ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Máster

Aplicación de la tecnología blockchain a soluciones de la Internet de las Cosas

(On blockchain and its usage in Internet of the Things networks)

Para acceder al Título de

Máster en Ingeniería de Tecnologías de Telecomunicación

Autor: Miguel Gómez Carpena

Octubre - 2018

E.T.S. DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACION

MÁSTER EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE MÁSTER

Realizado por: Miguel Gómez Carpena
Director del TFG: Jorge Lanza Calderón
Título: "**Aplicación de la tecnología blockchain a soluciones de la Internet de las Cosas**"
Title: "On blockchain and its usage in Internet of the Things networks "

Presentado a examen el día: 26 de Octubre de 2018

para acceder al Título de

MÁSTER EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre):
Secretario (Apellidos, Nombre):
Vocal (Apellidos, Nombre):

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del TFG
(sólo si es distinto del Secretario)

E.T.S. DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACION

Vº Bº del Subdirector

Trabajo Fin de Máster Nº
(a asignar por Secretaría)

Resumen

La tendencia actual es dotar cada vez de más inteligencia a las ciudades siguiendo el paradigma del Internet de las Cosas haciendo realidad el concepto de Smart City. Estas redes generan muchos datos que son almacenados en servidores de forma centralizada, rompiendo las características de un sistema distribuido como es la IoT.

La irrupción de la tecnología Blockchain se está aprovechando en multitud de aplicaciones más allá de la más conocida, las criptomonedas. Entre las diferentes opciones, Blockchain se está empleando en el despliegue de entornos totalmente descentralizados que rompen con el tradicional modelo cliente servidor. No solo se aplican al almacenaje distribuido, sino también a la ejecución distribuida de aplicaciones.

En este trabajo, se trata de aplicar los fundamentos de Blockchain para crear una aplicación descentralizada que permita la gestión de datos generados en una ciudad inteligente, de forma que, sin necesidad de llevarla a servidores, dicha información sea almacenada y esté disponible en la red.

Haciendo uso de soluciones de código y hardware abierto, se busca diseñar, implementar y evaluar un sistema que habilite las funcionalidades anteriormente descritas.

Abstract

The current trend is to provide more and more intelligence to cities following the paradigm of the Internet of Things making the concept of Smart City a reality. These networks generate many data that are stored in servers in a centralized way, breaking the characteristics of a distributed system such as the IoT.

The irruption of Blockchain technology is taking advantage of many applications beyond the best known, cryptocurrencies. Among the different options, Blockchain is being used in the deployment of totally decentralized environments that break with the traditional server client model. They not only apply to distributed storage, but also to distributed application execution.

This project, it is about applying the fundamentals of Blockchain to create a decentralized application that allows the management of data generated in an smart city, so that, without having to take it to servers, this information is stored and available on the network .

Making use of open source code and hardware solutions, we seek to design, implement and evaluate a system that enables the functions described above.

Índice

Índice de figuras

1. Introducción

El modelo en el que se ha basado Internet casi desde su creación en 1983 no ha cambiado hasta ahora. En este modelo tenemos servidores y clientes[1] que se conectan a dichos servidores para acceder a contenido o servicios. Esto está favoreciendo que cada vez más partes de Internet sean controladas por grandes empresas o incluso gobiernos[2], y que éstos puedan crear leyes que ponen en peligro la naturaleza neutral de Internet y beneficiarse de ello.

En el año 2008, Satoshi Nakamoto publica un artículo en el que propone una tecnología pensada para soportar el intercambio de dinero digital de forma segura a través de internet. Este dinero digital, llamado Bitcoin[3], pasa totalmente desapercibido hasta que en el año 2016, siete años después, se empieza a valorar la tecnología subyacente, el Blockchain.

A medida que iban saliendo artículos y que cada vez más gente se diera cuenta del potencial del Blockchain, el valor del Bitcoin empezó a subir desde menos de 500$ en 2016 hasta casi 20000$ un año después, lo que supone una subida de un 3000 % de su valor[4]. Pero esto no es lo importante, lo realmente importante es que esta subida a su vez atrae a más gente a empezar a buscar su propia versión de esta nueva tecnología, buscando nuevas aplicaciones que puedan explotar todo su potencial.

Desde entonces, han surgido las llamadas aplicaciones descentralizadas (*Decentralized Applications* (DApps))[5], que son aplicaciones que se ejecutan no en un servidor, sino en todos los nodos que formen parte de dicha red basada en Blockchain. Estas aplicaciones cada vez están ganando más popularidad, sobre todo por los que están a favor de una Internet descentralizada en la que vaya desapareciendo el concepto de servidor. En un año, se han lanzado a la red pública más de 1000 aplicaciones de este tipo[6], y de momento sus aplicaciones reales están siendo limitadas. Cualquier aplicación tradicional puede ser implementada como aplicación descentralizada, aunque no en todos los casos tenga sentido práctico.

Una de las aplicaciones más nombrada cuando se habla de Blockchain es la del Internet de las Cosas (*Internet of Things* (IoT))[7], ya que topológica y conceptualmente se encuentran varias similitudes. El Internet de las Cosas es un concepto muy amplio, en el que básicamente, todas las cosas están conectadas y son accesibles a través de la red Internet. Cada vez hay más implementaciones reales de este concepto. Una de ellas son las ciudades inteligentes (*Smart Cities*)[8].

Las implementaciones del concepto de Internet de las Cosas suelen tratar de forzar a que sigan la filosofía tradicional de Internet comentada previamente, haciendo que el despliegue de estas redes sea lento. Esto se puede comprobar mirando las estimaciones del Internet de las Cosas y los datos reales que, actualmente, no acaban de despegar como se esperaba[9].

Si en vez de intentar adaptar el Internet de las Cosas para que se pueda implementar de la forma tradicional en la Internet actual, se empieza a adaptar la red para integrar el concepto de Internet de las Cosas, seguramente sería un despliegue más rápido y que tendría más sentido que el que se está llevando a cabo actualmente. Blockchain es uno de esos cambios que puede ayudar a adaptar la red de hace 30 años a las nuevas necesidades que demandan los nuevos servicios.

1.1. Motivación

Las implementaciones de la Internet de las Cosas sobre la red de Internet actual sin modificar sus conceptos ni filosofía está complicando su adaptación global.

Un ejemplo más concreto en el que centrarse son las ciudades inteligentes, dónde los nodos están totalmente distribuidos y descentralizados, generando datos y accediendo a ellos continuamente. Como tratar estos datos de forma descentralizada es difícil usando una Internet basada en servidores y clientes. Actualmente, se están adaptando estas redes de forma que los datos generados en cada nodo sigan una jerarquía de tipos de nodos que vayan centralizando los datos en servidores donde se puedan almacenar, ordenar y visualizar de forma más sencilla con la infraestructura ya existente.

En vez de ver la descentralización de estos datos como un problema que se soluciona centralizándolos, se puede usar como una ventaja y tratar de usar el conjunto de todos esos nodos desplegados a lo largo de la ciudad para que ellos mismos soporten el envío, almacenaje y acceso de esos datos que generan. De esta forma se elimina la necesidad de tener servidores especializados para realizar estas tareas y se mantiene la filosofía del internet de las cosas.

Blockchain puede ser la clave para poder interconectar las ciudades inteligentes con Internet, ya que añade un nivel de abstracción descentralizado. Además, con la ayuda del incremento de capacidad de almacenamiento y cómputo que están experimentando este tipo de nodos, este planteamiento es todavía más factible a corto plazo.

1.2. Objetivos

El objetivo de este trabajo es el uso de Blockchain para dar soporte a los despliegues de ciudades inteligentes, de forma que almacenen los datos generados de forma distribuida dentro de su propia red, así como se gestione el acceso a esos datos ya sea desde la propia red como desde el exterior de la misma. De esta forma, se elimina la necesidad de mantener servidores que se encarguen de esas tareas explotando el carácter descentralizado de estas redes.

Para llevar a cabo este objetivo, se realizará un estudio en profundidad de la tecnología Blockchain, así como de las variantes que han ido surgiendo para ver cómo adaptarla a las necesidades de este trabajo en particular.

Se creará también una red de prueba sobre la que se irán desarrollando soluciones de código abierto basadas en aplicaciones distribuidas donde se implementarán las funcionalidades descritas anteriormente.

1.3. Organización del documento

La redacción de este trabajo se ha dividido en cinco capítulos, los cuales se muestran a continuación:

- **Capítulo 1:** Introducción.

  En este capítulo se describe el panorama actual en el que se sitúa este trabajo, del que se extraen las motivaciones que dan lugar al mismo y los objetivos perseguidos.

- **Capítulo 2:** Estado del arte.

  Presenta una visión de los principales conceptos teóricos sobre los que se apoya este trabajo, entre los que se resaltan los conceptos de Blockchain y las diferentes alternativas que se presentan.

- **Capítulo 3:** Diseño.

  Este capítulo introduce en detalle el problema que se nos plantea y propone el diseño de una solución que se adapte a los requisitos identificados.

- **Capítulo 4:** Implementación.

  En este capítulo se explican y justifican las diferentes decisiones tomadas para la implementación del diseño planteado y se detalla el proceso para el desarrollo de cada elemento que conforma el sistema.

- **Capítulo 5:** Conclusiones y líneas futuras.

  Finalmente se hace balance del trabajo y se reflexiona acerca de las posibles mejoras a realizar a corto y medio plazo.

2. Estado del arte

En los últimos años se está hablando mucho sobre las criptomonedas y, sobre todo del Bitcoin. Guste o no, el hecho de que se hable de esto es porque hay gente que gana dinero con ello, básicamente especulando con el valor de estas monedas digitales en el mercado, lo que atrae mucho público de ámbitos muy diversos. Pero, ¿por qué hay gente ganando dinero?, ¿qué tienen de especial estas monedas?, ¿fueron pensadas para lo que se están usando realmente?

En el año 2008 Satoshi Nakamoto publica el artículo "*Bitcoin: A Peer-to-Peer Electronic Cash System*"[3] en el que propone una tecnología que promete resolver el problema del doble gasto en el intercambio de dinero digital sobre una red pública. Esta tecnología es el Blockchain y la red que soporta, Bitcoin.

Desde ese momento, y gracias al impulso mediático, que supuso el crecimiento de su valor, todo el mundo tecnológico y financiero se fijó en el Bitcoin y, por tanto, en la tecnología que hay por debajo. Blockchain es la tecnología detrás de Bitcoin y de la mayor parte de las criptomonedas actuales y es lo que las hace especiales y diferentes a cualquier cosa anterior a ellas.

La primera mención a algo parecido a la tecnología Blockchain fue realizada por un grupo de investigadores llamados "Cypherpunks"[10] en 1992. Este grupo se reunía de vez en cuando y discutía sobre temas relacionados con la criptografía, la privacidad y de cómo usarlo en entornos prácticos del día a día. Creó varios sistemas como Hashcash[11] o B-Money[12], y otros muchos aportes tratando de solventar todos los problemas con los que se encuentra el envío de dinero digital a través de la red.

Este grupo ha influido claramente en la creación de Bitcoin, ya que hace referencias a muchos de los problemas que solía tratar este grupo en sus quedadas, como puede ser el problema del gasto doble en el que se profundizará posteriormente.

En el artículo de Bitcoin, se describe una versión de un protocolo *peer-to-peer*[13] de intercambio de monedas o dinero digital. Al mismo tiempo creó un software llamado Bitcoin con el que se empezó a crear una red que se quedó con el mismo nombre y en la que se intercambian Bitcoins. Esta red empezó a funcionar el 3 de enero de 2009.

El valor del Bitcoin enseguida empezó a crecer y otras criptomonedas empezaron a aparecer en el sector, usando simplemente la misma idea original y cambiando algunos parámetros, o añadiendo nuevas ideas y variaciones al propio protocolo. Otros incluso proponiendo otras tecnologías que no se pueden considerar Blockchain aunque compartan la misma filosofía.

Esto ha sido lo mejor y lo peor que le podía pasar al Blockchain, ya que le sirvió para ser noticia en todas partes y que se diera a conocer, y muchos empezasen a interesarse en la tecnología y como aplicarla a otros campos.

Cuando se habla de redes *peer-to-peer*, se hace referencia a redes en los que sus nodos son tratados de igual forma o no hay una jerarquía marcada en la red. El límite de esta descentralización está en las redes distribuidas. Para recordar de forma gráfica y rápida estas diferencias, se añade

la Figura 1. Son conceptos similares y está bien tenerlos bien definidos, ya que van a ser usados continuamente a lo largo del documento.

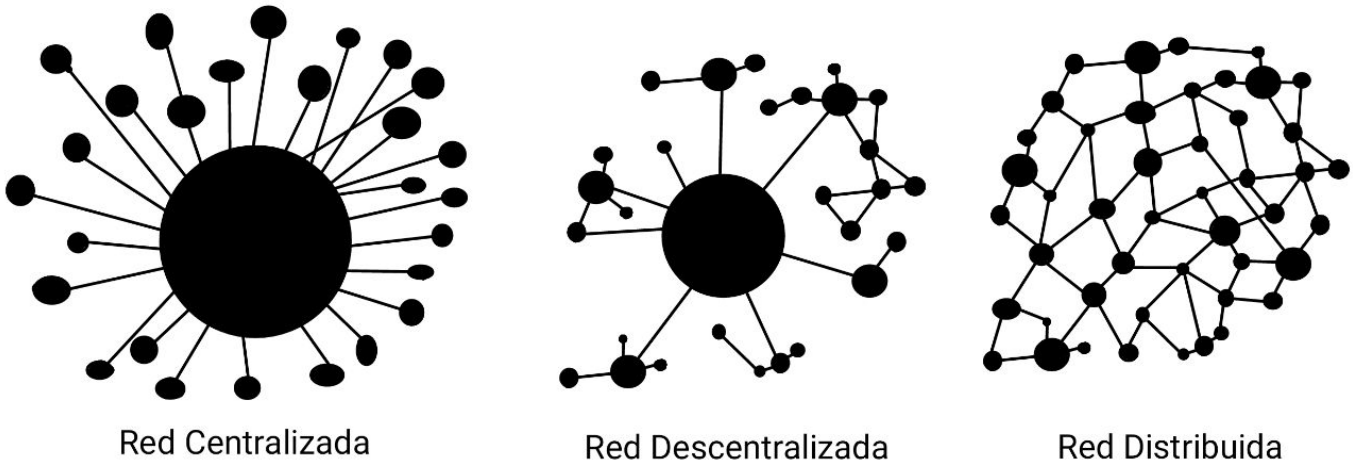

Figura 1: Tipos de redes

Para tener una idea global de lo que se entrará a explicar en las siguientes secciones, se puede ver globalmente el funcionamiento del Blockchain, explicado con lo que ya se sabe, es el de una red basada en *peer-to-peer*, donde todos los nodos se encuentran en el mismo nivel jerárquico, pero en vez de tener partes de información distribuida por la red, tienen toda la información todos los nodos. Esta información son todas las transacciones de datos, organizados en bloques, que se han generado en la red desde la primera. De esta manera, si solamente un nodo queda activo en un momento determinado, toda la información puede ser recuperada. De igual forma, si hay alguna información que no cuadra, se verifica con el resto. Está es una de las principales características que tiene el usar esta tecnología.

Pero esto es simplemente para tener una idea global de lo que se va ha hablar a continuación. La tecnología es mucho más compleja y, por ello, se va a empezar planteando la problemática que aborda para entender mejor por qué surge la idea de usar una forma nueva de hacer las cosas.

2.1. Problemática que aborda Blockchain

Los problemas a los que se enfrenta Blockchain no son nuevos y se han intentado resolver con anterioridad. A continuación se exponen dichos problemas para explicar, posteriormente, la forma en la que Blockchain los resuelve.

2.1.1. Problema del gasto doble

En la Figura 2, se muestra gráficamente el problema del gasto doble (*double-spending problem*)[14]. Alice tiene 10$ y le debe 10$ a Bob, y le hace una transacción a Bob y a la vez otra de

la misma cantidad a Charlie. El problema es que ahora hay 20$ en la red porque no hay nada que impida a Alice duplicar su dinero digital.

Para que quede más clara esta problemática en concreto, se plantea el siguiente escenario[15]:

"Imagina que estás sentado en un banco con un amigo que tiene una manzana en la mano. Tu quieres esa manzana y se la pides, el acepta y te dice que si y te la da, la pasa de su mano a la tuya. La manzana a pasado de su mano a la tuya, y tu lo has visto, antes la tenía él y ahora la tienes tu. Ahí no tienes ambigüedad en que la transacción se haya realizado correctamente.

Cuando se intenta pasar este ejemplo al mundo digital, esto se complica un poco. Ahora tu amigo tiene una manzana digital "manzana.ap" y te la va a dar a ti. Ya no es lo mismo que antes, ¿cómo sabes que esa manzana te la ha pasado sólo a ti? puede haber hecho antes una copia perfectamente igual y haberla mandado a dos personas al mismo tiempo, entre otras cosas. No tienes forma de saber que la "manzana.ap" que antes era suya ahora es tuya y sólo tuya."

El intercambio de bienes virtuales no es tan sencillo como el de bienes físicos. Este es un problema asociado al riesgo de gastar dos o más veces la misma cantidad de dinero digital, ya que una moneda virtual no deja de ser un archivo que puede ser duplicado o falsificado (siendo esto más o menos complejo de realizar).

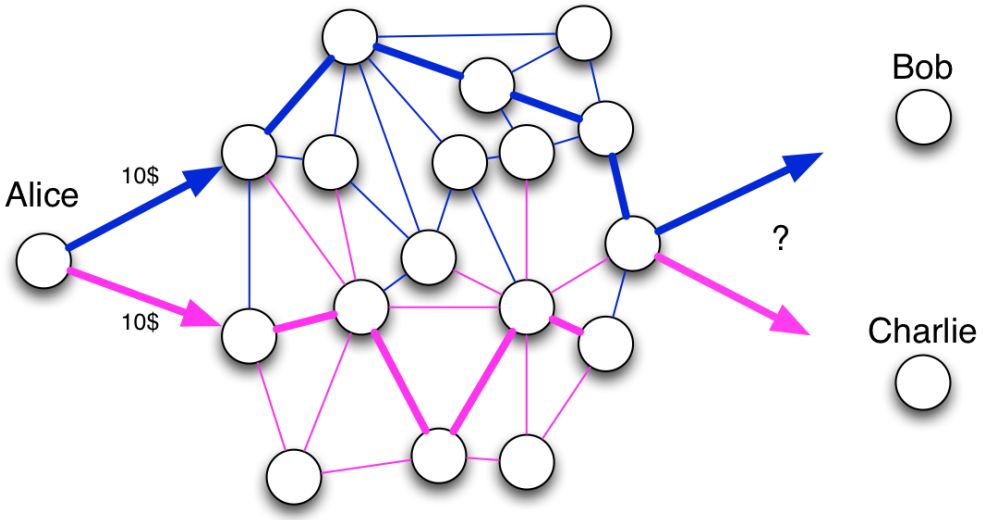

Figura 2: Problema del gasto doble

2.1.2. Tercera parte de confianza

Normalmente para solucionar el problema descrito con anterioridad se suele recurrir a una tercera parte en la que ambos confían y que se encarga de verificar y hacer esa transacción legítima. Esta manera de abordar el problema es como trasladar la forma de verificar transacciones físicas al mundo digital. En una transacción física, se usa la figura del notario en el que ambas partes confían.

Este notario básicamente mira el historial del bien que se vaya a transferir, confirma que todo es correcto y confirma que la transacción es válida.

De esta forma de verificar transacciones se puede ver que el elemento más importante es ese historial de bienes en donde se encuentra toda la información, ya que si éste tiene información incorrecta, la transacción no se podría hacer, o si alguien consigue acceso puede conseguir hacer la transacción dos veces, por ejemplo.

Si se lleva esta idea al mundo digital enseguida se ve que estas vulnerabilidades son más obvias, ya que este historial de transacciones debería de estar en un sitio muy seguro, y eso en red siempre es una complicación más. Si se consiguiera acceso a este historial, se podrían hacer válidas transacciones que en realidad no lo son, incluso copias o pérdidas de información clave.

Otro ejemplo de esta problemática se produce cuando se realiza una transacción en la que están involucradas varias identidades, ya que cada una tiene la información almacenada de forma distinta y en distintos formatos. Esto hace que sea costoso y lento, sobre todo en transacciones internacionales.

2.1.3. Información distribuida

Si este historial de transacciones se distribuyese entre todos los nodos, creando un libro de contabilidad distribuido, como se muestra en la figura 3, a través de la red, y estos nodos tuviesen una copia completa, entonces no tendríamos la necesidad de esa tercera parte de confianza. El sistema en este caso es mucho más robusto, ya que para engañarlo, se tendría que engañar a todo el sistema al mismo tiempo.

Ya no se habla de una persona ni de una entidad que verifica todo, sino de un sistema que todo el mundo puede ver y contribuir a mejorar. De hecho, hacen falta nodos que se dediquen a actualizar el historial de transacciones, a los cuáles se les recompensaría con una especie de comisiones por las molestias, comúnmente llamado “Minería de criptomonedas”.

Además, las comisiones mencionadas anteriormente es la forma en la que se van generando nuevas monedas virtuales. Algunas tecnologías tienen un límite y otras no. Esto se especifica desde el momento en que se genera la descripción del protocolo particular.

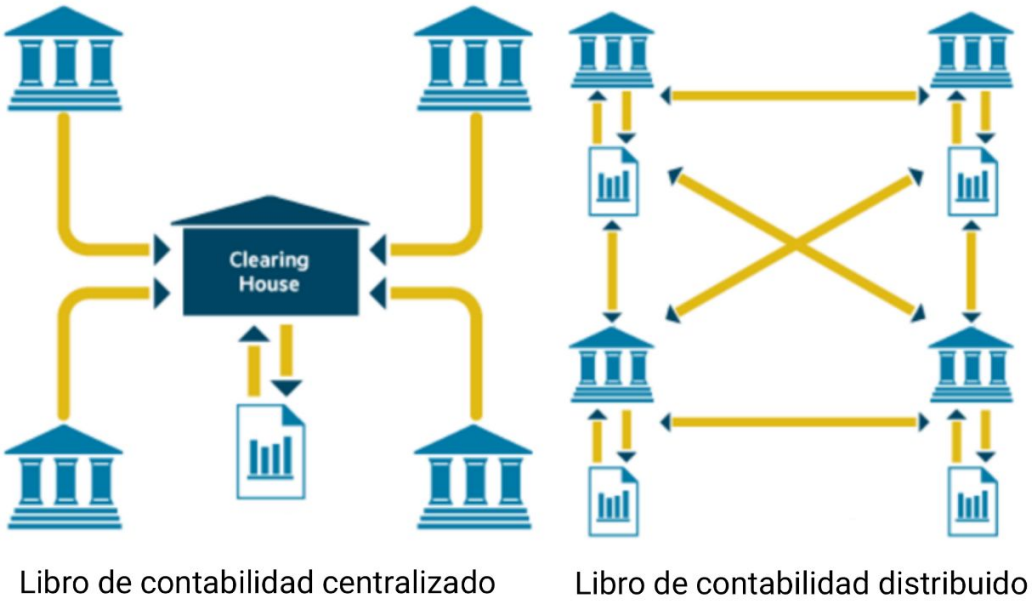

Figura 3: Información distribuida por todos los nodos

2.2. Bitcoin

Bitcoin es la primera red que consigue poner estas ideas en algo real. Surge como un método de intercambio de dinero basado en prueba criptográfica, en lugar de basado en un tercero de confianza, como ya se ha explicado. Esto permite que solamente las dos partes interesadas en el intercambio sean las necesarias para llevarlo a cabo, apoyadas por la red.

La red de Bitcoin es un sistema distribuido *peer-to-peer* que resuelve el problema del gasto doble usando un servidor de sellado temporal, generando una prueba computacional del orden cronológico de las transacciones. Este sistema es seguro siempre que el conjunto de lo que se denominan "nodos honestos" tengan más potencia de cómputo que el mayor conjunto de "nodos atacantes".

2.2.1. Transacciones

Una transacción se produce cuando un individuo, el propietario, le envía una cantidad de monedas electrónicas a otro, el beneficiario. Cada una de estas transacciones conllevan una serie de operaciones. En concreto, cada transacción contiene la clave pública del beneficiario, un hash[16] y la firma del propietario. De esta manera, una moneda electrónica se define en este contexto como una cadena de firmas digitales.

Por tanto, cada vez que se realiza una transacción se genera un hash con la clave pública del beneficiario y la transacción previa, después con este hash y la clave privada del propietario se firma la transacción en curso, por último se incluye la clave pública del beneficiario y con todo esto ya se tendría una transacción. Con todas estas operaciones se puede verificar cada transacción de

forma sencilla. Para seguir de forma gráfica el proceso, se incluye la figura 4.

Esta forma de trabajar no resuelve el problema antes mencionado, y no hay forma de verificar que uno de los propietarios no haya gastado dos veces la misma moneda. De esto se encarga la autoridad central de confianza, o casa de la moneda, que va comprobando cada una de esas transacciones para que eso no sea un problema. Con esta forma de trabajar, cada moneda debe regresar a la casa de la moneda para que esta distribuya una nueva, ya que solamente estas monedas son las que estarían libres de cualquier sospecha del doble gasto. El problema de esta solución centralizada es el que se ha comentado varias veces, todo depende de este sistema centralizado, como en un banco.

Para solucionar esto, Bitcoin propone que las transacciones sean anunciadas públicamente. De esta forma se puede tener un sistema en el que cualquiera pueda tener un historial de todas las transacciones y, al ser muchos, que el historial real sea aquel en el que la mayoría de los participantes estén de acuerdo. Así, el beneficiario tendría la certeza de que los propietarios previos no han firmado transacciones anteriores a la suya. En este sistema, la última transacción es la que cuenta y no se fija en el intento o no de dobles gastos posteriores.

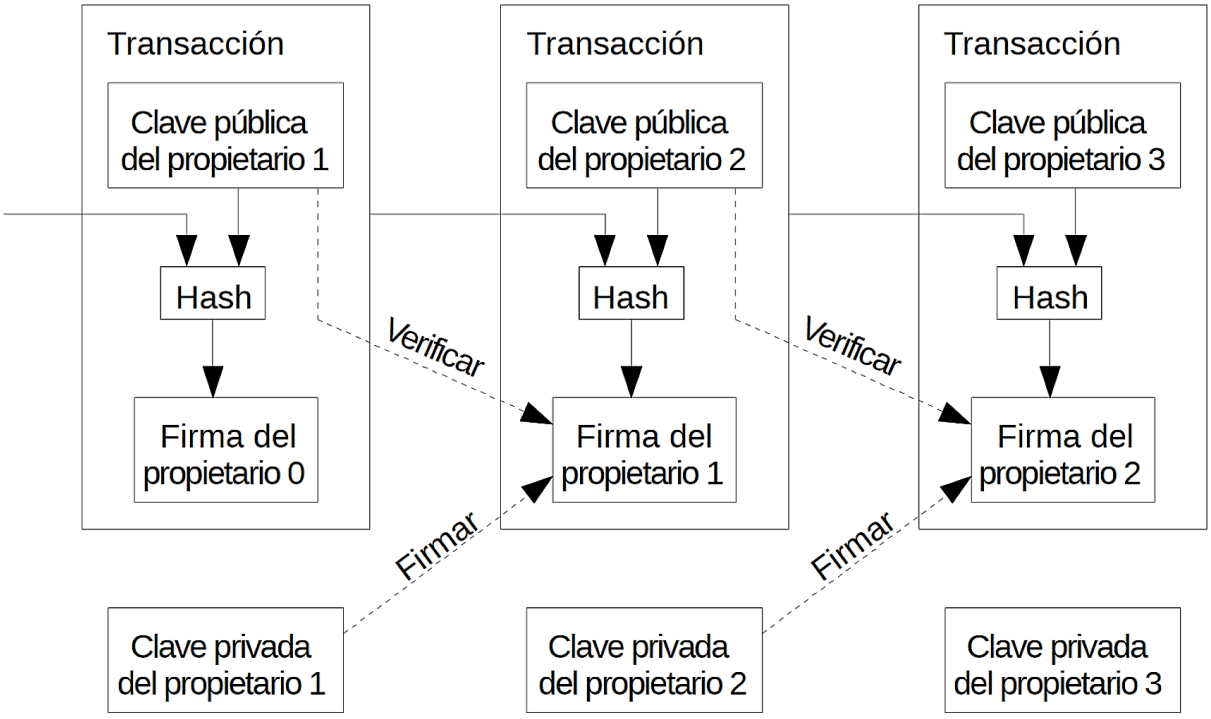

Figura 4: Funcionamiento de las transacciones en el Blockchain de Bitcoin

Como se puede ver en la figura 5 Una transacción tiene múltiples entradas y una o dos salidas para permitir que el valor que se va a transferir se combine o se divida en una o varias transacciones. Sería posible usar un sistema que se basara en monedas individuales, pero tratar cada céntimo que se mande como una transacción individual se convertiría en intratable. Pueden darse dos casos:

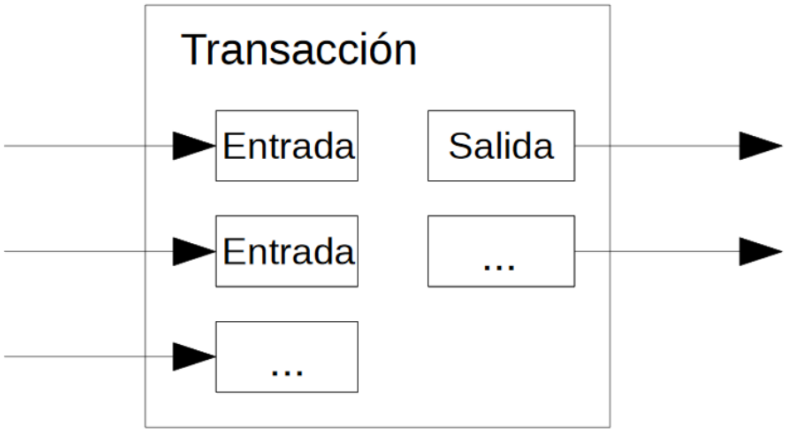

Figura 5: Estructura de las transacciones en Bitcoin

- El primero es que haya varios envíos de pequeña cantidad que se sumen como varias entradas de una transacción. En este caso, de varias transacciones, se consigue hacer una sola.
- El segundo es que haya un envío muy grande de dinero que sea dividido en varias transacciones, en cuyo caso la transacción tendrá una sola entrada de una transacción anterior mayor.

Las salidas siempre son dos como máximo: una fija para el pago y otra opcional para devolver el cambio al emisor. Cabe señalar que la diseminación de control, donde una transacción depende de varias transacciones, y esas transacciones dependen de muchas más, no supone aquí un problema. No existe la necesidad de extraer una copia completa e independiente del historial de una transacción.

2.2.2. Servidor de Sellado Temporal

La clave para solucionar el problema del gasto doble siempre ha estado en el tiempo. Por este motivo, se necesita un servidor de sellado temporal, que lo que hace es básicamente coger el hash de un bloque de transacciones y sellarlos temporalmente. De esta forma se añade este sellado temporal, el suyo y el del bloque anterior, a cada bloque y se notifica públicamente, de esta manera, con cada sellado se refuerza el sellado anterior, formando una cadena de bloques sellados temporalmente, de forma similar a la representada en la figura 6. Es como decir que este bloque de ítems con este hash existe en este tiempo, y el siguiente bloque tendrá el sellado temporal previo y el suyo y así uno tras otro.

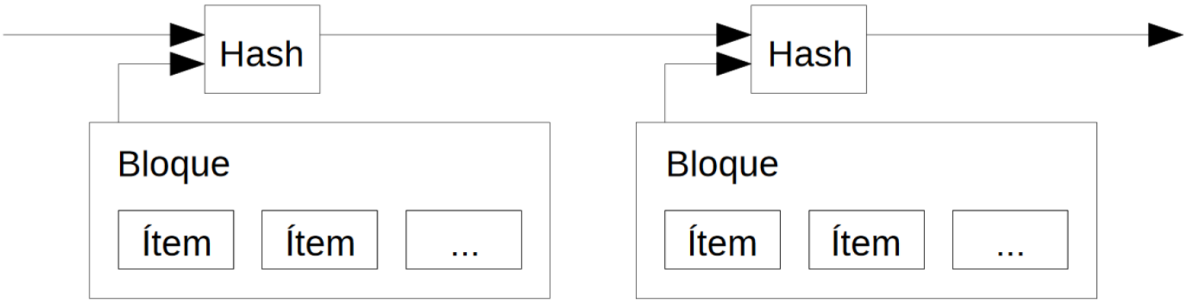

Figura 6: Funcionamiento del sellado temporal en Bitcoin

2.2.3. *Proof-of-Work*

En la sección anterior se ha hablado de la funcionalidad que se necesita de sellado temporal, pero como se comenta al principio, lo que se quiere conseguir es una red distribuida *peer-to-peer* y sin el uso de nada centralizado. Esto se consigue utilizando un concepto que proviene de un artículo de Cynthia Dwork y Moni Naor publicado en 1993, llamado en este caso *proof-of-work*. Este sistema es comúnmente usado como una contramedida a ataques de denegación de servicio o spam, utilizado en el sistema Hashcash.

Proof-of-work se usa en Bitcon para añadir una dificultad artificial al historial de transacciones y para la generación de bloques nuevos, para que cualquiera no pueda generar los bloques que quiera sin que le cueste nada hacerlo. Bitcoin es una red, y está dentro de la red Internet, por lo que se supone que es visible por todo el mundo con acceso a dicha red. En este contexto es normal pensar que va a haber nodos buenos y malos. Si se supiera distinguir unos de otros no haría falta usar nada de esto, ya que el problema estaría resuelto, pero en realidad no se puede saber a priori si un nodo va a ser malo. Es por esto que se necesita un método que dificulte el mecanismo necesario para añadir entradas al historial de transacciones, o lo que es lo mismo, para añadir bloques a la cadena.

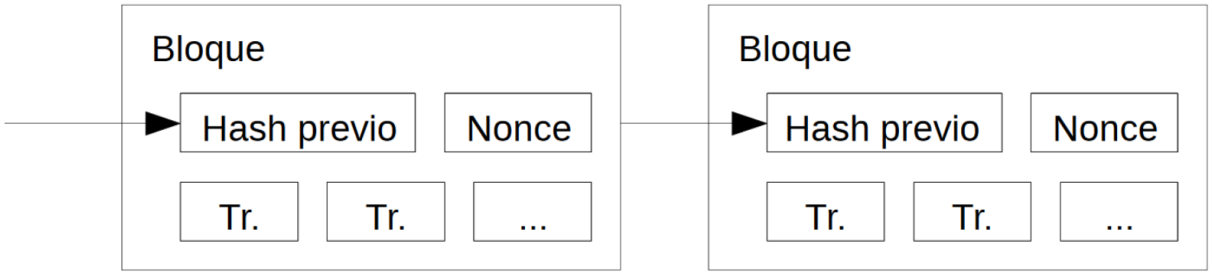

Figura 7: Generación de bloques usando *proof-of-work*

Lo que comúnmente se llama minado, es el proceso de generar bloques mediante este *proof-of-work* o prueba de trabajo. En Bitcoin todos los mineros compiten para encontrar un número tal que, cuando se haga un hash con el bloque, cumpla ciertas condiciones. Ese hash es almacenado en el siguiente bloque de la cadena y es la referencia que existe entre los bloques y es como se va

conformando la cadena, siguiendo la figura 7. SHA-256[17] (*Secure Hash Algorithm* (SHA)) es la función hash que se usa en Bitcoin y tiene una dificultad variable en función de las condiciones que se impongan. Se usa la función hash porque tiene una propiedad que aprovecha el Blockchain: es muy difícil encontrar el número para el cuál el hash cumpla ciertas condiciones, pero una vez encontrado es muy fácil verificar su validez por otros.

Las condiciones que se imponen son básicamente que los primeros x valores del hash sean "0", y esto se consigue incrementando un número llamado *nonce* hasta que se encuentre un valor que dé al hash del bloque los cero bits requeridos. De esta forma se puede variar la dificultad de encontrar dicho valor, que es exponencial al número de cero bits requeridos y se suele calcular tal que el tiempo de generación de bloques sea constante, usando una media móvil que apunta a un número medio de bloques por hora. Si se generan bloques demasiado rápido, se aumenta la dificultad (número de bits = 0 requeridos).

Se propone un ejemplo de uso para ver el funcionamiento de esta idea. Suponer que se tiene el *string* "Hello, world!" substituyendo al bloque en el caso de blockchain. El objetivo es encontrar una variación de su SHA-256 tal que sus tres primeros dígitos sean "0"("000XXXXXX...XXX"). Se comienza a buscarlo añadiendo al final del string original el *nonce* y haciendo la función de hash cada vez iterando este *nonce* hasta encontrar el objetivo. Encontrar esta solución en concreto lleva esta vez 4251 iteraciones.

"Hello, world!0": 1312af178c253f84028d480a6adc1e25e81caa44c749ec81976192e2ec934c64
"Hello, world!1": e9afc424b79e4f6ab42d99c81156d3a17228d6e1eef4139be78e948a9332a7d8
"Hello, world!2": ae37343a357a8297591625e7134cbea22f5928be8ca2a32aa475cf05fd4266b7
...
"Hello, world!4248": 6e110d98b388e77e9c6f042ac6b497cec46660deef75a55ebc7cfdf65cc0b965
"Hello, world!4249": c004190b822f1669cac8dc37e761cb73652e7832fb814565702245cf26ebb9e6
"Hello, world!4250": 0000c3af42fc31103f1fdc0151fa747ff87349a4714df7cc52ea464e12dcd4e9

Usar esta idea basada en *proof-of-work* tiene muchas ventajas a nivel de seguridad y robustez. A medida que se van generando bloques de esta manera, van quedando encadenados unos a otros y para cambiar un bloque se necesitaría cambiar todos los bloques anteriores. Por así decirlo, cuánto más adentro se encuentre un bloque en la cadena, más potencia de cómputo sera necesario para cambiarlo. Hay ocasiones que se producen ramificaciones o dos cadenas a la vez. En estos casos, se determina que la cadena válida es en la que más cómputo se ha dedicado, que será la decisión de la mayoría de los nodos de la red.

Haciendo uso de esta forma de trabajar, la mayoría está controlada por el grupo que más potencia de cómputo tenga. De esta forma, si esta mayoría es honesta o controlada por los nodos anteriormente denominados buenos, la cadena honesta crece mucho más rápido que cualquier otra y no puede ser alcanzada por cualquier conjunto de potencia de cómputo menor o igual a ella. De hecho, la probabilidad de que un atacante más lento alcance a los honestos disminuye exponencialmente a medida que se añaden bloques posteriores.

Para tratar de poner unos números sencillos y se pueda imaginar esta dificultado, se propone el siguiente escenario práctico:

En la red Blockchain, todos los nodos mineros están trabajando en el bloque número 91. En un momento dado, uno de ellos quiere cambiar una transacción suya (porque teóricamente sólo puede cambiar una que haya hecho él mismo) que se encuentra en el bloque número 74. Para poder hacer efectivo este cambio, tendría que realizar la computación equivalente a minar 18 bloques completos. Y no sólo eso, tendría que hacerlo antes de que el resto de la red acabase de minar el bloque en el que está trabajando.

2.2.4. Seguridad y privacidad

Ya que se ha empezado a hablar de seguridad a raíz de ver las ventajas que tiene el uso de *proof-of-work*, se va a profundizar un poco más es este aspecto y, también en el de la privacidad que la red de Bitcoin ofrece.

Como se ha comentado brevemente en el apartado anterior el ataque que puede sufrir la Blockchain Bitcoin es que un nodo o conjunto de nodos intenten crear una cadena alternativa más rápidamente que la cadena buena. Si esto se consiguiera, el atacante seguiría sin poder crear dinero de la nada ni adueñarse del dinero de otro, ya que los nodos no aceptarían una transacción inválida ni cualquier bloque que la contenga. Por tanto, un ataque con éxito solamente podría cambiar transacciones de su propiedad, y por tanto, recuperar exclusivamente dinero que hubiese gastado recientemente.

De todas formas, realizar un ataque con éxito directamente a la red de Blockchain es muy difícil. En el artículo original de Bitcoin, se ve como una carrera entre la cadena honesta y la cadena atacante, y se modela como un paseo aleatorio binomial[19]. En dicho artículo se entra a formular matemáticamente, incluso se propone un ejemplo en C, las conclusiones de dicho estudio se van a nombrar a continuación, pero no es el objetivo repetirlo aquí.

La conclusión más importante a la que se llega es que a no ser de que el atacante tenga suerte el principio de la creación de la cadena y consigue generar el primer bloque antes que un nodo honesto, cada vez lo tendría más difícil, ya que a medida que los nodos honestos van generando bloques, la probabilidad de que el atacante alcance a la cadena va decayendo de forma exponencial.

Respecto a la privacidad, siguiendo el modelo clásico basado en la tercera parte de confianza es relativamente fácil de conseguir dado que las transacciones se quedan entre los implicados en la transacción y la tercera parte de confianza, consiguiendo privacidad cara al resto del mundo. En el caso de usar Blockchain, esto no es tan sencillo dado que todas las transacciones se hacen públicas para cualquiera con acceso a la red.

A priori, puede parecer complejo proporcionar privacidad a una red con estas características, pero es bastante sencillo. La clave es romper el flujo de información en otro punto. Básicamente se trata de que las claves públicas que se usan para realizar las transacciones no estén ligadas a la identidad de quién las realice, como se intenta representar en la figura 8. La red puede ver que alguien está enviando una cantidad a otro alguien, pero sin que haya información vinculando la transacción con nadie. Este modelo de conseguir privacidad es similar al usado en bolsa en la actualidad.

Como cortafuegos adicional, debería usarse un nuevo par de claves en cada transacción para evitar que se relacionen con un propietario común. Con las transacciones multi-entrada será inevitable algún tipo de vinculación, pues revelan necesariamente que sus entradas pertenecieron al mismo propietario. El riesgo es que si se revela la identidad del propietario de una clave, la vinculación podría revelar otras transacciones que pertenecieron al mismo propietario.

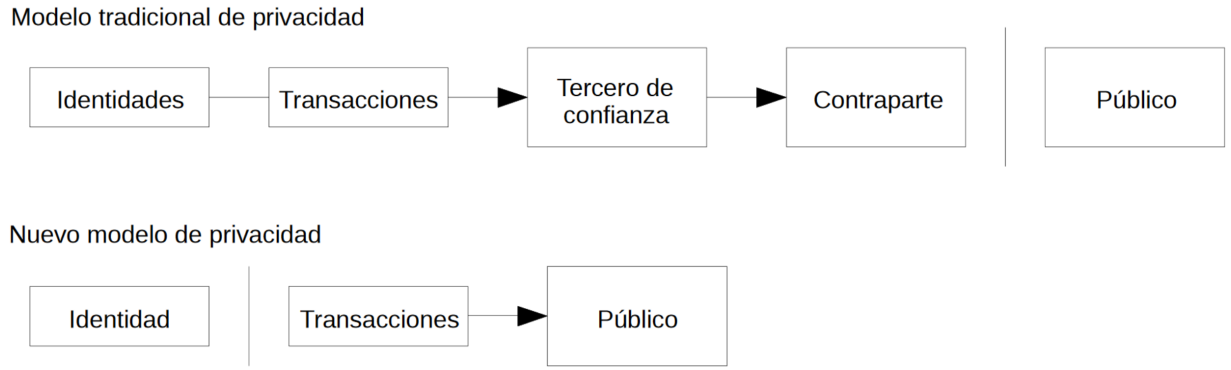

Figura 8: Conseguir privacidad en Blockchain

2.2.5. Visión global de la red

Ahora que ya se han presentado los fundamentos teóricos clave de una red basada en la tecnología Blockchain, se va a ver una visión global del sistema y cómo funcionan en conjunto todas las partes y conceptos explicados en profundidad previamente.

Se ha incidido durante todo el documento en las transacciones, y como se ha visto es el elemento sobre el que se empieza a construir la cadena de bloques. Cuando se van realizando las transacciones, se hacen públicas transmitiéndose a todos los nodos de la red. Estos nodos recogen estas transacciones y van creando bloques, esto lo hace cada nodo con su bloque. Cuando tienen un bloque, comienza cada uno a resolver su propia *proof-of-work*. En el momento en el que un nodo resuelve su *proof-of-work*, transmite su bloque al resto de nodos. Estos nodos comprueban la validez de las transacciones que contiene este bloque y si no han sido gastadas con anterioridad, y aceptan dicho bloque en el momento en el que se ponen a trabajar en el siguiente bloque y usan el hash del bloque aceptado como hash previo para generar el siguiente bloque. Si dos bloque se transmiten simultáneamente desde dos nodos, nunca van a recibirse también al mismo tiempo en todo el resto de nodos y éstos siempre trabajan con la versión del siguiente bloque que les llega antes, aunque también guardan la versión que llega tarde por si se convierte al final en la cadena más larga. Los nodos consideran correcta la cadena más larga siempre, y es la que trabajan. Una vez que la siguiente *proof-of-work* es encontrada, ya no habrá ninguna duda en cuál va a ser la ramificación más larga, así que los nodos que estaban trabajando en la ramificación más corta cambiarán automáticamente y se sincronizarán con la más larga, como se espera que hagan.

Todas las transacciones no tienen por qué llegar a todos los nodos, si alcanza a la mayoría de ellos no tardaría mucho tiempo en ser incluida en un bloque. La red también está preparada para perder temporalmente un bloque, ya que el nodo se dará cuenta de que le falta uno si recibe el bloque

siguiente, y lo comunicará a la red para que le den el anterior.

2.2.6. Incentivo

Los incentivos en una red basada en Blockchain se usan por tres razones principalmente y se resumen a continuación:

- La primera es necesaria dado que no hay entidad central que inyecte nuevas monedas a la red, para ir generando y poniendo en circulación monedas nuevas de manera continua y progresiva. La primera transacción de un bloque genera una moneda nueva, que será propiedad del creador de dicho bloque. En el artículo original se hace una analogía con los mineros de oro, que consumen recursos para añadir oro a la circulación. Y en este caso se estaría consumiendo tiempo de CPU[20](*Central processing Unit* (CPU)) y electricidad para añadir monedas a la circulación.
- Relacionado con el punto anterior, la siguiente razón es porque estos incentivos sirven para que los nodos quieran unirse a la red y generar bloques para la cadena y continuar haciéndolo en el tiempo, ya que si son ellos los que generan los bloques, son ellos los que se quedan con las monedas nuevas. Las comisiones de las transacciones también pueden considerarse como un incentivo extra. La comisión por transacción es la diferencia entre el valor de salida de ésta y el valor de entrada, siempre que el valor de salida sea menor que el de entrada. Si en un momento de tiempo el sistema evoluciona a un estado en el que no se necesiten generar monedas, se puede evolucionar hacia comisiones de transacción y estar completamente libre de inflación.
- La última razón es que estos incentivos mantienen el interés de estos nodos por mantenerse honestos y no formar una mayoría con intenciones de atacar la red, ya que les supondría menos beneficio que colaborar a que siga creciendo y consiguiendo dichos incentivos ellos mismos. Debe de ser más rentable siempre respetar las reglas y contribuir que atacar la cadena y jugar con la validez de su propia riqueza.

Como ya se ha explicado la forma en la que se distribuyen los incentivos en la red y cómo se generan los bloques, es lógico entrar un poco más en detalle de cómo se contribuye realmente en una *proof-of-work* y cómo evolucionan las formas de hacerlo adaptándose a los cambios en dificultad y volumen de nodos que va cambiando de forma dinámica en este tipo de redes, esta vez desde el punto de vista de los llamados “mineros” o nodos de la red.

En Bitcoin la recompensa por generar un bloque empezó siendo 50 Bitcoins y esta recompensa va disminuyendo a la mitad cada 210 000 bloques generados, que se calcula que es cada 4 años aproximadamente. A la hora de escribir este documento, la recompensa por bloque es 12.5 Bitcoins que serían unos 150 000 dólares. Por tanto, cuando la recompensa se haya dividido entre dos 64 veces, la recompensa por generar bloques nuevos sera de 0 Bitcoins y, por tanto, no se generarán nuevos Bitcoins. Como ya se ha comentado, en este momento el incentivo sería solamente el de las comisiones de las transacciones.

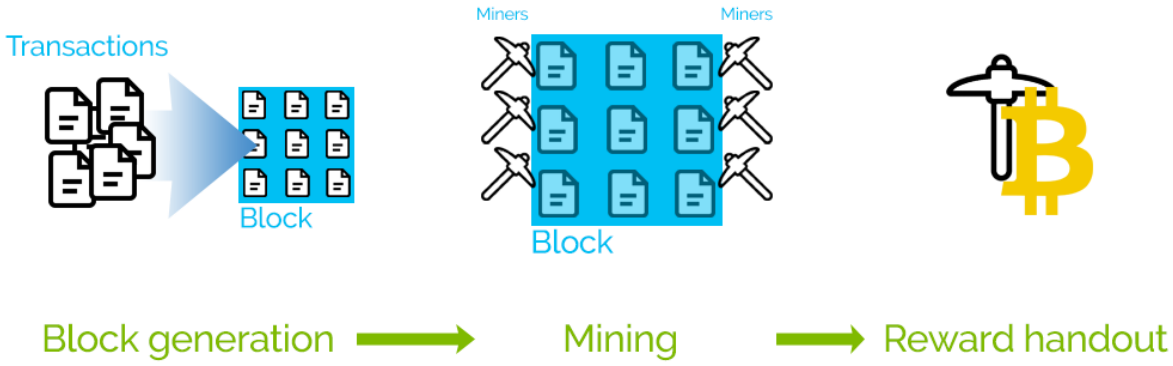

Figura 9: Forma tradicional de contribuir a una red basada en *proof-of-work*

La forma tradicional, mostrada en la figura 9, en la que se podía contribuir siendo un nodo de la red era construyendo un nodo, y conectándolo a la red. Este nodo se descargaría la cadena de bloques y en ese momento estaría sincronizado con la red para empezar a generar y validar bloques en ella. En un principio, al haber pocos nodos el valor computacional necesario de cada uno de ellos para poder generar un bloque antes que los demás era relativamente bajo, debido a que la dificultad para mantener el tiempo medio de generación de bloques era relativamente baja. Empezando con una CPU se podía llegar a generar el siguiente bloque antes que los demás con bastante probabilidad, pero a medida que los nodos se iban añadiendo a la red, el tiempo de generación de bloques iba disminuyendo por lo que la probabilidad de encontrar ese bloque con una CPU[20] ya era extremadamente baja, y así se ha ido cambiando el hardware de CPU pasando por GPU[21](*Graphics Processing Unit* (GPU)) a FPGA[22](*Field-programmable gate array* (FPGA)) y a hardware específico para generar el máximo hashrate (hashes por segundo) consumiendo la menor electricidad posible.

Esto ha ido cambiando mucho y muy rápido. Ahora, contribuir como un nodo a una red como Bitcoin esperando ser el primero en generar un bloque es muy difícil, ya que tienes que tener un porcentaje significativo de la red para tener una probabilidad significativa de conseguir bloques. Esto ha llevado a plantear sistemas de minado basado en grupos o *pool mining* (figura 10). En estos grupos, se colabora en la red en grupo y los bloques generados se distribuyen entre los miembros de este grupo de forma proporcional a la capacidad de cómputo que se esté aportando. De esta forma se asegura un incentivo relativamente pequeño y constante, en vez de esperar a que una probabilidad muy muy pequeña de conseguir un bloque te de una recompensa enorme. También tiene otra ventaja cada vez mayor, elimina la necesidad de tener la cadena de bloques sincronizada en el nodo. Siendo ahora el peso del Blockchain de Bitcoin de unos 160 GB, se puede tardar en sincronizar una semana, y el usar esta forma de contribuir a la red elimina todo esto, basta con tener una conexión a Internet.

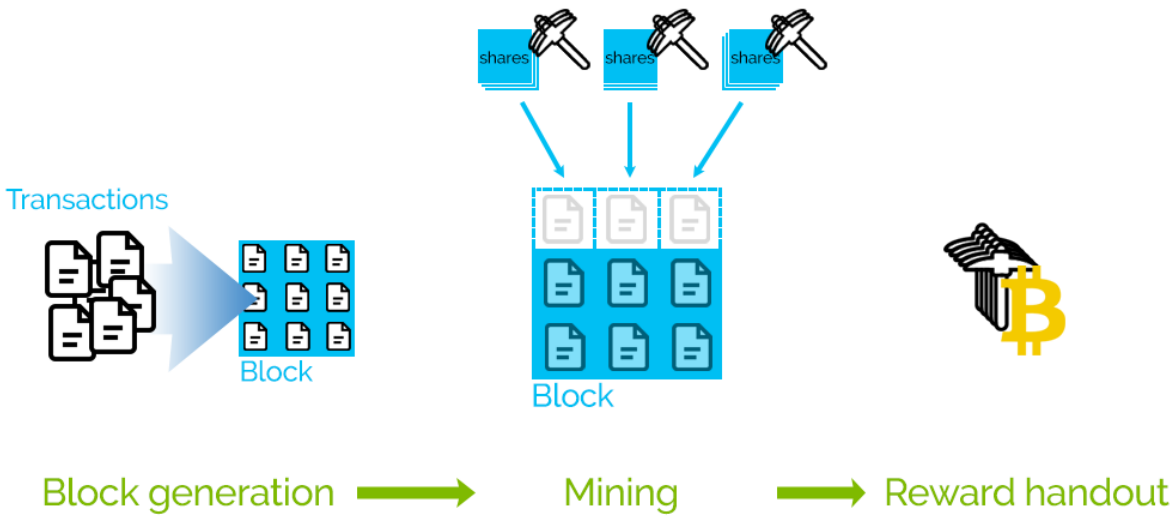

Figura 10: Forma de contribuir a una red basada en *proof-of-work* en grupo

2.2.7. Espacio de una Blockchain en disco

Un problema que se está empezando a generar es la descarga y almacenamiento de la red de blockchain en disco. Aunque los bloques sean de 1 MB cada uno, son muchos, y esto hace que actualmente se esté manejando en Bitcoin una Blockchain de 160 GB. Esto hace que un nodo que se quiera bajar y sincronizar con la red de Bitcoin, de la forma tradicional, actualmente necesite: primero almacenamiento suficiente para la cadena actual, y la futura; después una semana de tiempo para esperar a que se baje, aún con conexiones a internet muy rápidas; y por último, no tener algún fallo en la cadena y tener que repetir el proceso.

Según el cálculo que se realizó en el artículo original, esto no debería haber sido así. Se calculó usando las cabeceras de los bloques, ya que cuando la última transacción de una moneda está enterrada bajo suficientes bloques, las transacciones que se han gastado antes que ella se pueden descartar para ahorrar espacio de disco. Para facilitar esto sin romper el hash del bloque, las transacciones son hasheadas en un Árbol de Merkle[23], incluyendo solo la raíz en el hash del bloque. Los bloques viejos pueden compactarse podando ramas del árbol y los hashes interiores no necesitan ser guardados. En la figura 11, se muestra a la izquierda la estructura de un árbol de Merkle sin modificar, y a la derecha uno "podado", que sería como se guarda en Bitcoin.

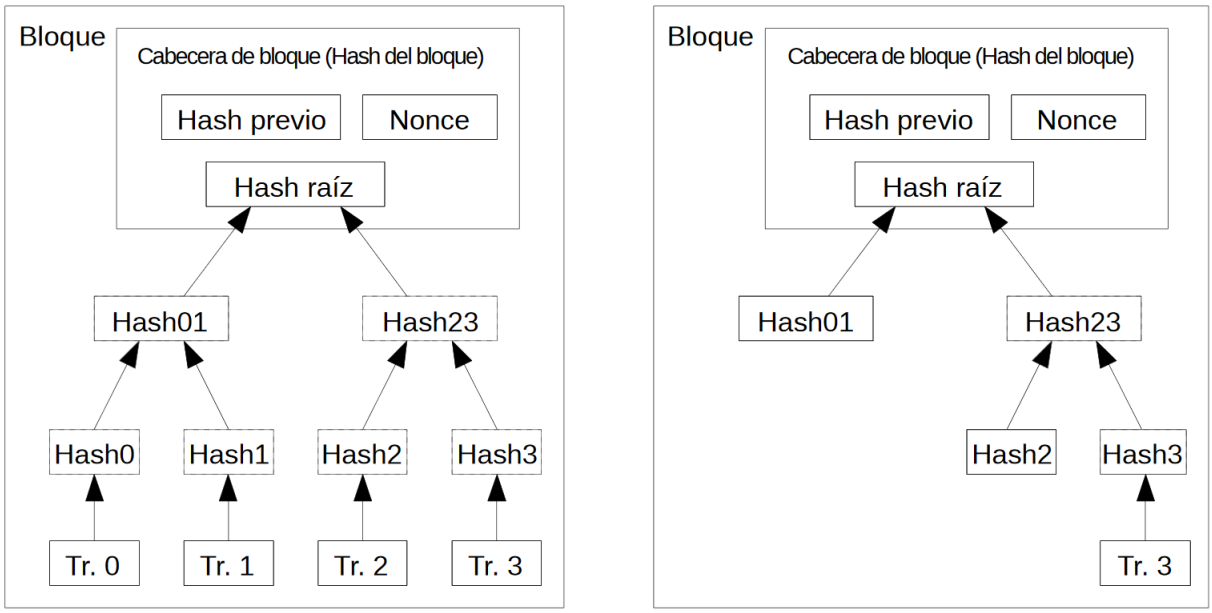

Figura 11: Ahorro de espacio en árbol de Merkle

Los cálculos que se hicieron fueron suponiendo que la cabecera de los bloque sin transacciones pesarían unos 80 bytes. Si suponemos que los bloques se generan cada 10 minutos, 80 bytes×6×24×365 = 4.2MB por año. Siendo habitual la venta de ordenadores con 2GB de RAM en 2008, y con la Ley de Moore prediciendo un crecimiento de 1.2GB anual, el almacenamiento no debería suponer un problema incluso si hubiera que conservar en la memoria las cabeceras de bloque.

Pero la realidad es que actualmente son 160 GB y que es uno de los problemas que tiene la red de Blockchain a día de hoy. Aún usando la opción de sincronizado rápido que se supone que hace uso de estos conceptos teóricos el tiempo necesario sigue siendo de días y no vale para todo este tipo de sincronización.

2.3. Ethereum

Una de las primeras criptomonedas en aplicar cambios a la Blockchain de Bitcoin fue Ethereum[24]. Que fue presentada en *The North American Bitcoin Conference* en Miami, Florida, en 2014 por Vitalik Buterin, Gavin Wood y Jeffrey Wilcke después de haber estado colaborando en la comunidad de Bitcoin.

El artículo original donde se describe el protocolo Ethereum y sus datos técnicos fue llamado "*A Next-Generation Smart Contract and Decentralized Application Platform*"[25] y ya da una idea del enfoque más generalista que tiene en comparación con Bitcoin. En esta sección se va a explicar que hace a esta red distinta de la de Bitcoin y las mejoras introducidas en su propia Blockchain.

Ethereum es una plataforma basada en una Blockchain abierta igual que Bitcoin que usa una Blockchain que no pertenece a ninguna persona o empresa, sino que es un proyecto de código abierto construido por mucha gente alrededor del mundo. Pero tiene algunos cambios importantes que hacen que sea algo distinta y orientada al desarrollo de aplicaciones descentralizadas encima de su red mediante el uso de lo que se denominó Smart Contracts o contratos inteligentes.

La idea original de la que surgió Ethereum fue el crear una sola Blockchain con la capacidad de ser reprogramada para realizar cualquier cálculo arbitrariamente complejo que soportase muchos otros proyectos.

2.3.1. Máquina virtual de Ethereum

Ethereum es una Blockchain programable. En vez de dar a los usuarios una serie de comandos predefinidos, permite crear a los usuarios sus propias operaciones de la complejidad que quieran. Por tanto, esta red sirve como plataforma para, en principio, cualquier tipo de aplicación descentralizada basada en Blockchain. Por tanto, no está limitado a una operación como eran las transacciones en Bitcoin.

Esto se consigue mediante el uso de la Máquina Virtual de Ethereum (*Ethereum Virtual Machine* (EVM)), que puede ejecutar código de complejidad algorítmica arbitraria. O lo que es lo mismo, es Turing completo[26]. Es análogo a pensar en la JVM[27] (*Java Virtual Machine* (JVM)), ya que lo que hace, a nivel funcional, es crear un nivel de abstracción en el que no importe lo que haya por debajo.En el caso de JVM es a nivel de sistema operativo, y en el de Ethereum a nivel de red Blockchain.

En la definición de una red Blockchain, está el de red *peer-to-peer*, y en Ethereum no es distinto y su base de datos esta mantenida por nodos conectados a la red. Cada uno de estos nodos tiene la EVM y está ejecutando las mismas instrucciones. Por esta razón se le denomina a Ethereum como un "Ordenador mundial". El hecho de que Ethereum ofrezca una paralelización enorme de recursos, no está pensado para una computación más eficiente y rápida, ya que eso se consigue mucho mejor en un ordenador tradicional o central. Lo que si consigue Ethereum al tener todos los nodos ejecutando la EVM es conseguir consenso descentralizado, que lleva a unos niveles de tolerancia a fallos muy grandes, un tiempo de inactividad igual a 0 y asegura que los datos que contiene su

Blockchain nunca sean cambiados y no puedan ser censurados.

Aunque en general cualquier aplicación puede beneficiarse de estas ventajas, hay ciertas de estas que pueden sacarle el mayor partido a las propiedades que ofrece esta red. Específicamente, Ethereum integra mejor las aplicaciones que automaticen una interacción directa entre iguales(*peers*) o para facilitar acciones coordinadas en grupo a través de la red. Por tanto, esto está mucho más allá de las aplicaciones financieras en las que se centra Bitcoin, y cualquiera que necesite confianza, seguridad y rendimiento es una aplicación indicada para usar Ethereum. Hay muchos ejemplos, pero los más importantes pueden ser los grupos que abarcan el registro de bienes, no sólo el dinero, votaciones, gobierno, y todo lo que tenga que ver con el Internet de las Cosas.

Bitcoin se centra en las transacciones y puede decirse que es su unidad fundamental. En cambio, la unidad fundamental en Ethereum son las cuentas. La Blockchain de Ethereum sigue el estado de todas las cuentas y todas las transiciones de estado son transferencias de valor e información entre las cuentas. Hay dos tipos de cuentas:

- Cuentas con propiedad externa (*Externally Owned Accounts* (EOA)), que son las cuentas que están controladas por claves privadas.
- *Contract accounts* o cuentas de contrato, que están controladas por su contrato y sólo pueden ser activadas por un EOA.

La diferencia básica entre los dos tipos de cuentas es que los EOAs son controlados por un usuario humano, ya que es él el que controla la clave privada de la cuenta. En las cuentas de contrato eso no es así y son controladas por su código.

2.3.2. Contratos Inteligentes

Este termino no es más que el código que hay en una cuenta de contrato, que son básicamente programas que se ejecutan cuando una transacción es enviada a una cuenta. De esta forma los usuarios pueden crear nuevos contratos creando código en la propia Blockchain.

Por tanto, todas las acciones de la cadena de Ethereum son iniciadas por una EOA y su código de contrato es ejecutado en la máquina virtual de Ethereum distribuida por todos los nodos que contribuyen activamente en la propia red como parte de la verificación de bloques nuevos. En la figura 12 se muestra como un contrato inteligente al ser compilado se genera el código binario Ethereum que puede ser desplegado en la EVM.

La ejecución de estos bloques de código necesita ser totalmente determinista, su único contexto es la posición del bloque en la cadena y los datos disponibles. Los bloques representan unidades de tiempo en la cadena y ésta se puede ver como una dimensión temporal que representa todos los estados como puntos discretos de tiempo designados por los propios bloques de la cadena.

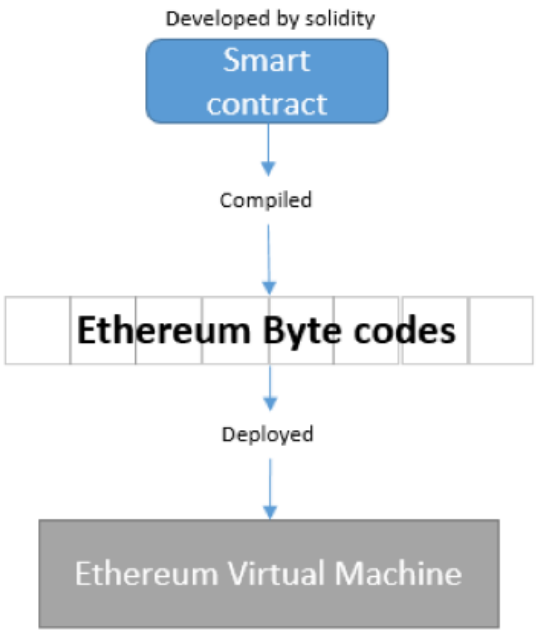

Figura 12: Máquina virtual de Ethereum

2.3.3. Transacciones

El término "transacción" en Ethereum no es el mismo que en Bitcoin. En Ethereum se usa para referirse al paquete de datos firmados que empaqueta un mensaje que va a ser enviado desde una EOA a otra cuenta en la *Blockchain*.

En el caso de Ethereum, cada transacción está compuesta de:

- El destinatario del mensaje.
- Una firma que identifica al remitente y prueba su intención de mandar el mensaje a través de la Blockchain al destinatario.
- El campo de "valor", que representa la cantidad de wei (Todos los Balances de ether son expresados por defecto en unidades de wei [cada ether equivale a 10^{18} wei]) que a transferir desde el remitente al destinatario.
- Un campo opcional de datos, que puede contener un mensaje enviado a un contrato.
- El campo "*startgas*", que representa el número máximo de pasos computacionales que la ejecución puede realizar.
- El campo "*gasprice*", que representa la comisión que el remitente está dispuesto a pagar para "gas"

2.3.4. Mensajes

Los contratos tienen la habilidad de mandar mensajes a otros contratos. Estos mensajes son objetos virtuales que solamente existen en el entorno virtual de Ethereum. Pueden ser vistos como llamadas a funciones.

Cada mensaje está compuesto de:

- El remitente del mensaje de forma implícita.
- El destinatario del mensaje.
- El campo de "valor", que representa la cantidad de wei que transferir con el mensaje a la cuenta de contrato.
- Un campo opcional de datos, que es la entrada de datos del contrato.
- El campo "*startgas*", que limita el gas máximo que puede usar la ejecución de código que el mensaje hace ejecutarse.

Por tanto, los mensajes son como las transacciones y la única diferencia es que un mensaje lo manda un contrato y una transacción la manda un agente externo. Un mensaje es producido cuando un contrato está ejecutando un código que hace uso de las *opcodes* "CALL" o "DELEGATECALL", que generan y ejecutan un mensaje. De la misma forma que las transacciones, los mensajes hacen que se empiece a ejecutar el código del destinatario, así que, los contratos pueden tener relaciones con otros contratos de la misma manera que las tienen los agentes externos.

2.3.5. Gas

Cada nodo que participa en la red Ethereum está ejecutando la EVM como parte de la verificación de bloques. Así que, básicamente, están continuamente cogiendo transacciones de una lista del bloque que están verificando y ejecutan el código que es llamado por esa transacción en la EVM. Hay que remarcar que todos los nodos hacen exactamente las mismas operaciones y almacenan exactamente los mismos valores, cosa que no ocurre en Bitcoin. Por tanto, toda la redundancia de Ethereum es una redundancia en paralelo, que no es óptima pero si una forma eficiente de alcanzar el consenso. El hecho de que el código de los contratos sea ejecutado de forma redundante en todos los nodos lo hace costoso, que generalmente crea un incentivo para no usar la *Blockchain* para cálculos que se pueden hacer fuera de la cadena.

El coste específico de ejecutar cada operación paralelamente en cada nodo tiene se expresa en unidades de gas. Este es el nombre para la comisión que el remitente de una transacción necesita pagar por cada operación realizada en la cadena de bloques de Ethereum. El nombre gas, o combustible en español, viene de imaginar que estas comisiones actúan como criptocombustible, ayudando al movimiento de los *smart contracts*.

Los nodos que ejecutan código compran el gas con ether, así que, son dos conceptos distintos y su valor está desacoplado deliberadamente. El gas va asociado al coste natural que tienen las unidades de computación y el ether está asociado a las fuerzas del mercado. Las dos están en un contexto de libre mercado, ya que el precio del gas lo deciden los nodos de Ethereum ajustando un mínimo con el cual pueden rechazar el procesar transacciones que tengan un precio de gas menor que su mínimo.

El coste total de una transacción se puede calcular usando dos factores:

- *gasUsed*, que es el gas total que es consumido por la transacción. Este valor representa la suma de todo el gas para todas las operaciones ejecutadas. Como el estimar el gas que se va a usar puede ser complicado, hay una API (*Application Programming Interface* (API)) que sirve para estimarlo que puede ser usado.
- *gasPrice*, que es el precio de una unidad de gas especificado en la transacción en unidades de ether. Puede ser tan bajo como cero, y es especificado por cada usuario. Es difícil que un nodo acepte una transacción con un *gasPrice* menor al que se usa por defecto de 0.05e12 wei.

2.3.6. Lenguajes de programación

Un contrato se puede ver como una colección de código y datos que residen en una dirección específica de la cadena de bloques de Ethereum. Estos contratos pueden pasarse mensajes entre ellos, hacer computación Turing completa y son almacenados en la cadena de bloques en un formato binario específico de Ethereum llamado "EVM bytecode".

Este bytecode es de bajo nivel, parecido a ensamblador, así que normalmente se usan lenguajes de más alto nivel y después es compilado en bytecode que ya entienda la cadena de bloques de Ethereum.

El más usado es "Solidity"[30], que es similar a JavaScript[31], y permite a los desarrolladores crear contratos de forma sencilla y compilarlo en bytecode. Tiene mucha documentación y es el más recomendado por Ethereum.

También existe "Serpent"[32], que es similar a Python[33], que es un lenguaje que se usa ampliamente en educación y es muy usado. Así se aprovechan todas las ventajas que ofrece Python en Ethereum. En este caso Serpent se compila en un lenguaje intermedio llamado "LLL"[34] (*Lisp Like Language* (LLL)), que se encuentra entre el Serpent y bytecode, y se puede ver como una especie de plantilla del EVM bytecode.

2.3.7. Acceso a contratos y transacciones

Después de ver cómo se crean los contratos y qué lenguajes se pueden usar, queda ver cómo se comunican los contratos usando la red de Ethereum.

Los nodos de Ethereum tienen una interfaz RPC[37] (*Remote Procedure Call* (RPC)), que provee a las aplicaciones distribuidas acceso a la cadena de bloques de Ethereum y a la funcionalidad que el nodo tiene. Se usa un subconjunto de la especificación de JSON-RPC 2.0[38] como protocolo de serialización y está disponible sobre HTTP[39] (*HyperText Transfer Protoco* (HTTP)) y sobre IPC[40] (*Inter-Process Communication* (IPC)).

Aunque se use la especificación JSON-RPC 2.0, hay dos convenciones que no son parte de esta:

- Los números están codificados en hexadecimal: esta decisión fue tomada debido a que algunos lenguajes no tenían soporte para trabajar con números extremadamente largos.
- Número de bloque por defecto: hay casos en los que no es posible dar un número exacto de bloque o no es muy conveniente, y por esto se ha creado el número de bloque por defecto que puede ser uno de los siguientes valores: [["earliest", "latest", "pending"]

2.3.8. Seguridad

Los usuarios de la red de Ethereum también deben pagar pequeñas transacciones a la red para protegerla de acciones o programas mal intencionados. El usuario que manda una transacción debe pagar por cada paso del programa que active, incluyendo cómputo y almacenamiento, que serán comisiones pagadas en ether, el token nativo de Ethereum. Estas comisiones se las quedan los nodos que trabajan activamente en la red.

dEn este aspecto la única diferencia remarcable respecto a Bitcoin es que el *Proof-of-Work* de Ethereum está pensado para que sea resuelto usando memoria y por tanto con hardware de uso genérico y no específico, como en Bitcoin con las ASICs[41] (*Aplication-Specific Integrated Circuit* (ASIC)), evitando así que la red tienda a la centralización de los nodos y mejorando la seguridad a largo plazo.

En cuanto a la manera de verificar los bloques Ethereum sigue basándose en la *Proof-of-Work* tradicional, como Bitcoin. Aunque se estén planteando pasar a otra forma de generar y verificar los bloques basado en *Proof-of-Stake*[43] que ya está siendo usado en otras criptomonedas y que se explicará en detalle más adelante. Además de *Proof-of-Work* en la red principal y pública de Ethereum, en las redes de prueba, como Kovan y Rinkeby, y en las redes privadas, se puede hacer uso también de *Proof-of-Authority*.

2.3.9. *Proof-of-Authority*

Esta versión del algoritmo no se basa en una prueba computacional, sino en la confianza de los nodos que hay en la red. Básicamente, hay nodos llamados "validadores" o "autoridades" que son los únicos autorizados explícitamente para validar transacciones y crear bloques. Es un concepto muy parecido a la *web of trust*, usada en PGP[42] (*Pretty Good Privacy* (PGP)) en el año 1992. Al

no tener que solucionar problemas matemáticos aleatorios, los tiempos de creación de bloques son predecibles y hay una mejora de la latencia muy considerable.

La primera red en incorporar el algoritmo fue Kovan, una de las redes de prueba de Ethereum, en 2017 y por el momento funciona bien en las redes que están controladas en mayor o menor medida como son las redes de prueba y en las redes privadas, en donde puedes confiar en los nodos que tu controlas y dejar que ellos validen bloques. Suele ser usada en combinación con *Proof-of-Stake* para solventar la problemática que este tiene.

La parte negativa de usar este algoritmo en una red pública, es que no conoces a los nodos, y por tanto no sabes en cuáles confiar para validar los bloques. O, si se generan nodos para validar, se estaría centralizando una parte importante de la red.

2.4. Decred

Ya que se ha introducido el término de *Proof-of-Stake*[43] en el apartado anterior y que, dado que Ethereum no implementa esta filosofía de momento, se va a introducir ahora una criptomoneda que si hace uso de esta forma de verificar transacciones que es Decred[44].

Decred es una criptomoneda lanzada en 2016 centrada en pagos seguros a través de internet mediante el uso de su propia *Blockchain*. Por tanto, es más similar a Bitcoin que Ethereum, ya que no proporciona una plataforma para crear aplicaciones descentralizadas ni nada parecido. Es parecido a Bitcoin, pero con cambios para solventar problemas tradicionales que surgen con las *Blockchain*.

Uno de los problemas que esta criptomoneda quiere evitar es el de que se creen *forks* que normalmente crean una sensación de inseguridad para inversores y pueden crear riesgo en la estabilidad y seguridad de una criptomoneda. También se basa en una mejora de la privacidad y no se permite la censura de nada dentro de su Bloackchain.

No es objetivo de este trabajo entrar en profundidad a explicar esta criptomoneda en particular, solamente se ha elegido por la forma que tiene de validar las transacciones, ya que no es ninguna de las usadas normalmente, sino un híbrido entre *Proof-of-work* y *Proof-of-stake*.

2.4.1. *Proof-of-stake*

El término *Proof-of-Stake* surge puramente a raíz de las *Blockchain*, no como *Proof-of-Work*, y busca llegar a un consenso distribuido en una red basada en cadena de bloques. En este tipo de algoritmos, el creador del bloque siguiente es elegido mediante una combinación entre selección aleatoria y edad o salud llamado "stake". Este stake se refiere a la cantidad de criptomoneda de dicha red posee el nodo, ya que contra más tenga, menos interés tendrá en corromper la verificación de bloques de esa red.

La ventaja más clara frente a *Proof-of-Work* es que es mucho más eficiente energéticamente y que en principio tiende menos a la centralización, ya que en PoW, tienden a generarse grandes

granjas de minado. PoS también tiene problemas estudiados con profundidad como el "nothing-at-stake" o el "costless simulation". En la figura 13 se muestra un resumen de las ventajas de PoS frente a PoW.

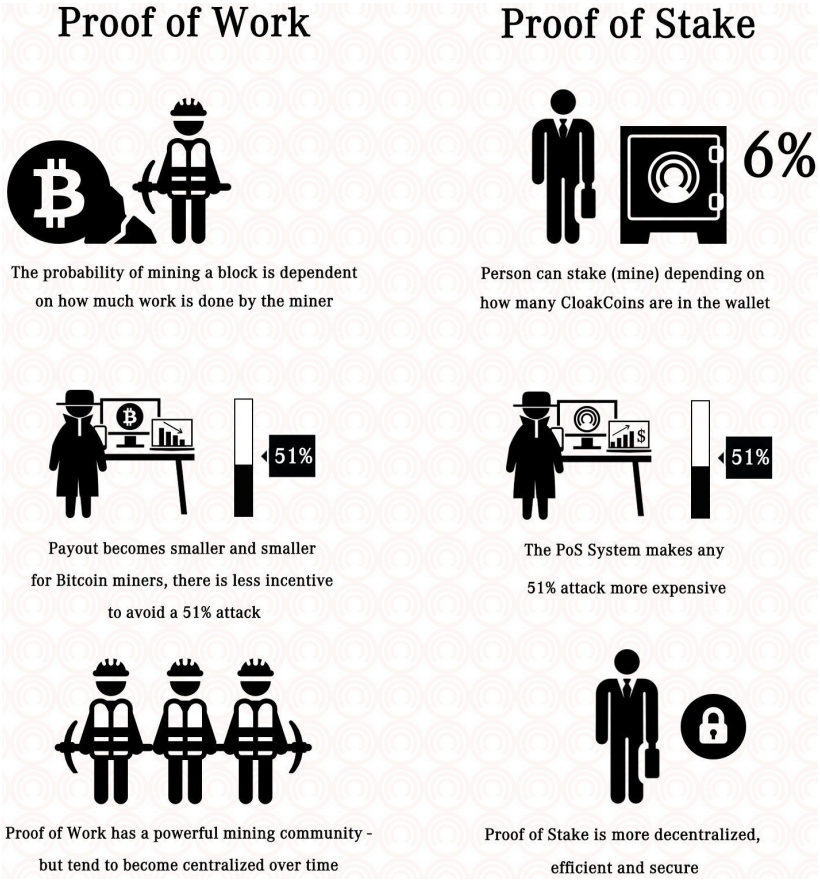

Figura 13: *proof-of-work* vs *proof-of-stake*

2.4.2. Diseño híbrido

La solución usada por Decred es usar un sistema híbrido basado en tickets y sigue el siguiente proceso:

1. Un nodo basado en PoW (*Proof-of-Work*) empieza a generar un bloque y selecciona las transacciones para poner dentro, y se inserta información necesaria para el sistema de PoS(*Proof-of-Stake*).
2. Los nodos basados en PoS votan la transacciones del bloque con su ticket. Este voto habilita el bloque a ser construido encima del bloque anterior y decide si las transacciones anteriores son válidas.
3. Otro nodo basado en PoW empieza a construir el bloque, insertando los votos de los nodos basados en PoS. Esto crea unas banderas que son incluidas al final y, basado en el voto de la mayoría, una bandera de un bit que indica si el árbol de transacciones es válido o no.

4. Se busca un *nonce* que satisfaga la dificultad de la red y el bloque es añadido en la cadena de bloques.

Teóricamente esto implica una mejor escalabilidad con el tiempo de esta criptomoneda que usando cualquiera de los dos sistemas por separado. En este caso, también se implementan algoritmos de firma con curvas elípticas[45] y funciones de *hash* basadas en BLAKE-256[46] para mejorar aún más sus prestaciones y diferenciarse del resto.

2.5. The Tangle

IOTA es una criptomoneda como el resto, pero The Tangle es la tecnología que soporta la red, que poco tiene que ver con una red basada en Blockchain salvo la filosofía y la funcionalidad. Esta tecnología surge a raíz de Blockchain pero la forma de conseguir hacer lo mismo es radicalmente distinta y, consecuentemente, se le va a dar mucha importancia dentro de este documento.

Tal y como se describe esta tecnología en el documento original[47], se ve a Tangle como el siguiente paso en la evolución natural del Blockchain. Aunque esto parezcan palabras mayores, una vez se vean los cambios que ha desarrollado y visto previamente los fallos que tiene Blockchain, no va muy desencaminada esta descripción. Técnicamente se trata de un DAG[48] (*Directed Acyclic Gragh* (DAG)) para almacenar transacciones y ofrece las herramientas necesarias para realizar micro-pagos seguros M2M (*Machine to Machine* (M2M)).

IOTA surge como una criptomoneda centrada en el IoT. En el IoT cobra más relevancia los micro-pagos, y por tanto es en lo que se centra IOTA, aunque no en exclusiva. Uno de los inconvenientes que se esta viendo en la red Blockchain de Bitcoin es el crecimiento de las comisiones en las transacciones con respecto al valor de las mismas. Esto en una red centrada en los micro-pagos es un problema aún mayor, incluso se daría el caso ilógico en el que la comisión tuviese más valor que la propia transacción. Es difícil pensar en una red de Blockchain sin comisiones, ya que éstas sirven como incentivo para que los nodos sigan generando bloques en la cadena, pero se pueden hacer cosas en este aspecto. En la figura 14 se compara gráficamente el aspecto externo de sus estructuras.

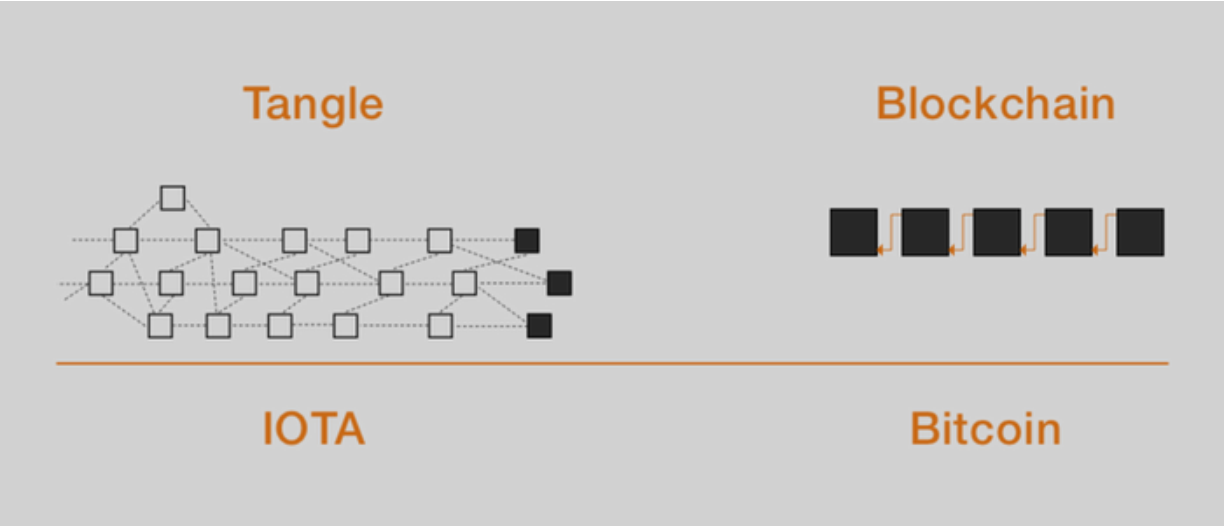

Figura 14: Tangle vs Blockchain topología

Otro problema que tienen las redes basadas en Blockchain es el mantener lo que se conoce como la naturaleza heterogénea del sistema, que es el estado natural de una red *peer-to-peer*. Como ya se ha explicado, en las redes basadas en Blockchain puro hay dos clases de participantes: nodos generando bloques y usuarios haciendo transacciones. Por tanto, en Blockchain se necesita discernir entre los distintos participantes, y esto genera conflictos que necesitan ser resueltos y esto consume recursos. Para resolver esto ya se empieza a ver que se necesita una tecnología que ya no puede ser Blockchain.

Lo que hace Tangle es mantener un libro de cuentas distribuido substituyendo a Blockchain por un DAG. Como un DAG hace uso de la teoría de grafos, de aquí en adelante se empleará también su terminología, ya que si no puede llevar a desentendidos. En esta red, las transacciones hechas por los nodos van a constituir los vértices (*sets*) del grafo tangle, que es el libro de cuentas que almacena las transacciones. Estos vértices se unen mediante aristas unidireccionales que se generan cada vez que una transacción es validada de la siguiente forma: cuando un nodo quiere realizar una transacción, tiene que validar antes dos transacciones anteriores, que serán las aristas unidireccionales que representan la validación de las transacciones. Por tanto tenemos dos conceptos, los nodos de la red que son entidades que hacen transacciones y validan otras transacciones, y en lugar de bloques, ahora las transacciones son almacenadas en un grafo en el que se representan como los vértices de éste.

Como usando esta filosofía no hay necesidad de nodos que sustenten la red ni comisiones para usarlo como incentivo, se va ha realizar una suposición que más tarde será desarrollada. Se define la primera transacción que se genera en el grafo tangle como "génesis" generada por una cuenta que tenía todos los tokens y fueron enviados a las llamadas "cuentas fundadoras", así que todos los tokens han sido generados en esta transacción génesis. Con esta suposición, siempre hay los mismos tokens y no son generados en ningún momento.

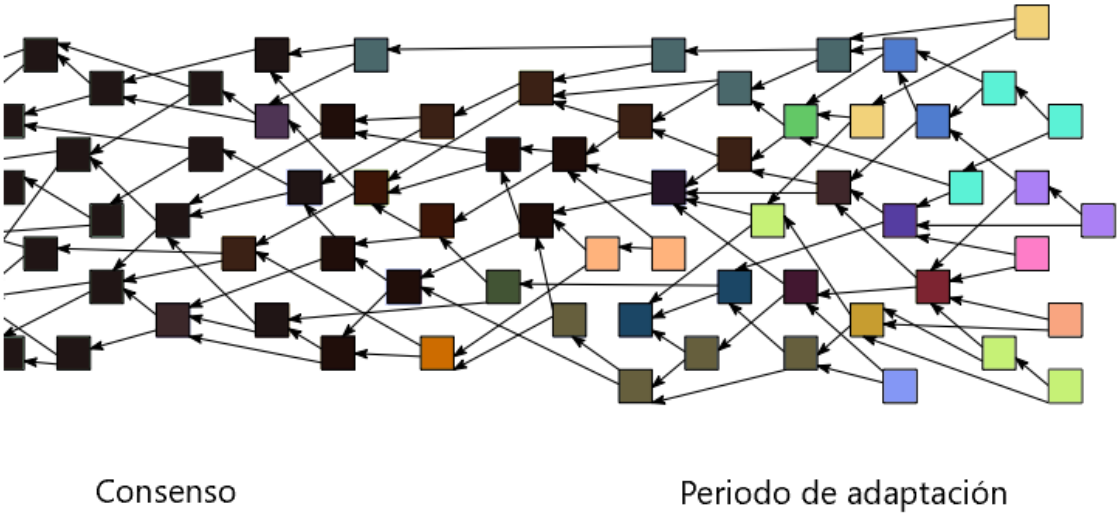

Figura 15: Evolución del consenso en *tangle*

La idea principal de tangle, por tanto, es que si quieres realizar una transacción, tienes antes que validar dos transacciones anteriores, contribuyendo así a la seguridad de la red. El cómo se valida una transacción no cambia, si un nodo no ve ningún conflicto con el historial de tangle, entonces es válida, sino no lo es. De esta forma se va generando un consenso a lo largo del tiempo, a medida que nuevas transacciones validen a las anteriores, tal y como se representa en la figura 15.

Un nodo que quiera realizar una transacción tiene que realizar los siguientes pasos para poder hacerlo:

- Utilizando un algoritmo, decide que dos transacciones anteriores va a comprobar.
- Chequea que las dos transacciones previamente seleccionadas no sean conflictivas.
- Por último para poder hacer su transacción válida necesita resolver un puzzle criptográfico similar al que realizan los nodos en una red de *Blockchain* para añadir bloques a la cadena. Tiene que encontrar un *nonce* tal que la función *hash* concatenada con los datos de la transacción de unos datos especificados.

Una peculiaridad de esta red que ya se puede ver es que es asíncrona, ya que en general los nodos no tienen por qué ver las mismas transacciones. Todo esto se va a ver con más detalle en las subsecciones siguientes.

2.5.1. Pesos

Como en este caso no hay nodos que se dediquen a validar transacciones o descartarlas y a generar bloques ordenados en una estructura lineal y ordenada, sino que es una estructura más

distribuida y caótica y, además, los nodos que generan las transacciones son los que validan otras transacciones, surge una necesidad de medir la confianza que van acumulando las transacciones a medida que van siendo confirmadas por los nodos.

Esto se consigue introduciendo dos clases de pesos:

- El peso de la transacción: este es un peso proporcional a la cantidad de trabajo que el nodo ha invertido en crear la transacción, y por tanto es un valor fijo que se genera al ser creada dicha transacción y queda como una propiedad de la misma. Por razones de seguridad estos números tienen que ser valores 3^n, donde n es un número entero positivo que pertenece a un intervalo no vacío finito.
- El peso acumulado: es el propio peso de la transacción sumado a la suma de los pesos de las transacciones que la han aprobado directa o indirectamente. Por tanto, este valor es variable y puede cambiar cada vez que una transacción es añadida.

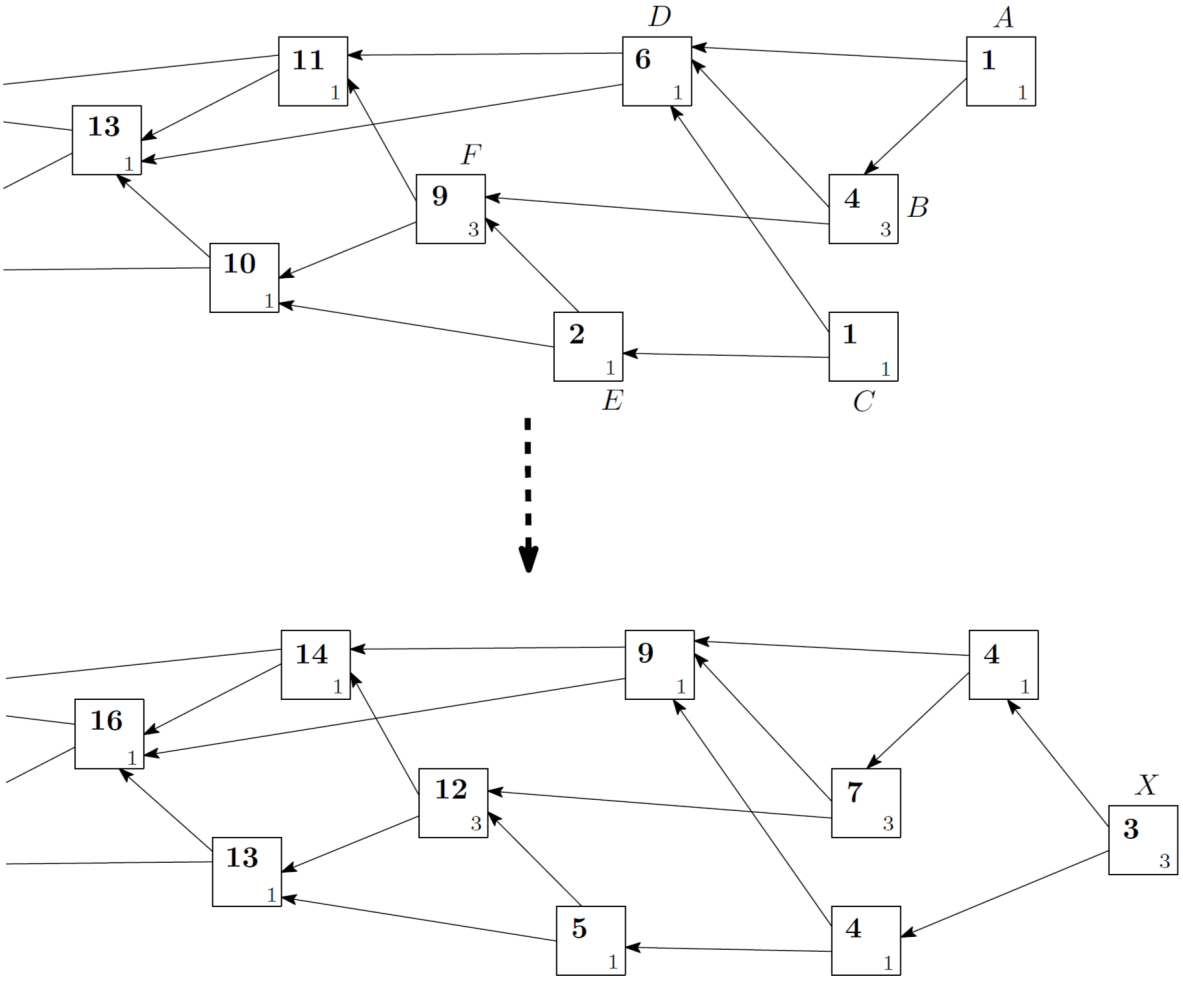

Figura 16: Ejemplo del funcionamiento de los pesos en *tangle*

En la figura 16 se representa una parte de una tangle en la que se muestran las transacciones

con los dos tipos de pesos y como afecta añadir una transacción al final al resto de pesos de las demás transacciones. Las transacciones son los cuadrados que consolidan los vértices. Las aristas son las flechas que representan las verificaciones. El número grande es el peso acumulado, y el número en la esquina inferior derecha es el peso fijado de cada transacción, que en este caso es 1 (3^0) o 3 (3^1).

Por ejemplo, si se mira la transacción F tiene un peso de 3 fijado y un peso acumulado de 9 que sería de sumar el peso de F más las verificaciones directas de B y E más las indirectas de A y C, por tanto, $3 + 3 + 1 + 1 + 1 = 9$. Este valor cambia al añadirse X, ya que verifica las transacciones A y C, que a su vez verificaban indirectamente a F, y por tanto se suma al peso acumulado de F, siendo ahora 12.

Los pesos de las transacciones son importantes a la hora de crear el consenso y usarlo en la seguridad de todo el conjunto. Se entiende que cuanto mayor sea el peso acumulado de una transacción más consenso hay en que dicha transacción es válida, o por lo menos más válida que otra de peso menor.

2.5.2. Puntas

En tangle, se llaman puntas a las transacciones que no han sido aprobadas, ya que de forma gráfica éstas se encuentran al final del grafo como si fueran sus puntas. Por ejemplo, en el estado inicial de la figura 16 había dos puntas A y C, pero cuando se introduce X se convierte en la única punta del grafo.

En el algoritmo de selección de transacciones a validar, que será explicado posteriormente, se intentará que las transacciones a validar sean preferentemente puntas o, en su defecto, transacciones con el menos peso acumulado, ya que así se ayuda a llegar antes al consenso de las transacciones que están más dentro en el grafo.

2.5.3. Altura, profundidad y puntuación

Los tres últimos conceptos de este tipo de redes es la altura, la profundidad y la puntuación, que son tres tipos de métricas para las que se requería saber lo que era una punta y la transacción génesis.

- Altura: es la longitud del camino más largo hasta la transacción génesis.
- Profundidad: es la longitud del camino más largo hasta una punta.
- Puntuación: es la suma de los pesos de todas las transacciones que ha aprobado directa o indirectamente más el propio peso de esa transacción. Se intenta aclarar con el ejemplo mostrado en la figura 17.

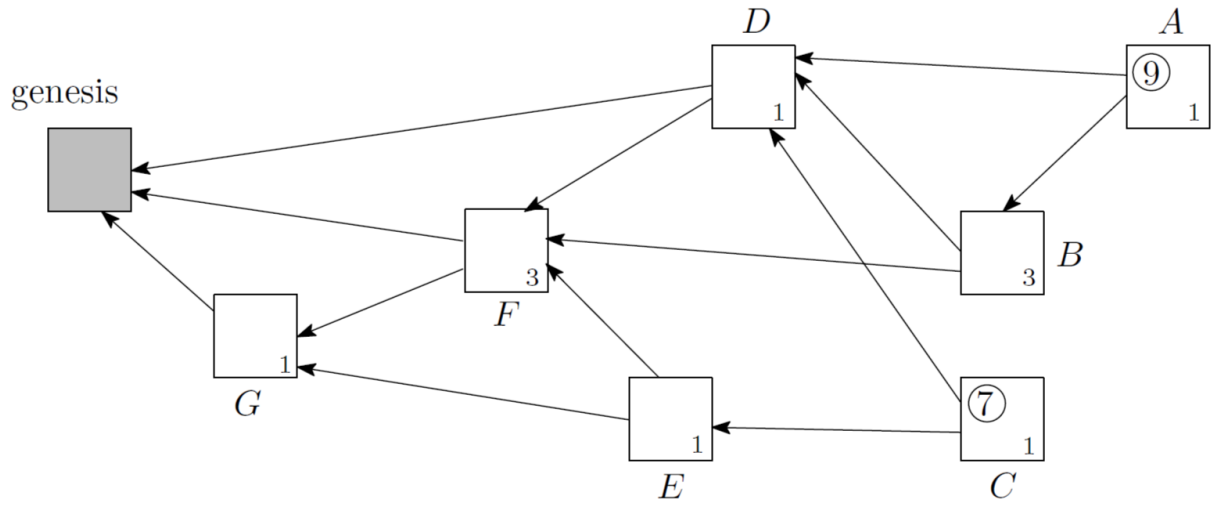

Figura 17: DAG con las puntuaciones de A y C calculadas

2.6. Conclusiones

Hasta el momento, en este trabajo, se ha comprobado como han ido surgiendo nuevas formas de implementar criptomonedas basadas en la idea principal de la Blockchain de Bitcoin, mejorando esos aspectos clave que hacían tener problemas a dicha red. Se ha explicado como Ethereum usando la misma tecnología básica ha sido capaz de desarrollar una plataforma en la que cualquier desarrollador puede crear aplicaciones descentralizadas mediante el uso de lenguajes de programación de alto nivel y usando su Blockchain para desplegarlas. También se ha visto una nueva forma de validar transacciones aunque tenga fallos y por eso actualmente sólo se estén usando (mayormente) sistemas híbridos como el de Decred. Y por último, se ha entrado a ver en detalle una forma radicalmente distinta de implementar la misma filosofía que Bitcoin pero sin siquiera hacer uso de una Blockchain y sustituyendo ésta por The Tangle para mejorar su escalabilidad.

Todo esto da una idea de lo mucho que se puede hacer partiendo de una misma idea, y de cómo son capaces los sistemas abiertos y la comunidad de generar proyectos tan variados y tan radicales, tan rápido y con un impacto muy grande, en el que todo el mundo puede participar, aprender y usar desde cualquier punto del mundo estos proyectos.

Las aplicaciones son muchas y muy variadas. Hay muchas en marcha, muchas pensadas y muchas sin pensar, pero con plataformas como la creada por Ethereum, cualquier idea puede ser implementada, sólo hace falta que sus características aprovechen el potencial de la tecnología subyacente.

Es una tecnología nueva que coge conceptos anteriores usados en diversos campos para solucionar un problema concreto. Pero su uso no está restringido al intercambio seguro de dinero digital, sino que se está investigando su uso desde hace años en otros campos muy diversos. Aún se está probando su uso a largo plazo, ya que como se ha visto puede tener fallos que pueden ser críticos.

3. Diseño

En este capítulo se comentarán los aspectos más importantes acerca del planteamiento y el diseño de la solución a desarrollar en este proyecto.

3.1. Descripción del problema

Las ciudades inteligentes normalmente se van desplegando de forma continuada en el tiempo, según las necesidades que vayan surgiendo y la inversión de la que se disponga. Por tanto, su forma no es fija y depende de cómo se vayan desplegando los nodos a lo largo de la geografía de la ciudad correspondiente.

De este modo, se va conformando una red de nodos interconectados entre sí sin ninguna jerarquía concreta, por lo tanto se trata de una red *peer-to-peer* variable en el tiempo y descentralizada o distribuida, en la que los nodos van recolectando datos de los distintos puntos de la ciudad.

El problema surge cuando estos datos necesitan ser transportados y almacenados. Siguiendo una visión tradicional del funcionamiento de estas redes, los datos son generados, recogidos o procesados por los nodos que después los envían a un servidor o servidores centrales en donde se almacenan todos los datos. De esta forma, es mucho más fácil tener los datos ordenados y accesibles por todo el que los necesite.

Este tipo de tratamiento de los datos rompe la naturaleza descentralizada de la red y no aprovecha los beneficios que presenta. Al estar los datos en un sólo sitio también puede presentar riesgos de seguridad o pérdida de datos.

Por lo tanto, el problema que se aborda en este proyecto es intentar aprovechar al máximo los beneficios que presentan este tipo de redes descentralizadas preservando en la medida de lo posible su naturaleza y aplicando, para ello, una tecnología en auge y prometedora como es el Blockchain.

3.2. Planteamiento de la solución

Como ya se ha comentado en las secciones previas, el Blockchain se basa topológicamente en una red *peer-to-peer* en la que no hay jerarquía, solamente nodos interconectados entre sí a través de Internet, al igual que las redes de las ciudades inteligentes.

Estas similitudes fueron vistas en el momento de plantear este proyecto, viéndose como una oportunidad de integrar esta tecnología en las redes diseñadas para las ciudades inteligentes, encargándose de las tareas de recolección, acceso e intercambio de datos que, por definición, en estas redes son descentralizados y tratarlos como tal.

De esta forma, podríamos llegar a tener nodos que recogen datos, ellos mismos lo almacenan en forma de bloques dentro de la cadena que está formado por todos estos nodos y, de esta forma,

estarían todos los datos distribuidos en todos los nodos y accesibles además por cualquiera que lo solicite. En un principio la idea de diseño es que sean los propios nodos de la red de la ciudad inteligente la que soporte la red Blockchain, pero no todos los nodos son iguales, y habrá que tener en cuenta las capacidades computacionales de cada uno.

Una vez definido claramente el objetivo, se plantean las diversas formas de abordar el problema para llegar a este objetivo. El problema de usar Blockchain es que no es una tecnología que esté completamente definida, sino que cada implementación tiene sus particularidades, ya que fue creada para una aplicación en particular y se ha ido generalizando hacia otras, al contrario de lo que suele suceder con otras. A continuación se plantean las posibilidades de las que se dispone:

- La primera opción, y más obvia, sería crear nuestra propia *Blockchain* específica y así controlar cada parámetro de la creación de la misma para que encajase perfectamente con las necesidades de las redes de sensores de las ciudades inteligentes. Esta opción, aunque puede ser la mejor de cara a largo plazo teniendo más recursos, para este proyecto no es viable. Se va fuera de los limites de este tipo de proyectos. Además, corre el riesgo de que se generen brechas de seguridad o escalabilidad.
- La segunda opción, sería recurrir a otras Blockchain ya creadas para sus aplicaciones específicas y adaptarlo para ser usada para abordar el problema planteado en este proyecto. Esta opción es más viable por lo menos para realizar una primera aproximación a lo que podría ser una implementación real para el tiempo del que se dispone. Dentro de esta segunda opción se plantean diversas soluciones, siendo todas opciones viables, en principio, para continuar desarrollando esta idea.

En primer lugar, está la Blockchain de Bitcoin. Ésta es en principio descartada ya que es la primera implementación, creada para una aplicación específica y de la que se ha visto que tiene problemas para implementarse tal cual para otro tipo de aplicaciones que no sea el intercambio de dinero virtual. Además, han ido surgiendo variantes de ésta mucho más flexibles y aplicables a otras aplicaciones.

Una de estas variantes es Ethereum. Ethereum es una Blockchain basada en proporcionar una capacidad de cómputo distribuido en donde ejecutar aplicaciones propias a cambio de pagar una especie de impuestos por usarlo. Para ello especifica una API y una serie de lenguajes de programación de alto nivel para crear dichas aplicaciones.

Otra de las opciones contempladas para ser usada es Decred. La particularidad que tiene esta Blockchain es que utiliza un híbrido de *Proof-of-Work* y *Proof-of-Stake* que, según ellos, asegura mayor nivel de seguridad y privacidad, siendo los usuarios de esta red incapaces de ser rastreados. También tiene facilidades para ser usada por terceros y para desarrollar encima de ella.

La última de las opciones valoradas en este proyecto es Tangle, que de por sí no encaja en la definición de Blockchain como tal. Es una evolución de esta tecnología que está enfocado a solventar todos los problemas de escalabilidad al usarse para un número elevado de micro-pagos mantenidos en el tiempo. Esta opción, en principio, puede ser la mejor para implementar la idea principal de

este proyecto, ya que su estructura es la que mejor cuadra con los requerimientos de las redes de las ciudades inteligentes.

Después de valorar todas estas opciones y ver cuál de ellas se ajusta más a las limitaciones de este proyecto, se decide usar Ethereum como red Blockchain que soporte la ciudad inteligente y se encargue de mantener la descentralización de los datos y gestionar su creación y acceso. Se elige esta opción porque es la única que esta pensada para la creación de nuevas aplicaciones y no para soportar una criptomoneda en particular, además de la gran comunidad que hay detrás y las herramientas que hay creadas por encima de esta red para crear aplicaciones nuevas y para realizar desarrollos sin tener que tocar nada del protocolo que hay por debajo.

3.3. Descripción funcional

En este apartado se pasa a nombrar y describir cada uno de los elementos funcionales que componen el sistema en su totalidad. El orden en el que se irán describiendo los mismos será empezando por los niveles más bajos hasta llegar al nivel más alto.

Los nodos conforman el nivel más bajo de este sistema, conforma la totalidad del hardware y no necesitan ningún tipo de hardware adicional para desarrollar todas las funciones de una ciudad inteligente. Estos nodos estarán distribuidos de la misma forma que se encontrarían en una ciudad inteligente tradicional. Las dos funciones más relevantes de estos nodos son las siguientes:

- La principal función que realiza el conjunto de nodos de esta red es soportar la red de Blockchain de Ethereum. En esta función se incluye la de generar los bloques y almacenarlos de forma distribuida. Aunque no todos tienen que encargarse de estas dos últimas, ya que no todos tienen que desempeñar las funciones de mineros y pueden ser simples nodos de la propia red. No todos tienen que contribuir activamente en la red, y serán los más dotados computacionalmente los que lleven el peso.
- La segunda y más importante función sería la recolección de los datos de la ciudad tal y como haría cualquier sensor tradicional y subirlo a la red, en este caso la red Blockchain y por tanto ser almacenada en un bloque de esta.

El segundo elemento funcional de este sistema es la propia red Ethereum soportada por los nodos y, por tanto, por encima de éstos. La red de Ethereum, a su vez, se encarga de soportar toda la cadena de bloques, así como su gestión interna a través de todos los protocolos que lo conforman. Esta red es de código abierto y está disponible en Github, que asegura un desarrollo continuo y abierto, lo que es bueno de cara a la seguridad y a la continuidad de la red en el tiempo. Lo más importante es que proporciona un único punto de acceso a la red desde Internet y eso permite tratar a toda la red de nodos distribuidos cómo un único ente, y es la red la que se encarga de repartir el procesado entre los nodos. Esta repartición computacional se hace a través de la ya explicada (sección 2.3.1) máquina virtual de Ethereum y se refiere a los nodos mineros de la red.

El tercer elemento funcional del sistema se correspondería con los *Smart Contracts* o contratos inteligentes, que no es más que código que se ejecuta en la máquina virtual de Ethereum.

En estos contratos se describe la funcionalidad especifica de cada aplicación descentralizada, en este caso se especificará el formato de los datos almacenados, cómo se almacenan, así cómo la gestión del acceso a dichos datos almacenados en la cadena de bloques.

Un último elemento funcional que puede ser considerado en este sistema pueden ser los nodos externos a la red que quieran o bien introducir datos nuevos a la red que conforma la ciudad inteligente, o bien consultar datos que se hayan generado en la propia red.

En la figura 18, se plantea, para cerrar esta sección, la idea de diseño que se ha pensado con todos los elementos funcionales que se han ido describiendo por separado. En ella, se muestra como los datos generados por la ciudad inteligente son recogidos por los nodos de la red, que pueden ser simples sensores que sólo recogen datos y los transmitan, o nodos más potentes que sí ayuden a mantener la red Blockchain. Estos nodos, mediante las funcionalidades que permita el contrato inteligente, almacenan esos datos en forma de bloques de la cadena. Con el PC se ha querido representar cualquier usuario externo a la red que quiera ver datos almacenados dentro de ésta.

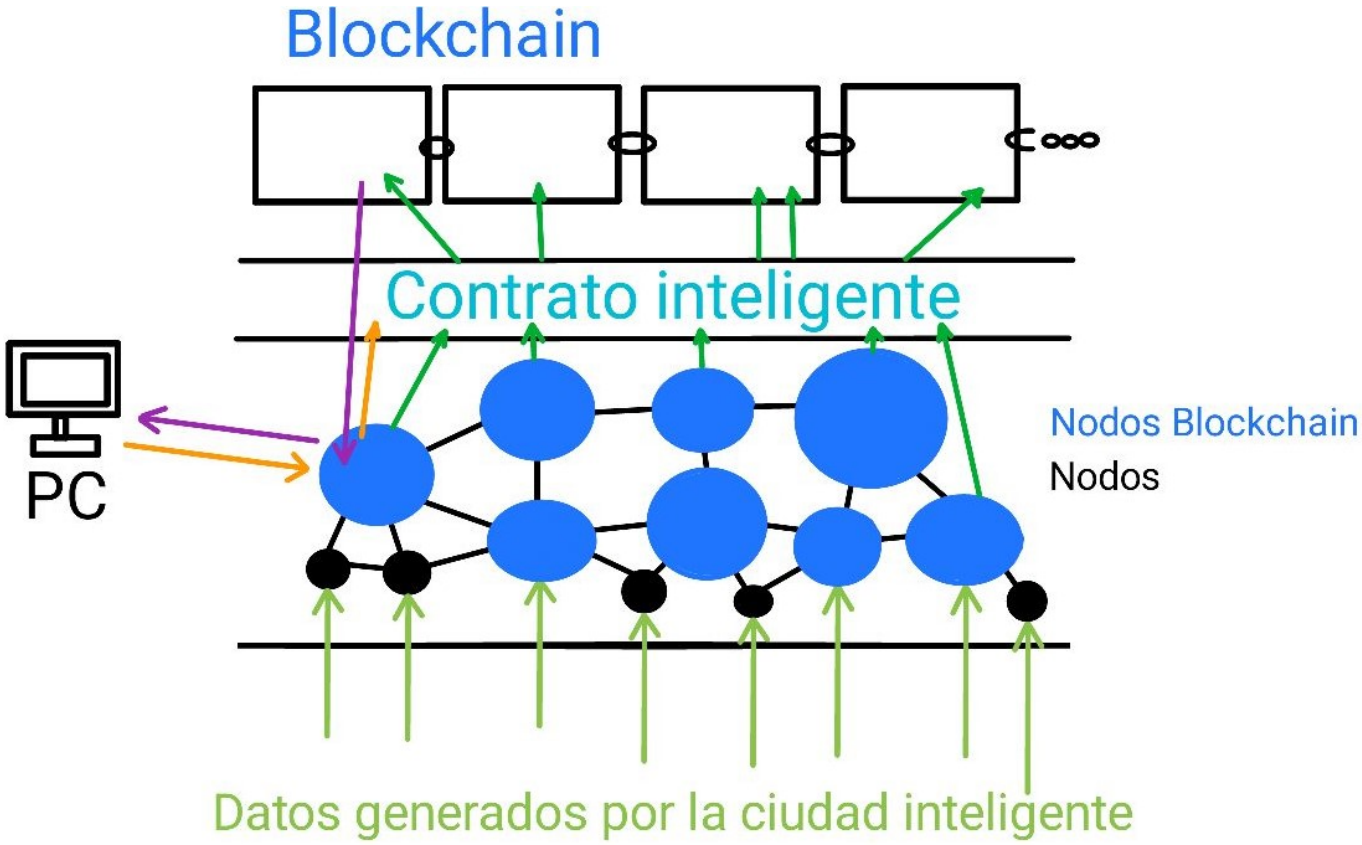

Figura 18: Diseño del sistema

4. Implementación

En este capítulo se detalla el proceso que se ha llevado a cabo para implementar en un sistema real la solución planteada en el capítulo anterior, teniendo en cuenta las limitaciones que presenta este tipo de escenarios.

4.1. Creación de la red privada de Ethereum

El primer paso, es montar una red Ethereum que soporte la lógica que se irá desarrollando posteriormente. En este punto se decide implementar una red privada de Ethereum, ya que, al final, las redes de las ciudades inteligentes suelen ser redes locales con un punto (o varios) de interconexión con Internet y una red privada de Ethereum sigue la misma filosofía. Además, al ser algo propio que no depende de terceros se tiene mayor control de la red. Otra cosa a tener en cuenta, es que al ser una red privada, no se usa dinero real, cómo si tendría que usarse si se quisiera montar sobre la red pública de Ethereum.

Para implementar dicha red se decide usar Raspberry Pi a modo de nodos de ésta, emulando los nodos reales de la red de la ciudad inteligente.

El proceso de creación de la red[35][36] requiere seguir una serie de pasos y repetirlos en cada uno de los nodos que vayan a conformar la red, en este caso tres nodos. Se enumeran los pasos a seguir y posteriormente se explica en detalle el procedimiento:

1. Configuración de los nodos: incluye la instalación del software necesario en la Raspberry Pi y el registro en Ethereum
2. Inicialización del nodo, mediante la creación y configuración del archivo génesis
3. Configuración del nodo *bootnode*
4. Arranque del nodo

Para que las Raspberry Pi sean dotadas de todas las funcionalidades de las que dispone un nodo Ethereum hace falta una herramienta software que nos permita su configuración. Se decide usar Geth[49], ya que es la herramienta oficial para realizar este tipo de configuración. Sirve, entre otras cosas, para comunicarse con redes Ethereum mediante JSON-RPC[38], usar una consola interactiva Javascript, usar comandos propios para la configuración de redes Ethereum, realizar transacciones y crear contratos inteligentes. Además, Geth soporta todo tipo de sistemas operativos y arquitecturas, entre los que se encuentran RaspbianOS y ARM[50].

Mediante el uso de esta herramienta, ya se puede crear una cuenta Ethereum, ya que sin esta, no se pueden hacer transacciones a nivel de red Blockchain de Ethereum. Al generarse esta cuenta, se almacena localmente la clave privada asociada al nodo, y su contraseña, que será necesaria para desbloquear la cuenta cada vez que quiera ser usada.

Ahora que ya se dispone de una cuenta Ethereum desde la que operar en la red, hay que fijar los parámetros que van a definir dicha red. Para hacer esto, se necesita crear un archivo de configuración que los contenga, y que luego sea usado por el nodo para crear el primer bloque de la red Blockchain, o bloque Génesis. Este es el paso más importante de todo el proceso de creación de la red privada, y hay que tener en cuenta que estos valores no pueden ser modificados sin borrar toda la cadena de bloques y crear otra red nueva.

Los parámetros más relevantes que contiene dicho archivo de configuración del bloque Génesis se nombran a continuación:

- Tipo de red: Proof-of-Work o Proof-of-Authority
- Tiempo entre bloques en segundos
- Cuentas autorizadas para crear bloques
- Cantidad de Ethereum que van a tener las cuentas en el momento de arrancar la red
- ID de la red

Las redes privadas de Ethereum permite la elección del algoritmo que se va a usar para la verificación y creación de nuevos bloques de la cadena. La red pública de Ethereum usa exclusivamente *Proof-of-Work* pero hay varias redes de prueba que usan tanto *Proof-of-Work* como *Proof-of-Stake* y *Proof-of-Authority* o una combinación de éstas.

Para la red privada, en un principio se usó la opción por defecto, que es *Proof-of-Work*. Este algoritmo, que está explicado en profundidad en el apartado 2.2.3, se basa en pasar una prueba computacional que realizan los llamados mineros. En este punto se necesitaba el apoyo de un ordenador, ya que las Raspberri Pi no pueden hacer las funciones de mineros porque necesitan más memoria RAM para cargar el DAG. Por lo tanto, se tiene que introducir un nodo más potente que permita realizar las funciones de minero.

En esta primera versión de la red privada de Ethereum, se usaron dos Raspberry Pi a modo de nodos normales, encargados de inyectar y leer información a la red; y un ordenador en el que, lógicamente, había corriendo dos mineros, que se encargaban de mantener la cadena de bloques, generándolos y verificándolos.

Como esta implementación de la red no es la que se buscaba en el diseño de la misma, se cambió el algoritmo de verificación y creación de bloques a *Proof-of-Authority*, explicado anteriormente en la sección 2.3.9. El usar esta versión del algoritmo, nos resta algo de seguridad, pero al ser una red privada controlada, no debería de haber ningún problema. Además ofrece una ventaja clave, como es la de quitar la necesidad de tener nodos minando en la red, y usar solamente los nodos que conforman la red de la ciudad inteligente, manteniendo aún más la descentralización. Esto conlleva una notable reducción del gasto energético, que es muy importante en las redes del Internet de las Cosas.

Elegido el tipo de red a implementar, se puede seguir entonces configurando la misma. Usando el mismo archivo de configuración del bloque Génesis, se inicializa cada nodo, ya que todos van a estar en la misma red y tiene que tener dichos parámetros iguales.

Para sincronizar todos los nodos, y que mantengan la misma versión de la cadena de bloques, es necesario que haya al menos un nodo llamado *bootnode*. Este nodo es el encargado de mantener dicha sincronía y todos los demás nodos deberán estar conectados a él tanto para conectarse por primera vez, como para descubrir vecinos o para cualquier otra de las tareas propias de las redes *peer-to-peer*. Este nodo tiene una pequeña carga computacional más que el resto y debería ser más estable que el resto, ya que si se cae, se perdería la sincronización de la red. En este caso se decide implementar esta funcionalidad en la Raspberri Pi versión 3 (se disponen de dos RPI 2 y una RPI 3). Este nodo será el único que requiera tener una dirección IP estática y una URL de tipo *enode* para que el resto sepa dónde se tienen que conectar.

El último paso para terminar de configurar la red privada es inicializar y arrancar el nodo mediante el siguiente comando:

```
1    $ geth --datadir node/ --syncmode 'full' --port 30310 --rpc --rpcaddr ...
       '12.12.0.209' --rpcport 8501 --rpcapi ...
       'personal,db,eth,net,web3,txpool,miner' --bootnodes ...
       'enode://4e543e6b3a5ebe21dad781458ada40055d187cb0df110a73d57d45470edd3d0bd6d7e ...
       5f686ba18290412d6baa0f912bc9224812fda1d23420746820b980b2b08@12.12.0.209:30310' ...
       --networkid 45 --gasprice '1' -unlock ...
       '0x3a0267c343a54e9332819231202df6b0efd6cf52' --password node/password.txt ...
       --mine
```

- –datadir ruta dónde se va a ir almacenando los archivos necesarios para el funcionamiento de la red
- –syncmode 'full' ayuda a prevenir la propagación de bloques mal descartados.
- –port es el puerto de nodo que lo identifica en la red *peer-to-peer*.
- –rpcapi habilita el uso de forma remota de la consola javascript de geth para controlar los nodos Ethereum mediante llamadas RPC usando la dirección y el puerto definidos con –rpcaddr y –rpcport
- –bootnodes especifica la URL en formato enode donde encontrar los *bootnodes*. En este caso es él mismo.
- –networkID especifica el identificador de red especificado en el génesis
- –unlock –password sirven para desbloquear la cuenta con la contraseña almacenada previamente
- –mine le dice al nodo que empiece a votar/sellar bloques. se mantiene la nomenclatura del PoW

En la figura 19, se muestra una captura con los tres nodos funcionando. El primero y el segundo son los autorizados para firmar transacciones y crear bloques, el tercero es uno no autorizado

para ello. El primer nodo además es el que se comporta también como *bootnode* y se pueden ver los *logs* que genera como tal. También se puede ver cómo cada uno de los nodos genera su propio punto de interconexión mediante IPC y RPC, desde los que se puede abrir una consola Javascript e interactuar con ellos. Se puede observar, además, como los nodos 1 y 2 son capaces de sellar (*seal*) bloques, aunque también se mantenga la nomenclatura de minar aunque sea una red basada en PoA.

En esta figura se refleja también el campo *gas* y *fees* están a cero. En nuestra propia red privada no tiene sentido pagar por las transacciones que se generan en la red o por el uso computacional que se haga de ella. Aunque en algunos casos si que tienen que usarse estos campos, no importa ya que el Ether en las redes privadas es "gratuito".

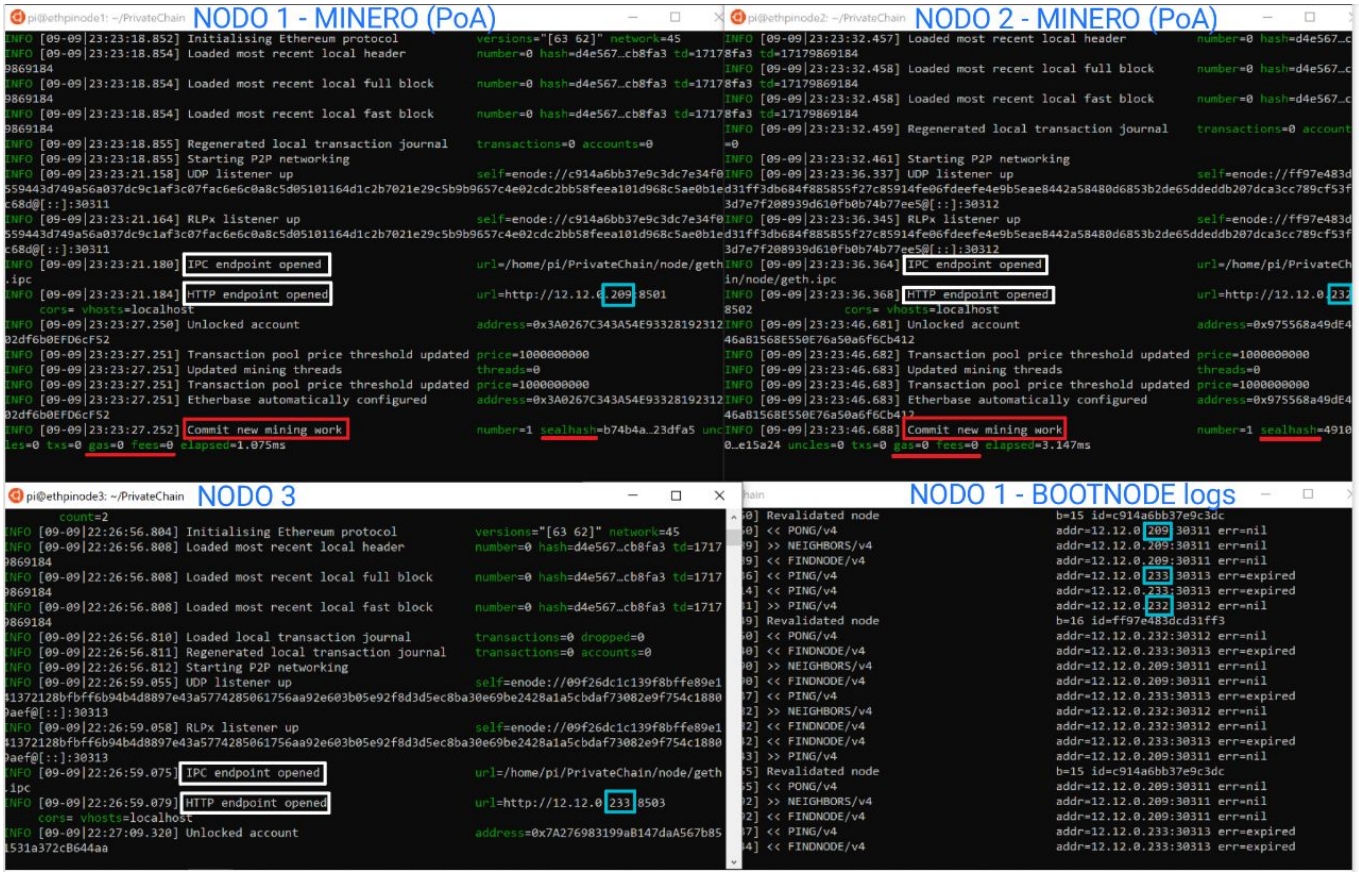

Figura 19: Red privada Ethereum basada en *Proof-of-Authority* funcionando y sincronizada

4.2. Desarrollo del contrato inteligente

Si bien la red ya está conformada, resta darle al sistema la inteligencia necesaria para que pueda dotar a la red de las funcionalidades que fueron pensadas en el Diseño del mismo. En este tipo de redes, esto se consigue mediante el uso de los contratos inteligentes.

De entre todos los lenguajes de alto nivel descritos para la definición de un contrato inteligente, en este caso se opta por el más usado y, por tanto, más documentado, que es Solidity.

Este lenguaje es simplemente una versión limitada de Javascript, adaptado a las peculiaridades del Blockchain. Básicamente se gestionan dos funcionalidades en la red:

- **Almacenamiento y lectura de datos en la cadena de bloques**

  Se implementa en dos funciones, una de escritura de datos y otra de lectura de datos. Estos datos están estructurados y ordenados para que puedan ser almacenados y leídos eficientemente. En particular, se almacena la siguiente información:

  - Fecha y hora en el que ha sido almacenado en la cadena de bloques
  - Dirección Ethereum que lo ha almacenado en la cadena de bloques
  - Tipo de dato almacenado en la cadena de bloques
  - Valor del dato almacenado en la cadena de bloques
  - Unidad del valor almacenado en la cadena de bloques

  Como opción de diseño se ha planteado que en nuestro caso la información sea almacenada en la propia cadena de bloques, dentro de cada transacción, en su campo de datos. Si bien es la mejor solución en este caso, ya que no se rompe la filosofía de que sean los propios nodos los que soporten todo el peso de la red, a medida que se quieran realizar cosas más complejas, el almacenamiento no será suficiente, y habrá que recurrir a otras opciones de almacenamiento. Una de estas opciones puede ser externalizar dicho almacenamiento fuera de la red, o almacenar solamente las referencias o el hash, con el que luego acceder a los datos reales.

- **Gestión de Acceso a los datos y seguridad**

  Ya se dispone de una red en la que, mediante las funciones previamente mencionadas, cualquiera puede introducir datos para que sean almacenados en ella y también leer los datos que quiera. Para dotar a esta red de una capa de seguridad básica, se a planteado gestionar el acceso a los datos en base a las direcciones Ethereum.

  En un primer momento se configura solamente una dirección Ethereum válida que puede inyectar datos en la red o leerlos, llamada dirección maestra. Esta dirección puede añadir más direcciones, que, desde eses instante, podrán leer y almacenar datos.

  Para tratar de impedir de una manera sencilla que no haya nodos que simplemente se dediquen a consumir datos generados por otros en la red, también se implementa un sistema básico de tokens. Simplemente se dan 5 tokens cada vez que se añaden datos, y se quita un token cada vez que se consumen datos. Así, para consumir cinco datos, tendrás que haber aportado por lo menos uno. Esto es totalmente configurable, y se puede balancear incluso de forma dinámica.

Para desarrollar las funcionalidades descritas en un contrato inteligente se a usado el entorno de desarrollo Remix[51]. Esta herramienta funciona como cualquier entorno de desarrollo, así que, corrige fallos de sintaxis, compila el código online e incluso puede desplegarlo en una de las redes que

Ethereum tiene desplegadas cómo redes de prueba[52], o en cualquier red privada de Ethereum. Ha sido usada para las primeras fases del desarrollo del contrato inteligente.

En la figura 20, se muestra la representación que hace el entorno de desarrollo Remix de cada función descrita en el contrato inteligente. Como se puede apreciar, las principales funciones son las de *writeData* y *readData*, que tienen la estructura de datos explicada previamente.

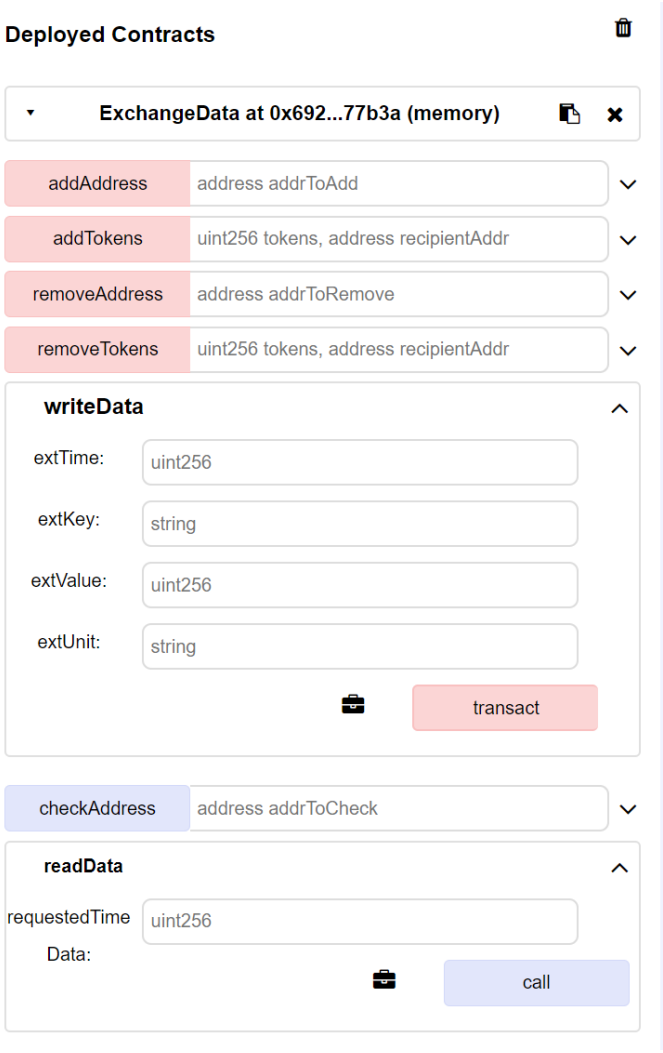

Figura 20: Representación gráfica de las funciones del Contrato inteligente

Posteriormente se han usado otras dos herramientas para realizar una simulación más real de lo que va a ser la implementación final. Para compilar un contrato inteligente sin recurrir a herramientas online, se puede hacer uso de Truffle[53], que permite el desarrollo de contratos inteligentes en varios lenguajes y su posterior compilación y despliegue en una red Ethereum. Esta es la herramienta de desarrollo más usada para la creación de aplicaciones distribuidas actualmente.

Junto con Truffle, se suele usar Ganache[54], que permite simular una red privada Ethereum real con los nodos virtuales que se quieran en un ordenador, de manera similar a usar máquinas virtuales. En esta red simulada se pueden mandar comandos , desplegar contratos inteligentes y

todas las cosas que se podrían hacer en una red real, todo esto mientras visualizas en tiempo real las transacciones, creación de bloques y los *logs* que se van generando la red.

Por lo tanto, este conjunto de herramientas han servido para crear una simulación dónde poder ver cómo se comporta el contrato inteligente creado con anterioridad. En la figura 21 se muestra la transacción correspondiente al despliegue del contrato inteligente en la red simulada por Ganache. Destacar de esta figura dos aspectos claves:

- El primero es que cuando un contrato inteligente se despliega, automáticamente se genera una dirección Ethereum que lo identifica como contrato. Esta dirección puede verse en "*CREATED CONTRACT ADDRESS*".
- El segundo aspecto está relacionado con el campo "*TX DATA*". El contrato inteligente está almacenado en ese campo de esa transacción, que luego va a ser parte de un bloque que será almacenado en la cadena de bloques. Se almacena el ABI[55] (*Application Binary Interface* (ABI)) de dicho contrato, que es una descripción del mismo que se usa para interactuar con él.

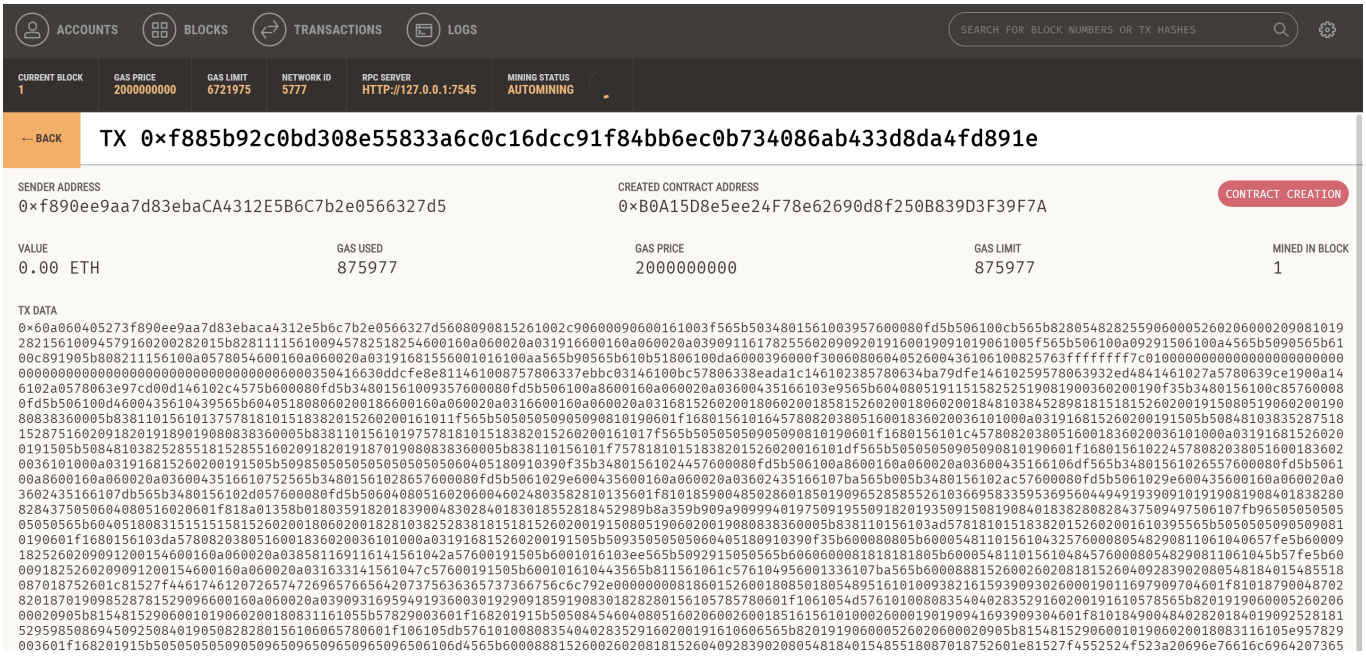

Figura 21: Despliegue del contrato inteligente en la red simulada de Ganache

Una vez desplegado el contrato en la red, ambas partes disponen del ABI. Por lo tanto, se puede hacer una llamada a la función de escritura de datos, codificarla con el ABI, mandar esos datos codificados en el campo de datos de una transacción Ethereum, decodificar esos datos desde el lado de la red con el ABI, ejecutar la función y mandar el resultado al llamante. Se explica este procedimiento de forma gráfica gracias a las figuras 22 y 23. En ellas se muestran las transacciones desde Ganache y el campo de datos decodificado gracias a Remix.

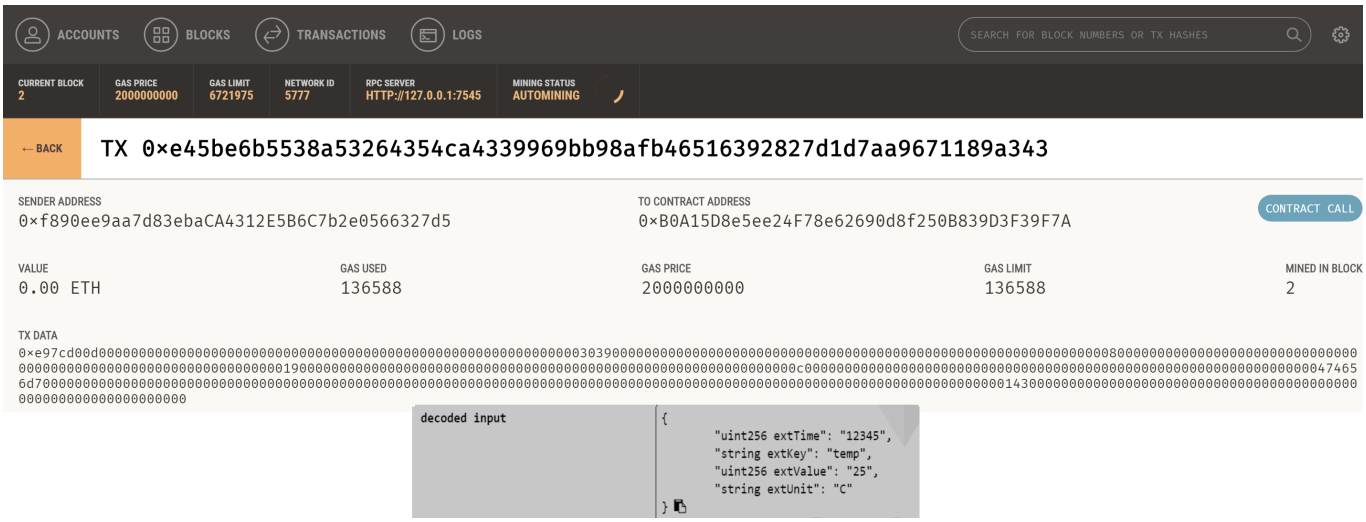

Figura 22: Llamada a la función de escritura del contrato inteligente en la red simulada de Ganache

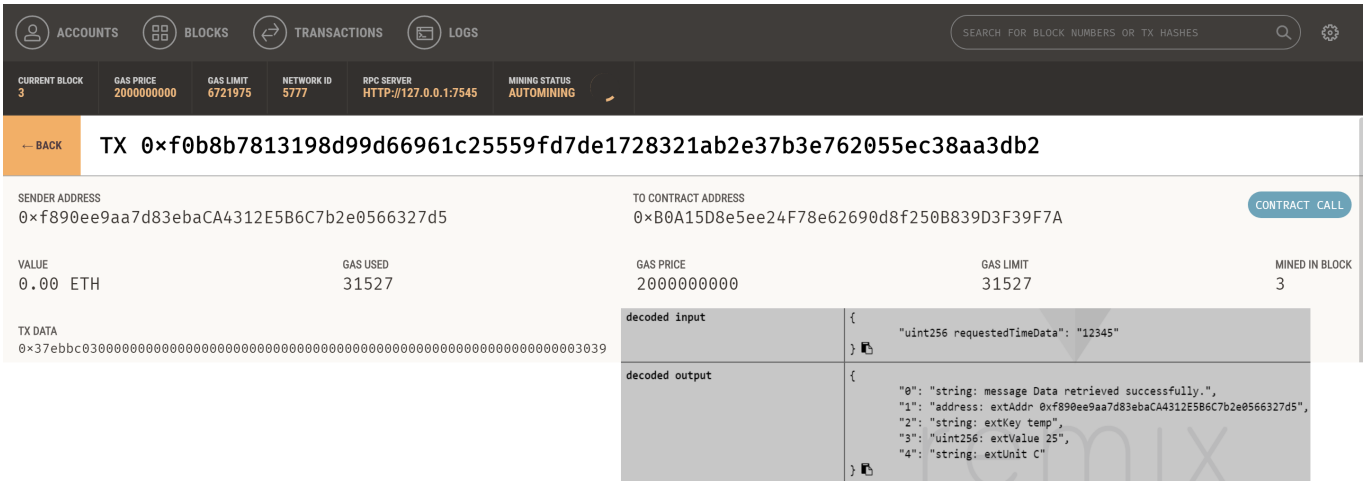

Figura 23: Llamada a la función de lectura del contrato inteligente en la red simulada de Ganache

4.3. Despliegue del contrato inteligente en la red privada de Ethereum

Hasta este punto, lo que se ha conseguido en este apartado de implementación es crear, por una parte, una red Ethereum privada basada en PoA en dos Raspberry Pi que se encargan de verificar y crear bloques, y una tercera que está también conectada a dicha red pero no se encarga de nada más. Por otra parte, se ha desarrollado la aplicación distribuida o el contrato inteligente en el que se describen las funcionalidades que se van a desplegar en la red y se ha comprobado su funcionamiento en una red simulada.

Ahora se va a proceder a hacer este despliegue en la red real de la misma forma que con la red simulada. Este despliegue es menos visual, ya que sólo se muestra que se ha minado el bloque de creación del contrato inteligente en la línea de comandos.

Por último, para programar las interacciones con las funciones del contrato inteligente se pueden usar las múltiples librerías de Web3[56], la más usada en este caso es la de Javascript. En este caso se ha usado la versión para Python (Web3.py) para que los nodos puedan subir sus medidas de forma autónoma.

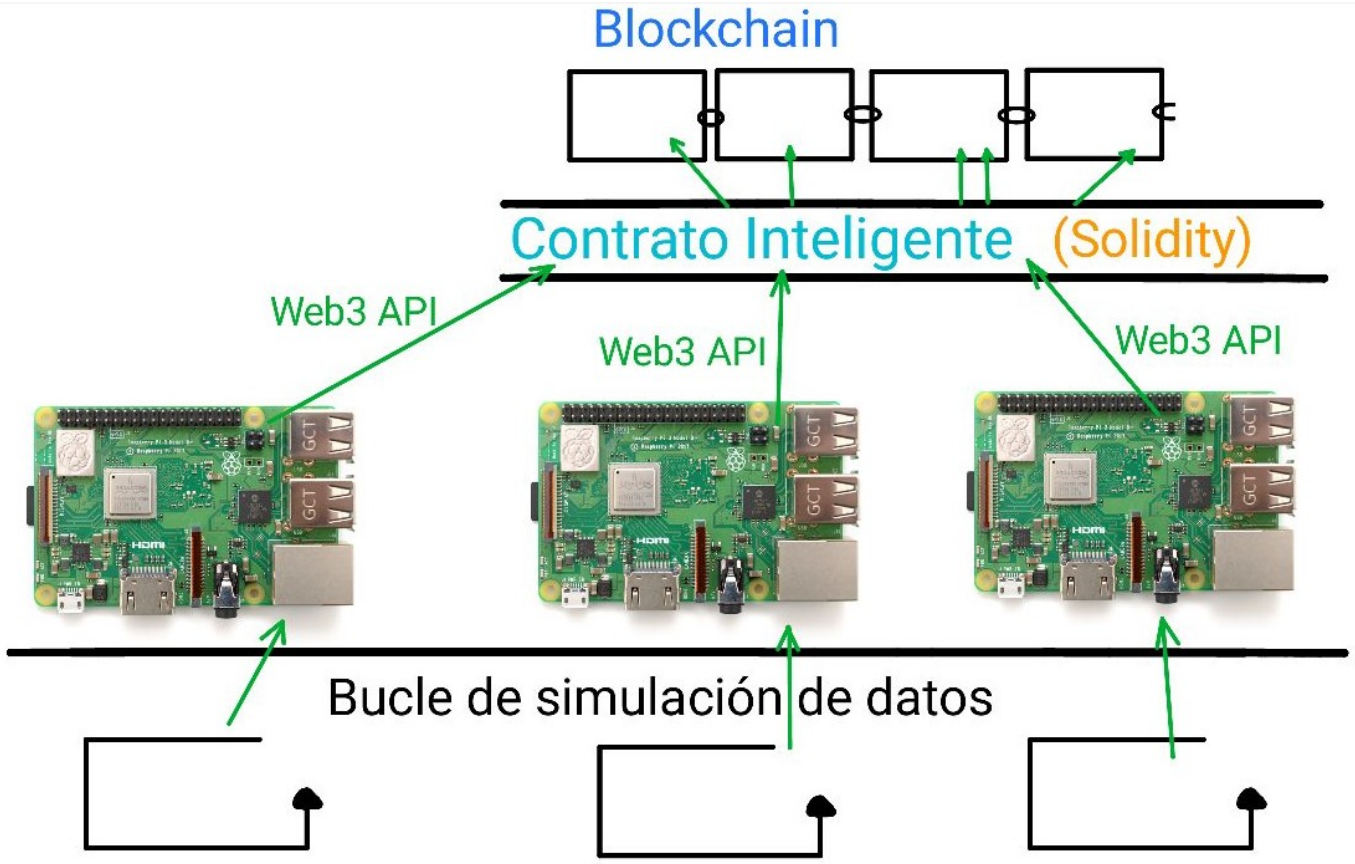

Figura 24: Arquitectura final implementada

En la figura 24, se muestra un esquema de la arquitectura final implementada en este trabajo. Las raspberry Pi generan datos aleatorios creados mediante un bucle implementado en Python que simulan datos generados por la red. Estos datos son subidos mediante el uso de Web3 al contrato inteligente que soportan, en este caso, dos de ellas. El contrato inteligente, mediante las funcionalidades programadas en Solidity, lo almacena en la red de Blockchain.

5. Conclusiones y líneas futuras

Para finalizar este trabajo, se analizarán las conclusiones extraídas y se plantearán las posibles líneas de mejora en una actuación futura.

5.1. Conclusiones

El trabajo planteaba como objetivo principal el desarrollo de una solución que permitiese gestionar de forma descentralizada los datos que son generados por las redes de las ciudades inteligentes. Para lograrlo, se han usado los fundamentos de la tecnología Blockchain, estudiando previamente las variantes más destacadas de ésta. De este estudio, se extrae que si bien la estructura de la tecnología Tangle es la que más ventajas ofrece a priori para el Internet de las Cosas, está centrada en los micropagos y no ofrece facilidades a desarrollos nuevos, ni el despliegue de aplicaciones distribuidas. Este es motivo por el que se elige la plataforma de Ethereum, porque permite la creación de aplicaciones distribuidas mediante el uso de lenguajes de alto nivel.

Seguidamente, se ha profundizado en el estudio de las herramientas de desarrollo creadas alrededor de dicha plataforma, con la que se están creando miles de nuevas aplicaciones descentralizadas.

Una vez adquiridos los conocimientos necesarios se ha procedido al montaje de una primera red privada, en la que se vio el método PoW con la consecuente capacidad de cómputo requerido. En esta situación se hace necesario incluir un ordenador. Posteriormente se modifica para emplear PoA por ser una red privada, reduciendo la necesidad de cómputo del sistema.

Finalizada la creación de la red privada, se desarrolla una aplicación descentralizada mediante un contrato inteligente para proporcionar la inteligencia que dotaría a la red de las funcionalidades de almacenamiento y acceso de datos distribuidos descritas.

Posteriormente, se implementa una red privada simulada en la que probar el funcionamiento de la aplicación descentralizada desarrollada, viendo a su vez de forma gráfica tanto su despliegue en dicha red, como su funcionamiento.

Con todas las partes desarrolladas, se procede, como último paso, a desplegar la aplicación descentralizada en la red privada real. Para simular una red de sensores con los mismo nodos que se usan para crear la red Ethereum privada,tal y como se planteaba e el diseño, se hace uso de librerías específicas que permiten simular datos y subirlos a la red Blockchain.

Al final, se ha conseguido el desarrollo de todo lo que se planteó en un principio mediante una funcionalidad sencilla empleando herramientas de código y hardware abiertas con las que facilitar el desarrollo de futuras implementaciones similares. Se la logrado mediante Ethereum el almacenado de datos descentralizados, se consigue que no haya sólo un punto de entrada, sino que los datos se queden por la propia red accesibles en todo momento. En cierto modo, se valida que el Blockchain puede ser una opción para dar respuesta al Internet de las Cosas en general y para las ciudades

inteligentes en particular.

5.2. Lineas futuras

Aunque a lo largo de este trabajo se ha conseguido almacenar la información que generan 3 sensores en una red blockchain soportada por dos de ellos, no queda muy claro hasta que punto son almacenados la totalidad de esos datos a lo largo del tiempo, ya que la memoria de los nodos es limitada, y se podrían empezar a perder los datos más antiguos.

Esto no es una limitación que tenga nuestro sistema particularmente, sino que es una limitación conocida por la comunidad, y que en la que se está trabajando actualmente. Más aún, Ethereum tiene una red de prueba llamada Swarm[57] que pretende ser una plataforma de almacenamiento distribuido basado en Ethereum, con suficiente redundancia y resistencia a pérdidas que sirva de almacenamiento permanente de datos distribuidos.

Otra de las limitaciones vista durante el desarrollo de este proyecto, es que este tipo de lenguajes orientados e una plataforma específica, dispone de unos tipos de datos limitados. Esto, limita las funcionalidades y el desarrollo. Un ejemplo de esto más concreto, es que, como no se puede hacer uso de bases de datos, para la búsqueda de datos estás limitado a buscar por un valor clave, si no quieres estar obligado a hacer un contrato inteligente para cada tipo de búsqueda.

Este problema se vería solucionado con el uso de una base de datos distribuida montada sobre la red de Ethereum. Un ejemplo de este tipo de bases de datos es OrbitDB[58], que se basa en *Inter-Planetary File System* (IPFS)[59] y que puede ser usado para mejorar este tipo de sistemas. También hay soluciones que externalizan la base de datos y sólo almacenan en la cadena de bloques las referencias, en forma de hash.

Se plantea también, como futuro estudio a seguir, extender el transcurso de la red con más nodos que permitan comprobar su escalabilidad y hacer una evaluación del rendimiento en las búsquedas y ver si se ven afectadas. Además queda espacio para el desarrollo de interfaces gráficas que mejoren la introducción y extracción de datos en la red.

Referencias

[1] IBM, «The client/server model». En línea: https://www.ibm.com/support/knowledgecenter/en/SSAL2T_9.1.0/com.ibm.cics.tx.doc/concepts/c_clnt_sevr_model.html [último acceso: Octubre 2018]

[2] Ben Dickson, «Why does the centralized internet suck?», Octubre 2017. En línea: https://bdtechtalks.com/2017/10/27/why-does-the-centralized-internet-suck/. [Último acceso: Octubre de 2018]

[3] Satoshi Nakamoto, «Bitcoin: A Peer-to-Peer Electronic Cash System», Octubre 2008. En línea: https://bitcoin.org/bitcoin.pdf. [Último acceso: Octubre de 2018]

[4] 99Bitcoins, «Bitcoin Historical Price & Events», En línea: https://99bitcoins.com/price-chart-history/. [Último acceso: Octubre de 2018]

[5] Medium, «2018 - The Year of DApps», Enero 2018. En línea: https://medium.com/the-mission/2018-the-year-of-dapps-dbe108860bcb. [Último acceso: Octubre de 2018]

[6] Josiah Wilmoth, «More Than 1,000 Ethereum DApps Have Launched since 2017», Mayo 2018. En línea: https://www.ccn.com/more-than-1000-ethereum-dapps-have-launched-since-2017/. [Último acceso: Octubre de 2018]

[7] IOT COUNCIL, «www.theinternetofthings.eu», 2016. En línea: www.theinternetofthings.eu. [Último acceso: Octubre 2016].

[8] Duncan McLaren y Julian Agyeman, «Sharing Cities: A Case for Truly Smart and Sustainable Cities», Noviembre 2015. MIT press.

[9] Louis Columbus, «10 Charts That Will Challenge Your Perspective Of IoT's Growth», Junio 2018. En línea: https://www.forbes.com/sites/louiscolumbus/2018/06/06/10-charts-that-will-challenge-your-perspective-of-iots-growth/#25df715b3ecc. [Último acceso: Octubre de 2018]

[10] Petri Basson, «The untold history of Bitcoin: Enter the Cypherpunks», Enero 2018. En línea: https://medium.com/swlh/the-untold-history-of-bitcoin-enter-the-cypherpunks-f764dee962a1. [Último acceso: Octubre de 2018]

[11] Adam Back, «A partial hash collision based postage scheme», Marzo 1997. En línea: http://www.hashcash.org/papers/announce.txt. [Último acceso: Octubre de 2018]

[12] Wei Day, «bmoney», 1998. En línea: http://www.weidai.com/bmoney.txt. [Último acceso: Octubre de 2018]

[13] Rüdiger Schollmeier, «A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications, Proceedings of the First International Conference on Peer-to-Peer Computing», 2002. IEEE.

[14] Bitcoin.com, «What is Bitcoin Double Spending?», Junio 2007. En línea: https://www.bitcoin.com/info/what-is-bitcoin-double-spending. [Último acceso: Octubre de 2018]

[15] Nik Custodio, «Explain Bitcoin Like I'm Five», Diciembre 2013. En línea: https://medium.freecodecamp.org/explain-bitcoin-like-im-five-73b4257ac833. [Último acceso: Octubre de 2018]

[16] Stephen Northcutt, «Hash Functions». En línea: https://www.sans.edu/cyber-research/security-laboratory/article/hash-functions. [Último acceso: Octubre de 2018]

[17] IETF, «US Secure Hash Algorithm 1 (SHA1)», Septiembre 2001. En línea: https://tools.ietf.org/html/rfc3174. [Último acceso: Octubre de 2018]

[18] Bitcoin Wiki, «Proof of work», Mayo 2016, En línea: https://en.bitcoin.it/wiki/Proof_of_work. [Último acceso: Octubre de 2018]

[19] Charles Zaiontz, «Binomial Distribution and Random Walks», 2013-2016. En línea: http://www.real-statistics.com/binomial-and-related-distributions/binomial-distribution-and-random-walks/. [Último acceso: Octubre de 2018]

[20] Vangie Beal, «CPU - Central Processing Unit». En línea: https://www.webopedia.com/TERM/C/CPU.html. [Último acceso: Octubre de 2018]

[21] Adam Shepherd, «What is a GPU?», Junio 2018. En línea: http://www.itpro.co.uk/hardware/30399/what-is-a-gpu. [Último acceso: Octubre de 2018]

[22] XILINX, «What is an FPGA?». En línea: https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html. [Último acceso: Octubre de 2018]

[23] Brian Curran, «What is a Merkle Tree? Beginner's Guide to this Blockchain Component», Julio 2018. En línea: https://blockonomi.com/merkle-tree/. [Último acceso: Octubre de 2018]

[24] Ethereum Fundation, «ethereum, Blockchain App Platform». En línea: https://ethereum.org/. [Último acceso: Octubre de 2018]

[25] Vitalik Buterin, «A Next-Generation Smart Contract and Decentralized Application Platform», septiembre 2017. En línea: https://github.com/ethereum/wiki/wiki/White-Paper. [Último acceso: Octubre de 2018]

[26] CCSU, «Turing Complete». En línea: https://chortle.ccsu.edu/StructuredC/Chap01/struct01_5.html. [Último acceso: Octubre de 2018]

[27] Matthew Tyson, «What is the JVM? Introducing the Java virtual machine», Mayo 2018. En línea: https://www.javaworld.com/article/3272244/core-java/what-is-the-jvm-introducing-the-java-virtual-machine.html. [Último acceso: Octubre de 2018]

[28] Bitcoin and the Double-Spending Problem, en http://blogs.cornell.edu/info4220/2013/03/29/bitcoin-and-the-double-spending-problem/

[29] Explaining blockchain—how proof of work enables trustless consensus, en https://keepingstock.net/explaining-blockchain-how-proof-of-work-enables-trustless-consensus-2abed27f0845

[30] Ethereum revision, «Solidity». En línea: https://solidity.readthedocs.io/en/latest/. [Último acceso: Octubre de 2018]

[31] Mozilla and individual contributors, «What is JavaScript?», 2018. En línea: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript. [Último acceso: Octubre de 2018]

[32] , «Serpent». En línea: https://github.com/ethereum/serpent. [Último acceso: Octubre de 2018]

[33] Python Fundation, «Python Software Foundation», 2001. En línea: https://www.python.org/psf/. [Último acceso: Octubre de 2018]

[34] Daniel Ellison, «An Introduction to LLL for Ethereum Smart Contract Development», Mayo 2017. En línea: https://media.consensys.net/an-introduction-to-lll-for-ethereum-smart-contract-development-e26e38ea6c23. [Último acceso: Octubre de 2018]

[35] Said Eloudrhiri, «Create a private Ethereum blockchain with IoT devices», Febrero 2017. En línea: http://chainskills.com/2017/02/24/create-a-private-ethereum-blockchain-with-iot-devices-16/. [Último acceso: Octubre de 2018]

[36] Salanfe, «Setup your own private Proof-of-Authority Ethereum network with Geth», Febrero 2018. En línea: https://hackernoon.com/setup-your-own-private-proof-of-authority-ethereum-network-with-geth-9a0a3750cda8. [Último acceso: Octubre de 2018]

[37] R. Srinivasan, «RPC: Remote Procedure Call Protocol Specification Version 2», Agosto 1995. En línea: https://tools.ietf.org/html/rfc1831. [Último acceso: Octubre de 2018]

[38] JSON-RPC Working Group, «JSON-RPC 2.0 Specification», Marzo 2010. En línea: https://www.jsonrpc.org/specification. [Último acceso: Octubre de 2018]

[39] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, «Hypertext Transfer Protocol – HTTP/1.1», Junio 1999. En línea: https://tools.ietf.org/html/rfc2616. [Último acceso: Octubre de 2018]

[40] D. C. Walden, «A System for Interprocess Communication in a Resource Sharing Computer Network», Agosto 1970. En línea: https://tools.ietf.org/html/rfc62. [Último acceso: Octubre de 2018]

[41] Broadcom, «APPLICATION-SPECIFIC INTEGRATED CIRCUITS (ASICS)». En línea: https://www.broadcom.com/products/custom-silicon/asics/. [Último acceso: Octubre de 2018]

[42] Andy Oram, John Viega, «Beautiful Security», Abril 2009.

[43] Ethereum.org, «What is Proof of Stake», 2018. En línea: https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQs#what-is-proof-of-stake. [Último acceso: Octubre de 2018]

[44] Decred.org, «Decred Documentation», 2018. En línea: https://docs.decred.org/research/overview/. [Último acceso: Octubre de 2018]

[45] Hans Knutson, «What is the math behind elliptic curve cryptography?», Abril 2018. En línea: https://hackernoon.com/what-is-the-math-behind-elliptic-curve-cryptography-f61b25253da3. [Último acceso: Octubre de 2018]

[46] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, Raphael C.W. Phan, «SHA-3 proposal BLAKE», Diciembre 2010. En línea: https://131002.net/blake/blake.pdf. [Último acceso: Octubre de 2018]

[47] Serguei Popov, «The Tangle», Abril 2018. En línea: https://iota.org/IOTA_Whitepaper.pdf. [Último acceso: Octubre de 2018]

[48] Hamza Surti, «Advanced Data Structures Part 1: Directed Acyclic Graph (DAG)», Abril 2016. En línea: https://medium.com/@hamzasurti/advanced-data-structures-part-1-directed-acyclic-graph-dag-c1d1145b5e5a. [Último acceso: Octubre de 2018]

[49] Felix Lange, «Geth», Diciembre 2017. En línea: https://github.com/ethereum/go-ethereum/wiki/geth. [Último acceso: Octubre de 2018]

[50] ARM University Program Resources, «The ARM Architecture». En línea: https://www.arm.com/files/pdf/ARM_Arch_A8.pdf. [Último acceso: Octubre de 2018]

[51] ethereum.org, «Remix - Solidity IDE». En línea: http://remix.ethereum.org/. [Último acceso: Octubre de 2018]

[52] Francis Boily, «Explaining Ethereum Test Networks And All Their Differences», Mayo 2018. En línea: https://medium.com/coinmonks/ethereum-test-networks-69a5463789be. [Último acceso: Octubre de 2018]

[53] CONSENSYS 2018, «TRUFFLE, SMART CONTRACTS MADE SWEETER». En línea: https://truffleframework.com/truffle. [Último acceso: Octubre de 2018]

[54] CONSENSYS 2018, «Ganache, ONE CLICK BLOCKCHAIN», Mayo 2018. En línea: https://truffleframework.com/ganache. [Último acceso: Octubre de 2018]

[55] Solidity, «Contract ABI Specification», 2016-2018. En línea: https://solidity.readthedocs.io/en/develop/abi-spec.html. [Último acceso: Octubre de 2018]

[56] Ethereum, «JavaScript API», 2018. En línea: https://github.com/ethereum/wiki/wiki/JavaScript-API. [Último acceso: Octubre de 2018]

[57] Ethereum, «SWARM», 2018. En línea: https://swarm-guide.readthedocs.io/en/latest/introduction.html. [Último acceso: Octubre de 2018]

[58] Protocol Labs Inc., «Peer-to-Peer Databases for the Decentralized Web», 2018. En línea: https://github.com/orbitdb/orbit-db. [Último acceso: Octubre de 2018]

[59] Michal Zalecki, «Using IPFS with Ethereum for Data Storage», Julio 2018. En línea: https://www.tooploox.com/blog/using-ipfs-with-ethereum-for-data-storage. [Último acceso: Octubre de 2018]

Siglas

ABI *Application Binary Interface*

API *Application Programming Interface*

ASIC *Aplication-Specific Integrated Circuit*

CPU *Central processing Unit*

DAG *Directed Acyclic Gragh*

DApps *Decentralized Applications*

EOA *Externally Owned Accounts*

EVM *Ethereum Virtual Machine*

FPGA *Field-programmable gate array*

GPU *Graphics Processing Unit*

HTTP *HyperText Transfer Protoco*

IoT *Internet of Things*

IPC *Inter-Process Communication*

IPFS *Inter-Planetary File System*

JVM *Java Virtual Machine*

LLL *Lisp Like Language*

M2M *Machine to Machine*

PGP *Pretty Good Privacy*

RPC *Remote Procedure Call*

SHA *Secure Hash Algorithm*