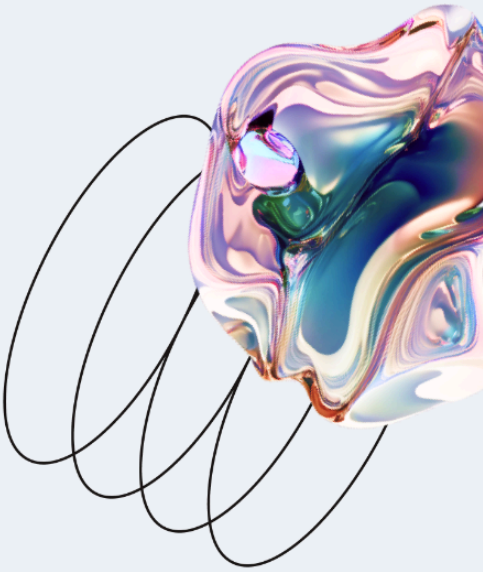
GeekBrains



Блокчейн второго поколения

Блокчейн

Оглавление

Введение

Всем привет! Мы продолжаем курс о блокчейне, на котором изучаем технические основы этой технологии и знакомимся с решением прикладных задач, как в сфере финансов, так и в разработке приложений.

На прошлой лекции мы говорили о том, как появился Биткоин и что в нём революционного. Остановились мы на том, что его применение ограничено использованием криптовалюты. То есть Биткоин — это специально разработанный блокчейн, а не приложение, работающее поверх блокчейна.

Сегодня мы познакомимся с блокчейнами второго поколения на примере Ethereum. Их отличие в том, что они формируют абстрактный базовый уровень, позволяющий запускать различные приложения на одной и той же платформе. Большинство крупнейших на сегодня публичных блокчейнов относятся именно ко второму поколению. Помимо Ethereum это, например, BNB Chain и Solana.

Почему мы будем рассматривать именно Ethereum? Эта платформа отлично подходит для знакомства с блокчейнами второго поколения. Сообщество разработчиков, работающих с ней, стремительно увеличивается — быстрее, чем у любого аналога. По сравнению с другими блокчейнами Ethereum больше всего ориентирован на разработчиков. Любой специалист знакомый с программированием приложений на JavaScript, может легко окунуться в мир Ethereum и очень быстро начать писать работающий код. Часто можно даже услышать, что для создания токена требуется всего лишь пять строчек кода. Но, конечно, это палка о двух концах. Писать код, может быть, и легко, а вот писать хороший и безопасный код — намного сложнее.

На этой лекции вы узнаете:

- как устроена экосистема Ethereum и из чего она состоит,
- как разрабатываются децентрализованные приложения и смарт-контракты в Ethereum.

Словарь терминов

Конечный автомат — модель дискретного устройства, имеющего один вход, один выход и в каждый момент времени находящегося в одном состоянии из множества возможных.

Полнота по Тьюрингу — характеристика исполнителя в теории вычислимости, означающая возможность реализовать на нем любую вычислимую функцию.

DApp (decentralized application) — децентрализованное приложение. В сущности, это смарт-контракт с пользовательским веб-интерфейсом. В более широком смысле DApp — это веб-приложение, построенное поверх открытых децентрализованных пиринговых инфраструктурных сервисов. Кроме того, многие DApp-приложения содержат децентрализованное хранилище и/или протокол с платформой для обмена сообщениями.

EVM (Ethereum Virtual Machine) — виртуальная машина, основанная на стековой архитектуре и выполняющая байт-код. В Ethereum модель выполнения определяет способ изменения состояния системы в зависимости от последовательности инструкций в байт-коде и небольшого массива данных источника в виртуальной среде.

Geth (Go Ethereum) — одна из самых известных реализаций протокола Ethereum, написанная на языке Go.

Web3 — третья версия Интернета, изначально предложенная Гэвином Вудом и нацеленная на веб-приложения — как с централизованным механизмом владения и управления, так и построенных на основе децентрализованных протоколов.

web3.js — JavaScript-библиотека, которая связывает приложения, запущенные в браузере, с блокчейном Ethereum.

Swarm — интерфейс для пирингового сетевого хранилища.

Whisper — пиринговый сервис обмена сообщениями.

Geth — клиент, который реализует спецификацию Ethereum и взаимодействует с другими клиентами. Go Ethereum. Одна из самых известных реализаций протокола Ethereum, написанная на языке Go.

Тестнет (тестовая сеть) — сеть, в которой запускаются симуляции основной сети Ethereum.

Газ — виртуальное «топливо», которое используется в протоколе Ethereum для выполнения смарт-контрактов.

Учетная запись с внешним владельцем (Externally Owned Account, EOA) — учетная запись, созданная пользователем-человеком в блокчейнсети Ethereum.

Учетная запись контракта — учетная запись с кодом, который выполняется при получении транзакции от другой учетной записи (EOA или контракта).

Смарт-контракт — программа, которая выполняется в рамках вычислительной инфраструктуры Ethereum.

Эфир (ETH) — стандартизированная криптовалюта, которая используется в экосистеме Ethereum и покрывает стоимость газа при исполнении смарт-контрактов.

Сравнение Ethereum с Биткоином

Для начала давайте опишем сходства и отличия Ethereum по сравнению с Биткоином. После этого будет проще разобраться в возможностях и характеристиках этой технологии.

С прошлой лекции вы помните, что технология блокчейн пришла вместе с Биткоином в 2009 году. После того как он выдержал испытание временем, люди поверили в потенциал блокчейна. Примеры использования вышли за пределы банковского и финансового сектора и охватило другие отрасли, такие как логистика, розничная торговля, электронная коммерция, здравоохранение, энергетика и правительственный сектор. Для этих целей разрабатывали специальные варианты блокчейна, которые решают конкретные задачи.

По мере роста популярности Биткоина, разработчики стали создавать различные децентрализованные приложения с совершенно новыми блокчейнами или пытались модифицировать клиент Bitcoin Core для увеличения набора функций. Это было сложно и отнимало много времени. На помощь разработчикам пришла альтернативная платформа Ethereum. Его идея заключается в том, чтобы разрабатывать различные блокчейн-приложения на единой платформе, а не создавать заново блокчейны для каждого приложения в отдельности.

Ethereum имеет много общих компонентов с другими открытыми блокчейнами:

- пиринговая сеть, объединяющая пользователей;
- византийский отказоустойчивый алгоритм консенсуса для синхронизации обновлений состояния;
- криптографические концепции — цифровые подписи и хеши;
- цифровая валюта — эфир.

Все или большинство из этих компонентов обычно объединяются в программный клиент. Например, эталонная реализация Биткоин разработана в рамках открытого проекта Bitcoin Core и оформлена в виде клиента bitcoind. В Ethereum, вместо

эталонной реализации, существует эталонная спецификация — математическое описание системы в Yellow Paper, так называемой желтой бумаге. Существует целый ряд клиентов , построенных в соответствии с этим описанием.

Несмотря на наличие собственной криптовалюты, основная цель Ethereum состоит не в том, чтобы быть платежной сетью. Хотя эфир является неотъемлемой частью платформы и необходим для ее работы, он задумывался как валюта для оплаты использования сети Ethereum в качестве глобального компьютера.

Также в отличие от Биткоина с его очень ограниченным скриптовым языком, платформа Ethereum задумывалась как программируемый блокчейн общего пользования, который работает на виртуальной машине, способной выполнять код произвольной и неограниченной сложности. Если язык скриптов в Биткоине был намеренно сделан малофункциональным, способным лишь проверять условия расходования средств, то язык Ethereum является тьюринг-полным языком. Подробнее, о полноте по Тьюрингу мы поговорим позже, а сейчас простыми словами можно сказать, что это означает, что Ethereum может играть роль универсального компьютера.

> 💡 Одноранговую систему электронных денег, такую как Биткоин, можно легко построить на платформе Ethereum в виде децентрализованного приложения (DApp). Любой актив, обладающий внутренней ценностью, например, земля, автомобили, дома, голоса избирателей, может легко передаваться в виде токенов через соответствующие приложения Ethereum.

Таким образом, до появления Ethereum термин «блокчейн» использовался для обозначения всех компонентов, присутствующих в Биткоине. Это была краткая отсылка к набору технологий, вобравших в себя его характеристики. Сегодня существует множество блокчейнов второго поколения с разными свойствами. Чтобы лучше понять характеристики того или иного блокчейна, нам нужны уточняющие критерии, такие как: открытый, публичный, глобальный, децентрализованный, нейтральный и устойчивый к цензуре.

Кто создал Ethereum

Проект Ethereum начат в то время, когда люди распознали всю мощь модели Биткоина и пытались найти ему новые применения, помимо криптовалюты как средства платежа. Но разработчики столкнулись с дилеммой: строить свои проекты поверх Биткоина или создавать новый блокчейн. В первом случае пришлось бы

смириться с узкими функциональными рамками этой сети и искать обходные решения. Скромный набор типовых транзакций и данных, размеров хранилища ограничивали круг применения приложений, которые можно было выполнять непосредственно в рамках протокола Биткоина. Все, что не вписывалось в эти рамки, требовало дополнительных уровней реализации поверх уже существующей сети, что сразу же нивелировало многие преимущества публичного блокчейна.

Если проект требовал больше свободы и гибкости, но при этом должен был функционировать в рамках сети, у него имелся лишь один вариант: переход на новый блокчейн. Для этого нужно было сделать много работы: собрать все элементы инфраструктуры, тщательно их протестировать и т. д.

Ближе к концу 2013 года Виталик Бутерин, молодой программист и поклонник Биткоина, начал размышлять о дальнейшем расширении его функциональности. Он начал распространять первичную документацию, в которой излагалась идея, стоящая за Ethereum: тьюринг-полный блокчейн общего пользования.

Одним из первых, кто связался с Виталиком, стал Гэвин Вуд. Он предложил в качестве помощи свои навыки программиста на C++. Гэвин стал соучредителем Ethereum, одним из его архитекторов и техническим директором проекта. Вот что написал Виталик в своей статье, «Предыстория Ethereum»:

«Это время, когда протокол Ethereum был всецело моим созданием. Однако с этого момента ко мне начали присоединяться новые участники. Безусловно, по части протокола самым выдающимся из них был Гэвин Вуд.
Гэвин Вуд также в значительной мере отвечал за корректировку образа и видения проекта: вместо платформы для создания программируемых денег с контрактами на основе блокчейна, которые могут хранить цифровые активы и передавать их в соответствии с заранее заданными правилами, Ethereum превратился в вычислительную платформу общего пользования. Все началось с едва заметных изменений в приоритетах и терминологии, и со все большим упором на концепцию Web 3, в котором блокчейн-сеть Ethereum была лишь частью пакета децентрализованных технологий наряду с Whisper и Swarm».

💡 Протокол связи **Whisper** позволяет децентрализованным приложениям связываться друг с другом. Он предоставляет распределенные функции обмена сообщениями. Whisper поддерживает одноадресную, многоадресную и широковещательную рассылку сообщений.

💡 **Swarm** — это платформа распределенного хранения статических файлов в режиме одноранговой пиринговой сети, а также служба распространения файлов. Swarm обеспечивает адекватную децентрализацию и избыточное хранение данных блокчейна Ethereum, кодов приложений DApp и тому подобной информации. В отличие от сети WWW, загрузка файлов в Swarm не сосредоточена на одном веб-сервере. Swarm имеет нулевое время простоя, устойчива к DDOS-атакам и отказоустойчива.

Как и Сатоши Накамото, Виталик Бутерин и Гэвин Вуд не просто изобрели новую технологию. Они объединили инновационные концепции с существующими технологиями так, как еще никто этого не сделал, и создали прототип платформы, чтобы продемонстрировать свои идеи всему сообществу.

На формирование и оттачивание видения Ethereum ушли годы. Наконец 30 июля 2015 года был добыт первый блок сети Ethereum. Всемирный компьютер начал служить человечеству.

Основные компоненты Ethereum

Конечный автомат

Термин, с которым вы наверняка столкнетесь при изучении Ethereum — «конечный автомат». Давайте разберемся, что он означает и какое отношение имеет к блокчейну.

💡 **Конечный автомат** — абстрактная математическая модель из теории алгоритмов, которая содержит конечное число состояний чего-либо. Строго говоря, это модель дискретного устройства, имеющего один вход, один выход и в каждый момент времени находящегося в одном состоянии из множества возможных.

Блокчейн Биткоина отслеживает состояние и принадлежность денежных средств. Можно рассматривать его как распределенный конечный автомат с консенсусом, в котором транзакции приводят к глобальному переходу состояния и изменению прав владения цифровыми активами. Переходы состояния ограничены правилами консенсуса, что в итоге позволяет всем участникам сойтись на общем состоянии системы, то есть достичь консенсуса, после майнинга нескольких блоков.

Ethereum тоже представляет собой распределенный конечный автомат. Однако он отслеживает не только состояние владения средствами, но и состояние хранилища общего назначения, в котором могут находиться любые данные вида «ключ-значение». Доступ к этим произвольным значениям осуществляется по ключу. Например, значение «Блокчейн второго поколения» имеет ключ «Название лекции». Это в каком-то смысле напоминает модель хранения данных в памяти, которая применяется в большинстве компьютеров общего назначения.

Ethereum обладает памятью, в которой хранятся как данные, так и код. Чтобы отследить изменения, которые в ней происходят, используется блокчейн. Как и обычный компьютер с хранимыми программами, Ethereum может загружать код в свой конечный автомат и запускать его, сохраняя итоговые изменения состояний в блокчейн. Два ключевых отличия от компьютеров общего назначения состоят в том, что изменения состояний в Ethereum выполняются в соответствии с правилами консенсуса, а само хранение состояний распределено по глобальной сети.

Таким образом, Ethereum является утвердительным ответом на следующий вопрос: «Возможно ли создать глобальный компьютер, работающий в соответствии с алгоритмом консенсуса, когда существует возможность отслеживания любого произвольного состояния и программирования конечного автомата?»

Полнота по Тьюрингу

Начав знакомство с Ethereum, вы также сразу же столкнетесь с термином «полнота по Тьюрингу». Считается, что Ethereum, в отличие от Биткоина, является тьюринг-полной системой. Но что именно это означает?

Этот термин назван в честь английского математика Алана Тьюринга, которого считают отцом информатики. В 1936 году он создал математическую модель компьютера. Она состояла из конечного автомата, который манипулирует символами, считывая и записывая их в последовательную память, напоминающую бумажную ленту неограниченной длины. Эта концепция послужила математическим фундаментом, который позволил дать ответы на вопросы об универсальной вычислимости, то есть все ли проблемы можно решить. Он доказал, что существуют классы задач, которые нельзя вычислить.

Алан Тьюринг пошел дальше и определил тьюринг-полную систему как такую, с помощью которой можно симулировать машину Тьюринга. Такая система называется универсальной машиной Тьюринга.

💡 **Полнота по Тьюрингу** — характеристика исполнителя в теории вычислимости, означающая возможность реализовать на нем любую вычислимую функцию.

Платформу Ethereum можно назвать универсальной машиной Тьюринга, так как она способна исполнять программы из памяти в конечном автомате, а чтение и запись данных в память делает ее тьюринг-полной. Ethereum может вычислить любой алгоритм, поддерживаемый любой машиной Тьюринга, при условии достаточного объема памяти.

Новаторская идея Ethereum заключалась в объединении вычислительной архитектуры общего пользования с хранимыми в памяти компьютера программами и децентрализованного блокчейна. Это позволило создать глобальный компьютер с единичным распределенным состоянием. Программы в Ethereum выполняются «везде», но продуцируют общее состояние сети, которое безопасно защищено правилами консенсуса.

Проблемы полноты по Тьюрингу

Читая о тьюринг-полноте Ethereum, вы могли прийти к выводу о том, что это некая возможность, которой не хватает тьюринг-неполным системам. Но все совсем наоборот. Полнота по Тьюрингу достигается очень просто. На самом деле простейший тьюринг-полный конечный автомат, известный на сегодня, имеет четыре состояния и оперирует шестью символами, а определение его состояния занимает всего 22 инструкции.

Полнота по Тьюрингу является очень опасной, особенно в системах с открытым кодом, таких как публичные блокчейны. Это связано с проблемой остановки выполнения вычислений. В своё время, Алан Тьюринг доказал невозможность решения этой проблемы. Нельзя определить, остановится ли когда-нибудь выполнение произвольной программы, зная только ее входные данные.

Например, современные принтеры являются тьюринг-полными, из-за чего они могут зависнуть, если послать им на печать определенные файлы. Тьюринг-полнота платформы Ethereum означает, что в ней можно выполнить программу любой сложности. Однако такая гибкость создает неоднозначные проблемы с безопасностью и управлением ресурсами. «Зависший» принтер можно выключить и включить заново. С публичным блокчейном этого сделать нельзя.

Последствия полноты по Тьюрингу

Итак, Тьюринг доказал, что компьютерная симуляция программы не может определить, завершится она или нет. Проще говоря, мы не в силах предсказать путь выполнения программы без ее фактического выполнения. Тьюринг-полные системы могут находиться в бесконечном цикле. Ввиду сложного взаимодействия между начальными условиями и кодом, бесконечные циклы могут возникать совершенно неожиданно и непреднамеренно.

В Ethereum это создает проблемы: каждый участвующий клиент должен проверить каждую транзакцию, запуская все смарт-контракты, которые вызывают исполнение транзакций. Но, как доказал Тьюринг, без выполнения операций по смарт-контракту блокчейн-сеть Ethereum не может предсказать, завершится ли он и как долго он будет исполняться. Случайно или преднамеренно, вы можете создать такой смарт-контракт, который в случае проверки его клиентами никогда не завершит свое выполнение. Это, в сущности, является DoS-атакой.

Конечно же, между практически мгновенным и бесконечным выполнением находится несметное количество «скверных и прожорливых» программ, занимающих много памяти, перегревающих центральный процессор и просто тратящих попусту ресурсы. В случае с глобальным компьютером это означает, что такая программа будет злоупотреблять ресурсами сети. Каким же образом платформа Ethereum ограничивает ресурсы, используемые смарт-контрактами, если она не может заранее предсказать их потребление?

Чтобы ответить на этот вопрос, в Ethereum был введен механизм измерения под названием газ (англ. gas). По мере выполнения смарт-контракта EVM (Ethereum Virtual Machine) тщательно учитывает каждую инструкцию. У всех инструкций есть заранее установленная стоимость в единицах газа. Когда транзакция инициирует выполнение смарт-контракта, в ней содержится значение объема газа, который устанавливает максимальное количество используемых ресурсов, доступных для потребления данным смарт-контрактом. Если объем газа, израсходованного при вычислении, превысит заданный лимит, EVM прервет его выполнение. Благодаря этому механизму, Ethereum поддерживает тьюринг-полные вычисления, ограничивая при этом расход ресурсов, доступных для потребления той или иной исполняемой программой.

Но теперь встает следующий вопрос: как получить газ для оплаты вычислений на глобальном компьютере Ethereum? Вы не найдете газ ни на одной из криптобирж. Его можно купить только с помощью транзакции внутри сети и только за эфир. Эфир необходимо послать вместе с транзакцией, и он должен быть специально

зарезервирован для покупки газа по приемлемой цене. Как и в реальном мире, цена газа не является фиксированной. Газ покупается для транзакции; если после выполнения вычислений какая-то его часть остается неиспользованной, она возвращается отправителю транзакции.

Децентрализованные приложения

Проект Ethereum начинался как блокчейн общего пользования, который можно запрограммировать для различных целей. Но очень быстро это видение расширилось до платформы программируемых децентрализованных приложений (decentralized applications или DApps). DApp-приложения имеют более широкий охват перспектив их применения, по сравнению со смарт-контрактами. Они включают в себя, как минимум, сам смарт-контракт и пользовательский веб-интерфейс. В целом DApp — это веб-приложение, построенное на основе открытых, децентрализованных, пиринговых инфраструктурных сервисов.

DApp состоит по меньшей мере из:

- смарт-контрактов на блокчейне;
- пользовательского веб-интерфейса.

Помимо этого, такое приложение может включать в себя следующие компоненты:

- децентрализованный (P2P) протокол и платформу для хранения данных;
- децентрализованный (P2P) протокол и платформу для обмена сообщениями.

Интернет третьего поколения

Теперь вспомним курс GeekBrains о Web 3.0. В 2004 году стал популярным термин «Web 2.0», описывающий развитие Интернета в сторону пользовательского контента, динамических интерфейсов и интерактивности. Это не техническая спецификация, а скорее идея нового направления в мире веб-технологий.

Концепция децентрализованных приложений стала новой естественной вехой в развитии Всемирной паутины, делая децентрализацию и пиринговые протоколы неотъемлемой частью веб-приложений. Термин, который описывает эту эволюцию, называется web3 и означает третью «версию» web. Изначально, предложенный Гэвином Вудом, он представляет новое видение и переход от сайтов с централизованным владением и управлением, к приложениям, основанным на децентрализованных протоколах.

В отличие от традиционной разработки и развертывания программного обеспечения, DApp не нужно размещать на внутреннем сервере. Код приложения внедряется в транзакции, которые затем отправляются на майнинговые узлы в сети Ethereum. Разработчики могут свободно кодировать любое приложение и развертывать его. Сеть сама выполняет код, а также проверяет транзакции и выдает выходные данные. Но если бы этот механизм работал без затрат, сеть мгновенно рухнула бы под валом мусорных транзакций. Как мы уже рассмотрели ранее, с каждой транзакцией блокчейна связаны расходы на газ, поэтому написание мусорного кода и его размещение в сети Ethereum становится очень затратным делом.

На семинарах вы рассмотрите JavaScript-библиотеку web3.js, которая связывает JavaScript-приложения, запущенные в браузере, с блокчейном Ethereum. Библиотека web3.js также включает в себя интерфейс для пирингового сетевого хранилища под названием Swarm и пирингового сервиса обмена сообщениями Whisper. Эти три компонента, включая запускаемую в веб-браузере библиотеку JavaScript, представляют собой полноценный пакет разработки, который позволяет создавать DApp.

Культура разработки в Ethereum

До сих пор мы обсуждали, как Ethereum отличается от блокчейнов первого поколения в плане назначения и технологий. Но Ethereum также имеет очень специфическую культуру разработки. В Биткоине разработка подчиняется консервативным принципам: все изменения тщательно исследуются, чтобы не нарушить работу ни одной из существующих систем. Большинство изменений принимаются только в случае, если они сохраняют обратную совместимость. Имеющиеся клиенты могут либо обновить ПО, либо остаться на старой версии.

Для сравнения, культура разработки в сообществе Ethereum сосредоточена на будущем, а не на прошлом. Среди программистов популярна полусерьезная мантра, «двигайся быстро, круши все на своем пути». Если изменение необходимо, оно реализуется даже в случае, если для этого требуется пересмотреть предыдущие допущения, нарушить совместимость или заставить всех клиентов обновиться. Культура разработки в Ethereum характеризуется высокими темпами инноваций и развития, а также готовностью развертывать прогрессивные улучшения, даже если для этого потребуется пожертвовать обратной совместимостью.

💡 15 сентября 2022 года состоялось крупнейшее обновление в блокчейне Ethereum под названием The Merge. Алгоритм консенсуса Proof of Work заменил Proof of Stake — более экологичный и децентрализованный.

Это означает, что вы как разработчик должны сохранять гибкость и быть готовыми к переформатированию своей инфраструктуры в случае изменения каких-либо исходных концепций. Одной из существенных проблем, которые стоят перед разработчиками в Ethereum, является противоречие между развертыванием кода в неизменяемую систему и разработкой платформы, которая продолжает эволюционировать. Вы не можете просто взять и «обновить» свои смарт-контракты. Вы должны быть готовы начать все сначала и развернуть их новые версии с последующей миграцией пользователей, приложений и средств.

Как ни странно, это также означает, что задача по созданию более автономных систем с децентрализованным управлением все еще не достигнута. Автономия и децентрализация требуют более стабильной платформы, чем та, которую Ethereum может предложить сейчас и на протяжении нескольких следующих лет. Чтобы платформа «эволюционировала», вы должны быть готовы к перезапуску своих смарт-контрактов, а это подразумевает определенную степень контроля над ними.

Но, с другой стороны, платформа Ethereum очень быстро движется вперед. Рано или поздно разработка платформы Ethereum замедлится, и ее интерфейсы зафиксируются. Но пока этого не случилось, лучше не оставаться на месте, потому что никто не собирается вас ждать.

Философия блокчейна Ethereum

Блокчейн Ethereum позаимствовал многие концепции из Bitcoin Core, поскольку он выдержал испытание временем. Он разработан на основе другой философии. Разработка Ethereum осуществлялась по следующим принципам:

- **Простота.** Блокчейн Ethereum спроектирован максимально простым, чтобы его было легко изучить и использовать как платформу для децентрализованных приложений. Сложности в реализации сведены к соблюдению минимальных требований на уровне консенсуса.
- **Свобода разработки.** Платформа Ethereum предназначена для поощрения любой децентрализации на основе блокчейна, не ограничивает и не поддерживает какие-либо конкретные варианты использования. Эта свобода безгранична — вплоть до того, что разработчик может закодировать

бесконечный цикл в умном контракте и запустить его. Очевидно, что цикл будет работать до тех пор, пока владелец контракта платит комиссию за транзакцию, и цикл, в конце концов, завершится, когда газ на счете закончится.

- **Отсутствие понятий о наборе функций.** Чтобы сделать систему максимально абстрактной, Ethereum не имеет встроенных функций для разработчиков. Вместо этого Ethereum обеспечивает поддержку тьюринг-полного языка и позволяет пользователям разрабатывать свои собственные функции так, как они хотят. В Ethereum можно запрограммировать все:начиная с базовых функций, заканчивая полноценными сценариями обработки транзакций.

Использование EVM

Вероятно, у вас уже есть криптокошелек, и вы отправили и получили эфир. До сих пор вы обращались к платформе Ethereum как к криптовалюте. Но это была лишь вершина айсберга. На самом деле криптовалюта — это лишь побочная функция децентрализованного глобального компьютера, которым является Ethereum. Эфир предназначен для осуществления оплаты за выполнение смарт-контрактов — компьютерных программ, работающих в эмулируемом компьютере под названием EVM (Ethereum Virtual Machine — виртуальная машина Ethereum).

EVM представляет собой глобальный единый компьютер, распределенный по всему миру. Каждая нода в сети Ethereum содержит локальную копию EVM, которая проверяет выполнение контрактов, а блокчейн Ethereum записывает изменяющееся состояние этого глобального компьютера по мере того, как тот обрабатывает транзакции и исполняет смарт-контракты.

При разработке EVM подразумевались следующие цели:

- **Простота.** Идея заключалась в том, чтобы сделать EVM как можно более простой на основе кодов языка низкого уровня. Вот почему количество низкоуровневых кодов операций и типов данных сведено к минимуму, но до такой степени, что с их помощью все еще можно составлять логические конструкции произвольной сложности. Всего предусмотрено 160 инструкций, из которых 65 логически различаются.

- **Абсолютный детерминизм.** Выполнение инструкций с одним и тем же набором входных данных должно давать одинаковый набор данных на выходе. Эго помогает поддерживать целостность EVM без какой-либо

неопределенности. Детерминизм вместе с понятием *вычислительный шаг* помогают приблизительно оценить расход газа, перед выполнением кода.

- **Оптимизация пространства.** В децентрализованных системах экономия места в хранилище является самой большой проблемой. Вот почему сборка EVM остается максимально компактной.
- **Оптимизации для специальных операций.** Машина EVM оптимизирована для выполнения некоторых специальных операций. Например, для конкретных типов арифметических операций в криптографии, чтения блоков или данных транзакций, взаимодействия с «состояниями» и так далее.
- **Простая безопасность.** В определенном смысле цена на газ помогает гарантировать, что EVM не подвержена взлому. Если бы газ не имел ценности, злоумышленники могли бы атаковать систему всеми возможными способами. В то время, как почти каждая операция на EVM требует затрат на газ, легко найти хорошую модель стоимости газа для добропорядочных пользователей.

Простыми словами, EVM записывает умные контракты и данные в блокчейн и выполняет транзакции и коды контрактов. То есть EVM служит средой выполнения для смарт-контрактов Ethereum и обеспечивает безопасное выполнение кода. Ethereum может измениться только после успешного выполнения инструкций в среде EVM.

Если вы не подключите EVM к основной сети, она будет исправно работать на изолированном узле. Запуск EVM в безопасной «песочнице» может использоваться для тестирования смарт-контрактов. Это облегчает разработку качественных, надежных и готовых к работе контрактов. Об этом мы подробнее поговорим позже.

Учетные записи Ethereum

То, что у вас, вероятно, уже создано в кошельке MetaMask, называется учетной записью с внешним владельцем (externally owned account или EOA). Ее характерной чертой является наличие приватного ключа, с помощью которого вы можете управлять доступом к средствам и контрактам.

Но, как вы уже могли догадаться, это не единственный тип учетных записей. Альтернативой служат учетные записи контрактов. В отличие от обычных EOA, они содержат код смарт-контрактов. Кроме того, у учетных записей контрактов нет приватного ключа. Их владение определяется логикой смарт-контрактов — программой, записанной на блокчейн Ethereum в момент создания учетной записи контракта и выполняемой машиной EVM.

Точно так же, как и у EOA, у контрактов есть адреса, они могут отправлять и принимать эфир. Но когда контракт выступает адресатом транзакции, он выполняется в EVM и использует ее данные в качестве ввода. Помимо эфира, транзакции могут содержать информацию, определяющую то, какую именно функцию контракта следует запустить и какие параметры ей будут переданы. Это позволяет транзакциям вызывать функции внутри контрактов.

На семинаре вы научитесь создавать, пополнять и использовать смарт-контракт с помощью вашего кошелька MetaMask и тестового эфира в тестнете.

Клиенты Ethereum

Клиент Ethereum — это приложение, которое реализует спецификацию этой платформы и взаимодействует с помощью пиринговой сети с другими клиентами. Чтобы быть совместимыми между собой, разные клиенты должны соответствовать эталонной спецификации и стандартизированным коммуникационным протоколам. Они разрабатываются разными командами на разных языках программирования, но при этом умеют «общаться» по единому протоколу с соблюдением одних и тех же правил. Таким образом, все они могут функционировать и взаимодействовать с одной и той же сетью Ethereum.

> 💡 Ethereum является проектом с открытым исходным кодом, а исходный код всех клиентов доступен под «свободными» опенсорс-лицензиями и может быть загружен и использован для любых целей.

Как уже говорилось ранее, Ethereum имеет формальную техническую спецификацию под названием Yellow Paper, или «желтая бумага». Благодаря этому, существует целый ряд клиентов, которые разрабатываются независимо друг от друга, но являются при этом совместимыми между собой. Ethereum имеет большое разнообразие клиентских реализаций, которые работают в блокчейн-сети, по сравнению с другими блокчейнами, что обычно рассматривается как хороший признак развития.

На сегодня существует шесть основных реализаций протокола Ethereum, написанных на шести разных языках:

- Parity, написанная на Rust.
- Geth, написанная на Go.
- cpp-ethereum, написанная на C++.

- Pyethereum, написанная на Python.
- Mantis, написанная на Scala.
- Harmony, написанная на Java.

На семинаре вы рассмотрите один из самых распространенных клиентов — Geth. Это клиент, написанный на языке Go, который активно разрабатывается организацией Ethereum Foundation и в связи с этим считается «официальной» реализацией клиента Ethereum. Вы узнаете, как его установить и настроить, а также создадите genesis-блок.

Запуск полноценной ноды

Работоспособность блокчейна, его устойчивость к нагрузкам и цензуре зависят от наличия множества независимо работающих и географически рассредоточенных полноценных нод, или, другими словами, узлов сети. Каждая полноценная нода может предоставить новым нодам данные о блоках и тем самым помочь им с инициализацией, она также может выступать в качестве оператора для авторитетной и независимой верификации всех транзакций и контрактов.

Запуск полноценной ноды требует определенных аппаратных ресурсов и пропускной способности сети. Для установки такой ноды необходимо загрузить более 200 гигабайт данных и сохранить их на локальном жестком диске. Нагрузка по обработке данных каждый день будет стремительно расти, по мере добавления новых транзакций и блоков.

Разработка для Ethereum не требует запуска полноценной ноды в мейннет. Практически все, что вам нужно, можно сделать с помощью ноды тестнета, локального закрытого блокчейна, такого как Ganache, или облачного клиента Ethereum, предоставляемого такими поставщиками услуг, как Infura.

В качестве еще одной альтернативы, вы можете воспользоваться удаленным клиентом, который не хранит локальную копию блокчейна и не занимается проверкой блоков и транзакций. Такие клиенты предоставляют функции кошелька и способны создавать и распространять транзакции. Удаленные клиенты можно использовать для подключения к существующим сетям, публичным блокчейнам, тестнетам с публичным или ограниченным доступом или приватным локальным блокчейнам. На практике вы, скорее всего, будете использовать такой удаленный клиент как MetaMask или MyEtherWallet.

💡 Термины «удаленный клиент» и «кошелек» являются взаимозаменяемыми, хотя между ними есть некоторые отличия. Помимо функций для работы с транзакциями, характерных для кошелька, удаленный клиент обычно предоставляет еще API-интерфейс (например, web3.js).

Не путайте концепцию удаленного кошелька Ethereum с его «легким клиентом». «Легкие клиенты» проверяют заголовки блоков, а также используют доказательства Меркла, для проверки включения транзакций в блокчейн и определения их эффектов, что делает их почти такими же безопасными, как полноценные ноды. Удаленные клиенты Ethereum, с другой стороны, не проверяют заголовки блоков и транзакции. Они полностью полагаются на полноценного клиента, который предоставляет им доступ к блокчейну, и тем самым значительно теряют свойства функциональности в части безопасности и анонимности.

Преимущества и недостатки полноценной ноды

Полноценная нода вносит свой вклад в функционирование сети, к которой вы ее подключаете, но влечет за собой незначительные или умеренные расходы. Давайте обсудим некоторые преимущества и недостатки запуска полноценной ноды.

Преимущества:

- нода поддерживает устойчивость сетей на основе Ethereum;
- авторитетно проверяет все транзакции;
- способна взаимодействовать с любым контрактом в публичном блокчейне без посредников;
- может напрямую деплоить контракты в публичную блокчейн-сеть;
- может запрашивать для чтения состояние по учетным записям и контрактам в автономном режиме;
- может обращаться к блокчейну, не выдавая сторонним лицам сведений о том, что именно вы читаете.

Недостатки:

- нода требует значительных и постоянно растущих аппаратных и сетевых ресурсов;
- при первом запуске на полную синхронизацию может уйти несколько дней;

- чтобы поддерживать синхронизацию, необходимо обслуживание, обновление и подключение к сети.

Преимущества и недостатки публичной тестовой сети

Независимо от того, является ли ваша нода полной, вы, скорее всего, будете подключать его к публичной тестовой сети. Давайте обсудим положительные и отрицательные стороны таких сетей.

Преимущества:

- Нода тестовой сети должна синхронизироваться и хранить значительно меньший объем данных — около 10 Гбайт, в зависимости от типа сети.
- Нода тестовой сети может полностью синхронизироваться за несколько часов.
- Для развертывания контрактов и выполнения транзакций нужен тестовый эфир, который ничего «не стоит». Его можно получить бесплатно, используя несколько faucet контрактов.
- Тестнеты являются публичными блокчейнами с множеством пользователей и контрактов, работающих в режиме реального времени.

Недостатки:

- В тестовой сети нельзя использовать «настоящие» деньги. Она работает на тестовом эфире, следовательно, вы не можете проверить свою безопасность в контексте реальных угроз, так как вы ничем не рискуете.
- Некоторые аспекты публичного блокчейна нельзя реалистично проверить в тестнете. Например, уплата комиссии является обязательной при отправке транзакций, но в тестовой сети она не воспринимается серьезно, так как газ в ней бесплатный. Более того, тестнеты не испытывают проблем с перегруженностью в отличие публичного мейннета.

Преимущества и недостатки локальной симуляции блокчейна

Во многих случаях для тестирования лучше всего запустить единственный экземпляр приватного блокчейна. Проект Ganache является одной из наиболее популярных симуляций, внутри которой можно организовать взаимодействие без

каких-либо других участников. Он имеет множество преимуществ и недостатков, характерных для публичного тестнета, но у него есть и некоторые отличия.

Преимущества:

- У него отсутствует синхронизация и минимальный объем данных на диске. Первый блок вы формируете самостоятельно.
- Не требуется получать тестовый эфир. Вы сами «вознаграждаете» себя за майнинг и затем можете использовать эту награду для тестирования.
- Нет других пользователей сети, вы — единственный пользователь.
- В сети находятся только те контракты, которые вы развернули после запуска сети.

Недостатки:

- Отсутствие других пользователей означает, что сеть ведет себя не так, как публичный блокчейн. Нет конкуренции за место для транзакций или то, какая из них будет первой.
- Отсутствие других майнеров делает процесс майнинга более предсказуемым. В связи с этим вам не удастся протестировать некоторые сценарии, которые могли возникнуть в публичном блокчейне.
- Отсутствие «чужих» контрактов в сети означает, что вам самостоятельно придется развернуть все необходимое для тестирования, включая зависимости и библиотеки контрактов.
- Вы не сможете воссоздать некоторые публичные контракты (например, контракт DAO) и их адреса для тестирования определенных сценариев.

Заключение

Подводя итог, давайте дадим финальное определение Ethereum. Это открытая, глобально децентрализованная вычислительная инфраструктура, которая выполняет программы, так называемые смарт-контракты. Она использует блокчейн для синхронизации и хранения изменений состояния системы, а также криптовалюту под названием эфир (ETH) для измерения и ограничения стоимости вычислительных ресурсов. Платформа Ethereum позволяет разработчикам создавать мощные децентрализованные приложения со встроенными экономическими функциями. Она предоставляет высокие доступность, подотчетность, прозрачность и нейтральность, но при этом ограничивает или устраняет цензуру и снижает определенные риски, связанные с контрагентами.

На этой лекции мы изучили основные элементы платформы Ethereum и описали принципы разработки в них. Мы определили концептуальные различия между блокчейном Ethereum и блокчейном Биткоина. Разобрались в том, как Ethereum и другие блокчейны второго поколения, такие как BNB Chain и Solana, облегчают разработку и запуск различных приложений на одной платформе. Также мы ближе познакомились с умными контрактами, которые децентрализованно выполняются на виртуальной машине.

Ethereum стал первым блокчейном второго поколения и, как и Биткоин, стал образцом для появления множества подобных проектов. На следующей лекции мы рассмотрим их недостатки и варианты решения, предложенные в блокчейнах третьего поколения.

Дополнительные материалы

1. Ethereum.org
2. «Осваиваем Ethereum. Создание смарт-контрактов и децентрализованных приложений», Андреас Антонопулос, Гэвин Вуд, 2022 год.
3. «Больше денег: что такое Ethereum и как блокчейн меняет мир» | Бутерин Виталик, 2023 год.
4. Статья Виталика Бутерина, «Предыстория Ethereum», 2017 год.
5. Обновление The Merge: как изменится Ethereum с переходом на Proof of Stake

Использованная литература

1. «Блокчейн. Руководство для начинающих разработчиков», Сингхал Бикрамадитья, Панда Прияншу Сехар, Дамеджа Гаутам, 2020 год.
2. «Осваиваем Ethereum. Создание смарт-контрактов и децентрализованных приложений», Андреас Антонопулос, Гэвин Вуд, 2022 год.