Accelerating Transactions Relay in Blockchain Networks via Reputation

Mengqian Zhang*, Yukun Cheng†, Xiaotie Deng‡, Bo Wang§, Jan Xie¶, Yuanyuan Yang‖ and Jiarui Zhang**
*Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China
†School of Business, Suzhou University of Science and Technology, Suzhou, 215009, China
‡Center on Frontiers of Computing Studies, Computer Science Dept., Peking University, Beijing, 100871, China
§Nervina Labs Ltd., Hangzhou, 310018, China
¶Cryptape Technology Co., Ltd., Hangzhou, 310012, China
‖**Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794, USA
Email: *mengqian@sjtu.edu.cn, †ykcheng@amss.ac.cn, ‡xiaotie@pku.edu.cn, §cipher@nervos.org, ¶jan@cryptape.com,
{‖yuanyuan.yang, **jiarui.zhang.2}@stonybrook.edu

***Abstract*—For a blockchain system, the network layer is of great importance for scalability and security. The critical task of blockchain networks is to provide a fast delivery of data. A rapid spread accelerates the transactions to be included into blocks and then confirmed. Existing blockchain systems, especially the cryptocurrencies like Bitcoin, take a simple strategy that requires relay nodes to verify all received transactions and then forward valid ones to all outbound neighbors. Unfortunately, this design is inefficient and slows down the transmission of transactions. In this paper, we introduce the concept of reputation and propose a novel relay protocol, *RepuLay*, to accelerate the transmission of transactions across the network. First of all, we design a reputation mechanism to help each node identify the unreliable and inactive neighbors. In this mechanism, two values are used to define one's reputation. Each node keeps a local list of reputations of all its neighbors. Based on the reputation mechanism, *RepuLay* adopts probabilistic strategies to process transactions. More specifically, after receiving a transaction, the relay node verifies it with a certain probability, which is deduced from the first value of sender's reputation. Next, the valid and unverified transactions are forwarded to some neighbors. Each neighbor has some probability to be chosen as a receiver and the probability is determined by its second value of reputation. Theoretically, we prove that our design can guarantee the quality of relayed transactions. Further simulation results confirm that *RepuLay* effectively accelerates the spread of transactions and optimize the usage of nodes' bandwidths.**

***Index Terms*—blockchain, data transmission, reputation mechanism, network protocol**

I. Introduction

The blockchain technology has experienced significant growth in innovation over the past decade [1]. Generally speaking, it is a distributed data ledger maintained by the nodes in the system. Some special nodes (known as *miners*) compete to pack up valid transactions into blocks. All blocks are linked in chronological order and the chain grows in an append-only manner. Structured as a peer-to-peer (P2P) architecture over the Internet, nodes in a blockchain system shall reach an agreement on the state of the chain, according to the specific consensus protocol that the system adopts.

978-0-7381-3207-5/21/$31.00 ©2021 IEEE

To keep a blockchain system running, information exchange plays a critical role. Like many other P2P networks, most blockchain systems broadcast blocks and pending transactions among peers by flooding [2]. Take the pioneering design bitcoin [3] as an example. By default, a normal node makes 8 outbound connections and receives up to 125 inbound connections. After receiving a transaction or block from an inbound neighbor, the relay node process it in two steps. First it validates the correctness of data. If valid, the relay node then forwards it to all outbound neighbors.

As the first step, whether to verify the received transactions or not is a dilemma facing the relay nodes. On the one hand, the openness and anonymity of blockchain network open the door to the selfish or even malicious behaviors, such as the Denial-of-Service (DoS) attacks [4]. In order to filter out invalid data in time, it is necessary for relay nodes to do some data screening before forwarding. On the other hand, users who generate the transactions expect low latency and prefer to skip the verification process, thus helping their transactions to be received by miners as early as possible. Currently, it is still an open research challenge to balance well the DoS prevention and non-verification requirement [5]. A proper solution would bring a faster transactions spread and better user experience.

For the second step, the simple strategy that relay nodes just forward data to outbound neighbors poses a risk to limit the data spread, as malicious neighbors may refuse further transmission. Differentiating the inbound from outbound neighbors is based on the following consideration: an outbound neighbor that the node proactively discovered is less likely to behave badly than a passively-connected inbound neighbor. Yet, with non-negligible probability, the outbound neighbors may be attackers. Previous studies [6], [7] show that by denying the delivery of data to victim Bitcoin nodes, attackers can delay the transmission of blocks and transactions for a considerable amount of time. Therefore, it is also a crucial issue to decide the receivers carefully when forwarding data.

In this paper, we introduce the concept of reputation to tackle the aforementioned problems, aiming to accelerate the

data transmission while without giving up quality of the relayed data in blockchain networks. The reputation-based approach has been widely used in P2P applications [8], [9]. A proper reputation mechanism can identify malicious behaviors [10], motivate cooperation among peers [11] and improves the systems' performance [12], [13]. In recent years, researchers in the area of blockchain also borrow the idea of reputation, for instance, to prevent the spread of spam transactions [14] and to provide incentives [15].

Specifically, we design a reputation mechanism to quantify peers' credibility by recording their historical behaviors. By the mechanism, each node could compute its neighbors' reputation. One's reputation is a tuple consisting of two values $R = \langle r_1, r_2 \rangle$. What the first value r_1 reflects is the accuracy of one's relayed data. According to r_1, the unreliable neighbors can be distinguished from others. The second value r_2 of one's reputation is its net number of data transmission. Based on r_2, the inactive neighbors could be found out.

With the help of the reputation mechanism, we propose *RepuLay*, a novel protocol aiming at speeding up the transactions relay in blockchain networks. More specifically, after receiving a transaction, the relay node verifies it with a certain probability, which is deduced from the first value of the sender's reputation (*i.e.*, r_1). During the forwarding process, the relay node probabilistically selects a certain number of nodes as receivers from all its neighbors, on the basis of the second value of their reputation (*i.e.*, r_2).

Moreover, we conduct a series of simulations to evaluate our designs. The simulation results show that nodes with different behaviors can be clearly distinguished by reputation. With respect to *RepuLay*, on the one hand, it effectively speeds up the data propagation in blockchain systems. On the other hand, it succeeds in saving bandwidth for nodes in the network.

In a word, the main contributions of the paper are summarized as follows.

1) We design a reputation mechanism, by which each node could identify its unreliable and inactive neighbors.
2) We propose a reputation-based transactions relay protocol. The protocol guides nodes in the blockchain network to probabilistically verify received transactions and choose receivers.
3) Theoretically, we prove that our design can guarantee the quality of relayed transactions. Further simulation results confirm that our protocol could accelerate the data propagation in blockchain networks and optimize the usage of nodes' bandwidths.

The rest of the paper is organized as follows. In Section II we introduce the background and some related works. Section III formulates the network environment and demonstrates the problem we focus on. The reputation mechanism to distinguish different types of nodes is proposed in Section IV. Further, we present a reputation-based transactions relay protocol in Section V and analyze it in Section VI. In Section VII, we conduct a series of simulations to evaluate the protocol. Finally, we conclude this paper in Section VIII.

II. Background and Related Work

A. Transaction, Block and Blockchain

Blockchain, known as the distributed ledger, keeps a record of transactions in the system. In a narrow sense, transaction refers to the transfer of assets. Bitcoin [3], one of the most influential blockchain applications, creatively proposes a purely peer-to-peer electronic cash system. It allows online payments to be sent securely from one party to another without going through a trusted third party. By introducing the concept of smart contract [16], Ethereum [17] generalizes the meaning of transactions. It enables that any event causing the state transition can be formulated as a transaction.

For a variety of reasons, transactions in the system may conflict with each other. Taking the simple payment as an example, an adversary may attempt to spend the same asset twice or even more times, which is called the double-spending attack. When facing conflicting transactions, the block generator has the final say over which one to accept. Specifically, over a period of time, according to the adopted protocol (like *proof-of-work* in Bitcoin), one node (known as *miner* in Bitcoin) would gain the right to pick up transactions and package them as a new block. By containing the hash value of the previous block, all blocks are linked as a chain, *i.e.*, a blockchain.

B. Blockchain Network and Data Transmission

The blockchain system establishes on the base of P2P network, which is public to any user. When a node starts for the first time, it could get some IP addresses of existing active nodes from the seed nodes, whose information has been hardcoded in the client. Once the connection is made, it could ask the peer for further addresses. Limited by the bandwidths, it is recommended to make 8 outbound connections and receive up to 125 inbound connections in Bitcoin [18].

To keep a blockchain system running, nodes play different roles depending on the functionality they support [19]. A small quantity of *full nodes* hold full information of all agreed blocks and the whole history of transactions included. On the contrary, most nodes only keep a copy of the blocks' header [20], which are known as *light nodes*. All nodes exchange blocks and pending transactions in a gossip-like manner [1]. Specifically, after receiving certain data from inbound connections, a node $\mathcal{A}$ needs to first verify it by prescribed rules. Then for the valid ones, node $\mathcal{A}$ broadcasts it to all outbound neighbors by an advertisement-based manner. As Fig. 1 shows, a complete data exchange includes the following three interactions:

1) Node $\mathcal{A}$ send each outbound neighbors a INV message, which contains the hash value of newly received data;
2) The neighbor who can not find data locally (*e.g.*, node $\mathcal{B}$ in Fig. 1) responds a GETDATA message in turn;
3) Node $\mathcal{A}$ replies with the complete information of the transaction (tx) or block.

It is worth noticing that the local verification process depends on historical data, which contributes significantly to the overall propagation time [21]. In the blockchain system,

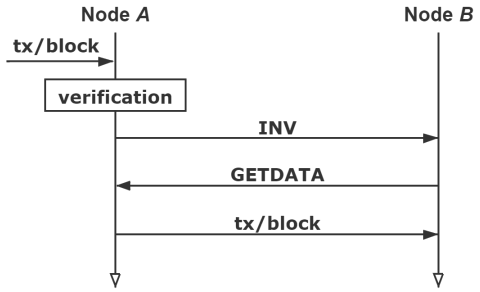

Fig. 1. Data exchange between peers.

only a few full nodes have the whole historical data. Most nodes have to ask the full nodes for help when needed and wait for reply, which reduces the speed of verification.

C. Existing Methods to Speed Up Data Propogation

In the network layer of blockchain, the point is data transmission, namely, to help valid transactions and blocks spread across the network as quickly as possible. [22] points out that propagation delay in the network is the primary cause for the blockchain fork, which is an inconsistent state of the system and should be avoided.

To date, several studies have focused on the methods to speed up the propagation of blocks. A popular design is to adjust the data structure, with the compact block [23] as a typical representative. Instead of sending the complete information, the compact block only announces the short transaction IDs. Then the receivers could reconstruct the block and process it as usual. If some transactions are not available from local sources, a round trip of GETDATA / tx is necessary as shown in Fig. 1. Graphene [24] further reduces the block overhead by using the Bloom filter and Invertible Bloom Lookup Tables. Another important work is the Fast Internet Bitcoin Relay Engine (FIBRE) [25], which is a protocol designed specially for miners and aims to provide high-speed transfer of blocks. To avoid some unwanted round-trip caused by packet losses, FIBRE protocol sends data with enough extra information such that receivers are able to reconstruct the original data.

Although aforementioned protocols significantly reduce block transfer latencies, researches on the transaction dissemination are unsatisfactory. It is suggested that increasing connectivity is an efficient way to speed up the propagation of transactions [22], as it could minimize the network hops between any two nodes. Yet, the required overheads on bandwidths are prohibitively expensive for most nodes. To reduce the transaction propagation delay, Bitcoin Clustering Based Ping Time protocol (BCBPT) [26] makes Bitcoin nodes to connect with those peers that are close in the topological term, based on ping latencies. However, this design leads to the structured network topology and further makes the system more vulnerable to the eclipse attacks [27]. In order to reduce the number of redundant messages, [28] proposes a probabilistic flooding approach. The probability to send INV messages to a certain neighbor is that the number of INV messages sent to the neighbor divided by the number of GETDATA messages received from it. Unfortunately, this straightforward design causes the previously highly-connected nodes suffering from a low probability of being sent an INV message when some of its neighbors leave the network. Erlay [29] combines limited flooding with intermittent set reconciliation to synchronize transactions among peers. It dramatically reduces the bandwidth used to announce transactions, but requires close cooperation of all nodes in the network.

In the following sections, we focus on the propagation of transactions, and use the terms *data* and *transactions* interchangeably.

III. Preliminaries

A. Threat Model

Considering that the network environment is rather complex, we analyze the involved transactions and nodes in this section. According to the validity of transactions, we classify them into two types.

- *Valid transaction*. A valid transaction meets all rules given by the system. It should be broadcasted to the whole network as quickly as possible.
- *Invalid transaction*. An invalid transaction can not pass the validation. It should be intercepted immediately to avoid further spread.

To keep the system running smoothly, nodes at the network plane are expected to do as follows: generate and forward only valid transactions, and respond the data requests from neighbors. Yet, like other P2P networks, several misbehaviors even attacks exist in the blockchain system. In this paper, we focus on the following two which are closely related to the data transmission.

- *Unreliable behavior*. The key feature of unreliable behavior is to send invalid data. It contains two possible cases. One is that the nodes carelessly forward invalid data from neighbors without any verification. The other case is that the node is precisely the original generator of the invalid data. For example, someone may create conflicting transactions to make a double-spending attack. Or attackers may generate an excessive amount of invalid transactions to launch the DDoS attacks. Whatever the reasons, the unreliable behavior leads to a further spread of invalid data, which will waste computational resources (*i.e.*, data storage, processing power, and bandwidth). More importantly, it cripples the network's ability to serve the valid data and slow down their transmission.
- *Inactive behavior*. The key feature of inactive behavior is to provide less data than others. More specifically, this kind of node looks inactive when forwarding the received data and responding to the data requests. Here are two possible reasons for this behavior. One cause is that the node deliberately refuses service to others, aiming to conserve its bandwidths or other selfish thoughts. The other possible reason is that the node is restricted

to some objective conditions. For instance, nodes with insufficient computation resources are unable to process and transmit data rapidly. Whether on purpose or not, the inactive behavior delays the ongoing data transmission, and should be detected and further solved.

For the sake of brevity, we call the node that behaves well, unreliably, inactively as honest, unreliable, and inactive node, respectively. It is worth noting that the classification is based on the nodes' behavior, rather than their identities. A node may switch among different behaviors. It is regarded as honest at the time of performing good behavior. Nevertheless, when it behaves unreliably, we treat it as an unreliable node.

B. Problem Definition

This work focuses on the network layer of blockchain systems, aiming to improve bandwidth utilization and speed up the data propagation in blockchain networks. The core of the network layer is to provide fast delivery of data. Relay nodes in the blockchain network shall help to broadcast each transaction from its generator to all full nodes. During the data transmission, speed is the key, as a fast transmission accelerates the transactions to be included into blocks and then confirmed. And the relay nodes, in theory, do not need to consider the data's quality. Miners in the consensus layer will verify each transaction later to ensure that only valid transactions have the chance to be packed into the block.

Current transmission protocol takes a simple strategy: relay nodes verify all received transactions and then forward valid ones to all outbound neighbors. As discussed earlier, this strategy suffers from the following drawbacks. First, repetitive verifications of all transactions (especially the valid ones) waste computational resources and slow down transmission. Second, the forwarding process gives the inactive neighbors the same chance to be chosen as receivers, which provide less help for further transmission. So to some extent, this simple protocol goes against the transmission task in the blockchain network, namely, to broadcast data as fast as possible.

In this paper, we speed up the data transmission by optimizing the verification process and forwarding process, respectively. In the first step, probabilistic verification is a better idea. Intuitively, when receiving transactions from honest nodes, fewer or no verification is needed before forwarding. On the contrary, when receiving some data from an unreliable node, the best choice is to verify them carefully. In the second step, relay nodes should avoid forwarding data to those neighbors behaving inactively, as they would send little or even no data to others.

As can be seen, the key task of the above thoughts is to distinguish different types of nodes. In order to address this issue, we introduce the concept of reputation to represent one's credibility. By recording and analyzing the neighbor's historical behaviors, the relay node could calculate its reputation, which helps to make better decisions on the data transmission.

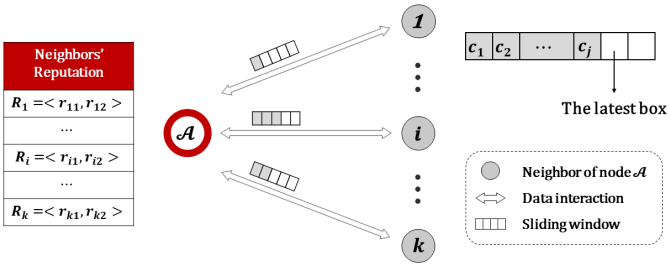

Fig. 2. Overview of the reputation mechanism. Node $\mathcal{A}$ keeps a reputation list of all neighbors locally. Each reputation is a tuple consisting of two values. One's reputation is determined by its data interactions with $\mathcal{A}$. Node $\mathcal{A}$ maintains a sliding window for each neighbor and writes a record in the latest box of the corresponding window when needed.

IV. Reputation Mechanism

In this section, we design a mechanism for each node to compute its neighbors' reputation. It is expected to identify nodes' behaviors and tell apart different types of neighbors by the reputation mechanism.

A. Reputation Overview

In our design, each node keeps a local list of all neighbors' reputations. Each reputation is a tuple consisting of two values $R = \langle r_1, r_2 \rangle$. Both values can be positive, 0, or negative. These two elements r_1 and r_2 respectively correspond to two factors we care about: quality and quantity. Simply put, quality is the correctness of relayed data. Considering that the unreliable nodes will send invalid transactions, the first value of their reputations (*i.e.*, r_1) will be relatively small. That is, this element could help tell apart the unreliable behavior. Quantity, as the second element, is the net number of data interactions. For those neighbors that behave inactively, their second value of reputation (*i.e.*, r_2) will be quite small. Thus this factor can effectively facilitate finding the inactive behavior. We roughly show the design of reputation in Fig. 2.

In the following sections, we elaborate the reputation mechanism from the perspective of an arbitrary node $\mathcal{A}$, whose neighbors are denoted by $P = \{1, \cdots, k\}$.

B. First Value of Reputation: Quality of Transmission

The first value r_1 of one's reputation is about the correctness of data it forwarded. Its computation in our mechanism depends on a design called *sliding window*, which contains the w latest verification records.

For each neighbor, the node maintains a window of size w. As $|P| = k$, node $\mathcal{A}$ needs to keep k windows simultaneously. When receiving a valid transaction from a neighbor i, it writes $+1$ in the latest box of this neighbor i's window. On the contrary, if the transaction is invalid, -1 is recorded. Then the first value of i's reputation is exactly the summation of all records in its window. In other words, denoting the j-th record in the window by c_j, then i's r_1 is calculated as

$$r_1 = \sum_{j=1}^{w} c_j \tag{1}$$

where $c_j = 0$ if there is still no record in the j-th box.

A window slides when one of the two things happens: some records are out of date, or there is no space for the newest record. Specifically, there is a parameter θ representing the term of validity, during which the records are active. Each time the node computes a neighbor's r_1, it removes all expired records in the window. Also, if the window has been full but a new record is coming, the oldest record would be forgotten by sliding the window and the new record can be written down.

As mentioned in Section III-A, a node may switch between honest and unreliable behaviors. Then we use *error frequency* to measure how unreliable a node is. The error frequency of a node is its frequency to send invalid data. Assume a neighbor's error frequency is $q \in [0, 1]$. Then according to the law of large numbers, as time goes on, its window will be full and r_1 will be stable at a certain value. To be specific, the window will contain both positive and negative records, corresponding to the valid and invalid transactions, respectively. Let W_p be the set of those positive records, and similarly, let W_n be the set of negative ones. Then the expected size of W_n is qw. And r_1 in expectation can be calculated as

$$\begin{aligned} E(r_1) &= E(\sum_{j=1}^{w} c_j) = E(\sum_{j \in W_p} c_j + \sum_{j \in W_n} c_j) \\ &= (1 - 2q)w. \end{aligned} \tag{2}$$

In particular, if $q = 0$, r_1 is w. That is, if a node carefully verifies every received transaction before forwarding, it will earn the maximum value owing to the hard work. Instead, if someone only sends invalid data (*i.e.*, $q = 1$), records in its window are all negative and as a result it will obtain the minimum value $-w$.

C. Second Value of Reputation: Quantity of Interactions

The second value r_2 of reputation focuses on the quantity of data interaction. Recall that during the data transmission, relay nodes would check whether they have processed received transactions before. In this paper, we call a transaction *useful* if it is valid and the node receives it for the first time.

Let f_i the number of useful data received from neighbor i. Also, neighbor i may come to ask for certain data. We denote the number of data that node $\mathcal{A}$ provides to i by t_i. Then the second value of the neighbor i is designed as

$$r_2 = f_i - t_i. \tag{3}$$

In the node's reputation list, a neighbor i's data supply could bring about an increase of its r_2. Instead, the data requests would consume its r_2. Different from that the computation of r_1 only refers to the latest records in the window, in the second dimension, r_2 is a cumulative value based on all the historical data interactions. From the perspective of node $\mathcal{A}$, if the neighbor i's data contribution f_i is less than the data gain t_i, its r_2 will be negative, which could be understood as a data loan. Then it is conceivable that the inactive neighbors, who constantly request data while providing little, shall have a relatively small r_2. In other words, node $\mathcal{A}$ could locate those inactive neighbors by the second value of reputation.

V. Reputation-based Protocol

In this section, we propose a reputation-based relay protocol, *RepuLay*, for data transmissions in blockchain networks. Based on our reputation mechanism, *RepuLay* adopts probabilistic strategies to verify and forward transactions. Without loss of generality, we introduce the protocol based on the following situation:

Node $\mathcal{A}$ has a set of neighbors $P = \{1, \cdots, k\}$. It maintains a local reputation list $[R] = \{R_1, \cdots, R_k\}$, where $R_i = \langle r_{i1}, r_{i2} \rangle$ and $1 \leq i \leq k$. At a certain time, node $\mathcal{A}$ receives a transaction tx from the neighbor i. Like existing designs, the first thing for $\mathcal{A}$ is to check locally, to avoid processing the same transaction more than once. If node $\mathcal{A}$ successfully finds tx in the local store, it will stop immediately. Otherwise, node $\mathcal{A}$ will save it and go to the following steps.

A. Probabilistic Verification

In the verification process, node $\mathcal{A}$ determines whether to verify this fresh tx. Instead of verifying all received data without distinction, the node verifies tx with a certain probability p_v, which is deduced from the first value of sender i's reputation (*i.e.*, r_{i1}). The probabilistic verification allows some data to be relayed directly without waiting for the verification result, thus reducing the transmission delay. However, it also poses a risk that the relay nodes may omit some invalid transactions. So the crux is to better balance speed with the quality of transmission. To achieve this, we design the probability function as follows.

1) The sender's sliding window is not full: For this case, the verification probability $p_v = 1$. That is, node $\mathcal{A}$ cannot judge the sender's behavior by too few historical records. So it chooses to verify tx this time.

2) The sender's sliding window has been full: For this case, the verification probability is designed as

$$p_v = \begin{cases} \alpha, & q \leq \frac{\eta}{1-\alpha}; \\ 1 - \frac{\eta}{q}, & else. \end{cases} \tag{4}$$

where $\alpha \in (0, 1)$ and $\eta \in [0, 1]$. Here, α is the minimum check probability. Even if the sender looks like an honest node, node $\mathcal{A}$ would still verify the received tx with a probability no less than α. The other parameter, η, is the maximum of nodes' error frequency that the system could tolerate. These two values will be pre-set by the system and can also be adjusted by nodes themselves. Together, the two parameters help promise the percentage of valid data relayed by $\mathcal{A}$. In addition, q is the sender i's error frequency. According to (2) and the law of large numbers, it can be estimated by

$$q = \frac{1}{2} - \frac{r_{i1}}{2w} \tag{5}$$

where w is the window size.

Based on the above designs, we propose Algorithm 1. As the pseudocode shows, taking the first value of sender's reputation and other parameters as inputs, the algorithm returns the suggested verification probability p_v.

Algorithm 1 Probabilistic Verification

Input:
First value of sender's reputation r_1;
Number of records in sender's sliding window $\#$;
Size of the sliding window w;
Upper bound of error frequency η;
Minimum check probability α.
Output:
A verification probability p_v.

if $\# < w$ **then**
  $p_v \leftarrow 1$;
else
  $q \leftarrow \frac{1}{2} - \frac{r_1}{2w}$;
  if $q \leq \frac{\eta}{1-\alpha}$ **then**
    $p_v \leftarrow \alpha$;
  else
    $p_v \leftarrow 1 - \frac{\eta}{q}$;
  end if
end if
return p_v

Algorithm 2 Selective Forwarding

Input:
Set of neighbors $P = \{1, \cdots, k\}$;
Local reputation list $[R] = \{R_1, \cdots, R_k\}$ where $R_i = \langle r_{i1}, r_{i2} \rangle$;
Number of receivers λ;
A trade-off parameter β.
Output:
The set of receivers.

$\Lambda \leftarrow \emptyset$;
$C \leftarrow P - \{i\}$; $\{i$ is the sender of data$\}$
while $C \neq \emptyset$ **and** $|\Lambda| < \lambda$ **do**
  for all $j \in C$ **do**
    $p_j = (1-\beta) \cdot \frac{e^{r_{j2}}}{\sum_{l \in C} e^{r_{l2}}} + \beta \cdot \frac{1}{|C|}$;
  end for
  Pick up a node j with probability p_j;
  $\Lambda \leftarrow \Lambda \bigcup \{j\}$;
  $C \leftarrow C - \{j\}$;
end while
return Λ

With the probability p_v, node $\mathcal{A}$ will verify tx. Once finding that tx is invalid, it will intercept this transaction immediately and stop forwarding. Otherwise, node $\mathcal{A}$ broadcasts it to some neighbors. It is worth mentioning that the latter involves two cases: tx passes the verification, or $\mathcal{A}$ does not verify it.

B. Selective Forwarding

In the forwarding process, tx is further broadcast to nodes in the network. Assume node $\mathcal{A}$ plans to send tx to λ neighbors, where λ is a customized parameter[1]. Which λ neighbors to choose is the key to this step.

Our protocol does not distinguish the outbound neighbors from inbound connections. Instead, the λ receivers are selected from all neighbors, according to the second value of their reputation (*i.e.*, r_2). As previously discussed, r_2 is the net contribution regarding useful data. Based on the principle of reciprocity, the neighbor with higher r_2 deserves a higher priority to receive this new tx. In addition, a larger r_2 means that the corresponding neighbor has enough computation resources to forward data and is willing to do so. Thus, the larger a neighbor's r_2 is, the more chances it should be chosen as the receiver. Specifically, node $\mathcal{A}$ selects receivers as follows:

(i) Initialize the candidate list $C := P \backslash \{i\}$, where P is the set of neighbors and i is the sender of tx;
(ii) For each $j \in C$, calculate its probability to be selected

$$p_j = (1-\beta) \cdot \frac{e^{r_{j2}}}{\sum_{l \in C} e^{r_{l2}}} + \beta \cdot \frac{1}{|C|} \tag{6}$$

where β is a constant parameter belonging to [0,1];
(iii) Pick node j up with probability p_j and update the candidate list $C = C \backslash \{j\}$;
(iv) Repeat (ii) and (iii) until λ receivers are selected.

[1]Like other designs, we recommend to set λ as a constant eight.

As (6) shows, the probability function contains two parts. The first part ensures that neighbors with higher r_2 are more likely to be chosen. The second part is for fairness, giving some opportunities to the neighbors with lower r_2, especially the following two kinds of nodes: the new neighbors whose r_2 are 0, and those indeliberately inactive neighbors that are limited by objective conditions. In addition, we introduce a parameter β to balance these two parts.

The above selective process has been presented in Algorithm 2. According to the local reputation list and some self-defined parameters, the algorithm probabilistically decides a set of nodes as receivers. Afterwards, node $\mathcal{A}$ sends them the hash value of tx.

C. Reputation Updating

This step updates neighbors' reputation according to the above verification process and forwarding process.

1) Updating the sender's r_1 and r_2: If node $\mathcal{A}$ actually verifies tx, it will be clear about its validity. Then node $\mathcal{A}$ could update the first value of the sender's reputation. More specifically, node $\mathcal{A}$ records $+1$ and -1 in the latest box of the sender's window, corresponding to valid and invalid respectively. Then i's r_1 can be updated by (1). And if node $\mathcal{A}$ does not verify tx, updating the sender's r_1 is not required. With respect to sender's r_2, two things lead to its updating: node $\mathcal{A}$ does not verify tx, or after verification it confirms that the tx is valid. Both cases make $\mathcal{A}$ increase sender's r_2 by 1.

2) Updating the receivers' r_2: After responding to the GETDATA message during the forwarding process, node $\mathcal{A}$ could update the second value of these receivers' reputation. In other words, this updating is only for those neighbors to which $\mathcal{A}$ sends the complete tx. Concretely, node $\mathcal{A}$ decreases their r_2 by 1.

Fig. 3 summarizes the flow of data transmission in *RepuLay*.

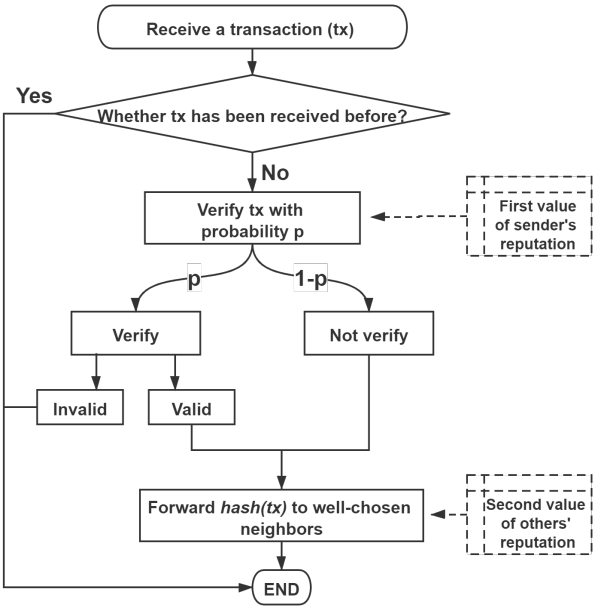

Fig. 3. Flow of data transmission in *RepuLay*.

VI. Analysis and Discussion

A. Analysis of Verification Probability Function

In order to analyze the effectiveness of our designed verification probability, we first give the following theorem and its proof. Here, the error probability of a node is its probability to send invalid transactions.

Theorem 1: Assume node $\mathcal{A}$ receives a transaction from a neighbor whose error probability is q. By verifying the transaction with a probability no less than $1 - \frac{\eta}{q}$, $\mathcal{A}$'s error probability will not exceed η.

Proof: Consider the following scenario. Node $\mathcal{A}$ receives M transactions from a certain neighbor over a period of time. With probability q, the transaction may be invalid. So in expectation, there are qM invalid transactions among them. When receiving a transaction from this neighbor, node $\mathcal{A}$ decides to verify it with probability p. Then the expected number of verified transactions is pM.

For node $\mathcal{A}$, the relayed transactions that it has verified must be valid. Its error frequency only comes from the unverified transactions. For the rest $(1-p)M$ unchecked transactions, some of them may be invalid. Let the random variable X describe the number of invalid ones among unchecked transactions. Then the probability of missing exactly k invalid transactions can be calculated by the formula

$$P(X=k) = \frac{\binom{qM}{k}\binom{(1-q)M}{(1-p)M-k}}{\binom{M}{(1-p)M}} \tag{7}$$

where $k \in [0, m]$ and $m = \min\{qM, (1-p)M\}$. It shows that the random variable X follows the hypergeometric distribution. Furthermore, we could obtain its mean as

$$E(X) = \sum_{k=0}^{m} k \cdot P(X=k) = M(1-p)q. \tag{8}$$

If node $\mathcal{A}$'s verification probability satisfies that $p \geq 1 - \frac{\eta}{q}$, we have $\frac{E(X)}{M} = (1-p)q \leq \eta$. That is, its error probability will not exceed η. ■

Based on Theorem 1, the minimum verification probability should be set as $\max\{0, 1 - \frac{\eta}{q}\}$. In our design, by introducing the parameter α, we increase the minimum value to $\max\{\alpha, 1 - \frac{\eta}{q}\}$, which further ensures the quality of transmission. Moreover, when computing q by (5), to make it reflect the actual behavior as far as possible, we require that the sender's window must be full. All these can lessen the chances of making mistakes.

B. Properties of Protocol

1) Independence: Each node could use the protocol to score its neighbors and make decisions by itself, only depending on the local information it has. It is of great importance that the whole process can be completed independently. This feature exactly fits the decentralization of blockchain systems.

2) Robustness: All reputation-based systems must consider a terrible manner called *dynamic personality attack*. That is, the attacker behaves properly for a period of time and builds up a high reputation to be highly trusted. However, at a specific time, the attacker begins to milk the accumulated reputation to deceive. In the context of this paper, readers may worry about the following case: no one verifies an invalid transaction, making it spread across the network. We reassure readers by claiming that the probability of this worst case is $(1-\alpha)^n$ under our design[2], which is negligible. Even if it happens, the system's safety will not be affected, as miners will carefully check the selected transactions before mining. Again, we emphasize that the network layer is only responsible for transmitting data while the safety should be assured by the consensus layer.

3) Flexibility: The flexibility of the protocol is embodied by two aspects. First, the involved parameters can be personalized by different nodes according to their own needs. For example, a node may prefer to check carefully rather than miss invalid data by accident. Then it could set a relatively small η and a relatively large α. If a node places a higher value on neighbors' recent behavior, a smaller window size w and shorter validity period θ will do some help. Second, the protocol is designed for the network layer and can be straightforwardly piggybacked on existing blockchain systems, without any impact on other layers (*e.g.*, the consensus layer).

VII. Performance Evaluation

In this section, we conduct a series of simulations to evaluate our design. First, we describe the simulation settings. Then we show the simulation results and give the corresponding analysis.

[2] n is the number of nodes in the network.

TABLE I
SIMULATION PARAMETERS

Parameter	Description	Default Value
$\mathcal{V}$	Number of nodes	1000
$\mathcal{E}$	Number of connections	16000
w	Size of the sliding window	50
η	Upper bound of error frequency	0.1
α	Minimum check probability	0.25
λ	Number of receivers	8
β	Fairness coefficient	0.1

A. Simulation Setup

This section describes the general simulation setup, including the network structure, nodes' behavior pattern and some evaluation metrics.

1) Implementation: In our simulations, a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is randomly constructed. The number of nodes $|\mathcal{V}|$ is set to a fixed value 1000, and the number of connections $|\mathcal{E}| = 16000$. In the network, a group of transactions is to be spread, and initially, they are randomly assigned to all nodes. For convenience, we discretize the time into time slots with a fixed length. In each time slot, different types of nodes behave as follows. The honest nodes verify and forward transactions as requested. Conversely, the inactive nodes refuse to relay transactions. The unreliable nodes forward transactions directly without any verification and deliberately generate invalid transactions at a stable frequency. The simulation proceeds until there is no data transmission in the network. The proportions of three kinds of nodes vary in different simulations, which will be detailed later.

Table I summarizes the main parameters involved in the protocol. If not specified, parameters are set to the default value.

2) Evaluation Metrics: In order to observe the effect of our protocol, we evaluate it from the following aspects.

- **Synchronization time of valid transactions:** We use the transactions' synchronization time among nodes to measure *RepuLay*'s transmission speed, which is the focus of this paper.
- **Average verification rate:** The verification rate is the percentage of verified transactions among all received. Considering that the time spent in the verification process can not be neglected, a lower verification rate will bring faster delivery.
- **Average accuracy:** The accuracy is the number of valid transactions over all relayed transactions. The higher accuracy means a better quality of transmission in the network.
- **Number of transactions sent per node:** This metric is used to measure bandwidth required when running *RepuLay*.

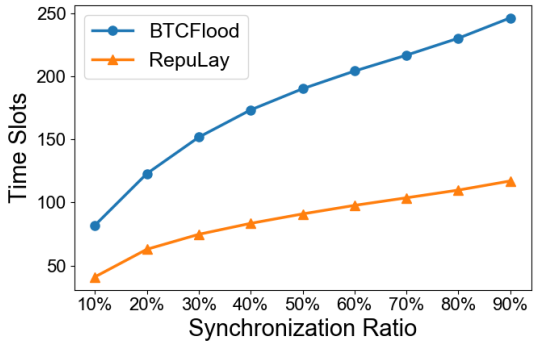

Fig. 4. Performance comparison between *RepuLay* and flooding protocol. The metric is the average time slots for transactions to reach a certain proportion of honest nodes in the network.

B. Transmission Speed

In this simulation, we compare *RepuLay* with the widely adopted flooding protocol (*BTCFlood* for short) regarding transmission speed. *BTCFlood* verifies all transactions and forwards valid ones to all outbound neighbors[3]. The proportions of honest, unreliable, and inactive nodes are 40%, 30%, and 30%, respectively.

The metric to measure the speed is the used time slots for valid transactions to reach nodes in the network. Figure. 4 visualizes the results under different strategies. The X-axis is the proportion of nodes that have received certain transactions. The Y-axis corresponds to the average synchronization time. Take the right-most marker on two lines as an example. In order to spread to 90% of nodes, *BTCFlood* spends 247 time slots in average, while *RepuLay* only requires 117 time slots. That is, *RepuLay* brings a significant improvement and is 2.x faster than the flooding protocol. In other words, the simulation result supports that our protocol can effectively accelerate the spread of transactions.

C. Effectiveness against Unreliable Nodes

The objective of this simulation is to evaluate the effectiveness against unreliable nodes. For this, 70% of nodes are assumed honest, who adopt *RepuLay* to verify the received transactions with a certain probability. The remaining 30 percent is all unreliable.

Based on neighbors' r_1, honest nodes adopt the probabilistic verification. It enables the relay nodes to forward some transactions without verification, but also brings a risk that they may miss invalid transactions. The solid and dashed lines in Fig. 6 respectively represent the average accuracy and verification rate of all honest nodes in each time slot. In the early stage, as the sliding windows are not full, honest nodes need to verify each received transaction. Thus both values are 1. The probabilistic verification does work after the sender's window is full. Compared with the sharp drop of the verification rate, the accuracy only has a slight decline.

[3]The number of outbound connections is also set to 8, which is the same with the number of receivers in *RepuLay*.

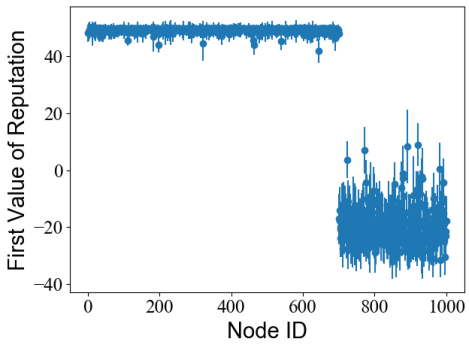

Fig. 5. The first value of nodes' reputation. Nodes numbered 1-700 are honest and others are unreliable.

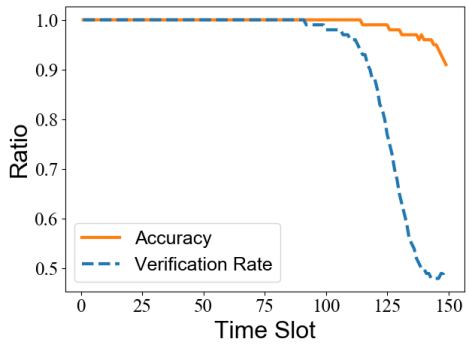

Fig. 6. The average accuracy and verification rate of honest nodes.

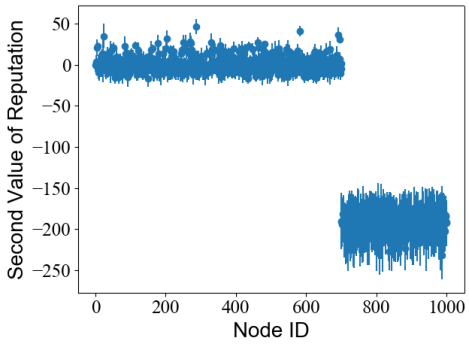

Fig. 7. The second value of nodes' reputation. Nodes numbered 1-700 are honest and others are inactive.

What stands out in Fig. 6 is that only verifying about 50% transactions can promise an accuracy above 90%. This result provides compelling evidence that *RepuLay* balances well between the speed of data transmission and the quality of relayed transactions.

D. Effectiveness against Inactive Nodes

This set of simulation evaluates *RepuLay*'s effectiveness against inactive nodes in the network. There are 70% honest nodes (numbered 1-700) and 30% inactive nodes (numbered 701-1000).

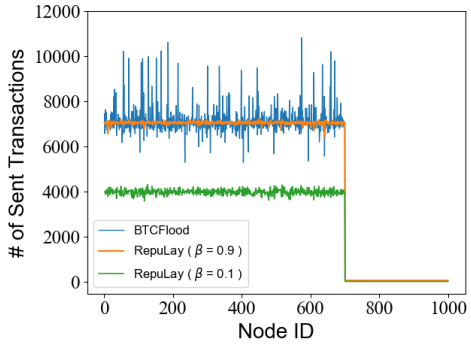

Fig. 8. Number of sent transactions.

After the simulation ends, we collect the second value of all nodes' reputation and visualize them in Fig. 7. The drawing method is the same as what we do in Fig. 5. As shown, two types of nodes (*i.e.*, honest nodes and inactive nodes) are clearly separated from each other with the help of r_2.

Further, we redo the simulation by making honest nodes to adopt different strategies, and count the total number of transactions sent by each node during the simulation. Here, three strategies are compared: *BTCFlood*, *RepuLay* with $\beta = 0.9$, and *RepuLay* with $\beta = 0.1$. Recall that β is a parameter appearing in the probability function (6) for choosing receivers. When setting $\beta = 0.9$, *RepuLay* puts more emphasis on the fairness, and selects receivers almost randomly. By contrast, *RepuLay* with $\beta = 0.1$ attaches more importance to the role of reputation.

As shown in Fig. 8, three lines correspond to three strategies. These three achieved the same goals, *i.e.*, broadcasting the same set of transactions to the whole network. Each line can be roughly split into a left and a right section, with the *Node ID = 701* as a demarcation point. First, let us look at the right part. As the inactive nodes refuse to relay transactions (except for the ones they generate), three lines of nodes numbered 701-1000 almost overlap and are near to zero. Actually, we are more concerned with the honest nodes and want to know how many transactions they need to send under different strategies. Considering the left part, *BTCFlood* and *RepuLay* with $\beta = 0.9$ have nearly the same mean (7134 and 7039 respectively), while the former fluctuates widely. Intuitively, for different nodes, the fluctuation in *BTCFlood* is related to the proportion of inactive neighbors. Surprisingly, *RepuLay* with $\beta = 0.1$ is stable and has a much smaller value. The gap is caused by sending transactions to inactive neighbors and can be considered as the bandwidth that honest nodes waste. A small number of sent transactions implies that *RepuLay* with $\beta = 0.1$ brings more efficient transmissions.

Overall, these results indicate that *RepuLay* not only accelerates the spread of transactions, but also helps nodes save their bandwidth.

VIII. Conclusion

In this paper, we propose a relay protocol called *RepuLay*, aiming to speed up the data transmission in blockchain networks without compromising the quality of service. First, we design a reputation mechanism. Each node in the system maintains a reputation list of its neighbors, according to their historical interactions. The reputation helps nodes to distinguish different types of neighbors, whose behaviors are defined from an economic point of view. Based on this, *RepuLay* takes a probabilistic approach to verify and forward transactions. Theoretical analysis and further simulations indicate that our protocol could accelerate the data propagation and save the bandwidths.

In terms of future work, we would consider the connection optimization and further improve *RepuLay*. What's more, we will give more security analysis and consider how the nodes behave at equilibrium from the perspective of game theory.

Acknowledgment

This work was supported by National Natural Science Foundation of China (No. 61761146005, 11871366). Yukun Cheng and Xiaotie Deng are corresponding authors.

References

[1] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim, "A survey on consensus mechanisms and mining strategy management in blockchain networks," *IEEE Access*, vol. 7, pp. 22 328–22 370, 2019.

[2] X. Xu, C. Pautasso, L. Zhu, V. Gramoli, A. Ponomarev, A. B. Tran, and S. Chen, "The blockchain as a software connector," in *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*. IEEE, 2016, pp. 182–191.

[3] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[4] M. Vasek, M. Thornton, and T. Moore, "Empirical analysis of denial-of-service attacks in the bitcoin ecosystem," in *International conference on financial cryptography and data security*. Springer, 2014, pp. 57–71.

[5] M. Dotan, Y.-A. Pignolet, S. Schmid, S. Tochner, and A. Zohar, "Sok: cryptocurrency networking context, state-of-the-art, challenges," in *Proceedings of the 15th International Conference on Availability, Reliability and Security*, 2020, pp. 1–13.

[6] A. Gervais, H. Ritzdorf, G. O. Karame, and S. Capkun, "Tampering with the delivery of blocks and transactions in bitcoin," in *CCS*. ACM, 2015, pp. 692–705.

[7] A. Miller, J. Litton, A. Pachulski, N. Gupta, D. Levin, N. Spring, and B. Bhattacharjee, "Discovering bitcoin's public topology and influential nodes," *et al*, 2015.

[8] L. Xiong and L. Liu, "A reputation-based trust model for peer-to-peer e-commerce communities," in *EEE International Conference on E-Commerce, 2003. CEC 2003.* IEEE, 2003, pp. 275–284.

[9] R. Zhou and K. Hwang, "Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing," *IEEE Transactions on parallel and distributed systems*, vol. 18, no. 4, pp. 460–473, 2007.

[10] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *Proceedings of the 12th international conference on World Wide Web*. ACM, 2003, pp. 640–651.

[11] P. Michiardi and R. Molva, "Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks," in *Advanced communications and multimedia security*. Springer, 2002, pp. 107–121.

[12] B. T. Adler and L. De Alfaro, "A content-driven reputation system for the wikipedia," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 261–270.

[13] E. Damiani, D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante, "A reputation-based approach for choosing reliable resources in peer-to-peer networks," in *Proceedings of the 9th ACM conference on Computer and communications security*. ACM, 2002, pp. 207–216.

[14] J. Zhang, Y. Cheng, X. Deng, B. Wang, J. Xie, Y. Yang, and M. Zhang, "Preventing spread of spam transactions in blockchain by reputation," in *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*. IEEE, 2020, pp. 1–6.

[15] M. Zhang, J. Li, Z. Chen, H. Chen, and X. Deng, "Cycledger: A scalable and secure parallel protocol for distributed ledger via sharding," in *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2020, pp. 358–367.

[16] N. Szabo, "Smart contracts: Building blocks for digital markets," http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html, Last accessed 13 May 2021.

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

[18] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "Sok: Research perspectives and challenges for bitcoin and cryptocurrencies," in *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2015, pp. 104–121.

[19] A. Reyna, C. Martín, J. Chen, E. Soler, and M. Díaz, "On blockchain and its integration with iot. challenges and opportunities," *Future Generation Computer Systems*, vol. 88, pp. 173–190, 2018.

[20] S. Han, Z. Xu, and L. Chen, "Jupiter: a blockchain platform for mobile devices," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 2018, pp. 1649–1652.

[21] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. E. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, D. Song, and R. Wattenhofer, "On scaling decentralized blockchains - (A position paper)," in *Financial Cryptography Workshops*, ser. Lecture Notes in Computer Science, vol. 9604. Springer, 2016, pp. 106–125.

[22] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *P2P*. IEEE, 2013, pp. 1–10.

[23] M. Corallo, "Compact block relay," https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki, Last accessed 13 May 2021.

[24] A. P. Ozisik, G. Andresen, G. Bissias, A. Houmansadr, and B. N. Levine, "Graphene: A new protocol for block propagation using set reconciliation," in *DPM/CBT@ESORICS*, ser. Lecture Notes in Computer Science, vol. 10436. Springer, 2017, pp. 420–428.

[25] FIBRE, http://bitcoinfibre.org/, Last accessed 13 May 2021.

[26] M. F. Sallal, G. Owenson, and M. Adda, "Proximity awareness approach to enhance propagation delay on the bitcoin peer-to-peer network," in *ICDCS*. IEEE Computer Society, 2017, pp. 2411–2416.

[27] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on bitcoin's peer-to-peer network," in *USENIX Security Symposium*. USENIX Association, 2015, pp. 129–144.

[28] H. Vu and H. Tewari, "An efficient peer-to-peer bitcoin protocol with probabilistic flooding," in *International Conference for Emerging Technologies in Computing*. Springer, 2019, pp. 29–45.

[29] G. Naumenko, G. Maxwell, P. Wuille, A. Fedorova, and I. Beschastnikh, "Erlay: Efficient transaction relay for bitcoin," in *ACM Conference on Computer and Communications Security*. ACM, 2019, pp. 817–831.