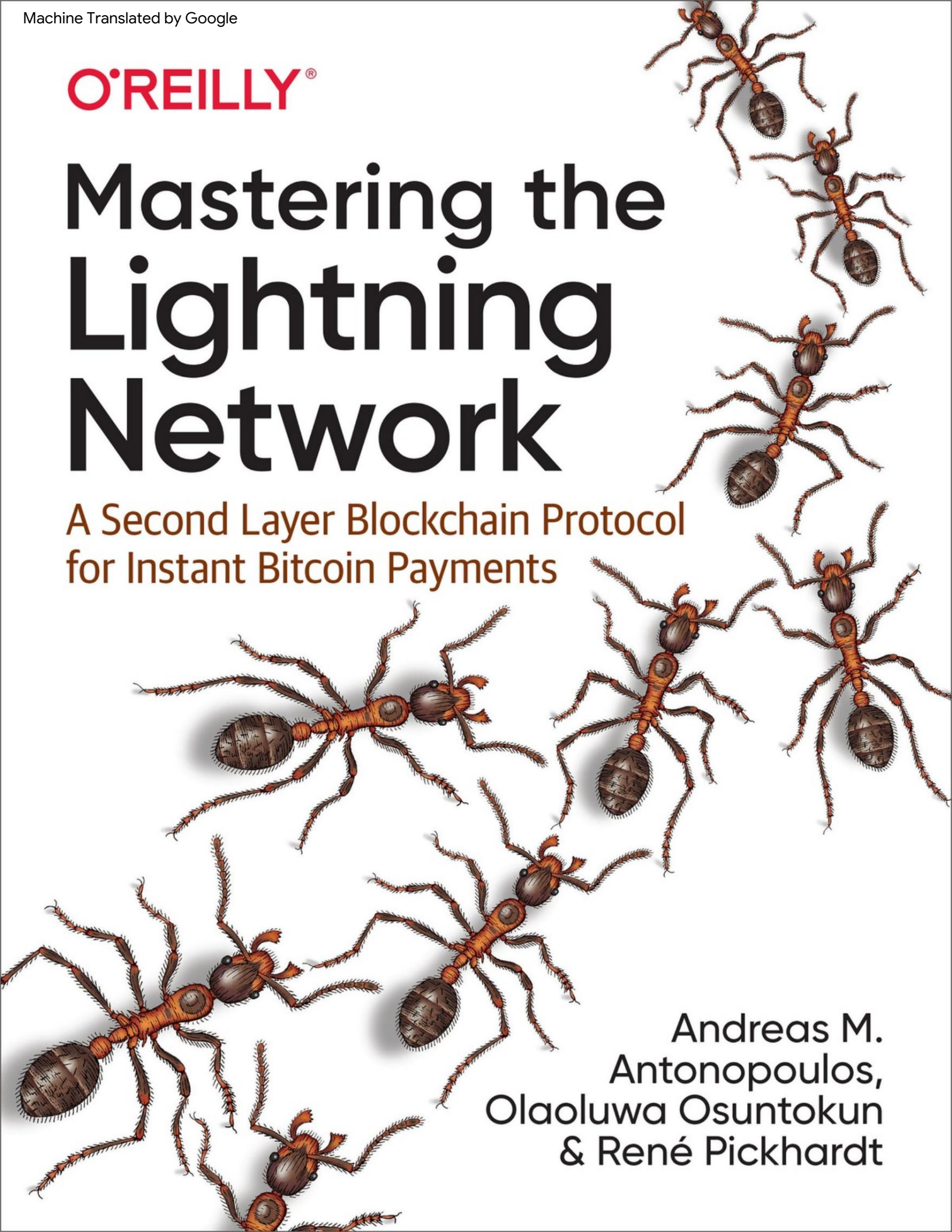
O'REILLY®

Mastering the Lightning Network

A Second Layer Blockchain Protocol for Instant Bitcoin Payments



Andreas M. Antonopoulos, Olaoluwa Osuntokun & René Pickhardt

Dominando el relámpago
La red

Un protocolo de cadena de bloques de segunda capa para Instant
Pagos de Bitcoin

Andreas M. Antonopoulos, Olaoluwa Osuntokun y René Pickhardt

Dominando la Red Lightning

por Andreas M. Antonopoulos, Olaoluwa Osuntokun y René Pickhardt

Impreso en los Estados Unidos de América.

Publicado por O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Los libros de O'Reilly se pueden comprar con fines educativos, comerciales o de promoción de ventas. Las ediciones en línea también están disponibles para la mayoría de los títulos (http://oreilly.com). Para obtener más información, comuníquese con nuestro departamento de ventas corporativo/institucional: 800-998-9938 o ***corporate@oreilly.com.***

Editora de adquisiciones: Michelle Smith

Editora de desarrollo: Michele Cronin

Editora de producción: Kristen Brown

Corrector de estilo: Tom Sullivan

Correctora: Sonia Saruba

Indexador: WordCo Indexing Services, Inc.

Diseñador de interiores: David Futato

Diseño de portada: Karen Montgomery

Ilustrador: Kate Dullea

Diciembre 2021: Primera Edición

Historial de revisiones de la primera edición

- 2021-11-22: Primer lanzamiento

Ver http://oreilly.com/catalog/errata.csp?isbn=9781492054863 para detalles de lanzamiento.

978-1-492-05486-3

[LSI]

Prefacio

Lightning Network (LN) es una red peer-to-peer de segunda capa que nos permite realizar pagos de Bitcoin "fuera de la cadena", es decir, sin comprometerlos como transacciones en la cadena de bloques de Bitcoin.

Lightning Network nos brinda pagos de Bitcoin que son seguros, baratos, rápidos y mucho más privados, incluso para pagos muy pequeños.

Sobre la base de la idea de los canales de pago, propuesta por primera vez por el inventor de Bitcoin, Satoshi Nakamoto, Lightning Network es una red enrutada de canales de pago donde los pagos "saltan" a través de una ruta de canales de pago desde el remitente hasta el destinatario.

La idea inicial de Lightning Network se propuso en 2015 en el innovador documento "The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments", de Joseph Poon y Thaddeus Dryja. Para 2017, había una Lightning Network de "prueba" ejecutándose en Internet, ya que diferentes grupos crearon implementaciones compatibles y se coordinaron para establecer algunos estándares de interoperabilidad. En 2018, Lightning Network se puso en marcha y los pagos comenzaron a fluir.

En 2019, Andreas M. Antonopoulos, Olaoluwa Osuntokun y René Pickhardt acordaron colaborar para escribir este libro. ¡Parece que hemos tenido éxito!

Público objetivo

Este libro está destinado principalmente a lectores técnicos con una comprensión de los fundamentos de Bitcoin y otras cadenas de bloques abiertas.

Las convenciones usadas en este libro

En este libro se utilizan las siguientes convenciones tipográficas:

Itálico

Indica nuevos términos, URL, direcciones de correo electrónico, nombres de archivo y extensiones de archivo.

Ancho constante

Se utiliza para listas de programas, así como dentro de párrafos para hacer referencia a elementos de programas como nombres de variables o funciones, bases de datos, tipos de datos, variables de entorno, declaraciones y palabras clave.

Negrita de ancho constante

Muestra comandos u otro texto que el usuario debe escribir literalmente.

Cursiva de ancho constante

Muestra texto que debe reemplazarse con valores proporcionados por el usuario o por valores determinados por el contexto.

PROPINA

Este elemento significa un consejo o sugerencia.

NOTA

Este elemento significa una nota general.

ADVERTENCIA

Este elemento indica una advertencia o precaución.

Ejemplos de código

Los ejemplos se ilustran en Go, C++, Python y con la línea de comandos de un sistema operativo similar a Unix. Todos los fragmentos de código están disponibles en el repositorio de GitHub en el subdirectorio de ***código .*** Bifurque el código del libro, pruebe los ejemplos de código o envíe las correcciones a través de GitHub.

Todos los fragmentos de código se pueden replicar en la mayoría de los sistemas operativos con una instalación mínima de compiladores, intérpretes y bibliotecas para los idiomas correspondientes. Cuando sea necesario, proporcionamos instrucciones básicas de instalación y ejemplos paso a paso del resultado de esas instrucciones.

Algunos de los fragmentos de código y la salida del código se han reformateado para su impresión. En todos estos casos, las líneas se han dividido con un carácter de barra invertida (\), seguido de un carácter de nueva línea. Al transcribir los ejemplos, elimine esos dos caracteres y vuelva a unir las líneas y debería ver resultados idénticos a los que se muestran en el ejemplo.

Todos los fragmentos de código utilizan valores y cálculos reales siempre que sea posible, de modo que pueda crear de un ejemplo a otro y ver los mismos resultados en cualquier código que escriba para calcular los mismos valores. Por ejemplo, las claves privadas y las claves y direcciones públicas correspondientes son todas reales.

Uso de ejemplos de código

Si tiene una pregunta técnica o un problema al usar los ejemplos de código, envíe un correo electrónico a ***bookquestions@oreilly.com.***

Este libro está aquí para ayudarle a hacer su trabajo. En general, si se ofrece un código de ejemplo con este libro, puede usarlo en sus programas y documentación. No es necesario que se comunique con nosotros para obtener permiso a menos que esté reproduciendo una parte significativa del código. Por ejemplo, escribir un programa que use varios fragmentos de código de este libro no requiere permiso. Vender o distribuir ejemplos de libros de O'Reilly requiere permiso. Responder una pregunta citando este libro y citando código de ejemplo no requiere permiso. La incorporación de una cantidad significativa de código de ejemplo de este libro en la documentación de su producto requiere permiso.

Referencias a Empresas y Productos

Todas las referencias a empresas y productos tienen fines educativos, de demostración y de referencia. Los autores no respaldan ninguna de las empresas o productos mencionados. No hemos probado el funcionamiento ni la seguridad de ninguno de los productos, proyectos o segmentos de código que se muestran en este libro. ¡Usélos bajo su propio riesgo!

Direcciones y transacciones en este libro

Las direcciones, transacciones, claves, códigos QR y datos de cadena de bloques de Bitcoin utilizados en este libro son, en su mayor parte, reales. Eso significa que puede navegar por la cadena de bloques, ver las transacciones que se ofrecen como ejemplos, recuperarlas con sus propios scripts o programas, etc.

Sin embargo, tenga en cuenta que las claves privadas utilizadas para construir las direcciones impresas en este libro han sido "quemadas". Esto significa que si envía dinero a cualquiera de estas direcciones, el dinero se perderá para siempre o (más probablemente) se apropiará, ya que cualquiera que lea el libro puede tomarlo usando las claves privadas impresas aquí.

> **ADVERTENCIA**
>
> NO ENVÍE DINERO A NINGUNA DE LAS DIRECCIONES EN ESTE LIBRO. Su dinero será tomado por otro lector, o se perderá para siempre.

Aprendizaje en línea de O'Reilly

> **NOTA**
>
> Durante más de 40 años, ***O'Reilly Media*** ha brindado capacitación, conocimientos y perspectivas en tecnología y negocios para ayudar a las empresas a tener éxito.

Nuestra red única de expertos e innovadores comparte su conocimiento y experiencia a través de libros, artículos y nuestra plataforma de aprendizaje en línea.

La plataforma de aprendizaje en línea de O'Reilly le brinda acceso a pedido a cursos de capacitación en vivo, rutas de aprendizaje en profundidad, entornos de codificación interactivos y una amplia colección de texto y video de O'Reilly y más de 200 editores más. Para obtener más información, visite ***http:// oreilly.com.***

Cómo contactarnos

La información sobre cómo ***dominar Lightning Network ,*** así como la edición abierta y las traducciones, están disponibles en ***https:// lnbook.info.***

Dirija sus comentarios y preguntas sobre este libro a la editorial:

O'Reilly Media, Inc.

1005 Carretera Gravenstein Norte

Sebastopol, CA 95472

800-998-9938 (en los Estados Unidos o Canadá)

707-829-0515 (internacional o local)

707-829-0104 (fax)

Envíe un correo electrónico ***a bookquestions@oreilly.com*** para comentar o hacer preguntas técnicas sobre este libro.

Para noticias e información sobre nuestros libros y cursos, visite ***http://oreilly.com.***

Encuéntrenos en Facebook: ***http://facebook.com/oreilly***

Síganos en Twitter: ***http://twitter.com/oreillymedia***

Míranos en YouTube: ***http://www.youtube.com/oreillymedia***

Contactando a Andreas

Puede ponerse en contacto con Andreas M. Antonopoulos en su sitio personal: ***https://aantonop.com***

Suscríbete al canal de Andreas en YouTube: ***https://www.youtube.com/aantonop***

Me gusta la página de Andreas en Facebook: ***https://www.facebook.com/AndreasMAntonopoulos***

Siga a Andreas en Twitter: ***https://twitter.com/aantonop***

Conéctese con Andreas en LinkedIn: ***https://linkedin.com/company/aantonop***

Andreas también quisiera agradecer a los patrocinadores que apoyan su trabajo a través de donaciones mensuales. Puedes apoyar a Andreas en Patreon en ***https://patreon.com/aantonop.***

Contactando a René

Puede ponerse en contacto con René Pickhardt en su sitio personal: ***https:// ln.rene pickhardt.de***

Suscríbete al canal de René en YouTube: ***https:// www.youtube.com/ user/ RenePickhardt***

Sigue a René en Twitter: ***https:// twitter.com/ renepickhardt***

Conéctese con René en LinkedIn: ***https:// www.linkedin.com/ in/ rene pickhardt-80313744***

René también quisiera agradecer a todos los patrocinadores que apoyan su trabajo a través de donaciones mensuales. Puedes apoyar a René en Patreon en ***https:// patreon.com/ renepickhardt.***

O puede apoyar su trabajo directamente con Bitcoin (también a través de Lightning Network) en ***https:// donate.ln.rene-pickhardt.de*** por lo que René agradece tanto como a sus patreons.

Contactando a Olaoluwa Osuntokun

Puede ponerse en contacto con Olaoluwa Osuntokun en su dirección de correo electrónico profesional: ***laolu@lightning.engineering***

Siga a Olaoluwa en Twitter: ***https:// twitter.com/ roasbeef***

Agradecimientos por Andreas

Le debo mi amor por las palabras y los libros a mi madre, Theresa, quien me crió en una casa con libros cubriendo todas las paredes. Mi madre también me compró mi primera computadora en 1982, a pesar de que se describe a sí misma como una tecnófoba. Mi padre, Menelaos, un ingeniero civil que publicó su primer libro a los 80 años, fue quien me enseñó el pensamiento lógico y analítico y el amor por la ciencia y la ingeniería.

Gracias a todos por apoyarme a lo largo de este camino.

Agradecimientos por René

Quiero agradecer al sistema educativo alemán a través del cual adquirí los conocimientos sobre los que se construye mi trabajo. Es uno de los mayores regalos que me dieron. Del mismo modo, quiero agradecer al sistema de salud pública alemán y a todas las personas que dedican su tiempo a trabajar en esa industria. Su esfuerzo y resistencia los convierten en mis héroes personales y nunca olvidaré la ayuda, la paciencia y el apoyo que recibí cuando lo necesitaba. Gracias a todos los estudiantes a los que se me permitió enseñar y que participaron en debates y preguntas interesantes. De ti aprendí más. También estoy agradecido con la comunidad de Bitcoin y Lightning Network que me dio una calurosa bienvenida y con los entusiastas y personas privadas que apoyaron financieramente y continúan apoyando mi trabajo. En particular, estoy agradecido con todos los desarrolladores de código abierto (no solo Bitcoin y Lightning Network) y con las personas que los financian para hacer posible esa tecnología. Un agradecimiento especial a mis coautores por cabalgar conmigo a través de la tormenta. Por último, pero no menos importante, estoy agradecido con mis seres queridos.

Agradecimientos por Olaoluwa Osuntokun

Me gustaría agradecer al increíble equipo de Lightning Labs, ya que sin todos ustedes, no existiría LND. También me gustaría agradecer al conjunto original de autores de la especificación BOLT: Rusty Russell, Fabrice Drouin, Conner Fromnkchet, Pierre-Marie Padiou, Lisa Neigut y Christian Decker. Por último, pero no menos importante, me gustaría agradecer a Joseph Poon y Tadge Dryja, los autores del artículo original de Lightning Network, ya que sin ellos, no habría Lightning Network sobre el cual escribir un libro.

Contribuciones

Muchos colaboradores ofrecieron comentarios, correcciones y adiciones al libro, ya que fue escrito en colaboración en GitHub.

A continuación se muestra una lista ordenada alfabéticamente de todos los colaboradores de GitHub, incluidas sus ID de GitHub entre paréntesis:

- 8go (@8go)
- Aaqil Aziz (@batmanscode)
- Alexander Gnip (@quantumcthulhu)
- Alfa Q. Smith (@alpha_github_id)
- Ben Skee (@benskee)
- Brian L. McMichael (@brianmcmichael)
- CandleHater (@CandleHater)
- Daniel Gockel (@dancodery)
- Dapeng Li (@luislee818)
- Darius E. Parvin (@DariusParvin)
- Doru Muntean (@chriton)
- Eduardo Lima III (@elima-iii)
- Emilio Norrmann (@enorrmann)
- Francisco Calderón (@grunch)
- Francisco Requena (@FrankyFFV)
- François Degros (@fdegros)
- Giovanni Zotta (@GiovanniZotta)
- Gustavo Silva (@GustavoRSSilva)
- Guy Thayakorn (@saguywalker)
- Haoyu Lin (@HAOYUatHZ)
- Hatim Boufnichel (@boufni95)

- Imran Lorgat (@ImranLorgat)
- Jeffrey McLarty (@jnmclarty)
- John Davies (@tigeryant)
- Julien Wendling (@trigger67)
- Jussi Tiira (@juhi24)
- Kory Newton (@korynewton)
- Lawrence Webber (@lwebbz)
- Luigi (@gin)
- Maximiliano Karasz (@mknoszlig)
- Omega X. Último (@omega_github_id)
- Owen Gunden (@ogunden)
- Patrick Lemke (@PatrickLemke)
- Pablo Wackerow (@wackerow)
- Randy McMillan (@RandyMcMillan)
- René Kohnke (@rene78)
- Ricardo Marqués (@RicardoM17)
- Sebastián Falbesoner (@theStack)
- Sergei Tikhomirov (@s-tikhomirov)
- Severin Alexander Bühler (@SeverinAlexB)
- Simone Bovi (@SimoneBovi)
- Srijan Bhushan (@srijanb)
- Taylor Masterson (@tjmasterson)
- Umar Bolatov (@bolatovumar)

- Warren Wan (@wlwanpan)
- Yibin Zhang (@z4y1b2)
- Zachary Haddenham (@senf42)

Sin la ayuda ofrecida por todos los que se enumeran aquí, este libro no habría sido posible. Sus contribuciones demuestran el poder del código abierto y la cultura abierta, y estamos eternamente agradecidos por su ayuda.

Gracias.

Fuentes

Parte del material de este libro proviene de una variedad de fuentes de dominio público, fuentes de licencia abierta o con permiso. Consulte el Apéndice D para obtener detalles sobre la fuente, la licencia y la atribución.

Parte I. Comprensión de Lightning Network

Una descripción general de Lightning Network adecuada para cualquier persona interesada en comprender los conceptos básicos y el uso de Lightning Network.

Capítulo 1 Introducción

¡Bienvenido a ***Mastering the Lightning Network!***

Lightning Network (a menudo abreviado como LN) está cambiando la forma en que las personas intercambian valor en línea, y es uno de los avances más emocionantes en la historia de Bitcoin. Hoy, en 2021, Lightning Network todavía está en pañales. Lightning Network es un protocolo para usar Bitcoin de una manera inteligente y no obvia. Es una tecnología de segunda capa sobre Bitcoin.

El concepto de Lightning Network se propuso en 2015 y la primera implementación se lanzó en 2018. A partir de 2021, solo comenzamos a ver las oportunidades que Lightning Network brinda a Bitcoin, incluida la mejora de la privacidad, la velocidad y la escala. Con el conocimiento básico de Lightning Network, puede ayudar a dar forma al futuro de la red y al mismo tiempo crear oportunidades para usted.

Suponemos que ya tiene algunos conocimientos básicos sobre Bitcoin, pero si no es así, no se preocupe: le explicaremos los conceptos más importantes de Bitcoin, aquellos que debe conocer para comprender Lightning Network, en el Apéndice A.
Si desea obtener más información sobre Bitcoin, puede leer ***Mastering Bitcoin,*** 2nd edition, de Andreas M. Antonopoulos (O'Reilly), disponible de forma gratuita en línea.

Si bien la mayor parte de este libro está escrito para programadores, los primeros capítulos están escritos para que cualquier persona pueda acceder a ellos, independientemente de su experiencia técnica. En este capítulo, comenzaremos con algo de terminología, luego pasaremos a analizar la confianza y su aplicación en estos sistemas y, finalmente, analizaremos la historia y el futuro de Lightning Network. Empecemos.

Conceptos básicos de Lightning Network

A medida que exploramos cómo funciona realmente Lightning Network, encontraremos cierta terminología técnica que, al principio, podría ser un poco confusa. Si bien todos estos conceptos y términos se explicarán en detalle a medida que avanzamos en el libro y se definen en el glosario, algunas definiciones básicas ahora facilitarán la comprensión de los conceptos en los próximos dos capítulos. Si aún no comprende todas las palabras de estas definiciones, está bien. Comprenderás más a medida que avances en el texto.

cadena de bloques

Un libro mayor de transacciones distribuidas, producido por una red de computadoras. Bitcoin, por ejemplo, es un sistema que produce una cadena de bloques. Lightning Network no es en sí misma una cadena de bloques, ni produce una cadena de bloques. Es una red que se basa en una cadena de bloques externa existente para su seguridad.

Firma digital

Una firma digital es un esquema matemático para verificar la autenticidad de mensajes o documentos digitales. Una firma digital válida le da al destinatario motivos para creer que el mensaje fue creado por un remitente conocido, que el remitente no puede negar haber enviado el mensaje y que el mensaje no fue alterado en tránsito.

Función hash

Una función hash criptográfica es un algoritmo matemático que asigna datos de tamaño arbitrario a una cadena de bits de un tamaño fijo (un hash) y está diseñado para ser una función unidireccional, es decir, una función que no es factible de invertir.

Nodo

Una computadora que participa en una red. Un nodo Lightning es una computadora que participa en Lightning Network. Un nodo Bitcoin es una computadora que participa en la red Bitcoin. Normalmente, un usuario de LN ejecutará un nodo Lightning ***y*** un nodo Bitcoin.

Dentro de la cadena versus fuera de la cadena

Un pago está ***en cadena*** si se registra como una transacción en la cadena de bloques de Bitcoin (u otra cadena de bloques subyacente). Los pagos enviados a través de canales de pago entre nodos Lightning y que no son visibles en la cadena de bloques subyacente se denominan pagos ***fuera de la cadena .*** Por lo general, en Lightning Network, las únicas transacciones en cadena son las que se usan para abrir y cerrar un canal de pago Lightning. Existe un tercer tipo de transacción de modificación de canal, llamada empalme, que se puede utilizar para agregar/eliminar la cantidad de fondos comprometidos en un canal.

Pago

Cuando se intercambia valor en Lightning Network, lo llamamos "pago" en comparación con una "transacción" en la cadena de bloques de Bitcoin.

Canal de pago

Una ***relación financiera*** entre dos nodos en Lightning Network, generalmente implementada mediante transacciones de Bitcoin de múltiples firmas que comparten el control sobre Bitcoin entre los dos nodos Lightning.

Enrutamiento versus envío

A diferencia de Bitcoin, donde las transacciones se "envían" transmitiéndolas a todos, Lightning es una red enrutada donde los pagos se "enrutan" a través de uno o más canales de pago siguiendo una ***ruta*** desde el remitente hasta el destinatario.

Transacción

Una estructura de datos que registra la transferencia de control sobre algunos fondos (por ejemplo, algo de bitcoin). Lightning Network se basa en las transacciones de Bitcoin (o las de otra cadena de bloques) para rastrear el control de los fondos.

Se pueden encontrar definiciones más detalladas de estos y muchos otros términos en el Glosario. A lo largo de este libro, explicaremos qué significan estos conceptos

y cómo funcionan realmente estas tecnologías.

PROPINA

A lo largo de este libro, verá "Bitcoin" con la primera letra en mayúscula, que se refiere al ***sistema Bitcoin*** y es un nombre propio. También verá "bitcoin", con una b minúscula, que se refiere a la unidad monetaria. Cada bitcoin se subdivide en 100 millones de unidades, cada una llamada "satoshi" (singular) o "satoshis" (plural).

Ahora que está familiarizado con estos términos básicos, pasemos a un concepto con el que ya se siente cómodo: la confianza.

Confianza en Redes Descentralizadas

A menudo escuchará a personas llamar a Bitcoin y Lightning Network "sin confianza". A primera vista esto es confuso. Después de todo, ¿no es la confianza algo bueno? ¡Los bancos incluso lo usan en sus nombres! ¿No es malo un sistema “sin confianza”, un sistema carente de confianza?

El uso de la palabra “sin confianza” pretende transmitir la capacidad de operar sin ***necesidad*** de confiar en los demás participantes del sistema. En un sistema descentralizado como Bitcoin, siempre puede optar por realizar transacciones con alguien en quien confíe. Sin embargo, el sistema garantiza que no pueda ser engañado incluso si no puede confiar en la otra parte en una transacción. La confianza es una propiedad agradable en lugar de imprescindible del sistema.

Compare eso con los sistemas tradicionales como la banca, donde debe depositar su confianza en un tercero, ya que controla su dinero. Si el banco viola su confianza, es posible que pueda encontrar algún recurso de un regulador o tribunal, pero a un costo enorme de tiempo, dinero y esfuerzo.

Sin confianza no significa desprovisto de confianza. Significa que la confianza no es un requisito previo necesario para todas las transacciones y que puede realizar transacciones incluso con personas en las que no confía porque el sistema evita las trampas.

Antes de entrar en cómo funciona Lightning Network, es importante comprender un concepto básico que subyace a Bitcoin, Lightning Network y muchos otros sistemas similares: algo que llamamos un ***protocolo de equidad.*** Un protocolo de equidad es una forma de lograr resultados justos entre los participantes, que no necesitan confiar unos en otros, sin necesidad de una autoridad central, y es la columna vertebral de los sistemas descentralizados como Bitcoin.

Equidad sin autoridad central

Cuando las personas tienen intereses contrapuestos, ¿cómo pueden establecer suficiente confianza para participar en algún comportamiento cooperativo o transaccional? La respuesta a esta pregunta se encuentra en el centro de varias disciplinas científicas y humanísticas, como la economía, la sociología, la psicología del comportamiento y las matemáticas.
Algunas de esas disciplinas nos dan respuestas “suaves” que dependen de conceptos como la reputación, la justicia, la moralidad e incluso la religión. Otras disciplinas nos dan respuestas concretas que dependen únicamente del supuesto de que los participantes en estas interacciones actuarán racionalmente, con su propio interés como principal objetivo.

En términos generales, hay varias formas de garantizar resultados justos en las interacciones entre personas que pueden tener intereses contrapuestos:

Requerir confianza

> Solo interactúa con personas en las que ya confía, debido a interacciones anteriores, reputación o relaciones familiares. Esto funciona lo suficientemente bien a pequeña escala, especialmente dentro de familias y grupos pequeños, que es la base más común para el comportamiento cooperativo. Desafortunadamente, no escala y sufre de sesgo tribalista (dentro del grupo).

Imperio de la ley

> Establezca reglas para las interacciones que son aplicadas por una institución. Esto escala mejor, pero no puede escalar a nivel mundial debido a las diferencias en las costumbres y tradiciones, así como a la incapacidad de escalar las instituciones de aplicación. Un efecto secundario desagradable de esta solución es que las instituciones

se vuelven más y más poderosos a medida que crecen y eso puede llevar a la corrupción.

Terceros de confianza

Ponga un intermediario en cada interacción para hacer cumplir la equidad. Combinado con el “estado de derecho” para supervisar a los intermediarios, esto escala mejor, pero adolece del mismo desequilibrio de poder: los intermediarios se vuelven muy poderosos y pueden atraer la corrupción. La concentración de poder conduce al riesgo sistémico y al fracaso sistémico ("demasiado grande para fallar").

Protocolos de equidad teórica del juego

Esta última categoría surge de la combinación de Internet y la criptografía y es el tema de esta sección. Veamos cómo funciona y cuáles son sus ventajas y desventajas.

Protocolos de confianza sin intermediarios

Los sistemas criptográficos como Bitcoin y Lightning Network son sistemas que le permiten realizar transacciones con personas (y computadoras) en las que no confía. Esto a menudo se denomina operación "sin confianza", aunque en realidad no es sin confianza. Debe confiar en el software que ejecuta y debe confiar en que el protocolo implementado por ese software dará como resultado resultados justos.

La gran distinción entre un sistema criptográfico como este y un sistema financiero tradicional es que en las finanzas tradicionales tiene un ***tercero de confianza,*** por ejemplo, un banco, para garantizar que los resultados sean justos. Un problema importante con tales sistemas es que le dan demasiado poder al tercero y también son vulnerables a un ***único punto de falla.*** Si el propio tercero de confianza viola la confianza o intenta engañar, la base de la confianza se rompe.

A medida que estudie los sistemas criptográficos, notará un cierto patrón: en lugar de depender de un tercero de confianza, estos sistemas intentan evitar resultados injustos mediante el uso de un sistema de incentivos y desincentivos. En

sistemas criptográficos confías en el ***protocolo,*** que es efectivamente un sistema con un conjunto de reglas que, si se diseña correctamente, aplicará correctamente los incentivos y desincentivos deseados. La ventaja de este enfoque es doble: no solo evita confiar en un tercero, sino que también reduce la necesidad de imponer resultados justos. Mientras los participantes sigan el protocolo acordado y permanezcan dentro del sistema, el mecanismo de incentivos en ese protocolo logra resultados justos sin aplicación.

El uso de incentivos y desincentivos para lograr resultados justos es un aspecto de una rama de las matemáticas llamada ***teoría de juegos,*** que estudia "modelos de interacción estratégica entre tomadores de decisiones racionales".[1] Los sistemas criptográficos que controlan las interacciones financieras entre los participantes, como Bitcoin y Lightning Network, se basan en gran medida en la teoría del juego para evitar que los participantes hagan trampa y permitir que los participantes que no confían entre sí logren resultados justos.

Si bien la teoría de juegos y su uso en sistemas criptográficos pueden parecer confusos y desconocidos al principio, es probable que ya esté familiarizado con estos sistemas en su vida cotidiana; simplemente no los reconoces todavía. En la siguiente sección usaremos un ejemplo simple de la infancia para ayudarnos a identificar el patrón básico. Una vez que comprenda el patrón básico, lo verá en todas partes en el espacio de la cadena de bloques y llegará a reconocerlo rápida e intuitivamente.

En este libro, llamamos a este patrón ***protocolo de equidad,*** definido como un proceso que utiliza un sistema de incentivos y/o desincentivos para garantizar resultados justos para los participantes que no confían entre sí. La aplicación de un protocolo de equidad solo es necesaria para garantizar que los participantes no puedan escapar de los incentivos o desincentivos.

Un protocolo de equidad en acción

Veamos un ejemplo de un protocolo de equidad con el que quizás ya esté familiarizado.

Imagine un almuerzo familiar, con un padre y dos hijos. Los niños son quisquillosos con la comida y lo único que aceptarán comer son patatas fritas. los

padre ha preparado un tazón de papas fritas ("papas fritas" o "chips" según el dialecto inglés que use). Los dos hermanos deben compartir el plato de papas fritas. El padre debe garantizar una distribución justa de fichas a cada niño; de lo contrario, el padre tendrá que escuchar quejas constantes (tal vez todo el día), y siempre existe la posibilidad de que una situación injusta se convierta en violencia. ¿Qué debe hacer un padre?

Hay algunas formas diferentes en que se puede lograr la equidad en esta interacción estratégica entre dos hermanos que no confían el uno en el otro y tienen intereses contrapuestos. El método ingenuo pero comúnmente usado es que los padres usen su autoridad como un tercero de confianza: dividen el tazón de papas fritas en dos porciones. Esto es similar a las finanzas tradicionales, donde un banco, contador o abogado actúa como un tercero de confianza para evitar trampas entre dos partes que desean realizar transacciones.

El problema con este escenario es que otorga mucho poder y responsabilidad en manos del tercero de confianza. En este ejemplo, el padre es totalmente responsable de la asignación equitativa de fichas y las partes simplemente esperan, observan y se quejan. Los niños acusan a los padres de tener favoritos y no distribuir las fichas de manera justa. Los hermanos pelean por las fichas, gritando "¡esa ficha es más grande!" y arrastrando al padre a su lucha. Suena horrible, ¿no? ¿Debe el padre gritar más fuerte? ¿Quitar todas las fichas? ¿Amenazar con no volver a hacer papas fritas y dejar que esos niños desagradecidos pasen hambre?

Existe una solución mucho mejor: a los hermanos se les enseña a jugar un juego llamado "divide y elige". En cada almuerzo, un hermano divide el tazón de papas fritas en dos porciones y el ***otro*** hermano elige qué porción quiere.
Casi de inmediato, los hermanos descubren la dinámica de este juego. Si el que divide comete un error o intenta hacer trampa, el otro hermano puede "castigarlo" eligiendo el tazón más grande. Lo mejor para ambos hermanos, pero especialmente para el que está dividiendo el tazón, es jugar limpio. Solo el tramposo pierde en este escenario. El padre ni siquiera tiene que usar su autoridad o hacer cumplir la justicia. Todo lo que el padre tiene que hacer es ***hacer cumplir el protocolo;*** siempre que los hermanos no puedan escapar de sus roles asignados de "divisor" y "selector", el protocolo en sí mismo garantiza un resultado justo sin

la necesidad de cualquier intervención. El padre no puede tener favoritos o distorsionar el resultado.

> **ADVERTENCIA**
>
> Si bien las infames batallas de fichas de la década de 1980 ilustran claramente el punto, cualquier similitud entre el escenario anterior y cualquiera de las experiencias reales de la infancia de los autores con sus primos es pura coincidencia... ¿o no?

Primitivas de seguridad como bloques de construcción

Para que un protocolo de equidad como este funcione, es necesario que existan ciertas garantías, o ***primitivas de seguridad,*** que puedan combinarse para garantizar el cumplimiento. La primera primitiva de seguridad es ***la ordenación/ secuenciación estricta del tiempo:*** la acción de "dividir" debe ocurrir antes que la acción de "elegir". No es inmediatamente obvio, pero a menos que pueda garantizar que la acción A ocurra antes que la acción B, entonces el protocolo se desmorona. La segunda primitiva de seguridad es ***compromiso con no repudio.*** Cada hermano debe comprometerse con su elección de rol: ya sea divisor o selector. Además, una vez que se ha completado la división, el divisor se compromete con la división que creó; no puede repudiar esa elección y volver a intentarlo.

Los sistemas criptográficos ofrecen una serie de primitivas de seguridad que se pueden combinar de diferentes maneras para construir un protocolo de equidad. Además de la secuenciación y el compromiso, también podemos utilizar muchas otras herramientas:

- Funciones hash para datos de huellas dactilares, como una forma de compromiso o como base para una firma digital
- Firmas digitales para autenticación, no repudio y prueba de propiedad de un secreto
- Cifrado/descifrado para restringir el acceso a la información solo a participantes autorizados

Esta es solo una pequeña lista de toda una "colección de fieras" de seguridad y primitivas criptográficas que están en uso. Primitivos más básicos y

Las combinaciones se inventan todo el tiempo.

En nuestro ejemplo de la vida real, vimos una forma de protocolo de equidad llamado "dividir y elegir". Este es solo uno de una miríada de protocolos de equidad diferentes que se pueden construir combinando los componentes básicos de las primitivas de seguridad de diferentes maneras. Pero el patrón básico es siempre el mismo: dos o más participantes interactúan sin confiar el uno en el otro mediante una serie de pasos que forman parte de un protocolo acordado. Los pasos del protocolo organizan incentivos y desincentivos para garantizar que, si los participantes son racionales, hacer trampa es contraproducente y la justicia es el resultado automático.
La aplicación no es necesaria para obtener resultados justos; solo es necesaria para evitar que los participantes rompan el protocolo acordado.

Ahora que comprende este patrón básico, comenzará a verlo en todas partes en Bitcoin, Lightning Network y muchos otros sistemas.
Veamos algunos ejemplos específicos a continuación.

Ejemplo del Protocolo de Equidad

El ejemplo más destacado de un protocolo de equidad es el algoritmo de consenso de Bitcoin, Prueba de trabajo (PoW). En Bitcoin, los mineros compiten para verificar transacciones y agregarlas en bloques. Para asegurarse de que los mineros no hagan trampa, sin confiarles autoridad, Bitcoin utiliza un sistema de incentivos y desincentivos. Los mineros tienen que usar electricidad y dedicar hardware a hacer "trabajo" que está incrustado como una "prueba" dentro de cada bloque.
Esto se logra debido a una propiedad de las funciones hash donde el valor de salida se distribuye aleatoriamente en todo el rango de posibles salidas. Si los mineros logran producir un bloque válido lo suficientemente rápido, son recompensados al obtener la recompensa del bloque por ese bloque. Obligar a los mineros a usar mucha electricidad antes de que la red considere su bloque significa que tienen un incentivo para validar correctamente las transacciones en el bloque. Si hacen trampa o cometen algún tipo de error, su bloque es rechazado y la electricidad que usaron para “probarlo” se desperdicia. Nadie necesita obligar a los mineros a producir bloques válidos; la recompensa y el castigo los incentivan a hacerlo. Todo lo que el protocolo debe hacer es asegurarse de que solo se acepten bloques válidos con Prueba de trabajo.

El patrón del protocolo de equidad también se puede encontrar en muchos aspectos diferentes de Lightning Network:

- Aquellos que financian canales se aseguran de tener una transacción de reembolso firmada antes de publicar la transacción de financiación.
- Cada vez que un canal se mueve a un nuevo estado, el estado anterior se "revoca" al garantizar que si alguien intenta transmitirlo, perderá todo el saldo y será castigado.
- Aquellos que reenvían pagos saben que si envían fondos, pueden obtener un reembolso o recibir el pago del nodo que los precede.

Una y otra vez, vemos este patrón. Los resultados justos no son impuestos por ninguna autoridad. Emergen como la consecuencia natural de un protocolo que premia la equidad y castiga el engaño, un protocolo de equidad que aprovecha el interés propio dirigiéndolo hacia resultados justos.

Bitcoin y Lightning Network son implementaciones de protocolos de equidad. Entonces, ¿por qué necesitamos Lightning Network? ¿No es suficiente Bitcoin?

Motivación para Lightning Network

Bitcoin es un sistema que registra transacciones en un libro público replicado globalmente. Cada transacción es vista, validada y almacenada por cada computadora participante. Como puedes imaginar, esto genera una gran cantidad de datos y es difícil de escalar.

A medida que Bitcoin y la demanda de transacciones crecían, el número de transacciones en cada bloque aumentaba hasta que finalmente alcanzaba el límite de tamaño del bloque. Una vez que los bloques están "llenos", el exceso de transacciones se deja esperando en una cola. Muchos usuarios aumentarán las tarifas que están dispuestos a pagar para comprar espacio para sus transacciones en el siguiente bloque.

Si la demanda sigue superando la capacidad de la red, un número cada vez mayor de transacciones de usuarios quedarán esperando sin confirmar. La competencia por las tarifas también aumenta el costo de cada transacción, lo que hace que muchas transacciones de menor valor (por ejemplo, microtransacciones) sean completamente antieconómicas durante los períodos de demanda particularmente alta.

Para resolver este problema, podríamos aumentar el límite de tamaño de bloque para crear espacio para más transacciones. Un aumento en la "oferta" de espacio en bloque conducirá a un precio de equilibrio más bajo para las tarifas de transacción.

Sin embargo, aumentar el tamaño del bloque traslada el costo a los operadores de nodos y les obliga a gastar más recursos para validar y almacenar la cadena de bloques. Debido a que las cadenas de bloques son protocolos de chismes, se requiere que cada nodo conozca y valide cada transacción que ocurre en la red.
Además, una vez validada, cada transacción y bloque debe propagarse a los “vecinos” del nodo, multiplicando los requisitos de ancho de banda. Como tal, cuanto mayor sea el tamaño del bloque, mayores serán los requisitos de ancho de banda, procesamiento y almacenamiento para cada nodo individual. Aumentar la capacidad de transacción de esta manera tiene el efecto indeseable de centralizar el sistema al reducir el número de nodos y operadores de nodos. Dado que los operadores de nodos no reciben compensación por ejecutar nodos, si los nodos son muy costosos de ejecutar, solo unos pocos operadores de nodos bien financiados continuarán ejecutando nodos.

Cadenas de bloques escalables

Los efectos secundarios de aumentar el tamaño del bloque o disminuir el tiempo del bloque con respecto a la centralización de la red son graves, como muestran algunos cálculos con los números.

Supongamos que el uso de Bitcoin crece de modo que la red tiene que procesar 40 000 transacciones por segundo, que es el nivel de procesamiento de transacciones aproximado de la red de Visa durante el pico de uso.

Suponiendo un promedio de 250 bytes por transacción, esto daría como resultado un flujo de datos de 10 megabytes por segundo (MBps) u 80 megabits por segundo (Mbps) solo para poder recibir todas las transacciones. Esto no incluye la sobrecarga de tráfico de reenviar la información de la transacción a otros

colegas. Si bien 10 MBps no parece extremo en el contexto de la fibra óptica de alta velocidad y las velocidades móviles 5G, excluiría efectivamente a cualquier persona que no pueda cumplir con este requisito de ejecutar un nodo, especialmente en países donde Internet de alto rendimiento no es asequible o no está ampliamente disponible. .

Los usuarios también tienen muchas otras demandas en su ancho de banda y no se puede esperar que gasten tanto solo para recibir transacciones.

Además, almacenar esta información localmente daría como resultado 864 gigabytes por día. Esto es aproximadamente un terabyte de datos, o el tamaño de un disco duro.

La verificación de 40 000 firmas del algoritmo de firma digital de curva elíptica (ECDSA) por segundo también es apenas factible (consulte este artículo en StackExchange), lo que hace que la ***descarga del bloque inicial (IBD)*** de la cadena de bloques de Bitcoin (sincronización y verificación de todo a partir del bloque de génesis) sea casi imposible sin hardware muy costoso.

Si bien 40.000 transacciones por segundo parece mucho, solo alcanza la paridad con las redes de pago financieras tradicionales en las horas pico.
Es probable que las innovaciones en los pagos de máquina a máquina, las microtransacciones y otras aplicaciones impulsen la demanda a muchos pedidos por encima de eso.

En pocas palabras: no puede escalar una cadena de bloques para validar las transacciones de todo el mundo de forma descentralizada.

Pero, ¿y si no se requiriera que cada nodo conociera y validara cada transacción? ¿Qué pasaría si hubiera una manera de tener transacciones escalables fuera de la cadena, sin perder la seguridad de la red Bitcoin?

En febrero de 2015, Joseph Poon y Thaddeus Dryja propusieron una posible solución al problema de escalabilidad de Bitcoin, con la publicación de "The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments". 2

En el documento técnico (ahora desactualizado), Poon y Dryja estiman que para que Bitcoin alcance las 47 000 transacciones por segundo procesadas en su punto máximo por Visa, se necesitarían bloques de 8 GB. Esto haría que ejecutar un nodo fuera completamente insostenible para cualquiera que no sea para empresas a gran escala y operaciones de grado industrial. El resultado sería una red en la que solo unos pocos usuarios podrían validar el estado del libro mayor. Bitcoin depende de los usuarios

validando el libro mayor por sí mismos, sin confiar explícitamente en terceros, para permanecer descentralizados. Poner a los usuarios fuera de los nodos en ejecución obligaría al usuario medio a confiar en terceros para descubrir el estado del libro mayor, rompiendo en última instancia el modelo de confianza de Bitcoin.

Lightning Network propone una nueva red, una segunda capa, donde los usuarios pueden realizar pagos entre pares, sin la necesidad de publicar una transacción en la cadena de bloques de Bitcoin para cada pago.
Los usuarios pueden pagarse entre sí en Lightning Network tantas veces como quieran, sin crear transacciones de Bitcoin adicionales ni incurrir en tarifas en cadena. Solo hacen uso de la cadena de bloques de Bitcoin para cargar bitcoins en Lightning Network inicialmente y para ***liquidar,*** es decir, para eliminar bitcoins de Lightning Network. El resultado es que se pueden realizar muchos más pagos de Bitcoin fuera de la cadena, con solo la carga inicial y las transacciones de liquidación final que necesitan ser validadas y almacenadas por los nodos de Bitcoin. Además de reducir la carga sobre los nodos, los pagos en Lightning Network son más baratos para los usuarios porque no necesitan pagar tarifas de blockchain y más privados para los usuarios porque no se publican para todos los participantes de la red y, además, no se almacenan permanentemente.

Si bien Lightning Network se concibió inicialmente para Bitcoin, se puede implementar en cualquier cadena de bloques que cumpla con algunos requisitos técnicos básicos. Otras cadenas de bloques, como Litecoin, ya son compatibles con Lightning Network. Además, varias otras cadenas de bloques están desarrollando soluciones similares de segunda capa o "capa 2" para ayudarlas a escalar.

Características definitorias de Lightning Network

Lightning Network es una red que opera como un protocolo de segunda capa sobre Bitcoin y otras cadenas de bloques. Lightning Network permite pagos rápidos, seguros, privados, sin confianza y sin permiso. Estas son algunas de las características de Lightning Network:

- Los usuarios de Lightning Network pueden enrutar pagos entre sí a bajo costo y en tiempo real.

- Los usuarios que intercambian valor a través de Lightning Network no necesitan esperar las confirmaciones de bloqueo para los pagos.
- Una vez que se completa un pago en Lightning Network, generalmente dentro de unos segundos, es definitivo y no se puede revertir. Al igual que una transacción de Bitcoin, un pago en Lightning Network solo puede ser reembolsado por el destinatario.
- Mientras que las transacciones de Bitcoin en cadena son transmitidas y verificadas por todos los nodos de la red, los pagos enrutados en Lightning Network se transmiten entre pares de nodos y no son visibles para todos, lo que resulta en una privacidad mucho mayor.
- A diferencia de las transacciones en la red Bitcoin, los pagos enrutados en Lightning Network no necesitan almacenarse permanentemente.
  Por lo tanto, Lightning utiliza menos recursos y, por lo tanto, es más barato. Esta propiedad también tiene beneficios para la privacidad.
- Lightning Network utiliza el enrutamiento de cebolla, similar al protocolo utilizado por la red de privacidad The Onion Router (Tor), de modo que incluso los nodos involucrados en el enrutamiento de un pago solo conocen directamente a su predecesor y sucesor en la ruta de pago.
- Cuando se usa sobre Bitcoin, Lightning Network usa bitcoin real, que siempre está en posesión (custodia) y control total del usuario. Lightning no es una ficha o moneda separada, ***es*** Bitcoin.

Casos de uso de Lightning Network, usuarios y sus historias

Para comprender mejor cómo funciona realmente Lightning Network y por qué la gente lo usa, seguiremos a varios usuarios y sus historias.

En nuestros ejemplos, algunas de las personas ya han usado Bitcoin y otras son completamente nuevas en la red de Bitcoin. Cada persona y su historia, como

enumerados aquí, ilustran uno o más casos de uso específicos. Los revisaremos a lo largo de este libro:

Consumidor

Alice es una usuaria de Bitcoin que desea realizar pagos rápidos, seguros, económicos y privados para pequeñas compras minoristas. Ella compra café con bitcoin, usando Lightning Network.

Comerciante

Bob es dueño de una cafetería, "Bob's Cafe". Los pagos de Bitcoin en cadena no escalan para cantidades pequeñas como una taza de café, por lo que usa Lightning Network para aceptar pagos de Bitcoin casi instantáneamente y por tarifas bajas.

negocio de servicios de software

Chan es un empresario chino que vende servicios de información relacionados con Lightning Network, así como Bitcoin y otras criptomonedas.
Chan vende estos servicios de información a través de Internet mediante la implementación de micropagos a través de Lightning Network.
Además, Chan ha implementado un servicio de proveedor de liquidez que alquila la capacidad del canal entrante en Lightning Network, cobrando una pequeña tarifa de bitcoin por cada período de alquiler.

Jugador

Dina es una jugadora adolescente de Rusia. Juega muchos juegos de computadora diferentes, pero sus favoritos son aquellos que tienen una "economía en el juego" basada en dinero real. Mientras juega, también gana dinero adquiriendo y vendiendo artículos virtuales en el juego. Lightning Network le permite realizar transacciones en pequeñas cantidades por artículos del juego, así como también ganar pequeñas cantidades por completar misiones.

Conclusión

En este capítulo, hablamos sobre el concepto fundamental que subyace tanto en Bitcoin como en Lightning Network: el protocolo de equidad.

Analizamos la historia de Lightning Network y las motivaciones detrás de las soluciones de escalado de segunda capa para Bitcoin y otras redes basadas en blockchain.

Aprendimos terminología básica que incluye nodo, canal de pago, transacciones dentro de la cadena y pagos fuera de la cadena.

Finalmente, conocimos a Alice, Bob, Chan y Dina, a quienes seguiremos a lo largo del resto del libro. En el próximo capítulo, conoceremos a Alice y recorreremos su proceso de pensamiento mientras selecciona una billetera Lightning y se prepara para hacer su primer pago Lightning para comprar una taza de café en Bob's Cafe.

---

1 La entrada de Wikipedia sobre teoría de juegos proporciona más información.

2 Joseph Poon y Thaddeus Dryja. “The Bitcoin Lightning Network: escalable fuera de la cadena Pagos Instantáneos.” BORRADOR Versión 0.5.9.2. 14 de enero de 2016. ***https:// lightning.network/ lightning-network-paper.pdf.***

Capítulo 2. Primeros pasos

En este capítulo, comenzaremos donde la mayoría de las personas comienzan cuando se encuentran con Lightning Network por primera vez: elegir software para participar en la economía de LN. Examinaremos las opciones de dos usuarios que representan un caso de uso común para Lightning Network y aprenderemos con el ejemplo. Alice, una cliente de una cafetería, usará una billetera Lightning en su dispositivo móvil para comprar café en Bob's Cafe. Bob, un comerciante, usará un nodo Lightning y una billetera para ejecutar un sistema de punto de venta en su café, para que pueda aceptar pagos a través de Lightning Network.

La primera billetera Lightning de Alice Alice es una usuaria

de Bitcoin desde hace mucho tiempo. Conocimos a Alice por primera vez en el Capítulo 1 de ***Mastering Bitcoin,*** cuando compró una taza de café en la cafetería de Bob mediante una transacción de Bitcoin. Si aún no está familiarizado con el funcionamiento de las transacciones de Bitcoin o necesita un repaso, lea ***Mastering Bitcoin*** o el resumen en el Apéndice A.

¡Alice se enteró recientemente de que Bob's Cafe acaba de comenzar a aceptar pagos de LN! Alice está ansiosa por aprender y experimentar con Lightning Network; ella quiere ser uno de los primeros clientes de LN de Bob. Para hacer esto, primero, Alice tiene que seleccionar una billetera Lightning que satisfaga sus necesidades.

Alice no quiere confiar la custodia de su bitcoin a terceros. Ha aprendido lo suficiente sobre criptomonedas para saber cómo usar una billetera. También quiere una billetera móvil para poder usarla para pagos pequeños sobre la marcha, por lo que elige la billetera ***Eclair ,*** una billetera móvil Lightning sin custodia popular.
Aprendamos más sobre cómo y por qué tomó estas decisiones.

Nodos de relámpagos

Se accede a Lightning Network a través de aplicaciones de software que pueden hablar el protocolo LN. Un ***nodo Lightning Network*** (nodo LN o simplemente nodo) es una aplicación de software que tiene tres características importantes. Primero, relámpago

los nodos son billeteras, por lo que envían y reciben pagos a través de Lightning Network, así como en la red Bitcoin. En segundo lugar, los nodos deben comunicarse entre pares con otros nodos Lightning que crean la red.
Finalmente, los nodos Lightning también necesitan acceso a la cadena de bloques de Bitcoin (u otras cadenas de bloques para otras criptomonedas) para asegurar los fondos utilizados para los pagos.

Los usuarios tienen el mayor grado de control al ejecutar su propio nodo Bitcoin y nodo Lightning. Sin embargo, los nodos Lightning también pueden usar un cliente ligero de Bitcoin, comúnmente conocido como verificación de pago simplificada (SPV), para interactuar con la cadena de bloques de Bitcoin.

Exploradores relámpago

Los exploradores de LN son herramientas útiles para mostrar las estadísticas de los nodos, los canales y la capacidad de la red.

La siguiente es una lista inexhaustiva:

- Explorador de rayos 1ML
- Explorador Lightning de ACINQ, con visualización elegante
- Explorador de relámpagos espaciales de Amboss, con métricas comunitarias y visualizaciones intuitivas
- El explorador Lightning de Fiatjaf con muchos diagramas
- hashXP Explorador relámpago

PROPINA

Tenga en cuenta que al usar exploradores Lightning, al igual que con otros exploradores de bloques, la privacidad puede ser una preocupación. Si los usuarios son descuidados, el sitio web puede rastrear sus direcciones IP y recopilar sus registros de comportamiento (por ejemplo, los nodos en los que están interesados los usuarios).

Además, se debe tener en cuenta que debido a que no existe un consenso global sobre el gráfico Lightning actual o el estado actual de cualquier política de canal existente, los usuarios nunca deben confiar en los exploradores Lightning para recuperar la información más actualizada. Además, a medida que los usuarios abren, cierran y actualizan canales, el gráfico cambiará y es posible que los exploradores Lightning individuales no estén actualizados. Utilice exploradores Lightning para visualizar la red o recopilar información, pero no como una fuente autorizada de lo que sucede en Lightning Network. Para tener una vista autorizada de Lightning Network, ejecute su propio nodo Lightning que creará un gráfico de canal y recopilará varias estadísticas, que puede ver con una interfaz basada en web.

Carteras relámpago

El término ***billetera Lightning*** es algo ambiguo porque puede describir una amplia variedad de componentes combinados con alguna interfaz de usuario. Los componentes más comunes del software de billetera Lightning incluyen:

- Un almacén de claves que contiene secretos, como claves privadas
- Un nodo LN (nodo Lightning) que se comunica en la red peer-to-peer, como se describió anteriormente
- Un nodo de Bitcoin que almacena datos de blockchain y se comunica con otros nodos de Bitcoin
- Un “mapa” de base de datos de nodos y canales que se anuncian en la Red relámpago
- Un administrador de canales que puede abrir y cerrar canales de LN
- Un sistema de primer plano que puede encontrar una ruta de canales conectados desde el origen del pago hasta el destino del pago

Una billetera Lightning puede contener todas estas funciones, actuando como una billetera "completa", sin depender de ningún servicio de terceros. O uno o más de estos componentes pueden depender (parcial o totalmente) de servicios de terceros que median esas funciones.

Una distinción ***clave*** (juego de palabras intencionado) es si la función de almacén de claves es interna o subcontratada. En las cadenas de bloques, el control de las claves determina la custodia de los fondos, tal como lo recuerda la frase “tus llaves, tus monedas; ni tus llaves, ni tus monedas.” Cualquier billetera que subcontrata la administración de claves se denomina billetera de ***custodia*** porque un tercero que actúa como custodio tiene el control de los fondos del usuario, no el usuario. Una billetera ***sin custodia*** o ***con*** autocustodia, en comparación, es aquella en la que el almacén de claves es parte de la billetera y el usuario controla directamente las claves. El término billetera sin custodia solo implica que el almacén de claves es local y está bajo el control del usuario. Sin embargo, uno o más de los otros componentes de la billetera pueden o no ser subcontratados y depender de terceros confiables.

Las cadenas de bloques, especialmente las cadenas de bloques abiertas como Bitcoin, intentan minimizar o eliminar la confianza en terceros y empoderar a los usuarios. Esto a menudo se denomina modelo "sin confianza", aunque "confianza minimizada" es un término mejor. En dichos sistemas, el usuario confía en las reglas del software, no en terceros. Por lo tanto, la cuestión del control de las claves es una consideración principal al elegir una billetera Lightning.

Todos los demás componentes de una billetera Lightning aportan consideraciones similares de confianza. Si todos los componentes están bajo el control del usuario, la cantidad de confianza en terceros se minimiza, brindando el máximo poder al usuario. Por supuesto, esto conlleva una compensación directa porque con ese poder viene la responsabilidad correspondiente de administrar software complejo.

Cada usuario debe considerar sus propias habilidades técnicas antes de decidir qué tipo de billetera Lightning usar. Aquellos con fuertes habilidades técnicas deben usar una billetera Lightning que pone todos los componentes bajo el control directo del usuario. Aquellos con menos habilidades técnicas, pero con el deseo de controlar sus fondos, deben elegir una billetera Lightning sin custodia. A menudo, la confianza en estos casos se relaciona con la privacidad. Si los usuarios deciden subcontratar alguna funcionalidad a un tercero, generalmente renuncian a cierta privacidad ya que el tercero obtendrá información sobre ellos.

Finalmente, aquellos que buscan simplicidad y conveniencia, incluso a expensas del control y la seguridad, pueden elegir una billetera Lightning de custodia. Esta es la opción técnicamente menos desafiante, pero ***socava el modelo de confianza de la criptomoneda*** y, por lo tanto, debe considerarse solo como un trampolín hacia un mayor control y autosuficiencia.

Hay muchas maneras en que las billeteras se pueden caracterizar o categorizar. Las preguntas más importantes para hacer sobre una billetera específica son:

1. ¿Esta billetera Lightning tiene un nodo Lightning completo o usa un nodo Lightning de terceros?
2. ¿Esta billetera Lightning tiene un nodo Bitcoin completo o usa un nodo de Bitcoin de terceros?
3. ¿Esta billetera Lightning almacena sus propias claves bajo el control del usuario (autocustodia) o las claves están en manos de un custodio externo?

PROPINA

Si una billetera Lightning usa un nodo Lightning de terceros, es este nodo Lightning de terceros el que decide cómo comunicarse con Bitcoin. Por lo tanto, el uso de un nodo Lightning de terceros implica que también está utilizando un nodo Bitcoin de terceros. Solo cuando la billetera Lightning usa su propio nodo Lightning, existe la opción entre el nodo Bitcoin completo y el nodo Bitcoin de terceros.

En el nivel más alto de abstracción, las preguntas 1 y 3 son las más elementales. De estas dos preguntas, podemos derivar cuatro categorías posibles. Podemos colocar estas cuatro categorías en un cuadrante, como se ve en la Tabla 2-1. Pero recuerde que esta es solo una forma de categorizar las billeteras Lightning.

	Nodo Lightning completo	Nodo Lightning de terceros
Auto-custodia	Q1: alta habilidad técnica, menos confianza en terceros, más sin permiso	P2: Habilidades técnicas por debajo del nivel medio, confianza en terceros por debajo del nivel medio, requiere algunos permisos
custodia	P3: por encima de las habilidades técnicas medias, por encima de la confianza media en terceros, requiere algunos permisos	P4: habilidades técnicas bajas, alta confianza en terceros, menos sin permiso

El cuadrante 3 (Q3), donde se usa un nodo Lightning completo, pero las claves están en manos de un custodio, actualmente no es común. Las billeteras futuras de ese cuadrante pueden dejar que un usuario se preocupe por los aspectos operativos de su nodo, pero luego delegar el acceso a las claves a un tercero que utiliza principalmente almacenamiento en frío.

Las billeteras Lightning se pueden instalar en una variedad de dispositivos, incluidas computadoras portátiles, servidores y dispositivos móviles. Para ejecutar un nodo Lightning completo, deberá usar un servidor o una computadora de escritorio, ya que los dispositivos móviles y las computadoras portátiles generalmente no son lo suficientemente potentes en términos de capacidad, procesamiento, duración de la batería y conectividad.

La categoría de nodos Lightning de terceros se puede volver a subdividir:

Ligero

> Esto significa que la billetera no opera un nodo Lightning y, por lo tanto, necesita obtener información sobre Lightning Network a través de Internet desde el nodo Lightning de otra persona.

Ninguna

> Esto significa que no solo el nodo Lightning es operado por un tercero, sino que la mayor parte de la billetera es operada por un tercero en la nube. Esta es una billetera de custodia donde alguien más controla la custodia de los fondos.

Estas subcategorías se utilizan en la Tabla 2-2.

Otros términos que necesitan explicación en la Tabla 2-2 en la columna "Nodo Bitcoin" son:

neutrino

Esta billetera no opera un nodo Bitcoin. En cambio, se accede a un nodo de Bitcoin operado por otra persona (un tercero) a través del Protocolo Neutrino.

electro

Esta billetera no opera un nodo Bitcoin. En cambio, se accede a un nodo de Bitcoin operado por otra persona (un tercero) a través del Protocolo Electrum.

Núcleo de Bitcoin

Esta es una implementación de un nodo Bitcoin.

btcd

Esta es otra implementación de un nodo Bitcoin.

En la Tabla 2-2, vemos algunos ejemplos de aplicaciones de billetera y nodo Lightning actualmente populares para diferentes tipos de dispositivos. La lista está ordenada primero por tipo de dispositivo y luego alfabéticamente.

Tabla 2.2. Ejemplos de Lightni populares

norte

gramo

en

a

yo

yo

y

t

s

Solicitud	Dispositivo	Nodo relámpago	Nodo Bitcoin	Almacén de claves
Cartera Azul	Móvil	Ninguna	Ninguna	custodia
Cartera Breez	Móvil	nodo completo	neutrino	Auto-custodia
Rayo móvil	Móvil	Ligero	electro	Auto-custodia
lntxbot	Móvil	Ninguna	Ninguna	custodia
Otro	Móvil	Ligero	neutrino	Auto-custodia
Monedero Fénix	Móvil	Ligero	electro	Auto-custodia
Zeus	Móvil	nodo completo	Autocustodia de Bitcoin Core/btcd	
electro	Escritorio	nodo completo	Bitcoin Núcleo/Electrónico	Auto-custodia
Zap escritorio	Escritorio	nodo completo	neutrino	Auto-custodia
c-relámpago	Servidor	nodo completo	Núcleo de Bitcoin	Auto-custodia
Servidor Eclair	Servidor	nodo completo	Bitcoin Núcleo/Electrónico	Auto-custodia
lnd	Servidor	nodo completo	Autocustodia de Bitcoin Core/btcd	

Bitcoin de la red de prueba

El sistema Bitcoin ofrece una cadena alternativa para propósitos de prueba llamada ***testnet,*** en contraste con la cadena Bitcoin "normal" que se conoce como ***red principal.*** En testnet, la moneda es ***testnet bitcoin (tBTC),*** que es una copia sin valor de bitcoin utilizada exclusivamente para pruebas. Cada función de Bitcoin se replica exactamente, pero el dinero no vale nada, ¡así que literalmente no tienes nada que perder!

Algunas billeteras Lightning también pueden operar en testnet, lo que le permite realizar pagos Lightning con testnet bitcoin, sin arriesgar fondos reales. Esta es una gran manera de experimentar con Lightning de forma segura. Eclair Mobile, que Alice usa en este capítulo, es un ejemplo de una billetera Lightning que admite la operación de testnet.

Puede obtener algunos tBTC para jugar desde un grifo de ***bitcoin testnet,*** que ofrece tBTC gratis a pedido. Aquí hay algunos grifos de testnet:

https:// coinfaucet.eu/ en/ btc-testnet

https:// testnet-faucet.mempool.co

https:// bitcoinfaucet.uo1.net

https:// testnet.help/ en/ btcfaucet/ testnet

Todos los ejemplos de este libro se pueden replicar exactamente en testnet con tBTC, por lo que puede seguirlos si lo desea sin arriesgar dinero real.

Complejidad de equilibrio y control

Las billeteras Lightning deben lograr un cuidadoso equilibrio entre la complejidad y el control del usuario. Aquellos que dan al usuario el mayor control sobre sus fondos, el mayor grado de privacidad y la mayor independencia de los servicios de terceros son necesariamente más complejos y difíciles de operar. A medida que avanza la tecnología, algunas de estas ventajas y desventajas serán menos estrictas y los usuarios podrán obtener más control sin mayor complejidad. Sin embargo, por ahora, diferentes empresas y proyectos están explorando diferentes posiciones a lo largo de este espectro de complejidad de control, con la esperanza de encontrar el "punto óptimo" para los usuarios a los que se dirigen.

Al seleccionar una billetera, tenga en cuenta que incluso si no ve estas compensaciones, todavía existen. Por ejemplo, muchas billeteras intentarán eliminar la carga de la gestión de canales de sus usuarios. Para hacerlo, introducen ***nodos*** centrales a los que todas sus billeteras se conectan automáticamente. Si bien esta compensación simplifica la interfaz de usuario y la experiencia del usuario, presenta un punto único de falla (SPoF) ya que estos nodos centrales se vuelven indispensables para la operación de la billetera. Además, confiar en un "hub" como este puede reducir la privacidad del usuario ya que el hub conoce al remitente y potencialmente (si construye la ruta de pago en nombre del usuario) también al destinatario de cada pago realizado por la billetera del usuario.

En la siguiente sección, volveremos a nuestro primer usuario y recorreremos su primera configuración de billetera Lightning. Ha elegido una billetera que es más sofisticada que las billeteras de custodia más fáciles. Esto nos permite mostrar algo de la complejidad subyacente e introducir algunos de los mecanismos internos de una billetera avanzada. Puede encontrar que su primera billetera ideal está orientada hacia la facilidad de uso, aceptando algunas de las compensaciones de control y privacidad. O tal vez sea más un usuario avanzado y quiera ejecutar sus propios nodos Lightning y Bitcoin como parte de su solución de billetera.

Descarga e instalación de una billetera Lightning

Cuando busque una nueva billetera de criptomonedas, debe tener mucho cuidado de seleccionar una fuente segura para el software.

Desafortunadamente, muchas aplicaciones de billetera falsas robarán su dinero, y algunas de ellas incluso llegan a sitios de software acreditados y supuestamente examinados, como las tiendas de aplicaciones de Apple y Google. Ya sea que esté instalando su primera o su décima billetera, siempre tenga mucho cuidado. Es posible que una aplicación no autorizada no solo robe el dinero que le confíes, sino que también podría robar claves y contraseñas de otras aplicaciones al comprometer el sistema operativo de tu dispositivo móvil.

Alice usa un dispositivo Android y usará Google Play Store para descargar e instalar la billetera Eclair. Al buscar en Google Play, encuentra una entrada para "Eclair Mobile", como se muestra en la Figura 2-1.

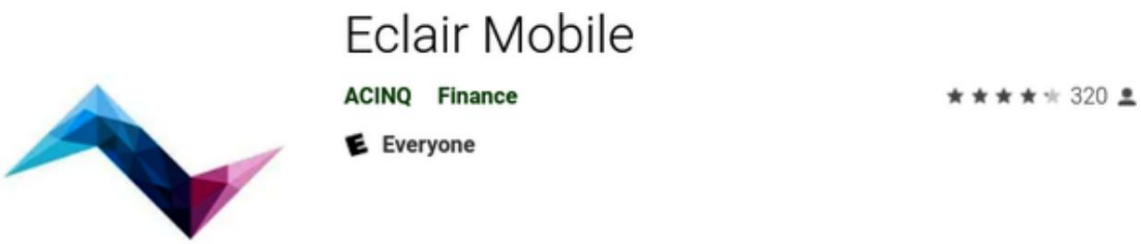


Figura 2-1. Eclair Mobile en Google Play Store

PROPINA

Es posible experimentar y probar todo el software de tipo Bitcoin sin riesgo (excepto por su propio tiempo) utilizando testnet bitcoins. También puede descargar la billetera de red de prueba de Eclair para probar Lightning (en la red de prueba) yendo a Google Play Store.

Alice nota algunos elementos diferentes en esta página que la ayudan a determinar que esta es, muy probablemente, la billetera "Eclair Mobile" correcta que está buscando. En primer lugar, la organización[2] ACINQ aparece como el desarrollador de esta billetera móvil, que Alice sabe por su investigación que es el desarrollador correcto. En segundo lugar, la billetera se ha instalado "más de 10 000" veces y tiene más de 320 reseñas positivas. Es poco probable que se trate de una aplicación no autorizada que se haya colado en Google Play Store. Como tercer paso, accede al sitio web de ACINQ. Ella verifica que la página web es segura comprobando que la dirección comienza con https, o con el prefijo de un candado en algunos navegadores. En el sitio web, va a la sección de descargas o busca el enlace a la tienda de aplicaciones de Google. Encuentra el enlace y hace clic en él. Ella compara que este enlace la lleva a la misma aplicación en Google App Store. Satisfecha con estos hallazgos, Alice instala la aplicación Eclair en su dispositivo móvil.

ADVERTENCIA

Tenga siempre mucho cuidado al instalar software en cualquier dispositivo. Hay muchas billeteras de criptomonedas falsas que no solo robarán su dinero sino que también podrían comprometer todas las demás aplicaciones en su dispositivo.

Crear una nueva billetera

Cuando Alice abre la aplicación Eclair Mobile por primera vez, se le presenta la opción de "Crear una nueva billetera" o "Importar una billetera existente".
Alice creará una nueva billetera, pero primero analicemos por qué se presentan estas opciones aquí y qué significa importar una billetera existente.

Responsabilidad con la Custodia de Claves

Como mencionamos al comienzo de esta sección, Eclair es una billetera ***sin custodia ,*** lo que significa que Alice tiene la custodia exclusiva de las claves utilizadas para controlar su bitcoin. Esto también significa que Alice es responsable de proteger y respaldar esas claves. Si Alice pierde las llaves, nadie podrá ayudarla a recuperar los bitcoins y se perderán para siempre.

> **ADVERTENCIA**
>
> Con la billetera Eclair Mobile, Alice tiene la custodia y el control de las claves y, por lo tanto, la responsabilidad total de mantener las claves seguras y respaldadas. Si pierde las llaves, pierde el bitcoin, ¡y nadie puede ayudarla a recuperarse de esa pérdida!

Palabras mnemotécnicas

Al igual que la mayoría de las billeteras de Bitcoin, Eclair Mobile proporciona una ***frase mnemotécnica*** (a veces también llamada "semilla" o "frase semilla") para que Alice realice una copia de seguridad. La frase mnemotécnica consta de 24 palabras en inglés, seleccionadas al azar por el software y utilizadas como base para las claves que genera la billetera. Alice puede usar la frase mnemotécnica para restaurar todas las transacciones y fondos en la billetera Eclair Mobile en caso de pérdida de un dispositivo móvil, un error de software o corrupción de la memoria.

> PROPINA
>
> El término correcto para estas palabras de respaldo es "frase mnemotécnica". Evitamos el uso del término “semilla” para referirnos a una frase mnemotécnica porque aunque su uso es común, es incorrecto.

Cuando Alice elige crear una nueva billetera, verá una pantalla con su frase mnemotécnica, que se parece a la captura de pantalla de la Figura 2-2.

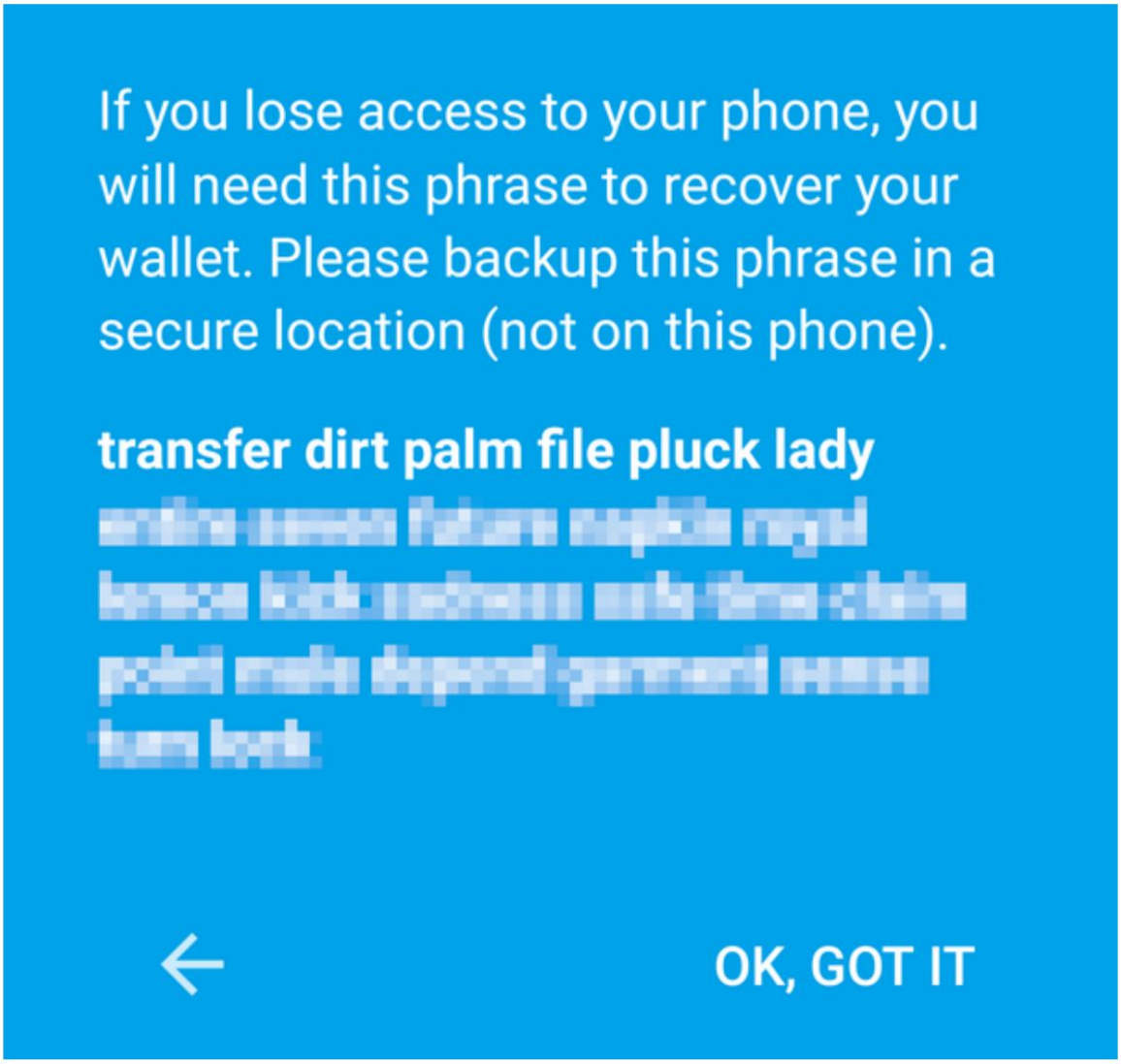


Figura 2-2. Nueva frase mnemotécnica de billetera

En la Figura 2-2, hemos oscurecido deliberadamente parte de la frase mnemotécnica para evitar que los lectores de este libro la reutilicen.

Almacenar el mnemotécnico de forma segura

Alice debe tener cuidado de almacenar la frase mnemotécnica de una manera que evite el robo pero también la pérdida accidental. La forma recomendada de equilibrar adecuadamente estos riesgos es escribir dos copias de la frase mnemotécnica en papel, con cada una de las palabras numeradas; el orden es importante.

Una vez que Alice haya grabado la frase mnemotécnica, después de tocar "OK GOT IT" en su pantalla, se le presentará una prueba para asegurarse de que lo haya hecho correctamente.

grabó la mnemotécnica. El cuestionario le pedirá tres o cuatro de las palabras al azar. Alice no esperaba un examen, pero como registró la mnemotécnica correctamente, pasa sin ninguna dificultad.

Una vez que Alice haya grabado la frase mnemotécnica y haya pasado el cuestionario, debe guardar cada copia en un lugar seguro separado, como un cajón de escritorio cerrado con llave o una caja fuerte a prueba de fuego.

ADVERTENCIA

Nunca intente un esquema de seguridad "hágalo usted mismo" que se desvíe de alguna manera de la recomendación de mejores prácticas en "Almacenamiento seguro del mnemotécnico". No corte su mnemónico a la mitad, no haga capturas de pantalla, no lo almacene en unidades USB o unidades en la nube, no lo cifre ni intente ningún otro método no estándar. Inclinarás la balanza de tal manera que te arriesgues a una pérdida permanente. Muchas personas han perdido fondos, no por robo, sino porque probaron una solución no estándar sin tener la experiencia para equilibrar los riesgos involucrados. La recomendación de mejores prácticas es considerada cuidadosamente por expertos y adecuada para la gran mayoría de los usuarios.

Después de que Alice inicialice su billetera Eclair Mobile, verá un breve tutorial que destaca los diversos elementos de la interfaz de usuario. No replicaremos el tutorial aquí, pero exploraremos todos esos elementos mientras seguimos el intento de Alice de comprar una taza de café.

Cargar Bitcoin en la billetera Alice ahora tiene una billetera

Lightning. ¡Pero está vacío! Ahora se enfrenta a uno de los aspectos más desafiantes de este experimento: tiene que encontrar una forma de adquirir bitcoins y cargarlos en su billetera Eclair.

PROPINA

Si Alice ya tiene bitcoins en otra billetera, podría optar por enviar esos bitcoins a su billetera Eclair en lugar de adquirir nuevos bitcoins para cargarlos en su nueva billetera.

adquirir bitcoins

Hay varias formas en que Alice puede adquirir bitcoins:

- Puede cambiar parte de su moneda nacional (por ejemplo, USD) en un intercambio de criptomonedas.
- Puede comprarle algo a un amigo o a un conocido de una reunión de Bitcoin a cambio de dinero en efectivo.
- Puede encontrar un ***cajero automático de Bitcoin*** en su área, que actúa como una máquina expendedora, vendiendo bitcoins por dinero en efectivo.
- Puede ofrecer sus habilidades o un producto que vende y aceptar pagos en bitcoin.
- Puede pedirle a su empleador o a sus clientes que le paguen en bitcoins.

Todos estos métodos tienen diversos grados de dificultad y muchos implicarán el pago de una tarifa. Algunos también requerirán que Alice proporcione documentos de identificación para cumplir con las regulaciones bancarias locales. Sin embargo, con todos estos métodos, Alice podrá recibir bitcoins.

Recibir Bitcoin Supongamos

que Alice ha encontrado un cajero automático local de Bitcoin y ha decidido comprar algo de Bitcoin a cambio de dinero en efectivo. En la Figura 2-3 se muestra un ejemplo de un cajero automático de Bitcoin, construido por Lamassu Company . Dichos cajeros automáticos de Bitcoin aceptan moneda nacional (efectivo) a través de una ranura de efectivo y envían bitcoin a una dirección de Bitcoin escaneada desde la billetera de un usuario usando una cámara incorporada.

Figura 2-3. Un cajero automático Lamassu Bitcoin

Para recibir el bitcoin en su billetera Eclair Lightning, Alice deberá presentar una dirección de Bitcoin de la billetera Eclair Lightning al cajero automático. El cajero automático puede enviar el bitcoin recién adquirido de Alice a esta dirección de Bitcoin.

Para ver una dirección de Bitcoin en la billetera Eclair, Alice debe deslizar el dedo hacia la columna de la izquierda titulada SU DIRECCIÓN DE BITCOIN (consulte la Figura 2-4), donde verá un código de barras cuadrado (llamado ***código QR)*** y una cadena de letras y números a continuación. que.

El código QR contiene la misma cadena de letras y números que se muestra debajo, en un formato fácil de escanear. De esta forma, Alice no tiene que escribir la dirección de Bitcoin. En la captura de pantalla (Figura 2-4), hemos difuminado ambos deliberadamente para evitar que los lectores envíen bitcoins sin darse cuenta a esta dirección.

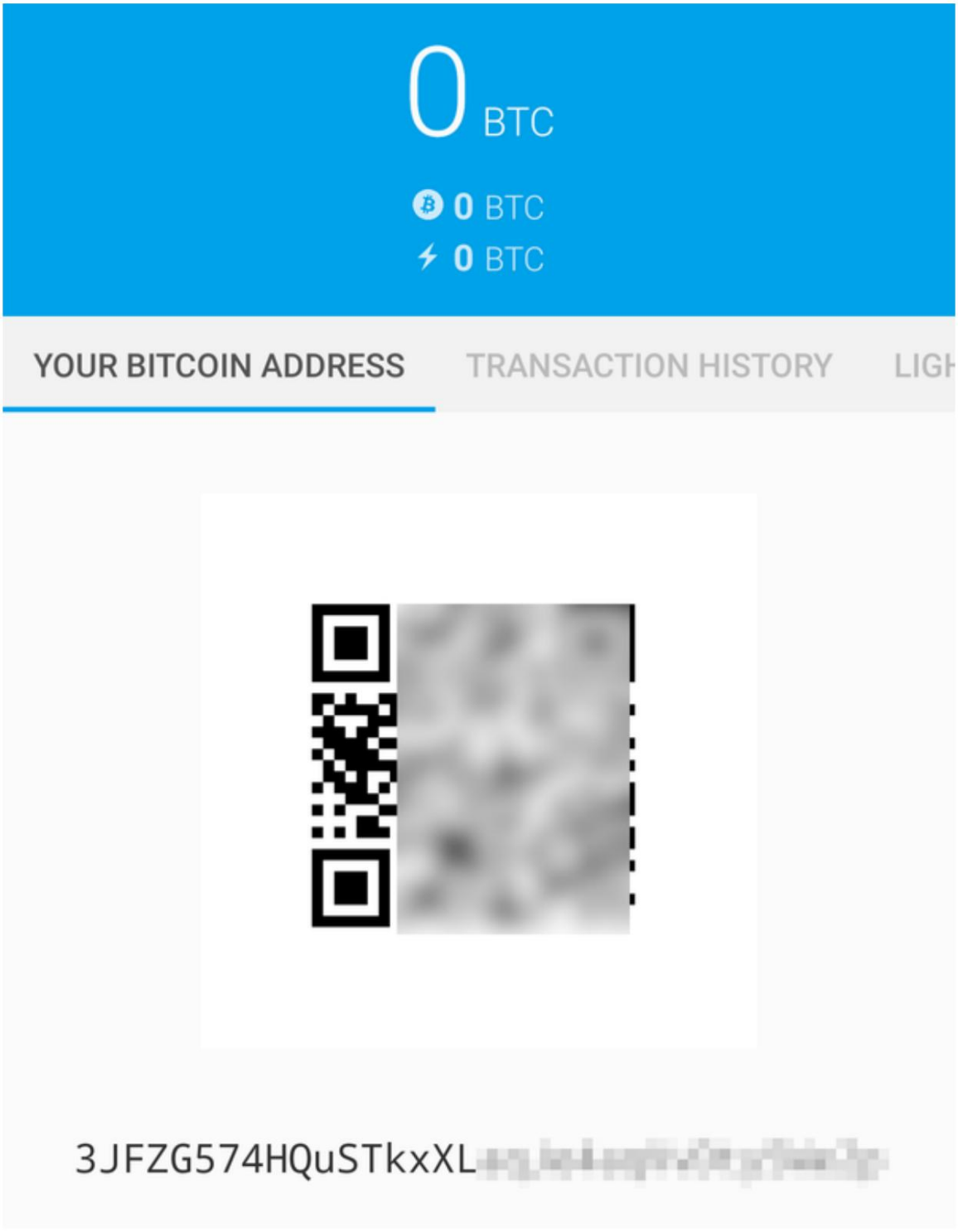


Figura 2-4. La dirección de bitcoin de Alice, que se muestra en Eclair

> **NOTA**
>
> Tanto las direcciones de Bitcoin como los códigos QR contienen información de detección de errores que evita que cualquier error de escritura o escaneo produzca una dirección de Bitcoin "incorrecta". Si hay un error en la dirección, cualquier billetera de Bitcoin notará el error y se negará a aceptar la dirección de Bitcoin como válida.

Alice puede llevar su dispositivo móvil al cajero automático y mostrárselo a la cámara integrada, como se muestra en la Figura 2-5. ¡Después de insertar algo de efectivo en la ranura, recibirá bitcoin en Eclair!

Figura 2-5. Bitcoin ATM escanea el código QR

Alice verá la transacción del cajero automático en la pestaña HISTORIAL DE TRANSACCIONES de la billetera Eclair. Aunque Eclair detectará la transacción de bitcoin en solo unos segundos, la transacción de bitcoin tardará aproximadamente una hora en "confirmarse" en la cadena de bloques de Bitcoin. Como puede ver en la Figura 2-6, la billetera Eclair de Alice muestra "6+ conf" debajo de la transacción, lo que indica que la transacción recibió el mínimo requerido de seis confirmaciones y que sus fondos ahora están listos para usar.

> PROPINA
>
> El número de confirmaciones en una transacción es el número de bloques extraídos desde (e inclusive) el bloque que contenía esa transacción. Seis confirmaciones es la mejor práctica, pero diferentes billeteras Lightning pueden considerar un canal abierto después de cualquier cantidad de confirmaciones. Algunas billeteras incluso aumentan la cantidad de confirmaciones esperadas por el valor monetario del canal.

Aunque en este ejemplo, Alice usó un cajero automático para adquirir su primer bitcoin, se aplicarían los mismos conceptos básicos incluso si usara uno de los otros métodos en "Adquirir Bitcoin". Por ejemplo, si Alice quisiera vender un producto o brindar un servicio profesional a cambio de bitcoin, sus clientes podrían escanear la dirección de Bitcoin con sus billeteras y pagarle en bitcoin.

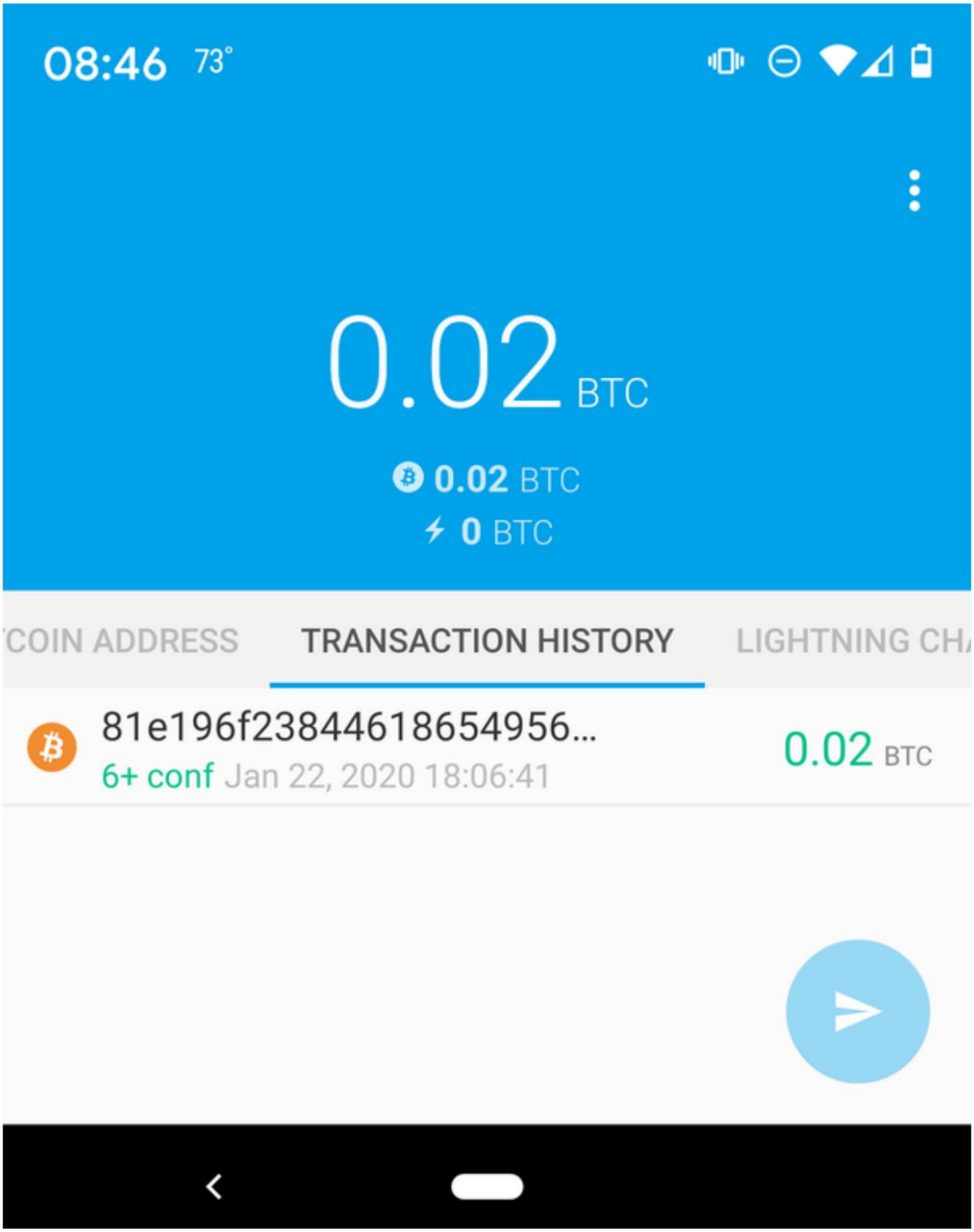


Figura 2-6. Alicia recibe bitcoin

De manera similar, si le facturó a un cliente por un servicio ofrecido a través de Internet, Alice podría enviar un correo electrónico o un mensaje instantáneo con la dirección de Bitcoin o el código QR a su cliente, y ellos podrían pegar o escanear la información en una billetera de Bitcoin para pagarle. .

Alice podría incluso imprimir el código QR y pegarlo en un letrero y mostrarlo públicamente para recibir consejos. Por ejemplo, podría colocar un código QR en su guitarra y recibir propinas mientras toca en la calle. [3]

Finalmente, si Alice compró bitcoins en un intercambio de criptomonedas, podría (y debería) "retirar" el bitcoin pegando su dirección de Bitcoin en el sitio web del intercambio. El intercambio luego enviará el bitcoin a su dirección directamente.

De Bitcoin a Lightning Network El bitcoin de Alice ahora está

controlado por su billetera Eclair y se ha registrado en la cadena de bloques de Bitcoin. En este punto, el bitcoin de Alice está ***en la cadena,*** lo que significa que la transacción se ha transmitido a toda la red de Bitcoin, verificado por todos los nodos de Bitcoin y ***extraído*** (registrado) en la cadena de bloques de Bitcoin.

Hasta ahora, la billetera Eclair Mobile se ha comportado solo como una billetera Bitcoin y Alice no ha utilizado las funciones de Lightning Network de Eclair. Como es el caso con muchas billeteras Lightning, Eclair une Bitcoin y Lightning Network al actuar como una billetera Bitcoin y una billetera Lightning.

Ahora, Alice está lista para comenzar a usar Lightning Network sacando su bitcoin de la cadena para aprovechar los pagos rápidos, económicos y privados que ofrece Lightning Network.

Canales de la red Lightning

Al deslizar hacia la derecha, Alice accede a la sección LIGHTNING CHANNELS de Eclair. Aquí puede administrar los canales que conectarán su billetera a Lightning Network.

Revisemos la definición de un canal LN en este punto, para aclarar un poco las cosas. En primer lugar, la palabra "canal" es una metáfora de una ***relación financiera*** entre la billetera Lightning de Alice y otra billetera Lightning. Lo llamamos canal porque es un medio para que la billetera de Alice y esta otra billetera intercambien muchos pagos

entre sí en Lightning Network (fuera de la cadena) sin comprometer transacciones en la cadena de bloques de Bitcoin (en la cadena).

La billetera o el ***nodo*** al que Alice abre un canal se llama su ***compañero de canal.*** Una vez "abierto", un canal se puede usar para enviar muchos pagos entre la billetera de Alice y su compañero de canal.

Además, el compañero de canal de Alice puede ***reenviar*** pagos a través de otros canales más hacia Lightning Network. De esta manera, Alice puede ***enrutar*** un pago a cualquier billetera (por ejemplo, la billetera Lightning de Bob) siempre que la billetera de Alice pueda encontrar una ***ruta*** viable saltando de un canal a otro, hasta llegar a la billetera de Bob.

PROPINA

No todos los pares de canal son ***buenos*** pares para enrutar pagos. Los pares bien conectados podrán enrutar los pagos a través de rutas más cortas hasta el destino, lo que aumenta las posibilidades de éxito. Los compañeros de canal con amplios fondos podrán enrutar pagos más grandes.

En otras palabras, Alice necesita uno o más canales que la conecten a uno o más nodos en Lightning Network. No necesita un canal para conectar su billetera directamente a Bob's Cafe para enviarle un pago a Bob, aunque también puede optar por abrir un canal directo. Cualquier nodo en Lightning Network se puede usar para el primer canal de Alice. Cuanto mejor conectado esté un nodo

es decir, a más personas puede llegar Alice. En este ejemplo, dado que también queremos demostrar el enrutamiento de pagos, no haremos que Alice abra un canal directamente a la billetera de Bob. En cambio, haremos que Alice abra un canal a un nodo bien conectado y luego use ese nodo para reenviar su pago, enrutándolo a través de cualquier otro nodo según sea necesario para llegar a Bob.

Al principio, no hay canales abiertos, así que como vemos en la Figura 2-7, la pestaña CANALES DE RAYOS muestra una lista vacía. Si te fijas, en la esquina inferior derecha hay un símbolo más (+), que es un botón para abrir un nuevo canal.

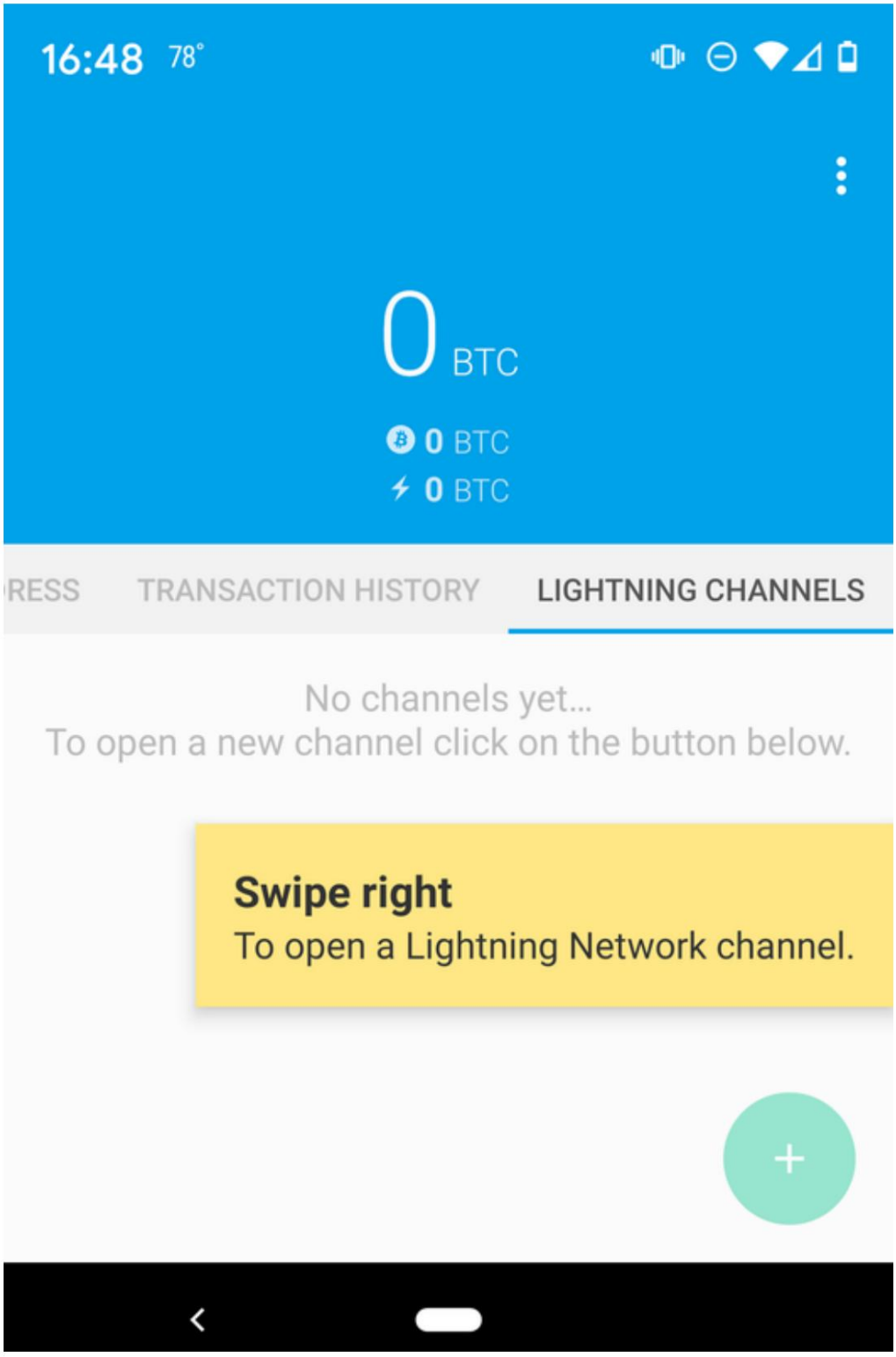


Figura 2-7. Pestaña CANALES DE RAYOS

Alice presiona el símbolo más y se le presentan cuatro formas posibles de abrir un canal:

- Pegar un URI de nodo
- Escanear un URI de nodo
- Nodo aleatorio
- nodo ACINQ

Un "URI de nodo" es un identificador de recursos universal (URI) que identifica un nodo Lightning específico. Alice puede pegar dicho URI desde su portapapeles o escanear un código QR que contenga la misma información. Un ejemplo de un URI de nodo se muestra como un código QR en la Figura 2-8 y luego como una cadena de texto.

Figura 2-8. URI de nodo como código QR

0237fefbe8626bf888de0cad8c73630e32746a22a2c4faa91c1d9877a3826e1174@1
.ln.aantonop.com:9735

Si bien Alice podría seleccionar un nodo Lightning específico, o usar la opción "Nodo aleatorio" para que la billetera Eclair seleccione un nodo al azar, seleccionará la opción Nodo ACINQ para conectarse a uno de los nodos Lightning bien conectados de ACINQ.

Elegir el nodo ACINQ reducirá levemente la privacidad de Alice, porque le dará a ACINQ la capacidad de ver todas las transacciones de Alice. También creará un único punto de falla, ya que Alice solo tendrá un canal, y si el nodo ACINQ no está disponible, Alice no podrá realizar pagos. Para simplificar las cosas al principio, aceptaremos estas compensaciones. ¡En los capítulos siguientes, aprenderemos gradualmente cómo ganar más independencia y hacer menos concesiones!

Alice selecciona el nodo ACINQ y está lista para abrir su primer canal en Lightning Network.

Apertura de un canal Lightning

Cuando Alice selecciona un nodo para abrir un nuevo canal, se le pide que seleccione cuánto bitcoin quiere asignar a este canal. En capítulos posteriores, discutiremos las implicaciones de estas elecciones, pero por ahora, Alice asignará casi todos sus fondos al canal. Dado que tendrá que pagar tarifas de transacción para abrir el canal, seleccionará una cantidad ligeramente menor que su saldo total.
4

Alice asigna 0,018 BTC de su total de 0,020 BTC a su canal y acepta la tasa de tarifa predeterminada, como se muestra en la Figura 2-9.

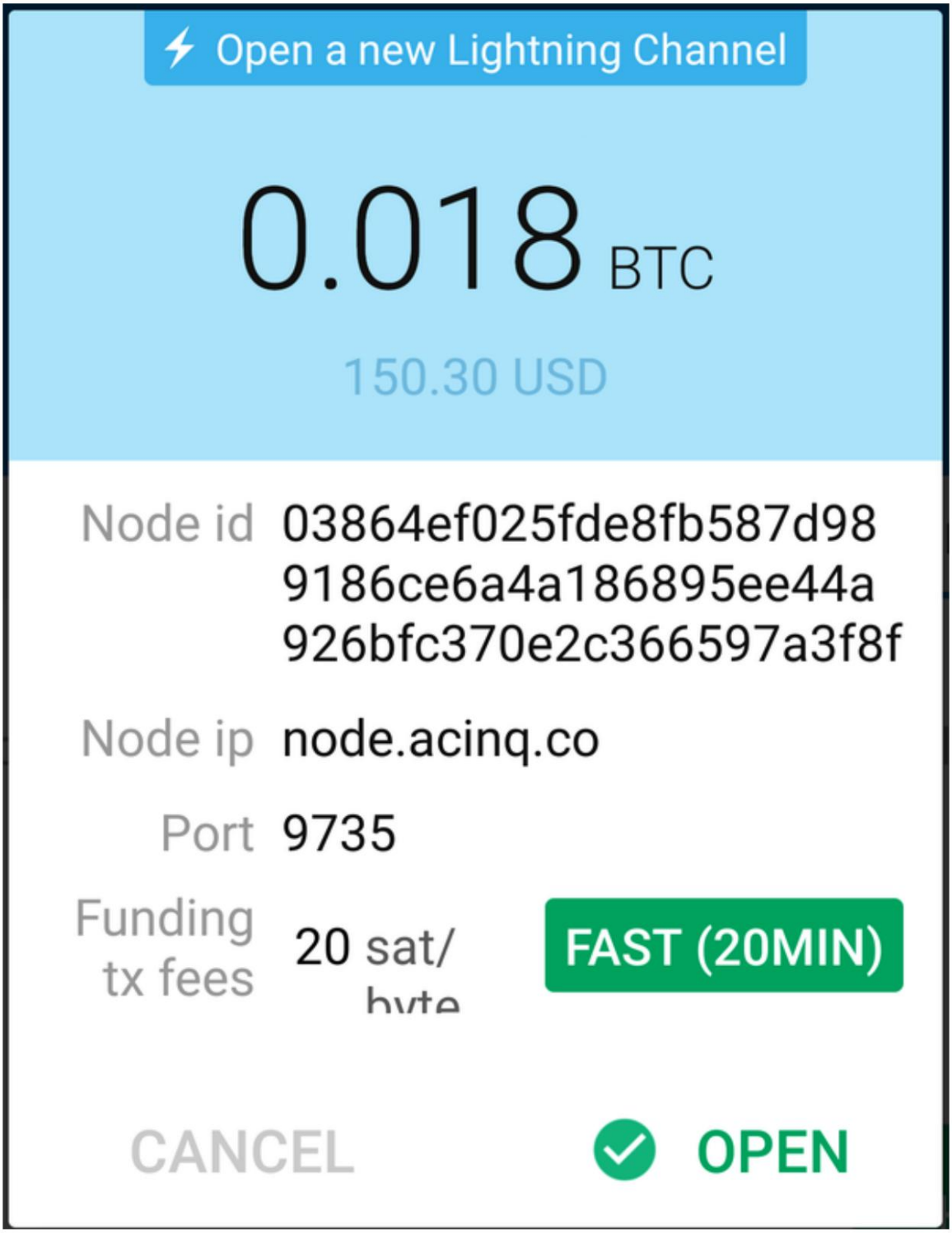


Figura 2-9. Abriendo un canal Lightning

Una vez que hace clic en ABRIR, su billetera construye la transacción especial de Bitcoin que abre un canal Lightning, conocido como la ***transacción de financiación.*** La transacción de financiación en cadena se envía a la red Bitcoin para su confirmación.

Alice ahora tiene que esperar nuevamente (consulte la Figura 2-10) para que la transacción se registre en la cadena de bloques de Bitcoin. Al igual que con la transacción inicial de Bitcoin que usó para adquirir su bitcoin, tiene que esperar seis o más confirmaciones (aproximadamente una hora).

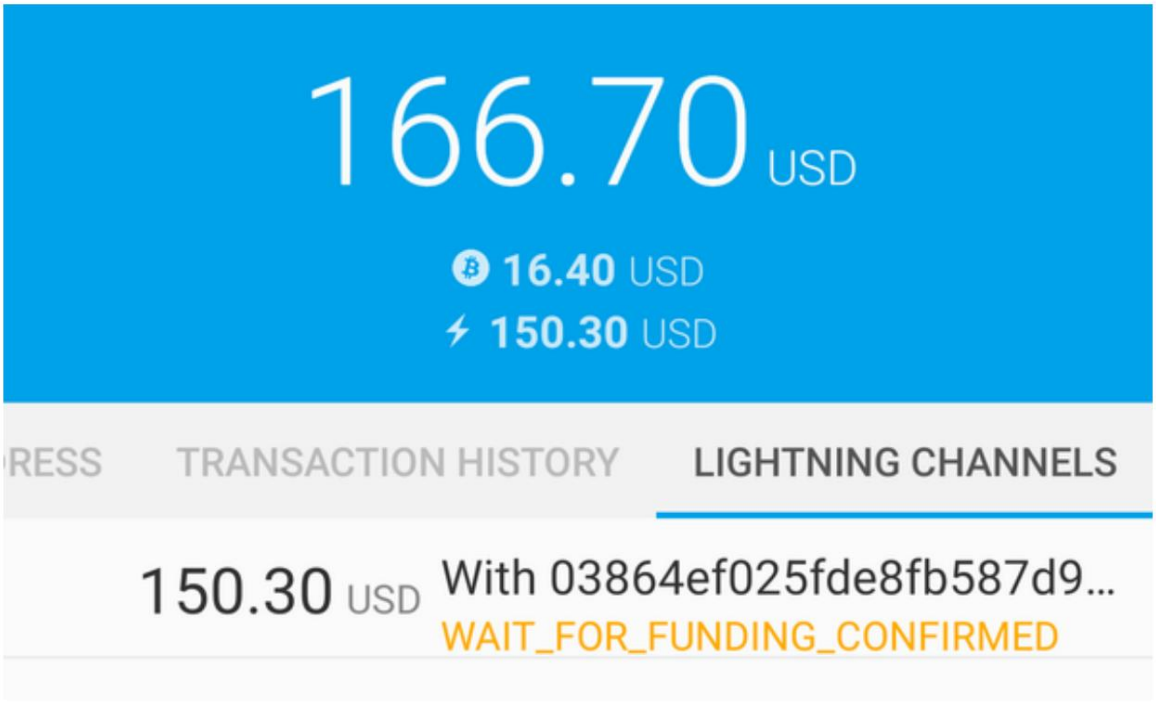


Figura 2-10. Esperando la transacción de financiación para abrir el canal

Una vez que se confirma la transacción de financiamiento, el canal de Alice al nodo ACINQ está abierto, financiado y listo, como se muestra en la Figura 2-11.

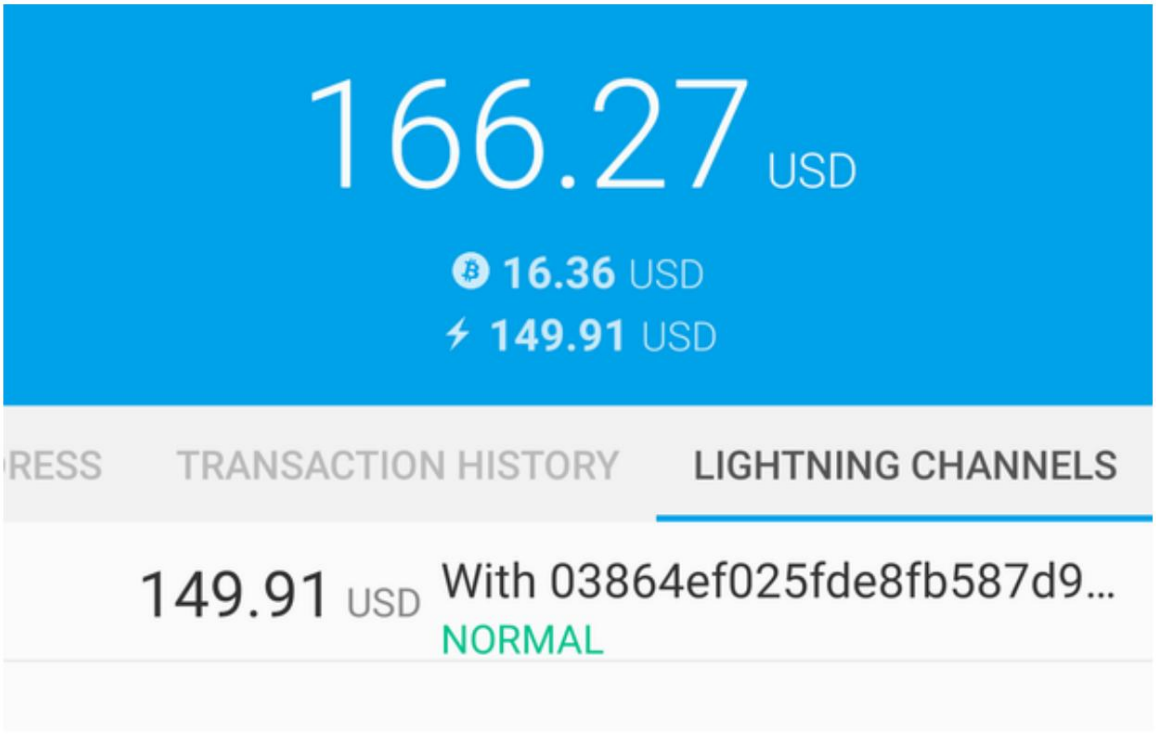


Figura 2-11. el canal esta abierto

PROPINA

¿Notaste que la cantidad de canales parece haber cambiado? No lo ha hecho: el canal contiene 0.018 BTC, pero en el tiempo entre capturas de pantalla, el tipo de cambio de BTC cambió, por lo que el valor en USD es diferente. Puede optar por mostrar los saldos en BTC o USD, ¡pero tenga en cuenta que los valores en USD se calculan en tiempo real y cambiarán!

Comprar una taza de café usando el rayo La red

Alice ahora tiene todo listo para comenzar a usar Lightning Network. Como puede ver, tomó un poco de trabajo y un poco de tiempo esperando las confirmaciones. Sin embargo, ahora las acciones posteriores son rápidas y sencillas. Lightning Network permite pagos sin tener que esperar confirmaciones, ya que los fondos se liquidan en segundos.

Alice toma su dispositivo móvil y corre a Bob's Cafe en su vecindario. ¡Está emocionada de probar su nueva billetera Lightning y usarla para comprar algo!

café de bob

Bob tiene una aplicación de punto de venta (PoS) simple para el uso de cualquier cliente que quiera pagar con bitcoin a través de Lightning Network. Como veremos en el siguiente capítulo, Bob utiliza la popular plataforma de código abierto ***BTCPay Server*** que contiene todos los componentes necesarios para una solución de comercio electrónico o minorista, tales como:

- Un nodo Bitcoin usando el software Bitcoin Core
- Un nodo Lightning que utiliza el software c-lightning
- Una aplicación PoS simple para una tableta

BTCPay Server simplifica la instalación de todo el software necesario, la carga de imágenes y precios de productos y el lanzamiento de una tienda rápidamente.

En el mostrador de Bob's Cafe, hay una tableta que muestra lo que ve en la Figura 2-12.

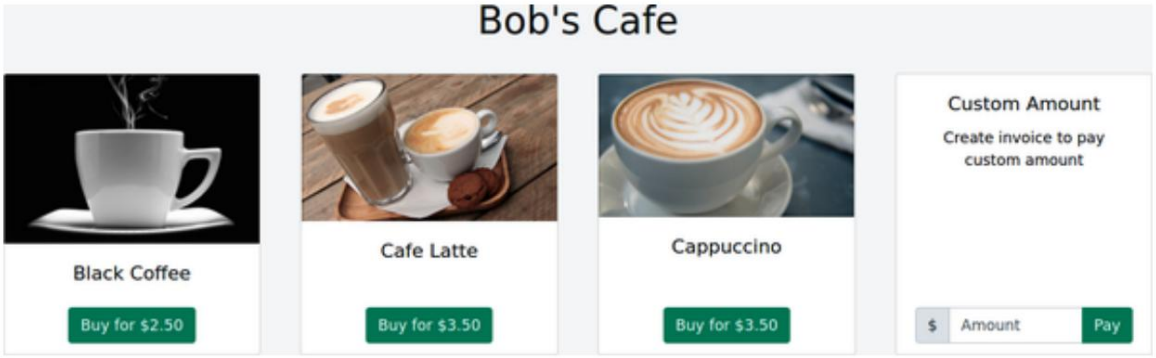


Figura 2-12. Aplicación de punto de venta de Bob

Una factura relámpago

Alice selecciona la opción Café Latte de la pantalla y se le presenta una ***factura Lightning*** (también conocida como "solicitud de pago"), como se muestra en la Figura 2-13.

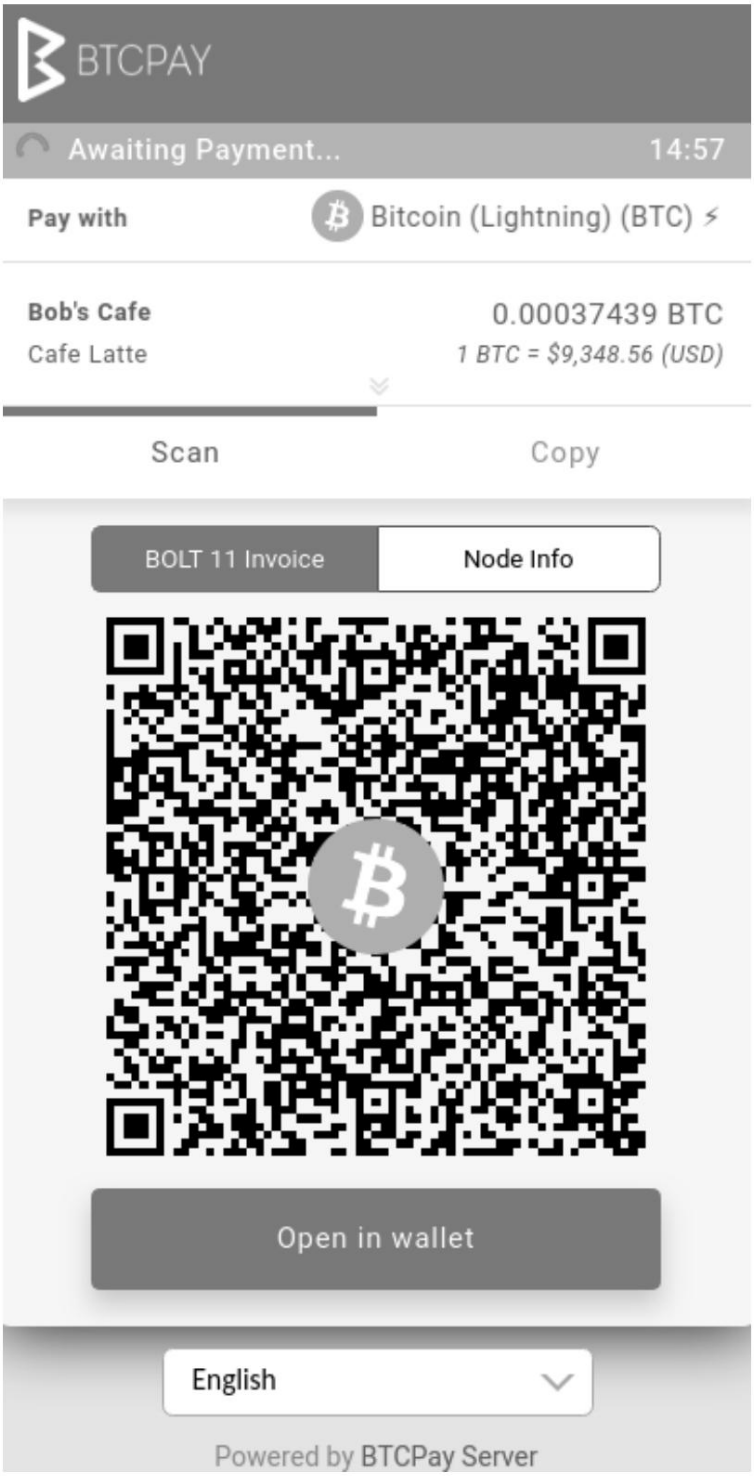


Figura 2-13. Factura relámpago por el café con leche de Alice

Para pagar la factura, Alice abre su billetera Eclair y selecciona el botón Enviar (que parece una flecha hacia arriba) en la pestaña HISTORIAL DE TRANSACCIONES, como se muestra en la Figura 2-14.

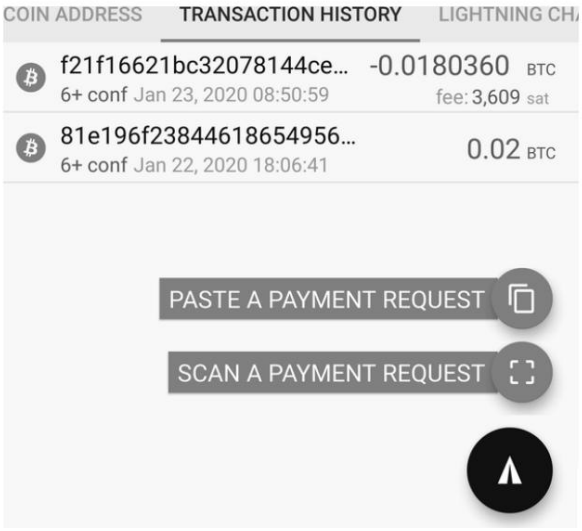


Figura 2-14. Alicia seleccionando Enviar

PROPINA

El término "solicitud de pago" puede referirse a una solicitud de pago de Bitcoin o una factura Lightning, y los términos "factura" y "solicitud de pago" a menudo se usan indistintamente. El término técnico correcto es "factura relámpago", independientemente de cómo se nombre en la billetera.

Alice selecciona la opción para "escanear una solicitud de pago", escanea el código QR que se muestra en la pantalla de la tableta (consulte la Figura 2-13) y se le solicita que confirme su pago, como se muestra en la Figura 2-15.

Alice presiona PAGAR y, un segundo después, la tableta de Bob muestra un pago exitoso. ¡Alice completó su primer pago de LN! Fue rápido, barato y fácil. Ahora puede disfrutar de su café con leche que compró con bitcoin a través de un sistema de pago rápido, económico y descentralizado. A partir de ahora, Alice puede simplemente seleccionar un artículo en la pantalla de la tableta de Bob, escanear el código QR con su teléfono celular, hacer clic en PAGAR y recibir un café, todo en segundos y sin una transacción en cadena.

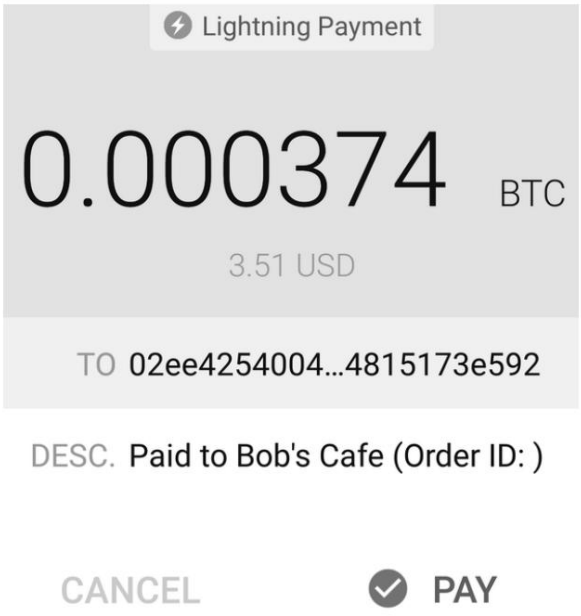


Figura 2-15. Confirmación de envío de Alice

Los pagos relámpago también son mejores para Bob. Confía en que le pagarán por el café con leche de Alice sin esperar una confirmación en la cadena. En el futuro, siempre que Alice tenga ganas de tomar un café en Bob's Cafe, puede optar por pagar con bitcoin en la red Bitcoin o Lightning Network. ¿Cuál crees que elegirá?

Conclusión

En este capítulo, seguimos a Alice mientras descargaba e instalaba su primera billetera Lightning, adquiría y transfería algunos bitcoins, abría su primer canal Lightning y compraba una taza de café al hacer su primer pago en Lightning Network. En los siguientes capítulos, veremos "debajo de las sábanas" cómo funciona cada componente de Lightning Network y cómo el pago de Alice llegó a Bob's Cafe.

---

1 Andreas M. Antonopoulos, ***Mastering Bitcoin,*** 2.ª edición, capítulo 1 (O´Reilly)

2 ACINQ: Desarrolladores de la billetera Eclair Mobile Lightning.

3 Por lo general, no es recomendable reutilizar la misma dirección de Bitcoin para múltiples pagos porque todos
Las transacciones de Bitcoin son públicas. Una persona entrometida que pasara podría escanear el código QR de Alice y ver cómo

muchos consejos que Alice ya ha recibido en esta dirección en la cadena de bloques de Bitcoin. Afortunadamente, Lightning Network ofrece más soluciones privadas para esto, ¡discutidas más adelante en el libro!

4 La billetera Eclair no ofrece una opción para calcular automáticamente las tarifas necesarias y asignar la cantidad máxima de fondos para un canal, por lo que Alice tiene que calcularlo ella misma.

Capítulo 3. Cómo funciona Lightning Network

Ahora que hemos seguido a Alice mientras configuraba una billetera Lightning y le compraba un café a Bob, miraremos debajo del capó y desempacaremos los diferentes componentes de Lightning Network involucrados en ese proceso.
Este capítulo brindará una descripción general de alto nivel y no profundizará en todos los detalles técnicos. El objetivo es más bien ayudarlo a conocer los conceptos y componentes básicos más importantes de Lightning Network.

Si tiene experiencia en informática, criptografía, Bitcoin y desarrollo de protocolos, entonces este capítulo debería ser suficiente para que pueda completar los detalles de conexión usted mismo. Si tiene menos experiencia, este capítulo le brindará una descripción general lo suficientemente buena para que le resulte más fácil comprender las especificaciones del protocolo formal, conocidas como BOLT (Base de la tecnología Lightning). Si es un principiante, este capítulo lo ayudará a comprender mejor los capítulos técnicos del libro.

Si necesita refrescar los fundamentos de Bitcoin, puede encontrar una revisión resumida de los siguientes temas en el Apéndice A:

- Claves y direcciones
- Funciones hash
- Firmas digitales
- Estructura de la transacción
- Entradas y salidas de transacciones
- Encadenamiento de transacciones
- Guión de Bitcoin
- Direcciones multifirma y guiones

- Bloqueos de tiempo
- Guiones complejos

Comenzaremos con una definición de una oración de lo que es Lightning Network y la desglosaremos en el resto de este capítulo.

Lightning Network es una red de ***canales de pago*** entre pares implementada como contratos inteligentes en la cadena de bloques de ***Bitcoin ,*** así como un protocolo de comunicación que define cómo los participantes configuran y ejecutan estos contratos inteligentes.

¿Qué es un canal de pago?

Hay varias formas de describir un canal de pago, según el contexto. Comencemos en un nivel alto y luego agreguemos más detalles.

Un canal de pago es una ***relación financiera*** entre dos nodos en Lightning Network, llamados ***socios de canal.*** La relación financiera asigna un ***saldo de fondos*** (denominado en millisatoshis) entre los dos socios de canal.

El canal de pago está gestionado por un ***protocolo criptográfico,*** lo que significa que los socios del canal utilizan un proceso predefinido basado en la criptografía para redistribuir el saldo del canal a favor de uno u otro socio del canal. El protocolo criptográfico garantiza que un socio de canal no pueda engañar al otro, por lo que los socios no necesitan confiar el uno en el otro.

El protocolo criptográfico se establece mediante la financiación de una ***dirección*** de firma múltiple 2 de 2 que requiere que los dos socios de canal cooperen y evita que cualquiera de los socios de canal gaste los fondos unilateralmente.

En resumen: un canal de pago es una relación financiera entre nodos, asignando fondos desde una dirección de múltiples firmas a través de un protocolo criptográfico estrictamente definido.

Conceptos básicos del canal de pago

Detrás del canal de pago hay simplemente una dirección de firma múltiple 2 de 2 en la cadena de bloques de Bitcoin, para la cual usted tiene una clave y su socio de canal tiene la otra clave.

Usted y su socio de canal negocian una secuencia de transacciones que gastan desde esta dirección de múltiples firmas. En lugar de transmitir y registrar estas transacciones en la cadena de bloques de Bitcoin, ambos se aferran a ellas, sin gastar.

La última transacción en esa secuencia codifica el saldo del canal y define cómo se divide ese saldo entre usted y su socio de canal.

Por lo tanto, agregar una nueva transacción a esta secuencia es equivalente a mover una parte del saldo del canal de un socio de canal a otro, sin que la red Bitcoin lo sepa. A medida que negocia cada nueva transacción, cambiando la asignación de fondos en el canal, también revoca la transacción anterior, de modo que ninguna de las partes pueda regresar a un estado anterior.

Cada transacción en la secuencia utiliza el lenguaje de secuencias de comandos de Bitcoin y, por lo tanto, la negociación de fondos entre usted y su socio de canal se gestiona mediante un contrato inteligente de Bitcoin. El contrato inteligente está configurado para penalizar a un miembro del canal si intenta enviar un estado del canal revocado previamente.

NOTA

Si tiene una transacción no publicada de una dirección de firma múltiple 2 de 2 que le paga parte del saldo, entonces una firma de la otra parte garantiza que puede publicar esta transacción de forma independiente en cualquier momento agregando su propia firma.

La capacidad de mantener una transacción parcialmente firmada, fuera de línea y sin publicar, con la opción de publicar y poseer ese saldo en cualquier momento, es la base de Lightning Network.

Enrutamiento de pagos a través de canales

Una vez que varios participantes tienen canales de una parte a otra, los pagos también se pueden "reenviar" de un canal de pago a otro configurando una ***ruta*** a través de la red que conecta varios canales de pago.

Por ejemplo, Alice puede enviar dinero a Charlie si Alice tiene un canal con Bob y Bob tiene un canal con Charlie.

Mediante el diseño de Lightning Network, es posible extender los contratos inteligentes que operan el canal para que Bob no tenga forma de robar los fondos que se reenvían a través de su canal.

De la misma manera que el contrato inteligente protege a los socios de canal para que no tengan que confiar entre sí, toda la red protege a los participantes para que puedan reenviar pagos sin confiar en ninguno de los otros participantes.

Debido a que los canales se construyen a partir de direcciones de múltiples firmas y las transacciones de actualización de saldo son transacciones de Bitcoin prefirmadas, ¡toda la confianza que se necesita para operar Lightning Network proviene de la confianza en la red descentralizada de Bitcoin!

Las innovaciones antes mencionadas son sin duda los principales avances que permitieron la creación de Lightning Network. Sin embargo, Lightning Network es mucho más que los protocolos criptográficos además del lenguaje Bitcoin Script. Es un protocolo de comunicación integral que permite a los pares intercambiar mensajes Lightning para lograr la transferencia de bitcoin. El protocolo de comunicación define cómo se cifran e intercambian los mensajes Lightning.

Lightning Network también utiliza un protocolo de chismes para distribuir información pública sobre los canales (topología de red) a todos los participantes.

Alice, por ejemplo, necesita la información de la topología de la red para conocer el canal entre Bob y Charlie, de modo que pueda construir una ruta a Charlie.

Por último, pero no menos importante, es importante comprender que Lightning Network no es más que una aplicación sobre Bitcoin, que utiliza Bitcoin.

transacciones y Bitcoin Script. No existe una "moneda Lightning" o una "cadena de bloques Lightning". Más allá de todas las primitivas técnicas, el protocolo LN es una forma creativa de obtener más beneficios de Bitcoin al permitir una cantidad arbitraria de pagos instantáneos con liquidaciones instantáneas sin la necesidad de tener que confiar en nadie más que en la red de Bitcoin.

Canales de pago

Como vimos en el capítulo anterior, Alice usó su software de billetera para crear un canal de pago entre ella y otro participante de LN.

Un canal solo está limitado por tres cosas:

- Primero, el tiempo que tarda Internet en transferir los pocos cientos de bytes de datos que requiere el protocolo para mover fondos de un extremo al otro del canal.
- En segundo lugar, la capacidad del canal, es decir, la cantidad de bitcoin que se asigna al canal cuando se abre.
- En tercer lugar, el límite de tamaño máximo de una transacción de Bitcoin también limita la cantidad de pagos enrutados incompletos (en curso) que se pueden realizar simultáneamente a través de un canal.

Los canales de pago tienen algunas propiedades muy interesantes y útiles:

- Debido a que el tiempo para actualizar un canal depende principalmente de la velocidad de comunicación de Internet, realizar un pago en un canal de pago puede ser casi instantáneo.
- Si el canal está abierto, realizar un pago no requiere la confirmación de bloques de Bitcoin. De hecho, siempre que usted y su socio de canal sigan el protocolo, no se requiere ninguna interacción con la red Bitcoin ni con nadie más que su socio de canal.

- El protocolo criptográfico está construido de tal manera que se necesita poca o ninguna confianza entre usted y su socio de canal. Si su socio no responde o intenta engañarlo, puede pedirle al sistema Bitcoin que actúe como un "tribunal", resolviendo el contrato inteligente que usted y su socio acordaron previamente.
- Los pagos realizados en un canal de pago solo los conocen usted y su socio. En ese sentido, gana privacidad en comparación con Bitcoin, donde cada transacción es pública. Solo el saldo final, que es el total de todos los pagos en ese canal, será visible en la cadena de bloques de Bitcoin.

Bitcoin tenía alrededor de cinco años cuando los desarrolladores talentosos descubrieron por primera vez cómo se podían construir canales de pago enrutables bidireccionales, de por vida indefinida, y ahora hay al menos tres métodos conocidos diferentes.

Este capítulo se centrará en el método de construcción de canales descrito por primera vez en el documento técnico de Lightning Network. por Joseph Poon y Thaddeus Dryja en 2015. Estos se conocen como canales ***Poon-Dryja*** y son el método de construcción de canales que se usa actualmente en Lightning Network. Los otros dos métodos propuestos son los canales de ***micropago dúplex ,*** introducidos por Christian Decker casi al mismo tiempo que los canales Poon-Dryja y los canales ***eltoo ,*** introducidos en "eltoo: A Simple Layer2 Protocol for Bitcoin" por Christian Decker, Rusty Russel y (coautor de este libro) Olaoluwa Osuntokun en 2018.

Los canales de eltoo tienen algunas propiedades interesantes y simplifican la implementación de los canales de pago. Sin embargo, los canales de eltoo requieren un cambio en el lenguaje Bitcoin Script y, por lo tanto, no se pueden implementar en la red principal de Bitcoin a partir de 2020.

Dirección multifirma

Los canales de pago se construyen sobre 2 de 2 direcciones de firmas múltiples.

En resumen, una dirección de múltiples firmas es donde Bitcoin está bloqueado, por lo que requiere múltiples firmas para desbloquear y gastar. En una firma múltiple 2 de 2

dirección, como se usa en Lightning Network, hay dos firmantes participantes y ***ambos*** deben firmar para gastar los fondos.

Las secuencias de comandos y las direcciones de firmas múltiples se explican con más detalle en "Secuencias de comandos de firmas múltiples".

Transacción de financiación

El bloque de construcción fundamental de un canal de pago es una dirección de firma múltiple 2 de 2. Uno de los dos socios de canal financiará el canal de pago enviando bitcoins a la dirección de firma múltiple. Esta transacción se denomina ***transacción de financiación*** y se registra en la cadena de bloques Bitcoin 1 .

Aunque la transacción de financiación es pública, no es obvio que se trate de un canal de pago Lightning hasta que se cierra, a menos que el canal se anuncie públicamente. Los canales generalmente se anuncian públicamente mediante nodos de enrutamiento que desean reenviar pagos. Sin embargo, también existen canales no anunciados y, por lo general, son creados por nodos móviles que no participan activamente en el enrutamiento. Además, los pagos del canal aún no son visibles para nadie más que los socios del canal, ni tampoco la distribución del saldo del canal entre ellos.

La cantidad depositada en la dirección multifirma se denomina ***capacidad del canal*** y establece la cantidad máxima que se puede enviar a través del canal de pago. Sin embargo, dado que los fondos se pueden enviar de un lado a otro, la capacidad del canal no es el límite superior de cuánto valor puede fluir a través del canal. Eso es porque si la capacidad del canal se agota con pagos en una dirección, se puede usar para enviar pagos en la dirección opuesta nuevamente.

> **NOTA**
>
> Los fondos enviados a la dirección de firma múltiple en la transacción de financiación a veces se denominan "bloqueados en un canal Lightning". Sin embargo, en la práctica, los fondos en un canal Lightning no están "bloqueados", sino más bien "liberados". Los fondos del canal Lightning son más líquidos que los fondos en la cadena de bloques de Bitcoin, ya que se pueden gastar más rápido, más barato y de forma más privada. Transferir fondos a Lightning Network tiene algunas desventajas (como la necesidad de mantenerlos en una billetera "activa"), pero la idea de "bloquear fondos" en Lightning es engañosa.

Ejemplo de un procedimiento de apertura de canal deficiente

Si piensa detenidamente en las direcciones de varias firmas de 2 de 2, se dará cuenta de que poner sus fondos en esa dirección parece conllevar cierto riesgo. ¿Qué sucede si su socio de canal se niega a firmar una transacción para liberar los fondos? ¿Están atrapados para siempre? Veamos ahora ese escenario y cómo lo evita el protocolo LN.

Alice y Bob quieren crear un canal de pago. Cada uno de ellos crea un par de claves pública/privada y luego intercambian claves públicas. Ahora, pueden construir una firma múltiple 2 de 2 con las dos claves públicas, formando la base de su canal de pago.

A continuación, Alice construye una transacción de Bitcoin enviando algunos mBTC a la dirección de firma múltiple creada a partir de las claves públicas de Alice y Bob. Si Alice no realiza ningún paso adicional y simplemente transmite esta transacción, debe confiar en que Bob proporcionará su firma para gastar desde la dirección de múltiples firmas. Bob, por otro lado, tiene la oportunidad de chantajear a Alice reteniendo su firma y negándole a Alice acceso a sus fondos.

Para evitar esto, Alice deberá crear una transacción adicional que gaste desde la dirección de múltiples firmas, reembolsando su mBTC. Luego, Alice hace que Bob firme la transacción de reembolso ***antes*** de transmitir su transacción de financiación a la red de Bitcoin. De esta manera, Alice puede obtener un reembolso incluso si Bob desaparece o no coopera.

La transacción de "reembolso" que protege a Alice es la primera de una clase de transacciones llamadas transacciones de ***compromiso,*** que examinaremos en

más detalle a continuación.

Transacción de compromiso

Una ***transacción de compromiso*** es una transacción que paga a cada socio de canal su saldo de canal y garantiza que los socios de canal no tengan que confiar entre sí. Al firmar una transacción de compromiso, cada socio de canal "se compromete" con el saldo actual y le da al otro socio de canal la capacidad de recuperar sus fondos cuando lo desee.

Al mantener una transacción de compromiso firmada, cada socio de canal puede obtener sus fondos incluso sin la cooperación del otro socio de canal. Esto los protege contra la desaparición del otro socio del canal, la negativa a cooperar o el intento de hacer trampa al violar el protocolo del canal de pago.

La transacción de compromiso que Alice preparó en el ejemplo anterior fue un reembolso de su pago inicial a la dirección multifirma. Sin embargo, de manera más general, una transacción de compromiso divide los fondos del canal de pago, pagando a los dos socios del canal de acuerdo con la distribución (saldo) que cada uno tiene. Al principio, Alice se queda con todo el saldo, por lo que es un reembolso simple. Pero a medida que los fondos fluyen de Alice a Bob, intercambiarán firmas por nuevas transacciones de compromiso que representan la distribución del nuevo saldo, con una parte de los fondos pagada a Alice y otra a Bob.

Supongamos que Alice abre un canal con una capacidad de 100 000 satoshi con Bob. Inicialmente, Alice posee 100 000 satoshi, la totalidad de los fondos del canal. Así es como funciona el protocolo del canal de pago:

1. Alice crea un nuevo par de claves pública/privada e informa a Bob que ella desea abrir un canal a través del mensaje open_channel (un mensaje en el protocolo LN).

2. Bob también crea un nuevo par de claves pública/privada y acepta aceptar un canal de Alice, enviando su clave pública a Alice a través del mensaje accept_channel.

3. Alice ahora crea una transacción de financiación desde su billetera que envía 100k satoshi a la dirección de múltiples firmas con un script de bloqueo: 2 <PubKey Alice> <PubKey Bob> 2 CHECKMULTISIG.
4. Alice aún no transmite esta transacción de financiación, pero envía a Bob el ID de la transacción en un mensaje de creación de fondos junto con su firma para la transacción de compromiso de Bob.
5. Tanto Alice como Bob crean su versión de una transacción de compromiso. Esta transacción se gastará de la transacción de financiación y enviará todos los bitcoins a una dirección controlada por Alice.
6. Alice y Bob no necesitan intercambiar estos compromisos transacciones, ya que cada uno sabe cómo se construyen y pueden construir ambas de forma independiente (porque han acordado un ordenamiento canónico de las entradas y salidas). Solo necesitan intercambiar firmas.
7. Bob proporciona una firma para la transacción de compromiso de Alice y se la devuelve a Alice a través del mensaje financiado_firmado.
8. Ahora que se han intercambiado las firmas, Alice transmitirá la transacción de financiación a la red Bitcoin.

Al seguir este protocolo, Alice no renuncia a la propiedad de su satoshi de 100k aunque los fondos se envíen a una dirección de 2 de 2 firmas múltiples para la cual Alice controla solo una clave. Si Bob deja de responderle a Alice, ella podrá transmitir su transacción de compromiso y recibir sus fondos.

espalda. Sus únicos costos son las tarifas de las transacciones en cadena. Mientras siga el protocolo, este es su único riesgo al abrir un canal.

Después de este intercambio inicial, se crean transacciones de compromiso cada vez que cambia el saldo del canal. En otras palabras, cada vez que se envía un pago entre Alice y Bob, se crean nuevas transacciones de compromiso y se intercambian firmas. Cada nueva transacción de compromiso codifica el último saldo entre Alice y Bob.

Si Alice quiere enviar 30k satoshi a Bob, ambos crearían una nueva versión de sus transacciones de compromiso, que ahora pagarían 70k satoshi a Alice y 30k satoshi a Bob. Al codificar un nuevo saldo para Alice y Bob, las nuevas transacciones de compromiso son el medio por el cual se "envía" un pago a través del canal.

Ahora que entendemos las transacciones de compromiso, veamos algunos de los detalles más sutiles. Puede notar que este protocolo deja un camino para que Alice o Bob hagan trampa.

Hacer trampa con estado anterior

¿Cuántas transacciones de compromiso tiene Alice después de pagar 30k satoshi a Bob? Tiene dos: el original que le paga 100k satoshi y el más reciente, que le paga 70k satoshi y Bob 30k satoshi.

En el protocolo de canal que hemos visto hasta ahora, nada impide que Alice publique una transacción de compromiso anterior. Una Alice infiel podría publicar la transacción de compromiso que le otorga 100k satoshi. Dado que Bob firmó esa transacción de compromiso, no puede evitar que Alice la transmita.

Se necesita algún mecanismo para evitar que Alice publique una transacción de compromiso anterior. Veamos ahora cómo se puede lograr esto y cómo permite que Lightning Network funcione sin necesidad de confianza entre Alice y Bob.

Debido a que Bitcoin es resistente a la censura, nadie puede evitar que alguien publique una transacción de compromiso anterior. Para evitar esta forma de hacer trampa, se construyen transacciones de compromiso de modo que si se transmite una antigua, el tramposo puede ser castigado. Al hacer que la penalización sea lo suficientemente grande, creamos un fuerte incentivo contra las trampas y esto hace que el sistema sea seguro.

La forma en que funciona la sanción es dando a la parte engañada la oportunidad de reclamar el saldo del tramposo. Entonces, si alguien intenta hacer trampa al transmitir una transacción de compromiso anterior, en la que se le paga un saldo más alto del que debe, la otra parte puede castigarlo tomando

tanto su propio saldo como el del tramposo. El tramposo lo pierde todo.

PROPINA

Puede notar que si Alice agota el saldo de su canal casi por completo, entonces podría intentar hacer trampa con poco riesgo. La penalización de Bob no sería tan dolorosa si el balance de su canal es bajo. Para evitar esto, el protocolo Lightning requiere que cada socio de canal mantenga un saldo mínimo en el canal (llamado ***reserva)*** para que siempre tengan "piel en el juego".

Repasemos nuevamente el escenario de construcción del canal, agregando un mecanismo de penalización para proteger contra las trampas:

1. Alice crea un canal con Bob y pone 100k satoshi en él.
2. Alice envía 30k satoshi a Bob.
3. Alice intenta engañar a Bob con su satoshi ganado de 30k al publicar una transacción de compromiso anterior que reclama el satoshi completo de 100k para ella.
4. Bob detecta el fraude y castiga a Alice tomando el satoshi completo de 100k para él.
5. Bob termina con 100k satoshi, ganando 70k satoshi por atrapar a Alice haciendo trampa.
6. Alice termina con 0 satoshi.
7. Tratando de engañar a Bob con 30k satoshi, ella pierde el 70k satoshi ella poseía

Con un fuerte mecanismo de penalización, Alice no se siente tentada a hacer trampa al publicar una transacción de compromiso anterior porque corre el riesgo de perder todo su saldo.

> **NOTA**
>
> En el Capítulo 12 de ***Mastering Bitcoin,*** Andreas Antonopoulos (el coautor de este libro) lo afirma de la siguiente manera: “Una característica clave de Bitcoin es que una vez que una transacción es válida, sigue siendo válida y no caduca. La única forma de cancelar una transacción es gastando el doble de sus entradas con otra transacción antes de que se extraiga".

Ahora que entendemos ***por qué*** se necesita un mecanismo de penalización y cómo evitará las trampas, veamos ***cómo*** funciona en detalle.

Por lo general, la transacción de compromiso tiene al menos dos salidas, pagando a cada socio de canal. Cambiamos esto para agregar un ***retraso*** de bloqueo de tiempo y un ***secreto de revocación*** a uno de los pagos. El bloqueo de tiempo evita que el propietario de la salida la gaste inmediatamente una vez que la transacción de compromiso se incluye en un bloque. El secreto de revocación permite que cualquiera de las partes gaste inmediatamente ese pago, sin pasar por el bloqueo de tiempo.

Entonces, en nuestro ejemplo, Bob tiene una transacción de compromiso que le paga a Alice ***inmediatamente,*** pero su propio pago se retrasa y es revocable. Alice también tiene una transacción de compromiso, pero la suya es todo lo contrario: le paga a Bob de inmediato, pero su propio pago se demora y es revocable.

Los dos socios de canal tienen la mitad del secreto de revocación, de modo que ninguno conoce todo el secreto. Si comparten su mitad, el otro socio de canal tiene el secreto completo y puede usarlo para ejercer la condición de revocación. Al firmar una nueva transacción de compromiso, cada socio de canal revoca el compromiso anterior dando a la otra parte su mitad del secreto de revocación.

Examinaremos el mecanismo de revocación con más detalle en "Revocación y confirmación", donde aprenderemos los detalles de cómo se construyen y utilizan los secretos de revocación.

En términos simples, Alice firma la nueva transacción de compromiso de Bob solo si Bob ofrece su mitad del secreto de revocación del compromiso anterior. Bob solo firma la nueva transacción de compromiso de Alice si ella le da la mitad del secreto de revocación del compromiso anterior.

Con cada nuevo compromiso, intercambian el secreto de "castigo" necesario que les permite ***revocar*** efectivamente la transacción de compromiso anterior al hacer que su transmisión no sea rentable. Esencialmente, destruyen la capacidad de utilizar los antiguos compromisos cuando firman los nuevos. Lo que queremos decir es que, si bien todavía es técnicamente posible utilizar compromisos antiguos, el mecanismo de sanción hace que sea económicamente irracional hacerlo.

El bloqueo de tiempo se establece en un número de bloques hasta 2016 (aproximadamente dos semanas). Si cualquiera de los socios del canal publica una transacción de compromiso sin cooperar con el otro socio, tendrá que esperar esa cantidad de bloques (por ejemplo, dos semanas) para reclamar su saldo. El otro socio de canal puede reclamar su propio saldo en cualquier momento. Además, si el compromiso que publicaron fue revocado previamente, el socio de canal ***también*** puede reclamar inmediatamente el saldo de la parte infiel, saltando el bloqueo de tiempo y castigando al infiel.

El bloqueo de tiempo es ajustable y se puede negociar entre socios de canal.
Por lo general, es más largo para los canales de mayor capacidad y más corto para los canales más pequeños alinear los incentivos con el valor de los fondos.

Para cada nueva actualización del saldo del canal, se deben crear y guardar nuevas transacciones de compromiso y nuevos secretos de revocación. Mientras un canal permanezca abierto, todos los secretos de revocación ***creados*** para el canal deben conservarse porque podrían ser necesarios en el futuro. Afortunadamente, los secretos son bastante pequeños y solo los socios de canal deben guardarlos, no toda la red. Además, debido a un mecanismo de derivación inteligente utilizado para derivar secretos de revocación, solo necesitamos almacenar el secreto más reciente, ya que los secretos anteriores pueden derivarse de él (consulte “Hacer trampa y penalizar en la práctica”).

Sin embargo, administrar y almacenar los secretos de revocación es una de las partes más elaboradas de los nodos Lightning que requieren que los operadores de nodos mantengan copias de seguridad.

> **NOTA**
>
> Las tecnologías como los servicios de torre de vigilancia o el cambio del protocolo de construcción del canal al protocolo eltoo podrían ser estrategias futuras para mitigar estos problemas y reducir la necesidad de secretos de revocación, transacciones de penalización y copias de seguridad del canal.

Alice puede cerrar el canal en cualquier momento si Bob no responde, reclamando su parte justa del saldo. Después de publicar la ***última*** transacción de compromiso en la cadena, Alice tiene que esperar a que expire el bloqueo de tiempo antes de poder gastar los fondos de la transacción de compromiso. Como veremos más adelante, hay una manera más fácil de cerrar un canal sin esperar, siempre que Alice y Bob estén en línea y cooperen para cerrar el canal con la asignación de saldo correcta. Pero las transacciones de compromiso almacenadas por cada socio de canal actúan como una prueba de fallas, asegurando que no pierdan fondos si hay un problema con su socio de canal.

anunciando el canal

Los socios de canal pueden acordar anunciar su canal a toda Lightning Network, convirtiéndolo en un ***canal público.*** Para anunciar el canal, utilizan el protocolo de chismes de Lightning Network para informar a otros nodos sobre la existencia, la capacidad y las tarifas del canal.

Anunciar canales públicamente permite que otros nodos los utilicen para el enrutamiento de pagos, lo que también genera tarifas de enrutamiento para los socios del canal.

Por el contrario, los socios del canal pueden decidir no anunciar el canal, convirtiéndolo en un canal ***no anunciado .***

> **NOTA**
>
> Es posible que escuche el término "canal privado" utilizado para describir un canal no anunciado. Evitamos usar ese término porque es engañoso y crea una falsa sensación de privacidad.
> Aunque un canal no anunciado no será conocido por otros mientras esté en uso, su existencia y capacidad se revelarán cuando el canal cierre porque esos detalles serán visibles en la cadena en la transacción de liquidación final. Su existencia también puede filtrarse en una variedad de otras formas, por lo que evitamos llamarlo "privado".

Los canales no anunciados todavía se utilizan para enrutar pagos, pero solo por los nodos que saben de su existencia, o que reciben "sugerencias de enrutamiento" sobre una ruta que incluye un canal no anunciado.

Cuando un canal y su capacidad se anuncian públicamente mediante el protocolo de chismes, el anuncio también puede incluir información sobre el canal (metadatos), como sus tarifas de enrutamiento y la duración del bloqueo de tiempo.

Cuando los nuevos nodos se unen a Lightning Network, recopilan los anuncios del canal propagados a través del protocolo de chismes de sus pares, creando un mapa interno de Lightning Network. Este mapa se puede usar para encontrar rutas de pago, conectando canales de extremo a extremo.

Cerrando el Canal

La mejor forma de cerrar un canal es... ¡no cerrarlo! Abrir y cerrar canales requiere una transacción en cadena, que generará tarifas de transacción.
Por lo tanto, es mejor mantener los canales abiertos el mayor tiempo posible. Puede seguir usando su canal para realizar y reenviar pagos, siempre que tenga suficiente capacidad en su extremo del canal. Pero incluso si envía todo el saldo al otro extremo del canal, puede usar el canal para recibir pagos de su socio de canal. Este concepto de usar un canal en una dirección y luego usarlo en la dirección opuesta se llama "reequilibrio" y lo examinaremos con más detalle en otro capítulo. Al reequilibrar un canal, puede mantenerse abierto casi indefinidamente y usarse para una cantidad de pagos esencialmente ilimitada.

Sin embargo, a veces cerrar un canal es deseable o necesario. Por ejemplo:

- Desea reducir el saldo retenido en sus canales Lightning por razones de seguridad y desea enviar fondos a "almacenamiento en frío".
- Su socio de canal deja de responder durante mucho tiempo y ya no puede usar el canal.
- El canal no se usa con frecuencia porque su socio de canal no es un nodo bien conectado, por lo que desea usar los fondos para

otro canal con un nodo mejor conectado.

- Su socio de canal ha infringido el protocolo debido a un error de software o a propósito, lo que le obligó a cerrar el canal para proteger sus fondos.

Hay tres formas de cerrar un canal de pago:

- Cierre mutuo (el buen camino)
- Forzar el cierre (de la mala manera)
- Incumplimiento de protocolo (la forma fea)

Cada uno de estos métodos es útil para diferentes circunstancias, que exploraremos en las próximas secciones de este capítulo. Por ejemplo, si su socio de canal no está conectado, no podrá seguir “el buen camino” porque no se puede lograr un cierre mutuo sin un socio colaborador. Por lo general, su software de LN seleccionará automáticamente el mejor mecanismo de cierre disponible según las circunstancias.

Cierre mutuo (el buen camino)

El cierre mutuo es cuando ambos socios del canal acuerdan cerrar un canal y es el método preferido de cierre del canal.

Cuando decida que desea cerrar un canal, su nodo LN informará a su socio de canal sobre su intención. Ahora tanto su nodo como el nodo del socio de canal trabajan juntos para cerrar el canal. No se aceptarán nuevos intentos de enrutamiento de ninguno de los socios de canal, y cualquier intento de enrutamiento en curso se resolverá o eliminará después de que se agote el tiempo de espera.

La finalización de los intentos de enrutamiento lleva tiempo, por lo que un cierre mutuo también puede tardar algún tiempo en completarse.

Una vez que no hay intentos de enrutamiento pendientes, los nodos cooperan para preparar una ***transacción de cierre.*** Esta transacción es similar a la transacción de compromiso: codifica el último saldo del canal, pero las salidas NO están gravadas con un bloqueo de tiempo.

Las tarifas de transacción en cadena para la transacción de cierre son pagadas por el socio de canal que abrió el canal y no por el que inició el procedimiento de cierre. Usando el estimador de tarifas en cadena, los socios del canal acuerdan la tarifa apropiada y ambos firman la transacción de cierre.

Una vez que la transacción de cierre es transmitida y confirmada por la red de Bitcoin, el canal se cierra efectivamente y cada socio del canal ha recibido su parte del saldo del canal. A pesar del tiempo de espera, un cierre mutuo suele ser más rápido que un cierre forzado.

Forzar el cierre (de la mala manera)

Un cierre forzado es cuando un socio de canal intenta cerrar un canal sin el consentimiento del otro socio de canal.

Esto suele suceder cuando uno de los socios de canal no está disponible, por lo que no es posible un cierre mutuo. En este caso, iniciaría una fuerza cercana para cerrar unilateralmente el canal y “liberar” los fondos.

Para iniciar un cierre forzado, simplemente puede publicar la última transacción de compromiso que tiene su nodo. Después de todo, para eso están las transacciones de compromiso: ofrecen una garantía de que no necesita confiar en su socio de canal para recuperar el saldo de su canal.

Una vez que transmita la última transacción de compromiso a la red de Bitcoin y se confirme, se crearán dos salidas gastables, una para usted y otra para su socio. Como discutimos anteriormente, la red Bitcoin no tiene forma de saber si esta fue la transacción de compromiso más reciente o una antigua que se publicó para robarle a su socio. Por lo tanto, esta transacción de compromiso le dará una ligera ventaja a su socio. El socio que inició el cierre forzado tendrá su producción comprometida por un bloqueo de tiempo, y la producción del otro socio se podrá gastar de inmediato. En el caso de que haya transmitido una transacción de compromiso anterior, la demora del bloqueo de tiempo le da a su socio la oportunidad de disputar la transacción utilizando el secreto de revocación y castigarlo por hacer trampa.

Al publicar una transacción de compromiso durante un cierre forzado, las tarifas en cadena serán más altas que las de un cierre mutuo por varias razones:

1. Cuando se negoció la transacción de compromiso, el canal
   los socios no sabían cuánto serían las tarifas en cadena en el momento futuro en que se transmitiría la transacción. Dado que las tarifas no se pueden cambiar sin cambiar los resultados de la transacción de compromiso (que necesita ambas firmas), y dado que el cierre forzado ocurre cuando un socio de canal no está disponible para firmar, los desarrolladores del protocolo decidieron ser muy generosos con la tasa de tarifa incluida. en las transacciones de compromiso. Puede ser hasta cinco veces más alto que lo que sugieren los estimadores de tarifas en el momento en que se negocia la transacción de compromiso.

2. La transacción de compromiso incluye salidas adicionales para cualquier intento de enrutamiento pendiente de los contratos bloqueados en el tiempo (HTLC), lo que hace que la transacción de compromiso sea más grande (en términos de bytes) que una transacción de cierre mutuo. Las transacciones más grandes incurren en más tarifas.

3. Cualquier intento de enrutamiento pendiente deberá resolverse en la cadena, lo que generará transacciones adicionales en la cadena.

> **NOTA**
>
> Los contratos de bloqueo de tiempo de hash (HTLC) se tratarán en detalle en "Contratos de bloqueo de tiempo de hash". Por ahora, suponga que estos son pagos que se enrutan a través de Lightning Network, en lugar de pagos realizados directamente entre los dos socios de canal. Estos HTLC se llevan como salidas adicionales en las transacciones de compromiso, lo que aumenta el tamaño de la transacción y las tarifas en cadena.

En general, no se recomienda un cierre forzado a menos que sea absolutamente necesario. Tus fondos estarán bloqueados por más tiempo y la persona que abrió el canal tendrá que pagar tarifas más altas. Además, es posible que deba pagar tarifas en cadena para cancelar o liquidar los intentos de enrutamiento, incluso si no abrió el canal.

Si conoce al socio de canal, puede considerar ponerse en contacto con esa persona o empresa para preguntar por qué su nodo Lightning está inactivo y

solicitar que lo reinicien para que puedan lograr un cierre mutuo del canal.

Debe considerar una fuerza cercana solo como último recurso.

Incumplimiento de protocolo (la forma fea)

Una violación de protocolo es cuando su socio de canal intenta engañarlo, ya sea deliberadamente o no, mediante la publicación de una transacción de compromiso desactualizada en la cadena de bloques de Bitcoin, lo que esencialmente inicia una fuerza (deshonesta) cerca de su lado.

Su nodo debe estar en línea y observar nuevos bloques y transacciones en la cadena de bloques de Bitcoin para detectar esto.

Debido a que el pago de su socio de canal se verá afectado por un bloqueo de tiempo, su nodo tiene algo de tiempo para actuar para detectar una infracción de protocolo y publicar una ***transacción de castigo*** antes de que expire el bloqueo de tiempo.

Si detecta correctamente la infracción del protocolo y aplica la sanción, recibirá todos los fondos del canal, incluidos los fondos de su socio de canal.

En este escenario, el cierre del canal será bastante rápido. Deberá pagar tarifas en cadena para publicar la transacción de castigo, pero su nodo puede establecer estas tarifas de acuerdo con la estimación de la tarifa y no pagar en exceso. Por lo general, querrá pagar tarifas más altas para garantizar la confirmación lo antes posible. Sin embargo, debido a que eventualmente recibirá todos los fondos del tramposo, es esencialmente el tramposo quien pagará por esta transacción.

Si no detecta la violación del protocolo y el bloqueo de tiempo caduca, recibirá solo los fondos que le asignó la transacción de compromiso que publicó su socio. Todos los fondos que recibió después de esto habrán sido robados por su socio. Si hay algún saldo asignado a usted, tendrá que pagar tarifas en cadena para cobrar ese saldo.

Al igual que con un cierre forzado, todos los intentos de enrutamiento pendientes también deberán resolverse en la transacción de compromiso.

Una violación de protocolo se puede ejecutar más rápido que un cierre mutuo porque no espera para negociar un cierre con su socio, y más rápido que un cierre forzado porque no necesita esperar a que expire su bloqueo de tiempo.

La teoría del juego predice que hacer trampa no es una estrategia atractiva porque es fácil detectar a un tramposo, y el tramposo corre el riesgo de perder ***todos*** sus fondos mientras solo espera ganar lo que tenía en un estado anterior. Además, a medida que Lightning Network madure y las torres de vigilancia estén ampliamente disponibles, los tramposos serán detectables por un tercero, incluso si el socio de canal engañado está desconectado.

Por lo tanto, no recomendamos hacer trampa. Sin embargo, recomendamos que cualquier persona que atrape a un tramposo lo castigue tomando sus fondos.

Entonces, ¿cómo detectar una trampa o una infracción de protocolo en sus actividades diarias? Lo hace ejecutando un software que monitorea la cadena de bloques pública de Bitcoin en busca de transacciones en cadena que correspondan a cualquier transacción de compromiso para cualquiera de sus canales. Este software es uno de tres tipos:

- Un nodo Lightning correctamente mantenido, que funciona las 24 horas del día, los 7 días de la semana
- Un nodo de torre de vigilancia de un solo propósito que ejecuta para ver sus canales
- Un nodo de torre de vigilancia de terceros que paga para ver sus canales

Recuerde que la transacción de compromiso tiene un tiempo de espera especificado en un número determinado de bloques, hasta un máximo de 2.016 bloques. Siempre que ejecute su nodo Lightning una vez antes de que se alcance el período de tiempo de espera, detectará todos los intentos de trampa. No es aconsejable correr este tipo de riesgo; es importante mantener un nodo bien mantenido funcionando continuamente (consulte "¿Por qué es importante la confiabilidad para ejecutar un nodo Lightning?").

Facturas

La mayoría de los pagos en Lightning Network comienzan con una factura, generada por el destinatario del pago. En nuestro ejemplo anterior, Bob crea una factura para solicitar un pago a Alice.

> **NOTA**
>
> Existe una forma de enviar un pago no solicitado sin factura, utilizando una solución en el protocolo llamada keyend. Examinaremos esto en "Pagos espontáneos Keysend".

Una factura es una instrucción de pago simple que contiene información como un identificador de pago único (llamado hash de pago), un destinatario, un monto y una descripción de texto opcional.

La parte más importante de la factura es el hash de pago, que permite que el pago viaje a través de múltiples canales de forma ***atómica .*** Atómico, en informática, significa cualquier acción o cambio de estado que se completa con éxito o no se completa; no hay posibilidad de un estado intermedio o acción parcial. En Lightning Network, eso significa que el pago recorre todo el camino o falla por completo. No puede completarse parcialmente de modo que un nodo intermedio en la ruta pueda recibir el pago y conservarlo. No existe tal cosa como un "pago parcial" o un "pago parcialmente exitoso".

Las facturas no se comunican a través de Lightning Network. En cambio, se comunican "fuera de banda", utilizando cualquier otro mecanismo de comunicación. Esto es similar a cómo se comunican las direcciones de Bitcoin a los remitentes fuera de la red de Bitcoin: como un código QR, por correo electrónico o mensaje de texto. Por ejemplo, Bob puede presentar una factura Lightning a Alice como un código QR, por correo electrónico o a través de cualquier otro canal de mensajes.

Las facturas generalmente se codifican como una cadena larga codificada en bech32 o como un código QR, para ser escaneadas por una billetera Lightning de un teléfono inteligente. La factura contiene la cantidad de bitcoin que se solicita y una firma del destinatario. El remitente utiliza la firma para extraer la clave pública (también

conocido como ID de nodo) del destinatario para que el remitente sepa dónde enviar el pago.

¿Notaste cómo esto contrasta con Bitcoin y cómo se usan diferentes términos? En Bitcoin, el destinatario pasa una dirección al remitente. En Lightning, el destinatario crea una factura y envía una factura al remitente. En Bitcoin, el remitente envía fondos a una dirección. En Lightning, el remitente paga una factura y el pago se enruta al destinatario. Bitcoin se basa en el concepto de una "dirección" y Lightning es una red de pago basada en el concepto de una "factura". En Bitcoin, creamos una "transacción", mientras que en Lightning enviamos un "pago".

Hash de pago y preimagen

La parte más importante de la factura es el ***hash de pago.*** Al construir la factura, Bob hará un hash de pago de la siguiente manera:

1. Bob elige un número aleatorio r. Este número aleatorio se llama ***preimagen*** o ***secreto de pago.***
2. Bob usa SHA-256 para calcular el hash ***H*** de ***r*** llamado ***pago hash:***

   H = SHA-256(r).

> **NOTA**
>
> El término ***preimagen*** proviene de las matemáticas. En cualquier función ***y*** = ***f(x),*** el conjunto de entradas que producen un cierto valor ***y*** se denominan preimagen de y. En este caso, la función es el algoritmo hash SHA-256, y cualquier valor ***r*** que produzca el hash ***H*** se denomina preimagen.

No existe una forma conocida de encontrar el inverso de SHA-256 (es decir, calcular una preimagen a partir de un hash). Solo Bob conoce el valor de r, por lo que es el secreto de Bob. Pero una vez que Bob revela r, cualquiera que tenga el hash ***H*** puede comprobar que ***r*** es el secreto correcto, calculando SHA-256(r) y comprobando que coincide con H.

El proceso de pago de Lightning Network solo es seguro si ***r*** se elige de forma completamente aleatoria y no es predecible. Esta seguridad se basa en el hecho de que las funciones hash no se pueden invertir ni aplicar fuerza bruta y, por lo tanto, nadie puede encontrar ***r*** de H.

Metadatos adicionales

Las facturas pueden incluir opcionalmente otros metadatos útiles, como una breve descripción de texto. Si un usuario tiene varias facturas para pagar, el usuario puede leer la descripción y recordar de qué se trata la factura.

La factura también puede incluir algunas ***sugerencias de enrutamiento,*** que permiten al remitente utilizar canales no anunciados para construir una ruta hacia el destinatario. Las sugerencias de enrutamiento también se pueden usar para sugerir canales públicos, por ejemplo, canales que el destinatario sabe que tienen suficiente capacidad de entrada para enrutar el pago.

En caso de que el nodo Lightning del remitente no pueda enviar el pago a través de Lightning Network, las facturas pueden incluir opcionalmente una dirección de Bitcoin en cadena como alternativa.

> **NOTA**
>
> Si bien siempre es posible "retroceder" a una transacción de Bitcoin en cadena, en realidad es mejor abrir un nuevo canal para el destinatario. Si tiene que incurrir en tarifas de cadena para realizar un pago, también podría incurrir en esas tarifas para abrir un canal y realizar el pago a través de Lightning. Una vez que se realiza el pago, le queda un canal abierto que tiene liquidez en el extremo del destinatario y puede usarse para enrutar los pagos de regreso a su nodo Lightning en el futuro. Una transacción en cadena le brinda un pago y un canal para uso futuro.

Las facturas relámpago contienen una fecha de vencimiento. Dado que el destinatario debe conservar la preimagen ***r*** para cada factura emitida, es útil que las facturas caduquen para que estas preimágenes no tengan que conservarse para siempre. Una vez que una factura vence o se paga, el destinatario puede descartar la preimagen.

Entrega del pago

Hemos visto cómo el destinatario crea una factura que contiene un hash de pago. Este hash de pago se utilizará para mover el pago a través de una serie de canales de pago, del remitente al destinatario, incluso si no tienen un canal de pago directo entre ellos.

En las próximas secciones, profundizaremos en las ideas y los métodos que se utilizan para entregar un pago a través de Lightning Network y utilizaremos todos los conceptos que hemos presentado hasta ahora.

Primero, veamos el protocolo de comunicación de Lightning Network.

El protocolo de chismes entre pares

Como mencionamos anteriormente, cuando se construye un canal de pago, los socios del canal tienen la opción de hacerlo público, anunciando su existencia y detalles a toda Lightning Network.

Los anuncios del canal se comunican a través de un ***protocolo de chismes*** entre pares . Un protocolo peer-to-peer es un protocolo de comunicaciones en el que cada nodo se conecta a una selección aleatoria de otros nodos en la red, generalmente a través de TCP/IP. Cada uno de los nodos que están conectados directamente (a través de TCP/IP) a su nodo se denominan sus ***pares.*** Su nodo, a su vez, es uno de sus pares. Tenga en cuenta que cuando decimos que su nodo está conectado a otros pares, no queremos decir que tiene canales de pago, sino que está conectado a través del protocolo de chismes.

Después de abrir un canal, un nodo puede optar por enviar un anuncio del canal a través del mensaje channel_announcement a sus pares.
Cada par valida la información del mensaje channel_announcement y verifica que la transacción de financiación esté confirmada en la cadena de bloques de Bitcoin. Después de la verificación, el nodo reenviará el mensaje de chismes a sus propios compañeros, y ellos lo reenviarán a sus compañeros, y así sucesivamente, difundiendo el anuncio en toda la red. Para evitar una comunicación excesiva, el anuncio del canal solo lo reenvía cada nodo si no lo ha reenviado previamente.

El protocolo de chismes también se usa para anunciar información sobre nodos conocidos con el mensaje node_announcement. Para que se reenvíe este mensaje, un nodo debe tener al menos un canal público anunciado en el protocolo de chismes, nuevamente para evitar un tráfico de comunicación excesivo.

Los canales de pago tienen varios metadatos que son útiles para otros participantes de la red. Estos metadatos se utilizan principalmente para tomar decisiones de enrutamiento. Debido a que los nodos pueden cambiar ocasionalmente los metadatos de sus canales, esta información se comparte en un mensaje channel_update. Estos mensajes solo se reenviarán aproximadamente cuatro veces al día (por canal) para evitar una comunicación excesiva. El protocolo de chismes también tiene una variedad de consultas y mensajes para sincronizar inicialmente un nodo con la vista de la red o para actualizar la vista del nodo después de estar desconectado por un tiempo.

Un desafío importante para los participantes de Lightning Network es que la información de topología que comparte el protocolo de chismes es solo parcial. Por ejemplo, la capacidad de los canales de pago se comparte en el protocolo de chismes a través del mensaje channel_announcement. Sin embargo, esta información no es tan útil como la distribución real de la capacidad en términos del equilibrio local entre los dos socios de canal. Un nodo solo puede reenviar la cantidad de bitcoin que realmente posee (saldo local) dentro de ese canal.

Si bien Lightning Network podría haber sido diseñado para compartir información de balance de canales y una topología precisa, esto no se ha hecho por varias razones:

- Para proteger la privacidad de los usuarios, no menciona cada transacción financiera y pago. Las actualizaciones del saldo del canal revelarían que un pago se ha movido a través del canal. Esta información podría correlacionarse para revelar todas las fuentes y destinos de pago.

- Para escalar la cantidad de pagos que se pueden realizar con Lightning Network. Recuerde que Lightning Network se creó en primer lugar porque notificar a cada participante sobre

cada pago no escala bien. Por lo tanto, Lightning Network no puede diseñarse de manera que comparta actualizaciones de balance de canales entre los participantes.

- Lightning Network es un sistema dinámico. Cambia constantemente y con frecuencia. Se están agregando nodos, se están apagando otros nodos, se cambian los saldos, etc. Incluso si todo se comunica siempre, la información será válida solo por un corto período de tiempo. De hecho, la información a menudo está desactualizada en el momento en que se recibe.

Examinaremos los detalles del protocolo de chismes en un capítulo posterior.

Por ahora, solo es importante saber que el protocolo chisme existe y que se utiliza para compartir información de topología de Lightning Network. Esta información de topología es crucial para entregar pagos a través de la red de canales de pago.

Búsqueda de rutas y enrutamiento

Los pagos en Lightning Network se reenvían a lo largo de una ***ruta*** hecha de canales que vinculan a un participante con otro, desde la fuente de pago hasta el destino del pago. El proceso de encontrar un camino desde el origen hasta el destino se llama búsqueda de caminos . El proceso de usar esa ruta para realizar el pago se llama ***enrutamiento.***

NOTA

Una crítica frecuente a Lightning Network es que el enrutamiento no está resuelto, o incluso que es un problema “irresoluble”. De hecho, el enrutamiento es trivial. Pathfinding, por otro lado, es un problema difícil. Los dos términos a menudo se confunden y deben definirse claramente para identificar qué problema estamos tratando de resolver.

Como veremos a continuación, Lightning Network utiliza actualmente un protocolo ***basado en fuentes*** para la búsqueda de rutas y un protocolo de enrutamiento de ***cebolla*** para enrutar los pagos. Basado en la fuente significa que el remitente del pago tiene que encontrar una ruta

a través de la red hasta el destino deseado. Enrutado cebolla significa que los elementos de la ruta están en capas (como una cebolla), con cada capa cifrada para que solo pueda ser vista por un nodo a la vez. Discutiremos el enrutamiento de cebolla en la siguiente sección.

Pathfinding basado en fuentes

Si supiéramos los saldos de canal exactos de cada canal, podríamos calcular fácilmente una ruta de pago utilizando cualquiera de los algoritmos de búsqueda de ruta estándar que se enseñan en cualquier clase de informática. Esto incluso podría resolverse de una manera que optimice las tarifas pagadas a los nodos por reenviar el pago.

Sin embargo, la información de saldo de todos los canales no es y no puede ser conocida por todos los participantes de la red. Necesitamos estrategias pioneras más innovadoras.

Con solo información parcial sobre la topología de la red, la búsqueda de rutas es un verdadero desafío, y aún se está realizando una investigación activa en esta parte de Lightning Network. El hecho de que el problema de la búsqueda de rutas no esté "completamente resuelto" en Lightning Network es un punto importante de crítica hacia la tecnología.

> **NOTA**
>
> Una crítica común de la búsqueda de rutas en Lightning Network es que no tiene solución porque es equivalente al problema del ***viajante*** de comercio (TSP) NP-completo, un problema fundamental en la teoría de la complejidad computacional. De hecho, la búsqueda de rutas en Lightning no es equivalente a TSP y cae en una clase diferente de problemas. Solucionamos con éxito este tipo de problemas (búsqueda de rutas en gráficos con información incompleta) cada vez que le pedimos a Google que nos dé indicaciones para llegar evitando el tráfico. También solucionamos con éxito este problema cada vez que enrutamos un pago en Lightning Network.

La búsqueda de rutas y el enrutamiento se pueden implementar de varias maneras diferentes, y múltiples algoritmos de búsqueda de rutas y enrutamiento pueden coexistir en el Lightning Network, al igual que múltiples algoritmos de enrutamiento y búsqueda de rutas

existen en internet. La búsqueda de rutas basada en la fuente es una de las muchas soluciones posibles y tiene éxito en la escala actual de Lightning Network.

La estrategia de búsqueda de rutas implementada actualmente por los nodos Lightning es probar rutas iterativamente hasta encontrar una que tenga suficiente liquidez para reenviar el pago. Este es un proceso iterativo de prueba y error, hasta que se logra el éxito o no se encuentra ningún camino. Actualmente, el algoritmo no necesariamente da como resultado la ruta con las tarifas más bajas. Si bien esto no es óptimo y ciertamente se puede mejorar, incluso esta estrategia simplista funciona bastante bien.

Este "sondeo" lo realiza el nodo Lightning o la billetera y el usuario no lo ve directamente. Es posible que el usuario solo se dé cuenta de que se está realizando un sondeo si el pago no se completa al instante.

> **NOTA**
>
> En Internet, utilizamos el Protocolo de Internet y un algoritmo de reenvío de IP para reenviar paquetes de Internet desde el remitente hasta el destino. Si bien estos protocolos tienen la buena propiedad de permitir que los hosts de Internet encuentren en colaboración una ruta para el flujo de información a través de Internet, no podemos reutilizar y adoptar este protocolo para reenviar pagos en Lightning Network. A diferencia de Internet, los pagos Lightning deben ser ***atómicos*** y los saldos de los canales deben permanecer ***privados.*** Además, la capacidad del canal en Lightning cambia con frecuencia, a diferencia de Internet, donde la capacidad de conexión es relativamente estática. Estas limitaciones requieren estrategias novedosas.

Por supuesto, la búsqueda de rutas es trivial si queremos pagarle a nuestro socio de canal directo y tenemos suficiente saldo en nuestro lado del canal para hacerlo. En todos los demás casos, nuestro nodo utiliza información del protocolo de chismes para realizar una búsqueda de rutas. Esto incluye canales de pago públicos conocidos actualmente, nodos conocidos, topología conocida (cómo se conectan los nodos conocidos), capacidades de canal conocidas y políticas de tarifas conocidas establecidas por los propietarios de los nodos.

Enrutamiento de cebolla

Lightning Network utiliza un ***protocolo de enrutamiento de cebolla*** similar a la famosa red Tor (The Onion Router). El protocolo de enrutamiento de cebolla utilizado

en Lightning se denomina ***SPHINX Mix Format,*** que se explicará en detalle en un capítulo posterior.[2]

> **NOTA**
>
> El enrutamiento de cebolla de Lightning SPHINX Mix Format solo es similar al enrutamiento de la red Tor en concepto, pero tanto el protocolo como la implementación son completamente diferentes de los utilizados en la red Tor.

Un paquete de pago utilizado para el enrutamiento se llama "cebolla".[3]

Usemos la analogía de la cebolla para seguir un pago enrutado. En su ruta desde el remitente del pago (pagador) hasta el destino del pago (beneficiario), la cebolla pasa de un nodo a otro a lo largo del camino. El remitente construye toda la cebolla, desde el centro hacia afuera. Primero, el remitente crea la información de pago para el destinatario (final) del pago y la cifra con una capa de cifrado que solo el destinatario puede descifrar. Luego, el remitente envuelve esa capa con instrucciones para el nodo en la ruta ***que precede inmediatamente al destinatario final*** y cifra con una capa que solo ese nodo puede descifrar.

Las capas se construyen con instrucciones, trabajando hacia atrás hasta que la ruta completa se codifica en capas. Luego, el remitente entrega la cebolla completa al primer nodo de la ruta, que solo puede leer la capa más externa. Cada nodo pela una capa, encuentra instrucciones adentro que revelan el siguiente nodo en el camino y pasa la cebolla. Como cada nudo pela una capa, no puede leer el resto de la cebolla. Todo lo que sabe es de dónde acaba de llegar la cebolla y hacia dónde se dirige a continuación, sin ninguna indicación sobre quién es el remitente original o el destinatario final.

Esto continúa hasta que la cebolla llega al destino del pago (beneficiario).
Luego, el nodo de destino abre la cebolla y descubre que no hay más capas para descifrar y puede leer la información de pago que contiene.

> **NOTA**
>
> A diferencia de una cebolla real, al pelar cada capa, los nodos agregan un relleno encriptado para mantener el mismo tamaño de la cebolla para el siguiente nodo. Como veremos, esto hace imposible que cualquiera de los nodos intermedios sepa algo sobre el tamaño (longitud) de la ruta, cuántos nodos están involucrados en el enrutamiento, cuántos nodos los precedieron o cuántos los siguieron. Esto aumenta la privacidad al evitar ataques de análisis de tráfico triviales.

El protocolo de enrutamiento de cebolla utilizado en Lightning tiene las siguientes propiedades:

- Un nodo intermediario solo puede ver en qué canal recibió una cebolla y en qué canal reenviar la cebolla. Esto significa que ningún nodo de enrutamiento puede saber quién inició el pago ya quién está destinado el pago. Esta es la propiedad más importante, lo que se traduce en un alto grado de privacidad.

- Las cebollas son lo suficientemente pequeñas como para caber en un solo paquete TCP/IP e incluso en un marco de capa de enlace (por ejemplo, Ethernet). Esto hace que el análisis del tráfico de los pagos sea significativamente más difícil, aumentando aún más la privacidad.

- Las cebollas se construyen de manera que siempre tendrán la misma longitud independientemente de la posición del nodo de procesamiento a lo largo de la ruta. A medida que se "pela" cada capa, la cebolla se rellena con datos "basura" encriptados para mantener el mismo tamaño de la cebolla. Esto evita que los nodos intermediarios conozcan su posición en el camino.

- Las cebollas tienen un HMAC (código de autenticación de mensajes basado en hash) en cada capa para que las manipulaciones de cebollas se eviten y sean prácticamente imposibles.

- Las cebollas pueden tener hasta alrededor de 26 lúpulos o capas de cebolla si lo prefiere. Esto permite caminos suficientemente largos. La longitud precisa de la ruta disponible depende de la cantidad de bytes asignados a la carga útil de enrutamiento en cada salto.

- El cifrado de la cebolla para cada salto utiliza diferentes claves de cifrado efímeras. Si una clave (en particular, la clave privada de un nodo) se filtra en algún momento, un atacante no puede descifrarla.
  En términos más simples, las claves nunca se reutilizan para lograr más seguridad.

- Los errores se pueden enviar de vuelta desde el nodo erróneo al remitente original, utilizando el mismo protocolo de enrutamiento de cebolla. Las cebollas de error son indistinguibles de las cebollas de enrutamiento a observadores externos y nodos intermediarios. El enrutamiento de errores habilita el método de "prueba" de prueba y error que se utiliza para encontrar una ruta que tenga suficiente capacidad para enrutar un pago con éxito.

El enrutamiento de cebolla se examinará en detalle en el Capítulo 10.

Algoritmo de reenvío de pagos

Una vez que el remitente de un pago encuentra una ruta posible a través de la red y construye una cebolla, cada nodo de la ruta reenvía el pago.
Cada nodo procesa una capa de la cebolla y la reenvía al siguiente nodo de la ruta.

Cada nodo intermediario recibe un mensaje Lightning llamado update_add_htlc
con un hash de pago y una cebolla. El nodo intermediario ejecuta una serie de pasos, llamados ***algoritmo de reenvío de pagos:***

1. El nodo descifra la capa externa de la cebolla y verifica la integridad del mensaje.

2. Confirma que puede cumplir con las sugerencias de enrutamiento, según el canal. tarifas y capacidad disponible en el canal de salida.

3. Trabaja con su socio de canal en el canal entrante para actualizar el estado del canal.

4. Agrega algo de relleno al final de la cebolla para mantenerla en una longitud constante ya que eliminó algunos datos desde el principio.

5. Sigue las sugerencias de enrutamiento para reenviar el paquete de cebolla modificado en su canal de pago saliente enviando también un mensaje update_add_htlc que incluye el mismo hash de pago y la cebolla.

6. Trabaja con su socio de canal en el canal saliente para actualizar el estado del canal.

Por supuesto, estos pasos se interrumpen y cancelan si se detecta un error, y se envía un mensaje de error al autor del mensaje update_add_htlc. El mensaje de error también se formatea como una cebolla y se envía hacia atrás en el canal entrante.

A medida que el error se propaga hacia atrás en cada canal a lo largo de la ruta, los socios del canal eliminan el pago pendiente y retroceden el pago en la forma opuesta a la que comenzó.

Si bien la probabilidad de falla en el pago es alta si no se liquida rápidamente, un nodo nunca debe iniciar otro intento de pago a lo largo de una ruta diferente antes de que la cebolla regrese con un error. El remitente pagaría dos veces si ambos intentos de pago finalmente tuvieran éxito.

Cifrado de comunicación punto a punto

El protocolo LN es principalmente un protocolo peer-to-peer entre sus participantes. Como vimos en secciones anteriores, existen dos funciones superpuestas en la red, formando dos redes lógicas que juntas son ***la Lightning Network:***

1. Una amplia red peer-to-peer que utiliza un protocolo de chismes para propagar información de topología, donde los pares se conectan aleatoriamente entre sí. Los pares no necesariamente tienen canales de pago entre ellos, por lo que no siempre son socios de canal.

2. Una red de canales de pago entre socios de canal. Los socios de canal también chismean sobre la topología, lo que significa que son nodos pares en el protocolo de chismes.

Toda la comunicación entre pares se envía a través de mensajes llamados ***mensajes Lightning.*** Todos estos mensajes están encriptados, utilizando un marco de comunicaciones criptográficas llamado ***Marco de Protocolo de Ruido.*** los Noise Protocol Framework permite la construcción de protocolos de comunicación criptográfica que ofrecen autenticación, encriptación, secreto de reenvío y privacidad de identidad. El marco de protocolo de ruido también se utiliza en varios sistemas populares de comunicaciones cifradas de extremo a extremo, como WhatsApp, WireGuard e I2P. Se puede encontrar más información en el sitio web de Noise Protocol Framework.

El uso de Noise Protocol Framework en Lightning Network garantiza que todos los mensajes en la red estén autenticados y encriptados, lo que aumenta la privacidad de la red y su resistencia al análisis de tráfico, la inspección profunda de paquetes y las escuchas. Sin embargo, como efecto secundario, esto hace que el desarrollo y la prueba del protocolo sean un poco complicados porque uno no puede simplemente observar la red con una herramienta de captura de paquetes o análisis de red como Wireshark. En su lugar, los desarrolladores tienen que usar complementos especializados que descifren el protocolo desde la perspectiva de un nodo, como el ***disector de rayos ,*** un complemento de Wireshark.

Pensamientos sobre la confianza

Siempre que una persona siga el protocolo y tenga su nodo asegurado, no existe un riesgo importante de perder fondos al participar en Lightning Network.
Sin embargo, existe el costo de pagar tarifas en cadena al abrir un canal.
Cualquier costo debe venir con un beneficio correspondiente. En nuestro caso, la recompensa para Alice por asumir el costo de abrir un canal es que Alice puede enviar y, después de mover algunas de las monedas al otro extremo del canal, recibir pagos de bitcoin en Lightning Network en cualquier momento, y que puede ganar tarifas en bitcoins al reenviar pagos para otras personas. Alice sabe que, en teoría, Bob puede cerrar el canal inmediatamente después de abrirlo, lo que genera tarifas de cierre en la cadena para Alice. Alice necesitará tener un poco de confianza en Bob. Alice ha estado en Bob's Cafe y claramente Bob está interesado en vender su café, por lo que Alice puede confiar en Bob en este sentido. Ahí

son beneficios mutuos tanto para Alice como para Bob. Alice decide que la recompensa es suficiente para que ella asuma el costo de la tarifa en cadena por crear un canal para Bob. Por el contrario, Alice no abrirá un canal a alguien desconocido que acaba de enviarle un correo electrónico sin invitación pidiéndole que abra un nuevo canal.

Comparación con Bitcoin

Si bien Lightning Network se basa en Bitcoin y hereda muchas de sus características y propiedades, existen diferencias importantes que los usuarios de ambas redes deben tener en cuenta.

Algunas de estas diferencias son diferencias en la terminología. También hay diferencias arquitectónicas y diferencias en la experiencia del usuario. En las siguientes secciones, examinaremos las diferencias y similitudes, explicaremos la terminología y ajustaremos nuestras expectativas.

Direcciones Versus Facturas, Transacciones Versus Pagos

En un pago típico con Bitcoin, un usuario recibe una dirección de Bitcoin (por ejemplo, escaneando un código QR en una página web o recibiéndolo en un mensaje instantáneo o correo electrónico de un amigo). Luego usan su billetera Bitcoin para crear una transacción para enviar fondos a esta dirección.

En Lightning Network, el destinatario de un pago crea una factura. Una factura Lightning puede verse como análoga a una dirección de Bitcoin. El destinatario previsto entrega la factura Lightning al remitente como un código QR o una cadena de caracteres, como una dirección de Bitcoin.

El remitente usa su billetera Lightning para pagar la factura, copiando el texto de la factura o escaneando el código QR de la factura. Un pago Lightning es similar a una "transacción" de Bitcoin.

Sin embargo, existen algunas diferencias en la experiencia del usuario. Una dirección de Bitcoin es ***reutilizable.*** Las direcciones de Bitcoin nunca caducan, y si el propietario de la dirección aún tiene las claves, siempre se puede acceder a los fondos que contiene. A

el remitente puede enviar cualquier cantidad de bitcoin a una dirección utilizada anteriormente, y el destinatario puede publicar una sola dirección estática para recibir muchos pagos. Si bien esto va en contra de las mejores prácticas por razones de privacidad, es técnicamente posible y, de hecho, bastante común.

En Lightning, sin embargo, cada factura solo se puede usar una vez para un monto de pago específico. No puede pagar más o menos, no puede usar una factura nuevamente y la factura tiene un tiempo de vencimiento incorporado. En Lightning, un destinatario debe generar una nueva factura para cada pago, especificando el monto del pago por adelantado. Hay una excepción a esto, un mecanismo llamado ***envío de claves,*** que examinaremos en "Pagos espontáneos de envío de claves".

Seleccionar salidas versus encontrar una ruta

Para realizar un pago en la red Bitcoin, un remitente debe consumir una o más salidas de transacciones no gastadas (UTXO). Si un usuario tiene múltiples UTXO, él (o más bien su billetera) deberá seleccionar qué UTXO (s) enviar. Por ejemplo, un usuario que realiza un pago de 1 BTC puede usar una única salida con un valor de 1 BTC, dos salidas con un valor de 0,25 BTC y 0,75 BTC, o cuatro salidas con un valor de 0,25 BTC cada una.

En Lightning, los pagos no requieren que se consuman insumos. En cambio, cada pago da como resultado una actualización del saldo del canal, redistribuyéndolo entre los dos socios del canal. El remitente experimenta esto como "mover" el saldo del canal de su extremo de un canal al otro extremo, a su socio de canal. Los pagos relámpago utilizan una serie de canales para enrutar del remitente al destinatario. Cada uno de estos canales debe tener capacidad suficiente para enrutar el pago.

Debido a que se pueden usar muchos canales y rutas posibles para realizar un pago, la elección de canales y rutas del usuario de Lightning es algo análoga a la elección de UTXO del usuario de Bitcoin.

Con tecnologías como los pagos atómicos multitrayecto (AMP) y los pagos multiparte (MPP), que revisaremos en capítulos posteriores, varios
Las rutas de rayos se pueden agregar en un solo pago atómico, al igual que

varios UTXO de Bitcoin se pueden agregar en una sola transacción atómica de Bitcoin.

Cambiar salidas en Bitcoin versus No cambiar en Relámpago

Para realizar un pago en la red Bitcoin, el remitente debe consumir una o más salidas de transacciones no gastadas (UTXO). Los UTXO solo se pueden gastar en su totalidad; no se pueden dividir y gastar parcialmente. Entonces, si un usuario desea gastar 0,8 BTC, pero solo tiene 1 BTC UTXO, debe gastar el 1 BTC UTXO completo enviando 0,8 BTC al destinatario y 0,2 BTC de vuelta a sí mismo como cambio. El pago de cambio de 0.2 BTC crea un nuevo UTXO llamado "salida de cambio".

En Lightning, la transacción de financiación gasta algo de Bitcoin UTXO, creando un UTXO de múltiples firmas para abrir el canal. Una vez que el bitcoin está bloqueado dentro de ese canal, partes del mismo pueden enviarse de un lado a otro dentro del canal, sin necesidad de crear ningún cambio. Esto se debe a que los socios del canal simplemente actualizan el saldo del canal y solo crean un nuevo UTXO cuando el canal finalmente se cierra mediante la transacción de cierre del canal.

Tarifas de minería versus tarifas de enrutamiento

En la red Bitcoin, los usuarios pagan tarifas a los mineros para que sus transacciones se incluyan en un bloque. Estas tarifas se pagan al minero que extrae ese bloque en particular. El monto de la tarifa se basa en el ***tamaño*** de la transacción en ***bytes*** que la transacción está utilizando en un bloque, así como en la rapidez con la que el usuario desea que se extraiga esa transacción. Debido a que los mineros generalmente extraen primero las transacciones más rentables, un usuario que desea que su transacción sea extraída inmediatamente pagará una tarifa ***más alta*** por byte, mientras que un usuario que no tiene prisa pagará una tarifa ***más baja*** por byte.

En Lightning Network, los usuarios pagan tarifas a otros usuarios (nodos intermediarios) para enrutar los pagos a través de sus canales. Para enrutar un pago, un nodo intermediario deberá mover fondos en dos o más canales de su propiedad, así como transmitir los datos para el pago del remitente. Típicamente, el

El usuario de enrutamiento cobrará al remitente en función del ***valor*** del pago, habiendo establecido una ***tarifa base*** mínima (una tarifa plana para cada pago) y una ***tasa de tarifa*** (una tarifa prorrateada proporcional al valor del pago). Por lo tanto, los pagos de mayor valor costarán más para enrutar, y se forma un mercado de liquidez, donde diferentes usuarios cobran diferentes tarifas por el enrutamiento a través de sus canales.

Tarifas variables según el tráfico versus el anunciado Tarifa

En la red de Bitcoin, los mineros buscan ganancias y, por lo general, incluirán tantas transacciones en un bloque como sea posible, mientras se mantienen dentro de la capacidad del bloque llamada ***peso del bloque.***

Si hay más transacciones en la cola (llamadas ***mempool)*** de las que caben en un bloque, comenzarán minando las transacciones que pagan las tarifas más altas por unidad (bytes) de ***peso de transacción.*** Así, cuando hay muchas transacciones en la cola, los usuarios tienen que pagar una tarifa más alta para ser incluidos en el siguiente bloque, o tienen que esperar hasta que haya menos transacciones en la cola. Naturalmente, esto conduce a la aparición de un mercado de tarifas en el que los usuarios pagan en función de la urgencia con la que necesitan que su transacción se incluya en el siguiente bloque.

El recurso escaso en la red Bitcoin es el espacio en los bloques.
Los usuarios de Bitcoin compiten por el espacio en bloque, y el mercado de tarifas de Bitcoin se basa en el espacio en bloque disponible. Los recursos escasos en Lightning Network son la ***liquidez del canal*** (capacidad de fondos disponibles para el enrutamiento en los canales) y la ***conectividad del canal*** (a cuántos nodos bien conectados pueden llegar los canales). Los usuarios de Lightning compiten por capacidad y conectividad; por lo tanto, el mercado de tarifas Lightning está impulsado por la capacidad y la conectividad.

En Lightning Network, los usuarios pagan tarifas a los usuarios que enrutan sus pagos. Enrutar un pago, en términos económicos, no es más que proporcionar y asignar capacidad al remitente. Naturalmente, los enrutadores que cobran tarifas más bajas por la misma capacidad serán más atractivos para enrutar. Por lo tanto, existe un mercado de tarifas donde los enrutadores compiten con

entre sí sobre las tarifas que cobran para enrutar los pagos a través de sus canales.

Transacciones públicas de Bitcoin versus relámpagos privados Pagos

En la red de Bitcoin, cada transacción es públicamente visible en la cadena de bloques de Bitcoin. Si bien las direcciones involucradas son seudónimas y normalmente no están vinculadas a una identidad, todos los demás usuarios de la red las ven y las validan. Además, las empresas de vigilancia de blockchain recopilan y analizan estos datos en masa y los venden a partes interesadas, como empresas privadas, gobiernos y agencias de inteligencia.

Los pagos de LN, por otro lado, son casi completamente privados. Por lo general, solo el remitente y el destinatario conocen completamente el origen, el destino y el monto de la transacción en un pago en particular. Además, es posible que el receptor ni siquiera conozca la fuente del pago. Debido a que los pagos se enrutan cebolla, los usuarios que enrutan el pago solo conocen el monto del pago y no pueden determinar ni el origen ni el destino.

En resumen, las transacciones de Bitcoin se transmiten públicamente y se almacenan para siempre. Los pagos relámpago se ejecutan entre unos pocos pares seleccionados, y la información sobre ellos se almacena de forma privada solo hasta que se cierra el canal.
Crear herramientas de análisis y vigilancia masiva equivalentes a las que se utilizan en Bitcoin será mucho más difícil para Lightning.

Espera de confirmaciones frente a liquidación instantánea

En la red Bitcoin, las transacciones solo se liquidan una vez que se han incluido en un bloque, en cuyo caso se dice que están "confirmadas" en ese bloque. A medida que se extraen más bloques, la transacción adquiere más "confirmaciones" y se considera más segura.

En Lightning Network, las confirmaciones solo importan para abrir y cerrar canales en cadena. Una vez que una transacción de financiación ha alcanzado un número adecuado de confirmaciones (por ejemplo, 3), los socios de canal consideran la

canal abierto Debido a que el bitcoin en el canal está asegurado por un contrato inteligente que administra ese canal, los pagos se liquidan ***instantáneamente*** una vez que el destinatario final los recibe. En términos prácticos, la liquidación instantánea significa que los pagos tardan solo unos segundos en ejecutarse y liquidarse. Al igual que con Bitcoin, los pagos Lightning no son reversibles.

Finalmente, cuando se cierra el canal, se realiza una transacción en la red Bitcoin; una vez que se confirma esa transacción, el canal se considera cerrado.

Envío de cantidades arbitrarias frente a restricciones de capacidad

En la red Bitcoin, un usuario puede enviar cualquier cantidad de bitcoin que posea a otro usuario, sin restricciones de capacidad. En teoría, una sola transacción puede enviar hasta 21 millones de bitcoins como pago.

En Lightning Network, un usuario solo puede enviar la cantidad de bitcoins que existe actualmente en su lado de un canal en particular a un socio de canal. Por ejemplo, si un usuario posee un canal con 0,4 BTC de su lado y otro canal con 0,2 BTC de su lado, entonces el máximo que puede enviar con un pago es 0,4 BTC. Esto es cierto independientemente de la cantidad de bitcoin que el usuario tenga actualmente en su billetera de Bitcoin.

Pagos de varias partes (MPP) es una función que, en el ejemplo anterior, permite al usuario combinar sus canales de 0,4 BTC y 0,2 BTC para enviar un máximo de 0,6 BTC con un pago. Los MPP se están probando actualmente en Lightning Network y se espera que estén ampliamente disponibles y utilizados para cuando se complete este libro. Para obtener más detalles sobre MPP, consulte "Pagos de varias partes".

Si se enruta el pago, cada nodo de enrutamiento a lo largo de la ruta de enrutamiento debe tener canales con una capacidad al menos igual a la cantidad del pago que se enruta. Esto debe ser válido para todos los canales por los que se enruta el pago. La capacidad del canal de menor capacidad en una ruta establece el límite superior para la capacidad de toda la ruta.

Por lo tanto, la capacidad y la conectividad son recursos críticos y escasos en Lightning Network.

Incentivos para pagos de gran valor frente a pagos de pequeño valor Pagos

La estructura de tarifas en Bitcoin es independiente del valor de la transacción. Una transacción de $ 1 millón tiene la misma tarifa que una transacción de $ 1 en Bitcoin, suponiendo un tamaño de transacción similar, en bytes (más específicamente, bytes "virtuales" después de SegWit [Protocolo de testigo segregado]). En Lightning, la tarifa es una tarifa base fija más un porcentaje del valor de la transacción. Por lo tanto, en Lightning, la tarifa de pago aumenta con el valor del pago. Estas estructuras de tarifas opuestas crean diferentes incentivos y conducen a diferentes usos con respecto al valor de transacción. Una transacción de mayor valor será más barata en Bitcoin; por lo tanto, los usuarios preferirán Bitcoin para transacciones de gran valor.
Del mismo modo, en el otro extremo de la escala, los usuarios preferirán Lightning para transacciones de pequeño valor.

Usando Blockchain como un libro mayor versus como un tribunal Sistema

En la red de Bitcoin, cada transacción finalmente se registra en un bloque en la cadena de bloques. Por lo tanto, la cadena de bloques forma un historial completo de cada transacción desde la creación de Bitcoin y una forma de auditar completamente cada bitcoin existente. Una vez que una transacción se incluye en la cadena de bloques, es definitiva. Por lo tanto, no pueden surgir disputas y no es ambiguo cuánto bitcoin está controlado por una dirección particular en un punto particular de la cadena de bloques.

En Lightning Network, el saldo en un canal en un momento determinado solo lo conocen los dos socios del canal, y solo se hace visible para el resto de la red cuando el canal está cerrado. Cuando se cierra el canal, el saldo final del canal se envía a la cadena de bloques de Bitcoin y cada socio recibe su parte de bitcoin en ese canal. Por ejemplo, si el saldo inicial fue de 1 BTC pagado por Alice, y Alice realizó un pago de 0,3 BTC a Bob, entonces el saldo final del canal es de 0,7 BTC para Alice.

y 0,3 BTC para Bob. Si Alice intenta hacer trampa al enviar el estado de apertura del canal a la cadena de bloques de Bitcoin, con 1 BTC para Alice y 0 BTC para Bob, entonces Bob puede tomar represalias al enviar el verdadero estado final del canal, así como crear una transacción de penalización. eso le da todo el bitcoin en el canal. Para Lightning Network, la cadena de bloques de Bitcoin actúa como un sistema judicial. Como un juez robótico, Bitcoin registra los saldos inicial y final de cada canal y aprueba sanciones si una de las partes intenta hacer trampa.

Fuera de línea versus en línea, asíncrono versus Sincrónico

Cuando un usuario de Bitcoin envía fondos a una dirección de destino, no necesita saber nada sobre el destinatario. El destinatario puede estar desconectado o en línea, y no se necesita interacción entre el remitente y el destinatario. La interacción es entre el remitente y la cadena de bloques de Bitcoin. Recibir bitcoin en la cadena de bloques de Bitcoin es una actividad ***pasiva*** y ***asíncrona*** que no requiere ninguna interacción por parte del destinatario o que el destinatario esté en línea en cualquier momento. Las direcciones de Bitcoin pueden incluso generarse fuera de línea y nunca se "registran" en la red de Bitcoin. Solo gastar bitcoin requiere interacción.

En Lightning, el destinatario debe estar en línea para completar el pago antes de que caduque. El destinatario debe ejecutar un nodo o tener a alguien que ejecute un nodo en su nombre (un custodio externo). En concreto, ambos nodos, el del remitente y el del destinatario, deben estar en línea en el momento del pago y deben coordinarse. Recibir un pago Lightning es una ***actividad activa*** y ***síncrona*** entre el remitente y el destinatario, sin la participación de la mayor parte de Lightning Network o la red Bitcoin (excepto los nodos de enrutamiento intermediarios, si los hay).

La naturaleza síncrona y siempre en línea de Lightning Network es probablemente la mayor diferencia en la experiencia del usuario, y esto a menudo confunde a los usuarios que están acostumbrados a Bitcoin.

Satoshis contra Millisatoshes

En la red Bitcoin, la cantidad más pequeña es un ***satoshi,*** que no se puede dividir más. Lightning es un poco más flexible y los nodos Lightning funcionan con ***millisatoshis*** (milésimas de satoshi). Esto permite enviar pequeños pagos a través de Lightning. Se puede enviar un solo pago de millisatoshi a través de un canal de pago, una cantidad tan pequeña que debería caracterizarse adecuadamente como un ***nanopago.***

La unidad millisatoshi no puede, por supuesto, establecerse en la cadena de bloques de Bitcoin con esa granularidad. Tras el cierre del canal, los saldos se redondean al satoshi más cercano. Pero durante la vida útil de un canal, son posibles millones de nanopagos a niveles de millisatoshi. Lightning Network rompe la barrera de los micropagos.

Puntos en común de Bitcoin y Lightning

Si bien Lightning Network se diferencia de Bitcoin en varios aspectos, incluso en la arquitectura y la experiencia del usuario, está construido a partir de Bitcoin y conserva muchas de las características principales de Bitcoin.

Unidad monetaria

Tanto la red Bitcoin como Lightning Network utilizan las mismas unidades monetarias: bitcoin. Los pagos relámpago usan el mismo bitcoin que las transacciones de Bitcoin. Como implicación, debido a que la unidad monetaria es la misma, el límite monetario es el mismo: menos de 21 millones de bitcoin. De los 21 millones de bitcoins totales de Bitcoin, algunos ya están asignados a 2 de 2 direcciones de firmas múltiples como parte de los canales de pago en Lightning Network.

Irreversibilidad y firmeza de los pagos

Tanto las transacciones de Bitcoin como los pagos Lightning son irreversibles e inmutables. No hay operación de "deshacer" o "contracargo" para ninguno de los sistemas.

Como remitente de cualquiera de los dos, debe actuar con responsabilidad, pero también, como destinatario, tiene garantizada la finalidad de sus transacciones.

Riesgo de confianza y contraparte

Al igual que con Bitcoin, Lightning requiere que el usuario solo confíe en las matemáticas, el cifrado y que el software no tenga errores críticos. Ni Bitcoin ni Lightning requieren que el usuario confíe en una persona, empresa, institución o gobierno. Debido a que Lightning se encuentra sobre Bitcoin y se basa en Bitcoin como su capa base subyacente, está claro que el modelo de seguridad de Lightning se reduce a la seguridad de Bitcoin. Esto significa que Lightning ofrece en términos generales la misma seguridad que Bitcoin en la mayoría de las circunstancias, con solo una ligera reducción en la seguridad en algunas circunstancias limitadas.

Operación sin permiso

Tanto Bitcoin como Lightning pueden ser utilizados por cualquier persona con acceso a Internet y al software apropiado, por ejemplo, nodo y billetera. Ninguna red requiere que los usuarios obtengan permiso, verificación o autorización de terceros, empresas, instituciones o un gobierno. Los gobiernos pueden prohibir Bitcoin o Lightning dentro de su jurisdicción, pero no pueden impedir su uso global.

Código abierto y sistema abierto

Tanto Bitcoin como Lightning son sistemas de software de código abierto creados por una comunidad global descentralizada de voluntarios, disponibles bajo licencias abiertas. Ambos se basan en protocolos abiertos e interoperables que funcionan como sistemas abiertos y redes abiertas. Global, abierto y gratuito.

Conclusión

En este capítulo, analizamos cómo funciona realmente Lightning Network y todos los componentes que la componen. Examinamos cada paso en la construcción,

operar y cerrar un canal. Observamos cómo se enrutan los pagos y, finalmente, comparamos Lightning con Bitcoin y analizamos sus diferencias y puntos en común.

En los próximos capítulos revisaremos todos estos temas, pero con mucho más detalle.

---

1 Si bien el documento técnico original de Lightning describía los canales financiados por ambos socios de canal, la especificación actual, a partir de 2020, supone que solo un socio asigna fondos al canal. A partir de mayo de 2021, los canales Lightning de doble financiación son experimentales en la implementación de c-Lightning LN.

2 George Danezis e Ian Goldberg, “Sphinx: A Compact and Provably Secure Mix Format”, en ***Simposio IEEE sobre seguridad y privacidad*** (Nueva York: IEEE, 2009), 269–282.

3 El término “cebolla” fue utilizado originalmente por el proyecto Tor. Además, la red Tor también es llamó la red Onion y el proyecto usa una cebolla como su logo. El nombre de dominio de nivel superior utilizado por los servicios Tor en Internet es ***cebolla.***

Capítulo 4. Software de nodo Lightning

Como hemos visto en capítulos anteriores, un nodo Lightning es un sistema informático que participa en Lightning Network. Lightning Network no es un producto o una empresa; es un conjunto de estándares abiertos que definen una línea de base para la interoperabilidad. Como tal, el software Lightning node ha sido creado por una variedad de empresas y grupos comunitarios. La gran mayoría del software Lightning es ***de código abierto,*** lo que significa que el código fuente está abierto y tiene una licencia que permite la colaboración, el intercambio y la participación de la comunidad en el proceso de desarrollo. De manera similar, las implementaciones de nodos Lightning que presentaremos en este capítulo son todas de código abierto y se desarrollan en colaboración.

A diferencia de Bitcoin, donde el estándar está definido por una ***implementación de referencia*** en el software (Bitcoin Core), en Lightning el estándar está definido por una serie de documentos de estándares llamados ***Basis of Lightning Technology (BOLT),*** que se encuentran en el repositorio ***lightning-rfc .***

No existe una implementación de referencia de Lightning Network, pero hay varias implementaciones interoperables, compatibles con BOLT y competidoras, desarrolladas por diferentes equipos y organizaciones. Los equipos que desarrollan software para Lightning Network también contribuyen en el desarrollo y evolución de los estándares BOLT.

Otra diferencia importante entre el software del nodo Lightning y el software del nodo Bitcoin es que los nodos Lightning no necesitan operar al unísono con las reglas de consenso y pueden tener una funcionalidad extendida más allá de la línea de base de los BOLT. Por lo tanto, diferentes equipos pueden buscar varias características experimentales que, si tienen éxito y se implementan ampliamente, pueden convertirse en parte de los BOLT más adelante.

En este capítulo, aprenderá a configurar cada uno de los paquetes de software para las implementaciones de nodos Lightning más populares. Los hemos presentado en orden alfabético para enfatizar que generalmente no preferimos ni respaldamos uno sobre el otro. Cada uno tiene sus fortalezas y debilidades, y elegir uno dependerá de una variedad de factores. Dado que se desarrollan en diferentes lenguajes de programación (p. ej., Go, C, etc.), su elección también puede depender de su nivel de familiaridad y experiencia con un lenguaje específico y un conjunto de herramientas de desarrollo.

Entorno de desarrollo Lightning

Si es un desarrollador, querrá configurar un entorno de desarrollo con todas las herramientas, bibliotecas y software de soporte para escribir y ejecutar el software Lightning. En este capítulo altamente técnico, recorreremos ese proceso paso a paso. Si el material se vuelve demasiado denso o si no está configurando un entorno de desarrollo, no dude en pasar al siguiente capítulo, que es menos técnico.

Uso de la línea de comandos

Los ejemplos de este capítulo, y más ampliamente en la mayor parte de este libro, utilizan una terminal de línea de comandos. Eso significa que escribe comandos en una terminal y recibe respuestas de texto. Además, los ejemplos se demuestran en un sistema operativo basado en el kernel de Linux y el sistema de software GNU, específicamente la última versión estable a largo plazo de Ubuntu (Ubuntu 20.04 LTS). La mayoría de los ejemplos se pueden replicar en otros sistemas operativos como Windows o macOS, con pequeñas modificaciones en los comandos. La mayor diferencia entre los sistemas operativos es el ***administrador de paquetes*** que instala las diversas bibliotecas de software y sus requisitos previos. En los ejemplos dados, usaremos apt, que es el administrador de paquetes para Ubuntu. En macOS, un administrador de paquetes común utilizado para el desarrollo de código abierto es Homebrew, al que se accede con el comando brew.

En la mayoría de los ejemplos aquí, construiremos el software directamente desde el código fuente. Si bien esto puede ser bastante desafiante, nos brinda más poder y control. ¡Puede elegir usar contenedores Docker, paquetes precompilados u otros mecanismos de instalación si se queda atascado!

PROPINA

En muchos de los ejemplos de este capítulo, utilizaremos la interfaz de línea de comandos del sistema operativo (también conocida como ***shell),*** a la que se accede a través de una aplicación de ***terminal .*** El shell primero mostrará un mensaje como indicador de que está listo para su comando. Luego, escribe un comando y presiona la tecla Intro, a lo que el shell responde con un texto y un nuevo mensaje para su próximo comando. El indicador puede verse diferente en su sistema, pero en los siguientes ejemplos se indica con un símbolo $. En los ejemplos, cuando vea texto después de un símbolo $, no escriba el símbolo $ sino el comando que le sigue inmediatamente. Luego presione la tecla Enter para ejecutar el comando. En los ejemplos, las líneas que siguen a cada comando son las respuestas del sistema operativo a ese comando. Cuando vea el siguiente prefijo $, sabrá que es un comando nuevo y debe repetir el proceso.

Para mantener las cosas consistentes, usamos el shell bash en todos los ejemplos de línea de comandos. Mientras que otros shells se comportarán de manera similar y podrá ejecutar todos los ejemplos sin él, algunos de los scripts de shell están escritos específicamente para bash shell y pueden requerir algunos cambios o personalizaciones para ejecutarse en otro shell. Para mantener la coherencia, puede instalar bash shell en Windows y macOS, y viene instalado de forma predeterminada en la mayoría de los sistemas Linux.

Descarga del repositorio de libros

Todos los ejemplos de código están disponibles en el repositorio en línea del libro. Debido a que el repositorio se mantendrá actualizado tanto como sea posible, siempre debe buscar la última versión en el repositorio en línea en lugar de copiarla del libro impreso o del libro electrónico.

Puede descargar el repositorio como un paquete ZIP visitando GitHub y seleccionando el botón verde Código a la derecha.

Alternativamente, puede usar el comando git para crear un clon controlado por versión del repositorio en su computadora local. Git es un sistema de control de versiones distribuido que utilizan la mayoría de los desarrolladores para colaborar en el desarrollo de software y realizar un seguimiento de los cambios en los repositorios de software. Descargue e instale git siguiendo las instrucciones del Proyecto Git.

Para hacer una copia local del repositorio en su computadora, ejecute el comando git de la siguiente manera:

```
$ git clonar https://github.com/lnbook/lnbook.git
```

Ahora tiene una copia completa del repositorio de libros en una carpeta llamada lnbook. Deberá cambiar al directorio recién descargado ejecutando:

```
$ cd lnlibro
```

Todos los ejemplos posteriores supondrán que está ejecutando comandos desde dentro de esta carpeta.

Contenedores Docker

Muchos desarrolladores utilizan un ***contenedor,*** que es un tipo de máquina virtual, para instalar un sistema operativo y aplicaciones preconfiguradas con todas las dependencias necesarias. Gran parte del software Lightning también se puede instalar usando un sistema de contenedores como ***Docker*** que se encuentra en Docker

página de inicio Las instalaciones de contenedores son mucho más fáciles, especialmente para aquellos que no están acostumbrados a un entorno de línea de comandos.

El repositorio del libro contiene una colección de contenedores Docker que se pueden usar para configurar un entorno de desarrollo consistente para practicar y replicar los ejemplos en cualquier sistema. Debido a que el contenedor es un sistema operativo completo que se ejecuta con una configuración consistente, puede estar seguro de que los ejemplos funcionarán en su computadora sin la necesidad de preocuparse por las dependencias, las versiones de la biblioteca o las diferencias en la configuración.

Los contenedores Docker a menudo se optimizan para que sean pequeños, es decir, ocupen el mínimo espacio en disco. Sin embargo, en este libro estamos usando contenedores para ***estandarizar*** el entorno y hacerlo consistente para todos los lectores.
Además, estos contenedores no están destinados a ejecutar servicios en segundo plano. En su lugar, están destinados a ser utilizados para probar los ejemplos y aprender interactuando con el software. Por estas razones, los contenedores son bastante grandes y vienen con muchas herramientas y utilidades de desarrollo.
Comúnmente, la distribución Alpine se usa para contenedores de Linux debido a su tamaño reducido. No obstante, proporcionamos contenedores creados en Ubuntu porque más desarrolladores están familiarizados con Ubuntu, y esta familiaridad es más importante para nosotros que el tamaño.

La instalación y el uso de Docker y sus comandos se detallan en el Apéndice B. Si no está familiarizado con Docker, ahora es un buen momento para revisar rápidamente esa sección.

Puede encontrar las últimas definiciones de contenedores y configuraciones de compilación en el repositorio del libro en la carpeta ***code/ docker .*** Cada contenedor está en una carpeta separada, como se puede ver a continuación:

```
$ árbol -F --charset=código asciii/docker

código/docker |--
 bitcoind/ | |-- bashrc
 |-- bitcoin.conf `--
 dirección_verificación.txt |--
```

```
                                |-- demo_mnemónico.txt
                                `--
        | | demo_privkey.txt
        |-- bitcoind-punto de entrada.sh
        |-- clic
        |-- Dockerfile
        `-- mine.sh*
| | | | | |-- c-rayo/
| |-- bashrc
        |-- clic
| | |-- c-punto-de-entrada-relámpago.sh
| |-- devkeys.pem
| |-- Dockerfile
        |-- fondo-c-relámpago.sh
| |-- relámpago/
| | | | `--
| configuración
logtail.sh
        `-- esperar-a-bitcoind.sh
| |-- relámpago/
        |-- bashrc
        |-- clic
        |-- Dockerfile
        |-- relámpago/
                `-- eclair.conf
        | |-- eclair-punto de entrada.sh
        |-- logtail.sh
        `-- esperar-a-bitcoind.sh
| | | | | | | |-- lnd/
| |-- bashrc
        |-- clic
| | |-- Dockerfile
| |-- fondo-lnd.sh
        |-- lnd/
| | |-| |-- lnd.conf
lnd-punto de entrada.sh
        |-- logtail.sh
| | `-- esperar-a-bitcoind.sh
|-- comprobar-versiones.sh
|-- docker-compose.yml
|-- Archivo MAKE
`--     ejecutar-pago-demo.sh*
```

Como veremos en las próximas secciones, puede construir estos contenedores localmente, o puede extraerlos del repositorio del libro en ***Docker Hub.***

Las siguientes secciones asumirán que ha instalado Docker y está familiarizado con el uso básico del comando docker.

Bitcoin Core y registro

La mayoría de las implementaciones de nodos Lightning necesitan acceso a un nodo Bitcoin completo para funcionar.

Instalar un nodo completo de Bitcoin y sincronizar la cadena de bloques de Bitcoin está fuera del alcance de este libro y es un esfuerzo relativamente complejo en sí mismo. Si quieres probarlo, consulta ***Mastering Bitcoin,*** "Capítulo 3: Bitcoin Core: la implementación de referencia", que analiza la instalación y el funcionamiento de un nodo de Bitcoin.

Un nodo de Bitcoin se puede operar en modo de prueba, donde el nodo crea una cadena de bloques de Bitcoin local simulada para fines de prueba. En los siguientes ejemplos, usaremos el modo de prueba de registro para permitirnos demostrar Lightning sin tener que sincronizar un nodo de Bitcoin o arriesgar fondos.

El contenedor de Bitcoin Core es bitcoind. Está configurado para ejecutar Bitcoin Core en modo de prueba y extraer 6 bloques nuevos cada 10 segundos. Su puerto de llamada a procedimiento remoto (RPC) está expuesto en el puerto 18443 y es accesible para llamadas RPC con el registro de nombre de usuario y el registro de contraseña. También puede acceder a él con un shell interactivo y ejecutar comandos bitcoin-cli localmente.

Construyendo el contenedor de Bitcoin Core

Preparemos el contenedor bitcoind. La forma más fácil es extraer el contenedor más reciente de ***Docker Hub:***

```
$ docker pull lnbook/bitcoind Uso de la
etiqueta predeterminada: Latest Latest:
extracción de lnbook/bitcoind 35807b77a593: extracción
completa e1b85b9c5571: extracción completa [...]
288f1cc78a00: extracción completa Resumen:



sha256:861e7e32c9ad650aa367af40fc5acff894e89e407ae194e89e407ae194
b550e2
Estado: Imagen más nueva descargada para lnbook/bitcoind:latest docker.io/lnbook/
bitcoind:latest
```

Como alternativa, puede crear el contenedor usted mismo a partir de la definición de contenedor local que se encuentra en ***code/ docker/ bitcoind/ Dockerfile.***

> **NOTA**
>
> No necesita compilar el contenedor si utilizó el comando de extracción anteriormente para extraerlo de Docker Hub.

Construir el contenedor localmente usará un poco menos del ancho de banda de su red, pero tomará más tiempo de su CPU para construirlo. Usamos el comando docker build para construirlo:

```
$ cd code/docker $ docker run
-it --name bitcoind lnbook/bitcoind Iniciando bitcoind...

Bitcoin Core iniciando Esperando que
bitcoind comience bitcoind iniciado

==================================================
Clave privada de demostración importada
Dirección de Bitcoin: 2NBKgwSWY5qEmfN2Br4WtMDGuamjpuUc5q1
Clave privada:
cSaejkcWwU25jMweWEewRSsrVQq2FGTij1xjXv4x1XvxVRF1ZCr3
==================================================
==================================================
Saldo: 0.00000000
==================================================
Minería 101 bloques para desbloquear algo de bitcoin [

34c744207fd4dd32b70bac467902bd8d030fba765c9f240a2e98f15f05338964
",

"64d82721c641c378d79b4ff2e17572c109750bea1d4eddbae0b54f51e4cdf23e ",

  [...]

"7a8c53dc9a3408c9ecf9605b253e5f8086d67bbc03ea05819b2c9584196c9294 ",
```

```
"39e61e50e34a9bd1d6eab51940c39dc1ab56c30b21fc28e1a10c14a39b67a1c3 ",

4ca7fe9a55b0b767d2b7f5cf4d51a2346f035fe8c486719c60a46dcbe33de51a
"
]
Minería 6 bloques cada 10 segundos Saldo:
50.00000000
[
"5ce76cc475e40515b67e3c0237d1eef597047a914ba3f59bbd62fc3691849055 ",

1ecb27a05ecfa9dfa82a7b26631e0819b2768fe5e6e56c7a2e1078b078e21e9f
",
"717ceb8b6c329d57947c950dc5668fae65bddb7fa03203984da9d2069e20525b ",

185fc7cf3557a6ebfc4a8cdd1f94a8fa08ed0c057040cdd68bfb7aee2d5be624
",
“59001ae237a3834ebe4f6e6047dcec8fd67df0352ddc70b6b02190f982a60384 ”,

“754c860fe1b9e0e7292e1de96a65ea78047feb4c72dbbde2a1d224faa1499dd
"
]
```

Como puede ver, bitcoind se inicia y extrae 101 bloques simulados para iniciar la cadena. Esto se debe a que, según las reglas de consenso de Bitcoin, el bitcoin recién extraído no se puede gastar hasta que hayan transcurrido 100 bloques. Al minar 101 bloques, hacemos que la base de monedas del primer bloque sea gastable. Después de esa actividad minera inicial, se extraen 6 bloques nuevos cada 10 segundos para mantener la cadena en movimiento.

Por ahora, no hay transacciones. Pero tenemos algunos bitcoins de prueba que se han extraído en la billetera y están disponibles para gastar. Cuando conectemos algunos nodos Lightning a esta cadena, enviaremos algunos bitcoins a sus billeteras para que podamos abrir algunos canales Lightning entre los nodos Lightning.

Interactuando con el contenedor central de bitcoin

Mientras tanto, también podemos interactuar con el contenedor bitcoind al enviándole comandos de shell. El contenedor está enviando un archivo de registro al terminal, que muestra el proceso de minería del proceso bitcoind. A interactuar con el shell podemos emitir comandos en otro terminal, usando el comando exec docker. Dado que anteriormente nombramos la ejecución contenedor con el argumento de nombre, podemos referirnos a él por ese nombre cuando ejecute el comando docker exec. Primero, hagamos una fiesta interactiva caparazón:

```
$ docker exec -it bitcoind /bin/bash
root@e027fd56e31a:/bitcoind# ps x
    PID TTY      COMANDO DE TIEMPO ESTADÍSTICO
      1 pt/0 7 ?  SS+    0:00 /bin/bash /usr/local/bin/mine.sh
                  SSL    0:03 bitcoind -datadir=/bitcoind -daemon
     97 puntos/1 0:00 /bin/bash
    124 puntos/0 0:00 dormir 10
    125 puntos/1 0:00 ps x
root@e027fd56e31a:/bitcoin#
```

Ejecutar el shell interactivo nos coloca “dentro” del contenedor. Se registra como usuario root, como podemos ver en el prefijo root@ en el nuevo indicador de shell root@e027fd56e31a:/bitcoind#. Si emitimos el comando ps x para ver qué procesos se están ejecutando, vemos tanto bitcoind como el script mine.sh se ejecutan en segundo plano. Para salir de este shell, presione Ctrl-D o escriba **exit** y volverá al indicador del sistema operativo.

En lugar de ejecutar un shell interactivo, también podemos emitir un solo comando que se ejecuta dentro del contenedor. Para mayor comodidad, el bitcoin-cli El comando tiene un alias "cli" que pasa la configuración correcta. Entonces vamos ejecútelo para preguntarle a Bitcoin Code sobre la cadena de bloques. Ejecutamos cli obtener información de la cadena de bloques:

```
$ docker exec bitcoind cli getblockchaininfo
{
    "cadena": "registro",
    "bloques": 131,
    "encabezados": 131,
    "mejor bloquehash":
2cf57aac35365f52fa5c2e626491df634113b2f1e5197c478d57378e5a146110
```

```
",
 [...]
    "advertencias": ""
}
```

El comando cli en el contenedor bitcoind nos permite emitir comandos RPC al nodo Bitcoin Core y obtener resultados codificados con notación de objetos JavaScript (JSON).

Además, todos nuestros contenedores Docker tienen un codificador/descodificador JSON de línea de comandos llamado jq preinstalado. jq nos ayuda a procesar datos con formato JSON a través de la línea de comandos o desde scripts internos. Puede enviar la salida JSON de cualquier comando a jq usando el | personaje. Este carácter, así como esta operación, se denomina "tubería". Apliquemos una tubería y jq al comando anterior de la siguiente manera:

```
$ docker exec bitcoind bash -c "cli getblockchaininfo | jq .blocks"
197
```

jq .blocks le indica al decodificador jq JSON que extraiga los bloques de campo del resultado de getblockchaininfo. En nuestro caso, extrae e imprime el valor de 197 que podríamos utilizar en un comando posterior.

Como verás en las siguientes secciones, podemos ejecutar varios contenedores al mismo tiempo y luego interactuar con ellos individualmente. Podemos emitir comandos para extraer información como la clave pública del nodo Lightning o para realizar acciones como abrir un canal Lightning a otro nodo. Los comandos docker run y docker exec, junto con jq para la decodificación JSON, son todo lo que necesitamos para construir una Lightning Network funcional que combine muchas implementaciones de nodos diferentes. Esto nos permite realizar diversos experimentos en nuestro propio ordenador.

El proyecto del nodo de relámpagos c-lightning

c-lightning es una implementación liviana, altamente personalizable y compatible con los estándares del protocolo LN, desarrollada por Blockstream como parte del Proyecto Elements. El proyecto es de código abierto y desarrollado de forma colaborativa en GitHub.

En las siguientes secciones, crearemos un contenedor Docker que ejecuta un nodo Lightning ac que se conecta al contenedor bitcoind que creamos anteriormente. También le mostraremos cómo configurar y compilar el software c lightning directamente desde el código fuente.

Creación de c-lightning como un contenedor Docker

La distribución de software c-lightning tiene un contenedor Docker, pero está diseñado para ejecutar c-lightning en sistemas de producción y junto con un nodo bitcoind. Usaremos un contenedor algo más simple configurado para ejecutar c-lightning con fines de demostración.

Extraigamos el contenedor c-lightning del repositorio Docker Hub del libro:

```
$ docker pull lnbook/c-relámpago
Uso de la etiqueta predeterminada:
último último: Extracción de lnbook/c-lightning

 [...]

Resumen:
sha256:bdefcefe8a9712e7b3a236dcc5ab12d999c46fd280e209712e7cb649b8
bf0688
Estado: Imagen descargada para lnbook/c-lightning:latest docker.io/lnbook/c-lightning:latest
```

Como alternativa, podemos crear el contenedor Docker c-lightning a partir de los archivos del libro que descargó previamente en un directorio llamado lnbook. Como antes, usaremos el comando docker build en el subdirectorio code/docker. Etiquetaremos la imagen del contenedor con la etiqueta lnbook/c-lightning, así:

```
$ cd code/docker $ docker
build -t lnbook/c-lightning c-lightning Envío del contexto de compilación al
demonio Docker 91.14kB Paso 1/34: ARG OS=ubuntu Paso 2/34: ARG
OS_VER=focal Paso 3/34: DESDE ${OS}:${OS_VER} como base del sistema
operativo

 ---> fb52e22af1b0

 [...]

Paso 34/34: CMD ["/usr/local/bin/logtail.sh"]
 ---> Corriendo en 8d3d6c8799c5
Quitar el contenedor intermedio 8d3d6c8799c5
 ---> 30b6fd5d7503
Construido con éxito 30b6fd5d7503
Etiquetado correctamente en lnbook/c-lightning:último
```

Nuestro contenedor ahora está construido y listo para funcionar. Sin embargo, antes de ejecutar el contenedor c lightning, debemos iniciar el contenedor bitcoind en otro terminal porque c-lightning depende de bitcoind. También necesitaremos configurar una red Docker que permita que los contenedores se conecten entre sí como si residieran en la misma red de área local.

PROPINA

Los contenedores Docker pueden "hablar" entre sí a través de una red de área local virtual administrada por el sistema Docker. Cada contenedor puede tener un nombre personalizado y otros contenedores pueden usar ese nombre para resolver su dirección IP y conectarse fácilmente a él.

Configuración de una red Docker

Una vez que se configura una red Docker, Docker activará la red en nuestra computadora local cada vez que se inicie Docker, por ejemplo, después de reiniciar. Por lo tanto, solo necesitamos configurar una red una vez usando el comando docker network create. El nombre de la red en sí no es importante, pero tiene que ser único en nuestro ordenador. De forma predeterminada, Docker tiene tres redes denominadas host, bridge y none. Nombraremos nuestra nueva red lnbook y la crearemos así:

```
$ docker red crear lnbook
ad75c0e4f87e5917823187febedfc0d7978235ae3e88eca63abe7e0b5ee81bfb
$ red docker ls
IDENTIFICACIÓN DE RED NOMBRE CONDUCTOR ALCANCE
7f1fb63877ea anfitrión del anfitrión del local
4e575cba0036 puente puente local
ad75c0e4f87e en el libro puente local
ee8824567c95 ninguna nulo local
```

Como puede ver, ejecutar docker network ls nos da una lista de los Redes acoplables. Nuestra red lnbook ha sido creada. podemos ignorar el ID de la red, porque se gestiona automáticamente.

Ejecución de los contenedores bitcoind y c-lightning

El siguiente paso es iniciar los contenedores bitcoind y c-lightning y conectarlos a la red lnbook. Para ejecutar un contenedor en un determinado network, debemos pasar el argumento network a docker run. Para hacer es fácil que los contenedores se encuentren, también le daremos un nombre a cada uno con el argumento del nombre. Empezamos bitcoind así:

```
$ docker run -it --network lnbook --name bitcoind lnbook/bitcoind
```

Debería ver el inicio de bitcoind y comenzar a extraer bloques cada 10 segundos. Déjelo en ejecución y abra una nueva ventana de terminal para iniciar c lightning. Usamos un comando de ejecución de docker similar con el argumentos de red y nombre para iniciar c-lightning de la siguiente manera:

```
$ docker run -it --network lnbook --name c-relámpago lnbook/c relámpago

Esperando a que comience bitcoind...
Esperando a que bitcoind extraiga bloques...
Iniciando c-relámpago...
2021-09-12T13:14:50.434Z rayo INUSUAL: Creando
directorio de configuración /lightningd/regtest
Inicio completo
Cartera c-lightning de financiación
8a37a183274c52d5a962852ba9f970229ea6246a096ff1e4602b57f7d4202b31
lightningd: archivo de registro abierto /lightningd/lightningd.log
```

```
lightningd: Creando el directorio de configuración /lightningd/regtest lightningd: Archivo de registro abierto /
lightningd/lightningd.log
```

El contenedor c-lightning se inicia y se conecta al contenedor bitcoind a través de la red Docker. Primero, nuestro nodo c-lightning esperará a que se inicie bitcoind, y luego esperará hasta que bitcoind haya extraído algo de bitcoin en su billetera. Finalmente, como parte del inicio del contenedor, un script enviará un comando RPC al nodo bitcoind, que crea una transacción que financia la billetera c-lightning con 10 BTC de prueba. ¡Ahora nuestro nodo c-lightning no solo se está ejecutando, sino que incluso tiene algunos bitcoins de prueba para jugar!

Como demostramos con el contenedor bitcoind, podemos enviar comandos a nuestro contenedor c-lightning en otra terminal para extraer información, abrir canales, etc. El comando que nos permite enviar instrucciones de línea de comandos al nodo c-lightning se llama lightning -cli. Este comando lightning-cli también tiene un alias como cli dentro de este contenedor. Para obtener la información del nodo c-lightning, use el siguiente comando docker exec en otra ventana de terminal:

```
$ docker exec c-relámpago cli getinfo {

    "identificación":
"026ec53cc8940df5fed5fa18f8897719428a15d860ff4cd171fca9530879c749
9e",
    "alias": "IRATEARTIST", "color":
    "026ec5", "num_peers": 0,
    "num_pending_channels": 0,


[...]

    "versión": "0.10.1", "blockheight":
    221, "red": "regtest",
    "msatoshi_fees_collected": 0,
    "fees_collected_msat": "0msat", "lightning-dir":
    "/lightningd/regtest"

}
```

Ahora tenemos nuestro primer nodo Lightning ejecutándose en una red virtual y comunicándose con una cadena de bloques de Bitcoin de prueba. Más adelante en este capítulo iniciaremos más nodos y los conectaremos entre sí para realizar algunos pagos Lightning.

En la siguiente sección, también veremos cómo descargar, configurar y compilar c-lightning directamente desde el código fuente. Este es un paso opcional y avanzado que le enseñará a usar las herramientas de compilación y le permitirá realizar modificaciones en el código fuente de c-lightning. Con este conocimiento, puede escribir código, corregir algunos errores o crear un complemento para c-lightning.

> **NOTA**
>
> Si no planea sumergirse en el código fuente o la programación de un nodo Lightning, puede omitir la siguiente sección por completo. El contenedor Docker que acabamos de crear es suficiente para la mayoría de los ejemplos del libro.

Instalación de c-lightning desde el código fuente

Los desarrolladores de c-lightning han proporcionado instrucciones detalladas para compilar c-lightning a partir del código fuente. Seguiremos las instrucciones de GitHub.

Instalación de bibliotecas y paquetes de requisitos previos

Estas instrucciones de instalación asumen que está compilando c-lightning en un sistema Linux o similar con herramientas de compilación GNU. Si ese no es el caso, busque las instrucciones para su sistema operativo en el repositorio de Elements Project.

El primer paso común es la instalación de bibliotecas de requisitos previos. Usamos el administrador de paquetes apt para instalar estos:

```
$ sudo apt-obtener actualización

Obtener:1 http://security.ubuntu.com/ubuntu bionic-security InRelease
```

```
 [88,7 KB]
Hit:2 http://eu-north-1b.clouds.archive.ubuntu.com/ubuntu bionic InRelease Get:3 http://eu-
north-1b.clouds.archive.ubuntu.com/ubuntu bionic actualizaciones InRelease [ 88,7 KB]



 [...]

Obtuvo 18,3 MB en 8 s (2180 kB/s)
Leyendo listas de paquetes... Listo

$ sudo apt-get install -y \
    autoconf automake build-essential git libtool libgmp-dev \ libsqlite3-dev python python3 python3-
    mako herramientas de red zlib1g-dev
\
    libsodium-dev gettext

Leyendo listas de paquetes... Listo Creando árbol
de dependencias Leyendo información de
estado... Listo Se instalarán los siguientes paquetes
adicionales:
    autotools-dev binutils binutils-common binutils-x86-64-linux
gnu cpp cpp-7 dpkg-dev fakeroot g++ g++-7 gcc gcc-7 gcc-7-base libalgorithm-diff-perl


   [...]

Configurando libsigsegv2:amd64 (2.12-2) ...
Configurando libltdl-dev:amd64 (2.4.6-14) ...
Configurando python2 (2.7.17-2ubuntu4) ...
Configurando libsodium-dev:amd64 (1.0.18-1) ...

 [...] $
```

Después de unos minutos y mucha actividad en pantalla, habrá instalado todos los paquetes y bibliotecas necesarios. Muchas de estas bibliotecas también son utilizadas por otros paquetes Lightning y son necesarias para el desarrollo de software en general.

Copiar el código fuente de c-lightning

A continuación, copiaremos la última versión de c-lightning del repositorio de código fuente. Para ello, utilizaremos el comando git clone, que

clona una copia controlada por versión en su máquina local, lo que le permite mantenerla sincronizada con los cambios posteriores sin tener que descargar todo el repositorio nuevamente:

```
$ git clone --recurse https://
github.com/ElementsProject/lightning.git Clonación en 'lightning'... remoto:
Enumeración de objetos: 24, listo. remoto: Conteo de objetos: 100%
(24/24), listo. remoto: Comprimir objetos: 100% (22/22), hecho. remoto:
Total 53192 (delta 5), reutilizado 5 (delta 2), pack-reutilizado 53168

Recepción de objetos: 100 % (53192/53192), 29,59 MiB | 19,30 MiB/s, listo.

Resolviendo deltas: 100% (39834/39834), hecho.

$ cd relámpago
```

Ahora tenemos una copia de c-lightning clonada en la subcarpeta ***lightning*** y hemos usado el comando cd (cambiar directorio) para ingresar a esa subcarpeta.

Compilando el código fuente de c-lightning

A continuación, usamos un conjunto de ***scripts de compilación*** que suelen estar disponibles en muchos proyectos de código abierto. Estos scripts de compilación usan los comandos configure y make, que nos permiten:

- Seleccione las opciones de compilación y verifique las dependencias necesarias (configurar)
- Compile e instale los ejecutables y las bibliotecas (make)

Ejecutar configure con la opción de ayuda nos mostrará todas las opciones disponibles:

```
$ ./configure --help Uso: ./
configure [--reconfigure] [configuración=valor] [opciones]

Las opciones incluyen:
```

```
--prefix= (predeterminado /usr/local)
    Prefijo para hacer instalar
--habilitar/deshabilitar-desarrollador (deshabilitar por defecto)
    Modo desarrollador, bueno para probar
--habilitar/deshabilitar-características experimentales (deshabilitar por defecto)
    Habilitar características experimentales --
enable/disable-compat (habilitar por defecto)
    Modo de compatibilidad, es bueno deshabilitarlo para ver si su software se rompe

--enable/disable-valgrind (predeterminado (detección automática))
    Ejecutar pruebas con Valgrind
--enable/disable-static (deshabilitar por defecto)
    Bibliotecas de enlace estático sqlite3, gmp y zlib --enable/disable-
address-sanitizer (deshabilitar por defecto)
    Compilar con desinfectante de direcciones
```

No necesitamos cambiar ninguno de los valores predeterminados para este ejemplo. Por lo tanto, ejecutamos configure nuevamente sin ninguna opción para usar los valores predeterminados:

```
$ ./configurar

Compilando ccan/tools/configurator/configurator...hecho comprobando python3-mako...
encontrado Haciendo que los usuarios de autoconf se sientan cómodos... sí
comprobando off_t es de 32 bits... no comprobando la compatibilidad con __alignof__...
sí


 [...]

Ajuste COMPAT... 1 PYTEST
no encontrado
Configuración STATIC... 0
Configuración ASAN... 0
Configuración TEST_NETWORK... regtest $
```

A continuación, usamos el comando make para crear las bibliotecas, los componentes y los ejecutables del proyecto c-lightning. Esta parte tardará varios minutos en completarse y utilizará mucho la CPU y el disco de su computadora.

¡Espere algo de ruido de los fans! Ejecutar hacer:

```
$ hacer
```

```
cc -DBINTOPKGLIBEXECDIR="\"../libexec/c-lightning\"" -Pared -Wundef -Wmis...

[...]

cc -Og ccan-asort.o ccan-autodata.o ccan-bitmap.o ccan bitops.o ccan-...
```

Si todo va bien, no verá ningún mensaje de ERROR que detenga la ejecución del comando anterior. El paquete de software c-lightning se compiló desde el origen y ahora estamos listos para instalar los componentes ejecutables que creamos en el paso anterior:

```
$ sudo hacer instalar

mkdir -p /usr/local/bin mkdir -p /usr/
local/libexec/c-lightning mkdir -p /usr/local/libexec/c-lightning/
plugins mkdir -p /usr/local/share/man/man1 mkdir -p /usr/local/share/man/
man5 mkdir -p /usr/local/share/man/man7 mkdir -p /usr/local/share/man/man8
mkdir -p /usr/local/share/doc /c-lightning install cli/lightning-cli lightningd/
lightningd /usr/local/bin [...]
```

Para verificar que los comandos lightningd y lightning-cli han sido instalados correctamente, le pediremos a cada ejecutable su información de versión:

```
$ relámpago --versión v0.10.1-34-
gfe86c11 $ relámpago-cli --versión
v0.10.1-34-gfe86c11
```

La versión consta de la última versión de lanzamiento (v0.10.1), seguida de la cantidad de cambios desde el lanzamiento (34) y, finalmente, un hash que identifica exactamente qué revisión (fe86c11). Es posible que vea una versión diferente de la que se muestra anteriormente, ya que el software continúa evolucionando mucho después de la publicación de este libro. Sin embargo, no importa qué versión vea, el hecho de que el

los comandos se ejecutan y responden con la información de la versión significa que ha logrado crear el software c-lightning.

Proyecto de nodo Daemon de Lightning Network

Lightning Network Daemon (LND) es una implementación completa de un nodo LN de Lightning Labs. El proyecto LND proporciona una serie de aplicaciones ejecutables, incluidas lnd (el propio demonio) e lncli (la utilidad de línea de comandos). LND tiene varios servicios de cadena de back-end conectables, incluidos btcd (un nodo completo), bitcoind (Bitcoin Core) y Neutrino (un nuevo cliente ligero experimental). LND está escrito en el lenguaje de programación Go. El proyecto es de código abierto y desarrollado de forma colaborativa en GitHub.

En las próximas secciones, crearemos un contenedor Docker para ejecutar LND, compilaremos LND a partir del código fuente y aprenderemos a configurar y ejecutar LND.

El contenedor acoplable LND

Podemos extraer el contenedor Docker de ejemplo LND del Docker del libro. Repositorio central:

```
$ docker pull lnbook/lnd
Uso de la etiqueta predeterminada: Latest
Latest: Extracción de lnbook/lnd 35807b77a593:
Ya existe e1b85b9c5571: Ya existe 52f9c252546e:
Extracción completa


 [...]

Resumen:
sha256: e490a0de5d41b781c0a7f9f548c99e67f9d728f72e50cd4632722b3ed3
d85952
Estado: Imagen más nueva descargada para lnbook/lnd:latest docker.io/lnbook/
lnd:latest
```

Alternativamente, podemos construir el contenedor LND localmente. El contenedor se encuentra en ***code/ docker/ lnd.*** Cambiamos el directorio de trabajo a ***code/ docker*** y ejecutamos el comando de compilación de docker:

```
$ cd code/docker $ docker
build -t lnbook/lnd lnd Envío del contexto de
compilación al demonio Docker 9.728kB Paso 1/29: DESDE golang:1.13
como lnd-base
 ---> e9bdcb0f0af9
Paso 2/29: ENV GOPATH /ir

[...]

Paso 29/29: CMD ["/usr/local/bin/logtail.sh"]
 ---> Usando caché --->
 397ce833ce14
Construido con éxito 397ce833ce14
Etiquetado correctamente en lnbook/lnd:latest
```

Nuestro contenedor ya está listo para funcionar. Al igual que con el contenedor c-lightning que creamos anteriormente, el contenedor LND también depende de una instancia en ejecución de Bitcoin Core. Como antes, necesitamos iniciar el contenedor bitcoind en otra terminal y conectar LND a través de una red Docker. Ya hemos configurado una red Docker llamada lnbook y la usaremos nuevamente aquí.

PROPINA

Normalmente, cada operador de nodo ejecuta su propio nodo Lightning y su propio nodo Bitcoin en su propio servidor. Para nosotros, un solo contenedor bitcoind puede servir a muchos nodos Lightning. En nuestra red simulada podemos ejecutar varios nodos Lightning, todos conectados a un solo nodo Bitcoin en modo de prueba.

Ejecución de los contenedores bitcoind y LND

Como antes, iniciamos el contenedor bitcoind en una terminal y LND en otra. Si ya tiene el contenedor bitcoind ejecutándose, no

necesita reiniciarlo. Simplemente déjelo en ejecución y omita el siguiente paso.
Para iniciar bitcoind en la red lnbook, usamos docker ejecutado así:

```
$ docker run -it --network lnbook --name bitcoind lnbook/bitcoind
```

A continuación, comenzamos el contenedor LND que acabamos de construir. Como se hizo antes, debemos adjuntarlo a la red lnbook y darle un nombre:

```
$ docker run -it --network lnbook --name lnd lnbook/lnd Esperando a que comience
bitcoind...
Esperando a que bitcoind extraiga bloques...
Comenzando lnd...
Inicio completo
Financiamiento lnd wallet
{"result":"dbd1c8e2b224e0a511c11efb985dabd84d72d935957ac30935ec42 11d28beacb","error":null,"id":"lnd-
run-container"}
 [INF] LTND: Versión: 0.13.1-beta commit=v0.13.1-beta, build=production,
logging=default, debuglevel=info [INF] LTND: Cadena activa: Bitcoin (network=regtest)

 [INF] RPCS: Generando certificados TLS...
```

El contenedor LND se inicia y se conecta al contenedor bitcoind a través de la red Docker. Primero, nuestro nodo LND esperará a que se inicie bitcoind, y luego esperará hasta que bitcoind haya extraído algo de bitcoin en su

cartera. Finalmente, como parte del inicio del contenedor, un script enviará un comando RPC al nodo bitcoind, creando así una transacción que financia la billetera LND con 10 BTC de prueba.

Como demostramos anteriormente, podemos enviar comandos a nuestro contenedor en otra terminal para extraer información, abrir canales, etc. El comando que nos permite enviar instrucciones de línea de comandos al demonio lnd es

llamado lncli. Una vez más, en este contenedor proporcionamos el alias cli que ejecuta lncli con todos los parámetros apropiados. Obtengamos la información del nodo usando el comando docker exec en otra ventana de terminal:

```
$ docker exec lnd cli getinfo {
        "versión": "0.13.1-beta commit=v0.13.1-beta",
```

"commit_hash": "596fd90ef310cd7abbf2251edaae9ba4d5f8a689", "identity_pubkey":
"02d4545dccbeda29a10f44e891858940f4f3374b75c0f85dcb7775bb922fdeaa 14",

[...]

}

Ahora tenemos otro nodo Lightning ejecutándose en la red lnbook y comunicándose con bitcoind. Si todavía está ejecutando el contenedor c lightning, ahora hay dos nodos en ejecución. Todavía no están conectados entre sí, pero los conectaremos entre sí.
pronto.

Si lo desea, puede ejecutar cualquier combinación de nodos LND y c-lightning en la misma Lightning Network. Por ejemplo, para ejecutar un segundo nodo LND, emitiría el comando de ejecución de la ventana acoplable con un nombre de contenedor diferente, así:

```
$ docker run -it --network lnbook --name lnd2 lnbook/lnd
```

En el comando anterior, comenzamos otro contenedor LND y lo llamamos lnd2. Los nombres dependen totalmente de usted, siempre que sean únicos. Si no proporciona un nombre, Docker creará un nombre único combinando aleatoriamente dos palabras en inglés como "naughty_einstein". Este fue el nombre real que Docker eligió para nosotros cuando escribimos este párrafo. ¡Qué divertido!

En la siguiente sección veremos cómo descargar y compilar LND directamente desde el código fuente. Este es un paso opcional y avanzado que le enseñará a usar las herramientas de compilación del lenguaje Go y le permitirá realizar modificaciones en el código fuente de LND. Con este conocimiento, puede escribir código o corregir algunos errores.

> **NOTA**
>
> Si no planea sumergirse en el código fuente o la programación de un nodo Lightning, puede omitir la siguiente sección por completo. El contenedor Docker que acabamos de crear es suficiente para la mayoría de los ejemplos del libro.

Instalar LND desde el código fuente

En esta sección construiremos LND desde cero. LND está escrito en el Go lenguaje de programación. Si desea obtener más información sobre Go, busque golang en lugar de go para evitar resultados irrelevantes. Debido a que está escrito en Go y no en C o C++, utiliza un marco de "compilación" diferente al marco GNU autotools/make que vimos en c-lightning anteriormente.
Sin embargo, no se preocupe, es bastante fácil de instalar y usar las herramientas de golang, y le mostraremos cada paso aquí. Go es un lenguaje fantástico para el desarrollo de software colaborativo porque produce un código muy consistente, preciso y fácil de leer, independientemente de la cantidad de autores. Go está enfocado y es "minimalista" de una manera que fomenta la coherencia entre las versiones del idioma. Como lenguaje compilado, también es bastante eficiente. Sumerjámonos.

Seguiremos las instrucciones de instalación que se encuentran en la documentación del proyecto LND .

Primero, instalaremos el paquete golang y las bibliotecas asociadas. Requerimos estrictamente la versión 1.13 o posterior de Go. Los paquetes de idioma oficiales de Go se distribuyen como archivos binarios del Proyecto Go. Por conveniencia, también están empaquetados como paquetes Debian disponibles a través del comando apt.
Puedes seguir las instrucciones del Proyecto Go o use los siguientes comandos apt en un sistema Debian/Ubuntu Linux como se describe en la página wiki de GitHub en el lenguaje Go:

```
$ sudo apt install golang-go
```

Compruebe que tiene la versión correcta instalada y lista para usar ejecutando:

```
$ ir versión ir
versión go1.13.4 linux/amd64
```

Tenemos 1.13.4, así que estamos listos para... ¡Ir! A continuación, debemos decirle a cualquier programa dónde encontrar el código Go. Esto se logra configurando la variable de entorno GOPATH. Por lo general, el código Go se encuentra en un directorio llamado ***gocode*** directamente en el directorio de inicio del usuario. Con los siguientes dos comandos, establecemos constantemente el GOPATH y nos aseguramos de que su shell lo agregue a su PATH ejecutable. Tenga en cuenta que el directorio de inicio del usuario se denomina ~ en el shell.

```
$ exportar GOPATH=~/gocode $
exportar PATH=$PATH:$GOPATH/bin
```

Para evitar tener que configurar estas variables de entorno cada vez que abre un shell, puede agregar esas dos líneas al final de su archivo de configuración de shell bash .bashrc en su directorio de inicio, usando el editor de su elección.

Copiar el código fuente de LND

Como ocurre con muchos proyectos de código abierto en la actualidad, el código fuente de LND se encuentra en GitHub ***(www.github.com).*** El comando go get puede obtenerlo directamente usando el protocolo Git:

```
$ ir a buscar -d github.com/lightningnetwork/lnd
```

Una vez que termine go get, tendrá un subdirectorio bajo GOPATH que contiene el código fuente de LND.

Compilando el código fuente de LND

LND usa el sistema make build. Para construir el proyecto, cambiamos el directorio al código fuente de LND y luego usamos make así:

```
$ cd $GOPATH/src/github.com/lightningnetwork/lnd $ hacer && hacer
instalar
```

Después de varios minutos, tendrá dos nuevos comandos, lnd e lncli, instalados. Pruébelos y verifique su versión para asegurarse de que estén instalados:

```
$ Lnd-Versión Lnd
Versión 0.10.99-Beta Commit = Clock/V1.0.0-106-
GC1EF5BB908606343D2636C8CD345169E064BDC91 $ LNCLI --Version
LNCLI Versión 0.10.99-Beta COMISM
```

Es probable que vea una versión diferente a la que se muestra anteriormente, ya que el software continúa evolucionando mucho después de la publicación de este libro. Sin embargo, independientemente de la versión que vea, el hecho de que los comandos se ejecuten y le muestren la información de la versión significa que ha logrado crear el software LND.

El proyecto del nodo de rayos Eclair

Eclair (relámpago en francés) es una implementación de Scala de Lightning Network realizada por ACINQ. Eclair también es una de las billeteras móviles Lightning más populares y pioneras, que usamos para demostrar un pago Lightning en el Capítulo 2. En esta sección, examinamos el proyecto del servidor Eclair, que ejecuta un nodo Lightning. Eclair es un proyecto de código abierto y se puede encontrar en GitHub.

En las siguientes secciones, crearemos un contenedor Docker para ejecutar Eclair, como hicimos anteriormente con c-lightning y LND. También construiremos Eclair directamente desde el código fuente.

El contenedor Eclair Docker

Extraigamos el contenedor Eclair del libro del repositorio de Docker Hub:

```
$ ventana acoplable tirar lnbook/eclair
Uso de la etiqueta predeterminada: Latest
Latest: Extracción de lnbook/eclair 35807b77a593: Ya
existe e1b85b9c5571: Ya existe


 [...]

c7d5d5c616c2: tirar completo
Resumen:
sha256:17a3d52bce11a62381727e919771a2d5a51da9f91ce2689c7ecfb03a6f
028315
Estado: Imagen más nueva descargada para lnbook/eclair:latest docker.io/lnbook/
eclair:latest
```

Alternativamente, podemos construir el contenedor localmente, en su lugar. ¡Ya eres casi un experto en las operaciones básicas de Docker! En esta sección repetiremos muchos de los comandos vistos anteriormente para construir el contenedor Eclair. El contenedor se encuentra en ***code/docker/eclair.*** Comenzamos en una terminal cambiando el directorio de trabajo a ***code/docker*** y emitiendo el comando de compilación de docker:

```
$ cd code/docker $ docker
build -t lnbook/eclair eclair Envío del contexto de compilación
al demonio Docker 11.26kB Paso 1/27: ARG OS=ubuntu Paso 2/27: ARG
OS_VER=focal Paso 3/27: DESDE ${OS} :${OS_VER} como base del sistema
operativo

  ---> fb52e22af1b0

 [...]

Paso 27/27: CMD ["/usr/local/bin/logtail.sh"]
  ---> Funcionando en fe639120b726 Quitando
el contenedor intermedio fe639120b726
  ---> e6c8fe92a87c
Construido con éxito e6c8fe92a87c
Etiquetado correctamente en lnbook/eclair:latest
```

Nuestra imagen ya está lista para ejecutarse. El contenedor Eclair también depende de una instancia en ejecución de Bitcoin Core. Como antes, necesitamos iniciar el contenedor bitcoind en otra terminal y conectar Eclair a través de un

Red Docker. Ya hemos configurado una red Docker llamada lnbook y la reutilizaremos aquí.

Una diferencia notable entre Eclair y LND o c-lightning es que Eclair no contiene una billetera bitcoin separada, sino que depende directamente de la billetera bitcoin en Bitcoin Core. Recuerde que con LND financiamos su monedero de bitcoins mediante la ejecución de una transacción para transferir bitcoins del monedero de Bitcoin Core al monedero de bitcoins de LND. Este paso no es necesario con Eclair. Cuando se ejecuta Eclair, la billetera Bitcoin Core se usa directamente como fuente de fondos para abrir canales. Como resultado, a diferencia de los contenedores LND o c lightning, el contenedor Eclair no contiene un script para transferir bitcoins a su billetera al inicio.

Ejecución de los contenedores bitcoind y Eclair

Como antes, iniciamos el contenedor bitcoind en una terminal y el contenedor Eclair en otra. Si ya tiene el contenedor bitcoind ejecutándose, no necesita reiniciarlo. Simplemente déjelo en ejecución y omita el siguiente paso. Para iniciar bitcoind en la red lnbook, usamos docker ejecutado así:

```
$ docker run -it --network lnbook --name bitcoind lnbook/bitcoind
```

A continuación, comenzamos el contenedor Eclair que acabamos de construir. Tendremos que adjuntarlo a la red lnbook y darle un nombre, tal como hicimos con los otros contenedores:

```
$ docker run -it --network lnbook --name eclair lnbook/eclair Esperando a que comience bitcoind...

Esperando a que bitcoind extraiga bloques...
Comenzando eclair...
Nodo Eclair iniciado INFO
obSecp256k1Context - biblioteca secp256k1 cargada con éxito

INFORMACIÓN fr.acinq.eclair.Plugin: cargando 0 complementos INFORMACIÓN
aeslf4j.Slf4jLogger: se inició Slf4jLogger INFORMACIÓN fr.acinq.eclair.Setup: ¡hola!

INFORMACIÓN fr.acinq.eclair.Setup - versión=0.4.2 commit=52444b0

[...]
```

El contenedor Eclair se inicia y se conecta al contenedor bitcoind a través de la red Docker. Primero, nuestro nodo Eclair esperará a que se inicie bitcoind, y luego esperará hasta que bitcoind haya extraído algo de bitcoin en su billetera.

Como demostramos anteriormente, podemos enviar comandos a nuestro contenedor en otra terminal para extraer información, abrir canales, etc. El comando que nos permite enviar instrucciones de línea de comandos al demonio eclair es llamado eclair-cli. Como antes, en este contenedor proporcionamos un alias útil para eclair-cli, llamado simplemente cli, que ofrece los argumentos y parámetros necesarios. Usando el comando docker exec en otra ventana de terminal, obtenemos la información del nodo de Eclair:

```
$ docker exec eclair cli getinfo {

    "versión": "0.4.2-52444b0", "id de nodo":

02fa6d5042eb8098e4d9c9d99feb7ebc9e257401ca7de829b4ce757311e0301d
e7",
    "alias": "eclair", "color":
    "#49daaa", "características": {


 [...]

    },
    "hash de cadena":
“06226e46111a0b59caaf126043eb5bbf28c34f3a5e332a1fc7b2b73cf188910f
",
    "red": "registro", "blockHeight":
    779, "publicAddresses": [],
    "instanceId": "01eb7a68-5db0-461b-
    bdd0-29010df40d73"
}
```

Ahora tenemos otro nodo Lightning ejecutándose en la red lnbook y comunicándose con bitcoind. Puede ejecutar cualquier número y cualquier combinación de nodos Lightning en la misma red Lightning. Puede coexistir cualquier número de nodos Eclair, LND y c-lightning. Para

Por ejemplo, para ejecutar un segundo nodo de Eclair, emitiría el comando de ejecución de la ventana acoplable con un nombre de contenedor diferente, de la siguiente manera:

```
$ docker run -it --network lnbook --name eclair2 lnbook/eclair
```

En el comando anterior iniciamos otro contenedor Eclair llamado eclair2.

En la siguiente sección también veremos cómo descargar y compilar Eclair directamente desde el código fuente. Este es un paso opcional y avanzado que le enseñará a usar las herramientas de compilación de lenguaje Scala y Java y le permitirá realizar modificaciones en el código fuente de Eclair. Con este conocimiento, puede escribir código o corregir algunos errores.

> **NOTA**
>
> Si no planea sumergirse en el código fuente o la programación de un nodo Lightning, puede omitir la siguiente sección por completo. El contenedor Docker que acabamos de crear es suficiente para la mayoría de los ejemplos del libro.

Instalar Eclair desde el código fuente

En esta sección construiremos Eclair desde cero. Eclair está escrito en el lenguaje de programación Scala, que se compila con el compilador Java.
Para ejecutar Eclair, primero debemos instalar Java y sus herramientas de compilación. Seguiremos las instrucciones que se encuentran en el documento ***BUILD.md*** del proyecto Eclair.

El compilador de Java requerido es parte de OpenJDK 11. También necesitaremos un marco de compilación llamado Maven, versión 3.6.0 o superior.

En un sistema Debian/Ubuntu Linux, podemos usar el comando apt para instalar OpenJDK 11 y Maven, como se muestra a continuación:

```
$ sudo apt install openjdk-11-jdk experto
```

Verifique que tiene instalada la versión correcta ejecutando:

```
$ javac -version javac
11.0.7 $ mvn -v Apache
Maven 3.6.1 Inicio de
Maven: /usr/share/maven
Versión de Java: 11.0.7, proveedor:
Ubuntu, tiempo de ejecución: /usr/lib/jvm/java 11-openjdk -amd64
```

Tenemos OpenJDK 11.0.7 y Maven 3.6.1, así que estamos listos.

Copiar el código fuente de Eclair

El código fuente de Eclair está en GitHub. El comando git clone puede crear una copia local para nosotros. Cambiemos a nuestro directorio de inicio y ejecútelo allí:

```
$ cd ~ $
clon de git https://github.com/ACINQ/eclair.git
```

Una vez que finalice git clone, tendrá un subdirectorio eclair que contiene el código fuente del servidor Eclair.

Compilando el código fuente de Eclair

Eclair utiliza el sistema de compilación Maven. Para construir el proyecto, cambiamos el directorio de trabajo al código fuente de Eclair y luego usamos el paquete mvn como este:

```
$ cd eclair $ mvn
paquete [INFO] Buscando
 proyectos ...
 [INFORMACIÓN] ------------------------------------------------ -----------
--------------
 [INFO] Orden de construcción del reactor:
 [INFO]
 [INFO] --------------------< fr.acinq.eclair:eclair_2.13 >-------
--------------
 [INFO] Edificio eclair_2.13 0.4.3-INSTANTÁNEA [1/4]

 [INFO] --------------------------------[ pom ]------------ -------
--------------
```

```
[...]

[INFORMACIÓN] ------------------------------------------------ ----------
--------------

[INFO] ÉXITO DE CONSTRUCCIÓN
[INFORMACIÓN] ------------------------------------------------ ----------
--------------

[INFO] Tiempo total : 01:06 min
[INFO] Finalizado el: 2020-12-12T09:43:21-04:00
[INFORMACIÓN] ------------------------------------------------ ----------
--------------
```

Después de varios minutos, la compilación del paquete Eclair debería completarse. Sin embargo, la acción de "paquete" también ejecutará pruebas, y algunas de estas se conectan a Internet y podrían fallar. Si desea omitir las pruebas, agregue -DskipTests al comando.

Ahora, descomprima y ejecute el paquete de compilación siguiendo las instrucciones para instalar Eclair desde GitHub.

¡Felicidades! ¡Ha creado Eclair desde la fuente y está listo para codificar, probar, corregir errores y contribuir a este proyecto!

Construyendo una red completa de diversos Nodos de relámpagos

Nuestro último ejemplo, presentado en esta sección, reunirá todos los diversos contenedores que hemos creado para formar una Lightning Network hecha de diversas implementaciones de nodos (LND, c-lightning, Eclair). Compondremos la red conectando los nodos y abriendo canales de un nodo a otro. Como paso final, enviaremos un pago a través de estos canales.

En este ejemplo, construiremos una Lightning Network de demostración compuesta por cuatro nodos Lightning llamados Alice, Bob, Chan y Dina. Conectaremos a Alice con Bob, Bob con Chan y Chan con Dina. Esto se muestra en la Figura 4-1.

Figura 4-1. Una pequeña red de demostración de cuatro nodos

Finalmente, haremos que Dina cree una factura y que Alice pague esa factura. Dado que Alice y Dina no están directamente conectadas, el pago se enrutará como HTLC a través de todos los canales de pago.

Uso de docker-compose para orquestar Docker Contenedores

Para que este ejemplo funcione, usaremos una herramienta de ***orquestación de contenedores*** que está disponible como un comando llamado docker-compose. Este comando nos permite especificar una aplicación compuesta por varios contenedores y ejecutar la aplicación lanzando todos los contenedores que cooperan juntos.

Primero, instalemos docker-compose. las instrucciones depende de tu sistema operativo.

Una vez que haya completado la instalación, puede verificar su instalación ejecutando docker-compose de esta manera:

```
$ docker-compose version docker-
compose version 1.21.0, build unknown [...]
```

Los comandos de docker-compose más comunes que usaremos son arriba y abajo, por ejemplo, docker-compose up.

Configuración docker-compose

El archivo de configuración de docker-compose se encuentra en el directorio ***code/ docker*** y se llama ***docker-compose.yml.*** Contiene una especificación para una red y cada uno de los cuatro contenedores. La parte superior se ve así:

```
versión: "3.3"
redes:
  Internet:

servicios:
  bitcoind:
    nombre_contenedor: compilación
    bitcoind:
      contexto: bitcoind
    imagen: lnbook/bitcoind:últimas redes:

      - Internet
    exponer: -
      "18443"
      - "12005"
      - "12006"

  Alicia:
    container_name: Alicia
```

El fragmento anterior define una red llamada lnnet y un contenedor llamado bitcoind que se conectará a la red lnnet. El contenedor

es el mismo que construimos al comienzo de este capítulo. Exponemos tres de los puertos del contenedor, lo que nos permite enviarle comandos y monitorear bloques y transacciones. A continuación, la configuración especifica un contenedor LND llamado "Alice". Más abajo también verá las especificaciones de los contenedores llamados "Bob" (c-lightning), "Chan" (Eclair) y "Dina" (LND nuevamente).

Dado que todas estas implementaciones diversas siguen la especificación BOLT y han sido ampliamente probadas para la interoperabilidad, no tienen dificultad para trabajar juntas para construir una red Lightning.

Inicio de la red Lightning de ejemplo

Antes de comenzar, debemos asegurarnos de que no estamos ejecutando ninguna de los contenedores Si un contenedor nuevo comparte el mismo nombre que uno que está ya se está ejecutando, entonces no podrá iniciarse. Usar ventana acoplable ps, ventana acoplable stop y docker rm según sea necesario para detener y eliminar cualquier contenedores corriendo!

> PROPINA
>
> Debido a que usamos los mismos nombres para estos contenedores Docker orquestados, podríamos necesita "limpiar" para evitar cualquier conflicto de nombres.

Para comenzar con el ejemplo, cambiamos al directorio que contiene el archivo de configuración ***docker compose.yml*** y ejecutamos el comando docker componer:

```
$ código de cd /docker
$ ventana acoplable-componer
Creando Chan... hecho
Creando a Dina                ... hecho
Creando bitcoind... hecho
Creando a Bob... hecho
Creando Alice                 ... hecho
Adjuntando a Chan, Dina, Alice, bitcoind, Bob
Alicia | Esperando a que comience bitcoind...
Beto                | Esperando a que comience bitcoind...
innoble             | Esperando a que comience bitcoind...
Chan                | Esperando a que comience bitcoind...
bitcoind            | Iniciando bitcoind...
bitcoind            | Esperando a que comience bitcoind
bitcoind            | comenzó bitcoind
bitcoind            | ==========================================

 [...]

Chan                | Comenzando eclair...
innoble             | Comenzando lnd...
Chan                | Nodo Eclair iniciado
Alicia              | ...Esperando a que bitcoind extraiga bloques...
Beto                | ...Esperando a que bitcoind extraiga bloques...
Alicia              | Comenzando lnd...
Beto                | Iniciando c-relámpago...
```

```
[...]
```

Después del inicio, verá un flujo completo de archivos de registro a medida que cada nodo se inicia e informa su progreso. Puede parecer bastante desordenado en su pantalla, pero cada línea de salida tiene el prefijo del nombre del contenedor, como se vio anteriormente. Si desea ver los registros de un solo contenedor, puede hacerlo en otra ventana de terminal mediante el comando docker-compose logs con el indicador f ***(seguir)*** y el nombre del contenedor específico:

```
$ docker-compose registros -f Alicia
```

Apertura de canales y enrutamiento de un pago

Nuestra red Lightning ahora debería estar funcionando. Como vimos en las secciones anteriores de este capítulo, podemos enviar comandos a un contenedor Docker en ejecución con el comando docker exec. Independientemente de si comenzamos el contenedor con docker run o comenzamos un montón de ellos con docker-compose up, todavía podemos acceder a los contenedores individualmente usando los comandos de Docker.

La demostración de pago está contenida en un script de shell Bash llamado ejecutar pago-demo.sh. Para ejecutar esta demostración, debe tener el shell Bash instalado en su computadora. La mayoría de los sistemas similares a Linux y Unix (p. ej., macOS) tienen bash preinstalado. Los usuarios de Windows pueden instalar el subsistema de Windows para Linux y usar una distribución de Linux como Ubuntu para obtener un comando bash nativo en su computadora.

Ejecutemos el script para ver su efecto y luego veremos cómo funciona internamente. Usamos bash para ejecutarlo como un comando:

```
$ cd code/docker $ bash
run-payment-demo.sh Inicio de la
demostración de pago
================================================ ====

Esperando que se inicien los nodos -
Esperando el inicio de bitcoind...
```

- Esperando la minería de bitcoind...
- Esperando el inicio de Alice...
- Esperando el arranque de Bob...
- Esperando el inicio de Chan...
- Esperando el inicio de Dina...

Todos los nodos han comenzado

==========

Obtener ID de nodo

- Alicia:

0335e200756e156f1e13c3b901e5ed5a28b01a3131cd0656a27ac5cc20d4e7112
9

- Beto:

033e9cb673b641d2541aaaa821c3f9214e8a11ada57451ed5a0eab2a4afbce7da
a

- Chan:

02f2f12182f56c9f86b9aa7d08df89b79782210f0928cb361de5138364695c742
6

- En:

02d9354cec0458e0d6dee5cfa56b83040baddb4ff88ab64960e0244cc618b99bc
3

==========

[...]

Configuración de conexiones y canales

- Alicia a Bob
- Conexión abierta desde el nodo de Alice al nodo de Bob

- Crear canal de pago Alice->Bob

[...]

Obtenga una factura de 10k sats de Dina

- En la factura:

lnbcrt100u1psnuzzrpp5rz5dg4wy27973yr7ehwns5ldeusceqdaq0hguu8c29n4
nsqkznjsdqqcqzpgxqyz5vqsp5vdpehw33fljnmmexa6ljk55544f3syd8nfttqlm
3ljewu4r0q20q9qyyssqxh5nhkpjgfm47yxn4p9ecvndz7zddlsgpufnpyjl0kmnq
227tdujlm0acdv39hcuqp2vhs40aav70c9yp0tee6tgzk8ut79mr877q0cpkjcfvr

==========

Intento de pago de Alice a Dina ¡Pago exitoso!

Como puede ver en el resultado, el script primero obtiene los ID de nodo (claves públicas) para cada uno de los cuatro nodos. Luego, conecta los nodos y configura un canal de 1,000,000 satoshi de cada nodo al siguiente en la red.
Finalmente, emite una factura por 10.000 satoshis desde el nodo de Dina y paga la factura desde el nodo de Alice.

PROPINA

Si el script falla, puede intentar ejecutarlo nuevamente desde el principio. O puede ejecutar manualmente los comandos que se encuentran en el script uno por uno y ver los resultados.

Hay mucho que revisar en ese guión, pero a medida que adquiera comprensión de la tecnología subyacente, más y más de esa información se aclarará. Le invitamos a revisar este ejemplo más adelante.

Por supuesto, puede hacer mucho más con esta red de prueba que un pago de tres canales y cuatro nodos. Aquí hay algunas ideas para sus experimentos:

- Cree una red más compleja lanzando muchos más nodos de diferentes tipos. Edite el archivo ***docker-compose.yml*** y copie las secciones, cambiando el nombre de los contenedores según sea necesario.
- Conecte los nodos en topologías más complejas: rutas circulares, hub-and-spoke o malla completa.
- Ejecute muchos pagos para agotar la capacidad del canal. Luego ejecute los pagos en la dirección opuesta para reequilibrar los canales. Vea cómo se adapta el algoritmo de enrutamiento.
- Cambie las tarifas del canal para ver cómo el algoritmo de enrutamiento negocia múltiples rutas y qué optimizaciones aplica. ¿Es mejor una ruta larga y barata que una ruta corta y cara?
- Ejecute un pago circular desde un nodo hacia sí mismo para reequilibrar sus propios canales. Vea cómo eso afecta a todos los demás canales y nodos.

- Genera cientos o miles de pequeñas facturas en un bucle y luego págalas lo más rápido posible en otro bucle. Mida cuántas transacciones por segundo puede extraer de esta red de prueba.

PROPINA

relámpago polar le permite visualizar la red con la que ha estado experimentando usando Docker.

Conclusión

En este capítulo analizamos varios proyectos que implementan las especificaciones BOLT. Construimos contenedores para ejecutar una red Lightning de muestra y aprendimos cómo construir cada proyecto a partir del código fuente. Ahora está listo para explorar más y profundizar.

Capítulo 5. Operación de un nodo Lightning Network

Después de haber leído hasta aquí, probablemente haya configurado una billetera Lightning. En este capítulo, llevaremos las cosas un paso más allá y configuraremos un nodo Lightning completo. Además de configurar uno, aprenderemos a operarlo y mantenerlo a lo largo del tiempo.

Existen muchas razones por las que es posible que desee configurar su propio nodo Lightning. Incluyen:

- Para ser un participante completo y activo en Lightning Network, no solo un usuario final
- Para administrar una tienda de comercio electrónico o recibir ingresos a través de pagos Lightning
- Para obtener ingresos de las tarifas de enrutamiento Lightning o mediante el alquiler de liquidez del canal
- Para desarrollar nuevos servicios, aplicaciones o complementos para Lightning La red
- Para aumentar su privacidad financiera mientras usa Lightning
- Para usar algunas aplicaciones creadas sobre Lightning, como las aplicaciones de mensajería instantánea con tecnología Lightning
- Por la libertad financiera, la independencia y la soberanía

Hay costos asociados con la ejecución de un nodo LN. ¡Necesita una computadora, una conexión permanente a Internet, mucho espacio en disco y mucho tiempo!
Los costos operativos incluirán los gastos de electricidad.

Pero las habilidades que aprenderá de esta experiencia son valiosas y también se pueden aplicar a una variedad de otras tareas.

¡Empecemos!

> **NOTA**
>
> Es importante que establezca sus propias expectativas correctamente en hechos precisos. Si planea operar un nodo Lightning ***únicamente*** para obtener ingresos mediante la obtención de tarifas de enrutamiento, primero haga su tarea con diligencia. Dirigir un negocio rentable mediante la operación de un nodo Lightning definitivamente ***no*** es fácil. Calcule todos sus costos iniciales y continuos en una hoja de cálculo. Estudie cuidadosamente las estadísticas de LN. ¿Cuál es el volumen de pago actual? ¿Cuál es el volumen por nodo? ¿Cuáles son las tarifas de enrutamiento promedio actuales? Consulte foros y solicite consejos o comentarios de otros miembros de la comunidad que ya hayan adquirido experiencia en el mundo real. Forme su propia opinión educada solo ***después de*** haber realizado este ejercicio de diligencia debida. La mayoría de las personas encontrarán su motivación para ejecutar un nodo no en la ganancia financiera, sino en otro lugar.

Elegir su plataforma

Hay muchas maneras de ejecutar un nodo Lightning, que van desde una pequeña mini PC alojada en su hogar o un servidor dedicado, hasta un servidor alojado en la nube. El método que elijas dependerá de los recursos que tengas y de cuánto dinero quieras gastar.

¿Por qué es importante la confiabilidad para ejecutar un Lightning Node?

En Bitcoin, el hardware no es particularmente importante a menos que uno esté ejecutando específicamente un nodo de minería. El software del nodo Bitcoin Core se puede ejecutar en cualquier máquina que cumpla con los requisitos mínimos y no necesita estar en línea para recibir pagos, solo para enviarlos. Si un nodo de Bitcoin deja de funcionar durante un período de tiempo prolongado, el usuario simplemente puede reiniciar el nodo y, una vez que se conecte al resto de la red, volverá a sincronizar la cadena de bloques.

Sin embargo, en Lightning, el usuario debe estar en línea tanto para enviar ***como*** para recibir pagos. Si el nodo Lightning está fuera de línea, no puede recibir ningún pago de nadie y, por lo tanto, sus facturas abiertas no se pueden cumplir. Además, los canales abiertos de un nodo fuera de línea no se pueden utilizar para enrutar pagos. Sus socios de canal notarán que está desconectado y no puede

contactarlo para enrutar un pago. Si está desconectado con demasiada frecuencia, es posible que consideren que el bitcoin bloqueado en sus canales con usted es una capacidad infrautilizada y pueden cerrar esos canales. Ya discutimos el caso de un ataque de protocolo en el que su socio de canal intenta engañarlo al enviar una transacción de compromiso anterior. Si está desconectado y sus canales no están siendo monitoreados, entonces el intento de robo podría tener éxito y no tendrá ningún recurso una vez que expire el bloqueo de tiempo. Por lo tanto, la confiabilidad del nodo es extremadamente importante para un nodo Lightning.

También están los problemas de fallas de hardware y pérdida de datos. En Bitcoin, una falla de hardware puede ser un problema trivial si el usuario tiene una copia de seguridad de su frase mnemotécnica o claves privadas. La billetera Bitcoin y el bitcoin dentro de la billetera se pueden restaurar fácilmente desde las claves privadas en una computadora nueva.
La mayor parte de la información se puede volver a descargar desde la cadena de bloques.

Por el contrario, en Lightning, la información sobre los canales del usuario, incluidas las transacciones de compromiso y los secretos de revocación, no se conocen públicamente y solo se almacenan en el hardware del usuario individual. Por lo tanto, las fallas de software y hardware en Lightning Network pueden resultar fácilmente en la pérdida de fondos.

Tipos de nodos Lightning de hardware

Hay tres tipos principales de nodos Lightning de hardware:

Computadoras de propósito general

Un nodo LN se puede ejecutar en una computadora doméstica o portátil con Windows, macOS o Linux. Por lo general, esto se ejecuta junto con un nodo de Bitcoin.

hardware dedicado

Un nodo Lightning también se puede ejecutar en hardware dedicado como Raspberry Pi, Rock64 o mini PC. Esta configuración generalmente ejecutaría una pila de software, incluido un nodo de Bitcoin y otras aplicaciones. Esta configuración es popular porque el hardware está dedicado a ejecutar y mantener el nodo Lightning únicamente y, por lo general, se configura con un "ayudante" de instalación.

Hardware preconfigurado

Un nodo LN también se puede ejecutar en un hardware especialmente diseñado, seleccionado y configurado específicamente para él. Esto incluiría "listo para usar" Soluciones de nodos de rayos que se pueden comprar como un kit o un sistema llave en mano.

Corriendo en la “Nube”

El servidor privado virtual (VPS) y los servicios de computación en la nube como Microsoft Azure, Google Cloud, Amazon Web Services (AWS) o DigitalOcean son bastante asequibles y se pueden configurar muy rápidamente. Un nodo Lightning se puede alojar por entre $20 y $40 por mes en dicho servicio.

Sin embargo, como dice el refrán, "'Cloud' son solo las computadoras de otras personas". Usar estos servicios significa ejecutar su nodo en las computadoras de otras personas. Esto trae consigo las correspondientes ventajas y desventajas. Las ventajas clave son la conveniencia, la eficiencia, el tiempo de actividad y posiblemente incluso el costo. El operador de la nube administra y ejecuta el nodo en un alto grado, brindándole automáticamente conveniencia y eficiencia. Brindan un excelente tiempo de actividad y disponibilidad, a menudo mucho mejor que lo que una persona puede lograr en casa. Si considera que solo el costo de la electricidad para ejecutar un servidor en muchos países occidentales es de alrededor de $ 10 por mes, luego agregue el costo del ancho de banda de la red y el hardware en sí, la oferta de VPS se vuelve financieramente competitiva. Por último, con un VPS no necesita espacio para una PC en casa y no tiene problemas con el ruido o el calor de la PC. Por otro lado, hay varias desventajas notables. Un nodo Lightning que se ejecuta en la "nube" siempre será menos seguro y menos privado que uno que se ejecuta en su propia computadora. Además, estos servicios de computación en la nube están muy centralizados. La gran mayoría de los nodos Bitcoin y Lightning que se ejecutan en dichos servicios se encuentran en un puñado de centros de datos en Virginia, Sunnyvale, Seattle, Londres y Frankfurt. Cuando las redes o centros de datos de estos proveedores tienen problemas de servicio, afecta a miles de nodos en las llamadas redes “descentralizadas”.

Si tiene la posibilidad y la capacidad de ejecutar un nodo en su propia computadora en casa o en su oficina, entonces esto podría ser preferible a ejecutarlo en la nube. No obstante, si ejecutar su propio servidor no es una opción, considere ejecutar uno en un VPS.

Ejecutar un nodo en casa

Si tiene una conexión a Internet de capacidad razonable en casa o en su oficina, ciertamente puede ejecutar un nodo Lightning allí. Cualquier conexión de "banda ancha" es suficiente para ejecutar un nodo ligero, y una conexión rápida también le permitirá ejecutar un nodo completo de Bitcoin.

Si bien puede ejecutar un nodo Lightning (e incluso un nodo Bitcoin) en su computadora portátil, se volverá molesto bastante rápido. Estos programas consumen los recursos de su computadora y necesitan ejecutarse 24/7. Sus aplicaciones de usuario, como su navegador o su hoja de cálculo, competirán contra los servicios en segundo plano de Lightning por los recursos de su computadora. En otras palabras, su navegador y otras cargas de trabajo de escritorio se ralentizarán. Y cuando su aplicación de procesamiento de textos congela su computadora portátil, su nodo Lightning también se desactivará, dejándolo incapaz de recibir transacciones y potencialmente vulnerable a los ataques. Además, nunca debe apagar su computadora portátil.

Todo esto combinado da como resultado una configuración que no es la ideal. Lo mismo se aplicará a su PC de escritorio personal de uso diario.

En cambio, la mayoría de los usuarios elegirán ejecutar un nodo en una computadora dedicada. Afortunadamente, no necesita una computadora de clase "servidor" para hacer esto. Puede ejecutar un nodo Lightning en una computadora de placa única, como una Raspberry Pi o en una mini PC (generalmente comercializada como PC de cine en casa). Estas son computadoras simples que se usan comúnmente como un centro de automatización del hogar o un servidor de medios. Son relativamente económicos en comparación con una PC o una computadora portátil.
La ventaja de un dispositivo dedicado como plataforma para los nodos Lightning y Bitcoin es que puede ejecutarse de forma continua, silenciosa y discreta en su red doméstica, escondido detrás de su enrutador o televisor. ¡Nadie sabrá siquiera que esta pequeña caja es en realidad parte de un sistema bancario global!

ADVERTENCIA

No se recomienda operar un nodo en un sistema operativo de 32 bits y/o una CPU de 32 bits, ya que el software del nodo puede tener problemas de recursos, provocando un bloqueo y posiblemente una pérdida de fondos.

¿Qué hardware se requiere para ejecutar un nodo Lightning?

Como mínimo, se requiere lo siguiente para ejecutar un nodo Lightning:

UPC

Se requiere suficiente potencia de procesamiento para ejecutar un nodo Bitcoin, que descargará y validará continuamente nuevos bloques. El usuario también debe tener en cuenta la descarga del bloque inicial (IBD, por sus siglas en inglés) al configurar un nuevo nodo de Bitcoin, lo que puede llevar desde varias horas hasta varios días. Se recomienda una CPU de 2 o 4 núcleos.

RAM

Un sistema con 2 GB de RAM ***apenas*** ejecutará nodos Bitcoin y Lightning. Funcionará mucho mejor con al menos 4 GB de RAM. El IBD será especialmente desafiante con menos de 4 GB de RAM. Más de 8 GB de RAM son innecesarios porque la CPU es el mayor cuello de botella para este tipo de servicios, debido a operaciones criptográficas como la validación de firmas.

unidad de almacenamiento

Puede ser una unidad de disco duro (HDD) o una unidad de estado sólido (SSD). Un SSD será significativamente más rápido (pero más costoso) para ejecutar un nodo. La mayor parte del almacenamiento se utiliza para la cadena de bloques de Bitcoin, que tiene un tamaño de cientos de gigabytes. Una compensación justa (costo por complejidad) es comprar un SSD pequeño para iniciar el sistema operativo y un HDD más grande para almacenar objetos de datos grandes (principalmente bases de datos).

> **NOTA**
>
> Raspberry Pis es una opción común para ejecutar software de nodo, debido al costo y la disponibilidad de piezas. El sistema operativo que se ejecuta en el dispositivo generalmente se inicia desde una tarjeta digital segura (SD). Para la mayoría de los casos de uso, esto no es un problema, pero Bitcoin Core es conocido por ser pesado en E/S. Debe asegurarse de colocar la cadena de bloques de Bitcoin y el directorio de datos Lightning en una unidad diferente porque la E/S intensiva a largo plazo puede hacer que una tarjeta SD falle.

conexión a Internet

Se requiere una conexión a Internet confiable para descargar nuevos bloques de Bitcoin, así como para comunicarse con otros pares de Lightning. Durante el funcionamiento, el uso de datos estimado oscila entre 10 y 100 GB al mes, según la configuración. Al inicio, un nodo completo de Bitcoin descarga la cadena de bloques completa.

Fuente de alimentación

Se requiere una fuente de alimentación confiable porque los nodos Lightning deben estar en línea en todo momento. Un corte de energía hará que los pagos en curso fallen. Para los nodos de enrutamiento de servicio pesado, una fuente de alimentación ininterrumpida (UPS) o de respaldo es útil en caso de cortes de energía. Idealmente, también debe conectar su enrutador de Internet a este UPS.

Respaldo

La copia de seguridad es crucial porque una falla puede provocar la pérdida de datos y, por lo tanto, la pérdida de fondos. Deberá considerar algún tipo de solución de copia de seguridad de datos. Esto podría ser una copia de seguridad automatizada basada en la nube en un servidor o servicio web que controle. Alternativamente, podría ser una copia de seguridad de hardware local automatizada, como un segundo disco duro. Para obtener los mejores resultados, se pueden combinar las copias de seguridad locales y remotas.

Cambiar la configuración del servidor en la nube

Al alquilar un servidor en la nube, suele ser rentable cambiar la configuración entre dos fases de funcionamiento. Una CPU más rápida y más rápida

se necesitará almacenamiento durante la EII (p. ej., el primer día). Una vez que la cadena de bloques se ha sincronizado, los requisitos de velocidad de almacenamiento y CPU son mucho menores, por lo que el rendimiento puede reducirse a un nivel más rentable.

Por ejemplo, en la nube de Amazon, usaríamos una RAM de 8 a 16 GB, una CPU de 8 núcleos (p. ej., t3-large o m3.large) y una SSD de 400 GB más rápida (más de 1000 operaciones de entrada/salida aprovisionadas por segundo [IOPS]) para la EII, reduciendo su tiempo a solo 6-8 horas. Una vez que esté completo, cambiaríamos la instancia del servidor a una RAM de 2 GB, una CPU de 2 núcleos (p. ej., t3.small) y el almacenamiento a una unidad de disco duro de 1 TB de propósito general. Esto costará casi lo mismo que si lo ejecutara en el servidor más lento todo el tiempo, pero lo pondrá en funcionamiento en menos de un día en lugar de tener que esperar casi una semana por la EII.

Almacenamiento de datos permanente (unidad)

Si usa una mini PC o alquila un servidor, el almacenamiento puede ser la parte más costosa, ya que cuesta tanto como la computadora y la conectividad (datos) juntos.

Echemos un vistazo a las diferentes opciones disponibles. Primero, hay dos tipos principales de unidades, HDD y SSD. Los HDD son más baratos y los SSD son más rápidos, pero ambos hacen el trabajo.

Los SSD más rápidos disponibles en la actualidad utilizan la interfaz Non-Volatile Memory Express (NVMe). Los SSD NVMe son más rápidos en máquinas de gama alta, pero también más costosos. Los SSD tradicionales basados en SATA son más baratos, pero no tan rápidos. Los SSD SATA funcionan lo suficientemente bien para la configuración de su nodo. Es posible que las computadoras más pequeñas no puedan aprovechar las SSD NVMe. Por ejemplo, la Raspberry Pi 4 no puede beneficiarse de ellos debido al ancho de banda limitado de su puerto USB.

Para elegir el tamaño, echemos un vistazo a la cadena de bloques de Bitcoin. A partir de agosto de 2021, su tamaño es de 360 GB, incluido el índice de transacciones, y crece aproximadamente 60 GB por año. Si desea tener algún margen disponible para el crecimiento futuro o para instalar otros datos en su nodo, compre al menos una unidad de 512 GB, o mejor aún, una unidad de 1 TB.

Uso de un instalador o ayudante

Instalar un nodo Lightning o un nodo Bitcoin puede resultar abrumador si no está familiarizado con un entorno de línea de comandos. Afortunadamente, hay una serie de proyectos que hacen "ayudantes", es decir, software que instala y configura los distintos componentes para usted. Aún necesitará aprender algunos encantamientos de línea de comandos para interactuar con su nodo, pero la mayor parte del trabajo inicial ya está hecho.

RaspiBlitz

Uno de los “ayudantes” más populares y completos es ***RaspiBlitz*** (Figura 5-1), un proyecto creado por Christian Rotzoll. Está diseñado para instalarse en una Raspberry Pi 4. RaspiBlitz viene con un kit de hardware recomendado que puede construir en cuestión de horas o, como máximo, en un fin de semana. Si asiste a un "hackathon" de Lightning en su ciudad, es probable que vea a muchas personas trabajando en su configuración de RaspiBlitz, intercambiando consejos y ayudándose unos a otros.

Puede encontrar el proyecto RaspiBlitz en GitHub.

Además de un nodo Bitcoin y Lightning, RaspiBlitz puede instalar una serie de servicios adicionales, como:

- Tor (ejecutar como servicio oculto)
- ElectRS (servidor Electrum en Rust)
- Servidor BTCPay (procesador de pago de criptomonedas)
- BTC RPC Explorer (explorador de cadena de bloques de Bitcoin)
- Ride The Lightning (GUI de administración de nodos Lightning)
- LNbits (monedero/sistema de cuentas Lightning)
- Spectre Desktop (multisig Trezor, Ledger, billetera Coldcard y Espectro-bricolaje)
- lndmanage (interfaz de línea de comandos para la gestión avanzada de canales)

- Loop (servicio de intercambio de submarinos)
- JoinMarket (servicio CoinJoin)

Figura 5-1. Un nodo RaspiBlitz

Minodo

miNodo es otro proyecto popular de "ayuda" de código abierto que incluye una gran cantidad de software relacionado con Bitcoin. Es fácil de instalar: "flashea" el instalador en una tarjeta SD y arranca su mini PC desde la tarjeta SD. No necesita ningún monitor para usar myNode porque las herramientas administrativas son accesibles de forma remota desde un navegador. Si su mini PC no tiene monitor, mouse o teclado, puede administrarlo desde otra computadora o incluso desde su teléfono inteligente. Una vez instalado, vaya a ***http:// mynode.local*** y cree una billetera Lightning y un nodo en dos clics.

Además de un nodo Bitcoin y Lightning, myNode puede instalar opcionalmente una variedad de servicios adicionales, como:

- Ride The Lightning (GUI de administración de nodos Lightning)
- OpenVPN (soporte de red privada virtual [VPN] para administración remota o billetera)
- lndmanage (interfaz de línea de comandos para la gestión avanzada de canales)
- BTC RPC Explorer (un explorador de cadena de bloques de Bitcoin)

paraguas

Famoso por su UX/UI (que se muestra en la Figura 5-2), ***Umbrel*** ofrece una manera muy fácil y accesible de poner en funcionamiento su nodo Bitcoin y Lightning en muy poco tiempo, especialmente para los principiantes. Una característica muy distintiva es que Umbrel utiliza Neutrino/SPV durante la EII para que pueda comenzar a usar su nodo al instante. Una vez que Bitcoin Core está completamente sincronizado en segundo plano, cambia automáticamente y desactiva el modo SPV. Umbrel OS es compatible con Raspberry Pi 4 y también se puede instalar en cualquier sistema operativo basado en Linux o en una máquina virtual en macOS o Windows. También puede conectar cualquier billetera que admita Bitcoin Core P2P, Bitcoin Core RPC, el protocolo Electrum o lndconnect.

No hay necesidad de esperar un día lluvioso, puedes ir directamente a Umbrel aprender más.

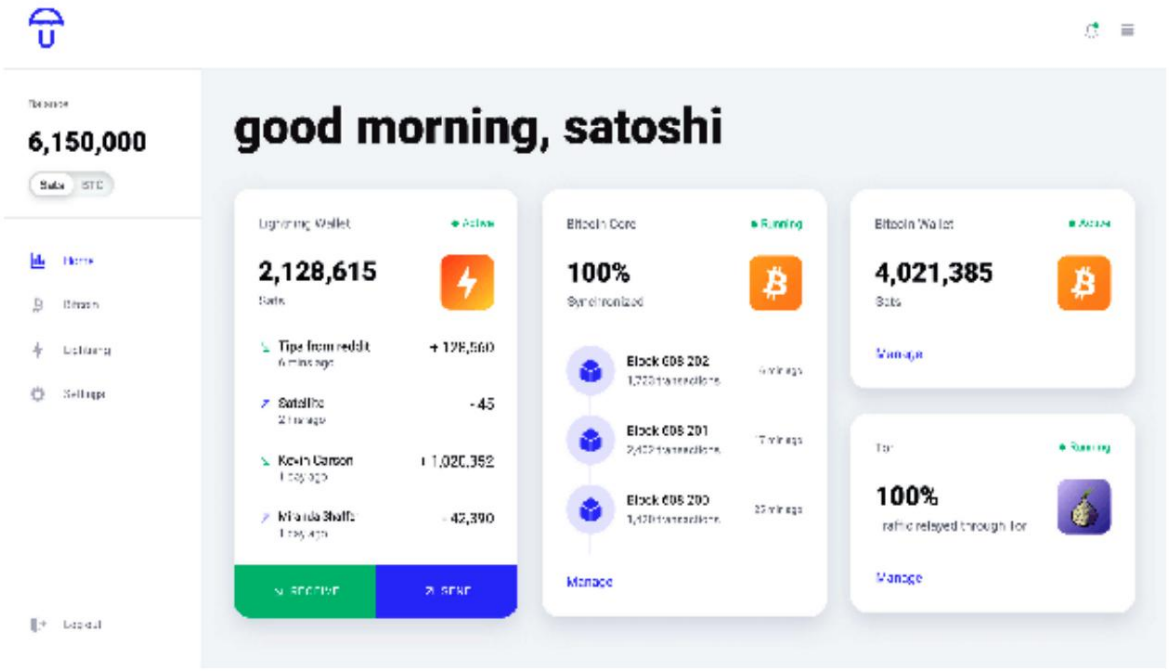


Figura 5-2. La interfaz web de Umbrel

Además de un nodo Bitcoin y Lightning, Umbrel presentó el Umbrel
App Store, donde puede instalar fácilmente servicios adicionales, como:

- Lightning Terminal (interfaz para gestionar la liquidez del canal, Loop
  entrada y salida de bucle)
- Ride The Lightning (GUI de administración de nodos Lightning)
- Spectre Desktop (coordinador de solo visualización para billeteras Bitcoin de
  múltiples firmas y de una sola clave)
- Servidor BTCPay (procesador de pago de criptomonedas)
- BTC RPC Explorer (explorador de cadena de bloques de Bitcoin)
- ThunderHub (supervisa y administra tu nodo)
- Sphinx Relay (manejo de conectividad y almacenamiento para el chat de Sphinx)
- mempool.space (visualizador de mempool y explorador de bloques)
- LNbits (monedero/sistema de cuentas Lightning)

Actualmente, Umbrel todavía está en versión beta y no se considera seguro.

Servidor BTCPay

Si bien no se diseñó inicialmente como un "ayudante" de instalación, la plataforma de pago y comercio electrónico ***BTCPay Server*** tiene un sistema de instalación increíblemente fácil que utiliza contenedores Docker y docker-compose para instalar un nodo Bitcoin, un nodo Lightning y una pasarela de pago, entre muchos otros servicios. Se puede instalar en una variedad de plataformas de hardware, desde una simple Raspberry Pi 4 (se recomiendan 4 GB) hasta una mini PC o una computadora portátil, una computadora de escritorio o un servidor antiguos.

Servidor BTCPay es una plataforma de comercio electrónico con autohospedaje y autocustodia con todas las funciones que se puede integrar con muchas plataformas de comercio electrónico, como WordPress WooCommerce y otras. La instalación del nodo completo es solo un paso de la instalación de la plataforma de comercio electrónico. Si bien se desarrolló originalmente como un reemplazo característica por característica del servicio de pago comercial y la API de ***BitPay ,*** ha evolucionado más allá de eso para convertirse en una plataforma completa para los servicios BTC y Lightning relacionados con el comercio electrónico. Para muchos vendedores o tiendas, es una solución llave en mano de una sola tienda para el comercio electrónico.

Además de un nodo Bitcoin y Lightning, BTCPay Server también puede instalar una variedad de servicios, que incluyen:

- c-relámpago o nodo LND Lightning
- soporte de Litecoin
- soporte Monero
- Servidor Spark (billetera web c-lightning)
- Servidor de carga (API de comercio electrónico c-lightning)
- Ride The Lightning (GUI web de administración de nodos Lightning)
- Muchas bifurcaciones BTC
- BTCTransmuter (servicio de automatización de eventos y acciones que admite el cambio de divisas)

La cantidad de servicios y características adicionales está creciendo rápidamente, por lo que la lista anterior es solo un pequeño subconjunto de lo que está disponible en BTCPay.

Plataforma de servidor.

Nodo Bitcoin o Relámpago Liviano

Una elección crítica para su configuración será la elección del nodo Bitcoin y su configuración. ***Bitcoin Core,*** la implementación de referencia, es la opción más común pero no la única opción disponible. Una opción alternativa es ***btcd,*** que es una implementación en lenguaje Go de un nodo Bitcoin. btcd admite algunas funciones que son útiles para ejecutar un nodo LND Lightning y no están disponibles en Bitcoin Core.

Una segunda consideración es si ejecutará un nodo de ***archivo de*** Bitcoin con una copia completa de la cadena de bloques (unos 350 GB a mediados de 2021) o una ***cadena*** de bloques recortada que solo conserva los bloques más recientes. Una cadena de bloques podada puede ahorrarle algo de espacio en el disco, pero aún necesitará descargar la cadena de bloques completa al menos una vez (durante la EII). Por lo tanto, no le ahorrará ningún tráfico de red. El uso de un nodo podado para ejecutar un nodo Lightning sigue siendo una capacidad experimental y es posible que no admita todas las funciones.

Sin embargo, muchas personas están ejecutando un nodo como ese con éxito.

Finalmente, también tiene la opción de no ejecutar ningún nodo de Bitcoin. En su lugar, puede operar el nodo LND Lightning en modo "ligero", utilizando el Protocolo Neutrino para recuperar información de la cadena de bloques de los nodos públicos de Bitcoin operados por otros. Correr así significa que está tomando recursos de la red Bitcoin sin ofrecer nada a cambio. En cambio, está ofreciendo sus recursos y contribuyendo a la comunidad de LN. Para los nodos Lightning más pequeños, esto generalmente reducirá el tráfico de la red en comparación con ejecutar un nodo Bitcoin completo.

Tenga en cuenta que operar un nodo Bitcoin le permite admitir otros servicios, además de un nodo Lightning. Estos otros servicios pueden requerir un nodo de Bitcoin de archivo (no podado) y, a menudo, no pueden ejecutarse sin un nodo de Bitcoin. Considere por adelantado qué otros servicios puede querer ejecutar ahora o en el futuro para tomar una decisión informada sobre el tipo de nodo de Bitcoin que seleccione.

El resultado final de esta decisión es: si puede pagar un disco de más de 500 GB, ejecute un nodo Bitcoin de archivo completo. Estará contribuyendo con recursos al sistema Bitcoin y ayudando a otros que no pueden permitírselo. Si no puede permitirse un disco tan grande, ejecute un nodo podado. Si no puede pagar el disco o el ancho de banda incluso para un nodo podado, ejecute un nodo LND ligero sobre Neutrino.

Elección del sistema operativo

El siguiente paso es seleccionar un sistema operativo para su nodo. La gran mayoría de los servidores de Internet se ejecutan en alguna variante de Linux. Linux es la plataforma preferida para Internet porque es un poderoso sistema operativo de código abierto. Linux, sin embargo, tiene una curva de aprendizaje empinada y requiere familiaridad con un entorno de línea de comandos. A menudo es intimidante para
usuarios nuevos.

En última instancia, la mayoría de los servicios se pueden ejecutar en cualquier sistema operativo POSIX moderno, que incluye macOS, Windows y, por supuesto, Linux. Su elección debe basarse más en su familiaridad y comodidad con un sistema operativo y sus objetivos de aprendizaje. Si quieres ampliar tus conocimientos y aprender a operar un sistema Linux, esta es una gran oportunidad para hacerlo con un proyecto específico y un objetivo claro. Si solo desea poner en marcha un nodo, vaya con lo que sabe.

Hoy en día, muchos servicios también se entregan en forma de contenedores, generalmente basados en el sistema Docker. Estos contenedores se pueden implementar en una variedad de sistemas operativos, abstrayendo el sistema operativo subyacente. No obstante, es posible que deba aprender algunos comandos CLI de Linux, ya que la mayoría de los contenedores ejecutan alguna variante de Linux en su interior.

Elija la implementación de su nodo Lightning

Al igual que con la elección del sistema operativo, la elección de la implementación del nodo Lightning debe depender principalmente de su familiaridad con el lenguaje de programación y las herramientas de desarrollo utilizadas por los proyectos. Si bien hay algunas pequeñas diferencias en las características entre los distintos nodos

implementaciones, que son relativamente menores, y la mayoría de las implementaciones convergen en los estándares comunes definidos por los BOLT.

La familiaridad con el lenguaje de programación y el sistema de compilación, por otro lado, es una buena base para elegir un nodo. Esto se debe a que la instalación, la configuración, el mantenimiento continuo y la solución de problemas implicarán interactuar con las diversas herramientas utilizadas por el sistema de compilación. Esto incluye:

- Utilidades Make, Autotools y GNU para c-lightning
- Ir a servicios públicos para LND
- Java/Maven para Eclair

El lenguaje de programación influye no solo en la elección del sistema de compilación, sino también en muchos otros aspectos del programa. Cada lenguaje de programación viene con una filosofía de diseño completa y afecta muchos otros aspectos, como
como:

- Formato y sintaxis de los archivos de configuración
- Ubicaciones de archivos (en el sistema de archivos)
- Argumentos de la línea de comandos y su sintaxis
- Formato de mensaje de error
- Bibliotecas de requisitos previos
- Interfaces de llamadas a procedimientos remotos

Cuando elige su nodo Lightning, también está eligiendo todas las características mencionadas anteriormente. Por lo tanto, su familiaridad con estas herramientas y filosofías de diseño facilitará la ejecución de un nodo. O más difícil, si aterrizas en un dominio desconocido.

Por otro lado, si esta es su primera incursión en la línea de comandos y el entorno de servidor/servicio, no estará familiarizado con ninguna implementación y tendrá la oportunidad de aprender algo completamente.

nuevo. En ese caso, es posible que desee decidir en función de una serie de otros factores, como:

- Calidad de los foros de soporte y salas de chat.
- Calidad de la documentación
- Grado de integración con otras herramientas que quieras ejecutar

Como consideración final, es posible que desee examinar el rendimiento y la confiabilidad de las diferentes implementaciones de nodos. Esto es especialmente importante si va a usar este nodo en un entorno de producción y espera un tráfico intenso y requisitos de alta confiabilidad. Este podría ser el caso si planea ejecutar el sistema de pago de una tienda en él.

Instalación de un nodo Bitcoin o Lightning

¿Decidió no utilizar un "ayudante" de instalación y, en su lugar, sumergirse en la línea de comandos de un sistema operativo Linux? Es una decisión valiente y trataremos de ayudarlo a que funcione. Si prefiere no intentar hacer esto manualmente, considere usar una aplicación que lo ayude a instalar el software del nodo o una solución basada en contenedores, como se describe en "Uso de un instalador o asistente".

ADVERTENCIA

Esta sección profundizará en el tema avanzado de la administración del sistema desde la línea de comandos. La administración de Linux es su propio conjunto de habilidades que está fuera del alcance de este libro. Es un tema complicado y hay muchas trampas. ¡Proceda con precaución!

En las próximas secciones, describiremos brevemente cómo instalar y configurar un nodo Bitcoin y Lightning en un sistema operativo Linux. Deberá revisar las instrucciones de instalación para las aplicaciones de nodo Bitcoin y Lightning específicas que decidió usar. Por lo general, puede encontrarlos en un archivo llamado ***INSTALL*** o en el subdirectorio ***docs*** de cada proyecto. solo lo haremos

describe algunos de los pasos comunes que se aplican a todos estos servicios, y las instrucciones que ofrecemos serán necesariamente incompletas.

Servicios en segundo plano

Para aquellos que están acostumbrados a ejecutar aplicaciones en su escritorio o teléfono inteligente, una aplicación siempre tiene una interfaz gráfica de usuario, incluso si a veces se ejecuta en segundo plano. Las aplicaciones de los nodos Bitcoin y Lightning, sin embargo, son muy diferentes. Estas aplicaciones no tienen una interfaz gráfica de usuario integrada. En su lugar, se ejecutan como servicios en ***segundo*** plano, lo que significa que siempre funcionan en segundo plano y no interactúan directamente con el usuario.

Esto puede crear cierta confusión para los usuarios que no están acostumbrados a ejecutar servicios en segundo plano. ¿Cómo saber si dicho servicio se está ejecutando actualmente? ¿Cómo lo inicias y lo detienes? ¿Cómo interactúas con él? Las respuestas a estas preguntas dependen del sistema operativo que esté utilizando. Por ahora, asumiremos que está utilizando alguna variante de Linux y las responderemos en ese contexto.

Aislamiento de procesos

Los servicios en segundo plano generalmente se ejecutan bajo una cuenta de usuario específica para aislarlos del sistema operativo y entre sí. Por ejemplo, Bitcoin Core está configurado para ejecutarse como usuario bitcoin. Deberá usar la línea de comando para crear un usuario para cada uno de los servicios que ejecuta.

Además, si ha conectado una unidad externa, deberá indicarle al sistema operativo que reubique el directorio de inicio del usuario en esa unidad. Esto se debe a que un servicio como Bitcoin Core creará archivos en el directorio de inicio del usuario. Si lo está configurando para descargar la cadena de bloques de Bitcoin completa, estos archivos ocuparán varios cientos de gigabytes. Aquí, asumimos que ha conectado la unidad externa y está ubicada en la ruta ***/external_drive/*** del sistema operativo.

En la mayoría de los sistemas Linux, puede crear un nuevo usuario con el comando useradd, así:

```
$ sudo useradd -m -d /external_drive/bitcoin -s /dev/null bitcoin
```

Las banderas m y d crean el directorio de inicio del usuario como se especifica en /***external_drive/ bitcoin*** en este caso. La bandera s asigna el shell interactivo del usuario. En este caso, lo configuramos en /***dev/ null*** para deshabilitar el uso de shell interactivo. El último argumento es el nombre de usuario bitcoin del nuevo usuario.

Inicio del nodo

Para los servicios de nodo Bitcoin y Lightning, la "instalación" también implica la creación de un ***script de inicio*** para asegurarse de que el nodo se inicie cuando se inicie la computadora. El inicio y cierre de los servicios en segundo plano está a cargo de un proceso del sistema operativo, que en Linux se denomina init o systemd.
Por lo general, puede encontrar un script de inicio del sistema en el subdirectorio contrib de cada proyecto. Por ejemplo, si tiene un sistema operativo Linux moderno que usa systemd, encontrará un script llamado ***bitcoind.service*** que puede iniciar y detener el servicio de nodo Bitcoin Core.

Aquí hay un ejemplo de cómo se ve el script de inicio de un nodo de Bitcoin, tomado del repositorio de código de Bitcoin Core:

```
[Unidad]
Descripción=demonio Bitcoin
After=network.target

[Servicio]
ExecStart=/usr/bin/bitcoind -daemon \ -pid=/run/bitcoind/
                                                bitcoind.pid \ -conf=/etc/bitcoin/bitcoin.conf \ -datadir=/
                                                var/lib/bitcoind


# Asegúrese de que el usuario del servicio pueda leer el directorio de configuración
PermisosStartOnly=verdadero
ExecStartPre=/bin/chgrp bitcoin /etc/bitcoin

# Gestión de proceso
####################

Tipo=bifurcación
PIDFile=/run/bitcoind/bitcoind.pid Restart=on-failure
```

```
Tiempo de esperaStopSec=600

# Creación de directorios y permisos
###################################

# Ejecutar como bitcoin:bitcoin
Usuario=bitcoin
Grupo=bitcoin

# /ejecutar/bitcoin
RuntimeDirectory=bitcoin
RuntimeDirectoryMode=0710

# /etc/bitcoin
Directorio de configuración=bitcoin
ConfigurationDirectoryMode=0710

# /var/lib/bitcoin
StateDirectory=bitcoin
EstadoModoDirectorio=0710

 [...]

 [Instalar]
WantedBy=multi-usuario.objetivo
```

Como usuario raíz, instale el script copiándolo en la carpeta de servicio systemd ***/lib/systemd/system/*** y luego vuelva a cargar systemd:

```
$ sudo systemctl daemon-recargar
```

A continuación, habilite el servicio:

```
$ sudo systemctl habilitar bitcoind
```

Ahora puede iniciar y detener el servicio. No lo inicies todavía, ya que no hemos configurado el nodo de Bitcoin.

```
$ sudo systemctl iniciar bitcoind $ sudo systemctl
detener bitcoind
```

Configuración de nodos

Para configurar su nodo, debe crear y hacer referencia a un archivo de configuración. Por convención, este archivo suele crearse en ***/etc,*** en un directorio con el nombre del programa. Por ejemplo, las configuraciones de Bitcoin Core y LND generalmente se almacenarían en ***/etc/bitcoin/bitcoin.conf*** y ***/etc/lnd/lnd.conf,*** respectivamente.

Estos archivos de configuración son archivos de texto en los que cada línea expresa una opción de configuración y su valor. Se asumen valores predeterminados para todo lo que no esté definido en el archivo de configuración. Puede ver qué opciones se pueden configurar en la configuración de dos maneras. Primero, ejecutar la aplicación de nodo con un argumento de ayuda mostrará las opciones que se pueden definir en la línea de comando. Estas mismas opciones se pueden definir en el archivo de configuración. En segundo lugar, normalmente puede encontrar un archivo de configuración de ejemplo, con todas las opciones predeterminadas, en el repositorio de código del software.

Puede encontrar un ejemplo de un archivo de configuración en cada una de las imágenes de Docker que usamos en el Capítulo 4. Por ejemplo, el archivo ***code/docker/bitcoind/bitcoind/bitcoin.conf:***

```
registro = 1
servidor = 1
debuglogfile=debug.log
depuración=1 txindex=1

imprimir en consola = 0

 [registro]
fallbackfee=0.000001
port=18444 noconnect=1

dnseed=0
DNS = 0
upnp=0
onlynet=ipv4
rpcport=18443
rpcbind=0.0.0.0
rpcallowip=0.0.0.0/0
rpcuser=regtest
rpcpassword=regtest
zmqpubrawblock=tcp://0.0.0.0:12005
zmqpubrawtx=tcp://0.0.0.0:12006
```

Ese archivo de configuración en particular configura Bitcoin Core para que funcione como un nodo de registro y proporciona un nombre de usuario y una contraseña débiles para el acceso remoto, por lo que no debe usarlo para la configuración de su nodo. Sin embargo, sirve para ilustrar la sintaxis de un archivo de configuración y puede realizar ajustes en el contenedor de Docker para experimentar con diferentes opciones. Vea si puede usar el comando bitcoind -help para comprender qué hace cada una de las opciones en el contexto de la red Docker que construimos en el Capítulo 4.

A menudo, los valores predeterminados son suficientes y, con algunas modificaciones, el software de su nodo se puede configurar rápidamente. Para ejecutar un nodo de Bitcoin Core con una personalización mínima, solo necesita cuatro líneas de configuración:

```
servidor=1
demonio=1
txindex=1
rpcuser=NOMBRE DE USUARIO rpcpassword=CONTRASEÑA
```

Incluso la opción txindex no es estrictamente necesaria, aunque garantizará que su nodo de Bitcoin cree un índice de todas las transacciones, lo cual es necesario para algunas aplicaciones. La opción txindex no es necesaria para ejecutar un nodo Lightning.

Un nodo Lightning c-lightning que se ejecuta en el mismo servidor también requiere solo unas pocas líneas en la configuración:

```
red = red principal
bitcoin-rpcuser=NOMBRE DE USUARIO
bitcoin-rpcpassword=CONTRASEÑA
```

En general, es una buena idea minimizar la cantidad de personalización de estos sistemas. La configuración predeterminada está cuidadosamente diseñada para admitir las implementaciones más comunes. Si modifica un valor predeterminado, puede causar problemas más adelante o reducir el rendimiento de su nodo. En resumen, ¡modifique solo cuando sea necesario!

configuración de la red

La configuración de la red normalmente no es un problema al configurar una nueva aplicación. Sin embargo, las redes peer-to-peer como Bitcoin y Lightning Network presentan algunos desafíos únicos para la configuración de la red.

En un servicio centralizado, su computadora se conecta a los "grandes servidores" de alguna corporación, y no al revés. La conexión a Internet de su hogar en realidad está configurada asumiendo que usted es simplemente un consumidor de servicios proporcionados por otros. Pero en un sistema de igual a igual, cada igual consume y proporciona servicios a otros nodos. Si está ejecutando un nodo Bitcoin o Lightning en su hogar, está brindando un servicio a otras computadoras en Internet. Su servicio de Internet por defecto no está configurado para permitirle ejecutar servidores y puede necesitar alguna configuración adicional para permitir que otros lleguen a su nodo.

Si desea ejecutar un nodo Bitcoin o Lightning, debe permitir que otros nodos en Internet se conecten a usted. Eso significa habilitar las conexiones TCP entrantes al puerto Bitcoin (puerto 8333 por defecto) o al puerto Lightning (puerto 9735 por defecto). Si bien puede ejecutar un nodo Bitcoin sin conectividad entrante, no puede hacerlo con un nodo Lightning. Un nodo Lightning debe ser accesible para otros desde fuera de su red.

De forma predeterminada, el enrutador de Internet de su hogar no espera conexiones entrantes desde el exterior y, de hecho, las conexiones entrantes están bloqueadas. La dirección IP de su enrutador de Internet es la única dirección IP accesible externamente, y todas las computadoras que ejecuta dentro de su red doméstica comparten esa única dirección IP. Esto se logra mediante un mecanismo llamado ***Traducción de direcciones de red (NAT),*** que permite que su enrutador de Internet actúe como intermediario para todas las conexiones salientes. Si desea permitir una conexión entrante, debe configurar el ***reenvío de puertos,*** que le dice a su enrutador de Internet que las conexiones entrantes en puertos específicos deben reenviarse a computadoras específicas dentro de la red. Puede hacerlo manualmente cambiando la configuración de su enrutador de Internet o, si su enrutador lo admite, a través de un mecanismo automático de reenvío de puertos llamado ***Universal Plug and Play (UPnP).***

Un mecanismo alternativo al reenvío de puertos es habilitar The Onion Router (Tor), que proporciona una especie de superposición de red privada virtual que permite las conexiones entrantes a una ***dirección de cebolla.*** Si ejecuta Tor, no necesita realizar el reenvío de puertos o habilitar las conexiones entrantes a los puertos Bitcoin o Lightning. Si ejecuta sus nodos con Tor, todo el tráfico pasa por Tor y no se utilizan otros puertos.

Veamos las diferentes formas en que puede hacer posible que otros se conecten a su nodo. Veremos estas alternativas en orden, de la más fácil a la más difícil.

¡Simplemente funciona!

Existe la posibilidad de que su proveedor de servicios de Internet o enrutador esté configurado para admitir UPnP de forma predeterminada y todo funcione automáticamente. Probemos este enfoque primero, en caso de que tengamos suerte.

Suponiendo que ya tiene un nodo Bitcoin o Lightning en ejecución, intentaremos ver si se puede acceder a ellos desde el exterior.

> **NOTA**
>
> Para que esta prueba funcione, debe tener un nodo Bitcoin o Lightning (o ambos) en funcionamiento en su red doméstica. Si su enrutador es compatible con UPnP, el tráfico entrante se reenviará automáticamente a los puertos correspondientes en la computadora que ejecuta el nodo.

Puede usar algunos sitios web muy populares y útiles para averiguar cuál es su dirección IP externa y si permite y reenvía conexiones entrantes a un puerto conocido. Aquí hay dos que son confiables:

- ***https://canyouseeme.org***
- ***https://www.whatismyip.com/port-scanner***

De forma predeterminada, estos servicios solo le permiten verificar las conexiones entrantes a la dirección IP desde la que se está conectando. Esto se hace para evitar que use el servicio para escanear las redes y computadoras de otras personas. Tú

verá la dirección IP externa de su enrutador y un campo para ingresar un número de puerto. Si no ha cambiado los puertos predeterminados en la configuración de su nodo, pruebe con el puerto 8333 (Bitcoin) y/o 9735 (Lightning).

En la Figura 5-3 , puede ver el resultado de verificar el puerto 9735 en un servidor que ejecuta Lightning, utilizando la herramienta de análisis de puertos ***whatismyip.com .*** Muestra que el servidor está aceptando conexiones entrantes al puerto Lightning. Si ves un resultado como este, ¡estás listo!

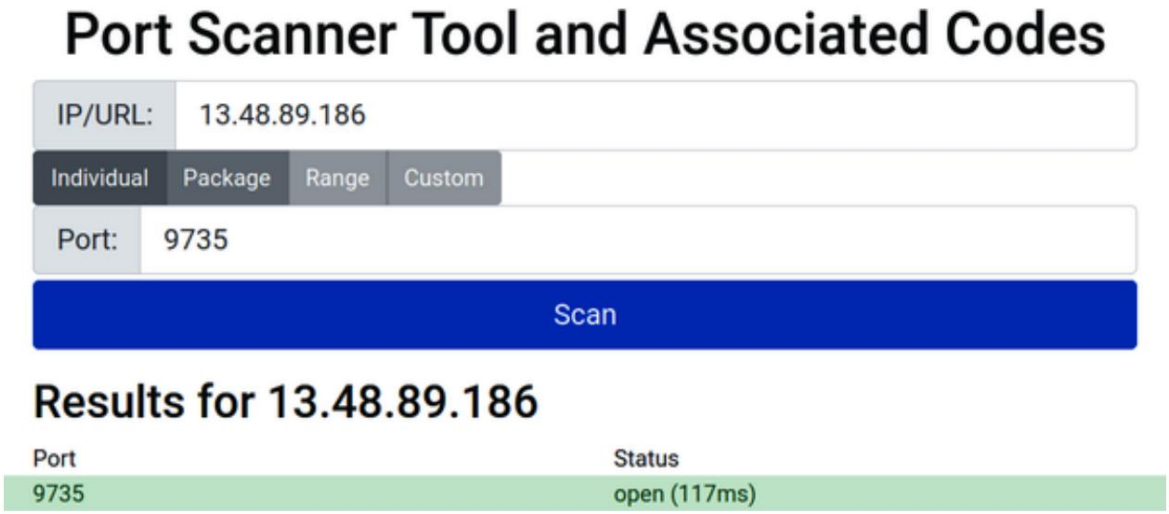


Figura 5-3. Comprobación del puerto de entrada 9735

Reenvío automático de puertos usando UPnP A veces, incluso si su enrutador de Internet es compatible con UPnP, puede estar desactivado de forma predeterminada. En ese caso, debe cambiar la configuración de su enrutador de Internet desde su interfaz de administración web:

1. Conéctese al sitio web de configuración de su enrutador de Internet. Por lo general, esto se puede hacer conectándose a la ***dirección de la puerta*** de enlace de su red doméstica mediante un navegador web. Puede encontrar la dirección de la puerta de enlace mirando la configuración IP de cualquier computadora en su red doméstica. A menudo es la primera dirección en una de las redes no enrutables, como 192.168.0.1 o 10.0.0.1. Verifique todas las pegatinas en su enrutador también para la ***dirección de la puerta de enlace.*** Una vez encontrado, abra un navegador e ingrese la dirección IP en el cuadro de búsqueda/URL del navegador, por ejemplo, "192.168.0.1" o "http://192.168.0.1".

2. Encuentra el nombre de usuario y la contraseña del administrador para la web panel de configuración del router. Esto a menudo está escrito en una etiqueta en el enrutador y puede ser tan simple como "administrador" y "contraseña". Una búsqueda rápida en la web de su ISP y modelo de enrutador también puede ayudarlo a encontrar esta información.

3. Busque una configuración para UPnP y actívela.

Reinicie su nodo Bitcoin y/o Lightning y repita la prueba de puerto abierto con uno de los sitios web que usamos en la sección anterior.

Uso de Tor para conexiones entrantes

El Onion Router (Tor) es una VPN con la particularidad de que cifra las comunicaciones entre saltos, de forma que ningún nodo intermediario pueda determinar el origen o el destino de un paquete. Tanto los nodos Bitcoin como Lightning admiten la operación sobre Tor, lo que le permite operar un nodo sin revelar su dirección IP o ubicación. Por lo tanto, proporciona un alto nivel de privacidad al tráfico de su red. Un beneficio adicional de ejecutar Tor es que, debido a que funciona como una VPN, resuelve el problema del reenvío de puertos desde su enrutador de Internet. Las conexiones entrantes se reciben a través del túnel Tor y su nodo se puede encontrar a través de una ***dirección de cebolla*** generada ad hoc en lugar de una dirección IP.

Habilitar Tor requiere dos pasos. Primero, debe instalar el enrutador Tor y el proxy en su computadora. En segundo lugar, debe habilitar el uso del proxy Tor en su configuración de Bitcoin o Lightning.

Para instalar Tor en un sistema Ubuntu Linux que usa el paquete apt gerente, ejecute:

```
sudo apt install tor
```

A continuación, configuramos nuestro nodo Lightning para usar Tor para su conectividad externa. Aquí hay una configuración de ejemplo para LND:

```
[Tor]
tor.activo=verdadero
```

```
tor.v3=verdadero
tor.streamisolation=true
escucha=localhost
```

Esto habilitará Tor (tor.active), establecerá un servicio de cebolla v3 (tor.v3=true), usará un flujo de cebolla diferente para cada conexión (tor.streamisolation) y restringirá la escucha de conexiones solo al host local, para evitar filtrando su dirección IP (escuchar = localhost).

Puede verificar si Tor está correctamente instalado y funcionando ejecutando un simple comando de una línea. Este comando debería funcionar en la mayoría de las versiones de Linux:

```
curl --socks5 localhost:9050 --socks5-nombre de host localhost:9050 -s
https://check.torproject.org/ | gato | grep -m 1 Felicitaciones |
xargs
```

Si todo funciona correctamente, la respuesta de este comando debería ser "Felicitaciones. Este navegador está configurado para usar Colina."

Debido a la naturaleza de Tor, no puede usar fácilmente un servicio externo para verificar si se puede acceder a su nodo a través de una dirección de cebolla. No obstante, debería ver su dirección de cebolla Tor en los registros de su nodo Lightning. Es una larga cadena de letras y números seguida del sufijo .onion. Su nodo ahora debería ser accesible desde Internet, ¡con la ventaja adicional de la privacidad!

Reenvío manual de puertos

Este es el proceso más complejo y requiere bastante habilidad técnica.
Los detalles dependen del tipo de enrutador de Internet que tenga, la configuración y las políticas de su proveedor de servicios y muchos otros contextos. Pruebe UPnP o Tor primero, antes de probar este mecanismo mucho más difícil.

Los pasos básicos son los siguientes:

1. Busque la dirección IP de la computadora en la que se encuentra su nodo. Esto generalmente lo asigna dinámicamente el Protocolo de configuración dinámica de host (DHCP) y, a menudo, se encuentra en algún lugar de 192.168.xx o 10.xxx. rango.

2. Busque la dirección de control de acceso a medios (MAC) de su nodo
interfaz de red. Esto se puede encontrar en la configuración de Internet de esa computadora.

3. Asigna una dirección IP estática a tu nodo para que sea siempre la misma. Puede
utilizar la dirección IP que tiene actualmente. En su enrutador de Internet, busque
"Arrendamientos estáticos" en la configuración de DHCP. Asigne la dirección MAC a
la dirección IP que seleccionó. Ahora su nodo siempre tendrá asignada esa dirección
IP. Alternativamente, puede mirar la configuración DHCP de su enrutador y averiguar cuál
es su rango de direcciones DHCP. Seleccione una dirección no utilizada ***fuera*** del rango de
direcciones DHCP. Luego, en el servidor, configure la red para que deje de usar DHCP y
codifique la dirección IP no DHCP seleccionada en la configuración de red del sistema
operativo.

4. Finalmente, configure "Reenvío de puertos" en su enrutador de Internet para enrutar
tráfico entrante en puertos específicos a la dirección IP seleccionada de su
servidor.

Una vez que haya terminado de reconfigurar, repita la verificación del puerto utilizando uno de los sitios web de las secciones anteriores.

Seguridad de su nodo

Un nodo Lightning es, por definición, una ***billetera caliente.*** Eso significa que los fondos (tanto dentro como fuera de la cadena) controlados por un nodo Lightning están controlados directamente por claves que se cargan en la memoria del nodo o se almacenan en el disco duro del nodo. Si un nodo Lightning está comprometido, es trivial crear transacciones dentro o fuera de la cadena para drenar sus fondos. Por lo tanto, es sumamente importante que lo proteja del acceso no autorizado.

La seguridad es un esfuerzo holístico, lo que significa que debe proteger cada capa de un sistema. Como dice el refrán: la cadena es tan fuerte como el eslabón más débil.
Este es un concepto importante en la seguridad de la información y lo aplicaremos a nuestro nodo.

A pesar de todas las medidas de seguridad que tomará, recuerde que Lightning Network es una tecnología experimental en etapa inicial y es probable que haya errores explotables en el código de cualquier proyecto que use. ***No ponga más dinero del que está dispuesto a arriesgar a perder en Lightning Network.***

Seguridad del sistema operativo

Asegurar un sistema operativo es un tema amplio que está más allá del alcance de este libro. Sin embargo, podemos establecer algunos principios básicos.

Para asegurar su sistema operativo, estos son algunos de los principales elementos a considerar:

Procedencia

Comience por asegurarse de que está descargando la imagen del sistema operativo correcta y verifique las firmas o sumas de verificación antes de instalarlo.
Extiende esto a cualquier software que instales. Vuelva a verificar cualquier fuente o URL desde donde descargue. Verifique la integridad y la corrección del software descargado a través de la verificación de la firma y la suma de verificación.

Mantenimiento

Asegúrate de mantener tu sistema operativo actualizado. Habilite la instalación automática diaria o semanal de actualizaciones de seguridad. Privilegio mínimo: configura usuarios para procesos específicos y dales el acceso mínimo necesario para ejecutar un servicio. No ejecute procesos con privilegios de administrador (p. ej., raíz).

Aislamiento de procesos

Utilice las funciones del sistema operativo para aislar los procesos entre sí.

Permisos del sistema de archivos

Configure el sistema de archivos con cuidado, según el principio de privilegios mínimos. No haga que los archivos sean legibles o escribibles por todos.

Autenticación fuerte

Utilice contraseñas seguras generadas aleatoriamente o, siempre que sea posible, autenticación de clave pública. Por ejemplo, es más seguro usar Secure Shell (SSH) con un par de claves criptográficas en lugar de una contraseña.

Autenticación de dos factores (2FA)

Utilice la autenticación de dos factores siempre que sea posible, incluido el segundo factor universal (U2F) con claves de seguridad de hardware. Esto se aplica a todos los servicios externos que pueda estar utilizando, como su proveedor de servicios en la nube. Puede aplicar esto también a su propia configuración, como su propia configuración de SSH. Utilice 2FA también para servicios indirectos. Por ejemplo, supongamos que está utilizando un servicio en la nube. Le dio a su proveedor de servicios en la nube una dirección de correo electrónico, por lo que también debe proteger su dirección de correo electrónico con 2FA.

Respaldo

Realice copias de seguridad de su sistema y asegúrese de proteger las copias de seguridad con cifrado también. Realice estas copias de seguridad periódicamente. Al menos una vez, pruebe si puede restaurar su copia de seguridad y si está completa y accesible. Si es posible, guarde una copia de sus copias de seguridad en un disco diferente para evitar que una sola falla del disco duro destruya ***tanto*** su nodo activo como sus copias de seguridad.

Gestión de vulnerabilidades y exposiciones

Utilice el escaneo remoto para asegurarse de haber minimizado la superficie de ataque de su sistema. Cierre todos los servicios o puertos innecesarios. Instale solo el software y los paquetes que realmente necesita y usa. Desinstala los paquetes que ya no usas. Se recomienda que ***no*** use su computadora de nodo para actividades que no sean de nodo que pueda realizar en otra de sus computadoras. Especialmente, si puede, ***no*** use su computadora nodo para navegar, navegar por Internet o leer su correo electrónico.

Esta es una lista de las medidas de seguridad más básicas. De ninguna manera es exhaustivo.

Acceso al nodo

Su nodo Lightning expondrá una API de llamada a procedimiento remoto (RPC). Esto significa que su nodo puede controlarse de forma remota mediante comandos enviados a un puerto TCP específico. El control de acceso a esa API de RPC se logra mediante alguna forma de autenticación de usuario. Según el tipo de nodo Lightning que configure, esto se realizará mediante autenticación de nombre de usuario/contraseña o mediante un mecanismo llamado ***macarrón de autenticación.*** Como su nombre lo indica, un macarrón es un tipo de galleta más sofisticado. A diferencia de una cookie, está firmada criptográficamente y puede expresar un conjunto de capacidades de acceso.

Por ejemplo, LND usa macarrones para otorgar acceso a la API de RPC. De forma predeterminada, el software LND crea tres macarrones con diferentes niveles de acceso, llamados administrador, factura y solo lectura. Según el macarrón que copie y use en su cliente RPC, tendrá acceso ***de solo lectura ,*** acceso a ***facturas*** (que incluye las capacidades de solo lectura) o acceso de ***administrador ,*** lo que le brinda control total. También hay una función de panadería de macarrones en LND que puede construir macarrones con cualquier combinación de capacidades con un control muy detallado.

Si utiliza un modelo de autenticación de nombre de usuario/contraseña, asegúrese de seleccionar una contraseña larga y aleatoria. No tendrá que escribir esta contraseña a menudo, ya que se almacenará en los archivos de configuración. Por lo tanto, debe elegir uno que no se pueda adivinar. Muchos de los ejemplos que verá incluyen contraseñas mal elegidas y, a menudo, las personas las copian en sus propios sistemas, lo que facilita el acceso a cualquier persona. ¡No hagas eso! Utilice un administrador de contraseñas para generar una contraseña alfanumérica aleatoria larga. Dado que ciertos caracteres especiales como $?/!*\&%"" pueden interferir con la línea de comando, es mejor evitarlos para las contraseñas que se usarán en un entorno de shell. Para evitar problemas, siga con caracteres alfanuméricos aleatorios largos. contraseñas

Una secuencia alfanumérica simple que tenga más de 12 caracteres y se genere aleatoriamente suele ser suficiente. Si planea almacenar grandes cantidades de dinero en su nodo Lightning y le preocupan los ataques remotos de fuerza bruta, seleccione una longitud de contraseña de más de 20 caracteres para que dichos ataques sean prácticamente inviables.

Copias de seguridad de nodos y canales

Una consideración muy importante al ejecutar un nodo Lightning es el tema de las copias de seguridad. A diferencia de una billetera Bitcoin, donde una frase mnemotécnica BIP-39 puede recuperar todo el estado de la billetera, en Lightning este ***no*** es el caso.

Las billeteras Lightning usan una copia de seguridad de frase mnemotécnica BIP-39, pero solo para la billetera en cadena. Sin embargo, debido a la forma en que se construyen los canales, la frase mnemotécnica ***no*** es suficiente para restaurar un nodo Lightning. Se necesita una capa adicional de copias de seguridad, que se denomina ***copia de seguridad de canal estático (SCB).*** Sin un SCB, un operador de nodo Lightning puede perder ***todos*** los fondos que están en los canales si pierde el almacén de datos del nodo Lightning.

ADVERTENCIA

No financie canales hasta que haya implementado un sistema para respaldar continuamente el estado de su canal . Sus copias de seguridad se deben mover "fuera del sitio" a un sistema y una ubicación diferentes de su nodo, para que puedan sobrevivir a una variedad de fallas del sistema (pérdida de energía, corrupción de datos, etc.) o desastres naturales (inundaciones, incendios, etc.).

Los SCB no son una panacea. En primer lugar, es necesario realizar una copia de seguridad del estado de cada canal cada vez que haya una nueva transacción de compromiso. En segundo lugar, la restauración desde una copia de seguridad del canal es peligrosa. Si no tiene la ***última*** transacción de compromiso y transmite accidentalmente un compromiso anterior (revocado), su compañero de canal asumirá que está tratando de hacer trampa y reclamará el saldo completo del canal con una transacción de penalización. Para asegurarse de que está cerrando el canal, debe hacer un ***cierre cooperativo.*** Pero un par malicioso podría engañar a su nodo para que transmita un compromiso antiguo y revocado durante ese cierre cooperativo, engañándolo al hacer que su nodo intente hacer trampa sin darse cuenta.

Además, las copias de seguridad de sus canales deben cifrarse para mantener su privacidad y la seguridad de su canal. De lo contrario, cualquier persona que encuentre las copias de seguridad no solo puede ver todos sus canales, sino que también podría usar las copias de seguridad para cerrar todos sus canales de una manera que le entregue el saldo a su canal.

colegas. En otras palabras, una persona malintencionada que obtenga acceso a sus copias de seguridad puede hacer que pierda todos los fondos de su canal.

Puede ver que los SCB no son una protección infalible. Son un compromiso débil porque intercambian un tipo de riesgo (corrupción o pérdida de datos) por otro tipo de riesgo (par malicioso). Para restaurar desde un SCB, debe interactuar con sus compañeros de canal y esperar que no intenten engañarlo brindándole un compromiso anterior o engañando a su nodo para que transmita un compromiso revocado para que puedan penalizarlo. A pesar de las debilidades de SCB, los SCB tienen sentido y debe realizarlos. Si no realiza SCB y pierde los datos de su nodo, perderá los fondos de su canal para siempre. ¡Garantizado! Sin embargo , si realiza SCB y pierde los datos de su nodo, entonces tiene una posibilidad razonable de que algunos de sus pares sean honestos y que pueda recuperar algunos de los fondos de su canal.

Si tiene suerte, puede recuperar todos sus fondos. En conclusión, lo mejor para usted es realizar SCB continuos en un disco que no sea el disco duro del nodo principal.

Los mecanismos de copia de seguridad del canal siguen siendo un trabajo en progreso y una debilidad en la mayoría de las implementaciones de Lightning.

Al momento de escribir este libro, solo LND ofrece un mecanismo incorporado para SCB. Eclair tiene un mecanismo similar implementado para implementaciones del lado del servidor, aunque Eclair Mobile ofrece una copia de seguridad opcional en Google Drive. c-lightning fusionó recientemente las interfaces necesarias para un complemento para implementar copias de seguridad de canales. Desafortunadamente, no existe un mecanismo de respaldo consistente y acordado en las diferentes implementaciones de nodos.

Las copias de seguridad basadas en archivos de las bases de datos del nodo Lightning son, en el mejor de los casos, una solución parcial porque corre el riesgo de hacer una copia de seguridad de un estado de base de datos incoherente. Además, es posible que no capte de manera confiable los últimos compromisos estatales. Es mucho mejor tener un mecanismo de copia de seguridad que se activa cada vez que hay un cambio de estado en un canal, lo que garantiza la coherencia de los datos.

Para configurar SCB en LND, establezca el parámetro ruta del archivo de copia de seguridad en la línea de comando o en el archivo de configuración. LND luego guardará un archivo SCB en esa ruta de directorio. Por supuesto, ese es solo el primer paso de la solución.

Ahora, debe configurar un mecanismo que supervise este archivo en busca de cambios. Cada vez que el archivo cambia, el mecanismo de copia de seguridad debe copiar este archivo en otro disco, preferiblemente fuera del sitio. Dichos mecanismos de respaldo están más allá del alcance de este libro. No obstante, cualquier solución de respaldo sofisticada debería poder manejar este escenario. Recuerde, los archivos de respaldo también deben estar encriptados.

Riesgo de billetera caliente

Como hemos discutido anteriormente, Lightning Network consiste en una red de ***billeteras calientes.*** Los fondos que almacena en una billetera Lightning están en línea todo el tiempo. Esto los hace vulnerables. Por lo tanto, no debe almacenar grandes cantidades en una billetera Lightning. Las grandes cantidades deben guardarse en una billetera ***fría*** que ***no*** esté en línea y que solo pueda realizar transacciones en la cadena.

Incluso si comienza poco a poco, a medida que pasa el tiempo, es posible que aún tenga una cantidad significativa de dinero en una billetera Lightning. Este es un escenario típico para los propietarios de tiendas. Si usa un nodo Lightning para una operación de comercio electrónico, es probable que su billetera reciba fondos con frecuencia, pero los envíe rara vez. Por lo tanto, terminará teniendo dos problemas simultáneamente. Primero, sus canales estarán desequilibrados, con saldos locales grandes que superan los saldos remotos pequeños. En segundo lugar, tendrás demasiado dinero en la billetera.
Afortunadamente, también puede resolver ambos problemas simultáneamente.

Veamos algunas de las soluciones que puede usar para reducir los fondos expuestos en una billetera caliente.

Fondos de barrido

Si el saldo de su billetera Lightning se vuelve demasiado grande para su tolerancia al riesgo, deberá "barrer" los fondos de la billetera. Puede hacerlo de tres maneras: en cadena, fuera de cadena y Loop Out. Veamos cada una de estas opciones en las próximas secciones.

Barrido en cadena

El barrido de fondos en la cadena se logra moviendo los fondos de la billetera Lightning a una billetera Bitcoin. Lo haces cerrando canales. Cuando cierra un canal, todos los fondos de su saldo local se "barren" a una dirección de Bitcoin. La dirección de Bitcoin para los fondos en cadena generalmente la genera su billetera Lightning, por lo que sigue siendo una billetera caliente. Es posible que deba realizar una transacción adicional en la cadena para mover los fondos a una dirección más segura, como una generada en su billetera de hardware.

El cierre de canales incurrirá en una tarifa en cadena y reducirá la capacidad y conectividad de su nodo Lightning. Sin embargo, si ejecuta un nodo de comercio electrónico popular, no le faltará capacidad entrante y puede cerrar canales estratégicamente con grandes saldos locales, esencialmente "agrupando" sus fondos para el movimiento en la cadena. Es posible que deba utilizar algunas técnicas de reequilibrio de canales (consulte “Reequilibrio de canales”) antes de cerrar canales para maximizar los beneficios de esta estrategia.

Barrido fuera de la cadena

Otra técnica que puede usar consiste en ejecutar un segundo nodo Lightning que no se anuncia en la red. Puede establecer canales de gran capacidad desde su nodo público (por ejemplo, el que administra su tienda) hasta su nodo no anunciado (oculto). De forma regular, "barrer" fondos haciendo un pago Lightning a su nodo oculto.

La ventaja de esta técnica radica en que se conocerá públicamente el nodo Lightning que recibe los pagos de tu tienda. Esto lo convierte en un objetivo para los piratas informáticos, ya que se supondría que cualquier nodo Lightning asociado con una tienda tiene un gran saldo. Un segundo nodo que no esté asociado con su tienda no se identificará fácilmente como un objetivo valioso.

Como medida adicional de seguridad, puedes convertir tu segundo nodo en un servicio Tor oculto para que no se conozca su dirección IP. Eso reduce aún más la oportunidad de ataques y aumenta su privacidad.

Deberá configurar un script que se ejecute a intervalos regulares. El propósito de este script es crear una factura en su nodo oculto y pagar esa factura desde el nodo de su tienda, transfiriendo así los fondos a su nodo oculto.

Tenga en cuenta que esta técnica no mueve fondos al almacenamiento en frío.
Ambos nodos Lightning son monederos activos. El objetivo de este barrido es mover fondos de una billetera caliente muy conocida a una billetera caliente poco conocida.

Barrido de intercambio submarino

Otra forma de reducir el saldo de su billetera caliente Lightning es usar una técnica llamada ***intercambio submarino.*** Los intercambios submarinos, conceptualizados por el coautor Olaoluwa Osuntokun y Alex Bosworth, permiten el intercambio de bitcoin en cadena por pagos Lightning y viceversa. Esencialmente, los intercambios submarinos son intercambios atómicos entre fondos fuera de la cadena Lightning y fondos dentro de la cadena Bitcoin.

Un operador de nodo puede iniciar un intercambio submarino y enviar todos los saldos de canales disponibles a la otra parte, quien a cambio les enviará bitcoins en cadena.

En el futuro, este podría ser un servicio pago ofrecido por nodos en Lightning Network que anuncian tipos de cambio o cobran una tarifa fija por la conversión.

La ventaja de un intercambio submarino por fondos de barrido es que no es necesario cerrar ningún canal. Eso significa que preservamos nuestros canales, solo reequilibrando nuestros canales a través de esta operación. Cuando enviamos un pago Lightning, cambiamos parte del saldo de local a remoto en uno o más de nuestros canales. Eso no solo reduce el saldo expuesto en la billetera caliente de nuestro nodo, sino que también aumenta el saldo disponible para futuros pagos entrantes.

Puede hacer esto confiando en un intermediario para que actúe como puerta de enlace, pero esto corre el riesgo de que le roben sus monedas. Sin embargo, en el caso de un intercambio de submarinos, la operación no requiere confianza. Los intercambios de submarinos son operaciones ***atómicas*** sin custodia . Eso significa que la contraparte en su intercambio submarino no puede robar sus fondos porque el pago dentro de la cadena depende de la finalización del pago fuera de la cadena y viceversa.

Intercambios submarinos con Loop

Un ejemplo de un servicio de intercambio de submarinos es ***Loop*** de Lightning Labs, la misma empresa que construye LND. Loop viene en dos variaciones: Loop In y Loop Out. ***Loop In*** acepta un pago en cadena de Bitcoin y lo convierte en un pago fuera de cadena de Lightning. ***Loop Out*** convierte un pago Lightning en un pago Bitcoin.

> **NOTA**
>
> Para utilizar el servicio Loop, debe ejecutar un nodo LND Lightning.

Con el propósito de reducir el saldo de su billetera caliente Lightning, usaría el servicio Loop Out. Para usar el servicio Loop, debe instalar algún software adicional en su nodo. El software Loop se ejecuta junto con su nodo LND y proporciona algunas herramientas de línea de comandos para ejecutar intercambios submarinos. Puede encontrar el software Loop y las instrucciones de instalación en GitHub.

Una vez que tenga el software instalado y ejecutándose, una operación de Loop Out es tan simple como ejecutar un solo comando:

```
bucle de salida --amt 501000 --conf_target 400 Tarifas máximas de
intercambio para 501000 sat Bucle de salida: 25716 sat Se solicitó una velocidad de
intercambio regular, puede tomar hasta 30 m0s para que se ejecute el intercambio.

¿CONTINUAR CAMBIO? (sí/no), ampliar el detalle de la tarifa (x): x

Tarifa estimada de barrido en cadena: Tarifa                149 sábado
máxima de barrido en cadena: Tarifa máxima de               14900 sáb
enrutamiento de intercambio fuera de cadena:                10030 sábado
Penalización máxima por no presentarse (prepago):           1337 sábado
Tarifa máxima de enrutamiento de prepago fuera de           36 sábado
cadena: Tarifa máxima de intercambio: ¿CONTINUAR            750 sáb
INTERCAMBIO? (s/n): y Intercambio iniciado


Ejecute `loop monitor` para monitorear el progreso.
```

Tenga en cuenta que su tarifa máxima, que representa el peor de los casos, dependerá del objetivo de confirmación que seleccione.

Tiempo de actividad y disponibilidad del nodo Lightning

A diferencia de Bitcoin, los nodos Lightning deben estar en línea casi continuamente. Su nodo debe estar en línea para recibir pagos, abrir canales, cerrar canales (cooperativamente) y monitorear violaciones de protocolo. La disponibilidad de nodos es un requisito tan importante en Lightning Network que es una métrica utilizada por varias herramientas de administración automática de canales (por ejemplo, piloto automático) para decidir con qué nodos abrir canales. También puede ver la "disponibilidad" como una métrica de nodo en exploradores de nodos populares (consulte "Exploradores Lightning") como 1ML.

La disponibilidad de nodos es especialmente importante para mitigar y resolver posibles infracciones de protocolo (es decir, compromisos revocados). Si bien puede permitirse interrupciones breves desde una hora hasta uno o dos días, no puede tener su nodo fuera de línea durante períodos más prolongados sin correr el riesgo de perder fondos.

Mantener un nodo en línea de forma continua no es fácil, ya que varios errores y limitaciones de recursos pueden y, en ocasiones, provocarán tiempo de inactividad. Especialmente si ejecuta un nodo ocupado y popular, se encontrará con limitaciones de memoria, espacio de intercambio, cantidad de archivos abiertos, espacio en disco, etc. Una gran cantidad de problemas diferentes harán que su nodo o su servidor se bloqueen.

Tolerar fallas y automatizar

Si tiene el tiempo y las habilidades, debe probar algunos escenarios de falla básicos en la red de prueba Lightning. En la red de prueba, aprenderá lecciones valiosas sin arriesgar ningún dinero. Cualquier paso que realice para automatizar su sistema mejorará su disponibilidad:

Reinicio automático del servidor de la computadora

> ¿Qué sucede cuando su servidor o el sistema operativo fallan? ¿Qué sucede cuando hay un corte de energía? Simule esta falla presionando el botón "reiniciar" en su PC o desconectando el cable de alimentación. Después de un bloqueo, reinicio o falla de energía, la computadora debería reiniciarse automáticamente. Algunas computadoras tienen una configuración en su BIOS para especificar cómo debe reaccionar la computadora ante fallas de energía. Pruébelo para asegurarse de que

la computadora realmente se reinicia automáticamente en caso de corte de energía sin intervención humana.

Reinicio automático de nodos

¿Qué sucede cuando su nodo o uno de sus nodos falla? Simule esta falla eliminando los procesos de nodo correspondientes. Si un nodo falla, debería reiniciarse automáticamente. Pruébelo para asegurarse de que el nodo o los nodos realmente se reinician automáticamente en caso de falla sin intervención humana.
Si este no es el caso, lo más probable es que su nodo no esté configurado correctamente como un servicio del sistema operativo.

Reconexión automática de red

¿Qué sucede si su red se cae? ¿Qué sucede cuando su ISP se cae temporalmente? ¿Qué sucede cuando su ISP asigna una nueva dirección IP a su enrutador o su computadora? Cuando la red vuelve, ¿los nodos que está ejecutando se vuelven a conectar automáticamente a la red? Simule esta falla desconectando y luego volviendo a conectar el cable Ethernet del dispositivo que aloja sus nodos. Los nodos deberían reconectarse automáticamente y continuar la operación sin intervención humana.

Configura tus archivos de registro

Todos los errores anteriores deberían dejar entradas de texto en los archivos de registro correspondientes. Aumente la verbosidad del registro si es necesario. Encuentre estas entradas de error en los archivos de registro y utilícelas para monitorear.

Supervisión de la disponibilidad del nodo

Supervisar su nodo es una parte importante para mantenerlo en funcionamiento. Debe monitorear no solo la disponibilidad de la computadora en sí, sino también la disponibilidad y el correcto funcionamiento del software del nodo Lightning.

Hay varias formas de hacer esto, pero la mayoría requiere cierta personalización.
Puede usar herramientas genéricas de monitoreo de infraestructura o de monitoreo de aplicaciones, pero debe personalizarlas específicamente para consultar Lightning

API de nodo para garantizar que el nodo se esté ejecutando, sincronizado con la cadena de bloques y conectado a los pares del canal.

Relámpago.reloj brinda un servicio especializado que ofrece monitoreo de nodos Lightning. Utiliza un bot de Telegram para notificarle cualquier interrupción en el servicio. Este es un servicio gratuito, aunque puede pagar (a través de Lightning, por supuesto) para recibir alertas más rápido.

Con el tiempo, esperamos que más servicios de terceros brinden monitoreo especializado de nodos Lightning pagable a través de micropagos. Quizás dichos servicios y sus API se estandaricen y algún día sean compatibles directamente con el software del nodo Lightning.

Atalayas

Las torres de ***vigilancia*** son un mecanismo para subcontratar el monitoreo y la resolución de sanciones de violaciones del protocolo Lightning.

Como mencionamos en capítulos anteriores, el protocolo Lightning mantiene la seguridad a través de un mecanismo de penalización. Si uno de sus socios de canal transmite una transacción de compromiso anterior, su nodo deberá ejercer la cláusula de revocación y transmitir una transacción de penalización para evitar perder dinero. Pero si su nodo está inactivo durante la violación del protocolo, podría perder dinero.

Para resolver este problema, podemos utilizar una o más torres de vigilancia para subcontratar el trabajo de monitorear las violaciones del protocolo y emitir transacciones de penalización. Hay dos partes en la configuración de una torre de vigilancia: un servidor de torre de vigilancia (o simplemente torre de vigilancia) que monitorea la cadena de bloques y un cliente de torre de vigilancia que solicita al servidor de torre de vigilancia este servicio de monitoreo.

La tecnología Watchtower aún se encuentra en las primeras etapas de desarrollo y no cuenta con un amplio respaldo. Sin embargo, en el siguiente pasaje enumeramos algunas implementaciones experimentales que puede probar.

El software LND incluye un servidor Watchtower y un cliente Watchtower.

Puede activar el servidor Watchtower agregando las siguientes opciones de configuración:

```
[atalaya]
atalaya.active=1
watchtower.towerdir=/path_to_watchtower_data_directory
```

Puede usar el cliente de torre de vigilancia de LND activándolo en la configuración y luego usando la línea de comando para conectarlo a un servidor de torre de vigilancia. La configuración es:

```
[wtclient]
wtclient.active=1
```

El cliente de línea de comandos de LND, lncli, muestra las siguientes opciones para administrar el cliente Watchtower:

```
$ lncli wtcliente

NOMBRE:
    lncli wtclient: interactúa con el cliente de Watchtower.

USO:
    comando lncli wtclient [opciones de comando] [argumentos...]

COMANDOS:
    agregar Registre una torre de vigilancia para usar en futuras sesiones/
copias de seguridad.
    eliminar Eliminar una torre de vigilancia para evitar su uso en el futuro
sesiones/copias de seguridad.
    torres Muestra información sobre todas las torres de vigilancia registradas. torre Muestra información
    sobre una torre de vigilancia registrada específica. stats Muestra las estadísticas de la sesión del
cliente Watchtower. política Mostrar la política activa del cliente Watchtower


configuración.

OPCIONES:
    --ayuda, -h mostrar ayuda
```

c-lightning tiene los enlaces de API necesarios para un complemento de cliente de Watchtower, aunque aún no se ha implementado dicho complemento.

Finalmente, un popular servidor de torre de vigilancia independiente es ***The Eye of Satoshi*** (TEOS). Se puede encontrar en GitHub.

Gestión de canales

Como operador de un nodo Lightning, una de las tareas recurrentes que deberá realizar es la gestión de sus canales. Esto significa abrir canales de salida desde su nodo a otros nodos, así como hacer que otros nodos abran canales de entrada a su nodo. En el futuro, la construcción de canales cooperativos puede ser posible, por lo que puede abrir canales simétricos que tengan fondos comprometidos en ambos extremos en la creación. Por ahora, sin embargo, los nuevos canales solo tienen fondos en un extremo, en el lado del originador. Por lo tanto, para ***equilibrar*** su nodo con capacidad tanto de entrada como de salida, debe abrir canales a otros y atraer a otros para que abran canales a su nodo.

Apertura de canales de salida

Tan pronto como ponga en funcionamiento su nodo Lightning, puede financiar su billetera Bitcoin y luego comenzar a abrir canales con esos fondos.

Debe elegir cuidadosamente a los socios de canal porque la capacidad de su nodo para enviar pagos depende de quiénes son sus socios de canal y qué tan bien conectados están con el resto de Lightning Network. También desea tener más de un canal para evitar ser susceptible a un único punto de falla. Dado que Lightning ahora admite pagos de varias partes, puede dividir sus fondos iniciales en varios canales y enrutar pagos más grandes combinando su capacidad. Al mismo tiempo, evite hacer sus canales demasiado pequeños. Dado que debe pagar tarifas de transacción de Bitcoin para abrir y cerrar un canal, el saldo del canal no debe ser tan pequeño como para que las tarifas en cadena consuman una parte significativa. ¡Todo es cuestión de equilibrio!

Para resumir:

- Conéctese a algunos nodos bien conectados
- Abrir más de un canal
- No abras demasiados canales
- No hagas los canales demasiado pequeños.

Una forma de encontrar nodos bien conectados es abrir un canal a un comerciante popular que venda productos en Lightning Network. Estos nodos tienden a estar bien financiados y bien conectados. Entonces, cuando esté listo para comprar algo en línea a través de Lightning, puede abrir un canal directamente al nodo del comerciante. El ID de nodo del comerciante estará en la factura que recibirá cuando intente comprar algo. Eso lo hace fácil.

Otra forma de encontrar nodos bien conectados es usar un Lightning Explorer (ver "Lightning Explorers") como 1ML y explore la lista de nodos ordenados por capacidad de canal y número de canales. No opte por los nodos más grandes, porque eso fomenta la centralización. Busque un nodo en el medio de la lista para que pueda ayudarlos a crecer. Otro factor a considerar podría ser el lapso de tiempo que un nodo ha estado en funcionamiento. Es probable que los nodos establecidos durante más de un año sean más confiables y menos riesgosos que los nodos que comenzaron a operar hace una semana.

Piloto automático

La tarea de abrir canales se puede automatizar parcialmente con el uso de un ***piloto automático,*** que es un software que abre canales automáticamente en función de algunas reglas heurísticas. El software de piloto automático todavía es relativamente nuevo y no siempre selecciona los mejores socios de canal para usted. Especialmente al principio, podría ser mejor abrir los canales manualmente. Los pilotos automáticos existen actualmente en tres formas:

- lnd incorpora un piloto automático que está completamente integrado con lnd y se ejecuta constantemente en segundo plano mientras está encendido.
- lib_autopilot.py puede ofrecer cálculos de piloto automático para cualquier implementación de nodo en función de los chismes y los datos del canal.
- Existe un complemento de c-lightning basado en lib_autopilot.py que proporciona una interfaz fácil de usar para los usuarios de c-lightning.

Tenga en cuenta que el piloto automático lnd comenzará a ejecutarse en segundo plano tan pronto como se encienda a través del archivo de configuración. Como resultado, comenzará a abrir canales inmediatamente si tiene salidas en cadena en su billetera lnd. Si quieres

tenga control total sobre las transacciones de bitcoin que realiza y los canales que abre, asegúrese de apagar el piloto automático ***antes*** de cargar su billetera lnd con fondos de bitcoin. Si el piloto automático se activó anteriormente, es posible que deba reiniciar su lnd antes de recargar su billetera con una transacción en cadena o antes de cerrar canales, lo que efectivamente le brinda fondos en cadena nuevamente. Es crucial que establezca valores de configuración clave si desea ejecutar el piloto automático. Eche un vistazo a esta configuración de ejemplo:

```
[lnd-autopilot]
autopilot.active=1
autopilot.maxchannels=40
autopilot.allocation=0.70
autopilot.minchansize=500000
autopilot.maxchansize=5000000
autopilot.heuristic=top_centrality:1.0
```

Este archivo de configuración activaría el piloto automático. Abriría canales siempre que se cumplan las dos condiciones siguientes:

1. Su nodo actualmente tiene menos de 40 canales abiertos.
2. Menos del 70% de sus fondos totales están fuera de la cadena de pago canales

Los números 40 y 0.7 se eligen de manera completamente arbitraria aquí porque no podemos hacer ninguna recomendación que sea válida para todos sobre cuántos canales debe tener abiertos y qué porcentaje de sus fondos debe estar fuera de la cadena. El piloto automático en lnd no tendrá en cuenta las tarifas de la cadena. En otras palabras, no retrasará la apertura de canales a un período de tiempo cuando las tarifas sean bajas. Para reducir las tarifas, puede abrir canales manualmente durante un período de tiempo cuando las tarifas son bajas, por ejemplo, durante el fin de semana. El piloto automático hará recomendaciones de canales siempre que se cumplan las condiciones e inmediatamente intentará abrir un canal utilizando las tarifas vigentes adecuadas. Según el archivo de configuración anterior, los canales tendrán un tamaño de entre 5 mBTC (minchansize = 500 000 satoshi) y 50 mBTC (maxchansize = 5 000 000 satoshi). Como es común, las cantidades

en el archivo de configuración se enumeran en satoshi. Actualmente, los canales por debajo de 1 mBTC no son muy útiles y no recomendamos abrir canales que sean demasiado pequeños y por debajo de esta cantidad. Con la adopción más amplia de pagos de varias partes, los canales más pequeños son una carga menor. Pero por el momento, esta es nuestra recomendación.

El complemento c-lightning, que fue escrito originalmente por René Pickhardt (coautor de este libro), funciona de manera muy diferente en comparación con el piloto automático lnd. Primero, difiere en los algoritmos usados para hacer las recomendaciones. No cubriremos esto aquí. En segundo lugar, difiere en su interfaz de usuario. Deberá descargar el ***complemento de piloto automático*** del repositorio de complementos de c lightning y activarlo.

NOTA

Para activar un complemento en c-lightning, colóquelo en el directorio ***~/.lightning/plugins ,*** asegúrese de que sea ejecutable (p. ej., chmod +x ~/.lightning/plugins/autopilot.py), luego reinicie lightningd.

Alternativamente, si no desea que un complemento se active automáticamente cuando inicie lightningd, puede colocarlo en un directorio diferente y activarlo manualmente con el argumento del complemento para lightningd:

```
relámpago --plugin=~/relámpago-plugins/autopilot.py
```

El piloto automático en c-lightning se controla a través de tres valores de configuración que se pueden establecer en el archivo de configuración o como argumentos de la línea de comandos cuando inicia lightningd:

```
[c-lightning-autopilot] autopilot-
percent=75 autopilot-num-
channels=10 autopilot-min-channel-
size-msat=100000000msat
```

Estos valores son la configuración predeterminada real y no es necesario establecerlos en absoluto.

El piloto automático no se ejecutará automáticamente en segundo plano como en lnd. En su lugar, debe iniciar una ejecución específicamente con lightning-cli autopilot-run-once si desea que el piloto automático abra los canales recomendados. Pero si desea que solo le brinde recomendaciones, de las cuales puede seleccionar manualmente los nodos, puede agregar el argumento de ejecución en seco opcional.

Una diferencia clave entre los pilotos automáticos lnd y c-lightning es que el piloto automático c-lightning también hará una recomendación para el tamaño del canal. Por ejemplo, si el piloto automático recomienda abrir un canal con un nodo pequeño que solo tiene canales pequeños, no recomendará abrir un canal grande. Sin embargo, si abre un canal con un nodo bien conectado que también tiene muchos canales grandes, probablemente recomendará un tamaño de canal más grande.

Como puede ver, el piloto automático c-lightning no es tan automático como el de lnd, pero le brinda un poco más de control. Estas diferencias reflejan preferencias personales y en realidad podrían ser el factor decisivo para elegir una implementación sobre la otra.

Tenga en cuenta que los pilotos automáticos actuales utilizarán principalmente información pública del protocolo de chismes sobre la topología actual de Lightning Network. Es obvio que sus requisitos personales para los canales solo pueden reflejarse hasta cierto punto. Los pilotos automáticos más avanzados usarían información histórica y de uso que su nodo ha recopilado cuando se ejecutó en el pasado, incluida información sobre los éxitos de enrutamiento, a quién le pagó en el pasado y quién le pagó. En el futuro, estos pilotos automáticos mejorados también podrían usar estos datos recopilados para hacer recomendaciones sobre el cierre de canales y la reasignación de fondos.

En general, al momento de escribir este libro, tenga cuidado de no depender o depender demasiado de los pilotos automáticos.

Obtener liquidez entrante

En el diseño actual de Lightning Network, es más común que los usuarios obtengan liquidez de salida ***antes*** de obtener liquidez de entrada. Ellos harán

entonces, al abrir un canal con otro nodo, y más a menudo podrán gastar bitcoin antes de poder recibirlo. Hay tres formas típicas de obtener liquidez entrante:

- Abra un canal con liquidez saliente y luego gaste algunos de esos fondos. Ahora el saldo está en el otro extremo del canal, lo que significa que puede recibir pagos.

- Pídale a alguien que abra un canal a su nodo. Ofrezca corresponder, para que ambos nodos estén mejor conectados y equilibrados.
- Use un intercambio submarino (por ejemplo, Loop In) para intercambiar BTC en cadena por un canal de entrada a su nodo.
- Pague un servicio de terceros para abrir un canal con usted. Existen varios servicios de este tipo. Algunos cobran una tarifa para proporcionar liquidez, algunos son gratuitos.

Aquí hay una lista de proveedores de liquidez actualmente disponibles que abrirán un canal a su nodo por una tarifa:

- El servicio Thor de Bitrefill
- relámpago para mí
- LNGrande
- Pararrayos

Crear liquidez entrante es un desafío tanto desde la perspectiva práctica como de la experiencia del usuario. La liquidez entrante no ocurre automáticamente, por lo que debe encontrar formas de crearla para su nodo. Esta asimetría de los canales de pago tampoco es intuitiva. En la mayoría de los demás sistemas de pago, se le paga primero (entrada) antes de pagar a otros (salida).

El desafío de crear liquidez entrante es más evidente si es comerciante o vende sus servicios para pagos Lightning. En ese caso, debe estar atento para asegurarse de tener suficiente liquidez entrante para poder continuar recibiendo pagos. ¿Qué pasa si hay una oleada de compradores en

su tienda, pero en realidad no pueden pagarle porque no hay más capacidad de entrada?

En el futuro, estos desafíos pueden mitigarse parcialmente mediante la implementación de canales de doble financiación, que se financian desde ambos lados y ofrecen una capacidad de entrada y salida equilibrada. La carga también podría mitigarse con un software de piloto automático más sofisticado, que podría solicitar y pagar la capacidad de entrada según sea necesario.

En última instancia, los usuarios de Lightning deben ser estratégicos y proactivos con respecto a la administración de canales para garantizar que haya suficiente capacidad de entrada disponible para satisfacer sus necesidades.

Canales de cierre

Como se discutió anteriormente en el libro, un ***cierre mutuo*** es la forma preferida de cerrar un canal. Sin embargo, hay casos en los que un ***cierre forzado*** es necesario.

Algunos ejemplos:

- Su socio de canal está desconectado y no se puede contactar para iniciar un cierre mutuo.
- Su socio de canal está en línea, pero no responde a las solicitudes para iniciar un cierre mutuo.
- Su socio de canal está en línea y sus nodos están negociando un cierre mutuo, pero se atascan y no pueden llegar a una resolución.

Canales de reequilibrio

En el curso de transacciones y enrutamiento de pagos en Lightning, la combinación de capacidades de entrada y salida puede desequilibrarse.

Por ejemplo, si uno de sus socios de canal enruta pagos con frecuencia a través de su nodo, agotará la capacidad de entrada en ese canal, al mismo tiempo que agota la capacidad de salida en los canales de salida. Una vez que eso suceda, ya no podrá enrutar los pagos a través de esa ruta.

Hay muchas formas de reequilibrar los canales, cada una con diferentes ventajas y desventajas. Una forma es usar un intercambio submarino (por ejemplo, Loop Out), como se describió anteriormente en este capítulo. Otra forma de reequilibrar es simplemente esperar a que los pagos enrutados fluyan en la dirección opuesta. Si su nodo está bien conectado, cuando una ruta específica se agota en una dirección, la misma ruta vuelve a estar disponible en la dirección opuesta. Otros nodos pueden "descubrir" esa ruta en la dirección opuesta y comenzar a usarla como parte de su ruta de pago, reequilibrando así los fondos nuevamente.

Una tercera forma de reequilibrar los canales es crear a propósito una ***ruta circular*** que envíe un pago desde su nodo de regreso a su nodo, a través de Lightning Network. Al enviar un pago en un canal con gran capacidad local y organizar la ruta para que regrese a su nodo en un canal con gran capacidad remota, ambos canales estarán más equilibrados. En la Figura 5-4 se puede ver un ejemplo de una estrategia de reequilibrio de ruta circular .

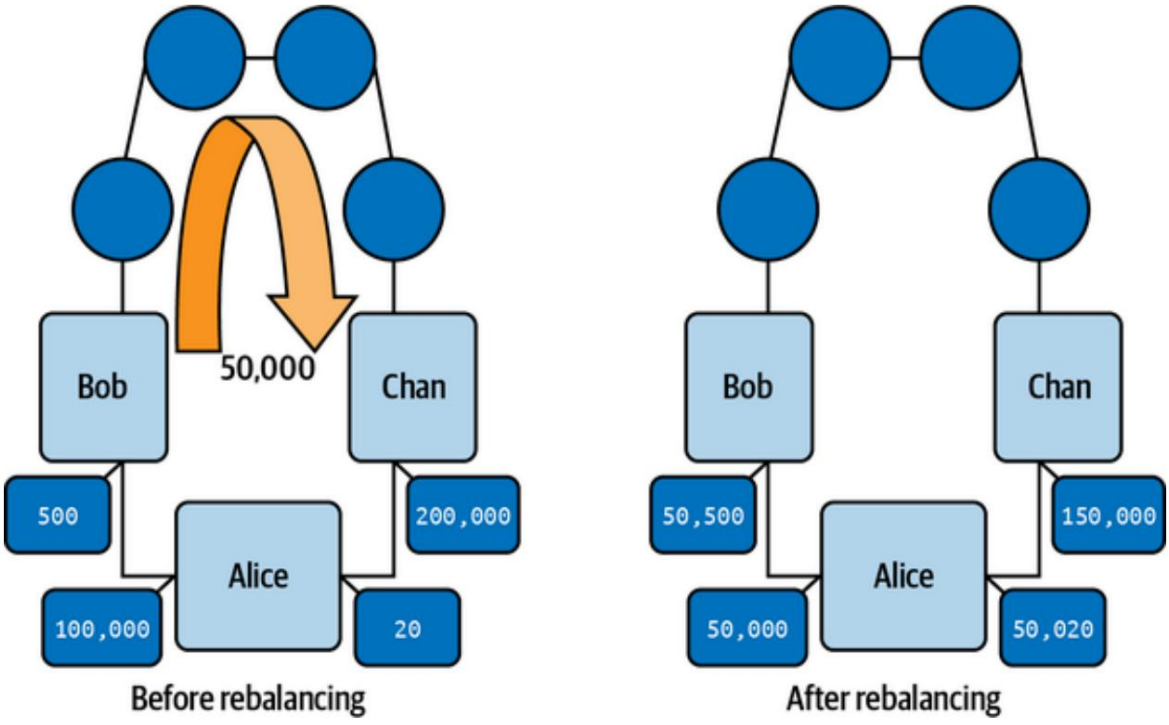


Figura 5-4. Reequilibrio de rutas circulares

El reequilibrio circular es compatible con la mayoría de las implementaciones de nodos Lightning y se puede realizar en la línea de comandos o a través de una de las interfaces de administración web, como Ride The Lightning (consulte “Ride The Lightning”).

El reequilibrio de canales es un tema complejo que es objeto de investigación activa y se cubre con más detalle en "Reequilibrio de canales".

Tarifas de enrutamiento

Ejecutar un nodo Lightning le permite ganar tarifas al enrutar los pagos a través de sus canales. Las tarifas de enrutamiento generalmente no son una fuente importante de ingresos y se ven empequeñecidas por el costo de operar un nodo. Por ejemplo, en un nodo relativamente ocupado que enruta una docena de pagos al día, las tarifas ascienden a no más de 2000 satoshis.

Los nodos compiten por las tarifas de enrutamiento al establecer su tasa de tarifa deseada en cada canal. Las tarifas de enrutamiento se establecen mediante dos parámetros en cada canal: una ***tarifa base fija*** que se cobra por cualquier pago y una ***tarifa de tarifa*** variable adicional que es proporcional al monto del pago.

Al enviar un pago Lightning, un nodo seleccionará una ruta para minimizar las tarifas, minimizar los saltos o ambos. Como resultado, surge un mercado de tarifas de enrutamiento a partir de estas interacciones. Actualmente, hay muchos nodos que cobran tarifas muy bajas o ninguna tarifa por el enrutamiento, lo que crea una presión a la baja en el mercado de tarifas de enrutamiento.

Si no elige ninguna opción, su nodo Lightning establecerá una tarifa base predeterminada y una tarifa de tarifa para cada nuevo canal. Los valores predeterminados dependen de la implementación del nodo que utilice. La tarifa base se establece en la unidad de ***millisatoshi*** (milésimas de un satoshi). El tipo de comisión proporcional se fija en la unidad de ***millonésimas*** y se aplica sobre el importe del pago. La unidad de millonésimas a menudo se abrevia con ***ppm*** (partes por millón). Por ejemplo, una tarifa base de 1000 (millisatoshi) y una tasa de tarifa de 1000 ppm (millonésimas) generaría los siguientes cargos por un pago de 100 000 satoshi:

P = 100, 000 satoshi
Fbase = 1, 000 millisatoshi = 1 satoshi
Frate = 1, 000 ppm = 1, 000/1, 000, 000 = 1/1, 000 = 0.001 = 0. 1%
Ftotal = Fbase + (P *Frecuencia)
ÿ Ftotal = 1 satoshi + (100, 000/1, 000) satoshi ÿ
Ftotal = 1 satoshi + 100 satoshi = 101 satoshi

En términos generales, puede adoptar uno de dos enfoques para las tarifas de enrutamiento. Puede enrutar muchos pagos con tarifas bajas, compensando las tarifas bajas con un volumen alto. Alternativamente, puede optar por cobrar tarifas más altas. Si elige establecer tarifas más altas, su nodo se seleccionará solo cuando no existan otras rutas más baratas. Por lo tanto, enrutará con menos frecuencia pero ganará más por enrutamiento exitoso.

Para la mayoría de los nodos, generalmente es mejor usar los valores de tarifa de enrutamiento predeterminados. De esta forma, su nodo compite en igualdad de condiciones con otros nodos que utilizan los valores predeterminados.

También puede usar la configuración de la tarifa de enrutamiento para reequilibrar los canales. Si la mayoría de sus canales tienen las tarifas predeterminadas pero desea reequilibrar un canal en particular, simplemente disminuya las tarifas en ese canal específico a cero o a tarifas muy bajas. Luego siéntese y espere a que alguien envíe un pago a través de su ruta "barata" y reequilibre sus canales como efecto secundario.

Gestión de nodos

Obviamente, administrar su nodo Lightning en la línea de comandos no es fácil. Le brinda la flexibilidad total de la API del nodo y la capacidad de escribir sus propios scripts personalizados para satisfacer sus requisitos personales. Pero si no desea lidiar con la complejidad de la línea de comandos y solo necesita algunas capacidades básicas de administración de nodos, debe considerar instalar una interfaz de usuario basada en web que facilite la administración de nodos.

Hay una serie de proyectos en competencia que ofrecen administración de nodos Lightning basada en la web. Algunos de los más populares se describen en la siguiente sección.

montar el relámpago

Ride The Lightning (RTL) es una interfaz gráfica de usuario web para ayudar a los usuarios a administrar las operaciones del nodo Lightning para las tres implementaciones principales del nodo Lightning (LND, c-lightning y Eclair). RTL es un proyecto de código abierto desarrollado por Shahana Farooqui y muchos otros colaboradores.

Puede encontrar el software RTL en GitHub.

La Figura 5-5 muestra una captura de pantalla de ejemplo de la interfaz web de RTL, tal como se proporciona en el repositorio del proyecto.

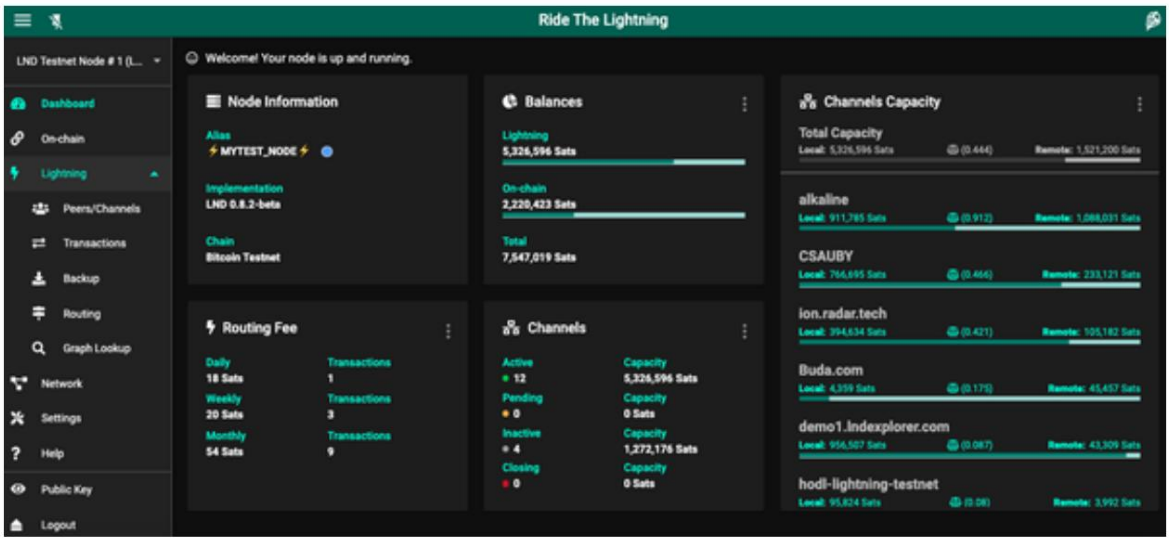


Figura 5-5. Ejemplo de interfaz web RTL

Indmon

Lightning Labs, los creadores de LND, proporcionan una interfaz gráfica de usuario basada en la web llamada lndmon para monitorear las diversas métricas de un LND.

Nodo relámpago. lndmon solo funciona con nodos LND. Es una interfaz de solo lectura para monitorear y, como tal, no le permite administrar activamente el nodo. No puede abrir canales ni realizar pagos. Encuentre lndmon en GitHub.

ThunderHub

ThunderHub es una interfaz gráfica de usuario basada en web muy estéticamente agradable, similar a RTL pero exclusiva de LND. Se puede utilizar para hacer

pagos, reequilibrar canales y administrar el nodo a través de una variedad de funciones.

Conclusión

A medida que mantenga su nodo y adquiera experiencia, aprenderá mucho sobre Lightning Network. Ser un operador de nodo es una tarea desafiante pero gratificante. Dominar estas habilidades le permitirá contribuir al crecimiento y desarrollo de esta tecnología y de Lightning Network. Además, obtendrá la capacidad de enviar y recibir pagos Lightning con el mayor grado de control y facilidad. Desempeñará un papel central en la infraestructura de la red y no solo será un participante marginal.

Parte II. La red Lightning en detalle

Una explicación detallada de todos los componentes de Lightning Network y cómo funcionan. Esta parte es muy técnica y se espera que el lector tenga algo de experiencia en programación e informática.

Capítulo 6. Arquitectura de la red Lightning

En la primera parte de este libro presentamos los conceptos principales de Lightning Network y trabajamos a través de un ejemplo completo de enrutamiento de un pago y configuración de las herramientas que podemos usar para explorar más. En la segunda parte del libro, exploraremos Lightning Network con muchos más detalles técnicos, diseccionando cada uno de los componentes básicos.

En esta sección, describiremos los componentes de Lightning Network con más detalle y brindaremos una perspectiva de "panorama general" para guiarlo a través de los siguientes capítulos.

El conjunto de protocolos de red Lightning

Lightning Network se compone de una colección compleja de protocolos que se ejecutan sobre Internet. En términos generales, podemos clasificar estos protocolos en cinco capas distintas que forman una ***pila de protocolos,*** donde cada capa se basa en los protocolos de la capa inferior y los utiliza. Además, cada capa de protocolo abstrae las capas subyacentes y "oculta" parte de la complejidad.

El diagrama de arquitectura que se muestra en la Figura 6-1 proporciona una descripción general de estas capas y los protocolos de sus componentes.

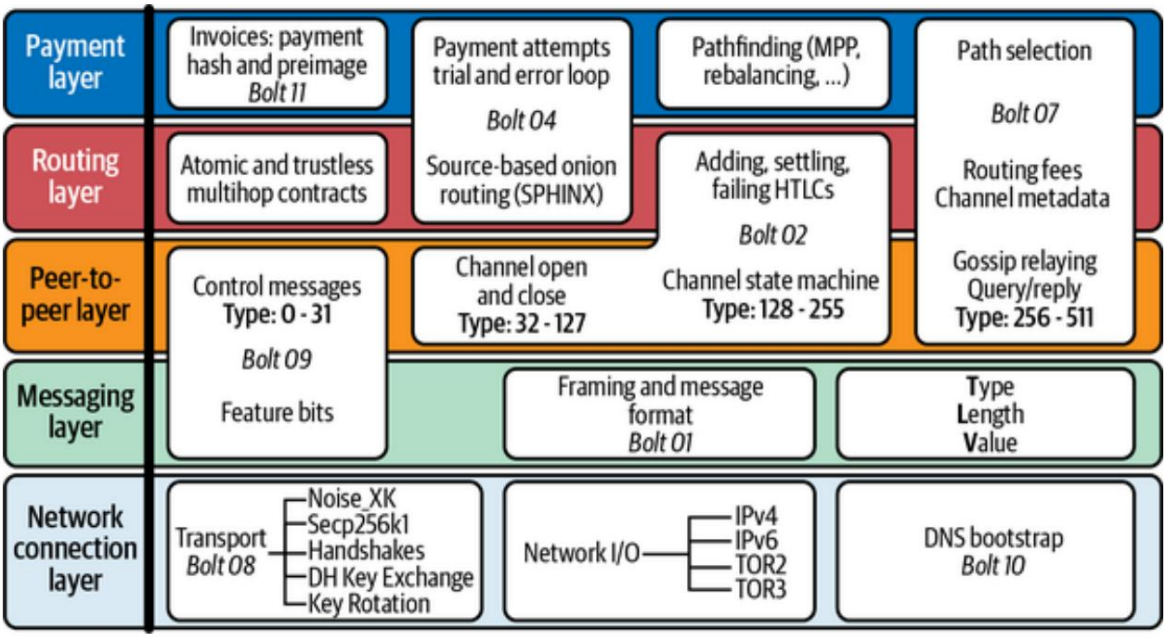


Figura 6-1. El conjunto de protocolos Lightning Network

Las cinco capas de Lightning Network, de abajo hacia arriba, son:

Capa de conexión de red

Contiene los protocolos que interactúan directamente con los protocolos centrales de Internet (TCP/IP), los protocolos superpuestos (Tor v2/v3) y los servicios de Internet (DNS). Esta capa también contiene los protocolos de transporte criptográfico que protegen los mensajes Lightning.

capa de mensajes

Esta capa contiene los protocolos que utilizan los nodos para negociar funciones, formatear mensajes y codificar campos de mensajes.

Capa punto a punto (P2P)

Esta capa es la capa de protocolo principal para la comunicación entre nodos Lightning y contiene todos los diferentes mensajes intercambiados entre nodos.

Capa de enrutamiento

Esta capa contiene los protocolos utilizados para enrutar pagos entre nodos, de extremo a extremo y atómicamente. Esta capa contiene la funcionalidad principal de Lightning Network: pagos enrutados.

Capa de pago

La capa más alta de la red, que presenta una interfaz de pago confiable para las aplicaciones.

Relámpago en detalle

En los próximos 10 capítulos, analizaremos el conjunto de protocolos y examinaremos cada componente de Lightning Network en detalle.

Pasamos bastante tiempo tratando de decidir el mejor orden para presentar este detalle. No es una elección fácil porque hay mucha interdependencia entre los diferentes componentes: cuando empiezas a explicar uno, descubres que atrae a muchos de los otros componentes. En lugar de un enfoque de arriba hacia abajo o de abajo hacia arriba, terminamos eligiendo un camino más serpenteante que comienza con los componentes básicos más fundamentales que son exclusivos de los canales de pago de Lightning Network y se mueve hacia afuera desde allí. Pero dado que esa ruta no es obvia, usaremos Lightning Protocol Suite que se muestra en la Figura 6-1 como mapa. En cada capítulo se centrará en uno o más componentes relacionados, y los verá resaltados en el conjunto de protocolos.
Algo así como un marcador de mapa que dice "¡Estás aquí!"

Esto es lo que cubriremos:

Capítulo 7, “Canales de pago”

En este capítulo veremos cómo funcionan los canales de pago, con mucha más profundidad de lo que vimos en las partes anteriores del libro.
Veremos la estructura y el Bitcoin Script de las transacciones de financiación y compromiso, y el proceso utilizado por los nodos para negociar cada paso en el protocolo.

Capítulo 8, “Enrutamiento en una red de canales de pago”

A continuación, juntaremos varios canales de pago en una red y enrutaremos un pago de un extremo al otro. En ese proceso, nos sumergiremos en el contrato inteligente de hash time-locked contract (HTLC) y el Bitcoin Script que usamos para construirlo.

Capítulo 9, “Operación del canal y reenvío de pagos”

Reuniendo los conceptos de un canal de pago simple y un pago enrutado usando HTLC, ahora veremos cómo los HTLC son parte de la transacción de compromiso de cada canal. También veremos el protocolo para agregar, liquidar, fallar y eliminar HTLC de los compromisos.

Capítulo 10, “Enrutamiento de cebolla”

A continuación, veremos cómo se propaga la información HTLC a través de la red dentro del protocolo de enrutamiento cebolla. Veremos el mecanismo para el cifrado y descifrado en capas que le da a Lightning Network algunas de sus características de privacidad.

Capítulo 11, "Chismes y el gráfico de canales"

En este capítulo, veremos cómo los nodos Lightning se encuentran entre sí y aprenderemos sobre los canales publicados para construir un gráfico de canales que puedan usar para encontrar rutas a través de la red.

Capítulo 12, “Búsqueda de rutas y entrega de pagos”

A continuación, veremos cómo cada nodo utiliza la información del protocolo de chismes para construir un "mapa" de toda la red, que puede usar para encontrar rutas de un punto a otro para enrutar los pagos. También veremos las innovaciones existentes en la búsqueda de rutas, como los pagos de varias partes.

Capítulo 13, "Protocolo de cable: encuadre y extensibilidad"

La base de Lightning Network es el protocolo peer-to-peer que los nodos utilizan para intercambiar mensajes sobre la red y sus canales. En este capítulo observamos cómo se construyen esos mensajes.

y las capacidades de extensión integradas en los mensajes con bits de características y codificación Tipo-Longitud-Valor (TLV).

Capítulo 14, “Transporte de mensajes cifrados de Lightning”

Pasando a la parte de nivel inferior de la red, veremos el sistema de transporte cifrado subyacente que garantiza el secreto y la integridad de todas las comunicaciones entre los nodos.

Capítulo 15, “Solicitudes de pago relámpago”

Una parte clave de Lightning Network son las solicitudes de pago, también conocidas como facturas Lightning. En este capítulo analizamos la estructura y la codificación de una factura.

¡Vamos a sumergirnos!

Capítulo 7. Canales de Pago

En este capítulo nos sumergiremos en los canales de pago y veremos cómo se construyen. Comenzaremos con el nodo de Alice abriendo un canal hacia el nodo de Bob, construyendo sobre los ejemplos presentados al principio de este libro.

Los mensajes intercambiados por los nodos de Alice y Bob se definen en "BOLT #2: Peer Protocol for Channel Management". Las transacciones creadas por los nodos de Alice y Bob se definen en "BOLT #3: Formatos de secuencias de comandos y transacciones de Bitcoin". En este capítulo, nos centramos en las partes "Channel open and close" y "Channel state machine" de la arquitectura del protocolo Lightning, resaltadas por un contorno en el centro (capa de igual a igual) de la Figura 7-1.

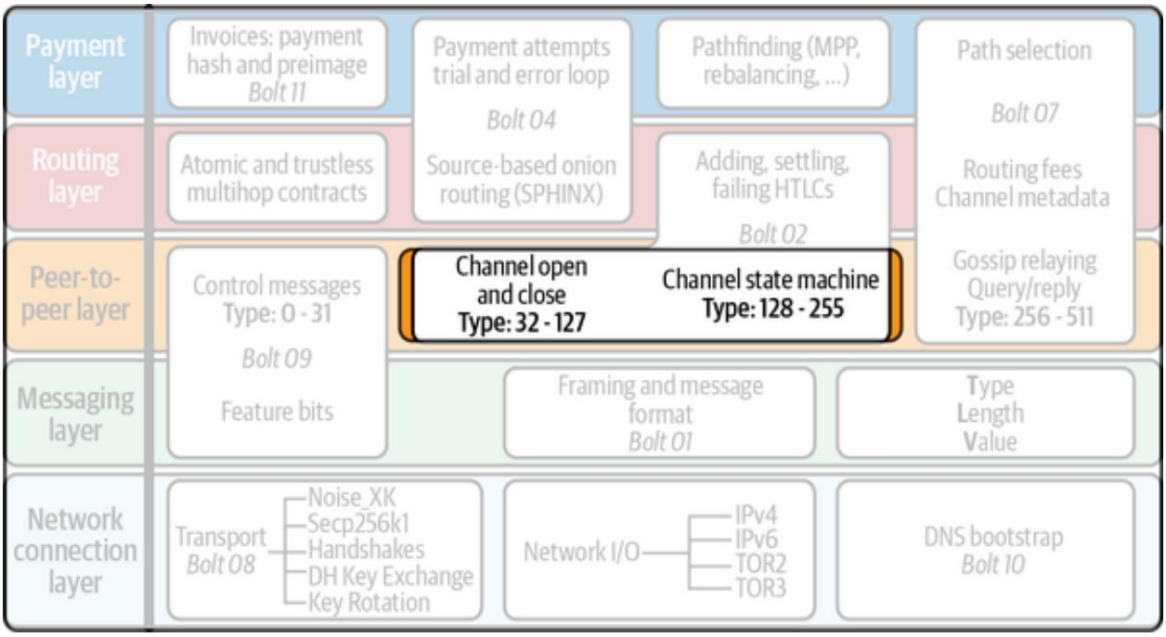


Figura 7-1. Canales de pago en el conjunto de protocolos Lightning

Una forma diferente de usar el sistema Bitcoin

Lightning Network a menudo se describe como un "Protocolo Bitcoin de capa 2", lo que hace que suene distinto de Bitcoin. Otra forma de describir

Lightning es como una "forma más inteligente de usar Bitcoin" o simplemente como una "aplicación sobre Bitcoin". Exploremos eso.

Históricamente, las transacciones de Bitcoin se transmiten a todos y se registran en la cadena de bloques de Bitcoin para que se consideren válidas. Sin embargo, como veremos, si alguien tiene una transacción de Bitcoin prefirmada que gasta una salida multisig de 2 de 2 que le da la capacidad exclusiva de gastar ese Bitcoin, efectivamente posee ese Bitcoin incluso si no transmite la transacción.

Puede pensar en la transacción de Bitcoin prefirmada como un cheque posfechado (o cheque), que se puede cobrar en cualquier momento. Sin embargo, a diferencia del sistema bancario tradicional, esta transacción no es una “promesa” de pago (también conocida como pagaré), sino un instrumento al portador verificable que es equivalente al efectivo. Siempre que el bitcoin al que se hace referencia en la transacción no se haya gastado en el momento del canje (o en el momento en que intente "cobrar" el cheque), el sistema Bitcoin garantiza que esta transacción prefirmada se puede transmitir y registrar en cualquier momento. . Esto solo es cierto, por supuesto, si esta es la única transacción prefirmada. Dentro de Lightning Network existen dos o más de tales transacciones prefirmadas al mismo tiempo; por lo tanto, necesitamos un mecanismo más sofisticado para seguir teniendo la funcionalidad de un instrumento al portador verificable, como también aprenderá en este capítulo.

Lightning Network es simplemente una forma diferente y creativa de usar Bitcoin. En Lightning Network, una combinación de transacciones registradas (en la cadena) y prefirmadas pero retenidas (fuera de la cadena) forman una "capa" de pagos que es una forma más rápida, económica y privada de usar Bitcoin. Puede ver esta relación entre las transacciones de Bitcoin dentro y fuera de la cadena en la Figura 7-2.

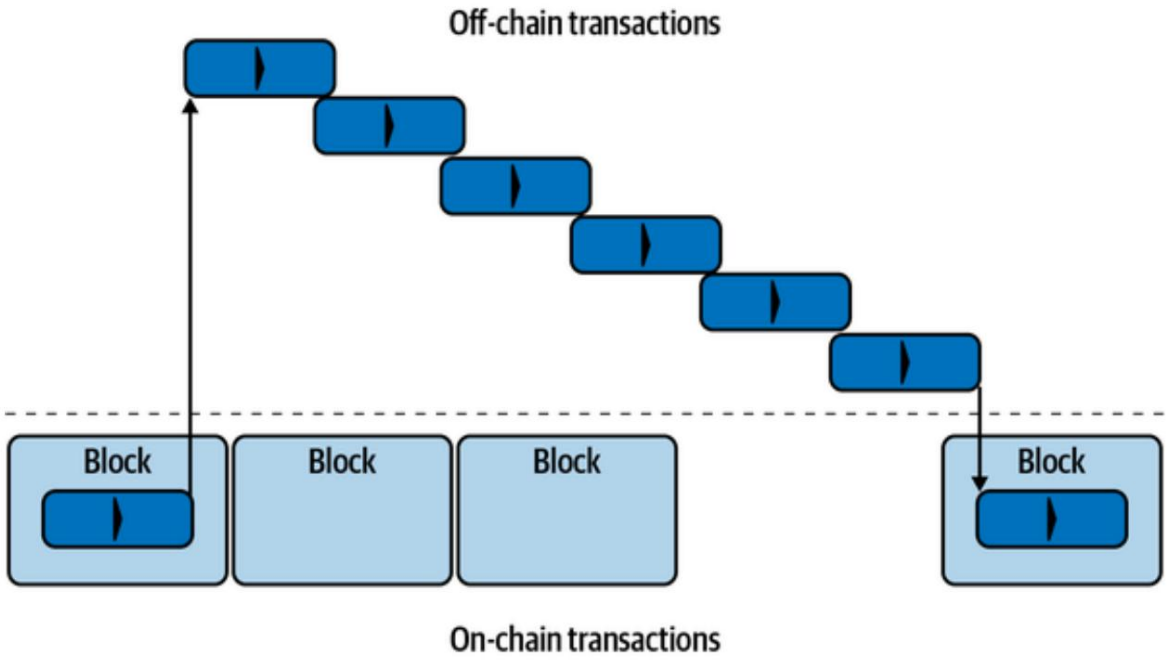


Figura 7-2. Canal de pago relámpago hecho de transacciones dentro y fuera de la cadena

El rayo es Bitcoin. Es solo una forma diferente de usar el sistema Bitcoin.

Propiedad y control de Bitcoin

Antes de comprender los canales de pago, debemos dar un pequeño paso atrás y comprender cómo funcionan la propiedad y el control en Bitcoin.

Cuando alguien dice que "posee" bitcoin, por lo general quiere decir que conoce la clave privada de una dirección de Bitcoin que tiene algunos resultados de transacciones no utilizados (consulte el Apéndice A). La clave privada les permite firmar una transacción para gastar ese bitcoin transfiriéndolo a una dirección diferente. En Bitcoin, la "propiedad" de bitcoin se puede definir como la ***capacidad de gastar*** ese bitcoin.

Tenga en cuenta que el término "propiedad" como se usa en Bitcoin es distinto del término "propiedad" usado en un sentido legal. Un ladrón que tiene las claves privadas y puede gastar Bitcoin es un ***propietario de facto*** de ese Bitcoin aunque no sea un propietario legal.

PROPINA

La propiedad de Bitcoin se trata solo del control de las claves y la capacidad de gastar Bitcoin con esas claves. Como dice el dicho popular de Bitcoin: "Tus llaves, tus monedas, no tus llaves, no tus monedas".

Diversidad de propiedad (independiente) y multigrado

La propiedad y el control de las claves privadas no siempre está en manos de una sola persona. Ahí es donde las cosas se ponen interesantes y complicadas. Sabemos que más de una persona puede llegar a conocer la misma clave privada, ya sea por robo o porque el poseedor original de la clave hace una copia y se la entrega a otra persona. ¿Todas estas personas son propietarios? En un sentido práctico, lo son, porque cualquiera de las personas que conocen la clave privada puede gastar el bitcoin sin la aprobación de nadie más.

Bitcoin también tiene direcciones de firmas múltiples donde se necesitan varias claves privadas para firmar antes de gastar (ver "Scripts de firmas múltiples"). Desde una perspectiva práctica, la propiedad en una dirección de múltiples firmas depende del quórum ***(K)*** y el total ***(N)*** definidos en el esquema K-of-N . Un esquema de firma múltiple 1 de 10 permitiría que 1 ***(K)*** de 10 ***(N)*** firmantes gasten una cantidad de bitcoin bloqueada en esa dirección. Esto es similar al escenario en el que 10 personas tienen una copia de la misma clave privada y cualquiera de ellas puede gastarla de forma independiente.

Propiedad conjunta sin control independiente

También está el escenario donde ***nadie*** tiene quórum. En un esquema 2 de 2 como el que se usa en Lightning Network, ninguno de los firmantes puede gastar el bitcoin sin obtener una firma de la otra parte. ¿Quién posee el bitcoin en ese caso? Nadie tiene realmente la propiedad porque nadie tiene el control. Cada uno posee el equivalente a una participación con derecho a voto en la decisión, pero se necesitan ambos votos. Un problema clave (juego de palabras) con un esquema 2 de 2, tanto en Bitcoin como en la ley, es lo que sucede si una de las partes no está disponible, o si hay un punto muerto en la votación y una de las partes se niega a cooperar.

Prevención de Bitcoin "Bloqueado" y no gastable

Si uno de los dos firmantes de un multigrado 2 de 2 no puede o no quiere firmar, los fondos se vuelven inutilizables. Este escenario no solo puede ocurrir accidentalmente (pérdida de llaves), sino que puede ser utilizado como una forma de chantaje por cualquiera de las partes: "No firmaré a menos que me pagues una parte de los fondos".

Los canales de pago en Lightning se basan en una dirección multigrado 2 de 2, con los dos socios de canal como firmantes en el multigrado. En este momento, los canales son financiados solo por uno de los dos socios de canal: cuando elige "abrir" un canal, deposita fondos en la dirección multigrado 2 de 2 con una transacción. Una vez que se extrae esa transacción y los fondos están en multisig, no puede recuperarlos sin la cooperación de su socio de canal, porque necesita su firma (también) para gastar el bitcoin.

En la siguiente sección, mientras analizamos cómo abrir (crear) un canal Lightning, veremos cómo podemos evitar la pérdida de fondos o cualquier escenario de chantaje entre los dos socios mediante la implementación de un protocolo justo para la construcción del canal con la ayuda de transacciones prefirmadas que gastan la salida multisig de una manera que les da a los pares en el canal la capacidad exclusiva de gastar una de las salidas que codifica la cantidad de bitcoin que poseen en el canal.

Construcción de un canal de pago

En "¿Qué es un canal de pago?", describimos los canales de pago como una ***relación financiera*** entre dos nodos Lightning, que se establece financiando una dirección de firma múltiple 2 de 2 de los dos socios de canal.

Supongamos que Alice quiere construir un canal de pago que le permita conectarse directamente a la tienda de Bob. Primero, los dos nodos (Alice y Bob) deben establecer una conexión a Internet entre sí, para que puedan negociar un canal de pago.

Claves privadas y públicas del nodo

Cada nodo en Lightning Network se identifica mediante una ***clave pública de nodo.*** La clave pública identifica de forma única el nodo específico y generalmente se presenta como una codificación hexadecimal. Por ejemplo, René Pickhardt actualmente ejecuta un Lightning Node (ln.rene-pickhardt.de) que se identifica mediante la siguiente clave pública de nodo:

```
02a1cebfacb2674143b5ad0df3c22c609e935f7bc0ebe801f37b8e9023d45ea7b
8
```

Cada nodo genera una clave privada raíz cuando se inicializa por primera vez. La clave privada se mantiene privada en todo momento (nunca se comparte) y se almacena de forma segura en la billetera del nodo. A partir de esa clave privada, el nodo deriva una clave pública que es el identificador del nodo y se comparte con la red. Dado que el espacio de claves es enorme, siempre que cada nodo genere la clave privada aleatoriamente, tendrá una clave pública única, que por lo tanto puede identificarlo de manera única en la red.

Dirección de red del nodo

Además, cada nodo también anuncia una dirección de red a la que se puede acceder, en uno de varios formatos posibles:

TCP/IP

Una dirección IPv4 o IPv6 y un número de puerto TCP

TCP/Tor

Una dirección "cebolla" de Tor y un número de puerto TCP

El identificador de la dirección de red se escribe como Dirección:Puerto, lo cual es consistente con los estándares internacionales para identificadores de red, como se usa, por ejemplo, en la web.

Por ejemplo, el nodo de René con clave pública de nodo 02a1ceb…45ea7b8 actualmente anuncia su dirección de red como la dirección TCP/IP:

```
172.16.235.20:9735
```

PROPINA

El puerto TCP predeterminado para Lightning Network es 9735, pero un nodo puede elegir escuchar en cualquier puerto TCP.

Identificadores de nodos

Juntas, la clave pública del nodo y la dirección de red se escriben en el siguiente formato, separadas por un signo @, como *NodeID@Address:Port.*

Entonces el identificador completo para el nodo de René sería:

```
02a1cebfacb2674143b5ad0df3c22c609e935f7bc0ebe801f37b8e9023d45ea7b
8
@172.16.235.20:9735
```

PROPINA

El alias del nodo de René es ln.rene-pickhardt.de; sin embargo, este nombre existe solo para una mejor legibilidad. Cada operador de nodo puede anunciar el alias que desee y no existe ningún mecanismo que impida que los operadores de nodo seleccionen un alias que ya se está utilizando. Por lo tanto, para hacer referencia a un nodo, se debe utilizar el *esquema IDNodo@Dirección:Puerto .*

El identificador anterior a menudo está codificado en un código QR, lo que facilita que los usuarios lo escaneen si desean conectar su propio nodo al nodo específico identificado por esa dirección.

Al igual que los nodos de Bitcoin, los nodos Lightning anuncian su presencia en Lightning Network "cotilleando" la clave pública y la dirección de red de su nodo. De esa manera, otros nodos pueden encontrarlos y mantener un inventario (base de datos) de todos los nodos conocidos a los que pueden conectarse e intercambiar los mensajes que se definen en el protocolo de mensajes Lightning P2P.

Conexión de nodos como pares directos

Para que el nodo de Alice se conecte al nodo de Bob, necesitará la clave pública del nodo de Bob o la dirección completa que contiene la clave pública, la dirección IP o Tor y el puerto. Debido a que Bob administra una tienda, la dirección de nodo de Bob se puede recuperar de una factura o una página de pago de la tienda en la web. Alice puede escanear un código QR que contiene la dirección e indicarle a su nodo que se conecte al nodo de Bob.

Una vez que Alice se ha conectado al nodo de Bob, sus nodos ahora son pares conectados directamente.

PROPINA

Para abrir un canal de pago, primero se deben conectar dos nodos como pares directos abriendo una conexión a través de Internet (o Tor).

Construyendo el Canal

Ahora que los nodos Lightning de Alice y Bob están conectados, pueden comenzar el proceso de construcción de un canal de pago. En esta sección, revisaremos las comunicaciones entre sus nodos, conocidas como ***Lightning Peer Protocol for Channel Management,*** y el protocolo criptográfico que utilizan para generar transacciones de Bitcoin.

PROPINA

Describimos dos protocolos diferentes en este escenario. En primer lugar, existe un ***protocolo de mensajes,*** que establece cómo se comunican los nodos Lightning a través de Internet y qué mensajes intercambian entre sí. En segundo lugar, está el ***protocolo criptográfico,*** que establece cómo los dos nodos construyen y firman las transacciones de Bitcoin.

Protocolo de pares para la gestión de canales

El Lightning Peer Protocol para Channel Management se define en BOLT #2: Peer Protocol for Channel Management. En este capítulo estaremos

revisando las secciones "Establecimiento del canal" y "Cierre del canal" del BOLT #2 con más detalle.

Flujo de mensajes de establecimiento de canal

El establecimiento del canal se logra mediante el intercambio de seis mensajes entre los nodos de Alice y Bob (tres de cada par): abrir_canal, aceptar_canal, financiación_creada, financiación_firmada, financiación_bloqueada y financiación_bloqueada. Los seis mensajes se muestran como un diagrama de secuencia de tiempo en la Figura 7-3.

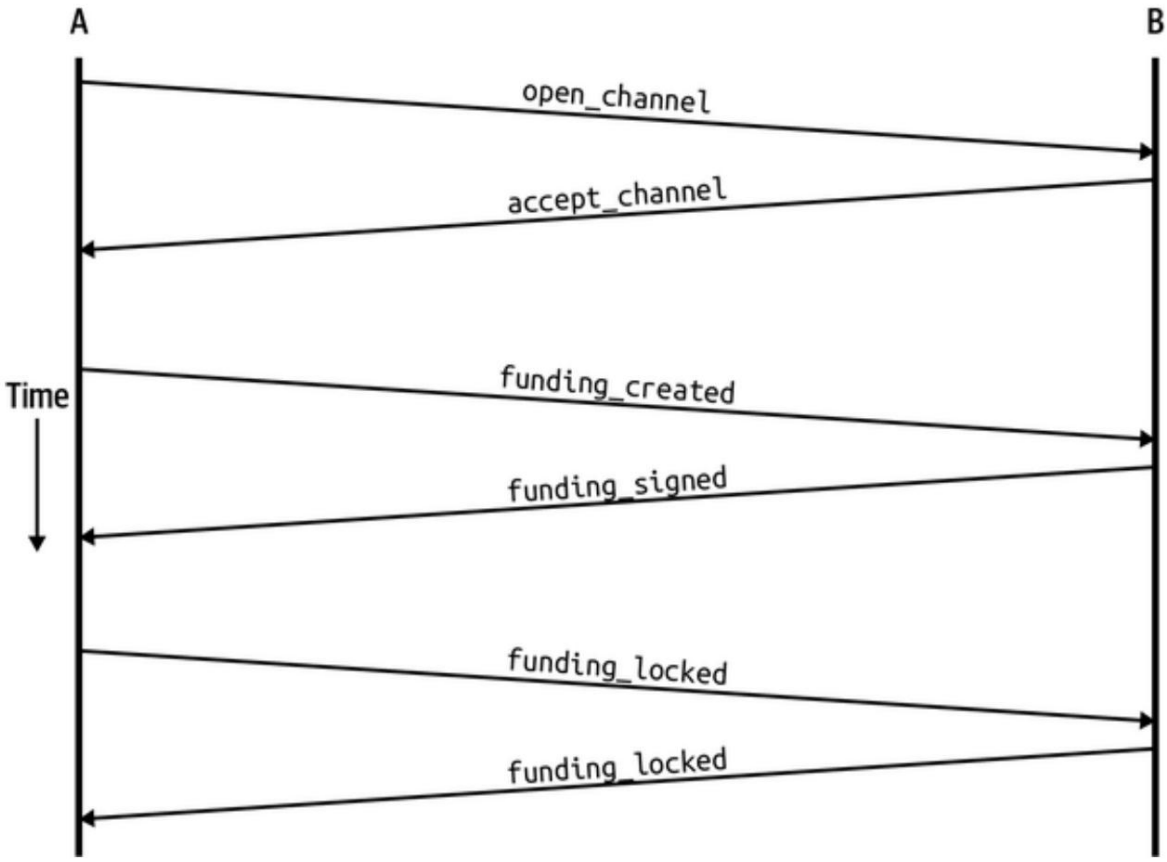


Figura 7-3. El flujo de mensajes de establecimiento de canal

En la Figura 7-3, los nodos de Alice y Bob están representados por las líneas verticales “A” y “B” a ambos lados del diagrama. Un diagrama de secuencia de tiempo como este muestra el tiempo fluyendo hacia abajo y los mensajes fluyendo de un lado al otro entre los dos compañeros de comunicación. Las líneas están inclinadas hacia abajo.

para representar el tiempo transcurrido necesario para transmitir cada mensaje, y la dirección del mensaje se muestra con una flecha al final de cada línea.

El establecimiento del canal involucra tres partes. Primero, los dos pares comunican sus capacidades y expectativas, con Alice iniciando una solicitud a través de open_channel y Bob aceptando la solicitud de canal a través de accept_channel.

En segundo lugar, Alice construye las transacciones de financiación y reembolso (como veremos más adelante en esta sección) y envía la financiación_creada a Bob. Otro nombre para la transacción de "reembolso" es una transacción de "compromiso", ya que se compromete con la distribución actual de saldos en el canal. Bob responde devolviendo las firmas necesarias con financiado_firmado. Esta interacción es la base del ***protocolo criptográfico*** para asegurar el canal y evitar robos. Alice ahora transmitirá la transacción de financiación (en cadena) para establecer y anclar el canal de pago. La transacción deberá confirmarse en la cadena de bloques de Bitcoin.

PROPINA

El nombre del mensaje financiado_firmado puede ser un poco confuso. Este mensaje no contiene una firma para la transacción de financiación, sino la firma de Bob para la transacción de reembolso que le permite a Alice reclamar su bitcoin de la multigrado.

Una vez que la transacción tiene suficientes confirmaciones (según lo definido por el campo de profundidad mínima en el mensaje accept_channel), Alice y Bob intercambian mensajes de financiación bloqueada y el canal ingresa al modo de funcionamiento normal.

El mensaje de canal_abierto

El nodo de Alice solicita un canal de pago con el nodo de Bob enviando un mensaje de canal abierto. El mensaje contiene información sobre las ***expectativas*** de Alice para la configuración del canal, que Bob puede aceptar o rechazar.

La estructura del mensaje open_channel (tomado del BOLT #2) se muestra en el Ejemplo 7-1.

***Ejemplo 7-1. El mensaje** de canal_abierto*

```
[chain_hash:chain_hash]
[32*byte:temporal_channel_id]
[u64:funding_satoshis] [u64:push_msat]
[u64:dust_limit_satoshis]
[u64:max_htlc_value_in_flight_msat]
[u64:channel_reserve_satoshis]
[u64:htlc_minimum_msat]
[u32:feerate_uper6:weerate_ to_self_delay]
[u16:max_accepted_htlcs] [punto:funding_pubkey]
[punto:punto_base_revocación]
[punto:punto_base_pago]
[punto:punto_base_pago_retrasado]
[punto:punto_base_htlc]
[punto:primer_por_punto_compromiso]
[byte:banderas_canal] [open_channel_tlvs:tlvs]
```

Los campos contenidos en este mensaje especifican los parámetros del canal que desea Alice, así como varios ajustes de configuración de los nodos de Alice que reflejan las expectativas de seguridad para el funcionamiento del canal.

Algunos de los parámetros de construcción del canal se enumeran aquí:

cadena_hash

Esto identifica qué cadena de bloques (p. ej., la red principal de Bitcoin) se utilizará para este canal. Por lo general, es el hash del bloque de génesis de esa cadena de bloques.

financiación_satoshis

La cantidad que Alice usará para financiar el canal, que es la capacidad total del canal.

channel_reserve_satoshis

El saldo mínimo, en satoshis, que se reserva a cada lado de un canal. Volveremos sobre esto cuando hablemos de sanciones.

empujar_msat

Una cantidad opcional que Alice "empujará" inmediatamente a Bob como pago al canalizar la financiación. ***Establecer este valor en cualquier valor que no sea 0 significa regalar dinero de manera efectiva a su socio de canal y debe usarse con precaución.***

to_self_delay

Un parámetro de seguridad muy importante para el protocolo. El valor del mensaje open_channel se utiliza en la transacción de compromiso del respondedor y accept_channel en la del iniciador. Esta asimetría existe para permitir que cada parte exprese cuánto tiempo debe esperar la otra parte para reclamar unilateralmente los fondos en una transacción de compromiso. Si Bob en algún momento cierra unilateralmente el canal en contra de la voluntad de Alice, se compromete a no acceder a sus propios fondos durante el retraso definido aquí. Cuanto más alto sea este valor, más seguridad tiene Alice, pero más tiempo tendrá Bob sus fondos bloqueados.

financiación_pubkey

La clave pública que Alice contribuirá al multisig 2 de 2 que ancla este canal.

X_punto base

Claves maestras, utilizadas para derivar claves secundarias para varias partes del compromiso, la revocación, el pago enrutado (HTLC) y las transacciones de cierre. Estos se utilizarán y explicarán en capítulos posteriores.

PROPINA

Si desea comprender los otros campos y los mensajes del protocolo Lightning peer que no analizamos en este libro, le sugerimos que los busque en las especificaciones de BOLT.
Estos mensajes y campos son importantes, pero no pueden cubrirse con suficiente detalle en el alcance de este libro. Queremos que comprenda los principios fundamentales lo suficientemente bien como para que pueda completar los detalles leyendo la especificación del protocolo real (BOLT).

El mensaje accept_channel En

respuesta al mensaje open_channel de Alice, Bob devuelve el mensaje accept_channel que se muestra en el Ejemplo 7-2.

***Ejemplo 7-2. El mensaje** accept_channel*

```
[32*byte:id_canal_temporal] [u64:dust_limit_satoshis]
[u64:max_htlc_value_in_flight_msat]
[u64:channel_reserve_satoshis] [u64:htlc_minimum_msat]
[u32:mínimo_profundidad] [u16:to_self_delay]
[u16:max_accepted_htlcs] [point:funding_pubkey:
punto_base_revocación] [punto:punto_base_pago]
[punto:punto_base_pago_retrasado] [punto:punto_base_htlc]
[punto:primer_por_punto_compromiso] [aceptar_canal_tlvs:tlvs]
```

Como puede ver, esto es similar al mensaje open_channel y contiene las expectativas y los valores de configuración del nodo de Bob.

Los dos campos más importantes en accept_channel que Alice usará para construir el canal de pago son:

financiación_pubkey

El nodo de Bob de la clave pública contribuye a la dirección multisig 2 de 2 que ancla el canal.

profundidad_mínima

El número de confirmaciones que espera el nodo de Bob para la transacción de financiación antes de considerar el canal "abierto" y listo para usar.

La transacción de financiación Una vez

que el nodo de Alice recibe el mensaje accept_channel de Bob, tiene la información necesaria para construir la ***transacción de financiación*** que ancla el canal a la cadena de bloques de Bitcoin. Como discutimos en capítulos anteriores, un canal de pago Lightning está anclado por una dirección de firma múltiple 2 de 2.
Primero, necesitamos generar esa dirección de firma múltiple para permitirnos construir la transacción de financiación (y la transacción de reembolso como se describe posteriormente).

Generación de una dirección de firma múltiple

La transacción de financiación envía cierta cantidad de bitcoin (funding_satoshis del mensaje de canal abierto) a una salida de firma múltiple 2 de 2 que se construye a partir de las claves públicas de financiación_pubkey de Alice y Bob.

El nodo de Alice construye un script de múltiples firmas como se muestra aquí:

2 <*Alice_funding_pubkey*> <*Bob_funding_pubkey*> 2 CHECKMULTISIG

Tenga en cuenta que, en la práctica, las claves de financiación se ***clasifican*** de forma determinista (usando el orden lexicográfico de la forma comprimida serializada de las claves públicas) antes de colocarse en el script testigo. Al aceptar esta orden ordenada con anticipación, nos aseguramos de que ambas partes construyan una salida de transacción de financiamiento idéntica, que está firmada por la firma de la transacción de compromiso intercambiada.

Este script está codificado como una dirección de Bitcoin Pay-to-Witness-Script-Hash (P2WSH), que se parece a esto:

bc1q89ju02heg32yrqdrnqghe6132wek25p6sv6e564znvrvez7tq5zqt4dn02

Construcción de la transacción de financiación

El nodo de Alice ahora puede construir una transacción de financiación, enviando la cantidad acordada con Bob (funding_satoshis) a la dirección multigrado 2 de 2. Supongamos que la financiación_satoshis fue de 140 000 y Alice está gastando una producción de 200 000 satoshi y creando un cambio de 60 000 satoshi. La transacción se parecerá a la Figura 7-4.

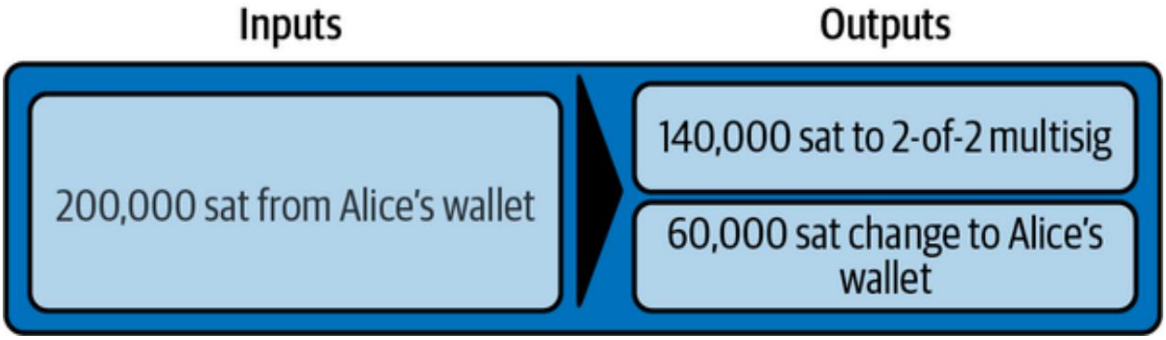


Figura 7-4. Alice construye la transacción de financiación

Alice ***no transmite*** esta transacción porque hacerlo pondría en riesgo sus 140,000 satoshi. Una vez gastado en el multigrado 2 de 2, no hay forma de que Alice recupere su dinero sin la firma de Bob.

CANALES DE PAGO DE DOBLE FINANCIACIÓN

En la implementación actual de Lightning, los canales son financiados solo por el nodo que inicia el canal (Alice en nuestro ejemplo). Se han propuesto canales de doble capitalización, pero aún no se han implementado. En un canal de financiación dual, tanto Alice como Bob contribuirían con insumos a la transacción de financiación. Los canales de doble financiación requieren un flujo de mensajes y un protocolo criptográfico un poco más complicados, por lo que aún no se han implementado, pero están planificados para una futura actualización de los Lightning BOLT. La implementación de c-lightning incluye una versión experimental de una variante en canales de doble financiación.

Retención de transacciones firmadas sin transmisión

Una característica importante de Bitcoin que hace que Lightning sea posible es la capacidad de construir y firmar transacciones, pero no transmitirlas. La transacción es ***válida*** en todos los sentidos, pero hasta que se transmite y confirma en la cadena de bloques de Bitcoin, no se reconoce y sus salidas no se pueden gastar porque no se han creado en la cadena de bloques. Usaremos esta capacidad muchas veces en Lightning Network, y el nodo de Alice usa la capacidad al construir la transacción de financiación: mantenerla y no transmitirla todavía.

Reembolso antes de la financiación

Para evitar la pérdida de fondos, Alice no puede poner su bitcoin en un 2 de 2 hasta que tenga una forma de obtener un reembolso si las cosas salen mal. Esencialmente, debe planificar la "salida" del canal antes de entrar en este arreglo.

Considere la construcción legal de un acuerdo prenupcial, también conocido como "prenupcial". Cuando dos personas contraen matrimonio, su dinero está vinculado por ley (dependiendo de la jurisdicción). Antes de contraer matrimonio, pueden firmar un acuerdo que especifica cómo separar sus bienes si disuelven su matrimonio por divorcio.

Podemos crear un acuerdo similar en Bitcoin. Por ejemplo, podemos crear una transacción de reembolso, que funciona como un acuerdo prenupcial, lo que permite que las partes decidan cómo se dividirán los fondos en su canal antes de que sus fondos se bloqueen realmente en la dirección de financiación de múltiples firmas.

Construcción de la transacción de reembolso prefirmada

Alice construirá la transacción de reembolso inmediatamente después de construir (pero no transmitir) la transacción de financiación. La transacción de reembolso gasta el multisig 2 de 2 en la billetera de Alice. Llamamos a esta transacción de reembolso una transacción de ***compromiso*** porque compromete a ambos socios de canal a distribuir el saldo del canal de manera justa. Dado que Alice financió el canal por su cuenta, obtiene el saldo total, y tanto Alice como Bob se comprometen a reembolsar a Alice con esta transacción.

En la práctica, es un poco más complicado, como veremos en capítulos posteriores, pero por ahora mantengamos las cosas simples y supongamos que se parece a la Figura 7-5.

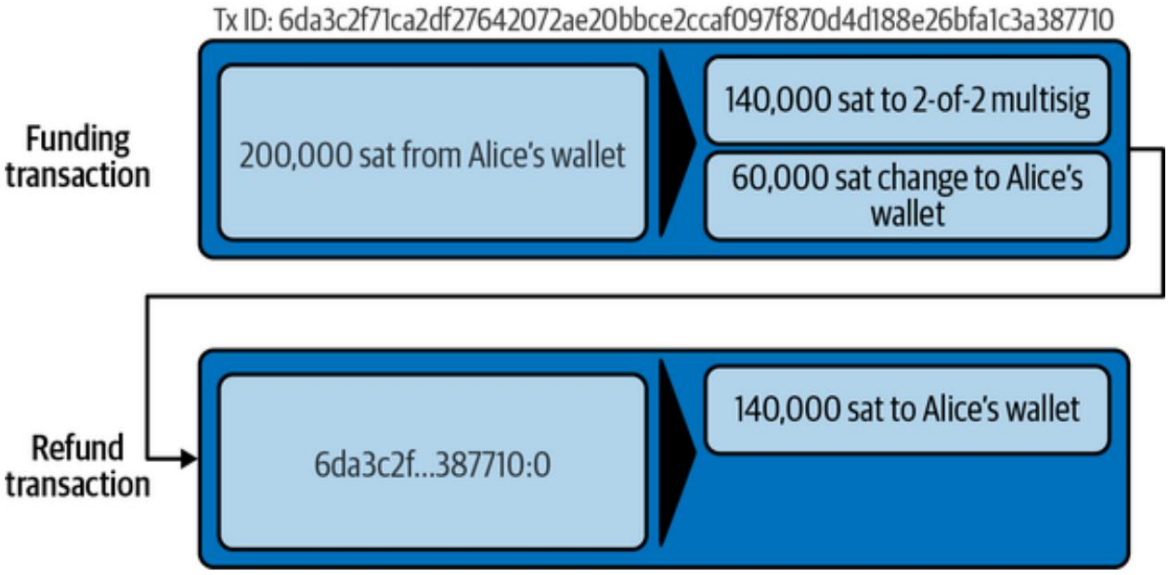


Figura 7-5. Alice también construye la transacción de reembolso

Más adelante en este capítulo veremos cómo se pueden realizar más transacciones de compromiso para distribuir el saldo del canal en diferentes montos.

Encadenamiento de transacciones sin transmisión

Ahora, Alice ha construido las dos transacciones que se muestran en la figura 7-5. Pero quizás te estés preguntando cómo es esto posible. Alice no ha transmitido la transacción de financiación a la cadena de bloques de Bitcoin. En lo que respecta a todos en la red, esa transacción no existe. La transacción de reembolso se construye para ***gastar*** una de las salidas de la transacción de financiación, aunque esa salida tampoco exista todavía. ¿Cómo puede gastar una salida que no ha sido confirmada en la cadena de bloques de Bitcoin?

La transacción de reembolso aún no es una transacción válida. Para que se convierta en una transacción válida deben suceder dos cosas:

- La transacción de financiación debe transmitirse a la red Bitcoin. (Para garantizar la seguridad de Lightning Network, también

requiere que sea confirmado por la cadena de bloques de Bitcoin, aunque esto no es estrictamente necesario para encadenar transacciones).

- La entrada de la transacción de reembolso necesita las firmas de Alice y Bob.

Pero aunque estas dos cosas no hayan sucedido, y aunque el nodo de Alice no haya transmitido la transacción de financiación, aún puede construir la transacción de reembolso. Puede hacerlo porque puede calcular el hash de la transacción de financiación y hacer referencia a él como una entrada en la transacción de reembolso.

¿Observa cómo Alice calculó 6da3c2…387710 como el hash de la transacción de financiación? Siempre y cuando se transmita la transacción de financiación, ese hash se registrará como el ID de transacción de la transacción de financiación. Por lo tanto, se hará referencia a la salida 0 de la transacción de financiación (la salida de dirección 2 de 2) como ID de salida 6da3c2…387710:0. La transacción de reembolso se puede construir para gastar esa salida de la transacción de financiación aunque aún no exista, porque Alice sabe cuál será su identificador una vez que se confirme.

Esto significa que Alice puede crear una transacción encadenada al hacer referencia a una salida que aún no existe, sabiendo que la referencia será válida si se confirma la transacción de financiación, lo que hace que la transacción de reembolso también sea válida.
Como veremos en la siguiente sección, este “truco” de encadenar transacciones antes de que se transmitan requiere una característica muy importante de Bitcoin que se introdujo en agosto de 2017: ***Segregated Witness.***

Resolviendo Maleabilidad (Testigo Segregado)

Alice tiene que depender de que se conozca el ID de transacción de la transacción de financiación antes de la confirmación. Pero antes de la introducción de Segregated Witness (SegWit) en agosto de 2017, esto no era suficiente para proteger a Alice.
Debido a la forma en que se construyeron las transacciones con las firmas (testigos) incluidas en la identificación de la transacción, era posible que un tercero (por ejemplo, Bob) transmitiera una versión alternativa de una transacción con una identificación de transacción ***manipulada*** (modificada). Esto se conoce como ***transacción.***

maleabilidad, y antes de SegWit, este problema dificultaba la implementación segura de canales de pago de por vida indefinidos.

Si Bob pudiera modificar la transacción de financiación de Alice antes de que se confirmara y producir una réplica que tuviera un ID de transacción diferente, Bob podría invalidar la transacción de reembolso de Alice y secuestrar su bitcoin. Alice estaría a merced de Bob para obtener una firma para liberar sus fondos y fácilmente podría ser chantajeada. Bob no podía robar los fondos, pero podía evitar que Alice los recuperara.

La introducción de SegWit hizo que los ID de transacción no confirmados fueran inmutables desde el punto de vista de terceros, lo que significa que Alice podía estar segura de que el ID de transacción de la transacción de financiación no cambiaría. Como resultado, Alice puede estar segura de que si obtiene la firma de Bob en la transacción de reembolso, tendrá una forma de recuperar su dinero. Ahora tiene una forma de implementar el equivalente de Bitcoin de un "prenupcial" antes de bloquear sus fondos en el multisig.

PROPINA

Es posible que se haya preguntado cómo Bob podría alterar (mallear) una transacción creada y firmada por Alice. Bob ciertamente no tiene las claves privadas de Alice. Sin embargo, las firmas ECDSA para un mensaje no son únicas. Conocer una firma (que se incluye en una transacción válida) permite producir muchas firmas de aspecto diferente que aún son válidas. Antes de que SegWit eliminara las firmas del algoritmo de resumen de transacciones, Bob podía reemplazar la firma con una firma válida equivalente que produjera una ID de transacción diferente, rompiendo la cadena entre la transacción de financiación y la transacción de reembolso.

El mensaje financiado_creado

Ahora que Alice ha construido las transacciones necesarias, el flujo de mensajes de construcción del canal continúa. Alice transmite el mensaje financiado_creado a Bob. Puedes ver el contenido de este mensaje aquí:

```
[32*byte:id_canal_temporal] [sha256:id_txid
de financiación]
```

[u16:funding_output_index]
[firma:firma]

Con este mensaje, Alice le da a Bob la información importante sobre la transacción de financiación que ancla el canal de pago:

financiación_txid

Este es el ID de transacción (TxID) de la transacción de financiación y se usa para crear el ID del canal una vez que se establece el canal.

índice_de_producción_de_fondos

Este es el índice de salida, por lo que Bob sabe qué salida de la transacción (p. ej., la salida 0) es la salida multigrado 2 de 2 financiada por Alice. Esto también se usa para formar el ID del canal.

Finalmente, Alice también envía la firma correspondiente a la clave pública de financiación de Alice y se utiliza para gastar del multisig 2 de 2. Bob necesita esto porque también necesitará crear su propia versión de una transacción de compromiso. Esa transacción de compromiso necesita una firma de Alice, que ella le proporciona. Tenga en cuenta que las transacciones de compromiso de Alice y Bob se ven ligeramente diferentes, por lo que las firmas serán diferentes. Saber cómo es la transacción de compromiso de la otra parte es crucial y forma parte del protocolo para proporcionar la firma válida.

PROPINA

En el protocolo Lightning, a menudo vemos nodos que envían firmas en lugar de transacciones completas firmadas. Esto se debe a que cualquiera de las partes puede reconstruir la misma transacción y, por lo tanto, solo se necesita la firma para que sea válida. Enviar solo la firma y no toda la transacción ahorra mucho ancho de banda de red.

El mensaje financiado_firmado

Después de recibir el mensaje de creación de fondos de Alice, Bob ahora conoce el ID de la transacción de fondos y el índice de salida. La identificación del canal se realiza mediante un "o exclusivo" bit a bit (XOR) de la identificación de la transacción de financiación y el índice de salida:

```
id_canal = financiando_txid XOR financiando_índice_de salida
```

Más precisamente, un channel_id, que es la representación de 32 bytes de un UTXO de financiación, se genera mediante XORing de los 2 bytes inferiores del TxID de financiación con el índice de la salida de financiación.

Bob también tendrá que enviarle a Alice su firma para la transacción de reembolso, según la clave pública de financiación de Bob que formó la multifirma 2 de 2.
Aunque Bob ya tiene su transacción de reembolso local, esto le permitirá a Alice completar la transacción de reembolso con todas las firmas necesarias y asegurarse de que su dinero sea reembolsable en caso de que algo salga mal.

Bob construye un mensaje de financiación_firmado y se lo envía a Alice. Aquí vemos el contenido de este mensaje:

```
[id_canal:id_canal] [firma:firma]
```

Difusión de la Transacción de Financiamiento

Al recibir el mensaje de financiación_firmado de Bob, Alice ahora tiene las dos firmas necesarias para firmar la transacción de reembolso. Su "plan de salida" ahora es seguro y, por lo tanto, puede transmitir la transacción de financiación sin temor a que se bloqueen sus fondos. Si algo sale mal, Alice puede simplemente transmitir la transacción de reembolso y recuperar su dinero, sin más ayuda de Bob.

Alice ahora envía la transacción de financiación a la red de Bitcoin para que pueda extraerse en la cadena de bloques. Tanto Alice como Bob estarán atentos a esta transacción y esperando confirmaciones de profundidad mínima (por ejemplo, seis confirmaciones) en la cadena de bloques de Bitcoin.

PROPINA

Por supuesto, Alice usará el Protocolo Bitcoin para verificar que la firma que Bob le envió sea realmente válida. Este paso es muy crucial. Si por alguna razón Bob estaba enviando datos erróneos a Alice, su “plan de salida” sería saboteado.

El mensaje de financiación_bloqueada

Tan pronto como la transacción de financiación haya alcanzado el número requerido de confirmaciones, tanto Alice como Bob se envían el mensaje de financiación_bloqueada entre sí y el canal está listo para su uso.

Envío de pagos a través del canal

El canal se ha configurado, pero en su estado inicial, toda la capacidad (140.000 satoshis) está del lado de Alice. Esto significa que Alice puede enviar pagos a Bob a través del canal, pero Bob aún no tiene fondos para enviar a Alice.

En las próximas secciones, mostraremos cómo se realizan los pagos en el canal de pago y cómo se actualiza el ***estado del canal .***

Supongamos que Alice quiere enviar 70 000 satoshis a Bob para pagar su cuenta en la cafetería de Bob.

Dividir el saldo

En principio, enviar un pago de Alice a Bob es simplemente una cuestión de redistribuir el saldo del canal. Antes de enviar el pago, Alice tiene 140 000 satoshis y Bob no tiene ninguno. Después de enviar el pago de 70 000 satoshis, Alice tiene 70 000 satoshis y Bob tiene 70 000 satoshis.

Por lo tanto, todo lo que Alice y Bob tienen que hacer es crear y firmar una transacción que gaste el multigrado 2 de 2 en dos salidas pagando a Alice y Bob sus saldos correspondientes. Llamamos a esta transacción actualizada una transacción de ***compromiso.***

Alice y Bob operan el canal de pago haciendo ***avanzar el estado del canal*** a través de una serie de compromisos. Cada compromiso actualiza los saldos para reflejar los pagos que han fluido a través del canal. Tanto Alice como Bob pueden iniciar un nuevo compromiso para actualizar el canal.

En la Figura 7-6 vemos varias transacciones de compromiso.

La primera transacción de compromiso que se muestra en la figura 7-6 es la transacción de reembolso que Alice construyó antes de financiar el canal. En el diagrama, este es el Compromiso #0. Después de que Alice paga a Bob 70 000 satoshis, la nueva transacción de compromiso (Compromiso n.° 1) tiene dos salidas que pagan a Alice y Bob sus respectivos saldos. Hemos incluido dos transacciones de compromiso posteriores (Compromiso #2 y Compromiso #3) que representan a Alice pagando a Bob 10,000 satoshis adicionales y luego 20,000 satoshis, respectivamente.

Cada transacción de compromiso firmada y válida puede ser utilizada por cualquiera de los socios del canal en cualquier momento para cerrar el canal transmitiéndolo a la red de Bitcoin. Dado que ambos tienen la transacción de compromiso más reciente y pueden usarla en cualquier momento, también pueden retenerla y no transmitirla. Es su garantía de una salida justa del canal.

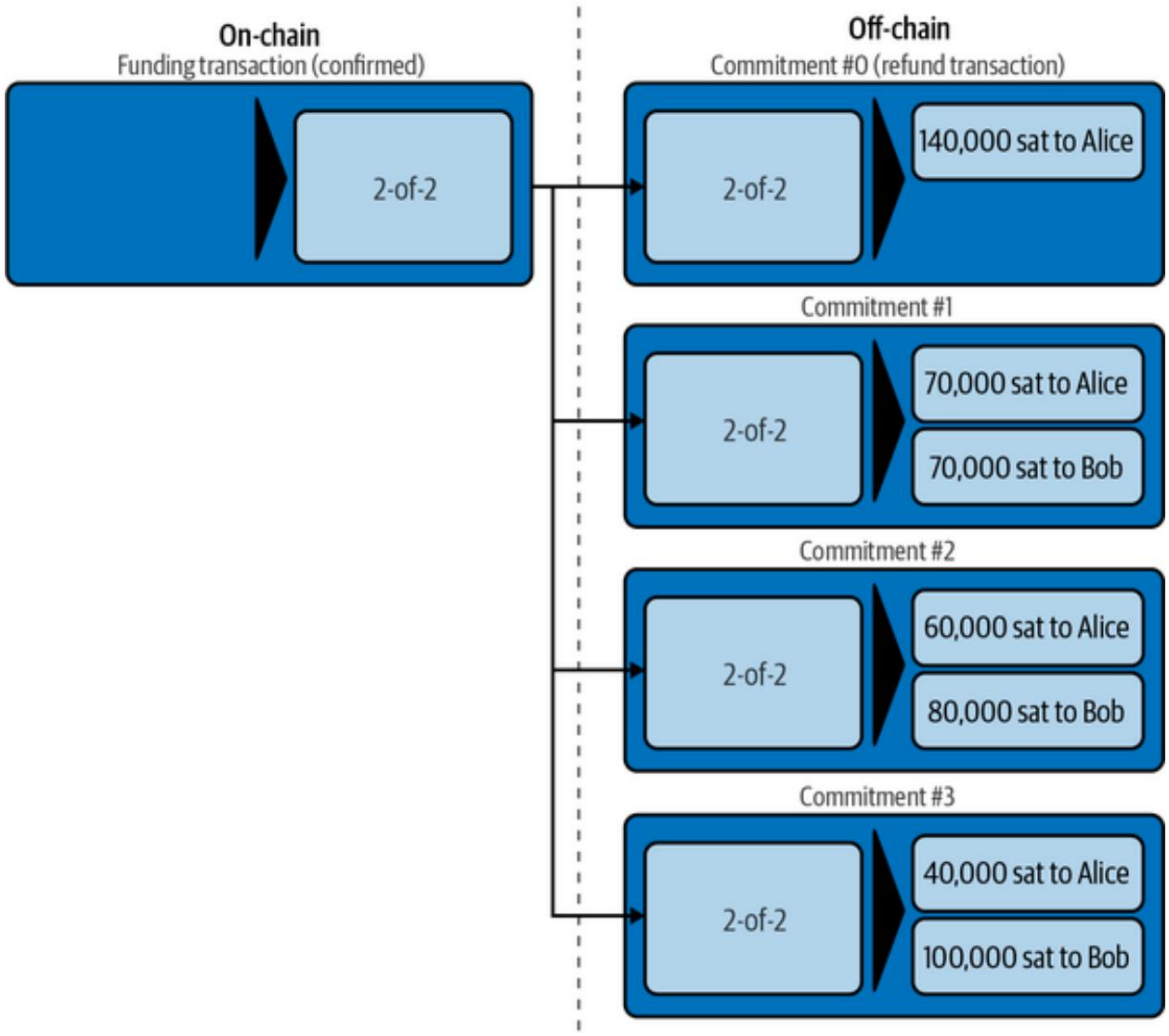


Figura 7-6. Múltiples transacciones de compromiso

Compromisos en competencia

Quizás se pregunte cómo es posible que Alice y Bob tengan múltiples transacciones de compromiso, todas ellas intentando gastar el mismo resultado de 2 de 2 de la transacción de financiación. ¿Estas transacciones de compromiso no son conflictivas? ¿No es esto un "gasto doble" que el sistema Bitcoin pretende evitar?

¡Ciertamente así es! De hecho, confiamos en la capacidad de Bitcoin para ***evitar*** un gasto doble para que Lightning funcione. No importa cuántas transacciones de compromiso construyan y firmen Alice y Bob, solo una de ellas puede confirmarse.

Mientras Alice y Bob mantengan estas transacciones y no las transmitan, la producción de fondos no se gastará. Pero si se transmite y confirma una transacción de compromiso, gastará la producción de fondos. Si Alice o Bob intentan transmitir más de una transacción de compromiso, solo se confirmará una de ellas y las demás se rechazarán como intentos (y fallas) de gastos dobles.

Si se transmite más de una transacción de compromiso, hay muchos factores que determinarán cuál se confirma primero: la cantidad de tarifas incluidas, la velocidad de propagación de estas transacciones competidoras, la topología de la red, etc. Esencialmente, se convierte en una carrera sin un predecible. Salir. Eso no suena muy seguro. Parece que alguien podría hacer trampa.

Hacer trampa con transacciones de compromiso antiguas

Veamos más detenidamente las transacciones de compromiso de la figura 7-6. Las cuatro transacciones de compromiso están firmadas y son válidas. Pero solo el último refleja con precisión los saldos de canales más recientes. En este escenario particular, Alice tiene la oportunidad de hacer trampa al transmitir un compromiso anterior y confirmarlo en la cadena de bloques de Bitcoin. Digamos que Alice transmite el Compromiso #0 y lo confirma: cerrará efectivamente el canal y tomará los 140,000 satoshis ella misma. De hecho, en este ejemplo en particular cualquier compromiso excepto el Compromiso #3 mejora la posición de Alicia y le permite “cancelar” al menos parte de los pagos reflejados en el canal.

En la siguiente sección, veremos cómo Lightning Network resuelve este problema, evitando que los socios de canal utilicen transacciones de compromiso más antiguas mediante un mecanismo de revocación y sanciones. Hay otras formas de evitar la transmisión de transacciones de compromiso más antiguas, como los canales de eltoo, pero requieren una actualización a Bitcoin llamada reenlace de entrada (consulte "Protocolo de Bitcoin e innovación de secuencias de comandos de Bitcoin").

Revocación de transacciones de compromiso antiguas

Las transacciones de Bitcoin no caducan y no se pueden "cancelar". Tampoco pueden ser detenidos o censurados una vez emitidos. Entonces, ¿cómo "revocamos" una transacción que otra persona tiene y que ya ha sido firmada?

La solución utilizada en Lightning es otro ejemplo de un protocolo de equidad.
En lugar de tratar de controlar la capacidad de transmitir una transacción, hay un ***mecanismo de penalización*** incorporado que garantiza que no sea lo mejor para un posible tramposo transmitir una transacción de compromiso anterior. Siempre pueden transmitirlo, pero lo más probable es que pierdan dinero si lo hacen.

> PROPINA
>
> La palabra "revocar" es un nombre inapropiado porque implica que los compromisos anteriores de alguna manera se invalidan y no se pueden transmitir ni confirmar. Pero este no es el caso, ya que las transacciones de Bitcoin válidas no se pueden revocar. En cambio, el protocolo Lightning utiliza un mecanismo de penalización para castigar al socio de canal que transmite un compromiso antiguo.

Hay tres elementos que componen el mecanismo de revocación y sanción del protocolo Lightning:

Transacciones de compromiso asimétrico

Las transacciones de compromiso de Alice son ligeramente diferentes de las de Bob.

gasto retrasado

El pago a la parte que tiene la transacción de compromiso se retrasa (bloqueado en el tiempo), mientras que el pago a la otra parte se puede reclamar de inmediato.

Claves de revocación

Se utiliza para desbloquear una opción de penalización por compromisos antiguos.

Veamos estos tres elementos a su vez.

Transacciones de compromiso asimétrico

Alice y Bob tienen transacciones de compromiso ligeramente diferentes. Veamos específicamente el Compromiso n.º 2 de la Figura 7-6, con más detalle en la Figura 7-7 .

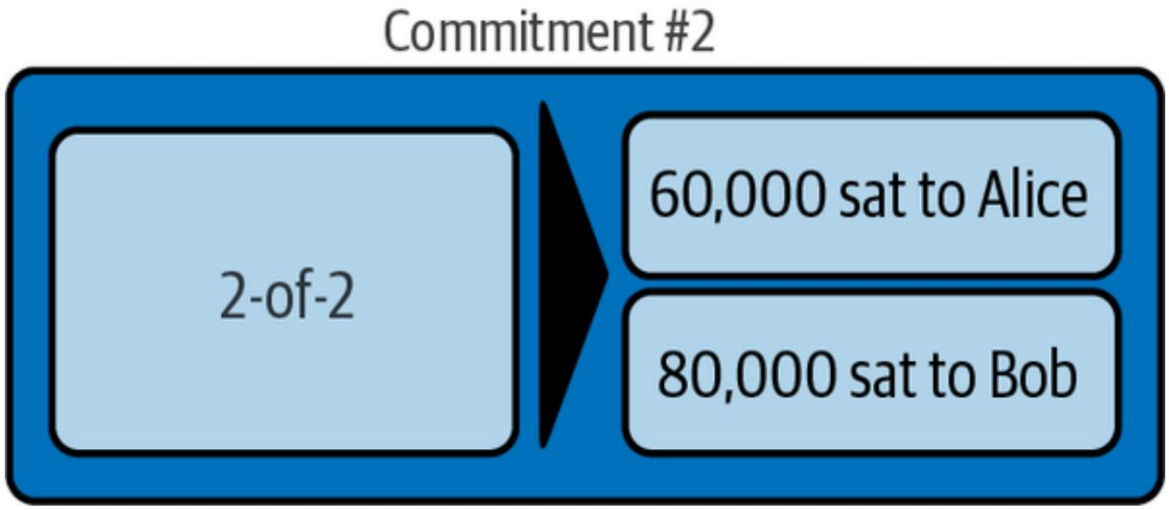


Figura 7-7. Transacción de compromiso #2

Alice y Bob tienen dos variaciones diferentes de esta transacción, como se ilustra en la figura 7-8.

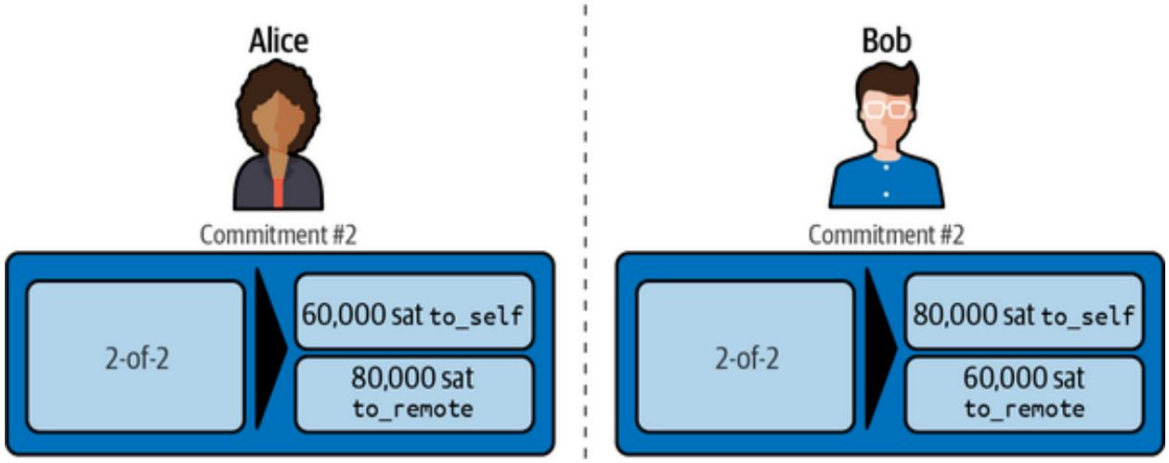


Figura 7-8. Transacciones de compromiso asimétrico

Por convención, dentro del protocolo Lightning, nos referimos a los dos socios de canal como uno mismo (también conocido como local) y remoto, según el lado que estemos viendo. Las salidas que pagan cada socio de canal se denominan to_local y to_remote, respectivamente.

En la Figura 7-8 vemos que Alice tiene una transacción que paga 60 000 satoshis a sí misma (pueden gastarse con las llaves de Alice) y 80 000 satoshis a_remote (pueden gastarse con las llaves de Bob).

Bob tiene la imagen reflejada de esa transacción, donde la primera salida es 80 000 satoshis para sí mismo (pueden gastarse con las llaves de Bob) y 60 000 satoshis para_remoto (pueden gastarse con las llaves de Alice).

Gasto retrasado (con tiempo) para sí mismo

El uso de transacciones asimétricas permite que el protocolo atribuya fácilmente la ***culpa*** a la parte infiel. Una invariante de que la parte ***emisora*** siempre debe esperar asegura que la parte "honesta" tenga tiempo de refutar la demanda y revocar sus fondos. Esta asimetría se manifiesta en forma de salidas diferentes para cada lado: la salida to_local siempre tiene un límite de tiempo y no se puede gastar de inmediato, mientras que la salida to_remote no tiene un límite de tiempo y se puede gastar de inmediato.

En la transacción de compromiso mantenida por Alice, por ejemplo, la salida to_local que le paga tiene un bloqueo de tiempo de 432 bloques, mientras que la salida to_remote que paga a Bob se puede gastar de inmediato (consulte la figura 7-9). La transacción de compromiso de Bob para el Compromiso n.° 2 es la imagen reflejada: su propia salida (to_local) tiene un bloqueo de tiempo y la salida to_remote de Alice se puede gastar de inmediato.

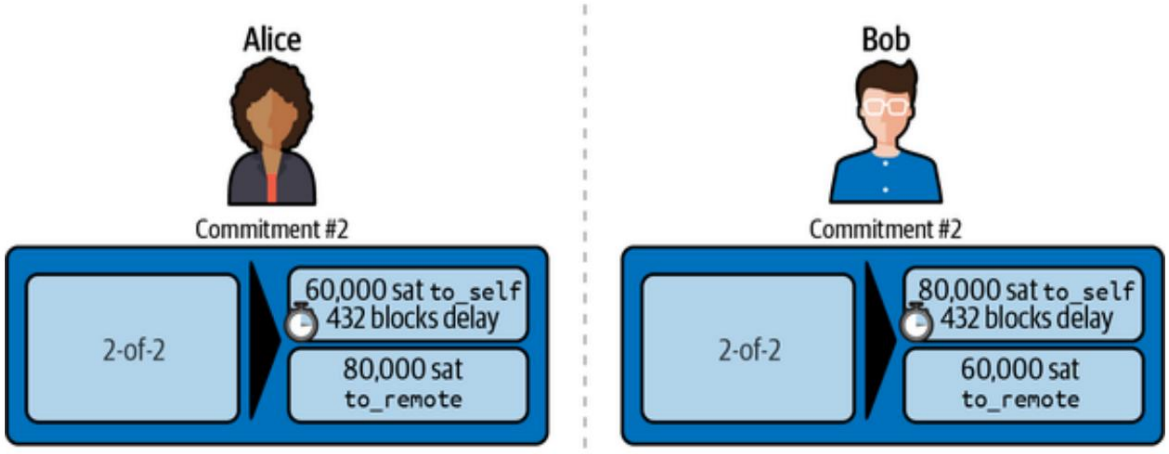


Figura 7-9. Transacciones asimétricas y de compromiso diferido

Eso significa que si Alice cierra el canal transmitiendo y confirmando la transacción de compromiso que tiene, no puede gastar su saldo por 432 bloques, pero Bob puede reclamar su saldo de inmediato. Si Bob cierra el canal utilizando la transacción de compromiso que tiene, no puede gastar su producción en 432 bloques, mientras que Alice puede gastar la suya de inmediato.

La demora existe por una razón: permitir que la parte ***remota*** ejerza una opción de penalización si el otro socio del canal debe transmitir un compromiso anterior (revocado). Veamos las claves de revocación y la opción de penalización

Siguiente.

El retraso lo negocian Alice y Bob, durante el flujo de mensajes de construcción del canal inicial, como un campo denominado to_self_delay. Para garantizar la seguridad del canal, la demora se ajusta a la capacidad del canal, lo que significa que un canal con más fondos tiene demoras más largas en los resultados para sí mismo en los compromisos. El nodo de Alice incluye un to_self_delay deseado en el mensaje open_channel. Si Bob encuentra esto aceptable, su nodo incluye el mismo valor para to_self_delay en el mensaje accept_channel. Si no están de acuerdo, se rechaza el canal (ver “El mensaje de apagado”).

Claves de revocación

Como discutimos anteriormente, la palabra "revocación" es un poco engañosa porque implica que la transacción "revocada" no se puede usar.

De hecho, la transacción revocada se puede usar, pero si se usa y se ha revocado, entonces uno de los socios del canal puede tomar todos los fondos del canal creando una transacción de penalización.

La forma en que esto funciona es que la salida to_local no solo está bloqueada en el tiempo, sino que también tiene dos condiciones de gasto en la secuencia de comandos: puede gastarse por ***sí mismo*** después de la demora del bloqueo de tiempo ***o*** puede gastarse de forma ***remota*** inmediatamente con una clave de revocación para este compromiso . .

Entonces, en nuestro ejemplo, cada lado tiene una transacción de compromiso que incluye una opción de revocación en la salida to_local, como se muestra en la Figura 7-10.

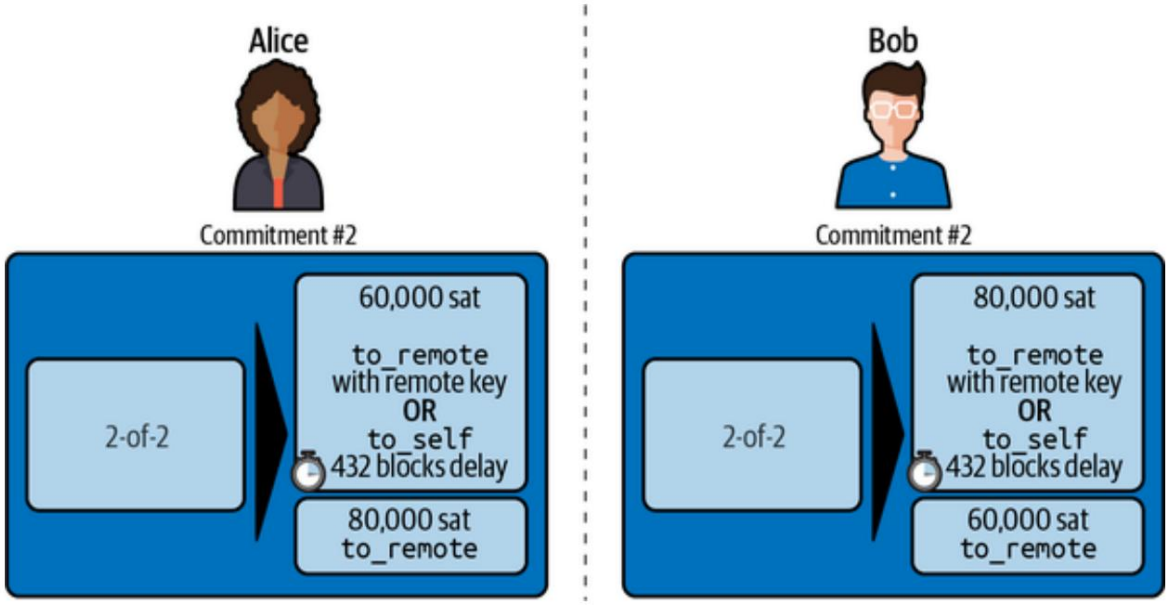


Figura 7-10. Compromisos asimétricos, diferidos y revocables

La transacción de compromiso

Ahora que entendemos la estructura de las transacciones de compromiso y por qué necesitamos compromisos asimétricos, retrasados y revocables, veamos el Bitcoin Script que implementa esto.

La primera salida (to_local) de una transacción de compromiso se define en TORNILLO #3: Transacción de compromiso, salida to_local, como sigue:

```
OP_SI
        # Transacción de penalización
        <recallpubkey>
OP_ELSE
        <to_self_delay>
        OP_CHECKSEQUENCEVERIFY
        OP_DROP
        <local_delayedpubkey>
OP_ENDIF
OP_CHECKSIG
```

Este es un script condicional (consulte "Scripts con múltiples condiciones"), lo que significa que la salida se puede gastar si se cumple ***cualquiera*** de las dos condiciones. La primera cláusula permite que la salida sea gastada por cualquiera que pueda firmar por

<revocaciónpubkey>. La segunda cláusula está bloqueada en el tiempo por bloques <to_self_delay> y solo puede gastarse después de esa cantidad de bloques por cualquier persona que pueda firmar para <local_delayedpubkey>. En nuestro ejemplo, habíamos establecido el bloqueo de tiempo <to_self_delay> en 432 bloques, pero este es un retraso configurable que negocian los dos socios de canal.
La duración del bloqueo de tiempo de to_self_delay generalmente se elige en proporción a la capacidad del canal, lo que significa que los canales de mayor capacidad (más fondos) tienen bloqueos de tiempo de to_self_delay más largos para proteger a las partes.

La primera cláusula permite que cualquier persona que pueda firmar para <revocationpubkey> gaste la salida. Un requisito crítico para la seguridad de este script es que la parte remota ***no puede*** firmar unilateralmente con la clave pública de revocación. Para ver por qué esto es importante, considere el escenario en el que la parte remota incumple un compromiso revocado previamente. Si pueden firmar con esta clave, simplemente pueden tomar la cláusula de revocación ***ellos mismos*** y robar todos los fondos en el canal. En su lugar, derivamos la clave pública de revocación para ***cada*** estado en función de la información de la ***parte*** propia (local) y remota. Se utiliza un uso inteligente de la criptografía simétrica y asimétrica para permitir que ambas partes calculen la clave pública de revocación de la clave pública, pero solo permiten que la parte honesta calcule la clave privada dada su información secreta, como se detalla en "Derivaciones de secreto de revocación y compromiso".

SECRETO DE REVOCACIÓN Y COMPROMISO DERIVACIONES

Cada lado envía un punto base de revocación durante los mensajes de negociación del canal inicial, así como un primer punto de compromiso. El punto base de revocación es estático durante la vida útil del canal, mientras que cada nuevo estado del canal se basará en un nuevo

primer_por_punto_de_compromiso.

Dada esta información, la clave pública de revocación para cada estado del canal se deriva a través de la siguiente serie de operaciones de hash y curva elíptica:

```
revocaciónpubkey = punto_base_revocación *
sha256(punto_base_revocación || por_punto_compromiso) +
por_punto_compromiso * sha256(por_punto_compromiso ||
punto_base_revocación)
```

Debido a la propiedad conmutativa de los grupos abelianos sobre los que se definen las curvas elípticas, una vez que la parte remota revela per_commitment_secret (la clave privada para per_commitment_point), self puede derivar la clave privada para revocationpubkey con la siguiente operación:

```
revocación_priv = (revocaciónbase_priv *
sha256(punto_base_revocación || por_punto_compromiso)) +
 (por_secreto_compromiso * sha256(por_punto_compromiso ||
punto_base_revocación)) mod N
```

Para ver por qué esto funciona en la práctica, observe que podemos ***reordenar*** (conmutar) y expandir el cálculo de la clave pública de la fórmula original para revocaciónpubkey:

```
revocaciónpubkey = G*(revocaciónbase_priv *
sha256(revocación_punto base || por_punto_de_compromiso) + G*
 (por_secreto_de_compromiso * sha256(por_punto_de_compromiso ||
punto_base_de_revocación))
```

= punto_base_de_revocación *
sha256(punto_base_de_revocación || por_punto_de_compromiso) + por_punto_de_compromiso
* sha256(por_punto_de_compromiso || punto_base_de_revocación))

En otras palabras, la revocación base_priv solo puede ser derivada (y utilizada para firmar la revocación pública) por la parte que conoce ***tanto*** la revocación base_priv ***como*** el per_commitment_secret. Este pequeño truco es lo que hace que el sistema de revocación basado en clave pública utilizado en Lightning Network
seguro.

PROPINA

El bloqueo de tiempo utilizado en la transacción de compromiso con CHECK SE QUENCEVERIFY es un bloqueo de ***tiempo relativo.*** Cuenta los bloques transcurridos desde la confirmación de esta salida. Eso significa que no se podrá gastar hasta que se transmita y confirme el bloque to_self_delay ***después de*** esta transacción de compromiso.

La segunda salida (a_remoto) de la transacción de compromiso se define en el PERNO n.° 3: Transacción de compromiso, Salida a_remoto, y en la forma más simple es un Pay-to-Witness-Public-Key-Hash (P2WPKH) para <remote_pubkey>, lo que significa que simplemente paga al propietario que puede firmar para <remote_pubkey>.

Ahora que hemos definido las transacciones de compromiso en detalle, veamos cómo Alice y Bob avanzan en el estado del canal, crean y firman nuevas transacciones de compromiso y revocan transacciones de compromiso antiguas.

Avanzando el estado del canal

Para avanzar en el estado del canal, Alice y Bob intercambian dos mensajes: mensajes de compromiso_firmado y revocación_y_reconocimiento. El mensaje de compromiso_firmado puede ser enviado por cualquier socio de canal

cuando tengan una actualización del estado del canal. El otro socio de canal entonces puede responder con revocar_y_acusar para ***revocar*** el antiguo compromiso y ***acusar recibo*** del nuevo compromiso.

En la Figura 7-11 vemos a Alice y Bob intercambiando dos pares de compromiso_firmado y revocación_y_reconocimiento. El primer flujo muestra una actualización de estado iniciada por Alice (de izquierda a derecha compromiso_firmado), a la que Bob responde (de derecha a izquierda revocar_y_ack). El segundo flujo muestra una actualización de estado iniciada por Bob y respondida por Alice.

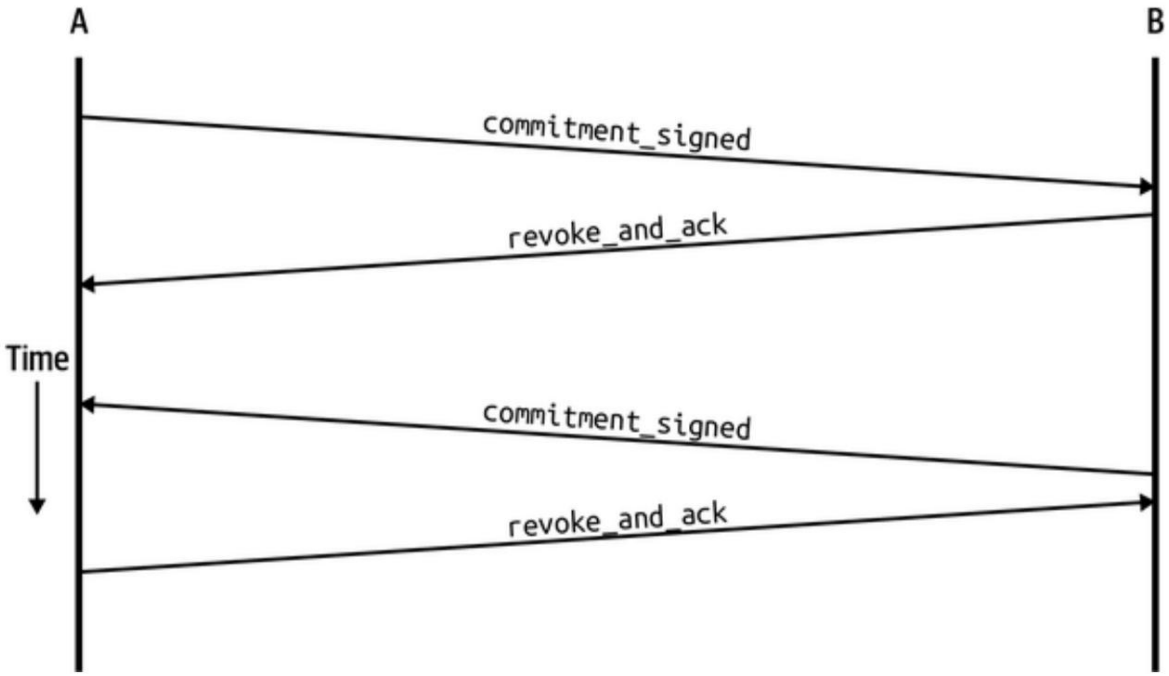


Figura 7-11. Flujo de mensajes de confirmación y revocación

El mensaje compromiso_firmado

La estructura del mensaje de compromiso_firmado se define en BOLT #2: Protocolo de pares, compromiso_firmado y mostrado aquí:

```
[id_canal:id_canal] [firma:firma]
[u16:num_htlcs]
[num_htlcs*firma:htlc_firma]
```

Canal ID

El identificador del canal.

firma

La firma del nuevo compromiso a distancia

num_htlcs

El número de HTLC actualizados en este compromiso

firma_htlc

Las firmas de las actualizaciones.

> **NOTA**
>
> El uso de HTLC para enviar actualizaciones se explicará en detalle en "Contratos de bloqueo de tiempo hash" y en el Capítulo 9.

El mensaje commit_signed de Alice le da a Bob la firma necesaria (la parte de Alice del 2 de 2) para una nueva transacción de compromiso.

El mensaje revoke_and_ack

Ahora que Bob tiene una nueva transacción de compromiso, puede revocar el compromiso anterior dándole a Alice una clave de revocación y construir el nuevo compromiso con la firma de Alice.

El mensaje revoke_and_ack se define en BOLT #2: Peer Protocol, revoke_and_ack, y se muestra aquí:

```
[channel_id:channel_id]
[32*byte:per_commitment_secret]
[point:next_per_commitment_point]
```

Canal ID

Este es el identificador del canal.

por_compromiso_secreto

Se utiliza para generar una clave de revocación para el compromiso anterior (antiguo), revocándolo efectivamente.

siguiente_por_punto_de_compromiso

Se utiliza para generar una clave pública de revocación para el nuevo compromiso, de modo que pueda revocarse más tarde.

Revocar y volver a comprometer

Miremos más de cerca esta interacción entre Alice y Bob.

Alice le está dando a Bob los medios para crear un nuevo compromiso. A cambio, Bob revoca el antiguo compromiso de asegurarle a Alice que no lo usará. Alice solo puede confiar en el nuevo compromiso si tiene la clave de revocación para castigar a Bob por publicar el antiguo compromiso. Desde la perspectiva de Bob, puede revocar con seguridad el antiguo compromiso dándole a Alice las claves para penalizarlo, porque tiene una firma para un nuevo compromiso.

Cuando Bob responde con revoke_and_ack, le da a Alice un per_commitment_secret. Este secreto se puede usar para construir la clave de firma de revocación para el antiguo compromiso, lo que le permite a Alice apoderarse de todos los fondos del canal mediante el ejercicio de una penalización.

Tan pronto como Bob le haya dado este secreto a Alice, no ***debe*** difundir nunca ese antiguo compromiso. Si lo hace, le dará a Alice la oportunidad de penalizarlo tomando los fondos. Esencialmente, Bob le está dando a Alice la capacidad de responsabilizarlo por transmitir un antiguo compromiso y, en efecto, ha revocado su capacidad de usar ese antiguo compromiso.

Una vez que Alice haya recibido el revoke_and_ack de Bob, puede estar segura de que Bob no puede transmitir el antiguo compromiso sin ser

penalizado Ahora tiene las claves necesarias para crear una transacción de penalización si Bob transmite un compromiso antiguo.

Hacer trampa y penalización en la práctica

En la práctica, tanto Alice como Bob tienen que monitorear las trampas. Están monitoreando la cadena de bloques de Bitcoin para cualquier transacción de compromiso relacionada con cualquiera de los canales que están operando. Si ven una transacción de compromiso confirmada en la cadena, verificarán si es el compromiso más reciente. Si se trata de un compromiso “antiguo”, inmediatamente deben construir y transmitir una transacción de sanción. La transacción de penalización gasta ***las*** salidas to_local y to_remote, cerrando el canal y enviando ambos saldos al socio de canal engañado.

Para permitir que ambas partes realicen un seguimiento de los números de compromiso de los compromisos de revocación aprobados, cada compromiso ***codifica*** el número del compromiso dentro de los campos de secuencia y tiempo de bloqueo en una transición. Dentro del protocolo, esta codificación especial se conoce como ***sugerencias de estado.*** Suponiendo que una parte conoce el número de compromiso actual, puede usar las sugerencias de estado para reconocer fácilmente si un compromiso transmitido fue revocado y, de ser así, qué número de compromiso se incumplió, ya que ese número se usa para buscar fácilmente qué el secreto de revocación debe usarse en el árbol de secretos de revocación (shachain).

En lugar de codificar la sugerencia de estado a simple vista, se usa una sugerencia de estado ***ofuscada*** en su lugar. Esta ofuscación se logra primero aplicando XOR al número de compromiso actual con un conjunto de bytes aleatorios generados de forma determinista utilizando las claves públicas de financiación de ambos lados del canal. Se usa un total de 6 bytes a lo largo del tiempo de bloqueo y la secuencia (24 bits del tiempo de bloqueo y 24 bits de la secuencia) para codificar la sugerencia de estado dentro de la transacción de compromiso, por lo que se necesitan 6 bytes aleatorios para XORing. Para obtener estos 6 bytes, ambos lados obtienen el hash SHA-256 de la clave de financiación del iniciador concatenada con la clave de financiación del respondedor. Antes de codificar la altura de compromiso actual, el entero se somete a XOR con este ofuscador de sugerencias de estado y luego se codifica en los 24 bits inferiores del tiempo de bloqueo y los 64 bits superiores de la secuencia.

Revisemos nuestro canal entre Alice y Bob y mostremos un ejemplo específico de una transacción de penalización. En la Figura 7-12 vemos los cuatro compromisos en el canal de Alice y Bob. Alice ha hecho tres pagos a Bob:

- 70,000 satoshis pagados y comprometidos con Bob con el Compromiso #1
- 10,000 satoshis pagados y comprometidos con Bob con el Compromiso #2
- 20,000 satoshis pagados y comprometidos con Bob con el Compromiso #3

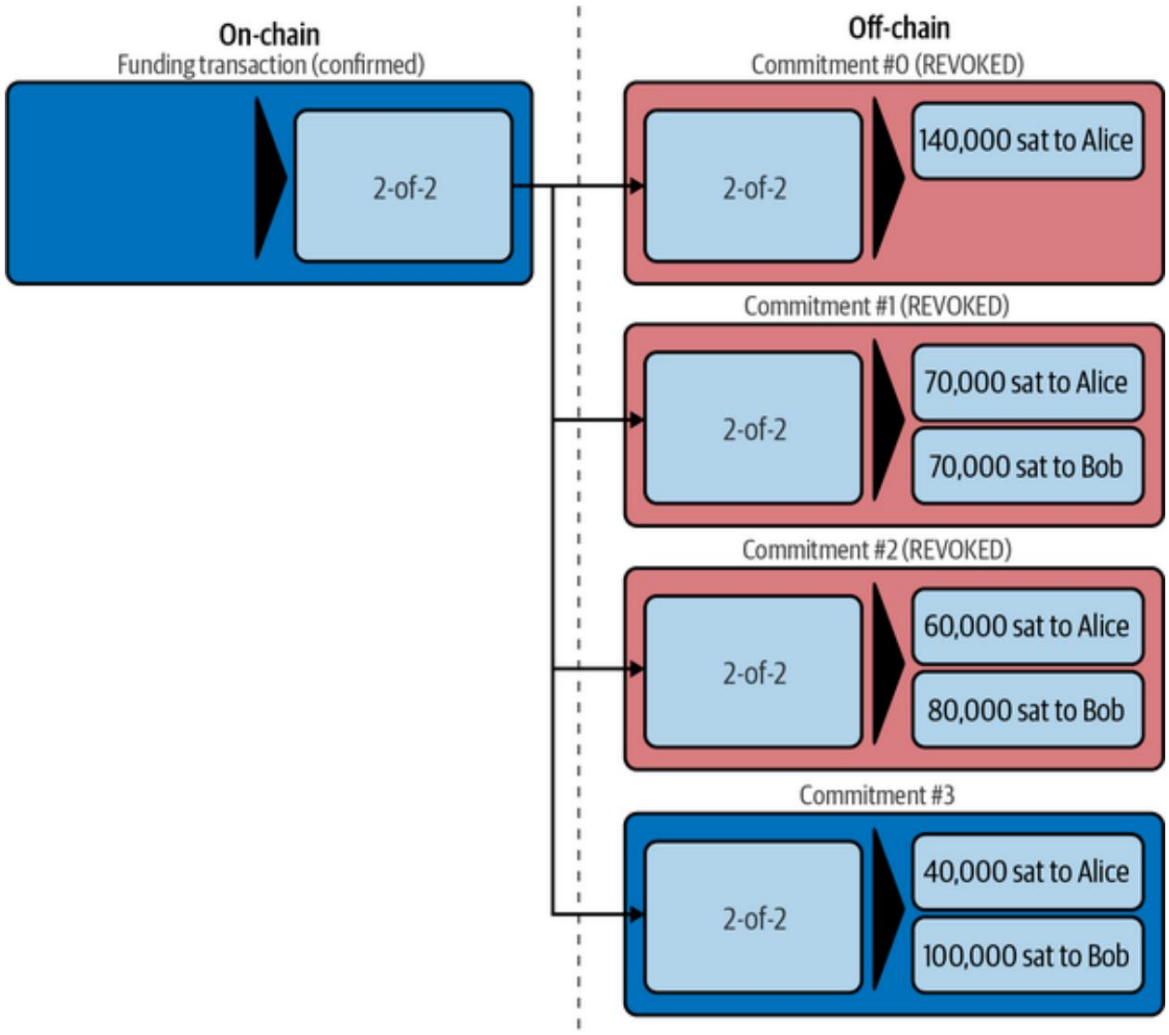


Figura 7-12. Compromisos revocados y vigentes

Con cada compromiso, Alice ha revocado el compromiso anterior (más antiguo). El estado actual del canal y el saldo correcto está representado por el Compromiso #3. Todos los compromisos anteriores han sido