay as expected in a real world blockchain system. Each server in the data center hosted 50 Docker containers with public IP addresses and port numbers ranging between 42421–42470. The port mapping scheme was specified in the Docker configuration file. Each server had an Intel Xeon processor with 4 cores, 16 GB RAM, and 500 GB hard drive.

3.3 | Communication model

To enable communication among the blockchain nodes, we used NodeJS, an open-source JavaScript framework that allows asynchronous execution of the JavaScript code. At each container, we also installed a blockchain node middleware, built using the express minimalist web framework. The middleware contained the rules of the consensus protocols and communication model. At the application layer, each node communicated using HTTP protocol with the GET and POST methods to exchange data (i.e. transactions and blocks) with other nodes. In Figure 4, we provide an overview of our testbed design. The Docker containers were hosted in five cities namely San Francisco, New York, Singapore, London, and Frankfurt. The broad distribution of nodes helped us to closely model real-world blockchain systems (globally distributed), where nodes are hosted across multiple cities. As a result, the latency observed in the transaction confirmation was realistic, adding reliability to our methodology and evaluation.

3.4 | System adjustments

Since each consensus protocol has unique rules, we tailored our system to correctly implement them. For PoW, we used the “work-token” library in NodeJS that allows us to modularly adjust the difficulty (see Figure 1) for the target threshold. We selected a difficulty limit that allowed block generation after every 10 ms. When the PoW protocol was executed, each node used its processing power to solve the challenge. Since the network was completely connected, a block is received by all the other nodes directly from the winner who produced the block.

For PoS, we randomly assigned stakes of value between 100 and 10,000. To simplify the auction process, we embedded stakes within the block header. In each round, every node generated its own block with a randomly selected stake and broadcast the block to all other nodes. It also received blocks from the other nodes with their specified stakes. Only the block with the highest stake was selected as the winner.

FIGURE 3 Audit log transaction obtained from [12]. The transaction captures the change in database by incorporating the previous data value and the updated data value. We observed that on average, the audit log transaction size varied between 1MB–4MB
To implement PoET, we sidestepped the cost of installing the Intel SGX at each node. Instead, we relaxed the trust model by replacing the SGX code with a trusted code that issued a random waiting time for each node. We acknowledge that replacing SGX relaxed the security guarantees of our testbed. However, within the scope of this work, we are primarily concerned with the performance evaluation rather than the security guarantees. Therefore, for PoET, as outlined in Section 2, random waiting times were allocated to each node and once the waiting time expired, the node released its block. The waiting time calibration prevented the issuance of multiple blocks within one round to avoid forks.

To implement Clique and PBFT, we followed a round-robin scheme for the selection of primary replicas. For Clique, we encoded the primary selection procedure in the Clique.js file at each node. The protocol followed the same workflow as outlined in Figure 2. For PBFT, we also switched the primary replica in each round.

4 RESULTS AND EVALUATION

We evaluate the scalability and performance of consensus protocols by (1) increasing the network size, (2) increasing the transaction size, and (3) limiting the transaction confirmation time. Based on the audit application requirements (Section 3.1), we set the upper bound on the network size (250 nodes), the transaction size (4 MB), and the transaction confirmation time (50 ms). For simplicity, and without losing generality, we treated one transaction as a unique block. For each protocol, we evaluated the transaction throughput as the number of transactions processed in one second. In Table 2 and Figure 5, we present the results obtained from our experiments. In Figure 5(a–e), we

| TABLE 2 | Results collected from our experiments on each consensus protocol. We varied the transaction size from 1MB to 10 MB and the network size N from 10 nodes to 250 nodes. The table values show the time (in milliseconds) taken to process each transaction |
|---------|---------------------------------|
|         | 0.5 MB | 1 MB | 2 MB | 4 MB | 10 MB |
| PoW     |        |      |      |      |       |
| N=10    | 20.5   | 24.1 | 19.5 | 21.8 | 39.5  |
| N=100   | 45.41  | 74.44| 53.49| 57.84| 71.91 |
| N=150   | 96.83  | 76.01| 88.6 | 65.5 | 81.13 |
| N=200   | 91.31  | 91.4 | 90.49| 95.01| 108.35|
| N=250   | 103.26 | 101  | 128.57| 116.44| 149.53|
| PoS     |        |      |      |      |       |
| N=10    | 2.83   | 2.71 | 2.70 | 2.78 | 2.94  |
| N=100   | 5.46   | 5.68 | 4.7  | 5.19 | 5.33  |
| N=150   | 5.89   | 5.08 | 5.23 | 6.43 | 5.32  |
| N=200   | 7.51   | 7.62 | 6.4  | 7.51 | 7.15  |
| N=250   | 7.75   | 7.19 | 6.45 | 7.12 | 7.13  |
| PoET    |        |      |      |      |       |
| N=10    | 1.79   | 1.84 | 1.81 | 1.71 | 2.02  |
| N=100   | 6.13   | 6.28 | 5.52 | 6.32 | 6.48  |
| N=150   | 8.02   | 9.57 | 7.70 | 7.84 | 8.96  |
| N=200   | 9.46   | 10.89| 11.17| 13.14| 11.45 |
| N=250   | 14.10  | 14.28| 14.69| 15.68| 15.87 |
| Clique  |        |      |      |      |       |
| N=10    | 0.78   | 1.30 | 0.92 | 0.61 | 1.08  |
| N=100   | 3.47   | 9.37 | 6.57 | 4.38 | 6.36  |
| N=150   | 3.16   | 4.29 | 3.33 | 4.19 | 7.51  |
| N=200   | 3.41   | 5.41 | 6.22 | 8    | 10.11 |
| N=250   | 5.01   | 10.42| 12.82| 14.41| 10.11 |
| PBFT    |        |      |      |      |       |
| N=5     | 2175.59| 2171.31| 2171.22| 2174.26 | 2178.39|
| N=10    | 2227.91| 2234.82| 2250.31| 2228.45 | 2229.23|
| N=30    | 14933.14| 14432.71| 9771.51| 15214.33| 17365.31|
Figure 5 shows that, on average, transactions experienced minimum latency in PoET and Clique. For Clique, the maximum latency was \( \approx 14 \) ms with a network size of 250 nodes and a transaction size of 4 MB. For PoET, the maximum latency was \( \approx 20 \) ms with a network size of 200 nodes and a transaction size of 1 MB. For PoW and PoS, the maximum latency was 140 and 30 ms at network sizes of 250 and 200 nodes, respectively. An increase in the latency in PoW was observed due to the distribution of the computing power as the number of nodes increased. As mentioned in Section 3.1, our audit system cannot tolerate latency beyond 50 ms. Therefore, PoW becomes infeasible when the network size increases beyond 50 nodes (see Figure 5(b)). Moreover, PoW is highly energy intensive which leads to the wastage of processing power when the network size increases. Therefore, due to low scalability beyond 50 nodes and a high energy inefficiency, PoW is not a suitable protocol choice for scalable blockchain-based audit systems.

Surprisingly, latency in PBFT increased significantly as the number of nodes increased beyond 10 nodes. The maximum latency was 35 s with transaction and network size of 10 MB and 30 nodes, respectively. Beyond 30 nodes, the latency became more significant, and we omit results from our plot in Figure 5(e). In all the consensus protocols, the latency in transaction confirmation increased with the network size due to propagation latency among nodes hosted in different cities.

For transaction throughput reported in Figure 5(f), we use the transaction size of 4 MB and vary the network size from 5–250 nodes. We observed that up to a network size of 50 nodes, Clique achieved the maximum throughput, followed by PoET and PoS. With a network size of 5 nodes, Clique achieved a throughput of 8000 transactions per second. After 50 nodes, the transaction throughput of Clique decreased significantly and PoET and PoS became more dominant. In contrast, PBFT had a very low throughput, showing that the models proposed by the prior work in blockchain-based audit logs [3, 4] may not be feasible in a real-world implementation, and calling for a different algorithm choice.

In summary, our results show that for the general purpose, PoS, PoET, and Clique can be used for audit systems. Among them, Clique is the most suitable protocol when the network size is below 50 nodes. Once the network size exceeds 50 nodes, the audit system can benefit by utilizing one of the three protocols. We also noticed that PoW is not feasible beyond 50 nodes since the latency margins exceed the threshold requirements of our audit system. Even at a network size below 50 nodes, the latency is much higher than PoS, PoET, and Clique. In all settings, PBFT’s performance was significantly lower than the other protocols, making it the least suitable solution for the blockchain-based audit systems.

5 | RELATED WORK

Audit systems use blockchains to harden the security of audit logs. In this section, we review prior works on improving the audit log security. Additionally, we also succinctly review the key properties of blockchains that are usefully applied in the blockchain-based audit logs.
Audit Logs. Towards secure audit log systems, [24, 25] proposed a temper detection scheme that detects an audit log compromise. However, a key limitation in their design is that it only detects audit log tampering while lacking the capability of preventing the attacker from manipulating data during an attack. Similarly, another tamper detection scheme was proposed by [26]. They designed a notary-based audit manipulation detection for RDBMS audit logs. They introduced a check field in the data tuples, which when modified, generates a hash value of the modified data and sends it to the trusted notarization service. The notarization service helps in determining the authenticity of new updates.

Blockchains. Blockchains enables secure and tamper-proof data management in a distributed system where participants can have competing interests [14, 15]. A blockchain ledger is a linked list of blocks, and each block in the ledger consists of an ordered list of transactions. Each block is linked with the previous block through a one-way hash function. Due to the properties of hash functions, blockchains follow an append-only model in which data becomes immutable. In a multi-party settings, the participants agree on the ledger state by executing a consensus protocol. Broadly speaking, a consensus protocol is a set of instructions jointly executed to reach to the same set of outputs defined under the protocol specifications [7].

Blockchain-based audit system. Towards blockchain-based audit systems, [10] used the Ethereum blockchain to create encrypted audit logs for IoT devices. [28] developed a mechanism to store integrity proof digests of audit logs in bitcoin blockchains. [9] proposed a logging system which uses blockchains to maintain unforgable records of health care data exchanges across multiple countries. [13] used blockchains to improve the security of the immutable logs by publishing integrity proofs on the blockchain network.

[1] proposed a framework that integrated Online Transaction Processing System (OLTP) system’s audit logs with blockchains. Their system used nHibernate (Object Relation Mapping) events to generate audit log transactions, and PBFT consensus protocol to obtain the agreement from the network. In [4], a design optimization of [1] was proposed using a hierarchical design to reduce the system complexity and increase the throughput. However, their proposed system only supports audit applications that inherently follow a hierarchical system model. Moreover, even in the optimized version, PBFT is used for consensus. In our work, we show that PBFT is incapable of meeting the audit system requirements.

6 | CONCLUDING REMARKS

This paper evaluates the performance of a blockchain-based audit system using five different consensus protocols, namely PoW, PoS, PBFT, PoET, and Clique. We developed a testbed to set up a blockchain network of 250 nodes and implemented the consensus rules of each protocol. We evaluated the performance of each protocol using the same transaction and network size. Our results showed that PoET, PoS, and Clique are the useful consensus protocols for blockchain-based audit due to low latency and high throughput. Among them, Clique achieved the highest transaction throughput and minimum transaction confirmation latency. Moreover, our results indicated that due to high message complexity and low latency, PBFT may not be the optimal choice for blockchain-based audit logs. Our work opens a new direction in the performance evaluation of blockchain systems by showing trade-offs in the real-world performance of key blockchain consensus protocols.

ORCID

David Mohaisen ▪ https://orcid.org/0000-0003-3227-2505

REFERENCES

1. Ahmad, A. et al.: Towards blockchain-driven, secure and transparent audit logs. In: International Workshop on Distributed Ledger of Things (DLoT) in conjunction with MobiQuitous, 2018, https://doi.org/10.1145/3269788.3269985
2. Ahmad, A. et al.: Performance evaluation of consensus protocols in blockchain-based audit systems. In: International Conference on Information Networking, 2021, https://doi.org/10.1109/ICOIN48656.2020.9016513
3. Ahmad, A., et al.: Secure and transparent audit logs with BlockAudit. J. Netw. Comput. Appl. 145, (2019), https://doi.org/10.1016/j.jnca.2019.102406
4. Ahmad, A. et al.: Blocktrail: A scalable multichain solution for blockchain-based audit trails. In: IEEE International Conference on Communications, 2019, pp. 1–6. https://doi.org/10.1109/ICC.2019.8761448
5. Angelis, S.D.: Assessing security and performances of consensus algorithms for permissioned blockchains. CoRR (2018), http://arxiv.org/abs/1805.03490
6. Bano, S., Al-Bassam, M., Danezis, G.: The road to scalable blockchain designs. USENIXLogin 42(4), (2017), https://www.usenix.org/publications/login/winter2017/bano
7. Bano, S. et al.: Consensus in the age of blockchains. CoRR (2017), abs/1711.03936, http://arxiv.org/abs/1711.03936
8. Bentov, I., Gabizon, A., Mizrahi, A.: Cryptocurrencies without proof of work. In: International Workshop on Financial Cryptography and Data Security, 2016, pp. 142–157. https://doi.org/10.1007/978-3-662-53357-4_10
9. Castaldo, L., Cinque, V.: Blockchain-based logging for the cross-border exchange of e-health data in Europe. In: International Workshop on Security in Computer and Information Sciences ISCIS, 2018, pp. 46–56. https://doi.org/10.1007/978-3-319-95189-8_5
10. Cha, S., Yeh, K.: An ISO/IEC 15408-2 compliant security auditing system with blockchain technology. In: Conference on Communications and Network Security, 2018, pp. 1–2. https://doi.org/10.1109/CNS.2018.8433185
11. Chen, L. et al.: On security analysis of proof-of-elapsed-time (poet). In: International Symposium on Stabilization, Safety, and Security of Distributed Systems, 2017, pp. 282–297. https://doi.org/10.1007/978-3-319-69084-1_19
12. ClearVillage. Audit log application service. (2020), http://www.clearvillageinc.com/
13. Cucurull, J., Puiggalí, J.: Distributed immutabilization of secure logs. In: International Workshop on Security and Trust Management STM, 2016, pp. 122–137. https://doi.org/10.1007/978-3-319-46598-2_9
14. Holotiuk, F., Pisani, F., Moormann, J.: The impact of blockchain technology on business models in the payments industry. In: Towards Thought Leadership in Digital Transformation (2017), http://aisel.aisnet.org/wi2017/track09/paper6
15. Hyvärinen, H., Risius, M., Friis, G.: A blockchain-based approach towards overcoming financial fraud in public sector services. Business Inform. Syst. Eng. 59(6), 441–456 (2017), https://doi.org/10.1007/s12559-017-0502-4
16. King, S., Nadal, S.: Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. Self-published paper (2012)
17. Lee, K.H., Zhang, X., Xu, D.: Loggc: Garbage collecting audit log. In: Conference on Computer and Communications Security, 2013, pp. 1005–1016. http://doi.acm.org/10.1145/2508859.2516731
18. Margulies, J.: A developer's guide to audit logging. IEEE Secur. Privacy 13(3), 84–86 (2015) https://doi.org/10.1109/MSP.2015.50
19. Merkel, D.: Docker: lightweight linux containers for consistent development and deployment. Linux J. 2014(239), 2 (2014)
20. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. (2008), Online, https://bitcoin.org/bitcoin.pdf
21. Saad, M. et al.: Partitioning attacks on bitcoin: Colliding space, time, and logic. In: International Conference on Distributed Computing Systems, 2019, pp. 1175–1187. https://doi.org/10.1109/ICDCS.2019.00119
22. Saad, M. et al.: Exploring the attack surface of blockchain: A comprehensive survey. Commun. Surv. Tutorials 22(3), 1977–2008. https://doi.org/10.1109/COMST.2020.2975999
23. Sawtooth: Hyperledger sawtooth. (2019), https://bit.ly/3cm7o3X
24. Schneier, B., Kelsey, J.: Cryptographic support for secure logs on untrusted machines. Paper presented at the 7th USENIX Security Symposium, January 26, 1998, San Antonio, TX. https://bit.ly/38s8Sc9
25. Schneier, B., Kelsey, J.: Secure audit logs to support computer forensics. ACM Trans. Inf. Syst. Secur. 2(2), 159–176 (1999), http://doi.acm.org/10.1145/317087.317089
26. Snodgrass, R., Yao, S.S., Collberg, C.: Tamper detection in audit logs. In: International Conference on Very Large Data Bases, 2004, pp. 504–515. http://www.vldb.org/conf/2004/RS13P1.PDF
27. Sompolinsky, Y., Zohar, A.: Accelerating bitcoin’s transaction processing: fast money grows on trees, not chains. IACR Cryptol ePrint Arch. 2013, 881 (2013), http://eprint.iacr.org/2013/881
28. Sutton, A., Samavi, R.: Blockchain enabled privacy audit logs. In: International Semantic Web Conference, 2017, pp. 645–660. https://doi.org/10.1007/978-3-319-68288-4_38
29. Weippl, E.R. et al.: Conference on Computer and Communications Security (2016), http://dl.acm.org/citation.cfm?id=2976749

How to cite this article: Ahmad A, Alabduljabbar A, Saad M, Nyang D, Kim J, Mohaisen D. Empirically comparing the performance of blockchain’s consensus algorithms. IET Blockchain. 2021;1:56–64. https://doi.org/10.1049/blc2.12007