ВВЕДЕНИЕ

Базы данных — это ключевой элемент любой автоматизированной системы. Вне зависимости от уровня доступа к данным вопрос о целостности и доступности данных всегда остаётся актуальным. Наиболее уязвимыми в этом отношении являются централизованные решения в силу существования единой точки принятия решений. Автоматизированные системы, построенные по таким принципам, в своём большинстве имеют один главный недостаток — необходимость доверия системе. В ряде критически важных автоматизированных систем учёта, таких как электронное голосование, учёт финансов, системы репутации, электронные аукционы, системы страхования, принципы централизации базы данных и аппарата принятия решений могут противоречить требованиям безопасности. По многим причинам такие сервисы, как централизованные социальные сети, остаются уязвимыми с точки зрения доступности и целостности данных.

Архитектура децентрализованных баз данных имеет явные преимущества в этом отношении. Благодаря принципам, заложенным в её основу, такая архитектура позволяет удовлетворять значительно более высоким требованиям к безопасности по сравнению с централизованной. Наиболее технологически важной составляющей таких баз данных является механизм синхронизации.

В данной работе рассмотрены протоколы распределённых баз данных на основе технологии blockchain. Согласно этой технологии, база данных представляет собой сеть из произвольного числа рабочих станций, каждая из которых сохраняет полную копию базы данных. Далее под распределённой базой данных будет пониматься множество копий базы данных и механизм их синхронизации. В общем случае осуществлять запись в базу может кто угодно, поскольку сеть базы данных существует поверх глобальной сети. За счёт p2p

сетевого соединения рабочие станции синхронизируют состояние своей копии базы данных друг с другом. При этом такие протоколы синхронизации распределённых баз данных, как Bitcoin, Ripple, Bitshares, Ethereum рассчитаны на ситуацию, в которой каждая отдельная рабочая станция не доверяет ни одной другой.

1 АРХИТЕКТУРА РАСПРЕДЕЛЁННЫХ БАЗ ДАННЫХ НА ОСНОВЕ ТЕХНОЛОГИИ BLOCKCHAIN

1.1 Архитектура базы данных

Логически распределённая база данных на основе технологии blockchain состоит из криптографически связанных блоков данных. Связность реализована за счёт криптографически стойкой хэш-функции таким образом, что каждый следующий блок данных содержит хэш-значение предыдущего блока. За счёт этого механизма целостность базы данных может быть легко проверена. Таким образом, база данных представляет собой цепочку блоков от нулевого (Genesis Block) до текущего (Last Block).

Поскольку такая архитектура базы данных позволяет достичь только односторонней связности блоков данных, то проверка целостности имеет смысл только в одном направлении — от текущего блока к нулевому.

Таким образом, первопричина проблемы синхронизации базы данных на основе blockchain состоит в том, что каждая отдельно взятая копия базы может быть модифицирована. Если один из блоков был модифицирован, то целостность базы нарушена. С целью восстановления целостности все последующие блоки данных могут быть поочерёдно модифицированы заменой хэш-значения предыдущих блоков. Таким образом, история локальной копии базы данных может быть переписана без потери целостности и, как результат, иметь пагубное влияние на всю базу данных в целом. Для решения этой проблемы существует дополнительный механизм, называемый консенсус. С помощью этого механизма все копии распределённой базы данных гарантированно имеют одинаковую историю.

Смысл такого механизма, как консенсус, заключается в том, чтобы доказать неизменяемость текущего блока данных (последних блоков данных). Имея такое

доказательство, можно утверждать, что все предыдущие блоки цепочки также останутся неизменны, поскольку они связанны математически.

Очевидным фактом является то, что добавленные в базу данные не могут быть позже изменены или удалены. На рисунке 1.1 схематически изображено логическое представление базы данных blockchain.

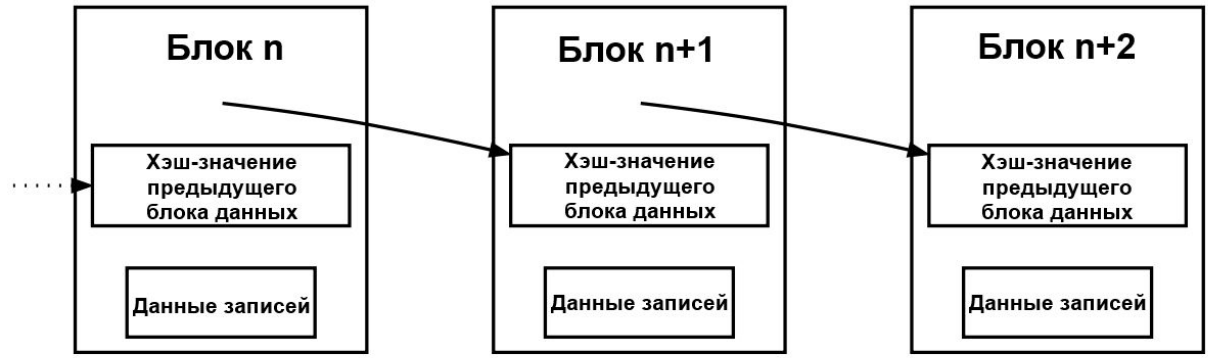

Рисунок 1.1 — Логическое представление базы данных на основе технологии blockchain.

1.2 Добавление данных в базу

Учитывая тот факт, что записи базы данных не могут быть изменены после достижения консенсуса, тем не менее, она может быть обновлена путём добавления новых записей. Новые записи могут иметь взаимозаменяющий характер. Например, если база данных используется для регистрации доменных имён, то вовсе не требуется удаление или модификация старой записи о каком-либо имени для внесения обновлений. В этом случае в базу данных добавляется обновлённая запись, а старая остаётся для поддержания целостности истории.

Процедура добавления данных в базу может быть описана несколькими основными шагами:

- формирование новой записи в соответствии с форматом базы данных;

- передача записи на рабочую станцию;

- включение записи в один из новых блоков данных;

- достижение консенсуса относительно нового блока данных.

На рисунке 1.2 отображено логическое представление распределённой базы данных, где точками (узлами сети) обозначены сами рабочие станции, а соединяющими линиями — каналы передачи данных. Каждый узел сети хранит полную копию цепочки блоков данных.

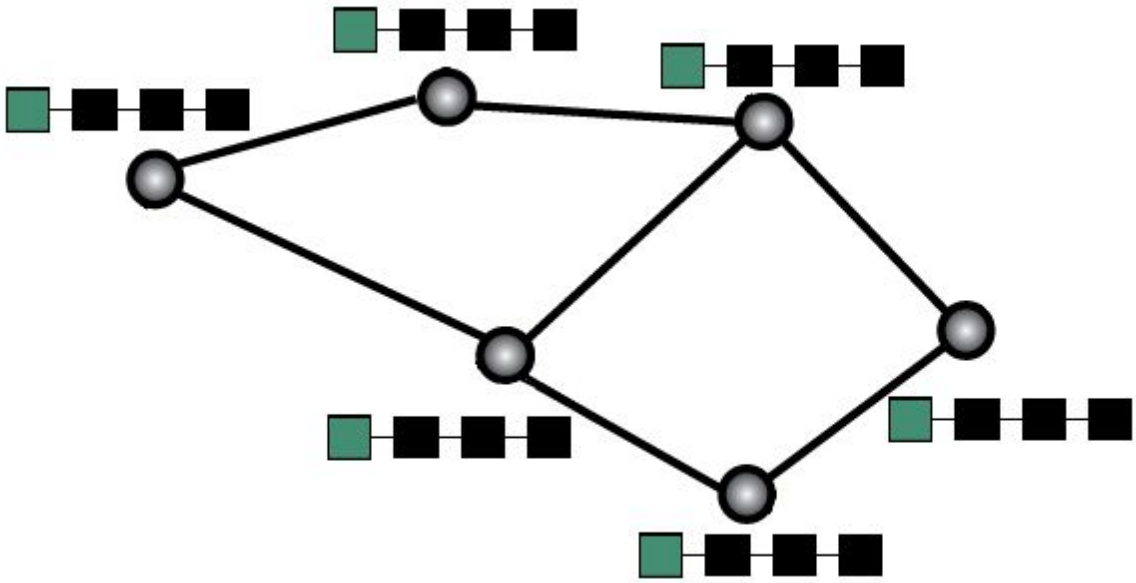

Рисунок 1.2 — Логическое представление распределённой базы данных.

1.3 Консенсус

Целью консенсуса является достижение единогласия относительно текущего состояния базы данных. Наиболее распространённый способ достижения консенсуса — это голосование. В общем случае каждая из рабочих станций голосует, предлагая новый блок данных. Таким образом, голос — это блок данных, который в последствии может быть утверждён. Поскольку каждый новый блок данных (голос) включает хэш-значение предыдущего блока, то

голосующая станция тем самым голосует за все предыдущие блоки данных. Блок данных, который набрал необходимое количество голосов (большинство), считается неизменяемым. Даже если существует альтернативное текущее состояние, то оно было выбрано меньшинством и по правилам протокола не является действительным. Как правило, меньшинство заинтересовано в изменении своих локальных копий в соответствии с состоянием базы данных, которое было выбрано большинством.

Принцип достижение консенсуса за счёт голосования не вызывает никаких вопросов, пока все рабочие станция остаются однозначно идентифицированы. В остальных же случаях рабочая станции не может доверять чужим голосам, т.к. полученный блок данных не авторизован (анонимен).

На рисунке 1.3 отображена цепочка блоков данных с альтернативной версией её конечного состояния. Следуя правилам протокола, верной версией состояния из всех альтернативных будет та, которая была построена с большим объёмом голосов.

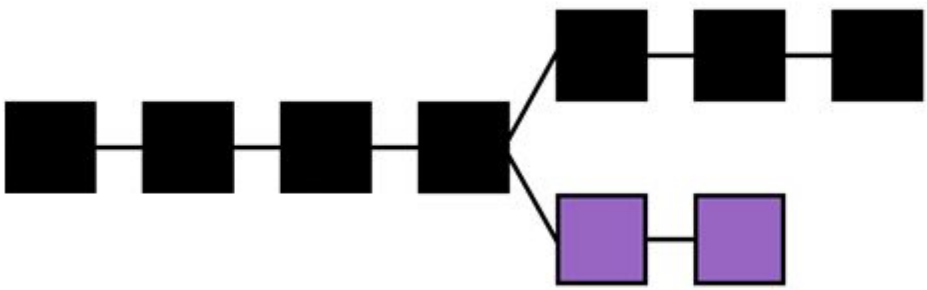

Рисунок 1.3 — Цепочка блоков с альтернативными версиями.

1.4 Голосование в распределённых анонимных сетях

В случае, когда распределённая сеть состоит из неизвестного числа не идентифицированных рабочих станций, которые друг другу не доверяют,

возникает вопрос, как организовать честное голосование. Этот вопрос решается предоставлением доказательства на право голоса. Согласно протоколу все рабочие станции знают правила генерации и проверки доказательства.

При формировании очередного голоса об обновлении базы данных рабочая станция прикрепляет доказательство на право голоса. Распространяясь по сети, новый блок данных может быть принят на рассмотрение другими рабочими станциями только при наличии доказательства. Важно отметить, что при передаче, блока данных по сети дополнительные средства обеспечения целостности и конфиденциальности не требуются. Это объясняется тем, что любое искажение блока данных нарушает доказательство права голоса.

1.5 Публично проверяемые доказательства

Для протоколов, в которых консенсус достигается с помощью голосования с доказательством права голоса, важным является алгоритм формирования и проверки таких доказательств. Ниже приведён список требований к доказательству на право голоса:

- возможность формирования рабочей станцией самостоятельно;
- задача по формированию уникальна для каждой рабочей станции;
- сложность формирования для всех станций одинакова;
- проверка может быть выполнена любой другой рабочей станцией;
- минимальная сложность проверки.

Таким образом, протоколы, реализующие консенсус относительно базы данных в распределённых сетях, могут использовать различные алгоритмы доказательства права голоса для обеспечения честного голосования. В общем случае алгоритмы доказательства делятся на несколько основных типов:

- proof-of-work;
- proof-of-stake;
- proof-of-identity;

- proof-of-importance.

Рассмотрим коротко каждый из типов.

Proof-of-work. Здесь право голоса эквивалентно объёму проделанной работы. Для доказательства права на голос нужно проделать некоторую работу, сформировать тому доказательство и приложить к своему мнению. Как правило, для этого типа доказательства используется вычислительная работа, например, решение некоторой математической задачи. Ярким примером протокола, в котором используется данный тип доказательства, является Bitcoin.

В протоколе Bitcoin в качестве работы выбрана математическая задача по поиску прообраза хэш-функции, для которого хэш-значение удовлетворяет некоторым условиям.

Proof-of-stake. Здесь право голоса конкретного голосующего пропорционально объёму его активов в базе данных. Для доказательства права на голос нужно предоставить доказательство владения активами, которые хранятся или учитываются в данной базе данных. Как правило, активы привязаны к паре ключей электронной цифровой подписи владельца, в таком случае доказательством права голоса будет ЭЦП, наложенная на предлагаемый блок данных. Примерами протоколов, использующих такой тип доказательства, являются Peercoin (самая первая реализация данного типа), NXT, Bitshares.

Proof-of-identity. Здесь право голоса конкретной рабочей станции зависит от того, доверяют ли ей остальные рабочие станции. Причём, это право будет разным для каждого из проверяющих, поскольку у каждого из них разный уровень доверия к голосующему. Доказательство представляет собой идентификацию голосующего, в большинстве случаев реализованную за счёт ЭЦП. В простом случае проверяющая сторона идентифицирует голосующую сторону по открытому ключу ЭЦП. Если идентификация пройдена вес голоса равен 1, в противном случае 0. Первым протоколом, использовавшим консенсус на основе такого голосования, является Ripple.

Proof-of-importance. Наиболее сложный тип доказательства из всех перечисленных. Как правило, здесь используется комбинированное доказательство, которое можно так же назвать доказательством важности участия голосующей стороны в процессе достижения консенсуса. Наиболее распространённый случай — это доказательство того, что голосующий является активным пользователем базы данных. Например, голосующий доказывает, что значительная часть записей в базе данных принадлежит ему. Примером протокола с данным типом доказательства является NEM.

Важно отметить, что перехват чужого доказательства не имеет смысла, т.к. одним из входных параметров при его формировании является сам голос. Как результат, доказательство верно только для конкретного голоса (блока данных).

1.6 Аутентификация новых записей

Существует несколько основных способов аутентификации:

- с предоставлением прообраза хэш-функции
- с использованием ЭЦП
- множественная ЭЦП

Рассмотрим данные способы на примере регистрации доменных имён. При каждом обновлении записи об одном и том же имени дополнительным полем указывается хэш-значение некоторых секретных данных. Таким образом, для аутентификации следующего обновления необходимо предоставить прообраз хэш-функции для значения, указанного в прошлом обновлении. В этом случае регистрация нового имени происходит без аутентификации.

В случае с использованием ЭЦП новое имя регистрируется с привязкой к открытому ключу подписи регистрирующей стороны. Все последующие обновления этого имени должны быть подписаны соответствующим личным ключом регистрирующей стороны.

Аутентификация с множественной ЭЦП требует наличия нескольких подписей. Зачастую используется для аутентификации записи с подтверждением несколькими лицами, а также с целью минимизации вероятности утраты контроля над записями. Например, для аутентификации записи необходимо предоставить любые две подписи, соответствующие трём зарегистрированным открытым ключам. Тогда пользователь может распределить хранение трёх личных ключей с возможностью утраты любого одного из них.

1.7 Частота появления блоков данных

Полагая, что цикл голосования — это некоторая хронологическая последовательность голосов, в которой каждая рабочая станция проголосовала один и только один раз, возникает вопрос, как организовать цикл голосования в распределённой анонимной среде. В силу того, что рабочие станции не имеют центра управления, к решению этого вопроса служит иной подход нежели определение чёткого порядка. Простейшее решение выглядит следующим образом: порядок голосования рабочих станций зависит от псевдослучайной величины, которая может быть рассчитана любой из рабочих станций. Причём для всех узлов, имеющих одно и тоже текущее состояние базы данных, эта величина будет одинаковой.

Рассмотрим два случая использования такого параметра: упрощённый и общий. В простом случае, когда все голосующие стороны имеют одинаковый вес голоса, число голосующих известно и все они идентифицированы (как минимум по открытому ключу ЭЦП), эта величина непосредственно определяет порядок голосования. В более общем случае голосования эта величина называется сложностью (difficulty). Сложность является одним из параметров алгоритма формирования и проверки доказательства на право голоса. Таким образом, доказательство права голоса должно удовлетворять общесистемному параметру сложности.

Как результат, инструментом регулирования частоты и порядка голосования в общем случае является сложность. Кроме сложности, алгоритм формирования доказательства может иметь дополнительные параметры, такие как время, объём активов голосующего в базе данных, некоторые данные о использовании базы данных голосующим. Таким образом, может существовать целое множество параметров, регулирующих возможность формирования голоса каждым конкретным голосующим.

Одним из важных требований, которым должен соответствовать протокол подобных систем, является непредсказуемость. Возможность детерминировать закон, определяющий порядок голосования за следующее состояние системы (базы данных), должна отсутствовать. Причём должна отсутствовать даже в вероятностном смысле. В противном случае кто угодно может предопределить момент времени, в который будет существовать достаточно высокая вероятность формирования нескольких голосов подряд. Что позволяет любому спланировать и провести атаку 51-го процента голосующей мощности (см. главу 2) с достаточно высокой вероятностью, реально не имея такой голосовательной способности.

1.8 Изменение сложности

Как правило, время формирования следующего блока строго не определено, поскольку возможность формирования доказательства на право голоса зависит от одной или нескольких псевдослучайных величин. Стало быть, на произвольных интервалах времени частота появления блоков данных может существенно разниться. Механизм регулирования сложности решает эту проблему на длинных интервалах времени. Наиболее распространённым вариантом управления сложностью является её приспосабливание с целью достижения константного времени формирования новых блоков данных на длинном интервале времени. Имея историю формирования блоков с фиксированными метками времени, можно оценить текущую частоту формирования голосов. В самом простом

варианте сложность может быть изменена пропорционально изменению частоты. Поскольку данные вычисления могут быть произведены каждой из рабочих станций, то в случае одинаковых состояний локальных копий баз данных рассчитанная сложность также будет одинакова. Нормальной ситуацией считается, когда распределение случайной величины об интервалах времени между сформированными блоками данных удовлетворяет нормальному закону распределения.

1.9 Механизмы защиты от DOS атак

В случае использования распределённой базы данных для массового обслуживания, возникает вопрос о защите от отказа в обслуживании. В рамках описанной технологии создания распределённой базы данных, пожалуй, единственным разумным способом атаки является переполнение. Поскольку доступ к публичной базе данных может получить кто угодно из любой точки сети, а сама база подразумевает только добавление записей, то чрезмерное количество новых записей приводит к классическому отказу обслуживания.

Существует два основных метода защиты от подобных ситуаций — это логический и экономический. Логический может быть описан правилами протокола, т.е. на уровне протокола существуют некоторые строгие ограничения, которые проверяет каждая рабочая станция независимо. Как правило, в эти ограничения входит максимальный размер блока данных, максимальный размер записи и т.п. Такие строгие ограничения, в общем случае, обойти невозможно, т.к. их соблюдение гарантированно консенсусом. Тем не менее, этого метода не достаточно, поскольку, атакующему ничего не мешает занять всю выделенную пропускную способность сети своими запросами.

Для экономического метода защиты всё несколько сложнее. Для реализации такого метода правила протокола должны предусматривать наличие внутреннего актива. В данном случае, под активом может пониматься репутация, акции и т.п.

Иными словами, база должна хранить данные о некотором балансе для каждого пользователя. Тогда пользователи имеют возможность оплачивать добавление новой записи в базу данных, а консенсус, в свою очередь, реализует механизм взимания платы и приоритетного добавления записей.

Идеологически, реализуется публичная распределённая база данных массового обслуживания с необходимостью оплаты каждой единицы записанных данных. Таким образом вопрос атаки отказа в обслуживании упирается ещё и в экономическую составляющую. Практически, необходимо использовать оба метода защиты.

1.10 Механизм вознаграждений

Рассматривая общий случай базы данных массового обслуживания, всех пользователей можно разделить на две группы. Первая группа — пользователи осуществляющие чтение и запись базы данных, вторая группа — пользователи поддерживающие рабочие станции распределённой сети, т.е. выполняющие синхронизацию и верификацию базы данных. Очевидно, что группы могут иметь пересечение, тем не менее пользователи первой группы заинтересованны в простоте и надёжности чтения записи данных, а пользователи второй группы должны быть заинтересованы в поддержании сети. Самым распространённым способом мотивации поддержания сети является механизм вознаграждений. Как правило участники входящие в состав второй группы получают вознаграждение за каждый сформированный блок данных. В таком случае вознаграждение представляет собой сумму плат всех включённых записей. Тогда рабочие станции стремятся формировать новые блоки данных, при этом включая как можно больше новых записей. Кроме того, рабочие станции заинтересованы верифицировать и добавлять записи в приоритете, основанном на плате за добавление в базу данных.

1.11 Временные метки

Не смотря на то, что формат данных распределённых баз часто предусматривает поля временных меток, они на самом деле там не нужны. Единственное для чего подсчёт времени может быть нужен, так это модификация параметра сложности в зависимости от частоты генерации блоков. Но даже в этом случае само время не важно, важно количество условных единиц времени, на протяжении которых блоки были сформированы. Тем не менее, для протокола важен единый синхронизированный механизм учёта времени. Говоря другими словами, временной триггер, которому все доверяют. Такая необходимость связана с тем, что все рабочие станции должны выполнять некоторые действия синхронно. Например, изменение сложности, переход на новые правила взаимодействия или консенсуса в рамках обязательных обновлений. Кроме того, триггер должен зависеть только от состояния базы данных, т.к. необходима возможность верификации всей цепочки блоков данных задним числом.

В качестве такого триггерного механизма лучше всего подходит нумерация блоков данных. Порядковый номер для Genesis Block равен нулю и для каждого следующего блока данных увеличивается на единицу. С одной стороны, это удовлетворяет требованиям ситуации, а с другой — это позволяет проводить верификацию всей базы данных без привязки к реальному времени. Таким образом, можно говорить, что в распределённой базе данных на основе blockchain течёт своё время, где единицей времени является один блок данных.

1.12 Эмиссия активов

Для распределённых баз данных массового обслуживания характерны два вида эмиссии внутреннего актива, а также их комбинация. Первый и преобладающий способ — это эмиссия как вознаграждение за поддержание базы данных, т.е. новые единицы активов выпускаются в каждом новом блоке данных и остаются в распоряжении его создателя. В совокупности с вознаграждениями за

верификацию новых записей рабочая станция за каждый сформированных голос получает количество активов, равное сумме эмитированных единиц и плат за добавление новых записей в базу данных. Это означает, что в первые минуты существования суммарное количество активов равно нулю, и возрастает с каждым следующим блоком данных. Второй способ предусматривает полную и одноразовую эмиссию всех активов. Как правило, создатель базы данных с таким способом эмиссии просто вносит запись в нулевой блок данных о том, что на его балансе есть некоторое количество активов. При этом сам протокол не предусматривает дополнительной эмиссии.

1.13 Сетевое взаимодействие

Вновь сконфигурированная и запущенная рабочая станция не имеет возможности сразу стать частью распределённой базы данных. На это есть как минимум две причины: рабочая станция не имеет данных о размещении других рабочих станций в сетевом пространстве, локальная копия базы данных пуста. Под пустой базой данных обычно понимается наличие только нулевого блока данных — Genesis Block, как начального состояния или хэш-значением это блока данных. Механизмы поиска других рабочих станций для базы данных с публичным доступом не реализуются и категорически не допустимы на уровне протокола. Причиной тому является то, что вновь запущенная рабочая станция не может доверять ни одному источнику данных о состоянии сети. Централизованные поисковые системы недопустимы в силу своей уязвимости и требований к доверию, а такие распределённые системы поиска как DHT поддаются простым атакам и даже в случае успеха не гарантируют подлинность результатов поиска. Стало быть, задача по организации сетевого взаимодействия вновь запущенной рабочей станции возлагается на её администратора. В самом распространённом случае администратор указывает несколько сетевых адресов других рабочих станций. После установление сетевого соединения вновь

стартовавшая рабочая станция выполняет начальную синхронизацию базы данных. На этом этапе указанные администратором сетевые узлы являются доверенными источниками данных, т.к. для баз данных характерна неоднозначность дальнейшего развития. В ходе начальной синхронизации рабочая станция может проводить полную верификацию всех записей и блоков данных по желанию. Также рабочая станция может расширять количество сетевых P2P соединений за счёт получения данных о расположении других рабочих станций в сети от текущих соединений. По завершению начальной синхронизации рабочая станция переходит в штатный режим работы. Причём, решение о таком переходе должно приниматься независимо, как правило, на основании того, что текущие соединения более не возвращают новые блоки данных по запросу. Вообще говоря, работа в режиме синхронизации — это частный случай работы в штатном режиме. Таким образом, рабочая станция постоянно стремиться получить наиболее свежие блоки данных (голоса других рабочих станций), при этом выбирая ту версию истории базы данных, за которую было отдано больше голосов. Такое поведение можно отнести к основным принципам достижения консенсуса.

1.14 Реорганизация

Переход рабочей станции между альтернативными версиями состояния базы данных называется реорганизацией (Reorganize). Причины появления альтернативных состояний можно разделить на две группы: умышленные и неумышленные. К умышленным причинам относятся намеренные отклонения от правил протокола, например, с целью навязывания одному или нескольким другим пользователям недействительного состояния базы данных. К неумышленным причинам возникновения альтернативных цепочек блоков данных относятся задержки сети на передачу данных между рабочими станциями и задержки на верификацию полученных блоков данных. Таким образом, все

рабочие станции не могут мгновенно получить данные о последнем состоянии сети. Существует необходимость перехода между альтернативными состояниями базы данных, кроме того, рабочая станция вынуждена самостоятельно принимать решение о том какому именно состоянию отдать предпочтение, т.к. внешним источникам информации доверие отсутствует. В общем случае, по правилам протокола, верной цепочкой блоков данных из всех альтернативных является та, у которой суммарный вес голосов больше. В результате, все рабочие станции, которые действуют согласно правилам, гарантированно будут иметь одинаковые локальные копии баз данных.

1.15 Выводы по главе

Распределённая база данных на основе blockchain — это множество копий одной и той же базы данных и выполнение правил достижения консенсуса по всем копиям. При этом, физически, данные базы — это набор аутентифицированных записей. Говоря простыми словами — это история всех записей (более правильно, запросов на запись) с самого начала её существования. На некотором уровне абстрагирования любая автоматизированная система представляет собой базу данных и механизм принятия решений по её обновлению. Стало быть, использование данного подхода делает возможным создание автоматизированной системы при отсутствии её физического представления. Такую систему можно рассматривать как третью сторону по предоставлению некоторых услуг, в то время как эта сторона существует, лишь как результат выполнения протокола всеми её клиентами.

2 ОПИСАНИЕ ПРОТОКОЛОВ

2.1 Bitcoin

Это — исторически первый протокол, реализующий синхронизацию распределённой базы данных на основе blockchain. Сеть состоит из произвольного числа неидентифицированных узлов. Частота формирования блоков данных 6 блоков в час. P2P соединения между узлами позволяют им обмениваться новыми записями и новыми блоками базы данных. Протокол реализует консенсус на основе голосования с доказательством проделанной работы (proof-of-work). В качестве работы для доказательства права голоса используется задача по нахождению прообраза хэш-функции SHA-256.

Блок данных имеет следующий формат:

- заголовок блока данных
- блок данных

Заголовок блока данных имеет следующий формат:

- версия протокола
- хэш предыдущего блока
- хэш записей (Merkle Root)
- метка времени
- сложность доказательства
- доказательство права голоса

Сложность задачи по поиску прообраза хэш-функции в рамках протокола Bitcoin задаётся таким образом, чтобы среднее время формирования одного блока данных было настолько велико, чтобы вероятность решения задачи несколькими рабочими станциями в один момент времени была незначительной. Кроме того, максимально допустимый размер блока данных жёстко ограничен логикой

протокола. Как результат, пропускная способность распределённой базы данных крайне мала, на практике составляет 1,7 KiB/s. Очевидно, что увеличение частоты формирования блоков данных приведёт к появлению большого числа конфликтующих блоков данных. Под конфликтующими блоками данных понимается два и более голоса, у которых хэш-значения предыдущего голоса совпадают. Сформированы такие блоки, как правило, разными рабочими станциями, приблизительно в одно время и имеют равные права на существование. Первопричиной возникновения конфликтующих блоков является латентность сети передачи данных. Вероятность возникновения прямо зависит от их размера. Получается, что при вероятностном (недетерминированном) подходе к определению порядка голосования и доказательстве права голоса по принципу proof-of-work имеем проблему. При увеличении частоты формирования блоков данных протокола Bitcoin снижается эффективность вычислительной мощности сети, направленной на достижение консенсуса, т.е. эффективность голосования падает. Важно отметить, что в других протоколах, использующих иной метод доказательства права голоса, эффективность голосования может не снижаться с появлением большого числа конфликтующих блоков. Это связано с тем, что процесс формирования голоса не требует больших вычислительных затрат.

Консенсус работает таким образом, что все рабочие станции верифицируют все запросы на запись в базу данных. Как минимум, рабочие станции должны следовать одному и тому же алгоритму проверки, который описан правилами протокола. Тогда результат верификации гарантированно будет один и тот же для каждого запроса на запись у всех рабочих станций, работающих в соответствии с протоколом. Вопрос заключается в следующем, как в таких условиях организовать верификацию запросов на запись, которые имеют набор непредсказуемых факторов: объём данных, формат данных, способ аутентификации. Простым решением была бы типизация всех возможный вариантов запросов на запись. Для каждого типа можно было бы описать

отдельный алгоритм верификации. Основная проблема такого решения — это ограниченность, как много бы ни создавалось типов записей, их число всегда будет конечно. Кроме того, добавление нового типа требует обновления ПО на каждой рабочей станции единовременно.

Протокол Bitcoin реализует другое решение этого вопроса. Bitcoin-script — это язык описывающий правила верификации записей. Он содержит данные и операции по их обработке для получения результата верификации. На более низком уровне этот язык описывает правила обработки стэка данных. Таким образом, можно составить запрос на запись произвольных данных в распределённую базу данных и описать собственные правила верификации этого запроса в функциональных рамках Bitcoin-script.

Существует естественная атака, направленная на нарушение работы консенсуса, как независимого механизма принятия решений. Суть атаки заключается в том, что атакующий может контролировать больше половины вычислительной мощности сети, направленной на решение задач по формированию доказательства на право голоса. Таким образом, атакующий получает абсолютное большинство голосов в системе принятия решений. Что в свою очередь даёт возможность формировать новые состояния базы данный не принимая во внимание мнения остальных. Конечно же, такое поведение будет сразу заметно всем пользователям, и решение о выборе главной цепочки развития базы данных будет зависеть от самих пользователей.

2.2 Ripple

Ripple — первый протокол реализующий синхронизацию распределённой базы данных с использованием доверительных отношений для достижения консенсуса. Сеть состоит из произвольного множества узлов, где каждый узел идентифицирован некоторым подмножеством других узлов. P2P соединения между узлами позволяют им обмениваться новыми записями и новыми блоками

базы данных. В отличии от Bitcoin, такие сетевые соединения имеют смысл только между рабочими станциями, которые друг другу доверяют. Доказательством на право голоса является подпись голосующего. Верифицировать такой голос может любая рабочая станция, но принять во внимание такой голос может только та рабочая станция, которая доверяет голосующему. Фактически голос имеет разный вес для каждого проверяющего в зависимости от уровня доверия к автору. Грубо говоря, консенсус устроен таким образом, что рабочая станция принимает следующий блок данных, если он подписан большинством известных избирателей. Такой блок данных может быть вытеснен только альтернативным блоком, который подписан большим количеством известных избирателей.

За счёт легковесности такой модели голосования в распределённой среде можно достичь высокой частоты формирования блоков данных по сравнению с Bitcoin. Кроме того, протокол Ripple реализует совершенно другой подход к структуре данных в blockchain. Кроме аутентифицированных записей протокол предусматривает физическое хранение актуального состояния по всем записям базы данных. Что, в свою очередь, сокращает время чтения актуального состояния записей из базы данных. Как результат, имеем значительное уменьшение времени верификации новых записей.

Модель консенсуса сети остаётся устойчивой, пока большая часть рабочих станций работает не в сговоре. Также для предполагаемого уровня стойкости, модель требует, для каждой рабочей станции, наличия защищённого сетевого соединения хотя бы с одной рабочей станцией, к которой данная имеет доверие. В противном случае, рабочая станция может быть неосведомлена о некотором состоянии базы данных. Недостатком такой модели является тот факт, что вновь запущенная рабочая станция не имеет возможности принять участие в голосовании (достижении консенсуса) вне зависимости от своих намерений, ресурсов и т.п. Для участия в принятии решений новой рабочей станции

необходимо получить доверие других рабочих станций, т.е. мнение данной рабочей станции имеет вес только для тех, кто ей доверяет.

Классическая атака 51-го процента, в рамках протокола Ripple, сводиться к похищению доверия. Атакующему сначала необходимо получить 51% доверия (абсолютное влияние) у 51% всех рабочих станций, а потом навязать своё мнение. Здесь под мнением понимается альтернативное состояние базы данных, где альтернативность не имеет никакого отношения к правильности.

2.3 Bitshares

Протокол Bitshares реализует модификацию proof-of-stake доказательства на право голоса для достижения консенсуса. Интересной особенностью протокола является его техника по управлению данными и их обработке (имеется в виду локальная копия базы данных). Каждая рабочая станция, фактически, ведёт две базы данных для всё той же единственной цели — консенсуса. Первая база данных — самая сложная с точки зрения технологий, использует технологию blockchain. Её преимущества с точки зрения структуры данных — это хронологический порядок записей, аудит целостности, возможность частичной синхронизации. Вторая база данных построена по банальным принципам хранения соответствий, чаще всего, такую структуру данных называют Ledger (учётная книга). Явные преимущества заключаются в размере базы и скорости доступа к данным, поскольку эта структура может хранить только некоторое конкретное состояние. Оптимизация протокола заключается в одновременном использовании преимуществ двух баз данных разных по струкуре. По мере работы рабочей станции ведутся обе базы данных, грубо говоря, одна для истории, а другая для скорости доступа.

Верификация организована таким образом, что все необходимые данные для проверки записей или блоков данных находятся в оперативной памяти в

соответствии со структурой Ledger. В то время как blockchain в своей основной массе хранится на диске.

Разработчики Bitshares пошли на оправданную хитрость для оптимизации размера записей. Было решено не добавлять в записи любые данные (типа открытых ключей и хэш-значений), которые когда либо были добавлены другими записями, а вместо этого вписывать их идентификаторы в Ledger.

Консенсус на основе модифицированного proof-of-stake работает по принципу разделения голосующих и верифицирующих рабочих станций. Каждый пользователь по желанию может выставить свою кандидатуру на пост верифицирующей рабочей станции. Потом среди всех пользователей проводиться голосование за кандидатов, где вес каждого голоса определяется суммой активов голосующего. По результатам голосования выбирается N (натуральное число) кандидатов, которые получают право формировать новые блоки данных. Таким образом, те пользователи базы данных, которые имеют право голоса в достижении консенсуса по принципу proof-of-stake, не являются при этом верифицирующими узлами. Реально же формировать блоки записей могут другие рабочие станции, который были выбраны в результате голосования.

Правила протокола гарантируют честное принятие решений, если большая часть активов, принимающих участие в голосовании контролируется честными пользователями.

Согласно правилам протокола фактическую верификацию записей и формирование блоков данных выполняет ограниченное множество рабочих станций. Это фактически означает, что механизм принятия решений на некотором интервале времени имеет конкретное физическое представление в виде N рабочих станций. Стало быть, одновременное нарушение работы этих рабочих станций влечёт временный отказ в обслуживании всей базы данных, а именно невозможность выполнять запись данных. Такого рода отказ будет продолжаться до тех пор, пока пользователи не осуществят голосование в пользу выбора других

верифицирующих рабочих станций. Механизм голосования работает асинхронным образом, а цепочка блоков может быть перестроена на альтернативную версию. Это даёт возможность самостоятельного восстановления работы распределённой базы данных после успешно проведённой DOS атаки.

2.4 Ethereum

Ethereum — это распределённая платформа смарт-контрактов. Смарт-контракт — это такая запись, которая требует выполнения некоторой дополнительной логики, кроме верефикации, перед добавлением в базу данных. Логически Ethereum — это тот же Bitcoin, но вместо bitcoin-script имеется полноценный язык описания логики. Идеологически эта платформа позволяет написать логику отношений между несколькими пользователями в виде цифрового контракта, а позже выполнить именно эту логику в независимой среде. Преимущество таких цифровых контрактов заключается в том, что они одновременно являются инструментом описания логики и инструментом её выполнения.

Solidity — это язык описания логики контракта, используемый в Ethereum. Это объектно ориентированный язык программирования со встроенной функциональностью по чтению всей базы данных и управлению данными самого контракта.

EVM (Ethereum Virtual Machine) — это среда, в которой выполняется Solidity. Поскольку Solidity на много более функциональный язык, чем bitcoin-script, для его выполнения требуется специальная среда (виртуальная машина). Создание контракта выполняется в несколько этапов: написание программного кода, трансляция кода в инструкции, запись инструкций в базу данных. EVM может обращаться ко всей цепочке блоков (только на чтение) и изменять своё состояние, где состояние виртуальной машины для каждого контракта своё. Кроме того, EVM может создавать новые контракты с

использованием инструкций (программного кода), которые кем-либо были добавлены в базу данных. Также код контракта во время выполнения в EVM может вызывать другие контракты. В этом отношении платформа напоминает реестр программных библиотек.

На момент написания этой работы протокол Ethereum реализует консенсус с использованием только proof-of-work. Важной особенностью Ethereum в использовании proof-of-work для доказательства права голоса является то, что среднее время формирования нового блока данных находится в диапазоне от 10-ти до 20-ти секунд. Очевидно, что такой короткий интервал времени между голосами рабочих станций в классическом proof-of-work недопустим. Короткий интервал делает схему достижения консенсуса неэффективной из-за проблемы появления альтернативных голосов (состояний базы данных). Кроме того, каждый альтернативный голос означает напрасно затраченные вычислительные ресурсы. В сети Ethereum эта проблема решена очень просто. Каждый следующий голос цепочки может опираться не на один предыдущий голос, а на несколько. Таким образом, вычислительная мощность, затраченная на формирование альтернативных голосов не останется потраченной напрасно. При этом, вес всей цепочки голосов рассчитывается как сумма вычислительных мощностей, затраченных на её формирование вместе с альтернативными голосами.

Для выполнения некоторого контракта, а точнее, вызова конкретного его метода, пользователю необходимо сформировать новую запись базы данных, где, указать вызов конкретного метода из конкретного контракта, передать параметры необходимые для выполнения. Такая запись распространяется по всем рабочим станциям и проходит верификацию каждой из них. Во время верификации записи, если запись требует вызова некоторого контракта, по правилам протокола рабочая станция должна загрузить на EVM состояние соответствующего контракта, а затем начать выполнение вызываемого метода с переданными параметрами. Результатом выполнения контракта на виртуальной машине будет

изменение её состояния относительно выполняемого контракта или других контрактов в случае их вызова в ходе выполнения данного. Кроме того, в ходе выполнения контракт может распоряжаться своими активами, отправлять их пользователю или другому контракту.

Основные принципы голосования при достижения консенсуса протоколом Ethereum в целом повторяют принципы протокола Bitcoin. Таким образом, основные моменты касательно атаки 51-го процента вычислительной мощности в сети Bitcoin справедливы и для сети Ethereum.

2.5 Выводы по главе

Протокол Bitcoin представляет собой инструмент для создания абстрактного учётного реестра. Свойствами такого реестра являются целостность и независимость. При этом низкоуровневая структура данных протокола позволяет реализовать дополнительные сервисы на уровне приложений без изменений в самом протоколе: например, сервисы, позволяющие увеличить уровень приватности пользователей, динамические способы аутентификации записей или даже другие распределённые реестры на основе этого.

Протокол Ripple является инструментом для создания распределённого реестра пользовательских активов. Консенсус данного инструмента решает проблему единогласия рабочих станций в более упрощённой среде функционирования, по сравнению с консенсусом в Bitcoin. Упрощение заключается в том, что каждая рабочая станция доверяет некоторому подмножеству других рабочих станций. Такая упрощённая среда функционирования позволяет достигать консенсуса между всеми рабочими станциями гораздо быстрее. На практике время формирования нового состояния базы данных может достигать одной секунды, что на данный момент является отличным показателем.

Протокол Bitshares позволяет создать распределённую систему с расширенным функционалом протокола Ripple, но при этом с принципиально отличной моделью консенсуса. С одной стороны, участие в консенсусе принимают все пользователи распределённого реестра пропорционально объёму своих активов, с другой стороны, фактическое принятие решений выполняется подмножеством избранных рабочих станций. Кроме того, данный протокол реализует более гибкую схему управления пользовательскими записями и опциональную возможность повышения приватности, по сравнению с другими рассмотренными протоколами.

Протокол Ethereum решает проблемы организации распределённой базы данных, предварительно разделив их на две основные составляющие: запись и обработка записей. Логически Ethereum — это протокол распределённой платформы смарт-контрактов, где взаимодействие пользователя с платформой сводится к размещению контракта и инициацией его выполнения. При этом выполнение логики контракта фактически является выполнением пользовательской программы на виртуальной машине каждой рабочей станции. Теоретически на платформе Ethereum можно описать логику любого из рассмотренных протоколов. Реально же выполнение пользовательских программных модулей в такого рода распределённой среде является весьма затратным.

ВЫВОДЫ

Распределённые базы данных находят широкое применение в ситуациях с повышенными требованиями к доступности и целостности базы данных, а также в системах учёта с открытым доступом.

В первой главе были рассмотрены теоретические основы технологии blockchain. Без привязки к конкретным реализациям были рассмотрены архитектурные и логические принципы построения распределённых баз данных, способы достижения консенсуса относительно их состояния, принципы взаимодействия между пользователями базы данных и способы аутентификации записей. Во второй главе были описаны основные детали реализации распределённых баз данных с использованием протоколов Bitcoin, Ripple, Bitshares, Ethereum. Третья глава посвящена сравнительному анализу рассмотренных протоколов по низкоуровневой структуре данных, способу достижения единогласия относительно состояния базы данных, по пропускной способности и по устойчивости модели достижения консенсуса к атакам. Четвёртая глава посвящена результатам практического анализа распределённой базы данных.

Ключевым элементом распределённой базы данных на основе blockchain является консенсус. Принципы достижения консенсуса, заложенные в протоколе, определяют многие свойства базы данных, такие как сфера функционирования, пропускная способность, время отклика, устойчивость к атакам, уровень приватности пользователей.

Структура данных рассматриваемых протоколов логически одинакова, но физически реализована с различиями. Это даёт возможность подобрать протокол, который будет наиболее производительным при решении конкретной задачи.

Как правило, протоколы реализующие распределённые базы данных на основе blockchain вынуждены решать проблемы оптимизации объёма цепочки блоков данных и их синхронизации, объёмов записей и их аутентификации. В этом смысле Bitcoin оптимизирован для повышения приватности пользователей и предоставления более гибкого инструмента для высокоуровневых приложений. Ripple оптимизирован для быстрого отклика и взаимоучёта пользовательских активов. Bitshares оптимизирован для быстрого отклика, взаимоучёта пользовательских активов и обработки большого числа запросов. Ethereum оптимизирован для реализации пользовательской логики в независимой среде.

Практический анализ протоколов распределённых баз данных показывает явную зависимость пропускной способности от следующих факторов. Время отклика распределённой базы данных прямо зависит от объёма базы данных в случаях, если протокол не реализует оптимизации доступа к данным структуры blockchain. Эффективность консенсуса имеет обратную зависимость от задержек в каналах передачи данных и прямую зависимость от пропускной способности каналов передачи данных, при этом заметно сказывается географическое расположение рабочих станций. В отдельных случаях, например, в случае достижения консенсуса рабочими станциями с использованием proof-of-work, эффективность достижения консенсуса в группе рабочих станций расположенных близко друг другу значительно возрастает. Причём такая группа рабочих станций может фактически иметь абсолютное влияние на механизм принятия решений, при этом не являясь большинством избирателей, но имея преимущество за счёт низких задержек синхронизации.

Недостатками распределённых баз данных на основе технологии blockchain являются низкая пропускная способность, высокие затраты на поддержание сети и рабочих станций.