Разработка смарт-контрактов в Ethereum

Тимур Машнин

18+

Тимур Машнин

Разработка смарт-контрактов в Ethereum

http://www.litres.ru/pages/biblio_book/?art=67111800
SelfPub; 2022

Аннотация

Эта книга рассказывает о принципах работы Ethereum, отличии Ethereum от Bitcoin. Вы узнаете что такое децентрализованные приложения Dapp и смарт-контракты, познакомитесь с инструментами разработки Dapp. Изучите высокоуровневый язык Solidity создания смарт-контрактов для виртуальной машины Ethereum. Познакомитесь со средой разработки Remix. Узнаете о практическом применении смарт-контрактов, стандартах ERC20, ERC-721, ERC-1155 и EIP-3156.

Содержание

Тимур Машнин
Разработка смарт-контрактов в Ethereum

Ethereum. Причины возникновения Ethereum

Когда в 2008 году появился Биткойн, он был полностью революционным.

Количество концепций, которые собрались при этом вместе, информатика, криптография и экономические стимулы, было удивительным.

Когда в 2009 году была запущена реальная сеть Bitcoin, многие думали, что этот проект потерпит неудачу.

Чтобы гарантировать работу сети, язык скриптов биткойна был преднамеренно предельно ограничительным.

На самом деле у Биткойна не совсем язык скриптов, он использует стек с операторами скриптов.

Язык скриптов в биткойне важен, потому что это то, что делает биткойн «программируемыми деньгами».

В рамках каждой транзакции биткойн есть возможность написать небольшую программу.

В результате вы можете автоматически переводить деньги с помощью компьютерного кода, и каждый может видеть правила, по которым эти деньги перемещаются, и знать, что эти правила будут соблюдаться.

До сих пор сеть Биткойна успешно развивается, и ее капитализация растет.

Однако все приложения, которые создаются в Биткойне – это инфраструктурные приложения, такие как кошельки и биржи.

Причина этого в ограниченности языка скриптов Биткойна.

Проект Ethereum взял этот ограниченный набор операторов и развил его в полноценный язык программирования.

Эфириум никогда бы не существовал без Биткойна в качестве предтечи.

При этом Ethereum во многом опережает Биткойн.

	bitcoin	ethereum
concept	digital money	smart contracts
transaction	send from alice to bob	send from alice to bob if.. • date = jan 1, 2018 • bob's balance < 10 eth
market cap (as of feb 2017)	~$18 billion	~$1 billion
founder	satoshi nakamoto (unknown)	vitalik buterin and team
release date	jan 2009	july 2015
release method	early mining	presale raised $18M in bitcoin

Язык программирования Ethereum позволяет делать гораздо больше, чем биткойн.

Язык скриптов Bitcoin намеренно ограничивает. В результате, в биткойн вы можете делать только основные вещи.

Этот язык также трудно понять и использовать.

Вместо большинства современных языков программирования, где код легко читается, скрипты биткойна выглядят как непонятный машинный код.

Напротив, языки программирования Ethereum (Solidity для тех, кто любит Javascript, и Serpent для тех, кто любит Python), позволяют делать практически все, что позволяет вам продвинутый язык программирования.

При этом, эти языки простые в использовании.

Эта комбинация полной функциональности программи-

рования и простоты использования очень важна.

Люди делают вещи в Эфириуме, которые сейчас невозможны в Биткойне.

Эфириум создал новое поколение разработчиков, которые никогда не работали с Bitcoin, но заинтересованы в Ethereum.

Биткойн мог бы иметь эту расширенную функциональность, но через создание многочисленных надстроек, которые бы работали с протоколом биткойнов, в то время как Ethereum предоставляет эту функциональность из коробки.

Помимо радикального различия в языках скриптов, в Ethereum намного лучше инструменты разработчика.

У Биткойна никогда не было полноценного набора инструментов для разработчиков, хотя он очень нужен, потому что работать с Bitcoin намного сложнее.

Эфиреум сделал жизнь разработчика намного проще.

Он имеет домашнюю страницу для разработчиков и собственную среду разработки (Mix IDE) среди прочих других.

Есть и другие преимущества Эфиреума, о которых достаточно написано.

Являются ли Bitcoin и Ethereum конкурентами или дополняют друг друга?

Это еще предстоит выяснить.

Возможно, биткойн останется протоколом, с помощью которого люди будут хранить свои деньги, потому что он более стабилен и надежен.

Это позволило бы Ethereum продолжать рисковать, позволяя разворачивать приложения в своей сети.

В этом случае Bitcoin будет являться скорее сетью денежных расчетов, в то время как Ethereum будет использоваться для запуска децентрализованных приложений.

Так что Bitcoin и Ethereum могут дополнять друг друга.

Что такое Эфириум?

Ethereum - это децентрализованная платформа, на которой работают смарт контракты

Согласно веб-сайту Ethereum, «Ethereum – это децентрализованная платформа, на которой работают умные контракты».

Биткойн же можно охарактеризовать как цифровые деньги.

Биткойн используется для перевода денег от одного чело-

века другому.

И он обычно используется в качестве хранилища денег и является основой для понимания обществом концепции децентрализованной цифровой валюты.

Ethereum отличается от Bitcoin тем, что он позволяет использовать смарт-контракты, которые можно охарактеризовать как высоко программируемые цифровые деньги.

Представьте, что вы автоматически отправляете деньги от одного человека другому, но только тогда, когда выполняется определенный набор условий.

Например, человек хочет купить дом у другого человека.

Традиционно в обмене участвуют несколько третьих сторон, в том числе юристы и агенты условного депонирования, что делает процесс излишне медленным и дорогостоящим.

С помощью Ethereum код может автоматически передать покупателю собственность и средства продавцу после того, как сделка будет одобрена без необходимости участия третьей стороны.

Потенциал для этого невероятный.

Подумайте о многочисленных приложениях, которые выступают в качестве третьей стороны, чтобы связать вас с другими на основе определенной логики (например, Uber и eBay).

Многие из централизованных систем, которые мы используем сегодня, можно было бы децентрализовать на Эфириуме.

Децентрализация важна, так как устраняет точки отказа или контроля.

Децентрализованные платформы удаляют посредников, что в конечном итоге приводит к снижению затрат для пользователя.

Bitcoin vs Ethereum

Давайте еще раз обсудим, чем отличается Эфериум от Биткойна.

	Bitcoin	Ethereum
Launched	2009	2015
Creator	"Satoshi Nakamoto"	Vitalik Buterin
Consensus algorithm (current)	Proof of Work	Proof of Work
Average block time	10 minutes	17.5 seconds
Function	Digital currency	Smart contract platform
Flexibility	Constrained (deliberately)	More
Transactions (example)	Simple value transfer	Conditional value transfer
Turing completeness (beyond the scope of this post)	No	Yes

Если кратко, Биткойн – это первый известный блокчейн.

А Ethereum – это блокчейн следующего поколения.

Биткойн был первоначально создан как одноранговая электронная платежная система, где валюта торгуется между адресами.

Ethereum был создан с идеей быть не только одноранговой денежной системой, а представлять одноранговую базу данных и распределенный виртуальный компьютер, компьютер, который манипулирует тем, что называется состоянием.

Вы можете думать о состоянии как о текущих значениях всех переменных в системе, согласованных всеми узлами посредством консенсуса.

Состояние в Ethereum изменяется в результате транзакций и работы виртуальной машины Ethereum.

В то время как у Биткойна есть упрощенный язык скриптов, одной из наиболее примечательных особенностей Ethereum является виртуальная машина, которая способна выполнять полноценный код.

Весь код, который запускается виртуальной машиной, может быть сохранен как часть цепочки блоков.

И вы можете программировать довольно сложные приложения, чья логика будет работать на цепочке блоков.

Из-за этого существует огромное сообщество разработчиков, растущее вокруг Ethereum, которые создают множество приложений и развивают экосистему Ethereum.

У системы Ethereum, как и у Биткойна, есть проблемы с масштабированием.

Например, вся система в настоящее время может обрабатывать только 17 транзакций в секунду.

Потому что вся сеть должна запускать каждое вычисление на каждом компьютере, а это значит, что вся система может

работать настолько быстро, насколько быстро в ней работают самые медленные компьютеры.

Доказательство работы неэффективно, и сам блокчейн продолжает расти.

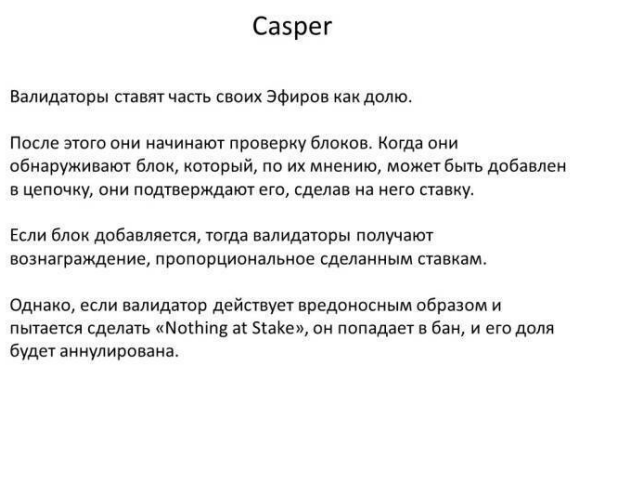
Сообщество пытается решить эти проблемы масштабирования с помощью нескольких подходов: доказательстве ставки, каналов состояний, Sharding и Plasma.

Переход на доказательство ставки вместо доказательства работы уменьшит значительную вычислительную нагрузку на сеть.

Вместо того, чтобы добывать блоки, ища допустимо низкое значение хэша, сеть будет создавать блоки путем распределения разрешения на их создание пропорционально доле валюты, предоставленной узлом в качестве ставки, а не исходя из количества вычислительной мощности, которое узел может иметь.

Эфериум предлагает свой вариант доказательства ставки по названием Casper.

Casper реализует процесс, с помощью которого можно наказывать вредоносные узлы в сети.



Последняя версия протокола представляет собой гибрид алгоритмов доказательства работы и доказательства ставки.

Для создания блоков по-прежнему используется работа майнеров, а для фиксации контрольных точек блокчейна применяется «доказательство ставки».

Доказательство ставки создаёт дополнительную прослойку безопасности поверх результатов доказательства работы.

Для этого участники доказательства ставки или валидаторы отправляют личные монеты в пул валидаторов.

И блок будет считаться найденным только в том случае, если предложение поддержат две трети участников.

Если голоса не набираются, цепь продолжает работу на усилиях майнеров.

Каналы состояний обеспечивают своего рода кластериза-

цию транзакций в сети.

Это уменьшит количество отдельных транзакций, которые сеть должна будет обрабатывать сама.

Следующий подход для решения проблемы масштабирования, это Sharding – это идея разделения сети на более мелкие части, которые работают независимо.

И наконец, Plasma – это серия смарт контрактов, запущенная на вершине основного блокчейна.

В Plasma все данные обрабатываются в дочерних блокчейнах, и только результаты отправляются в основной блокчейн.

Если сравнивать Биткойн и Эфериум,

В то время как биткойн создавался как альтернатива традиционным деньгам и, таким образом, является средством оплаты и сохранения стоимости, Ethereum разрабатывался как платформа для выполнения одноранговых контрактов и приложений.

Bitcoin и Ethereum - это разные реализации технологии blockchain с разными целями.

Биткойн был открытием, окном в технологию blockchain, за что мы должны поблагодарить и Сатоши Накамото, псев-

донима, который создал протокол биткойна.

Примерно через девять лет академических исследований консенсусных алгоритмов, одноранговой сети, криптографических токенов и, что наиболее важно, виртуальной машины, Ethereum захотел взять тот же принцип одноранговой связи и применить его к любому типу программного приложения.

Таким образом, в основе как Ethereum, так и Bitcoin, лежат одни и те же принципы блокчейна, одноранговой сетевой инфраструктуры и архитектуры, криптографии и консенсуса.

Основное различие между Bitcoin и Ethereum заключается в наличии виртуальной машины Эфериума, которая по сути является мировым компьютером и для которой мы можем программировать приложения.

Таким образом, хотя Биткойн и Эфириум работают на принципе распределенного реестра и криптографии, у них есть множество технологических отличий.

Например, язык программирования, используемый Ethereum, является полным, в то время как у биткойна язык программирования стековый.

Другие отличия состоят во времени создания блока и подтверждения транзакции, транзакция Ethereum подтверждается в течении секунд по сравнению с минутами биткойна.

Кроме того, Ethereum использует алгоритм хэширования Ethash, в то время как Bitcoin использует алгоритм хэширо-

вания, SHA-256.

Также Ethereum использует доказательство работы, устойчивое к ASIC.

В алгоритме майнинга Эфериума, майнеры должны извлекать случайные данные из состояния, вычислять некоторые случайно выбранные транзакции из последних N блоков в цепочке блоков и возвращать хэш результата.

Это имеет два важных преимущества.

Во-первых, контракты Ethereum могут включать в себя любые вычисления, поэтому ASIC для Ethereum должен быть, по существу, ASIC для общих вычислений, т. е. процессором.

Во-вторых, для добычи требуется доступ ко всей цепочке блоков, заставляя майнеров хранить весь блок-цепочку.

Таким образом, применение ASIC для Эфериума, по сути, бесполезно, потому что это просто общие вычисления.

И, в довершение всего, в блокчейн могут быть записаны контракты, которые специально предназначены для усложнения работы ASIC.

Еще одна интересная вещь, заключается в том, что так как майнеры работают с состоянием, память должна быть быстрой.

И стандартная DRAM на самом деле не справляется с этой задачей, поэтому для майнинга используются RAM графических процессоров.

Подытоживая, с общей точки зрения, у Биткойн и Эфи-

риум разные предназначения.

В то время как биткойн создавался как альтернатива традиционным деньгам и, таким образом, является средством оплаты и сохранения стоимости, Ethereum разрабатывался как платформа для выполнения одноранговых контрактов и приложений.

И основная цель эфира – не быть альтернативой традиционным деньгам, в отличие от биткойнов, а облегчать и монетизировать работу сети Ethereum.

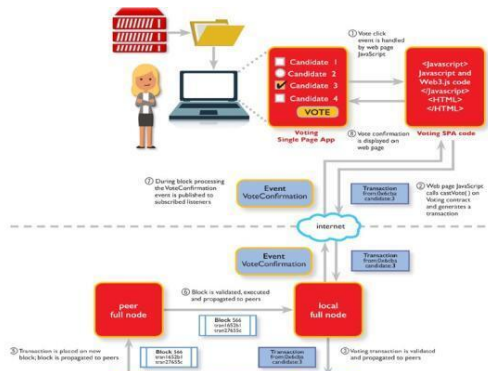
В целом, Bitcoin и Ethereum – это разные реализации технологии blockchain с разными целями.

Масштабирование

Давайте сравним децентрализованные системы блокчейна с централизованными системами.

Блокчейн взял на себя роль посредника в виде проверки, подтверждения и регистрации транзакций и, конечно же, выполнения консенсусного процесса для обеспечения целостности цепочки.

Все эти функции требуют времени и приводят к значительным накладным расходам по сравнению с централизованной системой.



Транзакции в Ethereum, например, обрабатываются на всех узлах и обрабатываются последовательно, а не параллельно.

В соответствии с порядком, определенным майнером.

Майнеры предпочитают сначала обрабатывать транзакции с более высокой комиссией.

После того, как блок добыт, этот порядок закреплен навечно. Этот порядок называется индексом транзакции.

И каждый полный узел хранит всю цепочку блоков.

Все это препятствует масштабируемости приложений blockchain.

Скорость обработки транзакций в блокчейне не является удовлетворительной по сравнению с централизованной обработкой.

Что такое масштабируемость?

Масштабируемость – это способность системы удовлетворительно работать на практических всех уровнях нагрузки.

Нагрузка в контексте блокчейна может быть транзакциями, количеством узлов, количеством участников и количеством аккаунтов и других атрибутов blockchain.

В случае блокчейна наиболее важной характеристикой является скорость транзакций или количество транзакций в секунду.

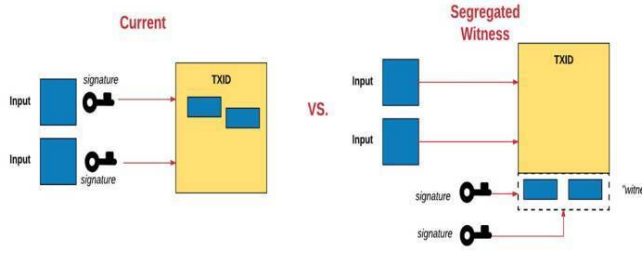
Это критическая характеристика для многих приложений, начиная от платежной системы и до управления цепочками поставок, чтобы система могла хорошо работать.

Например, для биткойна, в 2017 году, средняя комиссия сети достигла 15–20 долларов США, а скорость подтверждения транзакции стала доходить до нескольких дней.

Таким образом, можно сказать, что количество транзакций в секунду является показателем масштабируемости блокчейна.

Наиболее очевидным способом увеличения скорости транзакций является увеличение количества транзакций на блок.

Биткойн пытался сделать это двумя способами: с помощью вилки Segregated Witness и с помощью увеличения максимально разрешенного размера блока.



В Segregated Witness транзакции и подписи разделяются для обеспечения большего количества транзакций на блок.

Цифровая подпись занимает до 60 % пространства в транзакции.

И SegWit – это процесс, с помощью которого размер блока увеличивается путем удаления данных подписи из транзакций. Когда определенные части транзакции удаляются, это освобождает пространство для добавления дополнительных транзакций в блок.

Таким образом, подписи и скрипты выделяются в обособленную структуру, именуемую отдельным свидетелем.

И теперь, чтобы проверить все транзакции, узлу нужно загружать расширенный блок – это основной блок + отдельный свидетель.

Это была мягкая вилка, которая была реализована в 2017 году.

Она работает в текущей версии блокчейна биткойна.

Однако ограничение размера блока все равно фиксируется на одном мегабайте.

Второе предложение по увеличению масштабируемости состояло в том, чтобы помимо отдельного свидетеля, еще увеличить предел размера блока до большего размера в два мегабайта.

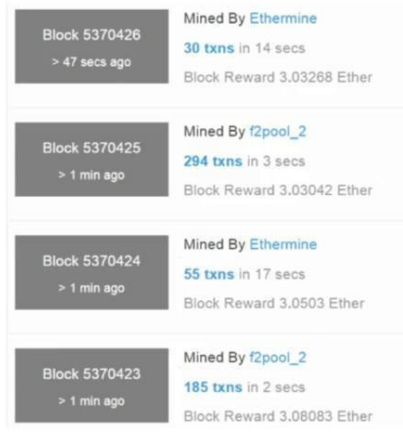
И это улучшение называется Segregated Witness 2X.

Это была запланированная жесткая вилка, которая должна была состояться в ноябре 2017 года, но не прошла из-за отсутствия поддержки сообщества.

Однако при этом часть сообщества, не приняв SegWit2x, произвела жесткий форк сети биткойна, увеличив размер блока биткойна до 8 Мб.

И свою ветку они назвали Bitcoin Cash.

15 мая 2018 года был обновлён протокол сети Bitcoin Cash с целью увеличения размера блока до 32 МБ.



Теперь давайте рассмотрим, как Ethereum относится к размеру блока.

В Ethereum размер блока может меняться и ограничен лимитом газа, указанным в заголовке блока.

Лимит газа на блок – это максимально допустимое количество газа в блоке для определения того, как много транзакций может поместиться в блок.

Например, у нас есть 5 транзакций, и каждая транзакция имеет лимит газа в 10, 20, 30, 40 и 50. Если лимит газа на блок – 100, тогда первые 4 транзакции могут поместиться в блок.

И майнеры решают какие транзакции поместить в блок.

Если попытаться включить транзакцию, которая использует больше газа, чем текущий лимит газа на блок – она бу-

дет отвергнута сетью и ваш Эфириум клиент выдаст вам сообщение “Транзакция превышает лимит газа на блок”.

Протокол позволяет майнеру блока регулировать лимит газа на блок на 0.0976 % в любом направлении.

Кто в Ethereum решает какой будет лимит газа на блок? – Майнеры. Отдельно от регулируемого протокола, стратегия с лимитом газа около 5,000,000 стоит по умолчанию в большинстве клиентов.

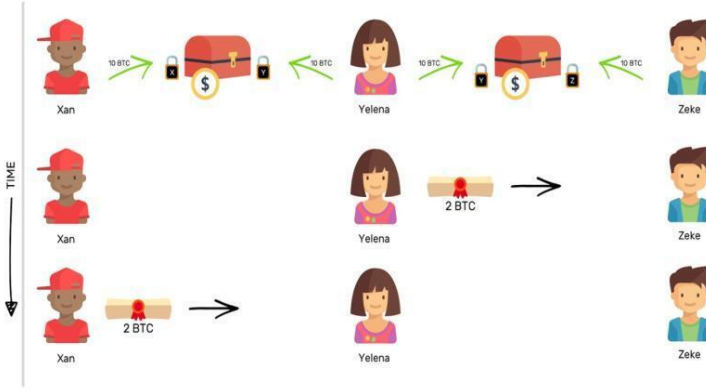
Майнеры могут изменить это, но многие этого не делают и оставляют настройки по умолчанию.

Этот размер переменного блока улучшил транзакционную скорость.

Ожидается, что он еще больше улучшится, когда Ethereum перейдет к доказательству ставки.

Мы только что обсудили сетевое решение для повышения скорости транзакций.

Теперь давайте обсудим механизм проведения транзакций вне сети.



Одним из решений для увеличения скорости транзакций является выгрузка некоторых транзакций из сети.

Это осуществляется между доверенными сторонами.

И эта функция в биткойне называется платежным каналом.

Платежные операции могут осуществляться с минимальными комиссиями со значительно более высокой скоростью между доверенными лицами.

В типичном платежном канале в цепочку блоков добавляются только две транзакции, но между участниками может быть сделано неограниченное или почти неограниченное количество платежей.

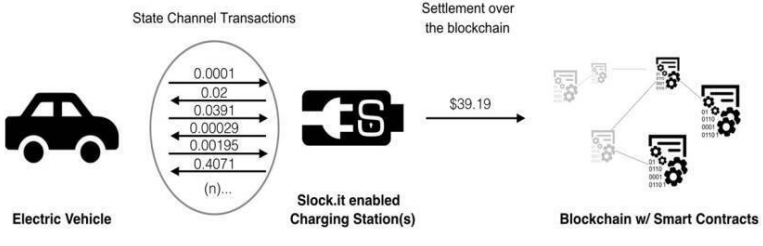
И примером такого решения служит сеть Lightning Network.

Сеть Lightning Network состоит из узлов и двунаправленных платежных каналов.

Платежный канал устанавливается между двумя узлами сети.

Каждый из двух узлов платежного канала блокирует в блокчейне Биткоина некоторую сумму средств для канала.

В дальнейшем пропускная способность канала будет определяться суммой заблокированных узлами средств.



Решение Ethereum расширяет эту концепцию платежного канала до смарт-контрактов.

Каналы состояния – это расширение концепции платежного канала для любой транзакции на уровне приложения.

Механизм называется State Channel, поскольку состояние

узла в основной цепочке блокируется при транзакции вне сети и периодически синхронизируется с соответствующими обновлениями с основной цепочкой.

Каналы состояния работают путем «блокировки» некоторой части состояния блокчейна в многосегментный контракт, контролируемый определенным набором участников.

Состояние, которое «блокировано», называется депозитом состояния.

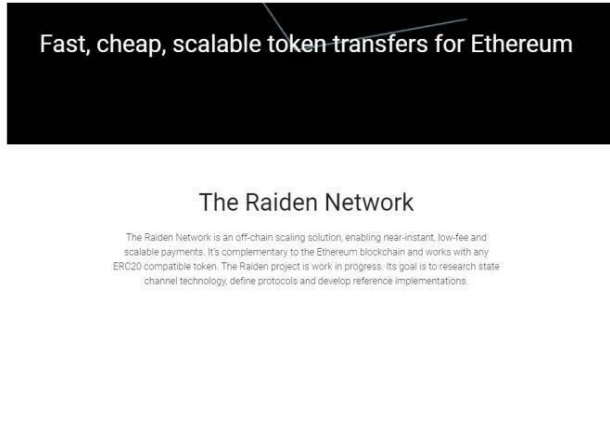
Это может быть определенное количество эфира или токенов.

После того, как депозит заблокирован, участники канала используют обмен сообщениями вне сети, чтобы обменивать и подписывать действительные транзакции ethereum, не развертывая их в цепочку. Это транзакции, которые могут быть добавлены в цепочку в любое время.

Обновление состояния канала всегда осуществляется единогласным решением сторон.

Все стороны подписывают и сохраняют свои собственные копии каждой транзакции вне сети.

Поскольку эти транзакции происходят полностью вне блокчейна, они имеют нулевые транзакционные сборы, и их скорость ограничена только их базовым протоколом связи.



Примером реализации такого протокола служит сеть Raiden Network, работающая поверх Эфериума.

Еще один подход к увеличению скорости обработки транзакций называется Sharding.

В настоящее время каждый отдельный узел, работающий в сети Ethereum, должен обрабатывать каждую транзакцию, проходящую через сеть.

Это дает блокчейну высокую степень безопасности из-за тотальной валидации, но в то же время это означает, что весь блокчейн так же быстр, как его отдельные узлы, но не сумма этих узлов.

То есть скорость блокчейна ограничивается скоростью отдельного узла сети.

В настоящее время транзакции Эфериума обрабатывают-

ся не параллельно, и каждая транзакция выполняется последовательно в глобальном масштабе.

Поэтому подход заключается в blockchain sharding, где мы разбиваем все состояние сети на кучу разделов, называемых shards, которые содержат свою собственную независимую часть состояния и историю транзакций.

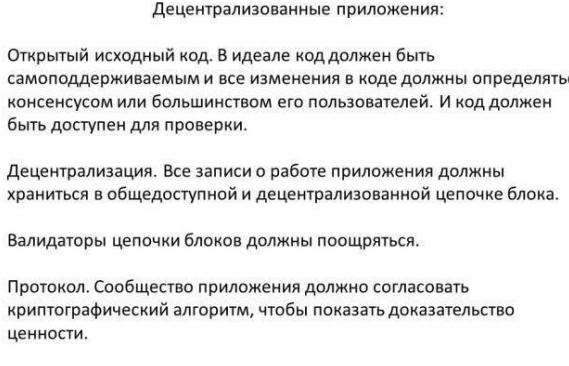
В этой системе определенные узлы обрабатывают транзакции только для определенных shards, тем самым в целом увеличивая пропускную способность транзакций.

Sharding является перспективным механизмом для масштабирования блокчейна.

Однако при таком подходе требуется решить такие сложные задачи, как меж shards коммуникация и общая безопасность такого разделенного блокчейна.

Децентрализованные приложения

Для понятия децентрализованного приложения может быть не одно определение.



Тем не менее, у децентрализованных приложений есть общие черты:

Это открытый исходный код. В идеале код должен быть самоподдерживаемым и все изменения в коде должны определяться консенсусом или большинством его пользователей. И код должен быть доступен для проверки.

Децентрализация. Все записи о работе приложения долж-

ны храниться в общедоступной и децентрализованной цепочке блока.

Валидаторы цепочки блоков должны поощряться.

Протокол. Сообщество приложения должно согласовать криптографический алгоритм, чтобы показать доказательство ценности.

Например, Bitcoin использует Proof of Work (PoW), и Ethereum в настоящее время использует Proof of Work с планами гибридного Proof of Work/Proof of Stake (PoS) в будущем.

Если мы придерживаемся вышеприведенного определения, первым децентрализованным приложением был фактически сам биткойн.

Биткойн – это самоподдерживающийся публичный журнал, который позволяет эффективные транзакции без посредников и централизованных органов.

Чтобы запустить проект децентрализованного приложения необходимо:

Создать технический документ белые страницы или белую книгу.

Ваш технический документ должен обозначить задачу, которую вы хотите решить.

Он должен четко указать намерения и цели приложения.

Опишите план распределения токенов и как вы собираетесь это делать.

Определите механизм достижения консенсуса и наймите

свою команду менеджеров и разработчиков.

Будьте честны с любыми техническими трудностями, которые вы предвидите, и четко изложите свои технические требования.

Откройте дискуссию по своему плану и сформируйте сообщество.

Получите обратную связь и соответствующим образом переработайте свой план.

После того, как приложение наберет достаточный импульс, определите дату продажи токенов.

Веб-сайт продажи токенов должен иметь всю информацию, которая может понадобиться инвесторам.

Начните разработку и приветствуйте новых разработчиков.

Децентрализованное приложение нуждается в первоначальном предложении монет или Initial Coin Offering (ICO).

Появление нового приложения в сообществе blockchain называется ICO.

ICO является мероприятием по сбору средств, которое основано на продаже токенов, которые потенциально могут принести в будущем прибыль для хорошо осведомленных и смелых инвесторов.

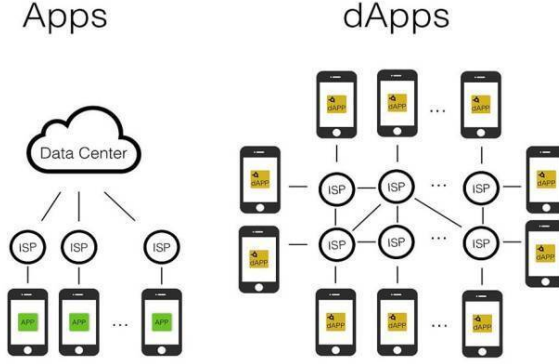
В ICO цена токена произвольно определяется командой, запускающей приложение.

После того, как токен регистрируется на бирже, его цена затем корректируется торгами.

Конечная стоимость токена будет определяться участниками сети.

Независимо от точности оценки токена ICO, остается фактом, что сам ICO является движущим силой блокчейна.

ICO – необходимый катализатор для открытия возможностей и расширения стоимости, которую предлагает блокчейн.



Итак, что такое Dapp?

Dapp или децентрализованное приложение решает задачу, которая требует использования функций blockchain и инфраструктуры blockchain для реализации своей цели.

Как правило, у Dapp есть веб-интерфейс, а также блокчейн и код, их соединяющий.

В такой архитектуре веб-интерфейс Dapp передает внешние действия от пользователей в инфраструктуру blockchain и возвращает ответ на них.

В Эфериуме Dapp инициирует транзакции для вызова функций в смарт-контракте.

Это, в свою очередь, записывает транзакции и изменение состояния в цепочке блоков.

И внешний интерфейс Dapp может быть таким же простым, как интерфейс командной строки.

Или это может быть сложное веб-приложение или мобильное приложение.

Инструменты разработки

Embark – это платформа, которая позволяет вам легко разрабатывать и развертывать децентрализованные приложения (DApps).

С Embark вы можете автоматически развертывать смарт контракты и использовать их в вашем JS-коде.

Embark позволяет следить за изменениями в смарт контракте, и, если вы обновите контракт, Embark автоматически заново развернет смарт контракт и приложение.

EtherScripter – инструмент визуального создания смарт контрактов для Ethereum.

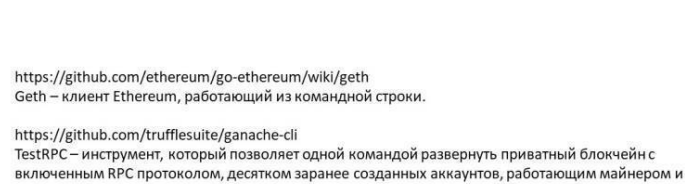
Truffle – это среда разработки и платформа тестирования

для Ethereum.

Populus – это фреймворк разработки смарт контрактов для Ethereum.

Mist – браузер для децентрализованных веб-приложений.

Parity – быстрый и легкий клиент Ethereum, который можно использовать для доступа к децентрализованным приложениям.



Geth – клиент Ethereum, работающий из командной строки.

TestRPC – это инструмент Truffle, который позволяет одной командой развернуть приватный блокчейн с включенным RPC протоколом, десятком заранее созданных аккаунтов, работающим майнером и так далее.

Remix – самая популярная браузерная среда разработки для создания смарт контрактов.

Введение в Ethereum

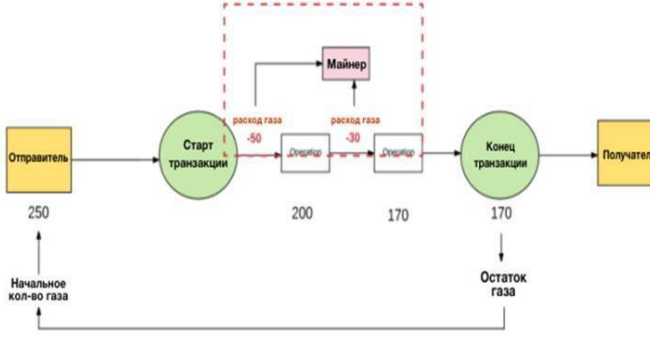
В Ethereum смарт контракты – это скрипты, которые могут обрабатывать деньги.

Эти контракты соблюдаются и заверяются сторонами, которых мы называем майнерами.

Майнеры добавляют транзакцию (выполнение смарт-контракта, оплату криптовалютой и т. д.) в публичную книгу, которая называется блоком. Блоки образуют блокчейн.

За добавление транзакции майнерам нужно оплатить что-то вроде «газа», стоимость которого определяется в контракте.

Когда вы публикуете смарт-контракт или выполняете смарт-контракт, или переводите деньги на другой аккаунт, вы платите некоторый эфир, который преобразуется в газ.



«Газ» – это название специальной единицы, используемой в Эфириуме.

Эта единица измеряет, сколько работы требуется для выполнения действия или набора действий.

Например, для запуска вычисления криптографического хеша требуется 30 газов плюс 6 газов для хэширования каждых 256 бит данных.

Каждая операция, которая может быть выполнена транзакцией или контрактом на платформе Ethereum, требует определенного количества газа, при этом операции, требующие большего количества вычислительных ресурсов, потребляют больше газа, чем операции, требующие небольшого количества вычислительных ресурсов.

Газ обеспечивает оплату соответствующей комиссии за

транзакции, переданные в сеть.

Этим мы гарантируем, что сеть не увязнет в работе, выполняя много вычислительной работы, которая никому не нужна.

Это другая стратегия оплаты, по сравнению с оплатой за транзакцию в Bitcoin, которая основана только на размере транзакции в килобайтах.

Так как Ethereum позволяет выполнять произвольно сложный компьютерный код, короткая длина кода может привести к большому количеству вычислительной работы.

Поэтому важно измерять эту работу, а не просто определять оплату в зависимости от размера транзакции или контракта.

Хотя газ – это единица измерения, нет никакого токена для газа.

То есть вы не можете владеть 1000 газами.

Вместо этого газ существует только внутри виртуальной машины Ethereum, как расчет того, сколько работы выполняется.

Когда дело доходит до фактической оплаты газа, комиссия за транзакцию взимается как определенное количество эфира – токена сети Ethereum.

Этим токеном майнеры вознаграждаются за создание блоков.

Сначала это может показаться странным.

Почему работа не измеряется сразу в эфире?

Ответ заключается в том, что эфир, как и биткойны, имеет рыночную цену, которая может быстро измениться.

Поэтому полезно отделить цену вычислительной работы от цены токена или эфира, так что стоимость работы не должна меняться каждый раз, когда движется рынок.

Операция имеет стоимость газа, но сам газ также имеет стоимость, измеряемую в эфире.

Для перевода токенов с одного кошелька на другой требуется 21000 газа.

Для создания смарт-контракта может потребоваться разное количество газа.

Для выполнения смарт-контракта также может потребоваться разное количество газа.

Выполнение вычислительных действий в виртуальной машине Эфириума стоит очень дорого.

Поэтому смарт-контракты Эфириума больше подходят для решения простых задач, таких, как реализация простой бизнес-логики или проверки цифровой подписи и других криптографических объектов, чем для реализации более сложных сценариев, как вычислительных, так и хранения данных, которые могут послужить источником высокой нагрузки на сеть.

И комиссионные платежи помогают предотвратить чрезмерную нагрузку на сеть.

Если бы в сети не было комиссионных платежей, злоумышленник легко мог бы нарушить работу сети, без ка-

ких-либо последствий, инициировав посредством транзакции выполнение бесконечного цикла вычислительных действий.

Таким образом, комиссии защищают сеть от преднамеренных атак.

Эфириум представляет собой машину состояний, функционирующую посредством транзакций.

Разные счета инициируют переход Эфириума из одного глобального состояния в другое.

В самом базовом смысле, транзакция является частью команды с криптографической подписью, которая генерируется счётом владельца, сериализуется и затем передаётся в блокчейн.

И есть два типа транзакций: сообщения и транзакции создания контрактов.

Можно сказать, что транзакции являются своего рода мостами между внешним миром и внутренним состоянием Эфириума.

При этом смарт контракты могут взаимодействовать друг с другом.

Контракты, существующие в глобальной области действия состояния Эфириума, могут взаимодействовать с контрактами внутри этой же области действия.

Они делают это посредством «сообщений» или «внутренних транзакций» в адрес других контрактов.

Можно сказать, что сообщения или внутренние транзак-

ции схожи с обычными транзакциями, но отличаются тем, что они НЕ генерируются счетами владельцев.

Их генерируют сами контракты.

Это виртуальные объекты, которые, в отличие от транзакций, никогда не сериализуются и существуют только в среде выполнения Эфириума.

Когда один контракт отправляет внутреннюю транзакцию на адрес счёта другого контракта, выполняется соответствующий код, прописанный в контракте-получателе.

При этом лимит газа, устанавливаемый счётом владельца, должен быть достаточным для выполнения транзакции, включая любые подзадачи – как, например, сообщения между счетами контрактов.

Если в цепочке транзакций и сообщений при выполнении отдельного сообщения был исчерпан лимит газа, то передача этого и всех следующих за ним сообщений, связанных с этой транзакцией, будет отменена.

Таким образом, газ – это единица, используемая для обозначения размера комиссии за определённое вычислительное действие.

Цена газа представляет собой то количество эфиров, которое вы готовы потратить на каждую единицу газа.

Она измеряется в «Gwei».

«Wei» – это наименьшая единица эфира, 1 эфир = 10^{18} Wei.

Один Gwei равен 1 000 000 000 Wei.

Для каждой транзакции отправитель устанавливает лимит газа и цену газа.

Произведение цены газа и лимита газа даёт максимальное количество Wei, которое отправитель готов заплатить за выполнение транзакции.

Предположим, что отправитель устанавливает лимит газа 50 000, а цену газа 20 Gwei.

Это означает, что отправитель готов потратить на выполнение этой транзакции не более чем 50 000 x 20 Gwei = 1 000 000 000 000 000 Wei = 0,001 эфира.

Таким образом, лимит газа представляет собой максимальное количество газа, которое готов оплатить отправитель.

При этом, на счету отправителя должно находиться достаточное количество эфиров для оплаты максимального количества газа.

После выполнения транзакции эквивалент любого количества неиспользованного газа возвращается на счёт отправителя в эфирах, будучи обменянным по первоначальной ставке.

Если отправитель не предоставляет необходимого для выполнения транзакции количества газа, и оно исчерпывается в процессе её выполнения, то такая транзакция признаётся недействительной.

В этом случае выполнение транзакции прерывается, любые произведённые ею изменения в состоянии сети отменяются и Эфириум возвращается в состояние, в котором он находился перед началом транзакции – как если бы её вовсе не было.

Так как к тому моменту, как отпущенное на транзакцию количество газа было исчерпано, машина уже затратила ресурсы на выполнение вычислений, логично, что в этом случае плата за газ отправителю транзакции не возвращается.

Все средства, уплачиваемые за газ отправителем транзакции, отправляются на адрес выгодоприобретателя, обычно это адрес майнера.

Как правило, чем выше цена газа, которую отправитель готов заплатить, тем большее вознаграждение получит майнер за выполнение транзакции, и, следовательно, тем выше вероятность того, что майнер выберет именно эту транзакцию.

Таким образом, майнеры могут выбирать, какие транзакции они хотят подтвердить либо проигнорировать.

Для того чтобы помочь отправителю понять, какую стоимость газа ему следует установить, у майнеров есть возможность сообщить публично о минимальной цене газа, при которой они будут выполнять транзакции.

Газ используется для оплаты не только вычислительных действий, но и хранения данных.

Общая плата за хранение пропорциональна наименьшему использованному объему, кратному 32 байтам числу.

В комиссии за хранение данных есть свои нюансы.

Например, так как увеличение размера хранилища увеличивает размер базы данных состояний для всех узлов, то для пользователей сети предусмотрен стимул к сокращению количества хранимых данных до необходимого минимума.

Поэтому, если транзакция инициирует выполнение действия, в результате которого объём занимаемого хранилища сокращается, то комиссия за выполнение этой операции не взимается, ПЛЮС производится возврат средств за освобождённый объём.

Таким образом, есть разница между транзакцией, в кото-

рой газ закончился и транзакцией, которая не имеет достаточно высокого уровня цены за газ.

Если цена на газ, которая установлена в транзакции, слишком низкая, никто даже не потрудится выполнить транзакцию.

Она просто не будет включена в блокчейн майнерами.

Но если предоставлена приемлемая цена на газ, а затем транзакция приводит к такой вычислительной работе, что лимит газа будет исчерпан, то этот газ считается «потраченным», и он не вернется обратно.

Майнер просто прекратит обработку транзакции, вернет любые сделанные изменения, но все равно включит ее в блок-цепочку в качестве «неудачной транзакции», удержав за нее плату.

Предоставление слишком большой цены за газ также отличается от предоставления слишком большого количества газа.

Если вы установили очень высокую цену на газ, вы в конечном итоге заплатите много эфира всего за несколько операций точно так же, как установление сверхвысокой комиссии за транзакцию в биткойне.

Однако, если вы указали нормальную цену на газ, и просто подключили больше газа, чем нужно, избыточная сумма будет вам возвращена.

Шахтеры взимают плату только за ту работу, которую они на самом деле делают.

Эфириум построен таким образом, что интервал между блоками в нём значительно меньше (~15 секунд), чем в других блокчейнах и, например, в Биткоине (~10 минут).

Это позволяет ускорить процесс обработки транзакций.

Однако один из недостатков короткого интервала между блоками заключается в том, что майнеры находят больше конкурирующих блоков.

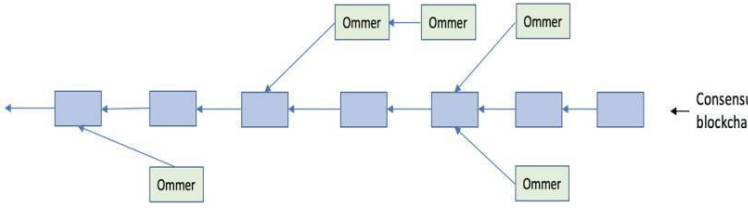
Эти конкурирующие блоки называют ещё потерянными, то есть намайненными блоками, но не включёнными в основную цепочку.

Оммеры, блоки, чей родительский блок тождественен родительскому блоку текущего блока, нужны для того, чтобы майнеры могли получить вознаграждение за включение в блокчейн таких потерянных блоков.

Оммеры, включаемые майнерами в блокчейн, должны быть «действительными», то есть быть сформированными не более чем в шестом поколении от настоящего блока.

После шести дочерних блоков, ссылаться на устаревшие потерянные блоки больше нельзя (так как запись в блокчейн более старых транзакций заметно усложнит процессы сети).

За подтверждение оммеров майнеры получают меньшее вознаграждение, чем за полный блок.



Виртуальная машина Ethereum является сердцем Ethereum.

Эта виртуальная машина предназначена для запуска всеми участниками одноранговой сети.

Она может читать и записывать в блокчейн как исполняемый код, так и данные, проверять цифровые подписи и запускать код, когда виртуальная машина получит сообщение, подтвержденное цифровой подписью, и информация, хранящаяся в блокчейне, говорит, что это целесообразно.

Ethereum представляет собой одноранговую сеть, где каждый одноранговый узел хранит одну и ту же копию базы данных blockchain и запускает виртуальную машину Ethereum для поддержания и изменения ее состояния, которое хранится в блокчейне.

Теперь разберемся со всем этим поподробнее.

Блокчейн Ethereum хранит блоки, которые в свою очередь хранят транзакции и состояние.

И блокчейн Ethereum во многом похож на блокчейн биткойнов, хотя он имеет некоторые отличия.

Основное различие между Ethereum и Bitcoin в отношении архитектуры blockchain заключается в том, что, в отличие от биткойнов, блоки Ethereum содержат копию как списка транзакций, так и самого последнего состояния.

Состояние хранится в специальной структуре данных, называемой деревом Merkle.

Таким образом, блокчейн Ethereum включает в себя корень состояния (по одному на блок), который хранит корневой хэш дерева, представляющего состояние системы во время создания блока.

И эта структура данных хранит специальные объекты, называемые аккаунтами или счетами, причем каждый аккаунт имеет 20-байтовый адрес и содержит четыре поля:

Счетчик nonce, используемый для проверки, что каждая транзакция может обрабатываться только один раз;

Текущий баланс эфира;

Код смарт-контракта аккаунта, если присутствует;

И хранилище аккаунта (пустое по умолчанию);

И данные состояния (вместе с балансами, контрактами) хранятся каждым клиентом Ethereum (или узлом Ethereum).

Есть два типа аккаунтов:

Счет внешнего владельца, который контролируется закрытым ключами и не имеет никакого кода, связанного с ним.

И счет контракта, который контролируются его кодом контракта, и имеет связанный с ним код.

Счет внешнего владельца может отправлять сообщения другим счетам внешних владельцев или другим счетам контрактов, создавая и подписывая транзакцию с использованием своего закрытого ключа.

Сообщение между двумя счетами внешних владельцев – это просто передача денег.

Но сообщение от счета внешнего владельца на счет контракта активирует код счета контракта, позволяя ему выполнять различные действия (например, передавать токены, записывать данные во внутреннее хранилище, майнить новые токены, выполнять вычисления, создавать новые контракты и т. д.).

В отличие от счетов внешних владельцев, счета контрактов не могут инициировать новые транзакции самостоятельно.

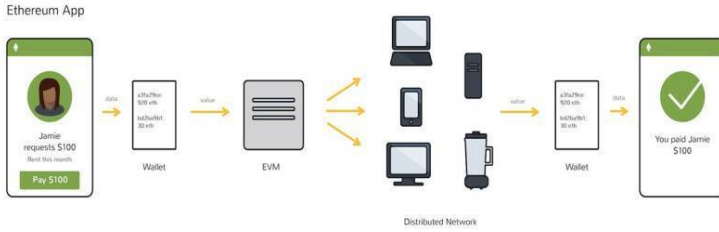
Вместо этого счета контрактов могут только запускать транзакции в ответ на другие транзакции, которые они получили.

Таким образом, Ethereum имеет набор аккаунтов.

У каждого аккаунта есть владелец и баланс в эфирах.

И если владелец аккаунта может доказать свою идентич-

ность, он может перенести эфир с его аккаунта на другой. Эта операция называется транзакцией.

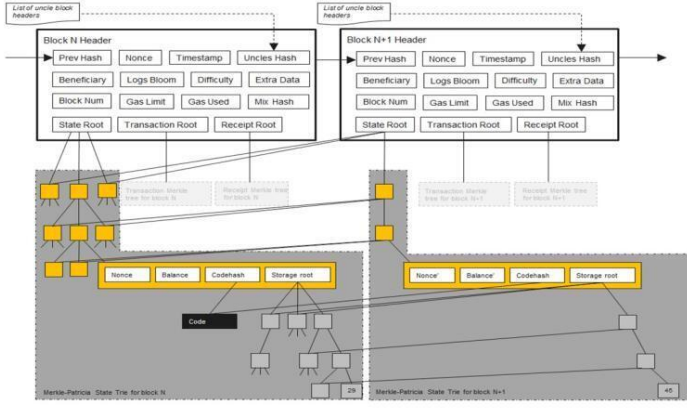


Виртуальная машина Ethereum представляет собой систему обработки транзакций.

То есть у нас есть состояние – это совокупность всех аккаунтов и их балансов.

Далее мы применяем одну или несколько транзакций, и мы получаем новое состояние – это обновленный набор аккаунтов и их балансов.

Аккаунт выполняет некоторый код, когда он получает транзакции, с помощью смарт-контракта аккаунта.



Когда смарт-контракт создается или, когда его пробуждает транзакция, код контракта может читать и записывать данные в хранилище аккаунта.

Если ваша транзакция удаляет данные из хранилища контрактов, то есть после того, как ваша транзакция будет выполнена, общее хранилище контрактов станет меньше, вы получите премию в виде газа.

Полный узел Ethereum синхронизирует блокчейн, загружая всю цепочку, от блока генезиса до текущего блока, выполняя все транзакции, содержащиеся внутри.

Также полный узел может быть без выполнения каждой транзакции.

Но независимо от этого, любой полный узел содержит всю цепочку блоков.

Но если узлу не нужно выполнять каждую транзакцию или быстро запрашивать исторические данные, нет необходимости хранить всю цепочку блоков.

Такие узлы называются легкими узлами.

Вместо того, чтобы загружать и хранить всю цепочку и выполнять все транзакции, легкие узлы загружают только цепочку заголовков блоков.

Что дает возможность легкому узлу при необходимости выбрать заголовок и загрузить данные этого блока.

Термин «транзакция» используется в Ethereum для ссылки на подписанный пакет данных, который хранит сообщение, отправляемое из внешнего аккаунта.

Транзакции содержат информацию о получателе сообщения, подпись, идентифицирующую отправителя, количество эфира для передачи от отправителя получателю, дополнительное поле данных, значение газа для транзакции и значение цены газа.

Контракт может получить доступ к данным, содержащимся в поле данных.

Как пример использования, если контракт функционирует как служба регистрации домена, тогда данные транзакции будут содержать домен и IP-адрес для его регистрации.

Контракт будет читать эти значения из данных сообщения и соответствующим образом размещать их в хранилище.

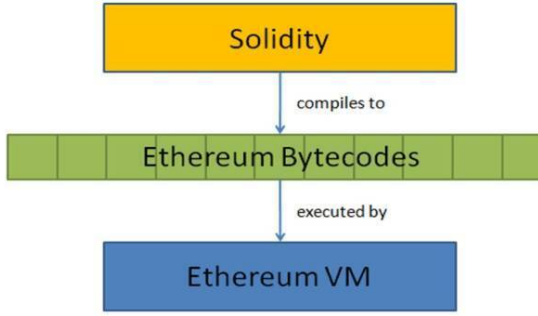
Контракты имеют возможность отправлять «сообщения» другим контрактам.

Эти сообщения – это виртуальные объекты, которые никогда не сериализуются и существуют только в среде исполнения Ethereum. Такое сообщение содержит информацию об отправителе сообщения, получателе сообщения, количество эфира для передачи вместе с сообщением, дополнительное поле данных и значение газа.

По сути, сообщение похоже на транзакцию, за исключением того, что она создается контрактом, а не внешним аккаунтом.

Как и транзакция, сообщение приводит к тому, что аккаунт получателя запускает свой код.

Таким образом, контракты могут иметь отношения с другими контрактами точно так же, как и внешние аккаунты.



Код в контрактах Ethereum пишется на низкоуровневом, основанном на стеке языке байт-кода, называемом кодом виртуальной машины Ethereum.

Код состоит из серии байтов, где каждый байт представляет операцию.

Для написания контрактов рекомендуется высокоуровневый язык Solidity, код которого затем компилируется в язык виртуальной машины Ethereum.

Виртуальная машина Ethereum при своем запуске оперирует глобальным состоянием, содержащим все аккаунты и включающим балансы и хранилища, транзакцией или сообщением, и кодом контракта.

И если транзакция добавляется в блок, выполнение кода, порожденное этой транзакцией, будет выполняться всеми узлами, теперь и в будущем, которые загружают и проверяют этот блок.

Поверх сети Ethereum создаются три типа приложений.

Первая категория – это финансовые приложения, предоставляющие пользователям способы заключения контрактов с использованием их денег.

Это включает в себя суб-валюты, финансовые деривативы, контракты хеджирования, сберегательные кошельки, и т. д.

Вторая категория – это полу-финансовые приложения, в которых задействованы деньги, но есть также не денежная сторона, например, оплата вычислительных задач.

Наконец, существуют такие приложения, как онлайн-голосование, которые не являются финансовыми.

Контракт сам по себе не является децентрализованным приложением.

Децентрализованное приложение состоим из комбинации контракта и графического интерфейса для использования этого контракта.

При этом интерфейс реализован как веб-страница HTML/CSS/JS со специальным Javascript API в виде библиотеки web3.js для работы с блок-цепочкой Ethereum.

Под капотом эта библиотека связывается с локальным узлом через вызовы remote procedure call RPC.

И такое приложение будет работать только в клиенте Ethereum, а не в обычном веб-браузере.

В Ethereum также есть два дополнительных протокола, реализующих поддержку одноранговового обмена сообщениями и статическими файлами.

Одноранговый распределенный протокол для обмена сообщениями получил название whisper.

Он предоставляет пользователям возможности для личного защищенного общения с поддержкой отправки сообщений одному или нескольким адресатам и рассылке широковещательных сообщений.

Одноранговый протокол для обмена статическими файлами получил название swarm.

Whisper – это одноранговый протокол для конфиденци-

ального обмена сообщениями с коротким сроком жизни.

Заголовок сообщений (тема) в Whisper хэшируется, а сами сообщения могут быть зашифрованы с помощью ключей в целях защиты данных.

Swarm представляет собой мотивированный файлообмен.

Файлы делятся на части, хранящиеся на узлах сети.

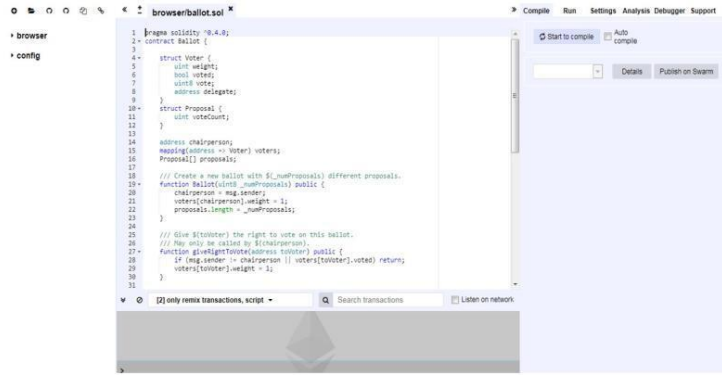
Для ведения учета отправленных и полученных частей файлов узлы используют специальный учетный протокол, а сама деятельность узлов оплачивается микроплатежами – мера, призванная поощрить кооперацию между ними.

Solidity Remix

Solidity – это высокоуровневый язык для виртуальной машины Ethereum с синтаксисом, похожим на JavaScript.

Программы на языке Solidity транслируются в байткод виртуальной машины Ethereum.

http://remix.ethereum.org/



Для разработки смарт контрактов и изучения языка Solidity рекомендуется использовать среду разработки Remix.

Remix – это среда IDE для языка программирования Solidity, которая имеет встроенный отладчик и среду тести-

рования.

Среда Ремикс позволяет разрабатывать смарт-контракты с помощью редактора Solidity, отлаживать выполнение смарт-контракта, обеспечивает доступ к состоянию и свойствам уже развернутого смарт-контракта, отлаживать уже совершенную транзакцию, анализировать код Solidity, чтобы уменьшить ошибки кодирования и обеспечить соблюдение лучших практик.

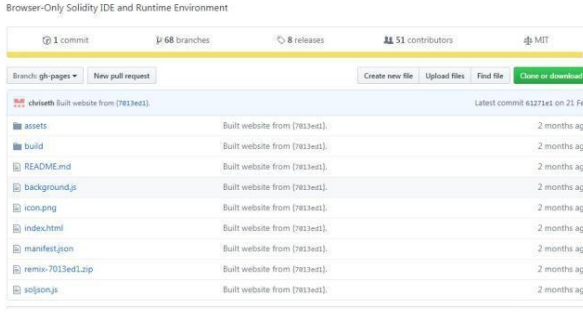
Вместе с Mist или любым инструментом, который использует библиотеку web3, Remix можно использовать для тестирования и отладки децентрализованного приложения.

Доступна онлайн версия среды Remix.

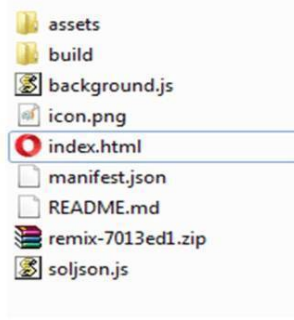
Также можно установить локальную версию среды Remix.

Преимущество запуска локальной версии среды Remix заключается в том, что вы можете связаться с клиентом узла Ethereum, запущенным на вашей локальной машине через API-интерфейс Ethereum JSON-RPC, и локально выполнить смарт контракты.

https://github.com/ethereum/remix-ide/tree/gh-pages



Для установки локальной версии среды Remix можно открыть ветку github gh-pages и скачать архив.



Затем распаковать его и открыть страницу index.html.

```
npm install remix-ide -g

remix-ide

http://localhost:8080
```

Также можно установить Remix как npm модуль.

После установки Remix запускается командой remix-ide и открывается в браузере по адресу localhost.

При такой установке также устанавливается модуль Remixd – модуль npm, который предоставляет веб-приложению Ремикса доступ к папке на локальном компьютере.

По умолчанию это папка user.

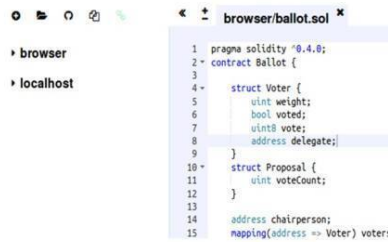
Из Remix IDE вам необходимо активировать это соединение с локальным компьютером.

Для этого нажмите на значок подключения к локальному хосту.

В результате общая папка будет доступна в проводнике файлов среды Remix.

```
npm install -g remixd

remixd -s ~/remix
```



При запуске среды Remix из архива с помощью страницы index.html, доступа к папке локального компьютера не будет.

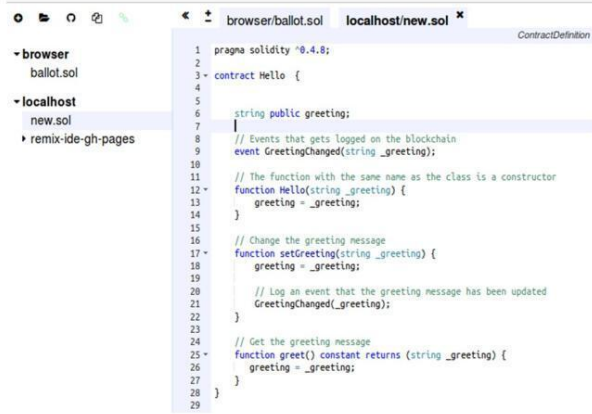
Для доступа нужно установить модуль Remixd глобально.

А затем запустить Remixd и расшарить какую-нибудь папку.

После этого можно перезапустить Remix и нажать кнопку соединения.

В результате общая папка будет доступна в проводнике файлов среды Remix.

Remix File Explorer



В проводнике файлов по умолчанию отображаются все файлы, хранящиеся в вашем браузере.

Вы можете увидеть их в папке браузера.

Вы всегда можете переименовать, удалить или добавить новые файлы в проводник файлов.

Обратите внимание, что очистка хранилища браузера удалит все файлы, которые вы написали.

Чтобы этого избежать, нужно использовать Remixd, который позволяет хранить и синхронизировать файлы в браузере с папкой локального компьютера.

При этом все изменения, сделанные в редакторе Remix

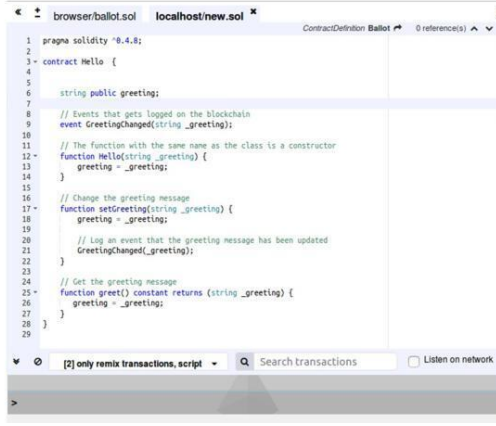
будут автоматически сохраняться в файле папки локального компьютера.

Помимо соединения с локальным компьютером, кнопки вверху проводника файлов позволяют создать новый файл в хранилище браузере, импортировать локальный файл в хранилище браузера, скопировать файл из хранилища браузера в другой экземпляр Remix.

Также можно опубликовать файл из хранилища браузера в анонимный публичный gist.

Gist – это сервис Github, который позволяет обмениваться отдельными файлами, частями файлов и полными приложениями с другими людьми.

Remix Solidity Editor



Редактор Remix позволяет перекомпилировать код при каждом изменении текущего файла или выборе другого файла.

Он также обеспечивает подсветку синтаксиса, сопоставляемую с ключевыми словами языка Solidity.

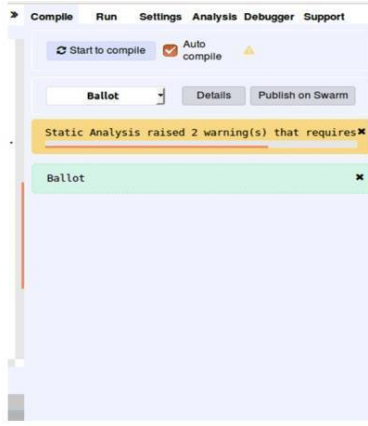
Редактор Remix отображает открытые файлы в виде вкладок, отображает предупреждения компиляции и ошибки.

Кроме того, Remix непрерывно сохраняет текущий файл (в течение 5 секунд после последних изменений).

Кнопка +/– в верхнем левом углу позволяет увеличить/уменьшить размер шрифта редактора.

Внизу редактора расположен терминал, который отображает журнал при отладке контракта.

Remix Compile



Remix запускает компиляцию каждый раз при изменении текущего файла или выборе другого файла.

Если в контракте много зависимостей и требуется много времени для компиляции, можно отключить автокомпиляцию.

После каждой компиляции обновляется список со всеми скомпилированными контрактами.

В диалоговом окне Details отображается подробная информация о текущем выбранном контракте.

Ниже отображаются ошибки компиляции и предупреждения.

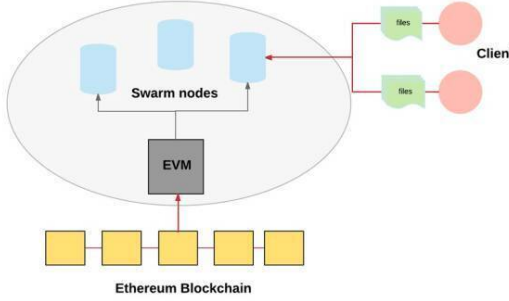
Здесь вы также можете опубликовать свой контракт на Swarm.

Remix Swarm

Swarm – это распределенная платформа хранения и служба распространения контента.

Хранение данных большого объема в самом блокчейне может стоить немалых денег.

Эта проблема решается с помощью децентрализованного хранилища Ethereum Swarm.



Swarm обеспечивает децентрализованное хранение данных в хранилищах узлов, владельцы которых отдают свои ресурсы в общее пользование.

Для работы децентрализованного хранилища Swarm устанавливаются узлы сети Swarm, которая работает поверх сети Ethereum.

При этом владельцы таких узлов получают вознаграждение за предоставление ресурсов, и стоимость размещения данных ниже, чем в традиционных облачных хранилищах.

Если вернуться к среде Remix, то при нажатии кнопки Publish on Swarm, в хранилище браузера будет создано хранилище Swarm, в которое сохранится файл.

При этом опубликованные данные будут содержать исходный код abi и solidity.

Application Binary Interface (ABI) – это механизм кодирования/декодирования данных в и из машинного кода виртуальной машины.

Исходный код ABI – это определение интерфейса контракта для доступа к бинарным данным контракта в блокчейне через интерфейс ABI.

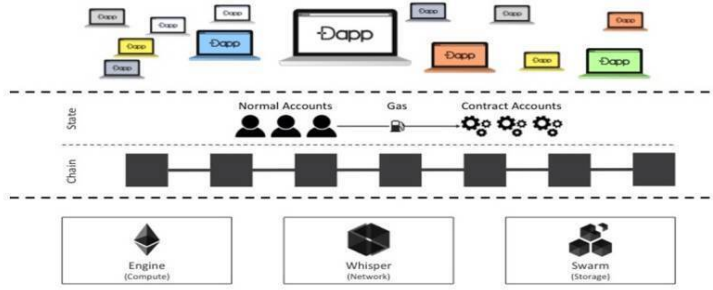
Application Binary Interface (ABI) представляет собой схему кодирования данных, используемую в Ethereum для работы со смарт-контрактами.

Для реальной публикации файла в сеть Swarm требуется установка узла Swarm.

Кнопку Publish on Swarm среды Remix можно использовать для автоматической проверки исходного кода смарт-контракта или для извлечения определения интерфейса

ABI.

При реальной публикации файла в сеть Swarm с помощью установленного узла Swarm, доступ к опубликованному файлу можно получить через свой локальный работающий узел Swarm по адресу http://localhost:8500/ и дальше URL адрес файла.



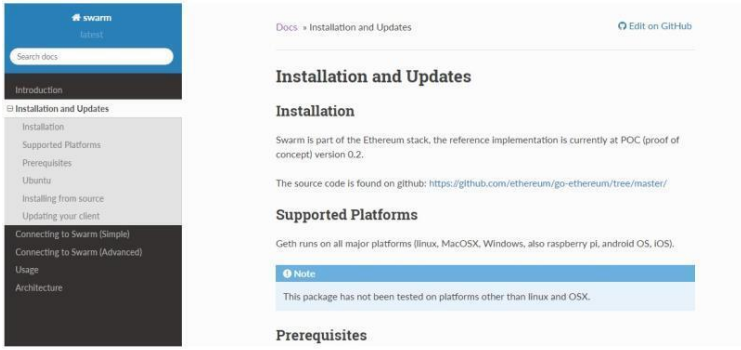
http://localhost:8500/bzz:/HASH

URL адрес файла представляет собой имя протокола bzz и дальше хэш файла.

Таким образом, Swarm определяет протокол bzz, который работает поверх сети ethereum.

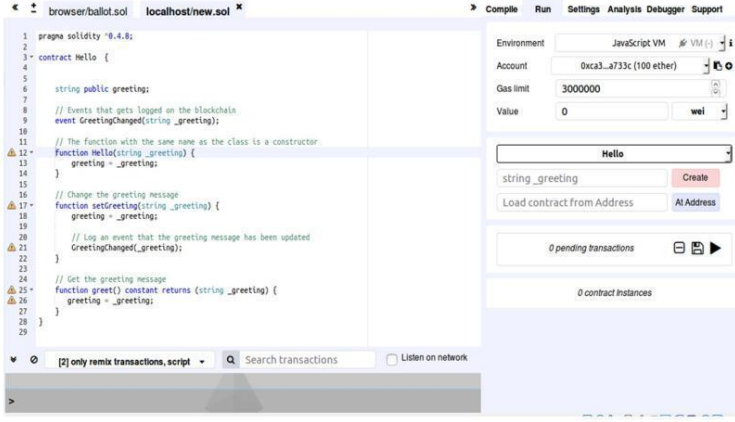
И сеть Swarm представляет собой набор узлов в сети Ethereum, каждый из которых запускает протокол bzz.

https://swarm-guide.readthedocs.io/en/latest/installation.html



Установку узла Swarm и его использование можно посмотреть в документации.

Remix Run



Вкладка Run среды Remix позволяет отправлять транзакции в текущую среду выполнения.

Здесь есть настройки, которые позволяют напрямую влиять на выполнение транзакции.

В списке можно выбрать среду выполнения.

Это JavaScript VM, где все транзакции будут выполняться в блокчейне браузера.

Это означает, что ничего не будет сохранено, и перезагрузка страницы браузера перезапустит новую цепочку с нуля, старая не будет сохранена.

Среда выполнения Injected Provider. Remix будет подклю-

чаться к инструменту со встроенным web3. Mist и Metamask являются примерами поставщиков, которые интегрированы с web3.

Среда выполнения Web3 Provider. В этом случае Remix будет подключаться к удаленному узлу.

И вам нужно будет указать URL-адрес выбранному поставщику, такому как geth, parity или любому другому клиенту Ethereum.

Что такое Web3?

Web3 – это официальная Javascript библиотека Ethereum, которая позволяет работать с Ethereum из кода пользовательского приложения.

web3.js – это библиотека, которая позволяет взаимодей-

ствовать с локальным или удаленным узлом ethereum, используя соединение HTTP или IPC.

Конец ознакомительного фрагмента.