Cadena de Bloques Æternity

La máquina de oráculo completamente funcional, descentralizada y sin confianza.

BORRADOR

Zackary Hess
zack@aeternity.com

Yanislav Malahov
yani@aeternity.com

Jack Pettersson
jack@aeternity.com

Resumen— Desde la llegada de Ethereum en 2014 ha habido gran interés en las aplicaciones descentralizadas sin confianza (contratos inteligentes). En consecuencia, muchos han intentado implementar sobre una Cadena de Bloques, aplicaciones con información del mundo real. Nosotros creemos que almacenar el estado de la aplicación y el código en la Cadena, está mal por varias razones.

Nosotros presentamos una arquitectura de Cadena de Bloques altamente escalable con un mecanismo de consenso que también es usado para revisar el oráculo. Esto hace al oráculo muy eficiente (económico), porque evita apilar un mecanismo de consenso sobre el otro. Los canales de estado están integrados para incrementar la privacidad y la escalabilidad.

Las fichas en los canales pueden ser transferidas usando los contratos inteligentes completamente funcionales que pueden acceder a las respuestas de los oráculos. Al no almacenar en la Cadena el código de los contratos ni de los canales de estado, podemos hacer contratos inteligentes más fáciles de analizar y rápidos procesar sin pérdidas sustanciales en la funcionalidad *de facto*.

Aplicaciones como mercados para activos sintéticos y mercados de predicción pueden ser implementadas eficientemente en escala global. Muchas partes tienen implementaciones de prueba de concepto codificada en Erlang. Herramientas de desarrollo y aplicaciones esenciales como cartera, sistema de nombres y sistema de identidad también serán distribuidos.

CONTENIDO

I. INTRODUCCIÓN

La intención de este documento es dar una vista previa de la arquitectura de la Cadena de Bloques de Æternity y sus posibles aplicaciones. Documentos más detallados serán lanzados en el futuro, específicamente para los mecanismos de consenso y gobernanza. Sin embargo, cabe señalar que nuestra arquitectura es holística; todos los componentes se unen y sinergizan, de forma modular.

El resto de este documento se divide en cuatro partes. Primero, presentaremos y discutiremos las ideas teóricas fundamentales que son la base de nuestra arquitectura. Segundo, hablaremos de las aplicaciones esenciales incluidas, otros posibles casos de uso y daremos pistas sobre cómo usar la plataforma como desarrollador. Tercero, presentaremos la implementación de la prueba de concepto actual escrita en Erlang. Concluiremos con una conversación, incluyendo las posibles vías que podemos tomar en el futuro y comparaciones con otras tecnologías.

A. Antecedentes

Las Cadenas de Bloques, principalmente Bitcoin, han mostrado una nueva forma de realizar el intercambio de valores en la Internet [1]. Esto ha traído una serie de avances prometedores: Ethereum demostró una forma de escribir contratos inteligentes Turing-completos asegurados por la arquitectura de cadena de bloques [2]; Truthcoin creó herramientas para crear oráculos en las cadenas de bloques [3], mientras que GroupGnosis y Augur mostraron cómo hacerlos más eficientes[4]; Casey Detrio demostró cómo crear mercados en las Cadenas de Bloques [5]; Namecoin mostró cómo hacer el equivalente distribuido de un servidor de nombres dominio [6]; Factom mostró cómo una cadena de bloques que que almacena hashes puede ser usada como prueba de existencia de cualquier información digital [7].

Estas tecnologías son muy prometedoras con respecto a proveer servicios legales y financieros de primera clase para todos. Sin embargo, hasta ahora han fallado en formar un todo unificado que realmente cumpla la promesa. Específicamente, todas las soluciones hasta ahora han carecido de al menos uno delos siguientes aspectos: gobernanza, escalabilidad, seguridad de codificación y acceso barato a información del mundo real [falta fuente]. Æternity busca mejorar la tecnología de punta en todos estos aspectos.

II. CADENA DE BLOQUES ÆTERNITY

Nosotros creemos que la falta de escalabilidad, seguridad de codificación y acceso barato a información del mundo real de las plataformas actuales de contratos inteligentes, vienen dados por tres problemas fundamentales. *Primero*, el diseño de estado predominante hace a los contratos inteligentes escritos para la plataforma difíciles de analizar [1] y ese diseño de estado combinado con órdenes secuenciales de transaccione, complica la escalabilidad [falta fuente]. *Segundo*, el alto costo de traer información del mundo real al sistema de forma descentralizada y sin confianza, complica o impide directamente la realización de muchas aplicaciones prometedoras [falta fuente]. *Tercero*, las plataformas son limitadas en sus capacidades de actualizarse a sí mismas, para adaptarse a nuevos conocimientos tecnológicos o económicos. Nosotros creemos que cada uno de estos tres problemas tiene claras vías de solución y deben ser exploradas.

Primero, investigaciones recientes sobre la tecnología de canales de estado sugiere que para muchos casos de uso, mantener el estado en la cadena, no es necesario [falta fuente]. En la mayoría de los casos, es completamente posible almacenar toda la información en canales de estado y solo usar la cadena de bloques para establecer cualquier resultado económico del intercambio de información y como alternativa en el caso de una disputa. Esto sugiere un enfoque distinto a la arquitectura de cadena de bloques en el que los contratos inteligentes Turing-completos existen en canales de estado y no en la cadena. Esto aumenta la escalabilidad ya que todas las transacciones se vuelven independientes y pueden entonces, ser procesadas en paralelo. Adicionalmente, esto quiere decir que los contratos nunca se escriben en el estado compartido, simplificando enormemente sus pruebas y verificación. Nosotros creemos que este diseño enfatiza que las cadenas de bloques sirven más para lógica financiera que para almacenamiento de información; ya existen soluciones de almacenamiento descentralizado que complementan perfectamente a las cadenas de bloques.

Segundo, aplicaciones como Augur han intentado traer información del mundo real a la cadena de bloques en forma descentralizada—esencialmente, en el proceso construyeron un mecanismo de consenso dentro de los contratos inteligentes [8], en lugar de utilizar el mecanismo de consenso de la cadena de bloques subyacente. Esto conduce a tener ineficiencias pero no incrementa la seguridad. La conclusión natural a esto es generalizar el mecanismo de consenso de la cadena de bloques para que pueda entregar información no solo del estado interno, pero también del estado del mundo externo. También puede afirmarse que el mecanismo de consenso de la cadena de bloques decide el resultado de ejecutar lo que la teoría de la complejidad apoda Máquina de Oráculo: una máquina teórica que es más poderosa que una Máquina Turing porque tiene respuestas a algunas preguntas que no necesariamente pueden ser computadas, como “¿Quién ganó el juego X de fútbol?” [falta fuente].

Tercero, es natural que el mecanismo de consenso también pueda ser usado para determinar los parámetros del sistema. Esto le

[1] La dificultad de analizar los contratos de estado fue muy claramente demostrada por el error de reentrada que derribó "The DAO". Esto sucedió a pesar de que el código había sido auditado por varios de los creadores de Ethereum, así como por la comunidad en general

permite adaptarse a cambios en las condiciones externas, así como adoptar nuevos desarrollos e investigaciones en el campo.

El resto de esta sección presenta la cadena de bloques de Æternity con mayor detalle, comenzando con un breve repaso de las cuentas, fichas, nombres y la estructura de los bloques. Esto va seguido de una explicación de nuestro enfoque en los canales de estado y los contratos inteligentes y entonces una discusión sobre cómo el mecanismo de consenso de la cadena de bloques puede ser usado para crear un mecanismo de oráculo eficiente y gobernar el sistema. Finalmente, hablamos sobre la escalabilidad desde varios ángulos distintos.

A. Fichas, cuentas y bloques

A pesar de "no tener estado" desde el punto de vista de los desarrolladores de contratos, la cadena de bloques de Æternity mantiene registro de varios componentes predefinidos de estado. Vamos a explicarlos, así como el contenido de cada bloque. Para mantenerlo simple, esta sección asume que cada nodo mantiene registro de toda la cadena de bloques. Posibles optimizaciones se describen en la sección II-E.

A.1) Ficha de acceso, Aeon: El uso de la cadena de bloques no es gratuito, requiere que el usuario gaste una ficha llamada aeon. Los Aeon son usados como medio de pago por cada recurso que uno consuma en la plataforma, también es la base de las aplicaciones financieras implementadas en la plataforma.

La distribución de aeon en el bloque génesis será determinado por un contrato hospedado en Ethereum. Más aeon serán creados a través de minería.

Todas las comsiones serán pagadas en aeon y todos los contratos inteligentes se establecen en aeon.

A.2) Cuentas: Cada cuenta tiene una dirección y un balance en aeon, también tiene un nonce que incrementa con cada transacción y el peso de su última actualización. Cada cuenta también debe pagar una pequeña comisión por la cantidad de tiempo que esté abierta. El costo de creación y mantenimiento de cuentas previene el spam y desincentiva la inflación del estado. La remuneración por eliminar cuentas incentiva la recuperación de espacio.

A.3) Sistema de Nombres: Muchos sistemas de cadenas de bloques sufren de direcciones ilegibles para sus usuarios. En línea con el trabajo Aaron Swartz y Namecoin, Æternity incluye un sistema de nombres que es descentralizado y seguro, mientras soporta nombres humanamente legibles [9]. El estado de la cadena de bloques incluye una asignación de cadenas de caracteres humanamente legibles a arreglos de bytes de tamaño fijo. Estos nombres se pueden usar para señalar cosas tales como direcciones de cuentas en Æternity o hashes como los de los árboles Merkle.

A.4) Contenido de los Bloques: Cada bloque contiene los siguientes componentes:

- El hash del bloque anterior.
- Un árbol Merkle de transacciones.
- Un árbol Merkle de cuentas.
- Un árbol Merkle de nombres.
- Un árbol Merkle de canales abiertos.
- Un árbol Merkle de oráculos que no han respondido sus respectivas preguntas.
- Un árbol Merkle de respuestas de oráculos.
- Un árbol Merkle de comprobaciones (pruebas) de oráculos.
- La entropía actual en el generador aleatorio de números.

El hash del bloque anterior es requerido para mantener un orden en la cadena de bloques. El árbol de transacciones contiene todas las transacciones que están incluidas en el bloque actual. Con la excepción del árbol de votos de consenso, todos los árboles están completamente bajo consenso: si un árbol es cambiado de un bloque para el siguiente, este cambio debe ser por una transacción en el árbol de transacciones del nuevo bloque y una prueba Merkle de la actualización debe ser incluida en el árbol de comprobaciones del bloque. El propósito de los tres árboles restantes será respondido en las siguientes secciones.

B. Canales de estado

Uno de los desarrollos más interesantes recientemente en el espacio de las cadenas de bloques, es el de los canales de estado. Ellos operan en el principio básico de que en la mayoría de los casos, solo la gente afectada por una transacción, necesita saber de ella. En esencia, las partes de un intercambio inician un estado en la cadena de bloques, por ejemplo un contrato en Ethereum o una entrada de múltiples contraseñas en Bitcoin. Ellos entonces simplemente envían actualizaciones firmadas a este estado entre cada uno. El punto clave es que cualquiera de ellos podría usarlas para actualizar el estado en la cadena de bloques, pero en la mayoría de los casos, no lo hacen. Esto permite que las transacciones se ejecuten tan rápido como la información pueda ser transmitida y procesada por las partes, en lugar de esperar que la transacción sea validada—y potencialmente finalizada— por el mecanismo de consenso de la cadena de bloques.

En Æternity, la única actualización de estado que puede ser establecida en la cadena de bloques es una transferencia de aeon y los únicos aeon que pueden transferirse son los que las partes del intercambio ya han depositado en el canal. Esto hace a todos los canales independientes entre sí, lo que tiene el beneficio inmediato de que todas las transacciones relacionadas a canales de estado, pueden ser procesadas en paralelo, aumentando enormemente el desempeño de las transacciones.

```
1 macro Oro f870e8f615b386aad5b953fe089 ;
2
3 Oro oracle
4 if 0 1000 else 0 0 end
5 0
```

Fig. 1. Un contrato simple codificando una apuesta en el precio del oro. El leguaje usado es el (similar a Forth) Chalang, que será presentado en la siguiente sección IV-A

La cadena de bloques es usada solo para establecer el resultado final o para resolver conflictos que sobrevengan, vagamente semejante al sistema jurídico. Sin embargo, por el hecho de que el comportamiento de la cadena de bloques será predecible, no habrá beneficio ni interés en disputar el resultado esperado de un canal de estado; actores maliciosos están incentivados a comportarse de manera correcta y solo establecer el estado final en la cadena de bloques. Todo puesto en conjunto, incrementa la velocidad de transacciones y el volumen por varios órdenes de magnitud, así como la privacidad.

B.1) Contratos Inteligentes: A pesar de que el único estado que puede ser establecido en la cadena es una transferencia de aeon, Æternity aún presenta un máquina virtual Turing-completa que puede ejecutar "contratos inteligentes". Los contratos en Æternity son estrictamente son estrictamente acuerdos que distribuyen fondos según algunas reglas, lo que mantiene un marcado contraste con los contratos de tipo entidad de, por ejemplo, Ethereum. Dos de las diferencias prácticas más notables son que de forma predeterminada, solo las partes involucradas conocen la existencia de su contrato y que solo las partes que tienen abierto un canal de estado pueden crear un contrato válido. Si las partes aceptan un contrato, ellas lo firman y mantienen copias para futuras referencias. Solo se envía a la cadena de bloques si su resultado es disputado, en cuyo caso el código sólo se almacena como parte de la transacción presentada, nunca en ningún otro estado. Si esto ocurre, la cadena de bloques distribuye las fichas de acuerdo al contrato y cierra el canal.

Como un ejemplo, la Fig. 1 muestra un contrato muy simple que codifica una apuesta en el precio del oro para una hora determinada. En la línea 1, la macro *Oro* guarda el identificador del oráculo en cuestión, que devolverá *true* (verdadero) si el precio del oro está debajo de 38$/g para el 1 de diciembre de 2016. El cuerpo del contrato es mostrado entre las líneas 2 y 4: primero colcamos el identificador del oráculo *Oro* y lo llamamos usando *oracle*, lo que dejará la respuesta del oráculo en la parte superior del código. Usamos esto para hacer una ramificación condicional: Si el oráculo devuelve *true*, nosotros colocamos *0* y *1000* al código, indicando que *0* aeon deben ser quemados y *1000* aeon deben ir al primer participante en el canal. De lo contrario, colocamos *0* y *0*, con el segundo *0* indicando que el otro participante recibe todos los aeon en el canal. Finalmente colocamos *0*, que es tomado como el nonce de este canal de estado. En el uso real, el nonce se generaría durante la ejecución.

Una cosa importante a tomar en cuenta es que los contratos en Æternity no mantienen ningún estado en sí mismos. Cada estado es mantenido por las partes del intercambio y enviadas como entrada en la ejecución. Cada contrato es en esencia una *función pura* que toma una entrada y entrega un nuevo canal de estado como salida[2]. Los beneficios del uso de funciones puras en el desarrollo de software en general y particularmente en el desarrollo de aplicaciones financieras, ha sido extensamente documentado por la academia y la industria por décadas [10] [falta fuente].

```
1 : hashlock
2 swap
3 hash
4 == ;
```

Fig. 2. Un hashlock simple.

```
1 macro Acuerdo a9d7e8023f80ac8928334 ;
2
3 Acuerdo hashlock call
4 if 0 100 else 0 50 end
5 1
```

Fig. 3. Usando el hashlock para enviar fichas de manera confiable a traves de un intermediario.

a) Interacción de contratos y contratos multi-pasos: Aunque todos los contratos carecen de estado y se ejecutan independientemente unos de otros, los estados y la interacción entre contratos aún pueden ser logrados a través del *hashlocking* [falta fuente]. Un hashlock sencillo es mostrado en la fig. 2. En la línea 1, definimos una función llamada hashlock que espera que el código contenga un hash *h* y un secreto *s.* Los intercambia en la línea 2, para hacer el hash al *secreto* en la línea 3, antes de llamar al operador de la igualdad en el *hash(v)* y en la línea 4. Esto devuelve *true* si el *secreto* es una preimagen del hash. Esta función puede ser usada para predecir la ejecución de ramas de código en diferentes contratos con la existencia del mismo valor de *secreto*.

Como un simple ejemplo de uso, los hashlocks hacen posible para los usuarios que no compartan un canal de estado, enviarse aeon entre ellos de manera confiable, mientras haya una vía de canales entre ellos. Por ejemplo, si Alice y Bob tienen un canal y Bob y Carol tienen un canal, entonces Alice y Carol pueden intercambiar a través de Bob. Ellas hacen esto creando dos copias del contrato mostrado en la fig. 3, una por cada canal. El *Acuerdo* en

la línea 1 es el hash de un *secreto* que Alicia elige. En la línea 3 lo colocamos en el código y llamamos a la función hashlock. Cuya rama del *if* que se ejecute dependerá del valor devuelto de hashlock. Una vez que esos contratos hayan sido firmados por todas las partes, Alice revelará el secreto, permitiendo a Bob y Carol usarlo para reclamar sus aeon.

El hashlocking también puede ser usado para, por ejemplo, crear juegos multijugadores en los canales, como se muestra en la fig. 4. Todos hacen un canal con el administrador del juego, que publica el mismo contrato a cada canal. Digamos que estamos en el estado 32 del juego, definido por la función *State32* y queremos actualizar todos los canales al estado 33 de manera confiable y simultánea. Cuando el administrador del juego revela el *secreto* hace que todos los canales se actualicen al mismo tiempo.

```
1 macro Acuerdo a9d7e8023f80ac8928334 ;
2
3 Acuerdo hashlock call
4 if State33 else State32 end
5 call
```

Fig. 4. Un ejemplo simplificado del uso del hashlock para jugar una partida multijugador en los canales.

b) Ejecución medida: La ejecución de los contratos es medida de forma similar a la usada por el "gas" en Ethereum. Pero Æternity usa dos recursos distintos para su medición, uno para tiempo y otro para espacio. Ambos son pagados por los aeons usados por la parte que solicita la ejecución.

Esto puede ser visto como indeseable, porque probablemente sea otra parte la que ocasiona la necesidad de que la cadena de bloques resuelva la disputa, en primer lugar. Sin embargo, mientras no todo el dinero en el canal sea usado para apostar, esto puede ser efectivamente anulado en el código del contrato, ya que tiene la habilidad de redistribuir los fondos de una parte a la otra. De hecho, generalmente es una buena práctica evitar el uso de los fondos en un canal para transacciones, porque eso desincentiva a la parte perdedora a cooperar en el cierre de un canal.

[2]Cabe señalar que dado que los contratos pueden leer las respuestas de los oráculos y algunos parámetros del entorno, no son funciones completamente puras. Sin embargo, las respuestas del oráculo nunca cambian una vez que han sido proporcionadas y se puede argumentar que se deben a la riqueza computacional de la máquina del oráculo, en lugar de ser una impureza. Los parámetros ambientales se consideran un "mal necesario" e idealmente serán compartimentados apropiadamente por lenguajes de alto nivel.

B.2) Ejemplo: Vamos a poner todas esas ideas sobre la tierra. En la práctica, si Alice y Bob quieren transferir usando un canal de estado en Æternity, ellos deben realizar el siguiente procedimiento.

1) Alice y Bob firman una transacción que especifica cuánto dinero está depositando cada uno dentro del canal y lo publican a la cadena de bloques.
2) Una vez que la cadena de bloques haya abierto el canal, ellos podrán abrir nuevos canales de estado, enviárselos entre ellos y firmarlos. Los canales de estados pueden ser una nueva distribución de los fondos en el canal o un contrato que determine una nueva distribución. Cada uno de estos canales de estado tiene un nonce creciente y están firmados por ambas partes, así, si una disputa se sobreviene, el último estado válido puede ser enviado a la cadena de bloques, que lo ejecuta.

3) El canal puede ser cerrado de dos formas distintas:

a) Si Alice y Bob deciden que han terminado de transferir y están de acuerdo con sus balances finales, ellos firman una transacción firmando eso y lo envían a la cadena de bloques, que cerrará el canal y redistribuirá el dinero en el canal correctamente.

b) Si Alice se niega a firmar una transacción de cierre por cualquier razón, Bob puede enviar el último estado firmado por ambos y solicitar que se cierre el canal usando este estado. Esto inicia una cuenta regresiva. Si Alice cree que Bob está siendo deshonesto, ella tiene la oportunidad de publicar un estado con un nonce más alto que ambos hayan firmado antes de que termine el conteo regresivo. Si ella lo hace, el canal se cierra inmediatamente. De lo contrario, lo cerrará cuando la cuenta regresiva termine.

C. Mecanismo de consenso

Dado que la mayoría de la computación ocurrirá en los canales y no en la cadena, usaremos un mecanismo de consenso de finalidad lenta. Esto hace que el consenso sea más asequible. El mecanismo de consenso usará una pequeña cantidad de prueba de trabajo, de modo que el número de posibles forks se mantendrá pequeño y puede caber en la memoria de todos. Decidir qué fork utilizar será fuertemente influenciado por los resultados de los mercados de predicción, que nos dicen cuáles son las mejores decisiones. Proponer y analizar nuestro mecanismo está fuera del alcance de este documento y se hará en un documento complementario inminente.

Avance: El mecanismo de consenso en Æternity tiene un rol poco común. Además de acordar nuevos bloques para la cadena de bloques, también acuerda las respuestas a las preguntas del oráculo y los valores de los parámetros del sistema. En particular, el mecanismo de consenso puede cambiarse a sí mismo. Sin embargo, cabe señalar que esto no está del todo libre de problemas. Por ejemplo, si se utilizara un simple mecanismo de prueba de trabajo (Proof-of-Work), sería bastante barato sobornar a los mineros para que corrompieran el oráculo. Por lo tanto, Æternity va a usar un

nuevo algoritmo de prueba de exitencia (Proof-of-Stake), con una adición de prueba de trabajo más pequeña, aprovechando los beneficios de ambos. Independientemente de esto, PoW va a ser usado para emitir nuevos aeón.

C.1) Oráculos: Una característica crucial para la mayoría de los contratos, ya sea codificada como texto o como código, es la capacidad de hacer referencia a los valores del entorno, como los precios de los diferentes bienes o si se produjo o no un determinado suceso. Un sistema de contratos inteligentes sin esta capacidad, es esencialmente un sistema cerrado y posiblemente no muy útil. Este es un hecho generalmente aceptado y ya hay varios proyectos que intentan traer datos externos a la cadena de bloques de manera descentralizada[8]. Sin embargo, para decidir si un hecho suministrado es cierto o no, éstos requieren esencialmente la implementación de un nuevo mecanismo de consenso sobre el mecanismo de consenso.

Ejecutar dos mecanismos de consenso uno sobre el otro es tan costoso como correr ambos por separado. Además, no aumenta la seguridad, porque el menos seguro todavía puede ser atacado y forzado a producir valores "*falso*". Por lo tanto, proponemos combinar los dos mecanismos de consenso en uno, esencialmente reutilizando el mecanismo que utilizamos para acordar el estado del sistema, para acordar también el estado del mundo exterior.

La forma en que esto funciona es la siguiente. Cualquier poseedor de aeons puede lanzar un oráculo comprometiéndose a contestar una pregunta de sí/no. Al hacerlo, también necesitan especificar el período de tiempo durante el cual se puede responder a la pregunta, que puede comenzar ahora o en algún momento en el futuro. El usuario que lance el oráculo debe depositar aeon en proporción a la duración del plazo, que será devuelto si el usuario proporciona una respuesta que se acepta como verdad, de lo contrario se quema. La cadena de bloques genera un identificador único para el oráculo que se puede utilizar para obtener la respuesta una vez que esté disponible.

Una vez que llega el momento de responder la pregunta, el usuario que lanzó el oráculo puede proporcionar una respuesta de forma gratuita. Una vez que el lanzador del oráculo ha proporcionado una respuesta o hasta que una cantidad de tiempo ha pasado, cualquier otro usuario puede presentar contrareclamos depositando la misma cantidad de aeon. Si no se han presentado reclamaciones contrarias al final del plazo, la respuesta suministrada por el usuario que lanzó el oráculo se acepta como verdad y el depósito se devuelve. Si se presentan contrareclamos, se utilizará el mecanismo de consenso de los bloques para responder al oráculo. Esto es más caro, pero como sabemos que podemos tomar al menos uno de los dos depósitos de seguridad, podemos usarlo.

D. Gobernanza

La gobernanza de los sistemas basados en cadenas de bloque ha sido un gran problema en el pasado. Cada vez que se necesita hacer una mejora al sistema, esto requiere un hard fork, que conduce generalmente a grandes discusiones entre todos los poseedores de valor. Incluso las cosas simples, como corregir una variable arbitrariamente establecida en el código fuente, como hemos visto con el debate de tamaño de bloque en Bitcoin, parecen ser muy difíciles en un sistema donde los incentivos de los usuarios no están alineados con los tomadores de decisiones y donde no hay una ruta de actualización clara. También hemos visto decisiones de gobernanza más complicadas, como el arreglo de un único BUG de contrato inteligente en "The DAO", que requirió la intervención rápida de los desarrolladores del sistema.

El principal problema de estos sistemas es fácilmente identificable: el proceso de toma de decisiones para una mejora o cambio de protocolo no está bien definido y carece de transparencia. El sistema de gobernanza de Æternity es parte del consenso. Utiliza los mercados de predicción para funcionar de la manera más eficiente y transparente posible.

Por otra parte, el mecanismo de consenso se define por una serie de variables que determinan cómo funciona el sistema y que van siendo ligeramente actualizadas por cada nuevo bloque. Desde cuánto cuesta hacer transacciones o preguntar a un oráculo, a modificaciones de valores de parámetros fundamentales como el tiempo del bloque.

Al usar mercados de predicción para modificar las variables que definen el protocolo, los usuarios pueden aprender a mejorar el protocolo de manera eficiente. Al usar mercados de predicción para decidir sobre potenciales hard forks, podemos ayudar a la comunidad a llegar a un consenso sobre qué versión del código usar. Cada usuario elige por sí mismo la métrica que busca optimizar, pero una simple estrategia por defecto sería maximizar el valor de sus activos.

E. Escalabilidad

E.1) Árboles de fragmentación: La arquitectura que se ha presentado hasta ahora es altamente escalable. Es posible ejecutar la cadena de bloques incluso cuando cada usuario sólo realiza un seguimiento de la parte del estado de la cadena de bloques que les interesa e ignora los datos de todos los demás. Al menos una copia del estado es necesaria para que los nuevos usuarios estén seguros sobre el subestado que les interesa, pero podemos fragmentar estos datos de forma arbitraria en muchos nodos para que la carga de cada nodo sea arbitrariamente pequeña. Los árboles Merkle se utilizan para probar que un subestado es parte del estado[11].

Es fácil imaginar un escenario donde ciertos nodos se especializan en el seguimiento de los árboles y se les paga por insertar y consultar.

E.2) Clientes livianos: Los clientes ligeros no descargan los bloques enteros. En primer lugar el usuario da a su cliente un hash

en el historial del fork que prefieren, una técnica también conocida como *subjetividad débil* [12]. Entonces el cliente sólo sabe descargar forks que incluyen un bloque con ese hash. El cliente sólo descarga los encabezados de los bloques. Los encabezados son mucho más pequeños que bloques completos; muy pocas transacciones son procesadas. Por simplicidad, no hicimos mención alguna de los encabezados de bloques al discutir la estructura de bloques en la sección II-A.4, pero contienen lo siguiente:

- El hash del bloque anterior.
- El hash raíz de todos los árboles de estado.

E.3) Canales de estado y paralelismo: Los canales de estado tienen inmenso rendimiento y la mayoría de las transacciones dentro de ellos nunca se ejecutan o incluso se registran en la cadena de bloques. Además, los canales no escriben en ningún estado compartido en cadena, por lo que todas las transacciones que realmente se registran en la cadena de bloques se pueden procesar en paralelo. Dado que la mayoría del hardware de consumidor vendido hoy en día tiene al menos cuatro núcleos de procesamiento, esto tiene el efecto inmediato de que el rendimiento de transacción se multiplica por aproximadamente un factor de 4.

Además, el hecho de que nunca habrá interacción concurrente compleja sugiere que sharding esta arquitectura blockchain debería ser relativamente fácil. Dado que el sharding de cadena de bloque es todavía bastante experimental, hemos elegido deliberadamente no perseguir ninguna técnica de sharding en el diseño inicial de Æternity. Sin embargo, si esto cambia en el futuro, Æternity debería ser una de las cadenas de bloques más fáciles cortar.

E.4) Transacciones pro segundo en un requerimiento de memoria dado: Las variables que definen el protocolo están siendo constantemente actualizadas por el consenso. A partir de sus valores iniciales por defecto, podemos calcular la tasa inicial predeterminada de transacciones por segundo.

```
Tenga en cuenta que este es un borrador y
probablemente cambiará

 Definimos las siguientes variables para los
            siguientes cálculos:

B = bloque\_tamaño en bytes
F = bloques\_a\_finalizar
R = tiempo\_a\_finalizar en segundos
T = tamaño de transacción en bytes

transacciones por segundo = B * F / ( T * R )

B = 1000000 bytes = 1 megabyte por bloque
F = 24 * 60 * 2 bloques por día
R / F = 30 segundos por bloque
R = 24 * 3600 segundos por día
T = 1000 bytes por transacción

1000000 * 24 * 60 * 2 / 1000 / 24 * 3600
= 1000000 / 1000 / 30
= aprox. 32 transacciones por segundo
 (suficientemente rápido para registrar cada
 humano en los próximos 8 años
```

Para operar un nodo, tenemos que mantener una copia de todos los bloques desde la finalidad, y tenemos que ser capaces de grabar 100 veces más información, en caso de que haya un ataque. Estimando que la finalidad es de 2 días, entonces habría 5760 bloques hasta finalidad. Así que el requisito de memoria es 5760 * un megabyte * 100 = 576000 megabytes = 576 gigabytes. Cuando no hay un ataque sucediendo, uno sólo necesitaría alrededor de 5,76 gigabytes para almacenar los bloques.

III. APLICACIONES

La naturaleza sin estado de los contratos inteligentes de Æternity facilita la construcción de las siguientes aplicaciones en la cadena de bloques de Æternity. Es especialmente adecuado para casos de uso de gran volumen.

A. Esenciales de cadenas de bloques:

Lo esencial de la cadena de bloques son necesarios como el aeón, billeteras, nombres y conceptos relacionados. Modularizan los componentes reutilizables que se pueden utilizar como los cimientos de las aplicaciones y se pueden mejorar.

A.1) Identidades: Cada cuenta tendrá un número de identificación único asociado. Los usuarios pueden registrar nombres únicos y vincular nombres a la raíz Merkle de una estructura de datos. La estructura de datos puede contener su identificación única, así como otra información sobre su cuenta. Nuestro objetivo es utilizar el formato JSON de Schema.org para representar cosas como personas o empresas. [13]

A.2) Billetera: Una billetera es una pieza de software que se utiliza para interactuar con Æternity. Una billetera maneja las claves privadas para el aeón y crea y firma transacciones. Se puede usar la cartera para enviar transacciones de canal y utilizar aplicaciones en la red de canales.

A.3) Prueba de existencia: Un tipo de transacción permite la publicación del hash de cualquier dato. Los participantes del sistema pueden usar los encabezados para comprobar que los datos existían en ese momento.

B. Aplicaciones de canales de estado

Los contratos inteligentes en canales de estado son perfectos para microservicios en la web que requieren un alto rendimiento de transacciones.

B.1) Toll API: La mayoría de las API que existen hoy en día están públicamente disponibles para que cualquier persona las llame, o bien están protegidas por un esquema de nombre de usuario-contraseña o tokens de acceso único. Los canales de pago permiten un nuevo tipo de API, donde se paga por cada llamada a la API, posiblemente cada solicitud HTTP. Pagar para acceder a una API resuelve problemas de DDoS y facilita la creación de APIs de alta calidad que siempre están disponibles.

Las respuestas de la API que requieren un pago son fundamentales para la creación de tipos de negocios hasta ahora imposibles y pueden desempeñar un papel importante en el surgimiento de la economía descentralizada. Ellos crean incentivos para que los propietarios de la información hagan accesibles públicamente los datos que de otra forma serían confidenciales.

B.2) Financiación masiva asegurada: Podemos implementar crowdfunding asegurado usando contratos de seguro dominantes.

Estos son los contratos inteligentes que se utilizan para recaudar dinero para un bien público, como un nuevo puente, una escuela o un mercado. Contratos de aseguramiento dominante difieren de los contratos de aseguramiento tradicionales como Kickstarter, en el sentido de que lo convierten en una estrategia dominante para participar. Si el bien no es financiado, todos los participantes reciben su aeón de vuelta más los intereses, por lo que están asegurados contra la reducción de su liquidez sin recibir el bien. Usando un oráculo, podemos asegurar que al proveedor del bien o servicio sólo se le pague si el bien o el servicio se entrega realmente.

B.3) Intercambios atómicos entre cadenas: Los intercambios atómicos entre cadenas permiten un intercambio sin confianza entre aeón y bitcoins[14], [15]. Estos se pueden implementar usando un hashlock, que bloquea las transacciones en ambas cadenas de bloques bajo el mismo valor.

B.4) Activos de valor estable y replicación de cartera: Podemos utilizar contratos inteligentes para programar activos sintéticos que se mantienen casi al mismo precio que un activo del mundo real. Por ejemplo, podríamos hacer un activo que se mantenga al mismo precio que el oro. Los derivados sintéticos se crean en pares iguales y opuestos. Para que un usuario tenga un activo que se mueva con el oro, un usuario diferente tendrá que tener un activo que se mueva inversamente al oro. Por ejemplo, Alice podría hacer un contrato con Bob para que Alice posea 1 gramo de oro. Fuera del dinero del contrato, el valor en aeon de un gramo de oro irá a Alice, y el dinero restante va a Bob. El contrato tiene una fecha de vencimiento para cuando será medido el precio del oro, y los fondos distribuidos a Alice y Bob.

B.5) Contratos de eventos: Los contratos de evento pagan cuando sucede un evento y no pagan cuando un evento no sucede, según lo que dice el oráculo. Aparte de ser interesante en sí mismos, estos pueden ser utilizados por varias aplicaciones diferentes:

a) Asuguradoras: Podemos utilizar contratos de eventos para implementar seguros. Por ejemplo, los boletos caros del espectáculo musical pueden llegar a ser inútiles si el clima va mal. Sin embargo, si el espectador recibe dinero si el oráculo decide que llovió el día del evento, la inversión puede ser protegida para que uno pueda darse el lujo de encontrar una alternativa emocionalmente adecuada. Un poco más en serio, los agricultores están a menudo interesados en el número total de pulgadas de lluvia en una temporada. Podemos asegurarlos contra la sequía que pueda marchitar sus cultivos.

b) Denuncias: Los contratos de eventos también pueden usarse para incentivar la revelación de información confidencial. Por ejemplo, podríamos apostar por el evento "La información que indica que la compañía A ha utilizado plaguicidas ilegales fue liberada el o antes del 24 de enero de 2017". Cualquier persona con

acceso a tal información se vería incentivada a primero apostar a que el acontecimiento ocurrirá y después lo lanzará.

B.6) Mercados de Predicción: Un mercado de predicción funciona permitiendo a los usuarios apostar si un evento futuro ocurrirá. Del precio de las apuestas podemos predecir la probabilidad futura[3], [8], [16]. Ellos son la manera más precisa de medir el futuro a un precio determinado. Una vez que el evento ha ocurrido, el mercado se resuelve usando el oráculo.

Como se señala en la sección II-D, podemos utilizar, por ejemplo, los mercados de predicción para predecir qué actualizaciones del software serán beneficiosas y cuáles serán perjudiciales. También podemos usarlos para estimar cuánto pueden lograr los candidatos en una elección, así que las mentiras y las promesas sin fundamento se pueden detectar más fácilmente.

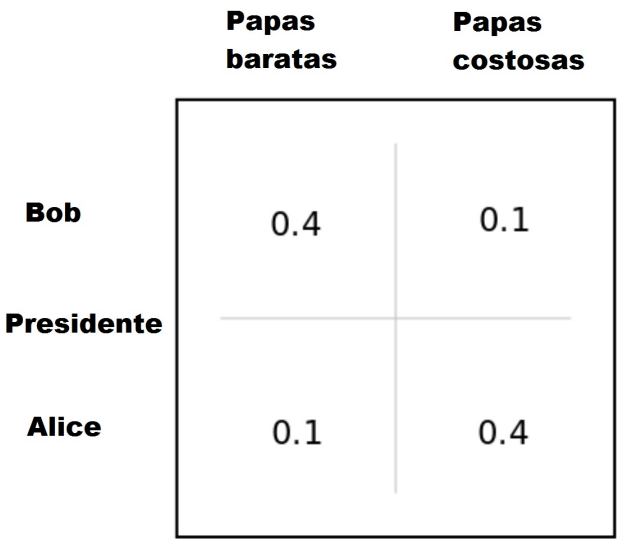

Fig. 5 Mercados de predicción multidimensionales

a) Mercados de predicción multidimensionales: Los mercados de predicción multidimensional nos permiten predecir la correlación entre posibles eventos futuros. Así, por ejemplo, uno podría predecir que si Alice es elegida líder, el precio de las papas bajará, y que si Bob gana, el precio subirá. Uno podría aprender que si Google utiliza el plan A para los próximos 3 meses, que probablemente ganará más dinero, y que si utiliza el plan B, probablemente ganará menos. O, como en la fig. 5, podemos ver que si Alicia sería elegida presidenta, hay una alta probabilidad de que el precio de la papa sea bastante bajo.

B.7) Mercado con batch trading a un precio específico: Hay dos enfoques disponibles para los atacantes que quieren robar aeon de un mercado. Pueden aprovecharse de que el mercado se divide en el tiempo, o pueden aprovechar su división en el espacio.

- Si el mercado está dividido en el espacio, entonces el atacante hace el arbitraje. Simultáneamente hace operaciones en ambos mercados a la vez para que su riesgo se cancele y ganar un beneficio.
- Si el mercado está dividido en el tiempo, entonces lee las transacciones que entran en el mercado y crea órdenes de compra y venta inmediatamente antes y después.

Para combinar los mercados en el espacio, todos deben usar el mismo creador de mercado. Para combinar los mercados en el tiempo, debemos tener el comercio hecho en lotes, a precio único. El creador de mercado tiene que comprometerse con cada persona en el precio decidido y si alguien encuentra compromisos contradictorios del creador de mercado, entonces todos sus clientes deben ser capaces de retirar todos sus canales. Si el creador de mercado se compromete a un precio justo, entonces igualará el mismo volumen de compradores y vendedores juntos, como se muestra en la fig. 6. De lo contrario, terminará en una situación similar a la de la fig. 7, tomando así un gran riesgo.

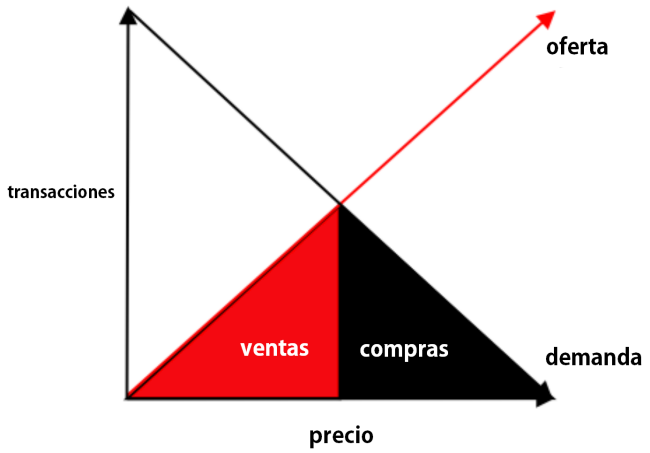

Fig. 6. La línea negra es la curva de demanda, la línea roja es la curva de oferta. Las ventas en rojo son del mismo tamaño que las compras en rojo. La línea vertical es el precio que seleccionó el creador del mercado. Todos los que quieran comprar a un precio superior, negocia a ese precio, todo el que quiera vender a un precio más bajo, negocia a ese precio.

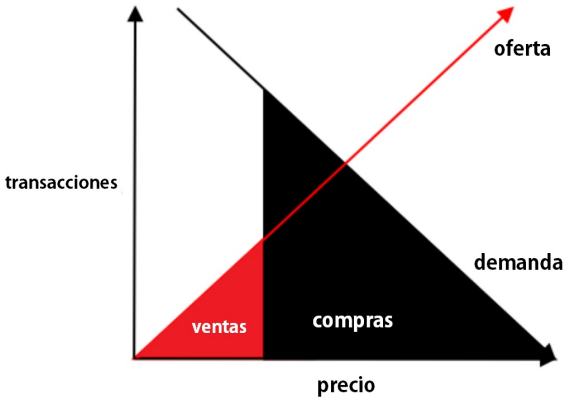

Fig. 7. El negro es mucho más grande que el rojo. El creador de mercado está vendiendo mucho más de lo que está comprando, asumiendo así un gran riesgo.

IV. IMPLEMENTACIÓN

La mayoría de los conceptos clave ya tienen implementaciones de prueba de concepto en Erlang. Esto incluye la cadena de bloques, el lenguaje de los contratos y la máquina virtual, el oráculo y los mecanismos de gobernanza, así como una versión antigua del mecanismo de consenso. Hemos utilizado Erlang / OTP porque facilita la escritura de código que puede responder a muchas solicitudes en paralelo y no se bloquea. Los servidores con el mayor tiempo de actividad en el mundo se basan en Erlang. Se ha utilizado para aplicaciones industriales durante 30 años, demostrando ser un producto confiable y estable.

A. Máquina Virtual y lenguaje de contratos

La máquina virtual está basada en pila y es similar a Forth y al lenguaje de scripting de Bitcoin, aunque en comparación con el último, es bastante rica. La MV soporta funciones en lugar de gotos, haciendo su semántica relativamente simple de analizar.

Una lista de opcodes de la MV se puede encontrar en nuestro Github[3].
Además, existe un lenguaje de nivel superior tipo Forth, llamado Chalang, que compila a código de bytes para la MV. Es compatible con macros y nombres de variables, pero mantiene el modelo de ejecución basada en pila [17]. Ejemplos de código Chalang también se pueden encontrar en nuestro Github[4].

B. Adopción através de integración web

La web es la plataforma de aplicaciones más popular. Nosotros vamos a suministrar herramientas de desarrollo web fáciles de usar Así como librerías-JS y APIs-JSON para las características principales de la cadena de bloques de Æternity.

C. Módulos de código abierto

Con el fin de ser fácilmente reutilizados en cadenas de bloques de consorcios privados y otros casos de uso, el software se escribirá en módulos con licencia MIT, tales como un módulo de consenso, que puede adaptarse a necesidades específicas.

D. Usabilidad y diseño UX

La interacción humana sin fricciones será un gran foco de nuestros esfuerzos de desarrollo. Más específicamente, nos aseguraremos de que quien controla la identidad, claves y transacciones, entienda todo claramente. Además, ofrecer un fácil acceso a través de las pasarelas web (web-gateways), será un foco central de desarrollo futuro. Los usuarios participando en los mercados de predicción a través de una interfaz móvil similar a Tinder (deslizar hacia la izquierda/derecha) y las sencillas billeteras web que se pueden integrar fácilmente en un sitio web a través de un iframe, serán la nueva norma.

V. DISCUSIÓN

Hemos proporcionado una explicación de cómo diseñar un sistema de transferencia de valores fundamentalmente más eficiente. El sistema descrito es de hecho una máquina de oráculo global que puede usarse para proporcionar servicios de toma de decisiones a escala global. En particular, todas las aplicaciones propuestas en la sección III pueden ser construidas fácil y eficientemente sobre Æternidad. Sin embargo, nuestro enfoque tiene tanto limitaciones fundamentales como vías de mejora. Éstos se discuten aquí.

A. Limitaciones y compensaciones

Si bien creemos que las compensaciones hechas en nuestra arquitectura son razonables, dado el aumento del rendimiento resultante en otras áreas, Æternity no es una solución global para aplicaciones descentralizadas. Más bien debería ser visto como un complemento sinérgico a las tecnologías existentes. Hay varias advertencias de las que debemos estar conscientes.

A.1) Estado en cadena: A pesar de tener muchas ventajas, la falta de estado programable de Æternity hace que sea impropio para aplicaciones que requieren un estado personalizado para estar bajo consenso. Por ejemplo, esto incluye DAOs como suelen concebirse, sistemas de nombres personalizados y submonedas que no están vinculadas al valor de un activo subyacente.

[3] https://github.com/aeternity/chalang/blob/master/opcodes.md
[4] https://github.com/aeternity/chalang/tree/master/examples

A.2) El problema de la opción gratis: Si Alice y Bob tienen un canal y Alice firma un contrato, básicamente le da a Bob una opción gratis cuando se lo envía: Bob puede elegir firmar y devolver (es decir, activar) el contrato en cualquier momento en el futuro. A menudo esto no es lo que se pretende. Para evitar este problema, los contratos de canal no se activan inmediatamente con la cantidad total. Están divididos en el tiempo o el espacio. Ambos participantes firmarían para el contrato en intervalos pequeños de modo que ninguno de los usuarios ofrezca nunca una gran opción libre al otro.

Por ejemplo, si las partes quieren apostar 100 aeon, entonces pueden firmarlo hasta en 1000 pasos que aumenten la apuesta en 0,1 aeon cada uno. Esto requeriría del paso de 1000 mensajes, 500 en cada dirección, que es suficientemente barato ya que el contrato nunca se envía a la cadena de bloques. Como otro ejemplo, si quisiera hacer un activo financiero que durara 100 días, uno podría firmarlo en 2400 pasos de una hora cada uno. Esto requeriría del paso de 2400 mensajes, 1200 en cada dirección.

A.3) Pérdida de liquidez y topologías de canales de estado: Cuando se usan hashlocks en la creación de canales de estado, tal como quedó demostrado en la sección II-B.1, cada intermediario debe bloquear al menos el doble de los aeon que se están enviando. Por ejemplo, Si Alice y Carol quieren intercambiar a través de Bob, Bob actuará como Carol cuando interactúe con Alice y viceversa.

Como esto es costoso para Bob, lo más probable es que quiera ganar una comisión como compensación. Si Alice y Carol esperan realizar muchas operaciones entre sí, pueden evitar esto creando un nuevo canal y moviendo los contratos activos sin confianza al nuevo canal usando un hashlock.

Sin embargo, dado que mantener un canal adicional abierto afecta negativamente la liquidez, se espera que el uso de intermediarios sea deseable en muchos casos, especialmente en los que las partes no esperan intercambiar mucho en el futuro. Por lo tanto, se espera que surja una topología de canal en la que ciertos usuarios ricos ganen dinero al transmitir transacciones sin confianza entre otros usuarios.

Cabe señalar que esto no constituye un punto de posibles fallas, ya que no confiamos en estos transmisores de transacciones con nada. Si un transmisor se desconecta antes de que se revele el secreto de un hashlock, la transacción no se realiza. Si se desconecta después, el único efecto "negativo" posible es que el transmisor no puede reclamar sus aeón.

B. Trabajo futuro

Hay varias formas posibles de mejorar la arquitectura actual.

B.1) Lenguaje de contratos funcional: Una dirección razonable para el futuro, es experimentar con lenguajes de alto nivel que se adhieran más al paradigma funcional. Mantener un seguimiento de pila implícita es una práctica generalmente propensa a errores y posiblemente no sea adecuada para un lenguaje de alto nivel, orientado a desarrolladores. Esto debería ser bastante fácil dado que los programas ya son funciones puras (modulo algunas variables de entorno) y simplificaría enormemente tanto el desarrollo como la verificación formal de los contratos. Si se hace esto, también podría tener sentido revisar la MV para que esté estrechamente acoplada con el nuevo idioma y hacer la compilación menos propensa a errores y menos dependiente de la confianza en los desarrolladores. Idealmente, la traducción del lenguaje superficial al código de MV sería simplemente una transcripción directa de la investigación revisada por pares, aunque probablemente habrá que hacer concesiones pragmáticas

B.2) Canales multipartes: Actualmente, todos los canales en Æternity están limitados a dos partes. Mientras que los canales multipartes pueden de facto ser alcanzados a través del hashlocking, esto puede ser costoso. Por lo tanto, planeamos investigar la posibilidad de agregar soporte para canales de *n*-partes, con un mecanismo de acuerdos de *m*-de-*n*.

GLOSARIO

Cadena de Bloques Una base de datos distribuida, a prueba de manipulaciones con acceso medido. La base de datos está definida por una creciente lista de bloques atados a hashes y puede tener reglas para anexarlos.

Aeon Un aeon representa una unidad de cuenta y un derecho de acceso a la cadena de bloques de Æternity. Es transferible.

Transacción Un mensaje de un usuario a la cadena de bloques. Así es como los usuarios pueden usar su moneda para acceder a la cadena de bloques.

Canal de estado Una relación entre dos usuarios registrados en la cadena de bloques. Permite a los usuarios enviar aeones en ambas direcciones y crear contratos inteligentes sin confianza entre ellos, que son aplicados y resueltos por la cadena de bloque.

Hash Un hash toma como entrada un binario de cualquier tamaño. Da una salida de tamaño fijo. La misma entrada siempre genera la misma salida. Dada una salida, no se puede calcular la entrada.

Hashlocking Así es como conectamos las partes de los canales para hacer contratos inteligentes que involucran a más de 2 personas. Un secreto es referenciado por su hash. Cuando se revela el secreto, puede actualizar varios canales al mismo tiempo.

Gobernanza Un proceso bien definido de toma de decisiones para el(los) futuro(s) protocolo(s) de la cadena de bloques.

Oráculos Un mecanismo que le habla a la cadena de bloques sobre los hechos del mundo en el que vivimos. Con el uso de

oráculos, los usuarios pueden predecir el resultado de eventos externos al sistema de la cadena de bloques.

Poseedor de valor Un usuario que tiene aeons o un derivado financiero en el sistema.

Validador Un validador es un usuario que participa en el mecanismo de consenso. En el caso de Æternity, cada poseedor de valor puede participar.

RECONOCIMIENTOS

Gracias a Vlad, Matt, Paul, Dirk, Martin, Alistair, Devon y Ben por sus pruebas de lectura. Gracias a esas y muchas otras personas por conversaciones intuitivas.

REFERENCIAS

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform," 2014. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper

[3] P. Sztorc, "Market empiricism," [Online]. Available: http://bitcoinhivemind.com/papers/1_Purpose.pdf

[4] M. Liston and M. Köppelmann, "A visit to the oracle," 2016. [Online]. Available: https://blog.gnosis.pm

[5] C. Detrio, "Smart markets for smart contracts," 2015. [Online]. Available: http://cdetr.io/smart-markets/

[6] Namecoin wiki, 2016. [Online]. Available: https://wiki.namecoin.org/index.php?title=Welcome

[7] P. Snow, B. Deery, J. Lu, *et al.*, "Factom: Business processes secured by immutable audit trails on the blockchain," 2014. [Online]. Available: http://bravenewcoin.com/assets/Whitepapers/Factom-Whitepaper.pdf

[8] J. Peterson and J. Krug, "Augur: A decentralized, open-source platform for prediction markets," 2014. [Online]. Available: http://bravenewcoin.com/assets/Whitepapers/Augur-A-Decentralized-Open-Source-Platform-for-Prediction-Markets.pdf

[9] A. Swartz, "Squaring the triangle: Secure, decentralized, human-readable names," 2011. [Online]. Available: http://www.aaronsw.com/weblog/squarezooko

[10] T. Hvitved, "A Survey of Formal Languages for Contracts," in Formal Languages and Analysis of Contract-Oriented Software, 2010, pp. 29–32. [Online]. Available: http://www.diku.dk/hjemmesider/ansatte/hvitved/publications/hvitved10flacosb.pdf

[11] R. C. Merkle, "Protocols for public key cryptosystems," in IEEE Symposium on Security and Privacy, 1980.

[12] V. Buterin, "Proof of stake: How I learned to love weak subjectivity," 2014. [Online]. Available: https://blog.ethereum.org/2014/11/25/proof-stake-learned-love-weak-subjectivity/

[13] "Schema.org schemas," 2016. [Online]. Available: http://schema.org/docs/schemas.html

[14] "Atomic-cross-chain-trading," 2016. [Online]. Available: https://en.bitcoin.it/wiki/Atomic%5C_cross-chain%5C_trading

[15] "Interledger," 2016. [Online]. Available: https://interledger.org/

[16] K. J. Arrow, R. Forsythe, M. Gorham, *et al.*, "The promise of prediction markets," Science, 320 2008. [Online]. Available: http://mason.gmu.edu/~rhanson/PromisePredMkt.pdf

[17] Z. Hess, "Chalang," 2016. [Online]. Available: https://github.com/aeternity/chalang