Одеський національний університет імені І. І. Мечникова
Факультет математики, фізики та інформаційних технологій
Кафедра оптимального керування та економічної кібернетики

Кваліфікаційна робота

на здобуття ступеня вищої освіти «магістр»

«Імітаційне моделювання та оптимізація ієрархічного консенсусу в блокчейнах контрольованого складу»

«Simulation Modeling and Optimization of the Hierarchical Consensus in Permissioned Blockchains»

Виконала: здобувачка денної форми навчання
спеціальності 113 Прикладна математика
Освітня програма «Прикладна математика»
Ворохта Аліса Юріївна

Керівник: канд. фіз.-мат. наук, доц. Страхов Є. М. _____
Рецензент: канд. техн. наук, доц. Мазурок І.Є.

Рекомендовано до захисту:

Протокол засідання кафедри

№ ___ від _____ 2022 р.

Завідувач кафедри

_____ _____

Захищено на засіданні ЕК № _____

Протокол № ___ від _____ 2022 р.

Оцінка _____ / _____ / _____

Голова ЕК

_____ _____

Одеса — 2022 р.

Odesa I. I. Mechnikov National University
Faculty of Mathematics, Physics and Information Technology
Department of optimal control and economical cybernetics

Diploma thesis

master

«Simulation Modeling and Optimization of the Hierarchical Consensus in Permissioned Blockchains»

Fulfilled by: full-time student
specialty 113 Applied Mathematics
Alisa Vorokhta

Supervisor: Ph. D. in physics and mathematics sciences, Associate Professor Yevhen Strakhov
Reviewer: Ph. D. in technical sciences, Associate Professor Igor Mazurok

Odesa — 2022

CONTENTS

ВСТУП

Наразі до технології блокчейн [1] виникає інтерес у представників найрізноманітніших сфер: від банковського сектору до земельного реєстру. Деякі компанії навіть базують свої проекти на технології блокчейн. Блокчейн – це децентралізована база даних, у якій інформація зберігається у вигляді «ланцюжка блоків» певної кількості транзакцій. Децентралізація [2] означає відсутність вузлів або груп з винятковим доступом до певних ресурсів.

Серед особливостей блокчейну можна також зазначити, що дані зберігаються в учасників мережі. Оскільки блокчейн децентралізований, і дані в ньому не можуть бути змінені або скасовані завдяки системі криптографічного захисту, ця технологія вважається дуже безпечною.

Технології блокчейн можна знайти застосування практично у всіх сферах діяльності. У [3] досліджується застосування технології блокчейн у транспортній логістиці. А у [4] було досліджено особливості застосування блокчейн-технології у цифровій економіці.

Одним із засобів дослідження є імітаційне моделювання. Воно допомагає досліджувати реальну фізичну систему, не проводячи з нею безпосередніх експериментів. Наприклад, у [5] було побудовано моделі та імітації системи освіти на основі блокчейну, а [6] досліджує методи моделювання систем блокчейн за допомогою Anylogic.

Оптимізація - це процес пошуку розв'язку, який найкраще задовольняє заданому критерію. [7] показує приклад застосування оптимізації у блокчейну для вибору консенсусу. У статті [8] розглядаються результати застосування оптимізації в блокчейні та запропоновані більш розширені завдання оптимізації для майбутньої роботи.

Мета дослідження: побудування та оптимізація моделей блокчейну контрольованого складу для дослідження її особливостей.

Об'єкт дослідження: блокчейн контрольованого складу з ієрархічним консенсусом.

Предмет дослідження: блокчейн контрольованого складу.

Метод дослідження: аналітичний аналіз та експерименти з моделлю у Python.

INTRODUCTION

Currently, representatives of various fields are interested in blockchain technology [1]: from the banking sector to the land registry. Some companies even base their projects on blockchain technology. Blockchain is a decentralized database in which information is stored in the form of a "chain of blocks"of a certain number of transactions. Decentralization [2] refers to the absence of nodes or groups with exclusive access to certain resources.

Among the features of the blockchain, it can also be noted that the data is stored by the network participants. Since the blockchain is decentralized, and the data in it cannot be changed or reversed due to the cryptographic protection system, this technology is considered very secure.

Blockchain technology can be used in almost all fields of activity. [3] explores the application of blockchain technology in transport logistics. And in [4] the peculiarities of the application of blockchain technology in the digital economy have been studied.

One of the research tools is simulation modeling. It helps to study a real physical system without conducting direct experiments with it. For example, in [5] were built models and simulations of a blockchain-based education system, and [6] explores methods for modeling blockchain systems using Anylogic.

Optimization is the process of finding a solution that best satisfies a given criterion. [7] shows an example of applying optimization to the blockchain for consensus selection. The article [8] discusses the results of the application of optimization in the blockchain and proposes more advanced optimization problems for future work.

The purpose: to build and optimize models of a permissioned blockchain to study its features.

Object of study: permissioned blockchain with hierarchical consensus.

Subject of study: permissioned blockchain.

Research method: analytical analysis and experiments with the model in Python.

CHAPTER 1

IMITATION MODELING AND OPTIMIZATION

1.1. What is Simulation Modeling?

Modeling is the process of creating a model. A model is a representation of how a certain system is built and runs. The model is a simplified version of the system it depicts. Choosing which real-world components to include is crucial before modeling the system. A model is the end product of this process, which is referred to as abstraction.

Simulation modeling [9] is the process of developing and evaluating a digital prototype of a physical model to predict how it will behave in reality. A model like this can be "played" in time. In this instance, the processes' randomness will determine the outcomes. These data allow for the creation of stable statistics. Imitation refers to experimenting using a model.

The process in which a simple model is built first, and then improved as important features are found, is called iterative modeling [10]. There are numerous models that may be developed for any physical system based on different assumptions and simplifications.

A simulation's output may be a design, a prediction about the system's potential actions or the impact of modifications on the system, or an explanation of its behavior. Predictions and test designs can be validated by taking measurements in the real world and comparing data with analysis and simulation results.

Any physical system can be represented by a variety of models, each with a wide range and level of detail. The aim of the modeling process is to find the model best suited to its purpose.

The applications of simulation in business are numerous, and they are frequently used when it is inconvenient to conduct trials on real systems, e.g. due to cost or time constraints. By providing precise perceptions, simulation modeling offers beneficial solutions across sectors [11].

1.2. Optimization

The main goal of the optimization problem [12] is to choose the element from the allowed set that meets a certain criterion as best. To formulate the optimization problem, an optimality criterion (the number of messages sent, the load on the network, etc.); varying parameters (for example, the number of network participants), the change of which allows you to influence the efficiency of the process; mathematical model of the process and limitations (hardware capabilities) are needed.

1.3. Python Programming Language

Python [13] is a high-level and general-purpose programming language. Its main goals are to increase developer efficiency, code readability, and quality, and ensure application portability. The language provides convenient high-level data structures. It supports a variety of programming paradigms, such as structured, object-oriented, imperative, and aspect-oriented. This makes Python a convenient programming language for modeling and optimization problems.

CHAPTER 2

FUNDAMENTALS OF DISTRIBUTED SYSTEMS AND BLOCKCHAIN

2.1. Distributed Systems

Distributed systems [14] are characterized by the dispersal of functions and resources between numerous components (referred to as "nodes") and the absence of a single control center, so the failure of one of the nodes does not lead to a complete stop of the entire system. The elements cooperate with one another to accomplish a common objective.

Client-server, three-tier, n-tier, or peer-to-peer architectures are the most common types of distributed programming.

- Client-server: architectures in which data is requested from the server by smart clients, who subsequently format and present it to users.
- Three-tier: architectures that enable the usage of stateless clients by moving client intelligence to a middle layer.
- N-tier: architectures that are frequently used to describe web apps that send requests to other enterprise services.
- Peer-to-peer: architectures that are identified by a distributed architecture, in which there is not a single administration server to supervise the entire system. In these networks, client and server functionality are combined, giving each participant the same capabilities. P2p networks' key benefit is that even a very high number of failing nodes will not cause the system to crash.

A distributed system must have the following properties: scalability, performance, and high availability [15]. Availability means that any request to a distributed system will result in a valid response, however, there is no assurance that each node's response will be identical. Scalability means that the system must have mechanisms for reacting to changes in the transaction flow within extremely broad bounds.

The biggest drawback of distributed systems is complexity [16]. With the increase in the number of machines, messages, and data being passed, the following problems arise:

- Data consistency. It can be difficult to synchronize changes to data and application states in a distributed network.
- Network and communication failure. It is possible for messages to be sent to the wrong nodes or in the wrong order, which degrades functioning and communication.

A distributed ledger (DLT) is a database that is distributed across multiple network nodes or computing devices.

The circumstance where various nodes assume different versions of adding data to the ledger occurs in the majority of systems. The issue with weak form finalization is the probabilistic or deterministic construction of a common prefix that expands with time for all ledger versions in honest nodes. The challenge in the strong form (fast finalization) is to design a protocol that fixes the order and content of new data in an unambiguous and irrefutable manner before it is completely dispersed over the network.

Blockchain is a prime example of one of the DLT data structure types.

2.2. Blockchain and Consensus

A blockchain is a growing sequence of transaction blocks that contain the cryptographic hash of the preceding block linked together using cryptography [17]. Blockchains are resistant to data modification. It happens due to the fact that once recorded, the data in any given block cannot be changed retrospectively without changing all the following blocks. Side chains are removed during finalization, and every block is arranged in a linear way.

There are three types of blockchains: permissionless, permissioned, or both [18]. Networks powered by permissionless blockchains are open to all users. They do not limit the rights of network nodes. A permissioned blockchain allows only specific companies or people with verified identities and the necessary clearance to manage the way data is processed there. For

this type of blockchain, there are several levels of data access, for example, such as:

- reading transactions from the blockchain, but with certain restrictions;
- proposal of transactions for inclusion in the blockchain;
- creation of new blocks.

Achieving overall system stability while dealing with a number of faulty processes is a fundamental challenge in distributed computing and multi-agent systems. These processes are described as consensus [19].

Byzantine fault tolerance (BFT)[20] is a property of a system that can withstand a class of failures that occur due to the "Byzantine Generals Problem". It was developed to represent a scenario in which a concerted strategy must be agreed upon by all system participants in order to prevent catastrophic system failure, yet some of these actors are unreliable. One of the most important properties of BFT is that if a system is made up of $3 \cdot f + 1$ nodes, f is the maximum number of faulty nodes that it can handle.

A Practical Byzantine Fault Tolerance (PBFT) [21] algorithm was developed, which ensures network security as long as the proportion of faulty nodes stays below the required level.

Different consensus mechanism algorithms operate according to various concepts.

The most well-known cryptocurrency networks use the Proof-of-Work (POW) consensus mechanism. This algorithm requires from the nodes participating in the network operation the result of a rather long work. This result can be easily verified. However, this mechanism requires high energy consumption and quite a long processing time.

The Proof-of-Stake (POS) is another common consensus algorithm. Compared to POW, it is less resource intensive. In addition, this mechanism requires less cost. In this algorithm, the probability of the formation of the next block in the blockchain by the participant is proportional to the share that the virtual currency tokens belonging to this participant make up from their total number. The disadvantage of this is that it encourages saving cryptocurrency rather than spending it.

Similarly, there are other consensus algorithms like Proof-of-Capacity (POC), Proof-of-Activity (POA), Proof-of-History (POH), etc.

CHAPTER 3

MODELING AND ANALYZING A PERMISSIONED BLOCKCHAIN WITH HIERARCHICAL CONSENSUS

3.1. Distributed Ledger Description

This section describes the work of DLT [22].

3.1.1. Terms

- A Node is a server that has been registered in the network and stores all the relevant data as Ledgers.
- Workers are logical structural components that are deployed on each Node. They have the necessary credentials required to participate in the POS protocol.
- Validator and Coordinator are independent elements of each Worker.
- Slot is the structural unit of the timeline of the network. Actions are taken into account simultaneously throughout a slot, and selected Validators in each subnet of each shard are required to produce and distribute their block.
- Epoch is a combination of slots. Epochs are used to summarize the network's intermediate results.
- Era is a combination of epochs. In the era the democratically coordinated changes to the values of the network setup take place.
- Coordinating and Shard networks are two functional subsystems that are implemented for the system's operation.
- BlockDAG is a directed cyclic graph (DAG) that has blocks as its vertices and references to preceding blocks as its edges.
- A block is called a spine block if it precedes all other blocks in the same slot.

3.1.2. Description

The Shard network is based on the DAG protocol. Block finalization, DAG linearization, and the selection of Validators that create blocks within a specific time slot are managed by the blockchain-based Coordinating network. As a result of the networks working together, the created blocks of transactions line up in a DAG, part of which is topologically sorted and finalized (Stream). The second part, consisting of not yet ordered blocks, forms Spray.

The protocol is based on the fast finality POS consensus. For more information about this DLT, see [22].

3.2. Hierarchical Consensus

The operation of the Coordinating network begins with the creation of a genesis block, which contains the initial list of Validators.

At the beginning of each epoch, a list of Committees for the next epoch is randomly generated for each slot.

In turn, at the beginning of each slot, the leaders of those Committees are chosen at random from among the Members of the Committees, and nodes are randomly chosen from the Committee Members to create blocks. In addition, each node of the Coordinating network sends to the connected nodes of the BlockDAG network information about the epoch, the list of block creators, and the finalized blocks. Moreover, each node of the Coordinating network receives information about spine blocks from the BlockDAG network node connected to it.

Algorithm 3.1 (Hierarchical Consensus [23])**.** Assume that the finalized spine block of the n-th slot f_n is already selected. By definition, f_0 is the genesis block.

1) Each Committee Member sends messages to the other Members with a list of visible spine blocks starting at $n+1$ slots, a block of the previous state, and a signature.
2) As a result of such a mailing, two options are possible:

a) There is a sequence of blocks that received more than $\frac{2}{3}$ votes, such that no longer sequence received more than $\frac{2}{3}$ votes.
b) No sequence received more than $\frac{2}{3}$ of votes. It is considered that a sequence of blocks receives a vote if it is the beginning of some of the block sequences which are parts of messages from the previous step. In this case, it is supposed that the resulting sequence of blocks is empty.

3) Each Committee Member that supports the resulting sequence of blocks sends a message to the other Members with this sequence, a block of the previous state, and a signature.
4) The leader of the Committee forms a message with the resulting sequence of blocks, a block of the previous state, and signatures of the Committee Members from the previous step and sends it to the other leaders of the Committees.
5) If more than $\frac{2}{3}$ of the Committees receive a nonempty sequence, each leader sends a message to the other leaders that contain messages from the previous step from other Committees and its signature.
6) The first leader forms and sends out the block that contains messages from the previous step, signatures of leaders, and its signature. Otherwise, the block is not published.

The Coordinating network's block acceptance rules:

1) A correctly produced Coordinating network block is passed on and added to the chain if it is obtained no later than the following slot.
2) A correctly produced Coordinating network block is passed on and put on the waiting list if it is obtained through one slot.
3) A block that is on the waiting list is added to the chain if a block that was generated immediately or with a delay references it.
4) A block of the Coordinating network is erased and not forwarded further if it is received more than two slots after it was produced.

In order for the procedures described above to be possible, it is necessary that all nodes know the public keys of each and their total number. This means that a permissioned blockchain is needed for this. For more information about the consensus, see [23].

3.3. Committee Formation Problem

Consider the problem of optimal partitioning of the set of Coordinators into Committees to make a finalization decision. The optimization problem is two criteria. First, we should minimize the number of messages transmitted. Secondly, to reduce the influence of faulty Coordinators on the finalization decision in each slot. In this case, we will call the faulty Coordinators that, for any reason, do not participate in the work of the Committee.

3.3.1. Minimizing the Number of Messages

Let

- n be the number of Coordinators;
- c be the number of Committees;
- m be the number of Committee Members.

It is obvious that $n = 32 \cdot c \cdot m$ from the assumption that all Coordinators take part in the work of the Committee once in an epoch. We consider that there are 32 slots in the epoch. When creating a block in the Coordination network, first the Committee Members exchange messages within the BFT protocol, and then the aggregators (Committee leaders) also exchange messages within the BFT protocol. Thus, the total number of messages is:

$$M = 2 \cdot m \cdot (m-1) \cdot c + 2 \cdot c \cdot (c-1) = 2 \left[\frac{n \cdot (m-1)}{32} + c \cdot (c-1) \right]. \quad (3.1)$$

Figure (3.1) illustrates dependency of number of messages depending on the number of Members and Committees by the formula (3.1):

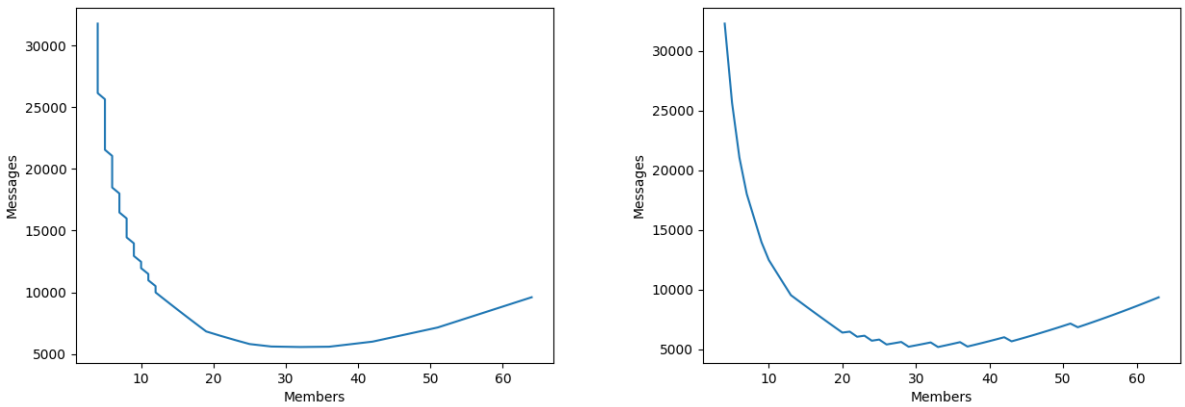

Figure 3.1. Dependency of messages on Members and Committees

Let us consider the question of the values of c and m for which the number of transmitted messages will be minimal.

Since $m = \dfrac{n}{32 \cdot c}$, the problem is reduced to minimizing the function (neglect the term that does not contain c):

$$M(c) = \frac{n^2}{1{,}024 \cdot c} + c^2 - c, \quad c \in \left[4, \frac{n}{128}\right]. \tag{3.2}$$

An additional condition is that c only accepts integer values. Also, due to the limitations of the BFT protocol, we assume that $c \geqslant 4$ and $m \geqslant 4$. From the last inequality, in particular, it follows that $c \leqslant \dfrac{n}{128}$. In addition, we assume that the number of Coordinators is sufficiently large. The approximate value of the minimum point can be calculated analytically:

$$c_0 \approx \sqrt[3]{\frac{n^2}{2{,}028}}. \tag{3.3}$$

Figure (3.2) illustrates the dependency of the number of messages depending on the number of Committees by the formula (3.2):

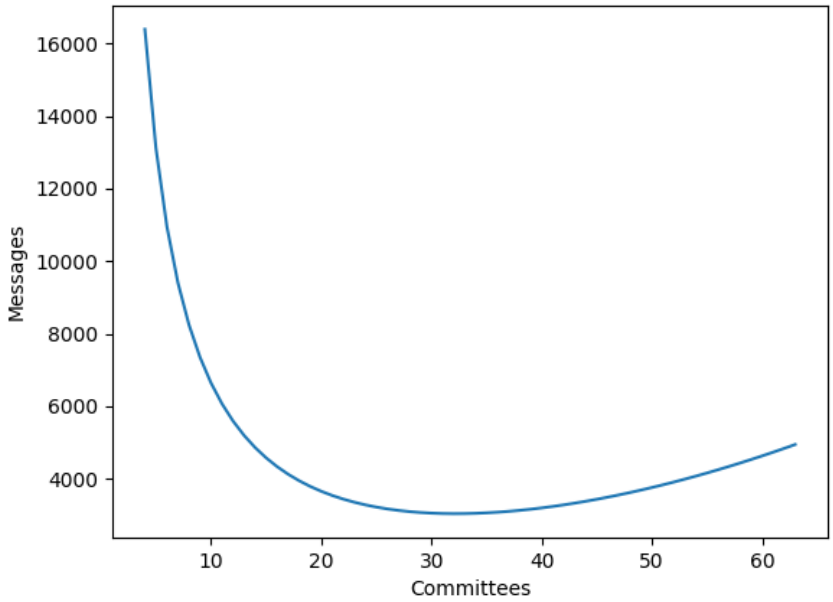

Figure 3.2. Dependency of messages on Committees (reduced formula)

Based on the properties of the BFT protocol, we can recommend choosing the number of Committees as an integer of the form $3 \cdot f + 1$ closest to c_0.

With the help of numerical optimization methods or mathematical tools, one can get a more accurate answer. However, for a sufficiently large number of Coordinators, the answer for an approximate solution and a more accurate one is the same.

3.3.2. Minimizing the Number of Missed Slots

Due to the fact that some of the Coordinators may turn out to be faulty, we need to find out how this will affect the decision in the Committees and the final decision on finalization. In this case, it is necessary to estimate what is the maximum proportion of faulty Coordinators that is acceptable without stopping or significantly delaying the decision-making process.

Formulation of the Problem

For a given number of Coordinators, find such a number of Committees and determine the number of their Members, at which the average number of

missed slots per epoch will be in a certain sense “minimal”. Missed slots are those slots in which it was not possible to accept the block. Obviously, the number of missed slots will depend on the proportion of faulty Coordinators and their distribution within the Committees. The task is to give a method for constructing a partition that, on average (in most cases), will give a smaller number of missed slots than other partitions. At the same time, the number of sent messages should also be taken into account, possibly minimizing it.

General Provisions

According to the protocol, a decision can be made both within the Committee by Members of the Committee and between Committees by aggregators only if there are more than $\frac{2}{3}$ of votes in favor of this decision. In this case, the percentage of majority (Share) required for making a decision will be the smallest if the number of Members as a number can be represented as $3 \cdot f + 1$, i.e. when divided by 3, the remainder is 1. A few examples of how Share changes depending on Members are given below (see figure (3.3)):

$$Majority = \frac{2}{3} \cdot Members;$$

$$Share = \frac{round\,(Majority - 0.5) + 1}{Members} \cdot 100\%.$$

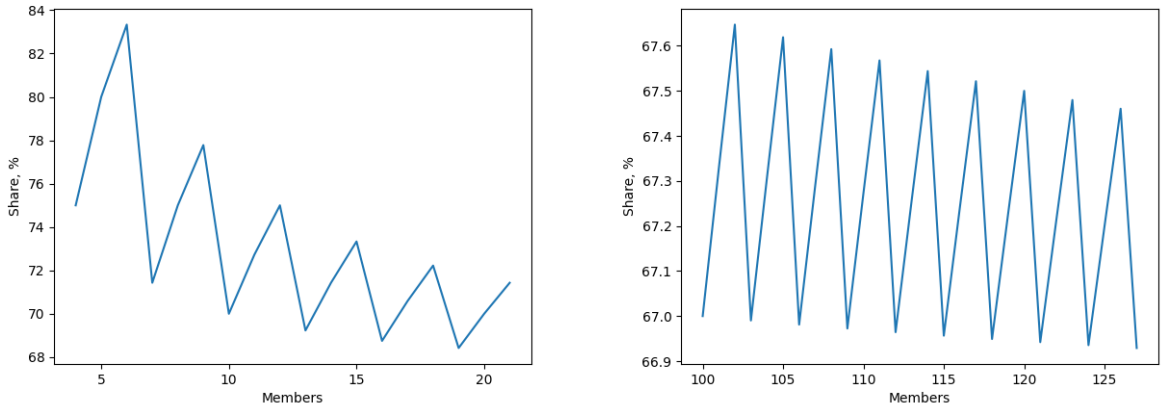

Figure 3.3. Dependency of Share on Members

Obviously, as the value of Members increases, the fluctuations in the value of Share decrease. This is also consistent with the results of mathematical

modeling. A smaller average number of missed slots per epoch was observed when the number of Committees and their Members were in form of $3 \cdot f + 1$.

Using simulation modeling, it was found that the critical value of the total number of faulty Coordinators, at which there is no large delay in the finalization of blocks, is 20% [24]. Figure (3.4) shows the number of missed slots per epoch depending on the proportion of faulty Coordinators. On the first graph, the number of Committees is 4 and the number of Committee Members is 64, on the second graph - on the contrary. On the third graph, the number of Committees is 128 and the number of Committee Members is 16.

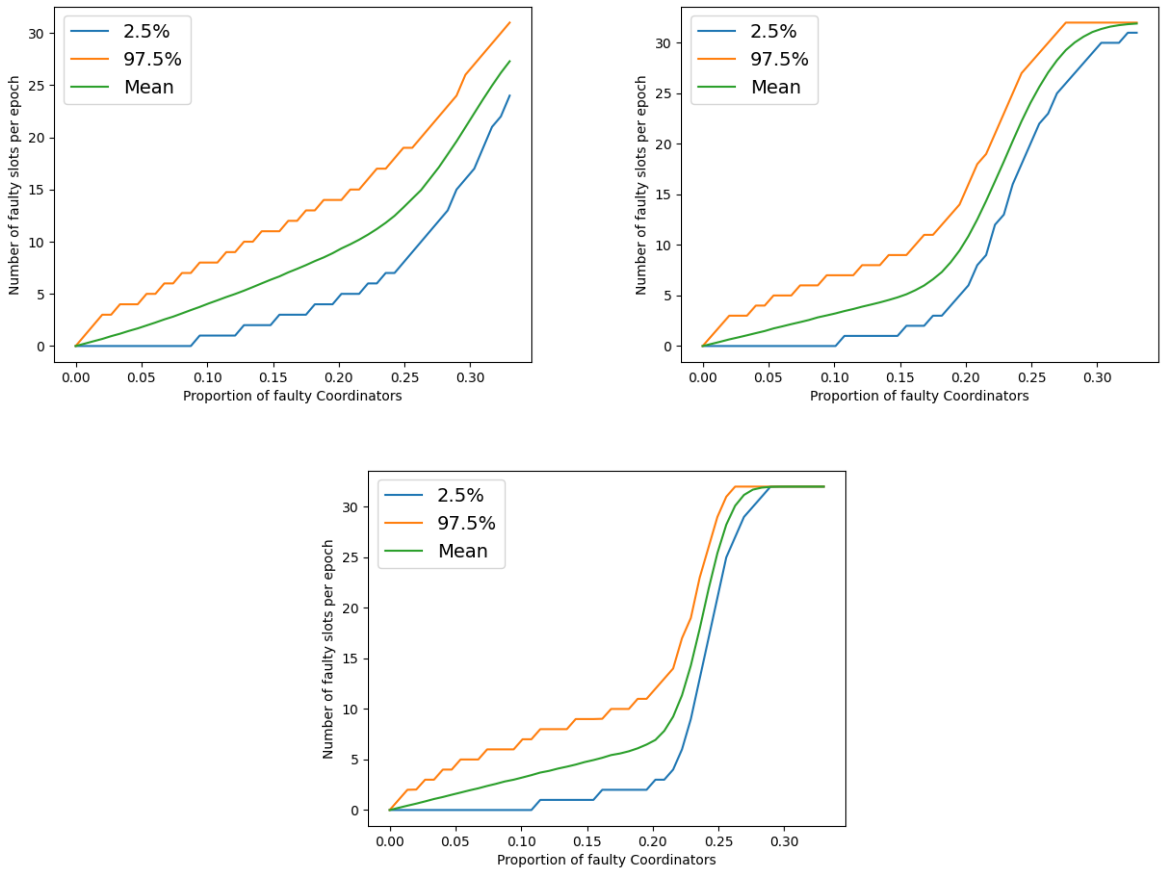

Figure 3.4. Dependency of the number of missed slots on the proportion of the faulty Coordinators

In addition, during the simulation, it was found that the best result (lower average number of missed slots per epoch) with a fixed number of Coordinators is achieved with a decrease in the number of Committees and with a corresponding increase in the number of their Members. Figure (3.5)

shows the number of missed slots per epoch depending on the number of Committees and Committee Members. Green dots indicate the number of obtained missed slots up to 11, yellow - from 11 to 21, and red - from 22.

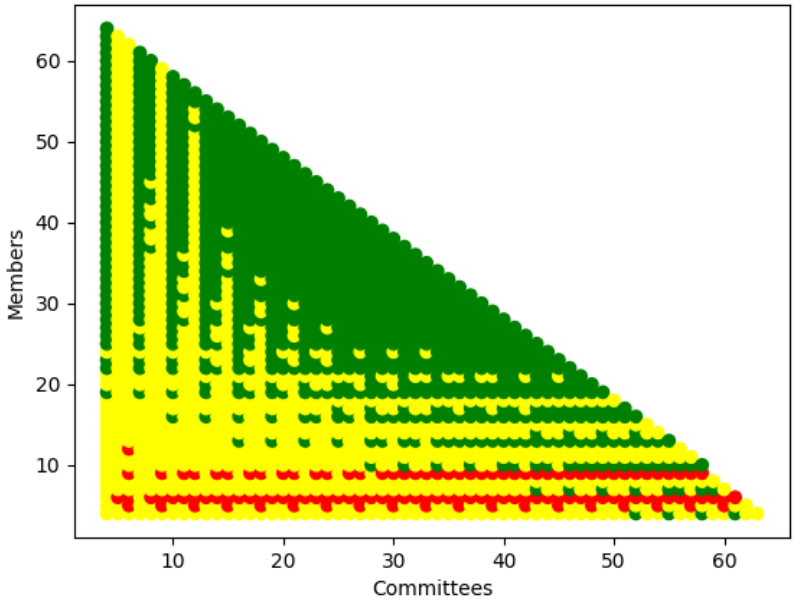

Figure 3.5. Dependency of the number of missed slots on the number of Committees and Committee Members

However, due to technical restrictions on the exchange of messages within the Committee, the number of Committee Members should not exceed 128.

Partitioning algorithm

Algorithm 3.2 (Partitioning algorithm)**.** Let

- $numberCoordinators$ be the total number of Coordinators in the network;
- $maxMembers = 127$ be the maximum possible number of Committee Members.

In the first version of the formulas, all variables are positive integers, and when dividing, the result is rounded down. In the second version of the formulas, non-integer division is used.

1) Minimum number of Committees to which all Members can be placed.

$$minCommittees = \frac{numberCoordinators - 1}{32 \cdot maxMembers} + 1;$$

$$minCommittees = ceiling\left(\frac{numberCoordinators}{32 \cdot maxMembers}\right).$$

It is assumed that in an epoch there are 32 slots and all Coordinators should participate in the work of Committees once per epoch.

2) The number of Committees of form $3 \cdot f + 1$.

$$Committees = \left(\frac{minCommittees - 2}{3} + 1\right) \cdot 3 + 1;$$

$$Committees = max\left(4, ceiling\left(\frac{minCommittees - 1}{3}\right) \cdot 3 + 1\right).$$

where $Committees$ is the minimum number of form $3 \cdot f + 1$ greater than $minCommittees$. This number of Committees will be used in each slot.

3) The mean number of Committee Members.

$$meanMembers = \frac{numberCoordinators}{32 \cdot Committees};$$

$$meanMembers = floor\left(\frac{numberCoordinators}{32 \cdot Committees}\right).$$

4) The minimum number of Committee Members of form $3 \cdot f + 1$.

$$minMembers = \left(\frac{meanMembers - 1}{3} + 1\right) \cdot 3 - 2;$$

$$minMembers = floor\left(\frac{meanMembers - 1}{3}\right) \cdot 3 + 1.$$

where $minMembers$ is the maximum number of the form $3 \cdot f + 1$ that does not exceed $meanMembers$. In other words, each of the Committees has $minMembers$ Coordinators.

5) Number of Coordinators not assigned to Committees.

$$restMembers = numberCoordinators - 32 \cdot Committees \cdot minMembers.$$

6) Number of triples formed from these Coordinators.

$$triples = \frac{restMembers}{3};$$

$$triples = floor\left(\frac{restMembers}{3}\right).$$

7) Number of Coordinators not included in the triples.

$$nextMembers = restMembers\%3.$$

8) Distribution of triples by Committees.

a) Case 1.

The number of first slots of the epoch in which all Committees will be increased by 3 Coordinators:

$$firstSlots = \frac{triples}{Committees};$$

$$firstSlots = floor\left(\frac{triples}{Committees}\right).$$

And in the next slot the number of Committees that are increased:

$$nextCommittees = triples\%Committees$$

by 3 Coordinators. Also in this slot, one can increase one more of the Committees by $nextMembers$ Coordinators. The following equality must be true:

$$numberCoordinators = 32 \cdot Committees \cdot minMembers+$$
$$+firstSlots \cdot Committees \cdot 3 + nextCommittees \cdot 3 + nextMembers.$$

b) Case 2.

The minimum number of Committees from each slot that will receive 3 additional Coordinators:

$$fullCommittees = \frac{triples}{32};$$

$$fullCommittees = floor\left(\frac{triples}{32}\right).$$

In addition, a certain number of slots should receive one more Committee each with an increased number of Coordinators by 3 (for example, at the

beginning of the epoch):

$$extraSlots = triples\%32.$$

As above, one more of the Committees can be increased, for example, one from the next slot by $nextMembers$ Coordinators. The equality must hold:

$$numberCoordinators = 32 \cdot Commitees \cdot minMembers+$$
$$+32 \cdot fullCommittees \cdot 3 + extraSlots \cdot 3 + nextMembers.$$

The result is:

- The number of Committees in each slot is Committees, the minimum possible value of the form $3 \cdot f + 1$.
- In all Committees, the number (also of the form $3 \cdot f + 1$) of Members is $minMembers$ or $minMembers + 3$, except perhaps for one Committee with 1 or 2 more Members than $minMembers$.
- The number of Committees with "increased" by 3 Members is triples. Such Committees can be allocated to slots in various ways.

Example

Let's take 16257 Coordinators. The minimum number of Committees:

$$minCommittees = ceiling\left(\frac{16257}{32 \cdot 127}\right) = 5.$$

The minimum number of Committees of the form $3 \cdot f + 1$, which is not less than the $minCommittees$:

$$Committees = ceiling\left(\frac{4}{3}\right) \cdot 3 + 1 = 7.$$

Each Committee will have in average:

$$meanMembers = floor\left(\frac{16257}{32 \cdot 7}\right) = 72$$

Members. Then, representing the number of Committee Members as $3 \cdot f + 1$, we get:

$$minMembers = floor\left(\frac{71}{3}\right) * 3 + 1 = 70.$$

Then the number of Coordinators who were not included in the Committees:

$$restMembers = 192001 - 32 \cdot 7 \cdot 70 = 577.$$

In order to add them to other Committees without violating the $3 + 1$ form, we divide the rest of the Coordinators into triples. We get:

$$triples = floor\left(\frac{577}{3}\right) = 192.$$

As a result, $nextMembers = 1$.

Case 1. Determine in how many first slots of the epoch all Committees will be increased by 3 Coordinators:

$$firstSlots = floor\left(\frac{192}{7}\right) = 27.$$

And in the next slot we increase:

$$nextCommittees = 192\%7 = 3$$

Committees for 3 Coordinators. In the fourth Committee in this slot, we add the remaining Coordinator.

Case 2. Determine the minimum number of Committees in each slot that will receive 3 additional Coordinators:

$$fullCommittees = floor(192/32) = 6.$$

The remaining Coordinator goes to the Committee of some slot.

Simulation Modeling

For the resulting partitions, graphs of the dependence of the number of missed slots on the number of Coordinators were plotted for various percentages of faulty Coordinators for both cases (see figure (3.6)).

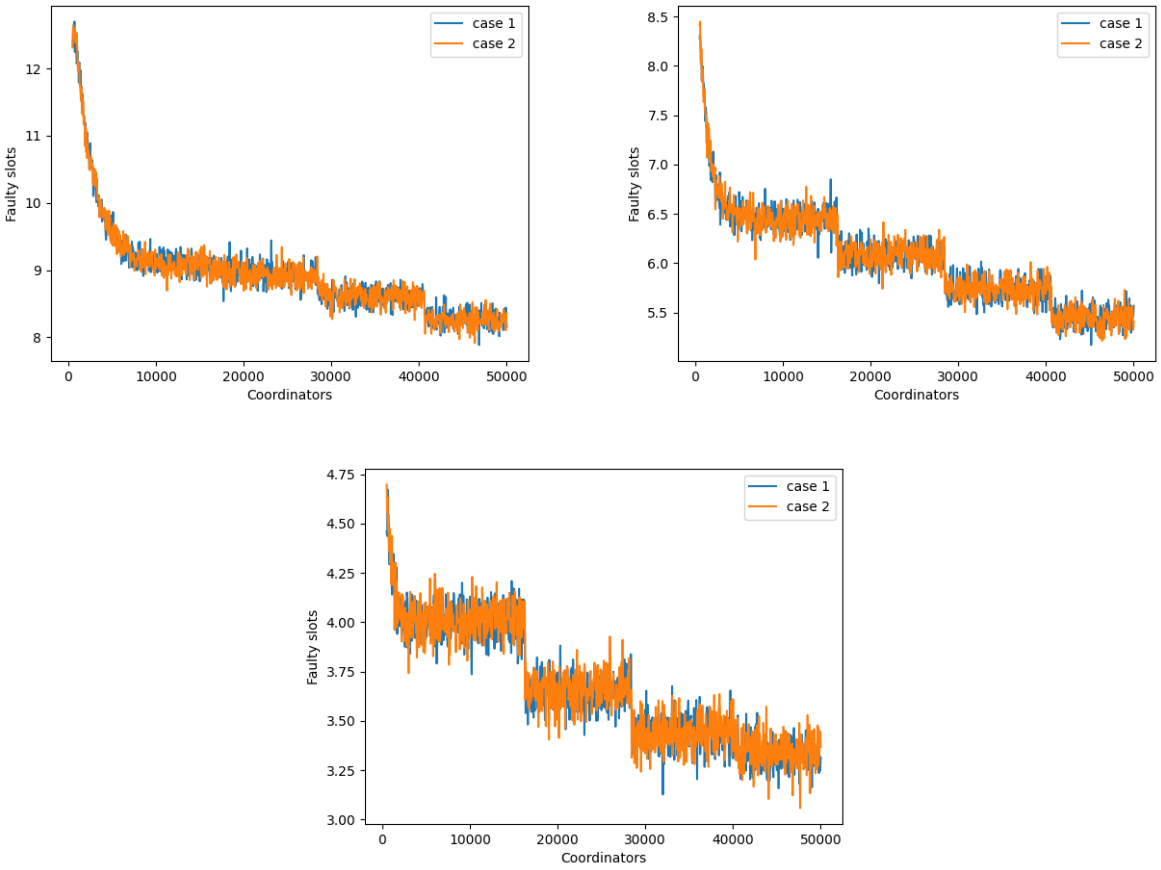

Figure 3.6. Dependency of the number of missed slots on the number of Coordinators

From the graphs, one can see that the average number of missed slots does not exceed 13 with a percentage of faulty Coordinators of $\leqslant 20\%$. It is also found that both proposed methods are not inferior to each other.

The code of the algorithm and simulation can be found in appendix A.

3.4. The Problem of Penalizing Faulty Coordinators

3.4.1. Types of Coordinators' Faultiness

The article [25] details the types of Coordinator misbehavior that can be captured on the Coordination Network ledger. Penalties are calculated automatically based on the recorded information. Let's consider the influence of faulty nodes on the processes in the Coordinating network. Since the finalization of the distributed ledger is performed in the Coordinating network, the following problems may cause it to be delayed.

1) **Vote Omissions**. It can be assumed that if a certain Coordinator does not vote several times in a row, it is out of order. In this case, it is penalized.
2) **Missing blocks**. If there is no previous block(s) in one or more slots in a row, the current slot leader refers to the most recently received block. A Coordinator is penalized for failing to make blocks. In both situations, all penalized Coordinators are prohibited from taking part in the network's operation for the current and following Eras.
3) **Duplicate Creation**. The current leader must only produce one block per slot in the Coordinating Network in accordance with the protocol's requirements. If a certain Coordinator does not follow the rule, it is penalized.
4) **Conflicting Messages**. A Committee Member can sign and send messages containing conflicting information. In this case, it is also penalized.
5) **Invalid proof**. The leader can submit invalid proof of attacks in its block. It is penalized for such kind of action.

3.4.2. Simulation Modeling

The fundamental rule when determining penalties is that they need to be much higher than any potential gain from potential attacks. To scale

the magnitude of the penalties, in this case, a multiplier of $\lambda \geqslant 1$ is applied. Finding a value that would be acceptable under various Coordinator misbehavior scenarios is one of the simulation's goals. It is necessary that faulty or inactive Coordinators eventually lose their right to take part in the consensus. By doing this, a critical mass of faulty nodes that might otherwise block the reaching of consensus is prevented. However, inadvertent Coordinator shutdowns or short equipment malfunctions should not result in a Coordinator's permanent ban.

The following scenarios for Coordinator failures are considered:

1) h_{online} hours are on and $h_{offline}$ hours off;
2) the shutdown duration is distributed according to the normal law with mean t_{mean} and standard deviation t_{std};
3) the probability of failure when performing one or another action is $P \in (0; 1)$.

In these cases, the period of time after which the Coordinator was no longer allowed to participate in the work is fixed and depends on the value of the various parameters listed above.

It is recommended to set a rather high value of $\lambda = 100$ based on the results. For instance, the penalty for failing to create two blocks in a row or for voting with the submitting of a conflicting message is 100 times greater than the equivalent incentives. In order to make the necessary equipment modifications, the Coordinator's involvement in the network is also temporarily interrupted (for the present and following eras). Otherwise, in the event of further failures, the stake value will fall below 50%, which will result in the Coordinator's permanent ban. As a result, the Coordinator's stake is preserved and it will be able to participate in the network in the future.

Table (3.1) considers the average number of eras after which a completely non-working Coordinator is removed from participation in the network depending on the scaling parameter. Each era lasts ≈ 9.1 hours.

Table 3.1.

The average number of eras after which a completely non-working Coordinator is banned depending on λ

λ	Eras
1	400000
10	39000
100	3500
1000	500
10000	50
100000	7

3.5. Conclusion

If the number of faulty Coordinators is obviously small, then one can use the formula (3.3) to calculate the number of Committees and Committee Members.

In the case when the smallest possible number of missed slots is important, the Partitioning algorithm performed well. Those Coordinators who did not get into $minMembers$ can be distributed in different ways to Committees. To reduce the exchange of messages within the Committee, the maximum number of Committee Members is set to 127.

The scaling multiplier λ for penalties is recommended to be set at $\lambda = = 100$. At the same time, a completely non-working Coordinator is permanently banned from participating in the network after a long period of time (taking into account the time of temporary suspension), and the penalties for violations are significantly greater than the rewards.

ВИСНОВКИ

Робота була присвячена дослідженню ієрархічного консенсусу в блокчейнах контрольованого складу за допомогою аналітичного аналізу та моделювання на мові програмування Python.

1) Було описано модель ієрархічного консенсусу на основі протоколу Gozalandia.
2) Було описано та вирішено проблему мінімізації числа повідомлень, якими обмінюються Координатори.
3) Було виявлено, що найбільший процент несправних Координаторів, при якому не відбувається великої затримки у фіналізації блоків - це 20%.
4) Було виявлено, що найкраще число для кількості Учасників Комітету та Комітетів має вигляд $3 \cdot f + 1$ для зменшення кількості пропущених слотів. При цьому кількість Комітетів має бути якомога меншою.
5) Було побудовано Алгоритм розбиття та виявлено, що кількість пропущених слотів є низькою при його застосуванні.
6) Було розглянуто проблему знаходження параметра масштабування λ, який відповідає за штрафи за атаки, роблячи їх недоцільними. Було рекомендовано значення цього параметру $\lambda = 100$.

Щоб краще дослідити цю тему, можна протестувати роботу такої децентралізованої системи.

Результати досліджень були представлені на науковій конференції [24].

CONCLUSION

The work was dedicated to the study of hierarchical consensus in permissioned blockchains using analytical analysis and modeling in the Python programming language.

1) A hierarchical consensus model based on the Gozalandia protocol was described.
2) The problem of minimizing the number of messages exchanged by Coordinators has been described and resolved.
3) It was found that the largest percentage of faulty Coordinators, at which there is no significant delay in block finalization, is 20%.
4) It was found that the best number for the number of Committee Members and Committees is $3 \cdot f + 1$ to reduce the number of missed slots. At the same time, the number of Committees should be as small as possible.
5) A Partitioning algorithm was built and the number of missed slots was found to be low when it was applied.
6) The problem of finding the scaling parameter λ, which is responsible for the penalties for the attacks, making them infeasible, was considered. The value of this parameter $\lambda = 100$ was recommended.

To better explore this topic, we should test the operation of such a decentralized system.

The research results were presented at the scientific conference [24].

BIBLIOGRAPHY

1. Wright C. Bitcoin: A Peer-to-Peer Electronic Cash System // University of Southern Queensland. — August, 2008. — DOI: http://dx.doi.org/10.2139/ssrn.3440802
2. Anderson M. Exploring Decentralization: Blockchain Technology and Complex Coordination // Anderson M. — Journal of Design and Science. — 2019. — Resource access mode: https://jods.mitpress.mit.edu/pub/7vxemtm3/release/2
3. Корнага Я. І. Дослідження та застосування технології блокчейн у транспортній логістиці / Корнага Я. І., Тільняк Ю. Я. — ВІСНИК ЖДТУ. — 2019. – № 1 (83). — DOI: https://doi.org/10.26642/tn-2019-1(83)-12-17
4. Жмуркевич А. Є. Особливості застосування блокчейн-технології у цифровій економіці / Жмуркевич А. Є., Вакулін Р. С. — Міжнародний науковий журнал "Інтернаука". — 2018. — № 6(2). — С. 14-17. — Режим доступу: http://nbuv.gov.ua/UJRN/mnj_2018_6(2)__5
5. Bajwa N. K. Modelling and Simulation of Blockchain based Education system / Bajwa N. K. —- Masters thesis, Concordia University. — 2018. — Resource access mode: https://spectrum.library.concordia.ca/984170/1/Bajwa_MASc_S2018.pdf
6. Spirkina A. Approaches to Modeling Blockchain Systems / [Spirkina A., Aptrieva E., Elagin V. et al.] — 2020 12th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT). — 2020. — DOI: 10.1109/ICUMT51630.2020.9222437
7. Золотарьова І. О. Інформаційні технології оптимізації роботи приватного блокчейн за допомогою вибору алгоритму консенсусу / Золотарьова І. О., Плеханова Г. О. — Системи обробки інформації. — 2020. — № 1(160). — С. 107-14. — DOI: https://doi.org/10.30748/soi.2020.160.14
8. Antwi R. A Survey on Network Optimization Techniques for Blockchain Systems / [Antwi R., Gadze J., Tchao E. et al.] — Algorithms 2022. — 15. — 193. — DOI: https://doi.org/10.3390/a15060193
9. Anu M. Introduction to modeling and simulation / Anu M. — WSC '97: Proceedings of the 29th conference on Winter simulation. — December, 1997.

— p. 7-13. — DOI: https://doi.org/10.1145/268437.268440

10. Downey A. Modeling and Simulation in Python / Downey A. — Green Tea Press. — 2017. — Resource access mode: https://greenteapress.com/modsimpy/ModSimPy3.pdf
11. Why use simulation modeling? // Anylogic: web site. — URL: https://www.anylogic.com/use-of-simulation/ (date of application: 05.12.2022)
12. Mathematical optimization / Wikipedia, the free encyclopedia: web site. — URL: https://en.wikipedia.org/wiki/Mathematical_optimization (date of application: 05.12.2022)
13. Rossum G. Python tutorial / Rossum G. — Technical Report CS-R9526. — Centrum voor Wiskunde en Informatica (CWI) Amsterdam. — May, 1995. — Resource access mode: https://ir.cwi.nl/pub/5007/05007D.pdf
14. Brewer E. Towards robust distributed systems / Brewer E. — Proceedings of the XIX annual ACM symposium on Principles of distributed computing. — Portland, OR: ACM, 2000. — Vol. 19, no. 7. — DOI: doi:10.1145/343477.343502
15. Grybniak S. Decentralized platforms: Goals, challenges, and solutions / [Grybniak S., Leonchyk Y., Masalskyi R. et al] — 2022 IEEE 7th Forum on Research and Technologies for Society and Industry Innovation (RTSI). — 2022. — pp. 62-67. — DOI: 10.1109/RTSI55261.2022.9905225
16. Distributed Systems - The Complete Guide / Confluent: web site. — URL: https://www.confluent.io/learn/distributed-systems/
17. Blockchain / Wikipedia, the free encyclopedia: web site. — URL: https://en.wikipedia.org/wiki/Blockchain (date of application: 05.12.2022)
18. Types of Blockchain / Geeks for Geeks: web site. — URL: https://www.geeksforgeeks.org/types-of-blockchain/ (date of application: 05.12.2022)
19. Frankenfield J. Consensus Mechanism (Cryptocurrency) / Frankenfield J. — Investopedia: web site. — URL: https://www.investopedia.com/terms/c/consensus-mechanism-cryptocurrency.asp (date of application: 30.11.2022)
20. Lamport L., Shostak R., Pease M. C. The Byzantine Generals Problem /

ACM Transactions on Programming Languages and Systems. — Vol. 4. — No. 3. — July 1982. — DOI: 10.1145/357172.357176

21. Castro M. Practical Byzantine Fault Tolerance / Castro M., Lickov B. — OSDI '99: Proceedings of the third symposium on Operating systems design and implementation. — February 1999. — p. 173–186. — DOI: 10.1002/9781119682127.ch7
22. Grybniak S., Dmytryshyn D., Leonchyk Y., Mazurok I., Nashyvan O., Shanin R. Waterfall: A Scalable Distributed Ledger Technology // IEEE 1st GET Blockchain Forum, California. — United States. — 2022. — In press.
23. Grybniak S. Waterfall: Salto Collazo. Tokenomics / [Grybniak S., Leonchyk Y., Masalskyi R. et al.] — IEEE International Conference on Blockchain, Smart Healthcare and Emerging Technologies. — Bucharest, Romania. — 2022. — In press.
24. Vorokhta A. Simulation Modelling of the Consensus Based on the Gozalandia / [Vorokhta A., Mazurok I., Leonchyk Y. et al.] — Adaptive Learning Management Technologies. — Kyiv. — 2022. — p. 31-33.
25. Mazurok I. An incentive system for decentralized DAG-based platforms / [Mazurok I., Leonchyk Y., Grybniak S. et al.] — Applied Aspects of Information Technology. — vol. 5(3). — 2022. — pp. 196–207.

APPENDIX A

COMPUTER CODE FOR THE PARTITIONING ALGORITHM

```
import math
import numpy as np
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt

maxMembers = 127
numbersCoordinators = np.arange(500, 50050, step=50)
resultsCase1 = []
resultsCase2 = []
for numberCoordinators in numbersCoordinators:
    minCommittees = math.ceil(numberCoordinators / 32 /
       maxMembers)
    Committees = max(4, math.ceil((minCommittees - 1) / 3) * 3
       + 1)
    meanMembers = math.floor(numberCoordinators / 32 /
       Committees)
    minMembers = math.floor((meanMembers - 1) / 3) * 3 + 1
    restMembers = numberCoordinators - 32 * Committees *
       minMembers
    triples = math.floor(restMembers / 3)
    nextMembers = restMembers % 3
    faultyCoordinators = int(numberCoordinators * 0.2)
    overallNumberMissedSlots = 0
    Coordinators =
       np.append(np.zeros(int(faultyCoordinators)),
       np.ones(numberCoordinators - int(faultyCoordinators)))

    # Case 1
    firstSlots = math.floor(triples / Committees)
    nextCommittees = triples % Committees
    for i in range(10000):
```

```
        np.random.shuffle(Coordinators)
        for slot in range(32):
            index = 0
            numberFaultyCommittees = 0
            leaderIndex = 0
            for committee in range(Committees):
                members = minMembers
                if slot < firstSlots:
                    members = minMembers + 3
                elif slot == firstSlots and committee <
                   nextCommittees:
                    members = minMembers + 3
                elif slot == firstSlots and nextCommittees ==
                   committee:
                    members = minMembers + nextMembers

                numberFair = np.sum(Coordinators[index:index +
                   members])
                numberFaulty = members - numberFair

                if Coordinators[index] == 0 or numberFaulty >=
                   1 / 3 * members:
                    numberFaultyCommittees += 1

                if committee == Committees - 1:
                    leaderIndex = index
                index += members

            if numberFaultyCommittees >= 1 / 3 * Committees or
               Coordinators[leaderIndex] == 0:
                overallNumberMissedSlots += 1

    missedSlotsAvg = overallNumberMissedSlots / 10000
    resultsCase1.append(missedSlotsAvg)

    # Case 2
    fullCommittees = math.floor(triples / 32)
    extraSlots = triples % 32
    overallNumberMissedSlots = 0
```

```
    for i in range(10000):
        np.random.shuffle(Coordinators)
        for slot in range(32):
            index = 0
            numberFaultyCommittees = 0
            leaderIndex = 0
            for committee in range(Committees):
                members = minMembers
                if committee < fullCommittees:
                    members = minMembers + 3
                elif committee == fullCommittees and slot <
                   extraSlots:
                    members = minMembers + 3
                elif committee == fullCommittees and slot ==
                   extraSlots:
                    members = minMembers + nextMembers

                numberFair = np.sum(Coordinators[index:index +
                   members])
                numberFaulty = members - numberFair

                if Coordinators[index] == 0 or numberFaulty >=
                   1 / 3 * members:
                    numberFaultyCommittees += 1

                if committee == Committees - 1:
                    leaderIndex = index
                index += members

            if numberFaultyCommittees >= 1 / 3 * Committees or
               Coordinators[leaderIndex] == 0:
                overallNumberMissedSlots += 1

    missedSlotsAvg = overallNumberMissedSlots / 10000
    resultsCase2.append(missedSlotsAvg)


plt.plot(np.arange(500, 50050, step=50), resultsCase1,
   label="case 1")
```

```
plt.plot(np.arange(500, 50050, step=50), resultsCase2,
   label="case 2")
plt.xlabel("Coordinators")
plt.ylabel("Faulty slots")
plt.legend()
plt.show()
```