Constellation

Операционная система на базе блокчейн с поддержкой микросервисов

25 ноября 2017

Резюме

Современные блокчейн-технологии, такие, как Bitcoin, Ethereum и другие, имеют системные ограничения в отношении проведения синхронных операций, которые делают их непригодными для массового внедрения в существующую техническую архитектуру распределенных пользовательских приложений. Механизмы консенсуса и архитектура смарт-контрактов не могут масштабироваться до уровня, который требуется для пользовательских приложений. Мы предлагаем отказоустойчивую, горизонтально масштабируемую, распределенную операционную систему, которая может реализовывать единые узлы в качестве мобильного клиента. Таким образом, мы представляем новую разработку безопасного консенсуса в современной архитектуре обработки данных, которая не требует сервера, с использованием асинхронной модели консенсуса ExtendedTrustChain, Proof-of-Meme, смарт-контрактов в качестве составных микросервисов с использованием JVM (виртуальная машина Java, далее - JVM) и первоначально реализованную в качестве конечного автомата на базе внешнего агента. Эта архитектура обеспечивает высокую пропускную способность транзакций, позволяя строить распределенные пользовательские приложения на базе Constellation.

Введение

Несмотря на децентрализацию в качестве цели, криптовалюты управляются и защищены все более централизованными сетями и майнинговыми компаниями, которые контролируют безопасность сети[1]. Изначальная цель блокчейн-технологии заключалась в обработке финансовых транзакций. С ростом новых сетей после Bitcoin, таких, как Ethereum, она приобрела более широкий смысл, который заключается в возможности проведения распределенных вычислений посредством EVM (виртуальная машина Ethereum, далее - EVM) и использовании логической

[1] Digital Currency Group «Соглашение о масштабировании Bitcoin на конференции Consensus 2017» medium.com/@DCGco

системы смарт-контрактов. Несмотря на то, что уже существуют разнообразные реализации этих распределенных протоколов, они изолированы от существующего программного обеспечения и, таким образом, существует барьер для их использования[2]. В первую очередь, это дорогостоящий и трудоемкий процесс, который представляет собой разработку, развертывание и поддержание устойчивых к сбоям распределенных приложений на базе незнакомых языков программирования и шаблонов проектирования. Кроме того, масштабирование на базе существующих блокчейн-технологий такого приложения в соответствии с требованиями по производительности даже небольшой компании практически невозможно. Чтобы преодолеть эти технические проблемы, блокчейн-технология нуждается в автономном протоколе, который будет функционировать как горизонтально масштабируемая, экономичная и эффективная распределенная операционная система.

Bitcoin был создан для решения проблемы распределенного консенсуса в отношении финансовых транзакций, однако, он функционировал на базе энергоемкого процесса консенсуса, известного как доказательство выполнения работы. Это обеспечивает вертикальную вычислительную мощность от небольшого количества людей с хорошо развитой и, следовательно, централизованной вычислительной мощностью[3]. Таким образом, сетевая безопасность и все вознаграждения попадают в руки одних и тех же людей. Споры о масштабировании привели к созданию нескольких Bitcoin форк, таких, как Bitcoin Cash и Bitcoin Gold. Эти отдельные фракции сильно различаются и наносят вред экосистеме. Устаревшие требования к пропускной способности (ограниченный размер блока, равный 1 МБ) и медленный алгоритмом привели к резкому увеличению времени, которое необходимо для транзакций и транзакционных сборов. Ethereum использует аналогичный, но ASIC-устойчивый (ASIC - специализированная интегральная микросхема) алгоритм консенсуса на базе доказательства выполнения работы, и постепенно переходит к принципу консенсуса на базе доказательства доли владения, Casper. Доказательство доли владения представляет собой демократический дисбаланс, в котором участники, у которых больше денег, принимают решения на основе консенсуса. Ethereum - это первая технология, в которой начали использовать смарт-контракты, однако, в силу их синхронного исполнения, работа с распределенными пользовательскими приложениями сильно затруднена. Совместное выполнение является ключом к работе с продуктами, которые ориентированы на пользователя, и Constellation достигает этого благодаря смарт-контрактам в архитектуре с поддержкой микросервисов.

---

[2] J. Evans «Блокчейн - это новый Linux, а не новый интернет» techcrunch.com

[3] «Bitcoin стал централизованным?» www.trustnodes.com

Что отличает Constellation?

Сообщество блокчейн-разработчиков уже долго находится в поиске новых вариантов реализации технологии распределенных реестров для решения проблемы масштабируемости, однако, решения до сих пор не найдено. Проблема все еще остается: как мы можем обрабатывать больше транзакций и предоставлять больше услуг за меньшее время с меньшими затратами? Мы предлагаем подход горизонтального масштабирования, который применяет методы, подобные MapReduce.

Горизонтальная масштабируемость - это применение параллельного программирования; это означает, что по мере подключения пользователей к сети увеличивается пропускная способность транзакций. MapReduce - это процесс разделения вычислений на простые операции, которые затем передаются в асинхронный DAG (направленный ациклический граф, далее – DAG), тем самым повышая эффективность уже параллельной программы.

Протокол Constellation реализует горизонтально масштабируемую архитектуру блокчейн, известную как Extended Trust Chain, с одноранговым уровнем, известным как протокол Gossip, который может быть развернут на мобильном устройстве. Constellation использует смарт-контракты и архитектуру с поддержкой микросервисов, позволяя подключать доступные сервисы и объединять их в распределенные приложения, благодаря распознаванию соглашения SLA (соглашение об уровне услуг) и/или сигнатуры типа.

Протоколы Gossip позволяют крупным сетям сообщать общее состояние сети в масштабе, который в несколько раз превосходит масштаб, допустимый для существующей технологии блокчейн[4]. В сети такого типа каждый член сети отслеживает соседние узлы и когда он получает новые сообщения, он в свою очередь передает это сообщение всем соседним узлам. Подобная сеть подразумевает некоторые интересные математические свойства, но в нашем случае она позволяет большому количеству подключенных устройств делиться состоянием сети и достигать консенсуса в масштабах, которые недопустимы в существующих реализациях блокчейн (см. Рис. 1).

Transaction Throughput (tps)			
Bitcoin	Ethereum	IOTA (Tangle)	Hylochain (1200 nodes - fixed neighbors)
3-4	15	500-800	4000-4800

[4] R. van Renesse «Блокчейн на базе Gossip?» zurich.ibm.com

Рисунок 1: Пропускная способность сопоставимых блокчейн для сравнения (transaction throughput – пропускная способность; nodes – узлы; fixed neighbors – зафиксированные соседние узлы).

Одним из подходов к консенсусу блокчейн, который позволяет горизонтальное масштабирование, является ExtendedTrustChain. ExtendedTrustChain представляет собой несколько типов узлов, для которых характерны разделение обязанностей и различные роли в сети, и которые асинхронно управляют различными аспектами протокола.

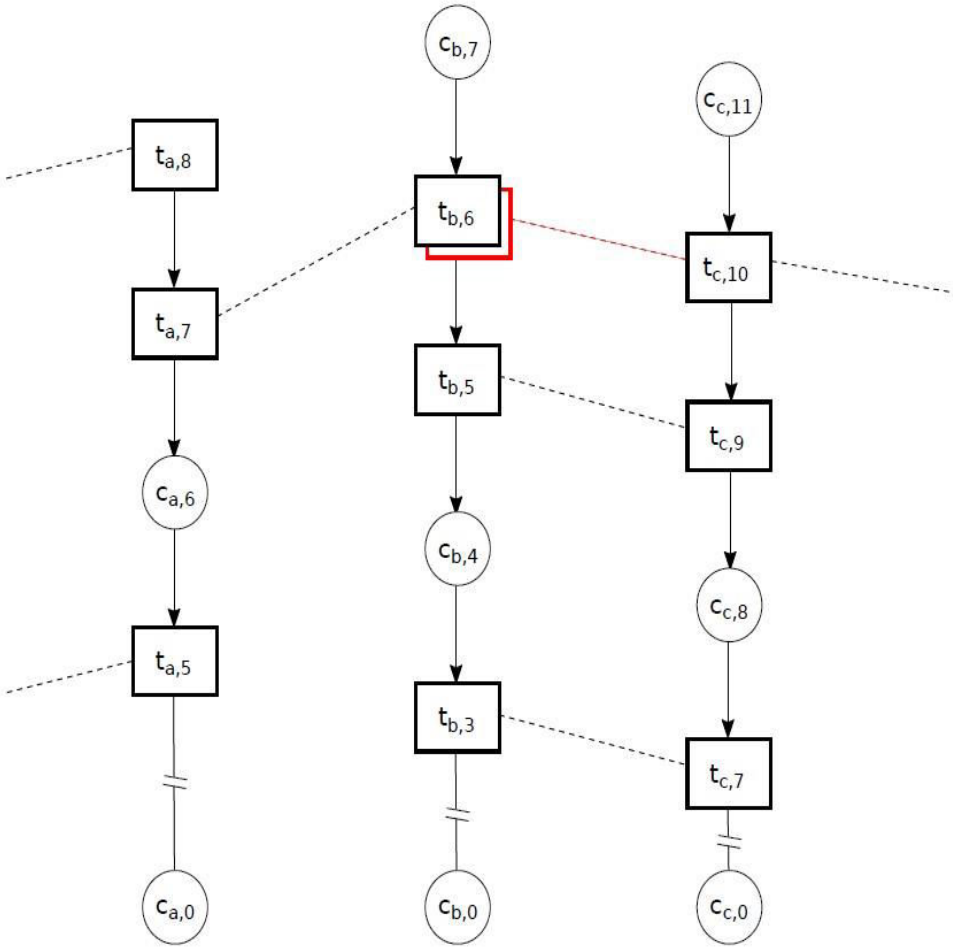

Рисунок 2: Диаграмма Extended Trust Chain, K. Cong и др.

Узлы

Так называемые, базовые узлы отправляют транзакции и размещают индивидуальную цепочку участника (при совершении транзакции каждая индивидуальная цепочка, вместе с другими цепочками в сети образует DAG и хешируется в линейные блоки).

Контрольные точки

Существуют узловые процессы, которые достигают консенсуса по транзакциям (которые заполняют буфер и хешируются в блоки контрольных точек). Эти коллекции транзакций хешируются и становятся следующим блоком в основной цепочке.

Агент проверки

Наконец, есть узлы агента проверки, которые запускают процессы проверки и размещают данные о текущем состоянии цепочки и, следовательно, добавляют данные о состоянии/истории цепочки узлов в сеть.

EVM против JVM

Разработка распределенных приложений в виде операций MapReduce на базе смарт-контрактов была предложена Ethereum с выходом обновления, известного как Plasma[5]. Однако, в качестве приложения к EVM, мы по-прежнему видим те же недостатки синхронной архитектуры в отношении масштабирования. Язык программирования Solidity был создан для устранения рисков недетерминированного выполнения кода в разработке смарт-контрактов. Необходимость детерминированной инструкции связана с проблемой расчета стоимости запуска смарт-контракта и в качестве превентивной меры против атак DDOS (распределенная атака типа «отказ в обслуживании»), а также против спама в сети. Однако, если скомпилированный байт-код может быть нотариально заверен в цепочке и проверен в JVM, то детерминированная компиляция становится не обязательным требованием[6]. Тогда появится возможность сокращения потребление gas, обхода лимитов на gas и использования смарт-контрактов со сложными микросервисами.

Разработка распределенных приложений путем соединения общих блоков логична сама по себе, это как сборка Legos, где каждая деталь представляет собой отдельный микросервис. Далее, мы реализуем

5 V. Buterin и др. https://plasma.io/plasma.pdf
6 M. Hearn Corda: «Распределенный реестр» docs.corda.net

функциональную совместимость на базе цепочки, которая позволяет блокчейн-компаниям предоставлять распределенные приложения для повторного использования в качестве строительных блоков для более сложных распределенных приложений. По-настоящему децентрализованная экосистема не может существовать с неделимыми смарт-контрактами, тем самым делая этот тип повторного использования кода невозможным и неэффективным.

Использование JVM в качестве канала для распределенных приложений является отраслевым стандартом сообщества Big Data. Пользовательские приложения развертываются в виде набора микросервисов в группах автомасштабирования, которые полагаются на простую конфигурацию, которая, в свою очередь, предоставляется JVM. В этих сценариях тысячи кластеров узлов могут быть созданы и автоматически масштабированы в виртуальных экземплярах AWS. Такую же масштабируемость можно реализовать с помощью архитектуры микросервисов Constellation, которая позволяет использовать ресурсы по мере необходимости; тем самым обеспечивая динамически масштабируемую архитектуру распределенного приложения.

Данный тип распределенной архитектуры по своей сути требует модульный принцип организации, который возможен благодаря функциональному программированию. Поэтому мы используем язык программирования Scala для создания интерфейса, который может быть реализован с помощью любого языка JVM.

Когда микросервисы проектируются как конкретные типы, распределенные приложения могут быть составлены непосредственно в исходном коде, что позволяет проводить проверку во время компиляции. Разработчики могут создавать новые функции, объединяя смарт-контракты применимых типов в составные приложения; совокупность вычислений, следовательно, составлена в виде одного или нескольких микросервисов, или распределенных приложений. Интерфейс смарт-контракта Constellation будет разработан с учетом ковариантной и контравариантной типизации, а это означает, что, если мы будем объединять совместимые микросервисы или распределенные приложения, мы будем незамедлительно знать об этом. Таким образом мы можем создавать различные группы, включая смарт-контракты от других блокчейн, например, во время ожидания решения проблем ликвидности как в случае кроссчейн Polka-dot[7].

Учитывая характер нашего функционального программирования, стоит отметить, что наш протокол консенсуса может быть описан теоретико-категориальным определением гилеморфизма.

[7] G. Wood и др. https://github.com/w3f/polkadot-white-paper/raw/master/PolkaDotPaper.pd

HyloChain - Архитектура консенсуса

Мы назвали нашу архитектуру консенсуса HyloChain, потому что базовая архитектура консенсуса Constellation является голоморфной. Один цикл процесса консенсуса принимает полученный хеш-блок предыдущего цикла и добавляет его в качестве регулярной транзакции в пул транзакций. Процесс заполнение пула транзакций схож с операцией развертывания. Как только к блоку контрольной точки добавляется предыдущий цикл плюс новые транзакции, он хешируется. Этот процесс схож с операцией свертывания.

Определение: HyloChain

Гилеморфизм $h : A \to C$ может быть определен с помощью его отдельных анаморфотных и катаморфных частей. Посмотрим на определение гилеморфизма:

$$h = [(c,\oplus),(g,p)] \tag{1}$$

Анаморфная часть может быть определена с помощью унарной функции $g : A \to BA$, определяющей список элементов B посредством повторного процесса развертывания, и предиката $p : A \to$ Булева переменная, что обеспечивает условие завершения.

Катаморфная часть может быть определена как комбинация начального значения $c \in C$ для свертывания и бинарного оператора $\oplus : B \times C \to C$, используемого для выполнения свертывания.

В нашем случае мгновенный обмен сообщениями - это наш анаморфизм, а хеширование этих сообщений плюс результат предыдущего блока - наш катаморфизм.

Если оператор определен как криптографическая хеш-функция, $\oplus$ = CHash и $g\ n = (n,\ n - 1)$ и $p\ n$ = False (поскольку наша цепочка не имеет условия завершения) для n-й итерации, HyloChain однозначно определяется как:

$$HyloChain = [(\text{GenesisBlock},\oplus),(g,p)] \tag{2}$$

где GenesisBlock - наш исходный элемент, а HyloChain - используемый блок генезиса, когда мы выполняем развертывание.

Связь с ExtendedTrustChain

Наша архитектура консенсуса HyloChain основывается на архитектуре ExtendedTrustChain и дополняет ее следующим образом. В ExtendedTrustChain каждый узел работает как учетная запись, поддерживает историю своей собственной сети и полагается на порядок транзакций для подтверждения значений (аналогично с HashGraph[8]). Транзакции подписываются инициатором, а контрагент затем транслируется в сеть (через мгновенный обмен сообщениями). Делегаты осуществляют консенсус в отношении транзакций в блоке контрольных точек и выбираются на основе их репутации. Результат консенсуса транслируется снова и добавляется как обычная транзакция к следующему блоку[9].

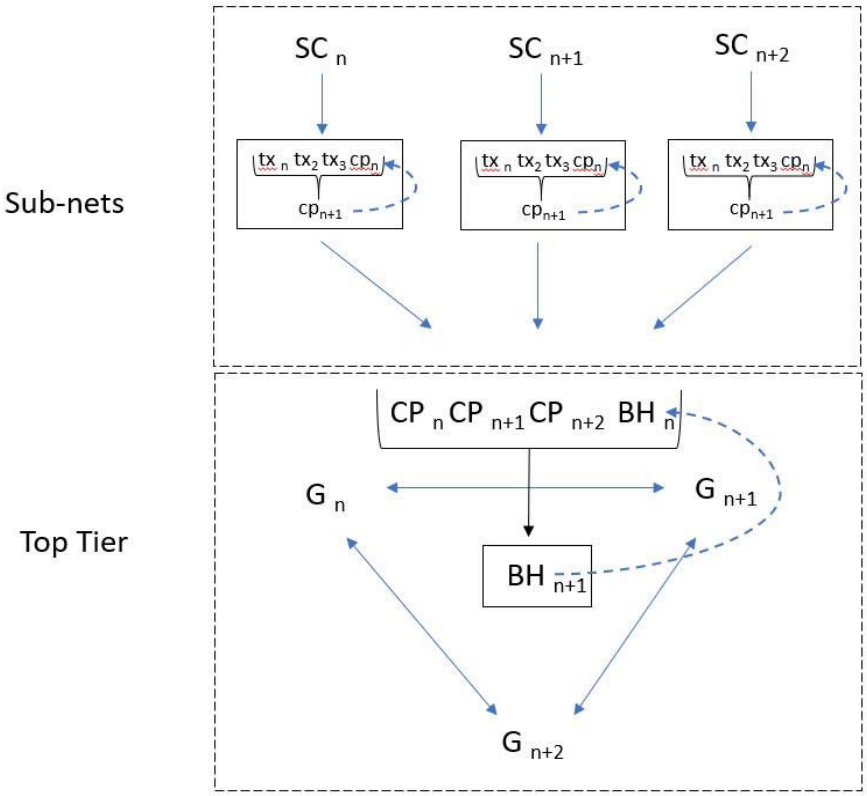

Рисунок 3: HyloChain (sub-nets – подсети; top tier – верхний слой)

8 L BAIRD и др. http://www.swirlds.com/downloads/SWIRLDS-TR-2016-01.pdf

9Kelong Cong, и др. https://repository.tudelft.nl/islandora/object/uuid:86b2d4d8-642e- 4d0f-8fc7-d7a2e331e0e9?collection=education

Отказоустойчивость

Отказоустойчивость HyloChain, разработанная Constellation использует типичную византийскую модель консенсуса. Вероятность того, что противник будет контролировать консенсус сети приближается к 1/2, если его контроль над участниками консенсуса приближается к 1/3 [10]. Чтобы обеспечить безопасность сети, мы можем использовать эти границы для выбора конкретных параметров для настройки отказоустойчивости, увеличивая пропускную способность транзакций и уменьшая задержку.

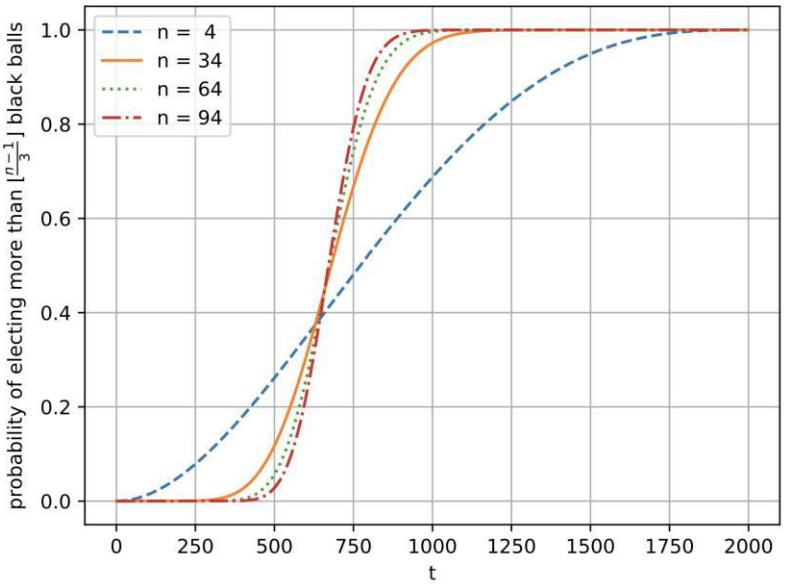

Рисунок 4: Вероятность выбора более чем (n - 1)/3 (обозначается как черные шары, как это вытекает из объяснения гипергеометрического распределения) при разных значениях t (где t - это количество византийских актеров) с фиксированным размером популяции в размере 2000, K. Cong и др.

Кроме того, основываясь на нашей византийской процедуре консенсуса (мы предварительно используем ACS (сервер асинхронной передачи данных), как это реализует M. Бен-Ор и др., который используется Honeybadger для алгоритма BFT (пакетная передача файлов, далее - BTF)) и двойном подписании транзакций, разветвить сеть будет невозможно. Единственным результатом неправильного консенсуса может стать цензурирование отдельных транзакций из блока контрольных точек, т.е. потеря транзакции, во время консенсуса. Это не приведет к потере капитала и решением будет повторная отправка транзакции в сеть.

10 M. Pease, R. Shostak и L. Lamport, «Достижение согласия при наличии отказов», журнал ACM (JACM), том. 27, вып. 2

Связь пропускной способности является линейной по отношению к числу участников, учитывая способность к горизонтальному масштабированию, однако, линейное увеличение отказоустойчивости требует значительного увеличения стоимости обмена данными[11]. Таким образом, мы должны определить параметры консенсуса, которые обеспечивают максимальную пропускную способность с ограниченной отказоустойчивостью и временем подтверждения транзакции. Как мы можем масштабировать сеть с такими границами? Наш подход состоит в том, чтобы сконструировать блокчейн нашей сети как иерархию подсетей (рис. 1), каждая из которых исполняет свой собственный консенсус, результаты которого передаются на более высокий уровень, который обрабатывает эти контрольные точки как обычные транзакции. Этот подход очень похож на Chain Fibers (также с точки зрения использования агентов проверки), однако отличается тем, что мы применяем метод понижения размерности локально-чувствительного хеширования для увеличения размера пула транзакций. Поскольку эффект этой архитектуры заключается в уменьшении сложности консенсуса из среднего случая O(n3) или квадратичного для каждого узла, пользуясь тем, что алгоритм консенсуса BFT Honeybadger может быть уменьшен до O(1) на делегата[12], наш коэффициент сложности будет выглядеть следующим образом:

$$O(n) \rightarrow O(m) : m << n \qquad (3)$$

Таким образом, это сублинейная функция по отношению к числу узлов в подсети.

В этом случае мы представляем эти блоки контрольных точек в качестве локально-чувствительных хеш-блоков. Каждая подсеть формирует локально-чувствительный хеш-блок, и отправляет его в «родительскую» сеть (Galaxies, описано ниже). Затем эта «родительская» сеть рассматривает локально-чувствительные хеш-блоки как обычные транзакции в следующем цикле консенсуса. Для повышения производительности сети выбор соседних узлов в подсети должен основываться только на минимизации задержек, так как каждый узел отслеживает собственную историю транзакций.

На каждом уровне масштабирования используется самоподобная архитектура передачи мгновенных сообщений, позволяющая узлам участвовать на каждом уровне, с необходимостью изменения только ресурсов, которые они предоставляют.

---

[11] стр. 50, K. Cong, и др.

12 M. Ben-Or и R. El-Yaniv. «Оптимальная, интерактивная и стабильная согласованность. Распределенная обработка данных», 16(4):249?262, 2003

Все узлы в Constellation могут быть «спящими», то есть они могут присоединиться или покинуть сеть в любое время. Когда новые узлы присоединяются к сети, их ресурсы распределяются в подсетях. После того, как подсеть достигнет достаточных уровней пропускной способности транзакций и криптографической защиты (количество непосредственных участников рынка по сравнению с общим количеством участников), как указано выше, узлы данных должны будут сформировать новую подсеть. Таким образом, новые подсети будут динамически распределяться по мере того, как участники уходят и присоединяются к сети. Цель этой самоподобной структуры и нашей модели выбора делегатов, приведенной ниже, заключается в том, чтобы включить динамическое автоматическое масштабирование на основе сетевых ресурсов, что обеспечит постоянную пропускную способность. Данная архитектура демонстрирует закон власти, основанный на сетях, способных к любому масштабированию, и которые известны отказоустойчивостью в сложных системах, таких как распределенная обработка данных[13].

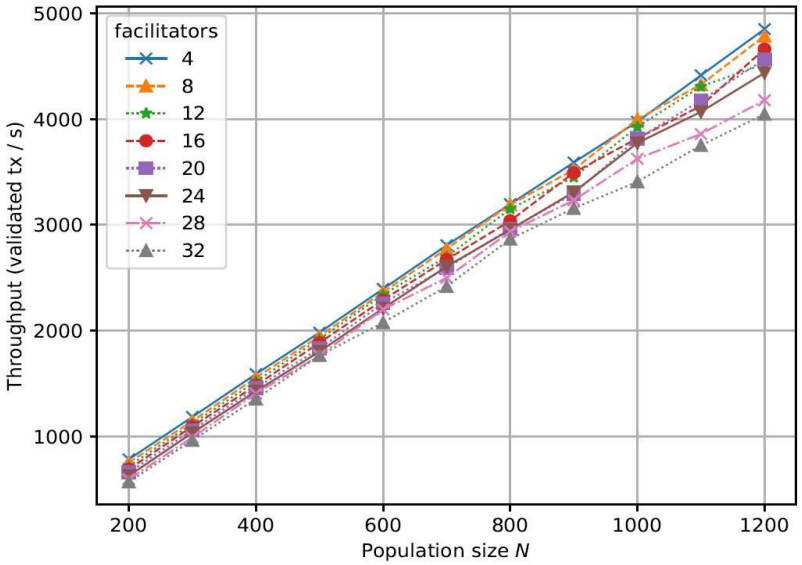

Рисунок 5: Пропускная способность HyloChain и количество участников, K. Cong и др.

13 Ravasz, E.; Barabasi (2003). «Иерархическая организация в сложных сетях», журнал Phys. Rev. E. 67: 026112.

Модель выбора делегатов: Proof-of-Meme

Отказоустойчивость TrustChains может быть улучшена с помощью системы репутации для выбора делегатов[14]. Поэтому мы представляем Proof-of-Meme; метод распределенного консенсуса, который использует историю участия узла в выборе делегата. Другими словами, meme - это единица, представляющая поддающуюся имитации идею или поведение, которые могут быть распространены. Таким образом, в нашем случае она представляет собой правомерное поведение в Constellation, которое вознаграждается и является неким стандартом для улучшения общей репутации узлов в системе. Proof of Meme - это меритократия по сравнению с доказательством доли владения, которое представляет собой плутократию.

Meme в нашем смысле - это вектор признаков, соответствующий учетной записи каждого узла; в более простом смысле - это матрица изменяющихся значений, используемых в качестве входных данных для детерминированного алгоритма машинного обучения. Мы идем по стопам REGRET, который использует объектную модель функций для описания репутации узла внутри сети; технически это тензорное произведение пространств признаков[15]. Соответствующая детерминированная модель машинного обучения, использующая эту объектную модель (meme) в качестве пространства признаков, используется для определения оценки репутации узла и транзитной вероятности выбора узла для достижения консенсуса.

Meme является основой оценки репутации, а оценка репутации влияет на вероятность выбора для консенсуса.

Выбор делегатов широко изучался в контексте улучшения алгоритма BFT в асинхронных механизмах консенсуса[16]. В частности, в своей работе над GURU А. Бирюков и др. показали, что типичная отказоустойчивость 1/3 вредоносных узлов в византийском консенсусе может быть улучшена до (и менее) 1/2. Мы добавили эти условия проверки для выбора наших делегатов в Proof-of-Meme.

Существует множество исследований, связанных с Extended Trust Chain, которые решают проблему моделирования репутации, устойчивой к атакам Сивиллы. В частности, P. Otte[17] использовал алгоритм NetFlow для

14 стр. 50, K. Cong, и др.

15 J. Sabater, «REGRET: модель репутации для сообществ» http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.8.7554rep=rep1type=pdf

16 «Guru: универсальный модуль репутации для распределенных согласованных протоколов», А. Бирюков и др. https://eprint.iacr.org/2017/671.pdf

17 P. Otte, «Механизмы доверия в распределенных системах, устойчивые к атакам Сивиллы»

https://repository.tudelft.nl/islandora/object/uuid:17adc7bd-5c82-4ad5-b1c8-a8b85b23db1f/datastream/OBJ/

предотвращения атак Сивиллы, а также модификации PageRank, а именно Temporal PageRank, для обновления состояния репутации. Мы планируем вначале воссоздать Temporal PageRank, а затем улучшить его, где это возможно, во время анализа производительности.

Использование модели репутации для выбора делегата с помощью Proof-of-Meme в сочетании с ролью каждого узла как отдельной учетной записи позволяет нашей свободной от ограничений сети обеспечивать прозрачность, которая стимулирует правомерное поведение в сети. Все транзакции и история консенсуса для каждого узла (и транзитно микросервисов) публично заверены нотариально. Это позволяет нам добиться такого уровня доверия, которого нет в существующих технологиях. Они, в свою очередь, обеспечивают правомерное поведение с помощью транзакционных сборов и неравных механизмов консенсуса.

Объяснение Proof-Of-Meme

Мы рассказали о трех направления работы, которые решают проблемы возможного сговора в распределенном консенсусе. Это все компоненты для выбора делегатов, как указано ниже. Мы будем использовать подход REGRET для разработки объектной модели для meme, который станет пространством признаков для оценки нашей репутации. Мы адаптируем Temporal PageRank для обновления состояния репутации. Мы используем условия проверки GURU для нашего метода выбора делегатов. Дополнительная информация будет добавлена в процессе разработки. Создание объектной модели репутации, использование детерминированной модели оценки репутации и реализация алгоритма выбора делегатов являются ключевыми элементами нашего развития.

Теперь мы подробно расскажем, как эта оценка используется для выбора делегата. Каждая функция представляет собой утилиту узла для сети (была ли замечена ошибка во время консенсуса, сколько памяти может быть зафиксировано и т. д.). Meme можно преобразовать в числовое значение с помощью тестовой модели, поместив его в функцию, как описано выше, и именно так мы будем количественно оценивать репутацию. Чтобы предоставить механизм, который подталкивает новые узлы к улучшению их репутации, распределение вероятностей баллов осуществляется с помощью сегментов памяти фиксированного размера; распределение определяет вероятность того, когда определенный участник будет выбран. С вычислительной точки зрения это, вероятно, будет реализовано в виде алгоритма кластеризации, который повторно группирует узлы в качестве подпрограммы. Это должно максимизировать пропускную способность, уменьшая количество прямых участников рынка, но сохраняя тот же уровень отказоустойчивости и времени подтверждения. Предпочтение будет отдаваться участникам с хорошей репутацией, но новые участники также будут иметь возможность участвовать и строить свою репутацию в сети. Журналы

производительности будут нотариально засвидетельствованы в цепочке, поэтому все действия участника, которые вредят сети, например, предоставление ошибочных ресурсов, отразятся на его репутации, в то время как репутация высокопроизводительного и заслуживающего доверия участника будет увеличиваться. Наша идея выбора делегата в индуктивном случае выглядит следующим образом:

1) Выполняется консенсус.

2) Хеш-блок передается в детерминированный алгоритм, который создаёт обновленные оценки meme для каждого делегата. Это выполняется на высшем уровне консенсуса (Galaxy), который отвечает за выбор непосредственных участников рынка.

3) Эта константа используется для перетасовки нашего дистрибутива (перемещение meme между сегментами на основе новых данных их производительности).

4) Предыдущий хеш-блок (результат последнего консенсуса) используется для сортировки содержимого каждого сегмента, и топ N из каждого сегмента (в соответствии с распределением вероятности) выбирается для этого цикла. Meme участника консенсуса, который вредит сети, например, предоставление ошибочных ресурсов (подлежит проверке в журналах), будет состыкован на следующих циклах выбора участников консенсуса.

Поскольку репутация meme является ключевым моментом, она исключает необходимость в транзакционном сборе. Микросервисы могут использовать meme с хорошей репутацией для хостинга услуг, что будет более выгодно, чем получение транзакционных сборов за достижение консенсуса. Вместо повышения цен, когда время ожидания довольно высоко, работа meme с низким рейтингом репутации будет ограничена; их транзакции будут обрабатываться с более низкой степенью приоритетности. Meme в этом случае будет предоставлять ресурсы (участвовать в консенсусе), тем самым увеличивая пропускную способность для себя и разрешая проблемы с пропускной способностью сети.

Владение Meme

Constellation не будет взимать комиссию за транзакции в сети. Репутация будет нашим механизмом предотвращения временных задержек и враждебных атак. Для расширения нашей сети будет использоваться стимул в виде новых токенов после первоначального блока блокчейн (который будет выделять токены для участников ICO) с возможностью возврата до фиксированной даты, когда создание токенов будет завершено. Токены будут распределены на основе репутации каждого meme.

Рассмотрим один из сценариев системы торговли, в частности, торговый аппарат. Торговый аппарат, который работает на базе Constellation, может быть реализован как микросервис, который принимает участие в консенсусе. Репутация meme будет расти с течением времени, а короткое время транзакций будет обеспечено даже в условиях необходимости высокой пропускной способности. Это жизненно важно, так как приложения в реальном мире нуждаются в высокой пропускной способности для обеспечения конкурентного обслуживания. Это означает, что даже если у узла, который обслуживает торговый аппарат низкая репутация meme, высокая репутация торгового аппарата гарантирует, что потребитель получит свой товар даже в условиях необходимости высокой пропускной способности.

Одноранговая архитектура

В наших предыдущих разделах речь идет об одноранговой архитектуре, и теперь это понятие будет детализировано.

Stars

Stars - это базовый элемент в Constellation. Чтобы напрямую взаимодействовать с сетью, пользователи создают узел типа Star с помощью своего устройства, и транзакции проводятся через этот экземпляр Star. Каждый элемент Star содержит локальную цепочку, состоящую из его истории в сети. Эта локальная цепочка используется для принудительного упорядочения и идентифицируется открытым ключом, частная копия которого используется для подписи транзакций. Сам элемент сети Star является упрощенным, совместимым с мобильными устройствами клиентом, который необходим для взаимодействия пользователя с сетью. Однако, при желании, Star также может принять участие в консенсусе, что позволит ему оценить репутацию своего meme. Если Star принимает участие в консенсусе, то он присоединяется к группе элементов Star, которую мы называем Star Cluster.

Star Clusters

Star Cluster - это коллекция элементов типа Star, избранных для участия в консенсусе. Общее число элементов типа Star ограничено показателем, который будет определен после достаточного экспериментального и статистического анализа, как описано выше. Когда мы достигаем этого порога, создается новый элемент типа Star Cluster, и каждый из них формирует локально-чувствительные хеш-блоки. Эти локально-чувствительные хеш-блоки обрабатываются как обычные транзакции и хешируются элементами типа Galaxies, а именно их компонентами, типа Black Holes.

Galaxies

Элементы типа Galaxies схожи с агентами проверки в ExtendedTrustChain, но они также служат в качестве группы автоматического масштабирования, выделяя ресурсы для новых элементов типа Star Cluster и поддерживая репутацию meme. Репутация meme рассчитывается путем анализа журналов консенсуса. Метаданные о содержимом предложенного блока, времени ожидания и сбоях отправляются с помощью локально-чувствительных хеш-блоков элементам типа Galaxies. Как только Galaxies получат эти данные, репутация meme обновляется до того момента, пока не будет выбран новый образец для следующего цикла консенсуса. Galaxies, поскольку они являются агентами проверки, удаляют недействительные транзакции и назначают делегатов в Star Clusters. Метаданные о производительности Galaxy хранятся в элементах типа Black Holes. По мере того как Stars достигает максимального уровня репутации, они могут получить право функционировать как Galaxies.

Black Holes

Black Holes - это блоки локально-чувствительных хеш-блоков. Это эквивалентно упоминанию их как блоков в цепочке блоков. Galaxies хранят полную историю блокчейн.

Смарт-контракты в качестве микросервисов

Распределенные системы с поддержкой масштабирования показывают отличные результаты в отношении архитектуры обработки данных, которая не требует сервера. В случае распределенной операционной системы это может быть достигнуто в виде сети распределенных микросервисов. Таким образом, смарт-контракты Constellation сами по

себе являются микросервисами, работающими на базе JVM. Они могут отправлять транзакции, подписываться в качестве контрагента и исполнять консенсус. Цель заключается в том, чтобы сами микросервисы работали как элемент типа Star с соответствующим meme, предоставляя услуги для согласованных сумм. Они могут выполнять ту же роль, что и смарт-контракты в Ethereum или Counterparty, но, кроме того, они обеспечивают более сложную логику, используя существующую кодовою базу в экосистеме JVM. Кроме того, они могут взаимодействовать с внешними программами через упрощенный интерфейс RPC. Если эти микросервисы построены на базе конкретных соглашений об уровне услуг или еще лучше, сигнатуре типа, их составная логика может быть скопирована и добавлена впоследствии в распределенные приложения интуитивно. Именно здесь вступают в игру операторы MapReduce. Микросервис на базе смарт-контрактов может быть разработан для отправки и получения моделей данных, которые улучшают вычислительную сложность (с учетом нашей асинхронной архитектуры) и могут быть перепрофилированы для новых приложений.

Принимая во внимание проведенные выше аналогии со звёздным небом, стоит отметить, что распределенные приложения в Constellation - это и есть созвездия (от англ. – constellation). Поскольку каждый микросервис является элементом типа Star, коллекции микросервисов, находящиеся в цепочке и/или составные части, могут быть соединены линией, представляющей приложение. И, пожалуй, стоит отметить, что если вы попробуете нарисовать эту схему, то увидите созвездие.

Constellation как операционная система на базе блокчейн

Вышеупомянутая архитектура обработки данных, не требующая сервера, представляет собой экземпляр распределенной операционной системы. Целью операционной системы является предоставление интерфейса для утилизации ресурсов основного оборудования. Это является неотъемлемой частью разработки приложений, позволяя использовать языки программирования высокого уровня и интуитивно понятные пользовательские интерфейсы. Распределенная операционная система отличается тем, что она предназначена для обеспечения интерфейса для базовых ресурсов распределенного вычислительного кластера. Блокчейн по своей сути является распределенной системой, предназначенной для создания приложений, совместимых с другими блокчейн-технологиями, которая нуждается в распределенной операционной системе для использования всей мощности базового кластера. Как правило, для операционных систем требуется программа, которая поддерживает базовое состояние оборудования; это называется ядром. Также, можно построить распределенную операционную систему на базе архитектуры обработки данных, не требующей сервера. В нашем сценарии, как и в MicroOs, операционная система представляет собой

набор микросервисов. Экстраполируя данную идею, с точки зрения разработки приложения, микросервисы являются строительными блоками для комплексных приложений, построенных с помощью Constellation, как, например, Legos. Цель состоит в том, чтобы упростить разработку приложений с помощью существующих микросервисов в качестве строительных блоков. Таким образом, благодаря интуитивно понятному пользовательскому интерфейсу, а также моментам, которые упоминались выше, теперь не только технические специалисты могут разрабатывать распределенные приложения с помощью Constellation.

Как мы уже говорили выше, способность индивидуума беспрепятственно (не используя технические инструменты) воплощать идею в приложение, можно охарактеризовать как трансмутацию потенциальной энергии из Ноосферы в материальный мир. Таким образом, мы решили назвать нашу валюту noOs, в качестве аналогии с потенциальной энергией Ноосферы, которая вдохновляет нас на новые цели и идеи.

Заключение

Мы представили новый вариант разработки безопасного консенсуса в современной архитектуре обработки данных, не требующей сервера, которая позволяет приложениям использовать блокчейн-технологию. Наш подход использует метод понижения размерности локально-чувствительного хеширования в качестве механизма «разделяй и властвуй», обеспечивающего безопасность цепочки в интернете. Элемент, который мы назвали meme обеспечивает более высокий уровень криптографической безопасности и потенциальную возможность отказа от транзакционных сборов. Кроме того, разрабатывая нашу модель распределенной архитектуры, мы проводим аналогию со звездным небом. Таким образом, вы увидите интуитивную корреляцию между распределенными приложениями и самоподобными шаблонами масштабирования, наблюдаемыми в природе.