Taming Propagation Delay and Fork Rate in Bitcoin Mining Network

Suhan Jiang and Jie Wu
Department of Computer and Information Sciences, Temple University
{suhan.jiang, jiewu}@temple.edu

2021 IEEE International Conference on Blockchain (Blockchain) | 978-1-6654-1760-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/BLOCKCHAIN53845.2021.00050

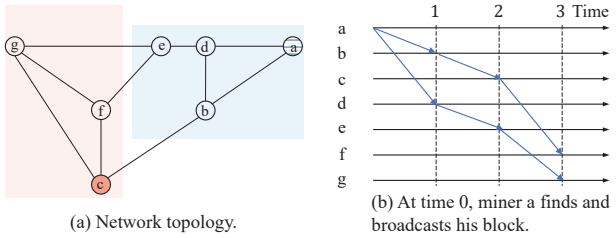
***Abstract*—Bitcoin builds upon an unstructured peer-to-peer overlay network to disseminate transactions and blocks. Broadcast in such a network is slow and brings inconsistencies, *i.e.*, peers have different views of the system state. Due to the delayed block propagation and the competition of mining, forking, *i.e.*, the blockchain temporarily diverges into two or more branches, occurs frequently, which wastes computation power and causes security issues. This paper proposes an autonomous and distributed topology optimization mechanism to reduce block propagation delay and hence reduce the occurrence of blockchain forks. In the proposed mechanism, a node can autonomously update his neighbor set using the information provided by his current neighbors, since each neighbor will recommend a peer from his own neighbor set, *i.e.*, a neighbor's neighbor, to this node. Each recommendation is based on a peer's propagation ability, which is characterized as a criteria function obtained through a combination of empirical analysis and machine learning. We further propose some metrics to evaluate a Bitcoin network topology. Experiment results reflect the effectiveness of the proposed mechanism and also indicate the correlation between block propagation time and fork rate.**

***Index Terms*—Blockchain, criteria function, fork, neighbor selection, P2P overlay, propagation delay.**

I. Introduction

The Bitcoin mining network is designed as a peer-to-peer (P2P) overlay [1, 2], where nodes, named as miners, are randomly connected. Blocks are transmitted over this network using a multi-hop broadcast scheme. That is, a block creator broadcasts his newly mined block to all of his neighbors first. Peers receiving such a new (unseen) block will relay it in the same manner until all nodes receive this block. Given a topology in Fig. 1(a), Fig. 1(b) shows the process how node a broadcasts his block (which is found at time 0) in the network. Suppose that the transmission delay is one time step for each node, then a's block is known by all nodes at time 3. Obviously, this distributed model brings inconsistencies to the Bitcoin system. Since each propagation hop induces a delay, a block reaches different peers at different times. Thus, peers may have different local views of the blockchain during the block propagation process. As a result, some miners are mining on top of the newest block while others are still extending a stale blockchain. It is unfair since uninformed miners may waste their mining power as well as electricity.

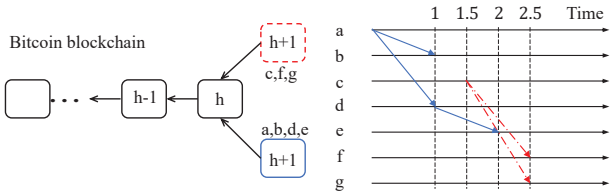
Beyond fairness, blockchain forking is another severe problem caused by the block propagation delay [3]. A fork occurs if a miner's block is still in propagation while another miner, who hasn't yet known this block, creates and starts to broadcast his own block of the same height. Fig. 2(b) shows such an example, where c' block is propagated since time 1.5. As two valid blocks are spreading in the network, each peer mines on top of the one he receives earlier. As is shown in Fig. 2(a), the blockchain consequently diverges into two branches, either of which is extended by part of mining power. By design, forks resolve as soon as one branch becomes longer, usually the one extended by more mining power, at which point the shorter branch is abandoned. In Fig. 2, a's branch can be accepted as part of the main chain if anothor miner, say e, successfully extends a's branch. A fork can sustain a long time, if mining power distributed to the two branches are close or equal. Forks lasting four blocks have been reported in the Bitcoin blockchain [4].



(a) Network topology.

(b) At time 0, miner a finds and broadcasts his block.

Fig. 1: Block propagation in the Bitcoin network.



(a) Blockchain forks and miners are divided into {a,b,d,e} and {c,f,g}.

(b) At time 1.5, miner b also finds and broadcasts his block.

Fig. 2: A fork occurs at height h.

Previous studies have shown that propagation performance of a P2P overlay can be improved by optimizing its underlying topology. We believe that the Bitcoin network can also take advantage of this method to reduce its block propagation delay as well as alleviate forking. Considering it from miner a's point of view, the network topology is optimized by the removal of a-b link and the addition of a new connection between a and c. Then, a's block propagation time to the entire network is shortened from 3 to 2. Meanwhile, it also

This research was supported in part by NSF grants CNS 1824440, CNS 1828363, CNS 1757533, CNS 1629746, CNS 1651947, and CNS 1564128.

978-1-6654-1760-0/21/$31.00 ©2021 IEEE
DOI 10.1109/Blockchain53845.2021.00050

avoids the occurrence of the previous blockchain forking. As c has already known a's block at time 1, he starts mining a new block on top of a's.

The previous example reflects the problems caused by block propagation delays and also motivates us to speed up Bitcoin block propagation by optimizing its underlying topology. To keep the decentralization nature of the Bitcoin network, all topology changes should happen in a completely autonomous and distributed way. That is, each miner spontaneously reconfigures the topology to reduce his block propagation delay by using local information. Based on this objective, we propose an autonomous topology optimization mechanism to speed up Bitcoin block propagation. The core of this mechanism is a distributed algorithm called Recommendation-based Neighbor Selection (RNS), which allows a miner to update his neighbor set in the following steps: each current neighbor will recommend a peer from his own neighbor set, *i.e.*, a neighbor's neighbor, to this miner, and then he himself selects neighbors from both the current neighbor set and the recommended peer set. Recommendations are made according to a peer's propagation ability (measured by a criteria function). Through empirical analysis and machine learning, we propose a criteria function only using a peer's local features, *e.g.* a peer's degree and its local clustering coefficient.

Besides block broadcast time, we also propose some other metrics to quantify performances of a Bitcoin/Bitcoin-like network topology. Then, we compare the performance of RNS optimized overlays with overlays obtained by other existing algorithms, in terms of our proposed metrics. The evaluation results not only demonstrate the effectiveness of our proposed mechanism, but also reflect an interesting corelation between block propagation time and fork rate. The major contributions of this paper are as follows:

- Through empirical analysis and machine learning, we fit a suitable criteria function to quickly quantify a node's propagation ability using its local information.
- Based on our criteria function, we propose a distributed recommendation-based neighbor selection algorithm, aiming to optimize the current Bitcoin network topology.
- We propose several metrics to effectively evaluate performances of a Bitcoin/Bitcoin-like network topology.
- We compare the proposed mechanism with several existing works by evaluating their corresponding topologies.

II. Current Mechanism and Motivation

1) Bitcoin Neighbor Selection Mechanism: Nodes in the Bitcoin network are identified by their IP addresses. Each node has a list of IP addresses of potential peers. The list is bootstrapped through a DNS server, and additional addresses are exchanged between peers. From his list, a node randomly selects 8 reachable peers, with which it forms long-lived outgoing connections. A node can be recognized as reachable or non-reachable, depending on whether or not to accept an incoming connection. Outgoing connections and incoming connections are functionally-equal. The only difference is that, a node's outgoing connections are initiated by himself, while his incoming connections are unsolicited. Reachable nodes can additionally accept up to 117 unsolicited connections from other nodes. This paper only considers a Bitcoin network composed of all reachable nodes. Thus, the total number of connections a node can have is 125 by default.

We now give a brief introduction on how a node decides his 8 outgoing neighbors. New outgoing connections are selected if a node boosts or if an outgoing connection is dropped by the network. A node with $\omega \in [0, 7]$ outgoing connections selects the $(\omega+1)$-th connection as follows: first, he decides whether to select from a tried table (nodes that he has connected to before) or a new table (nodes that are provided by the current neighbors but never contacted). The default algorithm makes tried addresses more likely to be selected when there are few outgoing connections or the tried table is large. Second, he selects a random address from the chosen table, with a bias towards addresses with fresher timestamps. After that, the node attempts to connect to the selected address. If the connection fails, he will repeat the above two steps. As a node also receives incoming connecting requests from other nodes, he accepts all those unsolicited connections until reaching the upper bound. A Bitcoin node never deliberately drops a connection, except when a blacklisting condition is met.

In the Bitcoin network, each node always wants to receive the newest block information in the system as soon as possible. Meanwhile, if he becomes a block creator, he also wants that his block could be broadcast immediately in order to avoid blockchain forking or at least take advantages in a forking competition. Block reception and dissemination heavily rely on his neighbors, who, to some extent, determine the way he communicates with the rest of the Bitcoin network. Existing research [5] on unstructured P2P file systems has proven the importance of a node's neighbor set for query dissemination and target data reception. It also has been observed in [6] that, blocks first announced by some nodes propagate consistently faster (or slower) than others. From an individual node's point of view, a good neighbor set can fasten the block propagation speed as well as shorten the block receiving time.

2) Motivation: As a typical P2P cryptocurrency network, the main purpose of Bitcoin network is to propagate information as fast as possible, which is similar to the purpose of a P2P content delivery network, and achieve consensus on a publicly shared ledger, which is unique to cryptocurrency network itself. Usually, information delivery in a content delivery network is within a small part of the network which may be traceable [7] while in the Bitcoin network, blocks are required to be propagated among all nodes. These two big differences make traditional P2P network optimization methods not applicable in optimizing Bitcoin topology, and also motivate us to focus on Bitcoin topology optimization.

III. Recommendation-based Neighbor Selection

Previous studies on P2P network optimization prove that using the proximity neighbor selection technique can improve the propagation performance in P2P networks. The existing research also shows that, some influential nodes with strong

TABLE I: Summary of Notations.

Symbol	Description
i	nodes in the blockchain network
N_i	node i's outgoing neighbor set
$\|N_i\|$	number of node i's outgoing neighbors
ω	index used to distinguish outgoing neighbors, $\omega \in [0, 7]$
n	number of nearby outgoing neighbors
D_i	number of all node i's neighbors
C_i	node i's local clustering coefficient
m_i	node i's mining power
S_i	node i's propagation ability,

propagation ability can accelerate information propagation in large-scale complex networks. Thus in the Bitcoin network, when selecting his neighbors, a node should take two factors into consideration. One is a peer's proximity and the other is a peer's propagation ability. It is non-trivial to measure these two factors due to the specificity of the Bitcoin network. A node can determine each known peer's suitability to be a neighbor if applying some suitable measurements. Traditionally, the proximity of two nodes in networks is captured by their geographical distance. In this paper, we apply the round-trip-time, which can be easily obtained through a ping message, to describe the proximity between two nodes. Lots of methods also have been proposed to rank a node's propagation ability, such as betweenness centrality, eigenvalue centrality, or k-shell. Most of them require a global view of the network topology, which is unrealistic for the Bitcoin network. In this paper, we formulate a criteria function to quantify a peer's propagation ability using local features.

Currently, a node obtains network information from DNS servers and his connected neighbors. Those information is provided in the form of a long list of potential peers' IP addresses. This large-volume but not informative list is useless for a node to efficiently select suitable neighbors. If a node gets more useful information from his neighbors, he definitely can connect to nearby peers of better propagation abilities. As each node could improve his block propagation and receiving time, we believe it definitely leads to a better global topology for the Bitcoin network. Putting all considerations mentioned above together, we propose a recommendation-based neighbor selection mechanism. Our proposed algorithm is a combination of recommendations from the existing neighbors and self-measurement with local information. The key insight of our research is that an efficient neighbor selection maps to the feature selection and the criteria function fitting in the field of machine learning. The following part of this section focuses on describing how a node propose performs neighbor selection using the proposed algorithm. And details on how to measure a peer's propagation ability are explained in section IV. Corresponding notations are shown in Table I.

Proposed Neighbor Selection Algorithm: We want to form a network where nodes are connected in a more efficient way for block propagation, while the network is still relatively random to prevent centralization. Thus, in our mechanism, a node only uses the proposed algorithm to determines his outgoing neighbor set, and always accepts all incoming requests within the limitation of 117 connections. Besides, we want our algorithm not only to be applicable for Bitcoin but also suitable for a new Bitcoin-like network's construction as well as for any existing Bitcoin-like network's reorganization. Thus, our neighbor selection algorithm consists of two parts: one is a Neighbor Finding algorithm, as is shown in Algorithm 1, designed for any node of which the outgoing neighbors are fewer than 8 to fill/refill his neighbor set, and the other is a Neighbor Update algorithm, as is shown in Algorithm 2, used by a node with 8 outgoing neighbors to periodically refine his neighbor set.

Algorithm 1 Outgoing Neighbor Set Filling

Input: node i's current neighbor set N_i, where $|N_i| < 8$
Output: an updated neighbor set N_i, where $|N_i| = 8$

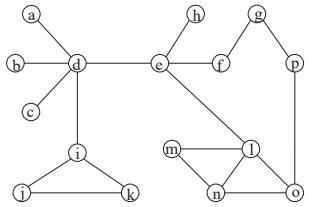
if i's possible neighbor list is empty **then**
  Initiate a potential neighbor list from DNS servers
while i has $\omega \in [0, n-1]$ nearby neighbors **do**
  Pick j of highest S_j from nearby-neighbor list
  if i successfully connects to j **then**
    Add j to N_i
    $\omega = \omega + 1$
while i has $\omega \in [0, 7-n]$ middle neighbors **do**
  Pick j of highest S_j from middle-neighbor list
  if i successfully connects to j **then**
    Add j to N_i
    $\omega = \omega + 1$
Return N_i

Algorithm 2 Outgoing Neighbor Set Update

Input: node i's current neighbor set N_i, where $|N_i| = 8$
Output: an updated neighbor set N_i, where $|N_i| = 8$

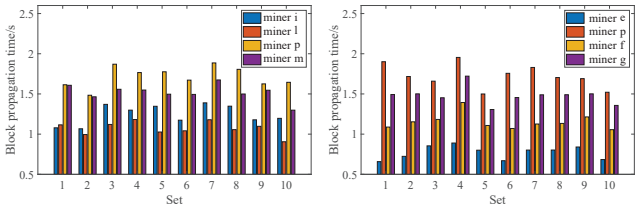
Get 8 peers recommended by current neighbors
Classify 16 peers as nearby or middle peers
for all nearby peers **do**
  Rank peers based on S_j
  Pick top n connectable peers and update N_i
  Record the remaining peers in the nearby-neighbor list
for all middle peers **do**
  Rank peers based on S_j
  Pick top $(8-n)$ connectable peers and update N_i
  Record the remaining peers in the middle-neighbor list
Return N_i

Generally, a node i determines whether a peer j is suitable as a neighbor based on two factors: (1) j's propagation ability, calculated with the criteria function, *i.e.*, S_j, (details on the criteria function will be shown in the next section) and (2) the proximity between i and j, measured by the round-trip-time, denoted by t_{ij}. The proximity plays two conflicting roles here. The suitability of j can be shaded by its long distance from i, even if j is a propagation-well node. The link latency makes the direct connection between i and j replaceable by several



Miner	Power (%)
d, e	12
i	7
l	9
rest	5

Fig. 3: A mining network of 16 nodes, each node expect d, e, i, l occupying 5% of the total mining power.



(a) $D_i = 3$ and $D_l = 4$ (b) $D_e = 4$ and $D_p = 2$

Fig. 4: Degree impact on a miner and its neighbor(s).

intermediate relays starting from one of i's current neighbors. However, a small t_{ij} is not always a preferred choice since it implies i and j may be located in the same 'social hub', and therefore, connecting to j helps little if i wants his block to go beyond this hub and spread the whole network effectively.

Based on the analysis above and also inspired by the prior work indicating that networks with small-world topology can spread information faster than lattice networks [8], we design our algorithm in a proximity-aware method. Node i will classify a peer j's proximity as near, middle, or far. It will choose n (a predefined parameter) nearby and $(8 - n)$ in the middle region based on peer's propagation ability. A far j will not be attempted even if its S_j is big. Node i thereby balances the propagation ability and the proximity when selecting a neighbor. As we find the peer-proximity classification standards and the value of n are influenced by the geographical distribution of all nodes in the network, we determine them appropriately in our experiment.

IV. Feature Selection and Function Fitting

We are aiming to select a small set of features which are easy to calculate for a node using local information while still accurately reflect a peer's propagation ability. All those features contribute to a criteria function, which helps a node determine the suitability of another node if selected as his neighbor. The propagation ability should be measured in two perspectives: (1) how well a neighbor can spread the node's block to the rest of the network, and (2) how fast a neighbor can notify the node of the newest blocks from the rest of the network. We propose several candidate features and apply empirical analysis to study their impacts. To illustrate, we use two simple topologies of a mining network with 16 nodes, one is shown in Fig. 3 and the other is a completely-connected graph where each node has 15 edges. We control network parameters, *e.g.* upload/download bandwidth, link latency, in different experiments for comparison. In the simulation, we treat the process of block generation and propagation for 100 rounds as a set and we repeat 10 sets in each experiment. Corresponding results and analysis are detailed in the following.

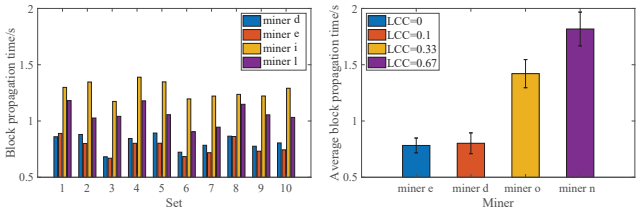
1) Impact of a Neighbor's Degree: We first analyze the impact of degree on the block propagation time and receiving time. To rule out the impact caused by nodes' different network environments, we make every connection between any two nodes with the same upload/download bandwidth and link latency in this experiment. The first comparison node pair is (j, m) and their corresponding neighbor pair is (i, l). Fig. 4(a) reflects two facts: (1) a higher-degree node itself tends to have a shorter block propagation time (by comparing node l of $D_l = 4$ and node i of $D_i = 3$), and (2) a higher-degree node can shorten its neighbor's block propagation time by comparing node m of $D_m = 2$ and node j of $D_j = 2$. Similar results can be obtained from Fig. 4(b) by choosing comparison node pair (f, g) and their corresponding neighbor pair (e, p). Thus, we conclude, a higher-degree neighbor has a better block propagation ability.

2) Impact of a Neighbor's Local Clustering Coefficient: Local clustering coefficient, denoted as C_i and expressed in Eq. 1, measures how well a node's neighbors are connected to each other, namely how close they are to being a clique.

$$C_i = \frac{|\{e_{j,k} | \forall j, k \in N_i\}|}{\frac{1}{2} D_i (D_i - 1)}, \tag{1}$$

where $\frac{1}{2} D_i (D_i - 1)$ represents the maximum possible number of edges among all node i's neighbors and $\{e_{j,k} | \forall j, k \in N_i\}$ is the set of edges connecting two of i's neighbors. The local clustering has remarkable impacts on network structure and functionality. Some literature showed that the clustering has negative correlation with degree in undirected networks and our experiments reach the same conclusion. As is shown in Fig. 5(a), node d has a higher local clustering coefficient compared with node e and its own block propagation time is longer than that of node e. Meanwhile, node d's neighbor, node i, also has a longer block propagation time compared to node e's neighbor, node l. Besides, by comparing node d of $D_d = 5$ and node e of $D_e = 4$, we can also be guided that, local clustering coefficient seems of more significance than degree. Fig. 5(b) also provides an intuitive sense that local clustering coefficient is negatively related to a node's propagation ability. Thus, we consider that a neighbor with a lower local clustering coefficient should be more suitable.

3) Impact of a Neighbor's Mining Power: In [9], the authors hold the view that there exists a small set influential nodes that skew broadcast fairness. According to their analysis, nodes with more mining power receive a block more efficiently than others. Inspired by their results, we also consider that a peer's mining power may also be a feature to reflect his propagation ability. We pick nodes f and g for comparison. According to the network topology, f and g are directly connected and have the identical mining power. However, f's neighbor e has a higher mining power than that of g's neighbor p. Fig. 6 presents a comparison of the propagation time for nodes f and g. Obviously, f's block generally propagates faster than g's. This means a neighbor's propagation ability has a positive correlation with its mining power. In particular,



(a) $C_d = 0.1$ and $C_e = 0$ (b) $D_e = 4$ and $D_p = 2$

Fig. 5: Local clustering coefficient impact on a miner and its neighbor(s).

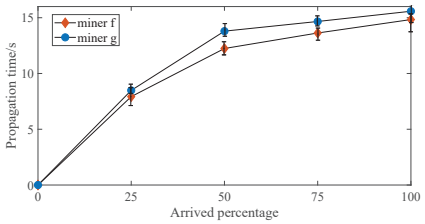


Fig. 6: Neighbors' mining powers impact a node's block propagation time.

once a block is relayed by a node, it means a portion of mining power supports this block. The more mining power extends on this block, the higher possibility this block has to be accepted by the network, if there exist competing blocks. Note that, [10] provide methods to estimate and measure the mining power of individual miners.

4) Criteria Function Fitting: Based on the empirical observation, we want to figure out a criteria function, taking as input a node's feature set value and generating as output a score to reflect this node's propagation ability. Such a criteria function allows a node to determine a peer's suitability of being a neighbor. Mathematically, a node's propagation ability, denoted by S_i, is scored by the criteria function defined as:

$$S_i = g(C_i) \sum_{j \in N_i} (D_j + 1) + m_i. \quad (2)$$

Obviously, $g(C_i)$ accounts for the effect of i's local clustering and plays a negative role in propagation. Inspired by the result from [11], we make two attempts here by adopting $g(C_i)$ as either an exponential function, *i.e.*, $g(C_i) = k \cdot \alpha^{-C_i}$, or a power function, *i.e.*, $g(C_i) = k \cdot C_i^{\alpha}$. To find a best fitting, we use machine learning to figure out the value of k and α for both attempts. Our result shows that a simple exponential function $g(C_i) = 10^{-C_i}$ is enough for S_i since complicate functions or parameter values add little meaning to score nodes but make the analysis more complicated. Indeed, the perspective and results of this paper are not limited by a very specific $g(C_i)$, as long as it is a decreasing function.

V. Metrics for Bitcoin-like Network Topology

In the following, we detail novel metrics by which a Bitcoin/Bitcoin-like topology can be evaluated as different algorithms definitely generate different topologies. Hopefully, these metrics can give a comprehensive and objective evaluation of a topology instantiation.

1) Block Propagation Time in the Bitcoin Network: We take as an important metric the time required to deliver a block from an originator to X percentage of nodes in the network, which is evaluated by most Bitcoin-like networks.

2) Consensus Delay: Since the network layer is to serve the consensus layer, we utilize consensus delay, proposed in [12], as another metric to quantify a Bitcoin network topology. As is defined in its original paper, consensus delay is that, for a specific execution and time, how long nodes have to look to find a point where they agree on the state.

3) Blockchain Fork Rate: The blockchain fork rate is another important metric, which is defined as the ratio of the number of blocks that are not included in the longest chain against the chain length. This metric indicates the effectively-utilized mining power in the current network and also indicates how much electricity waste of useless mining.

4) Fairness: As we claimed previously, it is unfair that some miners are still mining on the stale block while some have already started a new mining round, as it is an information-asymmetric situation. In this paper, we want to quantify fairness. We calculate the average block receiving time for each miner and the fairness is obtained using the maximal average block receiving time difference in the network. Optimally, the fairness is 0, *i.e.*, in the long run, any miner should wait for an identical time to receive a block if he is not the block creator.

VI. Evaluation

In this section, we evaluate the performance of various Bitcoin topology instantiations by leveraging our metrics and a blockchain simulator [13]. We set the value n to be 2 since it is the best configuration after extensive experiments. We consider a peer as nearby if the RTT is no larger than 100 ms, and consider a peer as middle if the RTT is between 100 ms and 450 ms, otherwise, he is a far peer. Besides, we distinguish between two node types, *i.e.*, relay nodes and miners, by attributing a particular non-zero mining power to each miner. For comparison purposes, we implement our proposed algorithm and another 4 algorithms, which are described as followings:

- Default: Randomly pick nodes from the known peer list to satisfy the 8-neighbor requirement.
- RTT based scoring (RTTS): This algorithm [14] allows each node to score a peer based on the round-trip-time between them and then decide its outgoing connection priority.
- Geographical distance based scoring (GDS): This algorithm [15] allows each node to score a peer based on the physical distance between them and then decide its outgoing connection priority.
- Time difference based scoring (TDS): This algorithm [16] allows each node to score a peer based on the time difference between block generation and receipt of this peer's INV message and then decide its outgoing connection priority.

A. Static Environments

We first study the effectiveness of all algorithms in a static P2P environment, where nodes do not join and leave.

1) Performance of New Network Constructions: Given a network from scratch, different algorithms can produce different topologies. In this part, we first define the number

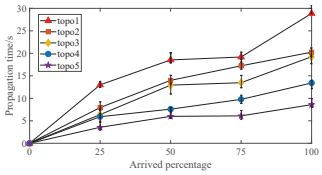


Fig. 7: Miners with different powers.

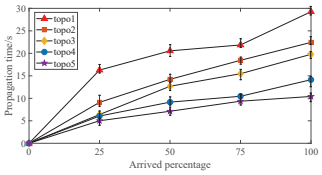


Fig. 8: Miners with identical power.

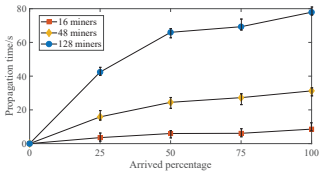


Fig. 9: Different numbers of miners.

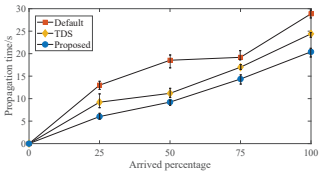


Fig. 10: Topology reorganization.

for each node type, and their geographical locations will be determined by the blockchain simulator. After that, we use each algorithm to construct a topology and evaluate the algorithm's effectiveness through the topology performance. In this experiment, we set the number of miners and the number of relay nodes to be 16 and 256, respectively.

Fig. 7 shows block propagation time used to reach a certain percentage of nodes under different network topologies. Obviously, the topology generated by our proposed algorithm performs best on this metric. In Table II, we show the results of other metrics and we can see our proposed algorithm performs well except fairness, which means a relatively big difference between the worst receiving time and the best receiving time. The reason behind this big gap may come from the reason that some node with a bad propagation ability is ignored by the entire network. Next, we equally distribute mining power among all miners, which was envisioned in the Bitcoin original whitepaper. According to Fig. 8, this slight change causes the propagation delay increases no matter what algorithm is applied to generate the network topology. This result further confirms the correctness that we consider a node's mining power as a feature of his propagation ability. We keep the relay node number unchanged while increasing the miner number. Obviously, the propagation delay becomes longer since the total number of nodes increases. However, Fig. 9 shows the delay increase is non-linear with the miner number increase, which indicates the network is still under-saturation.

2) Performance of Existing Network Reorganizations: As we stressed before, our neighbor selection algorithm is backward compatible, *i.e.*, it also improves an existing network topology after all joined nodes adopt our algorithm. To see how effective our algorithm is in improving an existing network, we first use the default algorithm to build the original topology and then optimize it with the proposed algorithm. As only TDS can update the network in a static environment, we show the propagation time of the original topology as well as the topologies generated by TDS and our proposed algorithm in Fig. 10. As mentioned in Table II, the original topology leads to a fork rate of 1.61%. After reorganization, TDS and our proposed algorithm can lower the fork rate to 1.54% and 1.18%, respectively. However, reorganization

Algorithm	Median	Broadcast	Consensus	Fork rate	Fairness
Default	13.04	19.93	755	1.61%	3.94
RTT	10.33	17.35	670	1.52%	3.82
GDS	8.79	14.19	607	1.12%	3.12
TDS	6.40	1.00	579	1.02%	3.24
Proposed	4.67	7.8	511	0.78%	3.14

TABLE II: Averaged median delay, broadcast delay, consensus delay, and worst-best receiving time among all 16 miners, and fork rate in the bitcoin network using various algorithms.

Topology	25%	50%	75%	85%	100%	fork rate
1	19.87	31.33	31.63	32.93	34.34	1.52%
2	19.81	30.71	31.27	33.38	35.77	1.82%
3	17.70	30.48	31.33	35.51	37.07	2%
4	17.06	29.99	32.10	36.44	38.50	2.22%
5	19.33	29.63	30.60	37.75	39.09	2.38%
6	18.35	27.66	34.04	38.46	39.98	2.56%

TABLE III: All-miner networks optimized by our proposed algorithm.

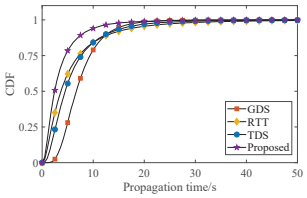


Fig. 11: Environment with churns.

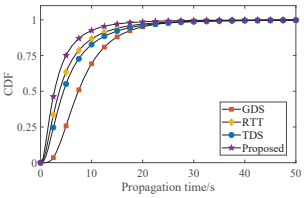


Fig. 12: Churns and link failures.

cannot reach the performance achieved by new-construction, which indicates the importance of designing a good topology formation mechanism for a Bitcoin/Bitcoin-like network.

We further consider an extreme cases, where all nodes are miners. We set the node number, *i.e.*, the miner number, as 48, and apply the default algorithm to generate 6 different random topologies. For each random topology, we run our proposed algorithm to optimize it. When we measure those optimized topologies, we obtain an interesting observation. Our previous experiment results confirm that the block broadcast is positively-correlated to the fork rate. However, it seems that, the positive correlation already happens at a certain point where even if not all miners are informed. According to Table III, we find if a topology's average block propagation time to 85% miners is shorter, then its fork rate is also lower, compared with a topology with a longer average 85% block propagation time. This observation triggers our interest in finding the relation between the block propagation time and the blockchain fork rate, which can be our future work.

B. Dynamic Environments

We further evaluate the network performance in a dynamic environment, where nodes and connections are changing. In the first setting, we add the joining and leaving of nodes, where the churn rate is modeled according to the distribution in [17]. In fact, the first three algorithms take effect only when the number of nodes changes. In the second setting, we simulate the connection failure between nodes to fully capture network dynamics and make our evaluation more sound. We plot the cumulative distribution function for each network topology and the corresponding results are shown in Fig. 11

and Fig. 12. It is clear that our proposed algorithm achieves high effectiveness compared with others.

VII. Related Work

Bitcoin network aims on information propagation while suffering from a significant delay. As Bitcoin contains two distinct types of information, *i.e.*, transactions and blocks, some works [18] focus on accelerating transaction propagation, while we are mainly concerned with block propagation. There are also many works in development to speed up block propagation. The resulting solutions can be roughly divided into three categories: (1) block compression to limit the amount of data that needs to be propagated [19–21] , (2) third-party relay networks for fast inter-miner communication [22, 23], and (3) network protocol design for topology optimization. There are a few works designing different network protocols for Bitcoin nodes [24]. [25] proposes a tree-based topology where a node should join to a tree as a leaf while ensuring the tree is as balanced as possible. [15] suggests a cluster-based topology and then their proposed algorithm requires a node chooses its closest neighbors according to their physical distances. However, these works are more applicable for constructing a new Bitcoin-like topology as they are designed for a node to choose suitable neighbors when it first enters this network or for constructing a topology from the scratch. instead of optimizing the existing Bitcoin network.

Our proposed mechanism can be used for a new Bitcoin-like network construction as well as an existing Bitcoin-like reorganization. [16] also proposes a topology reorganization algorithm, which allows a node to periodically update its outbound neighbor set and each neighbor is ranked by the difference between a block generation time and the receipt time of the block sent by this neighbor. Our algorithm also allows a node to update suitable neighbors based on scores. However, the measurement we use to rank a candidate neighbor is different from that in [16]. Meanwhile, we also introduce recommendation from current neighbors, providing a node with a better view of the global topology.

VIII. Conclusion

In this paper, we propose an autonomous topology optimization mechanism for the Bitcoin network. The main part of the mechanism is a recommendation-based neighbor selection algorithm, which allows miners to update their neighbor sets in a distributed fashion using information provided by the current neighbors. A criteria function is designed for miners to make recommendation and selection. Two metrics, *i.e.*, block propagation delay and blockchain fork rate, are used to quantify the performance of a Bitcoin network topology. Simulations show a good rate of decrease in block propagation delays (both average and maximum) and fork rates, compared to classic algorithms, and also prove the validation of our proposed propagation model.

References

[1] J. Wu, *Handbook on theoretical and algorithmic aspects of sensor, ad hoc wireless, and peer-to-peer networks*, 2005.

[2] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Communications Surveys & Tutorials*, 2005.

[3] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *IEEE P2P Proceedings*.

[4] S. Delgado-Segura, C. Pérez-Solà, J. Herrera-Joancomartí, G. Navarro-Arribas, and J. Borrell, "Cryptocurrency networks: A new p2p paradigm," *Mobile Information Systems*, 2018.

[5] R. Beverly and M. Afergan, "Machine learning for efficient neighbor selection in unstructured p2p networks," *SysML*, 2007.

[6] G. Pappalardo, T. Di Matteo, G. Caldarelli, and T. Aste, "Blockchain inefficiency in the bitcoin peers network," *EPJ Data Science*, 2018.

[7] X. Li and J. Wu, "Searching techniques in peer-to-peer networks." 2005.

[8] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," *nature*, 1998.

[9] 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.

[10] A. P. Ozisik, G. Bissias, and B. N. Levine, "Estimation of miner hash rates and consensus on blockchains," *arXiv preprint arXiv:1707.00082*, 2017.

[11] D.-B. Chen, H. Gao, L. Lü, and T. Zhou, "Identifying influential nodes in large-scale directed networks: the role of clustering," *PloS one*, 2013.

[12] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-ng: A scalable blockchain protocol," in *13th USENIX Symposium on Networked Systems Design and Implementation*, 2016.

[13] A. Gervais, G. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the 23nd ACM CCS*, 2016.

[14] W. Bi, H. Yang, and M. Zheng, "An accelerated method for message propagation in blockchain networks," *arXiv preprint arXiv:1809.00455*, 2018.

[15] M. Fadhil, G. Owenson, and M. Adda, "Locality based approach to improve propagation delay on the bitcoin peer-to-peer network," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management*, 2017.

[16] Y. Aoki and K. Shudo, "Proximity neighbor selection in blockchain networks," *arXiv preprint arXiv:1906.00719*, 2019.

[17] M. A. Imtiaz, D. Starobinski, A. Trachtenberg, and N. Younis, "Churn in the bitcoin network: Characterization and impact," in *2019 IEEE International Conference on ICBC*, 2019.

[18] G. Owenson, M. Adda *et al.*, "Proximity awareness approach to enhance propagation delay on the bitcoin peer-to-peer network," in *2017 IEEE 37th International Conference on Distributed Computing Systems*, 2017.

[19] M. Corallo, "Compact block relay. bip 152," 2017.

[20] M. T. Goodrich and M. Mitzenmacher, "Invertible bloom lookup tables," in *49th Annual Allerton Conference on Communication, Control, and Computing*, 2011.

[21] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, 1970.

[22] M. Corallo, "High-speed bitcoin relay network," 2013.

[23] "Fibre: Fast internet bitcoin relay engine." [Online]. Available: https://www.bitcoinfibre.org/

[24] S. Delgado-Segura, S. Bakshi, C. Pérez-Solà, J. Litton, A. Pachulski, A. Miller, and B. Bhattacharjee, "Txprobe: Discovering bitcoin's network topology using orphan transactions," in *International Conference on Financial Cryptography and Data Security*, 2019.

[25] J. Kan, L. Zou, B. Liu, and X. Huang, "Boost blockchain broadcast propagation with tree routing," in *International Conference on Smart Blockchain*, 2018.