This article has been accepted for inclusion in a future issue of this magazine.

Facing to Latency of Hyperledger Fabric for Blockchain-enabled IoT: Modeling and Analysis

Sungho Lee, Minsu Kim, Jemin Lee, *Member, IEEE*, Ruei-Hau Hsu, *Member, IEEE*, Min-Soo Kim, and Tony Q. S. Quek, *Fellow, IEEE*

Abstract—**Hyperledger Fabric (HLF), one of the most popular permissioned blockchains, has recently received attention for blockchain-enabled Internet of Things (IoT). However, for IoT applications to handle time-sensitive data, the processing latency in HLF has emerged as a new challenge. In this article, therefore, we establish a practical HLF latency model for HLF-enabled IoT. We first discuss the structure and the transaction flow of HLF-enabled IoT. After implementing real HLF, we capture the latencies that each transaction experiences and show that the total latency of HLF can be modeled as a Gamma distribution, which is validated by conducting a goodness-of-fit test (i.e., the Kolmogorov-Smirnov (KS) test). We also provide the parameter values of the modeled latency distribution for various HLF environments. Furthermore, we explore the impacts of three important HLF parameters including the average transaction generation rate, block size, and block-generation timeout on the HLF latency. As a result, this article provides design insights on minimizing the average latency for HLF-enabled IoT.**

I. Introduction

Since blockchain technology was developed as a novel solution to ensure data integrity in distributed systems, various blockchain-enabled Internet of Things (IoT) applications have recently emerged to address privacy and security vulnerabilities [1]. In the blockchain-enabled IoT applications, delivered IoT data is appended to the assigned remote blockchain storage. Blockchains can be classified into public (permissionless) blockchain and permissioned blockchain. In the permissioned blockchain, according to granting permission for read and write, there are also two types (i.e., consortium and private blockchains). The permissioned blockchain has recently received much attention in terms of fast transaction processing, privacy preservation, and lower power consumption [2]. Particularly, Hyperledger Fabric (HLF)[1], which is hosted by Linux Foundation and contributed by IBM since 2015, is gaining popularity [1] as one of the promising permissioned blockchains. Generally, the permissioned blockchain is less scalable than the public blockchain [3]. However, HLF can also have high scalable [4], so due to the scalability and the benefits of HLF, it has been used for many IoT applications.

As discovered in literature [5]–[8], HLF-enabled IoT applications include industrial IoT (IIoT), healthcare, wireless monitoring, and unmanned aerial vehicles (UAVs). For example, HLF is used for 5G IIoT-enabled cloud manufacturing using fog computing in [5]. In [6], decentralized IIoT data marketplace is proposed using HLF. For smart healthcare, HLF is exploited for IoT to share monitored patient vital signs [7] and to ensure security and privacy for medical data in the Healthcare 4.0 industry using UAVs [8]. Thus, it is expected that HLF will be more widely used to securely manage a high volume of IoT data.

However, in blockchain platforms including HLF, high latency is still recognized as a major issue [1]. Specifically, certain IoT applications, which are handling time-critical data, may have strict latency requirements for making correct and useful decision [9], and require fast transaction commitment. For example, vital signs of patients need to be promptly processed and transferred to smart hospital blockchain networks for effective medical care [7]. Therefore, for those applications, it is important to predict and understand the HLF latency, prior to their practical implementations. Nonetheless, in most works on HLF-enabled applications, the HLF latency is generally not considered or simply ignored, and the latency characteristics including its distribution have not been fully explored.

Recently, there have been some works that analyze the HLF latency [10]–[13].[2] Specifically, the latency is defined and measured based on several theoretical models including stochastic reward net (SRN) [10], generalized stochastic petri nets (GSPNs) [11], a hierarchical model based on transaction execution and validation [12], and a queueing-based model [13].

However, in all the previous works above, only the average latency is presented without showing the latency distribution, and in most works, the theoretical models are not verified in real HLF. When the latency is measured in real HLF, it is

Corresponding author is J. Lee.

S. Lee is with the Department of Electrical Engineering and Computer Science, Daegu Gyeongbuk Institute of Science and Technology (DGIST), Daegu 42988, Republic of Korea (e-mail: seuho2003@dgist.ac.kr).

M. Kim is with the Wireless@VT Group, Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061, USA (e-mail: msukim@vt.edu)

J. Lee is with the Department of Electrical and Computer Engineering, Sungkyunkwan University (SKKU), Suwon 16419, Republic of Korea (e-mail: jemin.lee@skku.edu).

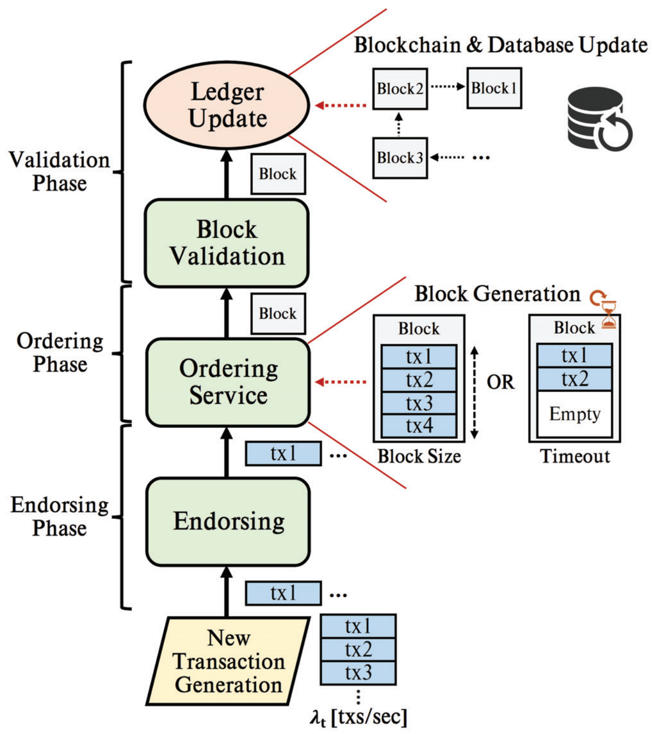
R. -H. Hsu is with the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung 80424, Taiwan. (e-mail: rhhsu@mail.cse.nsysu.edu.tw).

M. -S. Kim is with the School of Computing, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 34141, Republic of Korea (e-mail: minsoo.k@kaist.ac.kr).

T. Q. S. Quek is with Information Systems Technology and Design Pillar, Singapore University of Technology and Design, Singapore 487372 (e-mail: tonyquek@sutd.edu.sg).

[1] https://www.hyperledger.org/use/fabric.

[2] Those works are based on HLF v1.0 or higher, which exploits Apache Kafka/ZooKeeper for consensus.



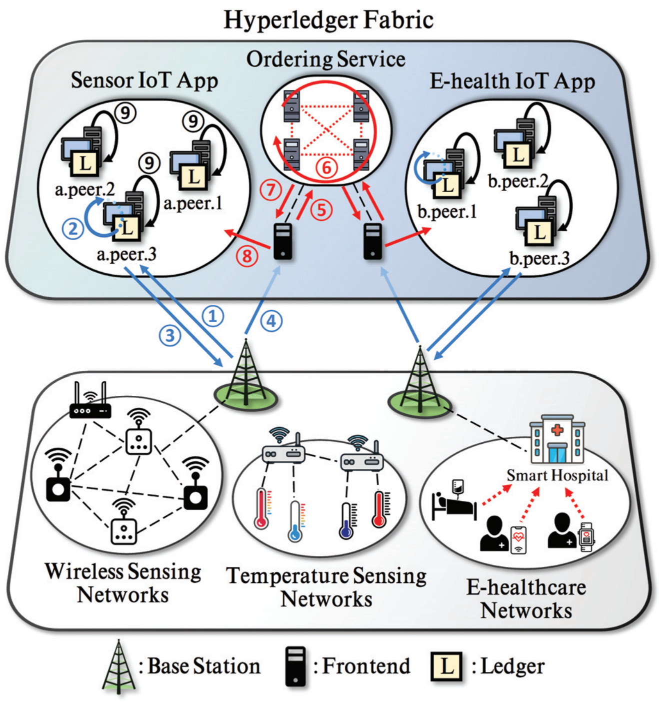


Fig. 1: Flowchart of HLF transaction processing (left) and structure of HLF-enabled IoT (right).

quite random. This randomness makes challenges in providing reliable HLF-based IoT systems, such as [7], especially when the latency-sensitive services are required. When the IoT service requires to complete the transaction processing within certain target latency for most of the time, the latency distribution can be more importantly and usefully used than the average latency-based results. Furthermore, HLF has a sophisticated structure, where many network parameters affect the latency simultaneously, so their impacts on the latency should be carefully investigated to lower the HLF latency.

In this article, therefore, we establish a latency model of HLF especially for HLF-enabled IoT, where time-sensitive data is generally managed. After characterizing the latency distribution, we also analyze the impacts of three important HLF parameters (i.e., the block size, block-generation timeout, and average transaction generation rate) on the HLF latency. We then discuss how the average HLF latency can be reduced by setting those parameters properly. The contributions of this article can be summarized as follows:

- We newly provide the probability distribution of the HLF latency by performing the probability distribution fitting on transaction latency samples, which are captured in real HLF. We also provide the parameter values of corresponding probability distributions for various HLF parameter setups. To the best of our knowledge, this is the first work that provides the distribution of the HLF latency.
- We develop the latency model of HLF with a new structure, which is adopted in the latest HLF releases. Hence, our work is more compatible with current HLF-enabled IoT, compared to other HLF latency models for the previous HLF structure.
- We explore the impacts of three main HLF parameters (i.e., the block size, block-generation timeout, and average transaction generation rate) on the HLF latency. Especially, we figure out the existence of the best values of those HLF parameters that minimize the HLF latency. This provides some design insights on HLF parameters to lower the HLF latency for more reliable HLF-enabled IoT.

II. HLF-ENABLED IOT

In this section, we first describe HLF and HLF-enabled IoT. We then demonstrate the transaction flow consisting of three phases, and provide the definitions of some important parameters in HLF.

HLF is a permissioned blockchain platform for a modular architecture, which is one of the sub-projects in the Hyperledger Project. In the recent version of HLF, Apache Kafka and Apache ZooKeeper are adopted for transaction collection and block generation. Apache Kafka is to generate new blocks after collecting transactions with the resistance to crash faults, and Apache ZooKeeper is to manage Apache Kafka nodes in terms of task coordination, cluster membership, and access control.

Figure 1 shows a flowchart of HLF transaction processing (left) and a structure of HLF-enabled IoT (right), where IoT data are managed in HLF. As shown in the right side of Fig. 1, generated data are delivered from IoT devices to a base station (BS) connected to HLF. The receiving BS transmits the data to HLF as new transactions, and the transactions are processed in the three phases as shown in the left side of Fig. 1. Specifically, the transactions are executed by the endorsing peer(s) (i.e., a.peer.3) and conveyed to the ordering service. The ordering service generates a new block using the transactions, and the block is appended to the ledger after validation at each peer. The transaction processing inside HLF is described in detail as below.

A. Transaction Flow

As shown in the left side of Fig. 1, HLF appends transactions to the ledger through three phases: endorsing phase, ordering phase, and validation phase. The processing phases enable HLF networks not only to support general-purpose programming languages, but also to remain tamper-proof. In this subsection, we describe the transaction flow in HLF for data updates.

1) Endorsing Phase: The endorsing phase is to check whether a transaction has been well-formed, authorized, and not been submitted in the past. When a transaction is generated, the endorsing peers independently execute the conveyed transaction based on the data in their copied ledgers, and returns the result to the client. If all of the results are identical, this means all copied ledgers have the same data for the targeted one of the transaction (i.e., synchronized). Then, the transaction is delivered to the ordering service, otherwise, the transaction is discarded. By doing this, the endorsing phase can prevent ledger divergence, which results from data inconsistency among the copied ledgers of the endorsing peers.

2) Ordering Phase: The ordering phase is to generate new blocks by ordering all transactions chronologically. Note that, as explained earlier, the ordering service is a node cluster, where a pre-defined consensus protocol is conducted for block generation. The consensus protocol is a method to achieve consensus on block generation among ordering nodes, such as Kafka/ZooKeeper in HLF. A newly generated block is transmitted to peers for the validation phase. Here, we focus on the Kafka/ZooKeeper-based ordering service, which has been currently the most widely used [9].

3) Validation Phase: The validation phase is the last phase in which the new block is validated by each peer individually. In this phase, each peer that receives the block conducts mainly 1) validation system chaincode (VSCC) to check whether the transactions in the block have been properly endorsed and 2) multi-version concurrency control (MVCC) to check whether the target data in the copied ledgers still have the same versions as those in the endorsing phase. The block is appended to the ledger after the validation at each peer. From the successfully validated transactions, their data are updated in the ledger. If any verification is failed, the transaction is marked as an invalid one and prohibited from updating the ledger.

B. Important HLF Parameters

In this subsection, we define three important HLF parameters, which mainly affect the HLF latency.

1) Block Size (S_b): The block size S_b is one of the configurable parameters of the ordering service. This parameter defines the maximum number of transactions in one block. In other words, the block size determines how many transactions the ordering service can collect the most in one block. The waiting transactions are promptly exported from the ordering service as a new block, when the number of transactions in the waiting queue reaches a pre-defined block size value.

2) Block-generation Timeout (T_b): The block-generation timeout T_b is another configurable parameter of the ordering service. This parameter defines the maximum time to wait for other transactions to be exported as a new block since the first transaction arrived at the waiting queue. Once the timer expires, the waiting transactions are included into a new block, regardless of their number in the waiting queue.

3) Average Transaction Generation Rate (λ_t): The average transaction generation rate λ_t refers to the number of transactions arrived in HLF per second. Being different from the two configurable parameters above, this parameter can be determined by many factors. For example, when an IoT device conveys data to HLF as transactions, the communication channel and data generation frequency of the device can affect this parameter.

III. HLF Latency Modeling

In this section, we conduct HLF latency modeling based on the probability distribution fitting method. We first define latency types in HLF, and provide the experimental setup of our HLF networks. For the probability distribution fitting method, we first capture transaction latency samples from the HLF networks. We then find the best-fit distribution for each latency type, and finally provide our HLF latency model. For evaluating goodness-of-fit of our modeling, we also conduct the Kolmogorov-Smirnov (KS) test. The feasibility conditions for our latency modeling are discussed with providing the cases, for which the probability distribution fitting method does not work well.

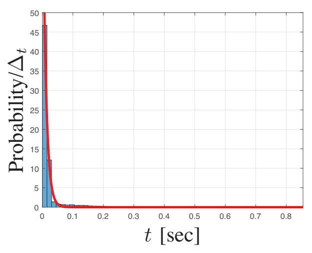
A. Latency Types in HLF

Before modeling the HLF latency, we first divide the total elapsed time of the data in the HLF network according to the transaction-processing stages, such as the endorsing latency, the ordering latency, the validation latency, and the ledger-commitment latency (i.e., the total latency). Note that we exclude the communication latency spent from an IoT device to reach the HLF network, as it is more depending on the communication standard that IoT devices use, and it has already been explored in many literature.

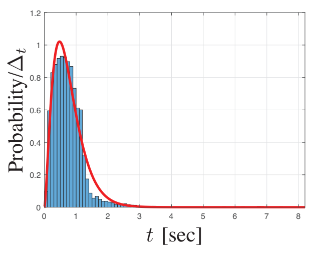
1) Endorsing Latency: The endorsing latency refers to the time taken to receive the transaction execution results from the endorsing peers since a transaction was generated. To be specific, it is defined as the sum of the latencies for step ①, ②, ③, and ④ in Fig. 1.

2) Ordering Latency: The ordering latency refers to the time taken to await until the transaction is exported as a new block from the ordering service. It is defined as the sum of the latencies for step ⑤, ⑥, ⑦, and ⑧ in Fig. 1. Note that the latency for step ⑥ consists of the consensus time and the waiting time for other transactions to be released as part of the new block.

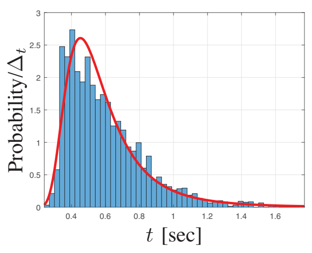
In contrast, the latency for step ⑧ is the sum of the transmission time and the waiting time of the released block for the validation phase. Thus, it may be considered as part



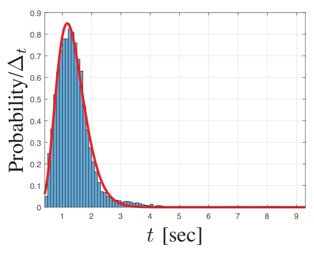
(a) Endorsing latency: Exponential dist. PDF [$\lambda = 94.5$]



(b) Ordering latency: Gamma dist. PDF [$\alpha_{\rm o} = 2.652, \beta_{\rm o} = 3.458$]



(c) Validation latency: Fréchet dist. PDF [$\alpha = 0.21, s = 0.479, m = 0.144$]



(d) Ledger-commitment latency: Gamma dist. PDF [$\alpha_{\rm t} = 7.362, \beta_{\rm t} = 5.445$]

Fig. 2: Histograms (blue bars) and their best-fit distributions (red lines) of the endorsing latency, ordering latency, validation latency, and ledger-commitment latency, where the block size $S_{\rm b}$ is 10, the block-generation timeout $T_{\rm b}$ is 1 [sec], and the average transaction generation rate $\lambda_{\rm t}$ is 10 [transactions/sec]. Note that Δ_t is the width of a bin.

of the validation latency. In this article, however, we include this latency into the ordering latency, when we depict the histograms in Figs. 2 and 4. Note that the latency for step ⑧ can be high if there are many blocks, waiting for the validation. Hence, in this article, the ordering latency is generally higher than the endorsing and validation latencies, forming the same distribution as that of the ledger-commitment latency.

3) Validation Latency: The validation latency refers to the time taken to validate and commit the block with ledger update. This latency is defined as the latency for step ⑨ in Fig. 1. All transactions in the same block are sequentially validated, but committed to the ledger all together as a batch. Therefore, they have the identical validation latency to each other. Note that the validation latency may vary according to peer's type. In this article, this latency is measured at the endorsing peer.

4) Ledger-commitment Latency (Total Latency): The ledger-commitment latency refers to the time taken to completely process the transaction from the beginning. Thus, it is the sum of the endorsing, ordering, and validation latencies.

B. Experiment Setup

For the implementation of our HLF network, we exploit HLF v1.3 on one physical machine using Docker, which is a software platform to implement a distributed system based on isolated Docker containers. The machine is with Intel® Xeon W-2155 @ 3.30 GHz processor and 16 GB RAM. When HLF is deployed on one physical machine, nodes communicate with each other through the Google remote procedure call (gRPC) [4]. In contrast, when HLF is deployed on multiple physical machines, transactions and blocks can experience the networking latency (i.e., wired communication latency) among the machines. However, generally, the networking latency is small, compared to the processing latency in HLF, so our latency modeling can still be effective even on multiple machines, which were also validated by our additional experiments with four machines.[3]

We bring up a Kafka/ZooKeeper-based ordering service consisting of four Kafka nodes, three ZooKeeper nodes, and three frontends that are connected to the Kafka cluster in the network (i.e., the minimum Kafka and ZooKeeper nodes for crash fault tolerance). Note that the frontend is not only to inject transactions from clients into the Kafka node cluster, but also to receive new blocks, which will be disseminated to peers. The HLF network has one endorsing peer and two committing peers.[4]

We perform 10 test runs for one experiment setup, and each test run includes 1,000 transaction proposals transmitted. All proposals attempt to update a key-value set, using appropriate parameters for the *changeCarOwner* function defined in the *Fabcar* chaincode[5]. Each of the equipped frontends has an equal probability of being selected to relay a transaction from our client to the Kafka node cluster, and to make a new block propagated. Note that all proposals in one test run do not access the same key-value set. Hence, we do not consider MVCC violation during the validation phase in this article.

Transactions are generated at the average rate of $\lambda_{\rm t}$ [transactions/sec], whose arrival model is a Poisson process.[6] At each iteration of test runs, we not only vary the average transaction generation rate $\lambda_{\rm t}$, but also measure the endorsing, ordering, validation, and ledger-commitment latency in order to analyze the impact of increasing $\lambda_{\rm t}$. Note that $S_{\rm b}$ and $T_{\rm b}$ refer to the block size and block-generation timeout, respectively.

C. Latency Modeling

For the latency modeling, we first obtain the histogram for each latency type defined earlier, and then figure out its best-fit probability distribution. As an example, Fig. 2 shows the latency histograms, obtained from the experiments and the discovered best-fit distributions (red lines) for the endorsing, ordering, validation, and ledger-commitment latencies. Here, we use the histogram fitting function (i.e, *histfit*) in Matlab

[3]When Kafka nodes, ZooKeeper nodes, frontends, and peers are deployed on four different machines using Docker Swarm for $S_{\rm b} = 10$, $T_{\rm b} = 2$ [sec], and $\lambda_{\rm t} = 10$ [transactions/sec], the measured latency is well-matched with the Gamma distribution with $\alpha_{\rm t} = 1.283$ and $\beta_{\rm t} = 2.267$.

[4]From numerous experiments, we found that our HLF latency model can be valid with different numbers of nodes (e.g., endorsing, Kafka, and committing nodes), except for the modeling infeasible cases, discussed later.

[5]https://github.com/hyperledger/fabric-samples/blob/release-1.3/chaincode/fabcar/go/fabcar.go.

[6]The Poisson process is the most widely used model for the random data/packet generation as also used in 3GPP [14]. The periodic data generation is also widely used, especially in the sensor networks. However, it makes the ledger-commitment latency less random and more predictable, so it is not considered in this article.

for available 21 distributions, and then determine the best-fit distribution for each latency type as follows.

1) Endorsing Latency: The best-fit distribution of the endorsing latency T_e is the exponential distribution $T_\text{e} \sim \text{Exp}(\zeta)$ (also see Fig. 2a), whose probability density function (PDF) and cumulative distribution function (CDF) are defined as

$$f_{T_\text{e}}(t) = \zeta e^{-\zeta t}, \qquad F_{T_\text{e}}(t) = 1 - e^{-\zeta t},$$

where $\zeta = \frac{1}{\mathbb{E}[T_\text{e}]}$ is the rate parameter.

2) Ordering and Ledger-commitment Latencies: The best-fit distributions of the ordering latency T_o and ledger-commitment latency T_t are the Gamma distribution, $T_\text{o} \sim \text{Gamma}(\alpha_\text{o}, \beta_\text{o})$ and $T_\text{t} \sim \text{Gamma}(\alpha_\text{t}, \beta_\text{t})$, respectively (also see Figs. 2b and 2d), whose PDF and CDF are defined as

$$f_{T_i}(t) = \frac{{\beta_i}^{\alpha_i}}{\Gamma(\alpha_i)} t^{\alpha_i - 1} e^{-\beta_i t}, \;\; F_{T_i}(t) = \frac{1}{\Gamma(\alpha_i)} \gamma(\alpha_i, \beta_i t),$$

where i is to indicate the ordering latency ($i = \text{o}$) and ledger-commitment latency ($i = \text{t}$), $\Gamma(\alpha_i) = \int_0^\infty x^{\alpha_i - 1} e^{-x}\, \text{d}x$ is the Gamma function, and $\gamma(\alpha_i, \beta_i t) = \int_0^{\beta_i t} x^{\alpha_i - 1} e^{-x}\, \text{d}x$ is the lower incomplete Gamma function. Note that $\alpha_i = \beta_i \mathbb{E}[T_i]$, where α_i and β_i are the shape and rate parameters, respectively.

3) Validation Latency: The best-fit distribution of the validation latency T_v is the three-parameter Fréchet distribution $T_\text{v} \sim \text{Frechet}(\alpha, s, m)$ (also see Fig. 2c), whose PDF and CDF are defined as

$$f_{T_\text{v}}(t) = \frac{\alpha}{s} \left(\frac{t-m}{s}\right)^{-1-\alpha} e^{-\left(\frac{t-m}{s}\right)^{-\alpha}}, F_{T_\text{v}}(t) = e^{-\left(\frac{t-m}{s}\right)^{-\alpha}},$$

where α, s, and m are the shape, scale, and location parameters, respectively.

Once the best-fit distributions are determined, we estimate the distribution parameters, such as ζ for the endorsing, (α, s, m) for the validation, and $(\alpha_\text{o}, \beta_\text{o})$ and $(\alpha_\text{t}, \beta_\text{t})$ for the ordering and ledger-commitment latencies. Specifically, we use the curve fitting tool (i.e., *cftool*) in Matlab to obtain the best-fit parameters of a given best-fit distribution. From Fig. 2, we can see that the discovered best-fit distributions match well with the histograms, especially with that of the ledger-commitment latency.

For the ledge-commitment latency, we obtain the best-fit Gamma distribution parameter pairs of the ledger-commitment latency for various HLF environments. Detailed parameter pairs are provided as a supplement on the archival version.[7] Note that the parameter ranges, used in this article, are in the ranges that other works used including [15]. The CDFs of some empirically measured values and their best-fit Gamma distributions are then demonstrated in Fig. 3. From Fig. 3, we can see that the CDFs of the best-fit Gamma distribution matches well with the empirically obtained ones.

D. Latency Modeling Validation

In order to validate and improve the reliability of our latency model, we conduct the KS test, which is a general test that evaluates how empirical and best-fit distributions match well with each other. If the KS statistics value, generated by the KS test, is smaller than a critical value, the empirical distribution can be considered as a well-matched one. Fig. 3 shows the ledger-commitment latency, when $S_\text{b} = 10$ and $T_\text{b} = 2$ [sec] for $\lambda_\text{t} = 8, 9, 10$ and 11, and the critical value is computed as 0.0513. Each of their KS statistics values is averaged over 10 test runs at 0.01 significance level and respectively computed as 0.0388, 0.0437, 0.0457, and 0.0444. We can see that they are all smaller than the critical value. Hence, the test is passed for all the cases, and we can conclude that the models are reliable. Note that our archival version includes more numerical data obtained from various HLF environments and their KS test results as explained earlier.

[7]https://arxiv.org/abs/2102.09166.

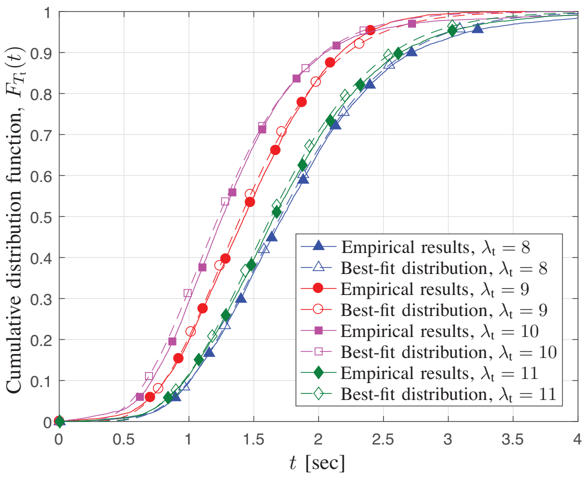


Fig. 3: Empirical CDFs and the best-fit Gamma distribution CDFs of the ledger-commitment latency T_t for different values of the average transaction generation rates λ_t [transactions/sec], where the block size S_b is 10 and the block-generation timeout T_b is 2 [sec].

E. Feasibility Conditions for Latency Modeling

During the modeling process, it is found that there exist some cases in which histograms of T_t do not fit well into a known probability distribution. In other words, the probability distribution fitting method is not always applicable. We have discovered two environments that make the probability distribution fitting method infeasible as follows.

1) Timeout-dominant Block Generation Environment: When the average transaction generation rate λ_t is small, transactions arrive in the ordering phase slowly. In this case, most blocks are generated when the timer set to T_b expires as each block is seldom fully filled with transactions (i.e., up to the amount of block size before timeout). In this case, a large portion of transactions can have the ordering latency equal to T_b, resulting in a peak on the ordering latency histogram around T_b (i.e., 1.2 [sec]), as shown in Fig. 4a. Due to this peak, the histogram of T_t in Fig. 4b cannot match with any known probability distributions.

2) Block Size-dominant Block Generation Environment: When transactions are relatively generated fast (i.e., high λ_t), most blocks are filled with transactions up to S_b fast, so blocks

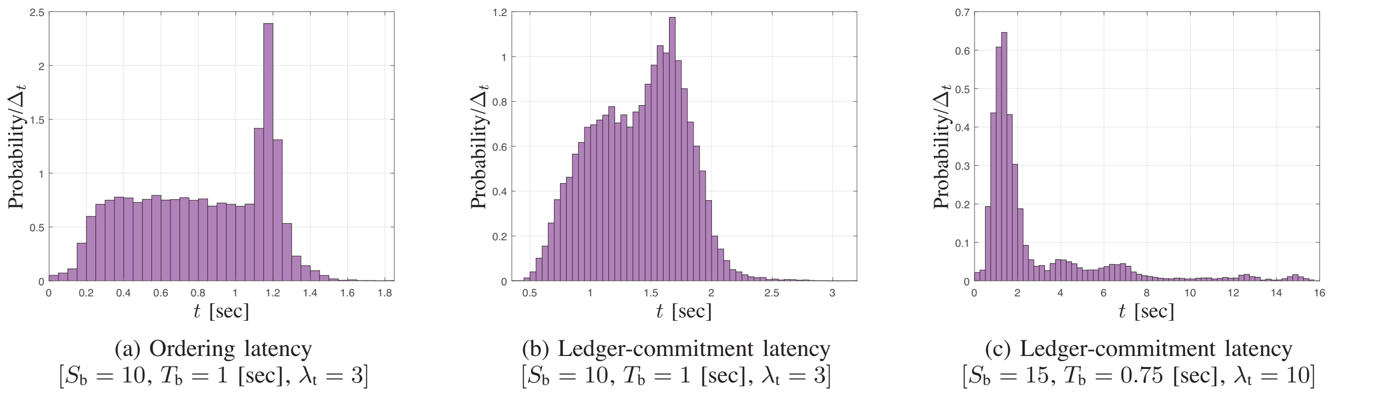

(a) Ordering latency $[S_\mathrm{b} = 10, T_\mathrm{b} = 1 \text{ [sec]}, \lambda_\mathrm{t} = 3]$ (b) Ledger-commitment latency $[S_\mathrm{b} = 10, T_\mathrm{b} = 1 \text{ [sec]}, \lambda_\mathrm{t} = 3]$ (c) Ledger-commitment latency $[S_\mathrm{b} = 15, T_\mathrm{b} = 0.75 \text{ [sec]}, \lambda_\mathrm{t} = 10]$

Fig. 4: Histograms of the ordering latency and the ledger-commitment latency T_t for infeasible latency modeling cases, where Figs. 4a and 4b are timeout-dominant and Fig. 4c is block size-dominant block generation environments, respectively.

are also generated fast with the full block size at the ordering phase. In this case, the ordering latency can be small, but there can be a large load at the validation phase due to the fast block generation, which makes blocks wait for validation. Hence, there can be many transactions with long waiting time for the validation, resulting in long-tail outliers longer than 4 [sec] on the histogram of T_t in Fig. 4c. This makes it difficult to fit the latency distribution into any known probability distributions.

Note that the two environments, discussed above, are the cases where S_b or T_b is too small or large, so a large portion of transactions experience long latency in the ordering or validation phase. For delay-sensitive HLF-enabled IoT, such HLF parameter setups will be generally avoided.

IV. Facing to Blockchain Latency: Influential Parameter Analysis on Latency

For HLF-enabled IoT handling time-sensitive data and requiring faster transaction commitment, it is important to design HLF in the way to minimize the average ledger-commitment latency. We can see in Fig. 5, the average ledger-commitment latency varies with different S_b, T_b, and λ_t. In this section, therefore, we explore their impacts on the average ledger-commitment latency. Moreover, we discuss how to optimally determine the three parameters to minimize the average ledger-commitment latency.

A. Average Transaction Generation Rate Control

Figure 5 (solid lines) demonstrates the impact of the average transaction generation rate λ_t for different S_b and T_b on $\overline{T}_\mathrm{t}$. Note that our ledger-commitment latency model can also be applied to high λ_t cases unless the modeling infeasible cases happen. From Fig. 5, we can observe two conflicting effects of λ_t on $\overline{T}_\mathrm{t}$. First, increasing λ_t makes transactions exported from the ordering service faster, so lower $\overline{T}_\mathrm{t}$ can be achieved. For instance, when S_b is 20 and T_b is 1 [sec] (blue line), $\overline{T}_\mathrm{t}$ decreases from 1.6 to 1.49 [sec] as λ_t increases from 10 to 12. However, $\overline{T}_\mathrm{t}$ starts to increase beyond some values of λ_t (e.g., for $\lambda_\mathrm{t} > 12$ in blue line) because of the fast block generation, which results in long waiting time for the validation to each block. In Fig. 5, we can also see that the optimal λ_t that minimizes $\overline{T}_\mathrm{t}$ increases for larger S_b or T_b (e.g., the optimal $\lambda_\mathrm{t} = 10$ with $S_\mathrm{b} = 10$, but the optimal $\lambda_\mathrm{t} = 17$ with $S_\mathrm{b} = 20$). Increasing the optimal λ_t with larger S_b or T_b is mainly to maintain or slowly change the block generation rate. Hence, the average transaction generation rate should be well-determined to minimize $\overline{T}_\mathrm{t}$.

Furthermore, in Fig. 5, we can also see the probability that the ledger-commitment latency is less than 3 [sec], i.e., $\mathbb{P}[T_\mathrm{t} < 3]$, (dotted line). This probability becomes the lowest when $\lambda_\mathrm{t} = 9$, different from $\lambda_\mathrm{t} = 10$ that minimizes $\overline{T}_\mathrm{t}$. Hence, for reliable IoT services, the latency distribution of HLF should be importantly considered to determine the HLF parameters, besides the average ledger-commitment latency.

B. Block Size and Block-generation Timeout Optimization

Figure 6 illustrates the impacts of the block size S_b (red lines) and block-generation timeout T_b (blue lines) on $\overline{T}_\mathrm{t}$ for empirical results and best-fit distributions. In Fig. 6, we can see that the average ledger-commitment latency of the best-fit distributions are well-matched with that of the empirical results. We can see that the shape of $\overline{T}_\mathrm{t}$ has three trends. Specifically, $\overline{T}_\mathrm{t}$ first decreases (i.e., decreasing area) and increases (i.e., increasing area), and finally saturates (i.e., saturating area). We explain the trend of those three areas as below.

1) Decreasing Area: First, $\overline{T}_\mathrm{t}$ decreases with the block size or block-generation timeout when the block size or the block-generation timeout are small or short (e.g., $S_\mathrm{b} < 8$ or $T_\mathrm{b} < 1.5$ [sec]). With small block size and short block-generation timeout, $\overline{T}_\mathrm{t}$ is generally high due to the fast block generation that gives large burden to the validation phase (i.e., step ⑧ in Fig. 1). Therefore, new blocks gradually stack up in each peer's queue for the validation, which leads to long validation delay. Hence, in this area, having larger block size or longer block-generation timeout helps to decrease $\overline{T}_\mathrm{t}$.

2) Increasing Area: When the block size or the block-generation timeout is relatively large (e.g., $10 < S_\mathrm{b} < 15$ or $2 < T_\mathrm{b} < 2.75$ [sec]), $\overline{T}_\mathrm{t}$ increases with the block size or the block-generation timeout. In this area, the validation latency is no longer large. However, transactions need to wait longer in the ordering phase to be in the form of a block as the block size or the block-generation timeout increases.

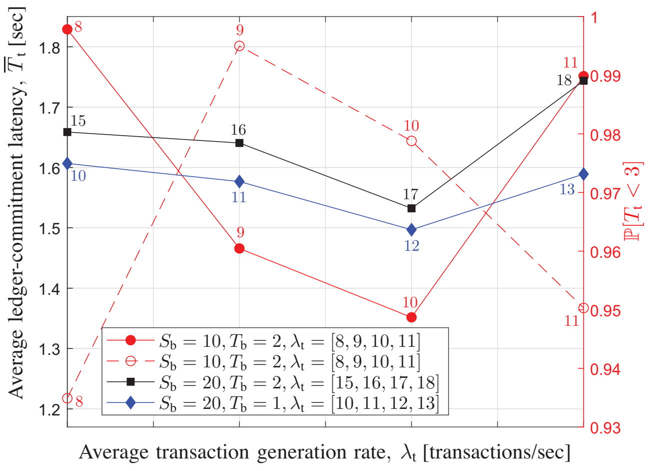


Fig. 5: Effects of the average transaction generation rate λ_t on the average ledger-commitment latency $\overline{T}_t$ (solid lines and left y-axis) for different values of the block size S_b and the block-generation timeout T_b, and on the probability that the ledger-commitment latency is less than 3 [sec], $\mathbb{P}[T_t < 3]$ (dotted line and right y-axis).

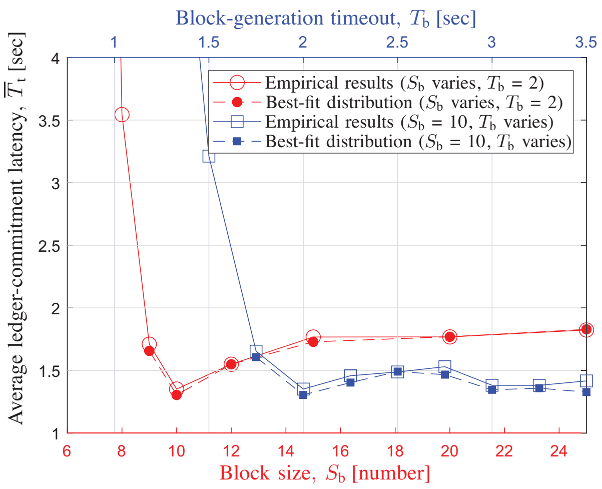


Fig. 6: Effects of the block size S_b and block-generation timeout T_b on the average ledger-commitment latency $\overline{T}_t$ for $\lambda_t = 10$ [transactions/sec], where the block size S_b varies and the block-generation timeout T_b is 2 [sec], and where the block size S_b is 10 and the block-generation timeout T_b varies, respectively.

3) Saturating Area: As the block size or the block-generation timeout keeps increasing (e.g., $S_b > 15$ or $T_b > 3$ [sec]), $\overline{T}_t$ eventually saturates. When the block size is large, most of new blocks are generated by timeout expiration before they are fully filled with S_b numbers of transactions. When the block-generation timeout is long, blocks are also generated with full size, and increasing the block-generation timeout no longer affects the latency. Therefore, $\overline{T}_t$ does not change in this area, even though the block size or the block-generation timeout increases.

Due to the trend on the impacts on the block size and the block-generation timeout on $\overline{T}_t$, it is important to properly determine S_b and T_b values to lower $\overline{T}_t$.

V. Conclusions

This article provides the latency model for HLF-enabled IoT, where time-sensitive data is generally managed. From the histograms of the ledger-commitment latency obtained from our real implementation of HLF, we figure out that the ledger-commitment latency can be modeled as the Gamma distribution in various HLF setups. We also conduct the KS test and verify that our modeling is reliable. Moreover, we explore the impacts of three important HLF parameters (i.e., the average transaction generation rate, block size, and block-generation timeout) on the average ledger-commitment latency. Specifically, when they are set to be small, the average ledger-commitment latency is fairly high due to the fast block generation that causes long validation latency. Hence, the average ledger-commitment latency decreases as those values increase, but eventually it increases again since the ordering latency increases. From those observations, it is shown that a proper setup of HLF parameters are important to lower the average ledger-commitment latency. This article can serve as a useful framework not only to predict the average and distribution of the ledger-commitment latency, but also to optimally design HLF-enabled IoT with low latency.

Acknowledgment

We appreciate Jia-Yin Guo (National Sun Yat-sen University) helping on the experiments with multiple physical machines for HLF latency modeling.

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.2020R1A2C2008878), in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF-2018R1A5A1060031), and in part by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2023-2020-0-01795) supervised by the IITP (Institute of Information & Communications Technology Planning & Evaluation). This work was also partially supported by the National Science and Technology Council of Taiwan under Grant 109-2221-E-110-041-MY3, Grant 111-2628-E-110-006-MY3, and Grant 111-2218-E-110-001-MBK, and in part by the Information Security Research Center, National Sun Yat-sen University, Taiwan.

References

[1] H.-N. Dai, Z. Zheng, and Y. Zhang, "Blockchain for Internet of Things: A survey," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8076–8094, Oct. 2019.
[2] M. Jo *et al.*, "Private blockchain in industrial IoT," *IEEE Netw.*, vol. 34, no. 5, pp. 76–77, Sep./Oct. 2020.
[3] J. Huang *et al.*, "Towards secure industrial IoT: Blockchain system with credit-based consensus mechanism," *IEEE Trans. Industr. Inform.*, vol. 15, no. 6, pp. 3680–3689, Jun. 2019.
[4] M. Du, Q. Chen, J. Chen, and X. Ma, "An optimized consortium blockchain for medical information sharing," *IEEE Trans. Eng. Manag.*, vol. 68, no. 6, pp. 1677–1689, Dec. 2021.

[5] T. Hewa, A. Braeken, M. Liyanage, and M. Ylianttila, "Fog computing and blockchain-based security service architecture for 5G industrial IoT-enabled cloud manufacturing," *IEEE Trans. Industr. Inform.*, vol. 18, no. 10, pp. 7174–7185, Oct. 2022.

[6] A. Dixit, A. Singh, Y. Rahulamathavan, and M. Rajarajan, "FAST DATA: A fair, secure and trusted decentralized IIoT data marketplace enabled by blockchain," *IEEE Internet Things J.*, Early Access, DOI: 10.1109/JIOT.2021.3120640.

[7] F. Jamil, S. Ahmad, N. Iqbal, and D.-H. Kim, "Towards a remote monitoring of patient vital signs based on IoT-based blockchain integrity management platforms in smart hospitals," *Sensors*, vol. 20, no. 8, p. 2195, Apr. 2020.

[8] S. Aggarwal, N. Kumar, M. Alhussein, and G. Muhammad, "Blockchain-based UAV path planning for Healthcare 4.0: Current challenges and the way ahead," *IEEE Netw.*, vol. 35, no. 1, pp. 20–29, Jan./Feb. 2021.

[9] S. Lee, M. Kim, J. Lee, R.-H. Hsu, and T. Q. S. Quek, "Is blockchain suitable for data freshness? An age-of-information perspective," *IEEE Netw.*, vol. 35, no. 2, pp. 96–103, Mar./Apr. 2021.

[10] H. Sukhwani, N. Wang, K. S. Trivedi, and A. Rindos, "Performance modeling of Hyperledger Fabric (permissioned blockchain network)," in *Proc. IEEE Int. Symp. Netw. Comput. Appl. (NCA)*, Cambridge, MA, USA, Nov. 2018, pp. 1–10.

[11] P. Yuan, K. Zheng, X. Xiong, K. Zhang, and L. Lei, "Performance modeling and analysis of a Hyperledger-based system using GSPN," *Comput. Commun.*, vol. 153, no. 1, pp. 117–124, Mar. 2020.

[12] L. Jiang, X. Chang, Y. Liu, J. Mišić, and V. B. Mišić, "Performance analysis of Hyperledger Fabric platform: A hierarchical model approach," *Peer Peer Netw. Appl.*, vol. 13, no. 3, pp. 1014–1025, May 2020.

[13] X. Xu *et al.*, "Latency performance modeling and analysis for Hyperledger Fabric blockchain network," *Inf. Process. Manag.*, vol. 58, no. 1, p. 102436, Jan. 2021.

[14] 3rd Generation Partnership Project, "GERAN improvements for machine-type communications (MTC)," 3GPP TR 43.868 V12.1.0, Tech. Rep., Dec. 2014.

[15] S. Shalaby *et al.*, "Performance evaluation of Hyperledger Fabric," in *Proc. IEEE Int. Conf. Inform., IoT, Enabling Technol. (ICIoT)*, Doha, Qatar, Feb. 2020, pp. 1–6.

Biographies

Sungho Lee [S'20] received his B.S. degree in computer education from Sungkyunkwan University (SKKU), Korea, in 2017. He is currently pursuing his Ph.D. degree with the Department of Electrical Engineering and Computer Science, Daegu Gyeongbuk Institute of Science and Technology (DGIST), Korea. His research interests include blockchain, wireless communications, and Internet of Things.

Minsu Kim [S'20] received his B.E. degree with honors in School of Undergraduate Studies and his M.S. degree in Information and Communication Engineering from Daegu Gyeongbuk Institute of Science and Technology (DGIST), Korea, in 2019 and 2021, respectively. He is currently a Ph.D. student at the Electrical and Computer Engineering Department at Virginia Tech, USA. He is awarded the best paper award at IEEE ICC 2022 and IEEE TAOS Technical Committee 2022. He is also the recipient of the Pratt Scholarship. His research interests include distributed machine learning, Internet of Things, and 6G networks.

Jemin Lee [S'06-M'11] received the B.S., M.S., and Ph.D. degrees in Electrical and Electronic Engineering from Yonsei University, Korea, in 2004, 2007, and 2010, respectively. She worked at the Information and Communication Engineering, Daegu Gyeongbuk Institute of Science and Technology (DGIST), Korea from 2016-2021. Currently, she is an Associate Professor at the Department of Electrical and Computer Engineering, Sungkyunkwan University (SKKU), Korea.

Ruei-Hau Hsu [M'15] received the Ph.D. degree in the Department of Computer Science and Engineering, National Sun Yat-sen University, Taiwan, in 2012. From 2017 to 2018, he was a Scientist at Agency for Science, Technology and Research, Singapore. Since August 2018, he is the Assistant Professor in the Department of Computer Science and Engineering, National Sun Yat-sen University, Taiwan.

Min-Soo Kim received his Ph.D. degree in Computer Science from KAIST, Korea, in 2006. He worked in Information and Communication Engineering at DGIST in Korea from 2011 to 2020. He is a full professor of School of Computing at KAIST, and his research interests include databases and machine learning.

Tony Q. S. Quek [S'98, M'08, SM'12, F'18] received the B.E. and M.E. degrees in electrical and electronics engineering from the Tokyo Institute of Technology in 1998 and 2000, respectively, and the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology in 2008. Currently, he is the Cheng Tsang Man Chair Professor, ST Engineering Distinguished Professor, and Head of ISTD Pillar with Singapore University of Technology and Design as well as the Director of Future Communications R&D Programme. He was honored with the 2008 Philip Yeo Prize for Outstanding Achievement in Research, the 2012 IEEE William R. Bennett Prize, the 2017 CTTC Early Achievement Award, the 2017 IEEE ComSoc AP Outstanding Paper Award, the 2020 IEEE Communications Society Young Author Best Paper Award, the 2020 IEEE Stephen O. Rice Prize, the 2020 Nokia Visiting Professor, and the 2022 IEEE Signal Processing Society Best Paper Award. He is a Fellow of IEEE and a Fellow of the Academy of Engineering Singapore.