RaiBlocks: Una Criptomoneda Con Red Distribuida Sin Costos

Colin LeMahieu
clemahieu@gmail.com

***Resumen*—Actualmente, la alta demanda y la escalabilidad limitada han aumentado el promedio del tiempo de transacciones y las tarifas en las criptomonedas populares, produciendo una experiencia insatisfactoria. Aquí presentamos a RaiBlocks, una criptomoneda con una arquitectura novedosa llamada Block-lattice (enrejado de bloques) donde cada cuenta tiene su propia blockchain (cadena de bloques), brindando una velocidad de transacción casi instantánea e ilimitada escalabilidad. Al haber una blockchain para cada cuenta, les permite actualizarla de manera asincrónica con el resto de la red, generando transacciones rápidas con una sobrecarga mínima. Las transacciones realizan un seguimiento del balance de las cuentas en lugar de los montos transferidos, lo que permite una reducción agresiva de la base de datos sin comprometer la seguridad. Hasta la fecha, la red de RaiBlocks ha procesado 4.2 millones de transacciones con un ledger (libro de contabilidad digital) de solo 1.7 GB, sin reducciones. Las transacciones en fracción de segundos y sin comisiones hace de RaiBlocks la criptomoneda ideal para las transacciones del consumidor.**

***Palabras Clave*—criptomoneda, blockchain, raiblocks, ledger, transacciones, block-lattice, digital**

I. Introducción

EL comercio en Internet se basa casi exclusivamente en instituciones financieras que sirven como entes confiables para procesar pagos electrónicos. Este sistema funciona lo suficientemente bien para la mayoría de las transacciones; sin embargo, todavía posee debilidades inherentes al modelo basado en la confianza. Las transacciones completamente irreversibles son inviables ya que las instituciones financieras no pueden evitar las disputas mediáticas. El costo de la mediación aumenta los costos de transacción, lo que limita el tamaño mínimo de una transacción práctica y suprime la posibilidad de micro-transacciones. Con la posibilidad de reversión, la necesidad de confianza aumenta y los comerciantes deben tener cautela de sus clientes y recopilar información superflua para minimizar el riesgo monetario, donde un cierto porcentaje de fraudes es aceptado como inevitable. Estas incertidumbres de costos y pagos pueden evitarse utilizando una divisa física, pero actualmente no existe ningún mecanismo para realizar pagos a través de un canal de comunicaciones sin un ente confiable.

La solución a esto es un sistema de pago electrónico, basado en pruebas criptográficas verdaderamente universales, permitiendo a las partes realizar transacciones directamente entre ellas sin la necesidad de un ente de confianza. Las transacciones legítimas que son computacionalmente irreversibles, protegen a los vendedores contra el fraude y se implementa una rutina con mecanismos de garantía para proteger a los compradores. En este documento, presentamos a RaiBlocks; una criptomoneda de baja latencia que tiene una escalabilidad ilimitada y sin costos de transacción.

Las estadísticas de las criptomonedas informadas en este documento son precisas a la fecha de publicación.

II. Antecedentes

En el 2008, un individuo anónimo bajo el seudónimo de Satoshi Nakamoto publicó un documento técnico que describe la primera criptomoneda descentralizada del mundo, el Bitcoin. [1]. Una innovación clave provocada por Bitcoin fue la cadena de bloques (blockchain); una estructura de datos pública, inmutable y descentralizada que se utiliza como un libro de contabilidad (ledger) para las transacciones de la moneda. Desafortunadamente, a medida de que el Bitcoin se desarrolló, varios problemas en el protocolo hicieron que el Bitcoin tuviera restricciones para muchas aplicaciones:

1. Escalabilidad limitada: Cada bloque en el blockchain puede almacenar una cantidad de datos limitada, lo que significa que el sistema solo puede procesar tantas transacciones por segundo, haciendo de los espacios en un bloque un producto básico. Actualmente, la tarifa promedio de una transacción es de $10.38 [2].
2. Alta latencia: El tiempo promedio de confirmación es de 164 minutos [3].
3. Consumo ineficiente: La red del Bitcoin consume un estimado de 27.28TWh por año, usando una media de 260KWh por transacción [4].

Bitcoin y otras criptomonedas funcionan al lograr un consenso en sus registros globales para verificar las transacciones legítimas mientras se resisten a los actores maliciosos. Bitcoin logra el consenso a través de una medida económica llamada prueba de trabajo (PoW en inglés). En un sistema PoW, los participantes compiten para calcular un número, llamado *nonce*, de modo que el hash de todo el bloque se encuentre en el rango del objetivo. Este rango válido es inversamente proporcional al poder de cómputo acumulado de toda la red de Bitcoin, con el fin de mantener un tiempo promedio consistente para encontrar un nonce válido. Al buscador de un nonce válido se le permite agregar el bloque a la blockchain; por lo tanto, aquellos que agotan más recursos computacionales para computar un nonce juegan un papel más importante en el estado de la blockchain. El PoW proporciona resistencia contra un ataque Sybil, donde una

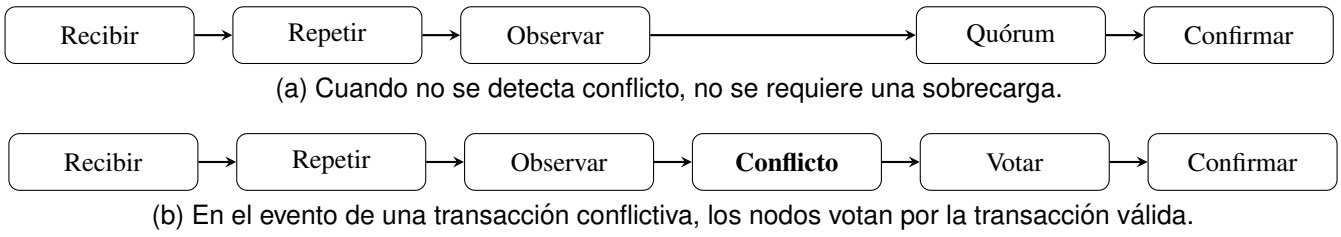

Figura 1. RaiBlocks no requiere una sobrecarga adicional para transacciones típicas. En el caso de una transacción conflictiva, los nodos deben votar por ella para mantenerla

entidad se comporta como entidades múltiples para obtener energía adicional en un sistema descentralizado, y también reduce en gran medida las condiciones de carrera que existen inherentes al acceder a una estructura de datos global.

Un protocolo de consenso alternativo llamado Prueba de Participación (PoS en inglés), fue presentado por primera vez por Peercoin en 2012 [5]. En un sistema PoS, los participantes votan con un peso equivalente a la cantidad de riqueza que poseen en una criptomoneda dada. Con este acuerdo, a los que tienen una mayor inversión financiera se les da más poder y se les incentiva inherentemente a mantener la honestidad del sistema o arriesgarse a perder su inversión. El sitema PoS elimina la competencia de poder de cómputo derrochador, solo requiriendo un software liviano que se ejecuta en hardware de baja potencia.

El documento original de RaiBlocks y la primera implementación beta se publicaron en Diciembre del 2014, lo que la convierte en una de las primeras criptomonedas basadas en gráficos acíclicos dirigidos (DAG en inglés) [6]. Poco después, otras criptomonedas DAG comenzaron a desarrollarse, sobre todo DagCoin/Byteball e IOTA [7], [8]. Estas criptomonedas basadas en DAG rompieron el estándar impuesto por el blockchain, mejorando el rendimiento y la seguridad del sistema. Byteball logra el consenso al basarse en una çadena principalçompuesta por ”testigos”honestos, con buena reputación y de confianza del usuario, mientras que IOTA logra el consenso a través del PoW acumulado de transacciones apiladas. RaiBlocks logra el consenso a través de una votación ponderada y equilibrada sobre transacciones conflictivas. Este sistema de consenso proporciona transacciones más rápidas y más deterministas, manteniendo un sistema fuerte y descentralizado al mismo tiempo. RaiBlocks continúa este desarrollo y se ha posicionado como una de las criptomonedas de mayor rendimiento.

III. Componentes de RaiBlocks

Antes de describir la arquitectura global de RaiBlocks, definimos los componentes individuales que integran al sistema.

III-A. Cuenta

Una cuenta es la porción pública de un par de códigos de firma digital. El código publico, también conocido como la dirección, se comparte con otros participantes de la red mientras que el código privado se mantiene en secreto. Un paquete de datos firmado digitalmente asegura que los contenidos fueron aprobados por el titular de la clave privada. Un usuario puede controlar muchas cuentas, pero solo puede existir una dirección pública por cuenta, por eso lo importante de los códigos tanto públicos como privados.

III-B. Bloque/transacción

El término *bloque* y *transacción* a menudo se usan indistintamente, donde un bloque contiene una sola transacción. La transacción se refiere específicamente a la acción mientras que el bloque se refiere a la codificación digital de la transacción. Las transacciones están firmadas por el código privado que pertenece a la cuenta en la que se realiza la transacción.

III-C. Ledger

El Ledger es el conjunto global de cuentas donde cada cuenta tiene su propia cadena de transacciones (Figura 2). Este es un componente de diseño clave que cae dentro de la categoría de reemplazar un convenio de tiempo de ejecución con un convenio de tiempo de diseño; todos convienen a través de la comprobación de las firmas que solo el propietario de una cuenta puede modificar su propia cadena. Esto convierte una estructura de datos aparentemente compartida, un ledger distribuido, en un conjunto de elementos no compartidos.

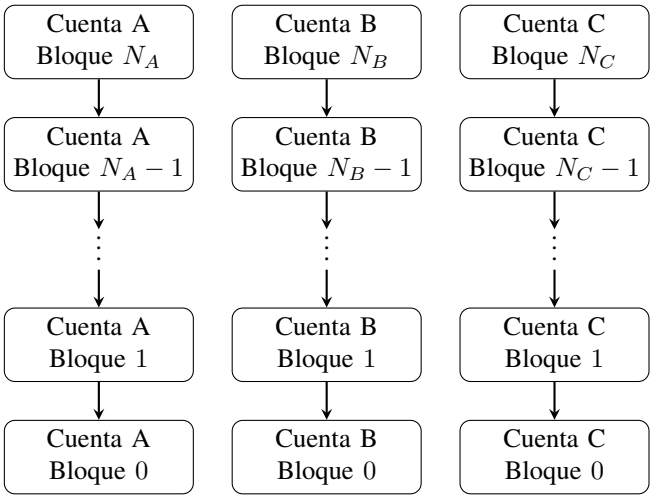

Figura 2. Cada cuenta tiene su propia blockchain, que contiene el balance historico de dicha cuenta. El bloque cero debe ser una transacción abierta (Sección IV-B)

III-D. Nodo

Un *nodo* es una pieza de software que se ejecuta en una computadora, conformando al protocolo de RaiBlocks y participa en la red. El software administra el ledger y cualquier cuenta que el nodo pueda controlar, si corresponde. Un nodo puede almacenar todo el ledger o un historial reducido que contiene solo los últimos bloques del blockchain de cada cuenta. Al configurar un nuevo nodo, se recomienda verificar todo el historial y reducirlo localmente.

IV. Descripción del Sistema

A diferencia de las blockchains de otras criptomonedas, RaiBlocks usa una estructura *block-lattice*. Cada cuenta tiene su propia Blockchain, equivalente al historial de transacciones/saldos de la cuenta (Figura 2). Cada cuenta solo puede ser actualizada por el titular; esto permite que cada blockchain se actualice de forma inmediata y asíncrona al resto del block-lattice, lo que da como resultado transacciones rápidas. El protocolo de RaiBlocks es extremadamente liviano; cada transacción se ajusta al tamaño mínimo del paquete UDP requerido para ser transmitido a través de Internet. Los requisitos de hardware para los nodos también son mínimos, ya que los nodos solo tienen que registrar y retransmitir bloques para la mayoría de las transacciones (Figura 1).

El sistema es iniciado con una *cuenta génesis* que contiene *balance génesis*. El balance de génesis es una cantidad fija y nunca se puede aumentar. El balance de génesis se divide y se envía a otras cuentas a través de transacciones que quedan registradas en el blockchain de génesis. La suma de los saldos de todas las cuentas nunca excederá el saldo inicial de génesis, lo que otorga al sistema un límite superior en cantidad y elimina su capacidad de aumentar.

En esta sección veremos cómo se construyen y propagan los diferentes tipos de transacciones en toda la red.

IV-A. Transacciones

Transferir fondos de una cuenta a otra requiere dos transacciones: un *envío* que deduce el importe del saldo del remitente y un *recibo* que agrega el importe al saldo de la cuenta receptora (Figure 3).

La transferencia de cantidades como transacciones separadas reflejadas en las cuentas del emisor y del receptor, tiene como proposito lo siguiente:

1. Secuenciar transacciones entrantes que son inherentemente asincrónicas.
2. Mantener pequeñas las transacciones, ajustandose al tamaño de un paquede UDP.
3. Facilitar la reducción del ledger, minimizando las huellas de los datos.
4. Aislar las transacciones liquidadas de las no liquidadas.

Más de una cuenta transfiriendo al mismo destino es una operación asincrónica; la latencia de la red y las cuentas de envío no están necesariamente en comunicación mutua, lo que

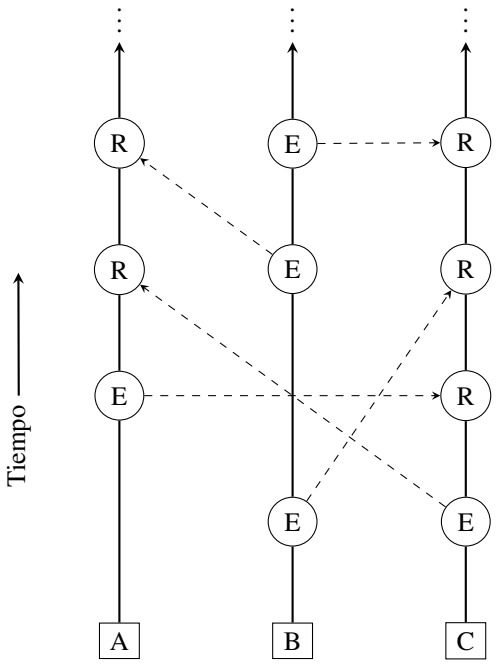

Figura 3. Visualización del block-lattice. Cada transferencia de fondos requiere un bloque de envío (E) y un bloque de recibo (R), cada uno firmado por el propietario de la cadena de cuentas (A, B, C)

significa que no existe una forma universalmente aceptable de saber qué transacción ocurrió primero. Como la suma es asociativa, el orden en que se ordenan las entradas no importa, y por lo tanto, simplemente necesitamos un convenio global. Este es un componente de diseño clave que convierte un convenio de tiempo de ejecución en un convenio de tiempo de diseño. La cuenta receptora tiene el control de decidir qué transferencia llegó primero y se expresa mediante el orden firmado de los bloques entrantes.

Si una cuenta desea realizar una transferencia grande que se recibió como un conjunto de muchas transferencias pequeñas, queremos representar esto de una manera que se ajuste dentro de un paquete UDP. Cuando una cuenta receptora realiza una secuencia de transferencias de entrada, mantiene el total acumulado del saldo de su cuenta, de modo que en cualquier momento tiene la capacidad de transferir cualquier cantidad con un tamaño fijo de transacción. Esto difiere del modelo de transacción de entrada/salida utilizado por Bitcoin y otras criptomonedas.

Algunos nodos no están interesados en gastar recursos para almacenar el historial de transacciones completo de una cuenta; solo están interesados en total de su balance acumulado. Cuando una cuenta realiza una transacción, codifica su balance acumulado y estos nodos solo necesitan realizar un seguimiento del último bloque, lo que les permite descartar datos históricos sin perder la precisión.

Incluso con un enfoque en los acuerdos de tiempo de diseño, hay posibilidad de una demora al validar las transacciones debido a la identificación y el manejo de los actores maliciosos en la red. Como los convenios en RaiBlocks se alcanzan rápidamente, del orden de milisegundos a segundos, podemos presentar al usuario dos categorías

familiares de transacciones entrantes: liquidadas y sin liquidar. Las transacciones liquidadas son transacciones donde una cuenta ha generado bloques de recepción. Las transacciones no liquidadas aún no se han incorporado al saldo acumulativo del receptor. Este es un reemplazo para la métrica de confirmaciones más complejas y desconocidas en otras criptomonedas.

IV-B. Creando una Cuenta

Para crear una cuenta, se debe emitir una transacción *open* (abierta) (Figura 4). Una transacción abierta es siempre la primera transacción de cada blockchain y se puede crear con el primer recibo de fondos. El campo *account* (cuenta) almacena el código público (dirección) derivada del código privado que se utiliza para firmar. El campo *source* contiene el hash de la transacción que envió los fondos. En la creación de la cuenta, se debe elegir un representante para que vote en su nombre; esto se puede cambiar más adelante (Sección IV-F). La cuenta puede declararse como su propio representante.

```
open {
   account: DC04354B1...AE8FA2661B2,
   source: DC1E2B3F7C...182A0E26B4A,
   representative: xrb_1anr...posrs,
   work: 0000000000000000,
   type: open,
   signature: 83B0...006433265C7B204
}
```

Figura 4. Anatomía de una transacción abierta

IV-C. Balance de la Cuenta

El saldo de la cuenta se registra dentro del ledger. En lugar de registrar el monto de una transacción, la verificación (Sección IV-I) requiere verificar la diferencia entre el saldo en el bloque de envío y el saldo del bloque anterior. La cuenta receptora puede entonces incrementar el saldo anterior medido en el saldo final dado en el nuevo bloque de recibo. Esto se hace para mejorar la velocidad de procesamiento al descargar grandes volúmenes de bloques. Al solicitar el historial de la cuenta, los montos ya están dados.

IV-D. Enviando desde una Cuenta

Para enviar desde una dirección, la dirección ya debe tener un bloque abierto existente, y por lo tanto un balance (Figura 5). El campo *previous* (previo) contiene el hash del bloque anterior en la blockchain. El campo *destination* (destino) contiene la cuenta para los fondos que se enviarán. Un bloque de envío es inmutable una vez confirmado. Una vez emitidos a la red, los fondos se deducen inmediatamente del saldo de la cuenta del remitente y esperan como *pending* (pendiente) hasta que la parte receptora firme un bloque para aceptar estos fondos. Los fondos pendientes no deben considerarse como una espera de confirmación, ya que el remitente no puede revocar la transacción.

```
send {
   previous: 1967EA355...F2F3E5BF801,
   balance: 010a8044a0...1d49289d88c,
   destination: xrb_3w...m37goeuufdp,
   work: 0000000000000000,
   type: send,
   signature: 83B0...006433265C7B204
}
```

Figura 5. Anatomía de una transacción de envío

IV-E. Recibiendo una Transacción

Para completar una transacción, el destinatario de los fondos enviados debe crear un bloque de recepción en su propia blockchain (Figura 6). El campo *source* hace referencia al hash de la transacción de envío asociada. Una vez que este bloque se crea y se transmite, el saldo de la cuenta se actualiza y los fondos se transfieren oficialmente a su cuenta.

```
receive {
   previous: DC04354B1...AE8FA2661B2,
   source: DC1E2B3F7C6...182A0E26B4A,
   work: 0000000000000000,
   type: receive,
   signature: 83B0...006433265C7B204
}
```

Figura 6. Anatomía de una transacción de recibo

IV-F. Asignando a un Representante

Los titulares de cuentas que tienen la capacidad de elegir a un representante para que vote en su nombre es una poderosa herramienta de descentralización que no tiene un análogo fuerte en los protocolos PoW o PoS. En los sistemas PoS convencionales, el nodo del propietario de la cuenta debe estar ejecutándose para participar en la votación. La ejecución continua de un nodo no es práctica para muchos usuarios; por lo que dar a un representante el poder de votar en nombre de una cuenta relaja este requisito. Los titulares de cuentas tienen la capacidad de reasignar el consenso a cualquier cuenta en cualquier momento. Una transacción *change* (cambio) cambia el representante de una cuenta restando el peso del voto del antiguo representante y agregando el peso al nuevo representante (Figura 7). No se transfieren fondos en esta transacción, y el representante no tiene el poder de gastar los fondos de la cuenta.

IV-G. Fork y Votos

Se produce un *fork* (bifurcado en inglés) cuando bloques $b_1, b_2, \ldots, b_j$ firmados con j reclaman el mismo bloque que su predecesor (Figura 8), causando un conflicto sobre el estado de una cuenta. Solo el titular tiene la capacidad de firmar bloques en su blockchain, por lo que un fork debe ser producto de una programación deficiente o intención maliciosa (*double-spend* o doble gasto en español) por parte del titular.

```
change {
   previous: DC04354B1...AE8FA2661B2,
   representative: xrb_1anrz...posrs,
   work: 0000000000000000,
   type: change,
   signature: 83B0...006433265C7B204
}
```

Figura 7. Anatomía de una transacción de cambio

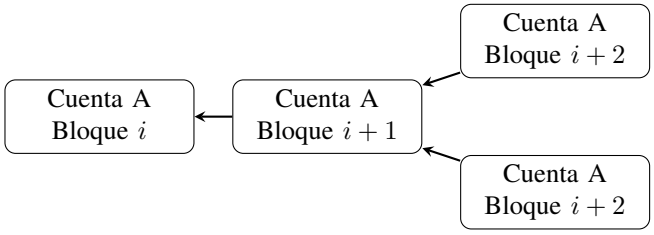

Figura 8. Un fork ocurre cuando uno o mas bloques firmados hacen referencia al mismo bloque anterior en el blockchain de una cuenta. Los bloques antiguos están a la izquierda; los bloques nuevos están a la derecha

Tras la detección de un fork, un representante creará un voto haciendo referencia al bloque $\hat{b}_i$ en su ledger y lo transmitirá a la red. El peso del voto de un nodo w_i es la suma de los saldos de todas las cuentas que lo han nombrado como su representante. El nodo observará los votos entrantes de los otros representantes en línea de M y mantendrá un recuento acumulado durante 4 períodos de votación, un total de 1 minuto, y confirmará el bloque ganador (Ecuación 1).

$$v(b_j) = \sum_{i=1}^{M} w_i \mathbb{1}_{\hat{b}_i = b_j} \tag{1}$$

$$b^* = \arg\max_{b_j} v(b_j) \tag{2}$$

El bloque más popular b^* tendrá la mayoría de los votos y se mantendrá en el ledger del nodo (Ecuación 2). El bloque(s) que pierde el voto se descarta. Si un representante reemplaza un bloque en su ledger, creará un nuevo voto con un número de secuencia más alto y transmitirá el nuevo voto a la red. Este es el **único** escenario donde los representantes votan.

En algunas circunstancias, los problemas de conectividad de red breves pueden causar que un bloque emitido no sea aceptado por todos los *peers* (pares de red). Cualquier bloqueo posterior en esta cuenta será ignorado como inválido por los peers que no vieron la transmisión inicial. Los peers restantes aceptarán una retransmisión de este bloque y los bloques subsiguientes se recuperarán automáticamente. Incluso cuando se produce un fork o un bloque perdido, solo se ven afectadas las cuentas a las que se hace referencia en la transacción; el resto de la red procede con el procesamiento de transacciones para todas las demás cuentas.

IV-H. Prueba de Trabajo (PoW)

Los cuatro tipos de transacciones tienen un campo de trabajo que se deben poblar correctamente. El campo de trabajo permite que el creador de la transacción calcule un nonce de modo que el hash del nonce concatenado con el campo anterior en las transacciones de recibo/envío/cambio o el campo de la cuenta en una transacción abierta esté por debajo de un determinado valor límite. A diferencia del Bitcoin, el PoW en RaiBlocks se usa simplemente como una herramienta antispam, similar a Hashcash, y se puede computar en el orden de segundos [9]. Una vez que se envía una transacción, el PoW para el bloque subsiguiente se puede pre-computar ya que se conoce el campo del bloque anterior; esto hará que las transacciones parezcan instantáneas para un usuario final, siempre que el tiempo entre las transacciones sea mayor que el tiempo requerido para calcular el PoW.

IV-I. Verificación de Transacciones

Para que un bloque sea considerado como válido, debe cumplir con los siguietnes atributos:

1. El bloque no debe estar en el ledger (transacción duplicada).
2. Debe estar firmado por el titular de la cuenta.
3. El bloque anterior es el bloque principal del blockchain de la cuenta. Si este existe pero no es el principal, es un fork.
4. La cuenta debe tener un bloque abierto.
5. El hash computado cumple con el valor límite del PoW requerido.

Si es un bloque de recibo, compruebe si el hash del bloque de origen está pendiente, lo que significa que todavía no se ha canjeado. Si es un bloque de envío, el saldo debe ser menor que el saldo anterior.

V. Vectores de Ataque

RaiBlocks, como todas las criptomonedas descentralizadas, puede ser atacada por grupos malintencionados intentando obtener ganancias financieras o causar el fallecimiento del sistema. En esta sección describimos algunos posibles escenarios de ataque, sus consecuencias y cómo el protocolo de RaiBlock toma medidas preventivas ante estos ataques.

V-A. Sincronización de bloques vacíos

En la sección IV-G, discutimos el escenario en el que un bloque puede no ser emitido correctamente, haciendo que la red ignore los bloques subsiguientes. Si un nodo observa un bloque que no tiene de referencia a un bloque anterior, tiene dos opciones:

1. Ignorar el bloque, ya que puede ser malicioso.
2. Solicitar una resincronización con otro nodo.

En el caso de una resincronización, se debe formar una conexión TCP con un nodo de arranque (*bootsrapping node* para facilitar la mayor cantidad de tráfico que se requiere en una resincronización. Sin embargo, si el bloque era

realmente un bloque malicioso, entonces la resincronización era innecesaria y aumentaba innecesariamente el tráfico en la red. Este es un ataque de amplificación de red y resulta en una denegación de servicio.

Para evitar la resincronización innecesaria, los nodos esperarán hasta que se haya observado un determinado límite de votos para un bloque potencialmente malicioso antes de iniciar una conexión con un nodo de arranque para sincronizar. Si un bloque no recibe suficientes votos, se puede suponer que son datos basura.

V-B. Desbordamiento de Transacciones

Una entidad maliciosa podría enviar muchas transacciones innecesarias pero válidas entre cuentas bajo su control en un intento de saturar la red. Sin tarifas de transacción, pueden continuar este ataque indefinidamente. Sin embargo, el PoW requerido para cada transacción limita la tasa de transacción que la entidad maliciosa podría generar sin invertir significativamente en recursos computacionales. Incluso bajo tal ataque en un intento de inflar el libro mayor, los nodos que no son nodos históricos completos pueden podar las transacciones antiguas de su cadena; esto limita el uso de almacenamiento de este tipo de ataque para casi todos los usuarios.

V-C. Ataque Sybil

Una entidad podría crear cientos de nodos de RaiBlocks en una sola máquina; sin embargo, dado que el sistema de votación se basa en el saldo de la cuenta, agregar nodos adicionales a la red no le otorgará votoz extra al atacante. Por lo tanto, no hay ninguna ventaja que se obtenga a través de un ataque de Sybil.

V-D. Ataque Penny-Spend

Un ataque Penny-Spend (envío de centavos) es cuando un atacante gasta cantidades infinitesimales en una gran cantidad de cuentas para desperdiciar los recursos de almacenamiento de los nodos. La publicación de bloques está limitada por el PoW, por lo que la creación de cuentas y transacciones esta limitada hasta cierto punto. Los nodos que no son nodos históricos por completo pueden reducir las cuentas debajo de una métrica estadística donde la cuenta probablemente no sea una cuenta válida. Finalmente, RaiBlocks está sintonizado para usar un espacio mínimo de almacenamiento permanente, por lo que el espacio requerido para almacenar una cuenta adicional es proporcional al tamaño de open block + indexing = 96B + 32B = 128B. Esto equivale a que 1GB puede almacenar 8 millones de cuentas Penny-Spend. Si los nodos desean reducir de forma más agresiva, pueden calcular una distribución en función de la frecuencia de acceso y delegar cuentas de uso poco frecuente a un almacenamiento más lento.

V-E. Ataque de PoW Pre-Computado

Dado que el propietario de una cuenta será la única entidad que agregue bloques a su blockchain, se pueden computar bloques secuenciales, junto con su PoW, antes de ser transmitidos a la red. Aquí el atacante genera una gran cantidad de bloques secuenciales, cada uno de valor mínimo, durante un período prolongado de tiempo. En cierto punto, el atacante realiza una denegación de servicio (DoS en inglés) inundando la red con muchas transacciones válidas, que otros nodos procesarán y harán eco lo más rápido posible. Esta es una versión avanzada del desbordamiento de transacciones descrito en la Sección V-B. Tal ataque solo funcionaría brevemente, pero podría usarse junto con otros ataques, como un ataque >50 % (Sección V-F) para aumentar la efectividad. La limitación de la tasa de transacción y otras técnicas se están investigando actualmente para mitigar los ataques.

V-F. Ataque >50 %

La métrica del consenso para RaiBlocks es un sistema de votación equilibrado. Si un atacante puede ganar más del 50 % de la fuerza de votación, puede hacer que el consenso fluctúe dentro de la red, haciendo que el sistema se rompa. Un atacante puede reducir la cantidad del balance que debe perder al impedir que los nodos buenos voten, mediante un DoS en la red. RaiBlocks toma las siguientes medidas para prevenir tal ataque:

1. La defensa primaria contra este tipo de ataque es que el peso del voto está ligado a la inversión en el sistema. El titular de una cuenta está intrínsecamente incentivado a mantener la honestidad del sistema para proteger su inversión. Intentar voltear el ledger sería destructivo para el sistema como un todo, lo que destruiría su inversión en totalidad.
2. El costo de este ataque es proporcional a la capitalización de mercado de RaiBlocks. En los sistemas PoW, se puede inventar tecnología que proporcione un control desproporcionado en comparación con la inversión monetaria y, si el ataque es exitoso, esta tecnología podría reutilizarse después de que se complete el ataque. Con RaiBlocks, el costo de atacar el sistema escala con el sistema mismo y, si un ataque tuviera éxito, la inversión hecha en el ataque no se puede recuperar.
3. Para mantener el cuórum máximo de votantes, la siguiente línea de defensa es la votación de representantes. Los titulares de cuentas que no pueden participar de manera confiable en la votación por razones de conectividad, pueden nombrar a un representante que pueda votar con el peso de su saldo. Maximizar el número y la diversidad de representantes aumenta la resiliencia de la red.
4. Un fork en RaiBlocks nunca es accidental, por lo que los nodos pueden tomar decisiones sobre cómo interactuar con bloques bifurcados. La única forma de que las cuentas no atacantes sean vulnerables a las bifurcaciones de bloques es si reciben un saldo de una cuenta atacante. Las cuentas que quieran estar seguras de un fork de bloques, pueden esperar un poco o mucho más antes de recibirlas de una cuenta que generó el fork u optar por no

recibirlas nunca. Los receptores también pueden generar cuentas separadas para usar cuando reciben fondos de cuentas dudosas, con el fin de aislar otras cuentas.

5. Una última línea de defensa que aún no se ha implementado es el *block cementing* (cementación de bloques). RaiBlocks hace todo lo posible para resolver un fork rápidamente a través de la votación. Los nodos se pueden configurar para cementar bloques, lo que evitaría retrocedan después de un cierto período de tiempo. La red está suficientemente asegurada al enfocarse en un tiempo de resolución rápido para evitar bifurcaciones ambiguas.

Una versión más sofisticada de un ataque $> 50\,\%$ se detalla en la figura 9. "Offline" es el porcentaje de representantes que han sido nombrados pero que no están en línea para votar. "Stake" es la cantidad de inversión con la que vota el atacante. "Active" es la cantidad de representantes que están en línea y votan de acuerdo con el protocolo. Un atacante puede compensar la cantidad de participación que debe perder tornando a otros votantes fuera de línea a través de un ataque DoS en la red. Si este ataque puede ser sostenido, los representantes atacados pierden la sincronización y esto se demuestra con "Unsync". Finalmente, un atacante puede obtener una pequeña ráfaga de fuerza relativa de votación al cambiar su ataque DoS a un nuevo conjunto de representantes, mientras que el conjunto anterior vuelve a sincronizar su ledger, esto se demuestra con "Attack".

Offline	Unsync	**Attack**	Active	Stake

Figura 9. Un potencial convenio de votación que podría reducir los requisitos para un ataque de 51 %.

Si un atacante puede causar que Stake>Active por una combinación de estas circunstancias, podrá invertir los votos en el ledger a expensas de su Stake. Podemos estimar cuánto podría costar este tipo de ataque examinando el capital de mercado de otros sistemas. Si estimamos que el 33 % de los representantes están fuera de línea o atacados a través de un DoS, un atacante tendría que comprar el 33 % de la capitalización de mercado para atacar el sistema mediante votación.

V-G. Envenenamiento de Arranque

Mientras más tiempo un atacante pueda mantener un código privado antiguo balance, mayor será la probabilidad de que los saldos que existían en ese momento no tengan representantes participando porque sus balances o representantes se hayan transferido a cuentas más nuevas. Esto significa que si un nodo se inicia en una representación antigua de la red donde el atacante tiene un quórum de participación de votación en comparación con los representantes en ese punto en el tiempo, podría oscilar las decisiones de votación a ese nodo. Si este nuevo usuario deseara interactuar con cualquier persona además del nodo atacante, todas sus transacciones serían denegadas ya que tienen diferentes bloques principales. El resultado es que los nodos pueden desperdiciar el tiempo de los nuevos nodos en la red al proporcionarles información incorrecta. Para evitar esto, los nodos se pueden emparejar con una base de datos inicial de cuentas y bloques principales conocidos; este es un reemplazo para la descargar de la base de datos hasta el bloque de génesis. Cuanto más cerca esté la descarga de ser la actual, mayor será la probabilidad de defenderse con precisión contra este ataque. Al final, este ataque probablemente no sea peor que alimentar de datos no deseados a los nodos durante el arranque, ya que no podrían realizar transacciones con nadie que tenga una base de datos contemporánea.

VI. Implementación

Actualmente, la implementación de referencia esta en lenguaje C ++ y ha estado produciendo lanzamientos desde 2014 en Github. [10]. A continuación presentamos como esta implementado RaiBlocks.

VI-A. Características del Diseño

La implementación de RaiBlocks cumple con el estándar de arquitectura descrito en este documento. Las especificaciones adicionales se describen a continuación.

VI-A1. Algoritmo de Firmas: RaiBlocks usa un algoritmo de curva elíptica ED25519 modificado con Blake2b hashing para todas las firmas digitales [11]. El ED25519 fue elegido para garantizar firmas rápidas, verificación rápida y alta seguridad.

VI-A2. Algoritmo de Hashing: Dado que el algoritmo de hash solo se usa para evitar el spam de la red, la elección del algoritmo es menos importante en comparación con las criptomonedas basadas en la minería. Nuestra implementación utiliza Blake2b como un algoritmo de asimilación contra el contenido del bloque [12].

VI-A3. Función de Derivación de Códigos: En la cartera de referencia, los códigos estan encriptados con una contraseña y la contraseña se alimenta a través de una función de derivación de códigos para protegerse contra los intentos de craqueo ASIC. Actualmente Argon2 [13] es el ganador de la única competencia pública destinada a crear una función de derivación de códigos resistente.

VI-A4. Intervalos de Bloques: Como cada cuenta tiene su propia blockchain, las actualizaciones se pueden realizar de forma asíncrona al estado de la red. Por lo tanto, no hay intervalos de bloques y las transacciones se pueden publicar al instante.

VI-A5. Protocolo de Mensajes UDP: Nuestro sistema está diseñado para operar de manera indefinida utilizando la cantidad mínima de recursos informáticos como sea posible. Todos los mensajes en el sistema fueron diseñados encajar dentro de un único paquete UDP. Esto también facilita que los peers ligeros con conectividad intermitente participen en la red sin restablecer las conexiones TCP a corto plazo. TCP se usa solo para los nuevos peers cuando quieren arrancar las cadenas de bloques de forma masiva.

Los nodos pueden estar seguros de que su transacción fue recibida por la red al observar el tráfico de transmisión de transacciones desde otros nodos, ya que debería ver varias copias repetidas hacia el.

VI-B. IPv6 y Multicast

Construir encima del UDP sin conexión permite a las implementaciones futuras usar la multidifusión IPv6 como un reemplazo para los desbordamientos de transacciones y la transmisión de votos. Esto reducirá el consumo de ancho de banda de la red y dará más flexibilidad política a los nodos en el futuro.

VI-C. Desempeño

En el momento de escribir estas líneas, la red RaiBlocks ha procesado 4.2 millones de transacciones, lo que deja con un tamaño de blockchain de 1,7 GB. Los tiempos de transacción se miden en el orden de segundos. Una implementación de referencia actual que opera en SSD de productos básicos puede procesar más de 10 000 transacciones por segundo, principalmente en IO.

VII. Uso de recursos

Esta es una descripción general de los recursos utilizados por un nodo de RaiBlocks. Además, repasamos ideas para reducir el uso de recursos para casos específicos de aplicación. Los nodos reducidos se suelen denominar nodos de verificación de pago simplificado, ligero o reducido (SPV en inglés).

VII-A. Red

La cantidad de actividad de la red depende de cuánto contribuye la red para la salud de una red.

VII-A1. Representativo: Un nodo representativo requiere un máximo de recursos de red ya que observa el tráfico de votos de otros representantes y publica sus propios votos.

VII-A2. Inconfiable: Un nodo inconfiable es similar a un nodo representativo pero solo es un observador, no contiene una cuenta representativa y no publica sus propios votos.

VII-A3. Confiado: Un nodo confiado observa el tráfico de votos de un representante en el que confía para realizar correctamente el consenso. Esto reduce la cantidad de tráfico de votos entrantes de los representantes que van a este nodo.

VII-A4. Ligero: Un nodo ligero también es un nodo confiado que solo observa el tráfico de las cuentas en las que está interesado, lo que permite un uso mínimo de la red.

VII-A5. De Arranque: Un nodo de arranque sirve partes o el todo del ledger para nodos que se ponen o intentan ponerse en línea. Esto se realiza a través de una conexión TCP en lugar de una conexión UDP, ya que implica una gran cantidad de datos que requieren un control de flujo avanzado.

VII-B. Capacidad de Disco

Dependiendo de las demandas del usuario, diferentes configuraciones de nodos necesitan diferentes requisitos de almacenamiento.

VII-B1. Histórico: Un nodo interesado en mantener un registro histórico completo de todas las transacciones requerirá la máxima cantidad de almacenamiento.

VII-B2. Actual: Debido al diseño orientado a mantener los balances acumulados con bloques, los nodos solo necesitan mantener los bloques más recientes o principales para cada cuenta a fin de participar en el consenso. Si un nodo no está interesado en mantener un historial completo, puede optar por mantener solo los bloques principales.

VII-B3. Ligero: Un nodo ligero no conserva datos del ledger local y solo participa en la red para observar la actividad en las cuentas en las que está interesado u opcionalmente crear nuevas transacciones con claves privadas que posee.

VII-C. CPU

VII-C1. Generación de Transacciones: Un nodo interesado en crear nuevas transacciones debe producir un nonce de PoW para pasar el mecanismo de aceleración de RaiBlocks. El cómputo de varios hardwares se compara en el Apéndice A.

VII-C2. Representantes: Un representante debe verificar firmas para bloques, votos y también producir sus propias firmas para participar en el consenso. La cantidad de recursos de CPU para un nodo representativo es significativamente menor que la generación de transacciones y debería funcionar con cualquier CPU individual en una computadora contemporánea.

VII-C3. Observador: Un nodo observador no genera sus propios votos. Como la sobrecarga por la generación de firmas es mínima, los requisitos de la CPU son casi idénticos a los de ejecutar un nodo representativo.

VIII. Conclusión

En este documento presentamos el sistema de una criptomoneda de baja latencia, veloz y sin costos, que utiliza una nueva estructura llamada block-lattice y una convenio de votación delegado por PoS. La red requiere recursos mínimos, sin hardware de minería de alta potencia y que puede procesar un alto rendimiento de transacciones. Todo esto se logra teniendo cadenas de bloques individuales para cada cuenta, eliminando los problemas de acceso e ineficiencias de una estructura de datos global. Identificamos los posibles vectores de ataque en el sistema y presentamos argumentos sobre cómo RaiBlocks es resistente a estas formas de ataque.

APÉNDICE A
BENCHMARK DEL HARDWARE PARA EL POW

Como se mencionó anteriormente, el PoW en RaiBlocks se utiliza para reducir el spam dentro la red. Nuestra implementación de nodos proporciona una aceleración que puede aprovechar las GPUs compatibles con OpenCL. La tabla I proporciona una comparación real de varios tipos de hardware. Actualmente, el límite del PoW es fijo, pero se puede implementar un límite que se adapte a medida que progresa la potencia de computo promedio.

Cuadro I
DESEMPEÑO DE POW DEL HARDWARE

Dispositivo	Transacciones por segundo
Nvidia Tesla V100 (AWS)	6.4
Nvidia Tesla P100 (Google,Cloud)	4.9
Nvidia Tesla K80 (Google,Cloud)	1.64
AMD RX 470 OC	1.59
Nvidia GTX 1060 3GB	1.25
Intel Core i7 4790K AVX2	0.33
Intel Core i7 4790K,WebAssembly (Firefox)	0.14
Google Cloud 4 vCores	0.14-0.16
ARM64 server 4 cores (Scaleway)	0.05-0.07

RECONOCIMIENTO

Nos gustaría agradecer a Brian Pugh por compilar y formatear este documento

REFERENCIAS

[1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008. [Online]. Available: http://bitcoin.org/bitcoin.pdf
[2] “Bitcoin median transaction fee historical chart.” [Online]. Available: https://bitinfocharts.com/comparison/bitcoin-median_transaction_fee.html
[3] “Bitcoin average confirmation time.” [Online]. Available: https://blockchain.info/charts/avg-confirmation-time
[4] “Bitcoin energy consumption index.” [Online]. Available: https://digiconomist.net/bitcoin-energy-consumption
[5] S. King and S. Nadal, “Ppcoin: Peer-to-peer crypto-currency with proof-of-stake,” 2012. [Online]. Available: https://peercoin.net/assets/paper/peercoin-paper.pdf
[6] C. LeMahieu, “Raiblocks distributed ledger network,” 2014.
[7] Y. Ribero and D. Raissar, “Dagcoin whitepaper,” 2015.
[8] S. Popov, “The tangle,” 2016.
[9] A. Back, “Hashcash - a denial of service counter-measure,” 2002. [Online]. Available: http://www.hashcash.org/papers/hashcash.pdf
[10] C. LeMahieu, “Raiblocks,” 2014. [Online]. Available: https://github.com/clemahieu/raiblocks
[11] D. J. Bernstein, N. Duif, T. Lange, P. Shwabe, and B.-Y. Yang, “High-speed high-security signatures,” 2011. [Online]. Available: http://ed25519.cr.yp.to/ed25519-20110926.pdf
[12] J.-P. Aumasson, S. Neves, Z. Wilcox-O’Hearn, and C. Winnerlein, “Blake2: Simpler, smaller, fast as md5,” 2012. [Online]. Available: https://blake2.net/blake2.pdf
[13] A. Biryukov, D. Dinu, and D. Khovratovich, “Argon2: The memory-hard function for password hashing and other applications,” 2015. [Online]. Available: https://password-hashing.net/argon2-specs.pdf