(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 986 314**

(51) Int. Cl.:

G06Q 30/06 (2013.01)

G06Q 20/38 (2012.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **01.12.2017 E 17425121 (5)**

(97) Fecha y número de publicación de la concesión europea: **24.07.2024 EP 3493141**

(54) Título: **Comunicaciones y secuenciación de blockchain**

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
11.11.2024

(73) Titular/es:

QUANT NETWORK LTD. (100.0%)
20-22 Wenlock Road
London N1 7GU, GB

(72) Inventor/es:

VERDIAN, GILBERT;
PATERSON, COLIN;
MONDELLI, GAETANO y
TASCA, PAOLO

(74) Agente/Representante:

VALLEJO LÓPEZ, Juan Pedro

DESCRIPCIÓN

Comunicaciones y secuenciación de blockchain

Campo técnico

Esta divulgación se refiere a métodos, sistemas, programas informáticos y SDK para mantener un registro de una secuencia de transacciones que aparecen en una o más blockchains. También se refiere a métodos, sistemas, programas informáticos y SDK relacionados con comunicaciones de blockchain.

Antecedentes

Las tecnologías de blockchain tienen el potencial para una enorme gama de usos, ofreciendo una forma en la que los datos pueden almacenarse de forma segura y fiable sin la necesidad de autoridades centrales de autenticación de datos. En consecuencia, las tecnologías de blockchain pueden permitir una gran cantidad de beneficios técnicos en una amplia gama de sectores, que incluyen, por ejemplo, transparencia de datos mejorada, almacenamiento de datos más robusto que no depende de una entidad de almacenamiento en particular, seguridad de datos mejorada y resistencia mejorada al fraude, y ya ha encontrado usos en sectores tan diversos como la distribución de energía eléctrica y el almacenamiento en la nube punto a punto (*peer-to-peer*) (P2P).

Blockchain se conceptualizó por primera vez en 2009 en un libro blanco titulado "Bitcoin: A Peer-to-Peer Electronic Cash System", publicado a nombre de Satoshi Nakamoto. Se puede encontrar una copia del libro blanco en: https://bitcoin.org/bitcoin.pdf. El libro blanco proponía una versión P2P de efectivo electrónico denominada "Bitcoin" que utilizaba lo que desde entonces se ha denominado una blockchain para registrar transferencias P2P de Bitcoin.

Una blockchain es un ledger digital distribuido y descentralizado, mediante el cual se puede almacenar un registro de datos efectivamente inmutable. Está descentralizado porque no requiere necesariamente una autoridad central para añadir datos a la blockchain o para mantener la integridad de la blockchain. Los datos pueden añadirse a una blockchain mediante una entidad que difunde una "transacción" a una red de nodos que participan en la blockchain, en donde la transacción comprende los datos que deben añadirse a la blockchain y elementos criptográficos que se ajustan al protocolo de la blockchain. Cada uno de los nodos que participan en la blockchain puede comprobar a continuación la validez de la transacción y, si se aprueba, añadir la transacción a un nuevo bloque en el que están trabajando.

Cada uno de los nodos también puede añadir al nuevo bloque en el que están trabajando otras transacciones nuevas que se han difundido a la red de nodos. Después de un periodo de tiempo, un nodo publicará su nuevo bloque a los otros nodos en la red, incluyendo el nuevo bloque las diversas transacciones nuevas así como datos criptográficos que unen el nuevo bloque al bloque precedente en la blockchain. Atando criptográficamente bloques consecutivos juntos de esta manera, si un bloque en la blockchain se altera en cualquier momento posterior (por ejemplo, por alguien que desea cambiar de manera fraudulenta los datos almacenados en la blockchain), puede ser fácilmente detectable.

Después de que un nuevo bloque ha sido publicado por un nodo, los otros nodos en la red pueden comprobar el contenido del nuevo bloque y, si se acepta, comenzar a trabajar en su siguiente bloque nuevo, que se vinculará de nuevo criptográficamente a su bloque precedente. Por lo tanto, puede verse que la blockchain crecerá cada vez que se publique y acepte un nuevo bloque, estando cada nuevo bloque aceptado criptográficamente vinculado a su bloque precedente. Este proceso de nodos que comprueban un nuevo bloque publicado y lo aceptan en la blockchain a menudo se denomina mecanismo de "consenso".

En el libro blanco, "Bitcoin: A Peer-to-Peer Electronic Cash System", se propone un protocolo particular para Bitcoin. El protocolo cubre aspectos tales como el formato y los tipos de información que deben incluirse en una transacción, el formato y los tipos de información que deben incluirse en nuevos bloques, las técnicas criptográficas y las recetas que deben usarse para transacciones y nuevos bloques, y el tipo particular de mecanismo de consenso que se utiliza (por ejemplo, en Bitcoin, el mecanismo de consenso utiliza una prueba de trabajo basada en hash en cada bloque de la blockchain, que también es el vínculo criptográfico entre cada bloque consecutivo). Si una transacción de difusión no se ajusta correctamente al protocolo, no se incluirá en ningún bloque nuevo. Análogamente, si un nuevo bloque no se ajusta adecuadamente al protocolo, la red de nodos no lo aceptará en la blockchain.

Aunque las blockchains, o tecnologías de ledger distribuido (DLT), como también se denominan, se propusieron por primera vez para habilitar la criptomoneda Bitcoin, posteriormente han encontrado muchos otros usos. Al principio, se lanzaron nuevas criptomonedas, tales como Ethereum, Ripple y Namecoin, cada una operando en sus propias blockchains particulares, utilizando sus propios protocolos particulares. Desde entonces, se ha reconocido que la tecnología de blockchain puede explotarse mucho más ampliamente que las criptomonedas y puede utilizarse en cualquier contexto en el que sea útil un registro de datos efectivamente inmutable. Esto ha llevado a una rápida expansión en el número de diferentes blockchains actualmente existentes, cada una adaptada a su propio caso de uso particular, y se anticipa que el número de diferentes blockchains continuará aumentando a medida que la

tecnología gana más impulso y reconocimiento.

AUGUSTANA DIGITAL COMMONS ET AL: "Augustana College Data Insertion in Bitcoin's Blockchain Augustana Digital Commons Citation Data Insertion in Bitcoin's Blockchain",

SCRIPT. 4. TRANSACTION MALLEABILITY. 5. OP RETURN. 6. COINBASE. 7. FREE SPEECH, [en línea] 31 de julio de 2017 divulga métodos para insertar datos arbitrarios en la blockchain de Bitcoin.

ROMAN MATZUTT ET AL: "POSTER: I don't want that content! On the risks of exploiting Bitcoin's blockchain as a content store", COMPUTER AND COMMUNICATIONS SECURITY, ACM, 2 PENN PLAZA, SUITE 701 NEW YORK NY 10121-0701 USA, [en línea] 24 de octubre de 2016, páginas 1769-177 muestra para Bitcoin que las blockchains ofrecen potencialmente múltiples formas de almacenar contenidos (maliciosos e ilegales) que, una vez almacenados, no pueden ser eliminados y son replicados por cada usuario participante.

Anónimo: "Whitepaper - Resources - Cosmos Network", 1 de enero de 2017 describe una red de muchas blockchains independientes, denominadas zonas. Las zonas están alimentadas por Tendermint BFT. La primera zona en Cosmos se llama Cosmos Hub. El concentrador y las zonas de la red Cosmos se comunican entre sí a través de un protocolo de comunicación entre blockchains (IBC, por sus siglas en inglés), una especie de UDP o TCP virtual para blockchains.

Gavin Wood: "POLKADOT: VISION FOR HETEROGENEOUS MULTI-CHAIN FRAMEWORK", 24 de julio de 2017, páginas 1-21, describe un marco heterogéneo de múltiples cadenas dispuesto para proporcionar compatibilidad hacia atrás con una o más redes preexistentes tales como Ethereum.

Sumario

En un primer aspecto de la presente divulgación, se proporciona un método implementado por ordenador definido en la reivindicación 1.

Manteniendo un registro de una secuencia de transacciones, pueden mitigarse las vulnerabilidades de seguridad y estabilidad asociadas con mensajes desordenados, mejorando de este modo la seguridad y estabilidad de los sistemas que utilizan la una o más blockchains. Este beneficio técnico es particularmente evidente cuando se registra la secuencia de transacciones que aparecen a través de dos o más (una pluralidad) de blockchains, puesto que la secuenciación de transacciones a través de múltiples blockchains es particularmente problemática y está abierta a problemas de seguridad y fraude.

Identificar la pluralidad de transacciones puede comprender leer el contenido de cada nuevo bloque añadido a cada una de la una o más blockchains.

Identificar la pluralidad de transacciones puede comprender además comparar las transacciones en cada nuevo bloque frente a un criterio de relevancia, en donde la pluralidad identificada de transacciones comprende transacciones que cumplen el criterio de relevancia. Mediante la comparación contra el criterio de relevancia, solo es necesario identificar transacciones correspondientes en el almacén de datos, lo que puede mejorar la velocidad con la que puede revisarse el almacén de datos en el futuro y reducir el tamaño del almacén de datos requerido.

Preferiblemente, almacenar el registro de la pluralidad de transacciones en el almacén de datos comprende, cada vez que se identifica una transacción en un nuevo bloque, almacenar un registro de esa transacción identificada en el almacén de datos, en donde la secuencia relativa de la pluralidad de transacciones es la secuencia relativa en la que se identificó cada una de la pluralidad de transacciones.

Almacenando, en al menos una de la una o más blockchains, un registro del conjunto de verificación más reciente de la pluralidad de conjuntos de verificación consecutivos, puede mejorarse la integridad y el no repudio del registro, aumentando de este modo la seguridad y la estabilidad del método.

El método puede comprender además comprobar que un registro del conjunto de verificación anterior en la pluralidad de conjuntos de verificación está presente en la al menos una de la una o más blockchains, y si un registro del conjunto de verificación anterior no está presente en la al menos una de la una o más blockchains: reproducir un registro del conjunto de verificación anterior en la al menos una de la una o más blockchains. De esta manera, pueden identificarse rápidamente bifurcaciones anteriores en las blockchains, manteniendo aún la fiabilidad de la secuencia de transacciones registrada.

El registro del conjunto de verificación anterior puede reproducirse en la al menos una de la una o más blockchains antes de almacenar, en al menos una de la una o más blockchains, el registro del conjunto de verificación más reciente.

El método puede comprender además: si un registro del conjunto de verificación anterior no está presente en la al

menos una de la una o más blockchains: comprobar que cada una de las transacciones identificadas en el conjunto de verificación anterior está presente en su blockchain correspondiente de la una o más blockchains, y si una o más de las transacciones identificadas en el conjunto de verificación anterior no están presentes en sus blockchains correspondientes de la una o más blockchains: reproducir la una o más transacciones en sus blockchains correspondientes.

La una o más transacciones pueden reproducirse en sus blockchains correspondientes antes de almacenar, en al menos una de la una o más blockchains, el registro del conjunto de verificación más reciente.

El identificador del conjunto de verificación del conjunto de verificación más reciente puede comprender un hash de los contenidos del conjunto de verificación más reciente.

Almacenar el registro del conjunto de verificación más reciente puede comprender difundir una transacción a al menos una blockchain de la una o más blockchains para su inclusión en la al menos una blockchain, en donde la transacción de difusión comprende el identificador único del conjunto de verificación más reciente.

El registro de la pluralidad de transacciones puede comprender una pluralidad de identificadores de transacción que se determinan cada uno basándose al menos en parte en los contenidos de las respectivas transacciones.

La pluralidad de identificadores de transacción puede comprender una pluralidad de hashes del contenido de la pluralidad de transacciones.

El registro de la pluralidad de transacciones puede comprender el contenido de al menos algunas de la pluralidad de transacciones.

En un segundo aspecto de la presente divulgación, se proporciona un sistema que comprende: uno o más procesadores; y una memoria que almacena un programa de software, en donde el programa de software, cuando se ejecuta por el uno o más procesadores, hace que el uno o más procesadores realicen un método de acuerdo con cualquier reivindicación anterior.

En un tercer aspecto de la presente divulgación, se proporcionan uno o más programas informáticos que, cuando se ejecutan mediante uno o más procesadores, hacen que el uno o más procesadores lleven a cabo un método de acuerdo con el primer aspecto.

En un cuarto aspecto de la presente divulgación, se proporciona un kit de desarrollo de software (SDK) que comprende un conjunto de herramientas de desarrollo de software para desarrollar el uno o más programas informáticos del tercer aspecto.

En un quinto aspecto de la presente divulgación, se proporciona un método para proporcionar interoperabilidad entre una capa funcional de la aplicación y una pluralidad de blockchains diferentes, en donde la capa funcional de la aplicación comprende operaciones funcionales que hacen uso de una o más de la pluralidad de blockchains diferentes, comprendiendo el método una capa de comunicaciones de blockchain: recibir una solicitud de blockchain desde la capa funcional de la aplicación, en donde la solicitud de blockchain se refiere a una operación de blockchain a realizar en una o más blockchains de la pluralidad de blockchains diferentes; e interconectar con la una o más blockchains de acuerdo con los protocolos de la una o más blockchains para realizar la operación de blockchain. Por lo tanto, la capa funcional de la aplicación puede desacoplarse de los protocolos de blockchain, permitiendo que las aplicaciones operen a través de, y conmuten entre, blockchains más directamente, permitiendo de este modo que cualquier amenaza de seguridad o estabilidad en blockchains particulares se mitigue más rápidamente.

La solicitud de blockchain puede comprender una solicitud para leer datos en la una o más blockchains. La solicitud para leer datos de la una o más blockchains puede comprender un identificador de datos indicativo de los datos que se van a leer de la una o más blockchains.

La interfaz con la una o más blockchains puede comprender leer los datos solicitados en la una o más blockchains, y en donde el método comprende además la capa de comunicaciones de blockchain: comunicar, a la capa funcional de la aplicación, los datos leídos de la una o más blockchains.

La solicitud de blockchain puede comprender una solicitud para escribir datos en la una o más blockchains.

La solicitud para escribir datos puede comprender los datos que se van a escribir en la una o más blockchains.

La solicitud de blockchain puede comprender una solicitud para realizar una transacción cruzada entre ledgers (XLT) entre una primera blockchain de la pluralidad de blockchains y una segunda blockchain de la pluralidad de blockchains, y en donde la solicitud para realizar la XLT comprende un identificador de datos de XLT indicativo de los datos de la primera blockchain que se transferirán a la segunda blockchain mediante la XLT.

Las comunicaciones entre la capa de comunicaciones de blockchain y la capa funcional de la aplicación pueden llevarse a cabo de acuerdo con métodos definidos por una interfaz de programación de blockchain (BPI).

En un sexto aspecto de la presente divulgación, se proporciona un sistema que comprende: uno o más procesadores; y una memoria que almacena un programa de software, en donde el programa de software, cuando se ejecuta por el uno o más procesadores, hace que el uno o más procesadores realicen un método de acuerdo con cualquier reivindicación anterior.

En un séptimo aspecto de la presente divulgación, se proporcionan uno o más programas informáticos que, cuando se ejecutan mediante uno o más procesadores, provocan que el uno o más procesadores lleven a cabo un método de acuerdo con el quinto aspecto.

En un octavo aspecto de la presente divulgación, se proporciona un kit de desarrollo de software (SDK) que comprende un conjunto de herramientas de desarrollo de software para desarrollar el uno o más programas informáticos del séptimo aspecto.

En un noveno aspecto de la presente divulgación, se proporciona un método para realizar una transacción cruzada entre ledgers, XLT, entre una primera blockchain y una segunda blockchain, comprendiendo el método: identificar, en la primera blockchain, una transacción XLT propuesta relacionada con una XLT; almacenar, en un registro de transacción en un almacén de datos, un registro de la transacción XLT propuesta; identificar, en la segunda blockchain, una transacción XLT disponible; almacenar, en el registro de transacción en el almacén de datos, un registro de la transacción XLT disponible, en donde el registro es indicativo de la secuencia relativa de la transacción XLT propuesta y la transacción XLT disponible; determinar si la transacción XLT propuesta y la transacción XLT disponible cumplen o no un criterio de XLT; y si la transacción XLT propuesta y la transacción XLT disponible cumplen los criterios de XLT, difundir una transacción XLT de confirmación a la primera blockchain y la segunda blockchain para ejecutar la XLT entre la primera blockchain y la segunda blockchain, en donde los criterios de XLT comprenden un primer criterio de que la transacción XLT disponible corresponde a la transacción XLT propuesta y un segundo criterio de que el registro de transacción es indicativo de la transacción XLT propuesta que precede a la transacción XLT disponible. Por lo tanto, una secuencia de transacciones a través de dos o más blockchains puede mantenerse y entenderse de manera fiable, incluso en el caso de bifurcaciones en una o más de las blockchains, mejorando de este modo la seguridad y fiabilidad, y reduciendo el riesgo de fraude, en transacciones cruzadas entre ledgers.

El método puede comprender además: difundir una o más transacciones a la primera blockchain y/o a la segunda blockchain, en donde la una o más transacciones comprenden uno o más identificadores de registro respectivos, y en donde cada identificador de registro se determina basándose al menos en parte en al menos parte del contenido del registro de la transacción, y en donde el uno o más identificadores de registro indican conjuntamente el registro de la transacción XLT propuesta y la transacción XLT disponible.

El uno o más identificadores de registro pueden comprender uno o más hashes de al menos parte del contenido del registro de la transacción.

Los criterios de la XLT pueden comprender un tercer criterio de que la una o más transacciones están incluidas en la primera blockchain y/o la segunda blockchain, de tal manera que el uno o más identificadores de registro se almacenan en la primera blockchain y/o la segunda blockchain.

El registro de la transacción XLT propuesta puede comprender un hash del contenido de la transacción XLT propuesta y el registro de la transacción XLT disponible comprende un hash del contenido de la transacción XLT disponible.

En un décimo aspecto de la presente divulgación, se proporciona un sistema que comprende: uno o más procesadores; y una memoria que almacena un programa de software, en donde el programa de software, cuando se ejecuta por el uno o más procesadores, hace que el uno o más procesadores realicen un método de acuerdo con el noveno aspecto.

En un undécimo aspecto de la presente divulgación, se proporcionan uno o más programas informáticos que, cuando se ejecutan por uno o más procesadores, hacen que el uno o más procesadores lleven a cabo un método de acuerdo con el noveno aspecto.

En un duodécimo aspecto de la presente divulgación, se proporciona un kit de desarrollo de software (SDK) que comprende un conjunto de herramientas de desarrollo de software para desarrollar el uno o más programas informáticos del undécimo aspecto.

En un decimotercer aspecto de la presente divulgación, se proporciona un método para mantener un registro de una secuencia relativa de transacciones realizadas en dos o más blockchains. De esta manera, la secuencia de

transacciones en diferentes blockchains puede ordenarse de manera fiable, posibilitando de este modo mejoras en seguridad y fiabilidad, particularmente en el caso de bifurcaciones de blockchain y/o XLT.

En un decimocuarto aspecto de la presente divulgación, se proporciona un sistema que comprende: uno o más procesadores; y una memoria que almacena un programa de software, en donde el programa de software, cuando se ejecuta por el uno o más procesadores, hace que el uno o más procesadores realicen un método de acuerdo con el noveno aspecto.

En un decimoquinto aspecto de la presente divulgación, se proporcionan uno o más programas informáticos que, cuando se ejecutan por uno o más procesadores, provocan que el uno o más procesadores lleven a cabo un método de acuerdo con el noveno aspecto.

En un decimosexto aspecto de la presente divulgación, se proporciona un kit de desarrollo de software (SDK) que comprende un conjunto de herramientas de desarrollo de software para desarrollar el uno o más programas informáticos del undécimo aspecto.

Dibujos

Se describen aspectos de la divulgación, únicamente a modo de ejemplo, con referencia a los siguientes dibujos, en los que:

la figura 1 muestra una representación de ejemplo de una capa de comunicación del overledger de acuerdo con un aspecto de la presente divulgación;

la figura 2 muestra una representación de ejemplo de un sistema que utiliza la capa de comunicación del overledger de la figura 1;

la figura 3 muestra una representación de ejemplo de un sistema adicional que utiliza la capa de comunicación del overledger de la figura 1;

la figura 4 muestra una representación de las etapas de un método de ejemplo realizado usando el sistema de la figura 3;

la figura 5 muestra una representación de ejemplo de un sistema adicional que utiliza la capa de comunicación del overledger de la figura 1;

la figura 6 muestra una representación de ejemplo de una bifurcación en una blockchain;

la figura 7 muestra una representación de ejemplo de un método para mantener un registro de una secuencia de transacciones que aparecen en una o más blockchains de acuerdo con un aspecto de la presente divulgación;

la figura 8 muestra una representación de ejemplo de dos blockchains en las que se realiza el método de la figura 7;

la figura 9 muestra una representación de ejemplo adicional de dos blockchains en las que se realiza el método de la figura 7;

la figura 10 muestra una representación de ejemplo de dos blockchains en las que se realiza una transacción cruzada entre ledgers de acuerdo con un aspecto de la presente divulgación;

la figura 11 muestra una representación de ejemplo de un sistema adicional de acuerdo con un aspecto de la presente divulgación; y

la figura 12 muestra una representación de ejemplo de un método realizado por el sistema de la figura 11.

Descripción detallada

Los inventores han reconocido que a medida que la tecnología de blockchain crece y se desarrolla, puede volverse cada vez más difícil para diferentes entidades hacer uso de la misma de manera segura y fiable. Por ejemplo, actualmente, si una entidad desea desarrollar una nueva aplicación de software que usa tecnología de blockchain, debe iniciar su propia blockchain nueva (que es extremadamente lenta y compleja, requiriendo el establecimiento de un protocolo de blockchain completamente nuevo) o utilizar una tecnología de blockchain existente, tal como Bitcoin o Ethereum.

Los inventores se han dado cuenta de que para hacer uso de una tecnología de blockchain existente, una entidad debe desarrollar su aplicación de software para que sea totalmente compatible con el protocolo para esa blockchain,

por ejemplo, usando las técnicas criptográficas, el formato de transacción y los canales de comunicación dictados por el protocolo. Por lo tanto, la selección de qué tecnología de blockchain usar es una decisión que necesita hacerse muy temprano en el proceso de desarrollo de software, potencialmente antes de que se entiendan completamente los requisitos y usos de la aplicación de software.

Cada blockchain tiene un protocolo particular que se formuló para satisfacer las necesidades particulares de la blockchain. Los protocolos de blockchain no se pueden cambiar sin provocar una bifurcación dura en la blockchain, lo que significa que las blockchains no son muy adaptables a los cambios necesarios. En consecuencia, si la entidad decide posteriormente que su tecnología de blockchain elegida ya no satisface sus necesidades (por ejemplo, porque la tecnología de blockchain elegida se ha vuelto inestable o insegura, o porque los requisitos de la aplicación de software han cambiado de tal manera que una tecnología de blockchain diferente sería favorable, o debido a que las blockchains más nuevas son superiores a la tecnología de blockchain originalmente elegida, pero ahora obsoleta), para comenzar a usar una tecnología de blockchain alternativa, deben modificar su software para ajustarse al protocolo de la tecnología de blockchain alternativa. En la práctica, esto puede ser muy difícil, si no imposible, ya que el software se desarrollará sobre las bases establecidas por el protocolo de la tecnología de blockchain elegida originalmente. En consecuencia, elegir la tecnología de blockchain correcta desde el principio puede ser muy importante, pero en la práctica puede resultar muy difícil de lograr debido a la naturaleza de rápido desarrollo de la tecnología de blockchain.

Además, cualquier dato que se haya almacenado previamente en una blockchain puede no transferirse directamente a través de una blockchain alternativa, debido a diferencias entre protocolos de blockchain.

En vista de estas dificultades a largo plazo que han sido reconocidas por los inventores, estos han ideado una solución técnica que consiste en introducir una nueva capa de comunicaciones para separar la funcionalidad de las aplicaciones de las blockchains que están utilizando, y dicha solución se proporciona en la presente divulgación. Al desacoplar la funcionalidad de la aplicación de los diversos requisitos de la(s) blockchain(s) en las que está operando la aplicación, puede mejorarse la interoperabilidad entre aplicaciones y blockchains. Como se apreciará a partir de los aspectos de la divulgación descritos con más detalle a continuación, esto puede significar que los cambios en la tecnología de blockchain en el futuro pueden responderse de manera más eficiente y flexible. Esto puede ayudar a responder a amenazas de seguridad (por ejemplo, si se reconoce que una tecnología de blockchain tiene un fallo de seguridad, las aplicaciones pueden cambiar más directamente a una tecnología de blockchain más segura), mejorando la seguridad y la fiabilidad (dado que las aplicaciones pueden operar más fácilmente a través de múltiples blockchains, almacenar datos duplicados a través de las blockchains y/o transferir datos entre blockchains) y/o mejorar la eficiencia y la velocidad (por ejemplo, cuando una tecnología de blockchain más nueva tiene velocidades de procesamiento aumentadas para transacciones, las aplicaciones pueden pasar más fácilmente a esa tecnología de blockchain más nueva).

A lo largo de la siguiente divulgación, los términos "blockchain", "tecnología de blockchain" y "tecnología de ledger distribuido (DLT)" se usan indistintamente. Además, a lo largo de la siguiente divulgación, el término "transacción" se usa en relación con blockchains. El experto en la materia apreciará que una "transacción" en este contexto no se limita a una transferencia de propiedad de algo, por ejemplo, una moneda. En su lugar, una "transacción" es el medio por el que se puede introducir algún tipo de información en una blockchain. Por ejemplo, una entidad puede difundir una "transacción" para introducirla en una blockchain, comprendiendo la transacción la información que se va a introducir en la blockchain, así como cualquier otro elemento de datos (tal como claves públicas, una firma, un hash, etc.) que se requieren para que la transacción se acepte en la blockchain. La información que se introduce en la blockchain puede referirse a una transferencia financiera de moneda, pero como alternativa puede ser cualquier otro tipo de información (por ejemplo, un dato que una entidad desea almacenar en una blockchain para su propia consulta posterior).

La figura 1 muestra una visualización de ejemplo de la introducción de una "capa de comunicaciones del overledger" 20, ubicada entre una capa funcional de la aplicación 30 y una capa de la DLT 10. La terminología "capa de comunicaciones del overledger" se utiliza a lo largo de esta divulgación para significar una capa de comunicaciones que desacopla los requisitos de protocolo de las DLT de las operaciones funcionales de las aplicaciones que utilizan las DLT (a veces denominadas "lógica de negocio").

En la figura 1, dos DLT, una primera DLT 12 y una segunda DLT 14, se representan como parte de la capa de la DLT 10, aunque se apreciará que la capa de comunicaciones del overledger 20 puede ser capaz de interactuar con cualquier número de DLT diferentes, por ejemplo, Bitcoin, Ethereum, Corda, Openchain, Hyperledger, etc. Cada DLT tendrá su propio protocolo particular para operaciones de blockchain, tal como lectura de datos de blockchain, escritura de datos en la blockchain y/o transferencia de propiedad de datos de una entidad a otra entidad en la misma blockchain, o a una blockchain diferente (una transacción cruzada entre ledgers) (diferentes blockchains pueden permitir que se realicen una o más operaciones de lectura, escritura y transferencia en la blockchain, dependiendo del propósito para el que se haya configurado la blockchain).

La capa de comunicaciones del overledger 20 comprende los diferentes métodos mediante los cuales las operaciones de blockchain pueden realizarse en la primera DLT 12 y la segunda DLT 14. Por ejemplo, la primera

DLT 12 puede ser Bitcoin, que tiene un protocolo particular para que se escriban nuevas transacciones en su blockchain. El experto en la materia entenderá bien el protocolo particular de Bitcoin, pero brevemente este puede incluir información tal como la clave pública del beneficiario (la "billetera" del beneficiario), una cantidad a transferir al beneficiario, un hash de la transacción anterior mediante la cual el pagador obtuvo el bitcoin que ahora se está transfiriendo y una firma del pagador (firmada usando la clave privada del pagador). El protocolo define el formato permitido de cada uno de estos elementos (tal como la longitud y/o el tipo de objeto), las técnicas criptográficas que deben usarse para generar el hash y la firma (por ejemplo, SHA-256) y la secuencia en que cada artículo va a aparecer dentro de la transacción. El protocolo también puede definir cómo se puede leer la información de la blockchain de bitcoin, por ejemplo, identificando qué información se mantiene en cada transacción y en cada bloque en la blockchain, el formato y la secuencia de la información y las técnicas criptográficas utilizadas para generar los hashes, las firmas y pruebas de trabajo (de modo que cada una pueda verificarse al leer la blockchain). En contraste, la segunda DLT 14 puede tener un protocolo completamente diferente que permite que se realicen diferentes operaciones de blockchain, tales como solo operaciones de lectura (por ejemplo, debido a que es una blockchain cerrada que cualquiera puede leer, pero solo las entidades autorizadas pueden tener derecho a ello), y cada bloque puede comprender información diferente, en una secuencia diferente y/o usar técnicas criptográficas diferentes a las de la primera DLT 12. La capa de comunicaciones del overledger 20 comprenderá los métodos que posibilitan que se realicen las diversas operaciones permisibles en cada una de la primera DLT 12 y la primera DLT 14.

La capa funcional de la aplicación 30 representada en la figura 1 comprende las operaciones funcionales (a veces denominadas "lógica de negocio") de una aplicación o aplicaciones que están utilizando una o más de las DLT. Cuando la capa funcional de la aplicación 30 de una aplicación desea realizar una operación de blockchain, transmite la información correspondiente (una solicitud de blockchain) a la capa de comunicaciones del overledger 20 (por ejemplo, que comprende los datos que desea escribir en una DLT, o la naturaleza de la información que desea leer de la DLT), que a continuación interactúa con la(s) DLT correspondiente(s) de acuerdo con los protocolos de la(s) DLT correspondiente(s) para realizar la operación de la blockchain. La capa de comunicaciones del overledger 20 puede devolver a continuación cualquier resultado correspondiente a la capa funcional de la aplicación 30. Esta operación puede apreciarse más completamente a partir de los ejemplos más específicos descritos más adelante en la presente divulgación.

De modo que diversas entidades diferentes puedan hacer uso de los métodos contenidos en la capa de comunicaciones del overledger 20, los métodos pueden encapsularse, por ejemplo, en un kit de desarrollo de software (SDK), al que nos referiremos en adelante como el SDK del overledger. Se apreciará que los métodos contenidos en la capa de comunicaciones del overledger 20 pueden encapsularse de cualquier otra manera equivalente adecuada. Al obtener el SDK del overledger, un desarrollador puede seleccionar más directamente qué DLT desea utilizar para la aplicación de software particular que está desarrollando y disponer de los métodos y herramientas necesarios para interactuar con su DLT seleccionado(s) proporcionado(s) por el SDK del overledger.

La figura 2 muestra una representación de ejemplo de un sistema 200 que comprende un proveedor de SDK 210, un desarrollador de aplicaciones 220, una aplicación desarrollada 230, una primera red de DLT 240 y una segunda red de DLT 250. El proveedor de SDK 210 puede ser responsable de crear y mantener el SDK del overledger, que en este ejemplo comprende los métodos para interactuar con al menos la primera DLT 12 y la segunda DLT 14, aunque puede comprender métodos de interacción para cualquier número de DLT (es decir, para una pluralidad de DLT). Cuando el desarrollador de aplicaciones 220 desea desarrollar una aplicación que utiliza la primera DLT 12 y/o la segunda DLT 14, puede obtener el SDK del overledger del proveedor de SDK 210 (por ejemplo, descargándolo del proveedor de SDK 210, o de cualquier otra entidad de alojamiento del SDK del overledger). A continuación, puede utilizar el SDK del overledger para desarrollar la aplicación 230, que a continuación puede interactuar con la primera DLT 12 y/o la segunda DLT 14 de acuerdo con los métodos definidos en el SDK del overledger. En la figura 2, la primera red de DLT 240 representa la red de nodos participantes para la primera DLT 12 y la segunda red de DLT 250 representa la red de nodos participantes para la segunda DLT 14. En el ejemplo particular representado en la figura 2, se muestra que la aplicación 230 es capaz de interactuar tanto con la primera red DLT 240 como con la segunda red DLT 250, aunque se apreciará que el desarrollador de la aplicación puede utilizar alternativamente el SDK del overledger para configurar la aplicación 230 para interactuar con solo la primera red DLT 240 o solo la segunda red DLT 250.

En un aspecto particular, como parte del desarrollo de la aplicación 230, el desarrollador de la aplicación 230 puede desarrollar una interfaz de programación para manejar comunicaciones entre la capa de comunicaciones del overledger 20 y la capa funcional de la aplicación 30 de la aplicación 230. A partir de aquí en adelante nos referiremos a esta interfaz como la Interfaz de programación de blockchain (BPI), que está configurada para ayudar a simplificar la interacción de la capa funcional de la aplicación 30 y la capa de comunicaciones del overledger 20. Por ejemplo, la capa de comunicaciones del overledger 20 de la aplicación 230 puede operar de acuerdo con los métodos definidos en el SDK del overledger para manejar las comunicaciones a y desde la primera red DLT 240 y/o la segunda red DLT 250. El desarrollador de aplicaciones 220 puede desarrollar la BPI para definir los tipos particulares de información que la capa funcional de la aplicación 30 de la aplicación 230 puede pasar a la capa de comunicación del overledger 20 en una solicitud de blockchain relacionada con una operación de blockchain que se realizará en un bloque más y/o los tipos particulares de información que puede recibir de la capa de comunicaciones

del overledger 20. Por ejemplo, puede ofrecer un comando de "escribir en la primera DLT 12", que puede especificar cualquier parámetro correspondiente para que los datos se escriban en la primera DLT 12 (tal como longitud máxima, tipo de objeto, etc.) y/o un comando de "leer de la segunda DLT 14", que puede especificar cualquier dato correspondiente que se necesite para leer información de la segunda DLT 14 (por ejemplo, si se desea leer datos en la segunda DLT 14 que una entidad particular ha escrito en la segunda DLT 14, el comando "leer de la segunda DLT 14" puede especificar que necesita el identificador de blockchain de la entidad particular, tal como su clave pública, usando la cual la capa de comunicaciones del overledger 20 puede buscar los contenidos de la segunda DLT 14 para transacciones que incluyen ese identificador de blockchain). En consecuencia, puede observarse que la capa funcional de la aplicación 30 de la aplicación 230 puede utilizar los diversos comandos ofrecidos por la BPI de modo que la capa de comunicaciones del overledger 20 puede interactuar con cualquiera de las DLT que son soportadas por el SDK del overledger.

Por lo tanto, la capa funcional de la aplicación 30 de la aplicación 230 está desacoplada de los requisitos de protocolo de capa de la DLT 10 específicos de la primera DLT 12 y la segunda DLT 14 en virtud de la capa de comunicaciones del overledger 20 que opera de acuerdo con el SDK del overledger. La capa funcional de la aplicación 30 necesita únicamente enviar y recibir datos de la capa de comunicaciones del overledger 20 de la manera definida por la BPI para interactuar con la primera DLT 12 y/o la segunda DLT 14. En consecuencia, si o bien la primera DLT 12 o bien la segunda DLT 14 deberían cambiar sus protocolos en el futuro, la capa de comunicaciones del overledger 20 puede manejar esto en virtud de cambios en el SDK del overledger. La capa funcional de la aplicación 30 de la aplicación 230 puede permanecer sin cambios, ya que aún enviará datos a, y recibirá datos de, la capa de comunicaciones del overledger 20 como se define en la BPI, o como máximo mínimamente cambiada si el cambio de protocolo de la DLT es tal que la BPI debe cambiarse de alguna manera (por ejemplo, si se cambia el tipo de objeto que puede escribirse en una DLT, entonces la BPI también puede cambiar, lo que puede requerir que la capa funcional de la aplicación 30 proporcione datos de un tipo de objeto diferente). Sin embargo, está claro que incluso en el caso en el que puede requerirse un cambio en la capa funcional de la aplicación 30, será un cambio mínimo, y los aspectos centrales de la capa funcional de la aplicación 30 pueden permanecer sin cambios. Del mismo modo, se puede ver lo mismo si el desarrollador de aplicaciones 220 toma una decisión para comenzar a usar una DLT diferente (por ejemplo, porque la DLT elegida originalmente ahora es inestable o tiene fallos de seguridad, o porque se ha lanzado una nueva DLT con velocidades y/o eficiencia mejoradas). Si esa DLT diferente es compatible con el SDK del overledger, el desarrollador de la aplicación 220 puede simplemente actualizar la BPI para ofrecer comandos para interactuar con esa DLT diferente, que la capa funcional de la aplicación 30 de la aplicación 230 puede utilizar a continuación. De nuevo, puede verse que los cambios requeridos en la capa funcional de la aplicación 30 de la aplicación 230 pueden ser solo mínimos o nulos, ya que la capa funcional de la aplicación 30 no se habrá desarrollado para interactuar con una DLT particular y el protocolo específico que requiere esa DLT; en su lugar, se desarrolla para llevar a cabo los requisitos funcionales de la aplicación y utilizar la BPI siempre que se requiera alguna forma de interacción de blockchain.

Los beneficios de desacoplar la capa funcional de la aplicación 30 de la capa de la DLT 10 usando la capa de comunicaciones del overledger 20 pueden apreciarse incluso adicionalmente considerando un contexto en el que el desarrollador de la aplicación 220 desarrolla una oferta de servicio para que la utilicen otros desarrolladores de la aplicación "cliente". La oferta de servicio puede referirse a cualquier tipo de servicio de blockchain que otros desarrolladores de aplicaciones puedan desear utilizar, por ejemplo, un servicio que permita la escritura de tipos particulares de información en blockchains (que se describe con más detalle más adelante en la sección "aspecto de mensajería") y/o un servicio que habilita transacciones cruzadas entre ledgers (que se describe con más detalle más adelante en la sección "aspecto de transacción cruzada entre ledgers") y/o un servicio que habilita la lectura de tipos particulares de información de las DLT, etc.

La figura 3 muestra un sistema 300 de ejemplo para demostrar una oferta de servicio que puede utilizarse por desarrolladores de aplicaciones "cliente". El sistema 300 comprende el proveedor del SDK 210 (descrito anteriormente), un desarrollador de la oferta de servicio 310, un desarrollador de la aplicación cliente 320, una aplicación cliente 330, la primera red de DLT 240 (descrita anteriormente) y la segunda red de DLT 250 (descrita anteriormente).

La figura 4 muestra etapas de método de ejemplo que ejemplifican cómo el desarrollador de la oferta de servicio puede utilizar el SDK del overledger.

En la Etapa S410, el desarrollador de la oferta de servicio 310 puede obtener el SDK del overledger del proveedor del SDK 210 (por ejemplo, descargándolo del proveedor del SDK 210, o de cualquier otra entidad de alojamiento del SDK del overledger).

En la Etapa S420, el desarrollador de la oferta de servicio 310 puede desarrollar una BPI para el servicio que está ofreciendo y encapsularla apropiadamente (por ejemplo, en un archivo de configuración de BPI, aunque puede usarse cualquier otra alternativa adecuada en su lugar). La BPI especificará funciones, comandos, etc., que pueden usarse mediante una capa funcional de la aplicación 30 de la aplicación cliente 330 para interactuar con una o más DLT, como se ha explicado anteriormente.

En la Etapa S430, el desarrollador de la aplicación cliente 320 puede obtener el SDK del overledger y el archivo de configuración de la BPI (por ejemplo, descargándolos del desarrollador de la oferta de servicio 310, o de cualquier otra entidad de alojamiento adecuada).

En la Etapa S440, el desarrollador de la aplicación cliente 320 desarrolla la capa funcional de la aplicación 30 de su aplicación cliente 330. Cuando la capa funcional de la aplicación 30 necesita interactuar con una DLT de cualquier manera, la capa funcional de la aplicación 30 puede simplemente utilizar un comando o función en la BPI de manera apropiada para comunicar una solicitud de blockchain a la capa de comunicaciones del overledger 20, la solicitud de blockchain relacionada con una operación de blockchain que se realizará en una o más blockchains. Por lo tanto, puede verse que el SDK del overledger y la BPI proporcionados por el desarrollador de la oferta de servicio 310 desacoplan la capa funcional de la aplicación 30 de la capa de la DLT 10. Por lo tanto, se puede considerar que el SDK del overledger realiza la función de la capa de comunicaciones del overledger 20 y la BPI define cómo la capa funcional de la aplicación 30 puede interactuar con la capa de comunicaciones del overledger 20 para llevar a cabo los servicios particulares que ofrece el desarrollador de la oferta de servicio.

En consecuencia, si el desarrollador de la aplicación cliente 320 decide en una fecha posterior que desea cambiar la DLT que usa para la aplicación cliente 320 (por ejemplo, inicialmente la capa de la aplicación funcional 30 de la aplicación cliente 330 puede haberse configurado para usar funciones y comandos de la BPI para interactuar con la primera DLT 22, pero más tarde se decide que debe usarse la segunda DLT 24 en su lugar), simplemente necesita cambiar las funciones y comandos de la BPI particulares que usa la capa de la aplicación funcional 20 (suponiendo que el SDK y la BPI del overledger son compatibles tanto con la primera DLT 22 como con la segunda DLT 24). Análogamente, si algo fuera a cambiar acerca de los protocolos de cualquiera de las DLT soportadas por el SDK del overledger, el SDK del overledger puede cambiarse y el desarrollador de la oferta de servicio 310 puede, en consecuencia, cambiar la BPI, si es necesario. Como resultado, como máximo, el desarrollador de la aplicación cliente solo debería necesitar cambiar las funciones y comandos de la BPI particulares que usa la capa de la aplicación funcional 30 de la aplicación cliente 330, en lugar de tener que cambiar cualquiera de los aspectos centrales de la capa de la aplicación funcional 30 de la aplicación cliente.

Opcionalmente, después de que el desarrollador de la oferta de servicio 310 haya creado el archivo de configuración de la BPI, puede firmarlo usando técnicas de cifrado de clave pública-privada estándar, de modo que el desarrollador de la aplicación cliente 320 pueda verificar que el archivo de configuración de la BPI que obtiene es válido.

El SDK del overledger puede incluir una serie de características diferentes que posibilitan que tengan lugar tipos particulares de interacciones de DLT. Dos de estas características particulares, el "aspecto de mensajería" y el aspecto de "transacción cruzada entre ledgers", se describen con más detalle a continuación.

Se apreciará que aunque la capa funcional de la aplicación 30 y la capa de comunicaciones del overledger 20 se describen ambas anteriormente como "capas", pueden verse alternativamente como módulos o paquetes de software. Por ejemplo, pueden ser dos módulos dentro de la aplicación 230 o la aplicación cliente 330. Como alternativa, cada uno puede estar ubicado dentro de diferentes aplicaciones, operando potencialmente en diferentes dispositivos electrónicos. Por ejemplo, la capa funcional de la aplicación 30 puede ser un módulo o paquete de software que opera en un dispositivo electrónico de cliente, y la capa de comunicaciones del overledger 20 puede ser un módulo o paquete de software diferente que opera en un dispositivo electrónico de la oferta de servicio, en donde las dos capas se comunican de acuerdo con la BPI a través de cualquier interfaz de comunicaciones adecuada. Además, la capa de comunicaciones del overledger 20 puede verse como un módulo o paquete de software configurado para permitir que se realicen operaciones de blockchain particulares en una cualquiera o más DLT diferentes, en donde el desarrollador de la oferta de servicio 310 permite a los desarrolladores de la aplicación cliente descargar la capa de comunicación del overledger 20 y hacer uso de la misma a través de la BPI, o utilizarla de forma remota a través de la BPI.

Aspecto de mensajería

Los inventores han reconocido que a menudo puede ser útil mantener en una blockchain un registro inmutable de información que la blockchain no está diseñada necesariamente para almacenar. A lo largo de lo siguiente, se usa el término "mensaje", pero debería entenderse que "mensaje" abarca cualquier tipo de información para la que existe un deseo de mantener un registro usando una blockchain.

Manteniendo un registro de un mensaje usando una blockchain, puede mantenerse un registro efectivamente inmutable del mensaje, de tal manera que el contenido del mensaje puede verificarse en el futuro. Las blockchains están equipadas para aceptar algunos tipos de información dentro de las transacciones que se incluyen en cada bloque, pero normalmente son muy prescriptivas en cuanto a cuál puede ser esa información. Por ejemplo, una serie de blockchains populares (tal como Bitcoin) se diseñaron originalmente para criptomonedas y transferencias financieras simples. Se han desarrollado otras blockchains para otros fines específicos, y es probable que en el futuro aún se desarrollen más blockchains para propósitos específicos adicionales. En el curso del desarrollo de la capa de comunicaciones del overledger 20, los inventores se dieron cuenta de que puede resultar muy útil poder

almacenar algunos tipos de información en blockchains que no se diseñaron para ese fin. Un ejemplo de uso particular es como parte del mantenimiento de un registro de una secuencia de transacciones que aparecen en una o más blockchains como se describe más adelante, aunque hay muchos otros usos y beneficios potenciales.

Habiéndose dado cuenta de los beneficios potenciales de almacenar mensajes en blockchains que no se diseñaron para ese propósito, los inventores investigaron los detalles de las transacciones dentro de las arquitecturas de blockchain existentes. Se observó que una serie de arquitecturas de blockchain incluyen un campo dentro de las transacciones, en el que pueden añadirse datos no esenciales. Por ejemplo, Bitcoin tiene un campo "OP-RETURN", Ethereum tiene un campo que se usa para código de bytes de script y Ripple tiene un campo "memos". Estos campos pueden denominarse en general como "campos de datos no esenciales". Sin embargo, cada uno de estos campos está relativamente limitado en tamaño y, por lo tanto, en utilidad.

La solución identificada por los inventores es usar tales campos para almacenar un valor que se basa al menos en parte en, y es únicamente indicativo de, un mensaje (tal como un hash o cifrado del mensaje) y luego almacenar el mensaje en otro lugar (fuera de la cadena). El valor que indica de forma única el mensaje puede actuar como un indicador (por ejemplo, un indicador de hash) del mensaje completo que se almacena fuera de la cadena. Al almacenar un valor en la blockchain que es únicamente indicativo del mensaje, la integridad del mensaje almacenado fuera de la cadena puede comprobarse en cualquier momento simplemente generando un valor de prueba basado en el mensaje almacenado fuera de la cadena y comprobándolo con el valor almacenado en la blockchain. Debido a que el valor de prueba se genera de la misma manera que el valor almacenado en la blockchain (por ejemplo, usando la misma receta de hash), si el valor de prueba coincide con el valor almacenado en la blockchain, entonces el mensaje almacenado fuera de la cadena tiene integridad (es decir, no se ha alterado). Si no coinciden, entonces el mensaje almacenado fuera de la cadena se ha manipulado de alguna manera. Esto se debe a que los contenidos de la blockchain son efectivamente inmutables, por lo que el valor almacenado en la blockchain no puede alterarse de ninguna manera.

El experto en la materia apreciará que el valor que es únicamente indicativo del mensaje puede generarse de cualquier manera adecuada. Por ejemplo, puede generarse mediante el hash del contenido del mensaje usando un algoritmo de hash tal como SHA-2 (por ejemplo, SHA-256, SHA-512, etc.) o SHA-3, de modo que el valor únicamente indicativo del mensaje es un hash, que actúa como un indicador de hash en la blockchain. Como alternativa, puede generarse cifrando el contenido del mensaje usando cualquier algoritmo de cifrado adecuado. La forma particular en la que se puede generar el valor que es unívocamente indicativo del mensaje se puede elegir en función de los requisitos de la blockchain particular en la que se almacenará y/o cualquier requisito criptográfico particular que deba cumplirse (por ejemplo, nivel de integridad ofrecido por el algoritmo, etc.). Por ejemplo, el campo de transacción dentro del que ha de almacenarse el valor puede tener una longitud máxima particular que excluye ciertos algoritmos de hash puesto que los hashes que producen pueden ser demasiado largos. El SDK del overledger puede incluir el o los algoritmos de hash y/o cifrado particulares que son apropiados para las DLT que admite, de tal manera que una capa funcional de una aplicación puede simplemente transmitir el mensaje a la capa de comunicaciones del overledger 20, que a continuación puede gestionarlo apropiadamente para almacenar un registro del mensaje en una DLT. La aplicación puede almacenar a continuación una copia del mensaje en otro lugar fuera de la cadena (por ejemplo, en la memoria local en el dispositivo en el que está operando la aplicación, o en la memoria remota tal como una base de datos remota o almacenamiento en la nube, etc.).

Como se ha mencionado anteriormente, puede haber muchos usos potenciales para el aspecto de mensajería descrito anteriormente. Un caso de uso de ejemplo particular, no limitante, se describe a continuación con referencia a la figura 5.

La figura 5 muestra un sistema 500 que comprende el desarrollador de la oferta de servicio 310, el desarrollador de la aplicación cliente 320, un back-end de la aplicación cliente 510, una primera aplicación cliente 520, una segunda aplicación cliente 530 y la primera red de DLT 240. El desarrollador de la oferta de servicio 310 puede crear una BPI para un servicio de mensajería que ofrece funciones de BPI para grabar mensajes en una o más DLT usando el aspecto de mensajería descrito anteriormente. El desarrollador de la aplicación cliente 320 puede usar a continuación el SDK del overledger y el archivo de configuración de la BPI del desarrollador de la oferta de servicio 310 para desarrollar una aplicación de mensajería que habilita a los usuarios de la aplicación de mensajería para enviarse mensajes de forma segura entre sí. La primera aplicación cliente 520 y la segunda aplicación cliente 530 pueden ser aplicaciones cliente duplicadas que operan en dispositivos electrónicos del primer y segundo usuario respectivamente, por ejemplo en dispositivos electrónicos móviles (tales como teléfonos inteligentes, tabletas u ordenadores portátiles) o en ordenadores de escritorio. La primera aplicación cliente 520 y la segunda aplicación cliente 530 pueden operar en los dispositivos electrónicos del primer y segundo usuario como instalaciones de software en los dispositivos, o como una aplicación basada en navegador web, etc.

Si el primer usuario desea comunicar un mensaje al segundo usuario, puede introducir el mensaje y una identificación del segundo usuario (la identificación puede tomar cualquier forma adecuada, por ejemplo, un nombre de usuario del segundo usuario de la aplicación cliente) en la primera aplicación cliente 520. La capa de la aplicación funcional 30 de la aplicación cliente puede configurarse para transmitir el mensaje a la capa de comunicaciones del overledger 20 de la primera aplicación cliente 520 para añadir a la primera DLT 12 usando la función/comando

apropiado en la BPI producida por el desarrollador de la oferta de servicio 310. La capa de comunicaciones del overledger 20 puede a continuación realizar un hash del mensaje de acuerdo con los métodos del SDK para interactuar con la primera DLT 12, construir una transacción apropiada para la primera DLT 12 que comprende el hash del mensaje y comunicar la transacción a la primera red de DLT 240. El hash del mensaje se incluirá, por lo tanto, en la primera DLT 12. La capa de comunicaciones del overledger 20 de la primera aplicación cliente 520 también puede configurarse opcionalmente para devolver una notificación a la capa funcional de la aplicación 30 cuando ha determinado que la transacción se ha añadido con éxito a la primera DLT 12. También se puede comunicar una copia del mensaje completo y el hash del mensaje al back-end de la aplicación cliente 510 (que puede ser, por ejemplo, un servidor, o base de datos, o sistema de almacenamiento basado en la nube ofrecido por el desarrollador de la aplicación cliente 320 para recibir, almacenar y recuperar mensajes), por ejemplo, recibiendo la capa funcional de la aplicación 30 el hash del mensaje desde la capa de comunicaciones del overledger 20 y comunicándolo a continuación al back-end de la aplicación cliente 510 junto con el mensaje. La comunicación entre la primera aplicación cliente 520 y el back-end de la aplicación cliente 510 puede asegurarse opcionalmente usando cualquier técnica criptográfica apropiada.

El segundo usuario puede darse cuenta entonces de que le está esperando un mensaje. Hay muchas formas diferentes en las que esto puede suceder, por ejemplo, el identificador del segundo usuario puede haber sido la clave pública de la DLT del segundo usuario, o la billetera de la DLT, o puede haber sido utilizado por la primera aplicación cliente 520 para obtener la clave pública de la DLT del segundo usuario, o la billetera de la DLT, y la transacción en la DLT 12 puede identificar la clave pública del segundo usuario, o la billetera de la DLT, como el destinatario de la transacción. En este caso, la segunda aplicación cliente 530 puede notar que una nueva transacción en la primera DLT 12 identifica al segundo usuario (si la segunda aplicación cliente 530 está configurada para supervisar los contenidos de cada nuevo bloque añadido a la primera DLT 12 para buscar cualquier transacción nueva que identifique la clave pública de la DLT del segundo usuario, o billetera de la DLT). Como alternativa, un servicio de terceros puede realizar este proceso de supervisión y notificar a la segunda aplicación cliente 530. La capa funcional de la aplicación 30 de la segunda aplicación cliente 530 puede obtener a continuación el hash del mensaje de la primera DLT 12 usando una función de "lectura" de BPI apropiada.

Una vez que la segunda aplicación cliente 530 tiene una copia del hash del mensaje, la capa funcional de la aplicación 30 puede comunicarlo al back-end de la aplicación cliente 510 junto con alguna forma de autenticación para autenticar al segundo usuario en el back-end de la aplicación cliente 510 (por ejemplo, usando cualquier forma de proceso de autenticación que hayan establecido la segunda aplicación cliente 530 y el back-end de la aplicación cliente 510, o proporcionando una firma digital firmada por la clave privada de la DLT del segundo usuario, que el back-end de la aplicación cliente 510 puede comprobar usando la clave pública asociada con el hash del mensaje almacenado en la primera DLT 240). Si se supera la autenticación, el mensaje completo correspondiente al hash del mensaje puede devolverse a continuación a la segunda aplicación cliente 530. La segunda aplicación cliente 530 puede entonces verificar la integridad del mensaje devuelto transmitiéndolo a través del mismo algoritmo de hash que usó la primera aplicación cliente 520 (por ejemplo, usando una función de BPI de "comprobar mensaje" apropiada, o de cualquier otra manera considerada adecuada por el proveedor de la oferta de servicio 310 y/o el desarrollador de la aplicación cliente 320) para crear un hash de prueba y luego comparar el hash de prueba con el hash del mensaje recuperado de la primera DLT 12.

Puede resultar evidente una serie de beneficios de este proceso de ejemplo particular. En primer lugar, el contenido del mensaje puede transmitirse de forma segura del primer usuario al segundo usuario, sin que este tenga que llevar a cabo técnicas complejas y a menudo difíciles de implementar de manera fiable, tales como correos electrónicos cifrados o documentos protegidos con contraseña que requieren que se acuerde una contraseña entre los dos usuarios. Además, el segundo usuario 530 puede estar seguro de que el back-end de la aplicación cliente 510 no ha manipulado el mensaje de ninguna manera, ya que de lo contrario el hash de prueba no coincidiría con el hash del mensaje almacenado en la primera DLT 12. Además, si en cualquier momento en el futuro hay alguna forma de disputa entre el primer y segundo usuarios con respecto al contenido del mensaje, el hash del mensaje almacenado en la primera DLT 12 puede usarse para resolver la disputa de manera absoluta.

El segundo usuario puede responder igualmente al mensaje de una manera análoga.

Se apreciará que este es meramente un caso de uso de ejemplo particular para el aspecto de mensajería de la presente divulgación y que pueden ser posibles muchos otros. Además, son posibles muchas implementaciones alternativas del caso de uso particular descrito anteriormente. Por ejemplo, el desarrollador de la oferta de servicio 310 puede proporcionar el back-end de la aplicación y la BPI puede configurarse simplemente para recibir el mensaje y la capa de conmutaciones del overledger 20 se encarga de introducir el hash del mensaje en la primera DLT 12 y también comunicar el mensaje y el hash del mensaje al back-end de la aplicación 510. En este caso, la capa funcional de la aplicación 30 de la aplicación cliente podría estar relacionada únicamente con otras funciones, tales como la GUI y otras funciones proporcionadas por la aplicación cliente. Además, la capa funcional de la aplicación 30 puede utilizar funciones o comandos ofrecidos por la BPI para introducir el hash del mensaje en no solo una, sino muchas, DLT, ofreciendo de este modo redundancia adicional y respaldo para el hash del mensaje.

Aspecto de la secuenciación de las transacciones

Un requisito importante de las operaciones de blockchain fiables es la secuenciación de las transacciones. Por ejemplo, se ha reconocido previamente que para evitar el fraude del doble gasto en las criptomonedas, cuando se difunde una nueva transacción a una red de blockchain, debe ser posible determinar si la criptomoneda identificada en esa nueva transacción se ha gastado previamente o no, e incluir únicamente la nueva transacción en un nuevo bloque si no se ha gastado previamente (es decir, evitar el doble gasto). Una serie de blockchains existentes logran esto en virtud de la naturaleza cronológica de los bloques en la blockchain, incluyendo cada bloque dentro del mismo una marca de tiempo, y comprobando en los bloques existentes que la criptomoneda identificada en la nueva transacción no se ha gastado antes.

Sin embargo, los inventores han reconocido que la secuenciación fiable de transacciones puede ser importante no solo en blockchains de criptomonedas, sino más en general para todos los tipos de blockchains. Por ejemplo, se apreciará que en diversos campos y protocolos de Internet y telecomunicaciones, la mensajería con una secuencia incorrecta puede explotarse como una vulnerabilidad de seguridad y estabilidad. Por ejemplo, existen numerosas formas en las que la secuenciación y la sincronización de los paquetes de TCP/IP pueden afectar a la seguridad de una seguridad, tales como mensajes que se salen de la secuencia correcta o conjuntos de mensajes incompletos que dan como resultado que la seguridad del sistema se vea comprometida ya sea en términos de integridad o de disponibilidad. Como tal, los inventores han reconocido que mantener la secuencia transaccional correcta dentro de las blockchains, por ejemplo, cuando se usa para transportar mensajes de acuerdo con el aspecto de mensajería descrito anteriormente, pero también cuando se usa para cualquier otro propósito, puede ser de importancia significativa en la consecución de un sistema seguro, fiable y resistente al fraude que utilice una o varias DLT.

Los inventores han identificado dos desafíos particulares para mantener la secuencia transaccional correcta en las DLT:

1. Si una blockchain se bifurca, una o más transacciones que aparecieron en la rama más corta de la bifurcación pueden perderse de la blockchain. El experto en la materia entenderá fácilmente el potencial para que las DLT se bifurquen, y las características de cómo se comportan las DLT en ese caso. Sin embargo, brevemente, no vale la pena que tradicionalmente, cuando se pierden transacciones a consecuencia de una bifurcación, puedan añadirse posteriormente en la rama más larga de la bifurcación mediante la red de nodos participantes cuando se ha reconocido que se han perdido transacciones.
La figura 6 muestra una representación de ejemplo de una bifurcación en una blockchain. La transacción 1 forma parte del Bloque B1 que aparece en la "Rama 1" de la blockchain. Sin embargo, como resultado de la bifurcación, se ha perdido efectivamente de la blockchain ya que la blockchain ha continuado en la rama más larga, la "Rama 2". La transacción 2 forma parte del Bloque B3, que aparece en la "Rama 2". Algún tiempo después, la Transacción 1 se anexa a la blockchain como parte del Bloque B8 en la "Rama 2". Por lo tanto, la Transacción 1 aparece más tarde en la blockchain que la Transacción 2. Sin embargo, si es importante que la Transacción 1 preceda a la Transacción 2 (por ejemplo, corresponden a mensajes consecutivos), esto puede provocar un fallo de seguridad o estabilidad, o puede permitir que tenga lugar un fraude, en una aplicación que está utilizando la blockchain.

2. Como se ha explicado anteriormente, en algunos contextos, puede ser deseable que una aplicación utilice dos o más DLT diferentes. Como parte de esto, a veces puede ser necesario realizar una transacción cruzada entre ledgers (XLT) para mover datos (por ejemplo, un mensaje (como se describió anteriormente), o una criptomoneda u otro tipo de información almacenada en una DLT) de una DLT a otra. Sin embargo, aunque cada blockchain puede desarrollarse para mantener una secuencia entre diferentes bloques en la blockchain (por ejemplo, usando marcas de tiempo), puede ser muy difícil discernir de manera fiable una secuencia entre transacciones que aparecen en diferentes blockchains, ya que cada DLT puede utilizar diferentes procedimientos de marcas de tiempo y/o sus relojes usados para las marcas de tiempo pueden estar desincronizados. Por lo tanto, cuando se utilizan dos o más (una pluralidad) de DLT, determinar y mantener una secuencia de transacciones que aparecen en las DLT puede ser muy importante para la seguridad y estabilidad de una aplicación que está usando las DLT.

La figura 7 muestra etapas del método de ejemplo de un proceso desarrollado por los inventores para mantener la secuencia transaccional usando "conjuntos de verificación". La figura 8 muestra una visualización de ejemplo de dos DLT y conjuntos de verificación para ayudar a comprender las etapas del método de la figura 7.

En la Etapa 710, la primera DLT 12 y la segunda DLT 14 se monitorizan para identificar cuándo se incluye una nueva transacción de interés en un bloque añadido a la primera DLT 12 o a la segunda DLT 14. La primera DLT 12 y la segunda DLT 14 pueden supervisarse leyendo el contenido de cada nuevo bloque cuando se añade a la DLT. Los criterios de relevancia para lo que constituye una transacción de interés es una cuestión de la implementación particular de este método. Por ejemplo, la BPI divulgada anteriormente puede configurarse de tal manera que cualquier transacción que se introduzca en una DLT utilizando la BPI incluya algún identificador particular que la distinga de las transacciones que se han introducido en la DLT por cualquier otro medio (por ejemplo, por otras entidades sin utilizar la BPI). El identificador particular puede ser, por ejemplo, la clave pública usada para la transacción, o algún otro tipo de información dentro de la transacción. En consecuencia, el criterio de relevancia

puede ser el identificador incluido por la BPI, de modo que cualquier transacción incluida en la primera DLT 12 o la segunda DLT 14 que usa el servicio proporcionado por el desarrollador de la oferta de servicio 310 (por ejemplo, transacciones añadidas por la primera aplicación cliente 520 o la segunda aplicación cliente 530, o, de hecho, por alguna aplicación de servicio que opera el desarrollador de la oferta de servicio para introducir transacciones en las DLT cuando sea necesario) puede considerarse correspondiente que cumpla el criterio de relevancia. Esto puede ser particularmente útil si la primera DLT 12 y/o la segunda DLT 14 son DLT de propósito general que pueden utilizar muchas entidades diferentes, pero el desarrollador de la oferta de servicio 310 necesita identificar transacciones relacionadas con su servicio para garantizar la secuenciación adecuada de esas transacciones para salvaguardar la seguridad y estabilidad de su oferta de servicio.

En la figura 8, las Transacciones A, C, E y O son todas transacciones de interés. Las transacciones etiquetadas como "Transacción X" no son de interés.

Cuando se añade un nuevo bloque a la primera DLT 12 o a la segunda DLT 14 y se identifica en la Etapa 710 que incluye una transacción o transacciones correspondientes, en la Etapa S720 se puede determinar un identificador de transacción (por ejemplo, un hash) para cada una, identificado basándose al menos en parte en el contenido de la transacción o transacciones. Este identificador se añade a continuación al conjunto de verificación más reciente, o "en curso". El conjunto de verificación forma parte de un registro fuera de la cadena de transacciones correspondientes que se almacena en un almacén de datos tal como en cualquier memoria adecuada, base de datos, matriz de datos, almacenamiento en la nube, etc. El identificador de transacción puede actuar como un indicador (por ejemplo, un indicador de hash) de la transacción correspondiente en la DLT. Cada nuevo identificador de transacción se añade al conjunto de verificación "en curso" en la secuencia en la que se detectó. Como tal, el registro de la transacción identificada es indicativo de la secuencia relativa de las transacciones, independientemente de si han aparecido en la primera DLT 12 o la segunda DLT 14. Como resultado, puede verse que monitorizando ambas DLT y añadiendo un identificador de transacción para cada transacción correspondiente tan pronto como se hayan añadido a la primera DLT 12 o a la segunda DLT 14, puede registrarse una secuencia relativa de transacciones a través de una o más DLT diferentes. El experto en la materia reconocerá que el conjunto de verificación "en curso" puede almacenarse usando cualquier técnica de almacenamiento de datos adecuada que conserve la secuenciación de las transacciones correspondientes. Una vez que se ha añadido una nueva transacción correspondiente al conjunto de verificación "en curso", el proceso puede volver a la Etapa S710 para la monitorización continua de la primera DLT 12 y la segunda DLT 14.

En un momento particular, el proceso puede continuar en la Etapa S730 para iniciar el proceso de almacenamiento de un registro (tal como un identificador de conjunto de verificación) del conjunto de verificación "en curso" (por ejemplo, un hash o cifrado basado al menos en parte en el contenido del conjunto de verificación "en curso") a la primera DLT 12 y a la segunda DLT 14. El momento particular en el que el proceso avanza a la Etapa S730 es una cuestión de implementación y puede ser, por ejemplo, después de que se haya invertido una cantidad de tiempo determinada monitorizando las DLT desde que se agregó el registro del conjunto de verificación anterior más reciente a las DLT, o después de que se haya añadido un número particular de transacciones correspondientes al conjunto de verificación "en curso", o después de que se haya añadido un número particular de bloques a la primera DLT 12 o a la segunda DLT 14 desde que el registro del conjunto de verificación anterior más reciente se añadió a las DLT, etc.

En la Etapa S730, se determina si está presente o no un registro (por ejemplo, un hash o cifrado) del contenido del conjunto de verificación precedente más reciente tanto en la primera DLT 12 como en la segunda DLT 14 (por ejemplo, para comprobar que no ha habido una bifurcación en ninguna de las DLT, lo que hace que ese registro se haya perdido). Si el registro no aparece en ambas DLT ("Sí" en la figura 7), el proceso continúa en la Etapa S750. Si no aparece en ambas DLT, ("No" en la figura 7), el proceso continúa en la Etapa S740. Esta comprobación puede realizarse buscando en ambas DLT el registro del conjunto de verificación precedente más reciente.

En la Etapa S740, el registro que falta puede reproducirse en la(s) DLT de la(s) que este falta (esencialmente, la transacción en la que el registro se introdujo por primera vez en la DLT, por ejemplo, usando el "aspecto de mensajería" anterior, se vuelve a difundir y añadir a la DLT). Sin embargo, antes de reproducir el registro que falta, puede determinarse en primer lugar si algunas de las transacciones que se identificaron dentro del conjunto de verificación precedente más reciente también faltan o no de la correspondiente DLT en la que deberían haber aparecido. Esto puede determinarse rápidamente usando la naturaleza del indicador (por ejemplo, un indicador de hash) de los identificadores de la transacción en el conjunto de verificación más precedente tal como se almacena en el registro de las transacciones. Cualesquiera transacciones que faltan pueden reproducirse a continuación en la DLT correspondiente antes de la reproducción del registro que falta del conjunto de verificación precedente más reciente. Aunque cualquier transacción reproducida puede aparecer ahora más tarde en una DLT que las transacciones a las que habría precedido la transacción que falta (si no hubiera estado ausente), los conjuntos de verificación mantendrán aún un registro apropiado de la secuenciación correcta de las transacciones. En consecuencia, los registros de conjunto de verificación que faltan, y cualquier transacción que falta asociada con los mismos, aún pueden añadirse de nuevo en una DLT, sin perder la secuenciación apropiada en la que tuvieron lugar las transacciones. Los detalles de esta etapa pueden apreciarse más completamente a partir del análisis de las figuras 8 y 9 a continuación.

En la Etapa S750, se determina si todas las transacciones identificadas en el conjunto de verificación "en curso" todavía aparecen o no en las DLT correspondientes (por ejemplo, comprobar que no se ha perdido ninguna debido a una bifurcación en una, o ambas, de las DLT). Si todavía aparecen todas en las DLT correspondientes ("Sí" en la figura 7), el proceso continúa en la Etapa S770. Si no aparecen todas en las DLT correspondientes ("No" en la figura 7), el proceso continúa en la Etapa S760. Esta comprobación puede realizarse en virtud de la naturaleza del indicador de los identificadores de transición en el conjunto de verificación "en curso".

En la Etapa S760, la transacción o transacciones identificadas como que faltan ahora pueden reproducirse en su correspondiente DLT en la que deberían haber aparecido (esencialmente, volver a difundirse y añadirse a la DLT). Aunque cualquier transacción reproducida puede aparecer ahora más tarde en una DLT que las transacciones a las que habría precedido la transacción que falta (si no hubiera estado ausente), el conjunto de verificación "en curso" mantendrá aún un registro apropiado de la secuenciación correcta de las transacciones. En consecuencia, las transacciones que faltan aún pueden añadirse de nuevo en una DLT, sin perder la secuenciación apropiada en la que tuvieron lugar las transacciones.

En la Etapa S770, se determina un registro del conjunto de verificación "en curso" basándose al menos en parte en el contenido del conjunto de verificación "en curso" (por ejemplo, una determinación de un identificador de conjunto de verificación tal como un hash o un cifrado basándose al menos en parte en los contenidos del conjunto de verificación) y se añade tanto a la primera DLT 12 como a la segunda DLT 14. El registro puede añadirse a ambas DLT utilizando las técnicas descritas en el "aspecto de mensajería" anterior. En consecuencia, se puede ver que al usar el "aspecto de mensajería" descrito anteriormente, se puede añadir un registro del contenido del conjunto de verificación "en curso" a ambas DLT, proporcionando de este modo un registro efectivamente inmutable del contenido de ese conjunto de verificación, usando la integridad de los datos, y por lo tanto puede lograrse una seguridad y fiabilidad mejoradas del registro de las transacciones correspondientes. El proceso puede volver a continuación a la Etapa S710 para continuar monitorizando las DLT y comenzar a trabajar en el siguiente conjunto de verificación "en curso".

Volviendo a la figura 8, podemos ver un registro de las transacciones 840 correspondientes almacenadas en un almacén de datos fuera de la cadena (que puede ser cualquier tipo adecuado de almacén de datos, o base de datos, o almacenamiento en la nube o cualquier otro tipo de memoria), que comprende el conjunto de verificación$_{n-1}$ 810, el conjunto de verificación$_n$ 820 y el conjunto de verificación$_{n+1}$ 830. El conjunto de verificación$_{n-1}$ 810 comprende un hash de la transacción Y y un hash de la transacción Z, que son transacciones anteriores en la primera DLT 12 o la segunda DLT 14 y no se representan de otra manera en la figura 8 por motivos de simplicidad. El conjunto de verificación$_n$ 820 comprende un hash de la transacción A 822', un hash de la transacción O 824' y un hash de la transacción C 826'. La transacción A 822 y la transacción C 824 aparecen en la primera DLT 12 y la transacción O 824 aparece en la segunda DLT 14. La secuencia en la que aparecen en el conjunto de verificación$_n$ 820 indica que la transacción O 824 se detectó después de la transacción A 822, pero antes que la transacción C 826. El conjunto de verificación$_{n+1}$ 830 comprende un hash de la transacción E 832', y puede verse en este ejemplo como el conjunto de verificación "en curso".

Como puede verse, un hash del conjunto de verificación$_{n-1}$ 810' y un hash del conjunto de verificación$_n$ 820' se han añadido previamente a ambas DLT, según la Etapa S770 descrita anteriormente. Cuando el conjunto de verificación$_n$ 820 era el conjunto de verificación "en curso", se apreciará que cuando llegó a la Etapa S730 descrita anteriormente, el hash del conjunto de verificación$_{n-1}$ 810' estaba presente en ambas DLT, por lo que el proceso podría continuar directamente a la Etapa S750. Del mismo modo, cuando llegó a la Etapa S750, la transacción A 822, la transacción O 824 y la transacción C 826 aún aparecían en sus DLT correspondientes, por lo que el proceso podría continuar directamente a la Etapa S770 para agregar un hash del conjunto de verificación$_n$ 820' a ambas DLT.

Como resultado de este proceso, puede observarse que se mantiene un registro fuera de la cadena de una secuencia de transacciones que aparece en dos blockchains en virtud de los conjuntos de verificación en el registro de las transacciones 840. El registro fuera de la cadena es indicativo de una secuencia relativa de las transacciones a las que corresponden. Al usar también el "aspecto de mensajería" descrito anteriormente para almacenar en ambas DLT un indicador de hash de cada conjunto de verificación, se puede mantener un registro de cada conjunto de verificación en las DLT, que puede ser de uso particular para mantener la integridad y seguridad de datos del registro fuera de la cadena de conjuntos de verificación, y en el caso de una bifurcación en la primera DLT 12 y/o la segunda DLT 14, como se explica a continuación.

La figura 9 muestra una visualización de ejemplo adicional de dos DLT y conjuntos de verificación para ayudar a comprender las etapas del método de la figura 7, en particular las Etapas S730 a S760. Aunque la figura 9 es similar a la figura 8, la figura 9 incluye una representación de una bifurcación en la primera DLT 12, que provoca que se hayan perdido dos transacciones y un registro de un conjunto de verificación de la primera DLT 12.

En la figura 9, la primera DLT 12 se bifurca antes de la inclusión de la transacción C 826. La transacción C 826, el hash del conjunto de verificación$_n$ 820' y la transacción E 832 aparecen todos en la "Rama 1" de la bifurcación. Sin embargo, en este ejemplo, es la "Rama 2" de la bifurcación la que avanza como DLT 12 y los contenidos de la

"Rama 1" se pierden efectivamente (al menos temporalmente). Habiendo añadido el hash del conjunto de verificación$_n$ 820' a las DLT, se comienza a trabajar en el conjunto de verificación$_{n+1}$ 930, que como puede verse incluye un hash de la transacción E 832' y un hash de la transacción G 934', en ese orden. En el Punto P identificado en la figura 9, el proceso de la figura 7 va a la Etapa S730, momento en el que se identifica que el hash del conjunto de verificación$_n$ 820' y la transacción C 826 ya no aparecen en la primera DLT 12 (ya que ambos aparecieron en "Rama 1", que es la rama de la bifurcación que se ha perdido). Como resultado, en la Etapa S740, se reproduce la transacción C (identificada por el número de referencia 926 en la figura 9, aunque se apreciará que el contenido de la transacción original C 826 y de la transacción C 926 reproducida será el mismo) y luego el hash del conjunto de verificación$_n$ se reproduce (identificado por el número de referencia 920' en la figura 9, aunque se apreciará que el hash original del conjunto de verificación 820' será el mismo que el hash del conjunto de verificación reproducido 920'). A continuación, en la Etapa S750, se determina que la transacción E 832 (que se identifica en el conjunto de verificación "en curso"$_{n+1}$ 930) ya no aparece en la primera DLT 12 porque se incluyó en la "Rama 1". Por lo tanto, en la Etapa S760, se reproduce la transacción E (identificada por el número de referencia 932 en la figura 9, aunque se apreciará que el contenido de la transacción original E 832 y la transacción E 932 reproducida será el mismo). Después de eso, el hash del conjunto de verificación$_{n+1}$ 930' se añade a ambas DLT en la Etapa S770. Por lo tanto, se puede ver que aunque la Transacción C 826, el hash del conjunto de verificación$_n$ 820' y la transacción E 832 desaparecieron todos de la primera DLT 12, pueden, no obstante, volver a añadirse a la primera DLT 12 después de la transacción G 934 (que originalmente tuvo lugar después de la transacción C 826 y la transacción E 832) sin perder un registro fiable de la verdadera secuencia de las transacciones, garantizando así la seguridad y estabilidad del sistema. En consecuencia, aunque la inclusión de hashes de los conjuntos de verificaciones en las DLT es opcional (en una alternativa, el registro fuera de la cadena de transacciones puede mantenerse simplemente sin ningún hash de los conjuntos de verificación almacenados en las DLT), puede apreciarse que el almacenamiento de un registro de cada conjunto de verificación en las DLT no solo permite una seguridad y fiabilidad mejoradas en virtud de habilitar la integridad de los datos de los conjuntos de verificación fuera de la cadena, sino que también permite que las bifurcaciones se identifiquen rápidamente y que los registros y/o transacciones que faltan se vuelvan a añadir a la(s) DLT(s) correspondiente(s).

Se apreciará que mientras cada bloque representado en la figura 9 comprende una única transacción, en la práctica es probable que cada bloque comprenda muchas transacciones. Además, aunque lo anterior describe un proceso que usa dos DLT, se apreciará que puede aplicarse a una única DLT, o a tres o más DLT diferentes. Además, aunque en lo anterior se describen "hashes" de los conjuntos de verificación y transacción, se apreciará que puede usarse alternativamente cualquier identificador de conjunto de verificación adecuado determinado basado al menos en parte en el contenido de los conjuntos de verificación y cualquier identificador de transacción adecuado basado al menos en parte en el contenido de las transacciones.

Además, mientras que en el proceso descrito anteriormente los hashes de los conjuntos de verificación se añaden a ambas DLT, en una alternativa pueden añadirse a solo una de las DLT. Dicho esto, puede ser preferible incluir los hashes de los conjuntos de verificación en dos o más de las DLT en las que está operando el proceso, ya que incluso en el caso de una bifurcación, debería haber en todo momento al menos un registro del conjunto de verificación anterior más reciente (siendo extremadamente improbable que dos DLT se bifurquen independientemente al mismo tiempo, haciendo que ambas pierdan sus hashes del conjunto de verificación anterior más reciente). Esto puede ser útil por razones de integridad de datos y, por lo tanto, de seguridad.

Además, adicionalmente o como alternativa al almacenamiento en los conjuntos de verificación de hash de las transacciones correspondientes, puede almacenarse una copia completa de la transacción correspondiente. Sin embargo, puede ser preferible almacenar hashes de la transacción correspondiente, en aras de la eficiencia del almacenamiento de datos. En una implementación de ejemplo, pueden almacenarse hashes de cada una de las transacciones correspondientes y puede almacenarse una copia completa de transacciones correspondientes recientes (por ejemplo, transacciones en los conjuntos de verificación uno, dos, tres, cuatro, más recientes), lo que puede ser útil para reproducir transacciones que faltan. Almacenando únicamente las transacciones correspondientes recientes, aún pueden minimizarse los requisitos de almacenamiento de datos.

Además, identificar y reproducir transacciones que faltan es opcional, aunque puede ser preferible realizar esas etapas para garantizar que las DLT estén lo más actualizadas posible y que la secuenciación de las transacciones en las DLT corresponda lo más cerca posible a la secuenciación de la transacción real tal y como se identifica en el registro de la transacción 840.

Aspecto de la transacción cruzada entre ledgers

Para ayudar aún más en la comprensión de la importancia del aspecto de secuenciación de transacciones descrito anteriormente, se describirá ahora un proceso de transacción cruzada entre ledgers (XLT) de ejemplo que utiliza el aspecto de secuenciación de transacciones.

En este ejemplo, se desea transferir a la segunda DLT 14 información que aparece en una transacción en la primera DLT 12. Esa información puede ser cualquier cosa, por ejemplo, una cantidad de criptomoneda, o un mensaje de hash (como se describe en el "aspecto de mensajería" anterior), o cualquier otro dato. La transacción puede ser

transferir la información de un primer propietario a un segundo propietario, o puede no haber transferencia de propiedad en absoluto (por ejemplo, una entidad puede simplemente desear almacenar información en una DLT diferente y eliminarla efectivamente de la anterior DLT).

La figura 10 muestra una visualización de ejemplo de dos DLT, y conjuntos de verificación correspondientes que componen un registro de transacciones 1040 almacenadas fuera de la cadena, en las que se va a realizar una XLT. La transacción S 1012 comprende la información a transferir desde la primera DLT 12 a la segunda DLT 14. El conjunto de verificación$_{n-1}$ 1010 comprende un hash de la transacción S 1012'. La XLT puede llevarse a cabo usando un proceso de confirmación de dos fases. El experto en la materia comprenderá bien los procesos de confirmación de dos fases, pero a continuación se describen brevemente en aras de la exhaustividad. Una propuesta para la transferencia de la información en la transacción S desde la DLT 12 a la DLT 14 se añade en primer lugar a la DLT 12 (identificada por "XLT propuesta (S) 1022" en la figura 10). Esto bloquea eficazmente la transacción S 1012 de modo que no puede usarse en ninguna otra transacción. El destinatario previsto (que en algunas circunstancias puede ser la misma entidad que el propietario actual de la información que se va a transferir, como se ha explicado anteriormente) puede indicar posteriormente en la DLT 14 su disponibilidad para aceptar la transferencia (identificada por "XLT disponible (S) 1026"). En una alternativa, en algunos protocolos de DLT, la disponibilidad puede indicarse reproduciendo "XLT propuesta (S) 1022" en la segunda DLT 14. Una vez que se ha indicado la disponibilidad en la segunda DLT 14, la transacción cruzada entre ledgers puede ejecutarse mediante una transacción de confirmación que se añade a ambas DLT (identificada por "XLT de confirmación (S) 1032" y "XLT de confirmación (S) 1034" en la figura 10). Como entenderá bien el experto en la materia, esta confirmación elimina eficazmente el elemento objeto de la transacción de la primera DLT 12 y lo pone en la segunda DLT 14.

Para que una XLT se ejecute de forma segura y fiable, sin el riesgo de fraude o fallo del sistema, antes de que pueda tener lugar la XLT, puede ser necesario garantizar que aparezca una indicación de disponibilidad en la segunda DLT 14 antes de que se introduzca la transacción de confirmación en las dos DLT. Además, puede ser preferible que la indicación de estar disponible siga siempre la propuesta, de modo que las dos puedan emparejarse y autenticarse adecuadamente. De esta manera, se puede garantizar que la indicación de disponibilidad no se relacione de hecho con alguna otra transacción similar, pero diferente (por ejemplo, relacionada con una propuesta de XLT entre las mismas partes, pero para los contenidos de una transacción anterior diferente en DLT 12). También se puede asegurar que la indicación de disponibilidad no cambia de manera fraudulenta algunos de los detalles de la XLT.

Anteriormente, como se ha explicado antes, determinar la secuencia correcta de transacciones en diferentes DLT ha sido muy difícil, particularmente en el caso de una bifurcación en una o más de las DLT. En consecuencia, ha sido difícil llevar a cabo XLT de forma segura, sencilla y fiable. Sin embargo, utilizando el aspecto de secuenciación de transacciones descrito anteriormente, puede registrarse la secuencia correcta de las transacciones correspondientes, incluso en el caso de una bifurcación en una o más de las DLT.

Por ejemplo, el aspecto de secuenciación de transacciones puede usarse para garantizar que se cumplen los criterios de XLT, tal como tanto la XLT propuesta (S) 1022 correspondiente a la XLT disponible (S) 1026 (y opcionalmente ambas que siguen apareciendo en la primera DLT 12 y la segunda DLT 14 respectivamente) y que la XLT propuesta (S) 1022 precede a la XLT disponible (S) (incluso en el caso de una bifurcación). En este sentido, la transacción de confirmación XLT (S) 1032 puede añadirse únicamente a las DLT después de que se hayan cumplido los criterios de la XLT. Opcionalmente, los criterios de la XLT también pueden comprender un criterio de que el hash del conjunto o conjuntos de verificación que identifican la XLT propuesta (S) 1022 y las transacciones de XLT disponibles (S) 1026 están presentes en una o ambas DLT, que a su vez aseguraría que las transacciones de tanto la XLT propuesta (S) 1022 como la XLT disponible (S) 1026 están presentes en las respectivas DLT. Se apreciará que mientras en este ejemplo el hash de la XLT propuesta (S) 1022' y el hash de la XLT disponible (S) 1026' aparecen en el mismo conjunto de verificación, pueden aparecer alternativamente en diferentes conjuntos de verificación. Si el hash de la XLT propuesta (S) 1022' aparece en un conjunto de verificación que precede al conjunto de verificación en el que aparece el hash de la XLT disponible (S) 1026', entonces puede confirmarse la secuencia correcta. Si el hash de la XLT propuesta (S) 1022' aparece en un conjunto de verificación que sigue al conjunto de verificación en el que aparece el hash de la XLT disponible (S) 1026', se puede confirmar que las transacciones están en la secuencia incorrecta, por lo que la XLT no debería tener lugar.

Como se puede ver en la figura 10, el conjunto de verificación$_n$ 1020 comprende un hash de la XLT propuesta (S) 1022", y va seguido de un hash de la XLT disponible (S) 1026' (el conjunto de verificación$_n$ 1020 también comprende hashes de otras transacciones que se consideran correspondientes, pero que no están relacionadas con la XLT que se analiza en el presente documento. En la práctica, es probable que una entidad que construye los conjuntos de verificación identifique muchas transacciones correspondientes en las DLT e identifique las mismas en los conjuntos de verificación. Algunas de esas transacciones correspondientes pueden relacionarse entre sí, tales como transacciones relacionadas con una XLT particular, y otras no se relacionarán entre sí de ninguna manera, por ejemplo, en relación con diferentes XLT, o cualquier otro tipo de transacción de DLT que la entidad considere de relevancia). El hash del conjunto de verificación$_n$ 1020' aparece en ambas DLT y posteriormente le sigue la XLT de confirmación (S) 1032 en la primera DLT 12 y la XLT de confirmación (S) 1034 en la segunda DLT 14 (que a continuación se identifican en el siguiente conjunto de verificación$_{n+1}$ 1030).

Las XLT de este tipo pueden implementarse de muchas maneras diferentes. La figura 11 muestra un ejemplo de una implementación particular, que comprende el desarrollador de la oferta de servicio 310, que en este ejemplo particular ha desarrollado una aplicación de servicio (o paquete/módulo de software) 1110 para llevar a cabo XLT, y también ha proporcionado una BPI al desarrollador de la aplicación cliente 320, para que pueda desarrollar su aplicación cliente de XLT particular (de manera alternativa, el desarrollador de la oferta del servicio 310 puede crear la aplicación cliente por su cuenta, en cuyo caso puede que no se ofrezca una BPI a desarrolladores externos). La primera aplicación cliente 1120 y la segunda aplicación cliente 1130 son duplicados de la aplicación cliente de la XLT que una primera entidad 1140 y una segunda entidad 1150 han obtenido respectivamente del desarrollador de la aplicación cliente 320. Las aplicaciones cliente en este ejemplo pueden implementar eficazmente la capa funcional de la aplicación 30 y el paquete o módulo de software 1110 puede implementar eficazmente la capa de comunicaciones del overledger 20.

La figura 12 muestra etapas de ejemplo de cómo el sistema de ejemplo representado en la figura 11 puede llevar a cabo una XLT.

En la Etapa S1210, la aplicación de servicio 1110 recibe una solicitud de la primera aplicación cliente 1120 para llevar a cabo una XLT para transferir un elemento de información que aparece en la primera DLT 12 a través de la segunda DLT 14 para transferir la propiedad a la segunda entidad 1150. Como se apreciará, la solicitud puede comprender cualquier información necesaria para realizar la XLT, como se define en la BPI.

En la Etapa S1220, la aplicación de servicio 1110 puede difundir una transacción XLT propuesta a la primera red de DLT 240, para añadirla a la primera DLT 12.

En la Etapa S1230, la aplicación de servicio 1110 puede a continuación difundir una transacción XLT disponible a la segunda red de DLT 250, para añadir a la segunda DLT 14. Opcionalmente, la aplicación de servicio 1110 puede haber comunicado la intención de realizar la XLT a la segunda aplicación cliente 1130 y a continuación únicamente realizar la Etapa S1220 o S1230 después de recibir la confirmación de la segunda aplicación 1130 de que debería tener lugar la transacción.

Mientras tanto, la aplicación de servicio 1110 está realizando el aspecto de secuenciación de transacciones descrito anteriormente y almacenando un registro de transacciones correspondientes que es indicativo de la secuencia de las transacciones correspondientes en un almacén de datos 1115.

En la Etapa S1240, la aplicación de servicio 1110 puede comprobar que la transacción XLT propuesta se identifica en el registro de transacción almacenado como si hubiera tenido lugar antes de la correspondiente transacción XLT disponible, y opcionalmente también que ha aparecido el hash del conjunto o conjuntos de verificación correspondientes en una o ambas DLT.

Una vez superada esta comprobación, en la Etapa S1250 la aplicación de servicio 1110 difunde la transacción XLT de confirmación a la primera red de DLT 240 y a la segunda red de DLT 250 para añadir a la primera y segunda DLT. Opcionalmente, la aplicación de servicio 1110 puede notificar a la primera aplicación cliente 1120 y/o a la segunda aplicación cliente 1130 de una XLT satisfactoria (que puede realizarse después de que la transacción XLT de confirmación haya aparecido en ambas DLT, o después de que el hash del conjunto de verificación establecido en el que se identifica la transacción XLT de confirmación se ha incluido en una o ambas DLT). En el caso de que no se supere la comprobación, después de un periodo de tiempo el proceso de la XLT puede agotarse y es posible que se le notifique un fallo a la primera aplicación cliente 1120 y/o la segunda aplicación cliente 1130.

Por lo tanto, puede observarse que al realizar el aspecto de secuenciación descrito anteriormente, y luego solo publicar la transacción de confirmación en las DLT después de que se haya verificado que la transacción XLT propuesta y la correspondiente transacción XLT disponible tienen lugar en la secuencia correcta en los conjuntos de verificación, la aplicación de servicio 1110 puede garantizar que la XLT se lleva a cabo de forma segura y estable.

Se apreciará que este es simplemente un ejemplo particular no limitante. En una alternativa, la primera aplicación cliente 1120 puede difundir por sí misma la transacción XLT propuesta a la primera red de DLT 240 y/o la segunda aplicación cliente 1130 puede difundir por sí misma la transacción XLT disponible a la segunda red de DLT 250, dependiendo de la BPI particular ofrecida por el desarrollador de la oferta de servicio 310 y/o la implementación de la aplicación cliente ofrecida por el desarrollador de la aplicación cliente 320. En este caso, la aplicación de servicio 1110 puede simplemente realizar el aspecto de secuenciación de transacciones descrito anteriormente, y a continuación llevar a cabo las Etapas S1140 y S1150 cuando ha identificado una transacción XLT propuesta y una transacción XLT disponible correspondiente en las DLT correspondientes. Además, la XLT puede implicar únicamente una entidad, ya que pueden estar simplemente moviendo su información de una DLT a otra DLT, con cualquier cambio en la propiedad.

Se apreciará que el SDK del overledger puede comprender las reglas y métodos para llevar a cabo el aspecto de secuenciación de transacciones y opcionalmente también el aspecto de la XLT. Como tal, el SDK del overledger

ofrece las herramientas mediante las cuales el desarrollador de la oferta de servicio 310 puede implementar cualquier aplicación/módulo de servicio que utilice el aspecto de secuenciación de transacciones (y opcionalmente también el aspecto de la XLT) en cualquier DLT de su elección que sea compatible con el SDK del overledger. A continuación, pueden crear una BPI correspondiente de su elección, para permitir que los desarrolladores de aplicaciones cliente 320 utilicen cualquier funcionalidad adecuada de la oferta de servicio, de modo que las interacciones con las DLT al realizar el aspecto de secuenciación pueden realizarse mediante la capa de comunicaciones del overledger 20 y desacoplarse de cualquier operación de capa funcional de la aplicación 30 en las aplicaciones cliente. En consecuencia, las aplicaciones cliente pueden utilizar los beneficios de seguridad y estabilidad proporcionados por el aspecto de la secuenciación de transacciones, mientras aún están desacopladas de la interacción con las DLT directamente.

Como resultado, se puede ver que al ofrecer un SDK del overledger que incluye las reglas y métodos para llevar a cabo el aspecto de secuenciación de transacciones, existe una gran flexibilidad y oportunidad para que los beneficios técnicos del aspecto de secuenciación de transacciones sean utilizados por una amplia variedad de diferentes entidades, de diferentes maneras, mientras aún se desacoplan interacciones de blockchain de la capa funcional de la aplicación 30.

El experto en la materia apreciará fácilmente que pueden realizarse diversas alteraciones o modificaciones a los aspectos descritos anteriormente de la divulgación sin apartarse del alcance de la divulgación.

Aunque todas las interfaces representadas por flechas en las figuras 2, 3, 5 y 11 muestran conexiones directas entre cada una de las diferentes entidades y módulos, se apreciará que puede haber cualquier número de entidades o módulos intermedios como parte de esas interfaces, por ejemplo, enrutadores de comunicaciones, etc. Además, cada una de las interfaces puede transportar datos/comunicaciones entre cada una de las entidades y módulos de acuerdo con cualquier arquitectura y protocolo de comunicaciones adecuados.

Los aspectos de la presente divulgación descritos en todo lo anterior pueden implementarse mediante software, hardware o una combinación de software y hardware. Por ejemplo, el aspecto de la secuenciación de transacciones y/o el aspecto de XLT y/o la capa de mensajería del overledger puede implementarse mediante software que comprende código legible por ordenador, que cuando se ejecuta en uno o más procesadores (tal como uno o más microprocesadores, o lógica programable) de cualquier dispositivo electrónico, realiza la funcionalidad descrita anteriormente. El software puede almacenarse en cualquier medio legible por ordenador adecuado, por ejemplo, un medio legible por ordenador no transitorio, tal como memoria de solo lectura, memoria de acceso aleatorio, CD-ROM, DVD, Blue-ray, cinta magnética, unidades de disco duro, unidades de estado sólido y unidades ópticas. El medio legible por ordenador puede distribuirse a través de sistemas informáticos acoplados a red de modo que las instrucciones legibles por ordenador se almacenan y ejecutan de una manera distribuida.

REIVINDICACIONES

1. Un método implementado por ordenador para mantener un registro de una secuencia de transacciones que aparecen cada una en una cualquiera de una pluralidad de blockchains (12, 14), comprendiendo el método:

identificar una pluralidad de transacciones, en donde al menos una de las transacciones en la pluralidad de transacciones está incluida en una primera blockchain de la pluralidad de blockchains y al menos otra de las transacciones en la pluralidad de transacciones está incluida en una segunda blockchain de la pluralidad de blockchains, en donde la primera blockchain opera de acuerdo con un primer protocolo de blockchain y la segunda blockchain opera de acuerdo con un segundo protocolo de blockchain que es diferente al primer protocolo de blockchain; y

almacenar un registro (840, 1040) de la pluralidad de transacciones en un almacén de datos, en donde el registro es indicativo de la secuencia relativa de la pluralidad de transacciones, en donde el registro de la pluralidad de transacciones comprende una pluralidad de conjuntos de verificación consecutivos (810, 820, 830, 930, 1010, 1020, 1030), y en donde al menos algunos de la pluralidad de conjuntos de verificación consecutivos comprenden uno o más identificadores de transacción que corresponden a una o más respectivas de la pluralidad de transacciones;
almacenar, en al menos una de la pluralidad de blockchains, un registro (810', 820', 930', 1010', 1020', 1030') del conjunto de verificación más reciente de la pluralidad de conjuntos de verificación consecutivos, en donde el registro del conjunto de verificación más reciente comprende un identificador de conjunto de verificación que se determina basándose al menos en parte en los contenidos del conjunto de verificación más reciente y es para mantener la integridad de los datos del conjunto de verificación más reciente;
comprobar que cada una de las transacciones identificadas en el conjunto de verificación más reciente está presente en la al menos una de la pluralidad de blockchains, y
si una o más de las transacciones identificadas en el conjunto de verificación más reciente no están presentes en su blockchain correspondiente de la pluralidad de blockchains: reproducir la una o más transacciones en su blockchain correspondiente.

2. El método de la reivindicación 1, en donde identificar la pluralidad de transacciones comprende:
leer el contenido de cada nuevo bloque añadido a cada una de la pluralidad de blockchains.

3. El método de la reivindicación 2, en donde identificar la pluralidad de transacciones comprende además:
comparar las transacciones en cada nuevo bloque con un criterio de relevancia, en donde la pluralidad identificada de transacciones comprende transacciones que cumplen el criterio de relevancia.

4. El método de la reivindicación 2 o la reivindicación 3, en donde almacenar el registro de la pluralidad de transacciones en el almacén de datos comprende, cada vez que se identifica una transacción en un nuevo bloque, almacenar un registro de esa transacción identificada en el almacén de datos, en donde la secuencia relativa de la pluralidad de transacciones es la secuencia relativa en la que se identificó cada una de la pluralidad de transacciones.

5. El método de cualquier reivindicación anterior, que comprende además:

comprobar que un registro del conjunto de verificación anterior en la pluralidad de conjuntos de verificación está presente en la al menos una de la pluralidad de blockchains, y
si un registro del conjunto de verificación anterior no está presente en la al menos una de la pluralidad de blockchains;
reproducir un registro del conjunto de verificación anterior en la al menos una de la pluralidad de blockchains.

6. El método de la reivindicación 5, que comprende además:

si un registro del conjunto de verificación anterior no está presente en la al menos una de la una o más blockchains:
comprobar que cada una de las transacciones identificadas en el conjunto de verificación anterior está presente en su blockchain correspondiente de la una o más blockchains, y
si una o más de las transacciones identificadas en el conjunto de verificación anterior no están presentes en sus blockchains correspondientes de la una o más blockchains:
reproducir la una o más transacciones en sus blockchains correspondientes.

7. El método de cualquier reivindicación anterior, en donde el identificador de conjunto de verificación del conjunto de verificación más reciente comprende un hash de los contenidos del conjunto de verificación más reciente.

8. El método de cualquier reivindicación anterior, en donde el registro de la pluralidad de transacciones comprende una pluralidad de identificadores de transacción que se determinan cada uno basándose al menos en parte en los contenidos de las respectivas transacciones.

9. Un sistema que comprende:

uno o más procesadores; y
una memoria que almacena un programa de software, en donde el programa de software, cuando se ejecuta por el uno o más procesadores, hace que el uno o más procesadores realicen un método de acuerdo con cualquier reivindicación anterior.

10. Uno o más programas informáticos que, cuando se ejecutan por uno o más procesadores, hacen que el uno o más procesadores lleven a cabo un método de acuerdo con cualquiera de las reivindicaciones 1 a 8.

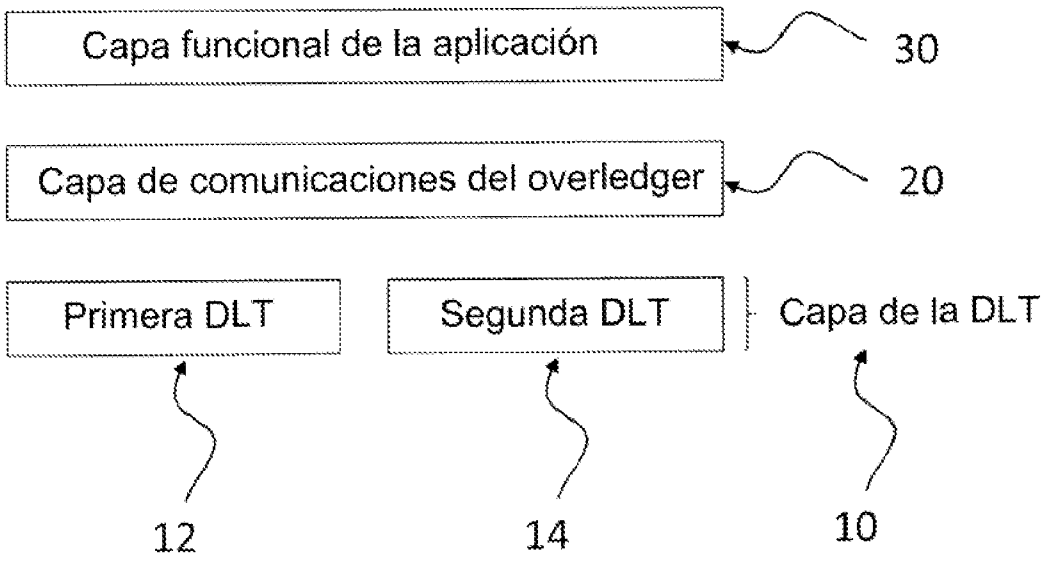
Capa funcional de la aplicación
30
Capa de comunicaciones del overledger
20
Primera DLT
Segunda DLT
Capa de la DLT
12
14
10

Fig. 1

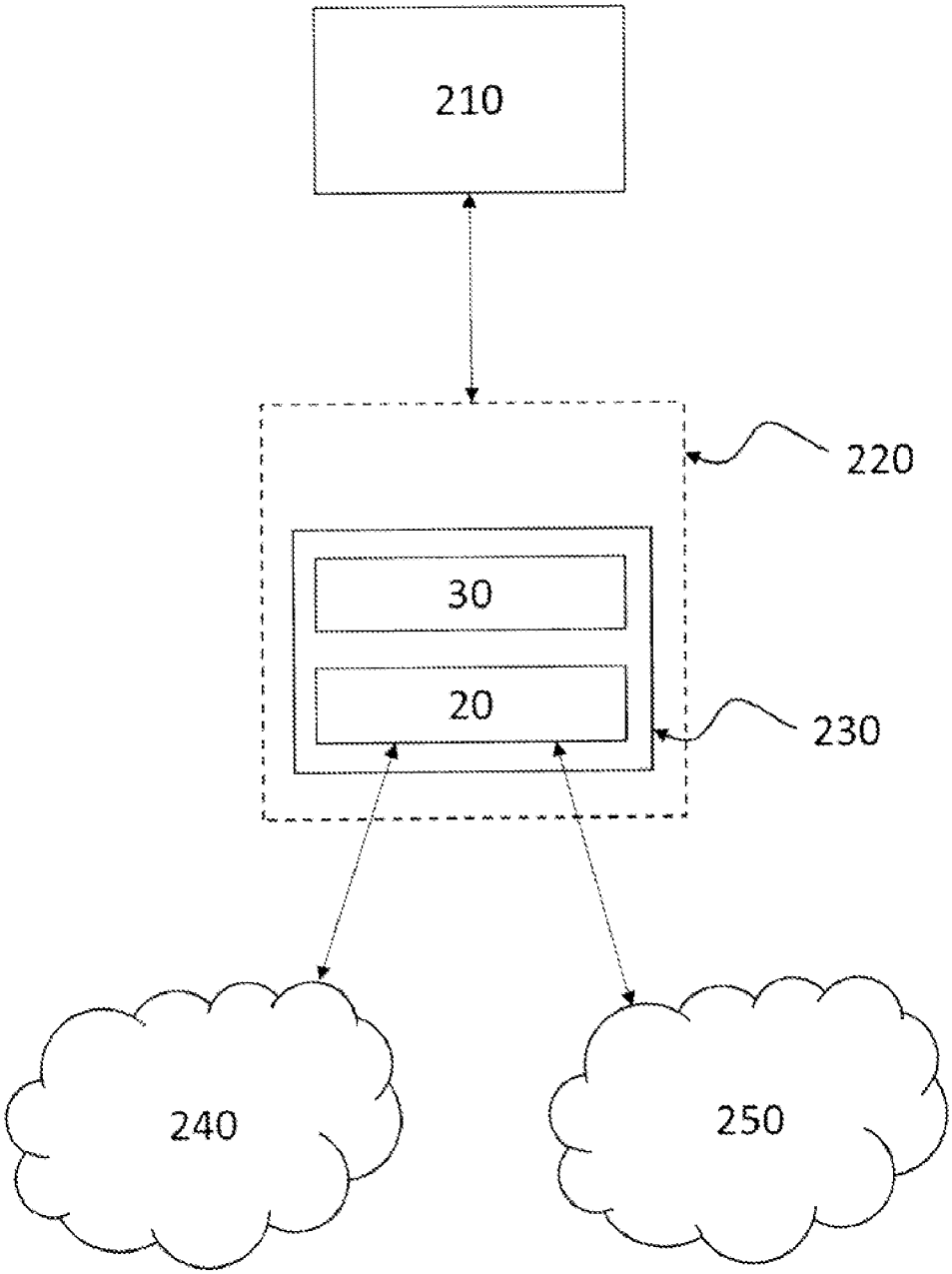

Fig. 2

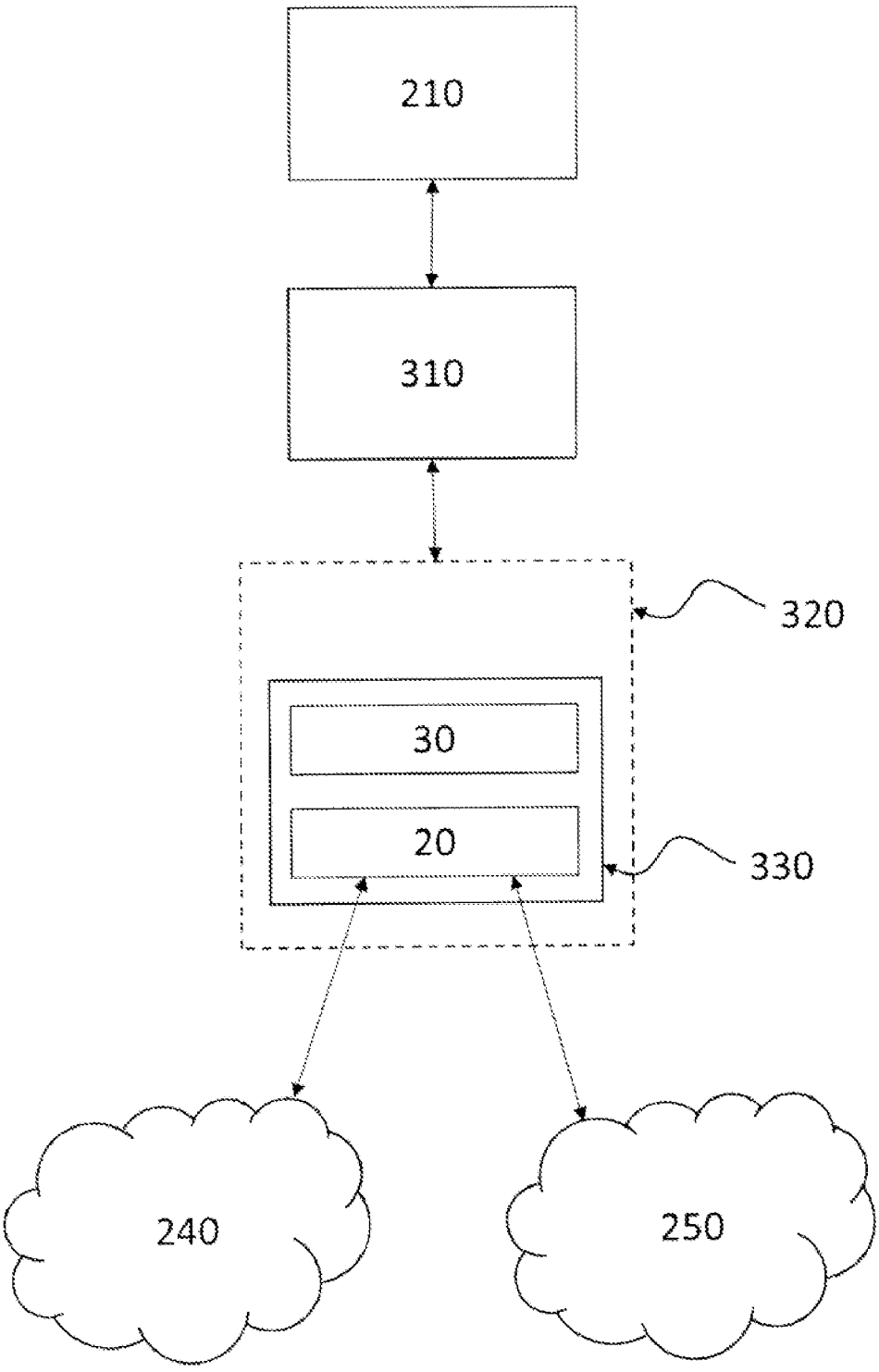

Fig. 3

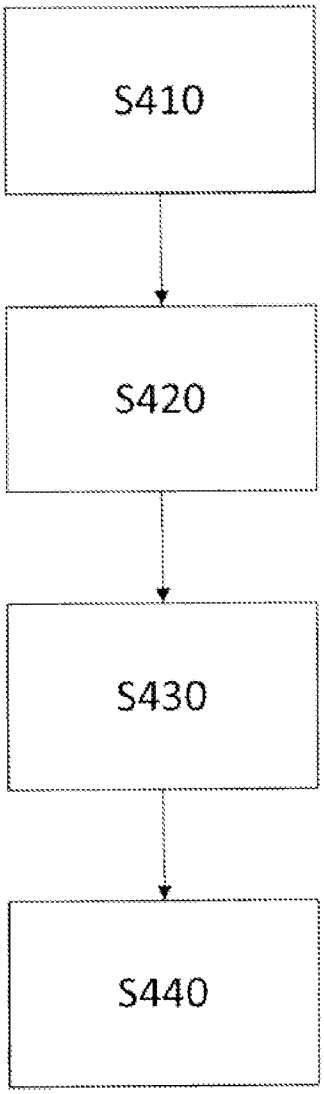

Fig. 4

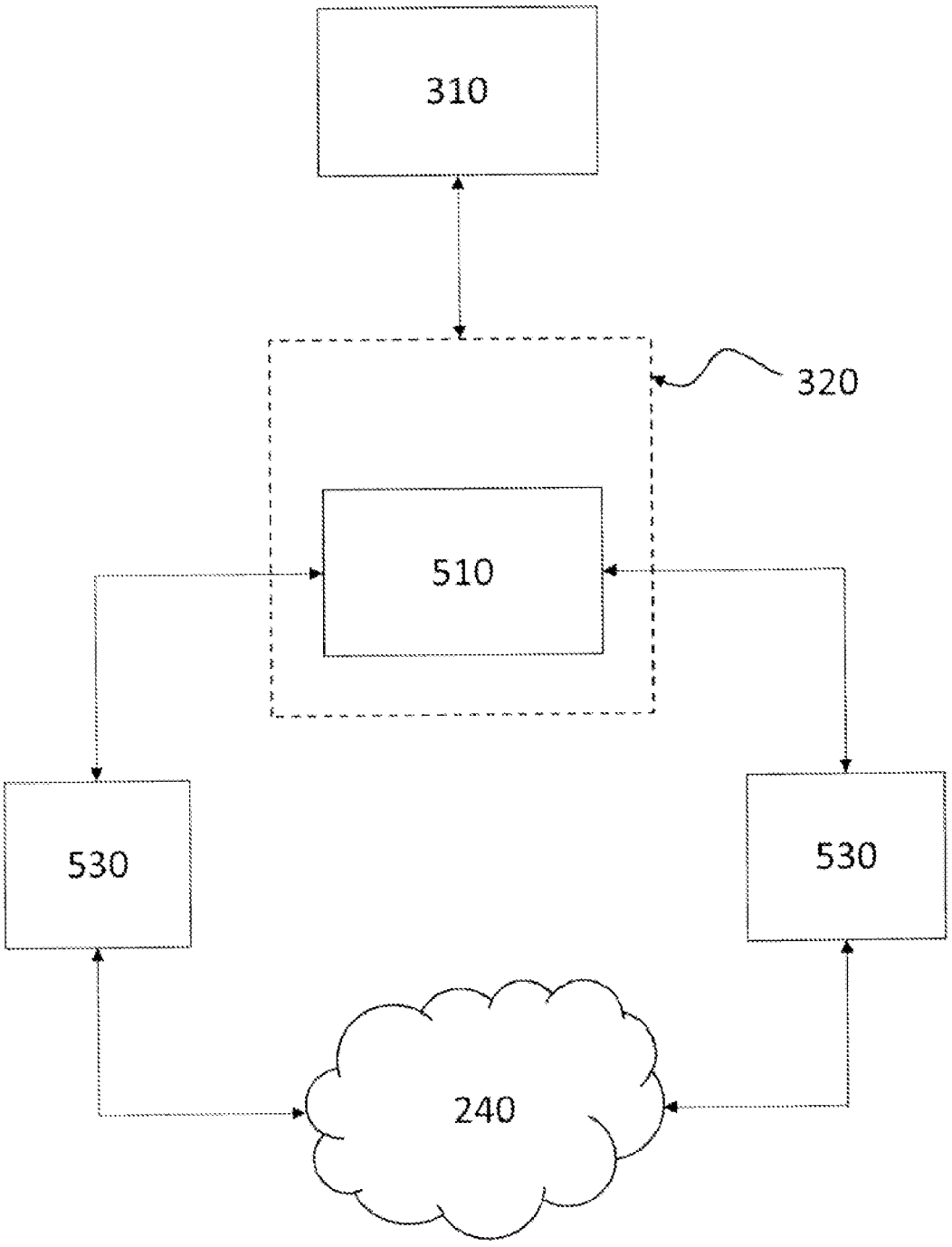

Fig. 5

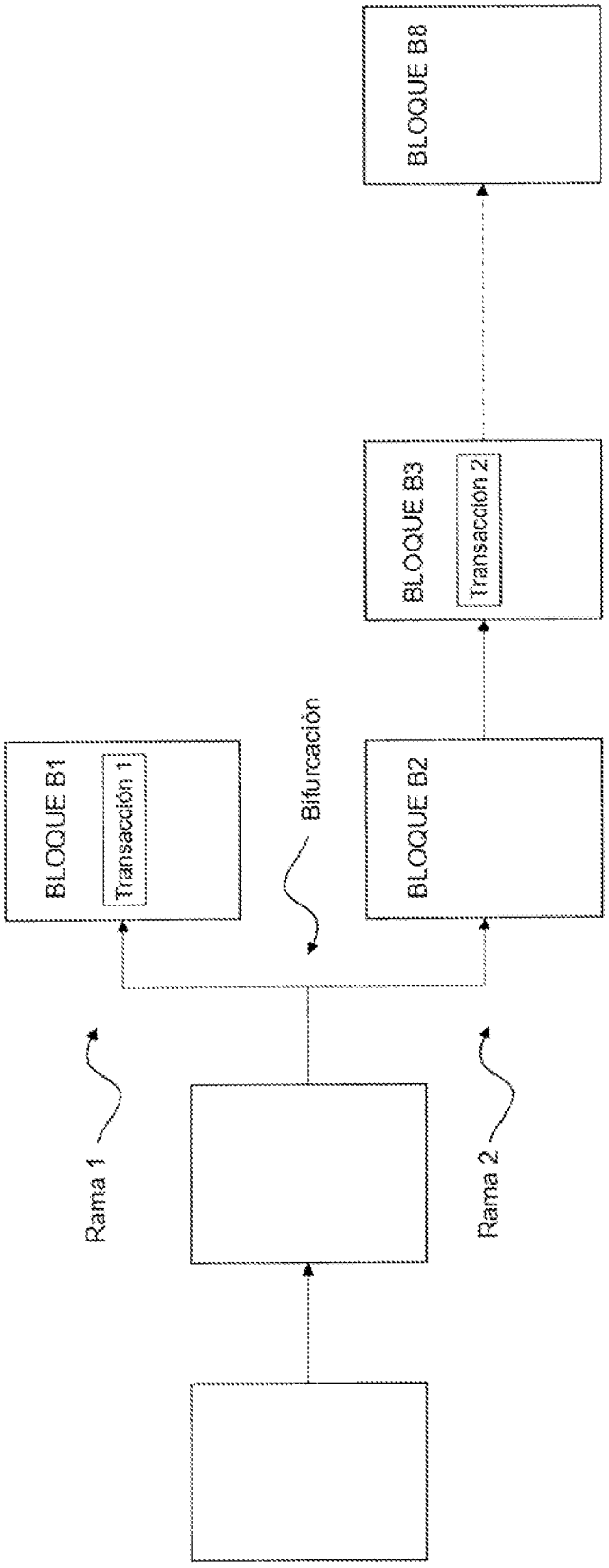
BLOQUE B8
BLOQUE B3
Transacción 2
BLOQUE B1
Transacción 1
Bifurcación
BLOQUE B2
Rama 1
Rama 2

Fig. 6

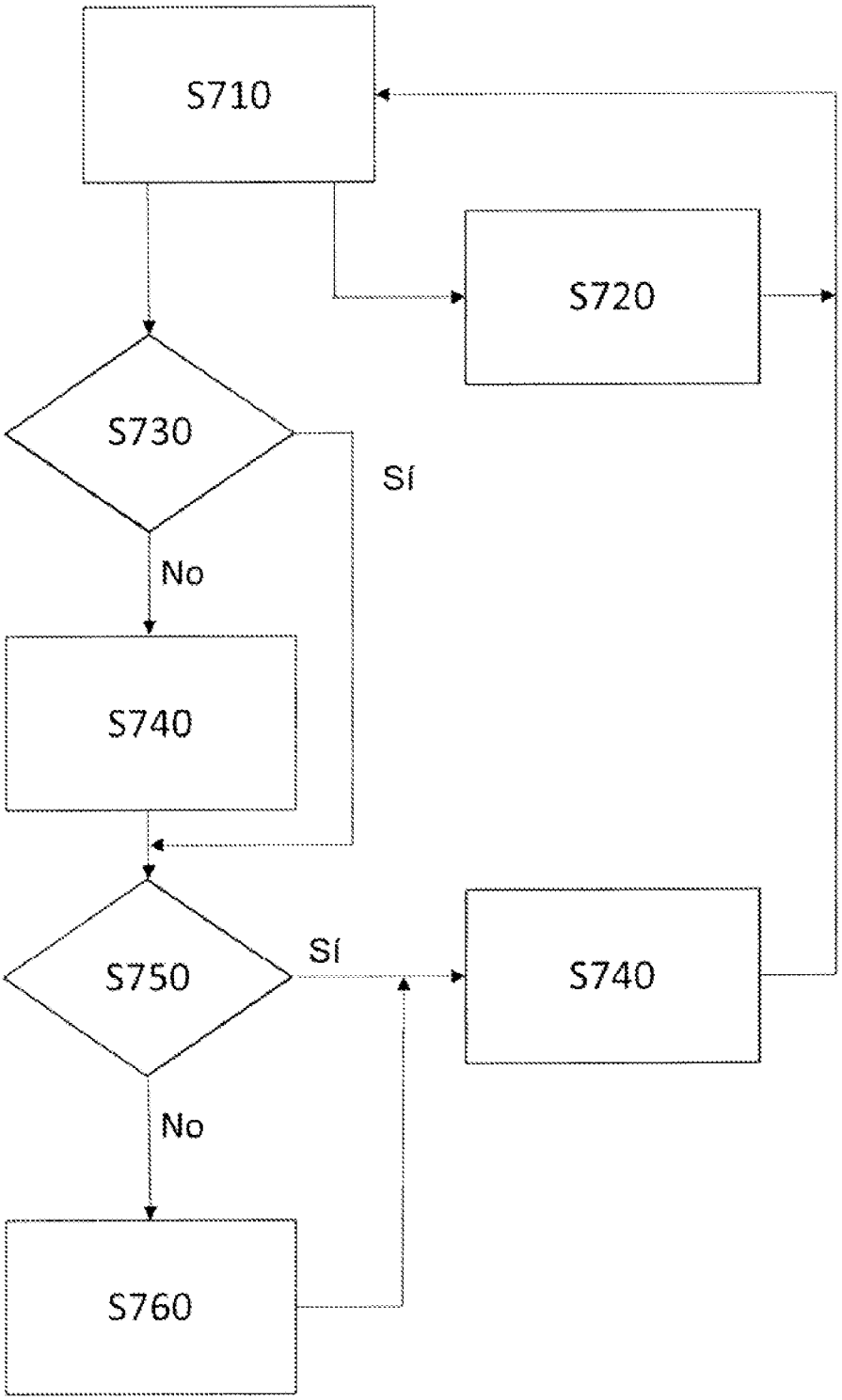

Fig. 7

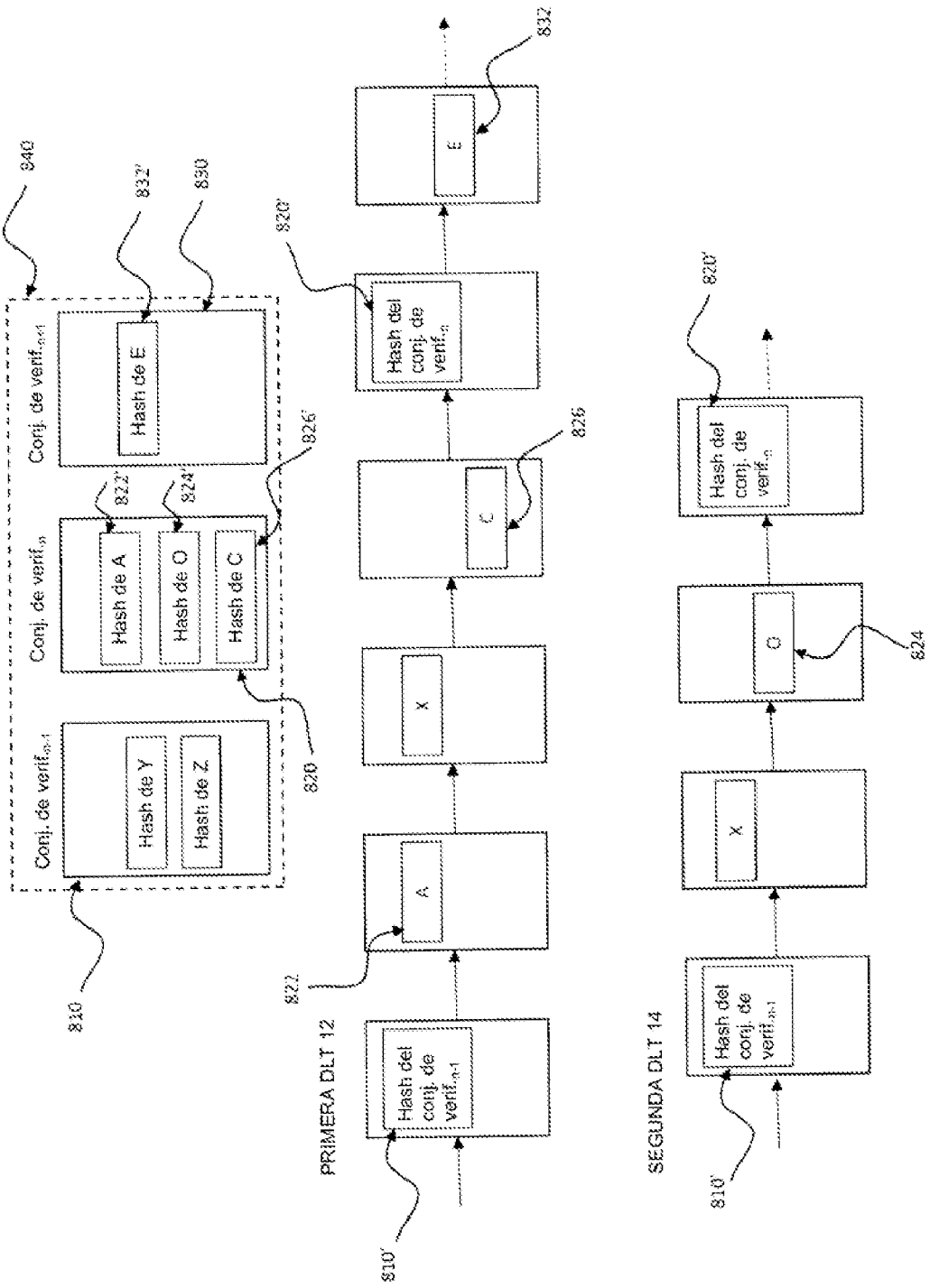
840
832'
850
820'
832
Conj. de verif.n+1
Hash de E
826
822'
824'
Conj. de verif.n
Hash de A
Hash de O
Hash de C
820
Conj. de verif.n-1
Hash de Y
Hash de Z
810
822
PRIMERA DLT 12
Hash del conj. de verif.n-1
A
X
C
E
Hash del conj. de verif.n
810'
SEGUNDA DLT 14
Hash del conj. de verif.n-1
X
O
824
Hash del conj. de verif.n
820'

Fig. 8

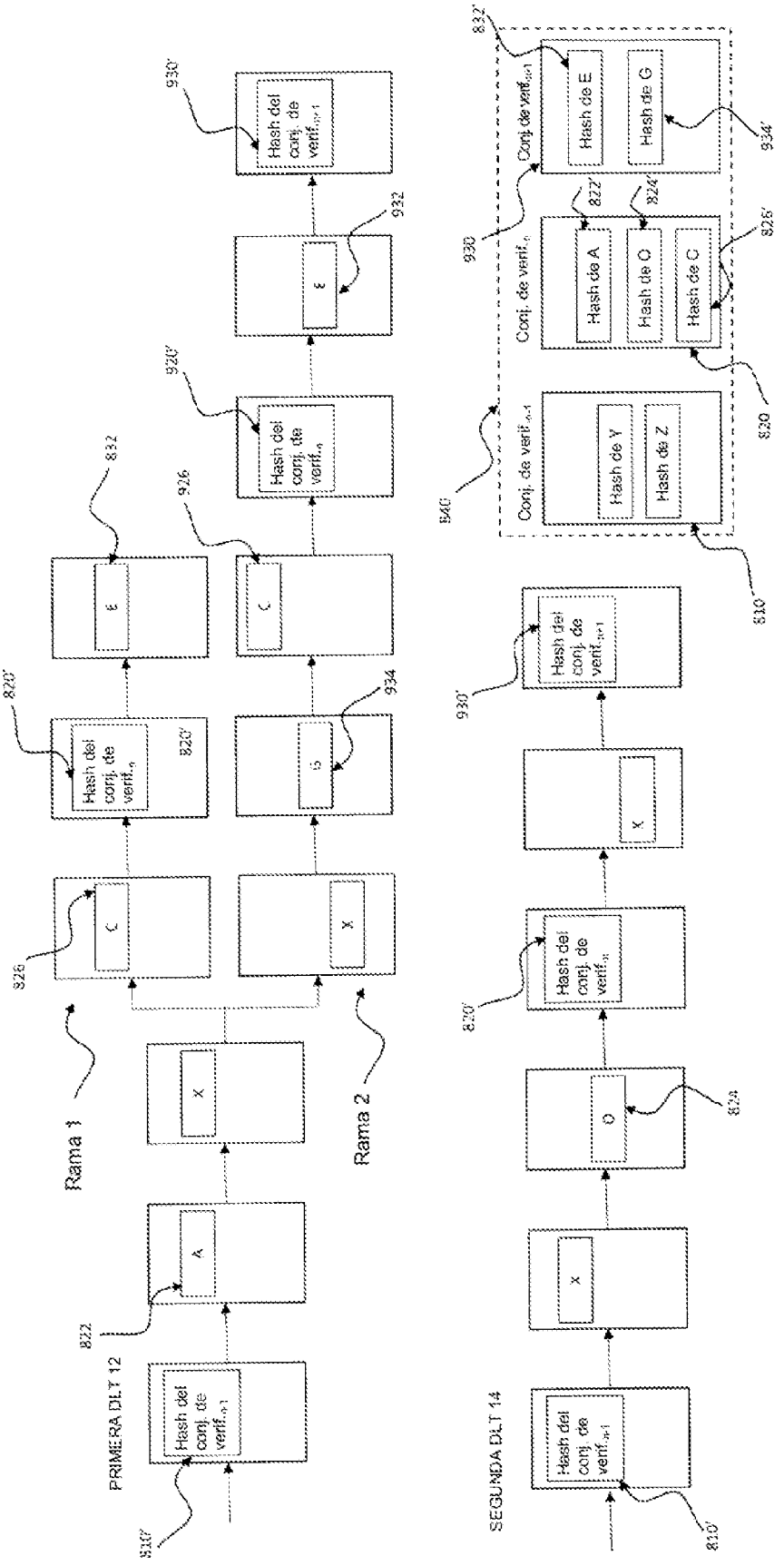
PRIMERA DLT 12
SEGUNDA DLT 14
Rama 1
Rama 2
Hash del conj. de verif.
Conj. de verif.
Hash de A
Hash de C
Hash de E
Hash de G
Hash de Y
Hash de Z
X
C
E
O

Fig. 9

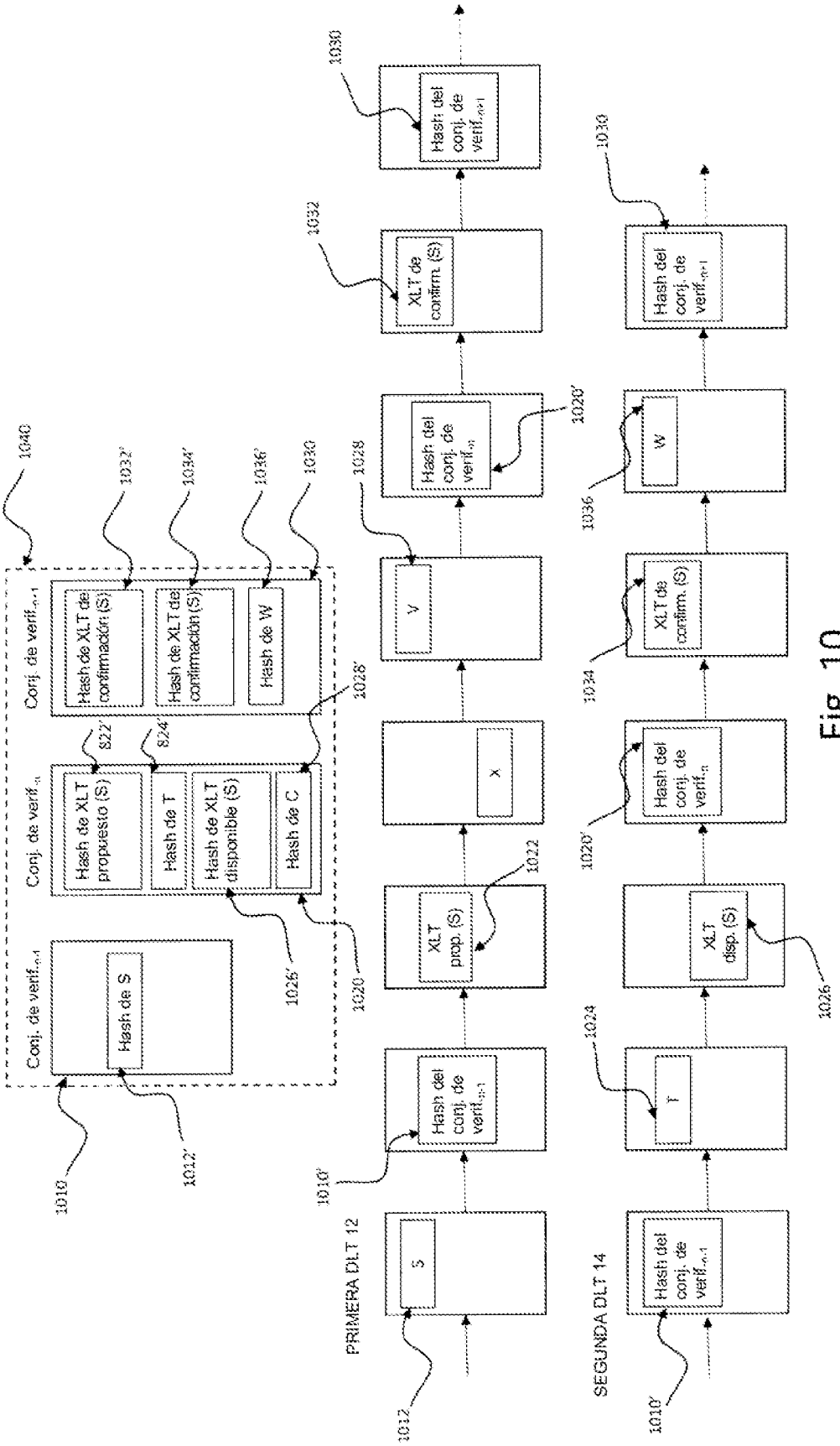
PRIMERA DLT 12
SEGUNDA DLT 14
Conj. de verif.$_{n-1}$
Conj. de verif.$_n$
Hash de S
Hash de XLT propuesto (S)
Hash de T
Hash de XLT disponible (S)
Hash de C
Hash de XLT de confirmación (S)
Hash de XLT de confirmación (S)
Hash de W
S
T
X
V
W
XLT prop. (S)
XLT disp. (S)
XLT de confirm. (S)
XLT de confirm. (S)
Hash del conj. de verif.$_{n-1}$
Hash del conj. de verif.$_n$
1010
1010'
1012
1012'
1020
1020'
1022
1024
1025'
1026
1028
1028'
1030
1030'
1032
1032'
1034
1034'
1036
1036'
1040
S22'
S24'

Fig. 10

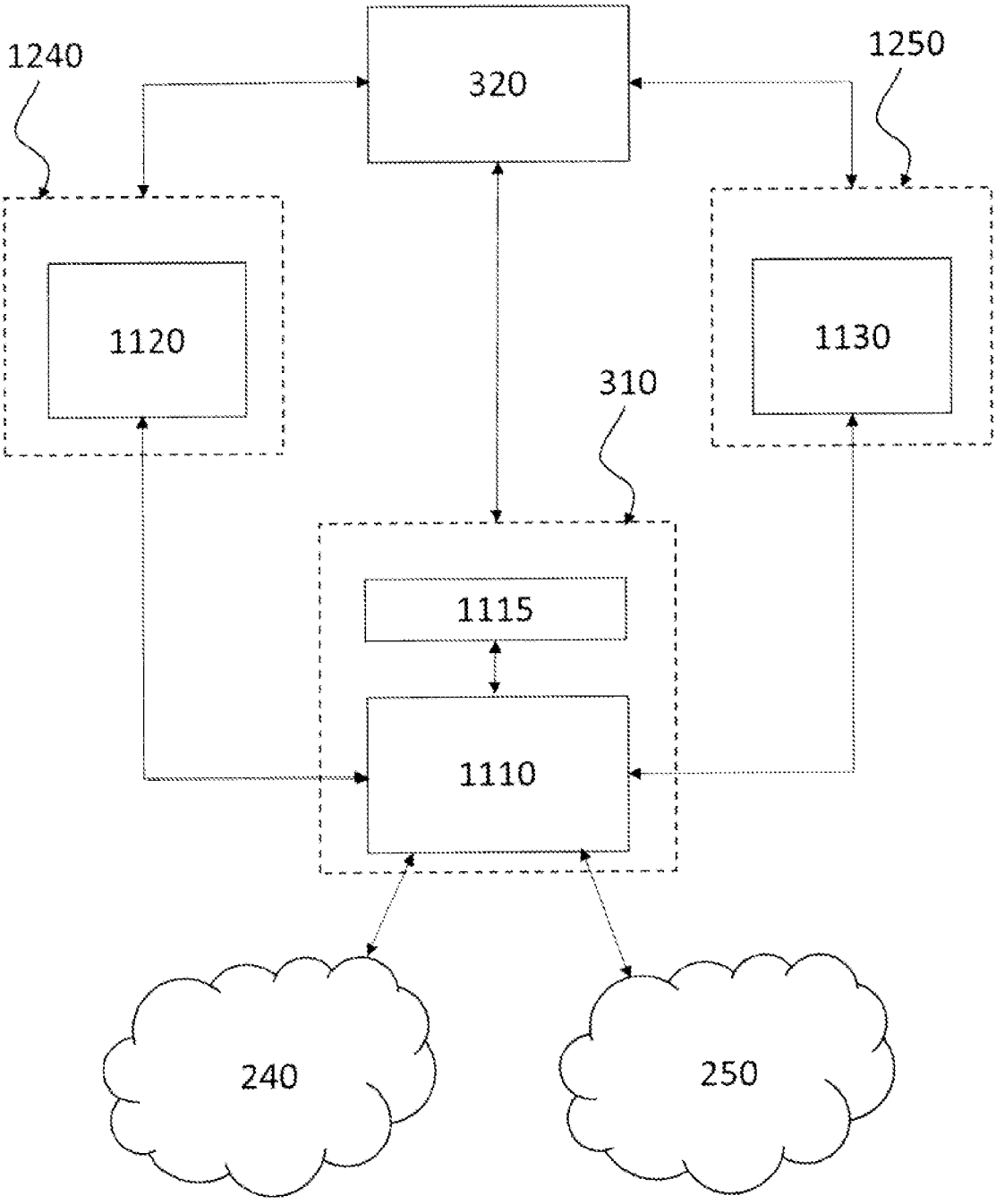

Fig. 11

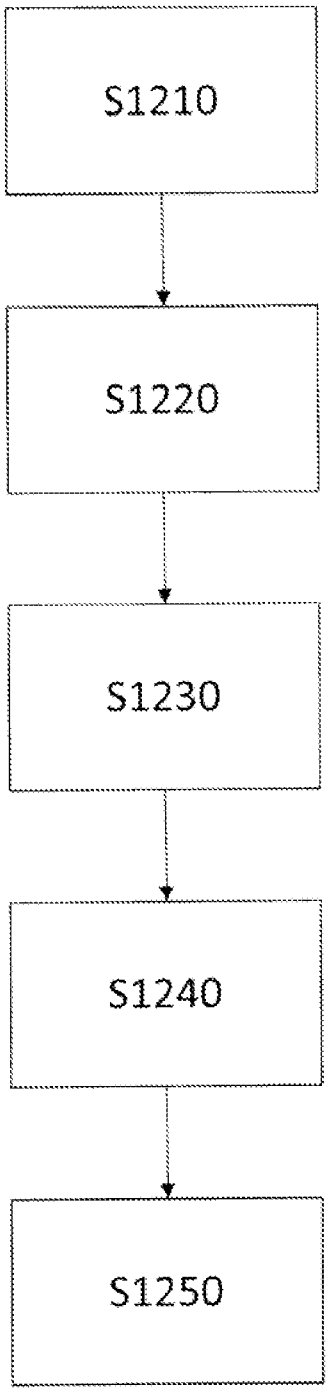

Fig. 12