Construyendo los rieles de pago del futuro sobre LNP/BP: arquitectura de nodos para un sistema de pagos descentralizado

Sebastián Arriola Bethancourt

UVG | UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

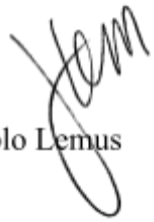
Construyendo los rieles de pago del futuro sobre LNP/BP: arquitectura de nodos para un sistema de pagos descentralizado

Trabajo de graduación presentado por Sebastián Arriola Bethancourt para optar al grado académico de Licenciado en Ingeniería en Ciencia de la Computación y Tecnologías de la Información

Guatemala,

2021

Vo.Bo.



Tribunal Examinador:

Juan Pablo Lemus

Primer Examinador Terna

MSc. Douglas Leonel Barrios

Segundo Examinador Terna

Ing. Roberto Chiroy

Fecha de Aprobación: Guatemala, 08 de diciembre de 2021

Prefacio

A principios de 2021, desconocía todo lo relacionado con la red Lightning de bitcoin. La motivación para el presente trabajo surgió debido a mi participación en la empresa IBEX Mercado y las noticias de El Salvador, donde Nayib Bukele anunció en una conferencia que El Salvador sería el primer país en adoptar el bitcoin como moneda de curso legal. La fecha en que entraría en vigor sería el 07 de septiembre del mismo año.

Los siguientes tres meses en IBEX presentaron un desafío para mí. Al mismo tiempo que debía aprender sobre la red Lightning, debíamos crear una solución que permitiera a los salvadoreños recibir pagos a través de esta. Por lo que decidí presentar en este proyecto el proceso de creación de una arquitectura de nodos necesaria para recibir y enviar pagos en la red Lightning. Esta es una tecnología que fue inventada en 2016 y esta sería la primera vez en recibir una cantidad masiva de usuarios de un día para otro. En el camino, encontramos varios desafíos relacionados a operar los nodos, ya que se requiere de un monitoreo constante sobre el balance de bitcoin que cada nodo tiene y sus conexiones con otros nodos de la red. Gracias al trabajo del equipo IBEX y a varios colaboradores externos, fuimos capaces de entregar una solución alternativa al Gobierno de El Salvador a tiempo.

Agradezco a mi asesor Juan Lemus, quien estuvo presente durante todo el proceso de desarrollo, cuyo conocimiento operacional de bitcoin fue indispensable para el éxito de la solución creada. Agradezco a mi familia por su apoyo incondicional, sin ellos nada de lo que he vivido habría sido igual o siquiera posible.

En algunos años, ya todo habrá cambiado y la red Lightning será mucho más madura. Espero que este trabajo sea de utilidad y que sirva como una guía para cualquier persona que quiera ser participante de la red Lightning de bitcoin.

Índice general

Índice de cuadros

Índice de figuras

Resumen

La red Lightning de bitcoin (LN) ha estado en desarrollo desde 2015 y su uso ha incrementado desde entonces. El Salvador fue el primer país en tomar bitcoin como una moneda de curso legal, lo que causó que el desarrollo de servicios de procesamiento de pagos fuera más necesario que nunca.

El objetivo principal de este trabajo fue diseñar e implementar una arquitectura de nodos para el procesamiento de pagos en la red Lightning de bitcoin. La arquitectura implementada sirvió como la infraestructura para servicios de procesamiento de pagos en bitcoin ofrecidos por IBEX Mercado.

Debido a la fase en la que se encuentra el desarrollo de la red Lightning es importante facilitar el crecimiento de la red, lo cual se pudo lograr por medio de documentación generada de primera mano, que describió los problemas comunes de operar nodos de Lightning y cómo resolverlos. Por otro lado, el éxito de la adopción global de bitcoin puede verse afectado por experiencias como las que tendrán los ciudadanos de El Salvador, por lo que fue importante proveer servicios alternativos a los oficiales.

La metodología utilizada fue diseñar una arquitectura de nodos, conceptualmente y en una maqueta, y luego fue implementada en los servicios en la nube de Amazon Web Services (AWS). Dicha arquitectura fue puesta en marcha y estuvo lista para el 7 de septiembre de 2021, día en que entró en vigencia la Ley bitcoin de El Salvador, y ha estado operando desde entonces, procesando más de 70,000 transacciones en su primer mes.

Palabras clave: bitcoin, red lightning, moneda de curso legal.

Abstract

The bitcoin Lightning Network (LN) has been in development since 2015 and its use has been increasing ever since. The Lightning Network had an important event this year, El Salvador was the first country to make bitcoin a legal tender, making development of bitcoin payment rails more needed than ever.

The main objective of this work is to design and implement a node architecture to process payments in the bitcoin Lightning Network. The implemented architecture will serve as the payment infrastructure for the services provided by the company IBEX Mercado.

Due to the stage at which the development of the Lightning Network is currently at, it is extremely important to facilitate the growth of the network, which can be aided by means of documentation describing common problems for node operators and how to solve them. The success of global bitcoin adoption can be affected by first hand experiences that citizens of El Salvador live. Providing alternatives to the official services provided by the Salvadorean government is of utmost importance.

The methodology of the presented work was to design a node architecture, first conceptually, and then implementing said architecture in the cloud provided by Amazon Web Services. This architecture was implemented and deployed to production on September 7th 2021, the same day in which bitcoin became a legal tender in El Salvador. The network has been processing payments ever since and more than 70,000 transactions were processed in its first month.

Keywords: bitcoin, lightning network, legal tender.

1. Introducción

Bitcoin fue creado por Satoshi Nakamoto en 2008 y es descrito como una moneda electrónica persona a persona (peer to peer), descentralizada (Nakamoto, 2008). En esencia, bitcoin es un protocolo que dicta las reglas necesarias para utilizar dicha moneda. Estas reglas fueron plasmadas en la implementación inicial o de referencia de bitcoin, llamada bitcoin core. Nos referimos al protocolo de bitcoin en corto como BP.

Bitcoin ha estado funcionando sin interrupciones desde enero de 2009, procesando millones de transacciones y alimentando su base de datos, el blockchain o cadena de bloques.

Uno de los compromisos que bitcoin hace para obtener seguridad es la velocidad de procesamiento de transacciones: bitcoin puede procesar 7 transacciones por segundo (TPS), lo cual es una debilidad cuando se compara con sistemas como VISA, que puede procesar alrededor de 65,000 TPS.

A lo largo de los años se han propuesto algunos cambios al protocolo de bitcoin que podrían permitir incrementar el número de TPS. Sin embargo, las propuestas se han rechazado porque le quitarían a bitcoin una de sus características más importantes: la descentralización. Actualmente cualquier persona con una computadora tan pequeña como una Raspberry Pi puede descargar el código fuente de bitcoin y tener su propio nodo, validando las transacciones y confiando únicamente en el código de bitcoin core (RaspberryPi Foundation, 2012). Dichas propuestas de cambiar el protocolo de bitcoin podrían hacer que los requerimientos de hardware puedan ser cumplidos únicamente por grandes entidades con mucho capital.

Una solución a la escalabilidad muy distinta, llamada Lightning Network (LN) fue propuesta por primera vez en 2016 y es descrita como una manera inteligente de usar el protocolo de bitcoin (Poon y Dryja, 2016). El LN es a su vez un protocolo, pero de Capa 2. Se le llama de capa 2, ya que construye sobre la capa 1 el protocolo de bitcoin. El principal objetivo del LN es incrementar la cantidad de transacciones que se pueden procesar, en teoría una transacción puede ser procesada en su totalidad tan rápido como pueda ser enviada por internet.

El presente trabajo profesional se enfoca en el diseño e implementación de una arquitectura de nodos y canales de pago construida en la red Lightning, que servirá de base para una solución de capa 3. El desarrollo presentado será la infraestructura de procesamiento de pagos en bitcoin de la empresa IBEX, de ahora en adelante “la empresa”.

2. Objetivos

2.1 Generales

- Diseñar e implementar una arquitectura de nodos y canales de pago que permita el procesamiento de pagos.
- Automatizar la administración de los balances de los canales de pago.

2.2 Específicos

- Determinar la cantidad de nodos necesarios.
- Determinar los canales de pago necesarios.
- Determinar la capacidad adecuada de los canales de pago.
- Definir políticas de rebalanceo.
- Aplicar las políticas de rebalanceo utilizando herramientas automatizadas.

3. Justificación

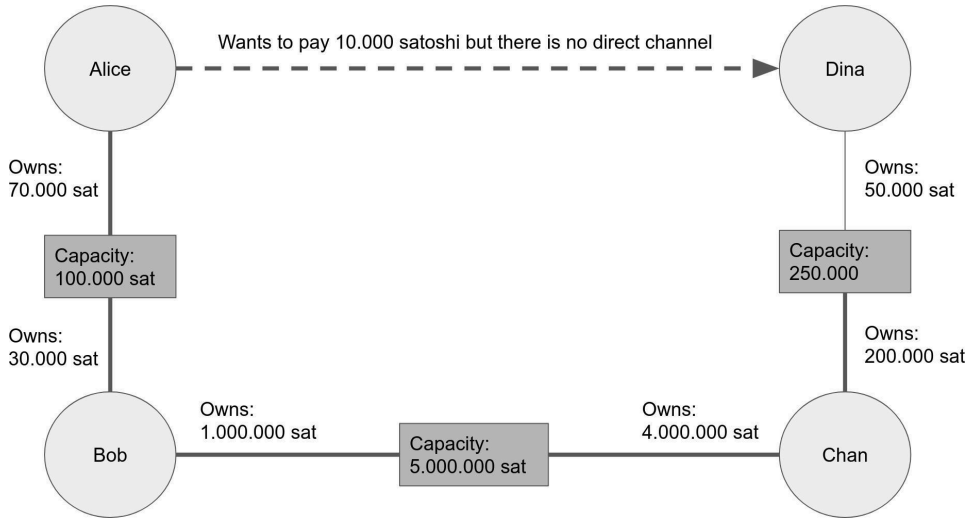
Bitcoin y especialmente el Lightning Network (LN) han tenido un gran desarrollo en los últimos años. Estamos en un punto en donde una persona con conocimientos técnicos generales puede instalar una implementación de un nodo de Lightning e iniciar a procesar pagos (Lightning Labs, 2015). La complejidad está, por ahora, en la administración de uno o varios nodos, lo cual incluye temas como la liquidez del nodo y de sus canales de pago, lo cual es clave para garantizar la procesabilidad de pagos.

Es importante aclarar que la unidad básica del bitcoin es el Satoshi, nombrado así en honor a Satoshi Nakamoto. Los Satoshis se pueden comparar con los centavos en las monedas tradicionales. Un bitcoin equivale a cien millones de Satoshis.

Con motivo de comprender el problema de la liquidez, se presenta la figura 1, en donde se observa una arquitectura con 4 actores: Alice, Bob, Chan y Dina, y varios canales de pago. Un canal de pago cuenta con 2 balances, comúnmente llamados local y remoto, y una capacidad (suma de ambos balances).

Para que Alice pueda enviar un pago de 10,000 Satoshis a Dina, cada canal de pago debe contar con al menos 10,000 Satoshis de liquidez saliente. Por ejemplo, Bob debe tener 10,000 Satoshis localmente en su canal con Chan, y Chan debe tener al menos 10,000 Satoshis localmente en su canal con Dina. La liquidez en los canales de pago es un tema activo en la investigación de la red Lightning.

Figura 1. Canales de Pago y Liquidez



Nota. Elaboración propia.

En este trabajo se diseñará e implementará una arquitectura que pueda servir de referencia en el futuro para personas que deseen procesar pagos en el LN. La arquitectura será diseñada tomando en cuenta las necesidades de tener control sobre los balances de los canales, y también se tomará en cuenta las técnicas y herramientas de rebalanceo de canales que existen actualmente. Dicha arquitectura funcionará como la infraestructura para los servicios que la empresa ofrecerá iniciando en septiembre del 2021.

4. Marco teórico

A. bitcoin

Bitcoin (Nakamoto, 2008) fue creado por Satoshi Nakamoto en el año 2008 y es descrito por él mismo como una moneda electrónica peer to peer, descentralizada. En esencia, bitcoin es un protocolo que dicta las reglas necesarias para utilizar dicha moneda electrónica. Estas reglas fueron plasmadas en la implementación inicial o de referencia de bitcoin, llamada bitcoin Core. Nos referimos al protocolo de bitcoin en corto como BP.

B. bitcoin core

Es la implementación oficial (bitcoin Core, 2009) del protocolo de bitcoin. Esta implementación fue lanzada al público en el año 2009 por Satoshi Nakamoto y ha sido desarrollada posteriormente por un grupo de desarrolladores. Es una implementación de código abierto, por lo que cualquier persona puede ingresar al repositorio y leer el código. En esencia, esta implementación contiene y aplica las reglas del protocolo de bitcoin, y por lo tanto, cualquier persona o entidad que ejecute esta implementación, forma parte de la red de bitcoin.

C. Satoshi (Unidad básica)

Un bitcoin es equivalente a 100, 000, 000 (cien millones) de Satoshis. El Satoshi es la unidad básica de bitcoin, como el centavo lo es para una moneda fiduciaria. En la implementación del protocolo de bitcoin, bitcoin Core, los montos se representan por medio de números enteros, en Satoshis. Los montos se muestran generalmente a los usuarios en unidades de bitcoin, por ejemplo: 0.1 BTC = 10,000,000 de Satoshis.

D. Transacción (bitcoin)

Una transacción en el protocolo de bitcoin es una estructura que representa la transferencia de valor entre los participantes del sistema de bitcoin. Tiene dos partes fundamentales: entradas y salidas. En una entrada se hace referencia a una transacción anterior, la cual cuenta con satoshis disponibles para gastar. Una salida especifica la dirección destino que recibirá los satoshis.

E. Blockchain o Cadena de Bloques

Todas las transacciones de bitcoin se almacenan en el blockchain o cadena de bloques, empaquetadas en unidades llamadas bloques. Cada nuevo bloque que se agrega contiene una referencia criptográfica al bloque anterior, formando una cadena. El blockchain logra la propiedad de la inmutabilidad por medio de las referencias criptográficas.

Otra perspectiva del Blockchain es su uso en una red descentralizada. Tomando como ejemplo la red de bitcoin, cada nodo en la red contiene una copia del blockchain, lo cual significa que contiene una copia del historial entero de transacciones.

F. Bloque

Un bloque es la unidad básica que se almacena en una cadena de bloques. En el protocolo de bitcoin, un bloque tiene la estructura que se observa en las tablas 1 y 2 (Antonopoulos, 2017).

Cuadro 1. Estructura de bloque

Tamaño	**Campo**	**Descripción**
4 bytes	Tamaño de Bloque	Tamaño del bloque en bytes.
80 bytes	Cabecera de Bloque	Ver cuadro 2.
1-9 bytes (VarInt)	Cantidad de Transacciones	Número de transacciones en este bloque.
Variable	Transacciones	Transacciones almacenadas en este bloque.

Nota. Elaboración propia

Cuadro 2. Estructura de cabecera de bloque

Tamaño	**Campo**	**Descripción**
4 bytes	Versión	Número de versión que se utiliza para identificar las actualizaciones al protocolo.
32 bytes	Hash del bloque anterior	Una referencia al hash del bloque anterior.
32 bytes	Raíz del árbol Merkle	El hash del nodo raíz del árbol Merkle de las transacciones de este bloque.
4 bytes	Tiempo	Tiempo aproximado de creación del bloque.
4 bytes	Objetivo de dificultad	La dificultad del algoritmo de prueba de trabajo para este bloque.
4 bytes	Nonce	Un valor utilizado para el algoritmo de prueba de trabajo.

Nota. Elaboración propia.

G. Función Hash

Es una función matemática que mapea información de un tamaño arbitrario a un mensaje de tamaño fijo, comúnmente referido como el hash de la información. Las funciones hash criptográficas son de una vía, denominadas así porque no es viable obtener la información original que produjo un determinado hash. Una función hash criptogŕafica cuenta con varias características (Wikipedia contributors, 2021):

- Determinística: la misma información de entrada siempre resultará en el mismo hash.
- Es rápida para computar el hash de cualquier mensaje de entrada.

- Es inviable generar un mensaje a partir de un hash.
- Un pequeño cambio en el mensaje de entrada produce un cambio vasto en el hash.

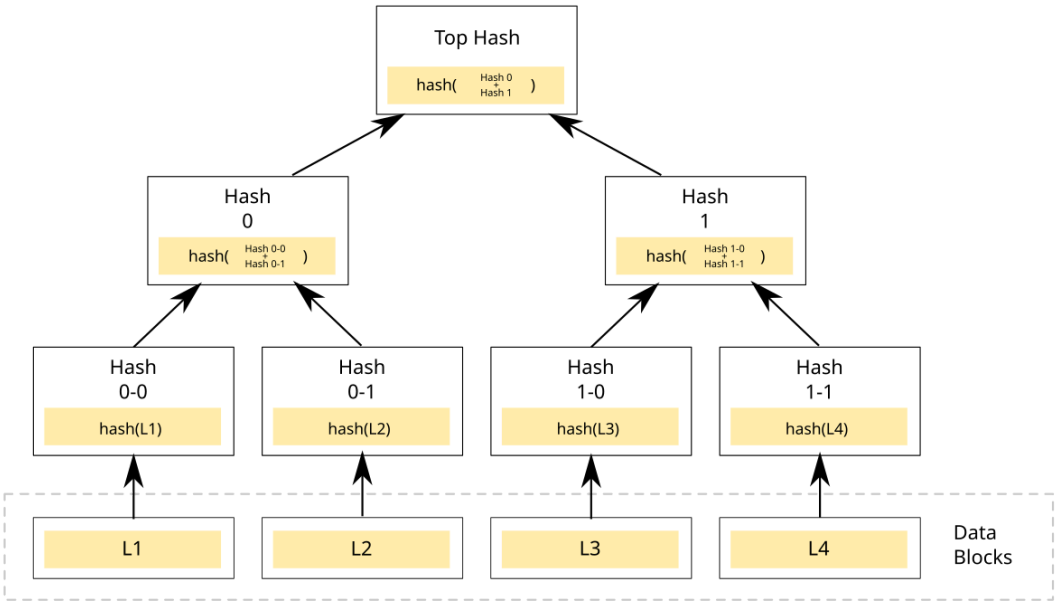
H. SHA-256

Secure Hash Algorithm 256, SHA-256 en corto, es parte de la familia de funciones SHA2. Es una función hash criptográfica que produce un hash de 256 bits. La implementación de bitcoin utiliza la función SHA-256.

I. Árbol Merkle

Una estructura de árbol en donde cada nodo del nivel más inferior contiene el hash de un mensaje, y todos los demás nodos contienen el hash de sus nodos hijo. Los árboles Merkle proveen una manera eficiente de verificar los contenidos de estructuras de datos de gran tamaño. En bitcoin Core, se utiliza esta estructura para verificar que una transacción pertenece a un bloque (Wikipedia contributors, 2021).

Figura 2. Árbol de Merkle



Nota. Adaptada de autor.

J. Prueba de Trabajo

El sistema de prueba de trabajo se basa en encontrar un valor cuyo hash inicie con una cantidad de ceros específica. El problema de encontrar dicho valor crece exponencialmente con la cantidad de ceros requeridos. En el protocolo de bitcoin se utiliza la función Hash SHA256 para computar un hash que cumpla con la dificultad especificada por la red en un momento determinado.

K. Lightning Network

El Lightning Network, es un sistema descentralizado que permite un alto volumen de pagos sin delegar la custodia de fondos a un tercero (Lightning Network, 2018). Antonopoulos et al. (2022) describen el LN como una manera muy inteligente de utilizar el protocolo de bitcoin.

Una de las principales críticas hacia bitcoin que impiden su uso como moneda es que permite un número bajo de transacciones por segundo (TPS), alrededor de 7 TPS. La LN fue propuesta para resolver el bajo número de TPS de bitcoin.

La LN es a su vez un protocolo de Capa 2. Se le llama de capa 2 ya que construye sobre la capa 1 el protocolo de bitcoin. El principal objetivo es incrementar la cantidad de transacciones que se pueden procesar y en teoría una transacción de Lightning puede ser procesada en su totalidad tan rápido como pueda ser enviada por el internet.

L. Lightning Network Daemon

LND (Lightning Labs, 2015), en corto, es una de las varias implementaciones del protocolo de LN. LND está escrito utilizando el lenguaje de programación Go y está disponible para diversas arquitecturas. Se puede instalar por medio de un binario precompilado, o también se puede compilar a partir del código fuente. Esta implementación es la más utilizada y por lo tanto es la que cuenta con mayor documentación y comunidad.

M. Nodos de Lightning

Un nodo de Lightning es un equipo de cómputo que cuenta con una implementación de software del LP ejecutándose. Los nodos de Lightning forman una red par a par o P2P.

N. Red Peer to Peer

Es un tipo de arquitectura de sistemas en la que no existe una entidad central que controle la comunicación. Cada entidad dentro del sistema se denomina un peer o un participante. Generalmente todos los participantes tienen las mismas capacidades dentro del sistema (Wikipedia contributors, 2021).

O. Canal de Pago

Un canal de pago representa una conexión entre dos nodos en la red Lightning. Un canal de pago es un contrato multi firma entre dos pares. La principal funcionalidad de la red Lightning se provee a través de los canales de pago, ya que permiten enviar una cantidad casi infinita de pagos sin la necesidad de publicar cada pago en el Blockchain.

P. Submarine Swaps

Un swap en corto, permite convertir bitcoin on chain a offchain, y vice versa. El término on chain se refiere al bitcoin que vive en la cadena de bloques de bitcoin. Se denomina bitcoin offchain al bitcoin que está invertido en canales de pago en la red Lightning. Por ejemplo, si se desea obtener bitcoin offchain, se puede utilizar un servicio en el cual se deposite bitcoin on chain, utilizando una transacción de bitcoin, y el servicio envíe la cantidad de bitcoin por medio de la red Lightning hacia el destino que se haya especificado (Lightning Labs, 2016).

Q. Watchtower

Un watchtower es un servicio que provee protección en caso de que un nodo tenga problemas de conexión al internet por períodos prolongados de tiempo. Al estar sin conexión, un nodo A corre peligro de que otro nodo B con el que el nodo A tiene un canal de pago publique una transacción que representa un estado antiguo de dicho canal de pago. Este ataque se conoce como execution fork.

Un watchtower vigila el blockchain, leyendo cada transacción publicada. Si se identifica

una transacción que representa un estado antiguo de un canal de pago, el watchtower publica al blockchain la transacción que representa el estado actual de dicho canal de pago. Esta transacción cierra el canal de pago, distribuyendo los fondos a cada par de manera correcta (ION, 2021).

R. Ride the Lightning

Aplicación web cuya funcionalidad es ser un dashboard para un operador de nodos de LND. Ofrece una interfaz gráfica que permite realizar muchas de las operaciones rutinarias de manera sencilla, por medio de formularios y botones. Esta aplicación debe ser instalada en un servidor, y se debe especificar los nodos a los cuales se desea conectar (Ride the Lightning, 2018).

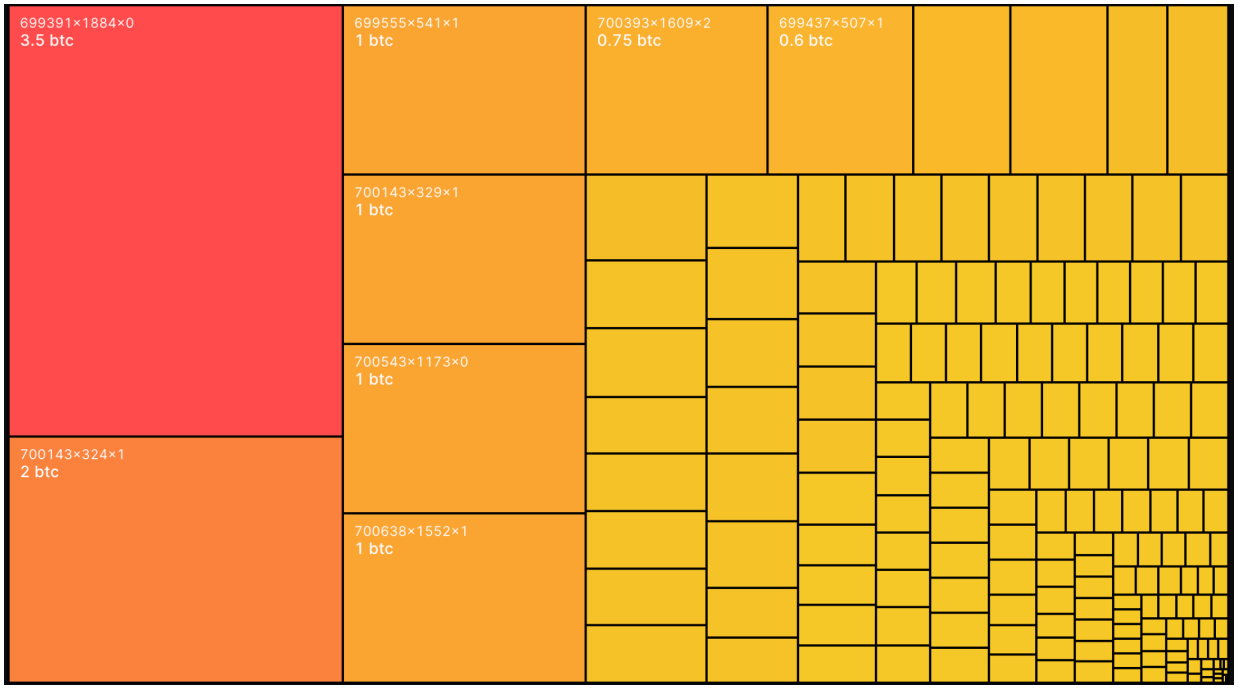
S. 1ml.com

Sitio web cuya funcionalidad principal es ofrecer un servicio de exploración simple de la red de Lightning. Ofrece estadísticas cómo: cantidad de nodos en la red, cantidad de canales públicos en la red, capacidad de la red. También ofrece un servicio de búsqueda de nodos y canales de pago. Se puede buscar un nodo por medio de su alias, llave pública, o dirección IP. También permite a un operador de nodos personalizar la página de sus nodos (1ML, 2016).

T. amboss.space

Sitio web cuya funcionalidad principal es ofrecer un servicio de exploración de la red de Lightning. A diferencia de 1ml.com, amboss ofrece estadísticas más complejas y también visualizaciones gráficas, como la que se muestra en la figura 3 (amboss, 2022).

Figura 3. Visualización de canales de pago en amboss.space



U. Amazon Web Services

AWS, en corto, provee una infinidad de servicios en la nube, para facilitar el desarrollo escalable. Entre sus servicios más destacados se encuentra EC2, el cual provee servidores virtuales, RDS para bases de datos, y ECS para orquestación de contenedores. Muchos de los servicios de AWS son elásticos, lo cual permite al servicio crecer su número de recursos alocados en momentos de alta demanda, y liberar recursos cuando la demanda disminuye (Amazon Web Services, 2002).

V. VPC

Una Virtual Private Cloud, VPC en corto, es una red virtual, lógicamente aislada de los demás recursos de AWS. Muchos de los servicios de AWS requieren ser creados dentro de una VPC. Una VPC permite configurar la conectividad y seguridad. Por ejemplo, para crear una base de datos, primero debemos crear una VPC o utilizar una existente. Luego, por medio de reglas de seguridad se puede definir si queremos que la base de datos sea alcanzable por una entidad externa públicamente, o si se desea que la base de datos sea privada, disponible únicamente para otros recursos dentro de la misma VPC (Amazon Web Services, 2002).

W. Grupo de Seguridad

Un grupo de seguridad, o security group en inglés, actúa como una pared de fuego virtual para controlar el tráfico entrante y saliente hacia un recurso. Un grupo de seguridad permite configurar de manera aislada las reglas del tráfico que puede entrar hacia un recurso y salir del mismo (Amazon Web Services, 2002).

X. Código QR

Un código QR es un tipo de código de barras en forma de matriz; es un código de barras en 2 dimensiones. Su uso se popularizó debido a su eficiencia en rapidez de lectura y cantidad de información que pueda almacenar comparado al código de barras tradicional (Wikipedia contributors, 2021).

5. Metodología

Los pasos que se detallan a continuación fueron ejecutados para cumplir el objetivo de este trabajo: diseñar e implementar una arquitectura de nodos sobre la red Lightning de bitcoin, que permita procesar pagos.

A. Primer prototipo

Se realizó un primer diseño de la red propuesta en una maqueta. Los objetivos principales de este primer prototipo fueron:

- definir tipos de nodos
- definir cantidad de nodos internos
- definir canales de pago entre nodos internos
- definir capacidad de canales de pago

Actualmente no existe una arquitectura estándar de nodos, ni una nomenclatura de tipos de nodos, debido a que la red Lightning es reciente y los participantes están experimentando y construyendo sus propias arquitecturas de acuerdo a sus necesidades. Las necesidades principales que se tomaron en cuenta fueron funcionalidad y disponibilidad.

Para definir los tipos de nodos, principalmente se pensó en que se necesita un tipo de nodo que esté conectado con la red pública, y otro tipo de nodo que esté detrás del primer tipo de nodo.

El tipo de nodo que estará conectado masivamente a la red pública se denominó “distribuidor”. Este nodo tendrá cientos de canales de pago. Incluir un tipo de nodo distribuidor en la arquitectura permite escalabilidad de la misma, ya que si se desea agregar un nodo a la arquitectura, únicamente se debe conectar con un nodo distribuidor para contar con la capacidad de enviar y recibir pagos a toda la red.

El otro tipo de nodo se denominó “mercader” y este tipo de nodo estará conectado siempre a un nodo distribuidor. El hecho de que esté conectado a un nodo distribuidor le provee a este tipo de nodo la capacidad de enviar y recibir pagos hacia toda la red. Además, en un canal de pago entre un nodo mercader y un nodo distribuidor se tiene control total de ambos extremos del canal, lo que permite rebalancear libremente el canal de pago sin riesgo de perder valor.

Esto es algo importante en el contexto de la empresa ya que la habilidad de rebalancear un canal de pago libremente permite reducir la cantidad de Satoshis que se invierten en dicho canal, reduciendo el costo para la empresa. La razón por la que se pueden crear canales de menor capacidad es porque los balances de dichos canales se pueden rebalancear cada cierto tiempo, libremente.

En cuanto a la cantidad de nodos, se necesitan al menos uno de cada tipo de nodo. Para contar con redundancia adicional, se decidió que se contaría con 3 nodos, 2 de tipo distribuidor y 1 de tipo mercader. De igual manera, se decidió contar con un canal de pago entre cada par de nodos, para tener redundancia en caso de que algún nodo falle.

En la maqueta se representan nodos, canales de pago, y la capacidad de dichos canales de pago. Para los canales de pago se utilizó cable metálico, y se colocaron unas bolitas de plástico en dichos cables (como en un ábaco).

B. Segundo prototipo

Se realizó una implementación en software de la arquitectura planteada en el primer prototipo. Esta implementación incluyó varias partes, desde la creación de una red privada dentro de AWS, hasta la apertura de canales de pago entre nuestros nodos y nodos externos.

Se comenzó por la creación de una VPC dentro de AWS, dedicada a este proyecto. La decisión de crear una VPC dedicada nos permitió aislar todos los recursos que más adelante se crearon dentro de esta red privada, y elegir con qué otras entidades compartir dichos recursos.

Una vez se contó con la red privada, se procedió a montar la dependencia básica del resto del proyecto: un nodo completo de bitcoin. Para esto, y debido al corto tiempo con el que se contó, se decidió utilizar un servicio de subscripción dentro de AWS, el cual provee de una instancia con la base de datos de bitcoin al día. La otra opción era montar desde cero el nodo de bitcoin, pero la descarga inicial de bloques lleva alrededor de 2 días y es necesario reservar una instancia con más recursos ya que este proceso es demandante de memoria RAM. El servicio de suscripción estuvo listo en minutos.

El siguiente paso fue la instalación del primer nodo de LND. Se utilizó la guía oficial que provee Lightning Labs (Lightning Labs, 2015). Se eligió una instancia con el sistema operativo

Ubuntu Server 20.04, ya que la guía oficial es precisamente para este sistema operativo y de tal manera se pueden eliminar posibles errores de compatibilidad y concentrarnos en lo que importa para este proyecto.

Una vez instalado este primer nodo, se procedió a inicializar el programa LND. La inicialización de LND incluye la creación de una llave privada, ya que LND cuenta con la funcionalidad de una billetera de bitcoin básica. La llave privada fue presentada por LND como una frase, de 12 palabras. Dicha frase fue apuntada en papel y almacenada en los archivos de IBEX. Luego de la configuración inicial de LND, se procedió a generar una nueva dirección de bitcoin. Se envió bitcoin a dicha dirección, para iniciar con la apertura de los primeros canales de pago.

La primera prueba que se realizó fue abrir un canal de pago con uno de los nodos de Wallet of Satoshi. La apertura del canal de pago se realizó por medio del comando openchannel, en donde se especifica la llave pública del nodo con el cual se desea abrir el canal, la dirección ip y el puerto de dicho nodo, y la cantidad de Satoshis que se desean colocar en el canal de pago. Este comando devuelve el ID de la transacción de apertura de canal, la cual se siguió usando la herramienta Mempool.space (Mempool, 2018).

Luego de esperar las 6 confirmaciones de la apertura del canal, se procedió a generar las primeras transacciones de Lightning. Esto lo realizamos por medio de la terminal de comandos, utilizando la herramienta lncli. Por medio del comando addinvoice, se obtuvo un requerimiento de pago, el cual se envió a un teléfono celular por medio de un servicio de mensajería. Este requerimiento de pago fue copiado al portapapeles en el teléfono celular, y luego se cargó en Wallet of Satoshi. La billetera reconoció el requerimiento de pago y procedió a mostrar una confirmación de pago, la cual se aceptó. La billetera mostró un mensaje de éxito indicando que el requerimiento de pago había sido pagado correctamente. Se procedió a revisar en el nodo LND el estado del requerimiento de pago, por medio del comando lookupinvoice, y se comprobó que el estado del pago se mostraba como SETTLED, o concluido.

Luego se realizó la prueba inversa: generar un requerimiento de pago desde Wallet of Satoshi y pagarlo desde el nodo LND. Para pagar un requerimiento de pago, se utilizó el comando payinvoice.

Una vez se tuvo el primer nodo corriendo y se realizaron las primeras pruebas, se procedió

a la instalación del segundo nodo, en su propia instancia de EC2. Este nodo se denominó internamente como: “D0”, abreviación de distribuidor cero. Internamente, los nodos distribuidores son un tipo de nodo cuyo objetivo principal es contar con una gran cantidad de canales de pago. El primer canal de pago que se aperturó en este nodo fue hacia el nodo inicial, o nodo IBEX.

El tercer nodo que se instaló fue el nodo “M0” o mercader cero. El objetivo principal de este nodo es ser la infraestructura de pago para múltiples mercaderes que utilizarán un servicio de capa 3 que también se desarrolló en la empresa.

El cuarto nodo que se instaló se denominó “A0” o API cero. Su objetivo principal es ser la infraestructura de pago para entidades terceras que consumirán los servicios de la empresa por medio de una interfaz de programación.

Una vez se contó con los cuatro nodos instalados y funcionando, se procedió a conectar los nodos entre sí, abriendo canales de pago entre el nodo D0 y M0, el D0 y A0, el M0 y A0, y entre D0 e IBEX.

El objetivo del nodo distribuidor es formar conexiones clave con otros nodos externos. Se procedió a investigar cuáles eran algunos de los nodos más importantes de la red pública. Para esto, se utilizaron dos servicios web, 1ml y amboss.space (1ML, 2016) (amboss, 2022). Se buscó que fueran los nodos de las billeteras de Lightning más utilizadas, así como algunos de los nodos más conectados de la red. Que un nodo esté bien conectado significa que tiene canales de pago abiertos con otros bien conectados. Algunos con los que se abrieron canales fueron los siguientes:

- Wallet of Satoshi (1ML, 2019)
- BCash_Is_Trash (1ML, 2019)
- ACINQ (1ML, 2019)
- OpenNode (1ML, 2019)

Al finalizar este segundo prototipo, se contó con una arquitectura interna de 4 nodos y una conectividad decente hacia la red pública, permitiendo recibir y enviar transacciones a las principales billeteras de Lightning.

C. Tercer prototipo - primeras pruebas de pagos

Al tener la arquitectura implementada, el enfoque pasó a probar la arquitectura. Las funcionalidades principales incluyeron: recibir pagos, enviar pagos, rebalancear manualmente los canales de pago y rebalancear los canales por medio de un script.

Para probar la funcionalidad de recibir pagos, se creó un requerimiento de pago en uno de nuestros nodos, utilizando la herramienta de línea de comando lncli. El comando para crear requerimientos de pago es addinvoice y se debe especificar el monto de dicho pago. Al obtener el bolt11 del requerimiento de pago, se generó un código QR. El QR fue escaneado utilizando la billetera Wallet of Satoshi y fue pagado.

Para la segunda prueba se realizó el proceso inverso. Se generó un requerimiento de pago por 5 Satoshis desde Wallet of Satoshi, y este fue pagado desde el mismo nodo utilizado en la primera prueba. La billetera utilizada permite exportar el requerimiento de pago creado como una cadena de caracteres en el formato bolt11. Esta cadena de caracteres fue ingresada en el comando payinvoice de la herramienta lncli para pagar el requerimiento de pago. Los resultados de estas pruebas de envío y recepción de pagos se presentan en la sección de Resultados.

D. Rebalanceo Manual de Canales

Luego de realizar varias pruebas de recibir pagos, el balance de un canal se ubica en su gran mayoría en el lado local del nodo. Esto quiere decir que el canal tiene poca capacidad de recibir pagos, pero mucha capacidad para enviar pagos. Debido a que se desea tener la capacidad tanto de recibir como de enviar pagos, se aprovechó el estado de algunos canales para realizar las primeras pruebas de rebalanceo de canales. El rebalanceo de canales consiste en realizar algún tipo de transacción para mover balance de lado local a remoto, o vice versa.

Para rebalancear un canal que cuenta con mucho balance local, se puede realizar un Loop Out. Un loop out es un tipo de submarine swap que convierte fondos de Lightning a bitcoin on chain. Se conoce a este tipo de swap como offchain a on chain. El proceso se basa en realizar un pago por medio de la red Lightning al servicio Loop. Una vez el servicio Loop detecte que recibió nuestro pago, procede a enviar bitcoin a nuestra dirección, por defecto es la dirección del nodo que solicitó el loop out. El comando para realizar un loop out es:

```
loop out --amt <monto>
```

La segunda prueba de rebalanceo que se realizó tuvo el objetivo de colocar balance del lado local de un canal de pago. Con el objetivo de seguir experimentando con la herramienta Loop (Lightning Labs, 2019), se optó por realizar un Loop In. Un loop in es un tipo de submarine swap que convierte fondos on chain a fondos offchain. El proceso se basa en realizar un pago por medio de la red bitcoin a la dirección del servicio Loop. Cuando el servicio Loop detecte que recibió el pago, procederá a enviar hacia nuestro nodo un pago por medio de la red Lightning.

Para solicitar un loop in, se debe especificar 2 parámetros: la llave pública del nodo por donde se desea recibir el pago, y el monto (en Satoshis). La herramienta Loop se encarga del resto de la complejidad, la cual debe solicitar una dirección on chain y realizar la transacción desde el nodo donde se está haciendo el loop in. El comando necesario para solicitar un loop in es:

```
loop in --last_hop
02f72978d40efeffca537139ad6ac9f09970c000a2dbc0d7aa55a71327c4577a80 --amt
20000000
```

La opción de last hop especifica la llave pública del nodo por donde se desea recibir el loop in, y la opción amt especifica el monto del loop in. En este ejemplo se solicitó un loop in por 20 millones de Satoshi. El comando devolverá la dirección de bitcoin a donde se realizó la transacción, como se observa en la figura 4. Una vez la transacción cuente con 2 confirmaciones, el servicio de Loop procederá a enviar el pago a través de la red Lighthing, hacia nuestro nodo, y especificará que el último salto se realice a través del canal que se especificó en el comando. Este canal verá su balance remoto reducido en 20 millones de Satoshis, y verá su balance local incrementado en 20 millones de Satoshis. Por medio de este loop in, se logró incrementar la liquidez de salida de nuestro nodo.

Figura 4 - Transacción on chain para solicitar loop in

Nota. Elaboración propia.

También se realizó un loop out. El loop out es el proceso inverso de un loop in. Nuestro nodo realiza un pago por medio de el LN hacia el nodo del servicio Loop. Una vez el servicio Loop reciba nuestro pago, realiza una transacción on chain a la dirección de nuestro nodo. El monto de esta transacción es el monto especificado menos la comisión que cobra el servicio de Loop. Por medio de un loop out, se le otorga al nodo una mayor liquidez entrante. El comando para iniciar un loop out es:

```
loop out --channel <idCanal> --amt <monto>
```

La opción idCanal puede especificar un solo canal, o también se puede especificar una lista de canales separadas por coma. La opción amt especifica el monto del loop out.

E. Script de rebalanceo y primera prueba.

Con el objetivo de descargar el trabajo de rebalancear los canales entre nuestros nodos, se desarrolló un script para automatizar el rebalanceo de canales. El script debe ejecutarse en un nodo, y mantendrá la distribución del balance del canal que le sea especificado por medio de parámetros. Es importante aclarar que este script debe ejecutarse únicamente entre nuestros propios nodos, ya que al enviar un pago para rebalancear un canal, efectivamente se está enviando valor hacia el otro nodo par. Por ejemplo, si se realizara un rebalanceo con un nodo externo, por medio del envío de un pago, nuestro nodo tendría menos valor. Al rebalancear nuestros canales internos, no hay problema de mover valor entre nuestros nodos, ya que dicho valor siempre está bajo nuestro control.

El script acepta parámetros:

- lower_bound: el límite inferior aceptado de la distribución del balance del canal. Un valor menor al especificado disparará el proceso de rebalanceo del canal.

- target_score: el objetivo de la distribución de balance del canal. El proceso de rebalanceo

calculará el monto del pago a ser enviado para que la distribución de balance del canal después de enviar el pago sea la especificada por medio de este parámetro.

Por ejemplo, asumiendo que el balance del canal es de 600,000 Satoshis del lado local, y 400,000 Satoshis del lado remoto, el lower_bound es de 0.5 y el target_score es de 0.75, el script detectará que la distribución del balance del canal (400,000 / 1,000,000 = 0.4) es menor a la deseada (0.5) e iniciará un proceso de rebalanceo. Este proceso calculará el monto necesario para llegar a una distribución de 0.75, equivalente a 750,000 Satoshis del lado remoto del canal. Luego, enviará un pago por este monto, 350,000 Satoshis, al nodo B. Luego de enviar el pago, el balance del canal será de 250,000 Satoshis del lado local y 750,000 Satoshis del lado remoto.

La primera prueba del script se realizó en un ambiente local que contó con dos nodos. El nodo A es el nodo en donde se ejecuta el script. El nodo B representa otro nodo de nuestra arquitectura. Los nodos están conectados por medio de un canal con capacidad de 1 millón de Satoshis. Los parámetros fueron:

lower_bound: 0.50

target_score: 0.75

La prueba simuló la recepción de 100 pagos a través del canal, en donde el nodo B envió un pago con un monto variable cada segundo. El script de rebalanceo revisa el balance del canal al recibir cada pago, y si la distribución es menor a la deseada, se envía un pago al nodo B por un monto calculado, como se explicó en el ejemplo. Durante la prueba, ocurrieron 10 eventos de rebalanceo. Los resultados de la prueba se presentan en la sección de resultados.

F. Compactación de base de datos

Cada nodo de LND almacena su estado en una base de datos local, llamada BBoltDB (etcd, 2013). Es recomendado configurar LND para que cada vez que se inicie el programa se realice una compactación de esta base de datos, ya que una base de datos sin compactar impacta negativamente el desempeño de LND.

En el caso de la arquitectura planteada en este trabajo, se esperaba que cada nodo iba a contar con una cantidad considerable de canales, por lo que fue indispensable realizar esta

configuración desde un inicio. La configuración se realiza por medio del archivo de configuración de LND, en el cual se debe colocar las siguientes opciones:

- db.bolt.auto-compact = true
    - Especifica que la base de datos se compacte en cada inicio del programa.
- db.bolt.auto-compact-min-age = 0
    - Especifica la edad mínima de la base de datos que debe pasar antes de que la base de datos se compacte. Un valor de cero deshabilita esta opción, la cual en conjunto con la opción anterior logra que la base de datos se compacte en cada reinicio, sin importar la edad.

Dependiendo del tamaño de la base de datos, el proceso de compactación puede tomar hasta 15 minutos, por lo que debe ser tomado en cuenta a la hora de reiniciar un nodo. En el caso de la arquitectura de la empresa , hemos reiniciado los nodos al menos una vez a la semana, y el proceso de compactación ha tomado alrededor de 2 minutos.

G. Otros parámetros importantes de LND

- minchansize: indica el tamaño mínimo de canal de pago, en Satoshis, que este nodo aceptará. Este parámetro es importante para evitar que se abran canales de pago hacia nuestro nodo con una capacidad muy baja para el contexto del negocio. La configuración que se realizó para los nodos de la empresa fue:
    - nodos d0 y m0: 5 millones de Satoshis.
    - nodo a0: 50 millones de Satoshis.

H. Respaldo de archivo channel.backup

En el caso de un fallo catastrófico de LND, existe un archivo que ayuda a que no se pierdan los fondos. Este archivo se llama channel.backup, y mantiene el estado más actual de los canales de pago del nodo, es decir, el archivo se actualiza cada vez que se abre o cierra un canal de pago.

Por medio de una guía desarrollada por Alex Bosworth (Bosworth, 2019), uno de los desarrolladores de LND, se configuró un proceso para que cada vez que el archivo sea actualizado por LND, se realice una copia de dicho archivo.

I. Proceso de reinicio de nodos

El proceso de reinicio es bastante sencillo pero debe siempre hacerse de la manera indicada para evitar la corrupción de los archivos importantes de LND. Los pasos para reiniciar un nodo son:

- detener LND por medio del comando lncli stop
- comprobar que el proceso está detenido. Esto puede hacerse de varias maneras, una de ellas es por medio del comando ps aux | grep lnd
- arrancar LND de nuevo por medio del comando lnd
- si se está compactando la base de datos, esperar al menos 1 minuto.
- ingresar el comando lncli unlock e ingresar la contraseña para desbloquear la billetera. (la contraseña se obtiene al instalar lnd y no es parte del enfoque de este documento)
- si se obtiene un error, es posible que el proceso de compactación aún esté en ejecución, esperar un minuto y volver a intentar el paso 5.

J. Configuración de Watchtower

En el caso de que uno de nuestros nodos esté fuera de servicio (offline), existe la posibilidad de que un par malicioso intente publicar una transacción que refleje un estado antiguo de un canal de pago con nuestro nodo. Un par malicioso está motivado a publicar un estado antiguo si tiene más balance en su lado local, ya que al publicar dicho estado, el par malicioso cierra el canal de pago y se queda con los fondos locales que cuente dicho estado antiguo.

Para evitar este escenario, LND implementó un concepto que llamaron Watchtowers. Un watchtower es un nodo de LND que vigila a uno o más nodos de LND. Al momento de que un par malicioso intenta publicar un estado antiguo, el watchtower se dará cuenta y utilizará el mecanismo de penalización de la red Lightning, el cual cierra el canal de pago y devuelve la totalidad de los fondos al nodo inocente. El nodo que intentó hacer trampa recibe cero fondos.

La configuración de un watchtower implica al menos dos nodos:

- el nodo servidor, es quien vigila a los nodos clientes
- uno o más nodos clientes, que especifican su nodo servidor.

Se utilizó una guía (*Set up a Watchtower and a Client on the Lightning Network*, 2019) para realizar esta configuración. Uno de nuestros nodos fue configurado como servidor, por medio de habilitar la opción watchtower.active=1 en el archivo de configuración lnd.conf. Los otros nodos se configuraron como nodos clientes, por medio de la opción wtclient.active=1. Para especificar que un cliente quiere que otro nodo sea su servidor watchtower se utiliza el comando:

```
lncli wtclient add <pubkey>@<host>:9911
```

- pubkey especifica la llave pública del nodo watchtower (nota: es una llave pública distinta a la llave pública del nodo).
- host especifica la dirección IP o dirección Onion del nodo.
- 9911 es el puerto utilizado por defecto

6. Resultados

A. Arquitectura de nodos

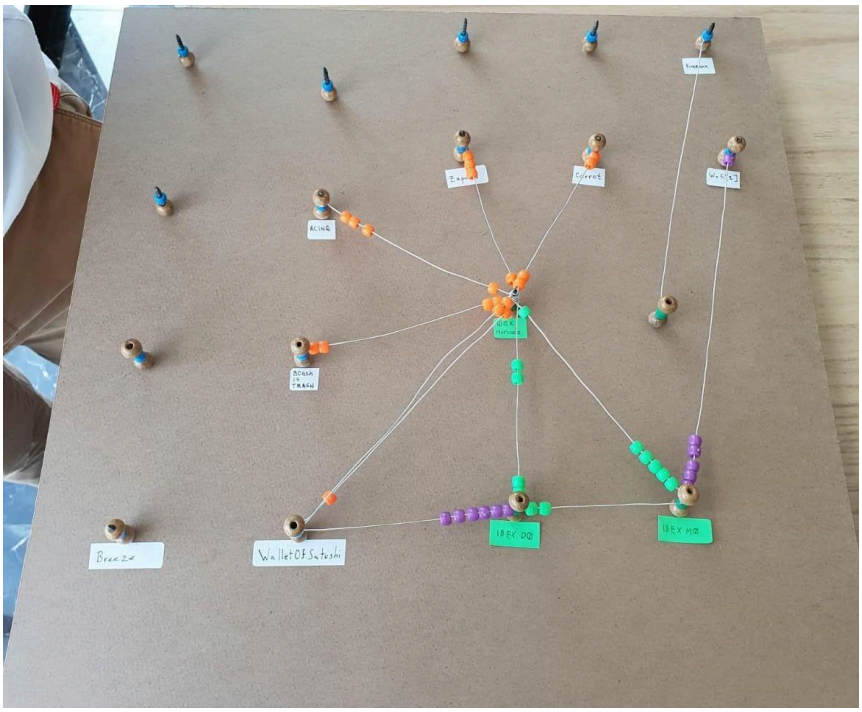
La empresa IBEX comenzó a ofrecer sus servicios como procesador de pagos a inicios de septiembre, principalmente debido a la ley bitcoin de El Salvador (Asamblea Legislativa, 2021). La arquitectura de nodos que se diseñó e implementó en este trabajo tenía el objetivo de ser la infraestructura que soportaría el procesamiento de pagos de múltiples empresas salvadoreñas.

En el primer prototipo se decidió que la arquitectura contaría con dos tipos de nodo:

- distribuidor;
- mercader.

El diseño planteado en el primer prototipo se implementó en la maqueta que se observa en la figura 5.

Figura 5. Maqueta de arquitectura de nodos



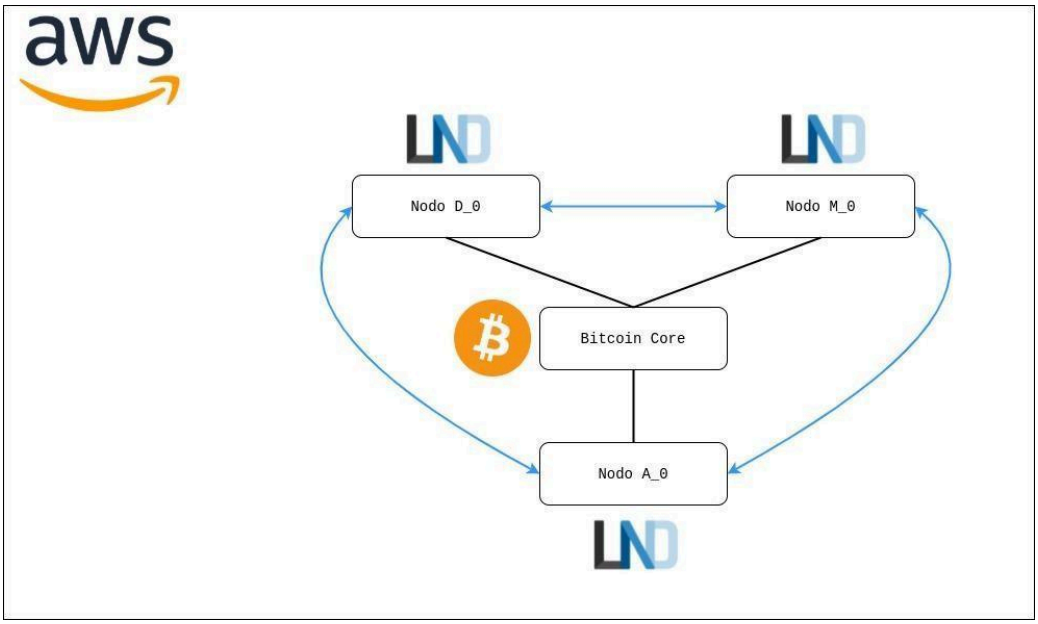
Nota. Elaboración propia

En este prototipo se decidió que la arquitectura contaría con cuatro nodos:

- IBEX: nodo destinado para realizar pruebas varias
- D0: nodo de tipo distribuidor, destinado a formar conexiones con muchos otros nodos
- M0: nodo de tipo mercader.
- A0: nodo de tipo mercader, destinado a proveer servicios por medio de una interfaz de programación.

En el prototipo 2 se realizó la implementación inicial de la arquitectura diseñada en el prototipo 1. Dicha implementación fue realizada sobre los servicios en la nube de Amazon Web Services (Amazon Web Services, 2002). Al finalizar la implementación inicial, se contó con 3 nodos de LND funcionales. La arquitectura implementada en esta segunda iteración es la que se puede observar en la figura 6. Solamente se muestran los nodos d0, m0 y a0.

Figura 6. Arquitectura implementada



Nota. Elaboración propia.

El tercer prototipo se centró en realizar pruebas sobre la arquitectura implementada. Se comprobó que se podía recibir y enviar pagos.

En la figura 7 se observa un requerimiento de pago codificado en un QR, la cual es la manera estándar en la que se representan requerimientos de pago para que los teléfonos móviles puedan escanear dicho código QR.

Figura 7. Bolt11 codificado como QR

Nota. Elaboración propia.

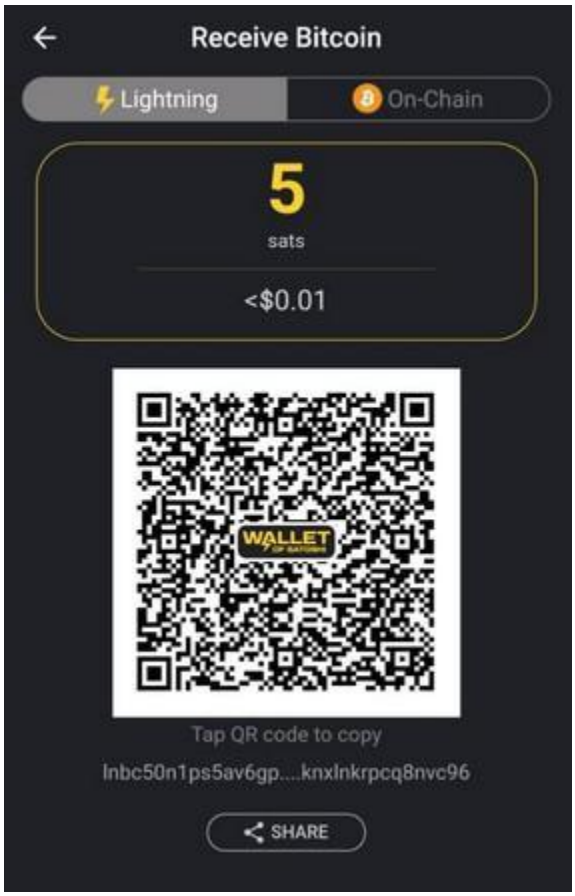
En la figura 8 se muestra el detalle de un requerimiento de pago generado en uno de los nodos. Este requerimiento de pago fue pagado desde la billetera Wallet of Satoshi y se puede observar su estado de SETTLED o finalizado en la parte inferior de la figura.

Figura 8. Detalle de requerimiento de pago

Nota. Elaboración propia.

En la figura 9 se muestra un requerimiento de pago generado desde Wallet of Satoshi. Dicho requerimiento de pago fue pagado desde uno de los nodos. El detalle del pago se observa en la figura 10.

Figura 9. Requerimiento de pago generado en Wallet of Satoshi



Nota. Elaboración propia.

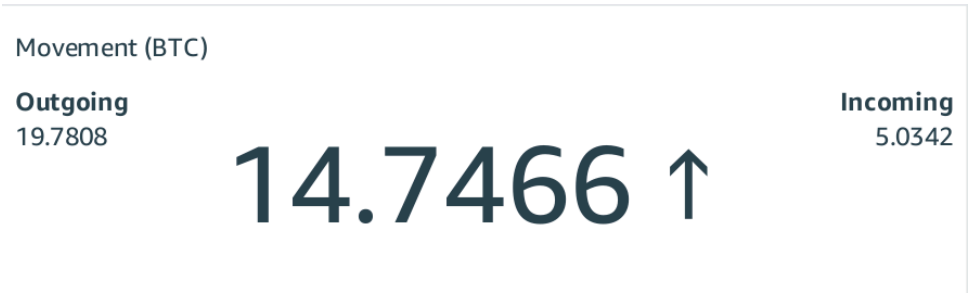
Figura 10. Pago realizado desde nodo LND

```
Payment hash: ca27d9fb09c98139b96b5682a321c8173316c029b36a6f04784cd9e693460ab2
Description: Wallet of Satoshi
Amount (in satoshis): 5
Fee limit (in satoshis): 5
Destination: 035e4ff418fc8b5554c5d9eea66396c227bd429a3251c8cbc711002ba215bfc226
Confirm payment (yes/no): yes
+------------+--------------+--------------+--------------+-----+----------+--------------------+----------------------+
| HTLC_STATE | ATTEMPT_TIME | RESOLVE_TIME | RECEIVER_AMT | FEE | TIMELOCK | CHAN_OUT           | ROUTE                |
+------------+--------------+--------------+--------------+-----+----------+--------------------+----------------------+
| SUCCEEDED  |        0.061 |        0.268 | 5            | 0   |   702149 | 769032517476745217 | WalletOfSatoshi.com |
+------------+--------------+--------------+--------------+-----+----------+--------------------+----------------------+
Amount + fee:   5 + 0 sat
Payment hash:   ca27d9fb09c98139b96b5682a321c8173316c029b36a6f04784cd9e693460ab2
Payment status: SUCCEEDED, preimage: 59894c16e8b5baf35e09e37a2b4708f19bfb58f1dafc9eb1a2b4020c78575282
```

Nota. Elaboración propia

La arquitectura se inició a utilizar en producción el 7 de septiembre del 2021, fecha en que la ley bitcoin entró en vigor en El Salvador. En el primer mes de uso, se han procesado más de 70,000 transacciones, con un valor total mayor a 24 BTC.

Figura 11. Valor entrante y saliente de transacciones



Nota. Elaboración propia.

Figura 12. Estadísticas de transacciones

Nota. Elaboración propia.

Figura 13. Cantidad y volumen de transacciones por día

Nota. Elaboración propia.

B. Script de Rebalanceo

La prueba realizada sobre el script de rebalanceo demostró que el script tiene la capacidad de mantener el balance de un canal de pago de acuerdo a los paŕametros especificados.

En la figura 16 se observa que el script realizó 10 eventos de rebalanceo, calculando el monto que debía enviar al nodo B para dejar el balance del canal de acuerdo a los parámetros especificados. En la figura 15 se muestra el estado del canal de pago luego de realizar la prueba.

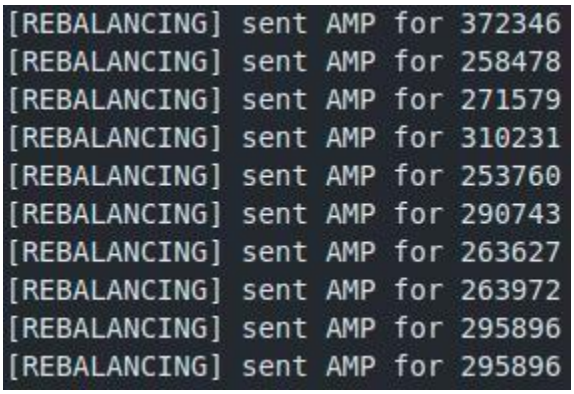
Figura 14. Detalle de canal de pago antes de iniciar la prueba


```
"active": true,
"remote_pubkey": "03e9fc738b2d7e025c15392d914815dd2240e105c00cfc9e710d36772498948c7a",
"channel_point": "05b16e1da425fe4e1960c9d3151e734accc8f42e186e3988fae92c355cd9a16b:0",
"chan_id": "550855325581312",
"capacity": "1000000",
"local_balance": "246530",
"remote_balance": "750000",
"commit_fee": "2810",
"commit_weight": "1116",
"fee_per_kw": "2500",
"unsettled_balance": "0",
"total_satoshis_sent": "641868517",
"total_satoshis_received": "642115047",
"num_updates": "8694",
```


Figura 15. Detalle de canal de pago al finalizar la prueba


```
"active": true,
"remote_pubkey": "03e9fc738b2d7e025c15392d914815dd2240e105c00cfc9e710d36772498948c7a",
"channel_point": "05b16e1da425fe4e1960c9d3151e734accc8f42e186e3988fae92c355cd9a16b:0",
"chan_id": "550855325581312",
"capacity": "1000000",
"local_balance": "373016",
"remote_balance": "623514",
"commit_fee": "2810",
"commit_weight": "1116",
"fee_per_kw": "2500",
"unsettled_balance": "0",
"total_satoshis_sent": "643179933",
"total_satoshis_received": "643552949",
"num_updates": "8817",
```

Figura 16. Detalle de eventos de rebalanceo



7. Análisis de resultados

Luego de tener un mes de datos, vemos que hay más volumen hacia afuera del sistema, por lo que la importancia de la liquidez saliente es importante. Se planteó una arquitectura de nodos en donde un nodo mercader estaría conectado a uno o varios nodos distribuidores. Este planteamiento cambió a medida que la red inició operaciones en producción. La cantidad de canales de los nodos distribuidores incrementó en pocas horas, pasando de menos de 20 canales por nodo a más de 100.

Después de unos días de estar operando, nos dimos cuenta que es bastante común que cada nodo tenga un pequeño número de canales inactivos. Por esta razón, y para agregar redundancia y resiliencia, se permitió que un nodo mercader formara conexiones con nodos de la red pública, y no únicamente con nuestros nodos.

Uno de los temas más críticos de operar una red de nodos de Lightning es la liquidez. Notamos el papel clave que juegan los servicios de Submarine Swaps, especialmente el servicio Trustless Loop. Este servicio se encuentra entre las herramientas que utilizamos diariamente para colocar balance en donde los nodos lo requieren. Cabe mencionar que es importante contar con fondos on chain locales en los nodos para utilizar el servicio Loop.

Otro tema crítico se trata de la base de datos que utiliza LND, BBolt. LND sigue en desarrollo beta, por lo que, de momento, es difícil cambiar esta base de datos, sin embargo ya existe un esfuerzo para lograrlo (Lightning Labs, 2021). La base de datos crece e impacta negativamente el desempeño de LND, por lo que el proceso de compactación se vuelve crítico de ejecutar al menos una vez por semana. Ya que el proceso de compactación puede tomar varios minutos, es de suma importancia notar que una aplicación de capa 3 que dependa de nuestra red no podrá utilizar nuestros servicios mientras el proceso de compactación esté ejecutándose. De momento se está realizando la compactación de las bases de datos de nuestros nodos una vez por semana y a horarios en la madrugada para limitar el impacto en las aplicaciones de capa 3.

Esta lección nos llevó a pensar la siguiente fase del proyecto, la cual es diseñar e implementar un balanceador de carga de nodos de LND. Básicamente se tendrá un servicio intermedio que pueda hablar con varios nodos, permitiéndonos quitar un nodo momentáneamente mientras el proceso de compactación y cualquier otro mantenimiento esté ejecutándose.

8. Conclusiones

La red Lightning de bitcoin se encuentra en constante desarrollo, por lo que es muy importante generar conocimiento a través de la experimentación. En este trabajo profesional, se tuvo la oportunidad de diseñar, implementar y probar una arquitectura de nodos que se encuentra operando principalmente en El Salvador.

El objetivo principal fue diseñar e implementar una arquitectura de nodos que permitiera el procesamiento de pagos en la red Lightning de bitcoin.

En la fase de diseño se decidió que se contaría con dos tipos de nodos: mercader y distribuidor. Los tipos de nodos permiten más control sobre lo que se expone a la red pública. La arquitectura implementada contó con tres nodos, lo cual eliminó el problema de un único punto de falla, y agregó resiliencia a la red.

Tener más de un nodo también permite crear canales de pago en los cuales se es dueño de ambos extremos del canal, lo cual permite mover el balance de dicho canal a discreción, sin riesgo alguno de pérdida de fondos.

Durante la implementación de la arquitectura, se consideró el uso de algunas precauciones de seguridad como respaldos automatizados y configuración de watchtowers.

El uso de la arquitectura en producción nos enseñó algunas lecciones valiosas,

- Es de vital importancia realizar el proceso de compactación de la base de datos de LND con una frecuencia proporcional a la cantidad de transacciones.
- También se identificó que una de las principales preocupaciones al operar nodos es mantener el balance de los canales de pago de acuerdo a la necesidad de la red.
- Se identificaron algunas herramientas manuales de rebalanceo de canales y se exploró una implementación inicial de un script de rebalanceo interno.

9. Recomendaciones

El script de rebalanceo de canales que se implementó revisa el estado del canal de pago cada vez que se recibe un pago. Este proceso puede que no sea lo más escalable a la hora de que un canal reciba varios pagos por minuto; el proceso de revisar el balance del canal podría representar más cómputo del necesario. Se recomienda investigar otras opciones de revisión del canal de pago, una de ellas podría ser revisar el balance cada cierto tiempo.

En cuanto al propósito de la arquitectura de nodos, si se utilizara para soportar múltiples aplicaciones de capa 3, se debe pensar en un servicio intermedio proxy que remueva el problema de un único punto de falla, es decir, que la aplicación de capa 3 no dependa de un único nodo de Lightning. El servicio proxy permitirá escalar la arquitectura de nodos horizontalmente, agregando y removiendo nodos bajo demanda y, también, permitirá realizar el mantenimiento de un nodo, sin impactar las aplicaciones que dependan de la arquitectura.

10. Referencias

Amazon Web Services. (2002). *Amazon Web Services*. AWS. https://aws.amazon.com/

Amazon Web Services. (2002). *Security groups: Inbound and Outbound Rules*. Amazon Web Services. https://docs.aws.amazon.com/quicksight/latest/user/vpc-security-groups.html

Amazon Web Services. (2002). *Virtual Private Cloud*. Amazon Web Services. https://aws.amazon.com/vpc/

Amboss. (2022). *Find Your Edge*. amboss. https://amboss.space/

Antonopoulos, A. M. (2017). *Mastering bitcoin: Programming the Open Blockchain*. O'Reilly.

Antonopoulos, A. M., Osuntokun, O., y Pickhardt, R. (2022). *Mastering the Lightning Network: A Second Layer Blockchain Protocol for Instant Bitcoin Payments*. O'Reilly.

Asamblea Legislativa. (2021, junio 9). *El Salvador, primer país del mundo en reconocer al bitcoin como moneda de curso legal*. Asamblea Legislativa. https://www.asamblea.gob.sv/node/11282

Bitcoin Core Developers. (2009). *bitcoin core*. Bitcoin Core. https://bitcoin.org/en/bitcoin-core/

Bitcoin Core Developers. (2009). *Running a Full Node*. Bitcoin Core. https://bitcoin.org/en/full-node#linux-instructions

Bosworth, A. (2019). *LND backup script for channel.backup using inotify*. Github. https://gist.github.com/alexbosworth/2c5e185aedbdac45a03655b709e255a3

Breez Technology. (2018). *Breez: bitcoin in Every App*. breez. https://breez.technology/

ETCD. (2013). *bbolt: An embedded key/value database for Go*. Github. https://github.com/etcd-io/bbolt

ION. (2021). *Watchtowers*. ION Lightning Network Wiki. https://wiki.ion.radar.tech/tech/research/watchtowers

Lightning Labs. (2015). *Get Started*. Builder's Guide. https://docs.lightning.engineering/lightning-network-tools/lnd/run-lnd

Lightning Labs. (2015, octubre 24). *Lightning Network Daemon*. Github. https://github.com/lightningnetwork/lnd

Lightning Labs. (2016). *Understanding Submarine Swaps*. https://docs.lightning.engineering/the-lightning-network/lightning-overview/understanding-submarine-swaps

Lightning Labs. (2019). *Lightning Loop*. Github. https://github.com/lightninglabs/loop

Lightning Labs. (2021). *kvdb: add postgres*. Github. https://github.com/lightningnetwork/lnd/pull/5366

Lightning Network. (2018). *The bitcoin Lightning Network*. Lightning Network Summary. https://lightning.network/lightning-network-summary.pdf

Mempool. (2018). *Mempool*. Mempool bitcoin Explorer. https://mempool.space/

Nakamoto, S. (2008, octubre 31). *bitcoin: A Peer to Peer Electronic Cash System*. bitcoin. https://bitcoin.org/bitcoin.pdf

nginx. (2004). nginx. https://nginx.org/

1ML. (2016). *Lightning Network Search and Analysis Engine*. 1ML. https://1ml.com/

1ML. (2019). *Node: ACINQ*. 1ML. https://1ml.com/node/03864ef025fde8fb587d989186ce6a4a186895ee44a926bfc370e2c366597a3f8f

1ML. (2019). *Node: BCash_Is_Trash*. 1ML. https://1ml.com/node/0298f6074a454a1f5345cb2a7c6f9fce206cd0bf675d177cdbf0ca7508dd28852f

1ML. (2019). *Node: OpenNode*. 1ML. https://1ml.com/node/03abf6f44c355dec0d5aa155bdbdd6e0c8fefe318eff402de65c6eb2e1be55dc3e

1ML. (2019). *Node: WalletOfSatoshi*. 1ML. https://1ml.com/node/035e4ff418fc8b5554c5d9eea66396c227bd429a3251c8cbc711002ba215bfc226

Papadis, N., y Tassiulas, L. (2020, diciembre 21). *Blockchain-Based Payment Channel Networks: Challenges and Recent Advances*. IEEE Explore. https://ieeexplore.ieee.org/document/9300150

Pickhardt, R., y Richter, S. (2021, julio 13). *Optimally Reliable & Cheap Payment Flows*

on the Lightning Network. Arxiv. https://arxiv.org/pdf/2107.05322

Poon, J., y Dryja, T. (2016, enero 14). *The bitcoin Lightning Network: Scalable Off-Chain Instant Payments*. lightning-network. https://lightning.network/lightning-network-paper.pdf

poweredbyibex. (2021). *IBEX*. Real-Time Settlement for the Next Era of Finance. https://www.ibexmercado.com/

RaspberryPi Foundation. (2012). *Raspberry Pi*. Raspberry Pi. https://www.raspberrypi.org/

Ride the Lightning. (2018). *RTL*. Github. https://github.com/Ride-The-Lightning/RTL

Set up a watchtower and a client on the lightning network. (2019). Lightning Node Management. https://www.lightningnode.info/advanced-tools/watchtower

Visa. (2018). *Visa Fact Sheet*. Visa UK. https://www.visa.co.uk/dam/VCOM/download/corporate/media/visanet-technology/aboutvisafactsheet.pdf

Wallet of Satoshi. (2018). *Wallet of Satoshi: The world's simplest bitcoin lightning wallet*. Wallet of Satoshi. https://github.com/Ride-The-Lightning/RTL

Wikipedia contributors. (2021, septiembre 22). *Peer-to-peer*. Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Peer-to-peer

Wikipedia contributors. (2021, septiembre 29). *Merkle tree*. Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Merkle_tree

Wikipedia contributors. (2021, octubre 10). *Cryptographic hash function*. Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Cryptographic_hash_function

Wikipedia contributors. (2021, octubre 30). *QR code*. Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/QR_code

11. Anexos

El script de rebalanceo que se realizó para este trabajo profesional se puede consultar en el siguiente enlace: https://github.com/sebdeveloper6952/py-lnd-rebalancing