Verification of Applying Plumtree Algorithm to Blockchain Networks

Yusuke KITAGAWA[†a)], ***Student Member***, **Kazuyuki SHUDO**[††], ***Member***, **Osamu MIZUNO**[†], ***Senior Member***, ***and*** **Ryohei BANNO**[†,††], ***Member***

SUMMARY Blockchain is gaining attention as a technology to support cryptocurrency. It is a system that prevents tampering by sharing a ledger for recording transactions among multiple computers on a network. However, one of the problems in using blockchain is the consumption of communication resources. It is caused by that the information received by a node can be delivered to the same node more than once through different neighbors. In this paper, we propose a method to reduce duplicate messages in a blockchain network by applying an algorithm called Plumtree. The proposed method enables forming loosely fixed propagation paths on a random topology. Through simulation experiments, we compared the proposed method with existing methods and confirmed that the number of messages can be reduced by more than 75 % when block propagation is simulated up to four blocks.

key words: *Blockchain*, *SimBlock*, *Plumtree, Peer-to-Peer network*

1. Introduction

The distributed ledger technology represented by blockchain is a system in which participants share records and keep them in a state that is difficult to tamper with. It is known for its applications in virtual currencies and financial services such as Bitcoin [1]. The Ministry of Economy, Trade and Industry (METI) estimates the domestic market size for blockchain to be approximately 67 trillion yen [2]. In a typical blockchain, such as Bitcoin, a large number of nodes randomly interconnect to form a Peer-to-Peer (P2P) network. Each node sends information to its neighbors in order to spread the information to all nodes. The node that receives the information also sends the information to another node, and so on. As a result, information that has already been received may be sent multiple times through different nodes [3], resulting in excessive consumption of communication resources. In this study, we aim to solve the problem of communication resource consumption in the Bitcoin network by using the Plumtree algorithm [4], which is known as an efficient broadcast method. The structure of this paper is as follows: Section 2 describes the blockchain, Section 3 gives the details of the proposed method, in section 4, we describe the evaluation experiments and their results, and finally, in section 5, we summarize and discuss the future work of this research.

2. Blockchain

In this section, we describe the blockchain techniques. Blockchain is an underlying technology for cryptocurrencies like Bitcoin, which is implemented based on a paper by Satoshi Nakamoto [1]. Blockchain enables distributed consensus building among participants in an open network without the need for trusting a specific third party and enables highly transparent transactions by making all history traceable. Blockchain is expected to be used in a variety of fields, as it is difficult to falsify data and has zero downtime.

2.1 Types of Blockchain

There are three types of blockchain networks as shown below.

- Public blockchain network
  A public blockchain network has no administrator, and anyone with an Internet connection can become a node and join the network.
- Private blockchain network
  Unlike a public blockchain network, a private blockchain network has a single administrator and requires permission from the administrator to join the network as a node.
- Consortium-type blockchain network
  A consortium blockchain network is a blockchain in which a specific number of administrators generate and approve blocks, providing both the decentralized nature of a public blockchain and the speed of a private blockchain. However, it has the disadvantage that it requires trust in a specific administrator, and the transparency of transactions inherent in the blockchain is lost.

In this paper, we focus on Bitcoin, the public type.

2.2 Broadcast

The Bitcoin block sharing method is explained using the sequence diagram of Bitcoin messages in Figure 1. The Bitcoin node first sends an inv message to the nodes when it receives a block. The node that receives the message checks if there are any missing blocks. If there is a missing block, it sends a getdata message to the node that sent the inv message and asks it to send the missing block. This mechanism eliminates the need to constantly send blocks to the entire network, thus reducing redundant transmissions

[†]The author is with Kogakuin University, Tokyo, Japan.
[††]The author is with Tokyo Institute of Technology, Tokyo, Japan.
a) E-mail: cm21016@ns.kogakuin.ac.jp

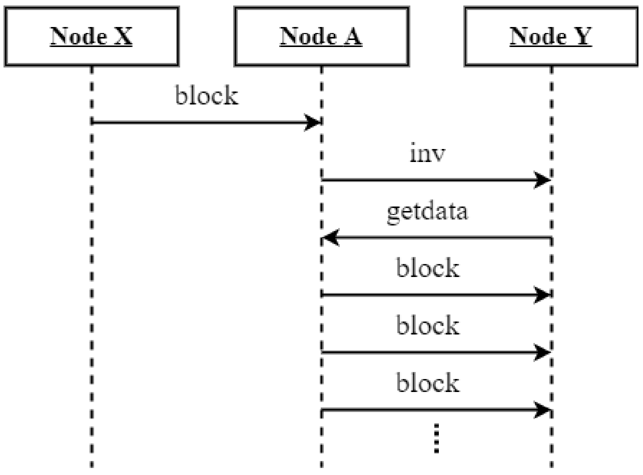

Fig. 1 Sequence diagram of messages by the existing method

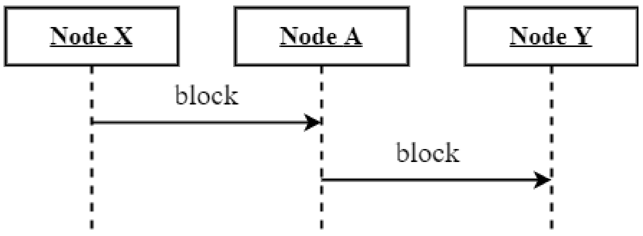

Fig. 2 Sequence diagram of messages by the proposed method

3. Proposed method

The problem with using blockchain is the consumption of communication resources. received messages are propagated to other nodes to spread the messages. Therefore, there is a problem that a message that has already been received is delivered multiple times via different nodes. In Bitcoin, the aforementioned mechanism reduces the unnecessary transfer of blocks, but there is a problem that inv messages consume communication resources. In a blockchain network, received messages are propagated to other nodes to spread the messages. Therefore, there is a problem that a message that has already been received is delivered multiple times via different nodes. There is also a concern that the propagation delay will increase due to the need for confirmation by the inv message. In this study, we solve the problem of communication resource consumption by using the function of reducing redundant message transmission in Plumtree based on the Bitcoin network.

3.1 Plumtree

Plumtree [4] is a broadcast method that combines Tree-based and Gossipbased approaches to reduce redundancy while maintaining high message reliability. In this method, each node transmits and improves efficiency by forwarding messages only in a Tree-based manner. It also uses gossip-based links between nodes to properly handle inter-node failures. In Tree-based broadcast, a tree consisting of all participant nodes is constructed and messages are transmitted only along the tree, which reduces the redundancy of messages. However, if the number of nodes increases or decreases, or if a network failure occurs, the tree becomes incomplete, and a message may not reach all nodes. In Gossip-based broadcast, when a node attempts to broadcast a message, the node randomly selects t nodes to send the message. The node that receives the message for the first time repeats this process to send the message, which not only provides high scalability

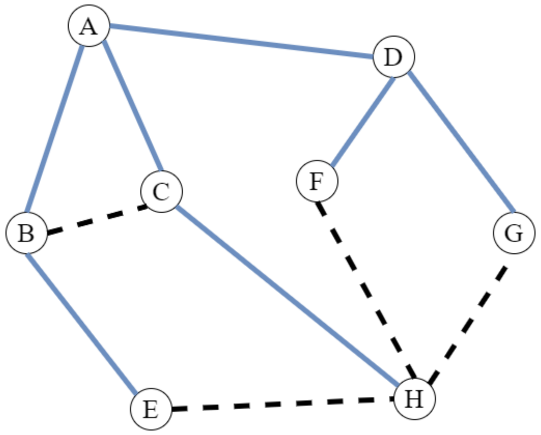

Fig. 3 Example of Spanning Tree

but also makes the system more resistant to network inadequacies and node failures. However, redundancy tends to be high.

3.1.1 Tree-based and Gossip-based

Plumtree is designed around the Gossip-based approach to message broadcasting, which is based on the idea that all nodes contribute equally to the propagation of a message. To achieve this idea, when a node broadcasts a message, it randomly chooses a node to send it to. The node that receives the message for the first time repeats this process. This property allows the system to respond flexibly to network failures and increases or decreases in the number of nodes. In Gossip-based, the following approaches are used.

- Eager push
  As soon as a message is received, it is sent to the neighboring nodes. Lazy push approach
- Lazy push
  When a node receives a message, it sends the message identifier to the neighboring nodes. If the node has not received the message, it makes a pull request.
- Pull
  Periodically, a node queries its neighboring nodes for information on recently received messages. Plumtree uses a combination of Eager push ’ s mechanism for sending a small number of messages and Lazy push ’ s mechanism for assisting transmission.

3.1.2 How Plumtree Works

Initially, the block is broadcasted using the same mechanism as Bitcoin, and a spanning tree is constructed using the Plumtree algorithm. In subsequent broadcasts, the constructed tree is used. The exchange of inv and getdata messages is omitted, and blocks are sent directly to the child nodes of the tree. The Plumtree algorithm is used to repair the tree when the number of nodes increases or decreases, or when the network fails.

3.2 Applying Plumtree in Bitcoin Network

In the proposed method, the first block is broadcasted using the same mechanism as Bitcoin. Upon receiving the block for the first time, each node stores the information of the sender node. Each node also stores the information of nodes to which it sends the block. These stored neighbor information from the spanning tree. The information that each node has is shown below.

- eagerPushPeers
  The set of adjacent nodes in the spanning tree
- lazyPushPeers
  The set of adjacent nodes other than eagerPushPeers in the spanning tree

For example, from Figure 2, the eagerPushPeers of node B include nodes A and E, and the lazyPushPeers include node C. The basic operation is that all nodes that have the information of eagerPushPeers send messages by eager push. The basic operation is to put all neighboring nodes into eagerPushPeers and send messages to the neighboring nodes by eager push. If a duplicate message is received, the nodes will be added to the lazyPushPeer. Once the first broadcast is complete, the spanning tree is formed. In a stable network, it is possible to broadcast only by eager push on the spanning tree. If node A intends to send a message to node C but node C has already left, it is unable to send the message to node H. In this case node H will not know if it has received new messages, so it will query its lazyPushPeers for new messages with Lazy push. If there is a new message, select one of the nodes E, F, or G and query the message with Lazy push. This Plumtree algorithm is used to construct the spanning tree. In subsequent broadcasts, the constructed tree is used. As shown in Figure 3, we omit the exchange of inv and getdata messages and send blocks directly to the child nodes of the tree. When the number of nodes changes or the network fails, the Lazy push and Pull are used for repairing the spanning tree.

4. Evaluation

We measured the number of messages and the number of hops by using SimBlock [5], which can simulate the information propagation of blockchain. This section describes the experimental environment and the result of the evaluation.

Table 1 SimBlock parameters

Parameter	Value
NUM_OF_NODES	8370
BLOCK_SIZE	803.565 Kbyte
CBR_USAGE_RATE	0
CHURN_NODE_RATE	0

Table 2 Comparison of the number of messages

Messages type	# of messages
inv (Proposed method)	144714
inv (Existing method)	578856
getdata (Proposed method)	144714
getdata (Existing method)	578856

4.1 Simblock

SimBlock is a simulator for a blockchain network consisting of many nodes on the Internet. SimBlock reproduces the network of an actual blockchain in use, so that users can examine the behavior of a blockchain as close to the real thing as possible. It allows us to change the behavior of the nodes that make up the blockchain network and verify how new techniques affect the blockchain network.

4.2 Implementation of proposed method

We implemented the proposed method by modifying SimBlock. When a node broadcasts its first block, it creates an array of child node sets using the information it received when it sent the block to its neighboring nodes. The array of child node sets is program to store the information of the node at the time when the block transmission was completed. The broadcast of the next block was set to be transmitted from the node that stored the information first, using the array of child node sets. For counting inv and getdata messages, we implemented a program to count the number of inv and getdata messages on each node and calculate the total number of messages received by all nodes.

4.3 Experimental configuration

The experimental parameters of SimBlock used in this experiment are shown in Table 1. The number of nodes and block size were set to 8370 and 803.565 Kbytes, respectively, based on Bitcoin chart information [6]. These are the values as of December 15, 2020. We did not use Compact Block Relay (CBR) and used default values for the rest of the parameters of SimBlock. Our experiments simulate block propagation up to four blocks. Each experiment was conducted 10 times and we calculated the average values.

4.4 Evaluation of the number of messages

We evaluated the number of messages when Plumtree is applied to a blockchain. Table 2 shows the number of messages in the Bitcoin blockchain for the proposed method and the

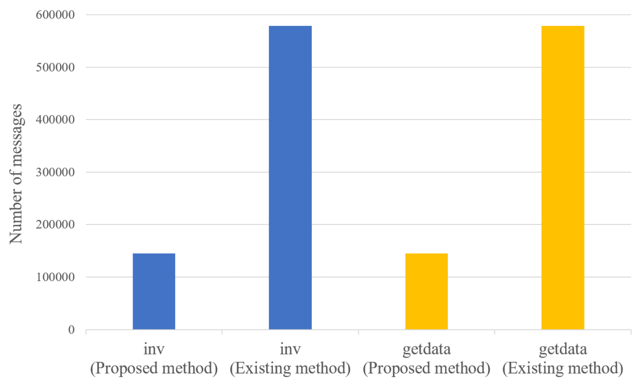

Fig. 4 Number of inv and getdata messages in Bitcoin

conventional method. Figure 4 shows the comparison of the number of messages between the proposed method and the conventional method for Bitcoin. The vertical axis is the total number of messages received by all the nodes, and the horizontal axis is the measurement item. 75% or more of the number of messages can be reduced by comparing the proposed method and the existing method. In the proposed method, inv and getdata messages are generated only at the first broadcast. As a result, the number of messages is reduced compared to existing methods.

4.5 Relationship between the number of hops and the delay time

We investigated the relationship between the number of hops and the delay time when Plumtree is applied to the blockchain. Figure 5 shows the relationship between the number of hops and the delay time of the proposed method. The vertical axis is the average delay per block, and the horizontal axis is the average number of hops. In the proposed method, the average delay increases as the average hop count increases. From the relationship between the number of hops and the delay, it can be seen that the Plumtree builds a route according to the node that sends the first block, so if subsequent blocks are sent by other nodes, they are transmitted using a route that is not the shortest. In addition, the larger the number of hops, the longer it takes for the block to reach the destination, and the propagation delay is expected to increase.

5. Conclusion

In this paper, we propose a message reduction method for blockchain networks and evaluate it by simulation. As a result, we found that the number of messages can be reduced by applying Plumtree compared to existing methods. In the future, we would like to study how to improve the construction of spanning trees in order to reduce the number of hops.

Acknowledgements
This work was supported in part by the JGC-S Scholarship Foundation. A part of this work was supported by JSPS KAKENHI Grant Number JP21H04872.

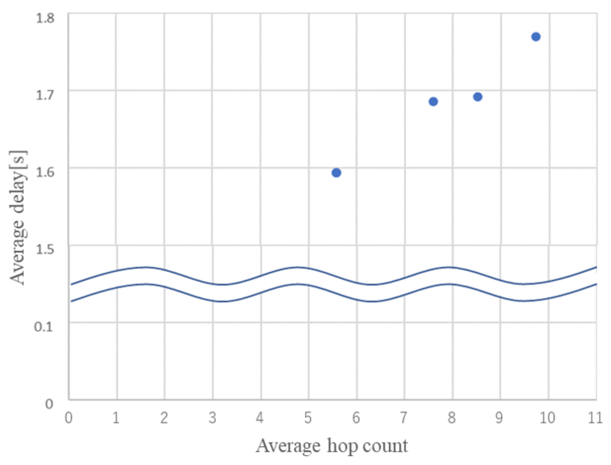

Fig. 5 Relationship between Bitcoin Hop Count and Latency

References

[1] Satoshi Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System", 2008.
[2] Ministry of Economy, Trade and Industry (METI), "FY2015 Survey Report on the Infrastructure Development for the Informatization and Servitization of Japan's Economy and Society (Market Research on Electronic Commerce)," (August 8, 2016)
[3] Andreas M. Antonopoulos by Takaya Imai, translated by Junichiro Hatogai, "Bitcoin and Blockchain: Technologies Supporting Cryptocurrencies", NTT Publishing, 2016.
[4] João Leitão, Jos Pereira, Luẽıs Rodrigues, "Epidemic Broadcast Trees", 26th IEEE International Symposium on Reliable Distributed Systems (SRDS 2007), pp.301-310, Oct. 2007.
[5] Yusuke Aoki, Kai Otsuki, Takeshi Kaneko, Ryohei Banno, Kazuyuki Shudo, "SimBlock: A Blockchain Network Simulator", Workshop on Cryptocurrencies and Blockchains for Distributed Systems (CryBlock, in conjunction with IEEE INFOCOM), pp. 325-329, April 2019.
[6] BitInfoCharts, https://bitinfocharts.com (Accessed December 15, 2020)