Distributed Consensus Algorithms in Blockchains

Anonymous
Department of Computer Science
Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY 14586
anonymous@cs.rit.edu

Abstract—A Blockchain is a public immutable distributed ledger and stores a variety of transactions. Because there is no central authority that regulates the system and users don’t trust each other, a blockchain system needs an algorithm for users to reach consensus on block creation. In this report, we will explore 3 consensus algorithms: Proof-of-Work, Proof-of-Stake and Proof-of-Activity.

Index Terms—Consensus Algorithms; Blockchains; Distributed Systems; Cryptography; Proof-of-Work; Proof-of-Stake; Proof-of-Activity;

I. INTRODUCTION

A Blockchain is a public immutable distributed ledger. Its application ranges from cryptocurrencies, which popularized the technology, to copyright registration, supply chains and smart contracts[1]. As with many other distributed systems, reaching consensus on a single global history among mutually distrusting parties is one of the most important problems in a blockchain system and a lot of consensus algorithms have been proposed and used. In this report, we will first go over the blockchain technology and its problems, and then examine Proof-of-Work and Proof-of-Stake, which are most popular choices today. Also, we will look into Proof-of-Activity, which is introduced in order to overcome the drawbacks that Proof-of-Work systems have.

II. BLOCKCHAINS

A Blockchain is a public immutable distributed ledger and has no central authority, eliminating a single point of failure[2]. Their application range from cryptocurrencies, which popularized the technology, to copyright registration, supply chains and smart contracts[1]. However, the blockchain technology involves a lot of difficult concepts from distributed systems and cryptography. Thus the following subsections summarize the key concepts of the blockchain technology in layman’s terms in order to introduce the preliminary concepts that are crucial to understand consensus algorithms. Also, we will look into the problems of blockchains that consensus algorithms address.

A. Architecture

New transactions are recorded in a block that will be appended by miners to the blockchain. To maintain consistency of transactions, each block stores the hash of the previous block header, forming a hash chain as shown in Figure 1. Since blocks are referred to by their hash values, tampering with their contents will cause discrepancies in the chain and is easily detected. Transactions are recorded as a Merkle tree, in which the value of a node is the concatenation of the hash values of the children, and the Merkle root is stored in the block header. Merkle trees make it easier to check the integrity of the transactions.

One of the notable characteristics of blockchains is that no transactions can be changed once published and one of the primary cryptographic technologies that contributes to the immutability is hashing. Hashing functions take digital information of any length and map it typically to a $2^n$ space, and have the following properties:

1) Preimage resistance: Given an output, it is computationally infeasible to find any input that hashes to that input, that is, given $y$, it is difficult to find an $x$ such that $h(x) = y$.

2) Second-preimage resistance: It is computationally infeasible to find any second input which has the same output as that of a specified input, that is, given $x_1$, it is difficult to find $x_2$ such that $h(x_1) = h(x_2)$.

Another notable cryptographic technology used in blockchains is asymmetric-key cryptography, in which different keys are used for encryption and decryption or authentication and verification[3], [4]. In a blockchain system private keys are used to digitally sign transactions and public keys are used to generate addresses and verify signatures. A user’s address is a short string that is composed of alphabets and numbers and is used to send and receive digital assets, and users can have multiple addresses by generating as many private/public key pairs as desired. Private keys, public keys and the associated address are typically stored in a wallet.
wallet also can calculate the total number of assets the user has and create key pairs by using a secure random number generator. It is important not to lose a private key since there is no central authority to recover it from and any asset associated with that key will be lost.

Using the aforementioned cryptographic concepts, blockchains keep track of transactions by storing them in blocks. A block is a fundamental unit of a blockchain and consists of the header, which contains metadata about the block, and the body, which contains transactions.

Special nodes called mining nodes, or miners, maintain the blockchain by publishing new blocks and transactions are validated by them by checking the digital signatures. Mining nodes are encouraged to make sure the transactions are valid since a new block will not be accepted if it contains any invalid transactions. In Bitcoin, for each block, transactions are summarized in a tree structure called Merkle tree[5], which heavily uses hashing and any change to the underlying data would be easily detected. Each block header contains the hash of the previous block’s header, forming a chain of blocks. Note that if a previously published block were altered, its hash would become different.

B. Consensus

Consensus models enable mutually distrusting nodes to work together without central authority or a trusted third party. Proof-of-work, in which whoever solves a computationally intensive puzzle obtains the right to publish the next block, is one of the most popular consensus models today. Another widely used model is the proof-of-stake consensus model, in which the more stake a node has the better chance it has in the new block creation. Consensus models are discussed in details in the following sections.

C. Misconceptions

One of the most common misconception is that no one controls a blockchain. It is not true since there’s a group of core developers that maintain and develop the system. Another common misconception is that there is no trust in blockchainsystems. Even though there’s no trusted third party that certifies transactions, there is still a great deal of trust needed in a various components in the blockchain system such as the cryptographic technologies that the system uses, the developers, most users of the blockchain not being colluding in secret (51% attack) and nodes accepting and processing transactions fairly.

D. Limitations

Even though the blockchain system can impose transaction rules and specifications, they cannot prevent users from acting dishonestly and maliciously, and once a large enough collusion happens, malicious mining nodes would be able to ignore transactions from specific users, forge an alternative chain in secret and submit it when the forged chain is longer than the real chain[6] (Attacking immutability), or refuse to distribute blocks to other nodes. Blockchains can produce hard forks to tackle such malicious activities.

Also, most of Proof-of-Work based blockchain systems require a lot of resources. For example, when a new full node is created it must obtain most of or all the blockchain data, which is as large as 100 GB as of January 2018 and uses a lot of network bandwidth[7].

E. Problems

Because of a lack of central authority, a blockchain system needs a consensus algorithm that allows mutually distrusting nodes to agree on block creation (transaction history). While different consensus algorithms have different block creation process, they all must tackle the following problems:

- **Double-spending**: Double-spending means spending money more than once. The consensus algorithm has to be able to detect and reject blocks that contain double-spending. In Figure 2, since Block 81a has an invalid transaction, no block should be built off Branch A.
- **Forking**: Because of propagation delays, some nodes might not have the latest blocks and build a block off an older block, causing a fork as shown in Figure 2 (Branches B and C). The consensus algorithm must be able to decide which branch to extend on. In the following sections, we will investigate 3 consensus algorithms.

III. PROOF-OF-WORK

In Proof-of-Work, also known as Nakamoto consensus, a user obtains the right to publish the next block by solving a computationally intensive puzzle[8]. For example, in Bitcoin miners look for a nonce such that the hash value of the block header is less than the target value. Bitcoin adjusts the puzzle difficulty every 2016 blocks so a new block is published every ten minutes. Figure 3 illustrates the mining process of Proof-of-Work. The past computational work put into a puzzle does not contribute to one’s likelihood of solving future puzzles, which incentives users to include the new block because they know the other miners will include it and start building on top of it.

A major pitfall of Proof-of-Work is its excessive use of energy in solving the puzzles. Bitcoin consumes more electricitiy that the entire country of Ireland[9]. Another major pitfall is that it is susceptible to the so called the 51% attack, especially when the size of the network is small. The 51% attack is a potential attack on the blockchain system where a malicious user has the majority of the computational power.
The attackers would be able to prevent new transactions from gaining confirmations, allowing them to halt payments between users. They would also be able to reverse transactions that were already published, meaning they could double-spend coins[10].

A. Implementation of Nakamoto Consensus Module

We have developed a consensus module for Scorex based on the Nakamoto consensus model. Scorex is a modular blockchain framework written in Scala that aims to enable easier blockchain experimentation and prototyping by modularizing the various building blocks that make up a working blockchain system[11]. Our Nakamoto-style consensus module was developed by implementing the ConsensusModule trait that Scorex provides. Figure 4 is a class diagram of the module. The public methods are implementation of the methods of ConsensusModule and the 2 private methods do calculations that are essential for Proof-of-Work.

The calculateBlockHash method takes a Block, applies SHA-256 twice to it and returns the hash value. The calculateDifficulty method calculates the mining target value such that it takes approximately 2 weeks to generate 2016 blocks. For every 2016 blocks the new target is calculated by dividing the old target by a correction factor, which is less than 1/4 then 1/4 is used. Table 1 shows an example of 2 rounds of target calculations. The target value that is calculated with the factor 3 in the first round is used for the second round. The block reward, which is calculated in the feesDistribution method, is the sum of the transaction fees and the block creation reward, which is initially set to 50 and is halved every 210000 blocks. Since Scorex is not designed to handle fractions of tokens, new reward amounts are rounded down.

B. Experiment

We conducted an experiment to investigate the impact of the change to the target on block creation by deploying 10 nodes on Docker containers with the initial difficulty set to the value of 61 F’s preceded by 3 zeros in hexadecimal and the difficulty adjustment interval set to 50. We generated 400 blocks and plotted 3 types of graphs.

Figure 5 illustrates target values with respect to block heights. Since we started off with a very easy target, the first target adjustment was drastic. Then the changes of the target became less abrupt as the block height increased and converged, as we expected, since the computational power remained the same. If the computational power had decreased, we would’ve seen an increase of the target in the graph. On the other hand, if the computational power had increased, the conversion would’ve happened with a lower target.

Figure 6 shows the block generation time for each block. Up until block 200, even though the target was adjusted a few times, the block generation intervals more or less stayed the same. However, after the target adjustment on block 200, the intervals started to spread out, which means it took more time to generate blocks on average, reflecting the decrease in the target.

Figure 7 and Figure 8 are the cumulative distributions of the block generation intervals for block 1 through 50 and block 301 through 350, respectively. Note that it took less than 2 second for about 90% of the first 50 blocks. This means that the block generation intervals are highly skewed and the graph is rather distorted. We attribute the unevenness to the target being too easy, causing blocks generated too fast, which make the system unstable. The CDF for blocks 301 through 350, on the other hand, is more even. This is because the system was stable and producing less extreme block generation intervals.

Table 1

<table>
<thead>
<tr>
<th>Round</th>
<th>Factor</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>Target1</td>
</tr>
<tr>
<td>2</td>
<td>1/4</td>
<td>Target2</td>
</tr>
</tbody>
</table>

The block reward for the first block is 50, and for each subsequent block, the reward is halved.
In the Proof-of-Stake model, the amount of stake a user has determines the chance of publishing new blocks[12], as shown in Figure 9. This model is based on the assumption that the more stake a user has in the blockchain system the more likely he or she wants the system to do well. It is generally cost prohibitive to obtain the majority of assets within a system in order to dominate, so the 51% attack is not feasible. Also, this model generally uses a lot less energy than the Proof-of-Work model since there are no resource intensive computations. There are various block selection mechanisms.

In a random selection, also known as chain-based Proof-of-Stake, the system randomly chooses users based on their stake.

Fig. 5. Target Transition

Fig. 6. Block Generation Intervals

Fig. 7. CDF of Blocks 1 through 50

Fig. 8. CDF of Blocks 301 through 350

Fig. 9. Users’ Stakes and Their Chances to Create Blocks

IV. PROOF-OF-STAKE

In the Proof-of-Stake model, the amount of stake a user has determines the chance of publishing new blocks[12], as shown in Figure 9. This model is based on the assumption that the more stake a user has in the blockchain system the more likely he or she wants the system to do well. It is generally cost prohibitive to obtain the majority of assets within a system in order to dominate, so the 51% attack is not feasible. Also, this model generally uses a lot less energy than the Proof-of-Work model since there are no resource intensive computations. There are various block selection mechanisms.

In a random selection, also known as chain-based Proof-of-Stake, the system randomly chooses users based on their stake.
to total system stake ratio.

A multi-round voting, also known as Byzantine Fault Tolerance Proof-of-Stake[13], gives staked users voting power in the block selection process and takes the following steps:

1) The system selects some staked users to generate proposed blocks.
2) The selected staked users vote for the next block. Repeat this process several times if necessary.
3) A new block is decided and generated.

In a coin age-based selection, also referred to as coin age proof-of-stake, staked coins have a property called “age” and users are allowed to create blocks by “spending” aged coins. The “spent” coin has its “age” reset to 0 and the user cannot use it again until after a certain time passes. A user cannot take over the system because of the cool-down timer that is attached to every coin spent generating blocks.

A pure Proof-of-Stake model alone has so called the nothing-at-stake problem, where nodes vote for every chain in case of a fork since they lose nothing by doing so, leaving the forks unresolved. Ethereum, which uses Byzantine Fault Tolerance Proof-of-Stake, tackles this problem by depriving miners that vote for multiple blocks of their rewards.

V. PROOF-OF-ACTIVITY

Proof-of-Activity[14] is a consensus algorithm that is a hybrid of Proof-of-Work and Proof-of-Stake and aims to overcome security problems that Proof-of-Work has by providing miners with more transaction fees to cover the cost of mining.

Bitcoin, together with the Proof-of-Work protocol, has been around for years and proved resistance against various attacks so far. However, when the mining reward decreases to near zero in the future, the only funds the miners obtain from the system are the transaction fees, and the pure Proof-of-Work system will not be able to provide them with sufficient economic incentives to maintain the network since without protocol-enforced limitations on what transactions can go into blocks, a rational miner will always decide to include every fee-paying transaction, even if the fee is very small, because it almost doesn’t cost anything to include a transaction in a block. As a result, miners won’t receive sufficient funds to cover the cost of mining and leave the network, making the system more susceptible to Proof-of-Work-based attacks, because there would be fewer honest miners to compete against.

Proof-of-Activity solves the incentive problem of Proof-of-Work by introducing Proof-of-Stake to provide miners with sufficient transaction fees to cover the cost of mining. Also, due to a faster block generation rate, the difficulty of the puzzles is expected to be lower than that of the pure Proof-of-Work model as shown in Figure 10, making it require less resources to maintain the system.

VI. CONCLUSION

As we have seen, there are different algorithms that allow mutually-distrusting blockchain users to reach consensus. Each one of the consensus algorithms we looked into has its own advantages and disadvantages, and therefore, it is crucial to choose the one that meets the needs and to be aware of the potential attacks.

It is worth noting that while Proof-of-Work and Proof-of-Stake have been around and proven to work, Proof-of-Activity hasn’t been used in practice. Therefore, we believe future work on consensus algorithms should focus on implementation and experimentation of Proof-of-Activity.

ACKNOWLEDGMENT

I’d like to thank Dr. Hu, without whom I wouldn’t have been able to learn this much only in a semester, for his genuine guidance on my study in the cutting-edge field of consensus algorithms in blockchains. I’d also like to thank my academic advisor Cindy and Dr. Bischof for their generous support when times were rough.

REFERENCES


