Hindawi
Wireless Communications and Mobile Computing
Volume 2023, Article ID 9957995, 20 pages
https://doi.org/10.1155/2023/9957995

Research Article

Performance Modeling of Hyperledger Fabric 2.0: A Queuing Theory-Based Approach

Ou Wu,[1] **Zhongxing Wang**,[2] **and Zhongjin Li**[3]

[1]*State Key Laboratory of Novel Software Technology, Software Institute, Nanjing University, Nanjing, China*
[2]*Institute of Technology, Shenyang Polytechnic College, Shenyang, China*
[3]*School of Computer, Hangzhou Dianzi University, Hangzhou, China*

Correspondence should be addressed to Zhongxing Wang; zhongxingwang1980@163.com

Received 13 January 2023; Revised 9 May 2023; Accepted 7 June 2023; Published 30 December 2023

Academic Editor: Tran Trung Duy

Hyperledger Fabric (shortened to Fabric) is an open-source, enterprise-level, permissioned distributed ledger technology platform with a highly modular, configurable architecture. It supports writing smart contracts in general-purpose programing languages and has become the preferred choice for enterprise-level blockchain applications. However, the transaction throughput of the Fabric system remains a critical factor that restricts the further application of this technology in various fields. Therefore, it is necessary to evaluate and optimize the performance of the Fabric blockchain platform. Existing performance modeling methods need to be improved in terms of compatibility and effectiveness. To address this, we propose a performance-compatible modeling method for Fabric using queuing theory, which considers the limited transaction pool and the situation where node groups are attacked. Using the Fabric 2.0 version as an example, we have established a model of the transaction process in the Fabric network. By analyzing the model's continuous 3D time Markov process, we solved the system stationary equation and obtained analytical expressions for performance indicators such as system throughput, system steady-state queue length, and system average response time. We conducted extensive analyses and simulations to verify the models' and formulations' accuracy and validity. We believe this approach can be extended to various scenarios in other blockchain systems.

1. Introduction

Hyperledger Fabric (shortened to Fabric) is a consortium blockchain platform that utilizes the smart contract paradigm and provides fully operable functionality. Due to its distributed nature and limitations in data processing, evaluating and optimizing the performance of various blockchains, including Fabric, has become a hot topic in academia [1–5]. With the rapid development and practical application of Fabric, there is a need for in-depth planning and deployment of network configurations. Developers need to know in advance whether the blockchain network's throughput and latency can meet their requirements [6]. Additionally, different versions and configurations of Fabric may impact the performance in terms of throughput, transaction rejection probability, average transaction response time, etc.

Many studies have used experimental and formal methods to evaluate the performance of Fabric with various versions [7, 8]. However, to our knowledge, no formal method considering comprehensive parameters has been proposed for Fabric 2.0 [9]. The Fabric 2.0 framework's different network parameters, such as peers, orders, organizations, block size, the number of finite transaction pools, as well as the state of the peer node group, should be considered to support the estimation of the system's throughput and transaction latency.

To address this gap, we proposed a theoretical performance evaluation model for Fabric 2.0 based on queuing theory, which has been accepted by BSEW of EASE 2022 [10]. The proposed model considers the limited transaction pool and constructs a 2D Markov process. However, it does not consider the scenario where the blockchain peer node group is under attack or when too many malicious nodes stop working. The state of the peer node group is also a critical factor that impacts the blockchain system's performance. Regardless of the consensus mechanism adopted by the peer node group, there will be a state where the node

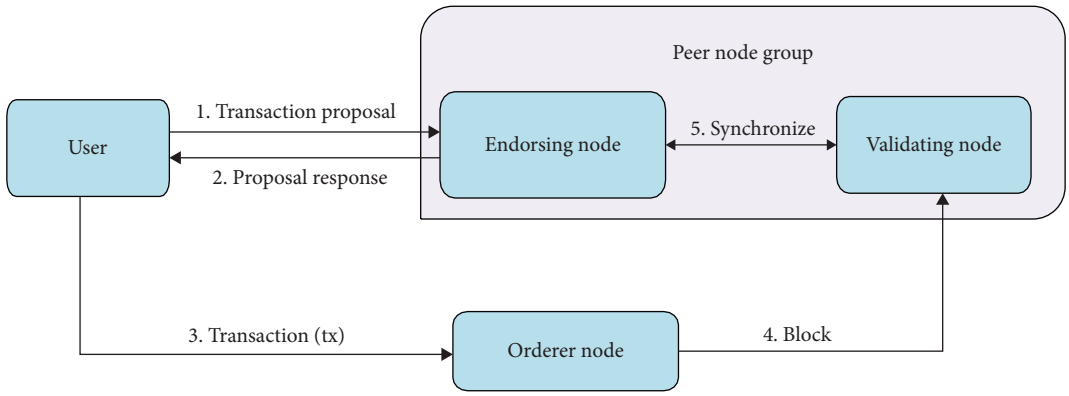

Figure 1: Fabric 2.0 transaction flow.

group is working correctly and a state where it stops working. Therefore, it is crucial to abstract this factor and integrate it into the performance model.

Building on the conference version paper, this paper extends and improves the proposed model by considering the operational status of the peer node group. The contributions of this paper are as follows:

(1) We establish a queuing theory system for the Fabric 2.0 transaction consensus process.

(2) We consider that the service process of transactions consists of two parts: block generation and consensus verification. We establish a queuing theory model that takes into account limited transaction pools and peer node group states. By analyzing the model's 3D continuous-time Markov process, constructing a subrate matrix, and solving the steady-state probability vector of the system's stationary equations. We obtain a finite series expression for performance metrics such as system rejection probability, system throughput, system queue length, and system execution time.

(3) We simulate and test the queuing theory model using the MATLAB R2016a software platform. We adjust parameters such as transaction pool capacity, transaction arrival rate, and block size to simulate the impact of system parameters on performance metrics. The results demonstrate that our proposed model is stable and accurate. We conducted error analysis on test data and theoretical data of throughput and transaction response time, verifying the model's applicability and effectiveness.

2. Background and Related Work

2.1. Hyperledger Fabric. Hyperledger Fabric (https://github.com/hyperledger/fabric) is an open-source permissioned blockchain platform that offers modular components, including membership services, chain codes, and subscription services. Figure 1 illustrates the three-phase transaction process for a Fabric 2.0 application attempting to update the ledger.

Proposal phase. In this phase, the application generates a transaction proposal and sends it to the designated node for endorsement. Each node participating in the endorsement will simulate the transaction by executing the chain code and respond to the transaction proposal by sending it back to the application.

Ordering and blocking phase. After completing the proposal phase, the application receives an endorsement proposal response from a specific set of nodes. At this stage, the application submits the endorsed transaction proposal response and the transaction itself to the order node. The ordering service creates transaction blocks based on the relevant configuration of the transaction proposal and distributes them to all peer nodes on the channel to proceed to the next stage.

Verification and submission phase. In this phase, after the peer node receives the transaction block from the order node directly or indirectly through gossip, it independently but consistently verifies each transaction in the block and updates the ledger.

For permissionless distributed blockchains like Ethereum and Bitcoin, any node can participate in consensus but is vulnerable to ledger forks. This means that different participants in the network may have different views on the order of transactions. Fabric's order node provides ordering services for transactions and solves these problems. Additionally, Fabric's design relies on a deterministic consensus algorithm, ensuring that blocks verified by peer nodes are final and correct. Several different implementations can achieve strict transaction ordering among order nodes, such as Solo, Kafka, and Raft. However, the Solo ordering service implementation is for testing only and contains only a single order node, so it has been deprecated. While the Raft allows different organizations to contribute nodes to a distributed ordering service, its ordering service is easier to set up and manage than a Kafka-based ordering service. Moreover, the Raft network is compatible with Solo users migrating to a single node. Therefore, Fabric 2.0 uses Raft as the consensus algorithm.

In the Raft consensus algorithm, a group of peer nodes implements the consensus process of transactions by electing a leader to manage the replication log. The application client

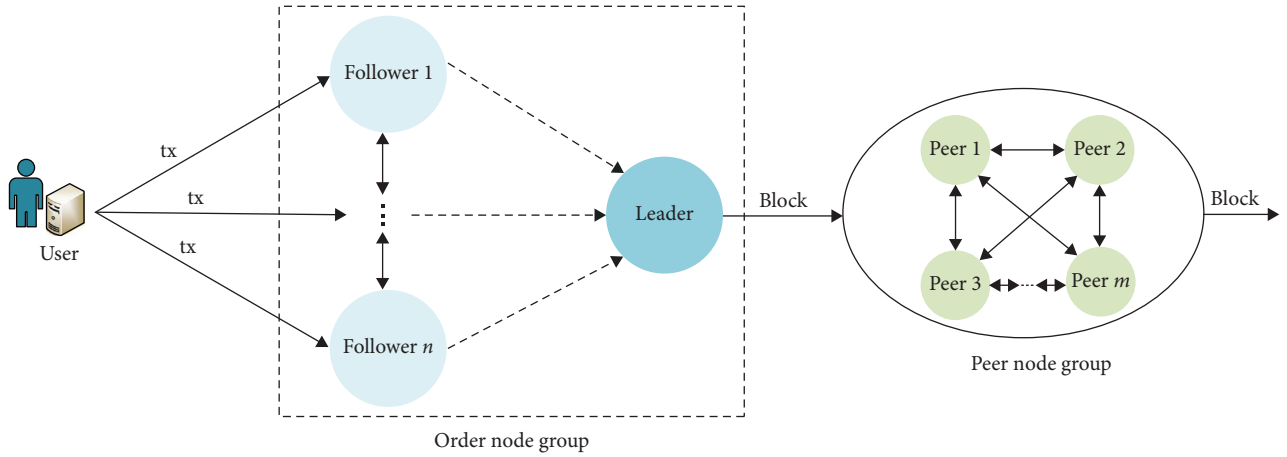

Figure 2: Consensus process of Fabric.

submits the transaction proposal containing the endorsement to the order node, which receives transactions from different application clients simultaneously. Each ordering node automatically routes received transactions to the current leader of the channel. These transactions are packaged into blocks in a defined order, stored in the ledger of the order node, and ready to be distributed to all peer nodes that have joined the channel. Thereafter, each peer node independently verifies the received block of transactions in a deterministic manner to ensure that the ledger remains consistent. Specifically, each peer node in the channel verifies each transaction in the block independently to ensure that the node recognizes the required organization. This involves verifying that the node recognition and recognition strategy match and that the transaction does not invalidate other running transactions. Invalid transactions remain in the blocks created by the ordering node, but the node marks them as invalid and does not update the ledger's state. When all peer nodes complete verification, the block is added to the chain. Figure 2 shows this consensus process.

2.2. Related Work of Performance Evaluation in Fabric. Many studies have employed experimental evaluation methods to assess the performance of different versions of Fabric, including Fabric v0.6 [3], v1.0 [2], v1.1 [11], v1.2.1 [12], v1.4 [13], and others. In a recent study, Dreyer et al. [9] examined the effect of indicators on the performance of Fabric 2.0 using testing methods and discovered that Fabric 2.0 outperforms the previous version in nearly all performance aspects.

In the domain of blockchain, modeling is another effective method for performance evaluation besides experiments [1]. Queuing theory is commonly used to model the performance of different blockchains, such as Bitcoin [14, 15] and Fabric [16]. Geyer et al. [17] introduced a queuing theory model to the Fabric platform and modeled the Solo sorting process as a queuing system. This model effectively captured the characteristics of the sorting phase in the solo implementation, but it is not applicable to the Raft or Kafka implementation of the later versions of Fabric. Jiang et al. [8] developed a hierarchical model for the Fabric v1.4.3 platform and applied queuing theory to analyze the impact of transaction arrival rate and endorsement timeout rate on the performance parameters of the Fabric transaction process.

However, the current models proposed for Fabric have limitations in terms of scalability. They do not consider a more detailed transaction processing process, nor do they investigate the relevant models of transaction pool limitations and node groups being attacked. Additionally, there is no existing research that provides an analytical solution for modeling the performance of various versions of Fabric, including 2.0. To address this challenge, our research builds an analytical model using queuing theory, which enables better performance analysis of Fabric 2.0.

3. Performance Modeling

Queuing theory is a mathematical method used to solve the performance and service quality of different types of queuing systems. In 1953, Daigle [18] proposed a classification method for queuing theory models. The representation of all classification types is described by three factors, namely $X/Y/Z$. Here, X represents the distribution of consecutive customer arrival time intervals during the input process, Y represents the service time distribution, and Z represents the number of service desks. In the 1971 Queue Theory Notation Standardization Conference, the above representation was extended to the form $X/Y/Z/A/B/C$. The meaning of the first three items remains the same, while the last three items are as follows: A represents the system capacity limit, B represents the number of customer sources, and C represents service rules such as first come first served (FCFS), last come first served, and random service.

The queues in computer communication networks and blockchain network systems are often arbitrarily complex and require the help of queuing theory to be solved. The main focus of this paper is to establish a related queuing theory model that analyzes the transaction process of Fabric 2.0 and provides a system performance evaluation.

3.1. Models and Parameters. Figure 3 illustrates the transaction consensus process of Fabric 2.0. When transactions are received by any order node, they are routed to the leader

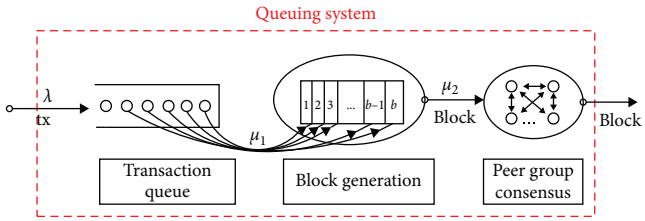

Figure 3: Fabric consensus system.

Table 1: Parameters and their meanings.

Parameter	Description
M	Exponential distribution
GI/G	General distribution
X_1	The time of a transaction was packaged
X_2	The time of a transaction was validated by peer node group
Y	The sum of the time the transaction was packaged and validated
N	The capacity of queuing system
b	The number of transactions contained in a block
λ	The average arrival rate of transactions in queuing system
μ_1	Average block generation service rate of a transaction
μ_2	Average consensus service rate of a transaction
α	Average failure rate of peer node group
β	Average repair rate of peer node group
$I(t)$	The number of transactions in the queue at time t
$J(t)$	The number of transactions in the block at time t
$K(t)$	The working status of the peer node group at time t
L_q	The average number of transactions in the queue
T_{exe}	Average transaction execution time
P_{rjc}	Transaction rejection probability
T_{resp}	Transaction response time
TSP	Transaction throughput
A_{vai}	The steady-state availability in queuing system

node's transaction pool and queued. The transactions are then packaged to generate blocks based on the set block generation time and size. These blocks are distributed to peer groups for the execution of smart contracts and transaction verification. Once all peers have verified the blocks, they are sent to the blockchain network.

Here, we provide two distribution functions used in this paper and all parameters and their respective meanings in Table 1.

Exponential distribution [18]. The probability density function of a random variable X that follows an exponential distribution with parameter λ is given by the following:

$$f_X(x) = \lambda e^{-\lambda x}, x > 0. \tag{1}$$

The mathematical expectation (or expected value) of a random variable X is given by the following:

$$E(X) = \frac{1}{\lambda}. \tag{2}$$

This means, that on average, the time between two consecutive events in a Poisson process is equal to $1/\lambda$. For example, if $\lambda = 2$, then the expected time between two events is 1/2 units of time.

Generalized Erlang distribution [19]. Consider a two-stage process where services in stage 1 are exponentially distributed with parameter μ_1 and services in stage 2 are exponentially distributed with parameter μ_2. Use $X_i, i = 1, 2$ to represent the time of each stage, and Y to represent the time sum of the two stages, that is, $Y = X_1 + X_2$. Then the distribution function of Y is the convolution of X_1 and X_2, that is, Y obeys the two-stage generalized Erlang distribution, and its probability density is as follows:

$$f_Y(y) = \frac{\mu_1\mu_2}{\mu_1 - \mu_2}(e^{-\mu_2 y} - e^{-\mu_1 y}), y > 0. \tag{3}$$

The mathematical expectation of a random variable Y is given by the following:

$$E(Y) = \frac{1}{\mu_1} + \frac{1}{\mu_2}. \tag{4}$$

4. Queuing System

We built a queuing system for the process of Fabric 2.0.

4.1. Introduction of the Queuing System. Arrival process. Clients randomly send endorsed transactions to the ordering node for queuing. Since the randomly sent transaction flow has no aftereffect (i.e., the number of transaction arrivals in nonoverlapping time intervals is independent) and stability, we assume that the transaction arrival follows a Poisson process. That is the interval between the arrival of two adjacent transactions follows the exponential distribution with parameter λ.

Service process. The transaction service is divided into two separate phases. The first stage is block generation, where transactions arrive at the orderer group and are queued for the leader to package them into blocks. The block generation time follows an exponential distribution with a parameter of μ_1. The second stage is transaction validation, where the leader sends the packaged block to the peer node group to verify the transactions. The transaction validation time follows an exponential distribution with a parameter of μ_2. So the transaction service time follows the generalized Erlang distribution, and the average service time is $\frac{1}{\mu_1} + \frac{1}{\mu_2}$.

Block generation rules. Transaction arrivals follow the FCFS principle.

The peer node group is attacked. We assume that the working status of the peer node group is divided into two types: normal service status (1) and out-of-service status when attacked (0). The failure time and repair time of the peer node group follow the exponential distribution with

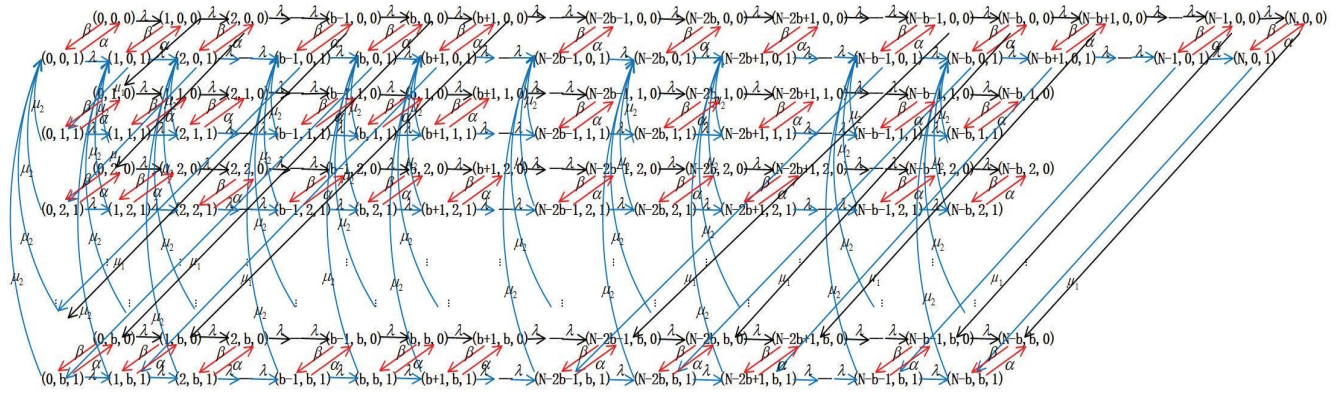

Figure 4: State transition diagram.

parameters α and β, respectively. The average failure time and average repair time are $\frac{1}{\alpha}$ and $\frac{1}{\beta}$, respectively.

The maximum system capacity. The queuing system can hold at most N transactions. When no block is generated, the transaction pool can receive at most N transactions. When the system performs consensus verification on a block, there are at most $N-b$ transactions in the queue.

Independence. We assume that all the random variables defined above are independent of each other.

4.2. A Continuous-Time Markov Process of Queuing System. In the queuing system, the order node group and the peer node group are considered two programs of a service station. The transaction service time follows the second-order generalized Erlang distribution. When the peer node group is under attack, the transaction is packaged to generate a block but not verified. Assuming that transactions arrive in the form of a Poisson process and the transaction pool accommodates a limited number of transactions, we establish a 3D continuous-time Markov process and obtain the system steady-state probability vector and performance metrics through matrix analysis.

Let $I(t), J(t), K(t)$ represent the number of transactions in the queue, in the block, and the service status of the peer node group (0 is attacked state, 1 is normal service state) at time t. Then, $(I(t), J(t), K(t))$ can be regarded as the state of the queuing system at time t, where $i = 0, 1, 2, \cdots, N; j = 0, 1, 2, \cdots, b$. Specifically, $(N-b+1, 0, 0), (N-b+2, 0, 0), \cdots, (N, 0, 0)$ and $(N-b+1, 0, 1), (N-b+2, 0, 1), \cdots, (N, 0, 1)$ denote that when no block is generated in the system, the number of transactions that can be accommodated in the queue can reach N. Similarly, $(N-b, 0, 0), (N-b, 1, 0), \cdots, (N-b, b, 0)$ and $(N-b, 0, 1), (N-b, 1, 1), \cdots, (N-b, b, 1)$ indicate that when the last block is generated, the number of cross-chain transactions that can be accumulated in the queue can reach $(N-b)$. Thus, for each case of $(I(t), J(t), K(t))$ in the system, we can write the following set Ω:

Obviously, the random process $(I(t), J(t), K(t))$ is a 3D continuous-time Markov process with the state space Ω. Figure 4 depicts the state transition relationship of $\{(I(t), J(t), K(t)) : t \geq 0\}$.

$$\begin{aligned}\Omega = \{(i,j,k) : i = 0, 1, \cdots, N; j = 0, 1, 2, \cdots, b; k = 0, 1\} = \{&(0,0,0), (0,1,0), (0,2,0), \cdots, (0,b,0); (0,0,1), (0,1,1),\\ &(0,2,1), \cdots, (0,b,1); (1,0,0), (1,1,0), (1,2,0), \cdots, (1,b,0); (1,0,1), (1,1,1), (1,2,1), \cdots, (1,b,1); \cdots; (b,0,0), (b,1,0),\\ &(b,2,0), \cdots, (b,b,0); (b,0,1), (b,1,1), (b,2,1), \cdots, (b,b,1); (b+1,0,0), (b+1,1,0), (b+1,2,0), \cdots, (b+1,b,0);\\ &(b+1,0,1), (b+1,1,1), (b+1,2,1), \cdots, (b+1,b,1); \cdots; (N-b,0,0), (N-b,1,0), (N-b,1,0), (N-b,2,0), \cdots,\\ &(N-b,b,0); (N-b,0,1), (N-b,1,1), (N-b,2,1), \cdots, (N-b,b,1); (N-b+1,0,0), (N-b+2,0,0), \cdots,\\ &(N-1,0,0), (N,0,0); (N-b+1,0,1), (N-b+2,0,1), \cdots, (N-1,0,1), (N,0,1)\}.\end{aligned} \tag{5}$$

Based on the state transition diagram, $P(i,j,k), i = 0, 1, \cdots, N; j = 0, 1, 2, \cdots, b; k = 0, 1$, represent the probability of state (i,j,k). Using this probability, we can derive all the state difference equations for this system.

For instance, from the state relationship in Figure 5 with state $(1,0,1)$, we can derive the following state difference equations: $(0,0,1)$ transitions to $(1,0,1)$ with a transaction arrival rate of $\lambda P(0,0,1)$; $(1,0,0)$ transitions to $(1,0,1)$ with a system repair rate of $\beta P(0,0,1)$; $(1,j,1) j = 1, 2, \cdots, b$ transition to $(1,0,1)$ with a transaction consensus validation rate of $\mu_2 P(1,j,1)$. Moreover, $(1,0,1)$ transitions to $(2,0,1)$ with a transaction arrival rate of $\lambda P(1,0,1)$; $(1,0,1)$ transitions to $(1,0,0)$ with a system failure rate of $\alpha P(1,0,1)$; and $(1,0,1)$ transitions to $(0,1,0)$ with a block generation rate of $\mu_1 P(1,$

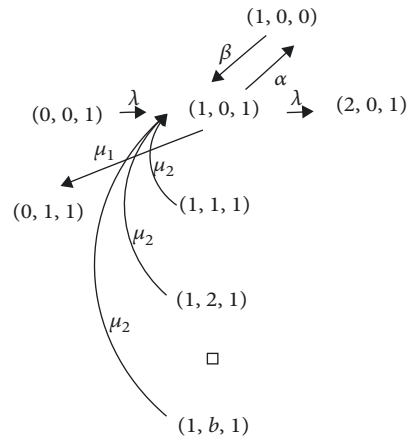

FIGURE 5: State transition relation in state (1, 0, 1).

$0, 1)$. Thus, the stationary equation for state $(1, 0, 1)$ is as follows:

$$-(\lambda+\alpha+\mu_1)P(1,0,1)+\beta P(1,0,0)+\lambda P(0,0,1)+\mu_2\sum_{j=1}^{b}P(1,j,1)=0. \tag{6}$$

Then, the state difference equations for all states are as follows:

(i) State $\{(0,0,0)\}$:

$$-(\lambda+\beta)P(0,0,0)+\alpha P(0,0,1)=0. \tag{7}$$

(ii) State $\{(0,0,1)\}$:

$$-(\lambda+\alpha)P(0,0,1)+\beta P(0,0,0)+\mu_2\sum_{j=1}^{b}P(0,j,1)=0. \tag{8}$$

(iii) State $\{(0,j,0), j=1,2,\cdots,b\}$:

$$-(\lambda+\beta)P(0,j,0)+\alpha P(0,j,1)+\mu_1 P(j,0,0)=0. \tag{9}$$

(iv) State $\{(0,j,1), j=1,2,\cdots,b\}$:

$$-(\lambda+\alpha+\mu_2)P(0,j,1)+\beta P(0,j,0)+\mu_1 P(j,0,1)=0. \tag{10}$$

(v) State $\{(i,0,0), i=1,2,\cdots,N-b\}$:

$$-(\lambda+\beta+\mu_1)P(i,0,0)+\lambda P(i-1,0,0)+\alpha P(i,0,1)=0. \tag{11}$$

(vi) State $\{(i,0,1), i=1,2,\cdots,N-b\}$:

$$-(\lambda+\alpha+\mu_1)P(i,0,1)+\beta P(i,0,0)+\lambda P(i-1,0,1)+\mu_2\sum_{j=1}^{b}P(i,j,1)=0. \tag{12}$$

(vii) State $\{(N-b,j,0), j=1,2,\cdots,b-1\}$:

$$-\beta P(N-b,j,0)+\lambda P(N-b-1,j,0)+\alpha P(N-b,j,1)=0. \tag{13}$$

(viii) State $\{(N-b,j,1), j=1,2,\cdots,b-1\}$:

$$-(\alpha+\mu_2)P(N-b,j,1)+\lambda P(N-b-1,j,1)+\beta P(N-b,j,0)=0. \tag{14}$$

(ix) State $\{(i,b,0), i=1,2,\cdots,N-b-1\}$:

$$-(\lambda+\beta)P(i,b,0)+\lambda P(i-1,b,0)+\alpha P(i,b,1)+\mu_1 P(b+i,0,0)=0. \tag{15}$$

(x) State $\{(i,b,1), i=1,2,\cdots,N-b-1\}$:

$$-(\lambda+\alpha+\mu_2)P(i,b,1)+\lambda P(i-1,b,1)+\beta P(i,b,0)+\mu_1 P(b+i,0,1)=0. \tag{16}$$

(xi) State $\{(N-b,b,0)\}$:

$$-\beta P(N-b,b,0)+\lambda P(N-b-1,b,0)+\alpha P(N-b,b,1)+\mu_1 P(N,0,0)=0. \tag{17}$$

(xii) State $\{(N-b,b,1)\}$:

$$-(\alpha+\mu_2)P(N-b,b,1)+\lambda P(N-b-1,b,1)+\beta P(N-b,b,0)+\mu_1 P(N,0,1)=0. \tag{18}$$

(xiii) State $\{(i,0,0), i=N-b+1,N-b+2,\cdots,N-1\}$:

$$-(\lambda+\beta+\mu_1)P(i,0,0)+\lambda P(i-1,0,0)+\alpha P(i,0,1)=0. \tag{19}$$

(xiv) State $\{(i,0,1), i=N-b+1,N-b+2,\cdots,N-1\}$:

$$-(\lambda+\alpha+\mu_1)P(i,0,1)+\lambda P(i-1,0,1)+\beta P(i,0,0)=0. \tag{20}$$

(xv) State $\{(N,0,0)\}$:

$$-(\beta+\mu_1)P(N,0,0)+\lambda P(N-1,0,0)+\alpha P(N,0,1)=0. \tag{21}$$

(xvi) State $\{(N,0,1), j=N-b+1,N-b+2,\cdots,N-1\}$:

$$-(\alpha+\mu_1)P(N,0,1)+\lambda P(N-1,0,1)+\beta P(N,0,0)=0. \tag{22}$$

(xvii) State $\{(i,j,0), i=1,2,\cdots,N-b-1; j=1,2,\cdots,b-1\}$:

$$-(\lambda+\beta)P(i,j,0)+\lambda P(i-1,j,0)+\alpha P(i,j,1)=0. \tag{23}$$

(xviii) State $\{(i,j,1), i=1,2,\cdots,N-b-1; j=1,2,\cdots,b-1\}$:

$$-(\lambda+\alpha+\mu_2)P(i,j,1)+\lambda P(i-1,j,1)+\beta P(i,j,0)=0. \tag{24}$$

Our objective is to solve the different equations presented above. However, solving thousands of differential equations can be a daunting task. Therefore, we need to identify a class of algorithms that can converge faster, thereby enabling us to solve them efficiently.

4.3. Algorithm Design. This section aims to utilize matrix geometry methods to solve the differential equations. However, due to the lack of symmetry in the equations, it is not possible to construct a minimum generator matrix with blocking characteristics. Therefore, we need to apply certain transformations to these equations.

Initially, we focus on the state equations $\{(i,0,k), i=N-b+1,N-b+2,\cdots,N-1; k=0,1\}$. This series of states indicates that the consensus verification in the system has ceased, and transactions will continue to accumulate until the queuing system reaches its maximum capacity N.

For Equations (19) and (20), let $i=N-b+1$, we have the following:

$$\begin{cases} -(\lambda+\beta+\mu_1)P(N-b+1,0,0)+\lambda P(N-b,0,0)+\alpha P(N-b+1,0,1)=0 \\ -(\lambda+\alpha+\mu_1)P(N-b+1,0,1)+\lambda P(N-b,0,1)+\beta P(N-b+1,0,0)=0 \end{cases}. \tag{25}$$

That is

$$\begin{cases} (\lambda+\beta+\mu_1)P(N-b+1,0,0)-\alpha P(N-b+1,0,1)=\lambda P(N-b,0,0) \\ \beta P(N-b+1,0,0)-(\lambda+\alpha+\mu_1)P(N-b+1,0,1)=-\lambda P(N-b,0,1) \end{cases}. \tag{26}$$

If

$$\begin{vmatrix} \lambda+\beta+\mu_1 & -\alpha \\ \beta & -(\lambda+\alpha+\mu_1) \end{vmatrix} \neq 0. \tag{27}$$

We have

$$\begin{cases} P(N-b+1,0,0) = \dfrac{\lambda(\lambda+\alpha+\mu_1)}{(\lambda+\alpha+\mu_1)(\lambda+\beta+\mu_1)-\alpha\beta} P(N-b,0,0) \\ + \dfrac{\lambda\alpha}{(\lambda+\alpha+\mu_1)(\lambda+\beta+\mu_1)-\alpha\beta} P(N-b,0,1) \\ P(N-b+1,0,1) = \dfrac{\lambda\beta}{(\lambda+\alpha+\mu_1)(\lambda+\beta+\mu_1)-\alpha\beta} P(N-b,0,0) \\ + \dfrac{\lambda(\lambda+\beta+\mu_1)}{(\lambda+\alpha+\mu_1)(\lambda+\beta+\mu_1)-\alpha\beta} P(N-b,0,1) \end{cases}. \tag{28}$$

Let

$$a_1 = \frac{\lambda(\lambda+\alpha+\mu_1)}{(\lambda+\alpha+\mu_1)(\lambda+\beta+\mu_1)-\alpha\beta}, \tag{29}$$

$$b_1 = \frac{\lambda\alpha}{(\lambda+\alpha+\mu_1)(\lambda+\beta+\mu_1)-\alpha\beta}, \tag{30}$$

$$a_2 = \frac{\lambda\beta}{(\lambda+\alpha+\mu_1)(\lambda+\beta+\mu_1)-\alpha\beta}, \tag{31}$$

$$b_2 = \frac{\lambda(\lambda+\beta+\mu_1)}{(\lambda+\alpha+\mu_1)(\lambda+\beta+\mu_1)-\alpha\beta}. \tag{32}$$

Then

$$\begin{bmatrix} P(N-b+1,0,0) \\ P(N-b+1,0,1) \end{bmatrix} = \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix} \begin{bmatrix} P(N-b,0,0) \\ P(N-b,0,1) \end{bmatrix}. \tag{33}$$

Similarly, let $i = N-b+2$, we get the system of the equation as follows:

$$\begin{aligned} \begin{bmatrix} P(N-b+2,0,0) \\ P(N-b+2,0,1) \end{bmatrix} &= \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix} \begin{bmatrix} P(N-b+1,0,0) \\ P(N-b+1,0,1) \end{bmatrix} \\ &= \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix}^2 \begin{bmatrix} P(N-b,0,0) \\ P(N-b,0,1) \end{bmatrix}. \\ &\vdots \end{aligned} \tag{34}$$

When $i = N-1$

$$\begin{aligned} \begin{bmatrix} P(N-1,0,0) \\ P(N-1,0,1) \end{bmatrix} &= \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix} \begin{bmatrix} P(N-2,0,0) \\ P(N-2,0,1) \end{bmatrix} \\ &= \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix}^{b-1} \begin{bmatrix} P(N-b,0,0) \\ P(N-b,0,1) \end{bmatrix}. \end{aligned} \tag{35}$$

Let

$$\begin{bmatrix} G_{j1} & G_{j2} \\ H_{j1} & H_{j2} \end{bmatrix} = \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix}^j, j = 1,2,\cdots,b-1. \tag{36}$$

Then

$$\begin{bmatrix} P(N-b+j,0,0) \\ P(N-b+j,0,1) \end{bmatrix} = \begin{bmatrix} G_{j1} & G_{j2} \\ H_{j1} & H_{j2} \end{bmatrix} \begin{bmatrix} P(N-b,0,0) \\ P(N-b,0,1) \end{bmatrix}. \tag{37}$$

That is

$$\begin{cases} P(N-b+j,0,0) = G_{j1}P(N-b,0,0) + G_{j2}P(N-b,0,1) \\ P(N-b+j,0,1) = H_{j1}P(N-b,0,0) + H_{j2}P(N-b,0,1) \end{cases}. \tag{38}$$

We can combine Equations (21) and (22) as follows:

$$\begin{cases} (\beta+\mu_1)P(N,0,0)-\alpha P(N,0,1)=\lambda P(N-1,0,0) \\ \beta P(N,0,0)-(\alpha+\mu_1)P(N-b,0,1)=-\lambda P(N-1,0,1) \end{cases}. \tag{39}$$

When

$$\begin{vmatrix} \beta+\mu_1 & -\alpha \\ \beta & -(\alpha+\mu_1) \end{vmatrix} \neq 0. \tag{40}$$

Then

$$\begin{cases} P(N,0,0)=\dfrac{\lambda(\alpha+\mu_1)}{(\alpha+\mu_1)(\beta+\mu_1)-\alpha\beta}P(N-1,0,0)+\dfrac{\lambda\alpha}{(\alpha+\mu_1)(\beta+\mu_1)-\alpha\beta}P(N-1,0,1) \\ P(N,0,1)=\dfrac{\lambda\beta}{(\alpha+\mu_1)(\beta+\mu_1)-\alpha\beta}P(N-1,0,0)+\dfrac{\lambda(\beta+\mu_1)}{(\alpha+\mu_1)(\beta+\mu_1)-\alpha\beta}P(N-1,0,1) \end{cases}. \tag{41}$$

Let

$$a_N=\frac{\lambda(\alpha+\mu_1)}{(\alpha+\mu_1)(\beta+\mu_1)-\alpha\beta}, b_N=\frac{\lambda\alpha}{(\alpha+\mu_1)(\beta+\mu_1)-\alpha\beta}, \tag{42}$$

$$c_N=\frac{\lambda\beta}{(\alpha+\mu_1)(\beta+\mu_1)-\alpha\beta}, d_N=\frac{\lambda(\beta+\mu_1)}{(\alpha+\mu_1)(\beta+\mu_1)-\alpha\beta}. \tag{43}$$

Then

$$\begin{bmatrix} P(N,0,0) \\ P(N,0,1) \end{bmatrix}=\begin{bmatrix} a_N & b_N \\ c_N & d_N \end{bmatrix}\begin{bmatrix} G_{b-1,1} & G_{b-1,2} \\ H_{b-1,1} & H_{b-1,2} \end{bmatrix}\begin{bmatrix} P(N-b,0,0) \\ P(N-b,0,1) \end{bmatrix}. \tag{44}$$

That is

$$\begin{cases} P(N,0,0)=\left(a_N G_{b-1,1}+b_N H_{b-1,1}\right)P(N-b,0,0) \\ +\left(a_N G_{b-1,2}+b_N H_{b-1,2}\right)P(N-b,0,1) \\ P(N,0,1)=\left(c_N G_{b-1,1}+d_N H_{b-1,1}\right)P(N-b,0,0) \\ +\left(c_N G_{b-1,2}+d_N H_{b-1,2}\right)P(N-b,0,1) \end{cases}. \tag{45}$$

Taking Equations (33–35) and (45) into (15–18), we get the following:

$$\begin{aligned} &-(\lambda+\beta)P(N-2b+j,b,0)+\lambda P(N-2b+j-1,b,0) \\ &\quad+\alpha P(N-2b+j,b,1)+\mu_1 G_{j,1}P(N-b,0,0) \\ &\quad+\mu_1 G_{j,2}P(N-b,0,1)=0, j=1,2,\cdots,b-1, \end{aligned} \tag{46}$$

$$\begin{aligned} &-(\lambda+\alpha+\mu_2)P(N-2b+j,b,1)+\lambda P(N-2b \\ &\quad+j-1,b,1)+\beta P(N-2b+j,b,0) \\ &\quad+\mu_1 H_{j,1}P(N-b,0,0)+\mu_1 H_{j,2}P(N-b,0,1) \\ &\quad=0, j=1,2,\cdots,b-1, \end{aligned} \tag{47}$$

$$\begin{aligned} &-\beta P(N-b,b,0)+\lambda P(N-b-1,b,0)+\alpha P(N-b,b,1) \\ &\quad+\mu_1\left(a_N G_{b-1,1}b_N H_{b-1,1}\right)P(N-b,0,0) \\ &\quad+\mu_1\left(a_N G_{b-1,2}+b_N H_{b-1,2}\right)P(N-b,0,1)=0, \end{aligned} \tag{48}$$

$$\begin{aligned} &-(\alpha+\mu_2)P(N-b,b,1)+\lambda P(N-b-1,b,1) \\ &\quad+\beta P(N-b,b,0)+\mu_1\left(c_N G_{b-1,1}+d_N H_{b-1,1}\right)P(N \\ &\quad-b,0,0)+\mu_1\left(c_N G_{b-1,2}+d_N H_{b-1,2}\right)P(N-b,0,1)=0. \end{aligned} \tag{49}$$

By combining Equations (7–14), (19–24), and (46–49), we can form a system of modified equations. This system can be used to obtain a $2(N-b+1)\times 2(N-b+1)$ minimum generator matrix.

$$Q=\begin{bmatrix} B_0 & A_0 & & & & & & & & \\ B_1 & A_1 & A_0 & & & & & & & \\ B_2 & & A_1 & A_0 & & & & & & \\ \vdots & & & A_1 & A_0 & & & & & \\ \vdots & & & & \ddots & \ddots & & & & \\ B_b & & & & & A_1 & A_0 & & & \\ & B_b & & & & & A_1 & A_0 & & \\ & & B_b & & & & & \ddots & \ddots & \\ & & & \ddots & & & & & \ddots & \ddots \\ & & & & \ddots & & & & A_1 & A_0 \\ & & & & & B_b & C_1 & C_2 & \cdots & C_{b-1} & A_M \end{bmatrix}, \tag{50}$$

where $A_0, A_1, B_0, B_i\ (i=1,2,\cdots,b), C_j\ (j=1,2,\cdots,b-1)$, A_M are $2(b+1)\times 2(b+1)$-order square matrix, and

$$A_0 = \begin{bmatrix} \lambda & & & \\ & \lambda & & \\ & & \ddots & \\ & & & \lambda \end{bmatrix}, \tag{51}$$

$$A_1 = \begin{bmatrix} -(\lambda+\beta+\mu_1) & \beta & & & & & & \\ \alpha & -(\lambda+\alpha+\mu_1) & & & & & & \\ & & -(\lambda+\beta) & \beta & & & & \\ & & \mu_2 & \alpha & -(\lambda+\alpha+\mu_2) & & & \\ & & \vdots & & & \ddots & & \\ & & & \ddots & & \ddots & -(\lambda+\beta) & \beta \\ & & \mu_2 & & & & \alpha & -(\lambda+\alpha+\mu_2) \end{bmatrix}, \tag{52}$$

$$B_0 = \begin{bmatrix} -(\lambda+\beta) & \beta & & & & & & \\ \alpha & -(\lambda+\alpha) & & & & & & \\ & & -(\lambda+\beta) & \beta & & & & \\ & & \mu_2 & \alpha & -(\lambda+\alpha+\mu_2) & & & \\ & & \vdots & & & \ddots & & \\ & & & \ddots & & \ddots & -(\lambda+\beta) & \beta \\ & & \mu_2 & & & & \alpha & -(\lambda+\alpha+\mu_2) \end{bmatrix}, \tag{53}$$

$$B_1 = \begin{bmatrix} 0 & 0 & \mu_1 & 0 & \cdots & 0 & 0 \\ & & & \mu_1 & & & \\ & & & \vdots & & & \end{bmatrix}, \tag{54}$$

$$B_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & \mu_1 & & \cdots & 0 & 0 \\ & & & & & \mu_1 & & & \\ & & & & & \vdots & & & \end{bmatrix}$$

$$\vdots \tag{55}$$

$$B_b = \begin{bmatrix} 0 & 0 & 0 & 0 & \cdots & \mu_1 & \\ & & & & & & \mu_1 \end{bmatrix},$$

$$C_1 = \begin{bmatrix} 0 & 0 & \cdots & 0 & \mu_1 G_{11} & \mu_1 G_{11} \\ & & & & \mu_1 H_{12} & \mu_1 H_{12} \\ & & & & & \end{bmatrix}, \tag{56}$$

$$C_2 = \begin{bmatrix} 0 & 0 & \cdots & 0 & \mu_1 G_{21} & \mu_1 G_{21} \\ & & & & \mu_1 H_{22} & \mu_1 H_{22} \\ & & & & & \end{bmatrix}$$

$$\vdots \tag{57}$$

$$C_{b-1} = \begin{bmatrix} 0 & 0 & \cdots & 0 & \mu_1 G_{b-1,1} & \mu_1 G_{b-1,2} \\ & & & & \mu_1 H_{b-1,2} & \mu_1 H_{b-1,2} \\ & & & & & \end{bmatrix},$$

$$A_M = \begin{bmatrix} -(\lambda+\beta+\mu_1) & \beta & & & & & & \\ \alpha & -(\lambda+\alpha+\mu_1) & & & & & & \\ & & -\beta & \beta & & & & \\ & \mu_2 & \alpha & -(\alpha+\mu_2) & & & & \\ & \vdots & & & \ddots & & & \\ & & \ddots & & \ddots & -\beta & \beta & \\ & \mu_2 & & & & \alpha & -(\alpha+\mu_2) \end{bmatrix}, \tag{58}$$

Let $\pi = (\pi_0, \pi_1, \pi_2, \cdots, \pi_{N-b})$ be the steady-state probability vector of matrix Q, and each subvector $\pi_i = (\pi_{i00}, \pi_{i01}, \pi_{i10}, \pi_{i11}, \pi_{i20}, \pi_{i21}, \cdots, \pi_{ib0}, \pi_{ib1}), i = 1, 2, \cdots, N-b$ is a $2(b+1)$-dimension row vector. Then, the modified steady-state equations can be expressed as follows:

$$\begin{cases} \pi Q = 0 \\ \pi e = 1 \end{cases}, \tag{59}$$

where e is a column vector of appropriate dimension, then we have the following:

$$\pi_0 B_0 + \pi_1 B_1 + \pi_2 B_2 + \cdots + \pi_b B_b = 0\ , \tag{60}$$

$$\pi_0 A_0 + \pi_1 A_1 + \pi_{b+1} B_b = 0\ , \tag{61}$$

$$\pi_{i-1} A_0 + \pi_i A_1 + \pi_{i+b} B_b = 0, i = 2, 3, \cdots, N-2b, \tag{62}$$

$$\pi_{i-1} A_0 + \pi_i A_1 + \pi_{N-b} C_{i-(N-2b)} = 0, i = N-2b+1, N-2b+2, \cdots, N-b-1, \tag{63}$$

$$\pi_{N-b-1} A_0 + \pi_{N-b} A_M = 0\ , \tag{64}$$

$$\pi e = 1. \tag{65}$$

Since the matrix A_0 is a diagonal matrix, according to the matrix analysis method to solve the steady-state probability vector in the study of Elhafsi and Molle [20], we express the diagonal matrix A_0 as $A_0 = \lambda I$ (I is a square identity matrix of $2(b+1)$-order). Let $R_{N-b} = I$, then

$$\pi_{N-b} = \pi_{N-b} R_{N-b}. \tag{66}$$

According to Equation (64),

$$\pi_{N-b-1} = \pi_{N-b}\left(-\frac{1}{\lambda} A_M\right) = \pi_{N-b} R_{N-b-1}. \tag{67}$$

Here, $R_{N-b-1} = -\frac{1}{\lambda} A_M$ is called the subrate matrix.

Taking Equations (67) into (63), we get the following:

$$\begin{aligned} \pi_{N-b-(i+1)} &= \pi_{N-b}\left[-\frac{1}{\lambda}(R_{N-b-i} A_1 + C_{b-i})\right] \\ &= \pi_{N-b} R_{N-b-(i+1)}, i = 1, 2, \cdots, b-1, \end{aligned} \tag{68}$$

Where

$$R_{N-b-(i+1)} = -\frac{1}{\lambda}(R_{N-b-i} A_1 + C_{b-i}), i = 1, 2, \cdots, b-1. \tag{69}$$

Taking Equation (68) into Equation (62), we get the following:

$$\begin{aligned} \pi_{N-b-(i+1)} &= \pi_{N-b}\left[-\frac{1}{\lambda}(R_{N-b-i} A_1 + R_{N-i} B_b)\right] \\ &= \pi_{N-b} R_{N-b-(i+1)}, i = b, b+1, \cdots, N-b-1, \end{aligned} \tag{70}$$

Where

$$R_{N-b-(i+1)} = -\frac{1}{\lambda}(R_{N-b-i} A_1 + R_{N-i} B_b), i = b, b+1, \cdots, N-b-1. \tag{71}$$

According to Equation (64), we get the following:

$$\pi_0 = -\pi_{N-b}(R_1 B_1 + R_2 B_2 + \cdots + R_b B_b) B_0^{-1} = \pi_{N-b} R_0. \tag{72}$$

Then

$$R_0 = (R_1 B_1 + R_2 B_2 + \cdots + R_b B_b) B_0^{-1}\ . \tag{73}$$

The solving process of R_i are shown in Algorithm 1.

Input: $I, \lambda, A_M, N, b, A_1, C_1, C_2, \cdots, C_{b-1}, B_0, B_b$
Output: $R_0, R_1, \cdots, R_{N-b}$
1. $R_{N-b} \longleftarrow I$
2. $R_{N-b-1} \longleftarrow -\frac{1}{\lambda} A_M$
3. for $i \longleftarrow 1$ to $b-1$ by 1 do
4. $R_{N-b-(i+1)} \longleftarrow -\frac{1}{\lambda}(R_{N-b-i}A_1 + C_{b-i})$
5. end for
6. for $i \longleftarrow b$ to $N-b-2$ by 1 do
7. $R_{N-b-(i+1)} \longleftarrow -\frac{1}{\lambda}(R_{N-b-i}A_1 + R_{N-i}B_b)$
8. end for
9. $R_0 \longleftarrow (R_1B_1 + R_2B_2 + \cdots + R_bB_b)B_0^{-1}$

Algorithm 1: R_i algorithm.

For solve π_{N-b}, we combine Equations (61) and (65)

$$\begin{cases} \pi_{N-b}(R_0A_0 + R_1A_1 + R_{b+1}B_b)B_0^{-1} = 0 \\ \pi_{N-b}(R_0 + R_1 + R_2 + \cdots + R_{N-b-1} + I) = 1. \end{cases} \tag{74}$$

Substitute the solved π_{N-b} into Equations (66–72), we obtain the steady-state probability vector $\pi = (\pi_0, \pi_1, \pi_2, \cdots, \pi_{N-b})$.

5. Performance Analysis

In order to ensure the stability of the queuing system, we have:

$$\lim_{t \longrightarrow +\infty} I(t) = L_q, \lim_{t \longrightarrow +\infty} J(t) = J_b, \lim_{t \longrightarrow +\infty} K(t) = K_b. \tag{75}$$

Based on the steady-state probability $\pi = (\pi_0, \pi_1, \pi_2, \cdots, \pi_{N-b})$ of this system, we present the finite series forms of several main indicators that reflect the performance of the queuing system, as follows:

(1) Average queue length in queuing system

$$E(L_q) = \sum_{i=1}^{N-b} \left(i \sum_{j=0}^{b} \sum_{k=0}^{1} \pi_{ijk} \right). \tag{76}$$

(2) Transaction rejection probability in queuing system

$$P_{\text{rjc}} = \sum_{j=0}^{b} \sum_{k=0}^{1} \pi_{N-b,j,k}. \tag{77}$$

(3) Average transaction execution time in queuing system

$$\begin{aligned} E(T_{\text{exe}}) &= \sum_{i=0}^{b-1} \sum_{h=0}^{\left[\frac{N-b-l}{b}\right]} \pi_{hb+l,0,1}(h+1)\left(\frac{1}{\mu_1} + \frac{1}{\mu_2}\right) \\ &+ \sum_{i=0}^{b-1} \sum_{h=0}^{\left[\frac{N-b-l}{b}\right]} \sum_{j=1}^{b} \pi_{hb+l,j,1}\left[\frac{1}{\mu_2} + (h+1)\left(\frac{1}{\mu_1} + \frac{1}{\mu_2}\right)\right] \\ &+ \sum_{i=0}^{b-1} \sum_{h=0}^{\left[\frac{N-b-l}{b}\right]} \sum_{j=1}^{b} \pi_{hb+l,j,0}\left[(h+1)\frac{1}{\mu_1}\right]. \end{aligned} \tag{78}$$

The proof process is analogous to the literature [15], where $\left[\frac{N-b-l}{b}\right]$ is an integer function.

(4) Average transaction response time in queuing system

$$E(T_{\text{resp}}) = \frac{E(L_q)}{\lambda(1 - P_{\text{rjc}})}. \tag{79}$$

(5) Throughput in queuing system

$$\text{TPS} = \lambda(1 - P_{\text{rjc}}). \tag{80}$$

(6) Availability in queuing system

$$A_{\text{vai}} = \sum_{i=0}^{N-b} \sum_{J=0}^{b} \pi_{ij1}. \tag{81}$$

System availability is defined as the probability that a group of peers is functioning correctly. In the event of an attack on the peer node group, the system may become unavailable, resulting in an accumulation of transactions in the queue.

To obtain the trends of performance indicators in relation to any relevant parameter, we can refer to Equations (76–81). However, numerical calculations require us to simulate the model results by setting the parameter variation range based on the system architecture's configuration.

6. Model Simulation and Validation

In this section, we will vary several critical parameters, such as transaction arrival rate, queuing system capacity, and block generation rate, to simulate the performance indicators of the Fabric 2.0 system.

6.1. Simulation Experiment Setup. To analyze the influence of system parameters on the above performance indicators of the system, we used the MATLAB R2016a software platform to simulate the impact of varying parameter values on the performance indicators. By setting different parameter values and running simulations, we verified the accuracy of our model and analyzed the system's sensitivity to different parameters.

6.2. Performance Evaluation

(1) Influence of transaction arrival rate λ

In this section, we set $\mu_1 = \mu_2 = 40\,(\text{tx/s})$, the capacity of the queuing system $N = 1{,}000\,(\text{txs})$, and the failure rate and the repair rate of the peer node group are $\alpha = \beta = 80\,(\text{tx/s})$, respectively. When the range of λ is set to 200–4,000 (tx/s), Figure 6(a)–6(f) shows changes in performance indicators such as the rejection probability of the system, and the average transaction response time in different block size b, and different λ.

From Figure 6(a)–6(f), we can observe that the performance indicators are significantly impacted by the transaction arrival rate λ. However, the throughput is not affected by the block size, which is proportional to the transaction arrival rate. When the block size is large, the transaction arrival rate decreases, and the average transaction execution time becomes unstable. This is because the system generates blocks irregularly due to the small number of transactions arriving. When the block size is increased, the queue length, average transaction response time, and transaction arrival rate are proportional and inversely proportional, respectively. However, when the block is larger, and the transaction arrival rate is larger, the rejection probability is correspondingly large, and the system availability becomes small. In view of the mutual restraint and constraints of several performance indicators, blindly following large blocks and high transaction arrivals will not have good performance.

(2) Influence of queuing system capacity (N)

To explore the effect of system capacity N on system performance, we set the variation range of N from 100 to 4,000 (txs). At this time, if the transaction arrival rate $\lambda\,(\text{tx/s})$is set too small, the performance indicators will be unstable, so we set $\lambda = 3{,}000\,(\text{tx/s})$, and $\mu_1 = \mu_2 = 100\,(\text{tx/s})$ and $\alpha = \beta = 100\,(\text{tx/s})$. Figure 7(a)–7(f) shows the trend of change with each performance indicator.

Figure 7(a)–7(f) illustrates that the system is in an unstable state when the system capacity N is less than 600. At this time, the images of various performance indicators of the system experience severe shaking. The main reason is that the system capacity is too small, and excessive instantaneous trading can lead to system collapse. When the capacity of the system is above 600 (txs), the system gradually stabilizes. As N increases, the queue length and response time increase proportionally, and the larger the block, the greater the effect and growth rate. But when the block is small, such as $b = 10\,(\text{txs})$, the large capacity of the system will cause the model to fail. Mainly because the block size is set too small, the system load increases, and the model fails. Therefore, an appropriate block size (such as 30–50 (txs)) can ensure the optimal performance of the system.

(3) Influence of transaction consensus rate (μ_1) or transaction generating rate (μ_2)

The parameters μ_1 and μ_2 represent the consensus rate for blocks and the block generation rate, respectively. Their values are related to the performance of ordering nodes and peer nodes. We assume that the ranges of μ_1 and μ_2 are both 0–100 (tx/s). When $\lambda = 1{,}000\,(\text{tx/s})$ and $N = 1{,}000\,(\text{txs})$, and α, β take values of 100(tx/s). Figure 8(a)–8(f) shows the variation of the performance metrics with respect to μ_1 at different block sizes.

From Figure 8(a)–8(f), it can be seen that the change of μ_1 has little effect on the length of the queue, the response time to the transaction, and the system throughput. However, with the increase of μ_1, the rejection probability first decreases and then tends to 0, the execution time decreases accordingly, and the system availability gradually increases and tends to be stable. This is because the higher the efficiency of the transaction block, the higher the probability of receiving the transaction, the lower the probability of rejection, and the shorter the transaction execution time. When μ_1 is fixed, changes in block size b have very little effect on throughput and execution time. As block size increases, the queue length and response time decrease accordingly, so large blocks are the best option in the current state. Since each performance indicator has a similar trend of change to μ_2, it will not be described in detail here.

(4) Influence of peer node group failure rate (α) or repair rate (β)

The parameters α, β represent the failure rate and repair rate of the peer node group, respectively. Their values are related to the severity of damage to the node group under attack and the speed of the repair process. The range of values for α, and β is 20–400 (tx/s), which allows us to calculate the average failure time and repair time for the system's current state. Assuming $\lambda = 1{,}000\,(\text{tx/s}), N = 1{,}000\,(\text{txs})$, and fixing μ_1 and μ_2 to 40 (tx/s), Figure 9(a)–9(f) shows the trend of performance indicators under different block size settings.

From Figure 9(a)–9(f), it can be seen that the change of α has little effect on the system throughput and transaction rejection probability. As α increases, the mean time to failure of the peer node group decreases, resulting in a decrease in queue length, transaction execution time, and transaction response time. The system availability also tends to be stable. These changes can be explained by the increase in transaction execution speed. When the block size is set to $b = 10\,(\text{txs})$, the system may become unstable, indicating that the block size should not be too small.

The impact of β on performance indicators is similar to α and does not require repetition here.

Since different parameters have an independent relationship, it is appropriate to study the impact of each parameter on the system's performance indicators. The figures provided in the previous section can also be used to conduct sensitivity analysis of different parameters on performance metrics. For example, the study shows that the system's throughput is sensitive to the transaction arrival rate (λ), and the transactions throughput (TPS) and λ are consistent over time.

The queue length of the system (L_q) is sensitive to parameters such as transaction arrival rate (λ), network size (N),

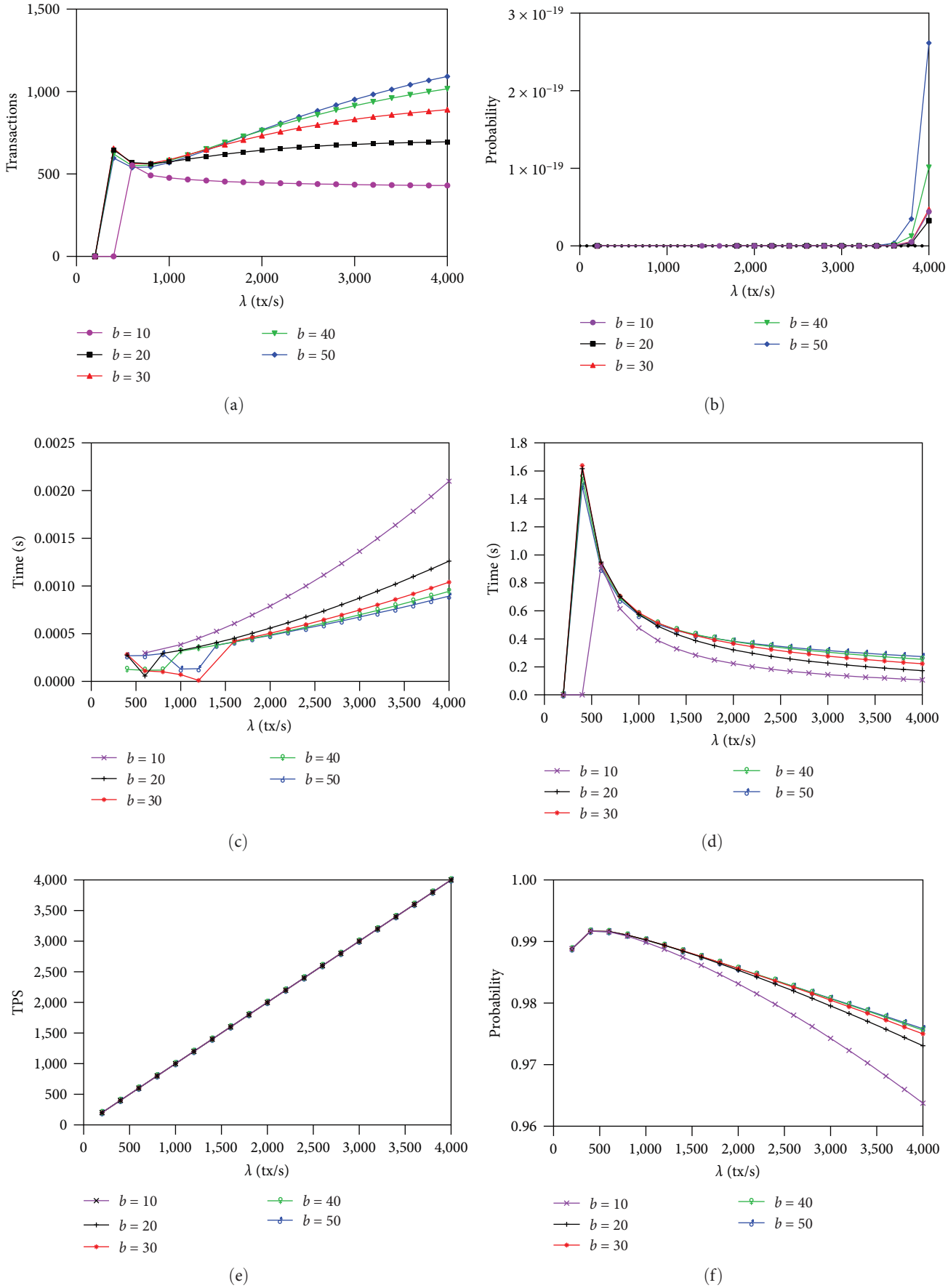

FIGURE 6: (a) The length of queue L_q, (b) the rejection probability P_{rjc}, (c) the execution time T_{exe}, (d) the response time T_{resp}, (e) the transaction throughput TPS, (f) the system availability A_{vai}.

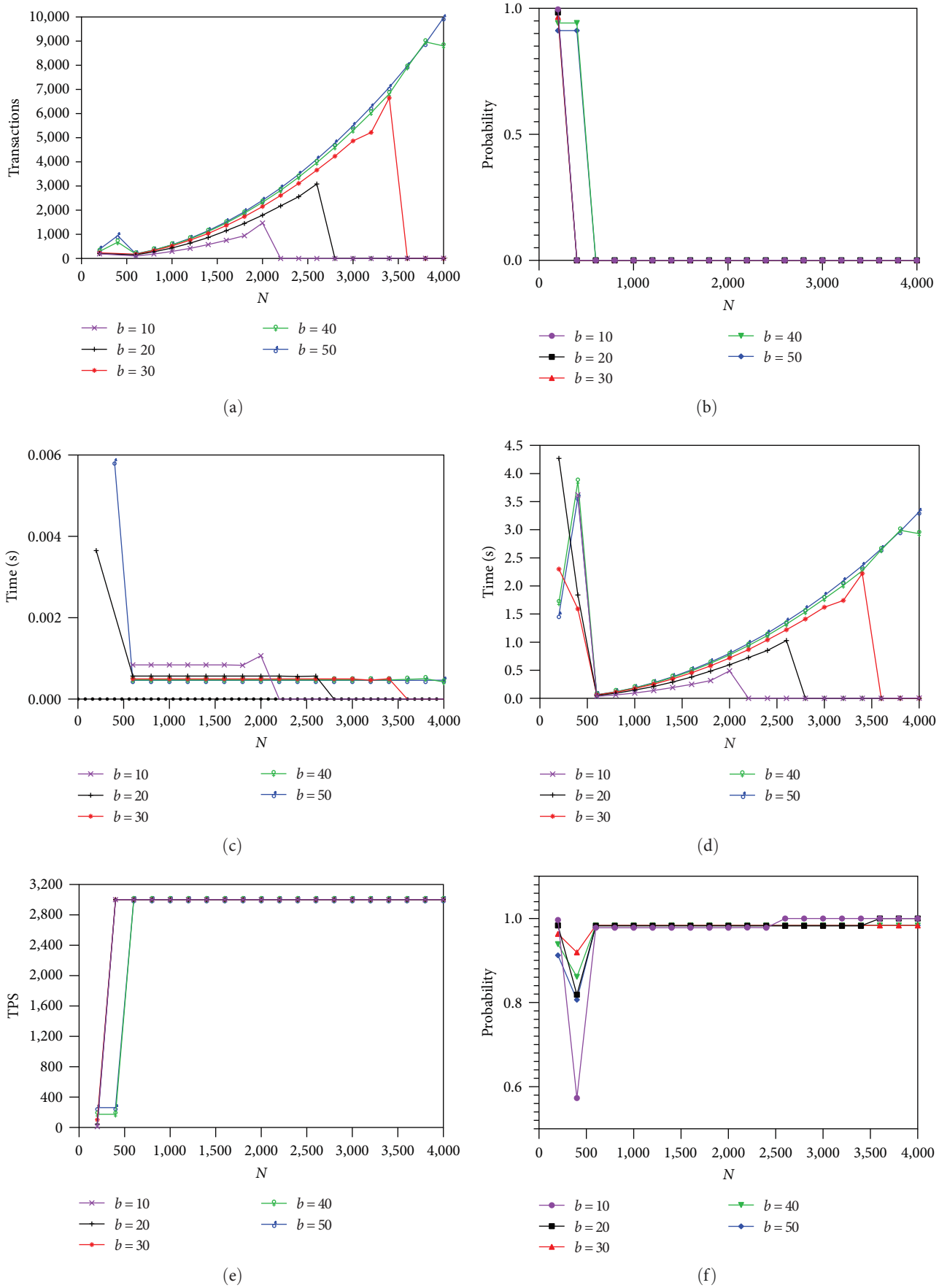

FIGURE 7: (a) The length of queue L_q, (b) the rejection probability P_{rjc}, (c) the execution time T_{exe}, (d) the response time T_{resp}, (e) the transaction throughput TPS, (f) the system availability A_{vai}.

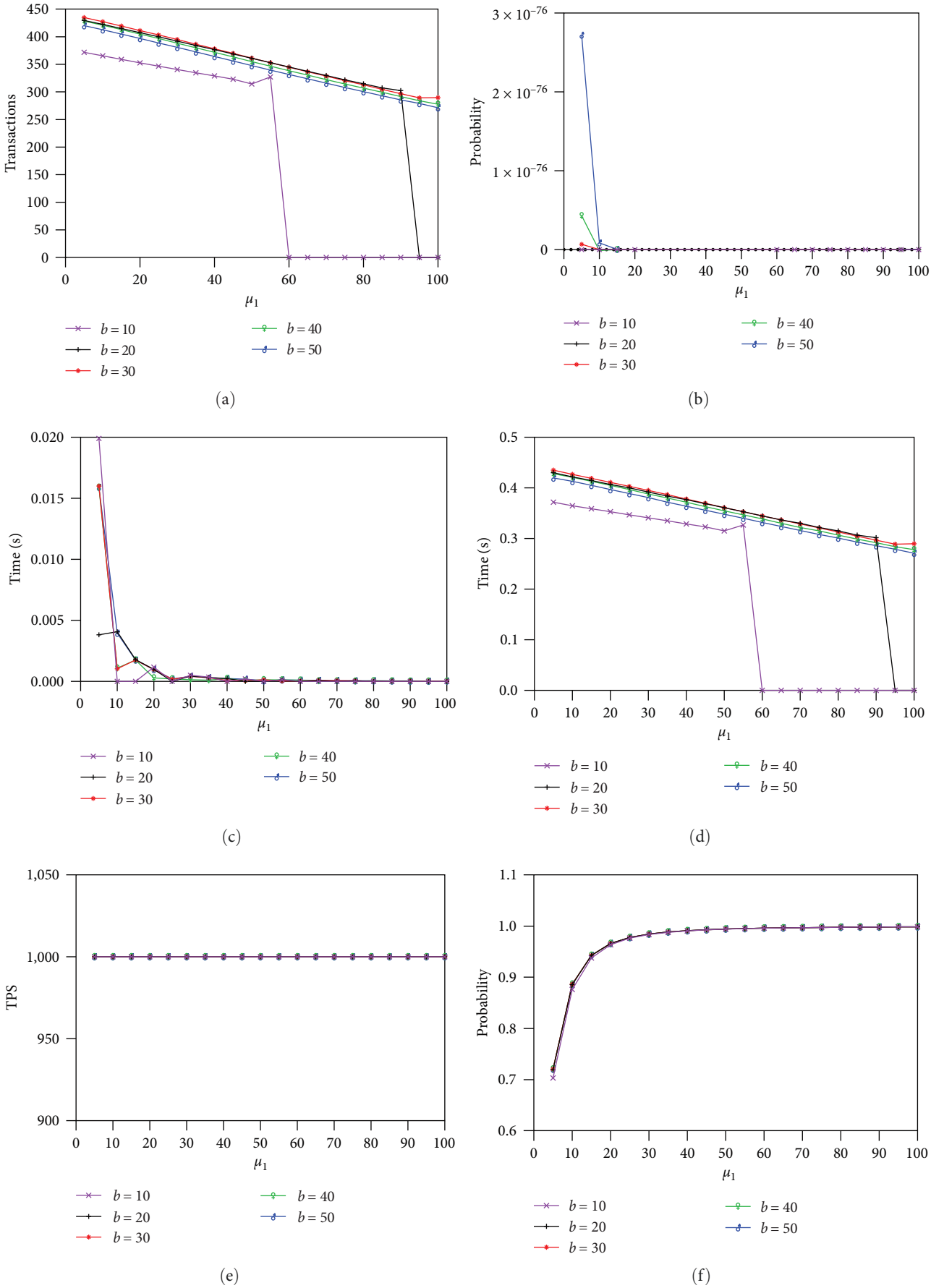

FIGURE 8: (a) The length of queue L_q, (b) the rejection probability P_{rjc}, (c) the execution time T_{exe}, (d) the response time T_{resp}, (e) the transaction throughput TPS, (f) the system availability A_{vai}.

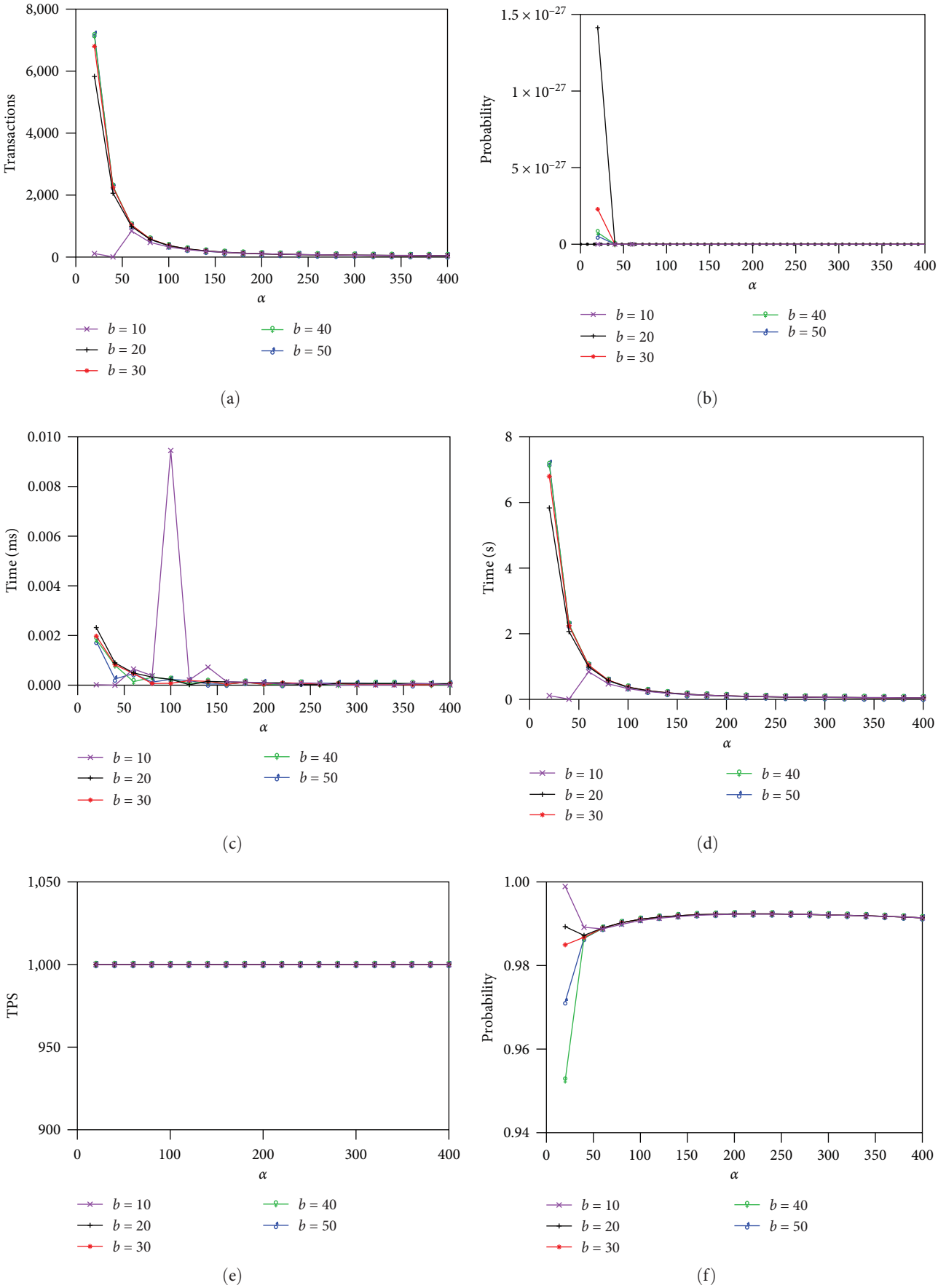

FIGURE 9: (a) The length of queue L_q, (b) the rejection probability P_{rjc}, (c) the execution time T_{exe}, (d) the response time T_{resp}, (e) the transaction throughput TPS, (f) the system availability A_{vai}.

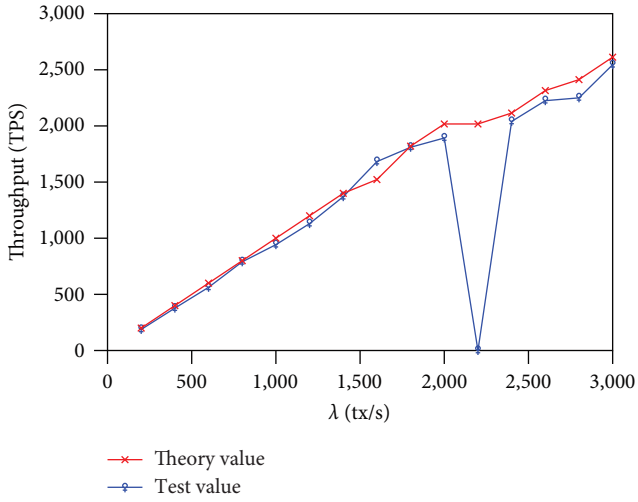

Figure 10: Comparison of the theoretical and test values.

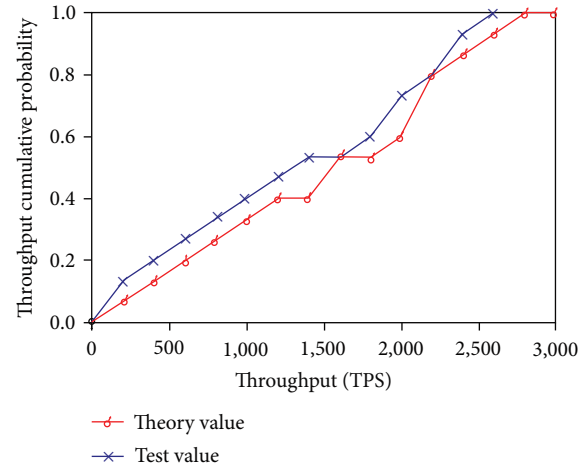

Figure 11: Throughput cumulative probability.

consensus rate (μ_1), and failure rate (α), especially when network size (N) increases. To ensure normal system operation, the block size setting (b) needs to be continuously increased. The execution time of the transaction (T_{exe}) is less affected by N, and other parameters, such as transaction pool capacity (N), have less impact on performance. The block size (b) can only play a role in combination with λ and other parameters. There is a relationship between several performance indicators and influencing parameters, so setting an appropriate block size is crucial to optimize the performance of Fabric and meet the system's performance requirements.

Our study highlights the critical role of transaction pool capacity (N) and node group working status (α) on the performance of the system cannot be overstated, in contrast to previous studies that have overlooked their importance. Therefore, it is crucial to set optimal values for N and α to enhance the efficiency of transaction processing and ensure the security of the system.

6.3. Model Experimental Verification. To validate the accuracy of our proposed model, we deployed a Fabric 2.0 network on a high-performance server with 48C 187G specifications, using the Raft ordering service to establish an ordering node and a validator group node that comprises three peers. We utilized Hyperledger Caliper (https://github.com/hyperledger/caliper), a benchmarking tool, to evaluate the performance of various blockchain solutions under custom use cases. The system's capacity was set to $N = 500\,(\text{txs}), b = 20\,(\text{txs}), \mu_1 = 40\,(\text{tx/s}), \mu_2 = 10\,(\text{tx/s})$ and the transaction arrival rate (λ) was vary from 500 to 3,000 (tx/s). We also assumed that the node group verification would halt for 50 ms, followed by a 40 ms restart, resulting in $\alpha = 20\,(\text{tx/s})$ and $\beta = 25\,(\text{tx/s})$. We conducted tests to measure the average transaction response time and system throughput and compared the results with the theoretical values.

As depicted in Figures 10 and 11, a comparison between the theoretical and test values of the system's throughput and cumulative probability was conducted. It is evident that the system's throughput test value dwindled to 1.41 (txs) when the transaction arrival rate reached 2,200 (tx/s), owing to a node cluster failure that halted the system's operation. Nonetheless, disregarding the test value's outlier, the theoretical throughput value aligns almost perfectly with the test value. Furthermore, the maximum throughput error does not exceed 162 (txs), which is negligible in comparison to the vast range of throughput levels. These results suggest that the throughput equation of the queuing system is highly effective.

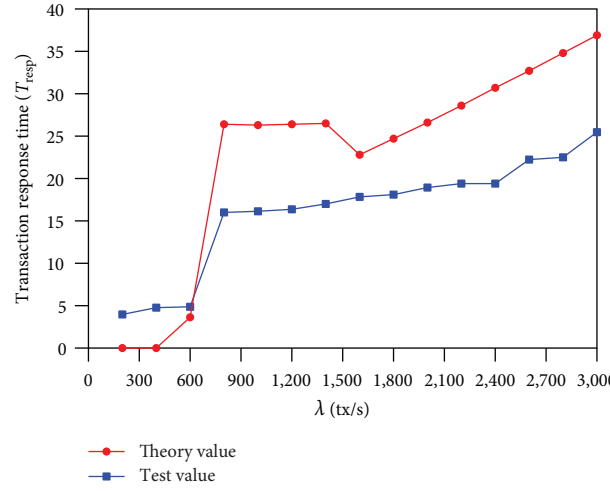

Figure 12: Comparison of the theoretical and test values.

As depicted in Figures 12 and 13, a comparison between the theoretical and test values of transaction response time and cumulative probability was conducted. Our findings indicate that when the transaction arrival rate exceeds 600 (tx/s), the test value of the transaction response time is lower than the theoretical value. This phenomenon can be attributed to the limited number of peer nodes, resulting in an actual verification rate for block consensus that surpasses the theoretical verification rate, thereby leading to a slightly better test value effect. Nonetheless, the maximum error between the theoretical and test values of response time is

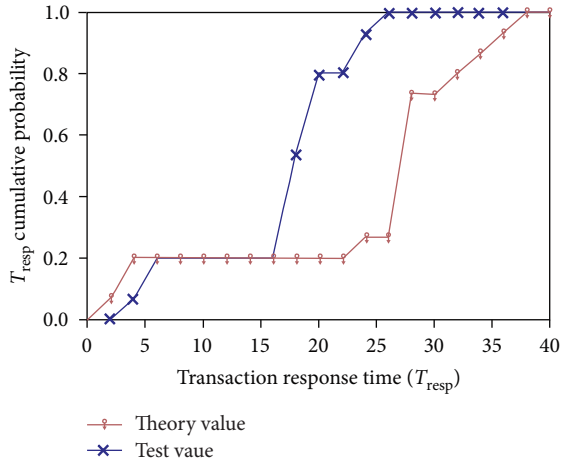

FIGURE 13: Tresp cumulative probability.

12.28 (ms), and the average error is 8.09 (ms), highlighting the need for repeated experiments to mitigate the error difference.

7. Discussion and Conclusion

In this study, we present a viable and scalable modeling approach based on Fabric 2.0. Our methodology involves the development of a queuing theory model that accounts for both the finite transaction pool and the fault state of node groups. We utilized the geometric matrix method and sub-rate matrix method to derive the solution of the difference equation and obtain crucial performance indicators related to the queuing system model, including the system queue length, system rejection probability, transaction delay time, and response time. Subsequently, we conducted a series of experiments to validate the simulation model by simulating the system's performance through a parameter change process simulation and verifying the model's effectiveness. Finally, we assessed the applicability of the proposed model by benchmarking experimental data against theoretical data.

The modeling approach proposed in this study can be extended to other blockchain systems with similar processes. As an example, we applied the approach to a typical blockchain cross-chain technology, Cosmos, to assess the model's applicability. The research outcomes have been published in the renowned international conference ICPADS 2022 [21].

The work presented in this paper has several limitations. First, our modeling approach is limited to detecting performance indicators such as maximum throughput and minimum transaction delay under current settings, such as a predetermined consensus mechanism and a fixed number of nodes. While adjusting system parameters under the current transaction arrival rate and block size settings can provide optimal system states, these parameters' optimal values cannot surpass the blockchain architecture's rigid constraints. Second, the performance metrics obtained from our model are probabilistic average values. In the case of abnormal conditions, such as system failure caused by a large influx of transactions or outlier events induced by consensus mechanism attacks, identifying their underlying causes may be challenging.

Therefore, future research directions should focus on two aspects. First, it is necessary to supplement other relevant variables and conduct performance analysis on blockchain systems to optimize the existing queuing theory models. Second, the research should aim to develop an error analysis for the consequences of outliers.

Data Availability

No underlying data were collected or produced in this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is jointly supported by the National Key Research and Development Program of China (no. 2019YFE0105500) and the Research Council of Norway (no. 309494), as well as the National Science Foundation of Zhejiang Province (no. LY22F020021), the Key Research and Development Program of Jiangsu Province (no. BE2021002), and the Innovation Project of State Key Laboratory for Novel Software Technology (Nanjing University) (no. ZZKT2022A25).

References

[1] C. Fan, S. Ghaemi, H. Khazaei, and P. Musilek, "Performance evaluation of blockchain systems: a systematic survey," *IEEE Access*, vol. 8, pp. 126 927–126 950, 2020.

[2] P. Thakkar, S. Nathan, and B. Viswanathan, "Performance benchmarking and optimizing hyperledger fabric blockchain platform," in *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pp. 264–276, IEEE, 2018.

[3] Q. Nasir, I. A. Qasse, M. A. Talib, and A. B. Nassif, "Performance analysis of hyperledger fabric platforms," *Security and Communication Networks*, vol. 2018, Article ID 3976093, 14 pages, 2018.

[4] C. Wang and X. Chu, "Performance characterization and bottleneck analysis of hyperledger fabric," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1281–1286, IEEE, 2020.

[5] S. Pongnumkul, C. Siripanpornchana, and S. Thajchayapong, "Performance analysis of private blockchain platforms in varying workloads," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, 2017.

[6] M. Kuzlu, M. Pipattanasomporn, L. Gurses, and S. Rahman, "Performance analysis of a hyperledger fabric blockchain framework: Throughput, latency and scalability," in *2019 IEEE International Conference on Blockchain (Blockchain)*, 2019.

[7] H. Sukhwani, N. Wang, K. S. Trivedi, and A. Rindos, "Performance modeling of hyperledger fabric (permissioned blockchain network)," in *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, pp. 1–8, IEEE, 2018.

[8] L. Jiang, X. Chang, Y. Liu, J. Mišić, and V. B. Mišić, "Performance analysis of hyperledger fabric platform: a hierarchical model approach," *Peer-to-Peer Networking and Applications*, vol. 13, no. 3, pp. 1014–1025, 2020.

[9] J. Dreyer, M. Fischer, and R. Tonjes, "Performance analysis of hyper-ledger fabric 2.0 blockchain platform," in *Proceedings of the Workshop on Cloud Continuum Services for Smart IoT Systems*, pp. 32–38, 2020.

[10] O. Wu, S. Li, L. Liu, H. Zhang, X. Zhou, and Q. Lu, "Performance evaluation method of hyperledger fabric 2.0," April 2022, https://conf.researchr.org/home/ease-2022/bsews-2022#event-overview.

[11] E. Androulaki, A. Barger, V. Bortnikov et al., "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the thirteenth EuroSys conference*, pp. 1–15, 2018.

[12] T. S. L. Nguyen, G. Jourjon, M. Potop-Butucaru, and K. L. Thai, "Impact of network delays on hyperledger fabric," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 222–227, IEEE, 2019.

[13] M. Kuzlu, M. Pipattanasomporn, L. Gurses, and S. Rahman, "Performance analysis of a hyperledger fabric blockchain framework: through-put, latency and scalability," in *2019 IEEE international conference on blockchain (Blockchain)*, pp. 536–540, IEEE, 2019.

[14] Y. Kawase and S. Kasahara, "Transaction-confirmation time for bitcoin: a queueing analytical approach to blockchain mechanism," in *International Conference on Queueing Theory and Network Applications*, pp. 75–88, Springer, 2017.

[15] Q.-L. Li, J.-Y. Ma, and Y.-X. Chang, "Blockchain queue theory," in *International Conference on Computational Social Networks*, pp. 25–40, Springer, 2018.

[16] J. Zhang, G. Han, and Y. Qian, "Queuing theory based co-channel interference analysis approach for high-density wireless local area networks," *Sensors*, vol. 16, no. 9, Article ID 1348, 2016.

[17] F. Geyer, H. Kinkelin, H. Leppelsack et al., "Performance perspective on private distributed ledger technologies for industrial networks," in *2019 International Conference on Networked Systems (NetSys)*, pp. 1–8, IEEE, 2019.

[18] J. N. Daigle, *Queueing Theory with Applications to Packet Telecommunication*, Springer Science & Business Media, 2005.

[19] S. E. Elmagrhraby, R. Benmansour, A. Artiba, and H. Allaoui, "Approximation of continuous distribution via the generalized erlang distribution," *IFAC Proceedings Volumes*, vol. 42, no. 4, pp. 240–245, 2009.

[20] E. H. Elhafsi and M. Molle, "On the solution to QBD processes with finite state space," *Stochastic Analysis and Applications*, vol. 25, no. 4, pp. 763–779, 2007.

[21] O. Wu, S. Li, Y. Wang, H. Li, and H. Zhang, "Modeling cross-blockchain process using queueing theory: the case of cosmos," in *2022 IEEE 28th International Conference on Parallel and Distributed Systems, ICPADS*, pp. 274–281, IEEE, Nanjing, China, 10–12 January, 2023.