Speedup of Information Propagation on Blockchains

Jin Cao
515030910536
Shanghai Jiao Tong University
wanda9987@qq.com

Abstract

Blockchain is one of the most popular field due to the upsurge of Bitcoin, which is a digital currency that relies on a network of volunteers that collectively implement a replicated ledger and verify transactions. In this paper, I will introduce some key and basic concepts on blockchains and Bitcoin first. Then some basic methods of information propagation on blockchains will be mentioned. After that, I will analyze how to speed up the information propagation to avoid blockchain forks as much as possible. Besides, since high speed usually means high bandwidth, it is worth considering a trade-off between speed of bandwidth.

1. Introduction

After acknowledgment of Bitcoin as a digital cryptocurrency, blockchain has become a trending subject to not only the research community but also the industrial society because of the enormous application area. The blockchain can be defined as a decentralized immutable public ledger which is updated and secured in a distributed structure among the untrusted parties. This ledger consists of ordered blocks, which may be composed of transactions like in Bitcoin or smart contracts as in Ethereum.

Since the inception of Bitcoin in late 2008, Bitcoin has enjoyed a rapid growth, both in value and in the number of transactions. Its success is mostly due to innovative use of a peer-to-peer network to implement all aspects of a currencies life cycle, from creation to its transfer between users. It goes beyond the scope of cash, allowing truly global transactions, processed at the same speed as local ones[1]. It offers a public transaction history and it introduces many new and innovative uses such as smart properties, micropayments, contracts and escrow transactions for dispute mediation.

The main problem Bitcoin sets out to solve is the distributed tracking and validation of transactions. For this, the network needs to reach a consensus about the balances of the accounts it tracks and which transactions are valid. Bitcoin achieves this goal with guarantees which are best described as eventual consistency: the various replicas may be temporarily inconsistent, but will eventually be synchronized to reflect a common transaction history.

As transactions are validated against the replica states, any inconsistency introduces uncertainty about the validity of a given transaction. Furthermore, an inconsistency may jeopardize the security of the consensus itself. This may facilitate an attacker that attempts to rewrite transaction history.

In this work, I take Bitcoin as an typical example to analyze how information is propagated in the network. Sometimes, the synchronization mechanism fails to synchronize the information stored in the ledger with a non-negligible probability, which could causes a prolonged inconsistency that goes unnoticed by a large number of nodes and weakens defenses against attackers. So it is necessary to propose some changes to the current protocol to mitigate them. In other words, what we need is to speed up the information propagation on blockchains.

2. Information propagation

The Bitcoin network is a network of homogeneous nodes. There are no coordinating roles and each node keeps a complete replica of all the information needed to verify the validity of incoming transactions. Each node verifies information it receives from other nodes independently and there is only minimal trust between the nodes.

2.1. Network Topology

When a node joins the network it queries a number of DNS servers. These DNS servers are run by volunteers and return a random set of bootstrap nodes that are currently participating in the network. Once

connected, the joining node learns about other nodes by asking their neighbors for known addresses and listening for spontaneous advertisements of new addresses. There is no explicit way to leave the network. The addresses of nodes that left the network linger for several hours before the other nodes purge them from their known addresses set. Each node attempts to keep a minimum number of connections p to other nodes open at all times[4]. Should the number of open connections be below p the node will randomly select an address from its set of known addresses and attempt to establish a connection. On the other side, incoming connection are not closed if they result in the number of open connections to be above the pool size p. The total number of open connections is therefore likely to be higher for nodes that also accept incoming connections.
Partitions in the connection graph are not actively detected, and should they occur the partitions will continue operating independently. While this is certainly desirable from a liveness point of view, the state tracked in the partitions will diverge over time, creating two parallel and possibly incompatible transaction histories. It is therefore of paramount importance that network partitions are detected. Such detection could be done by tracking the observed aggregated computational power in the network. A rapid decrease in the block finding rate might indicate that a partition occurred.

2.2. Propagation Method

For the purpose of updating and synchronizing the ledger replicas only transaction (*tx*) and block (*block*) messages are relevant[2]. These messages are far more common than any other message sent on the network and may grow to a considerable size. In order to avoid sending transaction and block messages to nodes that already received them from other nodes, they are not forwarded directly. Instead their availability is announced to the neighbors by sending them an *inv* message once the transaction or block has been completely verified[3]. The inv message contains a set of transaction hashes and block hashes that have been received by the sender and are now available to be requested. A node, receiving an inv message for a transaction or block that it does not yet have locally, will issue a *getdata* message to the sender of the inv message containing the hashes of the information it needs. The actual transfer of the block or transaction is done via individual block or tx messages. Figure 1 visualizes the protocol flow for a single hop in the broadcast. Node A receives a block, verifies it and announces it to its neighbors. Node B receives the inv message and, since it does not know about the block, it will issue a getdata message. Upon receiving the getdata message, Node A will deliver the block to Node B.
At each hop in the broadcast the message incurs in a

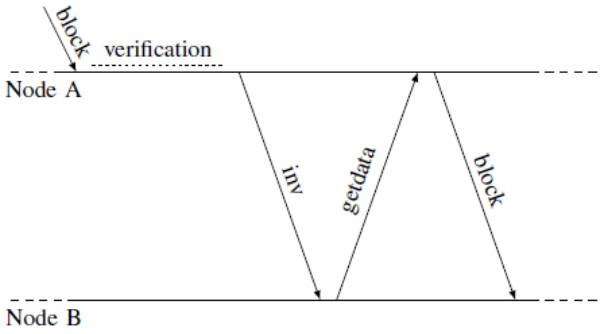

Figure 1. The procedure of forwarding a block message from Node A to Node B.

propagation delay. The propagation delay is the combination of transmission time and the local verification of the block or transaction. The transmission time includes an announcement in the form of an inv message, a request from the receiving party and a delivery. While the inv and the getdata messages are relatively small in size, the block message may be very large up to 500kB at the time of writing. Before the block is announced to the neighbors of a node, it is verified. The verification of a block includes the verification of each transaction in the block. Transaction verification in turn requires random access to data stored on disks.

3. Blockchain Forks

3.1. How It Happens

Let us consider the case of a block being disseminated in the network and how it may lead to a blockchain fork that is only detected by a minority of the nodes. Let $G = (V, E)$ be the network's underlying connection graph, V being the set of all nodes and E the set of connections between the nodes. Starting from a single partition $P_h \subset V$ containing all nodes whose blockchain head is at height h, i.e., they do not know any block for the next height $h+1$. Finding a new block b_{h+1} introduces a new partition $P_{h+1,b}$ which contains the nodes that believe this block to be the head, i.e., it is the first block for height $h+1$ they received. If no other block is found, then nodes adjacent to the cut between P_h and $P_{h+1,b}$ leave P_h and join $P_{h+1,b}$ until P_h is empty and the network as a whole proceeds with the new blockchain height $h+1$.
On the other hand, should another block b'_{h+1} for height $h+1$ be found by a node in P_h, it again introduces a

new partition $P_{h+1,b'}$. In this case nodes from P_h will join $P_{h+1,b}$ and $P_{h+1,b'}$ concurrently until P_h is empty, and all nodes are in one of the partitions with height $h+1$.

Only nodes adjacent to the cut between $P_{h+1,b}$ and $P_{h+1,b'}$ will know both b and b' and therefore able to detect the resulting blockchain fork. Nodes that are in the partition $P_{h+1,b}$, and not adjacent to $P_{h+1,b'}$, will only know b and be completely oblivious to the existence of a conflicting block. A partition $P_{h+1,b}$ could potentially contain only a single node, in the case that the nodes neighbors already know a conflicting block and immediately stop the propagation of b.

The above also applies for transactions that are being propagated. If two transactions that attempt to spend the same output are propagated in the network only the first transaction a node receives will be deemed valid, the second transaction will be invalid according to that node's state and will therefore not be announced to its neighbors.

In the case of transactions, stopping the propagation is a reasonable trade off, that protects the network from transaction spam, at the expense of individual users. However, in the case of blocks, stopping the propagation is not reasonable. The blockchain forks, that are hidden from a majority of the nodes by doing so, are an important indicator of an ongoing unresolved inconsistency. As valid, but potentially conflicting blocks, cannot be created at an arbitrary rate like transactions, forwarding them would not create the possibility of an attack.

3.2. Relationship between Delay and Forks

To analyze the propagation delay quantitatively, I implemented a simple bitcoin network protocol and connected to a large sample of virtual nodes in the network. The measuring node does not relay inv messages, blocks or transactions. Instead it tracks how transactions and blocks are propagated through the network by listening for the announcement of their availability in the form of inv messages. Once the measuring node receives an inv message containing the reference to a block we know that the node which sent the announcement has received and verified the block.

Figure 2 shows the normalized histogram of propagation time t_b for all blocks b in the measured interval. The normalization allows us to use this as an approximation of the probability density function of the rate at which nodes learn about a block. The curve is pretty fitted with Poisson distribution. The median time until a node receives a block is 5.9 seconds whereas the mean is at 10.9 seconds. The long tail of the distribution means that even after 25 seconds there still are about 4% of nodes that have not yet received the block. Since the final task is to reduce the number of blockchain forks, we need to consider the relationship between delay and forks and measure the number of blockchain forks to judge whether the speedup method is effective. It is easy to see that the less the propagation time is, the less the number of blockchain forks is. So it is useful to measure the number of blockchain forks on the current protocol.

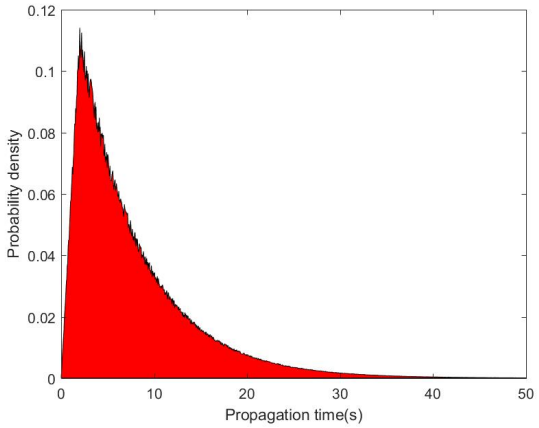

Figure 2. **The normalized histogram of times for all blocks.**

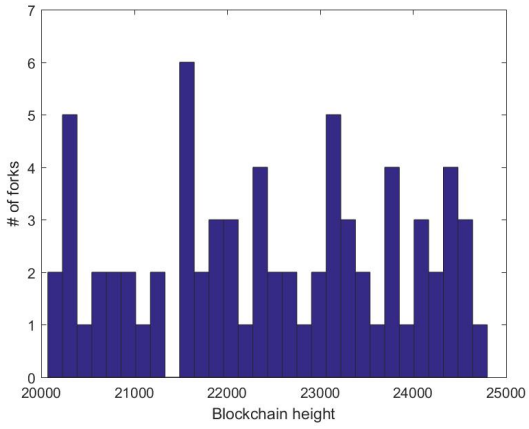

Figure 3. **The histogram of blockchain forks from height 20,000 to 25,000 with original protocol.**

I collected the blocks that have been propagated in the network between height 20,000 and height 25,000. Figure 3 shows the histogram of blockchain forks in these blocks. There were 72 blockchain forks in the observed 5,000 block interval, resulting in an fork rate $r = 1.44\%$.

4. Speedup of Information Propagation

4.1. Minimize Verification

A major contributor to the propagation delay is the time it takes a node to verify a block before announcing it to the network. There is a strong correlation between the size of a block and the time to verify it. As each hop in the propagation has to verify the block before relaying it to its neighbors the delay is multiplied by the length of the propagation paths.
In fact, the verification can be divided into two phases: an initial difficulty check and a transaction validation. The difficulty check consists of validating the proof-of-work by hashing the received block and comparing the hash against the current target difficulty. Additionally, it checks that the block is not a duplicate of a recent block and that it references a recent block as its predecessor to verify that the block is not a resubmission of an old block. The bulk of the validation is done in the transaction validation which checks the validity of each transaction in the block. The block can be relayed to the neighbors, as soon as the difficulty has been checked and before the transactions have to be verified. Therefore the behavior of the node could be changed to send an inv message as soon as the difficulty check is done, instead of waiting for the considerably longer transaction validation to be finished.

4.2. Pipelining Propagation

A further improvement can be achieved by immediately forwarding incoming inv messages to neighbors. The goal of this is to amortize the round-trip times between the node and its neighbors by preemptively announcing the availability of a block earlier than it actually is. The incoming getdata messages for the block are then queued until the block has been received and the above difficulty check has been performed, then the block is sent to the neighbors requesting it. Unlike the first change, this may cause some additional messages being sent as from the hash of the block no validation can be done. An attacker may announce an arbitrary number of blocks without being able to provide them when asked for it. Nodes receiving these spam announces will relay them to their neighbors. Should a node detect that one of its neighbors is announcing blocks that it cannot provide it can switch back to the original behavior of first verifying blocks before announcing them. Even though nodes can be tricked into forwarding inv messages that it cannot provide the block for, the impact is likely to be relatively small. Note that the same attack is already possible by creating an arbitrary amount of transactions and announcing them to the network. As the attacking node can provide the matching transaction, it will not be recognized as an attack.

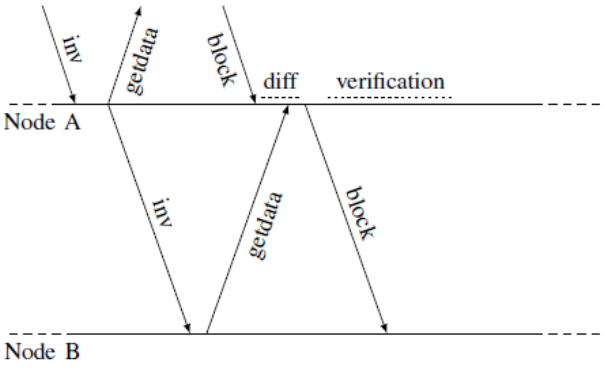

Figure 4. **The pipelining modifications speed up information propagation.**

Figure 4 shows the procedure of pipelining propagation. Notice the verification being divided into two phases (*diff* and *verification*) and the inv message being sent much earlier. That is, pipelining propagation is based on the method "Minimize Verification".

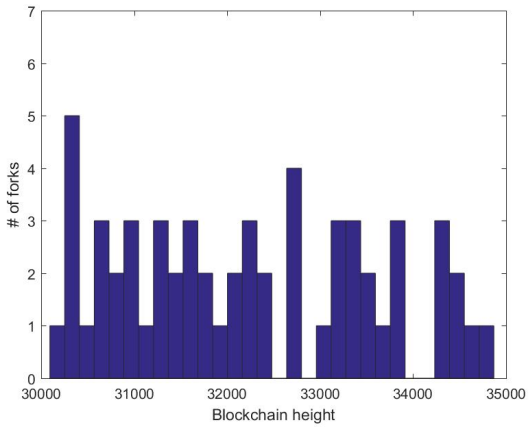

Figure 5. **The histogram of blockchain forks with pipelining modifications**

Figure 5 shows the histogram of blockchain forks with pipelining modifications. Comparing it with the original case shown in figure 3 an effective improvement can be seen. There were 58 blockchain forks and the fork rate was 1.16%, with a 19.44% improvement.

4.3. Relationship between Connectivity and Propagation

The most influential problem is the remote distance between the origin of a transaction or a block and the nodes. To minimize the distance between any two nodes I attempted to connect to every node in the network creating a star sub-graph that is used as a central communication hub, speeding up the propagation of inv messages, blocks and transactions. It should speed up information propagation but also suffer higher bandwidth.

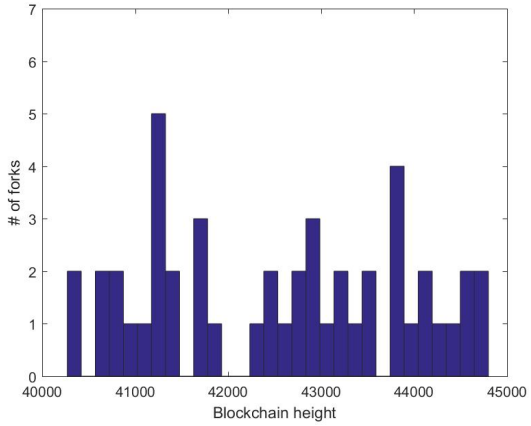

Figure 6. **The histogram of blockchain forks with 500 connections open**

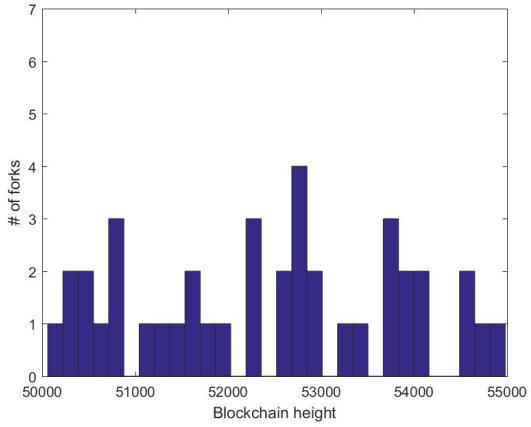

Figure 7. **The histogram of blockchain forks with 1,000 connections open**

So I tried to find the relationship between them to choose the most suitable connectivity with an acceptable bandwidth. I increased the average number of open connections based on pipelining protocol. Figure 6 and figure 7 shows the results with 500 connections and 1,000 connections respectively(initial number is around 32). When 500 connections open, the fork rate is 0.94%, with a 34.72% improvement. With 1,000 connections open, the fork rate is 0.80%, with a 44.44% improvement.

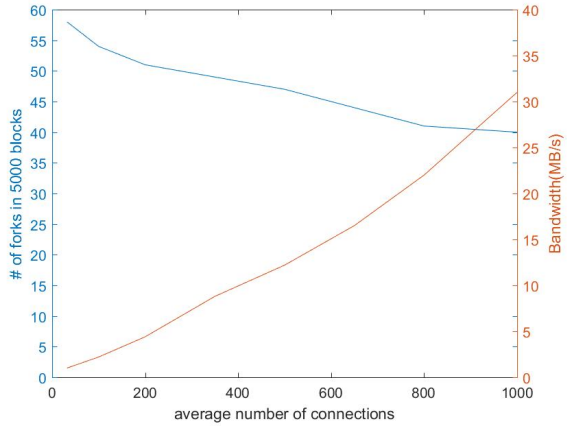

Figure 8. **Fork numbers and bandwidth with different connections**

To understand the relationship between fork numbers and bandwidth better, I tried different numbers of connections from 32 to 1,000, and got the result in Figure 8. It shows that with the increase in connections number, fork rate decreases slower and slower, and bandwidth increases faster and faster. This indicates that we can't just increase the connections number and expect it to work efficiently. We will meet the bottleneck that suffers high bandwidth. When 1,000 connections open, the bandwidth is around 31 MB/s, which is not so acceptable for users. In this way, we may choose fewer connections such as 400 to seek a fast propagation time(low fork rate) under the limit of bandwidth.

5. Conclusion

So far we have analyzed how information in the Bitcoin network is disseminated in order to synchronize the ledger replicas. The reliance on blocks not only delays the clearing of transactions, but it also poses a threat to the network itself. As blockchain forks are symptomatic for an inconsistency in the ledger replicas, it is important that the nodes in the network are aware about them. However, due to information eclipsing, most nodes are unable to detect them.
Finally, some changes were done to the current Bitcoin protocol that reduce the risk of a blockchain fork. The

measurements show a visible improvement and also the bottleneck due to bandwidth. The root cause of the problem maybe intrinsic to the way information is propagated in the network. To find more efficient method with more advanced propagation strategy will be a challenging task.

References

[1] Alqassem I, Svetinovic D. Towards Reference Architecture for Cryptocurrencies: Bitcoin Architectural Analysis[C]// Internet of Things. IEEE, 2015:436-443.

[2] Ersoy O, Ren Z, Erkin Z, et al. Information Propagation on Permissionless Blockchains[J]. 2017.

[3] Turner A, Irwin A S M. Bitcoin transactions: a digital discovery of illicit activity on the blockchain[J]. Journal of Financial Crime, 2017:00-00.

[4] Decker C, Wattenhofer R. Information propagation in the Bitcoin network[C]// IEEE Thirteenth International Conference on Peer-To-Peer Computing. IEEE, 2013:1-10.