Performance And Scalability Comparison of Raft And SmartBFT Consensus Mechanisms In A Private Blockchain Network

Umit Kilic, Mumine Kaya Keles

Computer Engineering Department, Adana Alparslan Turkes Science and Technology University, Adana, Turkiye (ukilic@atu.edu.tr, mkaya@atu.edu.tr)

Abstract: Blockchain's promise of trust and immutability has positioned it as a transformative technology in high-integrity domains. However, as adoption increases, challenges related to performance and node scalability become critical. This study presents the first empirical comparison of the Raft and SmartBFT consensus mechanisms within the Hyperledger Fabric framework. Key performance metrics, including latency, throughput, and CPU utilization, were evaluated. The results reveal a fundamental trade-off: while SmartBFT demonstrates higher resource consumption and limited scalability across increasing node counts, Raft provides greater efficiency and scalability, albeit with increased latency. Interestingly, the study also identifies unexpected performance behaviors of SmartBFT under specific conditions. These findings highlight the importance of selecting consensus algorithms based on application-specific requirements. All results are validated through statistical analysis. Overall, this work offers valuable insights into the design and optimization of Hyperledger Fabric-based data-sharing applications.

Keywords: blockchain, hyperledger fabric, raft, smartBFT, consensus mechanisms, performance.

1. INTRODUCTION

Blockchain is a relatively new technology that can provide solutions for efficient and secure data sharing across various domains, including healthcare (Azaria et al., 2016; Hasselgren et al., 2020; Xia et al., 2017), finance (Chen et al., 2020), the Internet of Things (Ghiro et al., 2021; Sigwart et al., 2019), and other fields (Chukwu and Garg, 2020; Islam et al., 2019). These capabilities stem from several inherent features of blockchain technology: anonymity, programmability, immutability, transparency, data provenance, decentralization, distributed ledger, and consensus (Hasselgren et al., 2020; Yaqoob et al., 2021).

The inherent features of blockchain have positioned it as a promising solution to the aforementioned areas. Consequently, efforts to optimize the system have intensified. Researchers are actively investigating strategies to enhance the scalability, privacy, latency, and resource efficiency of blockchain technology. Blockchain networks are classified as public, private, and consortium based on data access permissions (Tasca and Tessone, 2017; Zheng et al., 2017). This classification determines how nodes join the network and access data. In public blockchains, any node can participate and view records without prior authorization. In private blockchains, an internal node controls participation and data access. Consortium blockchains require identity registration and approval from governing entity (Zheng et al., 2017).

Private and consortium blockchain networks have attracted increased attention in data sharing due to their ability to enable fast transaction processing and privacy. Hyperledger (HL) ("Hyperledger Foundation," 2025), an open-source project supported by the Linux Foundation, is widely used for building private and consortium networks. It provides a modular framework to build enterprise blockchain solutions by incorporating various frameworks, tools, and libraries. The availability of multiple frameworks allows developers to select the one most suited to their specific requirements, as they differ in terms of architecture, features, and consensus mechanisms. This diversity facilitates the design, development, and deployment of customized blockchain networks.

HL Fabric (Hyperledger Fabric Docs, 2020), the first open-source distributed platform designed for deploying private and consortium blockchain networks, is among the most widely preferred frameworks within the HL project. Its modular architecture allows organizations to integrate different components based on specific application requirements. HL Fabric provides a high degree of control over access rights and permissions, making it particularly suitable for industries that demand strict privacy and regulatory compliance. HL Fabric includes a variety of configurable parameters, such as the consensus mechanism, endorsement policy, block size, block timeout, and channels, all of which must be properly configured to achieve optimal performance. Among these parameters, the selection of an appropriate consensus mechanism is of particular significance. Consensus is a fundamental component of the network, exerting a considerable influence on its overall performance (Chung et al., 2019). The processes for updating data and adding the block to the network are concluded using consensus algorithms. As well as completing transactions, it also plays a vital role in ensuring consistency and security. It is therefore effective in terms of network performance and scalability (Xiong et al., 2022).

Given the significant impact of consensus mechanisms on performance and scalability, this study focuses on evaluating a blockchain network designed for data sharing under different consensus protocols. As of the current release (version 3.0), HL Fabric supports two consensus algorithms: Raft and SmartBFT, the latter having been recently integrated into the

platform. HL Fabric was selected for this analysis due to its features. In addition to these features, HL Fabric's modular architecture facilitates the seamless interchange of consensus algorithms, supports the use of multiple general-purpose programming languages rather than one platform-specific language for smart contract development, and is accompanied by comprehensive documentation.

Although many studies have explored the performance of HL Fabric from various perspectives, including comparing versions, evaluating specific use cases, analyzing architecture, and investigating consensus mechanisms, none have directly compared the performance of Raft and SmartBFT, the two consensus algorithms officially supported in Hyperledger Fabric v3.0. This study seeks to address that gap by systematically examining the performance and scalability of Raft and SmartBFT under the same network conditions. Existing studies have evaluated the performance of blockchain networks built on HL Fabric from various perspectives. Initial research primarily focused on benchmarking throughput and latency under standardized workloads (Baliga et al., 2018; Nasir et al., 2018; Nguyen et al., 2019). Other studies investigated the impact of architectural parameters, such as block size and endorsement policies, identifying them as key bottlenecks in large-scale deployments (Gorenflo et al., 2020; Kuzlu et al., 2019; Piao et al., 2022). More recently, with the introduction of SmartBFT capabilities in HL Fabric, researchers have begun exploring the trade-offs between Crash Fault Tolerant (CFT) protocols like Raft and SmartBFT solutions (Salih and Wang, 2024). These studies contribute to a deeper understanding of HL Fabric performance evaluation by employing various approaches. However, to the best of our knowledge, no prior research has conducted a comprehensive scalability comparison specifically focusing on the newly integrated SmartBFT against the established Raft orderer, which remains a critical gap that this study aims to fill. Evaluating the HL Fabric blockchain network supports blockchain researchers and developers in selecting the most appropriate consensus algorithm for specific use cases.

This paper aims to provide a comprehensive performance evaluation of two consensus algorithms in the HL Fabric blockchain network, conducting experiments with various transaction numbers and network sizes. To ensure the reliability and reproducibility of the findings, the results are presented with standard deviation (std) values and supported by a rigorous statistical analysis including the Mann-Whitney U test and effect size (r) calculations. This analysis aims to identify which algorithm is more suitable based on the specific priorities and requirements of the target system, distinguishing between statistical significance and practical relevance.

The performance analysis in this study was conducted under controlled network conditions, focusing on a baseline environment without node failures or adversarial behaviors. Furthermore, the experiments were implemented on a single physical machine using Docker. While these setup establishes a robust baseline comparison by focusing on node scalability under stable transaction load, it serves as a necessary foundation for future stress tests aimed at identifying network saturation points and throughput bottlenecks in large-scale, geo-distributed deployments.

The research questions addressed in this study are as follows:

What are the differences between the Raft and SmartBFT consensus algorithms in terms of performance and scalability?

How do these algorithms respond to changes in network conditions?

What are the most suitable use cases for each consensus algorithm?

The rest of the paper is structured as follows. Section 2 presents related studies from literature. Section 3 provides a brief introduction to HL Fabric. Additionally, two consensus mechanisms, Raft and SmartBFT, are explained in the same section. Section 4 includes methodology and experimental setup. Section 5 has results and their discussion. Finally, the paper concluded, and future work is presented in Section 6.

2. RELATED WORK

Despite the availability of various blockchain platforms, a standardized methodology for evaluating their performance has yet to be established (Al-Sumaidaee et al., 2023; Fan et al., 2020; Nasir et al., 2018). This limitation also applies to networks built on the same platform. It is widely acknowledged that blockchain technology remains in its early stages of development. Consequently, studies in this domain are limited in number, and experimental setups differ significantly across literature. As HL Fabric was selected as the platform for this study, the scope is restricted to evaluation studies conducted using the same framework.

Several studies have explored the performance of networks built on HL Fabric under various conditions and configurations. For instance, in (Nasir et al., 2018), the authors performed two experiments: the first comparing performance between versions v0.6 and v1.0, and the second assessing scalability by increasing the number of nodes to 20. Performance was measured using HL Caliper, focusing on execution time, latency, and throughput. The results indicated that while v1.0 outperformed v0.6 in terms of performance, it demonstrated lower scalability. (Al-Sumaidaee et al., 2023) analyzed a private HL Fabric test network for healthcare applications by measuring the performance of create and read operations under varying transaction rates, worker counts, and transaction per second (TPS). Latency, throughput, and send rate were used as performance metrics. Their experiments, conducted using fixed- and linear-rate controllers configured through Caliper, revealed the best-case scenarios for these operations and provided insights into how TPS and workload characteristics impact performance. In (Chacko et al., 2023), investigated optimization strategies for HL Fabric-based blockchain networks in the context of supply chain management, digital rights management, and electronic health records. The evaluations were made based on success rate and latency metrics. The network was created on HL Fabric and various optimizations were made such as model running, changing transaction rate, block size adaptation, and so on. Consequently, transaction rate control and smart contract partitioning were among the suggested optimizations. The experimental results demonstrated that the implemented recommendations led to improvements of 20% and 40% in terms of success rate and latency, respectively. Related to this

line of research, (Lin et al., 2024) proposed an auto-tuning configuration optimization framework for further enhancing performance under dynamic conditions. In a modeling-based approach, (Melo et al., 2024) proposed a performance model for HL Fabric, considering indicators such as throughput, utilization, and mean response time. Their analysis across four use cases highlighted the influence of parameters like block size and timeout on throughput and network delay.

At the architectural level, a number of studies have addressed overall system design and its effect on performance. (Jain and Doria, 2023) conducted a comparative analysis of two consensus algorithms, Kafka and Solo, on an HL Fabric-based network for robotic path planning. The network was created for sharing information in the domain of path planning to avoid collision between robots. Their findings indicated that Kafka offered better performance in terms of latency and consensus time, while Solo was more CPU-efficient. Similarly, (Baliga et al., 2018) studied the performance and scalability of HL Fabric v1.0, showing that reducing the number of endorsements could enhance performance but potentially compromise security.

Other performance-focused studies investigated modular components and state databases in HL Fabric. For example, (Wen and Hsu, 2023) evaluated HL Fabric under high-concurrency transaction scenarios, identifying bottlenecks and demonstrating superior performance with MySQL-based state databases compared to CouchDB and LevelDB. In addition, architectural-level evaluations were conducted in (Gorenflo et al., 2020; Javaid et al., 2019; Kuzlu et al., 2019). (Kuzlu et al., 2019) examined how network load, smart contract complexity, and hardware configuration affect throughput, latency, and scalability. (Javaid et al., 2019) proposed a modified validation phase to improve overall performance, while (Gorenflo et al., 2020) presented architectural improvements that significantly increased throughput, from 3,000 to 20,000 transactions per second. (Yuan et al., 2020) further explored the performance impact of different ordering service configurations using a model to depict transaction flow.

Consensus mechanisms, being a core component of blockchain platforms, have also been explored, although to a lesser extent within the HL Fabric ecosystem. (Piao et al., 2022) investigated the impact of block size on consensus latency under the Raft algorithm, concluding that block size significantly affects performance. (Tende et al., 2021) compared Raft and the now-deprecated Kafka algorithm in terms of CPU and memory usage, reporting that Raft was more resource-efficient. (Sukhwani et al., 2017) evaluated the Practical Byzantine Fault Tolerance (PBFT) mechanism in HL Fabric v0.6, analyzing how consensus time scales with network size. (Nguyen et al., 2019) also explored PBFT within HL Fabric and observed a negative impact of network delay on performance.

Despite these valuable contributions, only a few studies have mentioned SmartBFT, and none, to the best of our knowledge, have directly compared Raft and SmartBFT in the context of Hyperledger Fabric. One technical report by (Salih and Wang, 2024) proposed a BFT-based consensus protocol called BDLS and compared it to Raft and SmartBFT using TPS as the primary metric under both LAN and WAN conditions. Although their work includes both Raft and SmartBFT, the primary focus was the performance of their proposed protocol, rather than a detailed comparative evaluation of the two existing algorithms in HL Fabric.

While numerous studies have examined the performance of HL Fabric from different angles, including version comparisons, use-case-specific evaluations, architectural analyses, and consensus investigations, none have provided a direct performance comparison between Raft and SmartBFT, the two consensus algorithms officially supported in Hyperledger Fabric v3.0. This study aims to fill that gap by systematically analyzing the performance and scalability characteristics of Raft and SmartBFT under identical network conditions.

3. BACKGROUND

This section provides an overview of Hyperledger Fabric and its associated consensus mechanisms.

3.1 Hyperledger Fabric

HL Fabric, one of the most widely adopted sub-frameworks within HL ecosystem, consists of several core components that are essential to the construction, maintenance, and security of the network (Hyperledger Fabric Docs, 2020). These components include channels, peers, organizations, smart contracts, certificate authorities, orderers, and the consensus mechanism, among others. Each element contributes to the overall functionality and reliability of the blockchain infrastructure (Zand et al., 2021). By integrating these components, HL Fabric offers a robust and scalable infrastructure for the implementation of permissioned blockchain networks tailored to enterprise needs. As of version 3, released in September 2024, HL Fabric supports two consensus mechanisms: Raft and SmartBFT (Hyperledger Fabric Docs, 2025).

HL Fabric offers several advantages over other blockchain platforms, making it a suitable choice for this study. Unlike platforms such as Ethereum (Buterin, n.d.) and Corda ("R3 Documentation," 2025), which follow more monolithic designs with fixed components, HL Fabric adopts a modular architecture with pluggable services. For example, its ordering service operates independently from the validation process, a separation often absent or unclear in platforms like Ethereum, Bitcoin (Nakamoto, 2008) and Corda. Another key advantage is language flexibility: HL Fabric supports smart contracts in multiple languages (Go, Java, Node.js), allowing developers to use familiar programming environments. In contrast, platforms such as Ethereum require the use of domain-specific languages (e.g., Solidity), which may increase the risk of programming errors. Additionally, HL Fabric's modularity enables customization of its consensus mechanism, unlike most platforms where consensus logic is hard-coded.

Despite its strengths, HL Fabric presents certain challenges. Its complexity, although well-documented, demands a strong understanding of its architecture. Compared to its competitors, it has a steeper learning curve due to its distinctive design. Furthermore, deploying a network requires extensive configuration and planning, making troubleshooting difficult

without adequate technical expertise (Androulaki et al., 2018; Buterin, n.d.; Hyperledger Fabric Docs, 2025; Nakamoto, 2008; "R3 Documentation," 2025; Thakkar et al., 2018; Valenta and Sandner, n.d.).

To evaluate the performance of the proposed network, this study employs HL Caliper ("Caliper," n.d.), a benchmarking tool designed for systematic assessment of blockchain systems. Caliper, written in JavaScript, measures four core metrics under predefined test conditions: transaction latency, throughput, send rate, and resource utilization. A final report summarizes these metrics, along with success and failure counts. Explanations of these terms are given in the next section.

3.2. Consensus mechanisms: Raft and SmartBFT

In blockchain systems, consensus refers to the process of reaching an agreement on values, enabling secure and consistent updates to the ledger (Lashkari and Musilek, 2021). Consensus algorithms are broadly categorized into two groups: Byzantine fault tolerant and non-Byzantine fault tolerant (non-BFT) (Bano et al., 2017), depending on whether they can tolerate malicious behavior from nodes.

Typically, non-BFT algorithms address only crash faults and are referred to as crash fault tolerant. A crash occurs when a node suddenly stops functioning and fails to resume operation. In contrast, BFT algorithms are designed to handle nodes that act maliciously, for instance, by sending conflicting or inconsistent messages, remaining dormant for long periods before reactivating, or otherwise attempting to disrupt the system (Zhou et al., 2023). Examples of CFT-based, non-BFT algorithms include Viewstamped Replication (Oki and Liskov, 1988), Paxos (Lamport, 1998), and Raft (Ongaro and Ousterhout, 2014). In the BFT category, commonly used algorithms include Proof of Work (Jakobsson and Juels, 1999), Proof of Stake (King and Nadal, 2012), Delegated Proof of Stake (Bach et al., 2018), Proof of Activity (Bentov et al., 2014), and SmartBFT (Bessani et al., 2014).

Consensus mechanisms play a foundational role in blockchain systems. They are responsible for validating transactions, grouping them into blocks, and ensuring the correct and secure updating of the distributed ledger. Due to its critical impact on the security, consistency, and performance of blockchain networks, consensus is often supported by an incentive mechanism, particularly in cryptocurrency ecosystems like Bitcoin and Ethereum, rewarding participants, also known as miners, for contributing to the process. The primary goal of a consensus algorithm is to ensure secure and consistent data synchronization across the network. In HL Fabric v3.0, two consensus algorithms are supported: Raft and SmartBFT. These will be examined in detail in the following subsections.

3.2.1. Raft algorithm

Raft is a CFT, non-BFT consensus algorithm based on log replication, proposed by Ongaro and Ousterhout (Ongaro and Ousterhout, 2014). It introduces a structured system of node roles, leaders, followers, and candidates, which can dynamically transition between states. The leader coordinates log replication and distribution, while followers passively receive information from the leader. Candidates, promoted by followers, may become the new leader if elected.

The algorithm has two main phases: leader election and log replication. During the leader election phase, all nodes start as followers. Each follower waits for a randomized timeout period; if it does not receive a heartbeat from the current leader during this interval, it transitions into a candidate. The candidate votes for itself and sends vote requests to other nodes. If it secures votes from a majority of the nodes, it becomes the new leader. Conversely, if another node claims leadership or the candidate fails to achieve a majority, the election fails (Ongaro and Ousterhout, 2014; Zhou et al., 2023). Following a successful election, the system enters the log replication phase. The client sends its request to the leader, which then creates new log entries and broadcasts them to followers in real time. Followers update their local logs and send acknowledgments back to the leader. Once the leader receives confirmations from a majority of the nodes, consensus is achieved, and the client is notified (Huang et al., 2020; Ongaro and Ousterhout, 2014; Singh et al., 2022). This process illustrated in Fig. 1.

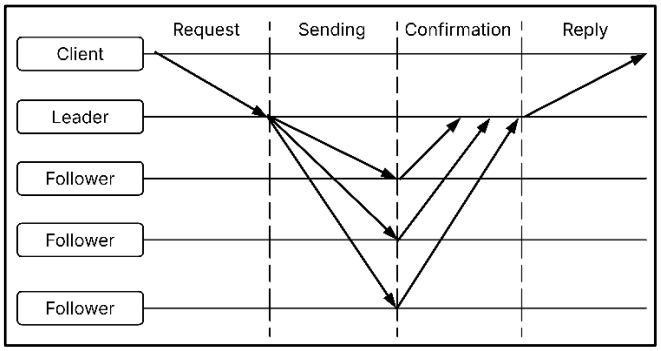

Fig. 1. The workflow of log replication in Raft.

In the event of a system failure that prevents the leader from reaching a majority of followers, it continues updating the nodes it can reach. Meanwhile, a new leader election is automatically triggered. After the system stabilizes, the former leader reverts to a follower role. Its uncommitted updates are rolled back, and it synchronizes with the new leader to ensure consistency and integrity (Huang et al., 2020; Ongaro and Ousterhout, 2014). To function correctly, Raft requires a minimum of $2f + 1$ nodes, where f is the number of nodes that may fail. Therefore, to tolerate a single node failure ($f = 1$), at least three ordering nodes must be present in the network.

3.2.2. SmartBFT algorithm

SmartBFT is a consensus algorithm derived primarily from BFT-SMaRt (Bessani et al., 2014; Sousa and Bessani, 2012) and PBFT (Castro and Liskov, 1999), with several enhancements. It has been implemented and integrated into HL Fabric by Barger et al (Barger et al., 2021; Manevich et al., 2025). BFT-SMaRt was proposed to address several limitations of PBFT: PBFT does not leverage the capabilities of modern hardware, suffers from numerous software bugs, is no longer actively maintained, and lacks a widely adopted implementation suitable for building services (Bessani et al., 2014). SmartBFT extends BFT-SMaRt by incorporating HL

Fabric-specific optimizations and is tailored for integration with HL Fabric.

In SmartBFT, consensus nodes are categorized into two types: primary nodes and replica nodes. Primary nodes function as leaders responsible for packaging transactions into blocks and finalizing them, while replica nodes assist in the block finalization process. Each consensus round includes exactly one primary node and multiple replica nodes. A leader election is conducted at the beginning of every consensus process (Bessani et al., 2014).

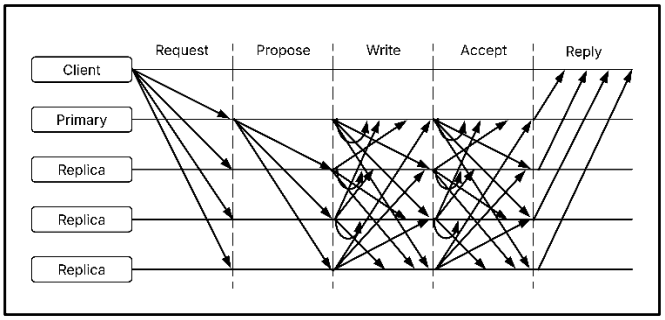

Fig. 2. The messaging process of SmartBFT (Bessani et al., 2014).

As illustrated in Fig. 2, the message flow begins when the client sends a request to the consensus nodes. Only the primary node processes the request; replica nodes initially ignore it. The primary node verifies the request and then broadcasts it to the replica nodes in the Propose phase. Subsequently, an all-to-all communication occurs in the Write and Accept phases. During these stages, replica nodes validate the message and broadcast corresponding verification messages. If the majority of nodes broadcast the same message, a Commit message is issued. Once commit messages from a majority are received, the proposed changes, initially cached, are committed to the ledger. The client is then notified of the successful transaction.

SmartBFT adheres to the fault tolerance rule of $3f + 1$, where f denotes the maximum number of Byzantine faulty nodes tolerated. Thus, to withstand one faulty node ($f = 1$), a minimum of four ordering nodes is required in the network (Barger et al., 2021; Bessani et al., 2014; Sousa and Bessani, 2012). SmartBFT was implemented to address the lack of BFT consensus algorithm in HL Fabric, despite previous attempts. It is designed as a potential upgrade to Raft, introducing improvements such as a customizable interface, enhanced validation mechanisms, refined communication phases, and seamless integration with the HL Fabric framework. Additionally, SmartBFT aims to optimize resource utilization by offloading fundamental tasks, such as cryptographic operations and point-to-point communication, to the underlying framework (Barger et al., 2021).

When comparing Raft and SmartBFT, their key differences can be summarized as follows (Castro and Liskov, 1999; Sousa and Bessani, 2012; Bessani et al., 2014; Barger et al., 2021; Xiong et al., 2022):

Messaging Complexity: As shown in Fig. 2, SmartBFT has a higher messaging complexity of $O(n^2)$, compared to Raft's $O(n)$. This increased complexity may lead to higher network traffic and resource consumption, particularly in large-scale deployments.

Fault Tolerance: SmartBFT can tolerate both crash and Byzantine faults, whereas Raft is limited to crash fault tolerance. This is because the BFT model assumes that nodes may behave maliciously or go silent for extended periods, a behavior also encompassing crash faults. However, this increased fault tolerance comes at the cost of higher resource requirements: SmartBFT requires $3f + 1$ nodes to tolerate f faults, while Raft only requires $2f + 1$.

Leader Election Strategy: Raft uses randomized leader elections to prioritize responsiveness, whereas SmartBFT employs a deterministic leader rotation mechanism, placing greater emphasis on security and correctness.

Table 1. Parameter settings for the experiments.

	Experiment 1 (Performance)		Experiment 2 (Scalability)	
	Parameter	**Value(s)**	**Parameter**	**Value(s)**
Fixed parameters	Peer No	2	No. of transactions	1000
	TPS	100	TPS	100
Changing parameters	Consensus Mechanism	Raft, SmartBFT	Consensus Mechanism	Raft, SmartBFT
	No. of transactions	100, 500, 1000, 2500, 5000, 10000, 20000, 40000	Peer No	2, 4, 6, 8, 10, 12, 14, 16, 18, 20

Trust Model: Both algorithms, Raft and SmartBFT, are leader-based approaches. However, the followers in Raft always rely on the leader, whereas SmartBFT's replica nodes operate under the assumption that the leader may be faulty. This leads to additional validation steps

4. METHODOLOGY

This section outlines the methodology employed in the study. The network architecture sub section details the structure of the HL Fabric-based network developed for the experiments. The Experimental setup subsection describes the hardware and software environments, configuration parameters, and

network specifications used during testing. The Evaluation metrics subsection explains the performance indicators collected to compare consensus algorithms.

4.1. The network architecture

In our implementation, a HL Fabric network was configured with two organizations, each comprising one peer node. During the experiments, the number of peers gradually increased to evaluate scalability. In terms of ordering service, three orderer nodes were deployed for Raft and four for SmartBFT, based on the minimum operational requirements of each consensus algorithm as discussed earlier.

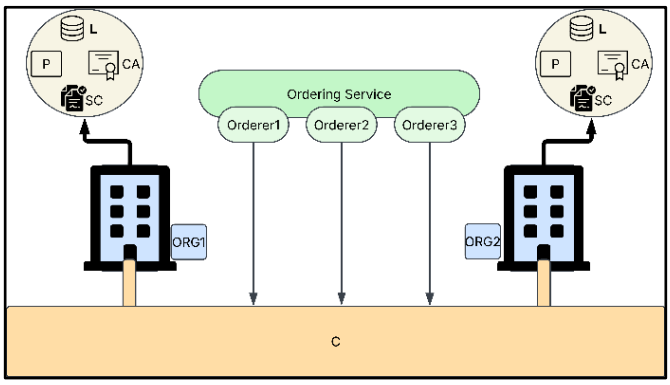

Fig. 3. the structure of the system built on the HL Fabric.

The overall system architecture is illustrated in Fig. 3, which represents the network structure using Raft as the consensus mechanism. The architecture for SmartBFT is identical, with the exception of the number of orderer nodes. In SmartBFT, four orderers are required to meet its fault-tolerance threshold.

In Fig. 3, the following abbreviations are used: SC (Smart Contract), L (Ledger), CA (Certificate Authority), P (Peer), C (Channel), and ORG (Organization). Organizations in this context may represent a variety of real-world entities such as research institutes, universities, research centers, or hospitals. Conceptually, an organization can either consist of a single institution or be formed through the collaboration of multiple entities.

4.2. Experimental setup

HL Caliper is a benchmarking framework designed for performance evaluation of private blockchain networks. It requires several configuration files, including a network configuration file, a workload file, and a benchmark configuration file. The network configuration file defines the network components (e.g., channels, organizations, peers) and is essential for accessing the blockchain during testing. The workload file determines the type of transactions to be executed and outlines how to interact with the deployed smart contract. Lastly, the benchmark file specifies measurement parameters such as control rate, TPS, links to workload modules, and monitoring configurations.

To conduct a comprehensive evaluation, various network configurations were tested. Transactions were submitted at a constant rate of 100 TPS, as recommended in (Chacko et al., 2023). The system's performance was evaluated under two experimental conditions. For Experiment 1 (Performance Evaluation), the total number of transactions (tx no) was incrementally increased while keeping the number of peers and TPS constant. The initial tx no was set to 100 and increased to 40,000. Each experiment was repeated ten times to compute average values and reduce variability. For Experiment 2 (Scalability Evaluation), tx no and TPS were kept constant, while the number of peers gradually increased from 2 to 20. One peer was added to each organization in every iteration. The system's behavior in response to an increasing number of nodes was used to assess scalability. The parameter settings for both experiments are shown in Table 1. To ensure the transparency and reproducibility of the experiments, other HL Fabric v3.0 network parameters than those mentioned in Table 1 were kept at their default production-ready values. These configurations are summarized in Table 2. Maintaining these settings across both Raft and SmartBFT scenarios allowed for a fair comparison of the consensus-specific performance overhead.

Table 2. Default values of other parameters.

Parameter	Value
Database	LevelDB
Block Size (Max message count)	10
Batch Timeout	2 seconds
Preferred Max Bytes	512 KB
Endorsement Policy	MAJORITY

The smart contract developed for this study was written in Go and includes functions for data sharing, such as querying and data creation. Query operations are considered read-only and are performed on committed ledger data. Since read operations do not alter the ledger, they do not require consensus. Therefore, only the results from write operations were considered for performance evaluation. All experiments were conducted on a local testbed with the following specifications: a 12-core Intel Core i7-12700F CPU (base clock 2.1 GHz, max turbo 4.9 GHz), 32 GB RAM, and Ubuntu 22.04 (Linux). HL Fabric version 3.0 was used. All Fabric components were deployed in Docker containers on the test machine. Docker (Merkel, 2014) is an open-source platform that enables the development, deployment, and management of applications within lightweight containers.

Three performance metrics generated by HL Caliper were used to evaluate the network: average latency, average throughput, and average CPU usage. Throughput refers to the average number of transactions per transaction. Latency denotes the time elapsed between submitting a transaction and receiving a response. CPU usage reflects the proportion of CPU resources consumed during execution ("Caliper," n.d.).

Table 3. Average latency, throughput, and CPU usage (± std) obtained from performance tests.

Tx No.	Avg. Latency (s)		Avg. Throughput (TPS)		Avg. CPU Usage (%)	
	RAFT	**SmartBFT**	**RAFT**	**SmartBFT**	**RAFT**	**SmartBFT**
100	0.067 (±0.005)	0.050 (±6.9e-18)	99.20 (±0.205)	96.860 (±1.583)	4.984 (±0.211)	5.273 (±0.148)
500	0.060 (±1.4e-17)	0.050 (±6.9e-18)	99.85 (±0.05)	99.270 (±0.293)	12.611 (±0.328)	13.660 (±0.219)
1000	0.063 (±0.005)	0.052 (±0.006)	99.92 (±0.04)	99.640 (±0.18)	14.631 (±0.162)	15.111 (±0.227)
2500	0.070 (±0.014)	0.050 (±6.9e-18)	99.96 (±0.08)	99.910 (±0.07)	17.249 (±0.127)	18.275 (±0.291)
5000	0.081 (±0.019)	0.075 (±0.0.033)	99.95 (±0.05)	99.900 (±0.078)	18.419 (±0.180)	19.492 (±0.212)
10000	0.080 (±0.023)	0.067 (±0.018)	99.98 (±0.04)	99.980 (±0.04)	18.503 (±0.199)	19.463 (±0.124)
20000	0.074 (±0.012)	0.081 (±0.016)	100 (±0)	99.950 (±0.05)	18.984 (±0.163)	19.734 (±0.103)
40000	0.060 (±1.4e-17)	0.064 (±0.020)	100 (±0)	100 (±0)	19.198 (±0.122)	20.041 (±0.112)

These metrics provide insights into the performance efficiency and resource utilization of the blockchain network. To ensure statistical reliability and minimize the effect of outliers or chance-based anomalies, each experiment was repeated ten times, and the average values were calculated.

To assess the significance of the observed differences between results, the Mann–Whitney U test (Mann and Whitney, 1947) was employed. Also known as the Wilcoxon rank-sum test, it is a non-parametric statistical test used to evaluate whether there is a significant difference between two independents sample groups. The resulting p-value from the test indicates the level of statistical significance. If the p-value is greater than 0.05, it suggests that there is no statistically significant difference between the two groups. Conversely, a p-value less than or equal to 0.05 indicates a statistically significant difference (MacFarland and Yates, 2016; McKnight and Najab, 2010). Furthermore, the effect size (r) was calculated to quantify the magnitude of these differences and determine their practical relevance, using formula $r = |Z|/\sqrt{N}$, where Z is the test statistics and N is the total number of the observations. According to the Cohen's guidelines, r values of 0.1, 0.3, and 0.5 represent small, medium, and large effect sizes, respectively (Cohen, 1988). The results of the statistical analysis are presented in the corresponding result tables.

5. RESULTS AND DISCUSSION

This section presents the analysis of system performance and scalability under varying network conditions. The corresponding results are provided in tables and discussed.

5.1. Performance comparison

The performance of the system built on HL Fabric is evaluated using three previously defined metrics: latency (s), throughput (TPS), and CPU usage. Each experiment is executed ten times to calculate the average values for these metrics. Table 3 shows the average latency, throughput, and CPU usage along with their std values for both consensus algorithms. The low std across all metrics indicate high experimental consistency and the reproducibility of the results. In general, Raft exhibits higher latency than SmartBFT for the majority of transaction numbers. Nevertheless, the difference between the two algorithms is minimal in magnitude. Although the workload has increased, the negligible impact on latency suggests that the network, and consequently Raft and SmartBFT, demonstrate sufficient robustness to manage this workload. While the latency value of SmartBFT remains constant at 0.05 for 1000 and 2500 transaction numbers (with very low variance), the latency value of Raft varies between 0.063 and 0.07. As a result, SmartBFT offers reduced latency at low transaction numbers; however, Raft is more stable and provides an advantage at high transaction numbers. The latency results conflict with traditional theoretical expectations that SmartBFT should be slower because it requires more message exchanges, given that it is a BFT-based algorithm. These results can be explained by the optimized communication and verification stages in the SmartBFT implementation, which enhance effectiveness at low scales. However, this conflict needs to be investigated in more detail, and this will be addressed in our future work. SmartBFT falls behind after 20,000 transactions, which indicates that it has reached its optimization limits at high transaction numbers.

With regard to average throughput, SmartBFT exhibits lower efficiency than Raft at low transaction counts (e.g., 96.86% for 100 transaction counts) yet approaches Raft as the number of transactions increases. As the number of transactions increases, the difference falls below 0.1 TPS, and the narrowing std values further confirm that both algorithms reach a stable performance plateau. These results are in line with expectations. Raft utilizes a less complex protocol, which results in a reduced network load.

At 40,000 transactions, Raft uses 19.198% and SmartBFT uses 20.041%. SmartBFT consistently has a higher CPU cost for all transaction levels. Raft is more efficient in terms of CPU utilization. As transaction numbers grow, CPU usage increases for both algorithms. SmartBFT consistently uses 5-10% more CPU than Raft, likely due to its complex consensus protocol. Compared to Raft, SmartBFT shows higher CPU utilization across all transaction volumes. For instance, at 100 transactions, Raft uses 4.984% CPU while SmartBFT uses 5.273%

Table 4. Average, median, p-values, and effect size of all performance experiments.

Criteria	p value	Effect size (r)	Average		Median	
			RAFT	SmartBFT	RAFT	SmartBFT
Latency (s)	2.16e-09	1.34	0.069	0.061	0.060	0.050
Throughput (TPS)	0.003395	0.65	99.857	99.439	100	99.900
CPU Usage (%)	0.000543	0.77	15.572	16.381	17.760	19.080

As can be seen from Table 4, the p-value of latency, throughput, and CPU usage is lower than 0.05, the standard significance threshold. This provides evidence that a statistically significant difference in latency, throughput, and CPU usage exists between Raft and SmartBFT. The p-value confirms that the latency, throughput, and CPU usage outputs are not due to a random sampling. Furthermore, the calculated effect size (r) values for all criteria are above 0.5, indicating a large practical significance. Notably, the latency metric exhibits an extremely large effect size (r =1.34), which demonstrates that the performance difference between the two algorithms is not only statistically significant but also substantial in real-world applications.

5.2. Scalability comparison

This subsection comprises a scalability comparison of Raft and SmartBFT algorithms. Initially, the network comprises two organizations, with one peer from each organization. The number of peers is gradually increasing for each organization. The total number of peers on the network commences at 2 and increases up to 20. In all scalability tests, the number of transactions is set to 1,000, and the TPS value is set at 100, as illustrated in Table 1.

Table 5. Average latency, throughput, and CPU usage (± std) obtained from scalability tests.

Peer No.	Avg. Latency (s)		Avg. Throughput (TPS)		Avg. CPU Usage (%)	
	RAFT	SmartBFT	RAFT	SmartBFT	RAFT	SmartBFT
2	0.063 (±0.005)	0.052 (±0.006)	99.920 (±0.04)	99.640 (±0.18)	14.631 (±0.162)	15.111 (±0.227)
4	0.071 (±0.02)	0.050 (±6.9e-18)	99.800 (±0.2)	99.670 (±0.155)	14.760 (±0.169)	15.456 (±0.256)
6	0.062 (±0.004)	0.050 (±6.9e-18)	99.900 (±1.4e-14)	99.600 (±0.167)	14.884 (±0.186)	15.637 (±0.094)
8	0.070 (±1.4e-17)	0.050 (±6.9e-18)	99.900 (±1.4e-14)	99.570 (±0.119)	15.127 (±0.19)	15.775 (±0.256)
10	0.070 (±1.4e-17)	0.050 (±6.9e-18)	99.900 (±1.4e-14)	99.610 (±0.158)	15.670 (±0.202)	16.253 (±0.129)
12	0.070 (±1.4e-17)	0.050 (±6.9e-18)	99.890 (±0.03)	99.560 (±0.143)	15.752 (±0.155)	16.274 (±0.162)
14	0.070 (±1.4e-17)	0.058 (±0.004)	99.880 (±0.04)	99.590 (±0.145)	15.959 (±0.165)	16.581 (±0.141)
16	0.071 (±0.03)	0.055 (±0.005)	99.850 (±0.05)	99.560 (±0.15)	16.466 (±0.195)	16.650 (±0.209)
18	0.076 (±0.005)	0.054 (±0.005)	99.810 (±0.03)	99.630 (±0.155)	16.750 (±0.152)	16.766 (±0.141)
20	0.099 (±0.02)	0.127 (±0.04)	99.260 (±0.8)	98.770 (±0.49)	16.791 (±0.155)	16.555 (±0.272)

Table 5 shows the average latency, throughput, and CPU usage along with their std values. In general, SmartBFT offers lower latency than Raft in scalability tests. However, when the number of peers reaches 20, the latency value of SmartBFT suddenly increases to 0.127s (an increase of about 135%), while the latency value of Raft increases to 0.099s (an increase of about 30%). Despite the sudden spikes, the low std values for both algorithms at the 20-peer mark confirm that these observations are consistent across experimental runs rather than being outliers. This point may represent a critical scalability threshold for SmartBFT, where complexity overwhelms improvements. This point is included in future

work to further investigate with a wider range of node numbers.

According to the throughput value, Raft is more stable and offers higher values. As the number of peers increases, minor decreases are observed in both algorithms. However, Raft continues to demonstrate superior performance. The minimal std observed in throughput results further underscores the operational stability of Raft across all peer counts. The rate of decrease of SmartBFT accelerates at a point when the number of peers is 20, and this is directly related to the increase in the latency value at the same point. As nodes spend more time engaged in communication and consensus-building processes, the number of transactions that can be processed per second is reduced. Consequently, Raft exhibited superior stability in terms of throughput value and obtained better outcomes in comparison with SmartBFT.

The CPU usage of SmartBFT is generally higher than that of Raft. For example, when the number of peers is 2, SmartBFT consumes 3% more CPU, and when the number of peers is 10, it consumes 3.7% more CPU. However, in the scenario involving 20 peers, Raft experiences slightly more CPU usage for the first time, while SmartBFT's CPU usage decreases. This unexpected decrease in SmartBFT's CPU usage is interesting. While SmartBFT's CPU usage decreases, the latency increases and throughput decreases at the same time. The consistency of the std values across these metrics suggests that this shift is a systematic behavior of the implementation at this scale. This situation suggests that the system has become network-bound rather than CPU-bound. In other words, nodes spend more time on completing communication rather than using the CPU for processing.

Table 6 summarizes comparative performance metrics and statistical outputs. SmartBFT outperforms Raft with a lower average (60 ms vs. 72 ms) and median latency (50 ms vs. 70 ms). The p-value (5.55e-24) indicates that this difference is highly statistically significant. Furthermore, the extremely large effect size (r =2.26) confirms that the latency advantage of SmartBFT is practically substantial and consistently observed across the scalability tests.

Table 6. Average, median, p-values, and effect size of all scalability experiments.

Criteria	p value	Effect size (r)	Average		Median	
			RAFT	SmartBFT	RAFT	SmartBFT
Latency (s)	5.55e-24	2.26	0.072	0.060	0.070	0.050
Throughput (TPS)	4.35e-23	2.21	99.811	99.520	99.900	99.600
CPU Usage (%)	0.00018	0.84	15.679	16.106	15.740	16.245

The throughput for Raft is 99.811 TPS, and for SmartBFT it is 99.520 TPS. The statistical test reveals a highly significant result (p = 4.35e-23). This is accompanied by another extremely large effect size (r =2.21), which suggests that Raft consistently maintains higher transaction rates in scalable environments, likely due to its more straightforward consensus process. The CPU utilization exhibited an increase for SmartBFT (16.1% vs. 15.7% average), which is consistent with the anticipated outcomes given the incorporation of cryptographic operations and the enhanced message complexity. Despite the negligible absolute difference, the p-value (0.00018) confirms statistical significance, while the effect size (r =0.84) reinforces that this resource consumption gap is a meaningful characteristic of the two algorithms as the network scales.

6. CONCLUSION

In this paper, a study on consensus algorithms, which constitute a fundamental component, is presented. A comparison of two consensus algorithms in terms of performance and scalability was conducted on the blockchain network implemented using the Hyperledger Fabric framework. In this context, two consensus algorithms supported by Hyperledger Fabric, Raft and SmartBFT, were utilized. The evaluation of the constructed networks was performed using the Caliper framework under the aegis of the Hyperledger project. The network monitoring was carried out via Hyperledger Explorer. In this study, latency, throughput, and CPU usage metrics obtained from Hyperledger Caliper were used as evaluation criteria in all experiments. Each experiment was repeated ten times to mitigate the influence of chance on the results. The inclusion of standard deviation for all measurements confirmed the high consistency and reproducibility of the experimental data. Although most networks in the literature have been deployed with a single orderer node, to ensure fairness, the network was operated with a number of nodes meeting the minimum requirements for each algorithm, preventing any advantage for either.

The results obtained are largely consistent with the theoretical characteristics of the algorithms. The performance and scalability outcomes indicated that Raft provides superior throughput and CPU utilization, whereas SmartBFT is distinguished by its lower latency values. However, the findings of the experiments also revealed surprising deviations from the theoretical expectations. Notably, SmartBFT exhibited superior latency compared to Raft at smaller scales and for fewer than 10,000 transactions per second. Theoretically, Raft is a simpler and less complex algorithm than SmartBFT. Furthermore, during scalability tests involving 20 peers, SmartBFT showed a decline in CPU utilization. These unexpected outcomes can be attributed to the optimized implementation of SmartBFT within Hyperledger Fabric. These intriguing results must be tested with more comprehensive and detailed experiments which are planned for future studies. The 20-peer configuration in the scalability test provides valuable insights regarding the network size to be established. This value may indicate the point at which

algorithm complexity becomes so dominant that it has a detrimental effect on performance, despite the improvements made in SmartBFT. The results of the tests performed were supported by Mann-Whitney U statistical test and supplemented with effect size calculations. It was shown that a statistical difference existed in all results, with large effect sizes confirming the practical relevance of the observed performance gaps.

In summary, the findings demonstrate that SmartBFT's low latency makes it well-suited for time-sensitive applications, whereas Raft is more suitable for large-scale and resource-constrained environments. Therefore, no single consensus algorithm can be considered universally optimal; rather, the selection should be guided by the specific requirements of the given scenario. The decision-making process must consider network priorities such as latency and resource efficiency.

The experiments were conducted under ideal network conditions, with no node failures or adversarial behavior introduced. This represents a limitation of the study, as real-world environments may exhibit less predictable and more hostile conditions. Another limitation is that the findings are derived from a single-node multi-container setup, which may not fully reflect the complexities of wide-area network latencies. Therefore, the findings represent a baseline performance comparison, and the scalability results should be interpreted within the limitations. Additionally, while the current work focuses on node scalability, future research will incorporate stress tests under increasing transaction loads to identify network saturation points. Furthermore, future studies will evaluate these consensus mechanisms in geographically distributed environments and under various fault scenarios, bridging the gap between controlled baseline studies and complex real-world blockchain deployments.

REFERENCES

Al-Sumaidaee, G., Alkhudary, R., Zilic, Z., Swidan, A., 2023. Performance analysis of a private blockchain network built on Hyperledger Fabric for healthcare. *Information Processing & Management* 60, 103160. https://doi.org/10.1016/j.ipm.2022.103160

Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., Muralidharan, S., Murthy, C., Nguyen, B., Sethi, M., Singh, G., Smith, K., Sorniotti, A., Stathakopoulou, C., Vukolić, M., Cocco, S.W., Yellick, J., 2018. Hyperledger fabric: a distributed operating system for permissioned blockchains, in: Proceedings of the Thirteenth EuroSys Conference. Presented at the *EuroSys '18: Thirteenth EuroSys Conference 2018, ACM*, Porto Portugal, pp. 1–15. https://doi.org/10.1145/3190508.3190538

Azaria, A., Ekblaw, A., Vieira, T., Lippman, A., 2016. MedRec: Using Blockchain for Medical Data Access and Permission Management, in: 2016 *2nd International Conference on Open and Big Data* (OBD). pp. 25–30. https://doi.org/10.1109/OBD.2016.11

Bach, L.M., Mihaljevic, B., Zagar, M., 2018. Comparative analysis of blockchain consensus algorithms, in: 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). Presented at the 2018 *41st International Convention on Information and Communication Technology, Electronics and Microelectronics* (MIPRO), pp. 1545–1550. https://doi.org/10.23919/MIPRO.2018.8400278

Baliga, A., Solanki, N., Verekar, S., Pednekar, A., Kamat, P., Chatterjee, S., 2018. Performance Characterization of Hyperledger Fabric, in: 2018 *Crypto Valley Conference on Blockchain Technology* (CVCBT). IEEE, Zug, pp. 65–74. https://doi.org/10.1109/CVCBT.2018.00013

Bano, S., Sonnino, A., Al-Bassam, M., Azouvi, S., McCorry, P., Meiklejohn, S., Danezis, G., 2017. Consensus in the Age of Blockchains. *https://doi.org/10.48550/ARXIV.1711.03936*

Barger, A., Manevich, Y., Meir, H., Tock, Y., 2021. A Byzantine Fault-Tolerant Consensus Library for Hyperledger Fabric, in: 2021 *IEEE International Conference on Blockchain and Cryptocurrency* (ICBC). IEEE, Sydney, Australia, pp. 1–9. https://doi.org/10.1109/ICBC51069.2021.9461099

Bentov, I., Lee, C., Mizrahi, A., Rosenfeld, M., 2014. Proof of Activity: Extending Bitcoin's Proof of Work via Proof of Stake [Extended Abstract]y. *SIGMETRICS Perform*. Eval. Rev. 42, 34–37. https://doi.org/10.1145/2695533.2695545

Bessani, A., Sousa, J., Alchieri, E.E.P., 2014. State Machine Replication for the Masses with BFT-SMART, in: 2014 *44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. IEEE*, Atlanta, GA, USA, pp. 355–362. https://doi.org/10.1109/DSN.2014.43

Buterin, V., n.d. *Ethereum White Paper.*

Caliper [WWW Document], n.d. URL *https://www.lfdecentralizedtrust.org/projects/caliper (accessed 12.23.24).*

Castro, M., Liskov, B., 1999. Practical Byzantine Fault Tolerance. OsDI 99, 173–186.

Chacko, J.A., Mayer, R., Jacobsen, H.-A., 2023. How To Optimize My Blockchain? A Multi-Level Recommendation Approach. *Proc. ACM Manag. Data* 1, 1–27. https://doi.org/10.1145/3588704

Chen, L., Cong, L.W., Xiao, Y., 2020. A Brief Introduction to Blockchain Economics, in: Information for Efficient Decision Making. *WORLD SCIENTIFIC*, pp. 1–40. https://doi.org/10.1142/9789811220470_0001

Chukwu, E., Garg, L., 2020. A Systematic Review of Blockchain in Healthcare: Frameworks, Prototypes, and Implementations. I*EEE Access* 8, 21196–21214. https://doi.org/10.1109/ACCESS.2020.2969881

Chung, G., Desrosiers, L., Gupta, M., Sutton, A., Venkatadri, K., Wong, O., Zugic, G., 2019. Performance Tuning and Scaling Enterprise Blockchain Applications. *https://doi.org/10.48550/ARXIV.1912.11456*

Cohen, J., 1988. Statistical power analysis for the behavioral sciences, 2nd ed. ed. *Lawrence Erlbaum Associates, Hillside*, NJ.

Fan, C., Ghaemi, S., Khazaei, H., Musilek, P., 2020. Performance Evaluation of Blockchain Systems: A

Systematic Survey. *IEEE Access* 8, 126927–126950. https://doi.org/10.1109/access.2020.3006078

Ghiro, L., Restuccia, F., D'Oro, S., Basagni, S., Melodia, T., Maccari, L., Cigno, R.L., 2021. What is a Blockchain? A Definition to Clarify the Role of the Blockchain in the Internet of Things. *https://doi.org/10.48550/ARXIV.2102.03750*

Gorenflo, C., Lee, S., Golab, L., Keshav, S., 2020. FastFabric: Scaling hyperledger fabric to 20 000 transactions per second. Int J Network Mgmt 30, e2099. *https://doi.org/10.1002/nem.2099*

Hasselgren, A., Kralevska, K., Gligoroski, D., Pedersen, S.A., Faxvaag, A., 2020. Blockchain in healthcare and health sciences—A scoping review. *International Journal of Medical Informatics* 134, 104040. https://doi.org/10.1016/j.ijmedinf.2019.104040

Huang, D., Ma, X., Zhang, S., 2020. Performance Analysis of the Raft Consensus Algorithm for Private Blockchains. *IEEE Trans. Syst. Man Cybern, Syst.* 50, 172–181. https://doi.org/10.1109/TSMC.2019.2895471

Hyperledger Fabric Docs, 2025. What's new in Hyperledger Fabric — Hyperledger Fabric Docs main documentation [WWW Document]. URL *https://hyperledger-fabric.readthedocs.io/en/latest/whatsnew.html (accessed 2.17.25).*

Hyperledger Fabric Docs, 2020. A Blockchain Platform for the Enterprise — Hyperledger Fabric Docs main documentation [WWW Document]. A Blockchain Platform for the Enterprise — Hyperledger Fabric Docs main documentation. *URL https://hyperledger-fabric.readthedocs.io/en/latest/ (accessed 2.17.25).*

Hyperledger Foundation, 2025.

Islam, N., Faheem, Y., Din, I.U., Talha, M., Guizani, M., Khalil, M., 2019. A blockchain-based fog computing framework for activity recognition as an application to e-Healthcare services. Future Generation Computer Systems 100, 569–578. *https://doi.org/10.1016/j.future.2019.05.059*

Jain, S., Doriya, R., 2023. Performance evaluation of hyperledger fabric-based consensus mechanism on multi-robot path planning. Multimed Tools Appl 83, 15769–15783. *https://doi.org/10.1007/s11042-023-16341-6*

Jakobsson, M., Juels, A., 1999. Proofs of Work and Bread Pudding Protocols(Extended Abstract), in: Preneel, B. (Ed.), Secure Information Networks. Springer US, Boston, MA, pp. 258–272. *https://doi.org/10.1007/978-0-387-35568-9_18*

Javaid, H., Hu, C., Brebner, G., 2019. Optimizing Validation Phase of Hyperledger Fabric, in: 2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS). *IEEE, Rennes,* FR, pp. 269–275. https://doi.org/10.1109/MASCOTS.2019.00038

King, S., Nadal, S., 2012. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake.

Kuzlu, M., Pipattanasomporn, M., Gurses, L., Rahman, S., 2019. Performance Analysis of a Hyperledger Fabric Blockchain Framework: Throughput, Latency and Scalability, in: 2019 *IEEE International Conference on Blockchain* (Blockchain). IEEE, Atlanta, GA, USA, pp. 536–540. https://doi.org/10.1109/Blockchain.2019.00003

Lamport, L., 1998. The part-time parliament. *ACM Trans. Comput.* Syst. 16, 133–169. https://doi.org/10.1145/279227.279229

Lashkari, B., Musilek, P., 2021. A Comprehensive Review of Blockchain Consensus Mechanisms. *IEEE Access* 9, 43620–43652. https://doi.org/10.1109/ACCESS.2021.3065880

Lin, J., Deng, R., Lu, Z., Zhang, Y., Duan, Q., 2024. TuneChain: An Online Configuration Auto-Tuning Approach for Permissioned Blockchain Systems, in: *2024 IEEE International Conference on Web Services* (ICWS). IEEE, Shenzhen, China, pp. 512–523. https://doi.org/10.1109/ICWS62655.2024.00072

MacFarland, T.W., Yates, J.M., 2016. Mann–Whitney U Test, in: Introduction to Nonparametric Statistics for the Biological Sciences Using R. S*pringer International Publishing,* Cham, pp. 103–132. https://doi.org/10.1007/978-3-319-30634-6_4

Manevich, Y., Meir, H., Barger, A., Tock, Y., 2025. hyperledger-labs/SmartBFT [WWW Document]. Github - hyperledger-labs/SmartBFT: Implementation of the SmartBFT. URL *https://github.com/hyperledger-labs/SmartBFT* (accessed 5.8.25).

Mann, H.B., Whitney, D.R., 1947. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics* 18, 50–60.

McKnight, P.E., Najab, J., 2010. Mann-Whitney U Test, in: Weiner, I.B., Craighead, W.E. (Eds.), T*he Corsini Encyclopedia of Psychology.* Wiley, pp. 1–1. https://doi.org/10.1002/9780470479216.corpsy0524

Melo, C., Gonçalves, G., Silva, F.A., Soares, A., 2024. A comprehensive hyperledger fabric performance evaluation based on resources capacity planning. *Cluster Comput* 27, 12395–12410. https://doi.org/10.1007/s10586-024-04591-4

Merkel, D., 2014. Docker: lightweight Linux containers for consistent development and deployment. *Linux J.* 2014, 2:2.

Nakamoto, S., 2008. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review* 21260.

Nasir, Q., Qasse, I.A., Abu Talib, M., Nassif, A.B., 2018. Performance Analysis of Hyperledger Fabric Platforms. Security and Communication Networks 2018, 1–14. *https://doi.org/10.1155/2018/3976093*

Nguyen, T.S.L., Jourjon, G., Potop-Butucaru, M., Thai, K.L., 2019. Impact of network delays on Hyperledger Fabric, in: I*EEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops* (INFOCOM WKSHPS). IEEE, Paris, France, pp. 222–227. https://doi.org/10.1109/INFCOMW.2019.8845168

Oki, B.M., Liskov, B.H., 1988. Viewstamped replication: A New Primary CopyMethod to Support Highly-Available Distributed Systems, in: *Proceedings of the Seventh Annual ACM Symposium on Principles of Distributed Computing* - PODC '88. ACM Press, Toronto, Ontario, Canada, pp. 1–7. https://doi.org/10.1145/62546.62549

Ongaro, D., Ousterhout, J., 2014. In search of an understandable consensus algorithm, in: 2014 *USENIX Annual Technical Conference* (USENIX ATC 14). pp. 305–319.

Piao, X., Li, M., Meng, F., Song, H., 2022. Latency Analysis for Raft Consensus on Hyperledger Fabric, in: Svetinovic, D., Zhang, Y., Luo, X., Huang, X., Chen, X. (Eds.), Blockchain and Trustworthy Systems, Communications in Computer and Information Science. *Springer Nature Singapore, Singapore*, pp. 165–176. https://doi.org/10.1007/978-981-19-8043-5_12

R3 Documentation [WWW Document], 2025. . R3 Documentation. URL *https://docs.r3.com/ (accessed 2.26.25).*

Salih, A.A., Wang, Y., 2024. BDLS as a Blockchain Finality Gadget: Improving Byzantine Fault Tolerance in Hyperledger Fabric. *IEEE Access* 12, 154600–154613. https://doi.org/10.1109/ACCESS.2024.3481319

Sigwart, M., Borkowski, M., Peise, M., Schulte, S., Tai, S., 2019. Blockchain-based Data Provenance for the Internet of Things, in: Proceedings of the 9th International Conference on the Internet of Things. Presented at the IoT 2019: *9th International Conference on the Internet of Things,* ACM, Bilbao Spain, pp. 1–8. https://doi.org/10.1145/3365871.3365886

Singh, A., Kumar, G., Saha, R., Conti, M., Alazab, M., Thomas, R., 2022. A survey and taxonomy of consensus protocols for blockchains. J*ournal of Systems Architecture* 127, 102503. https://doi.org/10.1016/j.sysarc.2022.102503

Sousa, J., Bessani, A., 2012. From Byzantine Consensus to BFT State Machine Replication: A Latency-Optimal Transformation, in: 2012 *Ninth European Dependable Computing Conference.* IEEE, Sibiu, pp. 37–48. https://doi.org/10.1109/EDCC.2012.32

Sukhwani, H., Martinez, J.M., Chang, X., Trivedi, K.S., Rindos, A., 2017. Performance Modeling of PBFT Consensus Process for Permissioned Blockchain Network (Hyperledger Fabric), in: 2017 *IEEE 36th Symposium on Reliable Distributed Systems* (SRDS). IEEE, Hong Kong, Hong Kong, pp. 253–255. https://doi.org/10.1109/SRDS.2017.36

Tasca, P., Tessone, C.J., 2017. Taxonomy of Blockchain Technologies. Principles of Identification and Classification. *https://doi.org/10.48550/ARXIV.1708.04872*

Tende, I.G., Aburada, K., Yamaba, H., Katayama, T., Okazaki, N., 2021. Performance Evaluation of Blockchain Based Agricultural Input Voucher System, in: 2021 *IEEE 10th Global Conference on Consumer Electronics* (GCCE). IEEE, Kyoto, Japan, pp. 637–638. https://doi.org/10.1109/GCCE53005.2021.9622001

Thakkar, P., Nathan, S., Viswanathan, B., 2018. Performance Benchmarking and Optimizing Hyperledger Fabric Blockchain Platform, in: 2018 *IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication* Systems (MASCOTS). IEEE, Milwaukee, WI, pp. 264–276. https://doi.org/10.1109/MASCOTS.2018.00034

Valenta, M., Sandner, P., n.d. Comparison of Ethereum, *Hyperledger Fabric and Corda.*

Wen, Y.-F., Hsu, C.-M., 2023. A performance evaluation of modular functions and state databases for Hyperledger Fabric blockchain systems. *J Supercomput* 79, 2654–2690. https://doi.org/10.1007/s11227-022-04762-3

Xia, Q., Sifah, E.B., Asamoah, K.O., Gao, J., Du, X., Guizani, M., 2017. MeDShare: Trust-Less Medical Data Sharing Among Cloud Service Providers via Blockchain. *IEEE Access* 5, 14757–14767. https://doi.org/10.1109/ACCESS.2017.2730843

Xiong, H., Chen, M., Wu, C., Zhao, Y., Yi, W., 2022. Research on Progress of Blockchain Consensus Algorithm: A Review on Recent Progress of Blockchain Consensus Algorithms. *Future Internet* 14, 47. https://doi.org/10.3390/fi14020047

Yaqoob, I., Salah, K., Jayaraman, R., Al-Hammadi, Y., 2021. Blockchain for healthcare data management: opportunities, challenges, and future recommendations. *Neural Computing and Applications* 1–16.

Yuan, P., Zheng, K., Xiong, X., Zhang, K., Lei, L., 2020. Performance modeling and analysis of a Hyperledger-based system using GSPN. *Computer Communications* 153, 117–124. https://doi.org/10.1016/j.comcom.2020.01.073

Zand, M., Wu, X., Morris, M., 2021. Hands-On Smart Contract Development with Hyperledger Fabric V2, 1st edition. ed. *O'Reilly Media, Inc*.

Zheng, Z., Xie, S., Dai, H., Chen, X., Wang, H., 2017. An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends, in: 2017 *IEEE International Congress on Big Data* (BigData Congress). IEEE, Honolulu, HI, USA, pp. 557–564. https://doi.org/10.1109/BigDataCongress.2017.85

Zhou, S., Li, K., Xiao, L., Cai, J., Liang, W., Castiglione, A., 2023. A Systematic Review of Consensus Mechanisms in Blockchain. *Mathematics* 11, 2248. https://doi.org/10.3390/math11102248