**Практическое руководство реализации Peer2Peer – сети с использованием блокчейн узла**

Гоша Давид Александрович

[Практическое применение блокчейн технологии в автоматизированих платежных системах 1](#_Toc1)

[Революционные концепции блокчейна 2](#_Toc2)

[Общие теоретические сведения о блокчейне 2](#_Toc3)

[1. Баланс 2](#_Toc4)

[2. Транзакция и UTXO модель 4](#_Toc5)

[4. Асимитричное шифрование 5](#_Toc6)

[5. Proof-Of-Work 6](#_Toc7)

[6. Майнинг 7](#_Toc8)

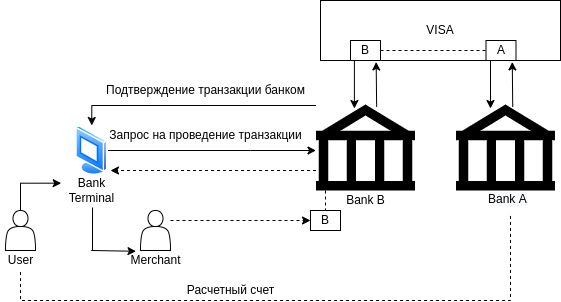
[Формирование блоков 9](#_Toc9)

[Топология сети и gossip протокол 15](#_Toc10)

###### **Практическое применение блокчейн технологии в автоматизированих платежных системах**

Понимания громоздкости и устарелости текущей самой популярной системы совершения денежных транзакций в мире очень важно, и преимущество блокчейна однозначно . Такой системой являются банковские платежи, а эмитентами услуг – Visa, Mastercard, Maestro. Для начала нужно понять почему транзакция в сети ethereum происходит быстрее банковских. Для этого нужно изучить цепочку посредников и предназначения каждого звена в экономике банковского сектора. Представим неких абстрактных покупателя и мерчанта, для восстановления всей цепочки транзакции. И проведем анализ начиная от прикладывания банковской карты к терминалу, до подтверждения транзакции. И так, ни одно юридическое лицо не может функционировать без счета в банке. ИП - может. Представим что наш мерчант это некий ООО. Для функционирования он открыл счет в банке B. Последний в свою очередь выдал платежный терминал. Далее наш абстрактный покупатель совершает покупку в ООО, после прикладывания карты к терминалу совершается несколько программных действий, в свою очередь мы рассмотрим только формирование транзакции. Примитив вида транзакции следующий , некий пользователь банка А хочет совершить перевод в размере 100 условных единиц валюты в банк продавца (Б), несколько реквизитов и МСС код. После чего терминал направляет пакет данных в банк В. В свою очередь, если транзакция интернациональная, должен обратиться к международной платежной системе, к примеру - VISA1. Непосредственно МПС Visa должна передать запрос банку клиента что бы получить ответ о наличии нужной суммы и статуса покупателя. Если все удовлетворяет условия совершения транзакции для обоих сторон, банк А отправляет ответ , которой движется по той же цепочки. Банк А – МПС – Банк Б – терминал. И к сожалению после всего объема проделанной работы , деньги мерчант получит только через три дня. Далее приведем диаграмму спецификации эквайринга. Где Банк Б – эквайер, Банк А – эмитент карты.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



*1Мы не будем уточнять, что оба банка должны заключить договор с VISA для предоставления услуг клиентам.*

!!!Дабы не затягивать первый из пунктов статьи мы не будем разбирать как происходят межрегиональные переводы или как их еще называют межбанковские , внутри банковские транзакции, нетинг, БЭСП и тд. Но я уверяю вас каждый из этих методов требует большое количество посредников , которые нагружают всю банковскую систему, бюрократическую проволочку и большое количество сотрудников банков, для поддержания инфраструктуры. Масштабируемость этого процесса безгранична и исчерпывается только суммой вложенных средств в поддержание и создания мощных серверов на каждом из сочленении посредников, тем самым получая достаточно не плохую производительность.

###### **Общие теоретические сведения о блокчейне**

###### Баланс

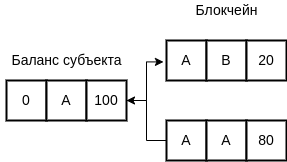
Уточним что топология нашей сети – одноранговая децентрализованная или же пиринговая, так как рассматриваемый пример это биткоин. Следовательно эта архитектура сети подразумевает под собой отсутствие центральной точки управления. Эта технология порождает ряд неоднозначных моментов, таких как двойная трата (double spend). Такая проблема была решена при помощи цепочки цифровых подписей. Для понимания сказано нам нужно абстрагироваться от привычной нам модели обмена фидуциарными деньгами. В биткоине отсутствует баланс или физические монеты как таковые. Существует один и только один леджер, который является цепочкой всех транзакций, на основе которого мы можем математическим путем установить истинный баланс узла. То есть сказанное ранее можно представить как цепочку передач прав на владение какой то часть общей эмиссии валюты, а сами коины находятся в состоянии мономорфизма и представлены только непосредственно для простоты человеческого восприятия. Нигде не фигурирует конечное поле балансов пользователей, это значения полиморфно.  
При создании следующей транзакции, некий сторонний узел должен проверить леджер, цепочку транзакций, связанных хешами предыдущих транзакций. Для того, что бы удостовериться что количество входов количества выходов. Фактически проверка, хватает ли у отправителя средств. Так, как у одного отправителя может быть ряд входящих транзакций и ряд исходящих, выведем следующую формулу, которая обозначит валидность суммы перевода:  
 ,

где i = сумма всех входящих переводов, o = суммой исходящих перерводов  
  
!!!!Существует проблема с двойной тратой. Звучит она так: если пользователь А передал нам, пользователю Б, транзакцию с суммой 100 уе, как мы можем быть уверены что такую же транзакцию не получил любой другой участник сети. В некоторых блокчейнах, по типу корды, данная проблема решается при помощи центральной, контролирующей ноды, но мы попробуем избежать насколько не надежной реализации. Представим что центральный контрагент скомпрометирован и может как совершать так и не совершать проверку, теперь по условию у нас появляется точка доверия, что не желательно в гибридных сетях. Решается эта проблема следующим образом, если у нас отсутствует некая центральная точка, то каждый узел должен являться валидатором и собирать все транзакции, таким образом зная все суммы перевода сети. А именно посредством выбора самой длинной цепочки блоков в сети, с которой соглашается большая часть ее участников. Понятно что сейчас это чудовищная нагрузка на сеть, и 99% участников не смогут принимать количество транзакций сопоставимо с Visa или MasterCard По этому, с целью оптимизации работы сети, было установлено следующее правило. Каждая нода устанавливает текущий блокчейн блоками, собирая некое количество транзакций и хэшируя результат выполнения. Этот процесс называется энкеринг. Это самое базовое преимущество использования блокчейн-технологии. В своей работе, Сатоши предлагает вносить данные в некий сторонний сервис, который будет подтверждать каждый хеш - датой, на которой произошло подтверждение транзакции и внесение ее в блокчейн. Фактически после такого алгоритма действий мы можем восстановить всю цепочку и доказать время и состояние баланса контрагента в определенном промежутке времени.

Реализовывать блокчейн можно разными способами, даже если сфера применения заранее известна. Так например, если разрабатываемая программа на основе блокчейна является платёжной системой, то получение баланса пользователя уже можно реализовать двумя способами: детерминированным и недетерминированным1, не говоря уже о том, из чего будет состоять блок, каковы ограничения блока, какие награды за майнинг и тд. Такое состояние дел несёт более негативный характер, так как безопасность итогового продукта не будет определяться общепринятыми стандартами, которые прошли открытый и долговременный анализ. Чтобы бороться с этим фактором принято придерживаться стандартов де-факто, подобия биткоина «Bitcoin» (который выступает в качестве классической платёжной системы) и эфириума «Ethereum» (в качестве платформы контрактов).

###### Транзакция и UTXO модель

Механизм создания транзакций без дополнительной нагрузки на подтверждение владением средств. Это значит, что при создании перевода мы можем распоряжаться только, полной частью полученных монет, вспоминаем о работе баланса. Так как физического понятия баланса не существует, это лишь функция, рекурсивно восстанавливающая по цепочке транзакций количество обязательств перед объектом. Следовательно потратить баланс можно только полностью. Частичная трата приведет к проблеме делении ветви и не соответствию в цепочке блокчейна. Механизм перевода монет следующий, субъект отправителя долен передать нужную сумму объекту получателя, если же сумму меньше баланса, субъект должен вернуть остаток себе.



Если же реализован недетерменированый протокол с хранением состояний в блоке, UTXO модель не потребуется. Так как изменения состояний леждера фиксируется в каждом блоке.

Решение проблемы двойной траты. Так как в сети отсутствует посредник или валидатор, которому по умолчанию доверяют все ноды без исключений, появляется проблема связная с поделкой транзакций. Отправитель А может отправить пользователю Б транзакцию эквивалентно своему балансу, затем отправить такую сумму пользователю В. Сеть должна располагать неким алгоритмом и рядом свойств для предотвращения таких типов атак. Для этого у каждой транзакции указывается ссылка на предыдущую, то есть последнюю общепринятую на данный момент времени и собственный уникальный идентификационный номер, от которого в будущем будут отталкиваться новые транзакции. В биткоине такой Id генерируется при помощи хеша самой транзакции и оказии, таким образом формируя ориентированный список и защищая транзакцию от изменений в будущем.

После чего происходит процесс поступления транзакции в mempol, место, где она уже попала в сеть, но еще не была записана не в один из блоков, следовательно, не была продекларирована фактически. В таком состоянии, транзакция находиться до добавления ее в блок. Блок это 1 мегабайт транзакций. Блоки нужны для оптимизации работы сети, ведь подтверждать тысячу транзакций едино разово менее ресурсозатратно и более рационально. Формирование блока начинаеться с создания леджера. Леджер это криптографически доказуемое упорядоченые транзакций для предотвращения уязвимости двойной траты. После формирование леждера, блок хешируеться по алгоритму дерева меркла.

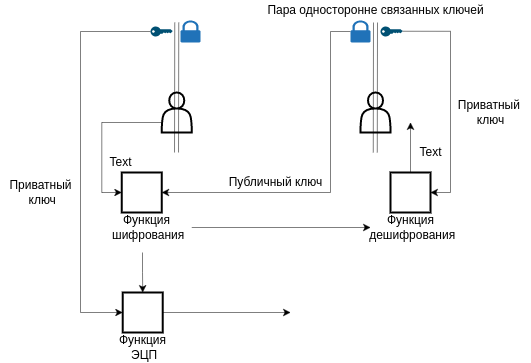
1. Конфиденциальность

Если в ранее рассмотренных пунктах механизм работы хоть и отличался от консервативных, сути и целей результата это не меняло. Суть конфиденциальности совершенно перевернута с ног на голову, в отличии от банковских платежей. Если в банковской программе есть некая точка доверия, то-есть сам банк, который скрывает нашу конфиденциальную информацию от других участников, хоть и полностью владеет ей. Принцип сокрытия в современных протоколах криптовалют совершенно иной. Вся информация о суммах и счетах, их транзакциями публично доступна каждом участнику сети. Мало того, они обязан хранить ее у себя. Но вот связать номер счета с его владельцем достаточно сложно2. Это влечет ряд неких проблем, которые нарушают эту концепцию, и ряд решений такие как миксера и создания нового кошелька для каждой транзакции. Хорошо что создание кошелька ограничивается генерацией пары ключей.

###### Асимитричное шифрование

Для понимания реализации работы сети в условно враждебной среде нам нужно ознакомится с основами асимитричного шифрования, мы не будем углубляться в такие математические аспекты о генерации случайных байтов для пары ключей или самим алгоритмам, которые реализуют односторонние функции3 с доказательством сложно обратимости. Мы лишь рассмотрим протокол, на котором базируется логика шифрования и создания цифровых подписей.

Спецификация взаимодействия двух субъектов в заведомо враждебной среде, на примере общедоступной сети биткоина. Как мы видим для шифрования данных нам понадобится два ключа, которые связаны односторонне. Есть отдельный раздел математики – модальная арифметика, которая занимается доказательством односторонней обратимости обоих ключей. Что это значит. Что имея приватный ключ мы с полиномиальной сложностью получим публичный ключ, а вот для получения приватного ключа зная публичный нам понадобиться 2n, где n - это половина длины ключа в байтовом эквиваленте. Если учитывать что алгоритм дешифратора имеет пренебрежительно малый шанс предугадывания по сравнению с генерацией всех значений, на получения ключа уйдут десятилетия используя все актуальные мощности мира. Сообщение, которое было зашифровано публичным ключом получателя, может прочитать только владелец приватного ключа, т.е получатель. Безусловно есть ряд атак направленных на анализ шифротекста с целью взлома, без получения значения ключа. Но как правило стандартизированные алгоритмы шифрования уже хорошо протестированы и максимально уязвимы только на первых итерация шифрования, по этому до сегодня самым эффективным методом дешифрации остается терморектальный метод.   
Похожим образом функционирует ЭЦП – электронная цифровая подпись. В сети она нужна для защиты от MITM4 атак. Работает схоже с шифрованием, но в метод подается приватный ключ отправителя, после чего шифруются хешированные данные о транзакции, любой пользователь, у кого есть публичный ключ отправителя может расшифровать информацию о транзакции и убедиться , что только владелец данного приватного ключ, то есть отправитель, мог подписать транзакцию. Публичный ключ это общедоступная информация, и по этому расшифровать ее может любой желающий участник сети. Именно так работает симметричное шифрование. Как уже упоминалось выше, это нужно для подтверждение отправителя транзакции, что бы исключить посредников перехвате сообщений или отправок от чужого лица.



###### Proof-Of-Work

В этой главе разберем реализацию консенсуса блокчейн-системы биткоина. Начнем с того, что whihe paper bitcion by Satosi Nakamoto совершается отсылка к работе Адама Бэка Hash-Cash , именно в ней впервые описан алгоритм подтверждение работы, в качестве универсальной технологии защиты от спама. Например что бы перед подтверждением блока, узлу пришлось выполнить определенную вычислительную работу, предположительно ему было бы не выгодно атаковать сеть засоряя тысячами транзакций в секунду. На самом деле, технология может использоваться и в других системах для форда спама. Но в нашем случае консенус будет отыгрывать совершенно другую роль. А именно равномерное распределение эмиссии монет и как протокол выбора лидера. Участники будут пытаться найти нужный хеш и получить право добавить блок в блокчейн.

Для хеширования используется протокол sha256. Который принимает некую строку всех транзакций в блоке и возвращает хеш, состоящий из 256 бинарных данных. Значения входят в диапазон массива от 256 нулей до 256 единиц. Если же мы установим правило, по которому сеть, будет принимать хеш, в котором первые 10 значений – бинарные нули, то мы существенно увеличим сложность подбора такого хеша из-за уменьшения диапазона подходящих хешей. Следовательно ограничив диапазон в 2 раза, примерное время хеширования увеличится в двое. Мы можем уменьшать диапазон до тех пор, пока не получим нужную для нас скорость добычи хеша, за среднее время5 . Но вспомним главное свойство хеширования. Если подавать одинаковые блоки информации, будем получать одинаковые хеши. По этому, для получения разностей и выполннения работы, а не генерации одного и того же хеша всей сети, было введено понятие nonce6. Теперь, подавая каждый раз оказию в функцию хеширования, мы будем получать разные хеши для одного блока, и перебирать их, пока один из узлов не получит вхождение в указанный диапазон хеша. Так как мощность сети или количество вычислителей, это динамическое свойство, зависящее от количества участников самой сети, был предусмотрен алгоритм динамического изменения сложности сети. К примеру, сегодня в сети 10 участников,   
вычислительная мощность который равняется некому значению n. То есть с этой мощностью подтверждение блока происходит в среднем раз в 10 минут. Если к сети подключиться еще 10 человек и мощность возрастет в 2n раз, блоки начнут подтверждаться в 2 раза быстрее, что в свою очередь хоть и ускорить процесс подтверждения, нарушит концепцию эмиссии монет в сети и приведет к удвоеной нагрузке. Для предотвращения этого каждые 1440 блоков или ~1 месяц реального времени происходит смена актуальной сложности. По этому, как правило в начале месяца блоки подтверждаются раз в 15 минут затем ускоряются до 7.

Форк (Fork) – Разветвление цепочки блоков. Существует вероятность одновременого нахождения хеша для двух разных блоков с разным значением оказии. В такой ситуации происходит хардфорк и сеть разделяется на 2 части, каждая из которых продолжает какую-то часть своей ветки. Следовательно разрешается проблема очень легко, кто следующий выигрывает в добыче блока, того ветвь признается главной из-за правила - самая длинная цепочка побеждает. Тут работает принцип электронного голосования, чем больше мощности майнит (голосует) за добычу следующего блока, тем больше вероятность его нахождения. Узлы которые проиграли, обязаны принять новую ветку. По этому важно ожидать больше 1 подтверждения от сети для конфирма перевода.

###### Майнинг

Майнинг очень тесно связан с консенсусом proof-of-work. Так как в сети нету центральной точки отказа, с которой по умолчанию соглашаются все узлы, возникает проблема под названием «Византийское соглашение». Проблема о принятии решения в заведомо враждебной среде. Это классическая криптографическая задача, которая имеет несколько достаточно спорных решений, но в биткоине она решается именно случайным выбором узла, блок которого признает действительным вся сеть. Участники сети, в частности майнеры. играют в криптографическую лотерею. Они проверяют блок, на действительность всех транзакций, сравнивая его хеш с хешем любого другого узла, если же в блоке будет какое-то незначительное изменение, хеш будет разительно отличаться и майнить будет попросту бессмысленно, так как такой блок, даже если удастся найти удовлетворяющий сеть хеш, принят не будет. После проверки блока, все учасники пытаются, при помощи перебора nonce, найти подходящий хеш. Как только такой хеш будет найден, узел получит поощрение от хранилища и право на установке и распространении нового блока, так как угадывание хеша достаточно ресурсозатратное вычисление, мы может предположить что майнеру будет не выгодно обманывать сеть, как как даже если попытаться изменить какое-либо значение в блоке, другие узлы это заметят и не станут принимать его блок.Награду за добытый блок он получает от хранилища.

Хранилище или coinbase, которое является неким блокчейн пользователем, но не имеющим приватного ключа. Это можно рассматривать как исключение из правил, которое формирует новое правило только для него в контексте майнеров. Суть хранилища сводится к тому, чтобы оно выплачивает из своей «казны» деньги за выполненную работу майнерам. При этом, хранилище может быть как самовосстанавливающимся, так и невосстанавливающимся. Если мы говорим о невосстанавливающихся хранилищах:

1. Майнер замайнил блок

2. Хранилище платит деньги майнеру

3. Майнеру перечисляется процент с транзакций   
  
Если мы говорим о самовосстанавливающихся хранилищах:

1. Майнер замайнил блок

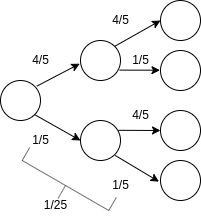
2. В хранилище перечисляется процент с транзакций

3. Хранилище платит деньги майнеру

Если разным майнерам посылается одновременно транзакция, то у майнеров возникает конкуренция или гонка по майнингу. Шансов выиграть больше у того, кто обладает большими мощностями, но вопрос в другом, как координируются майнеры?

Ведь в теории майнеры могут майнить одни и те же значения, тем самым расходуя понапрасну время и энергию. Для такого случая, создаются пул-сервера, к которым подключается группа майнеров заинтересованных в решении одной задачи и распределении ресурсов между собой. Пул-сервера координируют действие каждого отдельного майнера, таким образом, чтобы они не пересекались в переборе одинаковых значений.

Понятие эгоистического майнинга. Теоретически возможная уязвимость в процессе майнинга, которая может позволить майнеру получать вознаграждение чаще, не меняя размер хешрейта сети. Эта уязвимость в логике работы никак не решена алгоритмически. Можно утверждать ,что если существенно большая доля мощности сети сконцентрирована у selfish майнера, то в перспективе это может привести к централизации сети. Процесс следующий, если эгоистический узел добывает блок, то он не спешит публиковать его в сеть и делиться ним с остальными узлами. Создаться искусственный форк, который майнится узлом дальше , в надежде обогнать других, честный майнеров. Чем больше блоков у узла будет в приоритете, тем большие финансовые потери получать честные майнеры. Как только злоумышленник видит, что цепочка блоков честных майнеров начинает догонять его, он публикует свою, которую он добывал в тайне от сети. После чего, сеть по умолчанию принимает его продолжение блокченйна, как единственное действительное и отбрасывает все блоки, которые добывали честные майнеры, тем самым нанося им финансовый ущерб.



Пример попытки совершения эгоистического майнинга с процентной долей мощности злоумышленика - 20%. Из этого суждения получим формулу биномиального случайного блуждания qn  где,

q – Вероятность опередить добычу блока в сети.

n – Количество блоков, которые требуется опередить.

Так как честные майнеры или пулы участников, все реже получают вознаграждения, они переходят на сторону эгоистического майнера, нарушая главную концепцию биткоина – децентрализацию. По истечению какого-то времени, мощность , которую может сконцентрировать эгоистический пул может превысить 50%. Теперь, если наша майнинговая мощность ,больше общей мощности сети хотя бы на 1 хеш, существует 100% вероятность опережения сети при теоретически бесконечных временных затратах, если же наша мощность разительно больше, существует вероятность контролировать поведение всей сеть. А именно: игнорировать транзакцию, не принимать блоки или же создавать double spend.

Атака 51%, к которой подвержены все блокчейны с консенсусом доказательства работы. Это атака на блокчейн, где один субъект или организация захватывает наибольшую часть хешрейта, что может привести к нарушениям работы сети. В такой ситуации у злоумышленника будет достаточно мощности майнинга, чтобы намеренно исключать транзакции или изменять их порядок, а также отменять собственные транзакции, провоцируя проблему двойного расходования.

КОМИСИИ

1Детерминированность можно рассматривать как математическую функцию, результат которой зависит только от принятых аргументов и не от чего более. Её также называют концепцией «без состояний», отличной аналогией могут служить функциональные языки программирования (Haskell, LISP и тд.). Так например, чтобы получить баланс пользователя, необходимо будет перебрать всё множество блоков, которое относится к принятию / отправлению монет. Тем самым, создастся вся последовательность операций человека и полная закономерность его действий от начала и до конца.

Недетерминированность можно рассматривать как концепцию «с состояниями», и аналогией для этого могут служить императивные языки программирования (C, Go и тд.). Если необходимо перенести это свойство на блокчейн, то можно сделать так, чтобы состояние баланса пользователя сохранялось в отдельном блоке, а не в цепочке блоков. Таким образом, чтение баланса пользователя будет происходить с конца цепочки и прекратится в тот момент, когда найдётся нужный блок.

2Сложность сопоставления номера кошелька и его владельца ограничиваеться сервисами доверительного хранения данных о счете таких как: биржи, программные кошельки. Или же непосредственно OSINT деанонимизации субъекта.

3Односторонняя функция — математическая функция, которая легко вычисляется для любого входного значения, но трудно найти аргумент по заданному значению функции.  
4MITM Man in the middle — вид атаки в криптографии и компьютерной безопасности, когда злоумышленник тайно ретранслирует и при необходимости изменяет связь между двумя сторонами, которые считают, что они непосредственно общаются друг с другом.

5Среднее время — Существует вероятность, которая зависит от текущей сложности сети или диапазона удовлетворительного хеша, наивного угадывания блока, конечно она ничтожно мала. О смене сложности сети мы поговорим в дальнейшем.

6Существует еще одно — и, пожалуй, наилучшее — решение проблемы выбора вектора инициализации. Оно состоит из двух шагов. Вначале каждому сообщению, которое должно быть зашифровано с помощью заданного ключа, присваивается уникальное число, называемое оказией (nonce). Название этого термина происходит от фразы “number used once” — “число, используемое только один раз”. Основным свойством оказии является ее уникальность. Не допускается использовать одну и ту же оказию дважды с одним и тем же ключом. Обычно оказия представляет собой номер сообщения, присвоенный ему по некоторому принципу и, возможно, скомбинированный с какой нибудь другой информацией. Большинство современных систем уже применяют нумерацию сообщений, чтобы сохранять сообщения в правильном порядке, распознавать повторяющиеся сообщения и т.п. Оказию не обязательно сохранять в секрете, но она может быть использована только один раз. Это особенно важно для получателя сообщений, который получает оказию вместе с каждым сообщением и должен убедиться в том, что ее текущее значение не повторяется дважды.

###### Формирование блоков

Стоит начать рассмотрение блока с самой минимальной его версии, где он способен оставаться и функционировать как объект в блокчейне. Так для блока достаточно всего одного поля - хеша предыдущего блока. Эта урезанная версия проста для понимания, но и даёт при этом начало теоретической базы. Так например, представим, что у нас существует некий блок B0, который представляет начало цепочки, по терминологии он является генезис-блоком. Суть его в том, что он не имеет предыдущего блока и является единственным прародителем всех последующих. Можно представить блок B0 в следующем виде:

B0 = <случайная строка>

Случайная строка в генезис-блоке заменяет хеш блока и она в теории может быть любой, даже не случайной, так как это не влияет на безопасность. Все последующие блоки будут генерироваться исходя из следующей формулы:

Bi = hash(Bi-1),

где hash - криптографическая хеш-функция\* [2, с.84][2, с.595][3, с.104]

Таким образом, получается цепочка хешей, которую лишь с оговорками можно назвать блоком. Чтобы придать блоку «смысла», нужно поместить в блок данные (D) для хранения, но таким образом, чтобы эти данные зафиксировались в результате хеша. Теперь блок усложнился и вместо одного значения (хеша), он стал хранить два (данные и хеш), тем самым став массивом:

B0 = [<NULL>, <случайная строка>]

Bi = [D, hash(D || Bi-1{2})]

В данном примере были введены запись «||», что означает конкатенацию строк и запись «Bi-1{2}», что означает взять из массива Bi-1 второй элемент (индексация для массива будет начинаться с единицы).

Эта запись уже более похожа на блокчейн, потому что хранит хоть какие-то значения, предполагаемо полезные пользователю. Но может возникнуть логичный вопрос: для чего это всё было сделано? И ответ - для безопасности (хоть на данном этапе ещё есть масса проблем). Суть в том, что теперь невозможно изменить значение Bi{1} (или Bi-1{2}) не затрагивая Bi{2}, иначе изменение будет обнаружено, потому что значение блока не будет соответствовать его хешу.

Ситуация для злоумышленника будет плачевной в тот момент, когда ему необходимо будет изменить значение блока в середине или начале цепочки, так как ему придётся изменять все последующие хеши, которые идут после изменяемого блока. Но проблемы в этой схеме всё же остаются.

Что если в блокчейне хранятся переводы денежных средств? Будет очень обидно, если злоумышленнику удастся изменить ваше значение, хранимое в блоке, на значение куда больших размеров, либо сгенерирует новый блок, в котором укажет вас как отправителя.

Чтобы решить данный недостаток, необходимо ознакомиться с таким термином как цифровая подпись\*\* .

Цифровая подпись помогает идентифицировать создателя блока, тем самым исключая моменты, когда кто-либо может выдать себя за другого человека. Сама концепция цифровой подписи несёт ещё два термина - адрес (публичный ключ) и кошелёк (приватный ключ).

Подпись должна применяться на хеш блока, а не на его значение по двум основаниям. Во-первых, хеш всегда имеет фиксированную длину, соответственно и скорость подписания не будет варьироваться. Во-вторых, подпись будет затрагивать также информацию о хеше предыдущего блока, что полностью свяжет подпись с предыдущими блоками.

Блок усложнился до трёх элементов (значение, хеш, подпись):

B0 = [<NULL>, <случайная строка>,<NULL> ]

Bi = [D, hash(D || Bi-1{2}), sign(Bi{2})]

Теперь, если злоумышленник захочет изменить значение Bi{1}, ему необходимо будет изменить и значение Bi{2}, но если он изменит значение Bi{2} он не сможет никак изменить значение Bi{3} так, чтобы этот блок продолжал идентифицироваться как до изменения. Или иными словами, посредник будет способен изменить блок, но только при условии того, что он поставит свою подпись на блок и точно также изменит все последующие блоки. Таким образом, злоумышленнику прийдется изменить весь блокчейн, что конечно же является достаточно сложным процессом, но возможным.

Даже если ваш блок будет находиться в начале цепочки, а цепочка в миллион блоков, современному компьютеру будет под силу изменить значение в цепочке и далее сгенерировать все последующие хеши с подписями злоумышленника, тем самым стерев все настоящие блоки.Так как процесс генерации хеша полиномиальный и занимает пренебрежительно малое константное время. Генерация всех блоков, существующих на данный момент в блокчейне биткоина, займет не более 10ти минут на современном процессоре.

В данном случае, на помощь приходит proof-of-work\*\*\* . Это ещё большее усложнение блока, которое добавляет такой термин как сложность вычисления. Сложность эта основана на математически трудной задаче, подобия нахождения нужного хеша. Обычный компьютер может тратить на такие задачи по несколько часов, дней, а то и месяцев и лет, не говоря уже о возможности человека найти собственноручно решение. Чтобы регулировать сложность, была придумана задача вычисления не всего хеша, а лишь его части. Допустим, перебирается число и оно хэшируется, нужно найти такое хэшируемое значение, чтобы оно начиналось на четыре нулевых байта. Размер хеша, а точнее количество нулевых байтов в начале называеться таргетом. Если такой хеш находится, значит задача решена. Но здесь определённо существует минус, если хоть один человек найдёт хешируемое число, которое заведомо будет начинаться с большого количества нулей, то придётся изменить способ вычисления трудной задачи, например перейти на единицы, потом на двойки, тройки, четвёрки и тд. Это явно неудобно, соответственно был придуман усовершенствованный способ этой задачи, что если хешировать не просто инкрементируемое число, а хешировать конкатенацию инкрементируемого числа с текущим хешем блока? Будет получаться всегда случайный результат и нельзя будет использовать полученное инкрементируемое число повторно на другом блоке. Поиск нужного хеша ещё называется майнингом.

Теперь блок будет выглядеть следующим образом:

B0 = [<NULL>, <случайная строка>,<NULL> ,<NULL> ]

Bi = [D, hash(D || Bi-1{2}), sign(Bi{2}), pow(Bi{2})],

где pow возвращает оказию x = Bi{4}, при которой hash(x || Bi{2}) <= TARGET.

Следует отметить, что функция pow, предназначена не только для подтверждения работы, что бы замедлить генерацию блоков и растянуть эмиссию монет в сети. Основой целью доказательства работы является решением двух математических задач. Первая это протокол выбора лидера, где лидер, а именно майнер будет выбран как достойный внести некую информацию в сеть. Вторая это распределенное голосование, где вес голоса определяется ЭВМ участника. В данном случае результат работы pow, число, которое доказывает что

1. Майнер использовал правильный таргет
2. Текущую сложность добычи блока

Так же ряд неоднозначных вопросов возникает при объеденение участников в пулы и сопоставление майнера, с найденным хешем. То есть, блок не содержит никакой информации о том, кто майнил. Представим такую ситуацию, что вы сами не майните блок, а перекладываете эту задачу на группу других людей, мощность компьютеров которых в разы превышает вашу ЭВМ и за это вы платите некий процент от перечисленных средств. Далее один человек из этой группы наконец-то находит nonce и заносит его в блок. Но в этой группе есть злоумышленник, он копирует nonce и пересылает его вам. Теперь вам поступило два ответа о найденном хеше, кому доверять? Можно сказать, кто первый принёс, тот и получит награду, но можно поступить другим, более правильным способом. Пусть сам майнер будет создавать блок, а вы ему будете отправлять лишь транзакцию (а именно три первых поля блока), далее майнер будет вносить свой хеш и подпись в блок, и на свой хеш будет производить операцию pow.

Блок ещё больше усложняется:

B0 = [<NULL>,<NULL> , <случайная строка>,<NULL> ,<NULL> ,<NULL> ]

Bi = [[D, hash(D), sign(Bi{1}{2})], M, hash(M || Bi{1}{2}|| Bi-1{3}), signM(Bi{3}), pow(Bi{3})]

Когда майнер найдёт оказию, число x = pow(Bi{3}), никто более не сможет скопировать это число и подставить под свой результат, так как полученное значение не будет подходить под хеш другого блока, где будет указан другой майнер. Но и при этом, необходимая транзакция попадает в блокчейн, со значением, хешем значения и подписью хеша. То-есть, всё как нужно. Но и на данном этапе существует лазейка. Можно заменить без последствий sign(Bi{1}{2}) и никто этого не заметит. Вопрос конечно может стоят иначе, ради каких целей существующую подпись стороннего человека захочется изменять на свою подпись, при этом подстраиваясь под его хеш? Но увы всякое бывает и потому это также следует считать уязвимостью. В исправлении данной уязвимости может помочь раскрытие внутренностей волшебного значения D.

И опять блок начинает усложняться:

B0 = [<NULL>, <NULL>, <случайная строка>,<NULL> ,<NULL> , <NULL>]

Bi = [[[S, R, V], hash(Bi{1}{1}), signS(Bi{1}{2})], M, hash(M || Bi{1}{2} || Bi1{3}), signM(Bi{3}), pow(Bi{3})]

В данном случае, S - sender (отправитель), R - receiver (получатель), V - value (само значение или количество условных отправляемых едениц). Также изменился вид sign(Bi{1}{2}) на signS(Bi{1}{2}), указывая на то, чья это подпись. Теперь подпись нельзя заменить, не затрагивая хеша. Изменить создателя подписи тоже нельзя, потому что необходимо будет менять хеш-значение, что неминуемо ведёт опять к замене подписи. Подпись майнера также проблематично изменить, потому что его имя указано в хеше. Доказательство работы украсть также не получится, потому что оно привязано к хешу. Кажется будто вот она, идеальная защита! Но увы нет, ещё остались уязвимости.

Помните, что в блокчейне нельзя доверять никому? Собственно это справедливо и для майнеров. Но каким образом майнер способен навредить? Ведь была сделана подпись на хеш, а хеш указывает на отправителя, получателя и значение, то-есть ничего из этого нельзя изменить, а если и изменишь то всё полностью, что аналогично простому игнорированию добавления транзакции в блок. Это всё верно, но есть одна деталь, которая очень надоедливая - транзакция ни к чему не прикреплена. Соответственно, майнер или кто-либо другой сможет скопировать полностью транзакцию и продублировать её заново, при этом она также будет валидна, потому что все хеши и подписи правильные (ведь они и вправду настоящие). Чтобы избавиться от этой уязвимости, можно просто указать в транзакции хеш предыдущего блока.

Блок ещё немного усложнился:

B0 = [<NULL>,<NULL> , <случайная строка>,<NULL> ,<NULL> ,<NULL> ]

Bi = [[[S, R, V], hash(Bi{1}{1} || Bi-1{3}), signS(Bi{1}{2})], M, hash(M || Bi{1}{2} || Bi-1{3}), signM(Bi{3}), pow(Bi{3})]

Теперь даже если майнер захочет продублировать транзакцию, она будет находиться под новым блоком, а хеш предыдущего блока уже не будет совпадать с хешем, который находится в транзакции. А если майнер попытается оставить изменённые значения, даже несмотря на несоответствие, то другие майнеры будут просто отвергать его блок как невалидный.

И вот теперь у нас блоки выстраиваются в цепочку, все они защищены от подмены, изменения, дублирования и при чём от всех пользователей. Но что если, существует несколько майнеров и у каждого из майнеров разная цепочка блоков? Как они будут согласовывать общий блокчейн? Pow – это некий вид голосования с помощью ЭВМ. Представим что какие то два или более майнера нашли подходящий блок одновременно или почти одновременно,и распостранили оба блока в сеть. В этот момент блокчейн распараллеливается на несколько веток, и превращатеься в дерево. Таким образом UTXO перестает работать и становиться не совсем понятно как в таком treechain валидировать транзакции. Проблема решается с помощью майнинга. Теперь группа узлов которая приняла блок первого майнера и группа, принявшая блок второго – соревнуются. Кто первый найдет блок, имеет право заменить, и откатить цепочку оппонирующей группы узлов. Следовательно чем больше ЭВМ группы , тем больше шансов что она первая найдет хеш. По этому если большинство сети продолжило майнить какую-то из цепочек, они проголосовали за эту ветку блокчейна. Такие противоречащие ситуации называются soft fork, когда блокчейн разделяется на несколько ветвей, но при этом ещё существует возможность выбрать только одну «правильную» цепь.

И помимо soft fork’ов существуют ещё так называемые hard fork’и. Hard fork’и, в отличие от soft fork’ов, сами по себе не появляются. Они используются тогда, когда необходимым является изменение или улучшение приложения, или же создание нового блокчейна на основе уже имеющегося. Это приводит к тому, что новая ветка никак не будет «срастаться» со старой, тем самым становясь несовместимой.

Так же каждый блок должен совершать эмиссию монет в сеть. Таким образом, зная всё это, необходимо изменить генезис-блок:

B0 = [[[S, M, V], <NULL>, <NULL>], <NULL>, hash(B0{1}{1}), <NULL>, <NULL>,<NULL> ],

где S - хранилище, M - майнер генезис-блока, V - награда за генезис-блок. Как только генезис-блок сгенерировался, начинает действовать хранилище, у которого определённый и ограниченный запас ресурсов.

Кажется, словно уже обо всём сказано, но существует ещё ряд тонкостей в блокчейне.

Во-первых, если разным майнерам посылается одновременно транзакция, то у майнеров возникает конкуренция или гонка по майнингу. Шансов выиграть больше у того, кто обладает большими мощностями, но вопрос в другом, как координируются майнеры? Ведь в теории майнеры могут майнить одни и те же значения, тем самым расходуя понапрасну время и энергию. Для такого случая, создаются пул-сервера, к которым подключается группа майнеров заинтересованных в решении одной задачи и распределении ресурсов между собой. Пул-сервера координируют действие каждого отдельного майнера, таким образом, чтобы они не пересекались в переборе одинаковых значений.

Во-вторых, транзакций в блоке может быть несколько (и так чаще всего бывает). В таком случае возникает ещё одна уязвимость, а именно дублирование транзакции в одном блоке. Чтобы избежать этого, нужно вставлять в каждую транзакцию случайное число (r) и хешировать его вместе со всей другой информацией.

В-третьих, можно применить временные метки (Time), которые будут свидетельствовать о времени создания блока, для его последующего отслеживания в истории и определения текущей мощности блокчейна со стороны майнеров\*\*\*\*\*. Всё это возможно лишь при условии грамотной синхронизации с сервером, представляющим время.

В итоге, последняя версия блока будет выглядеть следующим образом:

B0 = [[[[, S, M, V],<NULL> ,<NULL> ]],<NULL> ,<NULL> , hash(B0{1}{1}{1}),<NULL> ,<NULL> ,<NULL> ]

Bi = [[[[r, S, R, V], hash(Bi{1}{1}{1} || Bi-1{4}), signS(Bi{1}{1}{2})], [...], ... ], M, Time, hash(M || Time || (Bi{1}{1}{2} || Bi{1}{...}{2}) || Bi-1{4}), signM(Bi{4}), pow(Bi{4})]

\*Криптографическая хеш-функция - это односторонняя функция, которая принимает строку произвольной длины и возвращает строку фиксированной длины. Свойство односторонности говорит, что легко получить y от x [y = f(x)], но сложно получить x от y [x = f-1(y)]. В качестве криптографических хешфункций рекомендуется использовать хеш-функции семейства SHA-2 (sha224, sha256, sha384, sha512) или SHA-3 (keccak). В [3, с.111] указано, что хешфукнции «в сыром виде» имеют недостатки, подобия удлинения сообщения и коллизий. От второго избавиться проблематично, по причине существования атаки дней рождения, тем самым принято выбирать хеш-функции большего порядка. От первого же недостатка избавиться можно, как пример, применяя алгоритм HMAC на функцию. Но в блокчейне такая атака будет бесполезной, так как она требует редактирования уже имеющихся данных и хеша данных, что влечёт за собой необходимость редактирования подписи, по причине дальнейшего несоответствия между хешем и подписью.

\*\*Цифровая подпись - элемент асимметричной криптографии. В таких алгоритмах шифрования как RSA является результатом функции 14 расшифрования, а проверка подписи осуществляется при помощи функции шифрования. Замысел в том, что функция расшифрования известна лишь создателю закрытого ключа, соответственно только он способен при помощи этого свойства подтверждать свои действия.

\*\*\*PoW (Proof-of-work) - доказательство работы, осуществляемое при помощи решения сложной математической задачи (например, нахождение нужного хеша). Помимо PoW существуют и другие способы доказательства, как например PoS (Proof-of-stake), PoA (Proof-of-authority), PoB (Proof-of-burn) и тд.

\*\*\*\*Стоит сказать, что если майнеры соберутся в одну коалицию и их суммарная мощность будет больше 50% мощности всех майнеров, то возможна атака, при которой будет возникать достаточно продолжительный soft fork. Тоесть, накапливание блоков в разных цепочках будет происходить одновременно, тем самым разделяя общий блокчейн на два лагеря. Из этого явления могут выступать спекулятивные возможности отмены транзакций за счёт выбора майнерами нужной цепочки.

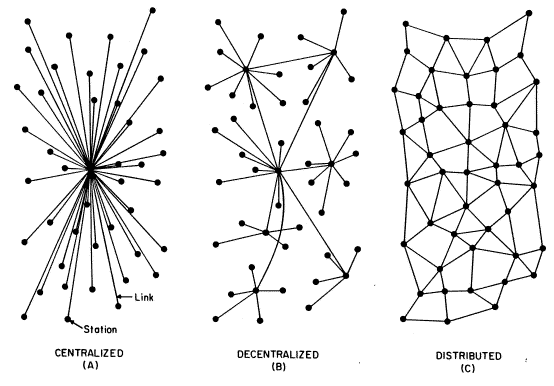
\*\*\*\*\*Вычислить текущую мощность майнеров можно при помощи меток времени. Если раньше на майнинг уходило в среднем 10 минут, а сейчас 5, то можно утверждать, что мощность майнинга увеличилась вдвое. При этом, тенденция к понижению необходимого времени сказывается неблагоприятным образом, так как будут возникать более частые случаи появления soft fork’ов. Из-за этого в блокчейнах предустанавливают функцию, которая регулирует текущую сложность блокчейна, подстраиваясь под константное время (например 10 минут).

###### **Топология сети и gossip протокол**

Чем больше вы изучаете сеть блокчейна, тем меньше она вам кажется полностью децентрализованной. В этой сети существуют сервера указывающие точное время, пул-сервера отвечающие за координацию действий узлов, а помимо всего прочего и сами узлы представляют из себя распределённый сервер, так как пользователи, которые осуществляют транзакции, являются клиентами.

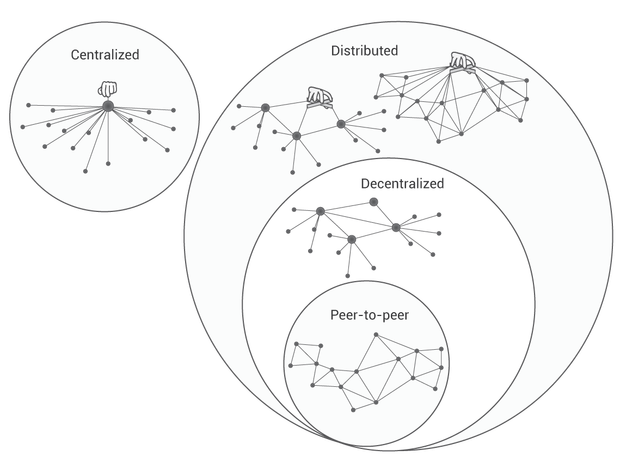
Сутью данного раздела является выявление особенностей блокчейна со стороны сети и построение связей между её участниками. В часности биткоин это оверлейная пиринговая сеть. Или не совсем так. Давайте разберем, оверлейная сеть это общий случай логической сети, создаваемой поверх другой сети. Узлы оверлейной сети могут быть связаны либо физическим соединением, либо логическим, для которого в основной сети существуют один или несколько соответствующих маршрутов из физических соединений. А пиринговая - оверлейная компьютерная сеть, основанная на равноправии участников. Часто в такой сети отсутствуют выделенные серверы, а каждый узел (peer) как является клиентом, так и выполняет функции сервера. В отличие от архитектуры клиент-сервера, такая организация позволяет сохранять работоспособность сети при любом количестве и любом сочетании доступных узлов. Участниками сети являются все узлы.

Так для начала стоит сказать, что существует два основных вида сети - многоранговые и одноранговые. Многоранговые (клиент-серверные) сети предполагают существование двух объектов - клиента и сервера, где клиент способен производить запрос, а сервер выдавать ответ. Одноранговые же (peer-to-peer) сети предполагают равноправие участников в действиях, или иными словами, пользователь в данной сети является одновременно и клиентом, и сервером (если исходить из терминологии многоранговых сетей). Таких пользователей именуют узлами. Так одноранговые сети есть подмножеством децентрализованных сетей, которые в свою очередь входят в множество распределенных. Распределенных сети включают в себя как одноранговые с топологией все ко всем, так и децентрализованные. Остальные типы распреленных сетей, нас интерсовать не будут.



Но помимо двух вышеперечисленных типов сетей существует некий гибрид, который так и называется - гибридная сеть. Её основная суть заключена в том, что существуют, как и в многоранговой сети, два основных объекта - клиент и сервер. Но особенность такой сети заключается в том, что в качестве сервера выступает одноранговая сеть, или иными словами сеть узлов. На картинке выше сеть А – многоранговая, В – гибридная, С – одноранговая.

И это определение гибридной сети всё же может вызывать вопросы. Например, что если, какой-нибудь человек поднимет сайт, хранение данных которого будет распределено между его серверами, тогда такая сеть будет называться гибридной? И можно ли будет назвать сеть гибридной, если существует только один узел?



Не ответив на эти вопросы, вряд-ли можно продолжать рассуждение о данном теме. Соответственно, необходимо найти в многоранговых и гибридных сетях принципиальное различие, и оно существует. Узлы в гибридных сетях, в отличие от серверов в многоранговых, не доверяют друг-другу, но и при этом выполняют общую работу. Из этого следует, что узлы не подчиняются какому-то конкретному человеку, либо узкому кругу лиц и тем самым несут децентрализованный характер. Но и из этого следует то правило, что если существует только один узел или узлы находятся в руках одной группы лиц, тогда такая сеть теряет свойства гибридной и переходит в фазу многоранговой.

В итоге, существуют следующие связи среди участников блокчейна:

1. Клиент -> [многоранговая] -> Узел
2. Узел -> [одноранговая] -> Узел
3. Узел -> [многоранговая] -> Пул-сервер
4. Узел -> [многоранговая] -> Сервер времени
5. Клиенты посылают узлам запросы на счёт получения баланса, блоков или занесения транзакции в блок.
6. Узел связывается с другими узлами для хранения общей цепочки. Узел может посылать другому узлу запросы о добавлении нового блока в блокчейн.
7. Узел запрашивает у пул-сервера необходимый диапазон майнинга.
8. Узел запрашивает у сервера времени текущее состояние времени.

Хоть количество многоранговых связей и преобладает, тем не менее, важность действий приходится на одноранговую связь. Основной вопрос здесь скорее лежит в том, ухудшается ли отказоустойчивость блокчейн сети, в моменты появления многоранговой связи? Возьмём для начала пример пул-сервера и представим злоумышленника пытающегося его отключить от сети методом DDoS атак. Сервер отключается, к нему доступ прекращён для некоторых узлов. В итоге, эта группа узлов может пойти по двум сценариям, либо подключиться к другому пул-серверу, либо генерировать блоки по отдельности, исходя из случайного числа. В данном 17 случае, сеть продолжает функционировать при любом обстоятельстве, хоть и с возможными оговорками по поводу падения производительности. Теперь же возьмём сервер времени. Отличие данного сервера от пулсервера заключается в том, что его можно использовать лишь единожды перед запуском узла. Иными словами, запросить у сервера точное время, установить это время на локальной машине и запустить узел. Таким образом, атаки на сервер будут приносить вред лишь появлению новых узлов, в то время как уже функционирующие узлы будут продолжать работу. При этом стоит учитывать, что серверов времени всегда несколько и произвести успешную DDoS атаку представляется сложной задачей. И даже если все серверы времени были отключены от сети, в это время продолжает функционировать сам блокчейн, в узлах которых хранится текущее время. Соответственно, формирующийся узел может запросить время не напрямую, у серверов времени, а косвенно, через уже работающий узел в блокчейне. Если подытожить всё вышесказанное, то отказоустойчивость блокчейна будет зависеть полностью от одноранговой архитектуры, при этом отключение от серверов может привести лишь к падению производительности некоторой группы майнеров, но никак не к понижению уровня отказоустойчивости.

Хорошо, если блокчейн сеть является по своей сути децентрализованной, тогда возникает логичный вопрос: каким образом клиенты (и сами узлы) будут подключаться к другим узлам? От куда они возьмут первоначальный список необходимых адресов? Это на самом деле очень частый вопрос в контексте децентрализованных сетей, так как каждая такая сеть решает подобные проблемы разными способами. Чаще всего, решением является использование сторонних каналов связи. Допустим, создаётся некий сервер, на котором будет располагаться список действующих узлов. Выведение из строя такого сервера не разрушит саму сеть, но перекроет доступ к информации о ней и её увеличению (что является проблемой лишь при начальном формировании блокчейна). Расширение списка будет происходить самими же узлами (то-есть они будут вносить свой адрес на сервер), так как заинтересованы в прибыли от майнинга. И чем чаще, на различных серверах адрес майнера будет попадаться, 18 тем чаще к нему будут обращаться клиенты, за помощью в майнинге (и сами майнеры, для добавления новых блоков).

Если в блокчейне учитывать добавление адресов, то количество связей в сети увеличится на две строки:

1. Клиент -> [многоранговая] -> Сервер адресов
2. Узел -> [многоранговая] -> Сервер адресов

Также стоит учитывать, что сам майнер способен выдавать список всех других майнеров, к которым он подключен, тем самым в большинстве случаев даже не нужно посещать сервера. Но может появиться один вопрос: из-за конкуренции майнеров в нахождении нужного хеша, не будет ли майнер пытаться выдавать урезанный список адресов, чтобы увеличить свои шансы на майнинг? Может и будет, но суть здесь в другом, если он не успеет замайнить блок и блокчейн обновится за счёт другого майнера, то блок, который майнил «майнер-мошенник» окажется невалидным. В итоге, ему придётся либо создать свою цепь блоков, игнорируя при этом весь другой блокчейн (что достаточно рискованно, так как это будет аналогично hard fork’у, с последующим привлечением майнеров на свой блокчейн), либо согласится с другими майнерами и принять тот факт, что транзакции от клиентов, которые посылались ему, оказались просроченными. Из этого случая клиенты поймут, что если транзакции не попали в новый блок, то скорее всего майнер выдал не весь список действующих узлов. Таким образом, клиентам лучше подключаться сразу к нескольким узлам и получать от них адреса других майнеров, отсеивая повторяющиеся. И есть ещё третий способ, благодаря которому можно получить список узлов. Авторы программы блокчейн-клиента (либо блокчейн-узла) могут поумолчанию внести в программу (либо в конфигурационный файл) список доверенных узлов, которые показали себя как действующие на протяжении долгого времени. И по мере обновления программы будет также обновляться этот список.

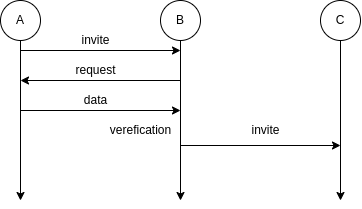
Со стороны сохранения мощности и доступности блокчейн-сети следует применять сразу все три способа нахождения адресов. Так например, если популярность данного блокчейна увеличилась на несколько порядков, то в теории первый способ нахождения адресов (при помощи сервера) может не использоваться, но тогда будет возникать уязвимость, когда доверенные узлы (третьего способа нахождения майнеров) скооперируются и не будут выдавать адреса всех других узлов, тем самым понижая мощность блокчейн-сети. Это скажется на более успешном пополнении баланса «майнеров-мошенников», при этом другие майнеры даже не получат шанса формировать новый блок, исходя из транзакций. При этом, если убрать третий способ, то будет всегда возникать необходимость обращения к серверу на инициирующем этапе запуска приложения. Если убрать первый и второй способы, то шансы занесения транзакции в новый блок будут пониженными (при условии того, что существует несколько клиентов с разным списком адресов). Если убрать второй способ, то возможность получать список новых адресов будет возложена только на сервер. Ну а если убрать первый и третий способы, то такая блокчейн-сеть будет терять лёгкость использования, из-за необходимости ручной настройки соединения.

Обмен данными в биткоине происходит по средствам широковещания, а именно при помощи Gosiip-протокола. Это группа протоколов в одноранговой компьютерной коммуникации, в которых распространение информации идёт способом, схожим с образом распространения эпидемий, и сводящимся к тому, что каждый или некоторые из узлов могут передавать обновляемые данные известным этому узлу соседям. Так как в сети биткоина , нам не известно точное количество узлов, это наверное лучший вариант коммуникации узлов. Существует 2 раунда обмена:

1. Обмен транзакциями
2. Обмен блоком

Этап обменна транзакциями происходит в течении ~10 минут, пока не будет найдена оказия одним из майнеров. После чего формируется блок и происходит этап обмена этим блоком.

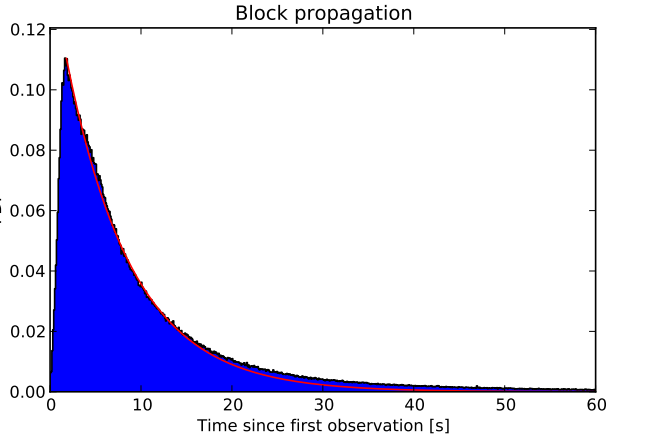
Процедура обмена блоком.



Сам алгоритм распространения блока выглядит следующим образом и состоит из 5ти этапов. Где на первом происходит приглашение узлом А к обмену с узлом В. Запрос за получение блока от узла В. А отправляет свой блок. В валидирует его и проверяет на действительность, потому что А мог его изменить или нарушить структуру. Только после этого В инициирует запрос обмена с своими узлами-соседями.

Этот алгоритм порождает ряд уязвимостей, связных со скоростью распостранения блоков по сети. На каждом переходе в широковещательной передаче сообщение подвергаеться задержке распространения. Задержка распространения представляет собой комбинацию времени передачи и локальной проверки блока или транзакции. Время передачи включает в себя объявление в форме сообщения приглашения, запрос от принимающей стороны стороны и доставку. Хотя сообщения invite и data имеют относительно небольшой размер (61B в большинстве случаев, так как немедленные сообщения содержат только один блок или транзакцию, о которой объявляется ), сообщение блока может быть очень большим - до 500 кБ на момент написания статьи. Прежде чем блок будет передан соседям узла, он проверяется. Проверка блока включает проверку каждой транзакции в блоке. Проверка транзакций, в свою очередь, требует случайного доступа к данным хранящимся на дисках. Пусть ti,j - это разница во времени между первым объявлением источника в сети и временем, когда узел j получает элемент i. Если узел o является источником элемента данных i, т.е. либо создателем блока, либо узлом, который создал транзакцию, то ti,o = 0. Времена ti,j, в которые узлы узнают о существовании узлы узнают о существовании элемента данных, следуют двойному экспоненциальному закону. Подобно о рандомизированному распространению слухов [12], распространение элемента данных можно разделить на две фазы: начальная фаза экспоненциального фаза роста, в которой большинство узлов, получающих сообщения inv сообщения, будут запрашивать соответствующий элемент данных, поскольку у них еще не имеют его, и фаза экспоненциального уменьшения, в которой большинство узлов, получающих объявления, уже имеют соответствующий элемент данных. узлов в сети. Наша реализация ведет себя точно как обычный узел с одной оговоркой: она не передает сообщения, транзакции или блоки. сообщения, транзакции или блоки. Он отслеживает, как транзакции и блоки распространяются по сети путем прослушивания объявления об их доступности в форме inv-сообщений. сообщения. Как только измерительный узел получает сообщение inv содержащее ссылку на блок, мы знаем, что узел. который отправил сообщение, получил и проверил блок. Измерительный узел собрал информацию о времени из блокчейна высотой 180'000 для 10'000 блоков. Временная информация содержит хэш блока, IP анонсирующего.

Для измерения задержки распространения мы реализовали протокол сети биткоина и подключились к большой выборке IP узлов и локальную метку времени, когда объявление было получено. Оценка для ti,j дается путем вычитания метки времени первого объявления блока из всех объявлений для этого элемента данных. На рисунке показана нормализованная гистограмма tb,j для всех блоков b в измеряемом интервале. Нормализация позволяет нам использовать это как аппроксимацию для функции плотностной вероятности скорости с которой узлы узнают о блоке. Обратите внимание: что мы не делаем различий между размерами блоков и вместо этого агрегируем по всем блокам. Медианное время до получения блока составляет 6,5 секунд, в то время как среднее значение составляет 12,6 секунды. Длинный хвост распределения означает, что даже после 40 секунд все еще есть 5% узлов, которые еще не не получили блок.



Такой тип уязвимости могут использовать как для преждевременного получения подтверждения блока так и для создания повторных транзакций. Второй тип атаки более опасный. Злоумышленник пытается отправить транзакцию жертве, где плотность узлов гораздо меньше. Параллельно создавая вторую транзакцию, которую отправляет в месте большого скопления мощностей, где с большей долей вероятности будет подтвержден следующий блок. Тем самым когда вторую транзакцию приняло большее процентов сети, первую, отправленую жертве, сеть игнорирует, и она не получает подтверждения.

Решение проблемы распространения блоков базируется на оптимизации валидации блока или транзакции. Разберем пункт 4 - verification. Он делиться на 3 подпункта:

1. Dificulty check
2. Transactions
3. Done

Dificulty check – проверка накопленной сложности, которая отражает вычислительные мощности в расчете хешей блоков этой цепочки.

Остальные шаги, такие как проверка транзакции на валидность, более временно затратные по этому предлагается отправлять инициирующее сообщение на обмен после пункта Dificulty check.

Существует второй подход обменна данными. Как только к нам приходить inv на обмен, вместе с запросов на получения данных, рассылается приглашения всем соседним узлам, строиться очередь приоритизации отправки транзакции, после проверки сложности объект рассылается согласно построенной очереди. Так же предложен и третий подход, предлагающий сменить топологию сети на star sub networ. Этот метод ведет к полному нарушению главной концепции криптовалют и децентрализованных сетей в целом. По этому является, хоть и достаточно производительной, так как каждая транзакция будет проходить через 1 узел, что способствует практически моментальному обмену, но достаточно сильно уменьшает отказоустойчивость сети.