Frente a la Competencia









MAGAZINE

Ardor Frente a la Competencia

Editor: apenzl
Escritor: segfaultsteve
Traducción: jose, rubenbc, George
Diseño: apenzl, en nombre de Nxter Magazine

Publicado 24.11.2017

Esta publicación también está disponible en inglés, chino, ruso y otros idiomas.

Agradecimientos especiales a: segfaultsteve, jose, rubenbc, fz1128, sergei, ANG y Jelurida.

ÍNDICE

Introducción

24.11.2017, Nxtville.

Enhorabuena, Nxt.

¡Es increíble lo rápido que pasa el tiempo!

Nxt fue lanzado hace cuatro años como la primera criptomoneda 2.0 en ser 100% Prueba de Participación (Proof of Stake – PoS), tal y como está explicado y detallado en el libro ‘ SNAPSHOT’ , y posteriormente Jelurida B.V. fue creada y registrada como “una compañía de desarrollo de software dedicada a la creación de las tecnologías blockchain de Nxt y Ardor”. Jelurida ahora publica las nuevas versiones de Nxt bajo la licencia ‘ JPL’, que pretende que, si surgen clones, estos sean justos con todas las partes.

Hoy, en el 4º aniversario de Nxt, Jelurida ha publicado el código fuente de Ardor.

Se recaudaron 15.000.000$ en la ICO de IGNIS, el token de la cadena hija de la plataforma Ardor. Ignis hereda todas las características de Nxt (como la transferencia de tokens, además de las transacciones inteligentes que permiten lanzar contratos inteligentes incrustados en el núcleo del código), y algunas otras novedades. Jelurida afirma que “estos fondos serán utilizados para continuar con el desarrollo, mantenimiento, progreso y promoción en todo el mundo de las plataformas blockchain de Nxt y Ardor, así como para proteger la propiedad intelectual del código base”. Ignis será lanzado con el Bloque Génesis de Ardor, que será forjado a las 0:00h UTC del 1 de enero de 2018.

Pero... ¿Qué es Ardor? ¿Qué es Nxt?

Los no familiarizados pueden EMPEZAR AQUÍ, los nuevos *nexteros* y desarrolladores deberían echar un vistazo a este documento técnico / whitepaper.

Hace alrededor de tres meses que segfaultsteve se unió a /r/ardor, afirmando no ser un "experto en Ardor", pero de inmediato comenzó a responder preguntas sobre la plataforma blockchain de Ardor, evidenciando un gran conocimiento y comprensión de su diseño. Poco tiempo después, publicaba su primer post original:

> *Muchos chicos están hablando entusiasmados sobre Plasma, así que decidí echarle un vistazo yo también. Estoy particularmente interesado en comparar y contraponer las "cadenas hijas" o "child chains" de Plasma y las de Ardor. No obstante, no me está resultando fácil, puesto que, siendo sinceros, el documento técnico de Plasma está escrito de una manera bastante pobre. Ya está, ya lo dije.*
>
> *Estoy publicando una exposición de ideas, así que todos podéis ayudarme a corregirlo si he malinterpretado algo. De cualquier manera, espero que también lo encontréis útil. :)*

Así comenzaba su exposición de ideas:

> *Según lo entiendo, la diferencia clave entre una cadena de Plasma en Ethereum y una cadena hija de Ardor es que los nodos que protegen la red de Ethereum (es decir, los mineros) no validan directamente las transacciones de la cadena de Plasma, mientras que en Ardor, todos los nodos en la red de Ardor, incluyendo los forjadores, validan las transacciones de las cadenas hijas.*
>
> *El resto de diferencias entre Plasma y Ardor (como los vínculos que los validadores publican en contratos inteligentes en la cadena madre; las "pruebas de fraude", que obliga a los estafadores a romper sus vínculos; el mecanismo de "salida masiva", que se supone debe proteger frente a ataques de retención de bloques...) todas son consecuencia de esta primera diferencia, en el sentido de que ninguna de estas cosas son necesarias en Ardor puesto que los nodos de forjado ya validan directamente las transacciones de las cadenas hijas.*
>
> *En Ardor, solo las cuentas que poseen ARDR pueden forjar, y cualquier transacción que cambie el saldo de ARDR es grabada directamente en la cadena de Ardor. Puesto que la red al completo (incluyendo los forjadores) valida las transacciones de las cadenas hijas, uno puede comprobar cualquier bloque en la cadena madre y verificar que la red en ese momento alcanzó el consenso sobre las transacciones de la cadena hija y, por tanto, el estado resultante de las cuentas de las cadenas hijas era válido. Como consecuencia, el estado actual de las cuentas es también válido (estoy dejando de lado algunos detalles, pero este es el grueso del funcionamiento, en mi opinión).*
>
> *En Plasma, sin embargo, los mineros de ETH no ven las transacciones de las cadenas*

hijas. En su lugar, el consenso en las transacciones de las cadenas hijas proviene de los "validadores" de las cadenas hijas, que son cuentas que se encargan de forjar los bloques de las cadenas hijas (por ejemplo, usando el algoritmo de Prueba de Participación) y enviar sus hashes a la cadena madre. Estos validadores han publicado vínculos en la cadena madre que están "vigilando" en busca de pruebas de fraude por parte de los usuarios, y si un usuario puede demostrar que un validador ha forjado un bloque inválido, el validador pierde el vínculo. De esta manera, los validadores tienen un incentivo para rechazar las transacciones fraudulentas. En última instancia, sin embargo, el mecanismo de consenso parece estar fundamentado en el deseo de los usuarios por auditar los bloques de las cadena hija y alertar sobre los validadores que hacen trampa.

Sin embargo, ¿qué sucedería si el validador (o un conjunto de ellos) de una cadena hija retuviese los nuevos bloques? En este caso no sería necesariamente posible construir una prueba de fraude, ya que parte de la información requerida aún no estaría disponible (aunque admito que estoy un poco confuso sobre esta parte). En este caso, para reducir al mínimo el daño que podría ser causado, en Plasma se describe un mecanismo bastante elaborado de "salida masiva" que permite a coaliciones de usuarios retirar los fondos en masa si detectan que los validadores están reteniendo los bloques. Otra diferencia entre Plasma y Ardor que probablemente vale la pena mencionar es que las cadenas de Plasma se pueden organizar en una estructura de datos de árbol, donde cada cadena de Plasma puede tener múltiples hijas, y esas hijas pueden tener más hijas, y así sucesivamente. En algunos casos, esto facilitará el escalado de grandes cálculos a través de una gran cantidad de cadenas hijas, utilizando un algoritmo similar a MapReduce.

Quizás lo he entendido mal, pero soy un poco escéptico con esta parte. Los grandes trabajos solo son escalables en casos de que cada nodo haga un pequeño subconjunto de cálculos. Sin embargo, si solo estoy validando / controlando mi parte del cálculo, y nadie está validando otra parte, entonces el resultado final obtenido será incorrecto. Parece que la única manera de combatir esa posibilidad es validando todos los cálculos (es decir, validando todas las transacciones de todas las cadenas hijas involucradas), lo que significa que con MapReduce no se logra ningún beneficio de escalabilidad.

Entonces, ¿qué hacemos con todo esto?

En mi opinión, la principal similitud entre Ardor y Plasma es que ambos reducen el crecimiento desmesurado de la cadena de bloques almacenando solo el hash de los bloques de las cadenas hijas en la cadena principal. Sin embargo, hay varias diferencias importantes:

- *En Ardor, todos los nodos validan todas las transacciones de las cadenas hijas, mientras que los nodos de Plasma solo necesitan validar las transacciones que les afectan. Esto podría aumentar en gran medida el rendimiento total de la red, ya que la mayoría de los nodos podrían ignorar la mayoría de los cálculos, literalmente todas aquellas que pertenezcan a aplicaciones no relacionadas. Una característica potencialmente similar, que está en la hoja de ruta de Ardor, es la*

delegación de la verificación de las transacciones de las cadenas hijas en subredes dedicadas, pero no existen muchos detalles en el documento técnico.

- *El concepto de "confirmar" una transacción en Ardor es bastante directo. Por el contrario, hay un período de tiempo en una cadena de Plasma donde un bloque se puede revertir, incluso aún cuando un validador lo haya mezclado y agregado a la cadena principal (creo), puesto que es posible que alguien presente una prueba de fraude. Me gustaría entender mejor este mecanismo.*

- *Ardor ya existe :)*

En los últimos meses, segfaultsteve ha leído, analizado y se ha centrado en diferentes plataformas blockchain, comparándolas con Ardor, y ha escrito un blog sobre esto en Nxter Magazine.

Sin más preámbulos, puedes adentrarte en la serie de artículos de análisis "Ardor frente a la Competencia", que dan un vistazo en profundidad a Ardor, Ethereum, Lisk, IOTA, Waves, Stratis, NEM y Komodo.

- apenzl

LISK

Ardor Frente a la Competencia, Parte 1

by segfaultsteve

Recientemente he decidido comenzar una serie de artículos que comparen y contrasten Ardor con otros proyectos de cadenas de bloques (*blockchains*) que aparentan tener objetivos y características similares. Aproximadamente cada semana escogeré un proyecto que su alcance coincida, aunque solo sea en una pequeña parte, con Ardor. Analizaré la documentación técnica del otro proyecto y publicaré un resumen de mis descubrimientos.
Esta semana he estado leyendo sobre Lisk.

Lisk

En pocas palabras, Lisk es una plataforma para el desarrollo aplicaciones descentralizadas (dapps) que se ejecutan en cadenas de bloques laterales (sidechains) ancladas a la cadena de bloques (blockchain) principal de Lisk. Para mantener la seguridad de la cadena principal utiliza un mecanismo de consenso llamado Prueba-de-Participación Delegada (Delegated Proof-of-Stake ó DPOS). Cada cadena lateral es responsable de su propia seguridad (algo parecido, pero por favor lee la descripción del “mercado de delegados” que se encuentra a continuación). El

protocolo utiliza una serie de transacciones predefinidas, similares a Nxt o Ardor, a diferencia del lenguaje de scrypt de bajo nivel que forma la base de Bitcoin o Ethereum.

Antes de entrar en detalle, debo comenzar diciendo que Lisk está definitivamente en una fase inicial de desarrollo. El equipo está en la mitad del proceso de volver a escribir el SDK de Lisk, encargado de apoyar el desarrollo de las cadenas laterales (sidechains) y está continuamente reconfigurando el Lisk Core, que es el nodo completo.

Con el código en flux, algunas cuestiones básicas de su arquitectura, en particular sobre las cadenas laterales (sidechains) y cómo intereactuan unas con otras y con la mainchain, todavía no parecen haber sido definidas. Por otro lado, me resultó muy difícil encontrar una fuente de información técnica y autorizada sobre Lisk, por lo que lo que presento aquí tal vez podría estar desactualizado. La mejor fuente de información que encontré fue la wiki, este artículo de uno de los co-fundadores, la hoja de ruta, y estos videos en YouTube. Desgraciadamente, ninguna de las tres fuentes es reciente, y los vídeos incluso no tratan el tema con mucha profundidad (aunque debo admitir que no he visto las más de 6 horas de estos videos). Si encuentras mejores referencia, te agradecería que me las hicieses llegar.

El grueso del marketing que rodea Lisk parece centrarse en el SDK, cuya meta es la de permitir construir y lanzar fácilmente un dapp seguro en una cadena de bloques adaptable. Los desarrolladores escribieron el SDK en Javascript porque querían hacer a Lisk accesible a una audiencia lo más amplia posible, y también desarrollaron el *backend* en Javascript (Node.js) porque, bueno… creo que nunca entenderé porque la gente insiste en escribir el *backend* en Javascript. :)

Pero claramente, la facilidad de desarrollar y desplegar una cadena de bloques a medida no es la única meta de Lisk. Si lo fuese, entonces ¿cuál sería el propósito de la cadena principal? Podrías también clonar Bitcoin o Nxt si todo lo que necesitas es solo un buen punto de inicio para construir tu propia cadena de bloques.

La arquitectura de cadena principal / cadena lateral (mainchain / sidechain) es realmente la característica distintiva de esta plataforma. Por lo que puedo determinar, la cadena principal tiene tres funciones primordiales:

1. La API de Lisk permite depósitos de LSK en la cadena principal para ser transferidos hacia/desde las cadenas laterales. Con esas dos transacciones será posible enviar LSK desde una cadena lateral a otra, por medio de la cadena principal. Desgraciadamente, de acuerdo al artículo escrito por uno de los co-fundadores, cuyo link se encuentra abajo, parece ser que enviar LSK hacia un *sidechain* requerirá enviarlo *al dueño de la cadena lateral*, lo que requiere cierto grado de confianza, obviamente. Para evitar este problema, será posible crear cadenas laterales que utilicen sus propios *tokens* de forjado, en lugar de LSK. Este token deberá ser *tradeado* por LSK para poder realizar las transacciones a través de la cadena principal con otras cadenas laterales. Alternativamente, puede que sea posible que una cadena lateral realizar transacciones directamente con otra cadena lateral sin tener que recurrir a la cadena principal, pero los

desarrolladores se encuentran todavía investigando como funcionaría esto.

2. A la larga, el equipo planea construir un “mercado de delegados” dónde los delegados que no estén asegurando la cadena principal pueden ofrecer seguridad a las cadenas laterales. Estos delegados, podrían ser pagados por el dueño de la cadena lateral o por sus usuarios. Una vez más, los detalles son un poco confusos, pero aquí parece haber mucho valor: presumiblemente la red de Lisk ya es mucho más grande que una cadena de bloques típica y el “mercado de delegados” proporciona a las cadenas laterales un conjunto de nodos disponibles listos para utilizarse y proteger a la cadena desde sus primeros pasos.

3. Algunos nodos de la red (no estoy seguro de cuáles) calcularán, de manera periódica, el *hash* de las cadenas laterales y los almacenará en la cadena principal como una manera de “validación básica de la integridad de la cadena lateral”. Sin embargo, no me ha sido posible encontrar detalles de cómo se espera que este mecanismo funcione.

Además de estas funciones, y del papel obvio que juega en la transferencia de LSK entre cuentas, la cadena principal en sí misma no parece tener otros usos previstos. Toda la actividad se supone que será realizada en las cadenas laterales.

Comparado con Ardor

¿Cómo se compara esta arquitectura con el esquema de cadena madre / cadenas hijas de Ardor? Quizá la diferencia más obvia es que cada cadena lateral debe de tener su propio conjunto de nodos para asegurarla, que puden ser proporcionados por el creador de la cadena lateral, por los usuarios o, a largo plazo, por el “mercado de delegados”.

En contraste, en la red Ardor todos los nodos validan las transacciones de las cadenas hijas, pero solo las cuentas que poseen ARDOR forjan. El hecho de que las cuentas con *tokens* de las

cadenas hijas no los utilicen para forjar significa que no importa lo pequeñas que sean estas cadenas hijas o lo desigual que sea su distribución. Son tan seguras como la cadena madre.

Una nota adicional sobre Lisk es que, hasta el momento en que el "mercado de delegados" sea lanzado, los creadores de las cadenas laterales elijen los nodos que forjan en sus cadenas, lo que parece requerir que los usuarios depositen un buen grado de confianza en ellos. Por otra parte, el equipo ha sugerido que Lisk será suficiente flexible para permitir que las cadenas laterales utilicen un algoritmo de consenso totalmente distinto, como el de Prueba-de-Trabajo (Proof-of-Work), lo que parece indicar que los creadores de estas cadenas laterales no determinarán qué nodos mantendrán la seguridad la cadena.

También existen planes para permitir a las cadenas laterales existentes cambiar sus mecanismos de consenso, incluso después del lanzamiento, pero de nuevo no pude encontrar más detalles sobre esto.

Claramente, tanto Lisk como Ardor pretenden ofrecer ventajas de escalabilidad sobre las *blockchains* tradicionales. Con Lisk las ventajas computacionales para el escalado son obvias, ya que cada nodo forjador valida únicamente las transacciones en una cadena de bloques, ya sea la cadena principal o la lateral. Sin embargo, la reducción del espacio de almacenamiento requerido (crecimiento desmesurado de la *blockchain*) es menos claro. En comparación, por ejemplo, con Ethereum, es obvio que para un nivel de actividad total similar, las cadenas principales en el ecosistema de Lisk crecerán a un ritmo menor que la cadena única de Ethereum, sencillamente porque las cadenas laterales no almacenarán los datos de cada una de las otras.

Comparada con Ardor, los ahorros de almacenamiento serían muy reducidos. La cadena madre de Ardor crecerá a ritmos similares que la cadena principal de Lisk (ya que ambas almacenarán solamente los *hashes* de las cadenas laterales o de las cadenas hijas, respectivamente) en lugar de los datos en sí. Sin embargo, en Ardor se eliminarán los datos de las cadenas hijas, eliminando así el problema del crecimiento desmesurado de la *Blockchain*, un problema que Lisk tendrá en cada cadena lateral.

Conclusión

¿Entonces que debemos hacer con Lisk? Honestamente, estoy muy decepcionado al escribir esto, pero creo que es demasiado pronto para decirlo.T odavía existen muchos detalles que tienen que materializarse:

- ¿Será posible convertir el *token* de una cadena lateral directamente a otra cadena lateral sin convertirlo a/desde LSK? ¿Cómo?
- Cuando el "mercado de delegados" abra, ¿será posible que los usuarios elijan a los delegados utilizando los *tokens* de la cadena lateral? ¿O tendrán que utilizar LSK? ¿O podrán los dueños de las cadenas laterales mantener el control sobre que delegados forjan?
- ¿Qué hará Lisk con los *hashes* de las cadenas laterales almacenados en la cadena

principal? ¿Será posible revertir las transacciones recientes de una cadena lateral para "restaurar" el estado que tenían cuando se produjo el *hash* correspondiente? Si esto fuese así, ¿existirá algún periodo de tiempo después del cual no fuese posible revertir las transacciones, para que la cadena siguiese siendo considerada inmutable?

- ¿Proporcionará el SDK de Lisk un mecanismo claro para cambiar el algoritmo de consenso de una cadena lateral ya existente? No estoy muy seguro como sería esto.
- ¿Qué sucederá si la cadena lateral utilizase una bifurcación de LSK? Obviamente, los tokens LSK de cada cadena lateral resultante no podrán ser simultáneamente respaldados por las mismas reservas de LSK en la cadena principal. Asumo que el creador de la cadena lateral decidirá cuál es la cadena "real", ya que mantiene las reservas en la cadena principal, pero no sé con seguridad si esta intrepretación es la correcta.
- Dependiendo del modo en que Lisk soporte las transacciones directas entre las cadenas laterales, este problema podría requerir de confianza adicional entre los creadores de la *sidechain*. En particular, si los creadores de la *sidechain* deben mantener reservas de los *tokens* de cada una de las cadenas para permitir transacciones entre cadenas, lo cual parece una manera plausible de hacerlo, entonces un *fork* en una *sidechain* podría darle al creador de la otra *sidechain* cierta influencia sobre qué rama de la bifurcación es la válida. Además, si la cadena lateral del *fork* realiza transacciones con otras varias *sidechain*, cada una de las cuales tuviese reservas del *token* dividido, entonces la situación podría ponerse fea bastante rápidamente

En mi opinión, la ventaja más importante de Lisk sobre las otras plataformas de cadenas de bloques, incluyendo Ardor, es que podría alcanzar una escalabilidad computacional natural derivada de la separación de cada dapp en su propia cadena de bloques. Si sumamos que las cadenas laterales pudieran realizar transacciones entre ellas fácilmente y sin necesidad de requerir confianza en un tercero, podría tener un potencial inmenso.

Si asumimos que el equipo de Lisk implementa exitosamente todas las funciones para hacerlo posible, entonces debemos de otorgarle la misma cortesía a Jelurida y asumir que ellos llevaran a cabo también sus planes de escalabilidad. En particular, una mejora potencial en la hoja de ruta de Ardor es la de limitar el procesamiento de las transacciones de las *child chains* a subredes dedicadas en la red de Ardor. Parece que esto lograría una escalabilidad computacional similar a Lisk, preservando al mismo tiempo la ventaja substancial de reducir el crecimiento desmesurado de la cadena de bloques.

Concluyendo, la arquitectura de cadena principal / cadenas lateral de Lisk podría potencialmente ayudarlo a escalar para permitirle acomodar un gran número de dapps, las cuales podrían interactuar de formas interesantes. Pero, por ahora, parece haber mucha incertidumbre en los detalles técnicos. El planteamiento de Ardor es muy diferente técnicamente, pero resuelve los mismos problemas. Principalmente el crecimiento de tamaño desmesurado de la cadena de bloques, la escalabilidad del potencial computacional y la habilidad de lanzar transacciones fácilmente entre las diferentes cadenas.

Será interesante ver cómo se desarrolla Lisk en los próximos dos o tres años, pero para entonces Ardor ya habrá estado funcionando por un largo periodo.

NEM/Mijin/Catapult

Ardor vs The Competition Pt 2

Esta semana he analizado NEM, una **blockchain** similar a Nxt en muchos aspectos. Puesto que estoy principalmente interesado en cómo resuelve cada *blockchain* el problema del escalado, también he investigado acerca de Mijin, una versión de NEM para *blockchains* privadas y Catapult, una re-reescritura de Mijin que promete grandes mejoras de rendimiento que también serán incorporadas a los futuros lanzamientos de NEM.

NEM

Aunque los desarrolladores centrales de NEM abandonaron su idea original de lanzar NEM como un *fork* de Nxt, optando por empezar el proyecto desde cero, NEM y Nxt continúan siendo bastante similares. Al igual que sucede en Nxt, la plataforma NEM ofrece una serie predefinida de transacciones permitidas que las aplicaciones pueden usar como elementos base de construcción con los que crear características más complejas, en oposición al uso de *scripts* de lenguaje de bajo nivel para la construcción de transacciones, como sucede con Bitcoin o Ethereum.

Ambas plataformas soportan un puñado de características de "*blockchain* 2.0", como pueden ser el envío de mensajes, la creación y transferencia de activos o el envío de transacciones que

requieren de la aprobación por múltiples cuentas (multifirma). Y ambas plataformas ofrecen funcionalidad a través de APIs basadas en HTTP, así que los desarrolladores pueden usar virtualmente cualquier lenguaje para escribir sus aplicaciones.

A pesar de las semejanzas, NEM también tiene algunas diferencias notables con Nxt.

Quizá la más fundamental es su novedoso algoritmo de consenso, llamado Prueba de Importancia (Proof-of-Importance). Este algoritmo es similar al de Prueba de Participación (Proof-of-Stake), excepto en que la probabilidad de que una cuenta recolecte (es decir, forje) el siguiente bloque depende no solo de su saldo de XEM, la moneda nativa en NEM, sino que también depende de hace cuánto que ha realizado transacciones con otras cuentas y cuánto XEM se intercambió en ellas. Las cuentas que tienen mucho saldo de XEM y que realizan transacciones con un gran volumen frecuentemente recolectarán más bloques que las cuentas con menos XEM o aquellas cuentas que realicen pocas transacciones.

Los autores del manual de Referencia Técnica de NEM defienden que, al compararse con la prueba de participación, el algoritmo de prueba de importancia de alguna manera ofrece menos peso a las cuentas más acaudaladas a la hora de elegir el derecho a forjar/recolectar el siguiente bloque (Sección 7.8). La prueba de importancia también es vital para el filtro de *spam* de NEM, puesto que requiere que un atacante no solo controle muchas cuentas, lo cual es sencillo de hacer, pudiendo *spamear* la red con un gran número de transacciones sin confirmar, sino que también exige disponer de un considerable saldo en cada cuenta y realizar transacciones frecuentemente con otras cuentas también de gran importancia.

Desde mi punto de vista, otra gran diferencia entre NEM y Nxt es el punto hasta que las características de "*blockchain* 2.0" están directamente integradas en la API. Por ejemplo, los activos (assets) en NEM, llamados "mosaicos", comparten varias de las características de las monedas del Sistema Monetario de Nxt, pero NEM no tiene integrado un *exchange* descentralizado para los mosaicos. (Como apunte, la Fundación NEM ha encargado a Blockchain Global la creación de una *exchange* centralizado tradicional para ofrecer *tokens* para ICO basados en mosaicos). Del mismo modo, mientras que se podría crear un mercados descentralizado sobre NEM dónde los usuarios podrían comprar y vender bienes y servicios, NEM no tiene este tipo de mercado integrado en su API del modo que Nxt lo tiene.

Finalmente, una diferencia sutil pero muy importante entre NEM y la mayoría de otras *blockchains*, incluyendo la de Nxt, es la manera en que maneja las transacciones multifirma. En lugar de permitir a cualquier cuenta generar transacciones multifirma, NEM presenta el concepto de *cuenta multifirma* y exige que todas las transacciones multifirma se originen desde esas cuentas. Cualquier cosignatario de la cuenta puede iniciar una transacción desde ella y la transacción solo se ejecutará si un número suficiente del resto de cosignatarios la aprueban.

En un principio, esto podría parecer una limitación, puesto que requiere una cuenta multifirma separada para juego de cosignatarios con los que un usuario quiere firmar pero tiene dos ventajas clave: la cuenta multifirma es una cuenta completa, capaz de recibir pagos, mensajes y mosaicos, por ejemplo. Además, se pueden añadir o eliminar cosignatarios, así que la custodia

de la multifirma se puede transferir. Es posible crear una cuenta multifirma “1 de 1”, es decir, una cuenta con un solo guardián que puede transferir la custodia a otro guardián, si así lo desea. De esta forma, las cuentas multifirma en NEM pueden actuar como contenedores transferibles para XEM, mosaicos y mensajes.

Una aplicación particular a destacar de este concepto es el servicio de notario integrado en NEM llamado Apostille. Con Apostille, el proceso de autenticación de un documento es el siguiente:

1. Establecer el *hash* y firmar el nombre del documento.
2. Crear una cuenta multifirma para el documento derivada de la firma resultante
3. Establecer el *hash* y firmar el contenido del documento.
4. Enviar un mensaje con el resultado a la cuenta mutilfirma del documento.

Hay que darse cuenta de el último paso también añade una marca de tiempo (timestamp) al documento, puesto que la transacción que transfiere el *hash* firmado del documento a la cuenta multifirma queda almacenada en la *blockchain.*

Como muestra de una posible aplicación de Apostille, los autores del white paper exponen un caso dónde el documento autenticado sería el de la propiedad de un vehículo. La propiedad del vehículo se puede transferir cambiando los cosignatarios en la cuenta multifirma que contiene la propiedad. Se pueden enviar mensajes describiendo trabajos de mantenimiento y reparaciones para almacenar el registro de servicio del vehículo. Mosaicos emitidos por gobiernos o aseguradoras podrían atestiguar el pago de tasas. En este sentido, la cuenta multifirma representa tanto al vehículo como el historial de todas las cuentas que han interaccionado con él.

De todas formas, ya es suficiente sobre NEM. Pasemos a Mijin.

Mijin

A grandes rasgos, Mijin es una versión de NEM que tres de los desarrolladores del núcleo de NEM y una compañía llamada Tech Bureau desarrollaron como una *blockchain* privada y permisionada. Como cualquier otra *blockchain* privada (y en contraste con NEM, que es público), una *blockchain* de Mijin es propiedad de una autoridad central (por ejemplo una compañía) y está controlada por una ella.

Este no es el lugar adecuado para entrar en discusiones sobre la utilidad de las *blockchains* privadas, pero puesto que tanto Mijin como Cataplut son partes importantes del ecosistema de NEM, permitidme la licencia. En mi opinión, cuanto más “privada” se hace una *blockchain* privada, menos útil resulta. Aunque puedo pensar en un caso de uso para las *blockchains* “de consorcios”, dónde un puñado de organizaciones independientes, que no confían necesariamente las unas en las otras, cooperan para asegurar la red frente a abusos por parte de cualquier otro miembro del grupo, tengo problemas para ver el valor de una *blockchain* controlada por una *única* autoridad. Desde mi punto de vista, una *blockchain* sin un consenso

que dependa de la no-confianza es básicamente una base de datos extremadamente ineficiente y lenta.

Sin embargo, se que hay mucha gente que no está de acuerdo conmigo, por lo que para lo que resta de artículo voy a asumir que las *blockchains* privadas tienen valor y existe un mercado para ellas, especialmente en los servicios financieros, los cuales parecen ser la principal industria a la que Tech Bureau pretende dar servicio utilizando Mijin.

No hay tanta información disponible en internet sobre Mijin como la hay sobre NEM, pero he descubierto algunos hechos interesantes que dan pistas sobre su potencial. Por un lado, aunque Mijin y NEM son proyectos completamente, Mijin usa la misma API que NEM (o, como mínimo, ambas ambas APIs presentan muchos aspectos comunes), lo que sugiere que sería relativamente sencillo para los desarrolladores escribir aplicaciones que funcionasen en cualquiera de las dos plataformas. Con una API común también se facilita la interacción entre las cadenas Mijin y la cadena pública de NEM, pero no he encontrado más información sobre los detalles de esta interacción.

Además, la web de Mijin afirma que Mijin soportará contratos inteligentes, aunque el white paper de Catapult parece contradecir esta afirmación cuando dice: “el enfoque aquí es hacer de los contratos inteligentes un componente externo, bien sea centralizado (es decir, tal y como se encuentra hoy en día los sistemas actuales) o descentralizado. Los resultados de estos contratos inteligentes introducirán su transacción en el libro de contabilidad a través de un proceso seguro”. Yo interpreto que los contratos en si mismo nunca serán almacenados en la *blockchain*ni serían ejecutados por todos los nodos de la red.

Y hablando de Catapult…

Catapult

Catapult es una reescritura de Mijin con la intención de incrementar el ratio de confirmación de las transacciones. A partir del *white paper* (enlazado más arriba), los primeros despliegues de Catapult serán a bancos y otras instituciones financieras, dónde el autor prevé que reemplazará a los entramados de “sistemas monolíticos inconexos”, que afirma se usan habitualmente hoy en día. A la larga, los desarrolladores también están planeando integrar Cataplut en NEM para facilitar el escalado de la *blockchain* pública.

Al igual que Mijin, Catapult es de código cerrado en estos momentos y no se han publicado muchos detalles técnicos. Sin embargo, fui capaz de encontrar información interesante rebuscando por el blog de NEM, especialmente en este hilo de uno de los desarrolladores. Catapult divide el trabajo que la red hace entre tres tipos de nodos:

- Nodos P2P, que añaden nuevos bloques a la *blockchain* y mantienen el consenso sobre su estado
- Nodos REST, que muestran las aplicaciones de cliente con todas las características clave que pueden usar de la API de Catapult

- Nodos API, que como los nodos P2P, almacenan la *blockchain* y pueden leer directamente de ella (creo), pero que no añaden bloques a la misma. Estos nodos sirven datos a los nodos REST para cumplir las solicitudes de las aplicaciones del cliente

Este despiece aparenta estar en consonancia con la arquitectura de tres niveles usada habitualmente para las aplicaciones web, dónde la *blockchain* (nodos P2P) son la base datos, los nodos REST son el *front-end* y los nodos API manejan la lógica de negocios de interpretar e interactuar con los datos de la base de datos.

Si esta analogía es correcta, entonces presumiblemente el objetivo de estar arquitectura es el de permitir a cada nivel escalar independientemente. Especialmente para una *blockchain* privada, el número óptimo de nodos P2P usados para establecer un consenso puede ser mucho más pequeño el que número de nodos REST y API requeridos para manejar todas las solicitudes que las aplicaciones envían a la red. El delegar estas responsabilidades a nodos separados de la red debería permitir que los nodos de cada tipo fuesen añadidos o eliminados según fuese necesario, para optimizar el rendimiento.

Aparte de esta nueva arquitectura, Catapult también realiza algunas optimizaciones para mejorar el rendimiento. Mientras que Mijin y NEM están reescritos en Java y usan HTTP para la comunicación con los nodos completos, Catapult está escrito en C++ y la comunicación al menos entre los nodos API y los nodos REST utiliza *sockets full-duplex* (a a través de ZeroMQ), permitiendo potencialmente una latencia inferior que el HTTP.

Un test de rendimiento de tres nodos Catapult situados en el mismo centro de datos y configurados para aceptar solicitudes de servicio provenientes de 10,8 millones de cuentas mostró que la red era capaz de un poco más de 3.000 transacciones por segundos. No queda completamente claro en la nota de prensa, pero parece como si cada uno de los tres nodos de este test jugasen todos los roles: P2P, API y REST. Para confundir todavía más las cosas, el diagrama que lo acompaña se refiere a los nodos API como “servidores de consumo de datos *blockchain*” y a los nodos RES como servidores de la “puerta API”.

Comparado con Ardor

Entonces ¿Cómo se compara NEM con Ardor?

Realmente, existen dos preguntas diferentes a formular, como mínimo: ¿cómo se comparan las características de NEM con las de Ardor? y ¿Cómo se enfrenta al tema de escalado NEM comparado con cómo lo hace Ardor?

Puesto que Ardor (la plataforma, no la cadena madre) permitirá usar todas las características actuales de Nxt, las comparaciones que he hecho más arriba entre NEM y Nxt aplican de igual modo a Ardor.

En particular, las *child chains* de Ardor tendrán a su disposición una gran variedad de tipos de transacciones integradas que admitirán un rico juego de características.

Por ejemplo, NEM no soporta de modo nativa un *exchange peer-to-peer* para los mosaicos, el pago de dividendos a los depositarios de mosaicos, transacciones condicionadas a la emisión de votos por parte de los depositarios de mosaicos (o la mayoría de los tipos de transacciones por fases de Nxt, por el mismo motivo), propiedades de cuenta, mercado descentralizado o nada que se parezca al *shuffling* o al sistema de alias integrado en Nxt.

La arquitectura de cadena madre / cadenas hijas añadirá también otras funciones extra.

En particular, los usuarios serán capaces de intercambiar diferentes *tokens* de *child chains* entre ellos directamente, sin tener que convertirlos previamente a ARDR. Esto será especialmente útil en el caso de las *child chains* ligadas a otros valores, dónde los usuarios podrán *tradear* con monedas ligadas al dólar directamente por monedas ligadas al bitcoin (por ejemplo), mientras que en NEM alguien que posea un mosaico ligado al dolar tendrá que venderlo por XEM para luego poder comprar un mosaico ligado al bitcoin.

A pesar de estas diferencias, NEM todavía ofrece un rico juego de características que los desarrolladores de aplicaciones pueden usar de maneras muy interesantes. Quizá el mejor ejemplo sea el uso creativo que hace Apostille de las características únicas multifirma de NEM. No estoy seguro de cómo sería replicar este tipo de funcionalidad en Ardor.

[EDITADO]: Lior Yaffe, desarrollador central y cofundador de Jelurida, nos hace llegar el siguiente comentario:

> *Con NXT esto se puede realizar usando los activos singleton para cada registro de licencias y enviándolo entre cuentas.*

Sobre el tema de cómo escalar, las diferencias son más destacadas.

En enfoque de Catapult, el cual NEM incorporará a largo plazo, consta de dos partes: una nueva arquitectura de tres niveles para distribuir las responsabilidades de la red entre tres nodos especializados; y una serie de optimizaciones a nivel de aplicación, por ejemplo el uso de C++ en lugar de Java. Tendremos que posponer nuestro veredicto hasta más adelante, cuando se hayan publicado resultados con los test de rendimiento, pero con la debida precaución todavía podemos lanzar unas especulaciones sobre las implicaciones de la nueva arquitectura.

La mayor ventaja parece ser para las *blockchains* privadas, dónde el usuario puede afinar las cantidades de los nodos de cada uno de los tres tipos. Además, en ese contexto, el crecimiento desmesurado de la *blockchain* no es un problema severo como lo es para las *blockchains* públicas puesto que las compañías pueden fácilmente destinar terabytes de almacenamiento en sus servidores para almacenar la *blockchain*.

La mejora del rendimiento de NEM con la nueva arquitectura, por otro lado, es más difícil de predecir. No está claro si cada par de la red tendrá que ejecutar los tres servicios (P2P, API, REST) o solo uno de los tres. En el primer caso, la ventaja para escalar de esta nueva arquitectura se perderá. En el segundo caso, el clásico equilibrio entre velocidad (menores nodos P2P, más nodos API y REST) y seguridad (mayor proporción de nodos P2P) se mantendría. Y puesto que nadie podría controlar el número de nodos de cada tipo en un red pública, la pregunta es que balance óptimo se debería buscar.

Por su lado, el diseño de Ardor intenta no obtener el máximo rendimiento, al menos inicialmente. En su lugar, el objetivo de escalado de Ardor busca reducir el tamaño y la tasa de crecimiento de la *blockchain*. Para ello utiliza una arquitectura única de cadena madre / cadenas hijas, dónde todos los nodos de la red validan todas las transacciones, pero sólo aquellos que pertenecen a cuentas que poseen la moneda de la cadena madre (ARDR) pueden forjar. Puesto que las monedas de las *child chains* no pueden forjar, el historia de transacciones de las *child chains* es irrelevante para la seguridad de la red, y puede podarse.

No obstante, vale la pena recordar que el escalado computacional está en la hoja de ruta de Ardor.

Concretamente, es posible que el procesado de las transacciones de las *child chains* sea delegado a diferentes subredes de la red Ardor en un futuro, permitiendo que la mayoría de nodos ignoren la mayoría de transacciones.

Conclusión

Tanto Ardor como NEM ofrecen un juego de características interesantes, que en muchos aspectos se solapan.

En general, mi impresión es que los desarrolladores muy probablemente serán capaces de construir aplicaciones igual de complejas en cualquiera de estas dos *blockchains*, con una facilidad comparable. En este sentido, ambas plataformas son competidoras directas.

En su enfoque sobre el escalado, Ardor y NEM son bastante diferentes.

Mientras que Catapult obtendrá una mejora significativa en el ratio de transacciones confirmadas por las *blockchains* privadas, soy más escéptico sobre la mejora del rendimiento que se podría alcanzar en una *blockchain* pública como la de NEM usando el mismo enfoque.

Por su lado, Ardor no intenta dirigir (por ahora) el tema del escalado computacional, pero ha encontrado una solución muy efectiva al problema del crecimiento desmesurado de la *blockchain*.

Supongo que el tiempo nos dirá si el escalado computacional o el crecimiento de la *blockchain*supondrá el mayor problema a largo plazo para la tecnología *blockchain.* El tiempo dirá también si alguna plataforma ha encontrado la solución adecuada.

IOTA

Ardor vs The Competition Pt 3

Esta semana he analizado IOTA, un libro de contabilidad distribuido que no utiliza *blockchain*.

¿Por qué Comparar Ardor con IOTA?

A primera vista, IOTA es tan diferente de Ardor como puede llegar a ser un libro de contabilidad distribuido. Utiliza un grafo acíclico dirigido (Directed Acyclic Graph – DAG), cuyos desarrolladores denominan el "Tangle", para representar el historial de transacciones, en lugar de almacenar las transacciones en una *blockchain*. La intención es que se use primordialmente para microtransacciones de máquina a máquina en el Internet de las Cosas (Internet of Things – IoT), una visión motivada por el hecho de que IOTA no tiene tasas por transacción. Y no soporta (todavía) las características de "*blockchain* 2.0" que forman parte principal del atractivo de Ardor. A simple vista, no parece que compita con Ardor.

Así que ¿Por qué incluir IOTA en esta serie titulada "Ardor Frente a la Competencia"?

Cómo he mencionado anteriormente, mi principal interés con esta serie es explorar la aproximación que realizan diversos libros de contabilidad distribuidos al escalado y aquí es dónde la comunidad de IOTA ha hecho algunas afirmaciones extraordinarias. Conforme he ido aprendiendo más sobre IOTA para entender mejor como escala, he llegado a la conclusión de que IOTA y Ardor ofrecen soluciones complementarias (o directamente opuestas) al problema del escalado:

Ardor reduce drásticamente el crecimiento desmesurado de la *blockchain* pero requiere que todos los nodos de la red se pongan de acuerdo sobre el estricto orden de las transacciones; IOTA, por su lado, consigue un resultado mayor relajando para ello las reglas un poco, permitiendo discrepancias temporales entre las transacciones, pero se enfrenta a un reto importante al afrontar el crecimiento del tangle. Estas dificultades, junto a lo que he aprendido sobre la seguridad del tangle, me parecieron lo suficientemente interesantes para dedicarle un artículo en esta serie.

Pero si, a pesar de todo, no estás convencido ¡Vuelve a visitarnos la semana que viene!

Tras este artículo, planeo cambiar mi objetivo desde la escalabilidad hacia las características y el ajuste al mercado. Stratis, Ark y Waves están en la agenda, aunque todavía no estoy seguro del orden en que serán publicadas.

El Tangle

Sin lugar a dudas, la característica más representativa de IOTA es el tangle.

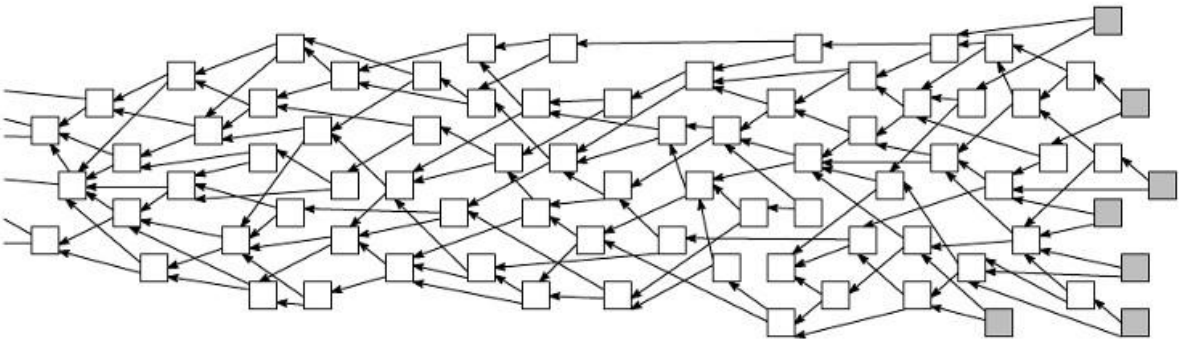
El resto de características únicas de IOTA, tales como la no existencia de tasas por transacción, el hecho de que las transacciones no siguen un orden estricto, aunque sigan siendo consistentes a largo plazo, y la noción de que (un poco de) spam realmente incrementa el rendimiento de la red, todas derivan del modo en que el tangle funciona.

Por esta razón, y también porque no quiero involucrarme en alguna de las controversias recientes entorno al proyecto IOTA, intentaré centrarme principalmente en entender y evaluar el tangle como conjunto, en lugar de diseccionar los detalles de su implementación específica en IOTA.

El tangle es un grafo acíclico dirigido cuyos vértices representan transacciones individuales y cuyos lados representan la “aprobación” de transacciones previas. Cada vez que un nodo transmite una nueva transacción a la red debe elegir dos transacciones previas para validarlas, a las cuales hace referencia en la nueva transacción que transmite. Conforme la nueva transacción penetra en la red, cada nodo la añade a su copia local del tangle, con un lateral apuntando a cada transacción que la nueva transacción aprobó.

He intento hacerlo lo mejor posible, pero esta descripción puede resultar confusa. Este diagramadebería ayudar. Cada cuadrado representa una transacción y las flechas que salen de cada transacción apuntando hacia otras dos representan la aprobación de las dos transacciones

previas. La transacción génesis se encontraría fuera del diagrama por su lado izquierdo, y las transacciones más nuevas, llamadas “puntas” (tips) en el white paper, se encuentra a la derecha, coloreadas de color gris.



¿Qué significa validar, y por tanto aprobar, una transacción? Conceptualmente, el nodo que hace la validación debe partir de las dos transacciones que está validando y seguir todos los caminos en orden inverso hasta la transacción del génesis, asegurándose de que nunca encuentra una contradicción (por ejemplo, un doble gasto, saldo insuficiente o cosas parecidas). Si hay una contradicción, escogerá otro par de transacciones para aprobar, sabiendo que ningún otro nodo aprobaría nunca la transacción que trata de transmitir si hubiese aprobado una serie de transacciones inconsistentes.

Hay que darse cuenta de que esto significa que cada nueva transacción no solamente aprueba cada una de las dos transacciones que ha escogido para ser validadas, sino que también indirectamente aprueba las transacciones que esas dos han aprobado, y las transacciones que esas transacciones aprobaron... y así todo el camino hasta el génesis. Esto es parte básica del “consenso a largo plazo” (eventual consensus) en el tangle.

En caso de que te estés preguntando acerca de la sobrecarga computacional para realizar esta validación, en la práctica se puede optimizar substancialmente. En los gráficos de esta páginapuedes observar que conforme caminas en el tangle desde las puntas (a la derecha del todo) hacia el génesis, a la larga alcanzas un punto en el pasado en el que todas las transacciones han sido (indirectamente) aprobadas por todas las puntas. En esos gráficos, las transacciones aprobadas por todas las puntas están coloreadas en verde. Por este motivo, se podría cortar el tangle a través de flechas que apunten a las transacciones en verde, validar los caminos desde esas transacciones particulares en verde hasta el génesis una sola vez, almacenar en cache los resultados y, desde ese momento en adelante, solamente validar tus nuevas transacciones hasta esas transacciones en verde. Esta optimización te evita el tiempo necesario para validar todo el tangle cada vez que quieras emitir una transacción y además permite que el tangle sea podado. Más detalle sobre esto a continuación.

Consenso

Una característica interesante de un libro de contabilidad basado en tangle como en IOTA es que los nodos que reciben las nuevas transacciones desde los pares *no tienen que validarlas inmediatamente*. De hecho, el tangle puede contener temporalmente transacciones contradictorias. Sin embargo, a la larga, un nodo debe decidir cual de las transacciones contradictorias debe aprobar (posiblemente indirectamente) al tener que añadir una nueva transacción.

¿Cómo elige entre transacciones en conflicto? Asumiendo que cada transacción sería válida si fuese considerada por separado, entonces la respuesta corta es que un nodo podría elegir aprobar cualquiera de ellas. Sin embargo, tiene un incentivo para aprobar aquella sobre la que el resto de la red continuará construyendo, para que igualmente su propia transacción sea aprobada a la larga. Se supone que la mayoría de los nodos de la red ejecutan el algoritmo de selección de transacciones para la aprobación, así que en caso de conflicto, un nodo tiene un incentivo para elegir la transacción que el algoritmo de referencia elegirá.

Para entender el algoritmo de referencia, es importante entender primero el concepto de peso acumulado (cumulative weight) de una transacción.

Cada nodo que emite una transacción tiene que hacer un poco de Prueba de Trabajo (Proof-of-Work – PoW), el cual determina el “peso propio” de la transacción. El peso acumulado de una transacción es entonces la suma de su peso propio con los pesos de las transacciones que han sido directa o indirectamente aprobadas. En un tangle genérico, el nodo puede decidir cuanto trabajo hacer para una transacción, pero en IOTA todas las transacciones requieren del mismo PoW y, por consiguiente, todas cuentan con el mismo peso. Cómo resultado, el peso acumulado de una transacción es proporcional al número de transacciones que aprueba directa o indirectamente.

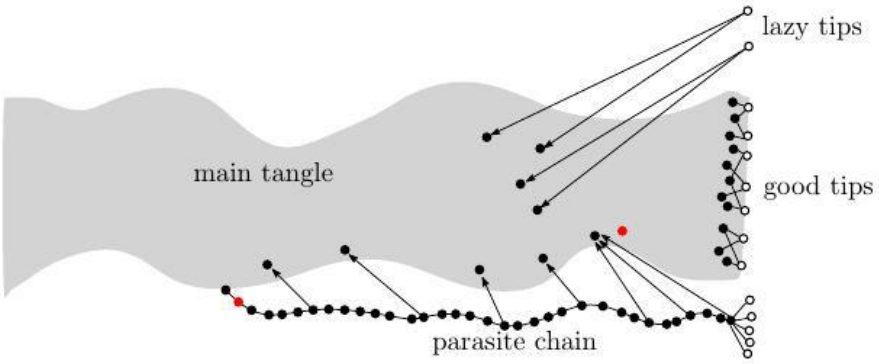
Entonces ¿Qué es el algoritmo de referencia? El autor del *white paper* lo llama Markov-Chain Monte Carlo (MCMC, ver sección 4.1), lo cual es una manera sofisticada de decir que es el viaje aleatorio por la tangle por aquellos caminos con mayor peso acumulado. Este artículo ya se está haciendo largo, así que evitaré los detalles. Es suficiente con decir que, cuando surgen transacciones en conflicto, el algoritmo MCMC resuelve el conflicto tendiendo a elegir aquella transacción que cuenta con el mayor peso acumulado tras ella. A la larga, un *subtangle* se convierte en dominante y el otro se queda huérfano. Esto proceso es análogo al que las *blockchains* usan para resolver *forks* y el peso acumulado de una transacción en IOTA es una media aproximada de su finalidad de la misma forma que al añadir bloques a una *blockchain*estamos confirmando transacciones cada vez con mayor certeza.

Por cierto, el hecho de que los nodos no necesiten validar inmediatamente cada nueva transacción recibida de sus pares tiene grandes implicaciones para el rendimiento. Cada nodo tiene que hacer menos trabajo de este modo, validando las transacciones solo cuando transmiten nuevas transacciones y tomando por seguro que las transacciones que han sido aprobadas indirectamente por todas las puntas han sido ya validadas por el resto de la red. Además, las validaciones corren en paralelo a través de la red, puesto que los diferentes nodos eligen diferentes subseries de transacciones para aprobar.

Seguridad

Hasta este momento, mayormente solo he regurgitado la información disponible en el *white paper* de IOTA. Por otro lado, el asunto de la seguridad del tangle es dónde la cosa se pone interesante. Recomiendo leer el análisis que se puede encontrar en el *white paper* sobre los diferentes posibles ataques sobre el tangle (también recomiendo leer el resto del whitepaper, porqué está muy bien redactado) y es por esto que no hablaré sobre la mayor parte de ese análisis aquí.

En su lugar, quiero centrarme en la amenaza más obvia, que es la de un ataque del 51%. Los desarrolladores de IOTA se refieren a ello como ataques del 34%, por unas razones que no estoy seguro de comprender. Creo que es porque si un atacante esperase hasta que suceda un *fork* de modo natural solo necesitaría un determinado poder de *hash* para sobrepasar el de los nodos en cada rama del *fork*, es decir, más de un 50% del poder de *hash* del *resto* de la red. De cualquier modo, el número exacto no es importante y para lo que queda de artículo haré referencia a él como “ataque del 34%”.



En IOTA, un ataque del 34% tendría la siguiente apariencia. Un atacante emite una transacción que supone el gasto de unos fondos, representado por el punto rojo de más a la derecha. A continuación, proceso (o quizá haya ya preprocesado) su propio subtangle “parásito”, el cual se

ancla al tangle principal en algún lugar aguas arriba de su transacción y que contendrá una transacción de doble gasto, representada por el punto rojo de más a la izquierda. Su objetivo es añadir el suficiente peso acumulado a su tangle parásito para convencer al algoritmo MCMC de que debe dejar huérfano el tangle principal y pasará a seguir el parásito.

Afortunadamente, las analogías con la *blockchain* son claras hasta ahora, pero todavía existe una más importante. Cómo en una *blockchain* PoW, el tangle está asegurado por el poder de *hash*actual de la red, puesto que este poder de *hash* es lo que añade peso acumulado al tangle legítimo. Sin embargo, al contrario de lo que sucede en una *blockchain* PoW, los nodos de IOTA sólo hacen PoW cuando emiten transacciones. Por tanto, la seguridad del tangle depende solo de la frecuencia de transacciones y la cantidad de PoW por transacción. Tómate un segundo para asimilar esta idea, porque es absolutamente crucial para entender la seguridad del tangle.

Puesto que la red de IOTA es todavía pequeña y el ratio de transacciones es baja, el equipo de IOTA ha establecido un único nodo de confianza, llamado el Coordinador, que es responsable en último lugar de decidir el estado actual del tangle. Su propósito es proteger la red frente a ataques del 34%, entre otros ataques. No voy a dedicarle más tiempo, pero os animo a que leáis esta crítica y la respuesta de los desarrolladores, y entonces saquéis vuestras propias conclusiones acerca de si IOTA puede llamarse descentralizado cuando funciona bajo la supervisión del Coordinador.

Vamos a ver si podemos obtener una estimación del orden de magnitud de lo segura que la red podría ser sin el Coordinador. En un reciente test de stress se alcanzaron cómodamente las 100 transacciones por segundo (tps) en una pequeña red de pruebas. El equipo sugiere que 1.000 tps por segundo serían factibles. No se cual es el requisito de PoW en IOTA, pero supongamos que el dispositivo IoT medio sea aproximadamente una Raspberry Pi que utiliza el 100% de su CPU durante 10 segundos para realizar el PoW requerido. De nuevo, estoy tratando de ser generoso; muchos dispositivos IoT son considerablemente menos poderosos que una Raspberry Pi, y pedir el máximo de la CPU durante 10 segundos para cada transacción probablemente sería un quebradero de cabeza.

Con estas premisas, podemos concluir que el poder de computación medio que asegurará la red es aproximadamente 10.000 x (el número de computaciones de Raspberry Pi en 10s) por segundo, o su equivalente, 100.000 veces el poder de computación de una Raspberry Pi. Hay múltiples matices posibles sobre el *benchmark* de los dispositivos, pero no nos vamos a preocupar por variaciones de factor dos o tres, ya que sólo pretendemos obtener una estimación del orden de magnitud, así que usaré algunos números gordos que he encontrado en internet.

Una Raspberry Pi3 puede realizar cientos de MFLOPS (megaflops, o millones de operaciones de coma flotante por segundo), mientras que los relojes de las GPU's de última generación pueden realizar miles de GFLOPS (gigaflops, o billones de FLOPS), 10.000 veces mayor poder de computación. Así que en nuestro caso hipotético, un atacante con ~10 GPUs podría sobrepasar el poder de cómputo de toda la red. Añádele otro factor de 10 porque haya sido descuidado en mis cálculos (quizá las operaciones con enteros son más lentas en una GPU que las operaciones

como flotante, por ejemplo) y todavía necesitarías solamente 100 GPUs para ejecutar los ataques.

Estoy seguro de que hay muchas cosas que puntualizar en este análisis. Quizá IOTA no se ejecutará continuamente en el extremo de la red, por ejemplo. En su lugar puede que se ejecute en routers y puertas de enlace a la que estos dispositivos IoT se conecten, los cuales suelen ser mucho más potentes.

De todos modos, lo que pretende decir es que PoW asegura con éxito *blockchains* como la de Bitcoin y Ethereum porque no está atado a la tasa de transacciones o a cualquier otro factor más allá del valor económico de la red. Conforme el valor de la recompensa por minado (en dinero fiat) aumenta según aumenta el valor de Bitcoin, los mineros añaden más hardware y consumen más energía para minarlo. El incentivo económico para minar asegura que la cantidad de poder de *hash* que asegura la red aumenta con el valor monetario de la red.

Por contra, en IOTA no hay incentivo económico para asegurar la red. Además, el poder de *hash*que asegura la red está ligado directamente al número de transacciones, el cual naturalmente tiene un límite superior que depende del ancho de banda y de la tipología de la red.

Sobre este último punto, los desarrolladores de IOTA han lanzado un argumento creativo, no mencionado en el *whitepaper*, que afirma que las limitaciones y la tipología de la red realmente mejoran la seguridad de la red. No he encontrado ningún argumento oficial sobre ello en ningún sitio, pero tras investigar un poco me encontré con esta conversación en Slack, que es el mayor argumento que he podido encontrar.

Esencialmente, uno de los desarrolladores de IOTA (en concreto Sergey Ivancheglo, alias Come-from-Beyond, posiblemente alias BCNext, uno de los creadores originales de Nxt), afirma que la red de IOTA consistirá en dispositivos IoT que se emparejarán exclusivamente con sus vecinos más próximos en una tipología de red en malla, en la que un atacante solo tendrá la posibilidad de emparejarse con un pequeño número de dispositivos en esa red en malla. Es decir, la gran mayoría de dispositivos no estarán disponibles desde internet u otra red de acceso y la única manera de enviarles mensajes sería a través de la malla del resto de dispositivos.

La idea general es que la malla como entidad será capaz de alcanzar un gran rendimiento, pero cada *link* individual en la malla tiene un ancho de banda lo suficiente bajo que haría que un atacante lo saturase al tratar de añadir el número de transacciones suficientes para que convencer a la red de que siguiese su subtangle parásito. Puesto que el atacante solo tiene unos pocos puntos de entrada a la malla, los saturará antes de que su tangle parásito acumule suficiente peso como para que su ataque sea fructífero.

Dejaré que seáis vosotros los que saquéis vuestras propias conclusiones sobre este argumento. Personalmente no creo que el equipo de IOTA haya publicado los suficientes detalles como para poder evaluarlo en profundidad.

Ya que estamos hablando de las limitaciones del ancho de bando, pasemos a hablar del escalado.

Escalabilidad

Puesto que cada nodo debe validar otras dos transacciones antes de poder emitir su propia transacción, al equipo de IOTA le gusta señalar que el *spam* realmente tiende a hacer la red más eficiente. Otros miembros de la comunidad IOTA se dejan llevar por este argumento, algunos veces hasta llegar a afirmar absurdamente que IOTA es "infinitamente escalable."

Cada nodo de la red de IOTA debe, a largo plazo, recibir cada transacción para mantener un tangle mínimamente consistente. Sin embargo, transmitir transacciones a los nodos remotos requiere de tiempo y si el ratio de transacciones es lo suficientemente alto, de modo que un nodo recibe muchas transacciones de sus nodos cercanos antes de que reciba la próxima transacción desde los nodos lejanos, el algoritmo MCMC continuará seleccionando las puntas enviadas por los nodos cercanos. A la larga, el tangle se divide, con solo los nodos cercanos realizando transacciones en la copia local del tangle y los nodos lejanos realizando transacciones entre ellos, una copia divergente.

Así que el ancho de banda y la tipología de la red tiene que establecer ciertas limitaciones en el ratio de transacciones de IOTA si el tangle tiene que ser consistente a lo largo de toda la red. Tendremos que esperar a que se completen más test de estrés para descubrir cuales son las limitaciones.

Adicionalmente, como todos los libros de contabilidad distribuidos, IOTA tiene que lidiar con el problema del crecimiento desmesurado. Cada transacción de IOTA ocupa unos 1,6kB, así que un tasa de transacciones de 100tps hará crecer el tangle a un ritmo de 160 kB por segundo, o unos 14GB al día. No es necesario decir que eso es un valor de almacenamiento irreal para un dispositivo IoT.

IOTA soluciona este problema tomando capturas periódicas del tangle, que mapea su estado actual en una nueva transacción génesis, permitiendo que el historial de transacciones sea eliminado. En el supuesto límite de un podado muy frecuente, un nodo solo tendría que almacenar una parte suficiente del tangle para ser capaz de ejecutar el algoritmo MCMC.
Sin embargo, sincronizar un nuevo nodo con la red es una historia diferente. O bien el nodo descarga la última captura desde un par de confianza, o bien arranca desde la transacción génesis original y busca su camino hacia delante en todo el tangle. No hay manera de unirse de manera eficiente a la red sin depender de la confianza en un tercero.

Finalmente, hay que mencionar que el equipo de IOTA ha propuesto un tipo de partición horizontal del tangle al que llaman "manada" (swarm) dónde un grupo de nodos unidos almacenan todo el tangle, pero ninguno de ellos lo almacena en su totalidad. Desafortunadamente, todavía no contamos con muchos detalles sobre cual es su funcionamiento.

Comparado con Ardor

¿Qué tiene todo esto que ver con Ardor?

En mi opinión, hay dos comparaciones que establecer, concretamente en los aspectos de la seguridad y la escalabilidad.

Sobre la **seguridad**, no está claro para mí que IOTA pueda alcanzar un ratio de transacciones suficientemente alto para ser considerado seguro sin el Coordinador, debido al valor monetario de la red actual, sin elegir una exigencia muy alta de PoW.

Ardor, por su parte, tiene la ventaja de que sus *child chains* están aseguradas por una única cadena madre.

Una "pequeña" *child chain* no requerirá de un nodo de confianza como es el Coordinador de IOTA para protegerla, porque su consenso es establecido por la red al completo y almacenado (a través de los *hashes* de los bloques de las *child chains*) gracias a los forjadores en la cadena madre.

Sobre la **escalabilidad**, tanto IOTA como Ardor comparten el requisito de que cada nodo de la red debe procesar todas las transacciones. Con IOTA, esto simplemente significa añadir transacciones al tangle, lo que es computacionalmente barato, mientras que con Ardor, cada nodo debe validar cada transacción. Además, el diseño inteligente del tangle asegura que el tiempo de confirmación por una transacción realmente disminuye conforme la red tiene más trabajo. No me sorprendería que IOTA alcanzase un mayor rendimiento que Ardor si ambas redes crecen.

Por otro lado, IOTA se enfrenta a un tremendo reto para combatir los problemas de crecimiento desmesurado del tangle si alguna vez tiene que alcanzar cientos de transacciones por segundo, mientras que Ardor ya ha solucionado este problema.

Finalmente, hay que mencionar que la hoja de ruta de Ardor delegará el procesado de las transacciones de la *child chains* a sub-redes dedicadas dentro de la red. Esto podría potencialmente alcanzar una mejora computacional similar a la propuesta de "manada" de IOTA, posiblemente permitiendo un rendimiento similar.

Reflexiones Finales

Si has llegado a leer hasta aquí (¡gracias!) y ya estabas familiarizado con IOTA, entonces sin duda te habrás dado cuenta de que he dejado fuera multitud de detalles, incluyendo su algoritmo de *hash* casero, la falla introducida deliberadamente en el código por Come-from-Beyond como sistema anti-copia, el uso de codificación ternaria y el misterioso procesador Jinn que ofrecerá soporte hardware para IOTA en los dispositivos IoT. En el transcurso de mi investigación, me he formado unas posiciones bastante fuertes al respecto sobre estos temas, pero dudaba acerca de compartirlas aquí por dos motivos.

Primero, no tengo suficiente información para hacer afirmaciones objetivas sobre estos temas. No soy un criptógrafo y no conozco casi nada sobre la computación ternario o Jinn. Lo mejor que podría hacer es ofrecer afirmaciones subjetivas sobre las decisiones de diseño que el equipo de IOTA había hecho, pero simultáneamente habría debilitado el foco principal de este artículo y habría abierto la puerta a criticas provenientes de otras personas que pudieran tener distintas percepciones subjetivas.

Segundo, y más importante, estoy más interesado en los conceptos fundamentales tras el tangle que en la implementación específica que IOTA hace de él. Sin importar si IOTA triunfa o fracasa, el tangle es una idea bella y se merece toda la atención que le podamos dedicarle.

Entonces ¿qué podemos decir sobre el tangle? Aunque me enamora lo elegante de su diseño y las sutilezas de su mecanismo de consenso, al final me temo que soy bastante escéptico de su usabilidad para el Internet of Things. Si quitas este aspecto, incrementa los requisitos de PoW varias órdenes de magnitud y encuentra un camino para ligar el umbral del PoW al valor monetario de la red sin evitar el acceso de los usuarios comunes a sus fondos y entonces creo que el tangle tiene un enorme potencial como libro de contabilidad distribuido.

La última pieza a encajar es como combinar la no necesidad de confianza, eficiencia y crecimiento desmesurado, un problema que Ardor ha solucionado extremadamente bien. Quizá alguien encontrará una manera de combinar los mejores elementos de ambos diseños en algún punto en el futuro. Un montón de cosas pueden suceder hasta entonces, especialmente en criptolandia.

Waves

Ardor Frente a la Competencia, Parte 4

Hasta este momento, uno de mis principales objetivos en esta serie ha sido analizar las diferentes enfoques para permitir escalara a la tecnología del Libro de Cuentas Distribuido. Sin embargo, esta semana y las dos próximas entregas me centraré en la vertiente empresarial de la tecnología *blockchain*. Voy a intentar explorar los problemas del mundo real que las *blockchains* permiten solucionar y la manera en que diferentes proyectos *blockchain* se han posicionado para encajar en este objetivo de mercado.

Estos temas se salen un poco de mi zona de confort, así que os agradezco vuestra paciencia en caso de que diga algo incorrecto o ingenuo. Y, cómo siempre, agradeceré vuestras críticas constructivas :)

El anterior descargo de responsabilidad es especialmente importante esta semana, porque esta semana he analizado Waves. Como recién llegado a Nxt que soy, he leído lo suficiente sobre su historia para saber que el fundador de Waves, Sasha Ivanov (alias Coinomat en nxtforum.org), había sido un miembro activo de la comunidad Nxt hasta el periodo turbulento que tuvo lugar a comienzos de 2016, momento en que se fue para crear Waves. No voy a intentar relanzar el debate sobre Ardor y el futuro de Nxt, el cual entiendo que finalizó cuando muchos emisores de activos como Sasha dejaron la comunidad, pero en caso de que estéis interesado, os

recomendaría leer el resumen que hizo apenzl en SNAPSHOT y las referencias que allí se pueden encontrar.

En cambio, en este artículo voy a ignorar en la medida de lo posible la historia entre Nxt y Waves y voy a aproximarme a ambos proyectos con la mente abierta y una visión de futuro. Creo que habría cierto valor en un análisis histórico detallado, pero simplemente no estoy capacitado para ello.

Una vez dicho esto, vamos a hablar sobre Waves.

Waves

En un primer vistazo, Waves aparenta ser una versión *básica* de Nxt. Es básicamente un *exchange*descentralizado (DEX), inspirado y conceptualmente similar al Intercambio de Activos de Nxt (Nxt Asset Exchange). Al igual que Nxt, usa el algoritmo de Prueba de Participación (Proof-of-stake) para el consenso y permite a los usuarios ceder su saldo a otras cuentas para forjar en grupo (*forging pools*). Recientemente ha añadido un modo para asociar un alias inteligible por humanos a un número de cuenta, replicando parcialmente la funcionalidad del Sistema de Alias de Nxt (Nxt's Alias System). Incluso un par de características que se encuentran todavía en desarrollo (un sistema de votación y un modo para enviar mensajes encriptados) duplican funciones que Nxt ya ofrece.

Al mismo tiempo, Waves carece de muchas de las características más poderosas con las que cuenta Nxt. Por el momento, no cuenta con nada similar a las transacciones por fases (phased transactions) u opciones para el control de cuenta, por ejemplo, aunque hay que indicar que tanto los contratos inteligentes como la multifirma de transacciones están en la agenda.

Adicionalmente, el white paper sugiere que el micromecenazgo (*crowdfunding*) será uno de los primeros usos para la plataforma Waves, aunque los *tokens* en Waves carecen de las propiedades de personalización que convierten al Sistema Monetario de Nxt en algo muy útil para estos usos. Por ejemplo, el Sistema Monetario ofrece la posibilidad de transferir los fondos cuando se ha alcanzado el objetivo de financiación, al estilo Kickstarter, y también la posibilidad de restringir el *trading* para evitar que los especuladores creen un mercado secundario. Al usar esta última característica, llamada moneda "Controlable" en la terminología de Nxt, es incluso posible que los emisores establezcan tanto un precio fijo de compra cómo de venta, permitiéndoles ofrecer a los compradores un reembolso parcial por sus *tokens*. Por contra, el *crowdfunding* en Waves está limitado a la emisión de un *token* a precio de mercado.

No obstante, en mi opinión sería un grave error descartar Waves o cualquier otra copia de Nxt que contase con menores características. Por una sencilla razón, Waves ofrece varias características clave que Nxt y otras plataformas no tienen, las cuales describiré a continuación. Quizá el más importante incluso sea que el equipo de Waves ha creado una marca sólida y ha ofrecido una visión clara y consistente desde el arranque de la plataforma. El campo está actualmente tan saturado, y las innovaciones se producen a tanta velocidad, que la combinación de un mensaje sencillo y claro con un fuerte esfuerzo en marketing y una habilidad

demostrada para entregar lo que prometen puede llegar a ser más importante para el éxito a largo plazo que las muchas o potentes novedades que pueda contener su tecnología subyacente.

Características Únicas

Una característica interesante que diferencia a Waves de otras muchas plataformas es el diseño de su DEX. Es una aproximación híbrida que combina un mecanismo centralizado de concordancia de órdenes de compra y venta, llamado *the Matcher*, con asientos descentralizados en la *blockchain* de Waves.

Cuando los usuarios ponen sus órdenes en Waves, el cliente de Waves envía estas órdenes a los nodos *Matcher*, quienes conservan el libro de órdenes para todos los pares comercializados. Cada nueva orden, o bien se empareja frente a una orden existente, o bien es añadida al libro de órdenes para el par en cuestión, pero en cualquier caso el usuario que emitió la nueva orden es notificado de inmediato sobre si su orden se completó. Todavía se necesita esperar hasta el siguiente bloque (o bloques) en la *blockchain* para que la transacción quede completamente confirmada, pero durante ese tiempo el usuario sabe con un alto grado de certeza cual ha sido el resultado de su orden.

Esto puede no parecer como una gran mejora respecto a un *exchange* completamente descentralizado, pero a partir de un puñado de transacciones que he ejecutado en Waves he de decir que la experiencia de usuario me ha impresionado. La posibilidad de ver un libro de órdenes con actualizaciones en tiempo real y saber inmediatamente si mis órdenes han sido ejecutadas supone una diferencia mayor de la que esperaba.

En principio, cualquier nodo completo puede convertirse en un *Matcher*. No obstante, el cliente ligero actualmente solo se conecta por defecto a los *Matchers* en nodes.wavesnodes.com, así que es posible que los *Matchers* del resto de la red no vean mucho volumen. Con las nuevas órdenes siendo transmitidas directamente a estos nodos centralizados, siendo sólo transmitidas al resto de la red en caso de que sean ejecutadas (creo), este diseño permite que el libro de órdenes sea anónimo. No se a ciencia cierta lo importante que es que un libro de órdenes sea anónimo, pero aparenta ser una característica que los *traders* pueden valorar mucho.

Otra característica diferenciadora de Waves es la habilidad de *tradear* cualquier *token* frente a cualquier otro *token* sin necesidad de convertirlo previamente a WAVES. En combinación con las pasarelas (*gateways*) integradas que emiten *tokens* ligados al dólar, euro u otras criptomonedas, esta característica permite que Waves funcione como un mercado de intercambio internacional y descentralizado. También permite que los emisores de activos lancen sus ofertas iniciales directamente en *tokens* ligados a monedas fiat. Con el cliente completo, es incluso posible pagar las tasas en *tokens* en lugar de con WAVES.

Adicionalmente, no hay que pasar por alto las otras características que están en desarrollo o en la hoja de ruta, que también diferencia a Waves de otras plataformas. Una de ellas es el sistema

de reputación que puntuará a las cuentas en función de su antigüedad, historial de transacciones y otros factores. Todavía no se han desvelado muchos detalles, pero el objetivo es ofrecer a los usuarios al menos una indicación a aproximada sobre la confiabilidad de un emisor de activos. El *whitepaper* va más allá al sugerir que el sistema de reputación actuará como "una forma de KYC/AML descentralizado" (normativas anti-lavado de dinero y de identificación del cliente). Aunque es difícil ver cómo un sistema de reputación descentralizado podría ayudar a los emisores a cumplir con las normativas KYC/AML, no es descabellado suponer que pudiese funcionar como un sistema análogo para la comunidad *blockchain*.

Hablando del cumplimiento de requisitos legales, Waves ha anunciado un nuevo proyecto, Tokenomica, que ofrecerá una "estructura 100% legal para diferentes tipos de *crowdsales* de tokens, incluyendo *crowdsales* de capital riesgo". Desafortunadamente, esa cita extraída de la hoja de ruta de 2017 es toda la información que he sido capaz de encontrar sobre Tokenomica. Mi impresión es que el proyecto todavía está en sus fases iniciales, pero muestra que el equipo se está tomando el cumplimiento de las normativas seriamente.

Para terminar, debería mencionar que el equipo de Waves también está planeando incorporar contratos inteligentes (*smart contracts*) en Waves. El lenguaje de *scripting* no será completamente compatible con Turing y no habrá equivalente al concepto de gas de Ethereum, probablemente porque no habrá bucles. Más allá de estos detalles, no existe todavía mucha más información disponible.

Finalmente, debo mencionar la propuesta que el equipo de Waves ha esbozado para escalar. Está constituida esencialmente por dos partes: un rediseño del proceso de forjado que trocea los grandes bloques en "microbloques" para optimizar la utilización del ancho de banda; y una optimización de la forma en que los balances de cuenta son almacenados (o, mejor dicho, no almacenados), de manera que se reducen los requerimientos de memoria para los nodos completos.

La primera de estas dos propuestas, llamada Waves NG, está basada en Bitcoin NG. Básicamente, una vez que un nodo se ha ganado el derecho de forjar el siguiente bloque, este emite inmediatamente un bloque clave (*key block*), que normalmente está vacío, y entonces transmite los microbloques que contienen las transacciones cada pocos segundos. La razón de este diseño es que, al transmitir un gran bloque, el intervalo de bloques es una manera mucho menos eficiente de utilizar el ancho de banda de la red, y los picos correspondientes en la actividad de la red establecen un mínimo inferior en el número de transacciones que la red puede manejar. Al propagar transacciones a través de una secuencia de microbloques, es posible incrementar la tasa media de datos en la red disminuyendo la tasa máxima de datos, reduciendo el efecto que el ancho de banda y la latencia imponen en el ratio máximo de transacciones.

El segundo componente del plan de escalado es implementar las ideas descritas en este artículode Leonid Reyzin, Dmitry Meshkov, Alexander Chepurnoy y Sasha Ivanov. Admito que no le he dedicado mucho tiempo, pero la idea es que no todos los nodos completos necesitarán almacenar en su memoria los saldos de todos los *tokens* para validar las transacciones. En su lugar, almacenarán un resumen compactado de esta información y aquellos forjadores que no

lo almacenen en su totalidad (o una subserie de ellos, si deciden forjar las transacciones de tan solo un *token* específico) generarán pruebas criptográficas de que han actualizado los saldos de cuenta correctamente. Los forjadores incluirán a continuación las pruebas y un resumen actualizado en la cabecera de cada bloque nuevo. Aquellos nodos que hayan elegido no almacenar los saldos de todos los *tokens* involucrados en esas transacciones todavía podrán validarlas usando el resumen actual y las pruebas de los forjadores para calcular y actualizar los resúmenes, la cual podrán comparar frente a la que que anunció el forjador.

Los autores afirman que este método puede dividir por cuatro la cantidad de memoria exigida en condiciones reales por un nodo completo. Además, si esta optimización es capaz de mantener toda la información requerida en memoria en aquellos casos en los que, de otra manera, debería haberse almacenado en el disco, la mejora del rendimiento podría ser mayor, alrededor de 20 veces mejor, según sugiere el autor.

Comparación con Ardor

Aunque un par de las características descritas no estaban disponibles en Nxt, encontramos características similares en Ardor.

Concretamente, la arquitectura de cadena madre / cadena hija (*parent-chain / child-chain*) de Ardor permitirá a los usuarios *tradear* cualquier para de monedas de cualquier *child chain*, algunas de las cuales podrían estar ligadas a monedas fiat u otras criptomonedas. También será posible establecer el precio de los activos en cualquier moneda de las *child chains* y pagar las tasas en la moneda de una *child chain* cuando la transacción sea realizada en una *child chain*determinada. No será posible *tradear* activos (*assets*) entre ellos directamente, aunque seguramente muchos de estos pares tendrían tan poco volumen que no valdría la pena añadir esta característica.

Sobre las mejoras que el equipo de Waves ha hecho con su DEX al centralizarlo parcialmente, sería posible imitar esta funcionalidad construyendo un coordinador de transacciones centralizado sobre Nxt/Ardor. De hecho, el proyecto InstantDEX consiguió algo similar en el pasado, usando Nxt para fijar las transacciones de una manera descentralizada.

Respecto al escalado, la propuesta para reducir los requisitos de almacenamiento en memoria para los nodos completos es fascinante, pero me pregunto si se reducirá la seguridad (si habéis leído algún artículo anterior de esta serie, probablemente ya os habréis dado cuenta de que siempre sospecho que las mejoras en rendimiento acarrearán una merma de la seguridad). Concretamente, si los nodos no tienen que almacenar el estado completo de cada cuenta y deben basarse en las pruebas y resúmenes en la cabecera de cada bloque para validar las transacciones que los contienen, entonces asumo que esto significa que los nodos no serán requeridos, ni siquiera serán capaces, de validar las transacciones sin confirmar antes de transmitirlas a los pares. No se que consecuencias puede traer el permitir a los nodos propagar transacciones potencialmente inválidas a la red, pero sólo el pensarlo me deja intranquilo.

El enfoque de Ardor es que todos los nodos validen todas las transacciones, pero que sólo una cantidad mínima necesaria de información sea almacenada permanentemente en el *blockchain*de Ardor. Concretamente, sólo aquellas transacciones que modifiquen los saldos de ARDR, el *token* de forja, necesitan ser almacenadas en la *blockchain* para que otros nodos puedan verificar, sin tener que depositar la confianza en un tercero, que cada bloque fue forjado por una cuenta que realmente podía hacerlo. Por su lado, todo el historial de transacciones de las monedas de cada *child chain* y los activos y monedas comercializadas en esas *child chains* no necesita ser almacenado en la cadena de bloques, y por tanto se puede podar, dejando tras de sí tan solo sólo los *hashes* criptográficos de esa información. El resultado es que que el tamaño de la *blockchain* permanece reducido y crece más lentamente que si se almacenase toda esa información extra.

Saber qué enfoque es mejor dependerá de si el almacenamiento permanente en la *blockchain* o el almacenamiento en memoria de los saldos de las cuentas representa un problema mayor conforme las *blockchains* crecen. No se como responder a esta pregunta pero hay un par de puntos relacionados que habría que remarcar. Uno es que la escala de tiempo de ambos problemas podría ser bastante diferente: Puedo prever una explosión de activos en la plataforma Ardor que podrían ocasionar un estrés en la memoria, mientras que el problema de la hinchazón de la *blockchain* sería un serio problema a largo plazo para Waves, especialmente si alcanza cientos o miles de transacciones por segundo, que es el objetivo actualmente fijado. Mi otro pensamiento es que Ardor ha requerido de una arquitectura completamente nueva para implementar su solución de escalabilidad, mientras que el enfoque de Waves no. Sin duda, sería más fácil para Ardor incorporar en algún momento la solución aportada por Waves que no que Waves implementara la solución de Ardor.

Finalmente, quizá el tema más interesante de esta comparación es el problema del cumplimiento con las regulaciones. Waves se ha posicionado a si misma como la plataforma para crear y emitir *tokens*, con una atención especial para el *crowdfunding*. Tanto es así que el equipo de Waves está mirando con detenimiento todas las normativas que afectan a las campañas de micro mecenazgo (que podría afectar al hecho de vender acciones, por ejemplo)

para ayudar a los usuarios a que cumplan con la ley. Mientras que la afirmación de que un sistema de reputación descentralizada pudiese reemplazar a los tradicionales requisitos de cumplimiento con la normativa KYC/AML cuesta de creer, sí que podría al menos ayudar a suprimir las estafas y a reducir las oportunidades para que actores malintencionados se aprovechen de otros. En ese sentido, podría alcanzar algunos de los objetivos que los reguladores pretenden alcanzar.

Ardor, por su parte, ofrecerá un par de mejoras sobre Nxt que pueden ser bastante valiosas para el cumplimiento de la normativa. Una de ellas es la emisión de activos que sólo podrán comercializarse bajo un tipo determinado de transacciones por fases. La otra es la inclusión de las transacciones por fases, que permite a una cuenta aprobar una transacción sólo si la cuenta tiene una propiedad específica. Al combinar estas dos características, un usuario puede emitir un activo que sólo se puede comprar por las cuentas que tienen una propiedad que, por ejemplo, haya sido asignada por un proveedor de servicios de KYC/AML para indicar que se ha verificado la identidad del propietario.

Si tu activo representa acciones de una compañía, o un fondo de inversión u otro tipo de acción, esta característica te permitirá demostrar ante los reguladores que conoces al comprador de tus *tokens*. Además, si eres un usuario interesado en comprar este tipo de activo, almacenar una prueba de tu identidad en la *blockchain* usando las propiedades de cuenta posiblemente te permitirá invertir menos tiempo tratando de convencer a los negocios de que tú eres quien dices ser y que no estás lavando dinero.

Además, será posible crear *child chains* que solo permitirán una subsecuencia de las características que la plataforma Ardor puede ofrecer. Esto permitirá a los creadores de la *child chain* desactivar ciertas características, como la del mezclado de monedas (*coin shuffling*), que podría hacer disparar las alarmas de algunos reguladores en determinadas jurisdicciones.

Conclusión

¿Qué hacemos entonces con Waves? Necesariamente debemos destacar el hecho de tomar un problema y tratar solucionarlo mejor que ningún otro antes. No hay duda de que abandonar el planteamiento de “Navaja suiza” que supone Nxt para centrarse en su lugar en el único objetivo de crear una gran plataforma para la comercialización de *tokens* hizo más sencillo lanzar, desarrollar y sacar al mercado Waves. También influye mucho el contar con una buena financiación, puesto que Waves recaudó $16M en su ICO.

Sin embargo, al mismo tiempo, no estoy seguro de que al realizar una comparación objetiva entre Waves y Ardor podamos concluir que Waves está tan maduro tecnológicamente como Ardor. (Por cierto, he tratado de realizar una comparación justa y objetiva en este artículo, pero no estoy diciendo que lo haya conseguido. Eso es algo que tú tendrás que decidir). Nxt ya es capaz de realizar todo lo que Waves hace, y eso si mencionar todas las cosas que Waves no puede hacer, a las cuales además Ardor también añade nuevas funcionalidades.

Quizá el último gran reto que queda para Ardor es realmente vender su visión de la manera en que la comunidad Bitcoin y la Fundación Ethereum han vendido sus visiones, y aquí es dónde Waves lleva ventaja. Siendo capaz de tantas cosas diferentes, pero no habiendo sido construido específicamente para ninguna en particular, Ardor tiene una ardua tarea por delante. La peor salida posible sería que tanto usuarios como negocios lo viesen como "simplemente otra plataforma *blockchain*", o quizá no consiguiesen comprender todo el rango de cosas que puede llegar a hacer, y terminasen por ignorarlo por este motivo.

Respecto a Waves, estoy ansioso por ver lo que el futuro depara. Las mejoras que ha hecho sobre el Intercambio de Activos de Nxt, aunque modestas en mi opinión, la hacen destacar como un DEX formidable. Si el equipo de Waves puede cumplir con su hoja de ruta, Waves será un serio competidor para los *exchanges*, tanto los centralizados como los descentralizados.

Stratis

Ardor Frente a la Competencia, Parte 5

Esta semana he analizado Stratis, una plataforma *blokchain-como-servicio* basada en el protocolo Bitcoin.

Stratis

El objetivo del proyecto Stratis es permitir a las empresas crear sus propias *blockchains* a medida, partiendo de una serie de funciones predefinidas. Además, el Grupo Stratis, el cual dirige el desarrollo de Stratis, ofrecerá servicios de consultoría para ayudar a los negocios a encontrar formas de usar la tecnología *blockchain* de manera efectiva, a la vez que, presumiblemente, también les ayudará a desplegar blockchains a medida en la plataforma Stratis.

Presentado de esta manera, Stratis aparenta ser muy parecido a Ardor. Pero la observar la mayoría de lo pormenores (al menos con los detalles disponibles públicamente sobre Stratis) las dos plataformas presentan diferencias significativas. En breve las exponemos.

Actualmente, la plataforma Stratis está compuesta de varias partes:

- NBitcoin, una implementación integral de Bitcoin en C# inspirada en Bitcoin Core;
- NStratis, un *fork* de NBitcoin que incorpora un algoritmo de minado de Prueba de Participación y un algoritmo alternativo de Prueba de Trabajo;
- el Nodo completo Bitcoin de Stratis, que puede funcionar tanto en la red de Bitcoin como en la de Stratis, que sirve de base para el resto de la plataforma;
- el Monedero Breeze, un monedero con verificación simplificada de pagos (*simplified payment verification – SPV*) tanto para Bitcoin como Stratis que implementa TumbleBit para convertir en privadas las transacciones; y
- el módulo de Identidad de Stratis, que permite a terceras partes comprobar la identidad de la persona que controla una cuenta de Stratis.

Hay que apuntar que la mayoría de estos componentes todavía están en fase alfa.

En la lista hay que destacar especialmente la integración de TumbleBit en el Monedero Breeze. El Documento Técnico de TumbleBit es bastante denso; si estás interesado en los detalles, recomendaría leer en su lugar esta excelente presentación elaborada por dos de los autores. En pocas palabras, TumbleBit utiliza canales de pago unidireccionales para transferir los fondos desde una serie de pagadores hacia un intermediario llamado Tumbler, y desde el Tumbler a una serie de receptores, sin que ninguna de las partes tenga que depositar su confianza en la otra. El elemento diferencial sobre otras implementaciones de sistemas de pago es que TumbleBit usa firmas ciegas RSA de una manera inteligente, de manera que se evita que el Tumbler sepa que transacción entrante se corresponde con una determinada transacción saliente. Si muchas cuentas realizan transacciones usando el Tumbler entonces es imposible trazar la relación entre una cuenta receptora y la cuenta emisora que los envió. Ni siquiera el Tumbler puede enlazar ambas cuentas.

El Monedero Breeze de Stratis ofrece la funcionalidad TumbleBit tanto para Bitcoin como Stratis, haciéndolo útil para una audiencia mucho mayor que si solo se pudiese utilizar en la red de Stratis. Además, puesto que el protocolo TumbleBit utiliza un canal de pagos fuera de cadena (*off-blockchain*), es posible realizar muchos pagos usando el Tumbler en aproximadamente la misma cantidad de tiempo que tomaría realizar un solo pago.

El módulo de Identidad de Stratis está todavía en un estado de prueba de concepto, pero pese a ello ya es funcional. Los usuarios pueden acceder a sus cuentas de Microsoft, Google o LinkedIn usando la aplicación móvil de Identificación de Stratis y estos servicios notificarán a Stratis del inicio de sesión realizado. Una cuenta especial, propiedad de Stratis, almacenará la confirmación de este inicio de sesión creando un *hash* con la información de identificación personal (por ejemplo, nombre y dirección de email) y almacenándolo en la *blockchain* de Stratis.

La confirmación por parte de Google de que una persona posee una determinada cuenta de Gmail puede que no sea el servicio de identificación más práctico, pero es sencillo ver como el mismo mecanismo podría ser usado para demostrar la propiedad de una determinada información que sea mucho más difícil de verificar. Por ejemplo, un agente del gobierno podría confirmar que alguien ha presentado un documento de identificación con su foto, junto a un nombre y una dirección. Si el usuario puede proveer su nombre y una dirección de manera que

concuerda con el *hash* presente en la *blockchain*, eso también podría convencer a un proveedor de servicios de que el usuario posee el correspondiente identificador con foto, puesto que el agente del gobierno ya ha confirmado anteriormente esas 3 porciones de información conjuntamente.

La integración de TumbleBit en el monedero Breeze y en el módulo de Identidad de Stratis son dos ejemplos de este tipo de características que Stratis pretende ofrecer en la plataforma. No estoy seguro de haber entendido completamente el funcionamiento de la arquitectura de Stratis pero, por lo que puedo entender, la idea es que la *blockchain* de Stratis delegue los procesos en segundo plano para cada nueva característica, tales como TumbleBit y la Identidad de Stratis, a un juego dedicado de nodos (*masternodes*). Por ejemplo, el futuro Nodo Breeze (no confundir con el monedero Breeze, el cual utiliza SPV en lugar de necesitar un nodo completo) será un *masternode* que dará servicio a Tumbler. De forma similar, existen planes para construir *masternodes* que procesen las transacciones de Identidad de Stratis, aunque no se realmente lo que esto significa y no puedo encontrar más detalles sobre ello.

En último lugar, es necesario mencionar que el equipo de Stratis ha planeado varias características más, siendo la más destacada el despliegue de cadenas laterales ancladas a la cadena de Stratis. Según lo entiendo, este será el principal mecanismo con el que Stratis para ofrecer a los clientes *blockchains* privadas y personalizables.

Desafortunadamente, no he sido capaz de encontrar más detalles acerca de como funcionarán las cadenas laterales en Stratis. El whitepaper de Stratis hace referncia al Documento Técnico sobre Blockchains de Blockstream, pero esa es la única pista que he encontrado hasta ahora sobre el diseño de Stratis. Particularmente, creo que no es tan fácil asegurar y transferir valor entre dos *blockchains* sin tener al menos algunos mineros en cada cadena para validar todas las transacciones en ambas cadenas. Los detalles, incluyendo cómo el protocolo de la cadena lateral gestiona los *forks* y las reorganizaciones, son cruciales para poder valorar la seguridad de este mecanismo.

Incluso suponiendo que las transferencias entre la cadena de Stratis y las cadenas laterales son seguras, todavía queda pendiente el asunto de la seguridad de las mismas cadenas laterales. El Documento Técnico de Stratis afirma numerosas veces que la cadena de Stratis proveerá, de alguna manera, seguridad para sus cadenas laterales, pero no explica como lo hará. Típicamente, las cadenas laterales son completamente independientes y deben asegurarse por si mismas.

Comparación con Ardor

En Ardor, por su lado, la cadena madre (*parent chain*) es la que ofrece la seguridad para cada cadena hija (*child chain*).

En realidad, esta es una de las diferencias más importantes entre la arquitectura cadena madre / cadena hija de Ardor y las típicas implementaciones de las cadenas laterales. Desafortunadamente, sin más información técnica por parte del equipo de Stratis, es imposible de llevar acabo una comparación adecuada entre su diseño y el modo en que es enfocado por parte de Ardor.

Una comparación que podemos establecer es entre la característica TumbleBit de Stratis y la característica de Coin Shuffling de Ardor. (Nótese que el *Coin Shuffling* no estará disponible en la misma cadena de Ardor, sino que estará disponible en Ignis, su primera *child chain*, así como en todas aquellas *child chains* que quieran implementarlo) Esta característica es la implementación de Nxt del algoritmo de mezclado de monedas, que permite a un grupo de usuarios transferir una cantidad determinada de monedas desde sus cuentas (entrada) a una serie de cuentas de salida, una por cada entrada, sin tener que confiar en el resto ni saber quien de estos usuario controla cada cuenta de salida. El algoritmo no es muy complicado y la sección 4.2 del Documento Técnico ofrece una buena visión general de como funciona.

No soy ningún experto en ninguno de los algoritmos, pero el planteamiento de TumbleBit parece tener un par de ventajas sobre el CoinShuffle. Dado que usa canales de pago *off-blockchain*, es potencialmente capaz de escalar hasta un mayor ratio de transacciones, además de añadir una medida de privacidad a los pagos, solventando dos problemas a la vez. Además, si el objetivo es evitar que un observador identifique la relación entre varios pagos (lo que podría filtrar informaciones sobre los clientes de un determinado negocio o de una cadena de suministro, por ejemplo), probablemente sería más conveniente hacer los pagos fueran retornados a la misma cuenta a través de TumbleBit en lugar de tener primero que mezclar cada pago a una nueva cuenta.

Sobre el tema de las verificaciones de identidad, creo que el módulo de Identidad de Stratis es una interesante prueba de un concepto, pero en mi opinión Ardor ofrece un juego de herramientas más amplio para los servicios relacionados con la identificación. Mientras que un servicio como el de Identidad de Stratis puede ser construido con facilidad en cualquier *blockchain*, Ardor ofrece un par de características únicas que podrían extender este servicio para ofrecer algunas aplicaciones interesantes.

En Ardor, los validadores de identidad serán capaces de confirmar el propietario de una cuenta a través de las Propiedades de Cuenta. Esto son bits de datos arbitrarios que pueden ser permanentemente asociados con una cuenta en la *blockchain*, de modo semejante a las confirmaciones en el módulo de Identidad de Stratis. Una característica novedosa que traerá Ardor es la habilidad de emitir activos que solo podrán ser intercambiados entre aquellas cuentas que cuenten con una determinada propiedad de cuenta.

En los casos en que las regulaciones del gobierno requieran que los emisores de activos necesiten saber quién ha comprado sus activos, esta características permitirá a los emisores restringir el intercambio de los activos en sus cuentas con otras cuentas en la que la identidad de sus propietarios haya sido convenientemente verificada por los proveedores de identidad establecidos. Este nivel de control podría ayudar a que las acciones lanzadas en la *blockchain*cuenten con un fundamento legal más firme y ayuden a los emisores a cumplir apropiadamente con la legalidad.

Aún sin cumplir con las normas regulatorias, los emisores de activos puede que encuentren otros usos para esta característica. Por ejemplo, un club u organización privada podría expresar los requerimientos para acceder a la membresía de un club como una serie de requisitos en las propiedades de cuenta, emitir un activo que solo unas cuentas elegibles pueden obtener y después usar el activo para pagar dividendos o para lanzar encuestas entre sus miembros.

Algunas Reflexiones sobre el Marketing

Incluso aunque hayas leído hasta aquí, puede que todavía te estés preguntando qué es exactamente la plataforma Stratis y cómo va a funcionar. Para ser honesto, yo mismo me he planteado estas cuestiones también, incluso tras dedicar muchas horas a estudiar Stratis. Con riesgo de hablar en el límite de mis posibilidades, creo que sería útil comparar y contrastar los esfuerzos de marketing de Jelurida y del Grupo de Stratis para arrojar algo de luz sobre por qué es tan difícil para mí responder a estas cuestiones tan básicas.

Al leer la web de Stratis y su *white paper* (enlazado más abajo), francamente tengo la impresión de que *estos recursos no estaban realmente escritos para mí*. El lenguaje que utilizan me recuerda al de los comerciales de mi empresa, y hace mucho descubrí que los ingenieros y el personal de ventas tienden a no entenderse del todo bien.

Leo que Stratis ofrece "sencillas y asequibles soluciones de extremo extremo" para "agilizar y acelerar el desarrollo de proyectos *blockchain*"; que es una "potente y flexible plataforma de desarrollo *blockchain* para cubrir las necesidades reales de los negocios y otras organizaciones que pretenden desarrollar, testear y desplegar aplicaciones en la *blockchain*"; y que su "arranque con un *click* implica que las nuevas cadenas pueden ser lanzadas con una velocidad sin precedentes, ajustadas a las necesidades del negocio"; pero todavía no entiendo realmente lo que significan todas estas cosas, y aún menos cómo Stratis las va cumplir.

Este tipo de lenguaje no ofrece información de valor para mí. Sin detalles técnicos, estoy completa y desesperadamente perdido. No obstante, se que hay mucha gente que se desenvuelven en este lenguaje de negocios y que estas personas probablemente podrían leer el *white paper* de Stratis y abstraer de una manera decente, o incluso muy decente, una idea concreta de lo que la compañía plantea hacer. En cambio, yo tuve que leer el *white paper* múltiples veces antes de que pudiese comprender la idea, y no estoy completamente seguro de que lo haya conseguido.

Por su parte, el *white paper* de Ardor contiene detalles técnicos sobre cómo funciona Ardor y qué lo distingue de otras plataformas *blockchain*. Es obvio al observar tanto el contenido como la forma en que está organizado, que los ingenieros han jugado un papel importante a la hora de escribirlo. Cuando terminé de leerlo por primera vez, entendí con claridad los problemas que Ardor soluciona y cómo los soluciona.

A donde quiero llegar con esta comparación es que las personas con mentalidad de negocios y las personas con mentalidad técnica a menudo no hablan el mismo lenguaje y los materiales de marketing que presentan el Grupo Stratis y Jelurida parecen reflejar estas diferencias. Personalmente, me resultó frustrante encontrar tan poca información técnica entre los recursos de Stratis. Y esta frustración probablemente haya evitado que yo pudiese comprender completamente Stratis.

Conclusión

¿Es mi análisis de Stratis demasiado duro? Quizá. Realmente creo que TumbleBits es una solución tecnológica interesante y me parece inteligente que el monedero Breeze lo implemente tanto para Stratis como para Bitcoin. Además, si nos olvidamos de los contenidos del *white paper*que dicen que la cadena de Stratis será la encargada de asegurar a sus cadenas laterales y en su lugar asumimos que cada cadena lateral será la responsable de su seguridad, entonces puedo usar mi imaginación para interpretar las lagunas y conseguir formarme una imagen mental de lo que Stratis será cuando esté completado.

No obstante, en esta imagen mental Stratis será básicamente un competidor de Lisk. Por supuesto que Stratis está basado en .NET y en el protocolo Bitcoin en lugar de en JavaScript y en los tipos de transacciones predefinidas de Lisk. Y el conjunto de características que ambos equipos pretenden ofrecer no son completamente iguales pero, en esencia, ambos proyectos pretenden ofrecer una *blockchain* pública central junto a una serie de herramientas para la creación de cadenas laterales en ella. Además, ambos proyectos están en fases bastante iniciales de desarrollo y esta puede ser la razón por la que no hay mucha información técnica disponible.

Ardor es bastante diferente. Construido usando el código base de Nxt, es mucho más maduro que Stratis, aún a pesar de no haberse lanzado todavía en la main net. Su arquitectura de cadena madre / cadena hija consigue el objetivo indicado en el Documento Técnico de Stratis (una forma de que los negocios puedan crear sus propias *blockchains* personalizadas sin tenerse que preocupar por su seguridad) mejor que las soluciones basadas en arquitecturas de cadenas laterales. Y alcanzar la rica variedad de características que ya soporta Ardor todavía le llevará bastante tiempo a Stratis.

Quizás también sea importante destacar que tanto Jelurida como la comunidad Nxt han hecho un gran trabajo para ofrecer públicamente información técnica sobre Nxt y Ardor. Esta información otorga gran credibilidad al proyecto de Ardor y fortalece la comunidad. En mi opinión, esto es lo que diferencia al marketing verdadero de la mera publicidad.

Komodo / SuperNET

Ardor Frente a la Competencia, Parte 6

Esta semana he analizado Komodo, la plataforma *blockchain* que conforma la base de Supernet.

SuperNET

Al igual que sucede con Waves, SuperNET fue fundado por una persona que fue muy activo en la comunidad Nxt en el pasado. Y tal y como hice en mi artículo sobre Waves, no voy a entrar en este tema en este artículo.

Es suficiente con decir que James/jl777 era desarrollador de SuperNET, el Multigateway y varios otros proyectos en Nxt, incluyendo un determinado número de activos en el Intercambio de Activos de Nxt, pero que abandonó la comunidad Nxt durante el turbulento periodo de finales de 2015 y comienzos de 2016. Desde entonces, ha creado la plataforma Komodo, que ahora funciona como elemento base de SuperNET.

La visión de SuperNET es permitir que los usuarios realicenr transacciones de forma sencilla entre diferentes tipos de criptomonedas, para disfrutar de todas las ventajas de cada moneda. Por ejemplo, si lo he entendido correctamente, una aplicación de SuperNET podría permitir que los usuarios realizasen transacciones privadas con Bitcoin, siendo convirtida desde y hacia una

moneda privada como Komodo en segundo plano. Desde el punto de vista del usuario, sería como si Bitcoin hubiese "tomodo prestada" la función de privacidad de Komodo.

Supernet en si mismo no es una *blockchain*. En cambio, es una estructura compuesta de varios componentes. Sus principales partes son:

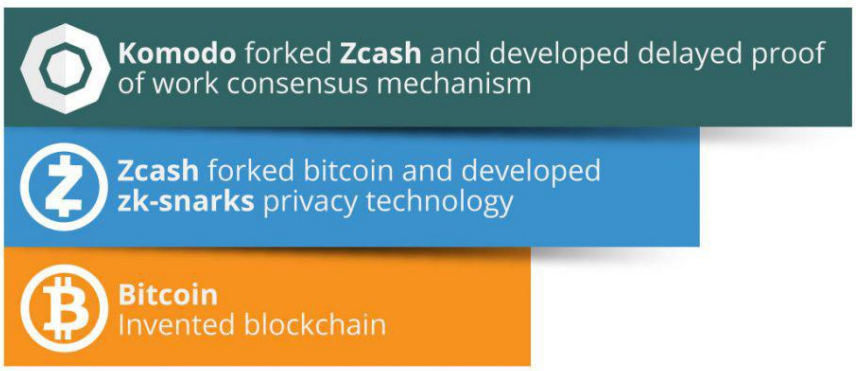
1. **Komodo**, una *blockchain* ligada a Bitcoin;
2. **assetchains y geckochains**, *blockchains* independientes ligadas a Komodo;
3. **el monedero Agama**, un monedero multimoneda;
4. **BarterDEX**, un *exchange* descentralizado (DEX) que será integrado en el monedero Agama;
5. **Iguana**, el código base sobre el que se cimienta el monedero de Agama y parte de Komodo.

Hay que mencionar que muchos de los textos sobre SuperNET hacen referencia al monedero de Agama como monedero Iguana, puesto que este era su nombre anterior.

El proceso de anclaje de los elementos 1 y 2 es el algoritmo de consenso *delayed proof-of-work*(Prueba de Trabajo Diferida) de Komodo, que describo a continuación. Hablaré de BarterDEX más adelante.

Prueba de Trabajo Diferida

Komodo es un *fork* de zCash, una *blockchain* que usa el algoritmo *zero-knowledge proofs* o de prueba de conocimiento-cero (via zk-SNARKs) para permitir que los usuarios realicen transacciones sin revelar sus números de cuentas al *exchange*. Komodo ha añadido varias características a su ramificación del código base de zCash, incluyendo el algoritmo de Prueba de Trabajo Aplazada (dPoW), así como un mecanismo para la creación de *blockchains* adicionales que son ancladas periódicamente a la cadena de Komodo.



El *white paper* del dPoW afirma que el mecanismo del dPoW permite que cualquier *blockchain* se asegure a sí misma usando el *hashpower* de Bitcoin, gracias a que

periódicamente se certifican ante Bitcoin. En pocas palabras, el consenso en las *blockchains* más débiles tiene lugar en dos fases: un consenso inicial siguiendo los métodos tradicionales (por ejemplo con PoW o PoS) y una segunda capa de consenso establecida periódicamente por una serie de nodos notarios, elegidos por los partícipes, que almacenan un *hash* de los bloques de las cadenas débiles en la *blockchain* de Bitcoin. Todos los nodos en la red están de acuerdo en que, en caso de *fork*, no reorganizarán la *blockchain* a un estado anterior previo al que el que fue certificado usando Bitcoin.

Sostiene el autor que de esta manera, la *blockchain* más débil hereda algo de la seguridad de Bitcoin. Incluso un atacante que contase con la mayoría del *haspower* de la red no podría modificar la *blockchain* más allá del último bloque certificado. Del mismo modo, alguien que espere hasta que una transacción en la *blockchain* más débil sea certificada en Bitcoin puede estar seguro de que no será revertida.

El *white paper* también propone un mecanismo para permitir que la red vuelva a su mecanismo inicial de consenso en caso de que un nodo que actúe como notario no esté disponible. La idea es que todos los nodos en la red tienen la posibilidad de minar, pero a los nodos notario se les asigna una dificultad menor que a los nodos normales. Como resultado, los nodos notario ganarán normalmente la mayoría de los bloques, pero si un atacante consiguiese de algún modo desactivarlos (con un ataque DDoS, por ejemplo) los nodos normales podrían continuar minando bloques y la *blockchain* continuaría su funcionamiento sin interrupción, solo que sin la seguridad adicional proporcionada por Bitcoin. De esta manera, la cadena dPoW es de algún modo menos centralizada de lo que parece en un primer vistazo.

Esta serie de razonamientos nos lleva a preguntarnos que es exactamente lo que se gana con el mecanismo de notarización/certificación. En particular, si un atacante es capaz de obtener el control de los nodos notario, puede evitar que estos firmen las transacciones Bitcoin que certifican los bloques de las cadena más débiles, obligando a la *blockchain* más débil a depender únicamente de su mecanismo de consenso inicial. Así que parece que la seguridad extra ofrecida por el proceso de certificación depende implícitamente en la existencia de una mayoría de nodos notario honestos.

[EDICIÓN: Tras hablar con jl777, he descubierto que Komodo permite que una minoría de los notarios (13 de 64) firmen cada transacción de certificación. Esto, a su vez, reduce las tasas de Bitcoin que deben pagarse y hacen más difícil el ataque expuesto, puesto que un atacante tendría que controlar la gran mayoría de los nodos para vencer al mecanismo de notarización. Mis afirmaciones originales estaban basadas en lo que escribió en el whitepaper del dPoW, el cual sugiere que 33 de los 64 notarios deben firmar las transacciones de certificación.]

Esto es básicamente el modelo de seguridad de las *blockchains* basadas en el modelo de Prueba de Participación Diferida (*delegated proof-of-stake – DPOS*) como Bitshares. Tanto en DPoW como en DPoS, los usuarios votan en función de su saldo (*stake*) en favor de una serie de cuentas “especiales” de las cuales depende la seguridad del resto de la red. Ambos sistemas también sufren las mismas debilidades: una responsabilidad para los usuarios para mantenerse al día con los temas “politicos” del sistema para saber que cuentas son lo suficientemente

confiables y merecen ser votadas, y la consiguiente apatía del votante que produce esta responsabilidad.

Con todas estas consideraciones, creo que no le veo mucho sentido al dPoW sobre otras alternativas. Si el mecanismo de consenso inicial de la cadena más débil es lo suficientemente seguro para protegerlo, dado su actual valor económico, entonces pagar Bitcoin para certificarlo parece un derroche de dinero. Por otro lado, si el consenso inicial no fuese suficiente, entonces parece que la seguridad del resto de la cadena depende de la elección de notarios honestos. Pero, en ese caso, ¿Por qué no utilizar DPOS y beneficiarse del mejorado rendimiento de las transacciones que las cadenas DPOS ya han alcanzado?

Dejando de lado estas consideraciones, no hay que pasar por alto el hecho de que la plataforma Komodo usa cadenas dPoW anidadas para ayudar a conseguir el objetivo de SuperNET de conectar diversas *blockchains* diferentes. Las cadenas adicionales de Komodo se llama "*assetchains*" y "*geckochains*". Estas cadenas se certifican a si mismas ante Komodo, quien a su vez se certifica ante Bitcoin. De nuevo se afirma que todas las cadenas involucradas heredan el nivel de seguridad de Bitcoin pero, tal y como se describe más arriba, buena parte de esto depende de los nodos notario de cada cadena.

A diferencia de los activos en Nxt y Ardor, o incluso en las *child chains* de Ardor, las cadenas de activos (*assetchains*) de Komodo son *blockchains* completamente independientes. Su única conexión con las cadenas de Komodo es el mecanismo de certificación dPoW. De esta manera, quizá se encuentren más próximas a las cadenas lateras que proponen Lisk y Stratis que a las férreamente dependientes child chains de Ardor.

Las *geckochains* son como las *assetchains* pero con soporte para contratos inteligentes. No he encontrado muchos detalles sobre las geckochains, y no parece que estén disponibles todavía, pero el cliente de Komodo ya soporta las *assetchains* a través de una interfaz de línea de comandos.

BarterDEX

El *exchange* descentralizado de SuperNET, llamado BarterDEX, permite a los usuarios realizar transacciones atómicas entre *blockchains* sin tener que depender en la confianza en terceros. El equipo todavía no lo ha integrado en la interfaz de usuario del monedero Agama pero ya están trabajando en ello. Mientras tanto, BarterDEX puede usarse por separado.

BarterDEX consiste en tres componentes principales: un juego de nodos designados para encontrar órdenes coincidentes; una serie de nodos "proveedores de liquidez", que actuarán como creadores de mercado; y un protocolo para que los usuarios intercambien entre sí monedas de diferentes *blockchains* en una única operación atómica.

Los nodos para encontrar órdenes coincidentes funcionan del mismo modo que en Waves: se encargan de centralizar parcialmente las órdenes de compra y venta para ofrecer una experiencia de usuario más atractiva. De este modo, los *traders* no tienen que esperar al

siguiente bloque en las *blockchains* en cuestión para saber si sus órdenes se han ejecutado o si deben cancelar una orden.

Los nodos Proveedores de Liquidez (*Liquidity provider – LP*) mantienen los saldos de al menos dos de las monedas soportadas y automáticamente comercializan con ellas con un margen de beneficios que es definido por el usuario, relativo a un exchange centralizado. Por ejemplo, es posible estable un nodo LP que *tradee* BTC y KMD en BarterDEX a la vez que en Bittrex. Los operadores de los nodos LP asumen el riesgo asociado con poseer fondos en un *exchange* centralizado, a cambio de beneficiarse de las oportunidades de arbitraje entre los dos mercados. El resto de usuarios de BarterDEX, por su parte, obtienen mayor liquidez y menores diferenciales entre las órdenes *bid* y *ask* que si no contaran con este servicio, sin tener que dejar sus monedas en un *exchange* centralizado.

Cuando la orden de un usuario se completa, posiblemente con una orden emitida por un nodo LP, BarterDEX usa un protocolo de canjeo atómico entre cadenas para fijar fijar la operación en las dos *blockchains* involucradas. Probablemente los detalles variarán dependiendo del par de monedas en cuestión, pero conceptualmente el proceso es similar en todos los casos. Se supone que una *blockchain* es compatible con Bitcoin o, como mínimo, soporta el equivalente a los Contratos de Ejecución Retardada *Hasheados* (*Hashed timelocked contracts – HTLCs*) de Bitcoin. La otra *blockchain* tiene que soportar 2 de 2 transacciones multifirma del tipo 2 de 2 (*2-of-2 multisign*).

Supongamos que Bob quiere *tradear* sus fondos en una cadena compatible con Bitcoin a cambio de las monedas de Alice en otra cadena. Tanto Alice como Bob crean un par de clave pública / clave privada y los *hashes* de las claves privadas. Alice envía a Bob una transacción multifirma 2 de 2 que él puede gastar una vez que conozca ambas claves privadas, momento en el que Bob enviará a Alice una transacción del tipo *timelocked* que Alice puede gastar cuando revele su clave privada. Una vez lo haya hecho, Bob la usará para desbloquear su transacción multifirma y la operación se completará.

El protocolo añade un poco de complejidad para proteger a cada parte en caso de que la otra parte cancele el procedimiento anticipadamente. Si Alice se va sin gastar la transacción que envió Bob, Bob puede recuperar sus fondos una vez que el bloqueo temporal sobre esa transacción caduque, utilizando tan solo su clave privada para ello. Por otro lado, para proteger a Alice del mismo riesgo, el protocolo requiere que Bob emita un “depósito” inicial en forma de transacción *hasheada* retardada. Si abandona antes de pagar a Alice, ella puede esperar a que cumpla el plazo del *timelock* sobre este depósito para recuperarlo.

Admito que esto solo es una visión superficial del protocolo de canjeo automático, pero espero que sirva para darte una idea de cómo funciona. La parte más importante aquí es que no existe ningún *exchange* centralizado para facilitar la transacción. Alice y Bob han intercambiado monedas en diferentes *blockchains* sin tener que confiar en la otra parte o en algún intermediario. Puedes encontrar más detalles sobre este proceso en el *white paper* de BarterDEX.

Comparado con Ardor

Entonces, ¿qué podemos hacer con Komodo y SuperNET? Esta cuestión depende en gran medida de si el algoritmo de prueba de trabajo diferida de Komodo ofrece un grado sustancial de mejora de la seguridad para Komodo y sus *assetchains*. Desde mi punto de vista, no lo consigue: ofrece aproximadamente el mismo grado de seguridad que el algoritmo de prueba de participación diferida, aún en el caso de que la *blockchain* notario fuese perfectamente inmutable.

Con esto en mente, las *assetchains* de Komodo se parecen mucho a las *side chains* desplegables por el usuario que tanto Lisk o Stratis pretenden ofrecer. En los tres proyectos, a diferencia de las *child chains* de Ardor, cada *assetchain* o *sidechain* es responsable de su propia seguridad. Sin embargo, Komodo parece tener ventaja sobre tanto Lisk como Stratis en términos de funcionalidad, puesto que los usuarios ya pueden desplegar sus propias *assetchains* y realizar transacciones de canjeo atómicas entre algunos pares.

Hay que darse cuenta de que las *child chain* de Ardor almacenan los *hashes* de sus bloques en la cadena de Ardor, de forma semejante a como Komodo almacena los de sus bloques en Bitcoin, pero hay una diferencia crucial: los nodos forjadores de Ardor validan todas las transacciones de todas las *child chains*. Cada *child chain* hereda de manera efectiva todo el poder de forja de la cadena de Ardor, convirtiéndolas en tan seguras como lo es Ardor y obviando la necesidad de mineros o forjadores separados.

Sobre los canjeos atómicos entre cadenas, Ardor y Komodo presentan mayores diferencias. Ardor soporta de manera nativa transacciones entre las *child chains* y también entre cada *child chains* y la cadena madre. Además, soporta un tipo de transacción por fase que es equivalente a la multifirma 2 de 2, permitiendo los mismos tipos de canjeos atómicos con las *blockchains*compatibles con Bitcoin que permite BarterDEX. Ardor incluso añade la capacidad de combinar múltiples operaciones condicionadas con los operadores booleanos AND, OR y NOT, permitiendo potencialmente a los usuarios a crear el equivalente a

las Transacciones de Ejecución Retardada *Hasheadas*. Usando el enfoque de BarterDEX, esta característica podría permitir los canjeos atómicos entre cadenas hacia cualquier *blockchain* que soportase multifirma 2 de 2.

Conclusión

La visión de SuperNET sobre las *blockchains* interconectas presenta mucho atractivo, y con la combinación de la plataforma Komodo, el monedero Agama y el exchange BarterDEX, Supernet ha hecho un gran progreso para materializar ese objetivo. Aunque soy escéptico acerca de que el algoritmo de prueba de trabajo diferida ofrezca un grado sustancial de seguridad adicional a Komodo y a sus *assetchains*, la capacidad de desplegar rápidamente una *assetchain* pone a Komodo, como mínimo, por delante de Lisk y Stratis en su carrera para crear una plataforma funcional con cadenas laterales. Además, encuentro que tiene mucho valor la capacidad de realizar canjeos automáticos entre cadenas usando BarterDEX.

Aún así, no puedo evitar pensar si existe en el corazón de SuperNET una tensión fundamental entre dos de sus objetivos. Por un lado, pretende integrar las mejores características de *blockchains* muy dispares entre sí, ofreciendo a los usuarios y desarrolladores una manera impecable de disfrutar de las ventajas exclusivas que cada cadena ofrece. Por otro lado, ofrece a Komodo como una única plataforma para solucionar la mayoría de los problemas, permitiendo las transacciones privadas, cadenas laterales creadas por los usuarios y, en el futuro, contratos inteligentes. El éxito en cualquiera de estos objetivos parece que obstaculizará conseguir el resto.

Ardor, por su parte, también tiene una visión atractiva, y quizá sea una visión más coherente: soportar una multitud de negocios y proyectos en sus *child chains*, ofreciendo a cada uno de ellos una serie de características listas para su uso, permitiendo que cada una de ella interactúe con el resto y evitar que ninguna de ellas se tenga que preocupar por su seguridad o por almacenar el historial de transacciones del resto. Ardor ya ofrece ya la mayoría de la tecnología para materializar esta visión. Sólo falta que los negocios, desarrolladores y los usuarios le den buen uso a esta tecnología.

Ethereum (Contratos Inteligentes)

Ardor Frente a la Competencia, Parte 7

Esta semana he analizado Ethereum, el cual creo que no necesita introducción alguna.

Al estudiar varios de los proyectos que he investigado en esta serie, en ocasiones ha sido difícil encontrar información técnica detallada sobre ellos. En Ethereum sucede exactamente lo contrario: hay tanta información disponible que es difícil resumirla en un artículo de una longitud razonable sin riesgo a simplificar en exceso las ideas.

Por este motivo, he elegido solo dos aspectos de Ethereum para compararlos con Ardor. Esta entrega compara sus contratos inteligentes con las transacciones inteligentes de Ardor y el próximo artículo comparará la manera en que ambas plataformas se enfrentan al problema del crecimiento desmesurado de la *blockchain* (*blockchain bloat*). Hay muchos más temas sobre los que me habría gustado hablar, como por ejemplo su plan para convertirse en *Proof-of-Stake*(Casper), su estrategia *state-channel* (Raiden y Plasma), su asociación con grandes compañías a través de la *Enterprise Ethereum Alliance* y ejemplos de proyectos que lo utilizan, pero poder hablar aunque solo sea de un par de estos temas con la rigurosidad necesaria ya es suficiente desafío. Además, los dos temas que he elegido ofrecen, en mi opinión, la comparativa

más interesante entre las dos plataformas (visita también el artículo de Ardor Frente a Plasma, enlazado más arriba, para algunas reflexiones sobre Plasma).

Sin más dilación, hablemos sobre los contratos inteligentes o *smart contracts*.

Los Contratos Inteligentes y la "Diversidad de Estados de una Cuenta"

El diseño de Ethereum combina elementos de Bitcoin y Nxt y añade además varias características novedosas. Al igual que Bitcoin, Ethereum usa un lenguaje de programación de bajo nivel para codificar las transacciones y almacena los contenidos de cada bloque en árboles del tipo Merkle cuyos *hashes* de raíz están almacenados en las cabeceras de los bloques (más sobre este tema en el próximo artículo). Al igual que Nxt, realiza un seguimiento del estado actual de los saldos de las cuentas y otros datos específicos de las cuentas directamente, en lugar de seguir el modelo de Bitcoin de Salida de Transacciones sin Gastar (*Unspent Transaction Output – UTXO*).

Las innovaciones más importantes que Ethereum añade a esta combinación son dos: la capacidad de almacenar *scripts* (contratos) en las llamadas "cuentas de contratos", las cuales realizan transacciones automáticamente sin estar controladas por un usuario; y la capacidad de que las transacciones persistan en una cuenta desde una transacción hasta la siguiente. El lenguaje de programación de Ethereum es también, de algún modo, más potente que el de Bitcoin, permitiendo que los contratos incluyan *loops* e invoquen a otros contratos.

Combinando estas ideas, es posible crear los llamados "contractos inteligentes" con estados, que son unas líneas de código y datos que residen en las cuentas de contratos como agentes autónomos, escuchando los *inputs* de los usuarios y otros contratos y realizando transacciones con ellos de acuerdo a las reglas definidas en el código de su contrato. El apelativo "*con estados*" de la frase anterior es fundamental: puesto que un contrato inteligente puede tener su propio estado interno, es posible que una transacción afecte la forma en que las transacciones subsecuentes son procesadas. Esto es una diferencia significativa con el modelo de Bitcoin, dónde los *scripts* de transacciones solo se ejecutan una única vez y dónde el concepto de "estado" disponible para un *script* esta esencialmente limitado a si un determinado *output* se ha gastado o no.

(Quizá te hayas dado cuenta de que no he dicho nada sobre la exhaustividad del Turing. Dependiendo de lo escrupuloso que te quieras poner, podrías argumentar en favor de cualquiera de las posiciones acerca de si el lenguaje de programación de Ethereum es realmente Turing en su totalidad. Tal y cómo explica el presentador de este excelente vídeo, ser completamente Turing puede actuar en ocasiones cómo un falso indicador. Es mucho más importante el hecho de que los contratos inteligentes está repletos de estados y pueden realizar transacciones entre ellos y con otros usuarios de maneras muy interesantes.)

Las aplicaciones potenciales de los contratos inteligentes se extienden más allá de establecer condiciones para la transferencia de dinero de una cuenta a otra. Incluso el *white paper* original (una gran lectura, por cierto) proponía un puñado de usos no-financieros, incluyendo el almacenamiento de archivos, votación, computación distribuida, gobernanza de organizaciones descentralizadas y mercados descentralizados. Desde entonces, los desarrolladores han encontrado multitud de otras aplicaciones, tales como mensajería descentralizada. Y, por supuesto, la aplicación de Ethereum más comúnmente utilizada hasta el momento: llevar a cabo ventas de *tokens* para varios proyectos.

Las "Transacciones Inteligentes" de Ardor

Si esa lista de aplicaciones te resultan familiares, puede que sea porque todas ellas ya han sido implementadas en Nxt yArdor como como "transacciones inteligentes" predefinidas. Introducidas por el predecesor de Ardor, Nxt, las transacciones inteligentes son porciones de funcionalidades de "*blokchain* 2.0" que los desarrolladores de Nxt y Ardor han publicado como parte del mismo protocolo. Estas transacciones permiten a los desarrolladores crear aplicaciones *blockchain* sin tener que escribir y testear sus propios contratos inteligentes.

Para permitir que los usuarios comunes (es decir, los no desarrolladores) se beneficien también de esta funcionalidad, los monederos oficiales de Nxt y Ardor incluyen un puñado de transacciones inteligentes incluidas. Se trata de:

- el Intercambio de Activos, dónde los usuarios pueden emitir activos, comercializarlos y pagar dividendos a los poseedores del activo;
- el Sistema Monetario, dónde los usuarios pueden emitir monedas y conducir diferentes tipos de campañas de *crowdfunding*;
- un sistema de mensajería, que permite que los usuarios envíen mensajes de texto simple o codificados;
- un sistema de votación, que permite que los usuarios realicen votaciones en función de la cuenta, balance de cuenta, balance de determinados activos o balance de determinadas monedas;
- un *coin shuffler* integrado, que puede otorgar a los usuarios un cierto grado de privacidad difuminando su historial de transacciones;
- un almacenamiento de datos descentralizado, que puede grabar el *hash* de un archivo permanentemente en la *blockchain* y, opcionalmente, almacenar el archivo en si mismo permanentemente gracias a los nodos de archivo;
- un mercado descentralizado, donde los usuarios pueden comprar y vender bienes y servicios entre pares (*peer-to-peer*);
- un nuevo Intercambio de Monedas (solo en Ardor), donde los usuarios pueden comprar y vender las monedas de las diferentes *child chains* directamente entre sí;
- un número de características avanzadas, tales como las transacciones condicionadas, que permiten a los usuarios establecer unas condiciones sobre cuándo y cómo otras transacciones son ejecutadas, y las propiedades de cuenta, que pueden usarse para asociar datos arbitrarios con una cuenta.

Por supuesto que estas no son las únicas transacciones que se pueden construir a partir de las transacciones inteligentes, pero sirven para mostrar el alcance de lo que se puede llegar a alcanzar con ellas. Todas estas características, junto a unas cuantas más, estarán disponibles en Ignis, la primera *child chain* de Ardor. Los creadores de otras *child chains* contarán con la opción de implementar tantas de estas características como necesiten hasta amoldarse a sus proyectos.

He escuchado diversas analogías para describir las transacciones inteligentes, pero mi favorita es que son como Legos, mientras que los contratos inteligentes son como arcilla: el primero de ellos no permite el mismo grado de control sobre los detalles más sutiles, pero son más rápidas y fáciles de utilizar que los segundos, y se puede combinar para formar productos finales con resultados espectaculares.

La analogía no es perfecta, por supuesto. Un potente argumento a favor de los contratos inteligentes es que, por ejemplo, es potencialmente posible que toda la lógica de negocios de una aplicación descentralizada (Dapp) se almacenada permanentemente y de forma inmutable en la *blockchain*, mientras que una Dapp construida a base de combinar transacciones inteligentes es muy probable que incluya algo de código externo. En este último caso, usar la Dapp puede requerir cierto grado de confianza en que el desarrollador no cambiará las reglas en futuras versionas de la misma.

Desde otro punto de vista, esta comparación también pone de relieve el mayor inconveniente de los contratos inteligentes: La facilidad con que los programadores pueden cometer errores de varios millones de dólares que no se puedan corregir.

Consideraciones de Seguridad

Casi cualquier software que es mínimamente complejo contiene imperfecciones y en muchas ocasiones estas imperfecciones hacen que el software pueda ser vulnerable a ataques de un *hacker*. Los desarrolladores de contratos inteligentes se enfrentan a una tarea

particularmente difícil, puesto que el código que escriben es inmutable y, como resultado, sus vulnerabilidades son permanentes.

Desafortunadamente, descubrir fallos catastróficos en contratos inteligentes no ha sido excepcional. El ataque que congeló el equivalente en ether a $150 M almacenado en monederos multifirma de Parity o el hackeo de $30 M del mismo monedero unos meses antes son solo los ejemplos más recientes que hemos visto en diversos titulares, pero no son los primeros y, con toda seguridad, no serán los últimos. Para una vista general de algunas de las vulnerabilidades comunes y un análisis de diversos ataques reales, incluyendo el desafortunado *hackeo* del DAO, recomiendo fervientemente leer este excelente documento elaborado por tres investigadores de la Universidad de Cagliari.

No hay que pasar por alto que ni el protocolo de Ethereum ni la Máquina Virtual de Ethereum (*Ethereum Virtual Machine – EVM*) fueron responsables de ninguno de estos ataques. Los seguidores de Ethereum a menudo sacan a relucir este punto, argumentando que Ethereum en si mismo es bastante seguro y todo lo que se necesita es que los desarrolladores escriban mejores contratos inteligentes. En un sentido literal tienen razón, por supuesto: En todos los casos Ethereum hizo lo que se suponía que debía hacer y la culpa recae en último lugar en los desarrolladores de los contratos inteligentes.

Pero personalmente me pregunto si este juicio que pretende absolver a Ethereum es demasiado rápido y si el problema podría ser mayor que unas pocas líneas erróneas en un contrato inteligente. De cualquier modo, por ahora me parece que el argumento fundamental de Ethereum es que otorga a los desarrolladores un poder tremendo pero unas herramientas insuficientes para usar ese poder de forma segura.

La ambición de los desarrolladores casi siempre excede el nivel de complejidad que se puede alcanzar mientras se mantiene, a la vez, el código perfectamente libre de errores, y siempre existirá la tentación de hacer que las funcionalidades tengan prioridad sobre la seguridad (esto es casi universal en el desarrollo de software, por cierto). Al inmortalizar el código que han creado, que contiene errores, almacenándolo en una *blockchain*, el brutal y despiadado Ethereum hará que paguen por sus pecados.

Afortunadamente, hay ciertas maneras de mitigar el riesgo de escribir contratos inteligentes vulnerables. Por ejemplo, es posible designar un contrato inteligente que pueda ser actualizado delegando para ellos sus responsabilidades en un segundo contrato, habitualmente denominado “contrato librería”, en una dirección que se puede cambiar para apuntar a una nueva librería más adelante.

Este enfoque permite a los desarrolladores parchear vulnerabilidades pero, como consecuencia, introduce la peliaguda cuestión de quien debe estar autorizado para cambiar a un nuevo contrato librería. Si es una única cuenta de un tercero, entonces el diseño introduce un cierto grado de confianza entre esa cuenta y el resto de usuarios. Por otro lado, si el desarrollador toma otro enfoque, tal como permitir votar a una mayoría de usuarios para aprobar cada nuevo contrato librería, entonces pueden emerger nuevos problemas a solucionar, tales como escribir un mecanismo de votación, asegurarse de que los usuarios están lo suficientemente informados

y comprometidos para votar, y evitar que un *hacker* cause un daño considerable durante todo el tiempo que se tarda en organizar una votación.

Otra aproximación muy prometedora sobre la seguridad de los contratos inteligentes es usar técnicas de verificación formal tomadas prestadas de las matemáticas. No sé mucho sobre estos métodos formales, así que tomad con pinzas lo que escribo aquí, pero no se si es más sencillo (o simplemente realizable) con programas sencillos cuya funcionalidad puede ser expresada como una serie de reglas sencillas y cortas. En estos casos, puede ser posible demostrar con certeza que el programa no contiene errores. Sin embargo, incluso técnicas evidentes como el *looping* o la recursividad pueden complicar el análisis significativamente, así que es mejor si el programa a testear es lo más sencillo posible.

¿Por qué insisto tanto en este tema? Poniendo todas estas ideas juntas podría parecer que la mejor manera para escribir contratos inteligentes implicaría: 1) mantenerlos lo más cortos y sencillos posibles; 2) delegar la lógica del núcleo de negocios a una librería de contratos que pudiese ser actualizada, en caso necesario; y 3) reutilizar librerías que han sido concienzudamente examinadas, para reducir al mínimo la cantidad de código nuevo. Si el segundo de estos puntos requiere que los usuarios de un contrato tengan que depositar su confianza en el autor de un contrato en cierto grado, como sucede habitualmente, entonces los contratos designados de acuerdo a los anteriores tres puntos comienzan a parecerse mucho a las transacciones inteligentes de Ardor: bits de código estable y concienzudamente testeado que permiten obtener las funcionalidades más comúnmente necesitadas, las cuales los desarrolladores pueden combinar para formar programas más complejos.

El Equilibro entre Seguridad y Flexibilidad

No estoy sugiriendo que las transacciones inteligentes de Ardor puedan realizar todo lo que puedan llegar a realizar los contratos inteligentes de Ethereum, ni estoy diciendo que combinaciones de transacciones inteligentes puedan siempre emular a los contratos inteligentes. Sin embargo, lo que estoy diciendo es que creo que hay una tensión natural entre la flexibilidad que una plataforma ofrece y la seguridad del código que, irremediablemente, los desarrolladores tienden a escribir usándola.

Desde este punto de vista, las plataformas *blockchain* están siempre asentadas en un bucle seguridad-flexibilidad. Próximo al extremo de la “seguridad” está Bitcoin, cuyo lenguaje de programación es deliberadamente bastante limitado para evitar que los usuarios bloqueen sus cuentas con *scripts* vulnerables (aunque esto es posible, por supuesto). Nxt y Ardor ocupan una posición intermedia en este espectro, limitando a los desarrolladores con un juego de tipos de transacciones predefinidas, pero incorporando una enorme variedad de funcionalidades en esos tipos.

Los contratos inteligentes de Ethereum, por su parte, abarcan todo el espectro. Es posible escribir *scripts* extremadamente simples, triviales y seguros en Ethereum y también es posible escribir *scripts* más complicados que contengan vulnerabilidades muy sutiles. Y lo que quizá sea igual de importante, es difícil para los usuarios diferenciar entre esos casos, y no sería razonable

en ningún caso esperar que el usuario lo hiciese. Usar Ethereum de manera segura implica evitar la parte del espectro que está más próxima a la “flexibilidad”, incluso aunque esto suponga la introducción de un cierto grado de confianza entre usuarios y desarrolladores.

Finalmente, vale la pena mencionar que Ardor ofrece una nueva característica, no disponible previamente en Nxt, que le ayuda a aproximarse al lado de la “flexibilidad”: la habilidad para combinar transacciones condicionadas usando operadores *booleanos* tales como AND, OR y NOT, para conseguir un comportamiento semejante a unos contratos inteligentes primitivos.

En pocas palabras, las transacciones condicionadas permiten a los usuarios condicionar la ejecución de una transacción latente a que tenga lugar un determinado evento, tal como la aprobación por parte de un determinado número de cuentas específicas, una votación en la que pueden participar todas aquellas cuentas que posean un determinado activo, el paso de una determinada cantidad de tiempo (*timelock*) o la revelación de un secreto (por ejemplo, un *hashlock*). En Ardor, las combinaciones de estos tipos de transacciones condicionadas pueden dar lugar a condiciones más complejas, tales como “la transacción X es válida si la mayoría de los accionistas de la compañía ABC la aprueban antes de una fecha Y, salvo que sean vetadas por una gran mayoría de los miembros de la junta de la empresa ABC.”

Sin duda, también se podrían combinar transacciones condicionadas de manera que permitiese la obtención de resultados inesperados, lo que podría incluir la pérdida de fondos. Pero yo diría que la ventaja sobre los contratos inteligentes en términos de seguridad todavía está ahí, puesto que los desarrolladores se pueden centrar en asegurarse que la lógica de la transacción para los negocios es segura, sin tener que preocuparse por los errores de bajo nivel (*low-level bugs*) tales como las condiciones de la carrera. Y por supuesto, la desventaja de ofrecer menor flexibilidad que los contratos inteligentes sigue también ahí.

Conclusión

Con un protocolo definido por una serie de transacciones inteligentes pre-compiladas en lugar de un lenguaje de programación de bajo nivel, Ardor probablemente nunca será capaz de ofrecer a los desarrolladores un rango de posibilidades tan grande como hace Ethereum, al menos en los casos en que todo deba hacerse en la cadena para no tener que depositar en la confianza entre las partes. Por otro lado, escribir contratos no triviales que sigan las mejores prácticas de seguridad requiere igualmente de confianza adicional entre usuarios y desarrolladores. Y, por supuesto, los usuarios de Ethereum tienen que acabar confiando en que los autores de los contratos inteligentes no han cometido ningún error y han escrutado concienzudamente y probado su código, para estar completamente seguros de ello.

Naturalmente, puedes pensar que esto mismo sucede con cualquier software, incluyendo las transacciones inteligentes de Ardor, pero hay una diferencia clave: simplemente hay mucho más código ejecutándose en Ethereum. Nxt ha sido de código abierto desde el comienzo, ofreciendo una gran oportunidad para ser revisado. El código de Ardor, que está basado en el código base de Nxt, será liberado pronto. Además, cada nuevo cambio añadido el protocolo ha sido chequeado en profundidad en una *testnet* pública antes de ser lanzado oficialmente. Lo mismo debería cumplirse para cada uno de los contratos inteligentes, pero con tanto código escrito, parece que es inevitable que existan más oportunidades para que los errores se cuelen en el entorno de producción.

En cualquier caso, sospecho que el grado en que la mayoría de las Dapps exitosas dependerán de código inmutable todavía es una pregunta sin respuesta. Si el acceso a una base de datos inmutable y a un puñado de operaciones simples sobre esos datos son suficientes para la mayoría de las aplicaciones, entonces las transacciones inteligentes de Ardor parece que tienen una clara ventaja sobre los contratos inteligentes. Por el contrario, si el concepto de que "el código es la ley" resulta ser esencial para la viabilidad de la mayoría de las Dapps, dónde cada Dapp requiera que la mayoría de su exclusivo código sea almacenado en la *blockchain* para no depender para nada de la confianza en un tercero, entonces el enfoque de Ethereum sea probablemente superior.

Confío en que habrá aplicaciones del mundo real que encajen en estas plataformas. Pero también me pregunto si a la larga quedará dejará claro que uno de los dos enfoques va a manejar la mayoría de las aplicaciones. Que enfoque resultará el ganador no está claro para mí, pero sospecho que el factor decisivo será la valoración de los usuarios sobre el grado de confianza que requiere cada opción. Y puesto que toda la atracción de la tecnología *blockchain* proviene de que permite a los usuarios realizar transacciones requiriendo de un mínimo grado de confianza, yo diría que el resultado de esa valoración será el realmente válido.

¡Gracias por leer este texto! Si te ha gustado este artículo, asegúrate de leer la siguiente parte de la serie, que comparará la forma en que Ardor y Ethereum gestionan el crecimiento desmesurado de la *blockchain*.

Ethereum (Hinchazón de la Blockchain)

Ardor Frente a la Competencia, Parte 8

Este artículo continua la comparación iniciada en la entrega anterior entre Ardor y Ethereum. En esta ocasión, se explora como se enfrenta cada plataforma al problema del hinchazón de la *blockchain*. Para mi sorpresa, ambas plataformas son más similares al respecto de lo que había pensado, aunque también existen diferencias significativas.

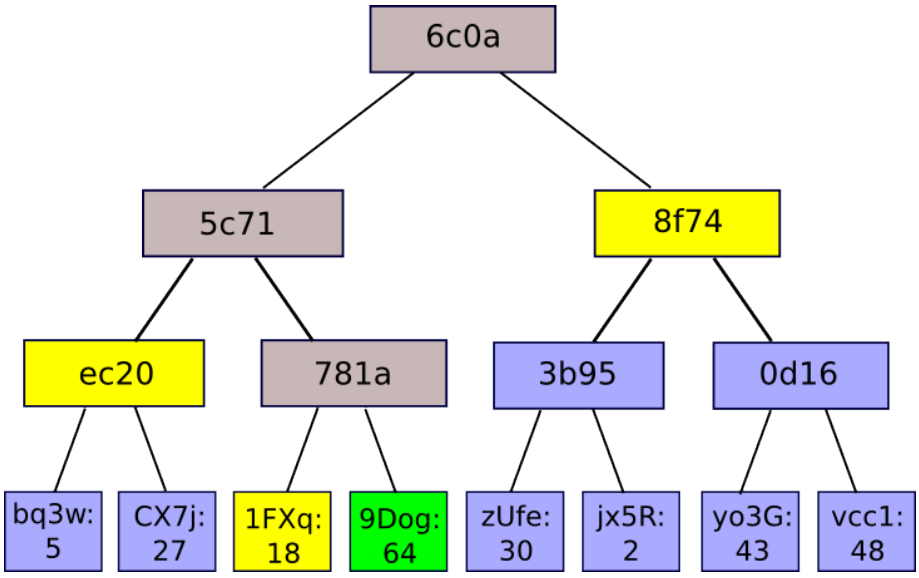
Es clave para esta comparación entender cómo está organizada la *blockchain* de Ethereum.

Estructura de Ethereum

Al igual que Nxt, Ethereum realiza un seguimiento del estado de todas las cuentas con cada bloque nuevo. Y al igual que Bitcoin, Ethereum organiza la información en cada bloque en un arbolo del tipo Merkle (en realidad, en tres de estos árboles) y almacena su *hash* de raíz en la cabecera del bloque.

¿Cómo funciona esto exactamente? Este diagrama que extraído de este artículo nos ayuda a

entenderlo.



Las hojas de un árbol Merkle (es decir, las de la parte de abajo) representan todos los datos reales almacenados. Cada nodo por encima de las hojas almacena un *hash* criptográfico de sus dos hijas. (Nótese que estoy usando el término "nodo" aquí para referirme al elemento del árbol, no a ordenadores en una red. Cada ordenador dela red almacena todo el árbol.)

El diseño tiene la capacidad de que si incluso se cambiase un simple *byte* de una sola hoja, el *hash* de su padre también cambiaría, así como el *hash* del padre de su padre, así sucesivamente hasta el nodo de más arriba, llamado "la raíz Merkle". En cierta manera, la raíz Merkle contiene un resumen de toda la información de las hojas.

Simplemente agrupando conjuntamente todos las hojas y estableciendo su *hash* de una sola vez se produciría un resultado similar, pero la estructura de árbol tiene una segunda propiedad interesante, que hace posible demostrar que una hoja particular del árbol está en el árbol sin necesidad de ver todo el árbol. Por ejemplo, en este diagrama es posible demostrar que la transacción realmente ha sido añadida proporcionando para ello información sobre su hermano (en amarillo), su padre (en gris) y los otros hermanos y padres hasta su camino de vuelta a la raíz. Otro usuario podría calcular los *hashes* relevantes en cada nivel del árbol y a continuación comparar la raíz de Merkle resultante con la que está almacenada en la *blockchain*. Estas "pruebas de Merkle" son la base del sistema de los clientes de verificación de pagos simplificados en Bitcoin (SPV) y también de diversas propuestas de escalado para Ethereum.

Ethereum utiliza tres árboles de Merkle separados para almacenar la información de cada bloque: uno para las transacciones de los bloques; un segundo para una serie de "receptores" de esas transacciones, los cuales representan los efectos de cada transacción; y una tercera para almacenar el estado instantáneo de cada cuenta, incluyendo sus saldos y datos asociados. Almacenar el estado completo del sistema en cada bloque aparenta ser un terrible derroche, pero puesto que cada bloque solo modifica una pequeña porción de hojas, la mayoría de las ramas del árbol de estado no cambian de bloque a bloque y cada nuevo árbol de estado puede

hacer referencia a ramas completas del estado anterior con un coste mínimo. Hay unas pocas complicaciones técnicas sobre este enfoque, y es por ese motivo por el que Ethereum en realidad usa una estructura de datos ligeramente diferente llamada árbol Merkle-Patricia, pero el concepto sigue siendo el mismo.

Los Nodos de Rápida Sincronización de Ethereum

El hecho más importante de todo esto es que las propiedades de las funciones de *hashes*criptográficos aseguran que sea prácticamente imposible construir dos árboles diferentes con la misma raíz. Como resultado, el registro de la raíz Merkle almacenado en la cabecera del bloque en Ethereum es suficiente para establecer que la red validó en ese momento la correspondiente transacción y estado de transacciones.

En otras palabras, incluso aún después de que un nodo haya "olvidado" los contenidos de un bloque antiguo, mientras siga conservando las (mucho más pequeños) cabeceras del bloque en el archivo, puede solicitarle a un nodo completo el contenido completo de un bloque y verificar por si mismo que el nodo completo no ha sido adulterado con ningún dato. (Hay que recordar que aquí y para el resto del artículo vuelvo a referirme a un "nodo" como a un par en la red, no un objeto del árbol de Merkle.)

Este enfoque es el que se utiliza la opción de sincronización rápida del monedero Go Ethereum (geth). Para realizar una sincronización rápida, un nuevo nodo primero descarga y verifica todas las cabeceras de los bloques, empezando por el bloque génesis (en realidad, solo uno de cada 100 cabeceras de bloque debe ser verificada; ver el enlace en GitHub para más detalles). Puesto que las cabeceras continúan la Prueba de Trabajo, este paso es suficiente para mostrar que la red alcanzó un consenso sobre la raíz de Merkle en cada cabecera en el momento en que el bloque fue minado.

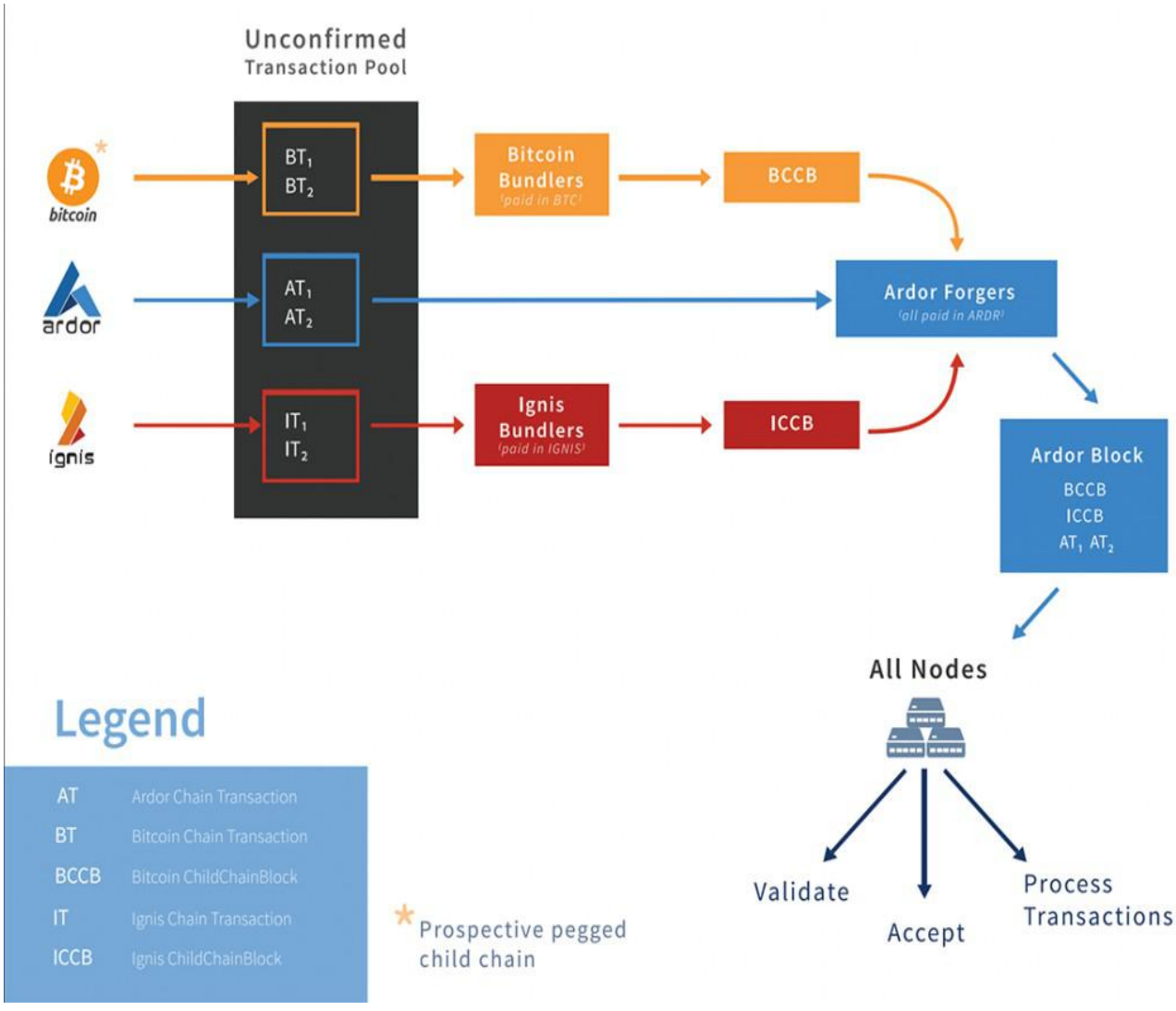
En algún punto del pasado reciente, supongamos que hace 1024 bloques, el nodo recibe una versión completa del estado del árbol proveniente de sus pares y la valida frente a la raíz Merkle en su correspondiente cabecera. Desde ese punto en adelante, el nodo descarga los bloques completos desde los pares y re-ejecuta todas las transacciones hasta que alcanza el bloque más reciente, en cuyo punto simplemente se convierte en un nodo completo ordinario.

Aunque Go Ethereum actualmente no lo soporta, es también posible que los nodos poden continuamente el árbol de estado según avanza el tiempo, manteniendo al mínimo la cantidad de datos de estado que deben ser almacenados.

El Podado de las *Child Chains* en Ardor

Si has analizado el arquitectura de Ardor de cadena madre / cadena hija (*parent chain / child chain*), esta estrategia posiblemente te resulte bastante familiar. Ardor toma un enfoque muy similar respecto a sus *child chains*.

En pocas palabras, la plataforma Ardor consiste en un única cadena madre del tipo Prueba de Participación (*proof-of-stake*), también conocida como Ardor, y una serie de *child chains*. La cadena madre soporta solo unos pocos de tipos de transacciones, básicamente aquellos requeridos para transferir Ardor y para la forja. Las c*hild chains*, por su lado, manejan todo el negocio real que tiene lugar en la plataforma usando las transacciones inteligentes que describí en el artículo previo de esta serie.



Solo la moneda de la cadena madre (ARDR) se puede usar para forjar. Las transacciones que involucran tan solo a las monedas de las *child chains* no afectan los saldos de la moneda de forja, así que no son esenciales para la seguridad de la *blockchain* y no necesitan ser almacenadas permanentemente. Los nodos “agrupadores” especiales en cada *child*

chain recolectan esas transacciones, las agrupan juntas, establecen su *hash*, y envían ese *hash* a la red usando un tipo de transacción especial llamado ChildChainBlock. Incluyen toda la información de la transacción junto a la transacción del tipo ChildChainBlock, así que los forjadores y resto de nodos pueden verificar que las transacciones de la *child chain* son válidas y producen realmente el *hash*indicado, pero los datos en sí mismos no son almacenados en la blockchain y tras un tiempo establecido se pueden podar. Todo lo que queda en la cadena madre es el *hash* de esa información.

Opcionalmente, los nodos especiales de archivo en cada cadena hija pueden almacenar el historial completo de las transacciones de esa *child chain*. En los casos en que el historial sea necesario, los nodos pueden obtenerlo, calcular el *hash* de de las transacciones de agrupación originales y verificar que los *hashes* se corresponden con los que están almacenados en la *blockchain*.

Llegados a este punto, espero que la comparación con la opción de sincronizado rápido de geth haya quedado clara: en ambos casos, los nodos no necesitan almacenar la gran mayoría de los datos de la transacción para ser capaces de verificar que la red aprobó todas esas transacciones en el momento en que fueron hechas. En Ethereum, es suficiente con verificar la prueba de trabajo en la cabecera de los bloques y la exactitud de una raíz Merkle determinada para ser capaz de confiar en el correspondiente árbol de estado. Ardor es ligeramente más complicado porque usa el algoritmo de prueba de participación para el consenso, pero almacenar todo el historial de transacciones de ARDR junto a las transacciones ChildChainBlock asegura que los nodos pueden verificar, desde el bloque génesis, que cada bloque fue forjado por un forjador legítimo.

Comparando Ambos Diseños

Llegados a este punto, espero que estés de acuerdo conmigo en que podemos establecer los siguientes paralelismos entre Ethereum y Ardor:

- Un nodo completo de Ethereum es similar a un nodo de Ardor que almacena el historial completo de cada *child chain*.
- Un nodo de sincronización rápida de Ethereum que pode de manera continua el árbol de estado es similar a un nodo ordinario de Ardor, el cual almacena toda la cadena madre pero poda todos los datos de la cadena hija.
- Ardor ofrece la posibilidad de ejecutar un nodo que almacene toda la cadena madre, además de los datos de las transacciones almacenadas de una sola *child chain*. Esta opción no tiene equivalente en Ethereum.

Por supuesto que estas analogías no son perfectas. En concreto, no hay que pasar por alto que las cabeceras de los bloques de Ethereum son considerablemente más pequeñas que los bloques completos de la cadena madre de Ardor. También he pasado por alto el mecanismo que Ardor utiliza para realizar el seguimiento de las instantáneas del estado completo del sistema y almacenar los *hashes* de esas capturas en la cadena madre.

Aún así, creo que la comparación es de ayuda. El tercer elemento de esta lista es especialmente interesante puesto que parece constituir la mayor diferencia cuantitativa entre ambos diseños. En Ardor, la capacidad para almacenar el historial de las transacciones de cada *child chain* en un juego de nodos de archivo diferenciado permite un tipo de partición vertical de la base de datos de la partición. Puesto que es probable que cada *child chain* sirva a diferentes negocios o proyectos, particionar el juego completo de transacciones por las líneas establecidas por las *child chains* parece una evolución natural. Quizás en Ethereum la mejora analogía sería un diseño dónde el usuario pudiese ejecutar un nodo completo para un único proyecto, como Golem, sin tener que ejecutar simultáneamente nodos para Augur, BAT o cientos de otros proyectos.

Sobre esto último, me llama la atención que los árboles Merkle de Ethereum puedan acomodar de manera natural ese diseño, dónde un "nodo completo de Golem" rastrearía toda la *blockchain*en búsqueda de todas las transacciones que involucrasen GNT, almacenaría todas las pruebas Merkle para esas transacciones y transiciones de estado permanentemente y descartaría los datos restantes. Tengo que admitir que no he pensado suficientemente sobre las implicaciones de esta idea, así que no voy a decir mucho más sobre ello.

En cualquier caso, ni esta hipotética estrategia para Ethereum, ni la arquitectura de cadena madre / cadena hija de Ardor, implica una fragmentación real de la *blockchain*, puesto que en ambos casos cada nodo todavía tiene que procesar todas las transacciones de toda la red. Estos diseño particionan el almacenamiento pero no el ancho de banda o el poder de computación requerido para ejecutar la *blockchain*. Una adecuada estrategia de escalado debería dirigir estos tres cuellos de botella.

Hablando de fragmentación…

Fragmentación

La visión de Ethereum para el escalado de la cadena a largo plazo es la fragmentación, una manera de particionar tanto el almacenamiento, los datos y el procesado de las transacciones. El objetivo es que la mayoría de los nodos de la red tengan que procesar las transacciones de un único fragmento, liberándolos de la carga de tener que validar y almacenar las transacciones que afecta a otros fragmentos.

Ni siquiera voy a tratar de evaluar las propuestas del equipo de Ethereum aquí, puesto que este artículo ya se está extendiendo demasiado, pero si estás interesado en este tema os recomiendo visitar su fantástico documento con Preguntas y Respuestas sobre fragmentación en GitHub.

Sin embargo, la razón por la que he sacado ha relucir la fragmentación es que los desarrolladores de Ardor han sugerido que están explorando maneras de llevar el procesado de las transacciones de las *child chains* a redes dedicadas dentro de la red Ardor. No nos han ofrecido datos técnicos todavía, y voy a evitar especular aquí sobre la manera en que podrían funcionar pero, personalmente, la idea me parece realizable.

Si los desarrolladores pudieran ofrecer más sobre esta idea, entonces la plataforma Ardor se parecería mucho al "diseño básico de una *blockchain* fragmentada", descrita en el documento del equipo de Ethereum. Esa sección del documento describe una serie de nodos "ordenadores" (agrupadores) encargados de recolectar (agrupar) las transacciones de un único fragmento (*child chain*), validándolos y almacenando su *hash* en una "cabecera de ordenación" (transacción del tipo ChildChainBlock) en la cadena madre. Los "super nodos completos" (actuales nodos de la cadena madre) procesarían todas las transacciones de todos los fragmentos; los "nodos *top-level*" (los futuros nodos de la cadena madre) procesarían solo los bloques de la cadena madre, pero no todos los datos de todas las ordenaciones; y los "nodos de un solo fragmento (los futuros nodos de las *child chain*) procesarían todas las transacciones en la cadena madre y un único fragmento.

Casi todas las complicaciones surge de la comunicación entre fragmentos y, como resultado, este diseño funciona mejor cuanto más independientes son los fragmentos. Como ya he mencionado, las *child chains* de Ardor pueden cumplir naturalmente este tipo de partición, dónde cada cadena soportará un proyecto separado, dónde las interacciones entre proyectos están permitidas pero son menos comunes que las transacciones dentro de un proyecto.

Conclusión

En estas fases tempranas, estas ideas son inciertas, claro está. Sin embargo, las posibilidades son emocionantes. El diseño de Ardor ya incorpora el algoritmo de consenso de Prueba de Participación, un objetivo separado que el equipo de Ethereum se ha establecido para si mismo, además de una partición razonable de los datos de la *blockchain*, lo cual es un requerimiento obvio para cualquier solución fragmentada. Entre las ausencias notables en Ardor encontramos las pruebas Merkle, u otras maneras compactas de que las particiones se comuniquen información de estado entre ellas, pero no parece que estas características pudieran ser introducidas en la plataforma a través de un *hard fork*. Los *hashes* del *snapshot* y los *hashes* de los bloques *child chain* que se convertirían raíces Merkle ya están presentes en el protocolo, después de todo.

Pero ¿qué podemos decir sobre el estado actual de ambos proyectos? Quizá el hecho más interesante que he descubierto al investigar y escribir este artículo es que Ethereum realmente escala mucho mejor de lo que yo había pensado. La opción de sincronización rápida de Go Ethereum para los nodos completos permite algunas de las ventajas del diseño de Ardor, y si algún día incorpora el podado del estado del árbol la analogía será incluso más cercana.

Por otro lado, el mayor inconveniente del actual diseño de Ethereum es que todavía tiene que haber nodos completos en algún lugar de la red, y esos nodos completos tienen que almacenar más de 300 GB de la *blockchain* de Ethereum. Conforme sigue creciendo, y en consecuencia, los costes de ejecutar un nodo completo crecen a la par, uno esperaría que la proporción de nodos completos en relación a los nodos ligeros y nodos de sincronización rápida descienda. Como consecuencia, cada nodo completo probablemente acabaría teniendo que gestionar un mayor volumen de solicitudes de otros nodos, mayor incremento de los costes (en términos de ancho de banda y poder computacional) para ejecutar un nodo completo.

Incluso sin fragmentación, el diseño de Ardor mitiga este problema potencial al romper los monolíticos nodos completos de Ethereum en juegos de nodos de archivo, cada uno de los

cuales almacenaría solo el estado de una *child chain*. Sería posible almacenar el historial de varios nodos simultáneamente si así se desease, pero unos pocos nodos, o potencialmente ninguno de ellos, serían requeridos para restaurar el historial completo del ecosistema completo.

No es necesario decir que escalar una *blockchain* es una ardua tarea. Más allá de los diversos proyectos que he analizado para esta serie, Ardor y Ethereum me parece que ofrecen la visión más convincente para el escalado de una cadena. Y mientras espero que ambas triunfen, tengo que admitir, juzgando únicamente por el progreso concreto que cada proyecto ya ha hecho hacia conseguir su objetivo, Ardor me parece que parte de una situación ventajosa.

Comentarios Finales

Ardor Frente a la Competencia

Esta es la entrega final de una serie de artículos que comparan Ardor con otros proyectos *blockchain* con características u objetivos similares. A continuación puedes encontrar las entradas anteriores de esta serie:

- Ardor Frente a Plasma
- Ardor Frente a la Competencia, Parte 1: Lisk
- Ardor Frente a la Competencia, Parte 2: NEM/Mijin/Catapult
- Ardor Frente a la Competencia, Parte 3: IOTA
- Ardor Frente a la Competencia, Parte 4: Waves
- Ardor Frente a la Competencia, Parte 5: Stratis
- Ardor Frente a la Competencia, Parte 6: Komodo/SuperNET
- Ardor Frente a la Competencia, Parte 7: Ethereum (Contratos Inteligentes)
- Ardor Frente a la Competencia, Parte 8: Ethereum (Hinchazón de la Blockchain)

Esta serie comenzó como una breve e informal entrada en Reddit dónde exponía mis reacciones iniciales al *paper* de Plasma. No supe en ese momento que me estaba embarcando en un viaje

por media docena de plataformas más, desde plataformas con cadenas laterales (Lisk, Stratis, de alguna manera Komodo) a plataformas con monedas coloreadas con características exclusivas (NEM, Waves), hasta un proyecto que evita la *blockchain* y usa una estructura completamente diferente (IOTA). Ahora que ha llegado el momento de concluir la serie, con los dos últimos artículos centrados de nuevo en Ethereum, creo que hemos llegado a un buen sitio para concluir nuestro viaje.

En esta serie hemos realizado un gran recorrido y sería imposible resumirlo todo aquí. En su lugar, me gustaría compartir mis reflexiones sobre un tema general que ha surgido recurrentemente al realizar mi investigación sobre estos proyectos.

Escalando con Seguridad

Tal y como he mencionado anteriormente, mi interés principal a lo largo de esta serie ha sido evaluar diferentes maneras de enfrentarse al problema de escalar una *blockchain*. Lo que he descubierto es que existen diversas estrategias, pero la mayoría tienen como contrapartida que afectan a la seguridad. Ciertamente no soy el primero en hacer esta observación, pero creo que debe ser repetido de nuevo en el contexto de esta serie.

En un extremo del espectro, la forma más seguro para lanzar un nuevo proyecto *blockchain* es probablemente emitir un *token* en una *blockchain* existente que se encargue de ofrecerle la seguridad necesaria. Este es el enfoque que buscan las monedas coloreadas de Nxt, NEM, Waves y Ethereum, por ejemplo. Las transacciones que involucran a estos *tokens* son almacenadas directamente en la *blockchain* subyacente y son, por tanto, tan seguras como cualquier otra transacción que tenga lugar en esa *blockchain*.

La parte negativa de este enfoque es que no permite escalar demasiado bien: cada nodo de la red tiene que procesar todas las transacciones de todos los *tokens* de la *blockchain*, incluso si los proyectos a los que esos *tokens* representan no tienen nada que ver el uno con el otro. Además, todos los datos de las transacciones son almacenados para siempre en la misma *blockchain*, haciendo que crezca proporcionalmente al volumen combinado de transacciones de los proyectos que corren en ella.

Así que los llamados métodos de escalado “verticales”, que pretenden que cada nodo realice la misma cantidad de trabajo más rápidamente, o se almacene la misma cantidad de datos de manera más eficiente, son la manera natural para escalar en este modelo. El proyecto Catapult de NEM es un buen ejemplo, puesto que se centra en optimizar el código del cliente completo y el protocolo de comunicación usado en la red. Waves NG, una optimización del protocolo de forjado, es otro ejemplo de lo mismo.

No obstante, este enfoque sobre la manera de escalar tiene límites. Llegará un momento en que, conforme se van añadiendo un mayor número de usuarios y transacciones, este diseño se romperá y la única forma viable para su escalado será del tipo “horizontal”, donde cada nodo de la red procese solo una sub-serie de todas las transacciones.

Una forma razonable de escalar una plataforma *blockchain* horizontalmente es trasladar cada proyecto a su propia *blockchain* independiente, lo que constituye la base de las plataformas de cadenas laterales como Lisk o Stratis. Este enfoque se sitúa en el otro extremo del espectro entre seguridad y escalabilidad: se particiona tanto el poder de computación como el espacio de almacenamiento requerido para ejecutar la plataforma y se permite que existan diferentes nodos para gestionar cada partición, pero este método de escalado implica un descenso de la seguridad. Particularmente, con N proyectos funcionando en una *blockchain* lateral, la cadena lateral más débil esta protegido como mucho por 1/N del total de los mineros o forjadores, y posiblemente nos encontremos muy por debajo de esa cifra, especialmente en sus primeros momentos.

Ardor se sitúa en una posición intermedia en el espectro seguridad-escalabilidad, consiguiendo reducir el espacio de almacenamiento para dar servicio a los proyectos que funcionan en sus propias *child chains* sin tener que reducir su seguridad, pero requiriendo todavía que toda la red procese cada transacción. Será interesante ver los detalles del plan de Jelurida para hacer que el procesado de las transacciones de las *child chains* tenga lugar en subredes dedicadas de la red, lo que ofrecería el escalado computacional y de ancho de banda perdido, pero hasta que llegue este momento, solo podemos especular.

IOTA es un poco un caso especial, puesto que los fundamentos de su diseño son diferentes de los de una *blockchain* en un par de puntos importantes. Sin volver a explicar todo el mecanismo del "consenso a largo plazo" en el tangle, permitidme afirmar que el tangle de IOTA (tal y como está implementado hoy en día) me parece que es principalmente una forma de escalado vertical, con un elemento de escalado horizontal. Cada nodo ve y almacena cada transacción, y a pesar de que los nodos pueden podar continuamente el tangle conforme pasa el tiempo, reduciendo los requisitos de almacenamiento, los "*permanodes*" de la red deben almacenar todo el historial del tangle para que los nuevos nodos puedan arrancar sin tener que depositar la confianza en un tercero. Por otro lado, los nodos no necesitan validar cada transacción, puesto que son capaces de aceptar que las transacciones que se realizaron tiempo atrás en el tangle ya han sido confirmadas por otros nodos de la red, puesto que todas las puntas hacen referencia a ellas.

En otras palabras, IOTA particiona el trabajo de computación requerido para validar las transacciones, pero no el ancho de banda requerido para retransmitirlas o los datos que deben ser almacenados.

A largo plazo, IOTA planea introducir los nodos "manada" para dividir las tareas de validación de las transacciones y almacenamiento del tangle. Esto será una manera de partición horizontal, pero todavía no he sido capaz de encontrar los detalles técnicos, así que en mi opinión pertenece a la misma categoría que el Plasma de Ethereum y las propuestas de fragmentación: una idea que suena plausible, pero que necesita mayor desarrollo antes de que pueda ser aceptada como una solución real.

Sobre esto, me gustaría hacer un apunte final acerca del enfoque de Ardor sobre el escalado: aunque no es la panacea, al menos en estas primeras fases, es importante no subestimar el valor de una arquitectura que ya existe y realmente funciona. Quizá no sea necesario decirlo,

pero los desarrolladores de Ardor no solo están lanzando hipótesis sobre soluciones teóricas a un problema difícil. Han demostrado que pueden diseñar un diseño ambicioso pero realista, implementándolo en un periodo de tiempo razonable y consiguiendo progresar hacia una *blockchain* realmente escalable. No todos los equipos puede afirmar lo mismo, ni aún aquellos que sus ideas iniciales sonaban prometedoras.

Reflexiones Finales

Se podría decir mucho más sobre todos estos proyectos, pero esto tendrá que ser suficiente por ahora. Espero que os hayan gustado estos artículos aunque tan solo sea la mitad de lo que he disfrutado yo escribiéndolos. Como nota personal, me gustaría agradeceros el que hayáis leído hasta aquí y por que hayáis compartido estos artículos con otros entusiastas *blockchain*. Ha sido muy gratificante ver como la gente ofrecía su apoyo, sus comentarios y expresaba todo tipo de reacciones. Me siento honorado y muy agradecido porque os halláis tomado el tiempo para leer mi trabajo.

Si me permitís dejaros con una reflexión de despedida, sería la siguiente: después de todo lo que se ha dicho y hecho, veo un tremendo potencial en varios de estos proyectos, pero estoy especialmente emocionado por Ardor. Su arquitectura de cadena madre / cadena hija solventa simultáneamente dos problemas muy importantes: como enfrentarse al crecimiento desmesurado de la *blockchain*, y como ofrecer una *blockchain* como servicio a los clientes que no cuentan con los recursos o experiencia para lanzar su propia *blockchain*. Nadie sabe que valor económico el mercado asignará a Ardor y a la forma en que soluciona estos problemas pero, en mi humilde opinión, Ardor sale favorecido en las comparaciones con la competencia en ambos puntos. Tengo muchas ganas por ver lo que el futuro nos depara.

Recursos

Jelurida
https://jelurida.com

Whitepaper
https://www.jelurida.com/sites/default/files/JeluridaWhitepaper.pdf

NXTER.ORG
http://nxter.org

Ardor
https://ardorplatform.org

Nxtwiki
https://nxtwiki.org

NxtPlatform
https://nxtplatform.org

ANG
http://www.ardornxt.group

Código fuente
https://bitbucket.org/Jelurida/ardor/src