BlockP2P: Enabling Fast Blockchain Broadcast with Scalable Peer-to-Peer Network Topology

Weifeng Hao[1], Jiajie Zeng[1], Xiaohai Dai[1], Jiang Xiao[1(✉)], Qiangsheng Hua[1], Hanhua Chen[1], Kuan-Ching Li[2], and Hai Jin[1]

[1] National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Scinece and Technology, Wuhan 430074, China
jiangxiao@hust.edu.cn

[2] Department of Computer Science and Information Engineering, Providence University, Taichung, Taiwan

Abstract. Blockchain technology offers an intelligent amalgamation of distributed ledger, *Peer-to-Peer* (P2P), cryptography, and smart contracts to enable trustworthy applications without any third parties. Existing blockchain systems have successfully either resolved the scalability issue by advancing the distributed consensus protocols from the control plane, or complemented the security issue by updating the block structure and encryption algorithms from the data plane. Yet, we argue that the underlying P2P network plane remains as an important but unaddressed barrier for accelerating the overall blockchain system performance. Our key insights from comparative assessments reveal the fact that P2P topology highly affects the broadcast speed of blockchain data, leading to poor performance and vulnerable to double spending attacks. In this paper, we introduce BlockP2P, a novel optimization design to accelerate broadcast efficiency and meanwhile retain the security. BlockP2P first operates the geographical proximity sensing clustering, which leverages K-Means algorithm for gathering proximity peer nodes into clusters. It follows by the hierarchical topological structure that ensures strong connectivity and small diameter based on node attribute classification. We finally propose the parallel spanning tree broadcast algorithm to enable fast data broadcast among nodes both in the intra- and inter- clusters. To clarify the influence of each tier, we carefully design and implement a blockchain network simulator. Evaluation results show that BlockP2P can exhibit promising performance compared to Bitcoin and Ethereum.

Keywords: Blockchain Peer-to-Peer network · Network clustering · Network topology · Broadcast algorithm

R. Miani et al. (Eds.): GPC 2019, LNCS 11484, pp. 223–237, 2019.
https://doi.org/10.1007/978-3-030-19223-5_16

1 Introduction

Today blockchain technology has attracted increasing attention as the cornerstone of trust across a wide realm of society sectors from finance, industrial logistics to healthcare. Its beneficial characteristics including traceability, decentralization, and transparency spout out the massive proposals of blockchain systems and projects. Unfortunately, the real-world blockchain adoption experiences serious technical challenges that impede its further development, especially from the aspect of the overall system performance. State-of-the-art researches mainly focus on advancing the consensus algorithms in the consensus layer [11], as well optimizing the data storage in the data layer [25]. However, few studies have been conducted from the network layer (i.e., lying between the two layers) that can update the topology under consensus guarantee, while adapting the dynamic on-chain blockchain data traffic.

In particular, the lack of consideration in the blockchain network layer can not only lead to poor performance, meanwhile bring about high risks of double spending attacks. As shown in Table 1, it normally takes on average 6 s to ensure a block received by 50% of the total nodes in the Ethereum network, and up to 10 s, on average, for 90% of the nodes [2]. Since the generation time of a new block is only 15 s in Ethereum, this network-level latency becomes a major barrier that limits the blockchain performance (i.e., *Transactions Per Second* (TPS)), leading to high potential of forks. Therefore, it is urgent to reduce blockchain network latency, so as to improve the overall performance of blockchain systems with stronger security.

Table 1. Broadcast time per block

Ratio of nodes	50%	90%
Time (s)	6	10

Existing trials related to optimize the network latency can be divided into two categories, according to the different topologies they are concerned about. On one hand, the fully distributed unstructured topology of Bitcoin network is optimized by shortening the network diameter. However, it will bring in huge computation overheads because each node needs to repeatedly calculate the network distance between it and all the rest ones [21,22]. On the other hand, Ethereum employs the fully distributed structured topology that increases the connectivity of the entire network, but the growth of Ethereum nodes will introduce high maintenance cost of such structured topology.

To compromise the above limitations, in this paper, we propose an optimized network protocol namely *BlockP2P* to minimize the total blockchain network latency. First, BlockP2P gathers the proximity peer nodes into clusters based on the K-Means algorithm. We then optimize the inter-cluster topology by organizing the nodes into a Harary-like graph with high connectivity and small diameter. Finally, a parallel spanning tree broadcast algorithm is designed to speed

up the data broadcast, by eliminating the multiple rounds of message in a single communication process. To facilitate the evaluation of performance in the large-scale network, we design and implement *BlockSim*, a simulator to simulate the running of blockchain network without affecting the accuracy of the evaluation results. With the help of BlockSim, we conduct several experiments to compare BlockP2P protocol with the counterpart protocols in Bitcoin and Ethereum. The experimental results show that BlockP2P protocol can effectively reduce blockchain network latency.

In summary, this paper makes the following novel contributions:

- To the best of our knowledge, this is the first in-depth analysis of influential factors of blockchain performance in the network layer, by uncovering the two sequential phases of the underlying P2P network.
- We introduce an optimized blockchain network protocol *BlockP2P* to reduce blockchain network latency.
- To verify the feasibility and efficiency of BlockP2P protocol, we design and implement *BlockSim*, a blockchain simulator for large-scale network simulation.
- Experimental results demonstrate that BlockP2P can effectively reduce network latency from three different aspects compared to Bitcoin and Ethereum.

The rest of the paper is organized as follows. Section 2 presents the background knowledge about the network protocol of current blockchain systems. Related work on network optimization is stated in Sect. 3. Section 4 elaborates the design of BlockP2P in three steps. Extensive experiments are conducted in Sect. 5 to evaluate the system performance in terms of reducing network latency. Finally, we conclude the paper in Sect. 6.

2 Background

P2P network enables direct information interaction between different nodes in the blockchain network. As shown in Fig. 1, the process of information interaction between two nodes can be divided into two phases: connection establishment marked by gray circles, and data transmission marked by red circles. Since the connection status among nodes is relatively stable and the time taken to establish connection is usually very short, the most important component of the total network latency is the broadcast latency in the phase of data transmission. However, configurations of both phases can have effects on the broadcast latency.

In the phase of connection establishment, different network topologies may be formed among the nodes. Different network topologies will have different effects on the broadcast latency, which can be measured by network connectivity and network diameter [5]. The network connectivity refers to the number of neighbor nodes connected to each node in the network. The larger the network connectivity is, the more neighbors a node can broadcast the data each time. In this way, the overall time spent on the network broadcasting can be reduced. The network diameter refers to the maximum network delay between any two nodes in the

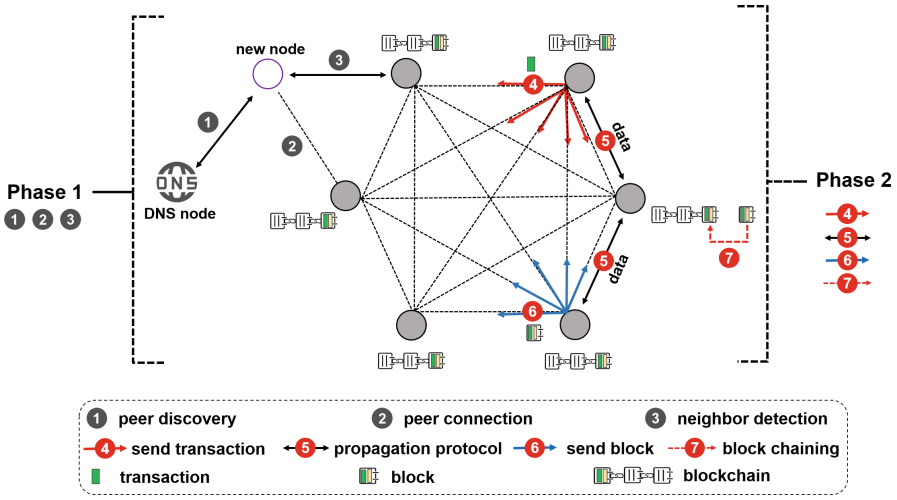


Fig. 1. Workflow of blockchain P2P network

network. The smaller the network diameter, the shorter the average broadcast time between any two nodes, thus accelerating the overall broadcast time across the network. As a result, optimizing the network topologies of nodes including the network connectivity and diameter can effectively reduce the broadcast latency.

In the phase of data transmission, Gossip algorithm is used to broadcast the data from a node to its neighbors [16]. During the process of broadcast, a node firstly selects the nodes from its neighbors to disseminate the data using the propagation protocol. The node which receives the data repeats the process above until all the nodes in the network have received the data. More specifically, the propagation protocol [8,15] used by Gossip algorithm further includes three steps. First, a node (i.e., sender) sends an *INV* message to its neighbor node before sending one piece of data (namely a transaction or a block in the context of blockchain). Second, the neighbor node determines whether it has received the data before. If not, it returns a *getdata* message back to the sender; otherwise, it ignores the *INV* message. Finally, before the end of timeout set by the sender, if the sender receives the *getdata* message, it sends the piece of data to the neighbor. It should be noted that a node only broadcasts data to its directly connected neighbors. The broadcast process will run in many rounds by each node, until each node in the network has received the data. As a result, Gossip protocol may lead to large data broadcast latency due to many rounds of broadcast. Besides, three steps of the propagation protocol bring extra communication rounds, which exacerbate the problem of large broadcast latency.

3 Related Work

The information propagation delay reveals the performance of blockchain systems [11], since the high latency increases the time for all the nodes to reach a consensus. The high network latency makes the system more vulnerable to malicious attacks [23]. Network latency is related to the interaction of information between nodes as mentioned in Sect. 2. The interaction consists of two phases: the connection establishment and the data transmission. We will present the existing optimization works in these two phases.

Connection Establishment. The topological structure of the mainstream blockchain systems can be divided into two categories: one is the unstructured topology in Bitcoin [20], the other is the structured topology in Ethereum [2] (i.e., Kademlia) and NKN [4] (i.e., Chord). To measure the quality of network topology, two metrics including network diameter and connectivity are adopted. Nodes in unstructured topology are randomly connected, which results in a large network diameter. In order to minimize the network diameter, BCBPT protocol utilizes the proximity clustering algorithm based on the number of network hops between nodes, and then connects the nodes that are physically proximal [12]. However, BCBPT brings in high algorithm complexity, as each node needs to calculate the network hops to all other nodes. Croman et al. reveal that Bitcoin cannot fully utilize the bandwidth in the network, which has serious impact on transactions processing, then they proposed to reduce the network latency of blockchain starting with optimizing the network topology [6]. Compared with the unstructured topology, the structured topology has a good network connectivity. But its network diameter is also very large, since the network latency between nodes is not taken into consideration when they try to establish a connection. Moreover, creating the structured network topology brings in huge computation cost, because of the large size of blockchain network (e.g., the size of nodes in Ethereum has almost reached to $10k$). The cost will increase significantly as the network size further increases, the same with the network latency.

Data Transmission. The blockchain network is the broadcast channel for data. Some efficient broadcast protocols [13,14] are proposed to speed up the progress of broadcast. Bitcoin employs the flood-based [10] algorithm to broadcast the data, while Ethereum adopts the gossip-based [24] broadcast algorithm. Both of these two algorithms bring huge redundant data in broadcast, because the data will be sent multiple rounds before it meets the terminational conditions of the broadcast. Besides, the multi-message transfer in the propagation protocol greatly lowers the speed of data broadcast [9,15]. An attempt to solve the problem above is conducted by Decker et al. [8], which tries to optimize the Bitcoin network by removing the process of verification and pipelining the process of block propagation. However, their ideas are only at the conceptual stage and further experiments are needed to prove it.

4 Design

Figure 2 gives an overview of how the BlockP2P protocol operates, which is composed of three parts: node clustering, topology construction, and broadcast optimization. First, to reduce the complexity of building the network topology for the whole network and ensure parallel broadcast between clusters, a *Geographical Proximity Sensing Clustering* (GPSC) method based on the K-Means algorithm [7] is devised. Second, a *Structured Hierarchical Network Topology* (SHNT) approach is proposed to construct the topology of node connection with a high network connectivity and a small diameter. Third, we design a *Parallel Spanning Tree Broadcast* (PSTB) mechanism to parallelize the broadcast processes in both intra-cluster and inter-cluster nodes.

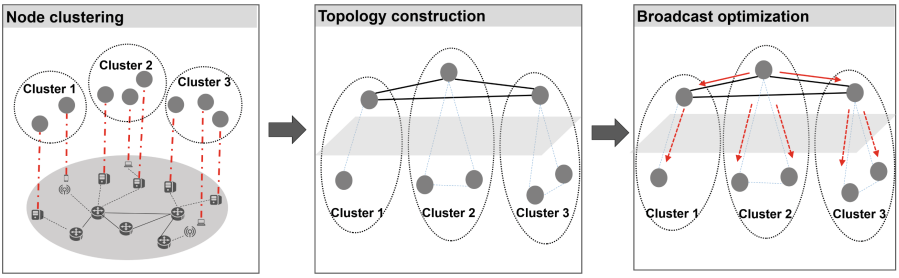


Fig. 2. Overview of the BlockP2P

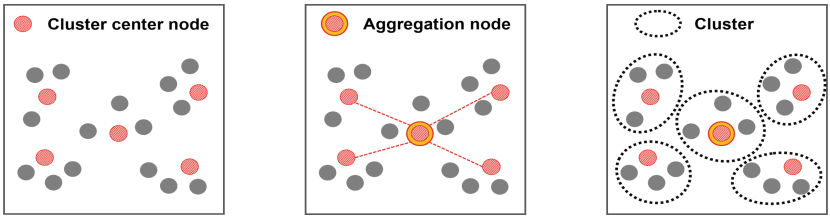
4.1 Node Clustering

To guarantee proximal and coequal clustering, BlockP2P implements the GPSC method to organize the nodes across the network into several clusters, based on the well-known K-Means algorithm. The average number of nodes in a cluster is the key parameter in the K-Means algorithm that requires careful design. On one hand, such number can not be set too large, otherwise, it will bring in high communication latency between two intra-cluster nodes. On the other hand, a small value may increase the communication cost between two inter-cluster nodes, since it enlarges the number of clusters. According to the previous studies in [5] and [17], the optimal setting of the number of nodes in a cluster should be $logN$. After the number of nodes in a cluster is set, GPSC organizes all the nodes into several clusters in three stages as depicted in Fig. 3(a).

Selection of Cluster Centers. First, we describe how GPSC selects the nodes as cluster centers. One simple way is to perform iterative computation of K-Means algorithm continuously. However, it brings in huge computation costs since it requires each node to measure the network latency between it and all the

other nodes, whose computation costs are too high. To reduce the computational complexity, GPSC creates a candidate subset for selecting cluster centers in advance. In particular, with network latency as the Euclidean distance between two nodes, GPSC selects the cluster centers in three steps as follows:

- **Step 1:** Calculate the Euclidean distance $T(n_i, n_j)(i \neq j)$ between any two nodes in the candidate subset. Find the nodes pair with the furthest distance to form a new set $S_m(1 \leq m \leq K)$, where K represents the number of network clusters, and then delete the two nodes from the candidate subset;
- **Step 2:** Add the node which is furthest from the new set to update S_m, and then remove the node from the original candidate set;
- **Step 3:** If the number of nodes in S_m is smaller than K, repeat Step 2; otherwise, the nodes in the set S_m are taken as the cluster centers.



(a) Selection of cluster centers (b) Choice of aggregation nodes (c) Network clustering

Fig. 3. Three stages of node clustering

Choice of Aggregation Nodes. To ensure all the cluster centers are evenly distributed, assistant nodes named *aggregation nodes* C_{aggre} are chosen, each of which is located at the geometric center of a cluster. It is difficult to calculate C_{aggre} only according to the network latency between two nodes. Therefore, GPSC adopts the method of *network coordinate system* (NCS) to figure out C_{aggre} [19]. First, GPSC sets a network coordinate for each cluster center node according to Eq. (1).

$$F(H_{S_1}, ..., H_{S_k}) = \sum_{S_i, S_j \in \{S_1, ..., S_k\}, i>j} \varepsilon(d_{S_i S_j}, \bar{d}_{S_i S_j}) \tag{1}$$

where S_i and H represent the cluster centers and the network coordinates of the cluster centers respectively, d and $\bar{d}$ represent the network distances between two nodes in the actual system and network coordinate system separately, and ε represents the error function. After getting the geometric center coordinates $\bar{C}_{aggre}$, GPSC chooses the cluster aggregation node according to Eq. (2).

$$\varphi(C_{aggre}) = \bar{D_{min}}(H_{S_i}, \bar{C}_{aggre}), \qquad S_i \in \{S_1, ..., S_k\} \tag{2}$$

where φ represents the matching function of the cluster aggregation node, $\bar{D_{min}}$ represents the minimum network distance between two nodes in NCS, and $\bar{C}_{aggre}$ represents the geometric center coordinates.

Network Clustering. Relying on the above prerequisites, GPSC finally clusters all the nodes according to an objective function $D(X_i, S_j)$ in Eq. (3).

$$D(X_i, S_j) = \omega_1 \times d_1(X_i, S_j) + \omega_2 \times d_2(S_j, C_{aggre}), \qquad \omega_1 + \omega_2 = 1 \tag{3}$$

where X_i and S_j represent the general node and the center node respectively, d_1 represents the distance between a node and a center while d_2 represents the distance between a center and an aggregation node. Besides, ω_1 and ω_2 are two weight factors. Compared to the $O(N^2)$ complexity of BCBPT [12], GPSC can decrease algorithm complexity to $O(K \cdot N)$, which enables the fast re-clustering of nodes in response to the possible network change, thus promoting the system's robustness.

4.2 Topology Construction

The execution of the GPSC algorithm could result in hundreds of nodes in a cluster. In this way, a node has to select a small subset from the cluster to constitute its neighbors, thus constructing the network topology. As previously mentioned in Sect. 2, network connectivity and cluster diameter can have significant effects on the blockchain broadcast performance. To enable each cluster to have an optimal network connectivity and diameter, we introduce the SHNT approach to construct the network topology as shown in Fig. 4. More precisely, SHNT consists of network initialization and maintenance processes.

Network Initialization. The nodes can be divided into SPV nodes and full nodes according to their roles in the blockchain network. Compared to the full nodes, the data broadcasted by SPV nodes is much less and lighter. Due to their different behaviours in the network, SHNT regards the SPV nodes and full nodes as leaf nodes and core nodes respectively. Besides, SHNT selects one core node from each cluster as the routing node, according to the node ID randomly, which ensures the security and randomness. Routing nodes allow the data transmitted from one cluster to another. Once a piece of data is transmitted from one routing node to another, the data can concurrently broadcast in these two clusters, thus speeding up the data transmission across the whole network. The detailed description of different nodes are listed as follows.

- **Leaf node:** consisting of SPV nodes, periodically sending node information to the core node and initiating a transaction.
- **Core node:** consisting of mining nodes, maintaining and managing leaf nodes of the cluster which they are located at, and forwarding transactions or blocks among nodes in the cluster.

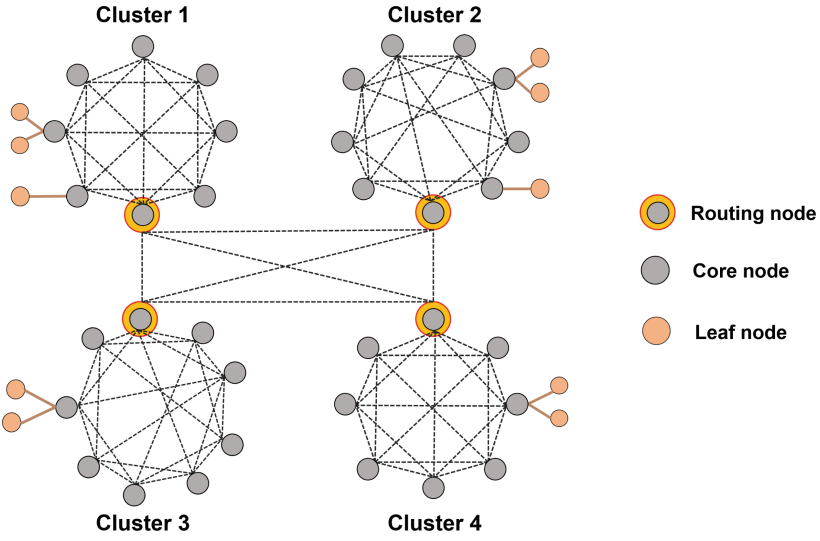


Fig. 4. Structured hierarchical network topology

- **Routing node:** selected from core nodes, storing routing node information about other clusters, and forwarding transactions or blocks among nodes in the cluster.

Based on the classification of blockchain network nodes, the construction process of blockchain network topology by SHNT can be divided into three steps. First, a leaf node is connected directly to a core node that is closest to it. Second, we construct the topology among the core nodes as a Harary-like graph, which has high connectivity and small diameter [5]. Third, a node from the core nodes is selected randomly as the routing node. Once being selected as the routing node, it will contain all the network information of other clusters. If the routing node breaks down, one of the other core nodes will replace it.

Network Maintenance. After initializing the network topology in Sect. 4.2, a natural and important concern is how to maintain the stability of network topology, especially when facing the dynamic changes, i.e., the join of new nodes and leave of old nodes. Recall that the average number of nodes in a cluster is of great importance. It will incur huge communication overhead, no matter the number of nodes is set too large or too small. To overcome the challenges brought by dynamic network, we adopt an automatic adjustment mechanism that can keep the size of each cluster stable within $O(\log N)$. In particular, the mechanism merges small clusters when many nodes churn to leave, and splits large clusters if a large number of nodes join in the same clusters. The minimal threshold to trigger *cluster merge* and maximal threshold to trigger *cluster split* are set to $\frac{\log N}{l}$ and $l \times \log N$, respectively.

4.3 Broadcast Optimization

In order to improve the broadcast efficiency, a *Parallel Spanning Tree Broadcast* (PSTB) method is adopted. As shown in Fig. 5, the data will be sent by one routing node to the rest according to the route table. Once a routing node receives the data from other clusters, it will send the data to the nodes in the same cluster along a spanning tree.

Inter-cluster Broadcast. The inter-cluster broadcast of data is based on the routing nodes of different clusters. Each routing node stores a route table, which records the information of all the other routing nodes. As shown in Fig. 5, once the routing node receives the data, it will not only broadcast in its cluster, but also forward the data to other routing nodes as well, according to the route table. Hence, this method can enable parallel and fast data broadcast. In order to reduce the security problems caused by the crash of routing nodes, PSTB proposes a backup mechanism for routing table, which randomly selects a node from the core nodes as the backup node.

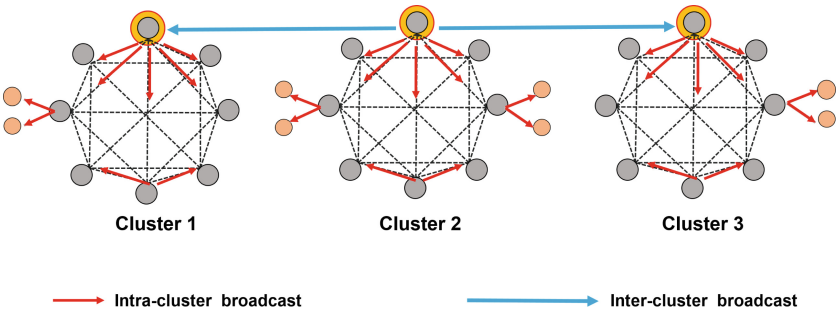


Fig. 5. Parallel spanning tree broadcast

Intra-cluster Broadcast. Only the routing node in a cluster will broadcast the data, which effectively avoids the huge network overheads brought by Gossip protocol [24]. If the broadcast source in the cluster is not a routing node, the source node will firstly send the data to the routing node. Once the routing node receives the data, it will broadcast the data in the cluster along the spanning tree. As for a spanning tree table, PSTB adopts the center-based approach to build it. First, the protocol selects a central node, and then all other nodes unicast the *INV* message to the central node to join the tree directly. In order to deal with the interference caused by the dynamic network changes, each routing node will update the spanning tree table periodically to enable the timeliness of the algorithm.

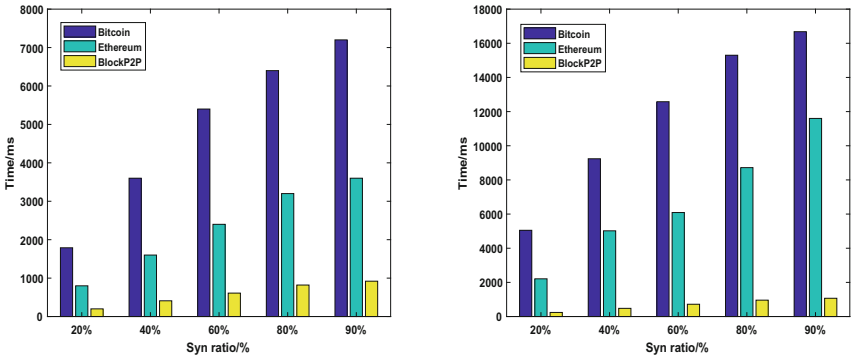
5 Evaluation

In this section, we conduct several experiments to evaluate BlockP2P. First of all, we introduce the experimental setup, including the platform configuration, implementation, and the evaluation metrics. Then, we analyze the experimental results from comprehensive perspectives, and compare BlockP2P with Bitcoin and Ethereum.

5.1 Experimental Setup

Platform Configuration. We conduct our experiments on two machines, each of which has two eight-core Intel Xeon E5-2670 2.60 Hz CPUs, 64 GB DRAM, 300 GB HDD, and InfiniBand network card, with CentOS 7.0 as the operating system.

Implementation. Without sacrificing the accuracy of experimental results, the simulation methods are adopted. In this paper, we design a generic blockchain network simulator named BlockSim based on peersim [18]. BlockSim is consisted of three core parts: *simulation-network*, *simulation-consensus*, and *simulation-data*. To simulate different network environments in the blockchain, developers can implement the interfaces provided by BlockSim as needed, which include topological connection, latency setting, network broadcast algorithm and so on. As for simulation-consensus, diverse consensus protocols can be implemented. In terms of simulation-data, common blockchain data structures can be customized, such as transactions and blocks. In particular, we implement three blockchain network protocols based on BlockSim, including Bitcoin, Ethereum, and BlockP2P, and compare their performance.



(a) Transaction synchronization (b) Block synchronization

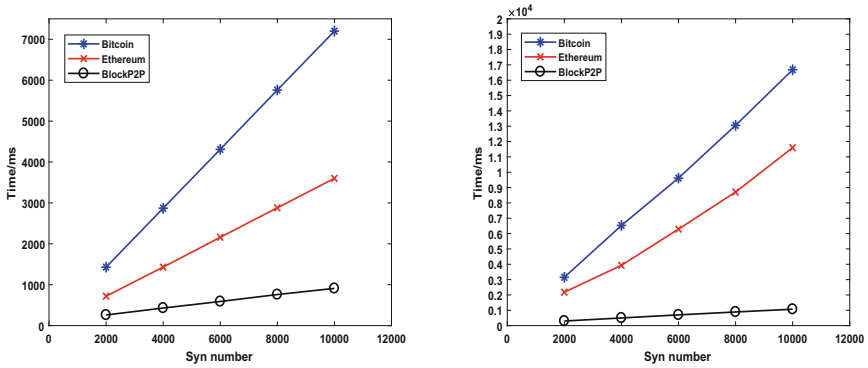
Fig. 6. Time taken to broadcast the transaction/block

Evaluation Metrics. In order to observe the network optimization effect of BlockP2P compared to Bitcoin and Ethereum, we establish the evaluation metrics about blockchain network performance from three aspects: (1) static performance with the number of nodes fixed (general performance); (2) dynamic performance with the number of nodes changing (network scalability); (3) stable performance with the number of nodes joining and leaving (network stability). First, general performance means how much the broadcast time of transactions and blocks consumes with the fixed nodes, when different synchronization ratios are reached. Here we consider the blockchain network scale in reality to find a reasonable maximum network size. In the real blockchain network, Bitcoin has 10,561 nodes [1], and Ethereum currently has 8,485 nodes [3]. Therefore, the maximum blockchain network size is fixed as 10,000 to fit in with the actual blockchain network size. Second, network scalability means how the broadcast time changes when the number of network nodes increase from 2,000 to 10,000, with the fixed synchronization ratio. In the case of increasing network size, different speed of the synchronization time change can reflect the blockchain system scalability. In the end, we evaluate the robustness of BlockP2P, by investigating the fluctuation of the time when lots of nodes join or leave in one data synchronization process for the evaluation of network stability. In more detail, we evaluate the network stability by measuring broadcast time of blocks fluctuates when 100 nodes join in or leave from the network every 2 s.

5.2 Experimental Results

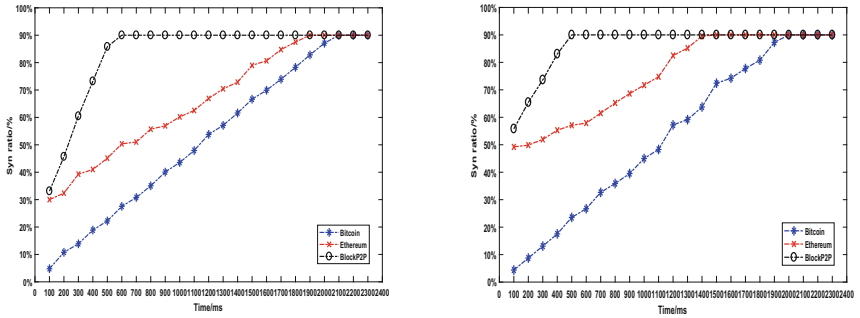
General Performance. We first measure the time used to broadcast the data to different synchronization ratios of nodes. With the total number of nodes fixed as 10,000, two group of experiments for transactions and blocks are conducted respectively. As shown in Fig. 6, the experimental results demonstrate that the broadcast time of BlockP2P is less than Bitcoin and Ethereum both in terms of transaction and block synchronization at different synchronization ratios. To be specific, when the block synchronization ratio reaches 90%, Bitcoin takes 15,000 ms, while BlockP2P only takes 1,100 ms, which can reduce the network broadcast latency by 90%. At the same time, network synchronization time of BlockP2P changes very little at different network synchronization ratios compared to Bitcoin and Ethereum. When the block synchronization ratio changes from 20% to 90%, synchronization time change for Bitcoin takes 11,000 ms, while BlockP2P only takes 850 ms. To sum up, BlockP2P can promote the network performance apparently.

Network Scalability. Now we fix the synchronization ratio and increase the number of nodes in each blockchain simulator, to evaluate the network scalability of BlockP2P. As shown in Fig. 7, as the number of the network node increases, the data synchronization time required also gradually increases, both in terms of transaction and block synchronization. However, as for the same size of network, the synchronization time taken by BlockP2P is smaller than Bitcoin and Ethereum. Specifically, when the number of nodes is 10,000, it takes



(a) Transaction synchronization (b) Block synchronization

Fig. 7. Influence of the number of nodes on the broadcast time



(a) Nodes join (b) Nodes leave

Fig. 8. Influence of the join or leave of nodes on the broadcast time

7,200 ms for a transaction to propagate to 90% of the total nodes. By contrast, it only takes 920 ms for BlockP2P. The similar phenomenon takes place in terms of block synchronization. In the meanwhile, network synchronization time of BlockP2P changes very little at different network synchronization numbers compared to Bitcoin and Ethereum. When the network synchronization number changes from 2,000 to 10,000, synchronization time change for Bitcoin takes 14,000 ms, while BlockP2P only takes 720 ms. As a result, we can conclude that BlockP2P protocol can provide a higher system scalability.

Network Stability. In this section, we try to verify if BlockP2P can maintain stability of latency when the number of nodes changes dynamically. With total number of nodes initialized as 10,000 and the synchronization ratio fixed as 90%, in one process of network synchronization, we increase or decrease 100 nodes every 100 ms to observe the time and fluctuation of the network. From Fig. 8, we can find that it only takes 400 ms, and the synchronization time fluctuates

slightly. Compared with the original network scale, network synchronization time has basically not changed, and fluctuation of time is weak. While Bitcoin reaches the final synchronization ratio of 90%, BlockP2P takes 2,200 ms. Compared with the original network scale, the synchronization time increases obviously, so the fluctuations are dramatic. Therefore, it shows that BlockP2P can maintain better network stability than Bitcoin and Ethereum, when large number of nodes leave from or join in the network.

6 Conclusion

The performance is the major challenge that influences the development of blockchain technology. Most of the researches draw their attention on the optimization of consensus layer or data layer, while lacking consideration of the underlying P2P network optimization. To complement these limitations, we take steps towards the network layer. We first comprehensively analyze the influential factors of the entire blockchain network propagation, from the connection establishment phase and the data transmission phase, respectively. Based on our key findings, we then carry out a novel network protocol *BlockP2P* to optimize the topology. To verify the feasibility and efficiency of BlockP2P, we design and implement a unified blockchain network simulator *BlockSim* to evaluate the performance in terms of latency. The experimental results demonstrate that in comparison to existing Bitcoin and Ethereum, BlockP2P can provide lower network latency for data broadcast, and maintain network scalability and stability.

References

1. Bitnodes. https://bitnodes.earn.com/nodes/
2. Ethereum. https://www.ethereum.org/
3. Ethernodes. https://www.ethernodes.org/network/
4. NKN. https://www.nkn.org/
5. Bhabak, P., Harutyunyan, H., Kropf, P.: Efficient broadcasting algorithm in Harary-like networks. In: Proceedings of the 46th International Conference on Parallel Processing Workshops, pp. 162–170. IEEE (2017)
6. Croman, K., et al.: On scaling decentralized blockchains. In: Clark, J., Meiklejohn, S., Ryan, P.Y.A., Wallach, D., Brenner, M., Rohloff, K. (eds.) FC 2016. LNCS, vol. 9604, pp. 106–125. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53357-4_8
7. Datta, S., Giannella, C., Kargupta, H.: K-means clustering over a large, dynamic network. In: Proceedings of the 2006 SIAM International Conference on Data Mining, pp. 153–164. SIAM (2006)
8. Decker, C., Wattenhofer, R.: Information propagation in the Bitcoin network. In: Proceedings of the IEEE Thirteenth International Conference on Peer-to-Peer Computing, pp. 1–10. IEEE (2013)
9. Delgado-Segura, S., Pérez-Solà, C., Herrera-Joancomartí, J., Navarro-Arribas, G., Borrell, J.: Cryptocurrency networks: a new P2P paradigm. Mobile Inf. Syst. **2018**, 16 (2018)

10. Donet Donet, J.A., Pérez-Solà, C., Herrera-Joancomartí, J.: The Bitcoin P2P network. In: Böhme, R., Brenner, M., Moore, T., Smith, M. (eds.) FC 2014. LNCS, vol. 8438, pp. 87–102. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44774-1_7
11. Eyal, I., Gencer, A.E., Sirer, E.G., Renesse, R.V.: Bitcoin-NG: a scalable blockchain protocol. In: Proceedings of the USENIX Conference on Networked Systems Design and Implementation, pp. 45–59. USENIX (2016)
12. Fadhil, M., Owen, G., Adda, M.: Proximity awareness approach to enhance propagation delay on the Bitcoin Peer-to-Peer network. In: Proceedings of the IEEE 37th International Conference on Distributed Computing Systems, pp. 2411–2416. IEEE (2017)
13. Georgiou, C., Gilbert, S., Guerraoui, R., Kowalski, D.R.: Asynchronous gossip. J. ACM (JACM) **60**(2), 11 (2013)
14. Karp, R., Schindelhauer, C., Shenker, S., Vocking, B.: Randomized rumor spreading. In: Proceedings 41st Annual Symposium on Foundations of Computer Science, pp. 565–574. IEEE (2000)
15. Kim, S.K., Ma, Z., Murali, S., Mason, J., Miller, A., Bailey, M.: Measuring Ethereum network peers. In: Proceedings of the Internet Measurement Conference, pp. 91–104. ACM (2018)
16. Li, X., Jiang, P., Chen, T., Luo, X., Wen, Q.: A survey on the security of blockchain systems. Future Generation Computer Systems (2017)
17. Luu, L., Narayanan, V., Zheng, C., Baweja, K., Gilbert, S., Saxena, P.: A secure sharding protocol for open blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 17–30. ACM (2016)
18. Montresor, A., Jelasity, M.: PeerSim: a scalable P2P simulator. In: Proceedings of the IEEE 9th International Conference on Peer-to-peer Computing, pp. 99–100. IEEE (2009)
19. Nagpal, R., Shrobe, H., Bachrach, J.: Organizing a global coordinate system from local information on an Ad Hoc sensor network. In: Zhao, F., Guibas, L. (eds.) IPSN 2003. LNCS, vol. 2634, pp. 333–348. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36978-3_22
20. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System (2008). https://bitcoin.org/bitcoin.pdf/
21. Nayak, K., Kumar, S., Miller, A., Shi, E.: Stubborn mining: generalizing selfish mining and combining with an eclipse attack. In: Proceedings of the 2016 IEEE European Symposium on Security and Privacy, pp. 305–320. IEEE (2016)
22. Neudecker, T., Andelfinger, P., Hartenstein, H.: Timing analysis for inferring the topology of the Bitcoin Peer-to-Peer network. In: Proceedings of the Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress. IEEE (2016)
23. Papadis, N., Borst, S., Walid, A., Grissa, M., Tassiulas, L.: Stochastic models and wide-area network measurements for blockchain design and analysis. In: Proceedings of the IEEE Conference on Computer Communications, pp. 2546–2554. IEEE (2018)
24. Sourav, S., Robinson, P., Gilbert, S.: Slow links, fast links, and the cost of gossip. In: Proceedings of the IEEE International Conference on Distributed Computing Systems, pp. 786–796. IEEE (2018)
25. Xu, Z., Han, S., Chen, L.: CUB, a consensus unit-based storage scheme for blockchain system. In: Proceedings of the 2018 IEEE 34th International Conference on Data Engineering (ICDE), pp. 173–184. IEEE (2018)