Proyecto de Grado

Presentado ante la ilustre Universidad de Los Andes

Integración de grafos acíclicos dirigidos para mejorar la escalabilidad en la cadena de bloques

www.bdigital.ula.ve

Por

Br. Ana Elizabeth Guerrero López

Tutor: Ph.D José Luis Paredes

Febrero, 2020

©2020 Universidad de Los Andes Mérida, Venezuela

C.C. Reconocimiento

Integración de grafos acíclicos dirigidos para mejorar la escalabilidad en la cadena de bloques

Br. Ana Elizabeth Guerrero López

Proyecto de Grado — Sistemas Computacionales, 65 páginas

Resumen: Desde el surgimiento de bitcoin en 2008, la tecnología de cadena de bloques ha tenido gran auge en los últimos años; con ella se pueden realizar transacciones sin necesidad de terceros lo que ha causado un gran impacto en las divisas digitales. Sin embargo, ha presentado problemas por los altos tiempos de procesamiento y validación de las transacciones. Innumerables desarrollos se han enfocado en mejorar dichos tiempos de validación, es por esto que en este trabajo se propone la inclusión de grafos acíclicos dirigidos como estructura de datos dentro de la cadena de bloques, enfocándose en mejorar la escalabilidad de los sistemas. El desarrollo de la red tiene como algoritmo de consenso el algoritmo de transacciones como prueba de participación con el cual se obtienen métricas de rendimiento satisfactorias similares a las de las criptomonedas más importantes. Así mismo, la red es implementada en un caso de uso demostrando que el campo de aplicación de la cadena de bloques es más amplia que el mundo de las divisas digitales.

www.bdigital.ula.ve

Palabras clave: cadena de bloques, grafo acíclico dirigido, DAG, blockdag, escalabilidad.

C.C. Reconocimiento

Índice general

www.bdigital.ula.ve

C.C. Reconocimiento

www.bdigital.ula.ve

C.C. Reconocimiento

www.bdigital.ula.ve

C.C. Reconocimiento

Índice de tablas

www.bdigital.ula.ve

C.C. Reconocimiento

Índice de figuras

www.bdigital.ula.ve
C.C. Reconocimiento

www.bdigital.ula.ve

C.C. Reconocimiento

Índice de algoritmos

www.bdigital.ula.ve

C.C. Reconocimiento

Capítulo 1

Introducción

Uno de los nuevos avances tecnológicos que ha tenido gran auge recientemente es la cadena de bloques. Esta nueva tecnología comenzó con el lanzamiento de la criptomoneda *bitcoin* (Nakamoto, 2008). Hoy en día han nacido nuevas vertientes para migrar la cadena de bloques a nuevos enfoques y ayudar a resolver diferentes problemas como en el Internet de las cosas, *big data*, así como todo aquel sistema que contenga grandes cantidades de datos y se quieran manejar de forma descentralizada.

www.bdigital.ula.ve

La seguridad que aporta la cadena de bloques ha sido muy bien aceptada y acertada en esta era tecnológica donde los ataques a los sistemas están a la orden del día. Es por esto, que se quiere brindar esta seguridad en nuevos campos de aplicación que hoy en día son vulnerables. Se han tomado varios enfoques para resolver los problemas de escalabilidad y adaptación de la cadena de bloques a estos sistemas, uno de ellos es la inclusión de grafos acíclicos dirigidos como base de su estructura (Choi y col., 2018a).

Los grafos acíclicos dirigidos aportan la escalabilidad y rapidez que es necesaria en la cadena de bloques, es por ello que es uno de los enfoques más acertados para resolver esta problemática.

La presente investigación pretende ser una incursión académica en el área de la cadena de bloques y grafos acíclicos dirigidos, enfocándose principalmente en el desarrollo de una red que ayude a resolver la problemática expuesta aprovechando las virtudes de cada tecnología.

C.C. Reconocimiento

1.1. Antecedentes

En este trabajo se tiene como antecedentes todos aquellos trabajos en el ámbito de la cadena de bloques donde se han implementado grafos acíclicos dirigidos como estructura de datos. Se concentrará la atención en aquellos trabajos donde la contribución sea la mas relacionada con los objetivos a cumplir.

En el trabajo de Popov (2018), se tiene como estructura de datos un grafo acíclico dirigido que llaman "el enredo" en el cual son guardadas las transacciones. La red esta compuesta por nodos que son entidades que emiten y validan transacciones. Para emitir una transacción, los usuarios deben trabajar para aprobar otras transacciones; por lo tanto, los usuarios que emiten una transacción contribuyen a la seguridad de la red. En dicho trabajo el objetivo es obtener un sistema de libro mayor distribuido sin el uso de la cadena de bloques para obtener características como cero comisiones, escalabilidad, transacciones rápidas y transferencias de datos seguras, enfocado principalmente en ser aplicado en el Internet de las cosas. La solución propuesta a este problema viene siendo el protocolo "*Tangle*" o "el enredo" basado en la estructura de datos de grafos acíclicos dirigidos y el procesamiento de transacciones en paralelo. Este desarrollo se encuentra implementado en la criptomoneda IOTA.

www.bdigital.ula.ve

Por otro lado, Churyumov (2016) propone una criptomoneda llamada *Byteball* que se encuentra diseñada sobre un grafo acíclico dirigido, por lo tanto sustituyen la cadena de bloques por esta estructura de datos. Cada transacción pertenece a un nodo de la red, no hacen uso de la prueba de trabajo que es el algoritmo más común en las criptomonedas más antiguas. La principal diferencia con Popov (2018), es la manera de confirmar transacciones ya que en este caso si es necesario pagar un monto equivalente al tamaño de la transacción, además de esto el nodo encargado de validar puede confirmar un grupo de transacciones acumuladas. La propuesta de este trabajo es realizar un sistema para el almacenamiento descentralizado e inmutable de datos arbitrarios como el dinero.

Del mismo modo, el trabajo realizado por Choi y col. (2018a) es importante en esta área ya que hace uso de grafos acíclicos dirigidos sin eliminar la cadena de bloques, manteniendo la estructura de la cadena principal con un protocolo al que denominan

C.C. Reconocimiento

"Lachesis". Este protocolo hace uso de la asíncronidad para sacar el mejor provecho de la red donde cada nodo puede crear libremente nuevos eventos al mismo tiempo. Se encuentra implementado en la criptomoneda denominada *Fantom*. El objetivo de *Fantom* es lograr un alto rendimiento y un almacenamiento de datos seguro haciendo uso de grafos acíclicos dirigidos y la cadena de bloques. La propuesta realizada es un libro de contabilidad distribuido basado en grafos acíclicos dirigidos, un sistema escalable y adaptable a diversos escenarios como ciudades inteligentes, manejo de tráfico y educación.

De los trabajos anteriormente expuestos se tomaron ideas para resolver el problema en cuestión. Cada uno de ellos tienen enfoques diferentes para resolver el problema de velocidad y escalabilidad de la cadena de bloques. En el caso de Popov (2018) se obtiene una red sin cadena de bloques, enfocada completamente en el uso del grafo acíclico dirigido donde se obtienen resultados muy satisfactorios en la criptomoneda implementada. Por su parte, Churyumov (2016) además de eliminar la cadena de bloques, elimina el concepto de bloques de la red. Por otro lado, el trabajo propuesto por Choi y col. (2018a) es implementado bajo el concepto de unificación de cadena de bloques con grafos acíclicos dirigidos, manteniendo las bondades de cada uno. Cada uno de ellos realizan un manejo diferente de los datos dentro del DAG, de los cuales se tomaron ideas principalmente del trabajo expuesto por Choi y col. (2018a) y Churyumov (2016).

www.bdigital.ula.ve

1.2. Planteamiento del problema

Desde el comienzo de la primera criptomoneda propuesta por Nakamoto (2008) hasta las nuevas criptomonedas propuestas por *Ethereum* (Buterin, 2013) e *Hyperledger* (Blummer, 2018), se ha dejado en evidencia su eficacia con respecto a transacciones. Sin embargo, cuando se quiere usar su tecnología base de cadena de bloques para el manejo de datos se comienza a ver fallos principalmente en su capacidad de volverse exponencial. Con la cadena de bloques cualquier nodo puede construir bloques y propagarlos por la red, todos y cada uno de los nodos tienen esa información del nuevo bloque y, además la información de todos los bloques anteriores. Pero qué pasa

C.C. Reconocimiento

si ese bloque contiene imágenes o un gran tamaño de datos o peor aún todos los nodos comienzan a enviar bloques de gran tamaño, la red podría convertirse en una cadena de bloques fuera de control.

Aunado a este problema se encuentra que para que un bloque sea válido y aceptado por los nodos de la red debe cumplir una prueba de trabajo que por naturaleza ofrece la cadena de bloques. Los nodos mineros deben competir entre ellos para poder incluir un nuevo bloque en la cadena y ganar una recompensa, pero en muchos casos es necesario de un computador muy potente para poder resolver los problemas computacionales y obtener la recompensa. Se han realizado muchos esfuerzos para disminuir el tiempo de inserción de un nuevo bloque a la cadena pero los mismos han tenido dificultades ya que se incurre en pérdidas de seguridad. De igual forma, la estructura de la cadena de bloques es limitada y no permite la escalabilidad de los sistemas. Cuando se habla de escalabilidad, en *blockchain* se hace referencia a la cantidad de transacciones que una red puede procesar.

Uno de estos enfoques ha sido migrar la cadena de bloques a la estructura de grafos acíclicos dirigidos, al hacer esto se gana rapidez en la inclusión de nuevos bloques pero se ve afectada la seguridad ya que se elimina o se vuelve casi nula la prueba de trabajo, por ende se han propuestos nuevos algoritmos de consenso como es el caso de criptomonedas como *Byteball* o IOTA.

www.bdigital.ula.ve

1.3. Objetivos

1.3.1. General

Mejorar la escalabilidad de sistemas basados en cadena de bloques con la inclusión de grafos acíclicos dirigidos a su estructura.

1.3.2. Específicos

- Realizar una revisión bibliográfica sobre grafos acíclicos dirigidos y cadena de bloques.

C.C. Reconocimiento

- Definir un protocolo de consenso que se ajuste a la estructura de grafos acíclicos dirigidos.
- Implementar grafos acíclicos dirigidos como estructura de datos en la cadena de bloques.
- Ejecutar pruebas de rendimiento y comparación entre grafos acíclicos dirigidos y la cadena de bloques tradicional mediante las métricas encontradas en la revisión bibliográfica.
- Desarrollar un ambiente de prueba donde se aplique el sistema en un caso de uso.

1.4. Justificación

Con la nueva era tecnológica cada vez es más imprescindible tener seguridad en los datos que se manejan desde pequeñas empresas como grandes conglomerados empresariales. La cadena de bloques o *blockchain* (término en inglés) ha prometido ser la solución a los problemas de seguridad necesarios debido a que cada nuevo bloque añadido a la cadena es inmutable y no puede ser modificado. Este proceso es caro y lento en comparación a la cantidad de transacciones que el mundo necesita realizar cada segundo, aunque es justificable si el fin que se quiere lograr es eliminar a los usuarios maliciosos en la red.

www.bdigital.ula.ve

En este sentido, los grafos acíclicos dirigidos aportan rapidez debido a que su estructura permite realizar inserciones con una dependencia clara y ya no es necesario recorrer toda la cadena para revisar un orden cronológico de la transacción. Así mismo, si el caso es la resolución de un problema se puede insertar un orden de pasos a realizar para resolver dicho problema. Esta virtud se puede traducir en un recorrido computacionalmente menos costoso que el recorrido que se requeriría hacer en la cadena de bloques para realizar la misma acción. De igual manera, permiten que los sistemas puedan adaptarse a otros campos de aplicación y no solo limitarse a transacciones. Estas virtudes permiten ganar rapidez pero se pierde seguridad ya que el mecanismo no aporta la inmutabilidad de los datos.

C.C. Reconocimiento

Al evaluar la eficacia de la cadena de bloques se puede observar que es seguro, inmutable y transparente, lo cual aporta grandes cambios a los procesos de hoy en día. Por otro lado, los grafos acíclicos dirigidos aportan escalabilidad y rapidez a las transacciones de datos. Al unir estas virtudes se saca gran provecho ya que se puede obtener un potente servicio que eliminaría las limitaciones de ambas tecnologías. Por esta razón se pretende aprovechar la estructura del grafo para mejorar la escalabilidad de sistemas basados en cadena de bloques.

1.5. Alcance

Diseñar e implementar una red que incluya grafos acíclicos dirigidos como estructura de datos en una cadena de bloques, aprovechando las virtudes de tener un sistema descentralizado y una estructura de datos que permite realizar una validación de bloques más rápida sin perder seguridad. Así como la inclusión y escalabilidad en otros casos de estudio. En este sentido, se pretende realizar un desarrollo de una red que cumpla ese objetivo.

www.bdigital.ula.ve

1.6. Metodología

A continuación se muestran los procedimientos más relevantes necesarios para alcanzar los objetivos planteados.

- En primer lugar, se realizará una revisión bibliográfica exhaustiva sobre grafos acíclicos dirigidos y cadena de bloques para obtener los conocimientos e ideas necesarias para cumplir los objetivos planteados.
- Para definir el protocolo de consenso se estudiarán los protocolos desarrollados hasta el momento para grafos acíclicos dirigidos y se escogerá aquel que se adapte mejor.
- Una vez seleccionado el protocolo de consenso se procederá a la implementación de grafos acíclicos dirigidos como estructura de datos en la cadena de bloques, siguiendo las fases de la metodología de desarrollo incremental.

C.C. Reconocimiento

- Por último se realizará una serie de pruebas funcionales y de rendimiento para comprobar la tasa de éxito o fracaso de la implementación, así como el desarrollo de un ambiente de prueba donde sea aplicado el sistema en un caso de uso.

1.7. Estructura del documento

El documento se organiza de la siguiente manera: el capítulo 2 presenta las bases teóricas del presente trabajo. Se describen los conceptos básicos de cadena de bloques así como los campos de aplicación donde ha sido usado. De igual forma se realiza un resumen de los protocolos de consenso más relevantes en el campo y las métricas de rendimiento importantes para la aplicación de la cadena de bloques. Adicionalmente, se describe la teoría de grafos y los conceptos de grafos acíclicos dirigidos, los cuales son conceptos claves para la investigación.

En el capítulo 3 se presenta el diseño y desarrollo de la red de cadena de bloques con la integración de grafos acíclicos dirigidos.

www.bdigital.ula.ve

El capítulo 4 se desarrolla la integración de los módulos desarrollados en el Capítulo 3 a un caso de uso para verificar su funcionamiento.

Finalmente, en el capítulo 5 se presentan las conclusiones y recomendaciones del trabajo realizado.

C.C. Reconocimiento

Capítulo 2

Marco Teórico

El siguiente capítulo presenta una breve descripción de los conceptos claves necesarios para comprender las bases teóricas de la problemática que se quiere resolver.

2.1. Cadena de bloques

www.bdigital.ula.ve

De acuerdo con Rennock y col. (2018) una cadena de bloques o *blockchain* es un libro de contabilidad digital de transacciones entre pares que puede distribuirse de forma pública o privada a todos los usuarios (y, por lo tanto, se dice que está descentralizado y distribuido). La tecnología *blockchain* utiliza la criptografía y un mecanismo de consenso para verificar las transacciones, lo que garantiza la legitimidad de una transacción, evita el doble gasto y permite transacciones de alto valor en un entorno de confianza. Además de esto ofrece transparencia y elimina la necesidad de intermediarios o administradores de terceros.

La cadena de bloques tiene múltiples campos de aplicación ya que puede aportar valor añadido teóricamente, a aquellas actividades que cumplan con las siguientes condiciones (Roig y Montero, 2018):

1. Requieran almacenar datos

2. Precisen que el acceso a estos datos sea compartido entre diferentes partes y

C.C. Reconocimiento

3. Estas partes no se conozcan entre ellas o no exista confianza mutua por otro motivo.

Según Garay y col. (2015) la creación de una *blockchain* robusta debe garantizar dos propiedades fundamentales:

- Disponibilidad: Asegura que una transacción honesta que ha sido emitida acabe siendo añadida a la cadena de bloques, evitando que se produzca una denegación de servicio (*Denial of Service, DoS*)[1] por parte de nodos corruptos.
- Persistencia: Cuando un nodo da una transacción como estable, el resto de nodos, si son honestos, validarán ésta como estable haciéndola inmutable.

2.1.1. Aplicaciones de la cadena de bloques

Actualmente, sin duda alguna, una de los principales aplicaciones de la tecnología *blockchain* es el uso en criptomonedas. Sin embargo, el campo de aplicación de esta tecnología es mucho más amplio como se explica a continuación.

www.bdigital.ula.ve

2.1.1.1. Criptomonedas

La cadena de bloques puede diseñarse como una base de datos verdaderamente descentralizada y sin una autoridad central. Puede, por tanto, servir como centro de intercambios de confianza entre múltiples entidades sin que unas deban confiar en la otras, eliminando por ejemplo a las autoridades centrales (como bancos), así mismo el criterio de emisión de nuevas unidades monetarias se encuentra prefijado (Retamal y col., 2017).

Una criptomoneda es un medio de cambio digital que utiliza tecnología criptográfica para asegurar la veracidad de las transacciones. Se denomina transacción a la información correspondiente a la acción de transferir un monto de dinero entre dos partes (Vileriño, 2017).

[1]Método utilizado para interrumpir el acceso de los usuarios legítimos a una red o recurso web objetivo. Por lo general, se logra sobrecargando el destino con una gran cantidad de tráfico, que provoca que el recurso objetivo funcione mal o se bloquee por completo (Academy, 2019).

C.C. Reconocimiento

Bitcoin

Con el nacimiento de *Bitcoin*, las criptomonedas tomaron impulso en el mundo de la computación. *Bitcoin* fue propuesta en el 2008 por una persona bajo el seudónimo «Satoshi Nakamoto» (Nakamoto, 2008). Fue la primer criptomoneda descentralizada que resolvía el problema de doble gasto sin necesidad de confiar en una tercera parte de confianza.

Estructura de los bloques

- El valor *hash*[2] del bloque previo. Este valor permite que los bloques queden vinculados secuencialmente formando una cadena.
- Marca de tiempo (*timestamp*). Esta marca de tiempo permite identificar el instante en el que fue creado el bloque.
- La dificultad con la que ese bloque fue hallado.
- Un conjunto de transacciones ordenadas.
- El valor del *nonce*[3], este es el valor encontrado por fuerza bruta en el proceso de minado; el minero prueba muchos *nonce* diferentes y el *hash* se vuelve a calcular para cada valor hasta que se encuentre un *hash* que contenga el número necesario de bits a cero.
- Un número de versión.
- Información. Por último, el bloque contiene la información.

En la siguiente Fig. 2.1 se observa un ejemplo de cómo es la estructura de un bloque en *bitcoin*.

[2]Función matemática para transformar una información determinada (como un texto o un bloque) en una una secuencia alfanumérica única de longitud fija. Ver Sección 2.1.3.

[3]Valor que se establece de modo que el *hash* contenga un número determinado de ceros.

www.bdigital.ula.ve

C.C. Reconocimiento

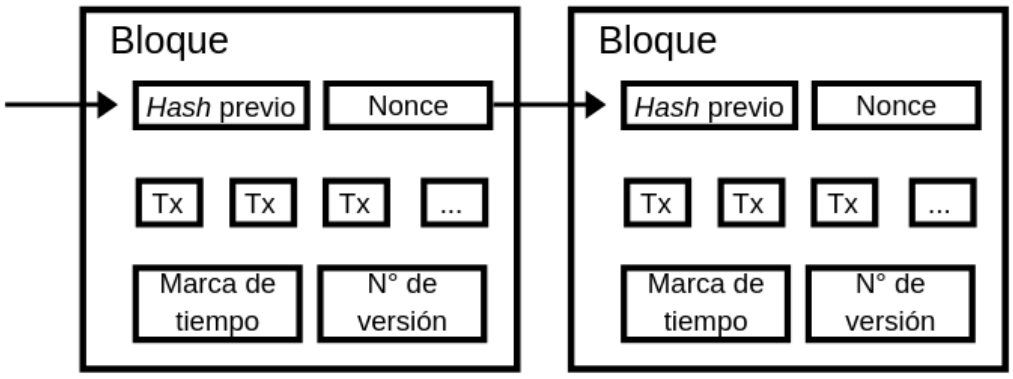


Figura 2.1: Estructura de un bloque en *bitcoin*. Adaptado de Nakamoto (2008).

Transacciones Cuando se requiere realizar una transferencia de dinero digital de un usuario a otro usuario, el proceso que se lleva a cabo para generar la transacción es crear la firma digital del *hash* criptográfico del par formado por la transacción anterior que usó ese dinero y la clave pública del destinatario. De esta forma, el destinatario puede verificar que el emisor era realmente el dueño del dinero, verificando la firma digital contra la transacción con el *hash* dado, y además, puede volver a transferirla usando su propia clave privada. En la Fig. 2.2 se presenta una esquematización de este proceso, en donde se realizan tres transacciones de la criptomoneda.

En este sentido, de acuerdo con Vileriño (2017) una transacción en *bitcoin* está compuesta por una lista de entradas (*hash*, índice, clave, firma) que hacen referencias a salidas de transacciones anteriores. Una entrada contiene:

www.bdigital.ula.ve

C.C. Reconocimiento

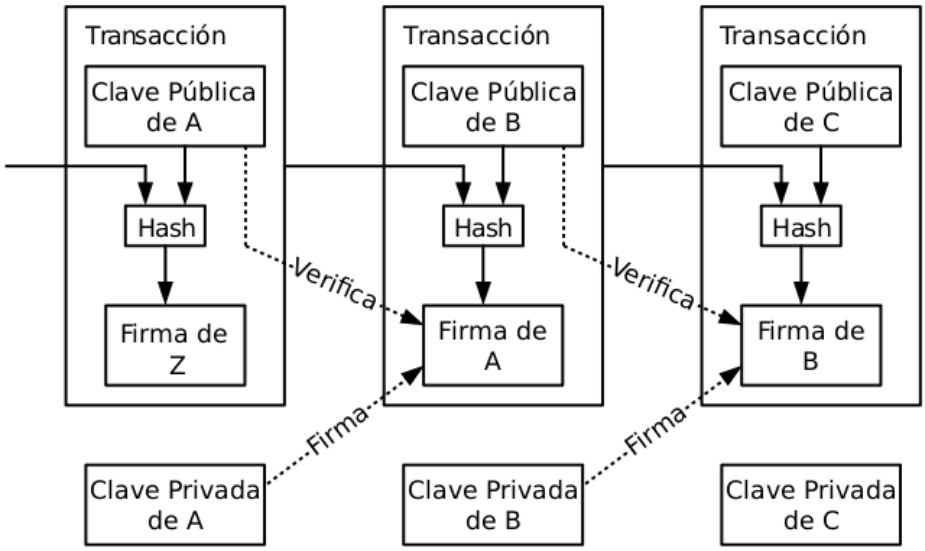


Figura 2.2: Esquema de transacciones de una criptomoneda. Adaptado de Nakamoto (2008).

- El *hash* de la transacción a la que hace referencia.
- El índice de la salida (*output*) deseada dentro de la transacción.
- La clave pública correspondiente a dicha salida.
- Una firma digital del *hash* de la transacción usando la clave privada correspondiente a la clave pública de la entrada.

La forma de verificar la validez de las referencias a salidas de transacciones anteriores es como se explica a continuación: se calcula el *hash* de la clave pública y se verifica que sea igual al que figura en la salida referenciada; luego basta con verificar la firma digital con esa clave pública. Después de verificar la validez de la misma es agregada a la cadena de bloques a formar parte de un bloque.

Nodos de la red Para que la red funcione correctamente es necesario el consenso de nodos que es la validación democrática de la misma. En el consenso de nodos es donde se validan todas las transacciones y son propagadas por la red, con el fin de evitar ataques maliciosos y dobles gastos.

www.bdigital.ula.ve

C.C. Reconocimiento

En *bitcoin* existen nodos creadores de transacciones y nodos mineros, los primeros son los encargados de crear transacciones. Por su parte los nodos mineros son aquellos encargados de propagar la información por la red; ante la llegada de un bloque a un nodo, éste lo valida, lo agrega a la cadena de bloques y lo retransmite a sus nodos vecinos (Nakamoto, 2008).

Para conseguir este consenso se parten de unas condiciones muy sencillas: cualquier agente puede conectarse a otro nodo de la red, y recibir transacciones para formar un bloque con ellas. Un bloque tiene un tamaño máximo de un *megabyte*. Cuando tienen un bloque formado, intentan resolver un rompecabezas computacional. El rompecabezas consiste en encontrar un parámetro (*nonce*) que consiga que al hacer el *hash* sobre todo el bloque (incluido el *nonce*) se obtenga un valor inferior a la dificultad actual establecida por la red.

Dicho de otra forma, se trata de encontrar un *nonce* que consiga un valor *hash* del bloque con un determinado número de ceros al inicio. Debido a las características de la función de *hash*, no es posible calcular estos valores analíticamente, es decir, para obtener un bloque válido, el minero debe recurrir a la fuerza bruta: probar valores del parámetro *nonce* hasta hallar uno válido. Además, en el caso de que el agente haya agotado todos los *nonces* sin cumplir el objetivo, puede actualizar la marca de tiempo que posee la cabecera del bloque para variar igualmente el resultado del *hash* (Cámara, 2018).

www.bdigital.ula.ve

Existen decenas de criptomonedas, todas ellas comparten su utilidad como sistema de pago. Algunas utilizan una *blockchain* propia y otras funcionan encima de la *blockchain* de *bitcoin*. Su funcionamiento es bastante heterogéneo y todas ellas pretenden aportar alguna mejora respecto a *bitcoin*.

2.1.1.2. Internet de las cosas y la cadena de bloques

A medida que la Internet de las cosas o IoT (de sus siglas en inglés, *Internet of Things*) vaya convirtiéndose en realidad, el número de dispositivos conectados a la red de redes crecerá exponencialmente (Greenough, Jhon, 2015). Ahora, además de computadores y equipos de red, se van conectando a Internet electrodomésticos, dispositivos de vigilancia y seguridad, sensores de todo tipo, entre otros, sin duda este

C.C. Reconocimiento

nuevo ecosistema ofrece una flexibilidad sin precedentes.

Sin embargo, este paradigma tiene también algunos retos a resolver. Entre ellos, destaca el ámbito de la seguridad; uno de los primeros problemas de seguridad que aparece en el entorno IoT es el nivel de supervisión de los dispositivos. En este sentido, en el entorno IoT, muchos dispositivos no están supervisados con los niveles usados en el mundo de la computación (ordenadores personales o teléfonos inteligentes). Este escenario, bien podría solucionarse con una *blockchain*; en este caso, se aprovecharía tanto su característica de sistema distribuido y su transparencia, como su robustez y fiabilidad.

2.1.1.3. Seguridad en *Big Data*

Varias técnicas de *big data* se usan actualmente para analizar la *blockchain* e incrementar sus niveles de seguridad. Estas técnicas permiten deducir las identidades de los nodos en las criptomonedas, detectar fraudes y mapear los flujos reales de dinero (Farell, 2015).

www.bdigital.ula.ve

La relación inversa, sin embargo, es aún más prometedora de acuerdo con Es-Samaali y col. (2017): utilizar la tecnología *blockchain* para dar seguridad y verificabilidad a entornos empresariales de *big data*. Con la explosión del *big data*, prácticamente toda empresa con un mínimo de clientes está interesada en sacar el máximo partido a sus datos para así mantenerse competitiva. Se trata de datos que habitualmente provienen de diversas fuentes, en diversos formatos, y son utilizados en diversos procesos por distintos departamentos de la empresa. Los peligros de estos sistemas resultan bastante evidentes: manipulación de los datos por parte de trabajadores internos, proveedores maliciosos, corrupción de los datos, fallos de almacenamiento, uso defectuoso, incumplimiento de legislaciones respecto a los datos personales y un largo etcétera.

En este contexto, la *blockchain* tiene mucho que aportar: transparencia, verificabilidad y portabilidad. Mediante *blockchain*, cada añadido en los datos, cada cambio, cada extracción para su uso o cada visualización se podría realizar utilizando un registro transparente y seguro.

Los anteriores ejemplos son de las aplicaciones más importantes que ha tenido la

C.C. Reconocimiento

tecnología de la cadena de bloques en nuevos campos de estudio, siendo estos algunos campos de estudio en que se ha aplicado.

2.1.2. Algoritmos de consenso

De acuerdo con Blummer (2018), el consenso se refiere al proceso de lograr un acuerdo entre los participantes de la red sobre el estado correcto de los datos en el sistema. El consenso lleva a que todos los nodos compartan exactamente los mismos datos. Por lo tanto, asegura que los datos en el libro mayor son los mismos para todos los nodos de la red y, a su vez, evita que los actores malintencionados manipulen los datos. Existen múltiples algoritmos de consenso en la actualidad, cada uno con sus ventajas y desventajas. Entre ellos se pueden nombrar como principales los siguientes:

Prueba de trabajo (PoW, del inglés *Proof of work*) Nakamoto (2008) describe la prueba de trabajo como un proceso que implica la búsqueda de un valor que, cuando se utiliza el *hash*, como con SHA-256, éste comienza con un número de bits definidos en cero. El trabajo promedio requerido es exponencial en el número de bits cero requeridos y se puede verificar ejecutando un solo *hash*.

www.bdigital.ula.ve

En otras palabras, la prueba de trabajo requiere que los participantes realicen un trabajo intensivo desde el punto de vista computacional pero fácil de verificar por otros miembros en la red. En el caso del *bitcoin*, los mineros compiten por añadir un conjunto de transacciones (el bloque), al *blockchain* global mantenido por la red. Para hacer esto, un minero debe ser el primero en descifrar correctamente el "*nonce*", para crear un *hash* que comienza con un número requerido de ceros. El minero ganador se anuncia públicamente y el resto puede comprobar fácilmente que hizo bien la prueba de trabajo. Una vez que todos están de acuerdo agregan el bloque a la cadena y el ciclo se repite.

Prueba de participación (PoS, del inglés *Proof of stake*) El algoritmo de prueba de participación es descrito por Cámara (2018), como un algoritmo que hace que la probabilidad de que un nodo sea elegido para generar o validar el siguiente bloque, sea proporcional a lo que tiene invertido en el *blockchain*, normalmente en

C.C. Reconocimiento

forma de monedas. La recompensa suele ser en forma de comisiones por transacción. El algoritmo se basa en la preselección del nodo que generará el siguiente bloque de entre una serie de nodos validadores que deben bloquear parte de su criptomoneda para poder optar a crear nuevos bloques. Tras la generación, el bloque debe ser aceptado por el resto de los nodos validadores mediante un proceso de votación. Los nodos reciben recompensa por participar en la generación y en la validación de la cadena.

La principal ventaja que ofrece la prueba de participación con la prueba de trabajo es la disminución del costo computacional que se requiere en PoW, por esta razón ha sido una de la propuestas más aceptadas por las criptomonedas. Un posible riesgo que debe tenerse en cuenta al implementar este tipo de algoritmos es el posible beneficio que un nodo puede obtener de votar en varios bloques a la vez, aún sabiendo que uno de ellos no es válido.

2.1.2.1. Algoritmos de consenso aplicados a grafos acíclicos dirigidos

Prueba de Participación Delegada (DPoS, del inglés *Delegated Proof of Stake*) En la prueba de participación delegada, en lugar de apostar monedas para validar transacciones, los propietarios de *tokens* votan por un grupo selecto para que cumpla la función de validar transacciones. El DPoS permanece "descentralizado" en el sentido de que todos los participantes de la red participan en la selección de los nodos que validan las transacciones, pero centralizado en el sentido de que un grupo más pequeño toma decisiones que aumentan la velocidad y la verificación de las transacciones.

Las implementaciones del DPoS mantienen una reputación, un proceso de votación continuo y un sistema de reorganización que mantiene a los validadores electos responsables y honestos. Las ventajas del DPoS con respecto a la prueba de trabajo es que es un algoritmo que permite la escalabilidad y proporciona una rápida verificación de las transacciones, pero la desventaja es que está parcialmente centralizado y el modelo de gobernanza aún se encuentra en estudios e implementaciones en proyectos de gran envergadura.

www.bdigital.ula.ve

C.C. Reconocimiento

Transacciones como Prueba de Participación (TaPOS, del inglés *Transaction As Proof Of Stake*) Las Transacciones como Prueba de Participación, es una de las características únicas de DPoS. Esta permite que cada transacción en la red pueda incluir opcionalmente el *hash* de un bloque reciente. Con ello, el firmante de la transacción puede estar seguro de que su transacción no puede aplicarse a ninguna otra instancia.

Esta característica evita situaciones de doble gasto. Además, su uso ayuda a que todas las partes terminen certificando la integridad del historial de transacciones (Larimer, 2018b).

2.1.3. Función *Hash*

La palabra *hash* tiene diferentes significados, pero el área de interés en este desarrollo es en el contexto de una función criptográfica. De acuerdo con Lizama y col. (2019), una función *hash* es un procedimiento criptográfico donde se emplea un algoritmo específico para transformar una información determinada en una una secuencia alfanumérica única de longitud fija, denominada *hash*. Entre las características esenciales de las funciones *hash* para el área de cadena de bloques se tiene:

www.bdigital.ula.ve

- Es una función unidireccional, es decir, es fácil generar el *hash* pero es prácticamente imposible reconstruir los datos.
- Es una función determinista, para una misma información de entrada siempre se genera un valor *hash* idéntico.
- Es resistente a las colisiones ya que el valor del *hash* debe ser único para cada contenido.

En la Fig. 2.3 se observa un breve ejemplo del funcionamiento de una función *hash*.

C.C. Reconocimiento

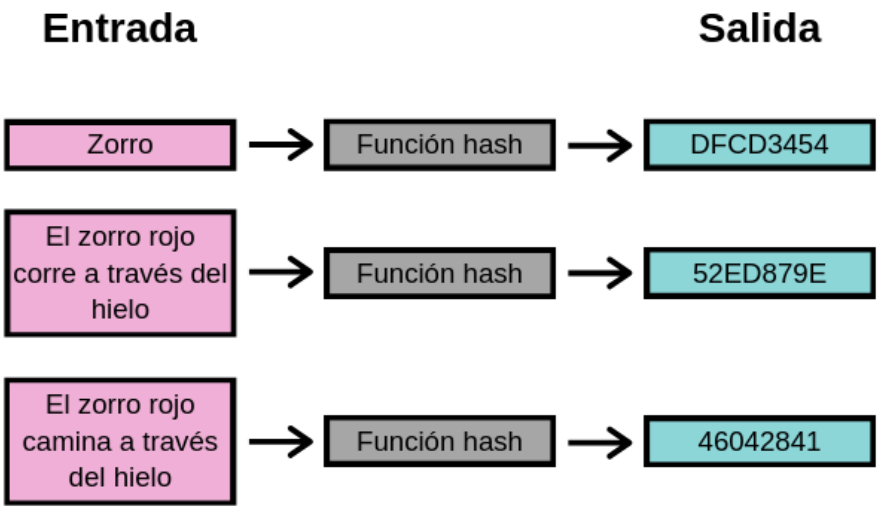


Figura 2.3: Ejemplo de una función *hash*. Adaptado de Dacak (2015).

2.1.4. Métricas de rendimiento

www.bdigital.ula.ve

Mahesh (2018) enumera las métricas de rendimiento como se explican a continuación:

1. Latencia de lectura: es el tiempo entre el momento en que se envía la solicitud de lectura y cuando se recibe la respuesta. Su fórmula se expresa en la Ec. 2.1.

$$\begin{aligned} \textit{Latencia de lectura} = \textit{Tiempo en que se recibió la respuesta} \\ - \textit{tiempo de envío de solicitud} \end{aligned} \tag{2.1}$$

2. Rendimiento de lectura: es una medida de cuántas operaciones de lectura se completan en un período de tiempo definido, expresado como lecturas por segundo (RPS, del inglés *Readings Per Second*). Esta métrica puede ser informativa, pero no es la medida principal del rendimiento de la cadena de bloques. De hecho, los sistemas normalmente se implementarán adyacentes a la cadena de bloques para facilitar lecturas y consultas significativas. El rendimiento de lectura se calcula como se expresa en la Ec. 2.2.

C.C. Reconocimiento

$$Rendimiento\ de\ lectura = \frac{Operaciones\ de\ lectura\ totales}{tiempo\ total\ en\ segundos} \tag{2.2}$$

3. Latencia de transacción: es la cantidad de tiempo necesario para que el efecto de una transacción sea utilizable en toda la red. La medición incluye el tiempo desde el momento en que se envía hasta el punto en que el resultado está ampliamente disponible en la red. Esto incluye el tiempo de propagación y cualquier tiempo de establecimiento debido al mecanismo de consenso establecido. La ecuación que define la latencia de transacción es la Ec. 2.3. Esta medida de rendimiento es comúnmente llamada tiempo de cola de la red.

$$Latencia\ de\ transacción = tiempo\ de\ confirmación\ -\ tiempo\ de\ envío \tag{2.3}$$

4. Rendimiento de transacción: es la tasa a la cual las transacciones válidas son confirmadas en todos los nodos de la red de la cadena de bloques en un período de tiempo definido. El rendimiento de transacción es calculado según la Ec. 2.4. Esta medida de rendimiento es comúnmente llamada transacciones por segundo (tps).

www.bdigital.ula.ve

$$Rendimiento\ de\ transacciones = \frac{Total\ de\ transacciones\ firmadas}{tiempo\ total\ en\ segundos} \tag{2.4}$$

2.2. Criptografía

Bajo la definición dada por Paredes (2006) la palabra criptografía proviene en un sentido etimológico del griego *Kriptos* = ocultar, *Graphos* = escritura, lo que significaría ocultar la escritura, o en un sentido más amplio sería aplicar alguna técnica para hacer ininteligible un mensaje. La criptografía es la ciencia encargada de diseñar funciones o dispositivos, capaces de transformar mensajes legibles a mensajes cifrados de tal manera que esta transformación (cifrar) y su transformación inversa (descifrar) sólo pueden ser factibles con el conocimiento de una o más claves.

C.C. Reconocimiento

La criptografía se puede clasificar históricamente en dos: La criptografía clásica y la criptografía moderna. El uso de criptografía en *blockchain* ha sido la criptografía moderna bajo el grupo de criptografía asimétrica, pero para entender el concepto se explicará también el grupo de criptografía simétrica.

2.2.1. Criptografía Simétrica

La criptografía simétrica o de clave secreta es aquella que utiliza algún método matemático llamado sistema de cifrado para cifrar y descifrar un mensaje utilizando únicamente una clave secreta. Se puede observar en la Fig. 2.4 que la línea punteada es el eje de simetría: lo mismo que hay de un lado existe exactamente igual en el otro, esto ilustra el hecho del porqué se le da el nombre de criptografía simétrica.

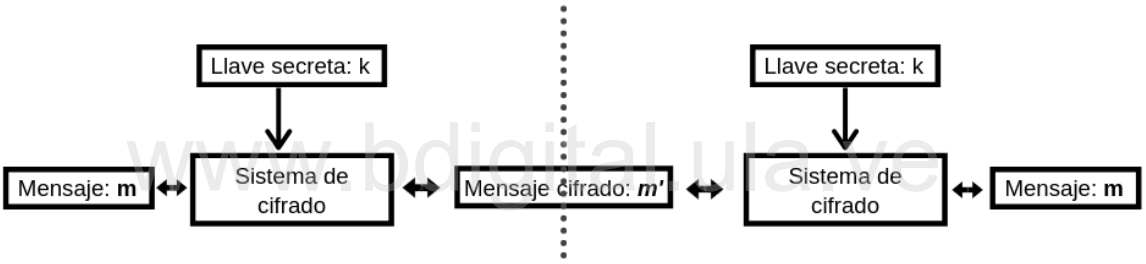


Figura 2.4: Criptografía Simétrica. Fuente Paredes (2006).

2.2.2. Criptografía Asimétrica

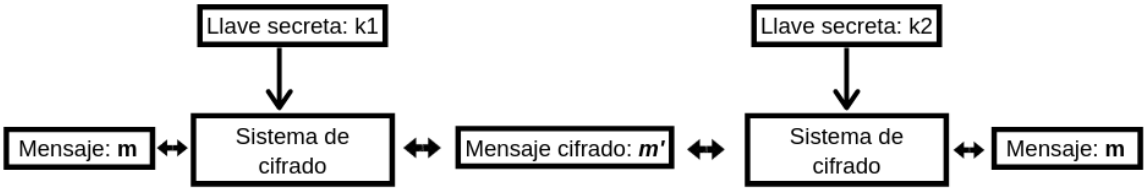


Figura 2.5: Criptografía Asimétrica. Fuente Paredes (2006).

C.C. Reconocimiento

Si se observa la Fig. 2.5, que ilustra la idea de criptografía de clave pública, se puede ver claramente que no existe simetría en ella, ya que de un lado de la figura se cifra o descifra con una clave pública y en el otro lado con una privada. De este hecho es de donde la criptografía asimétrica debe su nombre. Para este tipo de criptografía lo que se cifra con una clave se puede descifrar con la otra clave. Algunos ejemplos de este tipo de criptografía son RSA y Curvas Elípticas.

Este sistema de criptografía asimétrica es uno de los pilares de la tecnología de la cadena de bloques.

2.2.2.1. Rivest-Shamir-Adleman (RSA)

De acuerdo con Bhanot y Hans (2015), RSA significa Ron Rivest, Adi Shamir y Leonard Adleman, quienes lo desarrollaron y lo describieron públicamente en 1978. Un usuario de RSA crea y luego publica el producto de dos números primos grandes (P * Q), junto con un valor auxiliar (I), como su clave pública. Los factores primos (P * Q) deben mantenerse en secreto. Cualquiera puede usar la clave pública para cifrar un mensaje, pero con los métodos publicados actualmente, si la clave pública es lo suficientemente grande, solo alguien con conocimiento de los factores primos puede decodificar el mensaje de manera factible. El algoritmo RSA se puede usar tanto para el cifrado de clave pública como para las firmas digitales. Su seguridad se basa en la dificultad de factorizar enteros grandes.

www.bdigital.ula.ve

2.2.2.2. Criptografía de curva elíptica

La criptografía de curva elíptica (ECC, por sus siglas en inglés *Elliptic Curve Cryptography*) fue descubierta en 1985 por Victor Miller de IBM y Neil Koblitz de la Universidad de Washington como un mecanismo alternativo para implementar la criptografía de clave pública.

Según Bhanot y Hans (2015), ECC se basa en estructuras algebraicas de curvas elípticas sobre campos finitos, es decir, teoría de curvas elípticas. ECC crea claves más rápidas, más pequeñas y más eficientes en comparación con otros algoritmos de cifrado. Este algoritmo es tan eficiente que puede proporcionar un nivel de seguridad con una clave de 164 bits que otros sistemas requieren una clave de 1.024 bits para alcanzar

C.C. Reconocimiento

ese nivel de seguridad, es decir, ofrece la máxima seguridad con tamaños de bits más pequeños, por eso consume menos energía.

Básicamente, una curva elíptica es una curva plana sobre un campo finito (en lugar de números reales). El cifrado se realiza en forma de ecuación de curva elíptica que consiste en los valores puntuales que satisfacen la ecuación: $y^2 = x^3 + Ax + B$, donde A y B son los valores constantes de un punto.

La principal ventaja de ECC utiliza una longitud de clave corta que conduce a una velocidad de cifrado rápida y a un menor consumo de energía. La desventaja de ECC es que aumenta el tamaño del texto cifrado y la segunda desventaja es que ECC depende de ecuaciones muy complejas que conducen a aumentar la complejidad del algoritmo de cifrado.

2.3. Teoría de grafos

2.3.1. Definición de grafos

www.bdigital.ula.ve

Un grafo es una estructura que representa relaciones e interdependencias entre objetos, y las características que los relaciona. Se definirá el concepto de grafos acíclicos dirigidos (DAG, del inglés *Directed Acyclic Graph*) según lo que explica Rüegg y col. (2016)

2.3.2. Grafos acíclicos dirigidos

Definición 1. Un grafo G es un par $G = (V, E)$ consistente de un conjunto finito $V \neq \emptyset$ y un conjunto E de dos elementos subconjuntos de V. Los elementos de V son llamados vértices. Un elemento $e = \{a, b\}$ de E es llamada una arista con vértices finales a y b. Se dice que a y b son incidentes con e y que a y b son adyacentes o vecinos uno de otro, y se define como $e = ab$ o a e b.

Definición 2. Para determinar la relación que existe entre la información de los vértices (que las conexiones no modelan) se define un dígrafo. Un dígrafo, existe cuando el conjunto de conexiones $A = A(G)$ es dirigido, es decir, se distinguen entre las

C.C. Reconocimiento

conexiones $e_{ij} = (v_i, v_j)$ y $e_{ji} = (v_j, v_i)$, entonces el grafo $D = (V, A)$ se denomina grafo dirigido o dígrafo.

Definición 3. Ahora, si entre las conexiones el dígrafo tiene relacionado un número $T(v_i, v_j)$ que representa el costo de comunicación entre el vértice v_i y el vértice v_j, se tiene un grafo ponderado. Un grafo ponderado, es un par (G, W) donde G es un grafo y W es una función $W : E \to R^+$, de esta manera el peso de una conexión e es $W(e)$. El peso del grafo es $W(G) = \sum_{e \in E} W(e)$.

Definición 4. Un grafo que no tiene ciclos en conexiones paralelas es decir no tiene conexiones de la forma: $e_{v_i v_i}$ se denomina un grafo acíclico.

Definición 5. Dado un DAG $G = (V, E)$, donde cada nodo $v \in V$ tiene una anchura positiva w_v; una división por capas o niveles de G, (también llamada, estratificación de G) es una partición de su conjunto de nodos V dentro de subconjuntos disjuntos $V_1, V_2, \ldots, V_h$, tal que si $(u, v) \in E$ donde $u \in V_i$ y $v \in V_j$ entonces $i > j$. Un DAG con una estratificación o división por niveles, es llamado un dígrafo estratificado.

www.bdigital.ula.ve

Definición 6. La altura de un dígrafo estratificado, es el número de niveles h, del dígrafo.

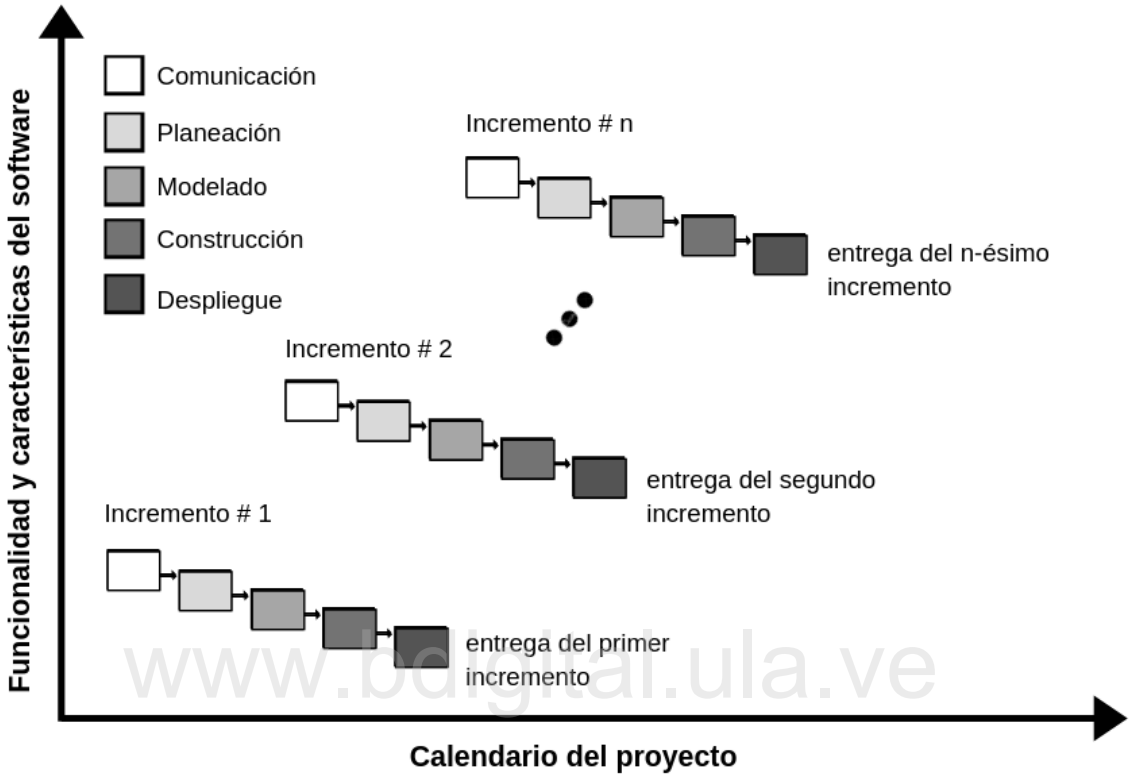
Definición 7. La anchura de un nivel V_k es tradicionalmente definido como, $w(V_k) = \sum_{v \in V_k} w_v$ y la anchura de un dígrafo estratificado (dividido en capas o niveles) es definido por la ecuación $w = max_{1 \leq k \leq h} w(V_k)$.

2.4. Modelo de desarrollo de software incremental

El modelo incremental es usado cuando los requerimientos iniciales del software están razonablemente bien definidos, pero el alcance general del esfuerzo de desarrollo imposibilita un proceso lineal. Cada secuencia lineal produce "incrementos" de software susceptibles de entregarse de manera parecida a los incrementos producidos en un flujo

C.C. Reconocimiento

de proceso evolutivo (Pressman, 2005). Las fases del modelo incremental se pueden observar en la Fig. 2.6.



www.bdigital.ula.ve

Figura 2.6: Fases de modelo incremental. Fuente Pressman (2005).

- **Comunicación**: En esta fase se recopilan, estudian y comprenden los objetivos centrales y específicos que persigue el proyecto, así como los requerimientos que ayuden a definir las características y funciones del software.
- **Planeación**: En esta etapa se definen varios incrementos en donde cada uno proporciona un subconjunto de la funcionalidad del sistema realizando un plan de proyecto de software.
- **Modelado**: Seguidamente, se procede a realizar el diseño del software que ayude a entender mejor los requerimientos del software y el diseño que lo satisfará. Así como la definición de los requerimientos que se van a entregar para el primer incremento o el siguiente incremento.

C.C. Reconocimiento

- **Construcción**: La siguiente fase consiste en la generación de código y las pruebas que se requieren para descubrir errores en el desarrollo del incremento realizado.
- **Despliegue**: Por último, se entrega el software al consumidor que lo evalúa y se obtiene retroalimentación para el siguiente incremento, hasta que se llegue a la versión final.

Con la finalización de este capítulo se obtienen los conocimientos necesarios para implementar los objetivos planteados.

www.bdigital.ula.ve

C.C. Reconocimiento

Capítulo 3

Diseño y pruebas

A continuación, se presenta el diseño de la red de cadena de bloques. En este capítulo se toman decisiones importantes a la hora de la elección del algoritmo de consenso más adecuado a la estructura de la red. Del mismo modo, se obtienen estadísticos importantes al medir el rendimiento de la misma.

www.bdigital.ula.ve

3.1. Requerimientos del sistema

La definición de los requerimientos del sistema ayuda a especificar que es lo que el sistema debe hacer, lo esencial del sistema así como entender y tener una idea de lo que el cliente tiene en su mente. Para el comienzo del desarrollo, bajo la metodología de modelo incremental, es necesario tener unos requerimientos establecidos y muy claros de lo que se quiere; la recolección de los mismos fue realizada en reuniones con el tutor semanalmente, el cual estuvo muy involucrado como usuario del sistema y cliente. De igual forma, el tutor figura como usuario del sistema para pruebas; es importante destacar que los requerimientos fueron cambiando a medida que se realizaban las entregas incrementales propias de la metodología seguida.

3.1.1. Requerimientos funcionales

Los requerimientos funcionales son las sentencias de los servicios que debe proveer el sistema, la forma en que se debe comportar o las cosas que no debería hacer. A

C.C. Reconocimiento

continuación, se listan los requisitos funcionales del sistema:

1. Cada participante de la red debe poder comunicarse con todos los demás participantes de la red.
2. Se debe incluir grafos acíclicos dirigidos como estructura de datos en la red.
3. Se debe incluir un algoritmo de consenso que maximice velocidad.
4. Debe poseer 2 tipos de participantes en la red: Creador de transacciones y minero.
5. El participante creador de transacciones es el encargado de generar transacciones dentro de la red.
6. El participante minero es el encargado de validar transacciones e incluirlas en los bloques.

3.1.2. Requerimientos no funcionales

www.bdigital.ula.ve

Los requerimientos no funcionales del sistema son aquellas restricciones sobre los servicios o funciones que ofrece el sistema. Ellos son:

- Se debe crear una red descentralizada.
- Los participantes de la red deben comunicarse bajo protocolos estándares de seguridad.
- La red debe ser implementada en un caso de uso que permita evaluar todos los protocolos de una red de cadena de bloques.
- Cada participante de la red solo debe ser capaz de ver las transacciones o información que lo involucra.

3.1.3. Elección del lenguaje para la cadena de bloques

En el mundo *blockchain* los lenguajes preferidos para desarrollo son C/C++, java y *python*, de acuerdo con Maldonado (2018). Cada uno de ellos tiene sus ventajas y beneficios que aportan a los desarrollos.

C.C. Reconocimiento

3.1.3.1. C/C++

Las ventajas que provee este lenguaje es que debido a su "bajo nivel" sus binarios son más rápidos y su capacidad de portabilidad es muy alta. Esta característica permite que las plataformas *blockchain* sean muy rápidas, con gran capacidad de escalabilidad, además de esto su capacidad de portabilidad hace que el sistema pueda ser multiplataforma sin grandes cambios.

3.1.3.2. Java

Java es un lenguaje muy versátil y multiplataforma, siendo usado para el *front-end* de varias criptomonedas ya que permite crear interfaces multiplataforma de forma rápida y sencilla.

3.1.3.3. *Python*

Por su parte, *python* se está convirtiendo en el software de análisis de datos por excelencia. Además su diseño orientado a objetos lo hace muy simple, tiene gran portabilidad, proporciona estructuras de alto nivel y puede ampliarse con bibliotecas adaptables. Es un lenguaje de uso simple y rápido, tiene muchas herramientas ya listas para su uso, además que el código generado es elegante, ordenado y de fácil entendimiento. Así mismo, la comunidad que da soporte a *python* es amplia, lo cual es importante para el desarrollo.

www.bdigital.ula.ve

Si bien *python* no es un lenguaje óptimo para alcanzar grandes velocidades, para este estudio se eligió por sus bondades y su portabilidad, lo cual es importante para cumplir con los objetivos planteados. Además de esto se tenia conocimientos en el mismo lo que facilitaba la programación de los módulos.

3.2. Comparación de la cadena de bloques con y sin DAG

Para cumplir con los requerimientos antes expuestos se debe construir una red de cadena de bloques y a su vez una red de cadena de bloques con grafos acíclicos dirigidos

C.C. Reconocimiento

como estructura de datos. Para ello se toma como base del desarrollo una estructura de transacción fija, para que la red pueda funcionar de forma similar en ambos casos.

3.2.1. Estructura de las transacciones

Una transacción es una transición de estado que cambia los datos dentro de la cadena de bloques de un valor a otro, pero una transacción puede ser un pago, una remesa, un contrato inteligente, una recompensa, es decir cualquier tipo de intercambio de información en la red.

La estructura de las transacciones propuesta es la siguiente:

- Autor.
- Destino.
- Contenido.
- Fecha.

www.bdigital.ula.ve

Las cuales pertenecen a un bloque, donde es indiferente si se trata del módulo de cadena de bloques o el módulo de cadena de bloques con DAG. La decisión de la estructura anteriormente mencionada proviene de la clásica estructura de un bloque en *bitcoin* explicada anteriormente (véase Fig. 2.2). Donde la clave pública de un participante A está representado con el campo **Autor**, la clave pública de un participante B está representado con el campo **Destino**, el campo adicional de **Contenido** es necesario para incluir los datos a transmitir y el campo **Fecha** es utilizado para trazar la cronología de las transacciones.

La autenticación de las transacciones se basa en la criptografía asimétrica. Cada participante tiene 2 claves, una pública que es conocida por todos los participantes de la red y una privada que es secreta; al crear una transacción el participante la firma con la clave pública del destinatario y el destinatario al recibirla la desencripta con su clave privada. El Algoritmo 3.1 muestra el procedimiento llevado a cabo para crear una transacción dentro de la red.

C.C. Reconocimiento

Algoritmo 3.1: Creación de una transacción.

```
1  def crearTransaccion(autor, contenido, destino):
2      origen ← fecha ← objetivo ← vacío;
3      para participante in participantes hacer
4          si autor == participante.destino entonces
5              origen = participante.clavePublica;
6          si destino == participante.destino entonces
7              contenido ← encriptar(contenido, participante.clavePublica);
8              fecha ← fechaActual();
9              objetivo ← participante.clavePublica;
10     blockchain.agregarTransaccion(origen, objetivo, contenido, fecha);
11 fin def
```

3.2.2. Estructura de los participantes

www.bdigital.ula.ve

Los participantes de la red son todos aquellos colectivos que van a jugar un papel dentro de la misma. En el caso de este desarrollo se hace uso de la siguiente estructura para cada participante:

- Una **ip** obtenida de la ip pública desde donde se conecta el participante.
- Una **clave pública** identificadora del participante.
- Un **estado** para identificar si se encuentra activo. Tiene dos posibles valores: cierto o falso.
- Un **tipo** de participante, para determinar si es un participante creador de transacciones o minero.

3.2.3. Módulo de cadena de bloques

Para la elaboración del módulo de cadena de bloques se optó por seguir la estructura base de una criptomoneda, la cual esta dada por Nakamoto (2008). Los elementos

C.C. Reconocimiento

principales de una cadena de bloques son las transacciones, los bloques y el consenso.

3.2.3.1. Bloques

En la red de cadena de bloques cada bloque puede contener una transacción o un listado de transacciones, las mismas forman parte de un bloque que tiene la estructura presentada en la Fig. 3.1:

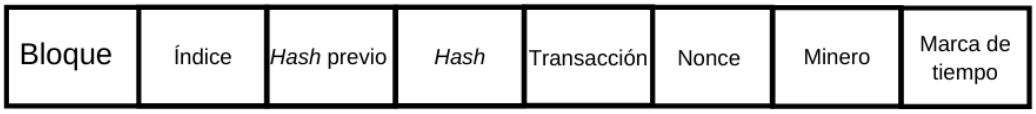


Figura 3.1: Estructura de un bloque dentro de la red.

- Un **índice** identificador del bloque.
- La **transacción** o transacciones que pertenecen al bloque.
- La **marca de tiempo** que permite identificar el instante exacto en que fue creado el bloque.
- El valor del **hash anterior** que es el *hash* perteneciente al bloque inmediatamente anterior. Este valor es el que permite que los bloques estén vinculados secuencialmente formando una cadena de bloques.
- El valor del ***hash*** calculado para el bloque.
- El ***nonce*** que es un valor que identifica la dificultad de la prueba de trabajo.
- El campo **minero** que es el identificador del participante minero que logró minar el bloque.

La estructura anteriormente expuesta es la estructura básica de una red de cadena de bloques (definido anteriormente en el Capítulo 2) con sus respectivas variaciones propias de este desarrollo.

www.bdigital.ula.ve

C.C. Reconocimiento

3.2.3.2. Cadena Principal

La principal característica de la cadena principal es la concatenación de bloques, para comenzar a formar la misma se realiza la creación de un bloque llamado el bloque «génesis». Este bloque es creado como bloque principal y a partir de él se comienza a formar la cadena principal. El Algoritmo 3.2 muestra el procedimiento llevado a cabo para la creación del mismo.

Algoritmo 3.2: Creación del bloque «génesis».

```
1 si No existe bloque «génesis» entonces
2     bloque Bloque;
3     bloque.indice ← 0;
4     bloque.transacción ← "génesis";
5     bloque.hashPrevio ← 0;
6     algoritmoConsenso ← algoritmoDeConsenso(bloque);
7     nuevoBloque(bloque, algoritmoConsenso);
8 fin
```

www.bdigital.ula.ve

En el momento de iniciar la red si no existe este bloque «génesis» no se puede formar la cadena principal, debido a que no se puede llenar el campo de *hash* previo en el bloque que esta por crearse. El bloque génesis se puede crear con cualquier información, lo realmente importante de este bloque es crear el primer *hash* para formar la cadena principal.

El proceso de formación de la cadena principal comienza con el bloque «génesis», luego los siguientes bloques comienzan a concatenarse uno detrás de otro formando una lista doblemente enlazada, como se observa en la Fig. 3.2. El procedimiento para crear un nuevo bloque es presentado en el Algoritmo 3.3.

C.C. Reconocimiento

Algoritmo 3.3: Creación de un nuevo bloque.

```
1 def nuevoBloque(bloque, algoritmoDeConsenso):
2     hashPrevio ← blockchain.hashDelBloqueAnterior;
3     si validarAlgoritmoDeConsenso() == Falso entonces
4         retornar Falso;
5     transaccionesNoConfirmadas ← [];
6     bloque.hash ← algoritmoDeConsenso;
7     añadirBloqueEnCadena(bloque);
8     retornar True;
9 fin def
```

Con la definición de la cadena principal se logra la inmutabilidad de la red, ya que si se quiere alterar un bloque de la red debe estar enlazado en la cadena y los *hashes* deben estar concatenados uno detrás del otro.

www.bdigital.ula.ve

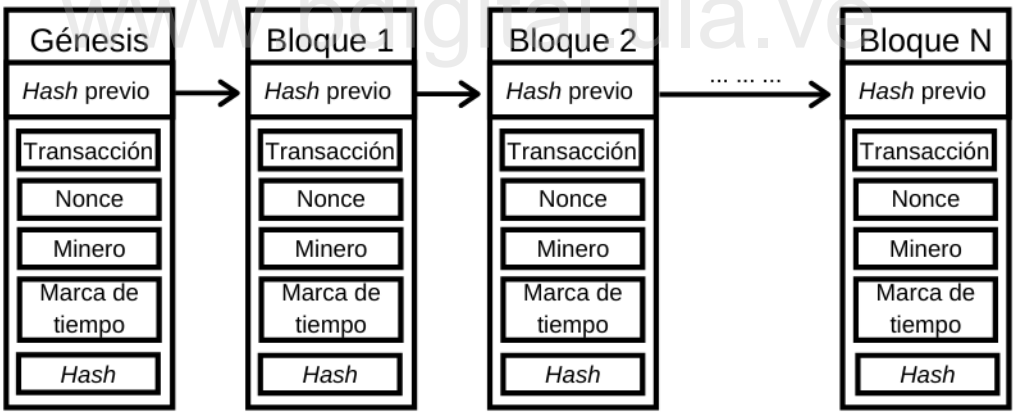


Figura 3.2: Estructura de cadena principal.

3.2.4. Módulo de cadena de bloques con DAG

3.2.4.1. Estructura del DAG

Un grafo es una estructura que representa relaciones e interdependencias entre objetos, y las características que los relaciona. Un grafo acíclico dirigido es un grafo

C.C. Reconocimiento

que no tiene ciclos, es decir que para cada vértice v no hay un camino directo que empiece y termine en v. En el caso del desarrollo la estructura del DAG es la siguiente, representada en la Fig. 3.3.

- Vértices: Representan los bloques de la red.
- Aristas: Representan la dependencia de inclusión con respecto a los nodos.

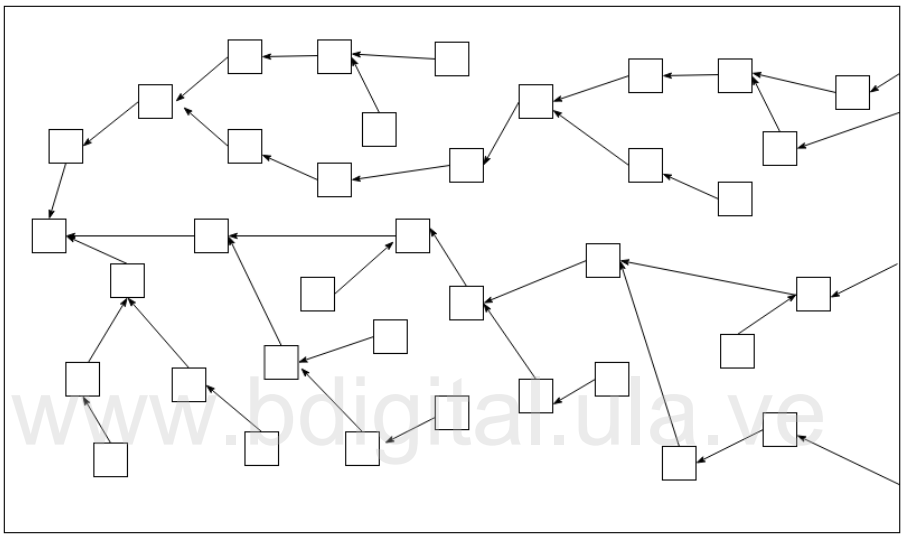

www.bdigital.ula.ve

Figura 3.3: Estructura del DAG dentro de la red.

3.2.4.2. Bloques

La estructura de los bloques no varía con respecto a la estructura de un bloque en la cadena de bloques, basada en listas doblemente enlazadas. La diferencia entre las estructuras radica en la forma de concatenar los bloques con el *hash* del bloque anterior.

El DAG comienza con una entidad llamada «génesis», similar al comienzo de la estructura de la cadena de bloques en (Nakamoto, 2008), presentado en el Algoritmo 3.4. De igual forma, se puede observar como se forma el DAG en la Fig. 3.4 representando el bloque génesis de color azul.

C.C. Reconocimiento

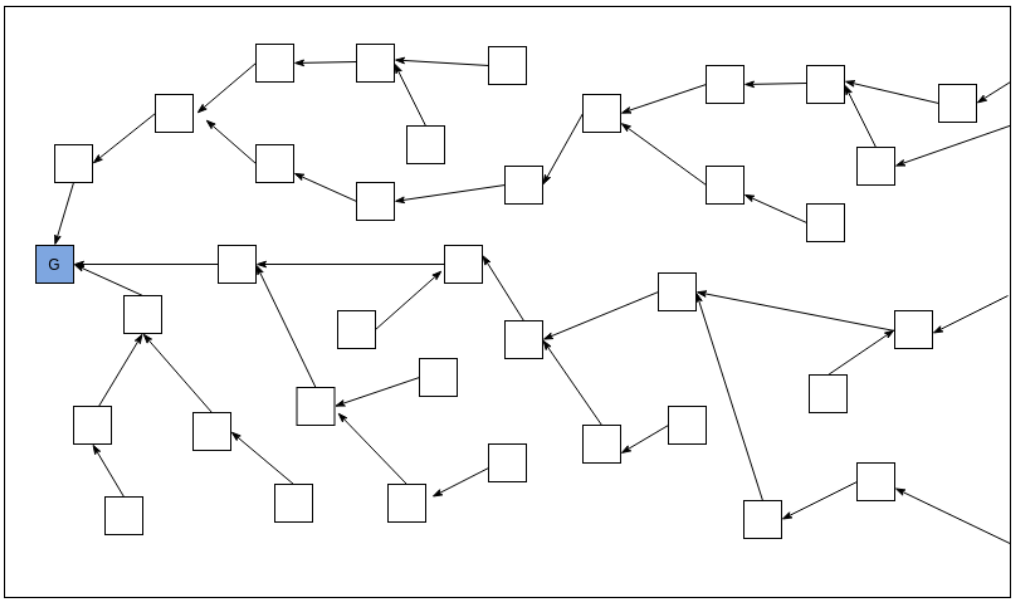


Figura 3.4: Estructura del DAG con bloque génesis.

www.bdigital.ula.ve

En el momento en que un participante quiere incluir un nuevo bloque a la estructura es referenciado al *hash* de la entidad génesis. Luego la inclusión de un nuevo bloque es referenciado al bloque anterior y así sucesivamente crece el DAG, sin limitaciones de inclusión de bloques consecutivos. Ésta estructura es similar a la estructura de la cadena de bloques, su principal diferencia radica en que no se sigue un orden de incluir un bloque detrás de otro. En el DAG existen bifurcaciones propias de la estructura, donde se pueden seguir caminos diferentes pero manteniendo la relación entre *hashes* de hijos a padres.

Con esta estructura se logra mantener la inmutabilidad de los datos de la cadena de bloques, ya que si se quiere alterar un bloque cambiará el *hash* del mismo y no coincidirá con el siguiente bloque. Esto es debido a que cada bloque sigue siendo confirmado por su padre, por todos los padres de padres y así sucesivamente. Este comportamiento se reflejaría en todos los hijos siguientes y rompería la referencia a este bloque por su *hash*.

C.C. Reconocimiento

Algoritmo 3.4: Creación del bloque «génesis» en un DAG.

```
1 si No existe bloque «génesis» entonces
2     bloque DAG;
3     bloque.indice ← 0;
4     bloque.transacción ← "génesis";
5     bloque.hashPrevio ← 0;
6     algoritmoConsenso ← algoritmoDeConsenso(bloque);
7     nuevoBloque(bloque, algoritmoConsenso);
8 fin
```

3.2.4.3. Cadena Principal

La cadena principal de Nakamoto (2008) es computacionalmente sencilla de calcular, usa una lista doblemente enlazada así que solo tiene que recorrerla para obtenerla. Sin embargo, en el caso del desarrollo se tiene un DAG como estructura de datos; por lo tanto no es tan sencilla la obtención de la misma por lo que se tienen que encontrar formas de usar la estructura del grafo para obtener una cadena principal.

www.bdigital.ula.ve

En tal sentido, el algoritmo que la red presenta consiste en recorrer cada uno de los arcos del DAG en el orden en que se crearon, de forma que se tenga un orden cronológico de las lista de bloques. Luego cada una de estas listas se ordenan en una lista mayor; una vez ordenada la lista con cada uno de los arcos, se procede a comprobar mediante el *hash* el orden de precedencia en la inserción de los mismos.

Seguidamente, en el Algoritmo 3.5 se presenta la variación realizada para la creación de un nuevo bloque en la estructura de datos DAG.

C.C. Reconocimiento

Algoritmo 3.5: Creación de un nuevo bloque en el DAG.

```
1  def nuevoBloque(bloque, algoritmoDeConsenso):
2      hashPrevio ← blockdag.hashDelBloqueAnterior;
3      si validarAlgoritmoDeConsenso() == Falso entonces
4          retornar Falso;
5      fin
6      transaccionesNoConfirmadas ← [];
7      bloque.hash ← algoritmoDeConsenso;
8      añadirBloqueEnDAG();
9      retornar True;
10 fin def
```

3.2.5. Cifrado

www.bdigital.ula.ve

Existen numerosos estudios donde se puede comparan los algoritmos ECC con algoritmos tradicionales como RSA. Uno de ellos es el realizado por Mashrufee y col. (2016), la Tabla 3.1 muestra los resultados obtenidos por este estudio.

En el estudio se concluye que el algoritmo de ECC es más eficiente con un tamaño de clave más pequeño en comparación con el algoritmo RSA y se considera principalmente para dispositivos con recursos limitados. Es por esta razón que las bondades del ECC son satisfactorias para el desarrollo, brindando seguridad y velocidad al mismo tiempo. La ventaja de este algoritmo es garantizar una alta seguridad con un tamaño de clave más corto que los tradicionales, además de esto permite realizar cálculos de alta velocidad.

C.C. Reconocimiento

Tabla 3.1: Comparación de rendimiento entre ECC y RSA.

Parámetros	ECC	RSA
Gastos generales computacionales	Se pueden guardar aproximadamente 10 veces más que la de RSA	Más que ECC
Tamaño de clave	Los parámetros del sistema y el par de claves son más cortos para el ECC	Los parámetros del sistema y el par de claves son más grandes para el RSA
Ahorro de ancho de banda	ECC ofrece un considerable ahorro de ancho de banda sobre RSA	Mucho menos ahorro de ancho de banda que ECC
Generación de clave	Más rápido	Más lento
Cifrado	Mucho más rápido que RSA	A buena velocidad pero más lento que ECC
Descifrado	Más lento que RSA	Más rápido que ECC
Eficiencia en dispositivos pequeños	Mucho más eficiente	Menos eficiente que ECC

www.bdigital.ula.ve

3.2.6. Consenso

Un algoritmo de consenso es el que se encarga de la toma de decisiones dentro de una red de cadena de bloques. En cada red se implemento un algoritmo de consenso para determinar cuál de ellos era el que se adaptaba de mejor forma al DAG. De igual manera, la red hace uso de un consenso de reconocimiento de participantes: en el momento en que un participante se conecta a la red, cada participante expone los participantes que el reconoce como pertenecientes a la red, se realiza un consenso y aquellos participantes que tengan más del 51 % de reconocimiento son aceptados. Este proceso se realiza para eliminar participantes sospechosos o que quieren incluirse

C.C. Reconocimiento

de forma malintencionada en la red. En el Algoritmo 3.6 se detalla paso a paso el procedimiento para llevar a cabo el consenso de participantes en la red.

Algoritmo 3.6: Consenso de participantes.

```
1  def consensoDeParticipantes():
2      nuevosParticipantes ← {};
3      para participante en blockdag.participantes hacer
4          si nuevosParticipantes existe entonces
5              nuevosParticipantes['cont'] ← nuevosParticipantes['cont']+1;
6          en otro caso
7              inicializar nuevosParticipantes como participante;
8          si participante == participanteEjecutor entonces
9              continuar
10         participantesDeUnParticipante ← obtenerParticipantes(participante);
11     para valores en participantesDeUnParticipante hacer
12         si nuevosParticipantes contiene valores entonces
13             nuevosParticipantes['cont'] ← nuevosParticipantes['cont'] + 1;
14         en otro caso
15             inicializar nuevosParticipantes como participante;
16     para participantes en nuevosParticipantes hacer
17         si nuevosParticipantes['cont'] >len(nuevosParticipantes) * 0.51
            entonces
18             añadirParticipante(participante);
19         en otro caso
20             eliminarParticipante(participante);
21 fin def
```

www.bdigital.ula.ve

C.C. Reconocimiento

3.2.6.1. Prueba de trabajo

El algoritmo de prueba de trabajo implementado se rigió por el propuesto por (Nakamoto, 2008). Como primer paso se requiere tener un participante del tipo minero el cual trabaja en segundo plano dentro de la red. Este participante debe encontrarse atento a cualquier nueva transacción que se genere dentro de la red. En el momento en que se crea una nueva transacción el participante minero procede a comenzar con su tarea: calcula un *hash*, si éste no cumple con la dificultad asignada por la red aumenta en 1 el valor del *nonce* y calcula de nuevo el *hash*. Este proceso es repetido por el participante minero tantas veces sea necesario para cumplir la dificultad asignada. Este proceso es explicado en el Algoritmo 3.7.

Algoritmo 3.7: Prueba de trabajo.

```
1  def hashCalculado(bloque):
2      bloque ← CifrarECC(bloque);
3      bloque ← ConvertirUTF8(bloque);
4      bloque ← ConvertirHexadecimal(bloque);
5      retornar bloque;
6  fin def
7  def pruebaDeTrabajo(bloque):
8      calcularHash = hashCalculado(bloque);
9      mientras calcularHash no comience con Blockchain.dificultad ceros hacer
10         Bloque.nonce ← Bloque.nonce + 1;
11         calcularHash ← hashCalculado();
12     retornar calcularHash;
13 fin def
```

Seguidamente, el primer participante minero que consiga el *hash* envía la información al resto de participantes para que ellos corroboren la validez del mismo. Si el bloque es aprobado por el 50 % +1 de los participantes del consenso se considera válido y es añadido a la cadena, si es invalido es descartado.

www.bdigital.ula.ve

C.C. Reconocimiento

La prueba de trabajo es un proceso computacionalmente costoso, pero que su resultado puede ser verificado rápidamente por la red, el Algoritmo 3.8 demuestra esta premisa.

Algoritmo 3.8: Validar prueba de trabajo.

```
1 def validarPruebaDeTrabajo(bloque, hashBloque):
2     si hashBloque comienza con Blockchain.dificultad ceros y hashBloque ==
        bloque.calcularHash() entonces
3         retornar Cierto;
4     en otro caso
5         retornar Falso;
6     fin
7     fin def
```

3.2.6.2. Transacciones como prueba de participación

www.bdigital.ula.ve

Como se explica en la Sección 2.1.2.1 el algoritmo de prueba de participación tiene dos variantes que son aplicadas en los grafos acíclicos dirigidos: El algoritmo de prueba de participación delegada y el algoritmo de transacciones como prueba de participación. Para la red se decidió usar el algoritmo de transacciones como prueba de participación debido a que no se estaba desarrollando una red con *tokens* que pudieran ser usados para el algoritmo de prueba de participación delegada. Este algoritmo es una variante del algoritmo de consenso de prueba de participación delegada.

DPOS fue creado por Larimer (2018a) en el inicio de las redes *BitShares* para favorecer la escalabilidad de los sistemas sin perjudicar la descentralización en la creación de bloques. Funciona mediante el uso de sistemas de reputación y votación en tiempo real para crear un panel de partes confiables limitadas. Estas partes son llamadas testigos, los cuales son seleccionados por los accionistas.

Los accionistas son todos los participantes de la red que tienen mayor cantidad de votos directamente proporcional a la cantidad de *tokens* que poseen. En el caso de Prueba de participación por transacciones es usado como valor de peso son la cantidad de transacciones que posee el participante. En el Algoritmo 3.9 se explica como se

C.C. Reconocimiento

realiza la elección de los testigos por parte de los accionistas de la red, en este caso aquellos participantes con el mayor número de transacciones.

Algoritmo 3.9: Selección de testigos en la red.

```
1 def seleccionTestigos():
2     participantes ← blockchain.participantes;
3     para participante en participantes hacer
4         participante.cont_trans ← calcularNumeroTransacciones(participante);
5     fin
6     participantes.cont_trans.sort() /* ordenar de mayor a menor        */
7     retornar participantes[0:21];
8 fin def
```

Los testigos son los encargados de tomar las decisiones dentro de la red para crear y rechazar bloques, además de la validación de transacciones. Si un testigo realiza mal su trabajo o se comprueba que está realizando transacciones fraudulentas es removido de su posición y suplantado por otro participante. En el Algoritmo 3.10 se muestra el procedimiento llevado a cabo para realizar el consenso.

www.bdigital.ula.ve

Algoritmo 3.10: Algoritmo de transacciones como prueba de participación.

```
1 def transaccionesComoPruebaDeParticipacion(bloque):
2     testigos ← seleccionTestigos();
3     calcularHash ← hashCalculado(bloque);
4     bloque.minero ← testigos.pop(0).public_key;
5     validarBloque(bloque);
6     nuevoBloque(bloque, calcularHash);
7     testigos.append(bloque.minero);
8 fin def
```

En el momento de validar la información devuelta por el anterior algoritmo, se hace uso del Algoritmo 3.11. El cual cumple el mismo objetivo que el realizado por la validación para la prueba de trabajo: verificar la validez del *hash* encontrado.

C.C. Reconocimiento

Algoritmo 3.11: Validar transacciones como prueba de participación.

```
1 def validarTaPoS(bloque, hashBloque):
2     si hashBloque == bloque.calcularHash() entonces
3         retornar Cierto;
4     en otro caso
5         retornar Falso;
6     fin def
```

3.2.7. Comparación de redes y elección de algoritmo de consenso

Como siguiente paso, se requiere realizar un experimento haciendo uso de los dos algoritmos de consenso anteriormente explicados. De igual forma, comprobar las diferencias de las estructuras de datos en cada una de las diferentes redes. Este experimento se realiza con el objetivo de corroborar conceptualmente los algoritmos explicados con anterioridad.

www.bdigital.ula.ve

3.2.7.1. Diseño del experimento

Para el experimento se generó un *script* con 2 participantes de prueba en cada red (cantidad mínima de participantes para establecer una comunicación). Cada red ejecuta un algoritmo de consenso diferente: Prueba de trabajo para la red de *blockchain* y transacciones como prueba de participación para la red *blockdag*. El objetivo del experimento consiste en probar el rendimiento en transacciones con indicadores de:

- Número de transacciones por segundo.
- Tamaño de la cola de transacciones.
- Tiempo de cola promedio.

En la simulación cada algoritmo recibe 50 transacciones por segundo para validar. Se quiere verificar cuál de los dos algoritmos valida más transacciones por segundo.

C.C. Reconocimiento

Para el algoritmo de la prueba de trabajo se estableció un *nonce* con dificultad 4 que tarda aproximadamente 10-20 segundos para minar cada bloque. El experimento se realizó por 300 segundos.

3.2.7.2. Resultados

Los resultados obtenidos para este experimento se resumen en la Tabla 3.2. Para el indicador de transacciones por segundo se obtuvo que por prueba de trabajo se validan máximo 0,73 transacciones por segundo, mientras que con transacciones como prueba de participación se obtuvo un máximo de 23,22 transacciones por segundo, como se puede verificar en las Fig. 3.5 y Fig. 3.6.

Tabla 3.2: Rendimiento de prueba de trabajo y prueba de transacciones como participación.

Indicadores	**Prueba de trabajo**	**Transacciones como Prueba de participación**
Transacciones por segundo	0,73	23,22
Tamaño de cola	7072	29
Tiempo de cola promedio	147,97	0,332

www.bdigital.ula.ve

C.C. Reconocimiento

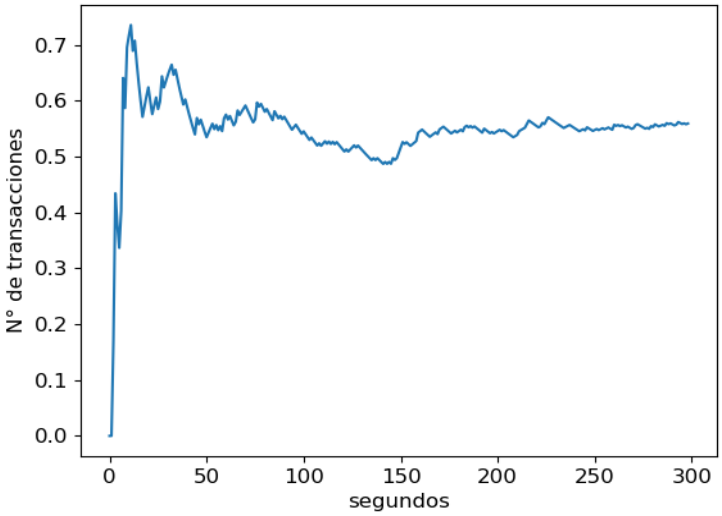


Figura 3.5: Transacciones por segundo para PoW.

www.bdigital.ula.ve

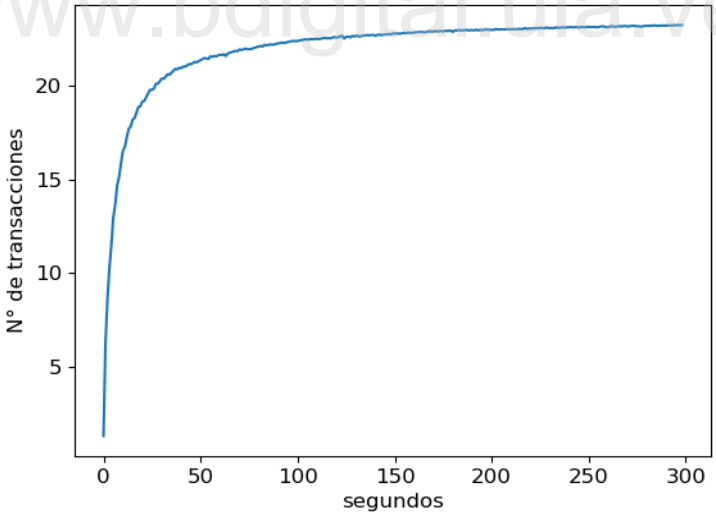


Figura 3.6: Transacciones por segundo para TaPoS.

En el caso de tamaño de cola se obtuvo como resultado para el algoritmo de consenso de prueba de trabajo un tamaño de cola de 7072 transacciones y para transacciones como prueba de participación un tamaño de cola de 29 transacciones, como se puede observar en las Fig. 3.7 y Fig. 3.8.

C.C. Reconocimiento

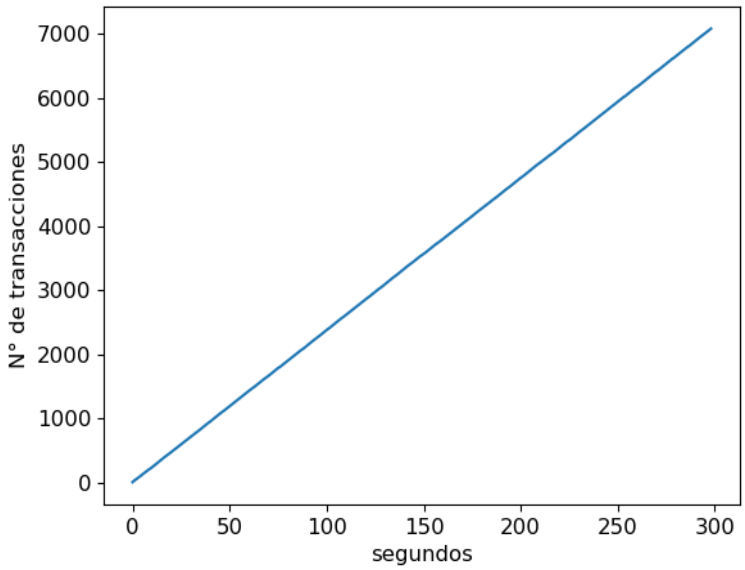
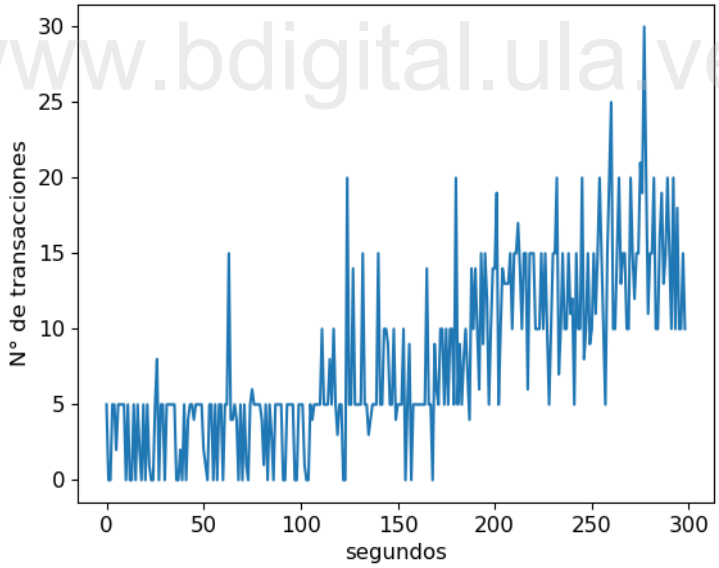


Figura 3.7: Tamaño de cola para PoW.



www.bdigital.ula.ve

Figura 3.8: Tamaño de cola para TaPoS.

Del mismo modo, para el tiempo de cola promedio en el caso del algoritmo de prueba de trabajo fue de 147,97 segundos en cola mientras que para transacciones como prueba de participación se tiene un máximo de cola de 0,332 segundos, como se muestra en las Fig. 3.9 y Fig. 3.10.

C.C. Reconocimiento

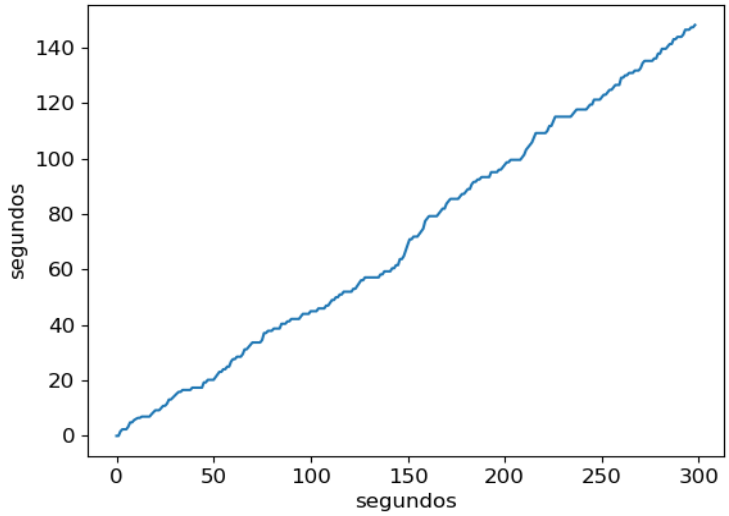


Figura 3.9: Tiempo de cola promedio para PoW.

www.bdigital.ula.ve

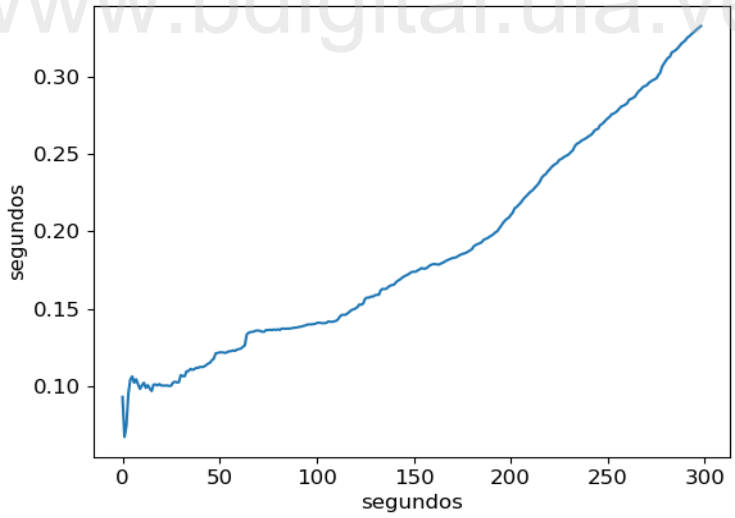


Figura 3.10: Tiempo de cola promedio para TaPoS.

3.2.7.3. Conclusiones

En los resultados obtenidos se puede observa la clara diferencia entre las dos redes. Por un lado la red *blockchain* con el algoritmo de prueba de trabajo se observa en la

C.C. Reconocimiento

Fig. 3.5 que al pasar el tiempo el tps comienza a disminuir; este comportamiento se puede ver también en las Fig. 3.7 y Fig. 3.9 al aumentar el tamaño de la cola y el tiempo de cola de promedio.

Por su parte, la red *blockdag* con el algoritmo de transacciones como prueba de participación obtuvo resultados satisfactorios como se puede observar en la Fig. 3.6 donde se mantiene en crecimiento el número de transacciones validadas por segundo; a pesar que el tiempo de cola promedio se ve en aumento (Fig. 3.10) no presenta un incremento significativo para la red.

Con los resultados obtenidos queda en evidencia la rapidez que aporta el algoritmo de transacciones como prueba de participación a la red. Por esta razón se determinó que es el algoritmo que mejor se adapta a los requerimientos expuestos.

3.2.8. Rendimiento de la red y algoritmo de consenso elegido

Seguidamente, se realizo una prueba de estrés para comprobar la estabilidad de la red. Se desea obtener mejores resultados del algoritmo de consenso al incrementar el número de participantes involucrados en la simulación.

www.bdigital.ula.ve

3.2.8.1. Diseño del experimento

En este caso, se realizo un *script* el cual involucra 12 participantes activos al mismo tiempo en la red. El objetivo de la prueba es maximizar los resultados obtenidos para los indicadores de:

- Número de transacciones por segundo.
- Tamaño de la cola de transacciones.
- Tiempo de cola promedio.

En la simulación el algoritmo recibe 200 transacciones por segundo para validar. De igual forma, el experimento se realizó por 300 segundos continuos.

C.C. Reconocimiento

3.2.8.2. Resultados

Los resultados de este experimento se resumen en la Tabla 3.3. Para el caso de transacciones por segundo se obtuvo un resultado de 57.40 tps, representado en la Fig. 3.11. Este número es considerablemente más alto que el obtenido en el experimento anterior.

Tabla 3.3: Rendimiento de la red con algoritmo prueba de transacciones como participación.

Indicadores	**Transacciones como Prueba de participación**
Transacciones por segundo	57,40
Tamaño de cola	403,63
Tiempo de cola promedio	2,73

www.bdigital.ula.ve

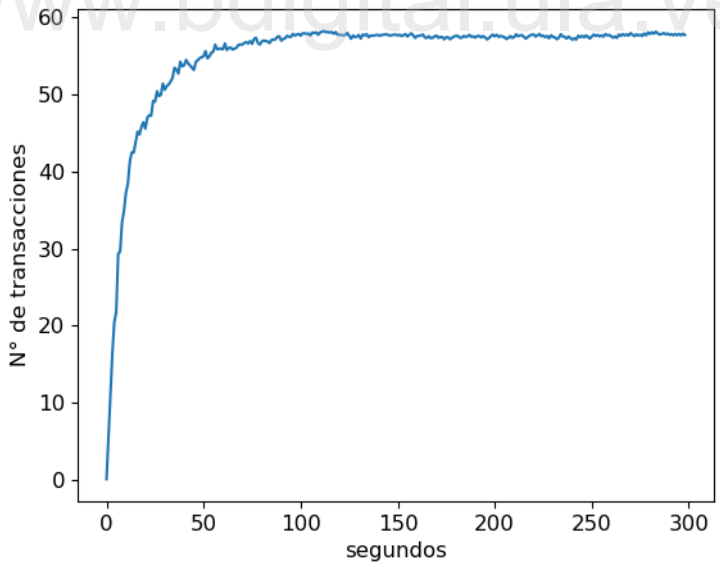


Figura 3.11: Transacciones por segundo.

En el caso del indicador de tamaño de cola de transacciones se evidenció un máximo de 403.63 transacciones. En la Fig. 3.12 se puede observar el comportamiento de la red.

C.C. Reconocimiento

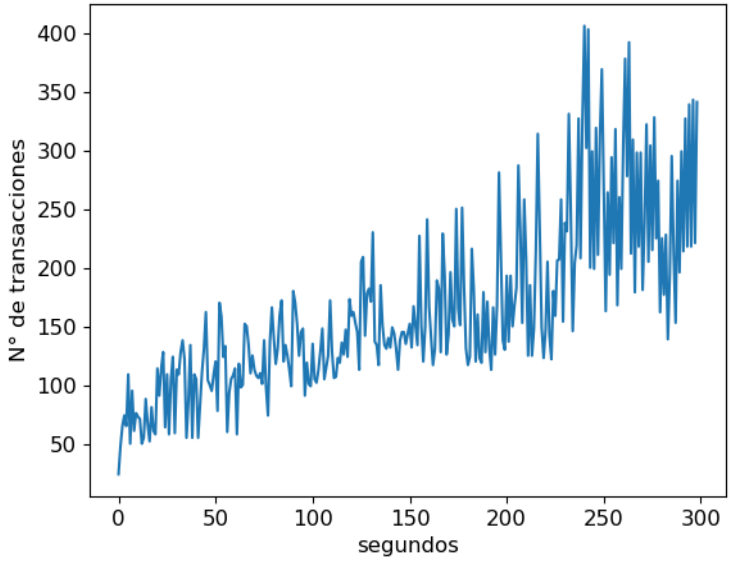


Figura 3.12: Tamaño de cola.

Así mismo, para el indicador tiempo de cola promedio se observo que el pico máximo obtenido de cola fue de 2.73 segundos, como se puede apreciar en la Fig.3.13.

www.bdigital.ula.ve

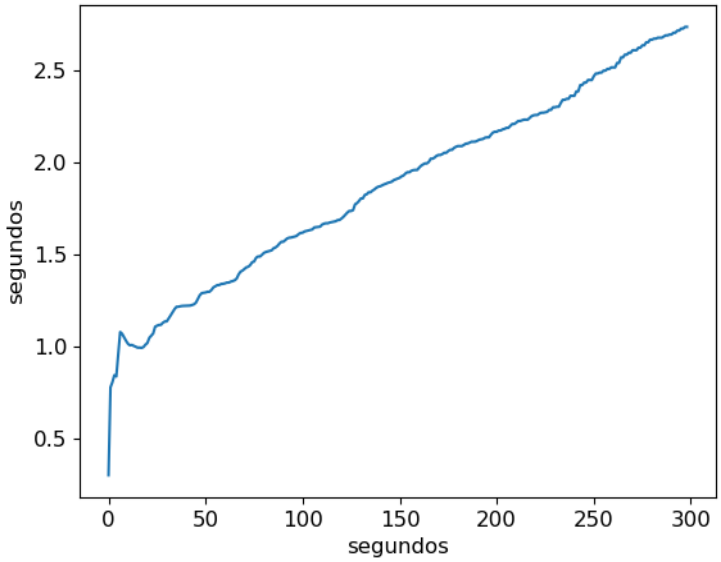


Figura 3.13: Tiempo de cola promedio.

C.C. Reconocimiento

3.2.8.3. Conclusiones

Al comparar los resultados de los dos experimentos es evidente la velocidad que aporta el algoritmo de consenso de transacciones como prueba de participación. Se puede observar que el número de transacciones validadas por segundo aumenta de 23 tps a 57,4 tps. Del mismo modo, el tamaño de cola tiene un pico de 403,63 transacciones en cola pero con el pasar del tiempo desciende.

Este comportamiento de la red se debe a que mientras la red va creciendo y existen más participantes activos dentro de la misma, la velocidad con que se validan las transacciones aumenta.

3.3. Comparación de los resultados obtenidos con algunas criptomonedas

A continuación, en la Tabla 3.4 se presenta una recolección de indicadores de las criptomonedas *bitcoin, bitcoin cash, ethereum e iota.*

www.bdigital.ula.ve

Tabla 3.4: Comparación de indicadores entre la red y algunas criptomonedas.

Red	**Transacciones por segundo**	**Tamaño de cola**	**Tiempo de cola promedio**
Bitcoin	7 tps	267,52 t/seg	430,98 seg
Ethereum	14,15 tps	571,1 t/seg	13,2 seg
Bitcoin Cash	61 tps	315 t/seg	587,76 seg
IOTA	50	No proporcionado	84 seg
BLOCKDAG	57,40 tps	403,63 t/s	2,73 seg

Con respecto a las transacciones por segundo, la red logra uno de los tps mas alto de la red, superando incluso a la criptomonoda IOTA que tiene una estructura de grafos acíclicos dirigidos. Así mismo, cuenta con uno de los tamaño de cola más pequeño, así como el menor tiempo promedio de cola. Esto demuestra que el desarrollo es capaz de soportar el flujo de transacciones introducidos sin sobrecargarse.

C.C. Reconocimiento

De igual forma, los resultados obtenidos fueron satisfactorios ya que se logro obtener unas métricas de rendimiento muy similares (y en algunos casos mejores) que las criptomonedas en curso. Es importante resaltar, que estos valores son obtenidos para una red que no se encuentra en producción y es una prueba bastante reducida con respecto a las transacciones que reciben las redes de criptomonedas; de la misma manera, no se tienen otros factores que influyen en ambientes de producción como retrasos en la conectividad de los participantes.

www.bdigital.ula.ve

C.C. Reconocimiento

Capítulo 4

Implementación

En el presente capítulo se lleva a cabo la implementación de la red en un caso de uso. Para esto, la red se implementó en un sistema de envío de mensajería seguro, en el cual las transacciones representan los mensajes intercambiados entre dos usuarios.

www.bdigital.ula.ve

4.1. Requerimientos del sistema

La recolección de los requisitos del presente modulo fue realizada de la misma forma explicada en el capitulo anterior: se realizaron reuniones semanales con el tutor.

4.1.1. Requerimientos funcionales

- Permitir enviar mensajes de forma segura entre los usuarios.
- Un usuario puede ser emisor y receptor de mensajes a la vez.
- Cada usuario debe tener un historial de conversaciones.
- Cada conversación solo puede ser vista por los usuarios que pertenezcan a ella; para cualquier usuario sin permiso debe estar encriptada.

4.1.2. Requerimientos no funcionales

- Hacer uso de la cadena de bloques para el manejo seguro de la información.

C.C. Reconocimiento

4.2. Arquitectura cliente-servidor

La arquitectura cliente-servidor tiene dos partes claramente diferenciadas, una de ellas es el servidor y por otro lado se encuentra el cliente o el grupo de clientes que harán uso del servidor. Para la integración de la red con el caso de uso es necesaria la inclusión de una arquitectura que se adapte a las necesidades del sistema.

4.2.1. Elementos de la arquitectura

4.2.1.1. Cliente

Un cliente puede ser un equipo que requiere los servicios de un equipo servidor, o bien un proceso que solicita los servicios de otro. Normalmente estas peticiones son realizadas por un usuario o esta involucrado directamente de interactuar con el usuario.

4.2.1.2. Servidor

www.bdigital.ula.ve

Un servidor por su parte, es un equipo que ejecuta servicios para atender las demandas de diferentes clientes, también es un proceso que ofrece el recurso o recursos que administra los clientes que lo solicitan.

La arquitectura cliente-servidor es propicia ya que la red se comporta como servidor y el caso de uso como cliente. Como se puede apreciar en la Fig. 4.1 un cliente A realiza la petición al servidor (red *blockdag*) para enviar un mensaje; el servidor responde la petición enviando el mensaje al cliente B.

C.C. Reconocimiento

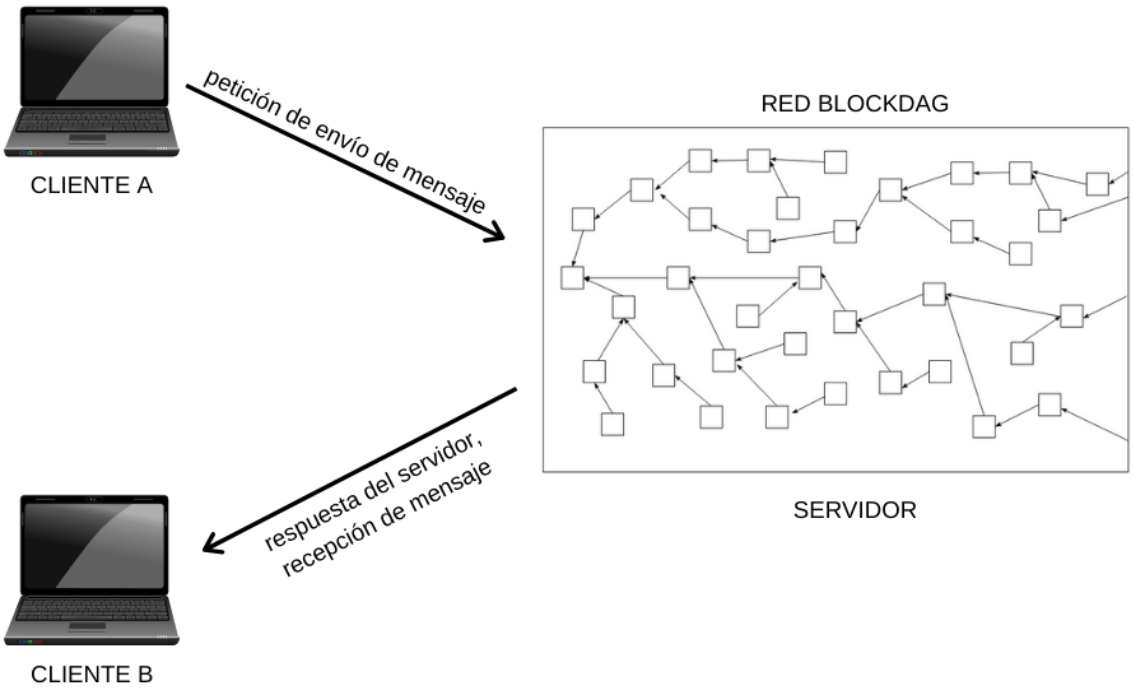


Figura 4.1: Esquema de arquitectura cliente servidor.

www.bdigital.ula.ve

4.3. Diseño de la base de datos

El envío de mensajería seguro requiere estar adaptado a la estructura de las respuestas que obtiene de la red *blockdag*, por tanto se diseño la base de datos que se observa en la Fig. 4.2.

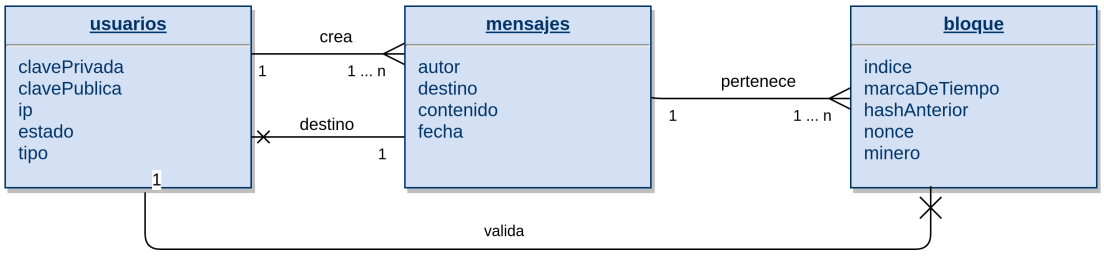


Figura 4.2: Diseño de la base de datos.

Los usuarios de la base de datos representan los participantes de la red, los cuales

C.C. Reconocimiento

están formados por la estructura de un participante: clave privada, clave pública, ip y estado; no contiene tipo ya que todos los usuarios son creadores de transacciones y la red es la que aporta los participantes validadores. De igual forma, los mensajes representan las transacciones de la red, donde el autor figura como el usuario que envía el mensaje, el destino es el usuario que recibirá el mensaje, el contenido del mensaje que se enviará y el campo fecha guarda el momento en el que el mensaje fue enviado. Por su parte, manteniendo la estructura de la red *blockdag*, se tiene una entidad bloque en la cual se encuentran encapsulados los mensajes.

4.4. Implementación

La red *blockdag* al fungir como servidor tiene varios servicios que aporta. Entre ellos se encuentran el envío de mensajes y el historial de mensajes recibidos.

4.4.1. Envío de mensajes

www.bdigital.ula.ve

El proceso de enviar mensajes es muy sencillo, tal cual como funciona enviar un mensaje en cualquier plataforma. La diferencia con otros sistemas es que cada participante puede enviar mensajes a cualquier usuario de la red, es decir sus contactos están conformados por todos los participantes que se encuentren activos dentro de la red. Seguidamente, se muestra la vista implementada para el envío de mensajes en la Fig. 4.3.

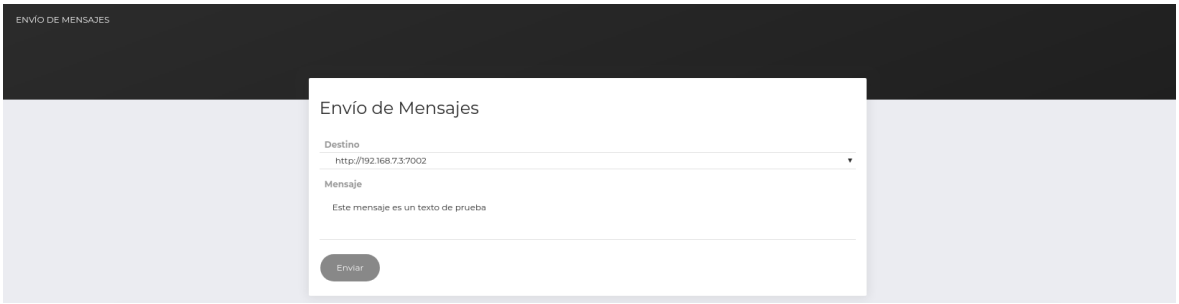
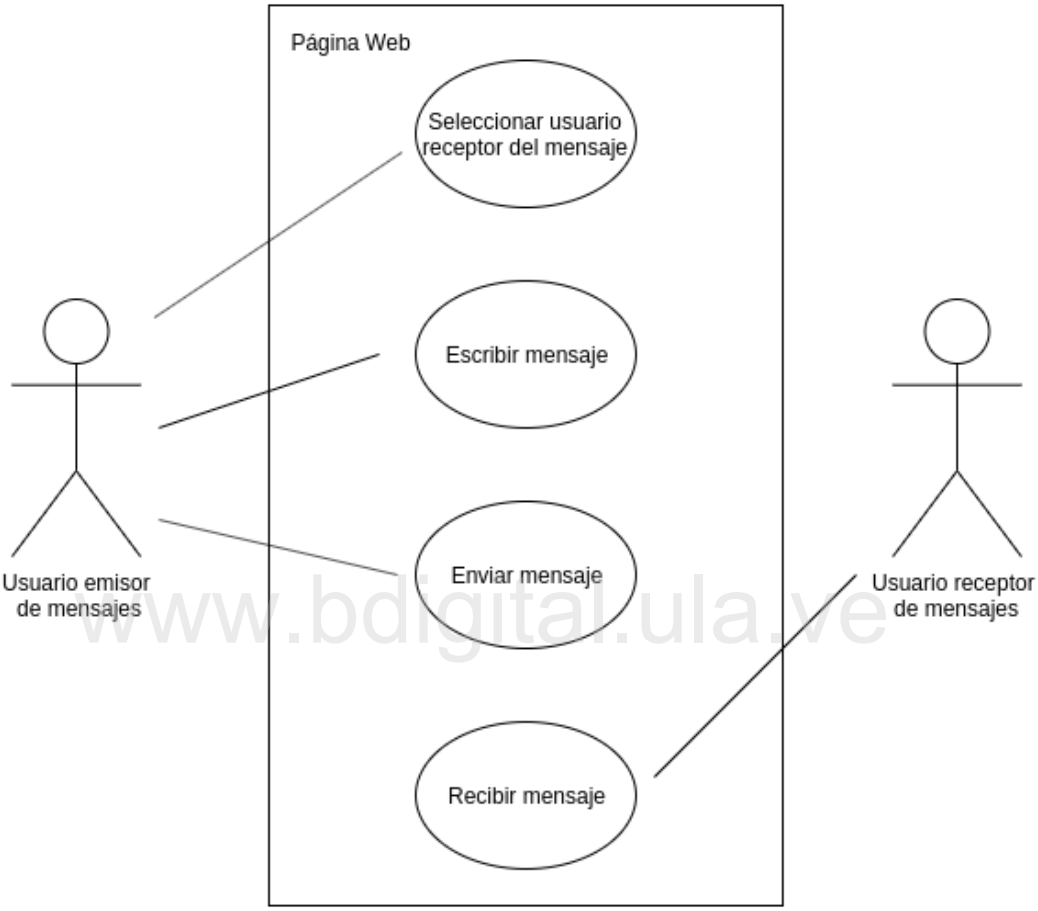


Figura 4.3: Vista de envío de mensajes.

Como primer paso, el usuario emisor del mensaje selecciona un usuario receptor del

C.C. Reconocimiento

mensaje (de la lista de participantes). Seguidamente, procede a escribir el mensaje y por último envía el mensaje. Por su parte, el usuario receptor del mensaje recibirá el mensaje. Estas tareas son especificadas en la Fig. 4.4



www.bdigital.ula.ve

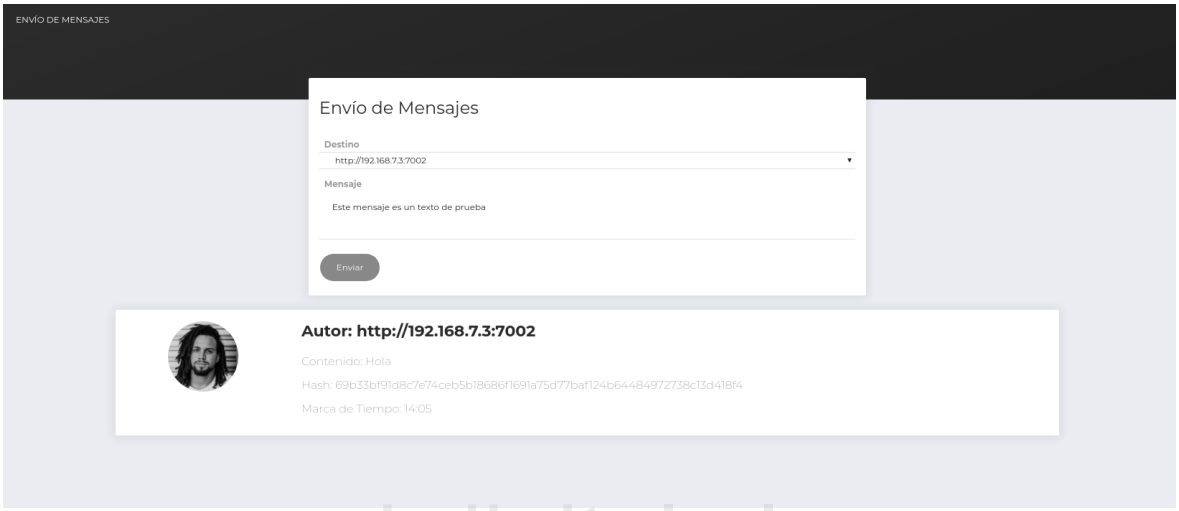
Figura 4.4: Diagrama de caso de uso para el envío de mensajes.

En este procedimiento, la red *blockdag* tiene un papel fundamental ya que es la que se encarga de obtener todos los participantes que se encuentran activos dentro de la red. De igual forma al usuario emisor enviar el mensaje, la red se encarga de crear la transacción, encriptarla y encapsularla en un bloque, así como de transmitir el mensaje creado a toda la red. Por parte del usuario receptor, la red se encargar de desencriptar la información en la que él se encuentra involucrado.

C.C. Reconocimiento

4.4.2. Historial de mensajes

En la vista de historial de mensajes se tienen todos aquellos mensajes que el usuario ha recibido. Como se puede ver en la Fig. 4.5 los campos se encuentran completamente visibles, es decir desencriptados.



www.bdigital.ula.ve

Figura 4.5: Vista de envío de mensajes e historial de mensajes.

Cualquier otro usuario que no se encuentre involucrado en la conversación, verá la información encriptada. Esto se debe a que la información se encuentra pública para cualquier participante de la red, pero tendrá campos encriptados en aquellos casos donde no se encuentre involucrado para brindar privacidad de los mensajes enviados.

En el caso en que un participante intente desencriptar la información no podrá ya que los datos se encuentran encriptados con la clave pública del usuario al que va dirigido y solo puede ser desencriptada con la clave privada del mismo.

C.C. Reconocimiento

Capítulo 5

Conclusiones y recomendaciones

5.1. Conclusiones

Al culminar el desarrollo se obtuvieron resultados favorables en la investigación, los cuales son resumidos a continuación.

www.bdigital.ula.ve

- Con los resultados obtenidos se logra cumplir el objetivo deseado de este trabajo: verificar que la inclusión de grafos acíclicos en una red *blockchain* es viable y presenta resultados satisfactorios.
- De igual forma, se pudo verificar en la Sección 3.2.7 que el algoritmo de consenso de transacciones como prueba de participación se acopla correctamente a la estructura de grafos acíclicos dirigidos.
- Por su parte, las pruebas de rendimiento realizadas a la red en la Sección 3.2.8, muestran cómo el algoritmo de transacciones como prueba de participación aporta velocidad a la misma sin perder seguridad en ciertos procesos.
- Así mismo, los resultados obtenidos de los experimentos en las Secciones 3.2.7 y 3.2.8 demuestran que, a pesar de que la red se ejecuta en una escala reducida para un ambiente académico, puede llegar a métricas de rendimiento similares a las que obtienen otras redes *blockchain*. Si bien es cierto que estos valores son satisfactorios, es necesario realizar pruebas exhaustivas en un ambiente real de

C.C. Reconocimiento

producción donde se corrobore su correcto funcionamiento bajo medidas de estrés reales.

- Se mejora la escalabilidad de los sistemas con esta nueva estructura, ya que al demostrar que es posible la inclusión de grafos acíclicos dirigidos se elimina el límite teórico al que la cadena de bloques esta sometida con las listas doblemente enlazadas y el consumo de memoria.

- De igual manera, se demuestra que esta nueva estructura *blockdag* es capaz de adaptarse a nuevos casos de uso como el envío de mensajes implementado en este trabajo. Esto abre muchas posibilidades a la hora de adaptar la cadena de bloques en nuevos enfoques en los cuales puede ser totalmente útil. Es decir, puede ser utilizado en cualquier actividad donde se requieran almacenar datos, se precise que el acceso a los datos sea compartido y las partes no se conozcan entre sí.

- Por otro lado, el algoritmo de consenso de participantes (ver Algoritmo 3.6), aporta un poco más de seguridad a la red sin afectar su funcionamiento. Esta es una medida de seguridad extra que puede ser tomada en cuenta en futuros desarrollos.

www.bdigital.ula.ve

5.2. Recomendaciones

Se recomienda realizar esfuerzos en mejorar la robustez del desarrollo con la inclusión de contratos inteligentes, así como implementar una arquitectura que permita un manejo de los datos de forma más óptima. Con estos aportes se obtendría un desarrollo más compacto y con una estructura más segura a la hora de ser aplicada en otro desarrollo. De igual forma, se puede evaluar la inclusión de un *token* dentro de la red para modificar el algoritmo de transacciones como prueba de participación con recompensas a los participantes testigos.

Así mismo, aplicar otras medidas de seguridad tradicionales al desarrollo, como límitar la red a que cada participante pueda estar conectado desde una sola computadora; estas medidas de seguridad sencillas pueden brindar mayor solidez en la seguridad de la red. Por otro lado, la inclusión de árboles de Merkle puede ser

C.C. Reconocimiento

satisfactorio al brindar rapidez en la búsqueda de *hashes* dentro de la red. Todas estas consideraciones tendrían como resultado una mejora considerable en la API de la red.

Para casos de uso donde se requiera el uso de un *token* o moneda virtual, se recomienda cambiar el algoritmo de consenso al algoritmo de prueba de participación delegada, explicado en la Sección 2.1.2.1. La razón de esta consideración es que DPOS ha demostrado ser un algoritmo muy beneficioso en redes con monedas virtuales.

www.bdigital.ula.ve

C.C. Reconocimiento

Bibliografía

Academy, B. (2019). ¿Qué es un ataque DoS? Recuperado de: https://www.binance.vision/es/security/what-is-a-dos-attack. Consultado el: 17-05-2019.

Bhanot, R. y Hans, R. (2015). A review and comparative analysis of various encryption algorithms. *International Journal of Security and Its Applications*, 9(4):289–306.

Blummer, B. (2018). An Introduction to Hyperledger. Reporte técnico, www.hyperledger.org. Recuperado de: https://www.hyperledger.org/wp-content/uploads/2018/08/HL_Whitepaper_IntroductiontoHyperledger.pdf. Consultado el: 12-04-2019.

Buterin, V. (2013). Ethereum white paper: a next generation smart contract & decentralized application platform. *ethereum.org*, 1:22–58. Ethereum.

Cámara, R. (2018). *Estudio de tecnologías Bitcoin y Blockchain*. Tesis de pregrado. Universitat Oberta de Catalunya (UOC). Barcelona, España.

Choi, S.-M., Park, J., Nguyen, Q., y Cronje, A. (2018a). Fantom: A scalable framework for asynchronous distributed systems. Reporte técnico, FANTOM. Recuperado de: https://arxiv.org/pdf/1810.10360.pdf. Consultado el: 10-04-2019.

Choi, S.-M., Park, J., Nguyen, Q., Cronje, A., Jang, K., Cheon, H., Han, Y.-S., y Ahn, B.-I. (2018b). OPERA: Reasoning about continuous common knowledge in asynchronous distributed systems. Reporte técnico, FANTOM Lab, FANTOM Foundation. Recuperado de: https://arxiv.org/pdf/1810.02186.pdf. Consultado el: 23-04-2019.

www.bdigital.ula.ve

C.C. Reconocimiento

Churyumov, A. (2016). Byteball: A decentralized system for storage and transfer of value. Reporte técnico, Byteball. Recuperado de: https://obyte.org/Byteball.pdf. Consultado el: 24-06-2019.

Dacak, C. (2015). Firma digital. Recuperado de: https://docplayer.es/12045256-Firma-digital-claudia-dacak-direccion-de-firma-digital-direccion-general-de-firma-digital-y-comercio-electronico.html. Consultado el: 25-09-2019.

Es-Samaali, H., Outchakoucht, A., y Leroy, J. P. (2017). A blockchain-based access control for big data. *International Journal of Computer Networks and Communications Security*, 5(7):137–147.

Farell, R. (2015). An analysis of the cryptocurrency industry. *Wharton Research Scholars*. Recuperado de: https://repository.upenn.edu/cgi/viewcontent.cgi?article=1133&context=wharton_research_scholars. Consultado el: 05-07-2019.

Foundation, F. (2018). Fantom: Whitepaper. Reporte técnico, FANTOM. Recuperado de: https://fantom.foundation. Consultado el 01-04-2019.

Garay, J., Kiayias, A., y Leonardos, N. (2015). The bitcoin backbone protocol: Analysis and applications. En *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, p. 281–310. Springer.

Greenough, Jhon (2015). The Internet of everything:2015. Recuperado de: https://www.businessinsider.com/internet-of-everything-2015-bi-2014-12. Consultado el: 10-06-2019.

Kusmierz, B. (2017). The first glance at the simulation of the Tangle: discrete model. Reporte técnico, http:/iota.org. Recuperado de: https://assets.ctfassets.net/r1dr6vzfxhev/2ZO5XxwehymSMsgusUE6YG/f15f4571500a64b7741963df5312c7e7/The_First_Glance_of_the_Simulation_Tangle_-_Discrete_Model_v0.1.pdf. Consultado el 05-06-2019.

www.bdigital.ula.ve

C.C. Reconocimiento

Larimer, D. (2018a). Delegated proof-of-stake consensus. Reporte técnico, Bitshares. Recuperado de: https://bitshares.org/technology/delegated-proof-of-stake-consensus/. Consultado el: 16-07-2019.

Larimer, D. (2018b). ¿Qué es DPoS? Recuperado de: https://academy.bit2me.com/que-es-dpos/. Consultado el: 10-07-2019.

Lizama, L., Montiel-Arrieta, L., Hernández-Mendoza, S., Flor, Lizama-Servín, L., y Simancas-Acevedo, E. (2019). Firma electrónica por medio de funciones hash para dispositivos móviles. *Ingeniería Investigación y tecnología*, 20(2):1–10.

Mahesh, S. (2018). Hyperledger Blockchain Performance Metrics. Reporte técnico, www.hyperledger.org. Recuperado de: https://www.hyperledger.org/wp-content/uploads/2018/10/HL_Whitepaper_Metrics_PDF_V1.01.pdf. Consultado el: 16-04-2019.

Maldonado, J. (2018). Recuperado de: https://www.criptotendencias.com/base-de-conocimiento/los-5-lenguajes-de-programacion-mas-usados-en-proyectos-blockchain/. Consultado el: 10-06-2019.

Mashrufee, A., Jahan, I., Rozario, L., y Jerin, I. (2016). A Comparative Study of RSA and ECC and Implementation of ECC on Embedded Systems. *International Journal of Innovative Research in Advanced Engineering*, 3(3):86–93.

Nagpal, A. y Gabrani, G. (2019). Python for Data Analytics, Scientific and Technical Applications. En *2019 Amity International Conference on Artificial Intelligence (AICAI)*, p. 140–145. IEEE.

Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Reporte técnico, www.bitcoin.org. Recuperado de: https://bitcoin.org/bitcoin.pdf. Consultado el: 01-03-2019.

Paredes, G. G. (2006). Introducción a la Criptografía. *Revista digital universitaria*, 7(7):1–17.

Popov, S. (2018). The tangle. *cit. on*, 1.4.3:131.

www.bdigital.ula.ve

C.C. Reconocimiento

Pressman, R. S. (2005). *Software engineering: a practitioner's approach*. Palgrave Macmillan, 8 edición.

Rennock, M., Cohn, A., y Butcher, J. (2018). Blockchain technology. *The Journal*, 1:35–43. Recuperado de: https://www.steptoe.com/images/content/1/7/v3/171269/LIT-FebMar18-Feature-Blockchain.pdf.

Retamal, C. D., Roig, J. B., y Tapia, J. L. M. (2017). La blockchain: fundamentos, aplicaciones y relación con otras tecnologías disruptivas. *Economía industrial*, 1(405):33–40.

Roig, N. P. y Montero, M. P. C. (2018). Tecnología blockchain: funcionamiento, aplicaciones y retos jurídicos relacionados. *Actualidad jurídica Uría Menéndez*, 1(48):24–36.

Rüegg, U., Ehlers, T., Spönemann, M., y von Hanxleden, R. (2016). A generalization of the directed graph layering problem. En *International Symposium on Graph Drawing and Network Visualization*, p. 196–208. Springer.

www.bdigital.ula.ve

Sommerville, I. (2005). *Ingeniería del software*. Pearson educación, Madrid, España, 7 edición.

Stephen, OÑea (2019). La competencia en curso de las *blockchain* por las transacciones por segundo. Recuperado de: https://es.cointelegraph.com/news/who-scales-it-best-inside-blockchains-ongoing-transactions-per-second-race. Consultado el: 05-05-2019.

Vileriño, S. (2017). *Estudio de los límites de generación de bloques en blockchain*. Tesis de Doctorado, Universidad de Buenos Aires, Argentina.

C.C. Reconocimiento