Investigating Byzantine Agreement Consensus Algorithm of Algorand

Yu Liu

Supervisor: Dr. Ling Chen
Dr. Wei Bian

School of Computer Science
University of Technology Sydney

This dissertation is submitted for the degree of
Master of Analytics

March 2020

I would like to dedicate this thesis to my loving parents …

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Yu Liu
March 2020

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, Yu Liu declare that this thesis, is submitted in fulfilment of the requirements for the award of Master of Analytics, in the school of Computer Science at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise reference or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution. This research is supported by the Australian Government Research Training Program.

Signature:

Date:

06/03/2020

Acknowledgements

I would like to express my sincerest gratitude to my principal supervisor, Dr Ling Chen, who gave me the opportunity to embark on this research degree and for being extremely patient and supportive in guiding me over the course of this journey. I would also like to thank my co-supervisor, Dr Wei Bian, who encouraged me to explore blockchain technology and encouraged me to put the pieces together to form a complete work. I could not have finished this research without their valuable help.

Thanks to all my research colleagues and friends in the school of Computer Science for their kind help to overcome my struggles. I would like to express my appreciation to Mr Wei Wu, Miss Jiamiao Wang, Mr Shaosheng Wang, Mr Yunqiu Xu, Miss Congai Li, and all my other lovely friends in this school.

Last but not least, I thank my parents for their support. They have always encouraged, guided, and supported me, no matter when I succeeded or failed.

Abstract

After its rapid development and broad adoption in its early stage, blockchain technologies are experiencing a bottleneck in terms of their scalability in processing transactions. There have been various proposals to overcome this difficulty, but very few are able to avoid the curse of the blockchain trilemma in relation to balancing scalability, decentralization, and security. However, Algorand demonstrates its superior capability to process transactions and maintain safety when the number of users increase. In particular, its consensus diminishes the probability of chain forks, which generates the feasibility of double-spend attacks in blockchains. In order to determine if Algorand could be the answer to the trilemma, this thesis presents an investigation of its consensus algorithms and a thorough analysis of its performance and some potential downsides of the proposal.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Background

Bitcoin, as the pioneer of blockchain, achieved great success over the past decade. It proposed a decentralized network that can validate and record transactions in a public ledger without trusted intermediates. Many applications adopted its distributed model and tamper-proof structure into their innovations, including cryptocurrencies, smart contracts, and supply chains [15, 29]. It is undeniable that blockchain has a tremendous potential to challenge the existing centralized systems. However, the majority of blockchain applications have to address the blockchain trilemma[22] hurdles before being seen as a feasible alternative to centralized models.

1.2 Consensus

Unlike traditional fiat currency, blockchain-based digital currency does not have an intermediate, such as banks or payment service companies, who have a higher trust level to verify and finalize payments. Without regulation from central authorities, dishonest users can deliberately make payments to different people with the same tokens. This malicious attempt, called double-spending, is the most critical issue for all blockchain applications. Because participants are distributed, there is a high possibility that a simple storage mechanism could lead to diverged versions of the ledger. To prevent the partitioning of the ledger in the network, blockchain relies on the notion that the majority of participants will follow a consensus protocol to reach agreement asynchronously.

Proof-of-work

There have been various categories of consensus protocols implemented and adopted by blockchain applications. Of these protocols, the earliest and most widely adopted is the proof-of-work consensus. In proof-of-work-orientated blockchains, users have to solve computational puzzles by repeatedly guessing pseudo-random answers to append new transactions to the public ledger. The user who finds the answer first will collect the transactions, propose a new block, and append it after the latest block in the blockchain. The block becomes valid when the majority of users confirm its validity and start proposing new blocks after it. In rare cases where two valid blocks are produced simultaneously and temporally produce a chain fork, one of the forks will gradually become the longest chain having most of the proof-of-work efforts if the majority of the users continue building on it. Although there is a low possibility that a chain fork will be produced, an attacker can utilize massive computing power to create forks and reverse transactions forcefully if his chain becomes the longest one. In this rare case, proof-of-work-based blockchains recommend that users confirm the transactions until specific numbers of new blocks are generated after the first observation of the transactions in the public ledger. For example, Bitcoin suggests observing at least six subsequent blocks to confirm transactions after appending into the ledger. Generating six blocks takes approximately one hour, which significantly impacts the efficiency of processing transactions and renders Bitcoin unpractical for day-to-day payments. Furthermore, Bitcoin periodically adjusts the mining difficulty to regulate block generation with a fixed interval. This not only ensures the security of Bitcoin as the total computational power increases but also stabilizes the price of coins and creates a healthy financial ecosystem. It is also the reason that most of the blockchain applications either adopt or extend the structure of the proof-of-work consensus. Inevitably, they also inherit some of the drawbacks that do not allow them to break through the trilemma.

Moreover, proof-of-work faces other issues during block generation. Firstly, the block size limits the volume of transactions that can be stored. Secondly, the massive amount of energy consumed by all miners to generate one block is purely wasted to generate a meaningless unique number. Last but not least, difficult mining generates the pooled-mining strategy that initially allows miners to share computational power and efficiently generate blocks. However, there have been a few private mining pools in which cryptocurrency companies have invested that have started to dominate the majority of the computational power in the system. Although none of the giant pools occupy more than 51% of the power in the system, it raises concerns of centralization if multiple mining pools decide to cooperate in the mining process.

Alternatives

Several alternative consensus algorithms have been proposed to address the issues of proof-of-work. The most accepted concept is the proof-of-stake, which randomly selects representatives according to their tokens to propose blocks and update the public ledger. Instead of allocating heavy computational work to all the miners, it selects a subset of users to generate blocks resulting in the minimum consumption of resources. Therefore, most of the proof-of-stake-based consensus can confirm transactions in a more efficient and quicker way. In terms of security, this scheme relies on honest users owning the majority of the total tokens to defend against the overwhelming probability of attack. Attackers must spend a significant amount of monetary value and take risk losing their investment to undertake an attack. It also eliminates the possibility of brute-force attack using computational resources and the dominance of centralized computational power. However, it is argued that relying on the weighted probability shifts the centralization to digital tokens. Compared with investing in computational power, stacking tokens is more simple and costs less without the heavy computational efforts required to re-produce the chain.

In addition to proof-of-stake, Byzantine Fault Tolerant-based blockchain, like Hyperledger Fabric[1], reaches consensus while assuming a fraction of the network is dishonest. Generally, these blockchain applications validate blocks by collecting more than a threshold of specified votes from the network with the assumption that the distributed system could remain functional as long as $2/3$ of the majority are honest and defend against malicious attacks. The drawback of this system is that most of these proposals select master nodes to take charge of broadcasting and validations, which inevitably causes the framework head towards centralization. It also potentially exposes users to other forms of attack, especially targeted attacks on users holding a significant number of tokens.

Algorand

Many proposals have attempted to mitigate the effect of chain forks and improve scalability through implementing a new consensus algorithm, but they all struggle to balance every aspect of the trilemma. Some variants of proof-of-work sacrifice the complicated cryptography of mining to shorten the interval of block generation. This could increase the volume of processed transactions but weaken the security of blockchains and increase the probability of chain forks. In the case of proof-of-stake, it solves the problems of centralized computational power and improves the efficiency of the network. However, a heavy reliance on the probability weighted by monetary value shifts the centralization from computing power to tokens. More importantly, the chain fork that directly causes the double-spending threats is not optimized.

However, Algorand, a proof-of-stake blockchain combined with Byzantine Fault Tolerance, achieves a breakthrough in relation to the trilemma. The distributed system consists of an efficient cryptographic sortition weighted by the user's weights, and validation processes secured by a Byzantine Fault Tolerance-inspired voting scheme. It is also capable of securing the ledger when a small proportion of users are malicious. As observed from its extraordinary results, Algorand can reach agreements within a short period, and the performance remains constant while scaling to a large number of users[23]. Furthermore, its block generation scheme eliminates the possibility of chain forks which are the main cause of double-spending attacks.

1.3 Research Questions

The blockchain trilemma refers to the struggle of achieving decentralization, scalability, and security without sacrificing one or two of these properties. As the hype surrounding blockchain continues to heat up, its scalability, in terms of handling a larger number of transactions more quickly desperately needs a solution. Improving scalability without weakening decentralization and security is easier said than done. In the case of Bitcoin, some of the community suggested increasing the block-size limit to achieve a more massive throughput of transactions. Theoretically, this is the most efficient and straightforward way to do this, without making significant changes to the existing framework, but the trade-offs are not just an increment in message delays. A longer propagation time exacerbates the problem of chain forks and partitions in the network. Furthermore, it increases the storage cost of maintaining the full blockchain, which could potentially lead to more centralized mining. The same trade-off applies not only to Bitcoin but also to other blockchain applications.

Algorand, a democratic and scalable cryptocurrency, is a promising blueprint to show that blockchain can achieve safety with minimum computational cost, a negligible probability of forks, a tolerance to malicious behaviors, and it can confirm transactions quickly. With these properties, it seems able to solve the trilemma problem. Although Algorand delivers promising results, a few have argued that its security assumptions are impractical in the real environment[14, 47]. In order to have a better understanding of how the consensus algorithm can solve the trilemma problem, this thesis presents an exploratory investigation of the consensus algorithms, broken down into the following questions:

- What is the best way to investigate Algorand's consensus algorithm?

- How does the consensus algorithm confirm a block quickly while scaling to a large number of users?
- How does the consensus algorithm perform and tolerate dishonest behaviors?
- What effects does the sortition algorithm have on agreement when the tokens are distributed unevenly?

1.4 Aims & Objectives

In order to thoroughly analyse Algorand's consensus algorithm with limited released resources, the investigation is carried out by fulfilling the following objectives to answer the research questions:

1. To construct a close-to-real distributed network simulator
2. To construct a blockchain with Algorand's consensus algorithm
3. To collect simulated data and analyze the performance of the consensus algorithm
4. To identify possible drawbacks and improvements to the consensus algorithm

1.5 Organization of Thesis

This thesis comprises five chapters. The overall structure is shown as follows:

- Chapter 1: Introduction
- Chapter 2: Literature Review
- Chapter 3: Investigation of Consensus Algorithm
- Chapter 4: Analysis & Findings
- Chapter 5 Conclusion

Chapter 2 provides a literature review on the standard structure of blockchain applications. It also reviews different consensus algorithms adopted by blockchains to reach agreement and the difficulties in improving them. Chapter 3 describes how the investigation is performed, including the implementation and analysis of a simulator and Algorand's consensus algorithms. Chapter 4 gives an analysis of the results and verifies the performance of the algorithms with different configurations. Lastly, chapter 5 concludes this thesis.

Chapter 2

Literature Review

2.1 Blockchain

2.1.1 Chain Structure

In the past decade, the world witnessed the rapid growth of blockchain applications in both the financial market and technology. Among various categories of blockchain applications, Bitcoin has to be the most well-known and valuable one. It was initially proposed by Satoshi Nakamoto in 2008 to serve as a public ledger that valid transactions without enforcing trusted third party authorities in a Peer-to-Peer network. The public ledger can be visualized as a chain of linked blocks where transactions are stored. In the chain, individual block binds itself with a parent block using a cryptography link. Within each block, transactions along with a set of metadata are serialized and hashed to generate a hash-based proof-of-work, which cannot be changed unless repeating the proof-of-work[37]. The proof-of-work generated hash serves as a unique and pseudo-random identification to a single block, which is called block hash in blockchain applications. Any attempt to add or remove data in an existing block will generate a new block hash and cause domino effects to the subsequent blocks of it. For example, if an attacker reversed a transaction and resulted in a new block hash in block 10 in Figure 2.1, the descendant blocks' block hash will be changed because of the previous block hash field has been changed. In order to validate the new blocks, he has to repeat the proof-of-work process for block 11 and block 12 until the latest block. The longer the chain grows, the more work the attacker has to do. Theoretically, he can never produce a new chain to bypass the chain maintained by the majority, unless he has a tremendous amount of computing power to control the entire network. It plays a critical role in securing the blockchain and preventing the network from reaching diverged versions of ledger. The similar kinds of strategies to guide the distributed network reaching an agreement are commonly called consensus algorithm. Apart from consensus algorithms, digital signature mechanisms are leveraged to encrypt, verify, and secure transactions, while blockchain does not have any trusted intermediate to validate the legality of transactions. For this reason, most of the blockchains feature with anonymity, autonomy, transparency, and immutability[32].

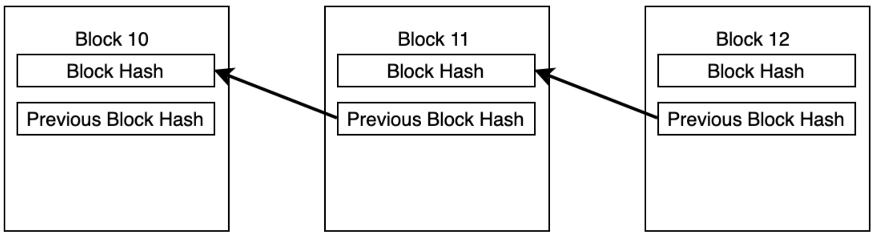


Fig. 2.1 Simplified Visualization of Blockchain

2.1.2 Asymmetric Encryption

In the absence of a trusted intermediate, blockchain relies heavily on asymmetric encryption to secure the safety of the data in the system. The asymmetric cryptography scheme generates a pair of private keys and public keys which allows the user to sign a message digitally with the private key and prove its correctness with the public key. The generation of a key pair is dependent on a one-way function that produces a public key from a private key, and the private key cannot be forged by only knowing the public key. Moreover, in this system, the sender can use the receiver's public key to encrypt a message, while the receiver has to decrypt the message using the private key [49]. Hence, in the blockchain, participants privately reserve their secret key but reveal their corresponding public keys to the entire network. This property gives the system the feature of anonymity. Users can send transactions as long as they provide publicly verifiable proofs to certify their ownership. Real-world identification is not necessarily involved or required in public blockchain applications.

2.1.3 Transactions

In a blockchain network, users make and broadcast transactions to announce ownership transfer from themselves to new owners. Despite payments in the digital cash model, transactions can be implemented in other scenarios, such as the real-time tracking of information in the supply chain industry and data records in the Internet of Things. Under most circumstances, one transaction consists of an input and output field, which indicates where the tokens are from and where they are to be paid. Both of these fields contain asymmetrically encrypted messages, which can be encrypted privately or verified by the public key.

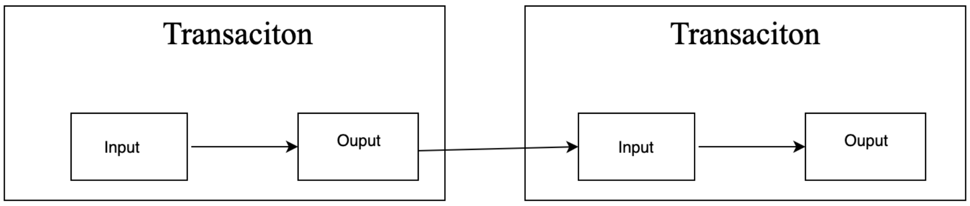
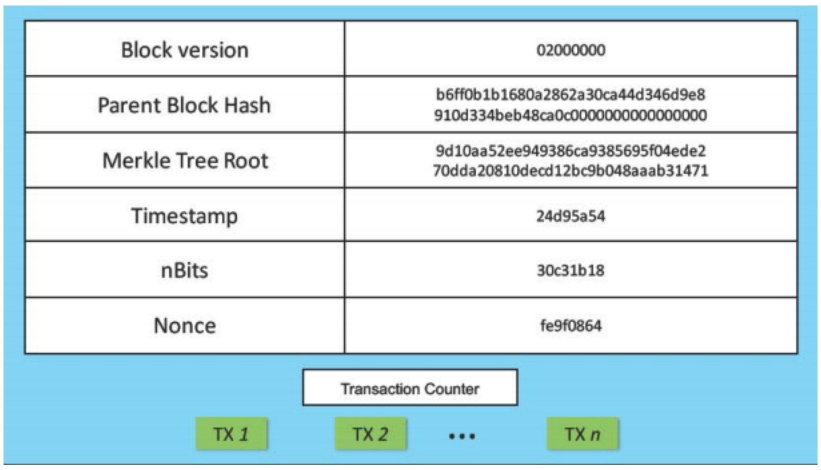


Fig. 2.2 Transaction Example

Figure 2.2 gives a brief example of a transaction in a blockchain. The input field points to the output field of a previous transaction where the sender receives money from either the system or other participants. Furthermore, the input field contains the number of tokens along with a digital signature of this transaction and the public key to unlock a locking script in the previous output field. To complete the transaction, the output field records the number of tokens that the receiver will get and it has a locking script that can be redeemed by the receiver's corresponding public key. After the transaction is collected into a new block and is appended into the longest blockchain, it can be considered to be a confirmed transaction.

2.1.4 Block

Blocks are the essential building blocks of blockchains. Starting with a genius block, a blockchain generates a new block pointing to an existing one in the chain. A block commonly has a unique hash value proving its uniqueness in the chain and other metadata required by different blockchain applications.



Block version	02000000
Parent Block Hash	b6ff0b1b1680a2862a30ca44d346d9e8 910d334beb48ca0c0000000000000000
Merkle Tree Root	9d10aa52ee949386ca9385695f04ede2 70dda20810decd12bc9b048aaab31471
Timestamp	24d95a54
nBits	30c31b18
Nonce	fe9f0864

Transaction Counter
TX 1
TX 2
...
TX n

Fig. 2.3 Block Structure[51]

Figure 2.3 illustrates a commonly used block structure, which contains:

1. Block version which specifies the version of protocols to validate this block

2. Parent Block hash which indicates the unique hash value of its predecessor block
3. Merkle Tree Root which is the data structure to store all the transactions' hashes and enables a fast search and verifies the procedure
4. Timestamp which gives the block generation time
5. nBits which indicates the mining difficulty in the proof-of-work based system
6. Nonce which is a field used in a computational puzzle to produce a unique hash value

2.2 Consensus Algorithm

Blockchain applications run in distributed networks where none of the nodes have a dominant power to control the direction of running. Therefore, this raises a crucial problem of how an individual node reaches agreement on new blocks or updates to the system when everyone shares a similar level of trust? Reaching consensus in such a network can be transformed into the Byzantine General's problem, which describes how a distributed computing system follows specific strategies to avoid total failure when information propagates imperfectly or part of the system fails[30]. In order to securely reach agreement, a variety of consensus algorithms have been developed for blockchain applications, including proof-of-work, proof-of-stake, and other strategies.

2.2.1 Proof-of-Work

The concept of proof-of-work requires a moderate amount of computational work from the requester to provide proof to the server to prevent service abuse and spam attack[25]. When Bitcoin was introduced back in 2008, it proposed a similar system as Adam Back's hash cash, which computes digital tokens from a CPU cost-function[3]. Generally, the proof-of-work system will initialize feasible computational puzzles of moderate difficulty for the user to solve. The solution will become proof for the user to claim its right to perform any operation on the blockchain. As soon as a solution has been proved, no changes can be applied unless the computation work is repeated. This is the most fundamental component of the system in defending it against malicious attack and avoiding messages spamming the network. To encourage users to solve puzzles and grow the chain, the system gives them incentives to keep producing new blocks and allows them to surcharge proportional to the size of the transactions for recording transactions in blocks.

Bitcoin

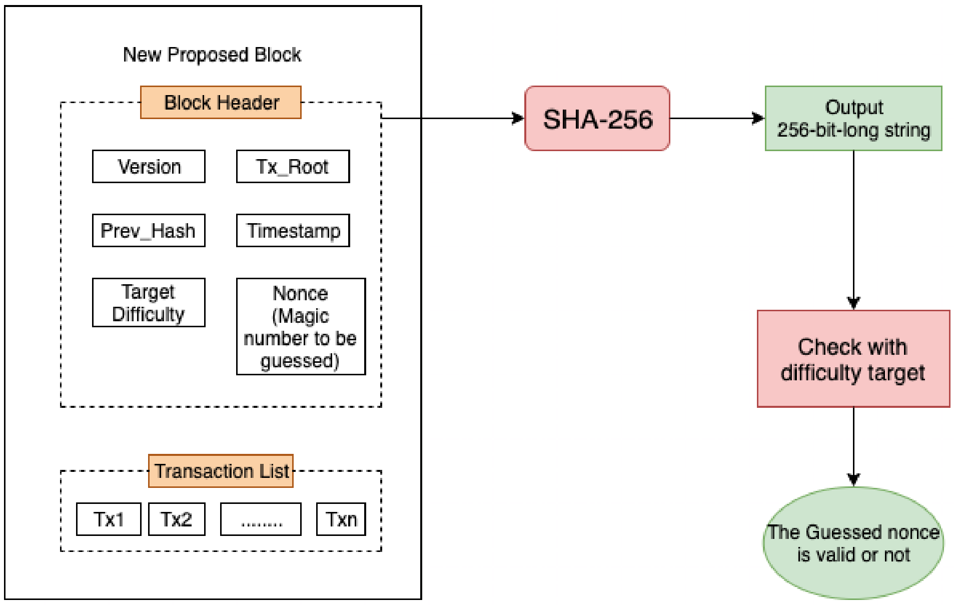

New Proposed Block
Block Header
Version
Tx_Root
Prev_Hash
Timestamp
Target Difficulty
Nonce (Magic number to be guessed)
Transaction List
Tx1
Tx2
....
Txn
SHA-256
Output 256-bit-long string
Check with difficulty target
The Guessed nonce is valid or not

Fig. 2.4 Computational Puzzle in Bitcoin[39]

Of all the proof-of-work-based blockchain, Bitcoin was the first and requires the most intensive computational work. The value of bitcoin tokens rose greatly to approximately US $20,000 by the end of 2017, and was seen as digital gold in the cryptocurrency market. Behind all the hype and madness of Bitcoin is a robust cryptography function and consensus algorithm. Firstly, Bitcoin generates a computational puzzle based on the SHA-256 hash function, which generates a unique output with a fixed length. The uniqueness suggests that the output will be different once any change has been applied to the input.

Further, the randomness of the hash function ensures that output is unpredictable and cannot be constructed without knowing the input. With these features, Bitcoin periodically generates a computational puzzle requiring a user to find an SHA-256 result to satisfy certain difficulty levels. Specifically, the user has to construct a block header containing a handful of fields of metadata described in the previous section. By hashing the concatenated data from the block header and incrementally changing the Nonce data from 0 by 1, the user will get a unique output every time. However, to limit the possibility that large numbers of users solve the puzzle simultaneously, Bitcoin only approves the result, starting with a fixed number of zeros. In addition, the hash output has to be lower than a target threshold value specified by the system. Otherwise, the output will not be qualified as a solution to the puzzle. Since the output of SHA-256 is random and unpredictable, the only way to find a solution is by guessing the nonce one by one. To claim the right to append a new block, users compete with each and whoever solves the puzzle first has to broadcast the block to the network. After it has been confirmed by the majority of the network, a new round will start. As only those blocks finalized in the longest chain will be considered valid, it is always suggested to start a new trial as soon as a new block is announced.

To regulate the block generation interval, Bitcoin introduces the field of difficulty discussed in the previous section. It acts as a threshold value that forces computational solutions to be lower than it. In addition to regulating solutions, Bitcoin will adjust the difficulty every 2016 rounds to regulate block production within a fixed interval of around 10 minutes. In Bitcoin, users who always attempt to solve cryptography puzzles and produce blocks are referred to as miners, which is widely applied to other proof-of-work-based applications. After successfully generating one block, miners will be rewarded with tokens that are newly generated in this block by the system to encourage them to maintain the liveness of the network. The excellent price of Bitcoin stimulates miners to invest in expensive computing equipment and they expect high profits to be returned as the mining reward. As a result, the total hashing power continues to rise, which forces the system to increase the difficulty to control block production intervals. Figures 2.5 and 2.6 clearly show that the hash rate and difficulty started to increase when Bitcoin was drawing wide attention from developers, miners, and investors. The current estimated time for a solo miner is approximately 98 years with moderate computing power. The probability of finding a new block is similar to looking for a specific star in the infinite universe.

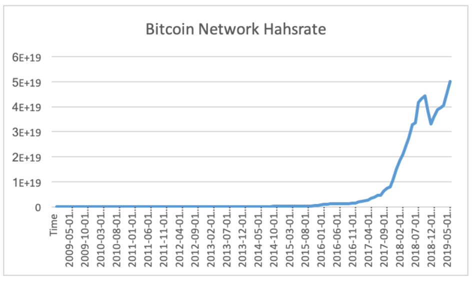


Fig. 2.5 Hash Rate

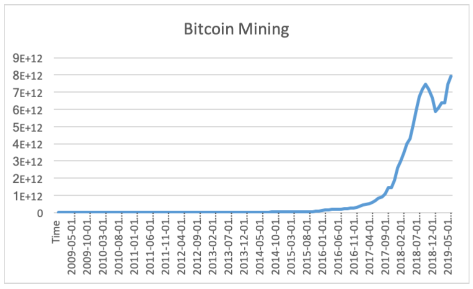


Fig. 2.6 Difficulty

Mining Pool

The enormous amount of effort required to create a new block generates a new strategy called Mining Pool where miners share their computing resources with others to find a nonce that solves the cryptography puzzle since the chance for a solo miner is merely zero. Mining pools are funded and managed by private companies focusing on the Bitcoin business. Public mining pools allow an individual miner to join the pool in order to gather more computing resources. In order to join the pool, individual miners will be assigned with relevant loads of computational work proportional to their hashing power. If a new block is successfully confirmed into the chain, the block reward will be distributed according to their contribution to the process. This seems feasible and is the only way for people to make profits from mining. However, everything has its pros and cons. It does not only allow a solo miner to participate in mining, it also leads to an issue of over-concentrated hash power and potential security threats.

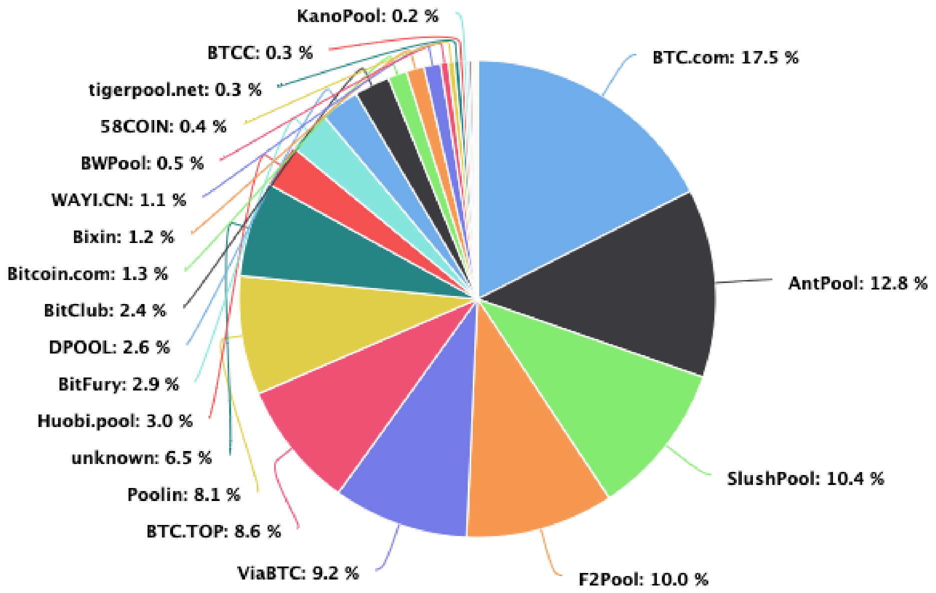


Fig. 2.7 Mining Pool Distribution[9]

As discussed in the Bitcoin white paper, the system remains secure as long as honest nodes control the majority of the computing power, estimated to be 51 %, and they will produce the longest chain and surpass the speed of malicious nodes [37]. Figure 2.7shows approximately 20 major mining pools possessing tremendous computing power control almost 100 percent of the total hash power and dominate block generation in the current Bitcoin network. From the perspective of maintaining the network, this guarantees that Bitcoin will keep supplying new coins, confirming transactions and growing chains. No matter how hashing powers are concentrated, the entire network will remain live when pools perform honestly.

However, mining pools in China are estimated to occupy around 80 percentage of the total hash power, which raises concerns that the over-powered hash rate will perform attacks on the consensus algorithm and control the growth of the chain. The hashing power has already exceeded the minimum security level, which suggests that they can manipulate the entire network's trade by producing blocks recording transactions in favor of themselves and initialize double-spending attacks.

Double Spending & Chain Fork

Why is the concentrated hashing rate in mining pools causing anxiety in the community? Because overpowered computational resources enable a user to perform a double-spend attack, a typical malicious behavior in the digital cash scheme. Different from fiat currency, digital currency can be spent more than one time before payment is finalized. This is where the bank comes into play as a trusted intermediate that manages tokens and verifies transactions. In a centralized system, a bank will create, verify, and approve a transaction for its customers. Let us say that user A wants to make a payment of $5 to merchant B via credit card. He will authorize his bank to create a $5 transaction from his account to the merchant's account. Before completing the transaction, the $5 turns into a pending state and the bank will deny any malicious behaviors from user A. As soon as the bank of merchant B receives the payment and successfully verifies the legality of the payment, it will confirm the transaction and append the new fund into B's account. Hence, a bank with a higher level of trust will take care of the payment processes and secure the customers' assets. However, the most apparent drawback of this centralized scheme is that intermediaries gain complete control of everything. If merchant B's bank insists that they did not receive a transaction from user A and hides the transaction request, merchant B does not have any evidence to claim back his money from the trade.

Moreover, malicious attackers can manipulate the system by only targeting central authorities. Users have no privacy when intermediaries store their payment history and personal details. Although a trusted bank or operators try their best to protect personal information, it is common that leaking could take place at any second. Moreover, governments with a certain authority level could pressure a financial service provider to provide customers' details and trading history. Therefore, the bank as a service provider and trusted intermediary is in the most vulnerable spot in the centralized system.

In contrast, Bitcoin is a decentralized system without any trusted intermediary involved in the system. There is no central node to ensure the ledgers in the distributed nodes are the same [51]. Following the rules of the proof-of-work consensus, the miner who solves the computational puzzle first broadcasts its result to the network. If the new block propagates to the entire network before any other miner solves the puzzle during this period, all nodes will reach a consensus on this new block. However, due to network latency, propagation is not fast enough to reach each node of the network simultaneously.

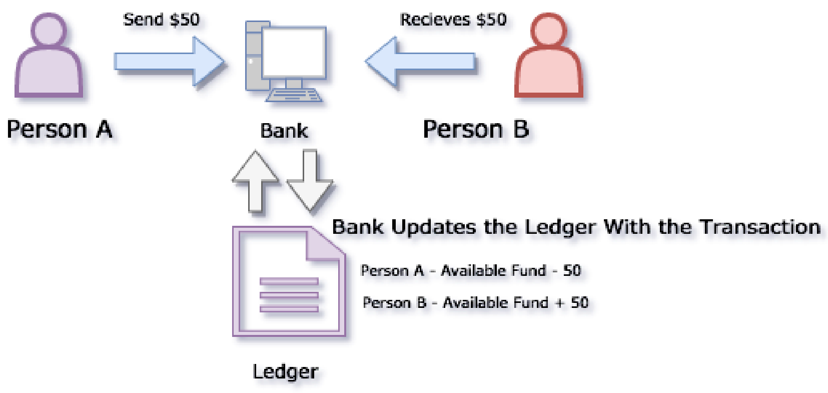

Send $50
Recieves $50
Person A
Bank
Person B
Bank Updates the Ledger With the Transaction
Person A - Available Fund - 50
Person B - Available Fund + 50
Ledger

Fig. 2.8 Simplified Payment Process of Bank System

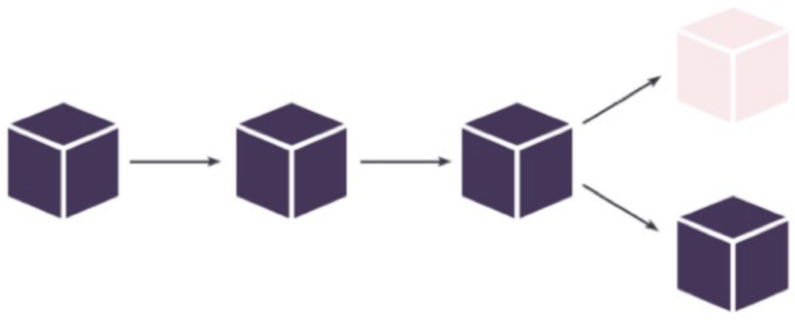


Fig. 2.9 Forked Chain

Meanwhile, some nodes may reach agreement on a different block and start a new round of mining. In this rare case, the network reaches agreement on a diverged version of blocks and generates two chains for miners to mine new blocks. To recover from a diverged ledger, Bitcoin states that only the longest chain that has the largest pool of computing power could serve as proof [37]. This also explains why Bitcoin's chain is never one single straight chain. Splits in the chain could take place at any time and at any height in the system. As long as the majority of miners start mining on one of the split chains, there will be one that outpaces another.

Blocks that are not in the longest chain are marked as invalid. In these invalid blocks, transactions that are not recorded in the longest chain's blocks are broadcast back to the transaction pool, where pending transactions are stored. By using this temporary weakness in the blockchain, attackers can send two transactions with the same token to different people, because the coin cannot be split like fiat currency. In Bitcoin, a user's balance is the sum of the total number of satoshi (minimum Bitcoin token unit) in his unused transaction outputs in his received payments. If user A only has a five satoshi balance from one transaction output field, and he needs to make a three satoshi payment to user B, the transaction will consist of five satoshis as input, three satoshis to user B and two satoshis back to user A as output fields. These five satoshis will be invalid until a block in the main chain records the transaction. Hence, a malicious user with enough computing power can temporally create a blockchain fork on purpose. One possibility is that the block has stored his transaction end up in the side chain rather than the main one. As a result, the transaction will be unconfirmed. The process of this attack is briefly illustrated in Figure 2.10, where Bob successfully performs a double-spend attack on Alice. The second possibility is that the attacker made two different payments with the same token. Two miners accidentally generated two blocks at the same height without noticing that Bob created two transactions using the same token. Both transactions are recorded separately in these two blocks. However, one of the transactions will become valid when one of the chains finally surpasses the other and becomes the main chain. In these two cases, the attackers have performed the double-spending attack successfully. For this reason, Bitcoin suggests that users observe a further six blocks to be mined after the block where the transactions are recorded. This is the most crucial reason why a bitcoin payment is not efficient or scalable like traditional payment services, such as Mastercard and PayPal.

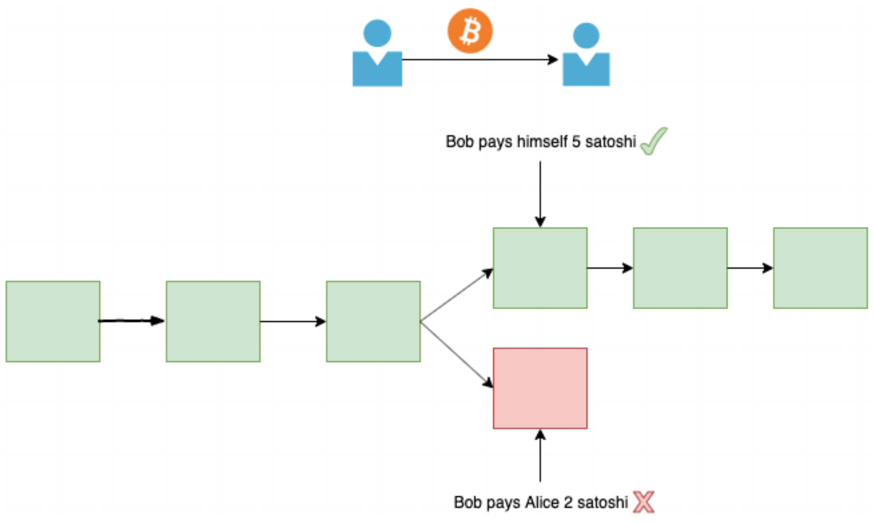


Fig. 2.10 Double Spend Attack

There are two types of Bitcoin forks which enforce the new network rules. Firstly, a soft fork is backward-compatible, which means that the old version of the protocols can still verify the new blocks as valid [2]. In contrast, a hard fork always requires nodes to upgrade their software. Otherwise, new blocks mined under the new rule will be seen as invalid by the old one. However, both of them will result in a fork because the enforced new rules will not validate the blocks mined following the old protocol after the adoption of new one. Hence, a software fork requires the majority of miners and nodes to update their software to enforce the new protocol in the new chain. Last but not least, the fork can be utilized by bitcoin to recover from cyberattacks or temporary malfunctions of the software.

Variants of PoW

As the mining difficulty continues to climb, it is becoming harder for solo miners to participate in mining the system. As discussed in the previous part, immense mining pools are dominating the block generations. Furthermore, the electricity consumption of the mining hardware raised the community's concerns with greenhouse gas emissions. Some argue that bitcoin mining is nonsense and a waste of computing power. Moreover, bitcoin's block generation interval is regulated by the system, which makes the safe confirmation of transactions seem impractical in day-to-day payments. In comparison, payment service providers such as Visa and Mastercard, handle approximately 1700 transactions per second [38]. To resolve the aforementioned drawbacks, both researchers and community members have proposed and implemented various applications which utilize either new protocols or improved algorithms.

In August 2017, Bitcoin cash permanently hard-forked from the Bitcoin main chain with an incompatible software update [31]. As Bitcoin was extremely popular at that time, the block size limit slows down the transaction speed, and the community needs an update to bypass the restriction. Bitcoin Improvement Proposal number BIP141[21] was proposed to improve block storage and transaction efficiency. However, a large group of miners and community members and miners who refused to adopt the new proposal kept mining with older protocols, resulting in a fork. They thought an increase in transaction volume in the block would not effectively mitigate the struggles of scalability and transaction speed. In contrast, they designated a breakdown on the limitation of the block size. Furthermore, they wanted to maintain Bitcoin as a pure transactional currency rather than a digital investment[43]. As a result, Bitcoin Cash hard forked from Bitcoin but inherited most of the Bitcoin's features. The most notable improvements are the increased limitation of block size to 8MB and decreased mining difficulty. The following graph shows the difference in the mining difficulty since the hard fork.

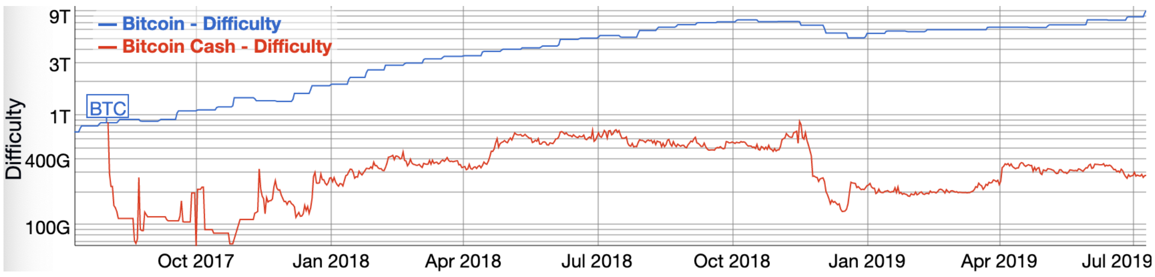


Fig. 2.11 Difficulty Comparison of Bitcoin and Bitcoin Cash[7]

The immense success of Bitcoin stimulated the generation of large number of alternative coins. Some of these altcoins are constructed on the framework of Bitcoin. Litecoin substitutes the SHA-256 hash with a Scrypt function. From the perspective of encryption, it is designed to prevent brute force attacks to password-based key derivation function, such as the SHA-256 function in Bitcoin, by requiring an intensive amount of memory [26]. The memory requirements make the paralleled calculations of the SHA-256 hash impossible in Litecoin. Additionally, Litecoin confirms transactions at a faster speed than Bitcoin and with shorter block generation intervals. Notably, Litecoin has become the second most adopted cryptocurrency after Bitcoin.

Similar to Litecoin, Tromp suggested a replacement of SHA-256 with the Cuckoo hash function that reduces the complexity of the computational puzzle. As shown below, the Cuckoo hash maps a key to a location in two tables, where individual keys have two alternative positions. If a new key is inserted into the position of A, A will be removed from the left table to the position indicated by the arrow pointing to the right table. The hash function keeps inserting new keys until it reaches a maximum number of iterations or the tables are completely filled [40]. In order to solve the Cuckoo hash-based puzzle, the Cuckoo hash table stores enumerated nonces as alternate key locations. If there is a graph containing a fixed length of cycles, the puzzle will be solved[45].

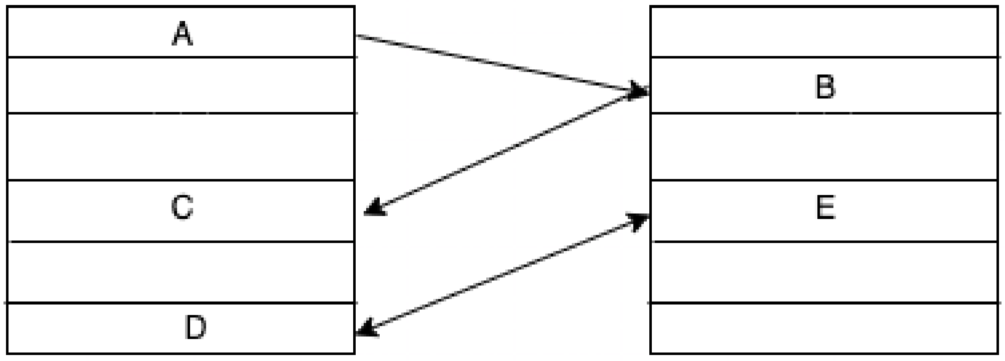


Fig. 2.12 Example:Cuck Hash tables

Different from utilizing cryptography functions, King presented Primecoin whose proof-of-work was established upon searching the prime number[27]. In Primecoin, miners have to compute prime numbers that extend the Cunningham chain of the second kind and bi-twin chain of the first kind. Scientists can access the prime number in the blockchain. This is one of the first blockchain applications that transfer the meaningless mining in Bitcoin to generate some scientific computing values.

2.2.2 Proof-of-Stake

The concept of proof-of-stake consensus was initially proposed by a Bitcoin community member to transfer computing power-weighted mining to a token-weighted vote[44]. The system weights individual users by their tokens over the total tokens in the system. A higher ratio indicates a higher trust level because users who possess a large number of tokens are less likely to attack the system to generate threats to their assets. Replacing the proof-of-work based consensus rule could potentially speed up the transaction process, reduce the cost of mining new blocks, and allow stakeholders to participate in the network. Therefore, the system will be secure as long as honest users own the majority of tokens.

Nextcoin is a pure proof of stake blockchain application, which assigns a feasible target for miners to compute. Equation 2.1 calculates target values[48], where B_e representing the individual's total effective tokens is the only parameter that differs from the probability of solving puzzles. To prevent the system from stalling due to a lack of active users, NextCoin allows coin leasing between accounts. Users can temporarily lease coins to other accounts for a limited period. These coins will be added to the receivers' effective balance temporarily.

$$T = T_s \times S \times B_e \tag{2.1}$$

1. T represents the target hash value
2. T_s represents the base target
3. S is the time since the last block's generation
4. B_e is the balance of the user

Peercoin is one of the earliest blockchain applications to adopt proof-of-stake consen- sus. The system qualifies miners according to the concept of the coins' age, which is calculated from the length of time the user has held the coins[28]. The coin's age significantly affects the difficulty of the computational puzzles the miners have to solve in the Peercoin's protocol. Miners with more extended holding periods and a larger quantity of coins will theoretically have a higher probability of mining the next block. However, there are some time-based constraints to regulate miners' behaviors. Firstly, coins have to be held by miners in their wallets for at least one month. The reset will force a miner to a cool-down period of thirty days to allow others to participate in mining with a higher probability[42]. Moreover, the cool-down period increases the cost of malicious users who conduct an attack with a significant amount of investment in coins. This might result in a fall in market price to increase the cost of the attack.

Bentov proposed a proof of activity scheme extended from the Bitcoin protocol. He argued that pure proof-of-stake mitigates some of the drawbacks of Bitcoin, such as reducing the power centralization, but it shifts the risks to the centralization of tokens[4]. In the proof-of-stake system, users possessing a substantial amount of tokens could enforce double-spending attacks, since they overwhelm others by a much larger probability of mining blocks in consecutive rounds. The attackers can also bribe or persuade stakeholders not to behave honestly or intentionally enforce a fork to reverse some of the transactions in the ledger. With this in mind, proof of activity takes the approach of following-the-satoshi which breaks down all active Bitcoins into the minimum unit satoshi and applies a pseudo-random value to it [4]. Then the algorithm searches the current holders of these satoshis through the transaction history. Assuming that there are ten coins in the system and 2 of these coins are randomly selected, if Bob has five coins and Alice has two coins, it is apparent that Bob has a higher probability of being the miner in the new round.

Rather than relying on just proof-of-stake or proof-of-work, 2-hop proposed a new chain structure combining both of these consensuses. Figure 2.13 illustrates the 2 hop chain, where the black outlined blocks represent the proof-of-work-based chain, and the red outlined blocks represent the proof-of-stake-based chain, but they are twisted with the constraints that a proof-of-work mined block can only be mapped to one proof-of-stake block in one round, and it has to link to blocks mined by both of the consensus rules [20]. To satisfy the constraints, the system performs proof-of-work mining and proof-of-stake mining alternately.

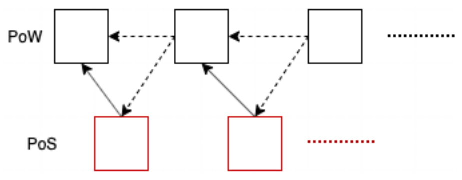


Fig. 2.13 2 Hop Blockchain Structure

This mining strategy secures the system if one side of the miners is malicious. For example, PoW miners record a fraudulent transaction in the block. However, the following PoS miners are aware of his malicious behavior and they decide to give up mining in this round. The proof-of-work miner cannot continue his attack because his proposed block did not accept the links from the PoS mined blocks. Hence, the system will not validate his proposal. This mining scheme reduces the risks of centralized computing and monetary power in a pure proof-of-work and proof-of-stake system.

2.2.3 Byzantine Fault Tolerance consensus

How to reach agreement in the decentralized and distributed blockchain network is a variant of the Byzantine General's Problem, which describes how a group of generals can reach agreement on a plan to either attack or retreat, however, because the generals are some distance apart from each other, they must rely on information being passed by messengers, who could be traitors sending erroneous messages[30]. A general makes a decision according to the messages he receives from the rest of the generals. It is assumed that all generals have to act the same to succeed in the war. In the computing world, this describes the tolerance of a distributed computing system when parts of it malfunction. Marshall Pease proved that a system with $3n+1$ nodes can tolerate up to n malicious nodes before total failure[41].

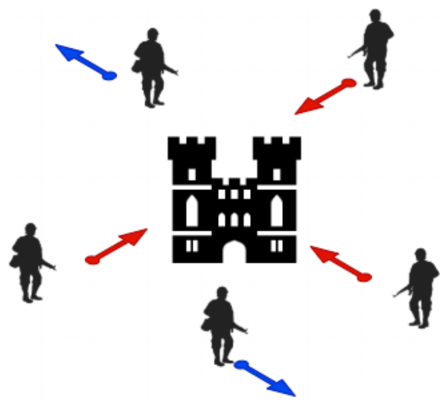


Fig. 2.14 Byzantine General Problem

Hyperledger Fabric is one of the early pioneers in developing blockchain applications based on Practical Byzantine Fault Tolerance(PBFT)[11]. It leverages the PBFT's safety assumptions that a distributed system with $3n+1$ can still work safely with up to n nodes being malicious or idle. In the Fabric network, each node joins as a validating peer. Within groups of nodes, the system selects one node as a leader, who is responsible for collecting transactions and creating blocks. In each round, nodes send transaction requests, verify transactions, and broadcast results to their peers. After a fixed amount of time has passed or the transactions are collected, each leader node broadcasts a block containing the unconfirmed transactions and collects the votes from the peers. If more than $2n+1$ nodes agree on the proposed block, it is finalized and appended into the blockchain. Similar to selecting a leader node in Fabric, the Honey Badger[36] deploys a group of master nodes to handle transactions and reach an agreement. Although the system is proved to handle a significant amount of transactions in short period, the set up of master nodes fails the nature of decentralization of blockchain.

Steall adopted the Federate Byzantine agreement, where nodes build up quorums based on their knowledge of other nodes' trust levels. Figure2.14 illustrates a tiered quorum structure. In the top tier, only one of nodes $v1, v2, v3, v4$ can be malicious. Otherwise, it will fail the security threshold specified by the Byzantine Fault Tolerant Problem. Since it depends on itself, it can only form quorum slices from this tier. In the middle tier, the nodes designate two nodes from the top tier to construct quorum slices. Nodes in the leaf tier depend on two nodes from the middle tier relying on the top tier. As the tiers and nodes are increasing, the trust hierarchy becomes more complex and robust.

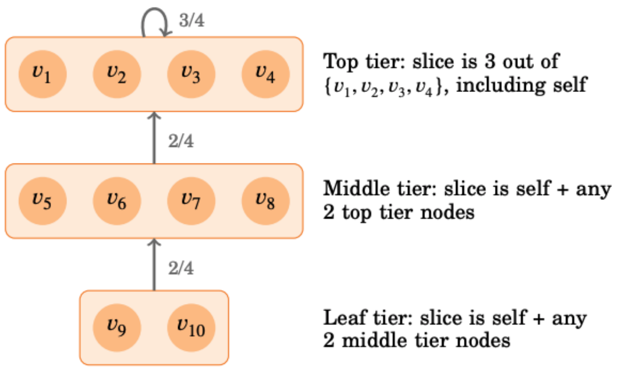

3/4
v_1 v_2 v_3 v_4
Top tier: slice is 3 out of $\{v_1, v_2, v_3, v_4\}$, including self
2/4
v_5 v_6 v_7 v_8
Middle tier: slice is self + any 2 top tier nodes
2/4
v_9 v_{10}
Leaf tier: slice is self + any 2 middle tier nodes

Fig. 2.15 Quorum Hierarchy[33]

2.3 Blockchain Trilemma

Blockchain has shown the public its potential to build a tamper-proof ledger system running without central authorities and challenging the traditional banking system. Many variations of blockchain have capitalized on Bitcoin's backbone protocol. However, Bitcoin's protocol has always been criticized for some of its apparent drawbacks, including security, scalability, centralized computing pools and energy waste due to the mining process. The above sections have analyzed some proposals to mitigate the drawbacks from different perspectives. Table 2.1 overviews and compares the consensus protocols.

The table shows that the variants of the proof-of-work consensus have limitations in terms of fulfilling every aspect of the expectation. Researchers call this the trilemma but blockchain applications can only achieve two of these[22]:

1. Decentralization
2. Scalability
3. Security

	Proof-of-Work	Proof-of-Stake	Byzantine Fault Tolerance
Computing Power	Intensive	Low	Low
Energy Cost	Moderate	Low	Low
Chain Fork	Possible	Difficult	Difficult
Double Spend	Possible	Difficult	Difficult
Pool Mining	Depending on mining difficulty	Hard to prevent	Master or leader nodes in some models cause centralization
Transaction Processing	Depending on mining, but most are impractical	Fast	Fast

Table 2.1 Consensus Protocol Comparison

Firstly, decentralization defines that a blockchain system runs in a fully distributed fashion, and none of the users can have dominant computing power to control the network. Most proof-of-work-based consensus fails to achieve this standard, especially those that require more than moderate level of powers. The concentrated mining pools have proven that leveraging a difficult computational puzzle is gradually causing centralization. Theoretically, a user occupying more than 50% of the network's total computing power can perform attacks on the ledger[52].

Second, scalability is always the most critical aspect of concern in a system which has a large number of users. Generally, scalability measures how well a system handles an increase in the number of participants and an increase in the consumption of computing power[10]. From the perspective of the participants, the blockchain system has a solid consistency in handling dramatically expanded diameters and the increased complexity of the network. However, its extremely low transaction throughput makes its adaptation in daily usage challenging. With the recommendation to observe six blocks mined after the block where transaction is confirmed, Bitcoin can only securely process 7 transactions per second[50]. In comparison to Bitcoin, centralized payment services intermediary claims to handle transactions from 2000 up to 56000 per second[46].

Last, a blockchain system should be secure from attacks that utilize computing resources to tamper with the ledger, such as double-spending and intentional forking chains[22]. So far, there have been no cases of attacks on a particular mainstream blockchain system. However, this does not guarantee a safe future. The majority of blockchains face potential threats from attackers. Specifically, either a proof-of-work or proof-of-stake-based blockchain is only safe under the assumption that 51% of the total computing power or monetary value is in the hands of honest users.

To conclude, it is apparent that blockchain systems have struggled to satisfy all properties in the trilemma. It is simple for a proof-of-work-based system to meet the requirement of full decentralization and to perform consistently with an increasing volume of the network. The most challenging part is to balance security and throughput. Most proof-of-work systems sacrifice throughput to ensure security by difficult computational puzzles. However, the concentrated mining power resulting from a moderate mining strategy also generates potential threats to the system.

In comparison, proof-of-stake-based blockchain mitigates the potential of computing power-based attacks and relies on a selection weighted by the user's tokens. However, it has been argued that proof-of-stake only shifts the competition from computing resources to monetary tokens. The system is still under the threat of individuals with too much power. Furthermore, the Byzantine Fault Tolerance system can ensure safety when a part of the network behaves dishonestly. However, some current applications which require leader nodes to take charge of the network are potentially causing centralization and it could expose the system to focused attack aimed at the leaders.

2.4 Algorand

The previous section explains that blockchains struggle to maintain a balance between decentralization, scalability and security. However, the newly proposed Algorand appears to break through the trilemma to build a secure, scalable and distributed ledger.

Algorand is a proof-of-stake blockchain utilizing a Byzantine Fault Tolerant-based consensus algorithm to maintain a public ledger without a central authoriy[23]. The consensus updates the ledger by randomly selecting representatives to propose blocks and perform consecutive rounds of voting to reach agreement. In order to validate a result in one step, it has to receive a minimum number of votes. Specifically, the threshold of validating a result has to be enough to secure the system when it is possible that there are malicious users attempting to stop the network from reaching agreement. A detailed process is be explained in the following paragraph.

At the beginning of each round, users privately perform cryptography sortition with a random seed and check if they are selected as a block proposer, according to binomial probability. Qualified users will collect pending transactions in the system and broadcast the proposed blocks. After new blocks are proposed, the system will start multiple rounds of the voting procedure to collect votes from the committee members on behalf of the entire network. Since the system leverages Byzantine Fault Tolerance to protect the ledger from malicious attack, those block candidates who receive more votes than the threshold in the final step will be confirmed and added to the blockchain.

The Algorand proposal has shown promising results that the network can reach consensus in a relatively short time and scale to a large number of users. Moreover, it claims that its proof-of-stake-based Byzantine Agreement eliminates chain fork with a negligible possibility of the occurrence of chain fork. As the majority of blockchain applications struggle to mitigate the side effects of chain forks and balance all three aspects of the trilemma, this impressive consensus opens a new chapter. In order to verify this outstanding result, an investigation and analysis of its consensus protocol is presented in the forthcoming chapters.

Chapter 3

Investigation of Consensus Algorithm

This section is dedicated to explaining the design of the simulator and investigating the consensus algorithms of Algorand. The simulator was built based on a discrete-event framework. The consensus algorithms were implemented according to the Algorand proposal[23] and modified to fit into the simulator. The simulator is programmed in Python.

3.1 Implementation of Simulator

3.1.1 Communication Model

Algorand nodes run in a decentralized peer-to-peer network, where nodes randomly establish connections with active nodes. Messages directly propagate through the communication pipe between two nodes. Each node either actively broadcasts messages or passively listens to its neighbours to participate in the network. Algorand spreads messages by leveraging the Gossip protocol, which requires nodes to broadcast incoming messages to their neighbors and their neighbors will broadcast the received messages to their neighbors. By repeating this process, the entire network will gradually receive the messages. Because there are no specific central servers that can handle an enormous number of requests and respond simultaneously, there are some downsides to this scheme, one being that propagation delay increases with the expansion of the network's diameters and complexity. Another drawback is that request and message flooding could temporally overload a node. Without a trusted authority and regulations, a short period of time with asynchronous updates can expose the node to potential Sybil attacks. Algorand specifies that an individual node has to verify messages before any action takes place and only broadcasts messages with a valid proof once to prevent message spamming[23].

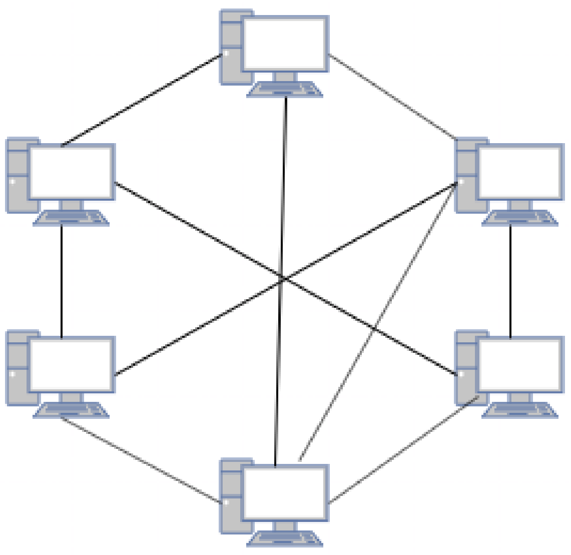


Fig. 3.1 Peer-to-Peer Network

Since no official simulator or source code had been released at the time of this investigation and as Algorand shares a similar network model and message size with Bitcoin, a discrete events-driven simulator with randomly sampled message propagation delays collected from Bitcoin was constructed. The Bitcoin network has been actively running with a large number of participants for ten years. Its propagation history, network complexity, and randomness should be a guideline to propagation delays in Algorand. Bitcoin core states that new connections are established only when a new node joins or an existing node is disconnected[13], and it was confirmed by Miller that the network topology remains relatively the same within a short period of time[35]. Therefore, the simulated network executes one round of Algorand's protocol with the same set of randomly sampled delays from Bitcoin's transaction delays. Randomly sampling delay data every round also satisfies the setup in Algorand's experiments, where each user replaces its peer after finishing one round.

In Algorand, there are three types of messages which play a significant role in forwarding the consensus. These messages individually contain block priority, block proposal, and votes. By referencing the experiments described in[23], a block priority message containing priority and sortition proofs is approximately 200 bytes. Compared with priority messages, the vote messages have some extra fields, storing a public key, a signature, and two hash values, which requires approximately 100 bytes. The size of these fields is estimated according to the Ed25519 encryption scheme specified in Algorand's cryptography library[18]. Similarly, Bitcoin transactions are, on average, 500 bytes in size, as observed from Bitcoin Visuals[6]. Although they share different sizes, Decker suggests that 96% of all transactions smaller than 1 KB are mainly caused by round-trip delays[16]. Hence, the size difference is negligible when sampling delays for priority and vote messages from the transaction delay data of Bitcoin. In the simulator, priority and vote delays are grouped into one category. Furthermore, the block proposal message is configured to be the same size as the block message in Bitcoin. But it can be tuned and approximated by adding 80ms delay for each kilobyte to a block over 20kB[16] to satisfy simulations with different sizes of blocks. These two kinds of delays are individually and randomly sampled from the delay data according to its probability distributions. In order to add more randomness, there is one more random shuffle within the delays before assigning it to the gossip pipes.

Moreover, two sets of Bitcoin delay data, shown in Figures 3.2 and 3.3, are not delays between the source node and its directly connected peers. It is collected from two monitors that connect all reachable nodes in Bitcoin. They track and collect the message delay time from the source node to all other monitored nodes. The delay data are calculated according to the time it was firstly announced and the time that the other nodes broadcast an INV message to notify the reception of that message. Therefore, it is a distribution of the time that one message takes to propagate to the entire network rather than delays between directly connected peers. However, the distribution in Figure 3.2 is not used to estimate block propagation delay because the Bitcoin network has a high-speed relay network to minimize the propagation of blocks between miners[5]. The Algorand proposal does not have the same strategy to propagate blocks. Furthermore, the proposal mentions that block messages take 5 seconds to reach the majority of the network and other small-sized messages take 1 second. In order to mitigate the effects caused by the sampled delays, these messages are respectively configured to reach 90% of the nodes in 5 seconds and 1 second. In the simulated network, each Algorand node has a connection with every other node instead of random connections to utilize the sampled data as reasonable delays.

```
Algorithm 1: $AssignDelay(self, bloc\_delay\_data, msg\_delay\_data)$
    //sample delay data;
    $block\_delay\_samples \leftarrow Random\_Sample(num\_nodes, block\_delay\_data)$;
    $msg\_delay\_samples \leftarrow Random\_Sample(num\_nodes, msg\_delay\_data)$;
    //construct pipe to simluate a message pipe;
    for $node$ in $Network$ do
        $pipe \leftarrow Gossip\_Pipe(self, node, block\_delay, msg\_delay)$;
        $node.pipes.append(pipe)$
    end
```

Algorithm 1 demonstrates the procedure of sampling data and constructing gossip pipes between nodes. Before running the consensus, each node performs random sampling to generate a set of delays, whose size is equal to the number of total nodes except itself. Then

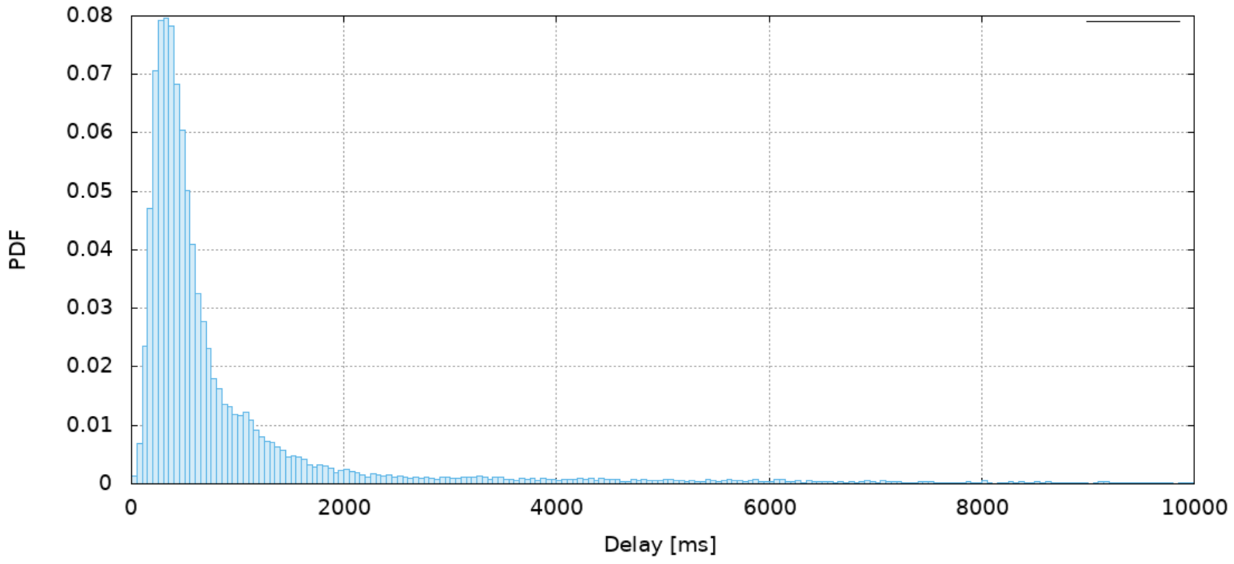


Fig. 3.2 Block Propagation Delay[19]

it initializes the gossip pipes to store the delay between itself and the rest of the nodes in the network.

Figure 3.4 shows the simplified process of gossiping and receiving the process in one Algorand node. The processor handles incoming messages and makes progress toward consensus. Whenever an event in the processor triggers message broadcasting, the sender unit will forward the message to the peers through the gossip pipes. To receive messages from the network, the listener runs as a background application that always listens to incoming messages. It will hold the message until its delay time has passed. This process stimulates transmission delay in the network.

Algorithm 2: Listener()

```
while $True$ do
    if $newMessages$ then
        $timer \leftarrow 0$;
        while $timer < pipe.delay$ do
            $timer += 1$//Internal clock ticks to delay the message processing;
            $timeout(1)$;
        end
        $recieved\_Msg.append(message)$
end
```

Algorithm 2 illustrates how the listener schedules delays for messages. First of all, the listener keeps running and is triggered whenever there is a new message sent through

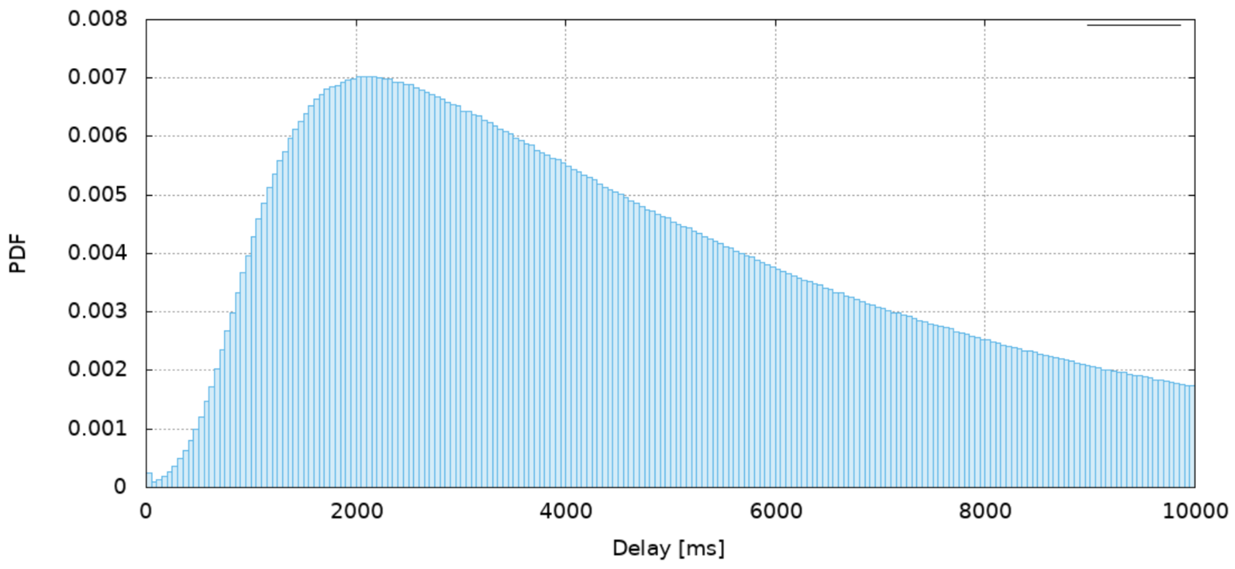


Fig. 3.3 Transaction Propagation Delay[19]

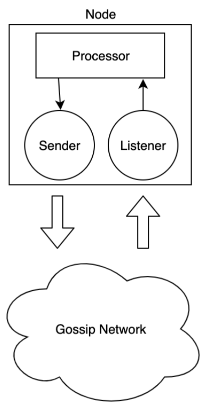


Fig. 3.4 Gossip Pipe and Receiver

the gossip pipe. For example, if there is a message that comes in at 0 seconds with a ten millisecond delay, the listener starts a timeout process to simulate the delay until the internal clock has ticked ten times. To clarify, one tick of the simulator's clock represents one millisecond. Moreover, rather than sleep the listener for ten milliseconds, ticking 1 millisecond by 1 can prevent itself from missing messages during the period of the timeout.

Algorand's consensus protocol consists of three phases, block proposal, reduction, and the Binary Byzantine Agreement. The result of each phase is e passed to the following one. Unlike proof-of-work-based blockchain, the result of each phase is determined by collecting votes from committee members except for block proposing. To participate in the network, Algorand users privately perform cryptography sortitions to check whether they can propose a block or vote for block proposals instead of competing with others to solve computational

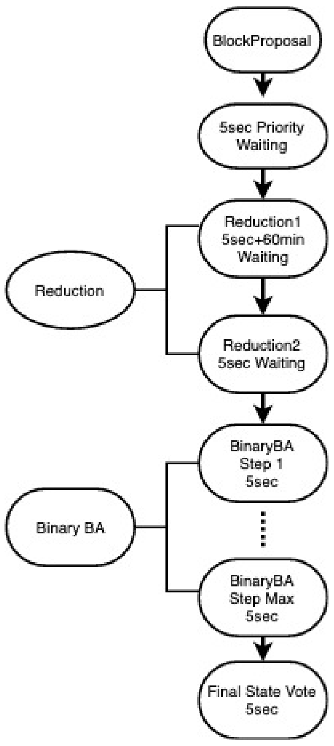


Fig. 3.5 Flowchart of Consensus

puzzles. Moreover, reduction and Binary Byzantine Agreement only validate the results with votes that pass the threshold. As long as the system satisfies the security assumption, no user can tamper with the result. In cases where a node hangs at one phase without enough votes passing the threshold, Algorand sets a maximum waiting time in every vote counting step. For example, if Bob does not observe any block hash with enough votes, he enters the next phase with an empty block hash after the maximum waiting time has passed. If Bob observes a block with enough votes with only 2 seconds, he immediately terminates the waiting and enters the next phase.

In Figure 3.5, the consensus starts with block proposing at the beginning of one round. Then, all users have to wait 5 seconds for the priority messages and block proposals to propagate. Users cannot enter the next phase unless they wait for 5 seconds . After learning who has the maximum priority, users have to wait for the corresponding block proposal to start the reduction. Because the block proposal is a much larger size than the priority message, it might take longer for all users to retrieve the block. Therefore, users who start Reduction1 first have a maximum waiting time of 65 seconds, where 60 seconds is the waiting time for the block to reach the majority of the network. The maximum waiting times of the subsequent steps are 5 seconds. In each step, a user either observes enough votes to skip some seconds of waiting or waits for the maximum seconds without enough votes and passes an empty hash to the next step. If a user reaches agreement first, he has to wait 5 seconds for the others to finish the round. This waiting applies to every user. The waiting mechanism indicates that Algorand's consensus time depends on the number of steps executed by individual users. Also, the execution of each user's step relies heavily on the synchronization of the network and interactions from other users. Assuming that the network is strongly synchronized and all users actively participate in the network, the best time for the entire network to reach consensus should be slightly longer than 10 seconds, which is the sum of the 5 second priority waiting time and the 5 second waiting time after observing final state of a candidate and some processing time.

From the above explanations, Algorand's consensus can be considered as an event and timing-triggered process. Therefore, the Algorand network is modelled based on a discrete-event driven simulator, which models individual users as processes, schedules events with delays, and interactions between processes. The simulator provides an estimation of Algorand's consensus performance with the randomly sampled delay data, fixed holding time mentioned above and the simulator's internal clock ticks.

The simulator starts one round of simulation by initializing a fixed number of Algorand nodes with equally distributed tokens. Algorand nodes occupy fundamental functions, including gossip pipes to send/receive messages in the simulated environment, asymmetric key pairs, cryptography sortition, chain ledger, and consensus protocols. The encryption scheme is not the focus of tihs investigation, and it only takes milliseconds to run one time. The verification of messages is replaced with sleep time approximated from the number of messages been processed. Furthermore, transactions are not involved with the consensus protocol because the proposers have stored transactions at the stage of constructing block proposals. If a dishonest proposer attempts to tamper with transactions in a block proposal, it will result in a change of block hash. Other nodes and the system can quickly notice the misbehavior. In the experiments, the default block size is 1 Mb, which is the same as Bitcoin's. Similar to the process of verification, there is a sleep time for block proposers to simulate the construction time. After configuring the delay data to each node, the simulator starts running all nodes simultaneously until the network reaches specific rounds of consensus or the entire network is stuck in a tentative state. Figure 3.6 displays an example of nodes running at the same time, and the double-sided arrows represent possible interactions with other nodes when they succeed in the cryptography sortition for either the block proposer or the committees. As soon as the simulation starts, the internal clock starts ticking to track the time consumption.

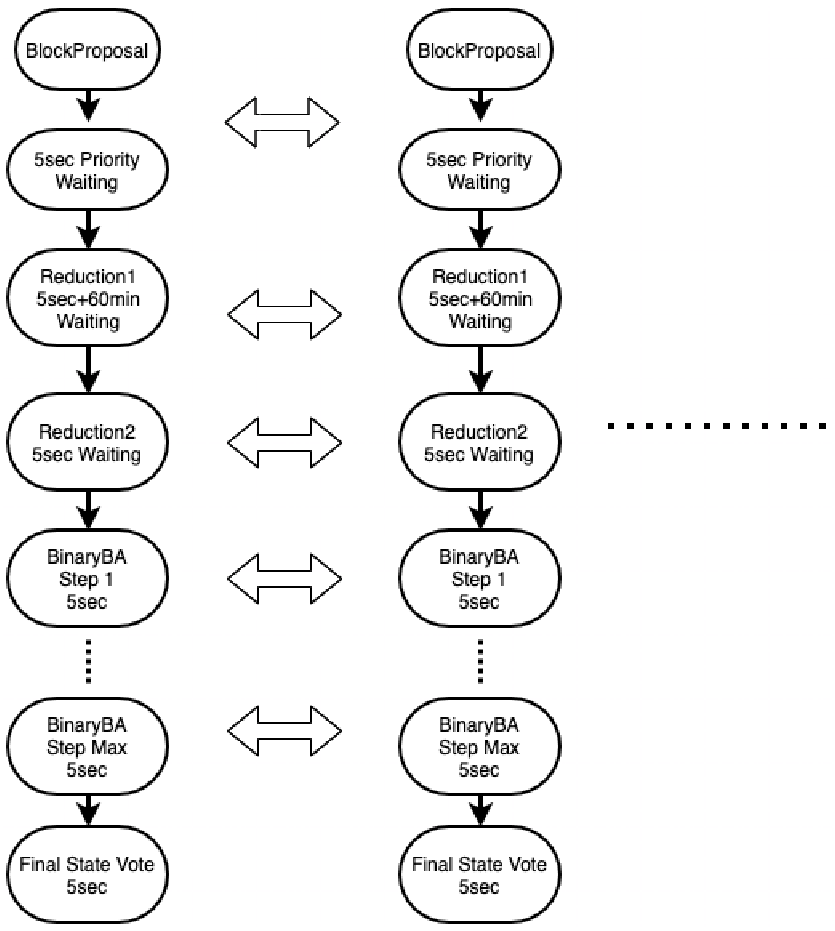


Fig. 3.6 Interaction Demo

3.2 Implementation of Consensus Algorithms

3.2.1 Security Assumption

The security of Algorand applies the Byzantine Fault Tolerance model to ensure that at least 2/3 of tokens are held by honest users. With this assumption, the system can make progress toward consensus even with malicious interference. Therefore, Algorand specifies that an adversary can bribe a small proportion of honest users, but it does not allow more than a minimum threshold-specified number of honest users. Otherwise, the first safety assumption will not be satisfied and will result in complete failure. Furthermore, it also declares that the system runs in a bug-free client. With these assumptions, simulated malicious users are modeled to perform attacks with valid sortition rather than brutally spamming the network or forging honest users' key pairs[23].

3.2.2 Blockchain

The blockchain follows the typical design used by the majority of the applications. In the simulation, each block has a pseudo-random block hash generated by hashing the concatenation of metadata. The SHA-256 hashing scheme produces a fixed-length collision-resilient result, which satisfies the ideal hashing primitives in Algorand[12]. The rest of the fields include a round number indicating the block height, a previous block hash, and the block proposer's verifiable random function(VRF) hash result which is used as a random seed for the next round's sortition.

As stated in the previous section, the investigation focus on the consensus algorithm and transactions was not implemented. Hence, the blockchain is a minimal design to supply the necessary parameters to the consensus algorithm.

3.2.3 Cryptography Sortition

Verifiable Random Function

Algorand constructs cryptography sortition based on the VRF which generates a pseudo-random hash and a public proof of its correctness. In the following equation, the VRF function takes a secret key with a given string X as input and produces a hash value and proof.

$$< hash, \pi > \leftarrow VRF_{SK}(X)$$

The hash value cannot be recreated without knowing the secret key. However, using the public key of the corresponding secret key and proof can verify whether the hash is the correct output. In Algorand, users privately perform sortition and reveal the hashing results with the public key for verification without exposing their secret keys. This property satisfies the asymmetric requirement of blockchain applications. The only way for attackers to forge a validated proof is random guesses which use an enormous amount of computing power. However, the fast-growing chain will make this kind of attack expensive and infeasible. This function plays an essential role in maintaining the security of the system and supporting the sortition algorithms.

Cryptography Sortition

In cryptography sortition, the input string is obtained by concatenating a publicly known random seed and role parameter. In order to make the seed random and publicly known, block proposers have to generate the hashing value by concatenating the last round's seed with the round number and store it in the proposal. The seed for the next round will only be determined when the network reaches a consensus on one of the block proposals. Seed generation is shown as follow:

$$< seed_r, r > \leftarrow VRF_{sk}(seed_{r-1}||r)[23]$$

Moreover, the role of the parameter limits the purview of the user's actions for each sortition result in one round. Users with an intention to participate in every step of the consensus have to perform the sortition with different role strings. Due to the pseudo randomness and collision-resistance of VRF functions, changes of input could significantly affect the result. This indicates that users selected as block proposers might not be qualified to be committee members caused by a different role parameter as input. It reduces the possibility of some users dominating an entire round of block proposing and voting. Performing sortition multiple times in one round also gives users who only own a relatively small number of tokens the chance to participate in the protocol.

The user starts sortition by computing the hash and proof of the VRF function with its secret key and the input string as previously explained. Then the sortition randomly selects a group of sub-users according to a binomial probability of an expected number over the total. The sub-user is defined as the minimum unit of Algorand's token. If there are ten tokens in total in the system, a user who owns three tokens will have three sub-users. The expected number of sub-users depends on how many proposers or committee members are expected to ensure safety. From the perspective of the entire system, random selection can be understood as the problem of coin-flipping. In the view of an individual user, the selection can be considered as how many sub-users out of w tokens he owns could be selected while τ sub-users are expected from W total tokens in the system.

Following the binomial distribution, the number of selected sub-users of an individual user owning w units of tokens has a probability of:

$$B(k;w,p) = \binom{w}{k} p^k * (1-p)^{w-k}, k = selectedsubusers$$

Then, the sortition cuts the user's probability distribution into w continuous intervals:

$$[\sum_{k=0}^{j} B(j;w,p), \sum_{k=0}^{j+1} B(j;w,p)), j \in 0,1,...w[23]$$

,

The last step in Algorithm 3 determines the number of selected sub-users according to the value of the division result of the VRF hash over the maximum value of $2^{hashlen} - 1$. The pseudo-random result from VRF is expected to be uniformly distributed within the interval of $[0, 2^{hashlen} - 1]$. The probability interval in which it falls determines how many sub-users are selected.

Algorithm 3: Sortition$(sk, seed, \tau, role, w, W)$[23]

```
$< hash, \pi > \leftarrow VRF_{sk}(seed||role)$ ;
$p \leftarrow \frac{\tau}{W}$ ;
$j \leftarrow 0$ ;
while $\frac{hash}{2^{hashlen}} \notin [\sum_{k=0}^{j} B(j; w, p), \sum_{k=0}^{j+1} B(j; w, p))$ do
    j++
end
return $< hash, \pi, j >$
```

- sk: user's private key
- Seed: random string stored in the last round's block
- τ: the expected number of proposers or committee members
- role: string to distinct block proposer or committee member
- w: user's total tokens
- W: total tokens in the system
- j: the number of selected subusers

It is possible that one user can have multiple sub-users selected by the sortition. Therefore, Algorand states that the number of selected sub-users will be the number of votes in the voting step. Moreover, the concatenating sortition hash and the index of sub-users will produce a priority for each block proposer. By referring to this priority, users will discard unnecessary messages and determine the first candidates to vote for in the subsequent Byzantine Agreement protocol.

3.2.4 Byzantine Agreement$\star$

Algorand's Byzantine Agreement $\star$ is the consensus protocol that guides users to learn about the current state of the public ledger in a distributed network. As shown in Algorithm 4, it consists of three components, which are Reduction, Binary Byzantine Agreement, and Voting. The first two algorithms contain a fixed number of sub-procedures executed in consecutive

Algorithm 4: BA$\star(ctx, round, block)$[23]

```
$hblock \leftarrow Reduction(ctx, round, H(block));$
$hblock_{\star} \leftarrow BinaryBA\star(ctx, round, hblock);$
//Check if "fina" or "tentative" consensus is reached;
$r \leftarrow CountVotes(round, FINAL, T_{Final}, \tau_{Final}, \lambda_{STEP});$
**if** $hblock_{\star} = r$ **then**
    **return** $< Final, BlockOfHash(hblock_{\star}) >$
**else**
    return $< Tenative, BlcokOfHash(hblock_{\star}) >$
**end**
```

order. Each sub-procedure validates a candidate depending on the result of its voting. The result of each procedure will determine the subsequent procedure's input.

Before starting the main procedure of the Byzantine Agreement⋆, users have to collect the block proposal with the highest priority to the best of his knowledge and verify the contents. Otherwise, it starts with an empty block. Byzantine Agreement⋆ starts the first sub-process Reduction with the hash of the block proposal. This process eliminates redundant block proposals and generates at most one non-empty hash result. Then, BinaryBA⋆ takes the Reduction result and performs votes to decide if the network agrees on the non-empty proposal or an empty proposal.

After finishing the Reduction and BinaryBA⋆, users need to check if the entire network has agreed to finalize returned block hash. If votes for the final state exceed the threshold, the agreement will be reached and a new round will start. Otherwise, a tentative state will be reached. Because Algorand did not specify how the node will recovery from the tentative state, users reaching a tentative state in the simulator will keep counting votes until it observes a valid result to push itself into a new round.

3.2.5 Voting

In Byzantine Agreement⋆, a valid result has to receive a more than a threshold-specified number of votes from committee members. The first step of the voting process is to randomly select a portion of users to represent the entire network. In each procedure that involves voting, users initiate the Committee Votes in Algorithm 5 with a privately performed cryptography sortition. The result will determine whether the user could participate in a specific step. To distinguish hash result in different steps, sortition requires a unique role parameter, which is a concatenation of a fixed role parameter, round number and step. Changing role string increases the variety of users to participate in the voting. It can prevent a fixed group of users to influence on the result of voting continuously. The number of selected sub-users obtained from sortition will be the number of votes that a user can assign to a voting message. Generally, if a user has j sub-users selected as committee members, he puts all his votes on the result he observes in the last step. Because the majority of the network has learned about the block at the stage of proposing, qualified users construct vote messages based on the block hash rather than the entire block. Hence, vote messages are relatively small in size, which enables vote messages to reach the entire network within a short period. With the block hash, a parameter ctx storing the round status is also passed into BA⋆. It binds the vote message to a specific step to perform cryptography sortition and verification.

Algorithm 5: CommitteeVotes($ctx, \tau, block_hash$)[23]

```
$role \leftarrow <\text{"}committee\text{"}, round, step>$;
$< sorthash, \pi, j > \leftarrow Sortition(user.sk, seed, \tau, role, w, W)$;
//only committee members construct votes and messages;;
if $j>0$ then
    $Vote\_Message(user.pk, ctx.round, ctx.step, block_hash, votes)$;
    $Gossip(Vote\_Message)$
end
```

The second step of voting is to process the messages and count the votes for each candidate. Most of the time, users start the subsequent CountingVotes immediately after executing CommitteeVote in a specific step. During a fixed period λ , CountVote reads the unprocessed vote message from a buffer. As explained in the previous section, a listener is always updating new vote messages to the buffer. Before appending a new vote value, readMessage checks the round and step of the messages and only returns correct ones. As soon as there is a block hash with more than $T \times \tau$ votes, the procedure terminates and passes the result to the following step. To clarify, T represents the minimum threshold to ensure the security of the Byzantine Fault Tolerant system. The threshold value may vary at different conditions, but it is commonly set to be $2/3$. Parameter τ is the expected committee size used in the sortition. These two parameters apply to almost every Byzantine Agreement⋆ step except the final step. In contrast, if a user cannot observe any result to satisfy the minimum threshold within the specified time λ, it will return a timeout parameter. The specified time λ prevents users from hanging forever at a particular step if there are not enough votes to push it above the threshold.

Algorithm 6: $CountVote(round, step, T, \tau, \lambda)$[23]

```
$timer \leftarrow 0$;
$votes \leftarrow \{\}$ //hash table store votes;
$vote\_msgs \leftarrow received\_Vote\_Msg[round, step].iterator()$;
while $timer < \lambda$ do
    $m \leftarrow vote\_msgs.next()$;
    $timer += 1$ //internal clock ticks in simulator;
    if $timer > \lambda$ then
        return $Timeout$
    else
        $< round, step, bhash, votes, voter > \leftarrow readMessage(vote\_msg)$;
        $counts[(round, step)][bhash] += votes$;
        for $bhash in counts[(round, step)]$ do
            if $bhash.votes > T * t$ then
                return $bhash$
            end
        end
    end
end
```

3.2.6 Reduction

Before starting the Reduction, users have to collect the proposal with the highest priority to the best of his knowledge to confirm the contents in the block. Due to the random topology of the Gossip network, there is no guarantee that block messages can reach every user with minimum delay. A small fraction of nodes might run behind the majority of the network and does not have the full knowledge of the priorities. These users might vote for a block candidate different from the majority of the network. To filter out the redundant block hashes in the voting procedure, Reduction in Algorithm 7 limits the proposed candidates to one with two consecutive rounds of voting.

Reduction initiates the first round of voting to collect the candidate for whom at least $2/3$ of committee members voted. There will be an extra λ_{BLOCK} waiting time in the first counting to leave enough time for block messages to spread in the network. Furthermore, an empty block is introduced here to prevent the consensus from not progressing when none of the block proposals are getting enough votes due to attack or network latency. After the first sub-Reduction, another round of voting is called to vote for the first round's result. The second round serves as a double-check to ensure the majority agree with the result of the previous step. As a result, there could only be one non-empty block hash to pass into the next phase [23].

Algorithm 7: $Reduction(ctx, round, hblock)$[23]

```
//step one;
$CommitteeVote(ctx, round, REDUCTION\_ONE, \tau\_step, hblock)$;
//some nodes might still waiting for block proposals, timeout extra time;
$hblock_1 \leftarrow CountVotes(round, REDUCTION_{ONE}, T_{STEP}, \tau_{STEP}, \lambda_{STEP} + \lambda_{BLOCK})$;
//step two;
$empty\_hash \leftarrow Empty\_Block(round, last_b lock)$;
if $hblock_1 = TIMEOUT$ then
    $CommitteeVote(round, REDUCTION\_TWO, \tau\_step, empty_{hash})$
else
    $CommitteeVote(round, REDUCTION\_TWO, \tau\_step, hblock_1)$
end
$hblock_2 \leftarrow CountVotes(round, REDUCTION_{TWO}, T_{STEP}, \tau_{STEP}, \lambda_{STEP})$;
if $hblock_2 = TIMEOUT$ then
    return $empty\_hash$
else
    return $hblock_2$
end
```

3.2.7 Binary Byzantine Agreement

BinaryBA$\star$ shown in Algorithm 7 determines which candidate from the Reduction step will be finalized as the new block. The algorithm iterates three consecutive voting steps within a specified maximum, guaranteeing a safe consensus can be reached while the network could be partially controlled by attackers. The rest of this section explains the algorithm in different scenarios.

Algorithm 8: BinaryBA⋆(round,block_hash)[23]

```
$step \leftarrow 1$ ;
$r \leftarrow block_hash$ ;
$empty\_hash \leftarrow Empty\_block(round, last_block)$ ;
while $step < MAXSTEPS$ do
    $CommitteeVote(round, STEP, \tau\_step, r)$
    $r \leftarrow CountVotes(round, step, T_{STEP}, \tau_{STEP}, \lambda_{STEP})$;
    if $r$=$TIMEOUT$ then
        $r \leftarrow block\_hash$
    else if $r \neq empty\_hash$ then
        for $step < s\prime < step + 3$ do
            $CommitteeVote(round, s\prime, \tau\_step, r)$
        end
        if $step$ =$1$ then
            $CommitteeVote(round, FINAL, \tau\_step, r)$
        end
        return $r$
    step++;
    $CommitteeVote(round, step, \tau\_step, r)$
    $r \leftarrow CountVotes(round, step, T_{STEP}, \tau_{STEP}, \lambda_{STEP})$;
    if $r$=$TIMEOUT$ then
        $r \leftarrow empty\_hash$
    else if $r = empty\_hash$ then
        for $step < s\prime < step + 3$ do
            $CommitteeVote(round, s\prime, \tau\_step, r)$
        end
        return $r$
    step++;
    $CommitteeVote(round, step, \tau\_step, r)$
    $r \leftarrow CountVotes(round, step, T_{STEP}, \tau_{STEP}, \lambda_{STEP})$;
    if $r$=$TIMEOUT$ then
        if $CommonCoin(round, step, \tau_{STEP}) = 0$ then
            $r \leftarrow block\_hash$
        else
            $r \leftarrow empty\_hash$
        end
    step++
end
```

Strong synchronization

Strong synchronization assumes that the majority of users in the network are honest. Furthermore, messages sent by honest users can propagate to the entire network within a known delay. With these assumptions, users perform BinaryBA$\star$ with the result from the Reduction step. Because the majority of users in this scenario are honest, , the Reduction step should return a non-empty block hash, which is likely to be proposed by the proposer with the highest priority. Therefore, honest users vote for and observe the same result in the first step of BinaryBA$\star$. An extra vote that confirms it as the final consensus is required after BinaryB$\star$. The final state has a higher threshold and a larger committee size than the other steps. If there are enough votes to push the candidate past the threshold, this indicates that a significant fraction of the complete network obtains the same block hash and confirms its validity. As a result, the network reaches a final consensus on the block hash.

In situations when a small proportion of users timeout in the first step, users who return have to conduct votes for the following three steps with the same result returned in BinaryBA$\star$. By doing this, users who cannot return in the same step will not fail at the counting in the additional steps due to a low participation rate. Because the threshold numbers are the same in every step, if a large number of users return in a particular step, it will be impossible for the rest of the users who do not observe the same result to collect enough votes in the subsequent steps. Otherwise, these users will spend time meaninglessly repeating the counting. Conducting an extra three rounds can push more users to return a non-empty hash in the fourth step of BinaryBA$\star$.

For instance, if a user times out in the count vote of the first step, he has to check this value again in the second step instead of starting with an empty hash. This checks whether the majority has returned a non-empty hash in the first step. After observing enough votes for the non-empty hash in the second, third and fourth steps, the user returns a non-empty hash in the fourth step which is the first step of the new iteration.

Weak synchronization

Weak synchronization is a situation where some adversaries try to interrupt the voting process and stop the network from reaching agreement or the message propagation cannot reach the entire network within an ideal interval. In a scenario of strong synchronization, the network can observe the final state immediately after the first step because the majority of users are honest and voting for the same block hash in both steps. The system will still work without an extra final vote. However, directly returning the result without calling an extra final vote will generate a diverged version when the network is weakly synchronized.

Let us assume that there are two groups of users with different latencies, where one can receive messages with acceptable delays. In contrast, the other group is not able to collect enough votes during the counting process. In this case, the first group might finish BinaryBA$\star$ with a non-empty block hash, but the other group cannot obtain enough votes to pass the threshold value and eventually returns an empty block hash value. Without the additional final vote, this will result in a diverged ledger. Similarly, a malicious attacker with enough votes can intentionally push one of the block hashes to obtain more than the threshold votes when the entire network is divided.

The additional final state voting can address this issue. As soon as one of the two groups obtains the result, they start the final state voting with a larger committee size to confirm agreement. Because the network is divided and votes for different values, there will not be enough votes to push either of the two values to surpass the threshold in the final vote. Algorand specifies this as a tentative consensus, which indicates that the system fails to reach a safe consensus due to the asynchronous network or malicious behaviors. In contrast, the strongly synchronized network should reach the final state shortly after returning a block hash in the first step of BinaryBA$\star$. Once the final state is observed for one candidate, there will not be enough votes for the other one to exceed the threshold.

If BinaryBA$\star$ starts with an empty hash, the original algorithm does not return it in the first step, even if it receives enough votes. The first step only returns and calls final on non-empty hashes. An empty hash needs the second step to receive enough votes and return it. There is no final vote called for empty hashes. Empty hashes will only be marked as the tentative state. Since a recovery protocol was not implemented in this research, there will be a difference in handling an empty hash. If the majority of users enters BinaryBA$\star$ with an empty hash and manages to return in the second step, they will check the count result of the first three steps. If the empty hash receives more than the threshold votes in all three steps, the final vote would be called on the empty hash. The simulator considers this case as a final state because the empty block does not pose a threat to the ledger. Users can start proposing and voting for a new block in the next round. However, if there are three consecutive rounds of reaching consensus on empty hashes with only two steps, the epoch of simulating will end. This indicates that the network needs a reboot to achieve faster message propagation.

Split of Vote

The above case mainly addresses issues caused by network latency or minimum inference from attackers. This section explains how the BinaryBA$\star$ algorithm ensures safety in a heavily asynchronized scenario, where messages propagate with dramatic delays or attackers control a significant proportion of the entire network.

With the condition that honest users are divided into two groups to obtain a different count result in step 1 of BinaryBA⋆, there is an attacker who can push either of the two groups to have more than the threshold votes which interferes with the user's subsequent counting process by manipulating the time at which the messages are sent. Without the attacker, honest users will gradually learn about the messages within a few iterations and reach a consensus on one of the block hashes. However, the attacker intentionally only sends vote messages to the group voting for the empty hash and lets the other group timeout in the CoutVote. As shown in Algorithm 8, the group which has timed-out in step one will continue with the non-empty hash again in the second step. Meanwhile, the group with the empty hash still votes for the empty hash because the attacker pushes them to observe enough votes to proceed to the second step without timing out. As long as the attacker can manage to sort enough votes, he can stop the network from reaching consensus by repeating the same action.

In order to protect the protocol from this attack, Algorand introduces a common coin that makes all users accept a random binary value in step 3 of BinaryBA⋆. As shown in Algorithm 9, a common coin will sort out the minimum hash value generated from hashing the concatenation of the committee's sort hash and indexes of the sub-users from all the received vote messages in the third step. This will return the last digit of the hash value, which is either 0 or 1. The binary values represent the non-empty block hash and empty block hash individually. If the lowest hash is from an honest member, all users should observe the same digit. Then, in the next step, all users should vote for the same result. This will recover the network from diverged voting and return within the next few steps.

Algorithm 9: CommonCoin($ctx, round, step$)[23]

```
$minhash \leftarrow 2^{hashlen}$;
for $msg \in received_{V}ote_{M}sg[round, step]$ do
    $< votes, sorthash > \leftarrow readMessage(msg)$;
    for $1 \leq j < votes$ do
        $h \leftarrow H(sorthash || j)$;
        if $h < minhash$ then
            $minhash \leftarrow h$
return $minhash$ $mod$ $2$
```

Chapter 4

Analysis & Findings

In this chapter, the results from the simulation of Algorand's performance are presented. The simulation focuses on three major aspects, including the completion time of one round with users of different sizes , stability with different percentages of dishonest users, and the distribution of the sortition results with unevenly weighted users. Following the result, an analysis of the observations from the results will be presented.

4.1 Round Completion Time

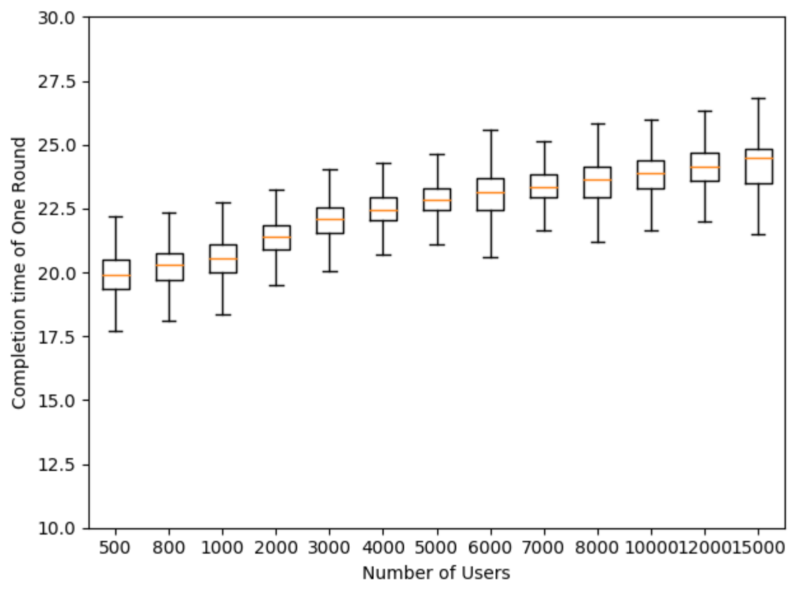


Fig. 4.1 Round Completion Time

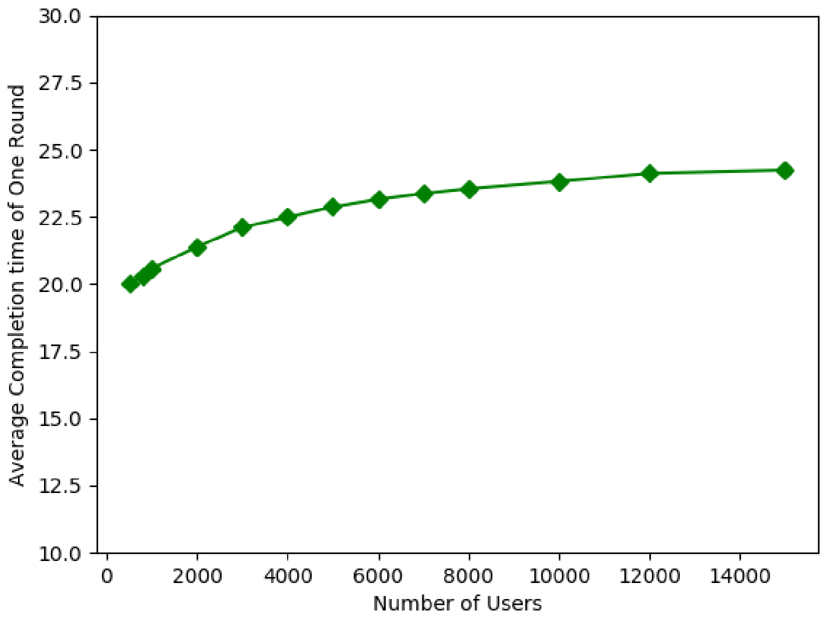


Fig. 4.2 Average Round Completion Time

The boxplot in Figure 4.1 demonstrates the minimum, three-quarter, and maximum time for users of different sizes to reach agreement. In addition, Figure 4.2 displays the average completion time per round of the corresponding setups. The completion time gradually increases with the size of the simulated users until the number approaches 10,000. When the number surpasses 6000, the average time starts to flatten, as shown in Figure 4.2.

Completing one round in Algorand consists of multiple timing windows. The first is a mandatory waiting period of ten seconds which allows priority messages to propagate in the network and wait for other users who have not finished the last round. This ten-second window ensures the propagation of priority message sent from qualified block proposers to reach the majority of users and allow a proportion of participants to receive block proposals. Secondly, there are two essential waiting processes that could be terminated once certain conditions are satisfied, these being waiting for block proposals and vote messages. In the waiting period for block proposals, the nodes who received the highest priority message cannot proceed to the voting phases without receiving and validating the entire block proposal. This helps eliminate the possibility of blocks containing repeated transactions or intentional double-spend attack. After receiving and confirming the block, the users can terminate the waiting period and enter the Byzantine Agreement phase of the consensus. In each step of the Byzantine Agreement, there is a waiting window which allows users to construct, broadcast and process vote messages to validate the most popular candidates.

As the size of a vote message is small, the majority of users can collect enough votes to validate a block proposal within seconds. Users immediately proceed to the subsequent step without wasting time on unnecessary waiting. Due to the random network connection scheme, a small proportion of users might not receive enough vote messages to validate any of the proposals. These users enter the next step of voting for an empty hash when the maximum waiting time finishes. This could prevent them from spending a large amount of time at a particular step. Although they have wasted time at the previous step, they can catch up to the other users by spending less time waiting for messages from committee members in the subsequent step. The flexibility of the time window is crucial to maintain the efficiency of the consensus.

In the worst-case scenario, the user will timeout in every step which will require establishing new connections. This will not have a significant impact on the system unless more than 20% of the weighted users suffer from connection issues. If this happens, the network will reach consensus on a tentative state and run a recovery protocol to boost synchronization.

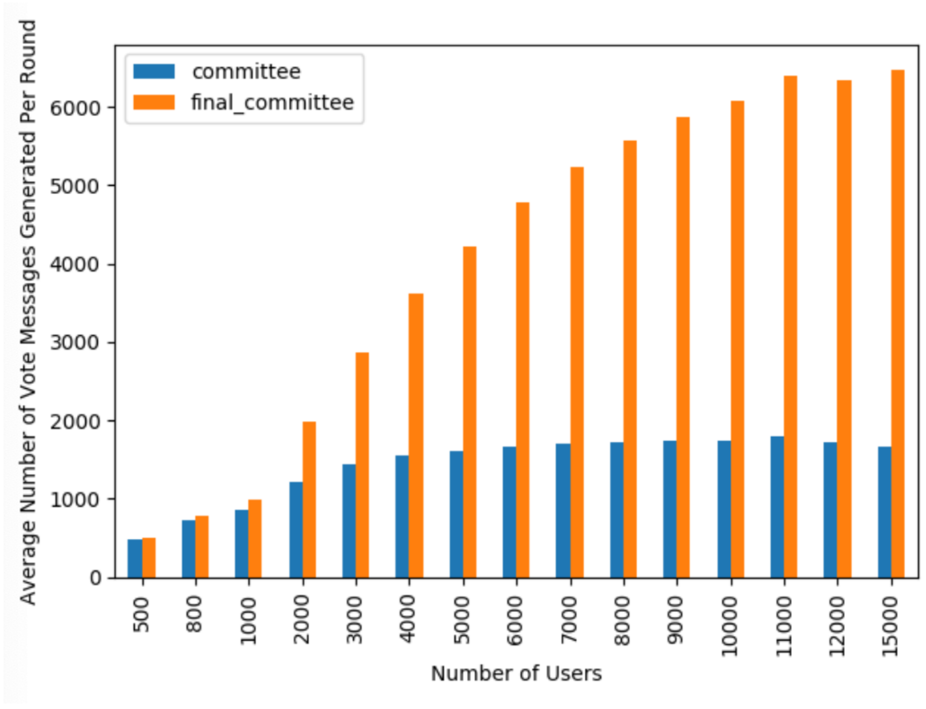


Fig. 4.3 Number of Vote Messages Generated

In addition to the aforementioned waiting periods which significantly influence the latency of the message propagation of the communication protocol, the verification of the vote messages also affects the time to complete one round of consensus. The expected number of committee members in each voting step is configured as constant in the simulation, where the final step is 10000, and the other steps are 2000. Figure 4.3 shows the average number of vote messages generated in each round with a different number of simulated users. The blue bar represents the vote messages in the Reduction step and binary BA, which approaches 2000 and remains constant when the number of users is larger than 2000. The reason for this is that the number of simulated users is smaller than the committee size. Some of the users have multiple sub-user which succeed in one sortition. Consequently, there are less vote messages with higher weights. A similar observation is obtained in the vote messages in the final step. However, the number of messages did not max out when the size of the users reaches 10000. This is caused by the configuration of tokens assigned to each user. The simulator assigns a fixed number of tokens to the users instead of evenly distributing tokens to users from a fixed set of total tokens because of the limitations of the RAM.

Although the limitations of the simulator restrict the scaling of users to an even larger number, the results still show that the protocol is capable of reaching consensus in a constant time while the number of users is increasing. The main factor affecting the speed of reaching agreement is the propagation of the blocks when the diameter of the network expands to be broader and becomes more complex. The size of the block mostly determines the propagation time if users have a decent network bandwidth. If the block can reach the majority of users who have a high probability of becoming committee members, agreement can progress faster in subsequent steps. Furthermore, it might not be wise to increase the waiting window to achieve a higher throughput of consensus by increasing block size. The increased idle time could increase the possibility of attackers performing brute force attacks. Also, the completion time of small-sized groups of users might not reflect performance in the real world because the Gossip protocol depends on each node broadcasting received messages to its neighbours[17]. The network connection required to propagate messages among 10000 users is more complicated than small-sized groups. The same-sized messages can reach the majority of the network with less routing and latency, which should generate a faster round completion time than the observation from the simulator.

Apart from the limitation of scaling the number of simulated users, the simulator is not able to simulate the consensus procedure of other blockchain applications. Unlike other categories of consensus algorithms, Algorand's consensus is mainly triggered either by satisfying certain criteria or surpassing a maximum waiting time. The system is always making progress. However, proof-of-work based consensus stops growing its ledger if there is no one can find a correct answer to the computational puzzle, which requires much more computing power than simulating Algorand. Moreover, timing of solving the computational puzzle and propagation significantly affects the result of consensus. The design of timer in the simulator could not provide a reasonable estimation to it. Therefore, the simulator is not used to simulate other consensus algorithms. Moreover, the main net of Algorand went online only for about 6 months. The team is also upgrading new protocols to the published application. The average round completion time calculated to be around 4 seconds. However, the outstanding performance at its current early stage can hardly be taken into account. From the observation of its ledger, newly generated blocks rarely contain transactions and the number of active users in the system also remains at a relatively low level. Especially, the active level is far behind some well adopted blockchain applications, including Bitcoin, Ethereum, Peercoin and others, which have numerous participators and handle high volumes of transnational activities. Therefore, a more substantial number of users engagement are needed to compare its performance of scalability with other categories of consensus algorithms.

4.2 Resistance to Dishonest Voting

The Algorand consensus algorithm can reach safe agreement while malicious users control up to 20% of the total tokens. In order to demonstrate its performance with interference from dishonest users, the simulator configures the weighted fractions of dishonest users who coordinate with each other to vote against the honest majority. The total number of users simulated in this experiment is 5000. In Figure 4.4, the x label shows the different proportions of the total number of tokens assigned to the dishonest users varying from 0% to 20%. The average completion time of one round demonstrates a linear increment in the completion time with an increasing percentage of tokens owned by dishonest users. The increment in the completion time is mainly in the vote-counting step. As the number of weighted tokens gradually increase, the probability of a dishonest user becoming a committee member increases at the same time. Because dishonest users broadcast the vote messages against the majority, honest users have to spend more time counting unnecessary messages from the dishonest users and waiting for vote messages from honest users to validate a proposal. Although the dishonest users manage to delay agreement for a few seconds, it does not have a significant effect on the safety of the consensus. Neither empty blocks nor forks of the chain are observed within the simulated rounds. However, the experiments can only reflect the stability of the consensus with the assumption that dishonest users can only perform attacks leveraging their advantages of sortition probability, and honest users are all active in the network. This raises the concern as to how Algorand ensures security if the weighted proportion of active users cannot reaches at a certain level. This issue will be discussed in the next section.

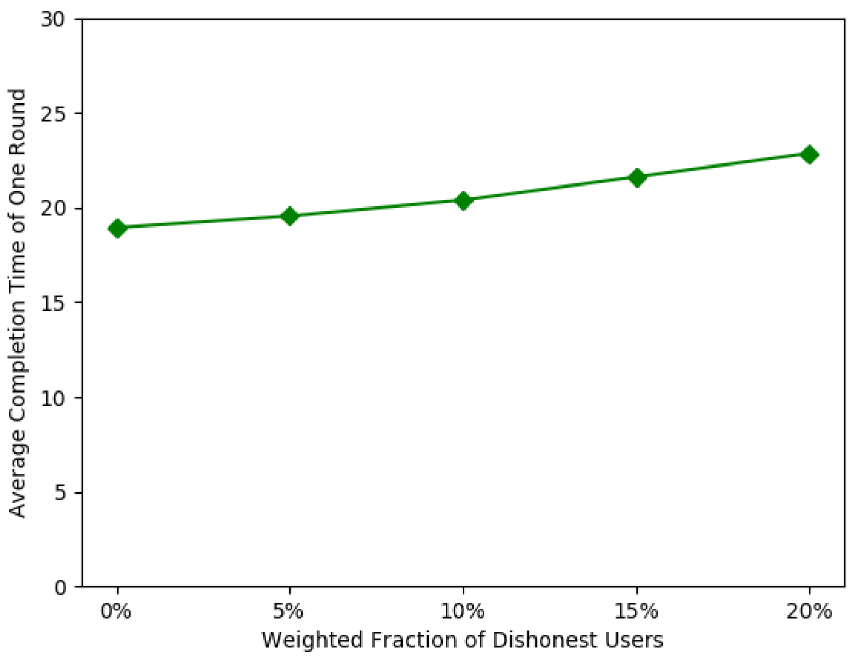


Fig. 4.4 Completion Time of Varying Proportions of Dishonest Users

4.3 Sortition

Different from proof-of-work based blockchain, Algorand does not rely on a group of miners to solve computational puzzles to construct new blocks. It utilizes a sortition algorithm that randomly selects a fraction of users weighted by their tokens to propose blocks and agree on one of them through multiple rounds of voting. The safety of the consensus relies on the fact that stakeholders investing significant amounts of money will not misbehave intentionally and pose potential threats to their assets.

It is undeniable that combining the randomness of VRF[34] and binomial probability is an excellent solution to select a subset of representatives according to their monetary weights in the system. Every user attempts sortition with the same success rate based on binomial probability while the number of trials is equivalent to their tokens. The cryptographic sortition generates continuous intervals starting from 0 to 1 based on the binomial probabilities of different successes, which is shown in equation 4.1. Then, it divides the pseudo-random hash value produced by VRF by the maximum value of its hash length. The interval into which it falls will determine the number of votes of that user. As long as the VRF is secure against Sybil attacks, malicious users cannot ensure that a user will become a committee member in a particular step unless it knows the secret key to obtain the hash value revealed in the vote message in advance.

$$[\sum_{k=0}^{j} B(j;w,p), \sum_{k=0}^{j+1} B(j;w,p)), j \in 0,1,...w \tag{4.1}$$

However, biased probability might generate some potential issues. The first is that users who own a small number of tokens might find it challenging to succeed in the sortition, while stakeholders who own a significantly large number of tokens could dominate the process with overwhelming probability. In the case of close to evenly distributed tokens, users share similar consecutive intervals as calculated by equation 4.1. The pseudo-random hash value determines the result of the sortition. In contrast to simulation, it is not feasible to distribute tokens evenly after deploying the application online. Furthermore, proof-of-stake-based blockchain commonly requires the Initial Coin Offer to spread coins into the system. Users who purchased coins earlier for a lower price might generate a massive gap between the rich and the poor in the system.

	100	1000	10000	100000	1000000	10000000
Block Proposer	0.999	0.999	0.999	0.997	0.974	0.771
Committee	0.999	0.998	0.98	0.818	0.135	2.06e^-9
Final Committee	0.999	0.99	0.904	0.367	4.53e^-5	3.71e^-44

Table 4.1 Probability of Failing at Sortition According to Tokens

Although favoring stakeholders with higher weights could theoretically ensure security, there is an overwhelming probability that this could lead to centralization. Proof-of-stake often attracts criticism as it shifts the concentration of computing power to a monetary value. Table 4.1 shows the probability that users with a different number of tokens fail at a selection of three different roles when the entire system has 1 billion tokens. The failure rate of the block proposer does not have a significant difference until the number of tokens surpasses 1 million because the expected number of the proposer is only 26. In comparison, when the number of committee members is set at 2000 and the number of final committee members is set at 10000, there is a slight improvement until the tokens increase to 10000. But the probability of failing at becoming a final committee member is still high, approaching 90 percent. From the perspective of probability, users holding less than 10000 tokens only have a minimal impact on consensus while users owning more than 1 million tokens are almost guaranteed to succeed in the sortition with a negligible probability of failure.

[1,100)	[100,1000)	[1000,10000)	[10000,100000)	[1000000,10000000)	[10000000,100000000)	[100000000,1000000000)
8608244	3890546	1873529	630628	142880	15281	2144

Table 4.2 Number of address according to balance in US dollar[8]

To visualize how the selection performs when the tokens are unevenly distributed among the users, we run an experiment splitting 5000 users into 6 different intervals of balances, which are estimated and sampled according to the distributions observed from Bitcoin in Table 4.2. An individual user is configured to have the minimum tokens in the interval. Due to the limitations of the simulator and the negligible success rate of selection, the first column of data in the table is ignored.

Figure 4.5 shows the average selected sub-users of each group becoming committee members per round. The blue color indicates the results of committee sortition with an expected size of 2000, and the orange indicates the results of selecting the final committee members with an expected size of 10000. The first observation is that users with 100 tokens barely succeed in being selected and the selected sub-users only contribute to a minimal fraction of committee members. In contrast, groups of users with 10000 tokens display a dramatic increment of votes while the number of users in the group decreases at the same time, as shown in Figure 4.6. This indicates that the weight of the votes of these individuals is much higher, especially the top three groups. Although these groups control the majority of votes, an individual user in these groups cannot push a candidate's vote to surpass the threshold. Because consensus requires at least 68.5 percent and 74 percent of the total votes from the committee and final committee respectively to validate a candidate in the corresponding step, a single user can not change the result from the majority unless his weighted fraction is more than the specified 20%[23]. Even though a user holds more than 20% of the total tokens, he has to obtain enough votes from the sortition to cause the honest majority to fail to validate the candidate in a particular step. This only results in final agreement on the empty hash without causing significant damage to the system.

Mainstream blockchain applications, such as Bitcoin and Ethereum, maintain liveness because of the high reward coins generated from mining. Miners participate in the network as long as they can cover the cost of mining and make profits from selling their coins when market prices are high. In contrast, proof-of-stake-based Algorand consensus depends on the selection algorithm to randomly select representatives to propose and validate new blocks. The only profits for active users in the current proposal are the transaction fees from generating a new block. Without incentives for being committee members, it relies on the willingness of users to participate in the network. Although the computation cost is minimum, users running active nodes have to pay for the communication cost and massive

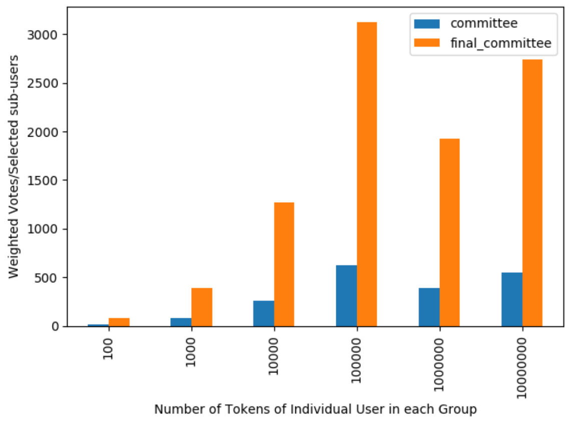


Fig. 4.5 Average Selected Sub-users per round

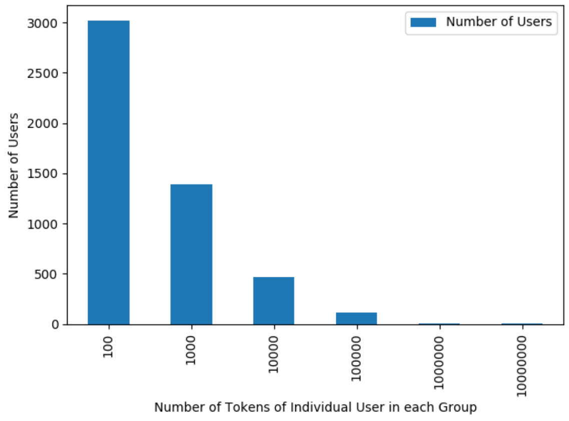


Fig. 4.6 Number of Users per Group

storage space for fast-growing chains. There is no encouragement for users to deposit a considerable amount of money into the system while paying for the cost of maintaining the node. Even if there is an incentive scheme to reward honest behaviors, the minimum number of tokens that would enable a user to succeed in sortition and participate in the voting might be too high. From the probability shown in Table 4.1, a user has to spend at least US $8000 to obtain 10,000 tokens, whose exchange rate is US $0.8 per Algorand token at the time of this research. This would barely give him success in one sortition with probabilities of 2% and 0.096%. Because the input string of the VRF differs when the step and the role changes, the sortition might allow him to succeed in one of the steps. However, there is nothing he can do to improve his sortition result unless he exchanges more tokens. Expensive investments and the low probability of winning selection might frustrate some users and cause them to remain passive participants who only broadcast transactions when necessary. Furthermore, the market price of cryptocurrencies is incredibly volatile. The high risk of losing monetary value by exchanging them to Algorand tokens might discourage users from further investments. Even worse, some current users might leave the network to prevent further loss caused by devalued tokens.

As a consequence, the low probability of success and the lack of incentives might lead to two possible scenarios, where users passively or inactively participate in the network or small stakeholders might cooperate with each other to create a pool to increase the probability of success and share the profits. In the first scenario, users holding fewer tokens occasionally participate in the network when making transactions. This will not have a negative impact on the progress of consensus until the weighted proportion of inactive users reaches the threshold of 20%. This also raises concerns as to how consensus ensures safety if the total number of tokens of the active users is less than 80% when malicious users attempt to attack the system. The result is that consensus will eventually stop in a tentative state and require recovery. Similarly, in-activeness also applies to stakeholders with a large number of tokens, which can create uncertainty in the consensus protocol. A stall in consensus caused by a sudden loss of a node holding a significant number of tokens has been reported in the released testnet[24]. The most critical factor is that the probability of selection is not dynamic in the current proposal. However, it is difficult to measure or estimate the number of users who may participate in the distributed system, because there are no constraints or penalties to force users to execute sortitions in a particular round continuously. If there is no transaction paid to or from a user, he can choose to stay inactive and allow consensus to stall without losing anything.

Without adequate regulation and penalties, a biased sortition might generate the strategy of pool mining where stakeholders holding a significant number of coins gather to dominate the block generation or small stakeholders cooperate to have a higher probability of a successful sortition.

In an ideal Algorand scenario, users would vote for the block proposal from the selected proposer whose priority is the highest in one round according to their received messages within the waiting windows. If the transactions recorded in the block proposal are valid, it should receive enough votes from committee members and be finalized as the newest block. However, Algorand propagates messages using the gossip protocol which requires users to spread messages to their randomly connected neighbors. There is no guarantee that an individual node can receive all the priority messages and blocks. Therefore, individual users start the consensus by voting for prioritized blocks, which is based on the received messages. To reduce the divergence caused by latency, the Reduction algorithm reduces candidates into only one non-empty block which receives more than the threshold number of votes. If the highest priority messages do not manage to reach the majority of users, they will also be filtered out by the Reduction. In other words, only the block proposal which receives enough votes in the Reduction step can be finalized into the blockchain. If there is a group of users with a dominating number of tokens reach agreement to work together, the most secure and effective way is to vote for their blocks at the reduction step without the risk of stopping the consensus and placing it in a tentative state. As long as the block is valid, the other small stakeholders have to passively compromise and accept the result to push the consensus into the next round. Similarly, users with a small number of tokens can create a pool to cooperate with others to have a better probability of being proposers and committee members. Joining the pool reduces not only the cost of their maintenance but also enlarges their impact on the voting process. Such agreements among stakeholders do not pose a threat to the system and may result in a shorter agreement time, but it definitely will gradually transfer the decentralized network towards centralization.

Chapter 5

Conclusion

With the rapid developments in the field of blockchain technology and the enormous prosperity it can bring, people nowadays are more aware of the benefits of a decentralized system. Simulta- neously, issues and limitations are becoming increasingly apparent. In particular, the inefficient confirmation of transactions is the most critical aspect as this does little to convince the world that blockchain can achieve at least the same performance as centralized services. The blockchain applications are seeking a breakthrough in scalability to significantly ameliorate the efficiency of processing and confirming transactions. Although there have been many approaches to solving the problems, inevitably, they cannot overcome the blockchain trilemma without weakening aspects of it.

Algorand has been proposed as an impressive consensus protocol combining the proof-of-stake and the practical Byzantine Fault Tolerance, which confirms transactions with a constant speed as the network expands. It also removes the security flaw of double-spending caused by chain forks by reducing the probability to be negligible. Based on the simulated results, the consensus algorithm demonstrates its ability to efficiently handle transactions and defend the ledger against weighted fractions of dishonesty. However, it is challenging to achieve an excellent balance between the three critical properties of the blockchain trilemma. As discussed in the analysis, the sortition scheme without the support and regulation from incentive and penalty schemes might lead to issues of liveness and centralization. Since the system relies on a minimum weighted fraction to maintain security, an incentive scheme is necessary to encourage the activeness of users and maintain the liveness at a certain weighted percentage to secure the ledger.

From a technological viewpoint, Algorand delivers an innovative solution to overcome the current struggles of blockchain applications. It demonstrates the tremendous potential that a blockchain application can securely and efficiently achieve in large-scale adoption. On the other hand, from the perspective of an individual user, the current framework might not be convincing enough to participate in Algorand with the risk of losing monetary value

References

[1] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al. (2018). Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, page 30. ACM.

[2] Antonopoulos, A. M. (2014). *Mastering Bitcoin: unlocking digital cryptocurrencies*. " O'Reilly Media, Inc.".

[3] Back, A. et al. (2002). Hashcash-a denial of service counter-measure.

[4] Bentov, I., Lee, C., Mizrahi, A., and Rosenfeld, M. (2014). Proof of activity: Extending bitcoin's proof of work via proof of stake. *IACR Cryptology ePrint Archive*, 2014:452.

[5] Bitcoin Fibre (2014). Public highly-optimized fibre network. http://bitcoinfibre.org/. [Online; accessed 20-August-2019].

[6] BitcoinVisuals (2019). bitcoin visual. https://bitcoinvisuals.com/. [Online; accessed 10-July-2019].

[7] BitInfoCharts (2019a). Bitcoin, bitcoin cash difficulty historical chart. https://bitinfocharts.com/comparison/difficulty-btc-bch.html#log&2y. Accessed: 2019-07-08.

[8] BitInfoCharts (2019b). Bitcoin rich list. https://bitinfocharts.com/top-100-richest-bitcoin-addresses.html. [Online; accessed 10-July-2019].

[9] BTC.com (2019). Mining pool distribution. https://btc.com/stats/pool?pool_mode=year. Accessed: 2019-07-08.

[10] Butterfield, A., Ngondi, G. E., and Kerr, A. (2016). *A dictionary of computer science*. Oxford University Press.

[11] Castro, M. and Liskov, B. (2002). Practical byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems (TOCS)*, 20(4):398–461.

[12] Chen, J. and Micali, S. (2016). Algorand. *arXiv preprint arXiv:1607.01341*.

[13] CommunityContributors (2019). Bitcoin core:bitcoin. https://bitcoin.org/en/bitcoin-core/. [Online; accessed 9-July-2019].

[14] Conti, M., Gangwal, A., and Todero, M. (2019). Blockchain trilemma solver algorand has dilemma over undecidable messages. *arXiv preprint arXiv:1901.10019*.

[15] Dannen, C. (2017). *Introducing Ethereum and Solidity*. Springer.

[16] Decker, C. and Wattenhofer, R. (2013). Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings*, pages 1–10. IEEE.

[17] Demers, A., Greene, D., Houser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H., Swinehart, D., and Terry, D. (1988). Epidemic algorithms for replicated database maintenance. *ACM SIGOPS Operating Systems Review*, 22(1):8–32.

[18] Denis, Frank (2013). The sodium cryptography library. https://download.libsodium.org/doc/. [Online; accessed 15-July-2019].

[19] DSN Research Group (2019). Dsn bitcoin monitoring. https://dsn.tm.kit.edu/bitcoin/. [Online; accessed 10-July-2019].

[20] Duong, T., Fan, L., and Zhou, H.-S. (2016). 2-hop blockchain: Combining proof-of-work and proof-of-stake securely.

[21] Eric Lombrozo, Johnson Lau, P. W. (2015). Segregated witness (consensus layer). Website.

[22] G.Hummer (2017). Sharding-faq. https://github.com/ethereum/wiki/wiki/Sharding-FAQ. [Online; accessed 10-July-2019].

[23] Gilad, Y., Hemo, R., Micali, S., Vlachos, G., and Zeldovich, N. (2017). Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 51–68. ACM.

[24] Ivica Milosevic (2019). Testnet stalled. https://forum.algorand.org/t/testnet-stalled/88. [Online; accessed 20-August-2019].

[25] Jakobsson, M. and Juels, A. (1999). Proofs of work and bread pudding protocols. In *Secure Information Networks*, pages 258–272. Springer.

[26] Josefsson, S. and Percival, C. (2016). The scrypt password-based key derivation function.

[27] King, S. (2013). Primecoin: Cryptocurrency with prime number proof-of-work. *July 7th*, 1:6.

[28] King, S. and Nadal, S. (2012). Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19.

[29] Korpela, K., Hallikas, J., and Dahlberg, T. (2017). Digital supply chain transformation toward blockchain integration. In *proceedings of the 50th Hawaii international conference on system sciences*.

[30] Lamport, L., Shostak, R., and Pease, M. (1982). The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401.

[31] Larson, S. (2017). Bitcoin split in two, here's what that means. Website. [online]https://money.cnn.com/2017/08/01/technology/business/bitcoin-cash-new-currency/index.html.

[32] Lin, I.-C. and Liao, T.-C. (2017). A survey of blockchain security issues and challenges. *IJ Network Security*, 19(5):653–659.

[33] Mazieres, D. (2015). The stellar consensus protocol: A federated model for internet-level consensus. *Stellar Development Foundation*, page 32.

[34] Micali, S., Rabin, M., and Vadhan, S. (1999). Verifiable random functions. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 120–130. IEEE.

[35] Miller, A., Litton, J., Pachulski, A., Gupta, N., Levin, D., Spring, N., and Bhattacharjee, B. (2015). Discovering bitcoin's public topology and influential nodes. *et al.*

[36] Miller, A., Xia, Y., Croman, K., Shi, E., and Song, D. (2016). The honey badger of bft protocols. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 31–42. ACM.

[37] Nakamoto, S. et al. (2008). Bitcoin: A peer-to-peer electronic cash system.

[38] Narayanan, A., Bonneau, J., Felten, E., Miller, A., and Goldfeder, S. (2016). *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press.

[39] Nguyen, G.-T. and Kim, K. (2018). A survey about consensus algorithms used in blockchain. *Journal of Information processing systems*, 14(1).

[40] Pagh, R. and Bldg, N. (2001). D.-a. c, and ff rodler. *Cuckoo Hashing*.

[41] Pease, M., Shostak, R., and Lamport, L. (1980). Reaching agreement in the presence of faults. *Journal of the ACM (JACM)*, 27(2):228–234.

[42] Peercoin project (2018). peercoin university. https://university.peercoin.net/#/time-as-an-alternative-scarce-resource. [Online; accessed 10-July-2019].

[43] Popper, N. (2017). Some bitcoin backers are defecting to create a rival currency. *The New York Times*.

[44] QuantumMechanic (2011). Proof of stake instead of proof of work.

[45] Tromp, J. (2014). Cuckoo cycle: a memory-hard proof-of-work system. *IACR Cryptology ePrint Archive*, 2014:59.

[46] VISA (2017). [online]https://usa.visa.com/run-your-business/smallbusiness-tools/retail.html.

[47] Wang, Y. (2019). Another look at ALGORAND. *CoRR*, abs/1905.04463.

[48] Wiki, N. (2014). Whitepaper:nxt — nxt wiki,. [Online; accessed 08-July-2019].

[49] Wikipedia contributors (2019). Public-key cryptography — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Public-key_cryptography&oldid=911283666. [Online; accessed 4-September-2019].

[50] w.Lim (2016). How do i compare the “scalability” capabilities between ethereum and bitcoin? https://ethereum.stackexchange.com/questions/3308/how-do-i-compare-the-scalability-capabilities-between-ethereum-and-bitcoin. [Online; accessed 10-July-2019].

[51] Zheng, Z., Xie, S., Dai, H.-N., Chen, X., and Wang, H. (2018). Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services*, 14(4):352–375.

[52] Zohar, A. (2015). Bitcoin: under the hood. *Communications of the ACM*, 58(9):104–113.