BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

SECURITY AND PERFORMANCE TESTBED FOR SIMULATION OF PROOF-OF-STAKE PROTOCOLS

TESTOVÁNÍ BEZPEČNOSTI A VÝKONU PROOF-OF-STAKE PROTOKOLŮ

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR **BC. JAKUB HUD**
AUTOR PRÁCE

SUPERVISOR **Ing. IVAN HOMOLIAK, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2022

Brno University of Technology
Faculty of Information Technology

Department of Intelligent Systems (DITS) Academic year 2021/2022

Master's Thesis Specification

25093

Student:	**Hud Jakub, Bc.**
Programme:	Information Technology
Field of study:	Information Technology Security
Title:	**Security and Performance Testbed for Simulation of Proof-of-Stake Protocols**
Category:	Security

Assignment:

1. Get familiar with existing proof-of-stake protocols and their popular hybrid variants. Study existing simulation testbeds for blockchain-oriented consensus protocols.
2. Make a theoretical comparison of these protocols in terms of throughput, scalability, security, failure-tolerance, liveness, safety, finality, etc. In security analysis, consider all existing proof-of-stake vulnerabilities as well as general ones.
3. Select at least three protocols (including ProPos and Hedera HashGraph) and implement them as part of simulation testbed in an arbitrary programming language.
4. Make a comparative study of the selected protocols using the results obtained from simulations. Experiment with the simulation of various threats.
5. Based on the simulation results and theoretical analysis, propose a few improvements, which you can validate using the testbed.

Recommended literature:

- Homoliak, Ivan, et al. "The security reference architecture for blockchains: Towards a standardized model for studying vulnerabilities, threats, and defenses." *arXiv preprint arXiv:1910.09775* (2019).
- Reijsbergen, Daniel, et al. "ProPoS: A Probabilistic Proof-of-Stake Protocol." arXiv preprint arXiv:2006.01427 (2020).
- A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In CRYPTO'17, 2017.
- Aoki, Y., Otsuki, K., Kaneko, T., Banno, R. and Shudo, K., 2019. SimBlock: a blockchain network simulator. arXiv preprint arXiv:1901.09777.
- Zhang, Ren, and Bart Preneel. "Lay down the common metrics: Evaluating proof-of-work consensus protocols' security." *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Homoliak Ivan, Ing., Ph.D.**
Head of Department:	Hanáček Petr, doc. Dr. Ing.
Beginning of work:	November 1, 2021
Submission deadline:	May 18, 2022
Approval date:	November 3, 2021

Abstract

This thesis deals with testing the security and performance of PoS-based protocols. Comparison of ProPos, Algorand, Hedera, Ouroboros and Tezos is presented on theoretical level in terms of performance, attack vulnerabilities and attacks mitigation. This thesis includes a simulation framework for testing Algorand, ProPos and Hedera protocols. The simulation framework is created using Omnet++ 5.4.1. Focus of the simulation experiments is on the performance of the selected protocols. Based on the results of the experiments a few improvements are discussed.

Abstrakt

Tato práce se zabývá testováním výkonu a bezpečnosti konsensus protokolů typu Proof-of-Stake (PoS). Na teoretické úrovni porovnává protokoly ProPos, Algorand, Hedera, Ouroboros a Tezos z pohledu výkonu, možností útoků a obraně proti těmto útokům. Součástí práce je implementace simulačního frameworku v Omnet++ 5.4.1 pro provádění simulačních experimentů s protokoly ProPos, Algorand a Hedera. Simulace se zabývají zejména výkonem těchto protokolů. Na základě zjištěných výsledků jsou navržena určitá vylepšení.

Keywords

proof-of-stake, PoS, blockchain, Omnet++, simulation, security, performance, testbed, Algorand, Hedera, ProPos, LaKSA

Klíčová slova

proof-of-stake, PoS, blockchain, Omnet++, simulace, bezpečnost, výkon, testbed, Algorand, Hedera, ProPos, LaKSA

Reference

HUD, Bc. Jakub. *Security and Performance Testbed for Simulation of Proof-of-Stake Protocols.* Brno, 2022. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Ivan Homoliak, Ph.D.

Rozšířený abstrakt

Tato práce se zabývá především Proof-of-Stake (PoS) konsenzus protokoly používanými v decentralizovaných systémech v kombinaci s technologií blockchain. Výsledkem je teoretické porovnání PoS protokolů a dále vytvoření simulačního testbedu, ve kterém jsou protokoly simulovány. Na základě výsledků jsou diskutována možná vylepšení protokolů.

Práce nastiňuje i další protokoly používané k dosažení konsenzu a jejich rozdíly. Blockchainové systémy používající konsensuz protokoly založené na PoS jsou stále na vzestupu, protože přináší určité výhody oproti protokolům typu Proof-of-Work (PoW). Tím, že jde o velmi mladé odvětí, nejsou ještě všechny vlastnosti PoS protokolů ani dalších konsenzus protokolů dostatečně prozkoumány. Cílem této práce je najít vylepšení již existujících protokolů využívajících princip PoS. Testbed implementovaný v rámci této práce testuje a experimentuje s protokoly Algorand, ProPos a Hedera a nabízí možnost jak pomocí simulací snadno zkoušet vlastnosti těchto protokolů při různých parametrech sítě nebo nastavení samotných protokolů.

Konsenzus protokoly fungují v úzkém spojení s blockchain technologií, proto je uveden přehled této technologie, její typy, fungování a využití. Jedna kapitola je věnována Bitcoinu, protože výrazně ovlivnil směr vývoje v tomto odvětví. Dále se práce zabývá možnostmi dosahování konsenzu v decentralizovaných systémech a konkrétními konsenzus protokoly (proof-of-stake, proof-of-resource, proof-of-importance a dalšími). Popisuje jejich fungování, proč byl daný protokol vytvořen (jaký problém se snaží řešit) i důležité metriky pro jejich porovnávání. Důraz je kladen na bezpečnost popsaných typů protokolů. Jsou uvedeny známé útoky na různé protokoly a možnosti prevence nebo minimalizace těchto útoků.

V kapitole 4 jsou popsány vybrané PoS protokoly ProPos, Algorand, Hedera, Ouroboros a Tezos. Je popsáno jak každý protokol dosahuje konsenzu a vlastnosti protokolu, např. jestli je zaměřen na co největší propustnost nebo větší decentralizaci apod. U každého protokolu se práce zabývá možnostmi útoků. Jsou popsány mechanismy, kterými protokol útok znemožňuje nebo alespoň výrazně snižuje pravděpodobnost jeho úspěšného provedení. Dále je popsán teoretický výkon každého protokolu vyjádřený jako propustnost nebo počet transakcí za vteřinu. Veškerá data jsou získána z publikací autorů daného protokolu.

V kapitole 5 je krátký přehled již existujících testbedů pro testování blockchainových systémů a různých konsenzus protokolů.

Dále už následuje popis vytvořeného testbedu. Práce popisuje návrh testbedu a jeho vnitřní design. Testbed je založen na simulačním nástroji Omnet++ 5.4.1 a simulace probíhaly na běžném pracovním počítači se systémem Ubuntu 18.04. Je popsán i způsob simulace sítě, tzn. propojení uzlů a jejich komunikace včetně transportní vrstvy.

V kapitole 7 jsou výsledky experimentů prováděných s protokoly Algorand, ProPos a Hedera. Pro každý protokol bylo provedeno několik experimentů. Každý experiment se skládal z mnoha simulačních běhů, ve kterých se měnil daný parametr. Především byl testován výkon protokolů a jak se chovají při různém počtu nefunkčních nodů, dále pak chování protokolů ProPos a Algorand v síti s vysokou latencí a výkon protokolu Hedera. V případě protokolu Algorand byl simulován i útok na protokol. Výsledky jsou prezentovány v podobě grafů a tabulek. Na základě výsledků je uvedeno porovnání protokolů a návrh možných vylepšení.

Security and Performance Testbed for Simulation of Proof-of-Stake Protocols

Declaration

I hereby declare that this Master's thesis was prepared as an original work by the author under the supervision of Mr. Ing. Ivan Homoliak, Ph.D. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

. . . .
Bc. Jakub Hud
May 18, 2022

Acknowledgements

I would like to thank to my supervisor Ing. Ivan Homoliak, Ph.D. for professional help and guidance. Also I want to thank to my parents for continuous support and help throughout my studies.

Contents

Chapter 1

Introduction

Blockchain technology gain its popularity with decentralized peer-to-peer digital currency Bitcoin. The properties of decentralized blockchain technology like immutability make it ideal for creating and exchanging digital assets. Since Bitcoin was created, many more decentralized applications have been emerging constantly showing that cryptocurrencies are not the only use case.

Consensus protocols are setting rules for building blockchain structure and ensures integrity and security of the network. In Bitcoin consensus is achieved using Proof-of-Work mechanism which is very energy consuming. To solve this problem, Proof-of-Stake protocols were introduced in 2011. These days more and more applications and cryptocurrencies are using PoS as it is seen as favorable alternative to PoW.

Because the use of a blockchain technology in modern applications is still in its early stages and as the number of projects using PoS mechanism is increasing rapidly, it is hard to evaluate all the important properties like security, throughput, finality and more. On top of that projects with different use cases are using very different approaches of reaching a consensus. In today's decentralized systems there is always some kind of trade-off between security, throughput and scalability.

1.1 Goals

This thesis has several goals. First being a theoretical comparison of some of the well known implementations of PoS projects in terms of throughput, scalability, security, failure-tolerance, liveness, safety, finality and more. Then to design testbeds for simulation of selected PoS implementations and evaluate the results with focus on security. And finally to propose some improvements based on the simulation results and to validate the improvements using simulation.

1.2 Organization

Chapter 2 briefly describes blockchain technology as it is used today. There is a short description of the design of Bitcoin, the first blockchain project with a mass usage. Then types of blockchain are discussed in more detail with their use cases.

Chapter 3 is all about consensus protocols, how they work, their properties and types. Well known attacks and their possible mitigations are also discussed in this chapter.

In Chapter 4 theoretical analysis of some existing PoS systems is presented. Consensus algorithm of each PoS project is described so that its properties can be seen better. Some properties (e.g. throughput, finality) were tested experimentally by previous research and the results are included this chapter. Theoretical comparison of PoS projects that are described here is then presented in a form of a table.

Chapter 5 deals with existing simulation testbeds Vibes, Bitcoin Simulator and Simblock, their use cases, properties and limitations.

In Chapter 6 the created testbed is presented. There is a description of how the testbed works with an explanation of the important design choices.

Then in Chapter 7 all simulation experiments performed are explained. The protocols tested are Algorand, ProPos and Heder. The results of the experiments are also presented and discussed in this chapter and the protocols are compared.

Chapter 2

Blockchain technology

2.1 Blockchain overview

Blockchain technology is very similar to a distributed ledger. It is a distributed database spread across nodes that are connected through P2P network. In blockchain however, the data are organized into blocks. When new block is created, it is cryptographically linked to the current most recent block. This ensures that when a block becomes a part of a blockchain it cannot be changed in the future because it would break the links between all following blocks. There is not a signle universally accepted definition of blockchain but we can list the key properties as follows [7]:

- data redundancy – each node has its own copy of the blockchain;
- transaction requirements check and validation;
- recording and storing the transactions in sequentially ordered blocks, creation of which is ruled by a consensus algorithm;
- transactions based on public-key cryptography and a transaction scripting language.

In case of cryptocurrencies, blocks contain transactions where each transaction may represent a transfer of crypto-tokens, application code (smart contracts) or other data. All nodes need to agree who will be the creator of a new block. Typically, this process is not strict, but randomness is involved. This means more than one new block could be created at the same time leading to multiple different versions of the blockchain (i.e., fork). This is undesirable situation because the same transaction may be present in multiple blocks. To resolve the situation longest chain rule could be used which means that every user always take the longest chain as valid. Transactions from the block that was on shorter branch are then returned to the mempool. In Figure 2.1 illustration of a simple fork is shown. In reality multiple branches with the same length may exist. The sooner the blockchain solution can resolve the fork, the faster the finality can be reached. Consensus algorithm describes a way to resolve such a situation as well as rules for creating a new valid block with ordered transactions.

While blockchain technology brings many advantages over traditional systems, there is number of drawbacks as well. Blockchain based systems often operate in trustless environment meaning nodes cannot trust each other. This is a problem when new node is connected to the network or when a node is offline for some time. In attempt to synchronize with the network, a node has to choose a valid version of the blockchain among all the blockchains

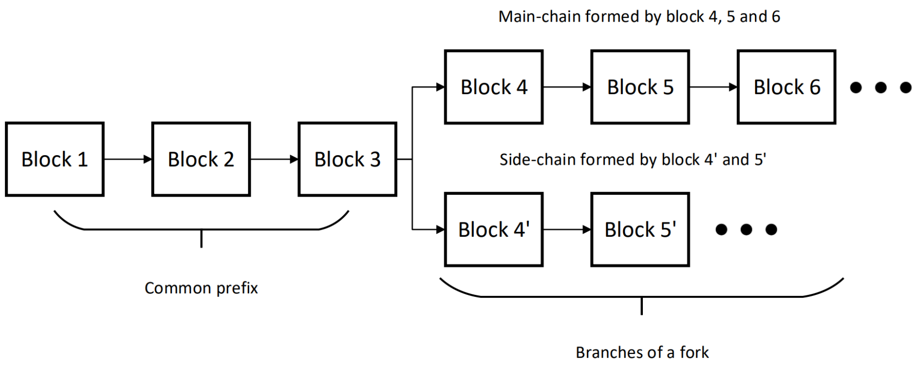


Figure 2.1: Example of a blockchain fork. The side chain cannot keep up whith the main chain and will be discarded [19].

presented to her by other nodes. Consensus algorithms are designed to solve this issue. Different approaches of the algorithms are discussed in more detail in Chapter 3.

Smart contracts

Smart contract is a term that in regard to the blockchain technology refers to an application code stored in the blockchain and executed by the network nodes (miners, stakers, etc.). The key properties of the smart contracts are the immutability of the computer code and their decentralized execution. Any user can view the code of the smart contract and can verify how the smart contract operates.

Smart contracts can manage cryptocurrency tokens and perform automatic tasks as well as communicate with each other. By linking more smart contracts complex decentralized applications are created. Smart contracts are often used to build trust in the system because of their public nature. The most notable project that supports smart contract today is Ethereum [57].

2.2 Blockchain use cases

Blockchain technology is not only used in the financial sector nowdays but many more areas including data storing and replication, data verification, voting in elections, lottery, marriage registrations, product tracing, identity management, temper-proof event logging and more [7]. In this section a few notable use cases are described.

Financial

Cryptocurrencies are taking advantage of the blockchain features such as immutability and enabling transfer of digital assets (crypto-tokens), between two parties without a middleman or a central authority. By design it should be almost impossible to censor transactions or seize crypto-tokens by force. The transfer of value is especially beneficial in interbank or global transactions where the conventional solutions may be more expensive and take more time.

Smart contracts enable much more possibilities for cryptocurrencies aside from value transfer. For example a lending and borrowing of the tokens. The lender provides an amount of the tokens for a period of time and gets an interest on his investment. The borrower gets the tokens but needs to lock a collateral as a guarantee. The collateral may be in the form of tokens of different kind than the ones that were borrowed and typically has greater value than borrowed funds. The idea is that the loan has to be secured by the collateral at all times even when the value of the collateral fluctuate over the time.

Another area taking advantage of smart contracts is decentralized exchanges. This type of exchange is using smart contracts to settle an exchange of the cryptocurrency tokens between two parties. The advantage over centralized exchanges is the elimination of middle man (in this case the exchange) that the users need to trust with their funds.

Data management and verification

A blockchain can also be used for storing data that are duplicated among many nodes. This approach reduces the risk of outages and hardware failures. Because of the duplicity of the data among the nodes, the data are always available even if some nodes go offline. Another advantage is data integrity and full history of edits. These properties are achieved by the nature of the blockchain itself. Example of a project that use blockchain in this way is Storj (Metadisk): Blockchain for Distributed Cloud Storage [48]. Other use cases related to data storage on blockchain include identity data management, tamper-proof event logging, content or product timestamping, healthcare record storing and more [8].

2.3 Bitcoin

First publicly known use of blockchain technology is in design of Bitcoin cryptocurrency [4]. The idea is to enable secure decentralized exchange of digital currency – Bitcoin. The system is fully decentralized and public meaning anyone can join the network and play a role in the consensus algorithm execution or read/write from/to blockchain structure.

Bitcoin consensus mechanism (Nakamoto Consensus) is based on a computational power that nodes spend to be able to create new blocks. The security of Bitcoin is based on belief that an attacker cannot create fake blocks without other nodes noticing because its validity is easily verifiable. It is a clever way of choosing a new block creator with fairness when the nodes do not trust each other.

When a node wants to publish a new block, she needs to add a piece of data (referred to as nonce) to the block such that hash of the whole block has certain number of leading zeros. The process of searching for new valid block is known as mining. See Figure 2.2 for visualization of blockchain structure. The number of leading zeros changes based on the network total computational power making it harder or easier to find new block. The goal is to maintain roughly the same time between the blocks.

Note that because of the use of a hash function, creating a valid block takes much more time than to verify its hash. Creator of a block is rewarded with Bitcoins to compensate for resources spent. There is an incentive for all nodes to stay synchronized with the current state of the blockchain at all times because mining on non-valid blockchain is a waste of resources. When multiple competing blockchains are created each node chooses the longest chain in terms of number of blocks (longest-chain rule). Consensus mechanisms based on spending computational power are called Proof-of-Work and are more discussed in Section 3.3.

Bitcoin and other cryptocurrencies utilizes asymmetric cryptography to verify the identity of the users of the network. For example when user wants to send some Bitcoins to

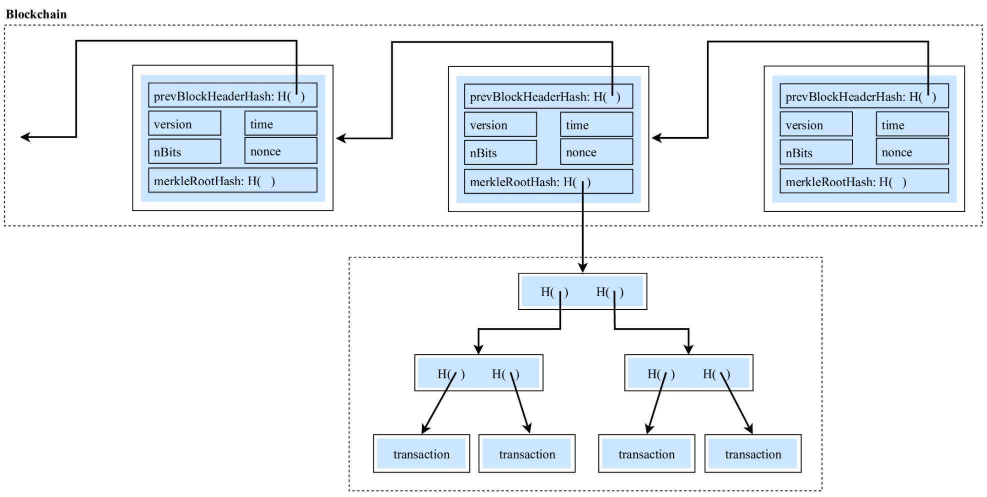


Figure 2.2: Simplified blockchain structure in Bitcoin [16, p. 4]

another user she creates a transaction specifying sender, recipient and amount and then digitally signs the transaction and broadcasts it to the network. All mining nodes verify the validity of received transactions in terms of user's identity and that the user has enough Bitcoins to satisfy the transaction and only then adds it to the block.

2.4 Types of blockchains

In this thesis blockchains are distinguished based on how new nodes enter the consensus mechanism as described in [27]:

Permissionless – Any node can join consensus protocol without permission. To make Sybil attacks as hard as possible Proof-of-Resource consensus protocols are used where the consensus power of a node is proportional to its resources allocated. Better scalability is possible because in PoR protocols there is less communication required to reach consensus than in the PBFT based protocols. Permissionless blockchains are currently used in most cryptocurrencies (e.g., Bitcoin [4], Ethereum [57]).

Permissioned – In permissioned blockchains there is a rather small group of validating nodes that are handling transaction validation and are building new blocks. These nodes are run or approved by centralized authority (e.g., an institution or a company). This way only approved nodes are contributing to the validation process enhancing system security. Still, some of the validating nodes may be malicious. Hence Byzantine fault tolerance is necessary. Practical Byzantine Fault Tolerance protocol (PBFT) [14] is used in many current blockchain projects.

Permissioned blockchains using PBFT and its variants can commit blocks more quicky than permissionless blockchains. Transactions are finalized as soon as the block containing them is added to the blockchain. The problem with these protocols is scalability. This is because they usually require broadcast messages among all nodes to reach consensus. PBFT is therefore only suitable for blockchains consisting of a few nodes [52].

Semi-permissionless – Anyone can join the protocol, but they have to get some form of permission from a consensus node first. Permission can be granted after locking some amount of crypto tokens (stake). This allows for better decentralization than in the case of permissioned blockchains. When a node decides to leave the consensus protocol, the tokens are unlocked again (usually after some period of time to prevent Sybil attacks). The consensus power is proportional to the size of the stake. However, in many protocols in addition to the stake size other metrics are used (e.g. age of staked tokens) to regulate consensus power of a node. Most common use of semi-permissonless blockchains is by projects with Proof-of-Stake consensus protocol or its hybrid variants, e.g. PeerCoin [38], TRON [53].

Chapter 3

Consensus protocols

In very simple terms consensus protocol is used to reach consensus on which transactions to put in the blockchain and in what order. This mechanism is important so that all nodes always have the same version of a blockchain. Also, consensus protocol must take into account that there may be malicious or faulty nodes. These could disrupt consensus protocol execution.

Properties expected from consensus protocol as enumerated by Cachin et al. [10] are liveness and safety. Other characteristics important when comparing consensus protocols are finality, throughput and scalability.

- **Liveness** – Liveness ensures that if a transaction is broadcasted and received by at least one honest node then it will eventually be delivered to all honest nodes.
- **Safety** – Safety ensures that if an honest node accepts (or rejects) a transaction then all other honest nodes make the same decision.
- **Finality** – Finality up to the block B means all blocks from genesis block up to the block B cannot be changed or replaced by another chain of blocks. Finality is reached when there are several successive blocks after block B making any changes to blocks up to the block B infeasible [27].
- **Throughput** – In relation to blockchain technology throughput says how much data can be written to the ledger per time unit. It is usually given in transactions per second (TPS).
- **Scalability** – Scalability describes how the properties of the protocol change when it is used by more users. For example when the protocol scales well, its throughout, security, finality etc. stays about the same even with much more users using it.

3.1 Failure models

When a node behavior is different from the specification, the node is considered faulty. In this thesis faulty behavior is classified as in [43]:

Fail stop – A node stops operating completely or continue operating in a way that other nodes detect its faulty behavior. This kind of failures naturally occur unintentionally, e.g., due to a software bug, connection problem, faulty hardware etc.

Byzantine failure – Node can generate arbitrary data and act in a malicious way cooperating with other Byzantine nodes targeting consensus protocol. Therefore it is important to consider Byzantine failure model in security-critical applications.

3.2 Establishing a consensus

There are two main techniques used to establish a consensus [28]. Combinations of these two are used as well:

Lottery-based - In lottery-based algorithms, leader is chosen by a lottery. Only leader can then produce a block and send it to the rest of the network. All other nodes can easily verify that the block produced is valid and produced by chosen leader. Usually, to elect a leader, nodes do not need to communicate with each other (e.g., Bitcoin). The advantage of this technique is scalability, since there is low communication overhead associated with reaching a consensus. On the other hand, there can be multiple leaders elected. In that case, more versions of blockchain called forks exist among nodes. To resolve the issue, all nodes use fork-choice rules. For Bitcoin, longest chain rule is used. Another approach is the strongest chain rule which calculates the quality of each chain. The algorithm for computation of the blockchain quality varies but usually the blockchain that is harder to create wins (e.g., hash of the last block has more leading zeros than the hash of a last block in the competing blockchain). The fork resolution leads to longer time to finality which then increases the risk of double-spending attacks.

Voting-based - In voting-based algorithms, all nodes vote for blocks. The advantage here is that when majority of nodes validates a transaction or block finality occurs. The downside is that each node needs to transfer messages to all other nodes before consensus is reached. Thus, the more nodes exist in the network, the more time it takes to reach consensus. Voting-based algorithms provide lower time to finality, but scalability is an issue. Example of voting-based algorithms are Byzantine Fault Tolerant (BFT) protocols (e.g. PBFT [14], FastBFT [22], Proteus [40]).

There are combinations of lottery-based and voting-based algorithms that aim to take the best from both approaches. One way to improve scalability is to choose only small group of consensus nodes by lottery and these nodes then run voting-based consensus algorithm (e.g., Algorand [1]).

3.3 Proof-of-Resource protocols

Proof-of-resource (PoR) protocols are based on sacrificing a scarce resource like computational power (Proof-of-Work), memory storage space (Proof-of-space) or crypto-tokens (Proof-of-burn). Consensus power of a node is proportional to the amount of resource sacrificed. The aim of this approach is to prevent Sybil attacks. In PoR algorithms leader election mechanism is lottery-based. To ensure fairness the probability of a node to be chosen as a block producer neeeds to be proportional to the resource sacrificed. In case of PoW algorithms, the nodes who are trying to produce new blocks are referred to as miners.

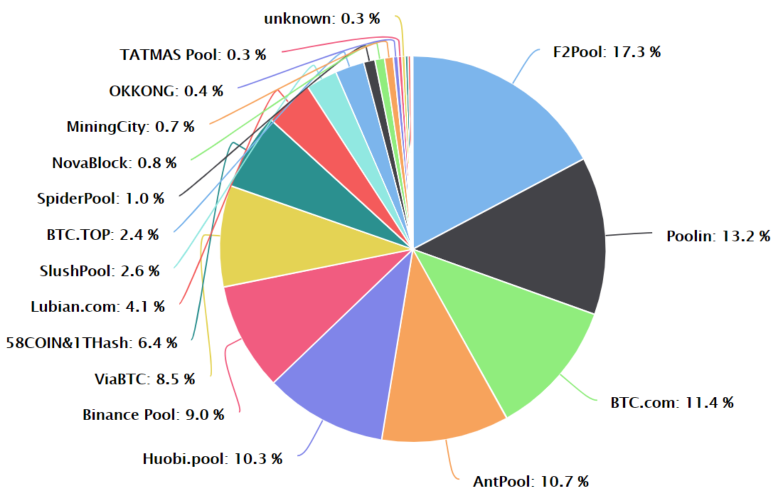


Figure 3.1: Mining power distribution in Bitcoin network 19.11.2020

Consensus algorithm used in Bitcoin is called Nakamoto Consensus (NC) [4]. It is a PoW based algorithm and in case of a fork, longest chain rule is used. Nodes do not need to communicate with each other while solving PoW puzzle. This makes NC scalable to much higher number of nodes than in case of pure BFT protocols. On the other hand, forks happen much more often which leads to longer time to finality. Another limitation of NC is in use of PoW which is very energy consuming and thus inefficient. One of the main properties of Bitcoin is decentralization. However, the miners with low mining power would have to mine for very long time before they find a block, so they form large groups called mining pools. This behavior leads to situation where a few mining pools account for significant portion of all mining power (see Figure 3.1[1]).

Attacks in PoR algorithms

51% attacks - Imagine we have a PoW based system and there is an adversary who is in control of more than 50% of all the computational power. Then she will be able to secretly mine a few blocks and then publish them. Since adversary is producing blocks faster, her published chain will be always longer. In this case adversary is in full control of which transactions will be included in a block and which will not. Furthermore, adversary could then perform a double-spend attack [27, p. 10]. When she pays for goods with cryptocurrency and then when it seems that finality was reached she publishes her longer chain excluding her transaction for said goods. All other nodes adopt this new chain because it is longer.

Selfish mining - It is assumed that when (honest) miner finds a block, she publishes it immediately. When a dishonest miner finds a block, she can withhold the block and continue mining on a private blockchain while all other nodes mine on the public blockchain. If the dishonest miner gets lucky and finds another block, she is now two blocks ahead of public blockchain and continue mining. When the public blockchain starts to catch up with the private blockchain and is only one block behind, then the dishonest node publishes her private blockchain. All nodes accept this blockchain because it is longer. In this situation

[1]https://btc.com/stats/pool

honest nodes wasted mining power on public blockchain which was thrown away, while dishonest node gets rewards for a few mined blocks. By repeating this process, the dishonest node gets more rewards than an honest node with the same mining power. Process of selfish mining is in Figure 3.2. This attack becomes especially feasible when miners group into mining pool. As it was shown by Ittay Eyal et al. [31] Bitcoin protocol will never be safe against this attack in case of adversary with only 1/3 or more of consensus power. At time of writing there is no single Bitcoin mining pool with that much mining power. However top three mining pools combined have around 41,9%, and top four combined have 52,6% as shown by Figure 3.1.

Feather-forking – This attack can be used to censor target transaction or any transaction from a specified user. An adversary publicly announces that she will fork blockchain if the transaction is included in the last block. Depending of the computational power of the adversary there is a chance that the adversary succeeds and mine a few blocks faster than the rest of the network. In this case the fork is adopted and the transaction is censored because the adversary doesn't include target transaction. If the adversary doesn't succeed she will give up when the main chain is *k* blocks ahead, e.g., two.

Of course the probability of a success of this attack is very low. But a very important element is that when the adversary publicly announces her intentions, she effectively creates an incentive for the other miners not to include target transaction because in the case when the adversary succeeds and her fork is adopted, other miners would waste their mining power on the other branch.

This attack is not profitable but enables an adversary to choose which transactions will be confirmed and which will not. [15]

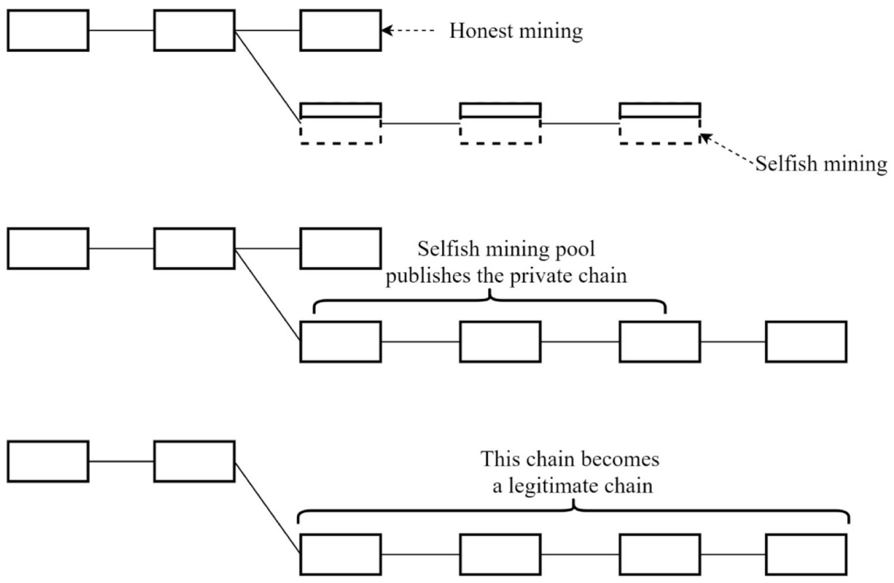

Honest mining
Selfish mining
Selfish mining pool publishes the private chain
This chain becomes a legitimate chain

Figure 3.2: Illustration of selfish mining attack in PoR protocol [42].

3.4 Byzantine-Fault-Tolerant protocols

In BFT protocols consensus is reached when the majority of nodes (typically $\frac{2}{3}$) is honest. Example of practical implementation is Practical Byzantine Fault Tolerance Protocol (PBFT) [14]. Here as long as the condition for maximum dishonest nodes is satisfied, safety and liveness is ensured.

In general, BFT protocols provide fast time to finality because when all honest nodes reach consensus on an action, it is very difficult to revert it later (at least $\frac{2}{3}$ of nodes would need to agree on this). The idea behind BFT protocols is that all nodes exchange messages and only when $\frac{2}{3}$ or more are in consensus then the protocol progresses. BFT without any optimalizations uses broadcast to deliver messages to all nodes. Messaging complexity is $O(n^2)$ where n is the number of nodes in the network. This yields problem with scalability. There are projects that try to address this issue (e.g., PBFT [14], FastBFT [22], Proteus [40]). Popular technique used to improve scalability is to limit broadcast and aggregate messages as they travel through the network.

Attacks related to BFT protocols are variations of 51% attack where attacker compromises more than $\frac{1}{3}$ of all nodes. However, attacker does not need to take control of so many nodes but can use DOS attack to make significant proportion of honest nodes unavailable and then try to disrupt the protocol with much less effort. Furthermore, as was shown by Amir et al. [3], in real world scenario the attacker only needs small number of nodes to make protocol so slow that it becomes practically unusable.

3.5 Proof-of-Stake protocols

The idea behind Proof-of-Stake (PoS) protocols is to use security deposit (i.e., stake) to participate in consensus mechanism. PoS protocols are semi-permissionless. As long as a node wants to participate in the consensus, the stake is locked. Node can choose to unlock the stake at any time. This is great improvement when compared to PoR, where scarce resource is wasted. There is strong incentive for staking nodes to remain honest because they would not want to attack blockchain where their stake is locked. PoS is lottery-based, and the consensus power is proportional to the size of the stake.

Vulnerabilities of PoS algorithms

Mining in PoS is not strictly tied to physical resources as in the case of PoR, which opens door to different attack vectors. For example, it is possible to mine at more chains at the same time with almost no extra effort (this is exploited in nothing at stake attack). Small party of nodes can even overthrow main chain by completely new chain built from genesis block (long-range attacks). At the time of writing, the security of PoS protocols is not formally proven.

Nothing at stake – Because the process of electing a leader that mine a block is random, there are always new forks emerging. The miners lose nothing by mining on more competing forks at the same time because they do not really spend anything in mining process as opposed to PoW. The problem with this behavior is that when there are two forks and all miners mine on both, adversary could make transaction for goods on one chain but not on the other. Adversary could then stop mining on the chain that contains her transaction.

After few blocks, this chain becomes shorter and is discarded. This way adversary double-spent her crypto-tokens.

Nothing at stake vulnerability poses a threat only when significant proportion of miners mine on more than one chain simultaneously. In current cryptocurrencies miners generally don't do that because they don't want to undermine credibility of the cryptocurrency and its value. Still this vulnerability could be exploited in the future. One solution to this problem has been proposed in Slasher algorithm [45]. The idea behind Slasher is that nodes are punished for mining on more than one chain and the node that spotted this undesirable behavior first is rewarded.

Long-Range attack – Nature of PoS allows adversary to create fork from block in distant past or even the genesis block. Adversary can generate chain longer than the main chain offline and then broadcast it to the network. Alternatively, she could buy or compromise private keys of a party that had large stake in the past, make fork from that moment in time and steal the stake. Many variants of this attack are described in [49]. Countermeasures include:

- Checkpointing – This technique is widely used in PoS blockchains. Only n number of most recent blocks can be reorganized. This mitigates long-range attacks, but short-range attacks are still possible. The number of blocks between any two checkpoints varies among PoS projects, but the goal is to make trade-off between checkpointing too often and not checkpointing often enough. For example, in case of Feathercoin [23] checkpoint is created every five blocks (a few minutes). In Casper the Friendly Finality Gadget [13] checkpoint is every 100th block yielding the time between checkpoints also a few minutes.

- Chain density statistics – "The expected number of participating players at any step of the protocol is known; thus alternative protocol execution histories that exhibit significantly smaller participation can be immediately dismissed as adversarial." [47]

Stake-Bleeding - In this attack adversary launches long-range attack but at the same time includes all the transactions from the main chain in the fork. The only difference in chains is that the adversary makes sure that she is the creator of all blocks in the new chain taking all the mining fees. If the chain exists for long enough (e.g., a few years), the accumulated fees can give adversary a consensus power to take control over the chain. Mitigations of this attack are the same as of long-range attacks (e.g., checkpointing) but also different techniques such as *context-sensitive transactions* as proposed by P. Gaži et al. [47].

"Context-sensitive transaction is a transaction that includes the hash of the blockchain at some recent prior point. It is easy to see that such transactions cannot be transferred to an alternative blockchain that is privately maintained by a malicious set of stakeholders." [47]

Grinding attack – In *stake grinding attacks*, before adversary creates a block, she tests different contents and headers to influence the odds of being selected as the leader again in the future. This attack is possible when election process uses past blocks as a source of randomness. In Ouroboros [35], this attack is mitigated using coin tossing protocol that collects randomness from all participating nodes and thus cannot be guessed in advance.

Denial-of-service attack (DOS) - Another strategy that the adversary may try is to exhaust resources of a node, e.g., CPU, storage or memory by flooding the network with transactions. If successful the adversary may make the whole network unresponsive and unable to process any legitimate transactions. A mitigation could be to raise transaction fee, limit the transaction rate or to ban misbehaving nodes.

The goal of the attack is to eliminate target node from participation in consensus protocol or to limit target node's ability to send transactions and interact with the application layer. When the block proposing node is the target of the attack then the protocol may perform significantly slower. Furthermore the adversary may get an unfair reward advantage by proposing blocks while slowing down or blocking other block proposers. The committee members may also be the targets of the attack giving the adversary higher proportional voting power.

DOS attack is a threat to PoR and PoS protocols alike but in PoS there is often a block proposer and committee members known in advance making the DOS attack more likely. In the next chapter it is discussed how different PoS protocols mitigate the risk of this attack. Counter measures include whitelisting peering nodes or an anonymization mechanism, e.g., VPN or redirecting the traffic through a DOS protection cloud solution. [27]

Sybil attack – In this type of attack, an adversary creates large number of fake entities trying to act as more than 1 user at a time. In the Blockchain technology this refers to creation of multiple nodes by a single user. The goal of the attack is to get an advantage over honest users. In a permissionless blockchain, an adversary may try to double-vote and influence the committee voting process in in her favor.

In PoS protocols the sybil attack is mitigated, by only permitting voters to vote under certain conditions, e.g., voters need to own a cryptocurrency tokens. Since fake tokens can't be created by an adversary, she is unable to create sybil entities [27].

Compounding of wealth – When node creates a block, and this block is added to the main chain, the node gets a reward which increases her chances of being selected as a leader again in the future. This way nodes that control large amount of particular cryptocurrency are getting even richer at increasing rate leading to concentration of funds. G. C. Fanti et al. [21] showed that possible solution to this problem is to use *geometric reward function*, where rewards increase geometrically over time.

3.6 Proof-of-Authority protocols

Proof-of-Authority (PoA) was first introduced by Dr. Gavin Woods [37], co-founder of Ethereum. In PoA the blocks are proposed by one of the trusted nodes called authority. The authorities aim to reach consensus on the order of transactions created by other nodes and are taking turns in proposing a block. It is assumed that more than 50% of the authorities are honest.

The authorities doesn't stake any coins but they are putting their reputation and identity on the line. This makes the network more scalable because of the lesser amount of messages that need to be exchanged but also because the authorities are known in advance. The cost for the higher performance and scalability is more centralization. The PoA-based protocols are used in permissioned blockchains as the new authorities need to be accepted by the other authorities.

There are similarities between PoA and PoS – instead of coins the nodes are using their reputation and identity which means they are invested in the network success. But the use case for PoA is different than for PoS because PoA aims to solve scalability issues in private blockchains while maintaining some level of decentralization. [36]

3.7 Proof-of-Importance protocols

Proof-of-Importance (PoI) protocols are similar to PoS but the consensus power of each node is not dependent on node's stake alone but also on the importance of the node. PoI is used by NEM [32] where to determine the consensus power of a node *importance score* is calculated. The score depends on node's stake and on node's transactions. Transactions that affect the importance score can only be from a recent past, e.g., about 30 days in NEM. The more transactions with higher value the node executed the higher is her importance score.

The goal of the protocol is to bring more decentralization and limit the concentration of funds where a few largest stakeholders are getting all the block rewards and grow even larger. [2]

3.8 Delegated Proof-of-Stake protocols

Delegated Proof-of-Stake (DPoS) is variation of a pure PoS mechanism. DPoS introduces group of N delegates that sign (*forge*) the blocks. Delegates are voted on by every transaction in the system. Daniel Larimer, developer of BitShares [5] - first cryptocurrency using DPoS mechanism - argues that by using decentralized voting DPoS is more democratic than comparable systems [6].

Probably the biggest drawback of DPoS in current projects (e.g., EOS [20], Lisk [29], BitShares [5], TRON [53]) is centralization of delegates. For example, when Lisk was launched, there quickly emerged groups that were doing everything they could to stay delegates for as long as possible to gather *forging rewards* (refers to fees for creating/mining a block in DPoS system) [30]. This led to low turnout of delegates. Properties associated with this problem are *kickback payments* and voting fees. When fees for participation in voting process are too high, many accounts with low balance have no incentive to vote leaving the whales (accounts with large amounts of funds) with influence (on the voting process) higher than proportional to their wealth.

Kickback payments are payments that voters receive from a delegate they voted for if it is elected. The idea is that delegate sends portion of the forging rewards to its voters. However, the delegate can send the kickback payments only to voters who voted with all their votes in its favor. Which again leads to low fluctuation of delegates undermining the principles of decentralization.

3.9 Liquid Proof-of-Stake protocols

Liquid Proof-of-Stake was first used in Tezos [50]. It is very similar to DPoS but any user can become a validator if she has enough coins. If she doesn't, she can delegate the voting rights to someone else. The goal is to offer more decentralization and rotation of the validators as opposed to DPoS. Switching of the delegate is fast making it easy to delegate vote rights to someone else, e.g, in a case when the delegate is misbehaving. The reward is still proportional to the validator's stake. The key differences are displayed in Table 3.1.

A disadvantage of LPoS over DPoS is that the validators are not likely to have a server grade equipment which makes the scalability more challenging.

	Liquid PoS (Tezos)	Delegated PoS
Delegation (Purpous)	Optional (minimizes dilution of small token holders)	Required to elect block producers (enables greater scalability)
Validator set	Dynamic (Size not fixed)	Fixed size (21 in EOS, 101 in Lisk, 27 TRON)
Design priorities	Decentralization, accountable governance, security	Scalability and usable consumer applications

Table 3.1: Key differences of LPoS and DPoS protocols [51].

Chapter 4

Analysis of existing PoS systems

This chapter presents analysis of some existing PoS projects and their comparison. Focus is on their properties related to the security and real-world usage.

4.1 ProPoS

ProPos [39] is PoS protocol designed for cryptocurrencies and other blockchain projects. ProPoS tries to solve high communication overhead, high reward variance and long confirmation times. Users choose the block they commit to (and with that also the chain) while acknowledging all the branches from the main chain (forks). Users are then able to calculate the probability that the block will be reverted. ProPoS is using GHOST protocol [25] that calculates (relaxed) PoW weight of a given chain considering all its forks. The advantage as the authors put it is as follows. *"As different chains within a subtree support the same chain prefix, an advantage of combining GHOST with the most-stake rule is that it requires an adversary to compete with the entire subtree and not only its main chain, which makes attacks on safety much more difficult."*

Process of creating a block is divided into rounds as follows. In the first round the committee of voters is elected. This is done by sampling all stake units in the system using pseudorandom function (PRF). The size of the voting committee elected is parameterized. In the example in ProPoS whitepaper the $\frac{1}{10}$ of the total stake is used. Voters can cast as many votes as they have stake units in the sample. Each vote supports chosen block. Then in second round the leader is elected. The leader collects the votes and includes them with the transactions in a block and includes the block in the chain with the biggest weight.

There is incentive for voters to cast the votes during the voting phase and not later because late votes are marked and not rewarded. Although leaders still include such votes to strengthen the main chain. Similarly, the leader has incentive to publish the block in the second round and not later because the block would not be included in the main chain and the leader would lose on her rewards.

Creators of ProPoS proofed that probabilistic safety is satisfied. In other words when user commits to a block the probability of overturning the block is below probability p chosen by the user. The proof is based on statistical *hypothesis testing*. In ProPoS checkpoints are not determined globally in the blockchain but clients decide which blocks they consider as checkpoints. Block is a checkpoint when probability of overturning it is below some predefined small threshold. Then the only way of reverting the checkpoint is manual reset of the client.

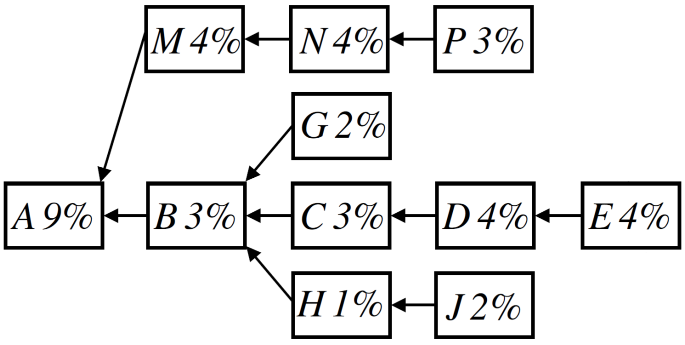


Figure 4.1: Forks in Propos [39, p. 5].

ProPoS is a fair protocol as shown in the whitepaper [39] in the sense that voters rewards and leader's rewards are proportional to their stake.

Authors of ProPoS conducted testing of the protocol in real world environment by deploying 100 nodes in different geographical locations. Every node peered with up to five peers. In each round all 100 nodes were voters, and one node was selected as leader. The size of the block was set to 2 MB. They experimented with different waiting times for votes and blocks. In the optimal setting they achieved throughput of over 2400 tps. As the authors note this was achieved while not optimizing the network by techniques like efficient message dissemination or geographical peer selection.

Possible attacks and their mitigations

Because of the use of PRF, the members of the committee are known in advance, which make it possible to launch a DOS attack. This risk is mitigated by using an anonymization layer. This problem is discussed by the authors. They suggest to use a lightweight network-level anonymity solution like Dandelion[18].

Nothing at stake attack is mitigated in ProPos by punishing the adversary that votes for more branches or if she votes for a weaker branch. An adversary that publishes more than 1 block with the same height is also punished by losing her stake. To incentivize the reporting of a misbehavior, the first user that reported the adversary is rewarded.

Because stake is needed in ProPos to vote and create blocks, a sybil attack on consensus algorithm is not possible.

To eliminate grinding attack, ProPos is using a random beacon to elect round leaders and voters. To construct a random beacon, the authors used approach by Dian et al. [46] where beacons are generated from random values in k previous blocks. It is noted in the ProPos whitepaper [39] that other approaches could be used.

Long range attacks are eliminated by using checkpoints. The checkpoints are not globally determined in ProPos but every user can choose at which probability of reverting is a block considered as checkpoint. This value might be pre-defined in the client's software.

4.2 Hedera

Instead of Blockchain, Hedera uses hashgraph structure to store all events in the network, Figure 4.2. Nodes are using gossip protocol to spread the information about their view of the hashgraph to others. In Hedera whitepaper [26], the communication mechanism

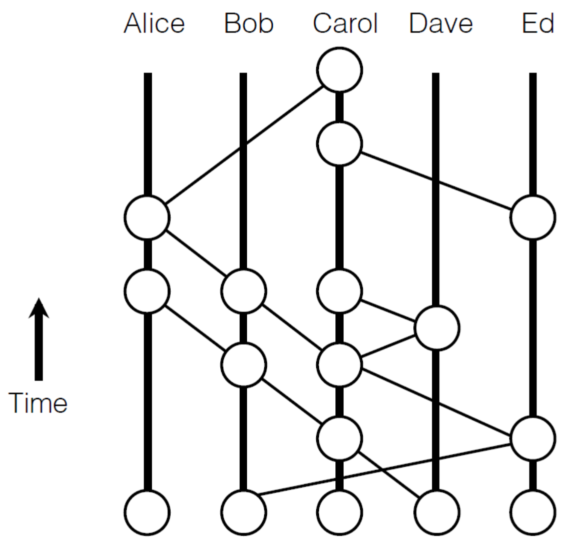


Figure 4.2: Directed graph showing gossip history. Vertices are representing gossip events. When Alice gossips to Bob, this event is represented as vertex in Bob's column with two downward edges to the preceding (parent) gossip events [26, p. 74].

is referred to as *gossip about gossip*, because the information being gossiped is the history of the gossip itself. There is no leader as in other blockchain solutions. Each node can create an event containing data (e.g. a transaction).

Idea behind the operation of the system as in Hedera whitepaper [26, p. 23] is following: *All nodes maintain a copy of the state. For instance, each node knows the balances of all network participants' crypto accounts. At the end of each round of the algorithm, each node calculates the new state by processing all transactions that were received in that round and before. Each node then digitally signs a hash of that shared state, puts it in a transaction, and gossips it out to the network. Then it collects those signatures from all other nodes.*

Definition 1. *An event x can see event y if y is an ancestor of x, and the ancestors of x do not include a fork by the creator of y. [26]*

Definition 2. *An event x can strongly see event y if x can see y and there is a set S of events by more than 2/3 of the members such that x can see every event in S, and every event in S can see y. [26]*

Each node creates an event when she wants to send new transaction to the network or when learned something new from another node. Each event has a round number assigned when it was created (this number may differ from round number when event was *received* decided by consensus). When newly created event in round r can see witnesses with stake more than $\frac{2}{3}$ of all stake then it is marked as witness and its round is set to $r + 1$. Witness is the first event created by a node in a round. This way the algorithm progresses forward with no need for nodes to send each other messages about start or end of a round. All nodes will get the information eventually through gossip protocol.

Witness is decided to be famous when many nodes see it by the start of the next round. Famous witnesses are used to derive consensus order of all other events in given round and subsequently order of transactions.

The decision of fame of a witness is made by each node using virtual voting. Virtual voting means that node calculates what votes would all other nodes have sent based on its current view of the hashgraph. No real messages need to be transferred. For every witness a node is periodically trying to decide if it is famous or not until supermajority of more than $\frac{2}{3}$ of weighted votes is reached and the decision is made.

An event is said to be „received“ in the first round where all the unique famous witnesses have received it, if all earlier rounds have the fame of all witnesses decided. Its timestamp is the weighted median of the timestamps of those events where each of those members first received it [26, p. 83]. Weighted median can be thought of as non-weighted median but all nodes' contribution to the data set is proportional to their stake.

A malicious node could try to create a fork by creating two or more events that has the same parent event. This kind of fork is quickly resolved thanks to concept of strong seeing. Formal proof can be found in [26, p. 84].

Hashgraph is fair because there is no leader node determining the consensus timestamp. Fair access is ensured thanks to the random nature of the gossip protocol. And the consensus timestamps are voted on so they have all the guarantees of being Byzantine.

Throughput of the Hedera network was tested by its authors using different number of Amazon AWS m4.4xlarge instances hosting hashgraph nodes in different locations. In Figure 4.3 trade-offs between throughput, latency, number of computers, and geographic distribution can be seen. The computers were spread across the globe in 8 different locations (Virginia, Oregon, Canada, Sao Paulo, Australia, Seoul, Tokyo and Frankfurt).

Possible attacks and their mitigations

There is neither voting nor proposing of blocks in Hedera. Still sybil attack could hurt the system. But the attack is prevented because to construct a hashgraph and confirm and decide the order of the transactions events are weighted by the stake of their creators.

For long range attack, an adversary could create a fork originating in an arbitrary distant past and build the hashgraph from there. But there is an address book which contains public keys of all the nodes and their stake. Every new address book needs to be signed by nodes that control more than 2/3 of the stake. The genesis address book is predefined. This leads to a sequence of address books where each is signed by the nodes from previous address book. This prevents an adversary to create a fork in the past because she would not have enough stake to continue building the hashgraph. An adversary would need to create a new genesis address book and create a hashgraph from the very beginning. But other nodes would not accept this new hashgraph as the genesis address book is predefined.

Because Hedera doesn't use a blockchain and the longest chain rule as the other protocols, there is no room for nothing at stake attack.

Also there is no committee or leader selection. The voting is virtual and the famousness of an event is decided by the view of the hashgraph of other nodes. This rules out a possibility of grinding attack.

4.3 Algorand

Algorand [1] is a cryptocurrency system designed for low latency and great scalability. Algorand tries to address many issues seen in other PoS projects such as slow fork resolution, possibility of DoS attack on consensus nodes and high time to finality. Experimental results suggests that Algorand confirms transactions with latency on the order of a minute.

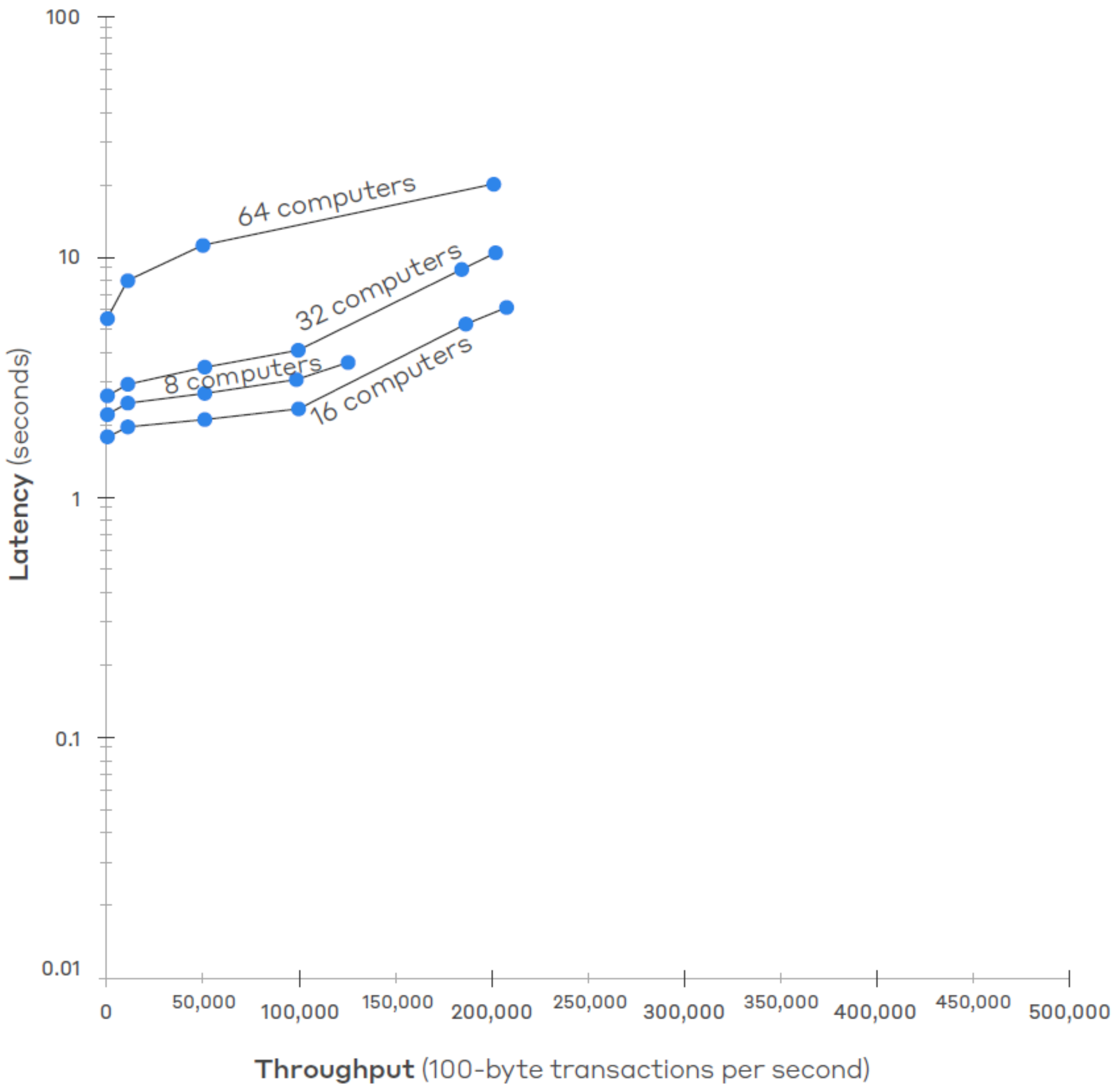


Figure 4.3: Hashgraph Latency vs Throughput 8 regions, m4.4xlarge [26, p. 15]

Algorand overview

Algorand nodes communicate using *gossip protocol.* To reach consensus, Algorand uses Byzantine agreement protocol called $BA\star$. The algorithm progresses in rounds and in each round new block is appended to the blockchain. Every round is divided into steps.

Each user may be selected for one or more roles. The roles are: *committee member* and *block proposer*. At the start of each step of a round, user determines if she was selected (user may be selected for multiple roles). Committee size has to be chosen to achieve a reasonable trade-off between liveness, safety and performance (in the experiments with 50,000 users, the authors choose committee size of 10,000 users for final step and 2,000 users for all other steps). The number of block proposing users in each step is between one and 70 with very high probability [1, p. 57]. Probability of a user to be selected is proportional to her stake. The selection process happens locally without any communication between peers needed utilizing verifiable random functions (VRFs) [56]. Thanks to VRFs, the outcome of the selection can be verified by all other users.

Users always listen to gossip and collect all the transactions in case they are selected to propose a block. Users that are selected to propose a block gossip the block header first which is small compared to whole block making the propagation time smaller. The block header contains priority and proofs of selection. Then the block with pending transactions is sent. Since the selection is random, more users can be selected. In that case, priority is used to select only one block from all blocks proposed. If consensus is not reached, then another step is initiated and new committee is selected. When blocks are proposed, Algorand users discard messages about blocks that do not have the highest priority seen by that user so far to minimize unnecessary gossip communication.

Note that for every step new committee members are selected and members only send one message (containing proposed block, priority and proofs). This approach mitigates DOS attacks, as the committee members are not known in advance.

In each step each member can reach either *tentative consensus* or *final consensus.* When a member reaches final consensus, every other member that reached final or tentative consensus in that round must agree on the same block. Final consensus is reached when more than $T \cdot \tau$ members agree on the same block. Where τ is expected number of nodes selected as committee members and $T > \frac{2}{3}$ is a fraction that defines voting threshold.

Final consensus means that $BA\star$ will not reach consensus on any other block for that round. Tentative consensus means that $BA\star$ was unable to guarantee safety, either because of network asynchrony or due to a malicious block proposer [1, p. 60].

The random selection of new committee members and block proposers is based on node's private key *sk* and a *seed*. In round r there is publicly known $seed_r$. Every committee member computes $seed_r$ from $seed_{r-1}$ and her private key *sk* using VRFs. $seed_r$ is then distributed with the proposed block. The random selection of user's role is based on node's private key *sk* and $seed_{r-1-(r \mod R)}$. This means the seed used in selection algorithm is refreshed every R rounds.

Detailed look at $BA\star$

Algorithm 1 shows pseudocode of $BA\star$. There are 2 importatnt procedures - Reduction() and Binary$BA\star$(). In Reduction(), consensus is reached either on the proposed block with highest priority seen by the majority of committee members or on an empty block. Reduction() is a two steps process. In the first step, committee members vote for highest priority block. Then in the second step, they vote for the block that received at least $T \cdot \tau$ votes in the previous step. Reduction() ensures that there is at most one non-empty block that can be returned by Reduction() for all honest users [1, p. 59]. Then in the Binary$BA\star$(), shown in Alg 3, consensus is reached on the block passed to Binary$BA\star$() or an empty block. Reduction ensures that all honest users pass maximum one (all users the same) non-empty block to Binary$BA\star$().

Let's break down ideal case scenario when the network is not partitioned and supermajority of users are honest. As already stated, new committee of users is selected for every voting. The voting occurs two times in Reduction() and then five times in Binary$BA\star$(). That is seven vote casting in total. Four out of five vote castings in Binary$BA\star$() algorithm occur at the same time. Which means these votes can be sent in one message reducing the number of messages gossiped through the network. The total number of messages with votes sent in a round is $(2 + 1 + 1) \cdot \tau$.

Algorithm 1: $BA\star$(ctx, round, block)

```
$hblock \leftarrow Reduction(ctx, round, H(block))$
$hblock_\star \leftarrow BinaryBA\star(ctx, round, hblock)$

$r \leftarrow \text{CountVotes}(ctx, round, FINAL, T_{FINAL}, \tau_{FINAL}, \lambda_{STEP})$
if $hblock_\star = r$ then
    return $\langle FINAL, BlockOfHash(hblock_\star)\rangle$
else
    return $\langle TENTATIVE, BlockOfHash(hblock_\star)\rangle$
```

Attack vectors

When adversarial stake is below $\frac{1}{3}$ (and reasonable assumptions about the network are met) Algorand guarantees safety and liveness. Mitigation of some of the attack vectors are described in the next section. However, what adversary can achieve is slowing down the progress of the consensus protocol making block time and time to finality longer. When adversary is chosen to propose a block, she gossips only the header with block priority and proofs and does not send the actual block. If the block's priority happens to be the highest among all proposed blocks, then users reach consensus on the adversarial block using only the information from the header and are waiting for the block itself. The timeout for block reception used in the Algorand whitepaper[1] is 1 minute. After the timeout, there is a fallback to an empty block. This way the adversary can slow down the progress of Algorand significantly even with stake smaller than $\frac{1}{3}$. Results of experiments with this type of attack are in Section 7.1.

Mitigation of some attack vectors

Computing $seed_r$ requires that every user's secret key sk_u is chosen well in advance of the selection seed used in that round, i.e., $seed_{r-1-(r \mod R)}$ [1, p. 56]. This is done to limit adversary's ability to manipulate the committee selection process and eliminate grinding attack.

When network is partitioned, the adversary can prevent reaching a consensus in some part of the network and force users to agree on empty blocks. In that case the $seed_r$ is computed using cryptographic hash function. Adversary can then compute seeds for future rounds in advance and choose secret keys so that she has higher probability of being selected as a committee member or block proposer (or both).

To mitigate such attack following mechanism is described in Algorand whitepaper [1, p. 56]: *Algorand relies on "weak synchrony" assumption (In every period of length b, there must be a strongly synchronous period of length $s < b$). Whenever user runs committee selection algorithm, Algorand checks the timestamp included in the agreed-upon block for round $r - 1 - (r \mod R)$, and uses the keys (and associated weights) from the last block that was created b-time before block $r-1-(r \mod R)$. The lower bound s on the length of a strongly synchronous period should allow for sufficiently many blocks to be created in order to ensure with overwhelming probability that at least one block was honest. This ensures that, as long as s is suitably large, an adversary u choosing a key sk_u cannot predict the seed for round r.*

Algorithm 2: Binary$BA\star$(ctx, round, block_hash)

```
$step \leftarrow 1$
$r \leftarrow block\_hash$
$empty\_hash \leftarrow H(Empty(round, H(ctx.last\_block)))$
while $step < MaxSteps$ do
    CommitteeVote$(ctx, round, step, \tau_{STEP}, r)$
    $r \leftarrow$ CountVotes$(ctx, round, step, T_{STEP}, \tau_{STEP}, \lambda_{STEP})$
    if $r = TIMEOUT$ then
        $r \leftarrow block\_hash$
    else if $r \neq empty\_hash$ then
        for $step < s' \leq step + 3$ do
            CommitteeVote$(ctx, round, s', \tau_{STEP}, r)$
        if $step = 1$ then
            CommitteeVote$(ctx, round, FINAL, \tau_{FINAL}, r)$
        return r
    step++

    CommitteeVote$(ctx, round, step, \tau_{STEP}, r)$
    $r \leftarrow$ CountVotes$(ctx, round, step, T_{STEP}, \tau_{STEP}, \lambda_{STEP})$
    if $r = TIMEOUT$ then
        $r \leftarrow empty\_hash$
    else if $r = empty\_hash$ then
        for $step < s' \leq step + 3$ do
            CommitteeVote$(ctx, round, s', \tau_{STEP}, r)$
            return r
    step++

    CommitteeVote$(ctx, round, step, \tau_{STEP}, r)$
    $r \leftarrow$ CountVotes$(ctx, round, step, T_{STEP}, \tau_{STEP}, \lambda_{STEP})$
    if $r = TIMEOUT$ then
        if $CommonCoin(ctx, round, step, \tau_{STEP}) = 0$ then
            $r \leftarrow block\_hash$
        else
            $r \leftarrow empty\_hash$
    step++
```

Another possible attack the authors outlined in the whitepaper is that consensus could get stuck if the honest users are split into two groups and are voting for different blocks. Neither group is large enough to reach consensus on their own. Adversary can then make any user vote as she wants by sending the user adversary's votes just before the timeout expires to cross the threshold for reaching consensus or not sending any votes to the user at all and let the timeout expire. To mitigate this attack, *common coin* mechanism is used. When user's timeout expires *common coin* is used to choose whether user commits to the block with highest priority or empty block. The binary value of the coin is predominantly the same for all users. This way adversary cannot know how other users vote in advance when the timeout expires.

The sybil attack is eliminated by choosing commitee members and block proposers based on their stake by VRF.

Long-range attacks are often mitigated by a checkpointing mechanism. There are no checkpoints in Algorand but because the $BA\star$ algorithm requires a sufficient fraction of stakeholders to vote in each step, LRA is not feasible [47].

Nothing-at-stake attack in Algorand is not possible because the finality is reached in every round. There are no forks that adversary could vote for simultaneously.

Forks resolution

When network is not partitioned for some time, forks may be created. This does not violate safety as described in Algorand whiepaper [1, p. 62].

To resolve these forks, Algorand periodically proposes a fork that all users should agree on, and uses $BA\star$ to reach consensus on whether all users should, indeed, switch to this fork [1, p. 62]. The fork proposing process is similar to block proposing and follows *recovery protocol* [1, p. 62].

In order for $BA\star$ to reach consensus on one of the forks, all Algorand users must use the same seed and user weights. This means that Algorand must use user weights and seeds from before any possible forks occurred [1, p. 62].

As mentioned before Algorand relies on the weak synchrony assumption. Meaning in every period of length b (e.g. 1 day), there must be a strongly synchronous period of length $s < b$. Using this assumption Algorand takes the seed from the most recent block from the next-to-last complete b-long period and weights from the last block that was agreed upon at least b-long time before it [1, p. 62].

Algorand performance

Latency of transaction confirmation was tested by authors [1, p. 64] using 1,000 Amazon's EC2 m4.2xlarge virtual machines, each of which had 8 cores and up to 1 Gbps network throughput. 50 users run on each virtual machine and users proposed 1 MB blocks. The results can be seen in the Figure 4.4. More tests were conducted with much bigger blocks. With increasing block size the confirmation time raises very slowly as shown by Figure 4.5.

4.4 Ouroboros

Ouroboros [35] is a PoS protocol, where the authors focus on formal specification of its properties and prove the protocol has these properties under certain assumptions. Ouroboros is currently used by cryptocurrency project Cardano [11]. However, many features are yet to be implemented.

Ouroboros overview

Ouroboros protocol is proceeding in slots (rounds). There is a block produced in each slot. Slots are grouped into epochs. All epochs have the same (parameterized) length (number of slots). The block producer for each slot is elected by stakeholder committee. The committee

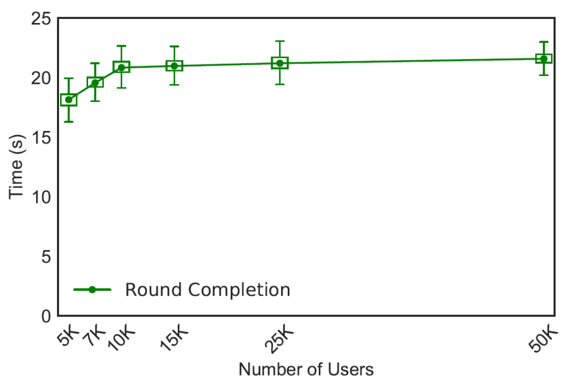


Figure 4.4: Latency for one round of Algorand, with 5,000 to 50,000 users. [1, p. 66]

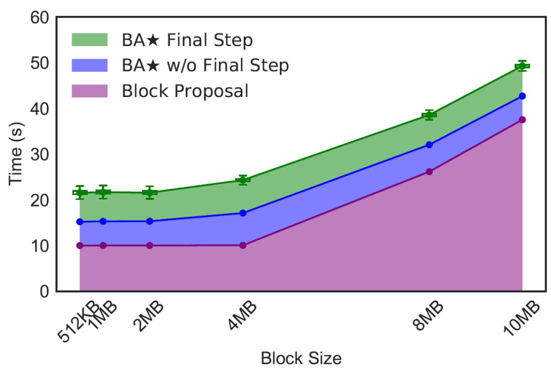


Figure 4.5: Latency for one round of Algorand as a function of the block size. [1, p. 66]

is refreshed every epoch. The probability of a stakeholder to be elected is proportional to its stake.

During each epoch the leaders are participating in coin-flipping mechanism based on public verifiable secret sharing (PVSS) [41]. This way at the end of the epoch common random string is generated. The string is then used in subsequent epoch for committee and leaders election. This also means that slot leaders for an epoch are known ahead of time.

Leaders are responsible for publishing blocks only but not for data they contain. The transactions and other data are handed to the leaders by *input endorsers*. Input endorsers are elected in the same way as the slot leaders. In each slot, the slot endorser endorses input to be included in the next slot. The advantage is that input endorser may endorse data from d previous input endorsers (d previous slots) that were not included in any block yet. Where d is a parameter of the protocol [35, p. 52]. In case the input endorser fail to endorse an input the slot leader publishes an empty block.

The number of messages related to PVSS is increasing with c^2 where c is the size of the committee. To limit the committee size (and communication complexity) there is a minimum stake requirement for users to be committee members. On the other hand, the committee size needs to be large enough so that when users are elected for committee randomly, there is still more than $\frac{1}{2}$ of honest committee members with overwhelming probability [35, p. 52].

In Ouroboros stake delegation is possible. When user's stake is not large enough or she simply doesn't want to participate in the committee, the stake can be delegated without losing control of the coins.

The authors of Ouroboros are focused on two main properties, liveness and persistence (finality) [35, p. 2]. In order for protocol to have these and other desired properties, the portion of adversaries must remain strictly below $\frac{1}{2}$. Note that in most PoS protocols the limit is $\frac{1}{3}$ of all users.

Possible attacks and their mitigations

The forking attacks are very limited in Ouroboros because the probability of successful fork is decreasing exponentially with the length of the fork (if there is $< 50\%$ of adversarial users) [35, p. 18].

Grinding attacks are not possible because the coin tossing protocol based on PVSS ensures unbiased uniform randomness as discussed in [35, p. 43]. For the same reason the sybil attacks are mitigated.

51% attack is possible when the adversary holds more than 50% of the overall stake (or the stake is delegated to her). This is better than in case of PoS protocols discussed earlier where the threshold is only $33,\overline{33}\%$.

Long range attack is mitigated by slot leaders rejecting blocks generated far ahead of time. Also when the longest-chain rule is applied the depth of newly adopted chain is limited. So the users do not adopt chain generated by adversary all the way from genesis block.

Nothing at stake attacks is mitigated by ignoring very deep forks. In addition to that the authors put forward a proof that adversary cannot successfully fork and mine on multiple chains see [35, p. 18, p. 60].

Ouroboros performance

Ouroboros performance was tested by its authors and the results can be found in the whitepaper [35, p. 61]. Various parameters of the protocol were set to get optimal performance but in different setting, the results may vary. Transaction confirmation time is within 5 minutes with 10% adversarial stake.

The experiments were conducted using 10, 20, 30 and 40 nodes (Amazon's EC2 c4.2xlarge instances) with similar results. Setting different slot durations were also not significant. With 40 nodes and slot duration of 5 seconds, the network achieved median value of 257.6 transactions per second [35, p. 63].

4.5 Tezos

Tezos [50] is a project focused on encouraging users to be part of a governance of the system. The consensus algorithm of Tezos is Liquid Proof-of-Stake (LPoS) which is very similar to DPoS but the number of delegates is not fixed. Authors of Tezos claim that this feature allows for greater decentralization.

To choose a node that has the right to create a block Tezos uses follow the coin strategy. To improve efficiency of follow the coin procedure, coins are grouped into rolls and information about the rolls ownership is stored in a database. A roll is just a certain amount of coins. A user can own more rolls. When the coins move between users, a roll can be broken down. But whenever there is enough coins in the user's account, the roll is formed again.

Blocks are grouped into cycles and each cycle consists of 2048 blocks. Target block time is 1 minute. Duration of the cycle is therefore around 34 hours. When creating a block, the producer has to lock a certain amount of tokens for next 7 cycles. These tokens are a security deposit that can be forfeited in a case when the creator mints more than 1 block with the same height. Creator of any next block (within the 7 cycles window) can denounce this misbehavior by including a proof of double signing in her block and receive a reward. The reward is not amount of the full security deposit but only a portion of it. This prevents an adversary to denounce her own misbehavior when it is exposed by another user and try to get the reward first in a compensation of the security deposit lost.

Blockchain protocol of Tezos is composed of 3 distinct protocols [50]:

- The network protocol that broadcasts transactions and blocks.

- The transaction protocol that specifies what makes a transaction valid.
- The consensus protocol that is responsible for reaching a consensus on blocks and transactions.

This way the consensus protocol is separate from the network shell and can be changed or replaced by another protocol in the future as long as the interface to the network shell is correct. Users can amend the consensus protocol by proposing and voting for amendments. For example it is possible to replace the current PoS protocol by a PoW protocol. Even the voting procedure itself can be changed by an amendment. Authors of Tezos argue that this mechanism is better than to create forks when the core of the protocol is updated.

Similarly to Ouroboros there is a focus on the formal verification and correctness of smart contracts and their efficiency. These properties are satisfied by a functional programming language OCaml [33] used by Tezos. OCaml is fast and has unambiguous syntax. To make implementation more lightweight and to minimize errors, the operations have only access to standard library and may not make any system call.

Possible attacks and their mitigations

Tezos is immune to Sybil attacks because of the follow the coin strategy. To get voting power a stakeholder needs to own the coins or be a delegate with other user's voting rights.

A countermeasure for nothing at stake attack is the punishment for double signers as was already discussed. This approach is similarly to the Slasher algorithm [45].

Grinding attack on Tezos should not be possible because the randomness for follow the coin mechanism is gathered from all the blocks in last cycle. A cycle consists of 2048 blocks making it extremely expensive to influence the seed.

Forks are mitigated by using a clock mechanism that dictates when a user can publish a block. This mechanism ensures that on a branch with small fraction of users, the block rate is very low. More details can be found in the Tezos whitepaper [50, p. 11].

4.6 Comparing performance of selected POS systems

	Throughput	Scalability	Liveness	Time to finality
Propos	2,400 tx/s (~360 kB/s)	Medium	Proven	15 s – 1 minute
Hedera	400,000 tx/s (~40 MB/s)	Very High	Proven	6 s
Algorand	1,000 tx/s (~208 kB/s)	High	Proven	~22 s
Ouroboros	~260 tx/s	High	Proven	5 – 20 minutes
Tezos	~40 tx/s	Medium	–	1 minute

Table 4.1: Comparison of the claimed performance of selected protocols.

The best performing protocol among selected in Table 4.1 in terms of TPS is Hedera. This protocol is designed to minimize the consensus overhead and focuses on the scalability and throughput.

On the other hand Tezos and Ouroboros are focus more on decentralization, security and other capabilities of the protocol so the throughput is much lower and time to finality longer. It is possible that the scalability of these protocol will be improved by future upgrades.

Algorand and ProPos are competing protocols and their properties are similar. Claimed TPS of ProPos is higher than of Algorand but the tests were conducted without an anonymization solution which is needed in the public blockchain to prevent DOS attack. The authors of ProPos propose to use a lightweight network level anonymization.

Chapter 5

Consensus protocols simulation testbeds

When design a blockchain consensus protocol, it is hard to think of all the problems that may arise when it is deployed in the real world, especially when communication over the network is involved. For better understanding, testing and simulation of these protocols, the simulation testbeds are used. Part of this work is dedicated to testing a few PoS protocols and then evaluating and comparing results. In this chapter some of the existing simulation testbeds are described.

5.1 Bitcoin Simulator

Bitcoin Simulator framework [24, 9] is built using ns-3 simulation environment. Its main purpose is to simulate PoW consensus protocols and evaluate their security and performance.

The framework consists of a blockchain instance and a blockchain security model. A blockchain instance is a representation of the PoW blockchain with given consensus and network parameters, e.g., block generation times, network delays, block size, data propagation mechanism, etc. Thanks to this settings, the Bitcoin Simulator is not limited to simulate only Bitcoin but virtually any PoW blockchain. The network layer is simulated as a relay network. To make the simulations more realistic, authors used data from real Bitcoin network statistics e.g., block size distribution and average number of nodes in the network [9].

The output of the blockchain instance is analyzed in the security model using Markov Decision Processes for double-spending and selfish mining. The analysis takes into account the adversarial mining power, impact of eclipse attacks, block rewards [24].

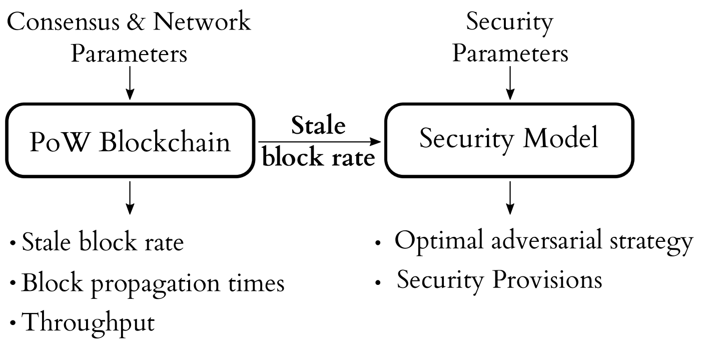
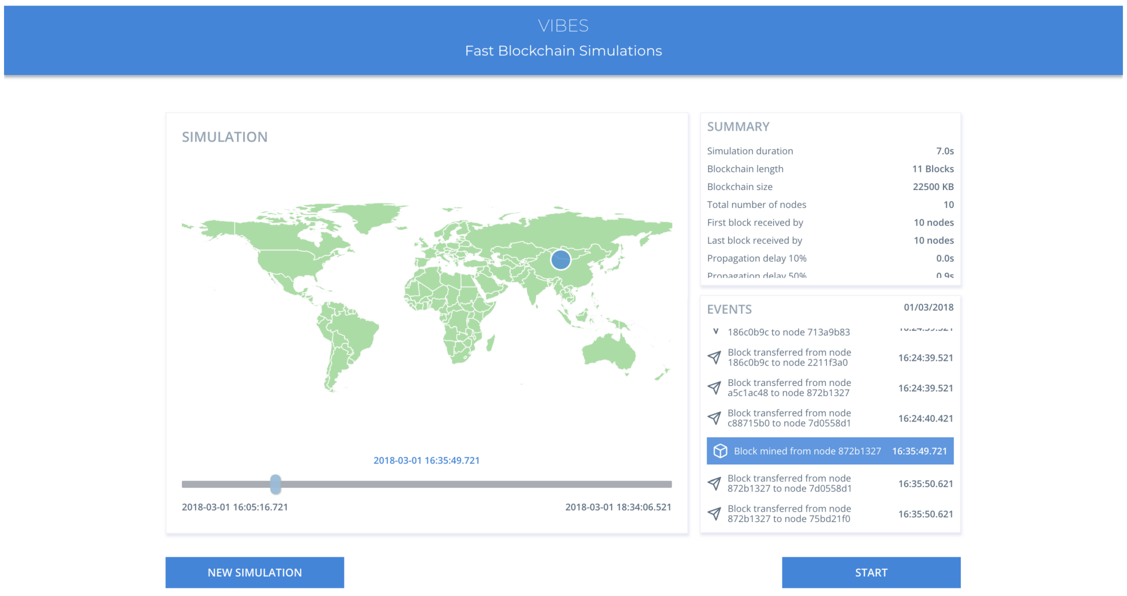


Figure 5.1: Components of the Bitcoin Simulator framework [24].

5.2 Vibes

Vibes [54] is a configurable blockchain simulator for large-scale P2P networks. Vibes simulates blockchain communication protocol on event level. Simulated nodes communicate with each other only by sending messages. The properties and topology of the network is configurable.

The core of the application is written in Scala programming language with use of Akka toolkit. The graphical user interface runs in a web browser. After the simulation the results are visualized graphically. User can view statistics of the simulated network, like throughput, propagation delay etc. as well as all the transactions and blocks. According to the authors, a user can simulate thousands of nodes on high-end powerful PC.



SUMMARY

Simulation duration	7.0s
Blockchain length	11 Blocks
Blockchain size	22500 KB
Total number of nodes	10
First block received by	10 nodes
Last block received by	10 nodes
Propagation delay 10%	0.0s
Propagation delay 50%	0.9s

EVENTS 01/03/2018

186c0b9c to node 713a9b83	
Block transferred from node 186c0b9c to node 2211f3a0	16:24:39.521
Block transferred from node a5c1ac48 to node 872b1327	16:24:39.521
Block transferred from node c88715b0 to node 7d0558d1	16:24:40.421
Block mined from node 872b1327	16:35:49.721
Block transferred from node 872b1327 to node 7d0558d1	16:35:50.621
Block transferred from node 872b1327 to node 75bd21f0	16:35:50.621

Figure 5.2: Vibes interface showing simulation results [55, p. 62].

5.3 Simblock

SimBlock [44] is a blockchain network simulator focused on testing the throughput and performance. User can set network bandwidth and propagation delay, number of nodes, block size, block generation interval. Nodes can be divided into different locations to simulate network delay between geographical regions.

User can also specify behavior of nodes so different consensus algorithms can be simulated. The mining is simulated by generating blocks with the same probability as if the nodes were actually mining.

Simblock is event driven Java based application. One of its limitations is that the simulations are done on block level, not transactions level. On the other hand, this permits for simulating high number of nodes (around 10,000 is still possible). The authors compared their results with an already existing simulator and also with real world data with very similar outputs [44, p. 327].

Chapter 6

Simulation testbed design

The goal of the simulation testbed is to experiment with the protocols and compare them in terms of performance and security. Protocols selected for testing are Algorand, Propos and Hedera. To mimic conditions of real world scenarios, nodes are connected in peer-to-peer style network. Data transmission delayes are modeled using TCP latency approximation from [12]. Meaning the transmission delays are calculated rather than to simulate TCP/IP stack. For more details about TCP model used see section 6.4.

When finding suitable simulation framework, two simulators were considered, ns-3 and OMNeT++. They are both fairly similar, but OMNeT++ was chosen for having better IDE and easier custom module implementation. Simulation experiments are performed using OMNeT++ 5.4.1 on Ubuntu 18.04 machine. On modern PC, the created testbed is capable of running experiments with thousands of nodes.

6.1 OMNeT++ framework

OMNeT++ is a C++ framework for building network simulation models. Its modular architecture allows for assembling models from existing modules. Modules can be simple or compound. Compound modules are created by nesting other modules (simple or compound). Modules communicate with each other by sending messages. Messages can be sent through defined channel or directly if needed. A module can also send a message to itself which is useful for modeling waiting for a specific time interval. For example, if a network node represented by its module needs to wait for 5 seconds and then take some kind of action, it schedules a message to itself for `simulationTime` + 5s. Then when the message arrives, the module performs the action. Every message has its kind and optionally a payload with arbitrary data.

Simulation runs consist of a series of discrete events. All the events are stored in a data structure called FES (Future Event Set). During the simulation run, events from FES are executed in the order of their timestamps. A typical event is an arrival of a message to a module. During the handling of the message, the module can add an event to FES (e.g., scheduling a new message) or remove an event from FES (e.g., canceling a message).

The structure of the simulation model is described in NED language (NEtwork Description). NED is used to specify what modules are present in the network and properties of module's connections, i.e., network topology. Every module itself has a corresponding description in the NED file containing outline of its parameters. The parameters (e.g., number of nodes, connections bandwidth) are used for configuring the network and other modules.

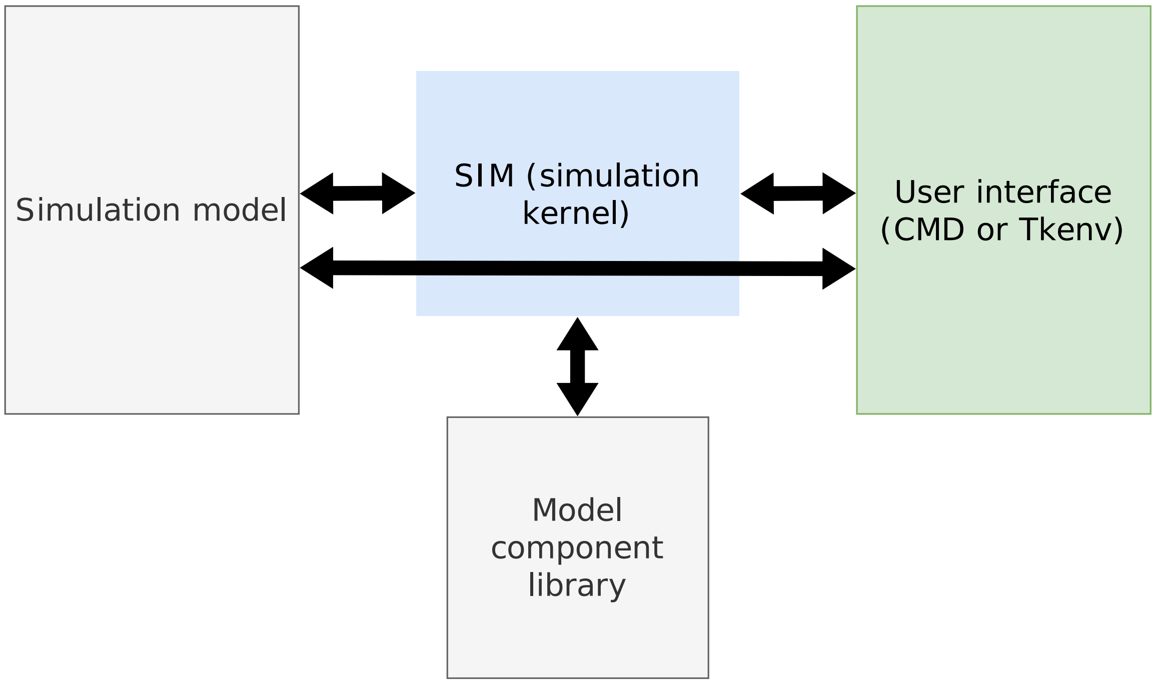


Figure 6.1: High-level architecture of OMNeT++ framework.

User can adjust simulation parameters in the initialization file to perform simulations with specific settings.

OMNeT++ inner structure is displayed in Figure 6.1. The **Model component library** includes C++ implementations for all the simple or compound modules and all classes needed by the modules (e.g., channels, message types, etc.). **Simulation model** is the model that is set up for the simulation. This model contains objects (modules, channels, etc.) that are all instances of the components in the model component library [34].

6.2 Testbed architecture

Testbed architecture can be seen in Figure 6.2. There are essential modules created for running the simulation and collecting the data. Modules are communicating via message exchange with coordination by the OMNeT++ class *Simulation*. Parameters for all the modules are in initialization file *omnetpp.ini*. There are several configurations in the file with different parameters' values for different simulation scenarios. Each protocol has a set of specific parameters, e.g., committee sizes and then there are parameters for the TCP model and for the network.

After each simulation run, an XML file is created. The file contains timestamped events tied to the execution of the consensus protocol, such as block proposal, voting events, appending of a block, etc. This file is then used for calculating statistics used for experiment evaluation in the Chapter 7. The processing of the file is done by a script in Python.

All the source code in C++ class files for modules described below was created as a part of this thesis. For the hash calculation a library Crypto++ was used[17].

Description of testbed modules

Simulation manager - Simulation manager object is an instance of class `cSimulation`. Simulation manager initializes the simulation and creates instance of Network module, manages FES (Future Event Set), simulation time and others.

Network module - Creates all the modules used in the simulation. Number of Node instances can be set in the initialization file. This module holds parameters that are specific for the whole simulation and not a single Node, e.g., number of faulty Nodes, number of adversarial Nodes, total amount of stake etc.

Node - Each Node behaves as an independent instance of the application of interest (Algorand, ProPos, Hedera) and contains its simplified implementation.

Communication module (TCP model) - Manages data transmissions between Nodes over the network. Uses TCP model to simulate TCP connections. Messages between Nodes always go through Communication module. The module offers direct message sending (with no delay) for special control messages.

Statistics collector - This module is used by other modules to save statistical data. At the start of the simulation, Statistics collector creates XML file and saves parameters specific for the current run (from omnetpp.ini file). OMNeT++ provides tools for collecting statistics already, but these are limited and are not suitable for complex data recording such as view of a blockchain for specific Node.

Topology manager - Dynamically creates connections between Nodes at the start of the simulation. This process happens after all the Nodes are initialized.

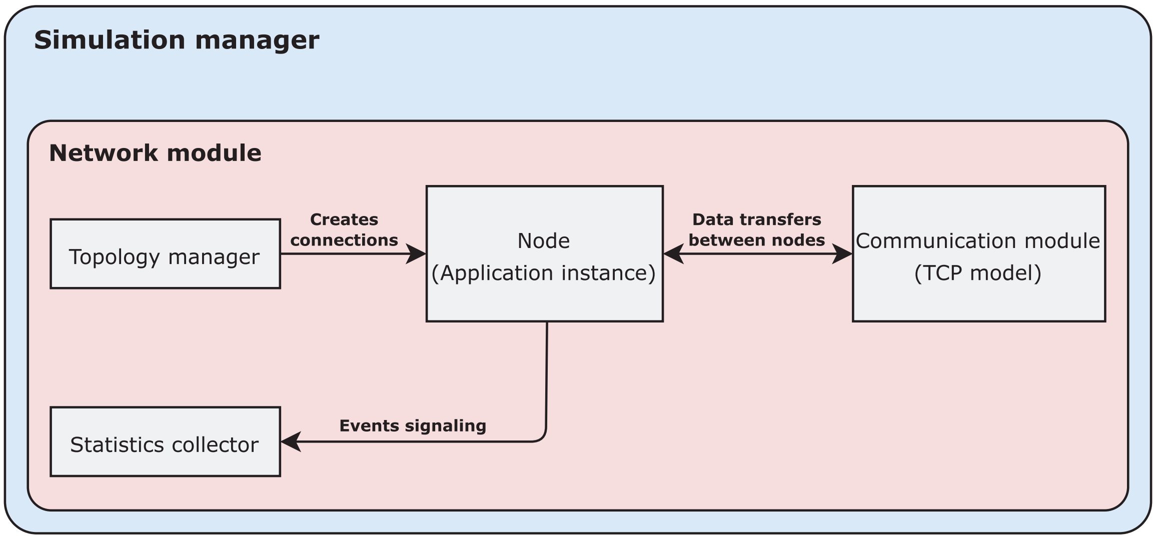


Figure 6.2: Simulation testbed architecture.

6.3 Network

When simulating PoS protocols, Nodes are connected in peer-to-peer style network. The topology can be described in the NED file of the Network module, but in the created testbed the connections between Nodes are created dynamically instead. This approach allows for greater control over the topology.

The connections are generated pseudo-randomly by Topology manager module. Seed value for the pseudo random number generator is parameterized. This way the same network can be generated for multiple simulation runs. The number of peers is also parameterized. Created network is a simplistic version of peer-to-peer network without any switches or routers. All the connections are made equal with the same throughput, latency, etc. and they do not change during the simulation.

The process of creating a network in Topology manager can be described as follows. Topology manager iterates over all the nodes in order and for every node picks another random node and makes the connection. If the picked node already has specified number of peers then another node is picked at random and the connection is created. Topology manager always connects nodes without creating a network partitions.

6.4 TCP latency modeling

Parameter	Meaning	Value
LossRate	loss rate of the transmission	0.0001%
DelayACK	ACK delay for first segment	100 ms
MSS	Maximum Segment Size	1448 B
w_{max}	maximum window size in segments	1000
w_1	initial window size in segments (at the start of slow start)	2
γ	rate of exponential growth of the congestion window	2
RTT	Round Trip Time	10 ms
t_0	average duration of the first timeout in a sequence of one or more successive timeouts	500 ms

Table 6.1: Description of TCP model parameters with values for achieving transfer speed of 1 Gb/s

The TCP latency approximation model from [12] was used to model data transfer using TCP/IP. The model is calculating data transfer delay for a data of given size. Tests conducted in the original paper suggests the model is accurate for short and long TCP data flows. The calculation is done by breaking down a TCP data transfer into several stages and calculating delay for each stage separately. Final delay is given by:

$$E[T] = E[T_{ss}] + E[T_{loss}] + E[T_{ca}] + E[T_{delack}]$$

where $E[T_{ss}]$ is expected delay of TCP slow start phase, $E[T_{loss}]$ is the expected additional delay caused by retransmissions or fast recoveries that happens at the end of the slow start phase, $E[T_{ca}]$ is the expected time to send the rest of the data after slow start and $E[T_{delack}]$ is the expected delay of the first delayed ACK - it is the time between the arrival of a single segment and the ACK for that segment (100 ms for BSD-derived stacks, and 150 ms for Windows) [12].

All the parameters for the model are displayed in Table 6.1 with values used for modeling data transfer speed of 1 Gb/s. Comparison of the estimated transfer delay and delay observed in real-world tests are in [12, p. 1749].

Chapter 7

Simulation experiments

Decentralized blockchain systems aim to replace centralized solutions in variety of use cases. One of the challenges in doing so is that the decentralized solutions tend to be much slower. The experiments in this thesis were made to check mainly the performance but also the security of the selected protocols. Namely time to finality, throughput and performance were tested under different network conditions and an attack on Algorand.

When testing ProPos and Algorand the experiments were made with 500 nodes, 50 of which were participating in the consensus agreement, i.e., they had a chance to be elected as a committee members for voting or for block proposing. Experiments with Hedera were conducted with 50 nodes only because all nodes in the system are participating in reaching a consensus.

In all the experiments, the nodes were connected to 8 randomly selected peers and the block size was set to 1 MB (there are no blocks in Hedera protocol). More implementation choices are described in the following sections for every protocol separately. The TCP connection parameters for ideal network used in the experiments are in section 6.4.

7.1 Algorand

Implementation logic of Algorand protocol is made the same as in [1]. Namely algorithms $BA\star$, Reduction and Binary$BA\star$ were implemented. Each node is autonomous and stores her version of the blockchain locally. Nodes are communicating using gossip protocol with no aggregations. There is one exception in Binary$BA\star$ algorithm, before final step the node is voting 3 or 4 times in a row at the same time. In the simulation testbed, the node sends only one message carrying all the votes (three or four) as it would in the real application.

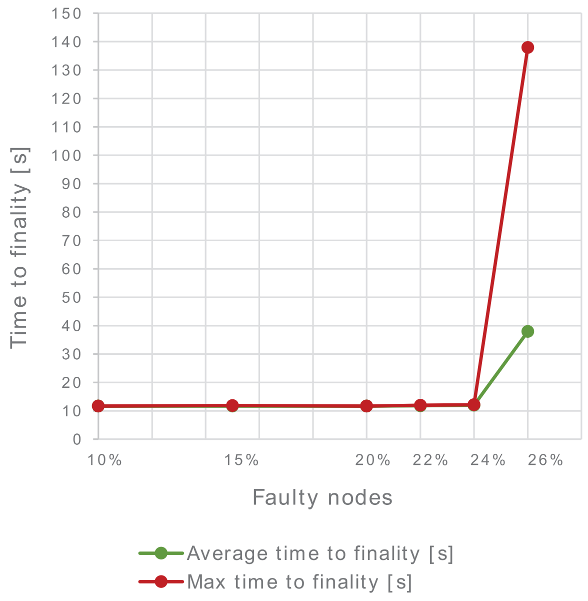
All cryptographic procedures like signatures, proofs, verifications etc. are not implemented because this is not important for the purpose of conducted experiments. Vote message size is set to 355 B, which corresponds to the data the message carries as described in [1, Algorithm 4]. Message with block propose proof is 200 B [1, p. 7]. Transactions are not sent at all. Instead each block is treated as though it contains maximum number of transactions up to the block's capacity.

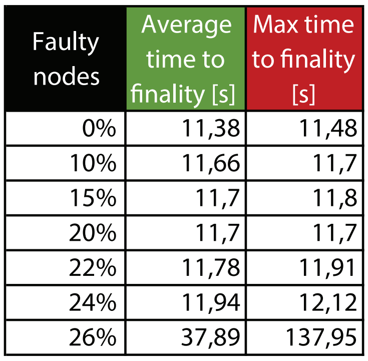
Algorand is using VRF for selecting committee members, block producers and creating block priority. VRF is generating uniformly distributed numbers and is run locally by each node. In the simulation testbed, the same is achieved with C++ generator `minstd_rand0`. The generation is also done by each node individually and the numbers are uniformly distributed. Seed for the generator can be set in the configuration file.

Time to finality with faulty nodes in Algorand

In this experiment, Algorand was simulated with different faulty nodes count. Each node that was set to faulty didn't send any messages to other nodes and discarded every received message. Time to finality was then calculated as a time from one confirmed block (block with final consensus reached) to the next confirmed block. Average and maximum values can be seen in the table in Figure 7.1.

Rounds in Algorand proceeds in steps. In each step 68,5%, of committee votes is needed to proceed and in final step, 74% of committee votes is needed to reach final consensus on a block [1, p. 12]. The results are shown in Figure 7.1. It can be seen that the more faulty nodes there, are the longer it takes nodes to gather enough votes for crossing the threshold of 68,5% or 74%. The time to finality is growing, but when there is less than 26% of faulty nodes, it is on average still under 12s. After the 26% mark, the time to finality is getting very long because it takes several rounds to reach a final consensus on a block and there are many tentative blocks in between. Results of this experiment suggest that Algorand tolerates byzantine nodes very well but only when there is less than 26% of them in the system.





Faulty nodes	Average time to finality [s]	Max time to finality [s]
0%	11,38	11,48
10%	11,66	11,7
15%	11,7	11,8
20%	11,7	11,7
22%	11,78	11,91
24%	11,94	12,12
26%	37,89	137,95

Figure 7.1: Correlation between number of faulty Algorand nodes and time to finality.

In Figure 7.2 we can see how block rate decreases with the increasing number of faulty nodes. There are two metrics, one is a rate of blocks with final consensus and the other is all block including the ones with only tentative consensus. The difference between these metrics can be seen again from 26% of faulty nodes when there is often not enough votes to confirm a block as final, but tentative blocks are still created. Then from 28% of faulty nodes, the protocol is unable to proceed further.

This problem could be solved by detecting offline nodes and changing the pool from which the committee is elected periodically, e.g., every hour. Or by changing the pool when the number of votes in n last rounds is below some threshold and there is a risk that there may not be enough votes in the future rounds.

When the block rate is about 5 blocks per minute, block size is 1 MB and transaction size is 250 B then the throughput of Algorand is ≈ 330 TPS. Note that this is not the best throughput that can be achieved because the experiment was done with 1 MB block size, but larger blocks could be used.

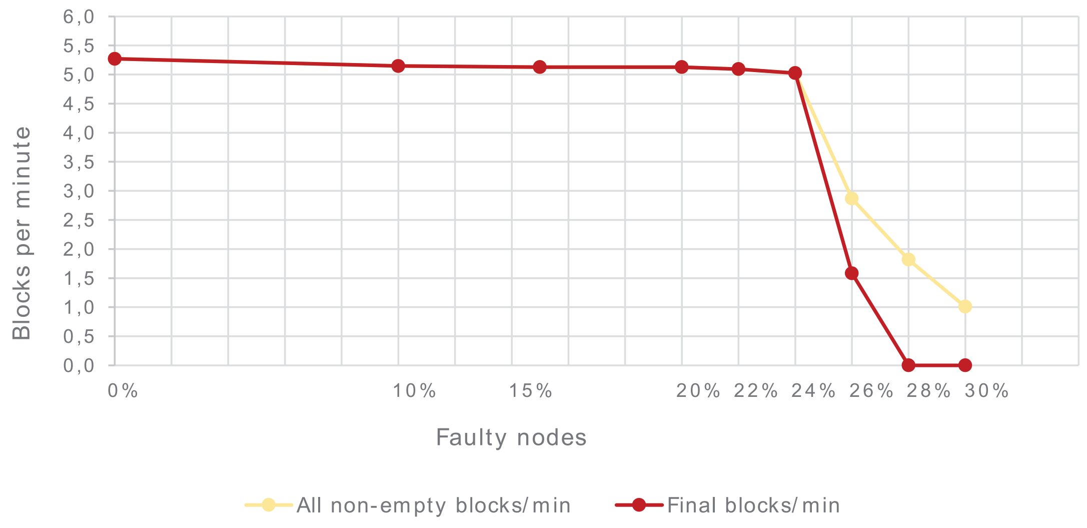


Figure 7.2: Correlation between number of faulty Algorand nodes and block time.

Attacks on Algorand

It is very hard and unlikely to create a fork in Algorand because if the network is synchronous then finality is reached within one round. This makes Algorand resistant against nothing at stake attack and long range attacks. Grinding attack is also mitigated as was already discussed in section 4.3.

In this thesis another type of attack was implemented where an adversary aims to slow down the protocol as much as possible. When the adversary is not chosen as a block producer she is trying to slow down the protocol by not voting for any block other than the adversarial block. When the adversary is chosen as a block producer, it only sends the proof message with block header but never creates and sends the actual block. When other nodes receive the block header, they continue with the round normally and vote for the adversarial block. The nodes expect to get the block not later than 1 minute after the start of a round. If the timeout of 1 minute is up, nodes fall back to the empty block. As a result, the protocol is slowed down because of waiting for a block that never arrives and furthermore no new transactions are added to the blockchain in that round.

The effect of this attack on block rate is shown in Figure 7.3. We can see that the block rate is dropping significantly quicker with an active adversary compared to the scenario where the nodes are faulty but not attacking the protocol. A countermeasure for this attack could be a punishment for the malicious node in a form of seizing (part of) her staked coins or remove the adversarial node from future elections. Lowering the timeout for receiving a block would not prevent this type of attack, but the impact would be reduced.

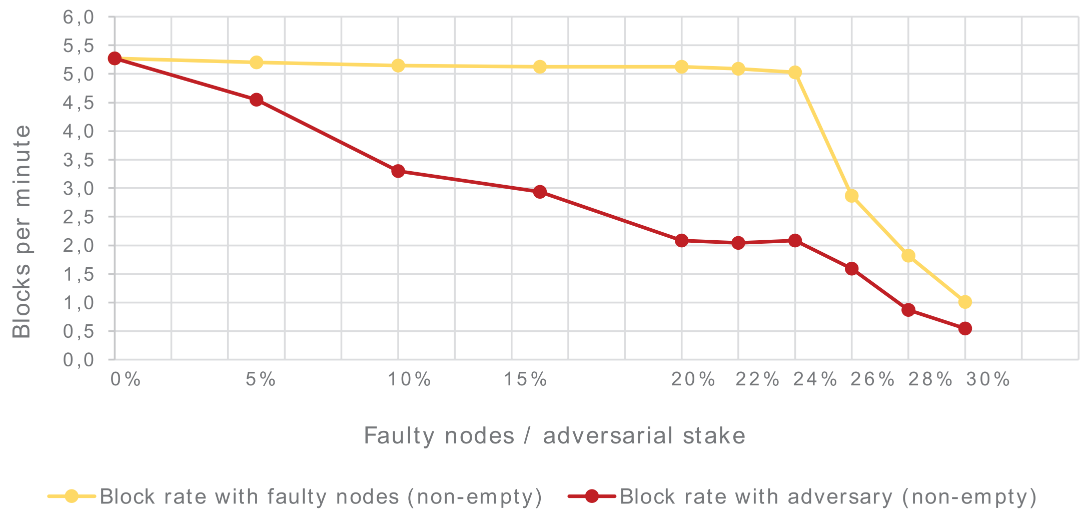


Figure 7.3: Comparison of influence of adversary and faulty nodes on block time.

7.2 ProPos

A round in ProPos consists of only two phases. Time of each phase can be set in the configuration file of the testbed. Times in the whitepaper [39] are 1,5s for voting phase and 4s for block propagation phase. Election of voters and a leader in a round is done by PRF (pseudorandom sampling). In the created testbed, PRF is implemented using C++ generator `minstd_rand0` and the seed for the generator is taken from the last block in the main chain. If the chosen leader fails to produce a block then the seed is updated and in next round, new committee and leader is chosen. Sending of the headers of the block is not discussed in ProPos whitepaper (unlike with Algorand) so in the simulation a leader sends only a full block. Size of the block is 1 MB and size of the vote message is set to 355 B.

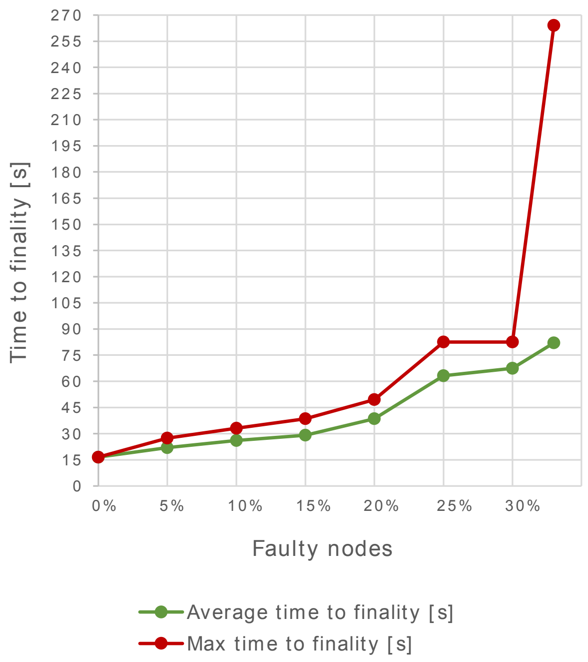
Every node stores her blockchain view individually with all the forks that are known to the node. Virtual blocks are different for every node because they contain votes that the node received but that are not included in any standard block yet (any standard block that the node knows about). Anonymization layer was not implemented for ProPos as it was not discussed in [39]. Note that for a real world use of ProPos, an anonymization layer is required to mitigate DOS attacks on committee members and leaders.

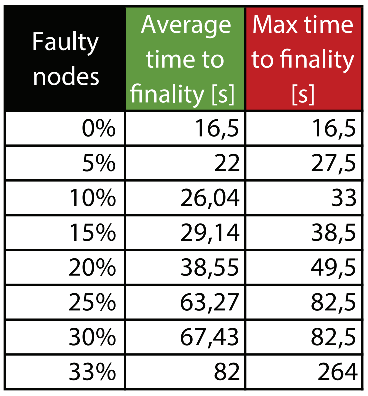
Time to finality with faulty nodes in ProPos

Time to finality is calculated as a time between block creation and first commit to this block by any node. From these times, the average and maximum is calculated.

As expected, with the number of faulty nodes increasing the time to finality is growing. This is because there are less and less votes contained in each block so that it takes more and more rounds before a block can be committed.

Results of the experiments are in Figure 7.4. We can see that average time to finality starts at 16,5s and then it is increasing quite rapidly. There is a spike at 33% mark of faulty nodes because it is often the case that there is not more than 66% of expected votes included in each block. When the number of faulty nodes exceeds 33% the time to finality becomes very high and protocol stops proceeding further.





Faulty nodes	Average time to finality [s]	Max time to finality [s]
0%	16,5	16,5
5%	22	27,5
10%	26,04	33
15%	29,14	38,5
20%	38,55	49,5
25%	63,27	82,5
30%	67,43	82,5
33%	82	264

Figure 7.4: Correlation between number of faulty ProPos nodes and time to finality.

7.3 Performance of ProPos and Algorand in a slow network

The goal of this experiment was to see what effect has the network latency on ProPos and Algorand. Connections between nodes are all the same but their latency can be set in the configuration in a form of round-trip time (RTT). This is a time it takes for a packet to go from one node to its peer and back. Then the time to finality of ProPos and Algorand was measured for different RTTs.

The results of a comparison of the protocols are in Figure 7.5. One obstacle in the testing appeared with ProPos. With RTT = 300 ms, the protocol was unable to continue. In ProPos, the time for the phases is defined by the protocol and with higher RTT, this time was not enough to propagate blocks through the network. To continue with the experiment, the time for the second phase of ProPos round was increased from 4s to 11s. Then with RTT further increasing, the phases times had to increase as well. The times used in this experiment were found out experimentally and are shown in Table 7.1.

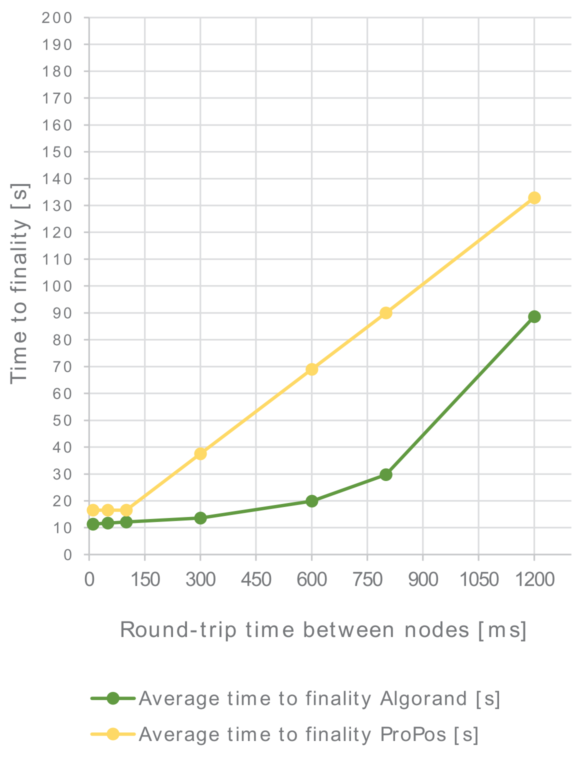
With Algorand, no such adjustments were needed because when the finality is not reached in a step of a round then the round continues with the next step until the final or at least tentative consensus is reached. This mechanic is also important for node's synchronization. For example, when one node is ahead of the others, it can't get enough votes in a step to reach a consensus, so the node continues with the next step until enough nodes catch up and the consensus can be reached.

As we can see from the results, the time to finality was growing faster for ProPos than for Algorand. This is because in ProPos finality was reached after 3 rounds. This means that when the duration of the round was increased by Δ the time to finality increased by $\approx 3\Delta$. In Algorand, a final consensus is reached in every round and the time to finality is increasing slower.

RTT	Δ_1	Δ_2	Average time to finality
0 – 100ms	1,5s	4s	16,5s
300ms	1,5s	11s	37,5s
600ms	2s	21s	69s
800ms	2,5s	27,5s	90s
1200ms	3,3s	41s	132,9s

Table 7.1: ProPos phases times for different network latency.

RTT 300ms could be a time measured in practice when the nodes are in different continents or if the connection is slow. Based on the simulation, when RTT is 300ms, the time to finality in ProPos is 37,5s but in Algorand it's only 13,6s. The experiment was done with 500 nodes, and number of peers was 8. This means the average number of hops for data to reach every node was $\log_8(500) \approx 3$. For 10 000 nodes we get $\log_8(10000) \approx 4,4$ hops so the time to finality would increase further.



RTT [ms]	Average time to finality Algorand [s]	Average time to finality ProPos [s]
10	11,37	16,5
50	11,68	16,5
100	12,1	16,5
300	13,58	37,5
600	19,88	69
800	29,8	90
1200	88,65	132,9

Figure 7.5: Comparison of Algorand and ProPos with different network latency.

7.4 Hedera

Testbed for Hedera is designed in a way that all nodes are constantly sending each other messages. A message carries all new information about a hashgraph that the node has. In practice, a transaction data is also part of the message. The size of the message was set to 200 B. Note that with Hedera in the created testbed, the gossip protocol was implemented in a way that nodes are sending their peers messages in given time intervals, but they are not immediately gossiping all received messages. The number of peers suggested in whitepaper [26] is 5 but in the experiments in this thesis, 8 peers were used for better comparison to ProPos and Algorand.

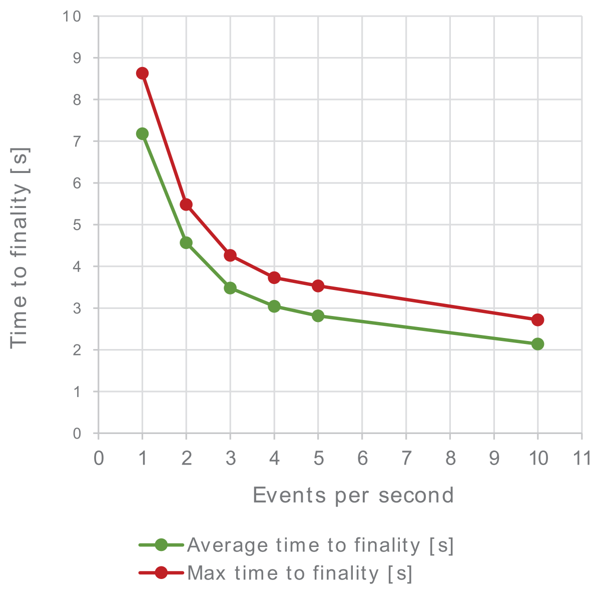
Every node is trying to confirm an event (transaction) as soon as her can. In the experiments, the time to finality is measured as a time between an event creation and its first confirmation by any node.

Performance of Hedera

Hedera aims to minimize communication overhead for reaching a consensus. Performance of Hedera protocol is more dependent on network conditions, node's processing power and total number of nodes in the system. To put it simply, the more the nodes communicate with each other, the faster the protocol confirms transactions because the faster the nodes update their view of the hashgraph.

In the Figure 7.6 the chart shows correlation of time to finality and number of messages each node sends hers peers per second. In the simulations, an ideal network with RTT 10 ms was used. As we can see with a rate of 10 messages per second, the time to finality can be as low as $\approx$ 2s.

In practice, there needs to be a trade-off between the message rate and the performance of the protocol (time to finality, throughput) because of the network limitations. One possible solution is to choose a group of master nodes where each has large number of peers and an excellent connectivity. This solution would bring higher performance but for the cost of greater centralization.



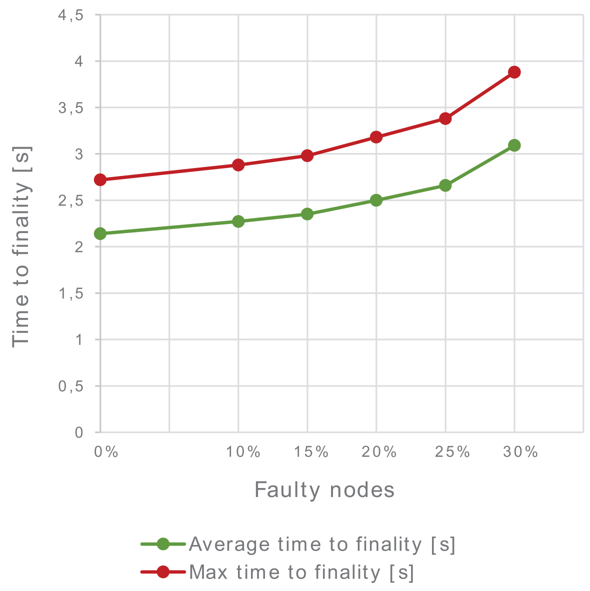
Event/s	Average time to finality [s]	Max time to finality [s]
1	7,18	8,63
2	4,57	5,48
3	3,48	4,26
4	3,04	3,73
5	2,81	3,53
10	2,14	2,72

Figure 7.6: Correlation between events per second created by each node and time to finality.

Time to finality with faulty nodes in Hedera

The experiments with Hedera were done with 50 nodes only, because with more nodes the number of messages in the system rise exponentially. The message sending rate was set to 10 and the network parameters were ideal.

As we can see in Figure 7.7, in these conditions the time to finality is growing relatively slowly – to 3s on average when there is 30% of non-functioning nodes. With 33% of faulty nodes and or more, the protocol is not able to continue further as no events are ever confirmed. This is because there is no randomness involved (as with VRF in Algorand or PRF in ProPos) hence the stake owned by faulty nodes is always missing and there is no possibility of events confirmation.



Faulty nodes [%]	Average time to finality [s]	Max time to finality [s]
0	2,14	2,72
10	2,27	2,88
15	2,35	2,98
20	2,5	3,18
25	2,66	3,38
30	3,09	3,88

Figure 7.7: Correlation between number of faulty Hedera nodes and time to finality.

7.5 Comparison of selected protocols

All three protocols tested in this thesis have their strengths and weaknesses. Only a few properties were tested. In practice, a protocol may be superior to others even though based on theoretical analysis and simulations it seems worse. At the time of writing, Algorand and Hedera networks are deployed and functional.

In Table 7.2 there is comparison of a few performance specific properties. Throughput and finality given is achieved in ideal network conditions and for block size of 1 MB.

The number of consensus messages is the number of messages that were sent (and gossiped) in the network per second and that are not transaction messages. For Algorand in every round the number of consensus messeges sent is $26N + kN \approx kN$ where N is the total number of nodes and k is a committee size. There are on average 26 block proposers in every round and each message needs to reach every node by gossip protocol. With round time of 11,38s this gives $\frac{kN}{11{,}38}$ consensus messages. Now when the network is synchronous, there are 4 steps in a round. In each step, a committee is elected. In Algorand whitepaper[1] and the simulations, the committee size is $\frac{1}{25}N$ for 3 standard steps and then $\frac{1}{5}N$ for the

	Throughput	Time to finality	Number of consensus messages
Algorand	87 kB/s ≈ 350 TPS	11,38s	$\sim 0.35kN/\text{s} \approx \frac{N^2}{36}/\text{s}$
ProPos	180 kB/s ≈ 730 TPS	16,5s	$\sim 0.18kN/\text{s} \approx \frac{N^2}{55}/\text{s}$
Hedera	—	2,14s	$\sim 10 \cdot 8N/\text{s}$

Table 7.2: Comparison of selected PoS protocols with data from experiments.

N - Number of nodes in the system
k - Number of voting committee members

final step. This gives us $k = 3\frac{N}{25} + \frac{N}{5} = \frac{8N}{25}$. Finally the number of consensus messages is $\frac{8N^2}{25*11{,}38} \approx \frac{N^2}{36}$.

For ProPos a similar calculation can be performed. There are only 2 steps in a round in ProPos and only 1 node is sending a block. With a round time of 5,5s we have $\frac{N+kN}{5{,}5} \approx \frac{kN}{5{,}5}$ messages per second. In ProPos whitepaper the suggested committee size is $\frac{N}{10}$. This gives us $\frac{N^2}{55}$ consensus messages per second.

For Hedera the number of consensus messages is simply $r \cdot p \cdot N/\text{s}$ where r is the message sending rate and p is the number of peers and the messages are not immediately gossiped further. When every node sends 10 messages per second to its 8 peers we have $10 \cdot 8N/\text{s}$.

It is hard to compare Hedera to the ProPos and Algorand based on the simulations in this thesis because the simulations for Hedera were performed with 50 nodes only. In this setting, the time to finality is much lower compared to the other two protocols and the number of consensus messages is also very low. This is because the messages are not immediately gossiped when received. But with more nodes, e.g., 10 000, the time to finality would not be so great and techniques that bring more overall communication would have to be used. Comparison of consensus message sending of selected protocols is in Figure 7.8.

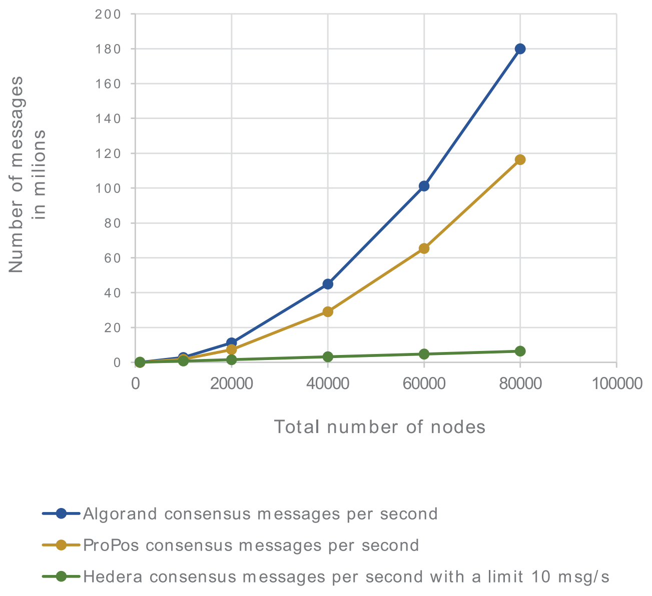


Figure 7.8: Consensus messages and network size.

Chapter 8

Conclusion

This thesis described blockchain technology, its properties and its current use cases. The main area of interest were techniques and protocols used for reaching a consensus in distributed systems. Problems of different solutions and attacks on PoS consensus protocols were discussed. A few PoS protocols were described and compared theoretically inTable 4.1, mostly based on results obtained in their whitepapers.

As part of this thesis, the simulation testbed for experimenting with Algorand, ProPos and Hedera was implemented. The testbed is created in OMNeT++ and thanks to its modular design can be extended by more protocols in the future. The testbed allows for configuring variety of parameters for each protocol and also the parameters of the network. The goal of the testbed is to perform simulations for testing the consensus layer of a blockchain system. For this reason, the operation of a given protocol is not simulated on a transaction level, but focuses on the operation of the consensus protocol itself and the handling of blocks and building the blockchain. Results of the simulations created using the testbed are in Chapter 7.

Results of the simulations suggest that Algorand tolerates faulty nodes very well if there is less than 26% of them. With block size of 1 MB, time to finality is about 12s and throughput is $\approx$ 330 TPS. In another simulation, an attack on Algorand was tested. An adversary was able to significantly slow down the protocol even with adversarial stake only around 10%.

With ProPos, the simulations also focused on the effect of faulty nodes on time to finality. Based on the results in Figure 7.4, faulty nodes in the network have larger effect on time to finality in ProPos, but the protocol can operate up to the 33% of faulty nodes. With all nodes honest and synchronous network, the throughput is about a double than the throughput of Algorand $\approx$ 730 TPS.

Then ProPos and Algorand were compared with changing network latency. With ProPos, the times to wait in each of the two steps were extended because otherwise the protocol could not proceed. In practice, with changing network conditions, the waiting times could be dynamic so the protocol would adapt automatically. The waiting times would extend when the network is asynchronous and then when the network is synchronous again the times would be reduced allowing the protocol to perform better.

With Hedera protocol, faulty nodes and time to finality was also tested. Results showed that the time to finality is decreasing relatively slowly, similarly to Algorand but Hedera can operate until there is 33% or more of the nodes faulty. Results of the simulations showed that Hedera performance is changing with the number of messages the nodes are sending each other – the more they send the faster the finality is achieved. This means that Hedera is more dependent on the network conditions than ProPos and Algorand.

Then the protocols were compared using all the results from the simulations. From the tested protocols, Hedera seems to be the most efficient with lower time to finality and higher TPS. The goal of the Hedera protocol is get near a theoretical limit where only sending of the transactions is enough to reach a consensus on the transactions and their order.

Next directions for work in this thesis is to perform more simulation scenarios with a goal of optimizing the protocols. For example, it could be useful to create a distributed simulation of Hedera and test its scalability. Another area the created testbed could be improved in, is an implementation of sharding.

Bibliography

[1] Gilad, Y., Hemo, R., Micali, S., Vlachos, G. and Zeldovich, N. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. In: *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*. ACM, 2017, p. 51–68. DOI: 10.1145/3132747.3132757. Available at: https://doi.org/10.1145/3132747.3132757.

[2] Oyinloye, D. P., Teh, J. S., Jamil, N. and Alawida, M. Blockchain Consensus: An Overview of Alternative Protocols. *Symmetry*. 2021, vol. 13, no. 8. DOI: 10.3390/sym13081363. ISSN 2073-8994. Available at: https://www.mdpi.com/2073-8994/13/8/1363.

[3] Amir, Y., Coan, B. A., Kirsch, J. and Lane, J. Prime: Byzantine Replication under Attack. *IEEE Trans. Dependable Secur. Comput.* 2011, vol. 8, no. 4, p. 564–577, [cit. 2020-11-20]. DOI: 10.1109/TDSC.2010.70. Available at: https://doi.org/10.1109/TDSC.2010.70.

[4] Nakamoto, S. and Bitcoin, A. A peer-to-peer electronic cash system. *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf*. 2008, vol. 4, [cit. 2020-11-14].

[5] Schuh, F. and Larimer, D. Bitshares 2.0: general overview. *Accessed June-2017.[Online]. Available: http://docs. bitshares. org/downloads/bitshares-general. pdf*. 2017.

[6] *BitShares Documentation - Delegated Proof of Stake (DPOS)* [online]. [cit. 2020-11-23]. Available at: https://how.bitshares.works/en/master/technology/dpos.html#background.

[7] Porru, S., Pinna, A., Marchesi, M. and Tonelli, R. Blockchain-Oriented Software Engineering: Challenges and New Directions. In: *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. 2017, p. 169–171. DOI: 10.1109/ICSE-C.2017.142.

[8] Zīle, K. and Strazdiņa, R. Blockchain use cases and their feasibility. *Applied Computer Systems*. 2018, vol. 23, no. 1, p. 12–20.

[9] Gervais, A. *Bitcoin Simulator* [online]. [cit. 2022-02-22]. Available at: http://arthurgervais.github.io/Bitcoin-Simulator/get_started.html.

[10] Cachin, C. and Vukolic, M. Blockchain Consensus Protocols in the Wild (Keynote Talk). In: Richa, A. W., ed. *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*. Schloss Dagstuhl -

Leibniz-Zentrum für Informatik, 2017, vol. 91, p. 1:1–1:16 [cit. 2020-11-19]. LIPIcs. DOI: 10.4230/LIPIcs.DISC.2017.1. Available at: https://doi.org/10.4230/LIPIcs.DISC.2017.1.

[11] Hoskinson, C. *Cardano Project* [online]. [cit. 2021-1-09]. Available at: https://iohk.io/projects/cardano/.

[12] Cardwell, N., Savage, S. and Anderson, T. Modeling TCP Latency. In:. 2000, p. 1742–1751.

[13] Buterin, V. and Griffith, V. Casper the Friendly Finality Gadget. *CoRR*. 2017, abs/1710.09437, [cit. 2020-11-22]. Available at: http://arxiv.org/abs/1710.09437.

[14] Castro, M. and Liskov, B. Practical Byzantine Fault Tolerance. In: Seltzer, M. I. and Leach, P. J., ed. *Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation (OSDI), New Orleans, Louisiana, USA, February 22-25, 1999.* USENIX Association, 1999, p. 173–186 [cit. 2020-11-18]. Available at: https://dl.acm.org/citation.cfm?id=296824.

[15] Zhang, R. and Preneel, B. Lay Down the Common Metrics: Evaluating Proof-of-Work Consensus Protocols' Security. In:. 2019, p. 175–192. DOI: 10.1109/SP.2019.00086.

[16] Frauenthaler, P., Sigwart, M., Borkowski, M., Hukkinen, T. and Schulte, S. *Towards Efficient Cross-Blockchain Token Transfers.* Technical Report. http://www. infosys. tuwien. ac. at/tast, 2019 [cit. 2021-01-10].

[17] Dai, W. and Project, C. *Crypto++® Library 8.6* [online]. September 2021. Available at: https://www.cryptopp.com/.

[18] Fanti, G., Venkatakrishnan, S. B., Bakshi, S., Denby, B., Bhargava, S. et al. Dandelion++: Lightweight Cryptocurrency Networking with Formal Anonymity Guarantees. *Proc. ACM Meas. Anal. Comput. Syst.* New York, NY, USA: Association for Computing Machinery. jun 2018, vol. 2, no. 2. DOI: 10.1145/3224424. Available at: https://doi.org/10.1145/3224424.

[19] Wang, K., Wang, Y. and Ji, Z. Defending Blockchain Forking Attack by Delaying MTC Confirmation. *IEEE Access*. june 2020, PP, p. 1–1. DOI: 10.1109/ACCESS.2020.3000571.

[20] Grigg, I. Eos-an introduction. *White paper. https://whitepaperdatabase. com/eos-whitepaper.* 2017, [cit. 2021-02-15].

[21] Fanti, G. C., Kogan, L., Oh, S., Ruan, K., Viswanath, P. et al. Compounding of Wealth in Proof-of-Stake Cryptocurrencies. In: Goldberg, I. and Moore, T., ed. *Financial Cryptography and Data Security - 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers.* Springer, 2019, vol. 11598, p. 42–61 [cit. 2020-11-22]. Lecture Notes in Computer Science. DOI: 10.1007/978-3-030-32101-7_3. Available at: https://doi.org/10.1007/978-3-030-32101-7_3.

[22] Liu, J., Li, W., Karame, G. O. and Asokan, N. Scalable Byzantine Consensus via Hardware-Assisted Secret Sharing. *IEEE Trans. Computers*. 2019, vol. 68, no. 1, p. 139–151, [cit. 2020-11-21]. DOI: 10.1109/TC.2018.2860009. Available at: https://doi.org/10.1109/TC.2018.2860009.

[23] wrapperband, G. *Feathercoin Wallet - Meta data* [online]. July 2017 [cit. 2020-11-22]. Available at: https://github.com/wrapperband/FeathercoinWalletGuide#advanced-checkpointing-acp.

[24] Gervais, A., Karame, G., Wüst, K., Glykantzis, V., Ritzdorf, H. et al. On the Security and Performance of Proof of Work Blockchains. In: ACM. *Proceedings of the 23nd ACM SIGSAC Conference on Computer and Communication Security (CCS)*. 2016.

[25] Sompolinsky, Y. and Zohar, A. Secure High-Rate Transaction Processing in Bitcoin. In: Böhme, R. and Okamoto, T., ed. *Financial Cryptography and Data Security - 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*. Springer, 2015, vol. 8975, p. 507–527 [cit. 2020-11-28]. Lecture Notes in Computer Science. DOI: 10.1007/978-3-662-47854-7_32. Available at: https://doi.org/10.1007/978-3-662-47854-7_32.

[26] Baird, D. L., Harmon, M. and Madsen, P. *Hedera: A Public Hashgraph Network & Governing Council*. 2020 [cit. 2020-1-3].

[27] Homoliak, I., Venugopalan, S., Hum, Q., Reijsbergen, D., Schumi, R. et al. The Security Reference Architecture for Blockchains: Towards a Standardized Model for Studying Vulnerabilities, Threats, and Defenses. *CoRR*. 2019, abs/1910.09775, [cit. 2020-11-18]. Available at: http://arxiv.org/abs/1910.09775.

[28] Hyperledger Team. *Hyperledger architecture, volume 1: Consensus* [online]. 2017 [cit. 2020-11-19]. Available at: https://www.hyperledger.org/wp-content/uploads/2017/08/HyperLedger_Arch_WG_Paper_1_Consensus.pdf.

[29] Kordek, M. and Beddows, O. *Lisk SDK* [online]. 2016 [cit. 2021-02-15]. Available at: https://github.com/LiskHQ/lisk-sdk.

[30] Günther, S. *Lisk — the mafia blockchain* [online]. [cit. 2020-11-23]. Available at: https://medium.com/coinmonks/lisk-the-mafia-blockchain-47248915ae2f.

[31] Eyal, I. and Sirer, E. G. Majority Is Not Enough: Bitcoin Mining Is Vulnerable. In: Christin, N. and Safavi-Naini, R., ed. *Financial Cryptography and Data Security - 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers*. Springer, 2014, vol. 8437, p. 436–454. Lecture Notes in Computer Science. DOI: 10.1007/978-3-662-45472-5_28. Available at: https://doi.org/10.1007/978-3-662-45472-5_28.

[32] Foundation, N. *NEM* [online]. 2018. Available at: https://nemproject.github.io/nem-docs/pages/Whitepapers/NEM_techRef.pdf.

[33] Leroy, X., Doligez, D., Frisch, A., Garrigue, J., Rémy, D. et al. *The OCaml system release 4.13* [online]. September 2021. Available at: https://v2.ocaml.org/releases/4.13/manual/index.html.

[34] *Simulation Manual, OMNeT++ version 5.6.1* [online]. [cit. 2022-02-06]. Available at: https://doc.omnetpp.org/omnetpp/manual/.

[35] Kiayias, A., Russell, A., David, B. and Oliynykov, R. *Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol* [Cryptology ePrint Archive, Report 2016/889]. 2016. Https://eprint.iacr.org/2016/889.

[36] De Angelis, S., Aniello, L., Baldoni, R., Lombardi, F., Margheri, A. et al. PBFT vs proof-of-authority: Applying the CAP theorem to permissioned blockchain. 2018.

[37] gavofyork, G. *PoA Private Chains* [online]. 2015. Available at: https://github.com/ethereum/guide/blob/master/poa.md.

[38] King, S. and Nadal, S. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *Self-published paper, August.* 2012, vol. 19, p. 1, [cit. 14.2.2021].

[39] Reijsbergen, D., Szalachowski, P., Ke, J., Li, Z. and Zhou, J. ProPoS: A Probabilistic Proof-of-Stake Protocol. *CoRR.* 2020, abs/2006.01427, [cit. 2020-11-28]. Available at: https://arxiv.org/abs/2006.01427.

[40] Jalalzai, M. M., Busch, C. and III, G. G. R. Proteus: A Scalable BFT Consesus Protocol for Blockchains. *CoRR.* 2019, abs/1903.04134, [cit. 2020-11-21]. Available at: http://arxiv.org/abs/1903.04134.

[41] Schoenmakers, B. A Simple Publicly Verifiable Secret Sharing Scheme and Its Application to Electronic. In: Wiener, M. J., ed. *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings.* Springer, 1999, vol. 1666, p. 148–164. Lecture Notes in Computer Science. DOI: 10.1007/3-540-48405-1_10. Available at: https://doi.org/10.1007/3-540-48405-1_10.

[42] Lee, S. and Kim, S. Rethinking selfish mining under pooled mining. *ICT Express.* 2022. DOI: https://doi.org/10.1016/j.icte.2022.03.003. ISSN 2405-9595. Available at: https://www.sciencedirect.com/science/article/pii/S2405959522000443.

[43] Schneider, F. B. Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial. *ACM Comput. Surv.* 1990, vol. 22, no. 4, p. 299–319, [cit. 2020-11-19]. DOI: 10.1145/98163.98167. Available at: https://doi.org/10.1145/98163.98167.

[44] Aoki, Y., Otsuki, K., Kaneko, T., Banno, R. and Shudo, K. SimBlock: A Blockchain Network Simulator. In: *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS).* 2019, p. 325–329 [cit. 2021-01-10]. DOI: 10.1109/INFCOMW.2019.8845253.

[45] Buterin, V. *Slasher: A Punitive Proof-of-Stake Algorithm* [online]. January 2014 [cit. 2020-11-22]. Available at: https://blog.ethereum.org/2014/01/15/slasher-a-punitive-proof-of-stake-algorithm/.

[46] Daian, P., Pass, R. and Shi, E. Snow White: Robustly Reconfigurable Consensus and Applications to Provably Secure Proof of Stake. In: *Financial Cryptography.* 2019.

[47] Gazi, P., Kiayias, A. and Russell, A. Stake-Bleeding Attacks on Proof-of-Stake Blockchains. In: *Crypto Valley Conference on Blockchain Technology, CVCBT 2018, Zug, Switzerland, June 20-22, 2018*. IEEE, 2018, p. 85–92 [cit. 2020-11-22]. DOI: 10.1109/CVCBT.2018.00015. Available at: https://doi.org/10.1109/CVCBT.2018.00015.

[48] Storj Labs, Inc. *Storj: A Decentralized Cloud Storage Network Framework* [online]. 2018. Available at: https://www.storj.io/storjv3.pdf.

[49] Deirmentzoglou, E., Papakyriakopoulos, G. and Patsakis, C. A Survey on Long-Range Attacks for Proof of Stake Protocols. *IEEE Access*. 2019, vol. 7, p. 28712–28725. DOI: 10.1109/ACCESS.2019.2901858.

[50] Goodman, L. *Tezos — a self-amending crypto-ledger* [online]. 2014. Available at: https://tezos.com/whitepaper.pdf.

[51] Arluck, J. *LIQUID PROOF-OF-STAKE* [online]. Available at: https://tezosguides.com/gouvernance/lpos/.

[52] Thai, Q. T., Yim, J. C. and Kim, S. M. A scalable semi-permissionless blockchain framework. In: *2019 International Conference on Information and Communication Technology Convergence (ICTC)*. 2019, p. 990–995 [cit. 2020-11-19]. DOI: 10.1109/ICTC46691.2019.8939962.

[53] Sun, J. *TRON* [online]. [cit. 2021-2-14]. Available at: https://tron.network/.

[54] Stoykov, L. *VIBES: Fast Blockchain Simulations for Large-scale Peer-to-Peer Networks* [online]. 2018 [cit. 2021-01-10]. Available at: https://github.com/i13-msrg/vibes.

[55] Stoykov, L. *Bitcoin-like Blockchain Simulation System* [online]. 2018 [cit. 2021-01-10]. Available at: https://github.com/i13-msrg/vibes/blob/master/docs/Attacks-simulation-thesis.pdf.

[56] Micali, S., Rabin, M. O. and Vadhan, S. P. Verifiable Random Functions. In: *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*. IEEE Computer Society, 1999, p. 120–130. DOI: 10.1109/SFFCS.1999.814584. Available at: https://doi.org/10.1109/SFFCS.1999.814584.

[57] Wood, G. et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*. 2014, vol. 151, p. 1–32.

Appendix A

Included CD contents

- Text of this thesis
- Source code of the thesis in LaTeX
- Source codes of the created testbed in C++
- Source codes of the scripts to analyze data in Python
- Raw simulation results data