BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

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

BEZPEČNOSTNÍ A VÝKONNOSTNÍ ANALÝZA PROOF-OF-STAKE PROTOKOLŮ

MASTER THESIS
DIPLOMOVÁ PRÁCE

AUTHOR
AUTOR PRÁCE
Bc. Jan Kotráš

VEDOUCÍ PRÁCE
SUPERVISOR
Mgr. Kamil Malinka, Ph.D.

BRNO 2020

Brno University of Technology
Faculty of Information Technology

Department of Intelligent Systems (DITS) Academic year 2019/2020

Master's Thesis Specification

Student: **Kotráš Jan, Bc.**
Programme: Information Technology Field of study: Information Systems
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, privacy, 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 and implement them as part of simulation testbed.
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, I., Venugopalan, S., Hum, Q., & Szalachowski, P. (2019). A Security Reference Architecture for Blockchains. arXiv preprint arXiv:1904.06898.
- Gilad, Y., Hemo, R., Micali, S., Vlachos, G. and Zeldovich, N., 2017, October. Algorand: Scaling byzantine agreements for cryptocurrencies. In Proceedings of the 26th Symposium on Operating Systems Principles (pp. 51-68). ACM.
- 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.

Requirements for the semestral defence:

- Items 1 and 2.

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

Supervisor: **Malinka Kamil, Mgr., Ph.D.**
Consultant: Homoliak Ivan, Ing., Ph.D., UITS FIT VUT
Head of Department: Hanáček Petr, doc. Dr. Ing.
Beginning of work: November 1, 2019
Submission deadline: July 31, 2020
Approval date: October 31, 2019

Abstract

This diploma thesis deals with blockchain technology with focusing on the consensus protocols, especially on proof-of-stake protocols type. This thesis describes blockchain technology followed by description of consensus in this technology. First part precisely describes a comparison of specific proof-of-stake protocols based on the theoretical knowledge. Second part of thesis focuses on the design and the implementation of a testbed. The testbed is used for comparison of proof-of-stake protocols practically. In the last section of thesis, we discuss our observed properties of proof-of-stake protocols and on this basement we indicate some next ways of evolution and improvement in the consensus and proof-of-stake protocols.

Abstrakt

Tato diplomová práce se zabývá technologii blockchain se zaměřením na konsenzus protokoly, zvláště protokoly typu proof-of-stake. V této práci naleznete popis těchto protokolů následovaný popisem konsenzu v technologii blockchain. Prvotní kapitoly detailněji popisují a porovnávají jednotlivé proof-of-stake protokoly na základě teoretických znalostí. Druhá část práce se zaobírá návrhem a implementací testbedu, který je následně použitý pro praktické porovnání proof-of-stake protokolů. V závěrečné částí práce je diskutováno nad zjištěnými výsledky pozorováním testbedu a zjištěnými vlatnostmi protokolů. Na tomto základě práce ve svém konci naznačuje další směřování consesus protokolů, ba jejich případné zlepšení, a zvláště proof-of-stake typu protokolů.

Keywords

Blockchain, Testbed, Proof-of-work, Proof-of-stake, Ouroboros, Algorand, Casper, Ouroboros Praos, Tezos, Simulator, NS-3, C++

Klíčová slova

Blockchain, Testbed, Proof-of-work, Proof-of-stake, Ouroboros, Algorand, Casper, Ouroboros Praos, Tezos, Simulator, NS-3, C++

Reference

Kotráš, Jan: *Security and Performance Testbed for Simulation of Proof-of-Stake Protocols*. Brno, 2020. Master Thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Mgr. Kamil Malinka, Ph.D.

Rozšířený abstrakt

Tato diplomová práce se zabývá technologií blockchain, se zaměřením na konsenzus protokoly. Blockchain je poměrně nová technologie, která je na velkém vzestupu. Vzestup nové technologie je doprovázený hledáním nových optimálnějších postupů, které technologie využívá. Jedním z těchto postupů je nový přístup ke konsenzu. Tento přístup je označovaný jako proof-of-stake (proof-of-stake protokoly). Primární zaměření práce je tento typ konsenzu, kde si práce klade za cíl zlepšení již existujích proof-of-stake protokolů. Práce tohoto dociluje díky vytvoření testbedu pro porovnání již existujících variant proof-of-stake protokolů, na jejímž základě práce navrhuje zlepšení současných protokolů.

Na začátku práce naleznete popis blockchain technologie jako takové. Práce zde uvádí koncept blockchainu, jeho výhody i nevýhody. Součástí popisu je nastínění i použití blockchain technologie v praxi. Na základě těchto podnětů je uvedeno, proč je dobré se zlepšování této technologie věnovat. V první části se práce věnuje jednotlivým konsenzus přístupům, jakožto proof-of-work, proof-of-stake, proof-of-capacity a další.

V třetí kapitole nalezneme definici vlastností a zranitelností, pomocí kterých můžeme konsenzus protokoly porovnávat. Následně jsou zde rozebrány již exitující testbedy, které práce využívá jako zdroj inspirace a to i přesto, že v současné době existují spíše testbedy a simulace pro proof-of-work protokoly.

Následně práce popisuje jednotlivé, již existující proof-of-stake protokoly – Algorand, Ouroboros, Ouroboros Praos, Casper, Tezos. Protokoly jsou popsány z technické stránky, tedy jak konsenzu dosahují a jaké jsou pro jeho dosazení potřeba předpoklady. U protokolů jsou zkoumány jejich výsledné vlasnosti, založené na definici v předchozí kapitole. Vlastnosti jsou zasazeny do tabulek, které přehledným způsobem sumarizují porovnání protokolů na teorietickém základě.

V druhé části se práce věnuje návrhu, implementaci a vyhodnocení našeho testbedu. Návrh nového testbedu nalezneme v kapitole 6. Při návrhu bylo zohledňováno možnosti širšího použití testbedu. Ikdyž je testbed primárně navržený pro tři protokoly – Algorand, Casper, Ouroboros, tak testbed sekundárně počítá s budoucími rozšířeními pro další protokoly. Návrh testbedu je následovaný jeho impelementací. Popis implementace nalezneme v kapitole 7. Pro implementaci bylo zvoleno simulační prostředí frameworku NS-3 v kombinaci s programovacím jazykem C++. Kapitola popisuje jak implementaci testbedu jako takového, tak i jednolivých proof-of-stake protokolů. Tyto protokoly jsou instanciovány do jádra vytvořeného simulátoru. V rámci implementace jednolitvých protokolů se také zaměřujeme na evaluaci jejich správné implementace. V této kapitole nechybí ani uvedení postupů, které implementaci zjednodušují, či zefektivňují simulaci různých stavů, např. stavů vedoucím k útokům na dosazení konsenzu.

Výsledně implentovaný testbed simuluje a porovnává protokoly v několika simulacích. Veškeré výsledky jsou uvedeny v kapitole 8. Výsledky těchto simulací jsou zanešeny do grafů či tabulek. Tabulky a grafy nám pak vizuálně popisují výkonost, propustnost, škálovatelnost implementovaných protokolů v různých situacích. Součástí simulací jsou i protokolově závislé simulace, které se zaměřují na zkoumání protokolově specifikických parametrů, např. Velikosti komise v případě protokolu Algorand.

Na závěr této práce, diskutujeme a navrhujeme možná zlepšení současných proof-of-stake protokolů. Návrhy jsou založeny jak na teoetickém, tak i na praktickém provnání protokolů. Návrhy se týkají zlepšení definovaných vlastností protokolů. Mezi výsledky patří např.: doporučená velikost komise pro protokol Algorand, doporučený typ algoritmu pro dosažení lepší škálovatelnosti a další doporučení, která nalezneme v kapitole 8 a 9.

Finálně, implementovaný testbed, položil základní kámen pro simulace proof-of-stake protokolů. Programátoři jednotlivých proof-of-stake protokolů si díky němu mohou ověřovat své teze praktickým způsobem. Simulátor je navržený pro další rozšiřování a to jak dalších proof-of-stake protokolů, tak i nových typů simulací. Díky těmto dvoum vlastnostem mohou být stále rozšiřovány návrhy pro zlepšení či navrhovány protokoly lepší a zcela nové.

Blockchains & Simulation: Security and Performance Testbed for Simulation of Proof-of-Stake Protocols

Declaration

Hereby I declare that this Master's thesis was prepared as an original author's work under the supervision of Mgr. Kamil Malinka, Ph.D. The supplementary information was provided by 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.

.....
Jan Kotráš
31.July 2020

Acknowledgements

I want to express huge thank to my supervisor Mgr. Kamil Malinka, Ph.D. and consultant Ing. Ivan Homoliak Ph.D. for technical factual comment and leading of this thesis. Also, I want to express thank to my family support and my girlfriend for english languag correction.

Content

1 Introduction

Nowadays, the word „blockchain“ is commonly known, mostly because of cryptocurrency Bitcoin. A lot of people connect Bitcoin and blockchain together, but the right definition is: Bitcoin is the cryptocurrency that is implemented on the base of blockchain technology.

Blockchain is a new technology with huge potential for lots of applications that connect more and more participants. The potential is based on the decentralization. Blockchain is now widely used in the cryptocurrencies, but there will come out new types of applications, like in the financial sector, electronic elections, medical (medical history), and so on.

In the blockchain does not exist any centralized authority. However, for the credibility of the blockchain network, there must be some regulations and the policy of blockchain traffic. For regulation of traffic in the chain, the blockchain uses consensus protocols. The most known type of algorithm, or protocol is a proof-of-work protocol, which is used in many applications, Bitcoin network. This protocol has some disadvantages though, that we will talk about in the first chapter of the thesis. The second most known protocol is a proof-of-stake. Proof-of-stake is a newer consensus approach, that tries to improve disadvantages of the proof-of-work approach. Proof-of-stake is currently used in some blockchain applications.

For implemented proof-of-work protocols already exists some theoretical and practical comparison, but, for proof-of-stake protocols there are only a few of them. So, the first goal of this thesis is going through more known implementations of proof-of-stake approach, explaining their functionality and comparing them on the theoretical base.

On the detailed explanation and comparison of proof-of-stake based protocols we design testbed, that will compare proof-of-stake protocols practically by the simulations. By comparison we can suggest the protocol‘s improvement, or new protocol.

At the beginning of thesis (chapter 2) we discuss the blockchain technology. The thesis writes there brief introduction of the actual status of blockchain technology and its usage. The chapter 2 contains some information about actual existing type of consensus and its description.

The chapter 3 writes about already existing testbeds and simulations. The proof-of-stake type of consensus actually does not have many testbeds. That is the reason why in this chapter writes mainly about proof-of-work testbeds and simulations. The chapter describes all mentioned testbeds, and it writes about their properties that can be used for our proof-of-stake testbed.

Chapters 2 and 3 describe theoretical existed problem. The chapter 4 writes about the motivations and goals of this thesis, which are based on the chapters above.

All main representatives of proof-of-stake protocols are described in the chapter 5. Chapter 5 describes protocols - Algorand, Ouroboros, Ouroboros Praos, Casper, Tezos. At the end of chapter, all protocols are compared and results of the comparison are written into tables.

The chapter 6 writes about designing of our testbeds for 3 selected protocols – Algorand, Ouroboros, Casper. In the chapter we can find a description of the designing process, description of used technologies, and the final design. The chapter is followed by chapter 7 that describes

proccess of implementation of the testbed. The chapter mentions some problems and solutions in the implementation part as well.

Chapter 8 shows the results of the simulator. There are results of testbed. All results are discussed and compared there. There are simulations of network throughput, scalability with discussion that is related to the privacy, security, finality.

Last chapter, chapter 9 summarizes the process of testbed creation, and the chapter creates a summary for the whole thesis with final recommendation for the proof-of-stake type of consensus.

2 Consensus protocols

The consensus is the key element of whole blockchain technology with huge impact on throughput and security. The chapter writes about consensus protocols in the blockchain technology. The chapter describes main approaches of consensus with the next focus to proof-of-work and proof-of-stake consensus type. The chapter describes process of consensus by specific approaches and it discusses their advantages and disadvantages.

Blockchain is a distributed continuously growing database. In the blockchain network does not exist a central authority. Blockchain network consists of participants (nodes). Every node can add some information to the blockchain database. Consensus protocols are described by algorithms. Consensus algorithms describe the validation of adding data into the blockchain database.

There are 3 types of blockchain networks we know about [1]. *Permissionless blockchain* like Bitcoin, where anybody can assign to network of blockchain, write/read an information. The second type is *permissioned blockchain,* where you can become a member only by given permissions. That permissions can be given by network, without central authority, although it can be. The third type is *private blockchain,* which is created for private domain or group of nodes.

This thesis focuses on the *permissionless blockchain.* For reaching the consensus in this type of blockchain network there are used consensus protocols. There is no possibility of the central direction of a network. On the other hand, in this type of consensus protocols, there are more problems connected to consensus.

- An individual node can drop down and consensus protocol has to resolve it.
- Consensus protocol works above some network, so there is a possibility of interrupting the connection.
- Ensure fair distribution of votes in the consensus.
- Consensus protocol has to be secure and resistant against attacks.
- Consensus protocols ensure, that the blockchain‘s data become immutable.

Consensus in blockchain can be established in many ways. Consensus is based on multiple entities that may not trust each other. Basic idea is to reach consensus by voting nodes of the network. Participants of blockchain communicate over the network and cooperate together to construct consensus without a central authority. That idea is based on the public election, so if 51% will agree, the consensus is valid. The problem is how to divide the power of votes into nodes. And this is the main difference between consensus protocols. Some of them are based on CPU power – proof-of-work. Another one, proof-of-stake, is based on the amount of „money“ that node owns.

Consensus protocols are a cornerstone of every blockchain technology. Almost all properties of the blockchain (throughput, security, scalability, and others – chapter 3.1) are based on the implementation of consensus. In the stacked model of blockchain we can find the consensus layer in the lower layers. It is illustrated by figure Figure 1[1].

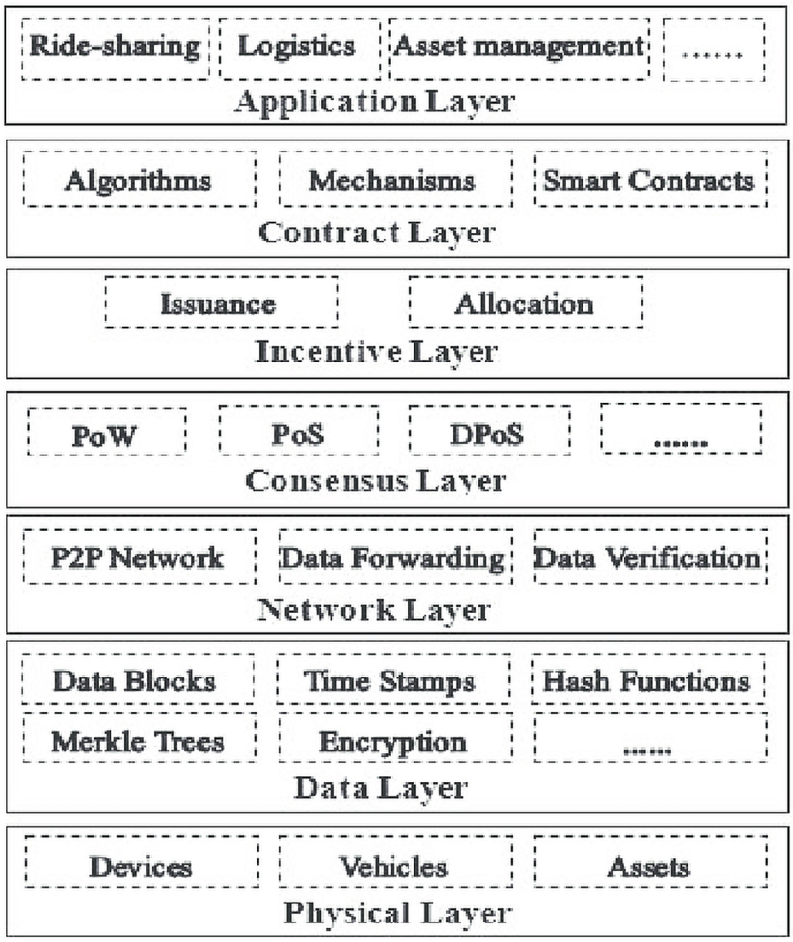


Figure 1: Stack model of the blockchain technology

2.1 Proof-of-work

One of the consensus approaches is proof-of-work. The proof-of-work is based on the physical resources "*Nodes vote with their CPU power, expressing their acceptance of valid blocks by working on extending them and rejecting invalid blocks by refusing to work on them. Any needed rules and incentives can be enforced with this consensus mechanism*" [2]. The idea of proof-of-work protocol was firstly published in 1993 by Cynthia Dwork and Moni Naor [3] and was later applied by Satoshi Nakamoto in the Bitcoin paper in 2008 [2].

1 https://www.researchgate.net/figure/An-ITS-Oriented-Blockchain-Model_fig4_332320425

The division of votes among all nodes is accomplished by resolving the *mathematical puzzle*. All blocks of blockchain ledger contaion some data like identification of previous block, sender, receiver and transaction data. The goal of the *mathematical puzzle* is to calculate SHA-256 hash (or any other hash algorithm) of the block. Proof-of-work adds some requirements on the transaction hash. One and main requirement: a hash produced by miners (miner is a node that wants to participate on the consensus) has to be lower than number or equal to it, which is set by blockchain network. To reach this format of the hash, proof-of-work protocols add one field into blockchain block called „Nonce“. The nonce is a 32 bit number. The goal of miners is to choose nonce‘s value into hash that is equal to or lower than target hash chosen by network. The nonce is the only value that can be changed by a miner in the blockchain block. There is not any sophisticated method of nonce‘s value calculation. The only way is to use brute force method.

Every block also contains „target“ hash information, which define the maximum value of the produced hash. Final hash is a 256-bit number. Final hash starts with few zero „0“ numbers. Count of zeros is determined by blockchain network. By the count of zeros, the network can control the difficulty of nonce calculation. The 32-bit size of the nonce means that there are four billion possible combinations. However, technically, it is much higher due to something called the extra nonce [4]. The structure is illustrated by Figure 2[2].

Figure 2: Blockchain structure

2 https://medium.com/@julianrmartinez43/understanding-proof-of-work-part-1-586d7ee6b014

A practical example of the calculation is below [5]. Every block in the example contains two main (hash calculation perspective) values. Nonce value and final block hash. Final hash, red part of the block, is calculated above block content, blue part. Requirements to final hash are it starts with 5 zeros. Miner is trying to calculate the target hash with 5 leading zeros, only by brute force method and changing nonce value.

Proof-of-work algorithm is mostly used in cryptocurrencies. But there are some disadvantages of this protocol:

- **51% attack** [6] - The history of blockchain is not changeable, because actual block points to the previous block by its hash, and that block points to his ancestor and so on. If you want to change some block, you have to recalculate all following confirmed blocks. In the context of proof-of-work protocol, it is possible, if you have majority of network performance. If anybody owns that performance, he is able to create fraudulent blocks of transactions for himself while invalidating the transactions of others in the network.
- **Electricity consumption** [7] - Another disadvantage is electricity consumption. During the calculation of *mathematical puzzle*, a CPU or any other device only tries to calculate right nonce‘s number. That is processed by the brutal force method. The brutal force method is not effective, so almost every electricity is consumed by useless results of nonce‘s value.

2.2 Proof-of-stake

„Proof of Stake (PoS) concept states that a person can mine or validate block transactions according to how many coins he or she holds. This means that the more Bitcoin or altcoin owned by a miner, the more mining power he or she has.“[8]

The first idea was published in the bitcointalk forum [9]. The motivation of that protocol was to save a huge amount of the computation power and in the perfect way also to create a more safe algorithm.

In the proof-of-stake protocol do not exist miners, but transaction validation is done by validators. The validator does not have to mine new blocks, validators only validate new blocks of the blockchain database. Validator‘s power is limited by percentage of token that her or his ownership of stake. For example, validator has 1 percentage of money (tokens), he is able to validate theoretically 1% of the transactions. 1% only theoretically because it depends on the specific proof-of-stake algorithm and used method. The selection of validator is illustrated by Figure 3[3].

3 https://lisk.io/content/5-academy/2-blockchain-basics/4-how-does-blockchain-work/7-proof-of-stake/8-pos-infographic.jpg

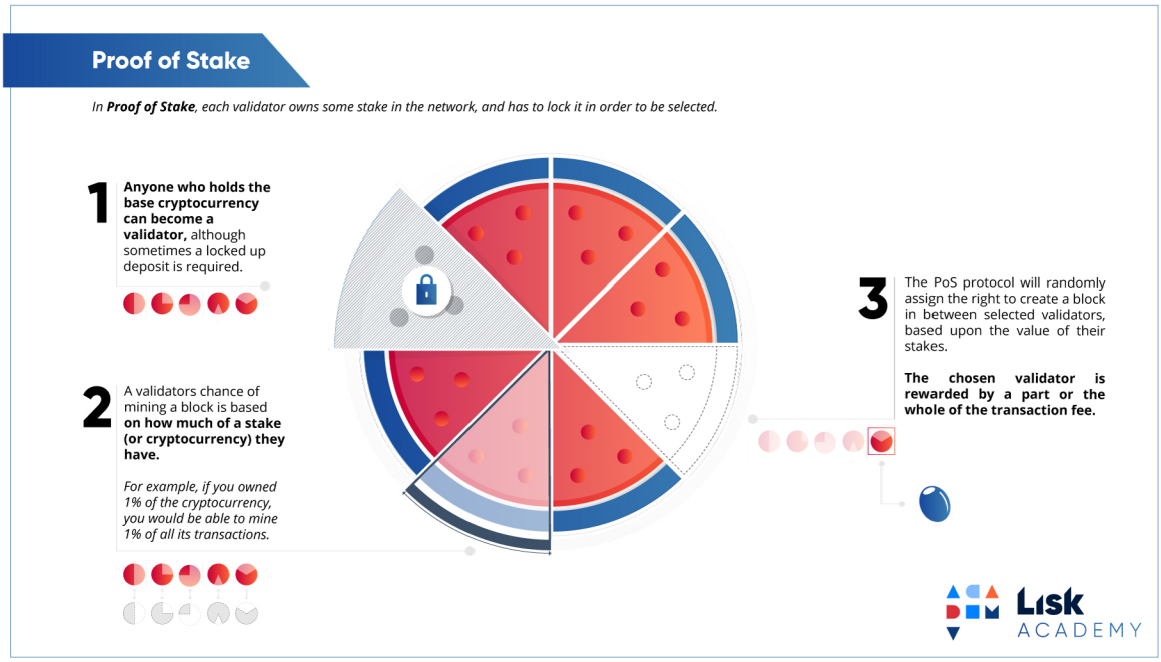


Figure 3: Proof of stake validator selection

The main idea is based on the percentage of validator ownership. However, the idea has one big disadvantage. If the network detects, that validator (forger) validated fraudulent transactions, there is no punishment. In this way, the next improvement is: The validator sends his money to the stack (stake) and starts doing the validation. When the network detects suspicious behavior, the forger node will lose a part of its stake.

In the proof-of-stake do not exist rewards for making blocks. The validator takes a transaction fee for every validated transaction as a reward for validating the transaction.

The proof-of-stake algorithms use a pseudo-random election process to select a node as the validator of the next block, based on a combination of factors that could include: the staking age, randomization and the node's wealth. The most used algorithms for choosing validator are *Randomized Blockchain Selection* and *Coin Age Selection*. In the *Randomized Blockchain Selection,* the validators are selected by the combination of amount of stake and by the lowest hash value. The second one, *Coin Age Selection* algorithm chooses, how long is their token waiting and the number of coins that are staked. Practically, it can be calculated like multiplying the number of days by the number of coins in the stake.

Proof of stake algorithm is now used in some cryptocurencies like: Ok cash [10], NAV coin [11], NEO [12], ARK [13], LISK [14], Cardano [15].

But the most discussed theme in nowadays cryptocurrencies and proof-of-stake protocols is the Ethereum cryptocurrency [16]. This cryptocurrency plans to change used the consensus algorithm. Ethereum actually uses the proof-of-work protocol, on the other hand, the implementators plan to migrate to the proof-of-stake protocol. For that purpose, the Ethereum should be divided into two separate cryptocurrencies. The first one is called Ethereum version *1.x*

with „old“ proof-of-work algorithm and the second one Ethereum 2.0, which will be hard fork of Ethereum *1.x* with proof-of-stake consensus algorithm [16].

In the list blockchain protocols (cryptocurrencies), each cryptocurrency has its own set of rules and properties for what the implementors of cryptocurrency think that is the best solution and combination of rules for their application.

This is a reason why we can find more wide types of solutions than in the context of proof-of-work protocols. This thesis is focused on the differences between specific proof-of-stake protocols implementation in next chapters.

2.3 Proof-of-burn

Proof-of-burn is an alternative approach for proof-of-stake or proof-of-work consensus type. Proof-of-burn, as the new alternative, is being tested. It is also popularly called proof-of-work without energy waste.

That consensus protocol is based on the burning (destroying) coins, that can be termed like virtual currency tokens. The basic idea was published by Iain Stewart, the designer of proof-of-burn algorithm - „*Burnt coins are mining rigs. Essentially, a miner burns his/her coins to buy a virtual mining rig that gives him/her the power to mine blocks. The more coins burned by the miner, the bigger the ensuing virtual mining rig*“ [17].

Burning of coins is processed by sending coin to un-specdable address [18] also called „eater address“ . Those addresses are randomly generated without the private key. Thankfully, that address has not any private key associated with it, there is no way how to get to „burned“ coins. Sending the tokens to burn does not consume many resources and it ensures that the network is still alive and active.

Participants can burn token from other blockchain technologies. So, it is possible to burn coins based on the proof-of-work protocol in the proof-of-burn algorithm. Thankfully, proof-of-burn inherits cons and pros of the algorithm that is used for coins burn.

The idea of the proof-of-burn technique is that users are willing to undergo a short-term loss for long-term investment by burning a cryptocurrency. Burned coins grant the right to write blocks in the proportion to the burned virtual tokens. Basically, if you burn more coins, then you have more right to mine blockchain block. Participants of burning can be rewarded for their activities.

Practically, proof-of-burn is used in the algorithm named Slimcoin [19]. Slimcoin actually combines three algorithms – proof-of-work, proof-of-stake and proof-of-burn. Proof-of-work prepares coins for burning. Proof-of-stake manages mining rights, and the whole system follows proof-of-burn ideas. In Slimcoin, the burn of coins does not give rights to the next block, but you can get a chance to receive blocks in the future time period – about a year.

2.4 Proof-of-capacity

Proof-of-capacity is also called proof-of-space. It is based on the same idea as proof-of-work protocol. Protocol calculates the nonce‘s value to satisfy requirements to the final hash.

But the proof-of-capacity does not use CPU or any other hardware to keep calculating, which means trying to find the right value of nonce. Proof-of-capacity tries to use the node hard-drive volume. Proof-of-capacity uses node hard-drive to save the list of possible solutions of the hash activities. That is calculated before mining activity comes. The bigger size of hard-drive = the node can store more possible solutions and it has a higher chance to win mining reward, resolve a mathematical puzzle.

Mining in the proof-of-capacity protocol is divided into two parts. First called plotting and second one mining. During the first part, the protocol calculates files, exactly plot files. Each file contains a large number of precalculated hashes. In the second part, the protocol starts mining. Mining is being processed on the incoming data with the support of the saved plot files. The protocol cooperates with hard-drive and tries to find right solution based on the precalculated files.

Proof-of-capacity protocols have less electricity consumption than the classic proof-of-work algorithm. Some literature says that the proof-of-capacity consumes only 1% of electricity, compared to proof-of-work algorithm. Another benefit is the price of hardware. Hard-drives are much cheaper than expensive ASIC hardware.

Burstcoin [20] is a cryptocurrency, that uses proof-of-capacity.

2.5 Proof-of-authority

Proof-of-authority protocols are validated and approved transactions for a future block made by validators. Validators allow putting the transaction in the block.

Proof-of-authority protocols are based on the reputation. That algorithm was inspired by the proof-of-stake algorithm, where nodes stake their coins. In the proof-of-authority nodes stake their reputation instead. Becoming a validator can be a long process, because future validator has to invest their reputation in the stake. In that context, the new node does not have enough reputation or it has an only little reputation, so it is pretty challenging to become a validator. It‘s a long-term process, because validator creates a better reputation over a long period, based on its behavior.

The protocol is more suitable for private blockchain protocols, which are used in private networks. That can be used in certain businesses and corporations, especially for high performance and precondition of high trustability of network nodes.

3 Consensus protocols testbeds

Consensus protocols have properties and vulnerabilities that are practically compared and discussed. The first part of this chapter is about consensus protocol‘s properties and vulnerabilities that will be studied in the next chapters.

Consensus protocols are researched by simulations or testbeds. The research can be focused on throughput, security, or any other properties. In the thesis we want to go through some proof-of-stake protocols an observe them. For that, we will use testbed. By the testbed, we will verify our speculations, which can improve the proof-of-stake type of consensus. But at first, we describe some exist testbeds and inspire there.

3.1 Studied properties

Each consensus protocol has its own implementation with another motivational idea. Some protocols are focused on the performance, some on the anonymity, and others to security. We have different approaches on the comparison of those protocols and their properties. Each subsection of this chapter describes properties and vulnerabilities individually.

Properties are focused on the bases of blockchain consensus protocols, especially proof-of-stake protocols. Declared properties will be used in the next chapters for specific protocols comparison.

3.1.1 Implementation and design properties

In the context of proof-of-stake consensus protocols, we can divide consensus algorithms into 3 groups by used type of blocks validation [21].

Lottery based protocols – These protocols are based on running lottery when output is leader or committer which will validate next new block for the blockchain database. During the processing of the lottery, the majority of calculation is done into nodes internally. This leads to the biggest advantage – small traffic over the network. Protocols often use ‚The Follow the Satoshi algorithm‘ [2]. Nevertheless, the biggest disadvantage of the lottery-based protocols is a chance of electing more nodes as leaders and then the creation of multiple blocks with same height = forks. The fork is a division of blockchain database history, and in the future, only one way of division will be truthful. Forks might potentially lead to some attacks – described in section 3.1.3.

Voting based protocols – New attached block is not validated by a leader, but all participants are voting, the new block is produced only collaboratively. But that voting needs a lot of network communication. Protocols that use the voting approach, almost always include Byzantine fault tolerant (BFT) protocols. On the opposite, the voting based protocol is a very low probability of forks creation, for example, Algorand [22].

Combination – In fact, practically strictly lottery-based or voting-based protocols do not exist. The practically used protocol is trying to combine the pros of all approaches to get a better consensus protocol. For example in Algorand, which uses BTF, all nodes are not voting (more in the next chapter).

3.1.2 Behavioural properties

The behaviour of every consensus protocol depends on the selected technology, implementation, features and other parameters. But for the consensus protocols‘ comparison are used properties that are described below. Protocols can be evaluated by that. There is a list of metrics [21] used in practice:

- **Scalability** – determines the property of protocols, how protocol reacts to increasing/decreasing the number of participants. Some protocols can be designed for a dedicated scale number of participants. Scalability also describes the change of other properties like throughput or security in case of number of participants change. There is no value or measurement. Scalability is described in words.
- **Throughput** – describes performance of the protocol. Throughput represents the number of transactions per second. Measurement of throughput may have some assumptions. Throughput varies over protocols. Throughput highly depends on the type of consensus protocol (Byzantine Fault-tolerand, The Follow the Satoshi,…)
- **Security** – it describes susceptibility/defensibility to some vulnerabilities. The form of security is a verbal description based on the theoretical models.
- **Privacy** – privacy expresses the protocol‘s property, how a protocol works with privacy users‘ data, and how much of private data are published. Also, privacy describes traceability of coin owners. Anonymity is a part of the privacy description.
- **Failure-tolerance** – property of protocol describes how protocols deal with some network issues. For example participant network outage, connection problems, etc.. That property is a word described property, but it can be evaluated numerically, for example, percentage of participant that can fail.
- **Liveness** – ensures that honestly generated transaction has been available for other network nodes. Liveness is stated as an amount of time until generated transaction is available for other network nodes.
- **Safety** – ensures, when a honest node accepts a transaction than other honest nodes accept this transaction.
- **Finality** – finality depended on the safety. Finality expresses time or depth of the accepted block in blockchain history when the block can not be removed or overturned.
- **Energy efficiency** – the ratio between electricity consumption and completed work. Theoretically, it might be calculated like watt per transaction. But really, protocols are mostly strictly divided into two groups: poorly efficient (proof-of-work) and efficient (proof-of-stake).
- **Immutability** – property of protocol and generated blockchain, that can not be easily modified.
- **Decentralization** – describes, if the protocol depends on some type of centralization (authority). Also describes, if all participants have the same influence to consensus.

3.1.3 Protocols‘ vulnerabilities

This subchapter describes basic blockchain vulnerabilities and focuses on the consensus protocol‘s vulnerabilities, especially proof-of-stake vulnerabilities. Subchapter does not focus on the complete list, but it focuses mainly on discussed problems. The goal is to describe types of vulnerabilities that will be used and compared in the next chapters and in the testbed.

- **51% vulnerability** [2] – is the most known type of blockchain attack. The attack consists of one attacker, who owns more than 50% of voting (mining – depends on consensus context) power. That power might be owned by more nodes, but practically, nodes belong to one dishonest behavior.
  It is hard to avoid 51% attack, because consensus protocols assume some percentage of honest nodes. Exactly more than 50% honest users, or 2/3 in BTF (Byzantine fault tolerant) protocols. But in contrast, when the attack is detected, the value of cryptocurrency gets rapidly low.
- **Double-Spending Attack** [23]– the second most known type of blockchain attack through all types of blockchain, consensus protocols. The attack refers to a consumer, who uses the same cryptocurrency coin multiple times for outcome transaction. The attack is usually a consequence of the 51% attack. This type of attack has a higher impact on protocols with lower finality.
- **Breaking Network Assumptions** – some protocols assume synchronized communication or any other assumptions. An attacker tries to break those assumptions. It is hard to defend, because protocols assume these properties of a network.
- **Time De-Synchronization Attacks** – some consensus protocols are based on physical time. For example, Ouroboros needs the division of physical time into epochs and slots (more in chapter 4). Epochs and slots are identified by the division. An attacker tries to break system time to break participant‘s nodes synchronization.
- **Selfish Mining** – vulnerability theoretically possible in proof-of-stake protocols (need predictable randomness for election) [24], but practically realized in proof-of-resouce protocols. The attacker builds part of the chain on his own and tries to publish it into the public chain. Because the attacker knows part of the chain he made, he can mine without any additional effort.
- **Nothing-at-Stake** [25] – it is proof-of-stake‘s specific vulnerability. The vulnerability can occur, if there is a fork in chain of blocks. When the fork occurs, the miners (validators) can perform work on both chains. And that could theoretically make double-spending problem. The attacker creates a fork and performs double-spending attack. The rest of miners are mining in their best self-interest – mine both chains. So, the chain, that becomes a part of public blockchain, is highly depended on the attacker‘s mining, because other miners are mining on both chains of the fork, as well. This attack is proof-of-stake specific attack, because node (validator) can validate more branches together without risking its stake. It is trying to increase its chance to be rewarded.

- **Grinding Attack** [26]– a type of attack, where the attacker tries to increase his chance of being selected in the future. The attacker tries to find a combination of different parameters (dependent of consensus protocol). For example, if the election process of proof-of-stake protocol is dependent on a hash of the previous block, then the attacker can use that dependence for a future election‘s participants. New randomness for each election process, like VRF process, is the way how to avoid this type of attack.
- **Fake stake** – proof-of-stake specific attack. The voting power of one participant‘s conclusion is depended on its stake. The attack/problem is based on validation of the participant‘s stake, which should be increasing along the chains, which are getting longer. During validation, the attacker might claim, that they own more stake than they actually do [27].

3.2 Actual existing testbeds

In the introduction we explain what a testbed is. The testbed is a process and a way how to test and verify some scientific theories or new technologies. The testbed can be implemented as some hardware testing or software testing. In this thesis, we will use software testing. Software testbed is a method for testing new ideas and approving some preconditions. Software testbed is implemented like isolated software, that simulates the behavior of new software future running environment. The testbed is based on the proof of concept.

Between existed testbed exits some testbed or studies that are trying to compare or simulated consensus protocols properties. Almost all testbeds are focused on proof-of-work protocols. Only a few studies focused on the proof-of-stake protocols, because proof-of-stake approach is only a few years old.

In this chapter, we go through testbed that focused on the proof-of-work and proof-of-stake protocols. We describe them minutely and we highlight properties that can be used for our designed testbed.

3.2.1 Bitcoin simulator

Bitcoin simulator [28] is an instance of the framework [29], which uses a bitcoin network instance.

The framework consists of two key parts. First, a blockchain instance, and second, a blockchain security model. Blockchain instance can be theoretically classical proof-of-work blockchains like bitcoin, litecoin, ethereum (or any other PoW-based). Output of concrete instance blockchain simulation is a block rate that is used as input for security model based on Markov Decision Processes (MDP) [30] for double-spending and selfish mining.

The first part – blockchain instance captures the dynamic setting of clasical proof-of-work protocols properties – number of nodes, number of blocks in simulation, block size, and more [28]. To reach more realistic simulation, the framework implements network layer simulation by advertisement - based information propagation, unsolicited block pushes, the relay network, the sendheader propagation mechanism among others.

The second part – security model. Extended Markov Decision Processes (MDP) is used as a model. By this process is determined optimal adversarial strategies. In the opposite, it determines combination of different blockchain setting, that leads to high vulnerability, especially double-spending attack and selfish mining.

Framework for network simulation uses NS-3 network simulator [29]. NS-3 is a discrete event simulator for a system that is based on the internet network. NS-3 supports all ordinary network protocols. The simulator focuses on easy simulations with sufficiently realistic behavior for classical purposes.

3.2.2 StrongChain Demo

Simulator, [31] that illustrates communication between nodes in realtime. The simulator is based on basic internal blockchain implementation by account/balance model [32].

The purpose of the simulator is just to demonstrate some features and vulnerabilities by proof-of concept, like selfish mining. On the other hand, the simulator demonstrates, how real simulator can be implemented without any sophisticated tools nor any other frameworks. The simulator is based on the basic Python libraries.

3.2.3 SimBlock

It is a simulator application developed in July 2019, that supports simulating of the behaviour of about 10 000 nodes with using a single PC. Application is Java-based and the code is publicly available on the internet [33]. Simulator currently supports Bitcoin, Litecoin and Dogecoin network.

The application is focused on performance and throughput testing. The result of simulation [34] is compared with the simulator [35]. The SimBlock divides accounts into regions (like America, Asia, …), where the delay can be configured. The application is based on pure Java programming language and ordinary libraries. There are no frameworks of third parties.

SimBlock is an event-driven application. Each participant of the simulator (blockchain node) communicate with other nodes by ordinary P2P communication. The node generates messages and mining events. The simulator reacts to those events. The simulator uses an internal instance of blockchain algorithms. The output is in the JSON format and it is challenging for humans to read it. However, the implementators of Simblock also developed visualizer – simblock-visualizer. That visualizes the layout of nodes around the world and communication with each other. The layout of nodes is illustrated by Figure 4 [4].

4 https://dsg-titech.github.io/simblock/

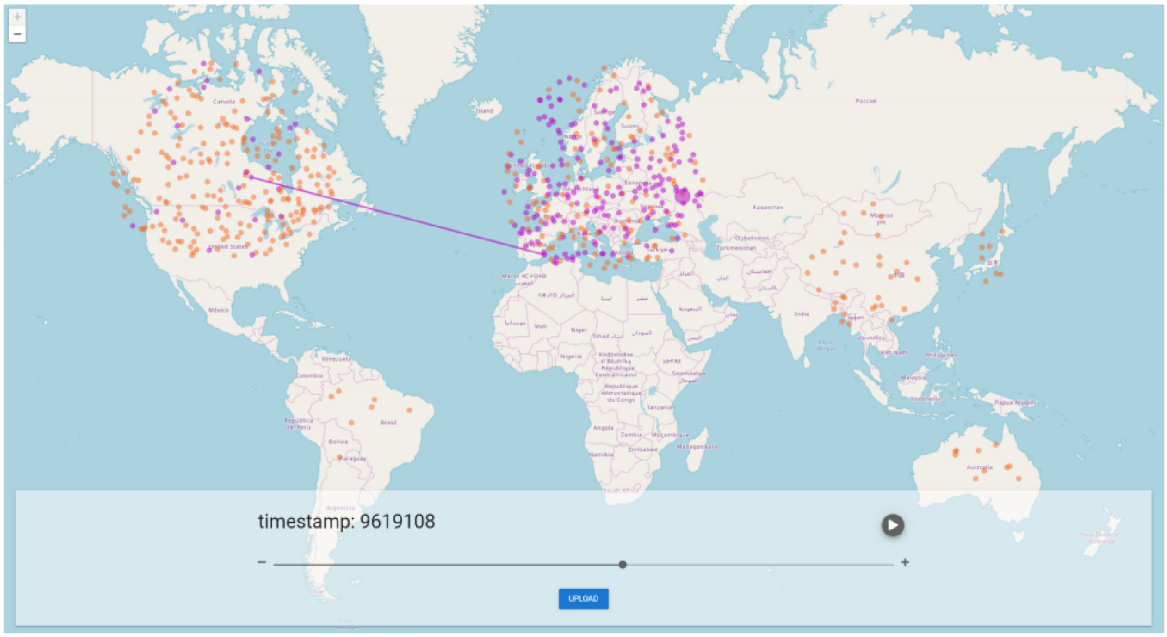


Figure 4: Simblock visualization via SimBlock-visualizer

Simblock was implemented like proof-of-work simulator for blockchains like Bitcoin. But in the last months (October 2019) we could see some work on proof-of-stake simulation in GIT history.

3.2.4 Framework - Evaluating PoW Consensus Protocols's Security

Framework for evaluation of proof-of-work protocols security [36]. Framework code calculates optimal strategies of different proof-of-work protocols. Calculations are based on the models that are defined as mathematical functions. Each part of the model (mathematical function or equation) defines evaluation of some blockchain property or vulnerability, these functions define evaluation metrics. The security model is based on the Markov decision processes.

The framework is trying to focus on proof-of-work protocols vulnerabilities, identifying them and evaluating them. It is based on evaluation analyzed behaviour of protocols in defined condition with the defined setting. The output of analysis is measurement of the protocol‘s vulnerabilities in the context of the defined pre-conditions. The analysis also creates a basement for suggested improvements.

The whole framework is based on calculations. The framework is not a simulation of protocol instances. The framework was implemented in the Matlab with support of the MDP toolbox for Matlab. The source code of the framework is available for public [37].

4 Problem definition (motivation)

As we mentioned in the chapter before, we know lots of types of consensus protocol types. Every type of protocol has its own advantages and disadvantages. Also, every of the protocol can be better in specific usage than other protocols. In this thesis we will focus on the proof-of-stake type of consensus protocols.

Actually, the proof-of-work type of consensus protocols is used most often. Proof-of-work protocols are widely used, because proof-of-work protocols used to be protocols that guarantee general targets of the blockchain network, like throughput, safety, scalability, finality. But actually, the proof-of-work type of consensus comes to an edge of its properties. The biggest disadvantage of proof-of-work protocols are (properties are related mainly to Bitcoin implementation proof-of-work consensus):

- **Efficiency** – proof-of-work type of consensus has huge problems with efficiency. The protocol calculates a lot of calculations, but only an insignificant part of them are effective calculations. Other calculations are there only for the construction of the proof-of-work type of the consensus. That leads to high electricity consumption of the whole network. Efficiency is an increasing problem, that starts to be a global problem. Here is a comparison with other human activities, and its energy used - Figure 5[5].
- **Performance** – performance, or throughput of the network, is limited. Limitation of the throughtput is made because of security parameters – We can improve throughput but with worse security as a consequence.
- **Finality** – in the proof-of-work protocols it might create lots of forks. That property is highly limited for marking the block as finished.

Mentioned disadvantages motivate lots of blockchain network implementators to try other consensus approaches. As the best alternative to proof-of-work protocols seems to be proof-of-stake protocols. Proof-of-stake protocols declare better properties (mainly mentioned) than proof-of-work protocols. This is the main reason, why this thesis focuses on the proof-of-stake consensus protocols.

In the proof-of-stake protocols field exist a few approaches on how to implement them. The approaches are described in the next chapter. In the next chapter, there are also discussed advantages and disadvantages of particular approaches.

In nowadays exists only a theoretical comparison between proof-of-stake implementation approaches. Also, exists only a theoretical comparison between specific proof-of-stake protocols. There are no testbeds or simulations that compare proof-of-stake approaches. But thanks to the practical comparison of protocols, we can find new strategies of implementations, new potential problems or other properties of protocols, to make it better.

As the solution of that, we will create proof-of-stake testbed inspired by proof-of-work testbeds. In the testbed, we will compare a few proof-of-stake protocols. We will study specific protocols‘ behaviour in specific situations. We will focus on the declared properties

5 https://d19czvic2hcumt.cloudfront.net/content/2020/02/bitcoin-energy-consumption-vs-banks-gold-paper.jpg

of the protocols. As a result of testbed-based comparison, we will get data based on the metrics that will declare protocols suitably.

The result will be used for the improvement of proof-of-stake protocols. Result should give us the answer for the question, „which protocol is better and why“, or „when is this protocol better than others and why“. Based on the result we can create suggestions to get better actual protocols, or we can design a brand new proof-of-stake protocol.

As the next result of the thesis, a proof-of-stake community gains a testbed with simulator. The simulator will include 3 proof-of-stake protocols instances and the simulator will be opened to new protocols instances. Thanks to that, implementators of the new proof-of-stake protocol will be able to compare this new protocol with existing protocols. Also, implementators will be able to compare configurations of this new protocol.

Comparison of Environmental Costs

	Energy Used (GJ)	Tonnes CO_2 Produced	Emission Trend
Gold Mining	475 million	54 million	Increasing
Gold Recycling	25 million	4 million	Decreasing
Paper Currency & Minting	39.6 million	6.7 million	Increasing
Banking System	2340 million	390 million	Increasing
Bitcoin Mining	3.6 million	0.6 million	Decreasing

Figure 5: Blockchain (bitcoin) electricity consumption

5 Proof-of-stake protocols

In this part of the thesis we will write about specific proof-of-stake algorithms. In the chapter, You can found a description of 5 proof-of-stake protocols – Algorand, Ouroboros, Ouroboros Praos, Casper, Tezos. The chapter will describe the properties of algorithms like throughput, scalability, security, privacy, failure-tolerance, liveness, safety, finality, etc. In the second part of that chapter, there is a summarization and a comparison of the protocols.

Proof-of-stake‘s next problem is a subjectivity for new validation nodes. By subjectivity we mean, when the node joins into the network, it has only a little amount of information about the whole network. So its decisions are based on subjective judgment – two different nodes can make other decisions, because their decisions are based on different information. In contrast, proof-of-work does not has this problem, because the proof-of-work network is based on objective decisions. For preventing from this problem the proof-of-stake protocols can use these types of mechanisms:

- Limitation of validating nodes – only currently bonded nodes.
- Withdrawal of stake must go through some period. Often called the „thawing“ period
- Forbid reverting blocks that were created before some length of blockchain.

Basically, the blockchain network is designed as peer-to-peer networks where all peers participate in the protocol equally to other peers. Network and protocol designs guratentee, that one node can participate in other nodes. But in some ways, the network based on proof-of-stake protocol may have the same trait of the server-peer network. That is because from all participants of the network, is created a list of chosen nodes (based on it‘s stake), that will validate transactions processed by the network. That trait of the network depends on the type of proof-of-stake approach of validator choose.

In the beginning, the base idea of the proof-of-stake protocol was: the voting power of validator is calculated from the amount of currency that the validator owns. But Vitalik, in July 2014 described a problem called „nothing-at-stake“ [38]. The problem presents this scenario: „*Validators can effectively break safety by voting for multiple conflicting blocks at a given block height without incurring cost for doing so*“. In other words: The system (network) is not punishing validator for voting to the different forks at the same time. In the concept proof-of-work networks, the validators (minners) are punished by distribution its mining power to two or more, forks where only one will be selected and others will be rejected. So, mining power used for that forks will be wasted. That problem is in a lot of proof-of-stake protocols be solved by „slashing“.

Slashing is process where validator creates deposit. Deposit is used as insurance for validator‘s inappropriate behavior, especially validation of the new blocks. All blocks validations by validator stand on validator‘s deposit. If the voted block will be approved by other validators then all validators get back their deposit with a transaction fee. On the other hand, the validator‘s deposit will be slashed.

5.1 Casper

Casper is a proof-of-stake protocol for Ethereum cryptocurrency. Casper is used in Ethereum named version 2, which migrates from proof-of-work algorithm to the proof-of-stake. Casper‘s protocol‘s family is divided into two separated algorithms. First, „Casper the Friendly Finality Gadget“ (FFG) [39] is a form of hybrid proof-of-stake/proof-of-work system. FFG is determined for migration of the Ethereum to the world of proof-of-stake networks. In the base, the FFG uses more srategy from proof-of-work protocols. And the proof-of-stake strategy is used to reach more scalability and lower energy consumption. The second one, Casper “the Friendly Ghost (TFG)“ and Casper „Friendly Binary Consensus (CBC)“ are pure proof-of-stake algorithms [40]. They are implemented by Vlad Zamfir. CBC or TFG are planned as a final protocol for the Ethereum network. Both protocols are long term projects and now they are in process of development.

Casper protocols can be also called Bounded proof-of-stake (BPoS) consensus protocols. „*They lock up part of their stake for a certain amount of time (like a security deposit), and in return they get a chance proportional to that stake to select the next block. Their voting power in the protocol is proportional to the amount of stake they are willing to lock up. Once the deposit is in place, it cannot be removed until a specified amount of time has passed. If these users are dishonest, they forfeit their deposit along with the privilege of participating in the consensus process.*“ [22]

One of Casper‘s family opinions is that Casper favours availability over consistency. Casper adopts a locking mechanism of stake. That means: the validator‘s stake is locked for a certain period of time in order to prevent invalid behaviour of validator. This is the main solution of „nothing-at-stake“ problem. Another issue, which Casper implements, is approval of new validators by old validators. Approval of new network‘s validators ensures more trustworthiness of the whole network.

A key goal of Casper is achieving “economic finality”. A block of the blockchain is economically finalized, when the block is a part of the blockchain forever or the block is denied and the actors voting that finalized blocks are penalized.

Economic finality in Casper is attained by validators‘ deposit. The deposit is submitted by validators. Thanks to the deposit, the validators can participate in voting the blockchain blocks. Finally, if the block was economically finalized, then the validators get back their deposit together with a transaction fee. On the other hand, if the block will be denied, then the validator‘s deposit is slashed.

Next feature for economic finality: Casper has an unique feature that Casper has parameterizable safety thresholds. By that functionality, one validator can have different thresholds for economic finality. Also, Casper does not limit the quantity of active validators instead of some other proof-of-stake protocols like Tendermint, whict limit that one.

Very discussed issue of consensus protocols is performance. On the other hand, validators (in context of PoW - miners) are working only on validation of transactions. Validators earn only transaction fees, so their income is limited by their permeability. That means, the validators have a direct influence to increase permeability of the network. However, the network performance is limited by slower validators during the sync.

Casper‘s protocols come from a second category called Byzantine fault tolerant based proof-of-stake protocols. It follows some rules which guarantee safety and liveness, and Byzantine Fault Tolerance up to 1/3 of validators.

The biggest advantage of Casper, like proof-of-stake, is that attackers can be identified and their deposit can be destroyed immediately.

Properties at a glance:

- Available - Casper’s nodes may have their blocks fork until they come to a consensus.
- Asynchronously safe.
- Live - Casper’s decisions can be live in partial synchrony, but are not live in asynchrony.
- Cartel-resistant - Casper’s entire premise is built upon resisting an oligopolistic attacker so no colluding set of validators can overtake the protocol.
- Safety - Depends on each validator’s estimate safety threshold.

5.2 Algorand

Algorand [41] is the first permissionless or permissioned pure proof-of-stake based blockchain. Algorand is an open source project developed mainly by MIT engineers lead by Silvio Micali [22]. The main idea is to build simple and fast consensus protocol and blockchain above them. This is the motivation, why the Algorand bases on hte pure proof-of-stake algorithm - the creators of Algorand say it is the protocol „without any complications“. Algorand has developed from 2017 and first launch was in June 2019. The whole consensus in the Algorand is created by expanded Byzantine consensus.

Some blockchain protocols suffer from balance against latency and confidence in transactions. For example, for high confidence we need long latency and conversely. Algorand tries to break dependency confidence to latency. Algorand fosuses on radical improvement a latency (performace) against actual existing blockchain protocols.

The key of whole Algorand is the new block generation use „new and fast Byzantine agreement protocol“ also called like „BA*“ [41]. This BA expands from classical message-passing Byzantine consensus with focusing on the performance and speed. BA* consists of 3-step loop. The BA* with more than 2/3 of the honest players ends in every loop with agreement by probability 1/3. This limitation the BA* inherits from classical Byzantine consensus protocols. The player is a randomly selected user among the a set of all users. The set of players of BA* is much smaller than set of all users, the set of players is s subset of all users. The selection of players is local - made, randomly among all users based on the user‘s stake weights. And this is the main property of the Algorand proof-of-stake views. For the election of the user, the BA* uses

verifiable random function (VRFs) [42] in private and non-interactive way. The computing is based on user‘s private key and public information of the blockchain. Since player is selected in private way, and adversary does not know who is a part of set selected users.

Each loop of the BA* consists of 3 steps. For each step the algorithm randomly selects the set of players from sets of all users. Also, members of the actual step do not know the next set of players, thankfully, there is no way, how to secretly pass internal value. Each step of the algorithm consists of players repeatedly exchanging the boolean value. All players‘ steps are not synchrony, so each player can process the loop at different times.

BA* works above the gossip network. That network is similar to the Bitcoin network. The network is used by players (a small set of randomly selected users). The player owns public and private key. The private key uses the gossip network to sign all messages. That ensures messages cannot be forged. Gossip protocol is communicating over the TCP protocol. Figure 6[6] shows how are players selected and then passed into the gossip network with final consensus broadcasting.

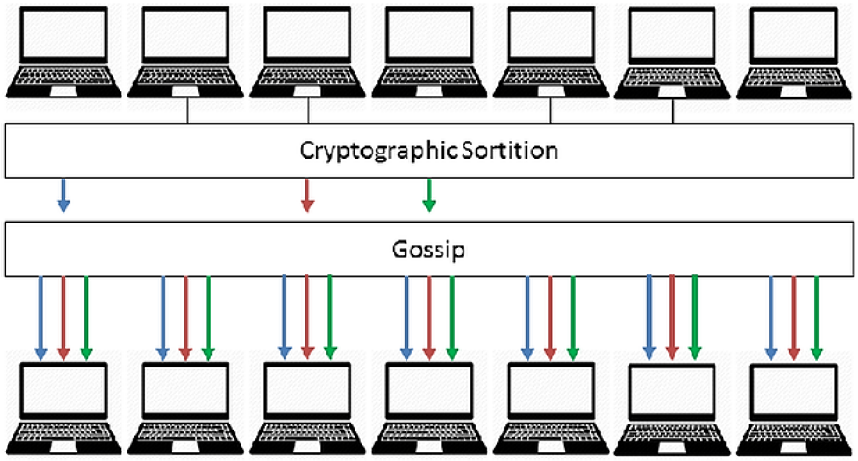


Figure 6: Algorand players selection

Although the Algorand is based on public a decentralized network, it still has some properties of centralization, especially in the early stages. One of them are tokens. In the early stages, the founder holds a set of tokens, which is not more than 49% and uses them to moderate network with only a few users. When the count of users increases, the percentage of the founder‘s tokens decrease. By that point, the implementator ensures network security in early stages.

Algorand‘s transaction history can‘t fork with overwhelmingly high probability and it avoids the double-spending problem. That probability is very low (it is about 10^{-18}). Traditional

6 https://people.csail.mit.edu/nickolai/papers/gilad-algorand-eprint.pdf

blockchain protocols support „forking“ of the blockchain ledger. After that can exist two short parallel forks of blockchain history according to different users. After several blocks of the chain have been added, we can be sure that all blocks are the same for all users. Algorand users can rely on the payments contained in the new block as soon as the block appears. This approach highly supports the time of a transaction process.

Although the network almost can‘t generate fork, the network implements various fork resolution procedures inspired by proof-of-work algorithms.

Because of using the BA*, avoiding fork agreement and other properties, the Algorand does not need any mining strategies. Users, especially players of agreement, do not stake their tokens (money) so the Algorand does not need any mining reward. The function of network is based only on users and BA*.

Practically, Algorand is able to create a new block of the blockchain in less than 40 seconds in experiments. Algorand is based on theoretical [43] and practical tests [44].

Algorand assumes for its work some theoretical values like:

- more than 2/3 of players are honest
- blockchain fork is created with probability less than 10^{-18}

Properties at a glance:

- Throughput – was only tested. 1Mbyte block of transactions with 50,000 users was confirmed in 22 seconds. That is approx 2200 transactions per sec.
- Scalability – Algorand achieves high scalability by choosing a players (committee) for each BA* round. Impact on performance depends of the size of the committee. In Algorand persists a problem with joining new user to the existing network. A new user has to fetch all existing blocks.
- Security - Algorand uses for security VRF function. Algorand is also based on asynchronous cipher methods. Algorand assumes, that 2/3 of users are honest, but Algorand does not actually implement some type of punishment for dishonest users. Hight part of security is also based on the non-fork implements and short steps of each consensus round.
- Privacy – Privacy is based on private keys. Users do not keep any private values except privacy key. Privacy key is used for signatures.
- Fault-tolerant – BA* protocol is inherited from classical BA protocols, that are fail-tolerant and benefit Algorand implement technique - Lazy Honesty [41]

5.3 Ouroboros

Ouroboros is a proof-of-stake protocol [45], which uses the election algorithm the Follow the Satoshi (FTS) validator‘s selection. Ouroboros is actually used in Cardano cryptocurrency.

The process of the Ouroboros consensus is based on the discreet division of real time into epochs and the epoch into slots. Practically, an epoch has *10*k* slots, where the slot‘s size is about minutes and *k* is a security parameter [15]. Each slot of epoch has its own voted leader.

Ouroboros uses cryptographic flow named publicly verifiable secret sharing (PVSS) [46]. By using PVSS, stakeholders create committee and vote new epoch leaders. Each stakeholder publishes a secret random number. That number will be revealed during the epoch. At the end of the actual epoch, a final public random number will be combined and determines each slot‘s leader for next epoch. The process is divided into these phases:

1) **Committee formation** – each stakeholder processes hte first part of PVSS scheme. Each stakeholder (committee member) creates a random number privately.
2) **Commit phase** - each stakeholder processes the second part of PVVS scheme. Stakeholder commits this encrypted number by stakeholder‘s public key. The stakeholder sends a message to other committee members. So each committee member should receive messages from other members.
3) **Reveal phase** – Each member sends an „opening“ phrase to other members and the members are available to open a secret – reveal the random number.
4) **Recovery phase** – a group of members can contain some adversary. In this phase, committee members check who has not revealed their number. And if some members have not revealed the number, the member posts his number to them once again.
5) **Create seed** – By combining (XOR) all secrect numbers we create a seed for FTS algorithm.

Those procedures explained above ensure unbiasedness during committee voting. The seed will be used for FTS algorithm. FTS algorithm needs randomness and seed is that randomness.

FTS collects all the smallest atomic pieces of coin, in the Cardono cryptocurrency called „Lovelace“ and their owners that is owned by stakeholder. FTS randomly selects one Loverlace and its owner becomes a slot leader. This part is a base of proof-of-stake behaviour, because more coins stakeholder owns, then higher probability stakeholder‘s selection.

To reduce a count of commit messages and to make protocol faster, the ouroboros limits size of committee member by minimal stake limit. The number of PVSS messages is in the m^2 category, where m is a committee size.

All messages sent in the Ouroboros network are synchronous. There is a chance of desynchronization attack (described in the Ourobors Praos paper - [47]) against Ouroboros if the condition of synchronous network is not accomplished. But phases of seed creation can‘t be asynchronous, because of security reasons and a chance of unbiasedness selection of the leader.

Properties at a glance:

- Throughput – has been only tested yet. The result was 257.6 trancation/sec [45] with the assumption mentioned in the Ouroboros paper.
- Scalability – scalability of Ouroboros should be well, mainly because of limited committee size. But that claim is not practically confirmed.
- Security - Ouroboros is a high secure blockchain consensus algorithm. Firstly, Ouroboros implementation focuses on security. Basic types of blockchain attacks were tested in [45]
- Privacy – Ouroboros does not focus on privacy primarily.
- Liveness – Liveness is ensured and based on theoretical calculations. Value is dynamic by protocol configuration. But practically is depended on multiple slot‘s time (minutes)
- Finality - Finality is ensured and based on theoretical calculations.

5.4 Ouroboros Praos

The Clasical Ouroboros is sometimes described as an evidence of proof-of-stake algorithms security. Ouroboros Praos [47] is an improvement of ordinary Ouroboros, which is based on practical experiences. Ouroboros Praos uses the same division of time into epochs and slots like ordinary Ouroboros.

For randomness, Ouroboros Praos uses a verifiable random function (VRF) instead of PVSS. VRF generates pseudo-random number based on an input – private key and other chain-based input. Anyone with a pair public key/ private key can verify, that the number was produced by chain-based input. It also verifies, that pseudo-random number was not produced before.

The agreed nonce is used as chain-based input for each epoch. The Nonce is produced by VRF values in block headers of the previous epoch. For each slot of the epoch, all participants calculate random number by VRF. If the number is less than value proportional to their stake, then the participant becomes a leader for that slot. It is possible that more participants, or none, become the leader of a slot. Code below shows, how this process works describes code 1.

```
T = calculateMyThreshold(myAmountOfStake)
foreach 1..nSlots -> i:
  y, proof = VRF(myPrivateKey, concat(epochNonce, i, "TEST"))
   if y < T:
    getReadyToGenerateBlockAt(i, y, proof)
```

Code 1: Pseudocode - VFR + leader selection

The described process is the main difference between Praos and classical Ouroboros. Participants are not limited by their stake to become a leader. Anyone with some stake can become a leader. On the other hand, by that process, there can be more leaders to one slot. And when more leaders are elected in the one slot, they will create a fork of blockchain, even if leaders are honest. That can lead to vulnerabilities.

Properties at a glance:

- Throughput – is not mentioned
- Scalability – a little bit better than Ouroboros, by another voting process
- Security - Ouroboros Praos is not vulnerable against grinding attack, than ordinary Ouroboros
- Privacy – Praos uses VRF, slot leader is not known publicly ahead of time. But that property is used more for security benefit than privacy
- Liveness – same as Ouroboros

5.5 Tezos

Tezos [48] is a decentralized blockchain that uses a proof-of-stake model for consensus. Consensus mechanism in Tezos does not have its own name. In this subchapter, we will call it Tezos as a consensus protocol name.

Tezos algorithm uses „liquid proof of stake“ LPoS, liquid democracy model. Model is based on direct and representative democracy, that floudly changes. Validators are called bakers and validation process is called baking – baking blocks.

The liquid democracy is illustrated by Figure 7 [7]. At the beggining, people vote for other people directly. The influence of voting is based on the stake - number of own tokens. That voted people (delegated) can vote for other people by their own tokens and also by assigned tokens. If a person does not like their own voted delegate, then a person takes back own vote. The described process is also found in Delegated proof-of-stake DPoS.

To become a baker (validator), a baker must hold at least a roll of tokens (roll corresponding to 10 000 tokens). Each block of the blockchain is baked by baker and signed by 32 other random bakers. If blockchain is successfully baked and signed, then the successful baker gets a reward.

7 https://static.blockgeeks.com/wp-content/uploads/2019/04/image3.png

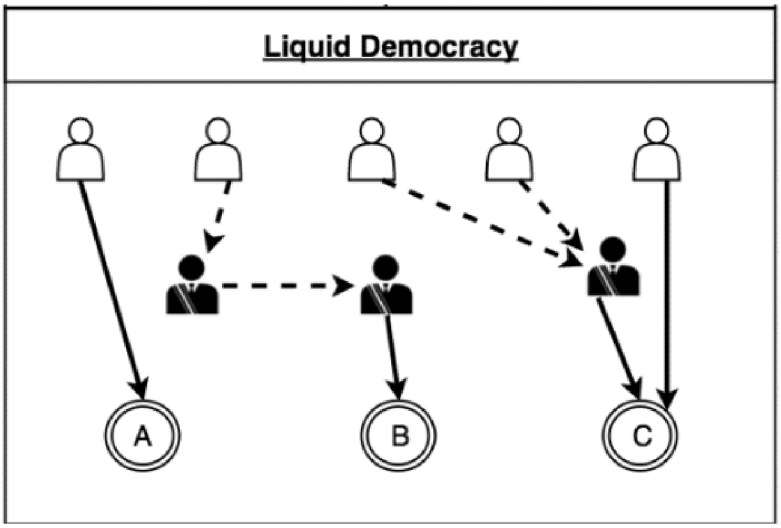


Figure 7: Liquid Democracy

5.6 Protocols comparison

This chapter is a summary of protocols comparison. Comparison is based on theoretical information, that is described above. Also, information is based on the papers cited in bibliography part of the paper, and the paper [49]. In some parts of table is cited short explanation why the protocol has those properties.

The first table (Table 1) compares protocols in fundamental properties like throughput, security, etc..Second table (Table 2) is focused on vulnerability comparison, some information in that table is based only on theoretical information, because a vulnerability of protocols has not been practically confirmed.

	Throughput	Scalability	Security	Privacy	Liveness	Finality
Casper‘s	…..	Scalable But not mentioned how	Based main on stake deposit	----	No liveness proof	…..
Algorand	1000 transactions/sec	Very high	Good – VRF	Good – VRF	About minute (with strong synchrony)	Immediate
Ouroboros	257 transactions/sec	High	Secure focused blockchain	Not focused	Multiple of slot‘s time (minutes)	High, Ensured
Ouroboros Praos	…..	…..	…..	Little better than Ouroboros	YES	…..
Tezos	40 transactions/sec	Low	…..	…..	…..	…..

Table 1: Protocols properties comparison

	Casper's	Algorand	Ouroboros	Ouroboros Praos	Tezos
51% vulnerability	PoS benefit – higly secure against attack Easy to handle – economic finality	Cannot resist 34% of adversarial control	PoS benefit – secure against attack	----------	PoS benefit – higly secure again attack
Double-spending Attack	No mechanism to avoid that Ordinary PoS level vulnerability	Avoid forks → avoid double-spending	Possible but with lower change than clasical bitcoin	Lots of forks – but consequence?	Defense
Assumptions	Requires a 2/3 graction of a deposited stake to be controlled by honest nodes No explicict claims about network	Requires a 2/3 graction of a deposited stake to be controlled by honest nodes	Network is synchronous Stakeholders do not remain offline for long period of time	Network is synchronous	---
Time De-synchronization attack	-----	Immune but high influence to liveness	If more than 50% parties get desynchronized our protocol can fail	Susceptible to a desynchronization attack	----
selfish mining	No mining	-------	Own reward mechanins Selfish mining attacks are neutralized	---------	---
Nothing-at-stake	Possible but reduced: Detection the vialation → penalize melfeasant validator	No defence	No big defence But, no motivation for stakeholders no higher profits by joining the attack	--------	---
Grinding attack	----	----	Tossing protocol – is not possible	Possible	---
Fake stake	----	----	Secure against attack	--------	---

Table 2: Protocols vulnerability comparison

6 Design

The goal of this thesis is a real comparison of proof-of-stake protocols, to evaluate cons and pros and propose possible improvements. For that purpose, a testbed will be created. This chapter will describe suggestion of a testbed‘s solution. The capter is divided into few separate parts. The first part is about basic testbed idea and base architecture. The second part of the chapter explains concrete suggestions of a solution. And, the third part mentions some used technologies and techniques.

In the previous chapter, we compare protocols theoretically. On this basement, we will design a testbed. Its design mainly focuses on the creation of the simulator, which will make same conditions for each proof-of-stake protocol. Throughput, scalability, and security are primary properties that designed testbed is focused on.

6.1 Design idea

In the third chapter, we were talking about the stack model of blockchain technology. We will use this model for the basic design. The stack model consists of a few layers, from physical (network) layer to application layer. The layer of consensus protocols lay between them. For design purposes, we separated the stack model into 3 parts, by Figure 8[8].

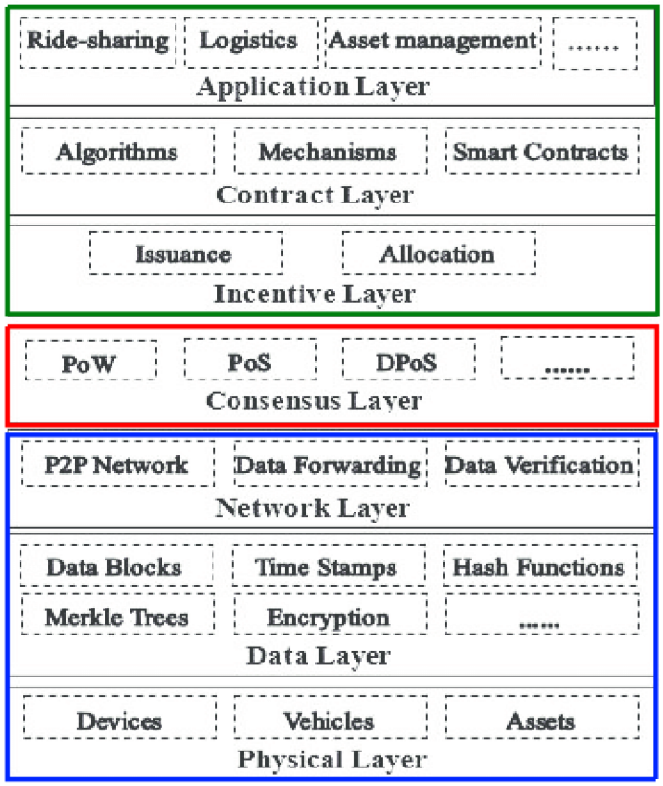


Figure 8: Stack model divisition

8 https://www.researchgate.net/figure/An-ITS-Oriented-Blockchain-Model_fig4_332320425

First, the blue part in the illustration consists of layers under the consensus layer. These layers care about networking, communication nodes between themselves, encryption and so on. That layer ensures the foundations of the consensus layer. In the context of the testbed, the layer is dispensable for implementation. The second part is consensus layer, coloured red in the illustration. This part is the whole center of the implemented testbed. Last layers are above the consensus layer. These layers get some practical sense to the blockchain technology. These layers contain some specific cryptocurrencies algorithms, smart contract implementation, etc. In the highest layer of this part, we can found user‘s applications for blockchain technology using.

As a part of application, we will create simulator to simulate bottom, blue, part of the stack model. In the real world, the blockchain technologies, especially blockchain technologies, which use proof-of-stake algorithms, communicate over the internet network. The internet is a huge, non-foreknow network. The internet is almost always illustrated like some network – cloud. Blockchain uses the internet network like a medium for communication, but the blockchain technology builds its own network topology above it.

Proof-of-stake protocols use the blockchain network, which is built over internet network. Blockchain network is P2P (peer-to-peer) network. So, in the simulation does not necessary to simulate internet network, it is possible to simulate only P2P network. This decision leads to some cons and pros. The disadvantage of the decision is a loss of some connectivities with real world using. Simulation loses unexpected behaviour of internet network and delay in communication, especially in communication across geographically distant continents. On the other hand, a huge positive is that the testbeds will be more focused on the main topic - consensus protocols. Internet simulation can add some unexpected and non-deterministic behaviour of implemented testbeds.

Those informations above lead to implementation of only P2P. P2P network will be implemented as a simple TCP based network. The network will not pay attention to any of complicated parts like routers, switches, etc. All nodes will have their own connection to all other nodes. It is illustrated by Figure 9[9].

The red part, is a layer of the consensus protocols. The layer implements some specific proof-of-stake protocols. We can assume, that every node in the P2P network is a participant of the blockchain network. So, basically, it is possible to implement the same proof-of-stake behaviour on each node. On this base, we can instantiate the same behaviour on every network node.

9 https://cdn.steemitimages.com/DQmZAFtWv5imtESqJdab7kZiCycT5FMKutoda2yvtBR6Ds2/peer-to-peer-network-1.png

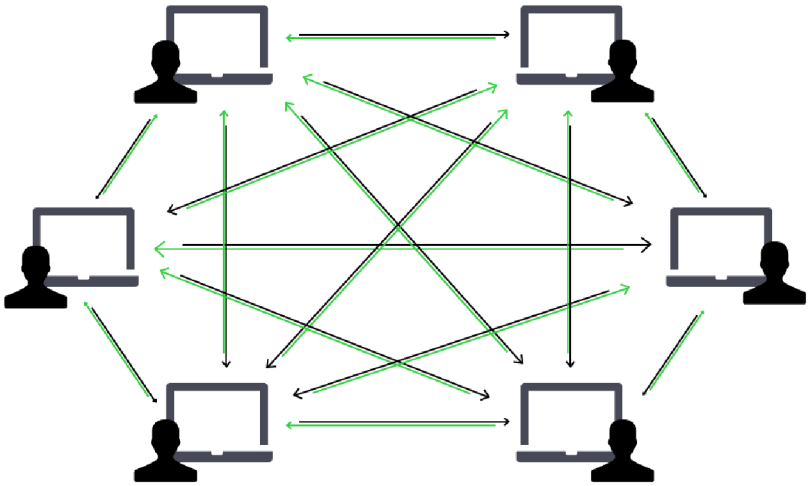


Figure 9: Peer-to-Peer Network

For the implementation of more proof-of-stake protocols in one simulator some procedure needs to be created. For the implementation of one protocol, it is possible to start specific design and programming right at the moment, because in paragraphs above, we demonstrated the basic way of implementation. But, testbed, dedicated to more protocols, needs to create a design for that. Layers of networking are solved, so we can focus only on the consensus layer design. The main idea is to separate common properties, implement them like the base of consensus layer. That creates basic interface for implementation of various proof-of-stake protocols. A simple idea is demonstrated, with selected protocols and using of network layer, in the Figure 10.

The green part, does not need any design or implementation, because we focus on the results of consensus layer. Instead of that part (layer), we will create metrics, that will evaluate results of the testbed. Based on the evaluated results we can create comparison and design improvements of actual protocols. Specific metrics and way of evaluation are described in the chapter below.

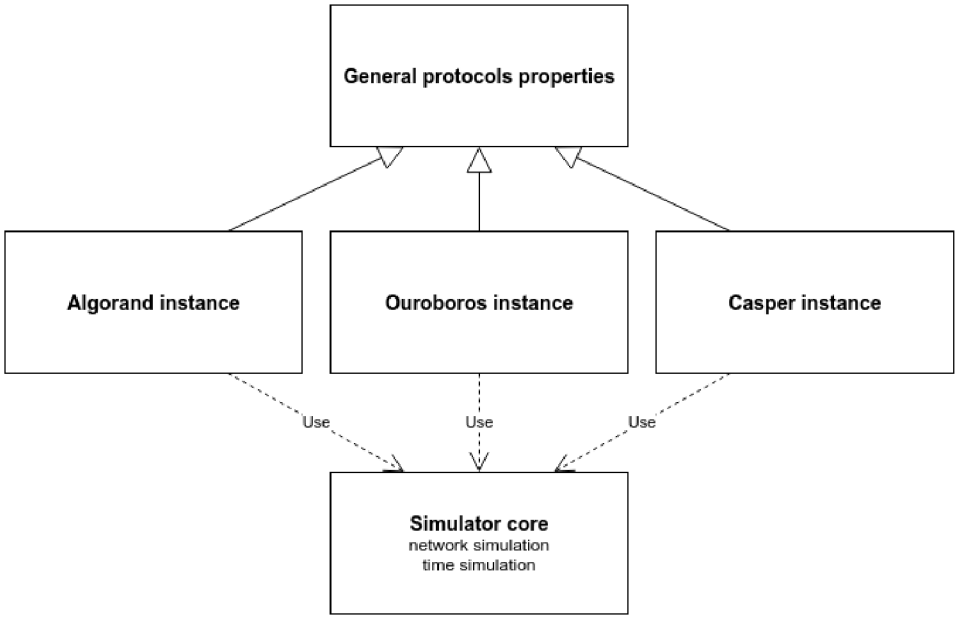


Figure 10: Basic idea of the consensus layer design

6.2 Network part design

In the previous chapter, we indicated the way of the design of the network part of the blockchain stack. All things considered: Common idea is to build point-to-point network and simulate higher layers of the stack model above it.

In the process of designing, we were looking for a framework or library that perfectly supports simulations of various types of network. On the other way, it will be enough, if the library supports only IPv4 and mesh topology. Also, the list of requirements is a possibility to implement its own behaviour of each participant of the network. Frameworks fulfilling requirements are Mininet [50] and NS-3 simulator [29]. For the implementation of the testbeds, we decide to use the NS-3 simulator, especially for its bigger user group. The next reason why we use this framework, is that the NS-3 is used in the Bitcoin simulator mentioned in the chaper 3. So, the framework is verified for blockchain simulations.

NS-3 is a discrete-event network simulator. NS-3 is a free software under GNU GPLv2 licence. The simulator focuses on networking simulation. It supports simulation, from simple networking like point-to-point topology to complex topologies that can contain devices like switch, router, wifi AP, etc. A big advantage is that the simulator bases on the discrete event, which enables to run simulator in the simulation time, not in real time.

The framework is a complex network simulator. It puts together small parts (layers) to final simulation. The framework is implemented like a connection network layer into functional unit. In the beginning, some implementations create a physical layer of network – nodes, topology.

Within next implementations, we create the next network layers that can use implementations from lower layers. On the last application layer, it forms applications to nodes. All these parts are implemented in the core of the simulator. A specific implementation of simulation only uses layers and connects them into functional unit. In the testbeds, we will use that procedure too, but in the end, we will implement our proof-of-stake specific applications for nodes of the network.

NS-3 is implemented in the C++ programming language [51], but it provides API for python programming language. The API can be used only for simulations that uses already implement parts, devices, nodes in the core of framework. For own implementation of the testbeds we need to redefine behaviour of nodes, and this code is available only in C++ source code. So, the implementation of testbed will be written in the C++ programming language.

As the physical design of topology, we decided to use bus topology. Decision is made with regard to NS-3 simulator. Implementation of more than 10 nodes in the mesh topology is quite complicated because every node needs *x-1* ports where *x* is a count of nodes. Bus topology is almost the same as designed topology mesh. But in the bus topology does not exist a dedicated physical connection between two nodes. It is illustrated by Figure 11[10]. In the bus topology, every node has one port, that is connected to the common line. In the routing, bus topology does not change routing against mesh topology. Nodes can communicate directly with each other.

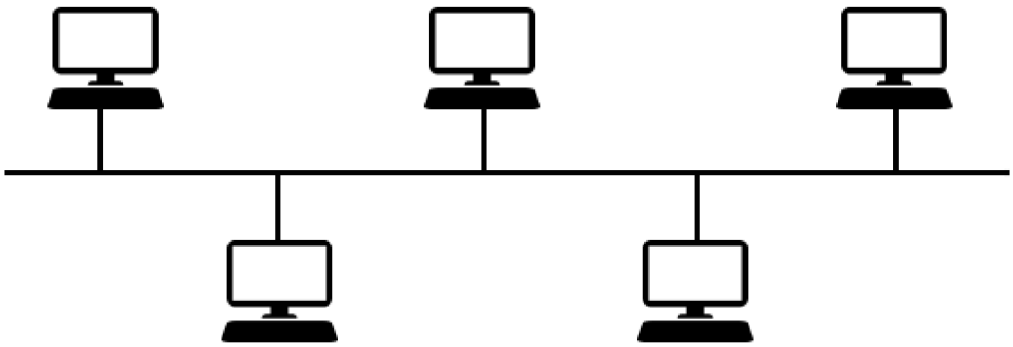


Figure 11: Bus topology

For routing, we will use classical IPv4 stack. By using IPv4 stack, we will avoid any network-specific problems. Theoretically, the IPv4 stack is possible to observe millions of nodes, but practically, it depends on the simulator performance. In the design, we suppose simulation from a hundred to a few thousand nodes.

10 https://1.bp.blogspot.com/DWD0kH2Ag4Y/XQR9Fmqi5VI/AAAAAAAAACs/me70TxvwhHsuRAGFxLNeIi0brKPJTaiuwCLcBGAs/s1600/bus.png

6.3 Consensus part design

For testbed, we choose protocols: Casper – the Friendly Ghost, Algorand, Ouroboros. The selection is based on the previous chapter. In the previous chapter, we describe proof-of-stake protocols and these three protocols are representative of different approaches of the proof-of-stake consensus. Also, protocols declare different behavior in various states. By this decision, we expect high differences in the comparison. It will highlight the positive and negative of its approach, and it will lead to the final recommendation of proof-of-stake protocol‘s design and implementation.

In the Table 3 is a small repetition of three protocol properties and differences by chapters above.

Casper (TFG)	Algorand	Ouroboros
BFT based protocol Locked deposit Voluntary size of the deposit Not limited count of validators Scalable	BFT based protocol – BA* Deposit = count of owner all tokens Limited count of validators Random selection of committee VRF function Very high scalability 1000 transactions/sec Liveness – lower minutes	FTS based protocol Deposit = count of owner all tokens Use PVSS flow Selection of validators – FTS 257 transactions/sec High scalability Liveness – minutes

Table 3: Differences of proof-of-stake protocols

As it was defined already NS-3 simulator divides whole simulator implementation into layers. The simulator contains the application layer part, where consensus protocols are implemented. The implementation of proof-of-stake protocols can be divided into three parts: Data part of the blockchain implementation, common properties of proof-of-stake and blockchain, specific properties of the protocol. By the data part of blockchain is defined as the data stored in the blockchain – whole blockchain composite by blocks and transactions – blockchain database. The division is based on a pre-condition that the implementation of the data part of blockchain is almost same for all protocols. If that pre-condition will be broken during implementation, it is possible to implement the required feature into the data part of blockchain without any influence on other protocols, other protocols will not be using this feature.

The data part of the blockchain is designed like classical blockchain structure. It is illustrated by Figure 12[11]. In the context of selected C++ programing language, that the blockchain is designed like class that contains a vector of block class instances.

Common properties will be implemented in the base class for specific proof-of-stake protocols. Common properties will contain regular transaction generation, receiving new blocks, receiving new transactions and other properties. Also, in the common properties will be calculated with every node‘s stake, every node will hold its stake.

11 https://blog.trendmicro.com/wp-content/uploads/2018/03/blog-1024x349.png

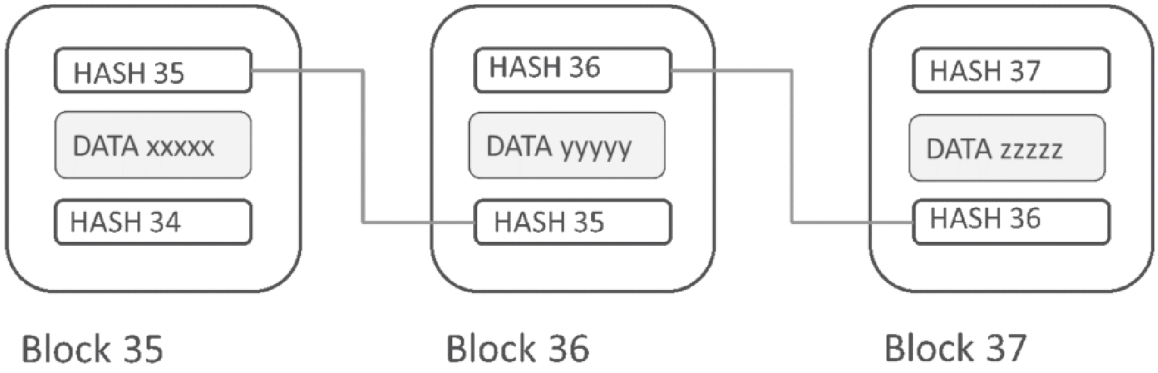


Figure 12: Blockchain scheme

6.4 Design of comparison – metrics

The protocols‘ properties, defined in the chapter above, will be compared and researched. There are properties defined from another perspective. Properties are converted in the way, that allows them to express properties as a number. By that, it is possible to get a number result of specific protocols and it is able to compare them and visualize in tables or graphs.

- Throughput – is defined as a count of flowed blocks through network per defined time. In the testbed, we will be calculating the count of transaction per second. Throughput comparison of protocols will be based on the same network‘s pre-conditions. For example, the same network size, same count of network participants. As a pre-condition, it calculates the same behavior of network participants – count of honest/dishonest participants. The metric acquires a positive number from 0 to theoretically infinity.
- Scalability – Metric, that express scalability of the blockchain network, which is related to the throughput. It explores the influent size of the network (count of participants) to performance. In the scalability, it will not calculate with dishonest participants. By discovered relation *throughput x scalability* is defined scalability in this thesis.
- Security – In basic, there are defined 8 types of proof-of-stake attacks (chapter 3.1.3). We define the security of the protocol like count of attacks that the protocol is resisted against. The security metric acquires number on a scale 0 to 8, where 8 is the best result, protocol resists against all defined attacks.
- Privacy – Privacy defines if data in blockchain are visible to all nodes or public. Privacy will not be the main goal of protocols comparison - of the practical part of the thesis (testbed).
- Failure-tolerance – It defines, that a protocol is resistant against breaking pre-condition to network part (layer). In the testbed, there was not defined numeric metrics that describes it. But a part of tests and comparisons will be fail-tolerance resistance testing.

- Liveness – It was defined above (chapter 3.1.2). The liveness states as amount of time till generated transaction will be available for other network nodes. In that context, the testbed will compare the protocols.
- Safety – A research of safety in the testbed will be calculated like boolean value – *True/False*. The value depends on the safety definition above (chapter 3.1.2). That means: if protocol complies the definition, then the value is *True* else it is *False*.
- Finality – In the context of implemented testbeds, the finality is defined as time, after that the block is finally accepted, and there is no way how to change or delete block and its data from the blockchain database.

7 Implementation

This chapter writes about the implementation of the testbed. The whole implementation process is divided into 3 parts. The first part is an implementation of the core of the testbed, we can call it a framework. The Second part is description of an implementation of two selected protocols [12]. The third part focuses on the validation of implemented testbed against the real protocol implementations.

We will write about these three parts in this chapter, but before that, we will describe detail technical solutions by framework NS-3.

7.1 Source code organization

NS-3 simulator‘s implementation is structured into modules. Some of the modules are core modules for base simulator running. We can find there basic implementation like the simulation of the discrete time, logging, base network communication, etc. Other modules use or extends the functionality of base modules and create applications for specific purposes, like point-to-point communication, wifi network, etc.

Each of these modules contains source code, that implements the functionality of the module. Moreover, almost every module has some runnable examples of using this module or runnable tests. Programators can inspire there because in examples are mostly main functions of module showed. One module can use the functionality of other modules. So, wifi module uses core modules, log modules, etc..

The way of implementation of our testbed into NS-3 is based on the module source code organization. The testbed is designed like one module of the NS-3 simulator. The testbed module is called „proof-of-stake-testbed“. This module uses other modules for communication, logging, etc.. By that, the source code of the testbed stands alone. The source code is separated directly from NS-3 simulator. This also creates other benefits like a comfortable use of the version control system.

12 We implemented only two types of protocols. The Casper implementation was above our planned programming capacity – Casper protocol was not implemented completely.

The module of the ns-3 simulator is organized as a directory. Our implemented module is based on the existing modules of NS-3 simulator. So, the module is designed and implemented in this directory structure, that is illustrated by Figure 13:

- **docs** – documentation folder. Contains document about testbed.
- **model** – main folder of the testbed functionality. We can find there almost all source code. The directory is divided into logical parts:
  - core models – source code for testbed basement, called a framework. There are source codes of blockchain functionalities and basic applications for work with blockchain.
  - Ouroboros models – extended core models and implemented Ouroboors protocol behaviour.
  - Algorand models - extended core models and implemented Algorand protocol behaviour.
  - Casper models - extended core models and implemented Casper protocol behaviour.
- **helper** – it is a directory that contains supportive source codes for models. We can find here network factories=blockchain factories. The directory is also divided into logical parts:
  - Core helpers
  - Ouroboros helpers
  - Algorand helpers
  - Casper helpers
- **test** – source code for tests
- **utils** – source code for independent functionality. Supports functions and other independent programs.
- **examples** – code of runnable examples of the testbed. The directory contains implemented applications that create field for comparative throughput, security, scalability, and other explored properties of protocols.

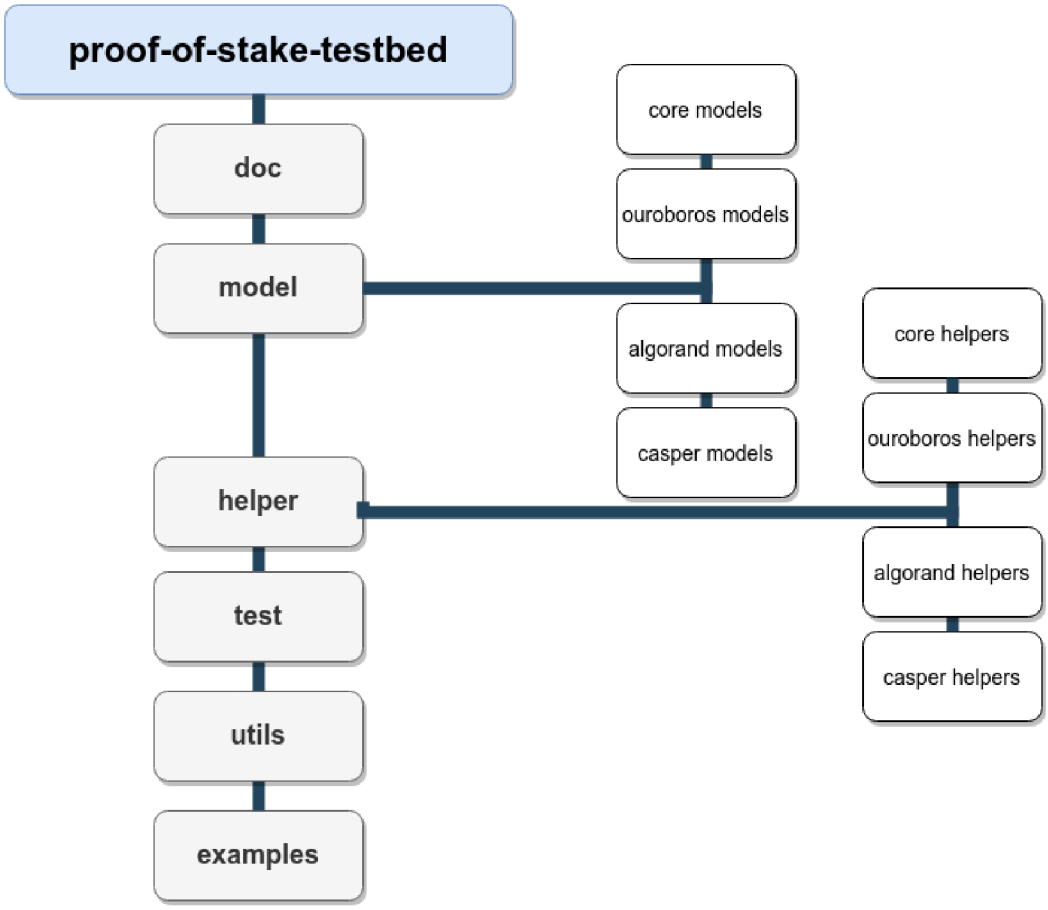


Figure 13: Module structure

7.2 Core models

Code models are source codes that provide basic functionality of blockchain and provide the basement for implementation of specific proof-of-stake protocols.

This list of models is logically divided into 3 parts: Blockchain simulation, proof-of-stake based application, helpers.

In the blockchain simulation is necessary to implement the blockchain database. The blockchain database is implemented by OOP (Object-oriented programming). The base structure is shown here, Figure 14, in the class diagram. The structure is composed of three elements BlockChain, Block, and Transaction.

Blockchain is the main element that will be created in every instance for each network node. Blockchain provides an interface for working with the whole blockchain database. Blockchain instance contains all blockchain blocks. Saving block into the database is implemented like a two-dimensional vector by C++. By that , it creates chain depencency of blocks and also

forks support. So, the database of blocks is *NxN* sized vector. The length of the vector expresses block dependence between each other and the height of the vector are forks of blockchain support.

Block is an element created in every round of consensus protocol. Block is mainly created by validator(s). Block contains transactions added by validator(s).

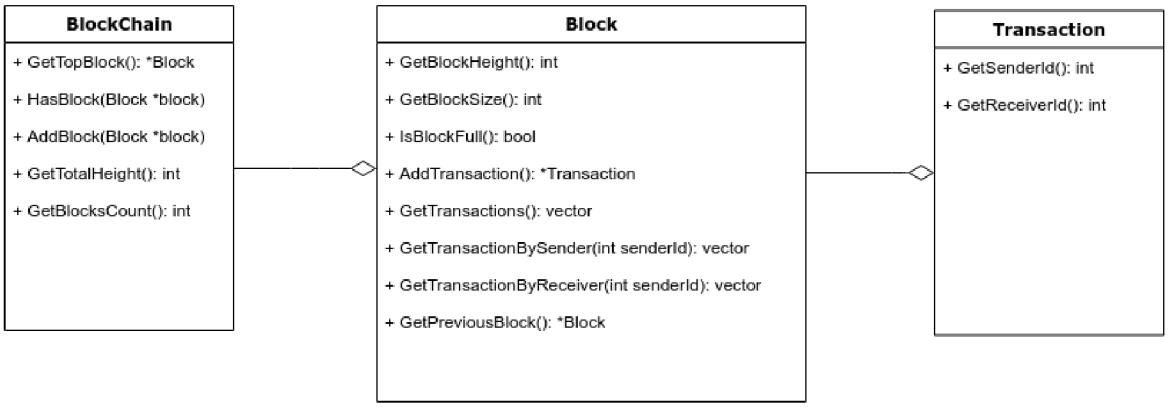


Figure 14: Blockchain class diagram

In the design chapter, we introduced a structure of the organization NS-3 applications. For the reminder, application is a program running on the network node. In the TCP/IP stack model, the application is on the top level – client application.

Application of proof-of-stake is an application that extends the clasical NS-3 application. It creates an abstract class that provides basic functionality for proof-of-stake protocols‘ simulation. Generalized functionality of proof-of-stake protocols si implemented there. By the abstract application, it creates protocol‘s specific application for Algorand, Ouroboros, and others proof-of-stake protocols. The abstract application ensures basic operations with blockchain. For example, the abstract application ensures receiving new transactions. But, the abstract application does not ensure protocol-specific behaviour, like voting for the leader in Ouroboros protocol.

List of properties that provide abstract application:

- **Receives messages** – receives messages and resolves a type of message.
- **Receives transactions** – process after receiving transaction.
- **Generation of transaction** – random generation of new transactions.

The abstract application implements only receiving a message and resolving a type of this message. If the abstract application does not know the type of message, then abstract application lets parse message by specific protocol application.

One type of messages that abstract application parse, is a transaction message type. The transaction is received, saved in the node and broadcasted to other know nodes.

The abstract application provides a generation of transactions. The generation of transactions is based on random generation. The generated transaction is sent broadcastly. Actually, it is possible to set up the frequency of generation in two ways. It is possible to generate transaction with frequency by classical random C++ function or by Possion distribution. Both ways of generation support setting main parameters in the config file. For random function, it is possible to set up the maximal time of two generated transaction delay. For Poisson distribution, it is possible tu setup mean value.

Core helpers. The implementation creates one main element that lightens the implementation of the client‘s applications. If we think of ordinary proof-of-stake programs, then the program has to solve lots of complicated things. Especially, things relate to a decentralized way of communication. The way how to lighten the implementation is the creation of the main element.

As a solution, we implemented the „helper“. Helper is one instance of the class, singleton. Every application (node) in the testbed has its own access to this helper, ilustrated in Figure 15. Helper is independent of implementation of applications. Helper provides only interface for questions, tasks, which can be complicated implemented in the node. Helper supervises consistency. For example, we can demonstrate the function of helper on the node question, „How many coins has node number 5?“. In a decentralized way, it is a quite complicated question. Node has to go through its local blockchain copy and calculate node stack. Also, the node has to deal with some problems with the synchronization. But, if the node has access to the helper, it asks the helper. Helper stores this data locally and returns answer clearly, without any synchronization problems, because the helper provides the same data for every node – singleton.

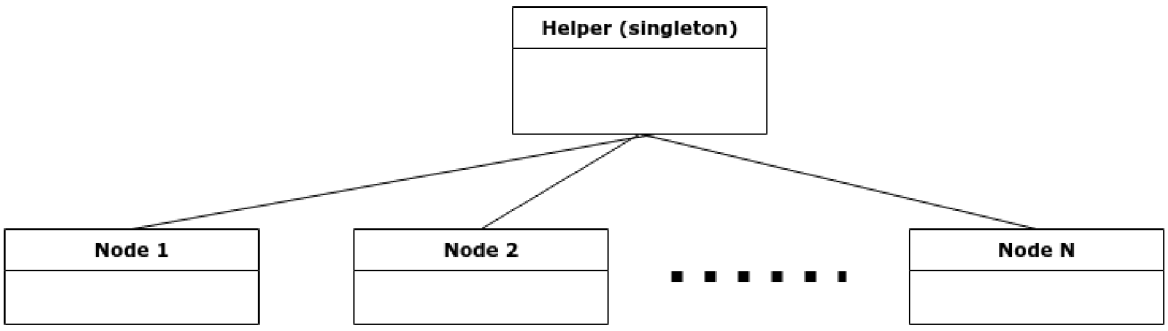


Figure 15: Helper structure

7.3 Communication model

7.3.1 Messages

NS-3 simulator supports the communication between nodes byclassical string messages. But, for our communication, we need to transfer more structured data, like sending new transactions, blocks, parts of blockchain, and others.

For sending structured data like the text we decided to use JSON format. The decision of using JSON format was based on the property, that JSON is human-readable.

The base structure of JSON is:

```
{
     type: 12,
     timestamp: 1588490943,
     …..
     other_data
     …..
}
```

Code 2: JSON message structure

Every message contains two required fields – type, timestamp. The type field demonstrates a type of the message. Type is an integer. By the value of type field, we can sort message. The second parameter is a timestamp. The value of the timestamp field is a classical Unix timestamp. The field contains a value of message created time. Also, a message has another specific fields. Other fields depend on the type of message.

JSON is a string with a special syntax. It can be created like a classical string without any support of programming language. But, creating and updating JSON as a plain string is quite complicated and uncomfortable. For better programmmer comfort, it is preferable to use some library that is supplied by the user (programmer) friendly interface to work with JSON.

For this testbeds and simulator was chosen library RapidJSON [52]. RapidJSON is a parser/generator of JSON for the C++ programming language. NS-3 supports adding libraries to source code of simulator. So, RapidJSON library was added to the simulator in accordance with NS-3 documentation.

In the source code, every JSON serializable and JSON deserializable message (transaction, block,…) implements two methods. Methods *ToJSON* and *FromJSON*. Method *ToJSON*, creates rapidjson document instance with filled data. Rapidjson document instance can convert to the string. Method *FromJSON* is a static method. It receives rapidjson document as a parameter and returns the instance of the deserialized object.

7.3.2 Network

In the design chapter, we designed bus topology network. This type of network is suitable for tuning the implementation of consensus protocols. Also, this type of network is suitable for small or specific consensus protocol tests. This is because the bus topology network is characterized by straight and expected behaviour, without any network complications, because every node of the network can contact other nodes directly.

But this bus topology network is not suitable for more complex simulations of blockchain technology. Bus topology network absences some type of unexpected behaviour across the network communication.

As more complex network topology we introduce two network topologies - in the Figure 16 and Figure 17[13].

The first network is compound by *N* local networks. The local network contains *M* nodes. The implemented network supports 254 nodes on each local network, but practically, we use local networks with size about 8 nodes. Local networks are connected together by one node of each local network. This creates the whole network topology. Edge nodes, that connect local networks together, are critical nodes. We can investigate behavior of network without the proper function of these nodes. Network addressing is based on the IPv4. Address space is 192.168.0.0 with mask 255.255.255.0. That means, we can create 254 networks, and every sub-network with 254 nodes. So, theoretically we can create and run 64 516 nodes. Practically, that count is a little bit lower, because some nodes are not only in one network. However, as a precondition for the testbed, the count of possible nodes is sufficient.

The second network type is compound by *N* nodes. Each node has a connection to another *X* random node. To be exact, $X = 8$. This type of network is called a distributed network. Address space is 192.168.0.0 with mask 255.255.0.0. It is possible to create *N* nodes by that, where $N=64516/X$. The Network is resistant against the fail of nodes. This type of network is highly used in the blockchain simulations.

13 https://networkcultures.org/unlikeus/resources/articles/what-is-a-federated-network/

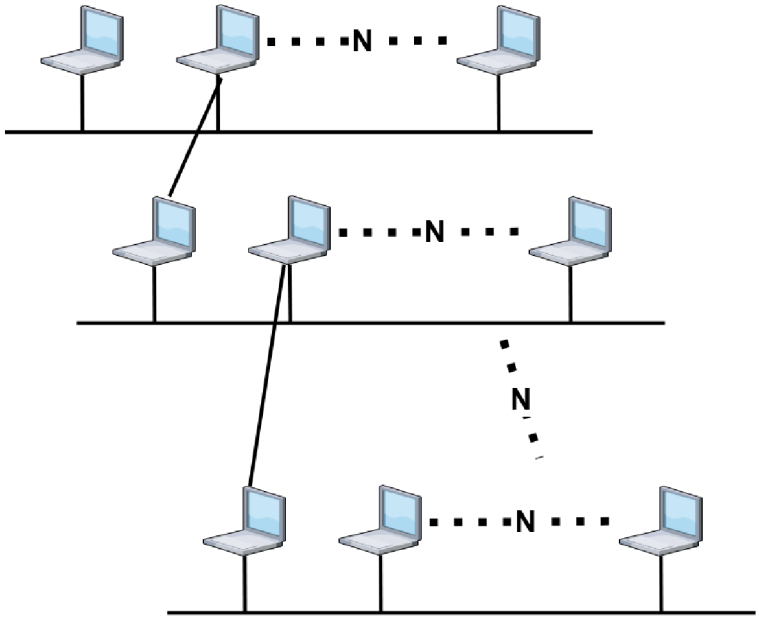


Figure 16: Network version 2

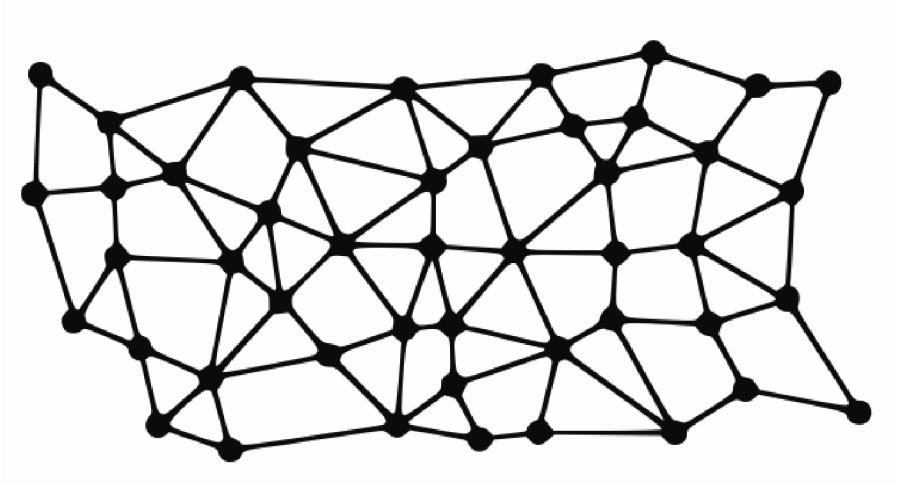


Figure 17: Network version 3 - distributed network

7.4 Ouroboros implementation

7.4.1 Implementation

For ouroboros implementation into testbeds, it needs to follow three steps. Ouroboros application creation, Ouroboros helper creation, extension types of messages.

Ouroboros consensus protocol is compound by four protocol steps (described in the chapter 5). Commitment phase, Reveal phase, Recovery phase, and Follow the satoshi phase. All these phases (steps) are implemented into the Ouroboros simulation. But, some parts of implementation are lightened against the original implementation of the protocol. All changes of lightweights are described below. Things that were lightweights have no, or only minimal impact to the results. Reasons why these things have no impact are described below too.

In the commitment phase, the node creates a secret, encrypts it, and broadcasts it to other nodes. In our implementation, the generation of secret does by a random function. So the secret is a pseudo-random integer. When other nodes receive the secret, nodes save it and broadcast it to other nodes. By that, the secret spreads over the whole network. All nodes save that secret. The secret will be used in future phases. Especially in the fourth phase for the FTS algorithm. In original code, the secret is encrypted by RSA. In our implementation, the secret is not encrypted and it is plain-text broadcasted. That is the main part of lightweight, with no impact on results. In the testbed, we focus primarily on the consensus protocols properties and vulnerabilities, but we do not focus any other types of vulnerabilities. In this thing, we did not focus on the vulnerabilities that are caused by non-encrypted data, that go through the network.

Second, revealed phase is left out in our implementation, because in our implementation secrets are not encrypted. Also, third phase is lighweighted, because we can consider the same precondition as mentioned in the paragraphs before – encryption by RSA. This point can has some minimal impact on network load and we have to think about it in future simulations.

The fourth phase is a phase, where Follow the satoshi algorithm is used. In our implementation, Follow the satoshi algorithm is lightened. For lightening the FTS implementation, it uses the services of global helper (described above). Helper generates a leader of the slot by pseudo random and by a selection of one coin (Lovelace). Owner of this coin becomes a leader for slot. Helper in basic implementation generates only one leader for one slot. Helper saves that the leader of slot is this node. So, when another node wants to know, who is the leader of some slot, then helper returns that value from its internal memory.

7.4.2 Evaluation

Here, we evaluate the proper implementation of the Ouroboros protocol. We will evaluate protocol implementation as a blackbox. We choose a few main properties, and then, these properties will be compared – Ouroboros paper definition with our Ouroboros implementation. As main properties of the Ouroboros protocol implementation we choose:

- Ouroboros needs a synchronization.
- Ouroboros can create blockchain forks.
- Ouroboros uses FTS (Follow the satoshi) algorithm.

Synchronization – In the first phase, we ran protocol with the synchronized state. The protocol testing was running a few times with different protocol‘s setting and we were observing, if the protocol‘s behavior is proper – protocol works in loops and in each loop a block is created. Finally, in every executed test, the protocol produces a block in every loop, and all nodes cooperated on the consensus.

In the second phase, we ran a protocol with broken synchronized state. We broke assumptions only on some nodes. We had been observing, if the protocol will be able to work without synchronization assumption and how the protocol reacts to that. Finally, the protocol was able to run only partially. It creates, especially on the broken nodes, „strange“ blocks (blocks that do not fit to the global blockchain database).

Fork – In the first phase, we ran the protocol with only one leader per loop and we observed, if protocol creates some fork. The protocol in this configuration should not create any fork during testing. Finally, the protocol in this configuration did not create blockchain forks.

In the second phase, we ran a protocol, and, for every loop we selected more leaders with different voting opinions. We expected that blockchain creates some forks. The number of forks during blockchain history should be equal to a number of different voting opinions. Finally, the protocol created a number of forks that were equal to the number of opinions.

Follow the satoshi - Ouroboros practically uses the FTS algorithm. However, in our implementation of the Ouroboros, Ouroboros does not use the FTS algorithm. In our implementation, for the leader‘s selection was not used pure original FTS, but for selection of the leader we used a centralized helper element, that simulates the FTS algorithm. We compared FTS with our solutions by blackbox approach. We supposed, that the FTS does not need any communication. It needs only a seed as an input and as an output the FTS produces the leader of the loop. Our central solution has the same input and output of leader selection. We established a solution of the FTS as resolved, based on this idea.

7.5 Algorand implementation

7.5.1 Implementaion

The process of the Algorand node is an asynchronous process with few steps. Steps are:

1) block proposal
2) soft vote
3) certify vote

Algorand‘s functionality is based on the two concepts, Variable Random Function, and Participant Keys. The implementation of these two concepts is discussed below.

On the white paper, we can find out, that the Algorand process is defined as more accounts on one node. In our implementaion, we neglect this and we create only one account in one node.

Implementation of Algorand extends simulator basement – application extension, helper extension, message types extension with Algorand specific messages.

Algorand helper extends the base helper that proposes the function of node stack manipulation. Algorand helper uses this function and on that function, it creates support for VRF simulation. That means, VRF is not implemented in the applications but VRF is simulated by a centralized way. Helper creates committee members, that are originally based on the VRF, and propose them to applications (nodes). This approach provides central control over committee members creation and we can easily manipulate with them. Central control has no impact on nodes decentralization. The central element is used for more comfortable simulations of various scenarios.

Algorand application extends the basic application. Application implements all three steps of Algorand process – block proposal, soft vote, certify vote. The application proposes new block, but not every application does it. Application, that proposes a new blocks, is based on the Algorand helper and its simulation of VRF.

The next two steps, soft and certify vote steps, are also based on the helper decisions. After the successful process of voting, all applications are able to add proposed blocks to blockchain. Algorand protocol contains some other phases, for example, recovery phase. These phases are not implemented in our testbed. Testbed contains only the main part of Algorand process.

7.5.2 Evaluation

In this chapter, we describe an evaluation of the proper Algorand implementation. For evaluation, we choose the main properties of the protocol. These properties are based on the Algorand paper:

- Algorand uses BFT* protocols
- Algorand is an asynchronous protocol
- In the Algorand it is almost impossible to create blockchain forks
- Algorand is composed of phases – block proposal, soft vote, certify vote
- Algorand uses VRF function

Asynchronization – Defined Algorand protocol is an asynchronous protocol. Each node of the network has not been in synchronization with other nodes.

To test this property we made two tests. The first test worked in a synchronous state and the second test worked asynchronously. Results should be the same, because the Algorand supports an asynchronization communication. After the run, we compare results of those two tests, already mentioned. The results were almost the same, there were only little differences, that had no direct influence on the global blockchain. Finally, the evaluation of asynchronous state was successful.

Phases and BFT* - Algorand protocol is composed of three main phases, that implement partially BFT protocol. Our implementation of the Algorand protocol is also made of those main phases. Phases are implemented separately to support the protocol asynchronization property. Each node starts phases individually by its clocks, and each node reacts individually to phases by received messages.

Forks - We tested different settings of protocol, different committee sizes, different network sizes, etc (chapter 8). Implemented protocol during tests had no tendency to create blockchain forks. So, this point of main Algorand properties was accomplished.

VRF - Algorand uses for committee selection the VRF algorithm. VRF function generates a secret without any communication among the users (nodes). VRF is running by node and by that the node knows if it will be a part of the committee. In our implementation, the VRF is replaced by a solution with a centralized approach. The principle is almost the same as the solution of FTS in other implemented protocols (for example - Ouroboros – chapter 7.4.2). For the selection of the committee, the implemented Algorand protocol uses centralized helper element. VRF function does not need any communication for its calculation. The helper does not need it as well. Helper and calculation of pseudo-VRF may be described by pattern Singleton. There exists only one helper instance that provides the calculation of VRF and it has control over the nodes calculations.

8 Tesbed results

This chapter contains the results of the testbeds. Subchapters are divided by logic. The first subchapter writes about testbed setting, used HW, and simulator basic properties for using. The second subchapter focuses on the simulations on the network part of the simulator. And the last subchapter describes simulations of Ouroboros and Algorand protocols. In each part, there is a discussion that discusses protocols‘ advantages based on the observation.

Results of simulation are often illustrated in the graphs. Graphs can be in some cases is hard to read in detail, so in the appendix, you can found results in tables.

8.1 Simulator description

For simulations, we primarily use a laptop with configuration: Intel Core i5-7200 2,5Ghz, 32GB RAM. This laptop was powerful enough for simulations that are described below. But for advanced simulation, we suppose to use more powerful HW, especially for the lower time of the simulation.

During the simulation, the simulator was able to simulate less than 1500 nodes effectively. More nodes led to too higher HW requirements. When we simulated the biggest network of all simulations, the simulator used about 10 GB of the RAM memory and one real second processed 1/78,2 of simulation‘s second.

We observed simulator behavior and requirements through all performed simulations, and here are properties of the simulator from the user’s point of view:

- One simulation is able to run on only **one CPU core** – one run simulation uses for its running only one CPU core. The simulator NS-3 does not support parallelization. It restricts using simulator a little, because parallelization can lead to faster simulations. But in a practical way, we can run more simulations separately at the same time and create parallelization manually.
- **RAM consumption** – simulations with bigger networks (1000+ nodes) need quite a lot of RAM memory, especially if a decentralized network is used in the simulation. During the testing, we found out, that for simulation of 1000 nodes the simulator needs approx 8-16GB. The number depends on the protocol setting.
- **Average real-time vs. simulation time** - We observed that the average proportion between real-time vs. simulation time is about 1/32.3 on the network with 512 nodes. It means - one simulated second takes 32.3 seconds in real-time.

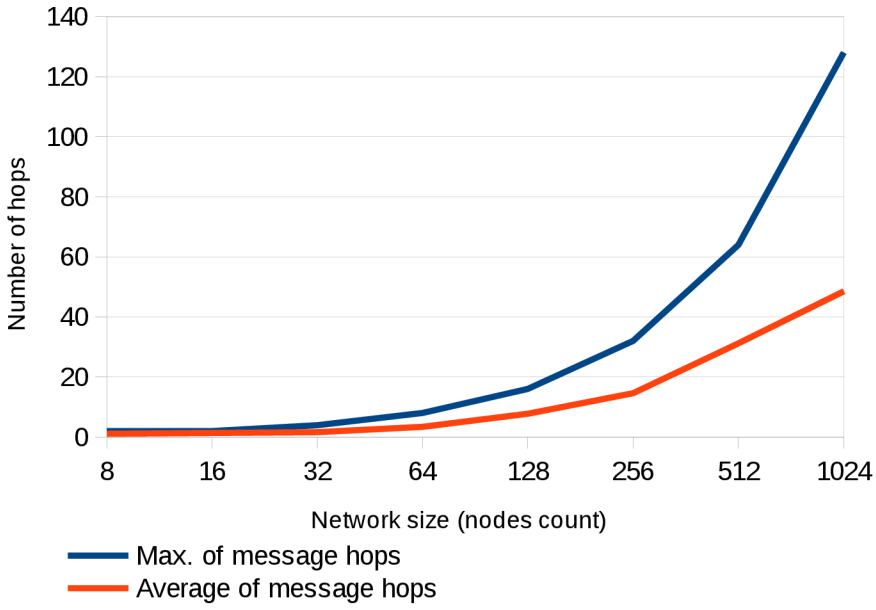
8.2 Network part simulations

Before simulations of specific proof-of-stake protocols, we had simulated behaviour of the network part of the simulator. In the simulations, we observed the behaviour of network, especially when we have changed network size or when we have selected some nodes and have turned them off – simulation of fail state.

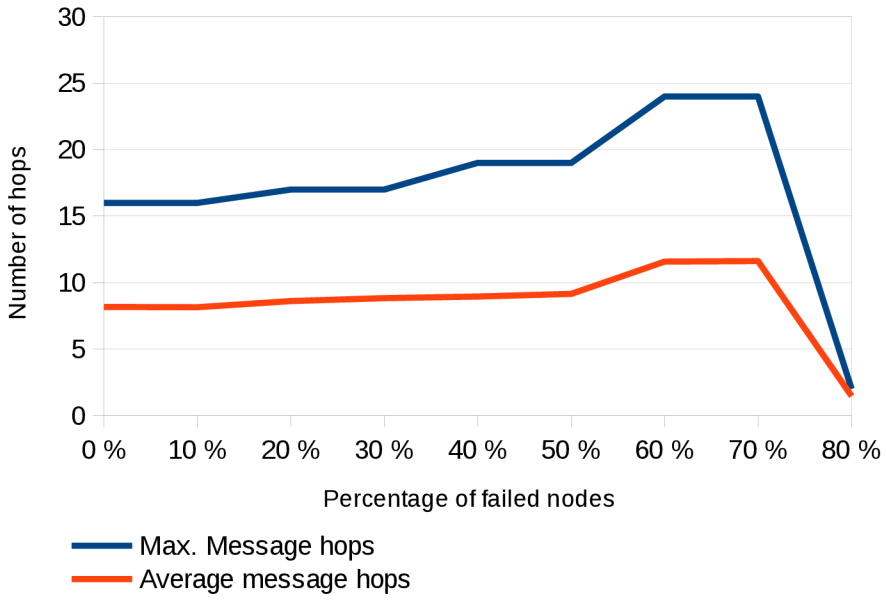
Observing of the network was based on the number - *number of hops*. This number is defined as a number of nodes that message goes through, until it reaches the end of its journey. This number was observed on the broadcast sending. In the simulation, we observed every message that goes through system and we noted maximal *number of hops* and the average *number of hops*. Then, we used that number in graphs.

Next two graphs show us behaviour of decentralized network:

- Graph 1 (Table 4) illustrates the relation between network size and *number of hops*. We can see that relation between network size and *number of hops* is linear.
- Graph 2 (Table 5) illustrates the relation between the number of failed nodes vs. *number of hops*. For simulation we use network with 128 nodes. We can see that the decentralized network is resistant against the failed nodes. The critical point is around 65% of nodes that failed. On the scale 0% - 50% failed nodes, the network does not indicate any influence of that behaviour. 50% - 70% indicate that the network is fragmented, but it is still operable. If in the network is more than 70% failed nodes, than the network is unusable because, the network is fragmented into a small parts, that are not connected to each other. That is a reason why the result show a very low number – *number of hops*.



Graph 1: Network size x hops number



Graph 2: Count of failed node x hops number

On the results of network layer simulations, we can claim that the network part has a low influence on the consensus layer. Mainly, because the distributed network is in the design part based on the low predisposition to the error states. So, the next simulations have not to consider any influence of the network layer.

8.3 Throughput comparison

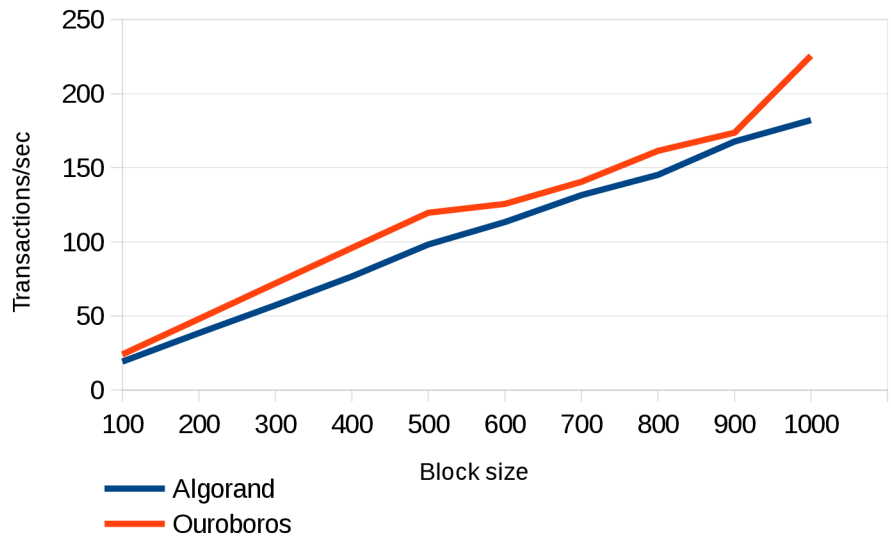
We ran simulations - their goal was to compare Ouroboros and Algorand protocol, by the perfomance (throughput) side. Simulations simulated basic processes in the network and we observed the throughput of individual protocols. In the network there were not any attacks, breaking assumptions, nodes fails, etc. Network consist of the 128 nodes. During the simulations we observed setting of protocols and its influence to the performance.

We found out, that performance of protocol is highly dependent on the protocol setting. We can explain it on the block size parameter, which has one of the most influential to the protocol‘s performance. As a result we created graph 3 (Table 6), that visualize this influence. We can see that the dependency between block size and performance is almost linear, especially in the lower numbers of the block size.

Almost same behaviour we observed when we changed any of protocols settings. For example, when we change slot size in the Ouroboros protocol, then it has the almost same influence on the performance results. When we set slot number lower, it leads to higher performance, because protocol, as the whole unit, generates blocks in higher frequency.

The result mentioned above may lead to the edge of the setting of individual protocols, where all parameters of the protocol setting will be set for higher performance. But, this way of protocol setting can probably cause attack issues and vulnerabilities. So, the implementators of protocol have to test the protocol and choose the right combination and balancing of the protocol setting.

It leads us to the recommendation for all proof-of-stake protocols. Implementators should invest in protocol setting testing. By optimization of protocol setting, they can significantly improve protocol throughput.



Graph 3: Algorand and Ouroboros performance (block size dependecy)

8.4 Scalability comparison

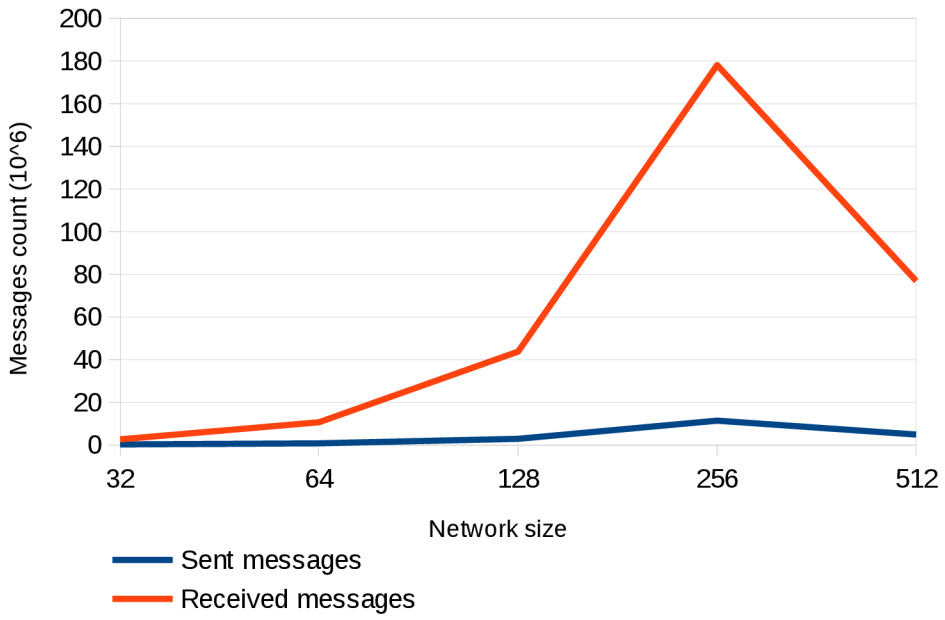
Simulation for comparison the scalability of the protocols, we observed by the change of throughput (performance) and change of the network load. We observed change of network performance (transactions/sec) and the changed count of messages, which go through the network when we increase the number on nodes in the network. Based on these results we made other ideas in the scalability issues. During simulation protocols were configured to classical setting (defined in its papers) and block size of 800 transactions.

The change of network load was observed on two metrics. The first metric is a count of messages that nodes send into the network. The second metric is a count of messages that all nodes received together. The number of received messages should be the same or higher than the number of sent messages. Especially, in this type of network, where the broadcast is used quite often. By observing two parameters we can indicate the trend of changing network load. On the trend, we can build next speculation and comparison of protocols.

In the graphs, we show results based on the network load. Graph 4 (Table 7) represents the results of the Algorand protocol and the second, Graph 5 (Table 8) represents results of the Ouroboros protocol.

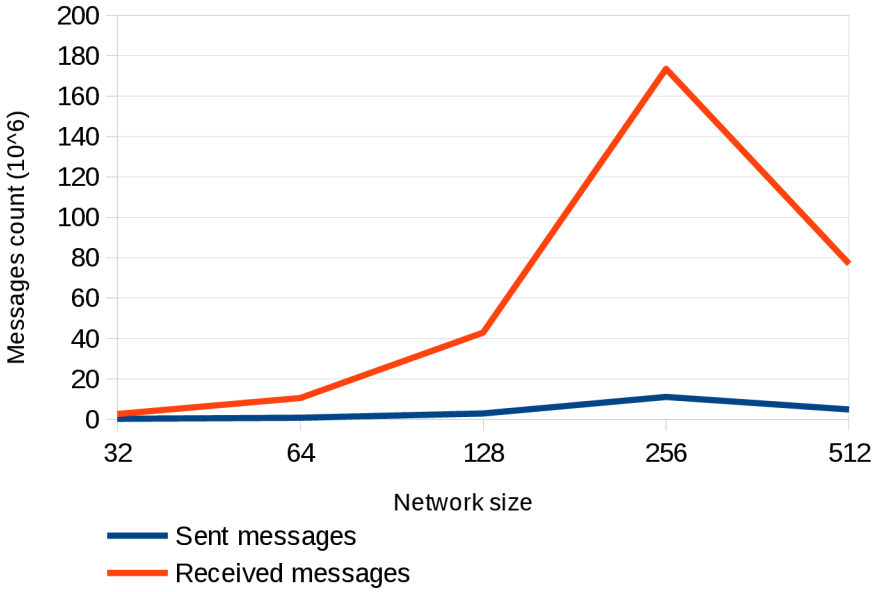
We can see almost the same trend in both protocols. Same trend with unexpected behaviour in the last measurement. This unexpected behaviour is based on a bigger change of transaction generation, because simulation was not able to simulate 512 nodes with top-level number of transaction generation. So, for our next conclusions, we will ignore the last measurrment because it is irrelevant.

At first, we can say that the Ouroboros and Algorand have almost the same behaviour in the scalability property. But by closer look, we can see that Algorand has more scalability issues than Ouroboros. That is because on the top result, Algorand generates about 5% more messages than Ouroboros and Algorand‘s nodes receive 5% more messages than Ouboros‘ nodes. We can tell that 5% is not a big number. But, we have to consider that 50% of network traffic is the generation of new transactions made by random nodes.



Graph 4: Algorand scalability

So, based on the network load, we concluded, that Algorand is a little bit worse in the scalability property than Ouroboros. We can assign that property to leaders group creation and voting in that group with compare to FTS protocol. But, that was only one comparison of scalability. Let‘s look at a comparison based on the performance.



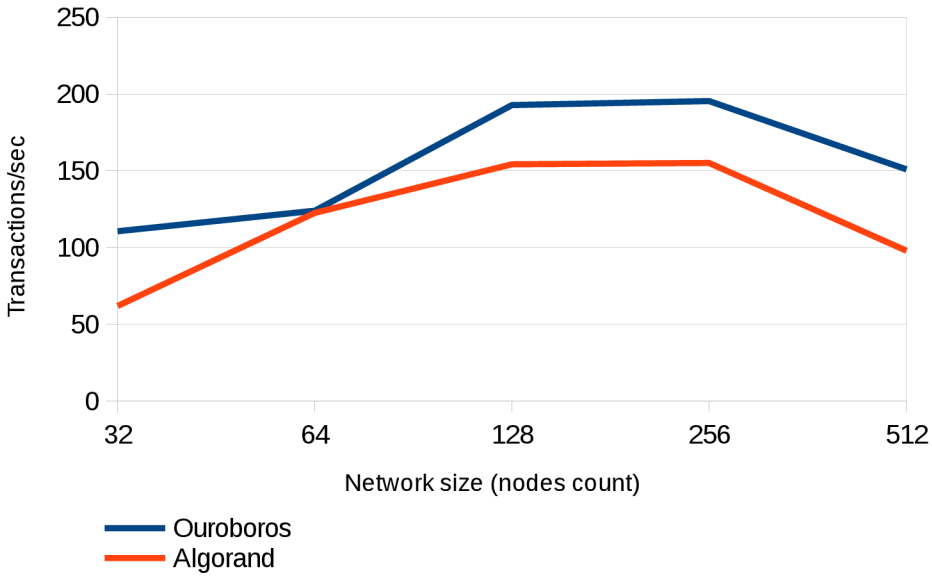
Graph 5: Ouroboros scalability

The next step is a comparison by throughput. We observe the throughput of protocols. This comparison has the same setting as the previous comparison. So, at the last measurement, we can see the unexpected result explained above.

Graph 6 (Table 9) illustrate the results of observing both protocols with vary network sizes. Again, we can see the almost same trend in each protocol result. Based on the results, we can divide results into two groups.

The first group consists of results that are identify by lower network size. We can see that the Algorand has better results and performance against the Ouroboros protocol. We attached this result to Algorand straight behaviour and straight goal - to create and to finalize blockchain block.

In the second group, there are results that show Ouroboros higher throughput in comparison to Algorand protocol. Algorand has lower throughput, because of creating and using the BFT committee. Throughput is lower by about 20%.



Graph 6: Protocols scalability

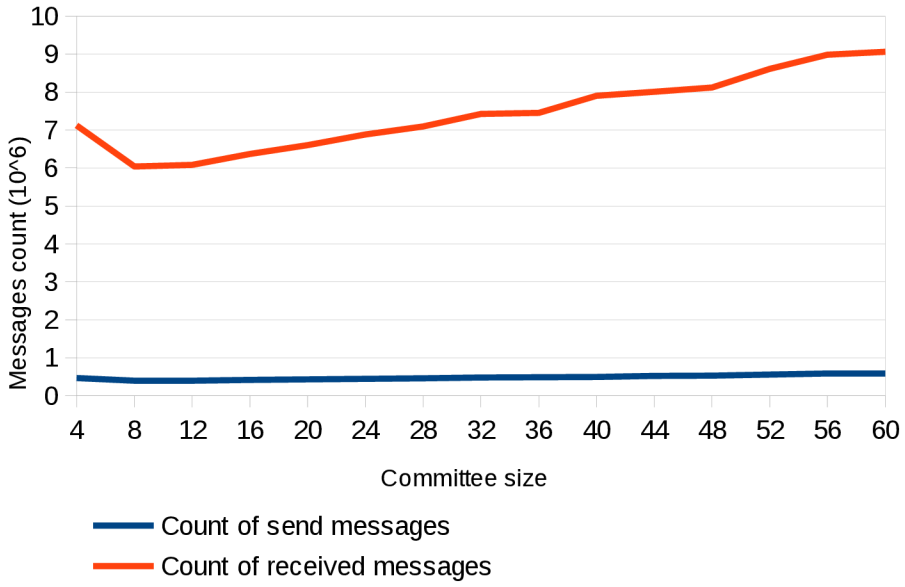
All things considered, in our testbed, the Ouroboros has better results for scalability issues. But, we have to mention, that the results were based on the basic network without any network complications, attacks, forks of blockchain, and others. So, via this result we can express, that the Ouroboros is better for pure network and its scalability. This is base on the simplest way of block voting. On the other hand, Ourobors has some problems in practical networks, the Algorand can be much better. It is caused by more by Algorand's more sophisticated way of block creating.

On that result based, we can create a recommendation or we propose improvement. We have to look at recommendations (improvement) from a scalability perspective. Proof-of-stake protocols that use the FTS algorithm for leader selection are more suitable for bigger networks and primarily for networks that generate forks exceptionally. On the other hand, protocols based on the BFT algorithm are more suitable for smaller networks with more potential forks.

8.5 Algorand specific tests

One of the important parameters of the Algorand protocol is committee size. The simulator enables to change committee size. The committee size in the simulator is generated by Poisson distribution and the config file of the simulator enables to set *Mean* value of it.

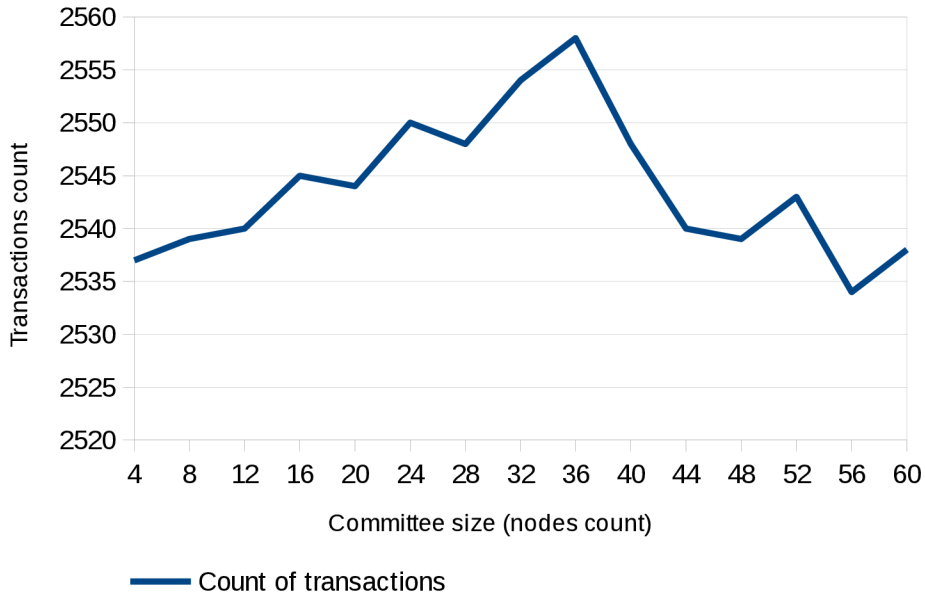
We did simulations of different committee size. Simulations were based on the network size consists of 128 nodes and blocks with 800 transactions. During simulations, we observed the impact of that changes to throughput and network traffic. The results of the observation were written into graphs 7 and 8 (Table 10).



Graph 7: Algorand: change of commitee size

The first result shows if we increase committee size then the network traffic increase. The increase of network communication can affect network throughput. The real impact of that to throughput is discussed in the next paragraphs. The best size of the committee, graph-based, is about number 10. The higher committee leads to higher network traffic, also very low committee size leads to higher network traffic and it can also lead to some vulnerabilities.

The second graph 8 shows the influence of committee size to the protocol‘s throughput. In the graph is the dependency of the count of transactions that go through the network during simulation and committee size. We can see that the better result is about 30 committee size. On the other side, results are almost the same, because results difference is low. Also, we have to think about the random generation of transactions. Summary, the committee size has no direct impact, or only very low, to protocol‘s throughput (we talk about non-extreme committee size). But committee size has a direct impact on other properties of the protocol, for example, security.



Graph 8: Algorand: change of commitee size

Finally, on the results, we can create a conclusion that the best committee size about 20 nodes. This number is based on our results. For future use of this committee size, it has to be verified by the next simulations – different network sizes, simulation of attacks, and others to prove this number.

8.6 Results conclusion

We observed proof-of-stake protocols behaviour. The behaviour of Algorand and Ouroboros protocols. During observing we focused on the performance, throughput, and scalability of the protocols. Here we discuss suggestions testbed‘s results-based to improve proof-of-stake protocols.

In some simulations and their results, we can see that the performance is highly dependent on the protocol setting. It leads to, that imlementators of the protocol should, go through lots of different settings of their protocol and they should try to find out the best setting. The setting can be and should be, different for different types of used protocol. It should be different for different of the used network layer, different lower layers, different network size. By that approach, implementators of protocol reach better performance without any restrictions of security or scalability. All that is based on the idea which says, all observed protocols have almost the same graph trend in the basic use.

Simulations that focuses on the scalability of simulated protocols get us some ideas to improve actual proof-of-stake protocols. By simulations and its results, we can assume that better scalability is reached by protocol with lower communication between nodes and algorithm FTS.

In this testbed, the representant is the Ouroboros protocol. Mainly, it is reached by lower network load and higher throughput of the network. Here we can suggest the idea that for better scalability is better to design and use protocol with lower communication load, especially which do not use broadcast communication.

In the Algorand protocol, we inspected committee size and its influence to protocols throughput. Tests with committee size were executed by the network with 128 nodes. During inspecting, we discovered that the best committee size is 20 nodes. This number was almost same for the network with double network size. On the smaller networks – on the testbed results, we can claim, that the best committee size in the Algorand is about 20 nodes.

Summary of all results can lead to that the Ouroboros protocol is better than the Algorand protocol. But, in real application, it may or may not be true. The testbed focuses mainly on simulations that do not work with any type of attack or breaking of assumptions of the protocol. Maybe, on the „attacker“ type of simulation, the Algorand will be much better than the Ouroboros protocol. The reasons are mentioned in chapter 5. Briefly, Algorand focus to high finality and blockchain without forks. So, as summary of testbed‘s results, we focused on the part of proof-of-stake protocols behaviour. We came with some suggestions to improvement of actual protocols, but for design a new „better“ protocol, that will be based on the result of this testbed, it needs to do more broad simulations than was done in this thesis.

9 Conclusion and future directions

This diploma thesis writes about proof-of-stake type of consensus in the blockchain technology. This thesis describes proof-of-stake protocols and compares them. For comparison, the thesis represents the creation process of the testbed that is followed by observing the testbed‘s results. Stand on the results, the thesis suggests improvements of proof-of-stake protocols.

At the beginning of the thesis, we introduced the main types of consensus, containing proof-of-stake consensus as well. Every protocol‘s characterization of the consensus approach described the main purpose of usage, also, pros and cons of this approach. Chapter 3 introduced the existing testbeds. It described, what the testbed is, also, it wrote about existing testbeds and simulations in the consensus of blockchain, especially in the proof-of-work and proof-of-stake approach.

Chapter 5 of the thesis wrote about the specific implementation of the proof-of-stake consensus approach. As main specific proof-of-stake protocols were chosen Algorand, Ouroboros, Ouroboros Praos, Casper FFG, Casper TFG, and Tezos. Our choice was based on the different approaches to the proof-of-stake consensus. We introduced the main properties of these proof-of-stake protocols in this part, and by them, we compared these protocols. As observed properties we added a description of possible attacks, that was used also for the comparison of the protocols. These protocols were described precisely and compared. As the result of the chapter we created tables that visualized the results of comparison theoretically based.

In the third section (chapters 6 and 7), we designed a testbed for proof-of-stake protocols comparison, based on the theory, which was introduced in the previous part. We presented our idea of testbed creation, that during designing gets exact properties. The section contains also a description of the testbed implementation. As an output of the section, we designed and implemented the testbed. There are described some details of solutions that were used for testbed implementation. After the implemented testbed part comes the fourth part of the thesis, which is testing and observing testbed results.

Chapter 8 of the thesis wrote about testbed results. During the testing, we observed and compared two proof-of-stake protocols, Algorand and Ouroboros. Casper protocol was implemented partially. Unfortunately, the Casper implementation was above our planned programming capacity. Casper and Algorand use BFT algorithm, and the goal of the thesis is to compare substantially different protocols. Thus, this decision did not significantly affect results of the work. The chapter showed results pictured mainly in graphs. All results are described. The type of used configuration is explained and, last, but not least, how results reflected in practical life. By the results, we are able to suggest some behaviour and possible improvements in actual proof-of-stake protocols.

The goal of work was to implement testbed that compares a few existing proof-of-stake consensus protocols and, based on that, to be able to suggest improvements of these protocols. Despite some implementation problems, such as complication of proof-of-stake protocol implementations, we theoretically compared 3 protocols, however, we implemented 2 protocols

fully. The thesis and testbed laid the groundwork for proof-of-stake protocols comparison. It compared the main proof-of-stake approaches between each other by its papers and definitions. By designing and implementation, the thesis and implemented simulator showed the way how proof-of-stake protocols can be practically evaluated and compared. By the combination of theoretical research and testbed simulations, the thesis suggested improvements of actual proof-of-stake protocols.

The simulator showed a good groundwork for the proof-of-stake simulations. Implementators can verify their conclusions. So, in the future, we would like to extend the number of implemented proof-of-stake protocols in the testbed. And we want to extend the number of implemented types of simulations. By these two points, we can suggest the next new improvements or we can design a new proof-of-stake protocol, that will be based on the combination of the best-observed properties from all protocols.

Bibliography

[1] Christian Cachin and Marko Vukolić. *Blockchain Consensus Protocols in the Wild* [online]. Zürich, Switzerland, 2017 [cit. 2020-20-7] Available at: https://drops.dagstuhl.de/opus/volltexte/2017/8016/pdf/LIPIcs-DISC-2017-1.pdf

[2] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System* [online]. November 2008, p. 8, [cit. 2020-20-7] Available at: https://bitcoin.org/bitcoin.pdf

[3] Cynthia Dwork and Moni Naor. *Pricing via Processing or Combatting Junk Mail* [online]. 1993 [cit. 2020-20-7] Available at: http://www.wisdom.weizmann.ac.il/~naor/PAPERS/pvp.pdf

[4] Arthur Gervais and Ghassan Karame and Srdjan Cupkun. *On the Security and Performance of Proof of Work Blockchains* [online]. October 2016, p. 3 [cit. 2020-20-7] Available at: https://www.researchgate.net/publication/309451429_On_the_Security_and_Performance_of_Proof_of_Work_Blockchains

[5] danielefavi. *blockchain-hahsblock-calculator* [online]. November 2017, [cit. 2020-20-7] Available at: https://github.com/danielefavi/blockchain-hash-block-calculator/blob/master/hash.php

[6] Ittay Eyal and Emin Gun Sirer. *Majority is not Enough: Bitcoin Mining is Vulnerable* [online]. November 2013 [cit. 2020-20-7] Available at: https://arxiv.org/pdf/1311.0243.pdf

[7] Karl J. O'Dwyer. *Bitcoin mining and its energy footprint* [online]. Limerick, Ireland, June 2014 [cit. 2020-20-7] Available at: https://ieeexplore.ieee.org/document/6912770/authors#authors

[8] Jake Frankendield. *Proof of Stake (PoS)* [online]. August 2019 [cit. 2020-20-7] Available at: https://www.investopedia.com/terms/p/proof-stake-pos.asp

[9] QuantumMechanic. *Proof of stake instead of proof of work* [online]. July 2011 [cit. 2020-20-7] Available at: https://bitcointalk.org/index.php?topic=27787.0

[10] *Okcash* [online]. 2020 [cit. 2020-20-7] Available at: https://okcash.org/

[11] *NAVCOIN* [online]. 2020 [cit. 2020-20-7] Available at: https://navcoin.org/en

[12] *NEO* [online]. 2020 [cit. 2020-20-7] Available at: https://docs.neo.org/docs/en-us/index.html

[13] *ARK* [online]. 2020 [cit. 2020-20-7] Available at: https://ark.io/

[14] *LISK* [online]. 2020 [cit. 2020-20-7] Available at: https://lisk.io/

[15] *CARDANO* [online]. 2020 [cit. 2020-20-7] Available at: https://www.cardano.org/

[16] *EthHub* [online]. 2020 [cit. 2020-20-7] Available at:https://docs.ethhub.io/

[17] Will Kenton. *Proof of burn (Cryptocurrency)* [online]. February 2020 [cit. 2020-20-7] Available at: https://www.investopedia.com/terms/p/proof-burn-cryptocurrency.asp

[18] Kostis Karantias and Aggelos Kiayias and Dionysis Zindros. *Proof-of-Burn* [online]. 2019 [cit. 2020-20-7] Available at: https://eprint.iacr.org/2019/1096.pdf

[19] *Slimcoin* [online]. 2020 [cit. 2020-20-7] Available at: https://slimco.in/

[20] *Burstcoin* [online]. 2020 [cit. 2020-20-7] Available at: https://www.burst-coin.org/

[21] Homoliak I. and Venugopalan S. and Hum Q. and Szalachowski. *The Security Reference Architecture for Blockchains: Towards a Standardized Model for Studying Vulnerabilities, Threats, and Defenses* [online]. Singapore, October 2019 [cit. 2020-20-7] Available at: https://www.researchgate.net/publication/336734516_The_Security_Reference_Architecture_for_Blockchains_Towards_a_Standardized_Model_for_Studying_Vulnerabilities_Threats_and_Defenses

[22] *Algorand* [online]. 2020 [cit. 2020-20-7] Available at: https://www.algorand.com/

[23] Jehyuk Jang and Heung-No Lee. *Profitable Double-Spending Attacks* [online] [cit. 2020-20-7] Available at: https://arxiv.org/pdf/1903.01711.pdf

[24] Ayelet Sapirshreing and Yonatan Sompolinsky. *Optimal Selfish Mining Strategies in Bitcoin* [paper]. May 2017 [cit. 2020-20-7] Available at: https://link.springer.com/chapter/10.1007/978-3-662-54970-4_30

[25] Brown-Cohen J. and Naryanan A. and Prosomas Ch. And Weinberg S. *Formal Barriers to Longest-Chain Proof-of-Stake Protocols* [online]. September 2018 [cit. 2020-20-7] Available at: https://arxiv.org/pdf/1809.06528.pdf

[26] Alexander Chepurnoy. *Interactive Proof-of-stake* [online]. January 2016, p. 4 [cit. 2020-20-7] Available at: https://arxiv.org/pdf/1601.00275.pdf

[27] *"Fake Stake" attacks on chain-based Proof-of-Stake cryptocurrencies* [online]. January 2019 [cit. 2020-20-7] Available at: https://medium.com/@dsl_uiuc/fake-stake-attacks-on-chain-based-proof-of-stake-cryptocurrencies-b8b05723f806

[28] *Bitcoin Simulator* [online]. 2020 [cit. 2020-20-7] Available at: http://arthurgervais.github.io/Bitcoin-Simulator/index.html

[29] *NS-3 network-simulator* [online]. 2020 [cit. 2020-20-7] Available at: https://www.nsnam.org/

[30] Oguzhan Alagoz and Heather Hsu and Andrew J. Schaeferand Mark S. Roberts. *Markov Decision Processes: A Tool for Sequential Decision Making under Uncertainty* [online]. [cit. 2020-20-7] Available at: http://www.pitt.edu/~schaefer/papers/MDPTutorial.pdf

[31] Ivan Homoliak. *StrongChain Demo* [online]. May 2019 [cit. 2020-20-7] Available at: https://github.com/ivan-homoliak-sutd/strongchain-demo

[32] Joachim Zahnentferner. *Chimeric Ledgers: Translating and Unifying UTXO-based and Account-based Cryptocurrencies* [online]. Hong Kong, 2018 [cit. 2020-20-7] Available at: https://eprint.iacr.org/2018/262.pdf

[33] Aoki Y. and Banno R. and Kaneko T. and others. *Simblock* [online]. 2019 [cit. 2020-20-7] Available at: https://dsg-titech.github.io/simblock/

[34] Aoki Y. and Banno R. and Kaneko T. and others. *SimBlock: A Blockchain Network Simulator* [online]. March 2019 [cit. 2020-20-7] Available at: https://arxiv.org/pdf/1901.09777.pdf

[35] Garvais A. and Karame O. and Wust K. and others. *On the Security and Performance of Proof of Work Blockchains* [online]. Switzeland, 2016 [cit. 2020-20-7] Available at: https://eprint.iacr.org/2016/555.pdf

[36] Zhang Ren and Preneel Bart. *Lay Down the Common Metrics: Evaluating Proof-of-Work Consensus Protocols' Security* [online]. Leuven [cit. 2020-20-7] Available at: https://www.esat.kuleuven.be/cosic/publications/article-3005.pdf

[37] *Evaluating PoW Consensus Protocols's Security* [online]. 2019 [cit. 2020-20-7] Available at: https://github.com/nirenzang/PoWSecurity

[38] Vitalik Buterin. *On Stake* [online]. July 2014 [cit. 2020-20-7] Available at: https://blog.ethereum.org/2014/07/05/stake/

[39] Vitalik Buterin and Griffith Virgil. *Casper the Friendly Finality Gadget* [online]. October 2017 [cit. 2020-20-7] Available at: https://arxiv.org/pdf/1710.09437.pdf

[40] Vlad Zamfir. *Casper the Friendly Ghost* [online]. December 2017 [cit. 2020-20-7] Available at: https://github.com/ethereum/research/blob/master/papers/CasperTFG/CasperTFG.pdf

[41] Chen Jing and Micali Silvio. *ALGORAND* [online]. May 2017 [cit. 2020-20-7] Available at: https://arxiv.org/pdf/1607.01341.pdf

[42] Micali Silvio and Rabin Michael and Vadhan Salil. *Verifiable Random Functions* [online]. [cit. 2020-20-7] Available at: https://people.csail.mit.edu/silvio/Selected%20Scientific%20Papers/Pseudo%20Randomness/Verifiable_Random_Functions.pdf

[43] Yongge Wang. *Another Look at ALGORAND* [online] .February 2020 [cit. 2020-20-7] Available at: https://arxiv.org/pdf/1905.04463.pdf

[44] Yossi Gilad and Rotem Hemo and Silvio Micali and others. *Algorand: Scaling Byzantine Agreements for Cryptocurrencies* [online]. [cit. 2020-20-7] Available at: https://people.csail.mit.edu/nickolai/papers/gilad-algorand-eprint.pdf

[45] Aggelos Kiayias and Alexander Russell and and others. *Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol* [online]. July 2019 [cit. 2020-20-7] Available at: https://eprint.iacr.org/2016/889.pdf

[46] Markus Stadler. *Publicly Verifiable Secret Sharing* [online]. March 2000 [cit. 2020-20-7] Available at: https://www.researchgate.net/publication/2456942_Publicly_Verifiable_Secret_Sharing

[47] Bernardo David and Peter Gazi and others. *Ouroboros Praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain* [online]. November 2017 [cit. 2020-20-7] Available at: https://eprint.iacr.org/2017/573.pdf

[48] Victor Allombert and Mathias Bourgoin and Julien Tesson. I*ntroduction to the Tezos Blockchain* [online]. September 2019 [cit. 2020-20-7] Available at: https://arxiv.org/pdf/1909.08458.pdf

[49] Shehar Bano and Alberto Sonnino and others. *SoK: Consensus in the Age of Blockchains* [online]. November 2017 [cit. 2020-20-7] Available at: https://arxiv.org/pdf/1711.03936.pdf

[50] *Mininet* [online]. 2018 [cit. 2020-20-7] Available at: http://mininet.org/

[51] *C++* [online]. 2020 [cit. 2020-20-7] Available at: https://isocpp.org/

[52] *RapidJSON* [online]. 2015 [cit. 2020-20-7] Available at: https://rapidjson.org/

Appendix A

Contents of the included storage media

- This thesis - *dp.pdf*
- Source code[14] – directory *src*
- Demonstration video with subtitles – *demo_algorand.mp4* + *demo_algorand.srt*

14 Available at: https://github.com/honza66/proof-of-stake-testbed

Appendix B

Results tables

Network size (nodes count)	Max. - number of hops	Average – number of hops
8	2	1,15
16	2	1,35
32	4	1,68
64	8	3,42
128	16	7,76
256	32	14,6
512	64	31,25
1024	128	48,5

Table 4: Network size x hops number

Percentage of failed nodes	Max. Message hops	Average message hops
0 %	16	8,17
10 %	16	8,15
20 %	17	8,6
30 %	17	8,82
40 %	19	8,94
50 %	19	9,15
60 %	24	11,57
70 %	24	11,62
80 %	2	1,46

Table 5: Count of failed nodex x hops number

Block size (trans. count)	Algorand (trans/sec)	Ouroboros (trans/sec)
100	19,27	24,1
200	38,55	48,1
300	57,33	71,98
400	76,69	95,89
500	98,3	119,65
600	113,38	125,6
700	131,52	140,56
800	145,21	161,53
900	167,8	173,62
1000	182,12	225,6

Table 6: Algorand and Ouroboros performance (block size dependecy)

Network size (nodes count)	Sent messages count	Received messages count
32	183197	2518812
64	712294	10595320
128	2830499	43694208
256	11333823	178149425
512	4854145	768849425

Table 7: Algorand scalability

Network size (nodes count)	Sent messages count	Received messages count
32	180313	2478983
64	700110	10413281
128	2775769	42849968
256	11037961	173499911
512	4847482	76876649

Table 8: Ouroboros scalability

Network size (nodes count)	Ouroboros	Algorand
32	110,5	61,89
64	123,86	122,63
128	192,77	154,21
256	195,37	155,1
512	150,81	97,81

Table 9: Protocols scalability

Committee size	Sent messages count	Received messages count	Transactions count
4	461158	7118977	2537
8	391075	6037088	2539
12	393890	6080579	2540
16	412413	6366591	2545
20	427545	6599971	2544
24	446036	6885431	2550
28	459671	7096083	2548
32	480777	7421841	2554
36	482996	7456032	2558
40	511713	7899301	2548
44	518720	8007622	2540
48	525958	8119121	2539
52	558104	8615451	2543
56	582226	8987726	2534
60	586988	9061458	2538

Table 10: Algorand committee size