# A Reputation-based Mechanism for Transaction Processing in Blockchain Systems

Jiarui Zhang, Yukun Cheng, Xiaotie Deng, *Fellow, IEEE*, Bo Wang, Jan Xie, Yuanyuan Yang, *Fellow, IEEE*, and Mengqian Zhang

**Abstract**—Blockchain protocols require nodes to verify all received transactions before forwarding them. However, massive spam transactions cause the participants in blockchain systems to consume many resources in verifying and propagating transactions. This paper proposes a reputation-based mechanism to increase the efficiency of processing transactions by considering the reputations of the sending nodes. Reputations are in turn adjusted based on the quality of transaction processing. Our proposed reputation-based mechanism offers three main contributions. First, we modify the verification strategy so that nodes set a probability of verifying a received transaction considering the likelihood of it being spam: transactions from a node with a low reputation have a high probability of being verified. Second, we optimize the transaction forwarding protocol to reduce propagation delay by prioritizing forwarding transactions to reputable receivers. Third, we design a data request protocol that provides alternative data exchange methods for nodes with different reputations. A series of simulations demonstrate the performance of our reputation-based mechanism.

**Index Terms**—Blockchain, Reputation-based Mechanism, Strategic Verification, Forwarding Protocol, Data Request.

## 1 Introduction

Blockchains have emerged as impressive and transformational ledgers that safely store transaction histories and participants' accumulated credits. Their use is particularly associated with Bitcoin [1], the most successful cryptocurrency, and others like Ethereum [2] [3] and EOS [4]. They have made cryptocurrencies the most visible examples of successful peer-to-peer (P2P) systems.

Blockchains' rapid development has been accompanied by denial of service (DoS) and distributed DoS (DDoS) attacks [5] [6]. Attackers generate spam transactions, which may be spread across the network by other nodes. Although verification by block generators prevents valid blocks from including the spam transactions, they can delay the transmission of valid transactions, and so waste communication resources. As block generators have to store and transfer all transactions that are waiting to be confirmed, spam transactions would fill their storage and network bandwidth, thus severely impeding the storage and dissemination of valid transactions. Therefore, the spread of spam transactions is incredibly wasteful and destructive to the network, and the smooth running of a blockchain system requires the efficient suppression of spam transactions.

This paper proposes a method to improve system efficiency. We design a reputation-based mechanism that considers the quality of received transactions. In general, a node would assign a higher reputation to a neighbor that sends more valid transactions. Any neighbor that keeps sending spam transactions will be assigned a lower reputation. Each node has a reputation threshold, and will not connect with neighbors with a lower reputation.

The reputation-based mechanism can also handle transaction processing between nodes, including the communication of transactions in the network and the communication of data on the blockchain. First, we propose a verification strategy supported by the reputation-based mechanism. Upon receiving a new transaction, a node needs to decide whether to verify it. Generally, the node will consider the sender's reputation: if it is low, the node does not trust the sender, and so assumes a high probability of it sending spam transactions. A sender with a high reputation will be assumed to be more likely to send valid transactions. Therefore, a node can choose to verify a transaction with a probability related to the sender's reputation. Second, we use the reputation-based mechanism to optimize transaction forwarding. Reputation can support the prioritization of neighbors to receive forwarded transactions, thereby speeding up the propagation of transactions to honest nodes. When limited by the network bandwidth, nodes will prefer to send transactions to those neighbors with higher reputations. Third, we design a protocol to allow nodes in blockchain systems to request data. While nodes with a high reputation can keep interacting with each other, nodes with a low reputation can also request data by exchanging their computational resources for services.

This paper offers the following main contributions.

- We design a reputation-based mechanism that allows each node to compute a reputation value for each of its neighbors. Intuitively, a node forwarding more valid and fewer spam transactions would have a


- *J. Zhang and Y. Yang are with the Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794, USA. E-mail: {jiarui.zhang.2, yuanyuan.yang}@stonybrook.edu*
- *Y. Cheng is with Suzhou University of Science and Technology, Suzhou, China. E-mail: ykcheng@amss.ac.cn*
- *X. Deng is with Peking University, Beijing, China. E-mail: xiaotie@pku.edu.cn*
- *B. Wang is with Nervina Labs, Ltd., Hangzhou, China. E-mail: cipher@nervos.org*
- *J. Xie is with Hangzhou Cryptape Technology Co., Ltd., Hangzhou, China. E-mail: jan@cryptape.com*
- *M. Zhang is with Shanghai Jiao Tong University, Shanghai, China. E-mail: mengqian@sjtu.edu.cn*

higher reputation.
- We propose a verification strategy that allows the system to reduce the spread of spam transactions by selectively verifying a proportion of the received transactions. A node will be more likely to verify a transaction from a disreputable sender.
- Supported by the reputation-based mechanism, we optimize the transaction forwarding protocol to reduce propagation delay. Specifically, a node preferentially forwards transactions to reputable neighbors.
- We design a two-phase data-request protocol to allow reputable nodes to exchange data, and we enable low-reputation nodes to pursue services in exchange for offering their computational resources.
- Extensive simulations for different environments demonstrate the performance of our mechanism. They show the proposed reputation-based mechanism to efficiently reduce the spread of spam transactions by screening out most of the malicious nodes generating them. Moreover, our optimized transaction forwarding protocol can reduce delays in the network's transaction propagation.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 formulates the overall environment and assumptions discussed in this paper. Section 4 proposes the reputation-based mechanism and discusses its features. Sections 5 and 6 discuss strategic verification and introduce an optimized transaction-forwarding protocol, respectively. For data exchange among nodes, we propose a data request protocol in Section 7. Section 8 reports a series of simulations demonstrating the performance of our mechanism. The last section concludes this paper with a summary and discussion of potential future work.

## 2 Related Work

Bitcoin was invented in 2008 by Satoshi Nakamoto [1] and has become one of the most popular P2P applications. It relies on a blockchain that records transactions generated by P2P network participants as a distributed ledger. Its properties, such as decentralization, openness, and immutability, have made Bitcoin the most attractive cryptocurrency. Encouraged by its success, other cryptocurrencies like Ethers in Ethereum [2] [3] and EOS in EOSIO [4] were developed. These cryptocurrencies have developed new features to make them attractive to some users and investors. Ethereum has introduced a smart contract deployment to extract the capability of blockchain. EOSIO has attempted to provide a decentralized application development environment that offers high transaction throughput.

Blockchains also have drawbacks, one being vulnerability to various types of attack. Researchers have sought to identify possible attacks on the blockchain and defend against them. For example, malicious blockchain mining strategies have been well studied [7]–[9]. Eyal et al. [7] showed that miners could strategically mine for unfair profits. Garay et al. [8] formalized Bitcoin's properties and offered a simple solution to improve security. Pass et al. [9] proposed a Fruitchain design to ensure fairness and decrease the variance of mining rewards.

DoS and DDoS attacks are common in blockchain systems. They involve attackers propagating a large number of spam transactions to congest the system. There are two main types. One propagates massive amounts of spurious data to waste resources. Luu et al. [10] studied this type of attack in a blockchain system with smart contracts and showed that numerous invalid transactions can either waste miners' computational resources or lead miners to accept blocks with incorrect script results. Pontiveros et al. [11] developed a strategy of enacting sluggish contracts with slow execution times in the virtual machine to give attackers an advantage over their competitors. The second type of attack propagates transactions with high transaction fees. Baqer et al. [12] presented an empirical example attack on Bitcoin, showing that the attack increases the transaction fees of non-spam transactions and delays their processing. An analysis of this type of attack in the Ethereum network [13] found an increase in the gas price of Ethereum. Some works have sought to solve these problems. Reyzin et al. [14] improved the speed of verification and reduced the verifier load via the design of authenticated dynamic dictionaries. Signorini et al. [15] proposed to build a threat database, which collects patterns of malicious transactions and detects potential attacks. Saad et al. [16] illustrated the impacts of DDoS attacks on cryptocurrency systems' memory pools. They also formulated an attack model that causes massive transaction backlogs and increases mining fees. They required transactions to meet a threshold of fee and age to prevent spam transactions [17]. Yang et al. [18] provided a spam-transaction intrusion-detection model based on deep learning.

A possible solution to DoS or DDoS attacks in P2P systems is a reputation mechanism for nodes to evaluate the trust level of other participants in the network. Resnick et al. [19] enumerated the pros and cons of eBay, which has one of the earliest centralized Internet reputation mechanisms. However, most P2P networks do not have centralized parts, thus decentralized reputation frameworks have been proposed [20]–[22].

Blockchains bring new concepts and challenges to the design of reputation systems. Some works have applied blockchains to reputation mechanisms to expand their capabilities in different scenarios. Carboni [23] demonstrated the feasibility of replacing a traditional centralized reputation system with a decentralized system deployed on the Bitcoin blockchain. Fortino et al. [24] proposed a reputation-based model involving reputation evaluation compatible among different local Internet-of-Things networks. Liu et al. [25] focused on the consumer–retailer reputation management and proposed an anonymous blockchain-based reputation mechanism. Khaqqi et al. [26] proposed an emission trading scheme model customized for Industry 4.0 integration. They combined blockchain technology and reputation evaluation to improve the scheme's efficacy and address fraud. Shala et al. [27] presented a blockchain-based trust-evaluation mechanism for machine-to-machine interactions.

Reputation mechanisms can also be applied to strengthen blockchains. Gai et al. [28] presented a proof-of-reputation, in which reputation is the incentive for both good behavior and block publication. Chai et al. [29] applied a blockchain-based resource-sharing paradigm on the

Internet-of-Vehicles. They used a reputation value to indicate the reliability of vehicles and employed proof-of-reputation to reduce the resource consumption by the consensus mechanism. Li et al. [30] considered the number of node communication links and degree of node trust to improve communication efficiency, but they did not specify the method of calculating the degree of trust. Nojoumian et al. [31] applied a reputation-based paradigm to prevent dishonest miners in the blockchain during the consensus process. Yu et al. [32] deployed a reputation mechanism based on voting and rotation on their RepuCoin blockchain for blockchain consensus. Compared with traditional blockchains, RepuCoin guarantees high security and higher transaction throughput. Oliveira et al. [33] proposed a blockchain reputation-based consensus to prevent malicious actions on the blockchain. Their mechanism selects a set of miners to generate new blocks, and randomly selects judges to monitor the miners.

## 3 Environmental Settings

We propose here environmental settings including verifications, transactions, and nodes.

### 3.1 Transaction Verification Cost

The continuous updating of transactions in a network relies on the generation of new blocks, which in turn relies on the maintenance of nodes. The generator of a new block would pack in it a subset of valid transactions after verifying them. As encouragement, the block generator receives transaction fees from the proposers of the packed transactions. In general, a block generator tends to pack transactions with more transaction fees and fewer verification costs to maximize its profits. Thus, the verification cost is a crucial consideration for block generators.

Generally, a valid transaction must meet some requirements. For instance, its format must be correct and it must contain legal signatures. To ensure the validity of transactions, nodes need to run different verification operations, which can be represented as a series of instructions. Inspired by different CPU instructions with different predetermined running cycle costs [34], the instructions that occur during blockchain verification also assigned different costs. We use the concept of **cycle** to measure one unit of computational cost. By indicating the cycle cost of each instruction in advance, the computational cost of verifying a transaction can be represented by the total number of required cycles. In this way, all nodes can unambiguously compute the total cycle cost of verifying each transaction.

When proposing a new transaction, the proposer is required to attach the corresponding cycle cost as an essential part of the transaction. There are three reasons for this. First, an incorrect cycle cost can indicate to others that the transaction sender has either not verified or misrepresented the cycle cost. Second, nodes can schedule the verification of transactions in cost order. Third, verification may cause a halting problem that is not computable [35], thus nodes are responsible for pointing out the computational cost of transactions.

### 3.2 Types of Transaction

Transactions in common blockchain environments can essentially be classified as valid or invalid. In our environment, each transaction not only contains regular transaction components but also a cycle cost to measure the resources required for verification. The correctness of the cycle cost represents another factor by which transactions can be classified. We define three types of transaction based on validity and the correctness of cycle cost. A transaction can be:

- Valid with correct cycle cost (**VC transaction**). The transaction is valid, and the attached cycle cost is correct. It can be verified, and the attached cycle cost matches the real cycle cost.
- Valid with incorrect cycle cost (**VI transaction**). The transaction is valid, but the attached cycle cost is not correct. It can be verified, and the attached cycle cost does not match the real cycle cost.
- Invalid. The transaction should not be forwarded, but should be intercepted and excluded from the blockchain.

The following discussion focuses on these three types of transaction. Invalid transactions as defined here are considered spam.

### 3.3 Types of Node

Nodes in blockchain networks may have complex behavior modes. To simplify the behavior modes of nodes in our environment, we classify nodes by their behaviors regarding the generation, verification, and forwarding of transactions. A node can be:

- Honest. An honest node use rules to verify transactions. It forwards verified VC transactions and verified VI transactions after revising the cycle cost. It also forwards transactions that it decides not to verify. It generates VC transactions.
- Lazy. A lazy node does not verify transactions but forwards all transactions. It generates VC transactions.
- Malicious. A malicious node does not verify transactions. It forwards all transactions without verification and generates VC, VI, and invalid transactions.

Honest nodes follow the rules and protocols when generating, verifying, and forwarding transactions. The blockchain system can run as expected when all nodes are honest. To encourage nodes to behave honestly, the network system should protect and promote honest nodes and penalize others. Unless otherwise stated, the nodes discussed in this paper are honest nodes by default.

Lazy and malicious nodes are collectively called dishonest. Lazy nodes facilitate the spread of invalid transactions but do not deliberately generate them. Malicious nodes facilitate the spread of invalid transactions and generate them to disrupt the network. Intuitively, malicious nodes are more harmful to the blockchain system than lazy nodes, as lazy nodes do not generate invalid transactions. Thus, the network system should punish lazy nodes slightly, and punish malicious nodes severely.

## 4 Reputation-based Mechanism

This section first presents our reputation-based mechanism that considers the quality of transactions from neighbors. In general, a node that transfers more VC transactions has a higher reputation value. We also summarize several properties and advantages of the mechanism and analyze some potential attacks on the mechanism.

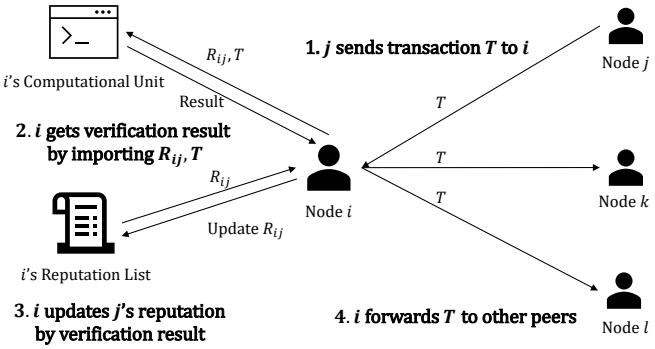



Fig. 1. Overview of how transaction forwarding and the reputation-based mechanism affect each other.

### 4.1 Design of the Reputation-based Mechanism

#### 4.1.1 Overview

We introduce the reputation-based mechanism from the perspective of an individual node $i$ for brevity. In the network, each node maintains a list of reputations, one for each of its neighbors. For example, node $i$ maintains reputation $R_{ij}$ for its neighbor $j$, which represents $i$'s view of $j$'s prior behavior.

When node $i$ establishes a connection with its neighbor $j$, it sets reputation $R_{ij}$ to an initial value, generally 0. If $R_{ij}$ falls below $i$'s pre-defined threshold, node $i$ will consider $j$ a dishonest node and terminate its connection with $j$. The minimum threshold is not greater than the initial value.

#### 4.1.2 Reputation value updating

Consider node $i$ receiving transaction $T$ from neighbor $j$. The attached cycle cost is $c\_cycle$, and the real cycle cost is $r\_cycle$. There are two cases when node $i$ receives transaction $T$. First, if node $i$ has not previously received transaction $T$, $i$ will verify $T$. After identifying the type of $T$, node $i$ updates $R_{ij}$ as follows.

- VC transaction, $c\_cycle = r\_cycle$:
  $R_{ij} \leftarrow R_{ij} + r\_cycle$.
- VI transaction, $c\_cycle \neq r\_cycle$:
  $R_{ij} \leftarrow R_{ij} - \max\{r\_cycle, c\_cycle\}$.
- Invalid transaction:
  $R_{ij} \leftarrow \min\{R_{ij}/2, R_{ij} - \max\{r\_cycle, c\_cycle\}\}$.

Second, if node $i$ has received transaction $T$ previously, it can update reputation $R_{ij}$ according to the existing verification result of $T$.

#### 4.1.3 Reputation attenuation

Long-term behaviors represent historic evidence of a node's type. However, nodes' behaviors may fluctuate for diverse reasons including corruption from attackers. When the latest data conflicts with previous data, we tend to give greater weighting to the latest data and reduce the influence of previous results. Thus, we propose an attenuation process to increase the impact of recent data.

Reputation attenuation is a proportional decay of the reputation value at fixed time intervals. Specifically, each node $i$ selects private values $t_{ij}$ and $p_{ij}$, which respectively indicate the attenuation interval and attenuation ratio of $R_{ij}$. Each reputation value $R_{ij}$ is updated to become $p_{ij}R_{ij}$ every $t_{ij}$ units of time, where $p_{ij} \in [0, 1]$ and $t_{ij} > 0$. This reputation attenuation ensures that the impact of a specific behavior on reputation is reduced over time.

We assume that the reputation $R_{ij}$ becomes stable after $m_{ij}$ units of time. It is proved that $p_{ij} = 1 - t_{ij}/m_{ij}$ [21]. Thus, node $i$ can accurately decide $t_{ij}$ and $p_{ij}$ by $m_{ij}$. When node $i$ considers that the stability of node $j$ is changing, it can dynamically adjust $m_{ij}$ and change the corresponding $t_{ij}$ and $p_{ij}$.

### 4.2 Properties of the Reputation-based Mechanism

#### 4.2.1 Privacy

The privacy of the reputation mechanism implies that only node $i$ can access and update reputation $R_{ij}$. Other nodes cannot accurately obtain $R_{ij}$.

#### 4.2.2 Independence

Independence means that reputation $R_{ij}$ is related only to the historic transactions transferred from node $j$ to node $i$. In other words, other transactions from node $k \neq j$ cannot influence $R_{ij}$. This prevents $R_{ij}$ from being influenced by possible strategic attacks from any other node except $i$ and $j$.

#### 4.2.3 Accountability

Node $j$ is accountable for its transmissions. Assume that node $j$ receives transaction $T$ from another node and forwards $T$ to node $i$. In our design, node $i$ knows that node $j$ sent $T$, but it is difficult to trace the path along with which $T$ came to node $j$. Thus, node $i$ can only adjust $R_{ij}$ according to its understanding of $T$ from node $j$.

### 4.3 Advantages of the Reputation-based Mechanism

#### 4.3.1 Low storage cost

The reputation-based mechanism has a low storage cost. Each node consumes extra storage to store reputation values for its neighbors and the results of verified transactions. Each node maintains one reputation value for each neighbor. According to the Bitcoin Core [36] protocol, a node can have 125 neighbors. Each node additionally consumes 8 bytes to store a 64-bit double-precision number for each neighbor and 1 kilobyte to store the reputation value of all neighbors. To avoid duplicate transmissions and verifications, each node needs to store the identifiers of received transactions and the verification results of verified transactions. For each transaction, nodes are required to

store three extra values indicating the validity, the verification flag, and the real verification cost. The validity and verification flag each need only 1 bit to be indicated. The real verification cost needs 8 bytes to be indicated as a 64-bit double-precision number. The total extra cost of a transaction is about 8 bytes. According to statistics [37], the size of one Bitcoin transaction ranges mainly from 350 to 800 bytes. Our reputation-based mechanism represents a comparatively low extra storage cost.

Other reputation-based mechanisms often require each node to receive and store the reputation information of all other nodes in the network, which entails high storage costs [27], [33].

#### 4.3.2 Low computational cost

The reputation-based mechanism has a low computational cost. Each node may update the relevant reputation value after receiving a transaction from a neighbor. Based on the local verification result, the update requires only some simple calculations. As the verification operation is much more complicated than reputation updating, our reputation-based mechanism has a relatively low computational cost.

Other reputation-based mechanisms often need to aggregate reputation values from multiple nodes and consume many resources to perform calculations with high computational complexity [25], [32].

#### 4.3.3 Scalability

Nodes are constantly joining or leaving the blockchain network. Under our reputation-based mechanism, a node joining or leaving will respectively make its neighbors add or remove only one reputation value. Increasing or decreasing the number of nodes will thus require all nodes in the network to perform only a small number of operations. Therefore, our reputation-based mechanism can achieve good scalability.

In contrast, other reputation-based mechanisms impose more costs on all modes when a node joins or leaves the blockchain network. For example, when a new node joins the network, other nodes incur a high cost to store evaluations of the new node on the blockchain [23] or create a complex smart contract to maintain the reputation of the new node [24]. The ensuing large demand for consumption resources in the network may result in poorer scalability.

### 4.4 Resistance to Common Attacks

#### 4.4.1 Accumulating reputation attack

An attacker may accumulate a high reputation and abuse the resulting trust for malicious ends. This is an accumulating reputation attack.

Two factors hinder this type of attack. First, an attacker must act honestly to accumulate a reputation. This requires employing the same computing resources as honest nodes to ensure a high quality of forwarded transactions. Second, the attacker cannot accurately know the reputation values its neighbors hold for the node, and thus cannot be certain whether its reputation is sufficient to launch an attack.

#### 4.4.2 Whitewashing attack

A whitewashing attack is common in reputation-based mechanisms. A node with a low reputation may leave and re-enter the network with a new identity. In our reputation-based mechanism, if a node launches a whitewashing attack, its reputation will be set to the initial value. A node with the initial reputation value is not trusted. Any invalid transaction sent by this node can thus be easily found.

In addition, when facing a new connection request, a node can design a proof-of-work (PoW) puzzle for the requester to solve before establishing a connection. This further increases the cost of whitewashing attacks.

## 5 Strategic Verification

This section introduces strategic verification, which is the most straightforward application of our reputation-based mechanism.

### 5.1 Transaction Verification in a Blockchain Network

To guarantee the security and correctness of the data on the blockchain, block generators need to verify all received transactions to ensure that every transaction in the block is valid. DoS and DDoS attacks, which spread a large number of invalid transactions, impose heavy verification workloads on the block generators. To alleviate the negative impacts from such attacks, traditional blockchain systems, such as Bitcoin, require every node to verify all received transactions and transfer only valid transactions. This is clearly costly for each node. There is consequently an incentive for some nodes to forward transactions as a free-rider without any verification, thus saving their computational resources. This is possible as there is no punishment for such malicious behavior under the current blockchain protocol.

Using the reputation-based mechanism, we propose a strategic verification mechanism to ensure that invalid transactions cannot be spread widely while reducing the number of verifications required. The main goal of our mechanism is to encourage honesty and punish nodes forwarding invalid transactions by adjusting their reputations. Intuitively, a node trusts a neighbor with a high reputation and doubts a neighbor with a low reputation. Therefore, a node could decide whether to check a transaction based on the reputation of the transaction sender. This can reduce the total number of verifications by accepting transactions from trusted nodes.

Note that strategic verification applies only to transaction forwarding. To ensure the security and correctness of the blockchain, if a block includes an invalid transaction, other nodes will not accept the block. Therefore, when generating a block, the generator must verify all selected transactions to ensure that only valid transactions can enter the block.

### 5.2 Verification Probability Function

Assume that node $i$ receives a transaction $T$ from neighbor $j$. Node $i$ will check $T$ with a probability obtained from the verification probability function $f_i$, which is related to reputation $R_{ij}$. The verification probability function is generally monotonically non-increasing, meaning that as

the reputation increases, the probability of verification decreases. Each node would select a verification probability function based on its conditions. Therefore, different nodes may have different verification probability functions.

### 5.3 Minimum Verification Probability

An attacker may attack the reputation-based mechanism by launching an accumulating reputation attack. It may initially behave honestly to deliberately increase its reputation. After enough time, it can act maliciously to attack the system. Such a node would have a high enough reputation for other nodes to assign a low probability to check transactions from the attacker. This would facilitate the accelerated spread of invalid transactions over the network.

Setting a minimum verification probability can prevent nodes from succumbing to this type of attack. Consider node $i$, setting its minimum verification probability $q_i$. No matter the amount of reputation accumulated by a neighboring node, node $i$ would still verify received transactions with a probability of at least $q_i$. If an attacker with a high reputation sends a large number of invalid transactions, they would still be checked, resulting in rapid reputation loss.

### 5.4 Strategic Verification Process

Assume node $i$ receives a transaction $T$ from neighbor $j$ and employs strategic verification. When node $i$ receives transaction $T$ for the first time, it decides whether to verify $T$. Instead of verifying all transactions, node $i$ verifies $T$ with a probability of $f_i(R_{ij})$, computed from the verification probability function. If node $i$ decides to verify $T$, node $i$ will update reputation $R_{ij}$ based on the rules in Section 4.1.2, according to the type of $T$. It will then decide the forwarding strategy. If node $i$ does not verify $T$, it means that node $i$ trusts transaction $T$ from neighbor $j$. In this case, node $i$ does not update reputation $R_{ij}$ and forwards $T$ in the same way as VC transactions are forwarded. If node $i$ has received $T$ before, it will not update $R_{ij}$ because it may not have verified the transaction before.

## 6 Transaction Forwarding Acceleration

Accelerating transaction forwarding in the blockchain network is a concerning problem. This section introduces an optimized transaction forwarding protocol. The reputation-based mechanism is applied to accelerate transaction forwarding.

### 6.1 Propagation Delay in a Blockchain Network

Propagation delays include delays to the propagation of both blocks and transactions. Delays to transaction propagation are more crucial because the speed of transaction propagation directly affects block propagation. More specifically, when a node receives a block, some transactions in the block may be unknown, thus the node has to verify these unknown transactions. The node can only further forward the block after receiving and verifying all of the unknown transactions. A substantial delay in transaction propagation will increase the potential amounts of unknown transactions in propagating blocks, thereby delaying block propagation. Therefore, we focus on reducing delays in transaction propagation.

### 6.2 Reducing Delay by Reputation

We briefly introduce Bitcoin's current settings. As the basic protocols in Nakamoto's work [1] do not cover networking, nodes in the same Bitcoin network may use differing network protocols and configurations. The developers of Bitcoin give a widely used network protocol [38]. By default, Bitcoin Core [36] allows up to 125 connections to different neighbors, eight of which are outbound. Nodes forward transactions through both inbound and outbound connections. To reduce unnecessary transaction forwarding, a node does not directly forward a complete transaction to its neighbors. As Fig. 2 shows, a node sends an inventory message (INV message) to ask whether a neighbor has the transaction presented by its hash value. If not, the node will send the specific transaction after receiving the get data request (GETDATA message) from the neighbor.

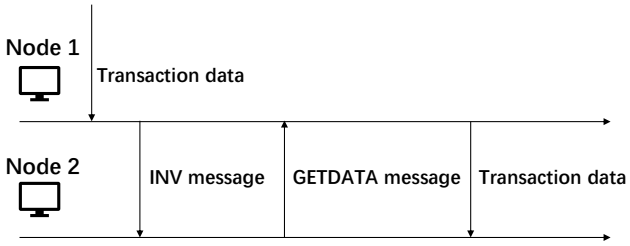



Fig. 2. Transaction propagation protocol between two nodes.

The simplest transaction propagation protocol is flooding. Each node first sends an INV message to all neighbors upon receiving a new transaction. It then sends the data to any neighbor sending a GETDATA messages. However, flooding exerts substantial pressure on two types of node: those with many neighbors and those close to the transaction proposer. Nodes with many neighbors must interact with all of them, while nodes close to the transaction proposer will receive and reply to GETDATA messages from most of its neighbors. Therefore, a small number of nodes may have to undertake much of the work of transmitting specific transactions to all nodes in the network, which makes propagation inefficient.

To improve the above protocol, we want to distribute the tasks of transaction propagation. In other words, we do not want any one node to reply to too many GETDATA messages. Given that each node has at most eight outbound connections, we restrict each node to replying to at most eight GETDATA messages. That means each node only forwards a specific transaction to eight neighbors at most, thereby potentially distributing large amounts of work to other nodes.

### 6.3 Transaction Forwarding Protocol

Our reputation-based mechanism accelerates propagation in the network. The method focuses on the selection and prioritization of forwarding targets during propagation. When a node is seeking a recipient for the propagation, a neighbor

with a higher reputation has a higher priority for being selected as the recipient.

More specifically, when a node decides to forward a transaction or a block, it sends an INV message to all its neighbors. After receiving the INV message, any neighbor requiring the data sends the GETDATA message. The first node replies to the eight most-reputable neighbors among all those that sent GETDATA messages.

There are two reasons why this strategy does not simply involve sending transactions to the nodes with the highest reputation. First, because the node only forwards the transaction to neighbors that have not received the transaction, this strategy will eventually enable all nodes to receive the transaction. Second, during data exchange, each node adjusts its neighbors' reputations. This would alter its prioritization of forwarding.

## 7 Data Request Protocol

Nodes in blockchain networks will request blockchain data or other services from other nodes when needed. In most existing blockchains, nodes respond spontaneously to data requests. However, as the size and scale of the blockchain grow, data request protocols will eventually require attention. A proper data request protocol assists nodes in selecting interaction objects, improving efficiency, and resisting attacks. This section discusses the reputation-based data request protocol.

### 7.1 Data Request in a Blockchain Network

In blockchain networks, a node that stores all block data is called a full node. Other nodes that are not full nodes need to interact with others to exchange the block data. When a node needs some block data, it should interact with other nodes by sending data requests. However, when facing many data requests, the limited bandwidth and huge communication overhead impede nodes from responding to all requests. Moreover, attackers can issue millions of fake requests, which may overwhelm nodes and make them fail to serve others. Therefore, we propose a data request protocol based on the reputation-based mechanism.

### 7.2 Two-Phase Data Request Protocol

Our data request protocol includes two phases. In the first, a node sends a data request to its neighbors. If the node has sufficient reputation from the perspective of its neighbors, those neighbors with the data will send them. If there is no reply, the node will enter the second phase: it sends a request to some nodes in the network. If the receiving nodes can provide data, they will issue their PoW puzzles. The requesting node can receive the data after completing the PoW puzzles.

#### *7.2.1 The first phase*

A pair of nodes with a stable interaction during transaction transmission will usually have a stable mutual reputation evaluation: they trust each other. Therefore, requests from nodes with stable reputation values will be accepted.

Assume that node $i$ tries to request data from its neighbors. It can send requests to multiple nodes. From the perspective of neighbor $j$, if the reputation of node $i$ is high enough, it will send a response to indicate that it can provide the data. After receiving replies from the neighbors, node $i$ can choose a subset of the neighbors from which to obtain the data.

#### *7.2.2 The second phase*

In many cases, a node would be unable to acquire the required data during the first phase. For example, a new node requests the latest block of the blockchain from existing nodes in the network, but the existing nodes do not have reputation records of the new node. To make up for this disadvantage, we design the second phase for nodes with a low reputation to obtain the required services at a cost.

Consider node $i$ requesting a specified data set from node $j$. If node $j$ has the corresponding data, it will generate a PoW puzzle, whose difficulty is proportional to the size of the data set, and send the puzzle to node $i$. Once node $i$ returns a correct solution to the puzzle, node $j$ will provide the requested data set to $i$ after proving the correctness.

Node $i$ may send multiple requests for the same data set to different nodes to guarantee that at least one node has the requested data. When receiving PoW puzzles from multiple nodes, node $i$ chooses to solve the puzzle from one of the most-reputable nodes. Specifically, for a new entrant with no reputation records, it could choose one of the responders at random. Node $i$ may not solve all received puzzles. As the cost of puzzle generation is low, node $i$ would not be punished if it does not solve received puzzles.

### 7.3 PoW Puzzle

In the second phase, before a node provides data, the requesting node must solve a PoW puzzle. A PoW puzzle is often a function that is hard to invert. A well-known PoW algorithm is hashcash [39]. When enumerating the solution to the hash function, mismatched results are discarded after testing, as they are useless. To make the best use of resources, nodes can design a PoW puzzle such that the whole computing process is meaningful. For example, the PoW puzzle can be designed to compute the discrete logarithm of an element in a cyclic group [40]. It is meaningful as the hardness of the elliptic curve discrete log problem is important in cryptography. BOINC [41] is another example of computational resource usage. It collects the distributed computational resources from independent nodes and uses them to compute large-scale problems cooperatively.

Our second-phase protocol can apply a variety of PoW mechanisms. Only two nodes—the sender and the receiver of the service—participate in each data request process and reach a consensus in our environment. Unlike strict PoW consensus for all nodes in the blockchain, two nodes can use various non-strict but scalable PoW puzzles to reach a consensus. The advantage of the simple PoW consensus is that it does not need to store strict consensus proofs, does not occupy space on the blockchain, and has strong scalability.

We also show that an attacker has no motivation to waste a requestor's computational resources by abusing the data request protocol or providing incorrect data. First, potential data providers are actively selected by node $i$, but an attacker cannot actively find requests, preventing the attacker

from sustaining an attack against a certain node. Second, a requesting node can verify the correctness of received data by requesting hash values from other peers, making incorrect data easily detected. Third, an attacker cannot impose an unreasonably large computation burden for one data request. As there is often more than one node in the blockchain network that can provide the same data, the data requester will choose to solve puzzles with reasonable costs. Overall, attackers cannot obtain stable and large profits by providing incorrect data, and the risk of being discovered is high.

## 8 Performance Evaluation

This section provides the results of simulations conducted as multiple sets of trials in various environments.

### 8.1 Simulation Settings

This subsection describes our simulation settings. For the convenience of simulation, we discretize time into periods of fixed length. The given results are for 200 such time slots, as all measured performance data subsequently change very little.

#### 8.1.1 Generating transactions

Each node has a 1% probability of generating a new transaction in any time slot. Honest and lazy nodes only generate VC transactions. A malicious node will generate equal amounts of VI and invalid transactions only. Nodes that generate some VC transactions are acceptable in some networks, but nodes that do not generate any VC transactions are unacceptable in every network. To make our simulations generally applicable, the malicious nodes do not generate any VC transactions.

We next introduce the method to generate the cycle cost of transactions. The cycle cost of each transaction is very similar to the gas, which is the pricing for running a transaction in Ethereum [3]. We crawled 388,691 transactions in Ethereum and used the gas number of these transactions as a reference to estimate the cycle cost in our simulations. Among the studied transactions, 157,967 transactions each had 21,000 gas[1]. About 86% of the crawled transactions had a gas number smaller than $10^5$, and no more than 0.5% had a gas number above $10^6$.

To generate transaction cycle costs in our simulations, one of the crawled transactions is selected at random, and its gas number is assigned as the cycle cost. For simplicity, any selected gas number greater than $10^6$ is assigned a cycle cost of $10^6$.

#### 8.1.2 Verifying transactions

Only honest nodes in these simulations verify transactions. When a node receives a transaction for the first time, it decides the probability of verifying it. All honest nodes here use the following verification probability function:

$$f(x) = \begin{cases} 1, & x < 0 \\ 1 - x/(4 \times 10^6), & 0 \le x < 3 \times 10^6 \\ 0.25, & x \ge 3 \times 10^6 \end{cases}$$

where $x$ is the sender's reputation value.

The function contains three cases:

- $x < 0$ indicates an unreliable sender whose sent transactions will certainly be verified.
- As a node continuously sends correct transactions, its reputation value will increase and its sent transactions need less frequent verification. Specifically, according to the statistics of transactions' cycle cost, the verification probability reduces by 0.5% every 20,000 cycles. In other words, it is set to be the linear function $1 - x/(4 \times 10^6)$.
- To avoid a possible accumulating reputation attack (see Section 5.3), the minimum verification probability is set as 25%. Even the most reliable sender will have a 0.25 probability of the receiver seeking validation.

#### 8.1.3 Transferring transactions

Nodes follow the rules in Section 3.3 to transfer transactions. As mentioned in Section 6, for each transaction, a node selects at most eight neighbors that have never received the transaction and forwards the transaction to them.

#### 8.1.4 Reputation attenuation

For any two nodes $i$ and $j$, we assume that the reputation $R_{ij}$ becomes stable after 100 time slots. As discussed in Section 4.1.3, $p_{ij} = 1 - t_{ij}/100$. We choose $p_{ij} = 0.9$ and $t_{ij} = 10$. The reputation value $R_{ij}$ is thus adjusted by node $i$ every 10 time slots to $R_{ij} = R_{ij} - \left\lfloor \frac{R_{ij}}{10} \right\rfloor$.

#### 8.1.5 Test graph and node types

We apply the Watts–Strogatz model [42] to generate $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, in which $\mathcal{V}$ is the node set and $\mathcal{E}$ is the connection set. In our simulations, $\mathcal{G}$ satisfies $|\mathcal{V}| = 2000$, $|\mathcal{E}| = 20000$, and we rewire each connection with the probability of $\beta = 0.5$. This model can generate graphs revealing the properties of real-world social networks.

We run simulations in two series of environments. The first comprises three sets of simulations with 60%, 70%, or 80% honest nodes. The rest are malicious. (There are no lazy nodes.) The second series of three simulations has half of the nodes honest, and 10%, 20%, or 30% lazy nodes. The remainder are malicious. Each simulation runs 10 times, and average results are used to draw figures. The different node types are evenly distributed in our simulations. As we conduct several simulations with differing proportions of each node type, a particular node might have different type in different simulations. For simplicity, the honest, lazy, and malicious nodes are respectively labeled as *H nodes*, *L nodes*, and *M nodes*.

1. 21,000 is the standard gas limit for regular transactions in Ethereum.

## 8.2 Simulation Results

### 8.2.1 Spread of invalid transaction

To measure the spread of invalid transactions, we define the *spread ratio* for each transaction. As our mechanisms and protocols only value the interests of honest nodes, the spread ratio of a transaction represents the fraction of all the honest nodes that receive it.

Fig. 3 shows the cumulative distribution of the spread ratio for invalid transactions in different environments. The total invalid transaction ratio is the fraction of all initial transactions that are invalid. Its lowest value on the $x$-axis is 0.6, because over 60% of invalid transactions would be blocked by any honest node receiving them. The results in Fig. 3(a) are environments without lazy nodes, and those in Fig. 3(b) are for environments with 50% honest nodes. Invalid transactions appear to spread less when the proportion of malicious nodes is higher. This occurs because an increase in the proportion of malicious nodes reduces the proportion of VC transactions. As any malicious node would then forward fewer VC transactions, its reputations would decline more quickly, and its sent invalid transactions would be discovered sooner. In all cases in Fig. 3, no invalid transaction spreads to over 10% of honest nodes, and over 90% of invalid transactions spread to less than 5% of honest nodes. Therefore, our mechanism can effectively prevent the spread of invalid transactions.

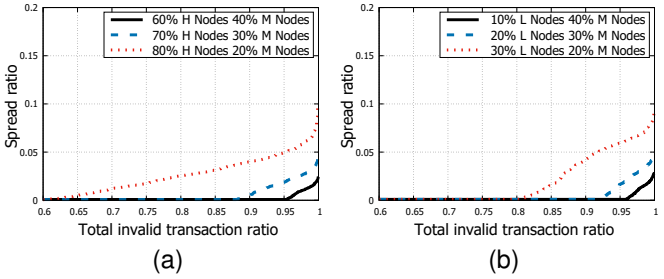



Fig. 3. Spread of invalid transactions among honest nodes on graph $\mathcal{G}$ in environments (a) without lazy nodes and (b) with half of nodes honest.

### 8.2.2 Changes in reputation and connections

Fig. 4(a)(c) shows the progression of average reputation scores from the perspective of honest nodes in environments without lazy nodes on graph $\mathcal{G}$. In all cases in Fig. 4(a), honest nodes rapidly gain reputation, and their scores stabilize between $2 \times 10^6$ and $2.5 \times 10^6$. The jaggedness is due to reputation attenuation. Fig. 4(c) shows an initial brief rise in average reputation for malicious nodes. A subsequent drop is followed by stabilization below $-1.0 \times 10^5$. The rise is faster and falls slower when the proportion of malicious nodes is lower. This is because increasing the proportion of malicious nodes allows each of them to transfer more invalid transactions, thereby decreasing their reputations.

To measure the changes in connections made by honest nodes, we focus on the proportion of current connections of a certain type relative to the initial number of connections of that type. As our concern is honest nodes, the considered connection types include at least one honest node. The progression of the ratio reflects changes in the nodes that

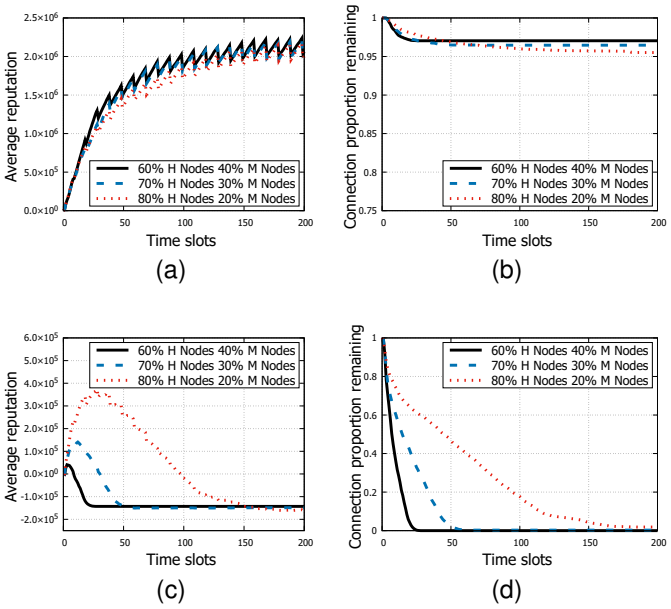



Fig. 4. Progression of average reputation for (a) honest and (c) malicious nodes from the perspective of honest nodes on graph $\mathcal{G}$ in environments without lazy nodes. Proportion of connections (relative to initial conditions) maintained between (b) two honest nodes and (d) one honest and one malicious node.

honest nodes select to be their neighbors. Fig. 4(b)(d) show results in environments without lazy nodes on graph $\mathcal{G}$. The honest nodes retain over 95% of their connections with other honest nodes, while almost all connections with malicious nodes are lost. This implies our reputation-based mechanism allows honest nodes to detect malicious nodes among their neighbors. As the proportion of malicious nodes increases, the speed of their disconnection from honest nodes increases in a trend matching their declining average reputation. However, the graphs do not imply that the number of remaining connections to malicious nodes is greater at low proportions of malicious nodes, as there were fewer malicious nodes to start.

Fig. 5 shows the results for environments with 50% honest nodes on graph $\mathcal{G}$. For this fixed ratio of honest nodes, increasing the proportion of malicious nodes increases the speed at which they are found. This is similar to the results in Fig. 4. The similarity of Fig. 5(d) and Fig. 5(f) indicates that lazy and malicious nodes are excluded similarly quickly. This is because the vast majority of transactions sent by each node are transactions from other nodes. Invalid transactions generated by a malicious node itself represent only a small proportion of all the transferred transactions. As lazy nodes and malicious nodes do not verify transactions, they transfer similar proportions of invalid transactions, and are thus similarly undesirable from the perspective of honest nodes.

### 8.2.3 Versatility on different graphs

To test the versatility of our mechanism for different graphs, we consider another random graph model called the power-law random graph [43]. We apply the algorithm given by Volchenkov and Blanchard [44] to generate $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$, in which $\mathcal{V}'$ is the node set and $\mathcal{E}'$ is the connection set. In our

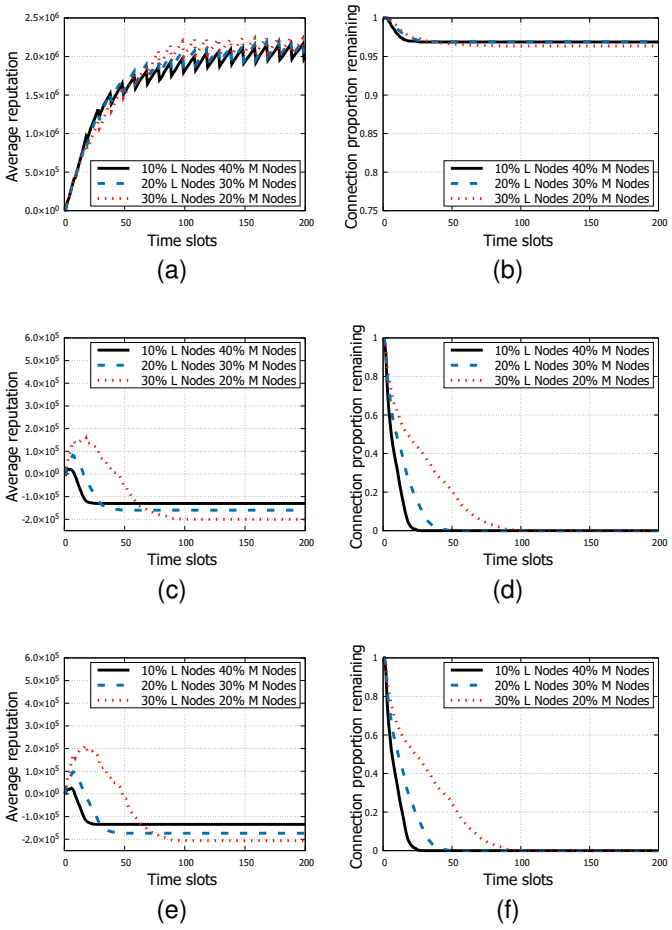



Fig. 5. Progression of average reputation for (a) honest, (c) malicious, and (e) lazy nodes from the perspective of honest nodes on graph $\mathcal{G}$ in environments with 50% honest nodes. Proportion of connections (relative to initial conditions) maintained between one honest node and (b) another honest node, (d) one malicious node, and (f) one lazy node.

simulations, $\mathcal{G}'$ satisfies $|\mathcal{V}'| = 2000$ and $|\mathcal{E}'| = 18229$. This algorithm can generate scale-free power-law random graphs that follow the topology of complex "real-world" networks.

Fig. 6 visualizes the spread of invalid transactions during simulations run on graph $\mathcal{G}'$. The simulations were equivalent to those run on graph $\mathcal{G}$, and the corresponding results are similar to those in Fig. 3. None of the cases in Fig. 6 have invalid transactions spreading to over 10% of honest nodes, and over 90% of invalid transactions spread to less than 5% of honest nodes. Thus, our mechanism can effectively prevent the spread of invalid transactions on both graphs $\mathcal{G}$ and $\mathcal{G}'$. The results in Fig. 7 for environments without lazy nodes on graph $\mathcal{G}'$ show reputation declines and disconnections for malicious nodes similar to those in Fig. 4. Overall, although the simulations are run on graphs generated by different algorithms, running simulations with graphs of similar scale and node proportions leads to similar results, thus indicating the versatility of our mechanism.

#### 8.2.4 *Propagation acceleration*

To test our reputation-based transaction forwarding protocol for propagating transactions in a network, we compare the following three forwarding strategies.

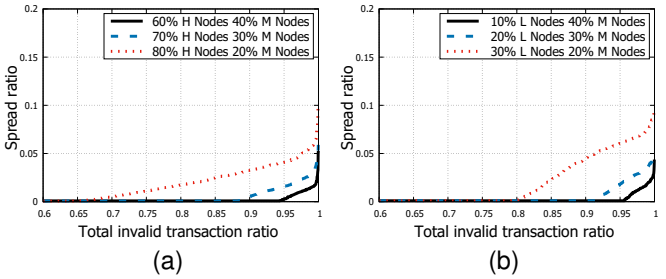



Fig. 6. Spread of invalid transactions among honest nodes on graph $\mathcal{G}'$ in environments (a) without lazy nodes and (b) with 50% honest nodes.

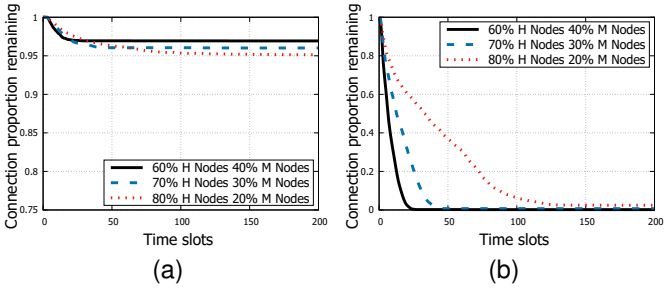



Fig. 7. Proportion of connections (relative to initial conditions) maintained on graph $\mathcal{G}'$ (in environments without lazy nodes) between one honest node and (a) another honest node and (b) one malicious node.

- Reputation strategy: Nodes forward a transaction to the eight most-reputable peers that do not have it.
- Random strategy: Nodes forward a transaction to eight random peers that do not have it.
- Mixed strategy: Nodes forward a transaction to the four most-reputable peers and four random peers that do not have it.

The environment for testing acceleration performance has 80% honest and 20% malicious nodes. The speed of propagating transactions is recorded as the time slots needed for each transaction to be received by at least 80% of honest nodes.

Fig. 8 shows the cumulative distribution of time slots required for at least 80% of honest nodes to receive transactions. Fig. 8(a) and (b) differ in maximum network bandwidth. The figure shows that the reputation strategy performs better than the random or mixed strategies, requiring about half of the time needed by the random strategy in 8(b).

## 9 Conclusion and Future Work

This paper proposes a holistic reputation-based mechanism derived from monitoring the quality of transactions from neighbors. Each node locally maintains a list of its neighbors' reputations derived from the verification results of transactions received from them. We have sought broad applicability of the reputation-based mechanism in the blockchain environment. The proposed strategy aims to prevent the spread of spam transactions while reducing nodes' verification workloads. We also optimized the transaction forwarding protocol to reduce propagation delay, and

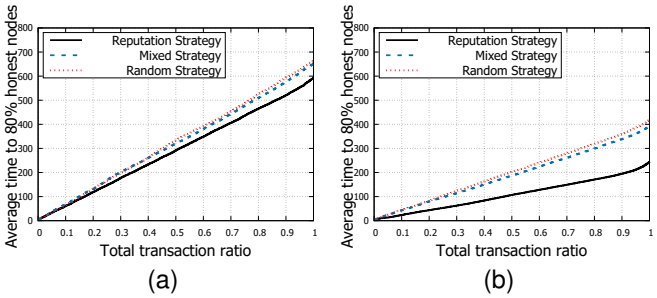



Fig. 8. Time slots needed for transactions to be received by at least 80% of honest nodes (i.e., 64% of all nodes in the test environment with 80% honest and 20% malicious nodes). Each node can transfer any transaction to any peer at most (a) 32 times or (b) 64 times in one time slot.

proposed a two-phase protocol for data exchange between blockchain participants. A series of simulations demonstrated the performance of our reputation-based mechanism and our proposed applications. The results show that our reputation-based mechanism can prevent the spread of invalid transactions, allow honest nodes to efficiently identify lazy and malicious nodes, adapt to different environments, and accelerate the speed of propagating transactions.

Strategic verification significantly reduces the number of verifications, but it does allow some spam transactions to be spread in the network, especially compared with carefully verifying all transactions. In addition, attackers might be able to flood the system with spam transactions for a short period after accumulating a high reputation. However, our simulation results provide compelling evidence that they will be effectively identified in the long-term. Note that the spam transactions will not pose threats to the system's security, as miners in the consensus layer will verify each transaction carefully before generating blocks.

Our future work will explore the relationship between fault-tolerance accuracy $\epsilon$ and verification probability complexity. Developing the reputation-based mechanism to aggregate more variables, thus establishing relationships among nodes' properties (such as capability, network bandwidth, and computational resources), is a substantial challenge. We also intend to discuss the numerical and function designs. We anticipate a huge amount of scenarios where reputation techniques can be applied. We will promote the reputation-based mechanism as a creative and efficient solution to the current concerns facing blockchains.

## Acknowledgement

This work is supported in part by the National Natural Science Foundation of China (Nos. 11871366 and 61761146005), the Qing Lan Project for Young Academic Leaders, and the Qing Lan Project for Key Teachers.

## References

[1] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," 2014. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper

[3] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[4] B. Xu, D. Luthra, Z. Cole, and N. Blakely, "Eos: An architectural, performance, and economic analysis," 2018.

[5] Q. Xu, C. Jin, M. F. B. M. Rasid, B. Veeravalli, and K. M. M. Aung, "Blockchain-based decentralized content trust for docker images," *Multimed. Tools Appl.*, vol. 77, no. 14, pp. 18 223–18 248, 2018.

[6] Z. Liu, N. C. Luong, W. Wang, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "A survey on applications of game theory in blockchain," *arXiv preprint arXiv:1902.10865*, 2019.

[7] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," *Commun. ACM*, vol. 61, no. 7, pp. 95–102, 2018.

[8] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Annu. Int. Conf. Theory Appl. Cryptogr. Tech.*, 2015, pp. 281–310.

[9] R. Pass and E. Shi, "Fruitchains: A fair blockchain," in *Proc. ACM Symp. Princ. Distrib. Comput.*, 2017, pp. 315–324.

[10] L. Luu, J. Teutsch, R. Kulkarni, and P. Saxena, "Demystifying incentives in the consensus computer," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 706–719.

[11] B. B. F. Pontiveros, C. F. Torres *et al.*, "Sluggish mining: Profiting from the verifier's dilemma," in *Int. Conf. Financ. Cryptogr. Secur.*, 2019, pp. 67–81.

[12] K. Baqer, D. Y. Huang, D. McCoy, and N. Weaver, "Stressing out: Bitcoin 'stress testing'," in *andInt. Conf. Financ. Cryptogr. Data Secur.*, 2016, pp. 3–18.

[13] S. Paavolainen, T. Elo, and P. Nikander, "Risks from spam attacks on blockchains for internet-of-things devices," in *2018 IEEE 9th Annu. Info. Technol., Electron. Mob. Commun. Conf. (IEMCON)*, 2018, pp. 314–320.

[14] L. Reyzin, D. Meshkov, A. Chepurnoy, and S. Ivanov, "Improving authenticated dynamic dictionaries, with applications to cryptocurrencies," in *Int. Conf. Financ. Cryptogr. Data Secur.*, 2017, pp. 376–392.

[15] M. Signorini, M. Pontecorvi, W. Kanoun, and R. Di Pietro, "Bad: Blockchain anomaly detection," *arXiv preprint arXiv:1807.03833*, 2018.

[16] M. Saad, M. T. Thai, and A. Mohaisen, "Poster: Deterring ddos attacks on blockchain-based cryptocurrencies through mempool optimization," in *Proc. 2018 Asia Conf. Comput. Commun. Secur.*, 2018, pp. 809–811.

[17] M. Saad, L. Njilla, C. Kamhoua, J. Kim, D. Nyang, and A. Mohaisen, "Mempool optimization for defending against ddos attacks in pow-based blockchain systems," in *2019 IEEE Int. Conf. Blockchain and Cryptocurrency (ICBC)*, 2019, pp. 285–292.

[18] J. Yang, T. Li, G. Liang, Y. Wang, T. Gao, and F. Zhu, "Spam transaction attack detection model based on gru and wgan-div," *Comput. Commun.*, vol. 161, pp. 172–182, 2020.

[19] P. Resnick and R. Zeckhauser, "Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system," in *The Economics of the Internet and E-commerce*. Emerald Group Publishing Limited, 2002, pp. 127–157.

[20] K. Aberer and Z. Despotovic, "Managing trust in a peer-2-peer information system," in *Proc. 10th Int. Conf. Inf. Knowl. Manag.*, 2001, pp. 310–317.

[21] S. Buchegger and J.-Y. Le Boudec, "A robust reputation system for mobile ad-hoc networks," Tech. Rep., 2003.

[22] M. Srivatsa, L. Xiong, and L. Liu, "Trustguard: Countering vulnerabilities in reputation management for decentralized overlay networks," in *Proc. 14th Int. Conf. on World Wide Web*, 2005, pp. 422–431.

[23] D. Carboni, "Feedback based reputation on top of the bitcoin blockchain," *arXiv preprint arXiv:1502.01504*, 2015.

[24] G. Fortino, F. Messina, D. Rosaci, and G. M. Sarné, "Using blockchain in a reputation-based model for grouping agents in the internet of things," *IEEE Trans. Eng. Manag.*, vol. 67, no. 4, pp. 1231–1243, 2019.

[25] D. Liu, A. Alahmadi, J. Ni, X. Lin, and X. Shen, "Anonymous reputation system for iiot-enabled retail marketing atop pos blockchain," *IEEE Trans. Ind. Inform.*, vol. 15, no. 6, pp. 3527–3537, 2019.

[26] K. N. Khaqqi, J. J. Sikorski, K. Hadinoto, and M. Kraft, "Incorporating seller/buyer reputation-based system in blockchain-enabled emission trading application," *Appl. Energy*, vol. 209, pp. 8–19, 2018.

[27] B. Shala, U. Trick, A. Lehmann, B. Ghita, and S. Shiaeles, "Novel trust consensus protocol and blockchain-based trust evaluation system for m2m application services," *Internet of Things*, vol. 7, p. 100058, 2019.

[28] F. Gai, B. Wang, W. Deng, and W. Peng, "Proof of reputation: A reputation-based consensus protocol for peer-to-peer network," in *Int. Conf. Database Syst. Adv. Appl.*, 2018, pp. 666–681.

[29] H. Chai, S. Leng, K. Zhang, and S. Mao, "Proof-of-reputation based-consortium blockchain for trust resource sharing in internet of vehicles," *IEEE Access*, vol. 7, pp. 175 744–175 757, 2019.

[30] J. Li, G. Liang, and T. Liu, "A novel multi-link integrated factor algorithm considering node trust degree for blockchain-based communication," *KSII Tran. on Internet & Info. Syst. (TIIS)*, vol. 11, no. 8, pp. 3766–3788, 2017.

[31] M. Nojoumian, A. Golchubian, L. Njilla, K. Kwiat, and C. Kamhoua, "Incentivizing blockchain miners to avoid dishonest mining strategies by a reputation-based paradigm," in *Sci. Info. Conf.* Springer, 2018, pp. 1118–1134.

[32] J. Yu, D. Kozhaya, J. Decouchant, and P. Esteves-Verissimo, "Repucoin: Your reputation is your power," *IEEE Tran. Comput.*, vol. 68, no. 8, pp. 1225–1237, 2019.

[33] M. T. de Oliveira, L. H. Reis, D. S. Medeiros, R. C. Carrano, S. D. Olabarriaga, and D. M. Mattos, "Blockchain reputation-based consensus: A scalable and resilient mechanism for distributed mistrusting applications," *Comput. Netw.*, vol. 179, p. 107367, 2020.

[34] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Elsevier, 2011.

[35] A. M. Turing, "On computable numbers, with an application to the Entscheidungsproblem," *Proc. Lond. Math. Soc.*, vol. 2, no. 1, pp. 230–265, 1937.

[36] Github, "Bitcoin core," 2021. [Online]. Available: https://bitcoincore.org/

[37] Tradeblock, "Historical data," 2021. [Online]. Available: https://tradeblock.com/bitcoin/historical

[38] BitcoinProject, "P2p network - bitcoin," 2021. [Online]. Available: https://developer.bitcoin.org/reference/p2p_networking.html

[39] A. Back *et al.*, "Hashcash-a denial of service counter-measure," 2002.

[40] M. Hastings, N. Heninger, and E. Wustrow, "Short paper: The proof is in the pudding," in *International Conference on Financial Cryptography and Data Security*. Springer, 2019, pp. 396–404.

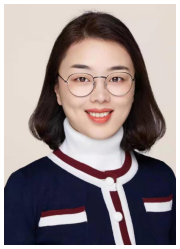
[41] D. P. Anderson, "Boinc: A system for public-resource computing and storage," in *Proc. 5th IEEE/ACM Int. Workshop Grid Comput.* IEEE Computer Society, 2004, pp. 4–10.

[42] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[43] W. Aiello, F. Chung, and L. Lu, "A random graph model for power law graphs," *Exp. Math.*, vol. 10, no. 1, pp. 53–66, 2001.

[44] D. Volchenkov and P. Blanchard, "An algorithm generating random graphs with power law degree distributions," *Physica A: Stat. Mech. Appl.*, vol. 315, no. 3–4, pp. 677–690, 2002.

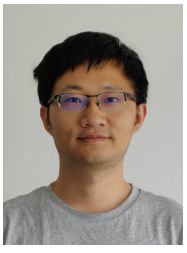

**Jiarui Zhang** received the BEng degree in computer science and technology from Shanghai Jiaotong University in 2017. He is currently working towards the PhD degree in computer engineering at Stony Brook University. His research interests include blockchain and mobile edge computing.



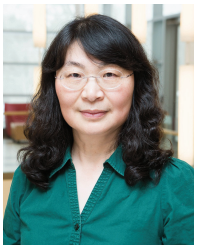
**Yukun Cheng** received the PhD degree from Shanghai University in 2010 and did her post-doctoral studies at College of Computer Science and Technology at Zhejiang University from 2010 to 2013. She is a professor at Suzhou University of Science and Technology. Her current research interests include the mechanism design problem in algorithmic game theory, internet market design, and combinatorial optimization.

**Xiaotie Deng** received his BSc from Tsinghua University, MSc from Chinese Academy of Sciences, and PhD from Stanford University in 1989. He is currently a chair professor at Peking University. He taught in the past at Shanghai Jiaotong University, University of Liverpool, City University of Hong Kong, and York University. Before that, he was an NSERC International Fellow at Simon Fraser University. Dengs current research focuses on algorithmic game theory, with applications to internet economics and finance. His works cover online algorithms, parallel algorithms, and combinatorial optimization. He is an ACM Fellow for his contribution to the interface of algorithms and game theory, and an IEEE Fellow for his contributions to computing in partial information and interactive environments.

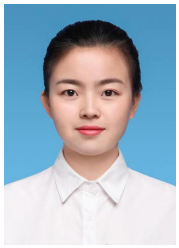
**Bo Wang** received a Master degree in theoretical physics from Peking University in 2010, and another Master degree in electrical engineering from Zhejiang University in 2006. He is currently the founder of Nervina Labs, and a researcher of Nervos Foundation. His research interests include blockchain economics, monetary theory, and artificial intelligence.



**Jan Xie** received the BEng degree in computer science and technology from Zhejiang University. He was a researcher working on Casper and sharding in the Ethereum research team. He led the development of the first open-source exchange Peatio, the largest Ethereum mining pool Sparkpool, and the high performance permissioned blockchain CITA. He is the founder of Cryptape and the architect of Nervos Network. His current research interests include blockchain scalability, smart contract model, cryptoeconomics and mechanism design.



**Yuanyuan Yang** received the BEng and MS degrees in computer science and engineering from Tsinghua University, Beijing, China, and the MSE and PhD degrees in computer science from Johns Hopkins University, Baltimore, Maryland. She is a SUNY Distinguished Professor of computer engineering and computer science at Stony Brook University, New York, and is currently on leave at the National Science Foundation as a Program Director. Her research interests include edge computing, data center networks, cloud computing and wireless networks. She has published about 470 papers in major journals and refereed conference proceedings and holds seven US patents in these areas. She is currently the Editor-in-Chief for IEEE Transactions on Cloud Computing and an Associate Editor for ACM Computing Surveys and IEEE Transactions on Parallel and Distributed Systems. She has served as an Associate Editor-in-Chief and Associated Editor for IEEE Transactions on Computers. She has also served as a general chair, program chair, or vice chair for several major conferences and a program committee member for numerous conferences. She is an IEEE Fellow.



**Mengqian Zhang** received her B.E. degree in computer science from Ocean University of China in 2018. She is currently pursuing a doctorate at Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. Her current research interests include blockchain, algorithmic game theory, and mechanism design.