On Consortium Blockchain Consistency: A Queueing Network Model Approach

Tianhui Meng, *Member, IEEE*, Yubin Zhao, *Member, IEEE*, Katinka Wolter, *Member, IEEE*, and Cheng-Zhong Xu, *Fellow, IEEE*

Abstract—Analyzing blockchain protocols is a notoriously difficult task due to the underlying large scale distributed networks. To address this problem, stochastic model-based approaches are often utilized. However, the abstract models in prior work turn out not to be adoptable to consortium blockchains as the consensus of such a blockchain often consists of multiple processes. To address the lack of efficient analysis tools, we propose a queueing network-based method for analyzing consistency properties of consortium blockchain protocols in this article. Our method provides a way to evaluate the performance of the main stages in blockchain consensus. We apply our framework to the Hyperledger Fabric system and recover key properties of the blockchain network. Using our method, we analyze the security properties of the ordering mechanism and the impact of delaying endorsement messages in consortium blockchain protocols. Then an upper bound is derived of the damage an attacker could cause who is capable of delaying the honest players' messages. Based on the proposed method, we employ analytical derivations to investigate both the security and performance features, and corroborate close agreement with measurements on a wide-area network testbed running the Hyperledger Fabric blockchain. With the proposed method, designers of future blockchains can provide a more rigorous analysis of their consortium blockchain schemes.

Index Terms—Consortium blockchain, analyzing framework, consistency, delay attack, queueing networks, permissoned blockchain

1 Introduction

Although the most important application scenario for a blockchain is a cryptocurrency, this technique is a very promising candidate in the context of the Internet of Things (IoT), enabling novel applications in smart health care, smart energy, asset tracking and smart transportation [1]. A blockchain can be defined as a distributed immutable ledger recording a range of transactions, maintained by peers without central authority with the help of a distributed cryptographic protocol [2]. All the peers participating in the blockchain network maintain copies of the ledger and execute a consensus protocol to agree on the ledger content, cryptographic hashes and digital signatures are used to ensure the integrity of transactions. As mentioned in [3], every blockchain technology is a distributed ledger technology (DLT).

Consortium blockchains are permissioned blockchains in which only the authorized organizations can gain access to the blockchain network [4], [5]. These blockchains are often associated with enterprise use, with a group of companies collaborating to leverage blockchain technology for improved business processes. Quorum,[1] Hyperledger[2] and Corda[3] are some examples of consortium blockchains. As consortium blockchains are open in all or part of the functions only for the members, such a blockchain is easier to manage and it is more flexible for business use. The consortium blockchain is also more efficient than public blockchains because the time and power-consuming Proof-of-Work (PoW) consensus is not needed.

As blockchain protocols run on large scale distributed networks, a main challenge is how to efficiently evaluate their performance and security. Motivated by the prevalence of Bitcoin [6] and other blockchain techniques, significant research has been devoted towards extending the capacity of blockchain systems, i.e., improving the efficiency of transaction processing [7], [8]. But there are still limitations in the evaluation of the performance of blockchain networks. Some authors analyze the performance of blockchains by running real blockchain applications. The cost of devices and services cost is a drawback and experimental evaluation of real systems lacks flexibility and scalability. To address this problem, a number of stochastic model-based works have been published [9], [10], [11], [12].

However, the abstract models for Nakamoto's protocol in prior works turn out to be unadoptable to consortium blockchains as the consensus of such blockchains often consists of multiple steps. It is difficult to take all the steps into consideration in a single model to analyze the protocol and evaluate

- *Tianhui Meng is with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China. E-mail: th.meng@siat.ac.cn.*
- *Yubin Zhao is with the School of Microelectronics Science and Technology, Sun Yat-Sen University, Zhuhai 519080, China, and also with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China. E-mail: zhaoyb23@mail.sysu.edu.cn.*
- *Katinka Wolter is with the Department of Mathematics and Computer Science, Freie Universität Berlin, 14195 Berlin, Germany. E-mail: katinka.wolter@fu-berlin.de.*
- *Cheng-Zhong Xu is with the State Key Lab of IoTSC & Department of Computer and Information Science, University of Macau, Macau 999078, China. E-mail: czxu@um.edu.mo.*

Manuscript received 12 Dec. 2019; revised 14 Dec. 2020; accepted 4 Jan. 2021. Date of publication 8 Jan. 2021; date of current version 20 Jan. 2021. (Corresponding author: Yubin Zhao.) Recommended for acceptance by D. Medhi. Digital Object Identifier no. 10.1109/TPDS.2021.3049915

1. https://www.goquorum.com/
2. https://www.hyperledger.org/
3. https://www.r3.com/

its performance. Furthermore, earlier work [13], [14] on analyzing consortium blockchains mainly uses the exponential distribution to describe the service time. However, for such blockchain systems, the exponential distribution is often not adequate for modelling the network behavior. Phase-type distributions allow to consider more general situations, yet they are still analytically tractable.

To alleviate the deficit in efficient analysis methods for consortium blockchain protocols, we propose a queueing network model, which represents key properties of the blockchain consensus. We use it to analyze the consistency of consortium blockchain protocols and summarise our key contributions and insights as follows:

i) We propose an analytical framework of consortium blockchain consistency based on a queueing network model, which considers the major factors that influence the consistency properties, including network delay, transaction arrival rate and ratio of compromised peers (the proportion of peers controlled by an attacker).
ii) The state space of the proposed queueing network model is large, which makes solving the model very difficult. We propose the decomposition of the queueing network into three separate parts and use the product of the three queues as an approximation of the network solution.
iii) Our method provides a formal representation of the main properties of the transaction stage in consortium blockchain consensus, including the transaction processing rate, the mean transaction latency and the system utilization. We apply our model to the Fabric system and study the key properties of the blockchain network. Measurements in the Hyperledger Fabric testbed corroborate the analytical results, as they match the analytical results very well.
iv) We investigate the impact of delays on the security of the consortium blockchain consistency using our model. We analyze delay attacks as introduced in [9] and convert them into an analytic problem. We derive an upper bound for the damage an attacker can cause when capable of delaying honest players' messages.

The analysis method is validated and justified by experimental results generated through a testbed running a Hyperledger Fabric blockchain. Our method advances the state of the art by providing a more rigorous analysis of consortium blockchain schemes for future blockchain designers. Based on the proposed model, the performance and security aspects of consortium blockchains can be analyzed and predicted. Furthermore, our technique abstracts from applications of consortium blockchain protocols, as one can neglect implementation details of the blockchain.

This paper is structured as follows: Section 2 surveys related work and Section 3 reviews the consortium blockchain architecture including a system model. The queueing network model is proposed in Section 4. Section 5 provides the analytical and experimental results. Security attributes are discussed in Section 6. Finally, Section 7 concludes this paper and discusses use cases of the proposed model.

2 Related Work

In this section, we discuss existing efforts on analyzing and evaluating blockchain protocols. Performance evaluation of Nakamoto's protocol has been investigated for several years. Decker *et al.* analyzed the block generation and propagation processes and presented a simple model to compute the growth of the Bitcoin network [15], [16]. Garay *et al.* for the first time presented a formal equivalence between the task solved by the Bitcoin protocol and the consensus problem in distributed computing [10], [17]. In [18] and [19] Kiayias *et al.* analyzed Nakatomoto's protocol in a synchronous situation and showed how to attack the chain growth by delaying the transaction confirmation time. Gervais *et al.* proposed a quantitative framework to analyze consensus of PoW blockchains and presented adversarial strategies for double-spending and selfish mining [20], [21], [22].

While Garay and Kiayias made the simplifying assumption that the network is fully synchronous [10], Pass *et al.* analyzed the consistency property of Blockchain protocols in asynchronous networks by carefully counting certain combined events called a "convergence opportunity" [9]. They introduce a *delay attack* on the consistency of Nakamoto's protocol in which the adversary delays the distribution of honest blocks to slow down the generation of new blocks on the honest chain, while mining efficiently their own blocks. The delay attack is a significant issue in PoW blockchains because the attackers do not need to make an extra effort to trigger an attack and, besides, this is a very effective attack. Through the delay attack, an attacker with less than 50 percent of the computation power can break the consistency of a blockchain system [23]. There have been a number publications on such attacks recently [23], [24], [25]. In [23] Kiffer *et al.* propose a Markov-chain-based method for analyzing Blockchain consistency and present more accurate bounds on consistency than the ones in [9]. Different from the above mentioned work, Papadis *et al.* developed stochastic network models to capture the block dynamics on each node in the blockchain network [11]. They also computed the block generation rate as a function of the nodes' communication rate and hashing power. Ilie *et al.* analyzed how to make the Bitcoin network quantum resistant and formulated the context in which a quantum enabled adversary would have to operate if it were to start attacking the Bitcoin network [26].

Li *et al.* [8], [27] were the first to provide a detailed analysis of the blockchain queueing theory. Focusing on the block-generation and blockchain-building processes, the authors established a $GI/M/1$ queueing system, based on which they derive the system's stable condition and the stationary probability vector. In [28] and [29], Kasahara *et al.* conducted research on applying queueing theory to deal with the transaction-confirmation time for Bitcoin. Memon *et al.* [30] proposed a queuing theory-based simulation model for understanding the working and theoretical aspects of the blockchain. Ricci *et al.* [31] introduced a queueing theory model to characterize the delay experienced by Bitcoin transactions. Existing queueing model-based approaches go towards building a general framework for PoW blockchains, whereas our queueing network-based method is proposed for analyzing consistency properties of consortium blockchains.

As one of the first to look at permissioned blockchains, Dinh *et al.* presented a framework called "BlockBench" for comparing the performance of different blockchain platforms (Ethereum, Parity and Hyperledger Fabric) using a set of micro and macro benchmarks [32], [33]. Thakkar *et al.* conducted an empirical study of Hyperledger Fabric by varying the parameters such as block size, endorsement policy, channels, resource allocation, and state database choices [34]. They also provided guidelines for configuring these parameters. Androulaki *et al.* introduced Hyperledger Fabric and tested the throughput and scalability of Fabric v1.1 with an application called Fabcoin [35]. In [36], Huang *et al.* propose an analytical model to analyze the network splitting probability in blockchain networks when using the Raft consensus algorithm [37]. Sousa *et al.* presented a Byzantine fault tolerant (BFT) ordering service for Fabric v1.0 on top of BFT-SMaRt and also present performance analysis for the ordering service [38], [39].

The existing approaches for evaluating consortium blockchain protocols are mainly based on running real blockchain applications, which reduces their generality. There is no general and efficient analysis method for consortium blockchains. While for public PoW blockchains the mining process dominates the system behaviour, in consortium blockchains without PoW other factors determine the system performance. Compared to existing methods, our queueing network-based model is new in the sense that it considers the major factors that influence the consistency properties, including network delay, transaction arrival rate and ratio of compromised peers. In addition, our method provides a formal way to evaluate both the performance and security aspects of consortium blockchain protocols.

3 Consortium Blockchain Architecture

This section briefly summarizes the consortium blockchain's architecture. First, we introduce the network and consensus of consortium blockchains. Then the system model is presented with the performance and security metrics we propose.

3.1 Blockchain Network Architecture

In most instances, several organizations get together as a consortium and form the consortium blockchain network. Fig. 1 shows a common Hyperledger Fabric blockchain network with three organizations. Each organization provides a peer for transaction endorsement and block commitment.[4] The Blockchain network comprises several modular components:

- The client applications (*App*) generate transaction proposals and send them to the blockchain network as input.
- The Ordering Service (*OS* or *Orderer*) is responsible for transaction ordering and block generation. The Orderer also provides basic access control for the channels by restricting who can read and write block data. An optional peer-to-peer gossip service disseminates the blocks that are output of the ordering service to the peers.
- A *Peer* (an endorsing peer or a committing peer) is a network entity that maintains a ledger and runs chaincode containers that perform read/write operations to the ledger. Peers are owned and maintained by consortium members (organizations).
- The Membership Services Provider (*MSP*) authenticates, authorizes, and manages identities on the blockchain network. The Certificate Authority (*CA*) issues X.509 certificates used to identify network components. The membership services code running in peers and orderers also authenticates and authorizes the blockchain operations.
- The *ledger* stores the current state (world state) of a blockchain system as a journal of transactions, which records all the state changes. As a ledger, blockchains are immutable, i.e., once a block is committed and added to the blockchain, it cannot be modified anymore.
- A *Channel* is a private blockchain overlay for data isolation and confidentiality. A channel-specific ledger is shared across the peers on the channel, and members must be properly authenticated to a channel in order to interact with it.
- A Chaincode (*Smart contract*) is a computer program that runs on the blockchain platform. It controls the transfer of digital assets between parties by managing access and modifications to a set of key-value pairs in the world state.

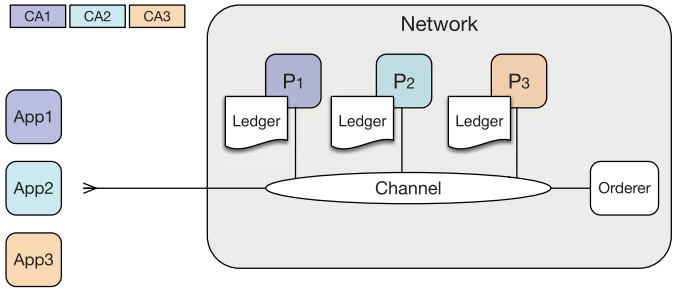


Fig. 1. A Consortium Blockchain network that consists three major components: (1) The client applications (*App*) that sent transaction proposals to the blockchain network; (2) The peers (**P**) of the blockchain network which execute and validate the transactions; and (3) The ordering service (*Orderer*) who collects the transaction proposals and packages them into blocks.

The consortium reaches an agreement on a set of policies that define the permissions of the network. When a transaction is processed in the network, in addition to the multitude of endorsements, validity and versioning checks that take place, identity verification must happen in all directions of the transaction flow. Access control lists are implemented on hierarchical layers of the network, from an ordering service down to the channels.

3.2 Consortium Blockchain Consensus

The consensus in a blockchain network is a process during which the nodes in the network provide a guaranteed ordering of the transactions and validate the blocks of transactions that need to be committed to the ledger. The consensus is often synonymous with a specific algorithm or mechanism,

4. In application scenarios, multiple peers in an organization will provide extra resilience in the case of planned or unplanned outages.

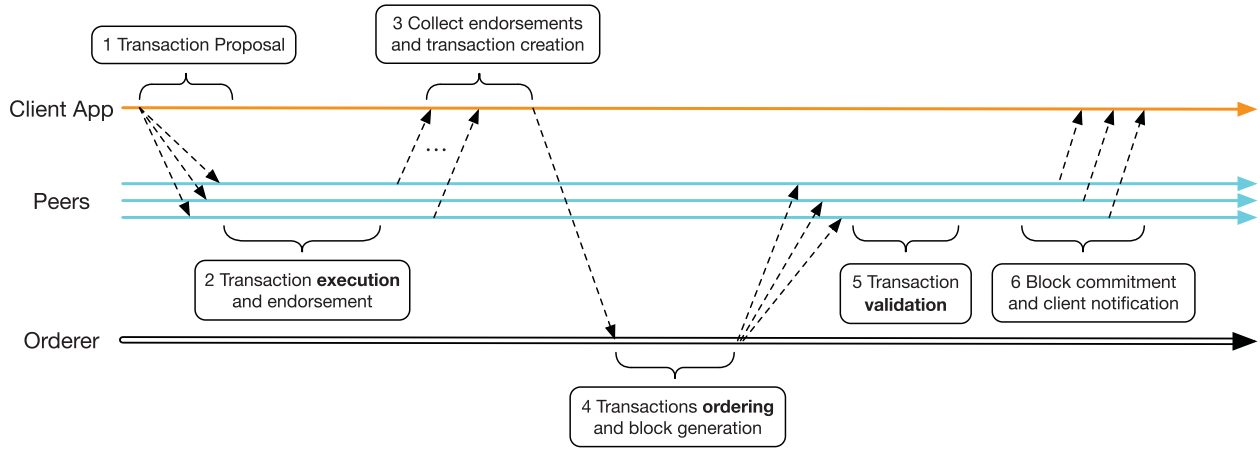


Fig. 2. The transaction flow of Hyperledger Fabric with the *execute-order-validate* architecture. The consortium blockchain consensus consists of several processes, from transaction proposal and endorsement, to ordering, validation and commitment.

such as Proof of Work (PoW), Proof of Stake (PoS), Raft and Practical Byzantine Fault Tolerance (PBFT) [40]. However, in consortium blockchains, as consensus is not dominated by PoW, it is determined by several more fine-grained processes, from transaction proposal and endorsement, to ordering, validation and commitment. The consensus of Hyperledger Fabric is the full-circle verification of the correctness of a set of transactions comprising a block.

Nearly all public blockchains adopt the "order-execute" architecture, which means that the consensus protocol first orders the transactions and then executes them on all peers sequentially in the same order. However, this approach has some drawbacks, such as the divergence of states at the peers caused by non-deterministic code and the performance deterioration caused by sequential execution. Fabric v1.0 introduces a novel *execute-order-validate* architecture, which separates the transaction flow into three steps as illustrated in Fig. 2. The *execute-order-validate* architecture has several advantages, including better scalability, a separation of trust assumptions from transaction validation and ordering, support for non-deterministic smart contracts, partitioning of smart-contract code and data across nodes, and using modular consensus implementations [41].

The consortium blockchain is a system that is 'semi-private' and has a controlled user group, but works across different organizations. It is a closed blockchain since only authorized users are allowed to access the ledger. Instead of PoW, consortium blockchains use deterministic consensus algorithms, thus any block a peer validates is guaranteed to be final and correct. Ledgers cannot fork the way they do in many other blockchain systems (like Bitcoin and Ethereum). In addition to promoting finality, separating the endorsement of chaincode execution from ordering also gives Hyperledger Fabric advantages in performance and scalability.

3.3 System Model

The architecture of a consortium blockchain network is shown in Fig. 1, which consists of three major components: (1) The client applications (*App*) that send transaction proposals to the blockchain network; (2) The peers (**P**) of the blockchain network which execute and validate the transactions; and (3) The ordering service (*Orderer*) who collects the transaction proposals and packages them into blocks.

We assume that the blockchain network consists of n peers and let $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ be the set of the peers. Each peer keeps a copy of the *ledger* that records the *transactions* of the blockchain system. In order to perform an in-depth analysis the consistency properties of consortium blockchain protocols, we focus on the following three key stages:

1) *Execution stage*: The endorsing peers validate and execute a proposed transaction, and thereby endorse it. This step corresponds to the "transaction validation" in other blockchains;
2) *Ordering stage*: The ordering service orders the transactions during the consensus protocol irrespective of transaction semantics, followed by creating and signing the block;
3) *Validation stage*: All peers receive the block and evaluate endorsements against the endorsement policy. It is also ensured that the versions of keys read by a transaction during the endorsement stage are the same as their current state in the local ledger at commit time.

3.4 Performance Metric

Considering the key processing stages of the consortium blockchain, we use a queueing network to model the consortium blockchain network's behavior and each stage is modeled as a single queueing node. We propose metrics to analyze the performance attributes of the blockchain system based on the queueing network model.

Let the stationary state of the execution stage, ordering stage and validation stage be

$$\vec{\pi}_e = (\pi_{e0}, \pi_{e1}, \pi_{e2}, \ldots, \pi_{ek}, \ldots), \vec{\pi}_o \text{ and } \vec{\pi}_v,$$

respectively. Each element in the steady state vector represents the probability that there are k customers in this stage. For example, π_{ek} means in the stationary state, the probability of k transactions in the execution stage is π_{ek}.

The performance of a blockchain system is determined by the transaction processing rate (usually expressed as *Transactions per second*, TPS) and the mean response time (also called *Transaction latency*). Let R be the random

variable that describes the response time. $E[\cdot]$ denotes the expectation of a random variable. Thus $E[R]$ represents the mean response time. ($E[R_e]$, $E[R_o]$ and $E[R_v]$ are the mean response times of the execution stage, ordering stage and validation stage respectively.)

$$\text{TPS} = \frac{N_{trx}}{T_p} \tag{1}$$

$$E[R] = \frac{1}{\lambda} E[N], \tag{2}$$

where N_{trx} is the number of transactions, T_p is the processing time and λ is the transaction arrival rate. We denote the random variable that describes the number of transaction in the system by N, and its expectation by $E[N]$. Given the stationary state distribution, the average number of transactions in each stage is obtained by

$$E[N_e] = \sum_{k=1}^{\infty} k\pi_{ek}\mathbf{1}_e, \tag{3}$$

$$E[N_o] = \sum_{k=1}^{\infty} k\pi_{ok}\mathbf{1}_o, \tag{4}$$

$$E[N_v] = \sum_{k=1}^{\infty} k\pi_{vk}\mathbf{1}_v. \tag{5}$$

Then the average number of transactions in the whole blockchain system is

$$E[N] = E[N_e] + E[N_o] + E[N_v]. \tag{6}$$

Another performance metric we propose is the utilization of each queue node. The *utilization factor* (or *utilization parameter*) U is defined as the fraction of time that the server is busy [42]. With π_0 as the steady state probability that the system is empty (an idle system), the probability of the system being busy is $1 - \pi_0$. The utilization factor in each transaction stage is given by:

$$U_e = 1 - \pi_{e0}, \tag{7}$$

$$U_o = 1 - \pi_{o0}, \tag{8}$$

$$U_v = 1 - \pi_{v0}. \tag{9}$$

3.5 Threat Model and Security Metric

We adopt the formalization of blockchain protocols proposed in [9], [10] and extend it to suit the consortium blockchain domain. A blockchain is defined as a pair of algorithms (Π, C) where Π is a stateful algorithm which maintains a local state variable *state* (in Hyperledger Fabric, it is referred to as the *World State*). The algorithm $C(state)$ is called the "record chain". It outputs a sequence of message blocks (e.g., in the Hyperledger Fabric system such blocks are ordered and validated sets of transactions). The goal is to let system participants include their messages in a block and the overall network have the same *state*.

A blockchain protocol is usually executed in an asynchronous network model that involves the following components:

- *Environment*: All the external factors related to the blockchain protocol execution are modeled by an environment $Z(1^\kappa)$, where κ is the security parameter. The environment provides all the inputs for the protocol, e.g., in the Hyperledger Fabric system, endorsement policies, network configurations and node types. The arrival rate of transaction proposals is denoted λ and the mean network delay is.
- *Honest peers*: The honest peers act and contribute according to a given blockchain protocol (Π, C). Each of them maintains a view of the blockchain and tries to include its messages in the following blocks.
- *Adversary*: An adversary A controls at most a ρ fraction of all the n peers. (ρ describes the ratio of compromised peers, which is the proportion of peers controlled by the attacker. Thus, $\eta = 1 - \rho$ is the fraction of honest peers.) The adversary is able to delay the messages sent by honest peers up to time units. In Hyperledger Fabric, adversary peers are able to delay the endorsement messages and client notifications.
- *Pluggable ordering*: The algorithm $C(state)$ collects proposed transaction update messages, orders them, and packages them into blocks. Hyperledger Fabric allows crash fault-tolerant (CFT) or Byzantine fault-tolerant (BFT) ordering.

This work considers *delay attacks* in the blockchain system. Pass *et al.* raise the issue of delay attacks on the consistency of blockchain protocols [9]. Through *delaying* honest blocks the attacker is able to hinder the creation of new blocks of the honest chain, while mining efficiently in parallel on its own private chain. In the consortium blockchain scenario, the attacker A is able to delay messages sent by honest nodes. Specifically, in the endorsement stage, the adversary is able to delay the endorsement messages of the honest players and to promote its own transaction to be processed at higher priority. We assume that the attacker is capable of delaying messages sent by honest peers up to time units (is the mean blockchain network delay). In consequence, there is a non-zero probability that *bad* endorsements are accepted by the honest peers.

We assume that a $\{k/n\}$ endorsement policy is used in the consortium blockchain network. This means that it would be sufficient if any k of the n organizations sign a transaction. The attacker may delay the endorsement message. Given that the adversary controls ρn peers, then the probability that a transaction receives the *good* endorsement service from the honest peers is

$$\alpha = \frac{C_{(1-\rho)n}^{k}}{C_n^k} = \frac{(n-k)![(1-\rho)n]!}{n![(1-\rho)n-k]!},^{5} \tag{10}$$

where C_n^k is the binomial coefficient, i.e., $\binom{n}{k}$. It is worth mentioning that in Hyperledger Fabric, the consensus algorithms are implemented in the ordering stage. However, the delay attack model is in the execution stage.

Assuming the normal service rate of the execution stage is μ_1, then the degraded service rate by the adversary is given by

$$\mu_2 = \xi\mu_1 = \frac{1}{1/\mu_1 +} = \frac{\mu_1}{1+\mu_1}, \tag{11}$$

5. If $k > (1-\rho)n$, $\alpha = 0$.

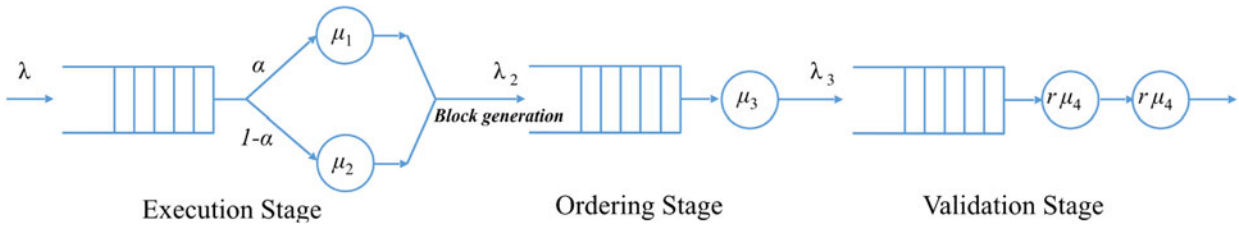


Fig. 3. Queueing network model for consortium blockchains with three nodes in series.

where ξ is called the *delay coefficient*. Considering the environment and the adversary's ability, we want to bound the damage an attacker could impose on a blockchain system. We define a deterioration factor $\mathcal{D}$ to represent the damage, which is a multiple of the mean response time when the attacker tries to delay other players' endorsement messages, comparing with the normal situation

$$\mathcal{D} = \frac{E[\widetilde{R_e}]}{E[R_e]}, \tag{12}$$

where $E[\widetilde{R_e}]$ is the compromised mean response time of the execution stage.

We introduce a queueing network model to deal with the challenges caused by the stepwise consensus process of the consortium blockchain protocol. In queueing theory, a "multiple-node" system is one in which a customer requires service at more than one node in the queueing network. Such a system may be viewed as a network of nodes, in which each node is a service center having a buffer [43].

In the next section, we define our abstraction and present the queueing network representation of consortium blockchain protocols.

4 The Queueing Network Representation

We abstract the consortium blockchain's work process into three key stages based on the *execute-order-validate* paradigm. Each stage is modeled as a single queueing node. Previous work on modeling consortium blockchains mainly uses exponential distributions to fit the service time. However, for blockchain systems the exponential distribution is not always appropriate to model the network behavior. Phase-type distributions are more versatile and allow to describe more general situations. Therefore we introduce phase-type distributed arrival or service times in our queueing systems to better model the blockchain protocol.

4.1 State Transition Graph

Our analyzing framework is abstracted as a queueing network model with three queueing nodes in tandem as shown in Fig. 3. However, for such a network the state space grows fast and is very large. The state transition process is quite complex (Fig. 4). A system state is described by $[(k,i),m,(l,j)]$, which is a tuple of the state of each queueing node. N is the number of transactions in the whole system. It is difficult to solve the model directly to obtain an analytical solution. Therefore, we simplify the queueing network by decomposition and solve the queueing nodes individually.

Supposing the transactions that are proposed by the clients arrive to the consortium blockchain network according to a Poisson process, we use a *product-form* solution as an approximation of the joint distribution in the three stages of the queueing network

$$\begin{aligned} &Prob\{N_e = n,\ N_o = m,\ N_v = r\} \\ &\approx Prob\{N_e = n\} \times Prob\{\ N_o = m\} \times Prob\{\ N_v = r\} \\ &= \pi_{e1}\mathcal{R}_e^{n-1}\mathbf{1}_e(1-U_o)U_o^m\pi_{v1}\mathcal{R}_v^{r-1}\mathbf{1}_v, \end{aligned} \tag{13}$$

where N_e is the number of transactions in the execution stage, N_o and N_v are the number of blocks in the ordering and validation stage. $\mathcal{R}$ is the Neuts' rate matrix that will be derived in the next subsection.

In the first stage of the consortium blockchain consensus, clients initiate a transaction by sending the signed transaction proposal to one or more endorsers. After collecting adequate endorsements, the client assembles them into a transaction message and sends it to the ordering service. The rate μ_2 in the execution queue node represents the endorsing service by adversary peers. In our threat model, there is a non-zero probability that the client receives bad endorsement service by an attacker. But in consortium blockchain implementations (e.g., Hyperledger Fabric), the client application has a deadline mechanism and if an endorsement takes too long, the client will change to another endorsing peer. Thus μ_2 will not be too small. The correlation of the first two queueing nodes can be safely assumed to be low.

After the execution stage, all clients send the endorsed transactions to the orderer. Once the ordering service collects a certain number of transactions (the block size), it batches them into a block and the block stream is the arrival traffic to the second queueing node. Even though the

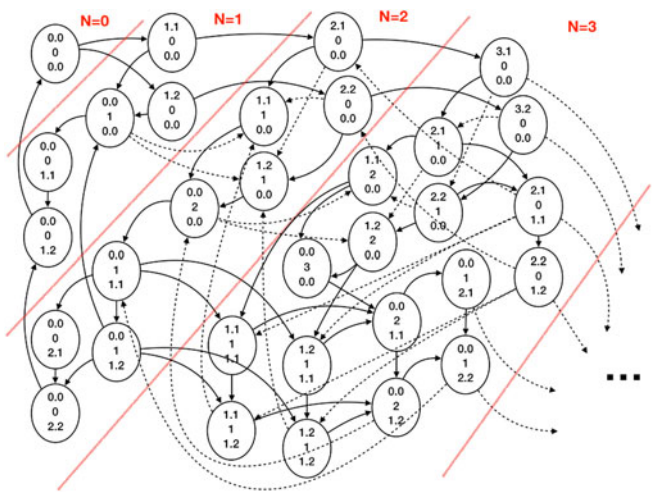


Fig. 4. State transition diagram of the queueing network model.

execution queue outputs a transaction flow that follows a phase-type distribution, we assume the block generation process is a Poisson process. Suppose that the block size parameter *BatchSize* is set to B, then the arrival rate to the ordering queue equals the mean service rate of the execution stage ($\overline{\mu_e}$) divided by the *BatchSize*

$$\lambda_2 = \frac{\overline{\mu_e}}{B} = \frac{\frac{1}{\frac{\alpha}{\mu_1}+\frac{1-\alpha}{\mu_2}}}{B} = \frac{\mu_1\mu_2}{B[(1-\alpha)\mu_1+\alpha\mu_2]}. \tag{14}$$

Following Burke's theorem [44], the departure process of an $M/M/1$ queue is a Poisson process with the same rate as the arrival process. Therefore, the departure rate of the ordering queue equals the arrival rate λ_2. That is $\lambda_3 = \lambda_2$ and the validation queue is not correlated with the ordering stage.

Finally, the product of the three queueing node solutions is used as an approximation of the network solution in steady-state. The above analysis enables us to simplify the solution of the queueing network by decomposing the network into separate parts and solving the queueing nodes independently in the following subsections.

4.2 Execution Stage

The first stage of the consortium blockchain transaction flow is the execution stage. In this stage, clients initiate a transaction by sending the signed transaction proposal to one or more endorsers who are specified by the endorsement policy. Endorsing peers *execute* the chaincode after they receive the proposal and send back a signed endorsement message to the client. The endorsement is composed of a readset and a writeset, together with metadata such as transaction ID and endorser ID.

The client waits for the endorsement messages from endorsing peers until it satisfies the endorsement policy of the transaction, which requires all peers to produce the same execution result as the policy specified. Then the client assembles endorsements into a transaction message and sends it to the ordering service. The endorsement policy plays a key role in the execution stage since it determines how fast the client can submit a transaction. In Hyperledger Fabric, the endorsement policies are logical expressions. For example, the endorsement policy *2/3 (Org_1, Org_2, Org_3)* means that it would be sufficient if any two of the three organizations sign the transaction.

The first stage of the consortium blockchain consensus is modeled as an $M/H_2/1$ queue as shown in Fig. 3. We formulate the state descriptor for the underlying queueing system as follows. The system state is described by (k, i), where k is the number of transactions in the queue, including the one in service, and i denotes the phase of service, i.e., $i = 0$ indicates the service of μ_1 and $i = 1$ denotes a service rate of μ_2 when the service is degraded by dishonest endorsing peers.[6]

4.2.1 Steady State Distribution

We first explore the steady state distribution of the queueing system. The following proposition gives the stationary state transaction distribution in the execution stage of the consortium blockchain consensus.

Proposition 1. *For the execution stage of a consortium blockchain network, the stationary state vector $\vec{\pi}_e$ is written as $\vec{\pi}_e = (\pi_{e0}, \pi_{e1}, \pi_{e2}, \ldots, \pi_{ek}, \ldots)$, where π_{e0} is the probability when there is no transaction waiting in this stage and π_{ek} $(k = 1, 2, \ldots)$ is a row vector of length r. Its ith component gives the probability of being in phase i when there are k transactions in the execution stage. Then, the stationary probability $\vec{\pi}_e$ of the underlying queueing model is given by*

$$\pi_{ek} = \pi_{e1}\mathcal{R}_e^{k-1} \quad \text{for } k = 2, 3, \ldots, \tag{15}$$

where $\mathcal{R}_e$ is the Neuts' rate matrix [45]

$$\mathcal{R}_e = \lambda(\lambda\mathbf{I} - S_e - \lambda\mathbf{1}_e\boldsymbol{\beta_e})^{-1}. \tag{16}$$

For the execution stage

$$\boldsymbol{\beta_e} = [\alpha \quad 1-\alpha],\ S_e = \begin{bmatrix} -\mu_1 & 0 \\ 0 & -\mu_2 \end{bmatrix}, \mathbf{1}_e = \begin{bmatrix} 1 \\ 1 \end{bmatrix}. \tag{17}$$

The initial vector is computed as

$$\pi_{e0} = 1 - U_e \tag{18}$$

$$\pi_{e1} = (1 - U_e)\boldsymbol{\beta_e}\mathcal{R}_e, \tag{19}$$

where $U_e = -\lambda\boldsymbol{\beta_e}S_e^{-1}\mathbf{1}_e$.

Proposition 1 enables us to evaluate the performance metrics in the first stage of the consortium blockchain consensus. The state transition diagram of the $M/H_2/1$ queue is shown in Fig. 5. We arrange the states into levels according to the number of transactions in the execution stage. From the state transition diagram, we obtain the transition rate matrix Q. The transition matrix has a special block structure – it is an infinite block tridiagonal matrix. Thus the queueing system can be solved by the *matrix-geometric* approach [46]. The proof of Proposition 1 can be found in the appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2021.3049915.

4.2.2 Performance Analysis

Based on Proposition 1, we derive the following performance properties of the execution stage.

Property 1. *The mean number of transactions in the execution stage has the following form:*

$$E[N_e] = \pi_{e1}(\mathbf{I} - \mathcal{R}_e)^{-2}\mathbf{1}_e. \tag{20}$$

The proof of Property 1 is found from page 468 of [46]. Property 1 describes the expected number of transactions when the system is in the steady state. The following two properties are well known and without proof.

Property 2. *Little's Law [47]: $E[N] = \lambda E[R]$ allows us to compute the average response time of the execution stage*

$$E[R_e] = E[N_e]/\lambda = \frac{\pi_{e1}(\mathbf{I} - \mathcal{R}_e)^{-2}\mathbf{1}_e}{\lambda}. \tag{21}$$

6. In order to simplify the model, the transmission time to the next stage is included in the service time (i.e., μ_1 and μ_2).

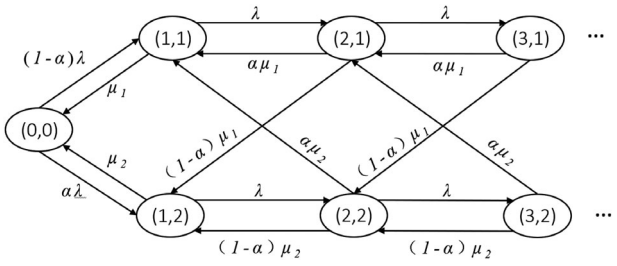


Fig. 5. State transition diagram of the Execution stage queue.

Property 3. *The utilization of the execution stage is given by*

$$U_e = 1 - Prob\{\text{an empty queue}\} = 1 - \pi_{e0}. \tag{22}$$

Properties 2 and 3 formulate the main performance metrics we use to evaluate the blockchain system, which are the response time (transaction latency) and the probability of a busy server.

4.3 Ordering Stage

After collecting enough endorsement messages from a set of peers, the client submits the transaction proposal to the ordering service to include it in the following block. The ordering service receives transactions from many different application clients concurrently. It arranges them in batches of submitted transactions and forms a deterministic sequence in which they are packaged into blocks. Finally, the ordering service outputs a hash-chained sequence of blocks. It is worth mentioning that the ordering service does not verify the correctness of the transactions in the blocks.

The number of transactions in a block depends on channel configuration parameters related to the desired size and maximum elapsed duration for assembling a block, i.e., *BatchSize* and *BatchTimeout* parameters. The blocks are recorded to the order service's ledger and distributed to the peers on the channel. If a peer is down at this time, or joins the channel later, it will receive the blocks after reconnecting to the ordering service node, or by gossiping through another peer.

Since the orderer collects transactions into blocks, we choose to model the processing of blocks and assume the generation of blocks is a Poisson process. Then, the ordering stage is modeled by a single queueing node with the arrival rate λ_2 of blocks and departure rate μ_3. Again, the processing of blocks and transmission time to the next stage are included in the expected service time $(1/\mu_3)$.

4.3.1 Performance Analysis

For the performance measures of the ordering stage, we formulate the following properties:

Property 4. *The utilization of the ordering stage (U_o), the mean number of transactions in the ordering stage ($E[N_o]$) and the mean response time of ordering ($E[R_o]$) are*

$$U_o = \frac{\lambda_2}{\mu_3} \tag{23}$$

$$E[N_o] = \frac{U_o}{1-U_o} = \frac{\lambda_2}{\mu_3 - \lambda_2} \tag{24}$$

$$E[R_o] = \frac{1}{\mu_3 - \lambda_2}. \tag{25}$$

In most test and development cases, the ordering service is in solo mode, which means there is only one ordering node. This is the simplest way to set up an ordering service. Hyperledger Fabric v1.4.1[7] introduced a crash fault tolerant (CFT) ordering service based on an implementation of the Raft protocol [48]. Raft is a consensus algorithm which implements a "leader and follower" model. A leader node is elected per channel and its decisions are replicated by the followers. The other CFT ordering service supported by Hyperledger Fabric is an adaptation of a Kafka[8] distributed streaming platform for use as a cluster of ordering nodes. But this approach utilizes a ZooKeeper ensemble for management purposes, therefore it is not designed to be run across large networks.

In any of the above three cases (Solo, Raft and Kafka based), we argue that the ordering stage can be modeled by an $M/M/1$ queue node. We will discuss security aspects of the ordering stage in Section 6.2.

4.4 Validation Stage

The blocks generated by the ordering service are distributed to all the peers on the channel directly from the orderer or through peer gossipping. As mentioned above, the ordering service is not responsible for verifying the correctness of transactions in the block. After receiving the new block, each peer will validate it independently, but in the same way as every other peer. This ensures that ledgers remain consistent across the blockchain network.

Specifically, each peer will first validate all the transactions in the block to ensure that they have been properly endorsed according to the endorsement policy of the chaincode, i.e., the Validation System Chaincode (VSCC) [34]. The transactions that fail the VSCC check are marked invalid in the block. Then, each peer performs a multi-version concurrency control (MVCC) [49] of the transactions in order to serially verify if the read-set version matches the current version. This can be seen as a ledger consistency check to ensure that the current world state is compatible with the state of the ledger when the transaction was proposed. Invalidated transactions are still retained in the immutable blockchain, but they are marked as invalid by the peer and do not update the world state.

4.4.1 Steady State Distribution

In our model, the validation stage of a consortium blockchain is modeled by a $M/E_r/1$ queue node whose arrival rate is λ_3. The service time has an Erlang-2 distribution with a sequence of 2 services, each with rate $2\mu_4$. As measured in [14], the mean elapsed time of VSCC and MVCC validation are in the same order of magnitude. Thus it is reasonable to model them by a sequence of services with the same rate.

In the $M/E_r/1$ queue, the density functions of the arrival and service processes are given by

$$a(t) = \lambda_3 e^{-\lambda_3 t}, \tag{26}$$

7. https://github.com/hyperledger/fabric/releases/tag/v1.4.1
8. https://kafka.apache.org/

$$s(x) = \frac{r\mu_4(r\mu_4 x)^{r-1}e^{-r\mu_4 x}}{(r-1)!}, x \geq 0. \quad (27)$$

Equivalently to the execution stage, the steady state solution of the transaction distribution in the validation stage is given as follows.

Proposition 2. *In a consortium blockchain network, if the stationary state vector $\vec{\pi}_v$ of the validation stage is written as $\vec{\pi}_v = (\pi_{v0}, \pi_{v1}, \pi_{v2}, \ldots, \pi_{vl}, \ldots)$, where π_{v0} is the probability when there is no transaction waiting in this stage and π_{vl}, $l = 1, 2, \ldots$ is a row vector of length r, its jth component defines the probability of being in the jth sequential service when there are l transactions in the validation stage. Then the stationary probability $\vec{\pi}_v$ of the underlying queueing model is given by*

$$\pi_{vl} = \pi_{v1}\mathcal{R}_v^{l-1} \quad \text{for } l = 2, 3, \ldots, \quad (28)$$

where $\mathcal{R}_v$ is the Neuts' rate matrix [45]:

$$\mathcal{R}_v = \lambda_3(\lambda_3\mathbf{I} - S_v - \lambda_3\mathbf{1}_v\boldsymbol{\beta_v})^{-1}. \quad (29)$$

For the execution stage

$$\boldsymbol{\beta_v} = [1 \;\; 0], \; S_v = \begin{bmatrix} -2\mu_4 & 2\mu_4 \\ 0 & -2\mu_4 \end{bmatrix}. \quad (30)$$

Since Hyperledger Fabric has two sequential validation stages, that is $r = 2$ the initial vector is computed by

$$\pi_{v0} = 1 - U_v \quad (31)$$

$$\pi_{v1} = (1 - U_v)\boldsymbol{\beta_v}\mathcal{R}_v, \quad (32)$$

where again $U_v = -\lambda_3\boldsymbol{\beta_v}S_v^{-1}\mathbf{1}_v$.

Proposition 2 provides an understanding of the stationary state of the validation stage, i.e., how many transactions are in VSCC and MVCC, respectively, when the consortium blockchain system has reached the steady state.

4.4.2 Performance Analysis

For performance measures of the validation stage, the utilization (U_v), the mean number of transactions in the validation stage ($E[N_v]$) and the mean response time of the validation stage ($E[R_v]$) are obtained in the same way as *Properties 1*, *2* and *3*

$$E[N_v] = \pi_{v1}(\mathbf{I} - \mathcal{R}_v)^{-2}\mathbf{1}_v \quad (33)$$

$$E[R_v] = \frac{\pi_{v1}(\mathbf{I} - \mathcal{R}_v)^{-2}\mathbf{1}_v}{\lambda_3} \quad (34)$$

$$U_v = 1 - \pi_{v0}. \quad (35)$$

Again, we formulate the response time and the utilization metrics that we use to evaluate the blockchain system in the next section.

It is worth mentioning that our model does not take PoW blockchains into consideration. PoW takes so long that all other processes during consensus are irrelevant from the timing perspective. This is different when one considers blockchains that do not use PoW for the consensus mechanism. Thus, our model can well represent the main dynamics in the consortium blockchain consensus.

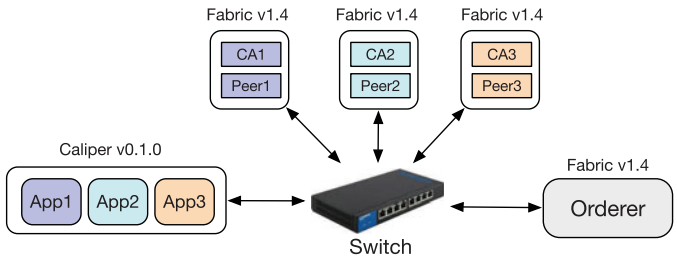


Fig. 6. Network setup for the empirical evaluation.

5 Analytical and Experiment Results

We apply our analytical model to the Hyperledger Fabric blockchain, a well-adopted consortium blockchain and study the obtained performance metrics. We use the model as proposed in Section 4. Our model is parameterized using data collected from the testbed. The analytical solutions are compared with the testbed measurements to validate the proposed model.

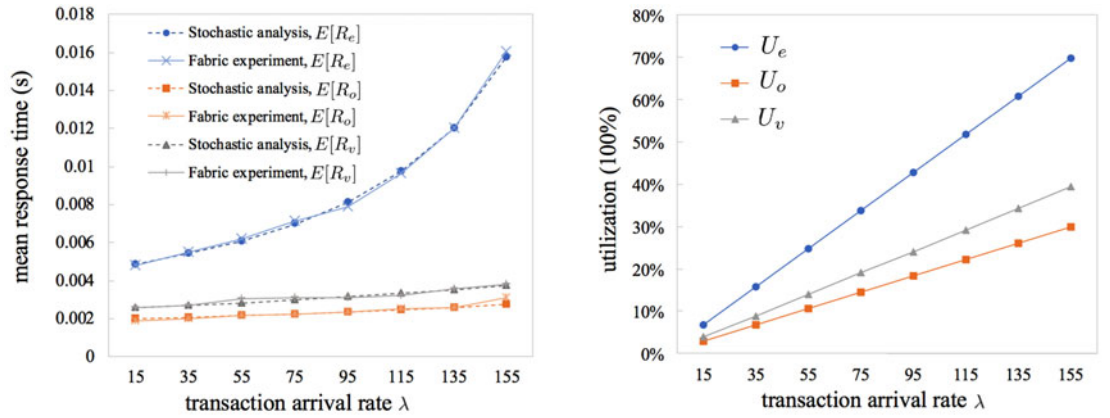
Experiment Network Setup. We have created a consortium blockchain testbed with one orderer and three organizations, where each organization has an endorsing peer and a certificate authority (CA). The network topology is shown in Fig. 6. Each blockchain node is launched as a Docker container and connected in a network using the Docker Swarm. We installed Hyperledger Fabric v1.4 on 4 physical machines (3 running as peers and 1 as the ordering service). Each peer runs on an independent machine with a *Peer* container and a *CA* container. The ordering service runs on a separate physical node in *solo* mode. Hyperledger Caliper v0.1.0[9] is installed on another machine to launch the client applications. The applications (App) interact with peer nodes using the Hyperledger Fabric Node.js SDK.[10] For performance testing, we leverage the *small bank* benchmark chaincode provided by the Caliper tool that simulates a real trading process in a banking application.

Each physical machine has 2 CPUs (Intel Xeon E5-2630 v3, 8 cores of 2.4 GHz) with 4×16 GB DDR4 RAM, and runs Ubuntu 16.04. All machines are connected with each other through a 1 Gbps switch (D-Link DGS-105). We use the Network Time Protocol (NTP) to synchronize the nodes so that we can measure the latency. The time taken to perform each transaction stage is measured by analyzing the caliper, peer and orderer log files. The simulation code and the raw data are published here.[11]

5.1 Mean Response Time

First, we consider the mean response time of the consortium blockchain. From *Properties 2* and *4*, we can compute the mean response time of the execution, ordering and validation stages ($E[R_e]$, $E[R_o]$ and $E[R_v]$). The comparison between the analytical solutions and the testbed measurements are shown in Fig. 7a. Please observe that the measurement values and

9. https://github.com/hyperledger/caliper
10. https://github.com/hyperledger/fabric-sdk-node
11. https://github.com/mth1haha/BlockchainQueueingNetwork



(a) Mean response time comparison between analytical solutions and testbed measurements.

(b) System utilization varing with the transaction arrival rate.

Fig. 7. Performance evaluations of Fabric blockchain at different transaction stages.

analytical estimates match very well, providing a strong experimental evidence for the proposed model.

Further, it is observed that when the transaction arrival rate increases, the *mean response time* at different transaction stages also increases monotonically. However, it is notable that the response time of the execution stage changes sharply when the transaction arrival rate is above 95 tps. This is because the endorsing peers are in charge of the transaction execution and this is a resource intensive job.

5.2 Block Size

We also evaluate the impact of the block size parameter *BatchSize* (B) on the performance of the consortium blockchain. Its effect in the high transaction traffic situation and the low traffic are compared in Fig. 8. Again, the analytical results match the experiment measurements as the dashed line and the histogram converge, which demonstrates that our proposed method is suitable to evaluate the performance of consortium blockchain protocols.

One can see that the mean response time in the ordering stage decreases with increasing block size. This is because when the block size parameter increases, in Hyperledger Fabric one block can contain more transactions. For a given transaction arrival rate λ, the number of blocks consequently decreases. The interval between blocks and the overhead of packing new blocks are reduced. Thus, the mean response time is reduced.

However, when the block size grows larger, the mean response time in the validation stage grows significantly. This is first attributed to the increased dispatch latency of larger blocks. The peers validate all the transactions in the newly generated block through VSCC (endorsement validation) and MVCC (read-set version validation) check. In MVCC, the read-set is computed by accessing the state database, and it takes more time to perform the bulk operation when the block is larger.

When transaction traffic is high, Fig. 8 shows an intense rise in the validation response time. But this upward trend is more moderate when the arrival rate is lower. From our results in both situations we see that the response time does not change too much in the execution stage. These results demonstrate that our proposed model can represent the properties of the consortium blockchain network well.

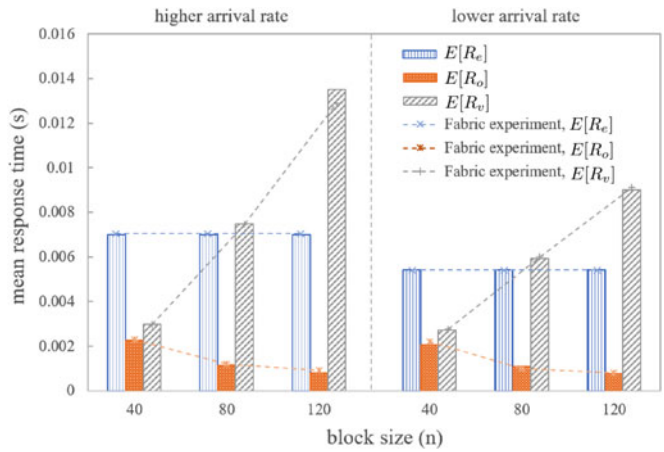


Fig. 8. The mean response time with different block size.

5.3 Utilization

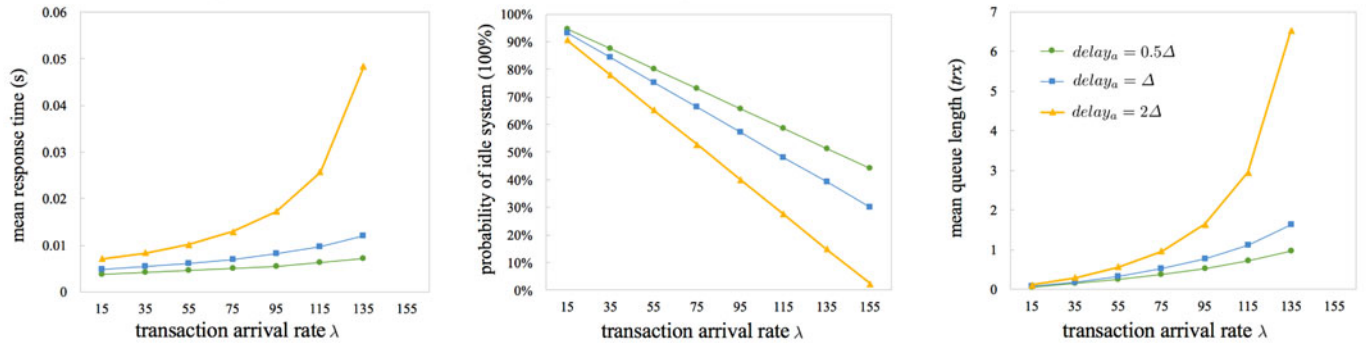
We conduct the utilization analysis of different transaction stages of the Hyperledger Fabric blockchain network using *Properties* 3 and 4 and the results are shown in Fig. 7b. Again, the execution stage is a performance bottleneck as the execution stage utilization is much higher than the utilization of the other two stages. The queueing system experiences 50 percent load at around $100tps$ and nearly 70 percent load when the system faces 155 transaction arrivals per second. Hence, in practical applications, more endorsing peers should be deployed to relieve the pressure of transaction execution. Make sure you set up your consortium blockchain with enough execution nodes. It is also recommended to choose suitable endorsement policies. Use k out of n policy with n at least $2 \times k$.

6 Security Analysis

In this section we use our model to study security aspects of consortium blockchain protocols.

6.1 Delaying Endorsement Message

In this section, we investigate the adverse impact of message delays on the security of the execution stage in consortium



(a) Deterioration of the mean response time caused by delaying endorsement messages

(b) Probability of an idle queueing system deterioration caused by delaying endorsement messages

(c) Deterioration of the mean queue length caused by delaying endorsement messages

Fig. 9. Performance deterioration of Fabric blockchain when the adversary is able to delay other players' endorsement messages up to 0.5Δ, Δ and 2Δ.

blockchain network. Using the proposed model, we analyze the *delay attack* (discussed in Section 3.5) in the consortium blockchain scenario and provide an upper bound of the damage an attacker can cause when it is capable of delaying the endorsement messages.

As was shown in Fig. 1, the blockchain network we take into account has three organizations and each organization provides a peer for transaction endorsement. The Hyperledger Fabric blockchain uses the endorsement policy *2/3 (Org_1, Org_2, Org_3)*, which means any two of the three organizations' endorsement are adequate. The peers are P_1, P_2 and P_3, and we assume that the adversary controls P_3 (that is a $\rho = 1/3$ fraction of all the peers). We can compute the probability of a transaction proposal receiving a good endorsement service as

$$\alpha = \frac{C_2^2}{C_3^2} = \frac{1}{\frac{3!}{2!}} = \frac{1}{3},$$

where C_n^k is the binomial coefficient, as above. Recall that in Section 3.5, the adversary A is able to delay the messages sent by honest nodes, and Δ is the mean blockchain network delay. In the endorsement procedure, we assume that the adversary is able to delay the endorsement messages of the honest players and tries to promote its own prioritized transaction to be processed. The second service phase μ_2 of the execution queue in Fig. 3 illustrates this situation. The deteriorated endorsing rate is

$$\mu_2 = \xi\ \mu_1,$$

where ξ is the *delay coefficient*. Based on our model, we can compute the ratio of compromised endorsements. We give the upper bound of the damage an attacker can cause when being capable of delaying the honest players' messages

$$\mathcal{D} = \frac{\frac{\widetilde{\pi}_{e1}(I-\widetilde{\mathcal{R}})^{-2}\mathbf{1}_e}{\lambda}}{\frac{\pi_{e1}(I-\mathcal{R})^{-2}\mathbf{1}_e}{\lambda}} \leqslant \frac{\widetilde{\pi}_{e1}(I-\widetilde{\mathcal{R}})^{-2}\mathbf{1}_e}{\pi_{e1}(I-\mathcal{R})^{-2}\mathbf{1}_e}, \tag{36}$$

where $\widetilde{\pi}_{e1}$ is the compromised initial probability and $\widetilde{\mathcal{R}}$ is Neuts' rate matrix degraded by delays.

We first evaluate how the Hyperledger Fabric blockchain network is affected when the adversary tries to delay the endorsement messages. Fig. 9 plots the results of the performance deterioration of the Fabric blockchain caused by delay attacks. From Fig. 9a one can observe that when the adversary can only delay the endorsement messages up to 0.5, the mean response time of Fabric's execution stage is still below $10\,\mathrm{ms}$. However, when the adversary is able to delay the endorsement messages up to, the mean response time is very near $10\,\mathrm{ms}$, which is unacceptable in practical implementations [34]. The situation is even worse as the delay increases up to 2Δ, as then the mean response time is around $50\,\mathrm{ms}$ and even several seconds when the transaction arrival rate is higher than $135\,tps$. As to the probability of observing an idle system, Fig. 9b reveals that the queueing system of the execution stage is more and more loaded as the delay time increases. When the adversary delays the endorsement messages up to 2Δ, the probability of finding the queueing system idle is around 2 percent with an arrival rate of $155\,tps$, which means the system is fully loaded. As to the mean queue length, from Fig. 9c we can see that when the attacker delays the endorsement message to 0.5Δ or Δ, the system is stable with less than on average 2 transactions waiting in the queue. But when the attacker delays the endorsement messages up to 2Δ, the mean queue length increases sharply with growing transaction arrival rate, which means the execution queue system is very busy.

In Fig. 10 we bound the damage an attacker could cause to the Hyperledger Fabric blockchain by delaying the endorsement messages of honest players. One can see from the figure that the mean response time continues to deteriorate with

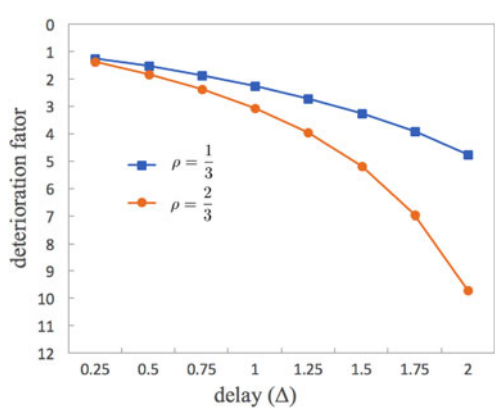


Fig. 10. Deterioration factor varying with different delay attack methods.

increasing delay time the attacker can use. When the adversary controls $\rho = \frac{1}{3}$ of all peers, the deterioration factor is up to 5, which means the response time increases by 5 times caused by the delay attack. However, if the adversary controls $\rho = \frac{2}{3}$ peers, the mean response time is even delayed to 10 times its original value.

6.2 Security of Ordering Service

The ordering service plays an important role in coordinating the operation of the Hyperledger Fabric blockchain network, since it can reconcile potentially conflicting versions of the ledger. Although Hyperledger Fabric claims that it supports a pluggable and modular ordering service, the first fault tolerant ordering service supported by Hyperledger Fabric is an adaptation of a Kafka distributed streaming platform. Therefore this implementation of the Hyperledger Fabric ordering service relies on the security of Kafka. Due to the security problems of the Kafka framework, e.g., any host on the network can initiate a malicious *Producer/Consumer* connection to the *Broker* to send compromised messages or pull private message data, from version 0.9.0.0 Kafka[12] supports the SASL mechanisms for authentication with Kerberos,[13] SCRAM-SHA-256 and OAuth 2.0 Bearer Token Usage.[14] Kafka also suggests that clients connect over SSL. So, when deploying the ordering service based on a Kafka cluster and Zookeeper, we need to consider the following issues:

- Generate a SSL key and certificate for each Kafka *Broker*.
- If the security key is saved on the server, it needs to be saved in ciphertext. The client key must be transmitted in ciphertext.
- It is better not to restart the Broker when a new user is added.
- Implementation of the *Topic* level permission and authorization model are needed.

In addition to Kafka, a Raft based ordering service is introduced in Fabric v1.4.1, which follows a "leader and follower" model. In each channel, a leader node is elected and its ordering decisions are replicated to the followers. Although the Raft based ordering service is crash fault tolerant, it is still vulnerable to the network split problem when more than half of the nodes are out of the current leader's control due to communication interruption caused by packet loss [36]. If the network splits, the Hyperledger Fabric blockchain network restarts a new leader election process. In Fabric, Raft ordering nodes identify each other using TLS pinning in order to prevent an attacker from impersonating a Raft node.

6.3 Validation Stage Security

In the validation stage of Hyperledger Fabric there is not much communication that would be subject to security concerns. After receiving the newly generated block distributed from the ordering service, each peer validates the block independently, but in a deterministic fashion, ensuring that ledgers remain consistent. If the adversary maliciously manipulates the validation process, the honest players will not accept its results. Therefore its validation results will not be recorded on the ledger, i.e., it does not affect the world state.

12. http://kafka.apache.org/documentation/
13. http://web.mit.edu/kerberos/
14. https://oauth.net/2/bearer-tokens/

7 Conclusion and Discussion

In this paper, we propose a framework for analyzing the consistency properties of consortium blockchain protocols. Based on a queueing network model, our method analyzes key properties of the transaction stage in consortium blockchain's consensus. We apply our model to the Hyperledger Fabric and study some relevant performance properties of the blockchain network. Our results show that the proposed model is not only efficient in analyzing the consistency properties, but also of theoretical significance for providing more rigorous analysis of new consortium blockchain schemes. From the security perspective, we analyze the delay attack proposed in [9] and provided an upper bound of the damage an attacker could cause to the consortium blockchain system.

The proposed model can be mapped to private blockchains whose consensus algorithm consists of multiple key processes. Though, the private blockchain is still a centralized network, this kind of blockchain is usually developed to be controlled by an organization. However, the proposed method is not suitable for analyzing public blockchains, since PoW (in Bitcoin and Ethereum) introduces a delay that dominates the consensus process and that is highly variable. PoW blockchains must be represented using different models. Our model considers the steady-state distribution of the consortium blockchain system and ignores the business idle time or the warm-up process. For a PoW blockchain the steady-state analysis is of lower relevance since fluctuations are very high and cause very important effects.

Acknowledgments

This work was partially supported by the Key-Area Research and Development Program of Guangdong Province (NO.2020B010164002), National Nature Science Foundation of China (No. 61801306), Science and Technology Development Fund of Macao S.A.R (FDCT) under number 0015/2019/AKP, Shenzhen Basic Research Program (No. JCYJ20170818153016513), Shenzhen Fundamental Research (No. JCYJ20180302145755311, JCYJ20180302145731531), Guangdong Special Fund for Science and Technology Development (No. 2019A050503001).

References

[1] C. Xu *et al.*, "Making big data open in edges: A resource-efficient blockchain-based approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 4, pp. 870–882, Apr. 2019.
[2] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *Proc. IEEE Int. Congr. Big Data*, 2017, pp. 557–564.
[3] P. Wang *et al.*, "Smart contract-based negotiation for adaptive QoS-aware service composition," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 6, pp. 1403–1420, Jun. 2019.
[4] J. Kang, R. Yu, X. Huang, S. Maharjan, Y. Zhang, and E. Hossain, "Enabling localized peer-to-peer electricity trading among plug-in hybrid electric vehicles using consortium blockchains," *IEEE Trans. Ind. Informat.*, vol. 13, no. 6, pp. 3154–3164, Dec. 2017.

[5] K. Croman *et al.*, "On scaling decentralized blockchains," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, 2016, pp. 106–125.
[6] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/en/bitcoin paper
[7] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-NG: A scalable blockchain protocol," in *Proc. 13th USENIX Symp. Netw. Syst. Des. Implementation*, 2016, pp. 45–59.
[8] Q.-L. Li, J.-Y. Ma, and Y.-X. Chang, "Blockchain queue theory," in *Proc. Int. Conf. Comput. Soc. Netw.*, 2018, pp. 25–40.
[9] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2017, pp. 643–673.
[10] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2015, pp. 281–310.
[11] N. Papadis, S. Borst, A. Walid, M. Grissa, and L. Tassiulas, "Stochastic models and wide-area network measurements for blockchain design and analysis," in *Proc. IEEE INFOCOM*, 2018, pp. 2546–2554.
[12] M. Zargham, Z. Zhang, and V. Preciado, "A state-space modeling framework for engineering blockchain-enabled economic systems," 2018, *arXiv: 1807.00955*.
[13] H. Sukhwani, J. M. Martínez, X. Chang, K. S. Trivedi, and A. Rindos, "Performance modeling of PBFT consensus process for permissioned blockchain network (hyperledger fabric)," in *Proc. IEEE 36th Symp. Reliable Distrib. Syst.*, 2017, pp. 253–255.
[14] H. Sukhwani, N. Wang, K. S. Trivedi, and A. Rindos, "Performance modeling of hyperledger fabric (permissioned blockchain network)," in *Proc. IEEE 17th Int. Symp. Netw. Comput. Appl.*, 2018, pp. 1–8.
[15] C. Decker and R. Wattenhofer, "Information propagation in the Bitcoin network," in *Proc. IEEE P2P Proc.*, 2013, pp. 1–10.
[16] C. Decker and R. Wattenhofer, "Bitcoin transaction malleability and MtGox," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2014, pp. 313–326.
[17] I. Zeljkovic and J. Garay, "Verifying transactions using out-of-band devices," U.S. Patent 9 003 519, Apr. 7, 2015.
[18] A. Kiayias and G. Panagiotakos, "Speed-security tradeoffs in blockchain protocols," *IACR Cryptol. ePrint Arch.*, vol. 2015, 2015, Art. no. 1019.
[19] A. Kiayias and G. Panagiotakos, "On trees, chains and fast transactions in the blockchain," in *Proc. Int. Conf. Cryptol. Inf. Secur. Latin America*, 2017, pp. 327–351.
[20] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 3–16.
[21] A. Gervais, G. O. Karame, V. Capkun, and S. Capkun, "Is Bitcoin a decentralized currency?," *IEEE Secur. Privacy*, vol. 12, no. 3, pp. 54–60, May/Jun. 2014.
[22] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," *Commun. ACM*, vol. 61, no. 7, pp. 95–102, 2018.
[23] L. Kiffer *et al.*, "A better method to analyze blockchain consistency," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 729–744.
[24] P. Wei, Q. Yuan, and Y. Zheng, "Security of the blockchain against long delay attack," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2018, pp. 250–275.
[25] M. Walck, K. Wang, and H. S. Kim, "TendrilStaller: Block delay attack in Bitcoin," in *Proc. IEEE Int. Conf. Blockchain*, 2019, pp. 1–9.
[26] D. I. Ilie, K. Karantias, and W. J. Knottenbelt, "Bitcoin cryptobounties for quantum capable adversaries," *IACR Cryptol. ePrint Arch.*, vol. 2020, 2020, Art. no. 186.
[27] Q.-L. Li, J.-Y. Ma, Y.-X. Chang, F.-Q. Ma, and H.-B. Yu, "Markov processes in blockchain systems," *Comput. Soc. Netw.*, vol. 6, no. 1, pp. 1–28, 2019.
[28] S. Kasahara and J. Kawahara, "Effect of Bitcoin fee on transaction-confirmation process," *J. Ind. Manage. Optim.*, vol. 15, no. 1, pp. 365–386, 2016.
[29] Y. Kawase and S. Kasahara, "Transaction-confirmation time for Bitcoin: A queueing analytical approach to blockchain mechanism," in *Proc. Int. Conf. Queueing Theory Netw. Appl.*, 2017, pp. 75–88.
[30] R. A. Memon, J. P. Li, and J. Ahmed, "Simulation model for blockchain systems using queuing theory," *Electronics*, vol. 8, no. 2, 2019, Art. no. 234.
[31] S. Ricci, E. Ferreira, D. S. Menasche, A. Ziviani, J. E. Souza, and A. B. Vieira, "Learning blockchain delays: A queueing theory approach," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 46, no. 3, pp. 122–125, 2019.
[32] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "BlockBench: A framework for analyzing private blockchains," in *Proc. ACM Int. Conf. Manage. Data*, 2017, pp. 1085–1100.
[33] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, "Untangling blockchain: A data processing view of blockchain systems," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 7, pp. 1366–1385, Jul. 2018.
[34] P. Thakkar, S. Nathan, and B. Viswanathan, "Performance benchmarking and optimizing hyperledger fabric blockchain platform," in *Proc. IEEE 26th Int. Symp. Model. Anal. Simul. Comput. Telecommun. Syst.*, 2018, pp. 264–276.
[35] E. Androulaki *et al.*, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.*, 2018, Art. no. 30.
[36] D. Huang, X. Ma, and S. Zhang, "Performance analysis of the raft consensus algorithm for private blockchains," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 50, no. 1, pp. 172–181, Jan. 2020.
[37] D. Ongaro and J. K. Ousterhout, "In search of an understandable consensus algorithm," in *Proc. USENIX Annu. Tech. Conf.*, 2014, pp. 305–319. [Online]. Available: https://www.usenix.org/conference/atc14/technical-sessions/presentation/ongaro
[38] J. Sousa, A. Bessani, and M. Vukolic, "A Byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform," in *Proc. 48th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, 2018, pp. 51–58.
[39] A. Bessani, J. Sousa, and E. E. P. Alchieri, "State machine replication for the masses with BFT-SMART," in *Proc. 44th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, 2014, pp. 355–362.
[40] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," *Future Gener. Comput. Syst.*, vol. 107, pp. 841–853, 2020.
[41] M. Vukolić, "Rethinking permissioned blockchains," in *Proc. ACM Workshop Blockchain Cryptocurrencies Contracts*, 2017, pp. 3–7.
[42] L. Lipsky, *Queueing Theory: A Linear Algebraic Approach*. Berlin, Germany: Springer, 2008.
[43] E. Gelenbe, "Product-form queueing networks with negative and positive customers," *J. Appl. Probability*, vol. 28, no. 3, pp. 656–663, 1991.
[44] P. Burke, "The output process of a stationary M/M/s queueing system," *Ann. Math. Statist.*, vol. 39, no. 4, pp. 1144–1152, 1968.
[45] M. F. Neuts, "Probability distributions of phase type," *Liber Amicorum Prof. Emeritus H. Florin*, Dept. Math., Belgium Univ. Louvain, 1975.
[46] W. J. Stewart, *Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling*. Princeton, NJ, USA: Princeton Univ. Press, 2009.
[47] J. D. Little and S. C. Graves, "Little's law," in *Building Intuition*. Berlin, Germany: Springer, 2008, pp. 81–100.
[48] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proc. USENIX Annu. Tech. Conf.*, 2014, pp. 305–319.
[49] C. H. Papadimitriou and P. C. Kanellakis, "On concurrency control by multiple versions," *ACM Trans. Database Syst.*, vol. 9, no. 1, pp. 89–99, 1984.

Tianhui Meng (Member, IEEE) received the BS degree in communication engineering from Tianjin University, Tianjin, China, in 2010, and the PhD degree in computer science from the Free University of Berlin, Berlin, Germany, in 2017. He is currently with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, as an assistant researcher. He serves as reviewer for a number of scientific journals and conferences. His research interests include blockchain, edge computing, cloud-edge collaboration, and network security.

Yubin Zhao (Member, IEEE) received the BS and MS degrees from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2007 and 2010, respectively, and the PhD degree in computer science from Freie Universität Berlin (FU Berlin), Berlin, Germany, in 2014. He is currently an associate professor with the School of Microelectronics Science and Technology, Sun Yat-Sen University, Zhuhai, P.R. China since November 2020. Before, he was an associate professor with Center for Cloud Computing, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China, since 2014. He serves as guest editor and reviewer for several scientific journals. His current research interests include mobile edge computing, wireless power transfer, and indoor localization.

Katinka Wolter (Member, IEEE) received the diploma degree and the PhD degree in computer science, both from the Technical University in Berlin, Berlin, Germany, in 1995 and 1999, respectively. She is a professor with the Department of Mathematics and Computer Science, Free University of Berlin. She leads a research group working on dependable systems. Her field of research is adaptive and resilient distributed computing systems using stochastic models and online versions of machine learning techniques.

Cheng-Zhong Xu (Fellow, IEEE) received the BSc and MSc degrees from Nanjing University, Nanjing, China, in 1986 and 1989, respectively, and the PhD degree from the University of Hong Kong, Hong Kong, in 1993, all in computer science and engineering. He is the dean with the Faculty of Science and Technology, University of Macau, a chair professor of computer and information science and the interim director with the Institute of Collaborative Innovation. He also holds a courtesy position as the director of the Center for Cloud Computing, Shenzhen Institutes of Advanced Technology (SIAT), Chinese Academy of Sciences. He was a professor of electrical and computer engineering with Wayne State University and the director of advanced computing and digital engineering with SIAT. His main research interests lie in parallel and distributed computing and cloud computing, in particular, with an emphasis on resource management for system's performance, reliability, availability, power efficiency, and security, and in big data and data-driven intelligence applications. The systems of particular interests include distributed systems and the Internet, servers and cloud datacenters, scalable parallel computers, and wireless embedded devices and mobile edge systems. He published two research monographs and more than 250 papers in journals and conference proceedings, including more than 50 in IEEE/ACM transactions; his publications received more than 9,000 citations with an H-index of 47. He was a Best Paper Nominee or Awardee of the 2013 IEEE High Performance Computer Architecture (HPCA), the 2013 ACM High Performance Distributed Computing (HPDC), IEEE Cluster'2015, ICPP'2015, GPC'2018, and UIC'2018. He also received more than 100 patents or PCT patents and spun off a business "Shenzhen Baidou Applied Technology" with dedication to location-based services and technologies. He received the most prestigious "President's awards for Excellence in Teaching" of Wayne State University in 2002. He serves or served on a number of journal editorial boards, including the *IEEE Transactions on Computers* (TC), *IEEE Transactions on Cloud Computing* (TCC), *IEEE Transactions on Parallel and Distributed Systems* (TPDS), *Journal of Parallel and Distributed Computing* (JPDC), *Science China: Information Science* and the *ZTE Communication*. He has been the chair of the *IEEE Technical Committee on Distributed Processing* (TCDP) since 2015.