

THE HIVE  
ТЕХНИЧЕСКИЙ ДОКУМЕНТ  
Версия 1.0



**RAYMON**

Raymon - Blockchain  
Platform  
Глейм Семен  
2018

A decorative graphic in the top-left corner consisting of grey circuit-like lines and blue geometric shapes (triangles and squares) on a black background.

# СОДЕРЖАНИЕ

РЕЗЮМЕ	3
ВВЕДЕНИЕ И ОПИСАНИЕ СИСТЕМЫ	4
ХРАНЕНИЕ ДАННЫХ	8
ВЕСА И ПРОЧЕЕ	9
УСТОЙЧИВОСТЬ СИСТЕМЫ И ОГРАНИЧЕНИЯ	10
ПУЛЫ	15
PROOF – OF – ME	16
ОБМЕН	18
БИЗНЕС - ПЛАТФОРМА PROFIT	22

## The Hive

A decorative graphic in the top-left corner consisting of a network of grey lines representing circuit traces, with several blue diamond shapes of varying sizes and orientations scattered throughout.

### РЕЗЮМЕ

В этой статье мы анализируем математические основы криптовалюты PaymonCoin (PMNC). Главной особенностью нашей криптовалюты является "Хайв" (от англ. Hive - улей), ориентированный ациклический граф (DAG) для хранения транзакций. Хайв является следующим эволюционным шагом технологии блокчейн, а также предлагает некоторые особенности, необходимые для создания системы микроплатежей в отсутствие комиссий и с достижением максимальной скорости транзакций, о которых мы расскажем далее.

Важным вкладом этой статьи является семейство алгоритмов Марковской цепи Монте-Карло (МЦМК). Эти алгоритмы осуществляют выбор вершин в графе для новых транзакций.

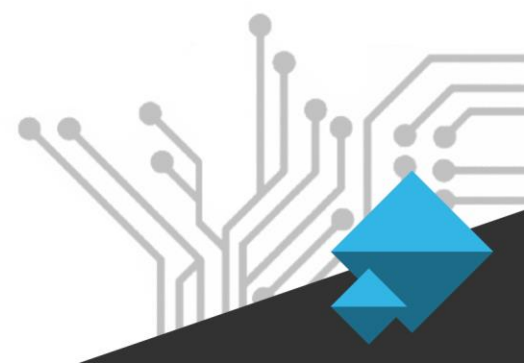
Контактная информация: [s.a.gleim@paymon.org](mailto:s.a.gleim@paymon.org)

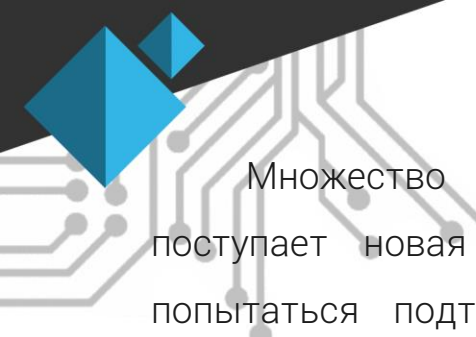


Торговля в интернете в основном полагается на финансовые учреждения, выступающие в качестве доверенных третьих сторон для обработки электронных платежей. Данная система работает достаточно хорошо для большинства операций, но страдает от слабости, присущей моделям, основанным на доверии. Bitcoin должен был стать системой, основанной на криптографических доказательствах вместо доверия, но он в свою очередь имеет ряд других минусов: высокая стоимость транзакций, большая задержка подтверждений, зависимость от майнеров. При уменьшении количества майнеров, уменьшится и количество подтвержденных транзакций за единицу времени. Возникает необходимость в электронной платежной системе, основанной на криптографических доказательствах, но не зависящей от майнеров.

В этой статье мы обсудим инновационный подход, который не совмещает в себе технологию Blockchain. Этот подход в настоящее время реализуется как криптовалюта под названием RaymonCoin. Цель этой статьи - сосредоточиться на общих особенностях Хайв и обсудить проблемы, возникающие при попытке избавиться от блокчейна и сохранить децентрализованность данных. Конкретная реализация протокола RaymonCoin не обсуждается.

В общем случае, криптовалюта работает следующим образом. Вместо привычного блокчейна, используется направленный ациклический граф (Directed Acyclic Graph, DAG). Транзакции, выпущенные нодами (пользователями сети), образуют множество вершин графа, которые используются для ведения истории транзакций.

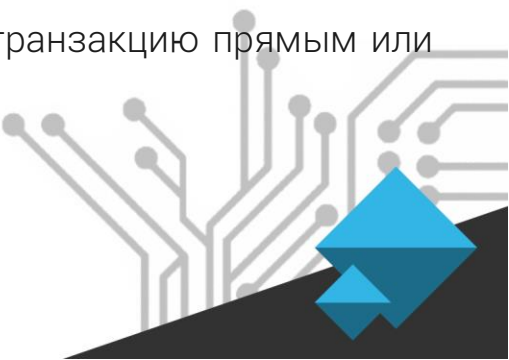



A decorative graphic in the top-left corner consisting of a network of grey lines and nodes, resembling a circuit board or a data network, with several blue triangles of varying sizes scattered around it.

Множество дуг графа описывается следующим образом: когда поступает новая транзакция, она должна сначала подтвердить или попытаться подтвердить (подробнее об этом позже) 2 предыдущие транзакции. Эти подтверждения могут быть представлены как ориентированные дуги графа (рис. 1). Если не существует ориентированной дуги между транзакцией А и транзакцией В, но существует ориентированный маршрут от А до В, длина которого больше либо равна 2, можно сказать, что А косвенно подтверждает В. Так же, существует начальная транзакция – “генезис”, которая в любом случае подтверждена всеми остальными транзакциями. Генезис можно описать следующим образом. В начале Хайв, существует адрес с балансом, который содержит в себе все токены. Генезис отправляет эти токены нескольким другим основополагающим адресам. Подчеркнем, что все токены созданы в транзакции генезис. В будущем больше не будет сгенерировано ни одного токена, а также не будет майнинга, в том смысле, что майнеры не будут получать денежное вознаграждение «из воздуха».

Сеть наполняют ноды. Ноды – это сущности, которые отправляют и подтверждают транзакции.

Основная идея Хайв следующая: для того, чтобы отправить свою транзакцию, пользователь должен выполнить работу по подтверждению двух других транзакций. Для ускорения этого процесса, в системе формируются группы нод (пулы), каждая из которых занимается подтверждением своих транзакций. Поэтому пользователи, совершающие транзакции, вносят вклад в безопасность сети. Предполагается, что ноды проверяют подтвержденные транзакции на наличие конфликтов. Если нода обнаруживает, что транзакция конфликтует с историей транзакций в Хайве, а также она имеет низкий рейтинг, нода не будет одобрять конфликтующую транзакцию прямым или косвенным образом.

A decorative graphic in the bottom-right corner, mirroring the one in the top-left, featuring a network of grey lines and nodes with blue triangles.



Поскольку в Raymon, а в частности в Хайв имеется система рейтингов, каждая транзакция может получать дополнительные подтверждения, она принимается системой с более высоким уровнем доверия. Другими словами, будет трудно заставить систему принять транзакцию с двойной тратой.

Для отправки транзакции, нода должна выполнить следующие действия:

1) Выбрать 2 другие транзакции для подтверждения, в соответствии с алгоритмом. В общем случае, эти транзакции могут совпадать.

2) Выявить рейтинг транзакции и проверить на наличие конфликтов если таковые у нее имеются.

3) Для того, чтобы узел выдал действительную транзакцию, он должен решить криптографическую головоломку, аналогичную той, что есть в блокчейне Bitcoin. Это достигается за счет того, что хеш некоторого значения попсо, объединенного с некоторыми данными из одобренной транзакции, имеет определенную форму. В случае протокола Bitcoin, хеш должен иметь как минимум predetermined число начальных нулей.

Важно отметить то, что данная система работает асинхронно. В общем случае, ноды видят не обязательно одинаковое множество транзакций. Также отметим, что Хайв может содержать конфликтные транзакции. Нодам не нужно достигать консенсуса относительно того, какие транзакции имеют право быть в Хайв. Тем не менее, в этом случае, когда существуют конфликтующие транзакции, ноды должны решить, какие будут отделены (orphaned). Главное правило, которого придерживаются ноды и по которому принимают решение о том, какую из двух транзакций выбрать, таково: нода (выполняет алгоритм «отбора HT» несколько раз, и смотрит, какая из двух транзакций наиболее подходит для того, чтобы стать непосредственно подтвержденной по выбранной HT.

Например, если транзакция была выбрана 97 раз из 100 итераций алгоритма, можно сказать, что она подтверждена с вероятностью 97%.

Прокомментируем так же следующий вопрос: что побуждает ноду проверять транзакции? Каждая нода считает различные статистики, одна из которых показывает, сколько новых транзакций было получено от соседа. Если конкретная нода "слишком ленив", она будет отклонен его соседями. Следовательно, даже если нода не отправляет транзакции, и, следовательно, не имеет прямого стимула для распространения новых транзакций, которые бы подтверждали его собственную транзакцию (повторения), у него все еще есть стимул, чтобы принимать в этом участие.

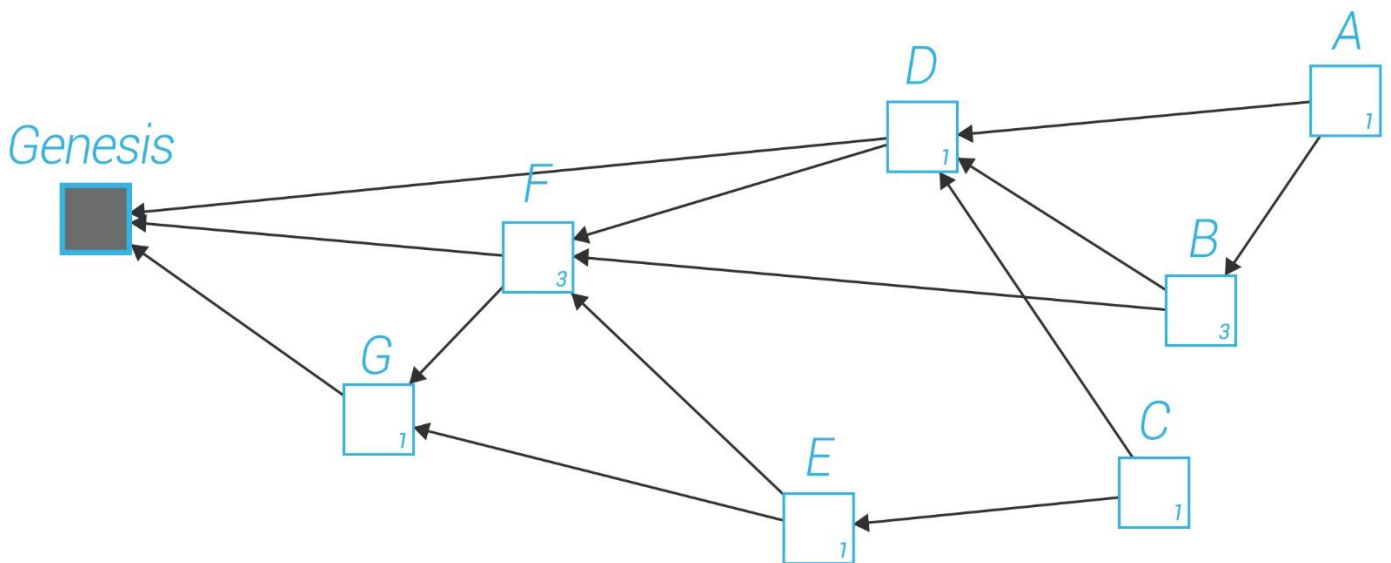


Рис. 1 - DAG

## ХРАНЕНИЕ ДАННЫХ

Данные в графе хранятся в формате трит, закодированные в строку в формате ASCII по определенному алгоритму.

В общей сложности существует 27 различных символов для кодировки трит, например, "q1w2e3r4t5y6u7i8o9p0azsxdcf". Из полученной строки берется значение символа по ASCII. Полученное десятичное значение переводится в триты делением на 27, первое значение - остаток от деления, второе значение - целая часть.

Допустим мы хотим преобразовать символ "P". "P" имеет десятичное значение ASCII - 80. 80 можно представить, как  $26 + 2 * 27$ . 26 - первое значение, 2 - второе значение. Взяв значения из алфавита по получившимся индексам получаем что первое значение - "f", второе - "w". Соответственно "P" преобразовалось в "fw".



Вес совершаемой нодой транзакции пропорционален количеству работы, которая была на нее затрачена. В текущей версии Хайв, вес может принимать значения  $3^n$ , где  $n$  – положительное целое число, находящееся в непустом конечном интервале допустимых значений. Не важно, каким образом значение веса было получено на практике. Идея заключается в том, что транзакция с большим весом «важнее», чем транзакция с меньшим весом.

Существует так же совокупный вес транзакции: он определяется как собственный вес некоторой транзакции, плюс сумма собственных весов всех транзакций, которые прямо или косвенно подтверждают эту транзакцию.

В Хайве существуют неподтвержденные транзакции (далее НТ).

Для транзакции в Хайве введем ее высоту – длину самого длинного ориентированного маршрута до генезиса и глубину – длину самого длинного обратного маршрута до некоторого НТ.

Также, введем понятие «очки». По определению, очки транзакции – это сумма собственных весов всех транзакций, подтвержденных этой же транзакцией, плюс собственный вес этой транзакции.

Далее будем полагать, что собственный вес любой транзакции равен 1.

Пусть  $L(t)$  – общее количество НТ в системе в момент времени  $t$ . Можно предположить, что стохастический процесс  $L(t)$  остается устойчивым. Точнее, можно ожидать, что процесс будет возвратным. В частности, возвратность подразумевает, что предел  $P[L(t) = k]$  при  $t \rightarrow \infty$  должен существовать и быть положительным для всех  $k \geq 1$ . Интуитивно, мы полагаем, что значение  $L(t)$  должно колебаться относительно постоянного значения, не уходя в бесконечность.

Чтобы проанализировать устойчивость  $L(t)$ , нам нужно сделать некоторые предположения. Первое предположение заключается в том, что транзакции приходят от большого числа независимых сущностей, поэтому процесс поступающих транзакций можно представить, как процесс Пуассона. Пусть  $\lambda$  – интенсивность потока. Для упрощения, предположим, что это значение не меняется с течением времени. Предположим, что все устройства имеют приблизительно одинаковую вычислительную мощность, тогда пусть  $h$  – среднее время вычислений, нужных для совершения транзакции. Тогда, предположим, что все ноды действуют следующим образом: для совершения транзакции, нода случайно выбирает 2 НТ и подтверждает их. Нужно отметить, что в общем случае, следовать этой стратегии – это не лучшая идея для “честных нод”, потому что она имеет ряд недостатков. В частности, она недостаточно защищает от «ленивых» или вредоносных нодов. С другой стороны, мы все еще рассматриваем эту модель, так как ее легко анализировать, и она может дать представление о поведении системы для более сложных стратегий выбора НТ.

Далее мы делаем еще одно упрощающее предположение, что любая нода в момент совершения транзакции наблюдает не фактическое состояние Хайва, а его состояние ровно  $h$  единиц времени назад.

Это означает, в частности, что транзакция, привязанная к Хайву в момент времени  $t$ , становится видимой в сети только в момент времени  $t + h$ . Мы также предполагаем, что количество НТ, грубо говоря, остается постоянным во времени и сосредоточено в окрестности числа  $L_0 > 0$ . В дальнейшем мы вычислим  $L_0$  как функцию от  $\lambda$  и  $h$ .

Обратите внимание, что в данный момент времени  $t$  мы имеем приблизительно  $\lambda h$  «скрытых типов» (которые были прикреплены на временном интервале  $[t - h; t)$  и поэтому еще не видны сети), также предположим, что обычно есть  $r$  «проявившихся НТ» (которые были добавлены до времени  $t - h$ ), поэтому  $L_0 = r + \lambda h$ . Ввиду устойчивости, мы можем предположить, что в момент времени  $t$  есть также около  $\lambda h$  вершин, которые были НТ в момент времени  $t - h$ , но больше не являются НТ. Теперь подумайте о новой транзакции, которая приходит в этот момент; тогда транзакция, которую она выбирает для одобрения, является НТ с вероятностью:

$$\frac{r}{r + \lambda h}$$

(поскольку ноде, которая произвела транзакцию, известно около  $r$  НТ, а также есть около  $\lambda h$  транзакций, которые больше не являются НТ, хотя эта нода считает, что они есть), поэтому среднее количество выбранных НТ равно:

$$\frac{2r}{r + \lambda h}$$

Главное наблюдение заключается в том, что в устойчивом состоянии, это среднее количество выбранных НТ должно быть равно 1, так как в среднем новая транзакция не должна менять количество НТ. Решая уравнение:

$$\frac{2r}{r + \lambda h} = 1$$

по  $r$ , получаем  $r = \lambda h$ , тем самым получаем

$$L_0 = 2\lambda h \quad (1)$$

Заметим так же, что для правила, где транзакция должна подтверждать  $k$  других транзакций, вместо 2, будет работать формула:

$$L_0(k) = \frac{k\lambda h}{k-1}(2)$$

Это, конечно, согласуется с тем, что  $L_0(k)$  должно стремиться к  $\lambda h$  при  $k \rightarrow \infty$  (в основном, единственными НТ будут те транзакции, которые до сих пор неизвестны сети). Также ожидаемое время для одобрения транзакции в первый раз составляет приблизительно  $h + \frac{L_0}{2\lambda} = 2h$ . Это связано с тем, что, по нашему предположению, в течение первых  $h$  единиц времени транзакция не может быть одобрена, и после этого процесс Пуассона подтверждений имеет интенсивность приблизительно  $\frac{2\lambda}{L_0}$ .

Заметим, что в любое фиксированное время  $t$  множество транзакций, которые были НТ в какой-то момент  $s \in [t; t + h(L_0, N)]$  обычно представляет собой разрез графа. Любой путь от транзакции до генезиса, отправленной в момент времени  $t' > t$ , должен пройти через этот набор. Важно, чтобы размер нового разреза в хайве время от времени становился маленьким. Затем можно использовать небольшие разрезы в качестве контрольных точек для возможной обрезки DAG и других задач.

Важно отметить, что вышеупомянутая «чисто случайная» стратегия утверждения на практике не очень хороша, потому что она не поощряет одобрение НТ. «Ленивый» пользователь может всегда одобрять фиксированную пару очень старых транзакций, поэтому не способствует утверждению более поздних транзакций, не подвергаясь наказанию за такое поведение. Также злонамеренный субъект может значительно увеличить количество НТ, выпустив много транзакций, которые утверждают фиксированную пару транзакций. Это позволит будущим транзакциям выбирать эти НТ с очень высокой вероятностью, эффективно отбрасывая НТ, принадлежащие «честным» нодам. Чтобы избежать подобных проблем, нужно принять стратегию, которая ориентирована на «лучшие» НТ.

Перед началом обсуждения ожидаемого времени, перед тем как транзакция получит свое первое одобрение, обратите внимание, что мы можем различать два режима работы сети:

Низкая нагрузка: типичное количество НТ невелико и часто становится 1. Это может произойти, когда поток транзакций настолько мал, что не представляется возможным, чтобы несколько разных транзакций одобряли один и тот же НТ. Кроме того, если задержка в сети очень низкая, а производительность устройств высокая, маловероятно, что появится много НТ. Это справедливо даже в случае, когда поток транзакций достаточно велик. Более того, мы должны предположить, что никто не пытается атаковать сеть, пытаясь искусственно раздуть количество НТ.

Высокая нагрузка: типичное количество НТ велико. Это может произойти, когда поток транзакций велик, а задержки вычислений вместе с задержкой в сети делают возможным, что несколько различных транзакций одобряют один и тот же НТ.

Это разделение довольно неформальное, и между этими двумя режимами нет четкой границы. Тем не менее, мы можем рассмотреть эти две разные крайности, чтобы лучше понять работу сети.

Ситуация в режиме низкой нагрузки относительно проста. Первое подтверждение происходит в среднем за  $\lambda^{-1}$ , так как одна из первых входящих транзакций одобрит данный совет.

Рассмотрим теперь режим высокой нагрузки, где значение  $L_0$  велико. Как уже упоминалось выше, можно предположить, что потоки Пуассона подтверждений для различных НТ независимы и имеют интенсивность приблизительно  $\frac{2\lambda}{L_0}$ . Поэтому ожидаемое время получения первого подтверждения для транзакции, составляет около  $L_0/(2\lambda) \approx 1.45h(1)$ .

Тем не менее, стоит отметить, что для более сложных стратегий одобрения, может быть, не стоит пассивно ждать долгое время, пока транзакция не будет одобрена другими. Это связано с тем, что «лучшие» НТ будут появляться и будут предпочтительны для подтверждений.

Скорее, в случае, когда транзакция ожидает утверждения в течение временного интервала, намного большего, чем  $L_0/(2\lambda)$ , хорошей стратегией было бы продвижение для этой транзакции с помощью дополнительной пустой транзакции. Другими словами, нода может отправить пустую транзакцию, которая подтвердит свою предыдущую вместе с одним из лучших НТ, чтобы увеличить вероятность получения подтверждения пустой транзакции.

## ПУЛЫ

Для ускорения процесса подтверждения транзакций, в системе есть объединения нодов, которые занимаются совместным подтверждением. Работает это следующим образом: при подключении ноды к системе, сеть сообщает ей, к какой группе она принадлежит. После этого происходит синхронизация по времени всех участников пула. Через определенные промежутки времени, сеть выбирает наиболее важную транзакцию и сообщает ее пулу, после этого каждый член пула начинает ее подтверждение. Поясним, как работает подтверждение в случае с пулами: пусть  $n$  – количество участников пула,  $d$  – среднее количество итераций, требуемых для нахождения попсе,  $i$  – индекс участника пула. Тогда,  $r = [(d/n)i; (d/n)i + d)$  диапазон значений, который требуется перебрать  $i$ -му участнику пула.

Это принцип, основанный на принципе Proof-Of-Work и системы рейтингов. Данный принцип защиты предполагает, что пользователь, который хочет взаимодействовать с системой, должен сперва подтвердить себя.

При добавлении новой транзакции в граф она подтверждает две прошлые. Транзакции для подтверждения подбираются по определенному алгоритму, который проверяет, не противоречат ли эти транзакции и не разрешают ли они конфликтующие транзакции. Для реализации подтверждения используется доказательство выполнения работы аналогичное HashCash Адама Бэка. Работа по доказательству достоверности транзакций включает в себя сканирование на значение, которое при хешировании с помощью алгоритма digest384 начинается с определенного числа нулевых трит.

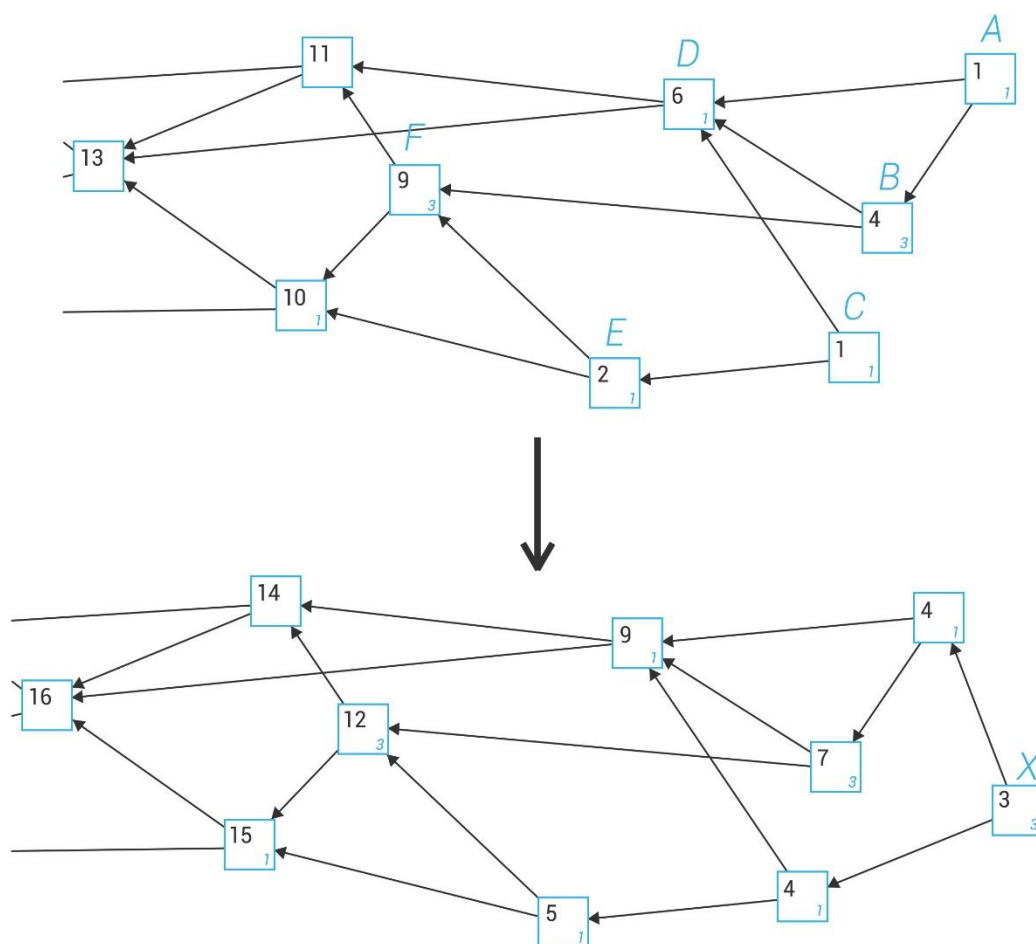


Рис 2. - Пример добавления новой транзакции



Алгоритм подсчета суммарного веса можно увидеть на рисунке. Каждый узел (квадрат) - это транзакция, число внизу - это собственный вес транзакции, число, выделенное жирно - совокупный вес. На рисунке 2 транзакция F прямо или косвенно подтверждается транзакциями A, B, C, E. Суммарная масса F равна 9 (1+3+1+1+3). Транзакции A, C являются концами графа. X, на втором рисунке их подтверждает и становится концом графа, увеличивая совокупный вес всех транзакций, которые она косвенно подтверждает, на 3.

Нахождение хеша и подсчет весов происходит на том устройстве, с которого была передана транзакция, тем самым само это устройство является майнером.

Для того, чтобы у пользователей был стимул в активном использовании Хайва, существует система рейтингов. Посчитав текущий рейтинг ноды, можно выявить ее активность, и в зависимости от этого, отдавать большее, либо меньшее предпочтение в подтверждении транзакций этой ноды. Работает это так: нода совершает определенную работу при совершении или подтверждении транзакции. Можно сказать, что эта работа записывается в Хайв в виде транзакций. Учитывая временные отметки каждой транзакции, можно создать алгоритм, который будет вычислять текущий рейтинг ноды. Приведем пример работы одного из таких алгоритмов. Возьмем множество всех транзакций ноды за определенный промежуток времени. Будем считать, что за одну транзакцию рейтинг ноды увеличивается на  $C$ , тогда как за сутки, рейтинг уменьшается на  $F$ . Отметим, что рейтинг не может быть отрицательным. Пусть  $C$  - количество начисляемого рейтинга за произведенную транзакцию,  $D$  - упорядоченное множество таких элементов  $t - t_c$ , включая элемент 0, где  $t$  - временная метка транзакции (в днях),  $t_c$  - текущая временная метка (в днях). Тогда рейтинг ноды в текущий момент времени.

$$R = \sum_{i=1}^{n-1} \max\{R + C + F(D_i - D_{i+1}), 0\}, \text{ где } n = |D|.$$

“Атомарный своп” (atomic swap) - обмен цифровыми валютами между двумя участниками напрямую без участия третьей стороны.

Допустим, Алиса и Боб хотят провести обмен одной криптовалюты на другую. Алиса переводит свои средства в своеобразное хранилище, в которой средства для обмена будут храниться до конца сделки. Для изъятия средств из этой ячейки потребуется секретный ключ и подпись Боба.

Алиса генерирует секретный ключ и его хеш. Затем Боб запрашивает у Алисы этот секретный ключ и создает аналогичную ячейку для хранения своих средств с таким же ключом. Заметим, что, как и в случае с ячейкой Алисы, Боб не сможет открыть свою ячейку без подписи Алисы. При этом, на данном этапе Алиса уже имеет возможность открыть ячейку Боба, подписав ее, и получить средства на свой счет. Когда Алиса получила средства, Боб получает ее подпись, с помощью которой он может открыть вторую ячейку и завершить обмен.

В случае, если один из участников прекращает сделку на полпути, атомарный своп отклоняет сделку и возвращает средства назад обоим участникам.

Смарт-контракт (англ. Smart contract – “умный контракт”) как раз требуется для совершения подобных сделок. Смарт-контракты хранятся в Хайв по тому же принципу, что и транзакции, и по-сути представляют собой байт-код, запускаемый на виртуальной машине Paymon (PVM).

PVM - простая виртуальная машина (VM), с архитектурой на основе стека. Размер слова машины (и, соответственно, размер элемента стека) составляет 256 бит. Это было выбрано для облегчения схемы хеша Кессак256 и вычислений на основе эллиптических кривых. Модель памяти представляет собой простой массив байтов с адресацией по словам.

Максимальный размер равен 1024. У VM также есть независимая модель хранилища данных; это похоже на концепцию памяти, а не на массив байтов, это массив слов с адресацией по словам. В отличие от памяти, которая является изменяемой, хранилище изменчиво и поддерживается как часть состояния системы. Все значения как в памяти, так и в хранилище изначально четко определены как ноль.

VM не соответствует стандартной архитектуре фон Неймана. Вместо того, чтобы хранить программный код в общедоступной памяти или хранилище, он хранится отдельно в виртуальном ПЗУ, взаимодействующем только через специализированную инструкцию.

VM может вызывать исключения в нескольких случаях, включая случаи выхода за нижнюю границу стека и выполнение недопустимых инструкций. При возникновении исключения, VM, скорее всего, немедленно остановится и сообщит о проблеме исполнителю (либо процессору транзакции, либо, рекурсивно, среде выполнения), которая будет обрабатывать их отдельно.

В дополнение к состоянию системы  $\sigma$  и оставшимся тактам для вычисления  $g$  в среде выполнения есть важная информация, которую должен предоставить агент исполнения; они содержатся в кортеже  $l$ :

- $I_a$  - адрес учетной записи, которой принадлежит исполняемый код.
- $I_o$  - адрес отправителя транзакции, инициировавшей это выполнение.
- $I_d$  - массив байтов, который является входными данными для этого выполнения; если исполняющим агентом является транзакция, это будут данные транзакции.

- $I_s$  - адрес учетной записи, вызывающей выполнение кода; если агент исполнения является транзакцией, это будет отправитель транзакции.

- $I_v$  - значение в PMNC, переданное этому аккаунту как часть той же процедуры, что и исполнение; если исполнителем является транзакция, это будет значение транзакции.

- $I_b$  – массив байтов, являющийся машинным кодом, который должен быть выполнен.

Модель выполнения определяет функцию  $P$ , которая может вычислять результирующее состояние  $\sigma'$ , оставшиеся такты  $t'$ , список самоубийств  $s$ , логарифмический ряд  $l$ , возврат  $r$  и результат  $o$ , учитывая эти определения:  
 $(\sigma', t', s, l, r, o) \equiv P(\sigma, g, l)$ .

Теперь мы должны определить функцию  $P$ . В большинстве практических реализаций это будет моделироваться как итеративная прогрессия пары, включающая полное системное состояние,  $\sigma$  и состояние ВМ,  $\mu$ . Формально мы определяем ее рекурсивно с помощью функции  $X$ . Эта функция использует функцию итератора  $O$  (которая определяет результат одного цикла конечного автомата) вместе с функциями  $Z$ , которые определяют, является ли текущее состояние исключительным условием остановки машины и  $H$ , задавая выходные данные инструкции тогда и только тогда, когда текущее состояние является нормальным условием остановки ВМ.

Пустая последовательность, обозначенная  $()$ , не равна пустому множеству, обозначенному  $\emptyset$ ; это важно при интерпретации вывода  $H$ , который оценивает  $\emptyset$ , когда выполнение должно продолжаться, но серия (потенциально пустая), когда выполнение должно прекратиться.

$$P(\sigma, t, I) \equiv X_{0,1,2,4}((\sigma, \mu, A^0, I))$$

$$\begin{aligned} \mu_g &\equiv t \\ \mu_{pc} &\equiv 0 \\ \mu_m &\equiv (0, 0, \dots) \\ \mu_i &\equiv 0 \\ \mu_s &\equiv () \end{aligned}$$

$$X(\sigma, \mu, A, I) \equiv \begin{cases} (\emptyset, \mu, A^0, I, ()), & \text{если } Z(\sigma, \mu, I) \\ O(\sigma, \mu, A, I) \cdot o, & \text{если } o \neq \emptyset \\ X(O(\sigma, \mu, A, I)), & \text{в противном случае} \end{cases}$$

где

$$\begin{aligned} o &\equiv H(\mu, I) \\ (a, b, c) \cdot d &\equiv (a, b, c, d) \end{aligned}$$

Обратите внимание, что мы должны отбросить четвертое значение в кортеже, возвращенном  $X$ , чтобы правильно оценить  $P$ , то есть,  $X_{0,1,2,4}$ .

$X$ , таким образом, зацикливается (имеется ввиду рекурсивно, но, как правило, предполагается, что реализации обычно используют простой итеративный цикл), пока не станет истинным, что указывает на то, что текущее состояние является исключительным и машина должна быть остановлена, при этом любые изменения отбрасываются; либо пока  $H$  не станет серией (а не пустым множеством), что указывает на то, что машина достигла контролируемой остановки.

Система имеет бизнес-платформу, благодаря которой можно легко подключить свой собственный магазин, сервис и т.п и принимать платежи за товары или услуги в криптовалюте. На сайте имеется удобный конструктор, который после добавления товаров/услуг генерирует смарт-контракты. Эти смарт-контракты обеспечивают работу платформы.

К примеру, когда пользователь выбирает товар или услугу и нажимает купить, у него появляется выбор, показать QR-код и оплатить товар на месте, или же выбрать доставку. Во втором случае смарт-контракт будет ждать, пока компания подтвердит покупку, проверит достаточно ли у пользователя средств, и произведет обмен.

Во избежание конфликтов между клиентом и компанией, существует рейтинг. Каждый пользователь, воспользовавшись услугами компании, может поставить оценку от 1 до 5. Также всегда можно открыть диспут с данной компанией и обсудить интересующий вопрос. Также в дальнейшем, за счёт созданных смарт контрактов внутри Nive и Profit, все наши пользователи смогут проводить собственные ICO и устраивать коллективные сборы под любые нужды. Ведь именно в Paymon отсутствуют комиссии, а транзакции имеют самый легкий вес по сравнению с конкурентами.

Рассмотрим подробнее вопрос о том, зачем нужен кешбек и откуда он берется. Кешбек - это своего рода награда за пользование услугами компании, поэтому, браться он будет со счета этой компании. В зависимости от рейтинга, количество выплаченных токенов (кешбек) будет равно.

$$M = \begin{cases} A * 0.0005, \text{ при } R > 10 \wedge R < 25 \\ A * 0.0010, \text{ при } R \geq 25 \wedge R < 50 \\ A * 0.0015, \text{ при } R \geq 50 \wedge R < 75 \\ A * 0.0020, \text{ в остальных случаях} \end{cases}$$