Performance Improvement of Hyperledger Iroha under Network Latency Conditions

Akiho Sakamoto
Ochanomizu University
Tokyo, Japan
akiho@ogl.is.ocha.ac.jp

Masato Oguchi
Ochanomizu University
Tokyo, Japan
oguchi@is.ocha.ac.jp

***Abstract*—This study investigates the performance bottleneck in transaction processing within Hyperledger Iroha when used as an inter-enterprise data-sharing platform. Hyperledger Iroha is an open-source blockchain-based system widely adopted for tamper-resistant data sharing across multiple sites. We propose a method that decouples proposal creation from the constraints of proposal timeouts and maximum proposal size, instead triggering proposal generation immediately upon transaction arrival at the ordering service. To assess its effectiveness, we conducted performance evaluations under network-latency conditions to approximate real-world deployment environments. Experimental results show that the proposed method significantly improves transaction throughput and provides greater resilience to network delays compared to the conventional approach. Notably, under a 5ms latency, the average throughput degradation rate was reduced from 11.3% to 8.6%. These results demonstrate that the proposed method enhances both the performance and latency tolerance of Hyperledger Iroha in practical applications.**

***Index Terms*—blockchain, hyperledger iroha, performance analysis**

I. Introduction

With the advancement of artificial intelligence (AI) and Internet of Things (IoT) technologies, the utilization of big data for innovation and the resolution of societal challenges has garnered increasing attention. IoT technologies enable the collection of heterogeneous data across a wide range of contexts, leading to the accumulation of massive datasets in modern society. In addition, the integration of these data with AI facilitates the development of new solutions and innovations, further increasing the societal significance of big data.

Big data can be categorized into three types: open, industrial, and personal [18]. Integrating and using these components effectively is desirable for societal advancement. However, industrial data are often confined to individual enterprises. Inter-enterprise data sharing faces significant barriers owing to the lack of shared data platforms and the challenges of ensuring data integrity, which have historically hindered its adoption. Creating additional business value is a strategic priority in the current business environment. This requires data collaboration beyond corporate boundaries, enabling the development of new business models, and addressing cross-industry challenges that cannot be resolved by a single organization.

Data reliability is a critical issue in inter-enterprise data sharing. Decisions based on erroneous data may have significant adverse effects on businesses and the society. Given the shared nature of data, errors originating from a single organization can spread to multiple stakeholders. Therefore, ensuring that the shared data are free from falsification or manipulation and that their provenance can be verified is essential.

Consequently, Blockchain has gained significant attention as a secure data-sharing platform. It validates all transactions executed in the system and manages them by linking verified transactions in a chain using hash values. Since all nodes in the network maintain the same ledger, blockchain ensures tamper resistance, fault tolerance, and thus provides highly reliable data management and traceability.

Blockchains can be classified into two types: permissionless and permissioned. Permissionless blockchains allow participation by an unspecified number of users and are employed as the underlying technology of cryptocurrencies such as Bitcoin [17]. In contrast, permissioned blockchains require prior authorization to join the network and are often used in closed environments such as inter-enterprise data sharing. Representative examples include Hyperledger Fabric [22] and GoQuorum [3].

However, blockchain's emphasis on data integrity comes at the cost of reduced write performance compared to conventional databases. This performance gap leads to transaction processing delays, posing significant challenges for applications that require real-time data synchronization across enterprise networks. Such delays are particularly problematic in scenarios where timely information exchange is critical, such as supply chain management and inter-enterprise data sharing.

Prior work [20] has identified the transaction ordering phase as a major throughput bottleneck in Hyperledger Iroha [23], a permissioned blockchain platform. To address this challenge, we propose a method that improves ordering phase efficiency through controlled timing of transaction batch generation, thereby enhancing overall transaction throughput. Following the approach of prior work, we target Hyperledger Iroha and build upon its findings to develop our solution.

To validate the practical applicability of our approach, we conduct performance evaluations under geographically distributed enterprise site conditions, with network latencies ranging from 0 to 5 ms, which represent typical inter-site

network environments. The main contributions of this work are as follows: (1) a method for optimizing batch generation timing in the ordering phase, and (2) validation of the proposed method's effectiveness under realistic network latency conditions.

II. Related Work

This section reviews existing studies on the performance evaluation and optimization of permissioned blockchains, as well as prior research on Hyperledger Iroha.

A. Performance Evaluation of Permissioned Blockchains

Baliga *et al.* [12] conducted a performance evaluation of Quorum, an Ethereum-based permissioned blockchain, by measuring variations in throughput and latency under different block times and consensus algorithms.

Oneda *et al.* [19] investigated changes in processing performance over time with varying concurrency levels for Hyperledger Fabric, Quorum, and Ethereum [5].

B. Optimization in Permissioned Blockchains

Thakkar *et al.* [21] evaluated the performance of Hyperledger Fabric and proposed optimizations based on their findings. They measured the impact of multiple parameters, such as block size and endorsement policies, on transaction throughput and latency. Their results identified major bottlenecks, including endorsement policy verification, sequential policy verification within blocks, and state verification and commit operations when using CouchDB [4]. They proposed and validated optimizations, such as caching and parallelization of endorsement policy verification, to improve throughput.

Nakaike *et al.* [16] analyzed the performance of Hyperledger Fabric database system and proposed several improvements. Their approach enhanced the performance by disabling GoLevelDB [6] compression and reducing the size of StateDB [10].

Javaid *et al.* [13] analyzed the performance of the validation phase in Hyperledger Fabric and redesigned this phase. By introducing chaincode information caching and parallel processing, they achieved improvements in throughput.

C. Performance Evaluation of Hyperledger Iroha

Muratov *et al.* [15] proposed optimizations for the voting step delay parameter in the Yet Another Consensus (YAC) algorithm implemented in Hyperledger Iroha. The experimental results demonstrated that the optimal delay value depends on the number of participating nodes, and that setting the delay peak appropriately for the network size can facilitate consensus formation while preventing throughput degradation.

Woznica *et al.* [11] evaluated the performance of Hyperledger Fabric, Hyperledger Sawtooth [9], and Hyperledger Iroha using a Hyperledger Caliper [7]. They measured the performance under varying network sizes, block sizes, numbers of submitted transactions, and network traffic distributions.

Most studies on the performance of permissioned blockchains have focused on Hyperledger Fabric, while research targeting Hyperledger Iroha remains insufficient. Therefore, this study focuses on Hyperledger Iroha, which is newer and practical implementation compared with Hyperledger Fabric, and proposes a method to improve transaction processing performance, followed by an evaluation of its effectiveness.

III. Hyperledger Iroha

A. Overview

Hyperledger Iroha is a private blockchain adopted as part of the Hyperledger Project [8]. It employs the YAC algorithm, which is Byzantine fault-tolerant. Compared with Practical Byzantine Fault Tolerance (PBFT), a representative consensus algorithm used in private blockchains, YAC can achieve consensus more efficiently. Although PBFT is vulnerable to attacks that can halt the consensus through simple scheduling mechanisms, YAC mitigates this issue by employing a dynamic peer list [15].

Hyperledger Iroha provides predefined commands that can be combined to implement functionalities equivalent to those of smart contracts. Because of its low central processing unit (CPU) and memory requirements, it is expected to be applicable to embedded systems such as IoT devices [14].

Currently, Hyperledger Iroha has been deployed in various countries and services. Notable examples include the central bank digital currency (CBDC) "Bakong" [1] issued by the National Bank of Cambodia, the campus currency "Byacco" [2] at the University of Aizu, as well as applications in smart contract-based insurance and authentication services that eliminate the need for IDs and passwords.

B. Architecture

The Hyperledger Iroha network is structured as shown in Fig. 1. Each node primarily comprises the following four components:

1) **Ordering service** – Synchronize with other nodes' ordering services and consolidate multiple transactions into a single proposal.
2) **Data validation process** – Verifies the validity of transactions based on the contents of the database.
3) **World state view (WSV)** – A database reflecting the current state of the Hyperledger Iroha ledger.
4) **Blockstore** – A distributed ledger that records the results of transaction processing.

C. Transaction processing

Fig. 2 illustrates the transaction-processing workflow in Hyperledger Iroha. Initially, the client submits a transaction to a nearby node. Upon receipt, the node performs stateless validation to verify that the transaction object, including its digital signature, is structured correctly. Transactions that pass stateless validation are forwarded to the ordering service.

The ordering service aggregates multiple transactions into a single proposal to improve the processing efficiency. Proposal creation is triggered by what first occurs:

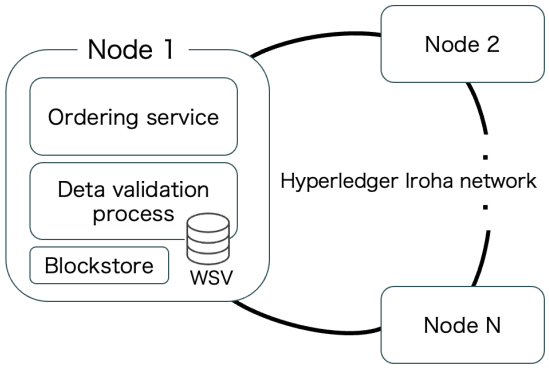

Fig. 1. Hyperledger Iroha architecture

- the expiration of the proposal creation timeout, or
- the accumulation of transactions reaching the maximum proposal size.

Once created, the proposal is propagated to all nodes in the network. Each node performs stateful validation by checking the transactions against its WSV, retaining only those that pass the validation. A block is then constructed from the validated transactions.

A consensus is reached among the participating nodes based on the generated blocks. Hyperledger Iroha employs the YAC algorithm, whereby consensus is achieved when more than two-thirds of the nodes confirm the correctness of a block's content. The agreed-upon block is then recorded in each node's block store and its results are reflected in the database.

Consequently, only transactions with guaranteed validity are recorded on the blockchain, ensuring that Hyperledger Iroha maintains an accurate and trustworthy ledger.

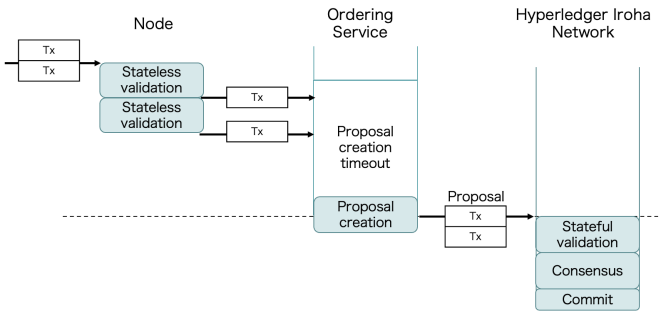

Fig. 2. Transaction processing

IV. Performance Limitations of the Existing System

A major performance bottleneck in the current implementation of Hyperledger Iroha is in the waiting time for the proposal creation event to be triggered. This bottleneck is influenced by two parameters: the *proposal creation timeout* and *maximum proposal size*.

Proposal Creation Timeout: This parameter affects both the resource consumption during idle system periods and the timing of proposal creation. Setting a longer timeout value can reduce the CPU and memory usage during idle states. However, because the timeout serves as a trigger for proposal creation, a longer value increases the delay between the arrival of a transaction and the creation of its corresponding proposal.

Maximum Proposal Size: This parameter takes effect when the number of received transactions reaches a specified limit, before the proposal creation timeout expires. Reducing the maximum proposal size allows this condition to be satisfied more frequently; however, this can lead to performance degradation under high transaction arrival rates.

Considering the server environment and performance characteristics of Hyperledger Iroha, both parameters must be set to relatively large values. However, this makes it more difficult to satisfy the proposal creation conditions, resulting in an increased waiting time between transaction arrival at the ordering service and proposal creation, which becomes a bottleneck in transaction processing.

Furthermore, both parameters are configured when the Hyperledger Iroha system is launched. Therefore, it is necessary to set these values appropriately based on the anticipated use cases and execution environment. However, in practice, the system may be operated under conditions different from those originally expected, rendering the configured parameter values suboptimal.

V. Proposed Method

The existing transaction collection algorithm in Hyperledger Iroha relies on parameters configured at startup, such as the proposal creation timeout and maximum proposal size. This dependence can introduce inefficiencies and delays in transaction processing. We propose a method that eliminates the parameter dependency by triggering proposal creation whenever transactions are available through an ordering service.

The behavior of the proposed method differs depending on whether the Hyperledger Iroha network is currently processing another proposal.

A. Case 1: No Ongoing Proposal Processing

The transaction processing flow in this case is illustrated in Fig. 3. When a transaction that has passed the stateless validation arrives at the ordering service, a proposal creation event is triggered immediately. The generated proposal contains all the transactions present in the ordering service at that moment. This proposal is then disseminated to all the participating nodes for processing. Unlike the conventional approach—where proposals are created only when the timeout expires or the maximum size is reached—this method initiates proposal creation without delay, thereby reducing unnecessary transaction waiting times.

B. Case 2: Ongoing Proposal Processing

The transaction processing flow in this case is illustrated in Fig. 4. In Hyperledger Iroha, only one proposal can be

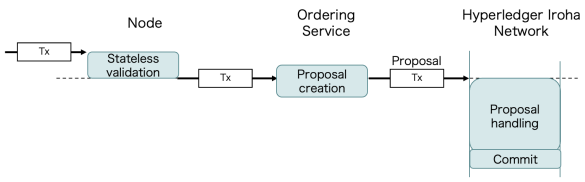

Fig. 3. Transaction processing flow without ongoing proposal handling.

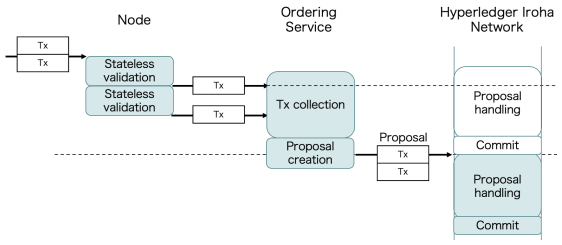

Fig. 4. Transaction processing flow with ongoing proposal handling.

processed at a time. Therefore, transactions arriving during the ongoing proposal processing are temporarily stored in the ordering service. Once consensus on the current proposal is reached and processing is completed, a new proposal is immediately generated using the transactions currently stored in the ordering service.

In the conventional approach, if the number of transactions arriving during proposal processing is below the maximum proposal size, the system waits until either the timeout expires or the maximum size is reached. By contrast, the proposed method initiates proposal creation immediately after completing the current proposal processing, provided that transactions are present in the ordering service. This is expected to further reduce the transaction waiting times.

C. Advantages

In the conventional method, proposal creation is triggered only when the timeout expires or the maximum proposal size is reached. By contrast, the proposed method allows each transaction to trigger proposal creation. This significantly reduces the waiting times in the ordering service and eliminates the need for parameter preconfiguration based on the anticipated operating environment, thereby enabling a more flexible and user-friendly system.

VI. Experimental Overview

A. Overview

This study evaluated the effectiveness of the proposed method under network latency conditions. We assumed Hyperledger Iroha to be a platform for inter-company data sharing, where the participating nodes were geographically distributed. Therefore, the evaluation was performed in an environment with artificial network latency.

Write transactions requiring consensus in the Hyperledger Iroha network were submitted, and the system throughput and latency were measured as evaluation metrics.

B. Experimental Setting

Experimental Environment: As shown in Fig. 5, three nodes were deployed on each of the three physical servers. Each node comprised an Ubuntu container running the Hyperledger Iroha process and a PostgreSQL container. All Docker containers were interconnected via a distributed network created using Docker's overlay networking.

Network delay was controlled using the `tc` command and applied only to interserver communication.

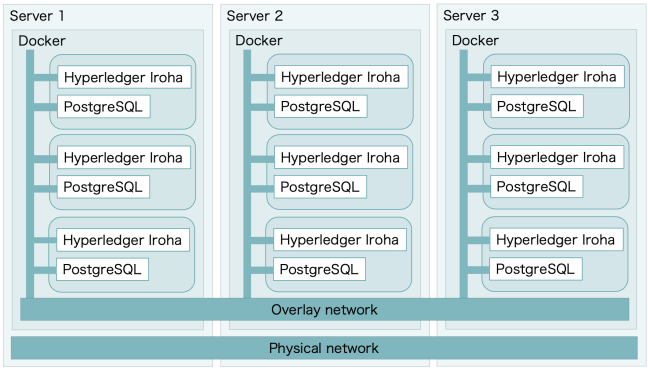

Fig. 5. Overview of the experimental environment.

Table I lists the specifications of the three servers used in the experiments. Although the servers shared identical operating systems and CPUs, their memory capacities differed.

TABLE I
SPECIFICATIONS OF THE PHYSICAL SERVERS USED IN THE EXPERIMENTS.

OS	Ubuntu 20.04 LTS
CPU	Intel(R) Xeon(R) Silver 4314 CPU @ 2.40GHz
Memory (Server 1)	192 GB
Memory (Server 2, 3)	512 GB

The Docker environment and Hyperledger Iroha version details are provided in Table II.

TABLE II
SOFTWARE VERSIONS USED IN THE EXPERIMENTS.

Docker	25.0.0
Ubuntu container	22.04 LTS
PostgreSQL container	9.5.25
Hyperledger Iroha	1.6.0

Network Latency: Intraserver communication delays among the nodes are considered negligible. For interserver communication, artificial latency was introduced using the `tc` command to emulate geographically distant sites. The delay values were set between 0 and 5 ms, assuming data sharing between the domestic sites. In this configuration, each server represents a cluster of geographically proximate nodes, and multiple clusters are distributed across sites.

Hyperledger Iroha Parameters: The proposal creation timeout was configured to a default value of 3000 ms, and the maximum proposal size was set to 100.

VII. Experimental Results

A. Comparison Under Identical Latency Conditions:

Figures 6 and 7 present the throughput and latency results for network delays in the range of 0–2 ms, whereas Figures 8 and 9 show the results for 3–5 ms.

Observation 1 (Figures 6–9): The proposed method achieved higher throughput and lower latency than the conventional method under all delay conditions. Throughput was improved by an average of 26.1%, with a larger improvement observed at transaction sending rates above 110. Latency was reduced by an average of 67.7%, and the reduction was more significant at lower transaction sending rates.

Observation 2 (Figures 6 and 8): For both methods, throughput increased with the transaction sending rate, while the growth rate of throughput declined at higher sending rates. However, in the conventional method, throughput temporarily increased at a sending rate of 100. This phenomenon is considered to occur because the number of transactions sent at once matched the maximum proposal size, leading to immediate proposal creation.

Observation 3 (Figures 7 and 9): The change in latency with respect to the transaction sending rate differed between the two methods. In the conventional method, latency was high at low sending rates, decreased between rates of 30 and 100, and then increased thereafter. In contrast, in the proposed method, latency increased monotonically as the transaction sending rate increased. These results indicate that the conventional method suffers from a large overhead caused by transaction waiting time.

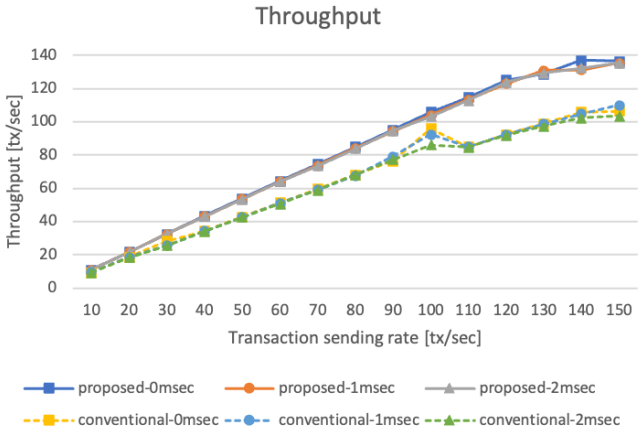

Fig. 6. Throughput for network delay in the range of 0–2 ms

B. Performance Variation under Changing Network Latency:

The throughput behavior of the proposed method under different network latencies is illustrated in Fig.10. In contrast, Fig.11 presents the corresponding results for the conventional method.

Observation 1 (Figure 10): For the proposed method, no significant change in throughput is observed when the network

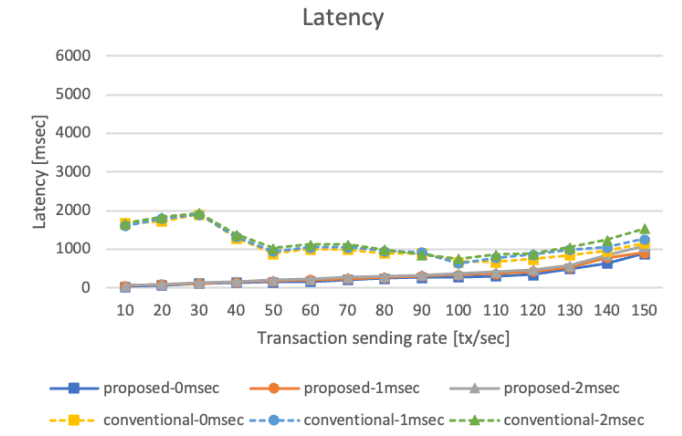

Fig. 7. Latency for network delay in the range of 0–2 ms

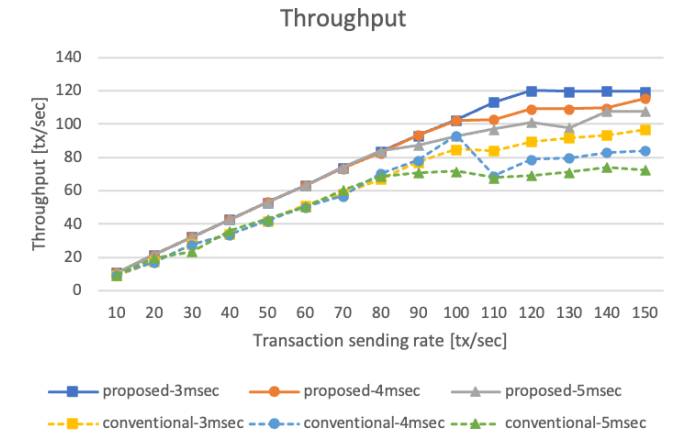

Fig. 8. Throughput for network delay in the range of 3–5 ms

latency ranges from 0 ms to 2 ms. Moreover, even with increased network latency, the throughput at low transaction submission rates remains unaffected. The onset of throughput saturation occurs at a submission rate of 130 when the latency is 3 ms, at 110 when the latency is 4 ms, and at 90 when the latency is 5 ms.

Observation 2 (Figure 11): For the conventional method, deviations from the linear growth trend occur at a transaction submission rate of 110 for latencies between 0 ms and 4 ms, and at 90 for 5 ms. Therefore, when comparing the throughput variations of the proposed and conventional methods, the proposed method can sustain higher transaction submission rates under low-latency conditions without performance degradation.

Observation 3 (Figures 10 and 11): Table III summarizes the average throughput performance drop under different network latency conditions. The proposed method exhibits smaller performance drops compared to the conventional method. In particular, under a 5 ms latency condition, the average performance drop of the proposed method is approxi-

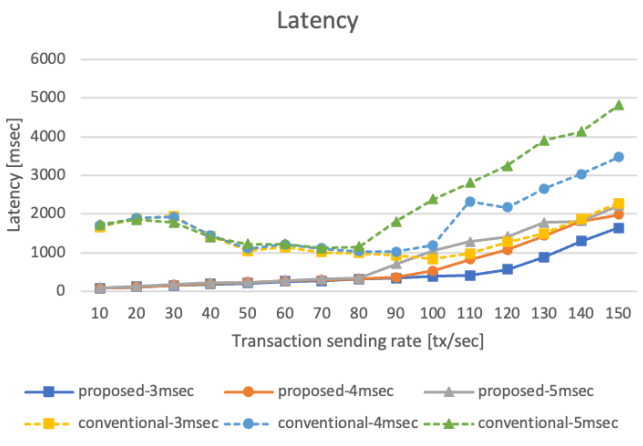

Fig. 9. Latency for network delay in the range of 3–5 ms

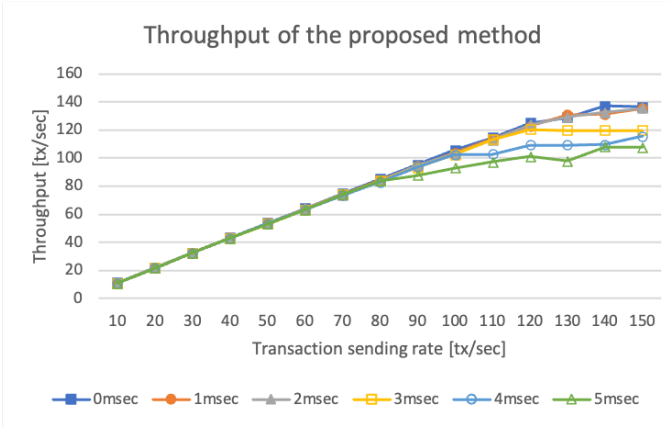

Fig. 10. Throughput variations of the proposed method

mately 8.6%, whereas that of the conventional method is about 11.3%.

TABLE III
THROUGHPUT DEGRADATION RATE UNDER DIFFERENT NETWORK LATENCY CONDITIONS

Network latency	Conventional method	Proposed method
1 ms	0.6	0.8
2 ms	2.2	1.0
3 ms	3.5	3.2
4 ms	7.1	5.8
5 ms	11.3	8.6

Observation 4 (Figures 10 and 11): We compare the maximum throughput obtained at latencies of 0ms and 5 ms for both methods. For the conventional method, the maximum throughput decreases from 106.5 at 0 ms to 74.1 at 5 ms. In contrast, for the proposed method, the maximum throughput decreases from 137.0 at 0 ms to 107.9 at 5 ms. This corresponds to a performance drop of 30.4% for the conventional method, compared to 21.2% for the proposed method. These results demonstrate that the proposed method exhibits resilience to latency, maintaining stable performance

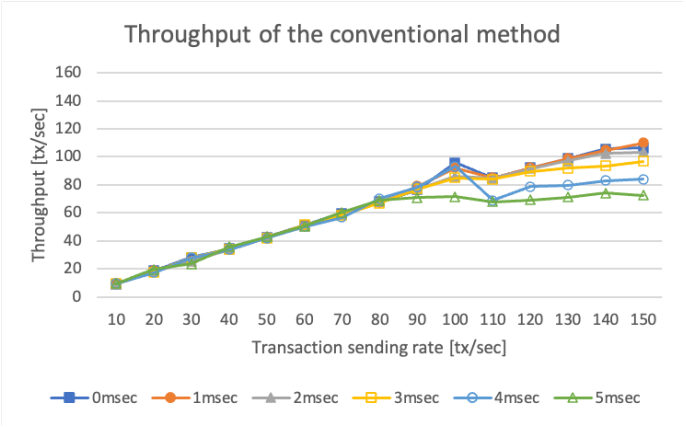

Fig. 11. Throughput variations of the conventional method

even as network latency increases.

C. Discussion of Results

This section discusses the effectiveness of the proposed method. The objective of this study is to enhance the transaction processing performance of Hyperledger Iroha in the context of inter-enterprise data sharing. In such scenarios, the network environment is unlikely to be a low-latency connection within the same organization; instead, it is expected to involve communication between different organizations located at a certain distance.

From the comparative experiments, improvements in transaction processing performance with the proposed method were observed under all tested network latency conditions ranging from 0 to 5 ms. Moreover, the proposed method demonstrated resilience to latency, showing less susceptibility to performance degradation compared to the conventional method, even when network delays were present.

Therefore, the proposed method, which is capable of sustaining high performance even under network latency conditions, can be regarded as a favorable approach for inter-organizational data sharing environments where network delays are inevitable.

VIII. CONCLUSION

This study aimed to enhance the data-writing performance of Hyperledger Iroha for use as an inter-enterprise data-sharing platform. We proposed a method to alleviate the transaction processing bottleneck by removing the dependency of proposal creation on the proposal creation timeout and maximum proposal size parameters, instead triggering proposal creation whenever transactions are present in the ordering service.

Performance was evaluated in an environment with network latency to simulate practical deployment conditions. The results confirmed that the proposed method improved transaction throughput and exhibited lower performance degradation under increased network latency than the conventional method. In particular, under a 5 ms latency, the average throughput degradation rate was reduced from 11.3% (conventional) to

8.6% (proposed). These findings demonstrate that the proposed method provides both performance improvement and latency tolerance.

ACKNOWLEDGMENT

This study was partly supported by JST CREST JPMJCR22M2.

REFERENCES

[1] Bakong. https://bakong.nbc.gov.kh/en/. Accessed: 2025-5-10.
[2] Byacco. https://soramitsu.co.jp/byacco. Accessed: 2025-8-14.
[3] Consensys goquorum. https://docs.goquorum.consensys.io. Accessed: 2025-3-17.
[4] Couchdb. http://couchdb.apache.org/. Accessed: 2025-11-16.
[5] Ethereum. https://www.ethereum.org/. Accessed: 2025-3-17.
[6] goleveldb. https://github.com/syndtr/goleveldb. Accessed: 2025-11-16.
[7] Hyperledger caliper. https://github.com/hyperledger-caliper/caliper. Accessed: 2024-12-28.
[8] Hyperledger foundation. https://www.hyperledger.org/. Accessed: 2024-12-28.
[9] Hyperledger sawtooth. https://github.com/hyperledger-archives/sawtooth-core. Accessed: 2025-11-16.
[10] statedb. https://github.com/cilium/statedb. Accessed: 2025-11-16.
[11] Woznica Arnold and Kedziora Michal. Performance and scalability evaluation of a permissioned blockchain based on the hyperledger fabric, sawtooth and iroha. *Computer Science and Information Systems 2022*, 19(2):659–678, 2022.
[12] Arati Baliga, Subhod I, Pandurang Kamat, and Siddhartha Chatterjee. Performance evaluation of the quorum blockchain platform, 2018.
[13] Haris Javaid, Chengchen Hu, and Gordon Brebner. Optimizing validation phase of hyperledger fabric. *2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 269–275, 2019.
[14] Ikkei Matsuda. Real use case of blockchain projects abroad —examples of japanese startup, who develops the world de fact blockchain standard. *The journal of science policy and research management*, 34(4):377–387, 2019.
[15] Fedor Muratov, Andrei Lebedev, Nikolai Iushkevich, Bulat Nasrulin, and Makoto Takemiya. Yac: Bft consensus algorithm for blockchain, 2018.
[16] Takuya Nakaike, Qi Zhang, Yohei Ueda, Tatsushi Inagaki, and Moriyoshi Ohara. Hyperledger fabric performance characterization and optimization using goleveldb benchmark. *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–9, 2020.
[17] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, 2008. Accessed: 2024-12-28.
[18] Ministry of Internal Affairs and Japan Communications. Information and communications in japan 2017 (summary). ”https://www.soumu.go.jp/johotsusintokei/whitepaper/eng/WP2017/chapter-2.pdf. Accessed: 2025-5-13.
[19] Rintaro Oneda, Yoshiki Akita, Ganhin Ka, Hinata Takebayashi, Takuto Ogura, and Takayuki Suzuki. Blockchain infrastructure software performance verification: A comparative analysis of hyperledger fabric, quorum, and ethereum (in japanese). *Journal of digital practices*, 10(3):506–523, 2019.
[20] Akiho Sakamoto and Masato Oguchi. A study on blockchain transaction processing methods using dummy transactions (in japanese). *17th Forum on Data Engineering and Information Management (DEIM)*, 2025.
[21] Parth Thakkar, Senthil Nathan, and Balaji Viswanathan. Performance benchmarking and optimizing hyperledger fabric blockchain platform. *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 264–276, 2018.
[22] LF Decentralized Trust. Hyperledger fabric. https://www.lfdecentralizedtrust.org/projects/fabric. Accessed: 2025-3-17.
[23] LF Decentralized Trust. Iroha. https://www.lfdecentralizedtrust.org/projects/iroha. Accessed: 2024-12-28.