Санкт-Петербургский государственный университет

ШАЛЕВА Анна Сергеевна

Выпускная квалификационная работа

Моделирование и оптимизация алгоритма распространения сообщений в одноранговой блокчейн-сети NEO

Уровень образования: магистратура
Направление 02.04.02 «Фундаментальная информатика и информационные технологии»
Основная образовательная программа ВМ.5827.2019 «Распределенные вычислительные технологии»

Научный руководитель:
доцент, кафедра компьютерного моделирования и многопроцессорных систем,
к.ф.-м.н., PhD
Корхов Владимир Владиславович

Рецензент:
технический директор, акционерное общество «Санкт-Петербургский Центр Компетенций НЕО»,
к.т.н.
Богатырев Анатолий Владимирович

Санкт-Петербург

2021

Оглавление

Введение

С возрастанием потребности в отказоустойчивости, масштабируемости и конфиденциальности компьютерных сетей высокую востребованность приобрела одноранговая (пиринговая, Peer-to-Peer) архитектура [1]. Одноранговые сети представляют собой альтернативу стандартной клиент-серверной архитектуре построения компьютерных сетей, а их основной идеей является объединение ресурсов множества равноправных компьютеров для формирования глобальной системы распределения контента.

На заре развития одноранговых сетей существовало несколько основных подходов к организации взаимодействия между участниками сети. Первый из них - многоадресная рассылка на сетевом уровне, также известная как IP мультикаст [1]. Данный подход позволяет осуществлять бессерверные коммуникации между участниками, однако его применение оказывается непрактичным в глобальном пространстве интернета [29]. Аналогичным подходом к конструированию одноранговых сетей стало создание полносвязной оверлейной сети на прикладном уровне [4], подразумевающее, что каждый участник сети имеет связь с любым другим. Подобные алгоритмы общения требуют высокой пропускной способности информационных каналов от каждого пира, возрастающей с увеличением общего количества участников сети, а также плохо масштабируются [1].

С развитием вычислительных мощностей, графических ускорителей и технологий параллельных вычислений масштаб и сложность структуры одноранговых сетей начали возрастать, что привело к необходимости оптимизации процесса сообщения между компонентами сети. С данной задачей успешно справились занимающие в настоящий момент лидирующие позиции структурно-ориентированные [2, 34] и эпидемические [12, 32] поколения алгоритмов, а также их гибриды. Структурно-ориентированные протоколы применяются в сетях, имеющих заранее определенную практически постоянную топологию (например, дерево), а их

преимуществом является эффективное использование сетевых ресурсов из-за отсутствия репликации сообщений. В противовес, с развитием сетей с динамической структурой и количеством участников широкое развитие получили эпидемические алгоритмы.

В протоколах многоадресной коммуникации, основанных на эпидемической модели распространения информации, итерация алгоритма обмена сообщениями состоит из нескольких фаз, в каждой из которых участники сети случайно и независимо друг от друга выбирают одного или нескольких соседей-партнеров для передачи порции информации - так называемого секрета [39]. При таком способе передачи информация распространяется по сети подобно сплетням или вирусу, из-за чего данное семейство алгоритмов получило название gossip. Gossip-алгоритмы принадлежат к вероятностному семейству алгоритмов и являются наиболее удачным решением для организации многоадресной коммуникации в динамических одноранговых сетях благодаря их стойкости к сбоям узлов и изменению структуры сети, высокой степени масштабируемости, наличию внутреннего уровня избыточности протокола, вероятностной устойчивости и надежным математическим оценкам эффективности [23]. Помимо широковещательной коммуникации данное семейство протоколов широко применяется при решении задач управления согласованностью в реплицированных базах данных [16], обнаружения сбоев и системного мониторинга [3, 5], организации распределенных брокеров, основанных на механизме подписок [28] и др.

Однако, данному классу алгоритмов свойственны недостатки, главным из которых является проблема “наводнения” сети [23], когда для достижения высокой степени надежности участники рассылают большое количество сообщений, превышая при этом пропускную способность каналов обмена информацией. Не менее важной задачей при проектировании gossip-протоколов является сокращение задержек при передаче сообщений [22].

За последнее десятилетие эпидемические алгоритмы нашли широкое применение в одноранговых сетях, основанных на технологии распределенных реестров, в том числе - технологии блокчейн [47]. Примером таких сетей служат наиболее известные блокчейн-системы Bitcoin [51], Hyperledger Fabric [27] и Neo [53]. В данном случае среда, в которой участники осуществляют обмен информацией, предполагается недоверенной - это означает, что сообщения, передаваемые участниками сети, могут содержать неточную или намеренно искаженную информацию. Таким образом, при организации общения в недоверенной среде возникают дополнительные затраты сетевых ресурсов при отправке, пересылке и принятии сообщений, связанные в первую очередь с их верификацией [47]. Учитывая данный факт, проблемы, общие для алгоритмов эпидемического семейства, становятся более актуальными в контексте блокчейн-сетей.

Несмотря на активные исследования и наличие большого количества зарекомендовавших себя подходов к оптимизации gossip-based протоколов, большинство из предлагаемых улучшений оказываются неприменимы в случае недоверенной среды (например, в случае блокчейн-систем), поскольку опираются на локальную информацию о сети, полученную участником от своих соседей [39]. Высокие требования к скорости распространения информации и утилизации сетевого трафика с одной стороны, и обеспечение надежности сообщения - с другой, поддерживают актуальной задачу разработки *бережливого* протокола общения в *недоверенной* среде, а дефицит научных трудов в данной области в совокупности с активным развитием блокчейн-технологий говорит о востребованности темы исследования.

В данной работе изложена классификация эпидемических алгоритмов и рассмотрены основные способы их оптимизации; изучен и приведен анализ протокола общения, использующегося в блокчейн-сети Neo, благодаря чему выделены ключевые особенности, характерные для gossip-based алгоритмов в недоверенной среде; исследована возможность применения

модифицированных эпидемических алгоритмов в недоверенной среде, в результате чего на основании наиболее удачных решений предложены способы оптимизации протокола общения Neo; проведены серия экспериментов по оценке эффективности предложенных способов оптимизации и сравнительная характеристика наиболее перспективных из них; исследована возможность применения предложенных способов оптимизации к другим gossip-based протоколам в недоверенной среде.

Постановка задачи

Целями данной работы являются:

1. Изучение ключевых особенностей эпидемических алгоритмов распространения сообщений в одноранговых сетях в недоверенной среде по сравнению со стандартными протоколами общения, применяемыми в одноранговых сетях с доверительной средой.
2. Разработка методов оптимизации алгоритма распространения сообщений в одноранговой блокчейн-сети.
3. Оценка эффективности разработанных методов путем моделирования блокчейн-сети с оригинальным и усовершенствованным протоколами на примере одноранговой блокчейн-сети Neo.

Разработанный метод оптимизации должен соответствовать требованиям, предъявляемым к алгоритмам общения в одноранговых сетях в недоверенной среде, основным из которых является минимизация опоры на информацию о характеристиках сети, получаемую участником сети от других участников сети.

Для оценки эффективности алгоритма распространения сообщений в одноранговой сети используются следующие ключевые метрики: вероятность нераспространения информационного фрейма и загруженность сети. Вероятность нераспространения информационного фрейма (*frame non-delivery probability*, p_{nd}) является вероятностной характеристикой события «неполучение порции информации хотя бы одним участником сети» за фиксированное количество итераций алгоритма распространения сообщений и определяется отношением количества экспериментов, в результате которых хотя бы один участник сети не получил распространяемую порцию информации, к общему количеству экспериментов:

$$p_{nd}(r) = 1 - \frac{\sum_{i=0}^{r} n_d(i)}{n_{peers}}$$

Где $n_d(i)$ - количество участников сети, получивших сообщение в i-ю итерацию алгоритма; n_{peers} - общее количество участников сети; r – количество итераций алгоритма. Практический метод определения p_{nd} представлен в части 3.1 настоящей работы.

Загруженность сети (*bandwidth utilization*, BU) показывает степень потребления сетевого трафика участниками сети и вычисляется как суммарный размер переданных сообщений за единицу времени:

$$BU(t) = \frac{\sum_i size(m_i)}{\Delta t}, \qquad i: t \le t_{send}(m_i) < t + \Delta t$$

Где m_i – i-е переданное сообщение; $size(m)$ – функция, вычисляющая размер сообщения m; Δt – временной интервал для сбора статистики; $t_{send}(m)$ – время отправки сообщения m.

Оптимизированный протокол распространения сообщений должен показывать не менее чем 40% уменьшение значения средней загруженности сети для участников сети и не менее чем 3% уменьшение вероятности нераспространения сообщения за 10 итераций алгоритма распространения сообщений.

Обзор литературы

Возвращаясь к недостаткам структурно-ориентированных алгоритмов, описанным во введении, стоит также отметить, что, несмотря на высокую степень надежности и эффективное использование сетевых ресурсов, данный класс алгоритмов плохо масштабируется и теряет свои полезные свойства в сетях с динамической структурой, поэтому далее будем рассматривать протоколы общения, основанные на эпидемической модели.

Впервые идея эпидемических алгоритмов была адаптирована к решению задачи поддержания актуальных состояний в реплицированных базах данных в [16]. Данная работа положила начало двум основным типам эпидемических алгоритмов: основанным на антиэнтропийной модели и на модели распространения слухов. В рамках терминологии теоремы CAP, данные алгоритмы обеспечивают окончательную согласованность благодаря свойствам доступности и устойчивости к разделению, жертвуя при этом согласованностью [48]. Базовая концепция данных алгоритмов предполагает наличие трех типов распространения сообщений [22]:

- *push*, когда каждый участник сети периодически отправляет случайно выбранному участнику имеющуюся у него информацию;
- *pull*, когда каждый участник сети периодически запрашивает обновления у других случайно выбранных участников сети;
- *push-pull*, являющаяся комбинацией первых двух моделей.

Наиболее ранние исследования в данной области были направлены на увеличение скорости доставки сообщений [10], уменьшение количества трафика в сети [28] и вероятностной надежности алгоритмов [36] за счет ограничения размера подгруппы, внутри которой участник сети может обмениваться сообщениями, введения параметра *fanout* - максимального количества получателей из подгруппы, а также за счет частичной

организации участников сети в иерархические структуры с использованием некоторых сетевых метрик. Некоторые важные проблемы, затронутые в [16], были решены только несколько лет спустя, например, сокращения трафика сети, отправляемого на дальние дистанции [44] с помощью введения и оптимизации метрики, ограничивающей длительность распространения сообщений, а также развитие механизмов контроля над сообщениями в сети [15].

Распространение широкомасштабных динамических одноранговых сетей [26], усложнение топологий оверлейных сетей [45] и возрастание актуальности задачи агрегации сетевой информации [24] послужили толчком для развития эпидемических алгоритмов. Широкую область применения нашли синхронные gossip-протоколы [41], в которых фазы обмена сообщениями запускаются одновременно для каждого участника сети и заканчивается строго до начала новой фазы. Такие протоколы подразумевают, что период каждой фазы больше времени задержки передачи между любыми двумя участниками сети, однако данное условие препятствовало использованию такие протоколов для коммуникации в режиме реального времени. Решением данной проблемы стала идея асинхронных gossip-протоколов, впервые представленная в [36]. В них участники не придерживаются рамок синхронных фаз обмена сообщениями; вместо этого общение осуществляется асинхронно при каждом полученном сообщении.

В синхронных и асинхронных gossip-протоколах количество фаз алгоритма и количество переданных за цикл сообщений должно быть минимизировано для достижения высокой скорости распространения информации. Данная проблема являлась особенно актуальна в сетях, работающих в режиме реального времени, из-за строгих требований к задержкам при передаче сообщений, что послужило началом более сильного разделения областей применения gossip-алгоритмов: push-алгоритмы стали

использоваться в системах, где важно минимальное время задержек и высока пропускная способность каналов передачи информации в сети [16]; pull-алгоритмы нашли свое применение в сетях, где необходима минимизация количества передаваемых сообщений.

В рамках оптимизации времени распространения сообщений в push-алгоритмах в [12] была впервые представлена адаптивная настройка параметра *fanout*, контролирующая количество фаз алгоритма. Также в [32] была изучена зависимость вероятности успешного распространения информации по сети от количества участников сети и количества циклов алгоритма распространения, что послужило началом серии теоретических оценок эффективности gossip-алгоритмов. В [6] была впервые представлена архитектура суперузла для запросов и маршрутизации в одноранговых сетях типа «один-к-одному», а авторы исследования [13] перенесли подход адаптивного параметра *fanout* на задачу об обнаружении новых участников в одноранговой сети.

Еще одним способом оптимизации времени распространения сообщений и сокращения объема трафика в сети стала гибридизация упомянутых выше подходов. В [25] в каждой фазе gossip-алгоритма используется объединенная операция push-pull для агрегации информации о сети. Авторы [38] и [41] разработали двухфазный алгоритм, где в первой фазе происходит обмен более легковесными заголовками информации с помощью push-операции, а во второй фазе используется pull-операция. Подобные вариации гибридных алгоритмов сочетают в себе способность к минимизации задержек push-операций с низкой избыточностью pull-операций.

Для организации общения в одноранговых стриминговых сетях был разработан трехфазный pull-алгоритм, называемый также «ленивый push» [17]. В таких сетях каждый цикл алгоритма состоит из трех: объявление об имеющейся информации, запрос отсутствующей информации и

непосредственно доставка информации. Трехфазные pull-алгоритмы при этом нацелены на распространение информации только одному слою участников сети и имеют более низкие требования к задержкам по сравнению с одноранговыми сетями, работающими в режиме реального времени.

Модификации асинхронных эпидемических протоколов используются и для других целей, например в [40] рассмотрено применение асинхронного gossip-протокола при решении проблемы усреднения в беспроводных сенсорных сетях, а [8] решает задачу суммирования компонентной функции в распределенной мультиагентной системе. В некоторых задачах [7, 14, 17, 21] применяются гибриды структурно-ориентированных и эпидемических протоколов, комбинирующие в себе бережливое потребление сетевого трафика структурных подходов со способностью справляться с динамическими изменениями состава и топологии сети gossip-протоколов. Однако данный подход требует знаний о топологии сети, поэтому спектр задач для его применения ограничен.

Наконец, за последние десятилетия широкое развитие получила еще одна область применения gossip-алгоритмов - сети, основанные на технологии распределенных реестров, в том числе - технологии блокчейн [47]. В контексте рассматриваемой задачи блокчейн можно отнести к неизменяемой глобальной реплицируемой базе данных. Наиболее популярные одноранговые блокчейн-сети Bitcoin [51], Hyperledger Fabric [27], Neo [53] используют асинхронные эпидемические протоколы обмена сообщениями для следующих целей: построение и содержание локального представления узла-участника о структуре сети, обнаружение новых участников сети, обмен метаданными состояний и, конечно, распространение блоков и транзакций по сети для поддержания актуального состояния локальной базы данных каждого участника. Данная сфера применения gossip-алгоритмов активно развивается, и наиболее значимой проблемой

является тот факт, что большинство алгоритмов, описанных выше, неприменимы к блокчейн-сетям, поскольку опираются на динамическую информацию о структуре и характеристиках сети, независимо агрегируемую каждым участником сети на основе полученных сообщений (например, количество участников сети, степень ее связности, загруженность частей сети и т.д.) [39]. При этом среда общения между участниками предполагается недоверенной - это означает, что сообщения, передаваемые участниками сети, могут содержать неточную или намеренно искаженную информацию, в том числе - информацию о структуре и характеристиках сети. Именно поэтому в большинстве используемых в блокчейн-сетях протоколах общения реализованы достаточно простые модели gossip-алгоритмов, такие как стандартные push или push-pull [20, 53]. Однако задача эффективного распространения сообщений в одноранговых блокчейн-сетях с минимальными задержками, эффективной утилизацией трафика и большой степенью надежности остается не менее актуальной, поскольку данные характеристики напрямую влияют на работоспособность и согласованность всей сети.

Поскольку в блокчейн-сетях ключевой метрикой эффективности протокола общения является размер задержек при передаче сообщений и меньшее внимание уделяется утилизации трафика в сети, большинство блокчейн-систем используют push или гибридную push-pull схемы алгоритмов gossip [47]. Принимая во внимание изложенные классификацию и историю развития эпидемических алгоритмов, в дальнейшей исследовательской работе в качестве базового алгоритма для оптимизации в недоверенной среде будет использован именно push-pull-gossip, вариант которого реализован в блокчейн-сети Neo. Однако, опираясь на опыт исследований, упомянутых в обзоре литературы, можно утверждать, что результаты данной работы могут быть обобщены на случай gossip-протоколов общения, использующихся в других сетях блокчейн.

Глава 1 Выбор инструментов для реализации задачи

Целью данной главы является сравнение и подбор оптимальных инструментов и средств для решения поставленных задач.

В данной главе приводятся обоснования выбора языка программирования для реализации оптимизированного протокола общения, выбора аппаратного обеспечения для проведения экспериментов по оценке эффективности сетевого протокола, а также выбора окружения для моделирования.

1.1 Выбор языка программирования

Поскольку целью настоящей исследовательской работы является оптимизация существующего протокола общения блокчейна Neo, для имплементации оптимизированного алгоритма целесообразно выбрать язык программирования, совпадающий с одной из реализаций данного протокола. Подобное решение позволит исключить фактор влияния программной имплементации протокола на результаты моделирования реальной сети и провести честный сравнительный анализ эффективности оригинального и улучшенного протоколов.

В настоящий момент существует три совместимых имплементации сетевого протокола Neo, написанные на языках программирования C#, Go и Python. Согласно предварительным исследованиям эффективности блокчейн-узла с каждой из имплементаций [54], лучшие результаты показывает реализация протокола на Go, следовательно, результаты сравнительного анализа эффективности оригинального и улучшенного протоколов будут иметь наибольшую научную ценность при выборе самой перспективной реализации протокола.

На момент написания данной работы версия 1.16 является самой поздней версией Go, постепенно заменяющей Go 1.15, содержит улучшения

и оптимизации, облегчающие имплементацию, поэтому будет использована для реализации оптимизированного протокола.

1.2 Выбор аппаратного обеспечения

Для получения наиболее достоверных результатов при оценке эффективности сетевого протокола необходимо моделирование полной блокчейн-сети, состоящей из нескольких десятков узлов, взаимодействующих между собой в течение длительного времени. Подобная система потребляет достаточно больше количество вычислительных ресурсов для обработки пересылаемых сообщений и дискового пространства для хранения репликации базы данных каждого узла сети. Таким образом, основными требованиями к аппаратному обеспечению для проведения экспериментов по моделированию за обозримое время являются наличие большого количества оперативной памяти и свободного дискового пространства.

Указанным требованиям удовлетворяет ЭВМ ресурсного центра “Вычислительный центр Санкт-Петербургского государственного университета” Wombat1. Данная ЭВМ имеет следующие характеристики:

- **ЦПУ:** 2 x Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60 ГГц
- **ОЗУ:** 256 ГБ
- **Графические процессоры:** 2 x Nvidia Tesla P100, по 16 ГБ видеопамяти

1.3 Выбор окружения для моделирования

При моделировании полной блокчейн-сети, состоящей из нескольких десятков узлов необходимо, чтобы каждый узел находился в изолированном окружении с возможностью регулирования потребления оперативной памяти и потребляемого дискового пространства. Кроме того, необходимо обеспечить одинаковые условия для запуска каждого узла на протяжении всего эксперимента и иметь возможность объединить узлы в подсеть. Также

немаловажным требованием к окружению для моделирования является способность автоматизации запуска большого количества однотипных контейнеров с узлами.

Данным требованиям полностью удовлетворяет программное обеспечение для развертывания и управления приложениями в средах с поддержкой контейнеризации Docker [52]. Данный инструмент предоставляет полностью контролируемую среду для запуска приложения, а также поддерживает эффективное управление серверными ресурсами. Будем использовать версию Docker Engine 18.09, установленную на указанной выше ЭВМ.

Кроме того, при проведении экспериментов по моделированию блокчейн-сети Neo необходим инструмент для отправки и регистрации транзакций в сети, а также для сбора некоторых сетевых метрик. В феврале 2020 года разработчики компании NSPCC представили инструмент с открытым исходным кодом NeoBench [49]. Данное программное обеспечение позволяет осуществлять бенчмарки сети-блокчейн Neo в различных режимах и предоставляет набор инструментов для анализа и визуализации результатов моделирования. NeoBench находится в стадии активной разработки, совместим с текущей версией блокчейна Neo, а также полностью удовлетворяет указанным требованиям, поэтому будет использован в качестве основы при построении окружения для моделирования.

Наконец, поскольку настоящая работа предполагает оптимизацию существующего протокола, для проведения экспериментов по моделированию блокчейн-сети необходимо выбрать реализацию узла блокчейн-сети. В работе используется наиболее перспективная реализация узла Neo, написанная на языке программирования Go [50] последней на момент написания настоящей работы версии NeoGo v0.94.0.

Глава 2 Исследование gossip-протоколов в недоверенной среде

Целью данной главы является изучение проблемы доверия в распределенных блокчейн-системах, ее влияния на используемые протоколы общения, а также систематизация знаний о протоколе, используемом в блокчейне Neo и выделение на его примере ключевых особенностей gossip-протоколов одноранговых сетей в недоверенной среде.

В данной главе рассматривается проблема византийских генералов и ее связь с функционированием одноранговых сетей в недоверенной среде. Проводится анализ одного из решений проблемы византийских генералов, примененный в блокчейн-сети Neo, и выделяется один из факторов, оказывающих влияние на эффективность работы сетевого протокола gossip в недоверенной среде. Проводится анализ реализации gossip-протокола в Neo, на основании которого выделяются ключевые особенности применения gossip-протоколов одноранговых сетей в недоверенной среде, оказывающие наибольшее влияние на эффективность сетевого протокола.

2.1 Проблема византийских генералов

Фундаментальным требованием, предъявляемым к распределенным вычислительным системам является сохранение работоспособности всей сети при выходе из строя одного или нескольких узлов. Часто для выполнения данного требования необходимо, чтобы процессы в системе согласовывали между собой некоторое значение, использующееся для дальнейших вычислений. Однако в недоверенной среде, когда процессы могут обмениваться недостоверной или искаженной информацией, согласование, или консенсус, представляют особую трудность. Данная задача носит название «проблема византийских генералов» и заключается в следующем: командующий генерал должен отдать приказ о нападении своим $n-1$ подчиненным генералам таким образом, чтобы:

1. Все генералы, не являющиеся предателями, подчинились одному и тому же приказу.
2. Если командующий генерал не является предателем, то все верноподданные генералы подчинились его приказу.

При этом подразумевается, что сообщение, переданное любым участником, может быть искажено, т.е. вместо приказа о нападении может быть передан приказ о ненападении и наоборот.

Впервые данная задача была решена для случая постоянного ограниченного количества генералов в [46] путем рекурсивного сведения случая m предателей из n генералов к случаю $m-1$ предателя. В [42] было доказано, что в распределенной системе с m неверно работающими процессорами - “генералами-предателями” можно достичь согласия только при наличии $2m+1$ верно работающих процессоров - “лояльных генералов” при условии возможности искажения сообщений при передаче. Алгоритм достижения согласия между процессами при такой формулировке задачи называется алгоритмом консенсуса, а византийская отказоустойчивость является характеристикой, определяющей систему, которая допускает класс отказов в формулировке задачи о византийских генералах.

Широкое развитие получили одноранговые сети, использующие различные вариации консенсусных алгоритмов для достижения согласия о состоянии данных в сети - блокчейн-сети. В сети блокчейн минимальной единицей информации являются транзакции, инициируемые пользователями. Транзакции, прошедшие верификацию и удовлетворяющие определенным критериям группируются в блоки, которые в свою очередь связываются в цепочку и сохраняются в локальной базе данных каждого узла-участника сети. Консенсусный алгоритм в данном случае выступает гарантом определенности состояния цепочки блоков для каждого участника сети. Одним из примеров сети блокчейн является блокчейн Neo, использующий

делегированный алгоритм консенсуса византийской отказоустойчивости (dBFT) [30]. В части 2.3 будут рассмотрены основные характеристики консенсусного алгоритма, применяемого в Neo, как факторы, оказывающие влияние на протокол общения в сети.

2.2 Делегированный алгоритм консенсуса византийской отказоустойчивости

Прежде чем перейти к изучению сетевого протокола блокчейн-сети Neo, необходимо рассмотреть алгоритм консенсуса как фактор, оказывающий значительное влияние на конструкцию сетевого протокола и его эффективность. Кроме того, в данной части будет введено несколько понятий, использующихся при описании сетевого протокола.

Алгоритм консенсуса, используемый для достижения согласия в блокчейн-сети Neo называется делегированным алгоритмом консенсуса византийской отказоустойчивости (delegated Byzantine Fault Tolerance, dBFT) [30], относится к подсемейству консенсусных алгоритмов, обеспечивающих устойчивость к атакам за счет “доказательства доли” - т.н. алгоритмы proof-of-stake, и сочетает себе масштабируемость алгоритма practical BFT со скоростью достижения решений делегированных алгоритмов доказательства доли. Данный консенсусный алгоритм предполагает наличие следующих ролей участников сети:

- *Консенсусный узел.* Участвует в консенсусной активности, т.е. собирает транзакции, полученные от регулярных узлов в блоки и выдвигает предложение о следующем блоке в сети.
- *Регулярный узел.* Распространяет информацию по сети, в том числе транзакции, блоки, консенсусные сообщения, но не участвует в консенсусной активности.

- *Узел-спикер.* Уникальный в каждой итерации консенсусного алгоритма член набора консенсусных узлов, ответственный за подтверждение и распространение предложенного блока по сети.
- *Узел-делегат.* Узел из числа консенсусных узлов, не являющийся спикером в данной итерации консенсусного алгоритма.
- *Узел-валидатор.* Узел, выбираемый из числа доверенного комитета сети, ответственный за верификацию провозглашенного узлом-спикером блока.

Алгоритм консенсуса dBFT является итеративным алгоритмом, каждая итерация которого состоит из четырех основных этапов [30]:

1. *Pre-prepare.* Из числа консенсусных узлов по определенному правилу выбирается узел-спикер данной итерации консенсусного алгоритма. Узел-спикер упаковывает транзакции из пула памяти в новый блок, а затем транслирует участникам сети сообщение PrepareRequest, содержащее провозглашаемый блок и метаинформацию.
2. *Prepare.* Узлы-делегаты из числа консенсусных узлов, не являющиеся спикерами на данной итерации алгоритма, получают сообщение PrepareRequest и верифицируют предложенный блок. В случае удовлетворения блока определенному набору правил, узлы-делегаты транслируют участникам сети сообщение PrepareResponce, содержащее верифицированный блок и метаинформацию.
3. *Persist.* Узлы-валидаторы, выбираемые из числа доверенного комитета сети, при получении $n - m$ сообщений PrepareResponce, транслируют участникам сети сообщение Commit.
4. *Выпуск блока.* После получения сообщений Commit блок считается принятым, а консенсусные узлы транслируют его участникам сети.

В данном алгоритме:

- n - общее количество консенсусных узлов в сети
- m - максимальное количество узлов-злоумышленников в сети
- h - текущая высота цепочки блоков
- v – текущий «вид» - значение для выбора спикера на данной итерации

На рисунке 1 представлено схематичное изображение одной итерации алгоритма консенсуса dBFT на примере четырех консенсусных узлов, один из которых является узлом-злоумышленником.

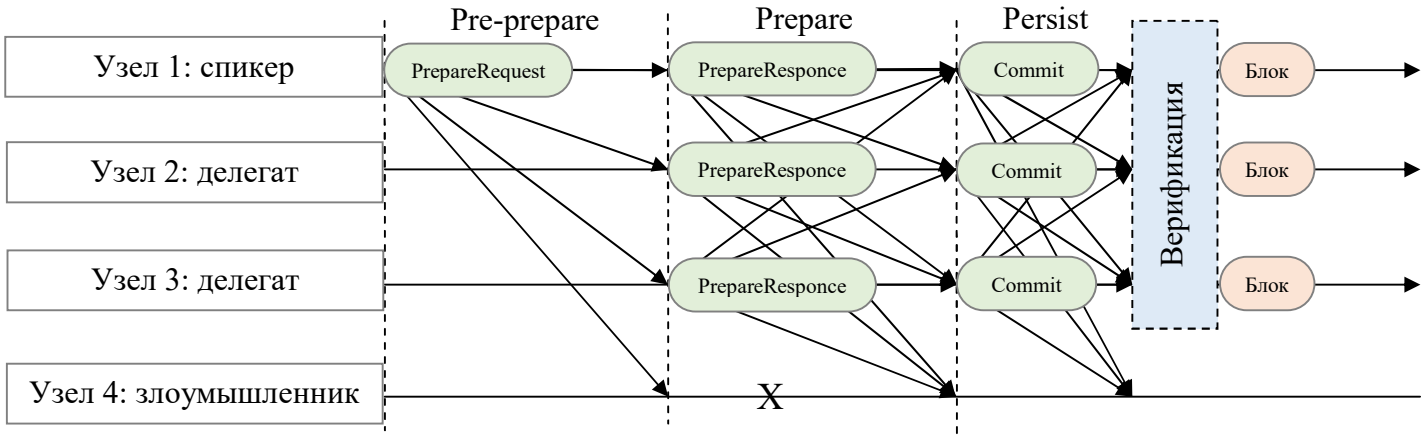

Рис. 1. Итерация алгоритма консенсуса dBFT с параметрами n=4, m=1

Таким образом, данный консенсусный алгоритм обеспечивает византийскую отказоустойчивость при наличии не более чем $m = \left\lfloor \frac{n-1}{3} \right\rfloor$ узлов-злоумышленников в блокчейн-сети. Общая схема *i*-й итерации данного алгоритма изображена на рисунке 2.

В заключение, после детального анализа механизма работы консенсусного протокола dBFT становится ясно, что основная нагрузка сетевого трафика, связанная с принятием и распространением блоков на начальной фазе работы сетевого протокола ложится на консенсусные узлы. Данное правило является общим для протоколов, основанных на “доказательстве доли”.

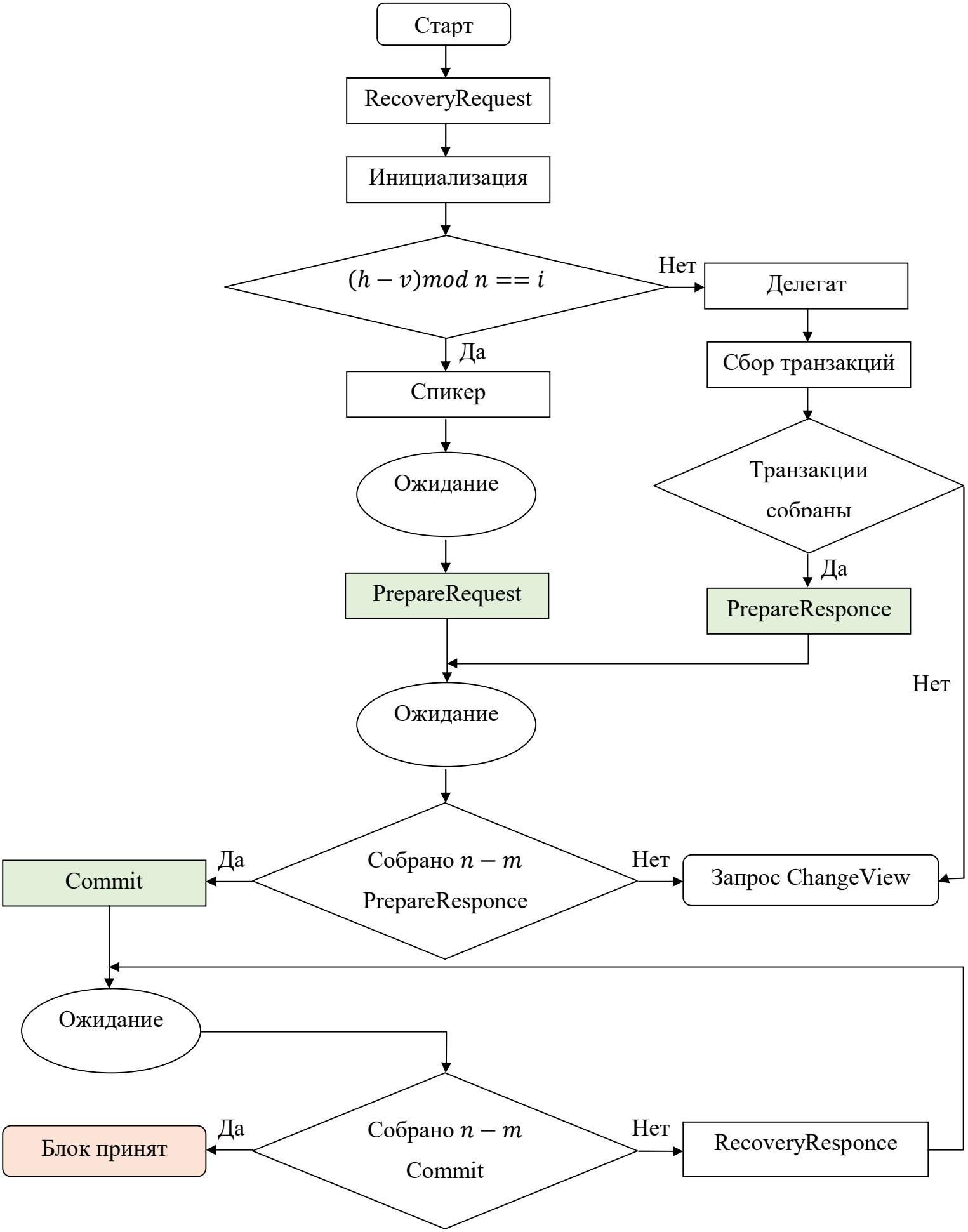

Рис. 2. *i*-я итерация алгоритма консенсуса dBFT

2.3 Анализ алгоритма распространения сообщений в Neo

Для дальнейшего анализа ключевых особенностей gossip-протоколов, применяемых в блокчейн-сетях, рассмотрим пример применения такого протокола в блокчейне Neo.

Neo, как и многие другие блокчейн-сети, использует эпидемический протокол для нескольких целей: построение и содержание локального представления узла о структуре сети, обнаружение новых узлов в сети, обмен метаданными состояний (например, текущей высотой цепочки блоков), распространение специальных сообщений для обеспечения работы консенсусного алгоритма и непосредственно распространение блоков и транзакций по сети. В данной работе основной акцент делается на последнем, причем поскольку алгоритмы распространения блоков и транзакций не имеют отличий, для определенности далее в исследовании будет рассмотрен процесс распространения блоков по сети начиная от момента их принятия консенсусным сервисом.

При дальнейшем рассмотрении предполагается, что распространение блоков происходит в среде с большим количеством участников, где прямая передача сообщения от консенсусного узла каждому участнику сети невозможна или является затратной операцией. Данное предположение необходимо для обоснования наличия в протоколе различных фаз и согласовано с действительным устройством функционирующей блокчейн-сети.

Опишем процесс распространения блока по сети. В момент принятия нового блока консенсусный сервис отправляет его копию консенсусным узлам, или узлам-лидерам, которые затем являются инициаторами процесса передачи информации - цикла gossip-алгоритма. Протокол Neo комбинирует в себе три базовых алгоритма распространения информации: *push*, *pull* и *recovery*, каждая из которых описана ниже.

2.3.1 Push-фаза

Neo использует так называемую «infect-and-die» модель push-операций [18]: когда участник сети получает блок первый раз (т.е. становится инфицированным), он пересылает данный блок всем n_{conn} участникам, с которыми он поддерживает соединение в настоящий момент. Однако, при повторном получении того же блока участник сети больше не распространяет его - т.е. он «пересылает» новый блок, а затем «умирает». Рассматривая данный процесс более подробно, новый блок сохраняется в буфер и распространяется только после момента добавления блока из буфера в локальную базу данных, а повторное добавление блока в буфер не допускается. При этом некоторые узлы сети получают данный блок несколько, в то время как другие могут не получить блок вовсе.

Пересылка блока проходит в два этапа: получив новый блок, на первом этапе участник сети делится с соединенными узлами уникальным идентификатором блока - его хешем. Те из узлов, которые еще не получили блок с указанным идентификатором, запрашивают полный блок у участника, инициировавшего передачу. На втором этапе происходит непосредственно пересылка блока узлам, отправившим команду об отсутствии блока.

2.3.2 Pull-фаза

Участники сети используют pull-фазу gossip-алгоритма для получения блоков, которые не удалось получить во время push-фазы. Циклически с интервалом времени t_{pull} при условии, что высота цепочки какого-либо из соседних узлов больше, чем локальная высота текущего участника сети, текущий участник сети инициирует pull-фазу, посылая определенное сообщение соседнему узлу. В ответ на данное сообщение соседний узел делится запрашиваемым блоком, после чего pull-фаза заканчивается.

2.3.3 Recovery-фаза

Recovery-фаза используется участниками сети, имеющими неактуальную локальную базу данных, например при вхождении нового узла в сеть либо после его долгой задержки. Во время данной фазы участник сети с устаревшим локальным состоянием цепочки блоков запрашивает метаданные о состоянии цепочки блоков у соединенных с ним узлов. Соседям с более актуальным состоянием цепочки блоков отсылается специальное сообщение, в ответ на которое соседние узлы делятся набором запрашиваемых блоков. На практике при нормальной работе сети данная компонента требуется только при присоединении нового узла к блокчейн-сети.

2.3.4 Влияние сетевого протокола на функционирование сети блокчейн

В то время как оптимизации масштабируемости и эффективности консенсусных протоколов уделяется достаточно большое внимание [33], сетевые протоколы и их влияние на состояние блокчейн-сети не подвергаются должным исследованиям. Однако, эффективность сетевого протокола оказывает прямое воздействие на частоту встречаемости конфликтов не только в Neo, но и в других блокчейн-сетях: медленное распространение транзакций по сети может вызывать временные коллизии при принятии новых блоков консенсусными участниками, поскольку для успешного завершения раунда протокола консенсуса необходимо, чтобы каждый из консенсусных узлов собрал одинаковый набор транзакций. Кроме того, большие задержки во время процесса распространения консенсусных команд по сети приводят к рассинхронизации и таймаутам во время работы консенсусного алгоритма, в результате чего увеличивается время принятия новых блоков, а, следовательно - увеличивается время выполнения транзакций.

Другим немаловажным фактором, попадающим под влияние сетевого протокола, является одна из ключевых характеристик сети - ее

загруженность. Эффективное использование сетевого трафика напрямую зависит от количества и интенсивности сообщений, пересылаемых участниками сети: большое количество синхронизационных сообщений в совокупности с низкой пропускной способностью каналов передачи информации в сети приводит к проблемам с распределением нагрузки, а также к дополнительным задержкам при принятии новых блоков и высокой стоимости работы всей сети. Грамотная оптимизация сетевого протокола позволяет минимизировать количество трафика в сети и решить указанную проблему.

2.4 Характерные особенности gossip-протокола в недоверенной среде

Исследование практического применения gossip-протокола в блокчейн-сети Neo, изложенное в части 2.3, а также изучение его влияния на состояние блокчейн-сети позволяет выделить характерные особенности gossip-протокола, применяемого в блокчейн-сетях с аналогичным консенсусным протоколом. В таблице 1 представлены выделенные особенности gossip-протокола в недоверенной среде, а также описано их влияние на функционирование сетевого протокола и сети блокчейн в целом.

Таблица 1. Характерные особенности gossip-протокола в недоверенной среде

Особенность	Причина	Влияние на функционирование сети
Высокая сетевая нагрузка на группу консенсусных узлов	Консенсусный протокол: источником распространения наиболее объемных сообщений в сети - блоков, является группа консенсусных узлов. Как правило, состав данной	Задержки при распространении информации в первой фазе gossip-алгоритма.

	группы редко меняется на протяжении всего времени функционирования сети, а ее размер не превышает десятка узлов.	
Минимальная агрегация информации о сети и ее использование	Недоверенная среда: возможность получения недостоверной информации от узлов-участников.	Превышение минимально необходимого для синхронизации количества сообщений. Следствием являются высокие аппаратные требования.
Неполная возможность выбора случайного узла в сети для обмена информацией	Недоверенная среда: высокая стоимость налаживания первого контакта между узлами за счет необходимости верификации.	Формирование связных подгрупп в сети с редким изменением членства.
Использование в push-фазе всех узлов, с которыми установлено соединение, в качестве адресатов	Недоверенная среда: отсутствие надежного критерия для выбора оптимального количества узлов-адресатов.	Обилие сообщений в сети и высокая нагрузка на каналы передачи информации.

Выделенные особенности будут приняты во внимание в части 3.3 при оптимизации сетевого протокола как факторы, оказывающие наибольшее влияние на эффективность сетевого протокола.

2.5 Выводы

1. Описана проблема доверия в распределенных блокчейн-системах и изучено ее влияние на проектирование сетевого протокола в недоверенной среде на примере делегированного алгоритма консенсуса византийской отказоустойчивости dBFT.
2. Исследованы области применения gossip-протоколов в блокчейн-сетях, а также детально рассмотрена реализация gossip-протокола в блокчейне Neo на примере распространения блоков по сети.
3. На основании результатов исследования выделены и обобщены особенности, характерные для gossip-протоколов в недоверенной среде, а именно - в сетях блокчейн, основанных на алгоритме “доказательства доли”. Охарактеризовано влияние, оказываемое выделенными факторами на характеристики, состояние и оперирование блокчейн-сети. Выделенные особенности являются теми пунктами, на которые стоит обратить наибольшее внимание при оптимизации сетевого протокола в Главе 3.

Глава 3 Разработка методов оптимизации алгоритма распространения сообщений в Neo

Целью данной главы является разработка методов оптимизации алгоритма распространения сообщений в блокчейн-сети Neo на основании исследованных в части “Обзор литературы” методов оптимизации gossip-алгоритмов, а также учитывая особенности gossip-протоколов в недоверенной среде, выделенные в Главе 2 настоящей работы.

В данной главе выделены и обоснованы ключевые метрики, использующиеся для определения эффективности работы сетевого протокола в сети блокчейн, проведена формализация условий задачи оптимизации протокола в недоверенной среде, разработаны и приведены способы оптимизации gossip-протокола в недоверенной среде на примере алгоритма обмена сообщениями в блокчейне Neo, а также описан выбор параметров модели оптимизированного алгоритма.

3.1 Целевые метрики

Задача настоящей работы предполагает наличие метрик для оценки эффективности протокола общения сети. Грамотное выделение таких метрик позволит провести честное сравнение между алгоритмами передачи сообщений и отсеять методы, показывающие недостаточно хорошие результаты оптимизации протокола.

Опираясь на анализ исследовательских работ, приведенный в части “Обзор литературы”, можно утверждать, что одной из наиболее важных общепринятых метрик для оценки эффективности gossip-протокола является вероятность нераспространения порции информации (вероятность нераспространения информационного фрейма, *frame non-delivery probability*) p_{nd}. Данная метрика представляет собой вероятностную характеристику события «неполучение порции информации хотя бы одним участником сети»

за фиксированное количество итераций алгоритма распространения сообщений и определяется отношением количества экспериментов, в результате которых хотя бы один участник сети не получил распространяемую порцию информации за фиксированное количество итераций алгоритма, к общему количеству экспериментов:

$$p_{nd}(r) = 1 - \frac{\sum_{i=0}^{r} n_d(i)}{n}$$

Где $n_d(i)$ - количество участников сети, получивших сообщение в i-ю итерацию алгоритма;

n - общее количество участников сети;

r – количество итераций алгоритма, $r \in \mathbb{Z}$.

Поскольку gossip-алгоритм, используемый в Neo, является асинхронным, т.е. не предполагает возможность четкого разграничения циклов распространения информации, для оценки вероятности нераспространения информации будет использован непрерывный аналог p_{nd} для асинхронных gossip-алгоритмов, вычисляемый за фиксированный временной интервал:

$$p_{nd}(t^*) = 1 - \frac{\int_0^{t^*} n_d(t)dt}{n_{peers}}, \qquad 0 < t^* \leq t_{end}, \qquad t^* \in \mathbb{R}$$

Где $n_d(t)$ - количество участников сети, получивших сообщение в момент времени t;

n_{peers} - общее количество участников сети;

t_{end} – время окончания распространения последнего блока.

В данном исследовании особое внимание будет обращено на $t^* = 10 * t_{block}$, где t_{block} - минимальный интервал между принятием двух последовательных блоков, определяемый конфигурацией блокчейн-сети.

Таким образом, данная метрика показывает, какова вероятность того, что блок не распространится на всех участников сети к моменту выпуска десятого следующего за ним блока, и характеризует скорость распространения блоков в сети. Целью оптимизации протокола является минимизация $p_{nd}(t^*)$.

Другими показательными метриками, тесно связанными с p_{nd} являются задержка на уровне блоков (*latency at the block level*, LBL) и задержка на уровне узлов (*latency at the peer level*, LPL). Задержка на уровне узлов LPL показывает, сколько времени требуется узлу сети для того чтобы получить блок от момента начала распространения блока консенсусными узлами:

$$LPL(i,t) = \frac{\sum_j 1}{n_{blocks}}, \qquad j: latency\left(peer_i, block_j\right) < t$$

Где i – индекс узла, $i \in \overline{0; n_{peers}}$;

j – индекс блока, $j \in \overline{0; n_{blocks}}$;

n_{peers} - общее количество узлов;

n_{blocks} - общее количество блоков;

$latency(peer_i, block_j)$ - функция, вычисляющая задержку получения j-го блока i-ым узлом от момента начала распространения блока.

Благодаря использованию данной метрики получится выделить группы узлов, получающие блоки с наименьшей задержкой, а также локализовать причину «отставания» для узлов, задержка при получении блоков которых велика. Целью оптимизации протокола является линеаризация LPL, т.е. уменьшение задержек при получении блоков для «отстающих» узлов сети.

Задержка на уровне блоков LBL показывает, насколько быстро блок достигает каждого узла сети с момента начала его распространения консенсусными узлами:

$$LBL(j,t) = \frac{\sum_i 1}{n_{peers}}, \qquad i: latency\left(peer_i, block_j\right) < t$$

Где i – индекс узла, $i \in \overline{0; n_{peers}}$;

j – индекс блока, $j \in \overline{0; n_{blocks}}$;

n_{peers} - общее количество узлов;

n_{blocks} - общее количество блоков;

$latency(peer_i, block_j)$ - функция, вычисляющая задержку получения j-го блока i-ым узлом от момента начала распространения блока.

Использование данной метрики позволит выявить блоки, которые распространяются с наибольшими задержками и локализовать причину их медленного распространения. Целью оптимизации протокола является минимизация LBL, т.е. сокращение задержек при распространении блоков на каждый узел сети.

Наконец, для оценки эффективности gossip-протокола будет использована метрика, называемая утилизация сетевого трафика (*bandwidth utilization*, BU). Данная метрика показывает степень потребления сетевого трафика участниками сети и вычисляется как суммарный размер переданных сообщений за единицу времени:

$$BU(t) = \frac{\sum_i size(m_i)}{\Delta t}, \qquad i: t \le t_{send}(m_i) < t + \Delta t$$

Где m_i – i-е переданное сообщение;

$size(m)$ – функция, вычисляющая размер сообщения m;

Δt – временной интервал для сбора статистики;

$t_{send}(m)$ – время отправки сообщения m.

Использование трафика в сети является не менее важной показателем, чем p_{nd}, LBL или LPL, поскольку пропускная способность сетевых каналов представляет собой наиболее ограниченный ресурс в одноранговых сетях [9]. Целью оптимизации протокола является минимизация потребляемого сетевого трафика, а также равномерное распределение потребляемых сетевых ресурсов между всеми участниками сети.

3.2 Формализация условий задачи

Исходя из условий поставленной задачи, исходными данными для практической оценки эффективности сетевого протокола являются два множества:

1. Матрица задержек $\boldsymbol{T} \in \mathbb{R}^{n_{peers} \times n_{blocks}}$, где t_{ij} – время получения блока *j* узлом сети *i*.
2. Матрица переданных сообщений $M \in \mathcal{M}^{n_{peers} \times n_{messages}}$ где m_{ij} - структура вида (*message*, *time*) – j-е сообщение, полученное узлом сети i за время $time \in \mathbb{R}$ от начала эксперимента.

Результирующими данными является функционал эффективности сетевого протокола, определяемый целевыми метриками:

$$\mathcal{F}\big(p_{nd}(t^*), LPL(i,t), LBL(j,t), BU(t)\big) = \mathcal{F}(t^*, t, i, j)$$

Определим алгоритм передачи сообщений в сети как отображение ω:

$$(\mathbb{R}^{n_{peers} \times n_{blocks}}, \mathcal{M}^{n_{peers} \times n_{messages}}) \xrightarrow{\omega} \mathcal{F}$$

Тогда задачей настоящего исследования является нахождение ω^* такого, что $\mathcal{F}^* > \mathcal{F}$.

3.3 Оптимизация протокола

Опираясь на способы оптимизации асинхронных gossip-протоколов, изложенные в части “Обзор литературы” настоящей работы, а также

принимая во внимание ключевые особенности gossip-алгоритмов, выделенные в Главе 2 и оказывающие значительное влияние на состояние и эффективность работы сети, были разработаны следующие способы оптимизации алгоритма общения, использующегося в блокчейне Neo.

3.3.1 Конфигурация числа fanout узлов

Согласно анализу gossip-протокола, реализованного в Neo и рассмотренного в части 2.3, при первом получении нового блока участник сети инициирует распространение блока всем узлам, с которыми поддерживается соединение в момент получения. Количество таких узлов (n_{conn}) регулируется параметром конфигурации сети на практике составляет от 10 до 50% от общего количества узлов сети. При размере сети в n_{peers} узлов при каждой итерации сетевого протокола участниками будет отправлено около $n_{peers} * \left(n_{conn} * n_{peers}\right) = n_{peers}^2 * n_{conn}$ сообщений, что несомненно превышает необходимый объем сообщений для обеспечения низкой p_{nd} при больших объемах сети. В то же время параметр n_{conn} не может быть уменьшен, поскольку напрямую влияет на связность всей сети и при малых значениях существенно замедляет скорость распространения сообщений по сети, а также увеличивает время принятия блоков.

Исходя из данных соображений, а также успешного применения данного подхода во многих эпидемических алгоритмах [21, 36] и наличия теоретических оценок для выбора оптимального параметра $fanout$ (f_{out}), целесообразно разделить максимальное количество участников сети, с которыми допускается единовременное поддержание открытого соединения, и количество узлов сети, с которыми происходит обмен новой информацией при ее получении. Другими словами, необходимо отделить параметр f_{out} от параметра n_{conn}.

На основании опыта статей [37, 39] ожидается, что выделение f_{out} в отдельный, не связанный с n_{conn} параметр позволит существенно сократить

объем сетевого трафика, не увеличивая при этом p_{nd} и средние задержки при распространении блоков по сети.

3.3.2 Рандомизация инициатора цикла gossip-алгоритма

Как было показано в части 2.3 настоящей работы, в консенсусных алгоритмах, основанных на «доказательстве доли», инициаторами распространения блоков на каждом цикле gossip-алгоритма выступает небольшая по численности группа узлов, называемых консенсусными. В протоколе Neo консенсусные узлы наравне с остальными узлами в сети при выпуске нового блока распространяют его хеш всем n_{conn} узлам. Однако консенсусные узлы являются первыми, кто узнает о новом блоке, поэтому, в отличие от обычных узлов, после фазы отсылки хеша консенсусные узлы всегда инициируют передачу самого блока. Данное поведение было также рассмотрено в части 2.3, где указано, что его результатом является высокая сетевая загруженность консенсусных узлов на протяжении всего времени работы блокчейн-сети по сравнению с регулярными участниками.

Для нормализации объема сетевого трафика в данном случае предлагается рандомизировать инициатора каждого цикла gossip-алгоритма путем выбора $f_{out\ leader} = 1$ и организации случайного выбора узла-получателя для консенсусных узлов. Другими словами, каждый консенсусный узел сети может делиться новыми блоками только с единственным случайно-выбранным участником сети из числа n_{conn} узлов, поддерживающих соединение с данным консенсусным узлом в текущий момент.

Ожидается, что подобное изменение протокола позволит существенно снизить сетевую нагрузку на консенсусные узлы, приблизив тем самым уровень утилизации трафика консенсусными узлами к уровню регулярных участников сети.

3.3.3 Использование техники infect-upon-contagion для push-фазы

Как было показано в части 2.3.1, в push-фазе gossip-алгоритма Neo используется стратегия «infect-and-die»: при получении нового блока участник сети пересылает данный блок каждому из n_{conn} узлов, с которыми поддерживается открытое соединение. При повторном получении блока участник не распространяет его, т.е. узел «пересылает» новый блок, а затем «умирает». Предлагается заменить данную стратегию push-фазы на стратегию «infect-upon-contagion» [43]. Ее отличие от описанной состоит в том, что при повторном получении блока узел не «умирает», а инициирует повторную передачу блока узлам-соседям. Согласно [18] данная стратегия обладает свойством балансировки сетевой нагрузки, поскольку она не возлагает чрезмерную ответственность на узел, который инициирует цикл gossip-алгоритма.

Данный подход требует условия остановки процесса распространения информации. Исходя из опыта [20], к каждому новому блоку будет прикреплен счетчик $counter$, начальное значение которого = 0. При достижении блоком очередного узла сети, данный счетчик будет инкрементирован, а блок будет передан f_{out} узлам-соседям по схеме 2.3.1. При достижении счетчиком определенного значения, называемого *time-to-live* (TTL), распространение данного блока останавливается. Следует также заметить, что параметры fout и TTL сильно связаны между собой, а выбор их значения обосновывается в части 3.4.

Исходя из данной характеристики, ожидается, что использование «infect-upon-contagion» стратегии в совокупности со способом оптимизации 2.3.1 уменьшит p_{nd} и сократит средние задержки при распространении блоков LPL и LBL за счет незначительного увеличения трафика в сети.

3.3.4 Итоговый набор улучшений протокола

Для более наглядного представления, разработанные способы оптимизации gossip-протокола Neo с пояснениями приведены в таблице 2.

Таблица 2. Список разработанных способов оптимизации gossip-протокола Neo

Способ оптимизации	Описание	Ожидаемый эффект
Конфигурация числа f_{out} узлов	Возможность отдельной настройки количества узлов-соседей для передачи нового блока.	Значительное сокращение объема сетевого трафика BU при неизменных p_{nd} и средних LPL и LBL.
Рандомизация инициатора цикла gossip-алгоритма	Ограничение $f_{out} = 1$ для консенсусных узлов и рандомизация узла-адресата.	Снижение сетевой нагрузки BU на консенсусные узлы.
Использование техники «infect-upon-contagion» для push-фазы	Возможность повторной передачи блока ограниченное TTL количество раз.	Уменьшение p_{nd} и сокращение LPL и LBL.

3.4 Выбор параметров

Для выбора оптимального соотношения f_{out} и TTL воспользуемся математическими оценками, изложенными в [39]. Для достижения желаемой вероятности нераспространения порции информации существует несколько оптимальных соотношений между параметрами f_{out} и TTL для достижения желаемой вероятности нераспространения порции информации. Положим желаемую $p_{nd} = 10^{-3}$, тогда исходя из соотношения [39] для дальнейших экспериментов будем использовать параметры $f_{out} = 4$ и $TTL = 9$.

3.5 Выводы

1. Формализованы условия задачи оптимизации gossip-протокола Neo и выделены наиболее значимые при оценке эффективности сетевого протокола метрики p_{nd}, LPL, LBL и BU.
2. Разработаны и описаны три способа оптимизации протокола. Приведен и обоснован ожидаемый эффект от применения данных способов оптимизации.
3. Приведено обоснование выбора параметров для разработанных способов улучшения.

Глава 4 Моделирование и оценка эффективности оптимизированного алгоритма

Целью данной главы является проведение экспериментов по моделированию блокчейн-сети Neo с использованием стандартного и усовершенствованного gossip-протоколов, а также оценка эффективности протоколов по результатам экспериментов.

В данной главе приведено подробное описание окружения, структуры и этапов экспериментов для неизмененного и усовершенствованного сетевых протоколов Neo; в виде сводных таблиц и графиков изложены результаты проведенных экспериментов, а также проведен детальный анализ оценки эффективности каждого предлагаемого улучшения. На основании анализа сформирован окончательный вариант улучшенного протокола, рекомендованный к использованию в блокчейне Neo.

4.1 Описание экспериментального окружения

4.1.1 Окружение

Для проведения каждого из экспериментов с помощью инструмента для моделирования и бенчмарков NeoBench была развернута блокчейн-сеть, состоящая из 101 узла:

- один узел осуществляет поддержку сервиса RPC и служит для принятия транзакций от клиента;
- четыре узла являются консенсусными и осуществляют деятельность по построению и принятию блоков;
- 96 узлов являются регулярными и занимаются хранением блоков и распространением сообщений по сети.

Каждый узел запущен в отдельном Docker-контейнере на кластере Wombat 1, характеристики которого описаны в части 1.2, и объединены в Docker-сеть. RPC клиент, параллельно отправляющий предварительно

сгенерированные транзакции в сеть и собирающий статистику эксперимента, также запущен в отдельном Docker-контейнере.

Исходя из соображений, изложенных в части 2.4, были выбраны следующие параметры блокчейн-сети, представленные в таблице 3.

Таблица 3. Параметры блокчейн-сети Neo для проведения эксперимента

Параметр	Описание	Значение
t_{block}	Время между принятием двух последовательных блоков (сек.).	2
$max_{tx\ per\ block}$	Максимальное количество транзакций в блоке.	50
n_{conn}	Количество участников сети, с которыми допускается единовременное поддержание открытого соединения для каждого узла.	10
max_{tx}	Общее количество транзакций для эксперимента.	50000

4.1.2 Ход эксперимента

Началом эксперимента считается принятие консенсусными узлами первого непустого блока. Окончанием эксперимента считается принятие консенсусными узлами последнего непустого блока. В ходе эксперимента выполняются следующие процессы:

1. RPC-клиент параллельно отсылает сгенерированные транзакции в сеть.
2. RPC-узел получает транзакции и распространяет их по сети.
3. Консенсусные узлы осуществляют деятельность по построению и выпуску новых блоков в сеть.
4. Регулярные узлы осуществляют деятельность по хранению блоков и распространению сообщений по сети.

В ходе эксперимента для каждого узла производится логирование времени принятия сообщений, относящихся к блокам (передаваемые хеши блоков и сами блоки), а также их размер и источник (консенсусный сервис или peer-to-peer сервис).

Таким образом, при заданных в таблице 3 параметрах, в ходе эксперимента ожидается, что примерно каждые 2-2,1 секунды в сети будет создаваться и распространяться новый блок, содержащий 50 транзакции, а в результате эксперимента будет принято 1000 блоков. Тогда длительность одного эксперимента будет составлять около 2000 секунд (33-35 минут). Выбранные количество блоков и время проведения эксперимента являются оптимальными, поскольку позволят собрать достаточно статистических данных для оценки эффективности сетевого протокола Neo.

4.2 Описание структуры экспериментов

Поскольку задача настоящей работы предполагает сравнение базового и улучшенного алгоритма, необходимо узнать исходное значение целевых метрик оригинального протокола. Те же эксперименты будут проведены для каждого из улучшений, предлагаемых в части 3.3, в отдельности, а также в комбинации. Кроме того, поскольку параметры f_{out} и TTL сильно взаимосвязаны, отдельно будет проведен эксперимент с комбинацией улучшений «конфигурация числа f_{out} узлов» и «использование техники infect-upon-contagion для push-фазы» для оценки их совместной эффективности.

Таким образом, в части 4.3 представлены эксперименты для следующих комбинаций улучшений:

1. Протокол без изменений.
2. Рандомизация инициатора цикла gossip-алгоритма.
3. Конфигурация числа f_{out} узлов.
4. Использование техники «infect-upon-contagion» для push-фазы.

5. Конфигурация числа f_{out} узлов и использование техники «infect-upon-contagion» для push-фазы.

Для каждой из перечисленных конфигураций в части 4.3 приводятся (рисунки 3-7):

а) график изменения значения вероятности нераспространения порции информации (блока) в зависимости от интервала времени, прошедшего с начала распространения порции информации (блока) $p_{nd}(t)$.

б) график задержек на уровне узлов в зависимости от времени, прошедшего с момента начала распространения блока консенсусными узлами $LPL(t)$.

в) график задержек на уровне блоков в зависимости от времени, прошедшего с момента начала распространения блока консенсусными узлами $LBL(t)$.

г) график средней утилизации сетевого трафика, относящегося к блокам, в зависимости от времени, прошедшего с момента начала эксперимента, для консенсусных узлов и для регулярных узлов $BU(t)$. На данном графике будет также отражено использование сетевого трафика за около 100 блоков до начала эксперимента и за несколько блоков после его окончания в сравнительных целях.

В целях улучшения наглядности результатов, а также более детального рассмотрения задержек для наиболее отстающих блоков и узлов, графики б) и в) имеют логарифмический масштаб по оси OY, основанный на логистическом распределении. Подобное представление позволяет сфокусироваться на “хвостах” изображенных кривых - первых 10% и последних 10% принятых блоков и изучаемых узлов. График а) также имеет логарифмический масштаб.

Для каждого из экспериментов приводится сводная таблица (таблицы 4-8), содержащая следующие результирующие показатели:

- Вероятность нераспространения порции информации $p_{nd}(t^*)$ за $t^* = 20$ секунд с момента начала распространения блока.
- Минимальные, медианные, средние и максимальные LPL и LBL.
- Средний уровень потребляемого сетевого трафика BU, относящегося к блокам, в состоянии “покоя” сети и во время эксперимента для консенсусных узлов и регулярных узлов.

4.3 Эксперименты

4.3.1 Моделирование сети с использованием исходного протокола

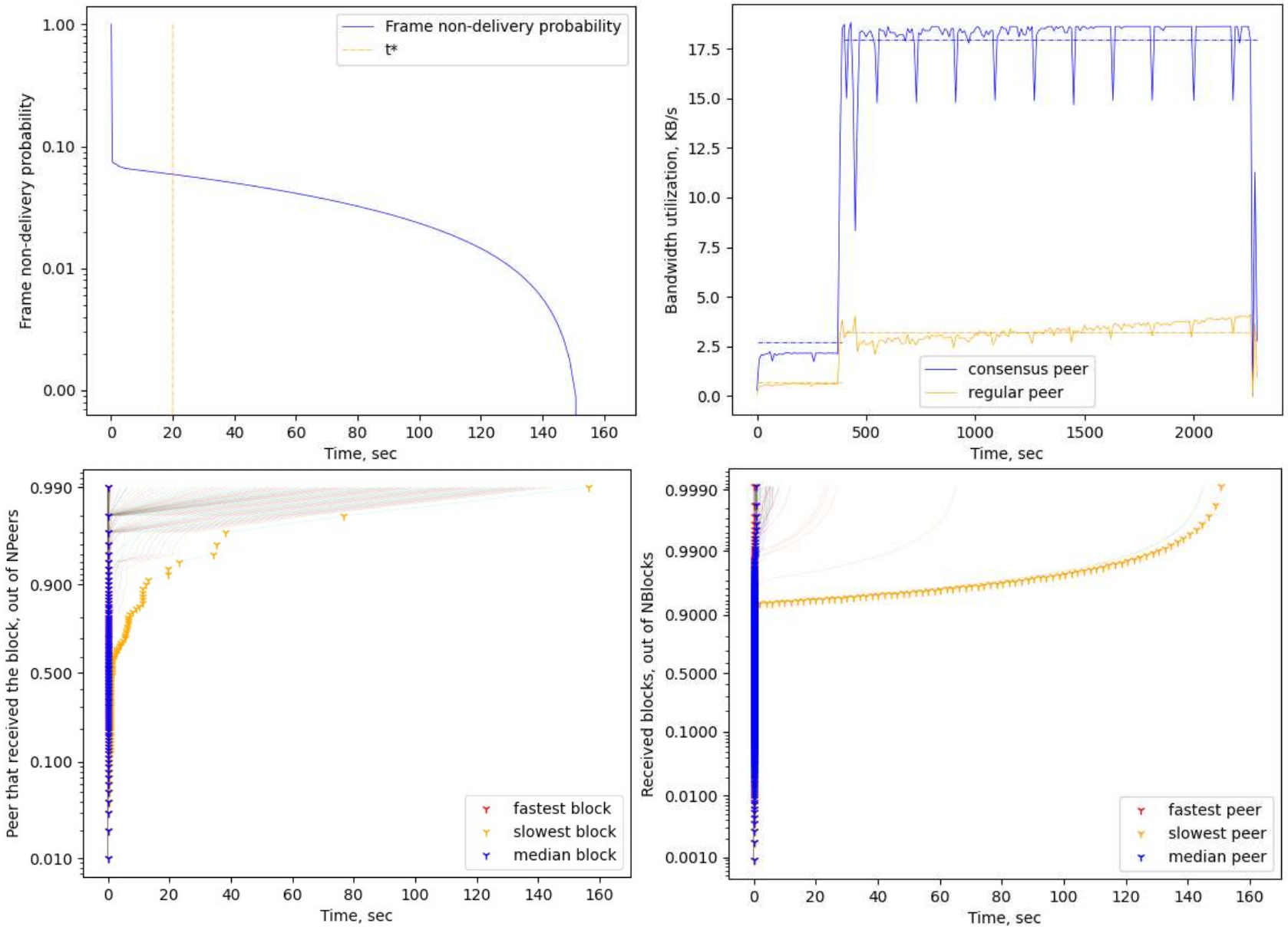

Рис. 3. $p_{nd}(t)$ - левый верхний угол, $BU(t)$ – правый верхний угол, $LBL(j,t)$ – левый нижний угол, $LPL(i,t)$ - правый нижний угол.

Таблица 4. Результаты моделирования сети с использованием исходного протокола

$p_{nd}(t^*)$	LPL, сек				LBL, сек			
	Мин.	Мед.	Сред.	Макс.	Мин.	Мед.	Сред.	Макс.
0.060	0.044	1.127	8.339	162.01	0.044	0.190	5.436	162.01
BU ср., KB/сек								
Консенсусный узел					Регулярный узел			
Неактивное сост.		Активное сост.			Неактивное сост.		Активное сост.	
2.733		17.949			0.718		3.244	

Исходный протокол блокчейн-сети Neo показывает довольно неплохие результаты при оценке эффективности: вероятность нераспространения блока за 20 секунд составляет всего 6%. Однако данный показатель достигается за счет высокой интенсивности потребления сетевого трафика, составляющей около 3 KB/сек. для регулярных узлов и 18 KB/сек. для консенсусного узла. Стоит отметить, что утилизация сетевого трафика консенсусными узлами может быть значительно сокращена, что является одной из целей настоящего исследования. Средняя и медианная задержки на уровне узлов и блоков невелики – составляют около 8 и 5 секунд соответственно. При этом около 5% всех блоков от 20 вплоть до 120 секунд, чтобы достичь наиболее удаленных узлов сети – задачей данного исследования является также разработка алгоритма передачи сообщений, обеспечивающего более равномерное распределение задержек при передаче.

4.3.2 Моделирование сети с рандомизацией инициатора цикла gossip-алгоритма

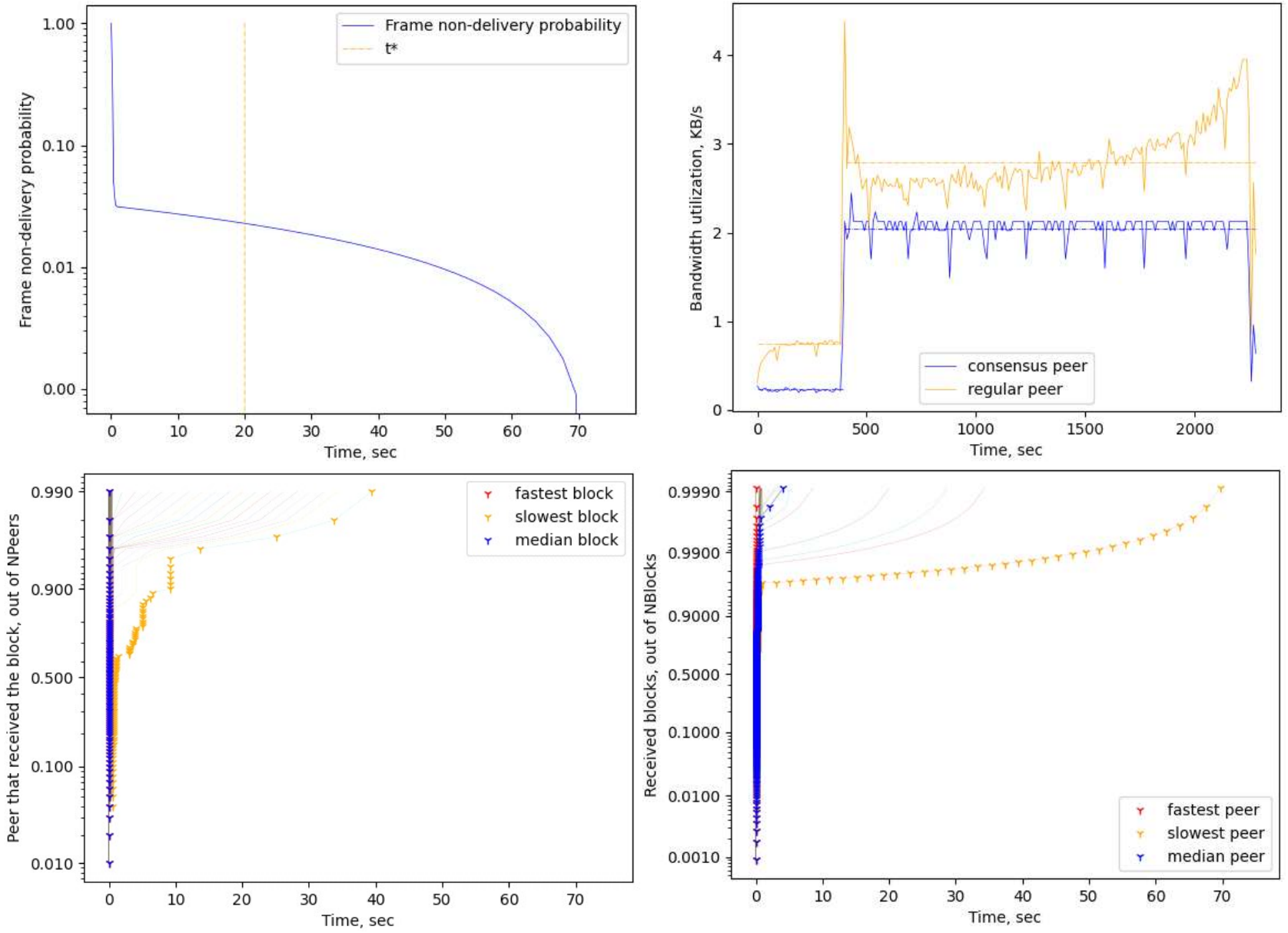

Рис. 4. $p_{nd}(t)$ - левый верхний угол, $BU(t)$ – правый верхний угол, $LBL(j,t)$ – левый нижний угол, $LPL(i,t)$ - правый нижний угол.

Таблица 5. Результаты моделирования сети с рандомизацией инициатора цикла gossip-алгоритма

$p_{nd}(t^*)$	LPL, сек				LBL, сек			
	Мин.	Мед.	Сред.	Макс.	Мин.	Мед.	Сред.	Макс.
0.023	0.028	9.195	4.022	74.760	0.008	0.148	1.315	74.760
BU ср., КВ/сек								
Консенсусный узел					Регулярный узел			
Неактивное сост.		Активное сост.			Неактивное сост.		Активное сост.	
0.238		2.048			0.752		2.7937	

Использование сетевого протокола с рандомизацией инициатора цикла gossip-алгоритма показывает впечатляющие результаты: вероятность нераспространения блока за 20 секунд составляет всего 2.3%, что на 3.7% меньше по сравнению с исходным алгоритмом. Однако наиболее значительный прирост эффективности наблюдается именно у показателя утилизации сетевого трафика консенсусными узлами, составляющий всего 2 КВ/сек в активном состоянии. Данное значение на 88% меньше исходного, при практически том же среднем уровне потребления трафика регулярными узлами. Кроме того, данное улучшение позволило значительно уменьшить задержки на уровне блоков и узлов для отстающих блоков и узлов соответственно, что позволяет достичь более равномерного распределения трафика на протяжении всего эксперимента.

4.3.3 Моделирование сети с конфигурацией числа f_{out} узлов

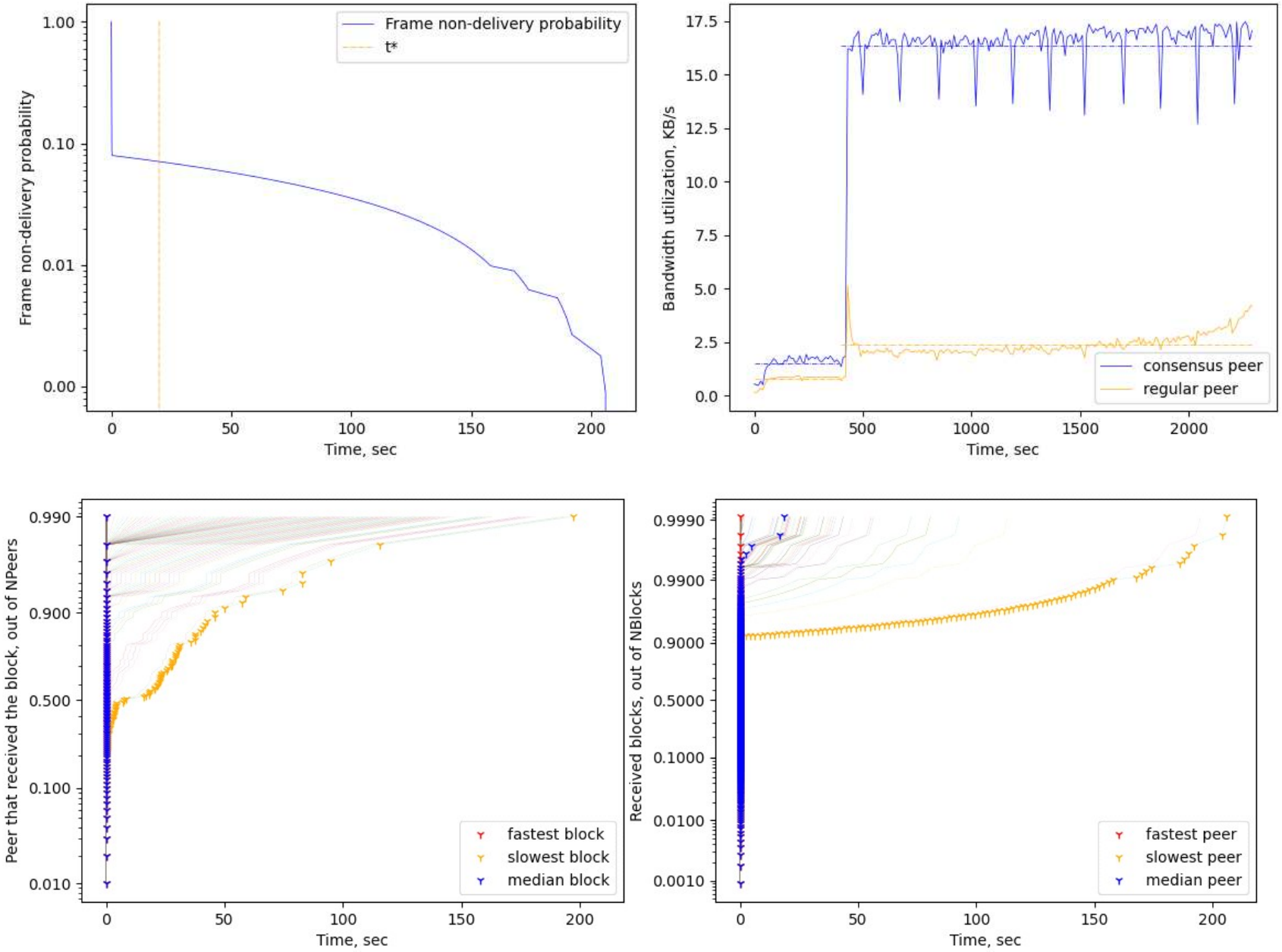

Рис. 5. $p_{nd}(t)$ - левый верхний угол, $BU(t)$ – правый верхний угол, $LBL(j,t)$ – левый нижний угол, $LPL(i,t)$ - правый нижний угол.

Таблица 6. Результаты моделирования сети с конфигурацией числа f_{out} узлов

$p_{nd}(t^*)$	LPL, сек				LBL, сек			
	Мин.	Мед.	Сред.	Макс.	Мин.	Мед.	Сред.	Макс.
0.071	0.044	20.725	22.182	208.07	0.006	0.124	7.533	208.07
BU ср., KB/сек								
Консенсусный узел					Регулярный узел			
Неактивное сост.		Активное сост.			Неактивное сост.		Активное сост.	
1.500		16.330			0.756		2.366	

Как и ожидалось, использование настраиваемого числа f_{out} без возможности конфигурации времени жизни блоков (TTL) не улучшило эффективность сетевого протокола. Вероятность нераспространения блока за 20 секунд увеличилась до 7.1%, а график $p_{nd}(t)$ стал более выпуклым и продолжительным: для того, чтобы все блоки гарантированно распространились между всеми узлами, понадобилось более 200 секунд, что составляет время принятия 100 блоков и не является допустимым значением. Кроме того, средний уровень утилизации сетевого трафика остался практически неизменным как для консенсусных, так и для регулярных узлов, а судя по графику LBL, более 55% всех узлов получают блоки с задержкой более 2 секунд.

Однако, данное поведение является предсказуемым, поскольку параметры f_{out} и TTL сильно взаимосвязаны и использование одного улучшения бед другого не принесет желаемых результатов.

4.3.4 Моделирование сети с использование техники «infect-upon-contagion» для push-фазы

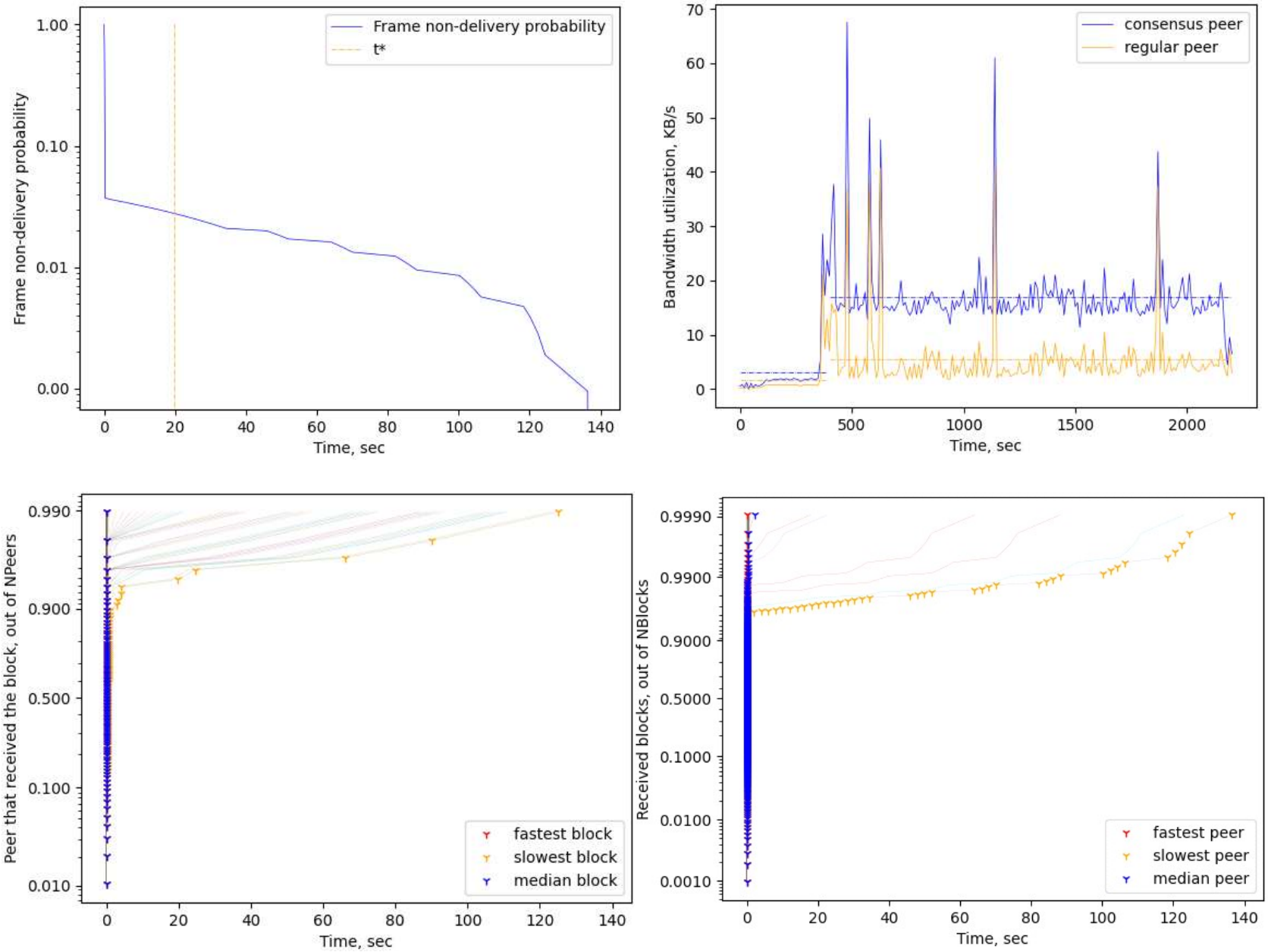

Рис. 6. $p_{nd}(t)$ - левый верхний угол, $BU(t)$ – правый верхний угол, $LBL(j,t)$ – левый нижний угол, $LPL(i,t)$ - правый нижний угол.

Таблица 7. Результаты моделирования сети с использованием техники «infect-upon-contagion» для push-фазы

$p_{nd}(t^*)$	LPL, сек				LBL, сек			
	Мин.	Мед.	Сред.	Макс.	Мин.	Мед.	Сред.	Макс.
0.028	0.029	4.170	5.395	138.40	0.006	0.150	2.309	138.40
BU ср., KB/сек								
Консенсусный узел					Регулярный узел			
Неактивное сост.		Активное сост.			Неактивное сост.		Активное сост.	
3.153		17.042			1.634		5.450	

Использование техники «infect-upon-contagion» для push-фазы gossip-алгоритма на 3 KB/сек увеличило средний уровень расходов сетевого трафика регулярными узлами, однако на графике BU видны ярко выраженные «пики» - моменты времени, когда сообщение передавалось по сети долгое время без выполнения критерия остановки передачи. Данное наблюдение говорит о необходимости использования дополнительного «ограничителя» для передачи сообщений, коим в следующем эксперименте будет выступать введенный параметр f_{out}.

Несмотря на неравномерность утилизации сетевого трафика, использование данной техники позволило уменьшить вероятность нераспространения блока за время t^* на 3.2%, а также, судя по поведению LBL, увеличить долю узлов, получающих блоки с задержкой менее 3 секунд до 90%.

4.3.5 Моделирование сети с конфигурацией числа f_{out} узлов и использованием техники «infect-upon-contagion»

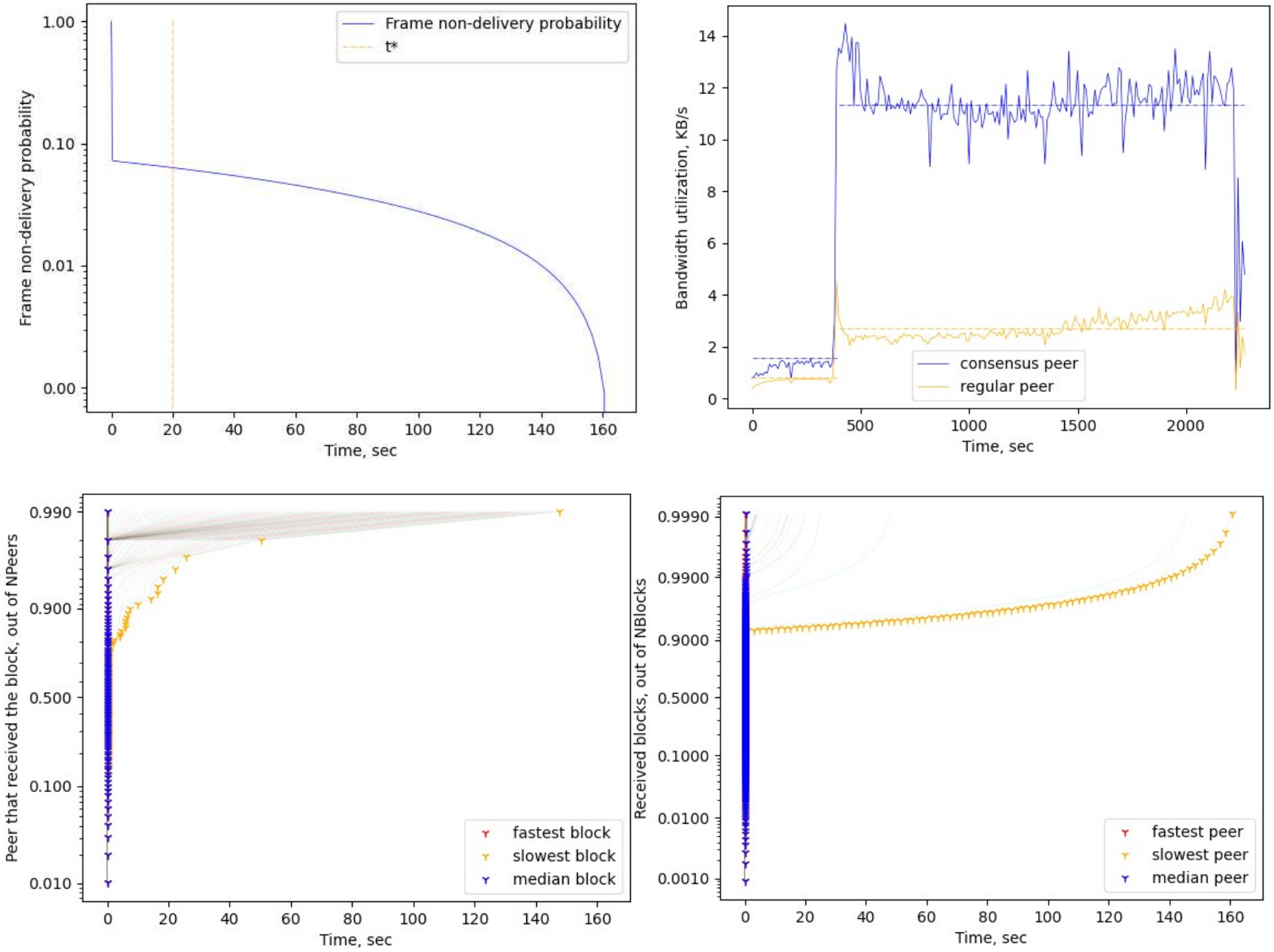

Рис. 7. $p_{nd}(t)$ - левый верхний угол, $BU(t)$ – правый верхний угол, $LBL(j,t)$ – левый нижний угол, $LPL(i,t)$ - правый нижний угол.

Таблица 8. Результаты моделирования сети с конфигурацией числа f_{out} узлов и использованием техники «infect-upon-contagion» для push-фазы

$p_{nd}(t^*)$	LPL, сек				LBL, сек			
	Мин.	Мед.	Сред.	Макс.	Мин.	Мед.	Сред.	Макс.
0.063	0.044	0.465	5.833	162.57	0.007	0.155	6.083	162.57
BU ср., KB/сек								
Консенсусный узел					Регулярный узел			
Неактивное сост.		Активное сост.			Неактивное сост.		Активное сост.	
1.578		11.325			0.813		2.726	

Наконец, комбинация техники «infect-upon-contagion» и введения числа f_{out} позволила достичь более равномерного распределения задержек при передаче блоков: медианные LPL и LBL составляют всего 0.465 и 0.155 секунды соответственно, а все узлы получают 92% блоков с задержкой менее 2 секунд.

При этом вероятность нераспространения блока за 20 секунд практически не изменилась, а уровень средний утилизации сетевого трафика консенсусными и регулярными узлами в активном состоянии уменьшился на 6 KB/сек и 1 KB/сек соответственно.

4.4 Оценка эффективности оптимизированного алгоритма

Таблица 9. Сводная таблица результатов моделирования

№	$p_{nd}(t^*)$	LPL, сек				LBL, сек				BU ср., КВ/сек			
										Консенсусный узел		Регулярный узел	
		Мин.	Мед.	Сред.	Макс.	Мин.	Мед.	Сред.	Макс.	Неактивное сост.	Активное сост.	Неактивное сост.	Активное сост.
1	0.060	0.044	1.127	8.339	162.01	0.044	0.190	5.436	162.01	2.733	17.949	0.718	3.244
2	0.023	0.028	9.195	4.022	74.760	0.008	0.148	1.315	74.760	0.238	2.048	0.752	2.7937
3	0.071	0.044	20.725	22.182	208.07	0.006	0.124	7.533	208.07	1.500	16.330	0.756	2.366
4	0.028	0.029	4.170	5.395	138.40	0.006	0.150	2.309	138.40	3.153	17.042	1.634	5.450
5	0.063	0.044	0.465	5.833	162.57	0.007	0.155	6.083	162.57	1.578	11.325	0.813	2.726

В таблице 9 представлена совокупность результирующих показателей для каждого из экспериментов. Для наглядности светло-зеленым цветом выделены значения показателей, лучшие по сравнению с показателями неизмененного протокола. Темно-зеленым цветом выделены лучшие показатели среди всей группы экспериментов.

Наиболее эффективным по итогам моделирования стал модифицированный протокол, основанный на рандомизации инициатора цикла gossip-алгоритма. Данное улучшение показало впечатляющие результаты по снижению уровня утилизации сетевого трафика: на 88% меньше для консенсусных узлов и на 27% меньше для регулярных узлов в активной фазе. При этом вероятность нераспространения блока за 20 секунд была снижена на 3.7%, а задержки на уровне блоков распространены более равномерно.

Добавление в протокол возможности конфигурации f_{out} узлов без комбинации с временем распространения блока показало ожидаемое увеличение вероятности нераспространения блока и задержек при передаче, поскольку данная схема распространения предполагает наличие в протоколе механизма отдельной репликации сообщений, которым является использование техники «infect-upon-contagion». Даная надстройка оказалась наименее эффективной при применении ее в одиночку.

Использование техники «infect-upon-contagion» для push-фазы gossip-алгоритма отразилось на равномерности распространения сетевого трафика – в отдельные моменты времени уровень его использования составлял до 65 KB/сек. консенсусными узлами и до 40 KB/сек. регулярными, что в разы превышает показатели немодифицированного протокола. Однако данная техника позволила уменьшить вероятность нераспространения блока на 3.2%.

Наконец, совмещение техники «infect-upon-contagion» с возможностью конфигурации f_{out} узлов, хотя и не обеспечило уменьшение вероятности

нераспространения блока, но способствовала более равномерному распределению трафика в сети. Таким образом, данная комбинация улучшений обладает высоким потенциалом к равномерному распределению трафика по сети без существенного увеличения вероятности нераспространения информационного фрейма, поэтому ее конфигурация будет изучена более подробно в дальнейшей исследовательской деятельности.

В заключение стоит отметить, что для достижения лучшей эффективности сетевого протокола Neo следует использовать метод рандомизации инициатора цикла gossip-алгоритма, как показывающий лучшие показатели вероятности нераспространения блока и утилизации сетевого трафика, а также обеспечивающий более равномерное распространение сетевого трафика. Дополнительные исследования необходимы для применения комбинации методов конфигурации числа f_{out} узлов и использования техники «infect-upon-contagion».

4.5 Выводы

1. Произведена настройка экспериментального окружения и выбор оптимальных параметров блокчейн-сети для проведения экспериментов по моделированию. Описан ход экспериментов по оценке эффективности сетевого протокола в блокчейн-сети на примере блокчейна Neo.
2. Проведена серия экспериментов по моделированию блокчейн-сети с оригинальным протоколам, а также с различными комбинациями предлагаемых улучшений протокола. Для каждого эксперимента приведены графики и сводные таблицы, отражающие поведение ключевых метрик на протяжении эксперимента.
3. Достигнуты лучшие показатели вероятности нераспространения блока за 20 секунд 2.3%, средней утилизации сетевого трафика 2.048 KB/сек и 2.793 KB/сек для консенсусного и регулярного узла соответственно, а

также максимальных задержек на уровне узлов и блоков 75 секунд при применении техники рандомизации инициатора цикла gossip-алгоритма.

4. По результатам экспериментов проведена сравнительная характеристика предлагаемых улучшений и выделены наиболее эффективные из них.
5. Реализован оптимизированный gossip-протокол для блокчейн-сети Neo, позволяющий достичь 88% уменьшения нагрузки на сетевой трафик для консенсусных узлов и 27% для регулярных при распространении блоков в сети, а также уменьшающий вероятность нераспространения блоков за 20 секунд на 3.7%.

Заключение

Результаты выполненной работы

6. Изучены важнейшие исследования и публикации в области применения эпидемических протоколов общения в одноранговых сетях, и, в частности, применение gossip-протоколов в блокчейн-сетях.
7. Исследована проблема доверия в распределенных блокчейн-системах и ее влияние на используемые протоколы общения. Рассмотрен и классифицирован сетевой протокол, применяемый в блокчейн-сети Neo, на основе чего выявлены ключевые особенности gossip-протоколов в недоверенной среде, влияющие на эффективность распространения информации в блокчейн-сетях.
8. Выявлена группа целевых метрик, определяющих эффективность работы сетевого протокола в сети блокчейн. Сформулирована задача оптимизации gossip-протокола блокчейна Neo.
9. На основе сравнительного анализа изученной литературы и выявленных особенностей gossip-протоколов в недоверенной среде сформулированы гипотезы о способах оптимизации сетевого протокола блокчейна Neo. Изложено математическое обоснование выбора параметров предложенных моделей.
10. Для каждого из выдвинутых способов реализован модифицированный сетевой протокол. Проведены эксперименты по оценке эффективности оригинального и модифицированного протоколов.
11. Достигнуты 88% и 27% уменьшение нагрузки на сетевой трафик для консенсусных и регулярных узлов сети соответственно; уменьшение вероятности нераспространения порции информации за 10 итераций алгоритма распространения сообщений на 3.7%; уменьшение задержки при передаче сообщений на уровне узлов и блоков на 54%.

12. Проведена сравнительная характеристика результатов экспериментов и выделение наиболее эффективных предложений по улучшению. Реализован оптимизированный gossip-протокол блокчейн-сети Neo.

Способы применения результатов работы

Настоящая работа является вкладом в мало изученную область исследования применимости оптимизаций gossip-алгоритмов в доверенной среде к организации сетевого общения в одноранговых блокчейн-системах.

Согласно заключениям, изложенным в части 2.4, выделенные характерные особенности применения gossip-протокола в блокчейн-сети Neo являются общими для группы блокчейн-сетей, использующих в качестве консенсусного алгоритма варианты «proof-of-stake», следовательно, могут быть применены в исследованиях эффективности протоколов указанных блокчейн-сетей.

Способы оптимизации gossip-протокола блокчейн-сети Neo, предложенные в части 3.3 также могут быть перенесены на более широкий круг блокчейн-систем, использующих консенсусные алгоритмы семейства «proof-of-stake».

Направления дальнейших исследований

1. Более детальное исследование комбинации техники «infect-upon-contagion» для push-фазы и возможности конфигурации параметра f_{out}.
2. Исследование возможности применения других способов оптимизации gossip-алгоритмов в доверенной среде к организации сетевого общения в одноранговых блокчейн-системах.
3. Изучение проблемы агрегации информации о сети в недоверенной среде и разработка надежных и эффективных методов агрегации сетевой информации в блокчейн-системах.

4. Исследование ключевых особенностей применения gossip-протоколов в блокчейн-системах, использующих консенсусные алгоритмы семейства «proof-of-work».

Список используемых сокращений

dBFT - Delegated Byzantine fault tolerance

LBL - Latency at the block level

LPL - Latency at the peer level

NU - Network utilization

P2P - Peer-to-peer

PoS - Proof of stake

PoW - Proof of work

RPC - Remote procedure call

TTL - Time-to-life

ЦПУ - Центральное процессорное устройство

ОЗУ - Оперативное запоминающее устройство

ЭВМ - Электронно-вычислительная машина

Список литературы

1. Распределенные системы. Принципы и парадигмы / Таненбаум Э, ван Стеен М // СПб.: Питер - 2003.
2. A case for end system multicast, Selected Areas in Communications. / Yang-hua C, Rao S // IEEE Journal - 2002, 1456–1471.
3. A gossip-style failure detection service. / van Renesse R, Minsky Y, Hayden M // Technical Report TR98-1687, Dept. of Computer Science, Cornell University - 1998.
4. A protocol for reliable decentralized conferencing. / Lennox J, Schulzrinne H // In NOSSDAV '03: Proceedings of the 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video. Monterey, CA, USA - 2003.
5. A robust and scalable technology for distributed system monitoring, management, and data mining. / van Renesse R, Birman K, Vogels W // Astrolabe: ACM Transactions on Computer Systems 21(2) - 2003, 164–206.
6. A scalable real-time peer-to-peer system with probabilistic timing assurances. / Fei H, Ravindran B, Jensen ED // RT-P2P: Shanghai, China: Embedded and Ubiquitous Computing, 2008. EUC '08. IEEE/IFIP International Conference - 2008, 97–103.
7. A scalable reliable multicast system for dynamic environments. / Melamed R, Keidar I // Cambridge, MA, USA: Network Computing and Applications, 2004. (NCA 2004). Proceedings. Third IEEE International Symposium - 2004, 5–14.
8. Algebraic gossip: a network coding approach to optimal multiple rumor mongering, Information Theory. / Deb S, Medard M, Choute C // IEEE Transactions on 2006 - 2006, 2486–2507.
9. Analysis of the Evolution of Peer-to-Peer Systems. / Liben-nowell D, Balakrishnan H, Karger D // Proceedings of the Annual ACM Symposium on Principles of Distributed Computing - 2004.

10. Bimodal multicast. / Birman K, Hayden M, Ozkasap O, Xiao Z, Budiu M, and Minsky Y // ACM Trans.Comput. Syst.,17(2) - 1999, 41–88.
11. Continuous Gossip-Based Aggregation through Dynamic Information Aging. / Rapp V, Graffi K // Proceedings - International Conference on Computer Communications and Networks, ICCCN - 2013.
12. Controlling gossip protocol infection pattern using adaptive fanout. / Verma S, Wei Tsang O // Distributed Computing Systems, 2005. ICDCS 2005. Columbus, OH: Proceedings. 25th IEEE International Conference - 2005, 665–674.
13. Dependably scheduling real-time distributable threads in large-scale, unreliable networks. / Kai H, Ravindran B, Jensen ED // RTG-L: Melbourne, Qld: Dependable Computing, 2007. PRDC 2007. 13th Pacific Rim International Symposium - 2007, 314–321.
14. Efficient and adaptive epidemic-style protocols for reliable and scalable multicast, Parallel and Distributed Systems. / Gupta I, Kermarrec A, Ganesh A // IEEE Transactions on 2006 - 2006, 593–605.
15. Efficient reconciliation and flow control for anti-entropy protocols / van Renesse R, Dumitriu D, Gough V, and Thomas C // Proceedings of the 2nd Workshop on Large-Scale Distributed Systems and Middleware, ACM,LADIS - 2008.
16. Epidemic Algorithms for Replicated Database Management / Demers, A et al. // Proc. of 6th ACM Symposium on Principles of Distributed Computing, Vancouver, PODC - 1987.
17. Epidemic broadcast trees. / Leitao J, Pereira J, Rodrigues L // Beijing, China: Reliable Distributed Systems, 2007. SRDS 2007. 26th IEEE International Symposium - 2007, 301–310.
18. Epidemic information dissemination in distributed systems. / Eugster P, Guerraoui R, Kermarrec A, and Massoulié L // IEEE Computer 37 (5) - 2004.
19. Epistemic Protocols for Distributed Gossiping. / Apt K, Grossi D, Hoek W // Electronic Proceedings in Theoretical Computer Science - 2016.

20. Fair and Efficient Gossip in Hyperledger Fabric / Berendea N, Mercier H, Onica E and Rivière E // Proc. of the IEEE ICDCS - 2020.
21. GoCast: Gossip-enhanced overlay multicast for fast and dependable group communication. / Chang R, Ward C // In Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference - 2005, 140–149.
22. Gossip and Epidemic Protocols / Montresor A // Lecture Notes in Artificial Intelligence - 2004, 265–282
23. Gossip-Based Broadcast. / Leitão J, Pereira J, Rodrigues L // Handbook of Peer-to-Peer Networking - 2010, 831-860.
24. Gossip-based aggregation in large dynamic networks. / Jelasity M, Montresor A, and Babaoglu O // ACMTrans. Comput. Syst.,23 - 2005.
25. Gossip-based computation of aggregate information. / Kempe D, Dobra A, Gehrke J // Cambridge, MA, USA: Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium - 2003, 482–491.
26. Gossip-based peer sampling. / Jelasity M, Voulgaris S, Guerraoui R, Kermarrec A, and van Steen M // ACM Trans. Comput. Syst.,25(3) - 2007.
27. Hyperledger fabric: a distributed operating system for permissioned blockchains. / Androulaki E at al. // arXiv preprint arXiv:1801.10228v2 - 2018.
28. Lightweight probabilistic broadcast / Eugster P, Guerraoui R, Handurukande S, Kouznetsov P, and Kermarrec A // ACM Trans. Comput. Syst.,21(4) - 2003, 341–374.
29. Multicast routing in datagram internetworks and extended LANs. / Deering SE, Cheriton DR // ACM Trans Comput Syst - 1990.
30. NEO Delegated Byzantine Fault Tolerance Consensus Mechanism / Monoppo M // Medium - 2018.
31. New directions in cryptography. / Deffie W, Hellman M // IEEE Trans. Inf. Theory IT-22 - Nov. 1976, 644-654.
32. On the complexity of asynchronous gossip. / Georgiou C, Gilbert S, Guerraoui R, Kowalski D // Canada, Toronto: Proceedings of the Twenty-Seventh ACM Symposium on Principles of Distributed Computing - 2008, 135–144.

33. Optimization Scheme of Consensus Mechanism Based on Practical Byzantine Fault Tolerance Algorithm / Zhipeng G, Lulin Y // Researchgate - 2020.
34. Peer-to-peer multipoint video conferencing with layered video. / Akkus I, Ozkasap O, Civanlar M // J Netw Comput Appl - 2011, 137–150.
35. Practical Byzantine Fault Tolerance / Castro M, Liskov B // Third Symposium on Operating Systems Design and Implementation - Feb. 1999.
36. Probabilistic reliable dissemination in large-scale systems. / Kermarrec A, Massouli´e L, and Ganesh A // IEEE Trans. Parallel Distrib. Syst.,14(3) - 2003, 248–258.
37. Push-gossip protocol efficiency with network topology propagation. / Vanin A, Bogatyrev V // Proc. of the 10th Majorov International Conference on Software Engineering and Computer Systems, MICSECS - 2018.
38. Push-pull gossiping for information sharing in peer-to-peer communities. / Mujtaba M // Proc. International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA). Las Vegas - 2003, 1393–1399.
39. RRG: redundancy reduced gossip protocol for real-time N-to-N dynamic group communication. / Luk, V.WH., Wong, A.KS., Lea, CT. et al. // J Internet Serv Appl 4, 14 - 2013.
40. Randomized gossip algorithms, Information Theory. / Boyd S, Ghosh A, Prabhakar B, Shah D // IEEE Transactions on 2006 - 2006, 2508–2530.
41. Randomized rumor spreading. / Karp R, Schindelhauer C, Shenker S, Vocking B // Redondo Beach, CA: Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium - 2000, 565–574.
42. Reaching agreement in the presence of faults. / Pease M, Shostack R, Lamport L // ACM 27, 2 - Apr. 1980, 228-234.
43. Simple gossiping with balls and bins. / Koldehofe B // Stud. Inform. Univ. 3 (1) - 2004, 43–60.
44. Spatial gossip and resource location protocols. / Kempe D, Kleinberg J, and Demers A // J. ACM,51(6) - 2004, 943–967.

45. T-Man: Gossip-based fast overlay topology construction. / Jelasity M, Montresor A, and Babaoglu O // Computer Networks,53(13) - 2009, 2321 – 2339.
46. The Byzantine Generals Problem / Lamport L, Shostack R, and Pease M // SRI International - 1981.
47. The Science of the Blockchain / Wattenhofer R // CreateSpace Independent Publishing; 1st edition - 2016.
48. Towards robust distributed systems. / Brewer E // PODC - 2000.
49. Исходный код проекта NeoBench - Accessed: 2021-04-11. URL: https://github.com/nspcc-dev/neo-bench
50. Исходный код проекта NeoGo - Accessed: 2021-04-11. URL: https://github.com/nspcc-dev/neo-go
51. Bitcoin: A Peer-to-Peer Electronic Cash System. / Nakamoto S // Cryptography Mailing list - 2009. Accessed: 2021-02-04. URL: https://bitcoin.org/bitcoin.pdf
52. Docker - Accessed: 2021-04-23. URL: https://www.docker.com/
53. NEO / Zhang E // Neo whitepaper - 2018. Accessed: 2021-01-26. URL: https://docs.neo.org/docs/en-us/basic/whitepaper.html
54. Neo 3.0.0-preview4 nodes benchmarking / Neo Saint Petersburg Competence Center (Neo SPCC) // Medium - Accessed: 2021-03-15. URL: https://neospcc.medium.com/neo-3-0-0-preview4-nodes-benchmarking-bb4ef291dcca