High-Bandwidth Node Selection in Compact Block Relay

Shinnosuke Masuda[1], Tsuyoshi Hasegawa[1], Taishi Nakai[1], Akira Sakurai[2,1], and Kazuyuki Shudo[1]

[1]Kyoto University
[2]Tokyo Institute of Technology

***Abstract*—Blockchain has a scalability issue of low throughput. Improving throughput is in a trade-off relationship with an increase in the fork rate, which is an indicator of security. It is known that reducing the block propagation time is effective in solving this trade-off. In fact, a protocol called Compact Block Relay has been proposed and implemented for Bitcoin to reduce the block propagation time. In this study, we propose a method to select high bandwidth node in Compact Block Relay in order to reduce block propagation time. The result of the simulation shows that our proposed method improves the average block propagation time by 7.3% compared to what is implemented in Bitcoin Core. Furthermore, the proposed method is more resistant to attacks such as Eclipse Attack and TendrilStaller compared to existing Bitcoin Core implementations.**

***Index Terms*—Blockchain, Eclipse Attack, TendrilStaller, Compact Block Relay, peer-to-peer**

1. Introduction

Since Bitcoin [1] was proposed by Satoshi Nakamoto in 2008, it has been applied in various fields due to its high tamper resistance and decentralization. However, Blockchain faces the challenge of low throughput. Here, throughput refers to the number of transactions that can be processed per second. The Bitcoin system processes a maximum of approximately 27 transactions per second [2], whereas systems like Paypal handle around 600 transactions per second [3], and Visa processes around 1,700 transactions per second [4]. There are solutions such as reducing the block generation interval or increasing the block size to improve throughput, but both solutions increase the fork rate, leading to a deterioration in security [5]. To improve throughput while keeping the fork rate low, it is effective to reduce the block propagation time [6], [7]. Compact Block Relay(CBR) [8] is a protocol that reduces bandwidth consumption and shortens propagation time in block propagation. Nodes using CBR send lightweight blocks containing transaction identification information and headers, rather than full blocks containing transaction bodies and headers. CBR has two modes: a high-bandwidth mode that consumes a large amount of bandwidth but shortens propagation time, and a low-bandwidth mode that consumes less bandwidth but has longer propagation time. In Bitcoin Core, typical nodes set three nodes as high-bandwidth mode nodes.

This study aims to improve the inefficiency of high-bandwidth node selection in CBR and shorten block propagation time. The purpose of this study is to reduce block propagation time through the improvement of CBR.

2. Related Work

There are several studies aimed at reducing the block propagation time. For example, there are methods for selecting neighboring nodes based on their propagation abilities [9], [10], prioritizing connections with nodes that propagate quickly. However, while these methods can be somewhat effective, there is a risk of network partitioning attacks, such as Eclipse Attack [11], as nodes autonomously change neighboring nodes based on certain criteria. Another example is the region-based neighboring node selection method proposed by Matsuura et al [12]. This method prioritizes connections with nodes geographically close based on their IP addresses. However, using a third-party API to determine regions from IP addresses raises concerns that contradict the principles of Blockchain. These studies involve methods that change their own neighboring nodes, whereas our method differs in that it does not change its own neighboring nodes, making it less susceptible to Eclipse Attack [11]. A relay network reduce block propagation time by providing a network that speeds up block propagation [13], [14].

3. Compact Block Relay

3.1. Compact Block Relay Structure

CBR was proposed by BIP-152 and implemented in Bitcoin Core. Nodes using CBR send compact blocks (CB) instead of legacy blocks when transmitting a new block. CBs contain only transaction IDs and headers, making them lighter than legacy blocks. Nodes receiving CBs need reconstruct the block if they have necessary transactions in their mempool. If not, they request and receive the missing transactions from the node that sent the CB.

CBR has two modes: high bandwidth mode and low bandwidth mode, as shown in Figures 1 and 2. Nodes receiving CBs need to pre-set their neighboring nodes as

Proc. 20th IEEE Int'l Conf. on Green Computing and Communications
(IEEE GreenCom 2024), August 2024

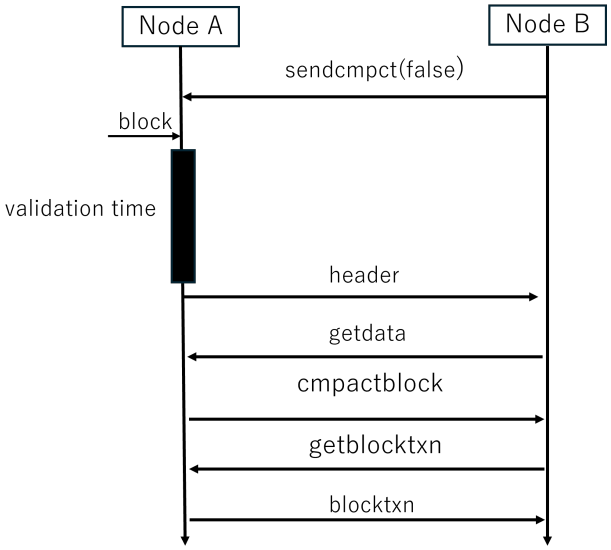

Figure 1. Low bandwidth mode.

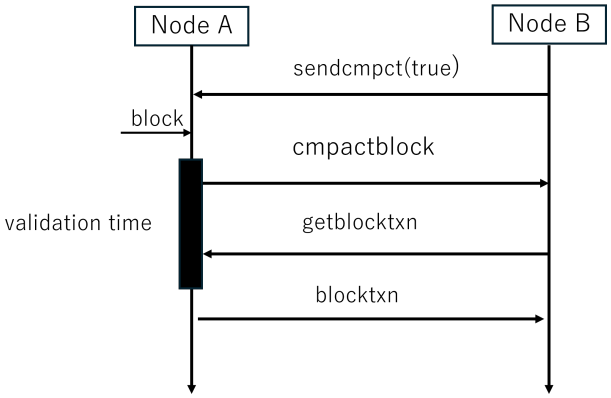

Figure 2. High bandwidth mode.

either low bandwidth mode or high bandwidth mode before receiving the CB, using the `sendcmpct` message. Sending the `sendcmpct` message with False makes the node behave in low bandwidth mode, while sending it with True makes the node behave in high bandwidth mode. The major difference between the two modes is that in low bandwidth mode, nodes verify the received block before notifying the compact block via the header, while in high bandwidth mode, nodes directly send the compact block without block verification. Therefore, high bandwidth mode nodes propagate blocks faster because they skip the verification time and block holding confirmation by the header. However, without block holding confirmation through headers, bandwidth is wasted if nodes already possess compact blocks. Bitcoin Core has a limit of up to three nodes that can be set as high bandwidth mode to reduce wasted bandwidth consumption.

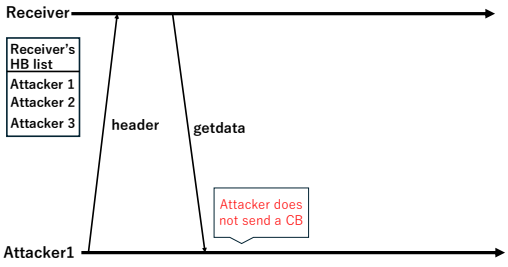

Figure 3. TendrilStaller

3.2. High Bandwidth Node Selection in Bitcoin Core

In Bitcoin Core, each node maintains a list called the HB list, which includes nodes designated as high bandwidth mode nodes and time each of those nodes last sent a block. If a low bandwidth node sends a `header` message earlier than the high bandwidth mode nodes send CBs, and the receiving node is able to construct that block, then the receiving node removes the entry in the HB list with the oldest block-sent-time and adds the low bandwidth mode node to the HB list with the time it last sent the block.

The problems with the high-bandwidth node selection implemented in Bitcoin Core are as follows:

1) Inefficient high bandwidth node selection
2) Vulnerable to TendrilStaller [15]

The first problem arises from the fact that low bandwidth node with fast block propagation abilities are at a disadvantage in terms of verification time, making it difficult for them to enter the HB list. Additionally, only one node can be evaluated per block, making it difficult to search for nodes with better propagation abilities.

The second problem arises because high bandwidth node selection is based on solely on the propagation abilities of the block. TendrilStaller is a block delay attack in which the HB list is filled with attacker nodes to make it difficult for the victim node to receive a CB from other neighbor nodes. As shown in Figure 3, an attacker performing TendrilStaller prepares three nodes and needs to infiltrate the neighboring nodes of target node. After the three nodes have infiltrated the neighboring nodes of the target node, the attack begins. Even if the three attacking nodes were set as low bandwidth nodes, they behave as high bandwidth mode nodes. They continue to directly send CBs to the HB list of the target node until all the attacking nodes can infiltrate it. After occupying the HB list of target node, the attacking node sends `header` message without verification and delays block propagation by not sending CBs even when `getdata` message arrive. Based on the above, there is a motivation not only to avoid having three or more attacker nodes as neighboring nodes, but also to avoid adding them to the HB list even if three of them do become neighboring nodes.

4. Proposed Method

In the proposed method, in order to address the issues of Bitcoin Core, it searches for nodes with fast block propagation abilities among its neighboring nodes and configures those nodes as high bandwidth mode nodes. Specifically, we calculate a score based on penalties, and set nodes with low scores as high bandwidth mode nodes. Light penalties are assigned to nodes with fast block propagation abilities, while heavy penalties are given to nodes with slow block propagation abilities.

4.1. Impose Penalties

This section describes how to calculate the penalty used to update the score. The proposed method not only adopts nodes with fast block propagation abilities as high bandwidth mode node, but also considers tolerance to Tendrill-Staller by penalizing high bandwidth mode nodes per block. The penalty calculation is performed for each block against high bandwidth mode nodes. For a node v that has sent a CB, if it can construct a block through v's CB, the penalty for v is set to 0. For a node k that has sent a CB after block construction or had sent a CB before v but failed to construct a block through that CB, the penalty for k is the order of arrival of k's CBs, excluding v. The penalty calculation allows for the evaluation of three nodes at once, making it easier to search for nodes with better propagation abilities.

4.2. Update Score based on Penalties

In this section, we explain the method for calculating scores based on penalties. Each node i holds a Score for each of its neighboring nodes k, and this score is updated as penalties are imposed on k. The score is used when sending `sendcmpct` message, and the three nodes with the lowest scores are set as high bandwidth mode nodes. We propose two methods for updating scores based on penalties.

The first is to replace the score with the latest penalty, where the score of a node i's neighboring node k can be expressed as follows:

$$Score_k^i \leftarrow Penalty_k^i \tag{1}$$

The second method accumulates penalties for previous scores, where the score of a node i's neighboring node k can be expressed as follows:

$$Score_k^i \leftarrow Score_k^i + Penalty_k^i \tag{2}$$

In the update method of (1), by selecting nodes with low recent penalties, it is possible to efficiently explore nodes with better block propagation abilities. Additionally, nodes with better propagation abilities are more likely to be selected consecutively as high bandwidth mode nodes. In the update method of (2), we explore nodes with fast propagation and can set such nodes as high bandwidth mode. However, even nodes with better propagation abilities will eventually be excluded from the HB list due to the accumulation of penalties. This is intended to prevent attackers from occupying the HB list, in addition to efficient selection of high bandwidth mode nodes. In addition, when considering practical applications, since the network is dynamic, it is necessary to assign the median score of the adjacent nodes to the score of a node that has newly joined its adjacent nodes, and to set the cumulative range for penalties to the recent several blocks. Hereafter, we will refer to the method using the score update of (1) as the proposed method (last one block), and the method using the score update of (2) as the proposed method (sum).

TABLE 1. PARAMETER SETTINGS

Number of nodes	1,000
Number of blocks	100,000
Percentage of nodes using CBR	100%
Compact block size	18KB [19]
Block verification time	174ms [15]
Bandwidth	Nagayama et al. [17]
Delay between nodes	Nagayama et al. [17]

5. Experiment

5.1. Simulation Settings

SimBlock [16] was used for the experiments. SimBlock can simulate the Bitcoin network in 2019. The bandwidth and delay between nodes used in the experiment were based on the values by Nagayama et al. [17], and the block verification time was based on the values by Ke et al. [15]. Furthermore, since SimBlock was implemented only for the low bandwidth mode, we also implemented and experimented with high bandwidth mode. According to Sakurai et al. [18], there is a positive correlation between the average block propagation time and the fork rate. Therefore, we measured the average block propagation time for the proposed method (last one block), the proposed method (sum) and the implementation in Bitcoin Core. We also additionally measured the 90th percentile and median values. Table1 shows the parameter settings used in the experiment.

5.2. Result

The distribution of average block propagation times for Bitcoin Core implementation and proposed method (last one block) is shown in Figure 4. The average and median block propagation times, 90th percentile and improvement rate for Bitcoin Core implementation and proposed method (last one block) are also shown in Table 2.

As shown Table 2, the proposed method (last one block) improves the average by 7.3%, the median by 7.2%, and even 90th percentile by 5.4% compared to Bitcoin Core implementation. We consider that this is because the proposed method (last one block) can more efficiently search for nodes with fast block propagation abilities than Bitcoin

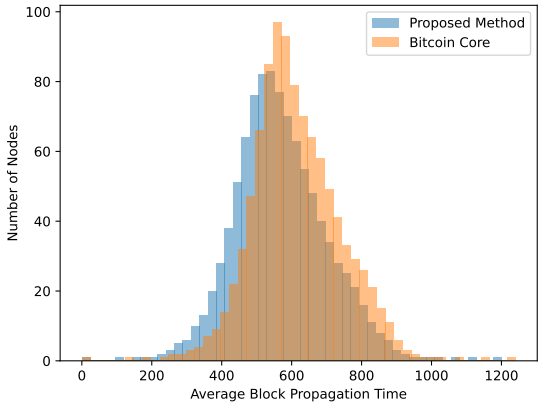

Figure 4. Distribution of average block propagation time for the proposed method (last one block).

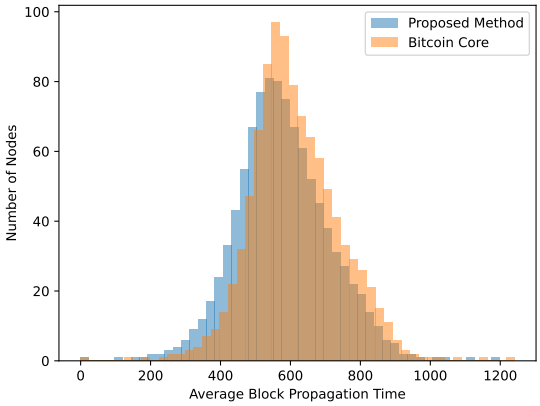

Figure 5. Distribution of average block propagation time for the proposed method (sum).

TABLE 2. COMPARISON OF REPRESENTATIVE VALUES OF BLOCK PROPAGATION TIME IN BITCOIN CORE AND PROPOSED METHOD (LAST ONE BLOCK).

Method	Average	Median	90th percentile
Bitcoin Core	613 ms	599 ms	784 ms
Proposed Method (last one block)	568 ms	556 ms	742 ms
Improvement Rate	7.3%	7.2%	5.4%

TABLE 3. COMPARISON OF REPRESENTATIVE VALUES OF BLOCK PROPAGATION TIME IN BITCOIN CORE AND PROPOSED METHOD (SUM).

Method	Average	Median	90th percentile
Bitcoin Core	613 ms	599 ms	784 ms
Proposed Method (sum)	578 ms	570 ms	753 ms
Improvement Rate	5.7%	4.8%	4.0%

Core implementation, and also because such nodes are more likely to be selected as high bandwidth mode nodes.

The distribution of average block propagation times for Bitcoin Core implementation and proposed method (sum) is shown Figure 5. The average, median, 90th percentile block propagation time and improvement rate for Bitcoin Core implementation and proposed method (sum) are also shown in Table 3.

As shown Table 3, the proposed method (sum) improves the average by 5.7%, the median by 4.8%, and even 90th percentile by 4.0% compared to Bitcoin Core implementation. We consider that the block propagation time could be shortened because, similar to the proposed method (last one block), nodes with fast block propagation abilities could be efficiently searched. However, due to the accumulation of penalties, even nodes with better propagation abilities will eventually be excluded from the HB list. Therefore, in terms of average, median, 90th percentile, the improvement rate was not better than the proposed method (last one block).

5.3. Impact on Security

As Sakurai et al [18] have shown, there is a positive correlation between block propagation time and fork rate. Therefore, it can be said that our method has reduced the fork compared to existing Bitcoin Core and improved security. Our two proposed methods do not change neighboring nodes, so there is less risk of Eclipse Attack. Additionally, by not using timestamps as in Aoki and Shudo [9]'s method, our approach is resistant to attacks where attackers intentionally tamper with timestamps. Furthermore, it is considered that the proposed method (sum) is more resistant to TendrilStaller than Bitcoin Core. This is because even nodes with better propagation abilities are eventually removed from the HB list due to the accumulation of penalties, reducing the likelihood of attackers occupying the HB list. In contrast, the proposed method (last one block) is more likely to continuously select nodes with low recent penalties, leading to a higher rate of improvement in average block propagation time compared to the proposed method (sum). However, this also increases the likelihood of attackers being continuously selected, thereby reducing resilience against TendrilStaller.

6. Conclusion

In this paper, we proposed methods that can reduce the block propagation time over the entire network by efficiently selecting high bandwidth mode nodes without changing one's own neighboring nodes. We also conducted experiments using SimBlock to compare the average, median, and 90th percentile of block propagation time between the Bitcoin Core implementation and the two proposed methods. As a result, the two proposed methods showed improvement over all metrics compared to the Bitcoin Core implementation. Furthermore, we considered the trade-off between fork rate and TendrilStaller in the proposed method (last one block) and the propsed method (sum). As a future task,

we plan to quantitatively evaluate the resistance of Bitcoin Core and the two proposed methods to TendrilStaller, and confirm the trade-off between the two methods.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number JP21H04872 and JP24H00691.

References

[1] S.Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*. https://bitcoin.org/bitcoin.pdf, 2008.

[2] E. Georgiadis, *How Many Transactions per Second can Bitcoin Really Handle ? theoretically*. IACR Cryptol. ePrint Arch., 2019, p. 416, 2019.

[3] "Paypal,inc—paypal q3-21 investor update," 2021. [Online]. Available: https://investor.pypl.com/

[4] Visa, accessed: 2024-06-22. [Online]. Available: https://usa.visa.com/

[5] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 3–16. [Online]. Available: https://doi.org/10.1145/2976749.2978341

[6] T. Nakai, A. Sakurai, S. Hironaka, and K. Shudo, "The blockchain trilemma described by a formula," in *2023 IEEE International Conference on Blockchain (Blockchain)*, 2023, pp. 41–46.

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

[8] bip 0152.mediawik. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki

[9] A.Yusuke and K.Shudo, *Proximity Neighbor Selection in Blockchain Networks*. IEEE International Conference on Blockchain (Blockchain), 2019.

[10] K. Wang and H. S. Kim, "Fastchain: Scaling blockchain system with informed neighbor selection," in *2019 IEEE International Conference on Blockchain (Blockchain)*, 2019, pp. 376–383.

[11] E.Heilman, A.Kendler, A.Zohar, and S.Goldberg, *"Eclipse Attacks on Bitcoin's Peer-to-Peer Network*. Proc. 24th USENIX Security Symposium (USENIX Security'15), Washington, D.C., USENIX Association, pp. 129–144, 2015.

[12] H. Matsuura, Y. Goto, and H. Sao, "Region-based neighbor selection in blockchain networks," in *2021 IEEE International Conference on Blockchain (Blockchain)*, 2021, pp. 21–28.

[13] FIEBRE, "https://bitcoinfibre.org/."

[14] U.Klarman, A. S.S.Basu, and E.G.Sirer, "bloxroute:a scalable trustless blockchain distribution network whitepaper," 2018.

[15] M. Walck, K. Wang, and H. S. Kim, *Tendrilstaller: Block Delay Attack in Bitcoin*. Proc. 3rd IEEE International Conference on Blockchain (IEEE Blockchain 2019), pp. 1–9, 2019.

[16] Y.Aoki, K.Otsuki, R. T.Kaneko, and K.Shudo, *SimBlock: A Blockchain Network Simulator*. IEEE, 2019.

[17] R.Nagayama, K.Shudo, and R.Banno, *Simulation of the Bitcoin Network Considering Compact Block Relay and Internet Improvements," IEICE Technical Report*. ; IEICE Tech. Rep., vol.119, no.460,pp.179–183, 2020.

[18] A. Sakurai and K. Shudo, "Impact of the hash rate on the theoretical fork rate of blockchain," in *2023 IEEE International Conference on Consumer Electronics (ICCE)*, 2023, pp. 1–4.

[19] A. P. Ozisik, G. Andresen, G. D. Bissias, A. Houmansadr, and B. N. Levine, "A secure, efficient, and transparent network architecture for bitcoin," 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:1218638