Содержание

Аннотация

Для достижения консенсуса при выборе следующего блока в блокчейне используется множество различных алгоритмов, большинство из алгоритмов блокчейн консенсуса основано на лидерах, то есть у каждого зафиксированного блока есть автор — лидер. При этом, в блокчейне важна устойчивость к цензуре (censorship-resistance). Одним из способов достичь этого свойства является использование алгоритмов блокчейн консенсуса без лидера, устойчивых к византийскому поведению менее трети участников из заданного списка. В работе разбираются такие алгоритмы. С помощью реализованного симулятора блокчейна было проведено экспериментальное сравнение работы алгоритма DBFT[4] в условиях Византийских атак при разной доле злоумышленников.

Ключевые слова

Блокчейн, консенсус без лидера, византийские отказы, устойчивость к цензуре, сравнительный анализ

1 Введение

Блокчейн - это распределенная база данных, которая использует криптографические методы для защиты информации. Она позволяет пользователям проводить транзакции без потребности в централизованных посредниках (банках, государстве, государственных или коммерческих предприятиях). Эта технология используется для хранения и передачи данных или активов. Технически, блокчейн состоит из цепочки блоков, состоящих из транзакций пользователей. Блоки связаны с предыдущими через хеши, что позволяет гарантировать сохранность транзакций и блоков, попавших в цепочку, в первоначальном виде.

Для достижения консенсуса при выборе следующего блока в блокчейне используется множество различных алгоритмов, которые можно разделить на 2 класса:

- **экономические**[1], в которых есть условие, при котором участник может добавить блок в обмен на какую-то награду

- **математические**, основанные на алгоритме (PBFT)[3] и предполагающие менее трети недобросовестных участников с византийским поведением. Чаще всего такие алгоритмы используются в приватных блокчейнах.

Помимо обычных требований к алгоритму консенсуса (agreement, termination, validity, integrity), в блокчейнах таким алгоритмам также нужно быть устойчивым к цензуре (censorship-resistance). Дело в том, что большинство протоколов основаны на лидерах, то есть у каждого зафиксированного блока есть автор — лидер. В таких условиях сложно предоставить гарантию, что любая валидная транзакция клиента когда-нибудь будет включена в блокчейн, так как отдельные участники, часто являющиеся авторами блоков, могут намеренно игнорировать и не включать некоторые неугодные им транзакции. Одним из решений этой проблемы являются протоколы блокчейн консенсуса без лидера, в которых блоки генерируются на лету из обязательств всех участников. Так никакой участник самостоятельно не может сильно повлиять на итоговый блок.

Такие алгоритмы блокчейн консенсуса без лидера принадлежат упомянутому выше классу "математических"алгоритмов. Кроме устойчивости к цензуре, они обладают другими полезными свойствами: нет узкого места в виде лидера, нет проблемы из-за медленного лидера. Такие алгоритмы блокчейн консенсуса без лидера и будут рассматриваться в этой курсовой работе.

Формально: рассматриваются условия, приближенные к приватному блокчейну. Есть N заранее известных участников сети, f из которых могут иметь византийское поведение, причем $N \geqslant 3f + 1$. Все эти участники обмениваются транзакциями и другими

[1]Также называются lottery-based. Примеры: proof-of-work(PoW)[12], proof-of-stake(PoS), и другие алгоритмы, название которых начинается с "Proof-of"

сообщениями, нужными для достижения консенсуса при выборе следующего блока. Используется алгоритм консенсуса без лидера. Рассматривается частично-синхронная или асинхронная модель сети. Процессы с византийским поведением могут вести себя произвольно: слать произвольные сообщения или вообще не отвечать, производить произвольные операции со своей копией блокчейна. При этом, византийские процессы имеют ограничения на скорость вычислений, как и добросовестные участники. Никакой византийский процесс не может выдавать себя за другого участника, не может подделывать чужую цифровую подпись. Также они могут влиять на сеть между другими участниками, но не способны задерживать их сообщения навсегда.

Цель этой курсовой работы - проанализировать и сравнить такие алгоритмы блокчейн консенсуса без лидера в условиях различных византийских атак.

В задачи входит ознакомление с предметной областью и алгоритмами консенсуса, реализация одного или некоторых таких алгоритмов, моделирование и проведение экспериментов, сравнение работы алгоритмов в условиях различных византийских атак, построение графиков и описание результатов в итоговом отчете.

Актуальность и значимость

Блокчейны сейчас, в основном, используют алгоритмы с лидером. Как уже было сказано выше, такие алгоритмы имеют ряд проблем, в том числе отсутствие гарантии на добавление каждой валидной транзакции (устойчивости к цензуре). Для перехода на алгоритм консенсуса без лидера нужно прежде тщательно проанализировать возможные риски разных таких алгоритмов консенсуса, чтобы определить лучших представителей. При этом, в статьях, представляющих такие новые алгоритмы, часто нет сравнений результатов работы алгоритма в условиях Византийских атак. В основном, там рассчитывают параметры алгоритмов только для хорошего случая, когда злоумышленников вообще нет, либо есть только fail-stop отказы. Часто не проверяют, как параметры алгоритма зависят от доли злоумышленников. Именно такие аспекты этих алгоритмов предполагается разобрать в этой работе, что обуславливает её актуальность и значимость.

2 Существующие работы и решения

Все такие алгоритмы консенсуса имеют примерно одинаковый принцип: сначала участники обмениваются транзакциями, собирают какое-то множество их, а затем для каждой транзакции запускается бинарный алгоритм консенсуса, определяющий будет ли включена в итоговый блок эта транзакция. По итогу получается блок, сгенерированный из обязательств всех участников.

В основе большинства таких алгоритмов блокчейн консенсуса без лидера лежат 2 алгоритма, работающие в присутствии византийских отказов: Reliable Broadcast(RB)[2] и

Binary Byzantine Agreement(BA)[11], алгоритм консенсуса, позволяющий прийти к согласию относительно бинарной величины.

Democratic Byzantine fault tolerance (DBFT)[4] - алгоритм, основанный на 2 протоколах, указанных выше (RB и BA). Работает в условиях частично-синхронной модели сети. Авторы модифицируют алгоритм BA так, что он становится детерминированным, а также оптимизируют сходимость нового алгоритма с помощью добавления слабых координаторов. Эти координаторы детерминировано назначаются каждый раунд и отправляют всем своё текущее значение величины, для которой надо прийти к консенсусу. Важно, что такие координаторы не являются лидерами: они не могут повлиять на итоговое значение, даже если являются византийскими. Эти координаторы нужны только для того, чтобы помочь сойтись алгоритму быстрее. В статье проводят эксперименты над бинарным консенсусом в условиях нескольких Византийских атак и сравнивают с бинарным консенсусом из HoneyBadgerBFT[10](будет разобран ниже), но не проверяют, как работа алгоритма зависит от доли злоумышленников. В статье о Red belly blockchain[5] формализуют и реализовывают то, как перейти от DBFT, к полноценному алгоритму консенсуса для блокчейна. Проводят эксперименты, сравнивая с HoneyBadgerBFT[10](будет разобран ниже), но замеряют в условиях Византийских атак только при максимальном числе злоумышленников, при этом Византийские атаки нацелены на RB, чтобы увеличить throughput. Зависимость от доли злоумышленников проверяют только при fail-stop отказах

HoneyBadgerBFT(HBBFT)[10] также является алгоритмом, основанном на последовательном применении RB и BA. Особенностью является то, что он работает в асинхронной модели сети. При этом, это не противоречит теореме FLP[7], так как алгоритм вероятностный (авторы оставили вероятностные вариант алгоритма BA). Для устойчивости к цензуре транзакции шифруются с помощью threshold encryption[1]. Получается, нужен добросовестный сервис, который будет выдавать эти секретные ключи всем на входе, что легко воспроизводится в контексте приватных блокчейнов, так как там и так нужно пройти через похожий сервис, чтобы попасть в сеть. Также авторы этого алгоритма используют не обычный RB, а RB со стирающими кодами(erasure codes) на основе кодов Рида-Соломона, оптимизируя таким образом Reliable Broadcast. Проводят эксперименты с постоянной долей fail-stop злоумышленников (четверть от всех участников, что значительно меньше теоретического максимума).

BEAT[6] - семейство из 5 протоколов, в основе которых лежит HBBFT. Каждый оптимизирует какое-то свойство HBBFT (throughput, latency). Для достижения оптимизаций использовали различные алгебраические техники. Например, использовали стирающие коды pyramid codes[9] вместо кодов Рида-Соломона. Реализовывают все свои алгоритмы и сравнивают их между собой и с HBBFT. Также, не описывают модель поведения отказывающих участников. Приводят графики с очень маленьким числом $f \in \{1, 2\}$.

В статье[8] тоже улучшают HBBFT, но акцентируют внимание на другом его недостатке - необходимость иметь централизованный добросовестный сервис на входе в сеть.

Авторы заменяют работу этого сервиса на свой протокол ABFT-Beacon, который выполняет эту задачу распределённо. Экспериментальных данных замеров и таких сравнений своего алгоритма консенсуса с другими не приводят.

Не все алгоритмы основаны исключительно на RB и BA. Например, в статье[14] описывается семейство вероятностных алгоритмов Snowflake. Идея заключается в том, что в каждом раунде участник отправляет сообщения не всем участникам, а только случайно выбранному подмножеству. Благодаря этому, протокол становится недетерминированным, но вместе с этим приобретает хорошие свойства. Сильно уменьшают объем необходимой коммуникации участника со всеми остальными участниками до константной в каждом раунде, а всего раундов $O(\log N)$. В предыдущих алгоритмах каждому участнику необходимо было отправить линейное число сообщений в каждом раунде. Таким образом,у протокола лучше масштабируемость, а также он может работать даже в случае недоступности некоторых участников. Реализовывают алгоритм и проводят эксперименты, но рассматривают только атаку, в которой отправляются некорректные транзакции, приводящие к проблеме двойного расходования(double spending attack).

Таким образом, были рассмотрены алгоритмы более всего подходящие под поставленную в начале задачу. Почти все из них опираются на вероятностные подходы, и только DBFT остается полностью детерминированным, поэтому далее речь будет идти преимущественно про него. Некоторые из авторов статей приводили экспериментальные данные, иногда даже сравнивали с другими алгоритмами. Но, как уже было упомянуто выше, такие сравнения сделаны только для узкого случая, то есть для определенного набора условий и параметров, плохо показывающих, как ухудшается работа алгоритмов при византийских отказах. Не было экспериментов, показывающих замеры при различной доле злоумышленников с Византийским поведением (не fail-stop). Отдельных исследований, сравнивающих эти алгоритмы, найдено не было.

3 Описание алгоритмов

Рассмотрим подробнее алгоритм DBFT и его составляющие (RB и BA).

3.1 Binary Byzantine Agreement (BA)

Алгоритм BA - алгоритм консенсуса для бинарной величины, как уже было сказано выше. Допускает асинхронную модель сети. Алгоритм состоит из раундов, в каждом из которых нужно $O(N^2)$ сообщений. Изначально у участника есть его значение est. В начале каждого раунда у участника есть пустое множество bin_values. Раунд состоит из 3 фаз:

- BV-broadcast: алгоритм для бинарных величин с $O(N^2)$ сообщений, псевдокод которого на Рисунке 1. В конечном счёте гарантируется, что множество bin_values

operation BV_broadcast MSG(v_i) **is**
(01) broadcast B_VAL(v_i).

when B_VAL(v) **is received**
(02) **if** (B_VAL(v) received from $(t+1)$ different processes and B_VAL(v) not yet broadcast)
(03) **then** broadcast B_VAL(v) % a process echoes a value only once %
(04) **end if**;
(05) **if** (B_VAL(v) received from $(2t+1)$ different processes)
(06) **then** $bin\_values_i \leftarrow bin\_values_i \cup \{v\}$ % local delivery of a value %
(07) **end if**.

Рис. 1: Псевдокод алгоритма BV-broadcast из статьи[11]

окажется непустым, одинаковым для всех корректных участников, а также гарантируется, что в него попадут только значения, которые изначально были хотя бы у $(f+1)$ участника.

- Дополнительный broadcast множества bin_values, когда оно становится непустым. Участник ожидает получения сообщений от $(n-f)$ различных участников, удовлетворяющих следующему условию: $\exists$ множество $values$, равное объединению множеств bin_values из этих сообщений, и это множество $values$ содержится в множестве bin_values участника.

- В третьей фазе участник использует common coin(распределенный генератор случайных бит, как в статье [13]), получая бит s. Если $values$ состоит из одного элемента, то участник кладёт это значение в est. Если же это значение ещё и совпадает с s, то s считается результатом алгоритма. Если множество $values$ состоит из двух элементов, то участник присваивает $est = s$. Участник переходит в следующий раунд.

Матожидание числа раундов, после которых все участники получат результат, равно 4.

Переход от BA к PSync

Авторы алгоритма DBFT представили свою модификацию алгоритма BA, назвав её PSync. Главное, они убрали рандомизированность, а именно, теперь в третьей фазе раунда с порядковым номером r бит $s = r \mod 2$. Также теперь между 1 и 2 фазой алгоритма выбирается слабый координатор (участник с номером $r \mod N$). Он делает дополнительный broadcast своего bin_values, как только тот становится непустым. Другие участники при получении этого сообщения от координатора приоритизируют его содержимое над своим bin_values, но только если оно содержится в их bin_values. Кроме того, авторы предполагают, что модель сети частично синхронная, поэтому ставят таймер в начале первогой фазы раунда. Только по истечению этого таймера участник переходит к выполнению второй фазы, даже если его множество bin_values стало непустым раньше. Это

сделано, чтобы выравнивать текущий раунд у разных участников.

Таким образом, асимптотика числа сообщений в каждом раунде осталась прежней - $O(N^2)$. По сравнению с BA, в PSync отсутствует лишняя коммуникация между участниками, нужная для common coin, но добавлены сообщения от координаторов.

3.2 Reliable Broadast (RB)

Алгоритм позволяет сделать broadcast любого значения в присутствии f злоумышленников, где $N > 3f$. Гарантируется доставка значения всеми корректными участниками, если участник, инициирующий RB, корректный. Если же участник, инициирующий RB, является злоумышленником, то гарантируется, что значение будет либо доставлено всеми корректными участниками, либо не доставлено никем из них. Алгоритм имеет 3 этапа, завершается максимум через 4 задержки сети, асимптотика числа сообщений - $O(N^2)$.

3.3 DBFT

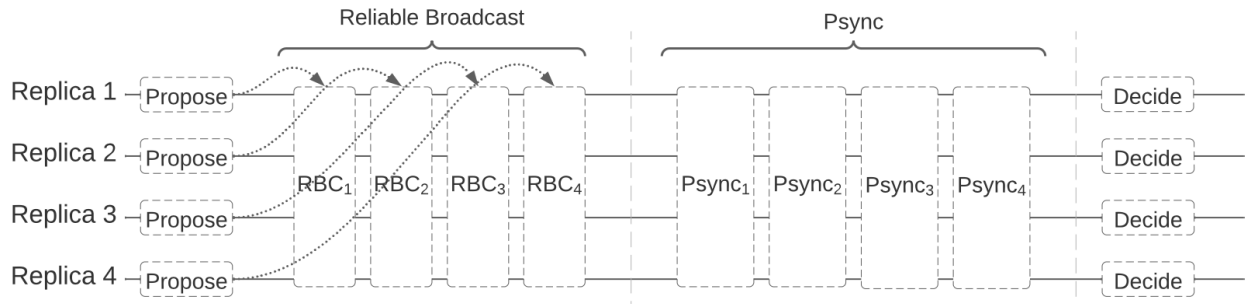

Рис. 2: Схема компонентов протокола DBFT из статьи[15]. RB и RBC - один и тот же алгоритм

Алгоритм DBFT - алгоритм консенсуса для массива значений (в нашем случае блока транзакций). Состоит из N алгоритмов RB и N алгоритмов PSync. Каждый участник выбирает несколько транзакций и отправляет его в алгоритм RB под своим индексом. Когда i-ый алгоритм RB доставляет транзакции, запускается i-ый алгоритм PSync с начальным значением 1. Если же $(N - f)$ экземпляров алгоритма PSync пришли к консенсусу, то все оставшиеся экземпляры алгоритма PSync (то есть те, для которых не были доставлены соответствующие транзакции алгоритмами RB) запускаются с начальным значением 0. В конце концов, участник ждёт, пока не придут к консенсусу все экземпляры PSync, а также пока не будут доставлены все значения, для которых соответствующий PSync пришёл к значению 1 при достижении консенсуса.

Таким образом, асимптотическое число сообщений у алгоритма DBFT - $O(N^3)$. А в лучшем случае (синхронная сеть, все участники честные) ему требуется всего 4 задержки сети: 3 для RB и 1 для PSync (при комбинации RB с PSync применяется оптимизация, поз-

воляющая пропустить первую фазу (BV-broadcast) первого раунда, после чего алгоритм сразу же достигает консенсуса с конечным результатом 1).

4 Симулятор

Для проведения экспериментального анализа был запрограмирован симулятор, код которого можно найти здесь: https://github.com/afnastya/leaderless-blockchain-consensus. Написан на языке C++. Симуляция исполняется на одном компьютере, каждый участник однопоточный, и исполняется в своём потоке. Такой дизайн выбран, чтобы иметь контроль над сетью: устанавливать произвольные задержки, перемешивать сообщения.

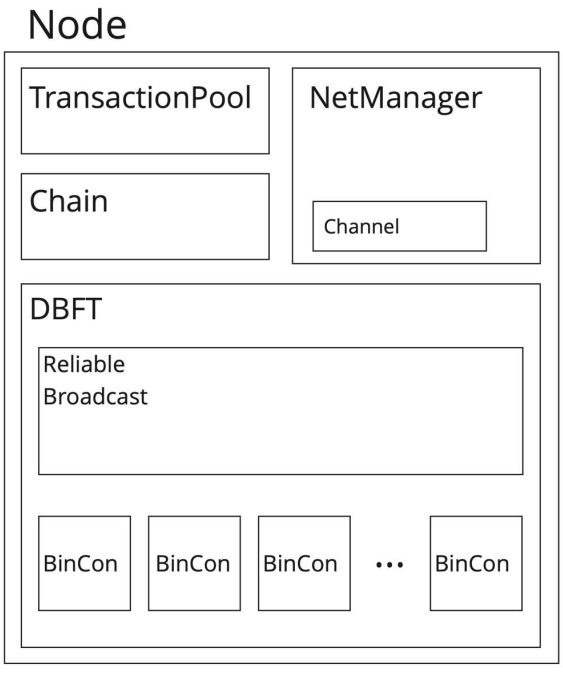

Рис. 3: Схема участника сети в симуляторе

На Рисунке 3 представлена схема участника сети. Разберём основные компоненты:

- **TransactionPool.** Хранилище транзакций, полученных участником, которые ещё не участвовали в алгоритме консенсуса. Отсюда участник берёт транзакции, когда запускает алгоритм DBFT для поиска нового блока. Транзакции в симуляторе представляют собой целые числа и не несут никакой информации (можно считать, что это хеши реальных транзакций). В проведенных симуляциях транзакции добавляются в TransactionPool'ы участников перед запуском всех участников. Они генерируются случайно, каждая отправляется в TransactionPool $f + 1$ участнику. Таким образом, каждая транзакция точно окажется хотя бы у одного честного участника. Чтобы уменьшить число дубликатов транзакций в блоках, используется алгоритм их сортировки из Red Belly Blockchain[5]: один из этих $f + 1$ участников является *primary*

для конкретной транзакции, в TransactionPool участника транзакции сортируются по возрасту и по тому, является ли участник *primary* для них.

- **Chain.** Локальная копия блокчейна участника.

- **NetManager.** Компонент, ответственный за взаимодействие с другими участниками. Все участники сети заранее известны, они общаются с помощью сообщений через каналы. У каждого участника есть свой канал и ссылки на каналы всех других участников. Подробно о различных видах каналов сети в симуляторе будет сказано в Разделе 4.1.

- **DBFT.** Реализация алгоритма консенсуса DBFT[4]. Внутри: класс RealibleBroadcast, реализующий алгоритм с одноименным названием для распространения транзакций, и вектор из объектов BinCon, реализующих бинарный консенсус для каждого блока транзакций, предложенного каким-либо участником. В симуляторе в блок каждый участник может предложить от 0 до *batch* транзакций (параметр *batch* может быть установлен в конфигурации симулятора, по умолчанию равен 10). После исполнения алгоритма DBFT есть фаза *reconciliation* для сборки полного готового блока, который добавится в блокчейн (как в Red Belly Blockchain[5]): проходим по всем транзакциям участников, которые решено добавить в блок по итогам DBFT, начиная с участника с номером остатка от деления высоты блока на число участников; добавляем в блок транзакцию, если она не противоречит уже добавленным. Так как у нас транзакции ничего не значат, а являются просто хешами, то в симуляциях все транзакции считаются противоречащими друг другу только, если равны.

4.1 Сеть

В симуляторе есть несколько вариантов сети. Разберём подробнее возможные конфигурации:

- **ManualNetwork.** Каналы между участниками здесь - потокобезопасные очереди сообщений. Сеть представляет собой отдельный поток со своей очередью сообщений, у участников также есть свои собственные очереди, из которых они берут пришедшие сообщения. При отправке сообщения участник добавляет его в очередь сети. Полезные свойства такой "ручной"сети в том, что её можно исполнять пошагово, или можно перемешивать очередь. Именно поэтому эта сеть использовалась для тестирования алгоритмов.

- **Network.** Каналы такие же, как и в предыдущем варианте сети. Отличие в отсутствии дополнительной очереди у сети, и отдельного потока для неё. Теперь при отправке участники добавляют своё сообщение напрямую в канал получателя. Сеть также использовалась для тестирования алгоритмов.

- **TimerNetwork.** Каналы между участниками здесь - планировщики из библиотеки Boost.Asio. Когда сообщение отправляется, в планировщик (который принадлежит получателю) добавляется таймер с случайной задержкой (из равномерного распределения) для этого конкретного сообщения. По истечению таймера выполняется функция, обрабатывающая это сообщение. Такая сеть использовалась непосредственно для экспериментов.

5 Модели экспериментов и Византийские атаки

Эксперименты проводились в конфигурациях с полностью всеми честными участниками и в различным числом злоумышленников от 0 до f. Злоумышленники были представлены как Fail-Stop отказы или с Византийским поведением.

Византийское поведение

Было просимулировано несколько моделей византийского поведения.

Первое поведение - **Rejector** - сразу отправляет во все бинарные консенсусы все сообщения с значением 0 на всех этапах в достаточное число раундов, не дожидаясь чужих сообщений. Византийские координаторы также отсылают всем нули. Цель этого поведения - заставить всех не принимать никакие транзакции, получив пустые блоки.

Второй поведение - **BinConCrasher** - сразу отправляет во все бинарные консенсусы все сообщения с обоими значениями (и 0, и 1) на всех этапах в достаточное число раундов, не дожидаясь чужих сообщений. Византийские координаторы отсылают каждому участнику случайное значение. Цель этого поведения - не дать сойтись алгоритму бинарного консенсуса.

Для большего эффекта, Византийские участники имеют номера от 0 до $(f-1)$ включительно, то есть, на самом деле, скорее всего все координаторы в алгоритмах окажутся Византийскими.

Пусть Византийские участники не предлагают никаких транзакций и не участвуют в RB. Заметим, что в указанных условиях такие атаки не будут оказывать желаемого воздействия на бинарные консенсусы. Действительно, даже если число злоумышленников максимально, то есть f, при обычном протекании симуляции все остальные $N - f$ честные участники завершат свои (то есть под индексами от f до $N - 1$ включительно) экземпляры алгоритма RB и начнут все соответствующие бинарные консенсусы со значениями 1. Тогда распределение начальных значений будет следующим: либо f нулей и $2f + 1$ единиц (при Rejector), либо f нулей и $3f + 1$ единиц (при BinConCrasher). А для того, чтобы значение 0 попало хотя бы в один bin_values и начало как-то влиять на ход алгоритма, количество значений 0 должно быть хотя бы $f + 1$ (Следует из свойств алгоритма BV-broadcast, о котором говорилось в Разделе 3.1). Таким образом, хотя бы один

честный участник должен начать бинарный консенсус какого-то честного участника со значением 0. Это может произойти только если число завершившихся экземпляров RB у этого честного участника окажется $\geq N - f$, при этом экземпляр RB, соответствующий этому бинарному консенсусу ещё не завершится. Но тогда среди этих завершившихся $N - f$ RB должен быть хотя бы один RB, принадлежащий византийскому участнику (при максимальном числе византийских участников [2]). Таким образом, чтобы гарантировано получить воздействие такими Византийскими участниками на бинарный консенсус, нужно чтобы эти Византийские участники также предлагали какие-то транзакции в свои RB. При этом неважно, что они предлагают (возможно, это какой-то мусор или транзакции, нужные им), главное, чтоб их экземпляры RB завершились.

Также нужно гарантировать, чтобы у честных участников некоторые RB не завершались, пока не завершатся $(N - f)$, бинарных консенсусов и эти честные участники не запустят оставшиеся бинарные консенсусы со значениями 0. Чтобы это гарантировать, в соответствующих экспериментах участники с индексами от f до $2f - 1$ включительно не получали сообщения "READY"(сообщение из третьего этапа RB) для бинарных консенсусов с индексами от f до $2f - 1$. Это было сделано на уровне канала, такие сообщения перехватывались и хранились в отдельном массиве, пока в канал к участнику не попадало сообщение от кого-то из этих f до $2f - 1$ участников из второй фазы бинарного консенсуса с индексом f до $2f - 1$ (так можно понять, что все остальные бинарные консенсусы уже закончились, и бинарные консенсусы с индексами от f до $2f - 1$ были запущены с начальными значениями 0, чего мы и добивались). В результате, экземпляров бинарных консенсусов с индексами от f до $2f - 1$ будет распределение начальных значений будет следующим: либо $2f$ нулей и $f + 1$ единиц (при Rejector), либо $2f$ нулей и $2f + 1$ единиц (при BinConCrasher). Заметим, что такие задержки сообщений не противоречат модели алгоритмов, так как сеть частично синхронная.

6 Результаты экспериментов

Замеры экспериментов были проведены на компьютере с 32 ядрами. Число участников сети от 4 до 31. Каждый эксперимент повторялся по 100 раз, взяты средние значения времени, также на графиках есть стандартное отклонение времени. На всех графиках время указано в секундах. Средняя задержка сети равна 10ms. Если не указано, то $batch = 10$, а число злоумышленников максимальное, то есть равно f, где $N = 3f + 1$.

[2]При числе византийских участников меньше максимального всё ещё возможно достичь таких условий, что RB честного участника не успеет завершиться, и соответствующий бинарный консенсус некоторые честные участники начнут со значением 0.

6.1 Без злоумышленников

Для начала замерим работу алгоритма без злоумышленников и каких-либо отказов. Зависимость времени поиска блока от числа участников сети можно увидеть на Рисунке 4, значения лежат в интервале от 0.04 до 0.09 секунд. Как следует из асимптотики алгоритма, которая упоминалась в Разделе 3.3, зависимость должна быть кубической.

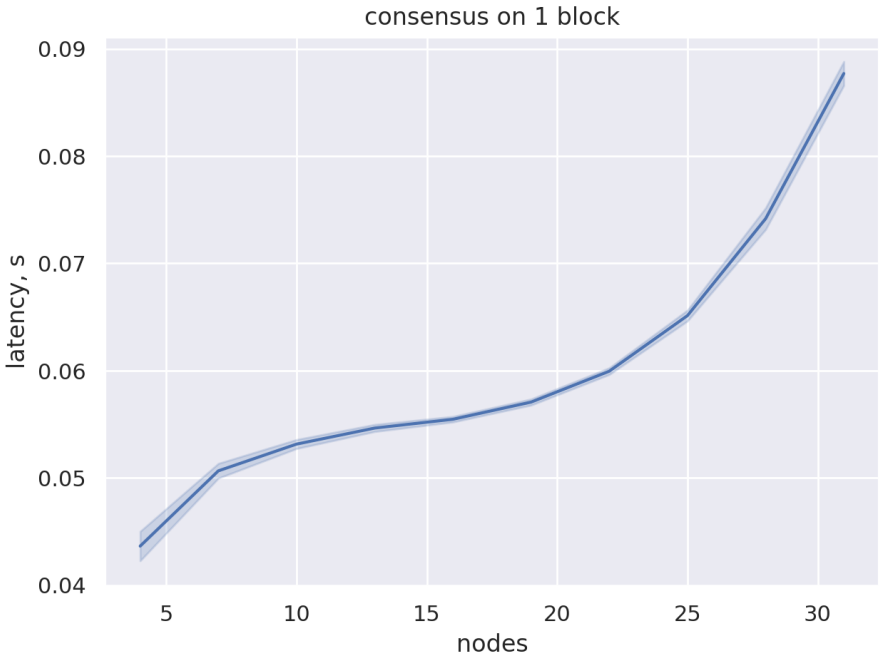

Рис. 4: Время достижения консенсуса для 1 блока в зависимости от числа участников сети

На Рисунке 5 показаны результаты эксперимента при константном числе участников - 16. Показана зависимость времени поиска блока от размера максимального числа транзакции, которые может предложить один участник. На самом деле, в симуляторе в данном случае не увеличивается число сообщений, увеличивается только размер сообщений в алгоритме RB и время, которое требуется, чтобы достать транзакции из TransactionPool и записать эти транзакции в финальный блок на этапе reconciliation. В реальном блокчейне время при увеличении значения *batch* будет вырастать гораздо сильнее за счёт увеличенных расходов времени на верификацию транзакций и reconciliation. Так как наши транзакции не имеют смысловой нагрузки, время вырастает несильно, и, как и ожидалось, линейно.

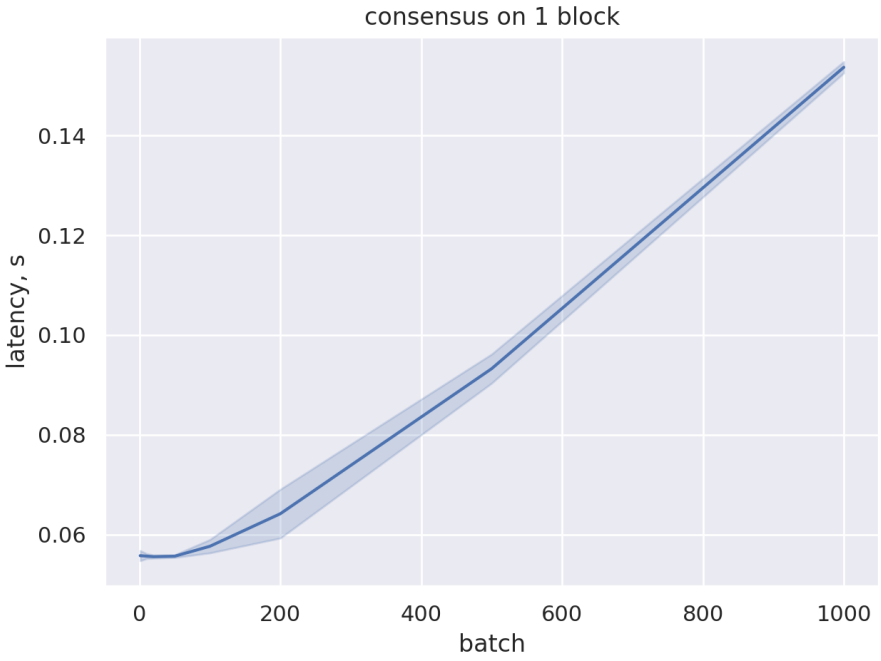

Рис. 5: Время достижения консенсуса для 1 блока в зависимости от максимального числа транзакций, которое может предложить 1 участник(то есть размера *batch*). Число участников сети константное, равно 16

6.2 Византийские атаки

Далее рассмотрим как зависит время достижения консенсуса для 1 блока от числа участников в условии Византийских атак. Здесь разные кривые (кроме Fair) показывают результаты для разного поведения злоумышленников. Число злоумышленников максимально. Кривая Fair - та же кривая, что и на Рисунке 4.

Видно, что кривая FailStop (то есть f участников никак не участвуют в алгоритме) похожа по виду на Fair, как будто бы получена из неё параллельным сдвигом на 0.07 секунд. Не удивительно, ведь FailStop отказы никак не увеличивают асимптотику алгоритма или число раундов. Отличие FailStop от Fair в том, что участнику приходится ждать сообщений от всех остальных честных участников прежде чем перейти к следующей фазе алгоритмов, когда при Fair он ждёт максимум $N - f$ самых быстрых. Две верхние кривые показывают работу алгоритмов при f участниках с Византийским поведением, располагаются выше, так как в таком случае в некоторых экземплярах алгоритмов PSync увеличивается число раундов на 1-2, относительно работы при всех честных участниках. Кривая BinConCrasher располагается выше, так как при ней больше число сообщений, из-за того что Византийские участники предлагают сразу оба значения. Относительно случая со всеми честными участниками, при Византийских атаках получилось достичь увеличения времени в 2-2.5 раза.

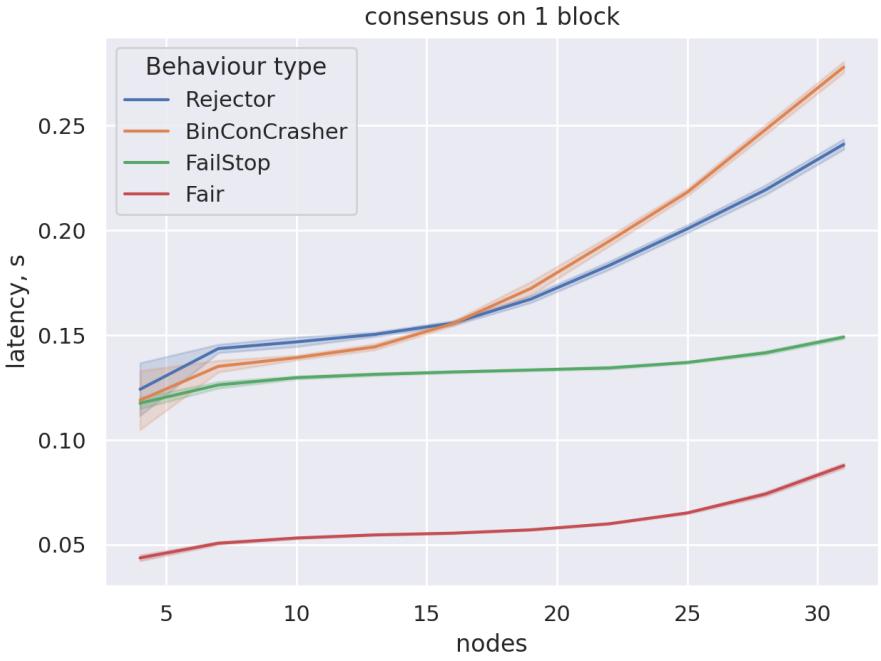

Рис. 6: Время достижения консенсуса для 1 блока в зависимости от числа участников сети. Число злоумышленников максимально.

6.3 Различная доля злоумышленников

Рассмотрим результаты экспериментов при постоянном числе участников сети и различной доле злоумышленников на Рисунке 7.

Видим, что работа алгоритма при Fail-Stop отказах уменьшается плавно, время увеличивается максимум в 2 раза. При Византийских же атаках время увеличивается максимум примерно в 2.5 раза. Заметим, что кривые начинаются в разных точках, при всех честных участниках у кривых с Византийскими атаками время поиска блока на 0.005 секунд больше. Это происходит из-за тех перехватов сообщений в экспериментах с Византийскими атаками, о которых рассказывалось в Разделе 5. В результате, некоторые бинарные консенсусы начинаются позже, и поэтому время поиска блока увеличивается.

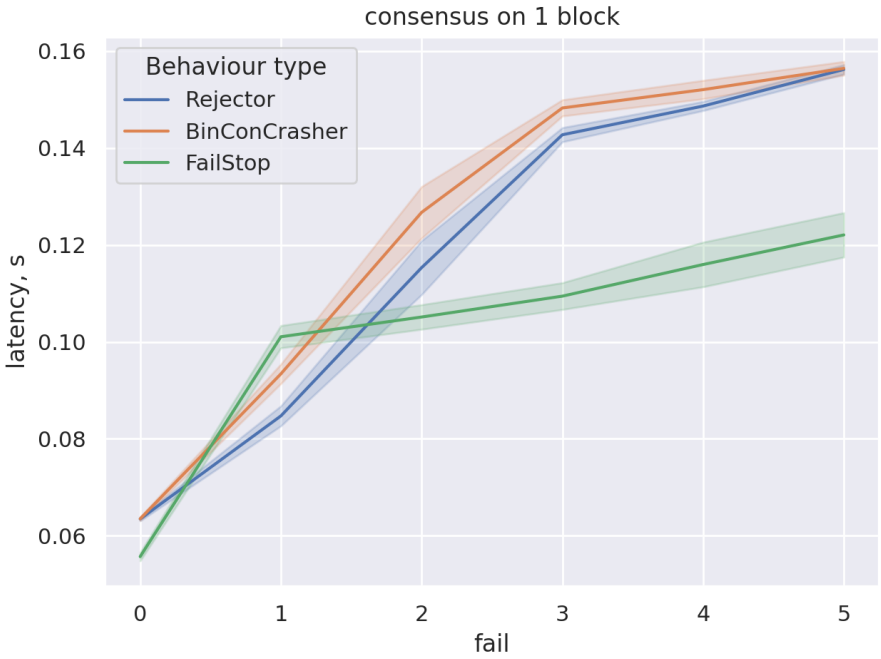

Рис. 7: Время достижения консенсуса для 1 блока в зависимости от числа злоумышленников. Число участников сети константное, равно 16

7 Заключение

Таким образом, в работе было приведено сравнение алгоритма DBFT в нормальных условиях и при различных отказах. Было предложено и обосновано несколько видов Византийского поведения, и сравнены результаты замеров алгоритмов при различной доле Византийских участников с таким поведением. В результате получилось достичь увеличения времени работы алгоритма в 2 - 2.5 раза. В дальнейшем можно продолжить реализацию других алгоритмов для симулятора и сравнить какой из них справляется лучше. Конечно, интересны больше детерминированные алгоритмы, поэтому также можно продолжить поиск таких.

Список литературы

[1] Joonsang Baek and Yuliang Zheng. Simple and efficient threshold cryptosystem from the gap diffie-hellman group. In *GLOBECOM'03. IEEE Global Telecommunications Conference (IEEE Cat. No. 03CH37489)*, volume 3, pages 1491–1495. IEEE, 2003.

[2] Gabriel Bracha. Asynchronous byzantine agreement protocols. *Information and Computation*, 75(2):130–143, 1987.

[3] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OsDI*, volume 99, pages 173–186, 1999.

[4] Tyler Crain, Vincent Gramoli, Mikel Larrea, and Michel Raynal. Dbft: Efficient leaderless byzantine consensus and its application to blockchains. In *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, pages 1–8. IEEE, 2018.

[5] Tyler Crain, Christopher Natoli, and Vincent Gramoli. Red belly: A secure, fair and scalable open blockchain. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 466–483. IEEE, 2021.

[6] Sisi Duan, Michael K Reiter, and Haibin Zhang. Beat: Asynchronous bft made practical. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 2028–2041, 2018.

[7] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.

[8] Adam Gągol, Damian Leśniak, Damian Straszak, and Michał Świętek. Aleph: Efficient atomic broadcast in asynchronous networks with byzantine nodes. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 214–228, 2019.

[9] Cheng Huang, Minghua Chen, and Jin Li. Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems. *ACM Transactions on Storage (TOS)*, 9(1):1–28, 2013.

[10] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The honey badger of bft protocols. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 31–42, 2016.

[11] Achour Mostéfaoui, Hamouma Moumen, and Michel Raynal. Signature-free asynchronous byzantine consensus with t< n/3 and o (n2) messages. In *Proceedings of the 2014 ACM symposium on Principles of distributed computing*, pages 2–9, 2014.

[12] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, page 21260, 2008.

[13] Michael O Rabin. Randomized byzantine generals. In *24th annual symposium on foundations of computer science (sfcs 1983)*, pages 403–409. IEEE, 1983.

[14] Team Rocket, Maofan Yin, Kevin Sekniqi, Robbert van Renesse, and Emin Gün Sirer. Scalable and probabilistic leaderless bft consensus through metastability. *arXiv preprint arXiv:1906.08936*, 2019.

[15] Gengrui Zhang, Fei Pan, Michael Dang'ana, Yunhao Mao, Shashank Motepalli, Shiquan Zhang, and Hans-Arno Jacobsen. Reaching consensus in the byzantine empire: A comprehensive review of bft consensus algorithms. *arXiv preprint arXiv:2204.03181*, 2022.