Elastic Resource Allocation Against Imbalanced Transaction Assignments in Sharding-Based Permissioned Blockchains

Huawei Huang, *Member, IEEE*, Zhengyu Yue, Xiaowen Peng, Liuding He, Wuhui Chen, *Member, IEEE*, Hong-Ning Dai, *Senior Member, IEEE*, Zibin Zheng, *Senior Member, IEEE*, and Song Guo, *Fellow, IEEE*

Abstract—This article studies the PBFT-based sharded permissioned blockchain, which executes in either a local datacenter or a rented cloud platform. In such permissioned blockchain, the transaction (TX) assignment strategy could be malicious such that the network shards may possibly receive imbalanced transactions or even bursty-TX injection attacks. An imbalanced transaction assignment brings serious threats to the stability of the sharded blockchain. A stable sharded blockchain can ensure that each shard processes the arrived transactions timely. Since the system stability is closely related to the blockchain throughput, how to maintain a stable sharded blockchain becomes a challenge. To depict the transaction processing in each network shard, we adopt the Lyapunov Optimization framework. Exploiting *drift-plus-penalty* (DPP) technique, we then propose an adaptive resource-allocation algorithm, which can yield the near-optimal solution for each network shard while the shard queues can also be stably maintained. We also rigorously analyze the theoretical boundaries of both the system objective and the queue length of shards. The numerical results show that the proposed algorithm can achieve a better balance between resource consumption and queue stability than other baselines. We particularly evaluate two representative cases of bursty-TX injection attacks, i.e., the continued attacks against all network shards and the drastic attacks against a single network shard. The evaluation results show that the DPP-based algorithm can well alleviate the imbalanced TX assignment, and simultaneously maintain high throughput while consuming fewer resources than other baselines.

Index Terms—System stability, sharded blockchain, queueing theory, imbalanced transaction assignment

1 Introduction

Blockchain technologies have gained enormous attention in the past few years, leading to the blooming of countless decentralized applications (DApps), such as digital currencies [1], blockchain games [2], vehicles [3], internet of things [4], and medical treatment solutions [5], etc. Nevertheless, the scalability is still a barrier preventing the blockchain technology from widely being adopted. Taking Bitcoin as an example, the mining time of a block is around ten minutes, thus the Bitcoin network is only able to handle a very limited number of Transactions Per Second (TPS). Meanwhile, the constant block size also restricts the scalability of the blockchain. Therefore, it is hard for a blockchain network to achieve the similar throughput of the mainstream payment network like Visa (more than 5000 TPS) or Paypal (about 200 TPS).

Sharding [6], designed originally as a principle of classical database, is now being considered as a promising solution to improving the scalability of blockchains [7]. The key idea of sharding-based technique is to make a distribution over the blockchain nodes. Through sharding, the workload of transaction processing overhead can be amortized by different blockchain shards. The blockchain sharding network is analogous to a highway with multiple toll stations, each of which only needs to handle a subset of all blockchain transactions. This parallel manner effectively alleviates the congestion occurred in traditional blockchain networks. Two obvious benefits of sharding technique are reviewed as follows. First, sharding technique can ensure that transactions can be processed with much shorter congestion latency. Second, the improved transaction throughput will encourage more users and applications to engage in blockchain ecosystems. This change will make blockchain consensus more profitable and increase the security of the blockchain network.

- *Huawei Huang, Zhengyu Yue, Xiaowen Peng, Liuding He, Wuhui Chen, and Zibin Zheng are with the School of Computer Science and Engineering, Sun Yat-Sen University, Guangzhou 510275, China. E-mail: {huanghw28, chenwuh, zhzibin}@mail.sysu.edu.cn, {yuezhy6, pengxw3, held3}@mail2.sysu.edu.cn.*
- *Hong-Ning Dai is with the Department of Computing and Decision Sciences, Lingnan University, 999077, Hong Kong. E-mail: hndai@ieee.org.*
- *Song Guo is with the Department of Computing, Hong Kong Polytechnic University, 999077, Hong Kong. E-mail: song.guo@polyu.edu.hk.*

Manuscript received 25 Apr. 2021; revised 24 Sept. 2021; accepted 27 Dec. 2021. Date of publication 11 Jan. 2022; date of current version 7 Mar. 2022.
This work was supported in part by the National Key R&D Program of China under Grant 2020YFB1006005, in part by the National Natural Science Foundation of China under Grant 61902445, in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2019A1515011798, in part by Guangzhou Basic and Applied Basic Research Foundation under Grant 202102020613, in part by Pearl River Talent Recruitment Program under Grant 2019QN01X130, in part by CCF-Huawei Populus euphratica forest fund under Grant CCF-HuaweiBC2021004, Hong Kong RGC Research Impact Fund (RIF) with the Project No. R5060-19, General Research Fund (GRF) with the Project No. 152221/19E, 152203/20E, and 152244/21E, in part by the National Natural Science Foundation of China under Grant 61872310, and in part by Shenzhen Science and Technology Innovation Commission under Grant R2020A045.
(Corresponding author: Huawei Huang.)
Recommended for acceptance by D. Mohaisen.
Digital Object Identifier no. 10.1109/TPDS.2022.3141737

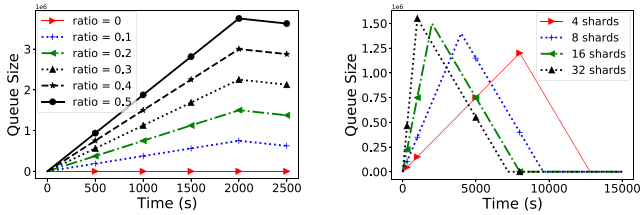
Motivation. In the sharded blockchain, the blockchain nodes are divided into a number of smaller committees [8], each is also called a network shard. In each committee, local consensus can be achieved with a set of designated transactions using a specified consensus protocol such as Practical Byzantine Fault Tolerance protocol (PBFT) [9]. Thus, the sharded blockchain can improve the blockchain TPS by exploiting the concurrent transaction processing in parallel shards. Several representative examples of sharded blockchain protocols include Elastico [8], Omniledger [10], RapaidChain [11], and Monoxide [12].

In this paper, we consider the Unspent Transaction Output (UTXO)-based transaction model. In the hash-based sharded blockchains [8], [10], [11], imbalanced transaction assignments in some committees can be induced either by the abnormal transaction execution [13] or by a low-quality even a malicious transaction assignment strategy [14]. Referring to Elastico [8], committees work on different disjoint sets of transactions assigned according to the committee ID. However, if a malicious transaction assignment strategy [14] aims to create a blockchain shard with low-speed transaction processing, the TPS of the entire blockchain can be degraded drastically. For example, reference [14] mentioned that a malicious transaction assignment may strategically put a large number of transactions to a single shard. Such a *single-shard flooding attack* [14] can be realized through manipulating the hash value of transactions.

Specifically, the last s bits of a transaction's hash specify which committee ID that the transaction should be assigned to. However, the attacker with enough UTXO addresses can create a huge number of malicious transactions towards the target shard. Let D be the ID of the target shard and $\mathcal{A}$ be the set of attacker's UTXO addresses. Hash function $\mathcal{H}(\cdot)$ adopts SHA-256, and $\mathcal{O}$ is the set of attacker's available addresses that can be generated arbitrarily by public-private key pair. Then, Eq. (1) in [14] shows how to generate malicious transactions in a sharding blockchain consisting of 2^s shards: *for* $\mathrm{TX}.in \in \mathcal{A}$

$$\textbf{while}\ \mathcal{H}(\mathcal{H}(\mathrm{TX}))\&\left(0^{256-s}\|1^s\right) \neq D\ \textbf{do}\ \mathrm{TX}.out \overset{@}{\leftarrow} \mathcal{O}, \tag{1}$$

where $\overset{@}{\leftarrow}$ denotes choosing a single element from the given set $\mathcal{O}$. In order to know the impact of such malicious transaction assignment. We implement the malicious TX-generation code given as Eq. (1) in Python and run the code using 12 operating-system threads on an AMD Ryzen 9 3900X 12-core processor. When the number of shards is 16, the TX-generation code can generate 8038886 TXs per second, among which 502011 TXs can be assigned to the target shard. Then, we conduct a group of simulations using 8 million Bitcoin TXs by injecting them into the TX-sharding system at the best fixed rates. Fig. 1a demonstrates the time-varying queue size of the target shard under attacking in a 16-shard blockchain system. We evaluate the performance of queue size while varying the the ratio of malicious TXs from 0 to 0.5. The TX rate is fixed to 4000 TPS, which is the best throughput of the 16-shard blockchain system. We observe that the queue size of the shard under attacking increases until all malicious TXs are injected. The results show that a larger ratio of malicious TXs implies a faster increase pace of the shard's queue size. Then, Fig. 1b



(a) Queue size vs malicious-TX ratio in a 16-shard blockchain.

(b) Queue size vs the # of shards with 20% malicious TXs.

Fig. 1. The motivation examples: the impact of single-shard flooding attacks [14] (also called *bursty-TX injection attacks* in this paper).

illustrates the target shard's queue size versus different numbers of shards when the ratio of malicious TXs is fixed to 20%. The results show that a larger number of shards are more vulnerable in single-shard flooding attacks. In this paper, we call such single-shard flooding attack the *bursty-TX injection* attack.

In summary, the bursty-TX injection can bring a large-size queue to a target blockchain shard, and thus cause a large congestion in that shard. Therefore, a critical issue is to maintain each shard in a stable status such that the imbalance of transaction's assignment can be quickly mitigated to guarantee low transaction confirmation latency in the shards suffering from the bursty-TX injection attack.

Challenges. We assume that the permissioned sharded blockchain executes in a cloud platform, which is either implemented in the local datacenter or rented from a popular cloud provider such as Alibaba cloud or Amazon cloud. In the context of such *permissioned* blockchain, the malicious transaction assignment strategy [14] may inject a large number of bursty transactions to some target shards. Furthermore, the resource budgets in a permissioned blockchain are much more limited than that in a permissionless blockchain. This is because the resources of a permissionless blockchain (e.g., Bitcoin blockchain) are provided by a wide range of miners all over the world. In contrast, the resources in a cloud-based permissioned blockchain are provided by the local datacenter or by the commercial cloud platform provider. Thus, a challenge is how to maintain shards stable in a resource-limited permissioned blockchain while taking the threat of the bursty-TX injection attack into account.

On the other hand, although the existing state-of-the-art blockchain sharding studies [8], [10], [11], [12] have proposed a number of transaction-processing solutions in the context of sharding-based blockchains, we still have not yet found any available solutions to solving the stability issue aforementioned. Therefore, a new strategy that can handle the imbalanced transaction assignments occurred in the shards of a permissioned blockchain is in an urgent need. To this end, we formulate the congestion of the permissioned sharded blockchain as the stability issue in a multi-queue system. In this system, some shards may congest if they are assigned a large number of transactions either by the abnormal transaction execution [13] or by the malicious transaction assignment strategy [14]. To alleviate the congestion occurred in some shards, we adopt the Lyapunov Optimization framework [15] to address the stability issue. Our idea emphasizes on how to efficiently allocate blockchain-network resources according to the observed status of each shard's Memory Pool (shorten as *mempool*), so as to

minimize the operation cost while simultaneously keeping the sharded blockchain stable.

Contributions. The contributions are stated as follows.

- In the PBFT-based permissioned blockchain, we study how to allocate budget-limited network resources to blockchain shards in an elastic manner, such that the transaction processing can be maintained stable in those blockchain shards, under the condition of imbalanced transaction assignments or even under the bursty-TX injection attacks. We formulate this problem using the Lyapunov Optimization framework. We then devise a *drift-plus-penalty* (DPP)-based algorithm striving for the near-optimal resource-allocation solution.
- We rigorously analyze the theoretical boundaries of both the system objective and the shard's queue length, while utilizing the proposed DPP algorithm.
- Finally, the numerical simulation results show that the proposed DPP algorithm can maintain a fine-balanced tradeoff between resource consumption and queue stability. In particular, the DPP-based algorithm can also well handle the bursty-TX injection attack in two representative scenarios.

The rest of this paper is organized as follows. Section 2 reviews the state-of-the-art studies. Section 3 describes the system model and problem formulation. Then, Section 4 depicts the proposed algorithm by exploiting the Lyapunov Optimization framework. Section 5 shows the evaluation results. Finally, Section 6 concludes this paper.

2 Related Work

Transaction Processing. Several studies investigate the transaction processing of blockchains using queueing theory. The representative studies are reviewed as follows. In [16], the authors focus on how to develop queueing theory of blockchain systems. They devised a Markovian batch-service queueing system with two different service stages, i.e., the mining process in the miners pool and the creation of a new block. Besides, they adopted the matrix-geometric solution to obtain stable condition of the system. Then, Ricci *et al.* [17] introduced a simple queueing-theory model to show the delay experienced by Bitcoin transactions. The proposed model associates the delay of transactions with both transaction fee and transaction value. Their result indicates that users typically experience a delay slightly larger than the residual lifetime of the total duration between block generations. Memon *et al.* [18] implemented the simulation of mining process in Blockchain-based systems using queuing theory. Kawase *et al.* [19] considered a queueing model with batch service and general input to understand the stochastic behavior of the transaction-confirmation process. In [20], the authors applied Jackson network model on the whole Bitcoin network where individual nodes operate as priority M/G/1 queueing systems. The usefulness of this model is demonstrated by efficiently computing the forking probability in Bitcoin blockchain. Although these previous works reviewed above adopted the traditional queueing theory to depict transaction's processing, they cannot offer available solutions to handling the stability problem for network shards when the bursty-TX injection attack occurs. In contrast, the approach we propose in this paper utilizes the Lyapunov optimization framework to resist the bursty-TX injection attack for the permissioned sharded blockchain.

Sharding Protocols. Sharding is first proposed by [21] to distribute data at global scale and support externally-consistent distributed transactions of distributed databases and the cloud infrastructure. This technology allows the network to partition into different parts. Motivated by this distributed design, several studies have discussed the intra-consensus protocols of blockchains under the sharding mechanism. For example, Luu *et al.* [8] first integrated sharding with the blockchain to achieve a *scale-out* system called Elastico, in which the capacity and throughput can be linearly proportional to the number of shards while maintaining decentralization and security. Elastico is viewed as the first candidate for a secure sharding protocol for permissionless blockchains, thereby scaling up the agreement throughput near linearly with the computational power of the network. It can also tolerate byzantine adversaries, by controlling up to one-forth computation capacity, in the partially synchronous network. Then, Kokoris *et al.* [10] presented a sharding protocol named OmniLedger, which ensures security and correctness by using a bias-resistant public-randomness protocol for choosing large and statistically representative shards to process transactions, and by introducing an efficient cross-shard commit protocol that atomically handles transactions affecting multiple shards. Zamani *et al.* [11] proposed RapidChain, the first sharding-based public blockchain protocol that is resilient to Byzantine faults from up to 1/3 of its participants, and achieves a complete sharding of the communication, computation, and storage overhead when processing transactions without assuming any trusted setup. Wang *et al.* [12] proposed Monoxide to maintain the simplicity of the blockchain system and enhance the capacity by duplicating equal and asynchronous zones, which work independently with minimal coordination. In [22], the authors presented the design of Chainspace which offers high scalability through sharding across blockchain nodes using a novel distributed atomic commit protocol named S-BAC, while offering high auditability. The most similar work to this paper is the sharding protocol proposed by Dang *et al.* [23]. In this work, aiming to realize high transaction throughput at scale, the authors proposed an efficient shard formation protocol, a trusted hardware-based performance-improving solution, as well as a general distributed transaction protocol. Comparing with [23], our study in this paper also strives for improving the sharding protocol for the PBFT-based permissioned blockchain. The difference is that the goal of [23] is to achieve high transaction throughput by proposing a shard formation protocol, while our work in this paper is towards stable transaction processing through efficiently allocating network resources to blockchain shards.

Resource Allocation. Several previous studies paid attention to the resource allocation of blockchain networks. For example, Jiao *et al.* [24] considered deploying edge computing services to support the mobile blockchain. The authors proposed an auction-based resource-allocation mechanism for the edge-computing service provider. Luong *et al.* [25] developed an optimal auction by exploiting the deep learning technique for the edge-resource allocation in the context

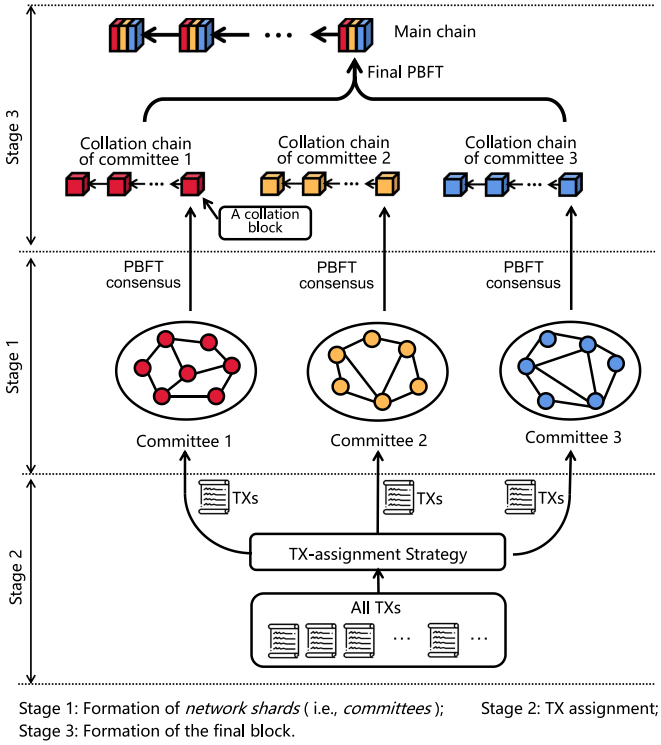


Fig. 2. A PBFT-based sharded permissioned blockchain, in which transactions are assigned to different committees (network shards) according to their hash values.

of mobile blockchain networks. Fang *et al.* [26] proposed a queue-based analytical model to solve the resources allocation problem in PoW-based blockchain networks. These previous works focus on the resource allocation for traditional blockchain networks. In contrast, the resource-allocation model we propose in this paper aims at the PBFT-based permissioned sharded blockchain by exploiting the queuing-based theory. More importantly, the proposed resource-allocation mechanism can particularly maintain the stability for each network shard.

3 System Model and Problem Formulation

3.1 Sharding-Based Permissioned Blockchain

Suppose that an enterprise needs to build a permissioned blockchain using the sharding technique. All blockchain nodes are executing in a local cloud platform. Those blockchain nodes are managed using sharding protocol, in which the local consensus is achieved by exploiting PBFT protocol. Thus, the natural goals of operating such a permissioned blockchain include: to keep the blockchain stable while processing transactions, and to consume the minimum resources while maintaining the local cloud platform. The cloud resources mainly include the computer power measured in the number of CPU cores, the network bandwidth, as well as the storage space provided by the cloud-platform virtual machines.

As shown in Fig. 2, the transaction-sharding protocol mainly includes the following stages: i) the blockchain network is divided into different committees (or called network shards); ii) each committee then independently processes different sets of assigned transactions in parallel to achieve a high throughput of transaction processing; and iii) collation blocks are aggregated to perform the final PBFT towards the formation of a new block on the main chain. Referring to the classic blockchain sharding protocol [8], the computing power is mainly used to perform the PoW-based committee formation, while the network bandwidth is exploited to run the PBFT consensus protocol for generating new blocks.

3.2 Blockchain Shards

In a sharding-based blockchain network, several groups of transactions are allocated to different network shards for processing. The blocks generated in each shard chain are called *collation blocks*, which are verified by all miners (also called *collators*) in this shard. The collators of each shard are selected by the block validator in the entire blockchain network through validator manager contract (VMC) [27], which is the core of a sharding mechanism.

Each shard holds a memory, i.e., the local mempool in each shard, where the arrived transactions are stored tentatively and waiting to be processed by the committee. When a new transaction is assigned to a network shard, it will be validated by the committee node that the transaction first arrives at. After validation, this transaction will be stored in the mempool and then be broadcast to other committee nodes; otherwise, the transaction will be rejected. At the beginning of each epoch, the miners in a network shard will select a set of transactions from the local mempool to generate a *collation block*. When a miner wins the mining in each epoch, it will broadcast the new collation block to its committee peers, which then validate the new block. Afterwards, the new block will be added to the shard's collation chain and all the transactions contained in this collation block will be removed from the local mempool of this shard.

3.3 Arrived Transactions in Each Network Shard

Under the same UTXO-based transaction model presented in [8], [10], [11], we consider that committees (network shards) work on disjoint sets of transactions. In each network shard, since the propagation time for spreading new transactions is much shorter than the time spending on achieving consensus towards the collation block, the propagation time of new transactions is negligible in our system model. Thus, new transactions are viewed as that they arrive at all committee nodes within a network shard simultaneously. After transaction's propagation, all committee nodes in this shard share the same set of transactions. That is, they hold the identical view of the local mempool. Therefore, the mempool can be viewed as a single-server queue which stores the assigned transactions waiting to be processed by the multiple nodes in the shard. In this queueing model, transactions arrive randomly following a Poisson distribution. When being packed to a collation block, the transactions contained in the block will be removed from the mempool. We call this removing action the transaction's dequeueing. In each epoch, all committee members generate a new collation block when they reach an agreement based on the PBFT consensus protocol. Every shard's queue represents the condition of the local mempool. Based on the queueing model aforementioned, a blockchain sharding network can be viewed as a multi-queue system shown in Fig. 3.

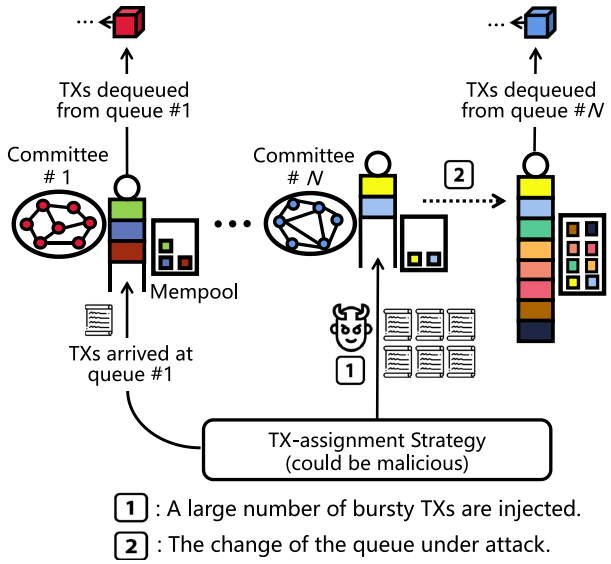


Fig. 3. Multi-queue model and the bursty-TX injection attack in the sharded blockchain.

3.4 Threat Model of Bursty-TX Injection Attack

Blockchain is typically implemented using conventional hardware, software and networks. Even a theoretically secure blockchain sharding protocol can be vulnerable to various attacks. Similar to the single-shard flooding attack presented in [14], we consider that the hash-based transaction assignment strategy could be malicious. In most hash-based sharded blockchains [8], [10], [11], the several ending bits of a transaction decide which network shard to place. Through manipulating the transaction's hash, the malicious transaction assignment strategy can inject a large-volume bursty transactions in a target shard at a specified epoch. The large amount of transactions injected to a target shard can cause the effect of denial-of-service (DoS) attack [28] in the target. When such an attack occurs, the transactions submitted by blockchain users may congest in the shard's mempool. Thus, the service quality of the blockchain system degrades drastically, because the user's transactions suffer from large confirmation latency in the shard under attack. Especially in the context of transaction-sharding protocol like Elastico [8], the target shard suffered from such bursty-TX injection attack can delay the generation of the local collation block, and then postpone the creation of the final block in the main chain.

3.5 Problem Formulation

We consider that the permissioned sharded blockchain studied in this paper runs in a time-slotted environment. All timeslots are indexed by $t \in \{0, 1, 2, 3, \ldots\}$, and the length of a timeslot is longer than the time of forming a collation block. We summarize important symbols and notations in Table 1. Let $I = \{1, 2, 3, \ldots, N\}$ be the set of mempool queues, thus each queue is indexed by $i \in I$. Then, K denotes the number of all types of aforementioned resources that can be allocated to the permissioned sharded blockchain.

We then use a vector $\mathbf{p}_i(t) \triangleq [p_i^1(t), p_i^2(t), p_i^3(t), \ldots, p_i^K(t)]$ to represent the total resources that can be allocated to shard $i \in I$ at timeslot t. Note that, $p_i^k(t)$, $k \in [K]$, denotes the amount of the k^{th} resource invested on the shard $i \in I$ to generate a collation block in timeslot $t \in T$. Each type of the k^{th} resource has a maximum budget denoted by $P_{\max}^k$. In order to represent the significance of each type of resources, we devise a weight vector $\mathbf{w} \triangleq [w_1, w_2, w_3, \ldots, w_K]$, where w_k indicates the weight of the k^{th} resource.

TABLE 1
Symbols and Notations

I	the set of queues, $\lvert I\rvert (= N)$ represents the size of I
T	the set of all timeslots, $t \in T$
K	the # of all types of resources, $k \in \{1, 2, 3, \ldots, K\}$
$\mathbf{p}_i(t)$	the vector of resources allocated for shard $i \in I$ at t
$p_i^k(t)$	the kth resource allocated to shard $i \in I$ at timeslot t
$P_{\max}^k$	the budget of the kth resource
w_k	the weight of the kth resource
$\mathbf{Q}(t)$	vector of actual queues, $\mathbf{Q}(t) = [Q_1(t), Q_2(t), \ldots, Q_N(t)]$
$Q_i(t)$	the queue length of shard $i \in I$ at timeslot t
$A_i(t)$	the arrival transactions of queue $i \in I$ at timeslot t
$B_i(t)$	the dequeued transactions of queue $i \in I$ at timeslot t
R	the reward to measure each unit of dequeued data
V	the parameter measuring the weight of penalty
α	the parameter reverse to the consensus difficulty
$L(\mathbf{Q}(t))$	the Lyapunov function of $\mathbf{Q}(t)$
$\Delta(\mathbf{Q}(t))$	the Lyapunov drift, $\Delta(\mathbf{Q}(t)) = L(\mathbf{Q}(t+1)) - L(\mathbf{Q}(t))$
$\mathbf{Z}(t)$	vector of virtual queues, $\mathbf{Z}(t) = [Z_1(t), Z_2(t), \ldots, Z_K(t)]$
$z_k(t)$	the increment arrived in virtual queue k at timeslot t
$\mathbf{\Theta}(t)$	the concatenated vector $\mathbf{\Theta}(t) = [\mathbf{Q}(t), \mathbf{Z}(t)]$

In every timeslot $t \in T$, we assume that $B_i(t)$ is the amount of transactions (TXs) processed by shard $i \in I$. The value of $B_i(t)$ is closely associated with the allocated resource $\mathbf{p}_i(t)$. Referring to [26], the data amount processed by shard $i \in I$ at timeslot t is calculated as follows:

$$B_i(t) = \sum_{k=1}^{K} \left[w_k p_i^k(t)\right]^{\alpha}, \forall i \in I, t \in T, \tag{2}$$

where $\alpha \in [0, 1]$ is a normalized parameter associated with the consensus speed of the PBFT protocol. In reality, α could be the normalized degree of network connectivity in a peer-to-peer blockchain network. The larger α is, the easier a collation block is generated with the given same amount of resources.

At the beginning of a timeslot, a set of TXs will be assigned to each network shard. The TXs arrived in shard $i \in I$ at timeslot $t \in T$ are denoted by $A_i(t)$. Afterwards, the committee nodes in the shard will choose a subset of TXs from the local mempool to participate in the consensus process. The verified transactions will be packed into a new collation block. Then, we use $Q_i(t)$ to represent the queue length of shard i in the beginning of timeslot t. Thus, the queue-length evolution can be expressed as

$$Q_i(t+1) = \max\{Q_i(t) - B_i(t) + A_i(t), 0\}, \forall i \in I, t \in T. \tag{3}$$

On the other hand, to represent the investment cost on the consensus of the sharded permissioned blockchain, we define $c_i(t)$ as the numerical resource consumption by the

network shard $i \in I$ in timeslot $t \in T$

$$c_i(t) = \sum_{k=1}^{K} p_i^k(t), \forall i \in I, t \in T. \quad (4)$$

Objectives of Blockchain-Network Operator. From the viewpoint of the operator of a permissioned sharded blockchain, the objectives are twofold: (i) to pursue a high speed of TX processing, and (ii) to lower the operating cost in terms of resource consumption during the TX processing. Those two objectives seem conflict with each other. The operator has to maximize the payoff of transaction processing while using a limited amount of resources. To integrate those two conflict goals together, we devise a penalty function $pen(t)$, which is calculated by the total resource consumption defined in Eq. (4) minus the TX-processing payoff. To measure the payoff while processing a unit of data dequeued from $B_i(t)$, we also define a constant R as the TX-processing reward parameter. Here, we clarify the simplification to measure each transaction equally as the reward parameter R when calculating the payoff of dequeued transactions. This is because the unique value of each transaction is blind to the proposed resource-allocation method. Only the number of transactions in each shard can be observed by the proposed method taking the transaction's privacy into account in the context of permissioned blockchain. Therefore, our system model treats every transaction equally and calculates the payoff of transaction's processing using the constant reward parameter R. In practice, R can be configured according to the empirical preference of the blockchain network operator, when he/she defines the relative weights of two penalty terms $c_i(t)$ and $B_i(t)$. Then, the penalty function can be written as follows:

$$\begin{aligned} pen(t) &= \sum_{i=1}^{N} [c_i(t) - B_i(t) \cdot R] \\ &= \sum_{i=1}^{N} \sum_{k=1}^{K} p_i^k(t) - R \cdot \sum_{i=1}^{N} \sum_{k=1}^{K} \left[w_k p_i^k(t)\right]^{\alpha}, \forall t \in T. \end{aligned} \quad (5)$$

Recall that a sharded permissioned blockchain network can be viewed as a multi-queue system as shown in Fig. 3. Besides the objective to minimize the penalty defined in Eq. (5), the network operator also intends to guarantee that each shard queue is in a stable condition during a long period, even under the large-volume bursty-TX injection attack. Using the notion of *queue stability* defined in [15], the multi-queue system is strongly stable if it satisfies

$$\lim_{t \to \infty} \frac{1}{t} \cdot \frac{1}{N} \sum_{\tau=0}^{t-1} \sum_{i=1}^{N} \mathbb{E}\{Q_i(\tau)\} < \infty. \quad (6)$$

The basic idea to prove the Eq. (6) is that in a multi-queue system, the assigned TXs in each queue will not be accumulated to infinity in the long run. Therefore, to keep the multi-queue system stable, each shard needs to process the TXs arrived in the queue with sufficient allocated resources. In this way, every TX assigned to this network shard can be processed in time. With the objective function and the constraints described above, we propose the following resource allocation problem for the sharded permissioned blockchain network

$$\begin{aligned} \textbf{min} \quad & \overline{pen} \\ \textbf{s.t.} \quad & \sum_{i=1}^{N} p_i^k(t) \le P_{\max}^k, \forall k \in [K], t \in T. \\ & \text{Queue stability depicted by Ineq. (6).} \\ \textbf{Var:} \quad & \mathbf{p}_i(t), \ \forall i \in I, t \in T. \end{aligned} \quad (7)$$

In next section, we design an adaptive resource allocation algorithm that can find a near-optimal solution to problem (7).

4 Dynamic Resource-Allocation Algorithm

To address the proposed resource-allocation problem (7) in the context of maintaining the stability of the multi-queue sharded permissioned blockchain, we design a dynamic resource-allocation algorithm using the stochastic Lyapunov optimization technique [15]. In practice, the proposed algorithm can execute in the same manner of the sharding protocol.

4.1 Algorithm Design

Since the PBFT-based sharded blockchain is viewed as a queueing system with $N > 0$ queues, we define $\mathbf{Q}(t) = [Q_1(t), Q_2(t), \ldots, Q_N(t)]$ as the vector of queue backlogs of all network shards. To quantitatively measure the *size* of the vector $\mathbf{Q}(t)$, a *quadratic Lyapunov function* $L(\mathbf{Q}(t))$ is defined as follows:

$$L(\mathbf{Q}(t)) \triangleq \frac{1}{2} \sum_{i=1}^{N} Q_i(t)^2, \forall t \in T. \quad (8)$$

We then define a *one-timeslot conditional Lyapunov drift* $\Delta(\mathbf{Q}(t))$ as follows:

$$\Delta(\mathbf{Q}(t)) \triangleq L(\mathbf{Q}(t+1)) - L(\mathbf{Q}(t)), \forall t \in T. \quad (9)$$

This drift is in fact the change in the Lyapunov function (8) over one timeslot. Suppose that the current queue state in timeslot t is $\mathbf{Q}(t)$, we have the following lemma.

Lemma 1 (Lyapunov Drift). *Given the quadratic Lyapunov function (8), and assuming $L(\mathbf{Q}(0)) < \infty$, for arbitrary non-negative constants $B > 0$ and $\epsilon > 0$, the following drift condition holds:*

$$\lim_{t \to \infty} \frac{1}{t} \cdot \sum_{\tau=0}^{t-1} \sum_{i=1}^{N} \mathbb{E}\{Q_i(\tau)\} < B/\epsilon. \quad (10)$$

Lemma 1 tells that if the drift condition Eq. (10) holds with $\epsilon > 0$, then all queues are strongly stable with the queue backlog bounded by B/ϵ. The proof of lemma 1 follows the same routine shown in [15], we omit the proof detail because of the space limitation.

Next, we need to handle the first constraint of problem (7) to ensure that the total consumption of the kth resource should be restricted by the budget $P_{\max}^k$. To solve this issue, we transform the inequality constraints into a queue

stability problem. By defining virtual queues $Z_k(t)$ for all $k \in [K]$, the update equation of $Z_k(t)$ is written as

$$Z_k(t+1) = \max\{Z_k(t) + z_k(t), 0\}, \forall k \in [K], t \in T, \tag{11}$$

where $z_k(t) = \sum_{i=1}^{N} p_i^k(t) - P_{\max}^k, \forall k \in [K], t \in T$. The initial length of the virtual queue $Z_k(0), \forall k \in [K]$ is 0.

Insight. Eq. (11) indicates that $Z_k(t+1) - Z_k(t) \geq z_k(t)$. By summing the inequality above over timeslots $t = 0, 1, \ldots, T-1$, and dividing both sides by T, we have $\frac{Z_k(T) - Z_k(0)}{T} \geq \frac{1}{T}\sum_{t=0}^{T-1} z_k(t)$. With $Z_k(0) = 0$, take expectation on both sides and let $T \to \infty$, the result is $\limsup_{T\to\infty} \frac{\mathbb{E}\{Z_k(T)\}}{T} \geq \limsup_{T\to\infty} \overline{z_k}(t)$, where $\overline{z_k}(t)$ is the time-average expectation of $z_k(t)$. If virtual queue $Z_k(t)$ is mean-rate stable, we get $\limsup_{T\to\infty} \overline{z_k}(t) \leq 0$, which implies that the constraint of resource restriction in problem (7) is satisfied.

With the objective to maintain both the actual and virtual queues, we combine both of them to devise a concatenated vector $\mathbf{\Theta}(t) = [\mathbf{Q}(t), \mathbf{Z}(t)]$, which can be updated using both Eqs. (3) and (11). Then, the Lyapunov function of $\mathbf{\Theta}(t)$ is defined as

$$L(\mathbf{\Theta}(t)) \triangleq \frac{1}{2}\sum_{i=1}^{N} Q_i(t)^2 + \frac{1}{2}\sum_{k=1}^{K} Z_k(t)^2, \ t \in T. \tag{12}$$

If we successfully make $L(\mathbf{\Theta}(t))$ maintain a small value for each timeslot t, both the actual queues $\mathbf{Q}(t)$ and virtual queues $\mathbf{Z}(t)$ can be "squeezed" to a small space of queue backlogs. In the following, we present how to achieve such the goal using the technique of *Lyapunov drift* [15].

Lemma 2 (Optimality Over Resource Allocation). *Let $\mathbf{p}^* = [\mathbf{p}_1^*, \mathbf{p}_2^*, \mathbf{p}_3^*, \ldots, \mathbf{p}_N^*]$ be the optimal resource allocation solution to problem (7). Suppose the sharded blockchain system satisfies the boundedness assumptions[15] and the law of large numbers. If the problem is feasible, then for any $\delta > 0$, there exists a stationary randomized policy that makes the resource allocation depend only on the state of multi-queue system, such that $pen(t) < pen^*(t) + \delta$ and $A_i(t) < B_i(t)$, where*

$$pen^*(t) \triangleq \sum_{i=1}^{N} [c_i^*(t) - B_i^*(t) \cdot R], \forall t \in T. \tag{13}$$

The proof of Lemma 2 is given in [15]. The next step is to minimize a concatenated vector $\Delta(\mathbf{\Theta}(t))$, which is defined as the *Lyapunov drift* of $\mathbf{\Theta}(t)$, and $\Delta(\mathbf{\Theta}(t)) = L(\mathbf{\Theta}(t+1)) - L(\mathbf{\Theta}(t))$. It should be noticed that minimizing the *Lyapunov drift* of $\mathbf{\Theta}(t)$ would enforce queues towards a lower congestion. However, only focusing on the drift part may incur a large penalty $pen(t)$. Thus, we then propose a dynamic resource allocation algorithm based on the *drift-plus-penalty* (DPP) technique [15] to maintain the stability of a multi-queue blockchain system, and to minimize $pen(t)$ simultaneously. To achieve the goal, we integrate the penalty function into the Lyapunov drift $\Delta(\mathbf{\Theta}(t))$. In every timeslot t, we try to minimize the following *drift-plus-penalty*:

$$\begin{aligned}&\Delta(\mathbf{\Theta}(t)) + V \cdot pen(t)\\ &\quad = \Delta(\mathbf{\Theta}(t)) + V \cdot \sum_{i=1}^{N} [c_i(t) - B_i(t) \cdot R], \forall t \in T,\end{aligned} \tag{14}$$

where V is a weight parameter representing how much we emphasize on minimizing the penalty $pen(t)$. When $V = 0$, the objective is to minimize the drift alone. Thus, to provide guarantees on $pen(t)$, we consider $V > 0$, which indicates a joint optimization taking both the system stability and the resource consumption into account.

Lemma 3 (Drift Boundary). *The boundary of the drift-plus-penalty expression shown in Eq. (14) satisfies*

$$\begin{aligned}&\Delta(\mathbf{\Theta}(t)) + V \cdot \sum_{i=1}^{N} [c_i(t) - B_i(t) \cdot R]\\ &\leq B + V \cdot \sum_{i=1}^{N} [c_i(t) - B_i(t) \cdot R]\\ &\quad + \sum_{i=1}^{N} Q_i(t)[A_i(t) - B_i(t)]\\ &\quad + \sum_{k=1}^{K} Z_k(t) z_k(t), \forall t \in T,\end{aligned} \tag{15}$$

where the positive constant B exists and is bounded by:

$$\begin{aligned}B \geq\ &\frac{1}{2}\sum_{i=1}^{N} \mathbb{E}\Big\{A_i(t)^2 + B_i(t)^2 | \mathbf{\Theta}(t)\Big\}\\ &+ \frac{1}{2}\sum_{k=1}^{K} \mathbb{E}\Big\{z_k(t)^2 | \mathbf{\Theta}(t)\Big\}\\ &- \sum_{i=1}^{N} \mathbb{E}\{\min\{Q_i(t), B_i(t)\} \cdot A_i(t) | \mathbf{\Theta}(t)\}.\end{aligned}$$

Proof. Exploiting the definition of *Lyapunov drift*, we have

$$\begin{aligned}\Delta(\mathbf{\Theta}(t)) &= L(\mathbf{\Theta}(t+1)) - L(\mathbf{\Theta}(t))\\ &= \frac{1}{2}\sum_{i=1}^{N}\Big[Q_i(t+1)^2 - Q_i(t)^2\Big]\\ &\quad + \frac{1}{2}\sum_{k=1}^{K}\Big[Z_k(t+1)^2 - Z_k(t)^2\Big]\\ &\overset{(a)}{\leq} \frac{1}{2}\sum_{i=1}^{N}\Big\{[Q_i(t) - B_i(t) + A_i(t)]^2 - Q_i(t)^2\Big\}\\ &\quad + \frac{1}{2}\sum_{k=1}^{K}\Big\{[Z_k(t) + z_k(t)]^2 - Z_k(t)^2\Big\}\\ &= \sum_{i=1}^{N}\Big\{\frac{1}{2}[A_i(t) - B_i(t)]^2 + Q_i(t)[A_i(t) - B_i(t)]\Big\}\\ &\quad + \sum_{k=1}^{K}\Big\{\frac{1}{2}z_k(t)^2 + Z_k(t) z_k(t)\Big\}\\ &\leq B + \sum_{i=1}^{N} Q_i(t)[A_i(t) - B_i(t)]\\ &\quad + \sum_{k=1}^{K} Z_k(t) z_k(t), \forall t \in T,\end{aligned}$$

where the inequation (a) follows from Eqs. (3) and (11). When adding $V \cdot \sum_{i=1}^{N}[c_i(t) - B_i(t) \cdot R]$ on both sides of the above derivation, the Eq. (15) holds. □

Because the arrival new TXs $A_i(t)$ in timeslot t is independent of $\mathbf{\Theta}(t)$, minimizing the right-hand-side (RHS) of

Eq. (15) for each timeslot t is equalized to solving the original optimization problem (7). Thus, we expand the RHS of Eq. (15) and rearrange the objective as

$$\begin{aligned} \textbf{min} \quad & \Psi(\mathbf{p}_i(t)) \\ \textbf{Var}: \quad & \mathbf{p}_i(t),\ \forall i \in I, t \in T, \end{aligned} \tag{16}$$

where $\Psi(\mathbf{p}_i(t)) = \sum_{i=1}^{N}\sum_{k=1}^{K}(Z_k(t)+V)p_i^k(t) - (Q_i(t)+VR)[w_k p_i^k(t)]^{\alpha}$.

Note that, problem (16) is a linear programming and $\Psi(\mathbf{p}_i(t))$ is a convex function, through partially differentiating $\Psi(\mathbf{p}_i(t))$ by $p_i^k(t)$ and rearranging terms, then we get

$$\frac{\partial \Psi(\mathbf{p}_i(t))}{\partial p_i^k(t)} = Z_k(t) + V - \alpha(w_k)^{\alpha}(Q_i(t)+VR)[p_i^k(t)]^{\alpha-1}. \tag{17}$$

From Eq. (17) we can find a real-numbered valley point $p_i^k(t) = \sqrt[\alpha]{[\alpha-1]\frac{Z_k(t)+V}{\alpha(w_k)^{\alpha}[Q_i(t)+VR]}}$ $(\forall i \in I, \forall k \in [K])$, which is the optimal resource allocation solution to the optimization problem (7). Then, given Eq. (17), the *DPP*-based resource-allocation algorithms are presented as Algorithms 1 and 2.

Algorithm 1. Drift-Plus-Penalty Resource Allocation

Input: T, V, R, α
Output: $\mathbf{p}_i(t), \forall i \in I, t \in T$
1 $\mathbf{Q}(0) \leftarrow \emptyset, \mathbf{Z}(0) \leftarrow \emptyset$;
2 **for** $\forall t \in T$ **do**
3 Invoking Algorithm 2 $(t, \mathbf{Q}(t), \mathbf{Z}(t), V, R, \alpha)$ to get the optimal resource allocation $\mathbf{p}_i(t), \forall i \in I$.
4 Update the actual and virtual queues $\mathbf{Q}(t)$ and $\mathbf{Z}(t)$ according to Eqs. (3) and (11), respectively.

Algorithm 2. Resource Allocation Per Timeslot

Input: $t, \mathbf{Q}(t), \mathbf{Z}(t), V, R, \alpha$
Output: $\mathbf{p}_i(t), i \in I$
1 **for** $\forall i \in I, \forall k \in [K]$ **do**
2 $p_i^k(t) \leftarrow \sqrt[\alpha]{[\alpha-1]\frac{Z_k(t)+V}{\alpha(w_k)^{\alpha}[Q_i(t)+VR]}}$.

4.2 Algorithm Analysis

In this section, we analyze the performance guarantee of the proposed DPP-based algorithm.

4.2.1 Upper Bound of System Objective

First, Theorem 1 tells that the proposed DPP algorithm can ensure the system objective guaranteed within an $O(1/V)$ distance to that of the optimal solution to problem (7).

Theorem 1. *Suppose that problem (7) is feasible and there exists an optimal resource allocation solution which can obtain an optimal value pen^*, and $L(\mathbf{\Theta}(0)) < \infty$. For any $V > 0$, the time-average penalty yielded by the proposed Algorithm 1 satisfies*

$$\limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{i=1}^{N}[c_i(\tau) - B_i(\tau)\cdot R] \le pen^* + \frac{B}{V}, \tag{18}$$

where B is depicted in Lemma 3.

Proof. Integrating the result of Lemma 2 into the RHS of Eq. (15), and let $\delta \to 0$, we have

$$\begin{aligned} &\Delta(\mathbf{\Theta}(\tau)) + Vpen(\tau) \\ &\le B + V\cdot pen^* + \sum_{i=1}^{N} Q_i(\tau)[A_i(\tau) - B_i(\tau)] \\ &\quad + \sum_{k=1}^{K} Z_k(\tau) z_k(\tau), \end{aligned} \tag{19}$$

where $\Delta(\mathbf{\Theta}(\tau)) = L(\mathbf{\Theta}(\tau+1)) - L(\mathbf{\Theta}(\tau))$.

Taking expectations on both sides, and summing the Eq. (19) over $\tau \in \{0, 1, \ldots, t-1\}$, it yields

$$\begin{aligned} &\mathbb{E}\{L(\mathbf{\Theta}(t))\} - \mathbb{E}\{L(\mathbf{\Theta}(0))\} + V\sum_{\tau=0}^{t-1} pen(\tau) \\ &\le B\cdot t + Vt\cdot pen^* + \mathbb{E}\left\{\sum_{\tau=0}^{t-1}\sum_{i=1}^{N} Q_i(\tau)[A_i(\tau) - B_i(\tau)]\right\} \\ &\quad + \mathbb{E}\left\{\sum_{\tau=0}^{t-1}\sum_{k=1}^{K} Z_k(\tau) z_k(\tau)\right\}. \end{aligned} \tag{20}$$

Eq. (3) implies that $Q_i(\tau) \ge 0$. Referring to Theorem 4.5 of [15] and the requirement of system stability, we have $\mathbb{E}\{A_i(\tau)\} - \mathbb{E}\{B_i(\tau)\} \le 0$. Eq. (11) secures $Z_k(\tau) \ge 0$. Since $z_k(t)$ denotes the inequality constraint in problem (7), we have $\mathbb{E}\{z_k(\tau)\} \le 0$. The Lyapunov function $L(\mathbf{\Theta}(\tau)) \ge 0$ is due to Eq. (12). With those inequalities illustrated above, rearranging the terms of Eq. (20) and dividing both sides by $V\cdot t$, we have

$$\frac{1}{t}\sum_{\tau=0}^{t-1} pen(\tau) \le pen^* + \frac{B}{V} + \frac{\mathbb{E}\{L(\mathbf{\Theta}(0))\}}{V\cdot t}. \tag{21}$$

Then, taking a $\limsup_{t\to\infty}$ and invoking Eq. (5), the conclusion of Theorem 1 holds. □

4.2.2 Upper Bound of the Queue Length of Shards

In this part, we analyze the congestion performance (measured in *queue length*) of network shards when adopting the proposed Algorithm 1. First, we give Assumption 1.

Assumption 1 (Slater Condition [15]). *For the expected arrival rates $A_i(t)$ and process rates $B_i(t)$, there exists a constant $\epsilon > 0$, which satisfies $\mathbb{E}\{A_i(t)\} - \mathbb{E}\{B_i(t)\} \le -\epsilon$.*

We see that such Slater Condition [15] is related to the system stability. Using this condition, we then have the following theoretical upper bound of queue length.

Theorem 2. *If problem (7) is feasible and Assumption 1 holds, then the proposed Algorithm 1 can stabilize the multi-queue blockchain system, and the time-average queue length satisfies*

$$\limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{i=1}^{N}\mathbb{E}\{Q_i(\tau)\} \le \frac{B + V(pen^* - pen^{min})}{\epsilon},$$

where B is defined in Lemma 3 and pen^{min} is the minimum resource consumption yielded by all the feasible solutions.

Proof. Referring to Eq. (20), we separate the actual queues apart from the inequation. Given $L(\mathbf{Q}(t)) \geq 0$, we have

$$
\begin{aligned}
&-\mathbb{E}\{L(\mathbf{Q}(0))\} + V\sum_{\tau=0}^{t-1} pen(\tau) \\
&\leq B\cdot t + Vt\cdot pen^{*} + \mathbb{E}\left\{\sum_{\tau=0}^{t}\sum_{i=1}^{N} Q_i(\tau)[A_i(\tau) - B_i(\tau)]\right\}.
\end{aligned}
$$

Exploiting Assumption 1, we then have

$$
\begin{aligned}
&-\mathbb{E}\{L(\mathbf{Q}(0))\} + V\sum_{\tau=0}^{t-1} pen(\tau) \\
&\leq B\cdot t + Vt\cdot pen^{*} - \sum_{\tau=0}^{t}\sum_{i=1}^{N} Q_i(\tau)\epsilon.
\end{aligned}
$$

Rearranging terms and dividing both sides by $t\cdot\epsilon$, we get

$$
\begin{aligned}
&\frac{1}{t}\sum_{\tau=0}^{t}\sum_{i=1}^{N} Q_i(\tau) \\
&\leq \frac{B\cdot t + \mathbb{E}\{L(\mathbf{Q}(0))\} + Vt\cdot pen^{*} - V\sum_{\tau=0}^{t-1} pen(\tau)}{t\cdot\epsilon} \\
&\leq \frac{B + V(pen^{*} - pen^{min})}{\epsilon} + \frac{\mathbb{E}\{L(\mathbf{Q}(0))\}}{t\cdot\epsilon}.
\end{aligned}
$$

Finally, we take a limsup as $t\rightarrow\infty$ to conclude the proof.□

Insights. Theorems 1 and 2 show that the proposed DPP-based algorithm has an $O(1/V) \sim O(V)$ tradeoff between resource consumption and queue-length performance. As the TX processing delay is proportional to the queue's length, thus the proposed algorithm can make a fine-balanced cost-delay tradeoff for the permissioned sharded blockchain.

5 Performance Evaluation

In this section, we conduct numerical simulations to evaluate the proposed *DPP Res. Allocation* algorithm in the context of PBFT-based sharded permissioned blockchain.

5.1 Basic Settings for Numerical Simulation

In simulations, the sharded blockchain network consists of a varying number of N network shards.

Dataset. For simulations, we generate numerical synthesized transaction dataset as the arrived TXs $A_i(t)$ for each network shard $i\in I$ at each timeslot $t\in\{1,2,\ldots,1000\}$ timeslots. The integer value of $A_i(t)$ is randomly distributed within the range [5, 25].

Resources. Network shards consume the cloud-platform resources to strive for consensus. We consider two types of resources in our simulation. The first type is the computing power (measured in the # of CPU cores). The second type is the network bandwidth, measured in Kbit/Second (Kb/s). Recall that we have mentioned that the computing power is mainly for committee formation and the network bandwidth is for PBFT consensus in Section 3.1. The weights of those two resources are variable depending on different network configurations. How to set the weights of different resources is not the focus of this paper. Thus, we set the weights of the two types of resources to 5 and 3, respectively. Other weight settings can be also evaluated similarly and thus omitted.

Other Parameters. We then fix R=5 to denote the reward of dequeueing each unit of TX data. The consensus parameter α for all network shards is set to 0.5. The total simulation duration is set as 1000 timeslots.

5.2 Metrics

To compare the performance of algorithms, we focus on the following metrics.

- *Queue Length*. The backlog of each queue which represents the number of unprocessed TXs in the mempool of each shard.
- *Computing-Power Consumption*. The consumption of computing-power for the all shards spending on processing TXs in each timeslot.
- *Network Bandwidth Consumption*. The bandwidth consumption for all network shards to process TXs waited in mempool per timeslot.

5.3 Baselines

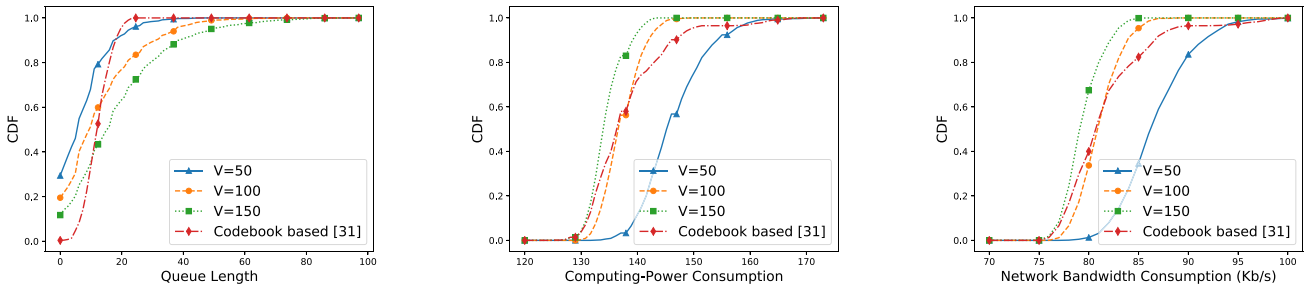
We consider the following baselines for the comparison with the proposed algorithm.

- *Top-S Res. Allocation* [29]. This baseline equally allocates each type of resource to the queues that locate in the top-S percentage of all network shards with respect to (w.r.t) their queue length.
- *Longest-First Res. Allocation* [30]. This baseline always allocates each type of resource to the queue that has the longest queue among all network shards w.r.t their queue length.
- *Average Res. Allocation*. This strategy allocates each type of available resources to all network shards on average at each timeslot.
- *Random Res. Allocation*. The last baseline allocates each type of available resources with a random amount to every network shard at each timeslot.

5.4 Performance Analysis

5.4.1 Effect of Tuning Parameter V

The first group of simulations evaluates the effect of parameter V. First, we fix $N=4$ and set V to 50, 100, and 150 in different executing cases. In addition, to study the effect of dynamically tuning parameter V, we also implement a codebook-based method referring to [31]. In such codebook-based approach, V is adaptively varied within the range [50, 150] according to the changes of shard's queue backlog and the resource consumption. The goal is to maintain a balanced trade-off between those two objective terms. For example, when the queue length of shards is observed too large, V is tuned to a small value accordingly. When too many resources are consumed by network shards, V is then tuned to a large value. For all network shards, the total computing-power budget and the total network bandwidth budget are set to 200 CPUs and 100 Kb/s, respectively. From Figs. 4a, 4b and 4c, we can observe that the proposed *DPP Res. Allocation* can stabilize all queues of network shards, since the length of 90% of all queues is within 40



(a) CDF of the queue length of all network shards per timeslot.

(b) CDF of the computing-power consumption of all network shards per timeslot.

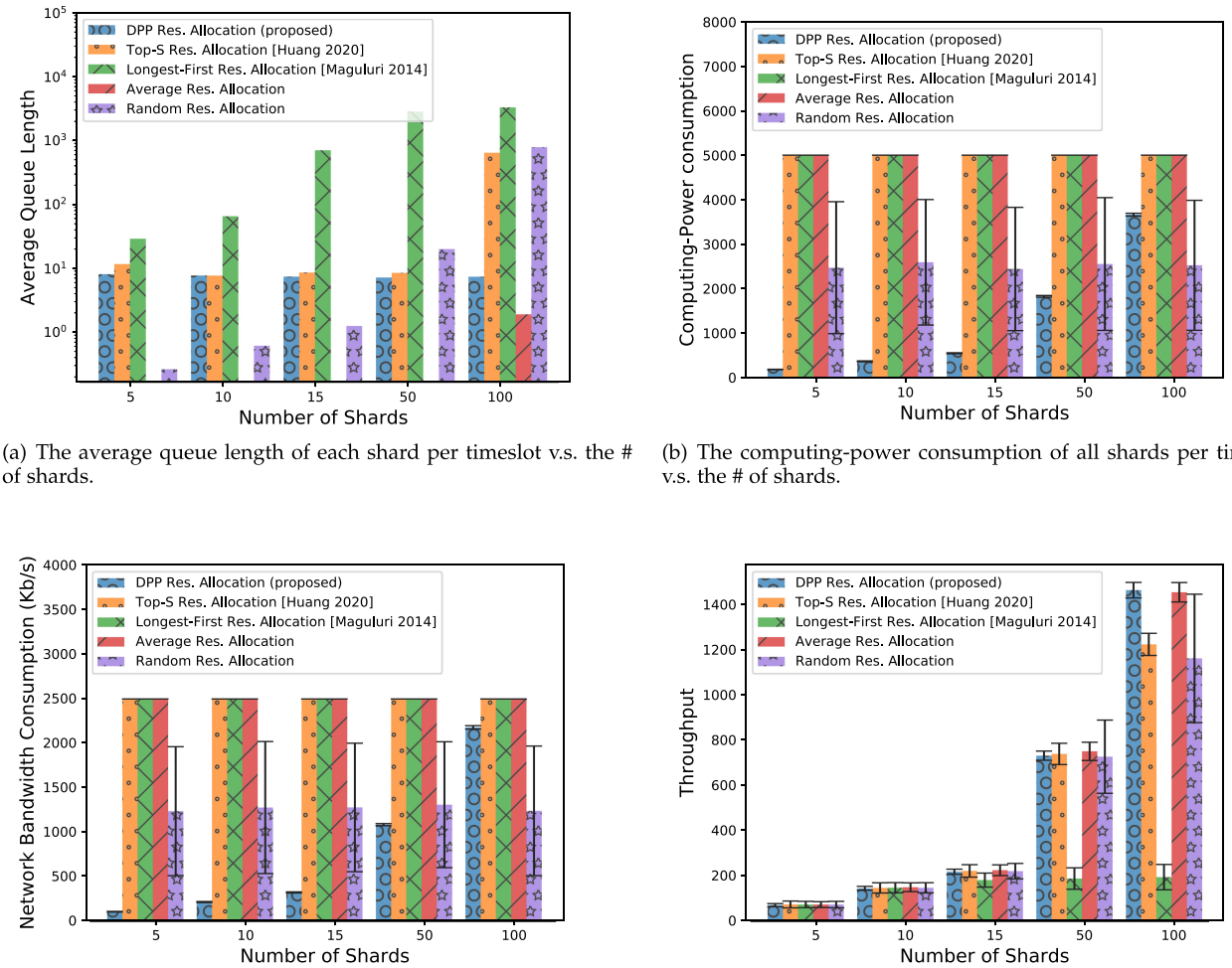
(c) CDF of the network bandwidth consumption of all network shards per timeslot.

Fig. 4. Performance evaluation while varying parameter V.

while varying the parameter V. Second, the virtual-queue technique can keep the two types of resources under their given individual budgets. Finally, we find that a larger V leads to a lower resource-consumption in terms of both computing-power and network bandwidth consumption. Thus, the resource consumption illustrates an O($1/V$) ratio. However, a large V increases the queue length and thus causes larger waiting latency for TXs. Therefore, the queue length shows an O(V) ratio. Those observations match the insights disclosed in the end of Section 4.2. In contrast, the codebook-based method has a similar resource-consumption performance with other cases under fixed V, but yields a more narrow range of queue length than other three cases. In conclusion, the tradeoff between the resource consumption and the queueing delay of TXs should be made carefully by tuning the parameter V.

5.4.2 Performance Comparison With Baselines

Through Fig. 5, we compare the proposed *DPP Res. Allocation* algorithm with the three baselines aforementioned. The



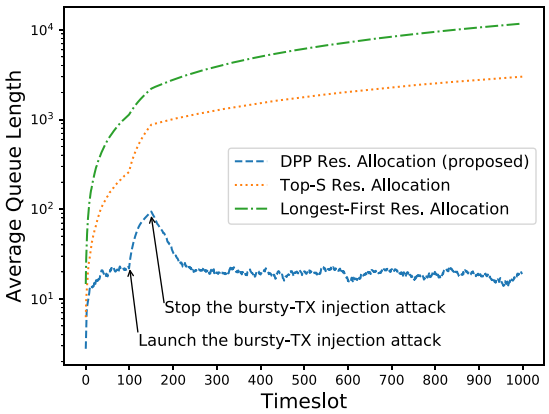
(a) The average queue length of each shard per timeslot v.s. the # of shards.

(b) The computing-power consumption of all shards per timeslot v.s. the # of shards.

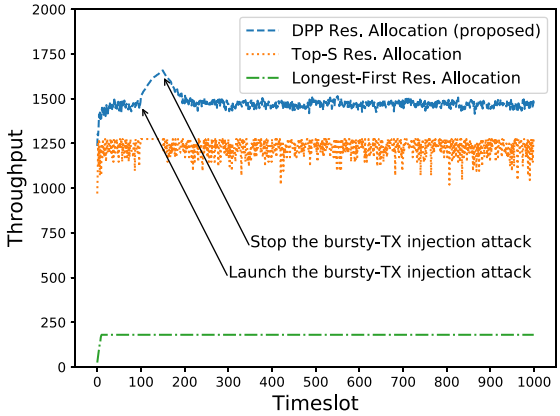
(c) The network bandwidth consumption of all shards per timeslot v.s. the # of shards.

(d) The throughput (total dequeued TXs in each timeslot) of all shards per timeslot v.s. the # of shards.

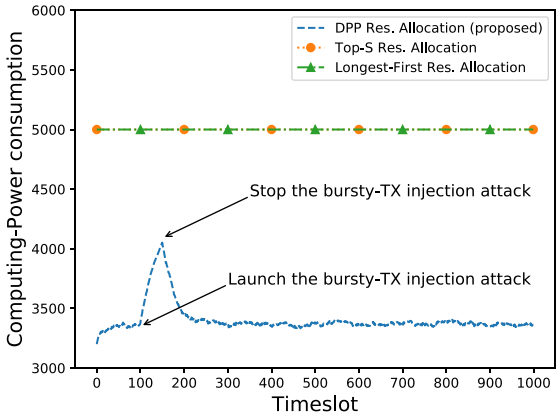
Fig. 5. Performance comparison with baseline algorithms.



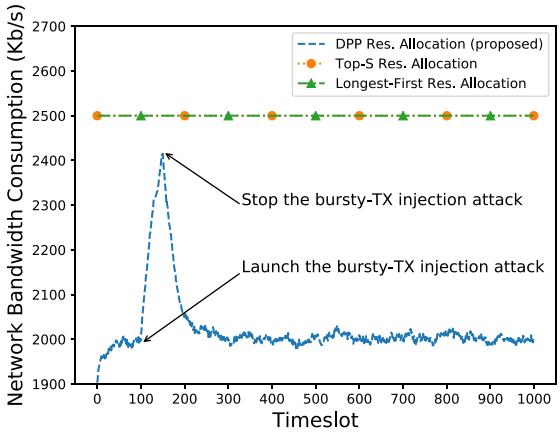
(a) The average queue length of each baseline under continued bursty-TX injection attacks.



(b) The throughput (total dequeued TXs in each timeslot) of each baseline under the bursty-TX injection attack.



(c) The computing-power consumption of each baseline under the bursty-TX injection attack.



(d) The network bandwidth consumption of each baseline under the bursty-TX injection attack.

Fig. 6. Performance evaluation under the continued bursty-TX injection attack against all network shards.

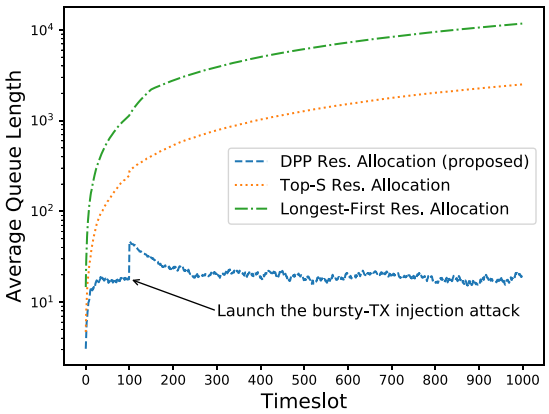
budgets of computing-power and network bandwidth resources are set to 5000 CPUs and 2500 Kb/s, respectively. The number of network shards varies from 5 to 100. For the *Top-S Res. Allocation*, we set S to 50%.

Interestingly, from Fig. 5a, we observe that when the shard number is smaller than 100, *Top-S Res. Allocation* has the similar queue length as the proposed *DPP Res. Allocation* does. However, when the shard number reaches 100, the *Top-S Res. Allocation* strategy cannot maintain the stable queue length anymore, because the the *Top-S* strategy only focuses on allocating resources for the queues that locate at the top S percentage of all w.r.t queue length. When the number of network shards grows large, the resources are not able to fulfill the requirement of maintaining short queues for network shards. As for the *Longest-First Res. Allocation*, it can only maintain stable queue length when the number of shards is less than 10. When the number of shards exceeds 15, the average queue length of *Longest-First Res. Allocation* soars to about 700. Therefore, this strategy fails to guarantee a stable queue length. The reason is that this baseline allocates all available resources to the longest queue every timeslot. Thus, other queues are ignored, making the average queue length grows sharply.

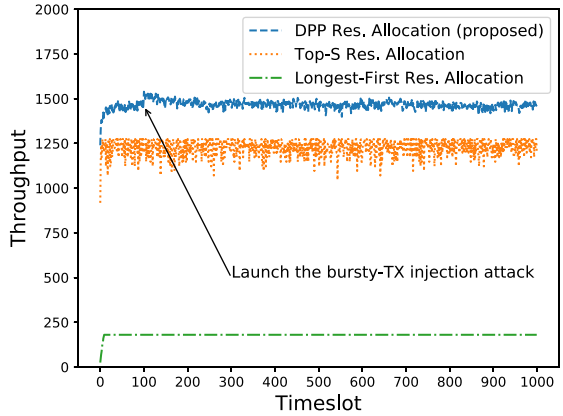
On the other hand, when the number of shards is less than 50, *Random Res. Allocation* yields a shorter queue length than *DPP Res. Allocation*. This is because *Random Res. Allocation* can provide sufficient required resources to all network shards with the given budgets. However, once the number of shards exceeds 50, the average queue length indicated by *Random Res. Allocation* increases drastically from 20 to 780, as shown in Fig. 5a. Since the queue length grows exponentially under the *Random Res. Allocation* strategy when the number of shards increases, its low resource-consumption properties shown in Figs. 5b and 5c are meaningless.

In the case of *Average Res. Allocation* strategy, we see that the queue length is even lower than that of *Random Res. Allocation* when the number of shards is small, and still maintains a very low level when the blockchain has 100 network shards. However, such low-level queue length is achieved by supplying all the available resources (i.e., the full computing-power budget shown in Fig. 5b and the full network-bandwidth budget shown in Fig. 5c) to all network shards. Therefore, the *Average Res. Allocation* strategy cannot achieve a balance between resource consumption and queue's stability.

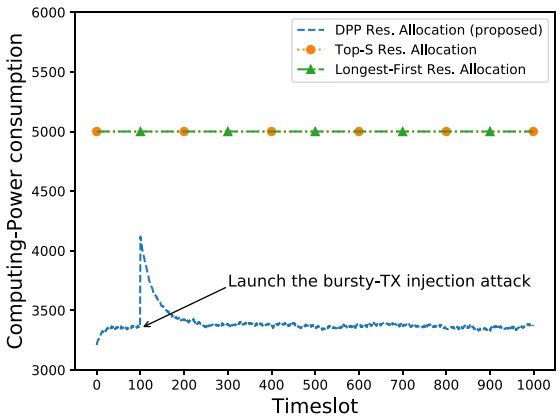
In contrast, the proposed *DPP Res. Allocation* algorithm can always maintain a stable queue length shown in Fig. 5a when the number of shards varies from 5 to 100. Also, Figs. 5b and 5c demonstrate that *DPP Res. Allocation*



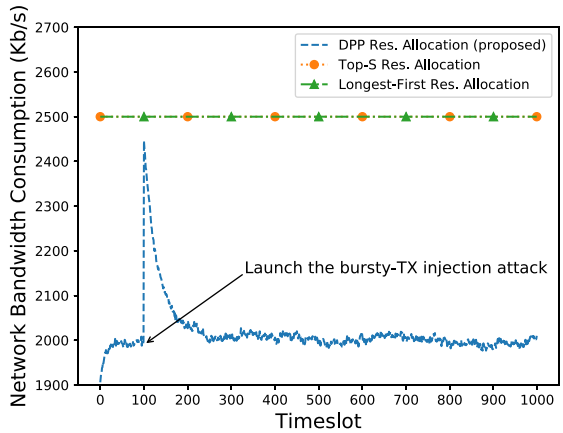
(a) The average queue length of each baseline under a drastic bursty-TX injection attack.



(b) The throughput (total dequeued TXs in each timeslot) of each baseline under the bursty-TX injection attack.



(c) The computing-power consumption of each baseline under the bursty-TX injection attack.



(d) The network bandwidth consumption of each baseline under the bursty-TX injection attack.

Fig. 7. Performance evaluation under the drastic bursty-TX injection attack against a single network shard.

requires a linearly-growing resource consumption when the number of shards increases from 5 to 100. Even though when the shard number reaches 100, the average resource consumption of *DPP Res. Allocation* is still lower than that of the other three baselines including *Top-S*, *Longest-First* and *Average Res. Allocation*.

Regarding throughput (calculated by the total # of dequeued TXs in each timeslot), Fig. 5d shows that when the shard number is less than 15, five strategies have similar throughput. However, when the number of shards exceeds 50, the throughput of *Longest-First Res. Allocation* is significantly lower than other four baselines. The reason is that *Longest-First Res. Allocation* only serves the shard having the most congested memory pool whenever the shard number varies. When the shard number reaches 100, the throughput of the proposed *DPP Res. Allocation* is higher than that of *Top-S* and *Random Res. Allocation* schemes, and is similar to that of *Average Res. Allocation*. Considering the throughput performance shown in Figs. 5b and 5c, it's not hard to see that the *DPP Res. Allocation* has the most efficient resource utilization among the five algorithms.

In summary, the proposed *DPP Res. Allocation* attains a fine-balanced tradeoff between queue stability and resource consumption comparing with other baselines.

5.4.3 *Continued Bursty-TX Injection Attacks Against All Shards*

In this part, we compare the performance under the continued bursty-TX injection attack of the proposed *DPP Res. Allocation* with two baselines, i.e., *Top-S* and *Longest-First Res. Allocation*. Since the other two baselines (*Average* and *Random Res. Allocation*) have very low efficiency on resource consumption, we omit their comparisons in this group of simulation. The budgets of computing-power and network bandwidth resources are set to 5000 CPUs and 2500 Kb/s, respectively. The number of shards is set as 100. The parameter S is fixed to 50% for *Top-S Res. Allocation*. To simulate the continued bursty-TX injection attack, we keep injecting a number of TXs to each network shard with a rate 25 TXs/timeslot between the 100th and the 150th timeslot.

Fig. 6a demonstrates the performance of queue length of three strategies under the continued bursty-TX injection attack. We can see that the proposed *DPP Res. Allocation* can quickly process the large number of injected TXs and maintain a short queue length in the 50 timeslots under attacking. In contrast, for the *Top-S* and *Random Res. Allocation* baselines, the average length of shard's queue has already become extremely congested even before we launch the continued bursty-TX injection attack at the 100th timeslot.

In terms of throughput, Fig. 6b shows that the proposed *DPP Res. Allocation* can always have the maximum throughput comparing with the other two baselines. As shown in Figs. 6c and 6d, *DPP Res. Allocation* requires the lowest resource consumption among the three algorithms. The reason is described as follows. Unlike *Top-S Res. Allocation* and *Random Res. Allocation*, the proposed algorithm does not have to utilize all of the available resources in every timeslot. When the continued bursty-TX injection attack begins, although *DPP Res. Allocation* tends to utilize a suddenly increasing resources, it can quickly enforce shard queues to return to stable status when the attack stops at the 150th timeslot.

5.4.4 Drastic Bursty-TX Injection Attack Against a Single Shard

Under the similar settings to that of the previous group of simulation, we are also interested in the performance of the three algorithm under the drastic bursty-TX injection attack against a specified network shard. The only difference is the implementation of the drastic injection attack. To simulate such the drastic bursty-TX injection attack, we inject 3000 TXs to only a single network shard at the beginning of the 100th timeslot.

Fig. 7 shows the evaluation results of the drastic bursty-TX injection attack. In contrast with the other two baselines, we observe the similar performance of the proposed DPP-based algorithm, in terms of average queue length, throughput, and the resource consumption. Comparing with the previous group of simulations, although the increases of both queue length and resource consumption are more sharp, the absolute values of those metrics are very similar to those of the continued injection attack. Importantly, the throughput of DPP-based algorithm still demonstrates the best among the three algorithms.

Thus, in summary, the proposed *DPP Res. Allocation* algorithm can maintain the stable queue length under both the continued and the drastic bursty-TX injection attacks, and illustrates a more efficient resource consumption than other two baselines.

6 Conclusion

System stability is critical to the key performance of the sharding-based blockchain. We study how to maintain stable queues for network shards by proposing a fine-grained resource-allocation algorithm for the PBFT-based sharded permissioned blockchain. Based on the multi-queue analytical model, we adopt the stochastic optimization technique to help us jointly consider both the resource consumption and the queue stability when allocating network resources to each blockchain shard. Through the proposed theoretical framework, we can choose how much we emphasize on resource-consumption or queue stability by dynamically tuning a weight parameter V. We also rigorously analyze the theoretical upper bounds of system objective and shard's queue length of the proposed DPP-based algorithm. Finally, the numerical simulation results show that the proposed DPP-based algorithm can effectively stabilize shard queues while requiring a reasonable level of resource consumption. Under two representative cases of bursty-TX injection attacks, the evaluation results demonstrate that the proposed DPP-based algorithm can well alleviate the imbalanced TX assignments with much shorter queue length, higher throughput, and lower resource consumption, comparing with other baselines.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Bus. Rev.*, pp. 1–9, Oct. 2008.

[2] T. Min, H. Wang, Y. Guo, and W. Cai, "Blockchain games: A survey," in *Proc. IEEE Conf. Games*, 2019, pp. 1–8.

[3] K. Liu, W. Chen, Z. Zheng, Z. Li, and W. Liang, "A novel debt-credit mechanism for blockchain-based data-trading in Internet of Vehicles," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 9098–9111, Oct. 2019.

[4] H.-N. Dai, Z. Zheng, and Y. Zhang, "Blockchain for Internet of Things: A survey," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8076–8094, Oct. 2019.

[5] M. G. Kim, A. R. Lee, H. J. Kwon, J. W. Kim, and I. K. Kim, "Sharing medical questionnaries based on blockchain," in *Proc. IEEE Int. Conf. Bioinf. Biomed.*, 2018, pp. 2767–2769.

[6] G. Yu, X. Wang, K. Yu, W. Ni, J. A. Zhang, and R. P. Liu, "Survey: Sharding in blockchains," *IEEE Access*, vol. 8, pp. 14 155–14 181, 2020.

[7] Q. Zhou, H. Huang, Z. Zheng, and J. Bian, "Solutions to scalability of blockchain: A survey," *IEEE Access*, vol. 8, pp. 16 440–16 455, 2020.

[8] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 17–30.

[9] M. Castro *et al.*, "Practical Byzantine fault tolerance," in *Proc. 3rd Symp. Operating Syst. Des. Implementation*, 1999, pp. 173–186.

[10] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "OmniLedger: A secure, scale-out, decentralized ledger via sharding," in *Proc. IEEE Symp. Secur. Privacy*, 2018, pp. 583–598.

[11] M. Zamani, M. Movahedi, and M. Raykova, "RapidChain: Scaling blockchain via full sharding," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 931–948.

[12] J. Wang and H. Wang, "Monoxide: Scale out blockchains with asynchronous consensus zones," in *Proc. 16th USENIX Symp. Netw. Syst. Des. Implementation*, 2019, pp. 95–112. [Online]. Available: https://www.usenix.org/conference/nsdi19/presentation/wang-jiaping

[13] L. N. Nguyen, T. D. T. Nguyen, T. N. Dinh, and M. T. Thai, "OptChain: Optimal transactions placement for scalable blockchain sharding," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst.*, 2019, pp. 525–535.

[14] T. Nguyen and M. T. Thai, "Denial-of-service vulnerability of hash-based transaction sharding: Attacks and countermeasures," 2020, *arXiv:2007.08600v2*.

[15] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures Commun. Netw.*, vol. 3, no. 1, pp. 1–211, 2010.

[16] Q.-L. Li, J.-Y. Ma, and Y.-X. Chang, "Blockchain queue theory," in *Proc. Int. Conf. Comput. Soc. Netw.*, 2018, pp. 25–40.

[17] 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.

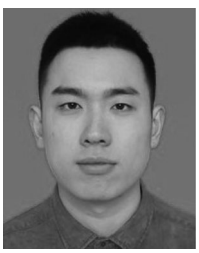
[18] R. A. Memon, J. Li, J. Ahmed, A. Khan, M. I. Nazir, and M. I. Mangrio, "Modeling of blockchain based systems using queuing theory simulation," in *Proc. 15th Int. Comput. Conf. Wavelet Act. Media Technol. Inf. Process.*, 2018, pp. 107–111.

[19] Y. Kawase and S. Kasahara, "A batch-service queueing system with general input and its application to analysis of mining process for bitcoin blockchain," in *Proc. IEEE Int. Conf. Internet Things IEEE Green Comput. Commun. IEEE Cyber Phys. Soc. Comput. IEEE Smart Data*, 2018, pp. 1440–1447.

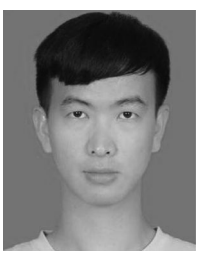
[20] J. Misic, V. B. Misic, X. Chang, S. G. Motlagh, and M. Z. Ali, "Block delivery time in bitcoin distribution network," in *Proc. IEEE Int. Conf. Commun.*, 2019, pp. 1–7.

[21] J. C. Corbett *et al.*, "Spanner: Google's globally-distributed database," in *Proc. 10th USENIX Symp. Oper. Syst. Des. Implementation*, 2012, pp. 261–264. [Online]. Available: https://www.usenix.org/conference/osdi12/technical-sessions/presentation/corbett
[22] M. Al-Bassam, A. Sonnino, S. Bano, D. Hrycyszyn, and G. Danezis, "Chainspace: A sharded smart contracts platform," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2018, pp. 1–15.
[23] H. Dang, T. T. A. Dinh, D. Loghin, E.-C. Chang, Q. Lin, and B. C. Ooi, "Towards scaling blockchain systems via sharding," in *Proc. Int. Conf. Manage. Data*, 2019, pp. 123–140.
[24] Y. Jiao, P. Wang, D. Niyato, and Z. Xiong, "Social welfare maximization auction in edge computing resource allocation for mobile blockchain," in *Proc. IEEE Int. Conf. Commun.*, 2018, pp. 1–6.
[25] N. C. Luong, Z. Xiong, P. Wang, and D. Niyato, "Optimal auction for edge computing resource management in mobile blockchain networks: A deep learning approach," in *Proc. IEEE Int. Conf. Commun.*, 2018, pp. 1–6.
[26] M. Fang and J. Liu, "Toward low-cost and stable blockchain networks," in *Proc. IEEE Int. Conf. Commun.*, 2020, pp. 1–6.
[27] H.-W. Wang, "Ethereum sharding: Overview and finality," Dec. 2017. [Online]. Available: https://medium.com/@icebearhww/ethereum-sharding-and-finality-65248951f649
[28] D. Dasgupta, J. M. Shrein, and K. D. Gupta, "A survey of blockchain from security perspective," *J. Banking Financial Technol.*, vol. 3, no. 1, pp. 1–17, 2019.
[29] H. Huang, S. Guo, W. Liang, K. Wang, and Y. Okabe, "Coflow-like online data acquisition from low-earth-orbit datacenters," *IEEE Trans. Mobile Comput.*, vol. 19, no. 12, pp. 2743–2760, Dec. 2020.
[30] S. T. Maguluri and R. Srikant, "Scheduling jobs with unknown duration in clouds," *IEEE/ACM Trans. Netw.*, vol. 22, no. 6, pp. 1938–1951, Dec. 2014.
[31] J. Koo, J. Yi, J. Kim, M. A. Hoque, and S. Choi, "Seamless dynamic adaptive streaming in LTE/Wi-Fi integrated network under smartphone resource constraints," *IEEE Trans. Mobile Comput.*, vol. 18, no. 7, pp. 1647–1660, Jul. 2019.

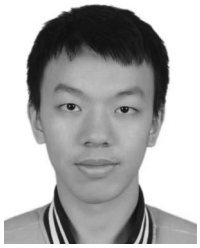
Huawei Huang (Member, IEEE) received the PhD degree in computer science and engineering from the University of Aizu, Aizuwakamatsu, Japan, in 2016. He is currently an associate professor with Sun Yat-Sen University. He has served as a research fellow of JSPS, and an assistant professor with Kyoto University, Japan. His research interests include blockchain and distributed computing. He is now serving as a guest editor of the *IEEE Journal on Selected Areas in Communications* and *IEEE Open Journal of the Computer Society*, the operation-committee chair for the IEEE Symposium on Blockchain at IEEE SERVICES 2021, and the TPC co-chair of GLOBECOM'2021/ICC'2022 Workshop on scalable, secure, and intelligent blockchain.



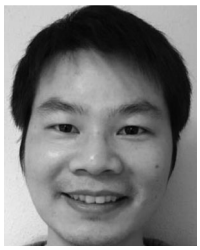
Zhengyu Yue is currently working toward the master's degree in the School of Computer Science and Engineering, Sun Yat-Sen University, Guangzhou, China. His research interest include blockchain.



Xiaowen Peng is currently working toward the master degree in the School of Computer Science and Engineering, Sun Yat-Sen University, Guangzhou, China. His research interest include blockchain.



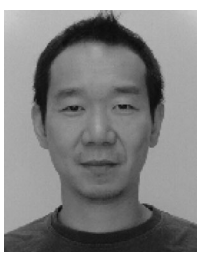
Liuding He is currently working toward the undergraduate degree in the School of Computer Science and Engineering, Sun Yat-Sen University, Guangzhou, China. His research interest include blockchain.



Wuhui Chen (Member, IEEE) received the bachelor's degree from Northeast University, Shengyang, China, in 2008, and the master's and PhD degrees from the University of Aizu, Aizuwakamatsu, Japan, in 2011 and 2014, respectively. From 2014 to 2016, he was a research fellow with the Japan Society for the Promotion of Science, Tokyo, Japan. From 2016 to 2017, he was a researcher with the University of Aizu. He is currently an associate professor with Sun Yat-sen University, Guangzhou, China. His current research interests include edge/cloud computing, cloud robotics, and blockchain.

Hong-Ning Dai (Senior Member, IEEE) received the PhD degree in computer science and engineering from the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong. He is currently with the Department of Computing and Decision Sciences, Lingnan University, Hong Kong, as an associate professor. His current research interests include blockchain and the Internet of Things. He has served as associate editors of the *IEEE Transactions on Industrial Informatics*, *IEEE Systems Journal*, and *IEEE Access*. He is also a senior member of the ACM.

Zibin Zheng (Senior Member, IEEE) received the PhD degree from the Chinese University of Hong Kong, Hong Kong, in 2011. He is currently a professor with the School of Computer Science and Engineering, Sun Yat-Sen University, China. He published more than 300 international journal and conference papers, including nine ESI highly cited papers. His research interests include blockchain, artificial intelligence, and software reliability. He was a recipient of several awards, including the Top 50 Influential Papers in Blockchain of 2018, the ACM SIGSOFT Distinguished Paper Award at ICSE2010, the Best Student Paper Award at ICWS2010.



Song Guo (Fellow, IEEE) received the PhD degree in computer science from the University of Ottawa, Ottawa, Canada. He is currently a full professor with the Department of Computing, Hong Kong Polytechnic University, Hong Kong. He has authored or coauthored more than 450 papers in major conferences and journals. His current research interests include big data, cloud and edge computing, mobile computing, and distributed systems. He was a recipient of the 2019 TCBD Best Conference Paper Award, 2018 IEEE TCGCC Best Magazine Paper Award, 2017 *IEEE Systems Journal* Annual Best Paper Award, and six other best paper awards from IEEE/ACM conferences. He was an IEEE Communications Society distinguished lecturer. He has served as an associate editor of the *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Emerging Topics in Computing*, etc. He also served as the general and program chair for numerous IEEE conferences. He currently serves in the Board of Governors of the IEEE Communications Society.