Санкт-Петербургский государственный университет

Математико-механический факультет
Кафедра Системного программирования

Андреев Сергей Иванович

Развитие платежной системы для единого взаимодействия с различными реализациями blockchain

Курсовая работа

Научный руководитель:
ст. преп. Кириленко Я. А.

Санкт-Петербург
2020

Оглавление

Введение

Криптовалюта — разновидность цифровой валюты, базирующаяся на идеях криптографии и децентрализованного контролирующего аппарата. На момент 2020 года насчитывается более 4000 различных криптовалют [6], в основу большой части которых заложена технология blockchain. Blockchain предоставляет из себя децентрализованную распределённую базу данных, хранящую выстроенную специальным образом непрерывную последовательность блоков. Эта технология обеспечивает безопасное хранение истории всех денежных операций в системе. Децентрализованный характер системы, прозрачность и надежность операций делают криптовалюты все популярнее с течением времени.

Постоянно развивающиеся криптовалюты различны в реализации, и работа с каждой из них имеет отличительные особенности. Это усложняет взаимодействие с различными криптовалютами и создает потребность в едином криптоплатежном шлюзе, инкапсулирующем в себе механизмы взаимодействия с различными криптовалютами. Прототип такого инструмента был разработан студентом кафедры Системного программирования в 2019 году [23], однако, данная реализация поддерживает только три криптовалюты и не поддерживает работу с платформой Ethereum. Данная платформа предоставляет не только еще одну криптовалюту, но и возможность работы со смарт-контрактами [5] — алгоритмами, предназначенными для формирования, контроля и предоставления информации о владении чем-либо. В частности, смарт-контракты позволяют создавать ERC токены — криптовалюту на базе блокчейна Ethereum. Некоторые их таких токенов достаточно популярны, например, токен Tether USDT активно используется на торговых биржах и вместе с Ethereum входит в топ-4 криптовалют по рыночной капитализации [18]. Это усиливает актуальность интеграции криптовалюты Ethereum и Erc-20 токенов в систему.

В настоящий момент для предоставления финансовых операций криптовалюту уже используют такие компании как Mastercard, Fidelity,

Northern Trust, Ripple и многие другие [21], что иллюстрирует актуальность использования криптовалюты банками и прочими компаниями в настоящее время.

1. Постановка задачи

Целью данной работы является разработка системы, которая позволит участникам финансовых отношений производить стандартные операции с различными криптовалютами, базирующимися на технологии blockchain.

Для достижения цели были поставлены следующие задачи:

- Ознакомиться с предметной областью
- Провести анализ существующих решений
- Доработать существующую систему
  - Реализовать поддержку криптовалюты Ethereum и Erc-20 токенов
  - Реализовать интеграционное тестирование полученной системы

2. Введение в предметную область

2.1. Blockchain

Blockchain предоставляет из себя децентрализованную распределённую базу данных [22]. Содержимое базы данных — выстроенная специальным образом последовательность блоков, которую хранит каждый узел сети. Блок содержит набор транзакций, хеш предыдущего блока, некоторые дополнительные данные, отличающиеся у различных реализаций blockchain, и хеш от всех данных, содержащихся в этом же блоке. Попытка изменения истории операций участником сети изменит хеш блока, содержащего информацию об этих операциях, и сделает всю последующую цепочку недействительной.

2.2. Ethereum

Blockchain Ethereum обладает рядом особенностей [9]. Адресное пространство разделяют два типа аккаунтов, то есть пар адрес - состояние: внешние аккаунты и самрт-контракты. Внешние аккаунты контролируются парами приватных-публичных ключей, а контракты контролируются своим кодом [3].

Транзакция представляет из себя сообщение от одного аккаунта к другому. Транзакции между внешними адресами обеспечивают только перевод средств, и не могут содержать дополнительной пользовательской информации.

Среди параметров посылаемой транзакции следует особо выделить параметры data и nonce. Параметр data используется для передачи байт-кода контракта или закодированных данных о вызываемом методе контракта и передаваемых аргументах. Параметр nonce — это счетчик отправленных с адреса инициатора транзакций. Для успешной обработки транзакции переданный в неё параметр nonce должен быть строго на единицу больше количества отправленных (и включенных в какой-либо блок) с адреса инициатора транзакций. Таким образом обеспечивается защита от повторной отправки однажды подписанной аккаунтом тран-

закции.

Транзакция с байт-кодом специального формата на нулевой адрес инициирует создание смарт-контракта: майнер выполняет байт-код из транзакции на тьюринг-полной Ethereum Virtual Machine (EVM) [7], причем выполнение каждой инструкции требует некоторой комиссии, взимаемой с инициатора транзакции.

Транзакция на адрес контракта инициирует выполнение заданного метода контракта. Причем вместе с транзакцией передается закодированная строка с хешем от имени метода, а также список аргументов. Выполнение кода производят майнеры на EVM, что также может требовать некоторой комиссии от инициатора транзакции.

С помощью комиссий за исполнение инструкций в evm регулируется проблема остановки и бесконечных зацикливаний.

2.3. Erc-20 токены

Erc-20 токен в Ethereum представляет из себя смарт-контракт, удовлетворяющий некоторым правилам. Данные правила, а также прочие договоренности платформы Ethereum, определяются страндартами ERC (Ethereum Request for Comments). Соответствующий стандарту Erc-20 [11] контракт, должен реализовывать специальный интерфейс, то есть иметь доступные любому участнику сети функции. Среди них: функции для получения информации о балансе аккаунта и о максимально возможном количестве единиц данного токена, функция для перевода средств, а также функция для предоставления некоторому аккаунту права распоряжаться фиксированной частью средств аккаунта-инициатора. Эти функции позволяют рассматривать токены как электронный актив, который так или иначе можно переводить с аккаунта на аккаунт. Стандарт серьезно упрощает взаимодействие с различными Erc-20 контрактами, однако, их внутренняя логика может различаться. Например, контракт может предусматривать дополнительной сбор комиссии в токенах при переводе или прочих операциях.

3. Обзор существующих решений

На данный момент существует ряд различных криптоплатежных шлюзов. Такие сервисы полезны для компаний и частных предпринимателей, желающих использовать криптовалюту для предоставления своих услуг, однако, не имеющих достаточно ресурсов и желания для самостоятельного освоения технологии blockchain. Набор некоторых популярных в настоящее время платежных криптовалютных сервисов [1] представлен в таблице 1.

Сервис	Валюты	Кастодиальный	Обслуживание	Open source
GoCoin [1]	6	+	SaaS	-
CoinPayments [2]	92	+	SaaS	-
AlfaCoin [3]	7	+	SaaS	-
CoinsBank [4]	4	+	SaaS	-
Paycoiner [5]	18	+	SaaS	-
Finch [6]	2	-	Self-hosted	+
BTCPay Server [7]	12	-	Self-hosted	+

Таблица 1: Популярные криптовалютные шлюзы

Сравнение проводилось с целью проанализировать наличие и применимость текущих решений с точки зрения затрат и возможности использования популярных криптовалют.

Выбранные для сравнения параметры позволили проанализировать сервисы на объем поддерживаемых криптовалют, способ хранения средств пользователя, наличие хостинга и доступность исходного кода. Кастодиальные сервисы хранят средства пользователя на своих счетах, предоставляя ему возможность их вывода с некоторой комиссией. Большая часть сервисов имеет закрытый исходный код, что не позволяет рассмотреть детали их реализации. За услуги хостинга с пользователя мо-

[1] https://gocoin.com/
[2] https://www.coinpayments.net/
[3] https://www.alfacoins.com/
[4] https://coinsbank.com/
[5] https://paycoiner.com/
[6] https://finchtech.io/
[7] https://btcpayserver.org/

жет взиматься некоторая плата, что может быть критично при масштабировании системы. Также self-hosted решения безопаснее включающих хостинг решений.

Finch является некастодиальным self-hosted продуктом с открытым исходным кодом, однако он поддерживает только две криптовалюты.

BTCPay Server поддерживает большее количество криптовалют, однако, помимо Bitcoin, все они являются производными от Bitcoin и только 5 из них входят в топ-100 и только одна из них входит в топ-20 криптовалют по рыночной капитализации.

4. Поддержка Ethereum и Erc-20

Дорабатываемый в контексте данной работы прототип состоит из двух модулей. Модуль Core отвечает за финансовые операции в контексте платежей и счетов. Платеж подразумевает перевод средств в некоторой криптовалюте на заданный адрес с последующим отслеживанием статуса соответствующей транзакции в блокчейне. Выставление счета подразумевает создание условий для определения факта перевода заданной суммы в распоряжение пользователя. Модуль Crypto осуществляет взаимодействие с различными реализациями blockchain. Таким образом, для поддержки системой криптовалюты Ethereum и Erc-20 токенов в рамках данной работы была произведена доработка модуля Crypto, что позволило ему соответствующим образом взаимодействовать с блокченойм Ethereum.

4.1. Взаимодействие с блокченойм Ethereum

Обеспечение взаимодействия с блокченойм Ethereum подразумевает реализацию сервисов, позволяющих

- Получать информацию о транзакциях сети, балансах аккаунтов
- Отправлять платежи в криптовалюте Ethereum и Erc-20 токенах
- Обеспечивать средство идентификации транзакций, направленных на оплату заданного счета
- Обеспечивать доступность переведенных в рамках выставленного счета средств
- Постоянно просматривать транзакции сети и уведомлять сервисы модуля Core о новых транзакциях

4.2. Идентификация направленных на оплату счета транзакций

Для выставления счета необходимо иметь возможность определять, была ли заданная транзакция создана с целью погашения этого счета. Этого можно достичь путем добавления дополнительной информации в транзакцию. Так, например, в транзакциях блокчейна Ripple возможно добавление дополнительных данных [13]. Однако, не в каждом блокчейне в транзакцию можно добавить стороннюю информацию. Например, в блокчейне Bitcoin единственная возможность понять, что средства были переведены для погашения конкретного счета — создание нового адреса, на который будет ожидаться поступление средств.

4.2.1. Ethereum

В блокчейне Ethereum, также как и в блокчейне Bitcoin, в транзакцию нельзя добавить стороннюю информацию [16]. Но в отличие от модели учета средств UTXO в Bitcoin [19], модель учета средств в Ethereum не позволяет использовать средства, имеющиеся на разных доступных адресах, при отправке транзакции. Поэтому после поступления средств на выделенный адрес необходимо перевести их на горячий кошелек – основной системный адрес. Было рассмотрено несколько подходов организации переводов на горячий кошелек.

Переводы без контракта

Самый простой подход — при выставлении счета создавать новый кошелек, адрес которого передавать плательщику. Далее, после обнаружения транзакции на созданный адрес, новой транзакцией переводить средства на горячий кошелек.

Прокси контракт

Другой подход — вместо создания внешнего адреса создавать смарт-контракт, который будет переводить все поступающие на его счет средства на адрес горячего кошелька. В данном случае плательщику сооб-

щается адрес смарт контракта. Плюс такого подхода в том, что инициатор счета оплачивает только создание контракта, а все исполнение кода контракта, в частности, перевод средств с адреса контракта на адрес горячего кошелька оплачивает плательщик. Причем размер данной комиссии меньше, чем размер комиссии за две транзакции между внешними адресами. Но этот подход имеет значительный недостаток: если плательщик вовсе не будет отправлять платежи, то средства на создание контракта будут потрачены напрасно.

Контракт с отложенным созданием

В блокчейне Ethereum смарт-контракт создается путем отправки транзакции с байт-кодом контракта на нулевой адрес. После исполнения байт-кода майнером, будет создан новый аккаунт-контракт, адрес которого будет рассчитан исходя из адреса инициатора транзакции и количества отправленных им транзакций [4]. Таким образом, адрес контракта можно рассчитать до его создания. Это позволяет модифицировать прошлый подход: для выставления счета рассчитать адрес контракта и сообщать его плательщику, при этом не создавая контракт. При обнаружении транзакции на данный адрес создать модифицированную версия обычного прокси-контракта: помимо пересылки входящих средств, сразу после запуска контракт посылает все свои уже имеющиеся средства на горячий кошелек. Однако, и у этого подхода есть свой минус: из-за дополнительной логики модифицированный контракт немного дороже обычного прокси контракта.

Сравнение подходов

Для сравнения подходов в приватной сети Ethereum были созданы описанные контракты. Это позволило рассчитать точную комиссию во внутренних учетных единицах платформы Ethereum — газе. При отправке транзакции участник сети сам указывает цену газа, то есть сколько Wei ($10^{-18} Ether$) он отдает за единицу газа в данной транзакции.

Ниже приведены таблицы, иллюстрирующие размер комиссии для

инициатора счета (табл. 2), плательщика (табл. 3), а также суммарную комиссию (табл. 4) при использовании перечисленных ранее подходов по автоматизации перевода средств на горячий кошелек. В качестве второго параметра рассматривается количество отправляемых платежей. При выставлении счета самое вероятное количество платежей, его покрывающих — 1, однако, в дальнейшем планируется расширить функциональность по выставлению счетов до выдачи адреса постоянному клиенту. В этой ситуации клиент будет на протяжении долгого времени делать переводы на выданный адрес. Поэтому для рассмотрения были выбраны ситуации с 0, 1, 5 И 10 платежами, которые необходимо переадресовать со сгенерированного адреса на горячий кошелек.

Количество переводов	Без контракта	Прокси контракт	Отложенный
0	0	109 595 (0.217$)	0
1	21 000 (0.045$)	109 595 (0.217$)	142 511 (0.282$)
5	105 000 (0.208$)	109 595 (0.217$)	142 511 (0.282$)
10	210 00 (0.4$)	109 595 (0.217$)	142 511 (0.282$)

Таблица 2: Комиссии получателя при переводе на горячий кошелек

Количество переводов	Без контракта	Прокси контракт	Отложенный
0	0	0	0
1	21 000 (0.045$)	29 370 (0.058$)	21 000 (0.041$)
5	105 000 (0.208$)	146 850 (0.291$)	138 480 (0.274$)
10	210 00 (0.4$)	293 700 (0.581$)	285 330 (0.564$)

Таблица 3: Комиссии отправителя при переводе на горячий кошелек

Количество переводов	Без контракта	Прокси контракт	Отложенный
0	0	109 595 (0.217$)	0
1	42 000 (0.083$)	138 965 (0.275$)	163 511 (0.323$)
5	210 000 (0.415$)	256 445 (0.507$)	289 361 (0.573$)
10	420 00 (0.830$)	403 295 (0.798$)	436 211 (0.863$)

Таблица 4: Суммарные комиссии при переводе на горячий кошелек

Помимо точной стоимости в газе в скобках указана эквивалентная ей сумма в долларах США, вычисленная по данным о средней цене газа и курсе Ether на момент 10.04.2020: средняя цена газа: 11.576 GWei ($10^{-9} Ether$), курс Ether: 170,92$

Итак, для реализации переводов на горячий кошелек в ситуации выставленного счета лучше был выбран первый подход, так как в этом случае суммарная комиссия будет минимальна.

4.2.2. Erc-20

Для Erc-20 токенов процесс перевода средств на горячий кошелек несколько усложнен, потому что после получения токенов созданный аккаунт не сможет самостоятельно инициировать перевод токенов на горячий кошелек: на его счету не будет Ether, необходимого для этого. На рисунке 1 схематично показан процесс перевода токенов на горячий кошелек.

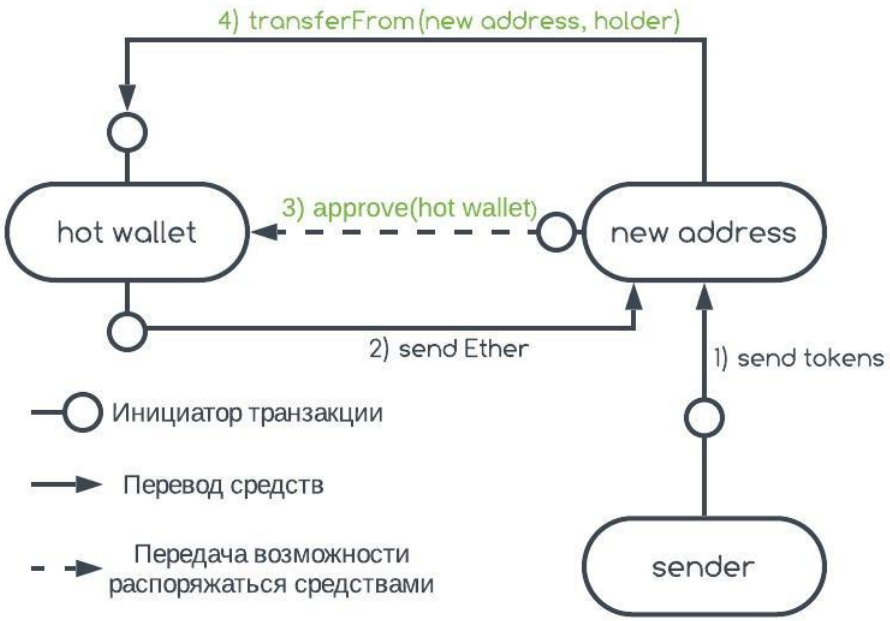

Рис. 1: Процесс перевода токенов на горячий кошелек

Для организации процесса используются функции, которые должен реализовывать каждый смарт-контракт стандарта Erc-20 (выделены зе-

леным). Система постоянно просматривает появляющиеся в блокчейне транзакции в поиске перевода токенов на созданный для счета аккаунт. Как только перевод обнаружен, совершается перевод небольшой суммы Ether с горячего кошелька на новый аккаунт. После поступления средств с нового аккаунта инициируется транзакция на адрес Erc-20 смарт-контракта, с помощью которой вызывается метод approve. Это позволит отправлять с адреса горячего кошелька транзакции, взаимодействующие со средствами нового аккаунта с помощью вызова у смарт-контракта метода transferFrom. При последующих переводах токенов плательщиком на новый аккаунт операции 2) и 3) больше не потребуются, что позволит сильно уменьшить размер суммарной комиссии.

4.3. Реализация модуля поддержки Ethereum и Erc-20

На рисунке 2 изображено строение модуля поддержки Ethereum и Erc-20. Зеленым цветом выделены сервисы, реализованные в рамках данный работы.

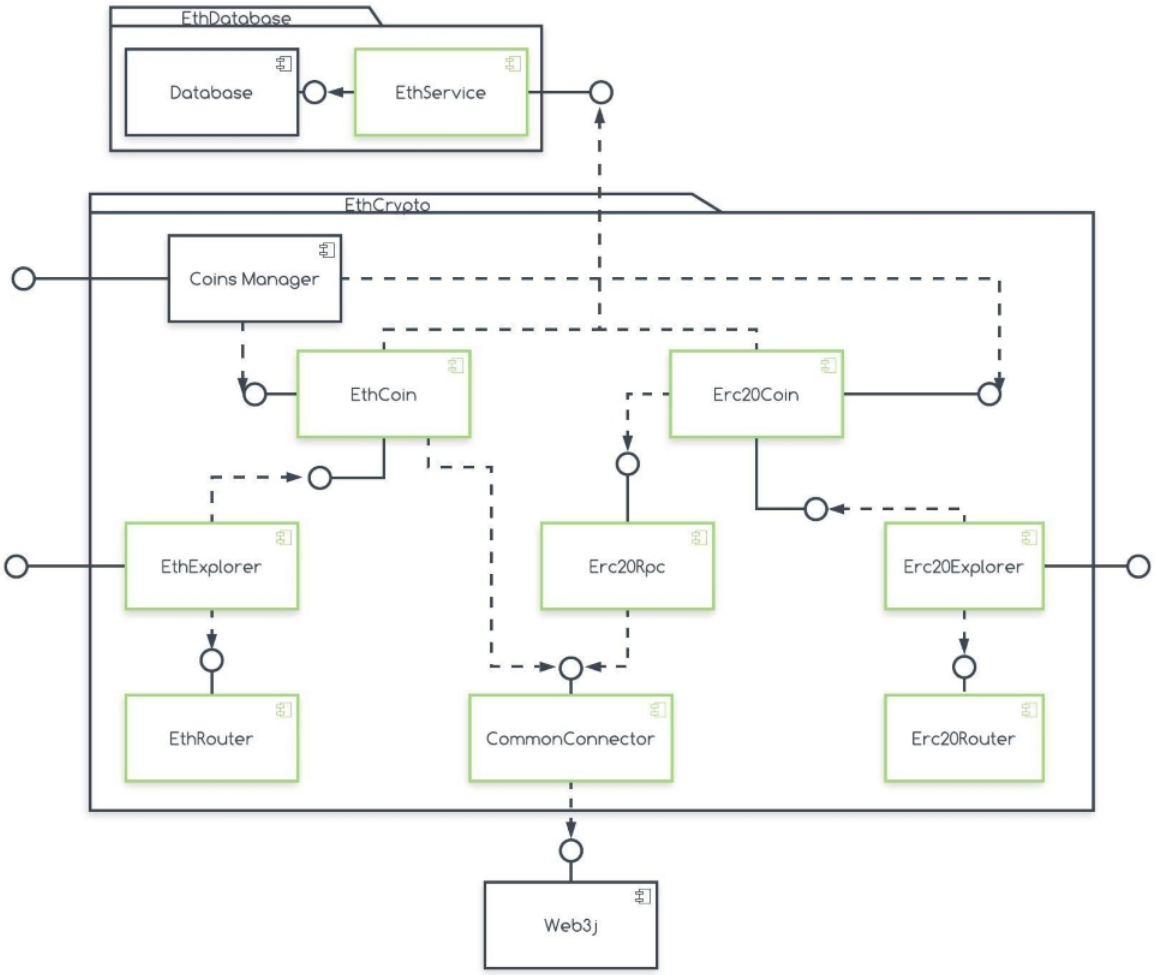

Рис. 2: Архитектура модуля поддержки Ethereum и Erc-20

4.3.1. EthCoin и Erc20Coin

Сервисы EthCoin и Erc20Coin реализуют интерфейс, с помощью которого модуль Core взаимодействует с блокчейном. Он обеспечивают возможность получения баланса, отправки платежа, получения информации о транзакции по ее хешу. В задачи данного сервиса входит ко-

ординирование работы сервиса, взаимодействующего с узлом блокчейна, сервиса Explorer и Router, управление информацией, связанной с конфигурацией подключения к узлу, данными системного аккаунта и прочими параметрами.

4.3.2. CoinConnector и Erc20Rpc

Оба этих сервиса выполняют функцию общения с узлом блокчейна, однако, сервис Erc20Rpc имеет расширенную функциональность, необходимую для работы со смарт-контрактами. Взаимодействие с узлом блокчейна осуществляется с помощью библиотеки web3j [20], которая в свою очередь взаимодействует с последним с помощью JSON-RPC API.

4.3.3. EthExplorer и Erc20Explorer

Для поддержки актуальной информации по поводу состояния выставленного счета необходимо постоянно просматривать все транзакции в сети. Сервисы EthExplorer и Erc20Explorer осуществляют постоянный мониторинг и публикацию транзакций, которые необходимо проверить. EthExplorer осуществляет непрерывный просмотр всех транзакций в сети, в то время как Erc20Explorer прослушивает генерируемые в контрактах блокчейна события [10] о переводе токенов или о передаче права на владение частью средств. Любой контракт стандарта Erc-20 должен создавать эти события при исполнении соответствующих функций.

4.3.4. EthRouter и Erc20Router

При обнаружении транзакций на сгенерированный для счета адрес средства необходимо перевести на горячий кошелек. Эту задачу выполняют сервисы Router, действуя по схеме, описанной ранее.

4.3.5. EthService

Для контроля состояния Ethereum кошельков, которыми владеет система, хранения данных о контрактах, реализующих конкретные то-

кены, был добавлен сервис EthService, взаимодействующий с базой данных. Одной из его задач является контроль счетчка транзакций у каждого из системных аккаунтов — nonce. При отправке транзакции в блокчейне Ethereum необходимо указать ее номер — nonce. Номер должен быть строго на единицу больше количества отправленных аккаунтом транзакций, так в блокчейне реализуется защита от повторной отправки подписанной аккаунтом транзакции [14].

5. Интеграционное тестирование

Для тестирования работы системы с каждой криптовалютой необходимо подключение к узлу сети соответствующего блокчейна. Разные криптовалюты имеют разные средства и подходы по развертыванию тестовой сети, что зачастую является сложным и трудоемким процессом. Поэтому было решено автоматизировать ручную настройку окружения для тестирования системы. Для этого было реализовано решение, развертывающее и настраивающее необходимым образом узлы блокчейна Bitcoin и Ethereum в контейнере, что вдобавок позволило запускать интеграционные тесты на CI. Библиотека testcontainers [17] для java позволяет автоматически запускать контейнер при старте тестов и останавливать его после их окончания. В качестве механизма контейнеризации был выбран Docker, т.к. именно с ним работает библиотека testcontainers, а также Docker есть для Linux, macOS, Windows.

5.1. Настройка окружения для Bitcoin

Для настройки окружения у Bitcoin был использован режим regtest инструмента Testnet [2], который позволяет создать тестовую сеть с возможностью мгновенной генерации блоков. Это значит, что в этой сети можно легко получить любую сумму на произвольном кошельке, а также быстро набрать необходимое количество подтверждений после отправки транзакции.

5.2. Настройка окружения для Ethereum и Erc-20

Для Ethereum и Erc-20 настройка окружения включает в себя создание частной Ethereum-сети с несколькими аккаунтами, имеющими изначальный баланс, а также развертывание тестового Erc-20 контракта и имитацию майнинга. Для развертывания узла был использован инструмент geth [12], с помощью которого в контейнере создавалась частная Ethereum-сеть.

В блокчейне Ethereum используются алгоритм майнинга Ethash [8],

реализующий протокол консенсуса Proof-of-Work. Майнинг, согласно Ethash, требует выбора подмножеств фиксированного ресурса в зависимости от одноразового номера и заголовка блока. Этот ресурс (данные размером в несколько гигабайт) называется DAG (directed acyclic graph). Таким образом, для чтобы иметь возможность получения новых блоков в такой тестовой сети, необходимо обеспечить доступность DAG для узлов тестовой сети. В таком случае, если DAG будет генерироваться внутри контейнера, то подготовка узла может занять достаточно продолжительное время (около 5 минут на обычном ПК), что неприемлемо для тестирования. Если же генерировать DAG заранее и копировать в контейнер, то размер контейнера будет слишком велик, что сделает контейнер достаточно неудобным в использовании. Решением возникшей проблемы стало использование Proof-of-Authority-протокола консенсуса Clique [15] при создании тестовой сети в geth. Протокол Proof-of-Authority подразумевает, что блоки могут создавать только определенные участники сети, таким образом, создание блока не должно сопровождаться подтверждающими вычислениями.

После развертывания сети необходимо создать в ней тестовый Erc-20 контракт. Для этого на языке Solidity был написан контракт, реализующий Erc-20 интерфейс. Помимо стандартной для таких контрактов логики была реализована логика по сбору комиссии в токенах за переводы. После запуска контейнера контракт создается посредством отправки полученного после компиляции байт-кода для EVM в тестовую сеть с помощью транзакции, как только узел будет готов.

Заключение

В рамках данной работы были достигнуты следующие результаты:

- Изучена предметная область
- Поддержана работа с криптовалютой Ethereum
- Поддержана работа с Erc-20 токенами
- Проведено интеграционное тестирование криптовалют Bitcoin, Ethereum и Erc-20 токенов

Список литературы

[1] 17 Best Cryptocurrency Payment Gateways In 2020. — 2020. — URL: https://coinfunda.com/best-cryptocurrency-payment-gateways-bitcoin/ (дата обращения: 28.04.2020).

[2] Bitcoin Testnet. — 2020. — URL: https://bitcoin.org/en/developer-examples#testing-applications (дата обращения: 28.04.2020).

[3] Bruno Skvorc Mislav Javor. Learn Ethereum: The Collection. — sitepoint, 2018.

[4] Buterin Vitalik. EIP-1014 specification. — 2018. — URL: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-1014.md (дата обращения: 28.04.2020).

[5] Contracts. — 2020. — URL: http://ethdocs.org/en/latest/contracts-and-transactions/contracts.html (дата обращения: 28.04.2020).

[6] Cryptocurrency List. — 2020. — URL: https://www.coinlore.com/all_coins (дата обращения: 28.04.2020).

[7] Ethan Buchman Zaki Manian. Notes on the EVM. — 2019. — URL: https://github.com/CoinCulture/evm-tools/blob/master/analysis/guide.md (дата обращения: 28.04.2020).

[8] Ethereum mining. — 2020. — URL: https://github.com/ethereum/wiki/wiki/Mining#so-what-is-mining-anyway (дата обращения: 28.04.2020).

[9] Ethereum white paper. — 2020. — URL: https://github.com/ethereum/wiki/wiki/White-Paper (дата обращения: 28.04.2020).

[10] Events in Solidity. — 2020. — URL: https://solidity.readthedocs.io/en/latest/contracts.html#events (дата обращения: 28.04.2020).

[11] Fabian Vogelsteller Vitalik Buterin. EIP 20: ERC-20 Token Standard. — 2015. — URL: https://eips.ethereum.org/EIPS/eip-20 (дата обращения: 28.04.2020).

[12] Official Go implementation of the Ethereum protocol. — 2020. — URL: https://geth.ethereum.org/ (дата обращения: 28.04.2020).

[13] Ripple Payment Transaction. — 2020. — URL: https://developers.ripple.com/payment.html (дата обращения: 28.04.2020).

[14] Rizgan Yossi. Making Sense of Ethereum Nonce. — 2017. — URL: https://medium.com/kinblog/making-sense-of-ethereum-nonce-sense-3858d5588c64 (дата обращения: 28.04.2020).

[15] Szilágyi Péter. EIP 225: Clique proof-of-authority consensus protocol. — 2017. — URL: https://eips.ethereum.org/EIPS/eip-225 (дата обращения: 28.04.2020).

[16] Tam KC. Transactions in Ethereum. — 2018. — URL: https://medium.com/@kctheservant/transactions-in-ethereum-e85a73068f74 (дата обращения: 28.04.2020).

[17] Testcontainers. — 2015. — URL: https://www.testcontainers.org/ (дата обращения: 28.04.2020).

[18] Top 10 cryptocurrencies by market capitalisation. — 2020. — URL: https://finance.yahoo.com/news/top-10-cryptocurrencies-market-capitalisation-160046487.html (дата обращения: 28.04.2020).

[19] UTXO — Bitcoin.org Developer Guide. — 2020. — URL: https://bitcoin.org/en/blockchain-guide#term-utxol (дата обращения: 28.04.2020).

[20] Web3j. — 2016. — URL: http://web3j.io/ (дата обращения: 28.04.2020).

[21] Исользующие блокчейн компании. — 2020. — URL: https://www.forbes.ru/biznes/374921-50-krupnyh-kompaniy-kotorye-ispolzuyut-blokcheyn-spisok-i (дата обращения: 28.04.2020).

[22] Кириллов И. Как технология блокчейн меняет современный мир. — СиБ, 2019. — URL: http://sib.com.ua/sib-02-99-2018/bitkoin.html (дата обращения: 28.04.2020).

[23] Скаредов С.А. Унификация взаимодействия с различными реализациями blockchain в качестве платежных систем. — Санкт-Петербургский государственный университет, Кафедра системного программирования, Программная инженерия, 2019. — URL: http://se.math.spbu.ru/SE/YearlyProjects/spring-2019/371/Skaredov-report.pdf (дата обращения: 28.04.2020).