Санкт-Петербургский государственный университет

Кафедра системного программирования
Программная инженерия

Федоров Роман Дмитриевич

Разработка имитации сценариев голосования в системе Blockchain benchmarking

Дипломная работа

Научный руководитель:
ст. преп. Я. А. Кириленко

Рецензент:
руководитель филиала "DSX Technologies" Г. В. Мавчун

Санкт-Петербург
2018

SAINT PETERSBURG STATE UNIVERSITY

Software Engineering

Roman Fedorov

Development of a voting scenarios simulation for the Blockchain benchmarking system

Graduation Thesis

Scientific supervisor:
Senior lecturer Iakov Kirilenko

Reviewer:
head of branch at "DSX Technologies" George Mavchun

Saint Petersburg
2018

Оглавление

Введение

Blockchain — реплицированная распределенная база данных особого вида, обеспечивающая высокую надежность хранения и добавления данных участниками, не доверяющими друг другу. Идея и первая реализация данной технологии была представлена Сатоши Накамото в 2009 году в криптовалюте Bitcoin [11]. Ключевой особенностью данной валюты стала возможность отправлять электронные деньги напрямую между участниками, без участия централизованного узла в сети, но исключив неверифицированные транзакции и предотвратив двойное расходование средств.

Технология Blockchain получила дальнейшее распространение и применение в сфере финансов и других сферах. Разработано большое количество различных криптовалют, использующих и добавляющих к реализации Bitcoin новые возможности. Так, например, в криптовалюте Ethereum была реализована идея Тьюринг-полных смарт-контрактов — формализованного описания договоров, исполняющихся автоматически [18].

Blockchain применяется не только в финансовой сфере, так, например, он активно используется в сфере логистики, делая цепи поставок товаров прозрачными, эффективными и проверяемыми [8]. Также одним из самых перспективных применений Blockchain считается использование распределенного реестра для проведения голосований, поскольку таким образом можно человеконезависимо подтверждать итоги, но сохранять тайну голосования [14] [15].

В данных примерах Blockchain применяется для хранения данных, однако, как упомянуто выше, существует множество различных реализаций, и выбор подходящей может быть затруднителен. Очевидно появляется проблема выбора подходящей реализации исходя из имеющихся ограничений. Основным недостатком Bitcoin считается количество обрабатываемых транзакций за единицу времени, что увеличивает комиссии за переводы и делает микротранзакции в сети слишком дорогими для пользователей. Существует несколько инструментов для те-

стирования производительности реализаций Blockchain, одним из которых является Blockchain benchmarking [4], разрабатываемый компанией DSX Technologies [3].

Выборы являются важным социальным явлением и изучением поведения избирателей занимаются многие ученые [16] [17]. Для симуляции голосования на Blockchain необходимо учитывать такие явления, как естественное изменение активности голосующих в период голосования, а также географическую распределенность участков для голосования и, соответственно, самих голосующих. К сожалению, текущие возможности систем для тестирования производительности реализаций Blockchain не позволяют гибко настраивать параметры для симуляции различных сценариев и, таким образом, не решают проблемы выбора реализации Blockchain для применения в голосованиях.

1. Постановка задачи

Целью данной работы является доработка системы Blockchain benchmarking для реализации возможности задания специфических сценариев голосования в тестировании производительности реализаций Blockchain.

Для достижения этой цели были поставлены следующие задачи.

- Изучить систему Blockchain benchmarking и её аналоги.
- Предложить и реализовать доработки, позволяющие гибко задавать сценарии голосования.
- Реализовать инструмент для анализа процесса имитации голосования.
- Произвести апробацию новой версии системы на конкретных примерах.

2. Обзор

2.1. О Blockchain

Blockchain является системой из многих участников, которые не полностью доверяют друг другу. Узлы сети вместе поддерживают глобальное состояние и выполняют транзакции, изменяющие это состояние, соглашаясь об общем состоянии транзакций и их порядке. Blockchain хранит полную историю всех изменений. Для удобства транзакции объединяются в блоки транзакций. Каждый блок имеет указатель на предыдущий, и хэш текущего блока зависит в том числе и от хэша предыдущего.

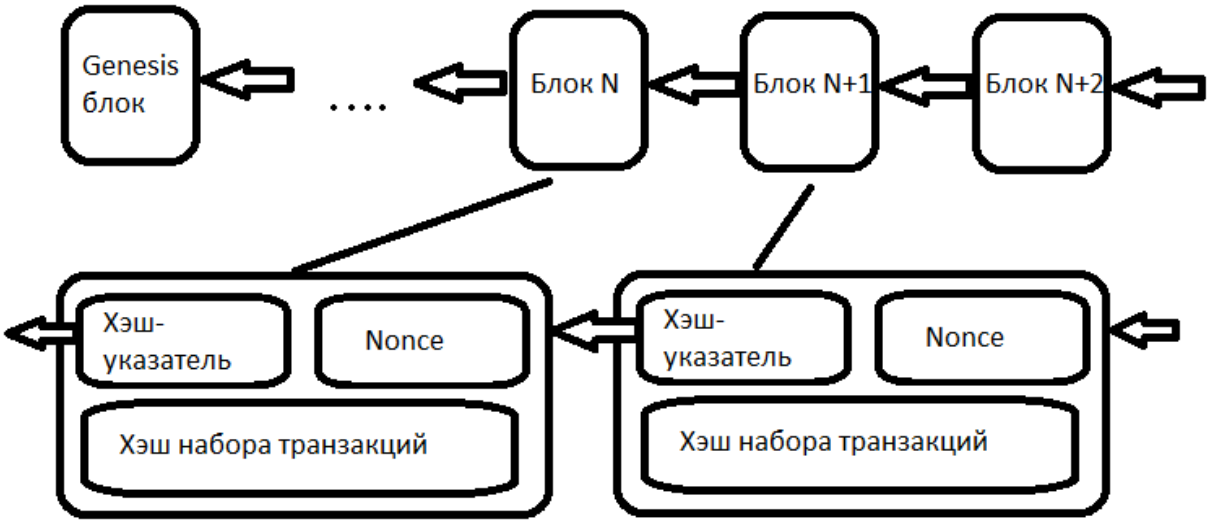

Рис. 1: Хранение данных в Blockchain

При попытке изменения содержимого блока изменится и его хэш, а значит нарушатся связи со всеми последующими блоками. Поэтому информацию в Blockchain считают неизменяемой до тех пор, пока злоумышленники не контролируют механизм создания новых блоков. В алгоритмах семейства Proof-of-Work, применяемых в наиболее известных реализациях Blockchain, таких как Bitcoin и Ethereum, для этого необходимо контролировать бóльшую часть вычислительных мощностей в сети. В таком случае атакующие могут вычислить альтернативную цепочку блоков, начиная с измененного блока, и подменить основную це-

почку на нее, поскольку узлы считают основной ветвь с наибольшим количеством блоков [11].

Сети Blockchain можно разделить на публичные и приватные. Публичные сети открыты для вхождения любых участников, такими сетями, например, являются криптовалюты (Bitcoin, Ethereum и другие). Более того, такие сети поощряют участников за присоединение и работу над верификацией транзакций, выплачивая вознаграждения за успешно созданные блоки. К приватной сети могут подключаться только узлы, имеющие разрешение или приглашение, таким образом поддерживается ограничение доступа к информации о транзакциях в сети. Разрабатываются реализации Blockchain, работающие только в приватном режиме, например, Hyperledger Fabric от Linux Foundation [7].

2.2. Инструменты сравнения производительности Blockchain

Поскольку Blockchain является достаточно молодой, но сложной технологией, то первые реализации инструментов для тестирования производительности появились в 2016 году [4]. Так как данные системы достаточно молоды, то можно предположить, что их функциональность покрывает лишь часть возможных сценариев тестирования производительности. И действительно, существующие инструменты не позволяют в полной мере выполнить цель, поставленную в данной работе, что раскрыто в дальнейшем обзоре.

2.2.1. Blockchain benchmarking

Blockchain benchmarking — проект по анализу производительности реализаций Blockchain, разрабатываемый компанией DSX Technologies с 2016 года [4]. Данный инструмент позволяет тестировать различные реализации Blockchain, получая информацию о пропускной способности и задержках в сети, потреблении вычислительных ресурсов узлами сети. Основными элементами приложения являются управляющий модуль, нагрузочные модули, логирующие модули. Проект поддерживает

такие реализации Blockchain как Ethereum, Hyperledger Fabric 0.6.

К преимуществам данного проекта можно отнести высокую точность собираемых данных (данные собираются для каждого отдельного узла), относительную простоту подключения новых реализаций Blockchain. К недостаткам стоит отнести недостаточное количество публично доступной документации, отсутствие гибкого задания нагрузочных сценариев (поддерживается только задание постоянного уровня нагрузки), отсутствие симуляции сетевых помех. Результаты тестирования сохраняются в формате текстовых файлов, существует частичная обработка сырых логов для получения основных результатов тестирования, однако отсутствует визуальное представление результатов тестирования, позволяющее более полно проводить анализ полученных результатов.

2.2.2. Hyperledger Caliper

Hyperledger — организация, разрабатывающая группу проектов, посвященных Blockchain, при поддержке Linux Foundation с 2015 года. Целью организации является разработка Blockchain решений с привлечением специалистов различных областей, с особым акцентом на производительность и надежность, для применения в крупных технологических, финансовых и иных корпорациях. Одним из основных проектов является Hyperledger Fabric — реализация Blockchain с открытым исходным кодом, но без запуска соответствующей криптовалюты, таким образом данный Blockchain является исключительно приватным.

Также одним из продуктов консорциума является Hyperledger Caliper — инструмент анализа производительности Blockchain [6]. Данное решение поддерживает реализации Fabric 1.0, Sawtooth 0.8, Iroha, позволяет собирать данные о проценте успешных транзакций, пропускной способности, задержках, потреблении ресурсов (процессора, памяти, сетевого соединения). Проект имеет достаточно подробную документацию, примеры тестовых конфигураций. После запуска тестов имеется возможность построить html-отчет с ключевыми параметрами для анализа в табличной форме.

К недостаткам данного инструмента можно отнести отсутствие гибкого задания нагрузочных сценариев (поддерживается только задание постоянного уровня нагрузки), ограниченное количество поддерживаемых реализаций Blockchain (все доступные реализации являются участниками консорциума Hyperledger), а также отсутствие симуляции сетевых помех.

2.2.3. Blockbench

Blockbench — проект по анализу производительности реализаций Blockchain, разрабатываемый на кафедре баз данных Национального университета Сингапура [1]. Проект использует особый подход к нагрузочному тестированию, так как ставит своей целью анализ исключительно приватных реализаций Blockchain с Тьюринг-полными смарт-контрактами. Исходя из этого, анализ делится на 4 основных части: изучение и сравнение уровня принятия решений о включении/невключении транзакций в блоки (уровень консенсуса), изучение модели данных блокчейна и сравнение эффективности различных моделей, изучение скорости исполнения смарт-контрактов, а также более общее сравнение приложений на Blockchain, использующих смарт-контракты. Каждую из частей тестирует отдельный смарт-контракт, написанный разработчиками проекта. В данном решении реализовано тестирование таких реализаций Blockchain как Ethereum, Parity, Hyperledger Fabric. После тестирования можно получить информацию о задержках и пропускной способности сети, потреблении ресурсов.

К достоинствам данного проекта относятся развитая документация (2 статьи и подробные описания на GitHub), поддержка симуляции разделения сети, тестирование производительности исполнения смарт-контрактов. К недостаткам можно отнести ограниченную применимость данного решения (поддерживаются только Blockchain с Тьюринг-полными смарт-контрактами, так, например, отсутствует возможность оценки производительности Bitcoin), сравнительно бóльшую трудоемкость включения новых реализаций Blockchain в тестирование (для них придется писать аналогичные реализации смарт-контрактов на языке, поддер-

живаемым конкретным Blockchain), а также недостаточную точность в сборе данных о работе сети Blockchain (отсутствие возможности просмотра времени появления конкретного блока на конкретном узле сети, таким образом невозможно узнать то, как блок был распространён по сети), а также отсутствие инструментов для анализа результатов тестирования.

2.2.4. Сравнительная таблица

	Blockchain benchmarking	Hyperledger Caliper	Blockbench
Публично доступная документация	Ограниченная документация	Достаточно подробная документация	Очень подробная документация
Поддерживаемые реализации Blockchain	Ethereum, Fabric 0.6	Ethereum, Parity, Fabric 0.6	Fabric 1.0, Sawtooth 0.8, Iroha
Область применимости	Полная	Полная	Ограниченная
Возможность тестирования смарт-контрактов	Отсутствует	Отсутствует	Присутствует
Задание гибкой генерации нагрузки	Отсутствует	Отсутствует	Написание отдельных смарт-контрактов
Поддержка симуляции проблем соединения в сети	Отсутствует	Отсутствует	Частичная

Получаемые результаты	Пропускная способность, задержки в сети с высокой точностью, потребление ресурсов	Пропускная способность, задержки в сети, потребление ресурсов	Пропускная способность, задержки в сети с низкой точностью, потребление ресурсов
Форма представления результатов тестирования	Упрощенная, текстовая	Табличная, основные показатели	Отсутствует

Таким образом, было решено усовершенствовать систему Blockchain benchmarking, реализовав возможность гибко задавать произвольные сценарии нагрузки, однако не потеряв универсальности данного решения, чтобы можно было переиспользовать один и тот же план нагрузки для тестирования различных реализаций Blockchain без необходимости реализовывать смарт-контракт для каждого нового Blockchain, а также реализовать симуляцию проблем сетевого соединения в тех же ограничениях. Кроме того, следовало улучшить пользовательскую документацию и реализовать инструмент представления подробных результатов тестирования в виде графиков.

3. Поддержка гибкого задания сценариев голосования

3.1. Архитектура системы

Система Blockchain benchmarking имеет модульную архитектуру, каждый модуль является Java-приложением.

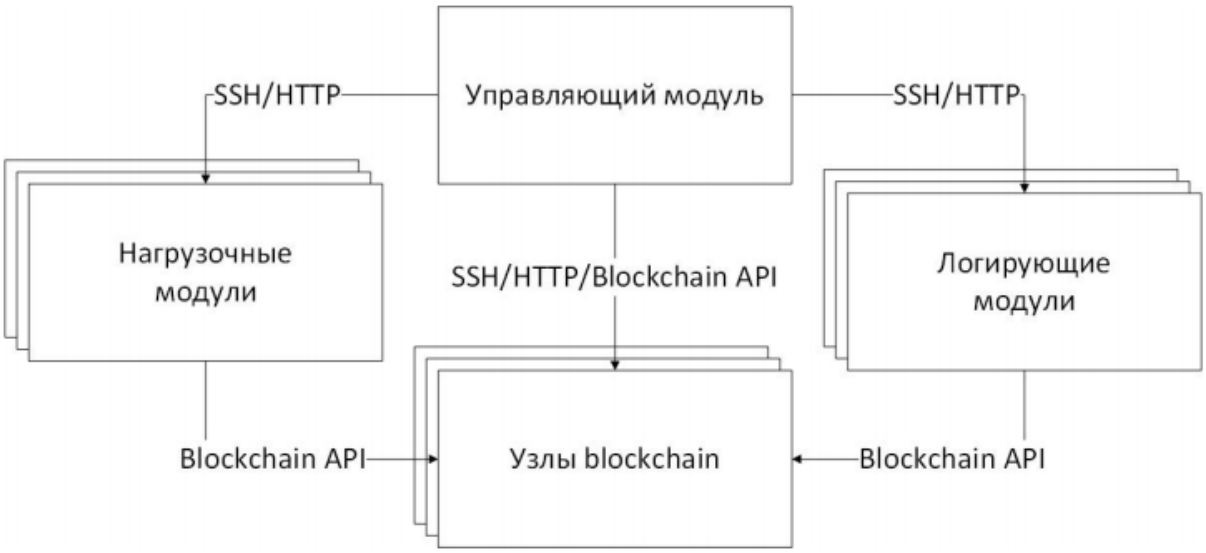

Рис. 2: Архитектура системы Blockchain benchmarking

Источник: Долголев Ф. П., Разработка системы измерения производительности реализаций blockchain, С. 15

Управляющий модуль инициализирует сеть Blockchain, запускает процесс тестирования и управляет им, собирает полученные данные. Для каждого узла Blockchain создается отдельный логирующий модуль, опрашивающий узел о появлении новых блоков с заданным интервалом времени. Нагрузочный модуль (один или несколько) генерирует новые транзакции заданного размера для узлов сети с задаваемой фиксированной периодичностью.

3.2. Поддержка задания сценариев нагрузки

На государственных выборах избиратели обычно голосуют в течение дня, однако в различное время дня активность голосующих из-

меняется [21]. Реализация модуля нагрузки поддерживала только симуляцию нагрузки с фиксированным временным интервалом между транзакциями, который задавался в конфигурации тестирования. Для поддержания гибкого задания сценариев было решено усовершенствовать модуль генерации нагрузки.

Blockchain представляет собой сеть узлов, которые могут многократно отправлять транзакции. Таким образом, можно поставить в соответствие узлу Blockchain избирательный участок на выборах, с помощью которого отдельные избиратели выражают свою волю. Самое явное альтернативное решение — ассоциировать каждого избирателя с отдельным узлом Blockchain, однако поддержание такого узла может быть достаточно дорогим и технически сложным для большинства избирателей, также в условиях такого соответствия невозможно было бы провести апробацию системы, поскольку потребовалось бы создание десятков и сотен тысяч узлов, что потребовало бы колоссальных финансовые расходы. Таким образом, в сети существует несколько узлов-избирательных участков, которые отправляют множество транзакций, а желающие проверить результаты могут создать свой узел и синхронизировать его с сетью для получения полных результатов голосования.

Так как активность голосующих на участках может различаться, необходимо было поддержать задание сценариев нагрузки для каждого отдельного узла сети. Для удобства задания сценариев для большого количества узлов был реализован сценарий по умолчанию, который выполняется, если сценарий для узла не задан.

Сценарий представляет функцию распределения голосующих во времени. Были реализованы основные математические функции, описывающие следующие явления:

- Неизменное количество голосующих. Данный сценарий уже был поддержан в системе, но в связи с изменением способа задания сценариев он был реализован заново. Пользователь задает число типа double, интенсивность — количество транзакций в секунду. Модуль генерации нагрузки рассчитывает фиксированную задержку между транзакциями, исходя из заданной интенсивности.

- Линейно возрастающее и убывающее количество голосующих. Пользователь задает начальную интенсивность генерации транзакций, изменение интенсивности генерации транзакций за секунду (положительное и отрицательное соответственно) и опциональную максимальную (или минимальную) интенсивность генерации транзакций. Модуль хранит время генерации первой транзакции, и при запросе рассчитывает текущую задержку, исходя из заданных параметров.

- Экспоненциально возрастающее количество голосующих. Пользователь задает основание степени показательной функции и опциональную максимальную интенсивность. Модуль хранит время генерации первой транзакции, и при запросе рассчитывает текущую задержку, исходя из заданных параметров.

Это очень простые функции, и для полноценной симуляции сложных процессов была добавлена возможность их комбинации. Пользователь задает для каждого конкретного узла сети упорядоченное множество периодов, включающих одну из вышеперечисленных математических функций распределения голосов и соответствующую ей длительность генерации голосов. Таким образом реализована возможность задания сложных сценариев, например, имитация нескольких периодов последовательного повышения и понижения интенсивности, что может быть полезно при проведении голосования в несколько раундов.

Для удобства хранения и управления конфигурациями было решено вынести описание конфигурации в отдельный файл. Для хранения был выбран формат JSON как достаточно простой формат, поддерживающий необходимые структуры данных (ассоциативные и упорядоченные массивы) и поддерживаемый большинством сред разработки.

3.3. Поддержка симуляции сетевых помех

Открытые популярные сети Blockchain содержат тысячи узлов, так, например, по состоянию на май 2018 года в сети Bitcoin более 10 тысяч

активных узлов [2], в сети Ethereum — более 15 тысяч [5]. Участники сети распределены по всему миру, поэтому при распространении информации в сети возможны задержки из-за удаленности географического расположения. Некоторые участники могут пользоваться нестабильным Интернет-соединением, что может создать дополнительные сложности в синхронизации состояния сети, поскольку для одобрения порядка добавления транзакций и блоков требуется распределенный консенсус, включающий соглашение от многих участников сети. Поэтому, как мы считаем, важно учитывать возможные сетевые проблемы при тестировании производительности реализаций Blockchain, на что, однако, не все исследователи обращают внимание. Например, в работе группы ученых из Национального центра компьютерных технологий Тайланда [13] узлы Blockchain запущены в облачном сервисе, в котором задержки между узлами очень низкие, что может привести к расхождению ожидаемой производительности с фактической при условии нестабильности соединения или высоких задержках.

Для симуляции сетевых проблем мы разработали соответствующий модуль. По аналогии с уже существующими модулями логирования потребления ресурсов и логирования появления блоков на узлах, данный модуль представляет собой Java-приложение, которое запускается параллельно с реализацией Blockchain.

Проанализировав возможные проблемы в сети, было решено добавить следующие:

- Симуляция потери соединения с узлом. Прерываются все сетевые соединения для заданного узла с другими узлами в сети Blockchain. Пользователь задает начало и конец блокировки относительно начала тестирования.

- Симуляция задержек. Пользователь задает фиксированную величину задержки, которая применяется ко всем соединениям данного узла с другими участниками Blockchain, времена начала и конца действия симуляции задержек относительно начала тестирования.

- Симуляция потери пакетов. Пользователь задает фиксированный процент от общего числа переданных пакетов для узла в сети Blockchain, которые будут отброшены во время тестирования, времена начала и конца действия симуляции потери пакетов относительно начала тестирования.

Для поддержания возможности последовательной симуляции нескольких проблем для каждого узла, было решено поддержать описание сетевых помех как множества из вышеперечисленных проблем. Также была поддержана возможность задания сценария по умолчанию для всей сети.

Поскольку скрипты для разворачивания тестовых Blockchain сетей в системе Blockchain benchmarking написаны для операционной системы Ubuntu, для реализации сетевых помех было решено использовать стандартные утилиты данной системы: IPTABLES [20] и NETEM [19].

Для удобства хранения и управления конфигурациями сетевых помех было также решено использовать JSON файлы.

4. Реализация инструмента для анализа процесса симуляции голосования

Система Blockchain benchmarking после запуска сценария имитации голосования предоставляет текстовую информацию, собранную в процессе тестирования. Для последующего анализа доступны следующие данные:

- Потребление ресурсов каждым узлом сети (Использование процессора, оперативной памяти, количество отправленной и полученной информации по сети).
- Время появления каждого блока на конкретном узле в сети.
- Время отправки узлом новой транзакции в сеть Blockchain.
- Принадлежность всех транзакций к соответствующим им блокам.

Исходя из этих данных, были выделены ключевые показатели, которые возможно представить в графическом виде.

Для отображения изменения нагрузки во времени необходимо отобразить количество отправленных транзакций за единицу времени для каждого узла и сети Blockchain в целом. Для построения таких графиков требуется только информация о времени отправки транзакций в сеть для каждого узла.

Узлы сети Blockchain объединяют транзакции в блоки и распространяют блоки по сети. Возможна ситуация, в которой 2 (или более) различных узла практически одновременно распространяют новый блок в сети, в таком случае участники запоминают все конкурирующие ветви и далее используют более длинную ветвь. Для корректной работы сети важным параметром является скорость распространения блоков, поскольку это позволяет уменьшить количество некорректной вычислительной работы, а именно случаев, когда участник сети пытается сформировать блок, после определенного блока, хотя этот блок уже не является последним в сети. Для отслеживания того, насколько быстро

распространяются транзакции и блоки в сети было предложено реализовать подсчет времени с момента создания транзакции до момента ее появления на определенном количестве узлов. Здесь могут быть полезными 2 разных представления: на одном показана зависимость задержки достижения фиксированного количества узлов от времени, этот график можно накладывать на график количества отправляемых транзакций, чтобы оценивать, как изменение нагрузки влияет на показатель задержки; с другой стороны важно рассмотреть то, как зависит время распространения от количества узлов в сети, на таком графике по оси абсцисс отложено количество узлов, на которые успешно распространился блок, по оси ординат — время распространения. Для вычисления данной информации необходимо использовать информацию о времени создания и отправки транзакций, времени появления блоков на узлах, принадлежности транзакций к блокам.

Информация о потреблении ресурсов для каждого узла сети позволяет оценить, насколько вычислительно сложными являются те или иные сценарии. Однако для удобства визуального восприятия было решено наложить графики количества отправляемых транзакций и потребляемых ресурсов.

Для реализации инструмента визуализации был использован язык программирования Python, как один из самых популярных языков для анализа данных. Для данного языка разработано большое количество библиотек, упрощающих подготовку данных и построение графиков. Для реализации инструмента были использованы библиотеки NumPy [12] и Matplotlib [10]. Исходный код был оформлен в среде Jupyter Notebook [9], позволяющей удобно совмещать исходный код программы и результат ее выполнения.

5. Апробация

Апробация системы проводилась на реализации Blockchain Ethereum с клиентом Geth v 1.8.9. Для запуска узлов Blockchain использовались виртуальные машины облачного решения Amazon AWS EC2 t2.medium (2 vCPUs, 2.3 GHz, Intel Broadwell E5-2686v4, 4 GiB оперативной памяти), для модуля генерации нагрузки с количеством узлов Blockchain не более 4 — t2.medium, в иных случаях — t2.xlarge (4 vCPUs, 16 GiB оперативной памяти).

Для тестирования гибких сценариев нагрузки мы запустили план симуляции со всеми четырьмя возможными типами генерации новых транзакций и построили график успешных транзакций во времени. Результат представлен на рисунке 3. Также на рисунке видно, что для каждого узла имеется возможность независимо задавать последовательность и периодичность сегментов генерации транзакций.

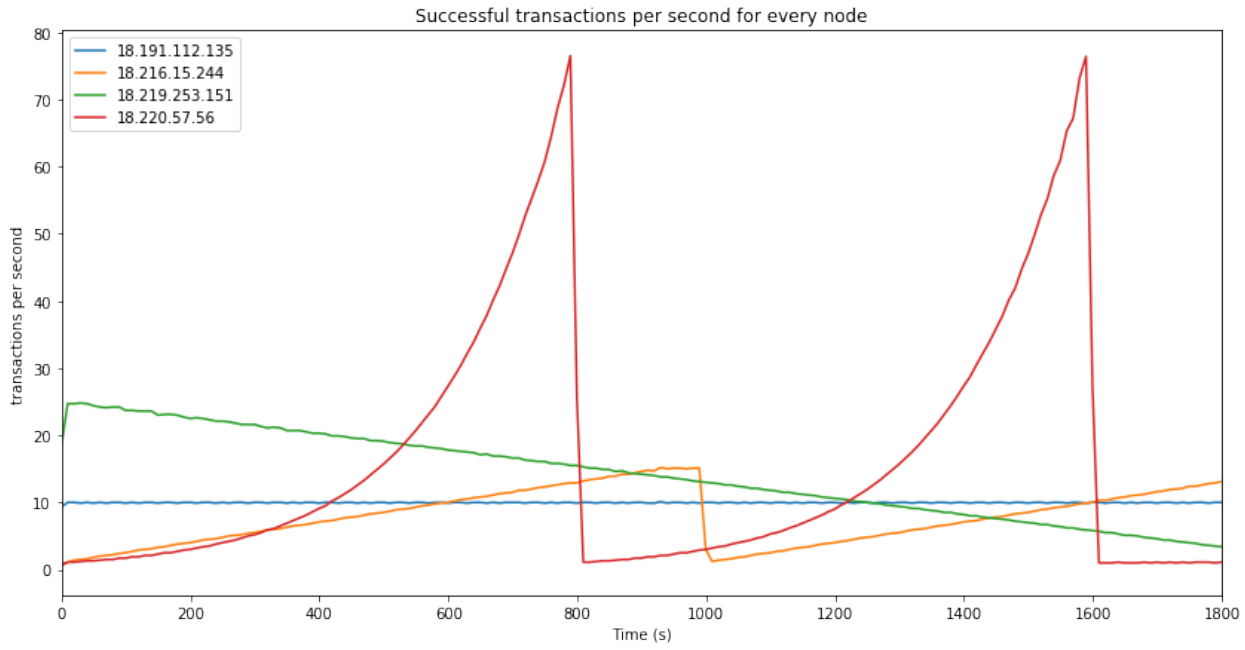

Рис. 3: Различные виды генерации транзакций

Для изучения влияния помех в сети мы провели тестирование с симуляцией сетевых задержек. В эксперименте сеть Blockchain состояла из 4 узлов, на каждый из которых подавалась фиксированная нагрузка с частотой в 15, 25, 35 и 45 транзакций в секунду соответственно

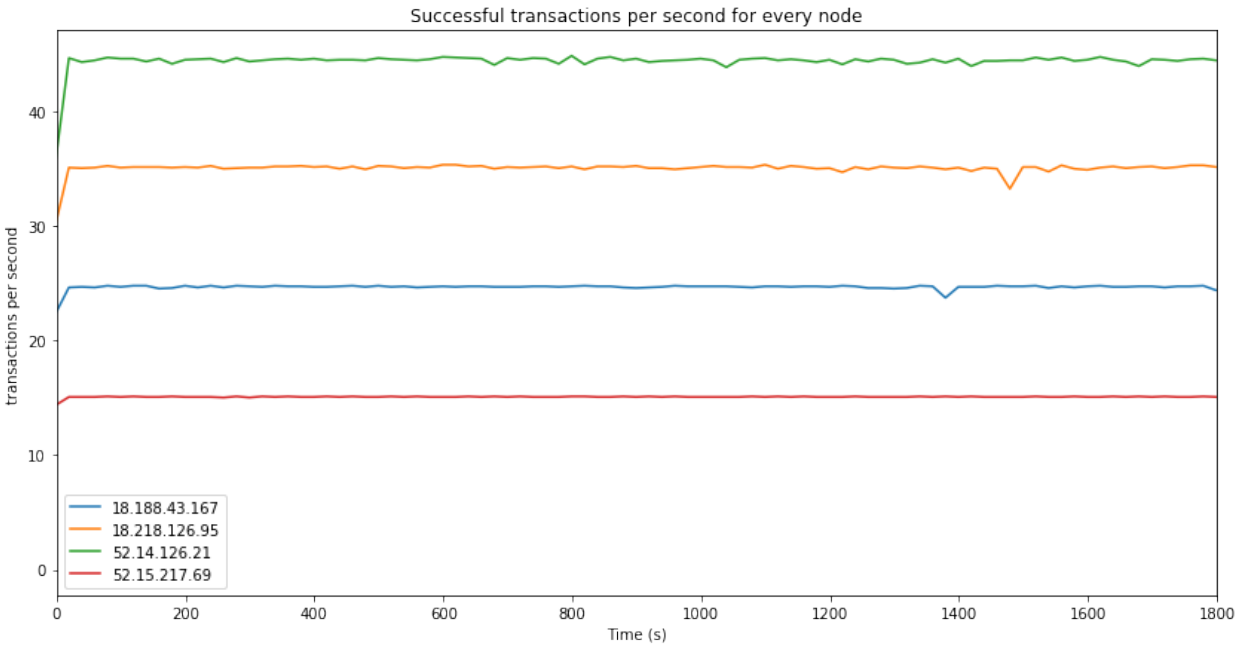

Рис. 4: Нагрузка для каждого узла в тестировании сетевых помех

(рисунок 4).

План симуляции сетевых помех был составлен таким образом, что первые 10 минут симуляции голосования в сети никаких помех не создавалось, задержка сообщений между всеми виртуальными машинами в одном датацентре не превышала 2 мс. Далее в течение 10 минут симулировалась задержка в 40 мс, далее в течение еще 10 минут задержка была повышена до 80 мс. Симуляция сетевых проблем не отразилась на количестве отправок успешных транзакций, однако на графике задержек распространения транзакций и блоков явно прослеживается рост задержки распространения с ростом задержек в сети. Так, во время первого периода средняя задержка распространения транзакции составила 2 секунды, во время второго — 3 секунды, во время третьего — почти 4 секунды с заметным ростом в самом начале третьего периода до 14 секунд (рисунки 5, 6).

По результатам анализа данной симуляции также стоит отметить рост потребления сетевых ресурсов (количества отправленной и полученной по сети информации) при неизменной нагрузке (рисунок 7), что говорит о том, что повышение задержек в сети ведет к увеличению потребления сетевых ресурсов, так как по результатам изучения симу-

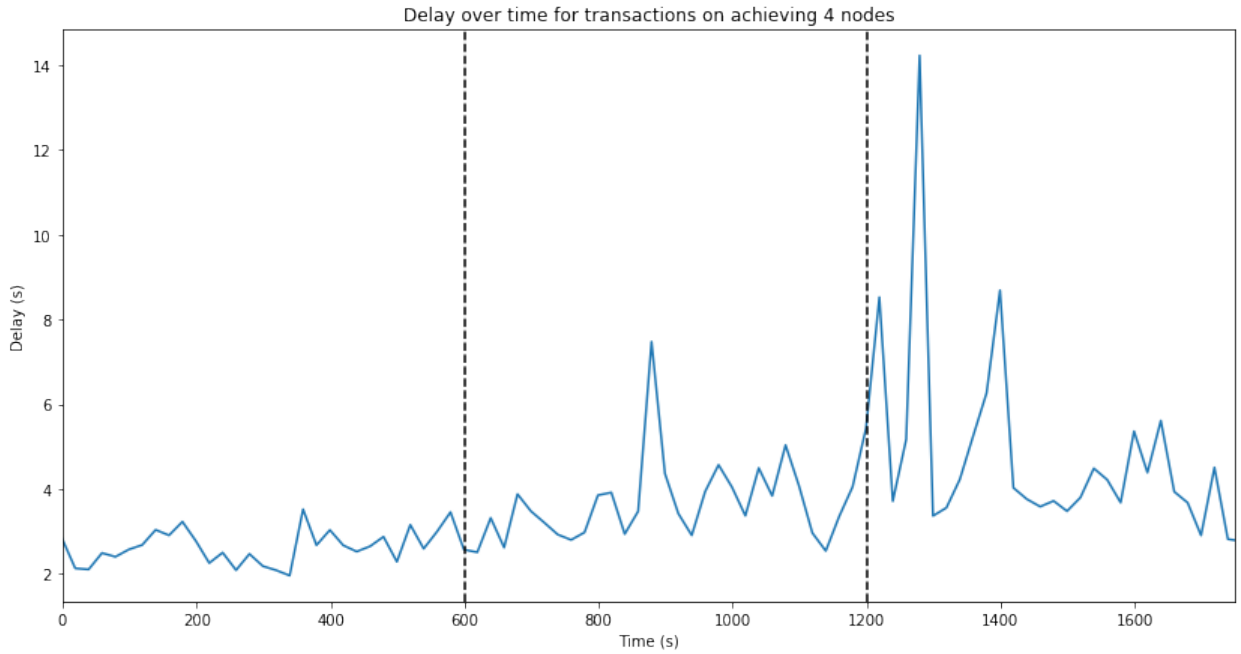

Рис. 5: Зависимость задержки распространения от задержек в сети. Пунктиром отделены 3 периода тестирования

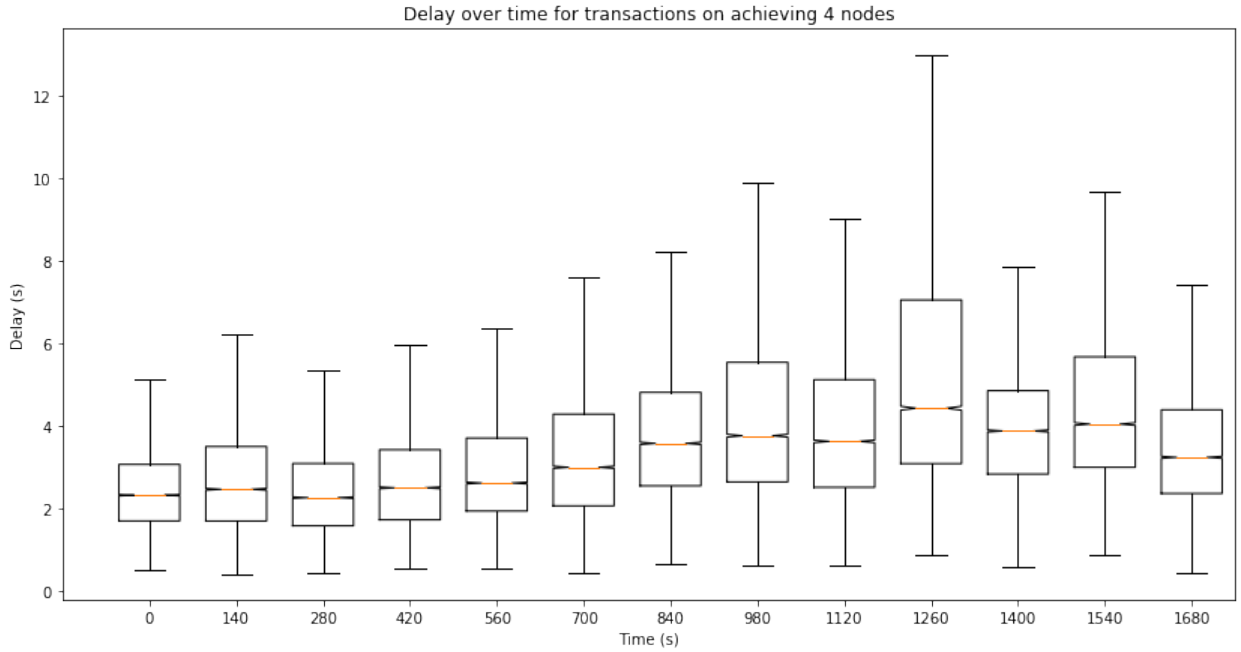

Рис. 6: Диаграмма размаха зависимости задержки распространения от задержек в сети

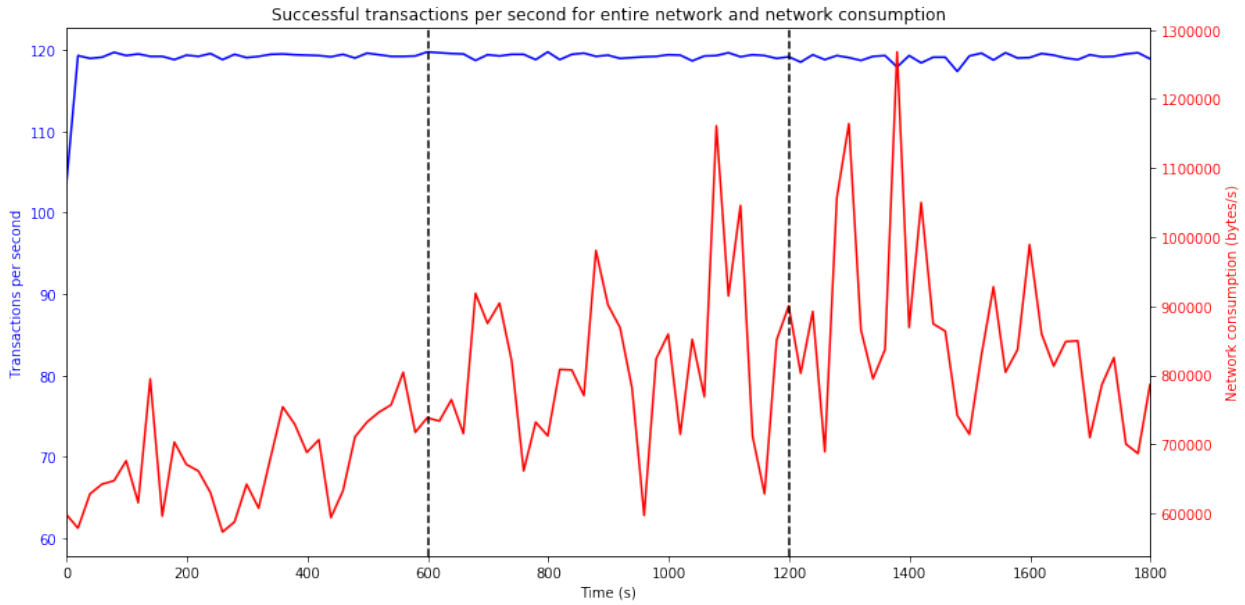

Рис. 7: Рост потребления сетевых ресурсов при неизменной нагрузке и увеличении задержек в сети

ляции других сценариев при неизменной нагрузке потребление сетевых ресурсов также оставалось неизменным.

Чтобы испытать сеть под высокой нагрузкой, мы провели тестирование с пиковой нагрузкой в 300 транзакций в секунду, что соответствует предельным значениям, обнаруженным в других работах [1] (рисунок 8).

По полученным данным можно сказать, что сеть под высокой нагрузкой может вести себя нестабильно, так как около второго пика нагрузки задержка распространения транзакций в сети резко возрастает, однако во время первого пика нагрузки такое поведение не обнаружено (рисунок 9).

Также полезным может являться наблюдение над использованием ресурсов процессора при таких нагрузках. Мы использовали двухъядерные виртуальные машины для запуска узлов Blockchain, при этом запуская майнинг на одном ядре. В процессе тестирования использование процессора не падало менее 50 процентов, так как одно ядро постоянно решало вычислительно сложные задачи по созданию новых блоков. Второе ядро было освобождено для решения текущих задач. В симуляции явно прослеживается зависимость использования процессо-

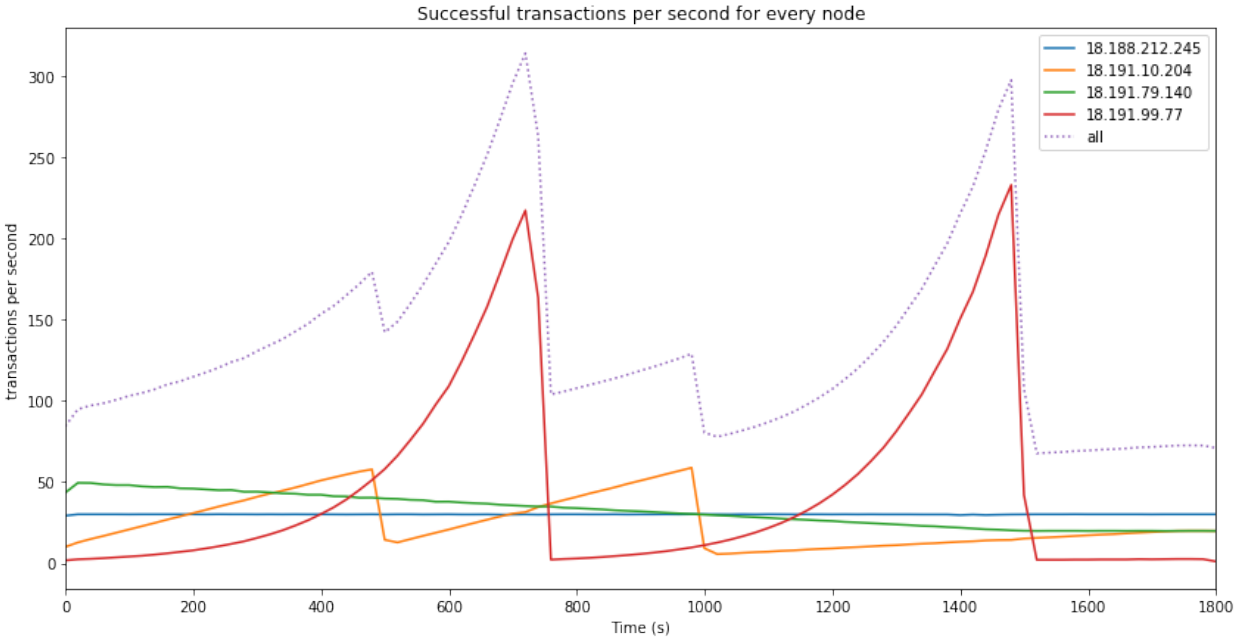

Рис. 8: Симуляция высоких нагрузок для сети Ethereum

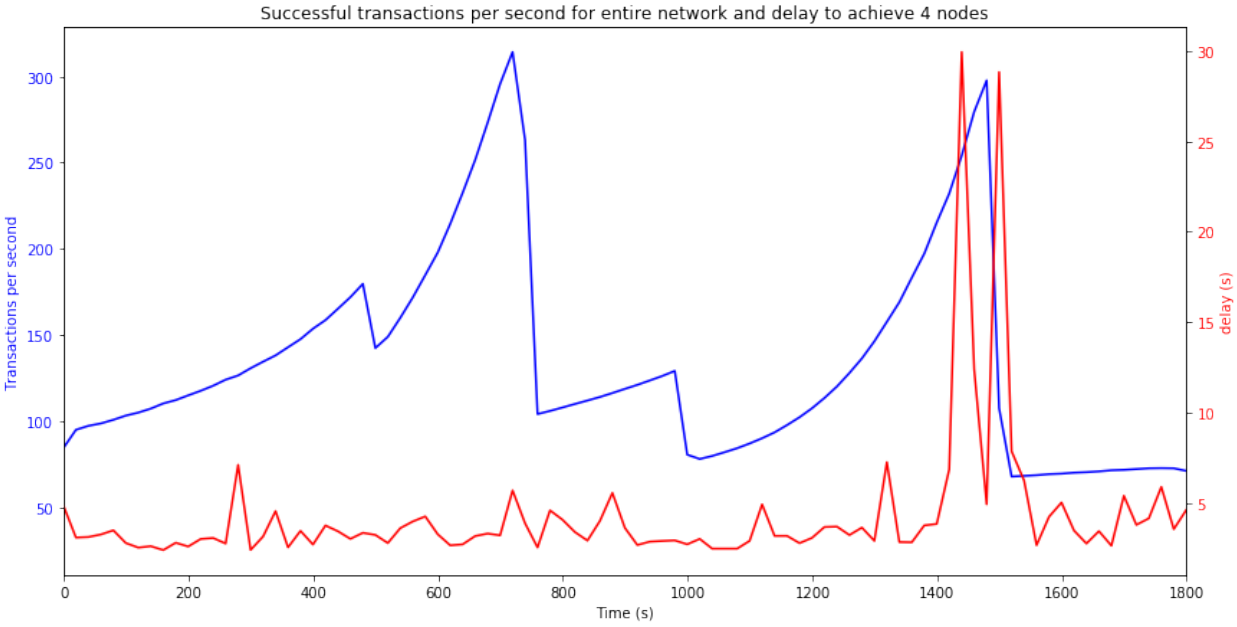

Рис. 9: Нестабильное поведение сети Ethereum под высокой нагрузкой

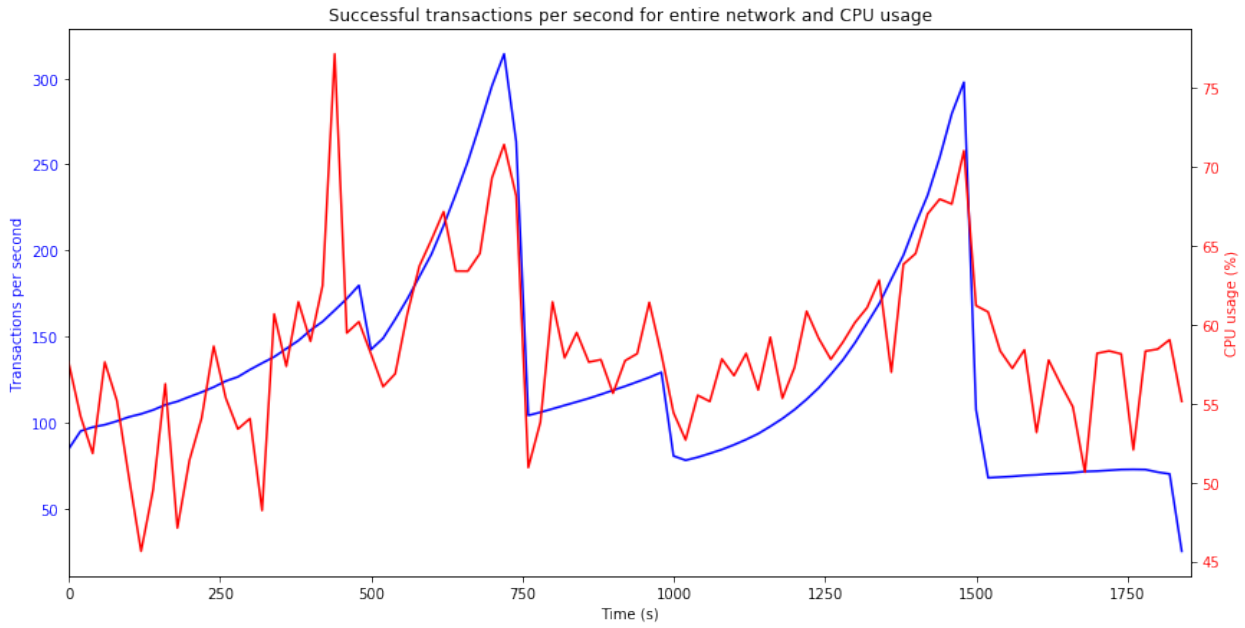

Рис. 10: Повышение использования процессора под нагрузкой

ра от нагрузки на узлы сети, загрузка процессора достигает 70 и более процентов при максимальных нагрузках и едва превышает 50 процентов при невысоких нагрузках (рисунок 10).

Для тестирования поведения сети при участии бóльшего количества участников мы провели тестирование на восьми виртуальных машинах, интенсивность имитации голосования на всех узлах была постоянной и составляла 20 транзакций в секунду. Интересным для анализа на такой конфигурации запуска является анализ того, как меняется время достижения определенного количества узлов по мере возрастания количества участников сети Blockchain. На рисунке 11 можно видеть, что задержка возрастает линейно с увеличением количества узлов в сети.

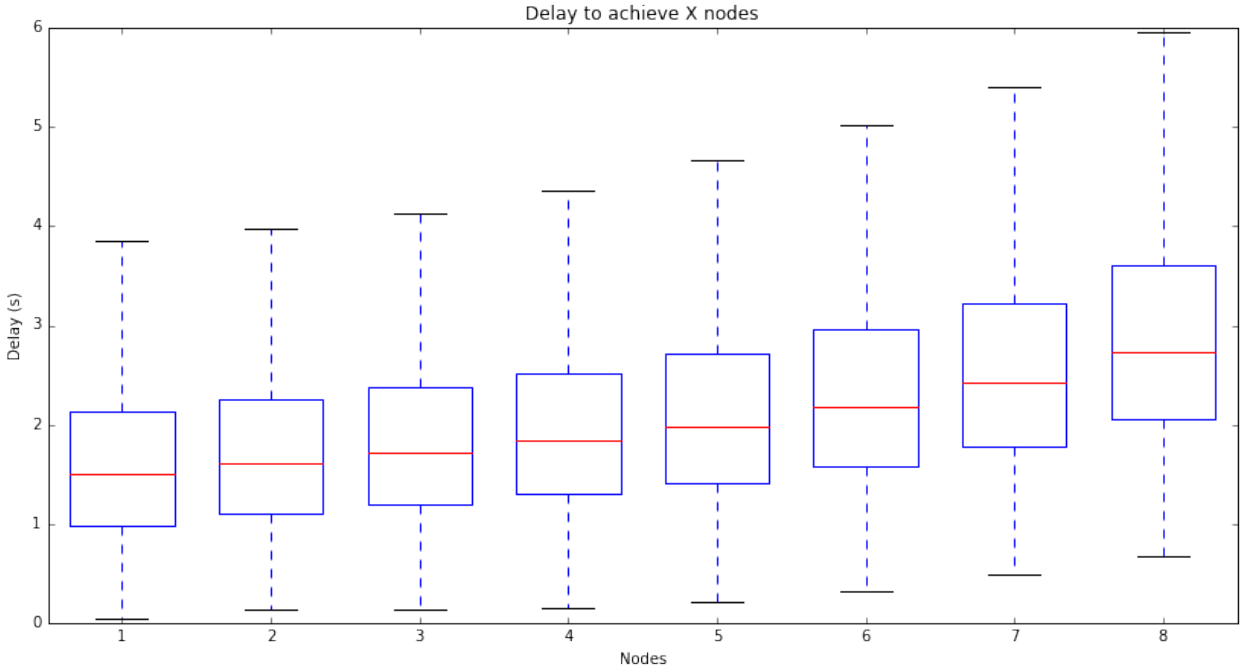

Рис. 11: Диаграмма размаха зависимости достижения определенного количества узлов от количества участников сети

6. Заключение

В рамках данной работы были получены следующие результаты.

- Проведен сравнительный анализ систем тестирования производительности реализаций Blockchain.
- В рамках системы Blockchain benchmarking реализована возможность гибко задавать сценарии генерации нагрузки и имитации сетевых помех для симуляции сценариев голосования.
- В рамках системы Blockchain benchmarking реализован инструмент для визуального анализа процесса имитации голосования.
- Произведена апробация системы Blockchain benchmarking на сценариях голосования с варьируемой нагрузкой и симуляцией сетевых помех.

Список литературы

[1] BLOCKBENCH: A Framework for Analyzing Private Blockchains / Tien Tuan Anh Dinh, Ji Wang, Gang Chen et al. // CoRR. — 2017. — Vol. abs/1703.04057. — 1703.04057.

[2] Bitnodes. Global Bitcoin nodes distribution. — URL: `https://bitnodes.earn.com/` (online; accessed: 2018-04-22).

[3] DSX Technologies. — URL: `https://www.dsxt.uk/` (online; accessed: 2018-04-22).

[4] Dolgolev Filipp. Blockchain Implementations Benchmarking Tool. — 2017.

[5] Ethernodes. The Ethereum nodes explorer. — URL: `https://www.ethernodes.org` (online; accessed: 2018-04-22).

[6] Hyperledger. Caliper. — URL: `https://github.com/hyperledger/caliper` (online; accessed: 2018-04-22).

[7] Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains / Elli Androulaki, Artem Barger, Vita Bortnikov et al. // CoRR. — 2018. — Vol. abs/1801.10228. — 1801.10228.

[8] IBM. Blockchain for supply chain. — URL: `https://www.ibm.com/blockchain/supply-chain/` (online; accessed: 2018-04-22).

[9] The Jupyter Notebook. — URL: `https://ipython.org/notebook.html` (online; accessed: 2018-04-22).

[10] Matplotlib. Python 2D plotting library. — URL: `https://matplotlib.org/` (online; accessed: 2018-04-22).

[11] Nakamoto Satoshi. Bitcoin: A peer-to-peer electronic cash system, http://bitcoin.org/bitcoin.pdf. — 2009.

[12] NumPy. Fundamental package for scientific computing with Python. — URL: `http://www.numpy.org/` (online; accessed: 2018-04-22).

[13] Pongnumkul Suporn, Siripanpornchana Chaiyaphum, Thajchayapong Suttipong. Performance Analysis of Private Blockchain Platforms in Varying Workloads // Computer Communication and Networks (ICCCN), 2017 26th International Conference on / IEEE. — 2017. — P. 1–6.

[14] Prisco Giulio. Russia's National Settlement Depository Successfully Tests Blockchain-Based E-Voting System. — 2016. — URL: https://bitcoinmagazine.com/articles/russia-s-national-settlement-depository-successfully-tests-block (online; accessed: 2018-04-22).

[15] Richard DeMarinis is Principal Software Engineer Enterprise Architecture Nas Hedi Uustalu is Head of Issuer Services Nasdaq Tallinn;, Fredrik Voss is Head of Blockchain Strategy Nasdaq. Is Blockchain the answer to E-Voting? NASDAQ belives so. — 2017. — URL: http://business.nasdaq.com/marketinsite/2017/Is-Blockchain-the-Answer-to-E-voting-Nasdaq-Believes-So.html (online; accessed: 2018-04-22).

[16] Spencer Douglas M, Markovits Zachary S. Long lines at polling stations? Observations from an election day field study // Election Law Journal. — 2010. — Vol. 9, no. 1. — P. 3–17.

[17] Stewart III Charles. Managing Polling Place Resources // Caltech/MIT Voting Technology Project Report. — 2015.

[18] Vitalik Buterin Gavin Wood Jeffrey Wilcke. A Next-Generation Smart Contract and Decentralized Application Platform. — URL: https://github.com/ethereum/wiki/wiki/White-Paper/ (online; accessed: 2018-04-22).

[19] Wiki Linux Foundation. netem. — URL: https://wiki.linuxfoundation.org/networking/netem (online; accessed: 2018-04-22).

[20] The netfilter.org project. — URL: https://www.netfilter.org/ (online; accessed: 2018-04-22).

[21] Габдульвалеев Азат. Динамика активности избирателей на выборах в г. Казани. — URL: https://trv-science.ru/2017/10/16/dinamika-aktivnosti-v-kazani/ (online; accessed: 2018-04-22).