МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование кафедры)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки, специальности)

Технология программирования
(направленность (профиль)/специализация)

БАКАЛАВРСКАЯ РАБОТА

на тему «Разработка децентрализованной пиринговой системы для криптовалюты»

Студент	А.О. Швечков	
	(И.О. Фамилия)	(личная подпись)
Руководитель	А.В. Очеповский	
	(И.О. Фамилия)	(личная подпись)
Консультанты	А.В. Прошина	
	(И.О. Фамилия)	(личная подпись)

Допустить к защите

Заведующий кафедрой к.т.н, доцент кафедры ПМИ, А.В. Очеповский _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)

«_____»_____2018 г.

Тольятти 2018

АННОТАЦИЯ

Название бакалаврской работы – «Разработка децентрализованной пиринговой системы для криптовалюты».

Объект исследования – процесс функционирования децентрализованных пиринговых систем.

Предмет исследования – алгоритмы консенсуса в blockchain системе.

Цель исследования – разработка авторского алгоритма достижения консенсуса для blockchain сети.

Для достижения цели решаются следующие задачи:

- проанализировать существующие алгоритмы консенсуса;
- выявить недостатки существующих процессов;
- сформировать требования к алгоритму консенсуса;
- разработать архитектуру новой автоматизированной системы сбора, проверки корректности копий и достижения одинаковой корректной копии во всех узлах сети;
- разработать и протестировать систему, на основе построенной архитектуры и выявленных требований.

Современные денежные переводы несовершенны: люди, пользующиеся банками вынуждены доверять безопасность и неприкосновенность данных центральному субъекту в виде банка. Технология блокчейн позволяет решить вопрос доверия, обеспечить неизменность данных и операций без посредников. Такая возможность осуществления транзакций реализуется при помощи пиринговых сетей.

Поскольку узлы в сети должны быть уверены в корректности получаемых данных, были внедрены специальные алгоритмы - алгоритмы консенсуса. В современных системах алгоритм консенсуса представляет собой

математическую задачу, решение на которую можно найти только методом перебора решений к математической функции. Этот процесс крайне энергозатратный и медленный по сравнению с централизованными системами, затраты вычислительных мощностей на поиск решения задачи консенсуса перебором крайне велики и не оправданы, поэтому было принято решение найти ему улучшенную альтернативу.

Бакалаврская работа выполнена на 46 страницах, состоит из введения, трёх глав, включающих 19 изображений, 2 таблица и 7 формул, заключения, списка используемых источников, включающего 26 источника, из них 5 на иностранном языке, и 1 приложения.

ABSTRACT

The title of the thesis is «Development of a decentralized peer-to-peer system for cryptocurrency».

The aim of the work is obtaining the maximum possible performance of simultaneously performed data processing functions by developing an algorithm for verifying the correctness of copies of data between network elements.

The object of the thesis is process that ensures that distributed data between network elements are exact copies.

The subject of the thesis is peer-to-peer network with applied consensus algorithm.

The first part of the thesis describes analysis of existing consensus algorithms, their shortcomings, and on the basis of the data obtained, requirements are formulated for the system being developed.

The second part of the thesis is devoted to development and implementation of the consensus algorithm.

The third part describes the testing of the developed consensus algorithm.

This work will make the use of crypto currency cheaper. This will allow to develop the technology, open up new opportunities and consolidate the basic idea of the crypto currency.

The graduation project consists of an explanatory note on 46 pages, introduction, including 19 figures, 2 tables, the list of 26 references including 5 foreign sources and 1 appendices.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современные денежные переводы несовершенны: люди, пользующиеся банками вынуждены доверять безопасность и неприкосновенность данных центральному субъекту в виде банка. Кроме того, для межбанковских переводов требуется значительное время и оплата больших комиссий. Технология блокчейн позволяет решить вопрос доверия, обеспечить неизменность данных и операций без посредников [13].

Блокчейн позволяет при помощи подписания последующих транзакций хешем предыдущей транзакции достичь неизменность цепочки данных - как только мошенник захочет изменить какую-то транзакцию, хеш будет изменен, а цепочка будет оборвана [15]. Поскольку хеш-функции основываются в первую очередь на криптографических алгоритмах, валюты, транзакции которых подписываются при помощи криптографических хеш-функций, стали называть криптовалютами [21].

Возможность осуществления транзакций без посредников реализуется при помощи децентрализованных пиринговых сетей - каждая единица сети не привязана к какому-либо единому сетевому узлу, а производит обмен данными с другими узлами [6]. Таким образом, в каждом узле сети сохраняются полные актуальные данные всех транзакций криптовалюты.

Принцип применения пиринговых сетей, лишенных постоянных узлов и единой точки входа, был назван децентрализацией. Децентрализованные программные решения позволяют всем узлам сети получать данные о состоянии системы вне зависимости от точки входа, а система становится полностью независимой от любых регуляторов или мошенников [22].

Поскольку узлы в сети должны быть уверены в корректности получаемых данных, были внедрены специальные алгоритмы - алгоритмы консенсуса. В современных системах алгоритм консенсуса представляет собой

математическую задачу, решение на которую можно найти только методом перебора решений к математической функции [16].

Актуальность данной работы состоит в разработке нового алгоритма, позволяющий уменьшить энергозатратность и скорость достижения консенсуса по сравнению с современными методами.

Новизна бакалаврской работы заключается в новом алгоритме решения достижения консенсуса.

Практическая ценность состоит в разработке нового алгоритма к достижению консенсуса в пиринговых системах.

Объект исследования – процесс функционирования децентрализованных пиринговых систем.

Предмет исследования – алгоритмы консенсуса в blockchain системе.

Цель исследования – разработка авторского алгоритма достижения консенсуса для blockchain сети.

Для достижения цели поставлены следующие **задачи**:

– проанализировать существующие алгоритмы консенсуса;

– выявить недостатки существующих процессов;

– сформировать требования к алгоритму консенсуса;

– разработать архитектуру новой автоматизированной системы сбора, проверки корректности копий и достижения одинаковой корректной копии во всех узлах сети;

– разработать и протестировать систему, на основе построенной архитектуры и выявленных требований.

Бакалаврская работа состоит из введения, трёх глав, заключения, списка литературы и приложений.

В первой главе анализируются существующие алгоритмы консенсуса, выявляются их недостатки и на основании полученных данных формулируются требования к разрабатываемой системе.

Во второй главе разрабатывается архитектура новой системы сбора, проверки корректности копий и достижения одинаковой корректной копии во всех узлах сети.

В третьей главе представляются результаты тестирования разработанных алгоритмов на основе построенной архитектуры и выявленных требований.

В заключении подводятся итоги исследования, формируются окончательные выводы и результаты проведенной работы.

1 АНАЛИЗ ДОСТИЖЕНИЯ КОНСЕНСУСА В СУЩЕСТВУЮЩИХ ПИРИНГОВЫХ СИСТЕМАХ

1.1 Обзор моделеи достижения консенсуса

Блокчейн - это распределенная база данных, которая храниться одновременно на множестве компьютеров [15] и представляет собой цепочку блоков, защищенную от изменений. Блок состоит из следующих составляющих:

- номер блока;
- тело блока или все транзакции за определенный момент времени;
- случайное или псевдослучайное число;
- хеш-функция блока;
- хеш-функция предыдущего блока.

Цепочку создает хеш-функция предыдущего блока (рисунок 1.1).

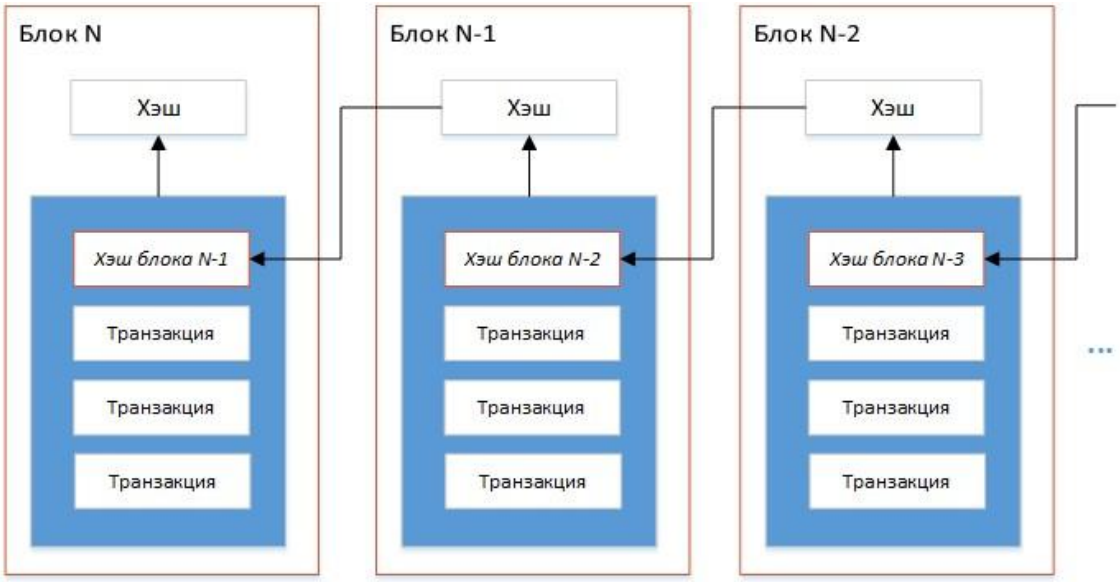

Рисунок 1.1 - Структура блокчейна

Криптовалютой, в свою очередь, является разновидность цифровой валюты, создание и контроль за которой базируется на криптографических методах [13].

Основополагающим фактором работы криптовалюты является децентрализация. Это предполагает отказ от центрального сервера, который будет распределять задачи. Это решается с помощью модели «точка-точка», (peer-to-peer, P2P), которая является основой для создания пиринговых распределенных систем. На рисунке 1.2 показана архитектура пиринговой системы, из которого видно, что в этой архитектуре отсутствует узел, управляющий работой других узлов [22].

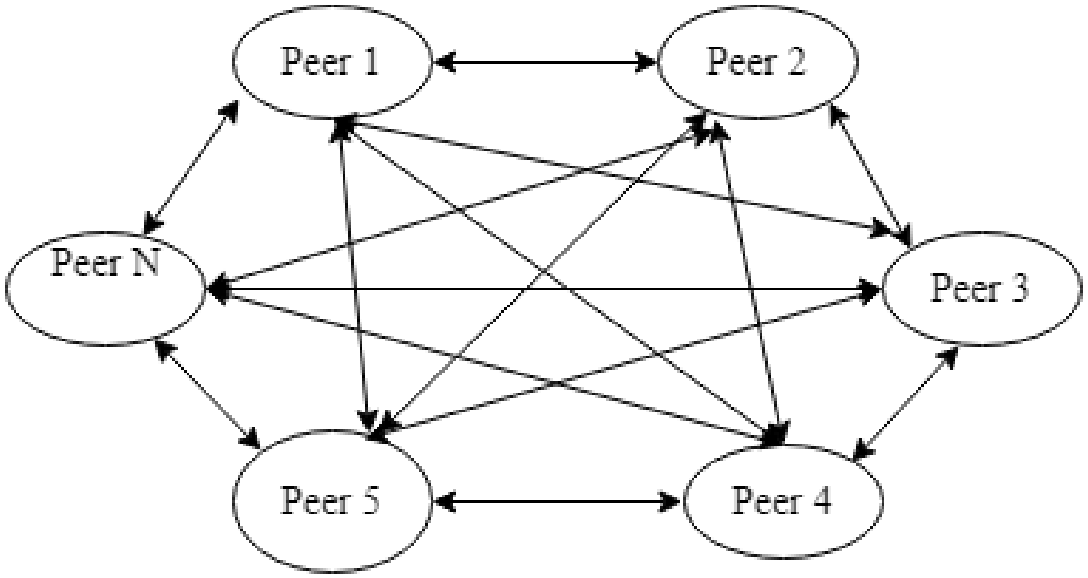

Рисунок 1.2 - Архитектура модели peer-to-peer

Механизм взаимодействия элементов в пиринговой системе основан на многоадресной рассылке, позволяющая организовать обмен данными. Именно связь «точка-точка» оказывается эффективной, когда между элементами пиринговой системы передаются большие объемы данных [10].

Поскольку узлы в сети должны быть уверены в корректности полученных данных, были внедрены специальные алгоритмы - алгоритмы консенсуса. В современных системах алгоритм консенсуса представляет собой математическую задачу, решение на которую можно найти только методом перебора решений к математической функции. Перебирая случайное или

псевдослучайное число, меняется значение хеш-функции блока. Сама хеш-функция включает в себя все данные блока, включая хеш предыдущего блока.

В современных децентрализованных системах используются следующие модели достижения консенсуса [16]:

- Proof-of-Work;
- Proof-of-Stake;
- Delegated Proof-of-Stake;
- Leased Proof-of-Stake;
- Proof-of-Capacity;
- Proof-of-Importance;
- Proof-of-Activity;
- Proof-of-Authority;
- Proof-of-Burn.

Все они направлены на решение византийской задачи генералов, то есть нахождения корректной информации при условии, что узлы передачи этой информации могут трактовать ложную информацию [1].

Алгоритм Proof-of-Work нацелен на решение задачи путем постановки задачи нахождения результата выполнения хеш-функции от определенного набора данных, отвечающего определенным требованиям. Нашедший получает награду; данные, использованные в результате перебора добавляются в общую цепочку блоков нашедшим, а проверить решение может любой желающий просто взяв хеш от полученного решения - он должен соответствовать требованиям [23].

Абстрагируясь от реальных кейсов применения Proof-of-Work, задачу можно представить в нахождении такого значения, хеш от которого будет содержать определенную последовательность определенных знаков. Поскольку предугадать результат хеш-функции невозможно, нахождение решения имеет

случайный характер. Процесс нахождения данных, хеш от которых будет соответствовать требованиям, указанным в задаче называется майнингом (mining) [11]. Поскольку мощности пиринговых сетей майнеров, узлов сети, стремящихся решить задачу, постоянно растут, сложность задачи постоянно повышается [14]. Делается это для того, чтобы было невозможно одновременно найти решение на несколько задач подряд нескольким решающим. Если такого требования не существует, цепочка блоков начинает иметь ветви, что противоречит задаче консенсуса.

Для решения подобных задач требуются значительные затраты ресурсов, что делает алгоритм дорогим в применении и малоэффективным. В виду случайности процесса майнинга невозможно избежать ветвления, поэтому частью алгоритма является условие признания актуальной ветки - самая длинная ветка считается актуальной [12]. В алгоритме Proof-of-Work существует возможность атаки “51%”, подразумевающая, что возможна ситуация, когда группа майнеров обладает большинством мощностей сети, что дает им возможность формировать дальнейшие блоки на свое усмотрение - включая только необходимые транзакции, делая работу системы других пользователей невозможной. Хотя прецедентов атаки подобного рода в крупных сетях не было, вероятность атаки существует, если учесть, что около 50% всей мощности сети находится в Китае, а майнеры ввиду значительной сложности нахождения решения в одиночку объединяются в группы, которые контролируются централизованно.

Алгоритмы Proof-of-Stake и Delegated Proof-of-Stake подразумевают, что узлы сети, добавляющие данные в общую цепочку, выбираются случайно из группы узлов, имеющих наибольший баланс. Очевидным минусом такого алгоритма является возможность получения большого баланса злоумышленником. В результате он с определенной вероятностью может

начать контролировать работу сети, делая невозможным ее использование другими пользователями.

У алгоритма Proof-of-Stake существует еще одна модификация - Leased Proof-of-Stake. В рамках этого алгоритма, любой пользователь имеет возможность передавать свой баланс в аренду майнинг-узлам, а за это майнинг-узлы делятся частью прибыли с пользователями. Таким образом, данный алгоритм консенсуса позволяет получить доход от майнинговой деятельности, не ведя самого майнинга. На данный момент он поддерживается только платформой Waves.

Proof-of-Capacity иногда еще называемый Proof-of-Space (PoSpace), еще один алгоритм консенсуса. Единственная платформа блокчейн, которая поддерживает данный алгоритм, это Burstcoin [24]. PoC работает по следующему принципу:

- каждый майнер вычисляет достаточно большой объем данных, который записывается на дисковую подсистему узла. Такой, первоначальный набор данных в PoC называется "участок";
- для каждого нового блока в блокчейне, майнер читает небольшой набор данных (1/4096, что приблизительно составляет 0.024%) от своего общего сохраненного объема и возвращает результат (дедлайн), как прошедшее время в секундах с момента создания последнего блока, после которого майнер сможет создать новый блок;
- майнер, получивший минимальное время дедлайна, подписывает блок и получает вознаграждение за транзакции.

Таким образом, вычислительные ресурсы необходимые майнеру для этой работы ограничены временем, которое необходимо для чтения файлов из дисковой подсистемы. Именно этот фактор позволяет производить майнинг с достаточно энергоэффективностью. Майнеры соревнуются между собой за

размеры сохранных данных, в отличие от скорости работы оборудования, которое является определяющей в майнинге построенном на PoW.

Алгоритм консенсуса Proof-of-Importance используемый блокчейн платформой NEM [25]. Значимость каждого пользователя в сети NEM определяется, как количество средств, имеющихся у него на балансе и количество проведенных транзакций с/на его кошелек. В отличии от более привычного PoS, который учитывает только баланс имеющихся средств у пользователя, PoI учитывает как количество средств, так и активность пользователя в блокчейн сети. Такой подход вовлекает пользователей не просто держать средства у себя на счету, но и активно использовать их.

Proof-of-Activity - это попытка объединить два наиболее популярных алгоритма, такие как Proof-of-Work и Proof-of-Stake, с целью увеличения уровня защиты от потенциально возможных атак (51% attack, Denial-of-Service attacks – DoS). Принцип работы алгоритма следующий:

- каждый майнер блокчейн сети пробует сгенерировать заголовок пустого блока, который включает в себя хеш предыдущего блока, публичный адрес майнера, индекс текущего блока в блокчейне и случайное или псевдослучайное число (nonce);
- после генерации заголовка пустого блока, отвечающего текущим требованиям сложности, узел рассылает этот заголовок в блокчейн сеть;
- все узлы сети рассматривают заголовок такого блока, как данные полученные от псевдослучайных владельцев. Используя хеш разосланного заголовка блока и хеш предыдущего блока с использованием алгоритма follow-the-satoshi выбираются заинтересованные стороны (стейкхолдеры);
- каждый стейкхолдер, находящийся online, проверяет полученный, пустой заголовок блока на его корректность. Во время проверки, каждый получивший заголовок, проверяет: является ли он одним из первых N-1

стейкхолдеров этого блока и в этом случае подписывает заголовок пустого блока своим секретным ключом и отправляет его в блокчейн сеть;

- когда N-й стейкхолдер видит, что он должен стать подписантом этого блока, он, в дополнение к заголовку пустого блока, добавляет блок с включенными транзакциями (количество включаемых транзакций он выбирает сам), все подписи N-1 от других стейкхолдеров и подписывает блок;
- стейкхолдер N рассылает новый, подготовленный блок. Узлы получают этот блок, убеждаются в его законности и добавляют этот блок в блокчейн.
- премия за транзакции, которую получил N-стейкхолдер, распределяется между майнером и N стейкхолдерами.

Proof-of-Authority алгоритм консенсуса, стоящий несколько особняком от остальных алгоритмов, так как для своей работы ему не требуется иметь вообще какого-либо майнинга, как в случае с PoW или PoS. В блокчейн сети базирующемся на PoAuthority, все транзакции и блоки проверяются посредством одобренных эккаунтов (валидаторов). Проведение транзакций и создание блоков, проходит в автоматическом режиме при помощи вычислительных мощностей валидатора.

Положительным моментом данного алгоритма является отсутствие майнинга и как следствие, существенное снижение затрат на его обслуживание.

Негативный момент использования данного алгоритма, как понятно из самого описания - ключевыми лицами, являются валидаторы, что приводит к централизации. Вероятно, в некоторых случаях, в приватных сетях и при помощи полностью (на сколько это возможно) доверенных аккаунтов это имеет смысл [26].

Следующий тип алгоритма консенсуса Proof-of-Burn. При его использовании майнер отправляет монеты на случайный адрес сгенерированного хеша, потратить средства с этого адреса практически

невозможно, так как вероятность подобрать к нему ключи стремится к нулю. За такое сжигание монет, майнер получает постоянный шанс найти PoB блок и получить за него награду. Шансы на майнинг увеличиваются при увеличении количества сожженных монет. Экономически этот процесс сжигания монет можно представить как покупку буровой установки для майнинга [18].

В таблице 1.1 представлено сравнение алгоритмов консенсуса по основным требованиям данной технологии [17].

Таблица 1.1 – Сравнение алгоритмов консенсуса

Алгоритм консенсуса	Децентрализация	Стоимость транзакции	Наличие математической задачи	Возможные атаки	Скорость проведения транзакции
PoW	есть	$1.39	есть	DoS, атака Сибиллы, Эгоистичный майнинг	1 час
PoS	есть	$0.31	есть	Краткосрочная атака, атака издалека, атака накоплением возраста монет, атака предвычислением, DoS, атака Сибиллы,	10 минут
DPoS	есть	$0.23	есть	Атака издалека, DoS, атака Сибиллы,	8 минут
LPoS	есть	$0.064	есть	Атака накоплением возраста монет, атака предвычислением, DoS	5 минут
PoC	есть	$0.012	есть	атака накоплением возраста монет, атака	7 минут

Продолжение таблицы 1.1

Алгоритм консенсуса	Децентрализация	Стоимость транзакции	Наличие математической задачи	Возможные атаки	Скорость проведения транзакции
				предвычислением, DoS	
PoI	есть	$0.0031	есть	Эгоистичный майнинг, DoS	3 минуты
PoA	нет	$0.39	есть	Атака издалека, атака Сибиллы,	1 минута
PoAuthority	есть	$2.86	нет	Атака накоплением возраста монет, атака предвычислением, DoS	6 минут
PoB	есть	$1.01	есть	Атака предвычислением, атака Сибиллы, DoS	3 минуты

Из сравнительной таблицы можно сделать вывод, что каждый из алгоритмов имеет ряд недостатков увеличивающие время, сложность и стоимость работы [2].

1.2 Формализация требований к новому программному продукту

Исходя из обзора моделей и их анализа, можно составить список требований к разрабатываемому алгоритму. Алгоритм консенсуса должен быть:

- четким [5];
- энергоэффективным;
- с низкой стоимостью проведения транзакций;
- с большой пропускной способностью;
- простым и стойким;

- низкое использование ресурсов: памяти и процессорного времени
- без майнинга;
- масштабируемым.

Полученный список требований был скомпонован в следующую модель информационной системы.

Система должна:

- обеспечивать консенсус узлов сети при помощи простой логики (для простоты разработки и корректировки);
- принимать и добавлять в цепочку блоков транзакции без дополнительных неоправданных расходов ресурсов;
- формировать цепочку транзакций, а не блоков (что повысит скорость и количество проведения транзакций).

Предлагаемый алгоритм не подразумевает связь одного узла сети со всеми другими напрямую для передачи транзакции по сети, поскольку скорость распространения транзакции в сети значительно выше, чем задержка, которую каждый узел будет испытывать при обмене данными. Для программной реализации данного алгоритма была выбрана платформа/язык Node.js. Выбор обосновывался из-за значительного прироста в скорости разработки, а также значительных готовых модулях для асинхронной сетевой работы и удобных модулях организации ответа узлов сети в формате JSON.

Выводы по главе 1

Были проанализированы существующие алгоритмы консенсуса. В каждом алгоритме был рассмотрен принцип работы, поддерживаемые платформы блокчейн, система вознаграждения, достоинства и недостатки алгоритмов консенсуса.

Проведённый анализ существующих алгоритмов показал, что ни одно из рассмотренных программных средств полностью не удовлетворяет требованиям разрабатываемого сервиса.

Исходя из полученных данных был сформулирован список требований к разрабатываемому алгоритму.

2 ПРОЕКТИРОВАНИЕ АЛГОРИТМА ДОСТИЖЕНИЯ КОНСЕНСУСА В ПИРИНГОВОЙ СЕТИ

2.1 Проектирование взаимодействия пользователя с узлом

Основной задачей сети криптовалюты является создание и подтверждение транзакций. В связи с этим рассмотрим процедуру создания транзакции и все взаимодействия в системе связанных с этим процессом [12].

Сначала пользователь формирует транзакцию, которая состоит из:

- получателя транзакции;
- суммы пересылаемой денежной единицы.

После этого пользователь, пользуясь списком доверенных нод, ищет любую свободную честную ноду. Список доверенных нод – это информация о каждой ноде, которая когда-либо появилась в сети. Список состоит из трех параметров:

- адрес ноды в виде ее приватного ключа;
- статус доверия: доверенная или не доверенная;
- статус нахождения в сети: online или ofline.

Если сеть считает ноду не доверенной, то отправленная пользователем транзакция на этот адрес никогда не подтвердится, а, следовательно, не попадет в блокчейн. Это связано с тем, что сеть не будет взаимодействовать сданной нодой и не будет принимать любые транзакции от данного узла [8].

Каждый пользователь имеет два ключа шифрования: публичный и приватный. Первый является адресом кошелька. Именно он передается другим пользователям для отправки транзакций на свой кошелек. Второй ключ – это ключ доступа к адресу, на котором находится криптовалюта. Приватный ключ представляет собой случайную строку, генерируемую клиентом.

Поскольку используемый криптографический алгоритм подразумевает выполнение определенного условия в приватном ключе, происходит проверка приватного ключа на соответствие условиям при помощи функции privateKeyVerify библиотеки secp256k.

Публичный и приватный ключи связаны математическим аппаратом друг с другом. Эта связь достигается благодаря использования эллиптической кривой и ее основных свойств.

На рисунках 2.1 представлен процесс подписи и отправки транзакции от пользователя к ноде.

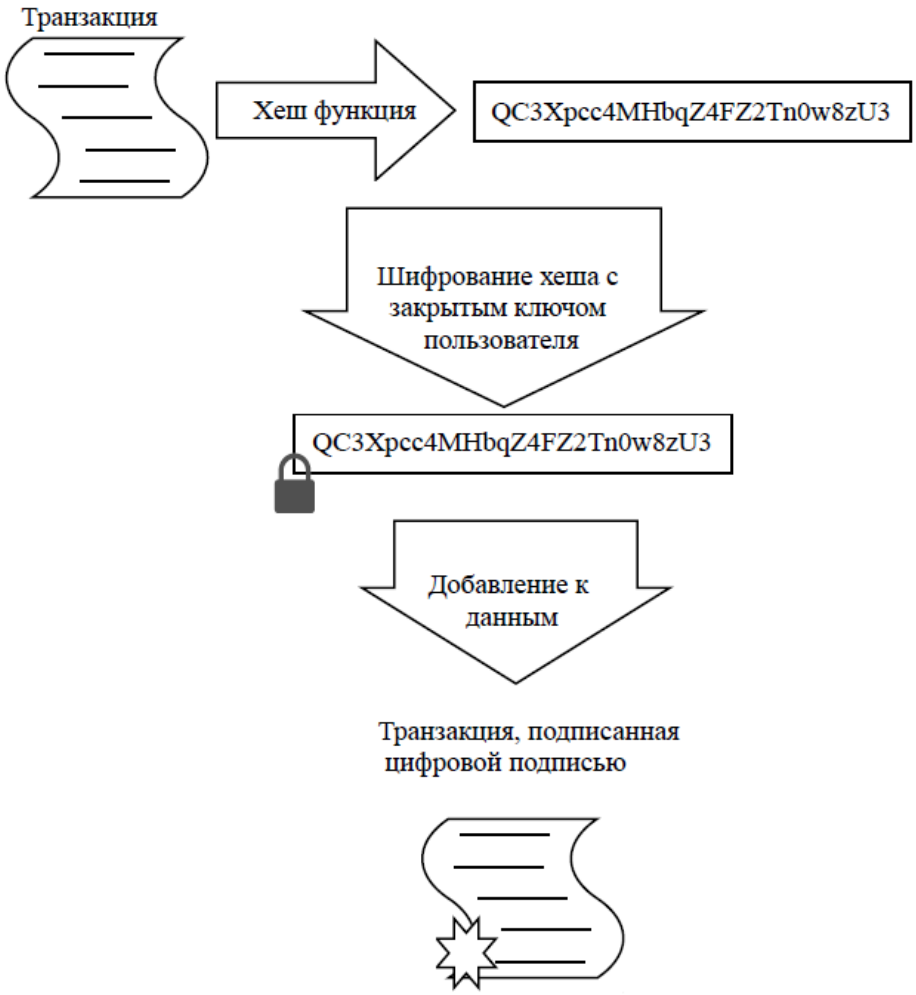

Рисунок 2.1 – Подписывание цифровой подписью

На рисунках 2.2 представлен процесс проверки ноды цифровой подписи, отправленной пользователем транзакции.

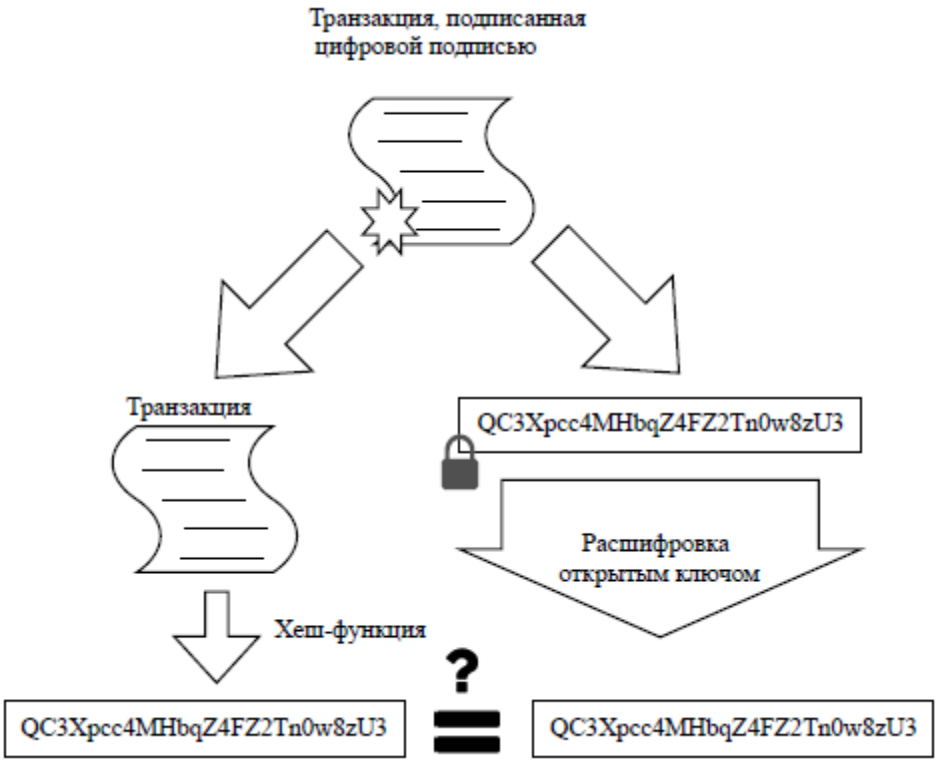

Рисунок 2.2 – Проверка цифровой подписью

Пользователь подписывает транзакцию и отправляет на узел. Процесс подписи состоит в следующем:

- при помощи хеш-функцию $hash()$ находиться хеш от транзакции s (формула 2.1);

$$h = hash(s) \tag{2.1}$$

- полученный результат шифруется приватным ключом (формула 2.2);

$$c = E(h) \tag{2.2}$$

- c добавляется к транзакции.

Как только узел получил подписанную транзакцию, начинается проверка подписи:

- транзакция и подпись разъединяется;
- от транзакцию s′ при помощи хеш-функции $hash()$ находится хеш (формула 2.3);

$$h' = hash(s') \tag{2.3}$$

- подпись расшифровывается публичным ключом (формула 2.4);

$$m = D(c) \tag{2.4}$$

- проверяет равенство хеша от транзакции с расшифровывания подписью (формула 2.5).

$$h' = m \tag{2.5}$$

При несовпадении, узел считает пользователя злоумышленником и отказывает ему в проведении транзакции.

2.2 Проектирование алгоритма достижения консенсуса

Каждая нода состоит из следующих компонентов:

- таблица ожидающих подтверждения транзакций;
- таблица подтвержденных сетью транзакций;
- блокчейн;
- список известных нод;
- приватный и публичный ключ;
- таблица счетов.

После того, как равенство хешей от транзакции с расшифровывания подписью совпали, нода записывает полученный результат в таблицу ожидающих подтверждения и подписывает результат совей электронной подписью. Нода может принять транзакцию или не принять, в зависимости от таблицы счетов (рисунок 2.3).

Адрес кошелька	Баланс
x9bgxLvn+0leM2Yxdj1+yG0Gbf0MITKH8e8CAwEAAQJAOMkabChHun+ImeUb	1023
TD07OkXFvHpNrnhxQwF577CeY7qdYbe7YgXYP0Mz4zOUVGzuGN1v7M/27tJ	541
YQIhAP35IIPLtQ4kAtzz/PBL2muxw02KuKk2J5KWayuUgAq/AiEArGJjIEhYVh4w	678
SWMPBBd8C3diDCkviwesi3E/czYn1NECIQCDOWOpdWmZYtXv4ElCx46VsoW	924
n+iWfw3kuJfn9QIgONt6tnsmJ8FtS8dQfJkCYQXPYRN0ivbNAnEn7GhAWwECI	113
QQIhAOJEA6XiHXpK/rOPtXvSA+4HQZe97Rj3vs65Gs6rigqFAiEAm7UkQ1K6CpI	774
MIIBOgIBAAJBALIuXyvnyTZowNu8CWMKC/nBB1TgSmbeCNAOQgw5n8PONc	652
JOH8UbW9qc4+p/ppA/0hdHlNX57RpZ7IkUMCAwEAAQJAdzoaStm9RSARNLd	1004
iAqG6l9uyah3AykXPvDsG+tkfHznMXmLUknWx0B90lvDEKptkV8Rp9VIWWLfL	374
6QIhAP/3ylEfAf0bSnoIGTSHED+xBq2PEY2cZ5xSa2ZD/md/AiEAsjQWKx2+h+r/	924
kiUk4wKIVB7sfC11ZUjLucpwh7mKGD0CIQDnS62R6AcQA50k8DR2dqWEKB70	545
c35cj01lN4OnWQIgZW8wjypHIaaqAs4RyQXQA9H2Wlil0hk1nXOVyivaZyECID	635
TfE8c0CWDnIqmIbJFBs57J4MxWxpZT6oB/L8psd/Ltc8j838960aIHmOxufo0Yt	787
7Yct+2eZ0GaJCPEvyuq3Nq7M6sJ2fO43qZkCAwEAAQJASwaHc5RtLlG43ewu	458
dygXIBPQ/NzQjP2WP12sd8K/EbE7PNIE31q8wf2LTCs9+6ipZevNjJLdoCKfvkU	1404
gQIhAPgKGifmv37fYwsPthDJOXqiLnPtDXaAEkuJBeRrFp+JAiEAzcOB6+Uqt59	244
TxLEQLkSSsHSPDG6CCx7jS8encv0pZECIHOiegkqGRJz4D+y5mUy+noCi/d9hy4	257
jMrsKXQQvK5JAiAFm4Bc51/kpSj6+q3yfZisBOAv18MUGfI8BL622ioOoQIhAJ9	364
mXvpv1HbpL62z7VVRWQiAWF1sqT1rMA4PVUCAwEAAQJA7pOsKAKez/MKm	8244

Рисунок 2.3 - Таблица счетов

Данная таблица состоит из адреса кошелька и баланса на нем. Она необходима, чтобы для подтверждения транзакции не нужно было проходиться по всему блокчейну для отслеживания актуального счета каждого пользователя. Каждая нода ведет данный список с момента появления online и если появляется транзакция пользователя, которого нет в таблице, информация о его счете ищется и проверяется в блокчейне, до генерации системой криптовалюты. Если данный пользователь есть в таблице, то информация считается достоверной и данные берутся из нее.

Таблица ожидающих подтверждения состоит из:

- адреса отправившего пользователя (публичный ключ);
- хеш транзакции, зашифрованный приватным ключом пользователя;
- адреса, на который отправлена транзакция;

- суммы транзакции;
- адреса ноды, записавшей решение;
- решения о принятии транзакции;
- хеш от всех выше сказанных значениях, зашифрованный приватным ключом ноды.

После того, как больше 51% сети ответит «подтверждаю» о конкретной транзакции, она записывается в таблицу подтвержденных сетью транзакций. Иначе если больше 51% ответит «не подтверждаю», то транзакция удаляется из таблицы ожидающих подтверждения.

В таблице подтвержденных сетью над каждой транзакцией пишется список нод, которые записали ее в данную таблицу. Как только все доверенные узлы выполнят данное действие, транзакция записывается в блокчейн. В случае, если в момент записи будет более одной подтвержденной транзакции, происходит сортировка по хеш-значению и записи в порядке возрастания.

2.3 Реализация обмена недостающих адресов между нодами

Нода проходится по списку известных нод и всем, кто отмечен как доверенный и online отправляется запрос, проверяющий работает узел или нет. Если приходит отрицательный ответ, то этот адрес отмечается как ofline.

Если узел работает, то сначала происходит обмен всеми известными адресами. Это позволяет добавить новые, заблокировать лживые и отметить ofline узлы. Рассмотрим более подробно процесс обмена адресов (рисунок 2.4).

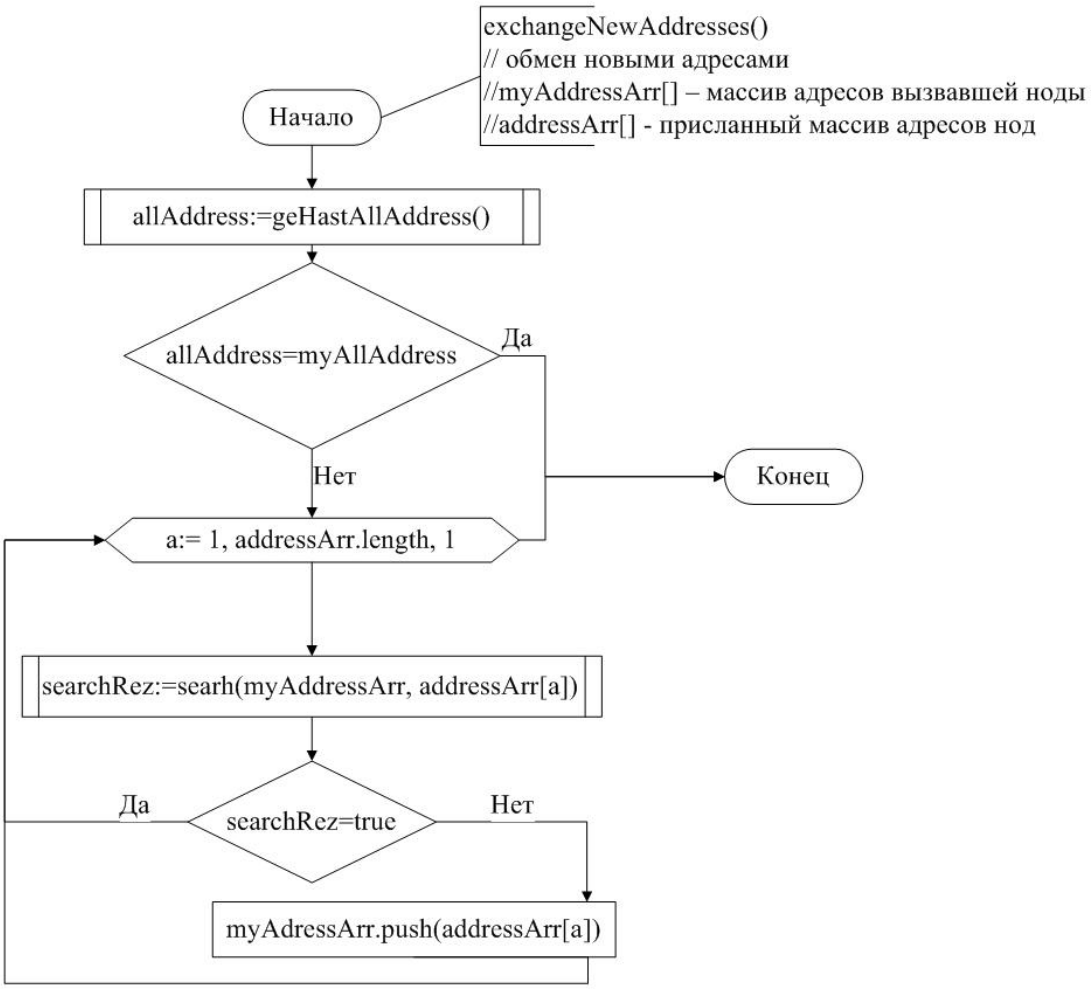

Рисунок 2.4 – Блок-схема обмена новыми адресами

Изначально сравнивается хеш-значения всех известных адресов. Благодаря использованию данного вида проверки было уменьшено время проверки и количество передаваемого трафика [3]. Если эти значения равны, то списки тоже равны. Иначе есть какие-то отличия и незнающей ноде пересылается список всех известных адресов.

Получив новый список адресов, начинается проход по каждому из них и поиск их у себя [7]. Если адрес не находится, то адрес добавляется в список, иначе данный адрес пропускается. Поиск происходит при помощи алгоритма, представленного на блок-схеме (рисунок 2.5).

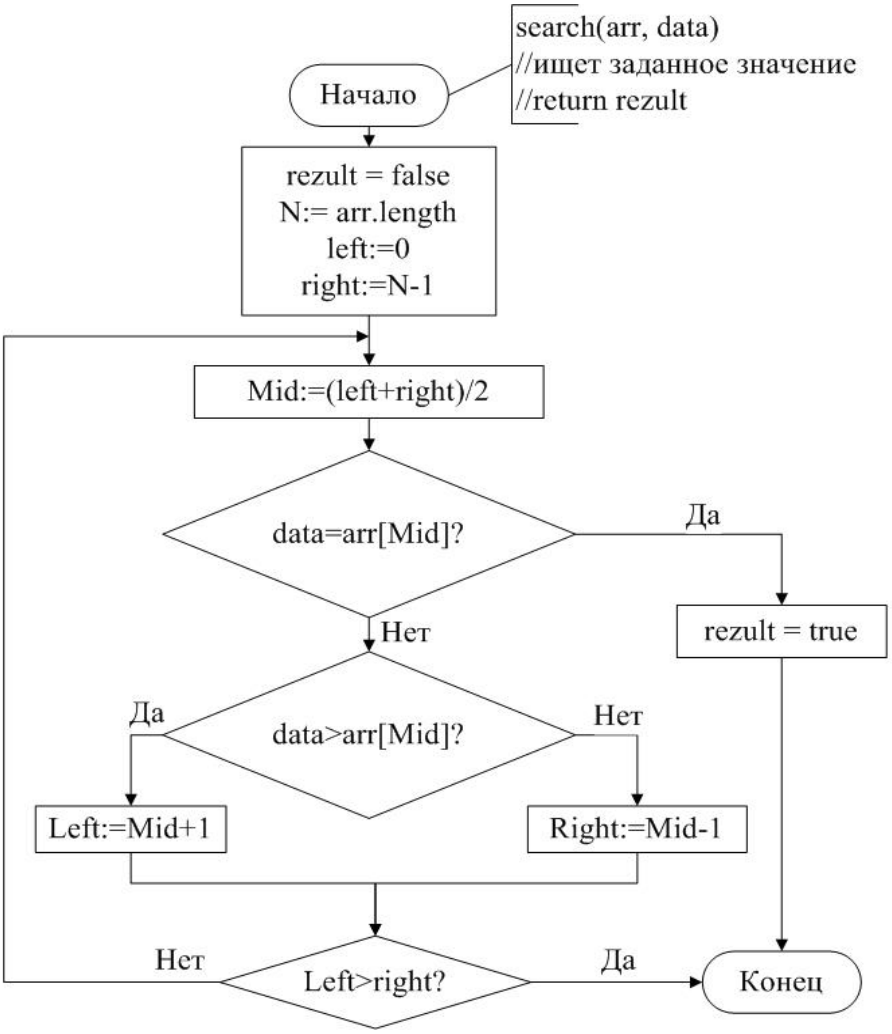

Рисунок 2.5 – Блок-схема поиска заданного значения в массиве

Функция search имеет два параметра на вход:

- arr – массив, в котором будет производиться поиск;
- data – значение, которое будет искаться в этом.

На выходе приходит булевое значение, обозначающее результат выполнения функции, где true показывает, что элемент найден, а false – нет.

Функция реализует алгоритм бинарного поиска, так же еще называемый методом деления отрезка пополам. Главное преимущества данного алгоритма –

это количество шагов, необходимых для поиска. Эта величина представлена в формуле 2.6.

$$c = \log_2 n, \tag{2.6}$$

где n - количество элементов.

2.4 Проектирование обмена недостающих транзакций в блокчейне

Следующим этапом ноды сравнивают номера последних транзакций в блокчейне. Если это число отличается, то узлу с наименьшим числом отправляются недостающие транзакции. Для поиска недостающих транзакций было выбрано несколько методов [4]:

- полиномиальное хеширование по модулю небольшого числа;
- построение дерева бора из хешей транзакций с выборкой вершины, в которой заканчивается наибольшее число хешей;
- сортировка хешей транзакций с текущего уровня строкового представления хеша;
- объединение хешей транзакций в цепочки заданной величины и нахождение хеша от них.

Первый метод предполагает оставлять транзакции с хешем, которые встречаются наибольшее число раз, а остальные - не принимать во внимание. Этот метод имеет недочет в виду вероятности коллизий, в результате чего может потребоваться дополнительный цикл поиска. В виду этого время работы алгоритма можно считать стремящимся к линейному.

Вторым методом является построение дерева бора, хеш транзакций с выборкой вершины, в которой заканчивается наибольшее число хешей. Поскольку происходит умножение на размер используемого алфавита, время

работы алгоритма получается гораздо больше, чем при полиномальном хешировании

Третий алгоритм представляется наиболее простым, но появляется дополнительная логарифмическая зависимость времени [7].

Четвертый алгоритм показал наибольшую скорость поиска в виду простоты. Хотя в методе присутствует наличие ошибки равной длине цепочки, это компенсируется скоростью нахождения ближайшей актуальной цепочки. Рассмотрим алгоритм более подробно на блок-схеме (рисунок 2.6).

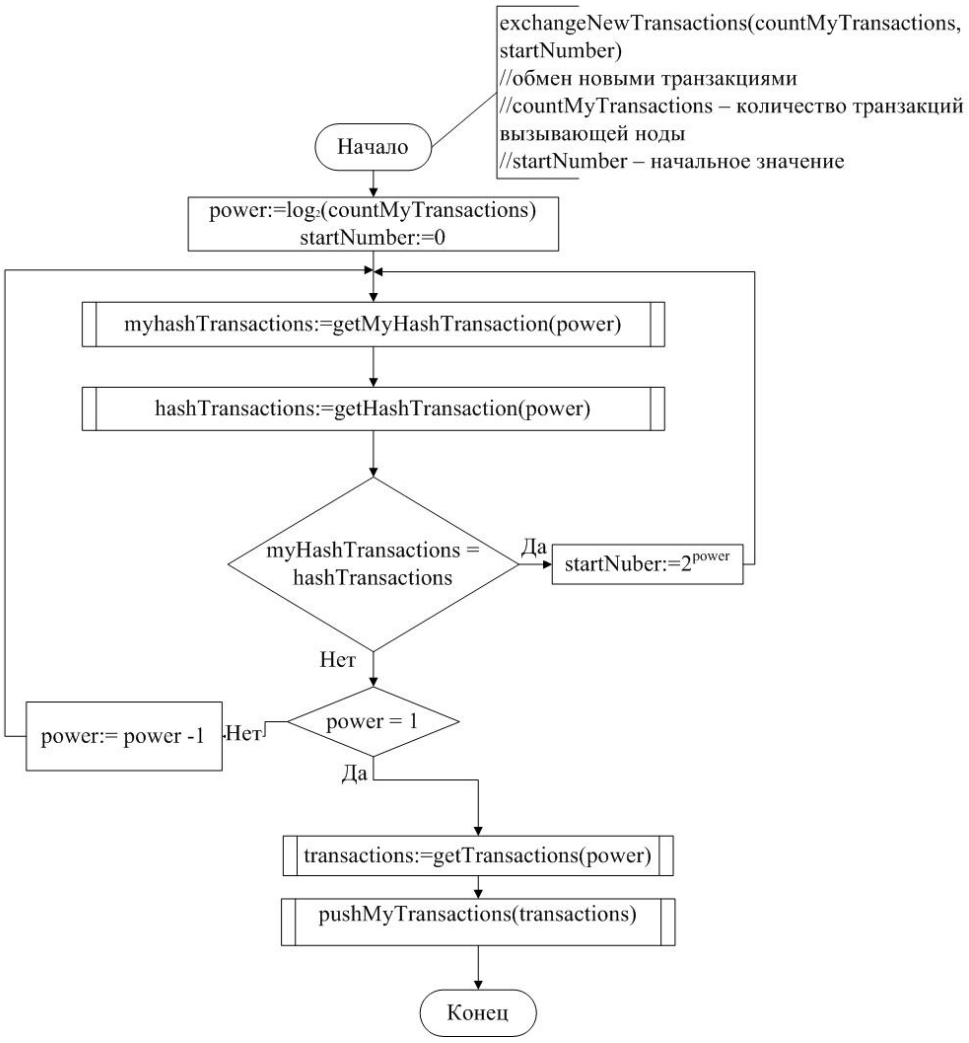

Рисунок 2.6 – Блок-схема обмена новыми транзакциями

Сначала нода определяет количество блоков в блокчейне в степени двойки. Если число не выражается без остатка, то всегда округляется в большую сторону. Полученное число характеризует количество элементов от начала блокчейна. Посчитав хеш от всех элементов, нода запрашивает тоже значение у подключенной к ней ноде и отправляет ей свое.

В связи с тем, что значения хеш-функции после стабилизации блокчейна для каждой степени двойки не будет меняться, было принято решение хранить эти значения в виде таблицы (рисунок 2.7).

Степень двойки	Хеш-значение
1	MIICXwIBAAKBgQC3Xpcc4MHbqZ4CmkKzeNC/XeoNAWftFZ2Tn0w8zU
2	Vlbq3Bp8mGzM/v/emJ2rRkM3YyaTw+1q6vLxS7kVmqNPjctu3UskJ1kXLa
3	MygMn+CFJdMwUMLeqja2tD34Qk5qrqeoRSvsjmomFgYVrnKOlPg0bVQ
4	AoGBAKZISrRPRLyTlcw6Yx4pAvNmkKX2jm26WkjR/eptKokqS9G+hz8d
5	2j4TR6htsRzt/GovqX36AEQihuaqGlmKU1IbunGSpsNu0gNVSs2sCFKyRaN
6	8jh048oiRPay9TSVgCgAOCc9zY0MeliB5TqzhM/pX+k7L0PhAkEA8nCza8
7	vXJdGapTcNcLHZISffu+Y1eVt+mddzbShUQMVAxzjuhxYguNcpq9w3nYut
8	oZyByq9SiwJBAMGgGtRG+BUkKMZOy+fUePpoVbHY5c65aDHLeo5Sl0X
9	1HlqPZB2VpGoLn/rmke5TDP2ftmxwI+0kAkCQQDxuPq7iiF9XTmIBgHbeB
10	x5mWcS0noYkOgOfWmBuYcfuUxYFmSE3cQsZ6MNzpjJBR2xwRY1w7D
11	kp+HMZC8dGX5apJo3pmLMzSnEBJLlAva1zpbCEOKwjAvfyKgGdULOR
12	Hcpv/jhXeFIFJRzx0ODO4QJBAOVXEjr7ryNaDyUr5t0Ij5n/MSKVJrAHeol
13	6h9etCm/PAB4k+j0Y5S76U1y457M2P6f23JQ5iN/pks9PdG/bXG0LM3UW
14	hD3KBFvKmgq4q8u5wYMJLaluM4lLjCr0pyYoKA3DETKqyNYOwO/956
15	AoGAYIEw8R1tJo7L6WxMD823j3J2BN3f0aHts4aocsZGmz6ZEduH3GRU/
16	CnTrL2XmRAiPtefphc2qsixt+OiJzZT380t03a5vUSLGy0DnPdWTyzsLCmipe

Рисунок 2.7 – Хранение хеш-значений от каждой степени двойки

Если хеши отличаются, то степень двойки уменьшается на единицу. Этот процесс повторяется до тех пор, пока эти хеш-значения не будут равны или степень двойки будет равна 1 (рисунок 2.8).

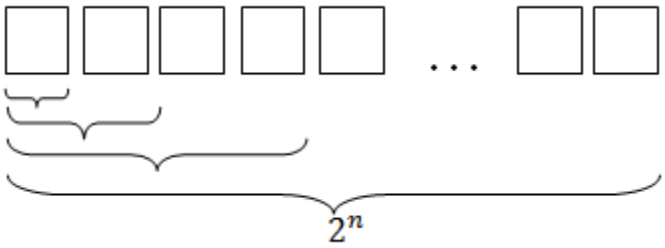

Рисунок 2.8 – Первая часть алгоритма

После этого мы сменяем точку отсчета на номер блока самого дальнего элемента из получившегося диапазона, и высчитываем в какой степени двойки можно измерить количество элементов после этого числа (рисунок 2.9).

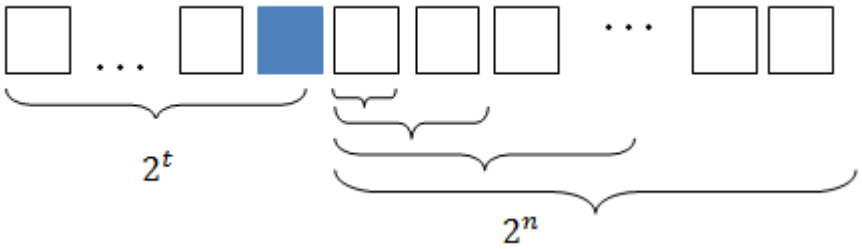

Рисунок 2.9 – Вторая часть алгоритма

Данная процедура повторяется до тех пор, пока не найдется последний повторяющийся элемент. Все значения, которые находятся правее выбранного диапазона, отправляются ноде.

Транзакции, находящие после последнего повторяющегося элемента, формируются в массив и соединяются с присланными транзакциями другой ноды. Этот массив сортируется без повторений по хеш-значению и записывается в блокчейн.

Затем идет сравнение результатов таблиц: подтвержденных сетью и ожидающих подтверждения транзакций. В связи с тем, что ноды подписывают электронной подписью каждое свое действие в таблицах, нет необходимости

каждому узлу общаться с каждым узлом. Ближайший сосед отправляет все известные ему действия других нод, и принимающему узлу для удостоверения правдивости информации необходимо расшифровать электронную подпись. Данный подход позволяет распространять информацию в геометрической прогрессии [7].

Вывод к главе 2

Во второй главе проведено проектирование пиринговой сети и алгоритма распределения данных между элементами сети, гарантирующий безопасность передачи данных.

Данная система позволяет защититься от известных возможных атак, обеспечить отсутствие майнинга, а также высокую скорость проведения транзакции. Минусом данного алгоритма являются задержки сети, но в виду того, что транзакция получает распространение по сети в геометрической прогрессии, скорость добавления транзакции в общую цепочку возрастает, а доступность данных каждой ноды гарантирует ее быстрое распространение и позволяет реализовать проверку добавления транзакции в список транзакций, ожидающих добавления в основную цепочку.

Разработанная система не подразумевает сложных просчетов и требует лишь достаточного для обмена соединения с сетью Интернет. Количество доступной оперативной связи и вычислительные мощности не играют значительной роли, поскольку отсутствует потребность в них. При недостатке ресурсов узел будет лишь медленнее отвечать на запросы, что не влияет на работоспособность остальной сети, поскольку нет прямой зависимости скорости проведения транзакции от скорости конкретного узла.

3 ТЕСТИРОВАНИЕ РАЗРАБОТАННЫХ ПРОГРАММНЫХ РЕШЕНИИ

3.1 Проведение вычислительных экспериментов

Ключевыми параметрами работы платежной системы является скорость и защищенность процесса проведения транзакции [19]. Для принятия решения о результате разработанной системы необходимо провести ряд тестов, которые позволят проверить работоспособность в реальных условиях.

Для оценки скорости проведения транзакций было разработано программное решение, позволяющее эмулировать работу сети с разработанным алгоритмом. В эмуляторе были реализованы возможности создания случайного количества узлов сети, в которую отправляется отправка одной транзакции. Измеряется время достижения полного консенсуса, то есть момент, когда транзакция будет записана в блокчейн.

В результате тестирования были выбраны несколько случайных моделей сети, которые стали контрольной выборкой для проведения тестирования сети при различных параметрах. На рисунке 3.1 представлены результаты тестирования.

Количество нод	Время, мкс
5	0,5
10	15,3
15	16,8
20	21,3
25	29,8
30	33,7
35	36,6
40	42,5
45	49,7
50	50,3

Рисунок 3.1 - Результаты тестирования

На рисунке показано среднее время достижения консенсуса при фиксированном количестве нод. В связи с тем, что вычислительные способности не позволяют эмулировать сеть более больших размеров, полученные данные были аппроксимированы для прогнозирования результатов работы системы. Для аппроксимации применялись следующие методы:

- линейная аппроксимация;
- экспоненциальная;
- логарифмическая;
- полиномиальная третей степени;
- степенная.

Для выбора метода, была рассчитана величина достоверности аппроксимации R^2 (рисунок 3.2).

Таблица 3.1 – Расчет величины достоверности аппроксимации

Метод аппроксимация	Величина достоверности аппроксимации R^2
линейная	0,9705
экспоненциальная	0,5698
логарифмическая	0,9469
полиномиальная третей степени	0,9810
степенная	0,8074

Наибольшую величину достоверности аппроксимации показал метод квадратичной аппроксимации. Для дальнейших вычислений будем использовать полученный метод.

Расширение сети можно выразить через формулу 3.1.

$$y = 0{,}0294x^3 - 3{,}2191x^2 + 204{,}62x - 676{,}67, \quad (3.1)$$

где x - количество нод в сети,

y - время достижения полного консенсуса.

Результаты аппроксимации представлены на рисунке 3.3.

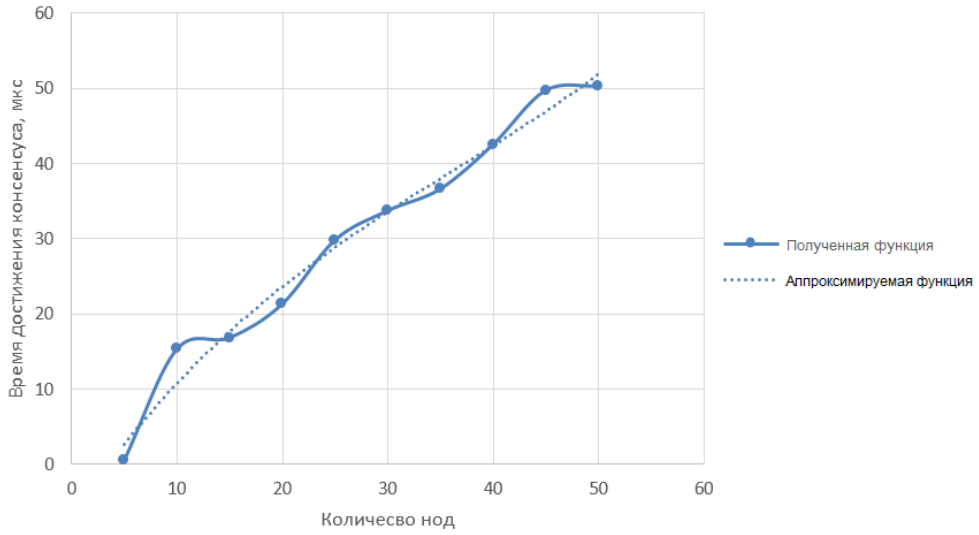

Рисунок 3.3 - Аппроксимируемая функция расширения сети

Достроив полученную функцию вперед, спрогнозируем поведение сети при ее расширении (рисунок 3.4).

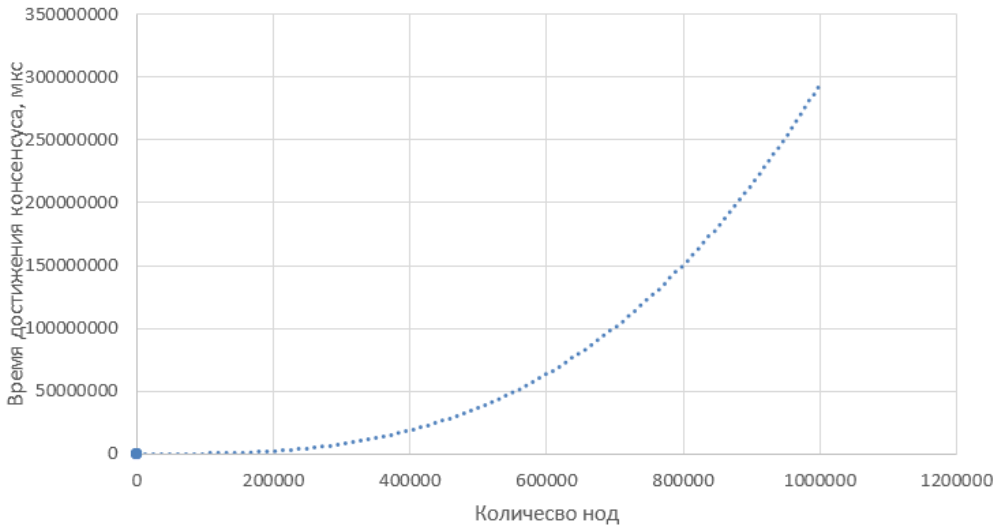

Рисунок 3.3 – Прогнозирование поведение сети при ее расширении

На графике видно, что при увеличении сети до 1 миллиона нод, скорость достижения консенсуса равна 300 000 000 микросекунд, что составляет 5 минут. В свою очередь в сети Bitcoin при данных условиях среднее время транзакции варьируется в районе 10 минут [20]. Это доказывает конкурентоспособность разработанной системы.

При увеличении количества транзакций на фиксированное число нод был получен следующий график (рисунок 3.4) изменения времени.

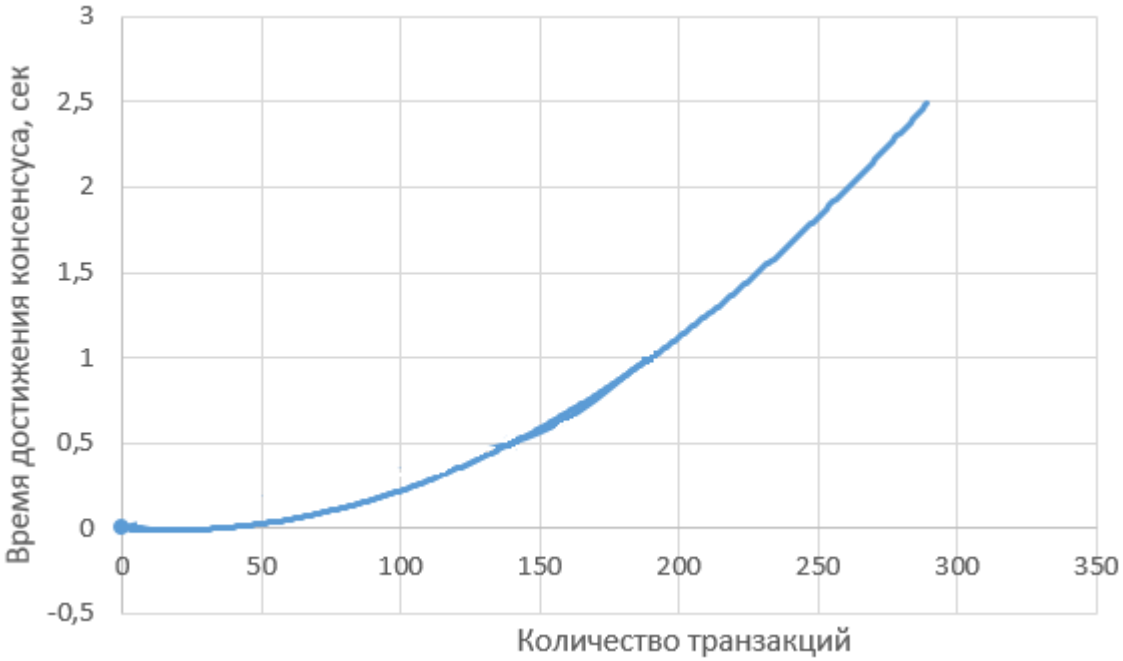

Рисунок 3.4 – Рост времени от увеличения количества транзакций

Из графика видно, что с увлечением количество транзакций время необходимое для нахождения консенсуса растет.

3.2 Корректировка разработанной информационной технологии для улучшения результатов

Алгоритм консенсуса предусматривает передачу данных между нодами в формате JSON. Это позволяет не тратить время на то, чтобы преобразовать данные необходимые для работы Node.js. Это связано с тем, что JSON основан на одном языке с Node.js JavaScript, и не требуется переформатировать данные для использования.

В попытках усовершенствования алгоритма была выдвинута гипотеза: для увеличения скорости передачи транзакций, данные необходимо сжимать [9].

Было проанализировано четыре утилиты для архивации и проведены сравнения между ними, для нахождения самого оптимального решения:

- 7-zip;
- FreeArc;
- WinRAR;
- WinZip 14.

В контрольном примере сжимались и передавались транзакции общим объемом 100 Мбайт. На рисунке 3.5 показан процент сжатия транзакций.

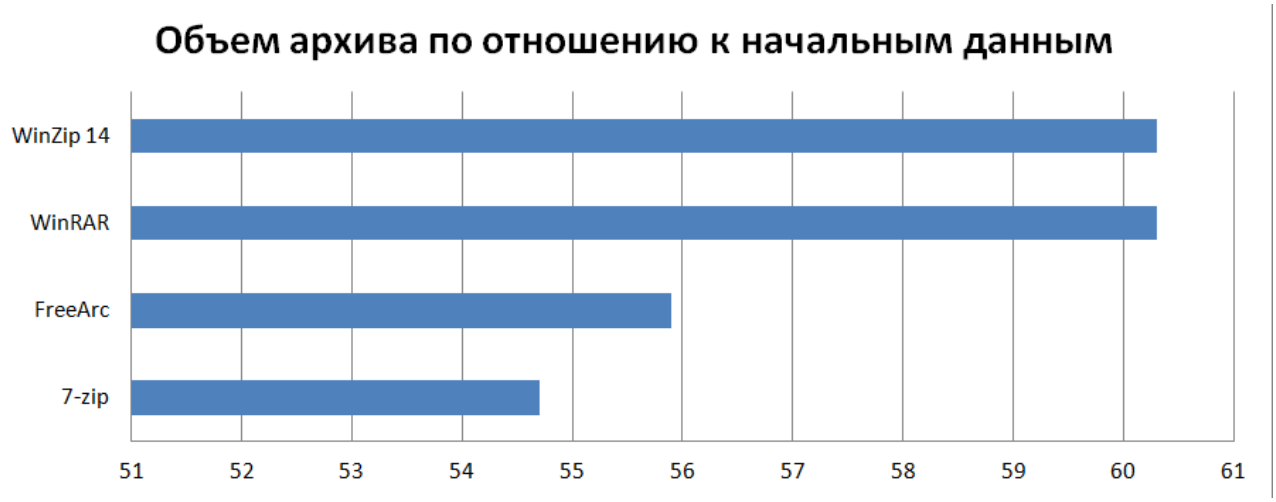

Рисунок 3.5 – Объем архива по отношению к начальным данным

На рисунке 3.6 изображено сравнение утилит по времени, потраченному на сжатие.

Рисунок 3.6 – Время, потраченное на архивацию

Исходя из полученных данных, можно сделать вывод, что 7-zip является лучшим архиватором для сжатия транзакций. Он выигрывает по скорости и отстает на 5% от лучшего результата по сжатия, но данным отклонением можно пренебречь.

На рисунке 3.7 изображено отношение времени к увеличению транзакций, которые будут заархивированы.

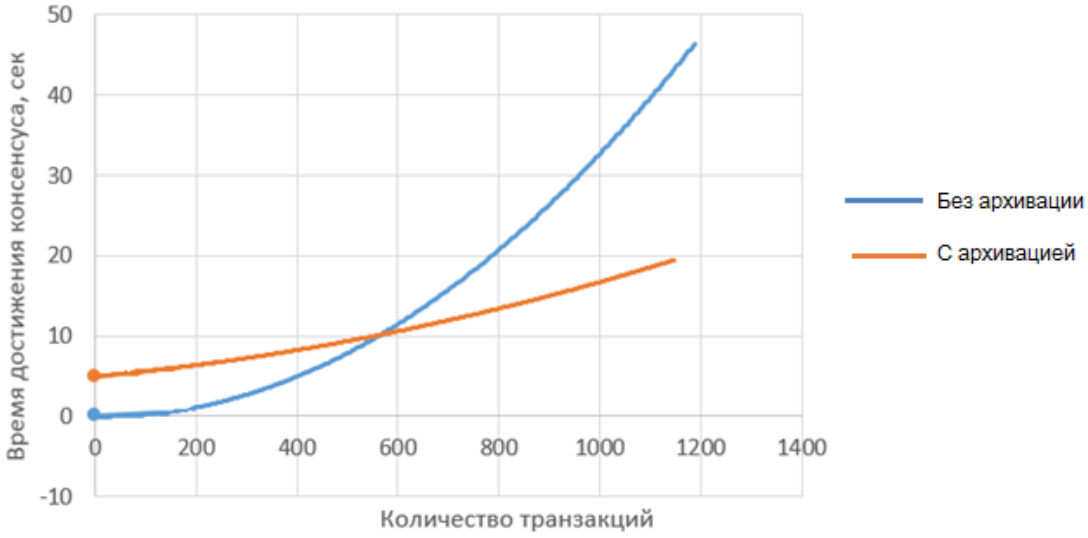

Рисунок 3.7 – Наложении графиков с архивацией и без нее

При наложении графиков видно, что сжатие становится эффективным при 550 транзакциях. Это можно объяснить тем, что архивация и разархивация занимают некоторое время и на маленьком количестве транзакций передача данных в чистом виде начинает выигрывать.

Вывод к главе 3

В третьей главе был разработан эмулятор работа сети, который позволяет измерить время достижения консенсуса при увеличении количества нод. В связи с тем, что физические возможности компьютера, на котором проводилось тестирование ограничены, эмпирические данные были аппроксимированны. Используя полученную функцию было рассчитано, что при увеличении сети до 1 миллион нод, скорость достижения консенсуса составляет 5 минут, в свою очередь в сети Bitcoin при данных условиях среднее время транзакции варьируется в районе 10 минут.

Так же была найдена зависимость времени достижения консенсуса относительно роста числа транзакций. Для увеличения скорости передачи транзакций была предложена архивация.

При поиске более подходящей утилиты сжатия был выбран 7-zip. Для архивированных данных была также высчитана зависимость времени достижения консенсуса относительно роста числа транзакций. При наложении графиков стало видно, что сжатие становиться эффективным при 550 транзакциях.

ЗАКЛЮЧЕНИЕ

Выполненная работа посвящена исследованию процесса, который гарантирует, что распределенные данные между элементами сети являются точными копиями. Цель работы определила ее основное направление – получение максимально возможной производительности одновременно выполняемых функций обработки данных посредством разработки алгоритма подтверждения корректности копий данных между элементами сети.

Для этого были рассмотрены и проанализированы современные децентрализованные системы и используемые ими модели достижения консенсуса. В ходе анализа были выявлены недостатки существующих технологий и определены пути решения выявленных проблем.

В процессе разработки были реализованы:

- алгоритм записи и распределения транзакций;
- механизм отправки транзакций пользователем;
- обновления списка адресов активных нод.

Было предложено и проанализировано несколько способов поиска транзакций у других нод.

Данная система позволяет защититься от известных возможных атак, обеспечить отсутствие майнинга, а также высокую скорость проведения транзакции. Система не подразумевает сложных просчетов и требует лишь достаточного для обмена соединения с сетью Интернет. Количество доступной оперативной связи и вычислительные мощности не играют значительной роли, поскольку отсутствует потребность в них.

Полученные алгоритмы были протестированы и была рассчитана их работоспособность при большом размере сети.

В результате работы была создана децентрализованная пиринговая система с реализованным алгоритмом подтверждения корректности копий данных между элементами сети. При увеличении сети до 1 миллион нод, скорость достижения консенсуса составляет 5 минут, в свою очередь в сети Bitcoin при данных условиях среднее время транзакции варьируется в районе 10 минут.

В попытках уменьшения скорости передачи транзакций, было произведено исследование, которое показало, что при архивации более 550 транзакций скорость передачи значительно увеличивается.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Левитин А. Алгоритмы: введение в разработку и анализ / А. Левитин, И. Красиков. – Москва: Вильямс, 2006. – 576 с.

2. Ализар А. Визуализация DDoS-атаки / А. Ализар // Хакер, 2013. – 10 с.

3. Арлоу Д. UML 2 и Унифицированный процесс/ Д. Арлоу, А. Нейштадт – СПб: Символ Плюс, 2007. – 624 с.

4. Волков Е. Численные методы / Е. Волков – Москва: Физматлит, 2003. – 248 с.

5. Вигерс К. Разработка требований к программному обеспечению / К. Вигерс, Д. Битти – СПб: BHV, 2014. – 736 с.

6. Финков М. Пиринговые сети / М. Финков – СПб: Наука и техника, 2006 – 272 с.

7. Дроздов С. Комбинаторные задачи и элементы теории вычислительной сложности / С. Дроздов – Таганрог: ТРТУ, 2000. – 58 с.

8. Кормен Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн – Москва: Вильямс, 2005. – 1296 с.

9. Гилл Ф. Практическая оптимизация / Ф. Гилл, У. Мюррей, М. Райт – Москва: Мир, 1985. – 509 с.

10. Таненбаум Э. Компьютерные сети. 5-е изд. / Э. Таненбаум, Д. Уэзеролл – СПб.: Питер, 2012. – 960 с.

11. Antonopoulos A. Mastering Bitcoin: Unlocking Digital Cryptocurrencies / A. Antonopoulos – Sebastopol: O'Reilly Media, 2014. – P. 298.

12. Burns B. Designing Distributed Systems / B. Burns – Sebastopol: O'Reilly Media, 2017 – P. 164.

13. Swan M. Blockchain: Blueprint for a New Economy / M. Swan – Sebastopol: O'Reilly Media, Inc, 2015 – P. 152.

14. Haddad S. Models and analysis in distributed systems / S. Haddad, F. Kordon, L. Pautet, L. Petrucci – London: Wiley, 2011. – P. 368.

15. Wattenhofer R. The Science of the Blockchain / R. Wattenhofer – Sebastopol: O'Reilly Media, 2016. – P. 124.

16. Обзор механизмов консенсуса блокчейна [Электронный ресурс]: https://ttrcoin.com/obzor-mehanizmov-konsensusa-blokcheyna.458/ (дата обращения 04.02.2017).

17. Реальная СТОИМОСТЬ транзакции BTC [Электронный ресурс]: https://golos.io/ru--stoimostx/@finansio/realnaya-stoimost-tranzakcii-btc-bitkoin-vy-budete-v-shoke (дата обращения 23.03.2017).

18. Стоимость транзакций в разных криптовалютах [Электронный ресурс]: http://cryptonews.one/kriptovaluta/stoimost-tranzakcijj-v-raznykh-kriptovaljutakh/ (дата обращения 24.03.2017).

19. Стратегии тестирования [Электронный ресурс]: http://www.4stud.info/software-construction-and-testing/lecture10.html#black-box (дата обращения 10.05.2017).

20. Bitinfocharts [Электронный ресурс]: https://bitinfocharts.com/ (дата обращения 29.04.2017).

21. Strict memory hard hash functions [Электронный ресурс]: https://bitslog.wordpress.com/2013/12/31/strict-memory-hard-hash-functions/ (дата обращения 24.09.2017).

22. Peer to Peer (P2P) [Электронный ресурс]: https://utmagazine.ru/posts/21310-peer-to-peer-r2r (дата обращения 14.10.2017).

23. The Proof-of-Work in Cryptocurrencies: Brief History. Part 1. [Электронный ресурс]: https://bytecoin.org/blog/proof-of-work (дата обращения 12.03.2018).

24. Burstcoin [Электронный ресурс]: https://www.burst-coin.org (дата обращения 15.03.2018).

25. Burstcoin [Электронный ресурс]: https://nem.io (дата обращения 13.03.2018).

26. Proof-of-authority – алгоритм консенсуса блокчейн, PoAuthority майнинг [Электронный ресурс]: https://profit-life.net/proof-of-authority-algoritm/ (дата обращения 19.04.2018).

ПРИЛОЖЕНИЕ А
Листинг кода

Содержание файла Peer.js

```
'use strict';
const path = require('path'),
url = require('url'),
util = require('util');
const bodyParser = require('body-parser'),
        eventEmitter2 = require('eventemitter2'),
        express = require('express'),
        flaschenpost = require('flaschenpost'),
        request = require('request'),
        requireAll = require('require-all'),
        sha1 = require('sha1');
const Endpoint = require('./Endpoint'),
         errors = require('./errors'),
         interval = require('./interval'),
         WellKnownPeers = require('./WellKnownPeers');
const EventEmitter2 = eventEmitter2.EventEmitter2;
const routes = requireAll(path.join(__dirname, 'routes'));
const logger = flaschenpost.getLogger();
const Peer = function (options) {
if (!options) {
    throw new Error('Options are missing.');
```

```
}
EventEmitter2.call(this, {
  wildcard: true,
  delimiter: '::'
});
this.protocol = options.protocol || 'https';
this.self = new Endpoint(options);
this.metadata = options.metadata || {};
this.successor = new Endpoint(options);
this.predecessor = new Endpoint(options);
this.successors = [];
this.fingers = [];
this.wellKnownPeers = new WellKnownPeers();
this.wellKnownPeers.add(this.self);
this.wellKnownPeers.add(options.wellKnownPeers || []);
this.handle = {};
this.app = express();
this.app.use(bodyParser.json());
this.app.post('/self', routes.self(this));
this.app.post('/status', routes.status(this));
this.app.post('/metadata', routes.metadata(this));
this.app.post('/successor', routes.successor(this));
this.app.post('/successors', routes.successors(this));
```

```
  this.app.post('/predecessor', routes.predecessor(this));
  this.app.post('/closest-preceding-finger', routes.closestPrecedingFinger(this));
  this.app.post('/find-predecessor', routes.findPredecessor(this));
  this.app.post('/find-successor', routes.findSuccessor(this));
  this.app.post('/join', routes.join(this));
  this.app.post('/notify', routes.notify(this));
  this.app.post('/stabilize', routes.stabilize(this));
  this.app.post('/fix-fingers', routes.fixFingers(this));
  this.app.post('/fix-successors', routes.fixSuccessors(this));
  this.app.post('/fix-predecessor', routes.fixPredecessor(this));
  this.app.post('/handle/:action', routes.handle(this));
};
util.inherits(Peer,  EventEmitter2);
Peer.prototype.remote = function (target) {
  if (!target) {
    throw new Error('Target is missing.');
  }
  if (!target.host) {
    throw new Error('Host is missing.');
  }
  if (!target.port) {
    throw new Error('Port is missing.');
  }
```

```
return {
  run: (fn, args, callback) => {
    if (!fn) {
      throw new Error('Function is missing.');
    }
    if (!args) {
      throw new Error('Callback is missing.');
    }
    if (!callback) {
      callback = args;
      args = {};
    }
    const targetAsUrl = url.format({
      protocol: this.protocol,
      hostname: target.host,
      port: target.port,
      pathname: fn
    });
    request.post(targetAsUrl, {
      body: args,
      json: true,
      keepAlive: true
    }, (err, res, body) => {
```

```
        if (err) {
          return callback(err);
        }
        res.resume();
        if (res.statusCode !== 200) {
          const errorSummary = {
            url: targetAsUrl,
            args,
            statusCode: res.statusCode,
            body: (body || '').trim('\n')
          };
          logger.warn('Failed to call a remote function.', errorSummary);
          return callback(new errors.UnexpectedStatusCode(`Unexpected status code ${res.statusCode} when running ${fn}.`, errorSummary));
        }
        callback(null, body);
      });
    }
  };
};
Peer.prototype.setSuccessor = function (successor) {
  if (!successor) {
    throw new Error('Successor is missing.');
```

```
  }
  const fromStatus = this.status();
  this.successor = new Endpoint({
    host: successor.host,
    port: successor.port
  });
  this.emit('environment::successor', this.successor);
  const toStatus = this.status();
  if (toStatus !== fromStatus) {
    this.emit(`status::${toStatus}`,  {
      from: fromStatus,
      to: toStatus
    });
  }
};
Peer.prototype.setPredecessor = function (predecessor) {
  const fromStatus = this.status();
  if (predecessor) {
    this.predecessor = new Endpoint({
      host: predecessor.host,
      port: predecessor.port
    });
  } else {
```

```
    this.predecessor = undefined;
  }
  this.emit('environment::predecessor', this.predecessor);
  const toStatus = this.status();
  if (toStatus !== fromStatus) {
    this.emit(`status::${toStatus}`, {
      from: fromStatus,
      to: toStatus
    });
  }
};
Peer.prototype.status = function () {
  if (!this.predecessor) {
    return 'unbalanced';
  }
  if ((this.self.id === this.successor.id) && (this.self.id === this.predecessor.id)) {
    return 'lonely';
  }
  if ((this.self.id !== this.successor.id) && (this.self.id !== this.predecessor.id)) {
    return 'joined';
  }
  return 'unbalanced';
};
```

```
Peer.prototype.fixSuccessor = function () {
  this.successors.shift();
  if (this.successors.length === 0) {
    return this.setSuccessor({
      host: this.self.host,
      port: this.self.port
    });
  }
  this.setSuccessor({
    host: this.successors[0].host,
    port: this.successors[0].port
  });
};
Peer.prototype.getEndpointFor = function (value, callback) {
  if (!value) {
    throw new Error('Value is missing.');
  }
  if (!callback) {
    throw new Error('Callback is missing.');
  }
  this.remote(this.self).run('find-successor', {
    id: sha1(value)
  }, (errFindSuccessor, successor) => {
```

```
    if (errFindSuccessor) {
      return callback(errFindSuccessor);
    }
    this.remote(successor).run('metadata', (errMetadata, metadata) => {
      if (errMetadata) {
        return callback(errMetadata);
      }
      callback(null, successor, metadata);
    });
  });
};
Peer.prototype.isResponsibleFor = function (value) {
  if (!value) {
    throw new Error('Value is missing.');
  }
  if (!this.predecessor) {
    return false;
  }
  const id = sha1(value);

  return interval({
    left: this.predecessor.id,
    right: this.self.id,
```

```
    type: 'leftopen'
  }).contains(id);
};
module.exports = Peer;
```

Содержание файла p2p.js

```
const http = require('http'),
https = require('https');
const _ = require('lodash'),
flaschenpost = require('flaschenpost'),
parse = require('parse-duration'),
Timer = require('timer2');
const Peer = require('./Peer');
const logger = flaschenpost.getLogger();
const p2p = {};
p2p.peer = function (options) {
if (!options) {
throw new Error('Options are missing.');
}
if (!options.privateKey && options.certificate) {
        throw new Error('Private key is missing.');
}
if (options.privateKey && !options.certificate) {
```

```
            throw new Error('Certificate is missing.');
}
const useHttps = !!options.privateKey && !!options.certificate;
options.protocol = useHttps ? 'https' : 'http';
const peer = new Peer(options);
const serviceInterval = parse(options.serviceInterval || '30s'),
wobbleFactor = serviceInterval * 0.5;
if (useHttps) {
        https.createServer({
        key: options.privateKey,
        cert: options.certificate
}, peer.app).listen(peer.self.port, () => {
              logger.info('Server started.', {
               endpoint: peer.self,
               status: peer.status()
            });
        });
} else {
        http.createServer(peer.app).listen(peer.self.port, () => {
        logger.info('Server started.', {
        endpoint: peer.self,
        status: peer.status()
        });
```

```
    });
}
[ 'stabilize', 'fix-successors', 'fix-fingers', 'fix-predecessor' ].forEach(fn => {
        new Timer(serviceInterval, { variation: wobbleFactor }).on('tick', () => {
                peer.remote(peer.self).run(fn, () => {});
        });
});
new Timer(serviceInterval, { variation: wobbleFactor }).on('tick', () => {
        peer.remote(peer.self).run('join', _.sample(peer.wellKnownPeers.get()), () =>
        {});
});
return peer;
};
```