Evaluation of Blockchain Service Level Agreement (SLA) Using Hyperledger Fabric (HLF)

Dhivya K 1, B Akoramurthy2, B Surendiran3, T Sivakumar4

1 M.Tech Scholar, Department of Computer Science, Pondicherry University, Pondicherry, India
2 PhD Scholar, Department of Computer Science and Engineering, National Institute of Technology Puducherry, Karaikal, India
3 Associate Professor, Department of Computer Science and Engineering, National Institute of Technology - Pondicherry, Karaikal, India
4 Assistant Professor, Department of Computer Science, Pondicherry University, Pondicherry, India

Emails: dhivyaakoramurthy@gmail.com, akor.theanchor@gmail.com, surendiran@gmail.com, tsivakumar72@gmail.com

Abstract
Different sectors are being revolutionized by distributed ledger technology. According to the 2022 market valuation, Hyperledger is now the second-largest blockchain platform for smart contracts. The creation of numerous apps may be sped up and simplified with smart contracts, but there are certain drawbacks as well. For instance, vulnerability contracts are created intentionally to weaken candor, smart contracts are employed to conduct fraudulent activities, and there are many redundant contracts that squander the efficiency of the system for no real reason. To solve these problems, we provide in this research Service Level Agreement (SLA) for Hyperledger smart contracts. We created Hyperledger smart contracts and focused on how smart contracts and consumers used data. By manually analyzing the transactions, we were able to extract four behavioral characteristics that may be used to differentiate between various contract types. Then, a smart contract is built using these to include 14 fundamental functionalities. We provide a data splitting algorithm for splitting the gathered smart contracts in order to create the experimental dataset. Then, we train and test our dataset using an LSTM network. The comprehensive experimental findings demonstrate that our method can discriminate between various contract types and may be used to identify malicious contracts and detect anomalies with acceptable precision, recall, and F1-score.

1. Introduction
The transactions in digital currencies are recorded using the decentralized, distributed technology known as blockchain (X. Li et al.). Decentralization, permanence, anonymity, and auditability are just a few of the advantages this gives the blockchain (Zheng et al.). As a result, blockchain technology has advanced quickly in recent years, and its use cases have expanded beyond the original digital currency distribution transaction to include banking, medical, the Internet of Things, software engineering, and other industries (X. Li et al.). The first successful implementation of blockchain technology was Bitcoin (Nakamoto). In 2015 (Desjardins) and 2016 (X. Li et al.), Bitcoin was hailed as the masterpiece in digital market and commodities, respectively. One of the systems for permissioned blockchains is Hyperledger Fab-
Evaluation of Blockchain Service Level Agreement (SLA) Using Hyperledger Fabric (HLF) 2023, Vol. 05, Issue 05S May

Androulaki, Barger, and Bortnikov). The Linux Foundation created it, and it is open-sourced. Every component serves a unique purpose according to its job. The four primary steps of the transaction flow are endorsement, ordering, validation, and committing. Prior to a network’s start, each component has to be tailored and defined according to the network’s specifications. To launch their bespoke fabric network, designers need to deal with a multitude of factors surrounding all components. The fabric is made up of a number of different parts, including peer nodes, clients, membership, an ordering service, and Chaincode. Most blockchain networks are built for the purpose of transferring and storing money, and in most cases, this involves producing a ”currency” that is used on the network. The word ”assets” is used generally in relation to Hyperledger Fabric (D et al.). Cash, real estate, a car, or even an insurance policy are all examples of assets. Its main difference from most other blockchain networks is that it will be used by workers of one or more organizations rather than by individuals. It provides a variety of possibilities rather than a single standard of work for the blockchain network. It may be simpler to construct a network in this fashion so that it may be implemented within the business model of the firm in question.

2. Background

By employing a model, the model and the Calipers benchmark tool, overall performance and delay of HLF version 1 were investigated in (Baliga et al.). The study investigated the effects of several Chaincodes variables and various transactions on transactions performance and latency under micro workloads. By incorporating different numbers of Chaincodes, routes, and neighbors, the authors evaluated the efficacy of Hyperledger Fabric characteristics. The evaluation findings demonstrated how sensitive HLF v1.0 performance is to the Ordered option. Also, the outcomes demonstrated that the HLF v1.0 Compiler was unable to manage a transaction in concurrently using numerous virtual CPUs (vCPUs). That can be viewed as a speed constraint for the system.

Through incorporating additional applications to the platform and nodes scales, Nasir et al. in (Nasir et al.) conducted an experimental efficiency study of two distinct flavors of HLF (v0.6 and v1.0) to examine the completion time, performance, delay, and adaptability. The findings demonstrated that across all important performance indicators, HLF v1.0 usually beat HLF v0.6.

In (Kuzlu et al.), the authors assessed the HLF v1.4 platform’s performance in terms of ’m getting, delay, and scaling with a range of network workloads, including diverse transaction volumes, types, and rates.

The research explained in (Wang) with developed different malicious behaviour patterns examined the effect of malicious conduct on the transactional latency and throughput aspects of HLF performance. The results revealed a considerable decrease in systems efficiency because of assault delay as well as the failure of certain duplicates.

A revolutionary architecture was put forth in Paper (W. Li et al.) to improve the durability of public blockchains. The new design solves the scalability problem of individual Byzantine-fault-tolerant-based (BFT) systems like HLF v0.6 and offers satellites chains to establish a collection of networks. Secure resource transfer across network is possible with suggested design.

A information middle and high the Ethereum Blockchain was created in (Takahashi, Kanai, and Nakazato). A range of process and function, including photos and 3d models, were used to test the study’s validity in order to show how well the blockchain technology performs overall when used for exchanging sensor information.

2.1. Hyperledger Fabric

Hyperledger fabric is a blockchain platform with permissions wherein users may see and trust one another. However, it might be set up according to the model of governance developed in order to foster confidence among the users. The orchestration and deployment of distributed ledgers inside coalitions depends on the participation of the collaborating entities. The blockchain is hosted by nodes (or peers), who also execute smart contracts and cooperate to keep apprised of the current state of the record. The common Chaincode can be implemented by all entities or developed privately thanks to the HLF’s channel idea. Chain codes may be sent to a set of nodes in a private manner, rendering them inaccessible to other nodes. Only those who subscribe to this very same network have access to the information.
2. Transaction Flow in Hyperledger Fabric

The transactional method used by HLF is execute-order-validate-and-commit (EOVC). The transactional process of the HLF private ledger is depicted in Figure 2. The stages involved are verification, sorting, and approval. Chaincode spellcasters are Docker (Shahbazi and Byun) container-based transactions. Therefore, separating them from other active chaincodes on the same host as well as the HLF codes would indeed be beneficial. The network nodes maintain a record of transactions as part of the private ledger technology known as HLF. The structure of the data and the status of the data are the two components of the blockchain. The state data describes the nodes’ actual situation at any given time, whereas the transaction log records all activities that have been invoked. By running the chaincode, several actions might be performed on the current data status. Transactions are created during processing and added to the log file. The state’s statistics may also alter as a result. The LevelDB, a compact library for creating a key-value data store integrated into the HLF node implementation, is used to generate the ledger of transactions. The combinations of session keys make up the status metadata. At the node level, the status database is hot-swappable. A straightforward query for key-value pairs is supported by LevelDB. The CouchDB and NoSQL databases that enable the execution of complicated requests may nevertheless substitute for it.

Prior to actually initiating the HLF transaction processing, it is necessary to reveal the credentials of involved stakeholders, associated MSPs, and neighbors. Upon that Orderer system’s activation, the network must be started with the appropriate organization’s MSPs and nodes joining the channel and initialising the blockchain. Furthermore, the link must have the necessary chaincodes implemented.

2.2.1. Phase 1 — Endorsement Phase:

According to the endorsement rule encoded in the Chaincode, client apps employing the HLF-SDK (Hyper Ledger Fabric Software Development Kit) generate a demand suggestion as well as propose the transaction to endorsing peers. The request is signed cryptographically by the client and sent over similar channels as the blockchain network. The suggested transactions are carried out by the endorsing peers, who also get the predictable output. An endorsement rule must be specified while implementing the Chaincode to determine whose peers (organisations) have had the authority to approve a transaction in the shared smart ledger so that it can be accepted as legitimate and added to the blockchain. To make sure that agreement is reached among all the peers in the process, the HLF takes a number of actions, including establishing endorsement guidelines for ordering services. The sequence of transactions is execute-order-validate-commit (EOVC). Client interactions were first carried out in sandbox environments to ascertain their read-write sets, or the collection of keys that the payment transactions will read from and write to. The transactions are then ordered by an ordering service before being verified and committed to the blockchain. This procedure is carried out by nodes with designated duties. The read-write sets, reply values, and cryptographic data that were generated as a consequence of blockchain code processing are all included in the endorsement reply. The endorsing peer delivers a proposed answer to the client after signing the transactional response with their identity using the Chaincode ESCC (Endorsement System Chain Code) system. At this moment, nothing has altered with the ledger. Client application programmes gather recommendations from a variety of endorsing peers and provide them to an ordering phase in accordance with the endorsement rule established by Chaincodes.

2.2.2. Phase 2 - Ordering Phase:

The ordering service audits each transaction in chaincode. They are arranged by network. The transactions that were ordered must then be delivered by the ordering service. A block, which is a transaction transmitted to every peer within the same order by the ordering service, is required by every peer. Any of the established ordering meth-
ods, including Kafka, Solo, or Raft, is used by the ordering service (Shahbazi and Byun). One ordering service node makes up the Solo ordering algorithm. It is employed for system design that runs on a node. The Kafka algorithm is more fully implemented. The Kafka group is created in order to produce and ingest transactions. As a result, it offers crash fault tolerance (CFT) and therefore is advised for a continuous growing list of records in a production setting. The Raft is comparable to the Kafka due to its own commander structure. Its CFT status is a benefit. The raft’s configuration is somewhat simple. In order to see if client nodes can transmit or accept blocks on a specific channel, the ordering service must run access control permission checks on the client nodes. SOLO use is permitted by Hyperledger Fabric version 1.4. It is a message sent from the middle of the gateway, which is simply configured with one node. In a configuration that is better suited for a development stage, no cluster functionality is provided. Nevertheless, the network now has a single failure point. Development teams don’t think about using it in operations.

Kafka would be used in the finished versions of HLF. High bandwidth and scalability are features of this communication program. It is possible to accept fault tolerance when using a cluster. Multiple communication channels could be tested with different setups because Kafka and SOLO both allow them. A user receives the Orderer to advertise the endorsed transactions to the different peers. The customer receives the responses, which are the endorsements from the endorsing peers. The client is prepared to distribute it to multiple network peers. This step is accomplished by invoking the orderer for the broadcast services. The anchoring peers within that organisation distribute the blocks to the different additional peers after obtaining the blocks containing the transaction. As a result, the person who ordered provides the communication layer of the HLF channel. It is in charge of maintaining the timeline and plays an important role in the consensus protocol.

2.2.3. Phase 3 - Validation Phase:
The transactions must be verified after the blocks has been distributed to all network nodes using the ordering phase provider or chatter protocol. Incorrect transactions are found and rejected through-
out the validation phase. Only legitimate transactions thus are committed as well as maintained in the fabric ledger as well as the world state. The validation services comprise of two sequential steps: read and write discrepancy checking, often known as Multi-version Concurrency Access Controls (MVCAC), and analysis of endorsement phase using the Validation System Chain Code (VSCC).

2.3. Key Indicators and Tuning Factors

The goal of this work is to examine the efficiency of the modular HLF in a distributed architecture with varied levels of nodes and circumstances in order to determine whether certain factors impact system performance. The research is confined to a deep examination of a few characteristics, while other elements are discussed in broad strokes in order to comprehend and characterise the interrelationships of nodes. As a result, the central emphasis is on examining holistic performance from the vantage point of peers. Concurrently, the Orderer and Gossip effects on the investigation were removed because they were fixed. Figure 3 depicts the overarching test system and its subsystems. A solitary HLF system of one user executing assessment utilities and one anchorage is included in the design.

2.3.1. Performances Parameters:

A report with accurate performance measurements that are relevant to multiple DLT platforms was created by the Hyper-ledger fabric Performance as well as Scalability Workgroup. It was applied inside the experiments and analyses that were described.

2.3.2. Transaction Efficiency:

Smart contracts are deployed, executed, and invoked at varying rates in various public blockchain system. Hence, it is necessary to keep an eye on transaction efficiency. It is calculated as the rate at which the HLF network commits legitimate transactions over a predetermined time frame. The assessment at a single peer is taken into account for the Hyper Ledger Fabric network with one channel. Furthermore, the trials were expanded to include numerous peers in the published framework and analyses (up to 500).

2.3.3. Latency Transaction:

It takes a while for the computer to verify the transactions after they have been transmitted to a connection. The length of time between the period a transaction which performed and the period is confirmed, committed, and the outcome is made accessible throughout the network is known as the transaction latency. Each transaction counts toward this metric. But in the majority of situations, the experiment offers different statistics on total transactions, including low, high, medium and standard deviations. The transactions verification at a one peer and several peers having various levels of load were examined in the study that was published. Three factors make up the estimated end-to-end transmission delay: the commit, ordering, validation and endorsement phase (Jamil et al.).

2.3.4. Scalability:

The capacity of both the deployed HLF network to assist growing the member base is calculated in this study. The scale of the network corresponds to the number of verifying peers taking part in SUT consensus. The limited network is displayed to reflect the overall number of nodes that are currently using the HLF public blockchain network.

2.3.5. Size of the Block:

The three parameters that make up the size of the block—the maximum transaction count per second is calculated, the absolute maximum byte values, and the recommended maximum bytes—present the number of transactions per block seconds. A block of transactions is batch processed. The assessment is further expanded in the study to incorporate batches of numerous blocks value (1 blocks, 10 blocks, 50 blocks and 100 blocks). Additionally, it investigates how various batch sizes affect HLF systems.

3. Environmental Setup

HLF’s efficiency and scalability were assessed by implementing several sets of parameters such as transaction sending time, size of the block, size of the network, and NetFlow delivery. The study was performed to evaluate has taken into account a number of measures, including network speed, average sale latency, and available resources. By expanding the connection, scalability was evaluated based on changes in bandwidth and transaction delay. The test findings highlight efficiency bottlenecks, explain the effect of a particular parameter on the HLF public blockchain network, and demonstrate how changes may be made to improve efficiency.

The detailed experimental design used in all of the
experiments is illustrated in Figure 4. In each case, a public blockchain HLF connection was established, including one organisation and numerous peers. A through-fo was used to construct the ordering system, which was executed on a different node. To make the prescribed responsibilities easier, a line of code was put into use on the network.

A permissionless HLF blockchain network was set up in a managed distributed manner. Many Amazon Web services (AWS) EC2 instances were set up as such an underground node-based network in order to get accurate outcomes. Table 1 summarizes the parameters of SUT. Running every instance on its own Virtual Machine (VM). To lessen the impact of connection delays throughout the studies, all VMs were connected to a single subnet. The identical experiment was carried out multiple times with various peer and node values. KV stands for a Key Value that will be communicated to the public blockchain network as a result of a transactions.

IoT gateways in AWS were designed after EC2 instances. The IoT system has adopted different messaging exchanges as chain transactions. This test benchmark system was performed on an AWS EC2 instances with 2vCPUs, 3.0 GHz Xeon Platinum processors, and 4GB RAM. The AWS EC2 machine was running Hyperledger Fabric version v1.4 together with peers, CA, OS, and Ubuntu 18.04 LTS. The effectiveness of the equipment choice (i.e., CPU and RAM) on the bandwidth, delay, and scalability of the developed public blockchain network was examined using that testing environment. VM runs Docker on multiple computer systems to issue transactions. HLF version 1.4 is a permissioned blockchain network application. To perform multiple chains of code, the VM hosts the HLC (Hyperledger Caliper). The various network peers (from 1 peer to 500 peers) in a scalable environment deploys Docker to connect Docker swarms to manage the speed of the container. Multiple nodes uses Ubuntu 18.04 LTS OS.

The strength of a Proof of Work (PoW) agreement is evident. because of it is thought to be safest solution for cryptocurrency application because of its pseudo anonymous character. The players in the chain already are familiar with one another in business environments like Distributed systems and telecommunications environments, thus it seemed superfluous in those settings. Consequently, private blockchain blockchains were made for use in businesses. Platforms that employ consensus mechanism that are simpler & utilise fewer resources, such Raft (Ongaro and Ousterhout), which was used in this work.

To reduce this effectiveness, straightforward transactions must be used. Every transaction produces a number that is added to a value according to the system time specified in the blockchain. These variables each produce a key-value pair that also contains a constant rate. Writing the ledger with transactions requires the blockchain. Peers are allowed to make transactions, and they execute in a secure, isolated Docker container. Every action is indeed a write operation that alters the world. The endorsers then independently execute the chaincode, create a transactional report based on the execution
FIGURE 4. Experimental setup for Performance Evolution

TABLE 1. System under test parameters and values

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of Blocks</td>
<td>30 tps</td>
</tr>
<tr>
<td>Benchmark out time</td>
<td>1000 ms</td>
</tr>
<tr>
<td>Rate of Tx Sending</td>
<td>1-500tps</td>
</tr>
<tr>
<td>Number of Blockchain</td>
<td>1,10,50,100 tps</td>
</tr>
<tr>
<td>Channels</td>
<td>1 channel</td>
</tr>
<tr>
<td>DB State level</td>
<td>Level DB</td>
</tr>
<tr>
<td>Transactions</td>
<td>1 KV of 20bytes size</td>
</tr>
<tr>
<td>Peers</td>
<td>100v CPU, 3.3 GHZ size, 10 GiB, Moderate network size</td>
</tr>
</tbody>
</table>

outcomes, and verify the reply. The verified transaction request answer is the last thing that the application receives.

4. Summary

To assess the performance and scaling of the Hyperledger Fabric framework in a virtualized environment, many instances with diverse hosts within the network (ranging from 1 to 500 hosts) have been tested. The studies were done in a single-host context with an identical setup. In Hyperledger Fabric version 1, the HLF implemented the concept of different firms. This version 1 was implemented on a multi-host system of virtualized implementations, and the system was measured. Number of operations per unit time, delay, node density, resource utilization, and acceptance policies are examples of key metrics. The outcomes were compared to a solitary host implementation system. It performed better on the majority of the measures. Unfortunately, that approach was not relevant in practice. Simultaneously, the efficiency of multiple-host designs may be improved by deploying numerous organisation concepts, each with its own systemiser.

The experiments have numerous parameters, such as the peers, size of blocks, and frequency at which transactions are sent, are listed in Table 2. Each test begins with transactions being sent at speeds ranging from 1 to 500 tps.

4.1. Analysis of Single host and Multiple host Transaction

In a single-host configuration, Figure 5 shows the average efficiency of various blocks over varied transaction transmission rates. The average delay for the same transaction sending rates is shown in Figure 6.

In a Multiple-host configuration, Figure 7 shows the average efficiency of various blocks over varied transaction transmission rates and the average efficiency for the same transaction sending rates is shown in Figure 8.

The resulting performance and average delay metrics are shown in Figures 9 and 10. The findings indicate that both situations exhibit a nearly identical delay pattern up to the highest peak. The trans-
TABLE 2. Parameter used for single and multiple transactions of network nodes

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peers</td>
<td>1, 5, 10, 20, 30, 40, . . . , 90, 100</td>
</tr>
<tr>
<td>Size of the Block</td>
<td>1, 10, 50</td>
</tr>
<tr>
<td>Rate of Transactions send</td>
<td>1, 5, 10, 20, 30, 40,. . . , 90, 100, 200,. . . ,400, 500</td>
</tr>
</tbody>
</table>

actions delay with a single server host endorsement indicates superior efficiency after the peak point. Despite having varied major position for different configurations, the bandwidth effect increases in each scenario.

Figures 11 and 12 demonstrate that improvement is possible with an improvement in block size. The block refers to the number of transactions that can be contained in a block before it is released to the public blockchain network. The Arrange a meeting Batch Size can have a big impact on how fast the system runs. The outcomes demonstrate that what a
small feature size blocks decreases performance.

**FIGURE 11.** Effect of block sizes on system with efficiency.

**FIGURE 12.** Effects of block sizes on transactions with latency

The efficiency and average delays for transactions with various send rates up to 2500 tps are shown in Figures 13 and 14.

**FIGURE 13.** Effects of size of the network on the system with efficiency.

The average peer CPU use is shown in Figure 15. Figure 16 illustrates the average disc write utilization and shows the linear increases with small batches as well as the amount of peers. Similar trends of aggregate memory usage by networking neighbours are shown in Figure 17.

**FIGURE 14.** Effects of size of the network on transaction with latency

**FIGURE 15.** Average CPU usage of Peers

**FIGURE 16.** Average writes disk usage of peers

**FIGURE 17.** Average memory consumption usage of peers

Figures 18 & 19 show the Network input and output Traffic of the peers respectively.
4.2. Evaluation Metrics

To assess the effectiveness of their systems, we employ the Precision (P), Recall (R), and F1 score (F1) metrics. The term “true positive” (TP) describes how many smart contract forecasts were accurate. False positive (FP) refers to the quantity of incorrectly classifying this kind as an alternative type. Also known as a false negative, this type has been mistakenly classified as a variety of different types. The better it is to discriminate between the various forms of smart contracts, the greater the value of precision, recall, and F1 score.

We have two ways of gathering contract information. One is to synchronize all previously processed transaction information using the client. The alternative is to extract a smart contract’s transactions and store them in JSON format using the APIs offered (Ongaro and Ousterhout). We also consult the DApp publication website (Christidis and Devetsikiotis) in accordance with the various uses of smart contracts before classifying the three most prevalent types.

Table 3 shows the number of smart contracts in each different type.

<table>
<thead>
<tr>
<th>Types</th>
<th>Stock Market</th>
<th>Media</th>
<th>Sports</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>782</td>
<td>338</td>
<td>2930</td>
</tr>
</tbody>
</table>

Table 4 shows the result of stock market applications with different smart contracts.

<table>
<thead>
<tr>
<th>Type</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Media</td>
<td>0.932(+-0.006)</td>
<td>0.924(+-0.009)</td>
<td>0.939(+-0.006)</td>
</tr>
<tr>
<td>Sports</td>
<td>0.932(+-0.019)</td>
<td>0.826(+-0.025)</td>
<td>0.876(+-0.019)</td>
</tr>
</tbody>
</table>

Table 5 shows the result of media applications with different smart contracts.

<table>
<thead>
<tr>
<th>Type</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Media</td>
<td>0.932(+-0.006)</td>
<td>0.924(+-0.009)</td>
<td>0.939(+-0.006)</td>
</tr>
<tr>
<td>Sports</td>
<td>0.932(+-0.019)</td>
<td>0.826(+-0.025)</td>
<td>0.876(+-0.019)</td>
</tr>
</tbody>
</table>

Table 7 shows the difference between each one type of contracts which has some differences which varies +/- values is shown for stock market, sports and F1-score.

Focus on F1 score is between 0.701 to 0.835 and overall results are shown using LSTM network.
### TABLE 6. Result of Sports applications with different smart contracts

<table>
<thead>
<tr>
<th>Type</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stock</td>
<td>0.912(+/0.013)</td>
<td>0.919(+/0.015)</td>
<td>0.916(+/0.013)</td>
</tr>
<tr>
<td>Market</td>
<td>0.013</td>
<td>0.015</td>
<td>0.013</td>
</tr>
<tr>
<td>Media</td>
<td>0.885(+/0.025)</td>
<td>0.837(+/0.020)</td>
<td>0.860(+/0.015)</td>
</tr>
</tbody>
</table>

### TABLE 7. Results of different types of each single smart contracts.

<table>
<thead>
<tr>
<th>Type</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stock</td>
<td>0.940(+/0.013)</td>
<td>0.858(+/0.016)</td>
<td>0.897(+/0.014)</td>
</tr>
<tr>
<td>Market</td>
<td>0.013</td>
<td>0.015</td>
<td>0.014</td>
</tr>
<tr>
<td>Sports</td>
<td>0.837(+/0.013)</td>
<td>0.848(+/0.015)</td>
<td>0.876(+/0.014)</td>
</tr>
<tr>
<td>Media</td>
<td>0.888(+/0.020)</td>
<td>0.706(+/0.026)</td>
<td>0.786(+/0.020)</td>
</tr>
</tbody>
</table>

### TABLE 8. Results of different types of smart contracts with LSTM Network

<table>
<thead>
<tr>
<th>Type</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stock</td>
<td>0.805(+/0.052)</td>
<td>0.783(+/0.036)</td>
<td>0.793(+/0.039)</td>
</tr>
<tr>
<td>Market</td>
<td>0.036</td>
<td>0.036</td>
<td>0.036</td>
</tr>
<tr>
<td>Sports</td>
<td>0.883(+/0.043)</td>
<td>0.698(+/0.056)</td>
<td>0.756(+/0.039)</td>
</tr>
<tr>
<td>Media</td>
<td>0.893(+/0.047)</td>
<td>0.778(+/0.065)</td>
<td>0.826(+/0.037)</td>
</tr>
</tbody>
</table>

5. Conclusion

This article offered a comprehensive empirical analysis of the extensible Hyperledger Fabric blockchain technology in a decentralized big network with changing peer and payload counts. A scalable framework for accurate and real-time monitoring of HLF systems was presented. The test data demonstrated that the proposed approach was feasible. On the other hand, it revealed that the framework’s throughput, delay, and resilience are dependent on system setup, blockchain network architecture, and smart contract complexity procedures. According to the findings, as the volume of transactions and array timeout increases, so does the delay. It is also observed that the number of blocks generated and the volume of transactions for each block had a greater effect on performance, or the number of operations performed per unit time. Due to the number of transactions in a single block, all those transactions are to be verified at the same instant, resulting in increased performance as block size increases. Furthermore, increasing the array timeout adds delay because each block must wait even though it has completed all transactions.

To optimize effectiveness, experiments with huge network sizes should take into account the optimal values of endorsements each ChainCode to a smaller peer group. For IoT applications with numerous concurrent processes, it might be suggested that higher batch-timeout and block sizes are essential for keeping good throughput. In order to attain low latency, batch timeout as well as block size must be minimal for IoT systems with lots of transactions.

Future work will take into account updating the HLF, assessing the use of legitimate data information, and investigating more test cases, such as examining the effects of maintaining many Orderers just on system’s efficiency as a whole. We’ll look more closely at other configuration options including having numerous Hyperledger Fabric organisations and having more endorsement peers.

6. Authors’ Note

The authors declare that there is no conflict of interest regarding the publication of this article. The authors confirmed that the paper was free of plagiarism.

References


© Dhivya K et al. 2023 Open Access. This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

Embargo period: The article has no embargo period.