Universidad de Buenos Aires

Facultad de Ciencias Exactas y Naturales

Departamento de Computación

Estudio de los límites de generación de bloques en blockchain

Tesis presentada para optar al título de
Licenciado en Ciencias de la Computación

Silvio Vileriño

Director: David Alejandro González Márquez

Codirector: Maximiliano Iván Geier

Buenos Aires, 2017

Resumen

Desde la aparición de Bitcoin en 2008, las criptomonedas evolucionaron rápidamente. Éstas permiten realizar transacciones de forma segura, sin depender de una tercera parte de confianza. Sin embargo, todavía no existen implementaciones que logren competir a gran escala con los sistemas existentes dados los altos tiempos de procesamiento y validación de las transacciones en la red.

Otras criptomonedas tales como *RSK* y *Ethereum* proponen reducir considerablemente el tiempo de aparición de bloques junto con cambios en los protocolos de consenso e intercambio de mensajes para mitigar los problemas de rendimiento mencionados anteriormente.

En este trabajo presentamos una metodología que permite analizar diversos aspectos de las redes blockchain. La misma permite obtener métricas acerca de la red en base a trazas de ejecución obtenidas utilizando un cliente de criptomoneda instrumentalizado en una red emulada.

Se diseñó un modelo de minado simulado que no consume recursos y se validó que su comportamiento es similar al algoritmo de minado real. Esto nos permitió simular escenarios complejos utilizando escasos recursos de cómputo.

Pudo corroborarse la existencia de una relación entre el tiempo entre bloques y la cantidad de forks que ocurren en la red. Así también se relacionó el diámetro de la red con el tiempo total de propagación de un bloque y su impacto en la cantidad de bloques *stale*.

Finalmente, el análisis sobre el algoritmo de propagación de bloques de Ethereum muestra

que la efectividad de heurísticas de envío de hashes depende del diámetro de la topología y de las latencias de la red subyacente.

Abstract

Since Bitcoin was born in 2008, cryptocurrencies have quickly evolved in the past decade. These cryptocurrencies allow secure transactions without relying on a trusted third party. However, their implementations still can't compete with existing payment systems due to the high processing and validation times.

Other platforms such as RSK and Ethereum propose a huge decrease in the target as well as changes in the consensus and propagation protocols in order to mitigate the problems described above.

In this work we introduce a novel methodology for analyzing a broad range of aspects on blockchain networks using emulated networks. This can be achieved by using an instrumentalized version of a real cryptocurrency client on a private network.

We designed a simulated mining model that doesn't use CPU resources and verified its behaviour is similar to the real mining algorithm. This model allowed us to recreate complex scenarios without requering massive computing power.

We have been able to establish a relationship between the target and the occurrence of forks in the network. Also, we found that increasing the network diameter directly affects the propagation time of a block to the entire network, increasing the amount of stale blocks due to a greater number of forks.

Finally, we present an analysis on Ethereum's block propagation algorithm. The results of this experiments showed that the hash propagation heuristics effectiveness varies depending

on the topology diameter and the latencies of the underlying network.

Al LICAR por darme un espacio para trabajar. A Esteban, David y Maxi por todas las tardes de café y manija.

A mis directores nuevamente, que me enseñaron como escribir un trabajo científico y se tomaron el trabajo de hacer un seguimiento de toda la tesis.

A la UBA y al DC por brindarme esta formación de excelencia en Ciencias de la Computación.

A mi familia por haberme acompañado durante toda la carrera y permitido acceder a estos estudios.

A mis amigos, algunos que incontables veces deje de ver por tener que estudiar, otros que me acompañaron durante estos 6 años y medio en esta universidad.

A mi abuela Alba por apoyarme siempre en todos mis proyectos.

Índice general

Capítulo 1

Introducción

Durante el paso de la última década tomaron impulso las criptomonedas, comenzando por *Bitcoin* y luego abriéndose paso diversas alternativas más novedosas que incluyen funcionalidades más avanzadas como *smart contracts* y mejoras en el rendimiento y la escalabilidad.

La principal novedad de estas criptomonedas es que permiten realizar pagos de forma segura sin la necesidad de una *tercera parte de confianza (trusted third-party)*. Sin embargo, las principales implementaciones tienen limitaciones para ser utilizadas a gran escala dados los altos tiempos de procesamiento y validación de las transacciones en la red.

Dentro de las alternativas que intentan salvar estos problemas de escalabilidad, se encuentran plataformas como RSK y Ethereum, que proponen reducir considerablemente el tiempo de aparición de bloques junto con cambios en los protocolos de consenso e intercambio de mensajes.

En este trabajo presentamos una metodología que permite analizar diversos aspectos de las redes blockchain.

La estructura del documento se explica a continuación: En los capítulos 2 y 3, introducimos los conceptos necesarios para comprender este trabajo. El capítulo 4 recopila información

acerca de trabajos similares ya existentes en la comunidad. Dedicamos el capítulo 5 a la explicación de la metodología utilizada en esta tesis. En el resto del trabajo se encuentran resultados acerca de diversos experimentos. Los primeros, presentados en el capítulo 6, están dedicados a la validación de nuestra metodología. En el capítulo 7 se encuentran los resultados sobre el análisis de redes blockchain utilizando la metodología propuesta.

Finalmente, en los capítulos 8 y 9 proponemos algunas ideas como trabajo futuro y exponemos nuestras conclusiones.

Para lectores familiarizados con los conceptos utilizados en este trabajo, sugerimos comenzar por el capítulo 5. Luego pueden leerse independientemente los capítulos 6 y 7.

Capítulo 2

Redes definidas por software

En este capítulo sentamos las bases teóricas necesarias para entender las redes definidas por software. Su uso está motivado por la necesidad de emular grandes entornos de producción, en entornos más pequeños. Por ejemplo, una red productiva con hosts en varios países, se puede modelar a escala utilizando una topología emulada donde la latencia en los enlaces de red representen la distancia entre hosts. Sobre este último modelo emulado, podemos extraer conclusiones acerca del comportamiento de la red productiva.

2.1 Frameworks de emulación de redes

A continuación presentaremos dos herramientas disponibles para crear redes definidas por software. Ambas utilizan las mismas técnicas para emular los componentes de una topología, pero se diferencian en los límites de escalabilidad. Finalmente, hablaremos de otros límites que tienen estas herramientas para emular determinados entornos.

2.1.1 Mininet

La aplicación «Mininet», creada por Lantz, Hellen y McKeown en [LHM10] permite emular una red dentro de una sola computadora. Esta aplicación permite definir una red utilizando hosts, switches y enlaces que los conecten.

Mininet utiliza *«namespaces»*[1] de Linux para aislar los diferentes elementos de la red. Dentro de estos namespaces, es posible virtualizar diferentes recursos del sistema operativo para conjuntos de procesos. Por ejemplo, podrían existir procesos que se encuentren en el mismo espacio de procesos[2] y que compartan el sistema de archivos, pero tengan diferentes interfaces de red virtuales e independientes entre sí.

En particular, Mininet crea un namespace de red diferente tanto para cada host como para cada switch, los cuales tienen su propia instancia del stack de red. Al agregarse un enlace para conectar dos elementos de la red, se instancian nuevas interfaces en los namespaces de los extremos del nuevo enlace. Cada enlace es configurable, pudiéndose modificar distintas propiedades tales como latencia, ancho de banda, *jitter*[3], porcentaje de pérdida de paquetes, etc. L a configuración de los enlaces es aplicada utilizando el subsistema de control de tráfico de Linux (`netem`) introducida en el trabajo de Hemminger et al. [H+05].

Luego de definir la topología de red, se debe tener en cuenta cómo se maneja el tráfico entre los elementos de dicha red. Cada host o switch debe saber qué hacer cuando le llega un paquete, es decir, cómo redireccionar el tráfico que pasa por él. En Mininet, esta tarea es delegada a un componente llamado *«Controlador»* de la red. El mismo es el encargado de procesar los paquetes para los cuales los switches no tengan ya un flujo de tráfico definido. En este trabajo utilizamos el controlador *Floodlight* [flo].

Si bien pueden ejecutarse aplicaciones nativas sin ningún tipo de virtualización, toda la emulación se ejecuta en una única máquina física. Esta última característica, limita a Mininet en la máxima escala que puede soportar utilizando hardware convencional.

El equipo de Mininet comenzó a trabajar en una alternativa llamada *«Mininet Cluster»* con el objetivo de eliminar esta limitación. Al momento de realizar este trabajo, el proyecto se encuentra muy inestable por lo que no se considero como una opción, pero en futuras ampliaciones de la herramienta seria posible utilizarlo sin modificaciones a la infraestructura

[1] https://en.wikipedia.org/wiki/Linux_namespaces
[2] https://lwn.net/Articles/531419/
[3] https://es.wikipedia.org/wiki/Jitter

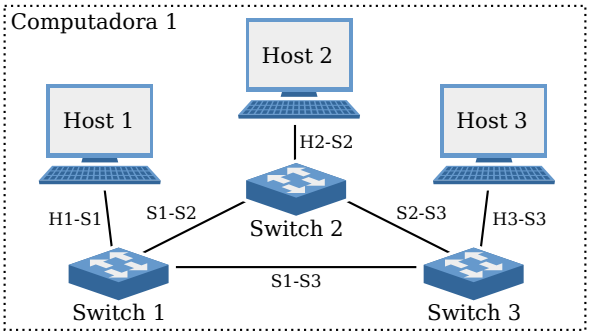


Figura 2.1: Ejemplo de Topología Mininet. En una misma máquina se ejecutan tres hosts virtuales, cada uno conectado a un switch. Entre cada par de componentes hay un enlace que los conecta respetando las propiedades configuradas.

propuesta.

2.1.2 Maxinet

En julio de 2014, surgió una alternativa a Mininet Cluster. Esta solución, llamada «Maxinet», fue creada por Wette et al. [WDS14] y permite distribuir una topología emulada de Mininet en un cluster de instancias de Mininet. Esta arquitectura permite utilizar *commodity hardware*[4] para la realización de experimentos de mayor tamaño, distribuyendo la carga de trabajo.

La idea detrás de Maxinet consiste en utilizar *METIS* [KK95] para particionar una topología de Mininet en varias topologías más pequeñas, que serán instanciadas en las distintas máquinas del cluster.

Cuando una topología se emula sobre Maxinet, lo que ocurre es que los hosts y switches quedan distribuidos entre varias máquinas físicas, conectándose mediante túneles *GRE*[5]. Un ejemplo de esta partición se puede verse en la figura 2.2.

2.2 Límites de la emulación en redes

Existen escenarios donde las herramientas que presentamos anteriormente tienen algunos problemas debido a cuellos de botella que ocurren en componentes lógicos de la arquitectura

[4] https://en.wikipedia.org/wiki/Commodity_computing
[5] https://es.wikipedia.org/wiki/GRE

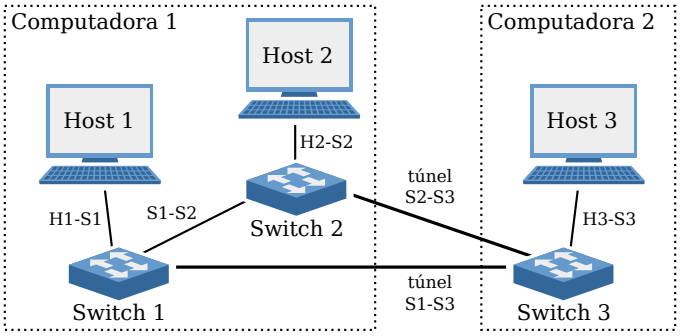


Figura 2.2: Ejemplo de Topología Maxinet. En una máquina se ejecutan 2 hosts y 2 switches, y en otra máquina son instanciados un host y un switch. Cada switch pertenece a la misma máquina que su host y están conectados mediante un enlace. Entre máquinas físicas, las conexiones entre switches son túneles GRE.

de la herramienta o debido a cuestiones del hardware donde se realiza la emulación. Esto limita las topologías que se pueden emular. A continuación presentamos dos ejemplos de esta problemática.

2.2.1 Controlador de red

En ambas herramientas mencionadas anteriormente existe una única instancia de este componente en la arquitectura de la emulación[6]. Esto provoca que, para cualquier topología lo suficientemente compleja, este componente actúe como un cuello de botella.

2.2.2 Emulación de enlaces de baja latencia

Existe una limitación en la configuración de latencias en los enlaces virtuales. Si dichos enlaces son configurados con latencias cercanas a las de la red física subyacente, se observa una varianza en los tiempos de respuesta de la topología virtual mucho mayor a la registrada en la red física.

Al existir diferencias tan pequeñas entre las latencias físicas y las virtuales, las conexiones físicas de los túneles GRE entre las distintas máquinas que participan de la emulación tienen una incidencia en la latencia emulada, provocando este efecto.

[6]Maxinet no distribuye el trabajo del controlador sino que solo particiona los componentes de la topología.

Capítulo 3

Criptomonedas y Blockchains

Este capítulo sienta las bases teóricas sobre Criptomonedas y Blockchains, necesarias para comprender este trabajo.

Comenzaremos con una breve descripción acerca de las criptomonedas. Nos basaremos en Bitcoin como criptomoneda de referencia para introducir algunos conceptos básicos, ya que otras criptomonedas expanden sobre estos últimos.

Estos conceptos se pueden clasificar en tres grupos: *Estructuras de datos*, *procesos* y *protocolo*. El primero se refiere a las definiciones de *transacciones*, *bloques* y *blockchain*. El segundo describe los procesos de *minado* y *consenso*. Finalmente, el *intercambio de mensajes* y los *algoritmos de propagación* son detallados al explicar el protocolo.

Luego, presentaremos Ethereum, una criptomoneda que mejora ciertos aspectos de Bitcoin. En este caso, solo explicaremos las diferencias en los procesos y el protocolo ya que las estructuras fundamentales no varían significativamente respecto a Bitcoin.

3.1 Introducción a las criptomonedas

Con el nacimiento de Bitcoin, las criptomonedas tomaron impulso en el mundo de la computación. 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.

Hay un gran interés por parte de la comunidad del mundo de las criptomonedas por tener un sistema competitivo contra sistemas de pago online como Visa o Paypal. Estos últimos pueden procesar gran cantidad de transacciones por segundo, con tiempos de confirmación casi instantáneos. Para lograr esto en criptomonedas como Bitcoin es necesario aumentar la cantidad de transacciones procesadas por segundo y minimizar el tiempo requerido para obtener garantías acerca de la persistencia efectiva de una transacción en la Blockchain.

3.2 Bitcoin

Bitcoin fue propuesta en el 2008 por una persona bajo el seudónimo «Satoshi Nakamoto» [Nak08]. Fue la primer criptomoneda descentralizada que resolvía el problema de *double spending* sin necesidad de confiar en una tercera parte de confianza.

Una transferencia en una moneda digital se puede equiparar al endosado de un cheque. El propietario de un cheque puede escribir en el dorso del mismo el nombre del destinatario del dinero, y este a su vez puede endorsarlo nuevamente. En este proceso, es posible conocer quienes fueron todos los intermediarios en la cadena, desde el dueño del cheque hasta el último en endosarlo.

En el mundo electrónico, se pueden obtener garantías similares utilizando firmas digitales y hashes criptográficos: cuando una persona quiere transferir dinero digital a otra se crea una transacción, que no es más que 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 figura 3.1 se presenta una esquematización de este proceso, en donde se realizan tres transacciones de una criptomoneda. Se puede ver que el dueño de la primer clave privada

firma el hash que hace referencia a la transacción anterior y a la clave pública del nuevo destinatario, creando una nueva transacción. Este proceso se repite una vez más entre el nuevo destinatario y una tercera persona.

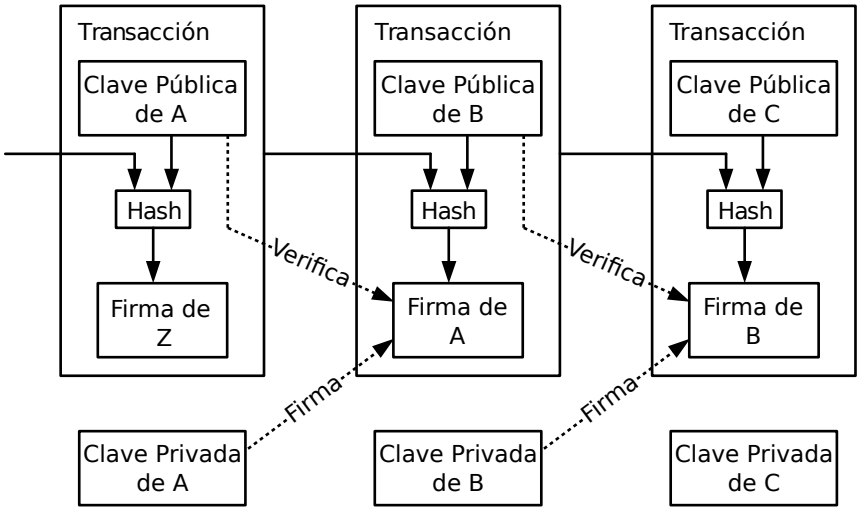


Figura 3.1: Esquema de transacciones en una criptomoneda. Fuente: [Nak08]

Existe un problema conocido como *doble gasto (double spending)* en donde el dueño de una transacción puede transferir **el mismo dinero** a dos personas distintas, y esta transacción sería valida ya que tendría la firma del dueño en ambas. Una solución a este problema podría ser contar con una entidad de confianza que se encargue de validar todas las transacciones.

Bitcoin propone una solución diferente al problema de double spending, sin necesidad de confiar en una tercera parte. La idea principal es la introducción de una estructura denominada **Blockchain**: Un registro distribuido y replicado de todas las transacciones que fueron procesadas. Cualquier persona puede registrar nuevas transacciones, las cuales son distribuidas a toda la red.

En la figura 3.2 se puede apreciar una esquematización de una Blockchain.

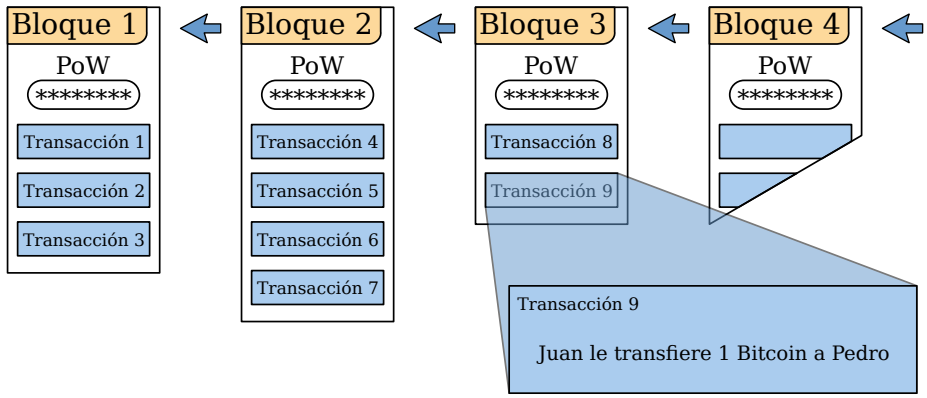


Figura 3.2: Esquema de blockchain con transacciones.

3.2.1 Estructuras de datos

Transacciones

En Bitcoin, una transacción está definida como una lista de *inputs* y *outputs*. Un output consiste en un par de $(cantidad, hash)$ donde el primer valor es la cantidad de Bitcoins que el destinatario va a recibir y el segundo es el hash criptográfico de la clave pública del destinatario.

La lista de inputs $(hash, índice, clave, firma)$ hace referencia a outputs de transacciones anteriores. Un input contiene:

- El hash de la transacción a la que hace referencia.
- El índice del output deseado dentro de la transacción.
- La clave pública correspondiente a dicho output.
- Una firma digital del hash de la transacción usando la clave privada correspondiente a la clave pública del input.

Para verificar la validez de las referencias a outputs de transacciones anteriores, se calcula el hash de la clave pública y se verifica que sea igual al que figura en el output referenciado. Luego basta con verificar la firma digital con esa clave pública.

En la figura 3.3 se puede ver un ejemplo de una transacción con varios inputs. Del lado

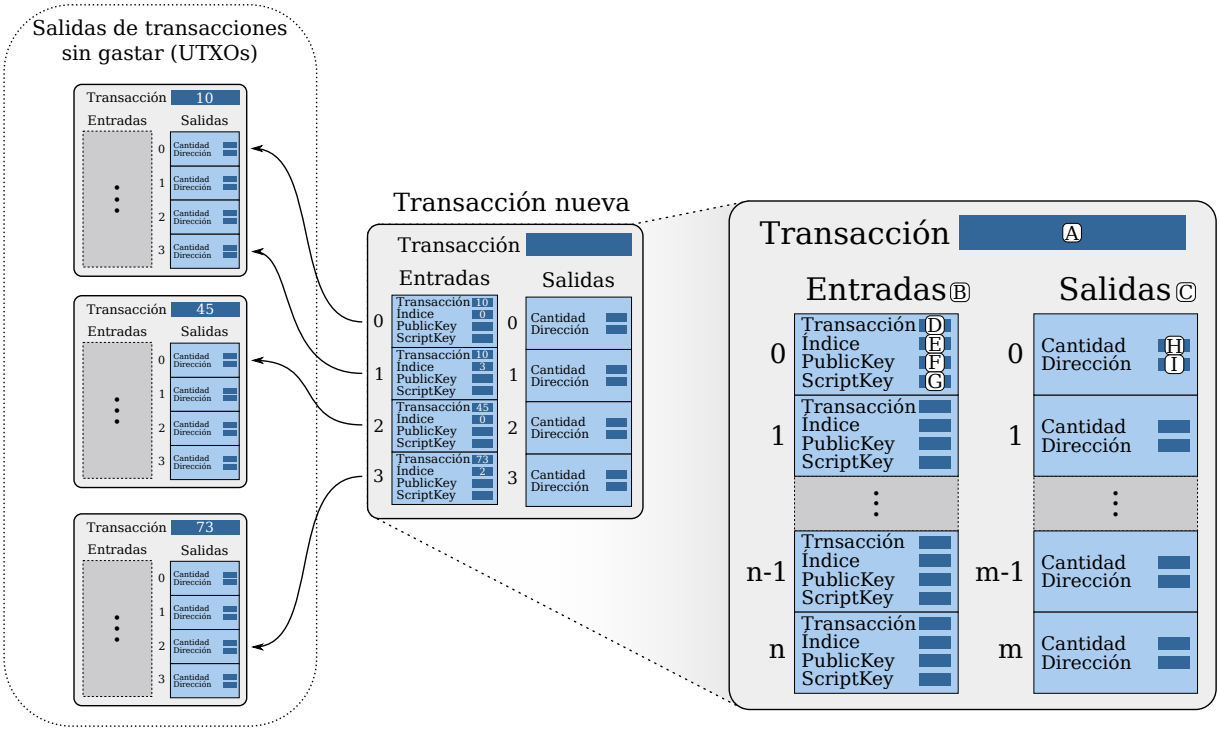


Figura 3.3: Ejemplo de una transacción en Bitcoin

derecho, se destacan los siguientes campos:

(A) El hash que identifica de forma unívoca a la transacción.

(B) La lista de inputs de la transacción, cuyos elementos hacen referencia a outputs de alguna transacción anterior que todavía no fueron utilizados.

(C) La lista de outputs de la transacción.

(D) El hash de la transacción a la que ese input hace referencia.

(E) El índice del output dentro de la transacción referenciada.

(F) La clave pública que se corresponde con la *address* del output.

(G) Una firma digital de la clave pública realizada con la clave privada correspondiente. Esto demuestra que se tiene control sobre dicho output.

(H) La cantidad de Bitcoins a transferir a ese output.

(I) La dirección destino, indicada por el hash de la clave pública del destinatario.

Para considerar a una transacción como válida debemos verificar que todos los inputs sean válidos y que la suma de los outputs referenciados por éstos sea mayor o igual que la suma de los outputs de la transacción. De existir un excedente entre los inputs y los outputs, éste dinero extra es denominado «transaction fee» y podrá ser reclamado por quien agregue la transacción a la Blockchain.

Los outputs de una transacción pueden referenciarse una única vez. Es decir, sólo se pueden considerar los outputs de transacciones que no fueron utilizados previamente. Estos son conocidos como «Unspent Transaction Outputs» (UTXOs).

Para que una transacción válida sea agregada a la Blockchain, ésta debe formar parte de un *Bloque*. Los bloques agrupan transacciones y son la unidad básica de la Blockchain.

Bloques

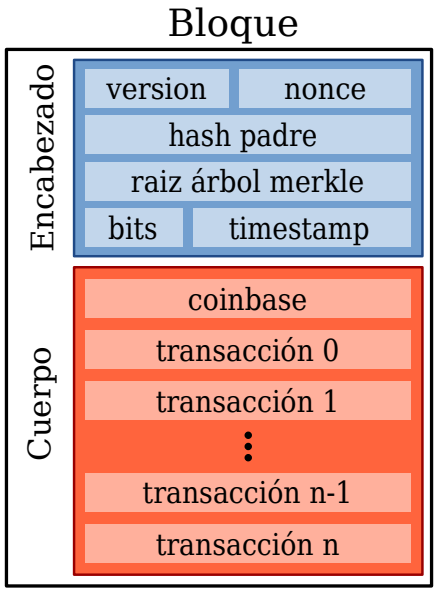


Figura 3.4: Ejemplo de bloque

Cada bloque está compuesto por:

- El *hash* del bloque anterior (su padre).
- Tiempo en segundos en que fue minado, desde *epoch*[1].

[1] 1 de Enero de 1970, a las 00:00:00 UTC

- La dificultad con la que ese bloque fue hallado.
- *nonce*: La prueba de trabajo, que demuestra que el bloque cumple con la dificultad que dice tener.
- Un conjunto de transacciones ordenadas.
- Un número de versión.

Los bloques se separan en *header* y body. En Bitcoin, el header contiene todos los campos mencionados anteriormente, con excepción de las transacciones, que se encuentran en el body. El header cuenta con el hash de un *Merkle Tree* que contiene las transacciones.

Un Merkle Tree es un árbol binario de hashes criptográficos, en donde cada nodo contiene el hash de la concatenación de sus dos hijos. En la figura 3.5 se puede ver cómo se forma la raíz de este árbol de hashes, usando todas las transacciones que se encuentran en el body del bloque.

Estos bloques son almacenados en una estructura llamada Blockchain, que explicaremos a continuación.

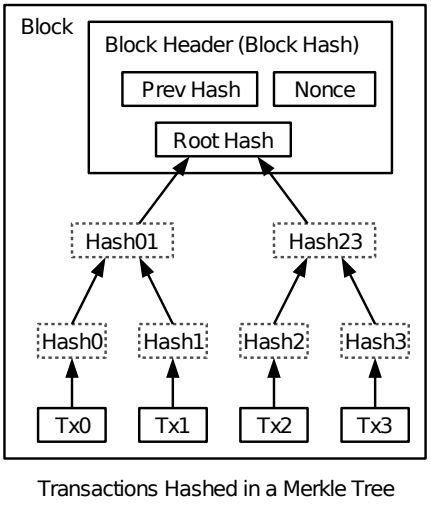


Figura 3.5: Ejemplo de Bloque con Merkle Tree. Fuente: [Nak08]

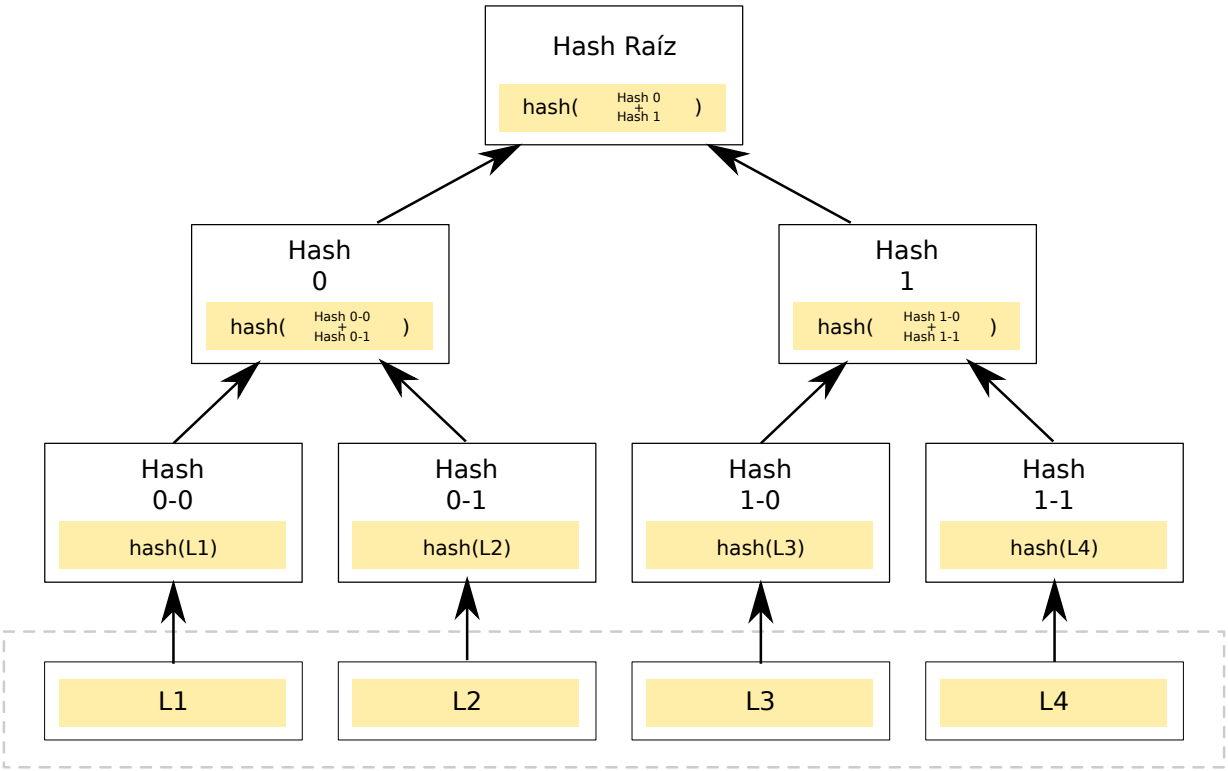


Figura 3.6: Ejemplo de Merkle Tree

Blockchain

La estructura de la Blockchain puede considerarse similar a un árbol cuyos nodos son Bloques. Definimos una cadena dentro de esta estructura, como un camino de bloques que comienza en la raíz y termina en alguna hoja de la Blockchain. Dentro del conjunto de estos caminos, existe una cadena distinguida llamada *main chain* o *cadena principal* cuya dificultad total, definida como la suma de dificultades de todos los bloques que la componen, es máxima. Los bloques que no pertenecen a dicha cadena distinguida son llamados bloques *«stale»*.

La raíz de la Blockchain es un bloque especial llamado *Bloque Génesis*. En Bitcoin, este bloque fue minado el 3 de enero de 2009, y además, en uno de sus campos contenía un titular del diario *«The Times»* de esa fecha[2], para dar prueba de que este bloque no fue creado con antelación a ese día.

[2] «The Times 03/Jan/2009 Chancellor on brink of second bailout for banks»

La altura de un bloque dentro de la Blockchain queda determinada por su número. Si un bloque contiene transacciones cuyos inputs hacen referencia a outputs que ya fueron gastados en un bloque de menor número dentro de su cadena, decimos que éste es inválido. Esta verificación es lo que permite evitar double spending dentro de una cadena.

Cualquier nodo puede construir bloques y propagarlos en la red. Estos últimos se denominan *nodos mineros*. Ante la llegada de un bloque a un nodo, éste lo valida, lo agrega a la Blockchain y lo retransmite a sus nodos vecinos. Para que un bloque sea válido y aceptado por los nodos de la red, debe contener una prueba de trabajo (PoW), que valide que el nodo minero gastó tiempo de CPU para poder construir el bloque.

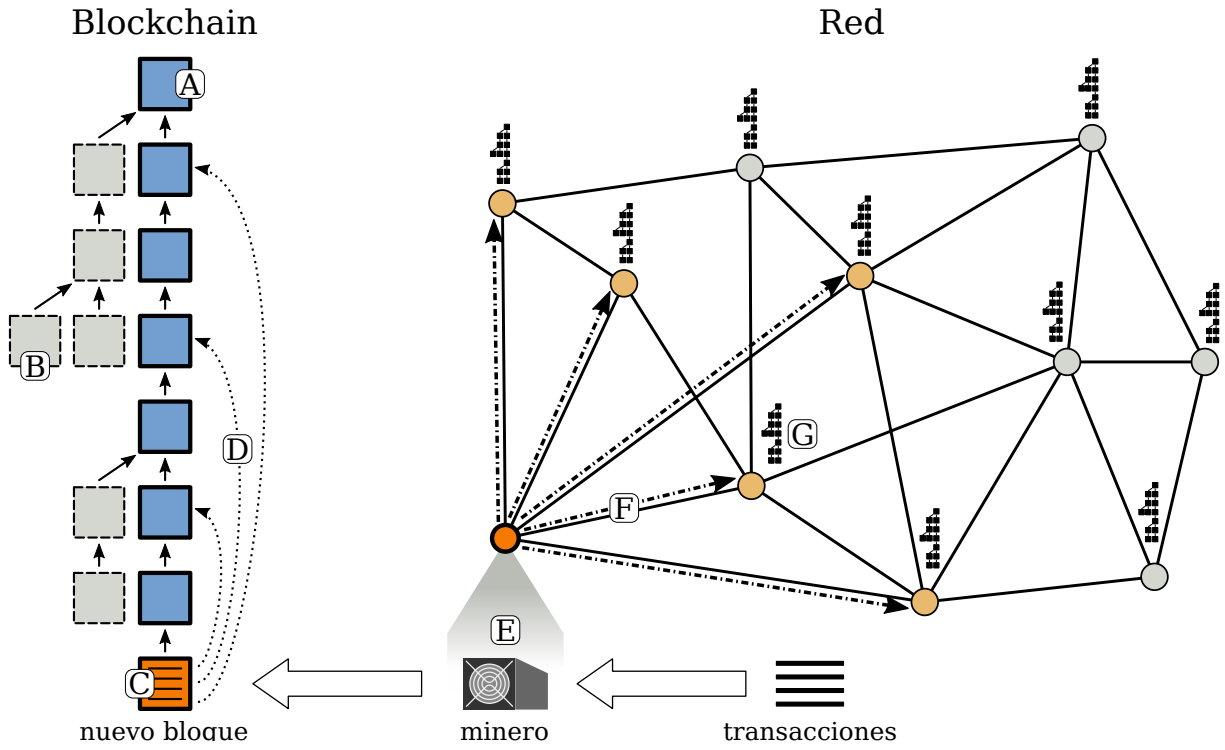


Figura 3.7: Vista general del funcionamiento de Bitcoin

En la figura 3.7 se puede ver el funcionamiento de la red. A continuación se detallan las componentes exhibidas en la figura:

(A) Cadena principal

(B) Bloque stale

(C) Último bloque de la cadena principal

(D) Referencia a una transacción anterior.

(E) Nodo minero

(F) Propagación de un bloque

(G) Blockchain propia del nodo

3.2.2 Procesos de minado y consenso

En Bitcoin los nodos mineros tienen el incentivo de una recompensa en Bitcoins si su bloque forma parte de la cadena principal de la Blockchain. Para crear nuevos bloques éstos hacen uso de su poder de cómputo para calcular el campo *nonce*.

Dicho campo contiene la Proof of Work, un número tal que el resultado de la evaluación de **SHA256(SHA256(header))**[3] contiene una cantidad determinada de 0's en sus bits más significativos. Esta cantidad es mayor o igual a un número que se deriva del campo *dificultad* del header del bloque.[4]

La motivación de este proceso es que obtener un hash que cumpla la propiedad pedida es computacionalmente costoso[5]. Los hashes calculados están uniformemente distribuidos, por lo cual la probabilidad de que un hash (256 bits) comience con d ceros, es $\frac{2^{256-d}-1}{2^{256}}$. El numerador de éste número surge del cálculo combinatorio de dejar fijos los primeros d digitos y variar aleatoriamente el resto de los bits. Al realizar el cálculo de casos favorables sobre casos totales se llega a la fracción expuesta.[6] Mientras mayor poder de cómputo posea el nodo minero, podrá probar más cantidad de nonces por unidad de tiempo aumentando sus posibilidades de encontrar uno válido.

[3] *SHA256* es una función de hash criptográfico.

[4] El campo dificultad almacena la representación de un número de 256 bits que es comparada contra el hash del bloque. Es una simplificación considerar la cantidad inicial de ceros.

[5] Considerar que al cambiar un bit del input, el hash calculado cambia totalmente.

[6] Hay que descontar el caso donde comienza con d ceros, y aleatoriamente el dígito $(d+1)$-ésimo tambien es un cero.

Para verificar que un bloque cumple con esta propiedad, alcanza con calcular dos veces el SHA256 del header, mientras que la dificultad de armar un bloque que sea válido crece exponencialmente con la cantidad de 0's requerida.[7] De esta forma, se cuenta con un proceso que demuestra que un minero realizó una cantidad determinada de trabajo para hallar un bloque válido.

Es importante destacar que una PoW solo sirve para un bloque, ya que esta depende del contenido del mismo. Si un bloque cambia cualquiera de sus campos (padre, transacciones, fecha), la PoW deja de ser válida. Además, como cada bloque tiene el hash del bloque padre, no es posible calcular la PoW con anticipación.

Debido a que el poder de cómputo de la red puede fluctuar mucho con el ingreso y egreso de nodos, Bitcoin incorpora un sistema de *ajuste de dificultad*, que aumenta o disminuye la dificultad que deben cumplir los bloques para ser válidos. Para esto, la red define un *target*, que es el tiempo promedio en que deben aparecer nuevos bloques en la red (en Bitcoin es 10 minutos). Cada 2016 bloques se cuenta cuántos de esos bloques[8] se construyeron en menos tiempo que el target, y en función de eso se aumenta o disminuye la dificultad hasta cuatro veces el valor anterior.

Bitcoin provee ciertos incentivos para que los mineros colaboren con la red. Por un lado, el minero que construye un bloque se queda con todos los fees de las transacciones de ese bloque. Por otro lado, Bitcoin recompensa a los mineros, permitiéndoles ser los beneficiarios de la transacción *coinbase* del bloque, una transacción especial que emite nuevos Bitcoins que antes no existían.

Esta es la única forma de crear Bitcoins, y es por eso que se conoce al proceso como «Minar Bitcoins».

La transacción coinbase debe ser la primera transacción del bloque. Además, el minero tiene control total sobre los campos de esta transacción. El output de esta última sólo puede ser

[7] La cantidad de hashes que cumplen esta restricción, disminuye a la mitad por cada cero adicional requerido.

[8] Debido a un error de software, en Bitcoin solo se revisan los últimos 2015 bloques

usado luego de que haya 100 bloques por encima del bloque minado. Esta decisión busca evitar que se obtenga dinero a partir de bloques *stale*.

La cantidad de Bitcoins otorgada por minar bloques disminuye en función del número de bloque. Comenzando con un valor de 50 BTC por bloque, cada 210.000 bloques (4 años aproximadamente) se reduce a la mitad. Al 26 de mayo de 2017, la recompensa por hallar un bloque en Bitcoin es de 12,5 BTC (cerca de US$ 33.625).

Un nodo minero calcula pruebas de trabajo para hallar bloques cuyo padre sea el último elemento de la cadena principal de su Blockchain. Al encontrar un bloque válido, éste lo propaga a sus vecinos, quienes lo validan, agregan a su Blockchain y propagan nuevamente. Este proceso se repite hasta que el bloque es propagado a toda la red.

Dada la naturaleza distribuida de Bitcoin, es posible que más de un minero encuentre un bloque válido al mismo tiempo y lo propague al resto de la red. Si dichos bloques se encuentran a la misma altura en la Blockchain, decimos que son *competitivos*.

En esta situación, los nodos de la red que reciben un bloque competitivo respecto al bloque bajo el cual están minando, tienen que decidir si su cadena principal sigue siendo válida o si necesitan restablecerla, pues podría ocurrir que el bloque competitivo pertenezca al final de una cadena previamente stale, cuya nueva dificultad total sea mayor a la de la cadena principal.

Este mecanismo se conoce como *Protocolo de Consenso*. En Bitcoin, ante la presencia de dos ramas competitivas, gana la que tiene mayor dificultad total, y ante un empate, cada nodo se queda con la que recibió primero.

Estos procesos competitivos generan una partición en los nodos de la red, como se puede observar en la figura 3.8, donde un grupo de nodos tiene una cadena principal, y otro grupo de nodos otra.

Cada grupo minará bajo la que considere su cadena principal. A esta situación se la conoce como un *fork* de la red. Eventualmente, una de las dos cadenas tendrá una dificultad

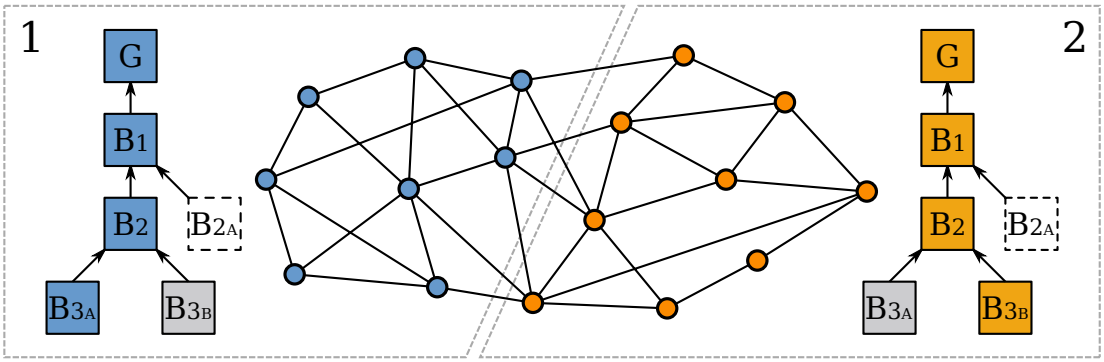


Figura 3.8: Ejemplo de una partición de los nodos de la red. El grupo de nodos 1 considera al bloque B_{3A} como bloque de su cadena principal, mientras que el grupo 2 considera a B_{3B}. El bloque B_{2A} quedó *stale* luego de que se minaron los bloques B_{3A} y B_{3B}.

total mayor al contar con nuevos bloques. Cuando esto ocurra, los nodos establecerán sus cadenas principales a la cadena con mayor dificultad total.

La probabilidad de que un bloque pertenezca a la cadena principal es proporcional a la dificultad máxima de las cadenas del subárbol minado bajo dicho bloque.

En la red Bitcoin, la divergencia más grande que ocurrió fue de cuatro bloques (sin contar los casos que se debieron a fallas de software). Se recomienda que para estar seguros de que el bloque va a pertenecer a la cadena principal, se espere a que se minen por lo menos seis bloques debajo de él (una hora considerando un target de diez minutos).

Esto indica que para tener cierta garantía de no ser víctimas de un ataque de double spending, simplemente se debe esperar a que haya más bloques minados bajo el bloque en donde aparece la transacción de interés. Si un atacante quisiese revertir dicha transacción, debería generar una rama equivalente bajo el bloque padre del cual la contenga. Es decir, el atacante tiene que controlar cierto porcentaje del poder de cómputo de la red para producir una cadena de mayor dificultad que la creada por los nodos honestos. A este ataque se lo conoce como ataque del 51 %.

3.2.3 Protocolo

Para poder saber a qué nodos conectarse, el cliente de Bitcoin cuenta con un archivo que contiene una serie de *«boot nodes»* a los que un nodo se puede conectar para pedir

direcciones de otros nodos.

Los principales mensajes utilizados por los nodos de Bitcoin son: GetAddr, Addr, Inv, GetData, GetHeaders, GetBlock, Block, Headers, Tx, Version y VerAck.

Version y VerAck son los dos primeros mensajes intercambiados entre los nodos al conectarse, indicando entre otras cosas la versión del protocolo que están usando y datos acerca de sincronización de la Blockchain.

Un nodo puede pedirle a uno de sus pares una lista de conocidos para aumentar su cantidad de conexiones. Esto lo hace mediante el mensaje *GetAddr*, que pide una lista de nodos que se consideren activos, es decir, que hayan mandado un mensaje en las últimas tres horas. La respuesta a este mensaje es un mensaje *Addr* que contiene una lista de nodos activos elegidos al azar dentro de sus conocidos.

Cuando un nodo encuentra un bloque nuevo, este envía un mensaje de *Inv* a todos sus peers, informándoles el hash del bloque hallado. Estos le piden información del bloque mediante el mensaje *GetData*, y el nodo responde con un mensaje *Block*. Algo similar ocurre cuando un nodo recibe una transacción. La misma se propaga mediante un mensaje Inv, y al recibir un GetData, envía la transacción en un mensaje Tx.

Cuando un nodo nuevo ingresa a la red, este debe sincronizar toda su Blockchain, es decir, traer todos los bloques que le faltan. Para ello, puede pedir varios bloques mediante los mensajes *GetBlocks* y GetHeaders.

3.3 Ethereum

Ethereum es una criptomoneda que mejora ciertos aspectos en términos del rendimiento y permite la especificación de condiciones de pago complejas en las transacciones mediante el uso de contratos inteligentes (*Smart Contracts*). Las principales mejoras respecto a Bitcoin son cambios en el protocolo de red que permiten bajar el tiempo entre bloques de 10 minutos a 14 segundos, haciendo posible que se realicen más transacciones por unidad de tiempo.

Si bien Ethereum puede ejecutar transacciones más complejas que Bitcoin mediante Smart Contracts, su ejecución está acotada por el dinero total a disposición en las cuentas. Cada instrucción ejecutada, byte transferido o almacenado en memoria, tiene un costo, denominado *«gas»*. Cuando se crea una transacción, se define el valor de los campos *«GasPrice»*, y *«GasLimit»*. «GasLimit» determina cuánto puede ejecutar esa transacción como máximo. «GasPrice» hace referencia a la conversión entre *«Ether»*, la moneda de Ethereum, y el «gas» usado por la transacción.

Por otro lado, análogamente al concepto de direcciones en Bitcoin, Ethereum introduce la noción de cuentas, cuyo estado se actualiza luego de la ejecución de las transacciones de cada bloque.

3.3.1 Procesos de minado y consenso

Como prueba de trabajo, Ethereum utiliza hashes criptográficos *SHA3*, junto a otras técnicas que evitan que sea eficiente minar usando hardware dedicado. La consecuencia más directa del uso de una función de hash diferente es que el proceso de minado de Ethereum no es compatible con el de Bitcoin.

Al ser el tiempo de generación de bloques considerablemente más corto en Ethereum que en Bitcoin, la probabilidad de que se generen forks en la red es mucho mayor, provocando que gran parte del poder de cómputo de la red sea desperdiciando en bloques stale. Es por esto que Ethereum adopta una versión simplificada[9] del protocolo de consenso *GHOST*, propuesto en el trabajo de Sapirshtein et al. [SZ13].

También se introduce la noción de **uncles**. Los mineros, al armar un bloque, pueden referenciar hasta dos bloques *stale* que serán sus uncles. Por cada referencia, el minero del bloque obtiene un 3,125 % adicional a su recompensa. Respecto a los uncles, se recompensa a cada uno de sus mineros con 93,75 % de la ganancia estandar por minar un bloque.

[9] Implementación simplificada del protocolo GHOST: https://github.com/ethereum/wiki/wiki/White-Paper#modified-ghost-implementation

Acorde a lo especificado en la sección *Blocktree to Blockchain* del documento de especificación formal de Ethereum [Woo14] y a la implementación actual en el código fuente[10], el cálculo de dificultad de cadenas en la blockchain es similar al de Bitcoin: La dificultad total de una cadena se calcula como la suma de las dificultades de sus bloques, ignorando los bloques uncles referenciados. Vale la pena destacar que, ante un empate entre cadenas, Ethereum implementa la heurística propuesta en el trabajo de Eyal et al. [ES14], que consiste en elegir una de ellas uniformemente al azar. Esta última busca minimizar la vulnerabilidad a ataques de minado egoísta.

El ajuste de dificultad en Ethereum se realiza dinámicamente según el *hashing power* de la red. Una mejora que ofrece Ethereum respecto a Bitcoin, es un ajuste más gradual en la dificultad. Existen dos versiones de algoritmos de ajuste de dificultad en Ethereum: *Frontier* y *Homestead*.

3.3.2 Protocolo

Al iniciar una conexión entre dos nodos, éstos realizan un *«handshake»*, enviando un mensaje de *Status*, en el que comunican la máxima versión del protocolo que soportan.

En Ethereum hay dos formas de notificar sobre un nuevo bloque: *NewBlockMessage*, que envía un bloque entero, o *NewBlockHashes*, que envía sólo el hash del nuevo bloque.

Un nodo puede pedir el header de un bloque (o de varios), mediante el mensaje *GetBlockHeaders*, y varios bodies mediante el mensaje *GetBlockBodies*, estos mensajes son respondidos con *BlockHeaders* y BlockBodies respectivamente.

A continuación explicamos en detalle el protocolo de propagación implementado en Ethereum, ya que juega un papel fundamental en lo que resta del trabajo.

La aparición de un nuevo bloque en un cliente puede deberse a dos motivos: Un nuevo bloque fue minado o un bloque previamente desconocido fue anunciado en la red. Ante cualquiera de estos dos eventos, el cliente realiza una propagación del bloque hacia sus

[10] https://github.com/ethereum/go-ethereum/commit/bcf565730b1816304947021080981245d084a930

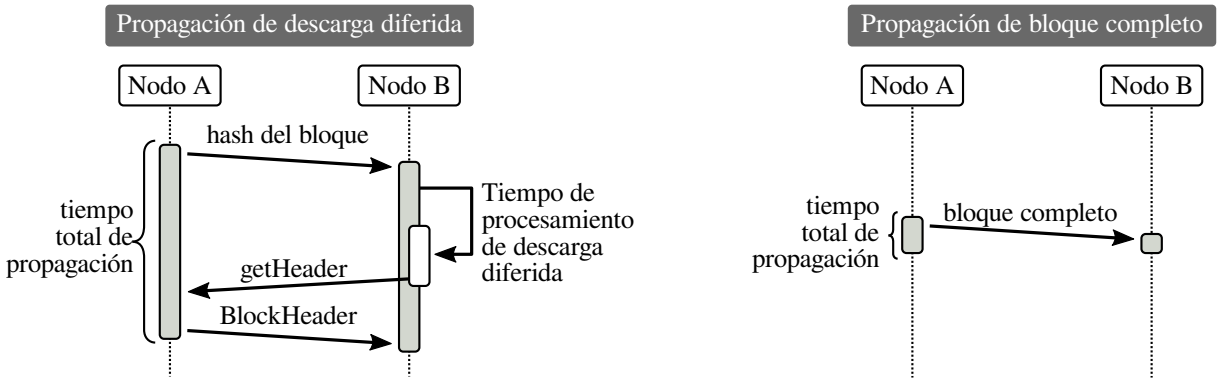


Figura 3.9: Ejemplo de propagación de un bloque en entre dos nodos. Diferentes tipos de transmisión.

peers. Para evitar transmisiones innecesarias a estos últimos, cada cliente lleva un registro de los bloques conocidos por cada uno.

Utilizando el registro antes mencionado, al momento de difundir un bloque, se elabora una lista de peers que potencialmente no tengan conocimiento de este último.

Sea P dicha lista de *peers* en algún orden:

1. Se toma la sublista comprendida por los primeros $\sqrt{len(P)}$ y se les envía el bloque completo a estos peers.

2. Al resto de los peers se les envía el **hash** del bloque.

Para lograr que ningún peer obtenga los bloques antes que otro, la lista de peers tiene un orden distinto en cada cliente, lo que genera un protocolo equitativo con todos los nodos de la red.

De esta forma un subconjunto de los peers obtiene el bloque de forma instantánea y lo agrega a la blockchain mientras que el resto, lo descarga en forma diferida en base al hash obtenido. El protocolo actualmente realiza pedidos de bloques programados cada 500 milisegundos.

Un árbol de propagación de bloques es tal que su raíz es el nodo minero y existe un camino dirigido hacia cada uno de los nodos del resto de la red. En cada arista, indicamos el tiempo de propagación y el tipo de transmisión.

Clasificamos las aristas del árbol en transmisiones de bloque completo y transmisiones de descarga diferida.

De forma similar caracterizamos los tipos de propagación en 3 clases según el camino en el árbol entre el minero y el nodo:

- Caminos de transmisión completa: Son aquellos donde todas sus aristas corresponden a transmisiones de bloque completo.
- Caminos de transmisión diferida: Son aquellos donde todas sus aristas corresponden a transmisiones de hashes y su posterior descarga diferida.
- Caminos combinados: Son aquellos caminos donde existen aristas de los dos tipos posibles.

Capítulo 4

Trabajo relacionado

Gran parte del análisis que se realiza en Bitcoin para descubrir la topología y propiedades de la red, se realiza utilizando clientes instrumentados, que se conectan a varios nodos de la red y recolectan información sobre sus pares.

Decker y Wattenhofer en [DW13] utilizan un cliente instrumentado para analizar el tiempo de propagación de bloques y generación de forks en la red. Su cliente se conecta a una gran cantidad de nodos de forma pasiva, es decir, que no reenvía los mensajes que recibe de los nodos a los que está conectado. En base a la información recopilada, analizan el comportamiento de la red durante 10,000 bloques, obteniendo los siguientes resultados: El tiempo de propagación promedio de bloque en Bitcoin es de 12,6 segundos, estiman la frecuencia de generación de forks en función del tiempo de propagación de bloques, concluyen que el tiempo de propagación de los bloques es la principal causa de *forks* en la red y hallan una correlación fuerte entre el tiempo de propagación de los bloques y el tamaño de los mismos.

Además proponen cambios para mejorar el tiempo de propagación de bloques: tener mayor conectividad entre nodos, propagar las notificaciones de bloque nuevo antes de recibir el bloque completo, y ser menos estrictos en la verificación de bloques antes de propagarlos. Tomaron nuevas mediciones con estos cambios y obtuvieron mejoras en los tiempos de pro-

pagación. Este trabajo es uno de los primeros acerca del efecto del tiempo de propagación en la generación de forks en la red.

Por otro lado, Donet et al. en [DPSHJ14] también mide el tiempo de propagación de bloques, pero haciendo foco en la topología de la red. El mismo busca calcular la distribución de IPs de nodos Bitcoin por país. Para ello, su cliente envía periódicamente mensajes de GetAddr a fin de obtener las IPs de nodos de la red, realizando consultas recursivas y obteniendo nuevos nodos en cada paso. Este experimento se realizó durante un período mayor a un mes. Para el análisis de los tiempos de propagación, sólo tomaron mediciones de 710 de bloques (cinco días).

Los resultados de este trabajo fueron:

- Si bien hay una correlación entre el tamaño de bloque y el tiempo de propagación, ésta no es lineal.
- Reportan la métrica de qué porcentaje de bloques se propagó antes de un tiempo dado a un porcentaje de nodos.
- Encuentran 87,000 IPs distintas de nodos de Bitcoin, pero solo 6000 IPs se mantuvieron constantes a lo largo del experimento.

Los autores publicaron junto con el trabajo la distribución de IPs por país que luego utilizaremos para armar topologías virtuales para nuestros experimentos.

Además del trabajo anterior, en otros sitios de la comunidad se puede obtener información de la geolocalización de nodos en el globo. Por ejemplo para Bitcoin tenemos [bit] y para Ethereum [eth].

Miller et al [MLP+15] crearon *Coinscope*, una herramienta para mejorar la caracterización de la distribución de nodos en la red. Ésta utiliza metadatos provistos por la respuesta de GetAddr y diferencia conexiones entrantes de salientes. Por otro lado, busca detectar qué nodos de la red son los más «influyentes». Para esto particiona la red y envía transacciones conflictivas a distintos nodos de la red. Luego, analiza qué transacciones terminan

apareciendo en un bloque minado.

Los resultados obtenidos fueron que, en general, los nodos tienen aproximadamente ocho peers, lo que se coincide con el cliente oficial de Bitcoin, sin embargo, hay algunos que tienen más de 100 peers. Los autores indican que estos últimos podrían ser pools de minería.

Si bien la red se asemeja un grafo aleatorio, el trabajo menciona una prueba de «comunidades» (Louvain) sobre los datos obtenidos. La prueba separa los nodos en n conjuntos, $A_1, \ldots, A_n$, que cumplen la propiedad de que $\forall A_i,\ i\ \in [1, n]$ vale que $\forall x \in A_i,\ \forall j \in [1, n],\ j \neq i,\ \#peers(x,\ A_i) > \#peers(x,\ A_j)$, donde $\#peers(x,\ A)$ es la cantidad de $peers$ que tiene x en el conjunto A.

Usando Coinscope toman muestras de la topología de la red y las comparan con la cantidad de comunidades de grafos aleatorios. El resultado es que la media de estas métricas se encuentra a 2 desvíos estándares de distancia de la media de las métricas en grafos aleatorios. Los autores proponen que esto se debe a que los nodos se empiezan a conectar a partir de «boot nodes» y de a poco se van expandiendo.

Con respecto a los nodos influyentes, el 2 % de los nodos es responsable de 75 % de las transacciones que aparecen en los bloques: si se le envía una transacción a esos nodos, es muy probable que aparezca en un bloque, por más que haya otra transacción conflictiva en la red. Esto sirve para detectar pools de minería, ya que estos tienen gran poder de cómputo y serían nodos influyentes.

Croman et al. en [CDE+16] analiza distintos problemas de escalabilidad en Bitcoin. Parte desde distintos enfoques como protocolos de red, «throughput» de transacciones, consenso y almacenamiento de datos. Detalla que uno de los cuellos de botella es el tiempo de propagación de bloques, sugiriendo como mejora posible, disminuir el diámetro de la red. Sin embargo, concluye que se requiere un rediseño de los protocolos y procesos como única solución viable a largo plazo.

Sompolinsky y Zohar en [SZ13] analizan el throughput de transacciones en Bitcoin y obtienen una cota superior para este valor, regida por el diseño del protocolo de consenso.

Presentan como solución el protocolo GHOST, el cual introduce el concepto de *«bloques uncles»* para disminuir la cantidad de forks y mejorar el «throughput».

En septiembre de 2016 fue aprobado el Bitcoin Improvement Proposal 512, una propuesta para mejorar el tiempo de propagación de bloques enviando menos información. La solución aprobada se denomina «Compact Blocks» ([cmp]), dejando rechazada a la propuesta de utilizar «XThin Blocks» ([xth]).

Existen también numerosos análisis teóricos sobre el protocolo de Bitcoin y su seguridad. Uno de los más relevantes es el de Garay et al [GKL15] en donde se analiza la seguridad del protocolo al tener una menor relación entre tiempo de generación de bloques y tiempo de propagación, entre otras cosas. En los trabajos de Kiayias et al. [KP15] y [KP16] se analiza la seguridad de blockchains con intervalos muy cortos entre bloques, incluyendo el efecto del protocolo de consenso usado por Ethereum.

Marco Vanotti en [Van16] analiza los efectos de decrementar el tiempo entre bloques y aumentar el diámetro de la red respecto a la generación de forks en la red. Utiliza Mininet y Maxinet para emular las topologías de red descriptas en los trabajos de Donet et al. [DPSHJ14] y Miller et al. [MLP+15]. También se realiza una validación entre Mininet y Maxinet con experimentos basados en la criptomoneda *RSK*.

Dihn et al. en [DWC+17] desarrolla una herramienta llamada *Blockbench* que permite integrarse a cualquier sistema de blockchain por medio de una API. El foco del trabajo es el uso de redes privadas de blockchain para el desarrollo de aplicaciones usualmente montadas sobre bases de datos estándar, como por ejemplo aplicaciones bancarias. El framework presentado mide distintos aspectos de performance, escalabilidad y tolerancia a fallos utilizando como entrada datos recolectados de clientes reales y datos generados sintéticamente. Realizan experimentos sobre Ethereum, Parity y Hyperledger Fabric y concluyen que las redes blockchain actuales no son adecuadas para procesamiento de datos a gran escala.

Gervais et al. en [GKW+16] diseñan un simulador sobre *ns-3* para analizar la seguridad de la red de Bitcoin ante variaciones en parámetros de la red, como tiempo entre bloques

o el algoritmo de consenso. Se modelan las Blockchains y protocolos de Bitcoin, Ethereum y otras criptomonedas, así como también algunos ataques conocidos. Este trabajo hace foco sobre la seguridad, mostrando que, por ejemplo, Ethereum requiere 37 bloques en la mejor cadena para lograr la misma seguridad que logra Bitcoin con solo seis, suponiendo que existe un atacante que controla un 30 % del poder de cómputo de la red.

Apostolaki et al. en [MAL17] analiza posibles ataques a Bitcoin alterando el comportamiento del protocolo de ruteo de internet o interceptando tráfico. Encuentran que un atacante puede aislar al 50 % del hashing power de la red, tomando el control de menos de 100 prefijos *BGP*[1]. Aislando partes de la red o ralentizando los tiempos de propagación, el daño realizado a Bitcoin no es despreciable, ya que puede causar que gran parte del hashing power se desperdicie o permitir el uso de exploits para realizar ataques de double spending. Finalmente proponen medidas para contrarrestar el efecto de estos ataques.

Existen trabajos acerca de ataques de minado egoísta a Bitcoin, así como también propuestas para mejorar la resistencia a éstos. Se presentan algunos de ellos a continuación.

Eyal et al en [ES14] introduce la noción de minero egoísta y explica sus impactos sobre Bitcoin. Finalmente propone una modificación al protocolo para disminuir la exposición a estos ataques.

Sapirshtein et al en [SSZ15] estudia el porcentaje mínimo de poder de cómputo necesario para llevar a cabo un ataque de estas características. También analizan varias propuestas para defenderse de estos ataques y su efectividad. Por último muestran que en escenarios de red cuyos enlaces tengan latencia elevada, incluso nodos con escaso poder de cómputo están motivados a minar de forma egoísta.

Zhang et al, en [ZP17] propone cambios al protocolo de Bitcoin para mejorar la defensa a ataques de minado egoísta. La modificación propuesta incluye un cambio en el protocolo de consenso, beneficiando a bloques que referencien bloques anteriores y sean enviados a tiempo. De esta forma, eliminan el incentivo de no compartir los bloques instantáneamente

[1] https://en.wikipedia.org/wiki/Border_Gateway_Protocol

luego de minarlos.

Todos estos trabajos muestran el interés de la comunidad por entender y mejorar la eficiencia de las criptomonedas. En esta tesis utilizaremos varias de las ideas mencionadas y de los datos provistos en estas publicaciones.

Capítulo 5

Metodología

Este capítulo detalla la metodología utilizada para la creación de la herramienta y la realización de experimentos.

En 5.1 explicamos la instrumentalización del cliente de criptomonedas para permitir la persistencia de eventos en archivos de registro, las implementaciones del módulo de minado simulado y las modificaciones al código que permitieran utilizar el algoritmo de minado real con tiempo entre bloques configurable. En la sección 5.2 hablamos acerca de detalles de la plataforma experimental. Las definiciones acerca de topologías y escenarios de experimentación están detalladas en 5.3. Finalmente, a partir se la sección 5.4 se introducen las métricas creadas para el análisis de resultados.

5.1 Instrumentalización del cliente

Con respecto al cliente Ethereum, se tomó como base la versión en lenguaje *Go*, ya que es la versión más utilizada por la comunidad según [eth]. Sobre este cliente se implementaron los módulos de minado simulado y de registro de trazas de ejecución.

5.1.1 Registro de trazas de ejecución

Se procedió a instrumentalizar el código con un conjunto de callbacks que son ejecutados ante la ocurrencia de ciertos eventos en la red y se creó un modulo para implementar la persistencia de estos últimos. Los eventos que consideramos relevantes para su registro son:

- *NewBlockReadyForBlockchain*: Este evento es registrado ante la aparición de un nuevo bloque válido para ser insertado en la blockchain.
- *LogBroadcastBlock*: Este evento es registrado cada vez que el cliente mina un nuevo bloque o ante la retransmisión de un bloque en la red.

Estos últimos son almacenados en un archivo con la siguiente información adicional acerca del contexto:

- *node*: Indica el emisor del evento.
- *monotonicNano*: Denota un reloj monótono dentro del nodo emisor. Este valor es tomado utilizando la syscall `clock_gettime` con el flag `CLOCK_MONOTONIC`[1].
- *realNano*: Indica el tiempo *wall-clock*[2] del nodo emisor del evento. Este valor es tomado utilizando la syscall `clock_gettime` con el flag `CLOCK_REALTIME`.

Adicionalmente, cada uno de los eventos contiene toda la información de contexto que resulte pertinente. Ante la propagación de un bloque, se guardan los destinatarios de dicho envío, y en eventos asociados a la recepción de bloques, se almacena el nodo emisor del broadcast.

Al finalizar el experimento, se recolectan todos los archivos de registro de todos los hosts del experimento, se agrupan en un solo archivo y se ordenan por wall-clock time. De esta forma se tiene un único archivo con una visión unificada y ordenada temporalmente de los eventos que ocurrieron en la red.

[1] https://linux.die.net/man/3/clock_gettime
[2] https://en.wikipedia.org/wiki/Wall-clock_time

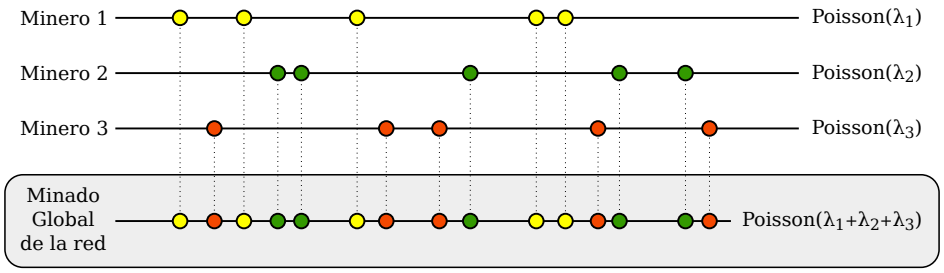


Figura 5.1: Suma de tres procesos de Poisson independientes. Cada línea simula el proceso de minado a lo largo del tiempo. El resultado es también un proceso de Poisson.

5.1.2 Minado simulado

Dada la necesidad de modelar el proceso de minado sin tener que invertir poder de cómputo real planteamos un modelo estadístico que simula su comportamiento.

Caracterización estadística del proceso de minado

Como los hashes calculados por los mineros son criptográficamente seguros, podemos suponer que están distribuidos uniformemente. Entonces, el cómputo realizado por los mineros puede modelarse utilizando un ensayo de *Bernoulli* con cierta probabilidad de éxito.

Debido a que el proceso de minado no es más que la repetición de estos ensayos en intervalos cortos de tiempo, puede pensarse el proceso de minado de un cliente como un proceso de *Poisson* con cierto parámetro λ_i que indica la cantidad de bloques minados por unidad de tiempo por dicho cliente. Si tenemos n mineros independientes en la red, existirán esta cantidad de procesos Poisson independientes.

Como indica la figura 5.1, si $\lambda_1, \ldots, \lambda_n$ son los parámetros para cada uno de los mineros de nuestra red, en la visión global de la red, donde todos los mineros corren en forma paralela, el proceso de minado es un proceso de Poisson con parámetro $\lambda_g = \sum_{i=1}^{n} \lambda_i$, correspondiente a la suma de procesos de Poisson independientes de cada minero.

Teniendo en cuenta que el porcentaje de hashing power del minero i-ésimo en la red está determinado por la fórmula $\frac{\lambda_i}{\lambda_g} \times 100$, pueden generarse distribuciones arbitrarias de poder de cómputo en la red para diversos escenarios.

En este modelo, el tiempo entre dos eventos en un proceso de Poisson de parámetro λ sigue una distribución exponencial con media $\frac{1}{\lambda}$, por lo cual si queremos que el tiempo medio entre bloques sea de 10 segundos, tendremos que $\frac{1}{\lambda} = 10$, resultando en que $\lambda = \frac{1}{10}$.

Utilizando esta caracterización del proceso de minado, implementamos nuestro módulo de minado simulado en el cliente instrumentalizado.

Abstracción del ajuste de dificultad

Dado que ni la topología ni el hashing power varían dinámicamente en ninguno de nuestros experimentos, queremos abstraernos del ajuste de tiempo entre bloques por dificultad. Es por esto que nuestro algoritmo de minado simulado no utiliza ningún mecanismo de este tipo. La aparición de bloques en la red sigue un proceso de Poisson acorde a la distribución de hashing power configurada de antemano.

Sin embargo, para los experimentos de validación del proceso de minado simulado propuesto, fue necesario agregar una opción a la aplicación que permita introducir un tiempo entre bloques personalizado para minado real. Esta modificación impacta en el mecanismo de ajuste de dificultad. Dado que Homestead es un algoritmo de convergencia que mantiene el tiempo entre bloques esté entre 10 y 20 segundos, bajo este modo personalizado de tiempo entre bloques se utiliza el algoritmo de convergencia Frontier, que administra los ajustes de dificultad en base a la convergencia del tiempo entre bloques a un valor establecido de antemano.

Implementación

Como dijimos anteriormente, la distribución que sigue el proceso de minado respecto al tiempo entre bloques es exponencial. Podemos entonces, simular esta parte del modulo de minado tomando un valor aleatorio que siga esta distribución y simplemente llamar a la función *sleep* del sistema operativo para que espere dicho tiempo sin utilizar recursos de CPU.

Parte de la dinámica del protocolo requiere la validación de los bloques recibidos. Dicho proceso demora un tiempo no despreciable pero depende únicamente del hardware donde se ejecuta. En nuestra plataforma experimental la verificación toma un tiempo promedio de 10 milisegundos. Con esta información, simulamos el tiempo de espera con una función sleep en nuestro algoritmo.

En este sistema dinámico donde todos minan sobre el último bloque de la cadena principal, es común la ocurrencia de *hazards*[3] ya que los mineros trabajan de forma concurrente en la red. Bajo estas condiciones, cada minero debe recibir notificaciones acerca de los cambios en la blockchain de la red. Ante la aparición de un nuevo bloque en la red que produzca un cambio de cadena principal, el minero debe reiniciarse y comenzar a minar sobre el último bloque de la nueva cadena. Nuestra implementación de minado simulado respeta esta condición ya existente en el algoritmo de minado real.

5.2 Plataforma experimental

A continuación detallaremos la infraestructura sobre la cual fueron ejecutados todos los experimentos. Además mencionamos requerimientos de sincronización a tener en cuenta en sistemas distribuidos de estas características.

5.2.1 Infraestructura

Todos los experimentos de este trabajo utilizan como topología física la infraestructura del *Cluster Museo*. Dicho cluster cuenta con 15 maquinas adquiridas por el CECAR[4] en 2009, cuyas especificaciones pueden verse en la tabla 5.1.

Además existe un nodo distinguido que es el punto de acceso al cluster, cuyas especificaciones están descriptas en la tabla 5.2. Este nodo administra entre otras cosas: El controlador de red y el servicio de frontend de Maxinet, un servidor de NFS para compartir el home, un servidor Nagios, etc.

[3] https://en.wikipedia.org/wiki/Race_condition
[4] http://cecar.fcen.uba.ar/

Especificaciones	
Procesador	2 x Intel(R) Xeon(TM) CPU 2.66GHz
Memoria	4 GB DDR3
Red	Ethernet 10/100/1000 Mbps
Disco	80GB 7200RPM SATA

Cuadro 5.1: Especificaciones de los nodos del cluster museo

Especificaciones	
Procesador	2 x Intel(R) Xeon(TM) CPU 2.66GHz
Memoria	8 GB DDR3
Red	Ethernet 10/100/1000 Mbps
Disco 1	80GB 7200RPM SATA
Disco 2	2TB 7200RPM SATA

Cuadro 5.2: Especificaciones del nodo distinguido del cluster museo

Para la interconexión de los nodos contamos con un switch `Tp Link TL-SG1024` con soporte para *Jumbo Frames*. Esta última característica es fundamental para nuestros experimentos sobre *Maxinet*, ya que deben poder transmitirse paquetes con MTU mayor a 1500 para compensar el overhead producido por los túneles GRE.

Para el monitoreo del cluster, se configuró un servidor *Nagios* en el nodo distinguido. Desde este último podemos observar la evolución en tiempo real de métricas tales como:

- Offset de *NTP*[5]: Tiempo de desfasaje de wall-clock entre un nodo y el servidor central.
- Estadísticas de tiempos de respuesta de red de los nodos.
- Carga de CPU, Uso de memoria RAM y porcentajes de espacio libre en los discos.

5.2.2 Sincronización de relojes

Dado que estamos corriendo dentro de un sistema distribuido con varias computadoras dentro de una red, debemos sincronizar los relojes de todos los nodos de forma tal que no ocurran alteraciones de orden entre los eventos. Por otro lado, mientras menor sea el *offset* promedio de diferencia de relojes entre los nodos, más precisas serán nuestras mediciones.

[5] https://es.wikipedia.org/wiki/Network_Time_Protocol

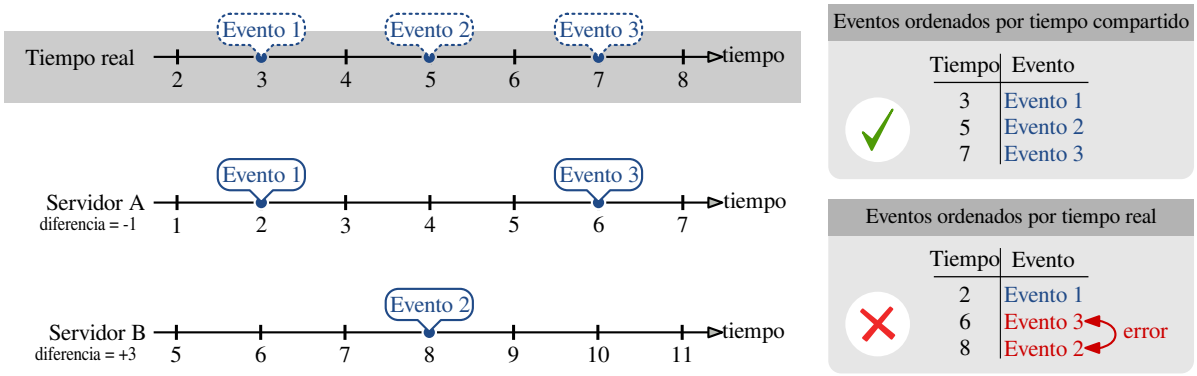


Figura 5.2: Ejemplo de problema de sistema distribuido sin correcta sincronización de relojes

Consideramos establecer el límite máximo aceptable de diferencia de sincronización en 0,3 milisegundos. Este valor corresponde al tiempo promedio de propagación de paquetes de red en nuestra plataforma experimental y nos permite asegurar que no existirán alteraciones de orden en los eventos.

Para lograr que la sincronización se mantenga en estos rangos, configuramos un servicio NTP con un *poll time* de 16 segundos y fijamos como reloj del sistema *hpet*[6] en cada host físico.

5.3 Topologías y redes privadas

Comenzamos esta sección introduciendo conceptos sobre topologías físicas, emuladas y lógicas. Estas últimas hacen referencia a distintas capas de red en los experimentos. Más adelante hablaremos de blockchains privadas sobre Ethereum y sobre la generación de topologías virtuales relevantes a nuestro trabajo.

5.3.1 Tipos de topologías

Definimos tres tipos de topologías según el contexto donde se utilicen. Cuando nos referimos a la *Topología lógica* de un experimento, estamos hablando de la relación de peers entre nodos en la red de la criptomoneda. Al referirnos a la *Topología emulada* estamos hablando de la red definida utilizando *SDNs* y cuando hablamos de la *Topología física* nos referimos

[6] https://es.wikipedia.org/wiki/High_Precision_Event_Timer

Topología lógica	Clique. Todos los nodos en la red de Ethereum tienen a todos los demás como peers.
Topología emulada	Anillo. El host i-ésimo solo tiene conectado al nodo siguiente o al primero.
Topología física	Arbitraria

Cuadro 5.3: Ejemplo de configuración de topologías

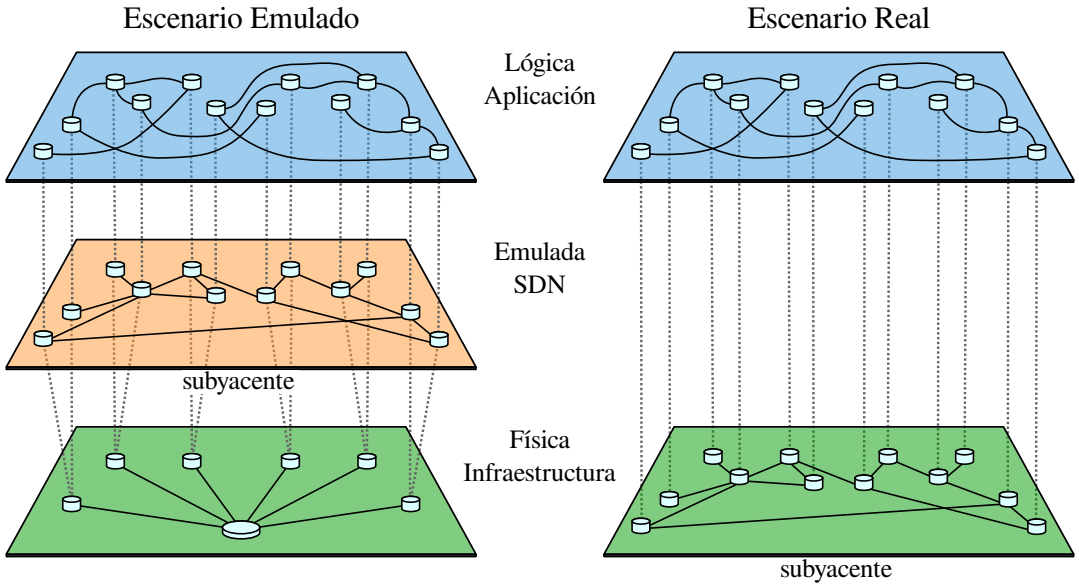


Figura 5.3: Relación entre topologías

a como están conectados los hosts de la infraestructura física. La tabla 5.3 describe una posible configuración válida.

Si nos basamos en la configuración de la tabla 5.3, cuando un nodo en Ethereum realiza un broadcast, ve al resto de la red como peers, pero en la topología subyacente las conexiones no necesariamente son directas. En este caso, un mensaje enviado desde el nodo 1 al nodo n, es un paso en la topología lógica, pero tiene que atravesar $n - 1$ links de red emulada subyacente y una cantidad desconocida en la red física.

En la figura 5.3 puede observarse otro ejemplo de la disposición de las topologías en niveles y la relación entre las diferentes capas.

Más adelante en el trabajo, las topologías lógica y emulada son a veces llamadas topologías virtuales. También cuando hablamos de la red subyacente nos referimos a la topología de

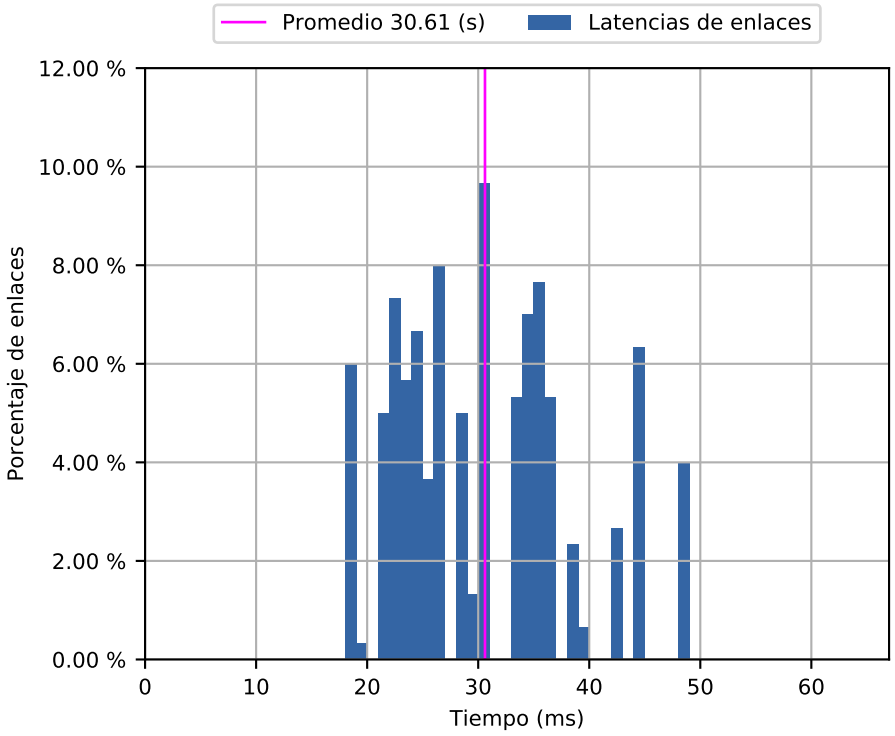


Figura 5.4: Distribución del dataset de latencias reales

red que se encuentre inmediatamente por debajo de la topología lógica.

5.3.2 Generación de las topologías virtuales

Para construir una topología virtual aleatoria se generó un grafo conexo aleatorio donde cada nodo tiene entre i y j *vecinos*. Luego se le asigna a cada nodo un país de forma aleatoria siguiendo la distribución del trabajo de Donet et al. [DPSHJ14] y para finalizar se asignan las propiedades correspondientes entre enlaces según los países de cada extremo de la arista, datos que son extraídos del trabajo de Miller et al. [MLP+15][7]. La distribución de latencias para los enlaces de este dataset puede verse en 5.4.

Para poder caracterizar redes de mayor diámetro con el hardware y las herramientas disponibles, se repiten los experimentos sobre varios conjuntos de latencias: Latencia real definida por los datos que mencionamos antes, Latencia 2X que duplica los valores anteriores, y finalmente Latencia 200 ms. que satura todos los links a una latencia de 200

[7]Dichas propiedades son: Porcentaje de pérdida de paquetes, Latencia y Jitter.

ms.

Como mencionamos en 2.2.2, al ser las latencias descriptas en 5.4 dos órdenes de magnitud mayores que las latencias de la red física subyacente, nos aseguramos de que las propiedades del enlace serán estables al utilizar Maxinet.

5.3.3 Blockchain privada sobre Ethereum

Para la realización de todos los experimentos, nuestra herramienta permite instanciar redes blockchain privadas basadas en Ethereum.

Como entrada, se recibe un archivo que describe la topología lógica de la red e incluye para cada nodo un nombre que lo identifique, sus nodos vecinos, su país, su IP y datos de configuración del minado como ser su hashing power λ_i en la red.

Este archivo es leído por la herramienta para generar una topología emulada con Maxinet idéntica a la topología lógica indicada, utilizando los datos geográficos para asignar latencias a los enlaces. Sobre esta red emulada, se lanza un cliente Ethereum en cada nodo y se asignan sus vecinos como peers, conformando la blockchain privada. Finalmente, se inicia el proceso de minado simulado en los nodos mineros correspondientes a la topología descripta.

5.4 Métricas

Utilizando los archivos de registro de eventos, nuestra herramienta genera las métricas detalladas a continuación.

5.4.1 Estadísticas del proceso de minado

Clasificación de bloques y rendimiento de la blockchain

A partir de los eventos *NewBlockReadyForBlockchain* y *LogBroadcastBlock* reconstruimos la blockchain y clasificamos a los bloques en dos conjuntos según su pertenencia o no a

la cadena principal. Aquellos que pertenezcan serán parte del conjunto de bloques de la cadena principal o *bloques efectivos*, mientras los que no se considerarán parte del conjunto de *bloques stale*.

La relación entre el cardinal del conjunto de bloques stale y el conjunto de bloques minados, nos da una noción acerca del rendimiento de la red. Diremos que una red es «más eficiente» a medida que el porcentaje de *bloques stale* sea menor.

Distribución de poder de cómputo

Utilizando la primera ocurrencia de un evento *LogBroadcastBlock* en el tiempo para cada bloque, podemos determinar quién fue el nodo que lo minó. De esta forma podemos mostrar la distribución de poder de cómputo entre los mineros.

Distribución de ganancias por minero

Utilizando las últimas dos métricas definidas podemos calcular para cada minero la diferencia porcentual entre los bloques minados y los bloques que quedaron en la cadena principal. Esto nos indica cuanto porcentaje del poder de cómputo desperdició cada minero en bloques stale.

Distribución de tiempos de generación de bloques

Cada bloque tiene un *timestamp* asociado a su aparición en la red. Con ellos podemos calcular el intervalo de tiempo entre las apariciones de bloques y así obtener la distribución estadística de este proceso.

Forks

El análisis de los forks se realiza tanto en profundidad como en ancho. Esto puede puede visualizarse mejor en la figura 5.5 que muestra la misma Blockchain con algunos forks. A la izquierda se cuentan, para cada bloque, cuántos bloques competitivos tuvo, agrupando

luego por estas cantidades. Por ejemplo, si en dos ocasiones hubo 3 bloques competitivos, esto representaría que 3 mineros hallaron un bloque antes de enterarse del que halló otro minero[8].

Bloques competitivos

1 bloque	2 bloques	3 bloques	4 bloques
3	2	2	0

Fork por profundidad

Prof=1	Prof=2	Prof=3	Prof=4
2	0	2	0

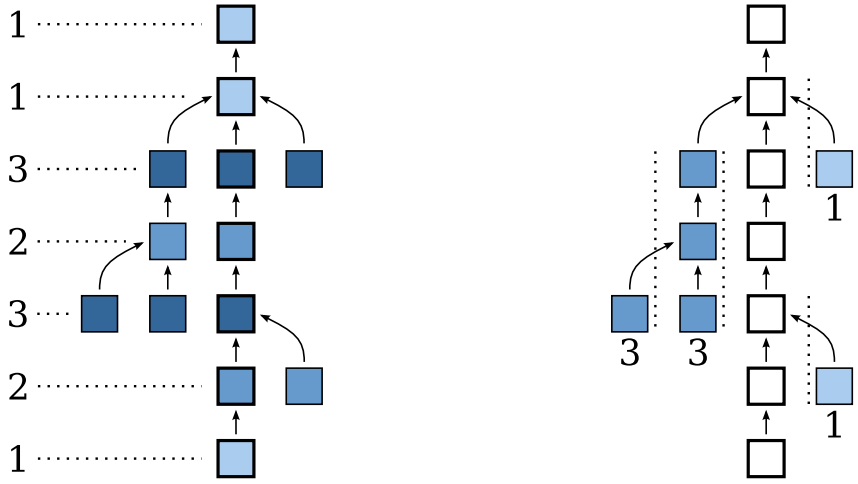


Figura 5.5: Ejemplo mostrando dos perspectivas distintas para analizar los forks de la red: por bloques competitivos (izquierda) y por profundidad de los forks (derecha).

A la derecha de la figura, se cuentan los forks según su profundidad. La profundidad de un fork está definida como la distancia desde el bloque de mayor altura hasta algún bloque de la cadena principal. Esta métrica nos aporta información acerca de cómo estuvo dividido el poder de cómputo de la red.

Consideramos la cantidad de forks de profundidad mayor o igual a 3 como un indicador de la **confiabilidad** de una red. Una cantidad no despreciable de estos fenómenos implican una degradación en el **rendimiento** de la red y provocan que la última deje de considerarse confiable por la cantidad de transacciones que quedan en bloques «stale» y deben ser reubicadas en nuevos bloques.

[8]Decimos que minaron el bloque «a la vez».

5.4.2 Propagación de bloques

Como primera medida para analizar este aspecto, reconstruimos el árbol de propagación para cada bloque transmitido.

Construcción del árbol de propagación

A partir de los eventos registrados, reconstruimos el grafo de propagaciones. Luego identificamos el subárbol inducido que se corresponde con las propagaciones efectivas, proceso que puede observarse en la figura 5.6. Para esto seguimos las siguientes reglas:

1. Dado que la topología lógica es un grafo, puede ocurrir que un bloque que se propaga en la red, le llegue varias veces a un nodo por distintos caminos. Cuando esto ocurre, consideramos únicamente la primera llegada del bloque y descartamos las recepciones subsiguientes.

2. Por una *race condition* existente en el código del cliente, puede ocurrir que se envíen de un cliente al mismo peer, tanto el bloque completo como su hash. En este caso, descartamos la recepción del hash y nos quedamos con la recepción del bloque completo.

3. Puede ocurrir que en el envío de un hash entre nodos, el receptor agende la descarga del bloque y que antes de finalizar la misma, reciba el bloque completo por otro camino. En este caso, descartamos la recepción del hash y nos quedamos con el bloque completo.

Sobre el árbol generado, calculamos cuales nodos recibieron el bloque en cada nivel[9]. Los datos extraídos de cada nivel incluyen nombre del nodo, tiempo total de propagación[10] y el tipo de propagación. Con la información obtenida calculamos las siguientes métricas:

[9]Una recepción en el nivel i-ésimo indica que hubo $i-1$ retransmisiones del bloque entre el minero y los nodos de dicho nivel.

[10]Diferencia de tiempo entre el primer broadcast del minero y la primera recepción del bloque en dicho nodo. De esta forma tenemos en cuenta el tiempo de procesamiento y retransmisión en cada nodo intermedio del camino.

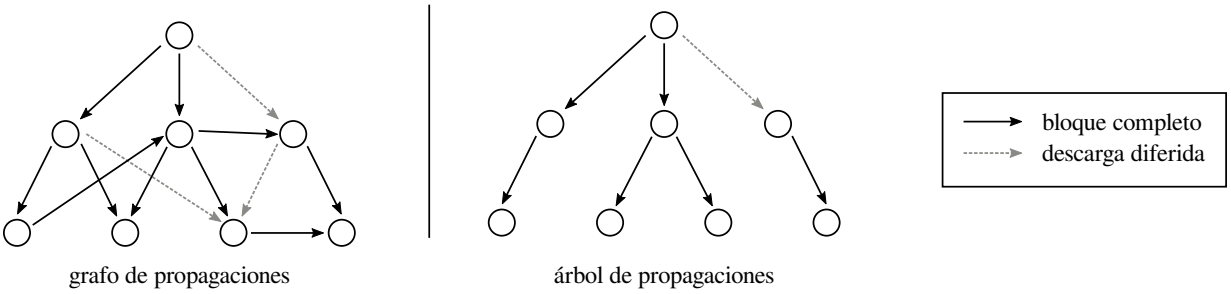


Figura 5.6: Grafo de propagación - Árbol generador de propagaciones

Histograma de tiempos de propagación

Con toda la información generada en el árbol de propagación, se generan los histogramas definidos en 5.4.

Alcance de propagación de bloques

Con esta métrica, se busca capturar cuánto tiempo se tarda, en promedio, en propagar un mensaje a distintos porcentajes de nodos de la red.

En la figura 5.7 se puede observar un posible resultado de esta métrica.

Histogramas de propagación de bloques	
Tipo	Detalle
Global	Se registran todas las propagaciones de todos los bloques hacia todos los nodos.
Tipo de propagación	Restringen el histograma global solo a caminos en el árbol según la clasificación definida en 3.3.2
Nivel	Muestran solo las propagaciones cuyos caminos desde el nodo emisor en el árbol tengan una longitud fija definida.
Bloque	Muestran los tiempos de propagación de un bloque en particular.
Nodo	Muestran los tiempos de propagación para solo un nodo de la red.

Cuadro 5.4: Tipos de histogramas de tiempos de propagación de bloques

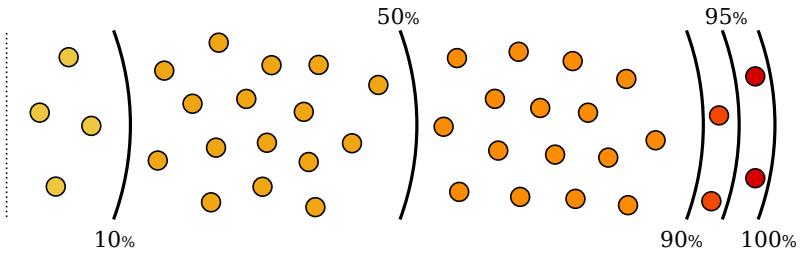


Figura 5.7: Esquematización sobre propagación de información a un porcentaje de la red

Clasificación de caminos y transmisiones de a pares

Los porcentajes de los tipos de caminos definidos en 3.3.2 respecto a todo el experimento se obtienen a partir de las cantidades indicadas por los árboles de cada bloque. Estos datos se suman agrupados por tipo y se obtiene un cálculo global. Finalmente, se expresan dichos valores como porcentajes.

De forma similar se calculan los porcentajes de los tipos de propagaciones inmediatas entre pares de nodos (aristas del árbol) a lo largo de todo el experimento.

Capítulo 6

Validación

6.1 Motivación

Este capítulo presenta la validación de la metodología presentada. En la misma se busca verificar la relación entre los procesos de minado real y simulado. Además se muestra que las herramientas utilizadas para emular los ambientes de red se comportan adecuadamente para los casos de estudio propuestos en el resto del trabajo.

6.2 Sincronización entre nodos

Comenzaremos presentando la validación de la sincronización de relojes en nuestra plataforma experimental. Se realizó un seguimiento con la herramienta de monitoreo Nagios, configurando notificaciones en caso de que el offset de sincronización de los relojes excediera un umbral de 0,3 ms. Concluimos que el servicio de sincronización funciona correctamente dentro de este rango. Cabe destacar que no se registró la aparición de eventos fuera de orden durante la realización de los experimentos de este trabajo.

Configuración del experimento	
Topología física	*Ethernet*/Museo
Topología emulada	Ninguna
Topología lógica	Clique - 10 nodos
Bloques analizados	3800
Porcentaje mineros	50 % de la red
Tiempos entre bloques	7, 14,5 y 21 segundos

Cuadro 6.1: Configuración de los experimentos de procesos de minado

6.3 Procesos de minado

La naturaleza de este tipo de experimentos, donde la aparición de bloques depende de un proceso sin memoria–los bloques anteriores no influyen sobre los próximos bloques a minar–, y las características del experimento no varían a lo largo del tiempo, permite que en lugar de repetir el mismo varias veces, se pueda tomar una muestra lo suficientemente grande, y que esta última sea representativa. La cantidad de bloques tomados depende de los requerimientos de cada experimento. Por ejemplo, un experimento sobre Ethereum cuya duración esta asociada a la aparición de 1.000 bloques en la red se corresponde en Bitcoin a un experimento similar con una duración de 6,94 días.

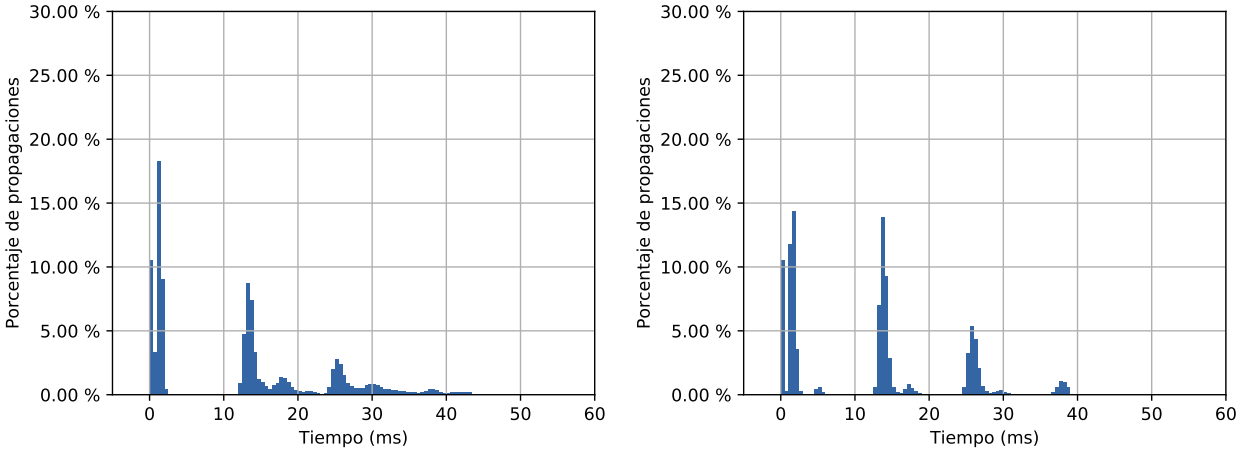
En esta sección analizaremos la similitud de los procesos de minado simulado y minado real teniendo en cuenta las métricas presentadas en la sección 5.4. En estos experimentos, la topología subyacente es la misma que la topología física, eliminando la capa de red emulada.

Los experimentos presentados en esta sección tienen las características presentadas en la tabla 6.1. Para cada valor de tiempo entre bloques se ejecutaron experimentos utilizando ambos modos de minado: Real y Simulado.

Dada la similitud de los experimentos para distintos targets, se incluirán a continuación todos los resultados para solo un valor de tiempo entre bloques: 21 segundos. Los resultados restantes se encuentran en el apéndice ubicado en A.

Métrica	Minado real	Minado simulado
Main Chain/Total minados	3800/3806	3800/3806
Tiempo entre bloques (promedio)	20,179 s	20,262 s
Forks profundidad 1	6	6
Forks profundidad >1	0	0
Competencias entre 1 bloques	3794	3794
Competencias entre 2 bloques	6	6
Competencias entre 3 bloques	0	0

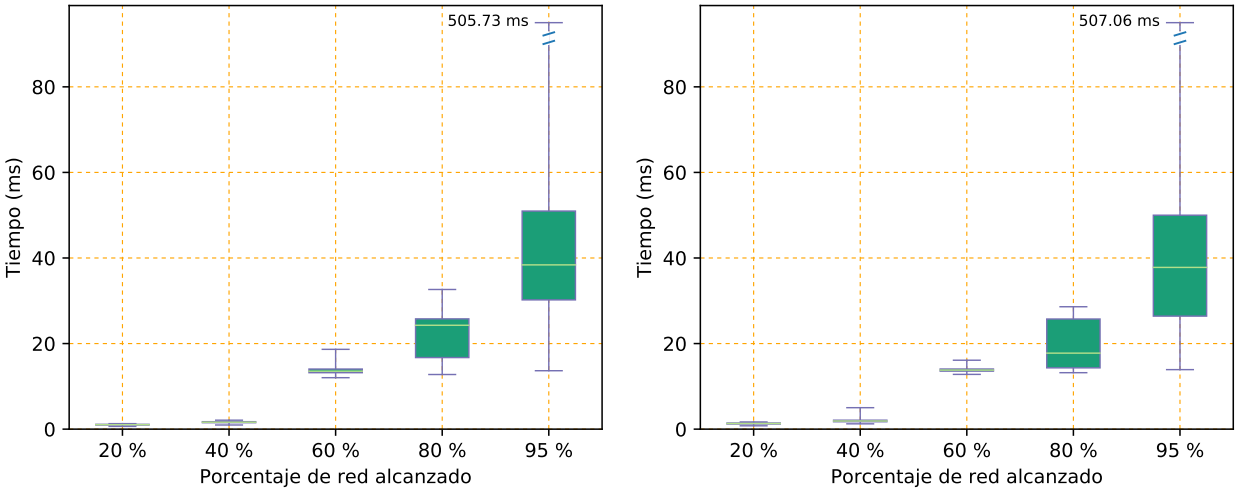
Cuadro 6.2: Métricas de minado - Tiempo entre bloques de 21s



(a) Minado real

(b) Minado simulado

Figura 6.1: Histograma de propagación de bloques - Tiempo entre bloques 21s



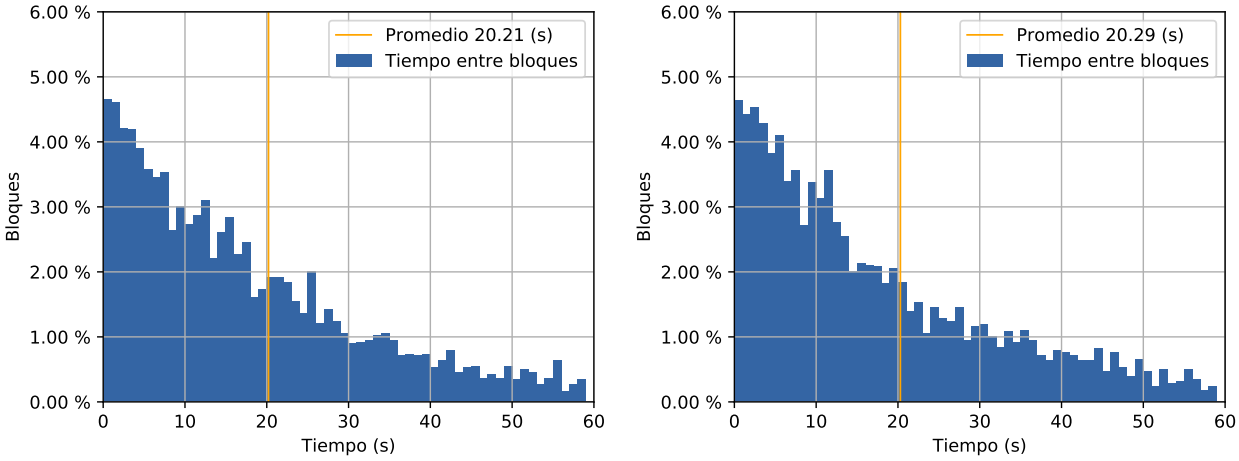
(a) Minado real

(b) Minado simulado

Figura 6.2: Alcance de propagación de bloques - Tiempo entre bloques 21s

Algoritmo de minado	Minero 1	Minero 2	Minero 3	Minero 4	Minero 5
Real	21,09 %	19,18 %	19,91 %	19,44 %	20,36 %
Simulado	20,38 %	18,86 %	20,46 %	19,81 %	20,46 %
Diferencia relativa	−3,48 %	−1,70 %	2,69 %	1,87 %	0,49 %

Cuadro 6.3: Distribución del Hashing Power - Tiempo entre bloques 21s



(a) Minado real

(b) Minado simulado

Figura 6.3: Histograma de tiempo entre bloques - Target 21s

Categoría de bloque	Minado real	Minado simulado
Stale	0,18 %	0,18 %
Cadena Principal	99,82 %	99,82 %

Cuadro 6.4: Rendimiento de la Blockchain - Tiempo entre bloques 21s

Como podemos ver en las tablas 6.2, 6.3 y 6.4 y en las figuras 6.1, 6.2 y 6.3, no se observaron diferencias significativas en ninguna de las métricas, lo cual nos hace concluir que ambos procesos de minado se comportan de manera similar.

6.4 Emulación de topologías de redes LAN

En esta sección analizaremos los resultados de utilizar Maxinet para emular una topología de una red local, incluyendo la asignación de valores de latencias en los enlaces virtuales que se corresponden con las medidas en la red física. Esta configuración está diseñada para llevar al límite la herramienta considerando lo mencionado en 2.2.2.

Los experimentos a continuación tienen las características presentadas en la tabla 6.5.

Configuración del experimento		
Característica	Ambiente real	Ambiente emulado
Topología física	Laboratorios DC	Ethernet/Museo
Topología emulada	Ninguna	Laboratorios DC
Topología lógica	Clique - 12 nodos	
Bloques analizados	4000	
% Mineros	50 % de la red distribuidos uniformemente	
Tiempos entre bloques	7, 14,5 y 21 segundos	
Proceso de minado	Simulado	

Cuadro 6.5: Configuración de los experimentos de emulación de redes LAN

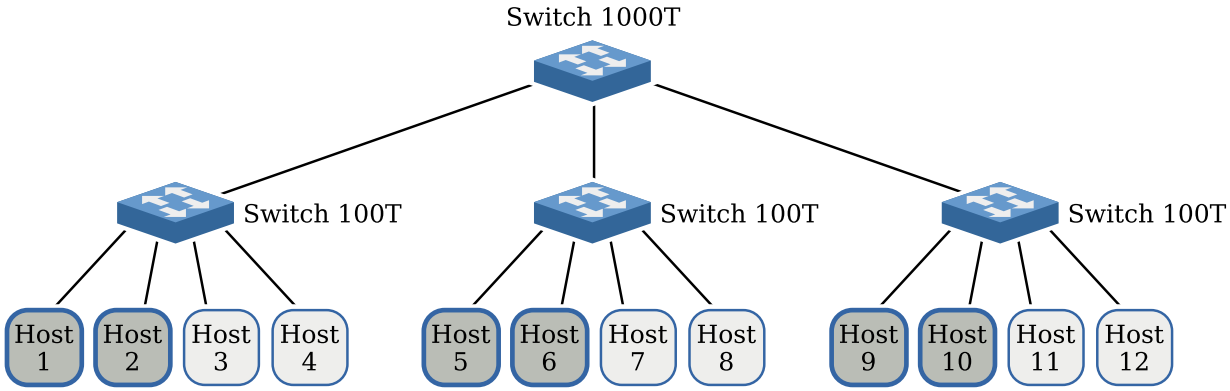


Figura 6.4: Topología laboratorios DC - 3 Laboratorios - 4 Máquinas por laboratorio. Los nodos mineros están distinguidos en gris.

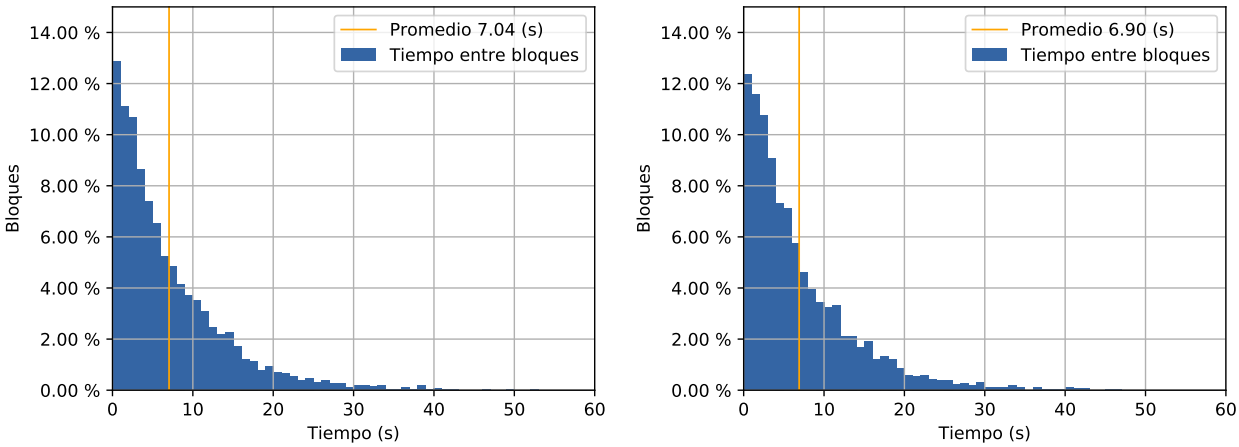
Se lanzaron experimentos sobre los dos ambientes antes detallados y se compararon los resultados. Nuevamente, los resultados de los experimentos son similares para los distintos tiempos entre bloques, por lo cual presentamos todos los resultados para un solo valor de target: 7 segundos. Para el resto de los resultados, referirse al apéndice, ubicado en A.

Métrica	Laboratorios DC	Maxinet
Main Chain/Total minados	4000/4026	4000/4040
Tiempo entre bloques (promedio)	6,996 s	6,837 s
Forks profundidad 1	25	39
Forks profundidad >1	0	0
Competencias entre 1 bloques	3975	3962
Competencias entre 2 bloques	25	37
Competencias entre 3 bloques	0	1

Cuadro 6.6: Métricas de minado - Tiempo entre bloques de 7s

Ambiente	Minero 1	Minero 2	Minero 3	Minero 4	Minero 5	Minero 6
Laboratorios DC	16,82 %	16,44 %	16,49 %	16,27 %	16,54 %	17,44 %
Maxinet	17,52 %	16,78 %	16,34 %	16,68 %	15,35 %	17,33 %
Diferencia relativa	4,00 %	2,03 %	−0,92 %	2,46 %	−7,75 %	−0,63 %

Cuadro 6.7: Distribución del Hashing Power - Tiempo entre bloques 7s

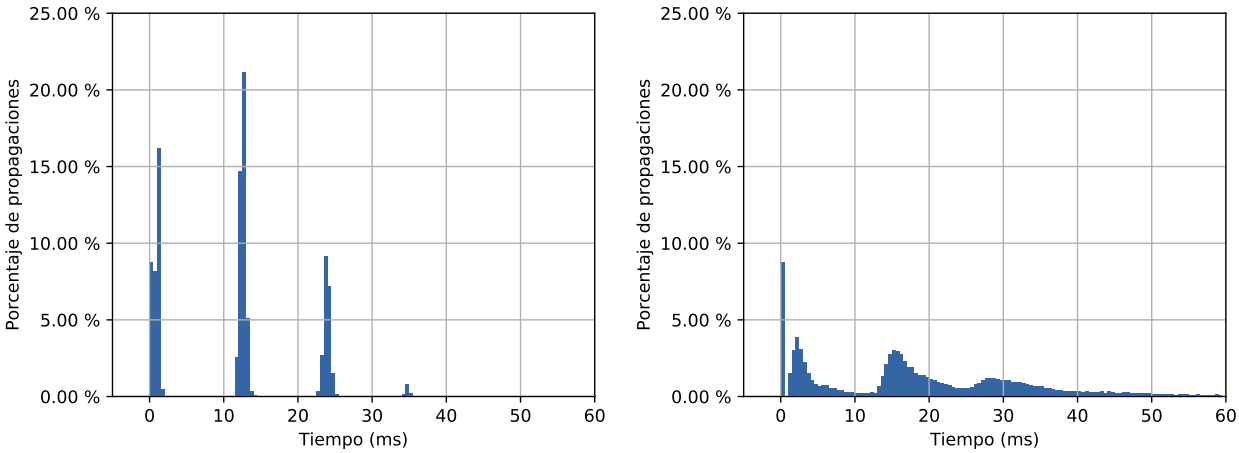


(a) Laboratorios DC (b) Maxinet

Figura 6.5: Histograma de tiempo entre bloques - Target 7s

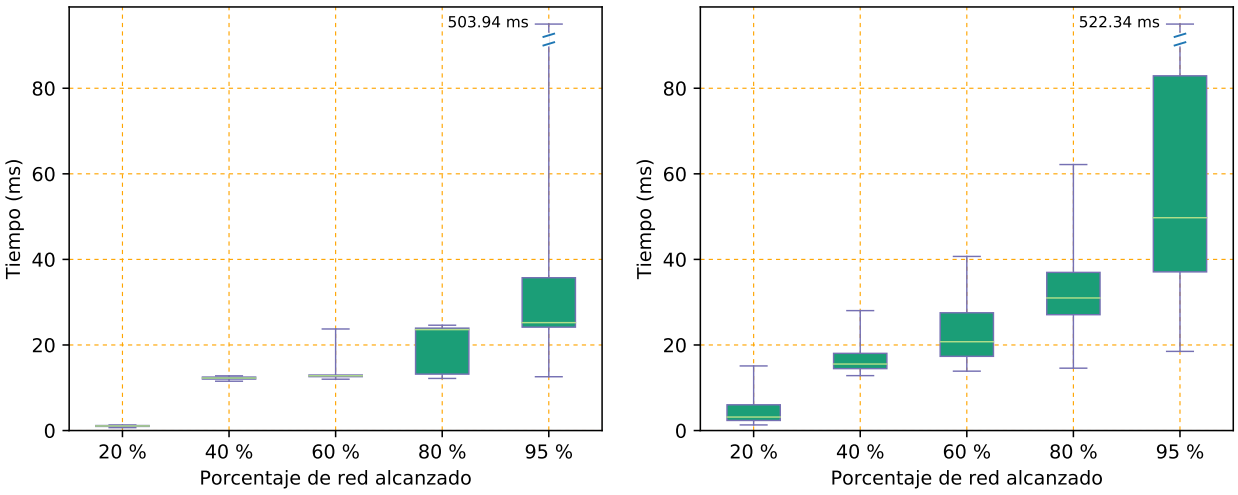
Tipo de minado	Laboratorios DC	Maxinet
Stale	0,65 %	0,99 %
Cadena Principal	99,35 %	99,01 %

Cuadro 6.8: Rendimiento de la Blockchain - Tiempo entre bloques 7s



(a) Laboratorios DC (b) Maxinet Topología Laboratorios DC

Figura 6.6: Histograma de propagación de bloques - Tiempo entre bloques 7s



(a) Laboratorios DC (b) Maxinet

Figura 6.7: Alcance de propagación de bloques - Tiempo entre bloques 7s

En la figura 6.5 y la tabla 6.7 no se registran diferencias significativas.

Respecto al resto de las métricas, como mencionamos en 2.2.2, al realizar emulaciones de latencias cercanas a la red física subyacente, se observa una alta varianza en los enlaces de la red emulada.

Este fenómeno produce diferencias en los tiempos de propagación respecto a la red como indican las figuras 6.6 y 6.7. Por otro lado, una alta varianza en los tiempos de propagación, puede generar mayor competitividad entre bloques al momento de los broadcasts, aumentando levemente la cantidad de forks que ocurren en la red como indica la tabla 6.6. Esto último también afectará la cantidad de bloques stale en la red, fenómeno que se aprecia en la tabla 6.8, donde hay un leve incremento.

Los gráficos de propagación en la figura 6.6 se asemejan a una suma de distribuciones normales con diferentes valores de media. En el ambiente emulado, cada una de estas aparenta tener mayor varianza que su contraparte en el ambiente real. Para poder explicar este fenómeno, realizamos un análisis más exhaustivo que se detalla a continuación.

Impacto de la varianza en las propagaciones

Los tiempos efectivos de propagación de un bloque están estrechamente relacionados con las propiedades de los enlaces de la red subyacente. Si consideramos que en dicha red, los enlaces tienen un valor de jitter bajo, las mediciones de tiempo de varias propagaciones no deberían alejarse de su valor promedio. En otro caso, si la varianza en la latencia fuera alta, los tiempos de propagación de varios envíos de bloques tendrían mayores diferencias entre sí. Este efecto puede verse en las figuras 6.8 y 6.9, donde en una red con baja varianza en los enlaces ambos gráficos se superponen completamente, ya que los tiempos efectivos de propagación se alejan poco del promedio. Por otro lado, en redes de enlaces con un jitter mayor, los gráficos muestran patrones similares, pero con leves diferencias entre ellos.

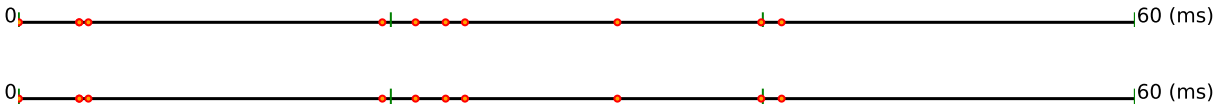


Figura 6.8: Propagaciones de dos bloques en red con baja varianza

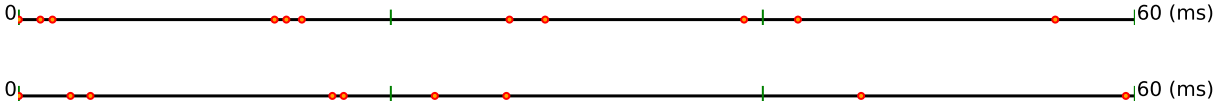
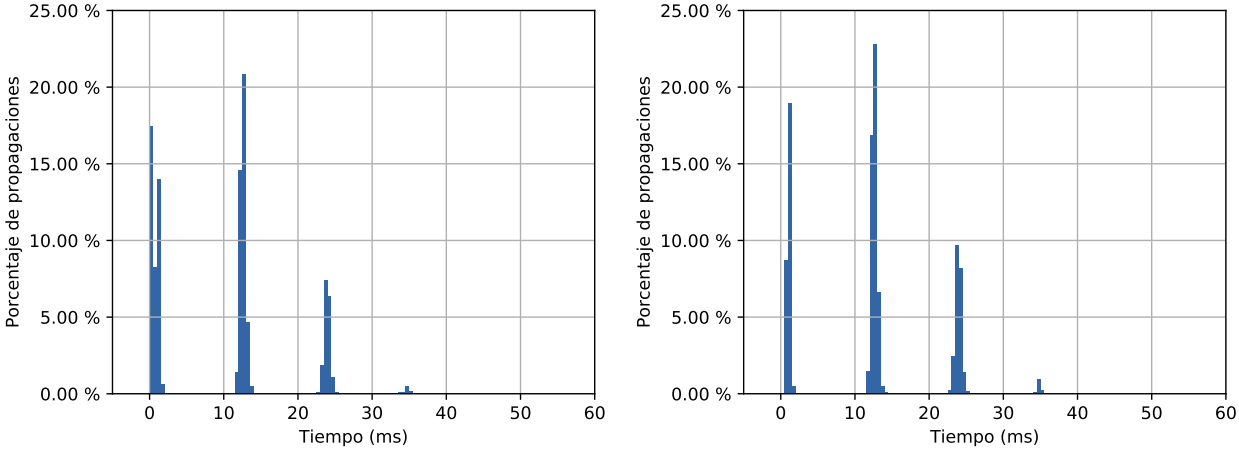


Figura 6.9: Propagaciones de dos bloques en red con alta varianza

Si observamos los histogramas presentados en las figuras 6.10 y 6.11, pueden apreciarse los efectos de varios fenómenos ocurriendo en la red:

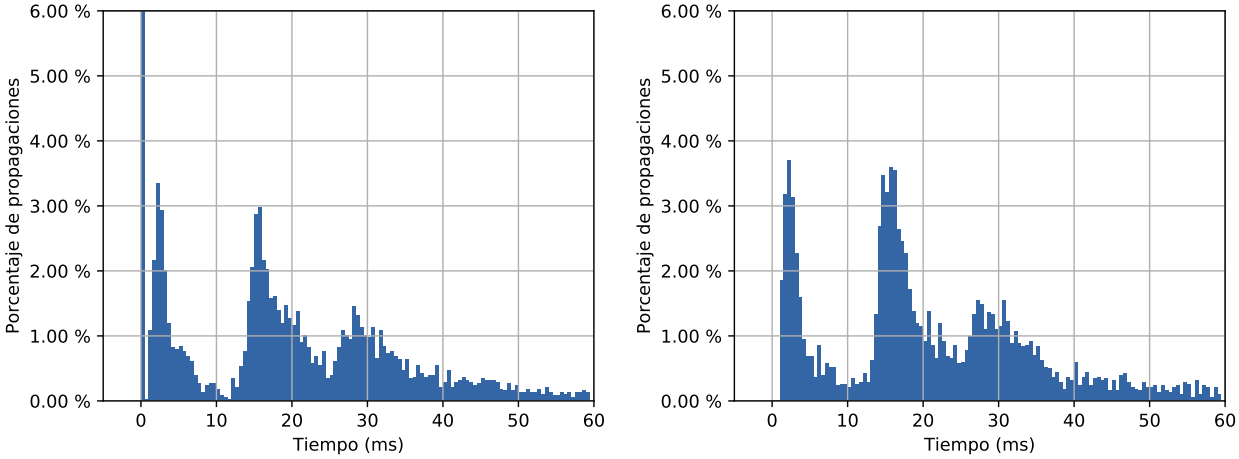
1. En caso de ser un nodo minero, aparece su porcentaje de hashing power como propagaciones con tiempo cero, lo cual se corresponde con los bloques minados por este nodo.

2. El resto de los tiempos se corresponden con llegadas de bloques por caminos de diversas características en el árbol de propagación.



(a) Nodo minero (b) Nodo pasivo

Figura 6.10: Histograma de propagación visto desde un nodo - Baja varianza en latencia

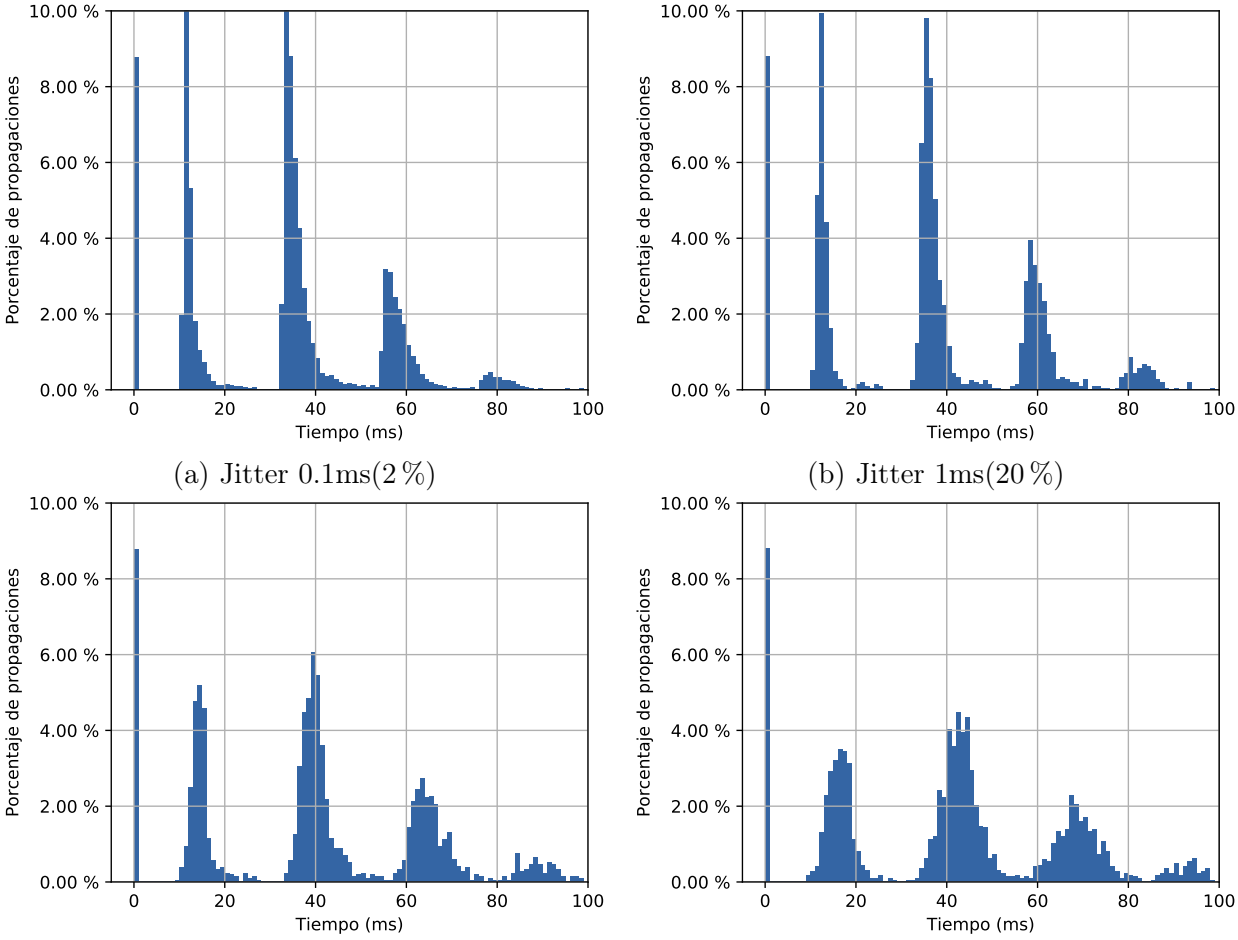


(a) Nodo minero (b) Nodo pasivo

Figura 6.11: Histograma de propagación visto desde un nodo - Alta varianza en latencia

Dado que los histogramas presentados en la figura 6.6 no son más que una combinación de todos los gráficos por nodos en la red, los fenómenos que mencionamos anteriormente se reflejan en ellos.

Para corroborar empíricamente estas hipótesis, realizamos experimentos manteniendo una latencia constante en los enlaces cuyo valor estuviera dos órdenes de magnitud por encima de los valores de latencia de los enlaces físicos, de forma tal que Maxinet funcionara de forma estable. En estos últimos experimentos, incrementamos gradualmente el jitter de los enlaces. Las figuras presentadas en 6.12 muestran una clara relación entre los valores de varianza de las distribuciones y el jitter configurado en los enlaces de la red emulada subyacente.



(a) Jitter 0.1ms(2 %)

(b) Jitter 1ms(20 %)

(c) Jitter 2.5ms(50 %)

(d) Jitter 4ms(80 %)

Figura 6.12: Experimentos de propagación sobre Maxinet: Link delay 5ms - Distintos jitters

Estos resultados nos indican que no es posible emular topologías de redes locales, pues tendremos interferencias en nuestras mediciones producidas por la herramienta de emulación de redes. Dado que el resto del trabajo propone escenarios de redes en Internet, cuyos datasets de latencias tienen valores mínimos dos órdenes de magnitud mayores que las latencias registradas en nuestra plataforma experimental, estos fenómenos no influirán en nuestros resultados.

Capítulo 7

Resultados

En este capítulo presentaremos los resultados de los experimentos realizados sobre Ethereum. Estos se dividen en:

1. Impacto de la disminución del tiempo entre bloques y aumento del diámetro de la red.
2. Análisis de los tiempos de propagación.
3. Relación entre el diámetro de la red y los tipos de transmisiones.

Para todos los experimentos de este capítulo se utilizó una misma topología lógica base generada aleatoriamente de 60 nodos. Cada nodo tiene un país asignado y 2 o 3 peers.

7.1 Variaciones de target y diámetro de la topología de red

A continuación presentamos experimentos que llevan al límite dos variables: El tiempo entre bloques y el diámetro[1] de la topología lógica.

Nuestras hipótesis establecen que al disminuir el tiempo entre bloques, ocurrirán más hazards entre mineros, mientras que al aumentar el diámetro de la red, los tiempos de propagación de los bloques irán en aumento, produciendo colisiones entre mineros más frecuentemente.

Esperamos, entonces, ver un incremento en las métricas de forks de la red a medida que se disminuye el tiempo entre bloques y/o se aumenta el diámetro de la topología lógica[2]. Los cambios en los forks deberían verse tanto en profundidad como en cantidad de bloques competitivos.

Para estos experimentos, la asignación de mineros en la topología lógica fue una elección aleatoria del 50 % de los nodos de la red, todos ellos con igual hashing power. Se ejecutaron experimentos variando el tiempo entre bloques entre {2, 4, 6, 8, 10} segundos y para el análisis de todos los experimentos se consideran los primeros 1000 bloques de la red.

En las próximas secciones presentamos los resultados para las diferentes combinaciones de diámetros y tiempo entre bloques.

Dado que las modificaciones realizadas sobre el tiempo entre bloques no tienen impacto en los tiempos de propagación, las propagaciones a diversos porcentajes de la red muestran ser casi idénticas para todos los tiempos entre bloques dentro de una misma topología emulada. La única variación significativa en estas métricas se da al incrementar la latencia de los enlaces de las topologías. Es por esto que presentamos solo a modo informativo estos gráficos en las figuras 7.2, 7.4 y 7.7.

[1] https://en.wikipedia.org/wiki/Distance_(graph_theory)

[2] Dado que nuestro podér de cómputo es limitado, el efecto de aumento de diámetro es simulado aumentando las latencias en los enlaces de la red emulada. De esta forma aumentamos la latencia de extremo a extremo en la red lógica como proxy para simular un incremento en el diámetro (cantidad de saltos).

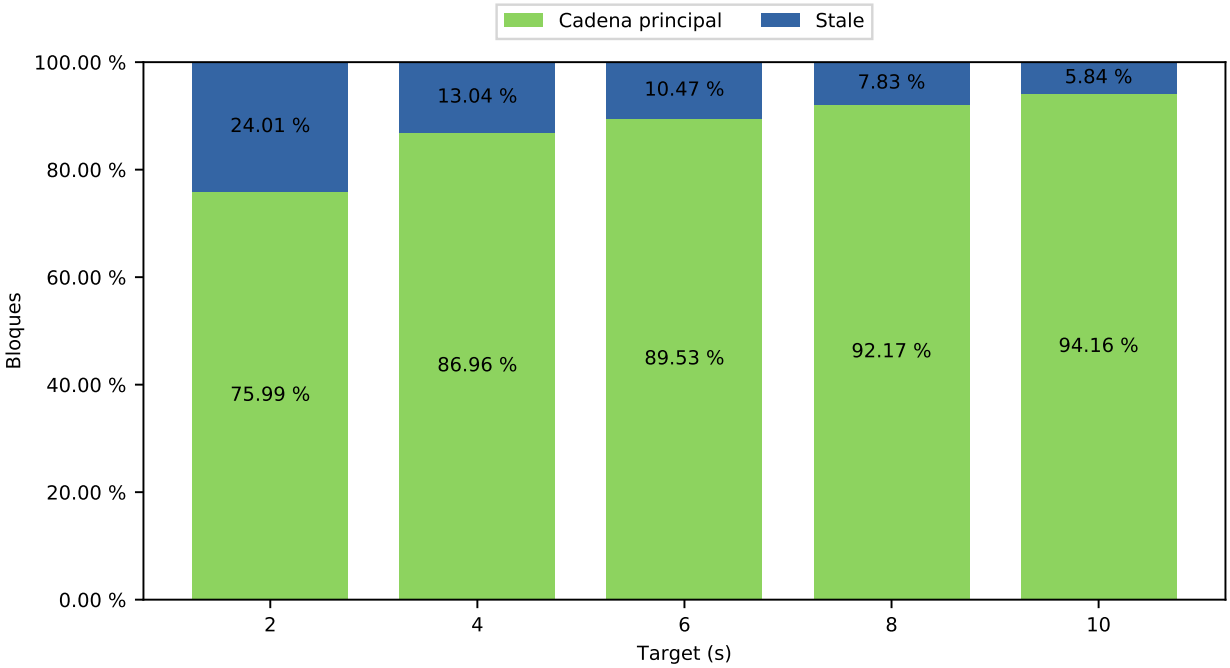


Figura 7.1: Rendimiento de la blockchain - Topología emulada con latencias reales.

Tiempo de Generación de Bloques			
Target (segs)	Media (ms)	Mediana (ms)	Main Chain/Total minados
10	10222,91	6997,97	1000/1062
8	7632,04	5350,87	1000/1085
6	6175,54	4250,20	1000/1117
4	4027,37	2888,26	1000/1150
2	1990,88	1450,09	1000/1316

Cuadro 7.1: Estadísticas de generación de bloques - Topología emulada con latencias reales.

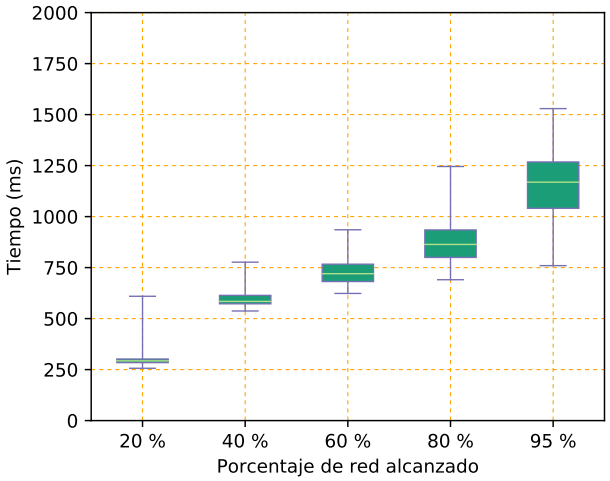
7.1.1 Latencia real

Respecto a la aparición de forks en la red, los resultados muestran que, efectivamente, al disminuir el tiempo entre bloques se observa un incremento en los forks en la red.

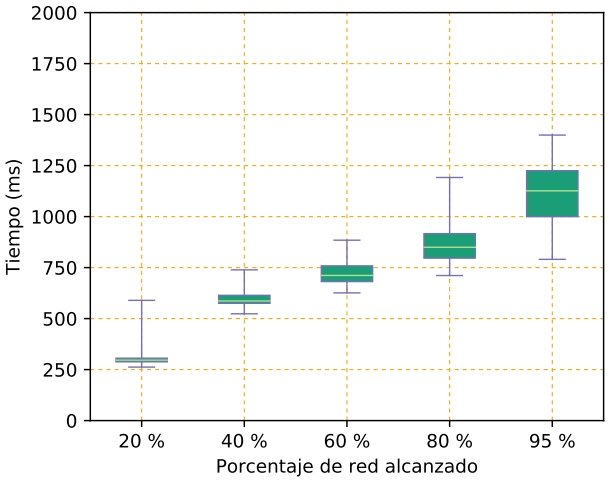
En la tabla 7.3 podemos ver qué porcentajes de la cadena principal participaron en competencias entre bloques[3]. Los resultados acerca de la profundidad de los forks pueden verse en la tabla 7.2.

Al disminuir el tiempo entre bloques, observamos cómo las competencias entre bloques

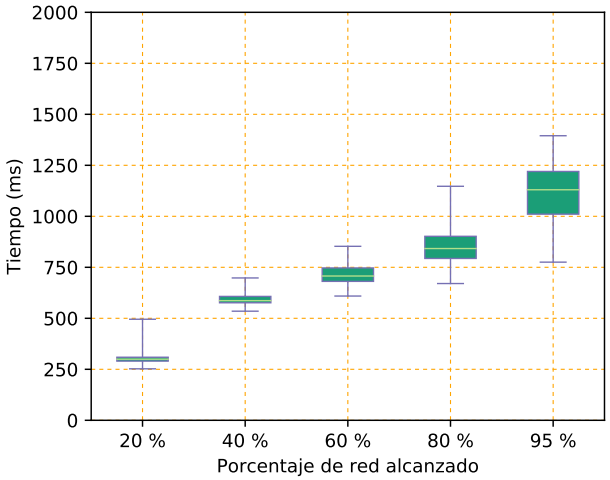
[3]Por ejemplo: En la fila 4, columna 4: el 1 % de los bloques de la cadena principal tuvieron lugar luego de una competencia entre 3 bloques.



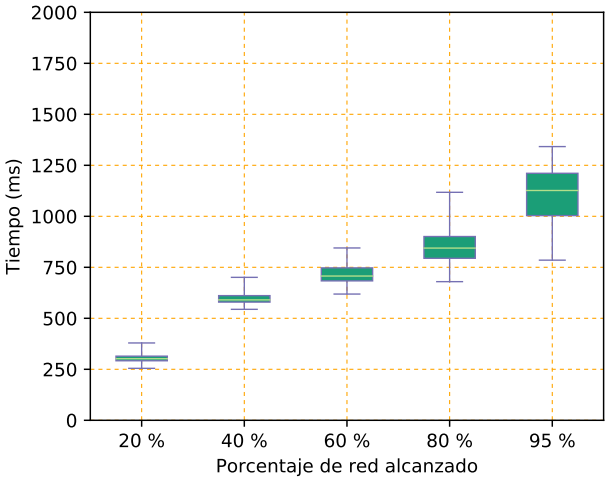
(a) Target 2s



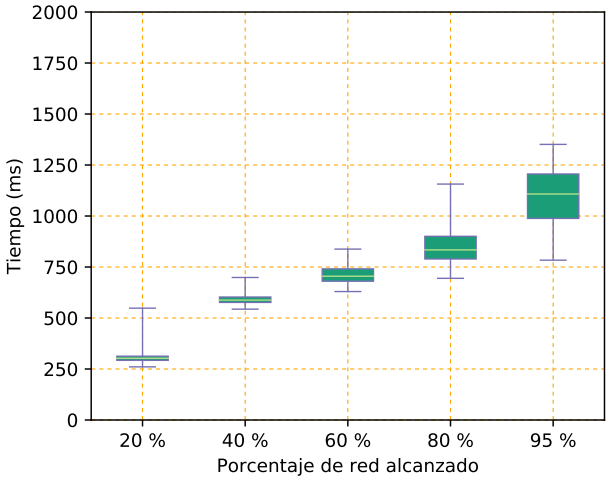
(b) Target 4s



(c) Target 6s



(d) Target 8s



(e) Target 10s

Figura 7.2: Alcance de propagación de bloques - Topología emulada con latencias reales.

Profundidad de forks				
Target (segs)	Prof = 1	Prof = 2	Prof = 3	Prof >3
10	57	2	0	0
8	84	0	0	0
6	102	7	0	0
4	127	11	0	0
2	234	31	6	1

Confiabilidad
Alta
Media
Baja

Cuadro 7.2: Estadísticas de profundidad forks - Mapa de color de la confiabilidad de la Blockchain - Topología emulada con latencias reales.

Cantidad de Bloques Competitivos					
Target (segs)	1 bloque	2 bloques	3 bloques	4 bloques	>4 bloques
10	94,10 %	5,70 %	0,20 %	0,00 %	0,00 %
8	91,70 %	8,20 %	0,10 %	0,00 %	0,00 %
6	89,00 %	10,40 %	0,60 %	0,00 %	0,00 %
4	86,10 %	12,90 %	1,00 %	0,00 %	0,00 %
2	72,20 %	24,50 %	2,90 %	0,40 %	0,00 %

Cuadro 7.3: Estadísticas de ancho de forks - Topología emulada con latencias reales.

se acentúan produciendo que menos bloques minados ingresen a la cadena principal sin competencias con otros bloques minados «a la vez». También existe una tendencia a la aparición de forks de mayor profundidad.

Estos fenómenos inciden en la confiabilidad y el rendimiento de la red. En la tabla 7.1 y en la figura 7.1 puede verse la evolución del rendimiento de la blockchain en función del tiempo entre bloques. Vemos que en el caso más extremo, con un tiempo entre bloques establecido en 2 segundos, un 24 % de los bloques minados fueron desechados por no tener lugar en la cadena principal. Según nuestras métricas en la tabla 7.2, a partir de un tiempo entre bloques de 2 segundos, ocurren forks con profundidad mayor o igual que 3, volviendo a la red poco confiable para realizar transacciones.

7.1.2 Latencia 2x

En las tablas 7.5 y 7.6 observamos las mismas tendencias que en la topología anterior, pero con un leve incremento en la cantidad total de forks. Observando la figura 7.4 atribuimos este fenómeno al mayor tiempo requerido para que un bloque se difunda a toda la red, el

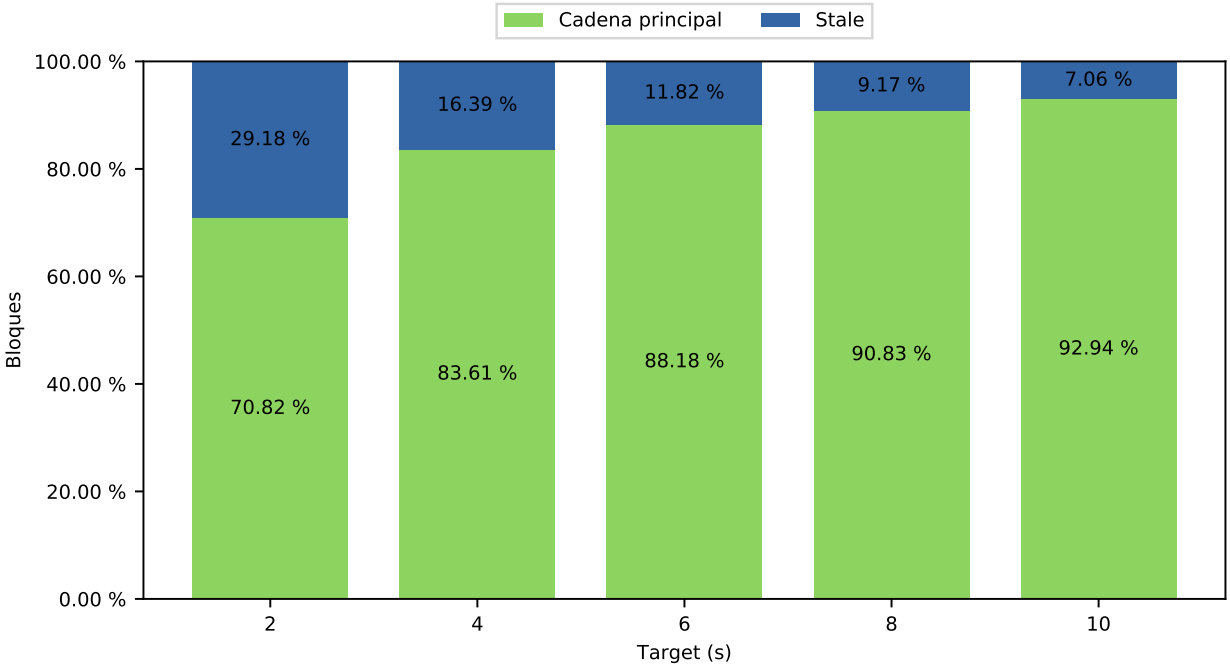


Figura 7.3: Rendimiento de la blockchain - Topología emulada con latencias duplicadas.

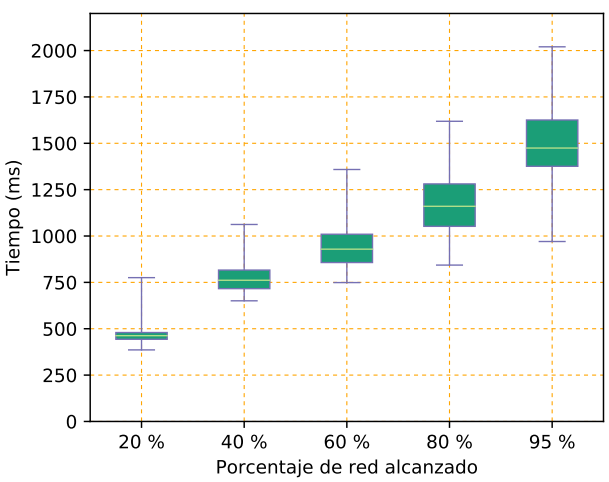
Tiempo de Generación de Bloques			
Target (segs)	Media (ms)	Mediana (ms)	Main Chain/Total minados
10	10298,55	7287,38	1000/1075
8	8184,43	5277,38	1000/1101
6	5982,86	4315,50	1000/1133
4	4162,84	2953,27	1000/1196
2	1981,34	1356,36	1000/1412

Cuadro 7.4: Estadísticas de generación de bloques - Topología emulada con latencias duplicadas.

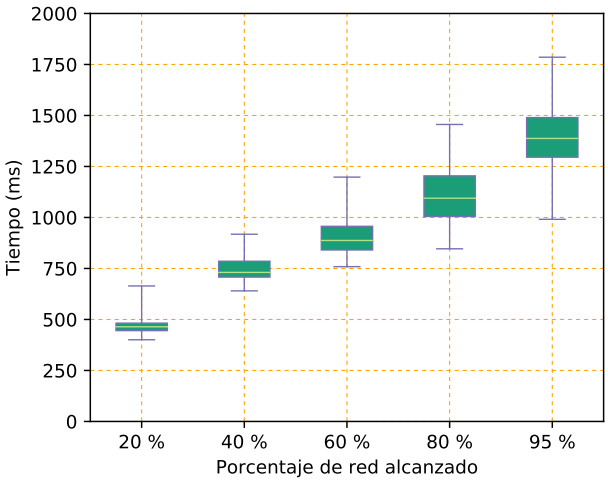
Profundidad de forks				
Target (segs)	Prof = 1	Prof = 2	Prof = 3	Prof >3
10	69	3	0	0
8	90	5	0	0
6	121	6	0	0
4	165	12	2	0
2	281	57	6	3

Confiabilidad
Alta
Media
Baja

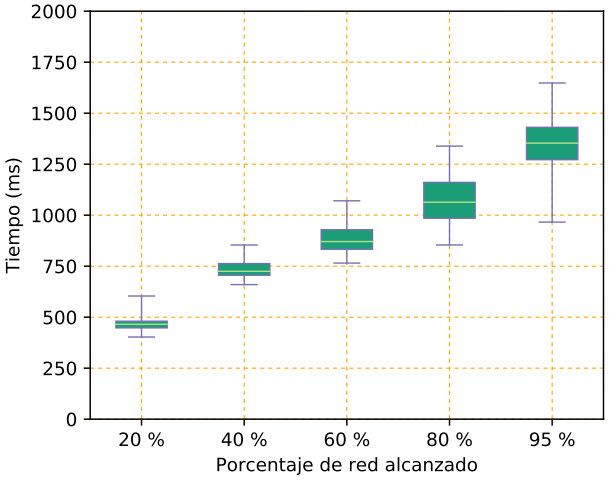
Cuadro 7.5: Estadísticas de profundidad forks - Mapa de color de la confiabilidad de la Blockchain - Topología emulada con latencias duplicadas.



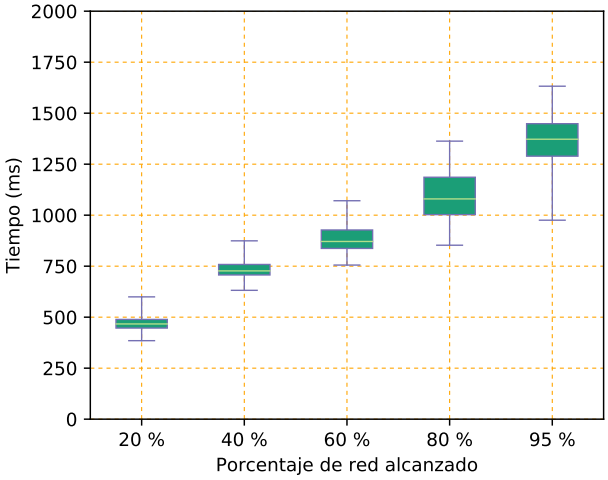
(a) Target 2s



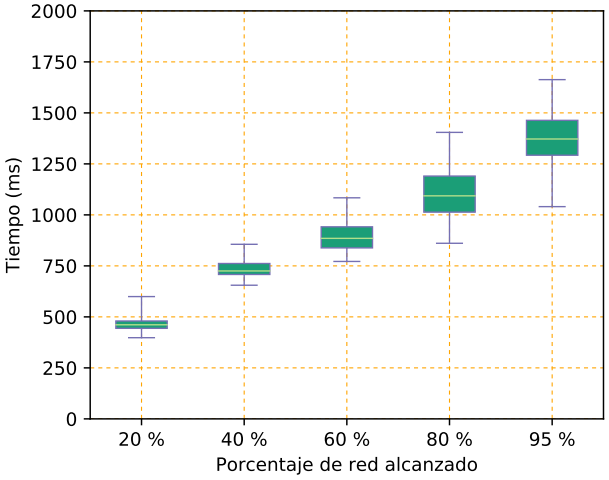
(b) Target 4s



(c) Target 6s



(d) Target 8s



(e) Target 10s

Figura 7.4: Alcance de propagación de bloques - Topología emulada con latencias duplicadas.

Cantidad de Bloques Competitivos					
Target (segs)	1 bloque	2 bloques	3 bloques	4 bloques	>4 bloques
10	92,60 %	7,30 %	0,10 %	0,00 %	0,00 %
8	90,40 %	9,20 %	0,40 %	0,00 %	0,00 %
6	87,50 %	11,70 %	0,80 %	0,00 %	0,00 %
4	82,60 %	15,40 %	1,90 %	0,10 %	0,00 %
2	65,50 %	28,30 %	5,80 %	0,40 %	0,00 %

Cuadro 7.6: Estadísticas de ancho de forks - Topología emulada con latencias reales.

cual tiene una relación directa con la probabilidad de hazards entre los mineros.

En términos de confiabilidad, observamos en la tabla 7.5 que ocurren forks de profundidad mayor o igual a 3 para un tiempo entre bloques de 4 segundos.

Por último, al observar el rendimiento de la red en la figura 7.3, el mismo se ve disminuido. La cantidad de bloques que quedan stale fue mayor que en los experimentos anteriores.

7.1.3 Latencia 200 ms

Para finalizar, presentamos una topología emulada con una latencia de 200 ms en cada enlace. Este valor, bastante mayor respecto a los utilizados hasta el momento, presenta un escenario extremo para analizar el impacto en el rendimiento y la confiabilidad de la blockchain.

Notemos en las figuras 7.7 que los tiempos de propagación a toda la red corresponden aproximadamente a 2,5 segundos pero esto aumenta para los valores de tiempo entre bloques de 2 y 4 segundos, probablemente por el aumento de tráfico en la red.

Al igual que en los experimentos anteriores, vemos en las figuras 7.9 y 7.10 que las tendencias de aumento en los forks de la red se mantienen pero sus valores son bastante más significativos que en topologías anteriores. Por otro lado, la figura 7.5 muestra una disminución pronunciada en el rendimiento de la red respecto al tiempo entre bloques, mostrando para un valor de 2 segundos, que el 40,97 % de los bloques minados terminaron siendo stale. Desde el punto de vista de confiabilidad de la red, ocurren forks de profundidad mayor o igual a 3 a partir de un tiempo entre bloques de 8 segundos, lo cual hace inviable el uso de

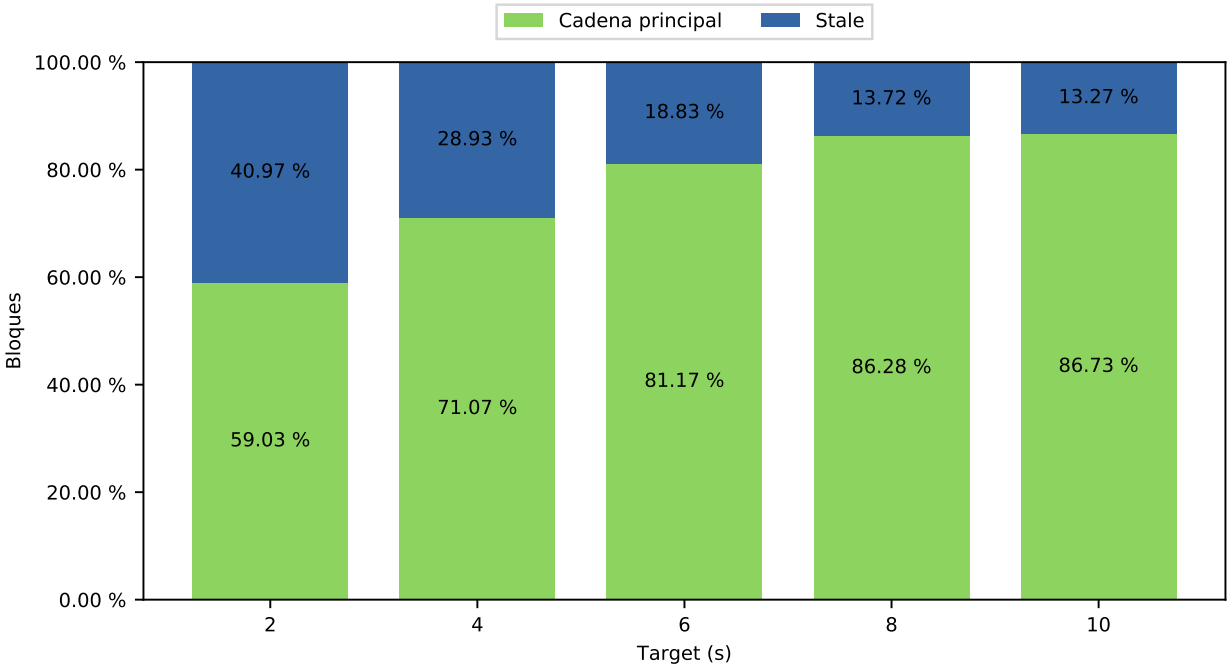
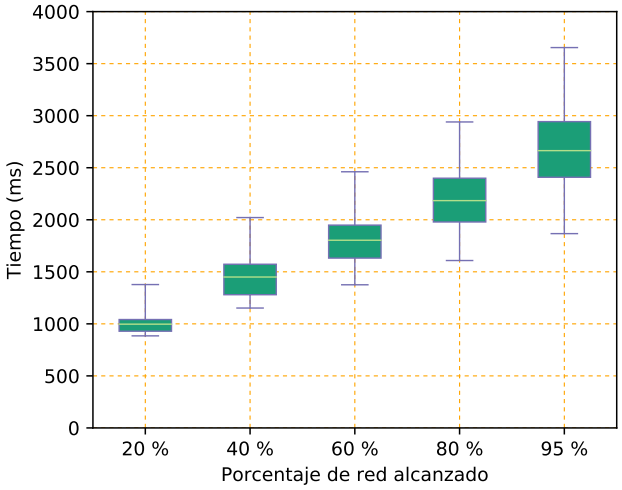
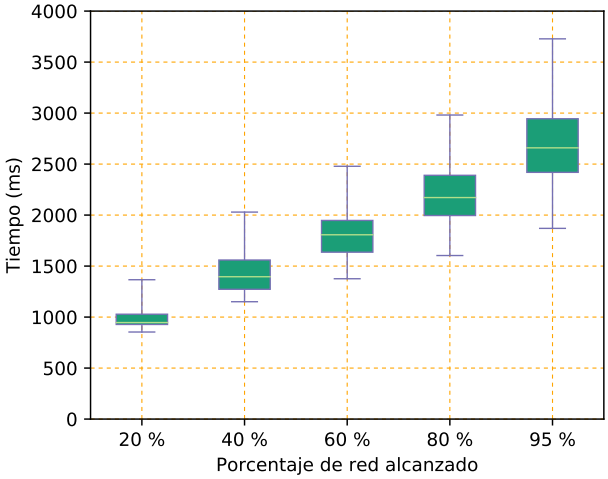


Figura 7.5: Rendimiento de la blockchain - Topología emulada con latencias de 200 ms.

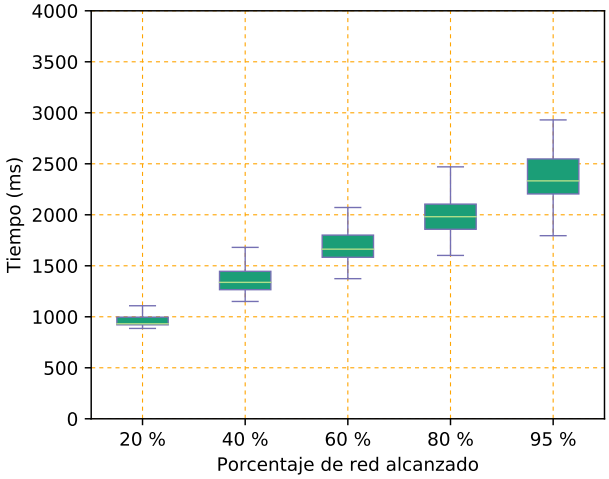
estas configuraciones para topologías de estas características.



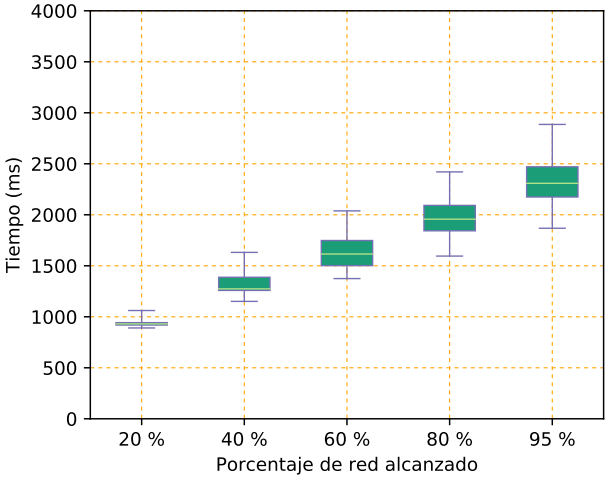
(a) Target 2s



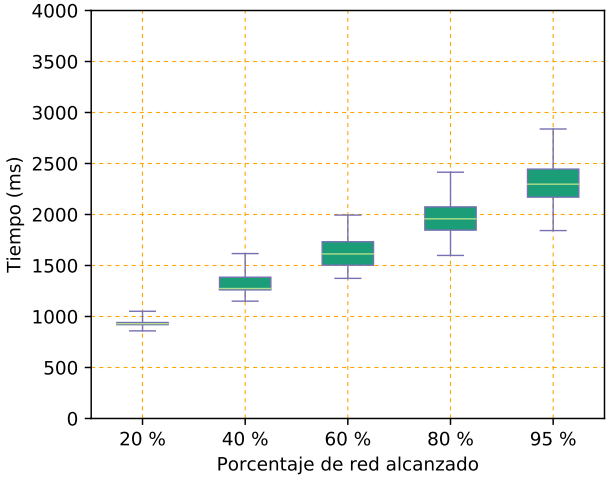
(b) Target 4s



(c) Target 6s



(d) Target 8s



(e) Target 10s

Cuadro 7.7: Alcance de propagación de bloques - Topología emulada con latencias de 200 ms.

Tiempo de Generación de Bloques			
Target (segs)	Media (ms)	Mediana (ms)	Main Chain/Total minados
10	9540,33	6454,65	1000/1153
8	7440,05	5339,37	1000/1159
6	5672,57	3966,35	1000/1232
4	3907,92	2887,53	1000/1406
2	2093,77	1419,62	1000/1694

Cuadro 7.8: Estadísticas de generación de bloques - Topología emulada con latencias de 200 ms.

Profundidad de forks				
Target (segs)	Prof = 1	Prof = 2	Prof = 3	Prof >3
10	135	9	0	0
8	140	5	2	0
6	189	21	1	0
4	276	38	14	2
2	393	97	30	9

Confiabilidad
Alta
Media
Baja

Cuadro 7.9: Estadísticas de profundidad forks - Mapa de color de la confiabilidad de la Blockchain - Topología emulada con latencias de 200 ms.

Cantidad de Bloques Competitivos					
Target (segs)	1 bloque	2 bloques	3 bloques	4 bloques	>4 bloques
10	86,00 %	12,80 %	1,20 %	0,00 %	0,00 %
8	85,40 %	13,40 %	1,20 %	0,00 %	0,00 %
6	79,30 %	18,40 %	2,20 %	0,10 %	0,00 %
4	65,50 %	29,90 %	4,60 %	0,50 %	0,00 %
2	47,20 %	39,40 %	10,70 %	2,30 %	0,40 %

Cuadro 7.10: Estadísticas de ancho de forks - Topología emulada con latencias de 200 ms.

7.1.4 Discusión

En las secciones anteriores observamos que existe una relación entre el target y la aparición de forks en la red. Mientras más corto sea el tiempo entre bloques, mayor cantidad de forks ocurrirán, afectando el rendimiento y la confiabilidad de la red. Teniendo en cuenta los resultados en los tres ambientes de experimentación podemos concluir que ante la elección de un tiempo entre bloques, para tener una red blockchain estable debe tenerse en cuenta el diámetro de la red que se necesite diseñar, ya que esto influye directamente sobre los tiempos de propagación, alterando el rendimiento y la confiabilidad de la Blockchain.

7.2 Análisis de tiempos de propagación

Este análisis se realiza con el objetivo de explicar el tipo de distribución que se observa en el histograma de tiempos de propagación global para todo un experimento. Mediante un análisis exhaustivo de los árboles de propagación pudimos caracterizar ciertos aspectos del protocolo de propagación y sus distribuciones asociadas.

La topología de estos experimentos es la misma que la utilizada en el capítulo 7.1.1. Nuevamente, se consideran los primeros 1000 bloques de la red. El tiempo entre bloques no tiene un impacto significativo en los tiempos de propagación. Dicho esto consideramos un valor de target de 10 segundos.

7.2.1 Análisis del contexto de las propagaciones

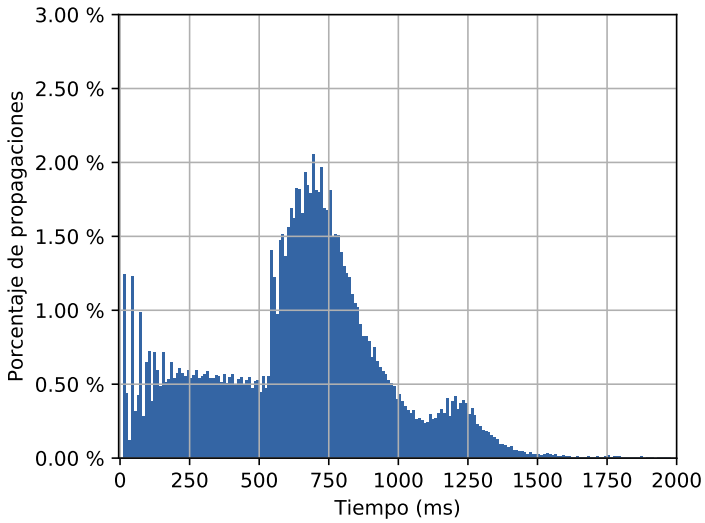


Figura 7.6: Distribución de propagación para todos los tipos de camino - Todos los niveles

La figura 7.6 nos muestra el histograma global de tiempos de propagación. Es decir, todas las propagaciones de bloques a lo largo de todo el experimento. Dicho gráfico se asemeja a una suma de varias distribuciones.

Dada la complejidad del algoritmo de propagación de Ethereum, tuvimos en cuenta el contexto en el que ocurren dichas propagaciones. Por ejemplo, mediante el análisis por niveles del árbol de propagación de un bloque podemos obtener los tiempos de propaga-

ción para caminos de longitud fija. Esto último es importante porque creemos que al fijar esta variable estaremos considerando un solo un subconjunto de distribuciones similares. Análogamente podemos tener en consideración los tipos de caminos y aristas sobre el árbol y de esta forma analizar la heurística de envío de hashes.

Formalmente, presentamos la siguiente partición en dos variables del dominio de propagaciones:

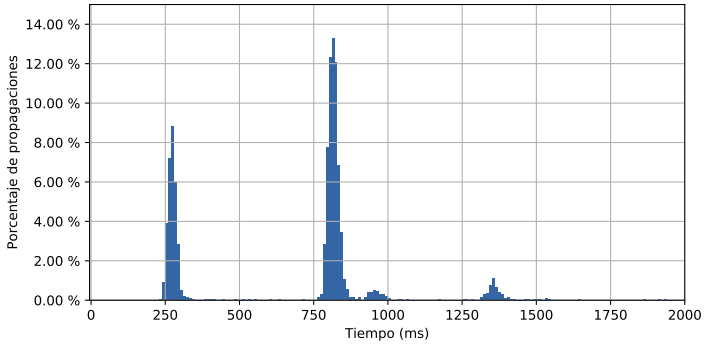
- Nivel en el árbol de propagación: $\{1, 2, \ldots, altura(árbol)\}$
- Tipo de camino: {Combinado, transmisión completa, transmisión diferida}

Las figuras en 7.7 indican la partición de los histogramas de propagación por nivel en el árbol. En ellas se observan intervalos con varias distribuciones de distintos tipos de propagaciones que ocurren durante el experimento, al incrementar los niveles, se produce un corrimiento en el tiempo efectivo de propagación, como es esperable ya que la transmisión del bloque tuvo más saltos.

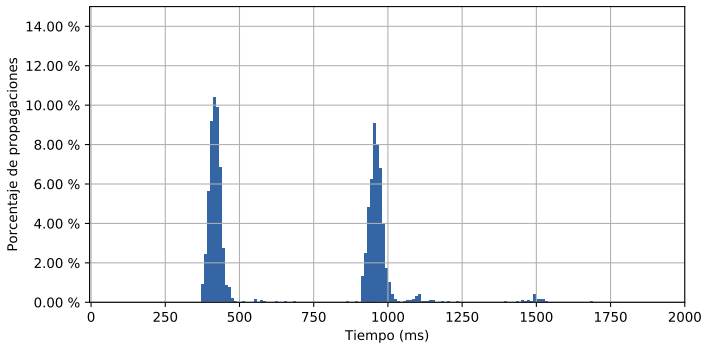
Los resultados alrededor del primer intervalo en los gráficos de todos los tipos de caminos están relacionados con las transmisiones de camino completo, ya que estas son las más rápidas en cuanto al tiempo efectivo de propagación. El resto de las distribuciones están relacionadas con los caminos combinados y a los caminos de transmisión diferida. Recordemos que el protocolo de Ethereum establece que las descargas programadas de bloques ocurran cada 500 milisegundos, efecto que puede apreciarse en los gráficos.

7.2.2 Distribución de las transmisiones de bloque completo

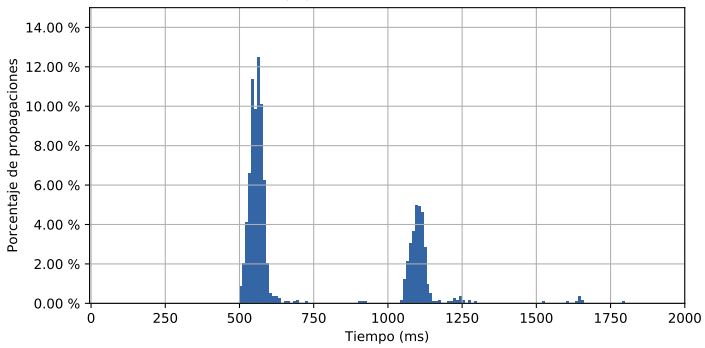
A continuación mostramos en las figuras ubicadas en 7.8 la evolución de estas distribuciones a medida que aumentamos el nivel dentro del árbol de propagación. Realizamos para todas estas mediciones de camino completo, un test de normalidad de Shapiro-Wilk que arrojó resultados positivos con precisiones mayores al 95 %. Esto último nos indica que los tiempos de propagación para caminos de bloque completo de igual longitud siguen distribuciones normales.



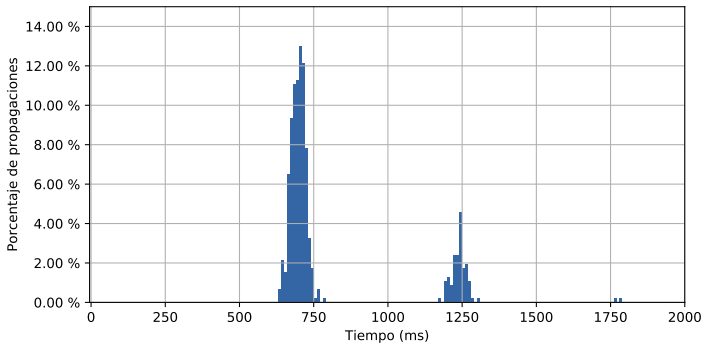
(a) Nivel 10



(b) Nivel 15

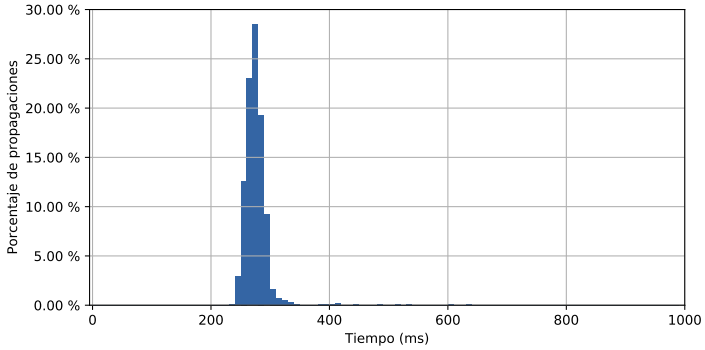


(c) Nivel 20

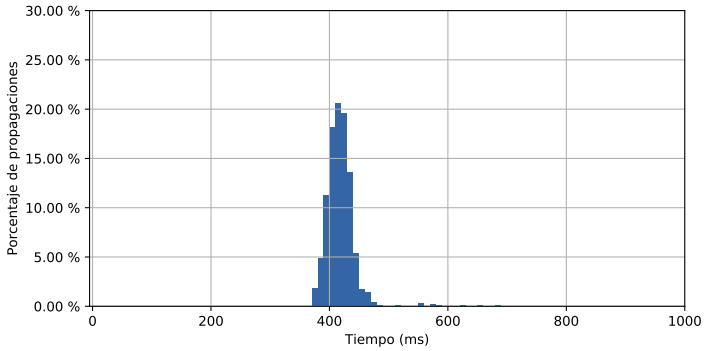


(d) Nivel 25

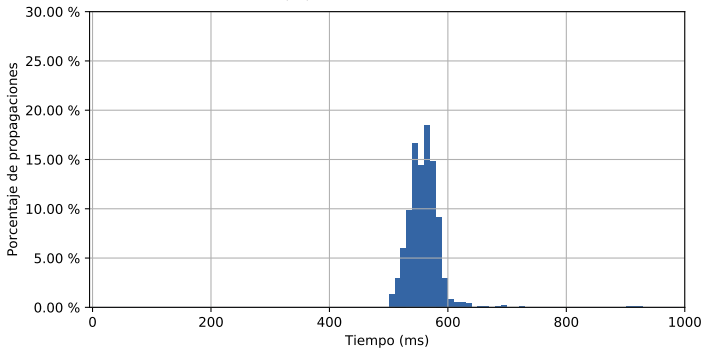
Figura 7.7: Distribución de propagación para todos los tipos de camino por nivel



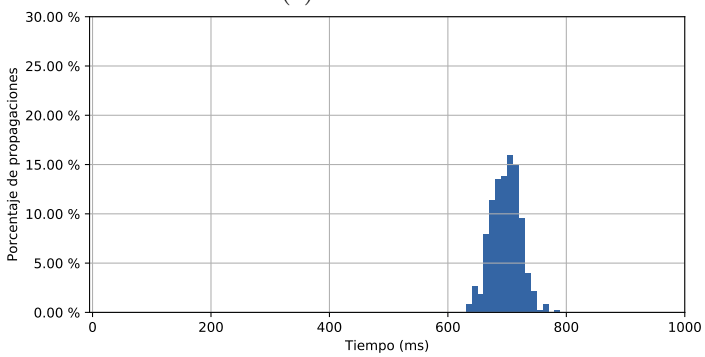
(a) Nivel 10



(b) Nivel 15



(c) Nivel 20



(d) Nivel 25

Figura 7.8: Distribución de propagación de transmisión completa por niveles

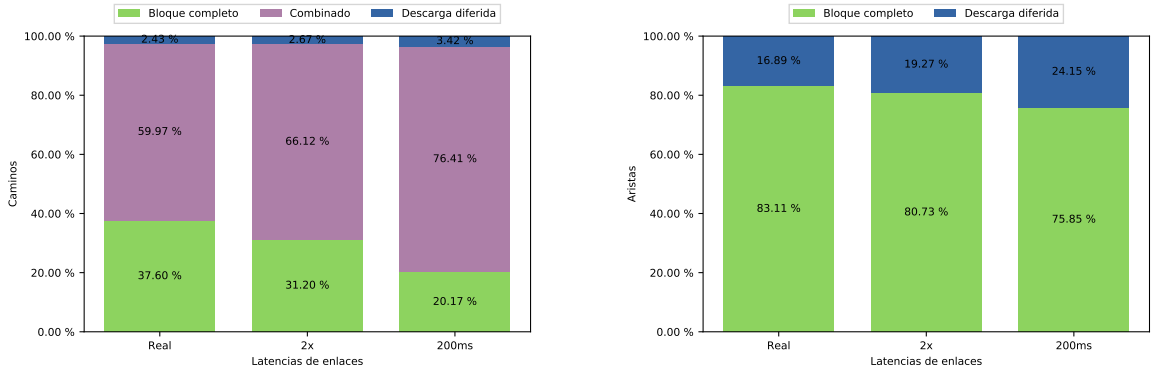
7.3 Diámetro de la red - Tipos de caminos

En esta sección analizamos otro aspecto del impacto del diámetro de la red sobre la propagación de bloques: Los tipos de propagaciones que ocurren durante el experimento.

Para diámetros pequeños, suponemos que la mayoría de las propagaciones se dará sobre caminos de bloque completo. A medida que aumentemos el diámetro de la red, esperamos ver un cambio en la distribución de los tipos de camino, con más apariciones de propagaciones de descarga diferida.

En la figura 7.9 podemos observar la distribución de los tipos de caminos y aristas a lo largo del experimento. Existe una tendencia decreciente en las transmisiones efectivas de bloque completo a medida que aumentamos el diámetro de la red. Atribuimos los resultados al siguiente fenómeno: En topologías de bajo diámetro las descargas diferidas que programan los nodos al recibir un hash, se ven interrumpidas por la recepción del bloque completo desde algún otro nodo de la red. Al aumentar el diámetro de la red, este fenómeno ocurre menos frecuentemente. De este último análisis podemos concluir que la ganancia en rendimiento obtenida utilizando la heurística de enviar solo el hash a ciertos peers está relacionada con el diámetro de la red.

Por otro lado, la tabla 7.11 indica diferencias en las alturas máximas de los arboles de propagación. Pudimos observar que al aumentar el diámetro de la red las alturas bajaron,



(a) Tipos de caminos (b) Tipos de aristas

Figura 7.9: Distribuciones de caminos y aristas en los árboles de propagación

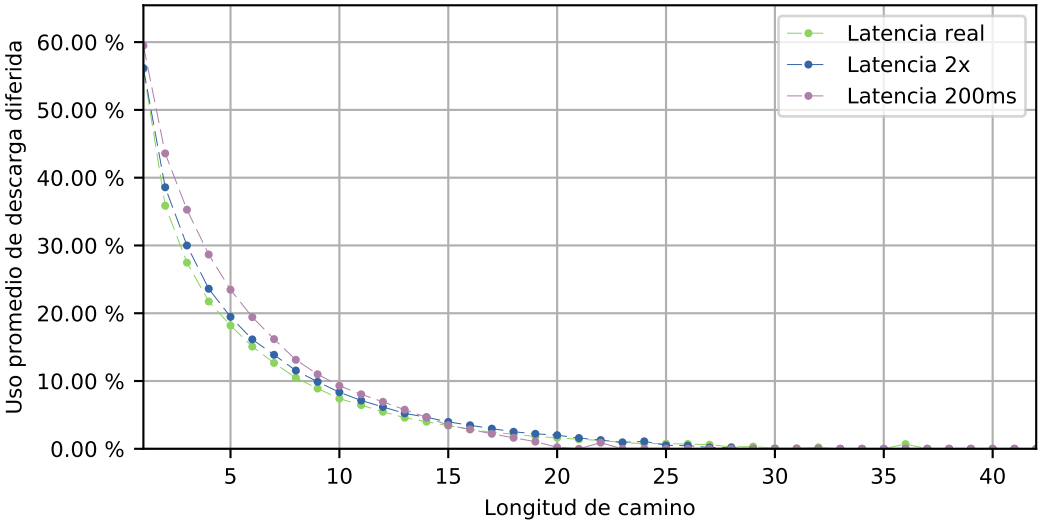


Figura 7.10: Evolución de los tipos de aristas predominantes en las propagaciones según longitud de los caminos del árbol de propagación

Máxima altura de los árboles			
Latencia	Real	2x	200 ms
Altura	42	32	23

Cuadro 7.11: Altura máxima de los arboles de propagación

lo cual indica que el grado de cada nodo aumentó[4]. Esto último está relacionado con la distribución de tipos de caminos y aristas: Para diámetros bajos, cada nodo propagará efectivamente los bloques solo a la raíz cuadrada de sus peers, ya que todos los envíos de hashes serán descartados por causa del fenómeno descripto en el párrafo anterior. En cambio, para diámetros más altos, al existir más aristas de descarga diferida en los caminos, cada nodo está propagando efectivamente los bloques a una cantidad mayor de sus peers.

En el figura 7.10 realizamos un análisis acerca de la proporción de transmisiones de descarga diferida que existen en promedio para caminos de distintas longitudes. Para ésto, se tomaron todos los caminos de una longitud fija k y se calculo en promedio cuantas ocurrencias de cada tipo de aristas existen a dicha profundidad en el árbol.

En dicho gráfico puede observarse que a medida que los caminos asociados a las propagaciones tienen más saltos, están formados por una mayor proporción de aristas de bloque completo.

Analizando la forma en la que se calculan los tiempos de propagación en un paso (Ver 3.9) podemos notar que el intervalo del *fetcher* es la componente predominante en los tiempos de propagación de las aristas. Entonces, cada arista de este tipo dentro de un camino, incrementa el tiempo total de propagación del mismo en al menos 500 ms. Por otro lado, los tiempos de propagación en un paso para aristas de bloque completo están determinadas por la latencia de la red subyacente, del órden de 30, 60 y 200 milisegundos para cada experimento. La probabilidad de recibir un bloque completo mientras se espera por la descarga diferida está determinada en gran parte por la relación entre el intervalo de descarga del *fetcher* y la latencia de los enlaces de la red subyacente. Es de esperar, entonces, esta distribución exponencial, dado que los tiempos máximos de propagación a toda la red de un bloque son del órden de 1300 ms (Figura 7.2), 1600 ms (Figura 7.4) y 2800 ms (Figura 7.7) para las latencias correspondientes.

[4]Dados dos árboles con la misma cantidad de nodos y distinta altura, el de menor altura debe tener nodos con mayor grado.

Capítulo 8

Trabajo futuro

En este capítulo se proponen algunas líneas de investigación que pueden servir como continuación a este trabajo. Las mismas se dividen en dos grupos: Modificaciones a la herramienta y análisis de aspectos de Ethereum que quedaron fuera del alcance de este trabajo.

8.1 Modificaciones al controlador de red

Dado que en nuestros experimentos la topología no cambia a lo largo de una ejecución, una posibilidad sería utilizar rutas estáticas para facilitar el trabajo del controlador, permitiendo emular topologías más complejas.

Para poder implementar esto en Maxinet, sería necesario realizar modificaciones en el código que permitan que una vez calculadas las tablas de ruteo de los switches, éstas queden fijas a lo largo de todo el experimento. También deberían establecerse rutas estáticas en el controlador. Floodlight permite realizar esto mediante un módulo llamado *Static Entry Pusher*.

Esta modificación permitiría realizar simulaciones en entornos con topologías más complejas.

8.2 Escenarios con transacciones

En este trabajo solo se analizó la creación y propagación de bloques, sin considerar que éstos contengan transacciones. Ésta decisión fue sustentada en la necesidad de tener un ambiente controlado donde las mediciones no fueran afectadas por cuestiones intrínsecas a la red tales como congestión, fragmentación o reordenamiento de paquetes.

La inclusión de este aspecto permitiría el estudio de la propagación de bloques de mayor tamaño. En marzo de 2017 Dihn et al. publicaron el borrador de la herramienta Blockbench en [DWC+17]. La misma permite la utilización de trazas para la simulación del uso de transacciones en redes privadas de criptomonedas. De integrar ésta funcionalidad a nuestra herramienta podríamos incluir transacciones en nuestros experimentos de forma controlada.

8.3 Testbed para análisis de seguridad

El uso de nuestra metodología podría funcionar como un testbed para nuevas criptomonedas que quieran analizar los efectos de diversos ataques sobre sus redes.

Por ejemplo, para simular ataques de minado egoísta, podría implementarse un nuevo módulo de minado simulado en un cliente que se comporte como atacante, y con dicha infraestructura simular escenarios en los cuales se asignen incrementalmente porcentajes de hashing power al atacante. De esta forma podría obtenerse el porcentaje mínimo de poder de cómputo necesario para llevar a cabo este ataque. Con estos mismos escenarios, podría analizarse el impacto de posibles medidas para minimizar vulnerabilidades.

8.4 Modificaciones al protocolo de propagación

Tanto nuestros resultados sobre Ethereum como también otros trabajos sobre Bitcoin[1] muestran que el tiempo de propagación de los bloques en la red es una de las principales causas de la aparición de forks.

[1] Decker y Wattenhofer en [DW13]

Una posible heurística para minimizar los tiempos de propagación podría ser modificar el intervalo de descarga explicita de bloques programados (Actualmente cada 500 ms).

Se espera que esta modificación baje drásticamente el tiempo de propagación a toda la red, disminuyendo la cantidad de forks en la red. Si esto ocurre, se podrían utilizar tiempos entre bloques más bajos y aumentar la cantidad de transacciones procesadas por minuto.

Nuestra herramienta nos provee un registro de todos los eventos de la red. Esto permitiría cuantificar el volumen de tráfico generado por cada nodo y extraer información estadística acerca de la efectividad de las transmisiones. Con esta nueva métrica podría evaluarse el impacto de nuevas heurísticas en los algoritmos de propagación como también en el volumen de tráfico total.

8.5 Modificaciones al protocolo de consenso

Dada la naturaleza distribuida de Ethereum, un aspecto importante es su protocolo de consenso. Este último se encarga de resolver conflictos entre cadenas competitivas de bloques. Nuestra herramienta, al utilizar el cliente real, permite evaluar nuevas implementaciones del protocolo de consenso y analizar su funcionamiento.

Capítulo 9

Conclusiones

En este trabajo proponemos una metodología para realizar un análisis estático de trazas de ejecución de criptomonedas, utilizando Ethereum. Para ello desarrollamos una herramienta configurable que permite la simulación de diversos escenarios con métricas extensibles según los aspectos de interés a analizar sobre la red blockchain.

Realizamos diversas validaciones para corroborar que nuestra metodología de experimentación en ambientes simulados nos permita extraer resultados semejantes a entornos de producción. Esta serie de experimentos corrobora la similitud entre el algoritmo de minado real y una simulación con un modelo estadístico que no consume recursos. En términos de emulación de topologías de red por software, observamos que hay que tomar ciertas precauciones respecto a los ambientes elegidos. Mientras estos últimos sean adecuados para las herramientas de emulación de redes, nuestra metodología se comportará de forma similar a un ambiente real.

Como primer resultado, hallamos una relación entre el tiempo entre bloques y el rendimiento de la red, haciendo énfasis en el análisis de forks en la red. Por otro lado, concluimos que el diámetro de las redes juega un papel fundamental en la elección de un tiempo entre bloques, ya que éste afecta directamente los tiempos de propagación y en consecuencia, el rendimiento de la blockchain.

Luego analizamos en detalle el funcionamiento del algoritmo de propagación de Ethereum. Relacionamos diversos aspectos de los histogramas de propagación con los distintos tipos de transmisiones en los árboles de propagación de los bloques. Además, para las distribuciones de tiempos de transmisiones de camino completo, realizamos tests de normalidad cuyos resultados indicaron la correspondencia de éstas con distribuciones normales.

Otro resultado relacionado con la propagación de bloques es la existencia de una relación entre el diámetro de la red y la distribución de los tipos de caminos que aparecen en los árboles de propagación, así como también la altura de estos últimos. A medida que el diámetro de la red es más grande, la heurística de envió de bloque por descarga diferida se vuelve más útil, logrando más transmisiones efectivas de este tipo y de esta forma impactando en la altura de los árboles de propagación.

Apéndice A

Experimentación completa de validación

Este apéndice contiene los resultados completos de los experimentos realizados a lo largo del trabajo.

1.0.1 Procesos de minado

En esta sección presentamos todos los resultados de los experimentos de validación entre minado real y minado simulado con las características descriptas en A.1.

Configuración del experimento	
Topología física	Ethernet/Museo
Topología emulada	Ninguna
Topología lógica	Clique - 10 nodos
Bloques analizados	3800
Porcentaje mineros	50 % de la red
Tiempos entre bloques	7, 14,5 y 21 segundos

Cuadro A.1: Configuración de los experimentos de minado real - minado simulado

Métrica	Minado real	Minado simulado
Main Chain/Total minados	8000/8071	8000/8053
Tiempo de target (promedio)	7,340 s	7,192 s
Forks profundidad 1	71	52
Forks profundidad >1	0	0
Competencias entre 1 bloques	7948	7929
Competencias entre 2 bloques	52	71
Competencias entre 3 bloques	0	0

(a) Tiempo entre bloques de 7s

Métrica	Minado real	Minado simulado
Main Chain/Total minados	3800/3811	3800/3812
Tiempo de target (promedio)	14,375 s	13,992 s
Forks profundidad 1	10	11
Forks profundidad >1	0	0
Competencias entre 1 bloques	3790	3789
Competencias entre 2 bloques	10	11
Competencias entre 3 bloques	0	0

(b) Tiempo entre bloques de 14,5s

Métrica	Minado real	Minado simulado
Main Chain/Total minados	3800/3806	3800/3806
Tiempo entre bloques (promedio)	20,179 s	20,262 s
Forks profundidad 1	6	6
Forks profundidad >1	0	0
Competencias entre 1 bloques	3794	3794
Competencias entre 2 bloques	6	6
Competencias entre 3 bloques	0	0

(c) Tiempo entre bloques de 21s

Cuadro A.2: Métricas de minado - Validación del proceso de minado

Algoritmo de minado	Minero 1	Minero 2	Minero 3	Minero 4	Minero 5
Real	19,82 %	20,32 %	19,98 %	20,02 %	19,85 %
Simulado	19,77 %	19,66 %	20,00 %	20,23 %	20,33 %
Diferencia relativa	−0,25 %	−3,36 %	0,10 %	1,04 %	2,36 %

(a) Tiempo entre bloques 7s

Algoritmo de minado	Minero 1	Minero 2	Minero 3	Minero 4	Minero 5
Real	20,10 %	20,23 %	19,92 %	20,62 %	19,10 %
Simulado	20,75 %	19,94 %	19,88 %	18,52 %	20,88 %
Diferencia relativa	3,13 %	−1,45 %	−0,20 %	−11,34 %	8,52 %

(b) Tiempo entre bloques 14,5s

Algoritmo de minado	Minero 1	Minero 2	Minero 3	Minero 4	Minero 5
Real	21,09 %	19,18 %	19,91 %	19,44 %	20,36 %
Simulado	20,38 %	18,86 %	20,46 %	19,81 %	20,46 %
Diferencia relativa	−3,48 %	−1,70 %	2,69 %	1,87 %	0,49 %

(c) Tiempo entre bloques 21s

Cuadro A.3: Distribución del Hashing Power

Categoría de bloque	Minado real	Minado simulado
Stale	0,89 %	0,66 %
Cadena Principal	99,11 %	99,34 %

(a) Tiempo entre bloques 7s

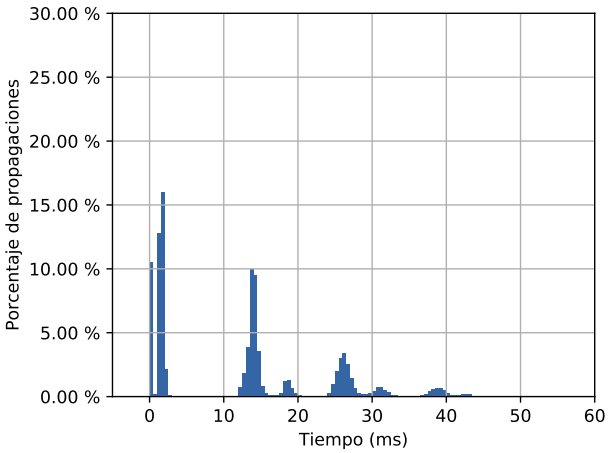
Categoría de bloque	Minado real	Minado simulado
Stale	0,29 %	0,31 %
Cadena Principal	99,71 %	99,69 %

(b) Tiempo entre bloques 14,5s

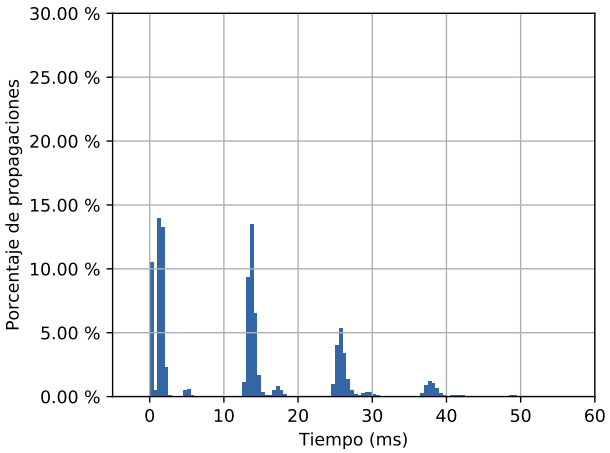
Categoría de bloque	Minado real	Minado simulado
Stale	0,18 %	0,18 %
Cadena Principal	99,82 %	99,82 %

(c) Tiempo entre bloques 21s

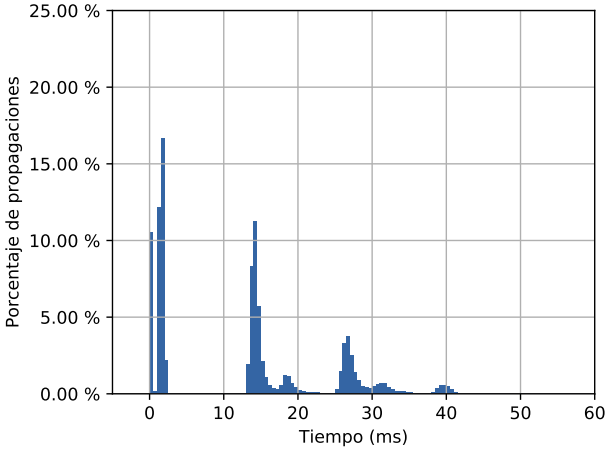
Cuadro A.4: Rendimiento de la Blockchain



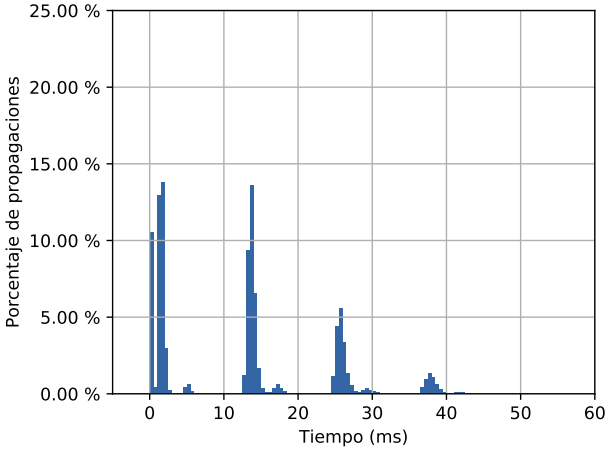
(a) Minado real - Target 7s



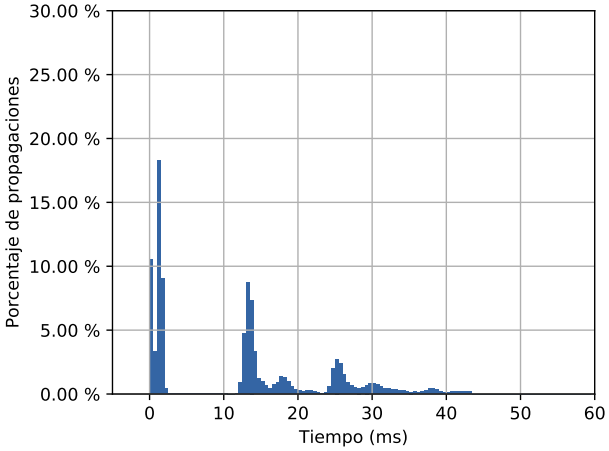
(b) Minado simulado - Target 7s



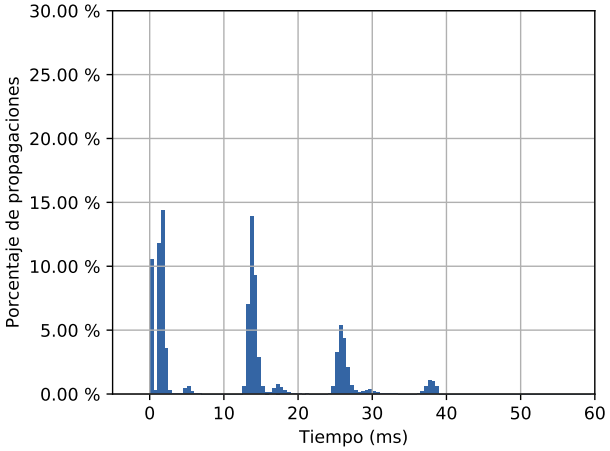
(c) Minado real - Target 14,5s



(d) Minado simulado - Target 14,5s

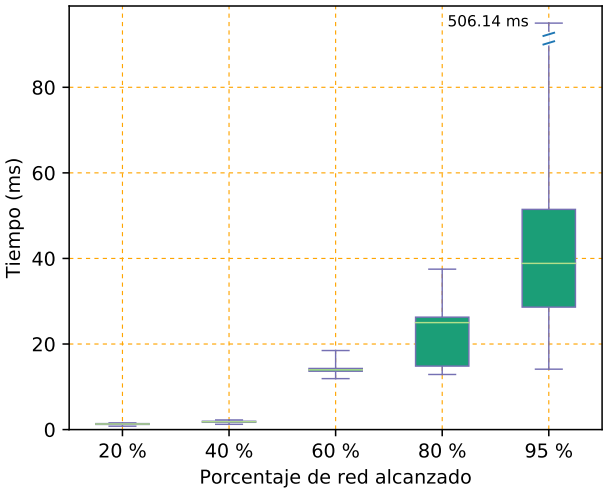


(e) Minado real - Target 21s

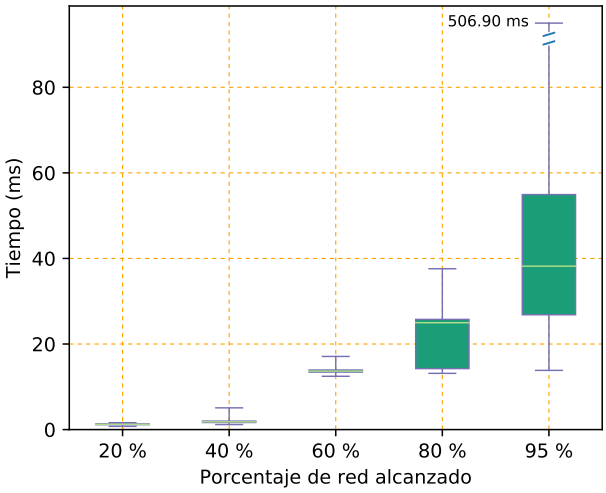


(f) Minado simulado - Target 21s

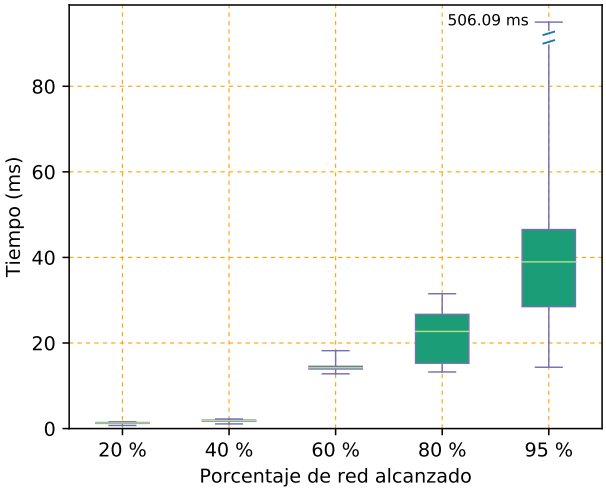
Figura A.1: Histograma de propagación de bloques - Validación procesos de minado



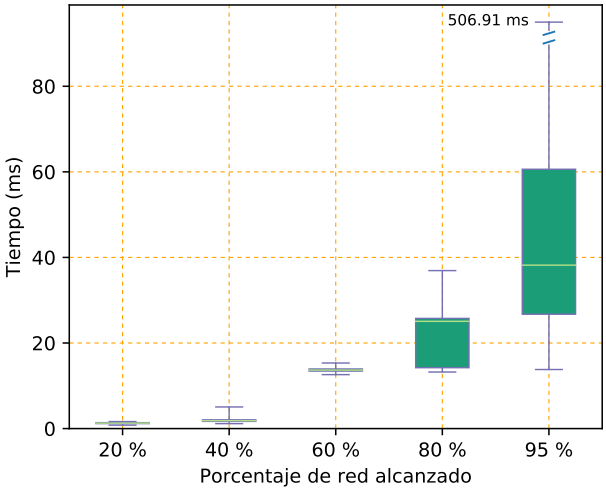
(a) Minado real - Target 7s



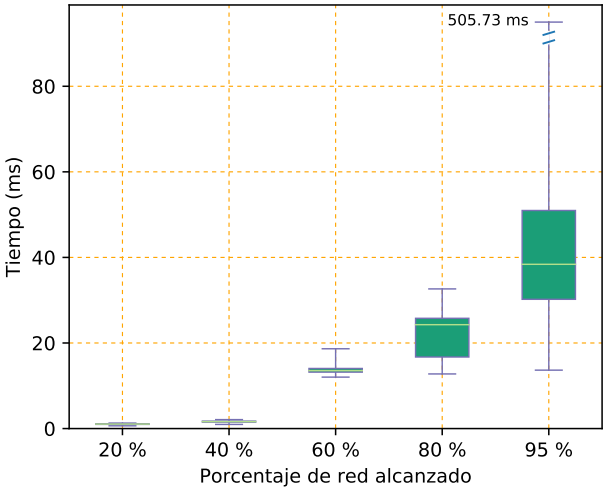
(b) Minado simulado - Target 7s



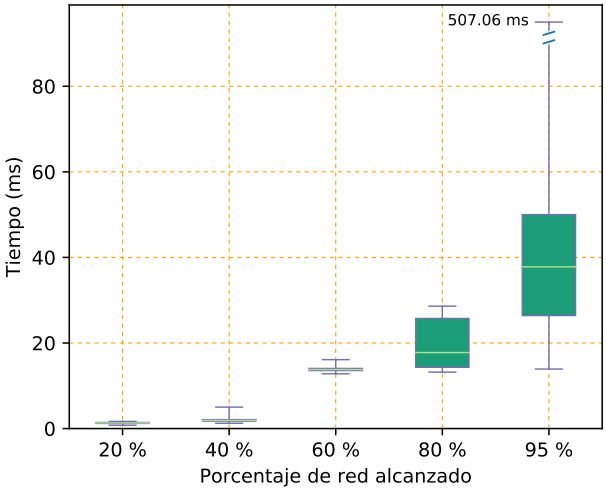
(c) Minado real - Target 14,5s



(d) Minado simulado - Target 14,5s

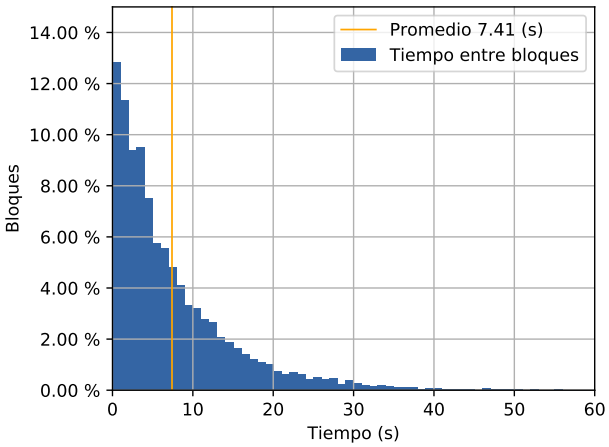


(e) Minado real - Target 21s

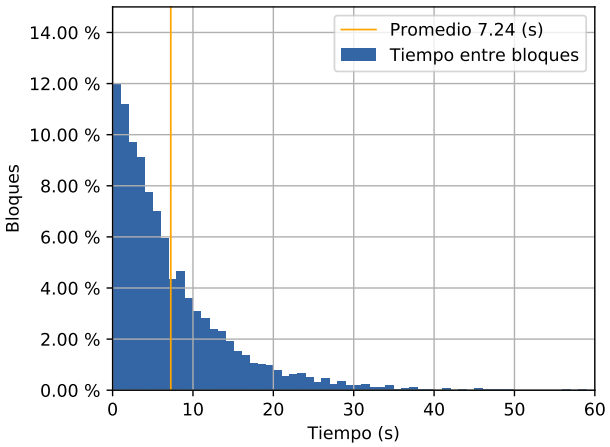


(f) Minado simulado - Target 21s

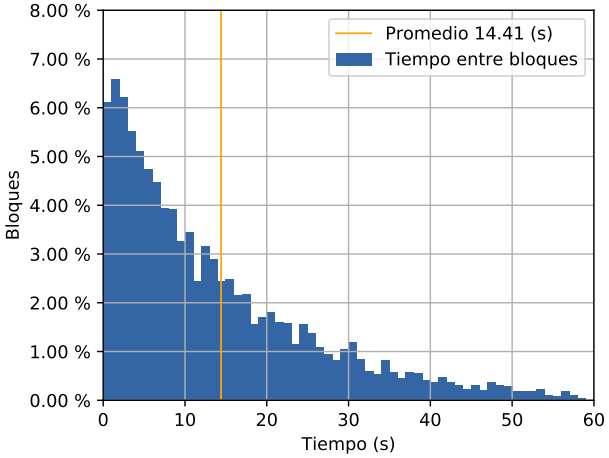
Figura A.2: Alcance de propagación de bloques - Validación del proceso de minado



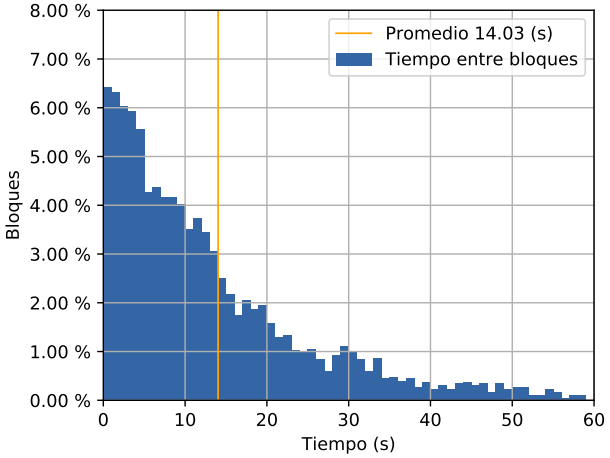
(a) Minado real - Target 7s



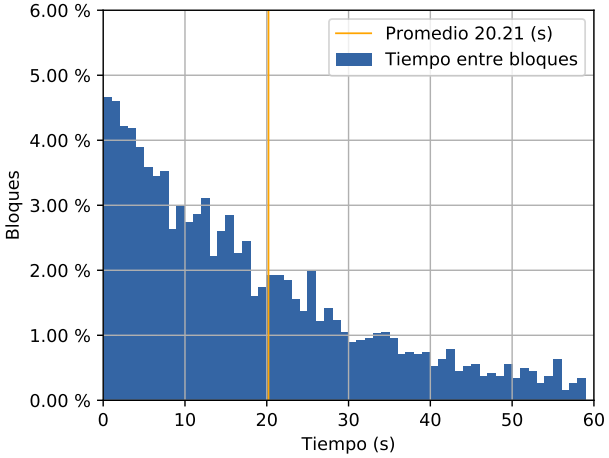
(b) Minado simulado - Target 7s



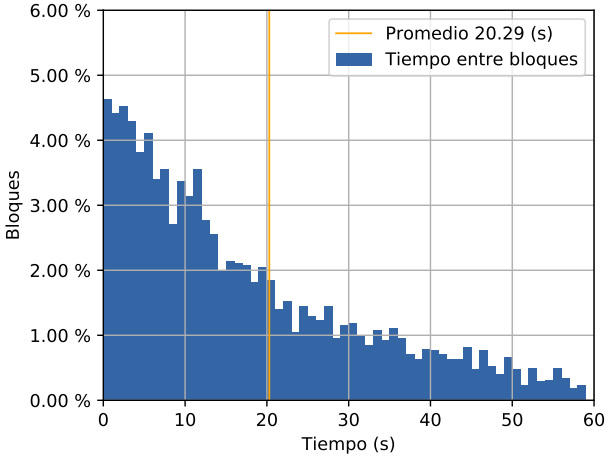
(c) Minado real - Target 14,5s



(d) Minado simulado - Target 14,5s



(e) Minado real - Target 21s



(f) Minado simulado - Target 21s

Figura A.3: Histograma de tiempo entre bloques

1.1 Emulación de topologías de redes LAN

Los experimentos a continuación tienen las características presentadas en A.5.

Configuración del experimento		
Característica	Ambiente real	Ambiente emulado
Topología física	Laboratorios DC	Ethernet/Museo
Topología emulada	Ninguna	Laboratorios DC
Topología lógica	Clique - 12 nodos	
Bloques analizados	4000	
% Mineros	50 % de la red distribuidos uniformemente	
Tiempos entre bloques	7, 14,5 y 21 segundos	
Proceso de minado	Simulado	

Cuadro A.5: Configuración de los experimentos de emulación de redes LAN

Métrica	Laboratorios DC	Maxinet
Main Chain/Total minados	4000/4026	4000/4040
Tiempo entre bloques (promedio)	6,996 s	6,837 s
Forks profundidad 1	25	39
Forks profundidad >1	0	0
Competencias entre 1 bloques	3975	3962
Competencias entre 2 bloques	25	37
Competencias entre 3 bloques	0	1

(a) Tiempo entre bloques de 7s

Métrica	Laboratorios DC	Maxinet
Main Chain/Total minados	4000/4017	4000/4024
Tiempo de target (promedio)	14,268 s	14,589 s
Forks profundidad 1	16	21
Forks profundidad >1	0	1
Competencias entre 1 bloques	3984	3977
Competencias entre 2 bloques	16	23
Competencias entre 3 bloques	0	0

(b) Tiempo entre bloques de 14,5s

Métrica	Laboratorios DC	Maxinet
Main Chain/Total minados	4000/4004	4000/4015
Tiempo de target (promedio)	20,601 s	20,712 s
Forks profundidad 1	4	14
Forks profundidad >1	0	0
Competencias entre 1 bloques	3996	3986
Competencias entre 2 bloques	4	14
Competencias entre 3 bloques	0	1

(c) Tiempo entre bloques de 21s

Cuadro A.6: Métricas de minado - Ambientes de red

Ambiente	Minero 1	Minero 2	Minero 3	Minero 4	Minero 5	Minero 6
Laboratorios DC	16,82 %	16,44 %	16,49 %	16,27 %	16,54 %	17,44 %
Maxinet	17,52 %	16,78 %	16,34 %	16,68 %	15,35 %	17,33 %
Diferencia relativa	4,00 %	2,03 %	−0,92 %	2,46 %	−7,75 %	−0,63 %

(a) Tiempo entre bloques 7s

Ambiente	Minero 1	Minero 2	Minero 3	Minero 4	Minero 5	Minero 6
Laboratorios DC	15,11 %	18,05 %	16,38 %	16,83 %	16,6 %	17,03 %
Maxinet	15,95 %	16,9 %	17,82 %	16,7 %	16,33 %	16,3 %
Diferencia relativa	5,27 %	−6,80 %	8,08 %	−0,78 %	−1,65 %	−4,48 %

(b) Tiempo entre bloques 14,5s

Ambiente	Minero 1	Minero 2	Minero 3	Minero 4	Minero 5	6 Minero
Laboratorios DC	15,98 %	17,53 %	16,7 %	16,85 %	16,33 %	16,61 %
Maxinet	18,16 %	16,64 %	17,14 %	16,41 %	15,77 %	15,88 %
Diferencia relativa	12,00 %	−5,35 %	2,57 %	−2,68 %	−3,55 %	−4,60 %

(c) Tiempo entre bloques 21s

Cuadro A.7: Distribución del Hashing Power - Ambientes de red

Categoría de bloque	Laboratorios DC	Maxinet
Stale	0,65 %	0,99 %
Cadena Principal	99,35 %	99,01 %

(a) Tiempo entre bloques 7s

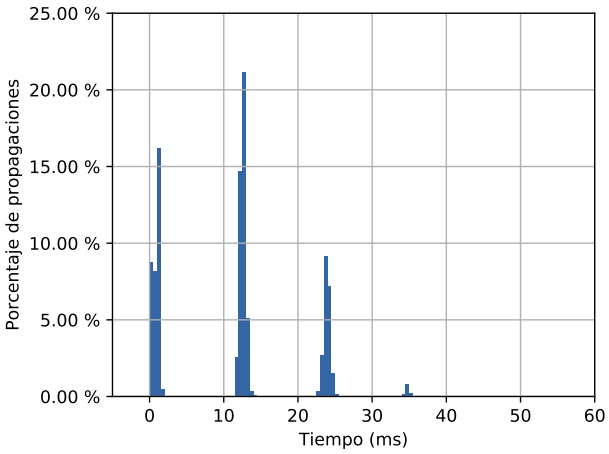
Categoría de bloque	Laboratorios DC	Maxinet
Stale	0,42 %	0,60 %
Cadena Principal	99,58 %	99,40 %

(b) Tiempo entre bloques 14,5s

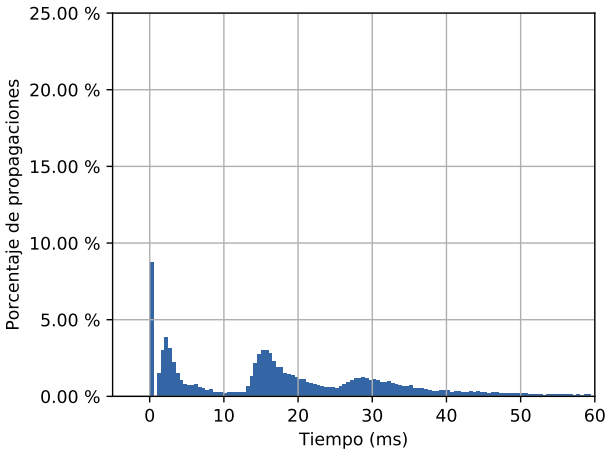
Categoría de bloque	Laboratorios DC	Maxinet
Stale	0,12 %	0,37 %
Cadena Principal	99,88 %	99,63 %

(c) Tiempo entre bloques 21s

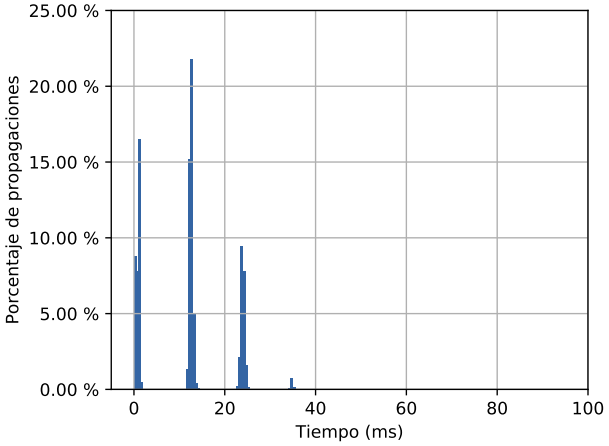
Cuadro A.8: Rendimiento de la Blockchain



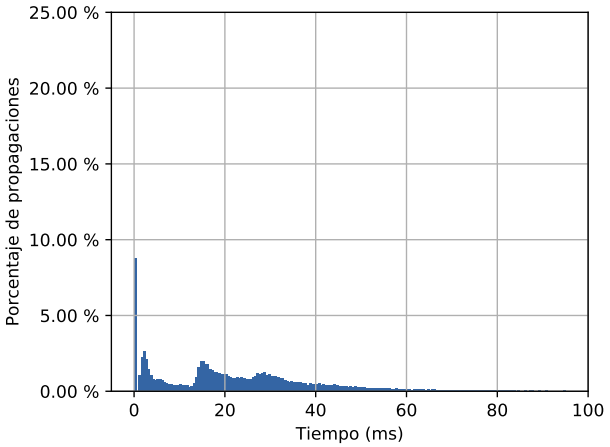
(a) Laboratorios DC - Target 7s



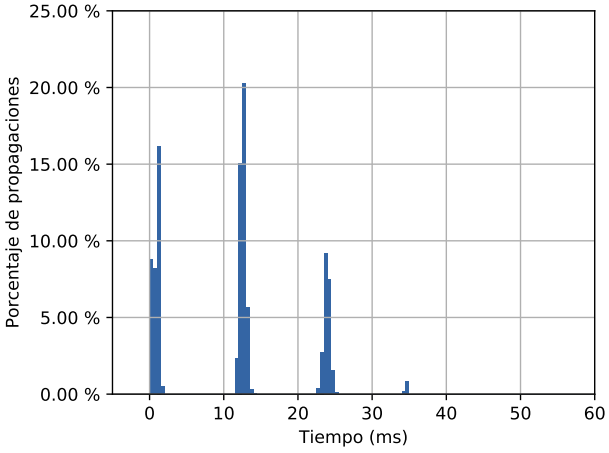
(b) Maxinet Topología Laboratorios DC - Target 7s



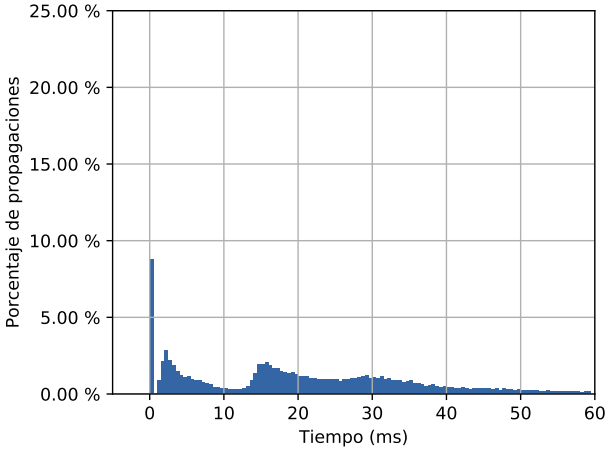
(c) Laboratorios DC - Target 14,5s



(d) Maxinet Topología Laboratorios DC - Target 14,5s

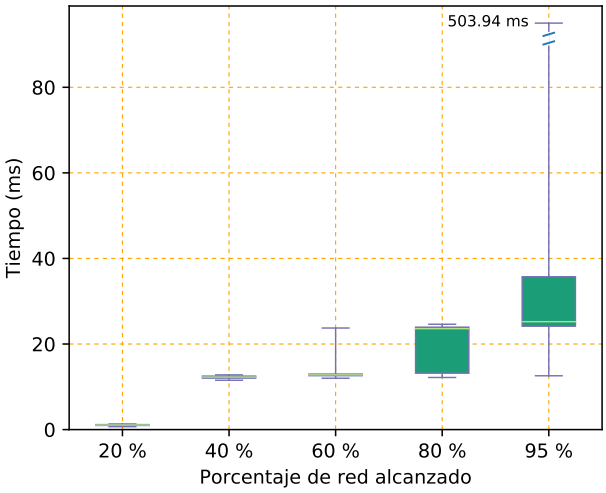


(e) Laboratorios DC - Target 21s

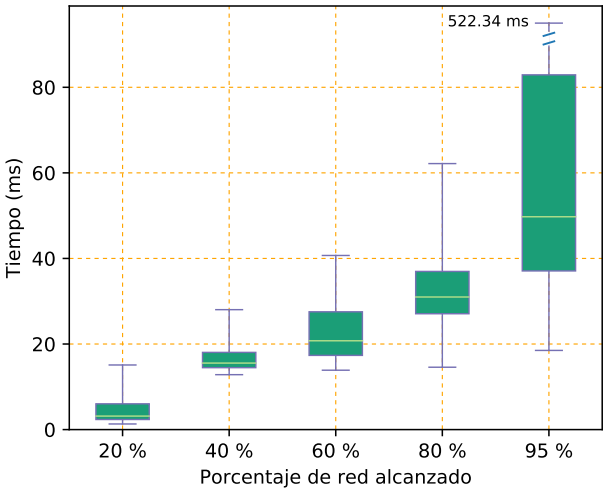


(f) Maxinet Topología Laboratorios DC - Target 21s

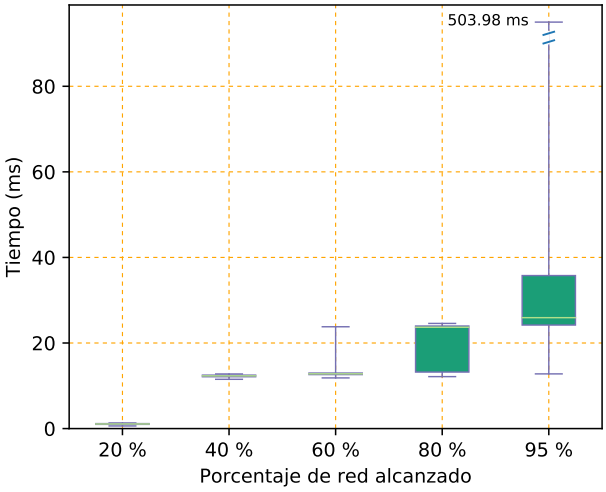
Figura A.4: Histograma de propagación de bloques - Ambientes de red



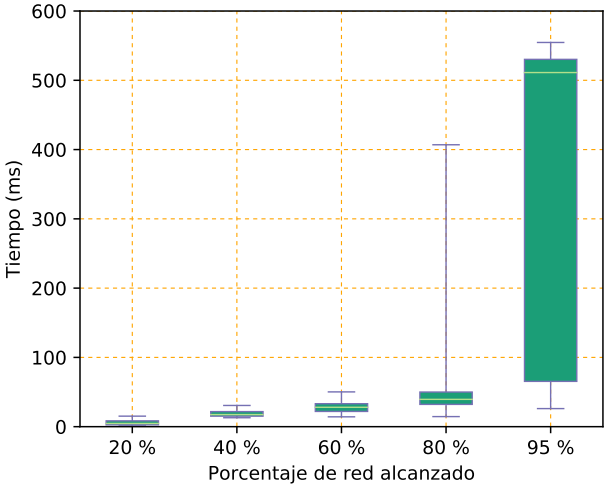
(a) Laboratorios DC - Target 7s



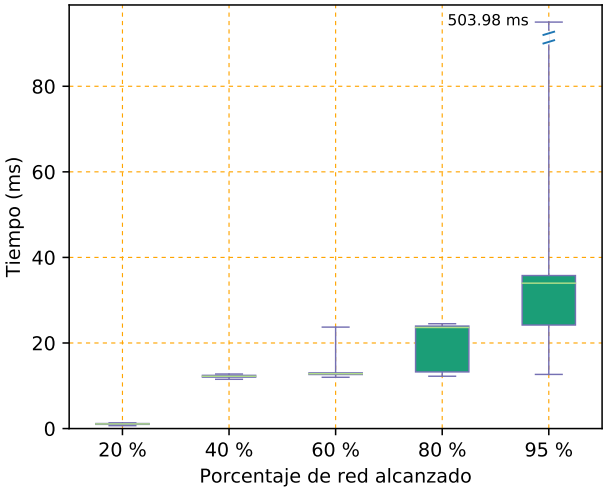
(b) Maxinet - Target 7s



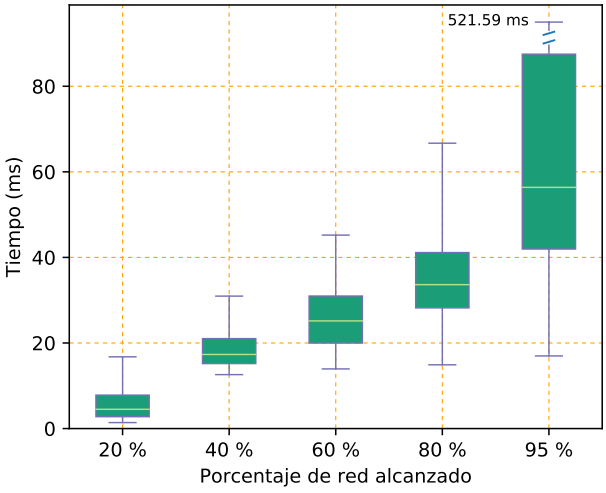
(c) Laboratorios DC - Target 14,5s



(d) Maxinet - Target 14,5s

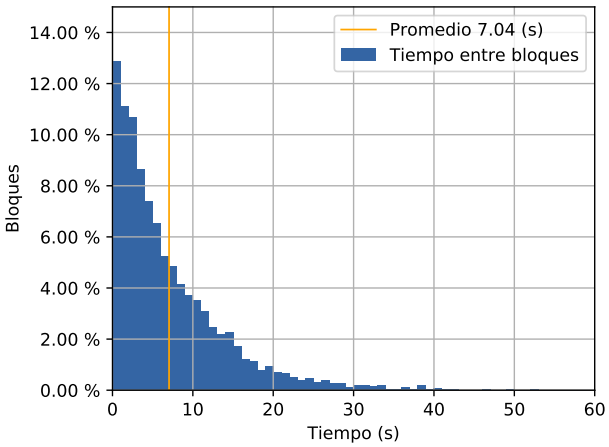


(e) Laboratorios DC - Target 21s

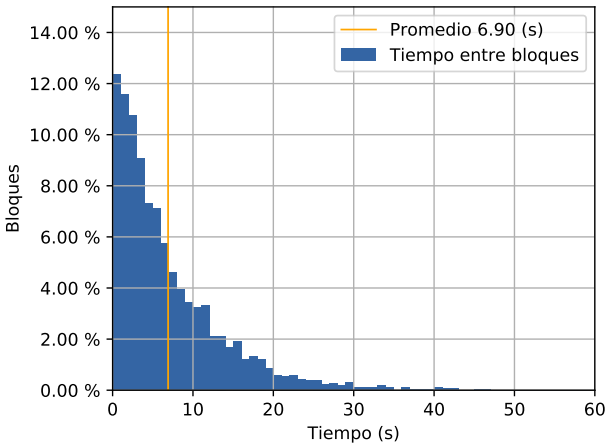


(f) Maxinet - Target 21s

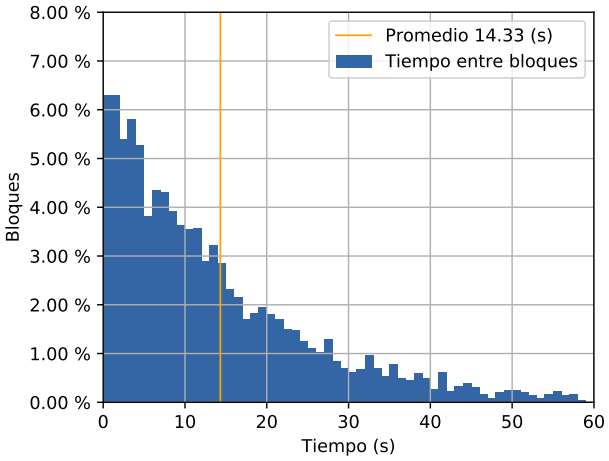
Figura A.5: Alcance de propagación de bloques - Ambientes de red



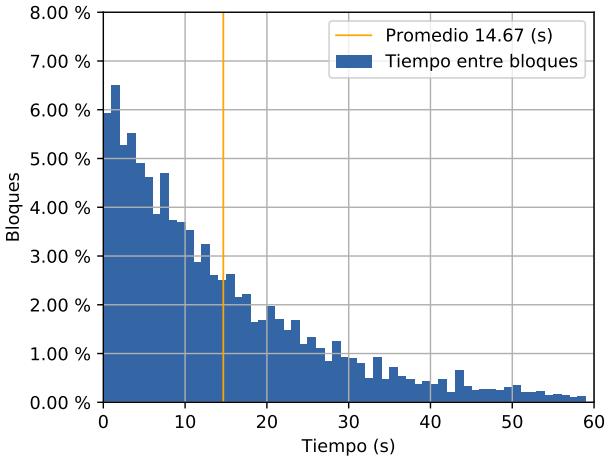
(a) Laboratorios DC - Target 7s



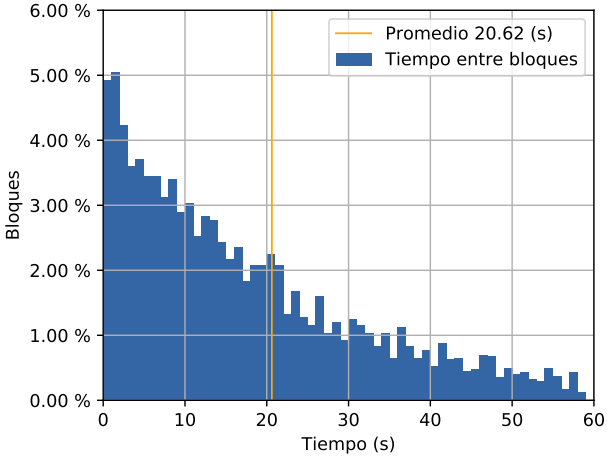
(b) Maxinet - Target 7s



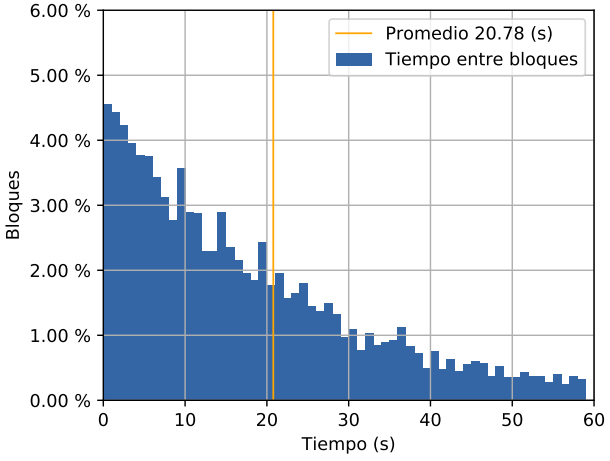
(c) Laboratorios DC - Target 14,5s



(d) Maxinet - Target 14,5s



(e) Laboratorios DC - Target 21s



(f) Maxinet - Target 21s

Figura A.6: Histograma de tiempo entre bloques - Ambientes de red

Impacto del jitter en la distribución de los tiempos de propagación

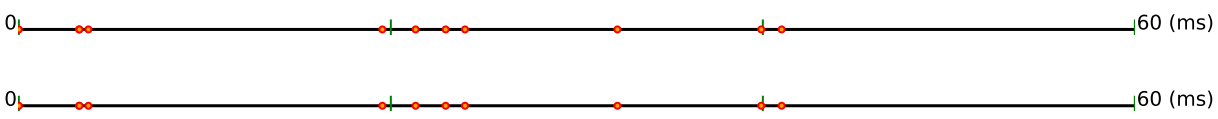


Figura A.7: Propagaciones de dos bloques en red con baja varianza

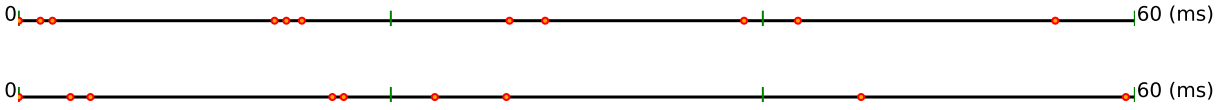
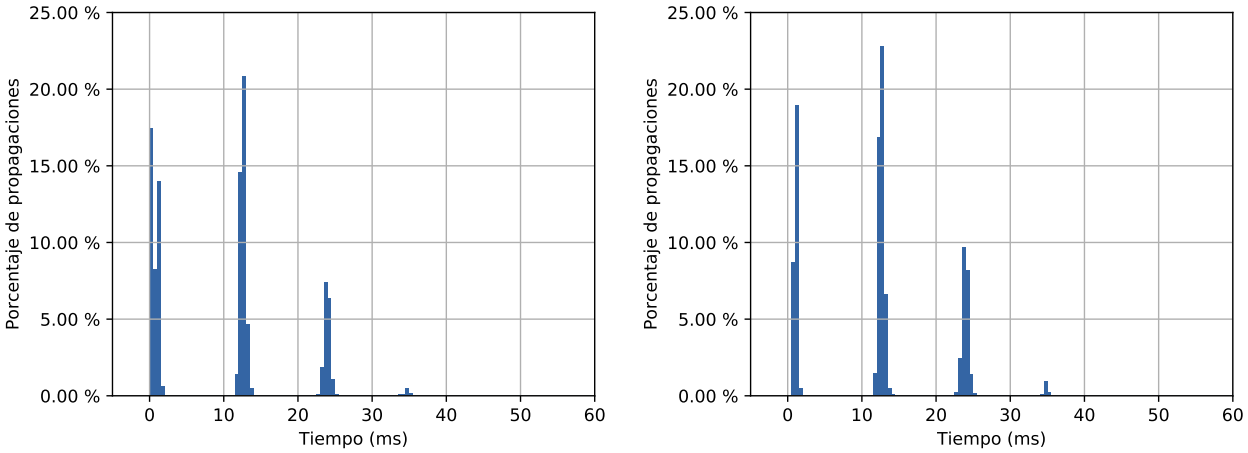


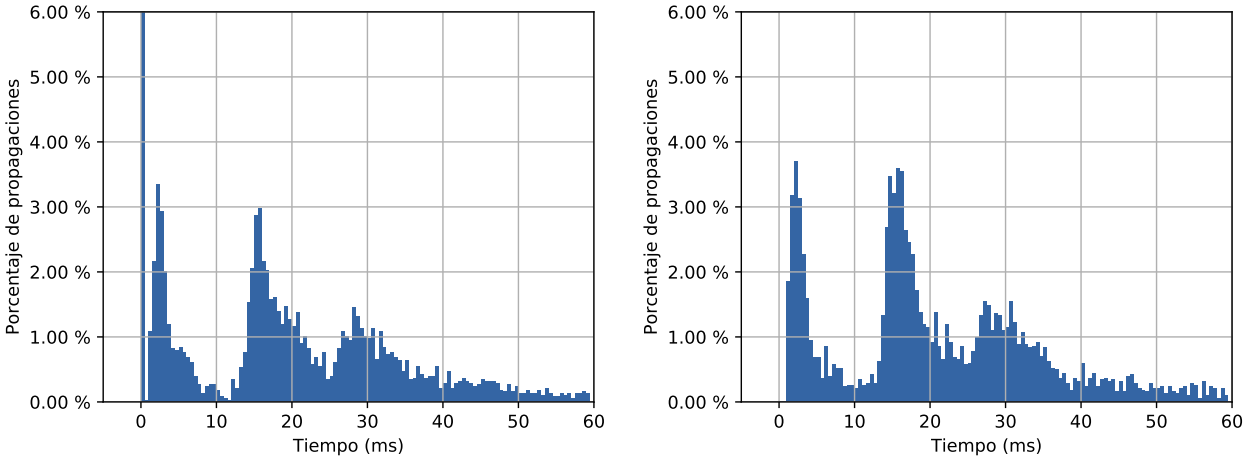
Figura A.8: Propagaciones de dos bloques en red con alta varianza



(a) Nodo minero

(b) Nodo pasivo

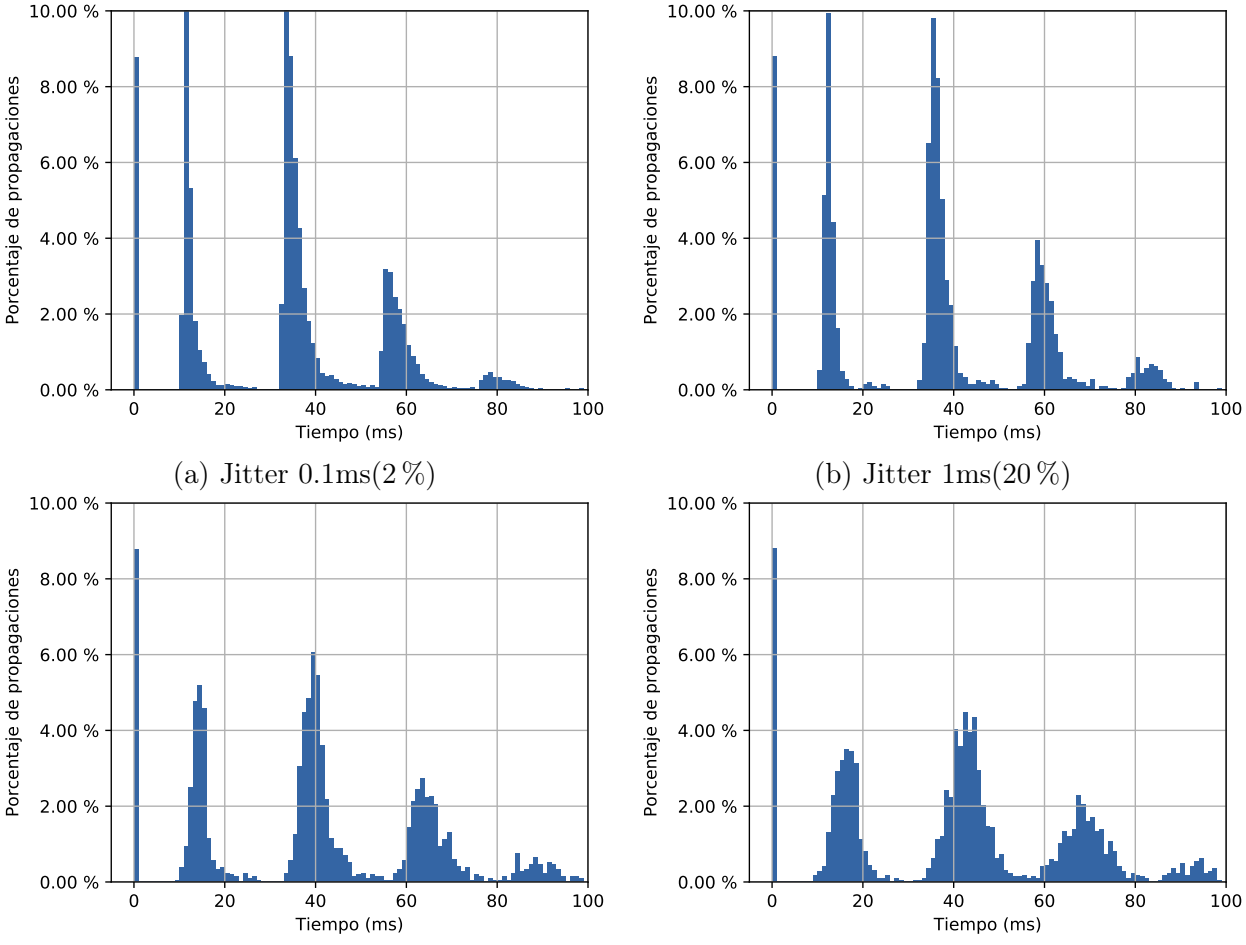
Figura A.9: Histograma de propagación visto desde un nodo - Baja varianza en latencia



(a) Nodo minero

(b) Nodo pasivo

Figura A.10: Histograma de propagación visto desde un nodo - Alta varianza en latencia



(a) Jitter 0.1ms(2 %)

(b) Jitter 1ms(20 %)

(c) Jitter 2.5ms(50 %)

(d) Jitter 4ms(80 %)

Figura A.11: Experimentos de propagación sobre Maxinet: Link delay 5ms - Distintos jitters

Bibliografía

[bit] Global bitcoin nodes distribution. https://bitnodes.21.co/. Accessed: 2016-09-30.

[CDE+16] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, and Emin Gün. On scaling decentralized blockchains. In *Proc. 3rd Workshop on Bitcoin and Blockchain Research*, 2016.

[cmp] Bip 152 - compact blocks. https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki. Accessed: 2016-10-10.

[DPSHJ14] Joan Antoni Donet Donet, Cristina Pérez-Sola, and Jordi Herrera-Joancomartí. The bitcoin p2p network. In *International Conference on Financial Cryptography and Data Security*, pages 87–102. Springer, 2014.

[DW13] Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings*, pages 1–10. IEEE, 2013.

[DWC+17] Tien Tuan Anh Dinh, Ji Wang, Gang Chen, Rui Liu, Beng Chin Ooi, and Kian-Lee Tan. Blockbench: A framework for analyzing private blockchains. https://arxiv.org/pdf/1703.04057.pdf, 2017. Accessed: 2017-03-22.

[ES14] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *International Conference on Financial Cryptography and Data Security*, pages 436–454. Springer, 2014.

[eth] Global ethereum nodes distribution. http://ethernodes.org/network/1.

[flo] Project floodlight. http://www.projectfloodlight.org/floodlight/. Accessed: 2016-09-30.

[GKL15] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 281–310. Springer, 2015.

[GKW+16] Arthur Gervais, Ghassan O. Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 3–16, New York, NY, USA, 2016. ACM.

[H+05] Stephen Hemminger et al. Network emulation with netem. In *Linux conf au*, pages 18–23, 2005.

[KK95] George Karypis and Vipin Kumar. Metis–unstructured graph partitioning and sparse matrix ordering system, version 2.0. 1995.

[KP15] Aggelos Kiayias and Giorgos Panagiotakos. Speed-security tradeoffs in blockchain protocols. Technical report, IACR: Cryptology ePrint Archive, 2015.

[KP16] Aggelos Kiayias and Giorgos Panagiotakos. On trees, chains and fast transactions in the blockchain. 2016.

[LHM10] Bob Lantz, Brandon Heller, and Nick McKeown. A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, page 19. ACM, 2010.

[MAL17] Apostolaki Maria, Zohar Aviv, and Vanbever Laurent. Hijacking bitcoin: Routing attacks on cryptocurrencies. In *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 2017.

[MLP+15] Andrew Miller, James Litton, Andrew Pachulski, Neal Gupta, Dave Levin, Neil Spring, and Bobby Bhattacharjee. Discovering bitcoin's public topology and influential nodes, 2015.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[SSZ15] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. *arXiv preprint arXiv:1507.06183*, 2015.

[SZ13] Yonatan Sompolinsky and Aviv Zohar. Accelerating bitcoin's transaction processing. fast money grows on trees, not chains. *IACR Cryptology ePrint Archive*, 2013:881, 2013.

[Van16] Marco Vanotti. Un avance hacia entornos de gran escala para experimentos con criptomonedas. 2016.

[WDS14] Philip Wette, Martin Dräxler, and Arne Schwabe. Maxinet: Distributed emulation of software-defined networks. In *Networking Conference, 2014 IFIP*, pages 1–9. IEEE, 2014.

[Woo14] Gavin Wood. Ethereum yellow paper, 2014.

[xth] Towards massive on-chain scaling: Presenting our block propagation results with xthin. https://medium.com/@peter_r/towards-massive-with-xthin-da54e55dc0e4#.kk1znlg1v. Accessed: 2016-10-10.

[ZP17] Ren Zhang and Bart Preneel. Publish or perish: A backward-compatible defense against selfish mining in bitcoin. In *Cryptographers' Track at the RSA Conference*, pages 277–292. Springer, 2017.