International Journal of Applied Mathematics
Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

COMPARATIVE ANALYSIS OF HYPERLEDGER FABRIC PERFORMANCE ACROSS VARIOUS ORDERING SERVICES

Dr. Pravin Jangid[1], Dr. Namdeo B. Badhe[2], Dr. Nupur Giri[3], Dr. Vinayak Ashok Bharadi[4]

Dept. Of Computer Engineering, *Shree L.R. Tiwari College of Engineering*, Thane, India

Dept. Of Information Technology, *Thakur College of Engineering and Technology*, Mumbai, India

Dept. Of Computer Engineering, *Vivekanand Education Society's Institute of Technology*, Mumbai, India

Dept. Of Information Technology, *Finolex Academy of Management and Technology*, Ratnagiri, India

pravinjangid@gmail.com, namdeob.badhe@thakureducation.org, nupur.giri@ves.ac.in, vinayak.bharadi@famt.ac.in

Abstract

Hyperledger Fabric is a well-known open-source initiative for deploying permissioned blockchain networks. The performance features of Hyperledger Fabric v2.2, such as the influence of ordering services, bottlenecks, and scalability, are difficult to understand due to the performance constraints of distributed systems. The consensus method used in Hyperledger Fabric to handle node failures while maintaining data consistency and integrity is critical in all blockchain services. This paper examines the resource usage and extensive performance evalua- tions of three key ordering services (Solo, Raft, and Kafka). The performance of the Hyperledger fabric is specifically examined in terms of throughput, latency, network size, scalability, and the number of peers serviced by the platform. The experimental results show that the proposed framework can undertake exten- sive performance evaluations of blockchain systems for large- scale applications. All three ordering services performed rather well, and we examine why Raft may be the best option for most organisations. This evaluation helps to clarify some of the trade- offs in Hyperledger Fabric v2.**x.**

***Index Terms*—**Blockchain, Consensus algorithm, Hyperledger Fabric, Hyperledger Caliper, Performance analysis, Kafka, Raft, Solo.

I. INTRODUCTION

Blockchain, derived from Bitcoin [1], is a set of continu- ously expanding data and transaction records called blocks, which are cryptographically linked and secured. It provides a decentralised network with records that are tamper-resistant and traceable.

Received: August 09 2025

International Journal of Applied Mathematics
Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Blockchains are immutable digital ledger sys- tems that are distributed (without a single repository) and often lack a central authority. Blockchain networks let trustworthy parties to transmit and receive transactions in a peer-to- peer way that is verifiable, eliminating the need for trusted middlemen. As a result, blockchain technology enables parties to settle transactions more quickly, resulting in faster asset transfers.

A blockchain is an open, distributed ledger that is hosted by a network of decentralised devices known as nodes. Blockchain transactions between two or more parties (whether trusted or not) can be recorded in a verifiable and immutable manner, reducing the risk of fraud. When transactions occur, additional blocks are created to record the information, the ledger is updated with these blocks, and the updated ledger is synced across all nodes in a blockchain system. A blockchain network may be either permissionless or permissioned. A permissionless or public network, such as Bitcoin or Ethereum [2], allows anybody to become a participant and make trans- actions. Due to the immense amount of nodes in a public

network, a proof-of-work consensus mechanism is employed to arrange transactions and build blocks. The network is more secure than permissioned blockchains due to consensus methods like proof of work and proof of stake, making it less vulnerable to 51% attacks or double spending ([3]). In a permissioned network, each participant's identity is known and cryptographically confirmed, allowing the blockchain to store who did which transactions. Furthermore, a blockchain network can have built-in access control methods to limit who can read and write data in the ledger, issue transactions, and participate in the network. A permissioned network is well-suited for enterprise applications that require authen- ticated participants. Each node in a permissioned network can be owned by multiple organisations. Enterprises value the ability to integrate multiple systems without having to develop a centralised solution, as well as establishing trust among untrustworthy partners or to bringing in a trusted third party. There is a lot of concern regarding the performance of permissioned blockchain networks, particularly their ability to manage a large volume of transactions while maintaining low latency.

Analysing and optimising performance is crucial in a digital context where blockchain solutions are becoming more com- mon. This study delves into the performance evaluation of the Fabric blockchain technology, examining how configura- tion factors and consensus mechanisms affect throughput and latency. This study uses controlled experimentation and thor- ough evaluation to find insights critical for informed decision- making and the growth of blockchain frameworks. The study provides actionable insights by evaluating configuration fac- tors, consensus mechanisms, and basic metrics like as through- put and latency. These insights support informed decision- making and enhance the Hyperledger Fabric [4] framework to meet real-world expectations. The primary purpose of this re- search is to do a thorough assessment of the Fabric platform's throughput and latency under different usage patterns, setup parameters, and consensus mechanisms. It seeks to obtain a thorough understanding of how various configuration settings, such as block size and batch size, affect the platform's perfor- mance metrics. Furthermore, the study investigates the effects of using various consensus algorithms, such as Kafka-based consensus and Raft, on

Received: August 09 2025

International Journal of Applied Mathematics
Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

the Fabric platform's performance. This investigation aims to provide insights and recommen- dations for optimising the Fabric blockchain's performance by discovering parameter combinations that result in best

throughput and minimal latency. The analysis also looks into the scalability aspects of Fabric's performance, specifically how the platform's behaviour changes as the system scales to accommodate more participants and transactions.

This study provides useful assistance to decision-makers in designing platforms for optimal efficiency. It makes recom- mendations for improving performance, highlighting on high throughput and low latency. It also delves into scalability insights, which aid in understanding system behaviour as Hyperledger Fabric scales for sustained performance. This research advances the development of Hyperledger Fabric by providing data-driven insights. Furthermore, its conclusions apply to a variety of businesses that require precise perfor- mance estimates. Finally, it is an educational resource for students, researchers, and practitioners of blockchain technol- ogy. The research's motivation lies in the need to unlock the full capabilities of the Fabric blockchain. As businesses use blockchain more, it's really important to have customized per- formance. Understanding how parameter adjustments and con- sensus choices impact efficiency and scalability is essential. This research provides decision-makers with valuable insights to configure things effectively, contributing to the advancement of blockchain technology's practical applications.

The remaining sections of this work are organised as follows. Section II introduces the relevant works. Section III provides an overview of Hyperledger fabric; Section IV covers the ordering services used in Hyperledger fabric. Section V describes configuration options and necessary metrics. Section VI includes the evaluation and analysis of the results, while Section VII presents the future work. Finally, Section VIII closes our paper.

II. Related Works

There has been considerable interest in the scalability and performance characteristics of public blockchain networks and private specifically the limiting factor of the consensus protocol and its security implications. [5] shows that scalability is a big challenge to blockchain impeding its mainstream adoption and gives a systematic review of blockchain scal- ability. Study stated the fact that scalability issues are due to lower throughput, high latency, large storage, and low read- performance. Since consensus is one of the key problems in blockchains, and there are several candidate consensus algorithms for public and consortium/private blockchains with different security and performance levels study [6] presents a simple but accurate analytical model to analyze the distributed network split probability as a function of the network size, the packet loss rate, and the election timeout period. [7] study also compares the Raft algorithm with Apache Kafka, a distributed messaging system which, implements many concepts present in Raft (strong leadership, append-only, log replication), shows that mechanisms conceived to handle one class of problems are very useful to handle a larger category in the context of distributed systems.

Due to the blockchain's limited capability to process sub- stantial transaction requests

Received: August 09 2025

Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

from a massive number of IoTdevices, study [8] presents a performance monitoring model to manage overhead, enhance system performance details, and improve scalability. This framework assesses how different network workloads affect HLF's performance in large-scale distributed setups. Evaluation parameters include throughput, latency, network size, scalability, and the platform's capacity to serve multiple peers. In conclusion, the study's bench- mark findings suggest that deploying HLF on compact IoT devices with restricted CPU and power capabilities is generally challenging. The current IoT device landscape often lacks the resources required for blockchain integration. Thus, a more suitable strategy involves implementing the blockchain on edge devices. Similarly there have been many perfor- mance evaluation and analysis in other fields such as supply chain management for traceability this study [9] introduces a complete food traceability system that can associate the enterprises of the whole circulation of food from production to consumption. UAV has always been a hot topic for data integrity and security this study [10] propose an effective traffic management system for drones capable of handling simple paths and deconfliction of multiple drones. In the field of healthcare author [11] enriches the field by comparing fixed and linear controllers through multiple experiments, high- lighting parameter effects and optimal scenarios. Importantly, the study unveils that higher worker counts escalate latency and risk network overload. Additionally, surpassing a certain TPS threshold can cause transaction failures, harming network throughput. In the realm of benchmarking blockchain perfor- mance, the choice of rate controllers from Hyperledger Caliper is crucial for managing transaction send rates effectively. An- other study for hyperledger fabric as a multimedia system [12] ensures multimedia data security using encryption techniques and access control methods. The performance evaluation of the research is measured based on various parameters such as transaction latency, throughput, asset latency, upload and download time, and power consumption for multimedia data. There are different studies to evaluate the performance of Ordering Services and Endorsement Policies in Hyper- ledger Fabric [13] research analyzes the efficiency of in- dividual stages within Hyperledger Fabric's novel execute- order-validate structure. Concludes that the execution phase exhibits better scalability under the OR endorsement policy compared to the AND endorsement policy. While all three ordering services (Solo, Raft, Kafka) displayed satisfactory performance, author also deduced that Raft might be the optimal selection for the majority of organizations. Similarly the study [14] outlines a two-phased strategy. The first phase aims to assess how different configuration settings, like block size, endorsement policy, and resource allocation, impact transaction speed and delay. In the second phase, Hyperledger Fabric v1.0 is optimized based on observations. The study conclude that hyperledger fabric performance in permissioned blockchain platform, is done so by altering parameter values like block size, endorsement policy, channels, resources, and state database choices. The study also investigates three basic optimizations: MSP cache, parallel VSCC validation, andbulk read/write during MVCC validation & commit phase.

[15] study presents a kernel-level analysis for the resource consumption and comprehensive performance evaluations of three major consensus algorithms. Study's reveal that resource

Received: August 09 2025

International Journal of Applied Mathematics
Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

consumption differs up to seven times, which demonstrates the importance of proper resource provisioning for BaaS (Blockchain as a Service).

III. HYPERLEDGER FABRIC

Hyperledger Fabric was the Linux Foundation's first con- sortium blockchain platform. The Hyperledger Fabric en- ables arbitrary smart contracts, also known as Chaincodes, written in multipurpose programming languages such as Go, Java, and Nodejs. Other existing blockchain platforms use smart contracts written in a domain-specific language, such as Ethereum's Solidity. Smart contracts support a variety of Hyperledger fabric applications, including IoT, healthcare, and record management [16]. Public blockchain networks, such as Bitcoin or Ethereum, enable anybody to join; however, Hyperledger is a permissioned blockchain network in which players know and can identify each other but not necessarily trust one another. Organisations can profit from DLT, or distributed ledger technology, [17] without the need for a cryptocurrency.

Hyperledger Fabric's pluggable consensus protocol sets it apart from other blockchain systems. Existing blockchain sys- tems use traditional architecture to support transaction order- ing based on the consensus protocol. Following that, each peer performs transactions in an order that is sequential. The execu- tion phase increases network latency, this is because the peers must review and execute all transactions in the block. Corda

[18] introduced the "Notaries" concept for private consortium blockchain groups, where notaries validate transactions and add blocks to the chain. Quorum [19] employs the Quorum-Chain protocol to reach consensus. Peer nodes, clients, and ordering service nodes from various organisations are essential components of the Hyperledger fabric network. Each network entity is assigned an identification by a Membership Service Provider (MSP). Chaincodes might be communicated privately and deployed to a group of peers, making them unavailable to other peers. The data and Chaincode are only visible to members that join the same channel. The Hyperledger fabric network must authenticate and identify peers using generated cryptographic materials. Thus, specific channel participants can be validated in this manner. The Ordering Service orders accepted transactions in blocks based on channel. Figure 1 shows the entire Hyperledger fabric system architecture.

Fabric's blockchain platform has grown in popularity due to its consortium-based structure. However, optimising its performance remains difficult because to its numerous setup options and consensus processes. This study tries to close this gap by thoroughly examining how these elements affect critical performance measures.

Received: August 09 2025

International Journal of Applied Mathematics
Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

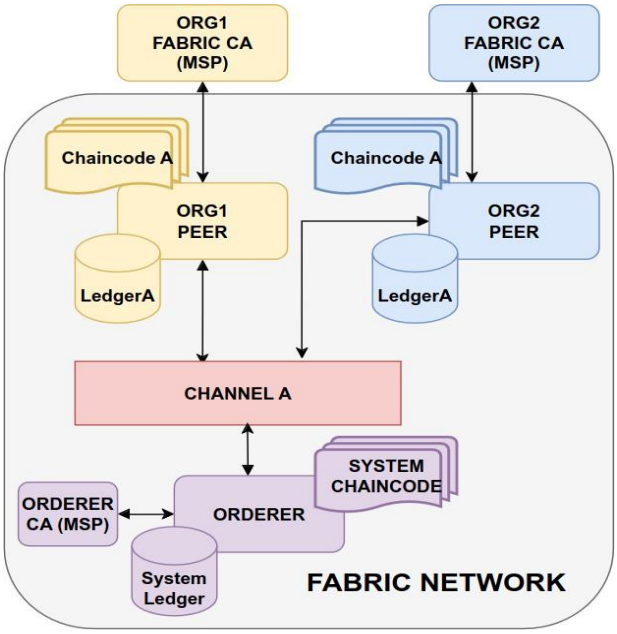


Fig. 1: Hyperledger Fabric Architecture

A. Tranasction Flow

The Hyperledger fabric uses the execute-order-validate-and- commit transaction model. Figure 2 depicts the transaction flow in the HFL blockchain platform. The client sends proposals (1 in Fig. 2) to the peer nodes. The peer nodes convert these into a transaction and return it to the client (2). The client then sends the transaction to the orderer nodes (3), which form a block (4). Finally, the orderer nodes transmit the block to the peer nodes, who register it in the ledger. The block is then committed to the Hyperledger Fabric system

(5). To help understand transaction flow, the operation is divided into four phases: endorsement, ordering, validation, and ledger update.

Endorsement Phase: In the endorsement phase, the client application initiates the transaction by sending a request to the endorser peers. This request includes all of the information and instructions required to carry out the intended operation. After receiving the request, the endorser peer does essential validation tests, such as validating Certificate Authority data, to ensure the transaction's integrity and legitimacy. The endorser peer then executes the specified chaincode, which contains the business logic required to complete the transaction. When the execution is finished, the endorser peer sends endorsement answers that indicate whether the transaction is accepted or refused based on the execution results. These responses are then returned to the client's application for additional review.

Ordering Phase: Following the endorsement step, accepted

transactions move on to the ordering phase. The client sends the endorsed transaction to the orderer peer, who organises and sequences transactions into blocks before including them in the ledger. When the orderer peer receives the transaction, it assigns it a location within a block and generates a new block with the transaction data. This block is afterwards distributed to the anchor nodes of other member organisations on the Fabric network. The

Received: August 09 2025

International Journal of Applied Mathematics
Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

orderer peer guarantees that transactions are ordered sequentially to maintain the ledger's integrity and consistency.

Validation Phase: In the validation stage, the orderer peer sends the newly created block to the anchor nodes, which then distribute it to the correct peers in their organisation. Every peer independently verifies that the transactions in the block conform to the specified endorsement policies and consensus rules. Verifying the digital signatures, looking for instances of double-spending, and making sure the transactions follow the predetermined business logic recorded in the chaincode are all part of this validation process. The peer ensures data consistency and integrity throughout the network by updating its local ledger with the authorised transactions following a successful validation.

Ledger Update Phase: Finally, the client application is notified of the transaction status during the ledger update phase, which includes information on whether the transaction was handled properly or if there were any issues. The client application can take appropriate action based on the trans- action outcome thanks to the input provided by this status message. As this is going on, every node in the Fabric network is now up to date with transaction data from its individual ledgers, which reflects the most recent distributed ledger state. The Hyperledger Fabric blockchain platform's dependability and integrity are guaranteed by this all-inclusive transaction flow, which includes phases for ledger updating, ordering, validation, and endorsement.

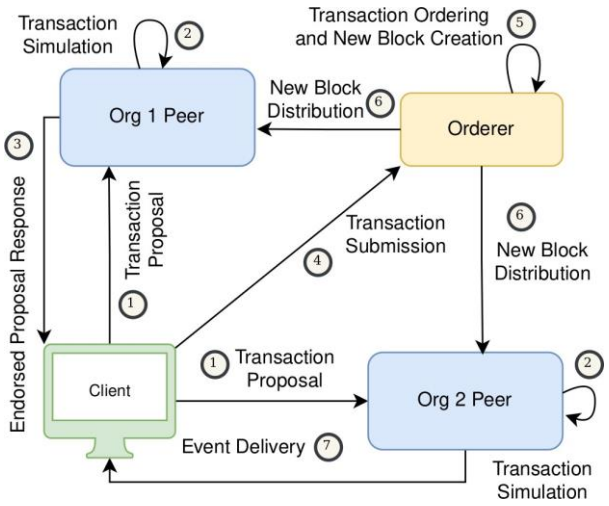


Fig. 2: Hyperledger Fabric Transaction Flow

CONSENSUS ALGORITHM

Consensus mechanisms are protocols or methods used to reach an agreement among distributed members in a blockchain network. They decide how transactions are vali- dated, sorted, and added to the blockchain. Hyperledger Fabric supports a variety of consensus techniques, each with its own set of advantages and disadvantages. Consensus systems allow network participants, known as nodes or validation mechanisms, to agree on one version of the truth without depending on a centralised authority. They enable decen- tralised validation and synchronisation of transactions across the network, eliminating double-

Received: August 09 2025

International Journal of Applied Mathematics
Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

spending and ensuring the blockchain's consistency. Hyperledger Fabric uses a variation of PBFT to create consensus among a selected group of network members known as "ordering service nodes." These nodes handle transaction ordering and validation. PBFT en- sures that a network can come to a consensus on transaction order as long as the majority of ordering service nodes are legitimate and communicate reliably.

A. Solo

Hyperledger Fabric's Solo orderer provides a simple single- node consensus mechanism. Solo, unlike other consensus techniques, does not require the participation of several nodes to achieve consensus. Its primary application is in testing and development scenarios because to ease and simplicity of setup. Solo is a poor choice for production networks since it lacks essential features like fault tolerance and Byzantine fault tolerance. These flaws make it unsuitable for delivering the robustness and dependability necessary in real deployment scenarios.

B. Raft

Raft emerges as an exactly built consensus algorithm for managing a replicated log in a distributed environment. Raft is set up in Hyperledger Fabric using the "raft" orderer. This technique supports the creation of consensus among ordering nodes via a number of complex processes such as leader election, log replication, and commit protocols. Raft fundamentally allows the network to elect a leader node, which is in charge of setting up transactions and maintaining consensus throughout the distributed system.

In the context of Hyperledger Fabric's implementation of the Raft consensus mechanism, several additional steps are involved. Transactions are initially delivered from the client to the orderer, which acts as the leader (also known as the leader orderer). The leader orderer proposes a block containing the received transactions to the follower orderers. Follower orderers collect votes on the proposed block and send them back to the leader orderer. Upon receiving enough votes, the leader orderer creates a new block containing the transactions. The leader orderer then broadcasts the newly created block to the follower orderers to ensure commitment. Follower orderers replicate the received block to maintain consistency across the network. Once the block is committed, the transaction is returned to the client, validating its inclusion in the ledger.

Notably, the use of the Raft consensus process improves the Hyperledger Fabric network's fault tolerance, allowing it to endure potential system crashes with resilience and efficiency.Kafka's distributed architecture and fault-tolerant design, Fab- ric ensures the orderly and consistent processing of trans- actions, thereby upholding the integrity of the ledger and fostering trust among network participants.

Received: August 09 2025

International Journal of Applied Mathematics
Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

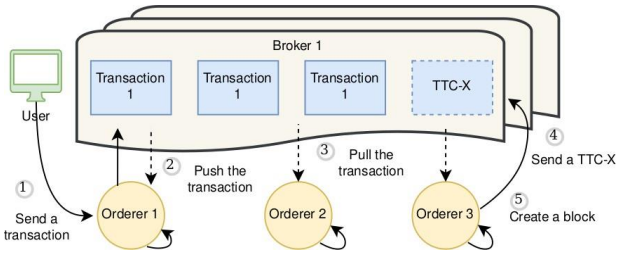


Fig. 4: Implemention of Kafka consensus

C. *Kafka*

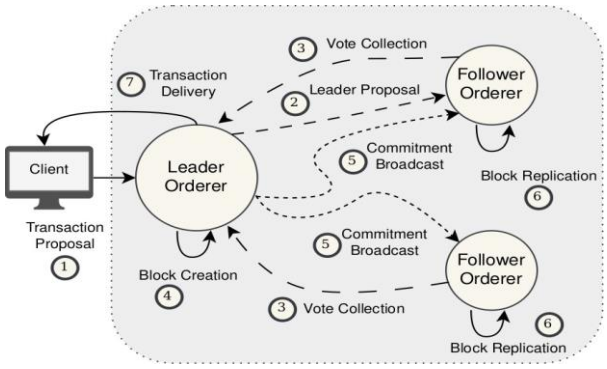


Fig. 3: Implemention of Raft consensusCONFIGURATION PARAMETERS AND KEY METRICS

This study intends to extensively analyse how well Hyper- ledger Fabric performs in a distributed setting by changing the number of nodes and conditions to understand how certain aspects affect its performance. We're mostly interested in how things function from the perspective of the network's

In Hyperledger Fabric's Kafka-based consensus mechanism, Apache Kafka serves as a pivotal component for coordinating transaction ordering among network nodes. When a client initiates a transaction, it is directed to one of the orderers, such as orderer 1. Upon receipt, orderer 1 promptly pushes the transaction to the Broker transaction pool within the Kafka infrastructure. This pool acts as a centralized repository where transactions are temporarily stored for subsequent processing. Once the transaction is available in the Broker pool, another orderer, like orderer 2, pulls it from the pool for further handling. This step ensures redundancy and fault tolerance by distributing the workload across multiple orderers. Mean- while, within the Kafka infrastructure, orderer 3 monitors the accumulation of transactions. When a sufficient number of transactions have been gathered, orderer 3 initiates the creation of a new block by sending a Transaction to Commit-X (TTC-

X) message to the broker.computers, with a focus on overall performance as well as an in-depth analysis of specific issues. We utilised Hyperledger Fabric version 2.2.1 and Hyperledger Calliper [20] to measure system performance.

Received: August 09 2025

International Journal of Applied Mathematics
Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Our experiments used a large number of machines as network peers, between 5 to 50 per group, each of which were set up on a local system with Docker. To ensure consistency, all of these PCs ran the same Ubuntu operating system. We choose Hyperledger Calliper as our tool because it is com- monly used and recommended by specialists. We conducted our experiments using a common chaincode known as Asset transfer, that helps in the management of network assets. This chaincode allows us to create, read, update, remove, and transfer assets, providing a realistic workload for testing. We also included three distinct organisations in our network to make our findings more suitable for real-world circumstances.

TABLE I: System under test parameter and metrics

Upon receiving the TTC-X message, the Kafka brokerorchestrates the aggregation of transactions into a coherent block. This block encapsulates the collected transactions along with necessary metadata, such as timestamps and crypto- graphic signatures. Orderer 3, having initiated the block cre- ation process, oversees its formation to ensure compliance with the network's consensus rules.

The completion of block creation marks a significant mile- stone in the transaction ordering process. The newly formed block represents a batch of transactions that are ready for inclusion in the distributed ledger. Subsequently, the block undergoes dissemination to all network participants to achieve consensus on its validity and integrity.

Through this Kafka-based consensus mechanism, Hyper- ledger Fabric achieves a high degree of reliability and re- silience in its transaction ordering process. By leveraging**Parameter Values**

Number of Peers	5 - 15
Number of Orderer	3 - 15
Batch size	1 - 25 transactions per block Batch timeout 0.1 - 5 seconds
Number of Transaction in caliper	500 - 2000
Transaction Sending rate	10 - 30
Number of Channels	1 Channel
World StateDB	CouchDB

A. Transaction Throughput

The rate at which legitimate transactions are committed by the Hyperledger Fabric network during a certain duration must be carefully monitored. This parameter, known as trans- action throughput, differs between blockchain systems and is essential for evaluating system performance. Transaction

hroughput, defined as the number of transactions executed per unit of time, offers information on the network's efficiency. In our research, the throughput of transaction *i* can be calculated

Received: August 09 2025

International Journal of Applied Mathematics
Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

using the following equation:EVALUATION AND ANALYSIS

Our objective is to closely emulate the setup described in [19], which involves utilizing a base configuration where each organization has 5 peers, each channel has 3 orderers, and each peer in an organization has 1 CouchDB instance. We

TPS_i $\underline{Count(T_x in(t_s, t_e))}$

$t_e - t_s$

(1)generate certificates for all nodes involved in the network using cryptogen. The aim is to thoroughly analyze various

where, T_x is the total number of submitted transactions, t_e is the last block commit time, and t_s is the initial transaction submission time.

B. Transaction Latency

Transaction latency refers to the time it takes for the system to confirm and commit a transaction after it has been submitted. It refers to the time it takes for a transaction to be accessible across the network after it is begun. Transaction latency, which is measured per transaction, is an important performance data for evaluating blockchain systems. In our research, the average delay of transaction i can be calculated using the following equation:transactions in Fabric v2.1 under these specified conditions.

A. Number Of Peer Node Changes

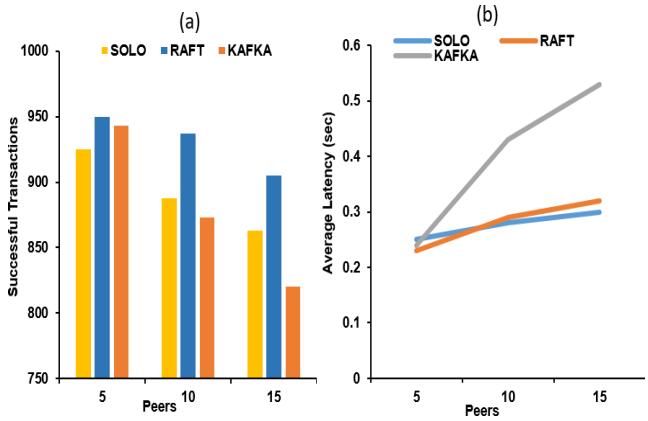
B. $\overset{L}{\underline{T}}$ $(tTxconfirmed - tTxinput)$

C. $AL_i =$ $\underline{x}$

D. $Count(T_x in(t_s, t_e))$

E.

F.



2) Fig. 5: Peers

International Journal of Applied Mathematics
Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

1) Successful Transactions: In Figure 5 (a) illustrates the

The equation represents the duration of a transaction, where t_s marks its initiation and t_e denotes its completion. The transaction transmitted to the peer is indicated by $T_{xconfirmed}$ and T_{xinput}.

C. Block Size

Transactions in Hyperledger Fabric are organised into blocks at the orderer and distributed to peers using a gossip protocol. Each peer processes one block at a time, hence block size is a significant factor. Three variables determine block size: the maximum transaction count, the absolute maximum bytes, and the desired maximum. The size of blocks affects both throughput and latency, resulting in a tradeoff between the two. Our research analyses the effects of different block sizes on network performance, offering insights into an important feature of blockchain performance.

D. Transaction Success and Failure

In instances where the batch size is insufficient, network congestion may arise, increasing the likelihood of transaction failures. This congestion can be particularly pronounced dur- ing periods of peak usage or when transaction volumes spike unexpectedly. Additionally, transaction failures may occur due to batch timeout issues, where transactions exceed the configured timeout period, resulting in failures. Furthermore, insufficient resources on nodes executing transactions, like as CPU or memory, can contribute to transaction failures, particularly when the network is experiencing large transaction volumes or nodes are underprovisioned.successful transaction number in increasing peer nodes envi- ronment. We set the X axis as 5, 10, 15 and set the maximum transaction test up to 1000. Raft outperform the Kafka in successful transaction with a notable difference of more than 100 transaction on 15 peers since increasing the number of peers has a strong negative impact on the system's throughput and it limits scalability produces significant overhead in both the endorsement and ordering phases of the transaction. As the number of endorsing peers grows, each client must wait for a more extensive set of endorsements from a larger group of peers to prepare the endorser transaction.

2) Latency: In Figure 5 (b) shows the latency comparision of the three ordering services in the increasing peers, since solo being a test ordering service normally used during intital stage of network does not add much contribution in test. Raft has a significant drop in average latency compared to Kafka due to its streamlined consensus mechanism where a single leader coordinates transactions, resulting in lower latency as there's less coordination overhead compared to Kafka's multi-leader replication model. With Kafka, each peer acts as a leader, leading to more coordination complexities and ultimately higher latency as the number of peers increases.

G. Number Of Orderer Node Changes

1) Successful Transactions: In Figure 6 (a) illustrates the successful transaction number on increasing orderer environ- ment. We set the X axis as 3, 5, 7, 11, 15 and set the maximum

Received: August 09 2025

International Journal of Applied Mathematics
Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

transaction test up to 1000. Figure (a) indicate that Raft consistently outperforms Kafka in terms of successful trans- action rates, with a notable difference of approximately 30-40 transactions. Orderer 15 hits the lowest successful transaction

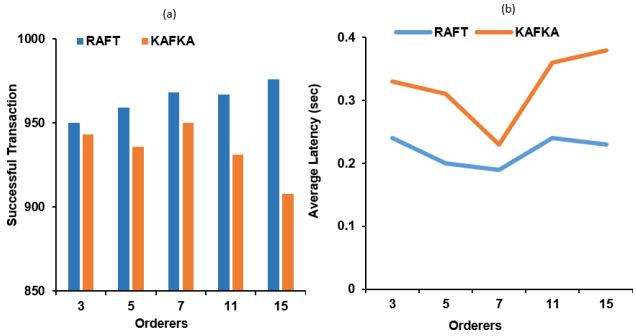


Fig. 6: Orderers

in kafka due to escalating overhead of data replication across a larger network. Kafka leader based replication mechanism may lead to increased dispute and disagreement in consensus issues as the number of nodes grows, resulting in higher failure rates.

2) *Latency:* In Figure 6 (b) shows the latency of the Kafka and Raft consensus algorithms as the number of orderer nodes increases the average latency increases as well. This is because the data replication performed by the leader node of Kafka and Raft is processed with the broadcasting of blocks, regardless of the number of orderer nodes. The latency of the Kafka increases gradually after 11 orderer due to the increasing overhead of data replication across a larger number of nodes. In contrast, Raft exhibits lower latency as the number of nodes increases, showcasing its efficient consensus mechanism.

H. Batch Size Changes

1) *Successful Transactions:* In Figure 7 (a) illustrates the successful transaction on increasing number of batch size, We set the X axis as 3, 5, 10, 15, 25 and Y axis up to 1000 transactions. There is a significant increase in Raft and Kafka consensus algorithm since increasing the batch size results in more number of transaction included in a block leading in less block overload in network leading to more number of successful transactions

2) *Latency:* In Figure 7 (b) illustrates the average latency on increasing number of batch size. The latency of all the or- dering service increases significantly with each batch increase in batch size since more transaction can be accumulated in a block which results in less block generation and low network congestion.

I. Throughput: In Figure 7 (c) illustrates the throughput for three consensus algorithms as the batch size increases. On Y axis we set the transaction per second to 30. Here, the throughput of the ordering services remains uniform around between batch size of 5 and 15. This is because, with these batch sizes, transactions that arrive at orderer nodes fill the blocks of the batch sizes. Thus, the number of generated blocks decreases as the batch size increases. Thus, the required processing and latency decrease, which leads to improved throughput.*Batch*

Received: August 09 2025

International Journal of Applied Mathematics
Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Timeout Changes

1) Successful Transactions: In Figure 8 (a) illustrates the successful transaction on increasing number of batch timeout, We set the X-axis to 0.1, 0.5, 1, 2, 5 and Y axis up to 1000 transaction. The successful transaction rate of Raft and Kafka increases by 30 transaction till batch timeout 2 after that it drop by 40 transaction due to significant timeout between two block of transaction causing more waiting time in network to finish block ordering and creation.

2) Latency: In Figure 8 (b) shows the average latency on increasing number of batch timeout. There is a increase in latency from 0.5, it remains uniform till 2 after that it increases again. The average latency of each of the ordering service increases with time since there is more waiting time in orderer new block creation and generation with trade off between successful transaction 8 (a). Batch timeout and Batch size is directly realted to increase in latency due to network overhead and congestion.

3) Throughput: In Figure 8 (c) shows the throughput on increasing number of batch timeout. There is a increase in throughtput of the consensus with timeout since, High batch timeout does not have much impact on throughtput of the consensus since there is enough waiting time between block in network, vice versa for lower batch timeout. With proper selection of batch timeout client can get a maximum throughput with more successful transactions.

J. Transaction Number Changes

1) Successful Transactions: In Figure 9 (a) displays the successful transaction count on increasing number of transac- tions. The X-axis is configured with values of 500, 1000, 1500, and 2000, while the Y-axis extends up to 2000 transactions. This graph was evaluated with a TPS (Transactions Per Sec- ond) of 30, resulting in minimal failed transactions. However, it is important to also examine other parameters such as average latency (9(b)) and throughput (9(c)) to ensure optimal performance for achieving maximum successful transactions.

2) Latency: In Figure 9(b) depicts the average latency on increasing number of transactions. The latency of Solo consensus remains consistent, as it employs only one orderer and is primarily utilized during the initial stages of develop- ment and network testing. In contrast, both Raft and Kafka exhibit higher latency, reaching approximately 0.35 seconds for 1000 transactions, while maintaining a constant latency of

0.2 seconds for other transaction volumes.

3) Throughput: In Figure 9 (c) illustrates the throughput on increasing number of transactions. The throughput of Raft outperform Kafka by 5% to 6% due to raft leadership structure allows for streamlined coordination of transactions, resulting in less overhead and higher throughput compared to Kafka's multi-leader replication model. Raft superior efficiency and resource management contribute to its higher throughput per- formance in this scenario.

Received: August 09 2025

International Journal of Applied Mathematics
Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

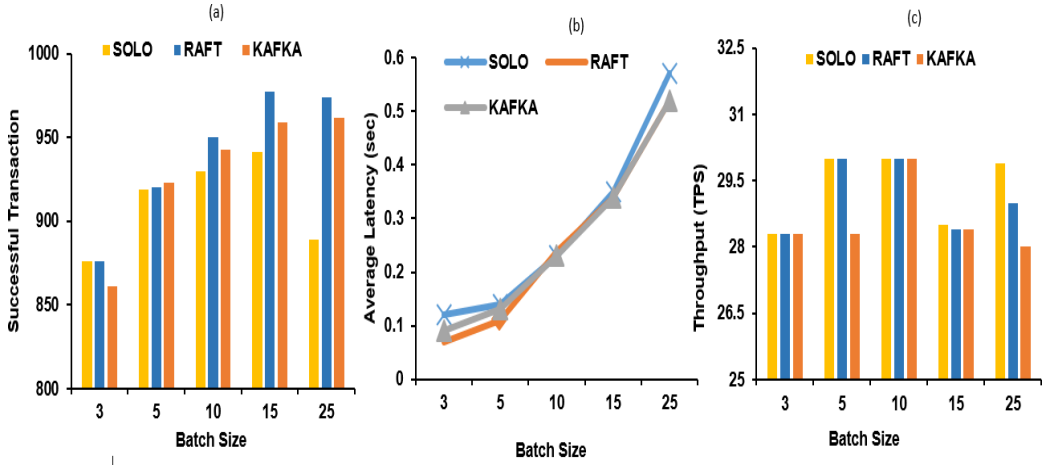


Fig. 7: Batch Size

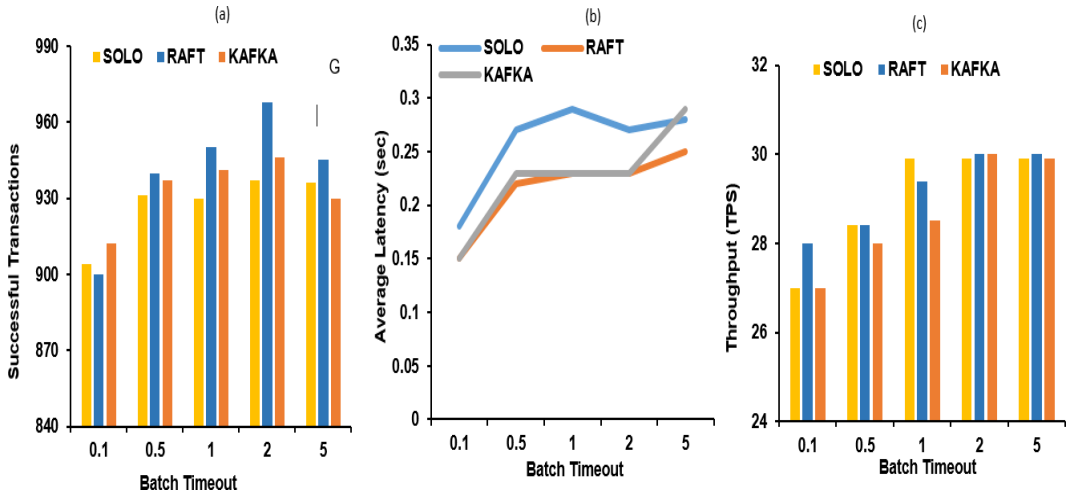


Fig. 8: Batch Timeout

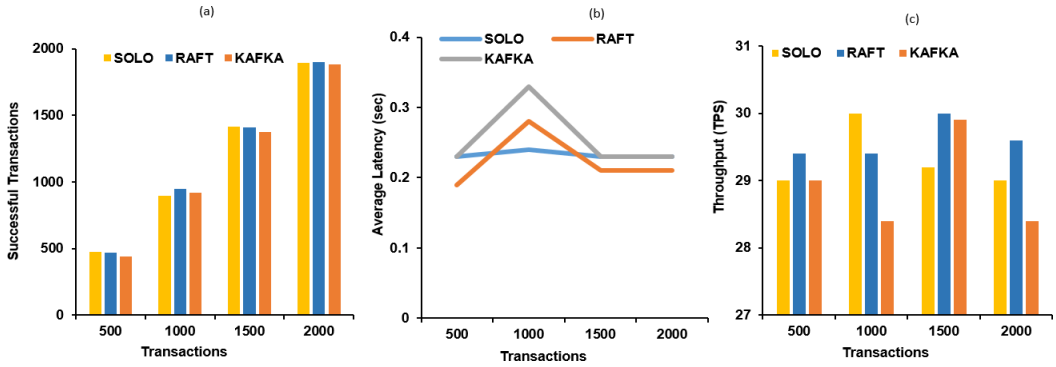


Fig. 9: Transactions

K. Transaction Per Second Changes

1) Successful Transactions: In Figure 10 (a), we observe a depiction of successful

International Journal of Applied Mathematics
Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

transaction counts as the number of transactions per second (TPS) increases. The X-axis is scaled from 10 to 50, representing the varying TPS values, while the Y-axis illustrates successful transactions, reaching up to 1000 transactions. As the TPS rises, network overhead and con- gestion escalate, leading to a higher incidence of transaction failures. Raft demonstrates superior performance compared to Kafka, exhibiting a higher count of successful transactions. Conversely, Solo's performance remains relatively constant due to its single orderer configuration, which limits the impact of increased TPS on transaction success rates.*Latency:* In Figure 10 (b), we illustrate the average latency with the increasing number of transactions per sec- ond (TPS). As TPS increases, latency decreases steadily, reflecting the inverse relationship between transaction volume and latency. With more transactions executed and orderer blocks generated rapidly within shorter durations, latency fall off. This declining latency trend under mines the efficient processing capabilities of the system under higher transaction loads.

2) *Throughput:* In Figure 10 (c), we illustrate the through- put with the increasing number of transactions per second (TPS). As TPS increases, both Kafka and Raft shows same throughput performance. This uniformity in throughput bring out the scalability of both consensus mechanisms in handling

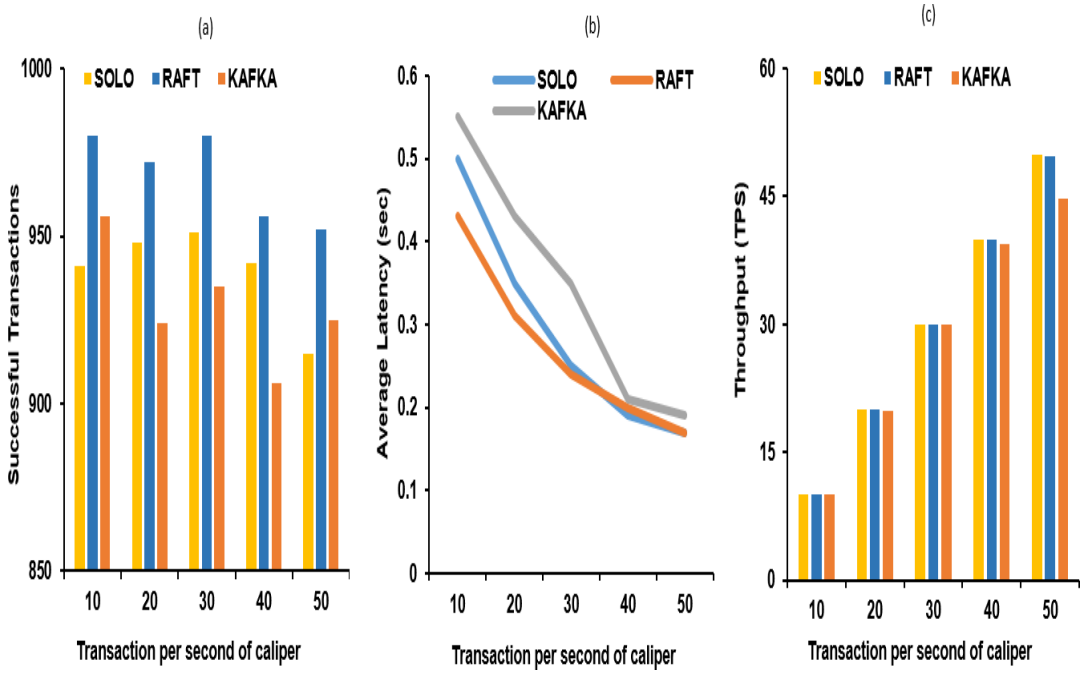


Fig. 10: Transactions per second of caliper

Received: August 09 2025

Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

higher transaction volumes without experiencing decline in performance.

VII. Future Work

In our study, we largely conducted trials in a local set- ting, using tools such as Docker for testing. However, it is crucial to note that this controlled environment may not completely match the complexities and variations found in real-world network and device settings. This constraint may have an impact on how well our findings apply to real-world deployment circumstances. Furthermore, despite our efforts to investigate numerous parameter combinations in Hyperledger Fabric, our research may not fully cover all possible case. Instead, we focused on sample combinations to get insight into ordering services' performance under various scenarios. Furthermore, while our study assumes a controlled network environment, future research should evaluate the potential impact of external factors such as network congestion on ordering service performance.

Looking ahead, we plan to integrate Hyperledger Fabric with external systems such as IoT devices and older enterprise systems. This research intends to provide interoperability standards and efficient data exchange processes, which will increase the platform's variety and usefulness. Furthermore, we are going to prioritise improving Hyperledger Fabric's security and privacy features, including investigating strategies to strengthen data encryption, access control, and identity management within the blockchain network. We will han- dle governance and compliance concerns [21], especially in regulated areas like finance or healthcare, by establishing governance models, auditing processes, and compliance frame- works to assure conformity to regulatory and industry norms. These activities will help to increase trust and confidence in a blockchain-based solutions for a variety of applications.

VIII. Conclusion

This paper investigates the performance of Hyperledger Fab- ric, a permissioned blockchain platform, through an analysis of configurable parameters such as block size, block timeout, transaction numbers, number of peers, number of orderers, anda comparison of different ordering services. Our study reveals that the total number of peers and orderers significantly affect successful transactions and latency in Kafka, whereas the total number of orderers in Raft has only a negligible impact on these metrics. Despite both Kafka and Raft employing leader- based replication, Raft demonstrates superior transaction suc- cess rates even with increased orderer nodes, indicating its resilience in distributed consensus management. Additionally, factors such as batch size and batch timeout play critical roles in fabric network performance, directly influencing through- put, latency, and successful transactions. Thus, in applications requiring both low latency and high transaction success rates, Raft emerges as a compelling consensus algorithm, offering a well-balanced solution for real-time data processing scenarios. This study helps us understand how much resources are used, which can help us choose the right consensus algorithms for Hyperledger Fabric.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized business review*, 2008.

Received: August 09 2025

[2] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, vol. 3, no. 37, pp. 2–1, 2014.

[3] T. Guggenberger, V. Schlatt, J. Schmid, and N. Urbach, "A structured overview of attacks on blockchain sys- tems." *PACIS*, p. 100, 2021.

[4] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin,

K. Christidis, A. De Caro, D. Enyeart, C. Ferris,

L. Laventman, Y. Manevich *et al.*, "Hyperledger fab- ric: a distributed operating system for permissioned blockchains," in *Proceedings of the thirteenth EuroSys conference*, 2018, pp. 1–15.

[5] A. I. Sanka and R. C. Cheung, "A systematic review of blockchain scalability: Issues, solutions, analysis and future research," *Journal of Network and Computer Ap- plications*, vol. 195, p. 103232, 2021.

[6] T. Wang, D. Huang, and S. Zhang, "Consensus algo- rithm analysis in blockchain: Pow and raft," *Wireless Blockchain: Principles, Technologies and Applications*,

pp. 27–72, 2021.

[7] M. Petrescu and R. Petrescu, "Log replication in raft vs kafka," *Studia Universitatis Babes, -Bolyai Informatica*, vol. 65, no. 66, pp. 10–24 193, 2020.

[8] H. Honar Pajooh, M. A. Rashid, F. Alam, and S. Demi- denko, "Experimental performance analysis of a scal- able distributed hyperledger fabric for a large-scale iot testbed," *Sensors*, vol. 22, no. 13, p. 4868, 2022.

[9] K. Gao, Y. Liu, H. Xu, and T. Han, "Design and implementation of food supply chain traceability system based on hyperledger fabric," *International Journal of Computational Science and Engineering*, vol. 23, no. 2,

pp. 185–193, 2020.

[10] A. Keith, T. Sangarapillai, A. Almehmadi, and K. El- Khatib, "A blockchain-powered traffic management sys- tem for unmanned aerial vehicles," *Applied Sciences*, vol. 13, no. 19, p. 10950, 2023.

[11] G. Al-Sumaidaee, R. Alkhudary, Z. Zilic, and A. Swidan, "Performance analysis of a private blockchain network built on hyperledger fabric for healthcare," *Information Processing & Management*, vol. 60, no. 2, p. 103160,

2023.

[12] P. Sharma, R. Jindal, and M. D. Borah, "Blockchain- based distributed application for multimedia system using hyperledger fabric," *Multimedia Tools and Applications*, vol. 83, no. 1, pp. 2473–2499, 2024.

[13] C. Harris, "Performance evaluation of ordering services and endorsement policies in hyperledger fabric," in *2023 33rd Conference of Open Innovations Association (FRUCT)*.

Received: August 09 2025

Volume 38 No. 2s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

IEEE, 2023, pp. 63–69.

[14] P. Thakkar, S. Nathan, and B. Viswanathan, "Perfor- mance 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, 2018, pp. 264–276.

[15] G. Yang, K. Lee, K. Lee, Y. Yoo, H. Lee, and C. Yoo, "Resource analysis of blockchain consensus algorithms in hyperledger fabric," *IEEE Access*, vol. 10, pp. 74 902– 74 920, 2022.

[16] N. Gupta, A. Gupta, A. Ahirekar, O. Garg, and M. Patil, "Orphanage management using ipfs and hyperledger fabric," *International Journal of Scientific Research in Engineering and Management*, vol. 07, p. 6, 04 2023.

[17] J. Li and M. Kassem, "Applications of distributed ledger technology (dlt) and blockchain-enabled smart contracts in construction," *Automation in construction*, vol. 132, p. 103955, 2021.

[18] R. G. Brown, J. Carlyle, I. Grigg, and M. Hearn, "Corda: an introduction," *R3 CEV, August*, vol. 1, no. 15, p. 14, 2016.

[19] A. Baliga, I. Subhod, P. Kamat, and S. Chatterjee, "Per- formance evaluation of the quorum blockchain platform," *arXiv preprint arXiv:1809.03421*, 2018.

[20] Hyperledger. (Accessed 2024) Hyper- ledger Caliper. [Online]. Available: https://www.hyperledger.org/projects/caliperJ. Z. Llamas Covarrubias and I. N. Llamas Covarrubias, "Different types of government and governance in the blockchain," *Journal of Governance and Regulation*, vol. 10, no. 1, 2021.

Received: August 09 2025