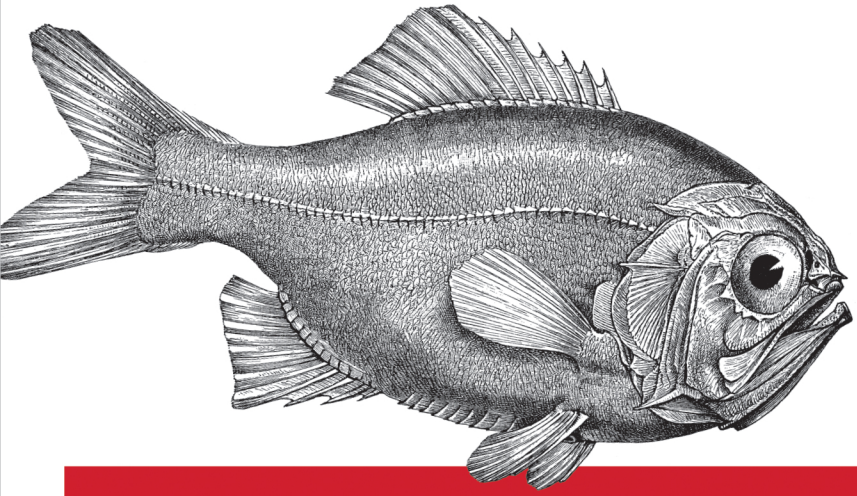
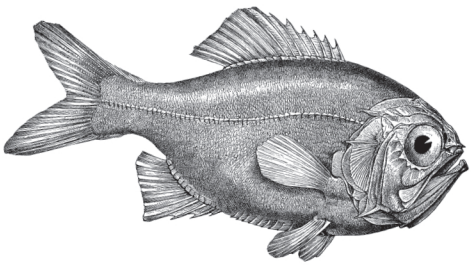
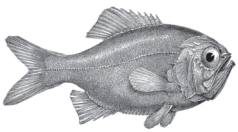
O'REILLY®



Децентрализованные приложения

Технология Blockchain в действии





ПИТЕР®

С. Равал

Decentralized Applications

Harnessing Bitcoin's Blockchain Technology

Siraj Raval

Beijing • Boston • Farnham • Sebastopol • Tokyo

O'REILLY®

С. Равал

Децентрализованные приложения

Технология Blockchain в действии

ПИТЕР®
Санкт-Петербург • Москва • Екатеринбург • Воронеж
Нижний Новгород • Ростов-на-Дону • Самара • Минск
2017

С. Равал

Децентрализованные приложения. Технология Blockchain в действии

Серия «Бестселлеры O'Reilly»

Перевел с английского *А. Киселев*

Заведующая редакцией	*Ю. Сергиенко*
Ведущий редактор	*Н. Римицан*
Литературный редактор	*О. Меркулова*
Художник	*С. Заматевская*
Корректоры	*Н. Витько, Н. Сидорова, Г. Шкатова*
Верстка	*Л. Егорова*

ББК 32.988.02-018.2
УДК 004.77

Равал С.

Р13 Децентрализованные приложения. Технология Blockchain в действии. — СПб.: Питер, 2017. — 240 с.: ил. — (Серия «Бестселлеры O'Reilly»).

ISBN 978-5-496-02988-9

Технология Bitcoin показала, как можно использовать криптографически сохраненные записи, модель с ограниченными ресурсами, открытый исходный код и пиринговые сети для создания нового типа успешных приложений.

Децентрализованные приложения гибче, прозрачнее и надежнее, чем современное программное обеспечение, созданное с применением традиционных моделей. Эта книга знакомит вас с основами создания децентрализованных приложений и принципами их разработки на примере нескольких доходных приложений. Причина такого коммерческого уклона объясняется тем, что прибыль (выгода) является основой успешного, надежного и перспективного децентрализованного приложения. Воспользуйтесь ими как трамплином к созданию вашего собственного приложения.

Держим курс на открытость и децентрализацию!

12+ (В соответствии с Федеральным законом от 29 декабря 2010 г. № 436-ФЗ.)

ISBN 978-1491924549 англ.
ISBN 978-5-496-02988-9

ООО «Питер Пресс», 192102, Санкт-Петербург, ул. Андреевская (д. Волкова), 3, литер А, пом. 7Н.
Налоговая льгота — общероссийский классификатор продукции ОК 034-2014, 58.11.12 — Книги печатные профессиональные, технические и научные.
Подписано в печать 23.12.16. Формат 60x90/16. Бумага офсетная. Усл. п. л. 15,000. Тираж 1000. Заказ 0000.
Отпечатано в ОАО «Первая Образцовая типография». Филиал «Чеховский Печатный Двор».
142300, Московская область, г. Чехов, ул. Полиграфистов, 1.
Сайт: www.chpk.ru. E-mail: marketing@chpk.ru
Факс: 8(496) 726-54-10, телефон: (495) 988-63-87

Оглавление

Спасибо тебе, Джейд!
Твоя истина делает меня свободным.

Введение

Соглашения, используемые в этой книге

В книге приняты следующие типографские соглашения.

Курсив

Используется для обозначения новых терминов, адресов URL, адресов электронной почты, имен файлов и расширений.

`Моноширинный шрифт`

Применяется для оформления листингов программ и программных элементов внутри обычного текста, таких как имена переменных и функций, базы данных, типы данных, переменные окружения, инструкции и ключевые слова.

`Моноширинный жирный`

Обозначает команды или другой текст, который должен вводиться пользователем.

`Моноширинный наклонный шрифт`

Обозначает текст, который должен замещаться фактическими значениями, вводимыми пользователем или определяемыми из контекста.

Использование примеров кода

Сопутствующие материалы (примеры кода, упражнения и пр.) можно скачать по ссылке *https://github.com/oreilly-media/decentralized_applications*.

Эта книга призвана помочь вам в решении задач. По большей части вы можете использовать код из книги в своих программах и документации. Вам не нужно связываться с нами по поводу получения разрешения на это, если только вы не начнете копировать достаточно существенные фрагменты кода. Например, написание программы, в которой используется несколько фрагментов кода из этой книги, не требует разрешения. А вот продажа или распространение компакт-дисков с примерами из книг издательства O'Reilly требует разрешения. Ответы на вопросы с использованием цитат из этой книги и приведением примеров не требуют получения разрешения. А вставка существенных объемов кода примеров из этой книги в документацию потребует разрешения.

Мы приветствуем, но не требуем добавлять ссылку на первоисточник при цитировании. Под ссылкой на первоисточник мы подразумеваем указание названия, автора, издательства и ISBN. Например: «Decentralized Applications by Siraj Raval (O'Reilly). Copyright 2016 Siraj Raval, 978-1-4919-2454-9».

За получением разрешения на использование значительных объемов программного кода примеров из этой книги обращайтесь по адресу *permissions@oreilly.com*.

Safari® Books Online

Safari Books Online — это виртуальная библиотека, содержащая авторитетную информацию в виде книг и видеоматериалов, созданных ведущими специалистами в области технологий и бизнеса.

Профессионалы в области технологий, разработчики программного обеспечения, веб-дизайнеры, а также бизнесмены и творческие работники используют Safari Books Online как основной источник информации для проведения исследований, решения проблем, обучения и подготовки к сертификационным испытаниям.

Библиотека Safari Books Online предлагает широкий выбор продуктов и тарифов для организаций, правительственных и учебных учреждений, а также физических лиц.

Подписчики имеют доступ к поисковой базе данных, содержащей информацию о тысячах книг, видеоматериалов и рукописей от таких издателей, как O'Reilly Media, Prentice Hall Professional, Addison-Wesly Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology и сотен других. За подробной информацией о Safari Books Online обращайтесь по адресу: *http://www.safaribooksonline.com/*.

Как с нами связаться

С вопросами и предложениями, касающимися данной книги, обращайтесь в издательство:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (в США и Канаде)
707-829-0515 (международный)
707-829-0104 (факс)

Список опечаток, файлы с примерами и другую дополнительную информацию вы найдете на веб-странице книги *http://bit.ly/decentralized-applications*.

Свои пожелания и вопросы технического характера отправляйте по адресу: *bookquestions@oreilly.com*.

Дополнительную информацию о книгах, курсах, конференциях и новостях вы найдете на сайте: *http://www.oreilly.com*.

Ищите нас в Facebook: *http://facebook.com/oreilly*.

Следуйте за нами в Твиттере: *http://twitter.com/oreillymedia*.

Смотрите нас на YouTube: *http://www.youtube.com/oreillymedia*.

От издательства

Ваши замечания, предложения, вопросы отправляйте по адресу comp@piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

На веб-сайте издательства www.piter.com вы найдете подробную информацию о наших книгах.

1 Что такое децентрализованные приложения?

В настоящее время активно формируется новая модель создания масштабируемых и эффективных приложений, основы которой были заложены технологией Bitcoin, включающей криптографическую регистрацию сделок, модель с ограниченными ресурсами и пиринговые технологии. Указанные характеристики послужили отправной точкой для создания нового типа программного обеспечения, получившего название «децентрализованные приложения». Эти приложения только начинают освещаться в печати, но, думаю, однажды они получат куда более широкое распространение, чем самые популярные веб-приложения. Децентрализованные приложения гибче, прозрачнее, надежнее и имеют более мотивирующую организацию, чем современное программное обеспечение, созданное с применением традиционных моделей. В ваших руках первая книга, которая поможет вам понять эти приложения и научиться разрабатывать их.

Что такое Bitcoin?

Прежде чем окунуться в тонкости децентрализованных приложений, поговорим немного о Bitcoin и Сети. За последнее десятилетие размер Всемирной паутины увеличился не в разы, а на порядки. Миллиарды людей получили доступ к Сети благодаря широкому распространению устройств, поддерживающих подключение к Интернету. На первый взгляд, стандартная многоуровневая архитектура взаимодействий, реализованная в комплекте протоколов Интернета, вполне удовлетворяет насущные потребности: канальный уровень передает некоторые данные в сеть; межсетевой уровень осуществляет их маршрутизацию; транспортный уровень обеспечивает сохранность и надежную доставку данных; а прикладной уровень определяет абстракции данных в форме приложений. Все четыре уровня слаженно работают друг с другом, обеспечивая обмен данными, но не денежными суммами. Bitcoin действует как пятый уровень, осуществляющий передачу денежных сумм, в соответствии со стандартами других уровней.

Мы имеем возможность осуществлять платежи через Сеть. Но в этом процессе участвуют неэффективные, устаревшие системы вроде Automated Clearing House (ACH), созданные еще в доинтернетную эпоху. Эти традиционные платежные системы жутко медлительны, поскольку все операции выполняются централизованно. Компьютеры не должны ждать сутками, пока пройдет платеж; они постоянно взаимодействуют между собой. У них должна быть возможность производить миллиарды микроплатежей друг другу за потребленные ресурсы, такие как электроэнергия и место в хранилище, и не платить огромные гонорары за услуги посредников. Именно эту проблему помогает решить Bitcoin.

С появлением Bitcoin стала наконец возможной мгновенная и децентрализованная передача эквивалентов денежных сумм. Анонимный создатель Bitcoin, использовавший вымышленное имя Сатоси Накамото (Satoshi Nakamoto), нашел эффективное решение *задачи византийских генералов* — задачи, десятилетиями досаждавшей криптологам. Процитируем определение задачи византийских генералов из оригинальной статьи (Lamport, 1982): «[Вообразите] группу генералов византийской армии, окружившей неприятельский город. Общаясь только посредством надежной связи, генералы должны согласовать план сражения. Но один или несколько из них могут быть предателями, стремящимися ввести в заблуждение остальных. Задача заключается в том, чтобы найти алгоритм, гарантирующий достижение согласия между лояльными генералами». Достижимость децентрализованного консенсуса в Bitcoin означает, что ни одна сторона больше не должна обращаться к центральному органу или доверять другим сторонам для обмена информацией, включая информацию в форме сделок с денежными суммами.

Bitcoin и другие криптовалюты помогут определить пятый уровень Интернета, с помощью которого компьютеры смогут передавать денежные суммы так же быстро и эффективно, как данные. Bitcoin — полезный инструмент для оперативного перечисления средств, но самая большая его ценность заключена в базовой технологии с названием *blockchain* (цепочка блоков), которая впервые в истории сделала возможным децентрализованный консенсус.

Цепочка блоков (blockchain) — это база данных с широкомасштабным тиражированием всех транзакций в сети Bitcoin. Цепочка блоков использует механизм консенсуса (согласования) с названием «доказательство выполнения

работы» (proof-of-work), предотвращающий проблему двойных расходов (double-spending) в Сети, десятилетиями преследовавшую криптологов. Под двойными расходами подразумевается проблема, когда мошенник может вторично потребовать оплаты, отрицая успешность первой транзакции.

Механизм доказательства выполнения работы решает эту проблему за счет так называемых *майнеров* (miners) в сети, отыскивающих криптографические доказательства с использованием своей аппаратуры. Майнеры — это узлы в сети Bitcoin, проверяющие транзакции с использованием истории цепочки блоков, датированных записей обо всех транзакциях, когда-либо выполнявшихся в сети. Теоретически кто-то мог бы изменить историю цепочки блоков у себя, но из-за механизма доказательства выполнения работы ему также потребуется значительная доля вычислительной мощности сети, чтобы проверка прошла успешно. Поскольку в настоящее время сеть Bitcoin имеет намного бóльшую вычислительную мощность, чем все суперкомпьютеры в мире, злоумышленнику будет чрезвычайно трудно взломать сеть.

Механизм доказательства выполнения работы — дорогой в плане потребления таких ресурсов, как электроэнергия и вычислительная мощность, но это единственный известный механизм предотвращения атак Сибиллы (Sybil)[1], при которых мошенник представляется множеством людей в Сети и получает ресурсы незаконно. Успешная атака Сибиллы на сеть Bitcoin скорее всего привела бы к полной

[1] *http://ru.bitcoinwiki.org/Уязвимости_Bitcoin#.D0.90.D1.82.D0.B0.D0.BA.D0.B0_.D0.A1.D0.B8.D0.B1.D0.B8.D0.BB.D0.BB.D1.8B. — Примеч. пер.*

девальвации валюты из-за утраты доверия к ее стабильности. Ресурсоемкость механизма доказательства выполнения работы — единственное, что обеспечивает его надежность в широкомасштабной сети.

Итак, у нас имеется новый инструмент с названием blockchain (цепочка блоков) — база данных с широкомасштабным тиражированием транзакций, — способный противостоять атакам Сибиллы. Технология blockchain впервые позволила достичь децентрализованного консенсуса без использования центрального сервера. Возможно, вам интересно узнать, как ею пользоваться, и такой интерес вполне объясним. Я собираюсь посвятить значительную часть книги описанию всех ее возможностей и способов их реализации. Но в данный момент важно понять, что эта структура данных — одна из многих, с помощью которых вы будете создавать надежные децентрализованные приложения.

Что такое децентрализованное приложение?

Многие из нас понимают под термином «приложение» программное обеспечение. Программное приложение — это программное обеспечение, созданное для определенной цели. В мире используются миллионы программных приложений, и подавляющая часть веб-приложений реализует централизованную модель клиент-сервер. Многие приложения реализуют распределенную модель, и лишь некоторые новейшие приложения являются децентрализованными. На рис. 1.1 показано визуальное представление этих трех программных моделей.

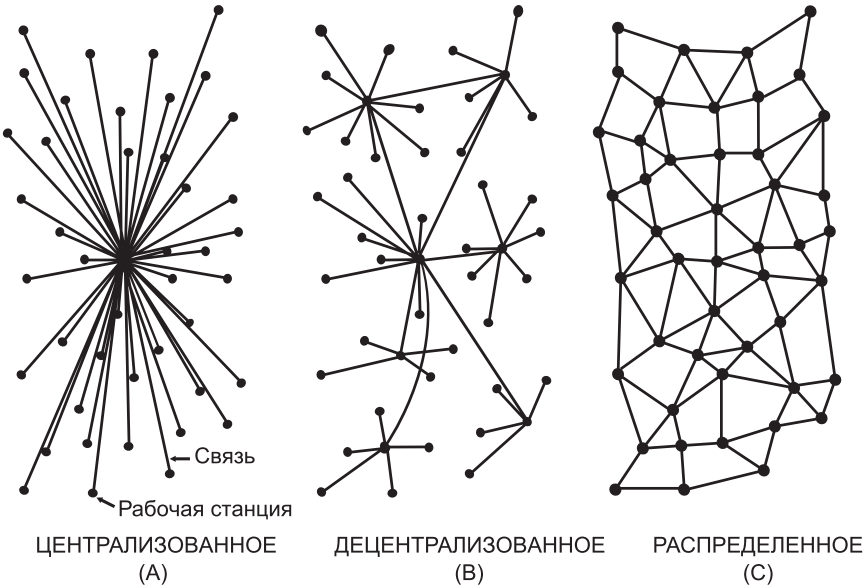


Рис. 1.1. Три разных типа программных приложений

Централизованная модель является наиболее распространенной в настоящее время. Централизованные системы непосредственно управляют работой отдельных блоков, а вся информация протекает через единый центр. Работа отдельных рабочих станций напрямую зависит от способности центра посылать и принимать информацию, а также осуществлять управление. Facebook, Amazon, Google и другие господствующие службы в Интернете реализованы с использованием данной модели. Мы будем называть эти гигантские службы стеками. Стеки оказывают нам ценные услуги, но они имеют существенные недостатки, о которых я расскажу в главе 2.

Чем отличаются децентрализованные и распределенные приложения?

Распределенными (distributed) называют системы, в которых вычисления распределяются между несколькими узлами. Под децентрализованными (decentralized) подразумеваются системы, в которых отсутствуют узлы, управляющие работой других узлов. Многие стеки, такие как Google, внутренне имеют распределенную архитектуру, позволяющую повысить скорость вычислений. Таким образом, система может быть одновременно централизованной и распределенной.

Может ли система быть одновременно распределенной и децентрализованной?

Да, может. Bitcoin — это распределенная система, потому что журнал регистрации сделок, цепочка блоков, хранится на множестве компьютеров. Она одновременно является децентрализованной, потому что в случае выхода из строя одного узла вся сеть сможет продолжить работу в нормальном режиме. Таким образом, любое приложение, использующее цепочку блоков наряду с другими пиринговыми технологиями, может быть одновременно распределенным и децентрализованным.

Тогда почему для книги не было выбрано название «Распределенные и децентрализованные приложения»?

Централизованные системы тоже могут быть распределенными. Программные приложения, способные достигать децентрализованного консенсуса, являются настоящим новшеством.

Значит, единственной отличительной чертой децентрализованных приложений является умение достигать децентрализованного консенсуса?

Область разработки децентрализованных приложений сейчас активно формируется, и в ее развитии участвует много специалистов, продолжающих экспериментировать с новыми моделями. Разные разработчики имеют различное представление о том, что такое децентрализованное приложение. Некоторые считают, что достаточно отсутствия центральной точки отказа, тогда как другие предполагают соответствие некоторым дополнительным требованиям. Основное внимание в данной книге уделяется доходным децентрализованным приложениям, то есть децентрализованным приложениям, с помощью которых разработчики и пользователи могут зарабатывать деньги. Причина такого коммерческого уклона объясняется тем, что прибыль (выгода) является основой успешного, надежного и перспективного децентрализованного приложения. Материальная заинтересованность стимулирует работоспособность разработчиков, лояльность пользователей и готовность майнеров хранить цепочки блоков. В связи с этим на рис. 1.2 показаны четыре основных свойства любого доходного децентрализованного приложения.

Свойство 1: открытый исходный код

Децентрализованные приложения с закрытым исходным кодом требуют от пользователей принять на веру, что они являются децентрализованными, как заявляют их разработчики, и не нуждаются в наличии централизованного источника данных. Поэтому закрытость приложения воспринима-

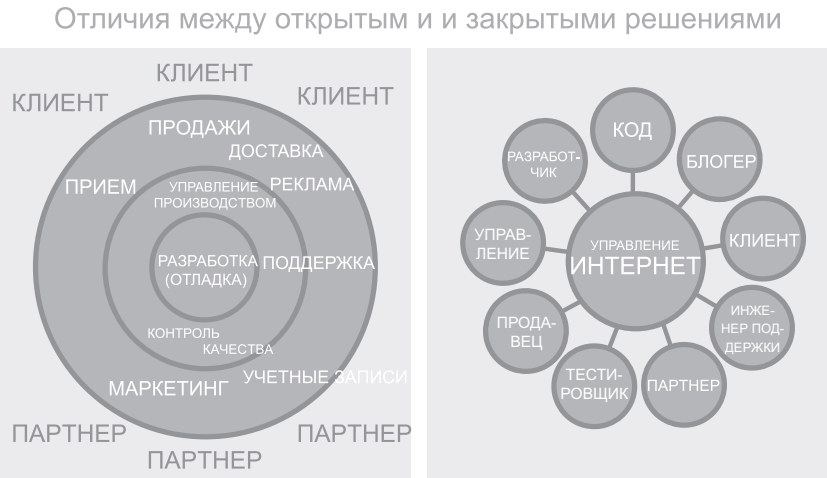


Рис. 1.2. Закрытые и открытые решения

ется пользователем как тревожный сигнал и препятствует его внедрению. Неприятие закрытых решений особенно ярко проявляется в отношении приложений, созданных для приема, хранения или передачи денежных средств пользователей. Хотя децентрализованное приложение с закрытым исходным кодом тоже может добиться успеха, оно изначально находится в невыгодной позиции, так как пользователи отдают предпочтение открытым альтернативам. Открытость исходного кода децентрализованного приложения меняет подход к работе таким образом, что связующим звеном становится Интернет, а не цепочка закрытых хранилищ.

Код любого приложения можно открыть. Почему это не делается?

Все традиционные бизнес-модели требуют, чтобы продаваемые продукты или услуги по своим качествам превосходили аналогичные продукты или услуги конкурентов.

Открытость продукта предполагает, что любой конкурент может взять результат вашего труда и продавать его как собственный продукт.

Что может побудить разработчиков открыть исходный код приложения, от которого они собираются получать доход?

Bitcoin — хороший пример открытого децентрализованного приложения, приносящего создателям неплохой доход. Сатоси получил некоторое первоначальное количество Bitcoin и позволил остальным заработать остальное. Поскольку количество Bitcoin ограничено и сеть сама по себе имеет огромную ценность для общества, предоставляя механизм доказательства выполнения работы (proof-of-work), цена Bitcoin начала расти, и, соответственно, начало увеличиваться состояние Сатоси. Открытость приложения обеспечивает прозрачность, необходимую для улучшения кода добровольными разработчиками, и повышает веру пользователей в ценность криптовалюты Bitcoin. Открыв свое децентрализованное приложение, вы получите доверие потенциальных пользователей. Конечно, любой желающий сможет начать новый проект на основе вашего приложения, но он не сможет клонировать команду разработчиков. Пользователи предпочитают поддерживать людей, лучше подготовленных к сопровождению децентрализованного приложения, и часто такими людьми оказываются разработчики оригинального проекта.

Свойство 2: внутренняя валюта

Вопрос, который постоянно всплывает в сфере децентрализованных приложений, — как превратить децентрализо-

ванное приложение в деньги? Традиционные способы монетизации централизованных приложений включают взимание платы за транзакции, показ рекламы, привлеченных клиентов, права доступа к пользовательским данным и за подписки на услуги. А как извлечь прибыль из децентрализованного приложения, если открыть его исходный код? Можно попробовать программно брать плату за транзакции в сети и автоматически переводить ее на счет разработчика, но такое решение не идеально, так как предполагает, что все пользователи проявят добропорядочность и никто не клонирует ваше приложение, удалив из него взимание ваших комиссионных. Никто не встроит рекламу, подписку на услуги и не реализует какую-либо другую централизованную бизнес-модель.

Как разработчик какого-нибудь децентрализованного приложения сможет извлекать из него прибыль?

Для этого необходимо разместить в сети дефицитные ресурсы и использовать ограниченное количество токенов — своеобразной внутренней валюты приложения (монет). Пользователям нужны такие токены, чтобы работать с сетью. Владельцы дефицитных ресурсов взимают плату в токенах. В сети Bitcoin владельцы (майнеры) дефицитных ресурсов (вычислительных мощностей) взимают плату за транзакции непосредственно с пользователей, чтобы они могли пользоваться услугами. Поскольку сеть постоянно растет, число пользователей увеличивается, а количество токенов остается неизменным, ценность монет также растет. Эту модель можно распространить на любое децентрализованное приложение. Роль дефицитных ресурсов могут играть: пространство для хранения данных, сделки, изображения, видеоролики, книги, объявления и так далее.

Означает ли это, что пользователи должны платить за использование любого децентрализованного приложения?

И да и нет. Хотя цепочки блоков остаются основным платежным средством, существуют другие способы стимулирования в децентрализованных приложениях. Можно начислять пользователям бонусы или даже давать им право продавать их данные или пространство хранения за токены. Помимо использования токенов, создатели децентрализованного приложения могут извлекать прибыль из виртуальных ресурсов, например продавая виртуальную недвижимость в ролевых играх, домены в особенном пространстве имен или даже дополнительные баллы к репутации.

Свойство 3: децентрализованный консенсус

До появления Bitcoin для достижения консенсуса в определении достоверности сделки всегда требовалась некоторая централизация. Если вы вносили плату за что-то, ваша транзакция должна была пройти через информационный центр, учитывавший все сделки. Сеть Bitcoin является одноранговой (Peer to Peer, P2P), то есть узлы в ней могут напрямую взаимодействовать друг с другом. Одноранговые (P2P) сети не являются чем-то новым; до цепочек блоков была изобретена технология распределенных хеш-таблиц (Distributed Hash Tables, DHT), использовавшаяся в BitTorrent. Таблицы DHT прекрасно подходят для хранения и передачи децентрализованных данных, но если нужны конструкции уровня приложения, такие как имена пользователей, обновления состояния, таблицы рекордов и так далее, требующие достижения согласия децентрализованным способом, вам не обойтись без цепочек блоков.

Цепочки блоков не устраняют потребность в таблицах DHT, а лишь дополняют их. Уникальность цепочек блоков заключается в их способности решить главную проблему безопасности, свойственную таблицам DHT: они не требуют от узлов доверять достоверности данных друг друга. Цепочка блоков — это децентрализованная база данных транзакций, причем это первая децентрализованная база данных с высокой защищенностью от взлома. Защищенность цепочек блоков была главной целью проекта. Кроме того, цепочка блоков является первым организационно и логически децентрализованным журналом транзакций. Следующая таблица наглядно показывает, что я имею в виду.

	Организационно централизованное	**Организационно децентрализованное**
Логически централизованное	Paypal	Bitcoin
Логически децентрализованное	Excel	Электронная почта

Новинкой цепочек блоков является децентрализованный консенсус. Если приложение требует достижения согласия в чем-то между всеми участвующими сторонами, вам придется использовать цепочки блоков. Простым примером может служить система именования пользователей, для которой не важно, кто владеет именем «@user», однако важно, чтобы все стороны согласились, что им владеет кто-то один. В прошлом было создано огромное количество децентрализованных протоколов, но все они требуют, чтобы узлы доверяли друг другу. Цепочка блоков — неизменяемая запись, копии которой хранятся всеми узлами, поэтому никто не сможет заявить, что он тоже является

пользователем @user. Это достигается за счет использования *умных контрактов* (smart contracts)[1].

Умный контракт — это фрагмент кода в реализации цепочек блоков. Когда выполняется некоторое предопределенное условие, умный контракт выполняет соответствующий пункт договора. Возникает вопрос: чем умный контракт отличается, например, от такого кода, написанного с применением Stripe API?

```
if (user.sendsMoney(customerID))
{
  runContract();
}

func runContract()
{
  println('hello world');
}
```

Главное отличие состоит в следующем: умные контракты находятся в цепочке блоков, а не на сервере. Они не требуют доверять какой-либо третьей стороне, например владельцу службы Stripe или сервера. Таким образом, более формально умный контракт можно было бы определить как криптографически защищенный код. Единственное, что нужно помнить, — не весь код децентрализованного приложения является умным контрактом; и хотя умные контракты имеют свою узкую область применения, для целей нашего обсуждения мы будем считать, что они занимают место «модели» в архитектуре децентрализованных приложений «модель-представление-контроллер». Более детально мы поговорим об этом, когда я перейду к обсуждению архитектуры децентрализованного приложения.

[1] *https://ru.wikipedia.org/wiki/Умный_контракт — Примеч. пер.*

Свойство 4: отсутствие центральной точки отказа

Децентрализованные приложения никогда не прекращают работу благодаря отсутствию центрального сервера, который мог бы остановиться. Данные в децентрализованных приложениях распределены между всеми узлами. При этом все узлы действуют независимо; если один из них остановится, другие продолжат работу в сети. Существует несколько децентрализованных баз данных, использующих эту особенность, например Interplanetary File System (межпланетная файловая система), BitTorrent и независимые таблицы DHT.

История появления децентрализованных приложений

На ранних этапах Всемирная паутина не имела такой практической пользы, как в наши дни, когда мы располагаем бесчисленным множеством приложений и служб на все случаи жизни, но уже тогда она позволяла ощутить, что такое распределенность. Всемирная паутина изначально не имела единого центра. Протокол HTTP соединяет все вычислительные устройства на планете, имеющие подключение к Интернету. В своей работе протокол HTTP опирается на множество доверительных серверов, преобразующих веб-адреса в сетевые адреса серверов. Кроме того, протокол HTTPS добавляет еще один уровень доверительных серверов и центров сертификации. Люди получили возможность устанавливать и включать в работу собственные серверы, к которым могут подключаться другие и хранить на них свои данные. Но вскоре стали появляться серверы прило-

жений, и родилась хорошо известная ныне централизованная модель владения данными. Почему развитие пошло по этому пути?

Ответ прост: потому что этот путь проще и идеологически, и программно. Он оказался самым простым в реализации, и он работал. Один человек или группа оплачивали содержание сервера и получали прибыль от пользователей их программного обеспечения. Одними из первых популярных централизованных приложений стали MySpace и Yahoo!. Более современные приложения, такие как Uber и Airbnb, децентрализуют отдельные сегменты бизнеса, предоставляя централизованное и доверенное хранилище данных. Они одними из первых позволили извлекать прибыль в разных сферах экономики. Их децентрализованная бизнес-модель предвещает появление еще более децентрализованных приложений.

На некотором этапе развития Всемирной паутины появился новый протокол, разработанный Брэмом Коэном, получивший название *BitTorrent*. Этот протокол был создан для решения проблемы длительности загрузки огромных медиафайлов через HTTP и как усовершенствование некоторых одноранговых (P2P) предшественников вроде Gnutella, Napster и Grokster. Проблема заключалась в том, что загрузка огромных файлов занимала очень продолжительное время, а с ростом Всемирной паутины увеличивались и размеры самих файлов. В то же время росла емкость жестких дисков и увеличивалось количество людей, подключенных к Интернету. BitTorrent решил проблему, превратив загружающих в раздающих.

Если вам требовался некоторый файл, вы могли загрузить его не из одного, а сразу из нескольких источников. Чем

популярнее файл, тем больше пользователей загружало и, соответственно, раздавало его. А чем больше источников, тем быстрее осуществляется загрузка. Сидеры (seeders)[1] вознаграждались более высокими скоростями загрузки, а личеры (leechers)[2], напротив, наказывались ограничением скорости. Система передачи данных, организованная по принципу «ты — мне, я — тебе», доказала свою эффективность для распространения медиафайлов большого размера, таких как фильмы и записи телевизионных передач.

Протокол BitTorrent продолжает развиваться и для многих является основным способом загрузки огромных файлов, таких как игры или фильмы. Благодаря скорости, устойчивости и наличию механизма вознаграждения протокол BitTorrent лучше подходит для загрузки больших объемов данных, чем HTTP.

Тогда почему развитие Всемирной паутины не пошло по этому пути?

Вероятно, из-за того что HTTP оказался первым, а также благодаря его инфраструктуре и потраченным на его развитие средствам и времени. В настоящее время активно ведутся исследования в направлении модернизации Всемирной паутины с применением BitTorrent-подобной технологии, и они почти наверняка увенчаются успехом, по-

[1] В терминологии протокола BitTorrent «сидер» («сид») — это узел, имеющий все сегменты распространяемого файла, то есть либо начальный распространитель файла, либо уже скачавший весь файл и оставшийся на раздаче. — *Примеч. пер.*

[2] В терминологии протокола BitTorrent «личер» («лич», от *англ.* leech — пиявка) — это узел, не имеющий пока всех сегментов, то есть продолжающий скачивание. Термин часто употребляется в негативном смысле: пользователь, который отдает гораздо меньше, чем скачивает. — *Примеч. пер.*

тому что BitTorrent имеет неоспоримые преимущества. Как только появилась технология BitTorrent, разработчики стали использовать ее для создания некоммерческих децентрализованных приложений. Давайте познакомимся с некоторыми примерами последних децентрализованных приложений.

PopcornTime

PopcornTime использует протокол BitTorrent для передачи потокового видео между пользователями в режиме реального времени, действуя подобно BitTorrent-клиенту Netflix. Это настоящий кошмар для американской ассоциации кинокомпаний (Motion Picture Association of America, MPAA). Никакой регулятор не способен установить свои ограничения, и теперь любой может получить свободный доступ к фильмам. PopcornTime оказался практичным децентрализованным приложением, действующим как децентрализованная версия Netflix. Создатели утверждают, что их приложение используется во всех странах и даже в двух странах, где нет Интернета. В PopcornTime отсутствует внутренняя конкуренция, и в нем не требуется поддержка децентрализованного консенсуса, поэтому нет необходимости использовать цепочки блоков. Это приложение просто передает потоковое видео, и тем самым обеспечивается его немалая ценность.

OpenBazaar

Целью разработки OpenBazaar было создание децентрализованной версии Ebay. В приложении OpenBazaar отсутствуют посредники, которые указывали бы продавцам,

что они могут продавать, а что — нет, или взимали бы плату за услуги. Оно основано на протоколе BitTorrent, однако проблема заключается в том, что продавцы должны иметь собственные хранилища. Они вынуждены настраивать собственные серверы, чтобы пользователи могли видеть продаваемые товары. В идеале продавцы могли бы просто выгружать свою информацию в сеть, возможно, за небольшую плату, и избавиться от лишнего беспокойства. Но для этого необходима децентрализованная система заинтересованных хранителей (майнеров в терминологии Bitcoin), о которых подробнее рассказывается в главе 4. Для передачи данных в OpenBazaar используется протокол BitTorrent, а для расчетов — криптовалюта Bitcoin.

FireChat

Приложение FireChat появилось в связи с известными событиями — демократическими протестами в Гонконге в 2014 году. Печально известный «Великий китайский файервол» (Great Firewall) блокирует доступ к продемократическим сайтам или к сайтам, распространяющим информацию, идущую вразрез с интересами китайского правительства. Протестующие боялись, что правительство попытается закрыть доступ к различным социальным сетям, чтобы лишить их общения посредством протокола HTTP. Поэтому они применяли FireChat, приложение, использующее новую возможность iOS 7 с названием Multipeer Connectivity (многосторонние соединения), позволяющую телефонам связываться друг с другом без посредников. Из-за отсутствия центральной точки отказа правительству пришлось бы вручную отключать каждый узел, что прак-

тически невозможно, поэтому протестующие могли уверенно общаться друг с другом.

Децентрализованный протест в лучшем его проявлении.

Lighthouse

Мы подробно обсудим Lighthouse в главе 5, а пока отмечу, что это кошелек для Bitcoin с набором встроенных умных контрактов. Умные контракты помогают вкладывать деньги в определенные проекты, так же как, например, Kickstarter. Когда проект достигает своей цели, появляется возможность вернуть средства из кошелька Lighthouse проекта, собирающего средства. Вкладчик может в любой момент отозвать свой вклад без уведомления создателя проекта. Lighthouse — отличный пример использования существующей инфраструктуры Bitcoin для создания своего децентрализованного приложения. Lighthouse — это всего лишь пользовательский интерфейс с несколькими встроенными умными контрактами Bitcoin. Приложение работает, опираясь на круг пользователей Bitcoin. Оно открыто, полагается на поддержку децентрализованного консенсуса, не имеет центральной точки отказа, но использует не собственную валюту, а Bitcoin. Это очень полезное приложение, но оно не приносит коммерческой выгоды создателям.

Gems

Gems — это приложение-мессенджер, разработанное с целью создания более справедливой бизнес-модели, чем WhatsApp. Gems выпускает собственную валюту и позволяет рекламодателям платить пользователям напрямую за

свои данные, без участия посредников, получающих свою долю. Пользователи тоже могут зарабатывать гемы, привлекая новых пользователей в сеть. Гемы — это метамонеты валюты Gems, основанной на Bitcoin, которые разработчики также получают за разработку и сопровождение программного обеспечения. Круг пользователей Gems постоянно расширяется, поэтому растет ценность валюты. Пользователи заинтересованы в расширении сети и получении дохода, как и разработчики. Gems можно рассматривать как пример долевого участия в развитии децентрализованного приложения. Его исходный код закрыт, поэтому у пользователей нет возможности убедиться в отсутствии единственной точки отказа. Gems — это коммерчески выгодное приложение, но, на мой взгляд, оно не обладает достаточным запасом надежности, чтобы выдержать конкуренцию с продуктами, отвечающими трем другим критериям.

Значит, не существует отдельных децентрализованных приложений, которые удовлетворяли бы всем четырем критериям: отсутствию центральной точки отказа, наличию внутренней валюты, поддержке децентрализованного консенсуса и открытому исходному коду?

Существует довольно много криптовалют, отвечающих всем четырем критериям, но криптовалюты не являются децентрализованными приложениями. Если говорить о децентрализованных социальных сетях, службах организации совместных поездок, поисковых движках, когда берется стек и децентрализуется, тогда ответ на заданный вопрос — нет. Однако это вполне возможно — необходимые технологии существуют, и как только появится что-то новое, многие разработчики устремятся на децентрализованную сторону,

чтобы делать серьезные деньги для себя и своих пользователей. Давайте поговорим о некоторых из этих поддерживающих технологий.

Поддерживающие технологии

Обсуждая историю появления децентрализованных приложений, я упомянул некоторые поддерживающие технологии. Цепочки блоков Bitcoin, конечно же, имеют первостепенное значение, поэтому мы рассмотрим эту технологию в первую очередь. Цепочки блоков помогают решить задачу византийских генералов. Данная задача ставит вопрос: как скоординировать распределенные узлы, чтобы прийти к некоторому консенсусу, способному противостоять злоумышленнику, старающемуся его разрушить? Алгоритм доказательства выполнения работы и цепочки блоков помогают ее решить.

С созданием криптовалюты Bitcoin стало возможным достижение децентрализованного консенсуса. Алгоритм доказательства выполнения работы не идеален — он потребляет значительные объемы вычислительных ресурсов и электроэнергии. Существуют альтернативные криптовалюты, решающие значимые задачи, такие как криптовалюта PrimeCoin, майнеры которой расходуют свои вычислительные ресурсы для поиска простых чисел. В мире, где валютой де-факто является Bitcoin, требуется тратить много энергии для поддержания сети, энергии, которой можно было бы найти лучшее применение.

Проблема в том, что доказательство выполнения работы — единственный известный алгоритм, способный противо-

стоять атакам Сибиллы. Поиск алгоритмов консенсуса не прекратился с появлением алгоритма доказательства выполнения работы и все еще продолжается, но пока это лучшее, что у нас есть. Говоря о перспективных конкурентах доказательства выполнения работы, можно отметить алгоритм *подтверждения доли* (*proof-of-stake*). Этот алгоритм также несовершенен, но его можно использовать как дополнение к алгоритму доказательства выполнения работы.

Подтверждение доли (proof-of-stake) — это механизм консенсуса, использующий вычислительные мощности для предотвращения атак Сибиллы на долю в сети. Обычно под долей подразумевается количество средств, которыми владеет майнер. То есть чем больше криптовалюты вы имеете, тем больший вклад в стабильность сети вносите и тем меньше вероятность, что вы выполните атаку «51 процент» с целью расщепления (fork) цепочки блоков. Делегированное подтверждение доли (delegated proof-of-stake) — усовершенствованная версия алгоритма подтверждения доли, в которой выбирается 101 делегат, имеющий право голосовать в генераторах блоков. Оба алгоритма — простого и делегированного подтверждения доли — все еще находятся в процессе исследования, и если они окажутся безопасными в долгосрочной перспективе, их можно будет использовать как дополнение или даже взамен алгоритма доказательства выполнения работы.

Определение терминов

Итак, почему выбран термин «децентрализованные приложения»? Почему не «децентрализованные прикладные организации» (ДПО) или «децентрализованные автономные корпорации» (ДАК)?

Пространство криптовалют наполнено различными терминами, определяющими теорию и отчасти практику экосистемы децентрализованных приложений. Чтобы лучше понять, почему я выбрал термин «децентрализованные приложения», я предлагаю рассмотреть все имеющиеся термины, применяемые для децентрализованных приложений, и посмотреть, что они означают. Начнем с самого термина «децентрализованные приложения».

Децентрализованные приложения (ДП)

«Децентрализованные приложения» — это название данной книги. Я точно так же мог бы выбрать термин ДО, ДПО или ДАК. Но почему именно «децентрализованные приложения»? Потому что во всех фразах присутствует общее слово «децентрализованные». Децентрализованные приложения — это суперкласс всех децентрализованных сущностей, имеющих отношение к программному обеспечению.

Децентрализованные организации (ДО)

Децентрализованными называют организации, в которых полномочия распределены между сотрудниками. Этот термин не относится к инструментам, используемым в организации; скорее он описывает, как она структурирована. Существуют разные степени децентрализации, и полная децентрализация не всегда является лучшим вариантом. В традиционных организациях имеется жесткая иерархия управления.

Децентрализованная организация дает право голоса всем своим сотрудникам и более равномерно распределяет полномочия между ними. Практика и ориентиры компании подконтрольны всем и могут храниться в децен-

тралізованной сети для большей надежности. Люди не должны быть единственными, кто принимает решения: для таких задач, как начисление зарплаты к определенной дате, можно использовать умные контракты. Децентрализованные организации не обязаны располагаться в каком-то одном городе; их подразделения могут быть разбросаны по всему миру. В некоторых системах (например, Bitcoin) возможность договориться считается ошибкой. В децентрализованных организациях возможность договориться рассматривается как достоинство. В политике децентрализованная власть называется демократией. Стартапы в последнее время все чаще выбирают более децентрализованную структуру, особенно с развитием инструментов удаленного сотрудничества, таких как Slack и GitHub.

Автоматизированные агенты (АА)

Под термином АА не обязательно понимается Скайнет (SkyNet) или искусственный интеллект (ИИ) вообще. Автоматизированные агенты известны уже как минимум лет десять. Термин АА просто означает программное обеспечение, выполняющееся без участия человека, то есть автономно. Прекрасным примером может служить компьютерный вирус. Разработчик создал и выпустил его, а он сам копирует себя или выполняет другой алгоритм, зашитый в него. Другим примером может служить демон — программа, которая выполняется как фоновый процесс в операционной системе, например сервер электронной почты. Автоматизированные агенты имеют свои достоинства и недостатки; они не требуют обслуживания, но неконтролируемые агенты могут превратиться в источник опасности для человечества — подробнее об этом рассказывается в главе 6.

Децентрализованные автономные организации (ДАО)

Именно так я сначала хотел назвать книгу. ДАО — это почти то же самое, что ДО, за исключением того, что решения принимают не люди, а искусственный интеллект. Протокол располагается в децентрализованном стеке и не учитывает никаких правовых ограничений. Люди не несут ответственности, они находятся с краю. Решения принимаются искусственным интеллектом, и ДАО обслуживает себя сама. ДАО определяются не только тем, что все решения принимаются искусственным интеллектом, они также имеют свой внутренний капитал.

Одним словом, каждая из них является подклассом децентрализованных приложений, и ДАО — это децентрализованное приложение, которое управляется искусственным интеллектом и в котором люди не несут никакой ответственности. В отличие от децентрализованных организаций возможность договориться в ДАО считается не достоинством, а ошибкой. Bitcoin является примером ДАО.

Децентрализованные автономные корпорации (ДАК)

Это один из спорных терминов. Некоторые полагают, что такая фраза сама по себе бессмысленна, потому что корпорация является сочетанием прежней системы правовых договоров и централизованной иерархии управления, которую мы пытаемся усовершенствовать. С другой стороны, ДАК — это подкласс ДАО, где ее членам выплачиваются дивиденды. Мне больше по душе первое мнение, потому что я не люблю термин «корпорация», и, если ДАО пожелает реализовать дивиденды

для своих людей и/или машин, она сможет сделать это как ДАО, не превращаясь в ДАК.

Итак, мы поговорили о децентрализованных приложениях, ДО, ДАО, АА и ДАК. Теперь взгляните на рис. 1.3, который поможет вам немного прояснить суть.

Мне очень нравится эта диаграмма, потому что она включает в себя все, о чем мы говорили до сих пор. Мы еще не на той стадии, когда возможно создание искусственного интеллекта (заветная цель, как обозначено на диаграмме), но мы уже на том этапе развития, когда можно приступить к созданию ДАО.

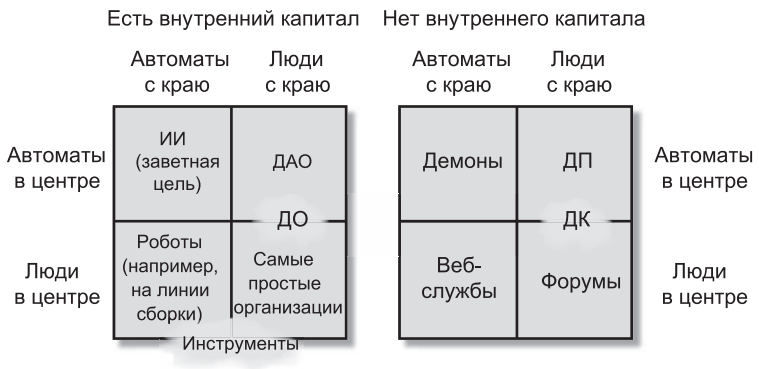


Рис. 1.3. Виды организаций (предоставлен Виталиком Бутериным (Vitalik Buterin))

Для простоты везде в книге будет использоваться термин «децентрализованные приложения». Поскольку децентрализованные приложения являются суперклассом всего децентрализованного программного обеспечения и я предполагаю обсудить различные инструменты и методики,

доступные для использования, вы сможете отлично подготовиться к выбору типа децентрализованного приложения для создания.

Я вывел эти определения в ходе исследования сообщества пользователей криптовалют, у меня не было цели дать новые названия понятиям или создать новые парадигмы. В действительности я стремился максимально упростить предметную область, чтобы вы могли в полной мере узнать, какие инструменты доступны для создания прибыльных децентрализованных приложений. Пространство идей централизованных приложений практически исчерпало себя, и пришло время начать новый виток развития после оценки всех за и против. Децентрализованные приложения — это следующая волна программного обеспечения, и, надеюсь, данная книга поможет вам приготовиться к встрече с ней.

Начало

Надеюсь, что мое введение в децентрализованные приложения было достаточно подробным. Многое еще предстоит объяснить, но вы уже должны иметь общее представление о предметной области и быть знакомы со всеми основными терминами и аббревиатурами, связанными с децентрализованными приложениями. В данной книге я хочу в первую очередь рассказать, что такое децентрализованные приложения, зачем они нужны и как выглядит экосистема преуспевающего децентрализованного приложения. Затем я опишу способы реализации собственных децентрализованных приложений с применением существующих инструментов. В заключение мы подробно рассмотрим несколько крупнейших игроков в области децентрализованных приложений.

2 Экосистема преуспевающих децентрализованных приложений

Предметная область цепочек блоков весьма запутанна. Существует бесчисленное множество стартапов, альтернативных криптовалют, идеологий и окружающих их умных слов, и может быть очень трудно разобраться во всем этом. Весьма полезно разделить предметную область на три категории, как описывается в книге Мелани Свон «Blockchain» (O'Reilly)[1] и других; blockchain 1.0 — это валюта, blockchain 2.0 добавляет контракты (запасы, ценные бумаги, финансовые ресурсы), и blockchain 3.0 охватывает применение за пределами чисто финансовой области, например в управлении и здравоохранении (децентрализованные приложения). В данной главе мы поговорим о том, что необходимо для развития всех трех

[1] *Свон М.* Блокчейн. Схема новой экономики. М.: Олимп-бизнес, 2016. — *Примеч. пер.*

категорий. Как разработчику децентрализованных приложений вам достаточно одной вещи — подходящих инструментов, которые помогут вам сделать свои децентрализованные приложения защищенными, надежными и прибыльными. В этой главе рассказывается, как может выглядеть экосистема преуспевающих децентрализованных приложений, то есть экосистема, в которой создавать децентрализованные приложения по-настоящему просто. Я также затрону технические требования к децентрализованным приложениям и текущие возможности.

В веб-приложениях, традиционно принадлежащих миру централизованного управления, используются четыре основных понятия: идентичность, выгода, данные и вычисления. Каждое из них требует доверия к поставщику услуг — доверия, которое может быть предано. Последние достижения в области технологий создания распределенных систем позволяют включить пользователей в управление ими, поэтому давайте подробнее рассмотрим эти технологии.

Децентрализованные данные

Для меня это самое важное понятие. В настоящее время мы доверяем наши данные стекам. Мы охотно отдаем свои данные в обмен на бесплатное обслуживание. Или мы платим им за хранение наших данных — однако мы получаем достаточно данных, чтобы они приобрели какую-то ценность, только если наши пользователи отдают нам *свои* данные бесплатно! Мы верим, что стеки не будут использовать наши данные не по назначению или продавать их

субъектам, которым эти данные не предназначены. Но благодаря Эдварду Сноудену мы теперь знаем, что доверие может и будет нарушено, как только мы доверим наши данные централизованной организации. Централизованные хранилища данных — мечта полицейского государства; все данные граждан легкодоступны и могут исследоваться без ведома их владельцев. Amazon Web Services, Google Drive, Dropbox и все остальные поставщики «облачных» услуг являются такими централизованными хранилищами, несмотря на используемую ими распределенную архитектуру.

Кроме того, с переходом от экономики, базирующейся на рабочей силе, к экономике, базирующейся на информации, с ее роботами и автоматизированными технологиями, развивающимися все более быстрыми темпами, данные становятся основной ценностью. Люди не способны конкурировать с роботами в плане рабочей силы, но могут конкурировать с ними за данные благодаря своей уникальной способности анализировать их, опираясь на свое восприятие мира пятью органами чувств. Мы не просто обладаем нашими данными, мы должны владеть ими, так как мир не стоит на месте.

Но как решить эту задачу? Как организовать хранение данных децентрализованным способом, чтобы никто, кроме нас, не мог владеть ими? Эта проблема интенсивно исследуется по меньшей мере последние десять лет, и было предложено несколько вариантов ее решения. Идеальное решение должно давать возможность хранить данные децентрализованным способом максимально надежно и без необходимости доверять кому бы то ни было.

Вариант 1: хранение данных непосредственно в цепочках блоков Bitcoin

Это самый бесхитростный способ. Да, он решает проблему децентрализованного хранения данных, потому что копия цепочки блоков с этими данными может храниться у любого и никто не сможет изменить ее. Данные можно зашифровать, например, с применением алгоритма SHA-256, чтобы любой, поддерживающий хранилище, мог хранить копию ваших данных, но только вы имели бы к ним доступ посредством закрытого ключа. Однако цепочки блоков Bitcoin не предназначены для обработки больших объемов данных! Их назначение, с которым они прекрасно справляются — хранение простого журнала транзакций. Даже при такой небольшой нагрузке цепочка блоков достигла за последнюю пару лет размера в 38 Гбайт. Загрузка цепочки блоков может занять до нескольких дней, а плохая масштабируемость и разбухание цепочки блоков превратились в неразрешимую и серьезную проблему для основных разработчиков. Выгружая данные в цепочку блоков, вы вынуждаете майнеров Bitcoin хранить ваши данные бесплатно, лишая их стимула поддерживать работу сети, поскольку их затраты начинают превышать доходы.

А если организовать хранение в другой цепочке блоков с более строгими ограничениями размера, позволяющими хранить дополнительные данные? Если затраты майнеров на хранение ваших данных будут оплачиваться вашей альтернативной криптовалютой, даже тогда размер цепочки блоков вырастет до безумной величины, и каждому желающему использовать вашу криптовалюту придется загружать чрезвычайно массивное хранилище. Это решение выглядит уже угрожающим, если просто представить боль-

шое количество пользователей, сохраняющих всего по несколько изображений; но мы быстро движемся вперед, к новой эпохе распространения данных, в которой наборы данных в несколько петабайт станут обычным делом. Хранение данных в цепочке блоков не лучший способ организации надежного децентрализованного хранения данных в краткосрочной перспективе, не говоря уже о долгосрочной.

Вариант 2: хранение данных в распределенной хеш-таблице

Распределенные хеш-таблицы (Distributed Hash Tables, DHT) потеряли в популярности в последнее десятилетие. Они распределяют не только копии данных, но и функции индексирования, которые обеспечивают поиск данных и гарантируют надежность. Ранние программы обмена файлами, такие как KaZaA, Napster и Gnutella, использовали собственные версии DHT с разными уровнями децентрализации. Одни имели централизованные трекеры (trackers), следящие за перемещением всех данных, другие (такие как Napster) имели централизованные источники, через которые передавались все данные и которые представляли собой единую точку отказа (в данном случае подверженную судебным преследованиям).

Первой по-настоящему действующей реализацией DHT стал протокол BitTorrent. Он до сих пор используется более чем 300 миллионами пользователей. Несмотря на децентрализованное хранение данных (BitTorrent Mainline DHT), он все еще зависит от работы централизованных трекеров (таких как Pirate Bay), выполняющих мониторинг сети. Такие сайты, как Pirate Bay, регулярно отключаются из-за судебных преследований, поэтому даже при всей надежности прото-

кола BitTorrent он все же имеет некоторую точку отказа. А пригодны ли распределенные хеш-таблицы BitTorrent для хранения данных в децентрализованном приложении? Нет, BitTorrent не предполагает децентрализованного хранения данных; это лишь протокол доступа к распределенным данным, обладающий высокой пропускной способностью благодаря применению стратегии передачи данных между сидерами и личерами по принципу «ты — мне, я — тебе».

Протокол передачи данных BitTorrent работает намного быстрее, чем HTTP, и поэтому превратился в способ передачи больших наборов данных, таких как видеофильмы с высоким разрешением. Проблема использования BitTorrent в качестве хранилища заключается в отсутствии стимулов хранить ваши данные длительное время на множестве узлов. Сеть построена так, что наивысший приоритет отдается наиболее востребованным файлам — люди должны быть заинтересованы в копировании и хранении ваших данных в сети как можно дольше. С другой стороны, когда используется центральный сервер с хорошей репутацией, такой как Amazon Web Services, вы знаете, что ваши данные не исчезнут, даже если этими данными пользуетесь только вы, потому что речь идет об их репутации: они по соглашению обязаны хранить ваши данные и не зависят от потребности других в хранении ваших данных.

Во-первых, нам требуются не только возможность децентрализованного хранения информации в DHT и высокая скорость передачи файлов по протоколу BitTorrent, но и гарантия сохранности данных. Поэтому совершенно необходимо как-то стимулировать других, чтобы они хранили данные в том или ином виде. Во-вторых, мы должны гарантировать, что ссылки на данные останутся действительными. Эта идея не нова. Одно из первоначальных пред-

ложений об организации Интернета предполагало использование постоянных ссылок. Данная идея была названа «Проект Xanadu»[1] и заключалась в такой организации Сети, где каждая ссылка была двусторонней: с одной стороны она указывала на адресата, а с другой — на источник. Это означает, что создатель контента всегда мог бы подтвердить принадлежность своих данных, потому что он всегда имел бы обратную ссылку на них. Такая сеть так и не была построена, поэтому у нас есть только Всемирная паутина, основанная на протоколе HTTP с односторонними ссылками, с которым мы выросли и который мы знаем и любим.

А существует ли система, реализующая указанные возможности? Да. Она называется межпланетной файловой системой[2] (Interplanetary File System, IPFS: *http://ipfs.io*). Это открытый проект, который в настоящее время находится на стадии альфа-версии. Я большой поклонник IPFS и был одним из первых участников проекта. Хуан Бенет, основатель проекта, размышлял над организацией хранилища пять лет и наконец оформил свои мысли в виде научной статьи о IPFS (*http://bit.ly/ipfs-whitepaper*). Я потратил много месяцев, чтобы разобраться в системе и понять, почему IPFS лучше других решений. И в настоящий момент я чувствую, что она имеет все шансы на успех.

Целью IPFS является оказание помощи в движении к децентрализованной сети с неограниченным сроком хранения данных, то есть к сети, в которой ссылки никогда не становятся недействительными и отсутствует единый центр, контролирующий ваши данные. Загрузив клиент IPFS, пользователь получает возможность добавлять любые дан-

[1] *https://ru.wikipedia.org/wiki/Проект_Xanadu. — Примеч. пер.*
[2] *http://ru.bmstu.wiki/IPFS_(InterPlanetary_File_System). — Примеч. пер.*

ные и взамен получать хеш, открывающий доступ к этим данным. IPFS — это контентно-адресуемая система, в отличие от Всемирной паутины, которая является системой, основанной на IP-адресах. Если в системе, основанной на IP-адресах, сервер имен выйдет из строя, все ее данные станут недоступными. Контентно-адресуемая система имеет более эффективную форму адресации, потому что доступность данных в ней не зависит от работоспособности единственного сервера. Запросив контентно-адресуемые данные, вы получите их быстрее, чем IP-адресуемые, поскольку система проложит маршрут к ближайшему от вас узлу, где хранится требуемая копия.

Как реализована эта система?

Для хранения данных IPFS использует DHT. Система основана на популярной реализации Kademlia DHT и заимствует некоторые идеи из Chord DHT и BitTorrent DHT. Когда пользователь выгружает данные в IPFS, они копируются на определенное количество других узлов, чтобы даже в случае выхода из строя какого-то одного узла данные остались доступными на других узлах. Кроме того — и так же, как в BitTorrent, — чем более востребованы данные, тем выше надежность их хранения, поскольку в результате каждой загрузки создается новая копия. Главной особенностью Chord были круги DHT, в которых создавались «хорды» для расширения поиска DHT среди узлов по всему миру, находящихся в непосредственной близости друг от друга в пределах больших хорд. То есть земной шар можно представить как серию последовательно увеличивающихся хорд (рис. 2.1), и поиск в такой системе можно выполнять особенно эффективно, перепрыгивая между хордами при необходимости.

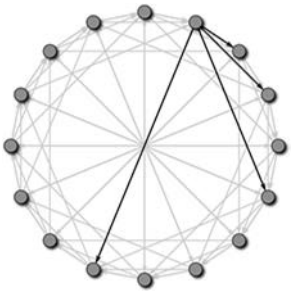


Рис. 2.1. Chord

Сейчас централизованные службы, такие как Amazon и Google, имеют дата-центры по всему миру, доступные пользователю на выбор, но обычно дата-центр выбирается автоматически. Даже при наличии дата-центров, разбросанных по всему миру, высокая эффективность передачи данных возможна только при наличии нескольких узлов, объединенных в по-настоящему децентрализованную систему, такую как Chord DHT.

Чтобы сформировать структуру DHT и дать пользователям возможность находить необходимые им данные, IPFS использует то, что называется *merkleDAG,* — простую и гибкую структуру данных, которую можно представить как серию узлов, связанных друг с другом. Если говорить точнее, это ориентированный ациклический граф (Directed Acyclic Graph, DAG). MerkleDAG можно рассматривать как связанный список или дерево. Когда в DHT добавляются данные, система генерирует пару из открытого и закрытого ключей SHA-256 и возвращает их пользователю. Разработчики могут связывать хеши программно для формирования собственного мини-merkleDAG, и в этом случае важно отметить, что все данные в IPFS формируют обобщенный

merkleDAG, включающий все узлы. Все данные в IPFS общедоступны, поэтому пользователь сам должен позаботиться об их шифровании. Владение закрытым ключом, открывающим доступ к данным, может служить доказательством права собственности на данные.

IPFS (архитектура которой показана на рис. 2.2) унаследовала скорость передачи BitTorrent и механизм поиска ближайших узлов, и разработчики IPFS полагают, что Всемирная паутина должна действовать аналогично. Возьмем для примера компьютерный класс в школе, каждый компьютер которого выполняет запрос к центральному серверу Facebook для получения видеофайлов. В этом случае понапрасну расходуется значительная доля полосы пропускания. Если файл уже имеется поблизости, он не должен загружаться издалека. В контентно-адресуемой системе, где адрес местонахождения контента известен, файлы могут

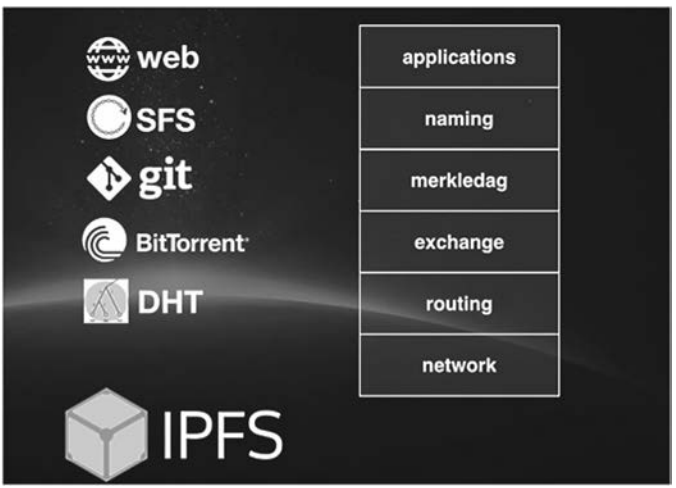


Рис. 2.2. IPFS

загружаться с узлов, находящихся поблизости. Узлы могут самостоятельно обмениваться информацией, без координации из центра; схема принимает за основу модель клиент-сервер, на которую опирается Всемирная паутина, и превращается в распределенную, по образу и подобию BitTorrent.

Какие улучшения привносит IPFS в сравнении с BitTorrent?

IPFS имеет родственный протокол Filecoin (*http://filecoin.io/filecoin.pdf*). Filecoin применяется для оплаты услуг майнеров (miners — узлы, хранящие данные) с помощью новейшего механизма «деньги за данные», который называется BitSwap. Здесь имеет смысл использовать криптовалюту: перевод средств осуществляется быстро, что позволяет производить микроплатежи за каждый байт в хранилище. В настоящее время Filecoin находится в стадии разработки, но IPFS уже доступна для использования. Операции в IPFS пока выполняются бесплатно, и майнеры хранят данные, что называется, «из любви к искусству». Однако рано или поздно все операции загрузки или выгрузки должны будут оплачиваться криптовалютой Filecoin. Вероятнее всего, Filecoin будет основана непосредственно на цепочках блоков Bitcoin, благодаря чему пользователи смогут оплачивать услуги хранилища монетами Bitcoin.

В довершение всего IPFS заимствует модель управления версиями из Git для моделирования версий всех данных. С этой целью в Git используется ориентированный асимметричный граф (DAG); IPFS использует его для структурирования всей системы. Пользователи смогут увидеть историю изменения своих данных (или любых других данных, доступных им в дешифрованном виде).

Итак, IPFS заимствовала лучшие идеи из Git, DHT, SFS, BitTorrent и Bitcoin и объединила их для создания децентрализованной сети хранения данных. Разработчики IPFS надеются, что однажды протокол IPFS:// заменит HTTP://, но уже сейчас они могут работать слаженно, о чем мы поговорим, когда приступим к обсуждению реализации.

IPFS является наиболее надежным, тщательно продуманным решением для организации децентрализованного хранилища из всех имеющихся проектов криптовалют, хотя в этой области есть достойные конкуренты, заслуживающие внимания. Давайте их рассмотрим.

Ethereum Swarm

Ethereum (*https://ethereum.org*) — это проект, целью которого является создание универсального (Тьюринг-полного) языка для вычисления цепочек блоков, включая децентрализованные хранилища. На момент написания этих строк (2016) усилия разработчиков были сосредоточены на обеспечении безопасности ДАО (в их терминологии «демократичная автономная организация»), а реализация собственно хранилища отодвинута на второй план.

StorJ

Проект StorJ (*https://storj.io*) активно развивается в последнее время; он имеет большой объем майнинга монет StorJcoins и замечательную архитектуру. Архитектура великолепна — она победила на хакатоне в городе Остин (США) — и, похоже, разработчики знают свое дело. Но, несмотря на все это, спустя год после хакатона проект не продвинулся дальше рекламной шумихи.

Maidsafe

Проект Maidsafe (*https://github.com/maidsafe/White-papers/blob/master/Project-Safe.md*), подобно Ethereum, стремится решить сразу несколько задач. В нем не используется принцип доказательства выполнения работы (proof-of-work) и основной целью является создание децентрализованной платформы для вычислений, хранения и криптовалюты. Разработчики трудятся над своей платформой уже шесть лет и, похоже, без особого прогресса.

Децентрализованные ценности

Система Bitcoin стала первым децентрализованным хранилищем ценностей, добившимся успеха. До появления Bitcoin для передачи ценностей через Всемирную паутину было необходимо доверие к стороннему поставщику услуг (банку). Bitcoin позволила децентрализовать перевод средств и восполнила потребность децентрализованных приложений в механизме децентрализованных платежей.

А что можно сказать обо всех альтернативных криптовалютах, таких как Litecoin, Dogecoin, Peercoin, Darkcoin и Kanyecoin? Вообще альтернативные криптовалюты создаются путем клонирования исходного кода Bitcoin и добавления дополнительных возможностей, которые по разным причинам были отвергнуты разработчиками Bitcoin. Например, создатель криптовалюты Litecoin (*https://litecoin.com*) стремился получить более высокую скорость платежей, чем в Bitcoin, поэтому он взял за основу проект Bitcoin, добавил некоторый код, ускоряющий операции, и получил Litecoin.

Litecoin имеет довольно большую рыночную капитализацию и по крайней мере год удерживается в первой пятерке криптовалют. Litecoin — редкий пример альтернативной криптовалюты, реализованной с похвальной целью: ускорение платежей. Большинство других альтернативных криптовалют не могут этим похвастаться: если они не были созданы ради развлечения, превратившись в мемы с деньгами за ними (как Dogecoin), они служат для реализации схем «накачки и сброса». Идея проста: некто создает новую криптовалюту, шлепает на нее ярлык и раскручивает ее с помощью средств массовой информации (kanyecoin). Он заявляет, что его криптовалюта имеет высокую ценность и что инвесторы смогут получить высокую прибыль, купив акции как можно раньше, потому что с течением времени стоимость криптовалюты неуклонно увеличивается.

Как только криптовалюта приобретет достаточную ценность, создатели продают ее за более стабильную криптовалюту, такую как Bitcoin, или за фиатные деньги. Это — типичная схема в мире альтернативных криптовалют, пагубно влияющая на всю экосистему криптовалют по очевидным причинам. Во-первых, такие криптовалюты пятнают репутацию криптовалют в целом, заставляя потенциальных инвесторов остерегаться этой области. Во-вторых, они составляют цепочкам блоков Bitcoin ненужную конкуренцию за долю на рынке, не привнося никакой ценности, что, в свою очередь, наносит вред стоимости Bitcoin и всех других широко используемых криптовалют, основанных на Bitcoin.

Bitcoin, как известно, использует схему доказательства выполнения работы, которая предполагает, что каждый майнер в сети должен сгенерировать доказательство затраченной им вычислительной мощности и обработки транзакций; в ответ

майнеры получают монеты Bitcoin как плату за поддержку сети. Некоторые люди, обращающиеся в сфере криптовалют, считают доказательство выполнения работы чрезмерно энергоемким и краткосрочным решением противостояния атаке Сибиллы, поэтому исследования криптовалют в области механизмов консенсуса не прекращаются. Алгоритм доказательства выполнения работы потребляет значительную вычислительную мощность, и общая стоимость электроэнергии, затраченной майнерами на поддержание сети, превышает 15 миллионов долларов США. Это чересчур расточительно, особенно если учесть, что теоретически можно было бы использовать более удачные решения для поддержания сети. К таким решениям относятся, например, две популярные альтернативы, предложенные взамен доказательства выполнения работы: подтверждение доли (proof-of-stake) и делегированное подтверждение доли (delegated proof-of-stake).

Несмотря на продолжающиеся исследования новейших механизмов консенсуса, таких как подтверждение доли (proof-of-stake), пока не найдено решение, столь же устойчивое к атакам Сибиллы, как алгоритм доказательства выполнения работы (proof-of-work). Хоть это и дорогостоящее решение, лучше его пока ничего не найдено. В сеть Bitcoin вложено больше трех миллиардов долларов США, и эту валюту принимает бесчисленное множество стартапов, инвесторов, средств массовой информации и торговых предприятий. Она имеет преимущество первопроходца и за пять с лишним лет завоевала признание широких масс. Нам не нужно начинать все сначала. Даже если появится новый механизм консенсуса, превосходящий алгоритм доказательства выполнения работы, нам следует полагаться на его реализацию основными разработчиками Bitcoin, а не на

создание альтернативной криптовалюты, тогда мы сможем развиваться быстрее как сообщество.

Эту идею можно назвать Bitcoin-максимализмом. Bitcoin-максималисты всячески рекламируют преимущества первопроходца, которыми обладает Bitcoin, и надежную защиту инвестиций в сеть. Отрицательным следствием Bitcoin-максимализма является то, что любые идеи, выходящие за рамки протокола Bitcoin, насколько бы ценными они ни были, быстро искореняются и не получают должного признания в сообществе; в результате прогресс стоит на месте.

Существует решение, сочетающее в себе лучшее из двух миров: оно называется *предложением боковой цепочки* (sidechain proposal). Идея предложения боковой цепочки основывается на статье (*http://bit.ly/back-sidechain*), написанной Адамом Беком с соавторами, создателем алгоритма доказательства выполнения работы согласно ссылке в статье Сатоси о Bitcoin (*https://bitcoin.org/bitcoin.pdf*). Данное предложение начинается с констатации факта, что для проведения экспериментов с механизмами консенсуса и любыми современными криптовалютами разработчики должны заимствовать реализацию цепочек блоков Bitcoin и на ее основе создавать совершенно новую реализацию цепочек с целью проверки своих гипотез.

Это плохо для Bitcoin и сложно для разработчиков. В качестве решения команда Бека предложила код, который позволяет свободно перемещать монеты Bitcoin между главной (цепочкой блоков Bitcoin) и боковой цепочками. Это означает, что вы легко можете создать совершенно новую цепочку блоков и подключить ее сбоку к цепочке блоков Bitcoin. В результате вы получаете безопасность алгоритма доказательства выполнения работы Bitcoin и из-

бавляетесь от необходимости создавать и настраивать собственную сеть. Дополнительно вы можете привлечь людей, уже инвестировавших в эксперимент с криптовалютой (людей, владеющих средствами в Bitcoin), в качестве потенциальных пользователей вашей цепочки, потому что они могут просто использовать уже имеющиеся у них монеты. Наконец, вы можете передавать монеты между двумя цепочками без конверсии. Работы над созданием двусторонних боковых цепочек уже ведутся и будут завершены в самое ближайшее время.

Цепочка блоков Bitcoin является наиболее защищенной цепочкой блоков; ее вычислительная мощность превышает мощность всех суперкомпьютеров в мире, вместе взятых, поэтому она считается самой устойчивой к атакам Сибиллы. Развертывание с нуля цепочки, использующей алгоритм доказательства выполнения работы, — сложная задача; так как вычислительные ресурсы на начальном этапе слишком малы, злоумышленник без особого труда сможет создать 51% общей вычислительной мощности сети и перехватить управление ею. Кроме того, разработчики не должны беспокоиться о развертывании цепочки блоков вдобавок к решению и без того сложной задачи создания децентрализованного приложения. Боковые цепочки предлагают приемлемое решение всем желающим поэкспериментировать с механизмами консенсуса или реализовать свою, новую криптовалюту.

А как быть, если вы не хотите создавать новую криптовалюту, а просто желаете выпустить собственную, внутреннюю криптовалюту для нужд децентрализованного приложения? Валюта, стоимость которой растет с расширением сети, позволяет пользователям обращаться к дефицитным ресурсам и мотивирует их на расширение сети? Тогда нет необходи-

мости создавать новую (боковую) цепочку. Можно просто создать ресурс непосредственно в Bitcoin. Примером может служить мой проект Colored coins (окрашенные монеты), хотя, как обычно, он имеет несколько примечательных альтернатив.

Counterparty

Counterparty (*https://counterparty.io*) — это протокол Bitcoin 2.0, дающий пользователям возможность создавать активы и управлять ими, выставлять на продажу и покупать их и даже создавать Тьюринг-полные контракты поверх Bitcoin. Звучит заманчиво, но проблема состоит в том, что Counterparty запекает все эти интересные возможности в протокол, вместо того чтобы выделить их в модули и организовать многослойную структуру. Возможность выпуска активов в цепочке Bitcoin и передачи их между пользователями с легкостью Bitcoin — отличная идея. Но она объединена с функцией дивидендов. Функция распределения дивидендов — достаточно хорошая особенность, но она является внутренней операцией Counterparty и не использует имеющиеся средства Bitcoin для контроля за активами. Вследствие всего этого протокол оказался чересчур перегруженным.

Торги — это пример добавления одной экспериментальной и сложной особенности к другой: активам. Но лучше было бы реализовать простые слои, каждый для чего-то одного. Модульность — отличительная черта хорошего программного обеспечения, а реализация Counterparty, несмотря на амбициозность, вообще не разбита на модули. Вообразите рынок библиотек разных протоколов, где эти библиотеки конкурируют друг с другом и лучшие

выходят победителями. А теперь представьте, что все эти полезные библиотеки безнадежно переплетены между собой. Вам придется установить либо их все, либо ни одной. Это был бы кошмар, и именно так реализован проект Counterparty.

Counterparty вводит нежелательный и запутанный элемент для тех разработчиков, которые планируют использовать его: валюту XCP. Если вы захотите создать внутреннюю валюту для своего приложения с использованием Counterparty API, вам придется иметь дело с XCP и всеми сопутствующими преобразованиями. Если вы пожелаете создать какой-либо актив, соответствующий протоколу, вам придется уничтожить 0,5 XCP — больше доллара США в текущих ценах. Денежная масса XCP фиксирована, а поскольку валюта постоянно уничтожается, когда кто-то выпускает новый актив, вся денежная база постепенно сжимается.

Сам факт существования валюты XCP и обязательность ее применения для доступа к некоторым возможностям Counterparty раздражают разработчиков. Чтобы ее использовать, вам придется постоянно следить за колебаниями курса (XCP/BTC). Да, существуют целые платформы, помогающие следить за курсом, в режиме реального времени следить за торгами и спросом на ликвидность, как на любом рынке. Только какой в этом смысл? Почему вы должны иметь дело со всем этим, когда вам нужно лишь создать внутреннюю валюту для своего приложения? Это серьезное препятствие, в действительности никому не нужное, и плохая идея.

Проект Counterparty постоянно обновляет свой клиент, из-за чего другие проекты, опирающиеся на его API,

такие как приложение Gems (*http://getgems.org*), получают неоднозначные результаты. Вследствие отсутствия модульности в проекте в случае возникновения ошибки рушится вся его конструкция. В общем и целом Counterparty имеет чересчур централизованное управление; существуют лучшие альтернативы (colored coins — окрашенные монеты) с модульной организацией и децентрализацией без использования дополнительной валюты.

Hyperledger

Проект Hyperledger (*https://www.hyperledger.org*) определяет себя как «независимый от типа валюты», в том смысле что позволяет эмитенту выпускать свои монеты, не основанные на какой-то валюте, то есть ни на Bitcoin, ни на фиатной валюте, ни на какой-то другой криптовалюте. Звучит неплохо, но в действительности это не так, потому что в основе лежит неизвестный механизм консенсуса. Мы уже познакомились со многими исследованиями в обсуждаемой области, но пока не видели ничего, что показало бы себя таким же универсальным, как алгоритм доказательства выполнения работы.

Один из простейших способов рассеять рекламный туман, окружающий любой проект Blockchain 2.0, — определить, какой механизм консенсуса в нем используется. Если он не использует алгоритм доказательства выполнения работы или не опирается на Bitcoin, посмотрите, насколько велика его рыночная капитализация и как много брешей в его системе безопасности было выявлено. Всякий раз, когда я иду таким путем, я натыкаюсь на бреши в системе безопасности. В табл. 2.1, представленной Михером Роем, кратко изложены различные убеждения, существующие в этой области.

Таблица 2.1. Политические убеждения в мире криптовалют

Убеждение/ утверждение	Платформа	Риски	Достоинства
Уровень I	Не применимо	Не применимо	Не применимо
Независимость от валюты	Hyperledger, Eris, Codius, Ripple/ Stellar	Отсутствуют решения для управления идентичностью и закрытым ключом. Нормативная неопределенность, обусловленная управлением транзакциями конечными пользователями. Недостатки, характерные для конкретной платформы, такие как алгоритм консенсуса	Поддержка любых активов, включая фиатные деньги, акции и криптовалюты. Позволяет воспроизвести все приложения, разработанные сообществом пользователей криптовалют. Относительная совместимость с существующими правилами
Максимализм криптовалюты	Bitcoin, Ethereum, Tendermint, Pebble, Ripple/ Stellar (отчасти) и т. д.	Для преодоления инерции отношения общества к новым формам ценностей требуется массовый эффект. Пока нет системы с эффективной кредитно-денежной политикой и методом консенсуса, высокой скоростью сделок и масштабируемостью. Связанные политические идеологии мешают росту	Существует сегмент рынка, недовольный традиционной банковской системой и готовый к переходу на криптовалюту. Значительный общественный интерес на данный момент
Bitcoin-максимализм	Боковые цепочки	Новые технологии, увеличивающие затраты на содержание сети, скорость сделок и масштабируемость вытесняют Bitcoin	Значительное преимущество Bitcoin как первопроходца
Гипербиткоинизация (Hyperbitcoinization)	Не применимо	Убеждение оказывается заблуждением	Нет

Независимость от валюты — это четкий набор представлений, но я считаю, что Bitcoin можно интегрировать в существующие финансовые системы. Мы миновали медовый месяц с Bitcoin, в течение которого многие думали, что эта криптовалюта займет доминирующее положение в мировом валютном ландшафте, и мы также поняли, что для банковской системы всегда найдется место, несмотря на то, что она устарела.

А как насчет Тьюринг-полных умных контрактов? Это вторая часть финансового инструмента, необходимого для создания децентрализованной платежной системы в децентрализованном приложении. Команда Ethereum достигла, пожалуй, наибольшего прогресса в данном направлении, но и цели они себе ставили амбициозные: создание Тьюринг-полной цепочки блоков, децентрализованной сети хранения данных, децентрализованного протокола взаимодействий, нового механизма консенсуса и новой (раскрученной) цепочки блоков, нового браузера для выполнения децентрализованных приложений на основе Ethereum и нового языка сценариев для программирования таких децентрализованных приложений.

Давайте вернемся на мгновение назад. Одна команда не может и не должна пытаться решать самостоятельно подобные задачи, достойные целой компании. Проект Ethereum собрал приличную сумму денег и массу восторгов, но, несмотря на гений Виталика Бутерина, я не думаю, что им удастся создать второй Bitcoin. Как сказал главный разработчик протокола Bitcoin Гэвин Андерсен, или они увязнут в решении проблем с безопасностью, или их цепочка блоков увеличится в разы.

Идея Тьюринг-полного языка сценариев полезна; такой язык позволит вам реализовать все, что требуется. Язык сценари-

ев Bitcoin был намеренно ограничен в возможностях, чтобы предотвратить создание некорректных сценариев (по злому умыслу или по неопытности), таких как бесконечные циклы. Гэвин Андерсен отмечал, что многие цели проекта Ethereum можно было бы реализовать в Bitcoin, и основные разработчики уже приступили к реализации некоторых из них.

В данной книге, посвященной децентрализованным приложениям, нас будут интересовать только создание активов и умные контракты; ставки, деривативы и протоколы, требующие отдельной валюты, не будут обсуждаться. С практической точки зрения, самый простой способ привлечь людей к использованию вашего децентрализованного приложения — гарантировать совместимость с валютой, которой они уже владеют, а самой известной из них является Bitcoin. Поэтому вы должны выпустить окрашенную монету, основанную или на Bitcoin, или на боковой цепочке, которая по сути является тем же Bitcoin с некоторыми дополнительными особенностями, такими как высокая скорость совершения сделок.

Децентрализованная идентичность

Понятие идентичности обсуждается в течение многих столетий, но в эпоху Интернета этот термин приобретает совершенно новый смысл. Что такое идентичность? Кто имеет идентичность? Как выглядит идентичность в Интернете?

Благодаря последним достижениям в области криптографии многие решения переходят на использование «инфраструктуры открытого ключа». Она предполагает хранение закры-

тых ключей у людей в надежном месте и децентрализацию идентификации. Только обладатель ключа сможет подтвердить свою идентичность. Хорошим примером является протокол BitAuth (*https://github.com/bitpay/bitauth*). BitAuth использует технологии Bitcoin для создания пары открытого и закрытого ключей с применением алгоритма secp256k1, открывает возможность аутентификации без пароля между веб-службами и дает вам идентификационный номер системы (System Identification Number, SIN) — хеш открытого ключа. Протокол BitAuth использует подписи для предотвращения атак вида «незаконный посредник» (man-in-the-middle, MITM) и одноразовое число для предотвращения повторных атак. Ваш закрытый ключ никогда не передается на сервер, и вы можете хранить его в надежном и безопасном месте. Идентификация осуществляется децентрализованно, поэтому нет нужды пользоваться услугами третьей стороны для хранения своей идентичности, вы сможете хранить ее у себя.

Имели место также попытки консолидации идентичности в Интернете с различной степенью успеха. Одной из наиболее известных таких попыток является протокол OpenID (*https://openid.net*). OpenID — это протокол децентрализованной идентификации, основанный на существующих веб-протоколах, таких как HTTP, SSL и URI. Идея состоит в том, что идентичность уже разбросана по Всемирной паутине, и с помощью протокола OpenID пользователи могут преобразовывать существующие идентификаторы URI в учетную запись для использования на любых сайтах, поддерживающих OpenID.

OpenID избавляет от необходимости связывать идентичность с поставщиком услуг, чтобы вы могли использовать единственный авторитетный источник для подтверждения

своей идентичности в разных службах. Данная попытка консолидации идентичности выглядит наиболее успешной на текущий момент: к числу провайдеров, поддерживающих OpenID, относятся такие компании, как Google, Yahoo! и Twitter. С одной стороны, это хорошо: теперь мы можем подтверждать свою идентичность на разных сайтах без необходимости регистрироваться снова и снова. Это хорошо не только потому, что удобнее, так как отпадает необходимость вводить информацию о себе снова и снова, но и потому, что исчезает необходимость доверять новым службам хранение данных, удостоверяющих личность. Но OpenID все же имеет потенциальную уязвимость, потому что вы должны доверить свои данные одному из поставщиков услуг.

Эта проблема также известна как треугольник Зуко (Zooko's triangle, рис. 2.3); для ее решения была разработана система хранения произвольных комбинаций вида «имя-значение» Namecoin (*http://www.namecoin.info*).

Рис. 2.3. Треугольник Зуко

Треугольник Зуко отражает гипотезу, утверждающую, что любой протокол способен достичь только двух характеристик из трех желаемых (значимость для человека, децентрализация, защищенность). Протокол OpenID решает проблемы защищенности и значимости для человека. Система Namecoin дополняет его децентрализацией. По сути Namecoin — это сторонняя служба идентификации и цепочка блоков, которую можно использовать в качестве посредника между вами и службой, запрашивающей вашу идентичность. Цепочка блоков Namecoin была одним из первых клонов цепочки блоков Bitcoin и выдержала испытание временем благодаря своей ценности.

В то время как большинство альтернативных криптовалют исчезает со сцены, Namecoin остается, потому что никакие другие инновации не смогли заполнить треугольник Зуко. Пользователь может зарегистрировать свое имя в цепочке блоков Namecoin, послав транзакцию с желаемым именем в пространство имен */id*. После этого Namecoin проверит ее уникальность и сохранит, если проверка пройдет успешно, или отвергнет в противном случае. Это означает, что пространство имен не позволяет выбрать одно и то же имя дважды. Пользователи могут регистрировать свои собственные (человеческие) имена, но может возникнуть новая проблема, проблема ограниченности круга человеческих фраз. Фрагментация идентичности имеет в этом отношении дополнительные преимущества, поскольку в каждой новой службе вы оказываетесь в совершенно новом пространстве имен.

Иными словами, это вынужденный компромисс: универсальный поставщик, заполняющий треугольник Зуко, — важное нововведение, но пространство имен ограничено. Как бы то ни было, то же относится и к регистрации до-

менных имен. В настоящее время регистрацией имен управляет ICANN — централизованная организация, входящая в состав Министерства торговли США. (На момент написания этих строк существовало предварительное соглашение, что данная организация официально станет самостоятельной в сентябре 2016 года.) Система Namecoin хорошо зарекомендовала себя в роли децентрализованной альтернативы организации ICANN для домена *.bit*. Сайты в домене *.bit* недоступны из обычных браузеров, таких как Chrome и Firefox. Чтобы попасть на них, пользователь должен воспользоваться веб-прокси *.bit* или установить расширение. Возможно, что с ростом популярности *.bit* поддержка протокола будет встроена в браузеры.

Большинству не нужен домен в *.bit*; он добавляет дополнительное препятствие между вами и конечными пользователями, поскольку требует установки дополнительного программного обеспечения или использования прокси-сервера для доступа к веб-сайту. Однако, если вы захотите сообщить другим некую информацию, распространению которой препятствует ваше правительство или другая организация, домен *.bit* избавит вас от интернет-цензуры.

Регистрация нового имени пользователя в цепочке блоков Namecoin осуществляется относительно просто. Вы просто обмениваете несколько монет Bitcoin на Namecoin, загружаете кошелек и регистрируете имя. Но как выполняется вход с применением цепочки блоков Namecoin? Как осуществляется аутентификация и авторизация? Недавно была создана служба NameID (*https://nameid.org*). Она объединяет лучшее из двух миров; пользователь может использовать свой идентификатор Namecoin */id* для входа на тысячи веб-сайтов, поддерживающих OpenID. Это способствует выходу Namecoin на широкий рынок.

Так в чем же проблема децентрализованной идентичности? В том, в чем и для децентрализованных данных и ценностей с IPFS и Bitcoin соответственно: пользователь вынужден хранить свой закрытый ключ. Это нормально для хакеров, обожающих децентрализованность и скрытность. Они являются наиболее идеологизированной частью общества, когда дело доходит до выбора правильных инструментов для работы в Интернете. Хакеры гордятся своим стремлением к использованию эффективных и совершенных инструментов. Они шифруют свою переписку с помощью GPG и используют клиент Tor для защиты своей истории перемещений в сети от любопытных провайдеров и правительственных структур.

Хакеры не видят проблемы в хранении дополнительных закрытых ключей ради децентрализации. А как быть с обычными людьми? Им правда все это интересно? Я не думаю, что скрытность и децентрализация для них так важны. Если учесть при этом средний уровень компьютерной грамотности и безопасности, то, мне кажется, можно смело утверждать, что большинство не способно или не готово обеспечить безопасное хранение закрытых ключей. Рыночный спрос на централизованное хранение объясняет успех Coinbase (*https://coinbase.com*), крупнейшего приложения Bitcoin. Coinbase — полная противоположность децентрализации: это банк для Bitcoin. Coinbase предлагает услугу по хранению закрытых ключей.

Бóльшая часть сообщества Bitcoin против централизации в любом ее виде; некоторые избегают даже намека на централизацию, например в форме трекеров в BitTorrent. Главный вопрос заключается в следующем: как далеко вы готовы зайти, чтобы децентрализовать свое программное обеспечение? Готовы ли вы децентрализовать свое доменное

имя и принудить своих пользователей хранить три отдельных комплекта ключей? Ответы на эти вопросы во многом зависят от целевой аудитории и от преимуществ, приобретаемых от децентрализации. Если речь идет о чем-то вроде децентрализованного Dropbox, конкурент современного Dropbox вполне может ответить «да». Если появится такой конкурент, обещающий децентрализацию данных с теми же преимуществами их безопасности, я более чем уверен, что найдется достаточно людей, которые решат, что такая система является отличным местом для хранения закрытых ключей, и поддержат ее.

Независимо от желания людей рано или поздно появится служба вида Storage-as-a-Service (хранение как услуга). Хочу признаться, что, даже будучи давним пользователем Bitcoin, для хранения своих монет я пользуюсь службой Coinbase. Я просто не хочу волноваться, что кто-то украдет их из моего компьютера! Я готов доверить Coinbase много больше, потому что эта служба хранит большие объемы активов других пользователей, я лично доверяю ее генеральному директору (по крайней мере больше, чем Марку Карпелесу[1], руководителю Mt Gox), и она обеспечивается двумя авторитетными инвесторами (Andreessen Horowitz и Union Square Ventures).

Я не считаю, что мы обязательно должны создать систему, не нуждающуюся в доверии. В этом отношении мне нравится пример с поездом. Представьте поезд, следующий из Сан-Франциско в Лос-Анджелес и в какой-то момент попадающий в аварию. Если поезд находится под централизованным управлением машиниста, всем понятно, на ком лежит от-

[1] Арестован в 2015 году в Японии по обвинению в краже криптовалюты. — *Примеч. пер.*

ветственность (на машинисте). Если управление поездом полностью децентрализовано и распределено между пассажирами, никто не может быть привлечен к ответственности персонально, и будет очень трудно найти виновника.

Децентрализация не является самоцелью; она должна иметь реальное применение. Децентрализованные приложения могут отличаться по уровню децентрализации в зависимости от области применения. Если говорить о децентрализованном приложении, использовавшемся протестующими в Китае, такое приложение, безусловно, должно быть децентрализованным сверху донизу. Если вашей предметной областью являются социальные сети, использование домена *.bit* будет, пожалуй, не самым лучшим решением.

Итак, если вы используете децентрализованное приложение, хранящее данные в IPFS и выпускающее внутреннюю валюту — окрашенные монеты в цепочке Bitcoin, — вы, возможно, также захотите использовать NameID для хранения своей идентичности. В этом случае у вас будет три комплекта ключей, хранящихся в некотором локальном или стороннем хранилище, для доступа к приложению и работы с ним.

Децентрализованные вычисления

Итак, мы обсудили децентрализованные идентичность, данные и ценности, а что можно сказать о вычислениях? Можно ли хранить децентрализованное приложение непосредственно в IPFS и запускать его оттуда? И да, и нет. IPFS — это всего лишь файловая система, и, как любая файловая система, она с успехом может использоваться для хранения статических веб-сайтов и обслуживания запросов

от веб-браузера. Но если говорить о серверных приложениях — динамических приложениях, требующих наличия командной оболочки и вычислительного окружения, таких как Node.js и Ruby on Rails, — для таких приложений использовать IPFS не получится. А если данные приложения хранятся в IPFS, где тогда хранить его исходный код?

Есть два ответа на этот вопрос. Первый ответ такой: данные хранить в IPFS, а исходный код — в традиционной виртуальной машине (ВМ) для веб-приложений, такой как Heroku. ВМ эмулирует конкретную компьютерную систему. В своей работе ВМ опирается на особенности и архитектуру гипотетического или существующего в действительности компьютера. Реализации ВМ могут включать специализированное программное и/или аппаратное обеспечение. Heroku — популярная служба «платформа как услуга» (Platform-as-a-Service, PaaS), предоставляющая возможности ВМ. Она способна выполнять динамический серверный код на Go и Node.js, а также хранить ваши данные с привлечением внутренней службы хостинга, использующей базу данных, такую как MongoDB.

Если исходный код хранить в Heroku, а данные в IPFS, пользователи поверят, что данные принадлежат только им, и вы не сможете продать данные кому бы то ни было. Но пользователи едва ли поверят, что открытый вами исходный код действительно является тем кодом, что выполняется на сервере. Помимо отсутствия доверия это также означает, что имеется центральная точка отказа (Heroku). Второй путь — хранить данные пользователей в IPFS и развернуть исходный код в децентрализованной ВМ, основанной на использовании IPFS. Это возможно? Да, именно для такого варианта создан проект Astralboot (*https://github.com/ipfs/astralboot*). По сути это сервер с поддержкой языка Go,

который хранит свои файлы непосредственно в IPFS и позволяет получить окружение Debian, основанное на IPFS. Если развернуть динамическое приложение поверх Astralboot, оно будет размещено в IPFS, и вам останется только настроить свое конкретное окружение поверх окружения Astralboot в Linux.

Существуют и другие возможности, например виртуальная машина Ethereum (Ethereum Virtual Machine, EVM). Цепочка блоков Ethereum имеет много отличий от цепочки блоков Bitcoin: другая организация блоков, Тьюринг-полные умные контракты, и действует она как децентрализованный конечный автомат (state machine). Я привык считать Ethereum виртуальной машиной, однако не полной и уж точно не той, что нужна большинству разработчиков. Извлечение данных из сторонних источников — практически неизбежная необходимость на современном рынке программного обеспечения; существует масса служб, конкурирующих в сфере сбора данных и предлагающих свои API для доступа к другим службам. Вместо того чтобы снова и снова изобретать велосипед и создавать для вашего приложения источники данных, пользующиеся доверием, предпочтительнее использовать сторонние API. Проблема виртуальной машины EVM в том, что она не позволяет извлекать данные из-за пределов цепочки блоков, если такая возможность не предусмотрена умными контрактами на сервере.

Она прекрасно подходит для новых API, действующих как «оракулы» (доверенные федеративные источники данных), но не годится для уже существующих служб. Ни Ethereum, ни Bitcoin не способны запрашивать данные за пределами самих себя. Это ограничение реализовано намеренно, с целью безопасности. Если бы имелась возможность вызывать

внешние API из цепочки блоков, злоумышленник мог бы обогнать цепочку блоков, отправляя различные запросы, и в конечном счете раздуть сеть. Таким образом, использование цепочки блоков в качестве ВМ — не лучшее решение.

Еще один проект, Go-circuit (*http://gocircuit.github.io/circuit*), создает маленькие серверные процессы на машинах в кластере. Они образуют надежную и эффективную сеть, которая позволяет распределенным процессам взаимодействовать и синхронизироваться с любой одиночной машиной. Go-circuit — распределенная среда выполнения, способная интегрироваться с Docker и предназначенная для программ на языке Go. Это отличное решение, если ваш проект написан на Go, иначе оно вам не подойдет.

В ходе рассуждений о децентрализованных вычислениях возникает вопрос: что может служить площадкой для таких вычислений? Представьте боковую цепочку, в которой алгоритм доказательства выполнения работы (proof-of-work) фактически выполняет некоторую полезную работу для сети — некоторые криптовалюты, такие как Gridcoin (*https://gridcoin.us*) и Primecoin (*https://primecoin.io*), уже идут этим путем. Но они не несут никакой пользы (способом, поддающимся проверке) для новых вычислений, необходимых пользователям; они основываются на вычислениях, необходимых создателям монет. Что действительно необходимо, так это пиринговые децентрализованные вычисления с простым и доступным интерфейсом, с помощью которого разработчики децентрализованных приложений могли бы развертывать свой код.

Необходимая нам сеть могла бы иметь собственную монету, а разработчики децентрализованных приложений платили бы ею майнерам (miners), движущей силе вычислений,

за выполнение их кода. Чем больше пользователей в такой сети, тем выше ценность сети и, соответственно, ценность монет. Это область исследований для криптологов, и я надеюсь, что уже в ближайшее время такие проекты, как Astralboot, который активно развивается, позволят нам увидеть действующие прототипы.

Даже если Astralboot окажется слишком сложным в настройке, у вас есть Heroku; если Heroku не подойдет и ваш код открыт, любой желающий сможет просто выгрузить его на новый сервер, постоянно доступный, с частными и общедоступными данными, поддающимися проверке и находящимися в IPFS. С появлением вычислительной площадки вы получите возможность запускать динамические приложения прямо из браузера, просто набрав доменный адрес в адресной строке.

Децентрализованные сети

До сих пор мы говорили о четырех основных направлениях децентрализации в децентрализованных приложениях: идентичности, ценностях, данных и вычислениях. Мы говорили о регистрации доменов, а также о том, что это требуется далеко не во всех случаях. Давайте теперь обсудим децентрализованные сети, чтобы тоже добавить их в указанный список.

Большинство пользователей подключаются к Интернету, пользуясь услугами провайдеров, которые играют роль шлюза между пользователями и Интернетом (рис. 2.4). Интернет-провайдеры в разных странах помогают людям быть на связи и выступают в роли централизованного хаба;

Рис. 2.4. Интернет

кроме того, они решают проблему последней мили, подключая конечных пользователей к высокоскоростным магистралям Интернета с высокой пропускной способностью. Термин «последняя миля» в телекоммуникационной индустрии обозначает последний фрагмент телекоммуникационной сети, обеспечивающий подключение конечного клиента. Обычно это сетевой кабель, проведенный в квартиру или офис клиента.

Интернет-провайдеры, такие как AT&T и Comcast, приносят большую пользу, давая доступ к Интернету там, где в настоящее время отсутствуют иные альтернативы. Но эти централизованные шлюзы также являются центральными точками отказа. Правительства могут закрыть их по своей прихоти или потребовать от провайдеров закрыть доступ к некоторым IP-адресам. В ответ, чтобы соблюсти законодательство страны, те будут вынуждены создать черный список IP-адресов, которые должны быть недоступны пользователям. И такие случаи были: во время событий «арабской весны» или в Гонконге, когда пользователи всерьез опасались закрытия провайдеров правительством. Некоторые правительства действительно осуществляли цензуру, регламентируя деятельность этих шлюзов с целью подавления восстания.

Все устроено именно так, потому что нет альтернатив, но они начинают появляться. Свежим примером является Firechat (*https://opengarden.com*) — приложение для iOS, созданное компанией Open Garden. Firechat позволяет владельцам телефонов общаться друг с другом напрямую, используя новую возможность в iOS 7 с названием Multipeer Connectivity (многосторонние соединения). Для ее работы не требуется интернет-провайдер. Firechat является при-

мером приложения, использующего меш-сеть[1]. Меш-сети — это децентрализованная версия стандартного, централизованного Интернета. Для работы в меш-сети пользователю не требуется соединяться с центральным шлюзом, чтобы получить доступ к сайту; он подключается непосредственно к ближайшему маршрутизатору, роль которого может играть ближайший компьютер.

Существует уже большое количество *меш-сетей*; самая крупная находится в Испании, она насчитывает более 50 тысяч пользователей, желающих иметь доступ к Интернету, который не может предоставить ни один интернет-провайдер. Эта меш-сеть действует и поныне. Во время урагана в Нью-Йорке меш-сеть использовалась для передачи ценной информации спасателям, когда обычная связь была перегружена. В Сан-Франциско существует масса ячеек, образующих «темные» сети, доступные только членам закрытых тайных сообществ. Меш-сети, как правило, не имеют доступа к обычному Интернету со всеми его данными. И без настройки туннелирования невозможен выход в обычную меш-сеть. Туннелирование можно реализовать, только организовав туннель в меш-сеть и обратно. В настоящее время такое двустороннее туннелирование не поддерживается ни одним из ведущих производителей оборудования, но есть возможность пользоваться гибридными решениями, как, например, в проектах Open Garden и CJDNS (*https://github.com/cjdelisle/cjdns*). Это немного сложнее, чем все остальное, потому что помимо изменений в программном обеспечении требуются изменения в аппаратной части. Маршрутизаторы должны иметь возможность

[1] Сеть с ячеистой топологией (*https://ru.wikipedia.org/wiki/Ячеистая_топология*). — *Примеч. пер.*

доступа к обеим сетям для перемещения данных из Интернета в меш-сеть.

Здорово, когда есть децентрализованная сеть, но подобные сети необходимы лишь в отдельных странах, где осуществляется цензура веб-сайтов в Интернете или вообще блокируется доступ к Всемирной паутине. Я думаю, что, если подобные ситуации начнут происходить в широких масштабах, действительная потребность в меш-сетях приведет к их бурному развитию. Используя цепочки блоков, можно заставить вычислительные устройства действовать подобно шлюзам и оплачивать маршрутизацию данных криптовалютой.

Подобно тому как криптовалютой можно расплачиваться за вычисления и хранение данных, ею можно оплачивать использованную полосу пропускания. Криптовалюта позволяет занять ниши, прежде принадлежавшие централизованным властным структурам. Это могут быть рынки вычислений, хранения данных, передачи информации и любых других дефицитных ресурсов, как «настоящих», так и «искусственных», которые только могут быть выдуманы людьми. Нам предстоит увидеть, как экономика все больше и больше смещается в информационную область, а рынок данных будет занимать все более доминирующее положение по мере того, как автоматизация будет вытеснять труд человека.

Децентрализованные рынки для децентрализованных активов

Вместе с рынками появляются финансовые инструменты, такие как деривативы, активы, валюты и фьючерсы. И тут возникает проблема: где обмениваться этими активами?

Традиционно для этой цели используются централизованные фондовые биржи, но как такие биржи будут работать в децентрализованной информационной экономике? Торговые активы уже были некогда децентрализованными, еще до появления разветвленной инфраструктуры, например в бартерных экономиках. Теперь мы можем объединить свободный обмен информацией через Интернет с децентрализованными моделями проверки владения и другими. В данном случае товары можно рассматривать как активы, не находящиеся под централизованным надзором. Правительства являются доверенными посредниками, гарантирующими законность трансграничного обмена и стабильность валюты. Доверие в коммерции зависит от обеспечения активов, и правительства — самый надежный источник в этом смысле.

Активы, создаваемые правительством (такие как валюта), всегда находились под его жестким контролем. А что получится, если в общую кучу добавить негосударственных игроков? В настоящее время в сети Bitcoin проводится множество нормативных исследований на предмет того, как правительство США и правительства других стран должны регулировать ее деятельность. В некоторых странах, например в Бангладеш, валюта Bitcoin вообще запрещена. В других она признана законным платежным средством и может использоваться в торгах на существующих фондовых биржах под контролем правительства.

Однако правительства большинства стран в лучшем случае с осторожностью относятся к Bitcoin, а в худшем — с подозрением. Правительства все еще пытаются понять это изобретение, так как ему всего семь лет. Они не торопятся менять свое отношение к нему, а правовая атмосфера мешает входу Bitcoin на их рынок. Для нерегулируемого

и свободного международного рынка нам необходима децентрализованная фондовая биржа, не зависящая ни от каких правительств или корпораций, контролирующих ее работу. В информационной экономике, где децентрализованные приложения — обычное явление, каждый пользователь приложения одновременно является акционером.

Акции, которыми они владеют, обычно являются валютами децентрализованных приложений, окрашенными монетами или цепочками Bitcoin. Курсы этих валют будут колебаться синхронно с колебаниями ценности соответствующих децентрализованных приложений: чем больше людей сочтет приложение полезным, тем выше будет его ценность и, соответственно, ценность акций. Кроме того, из доходов децентрализованного приложения по его акциям могут выплачиваться дивиденды, что сделает акции еще более ценными. Рано или поздно пользователи пожелают торговать этими активами, обменивая их на более прочные и надежные активы, такие как валюты, — возможно, через цепочку блоков Bitcoin, основанную на долларе США, вроде bitUSD. BitUSD — это новая криптовалюта, привязанная к доллару США, которую можно свободно обменивать на BTC (Bitcoin). Пользователи могут также обменивать их на стабильную криптовалюту с планируемой эмиссией или на валюту с собственным децентрализованным резервом. Пользователям нужна стабильность некоторых из их активов или, проще говоря, возможность конвертирования монет во что-то, что поддерживает их покупательную способность, когда их устраивает текущий курс и они не желают подвергаться риску потерь в результате инвестирования в децентрализованное приложение.

До сих пор обмен криптовалют зависел от централизованного источника ваших денег, поэтому вы вынуждены были

доверить ему осуществление обмена. Такое положение дел привело к многочисленным кражам из центральной организации, и нередко самими центральными организациями, в основном потому, что они не ограничиваются никакими регулирующими правовыми договорами, учитывая, что Bitcoin все еще изучается большинством правительств. Самый печально известный пример — биржа Mt Gox, которая основана в Токио Марком Карпелесом. Gox имела огромный оборот на заре Bitcoin, и многие пользовались ее услугами. Но в феврале 2014-го она закрылась без всяких объяснений, что привело к потере средств ее клиентов. Люди были в ярости, но из-за неясного правового статуса Bitcoin было сложно привлечь Карпелеса к ответу.

С тех пор люди стали меньше доверять биржам. В идеале можно было бы объявить цену активов/акций нашего децентрализованного приложения на государственных фондовых биржах, но тогда потребовалось бы получить одобрение множества регулирующих органов, а люди не хотят ждать. Более того, чтобы попасть на федеральную фондовую биржу, компания должна выполнить первичное публичное размещение акций (Initial Public Offering, IPO), для чего она должна иметь определенный капитал в несколько миллионов, нанять инвестиционных банкиров и юристов и подать горы документов. Это очень высокий барьер входа, а так как фондовые биржи удерживают монополию на торговлю в рамках целых стран, они в состоянии сохранять такое положение вещей и дальше.

Bitcoin дала нам возможность — выбор — отказаться от финансовой системы, и мы должны иметь такую возможность. Не потому, что правительство — это «зло» и «бесконечно печатает деньги», а потому, что люди должны иметь свободу выбора, и чем больше вариантов, тем лучше. Да-

вайте децентрализуем IPO и позволим стартапам на самых ранних стадиях продавать свои акции людям, которые не являются *аккредитованными инвесторами*. Требование обязательной аккредитации инвесторов ограничивает круг тех, кто мог бы вкладывать деньги, корпорациями, имеющими не меньше миллиона в банке. Краудфандинговые[1] сайты — хорошая штука, но из-за этого требования они не дают акции инвесторам. Нам нужна децентрализованная фондовая биржа. Как такая биржа могла бы работать?

Существует несколько идей, основанных на использовании Ripple-подобного (*https://ripple.com*) механизма торговли. (Под словами «Ripple-подобный» подразумевается возможность выбора того, с кем достигнут консенсус, в противоположность принципу доказательства выполнения работы (proof-of-work), согласно которому доверие отдается большей части работы.) Stellar (*https://stellar.org*) — в некотором смысле обновленная версия Ripple; его основатель Джед Маккалеб вместе с единомышленниками по команде Stripe, такими как Дэвид Мазиерис (выдающийся специалист по распределенным системам из Стэнфорда), решил создать новую криптовалюту. Stellar основан на исходном коде Ripple, но использует свой, новый механизм консенсуса, который все еще дорабатывается.

Основная идея заключается в возможности обмениваться своей валютой с кем-то другим, доверяя управление обменом третьей стороне. Доверенной третьей стороной может быть банк, центр обмена или даже друг. Система интересна тем, что в отличие от многих альтернативных криптовалют она децентрализована не полностью и добавляет элемент

[1] Краудфандинг (crowdfunding) — прямое финансирование физическими лицами проекта или предприятия. — *Примеч. пер.*

межличностного или организационного — то есть социального — доверия для содействия обмену.

Это не идеальное решение, но задача по-настоящему децентрализованного согласования с нулевым доверием не решена и едва ли будет решена, пока не произойдет большой скачок в развитии искусственного интеллекта. Централизация имеет свои минусы, но она имеет и плюс: подотчетность. В модели Stellar узлы-посредники несут ответственность за любые неудачи, и, если таковые происходят, репутация таких узлов снижается. Эта модель ценна сама по себе, но проблема проекта Stellar, как и многих других криптовалют, в том, что он вводит в оборот новую криптовалюту и правильность работы его механизма консенсуса еще предстоит доказать.

Доказательство выполнения работы — единственное известное решение, способное противодействовать атакам Сибиллы, которое может применяться в широком масштабе. И что же нам остается? Самая децентрализованная биржа, какую я знаю, называется Mercury (*http://mercuryex.com*) и создана разработчиком с псевдонимом Mappum. Mercury — это мультивалютный кошелек, использующий протокол Cross-Atomic Chain (CAC) для обмена криптовалютами. Протокол CAC позволяет, допустим, Алисе и Бобу, владеющими монетами в разных криптовалютах, произвести обмен без привлечения третьей доверенной стороны. Обе криптовалюты должны иметь протокол, поддерживающий такую возможность. Фактическое согласование операции выполняется на сервере, но кошелек с монетами хранится локально, поэтому отсутствует риск кражи. Mercury — проект с открытым исходным кодом, который можно найти в репозитории GitHub. Mercury пока находится в стадии разработки, но уже имеет несколько пользователей. С ростом популярности кошелька

будет появляться все больше пользователей, желающих хранить все свои активы в кошельке и обменивать их на другие ценности одним щелчком мыши. Это предтеча идеального кошелька, в котором можно хранить любые и все криптовалюты.

Однако едва ли когда-нибудь появится кошелек, поддерживающий все конкурирующие протоколы, так же как нет веб-браузеров, поддерживающих Xanadu или любых других неудачливых конкурентов протокола HTTP. Однако браузерами пользуются все, кто желает иметь доступ к Всемирной паутине. Bitcoin, окрашенные монеты и боковые цепочки победят в конце концов. Активы в BTC являются самыми надежными, потому что для преодоления защиты цепочки блоков Bitcoin необходима вычислительная мощность, превышающая мощность 500 суперкомпьютеров, неоспоримы преимущества первопроходца, обусловившие занятую долю рынка и высокую популярность, и имеется превосходное сообщество разработчиков, занимающееся развитием этой технологии. Любые компании могут добавлять поддержку своих монет в открытый кошелек Mercury без первичного публичного размещения акций.

Модель Mercury настолько убедительна, что, даже если этот проект проиграет конкурентную борьбу, барьер входа в IPO будет снижен для тех, кто выпускает активы в криптовалюте.

Практическая децентрализация

Соответствие государственным нормам является обязательным условием успеха в современном мире. Достаточно взглянуть на PayPal и ныне на Coinbase, чтобы увидеть, как такое

соответствие повышает доверие пользователей и способствует росту бизнеса на международном уровне. К сожалению, выпуск активов вашей корпорацией, зарегистрированной в США, для людей, не являющихся аккредитованными инвесторами на децентрализованной фондовой бирже, не соответствует государственным нормам. Итак, что могут сделать разработчики децентрализованных приложений, чтобы выйти из тени и занять такое же положение, как Facebook, Instagram и Vine? Вот три предложения.

- Создать свою корпорацию как некоммерческую организацию.
- В юридических документах назвать активы жетонами приложения, чтобы устранить препятствия.
- Добавить перечень своих активов в децентрализованную биржу, такую как Mercury.

Это хорошее начало, и указанных пунктов может оказаться достаточно для эффективной децентрализации. Вы сможете обойти юридические ловушки, связанные с децентрализованными биржами, и заняться прямым обменом. Обмен осуществляется децентрализованно, поэтому ему нельзя помешать, разве что остановить сервер. Но, так как приложение открыто, пользователи смогут постоянно создавать новые резервные копии сервера или подставлять свои адреса.

Использование боковых цепочек и окрашенных монет устраняет стресс от необходимости иметь дело с разными цепочками, которые могут не поддерживаться протоколом CAC, и писать куски кода для обеспечения совместимости. Если в основе лежит только Bitcoin, программисту проще писать приложения, использующие несколько разных ва-

лют. Например, представьте децентрализованную версию Dropbox, основанную на IPFS. Пользователи такого приложения смогут оплачивать комиссионные сборы сети (Filecoin), регистрацию своих имен в цепочке Namecoin и, возможно, какие-то внутренние услуги, предоставляемые службой. Как одна сделка может быть поделена между тремя разными валютами, попав в разные цепочки? Если в основе всего лежит Bitcoin, переводы между монетами можно осуществлять беспрепятственно.

Вот почему Bitcoin выглядит прочной основой цепочки блоков для всех других финансовых активов: пока она действует лучше, чем все остальное, проще поддается осмыслению и реализации. А как насчет выбора между монетами и вариантами их использования? Скорость сделок в Bitcoin оставляет желать лучшего. Litecoin как боковая цепочка позволит существенно ускорить их. Darkcoin поможет сохранить в тайне историю ваших сделок, маскируя и шифруя информацию о сделках в цепочке блоков. Primecoin и Gridcoin позволят использовать алгоритм доказательства выполненной работы для решения научных задач, вместо того чтобы впустую расходовать вычислительные мощности на поддержание сети. Как получить нужные нам возможности разных криптовалют без необходимости держать их все и выбирать между ними?

Решение заключается в создании универсальной обертки для всех криптовалют. Представьте кошелек, способный хранить любые криптовалюты, основанные на Bitcoin (к чему стремится проект Mercury). Он полезен, но все еще заставляет пользователя беспокоиться о том, какую валюту потратить и когда. Было бы удобнее, если бы пользователь мог видеть общий баланс в одной валюте и возможности, которые можно включить или выключить. Обертка

должна преобразовывать валюты в возможности, между которыми конечный пользователь легко сможет переключаться.

Установка на стороне сервера могла бы заключаться в простом развертывании модулей, например с помощью диспетчера пакетов Node.js. Нужна высокая скорость сделок? `bpm install lite`. Тайна сделок? `bpm install dark`. Помощь в обработке научных данных? `bpm install solar`. Сделки будут передаваться в разные цепочки блоков, обеспечивающие все возможности, затребованные пользователем. Пользователи смогут хранить активы отдельно, как они делают это сейчас, в онлайн-кошельках и портфелях или в своем валютном кошельке — универсальном инструменте управления активами.

Рассмотрим практические аспекты: хранить валюту в электронном бумажнике на своем компьютере опасно. Это децентрализованная альтернатива банкам, и некоторые полагают, что стоит рискнуть и принять тем самым идеологическое решение. Некоторым не нравятся банки — их можно подкупить или, учитывая, что это централизованные институты, разорить; много яиц в одной корзине. Но большинство людей не связывают себя идеологией! Это печальная истина; большинство не волнуют проблемы централизации, безопасности данных или прав собственности на них. Они просто хотят, чтобы нечто, чем они пользуются, работало хорошо и помогало им решать их проблемы.

Банк решает очень важную для людей проблему: обеспечивает надежное хранение ценностей. Услуга хранения появилась очень давно и продолжает пользоваться спросом. Coinbase, например, является одной из самых популярных служб, опирающейся на Bitcoin, и она никак не децентра-

лизована. По сути она превратилась в банк: позволяет хранить в защищенном онлайн-кошельке не только монеты Bitcoin, но и доллары США! Популярность Coinbase растет вместе с популярностью Bitcoin. Эта служба проста в использовании и избавляет от хлопот, связанных с хранением. Если когда-нибудь Bitcoin получит широкое распространение, банки вынуждены будут принять данную валюту. Звучит слишком радикально, особенно если учесть, что Сатоси создал Bitcoin с целью исключения проблемы возвратных платежей, присущей всем онлайн-сделкам. Почему может потребоваться хранить криптовалюту в банке? Это сродни хранению долларов на вашей банковской карте, а не в кошельке.

Bitcoin дает возможность выполнять сделки через Всемирную паутину без возвратных платежей, в точности как наличными. Но многие предпочитают пользоваться услугами таких организаций, как банки. Когда банк работает хорошо, это ценная услуга. Мы можем тратить свою валюту где угодно, не беспокоясь о возможности ее кражи, потому что вклад застрахован банком и его служба безопасности гарантирует сохранность нашей информации. Традиционные банки будут вынуждены сделать следующий шаг в своем развитии и принять Bitcoin как протокол передачи денег или уступить новым конкурентам, таким как Coinbase. Представьте, что ваш банк хранит ваш счет в Bitcoin. В этом случае вы сможете видеть свой баланс в валюте своей страны, иметь соответствующий номер счета, номер отделения банка и открытый ключ Bitcoin.

Если кто-то переведет вам средства в Bitcoin, они попадут непосредственно на ваш банковский счет. Если кто-то переведет вам средства в валюте вашей страны, они попадут непосредственно на ваш банковский счет. Вы сможете полу-

чить стабильность цен как одну из особенностей универсальной обертки, и ваш баланс останется таким же стабильным, как в данный момент. Вы сможете тратить свои монеты Bitcoin везде, где принимают государственную валюту, потому что они станут неразличимы.

Bitcoin найдет свою нишу. Большинство людей не волнует слово «Bitcoin», они просто хотят пользоваться своей валютой. Лидеры в пространстве Bitcoin, такие как Abra (*https://goabra.com*), даже не упоминают слово «Bitcoin». Они просто используют этот протокол для микроплатежей и быстрого перевода денег между странами за умеренную плату.

Abra — это стартап, цель которого — захватить гигантский рынок денежных переводов в развивающихся странах путем создания серии децентрализованных пунктов, обменивающих местные валюты на Bitcoin и обратно. Вместо Bitcoin они используют термин «цифровые деньги» — умный ход, чтобы не отпугнуть рядовых потребителей и увеличить объем продаж. В действительности людей, помыслы которых далеки от Кремниевой долины, Bitcoin вообще никак не волнует.

Рост количества Bitcoin-банкоматов впечатляет, но посетите любую из развивающихся стран с неразвитой финансовой инфраструктурой, и вы увидите, что переход от сети традиционных банкоматов к сети Bitcoin-банкоматов случится очень не скоро и, может быть, даже не в нашей жизни. Но Bitcoin и не должен стремиться к этому, он должен выступать как дополнение к существующей платежной инфраструктуре, делая ее быстрее и дешевле (банковские переводы, особенно международные), и привносить все возможности, какие только может предложить криптова-

люта пользователю, такие как микроплатежи — которые в свою очередь будут способствовать распространению децентрализованных приложений, подобно рынку микроблогов.

Никого нельзя заставить пользоваться банками, идентичностью, данными или централизованными вычислениями, но все это может сделать жизнь проще. У нас есть возможность децентрализовать в случае необходимости и надежда, что, по мере того как пользователи будут все больше и больше осознавать ценность своих данных, мир постепенно начнет понимать важность безопасного хранения самих открытых ключей.

3 Создание первого децентрализованного приложения

Достаточно теории, давайте займемся практикой. Далее я буду полагать, что у вас уже есть опыт разработки хотя бы одного программного приложения. Создание децентрализованного приложения не намного труднее создания обычного приложения. Основные усложнения заключаются в необходимости думать децентрализованными категориями и в отсутствии большого количества зрелых библиотек, которыми широко пользуются разработчики обычных приложений.

Данная глава проведет вас через создание децентрализованной версии Twitter и познакомит с:

- языком программирования Go;
- децентрализованной архитектурой и распределенным хранилищем данных IPFS;
- Kerala, интерфейсом к IPFS;

- Coinprism, электронным кошельком для окрашенных монет;
- Mikro, децентрализованным приложением для обмена сообщениями, обладающим внутренней экономикой.

Go

Мы напишем наше децентрализованное приложение на языке Go. Go вызвал огромный интерес со стороны разработчиков серверных приложений благодаря простому синтаксису, свободному от функций обратного вызова, высокой скорости и дружественному механизму параллельных вычислений в лице go-сопрограмм (go-routines). Разработчики двух других языков, Erlang и Rust, заявляют, что последние превосходят Go, и, возможно, так и есть в некоторых аспектах, но в отличие от библиотек Go библиотеки Erlang и Rust находятся на очень, очень ранней стадии развития.

JavaScript — еще один довольно популярный в наши дни язык программирования; с появлением Node.js разработчики на JavaScript больше не ограничены областью создания клиентских сценариев. Они могут создавать и развивать целые веб-стеки, пользуясь единственным языком (и, конечно, HTML/CSS). JavaScript — это язык Всемирной паутины, и разработчики на JavaScript могут применять для создания своих веб-приложений большое разнообразие фреймворков для JavaScript. Хотя JavaScript — мощный язык, он имеет свои слабые стороны. Он плохо приспособлен для реализации параллельных вычислений и имеет довольно запутанную объектную модель. Go лишен этих недостатков и лучше приспособлен для создания распределенных систем.

Мне приходилось писать веб-приложения на обоих языках, Go и JavaScript. Оба они имеют свои сильные и слабые стороны, но, на мой взгляд, Go лучше подходит для разработки децентрализованных приложений. Компания Google создала Go, потому что ей требовался язык, позволяющий реализовать максимально эффективные параллельные вычисления с огромными наборами данных в масштабе, характерном для нее. Go решил эту проблему, и с момента выхода его первой версии значительно увеличилось его применение для внутренних нужд Google.

Go обладает широтой возможностей и скоростью компилирующего языка C, а также элегантностью и выразительностью Ruby. Он предназначался для разработки распределенных систем, именно поэтому я возвращаюсь к нему, когда задумываюсь о создании децентрализованных приложений. Большим плюсом является тот факт, что IPFS написана на Go, — это упрощает интеграцию распределенного хранилища файлов в приложение благодаря отсутствию барьеров совместимости. В настоящее время существует множество веб-фреймворков на основе языка Go: Martini, Goji, Gorilla и даже стандартный пакет net/http. Мне нравится, когда стек зависимостей остается максимально легковесным, поэтому в своих приложениях я в основном использую пакет net/http и прибегаю к помощи других библиотек, только когда это действительно необходимо.

Централизованная архитектура

Существуют три распространенные парадигмы создания стандартных клиент-серверных веб-приложений. Поговорим немного о них.

REST

Относительно простая модель клиент-сервер, фактически ставшая стандартным способом организации обмена данными во Всемирной паутине. REST, или Representational State Transfer (передача репрезентативного состояния), — это набор правил и приемов создания масштабируемых веб-приложений, обычно основанных на модели клиент-сервер. REST — название практики (так же как AJAX), а не отдельной технологии. Она поощряет использование возможностей, которые давно имеются в протоколе HTTP, но редко применяются. Пользователь может просто ввести в адресной строке браузера строку URL (Uniform Resource Locator — единообразный локатор (определитель местонахождения) ресурса), в результате чего браузер пошлет HTTP-запрос. Каждый HTTP-запрос несет информацию в форме параметров, которую сервер может использовать, чтобы решить, какой HTTP-ответ послать обратно клиенту.

CRUD

Аббревиатура CRUD расшифровывается как Create-Read-Update-Delete (создать-прочитать-изменить-удалить). Это перечень основных операций, которые могут выполняться с данными в хранилище. С помощью указанных операций вы напрямую работаете с записями или объектами данных; без них записи являются всего лишь пассивными сущностями — как правило, таблицами и записями в базе данных. В то время как REST определяет стиль взаимодействий с действующей системой, CRUD описывает методы управления данными в системе. Обычно для выполнения операций CRUD со своими данными разработчики используют базы данных, такие как MongoDB или MySQL.

MVC

Аббревиатура MVC расшифровывается как Model-View-Controller (модель-представление-контроллер); в настоящее время MVC является наиболее популярной парадигмой программирования. Модели отвечают за логику и данные приложения. Представления отображают пользовательский интерфейс приложения. Контроллеры принимают ввод пользователей и производят необходимые вызовы методов объектов моделей и представлений для выполнения конкретных действий.

Децентрализованная архитектура: введение в IPFS

Итак, что происходит с парадигмами CRUD и REST в децентрализованной архитектуре? Они фактически объединяются в единую парадигму, поскольку данные находятся в децентрализованной сети компьютеров и не принадлежат никому конкретно, как это имеет место в случае с IPFS. Операции с локальными данными выполняются точно так же, как с удаленными. Вы и все остальные одновременно являетесь и серверами, и клиентами. Данное объяснение звучит сложнее, чем есть на самом деле. В качестве решения для организации хранения данных я выбираю IPFS, потому что она продвинулась намного дальше, чем ее конкуренты, и за годы исследований породила множество великолепных идей, подтвержденных на практике.

Децентрализованные приложения выполняются не на сервере, а локально, на всех компьютерах пользователей. Мы все еще не решили проблему децентрализации вычислений, а выгрузка приложения в централизованную виртуальную

машину (Virtual Machine, VM), например Heroku, противоречит цели децентрализации, поэтому правильнее распространять децентрализованное приложение в виде загружаемых двоичных файлов. Пользователи смогут загрузить их на свои компьютеры и получить доступ к децентрализованному приложению либо с помощью веб-браузера, либо напрямую, используя интерфейс клиента, например Spotify или Skype.

Децентрализованным приложениям необходимо хранилище данных в той или иной форме, и как таковые они одновременно будут играть роль узлов распределенного хранилища файлов на основе IPFS. Как вариант для хранения данных можно просто использовать сторонний узел IPFS на сервере, но тогда поставщик облачных услуг станет центральной точкой отказа. Неизбежно кто-то решит купить некоторое пространство на Amazon EC2, развернет там узел и предложит узел-IPFS как услугу, чтобы облегчить жизнь начинающим разработчикам. В этом случае данные будут копироваться оттуда, по мере того как люди будут запрашивать файлы. Облачный узел IPFS мог бы также пригодиться для мобильных децентрализованных приложений, учитывая, что отдельный узел IPFS принимает на себя значительную долю вычислений, связанных с обработкой, и помогает сэкономить значительную долю заряда аккумулятора ноутбука.

Работа узлов может стимулироваться выгружающими данными для хранения путем поощрения в долларах или криптовалюте. Создатель IPFS Хуан Бенет (Juan Benet) опубликовал статью (*http://filecoin.io/filecoin.pdf*) с описанием криптовалюты FileCoin, предназначенной именно для этого, но работа над ее реализацией так и не была начата, поэтому она не может дать нам никаких выгод. Меж-

ду тем любой желающий может предложить свои схемы стимулирования хранения данных в IPFS, чтобы избавиться от необходимости держать узлы постоянно подключенными к сети для доступа к данным, хранящимся на них. Чем выше уровень децентрализации, тем лучше. Даже если сервер узла IPFS отключился, а данные, хранящиеся на нем, пользуются спросом, весьма вероятно, что копия уже хранится у кого-то еще, запрашивавшего эти данные ранее. Таково одно из основных достоинств IPFS и причина, почему создатель называет ее permanent web (постоянная сеть). Теоретически вы можете также оплатить услуги сервера за закрепление ваших данных. Кому-то эти данные не нужны прямо сейчас, но могут понадобиться позже. Пока кому-то нужны ваши данные, они будут существовать.

Было бы здорово создать мобильное приложение, но в данном учебном примере я сосредоточусь на децентрализованном приложении для настольного компьютера, потому что в Swift/ObjC или Android все еще нет надежной обертки для IPFS.

Рассмотрим две ключевые команды в IPFS.

`ADD`

Добавляет данные в IPFS.

`CAT`

Читает данные из IPFS.

Обратите внимание на отсутствие команды `delete`. IPFS — постоянная сеть! После добавления данных в сеть их нельзя удалить, если только вы не являетесь единственным их

хранителем. Это обусловлено тем, что другие узлы будут хранить копию данных, как только они ее получат. Отметьте также отсутствие команды `update`, что связано с тем, что IPFS внутренне использует технологию Git. Изменяя файл, вы не удаляете его, а сохраняете разницу между версиями. Вы можете создать граф merkleDAG для этого файла, чтобы последний хеш соответствовал последней версии файла. Все прежние версии продолжают существовать, и вы при желании сможете извлечь их.

Добавляя данные в сеть, вы фактически рассылаете широковещательное сообщение, что у вас есть некоторые данные, а не посылаете их на другие компьютеры. Передача данных происходит только тогда, когда кто-то их запрашивает. А поскольку данные находятся в сети, манипуляции данными осуществляются командами, посылаемыми в сеть.

IPNS (уровень поддержки имен, располагающийся над IPFS) создает видимость возможности изменения и удаления через изменение имен. С помощью IPNS вы можете опубликовать DAG данных под своим неизменяемым идентификатором, и тогда любой, знающий ваш идентификатор, сможет получить хеш DAG. IPNS может хранить только один вход в DAG для каждого идентификатора, поэтому, если понадобится изменить или удалить данные, достаточно будет просто опубликовать новый DAG для данного идентификатора. Более подробно реализация будет обсуждаться далее в главе.

А что насчет архитектуры MVC?

Она имеет здесь место. Значит, нет никакой новейшей методологии структурирования моего кода? Нет, модели остаются прежними, контроллеры используют IPFS для

сохранения и извлечения данных, а представления — это самый обычный код HTML/CSS/JavaScript.

А умные контракты? Какую роль играют они?

В децентрализованном приложении есть определенные элементы, требующие достижения консенсуса через умные контракты, для чего обычно требуется сервер. Отличный пример — имена пользователей, как и финансовые операции, такие как депонирование и приобретение имущества. Технически умные контракты можно сравнить с моделями — вы можете передавать им свои данные через транзакции — но в действительности они не являются моделями в архитектуре MVC. Они могут действовать наряду с имеющимися моделями, но область их применения распространяется только на конкретные сценарии. Мы еще вернемся к умным контрактам далее и узнаем, как они реализуются. Как говорится, нам нужны умные модели, тонкие контроллеры и простые представления.

Eris Industries создала фреймворк для разработки децентрализованных приложений, который называется Decerver. На веб-сайте проекта можно найти большое количество литературы, описывающей, как пользоваться фреймворком, а также все его революционные методики, помогающие упростить создание децентрализованных приложений. Там говорится, что модели — умные контракты, но проблема в том, что умные контракты действуют по принципу «плати и играй» и не должны зависеть от создания моделей. Это ненужная сложность. Однако архитектура MVC все еще может применяться для создания децентрализованных приложений, и ваши контроллеры будут взаимодействовать с цепочками блоков и DHT вместо серверов.

Что мы строим?

В своем первом приложении мы реализуем децентрализованную версию Twitter. Механизм BitSwap в IPFS подразумевает, что все ближайшие узлы будут извлекать данные из узла, хранящего их локально. Децентрализованный Twitter — полезный инструмент, но это не первая попытка его реализации. Несколько лет назад бразильский разработчик Мигель Фрейтас написал децентрализованный Twitter, получивший название Twister. Увы, Twister имел много проблем безопасности, которыми не преминули воспользоваться спамеры, и Фрейтас был вынужден реализовать грубые исправления имевшимися у него инструментами. Исправления грубые, потому что в них используются такие приемы, как полный цикл доказательства выполнения работы при создании нового пользователя для проверки его идентичности после регистрации, для предотвращения атаки Сибиллы. Это создает высокий барьер входа для новых пользователей, желающих просто опробовать новую систему, не затрачивая существенно вычислительную мощность ради доказательства своей добропорядочности. Вдобавок Twister довольно сложен в установке и настройке.

Мы можем извлечь выгоду из создания новой децентрализованной версии Twitter, потому что опробуем на практике новые технологии, такие как IPFS и Bitcoin. Мы назовем приложение Mikro. Этот первый пример хорош еще и тем, что он демонстрирует применение MVP для создания децентрализованных приложений. Данные относительно просты: вы — пользователь и публикуете микрособщения. Вы можете искать новых пользователей и просматривать их микрособщения.

Подготовка

Подготовим окружение разработки на языке Go. Я всегда стараюсь избегать любых сложностей, где только возможно. К счастью, Go поставляется в виде пакетов для установки в Linux (*http://bit.ly/go-pkg-linux*) и Mac OS X (*http://bit.ly/go-pkg-mac*). (Прошу прощения у пользователей Windows, но мы будем рассматривать разработку в Unix-подобных системах.)

В упомянутых пакетах замечательно то, что они автоматически устанавливают Go в каталог *usr/local/go* и настраивают переменные окружения, определяющие пути. Такие переменные окружения часто оказываются источником проблем в программных конфигурациях. Они открывают доступ к вашим библиотекам из командной строки. Если бы пакеты не настраивали эти переменные автоматически, вы могли бы определить их вручную в командной строке, как показано ниже:

```
export GOROOT=$HOME/go
export PATH=$PATH:$GOROOT/bin
```

В данном примере переменная `$HOME` хранит путь к каталогу, куда выполнялась установка Go (`usr/local/`).

Установив Go, проверим его работоспособность. В папке *src/* создайте новую папку с именем *tests/* и внутри нее создайте файл *helloworld.go*. Введите в терминале следующую команду, чтобы приступить к редактированию файла:

```
nano helloworld.go
```

Добавьте в файл следующий фрагмент и сохраните его:

```
package main
import "fmt"
func main()
{
    fmt.Printf("hello, world\n")
}
```

Затем запустите его с помощью Go:

```
go run hello.go
```

Если в терминале появится текст `hello, world!`, значит, Go установлен правильно. Отлично — теперь можно установить дополнительные зависимости. Прежде всего нужно установить IPFS. Go существенно упрощает установку зависимостей непосредственно из источников. Чтобы установить IPFS, введите следующую команду в терминале:

```
go get -d github.com/ipfs/go-ipfs
```

После установки импортируйте настройки `bash`:

```
source ~/.bashrc
```

Команда `go get` автоматически загрузит и установит все необходимые зависимости в папку *src*, находящуюся в корневой папке Go. Перейдя в папку *src* с помощью команды `cd`, вы найдете еще одну папку с названием *github.com*. Go «нарежет» адреса URL, откуда загружались библиотеки, и превратит каждый компонент URL в отдельную папку. Так, например, в папке *github.com* будет создана папка *jbenet*. Внутри нее будет создана папка *goipfs* и так далее. Это

удобно, потому что при установке большого количества зависимостей из одного источника Go автоматически рассортирует их по отдельным папкам. Таким образом, все зависимости, загруженные с GitHub, окажутся в папке *github* и будут рассортированы по папкам с именами, соответствующими именам пользователей GitHub, чьи проекты вы загрузили.

Перед использованием IPFS нужно инициализировать ее конфигурационные файлы, выполнив команду:

```
ipfs init
```

Ее выполнение займет несколько секунд; она добавит в конфигурацию предопределенные узлы и присвоит вашему узлу пару ключей для идентификации в сети, которые понадобятся, когда вы будете добавлять или закреплять файл.

Если после завершения команды `init` ввести команду `ipfs`, в окне терминала появится следующий текст (см. с. 104–105).

В этом тексте перечислены все команды, поддерживаемые IPFS, и его вывод означает, что установка прошла успешно.

Теперь попробуем что-нибудь добавить в IPFS:

```
ipfs add hello
```

Эта команда должна вернуть нечто, напоминающее следующую строку:

```
# QmT78zSuBmuS4z925WZfrqQ1qHaJ56DQaTfyMUF7F8ff5o
```

```
ipfs - global p2p merkle-dag filesystem
ipfs [<flags>] <command> [<arg>] ...

Basic commands:

init       Initialize ipfs local configuration
add <path> Add an object to ipfs
cat <ref>  Show ipfs object data
ls <ref>   List links from an object

Tool commands:

config   Manage configuration
update   Download and apply go-ipfs updates
version  Show ipfs version information
commands List all available commands
id       Show info about ipfs peers

Advanced Commands:

daemon Start a long-running daemon process
mount  Mount an ipfs read-only mountpoint
serve  Serve an interface to ipfs
diag   Print diagnostics

Plumbing commands:

block   Interact with raw blocks in the datastore
object  Interact with raw dag nodes

Use 'ipfs <command> --help' to learn more about each command.
```

Это хеш только что добавленных данных. Сами данные все еще хранятся на вашем компьютере, но теперь имеется адрес, указывающий на них, и любой, знающий этот адрес, сможет получить файл непосредственно с вашего компьютера, если он подключен к сети. После загрузки другой пользователь тоже будет владеть копией данных. С этого момента другие, желающие получить те же данные, смогут

```
ipfs - global p2p merkle-dag файловая_система
ipfs [<флаги>] <команда> [<аргумент>] ...

Основные команды:

init       Инициализирует локальную конфигурацию ipfs
add <путь> Добавляет объект в ipfs
cat <ссылка>  Выводит объект данных из ipfs
ls <ссылка>   Выводит список ссылок из объекта

Инструментальные команды:

config   Управление конфигурацией
update   Загружает и применяет обновления go-ipfs
version  Выводит версию ipfs
commands Выводит список доступных команд
id       Выводит информацию об узлах ipfs

Дополнительные команды:

daemon  Запускает долгоживущий процесс-демон
mount   Монтирует ipfs в режиме доступа только для чтения
serve   Служит интерфейсом к ipfs
diag    Выводит диагностическую информацию

Команды обслуживания:

block   Взаимодействует с низкоуровневыми блоками в хранилище
        данных
object  Взаимодействует с низкоуровневыми узлами графа

Используйте 'ipfs <команда> --help', чтобы получить справку
        о выбранной команде.)
```

загружать их сразу с двух компьютеров — вашего и принадлежащего другому пользователю. Чем больше узлов, хранящих данные, тем быстрее выполняется загрузка, в точности как в BitTorrent. Но в отличие от BitTorrent файловая система IPFS имеет дополнительные преимущества в виде версионирования (versioning) и встроенной системы имен.

Теперь, после добавления данных в IPFS, можно попробовать получить их:

```
ipfs cat <хеш данных>
```

Эта команда должна загрузить файл `hello` и вывести его содержимое. Она прочитает файл, хранящийся непосредственно на вашем компьютере.

Следующая зависимость — Kerala. Kerala — это небольшая обертка, которую я написал для работы с IPFS и Colored Coins с целью упрощения создания децентрализованной версии Twitter, но она достаточно универсальная, чтобы ее можно было использовать в других децентрализованных приложениях. Kerala упрощает добавление данных в IPFS в форме MerkleDAG. Установить ее можно командой:

```
go get -u https://github.com/llSourcell/kerala
```

Следующий пример демонстрирует простоту добавления данных в IPFS и извлечения их оттуда:

```
// Запустить узел
node, err := kerala.StartNode()
if err != nil
{
    panic(err)
}

// Добавить текст в IPFS (создать MerkleDAG)
var userInput = r.Form["sometext"]
Key, err := kerala.AddString(node, userInput[0])

// Получить текст из IPFS (извлечь MerkleDAG)
tweetArray, _ := kerala.GetStrings(node)
```

Первый фрагмент кода запускает узел и превращает децентрализованное приложение в клиента IPFS. Он запустит

демона, который пошлет в сеть сообщение, обозначив себя узлом сети. Второй фрагмент добавляет текст в IPFS. Вы можете добавить в IPFS любые данные: видеофайлы, изображения, структуры данных. Но в данном примере используется метод `AddString`, добавляющий в IPFS простую строку. Каждый раз, когда вы добавляете строку, обертка создает новый хеш для этой строки. Затем она связывает этот хеш ссылкой с предыдущим. Ссылка — это абстрактный термин, просто обозначающий, что в ответ на запрос хеша последней строки будет возвращена последовательность хешей всех строк, связанных ссылками.

Ссылки хранятся в структуре данных, которая в терминологии IPFS называется MerkleDAG. Это ориентированный ациклический граф (Directed Acyclic Graph, DAG), который можно использовать для установления отношений между данными. MerkleDAG — отличное средство для децентрализованной версии Twitter; при публикации каждого сообщения обертка просто свяжет его с предыдущим хешем и сохранит новый хеш в локальном текстовом файле с именем *output.html*. Только вы знаете этот хеш и имеете доступ к данным, но вы сможете поделиться им с другими пользователями сети.

Последний фрагмент фактически выполняет команду ipfs cat, передавая ей хеш, связанный с вашим идентификатором (с помощью IPNS), и сохраняет результат в массиве для дальнейшего использования и отображения в представлении.

Вы будете также использовать еще одну легковесную зависимость — `httprouter`, — упрощающую создание веб-приложений. Установить ее можно следующей командой:

```
go get -u github.com/julienschmidt/httprouter
```

Теперь, когда все зависимости установлены, можно пойти дальше и загрузить исходный код децентрализованного

приложения, которое мы будем создавать. Я взял на себя смелость написать приложение заранее — объем получившегося кода слишком велик, чтобы просить вас вводить его вручную, — поэтому решил, что будет лучше, если я буду давать пояснения после того, как вы его загрузите, соберете и запустите. Введите в терминале следующую команду:

```
go get –u github.com/llSourcell/dapp
```

Для справки ниже приводится список всех модулей, импортируемых децентрализованным приложением. Все они, кроме IPFS, Kerala и `httprouter`, являются частью стандартной библиотеки Go:

```
import
    (
        "net/http"
        "github.com/julienschmidt/httprouter"
        "github.com/ipfs/go-ipfs/"
        "path"
        "html/template"
        "fmt"
        "log"
        "github.com/llSourcell/kerala"
    )
```

В окне терминала перейдите в папку проекта *dapp* и выполните команду `'go install .'`. Затем, оставаясь в этом же каталоге, выполните команду `'go run app.go'`, чтобы запустить приложение. Откройте в браузере страницу с адресом *localhost: 8080*, и вы увидите страницу своего профиля. Она должна выглядеть примерно так, как показано на рис. 3.1.

В ней пока нет сообщений, потому что вы ни одного еще не добавили. (На рисунке изображена страница с моим профилем после добавления нескольких сообщений.) Теперь отправьте четыре-пять сообщений, воспользовавшись по-

лем ввода. После отправки каждого сообщения возвращайтесь на главную страницу и обновляйте ее, чтобы увидеть их. Главная страница приложения одновременно является страницей вашего профиля, и на ней отображаются все ваши сообщения. Приложение имеет также страницу поиска, помогающую находить других пользователей и их профили. Давайте называть наше демонстрационное приложение Mikro.

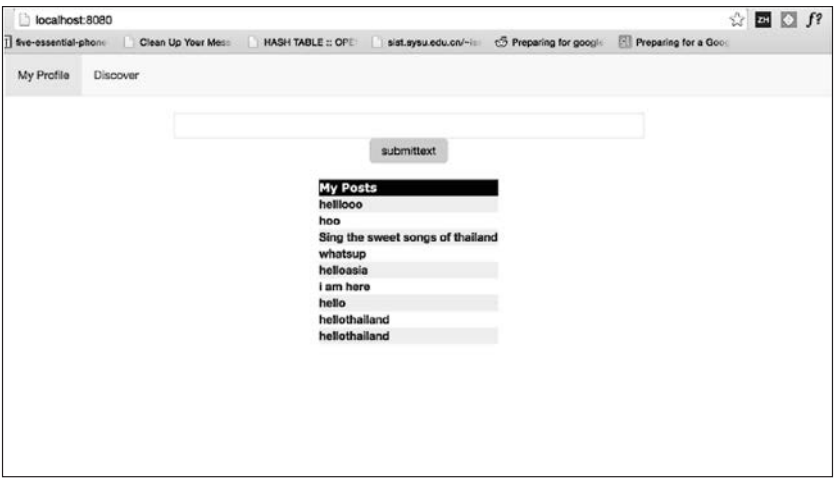


Рис. 3.1. Мой профиль

Маршрутизация

Рассмотрим сначала, как выглядят маршруты. Приложение использует универсальную и легковесную библиотеку маршрутизации (`httprouter`), опирающуюся на встроенный в Go пакет net/http. Напомню, что в стандартных веб-приложениях для загрузки страниц и отправки данных часто применяются HTTP-методы `GET` и `POST`. То же самое происходит в марш-

рутах; вместе с ними выполняются операции с данными (команды CAT и ADD файловой системы IPFS).

Маршруты определяются в методе main в файле app.go:

```
// [2] Определение маршрутов
router := httprouter.New()
// Маршрут 1 к главной странице (профилю)
router.GET("/", TextInput(node))

// Маршрут 2 к странице поиска
router.GET("/discover", displayUsers(node))

// Маршрут 3 к профилям других пользователей
router.GET("/profile/:name", TextInput(node))

// Маршрут 4 Добавление текста в IPFS
router.POST("/textsubmitted", addTexttoIPFS(node))

// [3] Связанные ресурсы
router.ServeFiles("/resources/*filepath", http.Dir("resources"))
http.Handle("/resources/", http.StripPrefix("/resources/",
    http.FileServer(http.Dir("resources"))))
http.Handle("/", router)

// [4] Запуск сервера
fmt.Println("serving at 8080")
log.Fatal(http.ListenAndServe(":8080", router))
```

Определение маршрутов начинается с инициализации структуры данных:

```
router := httprouter.New()
```

В качестве отступления отмечу, что Go не является языком объектно-ориентированного программирования (ООП) в традиционном понимании. Он следует модели, похожей на ООП, но имеющей свои отличия. Структуры в Go — это разновидность объектов. Структуры имеют поля и методы и действуют подобно объектам. В обычных языках ООП

для определения объектов используется ключевое слово `class`, что помогает организовать наследование, но в Go отсутствует механизм наследования. На первый взгляд это может показаться большим недостатком, но в действительности является достоинством языка. Наследование может превратить код в запутанный клубок, особенно когда имеется много классов и различных реализаций интерфейсов, наследуемых и расширяемых вниз по дереву иерархии. В Go для определения отношений между структурами и интерфейсами используются механизмы подтипов (отношение вида «является» (`is-a`)) и композиции объектов (отношение вида «имеет» (`has-a`)).

Первый маршрут определяет метод, который должен вызываться при переходе пользователя на страницу *localhost:8080/*. Эту страницу вы видели, когда запустили приложение в первый раз. Маршрут 2 ведет к странице поиска. На ней можно увидеть все узлы, в настоящий момент подключенные к сети и выполняющие приложение. Маршрут 3 — это модель URL. Обратите внимание на ключевое слово `:name` после `/profile/`. Оно используется для загрузки профиля произвольного пользователя; если в строке URL заменить `:name` идентификатором пользователя, браузер загрузит профиль этого пользователя. Роль идентификатора в данном случае играет идентификатор узла IPFS, созданный в момент запуска демона IPFS. Каждый узел IPFS получает собственный идентификатор, а так как ваш экземпляр Mikro является узлом IPFS, вы также получаете идентификатор. Маршрут 4 добавляет текст в IPFS, вызывая метод `POST`. Всякий раз, когда пользователь посылает сообщение, оно добавляется в IPFS посредством этого маршрута. Блоки `[3]` и `[4]` связывают сервер с ресурсами и запускают его для обслуживания порта 8080 на локальном компьютере.

Сохранение и извлечение данных

Обратите внимание на инструкцию запуска узла в самом начале метода `main`:

```
node, err := kerala.StartNode()
if err != nil {
    panic(err)
}
```

Это все, что требуется сделать приложению, чтобы стать частью сети IPFS.

После отправки нескольких первых сообщений в сеть вы могли наблюдать, как они друг за другом появляются в таблице под кнопкой с надписью «submit text» (отправить сообщение). Вы только что добавили свои первые данные в децентрализованное приложение! Помните, что добавление данных в IPFS не означает, что они были разбиты на миллион частей и разошлись по компьютерам других людей; что бы ни случилось, ни правительство, ни еще кто-то не сможет удалить их. На самом деле вы просто послали в сеть сообщение, что вы сохранили собственные данные. Они хранятся локально, в вашем компьютере. Если вы выключите компьютер, данные станут недоступными.

Эта проблема свойственна некоторым децентрализованным приложениям, таким как Twister: чтобы данные оставались доступными, компьютер должен быть все время подключен к сети. Но IPFS замечательна тем, что стремится к постоянству и делает возможным достижение состояния, когда данные доступны постоянно. Когда другие пользователи приложения Mikro прочитают ваше сообщение, они сохранят его копию у себя. И это будет происходить рекурсивно по всей сети. Чем больше людей получит ваши данные, тем более широкое распространение они получат.

Повсюду внутри приложения мы будем использовать структуру узла IPFS. Она требуется всем операциям CRUD/REST. Обеспечить ее доступность во всем приложении можно, например, объявив глобальную переменную; это, определенно, наиболее простое решение, но создание глобальных переменных считается дурным тоном, в частности потому, что может превратить отладку больших приложений в сущий кошмар. Чтобы избежать подобного, мы определим тип и будем передавать переменную во все вызовы методов маршрутов:

```
type IPFSHandler struct {
    node *core.IpfsNode
}
```

Мы завернем функцию маршрутизации в другую функцию, чтобы получить возможность передавать узел в виде переменной. Взгляните на код, реализующий добавление данных в сеть:

```
func addTexttoIPFS(node *core.IpfsNode) httprouter.Handle
{
    return func(w http.ResponseWriter, r *http.Request,
                ps httprouter.Params)
    {
        r.ParseForm()
        fmt.Println("input text is:", r.Form["sometext"])
        var userInput = r.Form["sometext"]
        Key, err := kerala.AddString(node, userInput[0])
        if err != nil {
            panic(err)
        }
    }
}
```

Здесь сначала производится парсинг формы, чтобы получить входной текст в виде строки, а затем он добавляется

в сеть IPFS с помощью метода `AddString` из библиотеки Kerala, которому в одном из параметров передается узел, а во втором — строка. В ответ мы получаем ключ в виде возвращаемого параметра и выводим его. Ключ — это хеш только что отправленных данных. Вот и все; именно так осуществляется добавление данных в сеть. Теперь давайте посмотрим, как читать данные из сети и отображать их на странице профиля.

Сразу после запуска приложение выполняет переход к домашней странице «/» и вызывает метод `TextInput(node)`. Подобно предыдущей функции, эта также обернута соответствующим методом `http`, благодаря чему мы получаем возможность передавать узел в переменной:

```
func TextInput(node *core.IpfsNode) httprouter.Handle {
    return func(w http.ResponseWriter, r *http.Request,
                ps httprouter.Params) {
```

Затем выполняется парсинг строки URL, чтобы определить наличие в ней идентификатора узла (то есть идентификатора пользователя). Этот метод вызывается для доступа не только к вашему личному профилю, но и к профилям других пользователей. По наличию идентификатора в URL решается, что делать дальше:

```
var userID = ps.ByName("name")
```

Эта инструкция сообщит, присутствует ли параметр с именем в URL. Если параметр отсутствует (это означает, что выполнен запрос на получение страницы с личным профилем), Kerala извлечет хеш merkleDAG по идентификатору собственного узла, используя стратегию IPNS разрешения имен. Если параметр присутствует, Kerala получит

DAG, ассоциированный с указанным именем. DAG — ориентированный ацикличный граф, поэтому, когда в него добавляется новый хеш, он указывает на все предыдущие хеши в обратном хронологическом порядке. Значит ли, что идентичность пользователя постоянно изменяется? Нет — это самая замечательная черта IPNS. Библиотека Kerala обеспечивает бесшовную интеграцию IPNS и IPFS друг с другом. Она ассоциирует узел HEAD конкретного графа DAG с конкретным идентификатором узла, выполняя повторную публикацию в IPNS при добавлении новых данных.

У нас имеются два возможных случая, которые обрабатываются по-разному. Первый случай — если URL не содержит параметра с идентификатором узла. Такой адрес URL соответствует домашней странице локального пользователя, а значит, приложение должно извлечь его сообщения:

```
if userID == "" {
    pointsTo, err := kerala.GetDAG(node, node.Identity.Pretty())
    tweetArray, err := kerala.GetStrings(node, "")
    if err != nil {
        panic(err)
    }
```

В данном случае сначала по вашему идентификатору определяется хеш в DAG. Затем по этому хешу извлекаются все сообщения.

Если сообщений нет, выводится уведомление об этом:

```
if tweetArray == nil {
    fmt.Println("tweetarray is nil")
    demoheader := DemoPage{"Decentralized Twitter", "SR", nil,
                  true, balance }
```

В противном случае полученные сообщения отправляются в интерфейс:

```
else {
    fmt.Println("tweetarray is not nil")
    demoheader := DemoPage{"Decentralized Twitter", "SR",
                  tweetArray, true, balance}
```

Второй случай — когда URL содержит параметр с идентификатором узла — означает, что выполняется попытка посмотреть профиль кого-то другого.

В этой ситуации приложение пытается разрешить идентификатор узла:

```
pointsTo, err := kerala.GetDAG(node, userID)
```

В случае успеха оно выполняет те же действия, что были описаны выше: извлекает сообщения из DAG и отправляет их в интерфейс. Если разрешение идентификатора завершилось неудачей, значит, пользователь не опубликовал ни одного сообщения в приложении Mikro, поэтому мы просто возвращаем `nil` и отображаем пустую страницу профиля.

Отображение данных в интерфейсе

Рассмотрим шаблон профиля в файле *index.html*.

Он начинается с загрузки Twitter Bootstrap и jQuery, двух популярных фреймворков для быстрой разработки простых веб-приложений:

```
<link rel="stylesheet"
href="http://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/
      bootstrap.min.css">
```

```
<script
  src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/
               jquery.min.js">
</script>

<script
  src="http://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/js/
       bootstrap.min.js">
</script>
```

После импортирования зависимостей создается навигационная панель, а затем добавляются два главных блока `div`: с формой для отправки нового сообщения и с таблицей опубликованных сообщений:

```
<center>
  <div id="submitform">
    <form action="/textsubmitted" method="post">
      <input type="text" name="sometext">
      <input type="submit" value="submittext">
    </form>
  </div>
</center>

<br>
<div id="posts">
  <form name="tableForm">
    <body onload="insertTable();">
    <div id="wrapper" align="center"></div>
  </form>
</div>
```

В форму отправки нового сообщения входят стандартное текстовое поле ввода и кнопка, посылающая введенный текст по адресу URL `/textsubmitted` методом `POST`. Когда пользователь щелкает на кнопке отправки, выполняется переход по указанному адресу URL, включающему пара-

метр со строкой, в результате чего происходит вызов метода `addTexttoIPFS`, обсуждавшегося выше.

Сообщения, опубликованные прежде, мы отобразим в таблице HTML, чтобы сохранить их упорядоченность. Поскольку добавление сообщений происходит динамически, таблица должна динамически изменяться в размерах с появлением каждого нового сообщения. Для реализации такого поведения воспользуемся JavaScript:

```
function insertTable(
{
    var arr = [
        {{range .Tweet}}
            {{.}},
        {{end}}
    ];

    console.log(arr.length);
    var num_cols = 1;
    var width = 100;
    var alignright = "<td style='text-align: right'>";
    var theader = "<table id='table1' width = ' "+ width +"% '>";
    var tbody = "";
    for(var j = 0; j < num_cols; j++)
    {
        theader += "<th text-align='left'><font face='verdana'>
                    My Posts" +
            " </font></th>";
    }

    var str1 = "{{ index .Tweet 1}}";
    for(var i = 0; i < arr.length; i++)
    {
        tbody += "<tr>";
        tbody += "<td>";
        tbody += "<b>" + arr[i] + "</b>";
        tbody += "</td>";
        tbody += "</tr>";
    }
```

```
    var tfooter = "</table>";
    var endalignright = "</td>";
    document.getElementById('wrapper').innerHTML = alignright +
             theader
        + tbody + tfooter + endalignright;
}
```

Конструкция из двойных фигурных скобок `{{ }}` используется для ссылки на данные (сообщения), передаваемые в интерфейс через структуру `Demoheader`:

```
var arr = [
    {{range .Tweet}}
        {{.}},
    {{end}}
];
```

Затем определяется размер этого массива вызовом стандартного JavaScript-метода `length`. Полученный размер используется для установки верхней границы цикла `for`, выполняющего итерации по элементам массива. Перед обходом массива создается заголовок таблицы HTML, который остается постоянным и не зависит от количества сообщений. Далее внутри цикла для каждого элемента массива создается новая строка, и в ней сохраняется текущее сообщение. По окончании производится объединение статических элементов с динамическими, созданными в цикле:

```
document.getElementById('wrapper').innerHTML = alignright +
         theader
    + tbody + tfooter + endalignright ;
```

В конце файла, внутри тега `style`, вы найдете определения нескольких стилей, добавленных мною. Обратите внимание на отсутствие в них какой-либо структуры. Это всего лишь демонстрационная версия, цель которой — познакомить вас

с основами создания децентрализованных приложений, а не руководство по оформлению.

Теперь перейдем к странице поиска. Главная страница приложения позволяет увидеть собственные сообщения, а URL вида *profile/:name* — страницы профилей других пользователей в сети. А как найти этих пользователей? С помощью страницы поиска, конечно! Большинство приложений социальных сетей предоставляют страницу поиска в том или ином виде, и Mikro не исключение. Если щелкнуть на кнопке Discover (Найти) в навигационной панели, откроется страница со списком всех узлов, который выглядит примерно так:

```
All peers
QmW3ssBgGLANKNKXiRxcQMmxg3FPd3tSwu2Dt96DBLbjBZ
QmRzjtZsTqL1bMdoJDwsC6ZnDX1PW1vTiav1xewHYAPJNT
QmaCpDMGvV2BGHeYERUEnRQAwe3N8SzbUtfsmvsqQLuvuJ
QmepsDPxWtLDuKvEoafkpJxGij4kMax11uTH7WnKqD25Dq
QmUy5jHXui2KzZRC3ofzHKYGmJVqAJTCsRRo2EZ6Wzwee7
```

Узлы идентифицируются по их идентификаторам, сгенерированным IPFS. Давайте посмотрим, как они извлекаются в *app.go*. Обратите внимание, что, когда осуществляется переход по маршруту `/discover`, вызывается метод `displayUsers`. Сначала этот метод извлекает из IPFS идентификаторы всех узлов:

```
// получить идентификаторы узлов
peers := node.Peerstore.Peers()
data := make([]string, len(peers))
for i := range data {
    // предполагается обратный порядок следования байтов
    data[i] = peer.IDB58Encode(peers[i])
}
fmt.Println("the peers are %s", data)
```

Идентификаторы узлов извлекаются из хранилища `Peerstore`; внутренне эта инструкция выполняет команду `ipfs swarm peers`. Далее создается массив, размер которого совпадает с количеством полученных идентификаторов, и выполняется его обход. С помощью метода `IDB58Encode` идентификаторы декодируются в удобочитаемые строки и сохраняются в массиве `data`. Затем полученный массив передается в интерфейс страницы *discover.html*. Страница *discover.html* очень похожа на страницу профиля пользователя. Она содержит таблицу HTML, динамически изменяющую свой размер и заполняемую списком узлов. Единственное отличие: в нее вместо массива сообщений передается массив узлов:

```
var arr = [
    {{range .Allpeers}}
        {{.}},
    {{end}}
];
```

Экономика децентрализованного приложения

Теперь мы добрались до самого интересного. Давайте превратим это маленькое децентрализованное приложение в его собственную микроэкономику. Вспомните обсуждение идеальной формы денег в главе 2. В настоящее время лучшим решением выпуска активов в децентрализованных приложениях являются окрашенные монеты (Colored coins). Мало кому захочется бороться с проблемами, сопровождающими развертывание цепочки блоков, только чтобы получить внутреннюю валюту приложения. В этом нет смысла, когда уже существует цепочка блоков Bitcoin с ее вычислительной мощью, превышающей суммарную мощность 500 с лишним

суперкомпьютеров и позволяющей успешно противостоять атакам Сибиллы. Несмотря на то что протокол Counterparty предлагает неплохое решение, он вводит новую валюту, что влечет за собой ненужные сложности, и имеет монолитную реализацию, лишенную модульной структуры. Применение окрашенных монет позволяет создавать активы в цепочке Bitcoin, которые никому не принадлежат и ценность которых колеблется с ценностью самого приложения.

Но как создать собственный комплект окрашенных монет? Я нашел веб-сайт Coinprism (*http://www.coinprism.com*), который является самым простым решением проблемы. Coinprism — это электронный кошелек для окрашенных монет. Вы можете создать свою учетную запись на указанном сайте и перейти на свою страницу кошелька. Создание одной окрашенной монеты стоит 0,0001 BTC. В настоящее время это необходимое зло, пока не появится какая-нибудь служба, которая возьмет оплату на себя, как сделала Onename в отношении идентичностей Namecoin. Лично я просто перевел 0,0005 BTC из моего кошелька Coinbase в кошелек с окрашенными монетами.

Затем нужно перейти в раздел Addresses and Transactions ▸ Create a New Color Address (Адреса и транзакции ▸ Создать новый окрашенный адрес). Веб-сайт предложит создать адрес, как показано на рис. 3.2.

Дайте название своей монете; обычно названия завершают словом «coin» (монета), но это необязательно. Название монеты может включать название вашего приложения, как в приложении Gems, конкуренте WhatsApp. В качестве типа адреса я выбрал Regular address (Обычный адрес), потому что не хочу тратить время и силы на организацию локального хранилища.

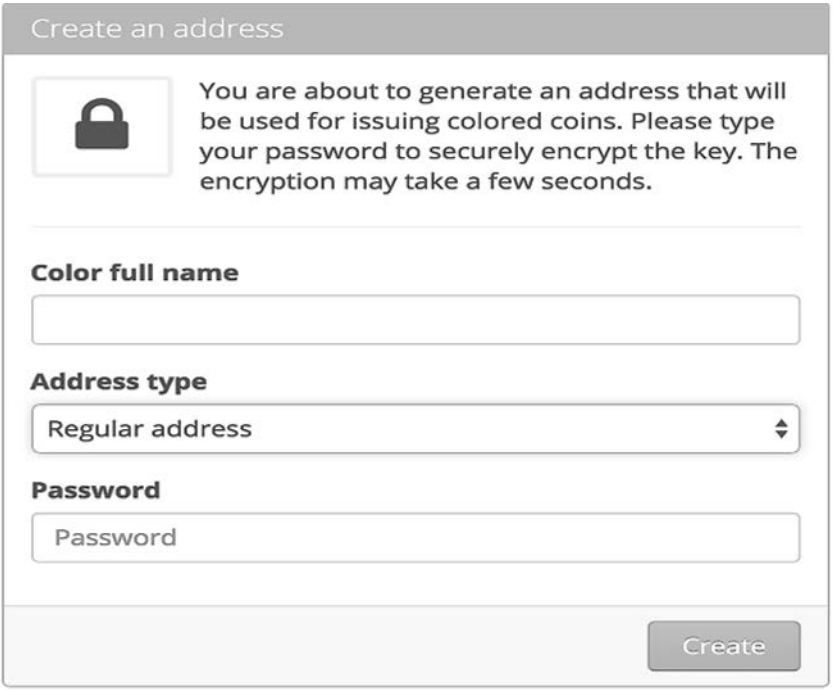


Рис. 3.2. Форма создания адреса

Затем переведите монеты с основного адреса на новый. После этого вы сможете выпустить с нового адреса окрашенные монеты для своих нужд. Ваш главный адрес будет хранить обе валюты: Bitcoin и окрашенные монеты. Заплатите 0,001 BTC, и вы сможете выпустить произвольное количество акций, соответствующих окрашенным монетам. Я выбрал 100 000 (как показано на рис. 3.3), но вы можете выбрать хоть миллион, если пожелаете. Лучше ошибиться в бóльшую сторону, чтобы у вас имелось достаточное количество монет и их хватило на всех, если вдруг ваше децентрализованное приложение начнет пользоваться большой популярностью.

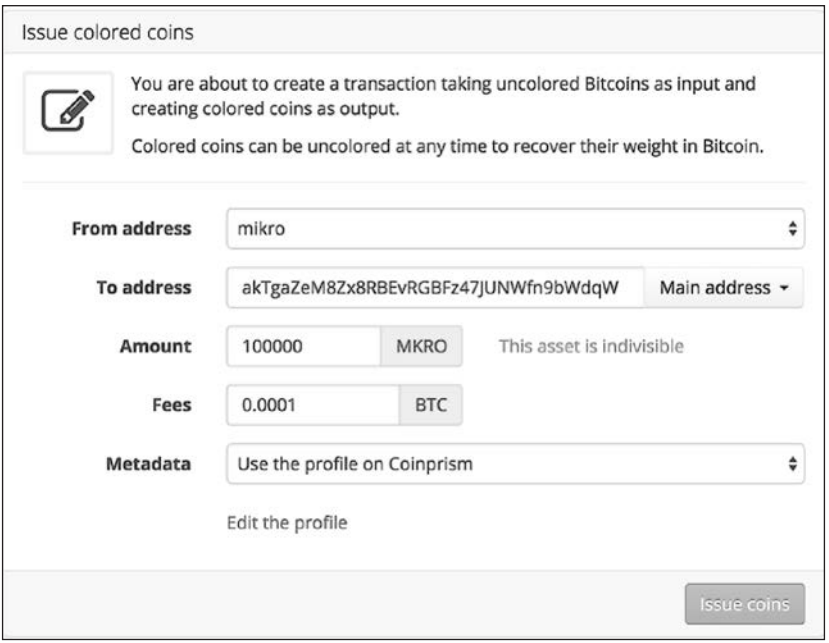


Рис. 3.3. Выпуск окрашенных монет

Когда транзакция завершится, на новом адресе появится *X* только что выпущенных монет вашего приложения, как показано на рис. 3.4.

По завершении этого шага вы должны увидеть новую валюту в своем кошельке, чуть ниже суммы, выраженной в Bitcoin. Поздравляю! Вам теперь принадлежат все вновь созданные активы. Считайте их акциями своего проекта. Вы можете передавать их кому пожелаете, и с увеличением популярности вашего проекта будет расти ценность ваших акций. Больше не требуется производить первичное публичное размещение акций (Initial Public Offering, IPO), чтобы стать публичной компанией.

Рис. 3.4. Ваш кошелек

Эти активы устраняют барьеры входа для пользователей, стремящихся получить выгоду от использования вашего приложения, что стимулирует пользователей расширять сеть и далее наращивать активы для доступа к дефицитным ресурсам в сети; в данном случае таким дефицитным ресурсом являются сообщения.

А теперь разберемся, как устроена экономика Mikro. Должны ли мы взимать с пользователей плату за размещение сообщений? Это возможно, но кому пойдут деньги? Вероятнее всего, они пойдут тому, кто предложит стороннее решение для хранения наших (шифрованных) данных. Но пока что у нас нет такой системы в IPFS, поэтому мы сделаем публикацию сообщений бесплатной, а их просмотр —

платным. Таким образом, пользователь сможет бесплатно опубликовать любое количество сообщений, но, чтобы прочитать чужие сообщения, он должен будет заплатить авторам небольшое количество монет. Иными словами, пользователи будут получать плату за вывод данных в сеть и смогут использовать заработанные деньги для просмотра чужих сообщений или потратить их на внешние расходы.

Библиотека Kerala — не просто обертка для IPFS, она также позволяет выполнять транзакции вызовом единственной функции. Как вы помните, метод `TextInput` можно вызвать для доступа к профилю любого пользователя, то есть, если он используется для загрузки страницы профиля другого пользователя, в котором имеются сообщения, выполняется следующий вызов:

```
hash := kerala.Pay("1000","1HihKUXo6UEjJzm4DZ9oQFPu2uVc9YK9Wh",
    "akSjSW57xhGp86K6JFXXroACfRCw7SPv637", "10",
    "AHthB6AQHaSS9VffkfMqTKTxVV43Dgst36",
    "L1jftH241t2rhQSTrru9Vd2QumX4VuGsPhVfSPvibc4TYU4aGdaa" )
```

Эта функция переводит на счет пользователя установленную сумму, дающую право прочитать все его сообщения, и возвращает хеш транзакции.

Метод `Pay` имеет следующие параметры:

```
Pay(fee string, from_address string, to_address string, amount
                string,
    asset_id string, private_key string) (string)
fee string
```

Обязательная сумма оплаты транзакции.

```
from_address string
```

Адрес ваших активов.

```
to_address string
```

Адрес активов пользователя, чьи сообщения вы желаете прочитать.

```
amount string
```

Сумма, которую пользователь A должен заплатить пользователю B за доступ к сообщениям последнего.

```
asset_id string
```

Идентификатор активов, созданных владельцем децентрализованного приложения.

Прямо сейчас все данные, имеющие отношение к криптовалюте, обслуживаются тестовой версией Coinprism API. Когда вы будете готовы перейти к промышленной эксплуатации приложения, вы сможете переключиться на использование промышленной версии API, и в этом вам поможет документация к Coinprism (*http://bit.ly/coinprismdocs*). (Вам просто понадобится переключить тестовый адрес URL на промышленный в настройках Kerala.)

Метод `Pay` создает транзакцию, получает в ответ целое число без знака и вычисляет его шестнадцатеричное значение. Затем он подписывает шестнадцатеричное значение и посылает его в сеть. Вызывающему коду возвращается хеш транзакции, который можно использовать, чтобы убедиться, что транзакция зафиксирована в цепочке блоков Bitcoin. Осуществление переводов в криптовалютах намного сложнее, чем хотелось бы, поэтому я посчитал полезным объединить все необходимые операции в один метод.

В библиотеке имеется еще один метод, `GenerateAddress`. Вы можете реализовать свое децентрализованное приложение так, что оно будет вызывать `GenerateAddress` при первом запуске, чтобы дать пользователю возможность определить свой адрес для активов, через который будут переводиться средства.

Нерешенные проблемы

В этом демонстрационном приложении остается несколько нерешенных проблем, но все они решаемы, и, вполне возможно, в будущем я добавлю их решение. Не откажусь и от вашей помощи.

Закрытые сети

Возможно, вы заметили, что в этом демонстрационном приложении нет «друзей». У вас есть профиль, вы можете отыскивать других пользователей и читать их сообщения после оплаты, но вы не можете «подружиться» с ними в традиционном для социальных сетей понимании. Для реализации механизма «дружбы» требуется организовать шифрование данных. Идея заключается в том, что ваш граф DAG шифруется парой открытого и закрытого ключей, и только те узлы, которым вы доверяете, могут получить доступ к вашему закрытому ключу, чтобы расшифровать и увидеть ваши данные. Если вы «рассорились» с кем-то, приложение может сгенерировать новую пару с открытым и закрытым ключами и повторно разослать закрытый ключ оставшимся друзьям. В этом случае бывший друг потеряет доступ к вашим данным. Так где же вся эта функциональность? Она называется IPFS

Keystore и все еще находится на стадии разработки. Но, когда эта книга попадет к вам в руки, она почти наверняка будет закончена, и останется лишь добавить несколько строк кода. Спецификацию можно найти по адресу: *https://github.com/ipfs/specs/tree/master/keystore*.

Удобочитаемые имена

Идентификаторы узлов на странице поиска смотрятся не очень привлекательно. Они уникальны, но плохо воспринимаются человеком. Выше в книге упоминалась система Namecoin как технология, дополняющая треугольник Зуко и позволяющая создавать децентрализованные, удобочитаемые и защищенные имена. Так как идентификатор узла уже уникален, вы можете попросить пользователя зарегистрировать удобочитаемое имя в цепочке блоков Namecoin и связать его с идентификатором узла. Всякий раз, когда вы просматриваете данные пользователя, приложение может проверить его идентичность по наличию в цепочке блоков Namecoin транзакции с идентификатором узла, соответствующим идентичности в Namecoin. Другой вариант — создать внутри приложения доверенную сеть, подобную системе репутаций. Конечно, самое простое решение — использовать службу сокращения имен (централизованное пространство имен), но в этом случае появляется центральная точка отказа. Я бы предпочел Namecoin.

Отображение только узлов Mikro, а не всей IPFS

На странице поиска отображаются все узлы в сети IPFS, а не только те, что выполняют приложение Mikro. С развитием IPNS этот недостаток легко можно будет преодо-

леть, а пока можно разместить файл, такой как *app.config*, в пространстве имен IPNS и в процессе поиска проверять наличие в файле сигнатуры того или иного узла, характеризующей приложение Mikro. Если сигнатура присутствует в файле, такой узел можно отобразить на странице поиска.

Гарантированные платежи

Наш код сначала вызывает метод `Pay()` из библиотеки Kerala, чтобы выполнить платеж, и только потом извлекает данные из IPFS. А что, если найдется злоумышленник, который удалит из исходного кода вызов `Pay()`, предшествующий извлечению данных? В общем и целом такое возможно. На этот случай у каждого пользователя мог бы выполняться обработчик события поступления платежей, посылающий отправителю платежа копию закрытого ключа после получения оплаты. Чтобы узнать адрес активов другого пользователя, приложение могло бы сохранять его в первой записи общедоступного графа DAG.

А какое место во всем этом занимают умные контракты? Система Namecoin имеет собственные умные контракты для регистрации имен. С момента создания Ethereum разработчики Bitcoin извлекли много уроков и реализовали более полный язык сценариев для поддержки широкого круга приложений, и наше — одно из них. Проект Ethereum внес значительный вклад в исследование технологии цепочек блоков, но горькая правда в том, что, когда речь заходит о децентрализованных приложениях, глобальный консенсус чаще всего оказывается ненужным и слишком дорогостоящим.

Цепочки блоков великолепно подходят для работы с финансовыми активами, но их применение для предоставления вычислительных услуг или услуг хранения весьма спорно. Примером, когда можно было бы задействовать умные контракты, служит сторонняя служба депозитария. Средства могли бы храниться в цепочке блоков до завершения сделки и после уведомления переводиться на указанный адрес. В данном приложении умные контракты не нужны, но мы увидим примеры их применения ниже.

Вот и все! Вы только что собрали из исходного кода и запустили свое первое децентрализованное приложение. Вы можете смело использовать его как основу для своего доходного открытого проекта.

4 OpenBazaar

Данная глава подробно рассказывает о децентрализованном рынке OpenBazaar (*https://openbazaar.org*). Мы обсудим причины его появления и общую структуру поддерживаемых им транзакций, рассмотрим реализацию экземпляра OpenBazaar, а также поговорим о его достоинствах и недостатках.

Зачем был создан OpenBazaar?

Bitcoin по-настоящему взволновал людей, подняв электронную коммерцию на новый уровень и обеспечив высокую скорость микроплатежей и улучшенную защищенность. Первыми учреждениями, приступившими к полномасштабному использованию Bitcoin, стали централизованные провайдеры, такие как Overstock и Dish Network. Поддержка Bitcoin позволила этим ведущим компаниям продемонстрировать свои технические возможности, но анонимность, присущая данной технологии, и мгновенная передача средств больше подходят для рынка нелегальных товаров: Silk Road (Шелковый путь).

Silk Road действовал подобно подпольной версии eBay. Это был централизованный веб-сайт, но попасть на него можно было только с использованием технологии, получившей название «луковая маршрутизация» (onion routing), через Tor. Его создатель как мог осложнил для случайных пользователей доступ к сайту, считавшемуся вершиной «темной паутины» (dark web). На сайте Silk Road люди продавали и покупали нелегальные наркотики, особенно в странах с жестким законодательством о борьбе с наркотиками. Сделки с наркотиками составляли лишь часть торговых операций на этом сайте, другая их часть была связана с межюрисдикционной продажей «легких» наркотиков, таких как марихуана (и даже табак), или ненаркотических товаров, таких как произведения и книги эротического характера, ювелирные изделия и тому подобное.

Сайт Silk Road действовал довольно долго, пока власти США не прекратили его деятельность. Это удалось по той простой причине, что сайт имел центральную точку отказа: все данные хранились на единственном сервере. Когда власти арестовали сервер, Silk Road прекратил работу, и все пользователи потеряли свои данные, связанные с этим вебсайтом. Спустя какое-то время нашелся человек, попытавшийся запустить вторую версию сайта, но он также был арестован федеральными властями, и Silk Road прекратил свое существование во второй раз.

В похожем положении находится сайт Pirate Bay; однако, учитывая относительно менее серьезный уровень его незаконности, власти боролись с ним менее последовательно. Сайт не раз закрывался различными правительственными организациями, и тем не менее он продолжает периодически появляться. Владельцы сайта просто оперативно выбирают новую площадку для размещения своего сервера после

очередного закрытия. Очевидно, что это не может продолжаться вечно, но у людей есть очень высокий спрос на вещи, которые нельзя получить иным способом в силу ограничений в законодательстве.

Помимо технической уязвимости, еще одной точкой отказа для приложений электронной коммерции является наличие ответственного руководителя, имеющего доступ ко всем данным. Новость об аресте Уильяма Росса Ульбрихта, известного также под псевдонимом Dread Pirate Roberts (грозный пират Робертс), основателя сайта Silk Road, разлетелась громкими заголовками по всему миру. Сейчас он отбывает наказание в тюрьме за незаконный оборот наркотиков и компьютерные преступления. После действий, проведенных властями с сайтом Silk Road, еще более очевидной стала потребность в децентрализованном рынке — рынке, на котором не было бы единственного человека, имеющего административный доступ ко всем данным, и который мог бы действовать локально, на любом компьютере. Именно из этой потребности родился OpenBazaar.

Что такое OpenBazaar?

В OpenBazaar нет центрального сервера, отвечающего за все. OpenBazaar представляет собой пиринговый клиент, доступ к которому не может ограничить ни одно правительство. Для работы OpenBazaar не требуется одобрение со стороны закона; здесь мы наблюдаем эволюцию неограниченного глобального рынка. Как выразились его создатели, «это eBay и BitTorrent в одном флаконе».

OpenBazaar является платформой, позволяющей продавцам и покупателям связываться друг с другом для продажи/

покупки товаров без привлечения третьей стороны, размещающей данные и взимающей комиссионные платежи. Разработчики OpenBazaar взяли за основу идею создания подлинно свободной торговой платформы, дающей людям возможность продавать и покупать товары без необходимости обращаться в центральный орган. Интернет еще никогда не был местом для чего-то, напоминающего базары прошлого: пиринговых торговых площадок, где продавцы и покупатели могли связываться друг с другом без посредников, наблюдающих за выполнением сделки. Создатели OpenBazaar надеются воплотить эту идею в Интернете.

Разработчики победили на хакатоне в Торонто со своим проектом DarkMarket. После этого они переименовали его в OpenBazaar, и с тех пор им удалось привлечь в свою команду дополнительных разработчиков. Они получают средства в основном в виде пожертвований от частных лиц. Они не имеют прибыли от проекта, и в этом состоит самый большой недостаток данного децентрализованного приложения: без стимулирования пользователей сети этот бизнес-план нельзя признать удачным. Недостаток можно смягчить введением метавалюты вместо прямого использования Bitcoin, что приведет к увеличению ценности проекта.

Как работает OpenBazaar?

OpenBazaar — это пиринговая сеть из равноправных узлов. Каждый может играть три роли: продавец, покупатель и/или арбитр. Вы можете сами выбрать роль, на основе которой будет строиться ваша репутация, и вы не ограничены выбором единственной роли. В качестве валюты в на-

стоящее время используется Bitcoin, что устраняет барьер для ввода новых валют, но не дает разработчикам возможности автоматически получать плату за свой труд. Давайте поговорим о том, как выглядит весь процесс с точки зрения каждой роли.

Продавец

Интерфейс OpenBazaar все еще продолжает дорабатываться, но уже имеет все элементы, необходимые для функционирования сайта. Продавцу требуется лишь перейти в раздел с настройками и ввести название своего магазина. Он также вводит изображение для профиля, адрес Bitcoin и идентификатор Namecoin (необязательно). После этого он заполняет свои учетные данные (рис. 4.1) и сохраняет их; данные при этом сохраняются на локальном компьютере продавца.

Продавец может общаться со своими покупателями либо напрямую в OpenBazaar, используя протокол обмена сообщениями, основанный на ZeroMQ, либо посредством

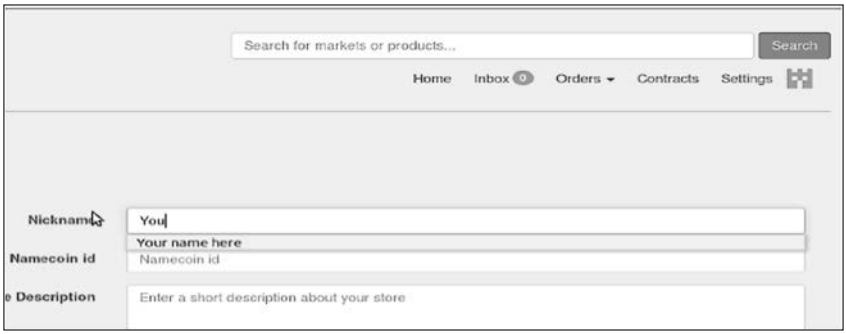


Рис. 4.1. Интерфейс OpenBazaar для ввода учетных данных

стороннего протокола, такого как электронная почта, bitmessage или своего собственного веб-сайта. Поскольку в настоящее время OpenBazaar находится в стадии альфа-версии, обновления в протоколе могут привести к удалению данных о магазине продавца. Чтобы избежать этого, разработчики предусмотрели для продавцов возможность создания резервной копии их данных, благодаря которой они легко могут восстановить свою информацию.

Наибольший интерес представляет механизм, которым пользуются продавцы для перечисления своих товаров на сайте OpenBazaar. В нем используется идея Рикардианских контрактов (Ricardian contracts, *http://bit.ly/ricardian*), упрощающих торговые сделки в сети. Контракты этого вида отличаются от умных контрактов тем, что они находятся за пределами цепочки блоков — в компьютере продавца. Рикардианский контракт по своей сути является способом определения ответственности стороны A при продаже товара стороне B. Он представляет единственную партию товара. Такие контракты используются в децентрализованных приложениях для подтверждения законности соглашений, подписанных обеими сторонами, и эти соглашения не могут быть подделаны после подписания (рис. 4.2).

В пользовательском интерфейсе контракт выглядит как простая форма, в которую вводится информация о товаре и его стоимости. Дополнительно он связывает ваш товар с вашим адресом Bitcoin и глобально-уникальным идентификатором (GUID), адресом Bitcoin покупателя и его GUID, а также с третьей стороной, выбранной на роль арбитра, которой доверяют обе стороны, заключающие сделку.

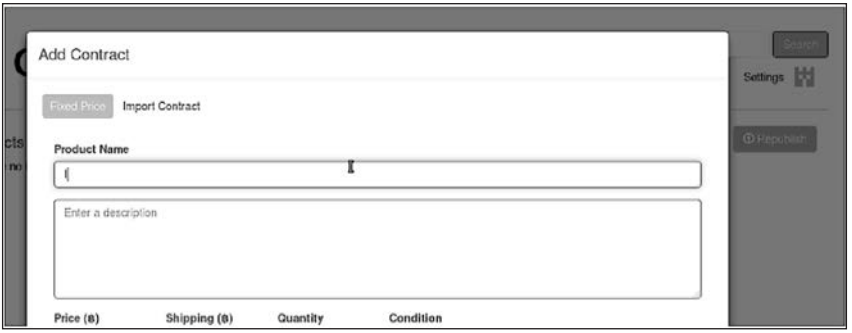


Рис. 4.2. Добавление контракта в OpenBazaar в пользовательском интерфейсе

Когда покупатель производит покупку, продавец получает уведомление, что на его товар поступил заказ. Покупатель получает информацию об арбитре, которому продавец доверил хранение средств. Покупатель может согласиться довериться указанному арбитру. В случае несогласия арбитр возвращает деньги покупателю. Если договоренность по арбитру достигнута, продавец может послать товар покупателю. Получив товар, покупатель дает указание арбитру перечислить средства продавцу. Если покупатель этого не делает, арбитр открывает процедуру диспута и после сбора информации с обеих сторон решает, какая из сторон права.

Покупатель

Покупатель вводит свои учетные данные в точности как продавец и при этом может выбирать арбитра. Арбитров выбирают покупатели, но решение о том, принять или отклонить указанного арбитра с учетом его репутации, принимает продавец. Когда писались эти строки, данное децентрализованное приложение еще находилось на раннем

этапе своего развития, а чтобы заслужить высокую репутацию, требуется время, поэтому лучшее, что можно предложить покупателям, — стараться выполнять мелкие сделки на случай, если арбитр окажется неблагонадежным. Со временем достойные арбитры займут верхние строчки; возможно, будут созданы арбитражные службы, и они станут доминирующими игроками, пользующимися наибольшим доверием.

Арбитр

Арбитром может стать любой желающий, просто установив флажок в настройках профиля. Всякий раз, когда покупатель добавляет в контракт третью сторону в качестве арбитра, выбранный арбитр получает средства в счет оплаты товара. На арбитра возлагаются обязанности улаживать споры и пересылать средства правой стороне. Арбитры могут взыскивать плату за разрешение конфликтов. Если обе стороны, продавец и покупатель, завершили сделку без обращения к арбитру, они ничего не платят за арбитраж. Если арбитр был вынужден вернуть средства покупателю или заняться улаживанием спора, он получает процент от суммы несостоявшейся сделки. Арбитры публично объявляют стоимость своих услуг на вкладке Services (Услуги) в своем профиле.

В настоящее время арбитры автоматически принимают все сделки, в которых они указаны как арбитры, но рано или поздно у них появится возможность просмотреть предстоящие сделки и принять или отклонить свое участие в них. Основные функции, доступные арбитру в пользовательском интерфейсе, показаны на рис. 4.3. На рис. 4.4 показан заполненный заказ.

Рис. 4.3. Интерфейс арбитра в OpenBazaar

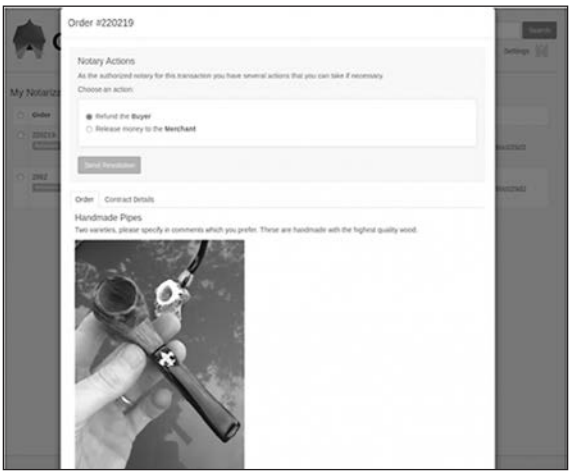


Рис. 4.4. Пример заказа на приобретение курительной трубки ручной работы

Установка OpenBazaar

Теперь, когда я рассказал, что такое OpenBazaar и как он работает, давайте загрузим его и опробуем. Поговорим о нем с технической точки зрения, рассмотрим достоинства и недостатки технологий, выбранных разработчиками для стека, и обсудим их архитектурные решения, принятые в данном децентрализованном приложении.

На момент написания этих строк не существовало двоичных файлов для установки, поэтому требовалось выполнить сборку из исходного кода.

Для этого прежде всего необходимо установить Python. Если вы пользуетесь последней версией OS X, в ней уже предустановлена версия Python 2.7. Иначе установите его вручную с помощью диспетчера пакетов Homebrew. Homebrew напоминает диспетчера пакетов apt-get в Linux. Я считаю в высшей степени полезным собирать децентрализованные приложения из исходного кода, потому что почти всегда возникает хотя бы одна ошибка, обусловленная отсутствием зависимости.

Для установки самого диспетчера Homebrew введите в окне терминала:

```
ruby -e "$(curl -fsSL https://raw.githubusercontent.com/
          Homebrew/install/master/install)"
```

Когда установка завершится, вы сможете без труда устанавливать пакеты, используя команды вида:

```
brew install <название_пакета>
```

Приложение OpenBazaar написано на языке Python. Это мощный объектно-ориентированный язык, накопивший за годы существования большое количество полезных библиотек. На нем написано много проектов распределенных приложений, таких как RPyc, поэтому его использование вполне оправданно.

Вам также потребуется Pip, диспетчер модулей для Python:

```
brew install pip
```

Теперь можно приступать к сборке OpenBazaar из исходного кода:

```
git clone https://github.com/OpenBazaar/OpenBazaar.git
cd OpenBazaar
./configure.sh
./OpenBazaar start
```

Возможные ошибки

Ниже перечислены ошибки, которые могут возникнуть в процессе сборки. Не забывайте, что проект все еще активно разрабатывается, поэтому исходный код может содержать ошибки.

Зависимости

Вы можете получить несколько ошибок из-за отсутствия необходимых зависимостей вследствие непрекращающейся активной разработки проекта. Не удивляйтесь, получив сообщение вида «X wasn't found» («X не найдено»). Просто последовательно устанавливайте недостающие зависимости вручную. То есть, получив сообщение об ошибке, определите зависимость, установите ее с помощью диспетчера Brew и вновь выполните команду `./OpenBazaar start`. Может появиться новое сообщение об отсутствии зависимости. Просто повторяйте описанный алгоритм, пока все необходимые зависимости не будут установлены.

Порты

Вам может встретиться следующая ошибка:

```
1. If you are using VPN, configure port forwarding or
disable your VPN temporarily
```

```
2. Configure your router to forward traffic from port 62112 for
both TCP and UDP
to your local port 62112
```

(Перевод:

```
1. Если вы используете VPN, настройте переадресацию портов или
   временно отключите VPN.
2. Настройте в своем маршрутизаторе переадресацию трафика
   с порта 62112 в ваш локальный порт 62112 для обоих
   протоколов, TCP и UDP.)
```

Данное сообщение означает, что один из портов, которые использует OpenBazaar, заблокирован брандмауэром или вашим маршрутизатором. Убедитесь, что эти порты доступны в вашей системе и в настройках маршрутизатора.

Сохранение и извлечение данных

Данные в OpenBazaar хранятся не в DHT, а локально, в базе данных SQLite на каждом узле. В файле *datastore.py* можно найти метод `set_item`, принимающий пару ключ/значение с некоторыми отметками времени и учетными данными. Он добавляет пару в базу данных, размещенную локально, на компьютере пользователя. Ниже приводится код из OpenBazaar, выполняющий это действие:

```
def set_item(self, key, value, last_published,
             originally_published,
             original_publisher_id, market_id=1):
    rows = self.db_connection.select_entries(
                "datastore",
                {"key": key,
                 "market_id": market_id}
            )
    if len(rows) == 0:
        self.db_connection.insert_entry(
```

```
            "datastore",
            {
                'key': key,
                'value': value,
                'lastPublished': last_published,
                'originallyPublished': originally_published,
                'originalPublisherID': original_publisher_id,
                'market_id': market_id
            }
    )
else:
    self.db_connection.update_entries(
        "datastore",
        {
            'key': key,
            'value': value,
            'lastPublished': last_published,
            'originallyPublished': originally_published,
            'originalPublisherID': original_publisher_id,
            'market_id': market_id
        },
        {
            'key': key,
            'market_id': market_id
        }
    )
```

Извлечь эти значения можно с помощью метода `_db_query`:

```
def _db_query(self, key, column_name):
    row = self.db_connection.select_entries("datastore",
          {"key": key})
    if len(row) != 0:
        value = row[0][column_name]
        try:
            value = ast.literal_eval(value)
        except Exception:
            pass
        return value
```

Ему нужно передать только ключ искомого значения, а также имя столбца, которое выглядит как `publisher_id`.

OpenBazaar использует таблицу DHT, но только не для хранения данных. Поводом к внедрению DHT в OpenBazaar послужил опыт проекта Kademlia (аналог комбинации BitTorrent и IPFS). DHT применяется как своего рода «желтые страницы» для узлов. Она представляет собой децентрализованный каталог узлов, с помощью которого узлы могут определить, как связаться друг с другом для торговли и совместного использования Рикардианских контрактов. Когда два узла соединяются посредством DHT, они могут напрямую передавать данные друг другу.

```
def __init__(self, market_id, key, call="findNode",
             callback=None):
    self.key = key
    # Ключ для поиска
    self.call = call
    # findNode или findValue, в зависимости от вида поиска
    self.callback = callback
    # Обработчик результатов поиска
    self.shortlist = []
    # Список узлов для поиска
    self.active_probes = []
    #
    self.already_contacted = []
    # В этот список добавляются узлы,
    # которым послан запрос findXXX
    self.previous_closest_node = None
    # Здесь сохраняется ближайший из найденных узлов
    self.find_value_result = {}
    # Сюда сохраняется значение, найденное
    # в результате поиска find_value
    self.slow_node_count = [0]
    #
    self.contacted_now = 0
    # Счетчик опрошенных узлов
```

```
    self.prev_shortlist_length = 0

    self.log = logging.getLogger(
        '[%s] %s' % (market_id, self.__class__.__name__)
    )

    # Создать уникальный ID (SHA1) для данного
    # запроса iterative_find с целью поддержки
    # параллельно выполняющихся операций поиска
    self.find_id = hashlib.sha1(os.urandom(128)).hexdigest()
```

В файле *node/DHT.py* метод `__init__` класса `DHTSearch` помогает отыскать с помощью DHT другие узлы, о которых требуется получить больше информации. Он предполагает, что ключ узла вам уже известен, благодаря чему поиск выполняется очень быстро, но реализация DHT в OpenBazaar опирается на рассылку широковещательных сообщений, посылаемых всем узлам, поэтому поиск простым перебором тоже возможен.

Все данные, имеющие отношение к пользователю и посылаемые в виде объекта JSON, проектировщики OpenBazaar помещают в объект с именем `data` в методе `proto_page` в файле *protocol.py*.

```
def proto_page(uri, pubkey, guid, text, signature, nickname,
               PGPPubKey, email, bitmessage, arbiter,
               notary, notary_description, notary_fee,
               arbiter_description, sin, homepage, avatar_url):
    data = {
        'type': 'page',
        'uri': uri,
        'pubkey': pubkey,
        'senderGUID': guid,
        'text': text,
        'nickname': nickname,
        'PGPPubKey': PGPPubKey,
        'email': email,
```

```
        'bitmessage': bitmessage,
        'arbiter': arbiter,
        'notary': notary,
        'notary_description': notary_description,
        'notary_fee': notary_fee,
        'arbiter_description': arbiter_description,
        'sin': sin,
        'homepage': homepage,
        'avatar_url': avatar_url,
        'v': constants.VERSION
    }
    return data
```

Эти данные, идентифицирующие пользователей, пересылаются между ними после обнаружения друг друга в DHT.

Другой замечательной чертой поддержки DHT является возможность поиска по ключевому слову. Так как ключевые слова определяются самими пользователями в настройках своих профилей, они могут описывать вид или категорию товаров, что делает поиск более простым и удобным.

```
def find_listings_by_keyword(self, keyword,
                     listing_filter=None, callback=None):
    hashvalue = hashlib.new('ripemd160')
    keyword_key = 'keyword-%s' % keyword
    hashvalue.update(keyword_key.encode('utf-8'))
    listing_index_key = hashvalue.hexdigest()

    self.log.info('Finding contracts for keyword: %s', keyword)

    self.iterative_find_value(listing_index_key, callback)
```

OpenBazaar, как и IPFS, использует DHT, но, в отличие от IPFS, не поддерживает копирование ассоциативных данных; неважно, насколько они востребованы, данные всегда хранятся только локально, на компьютере, где они были

созданы. Поскольку система не полагается на распределенные копии данных, в ней отсутствует встроенная поддержка версионирования (versioning) данных.

Идентификация

Узлы в OpenBazaar имеют собственные уникальные идентификаторы GUID. Они чем-то напоминают идентификаторы узлов в IPFS.

В файле *node/transport.py*:

```
def _generate_new_keypair(self):
    seed = str(random.randrange(2 ** 256))

    # Преобразовать ключи m/0/0/0 в BIP32
    wallet = bitcoin.bip32_ckd(bitcoin.bip32_master_key(seed), 0)
    wallet_chain = bitcoin.bip32_ckd(wallet, 0)
    bip32_identity_priv = bitcoin.bip32_ckd(wallet_chain, 0)
    identity_priv = bitcoin.bip32_extract_key(bip32_identity_priv)
    bip32_identity_pub = bitcoin.bip32_privtopub(bip32_
                         identity_priv)
    identity_pub = bitcoin.encode_pubkey(
        bitcoin.bip32_extract_key(bip32_identity_pub), 'hex'
    )

    self.pubkey = identity_pub
    self.secret = identity_priv

    # Сгенерировать SIN
    sha_hash = hashlib.sha256()
    sha_hash.update(self.pubkey)
    ripe_hash = hashlib.new('ripemd160')
    ripe_hash.update(sha_hash.digest())

    self.guid = ripe_hash.hexdigest()
```

Эти идентификаторы формируются с помощью протокола Bitcoin BIP32 (иерархических детерминированных кошельков) путем генерирования нового SIN и использования алгоритма SHA-256 для создания GUID. Идентификатор GUID так же уникален, как уникален каждый адрес Bitcoin, поэтому нет необходимости волноваться о появлении дубликатов.

Итак, мы можем уникально идентифицировать людей, прямо как в IPFS, используя технологию эллиптических кривых, реализованную в Bitcoin, но как сделать эти идентификаторы удобочитаемыми? В учетных данных в OpenBazaar помимо поля псевдонима имеется поле, куда можно ввести свой идентификатор Namecoin. Таким образом, люди могут иметь одинаковые псевдонимы и различаться по GUID. Это не самое лучшее решение: наверное, вы можете запомнить пять последних цифр в идентификаторе GUID кого-то и его псевдоним. Но Namecoin компенсирует данный недостаток, позволяя пользователям применять свои идентификаторы Namecoin.

```
def is_valid_Namecoin(Namecoin, guid):
    if not Namecoin or not guid:
        return False

    server = DNSChainServer.Server(constants.DNSCHAIN_
             SERVER_IP, "")
    _log.info("Looking up Namecoin id: %s", Namecoin)
    try:
        data = server.lookup("id/" + Namecoin)
    except (DNSChainServer.DataNotFound, DNSChainServer.
            MalformedJSON):
        _log.info('Claimed remote Namecoin id not found: %s',
                  Namecoin)
        return False

    return data.get('OpenBazaar') == guid
```

Данный код использует механизм DNSChain, чтобы убедиться в достоверности идентификатора Namecoin всегда, когда этот GUID указан в адресе Namecoin человека, идентичность которого требуется подтвердить. DNSChain — это гибрид сервера DNS, упрощающий доступ к данным Namecoin через программный интерфейс.

Таким образом, проблема идентификации в OpenBazaar решается посредством комбинации уникальных идентификаторов GUID и Namecoin, по аналогии с IPFS.

Репутация

А как решается проблема репутации? Репутация играет важную роль на любой торговой площадке; покупатели ищут надежных продавцов, которым они могут доверять, и наоборот. В централизованной модели репутацией отдельных лиц могут управлять владельцы сервера, и при соответствующем уровне безопасности у людей нет возможности обмануть систему и сфальсифицировать свою репутацию. В децентрализованной системе проверить репутацию намного сложнее.

Доверие в OpenBazaar представляет собой комбинацию синергетических систем двух типов: *глобальное доверие* и *проецируемое доверие*. Когда все члены сети одинаково доверяют определенному пользователю, такое доверие называется глобальным. Этот вид доверия устанавливается такими методами, как *proof-of-burn* (доказательство уничтожения) и *proof-of-timelock* (доказательство залога). Проецируемое доверие — доверие к узлу, различающееся для разных пользователей сети. То есть доверие к узлу проецируется каждым пользователем. Этот вид доверия устанавливается с использованием сети доверия (web of trust).

Рассмотрим каждый из методов подробнее.

Метод 1: proof-of-burn

Когда продавец создает магазин, он должен потратить определенную сумму в Bitcoin, которая будет потеряна и никогда не вернется. Это делает создание нескольких идентичностей неприемлемо дорогим и является основным механизмом противодействия атакам Сибиллы в OpenBazaar. Хотя такое решение не идеально, оно все же оказывает сдерживающее влияние. Чем больше сумма доказательства, тем дороже стоит создание учетной записи и выше барьер входа для потенциальных игроков, желающих воспользоваться услугой. Публичное и легко проверяемое уничтожение нескольких монет — это *ремередж* (remurrage) на остаток. Ремередж является понятием, обратным демереджу (demurrage — стоимость хранения валюты сверх установленного периода). Допустим, вы, сидя дома перед ноутбуком, создали валюту с 10 миллионами монет, которыми люди тут же начали торговать. Вы вышли прогуляться и, когда вернулись, обнаружили, что осталось всего 5 миллионов неуничтоженных монет. Если вы имеете какую-то сумму в этой валюте, вы получите своеобразные дивиденды сверх общего прироста экономики, ожидаемого от валюты с установленным количеством монет, такой как Bitcoin.

Децентрализованное приложение сначала генерирует адрес для уничтожения непосредственно из GUID узла.

```
def burnaddr_from_guid(guid_hex):
    _log.debug("burnaddr_from_guid: %s", guid_hex)

    prefix = '6f' if TESTNET else '00'
    guid_full_hex = prefix + guid_hex
    _log.debug("GUID address on bitcoin net: %s", guid_full_hex)

    # Изменить GUID, чтобы гарантировать надежность за счет
    # использования защиты от близких хешей в SHA256, обратив
```

```
# последний бит адреса.
guid_full = guid_full_hex.decode('hex')
guid_prt = guid_full[:-1] + chr(ord(guid_full[-1]) ^ 1)
addr_prt = obelisk.bitcoin.EncodeBase58Check(guid_prt)
_log.debug("Perturbated bitcoin proof-of-burn address: %s",
            addr_prt)

return addr_prt
```

Это самая простая транзакция с GUID. Все узлы смогут проверить, что указанный GUID уничтожил монеты (proof-of-burn — доказательство уничтожения), выполнив ту же функцию `burnadd_from_guid` с тем же параметром `guid_hex` и проверив уничтоженное количество в цепочке блоков.

Метод 2: proof-of-timelock

Доказательство уничтожения (proof-of-burn) позволяет сети превратить создание дополнительных идентичностей в дорогое удовольствие. Доказательство залога (proof-of-timelock) за счет откладывания некоторого количества монет на время (и связывания идентичности пользователя с этими неизрасходованными монетами как депозитом) гарантирует невозможность одновременного существования большого количества идентичностей, связанных с одной сущностью реального мира. Доказательство залога (proof-of-timelock) не является такой же сильной страховкой, как доказательство уничтожения (proof-of-burn): фактически доказательство уничтожения является постоянной версией доказательства залога.

В доказательстве залога узел, желающий установить доверие к псевдониму идентичности, должен доказуемо заблокировать определенную сумму внутри транзакции, которая через какое-то время будет возвращена. Особенностью такой

транзакции является невозможность завершить ее в течение определенного промежутка времени. Сеть знает, что транзакция в конечном счете завершится, а также время, в течение которого она останется заблокированной. Вся эта информация публично доступна и поддается проверке.

Один из основных недостатков доказательства залога — психологический: оно не дает такого же большого чувства ответственности, как доказательство уничтожением. Уничтожение денег имеет больший психологический вес, и для кого-то этот барьер может оказаться непреодолимым. Поэтому люди чаще используют доказательство залога.

Цепочка блоков Bitcoin сейчас не имеет прямой поддержки механизма доказательства залога. Несмотря на то что протокол Bitcoin поддерживает значение `nLockTime`, сам механизм не соблюдается действующими узлами. Это означает, что транзакция не будет доступна публично для проверки.

Доказательство залога идеально подошло бы для цепочки блоков Ethereum, где поддерживаются Тьюринг-полные умные контракты. В OpenBazaar было решено не использовать их, поскольку не было доказано, что это возможно на практике, а также из-за проблем с масштабируемостью и производительностью. Такое решение было мудрым, а рано или поздно появится боковая цепочка, которая смягчит данный риск.

Метод 3: trust-as-risk (наиболее реальный)

Разработчики все еще работают над моделью «сети доверия» (web-of-trust) и ее фактической программной реализацией, но, похоже, они идут в направлении использования модели «доверие как риск» (trust-as-risk). Они вынашива-

ют идею позволить людям расширять кредитные линии, когда кредитор доверяет стороне, которую он кредитует. Так, если вы действительно доверяете кому-то, вы можете добавить ему 0,1 Bitcoin в кредитную линию посредством транзакции, подписанной несколькими лицами; если вы прекращаете доверять ему, то можете отозвать свой кредит.

Индикаторы доверия должны храниться постоянно, децентрализованным способом. Так как разработчики OpenBazaar не желают добавлять их, чтобы не раздувать цепочку блоков Bitcoin (что можно только приветствовать), они склоняются к использованию Namecoin как неплохой альтернативы. Я считаю, что это вполне разумный подход.

Все может закончиться даже отказом от реализации «сети доверия» (web-of-trust), потому что это не так уж важно. В действительности, когда кто-то пытается обмануть нас и мы рискуем потерять деньги, мы просто звоним в банк и отменяем сделку с возвратом средств на наш счет. В сети OpenBazaar арбитр осуществляет посреднические услуги в сделках и теоретически способен предотвратить мошенничество. Узлы могли бы полагаться исключительно на арбитров. Было бы интересно посмотреть, какое влияние в сети имеет доверие; мне кажется, «сеть доверия» (web-of-trust) могла бы служить неплохим дополнением для большей безопасности сети.

Что в OpenBazaar можно улучшить?

Начнем с наиболее серьезного недостатка — отсутствия внутренней валюты. Bitcoin обеспечивает мгновенную ликвидность, и это хорошо для продавцов, но наличие внут-

ренней валюты было бы выгодно для первых пользователей и для самих разработчиков. Первые пользователи OpenBazaar рискуют своими монетами Bitcoin из-за отсутствия арбитров с хорошей репутацией, так как для создания репутации требуется время. Было бы лучше, если бы проект OpenBazaar выпустил собственную валюту для покупок внутри приложения. OpenBazaar мог бы устроить ее массовую продажу, установив начальную цену и ограничив количество жетонов (токенов).

Роль такой валюты могли бы играть окрашенные монеты; разработчики могли бы настроить контрактный адрес Bitcoin (для перевода монет), вычисляющий, сколько монет OBcoins вы получите взамен перевода в адрес OpenBazaar. С ростом стоимости OpenBazaar стоимость монет также росла бы. Первые пользователи OpenBazaar были бы вознаграждены за свои усилия по развертыванию сети, несмотря на риски; объем ликвидности для покупателей и продавцов увеличился бы; и, что самое важное, разработчики открытого программного обеспечения получали бы вознаграждение за свою работу. Финансирование — одно из основных конкурентных преимуществ централизованного закрытого программного обеспечения перед открытым. Просто в первом случае ведущие разработчики получают оплату за сопровождение и совершенствование приложения, а имея внутреннюю валюту, эту же модель можно было бы внедрить в разработку открытого программного обеспечения.

Сомнения также вызывает выбор модели хранения в OpenBazaar: данные хранятся в локальном хранилище SQLite, где отсутствуют встроенные механизмы создания резервных копий и восстановления данных из них. Если бы для хранения информации использовалась IPFS, это могло бы

повысить надежность. Чем больше людей посещает магазин, тем больше копий данного магазина создавалось бы в сети. Для успокоения владельцы магазинов могли бы уведомляться о количестве людей, скопировавших их шифрованные данные.

Знание конструктивных особенностей OpenBazaar (и сопутствующих ограничений), а также его достоинств и недостатков может помочь в разработке приложений разных типов. Некоторые из этих тем мы снова увидим в следующей главе при нашем исследовании проекта Lighthouse.

5 Lighthouse

Майк Хирн более пяти лет входил в число основных разработчиков Bitcoin и снискал большое уважение в сообществе за работу над BitcoinJ (Bitcoin SDK для Java) и лекции о Bitcoin, которые он читал по всему миру. Его последний проект называется Lighthouse (*https://www.vinumeris.com/lighthouse*); его цель — децентрализация краудфандинга. Хирн считает, что краудфандинговые сайты, такие как Kickstarter и Indiegogo, берут слишком высокие комиссионные из сумм, направляемых на финансирование проектов, за свою работу по обслуживанию серверов, рекламу, хостинг и модерацию. Прибавьте дополнительные сборы за обработку платежей платежными системами, такими как Stripe или Amazon Payments, и получится, что до 10% средств не доходит до проектов. Lighthouse — это попытка устранить посредников, чтобы проекты, привлекающие средства, получили все деньги, которые выделяются их сторонниками.

К тому же на Kickstarter существуют географические ограничения. Создать проект на этом сайте в настоящее время могут только люди, проживающие в Северной Америке, Новой Зеландии и Европе. Значит, подавляющее большин-

ство людей на планете не может создать свой проект на Kickstarter. С другой стороны, в странах, где краудфандинг запрещен, доступ к таким сайтам, как Kickstarter, может блокироваться по IP-адресам.

Более того, некоторые проекты нельзя создать на Kickstarter из-за правил, которые сайт проводит на правах центрального органа, во имя стандартов сообщества. Это относится не только к централизованным сайтам: децентрализованная модерация также возможна путем внедрения в цепочку блоков черных списков большинством голосов, которые должны принять все узлы.

Еще одним мотивом было желание дать возможность краудфандинга без размещения средств на депозитах, потому что депозиты в децентрализованном приложении — очень рискованное предложение. Может возникнуть множество проблем безопасности; теоретически протокол Bitcoin допускает возможность отзыва средств, и приложение Lighthouse стало одним из первых, реализовавших малоизвестную особенность протокола, предназначенную именно для этой цели.

Lighthouse является также отличным примером применения умных контрактов. Это приложение, вероятно, не самое сногсшибательное, но предпринятый в нем подход действительно имеет большую ценность, потому что дает возможность привлечь больше средств. Еще одной его отличительной чертой является высокая скорость перевода платежей в сравнении с другими платежными процессорами. Барьер входа для любого, привлекающего средства, стремится к нулю, а переводы выполняются со скоростью протокола без необходимости получать одобрение банков.

Функциональные возможности

Самый простой способ опробовать Lighthouse — перейти на веб-сайт (*https://www.vinumeris.com/lighthouse*) и загрузить двоичные файлы приложения для своей операционной системы. Затем дважды щелкнуть на ярлыке lighthouse, чтобы открыть вводную страницу, изображенную на рис. 5.1.

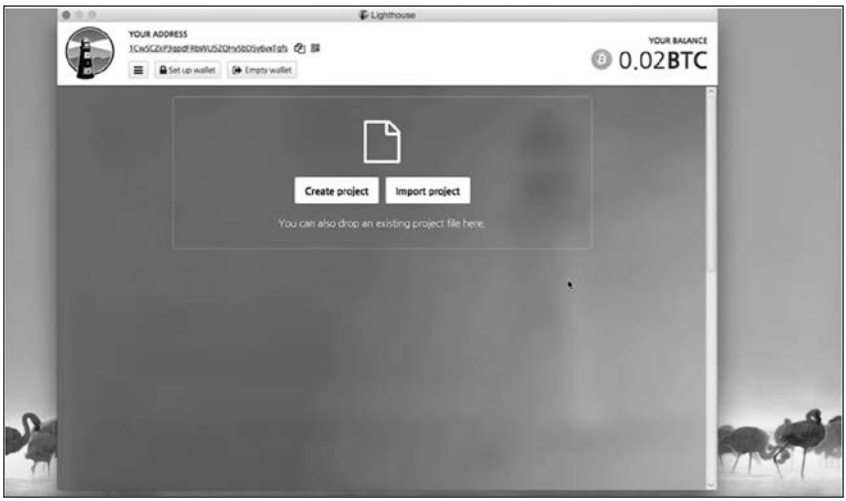


Рис. 5.1. Главная страница Lighthouse

На вводной странице можно выбрать одно из двух действий: создание проекта или импортирование существующего. В приложении нет страницы поиска (причина будет описана ниже). Вы можете перетащить существующий проект на страницу, и он будет отображен в полном формате, а вы сможете перевести ему средства (рис. 5.2).

Рис. 5.2. Пример проекта, собирающего средства

Или можно пойти еще дальше и предложить свои деньги в залог нового проекта, как показано на рис 5.3.

Вы можете указать свой счет Bitcoin в Lighthouse и финансировать из него проекты. Вы можете вернуть свой вклад в любой момент без всякого риска потерять деньги, если проект не достиг своей цели и создатель не собрал необходимую сумму. Вы можете также запустить свой проект и разрекламировать его в социальных сетях. Другие люди смогут загрузить файл вашего проекта и финансировать его.

Первая мысль, которая приходит в голову: насколько неэффективно выглядит решение, требующее импортировать и экспортировать файлы проектов, по сравнению с обычной страницей поиска, такой как в приложении Mikro из главы 3. Создатель Lighthouse выбрал такое решение просто потому, что создание страницы поиска слишком сложно.

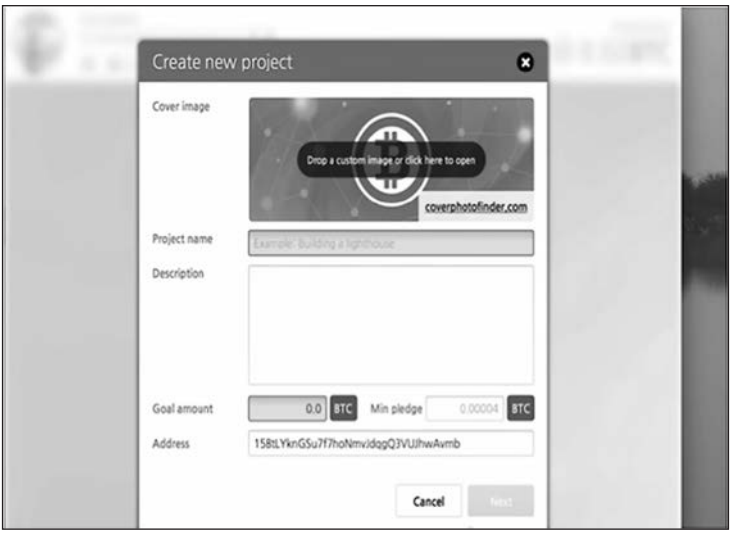


Рис. 5.3. Создание нового проекта

Оно увеличивает сложность и вероятность ошибок, а у него не было ни средств, ни времени для этого. Разработка децентрализованных систем сложна, и высока вероятность появления ошибок, невозможных в централизованных системах. Ошибки, связанные с синхронизацией пользовательского интерфейса и управления состоянием приложения, тяжело поддаются отладке; например, «я щелкнул на кнопке и увидел, что все мои пожертвования повторяются дважды, затем я перезапустил приложение, и ошибка исчезла». Ошибки синхронизации пользовательского интерфейса происходят в Lighthouse даже притом, что данные не используются совместно в сети P2P.

Здесь здорово помогла бы IPFS, если бы Хирн знал о ней. Ее можно было бы использовать для обмена файлами между узлами. Это решение, не такое дорогостоящее, как

Bitcoin, работало бы так же быстро, как BitTorrent, имело бы механизм версионирования, как Git, и обладало бы надежностью системы адресации по содержимому, где данные копируются на узлы, запрашивающие их. Даже простое применение любых механизмов DHT на основе Kademlia без IPFS было бы существенным усовершенствованием, но тут снова возникает проблема, характерная для разработки децентрализованного программного обеспечения: отсутствие средств. Внутренняя валюта могла бы помочь решить ее, но об этом мы поговорим в конце главы.

Впрочем, для передачи файлов пользователи могут использовать серверы. Они могут выкладывать файлы в федеративной сети серверов, где действуют узлы Lighthouse, и хранить файлы на них. Они могут использовать собственные хранилища, такие как Dropbox или Google Drive, и передавать друг другу ссылки на файлы. Совсем недавно появилась служба Lightlist — площадка для размещения всех проектов Lighthouse. В этой области почти наверняка возникнет серьезная конкуренция, что хорошо, поскольку не будет единственного сервера, управляющего всеми файлами; вместо него появятся несколько вариантов, а значит, степень децентрализации будет выше.

Интересной особенностью Lighthouse является использование возможности Bitcoin, доступной еще с версии Bitcoin 0.1, но полностью упускаемой из виду: `SIGHASH_ANYONECANPAY`. Lighthouse оказался в числе первых проектов, реализовавших применение этой возможности, позволяющей пользователям добавлять аннотации к своим подписям, которые сообщают другим, что они могут внести часть суммы в рамках сбора средств для данного проекта. `SIGHASH_ANYONECANPAY` дает возможность объединить несколько транзакций в одну большую транзакцию.

После того как вы подпишете свою Bitcoin-транзакцию закрытым ключом, ее невозможно будет изменить. Именно поэтому данную транзакцию можно без опаски передать всем находящимся в сети — никто не сможет изменить ее. Сбор средств в Lighthouse осуществляется как незавершенная Bitcoin-транзакция, которая принимает деньги из вашего кошелька и переводит их в кошелек сборщика средств. Поскольку транзакция, создающая деньги из воздуха, нарушает правила Bitcoin (только майнеры могут создавать монеты из воздуха), она не будет завершена, пока не будет закончен сбор всей необходимой суммы. После того как транзакция с флагом `SIGHASH_ANYONECANPAY` соберет достаточный объем пожертвований, она объединит все отдельные пожертвования, и вы получите действительный платеж, который затем добавится в цепочку блоков.

Давайте посмотрим, как это реализуется программно. Загрузите из репозитория GitHub (*https://github.com/vinumeris/lighthouse*) исходные коды Lighthouse, которые будут служить вам справочным руководством; мы не будем копировать их и собирать из них приложение. Мы знаем, что приложение Lighthouse делает, поэтому мы углубимся и посмотрим, как оно это делает.

Начнем с файла *PledgingWallet.java*:

```
public PendingPledge createPledge(Project project, Coin value,
                                  @Nullable KeyParameter aesKey,
                                  LHProtos.PledgeDetails details)
throws InsufficientMoneyException {
```

Эта функция предназначена для отправки пожертвований: она использует флаг `SIGHASH_ANYONECANPAY`, а также первый умный контракт, с которым мы познакомимся:

```
TransactionOutput stub = findAvailableStub(value);
```

Метод `createPledge` принимает рассматриваемый проект — его учетные данные, — а также сумму пожертвований и вызовом `findAvailableStub(value)` пытается найти единственный выход транзакции, удовлетворяющий сумме пожертвований, переданной в параметре.

Передача нескольких входов была бы не лучшим решением, потому что это увеличило бы сборы, выплачиваемые соискателем пожертвований. Выход транзакции с пожертвованием называется `stub`, а шаблон `tx`, расходующий их с использованием `SIGHASH_ANYONECANPAY`, представляет собой пожертвования. Шаблон `tx` возвращает контракт.

```
Coin totalFees = Coin.ZERO;
Transaction dependency = null;
if (stub == null) {
    final Address stubAddr = currentReceiveKey().toAddress
                             (getParams());
    SendRequest req;
    if (value.equals(getBalance(BalanceType.AVAILABLE_SPENDABLE)))
        req = SendRequest.emptyWallet(stubAddr);
    else
        req = SendRequest.to(stubAddr, value);
    if (params == UnitTestParams.get())
        req.shuffleOutputs = false;
    req.aesKey = aesKey;
    completeTx(req);
    dependency = req.tx;
    totalFees = req.fee;
    log.info("Created dependency tx {}", dependency.getHash());

    // Изменения оказываются в случайной позиции в выходе, поэтому
    // их требуется найти. Может так получиться, что будет найдено
    // два выхода одинакового размера, в этом случае неважно,
    // какой из них использовать.
    stub = findOutputOfValue(value, dependency.getOutputs());
    if (stub == null) {
        // Мы создали зависимость tx, чтобы получить выход,
        // а теперь не можем найти его. Это возможно, только если
```

```
        // был послан полный баланс, поэтому мы должны вычесть
        // выплату майнеру из суммы.
        checkState(req.emptyWallet);
        checkState(dependency.getOutputs().size() == 1);
        stub = dependency.getOutput(0);
    }
}
```

Если искомый выход не найден, приложение пробует создать выход требуемого размера и повторяет попытку.

Затем путем добавления флага `SIGHASH_ANYONECANPAY` к транзакции создается контракт.

```
Transaction pledge = new Transaction(getParams());
// TODO: Поддержку отправки нескольких входов в единственное
// пожертвование tx.
TransactionInput input = pledge.addInput(stub);
project.getOutputs().forEach(pledge::addOutput);
ECKey key = input.getOutpoint().getConnectedKey(this);
checkNotNull(key);
Script script = stub.getScriptPubKey();
if (aesKey != null)
    key = key.maybeDecrypt(aesKey);
TransactionSignature signature = pledge.calculateSignature
                                 (0, key, script,
    Transaction.SigHash.ALL, true /* пожертвовать может
                                     любой! */);
if (script.isSentToAddress()) {
    input.setScriptSig(ScriptBuilder.createInputScript
                       (signature, key));
}
else if (script.isSentToRawPubKey()) {
    // Эта ветвь никогда не будет выполняться в текущей
    // архитектуре, потому что деньги можно получить только через
    // адрес, но в будущем может появиться поддержка прямого
    // перевода средств по ключу с использованием протокола
    // платежей.
    input.setScriptSig(ScriptBuilder.createInputScript(signature));
}
```

```
input.setScriptSig(ScriptBuilder.createInputScript(signature, key));
pledge.setPurpose(Transaction.Purpose.ASSURANCE_CONTRACT_PLEDGE);
log.info("Paid {} satoshis in fees to create pledge tx {}",
          totalFees, pledge);
```

Вот как создатель Lighthouse реализовал свой умный контракт с применением низкоуровневого протокола Bitcoin. Очень неудобно, не находите? Низкоуровневый протокол Bitcoin не отличается дружественностью к разработчикам. Если вам когда-нибудь придется пользоваться им, вы поймете, о чем я говорю. Именно поэтому появилось множество служб, таких как *chain.com*, и наборов инструментов SDK, скрывающих неудобства протокола от разработчиков.

Таким образом, деньги, которые вы жертвуете, в действительности не покидают ваш кошелек немедленно. Они становятся лишь частью подписанной транзакции, недействительной для сети.

Когда пользователь решает создать проект, формируется сообщение-запрос в формате BIP70. Выходы определяются как обычно, и имеется лишь несколько отличий от привычного порядка перевода средств.

- Добавляется поле `title` с кратким описанием проекта в нескольких словах.
- Добавляется поле `image` с сериализованным изображением, придающим индивидуальность проекту в пользовательском интерфейсе. Хирн предусмотрел использование изображений с тем же соотношением сторон, что и у фотографий в Facebook, чтобы их можно было использовать и в других местах.
- Если присутствует поле `payment_url`, оно должно сообщать о расширенном протоколе, позволяющем за-

прашивать состояние проекта (текущую сумму пожертвований).

- Само платежное сообщение должно содержать недопустимую транзакцию с сигнатурой `SIGHASH_ANYONECANPAY`. Только общедоступные выходы, известные как UTXO (Unspent Transaction Output — неизрасходованный выход транзакции), можно израсходовать на пожертвования, которые будут считаться допустимыми для проекта. Кроме того, поле `memo` может содержать сообщение от пользователя (одобрившего проект). Допустимы также дополнительные поля с контактной информацией.

После форматирования платежное сообщение можно либо послать методом `POST` по адресу `payment_URL` для сбора и в конечном счете объединения с другими пожертвованиями, либо передать владельцу проекта иным способом (например, по электронной почте). Получив это сообщение, владелец сможет загрузить его с помощью своего клиента Lighthouse. Клиент предоставляет графический интерфейс для объединения пожертвований и отправки окончательной транзакции в сеть P2P.

Язык сценариев Bitcoin с годами стал значительно мощнее, в основном благодаря инновациям в технологии цепочек блоков и Тьюринг-полным умным контрактам из проекта Ethereum. Он хорошо подходит для реализации большинства умных контрактов, но в цепочке блоков Bitcoin поддержка Тьюринг-полных контрактов пока отсутствует. Протокол Counterparty поддерживает их, но он слишком раздут всякими ненужными особенностями. Существует, конечно, сам проект Ethereum, но в нем пока не реализована поддержка боковых цепочек, которая позволила бы использовать цепочку блоков Ethereum с безопасностью Bitcoin.

Lighthouse — отличный пример, демонстрирующий вмешательство политики в развитие протокола Bitcoin. Данному протоколу уже семь лет, и почти столько же времени Майк Хирн (Mike Hearn) входит в число основных его разработчиков. Он передал запрос на включение в ядро Bitcoin около 44 строк кода, который был тут же отвергнут. Затем этот код был принят, но после серьезных дебатов вновь исключен. Даже со своими положением и заслугами он не смог добавить простое исправление, опирающееся на поддержку флага `SIGHASH_ANYONECANPAY`.

Ведущие разработчики очень трепетно относятся к реализации протокола, что вполне оправданно — Bitcoin имеет устоявшуюся репутацию, и от его стабильности зависят миллиарды долларов. В этом смысле Bitcoin является самым ценным открытым проектом из когда-либо существовавших. По запросу Хирна о внедрении 44 строк кода разработчиками было дано 167 комментариев. Вместо списка исправлений он решил реализовать свои «запросы getuxtxo сообщений из множества UTXO для проверки пожертвований». Хирн даже заявил, что реализация упрощенной проверки платежей (Simplified Payment Verification, SPV) невозможна на данном этапе из-за необходимости преодолевать многочисленные политические препоны для внесения хоть каких-то изменений в основное ядро протокола.

Это и хорошо и плохо одновременно: хорошо потому, что слишком большое количество изменений может нарушить работу протокола, если внедрять их без тщательного анализа и обсуждения; а плохо потому, что мешает вносить действительно важные изменения. Боковые цепочки, будем надеяться, решат эту проблему, поскольку они позволяют создавать новые цепочки блоков для экспериментов, со-

храняя уровень безопасности, свойственный цепочке блоков Bitcoin.

Для Lighthouse Хирн создал протокол с названием Bitcoin TX со своим набором исправлений, который в настоящее время насчитывает около 16 активных узлов.

Кошельки SPV

Как вы помните, в децентрализованном приложении Mikro мы использовали стороннюю службу для создания и перевода монет между адресами. Библиотека Kerala содержит все необходимые для этого функции. Наше приложение получилось децентрализованным лишь частично, но уже является хорошим началом. Более децентрализованное решение, очевидно, заключается в запуске локального узла, но проблема в том, что цепочка блоков Bitcoin слишком велика для локального узла. За семь лет она выросла настолько, что для ее загрузки и синхронизации требуется порядка четырех часов и много гигабайт дискового пространства. Одним из альтернативных решений, обеспечивающих легковесность и децентрализацию, являются кошельки SPV (Simplified Payment Verification — с упрощенной проверкой платежей), которые Хирн реализовал с помощью BitcoinJ.

Можно построить такую реализацию Bitcoin, которая проверяет не все подряд, а полагается на соединение с доверенным узлом или возлагает надежды на высокую сложность как показатель достоверности. BitcoinJ как раз реализует такой режим.

Действуя в режиме SPV, клиенты могут подключаться к полным узлам и загружать только заголовки блоков. Сатоси

(Satoshi) описывает данную возможность в своей статье о Bitcoin. Клиенты могут убедиться, что заголовки правильно связаны между собой и сложность достаточно высока. После этого они запрашивают у удаленного узла транзакции, соответствующие некоторым шаблонам, вроде транзакций на ваш адрес. Копии транзакций возвращаются через ветвь Merkle, связывающую их с блоком, в котором они находятся. Протокол позволяет использовать древовидную структуру Merkle, чтобы обеспечить доказательство включения без необходимости загружать все содержимое блока.

Механизм SPV обеспечивает дополнительную оптимизацию, позволяя отбрасывать заголовки блоков, похороненные очень глубоко (то есть хранилище может исключать блоки, находящиеся ниже, чем на X заголовков). Если узел известен как заслуживающий доверия, уровень сложности не принимается в расчет. Если вы просто выбираете случайный узел, стоимость майнинга последовательности блоков с поддельной транзакцией для атакующего окажется выше, чем ценность приобретенного им в результате обмана. Меняя глубину вложения блока, можно менять скорость подтверждения и стоимость атаки.

Идентификация

```
public String signAsOwner(PledgingWallet wallet, String message,
                          @Nullable KeyParameter aesKey)
{
    DeterministicKey realKey =
        wallet.getAuthKeyFromIndexOrPubKey(authKey, authKeyIndex);
    if (realKey == null || (aesKey == null && realKey.
                    isEncrypted()))
        return null;
    return realKey.signMessage(message, aesKey);
}
```

Каждый проект получает собственный ключ аутентификации. Ключ аутентификации — это обычный ключ Bitcoin, полученный применением алгоритма secp256k1. Он хранится в кошельке пользователя и является производным от иерархии его ключей HD (Hierarchical Deterministic — иерархически детерминированные). В настоящее время проект использует его, только чтобы доказать серверу, что пользователь действительно является создателем проекта. Ключ можно было бы использовать также для подписания сообщений создателем и доказательства законности новой версии файла проекта.

Кроме того, автор получает простой в использовании шаблон BitcoinJ. Вы найдете его в репозитории GitHub (*https://github.com/bitcoinj/wallet-template*).

На рис. 5.4 изображен отличный пример хорошо проработанного интерфейса кошелька. Этот кошелек SPV реали-

Рис. 5.4. Внешний вид интерфейса кошелька

зован с применением BitcoinJ и шаблона HTML/CSS по умолчанию. Вы можете создать точно такой же интерфейс в своем децентрализованном приложении. Данное решение не идеально, поскольку в нем не используются метамонеты, так что вам будет трудно, если вообще возможно, заработать на том, что вы собираетесь создать, но это только начало. Существует также кошелек SPV для цветных монет, который называется ChromaWallet, но в нем отсутствует начальный шаблон, в отличие от BitcoinJ. Объединив шаблон из BitcoinJ с ChromaWallet, можно получить по-настоящему полезный инструмент, и я полагаю, что рано или поздно кто-то реализует данную идею.

6 La'Zooz

Что такое La'Zooz?

Приложения онлайн-такси в последние несколько лет стали появляться как грибы после дождя. Наиболее крупными из них являются Uber и Lyft, причем Uber, похоже, претендует на мировое господство. Только в 2014 году приложение Uber собрало 2 миллиарда долларов и считается одним из самых быстрорастущих стартапов в мире. Суть его проста: пользуясь преимуществом повсеместного распространения смартфонов, оно дает пользователям возможность доехать из одной точки в другую. Приложение Uber децентрализует индустрию такси, позволяя любому желающему выступить в роли водителя. Кроме того, оно позволяет любому заказать поездку из любой точки простым нажатием кнопок на телефоне, оснащенном датчиком GPS. С появлением Uber клиенты больше не должны ждать, пока на дороге появится свободное такси, а водители — блуждать по улицам в поисках клиента. Приложение Uber предоставляет услугу согласования и, очевидно, децентрализует ее. Технологии P2P во всей красе.

Но так ли все радужно? В последнее время в корпоративной культуре Uber разразилось несколько скандалов. Компания Uber получила печальную известность из-за приемов, используемых в своей деловой практике, и особенностей стимулирования и допуска водителей. Например, исполнительный директор Uber Эмиль Майкл предложил своей компании собрать компромат на одну критически настроенную женщину-репортера и других оппонентов. Другие скандалы связаны со слежкой за клиентами и особенностями оплаты труда водителей. Специальная программа God Mode в Uber, позволяющая в режиме реального времени видеть все поездки, совершаемые в любом месте, и получать доступ к данным любого клиента в социальных сетях, не раз была предметом споров. Были случаи, когда водители заваливали заказами компанию Lyft, только чтобы через некоторое время отменить их, из-за чего клиенты выбирали Uber.

Несмотря на эти проблемы, за последние несколько лет доходы Uber выросли в геометрической прогрессии и перевалили за миллиарды долларов. Компания предоставляет востребованную услугу: для поездок люди предпочитают пользоваться приложением, учитывающим местоположение пользователя, а не вызывать такси; можно с уверенностью сказать, что спрос на данную услугу едва ли начнет снижаться в ближайшее время. Но вторжение в частную жизнь и огромный дисбаланс в положении между корпорацией с миллиардными доходами и ее подрядчиками являются отрицательными чертами Uber, с которыми клиенты должны согласиться, пользуясь ее услугами.

Клиенты децентрализованной службы такси La'Zooz имеют возможность оплачивать услуги ее водителей монетами Zooz. Водители получают другое приложение, позволяющее

им «добывать» монеты Zooz просто за езду. La'Zooz реализует алгоритм с названием «доказательство перемещения» (proof-of-movement). Для определения факта движения используется система геопозиционирования GPS. Во время движения водители могут добывать монеты Zooz.

Протокол распределения

Попробуем понять, как будут распределяться монеты Zooz. Как мы уже знаем, водители вознаграждаются монетами Zooz за поездки. Сумма вознаграждения постепенно уменьшается, как и в сети Bitcoin. График на рис. 6.1 показывает, что такой подход действительно стимулирует расширение сети.

Команда разработчиков создала перспективный план развития сообщества — график будущих этапов проекта La'Zooz с точки зрения разработки, маркетинга и общего роста. Они полагают, что первые пользователи должны получать большее вознаграждение, чем последние, а также должно вознаграждаться привлечение новых пользователей в сеть.

Команда решила провести два раунда предварительной продажи, чтобы получить средства на разработку проекта. Это эквивалентно накоплению начального капитала для разработки прототипа перед официальным выходом с заявлением о выпуске нового продукта. Они создали Coinbase-хранилище с мультиподписью, требующее две из трех подписей для перевода средств. Хранилище — это простой адрес Bitcoin с мультиподписью, на который любой желающий может перевести деньги. Если двое из трех подписантов согласны, средства переводятся тому, кто посылал деньги.

Рис. 6.1. Монеты Zooz

Мультиподпись используется как умный контракт, который позволяет покупателям на распродаже посылать Bitcoin и получать взамен монеты Zooz пропорционально переведенной сумме. Существуют три возможных подписанта:

- доверенный член сообщества Bitcoin;
- независимый профессиональный аудитор;
- представитель сообщества разработчиков La'Zooz.

Кто и как будет выбирать этих людей, пока не решено. Восемь процентов всех монет, проданных на распродаже, будут

выдаваться всем покупателям, купившим монеты на предварительном этапе, в качестве поощрения. Почему именно этот процент и почему от суммы средств, проданных на распродаже? Почему бы просто не поощрить пользователей, принявших участие в проекте на предварительном этапе? Это кажется непонятным и нелогичным.

Когда в регионе имеется много водителей, но мало клиентов, или наоборот, возникает проблема. Она называется «проблемой курицы и яйца». Проект La'Zooz старается противодействовать ей с помощью алгоритма, устанавливающего пороговое число водителей в конкретном регионе. Если число водителей в регионе достигает установленного порога, приложение активируется, и только тогда клиенты получают возможность подавать заказы. Таким образом, приложение будет вводиться в действие по регионам, в точности как Uber, и полное развертывание будет производиться программно, а не вручную.

Структура ДАО

Целью La'Zooz является создание сети, управляемой сообществом, то есть где нет разницы между создателями и простыми пользователями. Любой, кто пользуется услугами La'Zooz, принадлежит к одной ДАО (децентрализованной автономной организации). В конце месяца проводится голосование, в ходе которого члены определяют вес голоса каждого члена сообщества и сумму его вознаграждения. Каждый член может проголосовать за веса голосов и суммы дивидендов тех членов, которых он знает (сеть доверия).

Новые веса рассчитываются по смешанной схеме; 75% входов, поступивших от нового голосования, и 25% от голосо-

вания предыдущего месяца. Это произвольные цифры, и они могут быть пересмотрены, но мне кажется, было бы лучше, если бы веса рассчитывались исключительно на основании нового голосования. Практика создания ДАО является относительно новой, поэтому чем проще членам будет понять, как принять участие в вашей организации и как работают организационные процессы, тем выше вероятность успеха ДАО. В конце каждого месяца, перед днем голосования, каждый член должен расписать, что он сделал для развития сети. Члены, участвующие в голосовании, смогут увидеть такие списки дел и решить, насколько они важны.

Голосование не является обязательным, и это правильно. Обеспечьте необязательность голосования, и члены не будут чувствовать себя обязанными действовать каким-то определенным образом; если они пожелают, они смогут перейти в более свободную ДАО. При создании правил для сообщества необходимо стремиться найти баланс между жесткостью и свободой. Сначала свод правил должен быть создан централизованно (создателями проекта) для получения начального документа как можно быстрее, а затем он должен дорабатываться и развиваться сообществом.

Проект La'Zooz разработал и выпустил документ (*http://www.lazooz.org/whitepaper.html*) с описанием механизма распределения, некоторыми формулами, лежащими в основе алгоритма распределения, и их планами на будущее. С тех пор как Сатоси опубликовал свою статью с описанием Bitcoin, создатели децентрализованных приложений также взяли за правило публиковать свои документы. Более того, некоторые известные венчурные инвесторы, такие как Дэвид Джонсон, считают, что публикация документов — «правильный» способ выпуска децентрализованного приложения. Я с этим не согласен. Правильный способ заключается

в том, чтобы обеспечить настоящую ценность децентрализованного приложения — дать нечто, чего не могут дать централизованные конкуренты, — и максимально просто описать его, используя приемы, известные большинству.

Если потребуется целевая страница с информацией, форум для сообщества со встроенной функцией голосования и видеоролики, поясняющие работу с вашим децентрализованным приложением, создайте их. Документы не являются обязательными и в действительности могут вызвать ненужную путаницу, если вся информация, связанная с приложением, хранится централизованно, как в случае с La'Zooz. Документ не должен повторять ваш бизнес-план, роли членов и все остальное, что связано с вашей организацией.

Как бы то ни было, команда La'Zooz создала юридическую некоммерческую организацию в Израиле, чтобы гарантировать свою чистоту перед государством. Каждый член сообщества вознаграждается и имеет взвешенный голос, пользуясь которым может влиять на дальнейшее развитие приложения, но сама команда образует юридическую организацию. В конце каждого месяца им наравне с остальными будет направляться определенная доля дивидендов. Интересно отметить, что в своем уставе они предусмотрели обязанность подчиняться мнению сообщества при определении доли средств, которую они могут получить.

Таким образом, ДАО позиционирует себя как делегативная демократия (liquid democracy). Любой обладает правом голоса, может участвовать в голосовании и делегировать свой голос; при этом создатели имеют свое представительство. Представители берут на себя основные заботы по решению организационных проблем, но, если сообщество выяснит, что они погрязли в коррупции или не предпри-

нимают необходимых усилий для поддержания и расширения сети, его члены могут выбрать и проголосовать за новый состав представителей.

Механизм вознаграждения, описанный в нормативном документе, недостаточно прозрачен и до сих пор дорабатывается. Вознаграждение должно начисляться членам за их работу в форме дивидендов. Внутренние валюты на начальном этапе не обладают ликвидностью; она появляется только с ростом сети. Внутренние монеты можно рассматривать как акции в сети. Умный контракт можно переписать так, чтобы дивиденды начислялись каждой паре из открытого/закрытого ключей, имеющей монеты Zooz.

Дивиденды должны начисляться пропорционально количеству монет, хранящемуся в конкретном кошельке. Дэн Лэример создал альтернативную криптовалюту с названием Bitshares, использующую алгоритм консенсуса «делегированное подтверждение доли» (Delegated Proof of Stake) и действующую похожим образом. В то время как монеты Zooz можно рассматривать как акции в ДАО La'Zooz и одновременно как внутреннюю валюту, дивиденды могут начисляться в Bitcoin или в Zooz. Монеты Bitcoin имеют более высокую ликвидность, тогда как монеты Zooz обладают бóльшим потенциалом роста в цене. Решение остается за создателями, но мне кажется, что лучшим выбором на роль валюты для выплаты дивидендов была бы Bitcoin.

Пользовательский интерфейс

Давайте посмотрим на некоторые дизайнерские решения, найденные в La'Zooz. Децентрализованные приложения до настоящего времени не отличались изысканными пользо-

вательскими интерфейсами, но в La'Zooz, похоже, понимают важность привлекательности оформления.

После запуска приложения, осуществляющего майнинг (добычу монет), пользователь видит приветствие и страницу со своими показателями. На рис. 6.2 и 6.3 изображены некоторые фиктивные показатели, включенные в тестовую версию. Приложение поддерживает работу в фоновом режиме и осуществляет майнинг во время движения пользователя; даже когда на переднем плане выполняются другие приложения, майнинг не прекращается.

Водитель может увидеть свой баланс в монетах Zooz (рис. 6.4), нажав на пиктограмму в боковой панели. При-

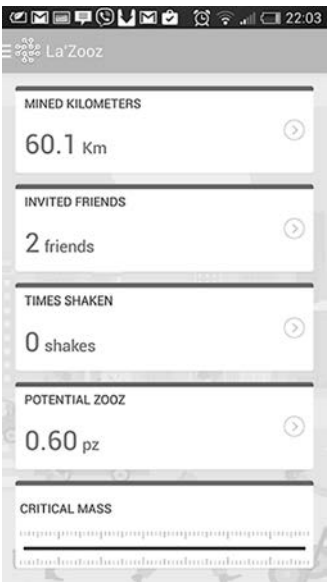


Рис. 6.2. Статистические показатели в приложении La'Zooz

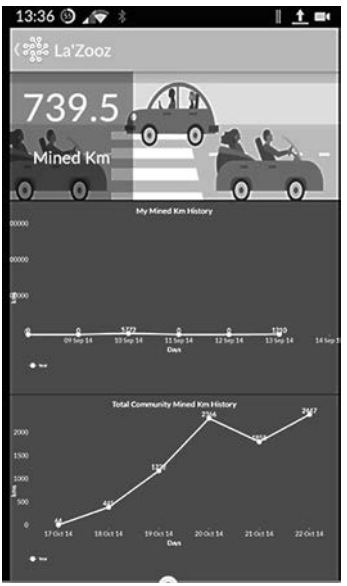


Рис. 6.3. График добычи монет

Рис. 6.4. Баланс Zooz

ложение Zooz служит кошельком для его валюты. Водитель может использовать QR-код и функцию экспорта, чтобы пересылать средства другим людям или проектам или получать средства от них. Интересным маркером является потенциальный баланс Zooz. Приложение вычисляет, сколько водитель сможет заработать, если продолжит добычу до начала следующего блока. А какой механизм противостояния атакам Сибиллы здесь используется? Что La'Zooz предпринимает, чтобы не позволить пользователю запустить несколько экземпляров приложения, осуществляющего майнинг, и выдавать себя за нескольких пользователей? Чтобы увидеть этот механизм, нужно погрузиться в код (*https://github.com/laZooz/lbm-client*).

Архитектура

Хранение и извлечение данных

В примере децентрализованного приложения Mikro мы видели, что для хранения данных можно использовать таблицы DHT, а для их извлечения — протокол BitTorrent. Это стало возможным благодаря появлению IPFS, кульминации двух технологий. OpenBazaar также использует таблицы DHT, но не имеет встроенного механизма копирования данных, что заметно снижает надежность. Если узел отключается от сети и никто перед этим не просматривал его данные, эти данные также исчезают из сети.

Разработчики Lighthouse даже не пытались использовать децентрализованное хранилище данных, потому что это слишком сложно. Было решено пересылать файлы проектов участникам, чтобы те могли загружать их в свои экземпляры Lighthouse. А как проблему хранения данных решает La'Zooz? В исходном коде можно увидеть файл *ServerComs*. Что он означает, взаимодействие с сервером (server communication)? Такое решение не выглядит децентрализованным. Давайте рассмотрим три метода в этом классе.

```
public void registerToServer(String cellphone)
{
    String url = StaticParms.BASE_SERVER_URL + "api_register";

    List<NameValuePair> params = new ArrayList<NameValuePair>();
    params.add(new BasicNameValuePair("cellphone", cellphone ));

    this.postRequestToServer(-1, -1, url, params);
}

public void
```

```
setLocation1dddsfsdfs(String UserId, String UserSecret,
                      String data)
{
    String url = StaticParms.BASE_SERVER_URL + "api_set_location";

    List<NameValuePair> params = new ArrayList<NameValuePair>();
    params.add(new BasicNameValuePair("user_id", UserId ));
    params.add(new BasicNameValuePair("user_secret", UserSecret
));
    params.add(new BasicNameValuePair("location_list", data ));
    this.postRequestToServer(-1, -1, url, params);
}

public void getUserKeyData(String UserId, String UserSecret)
{
    String url = StaticParms.BASE_SERVER_URL +
                 "api_get_user_key_data";
    List<NameValuePair> params = new ArrayList<NameValuePair>();
    params.add(new BasicNameValuePair("user_id", UserId ));
    params.add(new BasicNameValuePair("user_secret",
               UserSecret ));
    this.postRequestToServer(-1, -1, url, params);
}
```

Похоже, что эти методы служат для регистрации учетной записи пользователя на сервере, установки местоположения пользователя и извлечения пользовательских данных соответственно. А что за переменная `BASE_SERVER_URL`? Ее определение можно найти в классе `StaticParams`:

```
public static final String BASE_SERVER_URL = "https://client.
       laZooz.org/";
```

Оказывается, рассматриваемые методы служат для сохранения данных на центральном сервере и их извлечения оттуда. Как человека, давно изучающего децентрализованные приложения, такое меня не сильно удивляет. В этой области очень много лишнего шума; проекты действитель-

но могут быть яркими, иметь массу последователей и раздавать громкие обещания, однако прибегают к легким решениям таких критических задач, как хранение данных. Это может объясняться нехваткой средств на поиск мощных инструментов децентрализованного хранения, таких как IPFS, или просто отсутствием знаний того, как подобные задачи должны решаться в децентрализованных приложениях, чтобы обеспечить достаточный уровень децентрализации. Факт хранения пользовательских данных на центральном сервере делает приложение похожим на Uber, если не учитывать использование внутренней валюты и кооперативной организации вместо корпоративной. У ОС Android существует обертка для IPFS, которую разработчики La'Zooz могли бы использовать. Вы сможете найти ее по адресу: *https://github.com/dylanPowers/ipfs-android*.

Валюта

А какую цепочку блоков использует La'Zooz для выпуска внутренней валюты Zooz?

```
protected String doInBackground(String... params) {
ServerCom bServerCom = new ServerCom(MainActivity.this);
    JSONObject jsonReturnObj=null;
    try {
        MySharedPreferences msp = MySharedPreferences.
                                  getInstance();
        bServerCom.getUserKeyData(msp.getUserId(MainActivity.
                                  this),
                                  msp.getUserSecret(MainActivity.
                                                    this));
        jsonReturnObj = bServerCom.getReturnObject();
    } catch (Exception e1) {
        e1.printStackTrace();
    }
    String serverMessage = "";
```

```
try {
    if (jsonReturnObj == null)
        serverMessage = "ConnectionError";
    else {
        serverMessage = jsonReturnObj.getString("message");
        if (serverMessage.equals("success")){
            String ZoozBalance =
                jsonReturnObj.getString("Zooz_balance");
```

Метод запрашивает у сервера баланс кошелька. Значит, кошелек размещается не на локальном компьютере, а на сервере. Второй гол в пользу централизации! Это не сторонний ресурс для размещения кошельков, а собственный сервер La'Zooz. Поскольку цепочка блоков находится на их собственном сервере, не существует клиентского кода, который можно было бы изучить, чтобы выяснить ее тип; но я знаю, что они используют цепочку блоков Mastercoin, о чем они заявляли в разных социальных сетях.

Mastercoin — слой поверх цепочки блоков Bitcoin. Он вставляет данные в цепочку блоков посредством транзакций, и, с точки зрения майнеров Bitcoin, эти данные не имеют смысла. В отличие от Bitcoin или любой другой альтернативной криптовалюты, опирающейся на свою цепочку блоков, слой Mastercoin не способен действовать в роли механизма умных контрактов. Любой желающий может дважды перевести средства из заданного адреса Mastercoin. Ничто не помешает поместить в цепочку блоков конфликтующие Mastercoin-транзакции, и единственное, что делает протокол Mastercoin, — он определяет правило, позволяющее игнорировать одну транзакцию.

Но это не все. Некоторые функции La'Zooz требуют активного участия пользователей, но в протоколе нет ничего, что заставило бы их вести себя правильно. Хорошим примером

может служить функция «зарегистрировать поток данных» в протоколе Mastercoin, с помощью которой владелец адреса Bitcoin может объявить, что он будет публиковать данные, скрытые в транзакциях, исходящих от него. Владелец может, например, взять на себя обязательство еженедельно публиковать в таком потоке цены на бензин. Однако нет ничего, что потребовало бы от него посылать данные регулярно. Более того, отсутствует механизм, который мог бы помешать публиковать откровенную ложь. Этот недостаток полностью обесценивает поток данных для умных контрактов.

В отличие от Mastercoin, окрашенные монеты добавляют в цепочку блоков минимальный объем данных, поэтому они оказываются более легковесными для майнеров. В таком случае появляется возможность создавать умные контракты с помощью внутренней системы сценариев Bitcoin или боковой цепочки, поддерживающей создание Тьюринг-полных контрактов.

Контракты

На данный момент мы выяснили, что данные хранятся централизованно, на сервере проекта, а также что узел кошелька подключен к сети Mastercoin. Применение умных контрактов могло бы позволить организовать автоматическую распродажу, и создатели заявили, что для передачи средств используют Coinbase-хранилище с мультиподписью. А как насчет выплаты дивидендов? Автоматические платежи — отличительный признак умных контрактов. Контракт может находиться в цепочке блоков, и, зная, что создатели используют цепочку блоков Bitcoin через протокол Mastercoin, можно с уверенностью сказать, что для создания своих умных контрактов они должны использо-

вать язык сценариев Bitcoin. Достаточно взглянуть на главную страницу Ethereum, чтобы увидеть, что La'Zooz входит в число проектов, использующих Ethereum для поддержки умных контрактов.

Ethereum — замечательный проект, и его инструменты создания умных контрактов более зрелые, чем аналогичные инструменты Bitcoin, но дело в том, что цепочка блоков Ethereum пока не доказала свою работоспособность, и ее создателям придется или существенно уменьшить ее в размерах, или навсегда увязнуть в решении проблем с безопасностью, как сказал Гевин Андерсен (Gavin Andresen), главный разработчик протокола Bitcoin. В лучшем случае, когда протокол боковой цепочки будет выпущен, Ethereum превратится в боковую цепочку к Bitcoin, и, если кому-то действительно потребуется написать Тьюринг-полные контракты, он сможет сделать это, не доверяя безопасность своей внутренней валюты цепочке блоков Ethereum. В настоящее время использовать цепочку блоков Ethereum — не самое лучшее решение, если вы собираетесь создать свое децентрализованное приложение, приносящее прибыль.

Улучшения

La'Zooz — амбициозный стартап. Его основатели пытаются создать первую ДАО, вовлекающую работников по всему миру, которые получают дивиденды и имеют право голоса, и связанную с имеющейся правовой инфраструктурой. Пока, на стадии разработки, в проекте используется централизованный подход к хранению данных, ценностей и идентификации. На данный момент такое решение выглядит неплохо, но в будущем La'Zooz рискует получить не одну, а несколько точек отказа.

Для выпуска средств La'Zooz следует применять протокол окрашенных монет. Монеты Zooz должны использоваться и как акции сети, и как валюта. Выплату дивидендов можно было бы организовать путем создания умного контракта, выдающего дивиденды членам сети пропорционально их долям. Фактически, чтобы получить полную децентрализацию, приложение Zooz могло бы быть простым кошельком SPV; с другой стороны, при желании иметь веб-кошелек и для упрощения разработки можно было бы использовать Coinprism.

Подход к организации ДАО в проекте La'Zooz заслуживает самой высокой оценки. Если они документально закрепят, что люди, перечисленные в уставных документах компании, зарегистрированной в Израиле, обязаны подчиняться волеизъявлению сообщества, то они смогут избежать правовой неопределенности, связанной с ДАО, путем соблюдения местных норм и соблюсти принципы децентрализации посредством делегативной демократии. Голосование должно быть реализовано в виде дополнительной особенности — обычной вкладки в боковой панели.

Пользователи должны иметь возможность голосовать; любой желающий может вносить свои предложения по улучшению приложения, включению новых регионов в маркетинговую кампанию или смене лидера. Возможность голосования по таким предложениям следовало бы предоставить всем. Форум для голосования мог бы работать в стиле Reddit, когда пункты, набравшие наибольшее число голосов, перемещаются в верхнюю часть списка. Лидеры или представители ДАО, перечисленные в уставных документах, принимали бы эти предложения к исполнению, как того требуют юридические документы.

Данные не должны храниться централизованно, это само собой разумеется. Служба сохранения и извлечения данных должна быть основана на DHT, подобно IPFS. Использование сервера для хранения всех данных противоречит идее децентрализации, хотя в случае с проектом La'Zooz эту ситуацию можно расценивать немного иначе, потому что юридически он должен действовать так, как пожелает сообщество. Сообщество, скорее всего, потребует полной прозрачности; значит, никакие данные не могут быть проданы третьей стороне без одобрения сообщества.

Рабочие места в ДАО La'Zooz следует ограничить узким кругом профессий или, наоборот, включить полный комплекс. Полный комплекс профессий уже заложен в секторе стартапов, но в случае с ДАО под полным комплексом должны подразумеваться не только технические, но и любые другие специальности, обычно имеющиеся в компаниях, в том числе специалисты по маркетингу, технические специалисты и специалисты по отношениям с клиентами. Организовать наем, проверку и увольнение работников децентрализованным способом может оказаться непросто. Для входа в ДАО не должно быть никаких препятствий, а процесс проверки можно построить на уже имеющихся в La'Zooz механизмах (голосование о вознаграждении для членов и определение весов голосов). Процедуру увольнения можно было бы возбуждать по запросу недовольного члена.

Увольнение должно быть, по сути, равносильно запрету кому-то участвовать в работе, накладываемому распределенным способом. Например, недобропорядочный пользователь пробует провести DDoS-атаку на сеть или пытается выгрузить в нее детскую порнографию. Люди должны иметь возможность проголосовать за увольнение такого члена (те, кто знает его); тогда представители в DAO смогут закрыть

доступ к сети для адреса данной персоны, реализовав черный список в цепочке блоков, хранящейся локально на всех узлах. То есть, если адрес окажется в черном списке, он не сможет участвовать в транзакциях с любыми другими узлами сети. Черный список должен храниться в цепочке блоков, чтобы все узлы могли принять его.

La'Zooz не следует использовать Ethereum для создания умных контрактов, пока не будет готова боковая цепочка. Для реализации автоматизированных платежей в большинстве случаев с успехом можно применять язык сценариев Bitcoin, даже несмотря на то, что он не является Тьюринг-полным.

И, наконец, все инструкции, имеющие отношение к ДАО La'Zooz, не должны включаться в один документ. Сведения следует распределить по нескольким частям, описать максимально простым языком, понятным обывателю, и для упрощения поиска представить в виде веб-страниц, поскольку они более дружественны по отношению Google, чем документы в формате PDF.

В заключение

В данном обзоре нескольких развивающихся децентрализованных приложений я хотел поделиться с вами некоторыми мыслями, которые могли бы послужить трамплином к созданию вашего собственного приложения. Ориентируйтесь на два ключевых слова — открытость и децентрализация, — и вы не ошибетесь.

Об авторе

Сираж Равал (Siraj Raval) — разработчик децентрализованных приложений, предприниматель и технический писатель по велению души. Звезда YouTube, он все свое время посвящает собственному каналу Sirajology. Основал краудфандинговую платформу для разработчиков с названием Havi, создал несколько приложений для iOS, включая Meetup, работал в сфере хостинга для проектов с открытым исходным кодом. Сираж не только программист, но также путешественник, музыкант, постмодернист и аквалангист.

Обложка

На обложке книги «Децентрализованные приложения» изображен розовый пилобрюх (Hoplostethus mediterraneus), также известный как средиземноморский большеголов.

Эта глубоководная рыба распространена в Атлантическом и в западной части Индийского океанов. Обитает на глубинах от 100 до 1175 метров. Розовый пилобрюх достигает в длину 42 сантиметров, имеет продолговатую форму тела, большие глаза и раздвоенный хвост.

Как известно, розовый пилобрюх живет около 11 лет. Питается в основном ракообразными.

Многие представители фауны, изображаемые на обложках книг издательства O'Reilly, находятся под угрозой исчезновения; все они важны для нашего мира. Чтобы больше узнать о том, как помочь их сохранению, посетите страницу *animals.oreilly.com*.

Изображение для обложки взято из Dover Pictorial Archive. Шрифт обложки: URW Typewriter и Guardian Sans. Шрифт текста: Adobe Minion Pro; шрифт заголовков: Adobe Myriad Condensed; шрифт программного кода: Ubuntu Mono, созданный Далтоном Маагом.