Approaching an Optimal Bitcoin Mining Overlay

Suhan Jiang, *Student Member, IEEE,* and Jie Wu, *Fellow, IEEE,*

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, 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 indicate the correlation between block propagation time and fork rate. Thus, we analyze the relation between block propagation time and fork rate by applying an epidemic model to capture the block propagation process. We prove that a Bitcoin network topology with a relatively small network delay variance among all nodes produces a lower fork rate than another topology if its average block propagation time to 84% of the entire network is shorter.

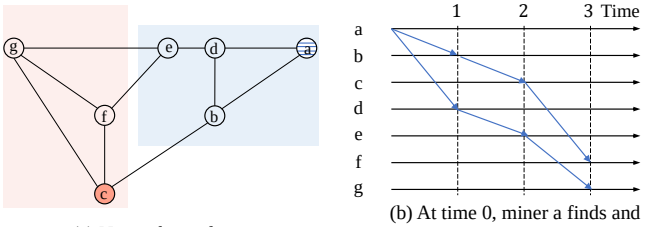
Index Terms—Blockchain, criteria function, fork, neighbor selection, P2P overlay, propagation delay.

1 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. The miners will attempt to extend the branch that they heard of first. The fork will be resolved 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 another miner, say e, successfully extends a's branch. A fork can be sustained for 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].

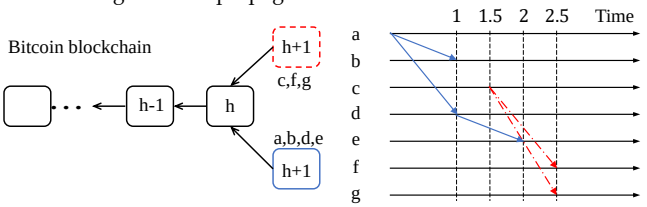
Previous studies have shown that propagation performance of a P2P overlay can be improved by optimizing its underlying topology [5–7]. We believe that the Bitcoin



(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.

• *S. Jiang and J. Wu are with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA, 19122. E-mail:{Suhan.Jiang and jiewu}@temple.edu*

This research was supported in part by NSF grants CNS 1824440, CNS 1828363, CNS 1757533, CNS 1629746, CNS 1651947, and CNS 1564128. Manuscript received January 5, 2023

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 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 decentralized 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 (tagged) miner to update his neighbor set in the following steps: each current neighbor will recommend a peer from his own neighbor set (a neighbor's neighbor), to the tagged miner, and then the tagged miner 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 his 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 correlation between block propagation time and fork rate. To figure out the relation between block propagation delay and blockchain fork rate, we apply a SEIR model for block propagation process and find mathematical proof that one topology produces a lower fork rate than another as long as its average block propagation time to $\frac{1}{2}+\frac{1}{2}erf(\frac{\sqrt{2}}{2})$ (around 84%) of the entire network is shorter.

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 his 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.
- We apply an epidemic model to describe the block propagation process and figure out the relation between block propagation time and blockchain fork rate.

The remainder of the paper is organized as follows. Section 2 summarizes current Bitcoin mining network and its neighbor selection algorithm. Section 3 presents our recommendation-based neighbor selection algorithm. In Section 4, we design a suitable criteria function to quantify a node's propagation ability based on his local information. We define several metrics to evaluate performances of a Bitcoin/Bitcoin-like network topology in Section 5. We describe our blockchain simulator in Section 6 and discuss simulation results in Section 7. In Section 8, we apply an epidemic model to describe the block propagation process and figure out the relation between block propagation time and blockchain fork rate. Section 9 briefly gives the related backgrounds, and we conclude our paper in Section 10.

2 Current Mechanism and Motivation

2.1 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 he 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. The topology optimization method applies to Bitcoin and to Bitcoin-like networks 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 bootstraps 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 [8, 9] 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 [10] that, blocks first announced by some nodes propagate consistently faster (or slower) than others and the extreme difference of block propagation time for the majority coverage can be more than 30 minutes where the fastest node only spent 2.3 s to reach 50% of the remaining peers. From an individual node's point of view, a good neighbor set can hasten the block propagation speed as well as shorten the block receiving time.

TABLE 1: 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,

2.2 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 a cryptocurrency network itself. Usually, information delivery in a content delivery network is within a small part of the network which may be traceable [11] 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.

Meanwhile, forking is a problem that cannot be ignored. According to the information from Blockchain.com, 527 orphaned Bitcoin blocks (the consequence of a fork) were observed from 03/18/2014 to 06/14/2017, a duration of 1184 days during which $170,496$ blocks were generated. Thus, the fork rate is $527/170,496$, around 0.31%. (Note, forks are no longer recorded after 06/14/2017.) The emergence of forks means wasting computing power in the system. More specifically, it is a waste of computing power contributed by honest nodes. Therefore, forks will seriously weaken the security of the blockchain, so that malicious nodes could successfully control the blockchain without 51% of the computing power.

Note that, our object is to optimize the current Bitcoin topology, while in reality, Bitcoin network may never converge to the so-called optimal topology. This is because, in such an optimal topology, it is expected that all nodes' local views achieve consistency all the time, which is not realistic due to the information propagation delay among them. The current Bitcoin network is not optimal as its transaction and block propagation delay is not negligible according to Bitcoin Monitoring.

3 Recommendation-based Neighbor Selection

Before we start our technical elaboration, we need to stress that, although the current Bitcoin mining networks are mainly constructed by pools, and a fast communication protocol called FIBRE has been proposed. the dominance of a small number of agents in the form of mining pools bring centralization to Bitcoin. Thus, we consider a real decentralized P2P network with a mass of nodes here.

Previous studies [12–14] 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 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 nature 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. This information is provided in the form of a long list of potential peers' IP addresses. This large but uninformative 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 can 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 performs neighbor selection using the proposed algorithm. And details on how to measure a peer's propagation ability are explained in section 4. Corresponding notations are shown in Table 1.

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

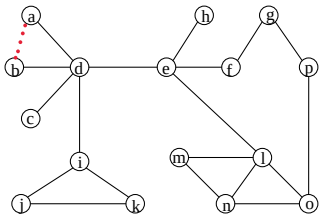
Input: node i's current neighbor set N_i, where $|N_i| = 8$
Output: an updated neighbor set N_i, where $|N_i| = 8$

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

3.1 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. The word "random" means selecting neighbors without any bias or tendency. In fact, any algorithm that guides nodes to measure other nodes with some criteria, would incur preference when selecting nodes for connection. This may lead to a case where some nodes are widely preferred and become the network central points. 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.

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 negatively affected by his long distance from i, even if j has favorable propagation qualities. The link latency makes the direct connection between i and j replaceable by several 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.



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.

Based on the analysis above and also inspired by the prior work indicating that networks with a small-world topology can spread information faster than lattice networks [15], we design our algorithm in a proximity-aware method. Node i will classify a peer j's proximity as nearby, middle, or far. From all nearby nodes, it will choose top-n (a predefined parameter) connectable nodes and put the remaining nodes in a nearby-neighbor list for future use. It will follow the similar steps to connect $(8-n)$ nodes in the middle region based on peer's propagation ability, and then the rest of middle peers will be recorded in a middle-neighbor list for future reference. Note that, a far j will not be attempted even if his 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.

3.2 Discussion on Incentive behind Recommendation

Any recommendation system has an underlying incentive in the form of a direct or an indirect reward. In our algorithm, nodes also have incentive to make recommendation to their neighbors. First, recommendation is bidirectional. Each node works toward shorter block propagation time as well as receiving time. Receiving time is the time used for a node to get the newest blocks created by other nodes. After receiving the newest block, a node will switch to mine on that block. A short receiving time indicates less power waste on mining stale blocks. This decreases the time wasted on extending stale blocks, and thereby saving his electricity cost. Also, his probability of winning block rewards will be improved, which leads to a higher profit in the long run. This is because: (1) in reality, to create a valid block, miners are required to solve a Proof-of-Work puzzle, the solution to which can be obtained by iteratively guessing. The process of guessing a solution can be modeled as a Bernoulli trial

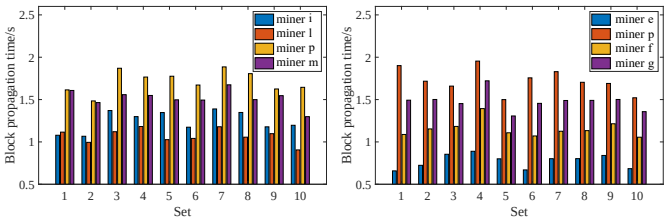
and the success rate of each trial is fixed. This indicates that, the more trials a miner is allowed to attempt, the higher chance he would have to find a solution. There are two ways for a miner to get more trials. One is to increase his mining power so that he can try more in the fixed time period, and the other is to prolong his mining time. Our point here is that a node with a good topology position can start mining the next block earlier than an edge node as it receives information faster. This is the way to prolong his mining time and thereby increases his probability of finding a solution and improving his long-term reward. (2) for a miner, the probability of finding a block is a slightly different from the probability of being rewarded, since being rewarded requires finding a block, then propagating the block faster then the conflicting block if a fork occurs. Thus, a node should increase its probability of being rewarded if it could shorten its block propagation time. Second, individual improvement leads to better communication in the network, which will decrease the whole fork rate, making Bitcoin a more robust and reliable system. In fact, Bitcoin market price is affected by the security and performance of the system itself, where a healthy network is necessary.

4 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.

4.1 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 himself 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 his neighbor's block propagation time by comparing node m with $D_m = 2$ and node j with $D_j = 2$. Similar results can be obtained from Fig. 4(b) by choosing



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

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

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.

4.2 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.

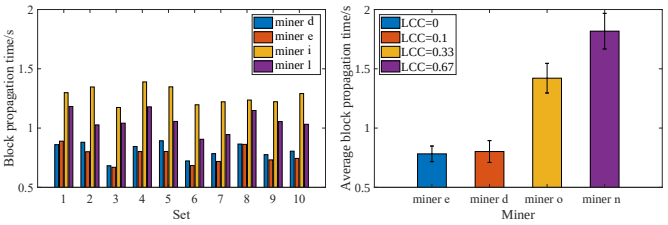
$$C_i = \frac{|\{e_{j,k} | \forall j, k \in N_i\}|}{\frac{1}{2} D_i (D_i - 1)}, \tag{1}$$

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. Note that, instead of using the traditional concept, we define a node i's local clustering coefficient based on its outgoing neighbor set N_i.

The local clustering has remarkable impacts on network structure and functionality. Studying the effects of clustering coefficient on the network evolving can provide insights into the understanding of the growing mechanism and further help us design a better criteria function and explain the observation on information spreading through the Bitcoin network. Some literature [16] 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 his 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.

4.3 Impact of a Neighbor's Mining Power

In [17], 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



(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 his neighbor(s).

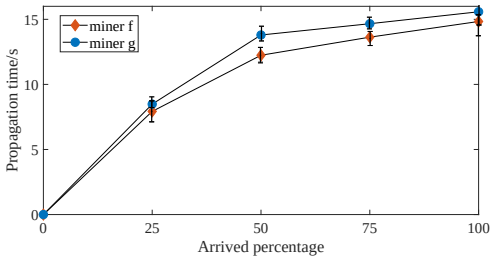


Fig. 6: Neighbors' mining powers impact a node's block propagation time.

the propagation time for nodes f and g. Obviously, node f's block generally propagates faster than node g's. This means a neighbor's propagation ability has a positive correlation with his mining power. In particular, 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, in terms of revealing the mining power of individual nodes, the following paper [18] provides a method to make estimations with quantifiable accuracy. This method requires a minerto periodically announce the smallest value he has discovered. His hash rate can be estimated using the equation of $\hat{h} = \frac{S}{\overline{V}\sigma}$. Here, $S = 2^{256} - 1$ is the size of the hash space, and σ represents the total seconds during which this miner attempts to mine a block. $\overline{V}$ is the mean of all the smallest values reported during that σ seconds.

4.4 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 in the Eq. (2).

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

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 [19], 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 the 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.

To sum up, S_i is computed by node i itself. Note that, a miner i will request its neighbor's neighbor list and find connections by itself, so that it will be able to obtain neighbors' node degrees and also calculate its clustering coefficient which are required fields in S_i. The request process should not be communication-heavy since we only consider outgoing neighbors, at most 8 for a miner.

5 Metrics for Bitcoin-like Network Topology

Since our proposed algorithm, as well as other existing works, will definitely generate different Bitcoin network topologies, another challenge is how to give a comprehensive and objective evaluation of a topology instantiation. In the following, we detail novel metrics by which a Bitcoin/Bitcoin-like topology can be evaluated. These metrics are designed relying on the premise of the PoW consensus mechanism.

5.1 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.

5.2 Consensus Delay

Since the network layer is to serve the consensus layer, we utilize consensus delay, proposed in [20], 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.

5.3 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.

5.4 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: in the long run, any miner should wait for an identical time to receive a block if he is not the block creator.

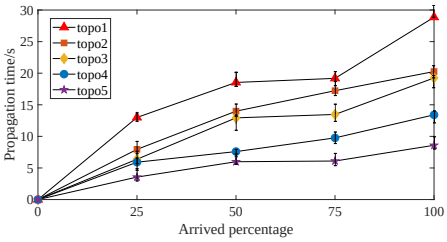


Fig. 7: 16 miners with different mining powers.

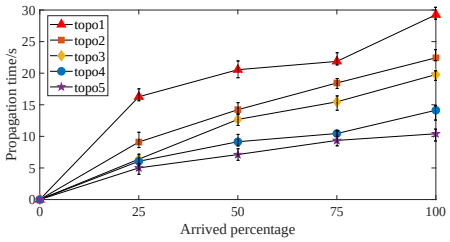


Fig. 8: 16 miners with identical mining power.

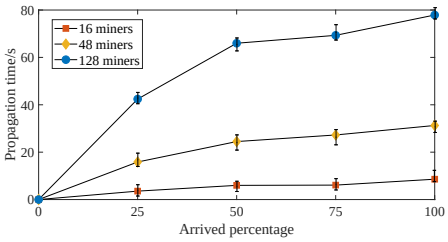


Fig. 9: Different numbers of miners.

6 Blockchain Simulator

We evaluate Bitcoin/Bitcoin-like network topologies with a blockchain simulator running on the PoW consensus. Our simulator is implemented based on existing systems [21, 22]. Our modification enables different topologies for the simulator. The inputs are node information, topology information, and block information. There are two node types, *i.e.*, relay nodes and miners, and you need to specify the relay number and the miner number, as well as miners' mining power. In terms of topology, it is optional, *i.e.*, you can specify your own topology, or leave it empty. Then the simulator will automatically create a topology based on real-world information collected from bitcoin network and the neighbor selection algorithm you implemented in the simulator. For block information, you can specify how many blocks to generate in total and how large your block size is. More details are given in the below.

6.1 Types of Nodes

In our simulator, we distinguish between two node types, *i.e.*, relay nodes and miners, by attributing a particular non-zero mining power to each miner. A relay node has functionalities of verifying blocks, and then relaying valid blocks to his neighbors. A miner, besides block verification and relay, can generate new blocks according to the PoW consensus. Bitcoin rewards miners who successfully append the blockchain, which is not implemented in our simulator, as we think this factor has negligible impact on the network performance evaluation.

6.2 Geographical Distribution of Nodes

For full nodes, we model their geographical locations using information provided by Bitnodes [23], and accordingly distributed full nodes to the respective region. For miners, we retrieve the mining pool distribution from blockchain.info [24], and accordingly distributed the mining pool's public nodes to respective regions.

6.3 PoW Consensus

The PoW consensus is applied by miners. Under this mechanism, the time it takes a miner to find a solution follows a geometric probability distribution, which can be approximated as an exponential distribution due to the improbability of a success in each guess and the rate of guessing. In our experiments we replace the proof of work mechanism with a scheduler that triggers block generation at different miners with exponentially distributed intervals. Thus, on average every 10 minutes, a new block is then attributed to a miner. Conforming with two blocks pointing to the same previous block, a miner mines on the first block he receives, and we assume that forks are resolved by the longest chain rule. Once a fork is resolved, the blocks that do not contribute to the main chain are considered stale blocks. Within our simulations, we do not consider difficulty changes among different blocks; the longest chain is therefore simply defined by the number of its blocks. Note, we do not model the propagation of transactions, since the main point of our simulator is to study the impact of the block propagation.

6.4 Mining Power Distribution

In our simulator, we design two methods: (1) all miners have to identical mining power, and (2) all miners have mining power following the distribution from blockchain.info [24].

6.5 Network Environment

We use the round-trip-time (RTT) between two nodes to quantify their proximity. The corresponding RTT between any two nodes is calculated and assigned in advance, based on their geographical distance using the function shown in [25]. To simulate a node's block verification time, we add a data processing latency T_p processing at each hop whenever a node needs to forward a block. For simplicity, we choose an empirical value of 45ms instead of modeling T_p as a function varying on the block size. We also tackle the bandwidth in a simple way by assuming that each node equally allocates his bandwidth to each connection and each connection bandwidth only varies on regions.

7 Evaluation Result

In this section, we evaluate the performance of various Bitcoin topology instantiations by leveraging our metrics and our blockchain simulator. We set the value x (see Algorithm 1) 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. 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 [26] allows each node to score a peer based on the round-trip-time between them and then decide his outgoing connection priority.
- Geographical distance based scoring (GDS): This algorithm [27] allows each node to score a peer based on the physical distance between them and then decide his outgoing connection priority.

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	10.00	579	1.02%	3.24
Proposed	4.67	7.8	511	0.78%	3.14

TABLE 2: Averaged median delay, broadcast delay, consensus delay, and worst-best receiving time (fairness) for all miners, and fork rate in the bitcoin network using various algorithms.

- Time difference based scoring (TDS): This algorithm [28] 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 his outgoing connection priority.

7.1 Definition and Measurement Recap

Before starting our experiments, we recap all definitions and measurements that we will use in the rest of this section.

(1) propagation time: the difference between the time when all miners receive a certain block and the time when it is created
(2) median delay: for a certain block, the median delay is the median among all miners' receiving times.
(3) broadcast delay: the average time for a block to reach all miners.
(4) consensus delay: for a specific execution and time, how long nodes have to look to find a point where they all reach an agreement on the state.
(5) fairness: for a certain block, fairness is measured by the difference of the last miner receiving time and the first miner receiving time in the network
(6) fork rate: the ratio of the number of blocks that are not included in the longest chain against the chain length.

7.2 Static Environments

We first study the effectiveness of all algorithms in a static P2P environment, where nodes do not join and leave. Recap that, for a network, we only need to specify the number for each node type. Each node's geographical location will be determined by the blockchain simulator based on the real-world probability model. And the construction of this network's topology is totally determined by its neighbor selection algorithm.

7.2.1 Performance of New Network Constructions

Given a network from scratch, different algorithms can produce different topologies. In this part, we first define the number for each node type, and their geographical locations will be determined by our 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. The mining power distribution and node geographical locations are determined by the simulator based on the probability.

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 2, we show

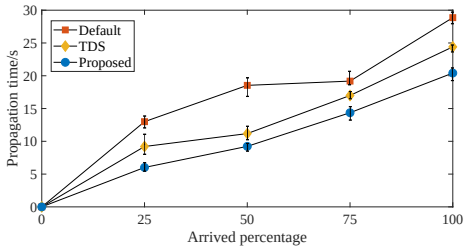


Fig. 10: Topology reorganization based on a default Bitcoin network.

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 3: All-miner networks optimized by our proposed algorithm.

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. To make our experiments more complete and more convinced, we conduct another 5 experiments with 16 miners and 256 relay nodes, and in each experiment, the node location distribution is different as it is determined by our blockchain simulator based on the probability. The averaged results over these 5 experiments are given in Table I, which further confirms our previous conclusion.

As miners' power is different in the previous experiments, we now switch to another ideal case by equally distributing mining power among all miners, which was envisioned in the Bitcoin original whitepaper. 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.

7.2.2 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 2, 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, reor-

ganization 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 3, 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 will be detailed in section VIII.

7.3 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 [29]. 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.

8 Blockchain Forking and Block Propagation

Blockchain forking is an interesting phenomenon specific in Bitcoin-like P2P networks. In [3], the authors have verified that the propagation delay in the Bitcoin network is the primary cause for its blockchain forks. Thus, many of the following works adopt either qualitative approaches, by analyzing historical or experimental data, or quantitative approaches, by building a theoretical model for the network, to figure out the relation between block propagation time and the probability of forking. In the following, we first give a comprehensive description on the blockchain forking. Then we provide a epidemic model to capture the block propagation process mathematically. Based on this model, we further analyze the relation between block propagation time and the probability of forking from the numerical perspective as well as the theoretical perspective.

8.1 On Forks in Bitcoin

The blockchain forks whenever there are two or more blocks pointing to the same previous block existing in the network. As each miner can only accept one of them, different miners may have different local copy of the blockchain. From the global view, the blockchain looks like a block tree with two or more branches, no longer a single chain. The longest path from the first block to a leaf block is defined as the main chain, and miners should reach consensus on this branch. The longest chain protocol resolves inconsistencies caused by forking while causing another problem. As when facing two conflicting blocks, a miner's acceptance of the first-seen block may not be the final acceptance. There is no guarantee that a fork ends in a single block. A fork's resolving time is another characteristic to measure the network.

A common view is that a network topology of a lower average broadcast delay can have lower chances of forking. Through our experiments, we find that, a topology with a shorter average block broadcast delay does produce a lower fork rate, while it is not necessary to shorten the broadcast delay for a lower fork rate. This interesting and reasonable observation motivates us to further consider the relation between block propagation and blockchain forking. To this end, we capture the process of block propagation using a SEIR model, which allows us to mathematically solve the problem: when evaluating two different Bitcoin network topologies in terms of fork rate, what percentage of all miners matters most?

8.2 Bitcoin Block Propagation Model

We propose to use an epidemic model to capture the dynamics of block propagation. The block creator broadcasts his block to all his neighbors while other nodes will receive, verify, accept and then relay the block to their own neighbors. If a node already accepts a block, he will reject a second-time advertisement from other nodes. A node will stop any further advertisement on a block after informing all his neighbors. This process can be described by a classical epidemic model called SEIR. Under the SEIR propagation model, a miner in the network can be in any one of the four possible states: susceptible state (S), exposed state (E), infected state (I), recovered state (R). All miners may change their own states at any time based on the network status. Now we describe how a miner's state changes in a single mining round. We assume, in the round h, miner i is mining on top of the block B_{h-1} and looking for the block B_h. He keeps in state S until he is notified of the existence of B_h by his neighbor(s). Then, he turns to state E, *i.e.*, to download and verify B_h. We only discuss the condition where all issued blocks are valid. After verification, i changes his state into I, starting to broadcast B_h to his neighbors. After announcing B_h to all neighbors, m gets into state R, where he stops advertising B_h.

Often, in a single mining round, there exists only one node issuing a valid block, which can be easily modeled by a SEIR model spreading one type of virus. The difficulty is how to model a blockchain fork, where two or more issuing nodes are competing for the same population as the receipt of a conflicting block. Since each recipient would attempt to extend (mine on top of) the first-seen block and reject the later one, in this manner, we can state that the receipt of a competing block will result in a form of cross-immunity between the nodes as once a block has been accepted, the accepting node will not accept another block that references the same source. That is, upon verifying B_h, i is immune to any other block of the height h. That is, if i receives another

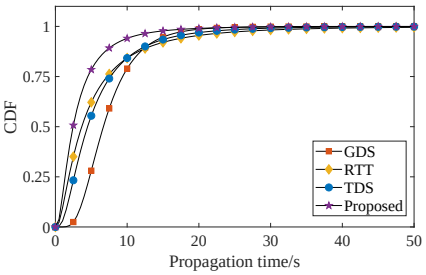


Fig. 11: A dynamic environment with churns.

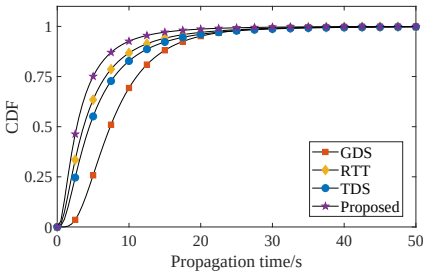
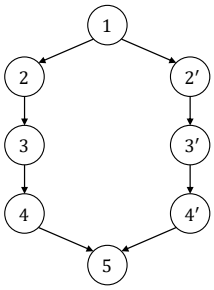


Fig. 12: Churns and link failures.

State	Explanation
S	self-mine a block of height h
E	receive and verify new block B_h
I	accept and relay B_h to neighbors
R	stop advertising B_h and mine on it

Fig. 13: State changes for a miner.



State	Explanation
1	self-mine a block
2	receive and verify new block B_h
3	accept and relay B_h, temporarily reject B'_h
4	stop advertising B_h while still mine on it
2′	receive and verify new block B'_h
3′	accept and relay B'_h, temporarily reject B_h
4′	stop advertising B'_h while still mine on it
5	converge on the accepted block of height h

Fig. 14: State transition diagram for two competing blocks.

block B'_h, he keeps in his current state without relaying B'_h to his neighbors.

We extend the original SEIR model by adding more states as shown in Fig. 14. Two miners release their own block within a time period that is sufficiently close, such that neither block has had sufficient time to be accepted throughout the entire network. Thus, a proportion of nodes on the network accept one of the competing blocks while the remaining proportion accepts the other. For an individual node, if he receives block B_h earlier than B'_h, he follows the left-side flow in Fig. 14. Otherwise, he follows the right-side one. However, no matter which block a miner chooses in the beginning, they will finally agree on only one of them.

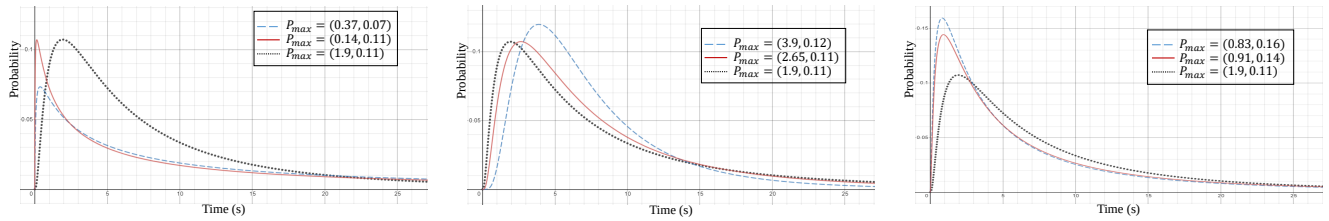
8.3 Numerical Analysis

Applying the SEIR model allows us to describe Bitcoin block propagation using a lognormal distribution, a continuous probability distribution of a random variable whose logarithm is normally distributed. A random variable X is said to follow a $Lognormal(\mu, \sigma^2)$ distribution if its probability density function (PDF) is expressed as $f_{\mu,\sigma}(x) = \frac{1}{x\sigma\sqrt{2\pi}} exp(-\frac{(\ln x-\mu)^2}{2\sigma^2})$. The corresponding density functions reflect the rate at which peers learn about a block. The propagation of a block can be divided into two phases: an initial exponential growth phase in which most of the nodes do not have it yet, and an exponential shrinking phase in which most of the nodes receiving an advertisement already have the block. Obviously, a lognormal distribution can perfectly describe these two phases, which from a side, confirms the rationality of using the SEIR propagation model. In statistics, skewness and kurtosis are important for data characterization. These two terms are captured by the tail and the maximum point in the PDF. We now try to interpret these two characters in the term of a network topology. The length of tail is a positive indicator of the worst end-to-end delay, and the maximum point's vertical axis value indicates the node degree of the network and its horizontal axis value reflects the node local clustering coefficient.

Now, we want to figure out what shape of a PDF leads to a low fork rate. First, we use data collected by [3] to approximate the PDF of the current Bitcoin network topology, which is the black dotted curve in Fig. 15. We consider this topology as a basic comparison. Based on it, we change skewness and kurtosis to obtain different shapes of PDFs (shown in Fig. 15) as well as the corresponding CDFs (shown in Fig. 16), which allow us to calculate their theoretical fork rates using Eq. 5. From our numerical analysis, we conclude that a PDF with a maximum point in the lower left corner always has a higher fork rate than that with a maximum point in the upper right area. This conclusion can be obviously verified according to Fig. 15, *i.e.*, in (a), the black dotted curve has a lower fork rate than the red solid curve as well as the blue dash curve, and in (b), the black dotted curve has a higher fork rate than the red solid curve as well as the blue dash curve. This indicates that a topology with a high degree and a low local clustering coefficient for an average node will have a high fork rate, which is coincident with our proposed algorithm.

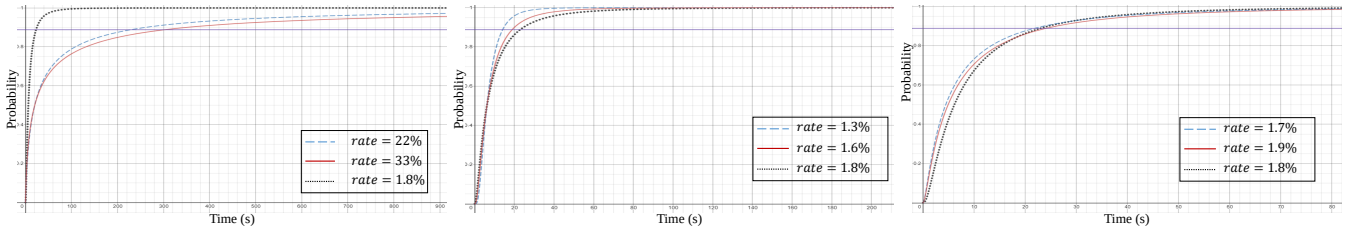
The unclear part lies in whether a PDF where the maximum point is in the upper left area can lead to a better or worse fork rate compared with a PDF where the maximum point is in the lower right area. The previous one indicates that a topology's average node degree and his local clustering coefficient are high, while the latter one indicates an opposite condition. We show the comparison sets in Fig. 15(c), the maximum points in both the red solid curve and the blue dash curve are located in the upper left side of that in the black dotted curve, but the fork rate of the red curve is higher than the black curve while the fork rate of the blue curve is lower than the black curve. In such cases, we guess there may exist a threshold, denoted by P, related to the fork rate optimization. That is, a shorter propagation time to P percentage of nodes, or more precisely, to P percentage of mining power, is enough to result in a lower fork rate.

According to our previous experiment, we find the value of P should be around 85%. That is, when comparing two different topologies, the one of which the CDF hits 85% faster would generate a lower fork rate. This result can be reflected by Fig. 16 where the purple solid line represents 85% nodes. We further investigate these six topologies by fitting their propagation times and announced node ratio to suitable lognormal distributions. We find that all of their σ values are in the range of $(0, 1)$. Thus, all discussions in the below is based on the assumption that $\sigma \in (0, 1)$. This requires that the formed topology has a relative small network delay variance among all nodes.



(a) (b) (c)

Fig. 15: Probability density function: the rate at which nodes learn about a block.



(a) (b) (c)

Fig. 16: Cumulative density function: the rate at which a block broadcasts in the network.

8.4 Theoretical Validation

We theoretically find a more precise value of P and prove its correctness. Given that $erf(x)$ is the Gauss error function [30], we have the following theorem:

Theorem 1. *Given two different Bitcoin network topologies, the one with a shorter average block propagation time to* $(\frac{1}{2} + \frac{1}{2}erf(\frac{\sqrt{2}}{2}))$ *percentage of the entire network produces a lower fork rate.*

Proof. We characterize a Bitcoin/Bitcoin-like network topology using a lognormal distribution with two parameters $(\mu.\sigma)$. For any constant $\epsilon \to 1^-$, there exists some t such that $F_{\mu,\sigma}(t) = \epsilon$. We normalize X within range $(0, t)$ and the corresponding PDF and CDF of its truncated distribution are

$$f_{\mu,\sigma}(x;t) = \begin{cases} f_{\mu,\sigma}(x)/F_{\mu,\sigma}(t) & 0 < x < t, \\ 0 & x \geq t. \end{cases} \tag{3}$$

and

$$F_{\mu,\sigma}(x;t) = \begin{cases} F_{\mu,\sigma}(x)/F_{\mu,\sigma}(t) & 0 < x < t, \\ 1 & x \geq t. \end{cases} \tag{4}$$

Ideally, for a symmetric network topology, of which the computational resource is fully distributed, the fork rate r is derived in the following equation:

$$r = 1 - (1 - \lambda)^{\int_0^t (1 - F_{\mu,\sigma}(x;t))dx}. \tag{5}$$

Since the proof-of-work is a Poisson process, the time difference follows an exponential distribution. And λ here is the parameter of the corresponding exponential distribution. In Bitcoin, λ is around 600s, which is the value of interval between two blocks. In fact, Eq. 5 is proposed and discussed in [3] in detail.

For any $topology_1$ and $topology_2$, characterized by (μ_1, σ_1) and (μ_2, σ_2), respectively, we can measure them by comparing their worst end-to-end delays, *i.e.*, t_1 and t_2, as well as their fork rates, *i.e.*, r_1 and r_2. Obviously, $T = \int_0^t (1 - F_{\mu,\sigma}(x;t))dx$ is enough to reflect the value of r, *i.e.*, a smaller T implies a smaller r. As Eq. (6) shows, T is determined by μ, σ, and ϵ, given that t is determined by μ, σ, and ϵ as well.

$$\begin{aligned} T &= \int_0^t (1 - F_{\mu,\sigma}(x;t))dx = t - \frac{1}{F_{\mu,\sigma}(t)} \int_0^t F_{\mu,\sigma}(x)dx \\ &= t - \frac{1}{2F_{\mu,\sigma}(t)} \left[2xF_{\mu,\sigma}(x) + e^{\mu+\frac{\sigma^2}{2}} erf(\frac{\sigma}{\sqrt{2}} - \frac{\ln x - \mu}{\sqrt{2}\sigma}) \right]_0^t \\ &= t - \frac{1}{2F_{\mu,\sigma}(t)} \left[\left(2tF_{\mu,\sigma}(t) + e^{\mu+\frac{\sigma^2}{2}} erf(\frac{\sigma}{\sqrt{2}} - \frac{\ln t - \mu}{\sqrt{2}\sigma}) \right) - 1 \right] \\ &= \frac{e^{\mu+\frac{\sigma^2}{2}}}{2F_{\mu,\sigma}(t)} \left(1 - erf(\frac{\sigma}{\sqrt{2}} - \frac{\ln t - \mu}{\sqrt{2}\sigma}) \right) \\ &= \frac{e^{\mu+\frac{\sigma^2}{2}}}{2\epsilon} \left(1 - erf(\frac{\sigma}{\sqrt{2}} - \frac{\ln t - \mu}{\sqrt{2}\sigma}) \right) \\ &= \frac{e^{\mu+\frac{\sigma^2}{2}}}{2\epsilon} \left(1 - (-1) \right) = e^{\mu+\frac{\sigma^2}{2}}. \end{aligned} \tag{6}$$

Note that, when t is big enough, $erf(\frac{\sigma}{\sqrt{2}} - \frac{\ln t - \mu}{\sqrt{2}\sigma})$ approximates to -1.

Next, we define $G(\mu, \sigma) \triangleq e^{\mu+\frac{\sigma^2}{2}}$. We can compare r_1 and r_2 by comparing $G(\mu_1, \sigma_1)$ and $G(\mu_2, \sigma_2)$. Now, we only need to prove the solution to Problem 1 given below is $\sqrt{2}/2$.

Problem 1 (LOWER BOUND, LB). *Find the minimum value C_{min}, such that, if $\delta_1 \geq \delta_2$, then $G(\mu_1, \sigma_1) \geq G(\mu_2, \sigma_2)$, where $\delta_1 = e^{\mu_1+\sqrt{2}\sigma_1 C_{min}}$ and $\delta_2 = e^{\mu_2+\sqrt{2}\sigma_2 C_{min}}$.*

Given the fact that the natural exponential function is monotonically increasing, we need to prove the following statement is true,

$$\begin{aligned} \text{if} \quad & \mu_1 + \sqrt{2}\sigma_1 C_{min} \geq \mu_2 + \sqrt{2}\sigma_2 C_{min} \\ \text{then} \quad & \mu_1 + \frac{\sigma_1^2}{2} \geq \mu_2 + \frac{\sigma_2^2}{2}, \end{aligned}$$

which is equivalent to $C_{min} \geq \frac{\sigma_1+\sigma_2}{2\sqrt{2}}$. When σ is within the range of $(0, 1)$, C_{min} comes out easily as $\frac{\sqrt{2}}{2}$. When we substitute $\frac{\ln t - \mu}{\sqrt{2}\sigma} = C_{min}$ into Eq. 4, the corresponding value is $\frac{1}{2} + \frac{1}{2}erf(\frac{\sqrt{2}}{2})$, which is slightly higher than 84%. □

8.5 Experimental Confirmation

To further confirm Theorem 1, we build up 5 random Bitcoin network topologies, and then measure the average block propagation time and the corresponding fork rate over each topology. All results are listed in Table 4. It is obvious that, the fork rate is positively related to 84% network block propagation time.

topo	76%	80%	83%	84%	85%	88%	92%	fork rate
1	27.8	29.3	29.6	30.6	31.1	31.4	32.3	1.28%
2	29.8	30.5	30.7	31.0	31.7	32.4	32.9	1.41%
3	27.0	30.1	31.2	32.7	34.1	34.6	34.7	1.49%
4	27.6	30.9	31.5	33.8	34.0	35.4	35.9	1.80%
5	29.3	31.6	31.7	33.7	34.2	36.5	37.1	1.91%

TABLE 4: Fork rate is positively related to 84% network propagation time in Bitcoin-like networks.

Algorithm	Median	Broadcast	Consensus	Fork rate	Fairness
Default	12.91	19.76	753	1.58%	3.87
RTT	10.70	18.08	679	1.54%	3.86
GDS	8.80	14.20	607	1.16%	3.13
TDS	6.31	10.15	580	1.09%	3.24
Proposed	4.87	8.12	523	0.81%	3.31

TABLE 5: Averaged results over 5 topologies , each having 16 miners and 256 relay nodes. In order not to change the number of other tables, we put this table here.

8.6 Guidelines on Future Network Optimization

Theorem 1 leaves us an open question: whether we can design a network optimization mechanism to improve 84%-mining-power propagation performance, such that fork rate in the Bitcoin blockchain can be decreased. Since Bitcoin mining power is dominated by several big mining pools, by only considering 84% mining power in the Bitcoin network, we are allowed to optimize a smaller and more stable Bitcoin network. This would definitely lower the complexity for the mechanism design but still benefits the whole system's consistency.

9 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 [12] 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 [31–35] , (2) third-party relay networks for fast inter-miner communication [36–38], and (3) network protocol design for topology optimization. There are a few works designing different network protocols for Bitcoin nodes [39]. [40] 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. [27] suggests a cluster-based topology and then their proposed algorithm requires a node chooses his closest neighbors according to their physical distances. [26] designs a similar algorithm while the difference lies in the definition of closeness, which is captured by the round-trip-time between two nodes. However, these works are designed for a node to choose suitable neighbors when he first enters this network or for constructing a topology from the scratch. They are more applicable for constructing a new Bitcoin-like topology 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. [28] also proposes a topology reorganization algorithm, which allows a node to periodically update his 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 [28]. Meanwhile, we also introduce recommendation from current neighbors, providing a node with a better view of the global topology.

We consider fork rate as an important metric for the Bitcoin network measurement, which is not fully discussed previously. We also investigate the relation between block propagation time and blockchain forking. Previous works on this topic focus on analyzing real-world data collected from Bitcoin blockchain [3, 41, 42] while we propose an SEIR model, which is widely applied to describe message dissemination in P2P networks [43, 44], to capture the block propagation process, allowing us to theoretically quantify the relation between block propagation time and blockchain fork rate.

10 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. We further figure out the relation between block propagation delay and blockchain fork rate by using an SEIR model to describe the block propagation process. 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 as well as the relation between these two metrics.

References

[1] 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*.

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

[3] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *13th IEEE International Conference on Peer-to-Peer Computing*. IEEE, 2013.

[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] Y.-h. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," *IEEE Journal on selected areas in communications*, vol. 20, no. 8, pp. 1456–1471, 2002.

[6] A. Nakao, L. Peterson, and A. Bavier, "A routing underlay for overlay networks," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, 2003, pp. 11–18.

[7] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-aware overlay construction and server selection," in *Proceedings. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3. IEEE, 2002, pp. 1190–1199.
[8] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making gnutella-like p2p systems scalable," in *Proceedings of ACM SIGCOM*. ACM, 2003.
[9] R. Beverly and M. Afergan, "Machine learning for efficient neighbor selection in unstructured p2p networks," *SysML*, 2007.
[10] G. Pappalardo, T. Di Matteo, G. Caldarelli, and T. Aste, "Blockchain inefficiency in the bitcoin peers network," *EPJ Data Science*, 2018.
[11] X. Li and J. Wu, "Searching techniques in peer-to-peer networks," 2005.
[12] 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 (ICDCS)*. IEEE, 2017, pp. 2411–2416.
[13] M. Castro, P. Druschel, Y. C. Hu, A. Rowstron *et al.*, "Exploiting network proximity in peer-to-peer overlay networks," Citeseer, Tech. Rep., 2002.
[14] L. Zhang, J. K. Muppala, and W. Tu, "Exploiting proximity in cooperative download of large files in peer-to-peer networks," in *Second International Conference on Internet and Web Applications and Services (ICIW'07)*. IEEE, 2007.
[15] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," *nature*, 1998.
[16] S. N. Soffer and A. Vazquez, "Network clustering coefficient without degree-correlation biases," *Physical Review E*, 2005.
[17] A. Miller, J. Litton, A. Pachulski, N. Gupta, D. Levin, N. Spring, and B. Bhattacharjee, "Discovering bitcoin's public topology and influential nodes," *URL https://allquantor. at/blockchainbib/pdf/miller2015topology. pdf*, 2015.
[18] A. P. Ozisik, G. Bissias, and B. N. Levine, "Estimation of miner hash rates and consensus on blockchains," *arXiv preprint arXiv:1707.00082*, 2017.
[19] 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.
[20] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-ng: A scalable blockchain protocol," in *13th USENIX Symposium on NSDI*, 2016.
[21] M. Carlsten, H. Kalodner, S. M. Weinberg, and A. Narayanan, "On the instability of bitcoin without the block reward," in *Proceedings of the 23nd ACM CCS*. ACM, 2016.
[22] 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*. ACM, 2016.
[23] "Bitnodes." [Online]. Available: https://bitnodes.earn.com/
[24] "blockchain.info." [Online]. Available: https://www.blockchain.com/pools
[25] S. Park, S. Im, Y. Seol, and J. Paek, "Nodes in the bitcoin network: Comparative measurement study and survey," *IEEE Access*, 2019.
[26] W. Bi, H. Yang, and M. Zheng, "An accelerated method for message propagation in blockchain networks," *arXiv preprint arXiv:1809.00455*, 2018.
[27] 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*. IEEE, 2017.
[28] Y. Aoki and K. Shudo, "Proximity neighbor selection in blockchain networks," *arXiv preprint arXiv:1906.00719*, 2019.
[29] M. A. Imtiaz, D. Starobinski, A. Trachtenberg, and N. Younis, "Churn in the bitcoin network: Characterization and impact," in *2019 IEEE International Conference on Blockchain and Cryptocurrency*. IEEE, 2019.
[30] "Error function." [Online]. Available: https://en.wikipedia.org/wiki/Error_function
[31] M. Corallo, "Compact block relay. bip 152," 2017.
[32] P. Tschipper, "Buip010: Xtreme thinblocks," in *Bitcoin Forum (1 January 2016). https://bitco. in/forum/threads/buip010-passed-xtreme-thinblocks*, vol. 774, 2016.
[33] A. P. Ozisik, G. Andresen, B. N. Levine, D. Tapp, G. Bissias, and S. Katkuri, "Graphene: efficient interactive set reconciliation applied to blockchain propagation," in *Proceedings of the ACM Special Interest Group on Data Communication*. ACM, 2019.
[34] M. T. Goodrich and M. Mitzenmacher, "Invertible bloom lookup tables," in *49th Annual Allerton Conference on Communication, Control, and Computing*. IEEE, 2011.
[35] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, 1970.
[36] M. Corallo, "High-speed bitcoin relay network," *November*, 2013.
[37] "Falcon - a fast bitcoin backbone." [Online]. Available: https://www.falcon-net.org/
[38] "Fibre: Fast internet bitcoin relay engine." [Online]. Available: https://www.bitcoinfibre.org/
[39] 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*. Springer, 2019.
[40] J. Kan, L. Zou, B. Liu, and X. Huang, "Boost blockchain broadcast propagation with tree routing," in *International Conference on Smart Blockchain*. Springer, 2018.
[41] T. Neudecker and H. Hartenstein, "Short paper: An empirical analysis of blockchain forks in bitcoin," in *International Conference on Financial Cryptography and Data Security*. Springer, 2019.
[42] B. Liu, Y. Qin, and X. Chu, "Reducing forks in the blockchain via probabilistic verification," in *2019 IEEE 35th International Conference on Data Engineering Workshops*. IEEE, 2019.
[43] K. Leibnitz, T. Hoßfeld, N. Wakamiya, and M. Murata, "Modeling of epidemic diffusion in peer-to-peer file-sharing networks," in *International Workshop on Biologically Inspired Approaches to Advanced Information Technology*. Springer, 2006.
[44] C. Liu and Z.-K. Zhang, "Information spreading on dynamic social networks," *Communications in Nonlinear Science and Numerical Simulation*, 2014.

Suhan Jiang received her B.S. degree in Computer Science and Engineering from University of Electronic Science and Technology of China, Chengdu, China, in 2016. She received her Ph.D. Degree from the Department of Computer and Information Sciences, Temple University, Philadelphia, Pennsylvania, USA, in 2022. Her current research focuses on cloud computing and distributing systems, including blockchain techniques.

Jie Wu is the Director of the Center for Networked Computing and Laura H. Carnell professor at Temple University. He also serves as the Director of International Affairs at College of Science and Technology. He served as Chair of Department of Computer and Information Sciences from the summer of 2009 to the summer of 2016 and Associate Vice Provost for International Affairs from the fall of 2015 to the summer of 2017. Prior to joining Temple University, he was a program director at the National Science Foundation and was a distinguished professor at Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. Dr. Wu regularly publishes in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including IEEE Transactions on Mobile Computing, IEEE Transactions on Service Computing, Journal of Parallel and Distributed Computing, and Journal of Computer Science and Technology. Dr. Wu was general co-chair for IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, ACM MobiHoc 2014, ICPP 2016, and IEEE CNS 2016, as well as program co-chair for IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and chair for the IEEE Technical Committee on Distributed Processing (TCDP). Dr. Wu is a CCF Distinguished Speaker and a Fellow of the IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.