

Article

Benchmarking Consensus Protocols for High-Performance Permissioned Blockchain Systems

Muhammet Furkan Özara ¹, Akhan Akbulut ^{1,2}, Mustafa Kara ³, Muhammed Ali Aydın ⁴
and Hasan Hüseyin Balık ^{5,*}

¹ R&D Center of Next4biz, Sahrayicedit Mah, Pakpen Plaza, No: 40/4, Istanbul 34734, Türkiye; furkan.ozara@next4biz.com (M.F.Ö.); akhan.akbulut@next4biz.com or a.akbulut@iku.edu.tr (A.A.)

² Department of Computer Engineering, Istanbul Kültür University, Istanbul 34158, Türkiye

³ Department of Computer Engineering, Turkish Air Force Academy National Defence University, Istanbul 34149, Türkiye; mustafa.kara@msu.edu.tr

⁴ Department of Computer Engineering, Istanbul University-Cerrahpasa, Istanbul 34098, Türkiye; aydinali@iuc.edu.tr

⁵ Department of Engineering and Natural Sciences, Istanbul Atlas University, Istanbul 34408, Türkiye

* Correspondence: hasan.balik@atlas.edu.tr; Tel.: +90-850-450-3439

Abstract

Blockchain applications rely on consensus protocols to maintain security, integrity, and coordination in decentralized environments while balancing performance, scalability, and resource cost. This study presents a comparative evaluation of four Hyperledger Besu consensus algorithms, namely Ethash, Clique, QBFT, and IBFT 2.0, executed under identical hardware, network topology, and genesis configurations to quantify latency, throughput, and system overhead. A distributed burst workload is employed to simulate high-intensity transaction conditions. Within this framework, sender and receiver accounts are automatically generated and funded, and 5000 transactions are submitted in parallel. Locally recorded millisecond-precision submission timestamps are aligned with second-resolution on-chain block timestamps to compute inclusion latency percentiles and end-to-end burst throughput. System and blockchain metrics, including block interval, transaction pool backlog, and block creation time, are collected through Prometheus over defined intervals and synchronized with transaction traces for time-series analysis. Across ten repeated runs per protocol, Clique achieves the lowest latency of approximately two seconds and the highest throughput of approximately 190 transactions per second. QBFT and IBFT 2.0 demonstrate stable and periodic performance near 110 transactions per second with low variance. In contrast, Ethash exhibits highly variable, high-variance and strongly right-skewed latencies, minimal throughput, and substantially higher energy consumption and disk input/output peaks. CPU, memory, and network traffic profiles further expose distinct operational trade-offs relevant to deployment scenarios. Statistical analyses using the Kruskal–Wallis and Dunn’s post hoc tests confirm significant differences among the protocols with large effect sizes. The proposed framework provides a rigorous and systematically documented methodology for comparative consensus performance evaluation in permissioned blockchain environments.

Keywords: blockchain; consensus algorithms; performance evaluation; Hyperledger Besu



Received: 4 August 2026

Revised: 18 September 2026

Accepted: 23 September 2026

Published: 28 September 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

The fundamental components of blockchain technology include cryptographic hashing, a distributed ledger, and smart contracts. The consensus protocol is the crucial in-

gradient that ensures the secure and sustainable operation of these components, allowing all nodes in the blockchain network to agree on a common transaction history without requiring a central authority [1,2]. Consequently, the network's security, integrity, and continuity are maintained [3]. Moreover, it serves as the essential element that guarantees the integrity and authenticity of the blockchain [4]. Consensus algorithms enable nodes in distributed systems to reach agreement on a single value or transaction set, thereby enhancing network scalability and transaction efficiency. The reduction in block creation time and the expedited verification of transactions have spurred the development of several consensus algorithms. Selecting a suitable consensus algorithm is a critical decision that directly influences the overall functioning of the blockchain. This technique addresses the trust issue in dispersed networks while maintaining the principle of decentralization [5].

The consensus mechanism comprises five fundamental components within blockchain architecture. The components are block proposing, information transmission, block validation, block finalization, and the incentive mechanism, respectively [6]. These stages encompass operations such as generating new blocks, disseminating transactions within the network, verifying block validity, endorsing verified blocks, and incentivizing honest participation. Given that each blockchain application possesses distinct requirements and priorities, it is infeasible to implement a universal consensus mechanism across all systems. Consequently, it is essential to evaluate existing algorithms regarding performance, security, and scalability to advance the technology and broaden application domains [7].

This paper examines the performance evaluation of consensus algorithms on blockchain platforms. Our objective is not to analyze all blockchain technologies comprehensively. We focus on consensus processes to critically assess blockchain technology. This subject is crucial for researchers to enhance consensus algorithms and for developers to comprehend the constraints of platforms [8]. Furthermore, we intend to establish a methodologically coherent evaluation framework, such as the Benchmarking Hyperledger Fabric performance evaluation model, which facilitates quantitative, systematic, and comparable analysis of blockchain-based consensus algorithms in an academic context. Consequently, a dependable infrastructure has been established for academic researchers and developers to perform systematic performance analysis.

The consensus algorithms selected for this study were Ethash (Proof of Work), QBFT (Quorum Byzantine Fault Tolerance), IBFT 2.0 (Istanbul BFT), and Clique (Proof of Authority), each configured within a simulated Hyperledger Besu-based private blockchain environment [9]. The selected algorithms represent different consensus design paradigms, including Proof of Work, Proof of Authority, and Byzantine fault-tolerant approaches. While the distinction between Proof-based and BFT-based mechanisms provides a useful high-level architectural grouping, consensus protocols cannot be fully characterized along this dimension alone. They also differ in validator admission, security assumptions and resources, agreement architecture, governance, finality, communication and computational requirements, and application-specific performance objectives [10,11]. In particular, QBFT and IBFT 2.0 are tailored to permissioned blockchain environments in which Byzantine Fault Tolerance and low-latency agreement are important design objectives [12]. Accordingly, the protocols evaluated in this study are interpreted not only according to their consensus family but also through their distinct operational and architectural characteristics.

Comparative analysis of these algorithms enables a comprehensive assessment of consensus protocols across performance, scalability, security, decentralization, and energy efficiency. The structure of this paper is as follows: Section 2 reviews related research. Section 3 details the conceptual foundations of the selected consensus algorithms and the technical mechanisms employed. Section 4 outlines the performance evaluation methodology, measurement approach, and experimental design framework. Section 5 presents

the results of the experimental study. Section 6 discusses and evaluates these findings. Section 7 concludes the paper and summarizes the main contributions.

2. Related Work

Consensus algorithms ensure the secure and stable operation of blockchain technology [13]. They determine the rules that the nodes in the blockchain must follow and help the network to be consistent. Extensive research has been conducted to develop different consensus mechanisms such as Proof-of-X, Byzantine Fault Tolerance (BFT), and DAG-based approaches [14,15]. Among the most common consensus algorithms are Proof of Work (PoW), BFT-based algorithms, and Proof of Authority (PoA) [16]. In private chain scenarios, some methods use fewer resources, while others are more efficient in terms of speed and security [17]. However, selecting an appropriate consensus algorithm for a given application remains challenging [18]. Although approximately 40 consensus algorithms have been developed in the field of blockchain, there is still a clear mismatch between the existing proposals and the algorithms that meet real needs [19]. Recent studies have further approached consensus selection as a multidimensional decision problem rather than as a choice based solely on consensus family or individual performance metrics. Svarcmajer et al. [10] organize the consensus design space across multiple architectural and operational dimensions and formulate protocol selection as a requirement-driven decision process. Similarly, Kumar and Yadav [11] combine multi-attribute decision-making with machine learning to support application-specific consensus-mechanism selection. These approaches reinforce the view that protocol suitability depends on the joint consideration of architectural properties, trust and security assumptions, performance requirements, and application context.

Ref. [20] presents a thorough comparative analysis of PoW, Proof of Stake (PoS), and Algorand's Pure PoS techniques. They evaluate different consensus techniques based on transaction throughput, latency, and network scalability. The review emphasizes low throughput and high latency of PoW, the restricted finality framework of PoS, and Pure PoS's claim to resolve all three dimensions of the blockchain trilemma concurrently. Ref. [21] assesses the QBFT, IBFT 2.0, and Clique consensus processes in Hyperledger Besu (HB) and observes that HB exhibits lower transaction volume than Hyperledger Fabric's RAFT mechanism. Nonetheless, BFT-based techniques provide superior security. These papers collectively clarify the tradeoffs between performance and security in consensus algorithms. Ref. [22] provides a cohesive permissioned blockchain architecture that employs the IBFT consensus algorithm on Hyperledger Besu and juxtaposes IBFT with PoW, Ethereum Clique, and the RAFT algorithm of Hyperledger Fabric. The experimental results show that IBFT provides up to 15 times lower latency and approximately 2 times higher transaction volume compared with PoW. In addition, it has been seen to offer better scalability and a lower error rate compared with RAFT and Clique. Yadav et al. [19] compare basic consensus mechanisms such as PoW, PoS, BFT, and hybrid protocols in terms of performance, scalability, security, and fault tolerance. By revealing the strengths and weaknesses of different consensus mechanisms with a holistic model, they provide a technical reference for the selection of an appropriate consensus in blockchain applications.

More recent studies further demonstrate that consensus-performance outcomes depend strongly on the experimental configuration and should therefore be interpreted within the specific workload, network, and hardware conditions under which they are obtained. Ferone and Verrilli [23] evaluated Clique, IBFT 2.0, and QBFT in a Hyperledger Besu environment under different transaction loads and validator configurations, showing that protocol behavior may vary as operating conditions change. Yalnalli et al. [24] similarly compared QBFT, IBFT 2.0, and Clique within Hyperledger Besu under high-volume work-

loads, further illustrating the influence of workload characteristics on latency, throughput, and resource utilization. From a hardware-oriented perspective, Kurisaka et al. [25] analyzed Ethereum consensus mechanisms on resource-constrained devices and showed that computational and energy limitations can substantially affect observed consensus performance. Jung et al. [26] examined the impact of horizontal and vertical resource scaling on permissioned-blockchain performance, while Rao et al. [27] provided a broader analysis of the factors governing blockchain scalability across different architectural settings. Taken together, these studies indicate that differences in protocol rankings across the literature should not necessarily be regarded as contradictory; rather, they reflect the sensitivity of consensus performance to workload design, validator configuration, hardware resources, and application context. Accordingly, the results reported in the present study are interpreted as a controlled characterization under the evaluated four-node and burst-workload configuration rather than as a universal ranking of the considered protocols. Some studies that compare consensus algorithms either did not present an experimental analysis and explained them completely theoretically [1,8,15,28], or performed an experimental comparison on a specific problem solution [16,29–31].

Against this background, the contribution of the present study lies in providing a controlled and multidimensional comparative characterization rather than proposing a new consensus mechanism or a universal protocol ranking. The four consensus configurations are evaluated under an identical hardware environment, fixed four-node topology, and common 5000-transaction burst workload, while transaction-level latency and throughput are examined together with system-level resource telemetry. In addition, each protocol is evaluated over ten independent runs, and the resulting distributions are analyzed using confidence intervals, coefficients of variation, non-parametric significance tests, post hoc comparisons, and effect-size measures. This combination enables the observed protocol differences to be interpreted not only in terms of absolute performance but also in terms of variability and system-level operational behavior.

3. Conceptual and Technical Examination

This section addresses both the conceptual underpinning of consensus algorithms and the technical specifics at the protocol level. The initial section elucidates the principles of blockchain architecture and consensus techniques. The second part provides a detailed examination of the operational processes, message complexity, fault tolerance, and performance characteristics of the Ethash, QBFT, IBFT 2.0, and Clique algorithms. This comprehensive method seeks to assess the theoretical frameworks of consensus algorithms alongside their practical implementation attributes.

3.1. Blockchain Consensus Algorithms

Blockchain is a list of records that stores all transactions made on the network and replicates them across every node that participates in the network [32]. When a new transaction occurs on the blockchain, some nodes verify its validity. Valid transactions are added to a block, which is then linked to the chain [3]. All nodes in the network become aware of the updates in the chain and update their own copies. However, if more than one node publishes a block at the same time, conflicts may occur. Therefore, consensus algorithms are used among nodes to update the chain consistently [33]. Blocks are stored in the record list. A decentralized, distributed system is created to protect the record list [15]. Each block creates a timestamp and a link to the previous block. As the link grows, the system becomes more secure [34]. Thus, once data are recorded in the system, it becomes difficult for an attacker to change them. The distributed system enables the creation of a secure, transparent, and immutable ledger managed by a network-wide group rather than

a central authority [35]. The blockchain architecture generally consists of four main layers: consensus, data model, execution, and application. Consensus is the process of reaching agreement on the current state of the blockchain [36]. This layer ensures node agreement and manages the creation of new blocks. The data model layer is responsible for block structures, transactions, and recorded data. The execution layer supports the operation of smart contracts and runtime environments. The application layer hosts blockchain-based applications [5].

Public, consortium, and private blockchains are distinguished by the methods by which participants reach consensus on the network. In public blockchains, all miners participate in the consensus process. In consortium blockchains, this process is performed by a specific group of nodes, whereas in private blockchains it is handled by a single organization [18]. When evaluating consensus algorithms, many criteria such as performance, efficiency, and security are considered together. In this evaluation, transaction efficiency (throughput), mining profitability, degree of decentralization, and security vulnerabilities are the main criteria. Transaction efficiency is measured with metrics such as transactions per second (TPS), block creation time, and validation latency. Mining profitability depends on energy consumption, transaction fees, and the need for specialized hardware. Decentralization is evaluated with network governance, permission model, and trust-based structures. Security is measured by resilience against threats such as double-spending, 51% attack, and Sybil attack. This multidimensional approach helps to systematically analyze the strengths and weaknesses of different consensus algorithms. Thus, the most appropriate consensus algorithm can be selected for specific applications. PoW is a mechanism that ensures the accuracy of transactions through a “Proof-of-Work” method based on computing power. Nodes attempt to produce a new block by solving a cryptographic puzzle. The miner who solves the puzzle first transmits the block to the network, and after verification, it is added to the chain. Miners verify transactions and earn rewards in this process. The security of PoW comes from the structure that makes it difficult to modify the hash chain. However, it is vulnerable to the 51% attack. It also has disadvantages, such as high energy consumption, high costs, centralization risk from mining pools, and network latencies that lead to forks. Nevertheless, because it has been tested for a long time, it is still one of the most common and reliable consensus mechanisms [37]. PoA has a structure in which authorized validators create blocks with identity-based “Proof of Authority” [38]. Given these characteristics, both algorithms fall into the Proof-based consensus category.

QBFT is a voting mechanism based on Byzantine Fault Tolerance (BFT), in which validators reach agreement through quorum-based voting rather than unanimous approval. Similarly, IBFT 2.0 is an advanced BFT consensus protocol based on voting [9]. This algorithm is used in private Ethereum networks. Nodes vote to accept the new block, thereby becoming resistant to malicious nodes. With these characteristics, QBFT and IBFT 2.0 fall into the category of Voting-based consensus algorithms [8,39].

Byzantine Fault Tolerance (BFT) is a consensus mechanism in distributed systems and blockchain technologies that ensure the network continues to function correctly even when some nodes behave maliciously or incorrectly. The foundation of BFT is based on a classic distributed systems problem known as the Byzantine Generals Problem. This problem aims to have the system reach a correct agreement even if some number of nodes (or generals) betray. Theoretically, if the system has fewer than $3t + 1$ nodes, t faulty nodes cannot be tolerated. BFT essentially operates in two stages: pre-commit and commit. The selected leader proposes a block; nodes verify this block and send a pre-commit message. When at least $2/3$ of the network approves, the commit stage begins, and the block is added to the chain. This process maintains the security (safety), liveness, and integrity of the system. BFT protocols were developed especially to increase the reliability of systems in

asynchronous networks. Today, different variants of BFT (such as PBFT, IBFT, QBFT) are used in blockchain networks. Thus, leader selection, messaging complexity, and block validation processes are optimized [20]. Table 1 presents the typical characteristics of the Ethash (PoW), QBFT, IBFT 2.0, and Clique (PoA) consensus algorithms evaluated in the study.

Table 1. Comparative Analysis of Consensus Algorithm Characteristics.

Properties	Ethash	QBFT	IBFT 2.0	Clique
Blockchain Type	Public	Permissioned	Permissioned	Hybrid
Finality	Probabilistic	Deterministic	Deterministic	Probabilistic
Participation Cost	High	Low	Low	Low
Expected Scalability Characteristics	High	Medium	Medium	High
Trusted Model	Untrusted	Semi-trusted	Semi-trusted	Semi-trusted
Algorithm Type	Proof-based	BFT-based	BFT-based	Proof-based (Authority-based)
Fault Tolerance	Adversarial hash power < 50%	<33% Validator	<33% Validator	<51% Authorized Node

3.2. Technical Analysis of the Selected Consensus Algorithms

The four selected consensus algorithms represent distinct consensus design paradigms that were supported by the Hyperledger Besu version used in the experimental environment. Their inclusion enables a controlled comparison of Proof of Work, Proof of Authority, and Byzantine fault-tolerant consensus mechanisms under identical experimental conditions.

3.2.1. Ethash

PoW is designed to preserve the integrity of a distributed ledger and to prevent double-spending attacks among untrusted nodes without relying on a central authority. It operates on a computation-based competitive principle and enables all nodes in the network to reach agreement on the same ledger state. PoW consists of three sub-protocols: (1) normal-operation, (2) difficulty-adjustment, and (3) chain-selection. In normal-operation, each miner constructs a candidate block by collecting valid transactions from the network. Miners then perform an intensive computational process to find a nonce that yields a block hash below the network's target. This process requires substantial energy, which is the reason for the name "Proof of Work". The first miner to obtain a valid block hash broadcasts the block to the network, and after other nodes verify the hash, transaction integrity, and block reference, the block is appended to the chain. The difficulty-adjustment protocol periodically updates the difficulty level in order to keep the block-generation interval statistically stable. When blocks are produced faster than the target interval, the difficulty increases. When blocks are produced more slowly, the difficulty decreases. In this way, the block time (for example, ≈ 10 min in Bitcoin) remains balanced even if the total hash power of the network fluctuates. This mechanism also defines the cost threshold for an adversary because a greater difficulty level makes it significantly more challenging for malicious miners to gain control over the network. The chain-selection protocol is used to resolve forks that occur when more than one valid block is generated at the same time. PoW adopts the rule that the chain with the greatest accumulated work is considered valid. For this reason, chain-selection is not deterministic and instead relies on probabilistic finality. As

additional blocks are appended on top of a given block, the probability of that block being reversed decreases exponentially. The message complexity of PoW is $O(N)$ because nodes communicate only for block propagation and validation. However, the computational complexity of the protocol is extremely high due to the competitive nature of hashing in the mining process. From a security perspective, PoW relies on a probabilistic consensus model based on an honest-majority hash-power assumption. The protocol becomes vulnerable when an adversary controls at least half of the effective hash power of the network. This assumption can be expressed by the following inequality:

$$H_{\text{adv}} < \frac{H_{\text{total}}}{2} \quad (1)$$

In Equation (1), H_{adv} denotes the effective hash power controlled by adversarial participants, while H_{total} represents the total effective hash power of the network. Accordingly, the relevant security assumption is based on the distribution of computational power rather than on the number of miners or nodes. Limitations of PoW in terms of energy consumption, transaction capacity, and scalability have led to the development of alternative protocols such as PoS, PoA, and Proof of Elapsed Time (PoET) in subsequent years [20]. Nonetheless, PoW is still regarded as the most mature and longest field-tested consensus protocol in terms of security. To address the significant energy consumption and limited transaction throughput inherent in PoW, several key protocols have emerged, including Bitcoin-NG, Ethereum's Ethash, Litecoin's Scrypt, Monero's RandomX, and Kadena's Chainweb [40]. These protocols aim to maintain PoW's fundamental security properties while resolving its core challenges.

Ethash is a specialized PoW algorithm developed for Ethereum. In this study, the mining difficulty of Ethash is fixed at "0x1".

So, we eliminate the adaptive difficulty mechanism and mining competition dynamics that normally exist in public PoW networks. The motivation for disabling PoW feedback mechanisms is to ensure a fair and controlled comparison with non-PoW consensus protocols such as QBFT, IBFT 2.0, and Clique. These protocols finalize blocks deterministically and do not incorporate computational competition or hash-rate-dependent variability. Without fixing the difficulty, Ethash block-generation time would fluctuate significantly based on instantaneous CPU availability, leading to an inconsistent experimental baseline. By setting a static difficulty value, block generation remains more stable and consistent across repeated experimental runs, enabling meaningful evaluation of performance differences without mining dynamics dominating the results. As a consequence, the Ethash measurements obtained in this environment should be interpreted as controlled private-network benchmark results and should not be generalized directly to public PoW networks, where hash-rate competition, difficulty-adjustment, miner population, and network propagation conditions are dynamic.

3.2.2. QBFT

QBFT is a BFT consensus protocol designed particularly for enterprise permissioned blockchain networks. It has emerged to reduce latency in block production in private networks, to manage dynamic validator sets, and to provide high security with low communication cost [12]. QBFT is essentially derived from the PBFT and IBFT protocols and includes optimizations addressing their shortcomings. The QBFT protocol may consist of three main sub-protocols: (1) normal-operation (block proposing and approving), (2) validator-set management (validator set update/dynamic validator change), and (3) view-change and fault recovery.

Normal-operation: During the normal-operation phase, validator nodes are grouped under a designated block proposer. The proposing validator prepares the candidate block and broadcasts it to the other validators. Consensus on the proposed block is then achieved through prepare and commit votes. The process is completed through the exchange of messages among validators. In QBFT, once block addition is completed, all honest validators reach agreement on the same block, and the block is appended to the chain. In the opposite case, for example, when the proposer fails or behaves incorrectly, the view-change mechanism is activated.

Validator Set Management (Dynamic Validator Management): One of the distinguishing features of QBFT is the ability to add or remove validators dynamically through a block-header smart contract or a dedicated management contract. For example, the initial validators are defined in the `extraData` field at the genesis of the blockchain, and in later transition blocks, the validator set can be updated through the contract. This management supports validator changes according to the organizational needs of the network and adds flexibility to network-governance processes.

View-change and Fault Recovery: When the proposing validator fails or becomes non-operational, QBFT ensures a safe transition to a new proposer. To select the new proposer, view-change messages are collected from a sufficient number of validators. This phase supports the fault-tolerant operation of the system. In addition, recovery mechanisms can be defined for changes in the validator set or for removed nodes. QBFT provides immediate finality, meaning that no forks occur and a block is considered final once it is appended. In this paper, the following properties are considered in order;

- The minimum validator count for QBFT is specified as 4. The aim is to tolerate at least one faulty validator within the system.
- The protocol is designed to operate in partially synchronous network models.
- The fault-tolerance requirement follows the classical BFT thresholds. If $N \geq 3f + 1$ validators are present, the system continues to operate against at most f Byzantine validators.
- In the designed algorithm, the message complexity is at the level of $O(N^2)$. This communication cost arises due to voting processes such as *prepare* and *commit* among all validators.
- Other characteristic advantages of the QBFT algorithm include the absence of forks, fast block finality, and support for dynamic validator changes. However, as the number of validators increases, latency and communication cost may increase. In addition, scalability limitations exist.

In our comparative experiment, the QBFT protocol is configured with a block-production interval of 2 s and a request timeout of 4 s. The `epochlength` parameter in the genesis file (set to 30,000) dictates the interval for validator rotation. This setup allows the system to systematically regulate validator transitions during prolonged operations. QBFT maintains consensus integrity through its BFT architecture, even in the face of a specified number of node failures or hostile activity within the network. This characteristic makes QBFT an appropriate consensus algorithm for private blockchain setups that require high reliability.

3.2.3. IBFT 2.0

Quorum developed this protocol as an enhanced version of the Practical Byzantine Fault Tolerance (PBFT) algorithm. In this system, block production occurs sequentially across predefined validator nodes, reducing transaction confirmation time and facilitating faster consensus within the network. The protocol employs a PoA-based BFT consensus mechanism, specifically designed for private and permissioned blockchain networks [22]. IBFT 2.0 represents an improved iteration intended to address the safety and liveness

limitations of its predecessor, the IBFT protocol. Its primary objective is to prevent forking by ensuring immediate finality in networks managed by validators with static identities. IBFT 2.0 comprises three main sub-protocols: normal-operation (block proposing and approval), validator set management (dynamic validator set updates), and viewchange and fault recovery.

Normal-operation: During the normal-operation phase, a proposer is selected from among the validators, and this proposer prepares the candidate block and transmits it to the other validators. The validators approve the block through pre-prepare, prepare, and commit messages. When the required signatures for the candidate block are collected, the block is appended to the chain. At this stage, all honest validators have reached agreement on the same block. IBFT 2.0 is optimized specifically to reach a decision within three message latencies.

Validator Set Management: The second sub-protocol of IBFT 2.0 enables the validator group to be updated dynamically. Within the scope of network governance, a new validator may be added or an existing validator may be removed. These changes are recorded in block headers through voting. For example, a majority (for instance, $2/3$) approval is required for a validator change.

View-change and Fault Recovery: If the proposing validator or the current block-production round fails (mandatory timeout, proposer failure, etc.), IBFT 2.0 triggers the view-change mechanism. In this process, the validators exchange messages to select a new proposer, and the network returns to normal operation. Through this sub-protocol, the system can operate safely and maintain liveness even in partially synchronous network models.

IBFT 2.0 is based on the classical BFT threshold condition $N \geq 3f + 1$ [41]. Here, f denotes the maximum number of faulty validators. The message complexity is at the level of $O(N^2)$ because mutual approval messages (*prepare* and *commit*) are exchanged among validators. The protocol provides immediate finality, meaning that once a block is added, the probability of it being reverted is practically negligible. In terms of security, the system can operate correctly in the presence of up to f faulty validators. As long as the large majority of the validator group (for example, $2f + 1$) behaves honestly, the protocol functions without issues. IBFT 2.0 addresses the safety and liveness vulnerabilities present in the earlier version of the IBFT protocol.

For our comparative experiment, the IBFT 2.0 protocol is defined with a block time of 2 s and a timeout parameter of 10 s. In the test environment, intentional time offsets are created among the nodes by using the timeshift agent mechanism, and in this way, the sensitivity of the IBFT 2.0 algorithm to time synchronization is evaluated. This method enables the analysis of IBFT's behavior under network latencies and synchronization drifts and allows the consistent performance of the system to be measured.

In BFT consensus protocols, system performance and communication overhead are inherently influenced by the size of the validator set. Due to their $O(N^2)$ message complexity, increasing the number of validators results in higher messaging costs, increased latency, and reduced throughput. Experimental evaluations of BFT-based systems face a fundamental methodological choice between stress-testing scalability through large validator configurations and prioritizing controlled conditions that enable fair and interpretable performance comparison. The experimental study explicitly prioritizes controlled comparison over stress-testing individual protocols. By fixing the validator count across QBFT and IBFT 2.0, the analysis isolates the intrinsic performance characteristics of different consensus families under identical conditions. This design choice ensures that observed differences in throughput, latency, and resource utilization can be attributed primarily to the consensus mechanisms themselves, rather than to secondary effects introduced by varying

validator participation or well-known scalability degradation inherent to BFT messaging. Accordingly, the QBFT and IBFT 2.0 measurements reported in this study characterize their behavior under the fixed validator configuration and should not be interpreted as scalability limits for larger validator sets.

3.2.4. Clique

It is a PoA consensus protocol developed by the Ethereum community and initially introduced with the Go Ethereum (Geth) client [1]. It is designed to provide high transaction throughput, low energy consumption, and fast block confirmation with probabilistic finality in permissioned or private networks. However, Clique does not provide instant finality and instead offers probabilistic finality [42]. Clique adopts a hybrid leader-based structure that eliminates the high energy cost of the PoW protocol while reducing the high message complexity observed in BFT protocols. Clique protocol consists of three subcomponents: normal-operation (block production and approval), signer management, and fork-resolution.

Normal-operation: During normal-operation, the authorized signers in the network produce blocks in a predefined order. In each block-production round, a leader signer is determined. The signer collects valid transactions, creates a new block, and broadcasts it to the network after adding its signature. The other authorized signers verify the validity and authenticity of the received block according to the protocol rules; unlike BFT-based consensus mechanisms, Clique does not require a multi-round quorum-certificate assembly for each block.

$$leader = signer[blockNumber \bmod N] \quad (2)$$

In Equation (2), N denotes the total number of signers. Through this mechanism, the block-production order becomes predictable, enabling low-latency consensus. Each signer adheres to timing constraints determined by the `minPeriod` (minimum time interval) and `epochlength` parameters during block production. These parameters prevent conflicts and block-race conditions within the network. If a signer fails to produce a block or behaves incorrectly, the next signer in the sequence takes over in the following round.

Signer Management: Clique supports the dynamic updating of the signer set. The addition or removal of a signer is performed through on-chain voting. Each signer includes its vote in the block header. When the voting results reach the majority of the network, the change takes effect. This governance mechanism enables the protocol to operate in a decentralized yet controlled (federated) structure. In addition, at epoch blocks, the signer list is periodically reset, and the updated list is incorporated into the chain state. In this way, past voting records are cleared and the administrative consistency of the system is preserved.

Fork-Resolution: Clique anticipates the possibility of occasional forks even in an open authority-network scenario. In such cases, the remaining nodes in the network consider the chain with the highest number of signatures as valid. The selection between conflicting blocks is made based on block weight and signer majority. Through this mechanism, Clique provides probabilistic finality rather than deterministic finality, although the practical fork rate remains very low.

Clique has a message complexity at the level of $O(N)$. Block propagation and signature verification are required, while no multi-round quorum-based voting is performed for each block. For this reason, it does not generate $O(N^2)$ messaging overhead seen in protocols such as PBFT or IBFT. The protocol demonstrates high performance in partially synchronous networks, and the transaction-confirmation time is fast.

From a fault-tolerance perspective, Clique relies on the majority-honesty assumption. Even if at most $\lfloor N/2 \rfloor - 1$ of the N signers behave maliciously, the integrity of the network is preserved. Although it does not provide a BFT guarantee like IBFT 2.0, Clique historically provided a lightweight PoA alternative for private and permissioned Ethereum networks due to its low communication overhead and low-latency block production. It was supported by Hyperledger Besu v25.7.0, which was used in the present experiments. However, Clique support was subsequently removed from Besu beginning with v26.4.0. Accordingly, Clique is included in this study as a comparative PoA reference within the evaluated software environment rather than as a currently supported Besu deployment option. The fault-tolerance condition of Clique can be summarized as follows:

$$f < \frac{N}{2} \quad (3)$$

In Equation (3), f denotes the number of malicious signers. Additionally, Clique is a deterministic PoA mechanism with a fixed validator set. In our comparative experiment, the block time is set to 2 s. Through the RPC interfaces (API), the activation of the Clique module enables the monitoring of block signatures and validator changes. The deterministic structure of Clique provides predictability in block production and offers high stability against network latencies.

4. Performance Evaluation Methodology and Measurement Approach

In this section, the test approach used to evaluate the system's performance characteristics, the measurement criteria, and the data collection process are explained in detail. By jointly considering the burst benchmark scenarios, the interpretation of the employed metrics, and the integration of the Prometheus-based monitoring infrastructure, a methodological framework is presented that supports consistent and systematically controlled performance evaluation.

4.1. Burst Benchmark

This study focuses on analyzing the behavior of different consensus algorithms under high-intensity ("burst") transaction loads. For this purpose, the `distributed_burst_benchmark.ipynb` scenario is developed and executed simultaneously on four different algorithms (Ethereum, QBFT, IBFT 2.0, and Clique). Algorithm 1 is designed to measure the performance of a consensus network under sudden and intensive transaction loads. The Burst Benchmark calculates the inclusion latency of transactions and the number of processed transactions per second (throughput) in scenarios where a large number of transactions are sent to the consensus network simultaneously by multiple senders. In this way, the transaction success and latency behavior of the consensus network under burst load are analyzed statistically. The workload is intentionally restricted to a fixed high-intensity burst scenario to maintain comparability across protocols. Consequently, the results do not characterize steady-state traffic, mixed transaction arrival patterns, or dynamically varying workloads, which may produce different queuing and block-production behavior.

Algorithm 1 Burst Benchmark Algorithm**Require:** *BURST_SIZE*, *NUM_SENDERS*, *TX_RECEIPT_TIMEOUT*, *POLL_INTERVAL***Ensure:** Summary metrics of inclusion latency and throughput

```

1: Divide total transactions among senders
2:  $per\_sender \leftarrow \lceil BURST\_SIZE / NUM\_SENDERS \rceil$ 
3:  $total\_sent \leftarrow 0$ 
4: for all senders in parallel do
5:    $nonce \leftarrow get\_current\_nonce(sender)$ 
6:   for  $i = 1$  to  $per\_sender$  do
7:     if  $total\_sent \geq BURST\_SIZE$  then
8:       break
9:     end if
10:     $tx \leftarrow construct\_transaction(sender, recipient[i], nonce)$ 
11:     $submitted\_time\_ms \leftarrow current\_time\_ms()$ 
12:     $tx\_hash \leftarrow broadcast(tx)$ 
13:     $submit\_time\_map[tx\_hash] \leftarrow submitted\_time\_ms$ 
14:     $nonce \leftarrow nonce + 1$ 
15:     $total\_sent \leftarrow total\_sent + 1$ 
16:  end for
17: end for
18: for all  $tx\_hash$  in parallel do
19:   Wait until receipt exists or timeout expires
20:   if receipt exists then
21:      $block\_time\_ms \leftarrow get\_block\_timestamp(receipt.blockNumber) \times 1000$ 
22:      $latency\_ms \leftarrow block\_time\_ms - submit\_time\_map[tx\_hash]$ 
23:     Append  $latency\_ms$  to latencies
24:     Append  $block\_time\_ms$  to block_times
25:   else
26:     Mark transaction as failed
27:   end if
28: end for
29: Sort latencies
30: Compute min, max, avg, p50, p90, p95, p99
31: Compute throughput:
32:  $approx\_tps \leftarrow successes / ((max(block\_times) - min(block\_times)) / 1000)$ 
33: return summary metrics

```

4.1.1. Dynamic Creation and Funding of Accounts

At the beginning of the experimental process, accounts with sender and receiver roles for submitting transactions to the consensus network are generated dynamically. The sender accounts are prepared by automatically transferring balance from a prefunded master account defined in the genesis configuration (for example, privateKey: 8f2a55...). The transferred balance is scaled according to the planned burst size and ensures that each sender account has sufficient funds to execute transactions. The funding process formula is:

$$balance_{sender_i} = \frac{burst_size \times value_per_tx}{num_senders} \quad (4)$$

Through the created mechanism, each sender account has tracked its own nonce sequence independently and the total transaction load has been distributed across the network in parallel. In this way, both the transaction-submission intensity and the load distribution among nodes have been balanced under the test conditions.

4.1.2. Transaction Submission and Recording of Timestamps

In each test scenario, a set of target receiver accounts corresponding to the specified burst size (for example, 5000 transactions) is generated dynamically. The sender accounts

transmit transactions to the respective receivers through a round-robin algorithm. The transactions are signed through a Web3 interface and delivered to the corresponding nodes via the *eth_sendRawTransaction* call. As shown in Algorithm 2, millisecond-precision timestamps are recorded during each transaction submission.

Algorithm 2 Millisecond Timestamp Function

```

1: function NOW_MS
2:   current_time_seconds  $\leftarrow$  Get_Current_Time_In_Seconds()
3:   current_time_milliseconds  $\leftarrow$  current_time_seconds  $\times$  1000
4:   return [current_time_milliseconds]
5: end function
  
```

Using the collected time-series data, the following metrics are calculated:

- *submitted_ms*: Moment the transaction is sent to the node.
- *block_time_ms*: Timestamp of the block in which the transaction is included.
- *inclusion_ms* = *block_time_ms* − *submitted_ms*: Inclusion latency of the transaction.

These measurements form the basis for the comparative analysis of the consensus algorithms in terms of transaction-confirmation times, block-production speeds, and network-response latencies. The locally submitted transaction timestamp is accurate up to milliseconds, but the on-chain block timestamp is available at the seconds level. The factor of 1000 applied to the latter is used only for representing both timestamps in the same units without increasing the precision of the block timestamp. Hence, each of the individual latency measurements contains the quantization uncertainty of the block timestamp, which is an important issue in case of latencies measured at the order of one second.

4.2. Measurement Metrics

Within the scope of the experimental evaluation, the results of tests for each consensus algorithm are analyzed using key performance indicators, including transaction latency, inclusion success, and per-second transaction throughput. For all transactions, the following metrics are calculated:

- *total_txs*: Total number of submitted transactions.
- *success/fail*: Number of successfully processed transactions and the number of failed transactions.
- P50: Median inclusion latency, i.e., the latency value at or below which 50% of the transactions are included in the chain.
- P95: Latency value at which 95% of the transactions are included (behavior under load).
- Max: Inclusion time of the slowest transaction (indicates queuing effects).
- *Approx_{tps}*: Average per-second transaction throughput.
- *Peak_{tps}*: Highest instantaneous TPS measured during the test.

These metrics enable a quantitative evaluation of the system's behavior under the defined high-intensity burst workload.

4.3. Prometheus Integration and Data Collection Process

In the experimental monitoring infrastructure, the Prometheus service collects metric data from all nodes at 2 s intervals. Among the collected data, the metrics *blockchain_block_interval_seconds*, *txpool_pending_transactions*, and *consensus_block_creation_time_seconds* particularly reflect the block-production dynamics of the system, the rate of transaction accumulation, and the variations in consensus duration.

The relevant metrics are transferred into the Python 3.11.8 environment through PromQL queries and aligned at the time-series level using the pandas library. In this way, on-chain metrics and transaction-level measurements are synchronized on the same time axis. This synchronization enables the quantitative examination of the correlation between TPS and latency. In particular, the relationship between block-production intervals and transaction queuing can be directly observed through Prometheus data, and a time-based analysis of the behavior of the consensus protocols under load can be performed.

The experimental data are analyzed in a Jupyter Notebook 7.3.3 environment, and various statistical and visual outputs are produced. The analysis process enables a comparative evaluation of the behavior of different consensus algorithms under dynamic transaction loads.

The produced analytical outputs are summarized below:

- By examining the throughput variation during burst moments, the system's response behavior under load and its transaction-load handling characteristics under the evaluated topology are assessed. The temporal change in different percentile latency values such as P50 and P95 enables the assessment of the stability and consistency characteristics of the consensus algorithms.
- System resource analysis is performed. CPU and memory utilization, as well as peak disk write throughput, are compared. In addition, the experimental study is strengthened through network-traffic analysis.
- Queue-length analysis examines the relationship between the transaction pool occupancy rate and block-production latency. This approach quantitatively demonstrates the impact of transaction accumulation on blockchain latencies.
- Cumulative Distribution Function (CDF) and line graphs are used to illustrate the distribution of transaction inclusion latency and to quantitatively show the variation in transaction confirmation times across protocols.

Based on these analytical outputs, the responses of each consensus algorithm to increasing burst load are evaluated quantitatively and comparatively and analyzed using time-series methods. As a result, performance differences in transaction-load handling and latency behavior under the evaluated topology become statistically observable.

5. Experimental Study

The methodology followed in the study is systematically designed to maintain the integrity of the experimental process and the consistency of the measurement procedure across repeated runs. In the first stage, each consensus network is automatically launched via Docker Compose, ensuring network component isolation. Within this scope, Prometheus and Grafana containers are integrated into each private blockchain network, creating a centralized monitoring infrastructure. This structure enables the continuous collection of both system-level resource usage and on-chain metrics.

In the second stage, burst-type transaction loads are sent to the nodes in parallel through the developed notebook scenario. During this process, Prometheus periodically collects system-level metrics such as CPU utilization, memory usage, block time, and transaction-queue occupancy from all components of the network. The dynamic behavior of the consensus algorithm under load can be observed in real time.

At the final stage, the time-series data obtained from Prometheus are transferred into the Python environment and analyzed using the pandas library. The resulting measurements are compared using performance indicators such as throughput, transaction latency, and resource utilization. This integrated approach provides a systematically documented performance-measurement infrastructure for the controlled comparison of different consensus protocols.

5.1. Experiment Settings

Experimental studies are conducted on a hardware and software infrastructure optimized for consensus-network performance tests that require high transaction capacity and low latency. The experimental environment is configured on Ubuntu 22.04.5 LTS (x86_64), running on a custom-built workstation, and the components used are selected with system stability, scalability, and resource efficiency in mind. The test infrastructure employs an Intel Core i9-14900KS processor (Intel Corporation, Santa Clara, CA, USA; 32 threads, 5.9 GHz), 128 GB DDR5 RAM, and an NVIDIA RTX A6000 graphics unit (NVIDIA Corporation, Santa Clara, CA, USA). This configuration provides high parallel computing power and memory bandwidth, enabling consistent measurement of performance variables related to block production and the consensus process. The use of a single hardware platform was intended to maintain identical computational conditions across all evaluated consensus configurations and thereby support a controlled within-platform comparison. Nevertheless, the reported absolute performance values should not be interpreted as hardware-independent estimates. Differences in processor capability, memory availability, storage performance, virtualization overhead, and other system resources may affect observed throughput, latency, and resource utilization, particularly in resource-constrained deployments. Hardware sensitivity across heterogeneous platforms is therefore considered outside the scope of the present study and is identified as an important direction for future evaluation.

All experimental blockchain networks were deployed using Hyperledger Besu v25.7.0 on Linux x86_64 with OpenJDK 21 (besu/v25.7.0/linux-x86_64/openjdk-java-21). The same Besu release was consistently used for the Ethash, Clique, IBFT 2.0, and QBFT configurations to avoid software-version-dependent effects on the comparative results. Clique was supported by this Besu release at the time of the experiments; its support was removed in later Besu releases, specifically in v26.4.0. Therefore, the Clique results reported in this study should be interpreted within the software environment evaluated here.

In this paper, independent private blockchain networks based on Hyperledger Besu are created for each consensus algorithm. Each private network consists of four nodes, each running in its own isolated Docker-based virtual network. These networks are positioned completely isolated from one another, while the nodes are interconnected through fixed peer addresses defined in the `static-nodes.json` file. On each Besu node, a metrics endpoint (metrics-port 9545) is enabled to allow Prometheus to collect data. Since the number of validators is restricted to four, the present experiments cannot be considered as an empirical test for scalability either on a network level or on a validator level. This is particularly important for BFT algorithms because their communication cost increases as the number of validators increases. Thus, the results cannot be generalized to larger sets of validators. This controlled configuration was selected to isolate protocol-level performance differences; the reported results are therefore specific to the four-node, fixed-block-period environment and do not extend to dynamic or geographically distributed deployments.

To ensure the reliability and statistical significance of the experimental findings, each consensus protocol was evaluated using repeated trial runs under identical conditions. For each consensus algorithm, the 5000-transaction burst workload was executed independently 10 times under identical experimental conditions. Repeated-run summary statistics and inferential analyses are based on these ten runs, while Table 2 presents the results of a single run to illustrate the transaction-level performance profile under the benchmark configuration. The use of a fixed workload enables a controlled comparison of protocol behavior rather than workload-driven variance. All experiments were carried out using four nodes and identical hardware and network environments. For Clique, QBFT, and IBFT 2.0, the configured block-production period was 2 s, whereas Ethash was evaluated using the fixed difficulty value 0x1. No changes were made to node count, block interval,

validator set, or leader rotation behavior, as the goal of this study was to isolate and compare the intrinsic performance characteristics of IBFT2, QBFT, Clique, and Ethash under strictly equivalent operating conditions. This approach eliminates confounding variables and ensures that the observed performance differences arise solely from the consensus mechanisms themselves rather than topology or configuration inconsistency.

Table 2. Performance Metrics of Consensus Algorithms under Distributed Burst Benchmark.

Algorithm	Median Latency P50 (ms)	95. Percentile P95 (ms)	Max Latency (ms)	Approx _{tps}
IBFT 2.0	8303	29,677	39,546	110.047
QBFT	7145	31,352	39,514	109.970
Clique	1006	1859	2003	211.533
Ethash	861,527	1,402,733	1,432,008	3.449

5.2. Monitoring Infrastructure

A centralized monitoring infrastructure utilizing Prometheus and Grafana is established to improve the traceability and observability of the testing process. Prometheus and Grafana services are integrated into the experimental environment to enhance the system's measurement and monitoring capabilities. Through these services, key metrics such as CPU and memory usage, block-production time, network latency, and transaction throughput are collected in real time and incorporated into the evaluation process.

Prometheus is configured to gather metric data from all nodes every 2 s. The aggregated data are visualized on Grafana through dynamic dashboards. These dashboards display critical performance indicators, including block-production time, CPU and memory utilization, network synchronization status, and TPS, in real time.

Dedicated genesis files are produced for each consensus protocol. For Clique, QBFT, and IBFT 2.0, the relevant genesis parameters were configured with a 2 s block-production period, whereas the Ethash configuration used the fixed difficulty value 0x1, while the epoch and timeout parameters are adjusted based on the structural attributes of each consensus type. This configuration facilitates a comparative evaluation of deterministic (Clique) and Byzantine Fault Tolerant (QBFT, IBFT 2.0) algorithms under identical conditions.

Every private blockchain network established in the experimental environment consists of four nodes, all initialized with identical genesis configurations. This framework ensures that each network starts from a uniform initial state and that all nodes operate under identical chain parameters.

Each node is configured with the following startup parameters:

- `-rpc-http-enabled`
- `-rpc-http-api = ETH,NET,IBFT,WEB3`
- `-metrics-enabled`
- `-metrics-port = 9545`
- `-sync-mode = FULL`

The applied configuration enables the collection of both intra-network communication metrics and system-level performance indicators. In particular, the parameters `-metrics-enabled` and `-metrics-port` allow Prometheus to collect data in time-series form. Using the Prometheus metrics obtained, values such as network latency, block-production time, and finality (the time required for transactions to become final) are derived. These metrics are used for the quantitative comparison of performance differences among the various consensus algorithms under dynamic conditions.

5.3. Docker-Based Isolation and Topology

The experimental environment is configured in an isolated manner using Docker containers. Each test network is executed on independent container clusters. Network traffic, transaction load, and resource usage can be analyzed separately without interacting with one another. Also, we enable the observation of transaction-load handling behavior under the fixed network topology and allow reliable comparison of the performance differences exhibited by various consensus algorithms under load.

In addition, Prometheus and Grafana services are integrated into the experimental environment to enhance the system's measurement and monitoring capabilities. Through these services, key metrics such as CPU and memory usage, block-production time, network latency, and transaction throughput are collected in real time and incorporated into the evaluation process.

5.4. Throughput Performance Analysis

The experimental results show that the four different consensus protocols exhibit distinct performance characteristics under identical hardware and network-topology conditions. In each test, 5000 transactions are submitted as a burst workload, and no transaction failures are observed. The results demonstrate that all networks operate stably across their verification, nonce management, and block-creation processes. In addition, synchronization protocols ensure consistent transaction pool management across participating nodes.

Approximate TPS indicates the transaction-processing capacity of the system. Throughput represents the number of transactions a blockchain system can process per unit time [43]. In this study, TPS is evaluated using an end-to-end burst throughput definition rather than a block-level or sliding-window metric. The TPS calculation is defined as:

$$TPS_{\text{burst}} = \frac{N_{\text{success}}}{t_{\text{last_included}} - t_{\text{first_submitted}}} \quad (5)$$

In Equation (5), N_{success} denotes the total number of successfully processed transactions within a single burst workload. $t_{\text{first_submitted}}$ represents the timestamp at which the first transaction in the burst is broadcast to the network, while $t_{\text{last_included}}$ corresponds to the block timestamp of the last transaction successfully included in the blockchain. This definition captures the effective end-to-end transaction-processing capacity under high-load conditions by incorporating both transaction queuing latencies and block-production dynamics. Higher TPS values indicate faster completion of burst transaction workloads from the client's perspective.

Table 2 contains the performance values on 1 run obtained under identical hardware, network-topology, and test conditions for the selected consensus algorithms.

IBFT 2.0 provides a consistent latency profile. The measurements record a median (P50) transaction-inclusion latency of 8303 ms, a P95 latency of 29,677 ms, and a maximum latency of 39,546 ms. Short-term latencies are observed during time-synchronization or view-change phases. The short block interval (2 s) and the use of four validator nodes enable IBFT2 to generate a high transaction throughput.

QBFT, with a median (P50) transaction-inclusion latency of 7145 ms, exhibits a slightly lower median latency than IBFT 2.0, although the P95 latency increases slightly to 31,352 ms. The maximum latency is measured as 39,514 ms, which is almost identical to that of IBFT 2.0. In terms of TPS, the value of 109.970 TPS indicates that QBFT operates at efficiency levels comparable to those of IBFT 2.0.

Clique exhibits the characteristic properties of a PoA structure, and the results clearly show how this structure is reflected in performance. The median (P50) transaction-inclusion

latency of 1006 ms, the P95 latency of 1859 ms, and the maximum latency of 2003 ms demonstrate that Clique operates with minimal latency in the transaction-validation process.

As a consequence of this low-latency profile, Clique reaches approximately 211.533 TPS, achieving the highest transaction capacity among all algorithms. The experimental results are consistent with the 2 s block period defined in the genesis configuration, which explains the periodic throughput pattern observed in the time-series analysis.

The Ethash algorithm shows high latency metrics, with a median (P50) transaction-inclusion latency of 861,527 ms and an estimated capacity of 3.449 TPS, indicating substantially lower throughput than alternative algorithms. Figure 1 presents time-series graphs that compare the block-production attributes, stability metrics, and operational efficacy of each method, providing a direct visual reference for the subsequent analysis.

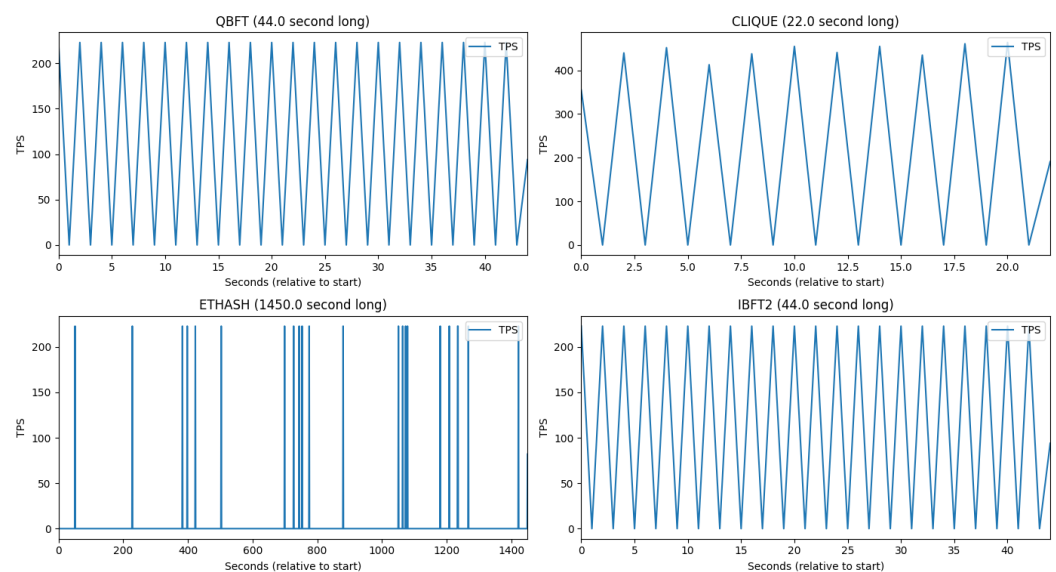


Figure 1. TPS performance.

The graph indicates that Clique achieves the highest throughput, averaging around 217 TPS during the experiment. Its peak value of 464 TPS results from the low latency and rapid block creation seen in the PoA process. By contrast, although QBFT and IBFT2 deliver lower throughput, both algorithms display highly consistent block-production patterns with minimal variation.

Ethash, as depicted in the graph, displays erratic behavior and significant volatility, with a consistently low average throughput of about 3.4 TPS. This performance arises from the energy-intensive, competitive nature of the PoW.

As shown in Figure 2, an analysis of the time-dependent TPS behavior from ten independent experimental runs conducted under uniform conditions for each consensus algorithm indicates that QBFT and IBFT2 yield highly consistent and periodic throughput patterns corresponding to the block-production interval. No significant variation in TPS behavior is noted across multiple trials, demonstrating a high level of temporal consistency. This stability underscores the efficacy of Byzantine Fault Tolerant consensus systems, whose deterministic scheduling and synchronization attributes provide reliable and resilient transaction processing, even under continuous strain. The results for Clique indicate significantly elevated TPS amplitudes compared with the other treatments. Nevertheless, these variations maintain structural consistency throughout all repeats, displaying similar waveform shapes and peak throughput values. Ethash outputs exhibit strong irregularity and volatility, characterized by TPS values that are sparse and irregularly distributed across

a rather extended time frame. This process engenders significant performance heterogeneity and constrains effective transaction throughput under similar experimental conditions.

In this study, a comprehensive analysis was conducted to evaluate the impact of different consensus algorithms on latency and throughput metrics not only through raw measurements but also in terms of statistical generalizability and significance. First, descriptive statistics for each metric, including the mean and standard deviation, were reported, together with the coefficient of variation (CV) as an indicator of stability and the 95% confidence interval (CI). Subsequently, the Kruskal–Wallis H test was employed to assess the statistical significance of differences among groups, Dunn’s post hoc test with Bonferroni correction was applied to determine which pairs differed, and Cliff’s Delta effect size analysis was used to quantify the practical magnitude of these differences. Since single-run results do not provide statistical generalizability, the generalization and significance interpretations in this study are based on outputs obtained from multiple experimental repetitions.

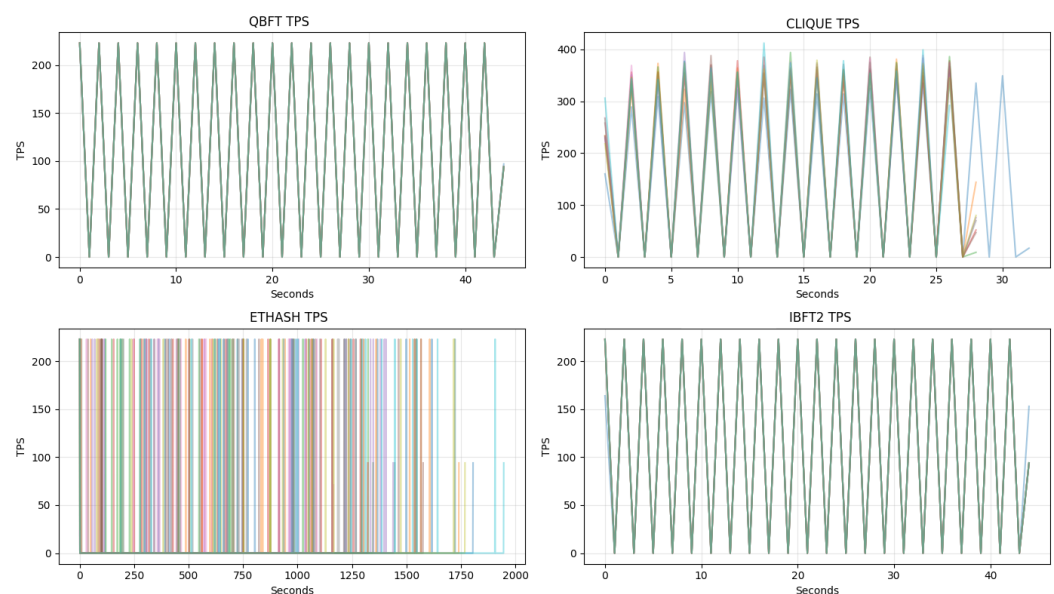


Figure 2. Time-dependent TPS behavior over 10 experimental runs. Different colored lines represent independent experimental runs for each consensus algorithm.

5.5. Descriptive Statistics: Mean, CV, and 95% CI

Descriptive statistics characterize both the central tendency and the dispersion of the measurements, thereby making the consistency across 10 runs explicitly observable. In this study, the following statistical measures are employed:

- **Mean (+/− deviation)** expresses the average performance across repeated runs and its variability.
- **Coefficient of variation (CV = Std/Mean)** enables the comparison of stability across metrics with different scales.
- **A 95% confidence interval (CI)** represents the uncertainty interval of the true mean.

Table 3 reports the descriptive statistics for P50 latency across the evaluated consensus algorithms, including base values, deviation ranges, coefficients of variation, and 95% confidence intervals. IBFT 2.0 has a P50 latency of 7591.18 ms, with deviations of +1173.82 ms/−2677.18 ms. It shows a coefficient of variation of 25.4%, indicating moderate dispersion. The 95% confidence interval is [4914, 8765] ms, reflecting a relatively wide spread around the mean. QBFT records a P50 latency of 6741.64 ms, with deviations of +1917.36 ms/−2180.64 ms. Its variability is slightly higher than IBFT 2.0, with

CV = 30.4%. The 95% confidence interval [4561, 8659] ms confirms substantial dispersion in latency values. Clique demonstrates the lowest P50 base latency of 1005.36 ms, with deviations of +97.64 ms/−36.36 ms. It also exhibits the lowest variability, with CV = 6.7%, indicating highly stable performance. The narrow 95% confidence interval [969, 1103] ms further supports the precision and consistency of its latency measurements. Ethash shows a markedly higher P50 latency of 1,084,964.36 ms, with large deviations of +342,747.64 ms/−473,010.36 ms. It has the highest variability among the evaluated algorithms, with CV = 37.6%. The very wide 95% confidence interval [611,954, 1,427,712] ms highlights the significant uncertainty and fluctuation in its latency performance.

Table 3. Descriptive Statistics (P50 Latency).

Algorithm	Mean (+/−) (ms)	CV (%)	95% CI (ms)
IBFT 2.0	+1173.82	25.4	[4914, 8765]
	7591.18		
	−2677.18		
QBFT	+1917.36	30.4	[4561, 8659]
	6741.64		
	−2180.64		
Clique	+97.64	6.7	[969, 1103]
	1005.36		
	−36.36		
Ethash	+342,747.64	37.6	[611,954, 1,427,712]
	1,084,964.36		
	−473,010.36		

Note: Bold values indicate the mean. The values shown above and below the mean indicate the positive (+) and negative (−) deviations from the mean, respectively.

As shown in Table 4, IBFT 2.0 has a P95 base latency value of 28,977.45 ms, and the deviations are +2321.55 ms/−3785.45 ms. The coefficient of variation of 10.6% indicates a relatively low level of dispersion. The [25,192, 31,299] ms 95% confidence interval reveals that the distribution around the mean is limited. QBFT recorded a P95 base latency value of 28,443.91 ms, and its deviations are in the range of +2919.09 ms/−5489.91 ms. Its variability is higher than that of IBFT 2.0 (CV = 14.8%). The [22,954, 31,363] ms 95% confidence interval indicates a noticeable but manageable dispersion in latency values. Clique shows the lowest p95 base latency value at 1885.36 ms, with deviations of +111.64 ms/−14.36 ms. It also has the lowest variability, with a CV value of 3.3%. This indicates a highly stable performance. The narrow [1871, 1997] ms 95% confidence interval supports that the latency measurements are estimated with high precision and consistency. Ethash has by far the highest P95 base latency value at 2,353,827.45 ms, with deviations of +837,259.55 ms/−934,326.45 ms. It has the highest variability among the evaluated algorithms (CV = 37.7%). The very wide [1,419,501, 3,191,087] ms 95% confidence interval reveals significant uncertainty and fluctuations in latency performance.

As shown in Table 5, Ethash has an approximate TPS mean of 2.13, which is substantially lower than that of all other algorithms. Its deviations are +1.31/−0.68, and the high CV value of 46.7% indicates significant variability in throughput. The wide [1.45, 3.44] 95% confidence interval further confirms the instability of its performance. IBFT 2.0 achieves an approximate TPS mean of 109.68, with deviations of +1.44/−0.49. Its CV value of 0.9% indicates a highly stable throughput profile. The narrow [109.19, 111.12] 95% confidence

interval supports the consistency and reliability of its performance. QBFT records a similar approximate TPS mean of 110.12, with deviations of +2.15/−0.84. It maintains low variability with a CV value of 1.4%, indicating stable throughput. The [109.28, 112.27] 95% confidence interval shows a tight distribution around the mean. Clique demonstrates the highest approximate TPS mean at 192.30, yielding greater average throughput than IBFT 2.0 and QBFT. Its deviations are +20.70/−32.14, and the CV value of 13.8% reflects a higher level of variability. The wider [160.16, 213.00] 95% confidence interval indicates more pronounced fluctuations across runs compared with IBFT 2.0 and QBFT.

Table 4. Descriptive Statistics (P95 Latency).

Algorithm	Mean (+/−) (ms)	CV (%)	95% CI (ms)
IBFT 2.0	+2321.55	10.6	[25,192, 31,299]
	28,977.45		
	−3785.45		
QBFT	+2919.09	14.8	[22,954, 31,363]
	28,443.91		
	−5489.91		
Clique	+111.64	3.3	[1871, 1997]
	1885.36		
	−14.36		
Ethash	+837,259.55	37.7	[1,419,501, 3,191,087]
	2,353,827.45		
	−934,326.45		

Note: Bold values indicate the mean. The values shown above and below the mean indicate the positive (+) and negative (−) deviations from the mean, respectively.

Table 5. Descriptive Statistics (Approximate TPS).

Algorithm	Mean (+/−) (TPS)	CV (%)	95% CI
IBFT 2.0	+1.44	0.9	[109.19, 111.12]
	109.68		
	−0.49		
QBFT	+2.15	1.4	[109.28, 112.27]
	110.12		
	−0.84		
Clique	+20.70	13.8	[160.16, 213.00]
	192.30		
	−32.14		
Ethash	+1.31	46.7	[1.45, 3.44]
	2.13		
	−0.68		

Note: Bold values indicate the mean. The values shown above and below the mean indicate the positive (+) and negative (−) deviations from the mean, respectively.

5.6. Non-Parametric Global Significance Test

Classical parametric methods (e.g., ANOVA) rely on assumptions of normality and homogeneity of variances. In this study, to avoid imposing these assumptions given the distributional characteristics of the measurements and/or the sample structure, the rank-

based non-parametric Kruskal–Wallis H test was employed. This test evaluates inter-group differences based on the ranks of the observations rather than the raw values, thereby providing a more robust comparison.

Hypotheses

H_0 : The distributions of the relevant metrics are identical across all consensus algorithms.

H_1 : At least one consensus algorithm exhibits a different distribution.

The Kruskal–Wallis results presented in Table 6 indicate statistically significant differences among the consensus algorithms across all evaluated metrics. Specifically, for the P50 metric, $H = 32.94$ with $p = 3.32 \times 10^{-7}$ was obtained; for the P95 metric, $H = 32.949$ with $p = 3.30 \times 10^{-7}$; for approximate TPS, $H = 32.933$ with $p = 3.33 \times 10^{-7}$; and for peak TPS, $H = 37.969$ with $p = 2.87 \times 10^{-8}$. Since all associated p -values are well below the 0.001 significance level, the null hypothesis (H_0 : the group distributions are identical) is rejected for each metric. This result confirms that the consensus algorithms are not only observably but also statistically strongly differentiated in terms of latency and throughput metrics.

Table 6. Kruskal–Wallis Global Significance.

Metric	H	p -Value
P50 (ms)	32.94	3.32×10^{-7}
P95 (ms)	32.949	3.30×10^{-7}
Approx _{tps}	32.933	3.33×10^{-7}
Peak _{tps}	37.969	2.87×10^{-8}

5.7. Dunn’s Post Hoc and Bonferroni Correction

The Kruskal–Wallis test only indicates that at least one group differs and does not identify which specific group pairs exhibit significant differences. Therefore, Dunn’s test was applied as a post hoc analysis in addition to the Kruskal–Wallis test. To control the risk of spurious significance arising from multiple pairwise comparisons, the Bonferroni correction was employed, and the adjusted p -values were reported as p_{adj} .

The results in Table 7 indicate that several pairs exhibit statistically significant differences at the level of $p_{adj} < 0.05$ for the P50, P95, and approximate TPS metrics. In particular:

- For P50 and P95 latency, the Clique–Ethereum comparison is extremely strong ($p_{adj} \approx 5.7 \times 10^{-8}$).
- For approximate TPS, the Clique–Ethereum comparison is again very strong ($p_{adj} = 5.74 \times 10^{-8}$).
- For peak TPS, it is observed that the significant differences are concentrated especially between Clique and the other algorithms.

The post hoc analysis conducted in this study provides a statistical foundation for the arguments in the manuscript regarding which algorithm differs from which. Through the use of multiple-comparison correction, the reliability of the results is increased.

Table 7. Dunn’s Post Hoc Significant Pairs.

Metric	Group 1	Group 2	z	p_{adj} (Bonf.)
P50 (ms)	QBFT	CLIQUE	2.812	0.0296
P50 (ms)	QBFT	ETHASH	−2.926	0.0206
P50 (ms)	CLIQUE	ETHASH	−5.738	5.74×10^{-8}
P50 (ms)	CLIQUE	IBFT2	−2.926	0.0206
P50 (ms)	ETHASH	IBFT2	2.812	0.0296
P95 (ms)	QBFT	CLIQUE	2.927	0.0205
P95 (ms)	QBFT	ETHASH	−2.812	0.0295
P95 (ms)	CLIQUE	ETHASH	−5.739	5.71×10^{-8}
P95 (ms)	CLIQUE	IBFT2	−2.812	0.0295
P95 (ms)	ETHASH	IBFT2	2.927	0.0205
approx _{tps}	QBFT	CLIQUE	−2.831	0.0279
approx _{tps}	QBFT	ETHASH	2.907	0.0219
approx _{tps}	CLIQUE	ETHASH	5.738	5.74×10^{-8}
approx _{tps}	CLIQUE	IBFT2	2.907	0.0219
approx _{tps}	ETHASH	IBFT2	−2.831	0.0279
peak _{tps}	QBFT	CLIQUE	−5.031	2.93×10^{-6}
peak _{tps}	CLIQUE	ETHASH	5.031	2.93×10^{-6}

5.8. Cliff’s Delta Test for Effect Size

p -values alone indicate the existence of a difference but do not convey how large or practically meaningful that difference is. To assess the magnitude and practical significance of the observed differences, Cliff’s Delta (δ), an effect size measure suitable for non-parametric pairwise comparisons, was computed in this study. Cliff’s Delta expresses the degree of separation between two group distributions in probabilistic terms. Values of $\delta = 1$ or $\delta = -1$ indicate that the observations of one group are completely dominant over those of the other, corresponding to maximal separation. For all comparisons reported in Table 8, $\delta = \pm 1.0$ was obtained and the magnitude was classified as large. This finding demonstrates that the choice of consensus algorithm plays a practically decisive role in terms of latency and throughput.

Table 8. Cliff’s Delta Effect Sizes.

Metric	Group 1	Group 2	δ (Cliff)	Magnitude
P50 (ms)	QBFT	CLIQUE	1.0	large
P50 (ms)	QBFT	ETHASH	−1.0	large
P50 (ms)	CLIQUE	ETHASH	−1.0	large
P50 (ms)	CLIQUE	IBFT2	−1.0	large
P50 (ms)	ETHASH	IBFT2	1.0	large
P95 (ms)	QBFT	CLIQUE	1.0	large
P95 (ms)	QBFT	ETHASH	−1.0	large
P95 (ms)	CLIQUE	ETHASH	−1.0	large
P95 (ms)	CLIQUE	IBFT2	−1.0	large
P95 (ms)	ETHASH	IBFT2	1.0	large
approx _{tps}	QBFT	CLIQUE	−1.0	large
approx _{tps}	QBFT	ETHASH	1.0	large
approx _{tps}	CLIQUE	ETHASH	1.0	large
approx _{tps}	CLIQUE	IBFT2	1.0	large
approx _{tps}	ETHASH	IBFT2	−1.0	large
peak _{tps}	QBFT	CLIQUE	−1.0	large
peak _{tps}	CLIQUE	ETHASH	1.0	large

5.9. System Resource Utilization Analysis

Figure 3 illustrates the time-series characteristics of CPU and memory utilization for the evaluated consensus algorithms. The graphs are essential for illustrating how the nodes use system resources during transaction validation, block production, and synchronization procedures throughout the experiment.

Memory usage across all protocols consistently falls within the 42.3–43.3% range. The proximity of these values suggests that the memory demands of the consensus methods are relatively consistent and largely unaffected by the underlying architecture. The server in the test environment, equipped with 128 GB of physical memory, exhibits a consumption ratio that also accounts for background activities active under idle conditions. The minimal fluctuation in memory utilization indicates that the techniques do not impose additional memory overhead and do not pose an operational limitation on scalability. Average CPU utilization remains between 20.5% and 24.2%. However, the CPU time-series data display notable differences among the protocols.

QBFT demonstrates the most consistent CPU utilization pattern. The graph shows a low-variance CPU curve over time, indicating that QBFT maintains an equitable distribution of computational load throughout the validation and block-production stages. The BFT-based architecture ensures frequent message rounds, thereby reducing abrupt CPU spikes. IBFT 2.0, although similar to QBFT in average usage, exhibits significant peaks of 40–50% during specific time intervals. These oscillations may be linked to brief intervals during which view alterations or rigorous validation processes are initiated. This suggests that the high-throughput benefit of the IBFT 2.0 algorithm may lead to immediate surges in CPU load.

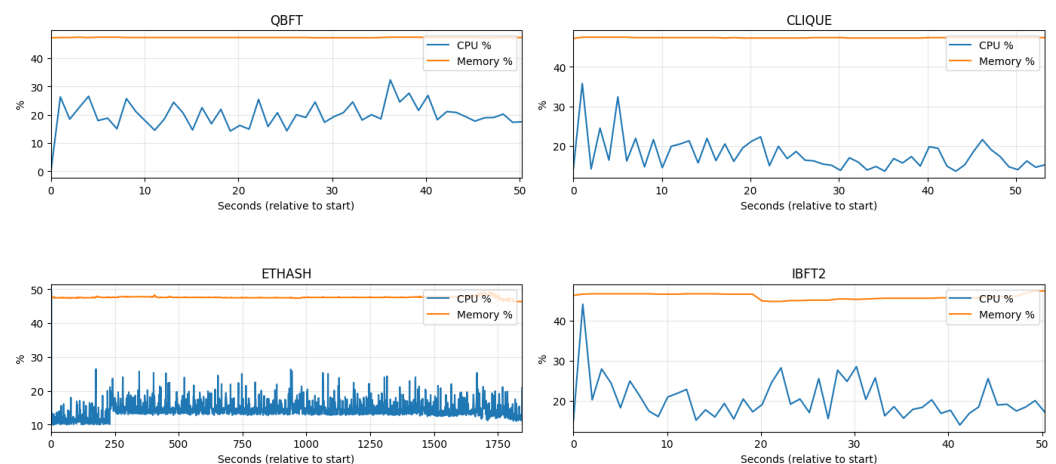


Figure 3. CPU–memory performance.

Ethash, owing to its Proof-of-Work architecture, demonstrates the most variability in CPU utilization. The CPU curve fluctuates significantly, indicating that hash-calculation activities consistently and erratically tax the machine. The significant variance aligns with the energy-intensive and computationally demanding nature of PoW techniques. Clique exhibits low CPU utilization; however, it shows greater short-term variability than BFT-based protocols. The rotation of leaders and the block-production cycle induce intermittent minor surges in CPU usage, in association with leader rotation and the configured block-production cycle.

Figure 4 illustrates how memory usage changes over time for four different workloads across ten independent experimental runs. According to the results, memory usage initially goes through a short adaptation phase and then stabilizes within a workload-specific band. In the QBFT process, memory usage stabilizes early and shows no clear directional change

over time. Differences between experiments remain limited, with each run maintaining its own level. This indicates that QBFT does not generate additional memory demand as execution time increases and that its memory footprint is predictable. Although one curve at the lower level deviates from the others, this difference does not grow over time and remains stable. Clique memory usage reaches equilibrium rapidly at the beginning of the experiments and then remains flat within a narrow range. There is no clear convergence between runs; each condition preserves its own memory level. The Ethash panel clearly differs from the other three workloads. Memory usage increases gradually over time, and especially in long runs, the lower curve approaches the upper bound. The increase is not abrupt but stepwise and time-dependent. IBFT2 exhibits memory usage within a narrow and regular band. No consistent increasing or decreasing trend is observed over time. The ordering between experiments is largely preserved, with curves progressing in parallel. This behavior shows that IBFT2 provides a stable memory profile in both short- and long-term executions. For QBFT, Clique, and IBFT2, memory usage is largely independent of time, whereas Ethash stands out with increasing memory demand as runtime grows. This difference indicates that evaluations based on a single average memory value can be misleading, particularly for Ethash-like workloads.

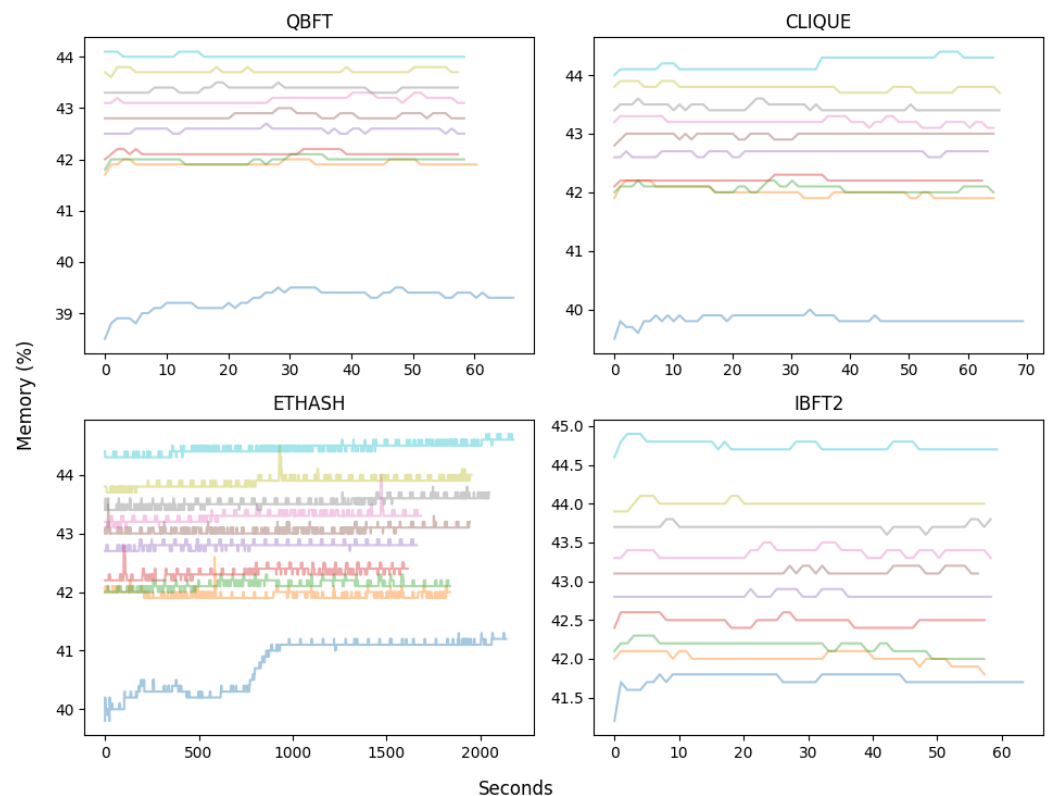


Figure 4. Memory utilization over time across multiple independent runs. Different colored lines represent the individual experimental runs for each consensus algorithm.

Figure 5 shows how CPU usage changes over time under four different workloads across ten independent experimental runs. A common pattern observed in all panels is a brief period of high CPU usage and fluctuation at the beginning of execution, followed by a transition to a lower and more stable level. This indicates that initial setup and synchronization costs diminish over time. In the QBFT panel, CPU usage reaches a pronounced peak at the start, but this increase quickly subsides. Although high-amplitude fluctuations are observed in the first few seconds, the system rapidly settles at an approximately constant level. As time progresses, differences between runs narrow and the curves converge. The results

indicate that QBFT imposes a limited and predictable CPU load during long-term operation. In the Clique experiments, initial CPU usage is both more variable and spread over a wider range. This variability decreases over time, and usage converges into a narrower band. This suggests that Clique generates a higher computational load during the initial execution phase, but CPU demand stabilizes once steady state is reached. Despite long-term execution, Ethash CPU usage generally remains within a stable band. Although short-lived peaks are observed at the beginning, these effects are quickly dampened and usage remains largely flat. No significant increasing or decreasing trend in CPU consumption is observed over time. IBFT2 exhibits sharp but short-lived CPU spikes during the initial phase. These spikes quickly subside, and the system transitions to a more stable operating regime. In the steady state, CPU usage stays within a narrow range, with curves from different runs progressing largely in parallel. This indicates that IBFT2 produces a consistent processing load after the initial overhead. The main differences among workloads are concentrated in the duration of the transition period and the magnitude of the initial peaks. Experimental results show that CPU provisioning should account for the startup phase in particular, and that average CPU usage alone is not a sufficient indicator for long-running workloads.

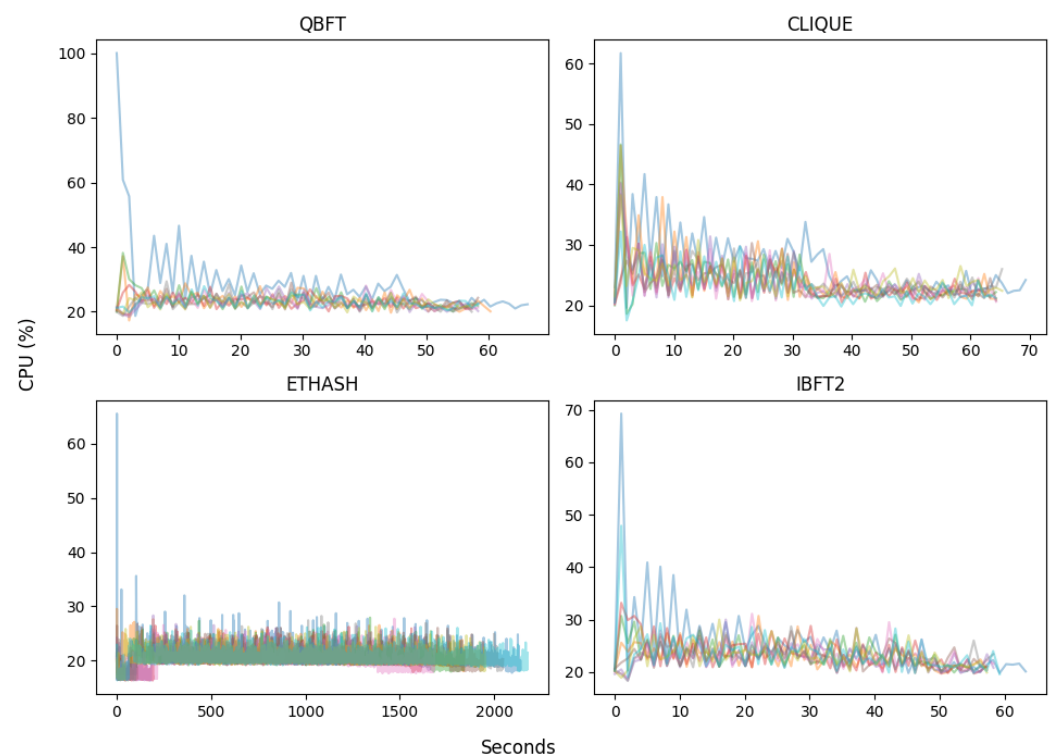


Figure 5. CPU utilization over time across multiple independent runs. Different colored lines represent the individual experimental runs for each consensus algorithm.

Table 9 presents CPU and memory (Mem) usage data for the consensus algorithms (C.A.) throughout the experiment, including minimum, average, and maximum values. The experimental results indicate that all algorithms exhibit comparable memory consumption. However, CPU utilization peaks higher in IBFT 2.0 and Ethash, resulting in a more pronounced periodic stress on system resources.

Figure 6 illustrates a comparison of consensus methodologies for peak disk write throughput. This measure is a crucial performance indicator, representing the maximum data volume that nodes can write to disk during intensive transaction processing. The rate of block formation, transaction volume, and computing characteristics of the consensus mechanism directly influence disk I/O activity. This analysis is essential for under-

standing the resource use profiles of the evaluated designs. The vertical axis illustrates peak disk write throughput in MB/s, enabling a direct comparison of approaches on a consistent scale.

Table 9. CPU and Memory Performance of Consensus Algorithms.

C.A.	CPU Min	CPU Mean	CPU Max	Mem Min	Mem Mean	Mem Max
QBFT	19.05%	23.623%	37.21%	42.33%	42.498%	42.62%
Clique	19.93%	24.170%	40.55%	42.51%	42.641%	42.76%
Ethash	16.43%	20.516%	31.29%	42.61%	42.820%	43.25%
IBFT 2.0	19.23%	23.350%	36.00%	42.87%	43.016%	43.12%

Ethash exhibits the highest peak disk write performance, reaching approximately 25 MB/s, indicating a significantly increased disk I/O requirement during peak activity. Clique, however, attains a significantly lower peak value of approximately 14 MB/s, signifying reduced disk write pressure and enhanced disk utilization during block formation. IBFT2 demonstrates the lowest peak disk write throughput at around 12 MB/s, indicating minimal disk I/O demands and a more storage-efficient execution profile. QBFT maintains a median state, attaining a peak disk write throughput of roughly 15 MB/s, signifying moderate disk consumption.

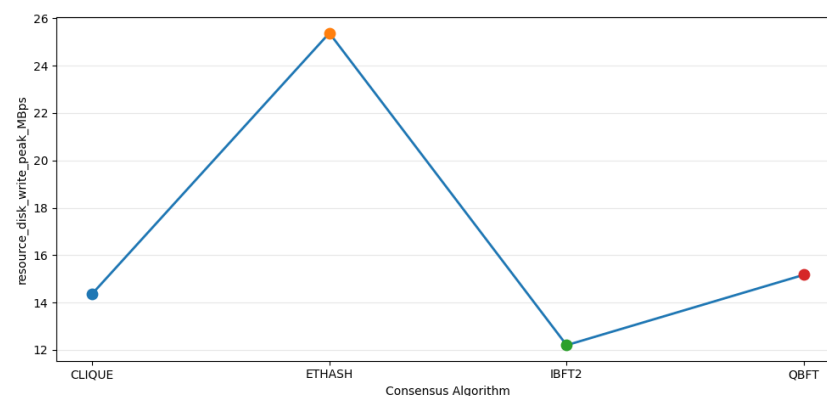


Figure 6. Consensus algorithm peak disk I/O performance. Marker colors are used for better reader experience.

Energy usage was measured using pyRAPL (v0.2.3.1), which accesses the Intel Running Average Power Limit (RAPL) counters through the powercap subsystem of Linux (/sys/class/powercap/intel-rapl). The measurements were taken for the PKG (CPU Package) domain of a single Intel Core i9-14900KS socket (Intel Corporation, Santa Clara, CA, USA), whereby the counter reading is performed before and after each benchmark run, meaning that one energy measurement corresponds to the whole run. RAPL gives energy usage in microjoules, while these numbers have been converted to kilojoules. Because RAPL aggregates power on a socket basis, the values given account for all processes running on the system during the run—the four Besu containers, the Prometheus and Grafana containers, and the benchmark client—and thus correspond to CPU package energy on the host being experimented with, rather than per node or overall system energy. Contributions to energy usage by hardware not contained in the CPU package, such as the GPU, storage, networking, and losses in the power supply, have not been captured by RAPL. Because RAPL reports cumulative CPU-package energy over the measurement interval, the total energy reported for each run depends on the duration of that run and should not be interpreted as energy attributable to individual transactions.

Figure 7 shows the distribution of package energy consumption across consensus algorithms over ten runs. When the energy values are converted to kilojoules, IBFT2 exhibits the lowest energy cost and the highest stability, with a mean consumption of 9.40794 ± 0.14340 kJ and a coefficient of variation of 1.5%. The measurements cluster within a narrow band of 9.20329–9.71754 kJ. QBFT has a similar mean level; however, with 9.67814 ± 0.66979 kJ and a coefficient of variation of 6.9%, it shows a wider dispersion compared with IBFT2. The values are spread over a range of 9.27266–11.51678 kJ. Clique shows a more limited distribution than QBFT, with a mean consumption of 10.76022 ± 0.48792 kJ and a coefficient of variation of 4.5%. The energy band is observed in the range of 10.32315–12.05113 kJ. Ethash, on the other hand, diverges markedly, with a mean consumption of 34.36700 ± 20.95630 kJ and a coefficient of variation of 61.0%. It is distributed over a wide range, such as 9.84874–61.55814 kJ. The fact that the Ethash algorithm produces observations extending to high values and has a pronounced spread indicates that its energy consumption is more sensitive to experimental conditions and less predictable across runs. In contrast, the concentration of IBFT2 measurements within a narrow range demonstrates high consistency in energy consumption and reveals a more stable profile in terms of energy efficiency.

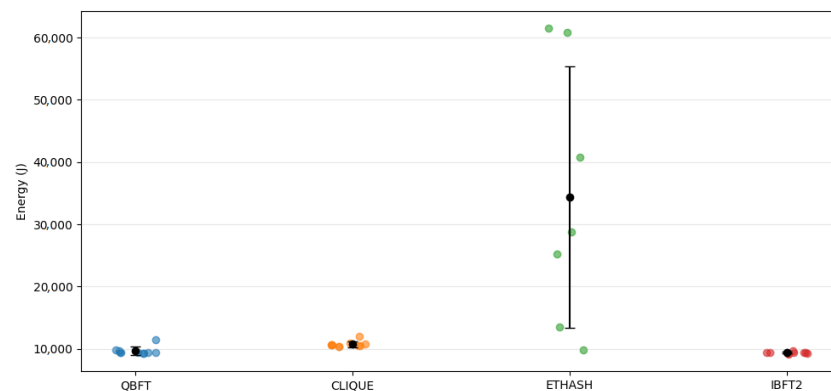


Figure 7. Energy Consumption distribution by consensus algorithm. Marker colors are used for better reader experience.

5.10. Network Traffic Analysis

In Figure 8, time-series outbound network traffic (Net Sent, MB/s) for the compared consensus algorithms is presented. In the graphs, along with the raw transmission rate, five-second moving averages (5 s MA) are used to display smoothed trends of short-term fluctuations. All reported traffic patterns and averages are derived from the results of 10 independent experimental runs. This visualization provides a comparative representation of the network load generated by the protocols based on block propagation, signature-verification traffic, and messaging intensity.

The Clique protocol produces the highest network traffic among all algorithms, with approximately 2.6 MB/s outbound and 5.5 MB/s inbound on average. This situation results from the frequent transmission of large data packets due to the short block period and the leader-rotation mechanism in the deterministic Proof-of-Authority structure of Clique. This high traffic level is an important characteristic that can increase communication costs, especially in multi-node networks.

The Ethash configuration generally exhibits relatively low average network throughput but pronounced short-duration traffic peaks. The mean outbound traffic rate is 2.02 MB/s, whereas the maximum reaches 34.52 MB/s, corresponding to approximately 17.1 times the mean value. Similarly, the inbound traffic rate increases from a mean of 1.68 MB/s to a maximum of 34.14 MB/s, approximately 20.3 times its mean value. These

measurements demonstrate substantial temporal variability in aggregate network traffic. However, the monitoring infrastructure records aggregate transmitted and received throughput and does not separately classify block propagation, transaction broadcast, peer synchronization, or other protocol-level messages. Consequently, the observed peaks cannot be attributed reliably to a specific Ethash protocol event from the available measurements. A detailed decomposition would require packet- or message-level tracing and is left for future work.

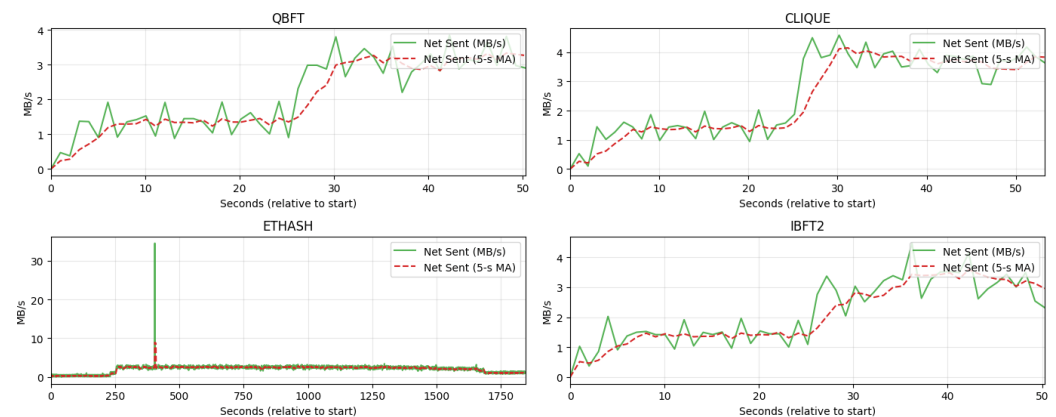


Figure 8. Network sent throughput across consensus algorithms. The red dashed line indicates the 5-s moving average.

QBFT and IBFT 2.0 protocols exhibit a similar traffic structure and operate in the range of approximately 2–3 MB/s outbound and 3–4 MB/s inbound on average. The experimental results show that the regular messaging phases (pre-prepare, prepare, and commit) of BFT-based algorithms create a predictable and balanced load on the network. In addition, the trend followed by the moving averages indicates that the traffic variance of these two algorithms is low and that their communication efficiency demonstrates high stability.

Table 10 summarizes the average and maximum values of the data-sending (Net Sent) and data-receiving (Net Recv) rates of consensus algorithms (C.A.) on the network. Clique exhibits the highest average bandwidth usage on both sending and receiving sides, while Ethash produces irregular but very high peak values (burst traffic). QBFT and IBFT2 have a medium-level communication load with a more balanced and predictable network-traffic profile.

Table 10. Data Transmission and Reception Rate Measurements.

C.A.	Net Sent Mean (MB/s)	Net Sent Max (MB/s)	Net Recv. Mean (MB/s)	Net Recv. Max (MB/s)
QBFT	2.14	3.85	3.79	7.96
Clique	2.57	4.58	5.48	11.03
IBFT 2.0	2.18	4.47	3.84	8.67
Ethash	2.02	34.52	1.68	34.14

5.11. Inclusion Latency Performance Analysis

Inclusion latency represents the time elapsed from the moment a transaction is submitted to the network until it is included in the selected block, and it is a critical performance indicator. This metric directly affects the following aspects:

- Transaction-handling capacity of the network;
- User experience;

- Finality speed;
- Suitability for real-time applications;
- Security–performance balance.

Figure 9 shows the inclusion latency distributions of the compared consensus algorithms. The shape of the latency histograms provides important indications not only about system performance but also about the architectural properties of the consensus protocol, the behavior of its timers, and the leader-selection mechanism.

The QBFT algorithm shows a distribution concentrated in the 0–5 s latency range, forming a noticeable clustering. Although most samples accumulate at low latency values, a long tail extending up to 40 s is observed. This indicates that QBFT provides low latency under normal conditions but can occasionally experience latency spikes.

The IBFT 2.0 histogram presents a distribution concentrated in the 1–10 s range and gradually decreasing as the latency increases. The maximum latency reaches 35 s. However, it exhibits lower variance compared with QBFT. It is understood that the deterministic finality mechanism of IBFT 2.0 is largely stable, although latency extensions can occur at certain times due to view-change events or late validator responses.

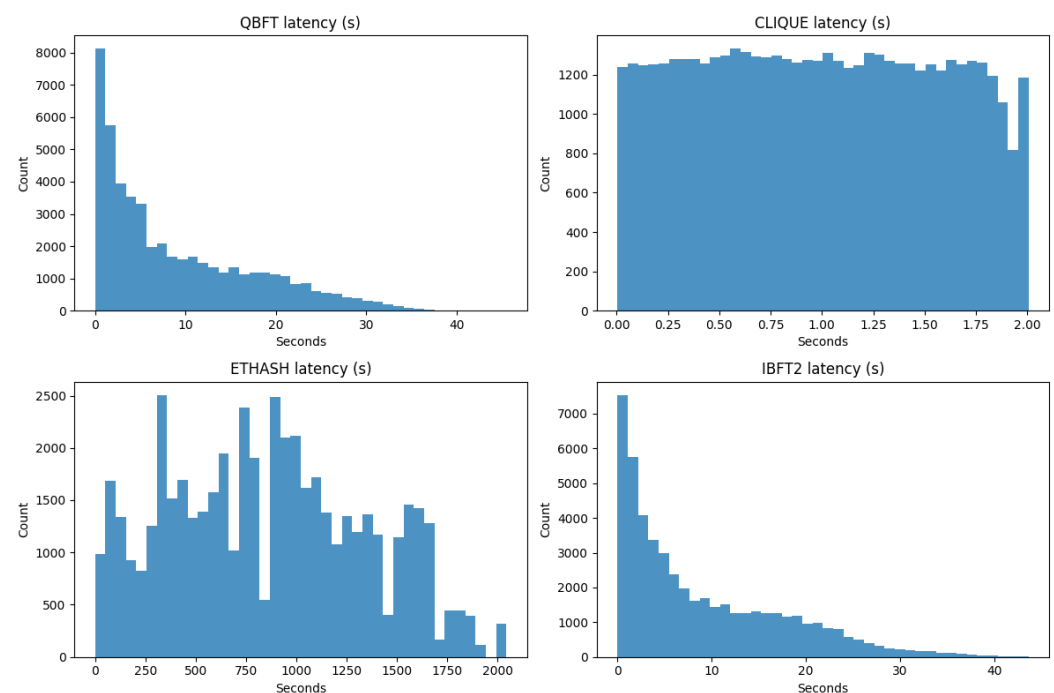


Figure 9. Inclusion latency distribution comparison.

In the Clique algorithm, the latency distribution forms an almost uniform line between 0 and 2 s. This flat distribution shows that transaction latency is tightly bound to the deterministic block-production cycle determined by the `blockperiodseconds=2` parameter rather than to network dynamics. Therefore, the variance of the latency is extremely low, and transaction confirmation times are shaped mostly according to the block scheduler.

In Ethash, latency values lie in a wide spectrum between 100 and 1400 s, and the distribution exhibits irregular jumps in clustered forms. This appearance is a direct reflection of block-production times that depend on random hash solving. The high variance and strongly right-skewed structure in the distribution stem from the non-deterministic nature of block production.

6. Discussion

However, the aforementioned discrepancies can be considered from the perspective of protocol characteristics that differ from the measurements. Indeed, the reported median latency of 1005.36 ms and 95th percentile (P95) of 1885.36 ms for Clique are not related to low message complexity: Under the fixed four-validator topology, differences in messaging complexity are not expected to dominate the multi-second inclusion latency differences observed here; however, message-level consensus latency was not measured separately. In Clique, block production does not require the multi-round quorum-certificate assembly used by QBFT and IBFT 2.0. Therefore, under the evaluated four-validator configuration, the configured block-production schedule is expected to be a major contributor to the observed inclusion latency profile. Although a uniformly distributed transaction-arrival process would theoretically yield a median waiting time of approximately $B/2$ for a block period B the present experiment uses a burst workload rather than uniform arrivals. Therefore, the observed median and P95 values should not be interpreted as a direct validation of this analytical model. Nevertheless, their magnitude is consistent with the configured 2 s block-production schedule being an important contributor to Clique's observed inclusion latency profile. The higher median inclusion latencies observed for QBFT and IBFT 2.0 are consistent with additional transaction-pool waiting and quorum-based block-production dynamics under the evaluated workload. However, because message-level consensus latency was not independently instrumented, the relative contribution of voting overhead and transaction-queue draining cannot be quantified separately from the available measurements.

Ethash differs from all three scheduled protocols by lacking the block scheduler. Block production happens as the memoryless search of the nonce space, so the distribution of the inter-block intervals is exponential with unity coefficient of variation and unbounded support, whereas all deterministic schedulers have near-zero coefficient of variation. This characteristic is likely to contribute substantially to the broader and more variable latency distribution observed for Ethash under the evaluated configuration. Residual waiting time for the next block does not depend on the elapsed waiting time, so no bounded worst case exists, contrary to Clique where it is constrained by the block period by construction. Blocks admit transactions en masse and have a single block timestamp, so the latency histogram in Figure 9 shows superposition of discrete mass points separated by the exponentially distributed gaps, explaining the irregular clusters. With saturation, the size of the block increases proportionally to the previous inter-block interval; thus, the Ethash configuration also exhibits pronounced peak-to-average variation in aggregate network traffic. However, because the monitoring infrastructure records aggregate transmitted and received throughput, the observed traffic peaks cannot be decomposed reliably into block propagation, transaction broadcast, peer synchronization, or other protocol-level message classes. The network-traffic and latency observations should therefore be interpreted as concurrent aggregate characteristics of the evaluated Ethash configuration rather than as measurements of the same specific protocol event. The high run-to-run variability observed in Ethash energy consumption should similarly be interpreted as a run-level measurement and not as energy attributable to a single protocol event.

According to the experimental results, the selection of a consensus mechanism cannot be made based on a single metric and requires a multidimensional evaluation that includes performance, security, resource efficiency, inclusion latency, and intended use. In this context, the obtained data clearly show that protocol selection in blockchain-based system design is a strategic and contextual decision.

Throughput analysis enables a scientific assessment of which structure is more suitable for specific use cases. In addition, since not only high average TPS values but also the level of fluctuation over time are critically important for system stability and sustainability,

throughput analysis provides a significant contribution to stability and reliability evaluations. The results also serve as a design guide for real-world applications. Furthermore, the analysis makes visible the effects of system-design parameters such as block-production times, messaging costs, and the number of validators on throughput, thereby clearly demonstrating the architectural implications of consensus-mechanism selection. The ability of throughput graphs to concretize performance trends, sudden drops, peak points, and systematic patterns provides strong visual evidence beyond verbal or textual explanations and increases the reliability and analytical depth of scientific reporting.

Experimental results obtained in the Throughput Performance Analysis show that BFT-based consensus protocols provide balanced performance in private blockchain applications in terms of high transaction volume, low latency, and fault tolerance. In this context, IBFT 2.0 and QBFT algorithms achieve rapid agreement among network nodes and exhibit stable performance under the evaluated transaction load and fixed four-node topology. The deterministic finality property of the IBFT 2.0 algorithm guarantees that blocks are confirmed irreversibly and, together with its low latency, makes this protocol a highly suitable structure for private (permissioned) blockchain networks.

The results further confirm that QBFT is a stable alternative for applications requiring low latency. They also show that QBFT exhibits smoother CPU and block-production behavior than IBFT 2.0, which produces short validation-driven peaks (Figure 5). Under the BFT architecture, QBFT is able to maintain system stability by limiting latency variance even when the transaction load increases. The fact that TPS levels are not as high as those of Clique indicates the latency cost of BFT protocols, which require multi-step messaging among validators.

The performance differences observed among the evaluated protocols are closely associated with the architectural characteristics of their underlying consensus mechanisms, particularly validator participation, message-exchange requirements, finality properties, and trust assumptions. Clique achieves the highest throughput and lowest inclusion latency in the evaluated environment because block production is coordinated by a predefined set of authorized signers without requiring the multi-round quorum communication employed by BFT-based protocols. The resulting reduction in coordination overhead improves transaction-processing efficiency while concentrating validator admission and block-production authority within the authorized signer set, thereby making decentralization dependent on signer governance and distribution. QBFT and IBFT 2.0 incur higher communication overhead because deterministic finality and Byzantine fault tolerance are maintained through quorum-based voting under the $N \geq 3f + 1$ condition. Ethash follows a distinct operating model in which computational competition and an honest-majority hash-power assumption replace validator-based voting, resulting in substantially higher computational cost, energy consumption, and transaction latency. The controlled four-node private-network implementation of Ethash with fixed mining difficulty characterizes performance and resource behavior under the benchmark configuration rather than the decentralization or adversarial security properties of a public PoW network. Clique's measured performance advantage within the evaluated permissioned environment is therefore attributable primarily to the lower coordination cost of the authorized-signer model and must be interpreted together with the corresponding validator-governance, trust, and finality assumptions.

Resource usage analysis enables the evaluation of performance not only through high-level outputs such as latency and throughput but also together with the computational load on the hardware. In particular, in corporate private blockchain applications, when selecting a consensus algorithm, metrics such as peak loads on the processor, characteristic fluctuation behavior, and the scalability of memory usage directly affect operational costs,

energy efficiency, and the design requirements of node architecture. When evaluated from this perspective, QBFT provides the most predictable structure operationally due to its CPU stability, while IBFT2 includes periodic CPU peak points despite its high performance.

Clique appears suitable for lightweight networks due to its relatively low CPU load. Ethash stands out as the most disadvantageous algorithm in terms of energy cost and resource efficiency because of its high CPU volatility.

Descriptive statistics (Tables 3–5 and 9) make the stability and uncertainty of the measurements visible. In terms of the repeated-run statistics, significant distinctions in performance stability can be seen. In regard to the approximate TPS, the values of CV for IBFT 2.0 and QBFT are 0.9% and 1.4%, respectively, which shows high stability in throughput performance across ten runs. However, the highest absolute throughput is shown by Clique, but it is more variable than other protocols (CV = 13.8%), while Ethash is not only the one with the lowest absolute throughput but also the one with the highest variability (CV = 46.7%). The same trend is shown by P95 latency, with Clique having the lowest relative variability (3.3%), then IBFT 2.0 (10.6%), QBFT (14.8%) and Ethash (37.7%). The Kruskal–Wallis test (Table 6) shows large, globally significant differences across all metrics among the algorithms. Dunn + Bonferroni post hoc analysis (Table 7) confirms which pairwise comparisons account for these differences. Cliff’s Delta (Table 8) further indicates that the effect sizes of these differences are very large. The choice of consensus algorithm is therefore a statistically validated and practically strong determinant of latency (P50/P95), throughput (approximate/peak TPS), and energy consumption.

Network traffic analysis results make it possible to evaluate consensus algorithms not only in terms of transaction latency or throughput but also with respect to network bandwidth requirements, inter-node communication costs, and transaction-load handling characteristics under the evaluated topology. For example:

- High communication intensity of Clique can lead to increased costs and greater bandwidth requirements in larger-scale networks.
- Ethash exhibits substantially higher peak-to-average network traffic variation than the other evaluated configurations, indicating less predictable short-term bandwidth demand under the tested workload.
- The balanced profile of QBFT and IBFT 2.0 can provide reliable and predictable network performance in corporate private blockchains.

When the inclusion latency analysis is examined, most of the latency values in the QBFT and IBFT 2.0 algorithms accumulate at low levels, and the distribution is right-skewed. This appearance shows that BFT-based algorithms can maintain system stability even under high transaction load, although latency spikes can occur at certain moments due to view-change events being triggered. Both IBFT 2.0 and QBFT provide deterministic finality; therefore, there is no rollback risk for confirmed blocks. In other words, this brings BFT protocols to the forefront in corporate blockchain applications where reliability is as critical as scalability. The latency behavior of Clique depends directly on the block scheduler. The deterministic block-production cycle minimizes latency variance while constraining the transaction inclusion time to the block period. Short-term latency increases can occur during leader rotation. Since the finality guarantee of PoA protocols is weaker compared with BFT protocols, they are generally not preferred in corporate applications requiring high security, but rather in test environments or private networks with lower security requirements. Ethash exhibits by far the lowest performance among the tested algorithms in terms of latency. The presence of both high and irregular latency values is a natural consequence of the random hash-solving process inherent in the PoW architecture. Due to probabilistic finality and large latency variance, the usability of Ethash in corporate

environments is limited. On the other hand, it can still be considered a strong option for security in public blockchains.

To avoid representing protocol security through a single ordinal category, Table 11 reports the consensus safety assumptions associated with each protocol. QBFT and IBFT 2.0 follow the classical Byzantine fault-tolerance condition $N \geq 3f + 1$, whereas Clique relies on an honest majority of authorized signers and Ethash relies on an honest-majority hash-power assumption. These entries describe protocol-level assumptions rather than experimentally measured security outcomes.

The experimental design does not include Byzantine fault injection, malicious-validator behavior, network-partition scenarios, or adversarial hash-power testing. Accordingly, the protocol-level safety assumptions reported in Table 11 represent analytical properties of the underlying consensus mechanisms rather than empirically validated security performance. The experimental evaluation is restricted to measurable operational characteristics, including latency, throughput, energy consumption, resource utilization, and network behavior under the defined benchmark configuration. Security-related interpretations are limited to the established fault models, trust assumptions, validator-admission structures, and finality properties associated with each protocol. Deployment-oriented conclusions should therefore be interpreted primarily in terms of performance and resource behavior under the evaluated conditions, while security-critical protocol selection requires separate adversarial and fault-injection validation.

Table 11. Operational Profiles and Consensus Safety Assumptions of the Evaluated Consensus Algorithms.

Feature	QBFT	Clique	IBFT 2.0	Ethash
CPU Stability	Stable	Fluctuating	Spiky behavior	Noisy
Memory Usage	Stable (42–43%)	Stable	Stable	Stable
Network Traffic	Medium	High	Medium	Sudden spikes
TPS	Medium	High	Medium	Low
Consensus Safety Assumption	<1/3 Byzantine validators ($N \geq 3f + 1$)	Honest majority of authorized signers ($f < N/2$)	($N \geq 3f + 1$)	Honest-majority hash-power assumption (adversarial hash power < 50%)
Potential Application Context	Permissioned networks prioritizing stable operational performance	Trusted private environments prioritizing throughput and low latency	Permissioned networks requiring deterministic finality and Byzantine fault-tolerance assumptions	Controlled PoW benchmarking and comparative evaluation

Table 11 summarizes the operational profiles and protocol-level consensus assumptions of the evaluated algorithms. The experimental results characterize CPU stability, memory usage, network traffic, and throughput, while the consensus safety assumptions describe the protocol conditions under which agreement is maintained.

7. Conclusions

Comprehensive performance measurements conducted on the same infrastructure have revealed that different consensus algorithms exhibit distinctly different profiles in terms of throughput, system resource usage, network traffic, and inclusion latency. Under the evaluated permissioned configuration, Clique achieved the highest measured throughput and lowest latency; however, this performance advantage is associated with a restricted authorized-signer model and should therefore be considered together with

its validator-governance, decentralization, and finality assumptions. QBFT exhibited a balanced operational profile under the evaluated configuration, with stable CPU behavior, steady memory consumption, and consistent network efficiency, making it relevant to permissioned deployments in which predictable resource behavior is an important consideration. IBFT 2.0 similarly provides deterministic finality and Byzantine fault-tolerance at the protocol level, although its suitability for security-critical deployments cannot be established from the present experiments and would require dedicated adversarial and fault-injection evaluation. Ethash, on the other hand, has exhibited energy-inefficient, unpredictable CPU behavior and a weak throughput performance profile due to the nature of its PoW architecture, indicating that it is not a practical candidate for modern enterprise, high-performance, or energy-sensitive blockchain environments. Overall, the results show that there is no single “best” approach in the selection of a consensus protocol, and the most suitable architecture must be determined in accordance with the security requirements, energy efficiency goals, and performance expectations of the application scenario.

Considering an engineering perspective, there is no one protocol that attains the state of absolute optimality. In the four-node burst workloads tested, Clique offers the best throughput and lowest latency. Under the evaluated configuration, this result demonstrates Clique’s strong throughput and latency profile for trusted, performance-oriented private environments in which its authorized-signer trust model and probabilistic finality are acceptable; however, Clique is not presented as a current Hyperledger Besu deployment recommendation because support for it has been removed from recent Besu releases. QBFT and IBFT 2.0 show comparable throughput under the evaluated workload and share the same Byzantine fault-tolerance threshold and deterministic-finality property, while exhibiting different latency variability and CPU-utilization profiles. Ethash, on the other hand, offers higher latency, variance, and resource consumption, serving as a benchmark for Proof of Work.

Such findings must be understood within the context of experimental limitations. The assessment makes use of one artificial workload, which includes 5000 transactions for value transfer sent as a burst to produce a stress condition for a write-intensive workload, for comparison of the four protocols under equal conditions, and not because the goal was to emulate transactions performed by any particular application. Therefore, smart contract execution, reading, and combined read/write transactions that would be seen in production systems are not included. The network consists of four co-located nodes with a two-second period between blocks, and therefore does not represent propagation delays for a wide area, which means that the relative performance of the protocols will vary in case of larger validator sets when the cost of BFT message exchanges becomes more significant. The number of transactions per one burst was kept constant and did not vary, which meant that per-protocol saturation points were not defined. In the Clique configuration, the maximum inclusion latency corresponds to a single block period, indicating that the transaction pool did not accumulate a backlog and that its reported throughput reflects the offered load rather than a capacity limit. Finally, security properties were defined based on the protocol properties described in the literature and were not found through any simulation of attacks. In future work, it is planned to first examine the behavior of consensus algorithms in different network topologies. In addition, incorporating next-generation consensus protocols into the experimental framework will make it possible to create a comparative performance map with existing algorithms. Moreover, building on recent multidimensional and decision-intelligent approaches to consensus selection [10,11], the development of artificial intelligence-based prediction models capable of forecasting metrics such as TPS, latency, resource consumption, and block finalization times will contribute to the creation

of intelligent decision-support systems that optimize consensus selection according to the application scenario.

Author Contributions: Conceptualization, M.F.Ö., A.A., M.K. and M.A.A.; methodology, M.F.Ö. and A.A.; software, M.F.Ö.; validation, M.F.Ö.; formal analysis, M.F.Ö.; investigation, M.F.Ö.; resources, A.A.; data curation, M.F.Ö.; writing—original draft preparation, M.F.Ö. and M.K.; writing—review and editing, M.F.Ö., A.A., M.K., M.A.A. and H.H.B.; visualization, M.F.Ö.; supervision, A.A. and H.H.B.; project administration, A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Scientific and Technological Research Council of Türkiye (TÜBİTAK) under grant no. TEYDEB-3240695. In addition, the R&D Center of Next4biz provided computational and technical infrastructure and resources used in conducting the experimental studies.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No public software repository or standalone dataset is associated with this study. The benchmark procedure, pseudocode, experimental configuration, software environment, workload parameters, monitoring setup, and statistical analysis methodology are described in the manuscript to support methodological transparency and independent reimplementations. The numerical results supporting the findings are reported within the manuscript.

Acknowledgments: This study is supported by the Scientific and Technological Research Council of Türkiye (TÜBİTAK) under grant no TEYDEB-3240695.

Conflicts of Interest: Mr. Muhammet Furkan Özara and Dr. Akhan Akbulut are employees of Next4biz. Dr. Akhan Akbulut is also affiliated with İstanbul Kültür University and serves as the Principal Investigator of the TÜBİTAK-funded project. Dr. Mustafa Kara and Dr. Hasan Hüseyin Balık received no financial compensation or other financial benefit related to this study, while Dr. Muhammed Ali Aydın contributed to the project in an academic advisory capacity. TÜBİTAK had no role in the design of the study; in the collection, analysis, or interpretation of the data; in the writing of the manuscript; or in the decision to publish the results. Apart from the relationships and support disclosed above, the authors declare no other conflicts of interest.

References

1. Jain, A.K.; Gupta, N.; Gupta, B.B. A survey on scalable consensus algorithms for blockchain technology. *Cyber Secur. Appl.* **2025**, *3*, 100065. [\[CrossRef\]](#)
2. Kara, M.; Laouid, A.; Hammoudeh, M.; AlShaikh, M.; Bounceur, A. Proof of chance: A lightweight consensus algorithm for the internet of things. *IEEE Trans. Ind. Inform.* **2022**, *18*, 8336–8345. [\[CrossRef\]](#)
3. Guo, H.; Yu, X. A survey on blockchain technology and its security. *Blockchain Res. Appl.* **2022**, *3*, 100067. [\[CrossRef\]](#)
4. Gao, G.; Guo, C.; Wan, X.; Xia, Z.; Shi, Y. Proof-of-GoS: An Efficient GoS-Based Consensus Algorithm for IOT. *IEEE Internet Things J.* **2025**, *12*, 30879–30890. [\[CrossRef\]](#)
5. Hao, Y.; Li, Y.; Dong, X.; Fang, L.; Chen, P. Performance analysis of consensus algorithm in private blockchain. In *Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV)*; IEEE: Piscataway, NJ, USA, 2018; pp. 280–285.
6. Chacko, N.M.; V G, N.; Balachandra, M.; T, M. Lightweight Consensus in Blockchain: A Systematic Literature Review. *ACM Comput. Surv.* **2025**, *58*, 1–37. [\[CrossRef\]](#)
7. Xiao, Y.; Zhang, N.; Lou, W.; Hou, Y.T. A survey of distributed consensus protocols for blockchain networks. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1432–1465. [\[CrossRef\]](#)
8. Chaudhry, N.; Yousaf, M.M. Consensus algorithms in blockchain: Comparative analysis, challenges and opportunities. In *Proceedings of the 2018 12th International Conference on Open Source Systems and Technologies (ICOSST)*; IEEE: Piscataway, NJ, USA, 2018; pp. 54–63.
9. Pradhan, N.R.; Singh, A.P.; Kumar, N.; Hassan, M.M.; Roy, D.S. A flexible permission ascription (FPA)-based blockchain framework for peer-to-peer energy trading with performance evaluation. *IEEE Trans. Ind. Inform.* **2021**, *18*, 2465–2475. [\[CrossRef\]](#)
10. Švarcmajer, M.; Kohler, M.; Krpić, Z.; Lukić, I. Consensus-Driven Framework for Data-Driven Optimization of Distributed Systems Through Blockchain Consensus Mechanism Selection. *Big Data Cogn. Comput.* **2026**, *10*, 154. [\[CrossRef\]](#)

11. Kumar, S.; Yadav, A.K. Blockchain-enabled IoT: A MADM and machine learning decision-intelligent framework for optimal consensus mechanism selection. *Future Gener. Comput. Syst.* **2026**, *185*, 108662. [\[CrossRef\]](#)
12. Delladetsimas, A.P.; Papangelou, S.; Iosif, E.; Giaglis, G. Leadership Uniformity in Timeout-Based Quorum Byzantine Fault Tolerance (QBFT) Consensus. *Big Data Cogn. Comput.* **2025**, *9*, 196. [\[CrossRef\]](#)
13. Xu, F.; Hu, S.; Sun, Y.; Hu, X.; Qi, J.; Sun, Y.; Dong, Z. FDSS: Flight data sharing scheme based on blockchain with dynamic, secure and efficient consensus algorithm. *Comput. Netw.* **2025**, *265*, 111275. [\[CrossRef\]](#)
14. Singh, A.; Kumar, G.; Saha, R.; Conti, M.; Alazab, M.; Thomas, R. A survey and taxonomy of consensus protocols for blockchains. *J. Syst. Archit.* **2022**, *127*, 102503. [\[CrossRef\]](#)
15. Hussein, Z.; Salama, M.A.; El-Rahman, S.A. Evolution of blockchain consensus algorithms: A review on the latest milestones of blockchain consensus algorithms. *Cybersecurity* **2023**, *6*, 30. [\[CrossRef\]](#)
16. Fan, C.; Lin, C.; Khazaei, H.; Musilek, P. Performance analysis of hyperledger besu in private blockchain. In *Proceedings of the 2022 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*; IEEE: Piscataway, NJ, USA, 2022; pp. 64–73.
17. Fan, C.; Ghaemi, S.; Khazaei, H.; Musilek, P. Performance evaluation of blockchain systems: A systematic survey. *IEEE Access* **2020**, *8*, 126927–126950. [\[CrossRef\]](#)
18. Bamakan, S.M.H.; Motavali, A.; Bondarti, A.B. A survey of blockchain consensus algorithms performance evaluation criteria. *Expert Syst. Appl.* **2020**, *154*, 113385. [\[CrossRef\]](#)
19. Yadav, A.K.; Singh, K.; Amin, A.H.; Almutairi, L.; Alsenani, T.R.; Ahmadian, A. A comparative study on consensus mechanism with security threats and future scopes: Blockchain. *Comput. Commun.* **2023**, *201*, 102–115. [\[CrossRef\]](#)
20. Lepore, C.; Ceria, M.; Visconti, A.; Rao, U.P.; Shah, K.A.; Zanolini, L. A survey on blockchain consensus with a performance comparison of PoW, PoS and pure PoS. *Mathematics* **2020**, *8*, 1782. [\[CrossRef\]](#)
21. Tkachuk, R.V.; Ilie, D.; Robert, R.; Kebande, V.; Tutschku, K. On the performance and scalability of consensus mechanisms in privacy-enabled decentralized renewable energy marketplace. *Ann. Telecommun.* **2024**, *79*, 271–288. [\[CrossRef\]](#)
22. Abdella, J.; Tari, Z.; Anwar, A.; Mahmood, A.; Han, F. An architecture and performance evaluation of blockchain-based peer-to-peer energy trading. *IEEE Trans. Smart Grid* **2021**, *12*, 3364–3378. [\[CrossRef\]](#)
23. Ferone, A.; Verrilli, S. Exploiting Blockchain Technology for Enhancing Digital Twins' Security and Transparency. *Future Internet* **2025**, *17*, 31. [\[CrossRef\]](#)
24. Yatnalli, V.; Bhusare, S.S.; Dhulavvagol, P.M.; Konnurmath, G.; Shetty, R. DABFT: A Novel Adaptive Byzantine Fault Tolerance Framework for High-Performance Blockchain Consensus. *Eng. Technol. Appl. Sci. Res.* **2025**, *15*, 25313–25317. [\[CrossRef\]](#)
25. Kurisaka, H.; Su, Y.; Nguyen, P.L.; Nguyen, K.; Sekiya, H. Performance Evaluation of Ethereum Consensus Mechanisms in IoT-Blockchain Systems Using Resource-Constrained Devices. *Clust. Comput.* **2025**, *28*, 763. [\[CrossRef\]](#)
26. Jung, S.; Yoo, Y.; Yang, G.; Yoo, C. Prediction of Permissioned Blockchain Performance for Resource Scaling Configurations. *ICT Express* **2024**, *10*, 1253–1258. [\[CrossRef\]](#)
27. Rao, I.S.; Kiah, M.L.M.; Hameed, M.M.; Memon, Z.A. Scalability of Blockchain: A Comprehensive Review and Future Research Direction. *Clust. Comput.* **2024**, *27*, 5547–5570. [\[CrossRef\]](#)
28. Gol, D.A.; Gondaliya, N. Blockchain: A comparative analysis of hybrid consensus algorithm and performance evaluation. *Comput. Electr. Eng.* **2024**, *117*, 108934. [\[CrossRef\]](#)
29. 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. [\[CrossRef\]](#)
30. Shi, Z.; Zhou, H.; Hu, Y.; Jayachander, S.; De Laat, C.; Zhao, Z. Operating permissioned blockchain in clouds: A performance study of hyperledger sawtooth. In *Proceedings of the 2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*; IEEE: Piscataway, NJ, USA, 2019; pp. 50–57.
31. Lohachab, A.; Garg, S.; Kang, B.H.; Amin, M.B. Performance evaluation of Hyperledger Fabric-enabled framework for pervasive peer-to-peer energy trading in smart Cyber-Physical Systems. *Future Gener. Comput. Syst.* **2021**, *118*, 392–416. [\[CrossRef\]](#)
32. Sghaier, R.; El Hog, C.; Djema, R.B.; Sliman, L. Machine Learning and Blockchain Synergy: Opportunities and Challenges for ML Models and Smart Contracts. *Blockchain Res. Appl.* **2025**, *100411*. [\[CrossRef\]](#)
33. Alsunaidi, S.J.; Alhaidari, F.A. A survey of consensus algorithms for blockchain technology. In *Proceedings of the 2019 International Conference on Computer and Information Sciences (ICCIS)*; IEEE: Piscataway, NJ, USA, 2019; pp. 1–6.
34. Nguyen, G.N.; Le Viet, N.H.; Elhoseny, M.; Shankar, K.; Gupta, B.; Abd El-Latif, A.A. Secure blockchain enabled Cyber-physical systems in healthcare using deep belief network with ResNet model. *J. Parallel Distrib. Comput.* **2021**, *153*, 150–160. [\[CrossRef\]](#)
35. Wang, Y.; Wang, H.; Cao, Y. Comprehensive review of storage optimization techniques in blockchain systems. *Appl. Sci.* **2024**, *15*, 243. [\[CrossRef\]](#)
36. Xiong, H.; Chen, M.; Wu, C.; Zhao, Y.; Yi, W. Research on progress of blockchain consensus algorithm: A review on recent progress of blockchain consensus algorithms. *Future Internet* **2022**, *14*, 47. [\[CrossRef\]](#)
37. Yu, S.; Qiao, Y.; Yang, F.; Bo, J. DPoW: A decentralized proof-of-work consensus mechanism for blockchain system. *Comput. Netw.* **2025**, *270*, 111490. [\[CrossRef\]](#)

38. Lu, S.P.; Lei, C.L.; Tsai, M.H. An efficient Proof-of-Authority consensus scheme against cloning attacks. *Comput. Commun.* **2024**, *228*, 107975. [[CrossRef](#)]
39. Salimitari, M.; Chatterjee, M.; Fallah, Y.P. A survey on consensus methods in blockchain for resource-constrained IoT networks. *Internet Things* **2020**, *11*, 100212. [[CrossRef](#)]
40. Kaur, S.; Chaturvedi, S.; Sharma, A.; Kar, J. A research survey on applications of consensus protocols in blockchain. *Secur. Commun. Netw.* **2021**, *2021*, 6693731. [[CrossRef](#)]
41. Jiang, W.; Wu, X.; Song, M.; Qin, J.; Jia, Z. Improved PBFT algorithm based on comprehensive evaluation model. *Appl. Sci.* **2023**, *13*, 1117. [[CrossRef](#)]
42. Islam, M.M.; Merlec, M.M.; In, H.P. A comparative analysis of proof-of-authority consensus algorithms: Aura vs Clique. In *Proceedings of the 2022 IEEE International Conference on Services Computing (SCC)*; IEEE: Piscataway, NJ, USA, 2022; pp. 327–332.
43. Wu, X.; Jiang, W.; Song, M.; Jia, Z.; Qin, J. An efficient sharding consensus algorithm for consortium chains. *Sci. Rep.* **2023**, *13*, 20. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.