electronics

MDPI

Article

Performance Analysis of Endorsement in Hyperledger Fabric Concerning Endorsement Policies

Xuefeng Piao [1], Hao Ding [1] and Huihui Song [2,*]

[1] School of Computer Science and Engineering, Harbin Institute of Technology, Weihai 264209, China
[2] School of New Energy, Harbin Institute of Technology, Weihai 264209, China
* Correspondence: songhh@hitwh.edu.cn

Abstract: As a typical representative of a permissioned blockchain system, Hyperledger Fabric has garnered substantial attention in recent years. Despite the application and promotion of Hyperledger Fabric in fields such as smart grid and smart healthcare, challenges persist with respect to its performance, especially its transaction latency. High latency always discourages Hyperledger Fabric from latency-sensitive applications. In this paper, we focus on the execution phase of the unique Execute-Order-Validate architecture of Hyperledger Fabric and conduct a comprehensive analysis of its endorsement policies. We summarize three theorems and give corresponding mathematical proofs based on the definition of the endorsement policy. To better analyze the generation of latency during the endorsement process, we further developed a theoretical model using queuing theory. Subsequently, we conducted multiple experiments on Hyperledger Fabric v2.0 for performance evaluation. With the experiment results, we discuss the reasons for the performance differences between different endorsement strategies and between equivalent logical expressions. Eventually, this paper offers some suggestions for endorsement policy selection which can provide a reference for the application of Hyperledger Fabric in the actual production environment.

Keywords: blockchain; latency performance; endorsement policy; Hyperledger Fabric

Citation: Piao, X.; Ding, H.; Song, H. Performance Analysis of Endorsement in Hyperledger Fabric Concerning Endorsement Policies. *Electronics* **2023**, *12*, 4322. https://doi.org/10.3390/electronics12204322

Academic Editors: Mehdi Sookhak and Rameez Asif

Received: 15 August 2023
Revised: 12 October 2023
Accepted: 17 October 2023
Published: 18 October 2023

1. Introduction

Blockchain, initially introduced alongside crypto currencies like Bitcoin [1], has risen to fame swiftly in recent years. Its remarkable capacity to establish trust among untrusted parties is a key driver, eliminating the need for a trusted third entity. The widespread application of blockchain technology has reshaped multiple industries both academically and industrially [2]. For most blockchain-enabled IoT applications, blockchain serves as a distributed and tamper-resistant ledger for storing IoT data. This effectively bridges the gap in the authenticity and trustworthiness of device data to a certain extent [3,4].

At present, an escalating number of blockchain frameworks or platforms like Ethereum, Hyperledger Fabric, Multichain, etc., have become available for development. This is largely attributed to the growing contributions from both corporations and open-source communities [5]. Among the numerous blockchain frameworks and platforms, Hyperledger Fabric is one of the most popular, non-profit and enterprise-oriented blockchain platforms of consortium blockchain maintained by the Linux Foundation since 2015 [6]. For Hyperledger Fabric, it delivers some key differentiating capabilities over other blockchain frameworks or platforms like modular and configurable architecture, pluggable consensus protocols, etc. In particular, arbitrary smart contracts authored in general-purpose programming languages can be developed and run on Hyperledger Fabric, which realizes the automation of multi-step processes [3].

Hyperledger Fabric is now being exploited for multiple IoT scenarios, ranging from healthcare [7] to smart grids [8]. Furthermore, for latency-sensitive scenarios, applications seeking to implement the corresponding functions on the basis of blockchain also need

to consider responsiveness in addition to security and privacy concerns [9,10]. However, Hyperledger Fabric tends to encounter high bandwidth overhead and latency during use, which prevents it from being applied to latency-sensitive scenarios.

For Hyperledger Fabric, a unique architecture called the Execute-Order-Validate (EOV) model is introduced for transaction processing and execution. The EOV model consists of the following three phases: the endorsement phase, ordering phase and validation phase. The endorsement policy mechanism plays an indispensable role in both phases, which specifies some network peers to confirm, validate and, finally, subsequently append the transaction to the ledger. In short, the endorsement policy is specified for a chaincode when it is approved and committed to the channel. From the perspective of the EOV model, the transaction latency can be seen as arising from those three phases, respectively. Moreover, the major origins of transaction latency can be regarded as the endorsement phase and ordering phase according to the conclusions presented by Thakkar et al. [11] and Kwon et al. [12].

In this paper, we dive into the exploration of endorsement policy in Hyperledger Fabric v2.0 and attempt to develop a comprehensive understanding of the trade-offs concerning the endorsement involved. The contributions of our work are listed as follows:

- We present three theorems and provide rigorous mathematical proofs for each of them in accordance with the definition of an endorsement policy in Hyperledger Fabric;
- We develop a latency analytical model focusing on the execution phase and theoretically explore the causes of latency based on the M/M/1 queue in queue theory and Little's law;
- We conduct multiple experiments under different endorsement policies on Hyperledger Fabric v2.0 and find performance differences between different endorsement strategies as well as between equivalent logical expressions;
- We discuss the impact of endorsement policies on the performance of a blockchain network, analyze the reasons for performance differences between different policies and offer some suggestions for policy selection.

The rest of this paper is organized as follows. The related work is surveyed in Section 2, followed by an overview of the EOV architecture in Section 3. Section 4 provides a detailed description concerning the endorsement policy of Hyperledger Fabric. And then Section 5 develops a theoretical model for the latency performance analysis of the endorsement process. Section 6 evaluates the performance of Hyperledger Fabric under different endorsement policies and provides a discussion. Finally, Section 7 concludes with remarks and future research directions.

2. Related Work

Despite its rising popularity, scalability remains a key challenge for blockchain technology. High scalability refers to the blockchain system's capacity to efficiently manage and process a substantial volume of transactions and data in order to satisfy the requirements of its users. However, from the point of view of performance metrics, high scalability usually requires a system with high throughput and low latency [13–15].

The magnitude of throughput is an important indicator of a blockchain system's ability to process transactions and many researchers have made great contributions to improving the throughput of Hyperledger Fabric. Gorenflo et al. [16] make architectural changes to Hyperledger Fabric, thus reducing computation and I/O overhead during transaction ordering and validation. And the changes are fully plug-and-play. Nakaike et al. [17] analyze the database systems used in Hyperledger Fabric and draw a conclusion that performance bottleneck arises from accesses to the databases that store the latest key-value pairs in the ledger data, indexes to transactions and the update history. Berendea et al. [18] introduce a novel gossip design for Fabric that achieves the simultaneous optimization of the propagation time, tail latency and bandwidth consumption. Such improvement also performs well on high throughput and concurrent applications. Hang et al. [19] propose a transaction traffic control approach based on fuzzy logic and implement it in

a smart contract. The blockchain performance is significantly improved by this control approach. Sharma et al. [20] explore Fabric from the perspective of database research and try to migrate certain concepts and technology from the database into blockchain. Their experiments demonstrate that this introduction improves performance by nearly a factor of three.

In IoT applications, high latency stands out as a critical challenge. Presently, the primary objective in blockchain optimization is the minimization of latency. Endorsement policy, identified as one of three major performance bottlenecks of Hyperledger Fabric [11,21], has attracted considerable interest in recent years. Thakkar et al. [11] propose to introduce aggressive caching or parallelization technology for endorsement policy validation, which effectively shortens the time required to verify the endorsement signature of a transaction. In particular, Kwon et al. [12] achieve a substantial improvement in original read transaction processing by distinguishing between read and write transactions in the endorsement phase. Liu et al. [22] then consider how to make the distribution of endorsement tasks more balanced for the endorsement phase of peer nodes in order to improve the performance of a blockchain system in the case of high concurrency. Shammar [23] et al. investigate the role endorsement plays in Hyperledger Fabric's consensus and involve it in their proposed control model, overcoming traditional centralized access control issues to a certain extent. Qin et al.'s study [24] is also concerned with the significance of endorsement policy in designing a control scheme and make dynamical modifications to the endorsement protocol according to their proposed user credibility incentive mechanism. The modifications work well for the protocol. Wang et al. [25] acquire, through their experimental results, proof that Hyperledger Fabric exhibits good scalability in the execution phase when the `OR` endorsement policy is employed, but not when the `AND` endorsement policy is employed. Soelman et al. [26] start from the common needs and requirements of blockchain networks to study the impact of different endorsement policies on the data authenticity of a ledger and give the corresponding improvement measures.

3. EOV Architecture of Hyperledger Fabric

3.1. Key Concepts

As an open-source blockchain platform designed for enterprise users, Hyperledger Fabric provides an exceptional modular and extensible architecture that facilitates its functionality and versatility. Here are several key concepts [27] that form the foundation of Hyperledger Fabric, as shown in Figure 1. As a whole, a fabric network consists of four modules, called Membership, Blockchain, Transactions and Chaincode, and the four modules are connected by the event stream. For users and developers, they can easily use or develop Hyperledger Fabric using the command line interface (CLI), software development kit (SDK) and application programming interface (API).

- Ledger
  Ledger refers to a tamper-resistant, distributed database that stores important factual information concerning business in a blockchain network, including both the current value of the attributes of the objects and the history of transactions that resulted in these current values.
- Node
  In Hyperledger Fabric, nodes play a crucial role as communication entities of the blockchain network. They are distributed across the network and collaborate to achieve consensus on the validity of transactions and maintain the shared ledger. In Hyperledger Fabric v2.x, the types of nodes can be broadly classified into two categories, i.e., peer nodes and orderer nodes. In addition, although all peer nodes are identical in Fabric, they can assume multiple roles depending on the network configuration and functional requirements. These roles include `Committer`, `Leader`, `Endorser` and `Anchor`. Figure 2 below shows the connection between different nodes [28]. Thus, the transaction process in Fabric can be viewed as a process of interaction among multiple nodes.

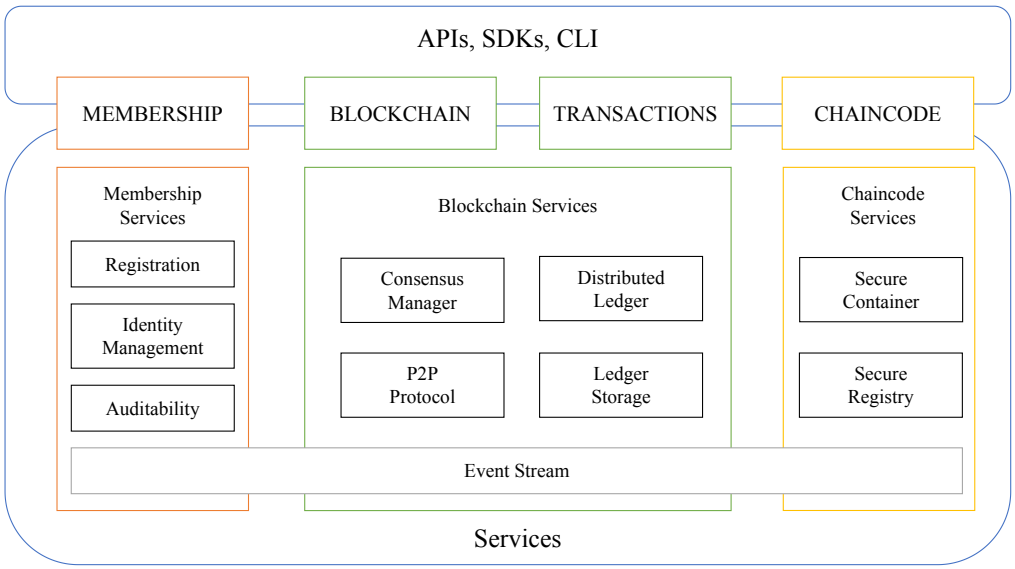

Figure 1. Key components in Fabric network.

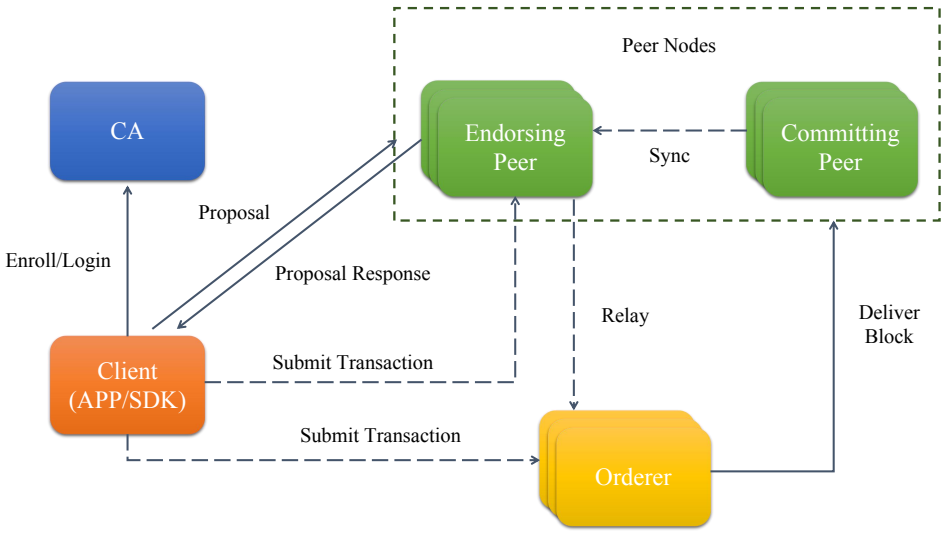

Figure 2. Connection between different nodes.

- Membership Service Providers (MSP)
  An MSP is a Hyperledger Fabric component that offers an abstraction of membership operations [29]. In particular, an MSP abstracts away all cryptographic mechanisms and protocols behind issuing certificates, validating certificates, and user authentication. In other words, an MSP manages the identities and access control of participants in the blockchain network. It is responsible for the authentication and authorization of interacting entities within the network like organizations, peers and clients.
- Channel
  In a fabric network, there can be multiple channels and each channel is isolated from each other. Participants in the same channel have a separate ledger and maintain it together. In addition, each transaction on the network is executed on a channel, where each party must be authenticated and authorized to transact on the corresponding channel using an MSP.
- Smart Contract/Chaincode
  A smart contract is an agreement that automatically executes relevant contract clauses based on business logic set by pre-written code in the context of a blockchain. In

Fabric, a smart contract is also called a chaincode. It specifies how data should be validated, processed and updated on the ledger [30,31].

3.2. Transaction Flow

Transactions flow through the various components of the Fabric network and require the collaborative efforts of the nodes in the network, as illustrated in Figure 3. The EOV process starts when the transaction is proposed and ends when the transaction is submitted to the ledger [6]. First of all, the client creates a transaction and sends it to a number of endorsing peers. Once receiving the transaction sent by the client, each endorsing peer proceeds to execute the transaction against its own state database, ultimately computing the read–write set specific to that transaction. If the transaction is executed without errors, it signifies that the committed transaction has successfully passed the business rule checks. As a consequence, the endorsing peer will provide its endorsement, signifying its approval to the client. Later, the client waits for a sufficient number of endorsements and then sends the transaction with its endorsement to the orderer nodes that implement the ordering service. The ordering service returns a response to the client after the transaction has been accepted for inclusion in the block. The orderer nodes will reach a consensus on the order of incoming transactions and subsequently create individual blocks in succession from the ordered transactions based on the `BatchSize` and `BatchTimeout` parameters configured by the Fabric network. Once a block is created, the orderer nodes broadcast it to all peers who subsequently proceed to carry out the following procedures. The peers then validate all transactions within the block and commit the block to the ledger and state database once it has passed. In addition, one of the peers will notify the client that the transaction has been committed.

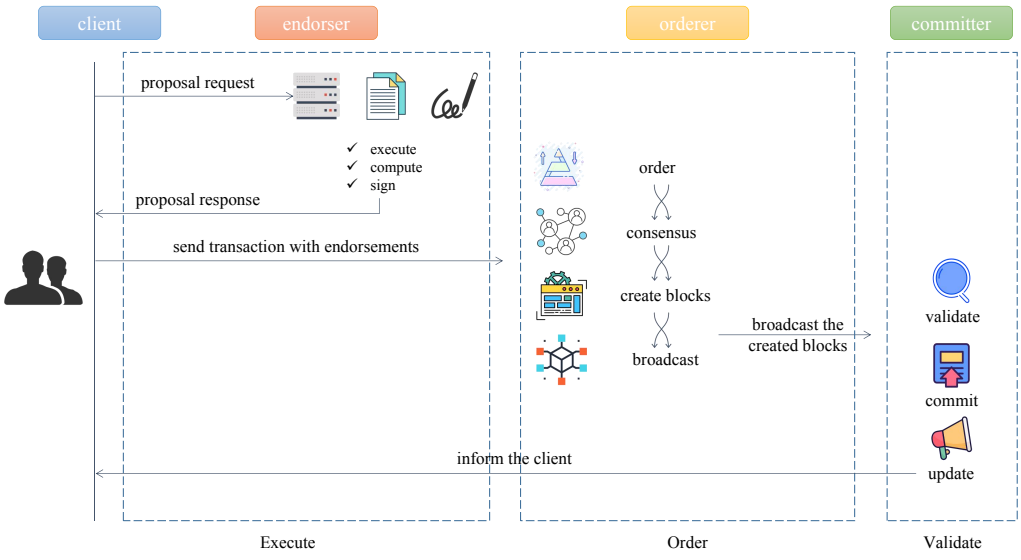

Figure 3. Transaction flow in Fabric.

3.3. Endorsement Flow in Fabric

In order to have a closer look at the endorsement phase, we further explore the implementation details of the endorsement flow in Fabric. To begin, once a transaction proposal is generated by the client, the client sends it to the endorsing peers along with its own signature for execution. Subsequently, the client awaits the receipt of a proposal response from the endorsing peers, which includes the simulation results and the endorsement signatures. As for the endorsing peers, when they receive the transaction proposal from the client, they utilize an MSP to validate and authenticate the identity of the client. Later, a chaincode is executed for simulation in an isolated space within the container, which is also called a sandbox. Notice that each endorsing peer executes the same transaction simulation

independently and eventually generates an endorsement result for that transaction in the execution phase. Through simulation, the endorsing peers get read–write sets of transactions. The read set is a collection of key-value pairs that the transaction needs to read from the ledger while the write set is a collection of key-value pairs that the transaction intends to write or update to the ledger. Next, the read set will be validated and the write set will be executed. If no error occurs at this point, the endorsing peers will provide the endorsement signatures, attach them to the transaction proposal and send the endorsement signatures back to the client. Once the client collects a sufficient number of endorsement signatures, it proceeds to commit the transaction in the subsequent ordering phase. The pseudocode, as shown in Algorithm 1, of how endorsement is implemented in Hyperledger Fabric's open-source code can help us better understand the process.

Algorithm 1: Endorsement phase in Fabric

```
begin
    initialize;
    check and validate the transaction proposal;
    if the transaction proposal is valid then
        execute chaincode for simulation;
        acquire the read-write set concerning the transaction;
        validate the read set;
        execute the write set;
        if no error then
            get execution results;
            perform endorsement signing;
            send the endorsement results back to the client;
        end
    end
    if the endorsement policy is satisfied then
        client sends transaction with endorsements to orderer nodes;
        proceed to the subsequent ordering phase;
    end
end
```

4. Endorsement Policy of Hyperledger Fabric

The latency incurred by the Fabric network during the execution phase mainly consists of two components: the network latency of the communication between client and endorsing peers and the processing latency of each endorsing peer [32]. During the whole endorsement process, the number of peers required to endorse a transaction is driven by the corresponding endorsement policy that is specified in the Endorsement System Chaincode (ESCC). As a result, the choice of endorsement policy exerts an influence on the transaction latency of the fabric network during the endorsement process. A proper endorsement policy is critical for improving network performance.

Fabric offers a domain-specific language with which it is possible to define endorsement policies conveniently and flexibly. Principals are the component used to identify entities or roles in the endorsement process, defining who is authorized to endorse transactions. Principals can be described as `MSP.ROLE`, where `MSP` represents the required MSP ID and `ROLE` represents one of the four accepted roles: member, admin, client and peer.

Furthermore, this language provides three basic logic operators, i.e., `AND`, `OR` and `k-OutOf`. Without a loss of generality, let us denote the total number of endorsing peers or organizations in a set specified by the corresponding endorsement policy as N for a better description in the following. For expression `AND`, the transaction proposal is supposed to get the signatures of all the specified N endorsers. And for expression `OR`, the transaction proposal needs to acquire the signature of any of the specified N endorsers, while for expression `k-OutOf`, the transaction proposal has to obtain the signatures of at least k

endorsers from the specified N endorsers. Moreover, nested combinations of the above three basic logical operators are also supported by Hyperledger Fabric.

The syntax of the endorsement policy language is `EXPR(E[, E...])`, where `EXPR` is either `AND`, `OR` or `OutOf`, and `E` is either a principal (with the syntax described above) or another nested call to `EXPR`. Equations (1)–(4) give examples of different endorsement policies. Thus, by setting different endorsement policies, a specified number of peers on a channel can be assigned to execute the chaincode and endorse the corresponding results.

Example 1 (the `AND` endorsement policy)**.**

$$AND('Org1MSP.member','Org2MSP.member','Org3MSP.member') \tag{1}$$

Example 2 (the `OR` endorsement policy)**.**

$$OR('Org1MSP.member','Org2MSP.member') \tag{2}$$

Example 3 (the `OutOf` endorsement policy)**.**

$$OutOf(2,'Org1MSP.member','Org2.member','Org3.member') \tag{3}$$

Example 4 (the nested endorsement policy)**.**

$$OR('Org1MSP.member', AND('Org2MSP.member','Org3MSP.member')) \tag{4}$$

Although there are many forms of endorsement policies that we can combine using the standard syntax and operators [27] like Equations (1)–(4), we come to some common conclusions, as in Theorems 1–3 below. Based on these three theorems, we can transform the expression of an endorsement policy freely between different operators, as in Equations (5) and (6), to meet the actual needs.

Theorem 1. *An endorsement policy that contains only the operator `OR` and N endorsers is logically equivalent to an endorsement policy that contains only the operator `1-OutOf`.*

Proof of Theorem 1. Let us denote the set of endorsing peers or organizations as S, and let $|S|$ represent the cardinality of S, i.e., the number of endorsing peers or organizations in set S. For `OR` policy, the transaction needs to receive an endorsement signature from any one of the specified peers or organizations in S, while for the `1-OutOf` policy, the transaction must obtain at least one signature from the specified peers or organizations in S. In the case where the `OR` policy is satisfied, the transaction receives a signature from any one peer or organization in S. At this point, the condition of the `1-OutOf` policy can also be satisfied. In the case where the `1-OutOf` policy is satisfied, the transaction receives at least one signature from the peers or organizations in S. At this point, the condition of the `OR` policy can also be satisfied. Therefore, the policies `OR` and `1-OutOf` are equivalent for any number of endorsing peers or organizations. □

Theorem 2. *An endorsement policy that contains only the operator `AND` and N endorsers is logically equivalent to an endorsement policy that contains only the operator `N-OutOf`.*

Proof of Theorem 2. Like the proof of Theorem 1 above, let us predefine set S and its cardinality $|S|$. For the `AND` policy, the transaction needs to receive endorsement signatures from all endorsing peers or organizations in S, while for the `N-OutOf` policy, the transaction must obtain at least N signatures from the specified peers or organizations in S. In the case where the `AND` policy is satisfied, the transaction receives N signatures from all peers or organizations in S. At this point, the condition of the `N-OutOf` policy can also be satisfied. In the case where the `N-OutOf` policy is satisfied, the transaction receives at

least N signatures from the peers or organizations in S. The total number of signatures is N, due to $|S| \leq N$. At this point, the condition of the `AND` policy can also be satisfied. Therefore, the policy `AND` and `N-OutOf` are equivalent for any number of endorsing peers or organizations. □

Theorem 3. *An endorsement policy that contains the operator* `k-OutOf` *can be converted into an endorsement policy consisting of a combination of the operator* `AND` *and* `OR`*.*

Proof of Theorem 3. Similar to the previous operation, let us predefine set S and its cardinality $|S|$. For the case where the `k-OutOf` policy is satisfied, then the transaction obtains at least k endorsement signatures from the peers or organizations in S. Notice that there are C_N^k schemes for the elements set $c_i (c_i \subset S, 1 \leq i \leq C_N^k)$ that provide the signatures. Then, for a particular c_i, the corresponding endorsement can be expressed as the form of the `AND` operation on all elements of the set c_i. After that, the result of the internal `AND` operation with each set c_i is performed on the `OR` operation. At this point, it is sufficient for at least any k elements to provide endorsement signatures to make the expression hold after the transformation. In this way, we convert the original `k-OutOf` endorsement by processing the elements inside each set with the `AND` operator and then combining their results using the `OR` operator. □

Example 5.

$$\begin{aligned} &OutOf(2,'Org1.member','Org2.member') \Leftrightarrow \\ &AND('Org1MSP.member','Org2MSP.member') \end{aligned} \tag{5}$$

Example 6.

$$\begin{aligned} &OutOf(2,'Org1.member','Org2.member','Org3.member') \Leftrightarrow \\ &OR(AND('Org1.member','Org2.member'), \\ &AND('Org1.member','Org3.member'), AND('Org2.member','Org3.member')) \end{aligned} \tag{6}$$

5. Latency Performance Analysis using Theoretical Modeling for Endorsement Process

From a macro perspective, transaction latency is a network-wide view of the amount of time taken for a transaction's effect to be usable across the network [6], while from a perspective of the EOV transaction flow, the transaction latency can be divided into three parts corresponding to three phases, i.e., the latency of the execution phase, latency of the ordering phase and latency of the validation phase. The latency of the execution phase or endorsement phase accounts for a significant proportion of the overall transaction latency [33]. On the one hand, the endorsing peers cause latency when simulating the execution of a chaincode in the docker containers. On the other hand, the client needs to wait for a sufficient number of endorsement signatures to satisfy the endorsement policy, which also incurs latency.

In the following, we attempt to theoretically model the latency generated by the execution phase in the Hyperledger Fabric system with the reference of the analytical latency model proposed by Xu et al. [34].

Initially, the client sends the transaction proposal to the endorsing peers, requesting them to provide endorsement signatures. The time it takes for a proposal to propagate through a network is related to the size of the packet being propagated, the bandwidth of the network and the average propagation latency of the network. It is assumed that the packet size of transaction proposals is α bits, that the bottleneck bandwidth of the network path is $\mathcal{B}$ bps and the average propagation latency of a network is $\mathcal{D}$. Therefore,

the communication latency of transaction proposals from client to endorsing peers T_1 is Equation (7).

$$T_1 = \alpha/\mathcal{B} + \mathcal{D} \tag{7}$$

Then, the endorsing peers simulate the execution of a transaction by ESCC and the execution process of the endorsement peers incurs latency. We further assume that the arrival of new transactions follows the Poisson Point Process with the arrival rate λ. So the arrival rate of data is $\lambda\alpha$ bit/s. And we define the nodes dealing with transactions in a first-in-first-out manner with the unlimited waiting rooms and the service time at each node follows the exponential distribution with mean μ. Additionally, the amount of data that each node can process per second can also be denoted as μ. Hence, we model the execution of transaction proposals as an M/M/1 queue, where arrivals are determined by a Poisson process and job service times have an exponential distribution. For the queue, it will process the transactions in order of arrival. Therefore, the traffic intensity ρ can be calculated as Equation (8). According to Little's law [35], the expected time T_2 for each endorsing peer to execute a transaction is Equation (9).

$$\rho = \frac{\lambda\alpha}{\mu} \tag{8}$$

$$T_2 = \frac{1}{\mu - \lambda\alpha} = \frac{1}{\mu(1-\rho)} \tag{9}$$

Eventually, the endorsing peers are supposed to return the endorsements to the client after completing the simulated execution of a transaction. Let us denote the packet size of an endorsement as β. Therefore, the communication latency of endorsements from endorsing peers to client T_3 is Equation (10).

$$T_3 = \beta/\mathcal{B} + \mathcal{D} \tag{10}$$

In summary, the overall latency in the endorsement process T_{endor} can be calculated using Equation (11) below. The latency of the endorsement process arises and grows little by little in the cycle of sending the transaction proposals, waiting for the transactions to be executed and returning the endorsement signatures.

$$T_{endor} = T_1 + T_2 + T_3 = \frac{\alpha}{\mathcal{B}} + \mathcal{D} + \frac{1}{\mu(1-\rho)} + \frac{\beta}{\mathcal{B}} + \mathcal{D} = \frac{\alpha+\beta}{\mathcal{B}} + \frac{1}{\mu(1-\rho)} + 2\mathcal{D} \tag{11}$$

6. Performance Evaluation and Analysis

The Hyperledger Fabric v2.0.1 is evaluated in our experiments. We deploy a 1-channel fabric network on four physical servers in an isolated LAN with a 1000 MB/s network bandwidth. The hardware configuration of each server is a 4-Core CPU, 16 GB main memory and 1TB magnetic disk; the Ubuntu Server 20.04LTS 64bit operating system is installed on each server. And four organizations are deployed on the servers with two peer nodes and one orderer node, respectively, as the topological structure shown in Figure 4. One organization is run on each physical server using docker containers. Four organizations, R1, R2, R3 and R4, build a network on channel C1 with the channel configuration CC1. Taking organization R1 as an example, peer nodes P1, P2 and an orderer node O1 belong to it. Peer nodes are primarily responsible for providing endorsement and broadcasting transactions, while orderer nodes are mainly responsible for managing the system channel and facilitating consensus mechanisms. In addition, organization R1 also hosts a copy of the distributed ledger L1 and smart contract S1. The remaining three organizations, R2, R3 and R4, are similar to those of organization R1.

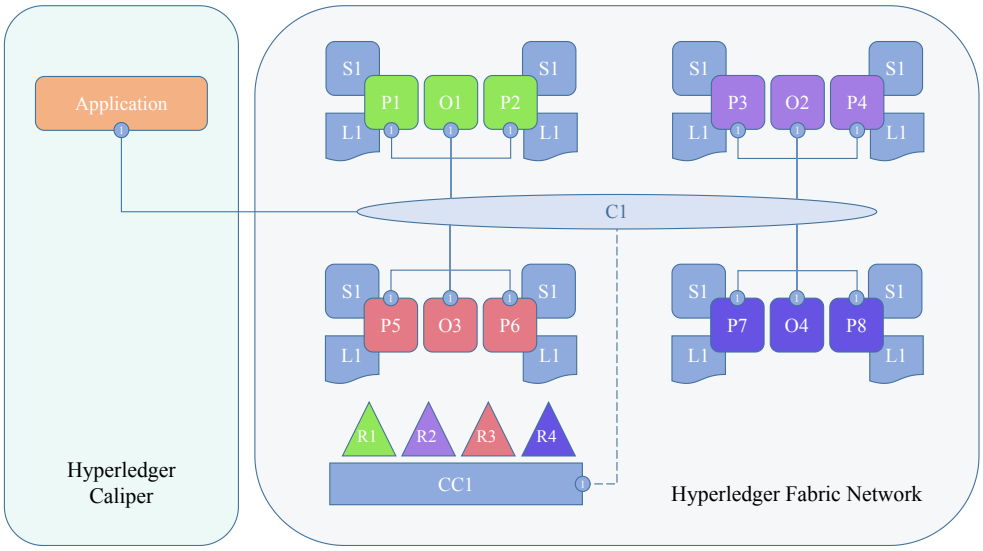

Figure 4. Topology of our Fabric network.

In terms of network configuration, the `BatchSize` is 100 and `BatchTimeout` is 1 s. The `BatchSize` is the maximum number of transactions a block can contain and the `BatchTimeout` is the maximum time elapsed for the duration of creating a block. In addition, Hyperledger Caliper [36], an open-source benchmarking tool for blockchain performance evaluation, is employed to measure the performance of a blockchain network against several predefined use cases. Our experiments are performed with different endorsement policies by sending transaction requests ranging from 20 Transactions Per Second(TPS) to 220 TPS with a 20 TPS interval. Table 1 provides a detailed reference of our experimental settings.

Table 1. Experimental Settings.

Parameter	Value	Unit
Number of Organizations	4	/
Number of Peer Nodes	2	per organization
Number of Orderer Nodes	1	per organization
`BatchSize`	100	/
`BatchTimeout`	1	second
Sending Rate	20/40/60/80/100/120/140/ 160/180/200/220/240	TPS

6.1. Experiment I: Impact of Parameter k in `k-OutOf` Policy

In the `k-OutOf` policy, the parameter k can be regarded as a variable with the range of any integer from 1 to N. The different values of k indicate the minimum number of endorsement signatures required in the endorsement policy. In order to explore the impact of different `OutOf` policies varying k on the transaction latency of a blockchain network, we test the average latency of four cases, `1-OutOf`, `2-OutOf`, `3-OutOf` and `4-OutOf`, as depicted in Figure 5. It can be found that, when the sending rate is at a relatively low value, the average latency of the four strategies is close with little difference. As can be seen from the figure, when the sending rate is at a relatively low value, from 20 TPS to 140 TPS, the average latency of the four policies is close with little difference. However, from 140 TPS onwards, the latency gap between different strategies has become noticeable and has been getting bigger. Overall, however, there is a trend that the larger the k, the higher the average latency at the same sending rate. That is, the larger the value of k becomes, the more endorsement signatures are required by the `OutOf` strategy, which means that it takes

longer to collect enough signatures to complete the endorsement process. In other words, an increase in the value of parameter k leads to an increase in the number of endorsing nodes that need to provide endorsement signatures. Accordingly, the communicating latency $T3$ in our proposed model becomes larger and thus the overall transaction latency becomes larger.

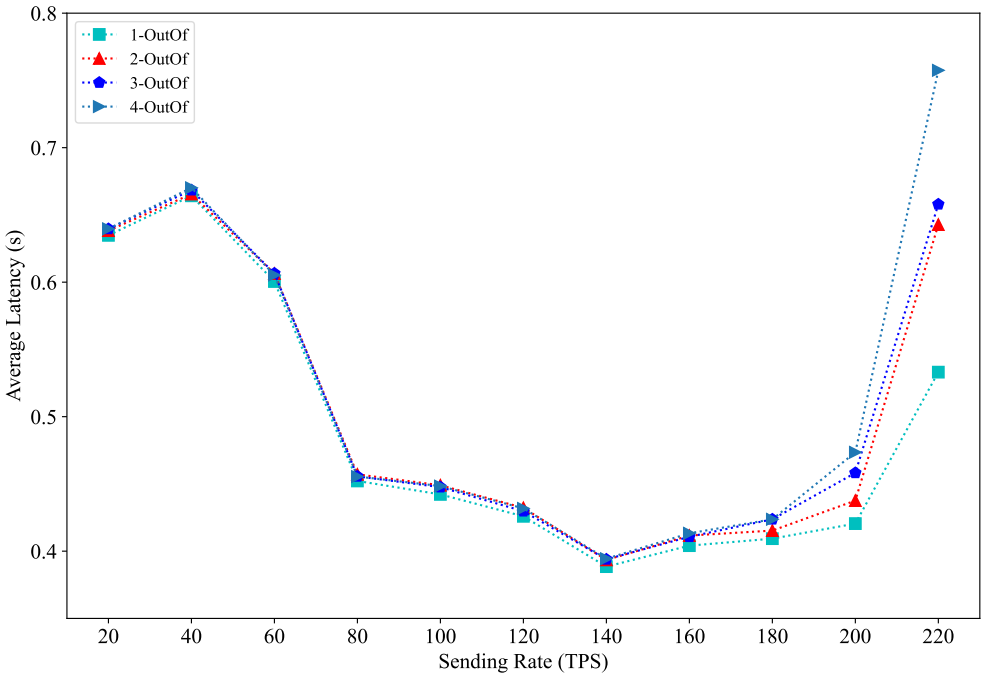

Figure 5. Average latency varying `OutOf` polices.

6.2. Experiment II: Comparison between `OR` Policy and `AND` Policy

Due to their logical simplicity and ability to cover most application scenarios, `OR` and `AND` are two of the most commonly used endorsement policies. In this paper, we also compare the performance of a blockchain network under these two policies, as shown in Figure 6. From an overall point of view, the `AND` policy has a higher latency than the `OR` policy at different sending rates. In particular, we can observe from the figure that from 200 TPS, the average latency of the `AND` policy shows a substantial increase compared to the average latency of the `OR` policy. When the sending rate reaches 220 TPS, the average latency of the `AND` policy even reaches nearly five times that of the average latency of the `OR` policy. At this point, the Fabric network has reached a performance bottleneck for the `AND` policy and the transaction latency will increase rapidly. The definition of the `AND` policy determines that it usually needs to involve more endorsing peers compared to the `OR` policy, which results in the need to communicate with more peer nodes. The network overhead of interacting with more nodes has an impact on the performance of Fabric and increases the latency of the blockchain system. However, the `AND` policy, which requires all the endorsing nodes to provide signatures, also provides system security, to some extent, compared to other endorsement policies.

6.3. Experiment III: Latency Comparison of Simple Equivalent Expressions

According to Theorem 1 and Theorem 2 above, the `OR` policy and `AND` policy can be equivalently transformed into a `1-OutOf` policy and `4-OutOf` policy, respectively, without adding complexity to the expression. Figures 7 and 8 represent the latency comparison of simple equivalent expressions. In terms of the comparison between the `OR` policy and the `1-OutOf` policy, the performance of the `1-OutOf` policy is slightly better than that of `OR` when the sending rate is less than 140 TPS. At the point where the sending rate receives 160 TPS, the difference between the two becomes very small. However, as the sending rate increases

further, the performance of the `OR` policy outperforms the performance of the `1-OutOf` policy. The performance gap between the two becomes more and more obvious and the `1-OutOf` policy will reach the performance bottleneck earlier than the `OR` policy. The `OR` policy only needs to communicate with any one of the nodes making the endorsement, while the `1-OutOf` policy needs to communicate with at least one node. This causes a relatively large communication latency when TPS is high.

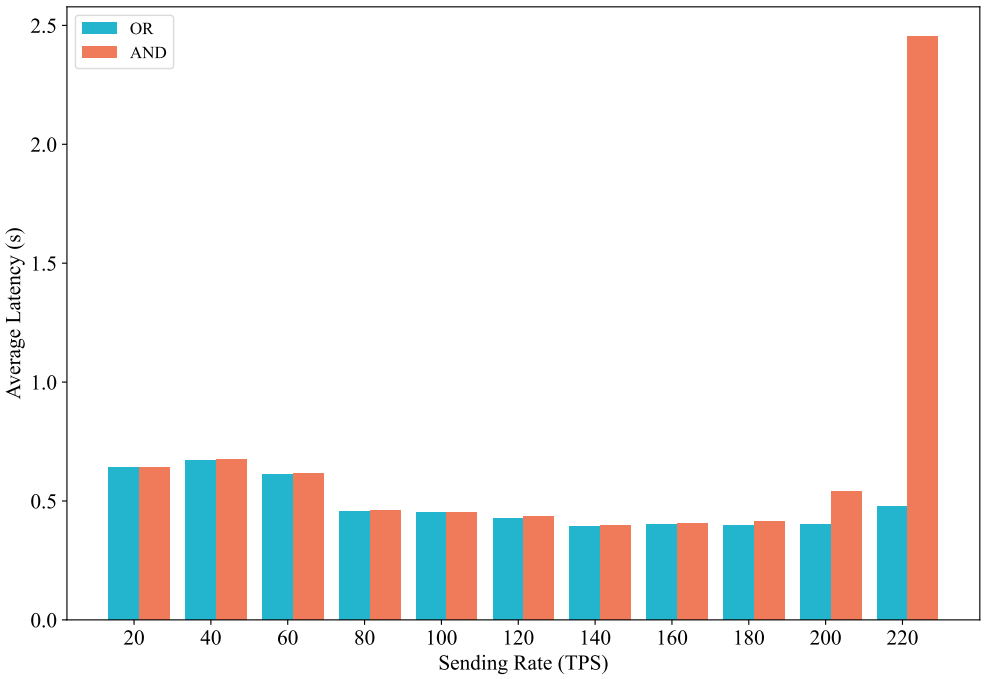

Figure 6. Average latency between `OR` policy and `AND` policy.

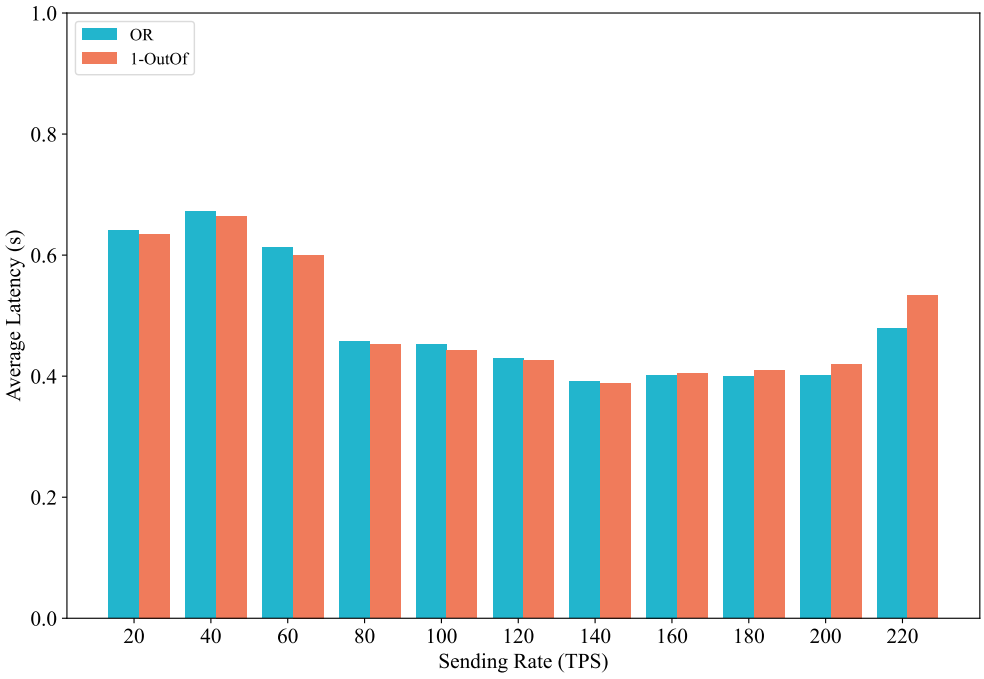

Figure 7. Average latency between `OR` policy and `1-OutOf` policy.

In terms of the comparison between the `AND` policy and the `4-OutOf` policy, 140 TPS can be viewed as the dividing line. On the whole, the performance of the `4-OutOf` policy

will be better than the performance of the `AND` policy in terms of latency. However, when the sending rate is less than or equal to 140 TPS, the performance difference between the two policies is relatively small and the performance is very close. While the sending rate exceeds 140 TPS, the latency of the `AND` policy grows rapidly and its average latency exceeds 2 s at 220 TPS, reaching the performance bottleneck. At this time, the `4-OutOf` policy can still keep the average latency of the system within 1 s, and has not yet reached the bottleneck.

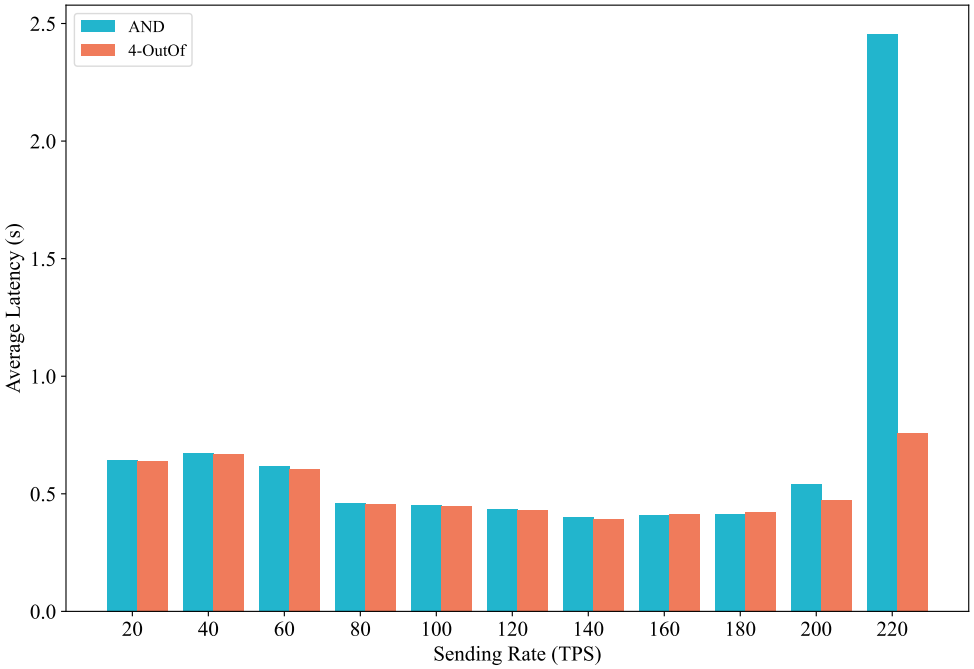

Figure 8. Average latency between `AND` policy and `4-OutOf` policy.

Overall, the `k-OutOf` policy performs better than its simple equivalent expression when the sending rate is kept at a low level. And this changes somewhat as the sending rate increases and needs to be analyzed on a case-by-case basis.

6.4. Experiment IV: Latency Comparison of Complex Equivalent Expressions

As Theorem 3 suggests in its finding, an endorsement policy expression can be transformed into an expression containing only the `AND` operator and the `OR` operator. This equivalent transformation will not change the meaning of the expression, but it will complicate it to some extent. Equation (12) is an equivalent expression of the `2-OutOf-4` policy. Therefore, we test the performance of the `2-OutOf` policy and its equivalent expression as shown in Figure 9. As can be seen from the figure, the `2-OutOf` policy and its equivalent expression maintain the same trend in the average latency under different sending rates. However, the average latency of the equivalent expression is consistently higher than that of the original expression. And the latency gap between the two expressions is also consistently increasing as the sending rate increases. The equivalent transformation of the endorsement policy leads to many more sub-expressions appearing in the expression, that is, it leads to the complication of the endorsement policy. Complexity makes it necessary for the endorsing peers to execute more computation and communication to validate the transaction, causing the endorsement process to become more time-consuming.

$$\begin{aligned} &OutOf(2,'Org1.member','Org2.member','Org3.member','Org4.member') \Leftrightarrow \\ &OR(AND('Org1.member','Org2.member'), AND('Org1.member','Org3.member'), \\ &AND('Org1.member','Org4.member'), AND('Org2.member','Org3.member'), \\ &AND('Org2.member','Org4.member'), AND('Org3.member','Org4.member')) \end{aligned} \tag{12}$$

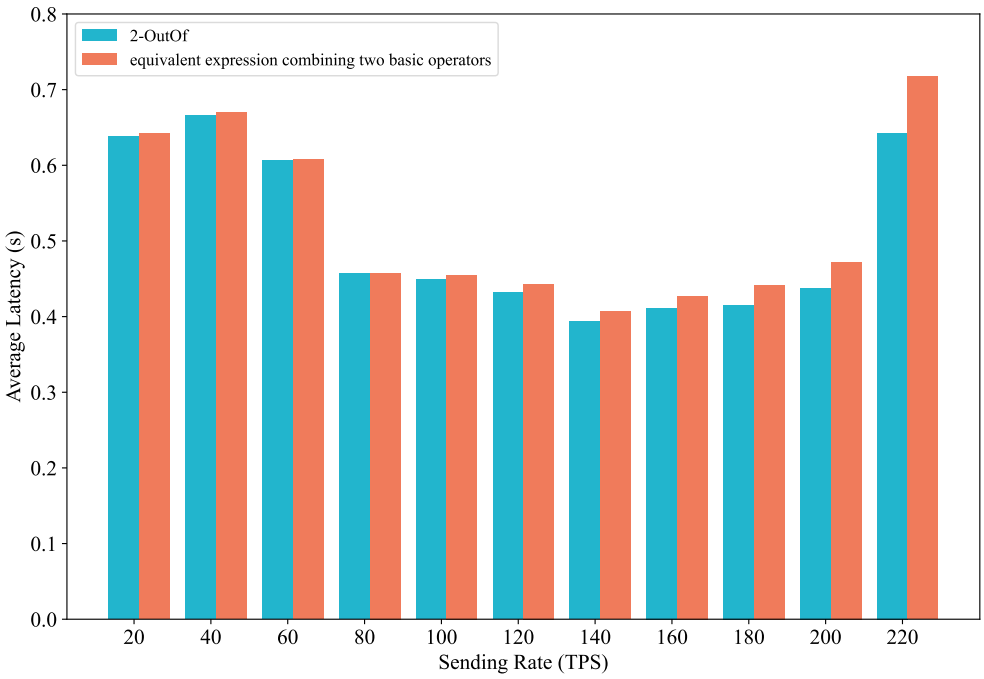

Figure 9. Average latency between `2-OutOf` policy and its corresponding endorsement policy.

When we look at Figures 5–9 together, we see that, as the sending rate increases, the overall average latency appears to increase and then decrease, followed by a gradual leveling off, and then finally a sudden increase. This trend of variation can be attributed to the settings of the parameters, `BatchSize` and `BatchTimeout`. When the sending rate is low, the number of transactions processed within the `BatchTimeout` does not reach `BatchSize`. At this point, `BatchTimeout` is a major factor in the creation of a block. But when the send rate is higher, blocks can be created based on `BatchSize` instead of `BatchTimeout`. And the higher the sending rate, the faster the blocks are created. Additionally, the latency fluctuation is relatively stable over a range of high sending rates, from 80 TPS to 180 TPS in our experiments. This is because the time taken to create a block within `BatchTimeout` is relatively constant. And the final steep increase is due to the system reaching a performance bottleneck.

Taking the four experiments we conducted together, we can summarise the following conclusions for reference when configuring an endorsement policy for the Fabric network. When choosing a `k-OutOf` endorsement policy, it is necessary to reduce the value of k as low as possible. An increase in the value of k leads to a degradation of the performance of a network, which is not conducive to application in latency-sensitive scenarios. In the case of the `OR` policy and the `AND` policy, the performance of the `OR` policy is better than the performance of the `AND` policy. Notice that this performance gap becomes more and more obvious as the sending rate of transactions increases. Thanks to Hyperldger Fabric providing a clear and concise syntax to define endorsement policies, it is possible to achieve easy transformation between different operators. However, we have found through experiments that, although there may be no logical difference between different expressions that can represent the same meaning, there are performance differences between logically equivalent expressions in actual tests. For simple transformations that do not increase the complexity of the expression, the `k-OutOf` policy performs well at low sending rates. When the sending rate becomes higher, the policy needs to be chosen according to the actual situation. For equivalent transformations that increase the complexity of the expression, the increase in complexity implies an increase in sub-expressions. This requires communicating with more nodes and processing more information, leading to a rapid increase in latency.

Combining the results of these four experiments, some inspirations on how to choose appropriate endorsement strategies can be summarized. If a developer is torn between the `OR` policy and the `AND` policy, the difference in performance between the two is not very significant when the sending rate of the transactions is kept at a low level. But when the transaction is sent at a larger rate, the `OR` policy is more recommended. Moreover, as far as the determination of the parameter k for the `k-OutOf` policy is concerned, the larger the value of parameter k, the higher the average latency of the transaction corresponding to that policy. Thus, a smaller value of k is worthwhile. Based on our three theorems presented above, there are cases where the expression of endorsement policies can be rewritten through transformations. Notice that logical equivalence does not necessarily imply equivalence in terms of production performance. The choice of the most suitable expression should be contingent upon the specific sending rate and operational demands.

7. Conclusions and Outlook

In order to promote blockchain-based IoT applications, this paper focuses on the transaction latency in the execution phase, i.e., the endorsement process, from the perspective of EOV architecture. We review the EOV architecture of Hyperldger Fabric and analyze the process of endorsement flow in it. And we summarize three theorems from the definition of the endorsement policy of Hyperldger Fabric, and give rigorous proofs for the three theorems from a mathematical point of view. In addition, based on queuing theory and Little's law, we develop a latency analytical model to analyze the composition of latency in the endorsement process and the reasons for its formation. Last but not least, we test and compare the transaction latency for different endorsement policies at different sending rates. We also analyze the reasons for the differences in performance between different endorsement policies and give suggestions on the choice of endorsement policies.

In further research, the EOV architecture of Hyperldger Fabric will be focused on continuously. We will continue to analyze the factors affecting the performance of Hyperldger Fabric from the ordering phase and validation phase, in addition to the execution phase, and will try to come up with optimizations for the performance of Hyperldger Fabric. Simultaneously, we will also delve deeper into the endorsement policy of Hyperldger Fabric and keep optimizing the endorsement process with fewer computational resources and lower energy consumption.

Author Contributions: Conceptualization, X.P. and H.S.; methodology, X.P.; software, H.D.; validation, X.P., H.S. and H.D.; formal analysis, H.D.; investigation, H.D.; resources, X.P.; data curation, H.S.; writing—original draft preparation, H.D.; writing—review and editing, X.P.; visualization, H.D.; supervision, H.S.; project administration, H.D.; funding acquisition, H.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, grant number 61773137, the Natural Science Foudation of Shandong Province, grant number ZR2023ME162, the Natural Science Foudation of Shandong Province, grant number ZR2020MF032, and the State Grid Corporation of China, grant number SGTC0000JWJS2200148.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets analyzed during this study can be obtained from the corresponding author on reasonable requests.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Nakamoto, S. *Bitcoin: A Peer-to-Peer Electronic Cash System*; BN Publishing: La Vergne, TN, USA, 2018.
2. Reyna, A.; Martín, C.; Chen, J.; Soler, E.; Díaz, M. On blockchain and its integration with IoT. Challenges and opportunities. *Future Gener. Comput. Syst.* **2018**, *88*, 173–190. [CrossRef]

3. Christidis, K.; Devetsikiotis, M. Blockchains and smart contracts for the internet of things. *IEEE Access* **2016**, *4*, 2292–2303. [CrossRef]
4. Wang, X.; Zha, X.; Ni, W.; Liu, R.P.; Guo, Y.J.; Niu, X.; Zheng, K. Survey on blockchain for Internet of Things. *Comput. Commun.* **2019**, *136*, 10–29. [CrossRef]
5. Connors, C.; Sarkar, D. Review of Most Popular Open-Source Platforms for Developing Blockchains. In Proceedings of the 2022 Fourth International Conference on Blockchain Computing and Applications (BCCA), San Antonio, TX, USA, 5–7 September 2022; pp. 20–26.
6. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, Y.; et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. In Proceedings of the Thirteenth EuroSys Conference, Porto, Portugal, 23–26 April 2018; pp. 1–15.
7. Antwi, M.; Adnane, A.; Ahmad, F.; Hussain, R.; ur Rehman, M.H.; Kerrache, C.A. The case of HyperLedger Fabric as a blockchain solution for healthcare applications. *Blockchain Res. Appl.* **2021**, *2*, 100012. [CrossRef]
8. Wang, S.; Taha, A.F.; Wang, J.; Kvaternik, K.; Hahn, A. Energy crowdsourcing and peer-to-peer energy trading in blockchain-enabled smart grids. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *49*, 1612–1623. [CrossRef]
9. Dai, H.N.; Zheng, Z.; Zhang, Y. Blockchain for Internet of Things: A survey. *IEEE Internet Things J.* **2019**, *6*, 8076–8094. [CrossRef]
10. Dabbagh, M.; Choo, K.K.R.; Beheshti, A.; Tahir, M.; Safa, N.S. A survey of empirical performance evaluation of permissioned blockchain platforms: Challenges and opportunities. *Comput. Secur.* **2021**, *100*, 102078. [CrossRef]
11. Thakkar, P.; Nathan, S.; Viswanathan, B. Performance benchmarking and optimizing hyperledger fabric blockchain platform. In Proceedings of the 2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), Milwaukee, WI, USA, 25–28 September 2018; pp. 264–276.
12. Kwon, M.; Yu, H. Performance improvement of ordering and endorsement phase in hyperledger fabric. In Proceedings of the 2019 Sixth International Conference on Internet of things: Systems, Management and Security (IOTSMS), Granada, Spain, 22–25 October 2019; pp. 428–432.
13. Zhou, Q.; Huang, H.; Zheng, Z.; Bian, J. Solutions to scalability of blockchain: A survey. *IEEE Access* **2020**, *8*, 16440–16455. [CrossRef]
14. Khan, D.; Jung, L.T.; Hashmani, M.A. Systematic literature review of challenges in blockchain scalability. *Appl. Sci.* **2021**, *11*, 9372. [CrossRef]
15. Xie, J.; Yu, F.R.; Huang, T.; Xie, R.; Liu, J.; Liu, Y. A survey on the scalability of blockchain systems. *IEEE Netw.* **2019**, *33*, 166–173. [CrossRef]
16. Gorenflo, C.; Lee, S.; Golab, L.; Keshav, S. FastFabric: Scaling hyperledger fabric to 20 000 transactions per second. *Int. J. Netw. Manag.* **2020**, *30*, e2099. [CrossRef]
17. Nakaike, T.; Zhang, Q.; Ueda, Y.; Inagaki, T.; Ohara, M. Hyperledger fabric performance characterization and optimization using goleveldb benchmark. In Proceedings of the 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Toronto, ON, Canada, 2–6 May 2020; pp. 1–9.
18. Berendea, N.; Mercier, H.; Onica, E.; Riviere, E. Fair and efficient gossip in hyperledger fabric. In Proceedings of the 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS), Singapore, 29 November–1 December 2020; pp. 190–200.
19. Hang, L.; Kim, B.; Kim, D. A transaction traffic control approach based on fuzzy logic to improve hyperledger fabric performance. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 2032165. [CrossRef]
20. Sharma, A.; Schuhknecht, F.M.; Agrawal, D.; Dittrich, J. How to databasify a blockchain: The case of hyperledger fabric. *arXiv* **2018**, arXiv:1810.13177.
21. Yamashita, K.; Nomura, Y.; Zhou, E.; Pi, B.; Jun, S. Potential risks of hyperledger fabric smart contracts. In Proceedings of the 2019 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE), Hangzhou, China, 24 February 2019; pp. 1–10.
22. Liu, C.; Li, M.; Wang, Y.; Wang, Y.; Huo, D.; Chen, Y. Achieve better endorsement balance on blockchain systems. In Proceedings of the 2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Dalian, China, 5–7 May 2021; pp. 581–586.
23. Shammar, E.A.; Zahary, A.T.; Al-Shargabi, A.A. An attribute-based access control model for Internet of things using hyperledger fabric blockchain. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 6926408. [CrossRef]
24. Qin, X.; Huang, Y.; Yang, Z.; Li, X. LBAC: A lightweight blockchain-based access control scheme for the internet of things. *Inf. Sci.* **2021**, *554*, 222–235. [CrossRef]
25. Wang, C.; Chu, X. Performance characterization and bottleneck analysis of hyperledger fabric. In Proceedings of the 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS), Singapore, 29 November–1 December 2020; pp. 1281–1286.
26. Soelman, M.; Andrikopoulos, V.; Pérez, J.A.; Theodosiadis, V.; Goense, K.; Rutjes, A. Hyperledger fabric: Evaluating endorsement policy strategies in supply chains. In Proceedings of the 2020 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS), Oxford, UK, 3–6 August 2020; pp. 145–152.
27. Hyperledger-Fabricdocs Documentation. Available online: https://hyperledger-fabric.readthedocs.io/en/latest/whatis.html (accessed on 16 October 2023).

28. Xu, L.; Chen, W.; Li, Z.; Xu, J.; Liu, A.; Zhao, L. Solutions for concurrency conflict problem on Hyperledger Fabric. *World Wide Web* **2021**, *24*, 463–482. [CrossRef]
29. Ravi, D.; Ramachandran, S.; Vignesh, R.; Falmari, V.R.; Brindha, M. Privacy preserving transparent supply chain management through Hyperledger Fabric. *Blockchain Res. Appl.* **2022**, *3*, 100072. [CrossRef]
30. Foschini, L.; Gavagna, A.; Martuscelli, G.; Montanari, R. Hyperledger fabric blockchain: Chaincode performance analysis. In Proceedings of the ICC 2020—2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–6.
31. Rauchs, M.; Glidden, A.; Gordon, B.; Pieters, G.C.; Recanatini, M.; Rostand, F.; Vagneur, K.; Zhang, B.Z. Distributed Ledger Technology Systems: A Conceptual Framework. 2018. Available online: https://ssrn.com/abstract=3230013 (accessed on 16 October 2023).
32. Lotfimahyari, I.; Giaccone, P. Optimal endorsement for network-wide distributed blockchains. *arXiv* **2023**, arXiv:2303.06459.
33. Sukhwani, H.; Wang, N.; Trivedi, K.S.; Rindos, A. Performance modeling of hyperledger fabric (permissioned blockchain network). In Proceedings of the 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, 1–3 November 2018; pp. 1–8.
34. Xu, X.; Sun, G.; Luo, L.; Cao, H.; Yu, H.; Vasilakos, A.V. Latency performance modeling and analysis for hyperledger fabric blockchain network. *Inf. Process. Manag.* **2021**, *58*, 102436. [CrossRef]
35. Little, J.D.; Graves, S.C. Little's law. In *Building Intuition: Insights from Basic Operations Management Models and Principles*; Springer: New York, NY, USA, 2008; pp. 81–100.
36. Hyperledger Caliper. Available online: https://github.com/hyperledger/caliper/ (accessed on 16 October 2023).