UNIVERSIDAD DE VALLADOLID

E. U. de Informática (Valladolid)

Grado en ingeniería informática

Estudio de soluciones para permitir pequeños pagos con bitcoin utilizando Lightning Network

Autor:
Marco de Benito Fernández

Tutor:
Diego R. Llanos Ferraris

Agradecimientos

A Diego R. Llanos Ferraris, catedrático y profesor en el área de Arquitectura y Tecnología de Computadores en la UVa y tutor académico de este trabajo fin de grado, por la ayuda recibida durante la realización del trabajo y memoria.

Al Grupo Trasgo por facilitarme un lugar de trabajo en sus instalaciones.

Finalmente, quiero agradecerle a mi familia el apoyo recibido durante toda mi etapa universitaria.

Resumen

El Bitcoin, a pesar de ser tan famoso, todavía es una moneda poco extendida en la sociedad. Con el fin de promover su introducción en tareas cotidianas, en este trabajo hemos realizado un análisis acerca de la red Lightning, un servicio de segunda capa de Bitcoin que permite la realización de micropagos con bitcoins. A esto le acompaña un estudio detallado de las necesidades básicas que tendría una empresa pequeña que quiera disponer de pagos por la red Lightning en su negocio.

Abstract

Even with all the fame Bitcoin has accumulated this past few years, it still remains a concept unknown for the majority of people. In this project we have carried out an analyisis of the Lightning Network, a second layer Bitcoin service that allows micropayments with bitcoins.Additionaly, a detailed study can be found on the basic needs a business would need to consider in the case of wanting to implement Bitcoin payments in their commerce.

Índice general

1. Introducción **1**

2. Planificación del proyecto **5**

2.1. Objetivos 5

2.2. Planificación de tareas 6

2.3. Análisis de riesgos 6

2.4. Estimación de presupuesto 9

3. Bitcoin **11**

3.1. Orígenes del Bitcoin 11

3.2. Blockchain 13

3.3. Transacciones y minado 16

3.4. Llaves y *wallets* 18

4. Lightning Network **21**

4.1. Características principales 22

4.2. Componentes de la Lightning Network 23

4.2.1. Carteras 23

4.2.2. Canales 24

4.3. Enrutamiento de Cebolla 25

4.4. Contratos Hash Time-Locked 27

4.4.1. Elección de camino y protocolo Gossip 30

4.4.2. Enrutamiento de cebolla en HTLC 31

5. Instalación y uso de un nodo Lightning **35**

5.1. Testnet 35
5.2. Opción 1: instalación de nodo Lightning 36
5.2.1. Bitcoin 36
5.2.2. Lightning Network 37
5.3. Opción 2: alquiler de nodo Lightning 41
5.3.1. Creación de un nodo con Voltage 41
5.3.2. Conexión remota a un nodo Lightning 44
5.4. Lightning Pool 47
5.5. Rentabilidad de nodos intermedios 48

6. Instalación y uso de una cartera Lightning 51
6.1. Cartera LND 51
6.2. Aplicaciones de carteras 54
6.2.1. Tipos de carteras 55
6.2.2. Uso de aplicaciones de carteras 56

7. Lightning Network en negocios 59
7.1. Negocios gestionados por una persona 60
7.1.1. Nodo personal 60
7.1.2. Alquiler de nodos 60
7.1.3. Aplicaciones de carteras 60
7.2. Negocios con más de un gestor 61
7.2.1. Nodo personal 61
7.2.2. Alquiler de nodos con aplicaciones de carteras 62
7.3. Negocios con más de una sede 62
7.4. Negocios con varios propietarios 63
7.5. Análisis económico de la red Lightning 64

8. Conclusiones y Líneas futuras 67
8.1. Conclusiones 67
8.2. Líneas futuras 68

Anexo **69**

Bibliografía **74**

Índice de figuras

2.1. Diagrama de Gantt del proyecto. 6
2.2. Diagrama de Gantt del proyecto final. 9

3.1. Ejemplo de *blockchain*. 15
3.2. Transacción de bitcoins 17

4.1. Camino de Pablo a Lucía. 25
4.2. Carta sellada para Lucía. 26
4.3. Carta enviada a Sara. 26
4.4. Camino de Pablo a Lucía para pago de 1 BTC. 28
4.5. 1ª parte de la transacción entre Pablo y Lucía. 29
4.6. 2ª parte de la transacción entre Pablo y Lucía. 29
4.7. Balance final de la transacción entre Pablo y Lucía. 30
4.8. De arriba a abajo: mensajes de Sara, Sergio, Miguel y Lucía. 32
4.9. Obtención del mensaje que Pablo envía a Sara. 33

5.1. Bitcoin Core en la red Testnet 37
5.2. Menú principal de Voltage. 42
5.3. Menú principal de Nodes. 42
5.4. Opciones de creación de nodo. 43
5.5. Menú principal del nodo. 43
5.6. Conexión a un nodo en Zeus. 44
5.7. Menú principal del nodo. 45
5.8. Creación de una dirección. 46
5.9. Sección de canales en Zeus. 47

5.10. Ingresos generados a partir de transacciones por valor α. 48
5.11. Ingresos por número de transacciones. 49

6.1. *Invoice* generado en Starblocks. 52
6.2. Confirmación de pago realizado. 53
6.3. Generación de un *invoice* (payment_request). 53
6.4. Código QR creado con Angularx-qrcode. 54
6.5. Código QR creado con Angularx-qrcode. 54

7.1. Gráficos de sectores relativos a las tasas por transacción 64

Índice de tablas

2.1. R01 - Problemas en la instalación de nodos. 7
2.2. R02 - Planificación del proyecto. 7
2.3. R03 - Contenido escaso para la realización del trabajo. 7
2.4. R04 - Imposibilidad de realizar pruebas en Testnet. 7
2.5. R05 - Problemas con el segundo TFG. 8
2.6. Coste de un equipo informático. 9
2.7. Coste energético. 9
2.8. Coste de un empleado. 10

3.1. Contenido de la cabecera de un bloque 15

6.1. Principales carteras y algunas de sus características 58

Glosario

- **Add-on:** programas que solo funcionan integrados en otros y que complementan sus funcionalidades.
- **Back-end:** en una aplicación, se refiere a los procesos encargados de acceder a datos.
- **Bitcoin:** criptodivisa descentralizada que se basa en la tecnología blockchain para proporcionar un sistema de pago.
- **Blockchain:** sistema de registro compartido, inmutable y descentralizado.
- **Bloque:** unidades de almacenamiento que componen una cadena de bloques o *blockchain*
- **Checksum:** una función que tiene el propósito de detectar cambios accidentales en un código.
- **Contrato hash time locked:** un sistema de pago en el que el receptor acepta recibir el pago antes de una tiempo tope.
- **Daemon:** un tipo especial de programa que se ejecuta en segundo plano.
- **Hash:** código único resultado de aplicar una función *hash* a una palabra o frase.
- **Hash rate:** hace referencia a la potencia de creación de *hashes*, normalmente medido en número de *hashes* generados por segundo.
- **Invoice:** un código que contiene cantidad de satoshis, y dirección a la que enviar un pago en la Lightning Network como
- **Llave:** un código que identifica a una cartera. Existen llaves públicas y privadas.
- **Nodo:** servidores que mantienen una copia de la *blockchain*.
- **Off-chain:** referido a transacciones. Que no se registran en la *blockchain*.
- **P2P (Peer to peer)**: una red en la que todos los participantes son cliente y servidor.
- **Proof of Work:** un método de verificación criptográfica en el que un individuo certifica que ha utilizado cierta capacidad computacional.
- **Protocolo Gossip:** un sistema eficiente que permite la comunicación de una red P2P

- **Multiplicación elíptica curva**: operación de añadir un número a sí mismo a lo largo de una curva elíptica.

Capítulo 1

Introducción

En esta introducción, como es preceptivo, exponemos de forma concisa el planteamiento del problema, antecedentes, objetivos, y la secuenciación seguida en este trabajo.

Planteamiento

Las criptomonedas son un medio digital de intercambio que han sido uno de los mayores fenómenos económicos de estos últimos años.

La irrupción de las criptomonedas ha supuesto la introducción de un nuevo método de pago alternativo a los establecidos. Más concretamente, el Bitcoin, la criptomoneda por excelencia, se ha comenzado a utilizar en algunas transacciones de bienes y servicios.

Aun así, existen varias razones por las que el Bitcoin todavía no se ha extendido demasiado como forma de pago. De entre ellas podemos destacar las siguientes:

- Es un servicio que requiere de varios minutos para realizar un pago. Debido a la tecnología que utiliza, la realización de un pago de forma instantánea es imposible, y por lo tanto, no es la mejor opción para efectuar pagos rutinarios.
- Al tratarse de un servicio basado en nueva tecnología, está poco extendido en la sociedad. El uso de servicios basados en nuevas tecnologías por parte de la sociedad requiere de un periodo de introduccion y maduración. Si además añadimos la inversión monetaria que necesita es normal descartar la idea cuando ya existen formas de pago mejor establecidas.

La imposibilidad de realizar pagos instantáneos mediante el ecosistema Bitcoin reside en la propia naturaliza de la divisa, y por lo tanto, es un problema persistente al que se ha tratado de dar soluciones.

La Lightning Network o red Lightning, es un servicio de segunda capa que funciona sobre Bitcoin y que permite la realización de micropagos en Bitcoin al instante. Esto obviamente resuelve el primer problema, pero al ser otro servicio, el usuario que desee

utilizarlo necesitará aún más conocimiento sobre su funcionamiento y esto no hace más que aumentar el segundo problema.

Además, la Lightning Network tiene la característica de que realiza operaciones *off-chain*, es decir, que Bitcoin no las registra. Esta circunstacia que en principio podría parecer un problema resulta ser beneficioso para el sistema Bitcoin. Debido a la naturaleza del Bitcoin, existe un límite de transacciones diarias que ronda las 600.000, y aunque actualmente se realicen unas 200.000, es posible que en un futuro surjan problemas asociados a este límite si no se toman medidas. Por suerte, como ya hemos comentado, la mayor parte de las transacciones de la Lightning Network son independientes de Bitcoin y no suponen una sobrecarga para la *blockchain* inherente a la gestión de Bitcoin.

Este nuevo sistema de pago hace necesario que los usuarios deban utilizar un software y hardware que permita realizar las necesarias transacciones. Para un usuario final estándar existen múltiples aplicaciones móviles que facilitan la tarea, pero en el caso de que un negocio trate de ofrecer un servicio de pago por Bitcoin, necesitará tener en cuenta varias alternativas y elegir la que mejor se adecúe a su situación.

Este trabajo pretende realizar un estudio sobre las diferentes formas en las que un negocio puede hacer uso de micropagos con bitcoins por medio de la red Lightning, siendo necesario explicar a su vez los principales conceptos en los que se basa Bitcoin y la red Lightning para facilitar una comprensión de su funcionamiento.

Antecedentes

La Lightning Network fue creada en 2015 por Thaddeus Dryja y Joseph Poon como posible solución al problema de escalabilidad (el problema asociado al máximo número de transacciones diarias). En 2016, los creadores fundaron, junto a un grupo reducido de personas, Lightning Labs, una empresa dedicada al desarrollo de la red Lightning. Y finalmente, en 2018 Lightning Labs lanzó una versión beta de la Lightning Network implementada en la red principal de Bitcoin.

En 2019, como forma de promocionar el Bitcoin, el usuario de Twitter "hodlonaut" comenzó una "antorcha de bitcoins" usando la red Lightning, por la cual se aumentó una cierta cantidad de bitcoins a medida que transitaba por las manos de varios individuos de confianza hasta llegar al límite permitido por la red. Uno de estos individuos fue Jack Dorsey, cofundador y exdirector ejecutivo de Twitter, el cual, viendo el potencial de la red, contrató a un equipo de trabajadores para que centraran sus esfuerzos en el desarrollo de la red Lightning.

Desde entonces, la cantidad de bitcoins dedicados al uso de la red Lightning se ha multiplicado por 7 llegando al máximo actual de 4000, y se prevee que para el 2030 existan 700 millones de usuarios en la red.

El desarrollo de la red Lightning permite utilizar bitcoins para realizar micropagos, y al no aumentar el número de transacciones diarias, permite la generalización del uso de Bitcoin en situaciones cotidianas. El uso de esta nueva tecnología obviamente requerirá

que los negocios que quieran implementar un servicio de pago por medio de la Lightning Network tomen las medidas necesarias para ello.

La Lightning Network es un sistema muy desconocido del que todavía no existen manuales para un público menos experimentado. Esta falta de documentación ha sido la principal causa que ha impulsado el comienzo de este trabajo.

Desde un punto de vista más personal, las prácticas en empresa que realicé estuvieron estrechamente relacionadas con el mundo de las criptomonedas. En concreto trataban el uso de Smart Contracts con Ethereum para registrar compras y ventas de energía. Aunque Ethereum esté más enfocado al desarrollo de aplicaciones que Bitcoin, si que me familiaricé con algunos de los inconvenientes que presenta Bitcoin. Así que el problema de micropagos instantáneos que pretende arreglar la red Lightning me pareció una opción interesante que valía la pena investigar.

Objetivos

Hemos visto algunos de los principales inconvenientes que existe para la difusión de los pagos con criptodivisas, y como la red Lightning puede solucionar algunos de estos problemas.

El objetivo principal de este trabajo es determinar la configuración más adecuada de hardware y software para que un negocio pueda ofrecer pagos por medio de la red Lightning. Esto supone realizar una investigación previa acerca de todas las opciones posibles de uso de la Lightning Network, y analizarlas en función de las ventajas y desventajas que ofrecen.

Para llevar a cabo un análisis apropiado habrá que instalar el software necesario que cada una de las configuraciones requiera y aprender a usarlas.

Durante el proceso de instalación se realizará un resumen de las principales aplicaciones y servicios software disponibles, y se citarán algunas de sus principales características que faciliten a los usuarios la elección de la mejor alternativa atendiendo a sus necesidades.

Tras realizar todas las instalaciones vendrá la fase de pruebas en la que se realizarán transacciones entre dispositivos de varias maneras con el fin de validar la funcionalidad del sistema.

Para poder lograr todos estos objetivos será clave obtener un conocimiento sólido acerca del funcionamiento de Bitcoin, y más específicamente de la red Lightning.

Estructura

Para concluir esta introducción haremos un breve comentario sobre el contenido de los capítulos que componen el trabajo.

En el primer capítulo "*Introducción*" se resumen el estado del arte, contenidos, motivación, y objetivos del trabajo.

En el segundo capítulo "*Planificación del proyecto*" se indican los objetivos que se pretenden conseguir, acompañado de una estimación de tiempo necesario para realizar las tareas asociadas. También se incluirá una lista de riesgos que puedan ocurrir durante la realización del trabajo, y por último, incluiremos un presupuesto aproximado del coste de realizar este trabajo.

En el tercer capítulo "*Bitcoin*"se realiza un resumen del marco histórico y las condiciones económicas en las que se desarrolla Bitcoin, seguido de un análisis de las herramientas principales que soportan el funcionamiento de Bitcoin.

En el cuarto capítulo "*Lightning Network*" se presentan algunas de las ventajas y los problemas que la red Lightning pretende arreglar. Al igual que con Bitcoin, se describen los componentes principales de la red, y se profundiza en el procedimiento de realización de transacciones haciendo hincapié en los contratos *hash time-locked* (HTLC) propios de la red Lightning.

En el quinto capítulo "*Instalación y uso de un nodo Lightning*" se describe la naturaleza de la red de pruebas de Bitcoin (Testnet), se analizan los requisitos necesarios para instalar un nodo Lightning y para contratar uno, desarrollando también la forma particular de usarlos. Por último se realiza un pequeño estudio acerca de la posibilidad del alquiler de canales en la red, y del posible negocio basado en la posesión de nodos.

En el sexto capítulo "*Instalación y uso de una cartera Lightning*" se describe el proceso de instalación y uso de diferentes tipos de carteras Lightning. Adicionalmente, se presentará un resumen de las principales características que poseen las carteras más conocidas.

En el séptimo capítulo "*Recepción de pagos por Lightning Network en un pequeño negocio: espacio de soluciones*" está dedicado a desarrollar las opciones que tiene un negocio para poder ofrecer pagos en la Lightning Network atendiendo a diferentes necesidades que pueda tener. A esto le acompaña un somero análisis económico del gasto que supone mantener un nodo Lightning comparado con un servicio de pago por medio de tarjetas de crédito/débito.

Por último, en el octavo capítulo "*Conclusiones y líneas futuras*" se presenta un apartado en el que se exponen los resultados más relevantes derivados de este trabajo, y se presentan algunas posibles líneas de actuación futuras.

Capítulo 2

Planificación del proyecto

La planificación de este proyecto se ha dividido en 4 partes:

- Objetivos: una serie de propósitos que serán necesarios para realizar el trabajo.
- Planificación de tareas: unos diagramas que muestren las tareas necesarias a realizar, su fecha de inicio y su duración.
- Análisis de riesgos: un estudio sobre los riesgos que pueden surgir durante el desarrollo del trabajo y como afectarían a la realización del mismo.
- Estimación de presupuesto: estimación de los fondos necesarios para realizar el proyecto.

2.1. Objetivos

A continuación se presentan los principales objetivos que necesitaremos lograr para llevar a cabo este proyecto.

- Estudio inicial: es necesario informarse acerca de los componentes principales y el funcionamiento de Bitcoin y de la red Lightning.
- Nodo Lightning : habrá que indagar sobre como instalar un nodo Lightning y se investigarán maneras alternativas de obtener un nodo. Debido a que es un servicio que funciona sobre Bitcoin, también tendremos que conseguir un nodo de Bitcoin.
- Investigación acerca de carteras: las carteras son necesarias para participar en la red Lightning. Así que tendremos que realizar una investigación acerca de que tipos de carteras hay disponibles y cuáles son sus principales características.
- Realización de pruebas: tendremos que comprobar el método de funcionamiento, y que el nodo y la cartera funcionan correctamente.

- Análisis de opciones para negocios: utilizando el conocimiento obtenido y las pruebas realizadas, habrá que especificar las opciones de las que dispone un negocio que quiera poder utilizar la red Lightning

2.2. Planificación de tareas

A partir de los objetivos explicados en el apartado anterior hemos realizado el siguiente diagrama para la planificación del proyecto

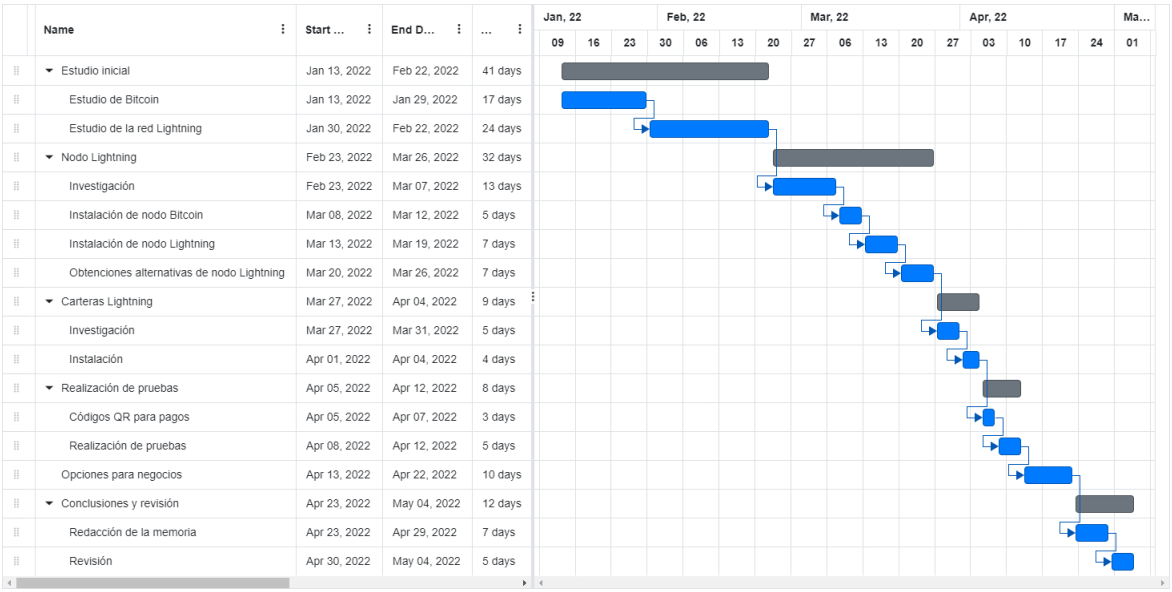


Figura 2.1: Diagrama de Gantt del proyecto.

Fuente: [1]

El diagrama de Gantt que refleja la realización del trabajo real la presentaremos tras el análisis de riesgos.

2.3. Análisis de riesgos

En esta sección se presentan algunos posibles inconvenientes que podrían encontrarse durante el proyecto, que efecto tendrían, y como podrían mitigar su efecto.

Identificador	R01 - Problemas en la instalación de nodos
Descripción	Problema que puedan causar retrasos o la imposibilidad de instalar un nodo Bitcoin/Lightning
Impacto	Medio
Probabilidad	Baja
Plan de mitigación	Replanificación de las tareas, o en caso de que no se consiga instalar, documentar las razones e investigar maneras alternativas

Tabla 2.1: R01 - Problemas en la instalación de nodos.

Identificador	R02 - Planificación del proyecto erróneo
Descripción	Mala estimación temporal con respecto a las tareas del proyecto
Impacto	Alto
Probabilidad	Media
Plan de mitigación	Reestructurar el trabajo para priorizar las partes fundamentales

Tabla 2.2: R02 - Planificación del proyecto.

Identificador	R03 - Contenido escaso para la realización del trabajo
Descripción	La cantidad de investigación y desarrollo que requiere el trabajo no justifican la realización de un TFG
Impacto	Alto
Probabilidad	Baja
Plan de mitigación	Replanificación de los objetivos

Tabla 2.3: R03 - Contenido escaso para la realización del trabajo.

Identificador	R04 - Imposibilidad de realizar pruebas en Testnet
Descripción	Debido a que la mayoría de carteras se utilizan en la red principal, es posible que con las pocas disponibles tengamos problemas en la realización de transacciones.
Impacto	Medio
Probabilidad	Medio
Plan de mitigación	Descargar y utilizar la red principal

Tabla 2.4: R04 - Imposibilidad de realizar pruebas en Testnet.

Identificador	R05 - Problemas con el segundo TFG
Descripción	Problemas con el otro TFG que reduzcan la cantidad de tiempo disponible.
Impacto	Alto
Probabilidad	Bajo
Plan de mitigación	Reestructuración de ambos TFGs priorizando las partes esenciales.

Tabla 2.5: R05 - Problemas con el segundo TFG.

A continuación indicamos los principales problemas que se han encontrado y que han afectado al desarrollo esperado del trabajo.

- Se ha subestimado ligeramente la cantidad de tiempo necesario para asimilar el contenido teórico de Bitcoin y la red Lightning.
- Se han encontrado problemas en la instalación del nodo Bitcoin. En un principio, se planteó instalar el nodo en una Raspberry ya que el nodo tiene que estar constantemente encendido. Tras varios errores fatales del proceso de instalación, se investigó la razón y se llegó a la conclusión de que se necesita al menos una Raspberry pi 4 o posterior para instalar Bitcoin (se disponía de una pi 3). Finalmente el nodo se instaló en un ordenador en las instalaciones del grupo Trasgo.
- Se ha subestimado gravemente la variedad de carteras Lightning y la información relativas a ellas. Además, la búsqueda de carteras que permitan acceso a la red de pruebas es reducida y se han encontrado problemas a la hora de la configuración.
- En la realización de pruebas se han vuelto a sufrir problemas con algunas carteras al intentar realizar transacciones con ellas.

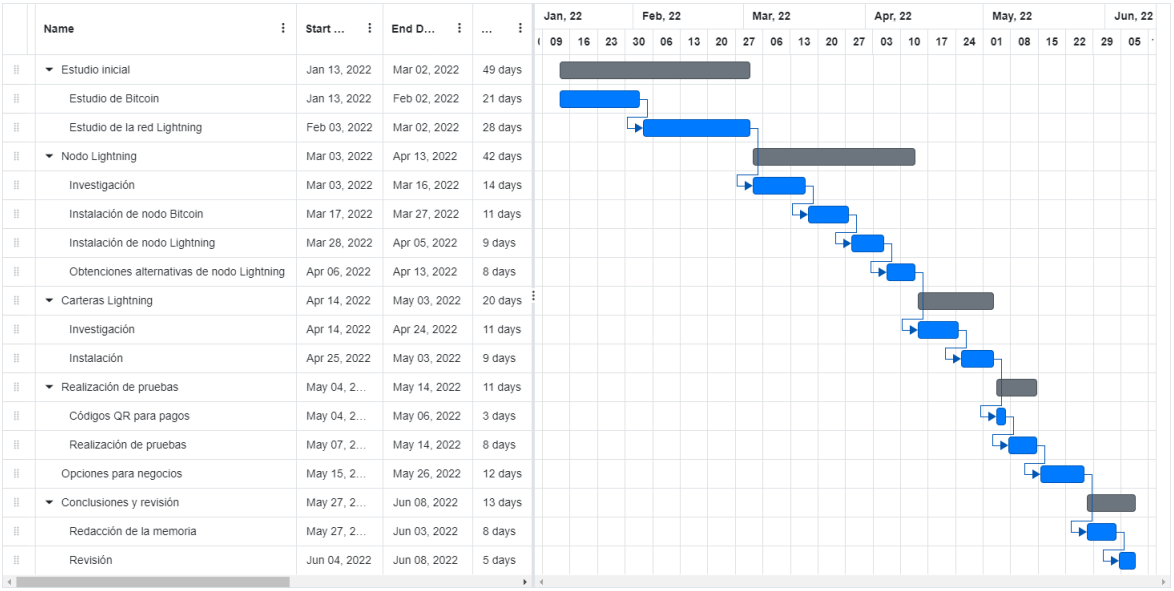


Figura 2.2: Diagrama de Gantt del proyecto final.

Fuente: [1]

2.4. Estimación de presupuesto

Finalmente exponemos una estimación de los fondos necesarios para realizar este proyecto.

Comenzando por el coste de un equipo, los costes teniendo en cuenta 8 horas por jornada y 253 días laborables al año pueden verse en la tabla 2.6.

Precio	Amortización	€/hora
1200€	4 años	0.14

Tabla 2.6: Coste de un equipo informático.

El coste energético teniendo en cuenta que el equipo ha estado encendido constantemente durante unos 72 días, con un gasto medio de 210 €/h, y estimando unos 50W ya que la pantalla se mantiene apagada durante la mayor parte del tiempo puede verse en la tabla 2.7.

Horas	€/MWh	Gasto del equipo	Gasto total
1728	210	50W	18.14 €

Tabla 2.7: Coste energético.

Por último, el trabajo en horas de un recién graduado estimando el sueldo en 18 €/h y que se han trabajado unos 120 días durante una media de 3 horas diarias puede verse en la tabla 2.8.

Horas	€/h	Coste
360	18	6480€

Tabla 2.8: Coste de un empleado.

Con esto, el coste total del trabajo acabaría siendo 7698.14€.

Capítulo 3

Bitcoin

Conceptualizada en 2008 por Satoshi Nakamoto y lanzada en 2009, Bitcoin es la criptomoneda con mayor capitalización del mercado.

Bitcoin es un conjunto de ideas y tecnologías que forman un sistema de pago virtual. Los usuarios pueden intercambiar bitcoins entre ellos de manera directa utilizando el protocolo Bitcoin, un protocolo de código abierto utilizado en ordenadores y en dispositivos móviles.

Bitcoin no solo tiene la ventaja de ser una moneda descentralizada *peer to peer*, si no que al utilizar la tecnología *blockchain*, se convierte en un sistema transparente y público, algo clave en un sistema monetario [2].

En este capítulo introductorio comentaremos los orígenes del Bitcoin, y revisaremos el funcionamiento y las partes clave de la criptomoneda que permiten su funcionamiento.

3.1. Orígenes del Bitcoin

En los 80, cuando la criptografía comenzó a popularizarse, algunos desarrolladores intentaron crear monedas virtuales como alternativa a los medios convencionales de pago.

Existen varios intentos de criptomonedas previos al Bitcoin, como DigiCash desarrollado por David Chaum en 1983, o BitGold por Nick Szabo en 1998. Aun así, estas monedas no poseían todas las propiedades críticas de una criptomoneda. En concreto, Digicash no era un sistema descentralizado, y bitGold fue creado como una reserva de oro alternativa y no como un sistema de pago.

Por lo tanto, la llegada de un sistema de pago anónimo, descentralizado, y robusto consolidó a Bitcoin como la primera criptomoneda.

Para ver exactamente que ventajas supone el uso de Bitcoin, primero tendremos que realizar un breve análisis de los sistemas monetarios que se han empleado y realizar una comparación con ellos.

A lo largo de la historia existen instancias de usos de monedas alternativas como cuentas

de vidrio o conchas marinas. Tanto las conchas marinas, usadas en América y África, como las cuentas de vidrio africanas, obtenían su valor debido a la reducida cantidad y dificultad de obtención de las mismas. Eventualmente, la mejora en la maquinaria permitió la obtención de conchas en masa, mientras que por otro lado, los europeos introdujeron una gran cantidad de cuentas de vidrio en África. En ambos casos, el crecimiento repentino de la moneda de cambio supuso la devaluación de la misma y finalmente la caída del sistema.

Por otro lado, el oro parece ser un sistema que ha aguantado el paso del tiempo, y que aunque la economía actual gire alrededor del dinero *fiat*, un sistema basado en la confianza en el gobierno, sigue siendo utilizado como una reserva monetaria.

Saifedean Ammous en su libro "*El patrón Bitcoin*" [3] realiza un análisis del Bitcoin y remarca las similitudes que este posee con el oro. Para entender esta comparación vamos a revisar que es el patrón oro exactamente.

El patrón oro nació en el siglo XIX. Este sistema garantizaba que un país solo podía tener una cantidad de divisa propia que equivaliera a la cantidad de gramos de oro que poseía dicho país. Por lo tanto evitaba problemas como la inflación y facilitaba el comercio internacional. ¿Pero, por qué el oro?

El oro posee la ventaja con respecto a los bienes mencionados anteriormente de que su obtención realmente es compleja y el peligro de incremento excesivo de las existencias es casi nulo. Además, el oro es un bien que puede dividirse sin perder su valor, y es duradero en el sentido de que no se mella. Todo esto hace que el oro sea el sistema monetario más extendido históricamente.

Todas estas características son también propias del Bitcoin: la creación de bitcoins nuevos está regulada por el "minado", un sistema que mantiene el ratio de generación de bitcoins estable; la moneda es divisible hasta la cienmillonésima unidad; y al ser digital la moneda no se puede "romper".

Pero si existe el oro, ¿por qué crear otra moneda con propiedades tan similares? ¿Y qué ofrece Bitcoin que no ofrezca el dinero actual?

Aunque la obtención de oro es difícil, siempre es posible aumentarla. Es decir, si el precio del oro aumentara por cualquier razón, es posible que varias empresas optaran por extraer más oro, y acabaríamos en una situación similar que con las conchas marinas. Por otro lado, la oferta monetaria en Bitcoin es finita, y su número no sobrepasará los 21 millones, que previsiblemente se alcanzarán en 2140.

Mientras que el Bitcoin no ofrece ningún problema a la hora de utilizar partes de bitcoins, el oro no es fácilmente divisible, por lo tanto no es práctico realizar pagos menores con él.

Por último existe el problema de la portabilidad y almacenamiento, que el Bitcoin simplemente no tiene al ser completamente virtual.

Por otro lado, el dinero *fiat* no posee ninguno de los problemas mencionados, pero también presenta algunos inconvenientes.

Para empezar, este sistema está controlado por una autoridad, esto quiere decir que el

dinero no tiene ningún respaldo económico y es posible imprimir más dinero. Esto puede acabar con situaciones de hiperinflación como las que se han visto en Zimbabue o Bolivia. Bitcoin es un sistema descentralizado, es decir, que no depende en la confianza de una autoridad del sistema. Además, al ser una plataforma internacional los ciudadanos de países con monedas menos estables pueden utilizarlo como moneda alternativa en épocas de inestabilidad económica.

Otro problema que presenta es la posibilidad de falsificación de las monedas. Aunque se toman varias medidas como marcas de agua, papeles especiales, y máquinas antifalsificaciones para evitarlo sigue siendo una moneda de "papel", y por lo tanto la impresión de billetes no fidedignos sigue siendo un riesgo. Por otro lado, la falsificación de bitcoins es imposible debido a la tecnología *blockchain* que utiliza. Una *blockchain* es una especie de libro de cuentas en el que se anotan todas las transacciones que se realizan en Bitcoin. La razón por la que falsificarlo resulta imposible es porque cada usuario dispone de una réplica de la *blockchain* propia que evita que alguien pueda "anotar" una transacción falsa.

Por último vamos a comentar la anonimidad de la que se dispone en Bitcoin a parte, debido a que ofrece tanto ventajas como inconvenientes. Como ventaja tenemos la posibilidad de realizar transacciones sin rastreos de un tercero y que impide la congelación de cuentas, pero se incita el comercio ilegal y blanqueo de dinero.

Aunque es improbable que el sistema monetario cambie a uno basado enteramente en Bitcoin, es posible que su papel como pago alternativo o reserva de valor se popularice bastante en los próximos años [3].

Para entender mejor el funcionamiento de la criptodivisa Bitcoin y su situación frente a otros medios de pago puede consultarse el análisis DAFO de Sánchez de Diego [4] u otros más actuales como [5]

A continuación, vamos a realizar un estudio en más profundidad sobre las características de la tecnología *blockchain* que hacen de Bitcoin un sistema fiable.

3.2. Blockchain

El concepto original de cadena de bloques o *blockchain* fue concebido por David Chaum en 1979. Y como bien él mismo indica en su libro, fue creado como respuesta al "desarrollo de sistemas informáticos en los que puedan confiar grupos que no necesariamente confían entre sí" [6]. Esta definición se adecúa al sistema de intercambio de monedas, pero también podrían venir a la mente otras aplicaciones menos populares, como trazabilidad de productos o contratos inteligentes. La tecnología *blockchain* presenta las siguientes características [7, 8, 9, 10]:

- Distribuída: Se trata de una red P2P, por lo que no existe la presencia de una entidad central en la que confiar. Esto permite que en caso de ataque no haya un punto central, y aunque parte de la red cayera, la red seguiría sobreviviendo a partir de sus otros nodos. No solo es una red distribuida en el sentido de quien la maneja,

si no que para poder verificar transacciones, todos los usuarios tienen que estar de acuerdo.

- *Trustless*: en el sentido de que la confianza no es necesaria. A diferencia del dinero *fiat* que se basa en la confianza entre países, las criptomonedas permiten que los usuarios intercambien monedas de manera segura aunque no confíen entre sí. Esto se debe a que no utilizar procesos de seguridad ya incorporados en el sistema no es una opción que el usuario pueda tomar.
- Trazable: los participantes han de poder identificar la procedencia de los bienes y sus dueños.
- Inmutable: una vez una transacción haya sido guardada en la *blockchain* es imposible modificarla de ninguna manera. Si se ha cometido algún error en la transacción, se puede crear otra para revertirlo.
- Pública: la información de los bloques en la *blockchain* es visible para cualquiera. Existen algunos tipos de *blockchain* denominadas privadas que pueden ser más restrictivas.
- Privada: la identidad real del usuario se mantendrá desconocida e independiente de la identidad en la *blockchain*. Un usuario puede poseer más de una identidad en una misma red.

A modo de resumen podríamos indicar que una *blockchain* es un sistema de registro compartido, inmutable, y distribuido [11].

Como indica su nombre, una blockchain es una "cadena de bloques". Estos bloques se identifican por medio de un *hash* propio y son creados a un ritmo medio de 10 minutos en Bitcoin[2, 3]. Son los responsables de guardar la información referente a las transacciones.

En caso de que estuviéramos ante una *blockchain* de 3 bloques (véase Figura 1.1). El último bloque tendría como *hash* anterior al bloque nº2, y éste a su vez, el del bloque origen (el bloque origen es una excepción a esta regla ya que su *hash* anterior está vacío). Esta es la característica por la que este sistema es denominado como cadena de bloques.

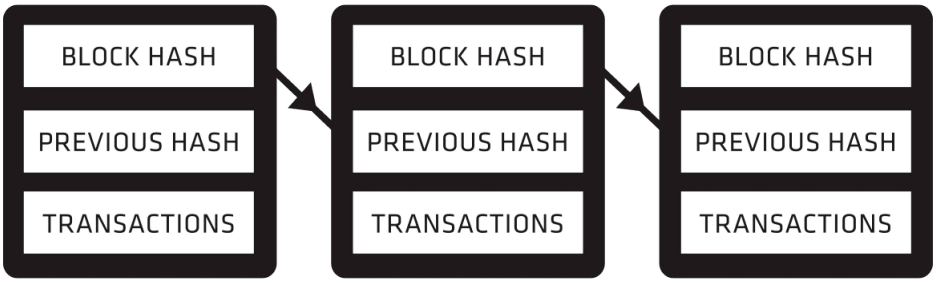


Figura 3.1: Ejemplo de *blockchain*.

Fuente: [12]

En Bitcoin cada transacción suele rondar los 250 bytes y el tamaño de un bloque promedia 1 MB, del los cual 80 bytes siempre componen una cabecera que almacena metadatos acerca del bloque y el resto contienen la información de las transacciones. La información contenida en la cabecera puede observarse en la tabla 1.1 [2, 13].

Tamaño	**Descripción**
4 bytes	Número que indica la versión de protocolos
32 bytes	Una referencia al *hash* del bloque anterior
32 bytes	*Hash* que resume las transacciones que contiene
4 bytes	Tiempo de creación aproximado de este bloque
4 bytes	El baremo de dificultad del algoritmo *proof of work*
4 bytes	Contador utilizado en *proof of work*

Tabla 3.1: Contenido de la cabecera de un bloque

Es posible que resulte extraño que habiendo indicado que los bloques se identifican por un *hash*, la cabecera no incluya dicho código. Esto es debido a que el *hash* resulta de aplicar la función SHA256 a la cabecera del bloque y por lo tanto es calculable en cualquier momento.

SHA256 es una función que forma parte de las funciones hash SHA-2 (SHA-224, SHA-256, SHA-384, SHA-512). Una función *hash* es un algoritmo que genera un código único (de 256 bits en este caso) a partir de unos datos de tamaño variable. Se utilizan en criptografía debido a que resulta- imposible obtener los datos a partir del código generado.

Un problema básico que afectará a la confianza en la red de bloques sería la falsificación de uno de estos bloques. Pero a pesar de que se describa como una red distribuida, el término "clonada" sería más apropiado.

La red *blockchain* no está repartida entre varios nodos de manera que una información específica se encuentre en un nodo concreto, sino que toda la información de la red se

encuentra en todos y cada uno de los nodos. Por eso, la modificación o falsificación de bloques es muy compleja, aunque todo sea dicho, no imposible.

En el caso de que una persona o grupo controle el 51 % del *hash rate* de la blockchain, lo cual se traduce en el 51 % de los votos para validar nuevos bloques, sería posible invalidar bloques validados dando lugar a un ataque del 51 %.

Para comprender mejor el protocolo, vamos a explorar lo que a efectos prácticos es el núcleo de Bitcoin, las transacciones y el minado.

3.3. Transacciones y minado

Las transacciones son posiblemente la parte más importante de cualquier criptodivisa, pues es ahí donde el intercambio de la moneda toma lugar. Una transacción estándar en Bitcoin tiene los siguientes elementos:

Un *hash* identificador de la transacción; una clave origen y una destino identificando tanto al usuario que envía dinero como al que lo recibe; una cantidad de bitcoins que se enviarán; y una tarifa que será substraída de la cantidad que relacionaremos más adelante con el proceso de minado.

Como se puede ver en la Figura 1.2 [14], la tarifa de 0.2$ más la cantidad recibida de 74.64$ equivale a la cantidad introducida por el emisario de la transacción.

Figura 3.2: Transacción de bitcoins

Fuente: Blockchain.com

Debido a la naturaleza pública de la blockchain, la mayoría de información que aparece en las transacciones está encriptada. Por ejemplo, el *hash* identificador de la propia transacción consiste en una cadena de bits formada por información de la misma a la que se le aplica dos veces la función *hash* SHA256.

Una vez se crea una transacción, esta se introduce en una especie de bolsa de transacciones donde esperan a ser verificadas.

El proceso de verificación y creación de bloques para la *blockchain* consta de los siguientes pasos [2, 15]:

1. Las nuevas transacciones creadas se envían a todos los nodos
2. Los nodos que reciben transacciones verifican que estas sean correctas (las condiciones de validación pueden verse en detalle en las funciones *AcceptToMemoryPool*, *CheckTransaction*, y *CheckInputs* en Bitcoin Core [2]) y las añaden a su bloque candidato, el bloque candidato es único para cada nodo y se convertirá en el bloque válido si ese nodo consigue "minar el bloque".
3. Cuando un nodo consigue "minar el bloque" este retransmite su bloque candidato como bloque válido al resto de nodos para que lo sepan. Cuando un nodo recibe un bloque, comparará las transacciones incluidas en el bloque con las de su candidato y eliminará las que coincidan, ya que ya se han validado.

La primera transacción del bloque se caracteriza por ser la que indica la recompensa de minar el bloque, esta recompensa se otorga a aquellos que consiguen minar un bloque debido a que el proceso de minado supone un gasto para el usuario. Está constituida por una cierta cantidad de bitcoins base (12.5 bitcoins por bloque a día de hoy) más la suma de las tasas recogidas de las transacciones que se verifican con el bloque.

El proceso de minado en Bitcoin se hace por medio del algoritmo *proof of work*, un sistema de autentificación en el que el interesado prueba la autenticidad del bloque indicando que cierta cantidad de capacidad computacional ha sido utilizada. De forma práctica se elige un *target* (actualmente 0x00000000000000000a40c000000000000000000000000000000000000000000000 en el bloque 728336 [16]) y los mineros intentan por medio de la función *hash* SHA256 generar un número menor que el *target* (por cada *hash* generado se tiene una probabilidad de $8.48e^{-22}$ %), el minero que lo consiga tiene derecho a convertir su bloque candidato en bloque válido.

4. Los nodos que recibe un bloque comprueban que las transacciones que incluye son correctas

5. Por último los nodos añaden el bloque que han recibido a la *blockchain* que tienen guardada y los mineros se preparan para intentar minar el siguiente bloque.

   Los nodos están compuestos por 3 tipos de bloques [2]:

   - Los correspondientes a la *blockchain* principal
   - Los correspondientes a *blockchains* secundarias, que se generan debido a que un bloque se mina casi al mismo tiempo por dos mineros diferentes. Se considera la *blockchain* principal como la cadena de bloques con mayor número de bloques almacenados, por lo que si un nuevo bloque se añade a una *blockchain* secundaria de misma longitud que la principal, esta pasaría a ser la principal, y la principal una secundaria.
   - Bloques huérfanos, bloques cuyo padre o *hash* anterior no se detecta en ninguna *blockchain* guardada. Se almacenan y se espera a recibir un bloque que conecte al huérfano con una de las cadenas.

Por último vamos a comentar la naturaleza de las carteras y las llaves, las herramientas principales que permiten realizar transacciones.

3.4. Llaves y *wallets*

Una *wallet* o cartera es un servicio que almacena las claves públicas y privadas del usuario, normalmente en formato virtual. Estas claves pueden ser usadas para acceder a la blockchain y consultar la cantidad de monedas asociadas. La generación de estas claves es la siguiente:

Una clave (o llave) privada es un número (c) entre 1 y 2^{256} que se calcula eligiendo una serie de bits aleatorios y aplicándolos el algoritmo SHA256.

A partir de esta llave privada, podemos adquirir su llave pública por medio de multiplicación elíptica curva [2, 17]. Para realizar esta operación se multiplica la llave privada (c) por el punto generador (G), que siempre es el mismo, y se obtiene una tupla (x,y) que es lo que finalmente compone la llave pública (C).

Si ahora a la clave pública se le somete a SHA256 y RIPEMD160 que son dos funciones *hash* se obtiene finalmente la dirección (A), una serie de 160 bits que representa al usuario en las transacciones. Pero las direcciones no vienen expresadas en cadenas de 20 bytes, para aumentar su legibilidad y evitar errores, a la dirección se le añaden 4 bytes de *checksum* al final de la dirección y un byte de prefijo indicando la versión, finalmente se le aplica una base 58 a estos 25 bytes y quedan los 34 caracteres que estamos acostumbrados a ver [18].

Tanto la multiplicación como las funciones *hash*, son no invertibles y por lo tanto la obtención de las claves públicas y privadas a partir de la dirección es imposible.

Tanto la clave pública como la privada no son fácilmente accesibles, por lo que para poder utilizar la misma cartera en dos dispositivos distintos ha de existir una contraseña que pueda desbloquearla.

Esta contraseña normalmente se trata de una frase formada de palabras que hace que sea más fácil de recordar. Aplicándole una función *hash* se consigue la clave privada asociada y por lo tanto la cartera es recuperable. Esta frase normalmente está compuesta de 12 o 24 palabras y a efectos prácticos su posesión implica la posesión de la cartera.

Hemos visto que Bitcoin es una red robusta y distribuida, con cantidad de consensos y criptografía que reafirma su seguridad, a pesar de ello, tampoco está a salvo de posibles ataques y obviamente también tiene algún punto débil, como los 10 minutos hasta validación de transacciones.

Los 10 minutos que eligió Nakamoto no solo hacen que las transacciones pequeñas parezcan menos atractivas, sino que bajo ciertas circunstancias puede hacer que la red sea más vulnerable a los ataques del 51 %. Por otro lado, menos tiempo haría que la red estuviera más cargada por la transmisión de bloques. Pero sea mejor o peor a nadie le gusta esperar 10 minutos para pagar un café.

En el siguiente capítulo exploraremos más a fondo este problema y se estudiará como el uso de *Lightning Networks* puede aliviarlo.

Capítulo 4

Lightning Network

El concepto original de Lightning Network fue diseñado en 2015 por Joseph Poon y Thaddeus Dryja. Su intención principal era resolver el problema de la escalabilidad en las transacciones de Bitcoin [19].

El problema de escalabilidad consiste en lo siguiente: a medida que el uso de Bitcoin aumente, la cantidad de transacciones aumentará hasta el punto en el que el número de transacciones por validar sea mayor que el ratio de validación de los bloques. Por lo tanto los usuarios tendrán que incluir tarifas más altas para intentar que su transacción se elija de entre las pendientes.

Se podría pensar que aumentar el tamaño de los bloques y seguir minando al mismo ritmo sería una manera apropiada de atacar este problema. Pero en realidad esto abre las puertas a otro problema.

Si se aumentara el tamaño de los bloques, la cantidad de capacidad computacional para procesar las transacciones de un bloque aumentaría, y como ya se ha indicado, todos los nodos de la *blockchain* tienen que procesar y validarlos. Esto se traduciría en un incremento en las especificaciones mínimas para poder participar en el proceso de minado y por lo tanto una inevitable centralización del proceso hacia unos pocos usuarios privilegiados.

Debido a que habría menos usuarios, el *hash rate* total se vería reducido y podría facilitar ataques a la moneda como el ya comentado ataque del 51 %.

Por lo tanto seguimos teniendo el problema de aumentar el ratio de transacciones que puedan procesarse pero intentando mantener la descentralización de la que la *blockchain* está tan orgullosa.

Pero todo sería más fácil si no todos los nodos de la red tuvieran que conocer y procesar todas las transacciones, ¿no?. Es ahí donde Joseph y Thaddeus propusieron la *Lightning Network* de Bitcoin basado en pagos escalables *off-chain*.

Para comprender mejor como funciona este sistema vamos a introducir el concepto de capas en bitcoin.

Capas de Bitcoin

Las capas en Bitcoin no son más que una manera de reunir protocolos según sus características, en concreto en Bitcoin existen 3 capas [20, 21].

1. La primera capa es la que hemos estudiado ya, en la que se incluye la *blockchain* de Bitcoin, donde se llevan a cabo transacciones, minería y procesos de validación.
2. La segunda capa es la que nos interesa, en ella existen protocolos que funcionan sobre Bitcoin y que añaden alguna funcionalidad. Los protocolos más extendidos de segunda capa de Bitcoin son Liquid Network, Omni Layer y Lightning Network.
   - Liquid Network es una cadena lateral diseñada para llevar a cabo intercambios de moneda rápidos. Una cadena lateral es una cadena que funciona junto a la cadena principal para aumentar eficiencia.
   - Omni Layer es una plataforma de creación e intercambio de monedas personalizadas.
   - Lightning Network es un protocolo que ofrece a los usuarios un sistema de microtransacciones rápidas *off-chain*. Comentaremos la definición y utilidad de intercambios *off-chain* más adelante.
3. Existe una tercera capa que es la que contiene las DApps o *descentralized apps*, pero Bitcoin no está diseñado para gestionar aplicaciones en esta capa y por lo tanto no se va a comentar.

Nuestro estudio está centrado en esta segunda capa, más en particular hacia la Lightning Network. Ésta constituye un protocolo, que aunque funciona sobre Bitcoin, presenta suficientes diferencias como para poder dedicarle un capítulo entero. En este capítulo vamos a ver un breve resumen de las características principales, algunos de los protocolos que se usan, y acabaremos uniendo conceptos con un ejemplo.

4.1. Características principales

Hemos comentado que la Lightning Network evita la obstrucción de transacciones en la *blockchain*, pero su funcionalidad no queda reducida simplemente a eso. Algunas de las ventajas adicionales que presenta son las siguientes[2, 22]:

- Los pagos *off-chain* que permite no necesitan confirmación de la *blockchain* y las tasas asociadas son menores.
- Al igual que en Bitcoin, los pagos son definitivos y solo son reembolsables por el receptor.
- A diferencia de Bitcoin, donde las transacciones pasan por todos los nodos, los pagos en la Lightning Network se transmiten entre pares, lo cual se traduce en mayor privacidad. Veremos que se entiende por transmisión entre pares en la explicación de los canales.

- Las transacciones no se almacenan permanentemente por lo que serán necesarios menos recursos de memoria.
- Utiliza encadenamiento de cebolla similar al protocolo Tor, evitando que nodos que no estén conectados directamente conozcan de la existencia del otro.
- La unidad mínima en la que se puede dividir el bitcoin se llama satoshi y 100 millones de satoshis equivalen a 1 bitcoin. A 30 de marzo de 2022 el satoshi tiene un precio de 0.0004741$ y debería ser suficiente como para poder representar cualquier suma de dinero. Aun así, la Lightning Network permite manejar cantidades menores a un satoshi.

Algunos de estos conceptos pueden resultar extraños debido a que todavía no se han comentado. Antes de empezar a explicarlos y para poder hacerlo correctamente vamos realizar un breve resumen de los elementos que componen una red Lightning.

4.2. Componentes de la Lightning Network

Al igual que la red Bitcoin, la red Lightning está compuesta por usuarios que desean intercambiar fondos, y nodos, que se encargan de administrar el proceso. Además, las redes Lightning disponen de canales, que permiten la comunicación entre pares que hemos comentado. En este apartado vamos a comentar la estructura y funcionamiento de los nodos y de los canales. En la red Lightning las carteras engloban al funcionamiento de los nodos, por lo que se explicarán desde el punto de vista general de las carteras.

4.2.1. Carteras

Las carteras en la red Lightning reúnen varias funcionalidades necesarias [22], algunas de las cuales son:

- Un llavero propio y secreto necesario para poder llevar a cabo la transacción.
- Un nodo de la red. Los nodos necesitan poder realizar 3 funciones: recibir y realizar pagos, comunicarse por medio de P2P con otros nodos, y acceder a la *blockchain* de Bitcoin para conseguir fondos.
- Un nodo Bitcoin que almacene información y pueda comunicarse con otros nodos de la red.
- Una base de datos con los nodos y los canales abiertos en la red Lightning.
- Un *manager* que pueda abrir canales.
- Un sistema que pueda encontrar una serie de canales que relacionen el origen de un pago con su destino.

Las carteras de la red pueden poseer todas estas funciones y considerarse carteras "enteras", o pueden apoyarse en terceros para gestionar alguna.

Las carteras pueden clasificarse según la privacidad que ofrecen y el nivel técnico que necesitan para utilizarse:

- Las carteras "enteras" que hemos comentado controlan sus llaves al igual que sus nodos Lightning y Bitcoin. Constituyen las carteras que menos permisos otorgan a terceros pero también son las que requieren de más conocimientos técnicos para poder utilizarse.
- Las carteras que utilizan algún servicio de terceros varían su nivel de privacidad dependiendo del servicio que contraten. Por ejemplo una cartera que maneje sus llaves pero no sus nodos puede considerarse poco técnica y relativamente privada.
- Por último están las carteras cuyo control de llaves y nodos recae en terceros, esto resulta en poca privacidad y poco nivel técnico necesario.

Al final es el usuario el qué debe juzgar que cartera utilizar atendiendo a sus necesidades. En el próximo capítulo extenderemos este tema aportando algunos nombres de las principales carteras.

4.2.2. Canales

Los canales en la red corresponden a una relación financiera entre dos nodos. Esta relación asigna un balance de fondos (en satoshis) entre los dos usuarios. Al crear uno, cada usuario dedica cierta cantidad de satoshis para que el canal pueda funcionar.

El proceso de creación y cierre de un canal requiere dos transacción que han de quedar registradas en la *blockchain*. Estas transacciones necesitan el permiso (otorgado por sus llaves) de los dos usuarios del canal dando lugar a una transacción multifirma.

Un protocolo criptográfico dirige el canal y es responsable de redistribuir los fondos según sea necesario. Las transacciones que se crean en un canal informan de la distribución de los fondos entre los usuarios, si es necesario redistribuirlos, se crea una nueva transacción indicando el nuevo reparto. Todas estas transacciones son invisibles para Bitcoin y es solo cuando el canal se cierra cuando la última transacción se acaba registrando.

Algunas limitaciones que pueden sufrir estos canales vienen dadas por su velocidad de internet o por el propio balance del canal, que no permite intercambios de cantidades mayores a las que maneja.

Estos canales se pueden utilizar en sucesión para intercambiar dinero entre usuarios que no disponen de un canal directo en la Lightning Network. La explicación escrita puede resultar confusa así que volveremos a estos canales cuando se estudien los contratos *hash time locked*.

4.3. Enrutamiento de Cebolla

El enrutado de cebolla, popularizado por el navegador Tor, es un método de comunicación encriptada, haciendo referencia al protocolo de encriptación que utiliza. En él, el emisor construye sobre el mensaje varias capas de encriptadas que los nodos intermediarios van "pelando" hasta que el núcleo del mensaje llega al nodo receptor. De esta manera se consigue que la única información que reciben los nodos intermedios sea el nodo del que han recibido el mensaje y a quién deben enviárselo.

La Lightning Network utiliza una implementación basada en el protocolo Sphinx. Aunque en esta sección vamos a centrarnos únicamente en su aplicación, se puede obtener una explicación más detallada de su funcionamiento acudiendo al artículo publicado por sus creadores G. Danezis e I. Goldberg [23].

Para visualizar más fácilmente el proceso de construcción y funcionamiento de este tipo de encriptación Andreas M. Antanopoulos [22] propone un ejemplo basado en cartas similar al propuesto a continuación.

Supongamos que Pablo quiere enviar un mensaje a Lucía y decide hacerlo por un camino que utiliza los canales mostrados en la figura 2.1.

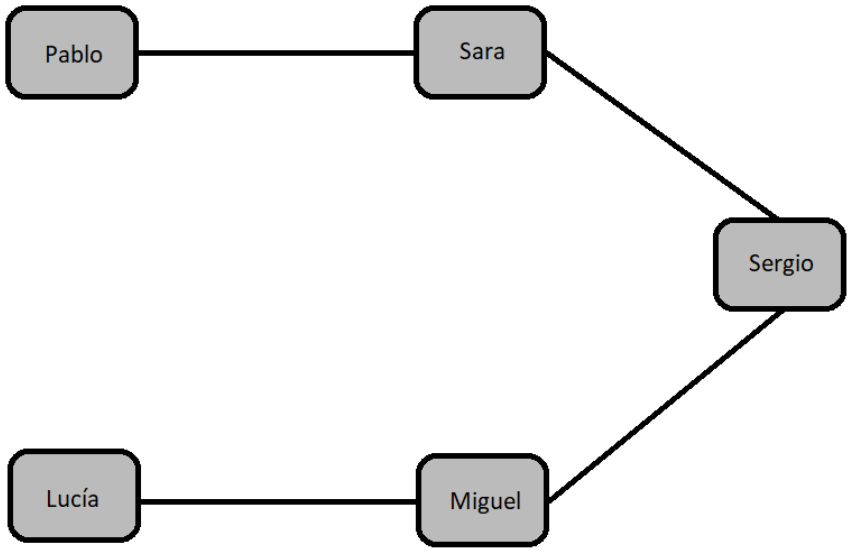


Figura 4.1: Camino de Pablo a Lucía.

Pablo conoce el camino, y los nodos intermedios por los que va a pasar el mensaje, pero quiere que la información solo llegue a Lucía, por lo que va a "envolver" el mensaje en un sobre (figura 2.2).

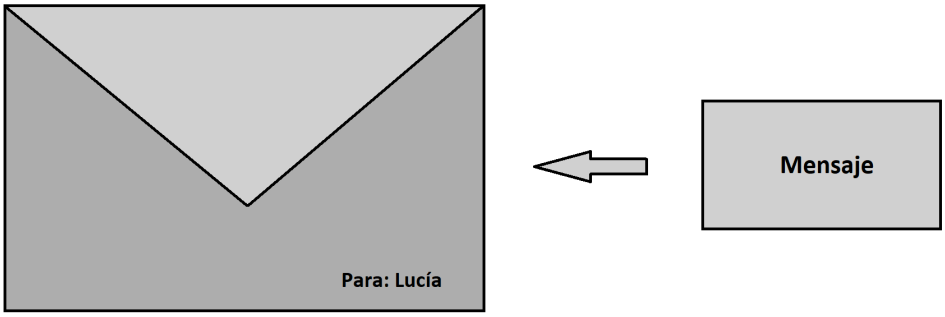


Figura 4.2: Carta sellada para Lucía.

Además de esto Pablo tampoco quiere que nadie sepa a quién va dirigido el mensaje, por lo que va a sellar esta carta dentro de otra dirigida a Miguel, y esta dentro de otra a Sergio, y finalmente dentro de otra que finalmente va a mandar a Sara. La forma final del mensaje se puede ver en la figura 2.3.

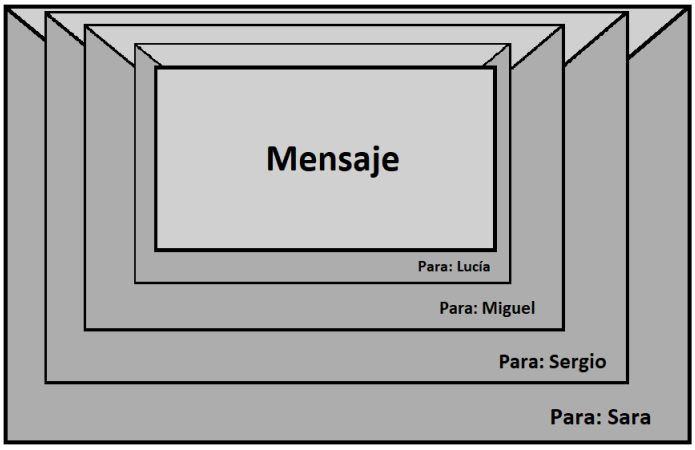


Figura 4.3: Carta enviada a Sara.

De esta manera una vez los intermediarios abren su sobre solo saben que tienen que mandar el mensaje a otra persona, y ni siquiera Miguel puede saber que Lucía es la receptora final.

A continuación vamos a ver qué es un contrato *hash time-locked* y cómo se aplican a los intercambios en la Lightning Network, comentaremos brevemente el proceso de búsqueda que realizan los nodos emisarios para elegir un camino, y especificaremos los pasos que se llevan a cabo para realizar una transacción.

4.4. Contratos Hash Time-Locked

Los contratos *hash time-locked* o HTLC (Hash Time-Locked Contracts) se utilizan en la Lightning Network como forma de pago entre dos nodos [22].

Los contratos que hemos comentado se usan para balancear los canales y son los que crean las transacciones. Las transacciones están compuestas de una entrada que representa el id del canal con la cantidad total del canal, y de 2 salidas que corresponden a los fondos individuales de cada usuario. Los HTLC forman parte de este pago como otra salida temporal del contrato.

Los HTLC son pagos creados por y para el mismo canal y solo ejecutables si se conoce cierta contraseña. Por ejemplo, Pablo y Sara comparten un canal y tienen 2 BTC cada uno, Pablo crea un HTLC de 1 BTC, lo que hace que el canal ahora esté compuesto por 2 BTC de Sara, 1 BTC de Pablo, y 1 BTC del HTLC. Esto tiene 2 desenlaces:

- Si Sara consigue la contraseña puede ejecutar el HTLC y reclamar el bitcoin asociado.
- Si por otro lado, no consigue la contraseña antes de cierto tiempo, Pablo creará otro contrato (que ambos usuarios necesitarán firmar) que actualizará su balance a los 2 BTC originales.

Existe el caso en el que el canal se cierre con un HTLC iniciado. En caso de que el cierre sea consensual, el HTLC se resolverá primero antes del cierre. Si el cierre es unilateral los HTLC quedan en el denominado "balance limbo", del que después de cierto tiempo son devueltos a su dueño.

Vamos a utilizar el ejemplo de la figura 2.1 para ver como funcionan estos HTLC con varios nodos. Por simplicidad vamos considerar que todos los canales están compuestos por un saldo total de 4 BTC repartidos 2 a 2, y que Pablo quiere enviar 1 BTC a Lucía. El nuevo ejemplo viene representado en la figura 2.4.

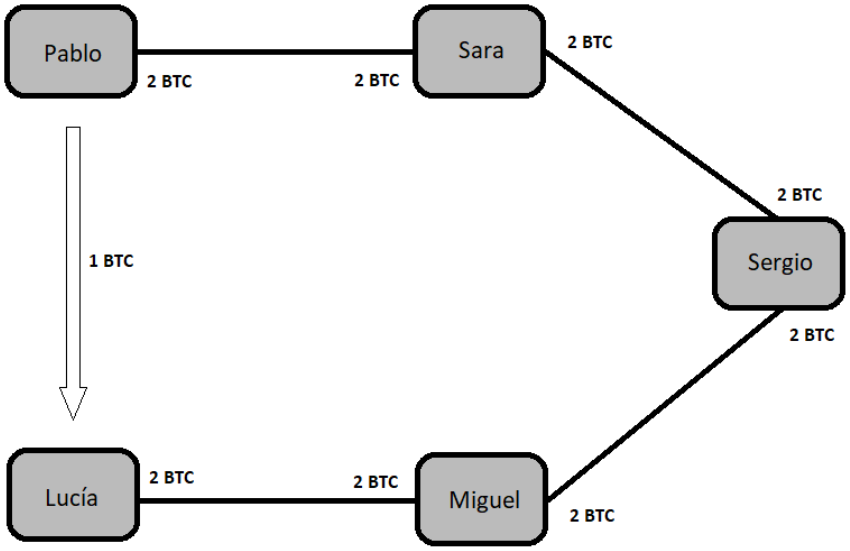


Figura 4.4: Camino de Pablo a Lucía para pago de 1 BTC.

Lo primero que hará Pablo es pedir a Lucía un *hash* H, Lucía generará una contraseña aleatoria R y enviará a Pablo el resultado de realizar H=SHA256(R) como el *hash* que ha pedido. Tenemos que recordar que sacar R a partir de H es prácticamente imposible.

Pablo ya sabe el camino que tiene que tiene que realizar y sabe que cada intermediario impone 0.1 BTC como tasa, el proceso de elegir un camino lo comentaremos más adelante. Así que prepara un HTLC con 1.3 BTC y la dice a Sara: "Si consigues la contraseña R que responde a H, yo, Pablo, me comprometo a pagarte 1.3 BTC, y como muestra de esto, tienes un HTLC que puedes ejecutar si conoces R".

Ahora Sara creará otro HTLC pero esta vez con Sergio, y le dirá algo muy similar a lo que le ha dicho Pablo, pero esta vez el HTLC será de 1.2 BTC. La figura 2.5 representa el proceso hasta ahora.

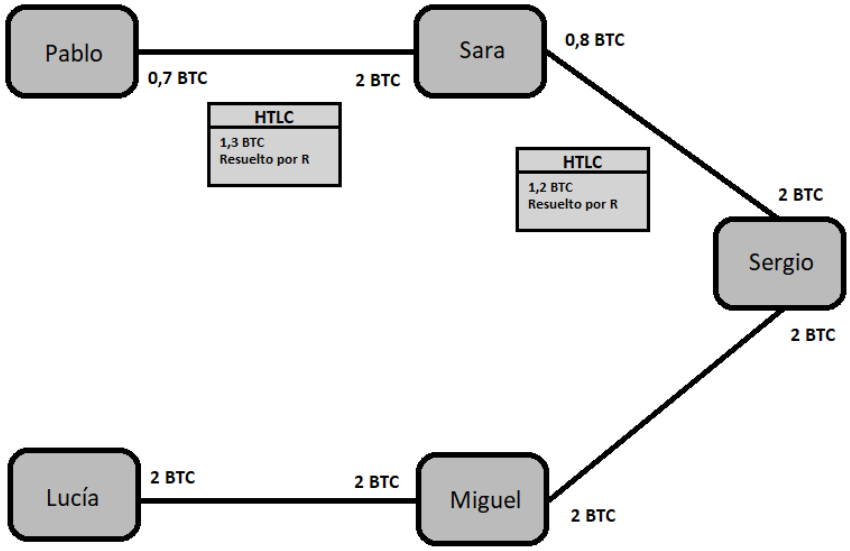


Figura 4.5: 1ª parte de la transacción entre Pablo y Lucía.

Sergio ahora creará otro HTLC con Miguel con valor de 1.1 BTC y finalmente Miguel creará otro con Lucía de 1 BTC (figura 2.6).

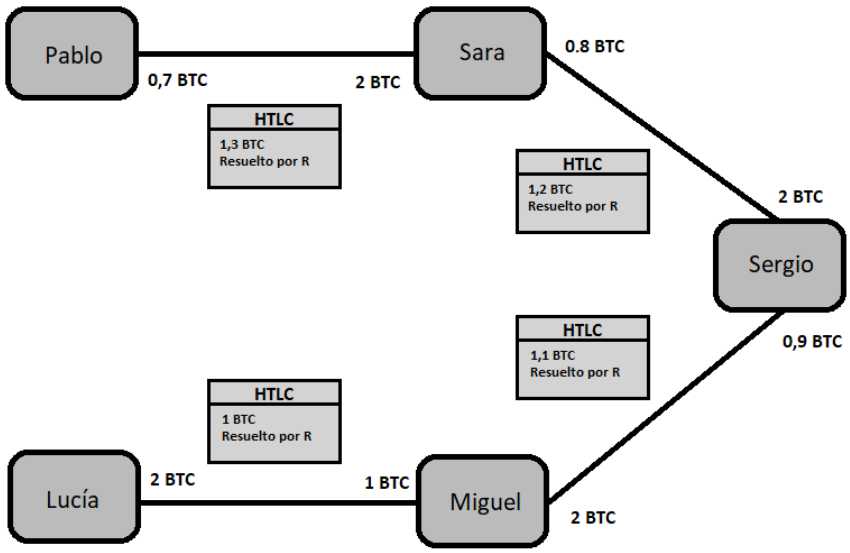


Figura 4.6: 2ª parte de la transacción entre Pablo y Lucía.

Lucía sabe R, por lo que puede reclamar el bitcoin del HTLC y finalmente recibir su pago, una vez lo ejecute, Miguel sabrá R, y podrá ejecutar el HTLC con Sergio, Sergio el que tiene con Sara, y finalmente Sara el que tiene con Pablo. El resultado final de los balances de la transacción puede visualizarse en la figura 2.7.

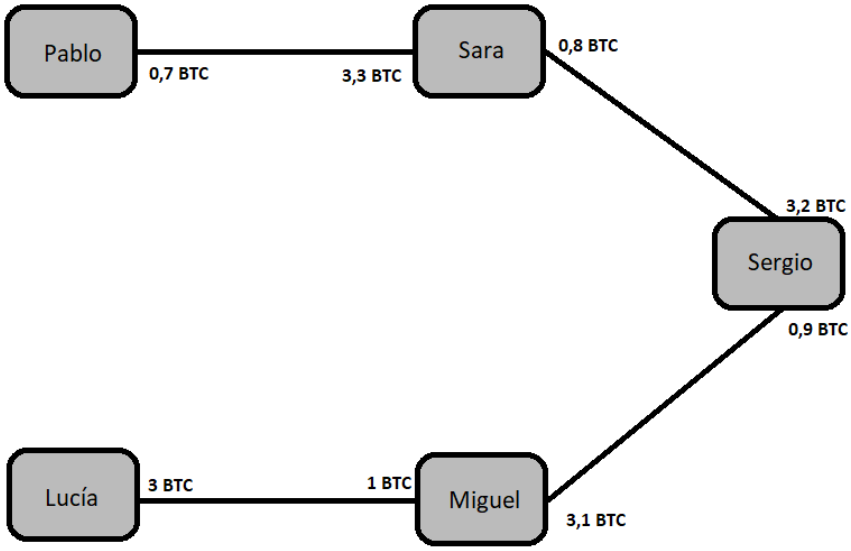


Figura 4.7: Balance final de la transacción entre Pablo y Lucía.

Hay que explicar algunos conceptos todavía difusos como el porqué Pablo sabe las tasas de los intermediarios, y como saben los intermediarios a quien hablar después de recibir un mensaje. Antes de adentrarnos en esto vamos a repasar el proceso de elección de camino que ha realizado Pablo.

4.4.1. Elección de camino y protocolo Gossip

Para conseguir que todos los nodos conozcan de la existencia de un canal creado recientemente se utiliza el protocolo Gossip, con él los canales pueden anunciar su existencia al resto y comunicar información como las tasas, el balance total o el tiempo máximo de expiración de HTLC. Para una explicación más detallada se recomienda dirigirse consultar el artículo de B. Teinturier y N. Saitug [24].

Existen ciertos criterios que tener en cuenta cuando se considera la elección de canales intermedios [22]:

- El camino necesita tener liquidez. No podemos enviar 1 BTC por un nodo que tenga 0.5 BTC aunque el canal que use sume 4 BTC en total.
- Caminos con menos tasas. Obviamente intentaremos elegir caminos que requieran de menos tasas.

Con ayuda del protocolo Gossip cada nodo tiene un mapa de la distribución y conexión de los nodos que forman la Lightning Network. Esto transforma el problema de elección de camino a un problema de estudio de grafos dirigidos etiquetados.

Por desgracia el protocolo Gossip solo informa del balance general del canal y no de la liquidez de cada nodo en el mismo, por lo tanto el emisor no sabe realmente los fondos

de los que dispone un nodo por el que pueda pasar el pago. El proceso de preguntar individualmente a cada nodo sería muy tedioso, y finalmente se adoptó la práctica de que el nodo crea una lista de caminos posibles que se pueden utilizar y los prueba hasta que uno funciona.

Aunque este proceso de selección de camino puede parecer poco óptimo, es un método que funciona relativamente bien para una red bien conectada y con pagos reducidos.

Se ha comentado el enrutamiento de cebolla pero no se ha explicado todavía su aplicación a los HTLC. Para finalizar el capítulo vamos a revisar la encriptación que se utiliza con los HTLC y terminar de comprender el método de pago en la Lightning Network.

4.4.2. Enrutamiento de cebolla en HTLC

Para estudiar la criptografía relacionada con los HTLC vamos a utilizar el ejemplo de la figura 2.1 con la que ya hemos trabajado.

La información que Pablo quiere enviar a Lucía está formado por los siguientes campos:

- *amt_to_forward*: la cantidad en milisatoshis que el emisor paga al receptor en este pago.
- *outgoing_cltv_value*: número de bloque en el que el pago expira.
- *payment_secret*: un secreto de 256 bits que permite a Lucía identificar el pago.
- *total_msat*: si este pago es único esto será igual al *amt_to_forward*, pero si es parte de un pago mayor este campo expresa la suma de todas los *amt_to_forward* de todos los subpagos.

Por otro lado la estructura de la información que los intermediarios Sara, Sergio y Miguel reciben es la siguiente:

- *short_channel_id*: un identificador del canal por el que tienen que retransmitir el próximo mensaje.
- *amt_to_forward*: cantidad de milisatoshis que tienen que enviar en el HTLC.
- *outgoing_cltv_value*: número de bloque en el que el pago expira.

La figura 2.8 contiene representaciones del contenido de lo que los participantes del pago reciben. La información que los intermediarios reciben contiene información del HTLC que tienen que crear con el siguiente canal. Por eso, tras recibir un HTLC de 1.3 BTC, Sara creará otro con Sergio por valor de 1.2 BTC en su canal compartido con *short_channel_id* 00000000000000.

Puede parecer raro que los contratos que se crean antes expiren más tarde, esto se debe al orden en el que se ejecutan. Para facilitar las cuentas supongamos que no hay bloques en la *blockchain*, y Pablo indica que va a permitir 15 bloques hasta que el contrato con

Lucía expire. Una vez Lucía ejecute el HTLC que tiene con Miguel (como muy tarde en el bloque 15), Miguel tendrá hasta el bloque 35 para ejecutar el HTLC con Sergio. Los 20 bloques son estándar pero los 15 bloques indicados se pueden modificar y en caso de no especificarse serán 18.

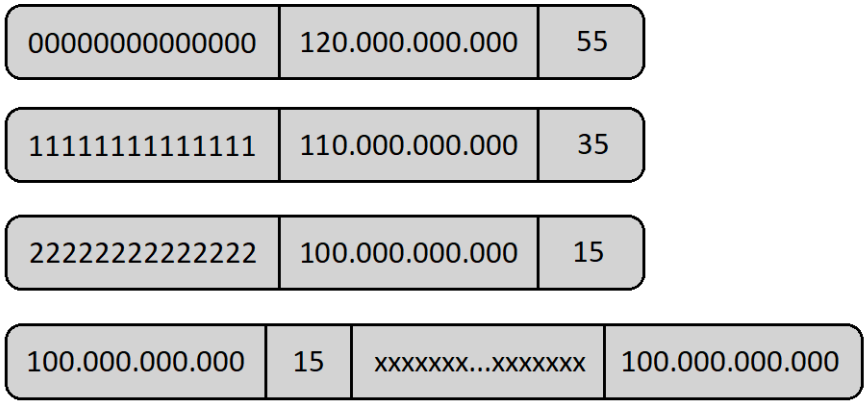


Figura 4.8: De arriba a abajo: mensajes de Sara, Sergio, Miguel y Lucía.

Para poder enviar toda esta información, Pablo va a tener que generar varias claves con las que encriptarla para que solo lo puedan leer sus respectivos destinatarios. Para ello se utiliza el protocolo Elliptic-curve Diffie–Hellman (ECDH) por el cual dos usuarios con llaves públicas generadas por curvas elípticas pueden crear un secreto compartido. Para evitar revelar su identidad Pablo utiliza una clave de sesión creada únicamente para esa transacción.

El primer mensaje que Pablo envía a Sara está compuesto de 1366 bytes divididos en:

- 1 byte para la versión
- 33 bytes para la llave de sesión comprimida
- 1300 bytes para la información.
- 32 bytes para un *checksum*

En realidad aunque los mensajes se vayan "pelando", todos los mensajes tienen la misma longitud. Esto se hace para que ningún nodo pueda suponer el destinatario basándose en la longitud del mensaje que tiene que enviar.

Para generar el mensaje que Sara va a recibir, Pablo tendrá que crear primero el de Lucía e ir envolviéndolo con los mensajes de los nodos intermedios.

El proceso de generación requiere de varios pasos en los que no se va a entrar en demasiado detalle, la figura 2.9 muestra el sistema de generación simplificado.

Figura 4.9: Obtención del mensaje que Pablo envía a Sara.

Con esto último damos por finalizado la teoría de la Lightning Network y podemos empezar a adentrarnos en la parte práctica. El siguiente capítulo realizará un estudio sobre los tipos y funcionalidades de nodos que puede haber en la Lightning Network y se explicarán los métodos de instalación particulares.

Capítulo 5

Instalación y uso de un nodo Lightning

La principal utilidad que podemos pensar cuando hablamos de la Lightning Network es la facilidad que proporciona para realizar microtransacciones de manera rápida, pero también se puede utilizar la red como modelo de negocio. Ya hemos comentado que aunque sean minúsculas, existen ciertas tasas que un usuario ha de pagar a los intermediarios para poder realizar transacciones, así que es posible poseer varios nodos con la finalidad de cobrar esas tasas y poder generar un beneficio.

Este capítulo tratará la instalación o adquisición de servicios que permiten la puesta en marcha de un nodo en la Lightning Network como puro intermediario en las transacciones de la red. Además se realizará un breve estudio de la rentabilidad que supone.

Antes de comenzar, explicaremos el concepto de Testnet, la red de pruebas de Bitcoin que utilizaremos para familiarizarnos con el método de uso. Aunque Testnet funcione igual que la red principal, es importante realizar una distinción entre ellas debido a que hay software que no da soporte a alguna.

5.1. Testnet

Los inicios de esta red de pruebas se remontan a 2010, aunque debido a ciertas dificultades, la red tuvo que reiniciarse en diferentes ocasiones. Así que en realidad la versión actual (Testnet3) se inició en 2012.

La moneda utilizada en Testnet es idéntica a la utilizada en la red principal (Mainnet), así que para evitar que se puedan enviar monedas de la Testnet a la Mainnet, las direcciones pertenecientes a Testnet empiezan todas por “2” o por “m”, mientras que las de Mainnet empiezan por “1” o por “3”. Con este cambio nos aseguramos de que cada moneda se mantiene en la red en la que se creó.

Hemos optado por utilizar Testnet para realizar las pruebas debido a que el método de uso es igual al de la Mainnet pero su moneda no tiene valor. Algunas de las ventajas e inconvenientes que conlleva el uso de la red se comentan más adelante.

5.2. Opción 1: instalación de nodo Lightning

Ya que es necesario mantener encendido el sistema en el que se ejecute el nodo de la Lightning Network, vamos a utilizar Linux (en concreto Ubuntu 20.0.4) como sistema operativo.

Como ya hemos explicado, la red Lightning es un servicio de segunda capa que funciona con cierta dependencia de Bitcoin. Debido a esto, vamos a dividir el proceso de puesta a punto de un nodo Lightning en dos partes, la instalación de la *blockchain* de Bitcoin, y la instalación del software Lightning.

5.2.1. Bitcoin

El software que utilizaremos para poder conectar con la *blockchain* es Bitcoin Core, el cual se puede descargar directamente desde la página oficial de Bitcoin [25], para este trabajo se ha utilizado la versión 22.0. Al descomprimir el archivo, tendremos una carpeta con 6 aplicaciones.

Podemos realizar la instalación de las aplicaciones con la siguiente línea de comandos:

```
sudo install -m 0755 -o root -g root -t /usr/local/bin
    bitcoin-22.0/bin/*
```

Las principales aplicaciones que utilizaremos son *bitcoin-cli*, *bitcoind*, y *bitcoin-qt*. *Bitcoind* sirve para iniciar el server de Bitcoin, *bitcoin-cli* permite comunicarse con el servidor creado por *bitcoind*, y *bitcoin-qt* proporciona un entorno gráfico para *bitcoind*.

Antes de realizar la descarga de la *blockchain* tenemos que crear un archivo de configuración de Bitcoin, bitcoin.conf, que ha de crearse en la carpeta donde se ejecuta Bitcoin (normalmente en ∼/.bitcoin). En concreto habrá que especificar que queremos descargar la *blockchain* de Testnet, que funcione como un servidor, que se ejecute como un daemon (ejecución en segundo plano), y finalmente habrá que escribir un usuario y contraseña para el servidor RPC, que es el que permite comunicación entre clientes. El contenido del fichero puede visualizarse en el Anexo I.

Con el fichero de configuración creado podemos iniciar la descarga de la *blockchain* con el siguiente comando:

```
bitcoind
```

Si en vez de esto utilizamos

```
bitcoin-qt
```

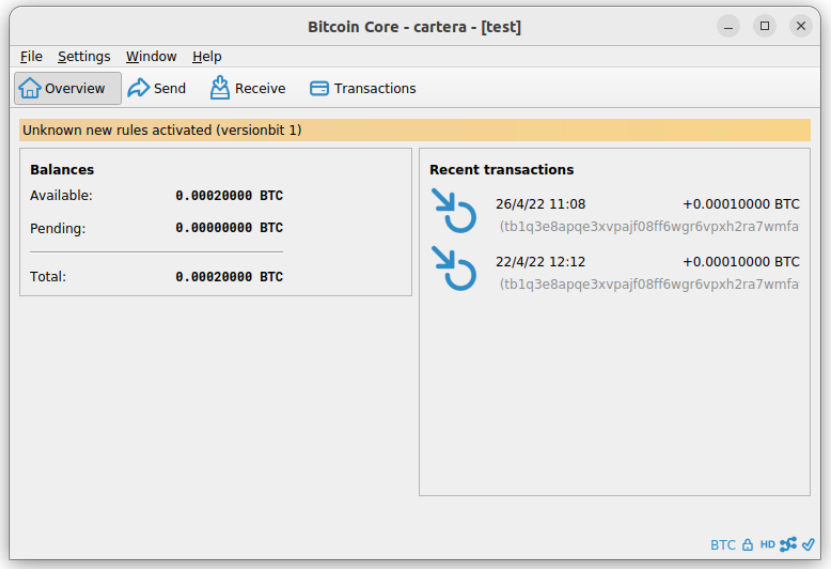


Figura 5.1: Bitcoin Core en la red Testnet

obtenemos una interfaz gráfica como la que presenta la figura 3.1 que informa del proceso de descarga y permite un fácil acceso a otras funcionalidades. En caso de que no se disponga de interfaz gráfica debido a que se esté utilizando una raspberry u otro dispositivo similar, podemos utilizar el comando

```
bitcoin-cli getblockchaininfo
```

como alternativa para recibir información de la *blockchain*.

La *blockchain* de la Mainnet ocupa alrededor de 400Gb mientras que la de Testnet ocupa algo menos de 40Gb. Esto hace que el proceso de descarga de Testnet sea mucho más rápido y se pueda llevar a cabo en uno o dos días.

Con la *blockchain* ya descargada y bitcoind ejecutándose ahora tenemos que realizar la instalación de la Lightning Network.

5.2.2. Lightning Network

El proceso de instalación de la Lightning Network es algo más tedioso. Lo primero que necesitamos es el *daemon*, que podemos descargar del github de *lightningnetwork* [26], de manera similar a Bitcoin Core, necesitaremos descomprimir el archivo e instalar su contenido con el comando:

```
sudo install -m 0755 -o root -g root -t /usr/local/bin
    lnd-linux-arm-v0.14.3-beta/*
```

Al igual que se hizo con Bitcoin Core, creamos un archivo lnd.conf en la carpeta de ejecución de Lightning Network (normalmente en ∼/*.lnd*). Tendremos que especificar una ip externa, que indica que el nodo acepta creaciones de canales, esto es importante debido a que queremos nodos muy conectados para que pasen por ellos la mayor cantidad de transacciones posible; podemos indicar un nivel de *debug* si queremos recibir mensajes de procesos y errores; tenemos que indicar que bitcoin tiene que estar activo, que utilizaremos la red Testnet y el *back-end* bitcoind; y por último, especificar la información de RPC que escribimos en bitcoin.conf. El contenido del fichero puede visualizarse en el Anexo II.

Habiendo instalado las aplicaciones descargadas, tenemos acceso a los comandos *lnd*, y *lncli*. *Lnd* es el *Lightning Network Daemon* y se encarga de gestionar los canales y enrutar, mientras que *lncli* se utiliza para comunicarse con los nodos *lnd*. Es posible ejecutar varios nodos lnd en un mismo dispositivo pero cada uno tiene que tener su directorio de datos y puerto.

Para describir los comandos que se van a utilizar en la instalación del servicio que va a soportar la red Lightning vamos a describir 2 versiones, una para el caso de un solo nodo, y otra para el caso de que haya varios nodos.

Iniciar el *daemon* de la red Lightning

Utilizando un nodo no necesitamos ningún parámetro y el comando de ejecución es

```
lnd
```

Si tenemos que referirnos a uno de entre varios nodos, tendremos que hacer que cada nodo tenga un puerto distinto en el apartado *externalip* de su lnd.conf. Además, habrá que especificar algunos parámetros al ejecutar el comando:

```
lnd --rpclisten=localhost:<puerto> --listen=localhost:<puerto>
    --datadir=<directorio>
```

- *rpclisten*: el puerto que utiliza el servidor RPC, necesita un puerto único.
- *listen*: el puerto que utiliza para conexiones P2P, necesita un puerto único.
- *datadir*: el archivo donde se va a ejecutar el *daemon* de lnd, cada usuario necesita uno propio.

Aunque se ha dicho que este capítulo trata la instalación de un nodo intermediario y no cubre los procesos de enviar y recibir transacciones, la presencia de una cartera es imperativa si se quiere que el nodo tenga canales, por lo tanto vamos a especificar el proceso de creación de una cartera.

Creación de una cartera Lightning

Para crear una cartera con un solo nodo podemos utilizar

```
lncli --network=testnet create
```

- *network*: indica el tipo de red a la que lnd se intenta conectar.

Al igual que con lnd tenemos que especificar algunos argumentos si queremos tener varios nodos, el comando para la creación de una cartera sería el siguiente:

```
lncli --network=testnet --rpcserver=localhost:<puerto>
    --datadir=<directorio> create
```

- *rpcserver*: el servidor RPC del lnd al que tiene que conectarse, el puerto tiene que ser el mismo que el puerto que indicamos en el argumento *rpclisten* del lnd.

Para poder utilizar la cartera el proceso *lnd* pide desbloquearla, lo cual se puede hacer ejecutando

```
lncli --network=testnet unlock
```

para un nodo, y

```
lncli --network=testnet --rpcserver=localhost:<puerto>
    --datadir=<directorio> unlock
```

para varios.

Todavía nos queda algún paso antes de poder dejar al nodo transmitiendo transacciones, en concreto tenemos que poder crear canales, y para ello necesitamos tener algo de capital.

Fundando una cartera Lightning

En realidad una cartera Lightning es igual que una cartera de Bitcoin y por lo tanto se pueden introducir fondos de la misma manera.

Lo primero que tenemos que hacer es generar una dirección para nuestro nodo lnd. Para un nodo el comando es:

```
lncli --network=testnet newaddress np2wkh
```

para muchos nodos tenemos que especificar el directorio y el servidor RPC:

```
lncli --network=testnet --rpcserver=localhost:<puerto>
    --datadir=<directorio> newaddress np2wkh
```

Como estamos utilizando la red Testnet tenemos acceso a un servicio llamado "grifo" (faucet) [27, 28, 29] con los que podemos mandar bitcoins a la cartera que queramos. Para realizar el envío necesitamos la dirección de la cartera Bitcoin (para obtenerla podemos utilizar el comando getinfo de lncli). Este servicio sólo está disponible para esta red debido a que la moneda no posee valor monetario.

Para comprobar el saldo de una cartera podemos utilizar:

```
lncli --network=testnet walletbalance

lncli --network=testnet --rpcserver=localhost:<puerto>
    --datadir=<directorio> getwalletbalance
```

para uno y varios nodos respectivamente.

Una vez comprobemos que el nodo dispone de fondos, podemos pasar a crear un canal con otro nodo.

Creación de un canal Lightning

Para crear un canal con un nodo necesitamos saber su llave pública, su ip externa, y su puerto. Por ejemplo, la página 1ML [30] tiene soporte de Testnet y ofrece su nodo para creación de canales. Para conectarnos con el utilizamos el siguiente comando para un nodo:

```
lncli  --network=testnet connect <Llave pública>@<externalip>:<puerto>
```

Con varios nodos y para el caso particular del nodo de 1ML habría que añadir lo siguiente:

```
lncli  --network=testnet --rpcserver=localhost:<puerto>
    --datadir=<directorio> connect 02312627fdf07fbdd7e5ddb136611
    bdde9b00d26821d14d94891395452f67af248@23.237.77.12:9735
```

Ahora que ya estamos conectados al nodo, tenemos que crear el canal e introducir fondos en el. Para ello tenemos que utilizar la llave pública del nodo.

```
lncli --network=testnet openchannel --node_key= <Llave pública>
    --local_amt=100000

lncli --network=testnet --rpcserver=localhost:<puerto>
    --datadir=<directorio> openchannel --node_key= <Llave pública>
    --local_amt=100000
```

- *local_amt*: la cantidad en satoshis que el creador va a introducir en el canal.

Una vez hecho esto si se quiere crear más canales para tener un nodo mejor conectado bastaría con repetir los últimos dos pasos.

Con esto damos por finalizado la instalación de un nodo Lightning, y pasamos a estudiar otra opción de modelo de negocio, en concreto la opción de alquilar un nodo.

5.3. Opción 2: alquiler de nodo Lightning

Es posible que un usuario no crea tener la suficiente familiaridad con la ejecución de comandos como para poder instalarlo, o simplemente no quiere pasar por todo el largo proceso de instalar un nodo, o a lo mejor no quiere dejar encendido un dispositivo constantemente. Sea por lo que fuere habrá usuarios que quiera una opción más fácil.

Como alternativa existe una plataforma que permiten alquilar sus nodos Lightning, Voltage [31], que por suerte soporta Testnet.

El nombre del producto que ofrecen se llama *Nodes*, fácilmente traducido a nodos. Este servicio permite al usuario un fácil acceso a un nodo sin necesidad de instalar software. Además, no es necesario mantener un dispositivo encendido ya que el nodo está soportado por un servicio externo.

5.3.1. Creación de un nodo con Voltage

El proceso de creación de un nodo es bastante simple, tras haber creado una cuenta e iniciar sesión. La plataforma nos presenta el menú principal, que puede visualizarse en la figura 3.2.

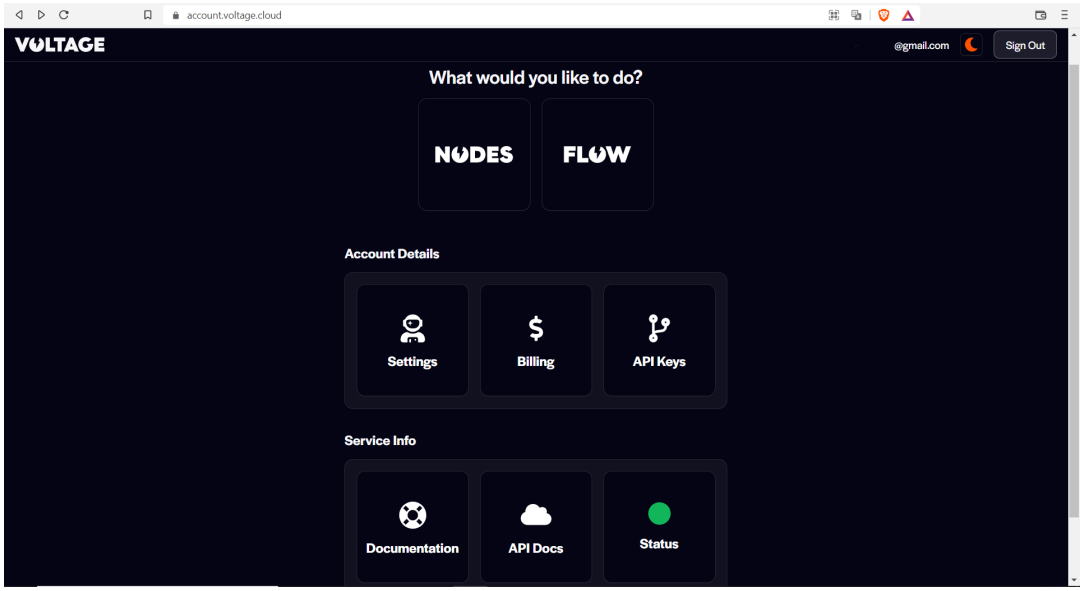


Figura 5.2: Menú principal de Voltage.

Aquí tenemos la opción de entrar en Nodes o en Flow. Flow es otro servicio que ofrece que comentaremos en el siguiente apartado, de momento, vamos a centrarnos en Nodes.

El menú de Nodes consiste en un resumen de los nodos abiertos que el usuario ha creado y un botón que permite la creación de nodos (ver figura 3.3). Debido a que en un primer momento el usuario no dispone de nodos vamos a ver como se crea uno.

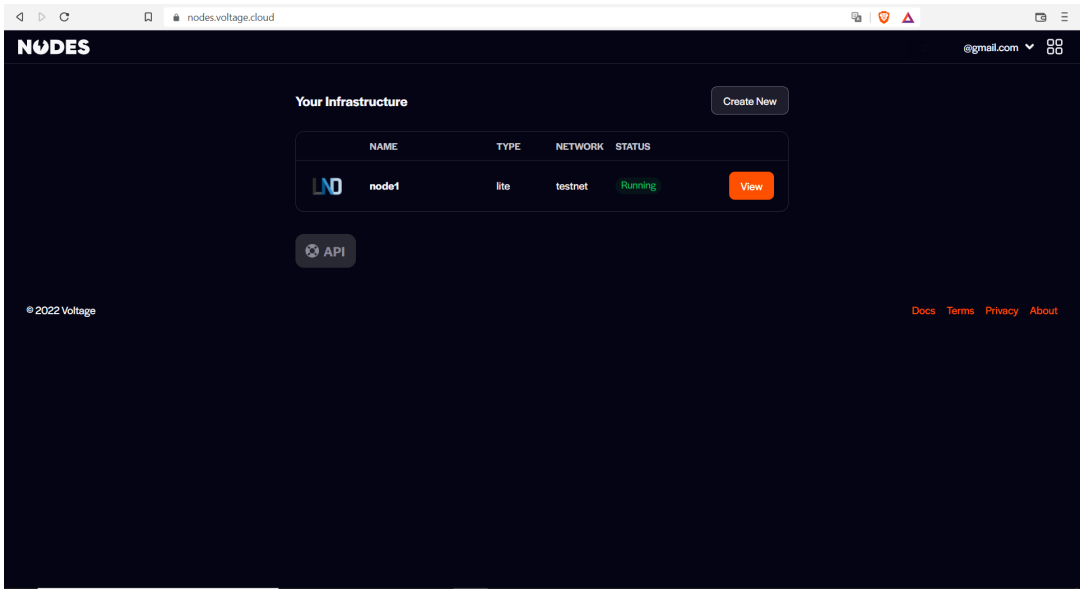


Figura 5.3: Menú principal de Nodes.

Tras seleccionar “Create New”, se nos mostrará una nueva página con los distintos tipos

de nodos que podemos crear, un nodo LND como el que hemos creado en el apartado anterior, un servidor BTCPay, y un nodo de Bitcoin Core. Nosotros vamos a elegir un nodo LND, tras lo cual, se nos mostrará una página la presentada en la figura 3.4.

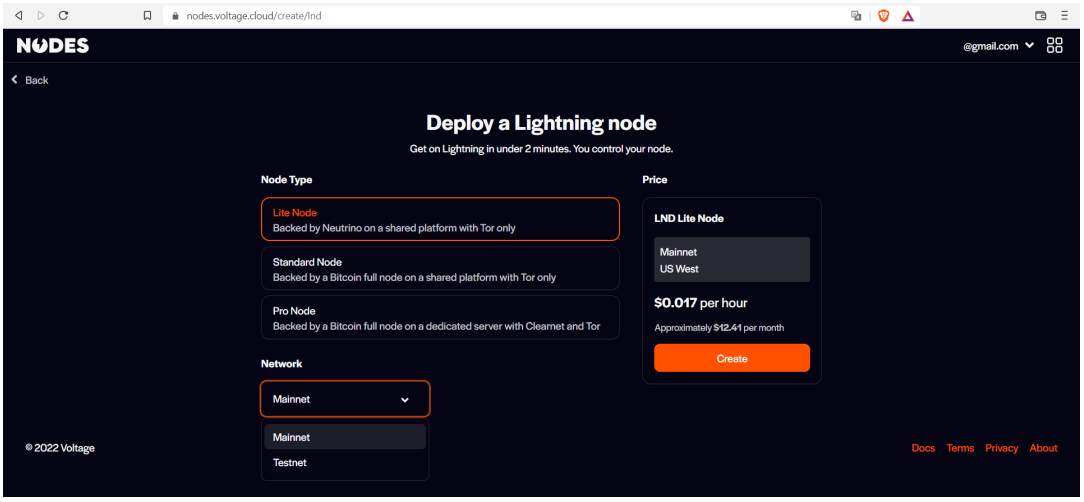


Figura 5.4: Opciones de creación de nodo.

Vamos a elegir la red Testnet como ya llevamos haciendo durante todo el trabajo, y para el tipo de nodo vamos a escoger el *standard*. Voltage recomienda utilizar el nodo *standard* para negocios o nodos de enrutamiento y elegir los Lite para nodos personales.

Tras seleccionar el tipo le tendremos que dar un nombre al nodo y establecer una contraseña con la que tendremos que desbloquear el nodo. Tras esto, se nos presentará el menú principal del nodo (ver figura 3.5).

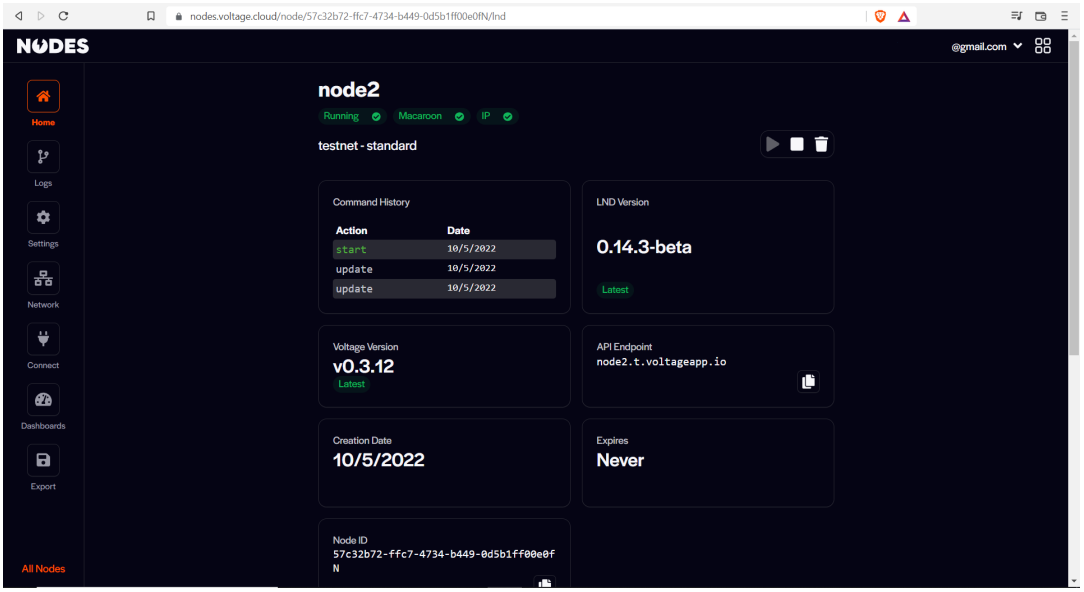


Figura 5.5: Menú principal del nodo.

Llegados a este punto, hay que comentar que Voltage no permite introducir fondos en el nodo, realizar transacciones, o crear canales, Voltage simplemente ofrece un nodo. Para poder operar con él vamos a tener que conectarnos a un nodo por medio de otro software, en concreto vamos a utilizar Zeus.

5.3.2. Conexión remota a un nodo Lightning

Zeus es una cartera de código abierto sin custodia de llaves que está disponible para dispositivos android e iOS. Para podernos conectar a nodos, Zeus ofrece la posibilidad de hacerlo mediante códigos QR (Figura 3.6).

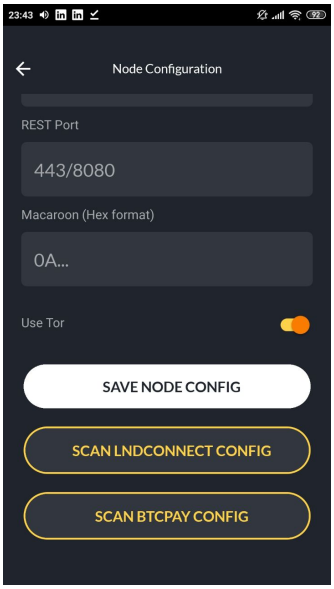


Figura 5.6: Conexión a un nodo en Zeus.

El código que hay que escanear lo proporciona Voltage en el apartado Connect de la barra lateral y seleccionando Zeus como aplicación (Figura 3.7).

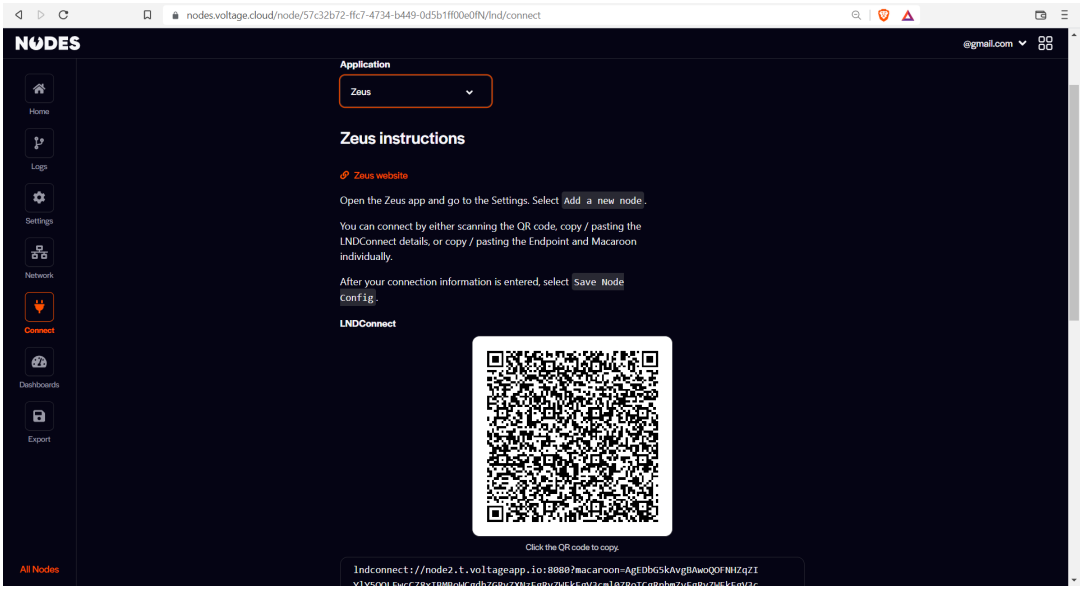
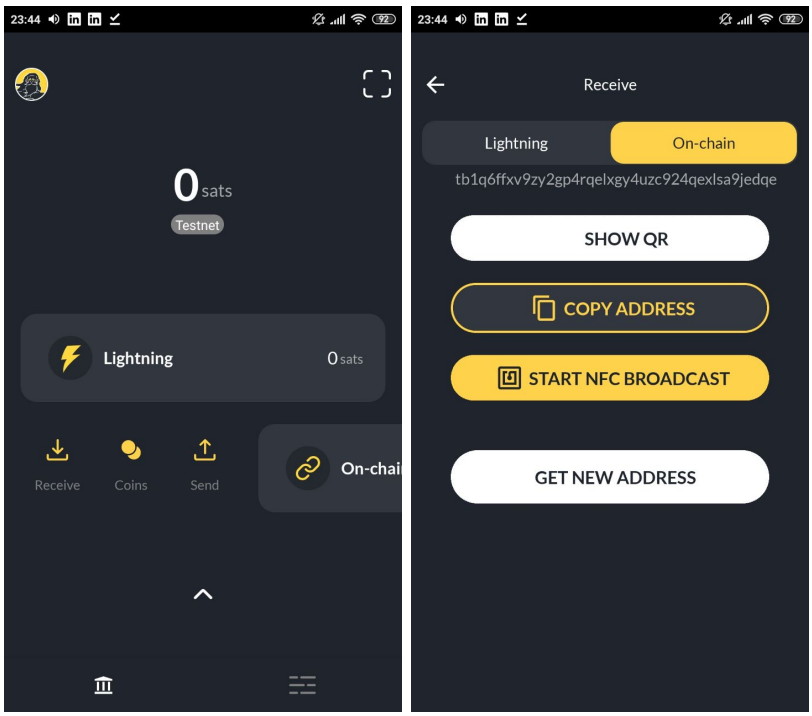


Figura 5.7: Menú principal del nodo.

Una vez estemos conectados tenemos que introducir algunos fondos, para lo cual podemos utilizar los “grifos” que utilizamos en el apartado 3.2.2.

La dirección que necesitamos para poder enviar bitcoins puede crearse desde Zeus como muestra la figura 3.8.



(a) Menú principal de Zeus. (b) Opciones al recibir bitcoins.

Figura 5.8: Creación de una dirección.

Por último, la creación de un canal pide unos campos similares a los que rellenamos cuando creamos el canal por comandos. Tras crear el canal deberíamos tener una página como la que presenta la figura 3.9.

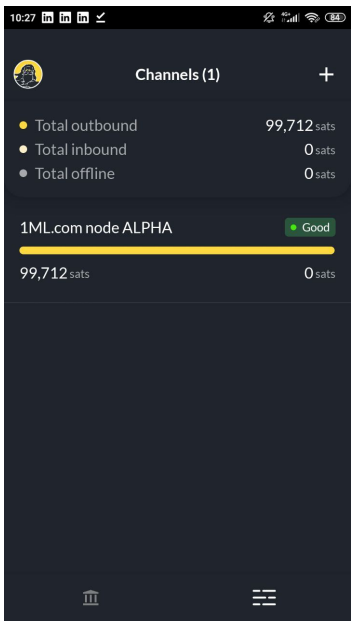


Figura 5.9: Sección de canales en Zeus.

Una vez hecho esto podemos crear los canales que queramos y despreocuparnos del nodo ya que sabemos que se mantendrá en ejecución.

Tras haber estudiado el alquiler de nodos y las ventajas que ofrece, vamos a realizar un breve estudio de Lightning Pool, un servicio de alquiler presente en la red Lightning pero que en vez de tratar con nodos trata con canales.

Es posible que el alquiler de canales no entre en la sección de instalación de nodos, pero si queremos usar nodos Lightning como intermediarios en la Mainnet es probable que tengamos que hacer uso de este servicio en algún momento.

5.4. Lightning Pool

Lightning Pool, desarrollado por Lightning Labs [32], es un mercado en el que usuarios con poca liquidez entrante pueden alquilar canales a usuarios con capital sobrante.

Tanto en el caso de la instalación como en el alquiler de un nodo, hemos creado canales con una capacidad de 100.000 satoshis, en el que la liquidez saliente es 100.000 y la entrante es 0, es decir, todos los fondos se encuentran en el nodo que hemos creado.

Esto no es un problema si queremos realizar un pago ya que el dinero puede viajar fácilmente al otro nodo, pero cuando queremos recibir un pago aparece un problema.

Usando Testnet, como llevamos haciendo todo el trabajo, podemos realizar un pago a alguna página o cuenta que los acepte y así balancear el canal, pero si queremos utilizar la red Lightning de manera seria, vamos a tener que acabar utilizando Mainnet, y balancear

el canal así ya no es posible. En estos casos, el usuario puede utilizar Lightning Pool para contratar un canal en el que toda la capacidad del canal se encuentre en el nodo externo.

Si recordamos, Voltage ofrece otro servicio llamado Flow, Flow es una interfaz para usar Lightning Pool. Desafortunadamente, Flow no ofrece soporte para Testnet debido probablemente a que la falta de liquidez se puede resolver fácilmente.

Por otro lado, este servicio también presenta potencial como modelo de negocio, ya que un usuario con capital extra puede ofrecer sus fondos para crear canales y recibir un pago a cambio.

Para finalizar el capítulo vamos a realizar un análisis sobre la rentabilidad que pueden ofrecer estos sistemas en la red Lightning. Debido a que Lightning Pool no es muy conocido vamos a ceñirnos al uso de nodos como intermediarios.

5.5. Rentabilidad de nodos intermedios

Debido a la naturaleza privada de la red Lightning, resulta imposible extraer información como la capacidad de los canales o cantidad de satoshis en las transacciones a partir de los datos disponibles. Por eso, la mayoría de los estudios realizados utilizan simuladores de tráfico para estimar ingresos.

Uno de los últimos estudios realizados [33] utiliza un simulador para visualizar como los ingresos de ciertas entidades varía atendiendo a factores como la cantidad de satoshis de las transacciones realizadas. Los resultados que obtuvieron pueden visualizarse en la figura 3.10.

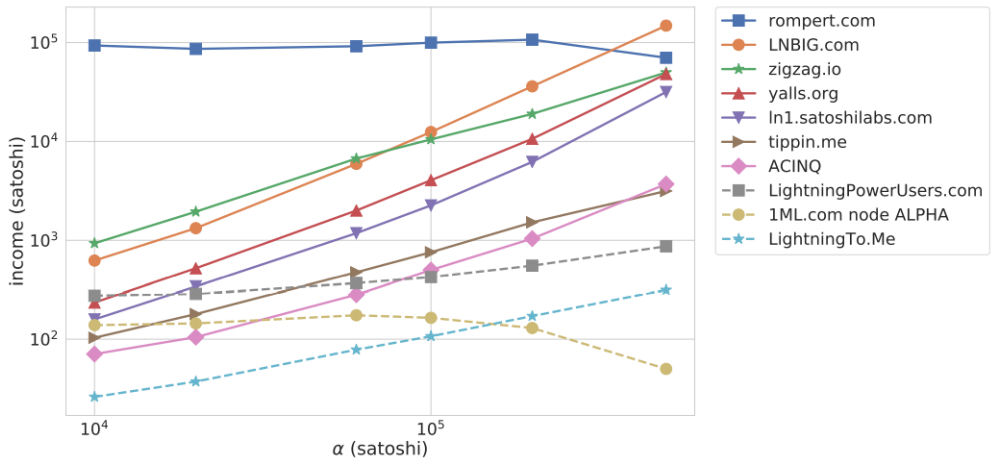


Figura 5.10: Ingresos generados a partir de transacciones por valor α.

Fuente: [33]

El estudio también peresenta una estimación del tamaño óptimo que deberían tener los canales de cada entidad si quisieran maximizar su ROI (Return of Investment o re-

torno sobre la inversión). Estos resultados muestran que muchos nodos deberían reducir la capacidad de sus canales a la mitad, y algunos incluso por debajo del 10 %.

En general el estudio indica que "la participación de la mayoría de los nodos en la red Lightning es económicamente irracional con la presente estructura de tasas". La razón por la que podemos ver que en la Figura 3.10 algunas entidades no generan casi beneficios es la estructura de tasas que tienen fijada.

Existen dos tasas en la red Lightning, una base aplicada por igual a todas las transacciones, y un porcentaje aplicado a cada transacción. La razón por la que vemos que 1ML.com node ALPHA genera menos beneficios cuando maneja transacciones con más satoshis es debido a que su estructura de tasas está compuesta por una base normal y una porcentual ínfima.

Existen estudios [34] que han investigado acerca de las tasas óptimas para generar beneficio sin que la red pierda su atractivo, aunque al no utilizar ningún tipo de simulación, los resultados obtenidos son relativamente complejos de interpretar.

Aunque puede que actualmente un negocio basado en nodos intermedios no sea un modelo sostenible, podemos ver que la figura 3.11 muestra que un aumento de las transacciones realizadas podría hacer de esta red un modelo rentable.

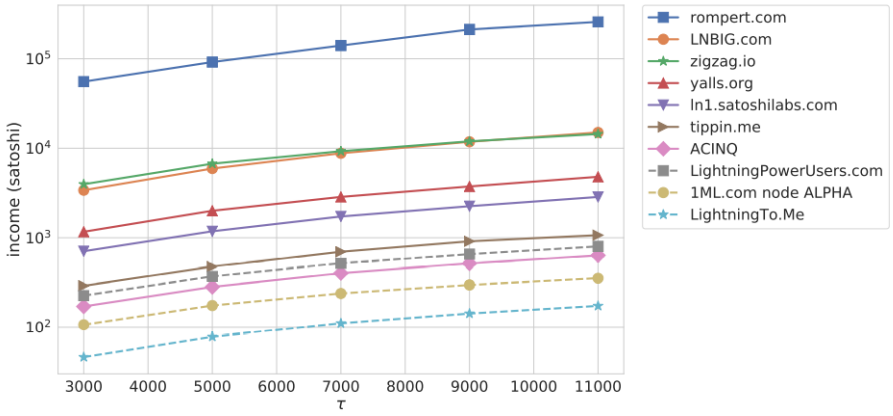


Figura 5.11: Ingresos por número de transacciones.

Fuente: [33]

Con esto damos por finalizado el capítulo relacionado con la instalación y uso de nodos en la red Lightning. Antes de poder realizar un análisis de los requerimientos de un negocio para poder participar en la red, necesitamos una mejor comprensión del funcionamiento de las carteras.

Capítulo 6

Instalación y uso de una cartera Lightning

Normalmente la instalación o adquisición de un nodo no suele tener motivos monetarios como los explicados en el capítulo anterior, sino que se suelen utilizar junto a una cartera para poder realizar pagos.

Este capítulo trata el uso de carteras con el fin de poder realizar transacciones en la red Lightning. Existen varios tipos de carteras que podemos utilizar atendiendo a varias especificaciones.

Las carteras Bitcoin se dividen en carteras "calientes" y "frías". Las calientes ofrecen un acceso rápido, son normalmente online, y suelen utilizarse para el uso de bitcoins. Por otro lado, las frías suelen usarse para almacenar bitcoins, por lo tanto su acceso requiere de más pasos, y suelen ser *offline*. Debido a la naturaleza de la red Lightning que se basa en el intercambio de monedas y requiere una conexión a la red constante es imposible que existan carteras frías para este servicio.

Al igual que hemos hecho hasta ahora, vamos a ceñirnos al uso de Testnet siempre que podamos para la realización de pruebas. Vamos a comenzar comentando la cartera que utilizamos al instalar LND con la que ya estamos familiarizados y luego comentaremos que tipos de carteras existen y qué ofrecen las principales aplicaciones.

6.1. Cartera LND

Debido a las necesidades de un nodo Lightning, ha sido necesario realizar la instalación de esta cartera durante el capítulo anterior, más concretamente en el apartado 3.2.2. Se va a evitar repetir el proceso por este mismo motivo y se va a continuar donde se dejó.

Si recordamos, acabábamos de crear un canal con el nodo de 1ML por valor de 100.000 satoshis. Es ahora cuando nos encontramos con un problema que ya hemos comentado, toda la capacidad del canal se encuentra de nuestro lado, por lo tanto solo podemos realizar pagos.

Para balancear el canal podemos utilizar páginas como Starblocks [35], que permiten realizar compras con bitcoins de Testnet. Para realizar un pago, la página nos genera un *invoice* que tenemos que utilizar (figura 4.1). Los comandos necesarios para ejecutar el pago correspondiente al *invoice* de la figura 4.1, para uno y varios nodos serían, respectivamente:

```
lncli --network=testnet payinvoice lntb15u1p3g...p2qq2hkrpy

lncli --network=testnet --rpcserver=localhost:<puerto>
    --datadir=<directorio> payinvoice lntb15u1p3g...p2qq2hkrpy
```

Figura 6.1: *Invoice* generado en Starblocks.

Figura 6.2: Confirmación de pago realizado.

Ahora que tenemos un canal con capacidad repartida entre los dos nodos, necesitamos otro nodo desde el que mandar un pago para comprobar que funciona correctamente. Para ello podemos utilizar el nodo de Voltage que hemos utilizado anteriormente. Se puede generar un *invoice* para uno y varios nodos respectivamente, ejecutando los siguientes comandos (el *invoice* se puede visualizar en la figura 4.3):

```
lncli --network=testnet addinvoice --amt=100

lncli --network=testnet --rpcserver=localhost:<puerto>
    --datadir=<directorio> addinvoice --amt=100
```

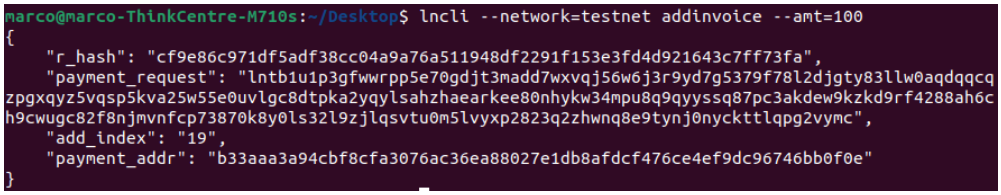


Figura 6.3: Generación de un *invoice* (payment_request).

Ahora, para poder pagar con el nodo alquilado de Voltage, tenemos que utilizar la aplicación de móvil Zeus que hemos conectado previamente. Pero tener que copiar el *invoice* en el móvil resulta inviable si queremos que estos pagos se realicen rápidamente.

Si recordamos, la aplicación de Zeus permite configurar el nodo de Voltage por medio de un QR, y la realización de pagos funciona de manera similar, así que en vez de copiar el *invoice* vamos generar un QR.

Para esto se ha utilizado Angularx-qrcode, una librería de Angular. Aunque la instalación de esta librería no forma parte de los objetivos del presente trabajo, incluiremos

el código necesario para su instalación en el Anexo III. Una vez instalada y ejecutada la aplicación Angular podemos escanear el código (figura 4.4) con la aplicación móvil y realizar el pago.

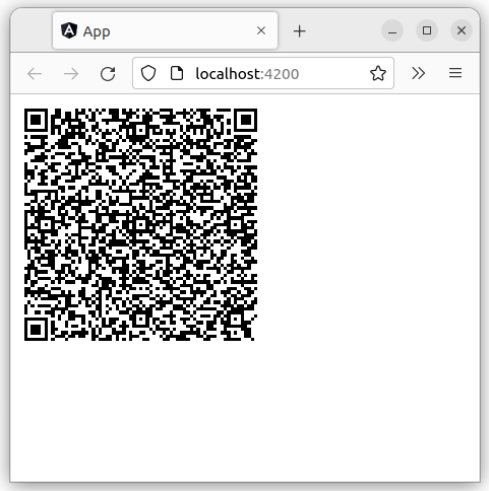


Figura 6.4: Código QR creado con Angularx-qrcode.

La confirmación de la recepción del pago puede observarse en la terminal en la que hemos ejecutado el comando lnd (figura 4.5).

```
2022-05-18 09:48:18.054 [DBG] INVC: Invoice(pay_hash=aa128eb56e8fad2e7b8a307ab1325516b87324319a53577c968aefe4e6d1d797,
 pay_addr=135a507f27e41d36be6cca9ba04cd0b4e9a5fbce7303507d68635b023c5b874a): settle resolution result outcome: settled
, at accept height: 2225946, amt=100000 mSAT, expiry=2225989, circuit=(Chan ID=2197391:6:0, HTLC ID=6), mpp=total=1000
00 mSAT, addr=135a507f27e41d36be6cca9ba04cd0b4e9a5fbce7303507d68635b023c5b874a, amp=<nil>
2022-05-18 09:48:18.054 [DBG] HSWC: ChannelLink(8000d4938ada4cbd9431ea43ae5ac30b89f2fcbfa37dc58a958e50a74c0e4981:0): r
eceived settle resolution for (Chan ID=2197391:6:0, HTLC ID=6) with outcome: settled
2022-05-18 09:48:18.054 [INF] HSWC: ChannelLink(8000d4938ada4cbd9431ea43ae5ac30b89f2fcbfa37dc58a958e50a74c0e4981:0): s
ettling htlc aa128eb56e8fad2e7b8a307ab1325516b87324319a53577c968aefe4e6d1d797 as exit hop
```

Figura 6.5: Código QR creado con Angularx-qrcode.

Por lo general, los usuarios no utilizan el cliente lncli para manejar su cartera y optan por aplicaciones y servicios que faciliten la tarea. Vamos a comentar algunas de estas aplicaciones y algunas características que las diferencian.

6.2. Aplicaciones de carteras

Es normal que la mayoría de usuarios prefieran no utilizar comandos en una terminal y opten por aplicaciones que hagan que su experiencia con la red sea más sencilla, pero a la hora de decidir una aplicación es posible no entender la diferencia entre ellas.

La primera división que podemos realizar de las carteras es atendiendo al dispositivo al que están dirigidas.

Como ya hemos comentado, la Lightning Network permite realizar micropagos con bitcoins de manera rápida y con tarifas bajas, lo cual lo convierte en una forma de pago

muy conveniente de tener a mano. Debido a esto donde más variedad de carteras podemos encontrar es en dispositivos móviles.

Existe un problema básico al intentar ejecutar un nodo Lightning en un móvil, la memoria necesaria para almacenar la *blockchain*. Además, al tratarse de dispostivos móviles no es posible mantener un cliente Bitcoin ejecutándose constantemente, por eso, la mayoría se conectan a clientes (que guardan una copia de la *blockchain*) como Neutrino o Electrum.

Por otro lado podemos diferenciar las carteras disponibles para ordenadores, que normalmente son aplicaciones de móviles que han decidido ampliar soporte. Estas carteras suelen ser aplicaciones de escritorio, pero algunas pueden utilizarse en navegador o incluso instalarse como add-ons.

Para ver exactamente que diferencias existen entre las carteras disponibles vamos a hablar sobre características principales que definen su comportamiento

6.2.1. Tipos de carteras

Existen bastantes aspectos que influyen en la seguridad, privacidad, disponibilidad, o libertad de una cartera. Con el fin de proporcionar suficiente información como para que el usuario pueda elegir correctamente la cartera más apropiada para sus necesidades, vamos a indagar más a fondo sobre el funcionamiento de las carteras.

Custodia

Cuando se habla de custodia en las carteras, se hace referencia a la pertenencia de las llaves que desbloquean la cartera.

Las carteras sin custodia son aquellas que no guardan información sobre las claves de recuperación, y por lo tanto es el usuario el que tiene que preocuparse de no olvidarla. Por otro lado, las carteras con custodia no comunican al usuario la clave de recuperación y simplemente permiten acceder a ellas con un usuario y contraseña como si se tratara de una cuenta.

Esta característica es posiblemente la más importante cuando se considera el uso de una cartera, ya que no tener la clave es equivalente a no ser propietario de la cuenta aunque los fondos si que sean del usuario.

Conexión a Bitcoin

La cartera tiene que tener acceso a Bitcoin para poder crear canales, este tipo de conexión puede ser tradicional como la que hemos utilizado en la instalación del nodo en Linux, o se pueden conectar a clientes Bitcoin como Neutrino.

Estos clientes suelen utilizarse en dispositivos móviles ya que evitan la necesidad de almacenar una copia entera de la *blockchain* o sincronizarla.

Por otro lado, debido a que no tienen la *blockchain*, tienen que confiar en otros nodos para poder validar transacciones además de poder dar fallos de conexión más fácilmente.

Cartera Bitcoin

Algunos usuarios quieren poder utilizar tanto la red Lightning como la red Bitcoin, y algunas carteras solo permiten el uso de la red Lightning.

Conexión a nodo

Es posible que el usuario tenga ya un nodo propio o contratado y quiera conectarse a el desde una cartera, en ese caso tendrá que asegurarse que elige una que lo permita.

Tor

No todas las carteras se comunican con el nodo Lightning usando Tor, y como ya hemos comentado el anonimato es recomendable.

Administración de canales

Aunque la administración de canales no es necesaria ya que en la mayoría de carteras se ofrece el control automático de canales, es posible que haya usuarios que quieran gestionar cuántos satoshis se utilizan, y con qué nodos se crean canales.

Implementación de Lightning Network

Este aspecto realmente no es importante si se utiliza una aplicación de cartera, pero en el caso de que se quiera instalar un nodo, se pueden utilizar varios software dependiendo en la experiencia del usuario.

Los principales son c-lightning, Eclair, LND, y Electrum. De ellas c-lightning y LND son las que ofrecen programación de nivel más bajo, y LND es la implementación más usada, y por lo tanto la que tiene más soporte y una mayor comunidad de usuarios.

En la tabla 4.1 podemos ver algunas de las carteras más extendidas y las especificaciones individuales de cada una.

6.2.2. Uso de aplicaciones de carteras

El uso de estas carteras es bastante similar entre ellas, todas permiten realizar pagos por medio de códigos QR, y todas tienen un modo de generar un QR a partir de un *invoice*. Realmente estas aplicaciones simplifican mucho el uso de una cartera y son esenciales para

los usuarios, independientemente de su experiencia, puedan realizar pagos mediante la Lightning Network en cualquier momento, y de una forma sencilla.

Con esto terminamos el capítulo de carteras Lightning y damos paso al último capítulo, en el que utilizaremos todo lo visto hasta para proponer opciones para negocios que quieren utilizar la red Lightning como método de pago opcional.

	Alby	**Blue**	**Eclair**	**Electrum**
Aplicación Web/Escritorio	Add-on	No/No	No/Sí	No/Sí
Aplicación Android/iOS	No/No	Sí/Sí	Sí/No	Sí/No
Testnet	Sí	No	Sí	No
Con custodia	Sí	Sí (default)	No	No
Conexión a Bitcoin	Neutrino	Servidor Electrum	Servidor Electrum	Servidor Electrum
Implementación de red Lightning	LNDhub	LND	Eclair	Electrum
Cartera de Bitcoin	No	Sí	Sí	Sí
Conexión a nodo	Sí	Sí	Sí	Sí
Tor	No	No	No	Sí
Administración de canales	No	Sí	Sí	Sí

	Lightning	**Phoenix**	**Spark**	**Zap**	**Zeus**
Aplicación Web/Escritorio	No/Sí	No/No	Sí/Sí	No/Sí	No/No
Aplicación Android/iOS	Sí/Sí	Sí/No	Sí/No	Sí/Sí	Sí/Sí
Testnet	Sí	Sí	Sí	Sí	Sí
Con custodia	No	No	No	No	No
Conexión a Bitcoin	Neutrino	Servidor Electrum	JSON-RPC	Neutrino	ZeroMQ
Implementación de red Lightning	LND	Eclair	c-lightning	LND	LND
Cartera de Bitcoin	Sí	No	No	No	No
Conexión a nodo	No	Sí	Sí	Sí	Sí
Tor	No	Sí	Sí	Sí	Sí
Administración de canales	Sí	No	Sí	Sí	Sí

Tabla 6.1: Principales carteras y algunas de sus características

Capítulo 7

Recepción de pagos por Lightning Network en un pequeño negocio: espacio de soluciones

En este capítulo realizaremos un estudio a priori de las opciones, con respecto al software y hardware, que los diferentes tipos de negocios deberían barajar si quieren implementar pagos por medio de la red Lightning.

Lo primero que tendremos que hacer será analizar la tipología de negocios que engloba la definición de "pequeño comercio". Según la Dirección General de la Industria y de la PYME [36], una pequeña empresa es una empresa que ocupa menos de 50 personas, y cuyo balance anual no supera los 50 millones de euros, dejando a las microempresas en entidades con menos de 10 personas, y un balance anual menor a 2 millones de euros.

Atendiendo a las distintas posibilidades que existen en la política de cobros y pagos dentro de las organizaciones, podríamos considerar distintos tipos de negocios:

- Negocios gestionados por 1 persona: puede tratarse de pequeñas tiendas, peluquerías, o quioscos. Estos negocios suelen ser dirigidos por una persona autónoma con alguno o muy pocos empleados. Normalmente el dueño es el encargado de realizar todas los cobros y pagos, y por lo tanto solo el necesitará tener acceso a la red Lightning.
- Negocios gestionados por más de 1 persona: normalmente negocios de restauración u hostelería. En este tipo de negocios el dueño suele realizar tareas directivas, dejando labores de gestión como pagos y cobros a sus empleados. Por lo tanto, será necesario que varias personas tengan acceso a la red Lightning.
- Negocios multisede: por lo general, cadenas locales.

Por otro lado, comentaremos la opción de que la empresa tenga varios propietarios como un caso a parte. Dentro de cada apartado, discutiremos, la naturaleza del nodo Bitcoin y Lightning, la cartera, y algunas medidas necesarias para asegurar el funcionamiento

correcto del sistema. Empezaremos con la situación que tiene menos restricciones, los negocios formados por 1 persona.

7.1. Negocios gestionados por una persona

Estas empresas ofrecen la mayor libertad cuando hablamos de implementar pagos con Lightning. Un negocio dirigido por una persona que quiere implementar Lightning prácticamente equivale a un individuo que quiere usar Lightning, y por lo tanto tendrá muchas opciones entre las que elegir.

7.1.1. Nodo personal

La instalación y gestión de nodos Bitcoin y Lightning conlleva muchas preocupaciones, como la realización de copias de seguridad de la *blockchain*, copias de seguridad de la cartera, sistemas preventivos en caso de cortes de luz, etc. Por eso, utilizar este sistema, aún teniendo los conocimientos necesarios, puede ser un poco abrumador.

7.1.2. Alquiler de nodos

Esta opción resulta bastante más razonable, ya que no tenemos que preocuparnos de la gestión de los nodos.

Para poder utilizar el nodo, el usuario puede elegir cualquiera de las carteras de la tabla 4.1 que permiten conexión a nodo como Alby para navegador, Zap para escritorio, o Blue para dispositivos móviles. Adicionalmente, se recomienda que la aplicación permita gestionar los canales con los que se conecta el nodo, debido a que el piloto automático puede realizar malas decisiones, como dedicar demasiados fondos a un canal o crear canales con nodos mal conectados.

El alquiler de un nodo nos ahorra la preocupación de gestionar uno propio, pero nos obliga a pagar un coste por el servicio.

7.1.3. Aplicaciones de carteras

Por último, llegamos a la opción de utilizar aplicaciones de carteras, ésta sería probablemente la más sencilla y más conveniente para este negocio. Veamos por qué:

- No requiere gestión de ningún nodo: muy similar a la opción anterior no requiere que el usuario se preocupe del estado de los nodos.
- Uso sencillo independientemente del nivel del usuario: el uso de la red Lightning puede ser complicado, por eso estas carteras intentan que la experiencia de uso sea lo más sencilla posible para el usuario.

- No requieren pago por el servicio: todas las carteras son de código abierto y gratuitas.
- Nadie más necesita tener acceso a la cartera: esta opción no solo es viable bajo esta categoría, sino que empresas con más de un empleado podrían usarlas también, siempre que solo necesiten un dispositivo desde el que poder realizar o recibir transacciones.

El tipo de aplicación que elegir depende de las preferencias del usuario, pero vamos a enumerar algunas que podrían considerarse claves.

- Custodia de la llave: una cartera sin custodia que requiera que el usuario recuerde la clave siempre será mejor, sobre todo si queremos utilizar la cartera en nuestro negocio. La pertenencia de una cartera viene dada por la pertenencia de la llave, de ahí viene la famosa frase "Not your keys, not your coins".
- Administración de canales: ya lo hemos comentado anteriormente, pero es bastante importante que el usuario pueda decidir los canales que quiere utilizar, sobre todo porque si el nodo no está lo suficientemente bien conectado, es posible que algunos pagos fallen y no puedan realizarse.
- Cartera de Bitcoin: también es importante que la cartera permita transacciones en Mainnet, no porque se vayan a realizar constantemente, pero que las pueda hacer en caso de que el usuario las necesite.
- Tor: se ha remarcado la importancia del anonimato en la red en varias partes del trabajo, pero si encima la cartera está asociada con una empresa es mucho más importante tomar todas las medidas de seguridad posibles.

Teniendo todas estas características en cuenta, las carteras más apropiadas de la tabla 4.1 sería Electrum.

7.2. Negocios gestionados por más de una persona con una sede

Si estamos hablando de un negocio en el que más de una persona puede estar a cargo de realizar cobros o pagos, se nos cierran algunas opciones de las que disponíamos en el apartado anterior. En concreto utilizar una aplicación de cartera ya no es posible, a no ser que se quieran utilizar varias carteras con nodos distintos, pero tratándose de un negocio acabaría siendo complicado de gestionar todos los fondos a uno común. Por otro lado, las otras opciones se vuelven bastante viables.

7.2.1. Nodo personal

Utilizando un nodo propio, podríamos conectarnos a él desde varios dispositivos utilizando algunas de las aplicaciones de tabla 4.1 y así permitir que varios empleados puedan

realizar transacciones. A continuación son algunas de las características del material que habría que tener en cuenta para poder levantar un nodo.

La presencia de un nodo propio implica la independencia de otros servicios, y que por lo tanto, siempre podemos disponer de nuestros bitcoins para utilizar la red. Por otro lado, insistimos en lo anterior, mantener un servidor requiere de atención y puede que si el negocio todavía no sea lo suficientemente grande como para justificar tener un nodo propio, sea mejor utilizar servicios de terceros.

7.2.2. Alquiler de nodos con aplicaciones de carteras

El alquiler de nodos de Voltage es bastante apropiado para la situación de negocio presente, ya que podemos conectarnos a él desde varios dispositivos, pero no tenemos que mantener el servidor en el que está corriendo el nodo.

Voltage presenta tres opciones de nodo:

- *Lite node*: un nodo con Neutrino, normalmente una opción para una cartera personal
- *Standard node*: un nodo Lightning completo, normalmente usados en negocios o routing.
- *Proffesional nodes*: un nodo Lighting completo con un servidor dedicado, normalmente usados en negocios.

La diferencia entre un nodo *standard* y uno *proffesional*, es que el *standard* comparte servidor con otros nodos, mientras que el *proffesional* tiene uno dedicado. Para el caso de negocios pequeños el *standard* sería el más apropiado dejando el *proffesional* para empresas más grandes.

Para conectarnos al nodo será crucial utilizar una aplicación que permita conectarse a nodos externos. Al igual que en el apartado anterior, elegiremos carteras que permitan administración de canales y TOR, lo cual nos deja con Electrum, Spark, Zap, y Zeus.

7.3. Negocios con más de una sede

El caso de que un negocio que quiera ofrecer un servicio de pago por Lightning en más de uno de sus establecimientos es un caso bastante similar al anterior en el que varias personas pueden utilizar el nodo Lightning. Pero, tendremos que especificar algún cambio para poder gestionar dos lugares de trabajo de manera ordenada.

En esta situación el alquiler de un nodo a terceros sigue siendo una opción válida, pero si se trata de un negocio lo suficientemente grande como una cadena local, es más que factible discutir la instalación de un nodo personal.

Para poder mantener un nodo personal se necesitará el siguiente hardware:

- Dos discos duros con un mínimo de 500 Gb: utilizados para almacenar la *blockchain* (uno de ellos para copia de seguridad). actualmente la *blockchain* pesa algo más de 400 Gb, idealmente el disco debería de ser de 1 Tb para evitar que haya problemas de memoria en un futuro. Para facilitar esta tarea, se puede hacer uso de sistema RAID para realizar copias de seguridad.
- Raspberry pi 4: si se quiere mantener el servidor desde una raspberry, se tendrá que utilizar por lo menos un modelo 4 ya que el 3 no es compatible con la tecnología. Puede utilizarse un NAS como sistema de almacenamiento que además permite realizar copias de seguridad en la nube.
- SAI: un sistema de alimentación initerrumpido es crucial en caso de fallos de suministro.

Como sofware para el nodo de Bitcoin se puede utilizar el que desarrolla Bitcoin Core [25] y como software del nodo Lightning recomendamos LND [32].

Ahora tenemos que barajar la opción de cuántos nodos instalar. Podemos instalar un nodo Bitcoin y uno Lightning, pero eso haría que si uno se cae ninguna de las sedes podría usar la red. Utilizando un nodo Bitcoin y uno Lightning por sede se arreglaría el problema, pero los gastos en electricidad podrían exceder los beneficios que ofrece.

Por último se podría tener un nodo Bitcoin con varios nodos Lightning asociados, esto cubriría el caso de que se pudiera caer uno de los nodos Lightning, manteniendo el coste relativamente bajo. Además, utilizar varios nodos ayudaría en el recuento de beneficios y gestión de cada sede.

Debido a que varias personas van a estar a cargo de realizar transacciones, sería una buena idea registrar quien ha supervisado cada operación por medio de una aplicación de identificación.

7.4. Negocios con varios propietarios

Como ya hemos comentado, el propietario de las llaves de la cartera es, a nivel práctico, el propietario de las monedas que contiene. Por eso, una situación en la que un negocio sea compartido entre varios propietarios puede ser delicada.

La opción más sencilla es que todos conozcan la llave, pero eso requiere de completa confianza de todos los participantes. Realmente no existen muchas opciones a parte de la nombrada.

Para evitar desgracias lo más sensato sería realizar cambios a euros con cierta frecuencia y mantener una cantidad pequeña en bitcoins en las carteras.

Para terminar el trabajo vamos a realizar un pequeño análisis de las diferencias en costes que supone el uso de la red Lightning en comparación con el “dinero de plástico” implementado en todo el mundo. Este resumen resulta crucial para las empresas ya sean grandes o pequeñas, ya que ninguna optaría por utilizar una tecnología poco extendida como la red Lightning si resulta económicamente menos viable.

7.5. Análisis económico del uso de la red Lightning frente a un sistema basado en tarjetas de crédito

En este apartado realizaremos una comparación de el coste económico que le puede suponer a un negocio el uso del sistema Lightning y lo compararemos con un sistema de tarjetas de crédito.

Las tasas de la red Lightning se dividen en dos tipos: una base y una porcentual por cada nodo intermedio de la transacción. La base tiene un valor estándar de 1 satoshi, y la porcentual un valor estándar de 10^{-6} satoshis. En la figura 5.1 podemos ver el porcentaje de los nodos que tienen estos valores por debajo, en, o por encima del estándar.

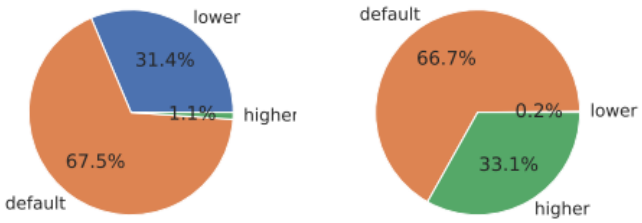


Figura 7.1: **Izquierda:** tasas base con respecto al estándar (1 satoshi). **Derecha:** tasas porcentuales con respecto al estándar (10^{-6} satoshis)

Fuente: [33]

Por lo tanto, las tasas de una transacción son:

$$y = (c + mx) * n$$

Donde:

- y: tasas totales por la realización de la transacción
- c: tasas base por nodo, utilizaremos 1 satoshi.
- m: tasas porcentuales por nodo, utilizaremos 10^{-6} satoshis.
- x: cantidad de satoshis en la transacción sin tasas.
- n: número de nodos involucrados sin contar inicial o final. Ninguna transacción debería superar los 6.

Estas tasas son muy bajas, démonos cuenta que la tasa base de 1 satoshi equivale a 0.028 céntimos, y para que la tasa porcentual influyera en el coste final deberían hacerse pagos de miles de euros, lo cual es difícilmente posible dadas las cantidades máximas de los canales.

Por otro lado, el coste de mantenimiento de un nodo si que requiere de cantidades menos significativas. El alquiler de un nodo de Voltage resulta en unos 30$ (a día 24/05/2022 ∼ 28€) al mes. Por otro lado, la instalación de un nodo requiere algunas cuentas más.

Teniendo en cuenta lo comentado anteriormente en el capítulo vamos a suponer que se va a utilizar un ordenador, discos de almacenamiento de 1Tb, y un SAI, más el coste eléctrico. Podemos estimar ∼300€ por un ordenador, ∼100€ por los dos discos duros, y ∼150€ por el SAI, a esto faltaría sumarle el coste eléctrico. Teniendo en cuenta que los equipos eléctricos se amortizan a 3 años, este coste supondría unos 15€ semanales.

Para el coste eléctrico vamos a utilizar 250 W/h como consumo del sistema. Por lo tanto, obtenemos un gasto mensual de 250W/h*24h/dia*30dias/mes = 180 KW. Con un precio actual de 0.27€ el KW/h tenemos un gasto de 48€ al mes.

El coste de ofrecer un servicio de pago electrónico basado en tarjetas de crédito/débito puede resumirse en el alquiler de un TPV, y un porcentaje de los beneficios obtenidos por medio del sistema.

El coste de un TPV podemos estimarlo en ∼40 € al mes, y el porcentual se sitúa alrededor del 0.4 % de los beneficios, llegando hasta el 3 % en caso de que se traten de tarjetas extranjeras. Es aquí donde reside la gran diferencia con la red Lightning.

En la red Lightning podemos estimar que 100 transacciones de 10€ conllevan una tasa total de 0.10€, mientras que un sistema de tarjeta de crédito conlleva 40€ en comisiones.

A mayores, podríamos tener en cuenta el coste que supondría tener que alquilar más de un TPV, lo cual no crea ningún problema si se dispusiera de un nodo Lightning, ya que solo haría falta conectar otra cartera.

Por lo tanto, podemos ver que un sistema de pago basado en la red Lightning resulta una opción muy conveniente que los pequeños comercios podrían utilizar para optimizar su negocio.

Capítulo 8

Conclusiones y Líneas futuras

8.1. Conclusiones

Cuando se planteó la elaboración de este trabajo se perseguía el objetivo principal de investigar el funcionamiento de la red Lightning con el fin de poder indicar una configuración de hardware y software que se adecúe a las necesidades de los pequeños negocios que quieren disponer de micropagos por Bitcoin.

Pasamos a exponer los principales resultados logrados durante la realización de este trabajo.

- Se ha obtenido de un conocimiento de la Lightning Network sólido: uno de los subobjetivos clave para la realización del estudio originalmente concebido. Este entendimiento de la red ha sido adquirido durante el periodo de redacción de los dos primeros capítulos "Bitcoin" y "Lightning Network". Y que ha sido una base crítica para ejecutar las secciones prácticas del proyecto.
- Se ha realizado la instalación de un nodo Lightning: se ha investigado y llevado a cabo la instalación de un nodo Lightning desde cero. Adicionalmente, se ha indagado en el alquiler de nodos y se han analizado ambas opciones en cuanto a las ventajas que ofrece cada una.
- Se ha realizado un análisis de las carteras en profundidad: se han descrito las principales características que componen a las carteras de la red, y se han definido los aspectos más importantes que se deberían considerar al elegir una cartera, como la custodia de las llaves, o la posibilidad de administrar canales Lightning. Este análisis ha sido completado con una tabla resumen de las características principales de las carteras más populares.
- Se han realizado transacciones entre carteras: disponiendo ya de varios nodos funcionales con carteras asociadas, hemos efectuado una serie de transacciones entre ellos con la finalidad de probar y anotar el procedimiento. Para ello, las transacciones se han realizado utilizando códigos QR e invoices (códigos de factura).

- Se ha realizado una descripción de las necesidades de pequeño negocios que quieran ofrecer pagos en la red Lightning: primero se ha realizado una distinción entre varios tipos de pequeños negocios, y se ha asignado una configuración a cada uno basándose en sus necesidades y en lo descubierto durante todo el trabajo.

8.2. Líneas futuras

Pasamos a exponer las futuras líneas de investigación que constituyen la prolongación natural del presente trabajo fin de grado.

Se propone realizar una implantación real en algún pequeño comercio que sirva para probar la operativa y adentrarnos en los pequeños problemas que surgirán en el proceso. Además, nos serviría como banco de pruebas para la obtención de los datos necesarios para realizar posibles análisis económicos más precisos.

En este sentido, se plantea realizar un análisis económico que estudie la rentabilidad que supone el uso de la red Lightning frente a otros sistemas similares como las tarjetas de crédito/débito.

Normalmente la realización de un análisis económico puede realizarse como parte de un trabajo, pero debido a que la Lightning Network se basa en el pilar fundamental del anonimato, resulta complicado realizar un estudio analítico y otros métodos como la experimentación son necesarios. Para ello es posible utilizar el simulador de [33] que puede encontrarse en [37]. Al ser código abierto es posible incluso modificarlo para que se adecúe mejor a la tarea en mano.

Anexo I: archivo bitcoin.conf

```
testnet = 1
server = 1
daemon = 1
rpcuser = <usuario>
rpcpassword = <contraseña>
```

Anexo II: archivo lnd.conf

```
[Application Options]
externalip=34.117.59.81:9735
debuglevel=debug,PEER=info


[Bitcoin]
bitcoin.active=true
bitcoin.testnet=true
bitcoin.node=bitcoind


[Bitcoind]
bitcoind.rpcuser=<usuario>
bitcoind.rpcpass=<contraseña>
```

Anexo III: creación de códigos QR con Angular

Instalación de Angular

```
sudo apt install nodejs
npm install -g @angular/cli
```

Creación de una aplicación

```
ng new my-app
cd my-app
npm install angularx-qrcode --save
```

Distribución y contenido de archivos

src/app/app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';
import { QRCodeModule } from 'angularx-qrcode';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    QRCodeModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

src/app/app.component.ts

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  public myAngularxQrCode: string = null;

  constructor () {
    this.myAngularxQrCode = 'ItSoluionStuff.com';
  }
}
```

src/app/app.component.html

```
<qrcode [qrdata]="<invoice>" [width]="256"
    [errorCorrectionLevel]="'M'"></qrcode>
```

Ejecutar servidor

```
ng serve
```

Bibliografía

[1] Online gantt. URL `https://www.onlinegantt.com/#/gantt`. Última consulta: 7/06/2022.

[2] Andreas M Antonopoulos. *Mastering bitcoin*. O'Reilly,, 2019.

[3] Saifedean Ammous. *The bitcoin standard: the decentralized alternative to central banking*. John Wiley & Sons, 2018.

[4] Jaime Sánchez de Diego Martínez-Cabrera. Bitcoins¿ revolución o historia? 2014.

[5] Maribel Mendez Media. Análisis foda. bitcoins - criptomonedas. URL `https://cladera.org/foda/foda_detailseco.php?id_subcategory=538`. Última consulta: 13/05/2022.

[6] David Lee Chaum. Computer systems established, maintained and trusted by mutually suspicious groups. page 1, 1979.

[7] Tiana Laurence. *Blockchain for dummies*. John Wiley & Sons, 2019.

[8] Don Tapscott and Alex Tapscott. *Blockchain revolution: how the technology behind bitcoin is changing money, business, and the world*. Penguin, 2016.

[9] Wattana Viriyasitavat and Danupol Hoonsopon. Blockchain characteristics and consensus in modern business processes. *Journal of Industrial Information Integration*, 13:32–39, 2019.

[10] Karim Sultan, Umar Ruhi, and Rubina Lakhani. Conceptualizing blockchains: Characteristics & applications. *arXiv preprint arXiv:1806.03693*, 2018.

[11] IBM. What is blockchain. URL `https://www.ibm.com/topics/what-is-blockchain`. Última consulta: 11/03/2022.

[12] AMD. Blockchain explained. URL `https://www.amd.com/en/technologies/blockchain-explained`. Última consulta: 11/03/2022.

[13] Jake Frankenfield. Block (bitcoin block). URL `https://www.investopedia.com/terms/b/block-bitcoin-block.asp`. Última consulta: 18/02/2022.

[14] Bitcoin explorer, blocks. URL `https://www.blockchain.com/btc/blocks`. Última consulta: 13/05/2022.

[15] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[16] Greg Walker. learn me a bitcoin. URL `https://learnmeabitcoin.com/technical/target`. Última consulta: 08/03/2022.

[17] Darrel Hankerson, Alfred J Menezes, and Scott Vanstone. *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.

[18] Tp's go bitcoin tests - addresses. URL `https://gobittest.appspot.com/Address`. Última consulta: 25/03/2022.

[19] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments, 2016.

[20] Lommy. The semantics of the bitcoin layers. URL `https://bitcoinmagazine.com/culture/the-semantics-of-the-bitcoin-layers`. Última consulta: 20/03/2022.

[21] What are the bitcoin layers? – layer 3 vs. layer 2 vs. layer 1 crypto. URL `https://phemex.com/academy/bitcoin-layer-1-vs-2-vs-3`. Última consulta: 20/03/2022.

[22] Andreas M Antonopoulos, Olaoluwa Osuntokun, and René Pickhardt. *Mastering the Lightning Network*. .ºReilly Media, Inc.", 2021.

[23] George Danezis and Ian Goldberg. Sphinx: A compact and provably secure mix format. In *2009 30th IEEE Symposium on Security and Privacy*, pages 269–282. IEEE, 2009.

[24] Bastien Teinturier and Neil Saitug. Basis of lightning technology. URL `https://github.com/lightning/bolts`. Última consulta: 27/03/2022.

[25] Bitcoin, . URL `https://bitcoin.org/es/descargar`. Última consulta: 16/01/2022.

[26] Olaoluwa Osuntokun. Lightning network. URL `https://github.com/lightningnetwork/lnd`. Última consulta: 05/04/2022.

[27] Yet another bitcoin testnet faucet. URL `https://testnet-faucet.mempool.co/`. Última consulta: 28/04/2022.

[28] Bitcoin testnet faucet, . URL `https://bitcoinfaucet.uo1.net/`. Última consulta: 28/04/2022.

[29] Bitcoin testnet3 faucet, . URL `https://coinfaucet.eu/en/btc-testnet/`. Última consulta: 28/04/2022.

[30] 1ml. URL `https://1ml.com/testnet/`. Última consulta: 29/04/2022.

[31] Voltage. URL `https://voltage.cloud/`. Última consulta: 24/04/2022.

[32] Lightning labs. URL `https://lightning.engineering/`. Última consulta: 22/04/2022.

[33] Ferenc Béres, Istvan Andras Seres, and András A Benczúr. A cryptoeconomic traffic analysis of bitcoin's lightning network. *arXiv preprint arXiv:1911.09432*, 2019.

[34] Simina Brânzei, Erel Segal-Halevi, and Aviv Zohar. How to charge lightning. *arXiv preprint arXiv:1712.10222*, 2017.

[35] Starblocks. URL `https://starblocks.acinq.co/`. Última consulta: 29/04/2022.

[36] Dirección general de la industria y de la pyme. URL `http://www.ipyme.org/es-ES/UnionEuropea/UnionEuropea/PoliticaEuropea/Marco/Paginas/NuevaDefinicionPYME.aspx`. Última consulta: 09/05/2022.

[37] Ferenc Béres. Lntrafficsimulator. URL `https://github.com/ferencberes/LNTrafficSimulator`. Última consulta: 17/05/2022.