ethereum

Fundamentos de Redes

Memoria de la exposición

Pedro Bonilla Nadal
Ana Peña Arnedo

28 de noviembre de 2017

1. Introducción.

Hoy en día, la información lo es todo. Las grandes compañías de internet han alcanzado un modelo de negocio con el que no necesitan exigirte dinero para conseguir beneficios. Realmente, éstos te cobran por una aceptación de ceder tu información con el fin de traficar con ella.

En la actualidad, existen nuevas propuestas cuyo objetivo es el de establecer un sistema de comercio que no necesite ser mantenido o controlado por un organismo, sino que, por lo tanto, establezca poder sobre sus usuarios. Este tipo de comercio 'tradicional' engloba organizaciones como, por ejemplo, amazon, que facilita la compra-venta respaldada por una compañía, o los bancos centrales, los cuales conservan las divisas y son los responsables del cuidado y mantención de las mismas. Como sabemos, el hecho de que una compañia esté a cargo de cuidar el sistema (es decir, que tenga un sistema centralizado) tiene una serie de beneficios e inconvenientes.

Por un lado, el sistema centralizado controlado por una compañía que desempeñe funciones como la contratación de especialistas para almacenar y proteger la seguridad reduce los tiempos de almacenamiento y actualización del sistema.

Por otro lado, donde hay una ventaja, hay un inconveniente. Como se ha podido ver en alguna ocasión[1], esta organización permite a grandes compañías y estados filtrar, robar o modificar la información sensible de los usuarios. Esta vulnerabilidad ha llegado a ser calificada como el 'pecado original' de internet, pues éste siempre intentó ser una plataforma de carácter descentralizado.

Satoshi Nakamoto inció en 2008 el desarrollo de bitcoin. Este hecho desencadenaría un progreso increíble en el área de las divisas descentralizadas, es decir, el desarrollo de un sistema monetario que no necesitase de un sistema bancario o de un valor predeterminado. También resultó muy sorprendente la aplicación de la tecnología de la cadena de bloques ('blockchain') como una herramienta basada en la distribución del consenso. A raíz de estas, surgieron otras teconologías basadas en la cadena de bloques, como namecoin. Lo que ethereum pretende es proveer una cadena de bloques con un leguanje turing completo integrado que pueda ser usado para el desarrollo de contratos inteligentes, permitiendo así a los usarios la crear proyectos solo escribiendo la lógica de los mismos en unas pocas líneas de código. Algunos de estos proyectos, sacados del la propia web `ethereum.org`, serían la creación de un crowdfounding que no se base en la confianza, si no en un contrato que almacene el dinero de los contribuyentes, una organización autónoma democrática.

En resumen, nuestra moneda lo que intenta es crear un 'ordenador global' que descentralice el actual cliente-servidor. Con Ethereum, en lugar de tener un servidor tendríamos un conjunto de nodos almacenados por voluntarios alrededor de todo el mundo. Sobre este sistema, la comunidad podrá competir por ofrecer servicios, además de consumirlos.

[1] `https://www.theguardian.com/world/2013/jun/06/nsa-phone-records-verizon-court-order`

Ejemplifiquemos la diferencia: navegar por una app-store cualquiera nos ofrecerá una serie de aplicaciones en las cuales contactaremos con un servidor que nos proveerá del servicio, generalmente gestionado por un tercero no relacionado (de manera directa) con la transacción.

Ethereum, si todo funciona, devolvería la propiedad de la información a su dueño, de modo que solo el usuario puede modificar la información, y no ninguna entidad externa.

2. Ethereum.

2.1. Historia.

En 2012, un joven de 19 años propuso una nueva plataforma con el objetivo de trasnfomar por completo internet. Vitalik Buterin, un programador de Toronto, empezó su investigación en la cadena de bloques en 2011. Co-fundó el portal web Bitcoin Magazine y trabajó para compañías de la materia. En el camino, pensó en una plataforma que fuera más allá de las posibilidades del bitcoin.

Con esta idea nació ethereum, una plataforma para la creación de contratos inteligentes ('smart contract'). Después de publicar en 2014 el white paper, otros desarrolladores se unieron al proyecto. Para lanzar el proyecto se inició un crowdfounding en julio de 2014 donde los participantes compraban **ether** . Una vez hubieron reunido más de 18 millones de dólares, se inició y en 2015 se lanzó una plataforma, no demasiado user-friendly, pero con comandos en línea que permitía la creación de aplicaciones descentralizadas.

Este nuevo tipo de contratos caló entre el público llamando la atención de gigantes tecnológicos como IBM y gran cantidad de desarrolladores. El dinero recaudado inicialmente está gestionado por Etherum foundaion[2], una compañía sin ánimo de lucro ubicada en Suiza.

2.1.1. Hardfork y el cisma de ethereum.

En 2016 una organización autónoma descentralizada llamada The DAO, con un conjunto de contratos inteligentes, reunieron un total de USD $150 millones en una crowdsale. Al final DAO explotó cuando, en junio, USD $50 millones en Ether fueron reclamados de manera anónima. El suceso inició un debate sobre si de debía hacer un **hardfork** y, como resutado de esta disputa, la red se dividió en dos: Ethereum, el objetivo de este trabajo, que continuó la cadena modificada, y Ethereum Classic, que continuó la cadena original. Desde entonces se generó un conflicto entre ambas comunidades.

Figura 1:

ethereum classic

[2] https://www.ethereum.org/foundation

2.2. Funcionamiento.

Una vez sabemos lo que es ethereum, profundizamos en el funcionamiento de la plataforma. Al usar ethereum, la 'app' no requiere ninguna entidad para almacenar y controlar sus datos. Para conseguirlo, ethereum hace uso del protocolo de bitcoin y su diseño de la cadena de bloques, aunque lo ajusta de manera que puede respaldar aplicaciones, además del dinero. Sin embargo, ethereum persigue abstraerse del diseño de bitcoin con el fin de que los desaprobadores puedan crear aplicaciones o acuerdos que incluyan medidas adicionales, nuevas normas de propiedad, formatos alternativos de transacciones o diferentes maneras de transferir estados.

El objetivo del lenguaje de programación 'Turing-completo' de ethereum es permitir a los desarrolladores escribir más programas (que los que permite la línea de comandos de bitcoin) en los que las transacciones en la cadena de bloques puedan gobernar y automatizar resultados específicos. Tengamos en cuenta que un lenguaje turing-completo es capaz de generar, en teoría, los mismos programas que son desarrollables C++. Esta flexibilidad que ofrece ethereum es la innovación fundamental de ethereum sobre otras plataformas basadas en el protocolo de bitcoin.

2.2.1. Tecnología como sistema de transición de estados.

Desde un punto de vista técnico, el 'ledger' de los sitemas de cadena de bloques puede verse como un sistema de transición de estados, donde hay un estado, que consiste en el estado de propiedad de todos los bitcoins existentes, además de una función de transición de estado que toma un estado y una transacción, generando un nuevo estado que será el resultado. De este modo obtendríamos un autómata, y además, al haber un número finitio de bitcoin, podemos decir que es finito en teoría (aunque no en práctica, pues la cantidad de variaciones de la red es casi inabarcable).

En un sistema bancario estándar, por ejemplo, el estado es una hoja de balance, una transacción es una petición para mover una determinada cantidad de dinero de 'A' a 'B', y la función de transición de estado reduce esa cantidad de dinero en la cuenta de 'A' y la incrementa en la cuenta de 'B'. Si, en primer lugar, la cuenta de 'A' tiene una cantidad menor a la indicada, la función de transición de estado devuelve un error. Por lo tanto, podría definirse formalmente:

```
APPLY( S , TX ) -> S' o ERROR
```

En el sistema bancario definido arriba:

```
APPLY( { Ana: 50 , Pedro: 50 } , "enviar 20 de Ana a Pedro" )
{ Ana: 30 , Pedro: 70 }
```

Sin embargo:

```
APPLY( { Ana: 50 , Pedro: 50 } , "enviar 70 de Ana a Pedro" )
ERROR
```

El estado en Bitcoin es la colección de todas las monedas (técnicamente, "transacciones no gastadas" o UTXO - "unspent transaction outputs") que han sido creadas pero aún no se han gastado, con cada UTXO que tiene una denominación y un propietario (definido por una dirección de 20 bytes que es, esencialmente, una clave pública criptográfica). Una transacción contiene una o más entradas, y cada entrada contiene una referencia a una UTXO existente y a una firma criptográfica producida por la clave privada asociada con la dirección del propietario;

y una o más salidas, cada una de las cuales contiene una nueva UTXO para añadirla al estado.La función de transición de estado APPLY(S , TX) − >S' puede definirse aproximadamente como sigue:

1. Para cada entrada en TX:
    i. Si la UTXO referenciada no está en S, devuelve un error.
    ii. Si la firma proporcionada no concuerda con el propietario de la UTXO, devuelve un error.
2. Si el conjunto de denominaciones de toda entrada UTXO es menor que el conjunto de denominaciones de toda salida UTXO, devuelve un error.
3. Devuelve S con toda entrada UTXO eliminada y toda salida UTXO añadida.

La primera mitad del primer paso evita que los que envían transacciones gasten monedas que no existen, la segunda mitad del primer paso evita que los que envían transacciones gasten las monedas de otras personas, y el segundo paso impone la conservación del valor. A la hora de usar esto para pagos, el protocolo es como sigue. Supongamos que Ana quiere enviar 11.7 BTC a Pedro. Primero, Ana buscará un conjunto de UTXO disponibles que ella posea que sume un total de al menos 11.7 BTC. Realmente, Ana no será capaz de obtener exactamente 11.7 BTC; digamos que lo menos que puede obtener es 6+4+2=12. Entonces ella creará una transacción con esas tres entradas y dos salidas. La primera salida será 11.7 BTC con la dirección de Pedro como su propietario, y la segunda salida serán los restantes 0.3 BTC de "cambio", siendo su propietaria la propia Ana.

2.2.2. Cadena de Bloques de Ethereum.

Una cadena de bloques es una base de datos distribuida, formada por cadenas de bloques diseñadas para evitar su modificación una vez que un dato ha sido publicado usando un sistema basado en el tiempo y el consenso de los usuarios de ésta. Cada bloque está enlazando a un bloque anterior. Por esta razón, es especialmente adecuada para almacenar de forma creciente datos ordenados en el tiempo y sin posibilidad de modificación ni revisión. Este enfoque tiene diferentes aspectos:

- Almacenamiento de datos.- Se logra mediante la replicación de la información de la cadena de bloques
- Transmisión de datos.- Se logra mediante peer-to-peer
- Confirmación de datos.- Se logra mediante un proceso de consenso entre los nodos participantes. El tipo de algoritmo más utilizado es el de prueba de trabajo, 'proof-of-work', en el que hay un proceso abierto competitivo y transparente de validación de las nuevas entradas llamado minería.

Los datos almacenados en la cadena de bloques normalmente suelen ser transacciones, por lo que es frecuente llamar a los datos transacciones. Sin embargo, no es necesario que lo sean. Realmente podríamos considerar que lo que se registran son cambios atómicos del estado del sistema. Por ejemplo, una cadena de bloques puede ser usada para estampillar documentos y segurizarlos frente a alteraciones.

La gran diferencia que ofrece ethereum es que sus nodos almacenan también el estado más reciente de cada contrato inteligente, además de todas las transacciones de ether. Para cada aplicación de ethereum, la red tiene que mantener un seguimiento del 'estado' o información

actual de todas estas aplicaciones, incluyendo el saldo de cada usuario, el código completo del contrato inteligente y dónde se almacena todo. Bitcoin usa salidas de transacción no utilizadas para rastrear quién tiene cuánto bitcoin. Aunque suena complejo, la idea es bastante simple. Cada vez que se hace una transacción de bitcoin, la red 'rompe' la cantidad total como si fuera dinero impreso, emitiendo bitcoins de vuelta de una forma que hace que la información manejada se comporte como las monedas o cambio físico.

Para efectuar futuras transacciones, la red de bitcoin tiene que añadir todas las piezas de cambio, que se clasifican en 'gastadas' o 'no gastadas'. Ethereum, por otro lado, utiliza cuentas. Como en fondos de cuentas bancarias, las 'fichas' de ether aparecen en una cartera, y pueden ser transferidos a otra cuenta. Los fondos siempre están en algún sitio, pero no tienen lo que podría llamarse una relación continua.

2.2.3. Cuentas.

En Ethereum, el estado está constituido por objetos llamados cuentas, con cada cuenta formada por direcciones de 20 bytes y transiciones entre estados, siendo la transferencia directa de valor e información entre cuentas. Cada cuenta puede contener hasta 4 campos:

- Nonce: es utilizado para asegurarse de que cada transacción solo se puede hacer una vez.
- Valor: cantidad de ether que posee la cuenta.
- Código de contrato: código que especificfa el funcionamiento de una cuenta con contrato. No es obligada su existencia.
- Almacenamiento: vacío por defecto, contiene información, relacionada o no, con la cuenta.

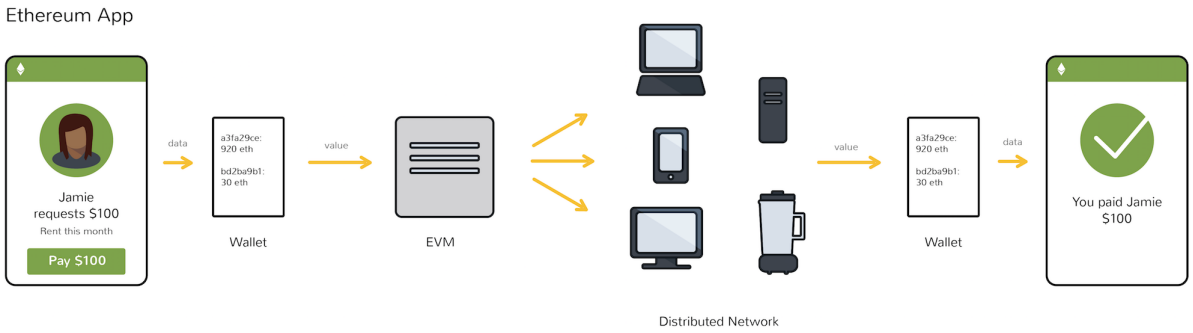
Hay dos tipos de cuentas, cuentas propias externas (externally owned account), controladas por claves privadas, y cuentas con contrato, controladas por el código del contrato. En una cuenta externa no hay código. En una cuenta dirigida, cada vez que hay un mensaje, el código de la cuenta se ejecuta con derecho a reescribir la misma.

2.2.4. Máquina Virtual de Ethereum

Con ethereum, cada vez que se usa un programa, una red de miles de computadores lo procesa. Los contratos escritos en un lenguaje de programación específico de contrato inteligente se compilan en 'bytecode', lo que una prestación llamada 'ethereum virtual machine' (EVM) puede leer y ejecutar. Todos los nodos ejecutan este contrato usando sus EVMs. Cada vez que un usuario realiza alguna acción, todos los nodos de la red tienen que estar de acuerdo en que ese cambio se ha efectuado. El objetivo aquí es que la red de mineros y nodos tomen la responsabilidad de transferir el cambio de estado a estado, en lugar de cualquier autoridad como PayPal o un banco. Los mineros de bitcoin validan el cambio de propiedad de bitcoins de una persona a otra. La EVM ejecuta un contrato con las reglas que el desarrollador programó inicialmente.

El cálculo real en la EVM se consigue mediante un lenguaje 'bytecode' basado en pilas, pero los desarrolladores pueden escribir contratos inteligentes en lenguajes de alto nivel como Solidity o Serpent, más fáciles de leer y escribir para las personas, dado que bytecode es un lenguaje poco legible, pero más interpretable a un ordenador, análogo al lenguaje máquina. Los mineros son los que evitan un mal comportamiento. A diferencia de en bitcoin, la cantidad de nodos es escalable, por lo que la ausencia de mineros no será un problema a largo plazo. Por ejemplo, deben asegurarse de que nadie gasta dinero más de una vez, y rechazar contratos inteligentes que no se han pagado. Existen varios miles de nodos de ethereum ahí fuera, y cada uno de ellos

está compilando y ejecutando el mismo código.



Podemos pensar que todo esto tiene un coste mucho mayor al de cálculos ordinarios, lo cual es cierto. Por ello la red solo debe usarse para casos de uso particular. El tutorial oficial de desarrollo de ethereum reconoce esta ineficacia, declarando: "A grandes rasgos, una buena heurística para usar es que no podrás hacer nada en la EVM que no puedas hacer en un teléfono inteligente de 1999".

Este lenguaje será fundamental para la implementación de los contratos inteligentes o samrt contract, pues es lo que permite generar los entes de funcionamiento autonómo para el desarrollo de estos.

2.2.5. Mensajes, transacciones y estado de transición de ethereum

Los mensajes en ethereum son parecidos, en cierto modo, a las transacciones en otros sistemas de cadena de bloques, aunque con algunas características llamativas. Un mensaje puede ser creado tanto por una entidad externa como por un contrato, los mensajes pueden contener datos o, si el mensaje es recibido por un contrato, éste tiene la opción de responder. Esto implica que un mensaje en etherum puede tomar el aspecto de función.

El termino transacción es usado en ethereum para referirse a un paquete firmado que contiene un mensaje para ser enviado por una cuenta externa. Las transacciones serían el recipiente del mensaje, una firma identificando el remitente, la cantidad de ether que se pretende mandar, y además, los dos valores importantes de Gasprice y Startgas.

Para prevenir el uso de bucles infinitos, cada transacción requiere marcar cuántos pasos de computación pueden ser realizados. Startgas es el límite, y Gasprice es la cuota a pagar al minero por iteración. Si la transacción se queda sin "gas", todos los cambios de estado en la cadena se revierten. Si llega a un halts con algun gas restante, éste se le envía al contratante. El precio a pagar por esto se paga en **ether**.

Como veremos, es importante que una transacción tenga los mismos derechos que un usuario para el desarrollo conveniente de los contratos inteligentes, incluyendo la habilidad de mandar mensajes de unos a otros.

2.2.6. Contratos inteligentes.

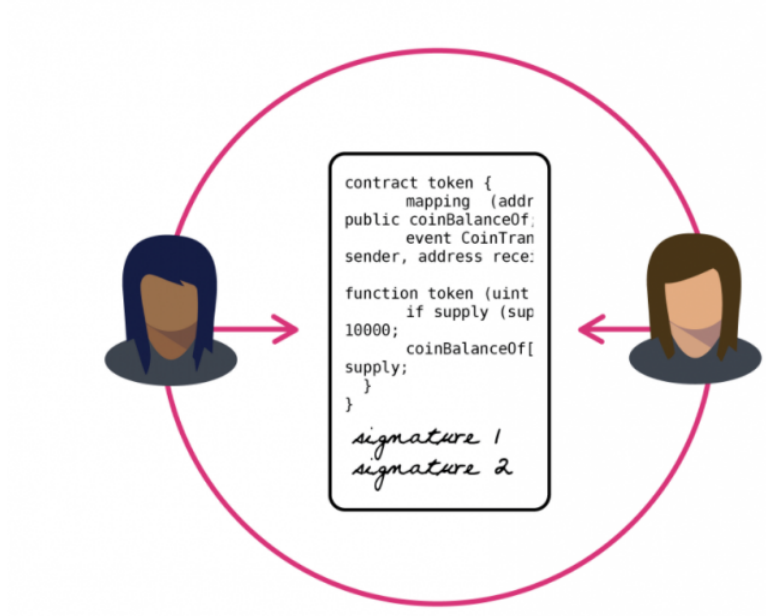
Un contrato inteligente (en inglés Smart contract) es un programa informático que facilita, asegura, hace cumplir y ejecuta acuerdos registrados entre dos o más partes (por ejemplo, personas

u organizaciones). Como tales, ellos les ayudarían en la negociación y definición de tales acuerdos que causarán que ciertas acciones sucedan como resultado de que se cumplan una serie de condiciones específicas.

Un contrato inteligente es un programa que vive en un sistema no controlado por ninguna de las partes, o sus agentes, y que ejecuta un contrato automático el cual funciona como una sentencia 'if' de cualquier otro programa de ordenador. La diferencia es que se realiza de una manera que interactúa con activos reales. Cuando se dispara una condición pre-programada, no sujeta a ningún tipo de valoración humana, el contrato inteligente ejecuta la cláusula contractual correspondiente.

Tienen como objetivo brindar una seguridad superior a la ley de contrato tradicional y reducir costos de transacción asociados a la contratación. La transferencia de valor digital mediante un sistema que no requiere confianza abre la puerta a nuevas aplicaciones que pueden hacer uso de los contratos inteligentes, como es el caso de ethereum.

Bitcoin inició este proceso antes, pero su uso está limitado al uso únicamente como divisa. Sin embargo, el lenguaje turing-completo de ethereum que permite a los desarrolladores escribir sus propios programas, soporta que estos desarrollen sus propios contratos inteligentes, es decir, agentes que que controlen todo el proceso del negocio.



Estos agentes, no controlados por ninguna de las partes, son los que, una vez aceptado el contrato, fuerzan su cumplimiento o devuelven error una vez el contrato se ha terminado de forma fallida al estado original la cadena de bloques.

Un punto importante a considerar como fallo de los contratos inteligentes es que ambas partes han de tener acceso al código del smart contract y, por tanto, comprender activamente (no porque se lo haya explicado alguien) qué hace y por qué lo hace el programa brindado. Si estas condiciones se cumplen y ambas partes saben lo que están aceptando, su cumplimiento es obligado, de modo que si no se cumpliera el contrato no tendría ningún efecto.

Los contratos inteligente pueden, entre otras muchas cosas:

- Sevir como un programa de firma de comunidad, en el cual los fondos se invierten solo cuando hay una mayoría prefijada de personas que aceptan la inversión. Del mismo modo este contrato podría ser reformado en el código que lo constituye cuando la misma mayoría (por ejemplo, del 80 %) lo acpeta.
- Controlar acuerdos entre ususarios, por ejemplo, cuando uno contrata un seguro con el otro.
- Gestionar el funcionamiento de otros contratos (recordemos que, en ethereum, un contrato puede realizar las mismas acciones que un usuario).

Pongamos un ejemplo de este último:

Cuando, en una compañía, una mayoría acepte bajar la temperatura del AC, otro contrato comprobaría la previsión meteorológica para ver si esta acción debe tener carácter permanente, y otro contactaría con la base de datos para enviar la información recogida en los anteriores a la base de datos. El precio de la transacción depende de la energía que cueste esta transacción.

2.2.7. Minería y Prueba de Trabajo.

La minería juega un papel importante en asegurarse de cómo funciona ethereum, pero de forma subliminal. Además, se encarga del objetivo de generar nuevos ether sin la necesidad de un banco, aunque este no es su único rol. Normalmente, es la compañía que centraliza las operaciones (como un banco) la que se asegura de mantener resgistros adecuados de los datos. Sin embargo, los sistemas de cadena de bloques introducen un nuevo modo de mantener el registro de acciones, cuando es la red entera la que se asegura de mantener el registro de transacciones, en lugar de un intermediario, y las anota en un registro púbilco.

Aunque un sistema que no requiera de confianza, o que requiera de la más mínima, es el objetivo principal, alguien tiene que asegurarse de que nadie hace trampa. La minería es lo que permite realizar este registro descentralizado.

Los mineros llegan al consenso sobre las historias de las transacciones a la vez de evitar posibles fraudes, como el doble gasto de ethers; un problema que no había sido resuelto sin las cadenas de bloque con pruebas de trabajo.

Recordemos que la prueba de trabajo conlleva requerir, para realizar una acción, de un coste extra de computación. Las primeras aplicaciones de estos sistemas eran para evitar comportamientos indeseables, como ataques DDoS o envío de spam. Si consigues que estas acciones malintencionadas cuesten un trabajo al atacante, desincentivas el ataque (al menos parcialmente).

Pero este trabajo tiene que tener una característica clave: asimetría. Debe ser difícil llevarlo a cabo (aunque factible), mientras que verificar que es real tiene que tener un coste casi nulo, para que sea posible comprobar la correción de muchos clientes con relativa facilidad.

Aunque a día de hoy se sigue buscando otros métodos de consenso sobre el que validar transacciones, la minería es lo que actualmente mentiene la credibilidad del sistema. El sistema de minería utiliza la fuerza de computacíon para ir resolviendo por fuerza bruta una prueba hasta que una de ellas acierte por casualidad. Más específicamente, los mineros ejecutarán los metadatos de la cabecera en un hash, solo cambiando un valor. Si el minero encuentra el hash con el valor que coincide con el del actual objetivo, el minero será recompensado con ether y extenderá a través de la red la propiedad de ese bloque. Cuando el minero 'B' ve que el minero

'A' ha encontrado el bloque que él minaba, éste pasa a minar el siguiente.

Es dificil hacer más rápido este proceso, mientras que es muy rápido comprobar que el hash ofrecido por el minero es correcto. Por eso, este proceso nos cumple las condiciones para ser una prueba de trabajo.

Un minero encuentra un bloque cada 12-15 segundos. Si los mineros empezasen a encontrar ether cada menos o más teimpo , el algoritmo se reajustaría para que la velocidad de la inflacción se mantenga constante.

Las ganancias de los mineros dependen tanto de la suerte como de su fuerza para computar datos. El algoritmo en especifico usado se llama ethash.

Se está intentando cambiar a un sistema donde no se dependa tanto de los mineros, cambiando la 'proof-of-work' por la 'proof-of-stake'. textcolorredbuscar proff-of-stake e incluir

2.2.8. Hardfork.

Pese a que al debate generado en bitcoin sobre la **esclabilidad** ethereum está organizado apra que este no sea un problema relevante. En esta plataforma los incidente que han hecho que estos fork o bifurcaciones hayan tomado un papel más importante son de otro caracter como el incidente de la **DAO**.

Básicamente, el término fork hace referencia al despliegue de cambios en el código de la cadena de bloques. La bifurcaión sucede cuando el equipo detras de ethreum quiere implantr cambios en la estructura por diversos motivos. Como la cadena de bloques es una estructura de datos descentralizada, hay diferentes, esto provoca que estas situcaciones generen cadenas de bloques alternativas. Es lo que se conoce como bifurcación de la Blockchain, y aquí es donde entran en juego los conceptos de hardfork y softfork.

Es una divergencia permanente en la Blockchain que ocurre cuando los nodos no actualizados no pueden validar bloques creados por los nodos que se apegan a las más recientes reglas de consenso. Actualmente hay una división filosófica en la comunidad de la cadena de bloques, entre las concepciones de lo que el activo en realidad es y aquello que debería ser. Esta discusión nace porque una comunidad con fin a ser descentraliza, no debe permitir que una compañía modifique la forma de la cadena de bloques porque ella así lo consideré pues esto solo implicaría que se está cambiando de manos la centralización, si bien en algunos casos ocurridos en el pasado, no hubo otra opción que se pudiera bajarar como solución a los conflictos. En ethereum han sucedido varios hardfork, tres en el año 2016. Ethereum se convirtió en dos cadenas de bloques, Ethereum y Ethereum Classic, en un intento de devolver a los inversores fondos desviados del desastre de DAO.

2.2.9. Softfork.

A diferencia del 'hard fork', el 'soft work' introduce código que sí es compatible con las versiones más antiguas del programa, por lo que no es posible que existan disrupciones ya que los usuarios sólo tienen que actualizar su software para empezar a beneficiarse de las nuevas funciones. De este modo una bifurcación suave es menos conflictiva que una dura. Tan pronto se implementa una bifurcación suave, los nodos actualizados intentan alcanzar una clara mayoría en la red y recordarnos que un porcentaje de la misma está determinado por el tamaño del bloque; de

ocurrir lo contrario, el 'soft fork' falla y la cadena original sigue intacta. Así, de conseguir el cambio, se consigue por consenso. Esto, de todos modos lleva a la situación en la cual es posible que no se consiga una implatación del nuevoe estandar, por lo cual se discute si realmente estos son preferibles a los hardfork.

3. Aplicaciones.

Para esta sección, vamos a seguir el white-paper fundacional de la criptodivisa, y a explicar las diferentes aplicaciones, entre otras muchas, que se planeaban conseguir e¡con ethereum.

En general, hay tres tipo de aplicaciones montadas encima de ethereum. La primera categoría es aplicaciones de financiación, las cuales habilitan una vía más amplia de gestionar los contratos. La segunda categoría son aplicaciones semifinancieras,donde el dinero está involucrado pero hay una gran parte de importancia en valores no monetarios. Por último estarían las plataformas para organizaciones de gobieron descentralizado y voto, no monetarias en ningún sentido.

3.1. Sistemas de Token.

Los sistemas de tokens basados en cadena de bloques tienen multitud de aplicaciones, desde representar alguna divisa o materia relacionada con el mundo real como el precio del venta de oro o intercambio de USD. Otros usos serían tokens en relación a conceptos como propiedad intelectual o sistemas no relacionados en ningún punto, no financieros en ningún aspecto.

Los sistemas de token son muy fáciles de implementar con ethereum. El punto clave para entender el funcionamiento es darse cuenta de que cualquier sistema de token es, fundamentalmente, una base de datos con una operación: Quita x unidades a A y dáselas a B, con condición de que:

- A tiene al mmenos X unidades antes de la transacción.
- La transacción está aprobada por A.

Todo lo restante sería implementar esta lógica en un contrato. Un código básico sería:

Algorithm 1 Contrato de Tokens.

```
1:  procedure TOKEN(msg, contract)        ▷ msg contiene la información de la comunicación
2:                                              ▷ contract contiene información del contrato
3:      from = msg.sender
4:      to = msg.data[0]
5:      value = msg.data[1]
6:      if contract.storage[from] >= value then
7:          contract.storage[from] = contract.storage[from]  value
8:          contract.storage[to] = contract.storage[to] + value
9:      end if
10: end procedure
```

Esto sería una implementación en pseudo-código de la lógica básica del sistema bancario ya descrito.

3.2. Sistema de identidad

Namecoin ya intentó montar sobre la cadena de bloques un sistema de registro de nombres, donde los usuarios pueden almacenar sus datos en una base de datos pública junto con otros datos.

El uso predominante de esto fue un sistema parecido al DNS. Mapeando dominios como "namecoin.bit.[en] una dirreción IP. El algoritmo básico para implementar algo del estilo en ethereum sería:

Algorithm 2 Sistemas de identidad.

```
1: procedure IDENTIDAD(msg, contract)          ▷ msg contiene la información
2:     if contract.storage[tx.data[0]] ≠ 0 then
3:         contract.storage[tx.data[0]] = tx.data[1]
4:     end if
5: end procedure
```

El contrato es muy simple. Consiste en que todo dentro de la red de Ethereum es como una base de datos que puede ser añadida pero no modificada o borrada. Cualquiera puede registar un nommbre con un valor, y este nombre se queda registrado para siempre.

Modificaciones más sofisticadas podrían tener función de consulta, dando acceso a otros contratos a ver la información, o un mecanismo que permitiera cambiar el dominio.

3.3. Decentralized File Storage

En el último tiempo han surgido un grupo de aplicaciones populares para a compartición de archivos. Aún así, el mercado aquí es bastante ineficiente, en particular en el valle desamparado del 20-200 GB donde no hay cuentas gratis ni descuentos de compañía.

Los contratos de ethereum podrían permitir el desarrollo de un sistema de alacenamiento de archivos, donde usuarios podrían ganar pequeñas cantidades de dinero alquilando espacio en sus propios discos duros, reduciendo los costes de estos sistemas.

La parte clave sería implementar el "protocolo del Dropbox descentralizado". Este contrato funcionaría así:

- Primero, uno parte los datos quedesea almacenar en bloques, encryptando estos para otorgar privacidad, y construye un arbol de guía (merkle tree) para estos. Cada N bloques, y o ofrece a público que quiera almacenarlo.
- Los host, compiten por almacenarlo, y el primero en aprobar una prueba de verificación de que posee el archivo, se le ceden X ether.
- Cuando un usuario quiere re- descargar este archivp, realiza un micropago para recuperar el archivo. El contrato no ahce pública la transacción hasta que no se ha tranferido todo el bloque.

Una parte importante del protcolo consiste en ue sea el contrato quien compruebe periodicamente la conssitencia del archivo en el disco que lo aloja, para ver durante cuanto tiempo se tiene que estar pagando por este serricio, y ofreciendo de nuevo el bloque a almacenar cuand el host lo decida quitar.

3.4. Organizaciones descentralizadas.

El concepto de organiación descentralizada, explicado por nuestros compañeros Dani y Andrés. Este concepto, resumiendo, es el de una entidad que tiene un cierto número de miembros que con un a mayoría, por ejemplo del 80 %, puede administrar los fondos de la entidad y el código del contrato.

Los métodos para repartir los fondos de una DAO van desde dar comisiones o salarios, hasta métodos más intrincados como crear una propia moneda interna para recompensar el trabajo. Esto, en esencia, replica el funcionamiento de una compañía tradicional, pero usando solo la tecnología de la cadena de bloques para asegurar su fucnionamiento. This essentially replicates the legal trappings of a traditional. Mucha gente habla del modo capitalista de una DAO, también llamada corporación autonoma descentraliza, o por sus siglas en ingleś DAC, donde todos los miembros tendrán igual poder e decisión sbre las acciones y serán participes de los mismo dividendos. Esta compañía tendrá un código de contrato, solo modificable si se decide por una mayoría superior a la establecida por convenio (sea, 67 %).

Una línea general de cómo constituir una DAO se podría hacer de manera sencilla, como por ejemplo, haciendo una porción de código auto modificable si dos tercios de los miembros están de acuerdo en modificarlo. Aunque el código es en teoría inmutable, uno puede fácilmente tener mmutabilidad de facto teniendo porciones de código en diferentes contratos y dirigiendo desde la parte modificable del principal a estos. En este diseño habría tres tipos de transacciones:

- [0,i,K,V]: Registrar una proposición con índice i para cambiar la dirección del contrato K a la dirección del contrato V.
- [v,i]: registrar un voto en la proposición i. El índice v marcará la intención de ese voto,
- [2,i]: Finalizar la proposición si hubo suficientes votos. Esta equivaldría a votar en blanco.

El contrato tenŕia condiciones para cada una de estas y mantendría un resgistor de todas las propocisiones abiertas. También tendría una lista de todos los miembros.

Exisitirían otros medios más sofisticados para implementar este tipo de asociaciones, como contratos con partes decódigo especiales para tratar la permanencia o no de gente en la compañía o delegación de votos en el estilo de la democracia representativa.

4. Particularidades.

4.1. Ether.

Como ya comentamos anteriormente, ethereum aspira a funcionar tanto como una especie de internet descentralizado como una app store descentralizada, apoyando así un nuevo tipo de aplicación (“dapp”) en el proceso.

Sin embargo, aunque nadie sea dueño de ethereum, el sistema que respalda esta funcionalidad no es gratis. Mejor dicho, la red necesita el ‘ether’, una pieza única de código que puede usarse para pagar por los recursos computacionales necesarios para ejecutar una aplicación o programa. El ether no es más que la divisa utilizada en ethereum para las transacciones. Al igual que el dinero en efectivo, no necesita de terceros para efectuar o aprobar una transacción. Pero en lugar de funcionar como una moneda o pago digital, el ether busca proporcionar gas para las apps descentralizadas en la red. El ether es la herramienta necesaria para el funcionamiento de los smartcontract ya explicados, y del mismo modo, es la recompensa que obtienen los mineros

por su trabajo.

Aunque pueda sonar complicado, podemos pensar en un ejemplo más concreto de cómo los tokens pueden potenciar la experiencia de un usuario. Pensemos en un ejemplo de un cuaderno descentralizado online. Para añadir, borrar o modificaruna nota, necesitamos pagar una cuota o tarifa de transacción en ether para conseguir que la redprocese el cambio.

De los ether que ya existen, 60 millones fueron comprados por usuarios en una campaña de crowdfunding en 2014. Otros 12 millones de ether fueron destinados a la Fundación Ethereum, un grupo de investigadores y desarrolladores que trabajan en la tecnología subyacente. Cada 12 segundos, 5 ethers (ETH) son también repartidos entre los mineros que verifican transacciones en la red. Como mucho se minan dieciocho millones de ether al año.

4.2. DoS attack.

Dado el incidente del DAO, nos dispondremos a explicar en qué consiste este ataque que sufrió la red de Ethereum.

En términos generales, un 'denial-of-service attack' (DoS attack) es un ciberataque en el que el autor del mismo tiene el objetivo de inhabilitar una máquina o recurso de red de cara a sus usuarios interrumpiendo temporal o indefinidamente los servicios de un host conectado a internet. Típicamente, el DoS se realiza 'inundando' la máquina o recurso objetivo con peticiones superfluas en un intento de sobrecargar el sistema y evitar que se cumplan todas o parte de las peticiones legítimas.

En un ataque DoS distribuido (DDoS), el tráfico entrante que inunda a la víctima proviene de muchas fuentes distintas, lo que efectivamente hace que sea imposible detener el ataque bloqueando una simple fuente.

Un ataque DoS o DDoS podría compararse a un grupo de personas amontonándose en la puerta de entrada a una tienda o negocio, impidiendo el paso de un grupo de personas legítimas e interrumpiendo el funcionamiento normal. Nos centraremos en analizar los ataques DDoS. DoS DISTRIBUIDO (DDoS) Un DDoS es un ciberataque en el que el autor utiliza más de una dirección IP única (a menudo miles de ellas). Como hemos comentado anteriormente, el tráfico entrante que pretende inundar el sistema se origina en diversas fuentes, por lo que es imposible detener el ataque con un filtrado de ingreso para asegurar que los paquetes que entran son en realidad de las redes de las que dicen ser originados. A su vez, hace que sea muy complicado distinguir entre tráfico de un usuario legítimo y tráfico de un ataque cuando está extendido por tantos puntos de origen. Muchos de los ataques implican la falsificación de las direcciones IP del remitente (IP spoofing o suplantación de IP) como una alternativa o aumento de un DDoS, así como para conseguir que no se pueda identificar fácilmente la localización de las máquinas atacantes ni derrotarlas.

4.2.1. Técnicas de ataque

Un ataque DDoS es normalmente el resultado de sistemas múltiples comprometidos, por ejemplo, una botnet, inundando el sistema objetivo con tráfico. Una botnet es lo que viene siendo una red zombie de ordenadores programados para recibir comandos sin el conocimiento de los propietarios. Cuando un servidor se sobrecarga con conexiones, las nuevas conexiones ya no pueden aceptarse. Las principales ventajas para el atacante de usar un ataque DDoS es que múltiples máquinas pueden generar mas tráfico de ataque que una sola máquina, al igual que es

más difícil desconectar todas las máquinas, y que el comportamiento de cada máquina atacante puede ser muy sigiloso, haciendo aún más complicado rastrearlas y apagarlas. Estas ventajas desafían a los mecanismos de defensa.

En 2015, algunas de las botnets DDoS crecieron en prominencia, tomando como objetivo instituciones financieras. Los ciber-extorsionistas normalmente empiezan con un ataque de bajo nivel y un aviso de que un ataque mayor va a llevarse a cabo si no se paga un rescate en Bitcoin.

4.3. Escalabilidad

Una preocupación común acerca de Ethereum es el tema de la escalabilidad. Ethereum sufre el defecto de que cada transacción tiene que ser procesada por cada nodo en la red. Con Bitcoin, el tamaño de la actual cadena de bloques se basa en aproximadamente 20 GB, creciendo 1 MB por hora. Si la red de Bitcoin fuese a procesar 2000 transacciones de Visa por segundo, crecería 1 MB cada 3 segundos (1 GB por hora, 8 TB por año). Es probable que Ethereum sufra un patrón de crecimiento similar, empeorado por el hecho de que habrá muchas aplicaciones por encima de la cadena de bloques de Ethereum, en lugar de sólo una moneda como en el caso de Bitcoin, pero a su vez mejorado por el hecho de que los nodos completos de Ethereum necesitan almacenar sólo el estado en lugar del historial entero de la cadena de bloques.

El problema que tiene una cadena de bloques de tamaño tan grande es el riesgo de centralización. Si el tamaño de la cadena de bloques crece hasta, por ejemplo, 100 TB, el escenario probable sería que solo un pequeño número de grandes empresas ejecutarían nodos completos, con todos los usuarios habituales usando nodos SPV (Simplified Payment Verification) ligeros. En una situación así, surge la preocupación de que los nodos completos se unan y todos acepten hacer trampa o estafar de alguna manera que les sea rentable, como por ejemplo cambiar la recompensa de bloque, autoasignarse ether, es decir, al igual que en el sistema capitalista, se cumpliría un fallo por oligopolio. Los nodos ligeros no tendrían forma alguna de detectar esto inmediatamente. Por supuesto, probablemente existiría al menos un nodo completo que fuese honesto, y después de unas horas la información sobre el fraude se volcaría a través de canales como Reddit, aunque llegados a este punto sería demasiado tarde: sería responsabilidad de los usuarios habituales organizar un esfuerzo para poner en la lista negra los bloques dados, un problema masivo y probablemente inviable de coordinación en una escala similar a la de un exitoso ataque del 51 %

A corto plazo, Ethereum usará dos estrategias adicionales para lidiar con este problema. En primer lugar, debido a los algoritmos para minar basados en la cadena de bloques, al menos cada minero se verá obligado a ser un nodo completo, creando así un límite más bajo en cuanto al número de nodos completos. En segundo lugar y más importante, sin embargo, se incluirá una raíz de árbol de estado intermedio en la cadena de bloques después de procesar cada transacción. Aunque la validación de bloques sea centralizada, mientras que exista un nodo de verificación honesto, el problema de la centralización puede eludirse a través de un protocolo de verificación. Si un minero publica un bloque inválido, o bien ese bloque tendrá que estar mal formateado, o bien el estado S[n] será incorrecto. Dado que se sabe que S[0] es correcto, tendrá
que haber un primer estado S[i] que sea incorrecto donde S[i-1] es correcto. El nodo de verificación proporcionaría el índice i, junto con una 'prueba de invalidación' que consiste en el subconjunto de nodos del árbol del usuario que necesitan procesar

```
APPLY( S[i-1] , TX[i] ) -> S[i]
```

Los nodos serían capaces de usar esos nodos para ejecutar esa parte del cálculo, y ver que el estado S[i] generado no concuerda con el S[i] previsto.

Otro ataque más sofisticado involucraría a los mineros maliciosos que publicasen bloques incompletos, ya que la información completa ni siquiera existiría para determinar si los bloques son válidos o no. La solución a este problema es un protocolo 'desafío-respuesta': los nodos de verificación emiten 'desafíos' en forma de índices de transacción objetivo, y al recibir un nodo, un nodo ligero toma el nodo como no fiable hasta que otro nodo, ya sea por el minero o cualquier otro verificador, proporcione un subconjunto de los nodos de dicho usuario como una prueba de validez.

4.3.1. Ataque 51 %

Este tipo de ataque es bastante simple, y consistiría en tener un 51 % del poder computacional de la red. Este suceso brindaría la opción de escribir cosas maliciosas en la cadena, y extenderlas por toda la red de modo que las transacciones verdaderas quedasen ocultas por estas.

4.4. Minería centralizada

El algoritmo de minería de Bitcoin básicamente funciona teniendo mineros calculando SHA256 (Secure Hash Algorithm) en versiones ligeramente modificadas de la cabecera de bloque millones y millones de veces, hasta que eventualmente aparezca un nodo con una versión cuyo hash sea menor que el objetivo (actualmente alrededor de 2 elevado a 190).

Sin embargo, este algoritmo de minería es vulnerable respecto a dos formas de centralización. En primer lugar, el ecosistema minero ha pasado a ser dominado por ASICs (application-specific integrated circuits), chips computadores diseñados para la tarea específica de la minería de Bitcoin, y por lo tanto miles de veces más eficientes. Esto quiere decir que la minería en Bitcoin ya no es una búsqueda altamente descentralizada e igualitaria, ya que requiere millones de dólares de capital para participar en ello eficazmente. En segundo lugar, la mayoría de mineros de Bitcoin realmente no efectúan la validación de bloques localmente; en su lugar, dependen de una mining pool centralizada para proporcionar las cabeceras de bloque. Este problema es discutiblemente peor: actualmente, las dos mining pools principales indirectamente controlan aproximadamente el 50 % del poder de procesamiento en la red de Bitcoin, aunque esto está mitigado por el hecho de que los mineros pueden cambiarse a otras mining pools si una pool o coalición pretende llevar a cabo un ataque del 51 %.

El propósito actual en Ethereum es usar un algoritmo de minería basado en generar aleatoriamente una única función hash por cada 1000 'nonces', usando un rango de computación lo suficientemente amplio para eliminar el beneficio del hardware especializado. Tal estrategia ciertamente no reducirá la ganancia de centralización a cero, pero tampoco lo necesita. Observemos que cada usuario individual, en su escritorio o portátil privado, puede realizar una determinada cantidad de actividad minera casi gratis, pagando simplemente el coste de la electricidad, pero llegados al punto de la utilización del 100 % de la CPU de sus ordenadores, la minería adicional requerirá que paguen tanto por la electricidad como por el hardware. Las compañías mineras ASIC tienen que pagar por la electricidad y el hardware desde el primer hash.

Por lo tanto, si la ganancia de centralización puede mantenerse bajo esta proporción, $(E+H)/E$, entonces incluso si se fabrican ASICs seguirá habiendo espacio para mineros corrientes. Adicionalmente, el propósito es diseñar el algoritmo de minería de manera que el hecho de minar requiera acceso a la cadena de bloquea completa, obligando a los mineros a almacenar la cadena

de bloques entera y, al menos, ser capaces de verificar cada transacción. Esto suprime la necesidad de mining pools centralizadas; aunque las mining pools pueden ejercer el verdadero papel de equilibrar la aleatoriedad de la distribución de recompensas, esta función puede ejercerse de igual manera por pools 'peer-to-peer' sin control central. Esto, a su vez, ayuda a combatir la centralización, incrementando el número de nodos completos en la red de forma que ésta permanece razonablemente descentralizada, incluso si la mayoría de usuarios habituales prefieren clientes ligeros.

Índice