Universidad de Buenos Aires

Facultad de Ciencias Exactas y Naturales

Departamento de Computación

Impacto de la red FIBRE en Bitcoin: ¿Los mineros crearon un sistema casi centralizado?

Tesis presentada para optar al título de
Licenciadoen Ciencias de la Computación

Julio Augusto, Mascitti

Director: Esteban Mocskos

Buenos Aires, 2022

Resumen

En 2008, alguien bajo el pseudónimo de Satoshi Nakamoto propuso una solución computacional al problema de intercambiar valor entre participantes desconocidos sin la necesidad de un tercero de confianza que medie dichas transacciones. De esta manera nació Bitcoin y revolucionó la industria financiera proveyendo un sistema distribuido sin la necesidad de una autoridad central que regule las transacciones. Este sistema comenzó a ser una alternativa real a los sistemas bancarios tradicionales. A fines de 2018 durante uno de los principales picos de utilización de las criptomonedas, se pusieron de manifiesto distintos problemas operativos. Por ejemplo, se evidenció la limitada capacidad de procesar un volumen importante de transacciones.

Como una forma de disminuir los cuellos de botella que presenta Bitcoin, se propuso la utilización de redes de retransmisión. Una red de retransmisión plantea una modificación a la forma en que se distribuye la información entre pares, buscando que algunos nodos seleccionados de la red tengan prioridad para el intercambio de información en lugar de usar la propuesta original de Bitcoin.

En esta tesis nos enfocamos en analizar el impacto de las redes de retransmisión, intentando verificar si son una alternativa viable que mejora las prestaciones del sistema. Para tal fin, generamos toda la infraestructura necesaria para ejecutar un sistema Bitcoin sobre una red emulada considerando todos los actores necesarios en condiciones similares a las reales. Los experimentos se realizan utilizando los clientes del protocolo Bitcoin real con unas mínimas modificaciones que permiten modelar el proceso de minado de un bloque sin tener

que utilizar recursos del hardware para tal fin. Se realizaron las mediciones de interés una vez que el sistema ha generado 1000 bloques para poder disminuir los posibles efectos artificiales originados durante la etapa inicial de funcionamiento del mismo. Los resultados finales de los experimentos surgen de tomar una muestra de 500 bloques minados, con el funcionamiento de todo el sistema en marcha.

Como resultado principal, pudimos concluir que las redes de retransmisión otorgan interesantes beneficios al sistema a la hora de aumentar la velocidad de consenso y disminuir el desperdicio de recursos. Por otro lado, pudimos verificar que la utilización de las redes de retransmisión no llevan a una mayor centralización del sistema ya que aquellos clientes que usufructúan esta red no consiguieron mejorar su participación en término de recompensa. Sin embargo, lo que sí consiguieron fue ser más eficientes. No solo el sistema en general mejoró su eficiencia, sino que aquellos clientes que estaban al alcance de la red de retransmisión consiguieron tener una mejora proporcionalmente mayor al resto de los clientes del sistema.

Índice general

Índice de figuras

Capítulo 1

Introducción

Bitcoin apareció en 2008 con la publicación del trabajo: "Bitcoin: A Peer-to-Peer Electronic Cash System" escrito por un autor bajo el seudónimo de Satoshi Nakamoto [Nak08]. Ésta fue la primer propuesta de un sistema distribuido que solucionaba el problema del *double spending* (doble gasto) sin la necesidad de que una entidad centralizada valide las transacciones realizadas dentro del sistema. Utilizando conceptos criptográficos, se convirtió en el primer sistema distribuido y descentralizado para la transferencia de dinero. Desde entonces, proliferaron distintas criptomonedas orientadas a solucionar varios problemas de la vida real.

Con ya más de trece años en funcionamiento, Bitcoin se convirtió en un negocio multimillonario que va más allá del intercambio de dinero entre distintos usuarios. Uno de los actores más importantes dentro del universo Bitcoin son los denominados **mineros**. Estos son los encargados de generar nuevos bloques que contienen las transacciones válidas (éste proceso lo detallaremos más adelante).

Tanto las transacciones como los bloques viajan por la red de este sistema distribuido de manera tal que todos los integrantes puedan tener la información relevante y conocer el estado del mismo. Cada cliente se copia la información localmente y luego la propaga al resto de la red. Con esta información, se llega a un consenso distribuido y todos convergen

a un único estado del sistema.

Ser parte activa de la red como **minero** es una actividad rentable para muchas personas y organizaciones alrededor del mundo. Esto tiene dos efectos remarcables, por un lado motiva la investigación y el desarrollo de nuevos protocolos para conseguir un sistema más robusto y rentable; pero por el otro, hay un gran incentivo para que el sistema no se modifique ante el riesgo de que estos actores puedan perder parte de los beneficios que consiguen actualmente.

Dado que el código de Bitcoin es abierto, cualquier persona es capaz de modificar el código y generar su propia versión. Pero como el sistema está gobernado por el software que utilizan los nodos, cualquier modificación no genera ningún cambio de por sí en el protocolo, a menos que tenga una adopción generalizada entre la comunidad.

De esta manera nace el concepto de *Hard Fork*. Cuando una parte de la comunidad decide agregar una nueva característica al sistema, realiza los cambios necesarios y, todos aquellos que deciden adoptar este nuevo protocolo, comienzan a ejecutar la nueva versión del sistema bifurcándose del protocolo original y partiendo la red en dos. Los que deciden no actualizar los cambios, siguen corriendo sobre la versión original con sus particularidades y aquellos que decidieron migrar a la nueva versión pasan a estar en otra red con su propio funcionamiento.

En la historia de Bitcoin, reiteradas veces se produjeron *Hard Forks*, que dieron nacimiento a nuevas criptomonedas como `Bitcoin Cash` o `Bitcoin Gold` que fueron lanzadas en el año 2017. Esto suele suceder porque parte de la comunidad cree que alguna modificación (por ejemplo el tamaño del bloque que puede permitir más transacciones) es beneficiosa para el sistema, y otra parte lo rechaza. Por lo tanto, se divide la red entre estos dos grupos y pasan a ser sistemas distintos.

Es complicado analizar el éxito de estos nuevos sistemas. Si nos dejamos llevar por la valuación del mercado y su capitalización, tienen un éxito relativo. Varios de estos *forks* se produjeron en un momento en donde el mercado de las criptomonedas estaba en alza

y su valor se vio impulsado por este fenómeno. Pero al momento de la baja ocurrida a fines de 2019, todas las criptomonedas tuvieron una pérdida sustancial de su valor. Hoy en día, `Bitcoin Gold` sigue por debajo de su precio de salida y `Bitcoin cash`, aunque está por arriba, no llega a valer una cuarta parte de su máximo histórico. Por su parte, en el mismo período, Bitcoin luego de tener una corrección del 80 % de su valor, hoy en día se multiplicó por cinco al valor que se encontraba al momento de producidos los mencionados *hard* forks[(1)].

Como alternativa a cambiar los protocolos o software del cliente generando *hard* forks, existen otros enfoques orientados a adecuarse a las condiciones del sistema mejorando otros aspectos del funcionamiento de dicho sistema. Entre ellas, podemos destacar algunas que trabajan sobre una red de despacho de bloques más eficiente que la del propio cliente oficial Bitcoin, denominadas **redes de retransmisión de bloques**. Es decir, es posible que los clientes utilizando redes de retransmisión puedan convivir con clientes que no las utilicen sin mayores inconvenientes. Estar conectado a estas redes de retransmisión, aportaría beneficios considerables al sistema en su conjunto sin modificar ningún aspecto del protocolo original. Estos beneficios o mejoras serían, principalmente, para aquellos que estén conectados pero, indirectamente, también habría mejoras para el resto de la red.

En esta tesis, queremos explorar y analizar los beneficios de utilizar una red de retransmisión de bloques como la planteada por `FIBRE`. A parte de analizar si el sistema mejora su eficiencia en cuanto al uso de recursos, nos enfocaremos en estudiar si este tipo de organización no implica cierto grado de centralización del sistema o si algún actor o conjunto de actores se ve beneficiado en desmedro del resto de los participantes.

Para esto vamos a utilizar herramientas de emulación, que permiten definir un sistema de escala media controlando todos sus parámetros de funcionamiento (desde rutas hasta latencia de cada enlace). De esta manera se podrá analizar su comportamiento y el impacto de contar o no con redes de retransmisión utilizando los clientes reales y controlando las condiciones de cada experimento.

[(1)] Se puede analizar su capitalización y ver los históricos en https://coinmarketcap.com/

A lo largo de la tesis vamos a realizar una descripción detallada del sistema completo que vamos a analizar, las herramientas utilizadas para realizar dicho análisis, los experimentos propuestos para resolver distintas preguntas y los análisis pertinentes de la información recolectada.

1.1 Bitcoin

Una criptomoneda es un medio de intercambio digital que utiliza tecnología criptográfica para asegurar la veracidad de las transacciones.

La transferencia de una moneda digital puede pensarse como el endosado de un cheque. Una persona puede escribir en el dorso del documento el nombre del destinatario, el cual podría endosarlo nuevamente si así lo desease. También es posible saber si la última persona en endosar el cheque es el dueño original del documento.

En el mundo electrónico, se puede lograr algo similar con firmas digitales y ***hashes* criptográficos**. Cuando una persona quiere transferir dinero digital a otra, se crea una **transacción**. Este proceso consta de la firma digital del *hash* criptográfico de la transacción anterior que utilizó ese dinero y la clave pública del nuevo destinatario para que pueda disponer de estos fondos. De esta forma, el destinatario puede verificar que el emisor era realmente el dueño del dinero, validando la firma digital de la transacción con el hash dado, y además, puede volver a transferir los fondos usando su clave privada.

En la figura 1.1 se presenta una esquema de este proceso, en donde se realizan tres transacciones de una criptomoneda. Vamos a hacer hincapié solo en los conceptos criptográficos utilizados para poder llevar adelante un intercambio de criptomonedas, y luego, en la sección 1.1.1.1 hablaremos más en detalle de la estructura de la transacciones.

`A` es la representación de una transacción en donde **Ana** recibió dinero por parte del **José** en algún momento del tiempo. Y `B` representa una transferencia entre **Ana** y **Juan** un tiempo después. No es relevante en este momento, pero el inicio de todas las cadenas de transacciones son las transacciones ***coinbase*** detalladas en la sección 1.1.2. Desde

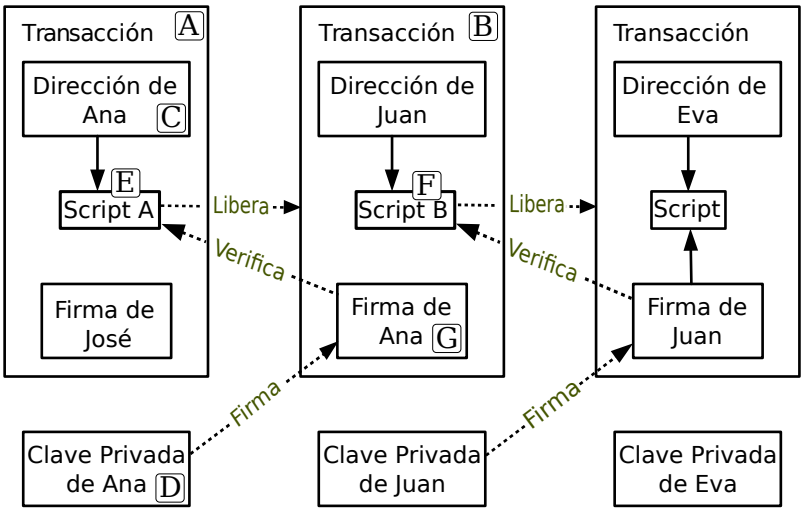


Figura 1.1: Esquema de cómo los bitcoins son transferidos entre distintos usuarios por medio de las transacciones entre desconocidos [Nak08].

la transacción *coinbase*, que es la única que solo tiene un único destinatario sin un emisor, las criptomonedas van pasando de un cliente a otro continuamente por medio de las transacciones.

`C` es la **dirección**[2] de Ana, que es simplemente un identificador para poder recibir criptomonedas. Que se encuentre dentro de la transacción `A`, implica que Ana es la destinataria de dicha transacción y por consiguiente, de los fondos enviados.

Ana con su **clave privada** `D`, es la única capaz de generar la **firma digital** `G` que se utiliza para desbloquear los fondos asociados a la transacción `A`.

`E` es el ***script A***[3] que vincula a la dirección de Ana con los fondos recibidos por José. Y para ser desbloqueado, se necesita la firma digital que demuestra que se tiene control de la clave privada de la dirección de Ana.

En la nueva transacción `B`, proveyendo la **firma digital** `G`, podemos liberar los fondos de la transacción `A` para que queden a disposición de Juan, bloqueados en el *script B* `F`, nuevamente a la espera de ser liberados con la firma correspondiente.

[2] Es la aplicación de una función de HASH a la clave pública del cliente X. Más específicamente RIPEMD160(SHA256(<Pub Key X>))

[3] Bitcoin utiliza un lenguaje de *scripting* para sus transacciones. La operación más común se denomina Pay-to-Public-Key-Hash que permite destinarle los fondos a una clave pública.

Este proceso se repite una vez más entre *Juan* y *Eva*, volviendo a reasignar los fondos de los clientes.

El sistema de transacciones de Bitcoin es mucho más complejo, debido a su lenguaje de *script* de transacciones. Permite realizar varios tipos de transacciones, no solo entre dos participantes. Pero para el alcance de esta tesis, es suficiente con entender este simple proceso entre únicamente dos participantes.

El diagrama anterior deja lugar a un problema conocido como **doble gasto** (*double spending*): el emisor podría crear más de una transacción a partir de una misma operación previa. En este escenario sólo una de las nuevas transacciones debería ser válida, ya que de lo contrario el emisor estaría multiplicando dinero. Una solución a este problema podría ser contar con una entidad de confianza que se encargue de validar todas las transacciones.

El sistema de Bitcoin fue la primer criptomoneda descentralizada que propuso una solución al problema de *double spending* sin necesidad de que haya una **tercera parte de confianza** (*trusted third-party*). La idea principal consiste en que exista un único registro de todas las transacciones que fueron procesadas, en el cual cualquier usuario puede agregar operaciones. Todos los nodos de la red van a guardar una copia de dicho registro.

Este registro replicado y distribuido se conoce como **Blockchain** y cada cliente mantiene su propia copia de la información. Consiste en un árbol que contiene transacciones en cada uno de sus nodos. Luego, si un usuario introduce una nueva transacción a la red, es fácilmente detectable si en la Blockchain existe otra operación que hace referencia a la misma transacción «padre». Si esto último ocurre, se considera que el usuario está intentando hacer *double spending* y la transacción es invalidada por los nodos de la red. En la figura 1.2 se puede apreciar una esquematización de la blockchain, en donde se ve cómo cada bloque que contienen información particular del mismo y las distintas transacciones, están entrelazados apuntando a su bloque padre.

El origen de toda la *Blockchain* de Bitcoin, es un bloque especial llamado *Bloque Génesis* (el número 0), el cual fue minado el 3 de enero de 2009. Este bloque contiene en uno de

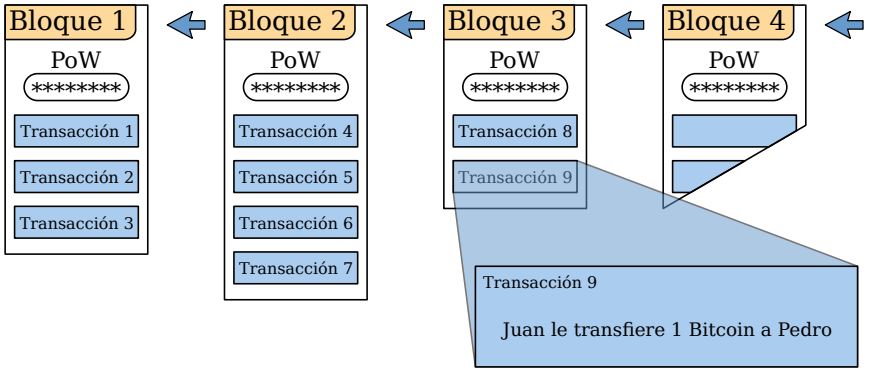


Figura 1.2: Representanción de la blockchain. Cada bloque incluye una referencia al bloque anterior junto con información particular del bloque y un conjunto de transacciones.

sus campos un titular del diario *«The Times»* de esa fecha[4], para dar prueba que este bloque no fue creado antes de ese día. A partir de éste, se encadenaron todos los bloques que contienen todas las transacciones ocurridas en el sistema y hoy en día tiene una altura de alrededor de 700.000 bloques[5].

1.1.1 Estructuras de datos

En esta sección se describirán en detalle las estructuras de datos utilizadas en Bitcoin. Estas estructuras son la base de todas las criptomonedas descentralizadas, por lo cual es importante analizar sus conceptos fundamentales.

1.1.1.1 Transacciones

Una **transacción** en Bitcoin está definida por una lista de *inputs* y una de *outputs*. Un *output* consiste en un par `(cantidad, dirección)`, 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, indice, clave, firma)]` hace referencia a *outputs* de transacciones anteriores. Cada uno de estos campos se interpreta de la siguiente manera:

- El *hash* de la transacción a la que hace referencia.

[4] «The Times 03/Jan/2009 Chancellor on brink of second bailout for banks»
[5] https://blockchair.com/es/bitcoin/blocks

- 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*.

De esta forma, cada *input* es una referencia a un *output* anterior en la blockchain. Para verificar que el uso del *output* es legítimo, se calcula el *hash* de la clave pública y se verifica que sea igual a la que figura en el *output* usado. Luego, basta con verificar la firma digital con esa clave pública para asegurarnos la autenticidad de la operación.

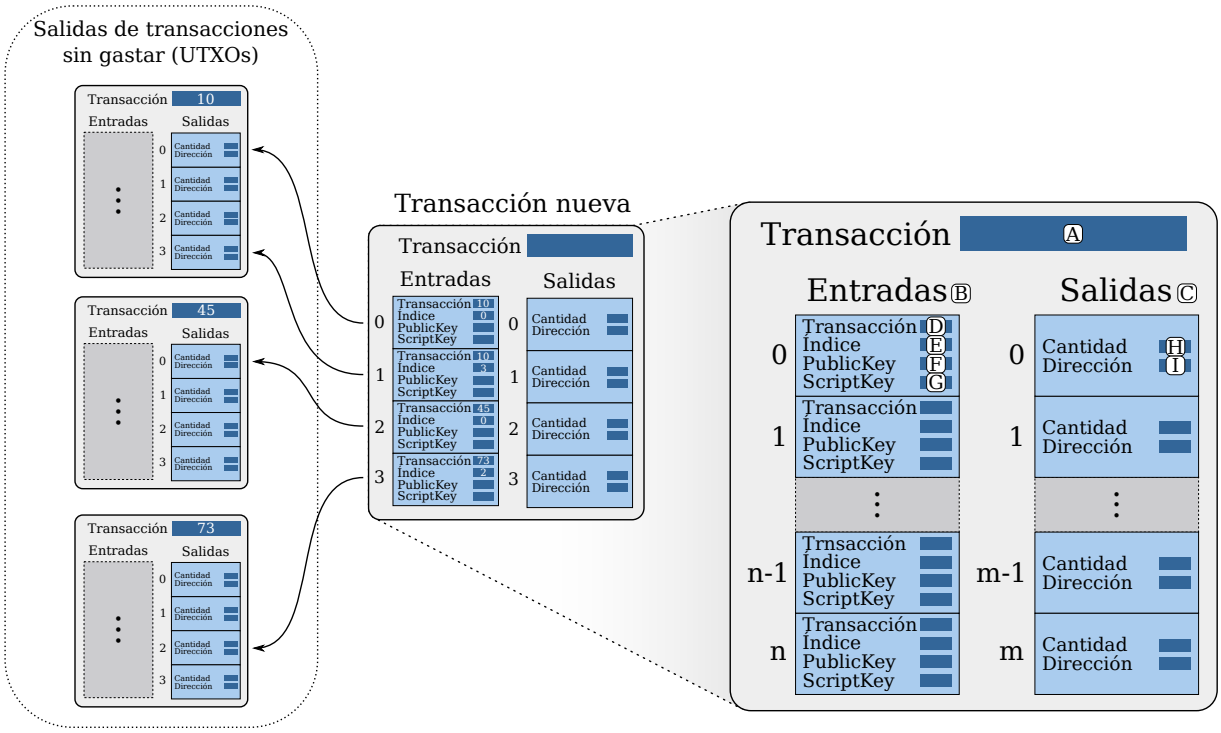


Figura 1.3: Ejemplo de una transacción en Bitcoin, sus *inputs* hacen referencias a *outputs* de transacciones anteriores.

En la figura 1.3 se puede ver un ejemplo de una transacción con varios inputs. Del lado 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, donde cada elemento hace referencia a un *output* de alguna transacción previa que todavía no haya sido usado.

(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* utilizado de la transacción referenciada.

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

(G) La firma digital que demuestra que se tiene control sobre los fondos utilizados.

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

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

Esta es la versión más sencilla de transacciones. Bitcoin soporta un mecanismo llamado *Pay To Script Hash (P2SH)* que permite dar un hash de un *script* que es almacenado como *address* en el *output*. Luego para usar este *output*, se deben proveer parámetros para ejecutar el *script* correspondiente al hash del *output*. Para que esta transacción sea válida, el *script* debe ejecutarse satisfactoriamente. Cabe mencionar que estos *script*s están limitados en el sentido en que no tienen ciclos, restringiendo el poder de expresividad.

Para poder afirmar que una transacción es válida, se debe verificar que todos los inputs también lo son, y que la suma de los *outputs* referenciados por esos inputs es mayor a la suma de los *outputs* de la transacción. La diferencia entre ambas sumas podrá ser reclamada por quien agregue la transacción a la Blockchain. Este dinero extra es lo que se conoce como *«transaction fees» (comisión)*.

Una vez que un *output* de una transacción fue utilizado, ya no se puede volver a usar. Es decir, solo se pueden incluir en nuevas transacciones *outputs* que no fueron utilizados previamente. Estos últimos se conocen como *«Unspent Transaction Outputs» (UTXOs)*.

1.1.1.2 Bloques

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

Visto desde otro modo, si tomamos la Blockchain como un árbol, los bloques son sus nodos.

Cada bloque está compuesto por:

- El *hash* del bloque anterior (su padre).
- Tiempo en segundos en que fue minado, contados a partir del *epoch*[6].
- La dificultad con la que fue hallado.
- `nonce`: Un campo en el que se puede poner cualquier número, permitiendo así modificar el resultado del cálculo del hash del bloque.
- Un conjunto de transacciones ordenadas.
- El número de versión.

No toda combinación de valores en estos parámetros resulta en un bloque válido. Para que un bloque sea válido y aceptado por la red debe contar con una prueba de trabajo, la cual se conoce como *Proof of Work (PoW)*. Ésta debe ser difícil de calcular pero fácil de verificar. El mecanismo que se usa para hallar y validar el `PoW` se conoce como *proceso de minado*, y será explicado más adelante.

Los bloques se separan en *header* y *body*. En Bitcoin, el *body* consiste en un *Merkle tree* cuyas hojas son los hashes de las transacciones contenidas en el bloque. Por otro lado, el *header* incluye el resto de los campos mencionados anteriormente, además del hash de la raíz del *Merkle Tree* del *body*.

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 1.4 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. *Hasheando* susesivamente la información como se muestra en la figura, cualquier cambio que se realice en alguna transacción, indefectiblemente cambia el resultado final del *Merkel Tree Hash*.

[6] 1 de Enero de 1970, a las 00:00:00 UTC

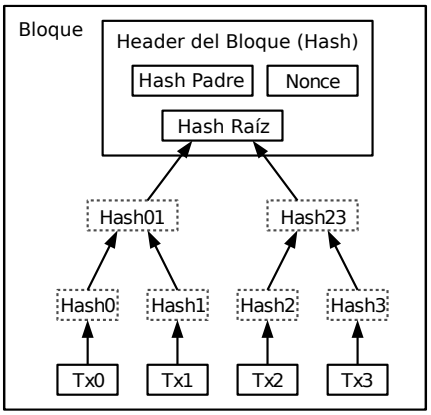


Figura 1.4: Representación del algoritmo de hash del Merkle Tree. Se calculan los hashs sucesivamente a partir de la información de las transacciones para obtener un único hash que representa al conjunto específico de las transacciones originales. La probabilidad de realizar alguna modificación en las transacciones del bloque y que el nuevo hash resultado sirva para el POW es prácticamente nula [Nak08].

1.1.1.3 Blockchain

La estructura de la Blockchain puede verse como un árbol de bloques. No puede ser considerada una lista, ya que podría suceder que dos bloques tengan el mismo padre, dando lugar así a ramificaciones. La Blockchain tiene una cadena[7] principal, llamada *main chain*, que es la cadena con mayor dificultad total (suma de dificultades de todos los bloques que la componen). Los bloques que no pertenecen a la *main chain* son llamados bloques *«stale»*.

El número de un bloque es la altura del bloque desde la raíz de la Blockchain. Los nodos que construyen bloques son denominados *nodos mineros*. Cualquier cliente de la red puede tomar este rol. Al recibir un bloque, los nodos lo validan y lo agregan a su Blockchain. Si el nuevo bloque tiene como padre a un bloque *stale*, puede cambiar la cadena que el nodo considera como principal. Este escenario ocurre si la adición de dicho bloque genera una cadena con mayor dificultad total que la previamente considerada como principal.

1.1.2 Proceso de minado y consenso

Los mineros obtienen una recompensa en bitcoins cuando agregan un bloque a la cadena principal de la Blockchain. Un minero debe calcular satisfactoriamente la PoW de un nuevo bloque para poder incluirlo en la Blockchain, gastando poder de cómputo en el proceso.

[7] Se entiende como cadena a la lista de bloques desde un nodo hasta la raíz del árbol.

La *Proof of Work* del bloque es un número tal que **SHA256(SHA256(header))**[8], tiene una cantidad dada de 0's en sus bits más significativos. En bloques válidos, esta cantidad es mayor o igual a un número que se deriva del campo *dificultad* del *header* del bloque[9].

La motivación de este proceso yace en el costo computacional de obtener un hash que cumpla los requisitos pedidos. `SHA` es una función criptográfica con la propiedad de que no existe una manera directa de calcular la función inversa, y por lo tanto la única forma de encontrar un hash válido es recalcularlo varias veces cambiando valores del *header*, hasta encontrarlo. Los hashes calculados están uniformemente distribuidos, por lo cual, la probabilidad de que un hash (256 bit) comience con d ceros, es $\frac{2^{256-d}}{2^{256}}$.

El esfuerzo necesario para armar un bloque que cumpla con la dificultad de la red crece exponencialmente con la cantidad de ceros requerida. Esto se debe a que agregar un cero extra a la dificultad disminuye a la mitad la cantidad de hashes que cumplan con esa restricción. Sin embargo, para verificar que un bloque cumple con esta propiedad, alcanza con calcular dos veces el `SHA256` del *header*. De esta forma, se puede comprobar fácilmente que un minero realizó una cierta cantidad de trabajo para hallar un bloque.

Es importante destacar que una PoW solo sirve para un bloque, ya que ésta 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 antelación.

Debido a que el poder 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 se espera que aparezcan nuevos bloques en la red (en Bitcoin es 10 minutos). Cada 2016[10] bloques, se cuenta cuántos aparecieron en menos

[8] `SHA256` es una función de hash criptográfico.
[9] Técnicamente el campo dificultad es un número de 256 bit y el hash del bloque debe ser menor a este número. Decir que debe empezar con una cierta cantidad de ceros es una simplificación del requerimiento.
[10] Debido a un error de software, en Bitcoin solo se revisan los últimos 2015 bloques

tiempo que el *target*, y en función de esto se aumenta o disminuye la dificultad hasta cuatro veces el valor anterior. Esto se traduce en que el hash del *header* del bloque comenzará con hasta dos ceros más o menos que el padre, según aumente o disminuya la dificultad, respectivamente.

Bitcoin provee incentivos para que los mineros colaboren con la red. Por un lado, el minero que construye un bloque válido es acreedor de todas las comisiones de las transacciones incluidas en él. Por el otro, cada bloque cuenta con una transacción especial llamada *coinbase*, la cual tiene un solo *output* y carece de inputs. El minero tiene control total sobre los campos de esta transacción. Por lo tanto, el nodo que descubre un bloque puede asignar una dirección de su billetera al *output* de la *coinbase*, convirtiéndose así en acreedor del valor de la transacción. Ésta es la única forma de crear bitcoins, y es por eso que se conoce al proceso como «Minar Bitcoins», ya que el minero en realidad está introduciendo nuevos bitcoins al sistema. El *output* de la *coinbase* sólo puede ser usado luego de que hayan aparecido 100 bloques por encima del que la contiene. Esta política trata de evitar que se obtenga dinero a partir de bloques *stale*. Sin embargo, esta decisión de diseño no es parte del protocolo de consenso de la red. Un usuario puede gastar ese crédito, sabiendo que en un futuro cercano podría quedar invalidado.

La cantidad de bitcoins otorgada por minar bloques disminuye en función del número de bloque. Cada 210000 bloques (cuatro años aproximadamente) se divide a la mitad, comenzando con un valor de 50BTC por bloque. Esta política de reducción resulta en que existe una cota sobre la cantidad total de bitcoins que pueden existir en el sistema, la cual es aproximadamente 21 millones.

El 12 de Mayo de 2020 se redujo nuevamente a la mitad la recompensa por minar un nuevo bloque. Pasó de otorgar 12,5BTC por cada nuevo bloque minada, a 6,25BTC (equivalente a U$S55000).

Los mineros confeccionan un nuevo bloque con algunas de las transacciones válidas que

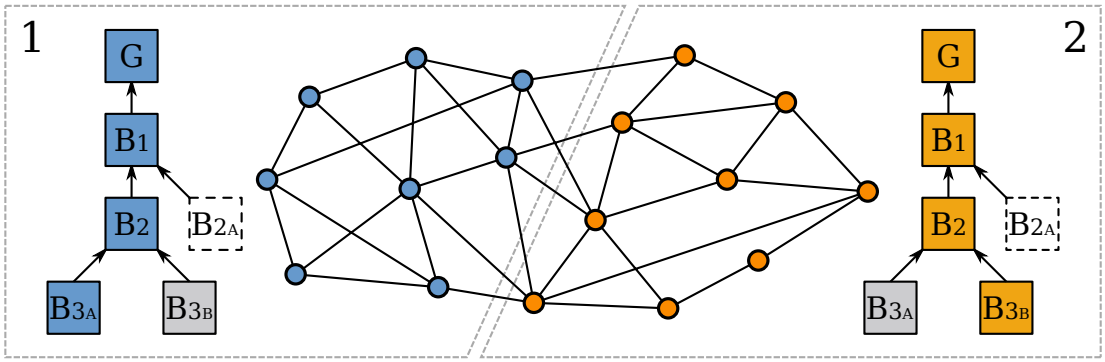


Figura 1.5: Ejemplo de una partición de los nodos de la red. El grupo de nodos 1 considera al bloque B_{3A} como bloque de su cadena principal, mientras que el grupo 2 considera a B_{3B}. El bloque B_{2A} quedó *stale* luego de que se minaron los bloques B_{3A} y B_{3B}.

todavía no pertenecen a la *Blockchain* y comienzan a incrementar el *nonce*[11] intentando validarlo. Cuando un minero encuentra un hash que hace válido al nuevo bloque, propaga dicho hash por toda la red. Luego los demás nodos piden el bloque completo, lo validan, lo agregan a la Blockchain y lo propagan de la misma manera.

Debido a la naturaleza distribuida de Bitcoin, es posible que dos nodos distintos de la red encuentren bloques válidos diferentes con la misma altura dentro de la Blockchain. Esto sucede cuando un minero genera un bloque antes de que sea notificado de la existencia de otro bloque con la misma altura. Ante esta situación, los nodos de la red deberán decidir cuál de los dos bloques forma parte de la cadena principal. En Bitcoin, ante dos ramas competitivas, gana la que tiene mayor dificultad total (suma de las dificultades de los bloques), y ante un empate, cada nodo se queda con la que recibió primero (*first-seen rule*). Esto es lo que se conoce como *Protocolo de Consenso*.

Esta situación deriva en una partición en los nodos de la red, como se puede observar en la figura 1.5, donde un grupo de nodos tiene una cadena principal con el último bloque el *B3a*, y el otro grupo de nodos contempla al bloque *B3b* como el final de la cadena. Cada grupo de nodos minará sobre la que considere su cadena principal. A esta situación se la conoce como un *fork* de la red. Eventualmente, sucederá que una de las dos cadenas tenga una dificultad total mayor (porque se minaron más bloques en esa cadena), y de a poco

[11] En la práctica no basta solo con incrementar el *nonce*, ya que este campo solo cuenta con 32 bit, es posible que se itere completamente y no se cumpla con la dificultad pedida. Se modifican otros campos también, como el tiempo del bloque y la transacción *coinbase*.

todos los mineros van a aceptar dicha cadena como su cadena principal. Luego, al minar sobre ésta, tendrá una mayor dificultad acumulada y, por lo tanto, será la aceptada por toda la red.

Cuando un bloque es agregado a la Blockchain, cada bloque que se agregue por encima de él (es decir, que lo tenga como un ancestro), aumenta la probabilidad de que ese bloque pertenezca a la mejor cadena. Por ejemplo, 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), y se recomienda que para estar seguros de que el bloque va a pertenecer a la mejor cadena, se espere a que se minen por lo menos seis bloques sobre 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 sobre el bloque en donde aparece la transacción de interés. Si un atacante quisiese revertir esa transacción, debería agregar un bloque cuyo padre sea el mismo que el del bloque que la contiene y, adicionalmente, crear tantos bloques como los que hayan sido agregados en la cadena original. Es decir, el atacante tiene que controlar más de la mitad del poder de cómputo de la red para poder crear una cadena que sea más larga que la creada por los nodos honestos. A este ataque se lo conoce como un ataque del 51 %, y no se conocen mecanismos para mitigarlo.

1.2 Redes de retransmisión

Bitcoin fue diseñado como una red *peer-to-peer*, donde los nodos se conectan entre sí de manera aleatoria. Al iniciar el cliente, *bitcoin* tiene varias formas de conectarse con otros *peers* de la red.

La primera vez que se ejecuta el protocolo y el cliente no tienen ninguna información recolectada de la red, comienza un algoritmo de descubrimiento de *peers*. Se comunica con un servidor de nombres (`DNS`) para que le suministre direcciones `IP` de clientes supuestamente

activos y establecer una conexión. Si no recibe ninguna respuesta y pasan 60 segundos sin poder conectarse con ninguno, el código tiene algunas direcciones `IP` *harcodeadas* para utilizar. Una vez conectado con algún cliente de la red, le pregunta por más clientes disponibles, los agrega a su base de datos y comienza a formar parte activa de la red de Bitcoin. Cuando reinicia el protocolo por el motivo que sea, intenta por 11 segundos conectarse a aquellos *peers* que tiene en su base de datos local, los cuales fueron recolectados a medida que interactuó con la red. Si por algún motivo ninguno de sus *peers* le responde, vuelve a realizar el descubrimiento que hizo la primera vez.

Las transacciones y los bloques son propagados sobre esta red por los nodos que la integran. Esto funciona bien, haciendo que sea un sistema completamente distribuido, sin que nadie sea capaz de controlarlo. Pero también tiene sus desventajas, la red *peer-to-peer* es relativamente lenta. A veces los mineros desperdician recursos tratando de extender la *Blockchain* sobre un bloque que no termina formando parte de la cadena principal, mientras un nuevo bloque válido ya está circulando por la red. Vamos a detallar más información sobre esto en la sección 1.4, en donde se hacen distintos análisis del tiempo de propagación, los *forks* y el tiempo desperdiciado por los mineros.

Desde ya hace algunos años, aparecieron lo que se conoce como *«Relay Networks» (Redes de retransmisión)*, orientadas a incrementar la velocidad de propagación de los bloques. Trabajan principalmente con dos conceptos: i) la compresión de bloques, para disminuir la cantidad de datos que necesitan ser transmitidos sobre la red y por consiguiente reducir los tiempos de transmisión, ii) la velocidad de transmisión, para disminuir el tiempo que tardan en propagarse los bloques.

Estos proyectos se componen por un grupo reducido de nodos, como está esquematizado en la figura 1.6. Su propósito es retransmitir a todos sus *peers* los bloques y transacciones que les llegan del resto de la red. Las redes de retransmisión apuntan a que todos los clientes de la red estén conectados a alguno de sus nodos. Si se sigue esta estrategia, se reduce drásticamente el diámetro de la red, el cual afecta en los tiempos de propagación y de aceptación de la red, que a su vez influyen en la aparición de *forks* y distintos fenómenos

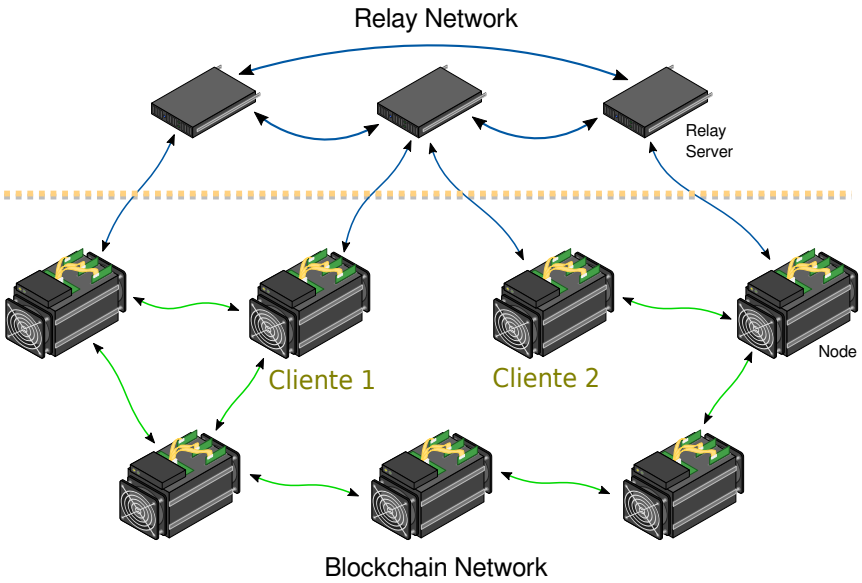


Figura 1.6: Esquema de una red de retransmisión de bloques funcionando junto al protocolo de Bitcoin. Podemos observar como la distancia mínima entre cliente 1 y el cliente 2 en la red `P2P` de Bitcoin, es de cinco *hops*. Utilizando la red de retransmisión, esta distancia pasa a ser de dos *hops*.

que están directamente vinculados con el desperdicio de poder de cómputo.

Podemos ver en la figura 1.6 que, de no existir la red de retransmisión de bloques, aquellos bloques minados por el minero **Cliente 1** tendrían que pasar por lo menos por cuatro clientes antes de llegar al minero **Cliente 2**. Teniendo la red de retransmisión funcionando, la comunicación de bloques entre los mineros **Cliente 1** y **Cliente 2** pasa por un solo servidor de la red.

Entre algunas redes de retransmisión podemos nombrar a *Bitcoin Fast Relay Network (BFRN)*[(12)], la cual fue la predecesora de `FIBRE`. Contaba con nueve nodos ubicados estratégicamente alrededor del mundo. Los mineros podían conectarse al nodo más cercano para recibir y enviar bloques dentro de esta red. Su creador, Matt Corallo, alienta a que otros provean recursos para armar redes de retransmisión propias poniendo a disposición el código[(13)]. Su idea no es reemplazar la red `P2P` original del protocolo y que estas nuevas redes de retransmisión no tiendan a centralizar el protocolo, si no mejorarlo y hacerlo más robusto.

[(12)] https://bitcoinrelaynetwork.org
[(13)] https://github.com/TheBlueMatt/RelayNode

Los bloques en la red de Bitcoin son transmitidos en varios paquetes `IP`. Todos los clientes en la red reciben cada paquete y reconstruyen el bloque. Solo cuando pudieron reconstruir el bloque y validarlo, cada cliente comienza a retransmitirlo nuevamente a otros nodos. *Falcon Relay Network*[14] utiliza un técnica denominada *cut-through routing*, donde los nodos no esperan a recibir todos los paquetes para retransmitirlos. Va retransmitiendo los paquetes a medida que los recibe, mientras inicialmente solo verifican los *headers* del bloque. Su desventaja es que solo pueden validar el bloque completo cuando hayan recibido todos los paquetes correspondientes, corriendo el riesgo de transmitir paquetes inválidos (por ejemplo los de un bloque que contenga una transacción ya utilizada) a través de la red, desperdiciando recursos.

Por último haremos una descripción detallada de `FIBRE` en la sección 1.2.1, la cual utilizaremos para el desarrollo de esta tesis. Elegimos este proyecto debido a que no solo está en funcionamiento hace algunos años, si no que es *open source* y disponemos del código fuente para poder montar nuestra propia red privada de retransmisión.

1.2.1 Fibre

The Fast Internet Bitcoin Relay Engine conocido como **FIBRE**[15], está compuesta por cinco servidores distribuidos a través del hemisferio norte como se muestra en la figura 1.7 que exponen en su página web.

Funciona como un agregado a la tradicional red `P2P` de Bitcoin. Es decir, los clientes que se conectan a alguno de los nodos de `FIBRE` no dejan de usar el método estándar para establecer conexiones con otros clientes de la red. Entonces, si en algún momento la red `FIBRE` dejara de funcionar, los clientes no perderían su conectividad, ya que en ningún momento dejan de utilizar la red principal de *Bitcoin*.

`FIBRE` se enfoca en un posible problema que tiene Bitcoin al utilizar el protocolo de `TCP` para realizar la comunicación de sus bloques. Cuando un paquete se pierde en una comunicación

[14] https://www.falcon-net.org
[15] http://bitcoinfibre.org

Figura 1.7: Distribución de servidores FIBRE alrededor del mundo. Imagen obtenida de la página oficial del protocolo http://bitcoinfibre.org

TCP-IP, el que lo envía no es capaz de enterarse hasta pasado el *Round Trip Time*, para recién luego intentar retransmitir dicha información. Para solucionarlo, FIBRE propone agregarle al protocolo original de Bitcoin, una red de retransmisión de bloques con algunas modificaciones.

Utiliza compresión de bloques[16], para disminuir la cantidad de datos a transmitir sobre la red. Por otro lado, propone utilizar *User Datagram Protocol* (UDP) para la transmisión de información y el envío de datos extras (*Forward Error Correction*) para compensar la eventual pérdida de paquetes del protocolo UDP, agilizando el envío de los bloques.

Su utilización puede llegar a resultar polémica principalmente por dos motivos. Por un lado, la red impone diferentes condiciones de latencia dependiendo del lugar del mundo desde donde operan los clientes. Como todos los servidores de FIBRE están en el hemisferio norte, los bloques de un cliente minando en el hemisferio sur necesariamente van a tener tiempos de propagación mayores a los de bloques descubiertos en el hemisferio norte.

Por otro lado, las redes de retransmisión agregan un cierto grado de centralización a la red

[16] https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki

de Bitcoin, ya que una misma persona es dueña de todos los servidores de la red `FIBRE`. Esta situación no necesariamente implica una centralización completa, porque si `FIBRE` descartara algunos bloques de la red, estos bloques eventualmente llegan a destino mediante la red `P2P` tradicional. Sin embargo, entre otras cosas, `FIBRE` podría retrasar (hasta cierto punto) la retransmisión de bloques, y nadie se podría dar cuenta, ya que la red `P2P` original se supone mucho más lenta. Entonces, el dueño de `FIBRE` podría decidir empeorar las condiciones de red para un cliente en particular, retrasando solo la retransmisión de sus bloques. Esta disparidad de poder hace que la red no sea completamente descentralizada, en contraposición a la idea original de Bitcoin.

1.3 Propósito de esta tesis

El propósito de esta tesis es investigar cómo se comporta Bitcoin con el agregado de una red de retransmisión de bloques como `FIBRE`. Vamos a suponer un comportamiento legítimo de todos los actores involucrados. No se contemplará la posibilidad de que alguno realice alguna acción que le permita explotar alguna cualidad del sistema en beneficio propio.

De esta manera, vamos a construir una red de Bitcoin en donde se realicen transacciones y generen nuevos bloques periódicamente. Para poder relevar información del funcionamiento de la red `FIBRE`, vamos a correr la misma configuración de los experimentos con la red `FIBRE` activada y desactivada.

Queremos intentar cuantificar si el agregado de esta red de retransmisión le aporta algún tipo de beneficio al sistema de *Bitcoin*. Como pueden ser disminuir el tiempo de convergencia de la red y la aparición de *forks*, el tiempo de propagación de lo bloques dentro la red o la cantidad de recursos desperdiciados del sistema. E intentaremos ver si se puede medir el nivel de centralización del sistema y si algunos actores dentro del mismo se ven beneficiados o perjudicados por el accionar de la red de retransmisión. Verificaremos si algunos actores son capaces de recibir los bloques anticipadamente y si esto implica que su desperdicio de recursos mucho menor al del resto.

1.4 Trabajo relacionado

Decker y Wattenhofer [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 s, 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 propagación. 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. [DPSHJ14] también mide el tiempo de propagación de bloques, pero haciendo foco en la topología de la red y buscando obtener la distribución de direcciones IPs de los nodos Bitcoin por país. Para ello, su cliente envía periódicamente mensajes de `GetAddr` (interno del protocolo de Bitcoin) a fin de obtener las direcciones `IP` 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 el porcentaje de bloques que se propagó antes de un tiempo dado a un dado porcentaje de nodos.
- Encuentran 87 000 direcciones `IP` distintas de nodos de Bitcoin, de las cuales solo 6000 se mantuvieron constantes a lo largo del experimento.

Miller et al [MJ15] 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 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 a 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 medidas se encuentra a dos 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.

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

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, GKL16] 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. [MJ15].

Silvio Vileriño [Vn17] diseña una metodología para el estudio de sistemas basados en blockchain. Luego utiliza la misma para experimentar sobre los límites del funcionamiento de Ethereum cuando se reduce el tiempo entre bloques.

Dihn et al. [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.

Kai Otsuki et al. [OABS19] analizan los efectos de las redes de retransmisión, utilizando

SimBlock, un simulador de redes blockchain. Realizando una simplificación del modelo de red de retransmisión, hacen simulaciones con 6000 clientes de bitcoin, y analizan el tiempo de propagación y la cantidad de *forks* variando la relación entre clientes conectados a la red de retransmisión. Sus conclusiones muestran cómo las redes de retransmisión disminuyen tanto el tiempo de propagación de los bloques dentro de la red, como la cantidad de *forks* en la misma y también no perciben beneficio alguno de aquellos clientes que están dentro de la red se vean beneficiados.

Nicolas DeCarli en [DC19] analiza la viabilidad de modificar el *target time* de la red de Bitcoin, para mejorar el *throughput* del sistema. Utilizando *SherlockFog*, realiza experimentos con una red emulada de Bitcoin de hasta 480 clientes.

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

Capítulo 2

Metodología

Este capítulo está dedicado a explicar las herramientas y métodos utilizados en la realización de los experimentos de este trabajo.

Explicamos las modificaciones realizadas al cliente real de Bitcoin utilizado en los experimentos, el cual intentamos manipular lo menos posible para que represente fielmente al protocolo original. Tuvimos que deshacernos del esfuerzo que implica el minado real y el ajuste de dificultad, para que con los recursos que disponíamos poder trabajar con redes de 200 clientes. Modificamos algunos aspectos de su funcionamiento interno y le agregamos nuevas funcionalidades para poder recolectar la información necesaria para la obtención de datos relevantes a las métricas a cuantificar.

Como pudimos ver anteriormente, dentro de la red de Bitcoin, la transferencia de valor se realiza por medio de transacciones. Éstas son creadas entre participantes, luego intentan ser agregadas en algún bloque para que las mismas queden asentadas dentro de la *blockchain* y pasen a ser una transacción válida para todo el sistema. Queremos realizar experimentos que nos puedan dar información sobre la influencia de una red de retransmisión como *FIBRE* en el protocolo de Bitcoin real, por lo tanto, vamos a generar transacciones y bloques que van a circular dentro de la red llegando a todos los participantes.

Para correr los experimentos, necesitamos tanto una red física como lógica. Nos enfocamos en los distintos modelos utilizados, fundamentando las decisiones tomadas y explicando cómo construirlos para conseguir una red lo más similar posible a la real. Introducimos la herramienta `Sherlock Fog`, que nos sirve para la emulación de la red y permite definir la topología de conectividad que queramos así como la latencia entre los enlaces.

También presentamos y discutimos las magnitudes a medir en los experimentos, con las cuales vamos a validarlos y recolectar información relevante a las incógnitas planteadas en esta tesis.

Por último, describimos el diseño experimental utilizado en detalle, explicando las configuraciones que se utilizaron en cada uno de ellos y por qué elegimos dicha configuración. De la misma forma, hacemos un repaso de cómo conseguimos automatizar los experimentos, detallando las tecnologías utilizadas y los módulos de software desarrollados para tal propósito. Todo esto no se puede llevar a cabo si no contamos con una infraestructura real donde correr los experimentos, la cual se detallada al final del capítulo para poder comprender las limitaciones que tuvimos.

2.1 Modificaciones al cliente

Tomamos el cliente que provee la página oficial de *FIBRE*[1] como punto de partida. Éste se basa en el cliente `BTC Core v0.16.3` con el agregado de *«Remote Procedures Calls» (RPC)* para la interconexión de la red *FIBRE*.

Por medio de la configuración inicial del cliente Bitcoin, se puede agregar la opción `-udpport` que define un puerto UDP sobre el cual se va a realizar la comunicación de paquetes dentro de la red *FIBRE*. Luego, se establecen las conexiones dinámicamente con el llamado RPC `addudpnode`, con el cual se genera la topología de la red *FIBRE*. Por último, también cuenta con opciones de *debugging* específicas de *FIBRE*, que se pueden agregar para que sean visibles en archivo `debug.log` del cliente de BTC Core.

[1] https://github.com/bitcoinfibre/bitcoinfibre

Cambiamos algunos aspectos de su funcionamiento interno y le agregamos nuevas funcionalidades para poder recolectar la información necesaria para los experimentos. Modificamos el código del cliente Bitcoin para manipular la dificultad de la red y poder realizar el minado de bloques sin desperdiciar tiempo de cómputo. A continuación, explicamos cómo logramos realizar el *Proof of Work* sin necesidad de invertir poder de cómputo para dicho propósito.

Implementamos, dentro del cliente, un módulo propio de manejo de *logs* que registra todos los datos necesarios de los experimentos. Por último, realizamos cambios de protocolo a la red **regtest** del cliente, la cual es una red privada incorporada en el cliente oficial. Esta red tiene parámetros predeterminados distintos a la **mainnet** (la red de producción), y se usa principalmente para hacer pruebas y experimentos. Como en nuestros experimentos queremos emular a la red real, listamos todos los cambios hechos para que la red *regtest* de nuestro cliente se comporte como la *mainnet*.

2.1.1 Acondicionamiento de la red Regtest

Cuando se inicia el cliente, se le puede indicar que use una de las tres redes disponibles: *mainnet*, *testnest* o *regtest*. La primera es la red oficial, en la cual cada bitcoin cotiza en el mercado. La segunda es una red pública global paralela a la *mainnet*, los bitcoins de esta red no tienen un valor de conversión a monedas reales. La última es una red privada local, la cual se usa para testear y *debuggear* la mayoría de los aspectos del cliente. En nuestros experimentos vamos a usar la red *regtest* que, al ser privada y local, nos da un ambiente controlado.

Como la red *regtest* está pensada para hacer tests, viene configurada con parámetros distintos a *mainnet*. Por ejemplo, la frecuencia con la que la recompensa de minado es dividida por dos es diferente. Como queremos emular la red real, todos los parámetros de configuración fueron elegidos con los mismos valores que en la red *mainnet*.

Además de la discrepancia en los parámetros de la red, existían diferencias en el funcio-

namiento del programa cuando se ejecuta usando *regtest* en lugar de *mainnet*. En *regtest*, cada vez que se validaba una transacción, no solo se realizaban los chequeos relacionados a la nueva operación, sino que también se verificaban todas las que ya estaban asentadas en la *blockchain*. El mismo mecanismo cuadrático se efectuaba en la validación de bloques, verificando todos los anteriores. Estos chequeos fueron deshabilitados ya que tampoco se efectúan cuando el cliente utiliza *mainnet*.

Habiendo realizado los cambios mencionados, podemos usar *regtest* para realizar experimentos, y obtener resultados válidos para la red real.

2.1.2 Manejo de logs

El cliente posee un sistema de manejo de *logs* con fines de *debugging*, éste recolecta mucha información irrelevante para nuestros experimentos y no registra datos que sí necesitamos.

Por lo tanto agregamos al cliente un nuevo sistema de manejo de *logs* para poder medir el comportamiento de la red al realizar los experimentos. Éste consiste en dejar registrada información en un archivo cada vez que ocurre un evento relevante. Cuando el cliente se entera mediante un *peer* que apareció un nuevo bloque, deja registrado su hash. Luego, al momento de asentar ese bloque en la blockchain local, en una nueva entrada se vuelve a escribir el hash, acompañado del hash del padre. Por último, si el programa mina un nuevo bloque, se agrega al *log* la cantidad de transacciones que contiene, su hash y el hash del padre. Además de la información mencionada, todas las entradas del *log* tienen un *timestamp* que indica en qué momento ocurrió el suceso.

En resumen, las tres entradas posibles en el *log* son las siguientes:

- Al minar un nuevo bloque:
  <Hash bloque creado><Hash padre><Cantidad transacciones><Timestamp>
- Al descubrir un hash de un nuevo bloque:
  <Hash bloque descubierto><Timestamp>

- Al asentar un nuevo bloque proveniente de la red:
  <Hash bloque aceptado><Hash padre><Timestamp>

Utilizando los hashes de bloques y los ancestros podemos reconstruir la *blockchain*. Luego, teniendo ya el árbol completo, podemos calcular la información relacionada a *forks* detallada en la sección 2.4.1

Por otro lado, usando los *timestamps* de creación, descubrimiento y aceptación, podemos saber los tiempos de propagación de cada bloque a lo largo de la red. Los mismos son detallados en la sección 2.4.2

Además, usando los *timestamps* de creación podemos saber cuál fue el tiempo entre bloques promedio. Esta magnitud nos sirve de control como se detalla en la sección 2.4.5.

Por último, la cantidad de transacciones por bloque nos permite validar que sea, en promedio, aquella que configuramos para dicho experimento. Este dato también es mencionado en la sección 2.4.5, ya que forma parte de la validación de los experimentos.

2.1.3 Simplificando el *Proof of Work*

Para implementar el proceso de minado simulado descripto en la sección 2.2.1, necesitamos poder generar bloques válidos en tiempo constante.

En BTC Core existe la función `GetNextWorkRequired`, la cual determina cuál debe ser la dificultad del próximo bloque. Ésta es utilizada, principalmente, para validar que los bloques que llegan de los *peers* tengan un *hash* que cumpla con la dificultad de la red. Este método también es utilizado para determinar la dificultad actual de la red al momento de minar nuevos bloques.

Modificamos `GetNextWorkRequired` para que siempre retorne la dificultad mínima. De esta manera no modificamos la lógica de recalculado de dificultad. Tampoco fue necesario cambiar aspectos de la generación y validación de bloques, por lo que la *blockchain* construida conserva todas las características propias del sistema Bitcoin.

Por último, la red real tiene una dificultad mínima determinada, que es utilizada cuando el ajuste de dificultad resulta en un valor menor o igual. En el cliente cambiamos este número a $2^{256} - 1$, esto permite asignar dificultades menores a la mínima posible en la red real.

Con estas modificaciones, no desperdiciamos tiempo y poder de cómputo buscando un *nonce* para generar un nuevo bloque válido. El primer *hash* calculado para cualquier bloque va a cumplir con los requisitos de dificultad, y por lo tanto, minar un bloque válido va a ser solo una cuestión de agregar transacciones y llenar cada campo del *header* correctamente.

De esta manera, no modificamos nada intrínseco al funcionamiento del protocolo Bitcoin, pero conseguimos realizar el *Proof of Work* sin necesidad de desperdiciar poder de cómputo.

2.2 Modelando el comportamiento de Bitcoin

En esta sección vamos a detallar cómo modelamos el funcionamiento de *Bitcoin*. Queremos realizar análisis, principalmente, sobre los bloques generados en la red, su distribución y cuáles terminan perteneciendo a la cadena principal.

Explicamos cómo emulamos la generación de bloques intentando imitar lo que sucede en el sistema real.

Para intentar reflejar el comportamiento de la red real, también vamos a generar transacciones entre participantes, para que las mismas viajen por toda la red, ocupando espacio en la comunicación y el procesamiento de información. Esto lo hacemos para que los bloques no viajen vacíos por la red y el sistema haga casi todo el trabajo para el cual el protocolo está diseñado, salvo el *Proof of Work* como aclaramos anteriormente.

2.2.1 Minado simulado

Planteamos un modelo estadístico que simula el comportamiento de minado de bloques. Esto nos permitió emular el funcionamiento de la red de Bitcoin sin tener que invertir poder de cómputo real.

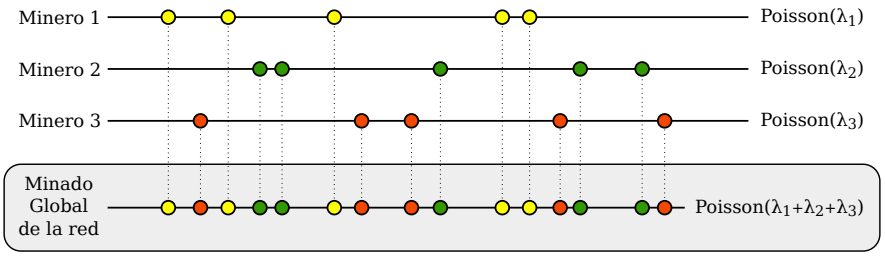


Figura 2.1: Suma de tres procesos de Poisson independientes. Cada línea simula el proceso de minado a lo largo del tiempo. El resultado es, también, un proceso de Poisson donde su parámetro característico es la suma del de los tres procesos de Poisson que lo componen.

Como los *hashes* calculados durante el minado de bloques son criptográficamente seguros, podemos suponer que están distribuidos de manera uniforme. Entonces, cada vez que computamos un *hash* tenemos la misma probabilidad de que sea un valor que cumpla con la dificultad de la red. Por lo tanto, la acción de calcular un *hash* puede modelarse como un ensayo de *Bernoulli*.

Dado que minar consiste en la repetición de estos ensayos en intervalos cortos de tiempo, el acto de minado de un cliente puede pensarse como un proceso de *Poisson* caracterizado por el parámetro λ_i, que indica la cantidad de bloques que el nodo i genera por unidad de tiempo. Si la red tuviera n mineros independientes en la red, habría n procesos Poisson independientes.

Como indica la figura 2.1, siendo $\lambda_1, \ldots, \lambda_n$ los parámetros para cada uno de los mineros de nuestra red, en la visión global donde todos los mineros están ejecutando en forma paralela, el proceso de minado es un proceso de Poisson con parámetro $\lambda_g = \sum_{i=1}^{n} \lambda_i$.

En un proceso de Poisson de parámetro λ_g, la cantidad de tiempo que transcurre entre cada par de eventos sigue una distribución exponencial con media $\frac{1}{\lambda_g}$. Entonces, si queremos que el tiempo entre bloques promedio de la red sea de 600 s, tendremos que $\frac{1}{\lambda_g} = 600$, resultando en que $\lambda_g = \frac{1}{600}$. El modelo no tiene restricciones sobre la forma en que distribuimos el *hashing power* de la red entre los mineros que actúan.

2.2.2 Modelo de transacciones

La forma de intercambiar valor entre los participantes del sistema *Bitcoin*, es a través de las transacciones. Pudimos ver en la sección 1.1.1.1 cómo funcionan y cómo se comportan.

Ahora vamos a mostrar cómo hacemos para que dentro de nuestro sistema se generen transacciones que van a ser transmitidas por los clientes dentro de la red y eventualmente quedarán impactadas en la *Blockchain*.

Decidimos que todos los clientes involucrados en el sistema se encarguen de generar transacciones para distribuir esta carga de trabajo equitativamente. De esta manera, cada cliente va a estar generando algunas transacciones periódicamente, con el objetivo global de que el promedio de transacciones por bloques oscile entre los valores que maneja actualmente *Bitcoin*.

Para esto cada cliente va a tener que disponer de fondos para realizar dichas transacciones, y luego comenzar a generar las transacciones periódicamente como se detalla a continuación.

2.2.2.1 Fondeo de clientes

Para que todos los clientes dispongan de fondos para realizar transacciones continuamente a lo largo de todo el experimento y haya una transmisión y validación de las transacciones por parte de cada cliente, se decidió utilizar una etapa de inicialización del mismo. Durante esta etapa no se toman mediciones que correspondan con los resultados finales del experimento. Solamente es utilizada para establecer la infraestructura necesaria para el desarrollo posterior del mismo.

Se inicializan todos los clientes que intervienen en el experimento con su *Hashing Power* preestablecido. Luego se arma la red lógica de Bitcoin pertinente y, si corresponde, la red *FIBRE* conectando los clientes entre sí. A cada cliente se le asignan dos billeteras en las cuales va a recibir fondos para poder luego realizar transacciones. Esta decisión se tomó para duplicar la cantidad de *inputs* con los que contaba cada cliente para realizar transacciones en el experimento.

En un principio, la red está vacía tanto de bloques como de transacciones. Un cliente especial se conecta a todo el resto de los clientes y luego genera 990 bloques repartiendo las recompensas obtenidas por el minado de dichos bloques entre todas las billeteras del sistema.

Por ejemplo, si contamos con un sistema de 200 clientes, la recompensa de cada bloque que es de 50 BTC es dividida entre las 400 billeteras del sistema. De esta manera, se repartieran 0,125 BTC a cada billetera.

Una transacción *coinbase* (la recompensa por minar un bloque) debe esperar a tener 100 bloques por encima para poder ser utilizada como saldo en una transacción. Entonces, cada cliente dispone de 890 transacciones asociadas a cada una de sus billeteras, para luego poder utilizarlas como *input* en sus nuevas transacciones.

El sistema espera a que todos los clientes reciban la información de los 990 bloques minados antes de comenzar su funcionamiento normal. Una vez que todos tienen actualizada su *blockchain* habiendo recibido los 990 bloques, el cliente que generó dichos bloques se desconecta del sistema y el resto comienzan a generar bloques como se detalla en la sección 2.2.1 y transacciones como explicaremos en la sección 2.2.2.2.

El experimento recién comienza a recolectar la información relevante del sistema a partir del bloque número 1000. Se utilizan los últimos diez bloques para que el sistema se **estabilice**, y que las mediciones sucedan ya con el sistema completamente en funcionamiento y sean todas bajo las mismas condiciones (todos los clientes funcionando, conectados, realizando transacciones y generando bloques).

2.2.2.2 Generación de transacciones

Cada cliente comienza a generar transacciones con los fondos que dispone. Para la nueva transacción, utiliza como *input* alguna de las que tiene disponible (en un comienzo las del fondeo del sistema). El destinatario de la nueva transacción, es alguna de las dos billeteras propias que cada cliente posee. Por motivos del protocolo, en cada transacción se utiliza

un *fee* mínimo de 0,000002 BTC que se resta al importe del único *input* utilizado.

De esta manera el valor original de la transacción de fondeo se va decrementando cada vez que se lo utiliza para generar una nueva transacción en un valor de solo 0,000002 BTC.

El dinero no desaparece en la red de Bitcoin. Todos los *fee* de las transacciones, van a ser asignados a aquel que ingrese dicha transacción en un bloque y pueda agregarlo a la *blockchain* minando dicho bloque.

Entonces, las transacciones que generan los clientes constan de un único *input* y un único *output*. El *input* es alguna transacción anterior destinada a dicho cliente, a la cual le descuenta 0,000002 BTC del importe. Y como destinatario elige la dirección de alguna de sus dos billeteras disponibles, que las asocia al *output*.

Al sistema de transacciones se le pueden modificar varios parámetros. La idea es generar transacciones cada cierto intervalo de tiempo, que se busca que siga una distribución $\mathrm{U}(X, Y)$, con esperanza Z. La cantidad de transacciones que se generan por intervalo de tiempo sigue otra distribución $\mathrm{U}(i, j)$ con esperanza p.

Por lo tanto, se espera que cada cliente genere p transacciones en un intervalo Z de tiempo.

Con estas variables, tratamos de imitar el comportamiento que podemos ver en la figura 2.2, la cual obtuvimos de un sitio de internet[(2)] que realiza varias estadísticas del sistema en vivo.

Por ejemplo, para un sistema con 200 clientes, con un intervalo de generación de transacciones que sigue una distribución $\mathrm{U}(120, 192)$ segundos con su respectiva esperanza de 156 s, generando transacciones con una distribución $\mathrm{U}(0, 5)$ con esperanza 2,5, esperamos que genere alrededor de 1923 transacciones en un intervalo de 600 s. De tal manera obtendríamos un promedio cercano a los 1900 transacciones por bloque.

Siguiendo con el ejemplo de un sistema con 200 clientes, cada uno tiene 890 transacciones con fondos asociados a cada una de sus billeteras. Por lo tanto, inicialmente dispone de

[(2)] blockchain.com

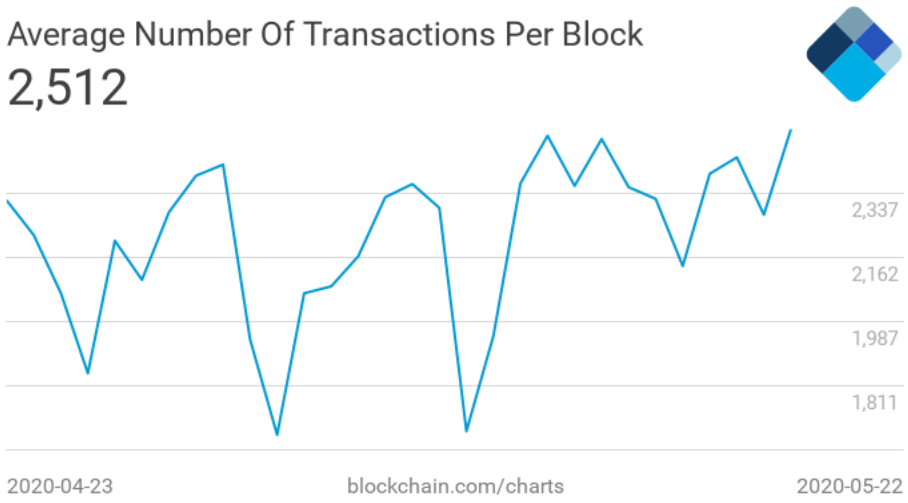


Figura 2.2: Cantidad de transacciones en un intervalo de 30 días, tomado de la página https://www.blockchain.com

1780 $(890 * 2)$ *inputs* para realizar nuevas transacciones. Se espera que una transacción asentada en un bloque en la *blockchain* tenga al menos seis bloques por encima de ella para poder ser utilizada como saldo en una nueva transacción. Para que un cliente no disponga de ninguna transacción para utilizar de *input*, deberían pasar más de 30 horas sin que se generen cinco bloques en el sistema. Tomando como base la transacción de fondeo, con un saldo de 0,125 BTC, puede ser utilizada mas de 6200 veces antes de que su saldo sea menor al *fee* de 0,000002 BTC.

Se realizarán experimentos recolectando información de menos de 1000 bloques, por motivos de tiempos. Si recolectáramos información de 1000 bloques, en un experimento con un *target* de 600 s, el mismo tardaría más de siete días en finalizar. De esta manera nos aseguramos que disponemos de una cantidad de transacciones más que suficientes para el continuo funcionamiento del sistema.

2.3 Modelo de red

En esta sección vamos a detallar qué modelos de red consideramos para realizar los experimentos. Comenzamos utilizando algunos escenarios para validar que la emulación del sistema representa fielmente lo que pasaría en el sistema real. Pasaremos cada vez a escenarios más complejos, en los cuales no resulta posible determinar si hay errores en nuestro

sistema o si su comportamiento es el del sistema en producción. En función de que pudimos verificar en modelos controlados que nuestro sistema funcionó bien, supondremos que en los modelos más complejos se estaría comportando como en la realidad.

Surgieron varios problemas al tener que reducir un sistema de 10 000 clientes a uno de algunas centenas de clientes, los cuales vamos a discutir en las siguientes secciones. Entre ellos, podemos mencionar los *delays* correspondientes a las comunicaciones dentro del sistema Bitcoin, cómo modelar la distribución del poder de cómputo entre todos los clientes que tenemos disponibles para intentar capturar el comportamiento del sistema real y cómo representar una red de retransmisión que nos sirva para entender al aporte de *FIBRE* al protocolo.

2.3.1 Topología física

Como Bitcoin es un sistema distribuido, los eventos producidos por cada nodo **no** se propagan de manera instantánea. El tiempo que transcurre entre que un nodo genera un nuevo evento (por ejemplo minar un bloque) y dicho evento llega al resto de los clientes de la red, se lo conoce como *tiempo de propagación*.

Desde el momento en el que un nodo r mina un bloque, hasta que dicho bloque le llega a otro nodo s (es decir, durante el tiempo de propagación de dicho bloque), todo el esfuerzo dedicado al minado por parte de s es desperdiciado, dado que el bloque que está buscando s ya fue descubierto por r. Además, si durante el tiempo de propagación, s descubriera un bloque de la misma altura que el que ya fue minado por r, se crearía un *fork* en la blockchain. Por lo tanto, es claro que en el sistema Bitcoin los tiempos de propagación afectan directamente a su funcionamiento.

La distribución geográfica de los clientes y la carga dinámica de la red y el sistema influyen en el funcionamiento del mismo. Como queremos analizar cómo afectan al sistema las redes de retransmisión, no nos es relevante modelar esto. Podemos suponer que un sistema con *delays* fijos es capaz de capturar los fenómenos que quisiéramos analizar en el mundo real.

Por lo tanto, decidimos utilizar una latencia de 100 ms entre todos los clientes de la red.

Luego cuando llegamos a los experimentos finales, nos topamos con el inconveniente de que su diámetro era relativamente pequeño y la conectividad entre los clientes era elevada. Esto produjo que los tiempos de propagación sean despreciables y no representaran la realidad del sistema en funcionamiento actualmente. Dado que la topología que utilizamos resulta ser un modelo estándar para modelar Internet, decidimos incrementar a 500 ms los tiempos de *delays* entre clientes. De esta manera conseguimos unos tiempos de aceptación más cercanos a los que maneja la red de Bitcoin.

La topología física que definimos mediante la herramienta `Sherlock Fog`, la mantenemos constante a lo largo de todos los experimentos. Tanto aquellos que son exploratorios para entender el buen funcionamiento del sistema, como los finales sobre los cuales realizamos las mediciones pertinentes. La idea fue conseguir una topología física, que nos permita conectar cualquier par de clientes de Bitcoin, en un tiempo establecido. Luego, las conexiones por medio del cliente de Bitcoin, van a conseguir la topología lógica de conexión entre clientes de Bitcoin que pretendamos.

`Sherlock Fog` nos permite definir *hosts* y latencia entre ellos. Para distribuir la carga del sistema entre los cuatro servidores utilizados para los experimentos, repartimos equitativamente la cantidad de clientes entre ellos como se muestra en la figura 2.3. Armamos una clique $K(4)$, en el que cada *host* (b) va a estar corriendo en uno de los servidores. Los enlaces de esta clique (d) no van a tener ninguna latencia, por lo cual, las comunicaciones que utilicen estos caminos no van a tener ningún *delay* adicional.

Luego, se les va a conectar un host (a) asociado a cada uno de los clientes de Bitcoin que va a correr en dicho servidor, con un enlace (c) de una latencia de la mitad del target establecido. De esta manera, la comunicación entre cualquier par de clientes de nuestro sistema va a seguir un camino que utilice dos enlaces del tipo (c) y uno del tipo (d). Consiguiendo tener la latencia establecida sin importar si se encuentran dentro del mismo servidor o no.

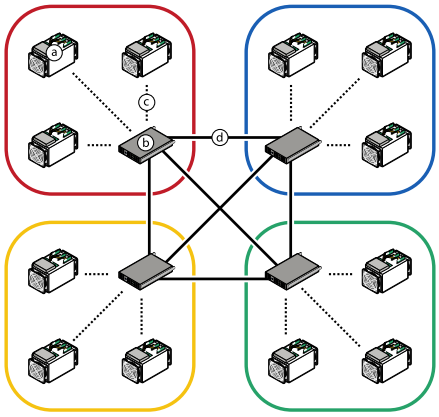


Figura 2.3: Modelo de red utilizando 4 servidores para correr los distintos clientes del experimento. Los host **b** son exclusivamente utilizados para interconectar los distintos servidores. La clique formada por los enlaces **d** tienen un delay de 0 ms por lo tanto no tiene costo pasar por los mismo. Los host **a** son los encargados de correr cada cliente BTC que se va a utilizar en el experimento y mediante los links **c** están configurados con la mitad del tiempo que queremos de delay entre los clientes BTC. De esta forma para llegar desde cualquier cliente BTC a cualquier otro que se encuentre tanto en el mismo servido como en algún otro, existe el delay establecido.

Como ejemplo, en los experimentos finales, los enlaces del tipo (c) tienen una latencia de 250 ms y por lo tanto la latencia final entre cualquier par de clientes de la red podría ser de 500 ms. Luego, con un overlay establecido, solo los clientes que sean adyacentes en esta topología lógica, van a comunicarse en 500 ms. Más adelante, cuando hablamos sobre la automatización de los experimentos, detallamos el funcionamiento de `Sherlock Fog`

2.3.2 Distribución del Hashing Power

Como vimos en la sección 1.1.2, en Bitcoin se utiliza un sistema de consenso para decidir qué bloques pertenecen a la blockchain y cuáles no. Dicho consenso se logra a través del proceso de minado, en el cual los nodos usan poder de cómputo para decidir cuál va a ser el siguiente bloque de la blockchain.

Cuando un minero intenta agregar un nuevo bloque a la *blockchain*, comienza a probar reiteradamente distintos *hashes* hasta que da con uno que cumpla la dificultad. La cantidad de *hashes* que prueba por unidad de tiempo es denominado *hash rate*.

El *Hashing Power* de la red es la cantidad de *hashes* por unidad de tiempo que están generando todos los mineros de la red en determinado momento. Esto varía producto del dinamismo de la red (clientes activos), como se puede ver tanto en la figura 2.4 como en la

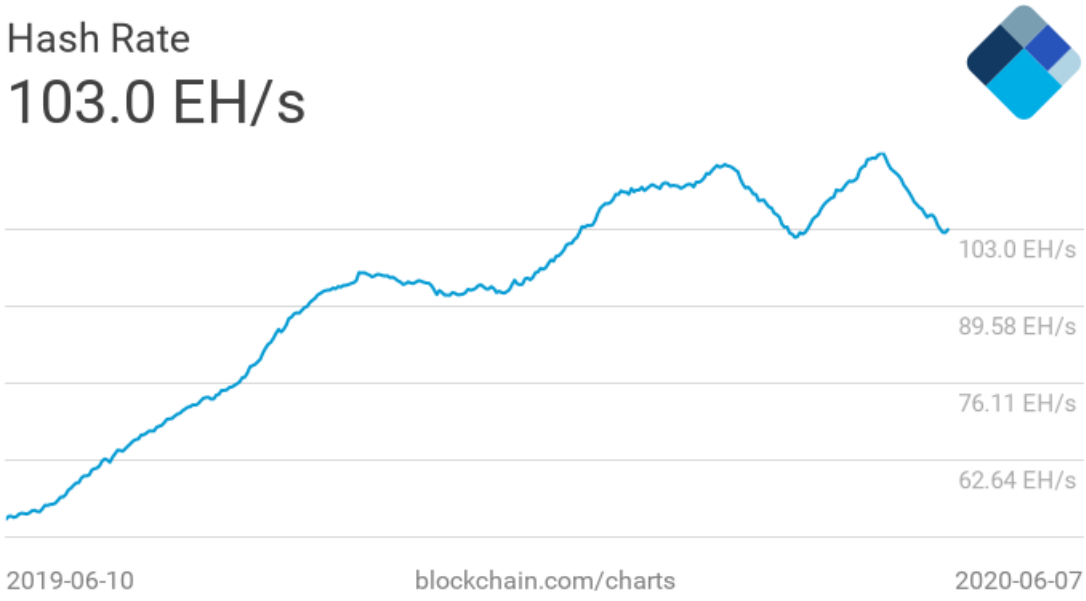


Figura 2.4: Variación del Hash Rate en el último año, tomado de la página https://www.blockchain.com

página https://www.blockchain.com/charts/hash-rate que recolecta varias estadísticas.

Menos del 1 % de los nodos de la red controla más del 86 % del *hash rate* de toda la red. Estos nodos con gran poder de cómputo son llamados *pooles de minería*, ya que varios de ellos consisten en la unión de muchos clientes mineros que no están conectados de manera directa a la red. Dichos clientes distribuyen el espacio de búsqueda de la *proof of work*. De esa manera, como además introducen bloques a la red desde un mismo conjunto reducido de IPs, para los demás nodos de la red, este conjunto de clientes es visto como un único nodo que tiene una gran parte del *hashing power* de la red.

Podemos ver tanto en la figura 2.5 como en la página https://btc.com/stats/pool?pool_mode=month, la distribución del *hasing power*. Se estima el *hashing power* de cada *pool* fijándose qué porcentaje de los bloques fue minado por cada uno durante cierto período de tiempo.

El porcentaje de *hashing power* de la red que es controlado por *pools* de minería varía en el tiempo. En algunos casos llegó a ser más del 95 % del total.

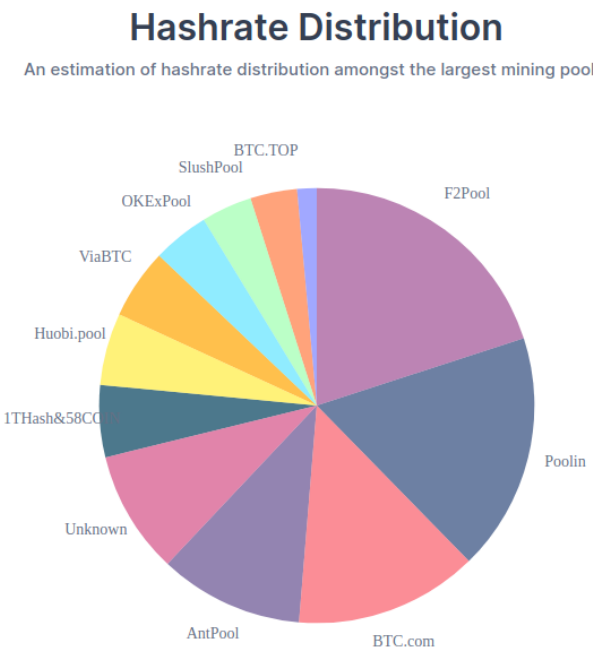


Figura 2.5: Estimación del Hashing Power, tomado de la página https://www.blockchain.com

En nuestro modelo de la red de *Bitcoin* un grupo reducido de clientes que representará a los pooles de minería, concentrarán el 86 % del *hasing power* de la red. Mientras que el 14 % restante va a estar distribuido en otro grupo de clientes con mayor cantidad de participantes.

(a) **Pooles de minería FIBRE**:

Son un grupo reducido de clientes que van a concentrar el 43 % del hashing power de la red y eventualmente van a estar comunicados dentro de la red de FIBRE

(b) **Pooles de minería Normales**:

Este grupo va a concentrar el otro 43 % del *Hashing Power*, podremos utilizarlo para realizar comparaciones contra aquellos que participen en la red de FIBRE.

(c) **Mineros Comunes**:

El 14 % restante del *Hashing Power*, va a estar distribuido entre una gran cantidad de clientes que vienen a representar a aquellos clientes que tienen una participación despreciable con respecto a la generación de nuevos bloques, pero son fundamentales a la hora de transmitir transacciones, bloques y generar consenso dentro de la red.

Vamos a poder sacar conclusiones no solo del funcionamiento general del sistema, si no que también, vamos poder diferenciar entre los distintos grupos que intervienen dentro de nuestro modelo.

2.3.3 Red de retransmisión

En la sección 1.2 pudimos observar cómo las redes de retransmisión están pensadas para conectar lugares distantes y así reducir el *delay* de clientes de Bitcoin que utilizarán la red *P2P* del protocolo. La red pública de *FIBRE*, está constituida por seis servidores distribuidos a lo largo de todo el hemisferio norte.

En nuestros experimentos, al contar con una cantidad de clientes dos órdenes de magnitud menor a los de la red real y no modelando la distribución geográfica o alguna variabilidad en los *delays* de los clientes, tuvimos que simplificar la red de retransmisión a su mínima expresión. Decidimos utilizar un solo servidor, al cual se van a conectar aquellos clientes que pertenezcan a la red de FIBRE.

Este cliente que actúa como ***HUB*** y enmascara el funcionamiento interno de la red de retransmisión, va a estar configurado para solo transmitir bloques e ignorar las transacciones que circulan por el sistema de Bitcoin. Por lo tanto, los clientes que estén conectados a el tienen una alternativa a la red tradicional *P2P* para compartir sus bloques.

2.4 Medidas realizadas

Vamos a plantear distintas medidas con el objetivo tanto de poder validar los experimentos, como recolectar información relevante para intentar responder las dudas planteadas en esta tesis.

Primero presentamos magnitudes simples relacionadas a dos fenómenos observables: La cantidad de *forks* en la blockchain y los tiempos de propagación de bloques dentro la red. Dos fenómenos básicos en el sistema del cual se puede inferir algún tipo de degradación en la red.

Discutimos cómo replicar las mediciones que se realizaron en la tesis de Nicolás DeCarli [DC19], que trabaja sobre el concepto de ***Hashing Power* Desperdiciado**. Ésta analiza un poco más en profundidad la rentabilidad del sistema y no solo verifica su degradación si no que lo pondera en función de la participación en el mismo. Identificando el *Hashing Power* desperdiciado por cada grupo de clientes que utilizamos en los experimentos, vamos a poder verificar si alguna parte de la red se ve más beneficiada que otra.

También, vamos a analizar cómo impacta la centralidad de distintos nodos en la red. Midiendo cuánto tardan en llegar los bloques a los distintos grupos de clientes dentro de la red y contando la cantidad de bloques minados por cada grupo que pertenecen o no a la cadena principal, vamos a poder definir y analizar el comportamiento y eficiencia de cada tipo de actor en el sistema.

Vamos a realizar el seguimiento de algunas magnitudes de control del experimento con la cuales verificamos que éste esté siguiendo los parámetros establecidos. Y, por último, tendremos algunas mediciones sobre la infraestructura del sistema, para validar que no haya problemas externos a la emulación que puedan afectar los resultados de los experimentos.

2.4.1 Forks

Reconstruyendo la blockchain resultante, podemos identificar todos los bloques *stale* que generaron algún *fork* dentro de la red. Podemos ver la cantidad de *forks* que hubo en la blockchain y además para cada uno identificamos su altura, es decir, su distancia a la *mainchain*. Vamos a poder identificar la cantidad de *forks* por cada tipo de nodo.

2.4.2 Tiempos de propagación

Cada vez que aparece un bloque nuevo en la blockchain, se anuncia mediante la distribución del hash de su *header*. Luego, cuando cada minero tiene toda la información correspondiente al bloque, lo acepta, es decir, queda impactado en su copia local de la blockchain. En los experimentos, para cada bloque de la blockchain, calculamos el tiempo de propagación de

los dos eventos. Es decir, calculamos cuánto tardó en llegar el hash de su *header* a cada minero, y cuánto tardó cada minero en aceptar el bloque.

2.4.3 Wasted Hashing Power

Si bien las medidas anteriores dan una noción de la calidad de la red, lo hacen desde un punto de vista puramente técnico. Si aumenta la cantidad de *forks* o los tiempos de propagación, se degrada la convergencia de la red, es decir, aumenta el esfuerzo requerido para minar una determinada cantidad de bloques. Sin embargo, no es trivial cuantificar el impacto económico que producen las variaciones en los aspectos técnicos de la red. A partir de las magnitudes presentadas anteriormente, presentamos *wasted hashing power* que captura el porcentaje de *hashing power* que no fue utilizado para la convergencia de la red. Utilizando esta medida, es posible determinar qué proporción de una inversión económica es realmente usada.

En la sección 1.1.1.2 detallamos los atributos de los bloques en Bitcoin. Algunos de estos atributos, como la dificultad de la red, necesitan ser congruentes con la *mainchain*. Es decir, si por ejemplo un minero introduce a la red un nuevo bloque, que tiene como dificultad un valor distinto al determinado por la red, los demás nodos van a rechazar dicho bloque.

Los mineros utilizan el último bloque conocido para determinar cuál es la dificultad de la red. Si el cliente efectivamente tiene el extremo (último bloque) de la *mainchain*, también conoce la dificultad de la red actual, y equivocarse al establecerla en el bloque sería un error evitable. Si, en cambio, el cliente no tiene el extremo de la *mainchain* correcto, el escenario es más complejo. Aunque un cliente actúe de manera correcta y escriba como ancestro al extremo de su copia local de la *mainchain*, como Bitcoin es un sistema distribuido, puede suceder que el cliente desconozca que ya haya aparecido un bloque en la red con ese mismo ancestro. En este caso, mientras al cliente no le hayan llegado todos los datos necesarios para poder asentar el bloque en su blockchain, lo mejor que puede hacer es seguir calculando *hashes* con el extremo de la blockchain que conoce. En esa situación, el campo ancestro se establece de manera errónea, pero el cliente no puede evitar equivocarse. Lo que queremos

remarcar es que puede haber errores inevitables en el proceso de minado producidos por la naturaleza distribuida del sistema Bitcoin.

Como explicamos en la sección 1.1.2, en el proceso de minado cada nodo aporta su *hashing power* para que, en conjunto, minen en promedio un nuevo bloque cada diez minutos. Sin embargo, notemos que la probabilidad de que un *hash* cumpla con la dificultad de la red es independiente de la validez de los demás parámetros. Por lo tanto, si un minero logra calcular un *hash* que cumpla con la dificultad requerida, y luego el bloque termina siendo inválido por otros motivos, la red va a tener que esperar en promedio diez minutos para generar un nuevo bloque. En este caso, el cliente que generó el bloque inválido desperdició su *hashing power* mientras minaba para dicho bloque, ya que no lo estaba usando para contribuir al objetivo grupal de generar un bloque válido cada diez minutos. Dicho de otra manera, vamos a considerar que un cliente está desperdiciando su *hashing power* cuando genera *hashes* de un bloque que tiene algún campo inválido.

En el presente trabajo suponemos que todos los nodos actúan de manera óptima, es decir, no vamos a tener en cuenta errores evitables. Todos los campos que tienen que ser consistentes con la red dependen del último bloque de la *mainchain*. Por lo tanto, si generamos un *hash* válido para un bloque nuevo que tiene el campo ancestro correcto, entonces el bloque va a ser válido. Lo que sucede a veces es que la correctitud del campo ancestro puede ser determinada luego de que transcurre bastante tiempo desde la generación del bloque. Por ejemplo, cuando ocurre un *fork* en la red, recién luego de que se determine cuál de los bloques con misma altura termine formando parte de la *mainchain*, cada nodo va a saber si estaba ayudando a minar sobre la *mainchain* o no.

En la práctica, cada cliente desperdicia su *hashing power* en dos situaciones: (i) durante el tiempo que transcurre entre que se genera un bloque nuevo y el cliente lo acepta; y (ii) cuando se produce un *fork* en la blockchain y el cliente mina sobre un bloque que no termina en la *mainchain*. La sumatoria de todos esos tiempos para un cliente dado es el tiempo en el que dicho cliente desperdició su *hashing power*. Llamemos d_i al tiempo que el nodo i pasó desperdiciando su hashing power, y $h_i \in [0, 1]$ a su proporción de *hashing*

power con respecto al total de la red. Entonces, definimos al tiempo total desperdiciado de un experimento x como $D_x = \sum_{i \in nodos} d_i \cdot h_i$. Luego, dado que el tiempo total en el que el experimento x transcurrió fue T_x, el porcentaje de *wasted hashing power* de x se puede calcular como $(D_x/T_x) \cdot 100$.

La figura 2.6 ilustra una fracción de la línea de tiempo relacionada al proceso minado de la blockchain. La misma incluye los eventos relacionados a cuatro clientes de la red, y remarca los momentos en los que dichos clientes desperdiciaron su *hashing power*.

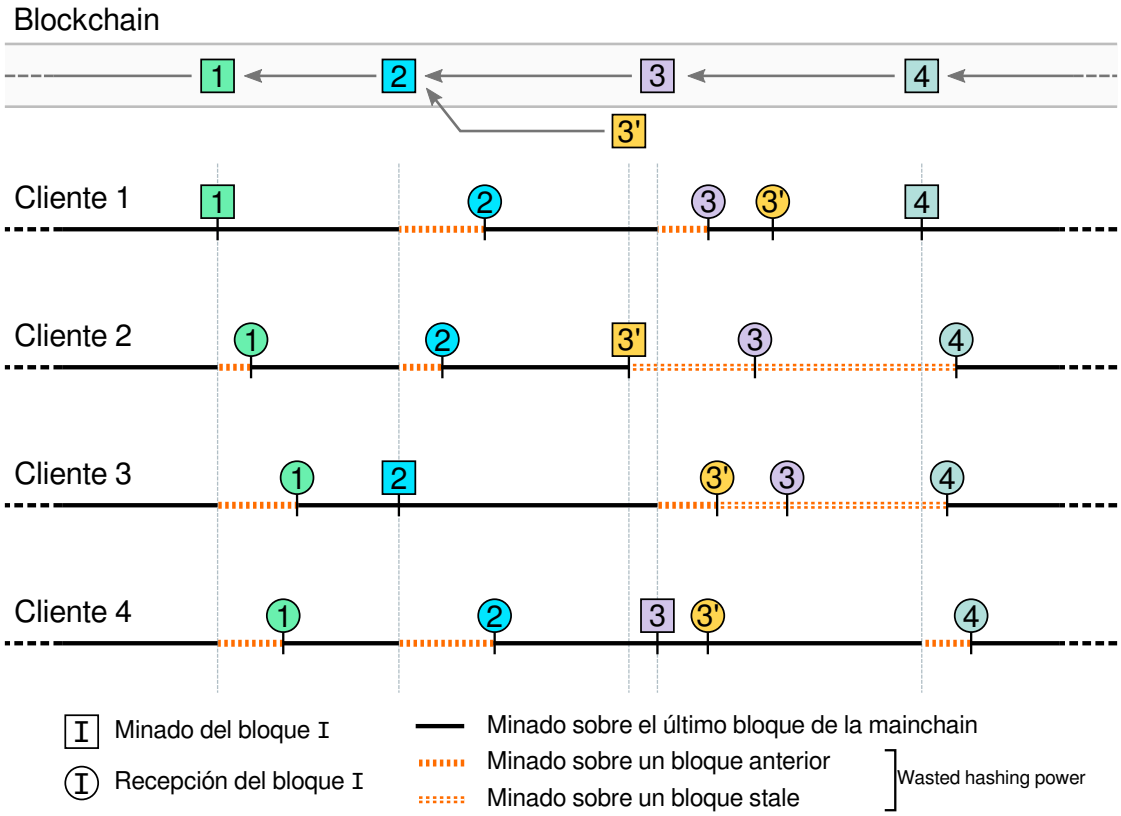


Figura 2.6: Definición del HashingPower desperdiciado. Podemos observa como cuatro clientes minando en simultáneo sobre la misma blockchain, tienen momentos en los que su poder de cómputo no está siendo utilizado para extender la cadena más larga.

En la sección de resultados vamos a utilizar como métrica el porcentaje de *wasted hashing power* de un experimento. De esta manera vamos a medir el impacto monetario que resulta de introducir la red de FIBRE en el sistema.

2.4.4 Centralidad

La naturaleza distribuida del protocolo Bitcoin hace que funcione como un sistema completamente descentralizado, en el cual no hay ningún actor que pueda tener control de las transacciones o los bloques ajenos.

Esto no es del todo cierto. Dado que si alguien es capaz de controlar un gran porcentaje del poder de cómputo de la red, sería factible realizar un **ataque del** 51 %. Este ataque consiste en que alguien que dispone de más del 50 % del poder de cómputo de la red, puede deliberadamente utilizar ese poder de cómputo en su favor. En un escenario de *Double Spending* en donde existen dos transacciones que utilizan los mismos fondos, una legítima transfiriendo esos fondos a otro interlocutor y otra apócrifa en la cual se asignan los fondos a una billetera propia. El atacante puede decidir volcar todo su poder de cómputo a minar aquel bloque que tenga la transacción apócrifa y así conseguir revertir la transacción original.

En nuestros experimentos vamos a suponer que todos los participantes son legítimos y no están confabulando para intentar hacer algún tipo de trampa en el sistema. Pero queremos analizar si es factible que al ser utilizada la red de retransmisión, exista algún fenómeno que eventualmente pueda ser explotado por actores maliciosos.

De esta manera, si encontramos algunas situaciones en las cuales ciertos participantes se vean beneficiados en detrimento del resto de la red, podemos considerar que el protocolo tiende a centralizarse alrededor de aquellos que puedan usufructuar dichos beneficios.

2.4.4.1 Bloques dentro de la *mainchain*

Identificando los tres grupos mencionados anteriormente, vamos a hacer un seguimiento tanto de aquellos bloques que terminan perteneciendo a la cadena principal, como aquellos que quedaron *stale* productos de algún *fork*.

2.4.4.2 Convergencia

En una red estática como la que utilizaremos en los experimentos, tanto el tiempo de descubrimiento como el tiempo de aceptación promedio para cada cliente dentro de la red, tienden a estabilizarse alrededor de algún punto. Por lo tanto, calculando estos datos con la red de retransmisión activada y desactivada podemos ver si la misma tiene algún tipo de efecto en estas variables.

Concentrado estos datos por grupos de clientes, podremos caracterizar dichos grupos en función de la presencia de la red de retransmisión y ver si esta afecta equitativamente a todos ellos.

2.4.5 Verificación del experimento

En cada experimento, establecemos valores a dos variables: el tiempo entre bloques objetivo y la cantidad de transacciones que son introducidas al sistema por segundo. Con el fin de controlar que el sistema se haya comportado acorde a los valores deseados, controlamos que ciertas mediciones obtenidas estén en un rango esperable.

Por un lado, calculamos el tiempo entre bloques promedio observado, teniendo en cuenta también los bloques que no forman parte de la *mainchain*. De esta manera, controlamos que el proceso de minado simulado se comporte como esperamos.

Por otro lado, utilizamos la cantidad de transacciones en cada bloque para determinar cuántas, en promedio, tuvieron los bloques que quedaron en la *mainchain* del experimento. Este valor, junto al promedio de tiempo entre bloques de la *mainchain*, nos indica la cantidad de transacciones por segundo promedio que hubo durante el experimento.

2.4.6 Limitaciones de Hardware

La red de *Bitcoin* es un sistema dinámico, que modifica su estado en función del tiempo y la carga que esté soportando. Para los experimentos contamos con hasta 8 servidores de 64 cores cada uno conectados por una red de alta velocidad como se detalla en la sección 2.7.

Si bien tiene un poder de cómputo considerable, si se superara la carga que admiten, podrían afectar negativamente el desarrollo de la emulación generando resultados que no representarían el funcionamiento del sistema real.

Utilizamos tres herramientas a fin de determinar que el sistema se encuentra en un rango de funcionamiento aceptable. Lo primero que medimos es la carga de los procesadores. Con el *load average* controlamos que en ningún momento haya algún proceso a la espera de un procesador para ser ejecutado, lo que produciría *delays*. Luego desarrollamos una aplicación que mide el *bandwidth* (ancho de banda) usado de la placa de red del sistema. Una instancia de este proceso es iniciada en cada host físico.

Por otro lado, en cada host virtual iniciamos un *engine* que cada cierto período de tiempo, le envía un paquete de 18 bytes a otro host virtual. El tiempo en que permanece dormido cada vez es aleatorio, siguiendo una distribución uniforme con media 1,5 s y varianza 0,5 s. La selección del host destino también es aleatoria y uniforme entre todos los posibles destinatarios.

Con la primer herramienta controlamos que los recursos del sistema no se vean sobrepasados por el funcionamiento tanto de los clientes de Bitcoin como la herramienta de simulación que corren simulánemamente. Luego también, podemos ver que no se haya saturado el ancho de banda de la red, producto de todas las comunicaciones que se realizan entre los distintos actores de los experimentos. Y por último, podemos medir la varianza de la latencia de la red a través del tiempo para ver que se mantenga dentro de los valores esperados.

Finalmente, cabe aclarar que el *overhead* que introduce la utilización de estas herramientas no afecta los resultados de los experimentos ya que el consumo de CPU por estas herramientas es despreciable. Con respecto a la red, ésta es utilizada solamente por el *engine* de ping, el cual en promedio agrega 3,2 kB/s al tráfico de la red. Como los equipos donde realizamos los experimentos tienen enlaces de red de 1 Gbit/s, 3,2 kB/s de datos no resulta en un *overhead* perceptible.

2.5 Diseño de los experimenos

Los experimentos de este trabajo consisten en una red de Bitcoin de 200 clientes, los cuales están minando y enviando transacciones nuevas a sus *peers* en todo momento. Luego de que la *mainchain* alcanza 500 bloques de altura, detenemos los clientes y guardamos todos los datos estadísticos generados. La red de Bitcoin utilizada sigue el modelo propuesto en la sección 2.2, y los nodos utilizan el cliente descripto en la sección 2.1. En cada experimento vamos a establecer dos variables: (i) el tiempo entre bloques esperado (*target time* de la red), y (ii) la cantidad de transacciones que hay en promedio en cada bloque.

Se realizan experimentos con la misma topología lógica de la red de *bitcoin*, para distintas configuraciones de su *target* y modificando la cantidad de clientes dentro de la red de retransmisión *FIBRE*. Las mediciones de las variables de interés se realizan activando y desactivando la red *FIBRE* para poder verificar cómo se comporta el sistema con el agregado de esta red.

La red de *bitcoin* tiene un *target time* definido de 600 s. La comunidad *bitcoin* presenció profundas discusiones sobre este tópico y si alguna modificación podría proporcionar beneficios para el sistema. La tesis de Nicolás DeCarli [DC19] propone modificar el *target time* para intentar aumentar el *throughput* del sistema, llegando a la conclusión de que reducirlo trae ciertos beneficios. Pero estos análisis se realizaron con un cliente de *bitcoin* modificado para poder aumentar sustancialmente la cantidad de transacciones que puede validar el sistema.

Nosotros vamos a tomar tres escenarios posibles, variando el *target time* entre 600 s, 300 s y 150 s. Queremos verificar cómo se comporta el sistema bajo las distintas configuraciones y analizar si la reducción del mismo no trae aparejados inconvenientes o, inclusive aún, algún tipo de beneficio.

Un parámetro que queda fijo a lo largo de todos los experimentos es la cantidad total de transacciones que se agregan al sistema. Como se detalló en la sección 2.2.2.2, se intenta conseguir tener la misma cantidad de transacciones que se generan en la red de *bitcoin*

actualmente. Por lo tanto, si se puede reducir el *target time* sin que se vea perjudicada la calidad del sistema, se conseguirían mejorar los tiempos de confirmación de las transacciones.

2.6 Automatización de los experimentos

Para la puesta en marcha de un experimento, es necesario llevar a cabo varios procesos que generen el escenario requerido. Primero hay que crear los hosts virtuales y agregar enlaces de red entre ellos para construir la topología física de red deseada, la cual es la misma para todos los experimentos. Luego tenemos que inicializar todos los clientes de Bitcoin que forman parte del experimento, conectándolos acorde al *overlay* pretendido y estableciendo así la topología lógica que va a utilizar dicho experimento para transmitir la información por el sistema (tanto las transacciones como los bloques y toda aquella información que necesita para funcionar). Finalmente, debemos instanciar un cliente extra que minará 990 bloques y repartirá la recompensa entre todos los clientes del sistema para que posean fondos para realizar las transacciones necesarias. Cuando todo está listo, el sistema comienza a trabajar con los clientes generando bloques y transacciones para conseguir los objetivos globales. Las mediciones se comienzan a realizar a partir del bloque número 1000, ya con todo el sistema en marcha.

En Geier et al. [GM18] se introduce el lenguaje `fog`, que es utilizado para describir una topología de red IP y definir comandos a ejecutar en los hosts de dicha red. En el mismo trabajo se presenta `SherlockFog`, una aplicación que toma un *script* escrito en `fog` y ejecuta comandos de *shell* que logran una emulación de la topología deseada y ejecutan en los hosts virtuales lo especificado en el *script*. Cada nodo de la red generada es emulado mediante un *container* de Linux que es un ambiente aislado adecuado para realizar experimentos. Utilizamos `SherlockFog` para emular la red IP y correr, de manera automatizada, los módulos necesarios para inicializar los experimentos.

En la figura 2.7 podemos ver una esquematización de cómo funciona `SherlockFog`. Podemos diferenciar los siguientes pasos:

1. El usuario elige una aplicación y una topología y construye un *script* que define un el experimento en SherlockFog. Dicho *script* utilizará un conjunto de nodos físicos especificados por el usuario.

2. SherlockFog es ejecutado en el coordinador, el cual se conectará a cada nodo para inicializar la red virtual.

3. Se generan enlaces virtuales correspondientes a la topología descripta en el *script* de entrada. Se utiliza *ruteo* estático para permitir la comunicación entre cada interfaz de red virtual.

4. El código de cada aplicación es ejecutado en los nodos virtuales según lo que haya especificado el usuario en el *script* de entrada.

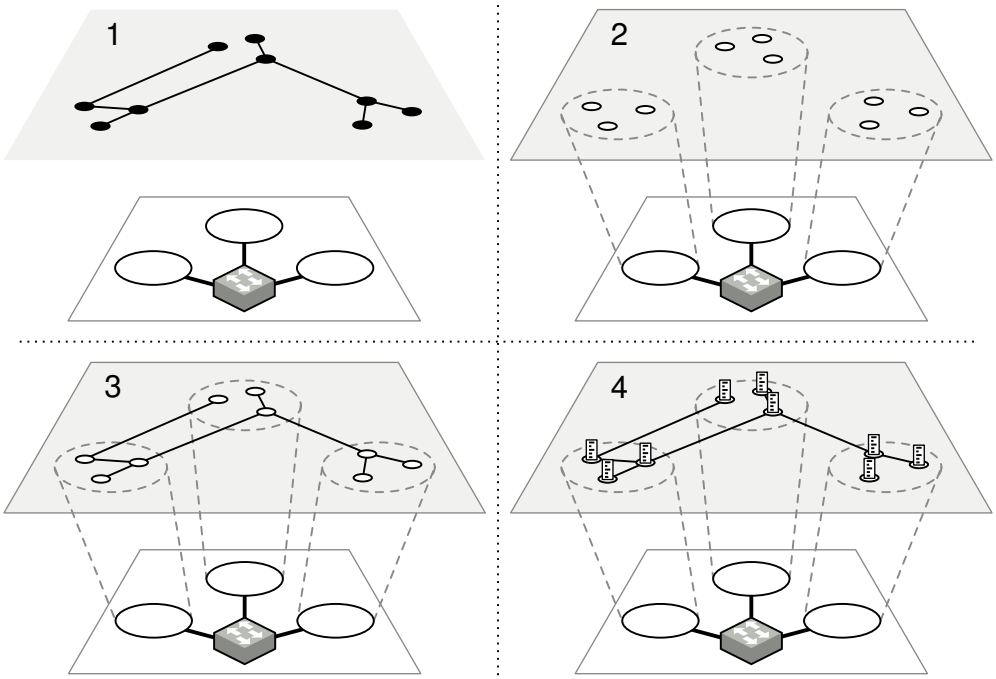


Figura 2.7: 1) La definición en lenguaje `fog` del experimento a realizar. 2) La creación de los contenedores necesarios. 3) La emulación de la red lógica pretendida. 4) La ejecución de distintos *scripts* en cada nodo de la red. `SherlockFog` permite hacer interactuar varios clientes del protocolo BTC (ya sea en el mismo host físico como en distintos), teniendo control de la red que los comunica.

SherlockFog permite modificar los parámetros en tiempo de ejecución, y repetir los experimentos con distintos parámetros o topologías. La salida puede ser recolectada para su posterior análisis.

Por las siguientes consideraciones se decidió utilizar SherlockFog para llevar adelante los experimentos de esta tesis.

- SherlockFog puede ser ejecutado en hardware convencional.
- Los *scripts* son escritos en un lenguaje propio denominado *fog*, que permite definir la topología de red, parámetros de los experimentos y comandos para ejecutar la o las aplicaciones.
- La herramienta puede ser utilizada sobre un único nodo físico o sobre varios, permitiendo escalar la red emulada sobre otros recursos.
- La red, CPU y la memoria pueden ser compartimentadas entre nodos virtuales que se instancian sobre un mismo sistema utilizando técnicas de virtualización liviana.
- El código de usuario se puede ejecutar sin ningún tipo de modificación, permitiendo la evaluación de programas tanto de código abierto como cerrado.
- La dinámica del sistema de comunicaciones puede ser modelada cambiando el ancho de banda o la pérdida de paquetes de un enlace en tiempo de ejecución.

Construimos un programa que toma un JSON que representa una instancia del modelo descripto en la sección 2.3, luego crea un script con formato *fog*, y ejecuta `Sherlock Fog` usándolo como entrada. El *script* está diseñado para que `SherlockFog` inicialice un experimento con el escenario solicitado.

Teniendo la capacidad de utilizar varios hosts físicos para emular la red, en nuestro caso utilizaremos los cuatro servidores descriptos en la sección 2.7. Tomando la configuración del archivo en formato *fog*, se levantan distintos *containers* en varios *hosts* físicos y se establece la topología física con las latencias que pretendemos. Luego en cada uno de los hosts, se va a poder ejecutar distintos *scripts* ejecutando los clientes de bitcoin, que puedan usufructuar la red emulada.

En los párrafos subsiguientes detallamos el proceso que automatiza los experimentos, los programas y módulos mencionados son lanzados por `SherlockFog` en los hosts emulados.

En primer lugar, se inicializan todos los clientes de Bitcoin que van a ser parte del experimento y cada uno agrega como *peer* a los nodos requeridos para generar la topología deseada. Una vez construido el escenario de la red Bitcoin, una instancia más del cliente es iniciada y se conecta a todos los nodos de la red del experimento para minar 990 bloques y repartir sus recompensas.

Cuando todos los clientes del sistema tienen la Blockchain inicial impactada en su base local, el cliente adicional se desconecta del sistema. Si el experimento cuenta con la red de retransmisión activada, se utiliza este cliente adicional para modelar a la red de `FIBRE` descripta en la sección 2.3.3. Aquellos clientes que forman parte de dicha red agregan a este cliente a su lista de *peers*.

De esta forma queda constituida toda la infraestructura necesaria para el desarrollo del experimento.

Luego, en cada host que se encuentra corriendo un cliente del protocolo de Bitcoin, se inicia un programa para llevar adelante la emulación del sistema. Éste consiste en dos *threads* que interactúan con el cliente BTC por medio de llamados RPC, de los cuales uno es el encargado de generar bloques y el otro las transacciones del sistema.

En la figura 2.8 podemos ver retratada la interacción entre los *threads* del programa y el cliente de Bitcoin.

El fin del *thread* 1 es implementar la simulación del proceso de minado de bloques mencionada en la sección 2.2.1. Para lograr esto cuenta con un ciclo infinito que: (i) genera un número al azar que representa una cantidad de milisegundos y se duerme por ese tiempo, (ii) al despertarse le indica al cliente asignado que mine un bloque, el cual es generado inmediatamente dado que la dificultad se encuentra en 0. Para poder simular el comportamiento del escenario deseado, los números aleatorios generados por el *engine* del cliente siguen una distribución exponencial con parámetro $\lambda = h/t$. Siendo h la proporción de *hashing power* que tiene el cliente, y t el tiempo entre bloques esperado.

Por otro lado, para generar transacciones, el *thread* 2 del programa también ejecuta un

ciclo infinito que se duerme por una cantidad aleatoria de milisegundos e interactúa con el cliente para introducir transacciones al sistema como se detalla en la sección 2.2.2.2. Los números aleatorios generados en este procedimiento siguen una distribución uniforme. La media y varianza de la misma van a ser configuradas para que la cantidad de transacciones por bloque promedio del experimento se corresponda con los valores de la red actual de Bitcoin.

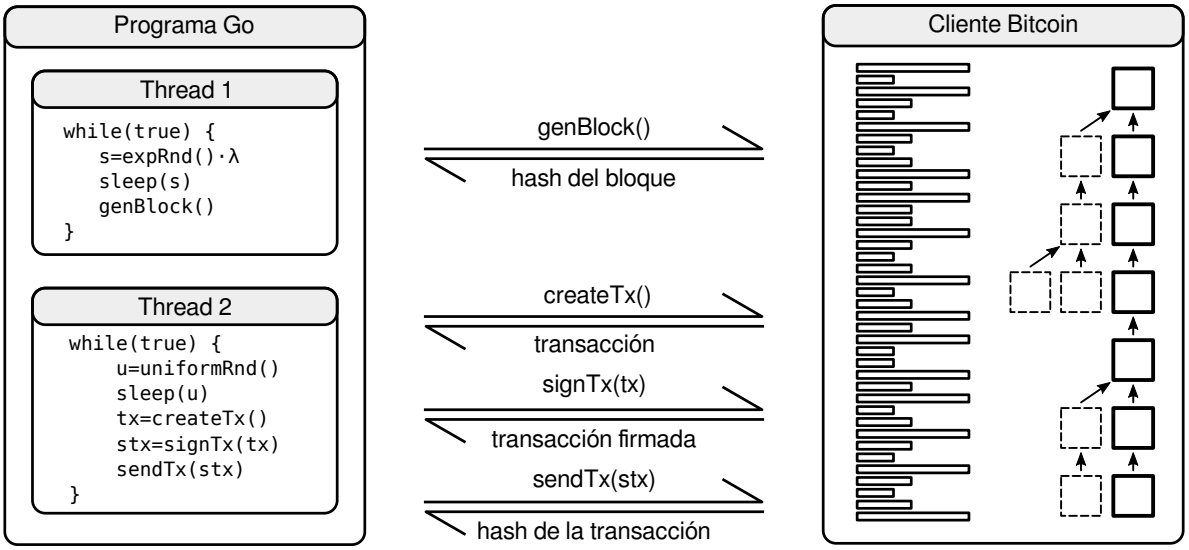


Figura 2.8: Diagrama de interacción de nuestro programa con el cliente de Bitcoin.

Aunque Bitcoin tiene un sistema de transacciones que permite realizar operaciones bastante complejas, decidimos solo utilizar transacciones entre dos participantes. En todos nuestros experimentos las transacciones tienen un solo *input* y un solo *output*.

Al inicializar cada cliente de Bitcoin, generamos dos direcciones de pago nuevas vinculadas a su billetera digital. Repartimos la recompensa de los primeros 890 bloques (esto ocurre porque se minaron 990 bloques, pero la recompensa de un bloque solo se puede utilizar luego de que se minen 100 bloques por encima del mismo) entre todas las billeteras del sistema, multiplicando por dos la cantidad de *inputs* que tiene disponible cada cliente para realizar nuevas transacciones. De esta manera, cada cliente puede realizar como mínimo hasta 1780 transacciones antes de agotar los *inputs* habilitados para realizar nuevas transacciones (sin que éstas sean impactadas en la *blockchain* y les permita realizar más transacciones). La

distribución de crédito inicial se hace de manera equitativa, resultando en que todos los clientes tengan la misma cantidad de dinero disponible. Luego, cada vez que se genera una transacción mediante un cliente determinado, se asigna el *output* a una de las direcciones del mismo. Además, como *input* de la transacción se utiliza un *output* de la otra dirección del cliente. En cada operación se usa la menor cantidad posible de *fee*, para hacer así que el crédito inicial alcance para todo el experimento.

Cuando las mediciones del experimento comienzan (bloque número 1000), cada cliente puede utilizar 1800 *inputs* (900 bloques $*$ 2 billeteras). A lo largo del experimento, se terminan de desbloquear los fondos de las recompensas de los últimos 90 bloques del proceso de fondeo, lo cual incrementa la cantidad de *inputs* para poder generar nuevas transacciones en 180. Para finalizar, desde el momento que se comienzan a minar bloques orgánicamente dentro del sistema, todos los *fees* utilizados en las transacciones y las nuevas recompensas, van a parar a aquel cliente capaz de minar un nuevo bloque, por lo tanto algunos clientes van a ver aumentado incluso algo más la cantidad de dinero e *inputs* disponibles para realizar transacciones. Todo este proceso se realizó para que todos los participantes del sistema puedan estar generando transacciones a lo largo de todo el experimento, incluso aquellos que tienen un muy bajo poder de cómputo, lo cual implicaría que la probabilidad de que agreguen bloques a la *mainchain* sea despreciable y pudieran llegar a quedarse sin créditos.

2.7 Infraestructura Utilizada

Todas las ejecuciones de los experimentos con la red de retransmisión activada y desactivada, fueron realizadas en paralelo utilizando cuatro servidores `Dell PowerEdge C6145`, cada uno con las siguientes características:

- CPU: Cuatro procesadores AMD Opteron 6276 de 16 núcleos cada uno.
- RAM: 128 GB DDR3.
- SO: `Debian GNU/Linux 7`.

- Kernel: `Linux 4.19`.
- Interfaz de red: Ethernet de 1 Gbit/s.

En total, se utilizaron 256 núcleos para emular redes de Bitcoin con 200 mineros, los cuales tenían a su disposición un poco más de 2,5 GB de RAM cada uno.

Como detallamos en la sección 2.6, usamos `SherlockFog` para emular la topología de red IP, que permite distribuir los hosts de dicha red a través de distintas máquinas físicas. En nuestros experimentos utilizamos cada servidor para emular exactamente 50 mineros, es decir, repartimos equitativamente un cuarto de la emulación a cada uno.

Capítulo 3

Validación del sistema

En este capítulo detallamos cómo fue el proceso de validación de nuestro sistema que abarca a todas las herramientas y modelos utilizados posteriormente en la etapa de experimentación. Debido a que la infraestructura experimental comprende una variedad de componentes y tecnologías de software y hardware, buscamos establecer los distintos parámetros de configuración y, en lo posible, los límites que podíamos alcanzar manteniendo confianza en los resultados obtenidos.

En la sección 3.1, definimos dos topologías y las utilizamos para validar que la emulación se comporta como esperamos y verificar que tenemos control y entendimiento de los experimentos. Definimos la interconexión de los clientes de Bitcoin y cómo van a intercambiar información para el funcionamiento del sistema. Estas conexiones se establecen en un archivo de configuración para cada experimento, y luego cada instancia de cliente Bitcoin agrega sus conexiones por medio del llamado RPC `-addnode <IP>:<Puerto>`.

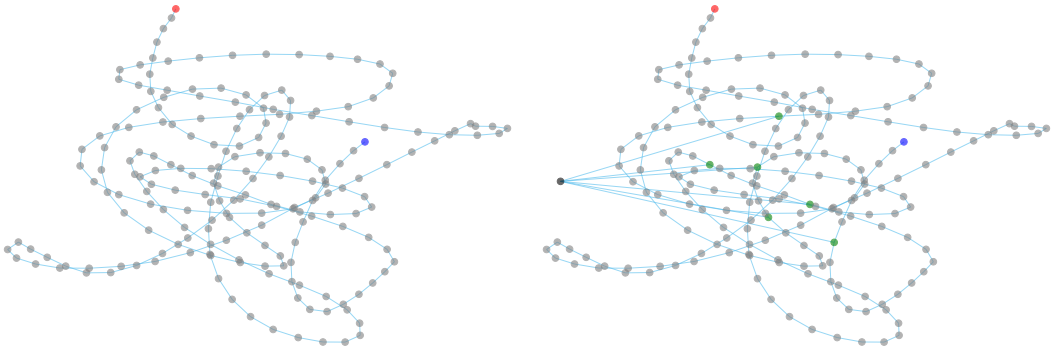
En la sección 3.2, detallamos los resultados obtenidos y como estos nos hicieron ajustar distintas configuraciones para poder plantear los experimentos finales.

3.1 Analizando la emulación

Las siguientes topologías fueron concebidas con la idea de tener control sobre los experimentos a realizar. No nos vamos a enfocar en analizar las medidas descriptas en la sección 2.4 dado que no representan ningún escenario que pueda aportar información relevante. Podremos plantear hipótesis claras sobre cómo debería comportarse cada experimento y verificar, luego, si nuestro entendimiento tanto del protocolo como de la herramienta de emulación coincide con los resultados esperados.

3.1.1 Camino

Se elige esta topología, que podemos ver en la figura 3.1, debido a que su simplicidad permite realizar un seguimiento detallado de los diversos eventos que ocurren en el sistema. En la figura 3.1(a), el nodo de color rojo va a ser el encargado de generar todos los bloques y vamos a analizar su transmisión por la red de Bitcoin, pasando en orden por todos los nodos grises, hasta llegar por último al nodo de color azul. Luego, en la figura 3.1(b) agregamos una pequeña red de retransmisión representada con el nodo negro, la cual interconecta a los nodos de color verde, para verificar que el sistema se comporte según lo esperado, pero siempre con el objetivo de evaluar el comportamiento del sistema en una configuración que resulte relativamente sencilla de controlar. En este nuevo escenario, los bloques ya no deberían viajar únicamente en línea recta por el camino que habíamos preestablecido y comenzamos a tener un poco más de incertidumbre, con los bloques propagándose también por medio de la red `FIBRE`.



(a) Sin FIBRE (b) Con FIBRE

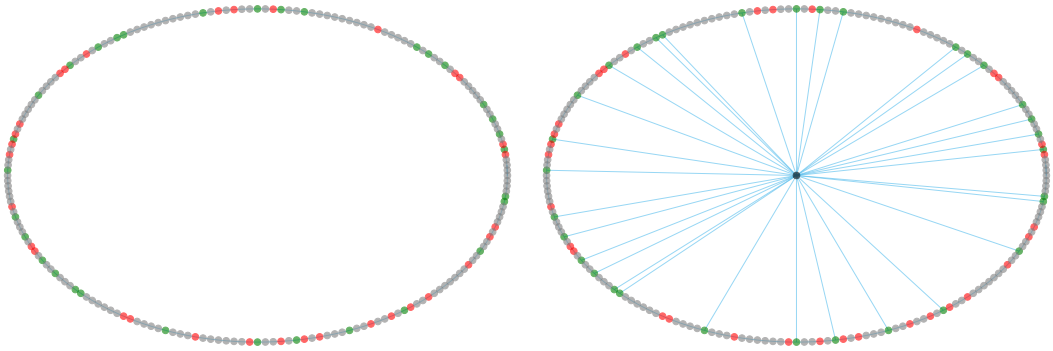
Figura 3.1: Red lógica en forma de camino. El nodo rojo es donde se origina la generación de bloques. El azul el último en recibir la información que se propaga por la red. El negro representa la red FIBRE, que al estar activa consigue disminuir la distancia entre los nodos verdes modificando las distancias en la red.

Como parte de la evaluación del escenario, realizamos el seguimiento de los bloques a través del camino que define la topología con el fin de determinar que sea correcto.

3.1.2 Anillo

Modificando un poco la topología anterior, pasamos a una en la cual todos los clientes tienen exactamente la misma cantidad de vecinos (2). Por lo tanto, ahora la transmisión de bloques no es unidireccional. Al mismo tiempo la generación de bloques va a estar distribuida entre todos los participantes y no en uno solo como en el escenario anterior.

En la figura 3.2 podemos ver la diferencia que presenta la topología cuando no está la red FIBRE como se ve en la figura 3.2(a) y cuando si está como se muestra la figura 3.2(b). La repartición del *hashing power* sigue el diseño que luego tendrán los experimentos finales: 64 clientes se van a repartir el 86 % del poder de cómputo, mientras que el resto de los 136 clientes se distribuyen el 14 % restante. Los nodos de color gris son aquellos que tienen un poder de cómputo despreciable comparados con los clientes de color rojo y verde que son los considerados *pools* de minería. Estos van a tener el mismo poder de cómputo para minar bloques, pero los nodos de color verde, en el escenario que la red de FIBRE está activada, van a estar conectados a la misma como se muestra en la figura 3.2(b). Utilizamos 32 clientes conectados a la red de FIBRE y vemos cómo se comporta el sistema activando y desactivando dicha red.



(a) Sin FIBRE (b) Con FIBRE

Figura 3.2: Red lógica en forma de anillo. Los nodos de color gris son mineros con poco poder de cómputo. Tanto los rojos como los verdes, son *pools* de minería con el mismo poder de cómputo considerable. El nodo negro representa la red `FIBRE` activada, que interconecta a los *pools* de minería de color verde.

Con estas configuraciones perdemos la capacidad de saber fehacientemente de antemano cómo va a ser el camino de los eventos dentro del sistema. Pero podemos poner a prueba el funcionamiento de la red de retransmisión. Con la red de `FIBRE` desactivada el camino más largo entre dos clientes (**diámetro** de la red) es de $N/2$. Al activar la red de retransmisión, conseguimos que el diámetro se reduzca en dos órdenes de magnitud y esperamos que se puedan evidenciar cambios drásticos en los resultados del experimento.

3.2 Comprendiendo la emulación

En esta sección describimos los resultados que fuimos obteniendo en los distintos experimentos y discutimos las dudas que surgieron a lo largo de los mismos.

3.2.1 Camino

Analizando el experimento de la figura 3.1(a) que es un camino sin la red `FIBRE` activada, pudimos verificar que tanto la transmisión de bloques como de transacciones, seguían el orden preestablecido por nuestra configuración.

Al procesar toda la información del experimento, uno de los archivos que generamos contiene de todos los bloques minados, la información relevante a los tiempos de descubrimiento y aceptación.

A modo de ejemplo, ponemos los datos de algunos bloques recolectados que muestran que los mismos siguieron el camino esperado. Tenemos el identificador del bloque, su *hash*, el *hash* de su padre, la cantidad de transacciones en el bloque y la lista de tiempos de descubrimiento en donde ordenadamente muestra que cliente y en que momento descubrió dicho bloque.

Nro Bloque: 97
Hash: f5aed5edc8526092f67e3afb37480d69928a13781336c9cb11e3f9beebeb705e
Hash Padre: 09b5ecc71eda6c85e107ca356bfbab3ea723fdac8a55565550335bd6ac27968f
TXs: 1326
Discovery times: 0: 0 s, 68: 1,721 489 s, 116: 6,300 666 s, 17: 10,830 858 s ...

Nro Bloque: 203
Hash: fe77c55ad1df2fec886957d1a7fb35a5f488c1e2b25f2a16a2e713627ff3d297
Hash Padre: a3d88afd4649d20b1ef0880f06ddee2ad3cc915b77427419c3650195e121aebc
TXs: 1714
Discovery times: 0: 0 s, 68: 1,484 448 s, 116: 3,018 241 s, 17: 4,550 831 s ...

Nro Bloque: 449
Hash: 2bb8d4df291c6c425537156f4085df680d427d2b271ccd49e75e6d20516d68f1
Hash Padre: 25632913167792c6dddd3da833db847d1b596e0f429f1c569ee190e9bf88727e
TXs: 233
Discovery times: 0: 0,0 s, 68: 1,499 666 s, 116: 6,012 758 s, 17: 25,567 841 s ...

En la figura 3.3 podemos ver los tiempos de descubrimiento promedio para los dos escenarios, con la red FIBRE activada y desactivada. Se puede apreciar inmediatamente que los tiempos de descubrimiento de color verde que representan el escenario de la figura 3.1(b) con la red FIBRE activada mejoran sustancialmente los tiempos de propagación.

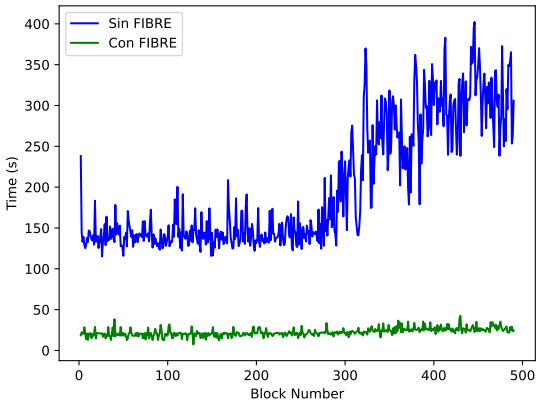
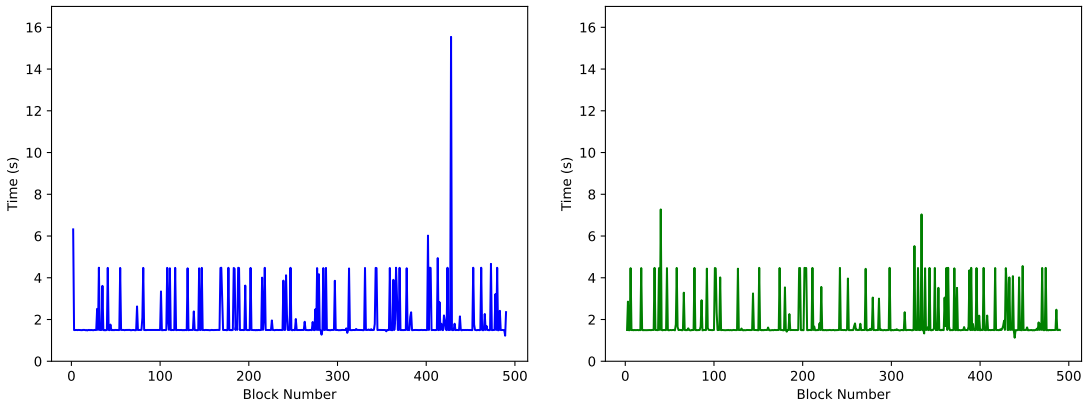


Figura 3.3: Tiempos de descubrimiento promedio. De color azul los tiempos del escenario 3.1(a) en donde `FIBRE` no está activado y los bloques siguen un camino secuencial y de color verde el escenario 3.1(b), donde la incorporación de `FIBRE` modifica la topología lógica de la red y por consiguiente el camino de los bloques por la misma.

Nos surgieron algunas dudas con respecto a los gráficos resultantes de estos experimentos. No comprendíamos por qué la medición de tiempo mínimo de propagación de bloques no era constante y tenía varios picos. Tal como ilustran las dos figuras 3.4 en las cuales el fenómeno se da tanto cuando está activada la red `FIBRE`, como cuando no lo está.

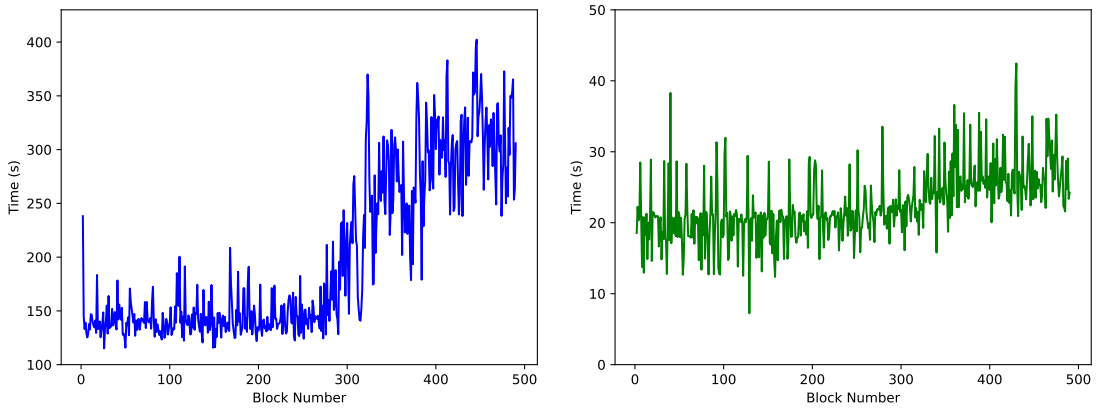


(a) Sin FIBRE

(b) Con FIBRE

Figura 3.4: Tiempo de descubrimiento mínimo. De color azul los tiempos del escenario 3.1(a) y de color verde 3.1(b).

También nos resultó inquietante que el tiempo de propagación promedio no sea estable y que comience a incrementarse a partir de la mitad del experimento. La figura 3.5 muestra esta situación en los dos escenarios (con y sin `FIBRE`), siendo mucho más evidente en el primer caso, pero también se vislumbra la tendencia cuando `FIBRE` está activado.



(a) Sin FIBRE (b) Con FIBRE

Figura 3.5: Tiempo de descubrimiento promedio en distintas escalas, mostrando en ambos casos anomalías incrementandose los tiempos a partir del bloque minado número 300.

Realizamos un seguimiento detallado, tanto de la información que recolectamos como de los archivos `debug.log` para encontrar una explicación a los fenómenos que no entendíamos. Pudimos ver en esos archivos una correlación entre los picos del tiempo mínimo de propagación, con la generación de bloques con poco tiempo de diferencia y bloques con una gran cantidad de transacciones. Comenzamos a sospechar que esto era un comportamiento razonable dentro del buen funcionamiento del sistema.

Para validar esta hipótesis, diseñamos un nuevo experimento que nos aportó mucha claridad. Utilizando la misma configuración, realizamos el experimento con el sistema descripto anteriormente y otros tres escenarios con pequeñas modificaciones con el fin de entender el comportamiento del sistema.

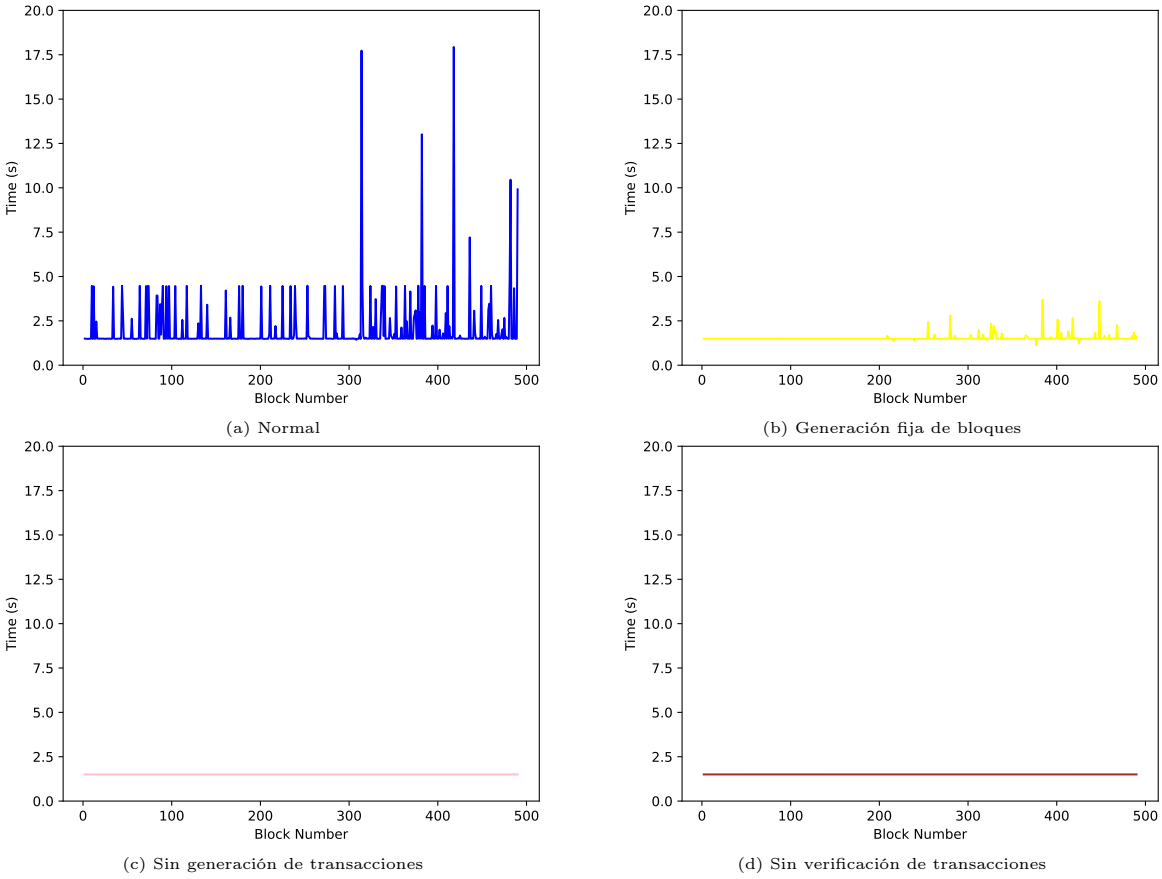
1. Normal
    - 120 clientes
    - 300 segundos de target
    - 1 segundo de delay entre clientes
    - `FIBRE` desactivado
    - Generación de transacciones cada 60 segundos

2. Generación de bloques en intervalos fijos.

3. Sistema sin transacciones.

4. Sistema sin verificación de transacciones.

La figura 3.6 presenta los los tiempos de descubrimiento mínimo para los cuatro escenarios.



(a) Normal

(b) Generación fija de bloques

(c) Sin generación de transacciones

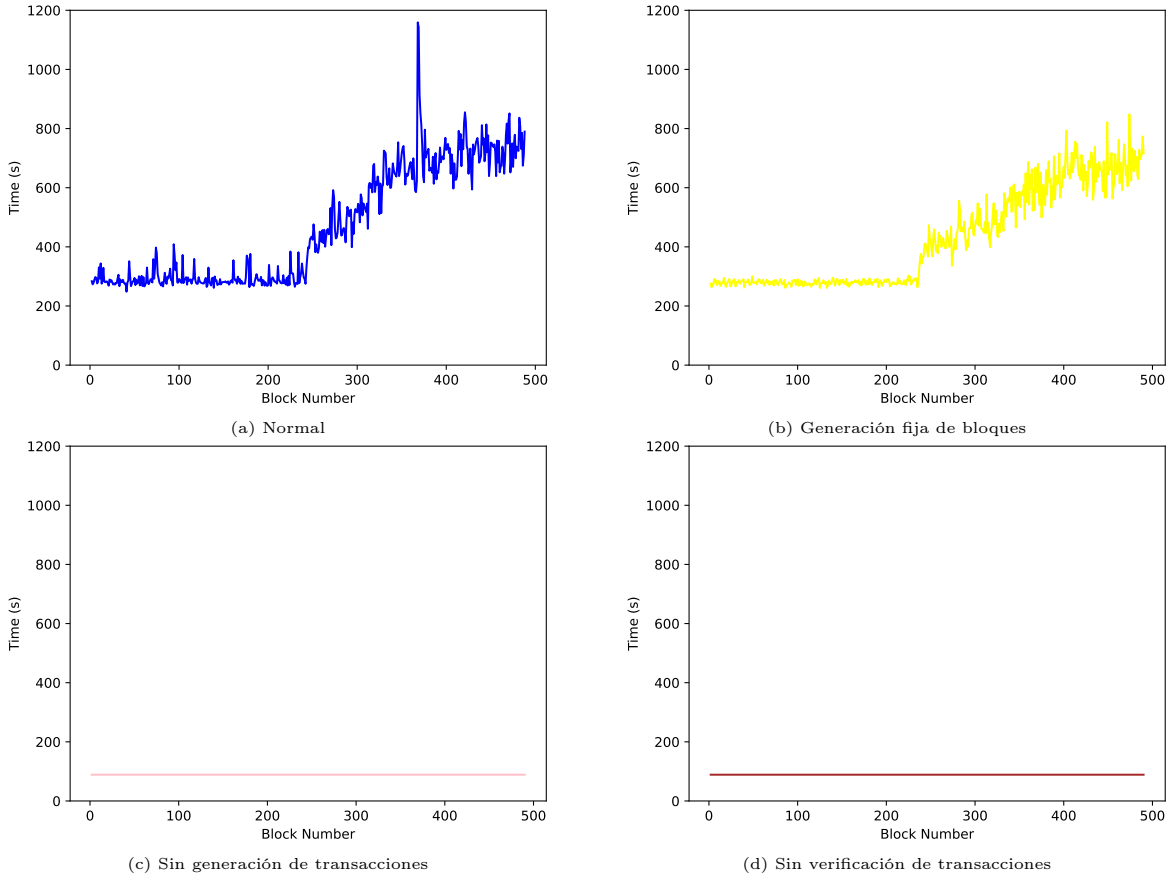
(d) Sin verificación de transacciones

Figura 3.6: Tiempo de descubrimiento mínimo para distintos escenarios de prueba.

Vemos como tanto el sistema que no genera transacciones (figura 3.6(c)), como aquel que no realiza ninguna verificación (figura 3.6(d)), a la hora de recibir los bloques tienen un tiempo de descubrimiento mínimo estable a lo largo de toda la ejecución. Esto es razonable, dado que en estos dos escenarios el sistema solo se encarga de transmitir información de un lado a otro sin ningún tipo de procesamiento.

En la imagen del experimento con generación de bloques en un tiempo espaciado y fijo (figura 3.6(b)), podemos ver cómo el tiempo mínimo es mucho más estable que en el expe-

rimento original con algunas pequeñas perturbaciones luego de la mitad del experimento pero en menor medida que el experimento original.



(a) Normal

(b) Generación fija de bloques

(c) Sin generación de transacciones

(d) Sin verificación de transacciones

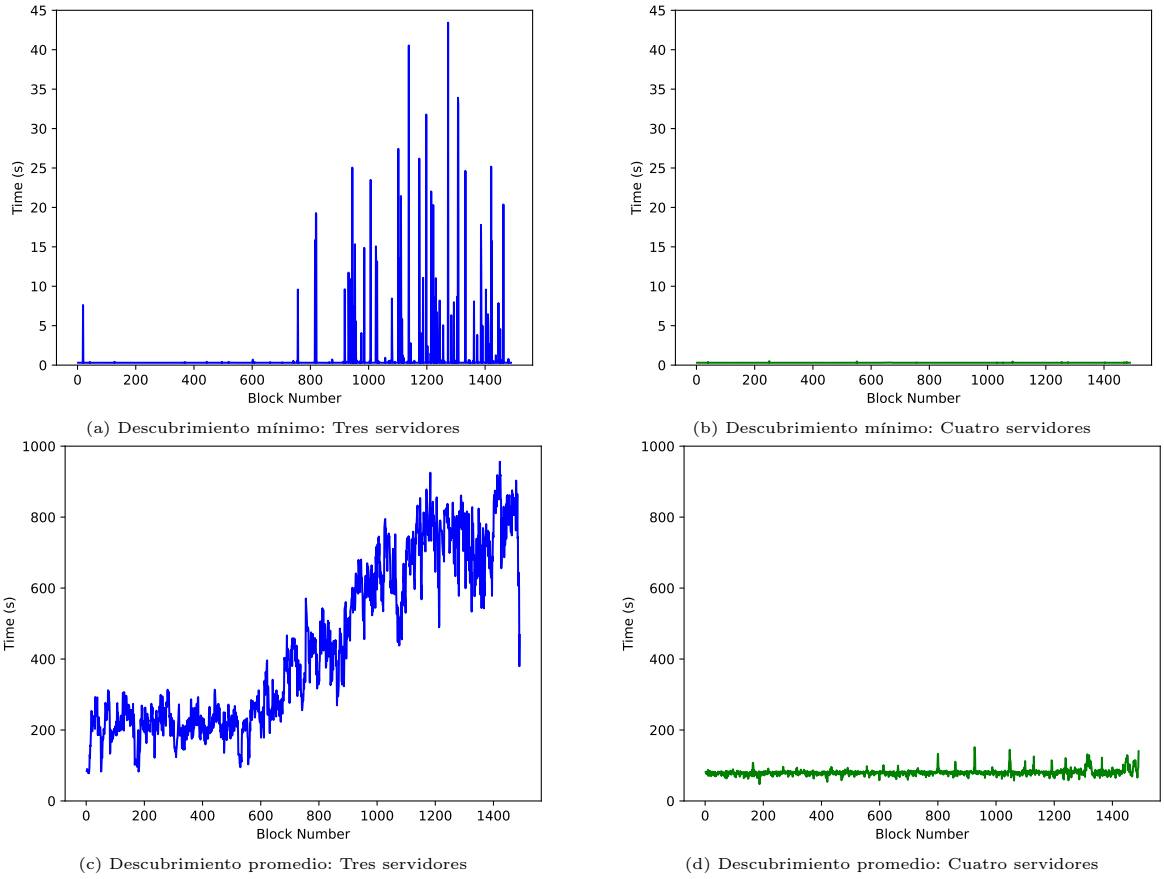
Figura 3.7: Tiempo de descubrimiento promedio para distintos escenarios de prueba.

Sin embargo, seguimos sin encontrar una explicación al incremento paulatino del tiempo de descubrimiento promedio a lo largo del experimento. Pudimos volver a verificarlo en la figura 3.7 y ver cómo están relacionados el incremento de las perturbaciones en el tiempo mínimo con el incremento en la curva del tiempo promedio. Esto sucede tanto en el caso de estudio como en el experimento con generación de bloques fija.

Comenzamos a sospechar que podrían ser problemas con la infraestructura en donde se ejecutan los experimentos.

Pensamos un experimento para analizar la cantidad de clientes con el que se carga cada servidor. Corrimos experimentos con 180 clientes en tres servidores (60 clientes por ser-

vidor) y en cuatro servidores (45 clientes por servidor). Tuvimos resultados contundentes como se ven en la figura 3.8.



(a) Descubrimiento mínimo: Tres servidores

(b) Descubrimiento mínimo: Cuatro servidores

(c) Descubrimiento promedio: Tres servidores

(d) Descubrimiento promedio: Cuatro servidores

Figura 3.8: Tiempo de descubrimiento mínimo y promedio en un sistema con 180 clientes distribuidos en distinta cantidad de servidores.

Pudimos verificar que la cantidad de clientes por servidor afecta directamente el desempeño del sistema saturando la infraestructura y produciendo resultados que no representaban el comportamiento del sistema. La ocupación de la placa de red y el *delay* entre todos los hosts emulados no proveyó ningún patrón que nos mostrara que el sistema estaba saturando, con lo que estas dos magnitudes no podían ser utilizadas como indicadores de un mal funcionamiento de la infraestructura. En base a estos resultados, decidimos limitar a 50 clientes por servidor y seguir haciendo experimentos con otras configuraciones.

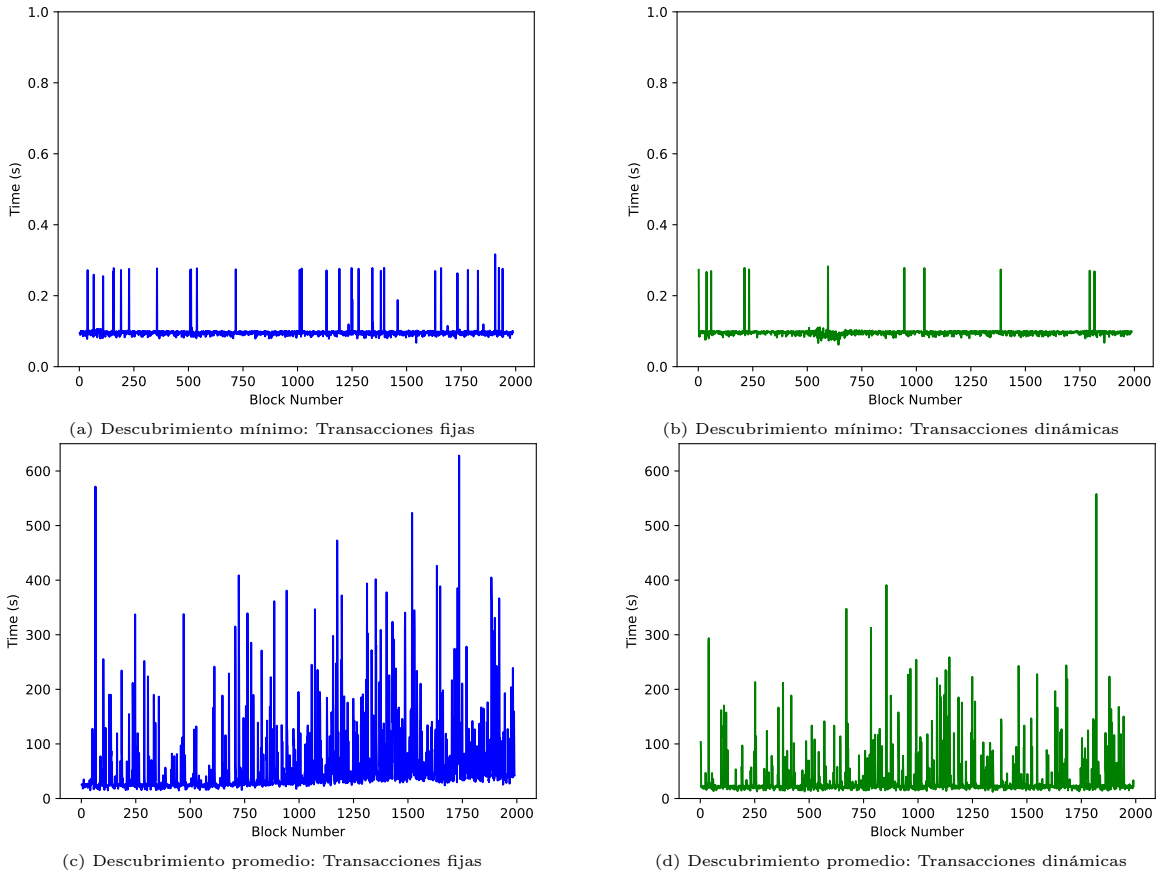
3.2.2 Anillo

El anillo es una topología simétrica en la cual todos los nodos tienen **exactamente** dos conexiones. Nuevamente, al aumentar la cantidad de transacciones en el sistema volvimos a tener problemas con la infraestructura. El tiempo de aceptación promedio no se mantenía estable y se incrementaba a medida que avanzaba el experimento.

Luego de varias pruebas, pudimos concluir que esto era producto del sistema de generación de transacciones y decidimos modificarlo. Pasamos de una configuración en la cual todos los clientes generan una transacción cada 60 segundos, a un sistema en el cual cada cliente genera entre 0 y x transacciones en un intervalo de tiempo mayor, tal como explicamos en la sección 2.2.2.2.

De esta manera conseguimos resultados más estables. Podemos ver la figura 3.9 que compara los tiempos de descubrimiento mínimo y promedio del mismo experimento utilizando los dos sistemas de generación de transacciones. Cuando se utiliza el sistema de generación de transacciones dinámico, mejoran los tiempos y se consiguen respuestas con menos variaciones. Comparando la figura 3.9(a) con la 3.9(b), vemos que hay muchos menos picos en el tiempo mínimo de descubrimiento. Por otro lado, la figura 3.9(c) tiene picos en el tiempo promedio de descubrimiento más elevados en mayor cantidad de ocasiones e, incluso, podemos ver cómo la base no se ve como una línea recta, si no que alrededor del bloque número 750 comienza a tener una pendiente positiva. Comparando el comportamiento anterior con el reflejado en la figura 3.9(d), vemos un gráfico menos denso y lo más importante es que la base del tiempo promedio de descubrimiento de bloques se mantiene como una línea recta entendiéndose como estable a lo largo de todo el experimento.

Mientras ejecutábamos los experimentos desarrollamos una herramienta que nos permitió monitorear el estado de la infraestructura para determinar los casos en que se podrían producir resultados inválidos. En Linux se puede analizar la carga del sistema verificando la cantidad de procesos que están siendo ejecutados y los que se encuentran en estado *waiting*. Por medio del seguimiento de la carga promedio de cada uno de los servidores, se puede



(a) Descubrimiento mínimo: Transacciones fijas

(b) Descubrimiento mínimo: Transacciones dinámicas

(c) Descubrimiento promedio: Transacciones fijas

(d) Descubrimiento promedio: Transacciones dinámicas

Figura 3.9: Tiempo de descubrimiento mínimo y promedio para distintos modelos de generación de transacciones.

determinar si éste ha estado funcionando dentro de los parámetros de carga aceptables para que el sistema se comporte adecuadamente. Las magnitudes que se monitorean son la carga media del servidor en el último minuto, últimos cinco minutos y últimos quince minutos.

El valor de *load average* está vinculado con la cantidad de procesadores. Describamos un ejemplo en un sistema con ocho núcleos.

I) Load Average = 4

- El 50 % de los procesadores estuvieron ociosos (estado *idle*) y ningún proceso estuvo en estado *waiting* a la espera de recursos.

II) Load Average = 8

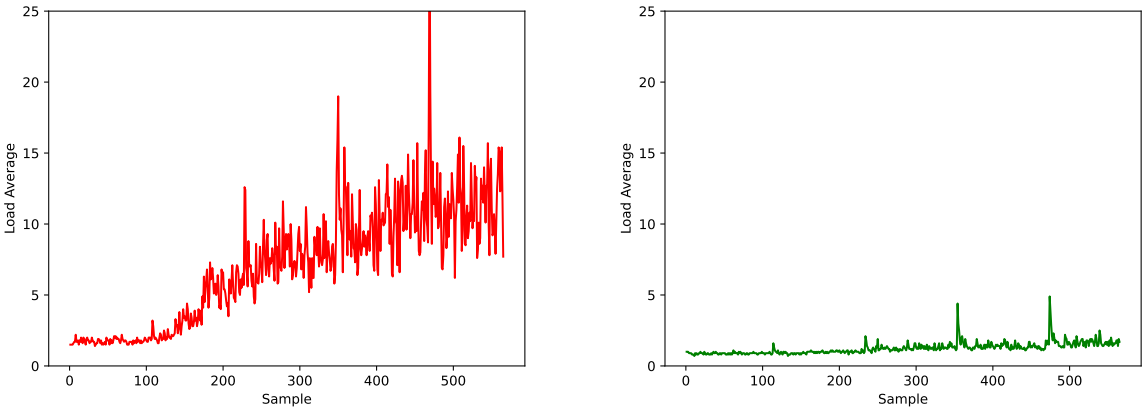
- Todos los procesadores estuvieron trabajando y ningún proceso estuvo en estado

de *waiting*.

III) Load Average = 10

- El sistema estuvo sobrecargado un 25 %, en promedio dos procesos estuvieron en estado *waiting* a la espera de recursos.

Decidimos almacenar el valor del *load average* de 15 minutos en intervalos de 11 minutos por servidor a lo largo del desarrollo de los experimentos. Notamos que, en los cuatro servidores utilizados, este valor comienza a incrementarse cuando ejecutábamos los experimentos que presentaron luego resultados inválidos. En cambio, en los experimentos que no notamos perturbaciones ni comportamientos inesperados, pudimos ver cómo este valor se mantenía relativamente bajo. La figura 3.10 compara dos experimentos mostrando el *load average* de 15 minutos. En la figura 3.10(a) se ve un claro incremento en la carga del sistema que presenta alguno picos agudos pero a la vez un sostenido incremento lo cual vemos directamente correlacionado con el aumento de la base del tiempo promedio de descubrimiento de bloques que se percibe en la figura 3.9(c), esta configuración produjo resultados inválidos del experimento y tuvieron que ser descartados. La figura 3.10(b), por otro lado, muestra una evolución de la carga que se mantiene casi constante (salvo algún pico aislado) presentando un sistema que se encuentra *holgado* respecto al uso de los recursos de cómputo.



(a) Funcionamiento incorrecto

(b) Funcionamiento correcto

Figura 3.10: El gráfico que surgió sobre el *load average* de 15 minutos para uno de los servidores. En los tres restantes, se dieron resultados similares. Vemos que los escenarios con muchos clientes por servidor y una generación fija de transacciones se corresponden con una carga muy superior al del escenario que consideramos correcto.

Hicimos una alerta que nos avisa cuando se supera el 100 % de carga del servidor (*load average* > 64 en el caso de los servidores utilizados por contar con 64 núcleos) analizando el *load average* de 1 minuto, pero vimos que aunque con una frecuencia mucho menor, este valor se supera en los dos experimentos. Por lo tanto, no pudimos utilizar esta medida como indicador para saber si un experimento era correcto.

Terminamos estableciendo una cota arbitraria de 10 para el *load average* de 15 minutos y definimos que aquellos experimentos que no la superen en ningún momento en ninguno de los servidores utilizados, iban a ser considerados experimentos válidos.

Capítulo 4

Experimentación

En este capítulo nos vamos a centrar en analizar escenarios complejos y ver el impacto de las redes de retransmisión en condiciones que tiendan a ser las de funcionamiento del sistema real.

A partir de la siguiente sección vamos a mostrar los detalles de generación de estos escenarios en sus distintas variantes.

Utilizando la topología física detallada en la sección 2.3.1, que es una constante que no sufre modificaciones a lo largo de todos los experimentos. Establecemos tres escenarios con la misma topología lógica, pero variando la distribución del *hashing power*. Por último, agregando la red de `FIBRE`, indefectiblemente se termina modificando la topología lógica cambiando las distancias entre los distintos nodos de la red P2P original del protocolo Bitcoin.

4.1 Barabási

El modelo Barabási-Albert [AB02] surgió como una opción interesante para poner a prueba el funcionamiento del sistema de Bitcoin. Las redes libres de escala se utilizan frecuentemente para proponer sistemas que tengan características topológicas similares a Internet, redes sociales y redes *peer-to-peer*. En contraposición al modelo de redes aleatorias Erdös

y Rényi, el modelo de Barabási intenta representar el crecimiento continuo de nodos en la red y la preferencia por conectarse a nodos que ya tienen muchas conexiones. Si bien no representa la dinámica real de la generación de conexiones que utiliza el protocolo del cliente Bitcoin como se presentó en la sección 1.2, para la escala que manejamos es un escenario atractivo por la topología que se genera y la posibilidad de analizar el impacto de la red de retransmisión. Pueden interpretarse que las comunidades que se generan alrededor de los nodos superconectados actúan como clientes que se mantienen continuamente activos y con tiempos de respuestas razonables y no así aquellos que utilizan la red intermitentemente. Si bien el protocolo de Bitcoin se basa en una generación de conexiones basado en la selección aleatoria, se vio en la red que algunos nodos incrementan el número inicial de conexiones por medio de una política propia de selección que no necesariamente se basa en la aleatoriedad.

En el modelo Barabási, se define la cantidad final de nodos y la cantidad promedio de conexiones que se agregan con cada nodo. En el protocolo de Bitcoin, por defecto, cada cliente comparte información con 8 peers. Al tener que representar a escala una red lógica que interconecta 200 clientes, este valor fue reducido a 1 para no obtener una red superconectada y que los tiempos de propagación sobre la misma no sean casi instantáneos. Por lo tanto los grafos resultantes fueron de generar una red `Barabasi(200,1)`.

Por último, vamos a repartir el *hashing power* de la red en tres escenarios que vamos a detallar a continuación para intentar ver cómo se comporta el sistema con diferentes distribuciones de poder de cómputo. Distribuimos intencionalmente los distintos *pools* de minería dentro de la red para que no queden ni super-conectados ni en la periferia del grafo generado. Siendo los *pools* de minería aquellos que insertan la mayor cantidad de bloques en la red, que estén muy juntos o muy separados no representa fielmente lo que ocurre en el sistema real. Tenemos distintos *pools* de minería distribuidos alrededor del mundo, e intentamos que eso se vea reflejado en las comunidades resultantes del grafo Barabási.

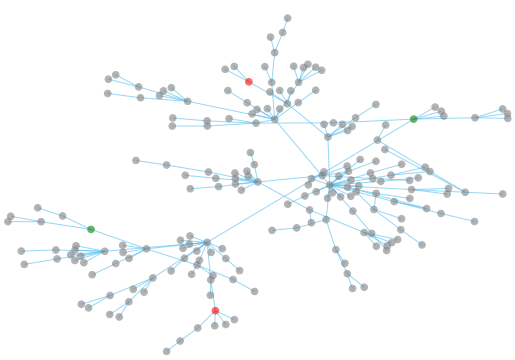
4.1.1 Distribución del *Hashing Power*

En todos los experimentos se asigna el 86 % del *hashing power* a los *pools* de minería y el 14 % restante se distribuye entre los clientes comunes. La diferencia que tenemos entre cada experimento es la cantidad de *pools* y clientes comunes y, por consiguiente, la cantidad de *hashing power* asociado a cada uno. En todos los experimentos, la mitad de los *pools* de minería van a estar conectados a la red `FIBRE` cuando ésta esté encendida.

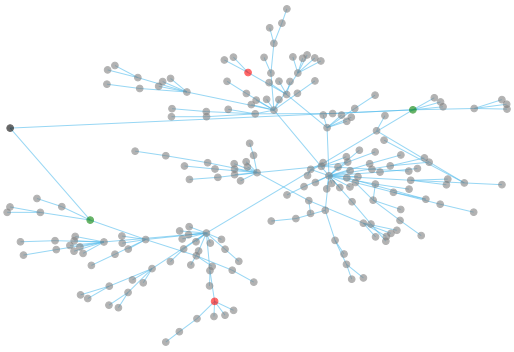
En el primer escenario con 4 *pools* y 196 clientes comunes, cada *pool* de minería va a tener el 21,5 % del *hashing power*, mientras que el resto de los clientes un 0,0714 %.

En el segundo escenario con 16 *pools* y 184 clientes comunes, cada *pool* de minería va a tener el 5,375 % del *hashing power*, mientras que el resto de los clientes un 0,0761 %.

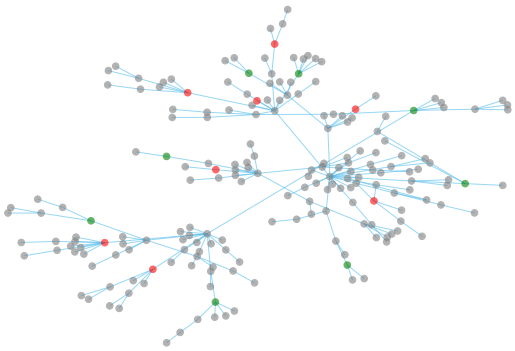
En el tercer y último escenario con 64 *pools* y 136 clientes comunes, cada *pool* de minería va a tener el 1,3437 % del *hashing power*, mientras que el resto de los clientes un 0,1029 %.



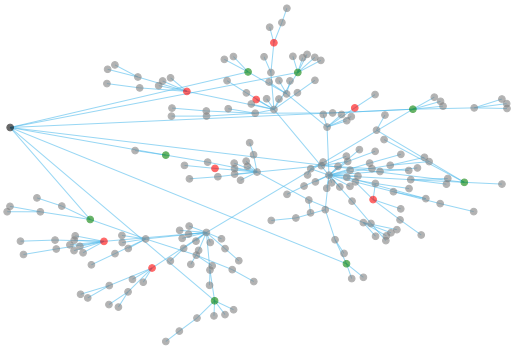
(a) 4 *pools* de minería sin la red FIBRE encendida.



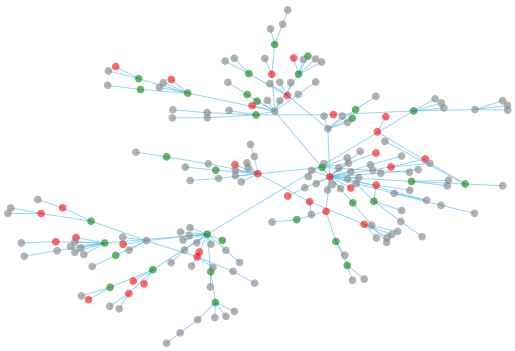
(b) 4 *pools* de minería con la red FIBRE encendida y 2 *pools* de minería conectados.



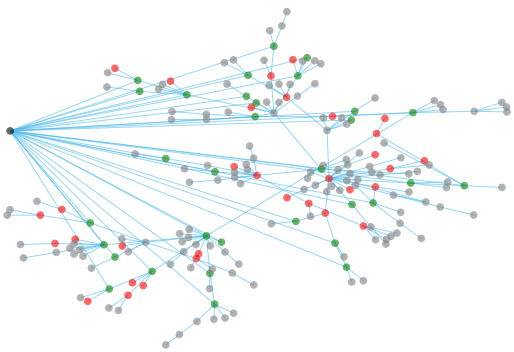
(c) 16 *pools* de minería sin la red FIBRE encendida.



(d) 16 *pools* de minería con la red FIBRE encendida y 8 *pools* de minería conectados.



(e) 64 *pools* de minería sin la red FIBRE encendida.



(f) 64 *pools* de minería con la red FIBRE encendida y 32 *pools* de minería conectados.

Figura 4.1: Distribución de nodos para la topología Barabási seleccionada. En color negro la red FIBRE, en color rojo los *pools* de minería que no pertenecen a la red de retransmisión FIBRE y en verde aquellos que pasan a estar conectados cuando la misma está encendida. En color gris los clientes con un bajo poder de *hashing power*.

4.2 Resultados

En esta sección se presentan los resultados y se realiza el análisis de los experimentos realizados. Para lograr una robustez estadística, se ejecutó varias veces la misma instancia de cada experimento y se promediaron los resultados para minimizar fenómenos aleatorios y particulares que se puedan producir en algún experimento aislado. Realizamos la mayor cantidad de iteraciones que el tiempo disponible de acceso a la infraestructura de cómputo nos permitió. Con un mínimo de tres experimentos en el caso que el *target time* de la red es de 600 s, caso en que los experimentos demoraron una semana de cómputo aproximadamente.

En la sección 4.2.1, comenzamos analizando la cantidad de *forks* en el sistema, que nos puede dar una idea rápida de cómo se comporta el sistema bajo las distintas configuraciones.

Luego, en la sección 4.2.2, analizamos los tiempos de propagación del protocolo que creemos que tienen una importancia superlativa en el funcionamiento del mismo.

En la sección 4.2.3 vemos cómo se comporta el *hashing power* desperdiciado, que es una medida que resulta más específica a la hora de analizar el rendimiento del sistema y el desperdicio de recursos asociado.

Por último, en la sección 4.2.4 realizamos varias medidas en los tres tipos de clientes propuestos al realizar los experimentos, con la idea de analizar si alguno de estos consigue algún diferencial que se pueda explicar por la introducción de la red de retransmisión.

4.2.1 Forks

La distinción que habíamos propuesto en un principio para distintos niveles de *forks* no nos resultó de utilidad, dado que hubieron muy pocas bifurcaciones de dos bloques y ninguna de nivel tres a lo largo de todos los experimentos realizados y sus repeticiones. Por lo tanto, solo vamos a analizar la cantidad de *forks* totales de cada experimento, que incluye los *forks* de nivel 1 y 2.

En la serie de figuras 4.1 podemos ver la cantidad de *forks* que hubo en todos los escenarios planteados. Pudiendo hacer distinción entre los *forks* que fueron introducidos por los *pools* de minería que pertenecían a la red `FIBRE` (con la misma apagada o prendida), aquellos *pools* que solo se comunicaban por la red tradicional (MINER) y el resto de clientes comunes (NODE).

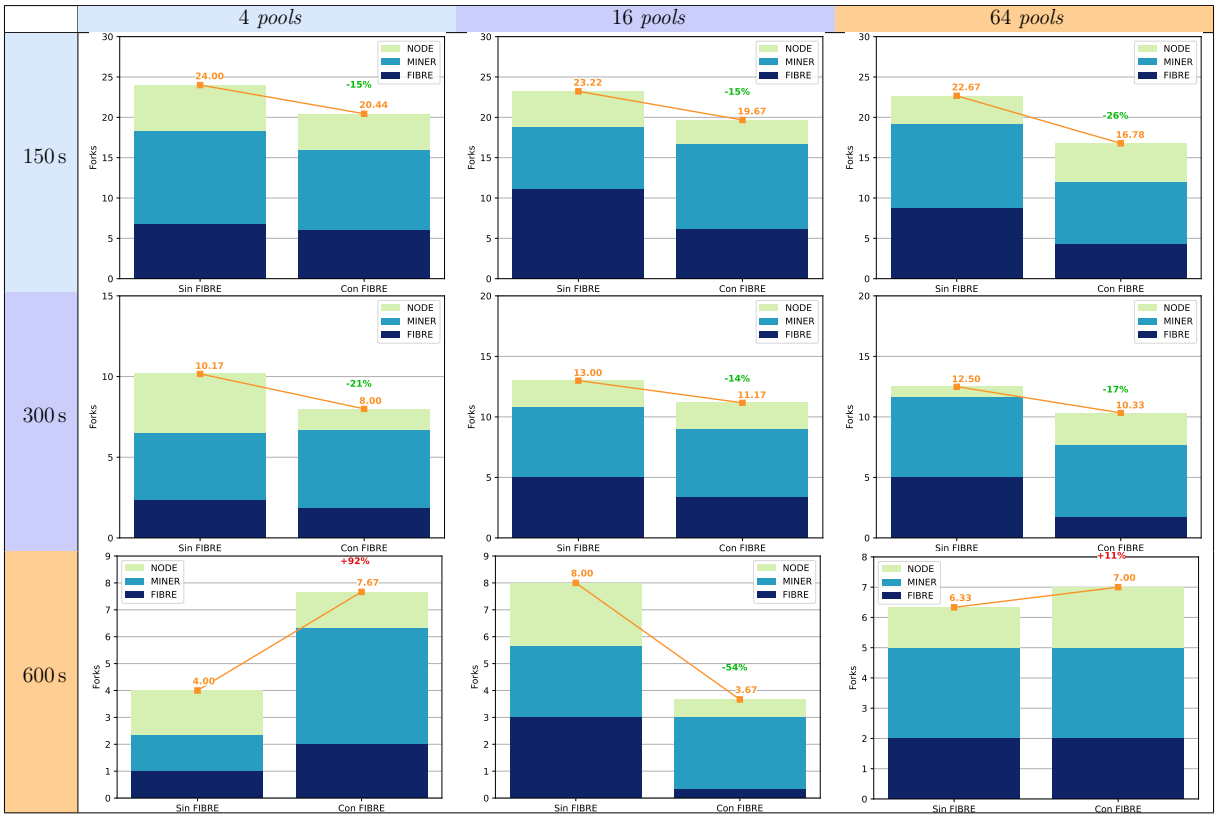


Tabla 4.1: Comparación de la cantidad de *forks* promedio para cada experimento realizado

Rápidamente podemos apreciar cómo la cantidad de *forks* en la red disminuyen en función del aumento del *target time* de la misma. Sin importar si la red de retransmisión `FIBRE` se encuentra activa o no. En los experimentos con un *target time* de 150 s la cantidad de *forks* son alrededor del 4 % del total (alrededor 20). Con 300 s de *target time* este valor disminuye a casi un 2 % (alrededor de 10). Por último, con el *target time* original del protocolo de 600 s, la cantidad de *forks* producidos en los experimentos es de alrededor del 1 % del total de bloques recolectados (alrededor de cinco).

Inversamente a lo que imaginábamos, no hay evidencia suficiente como para afirmar que el

aumento de la cantidad de *pools* de minería que se reparten la mayor cantidad del *hashing power*, produzca algún tipo de incremento en la cantidad de *forks* del sistema. En los experimentos con 4, 16 y 64 pooles de minería respectivamente, vemos que sus resultados mantienen promedios similares para cada uno de los *target time* establecidos. Alrededor de 20 forks para 150 segundos de target time, 10 forks para los experimentos con 300 segundos de target time y unos 5 forks para cuando se utilizan 600 segundos de target time.

Con respecto a la injerencia del agregado de la red de retransmisión de `FIBRE` a los experimentos, no podemos identificar ningún patrón que nos permita concluir algo relevante con respecto a la cantidad de *forks* totales del sistema. Vemos que, en la mayoría de los casos, disminuyen la cantidad total de *forks* en los experimentos que tienen la red de `FIBRE` activada. Pero existen caso en los cuales, a pesar de activar la red `FIBRE`, la cantidad de *forks* totales se ve incrementada. En los experimentos con 600 s de *target time*, podemos ver como con cuatro pooles de minería y red de FIBRE activa casi se duplican la cantidad de forks totales del experimento para, luego en el experimento con 16 pooles de minería, se de el escenario inverso y, por último, con 64 pooles de minería estén casi equiparados.

Otro análisis que no brindó información relevante es la diferenciación entre los distintos tipos de clientes. No pudimos encontrar ningún patrón que sigan los experimentos con respecto a la segmentación de los clientes. Vemos que en todos los clientes (mineros comunes, *pools* de minería y *pools* de minería FIBRE) a veces crece su cantidad de *forks* y otras veces decrece, cuando la red de `FIBRE` está apagada o prendida.

En el tabla 4.2 podemos ver el detalle de la cantidad promedio de *forks* para cada tipo de cliente de todos los experimentos con su respectivo desvío estándar.

Cantidad Forks		4 *pools*		16 *pools*		64 *pools*	
		Estado de la red `FIBRE`					
		Desactivada	Activada	Desactivada	Activada	Desactivada	Activada
Target 150 s	Comun	5.22 (6.62)	4.00 (3.56)	4.44 (3.36)	3.00 (2.44)	3.56 (5.14)	4.78 (2.17)
	Minero	10.00 (9.56)	8.56 (10.02)	7.67 (9.11)	10.56 (6.47)	10.33 (9.33)	7.78 (3.95)
	Fibre	5.44 (1.58)	4.56 (2.47)	11.11 (8.10)	6.11 (5.43)	8.78 (15.28)	4.22 (4.62)
	Total	20.67 (35.11)	17.11 (12.99)	23.22 (25.95)	19.67 (12.44)	22.67 (33.78)	16.78 (21.95)
Target 300 s	Comun	3.67 (1.22)	1.33 (0.22)	2.17 (1.14)	2.17 (0.81)	0.83 (0.47)	2.67 (0.89)
	Minero	4.17 (4.14)	4.83 (4.47)	5.83 (3.14)	5.67 (20.89)	6.67 (5.56)	6.00 (1.00)
	Fibre	2.33 (1.56)	1.83 (1.14)	5.00 (2.67)	3.33 (1.22)	5.00 (4.33)	1.67 (2.22)
	Total	10.17 (5.81)	8.00 (5.00)	13.00 (6.00)	11.17 (17.14)	12.50 (18.25)	10.33 (2.56)
Target 600 s	Comun	1.67 (1.56)	1.33 (0.22)	2.33 (1.56)	0.67 (0.22)	1.33 (0.22)	2.00 (0.67)
	Minero	1.33 (0.22)	4.33 (0.22)	2.67 (2.89)	2.67 (2.89)	3.00 (2.00)	3.00 (2.67)
	Fibre	1.00 (0.67)	2.00 (2.00)	3.00 (2.00)	0.33 (0.22)	2.00 (2.67)	2.00 (2.00)
	Total	4.00 (2.67)	7.67 (2.89)	8.00 (4.67)	3.67 (0.89)	6.33 (4.22)	7.00 (2.67)

Tabla 4.2: Datos del promedio de la cantidad de *forks* y su desvío estándar para todos los experimentos realizados

Nos llamó la particularmente la atención los valores del desvío estándar para los experimentos con 150 s de *target time*, en el cual los *forks* totales llegan a tener un desvío estándar del 150 % del valor promedio. Son muy variables los resultados de los experimentos, pero podemos ver cómo mejora el desvío estándar en función del aumento del *target time* de la red. Aunque encontramos un caso bastante raro en donde el promedio de su cantidad de *forks*, de los *pools* de minería que no utilizan la red `FIBRE` para el experimento de *target time* de 300 s es de $5,67$ con un desvío estándar de 20,89. En los experimentos con 600 s de *target time*, baja hasta alrededor del 50 % el desvío estándar, aunque siga siendo un valor bastante elevado. Pensamos que esta estabilización de los resultados es exclusivamente producto del aumento del *target time*, pero al tener solo tres muestras para cada uno de los experimentos, no podemos confirmarlo. No pudimos determinar a que se debía este fenómeno, conjeturamos que podía ser producto del tamaño de la red o la topología utilizada y nos llevamos la sensación de que esta magnitud es muy sensible a los tiempos de propagación de la misma.

Analizar únicamente la cantidad de *forks* en la red no es algo que pueda capturar la degradación del sistema de forma certera. Cuando se produce un *fork*, influye en medida de como éste particiona la red. El peor escenario es cuando esta partición es alrededor del 50 %. Eventualmente, cuando este *fork* se resuelva y solo un bloque pase a formar parte de la cadena principal, alrededor del 50 % de los recursos fueron desperdiciados intentando

extender la cadena principal sobre un bloque que terminó desestimándose. En cambio, si se produce un *fork* que solo alcanza a unos pocos clientes, la mayor parte de los recursos del sistema van a seguir orientados a trabajar sobre la cadena principal.

La cantidad de *forks* del sistema parece una medida atractiva, dado que se puede calcular muy fácilmente y nos podría dar una idea general de cómo es el funcionamiento del sistema. Pero una dimensión más interesante la presenta el analizar los tiempos de propagación o el poder de cómputo desperdiciado ya que pueden entregar más detalle acerca del funcionamiento del sistema. Ambas magnitudes se presentan en las siguientes secciones.

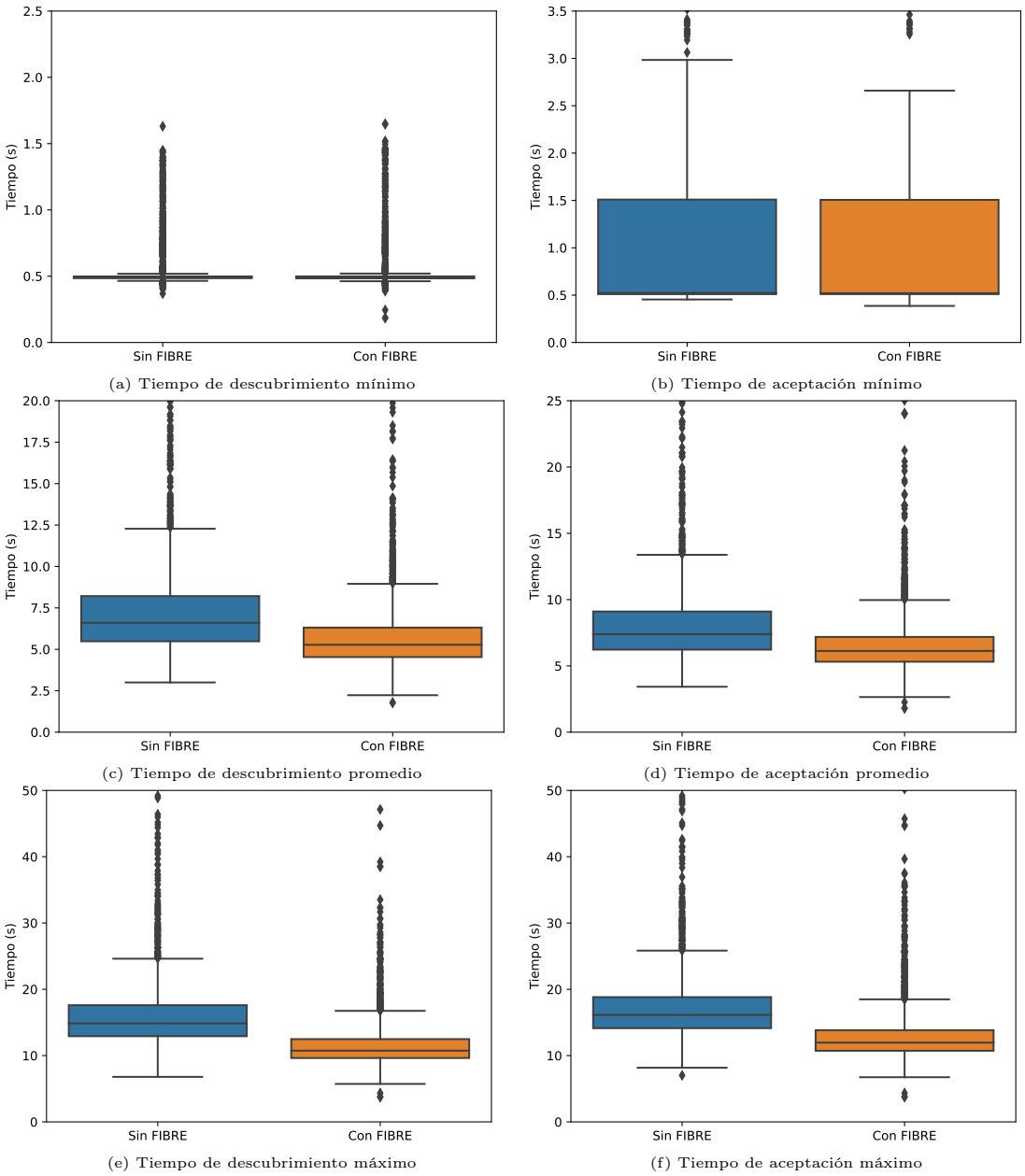
4.2.2 Tiempos de propagación

Utilizamos la figura 4.2 que presenta los tiempos de propagación en los experimentos con 300 s de *target time* y una cantidad de 16 *pools* de minería para realizar algunas reflexiones sobre esta medida.

Los tiempos mínimos de propagación se calcula en función del cliente más cercano a aquel que minó el nuevo bloque. Con la red como la planteamos, teniendo todos la misma latencia entre clientes este valor debería encontrarse cercano a los 500 ms.

Tanto la figura 4.2(a) que muestra el tiempo de descubrimiento mínimo, como la figura 4.2(b) que muestra el tiempo de aceptación mínimo, presentan algunas anomalías con valores menores al establecido en el *delay* de la red. Este comportamiento se lo atribuimos a la naturaleza de la sincronización de los relojes en los distintos servidores que toman las muestras. Vemos que la mediana se encuentra exactamente en el valor de *delay* establecido para la red. Y la cantidad de valores por debajo de dicho *delay* es despreciable en comparación al tamaño de la muestra, que para el caso de las figuras que corresponde a un experimento con 300 s de *target time*, el cual contempla seis repeticiones con 500 bloques minados en cada uno, dando un total de 3000 muestras.

Nos percatamos de estas anomalías en la etapa de análisis detallado de los resultados de los experimentos. Para ese momento ya no contábamos con la infraestructura disponible



(a) Tiempo de descubrimiento mínimo

(b) Tiempo de aceptación mínimo

(c) Tiempo de descubrimiento promedio

(d) Tiempo de aceptación promedio

(e) Tiempo de descubrimiento máximo

(f) Tiempo de aceptación máximo

Figura 4.2: Tiempos de descubrimiento y aceptación para el experimento con *target time* 300 s y 16 *pools*.

para realizar más expermientos. Pero se podría poner a prueba nuestra hipotesis realizando experimentos en un único servidor y verificando que los tiempos de propagación mínimos, nunca se encuentren por deabjo del *delay* establecido.

En la figura 4.2(a) vemos que los *outliers* con un tiempo mayor a la mediana, que pueden ser producto de problemas estocásticos (como la pérdida de paquetes o algún retraso en el cliente), nunca llegaron a ser mayores a cuatro veces el *delay* establecido para la red, lo

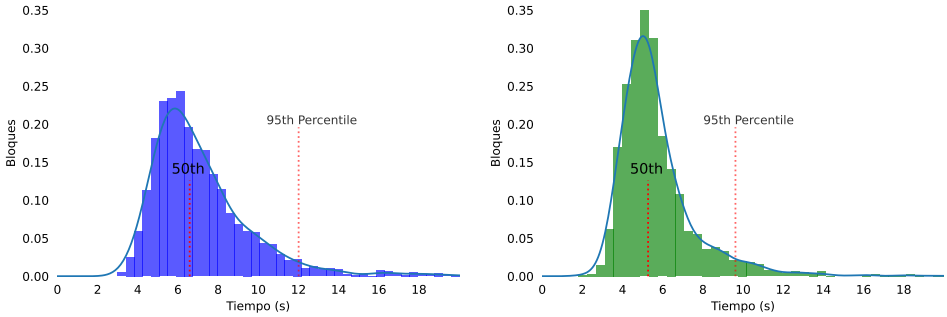
cual es algo razonable. Y en la figura 4.2(b), podemos ver un comportamiento similar en este aspecto.

Siendo que los tiempos de propagación mínimos se enfocan principalmente en el *delay* que existe entre dos clientes al momento de descubrirse un nuevo bloque y observando que los experimentos se comportaban como esperábamos y en ninguno surgió información relevante, se decidió no profundizar en un análisis esta medida.

Con respecto al tiempo máximo, pasó algo parecido. Representa el valor del último cliente que recibe el bloque minado. Esto sí depende de la posición en la red del cliente que descubre un nuevo bloque. Si el cliente es central, la información del nuevo bloque se esparce rápidamente a toda la red. En cambio, si es un cliente que se encuentra en la periferia del grafo, tiene que recorrer un camino más largo hasta llegar a su peer más lejano. Estos valores tenían una dispersión muy grande, porque por ejemplo, producto de un *fork* un cliente puede tardar en recibir un bloque el tiempo que lleve la resolución de dicho *fork*. En las figuras 4.2(e) y 4.2(f) como es de esperarse, manejan distribuciones muy similares con tiempos de aceptación mayores a los de descubrimiento.

Otro dato que pudimos verificar rápidamente, es que el tiempo de descubrimiento y el de aceptación de bloques, están positivamente relacionados. Tanto con la red FIBRE encendida como cuando está apagada, con distribuciones similares, mantienen la misma relación los tiempos de descubrimiento con los de aceptación. Este escenario se repite en todos los experimentos y es razonable dado que su diferencia proviene de lo que tarda en llegar la información del *header* y la del bloque completo. Si la red es más lenta o más rápida, va a afectar proporcionalmente tanto al tiempo de descubrimiento como al de aceptación.

Producto de esto último, decidimos enfocar nuestro análisis únicamente sobre los resultados de los tiempos promedio de descubrimiento de los bloques. De las seis medidas que planteamos originalmente para los experimentos, la más relevante fue ésta.



(a) Desactivada (b) Activada

Figura 4.3: Tiempos de descubrimiento promedio con *target time* de 300 s y 16 *pools* de minería.

Para analizar los tiempos de descubrimiento promedio, para cada experimento generamos los histogramas que se muestran en la figura 4.3. Podemos ver, por separado, el tiempo de descubrimiento promedio con la red de retransmisión `FIBRE` desactivada en color azul y con la red `FIBRE` activada en color verde. Con estos histogramas pudimos ver que todos los experimentos tenían una distribución similar de sus datos, y marcamos explícitamente el 50 y 95 percentíl para poder compararlos. De esta manera, vemos que en todos los experimentos no solo se mantiene la misma distribución, si no que los datos se concentran en valores más pequeños cuando la red FIBRE se encuentra activada. En el ejemplo de la figura 4.3(a), podemos ver en alrededor de 7 s y 12 s el 50-percentil y 95-percentil de la muestra, con la red de retransmisión `FIBRE` desactivada. Al activar `FIBRE` en la figura 4.3(b), vemos cómo estos valores se reducen a alrededor de 5 s y 10 s segundos respectivamente.

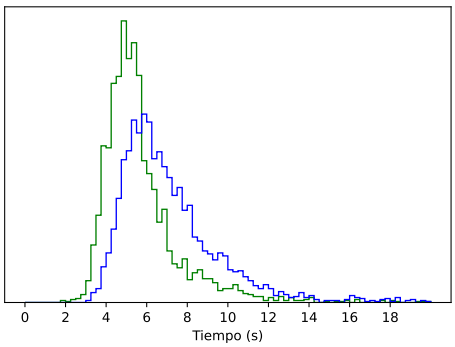


Figura 4.4: Tiempo de descubrimiento promedio con *target time* de 300 s y 16 *pools* de minería solapados para evidenciar las diferencias.

En el gráfico de la figura 4.4 superponemos los dos histogramas para el mismo experimento.

Esto nos permite ver rápidamente cómo la muestra se concentra en valores sensiblemente menores con la red de retransmisión de `FIBRE` activada.

La tabla 4.3 presenta una vista general de los resultados de todos los experimentos. Los datos correspondientes al histograma con la red de retransmisión `FIBRE` activada presentan una mejora sensible. Esto implica que aquellos experimentos que tiene la red de retransmisión `FIBRE` activada, consiguen una mejora considerable en sus tiempos de propagación.

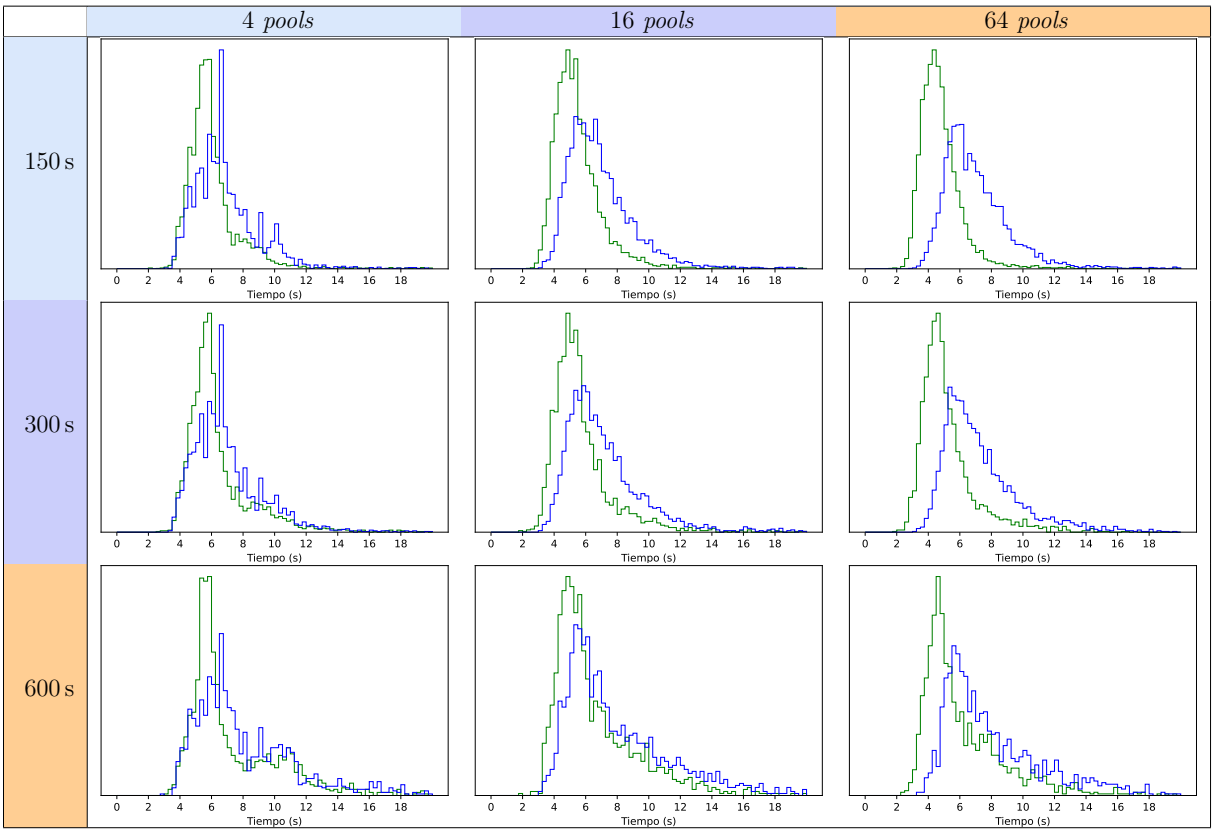


Tabla 4.3: Comparación de los tiempos de descubrimiento promedio para cada experimento realizado

Con estas figuras, es evidente el impacto positivo de la red de retransmisión `FIBRE` en el protocolo. Parece mostrar un impacto mayor cuando el poder de cómputo está más distribuido, como por ejemplo en los experimentos con 64 *pools* de minería.

Se corroboran nuestras hipótesis de como el agregado de la red `FIBRE` disminuye los tiempos de propagación de la información dentro del sistema. En la próxima sección vamos a analizar cómo esto impacta directamente en el rendimiento del protocolo.

4.2.3 *Hashing Power* desperdiciado

Esta magnitud comunica en mayor detalle la degradación de la red respecto al uso de recursos. No es lo mismo que un cliente con el 50 % del poder de cómputo utilice sus recursos para extender la cadena principal sobre un bloque que termina siendo descartado, que uno que solo tiene el 1 %. En el primer caso el 50 % del poder de cómputo del sistema fue desperdiciado, mientras que en el segundo solo el 1 %.

En el tabla 4.4, se reflejan los valores del porcentaje de *Hashing Power* desperdiciado desagregado por el tipo de cliente para todos los experimentos realizados.

Hay una tendencia clara que muestra como la utilización de la red de retransmisión aporta beneficios considerables al sistema disminuyendo el poder de cómputo desperdiciado por sus participantes dentro del sistema. Salvo para el caso del experimento con 600 segundos de target y 4 *pools* de minería que muestra un leve aumento del *Hashing Power* desperdiciado cuando la red `FIBRE` se encuentra activada. En promedio tiene una mejora de un 27 %.

Algo que también podemos destacar es como disminuye globalmente el tiempo de cómputo desperdiciado en función del *target time* de la red. De esta forma vemos que con la red de retransmisión activada y un *target time* de la mitad del original del protocolo (300 segundos), tenemos valores de *hashing power* desperdiciados similares al modelo estándar. Siguiendo esta lógica, se podría disminuir el *target time* de la red a la mitad manteniendo los valores de funcionamientos actuales de la red e incrementar el *throughput* disminuyendo el tiempo de confirmación de las transacciones.

Habiendo visto que los tiempos de propagación mejoraron con la presencia de la red de retransmisión, estos resultados son esperables. También podemos inferir que en aquellos experimentos en los cuales hubo aumentos en la cantidad de bloques *stale* con la red de retransmisión `FIBRE` activada, como se vio en el tabla 4.1, los mismos no tuvieron una gran participación dentro de la red y la mayor parte del poder de cómputo siempre estuvo dedicado a trabajar sobre la cadena principal. Con estos casos podemos apreciar como esta magnitud es mucho más sensible y marca mejor el rendimiento del protocolo.

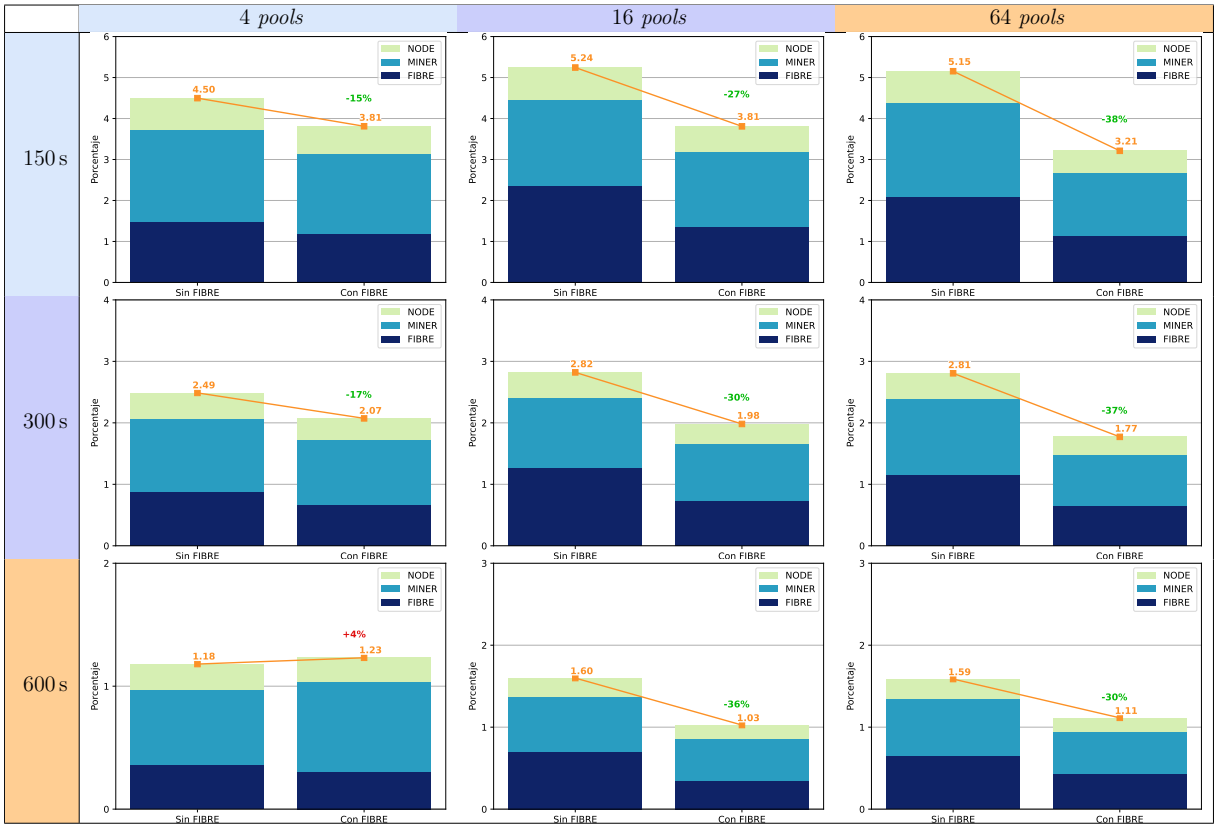


Tabla 4.4: Porcentaje de *Hashing Power* desperdiciado promedio variando la cantidad de *pools* de minería.

En la tabla 4.5, podemos ver el detalle del porcentaje de *Hashing Power* desperdiciado para cada tipo de cliente de todos los experimentos con su respectivo desvío estándar. En este caso los desvíos estándar son muy bajos en comparación con los que se producían al analizar la cantidad de *forks* del sistema. Por lo tanto podemos concluir que esta es una medida que no solo representa mucho mejor el rendimiento del sistema, si no que es una medida mucho más estable para los experimentos de naturaleza estocástica de este sistema distribuido.

4.2.4 Centralidad

Para analizar la centralidad de la red vamos a hacer hincapié en comparar los distintos grupos de clientes que definimos en un principio. Recordemos los tres grupos. El primero son pools *de minería* `FIBRE` que tiene un poder de cómputo elevado y en aquellos experimentos que esté activada la red de retransmisión `FIBRE`, van a estar interconectados por la

Hashing Power		4 *pools*		16 *pools*		64 *pools*	
		Desactivada	Activada	Desactivada	Activada	Desactivada	Activada
Target 150 s	Comun	0.781 (0.008)	0.678 (0.011)	0.785 (0.004)	0.629 (0.007)	0.784 (0.005)	0.544 (0.002)
	Minero	2.254 (0.089)	1.949 (0.087)	2.116 (0.049)	1.822 (0.072)	2.283 (0.048)	1.528 (0.018)
	Fibre	1.462 (0.056)	1.184 (0.058)	2.343 (0.035)	1.358 (0.140)	2.087 (0.049)	1.140 (0.017)
	Total	4.497 (0.352)	3.812 (0.389)	5.245 (0.213)	3.810 (0.515)	5.153 (0.261)	3.211 (0.091)
Target 300 s	Comun	0.421 (0.003)	0.359 (0.002)	0.421 (0.003)	0.324 (0.001)	0.424 (0.003)	0.297 (0.001)
	Minero	1.195 (0.014)	1.057 (0.026)	1.138 (0.026)	0.935 (0.011)	1.242 (0.024)	0.836 (0.006)
	Fibre	0.871 (0.055)	0.657 (0.032)	1.262 (0.029)	0.723 (0.031)	1.140 (0.023)	0.639 (0.005)
	Total	2.487 (0.128)	2.072 (0.132)	2.821 (0.146)	1.982 (0.100)	2.807 (0.127)	1.771 (0.026)
Target 600 s	Comun	0.214 (0.000)	0.205 (0.001)	0.240 (0.002)	0.175 (0.001)	0.244 (0.001)	0.179 (0.001)
	Minero	0.613 (0.006)	0.724 (0.023)	0.661 (0.027)	0.509 (0.010)	0.704 (0.008)	0.516 (0.006)
	Fibre	0.353 (0.002)	0.302 (0.003)	0.698 (0.013)	0.342 (0.000)	0.639 (0.007)	0.419 (0.006)
	Total	1.180 (0.014)	1.230 (0.046)	1.600 (0.105)	1.026 (0.020)	1.587 (0.042)	1.113 (0.032)

Tabla 4.5: *Hashing Power* desperdiciado promedio en los distintos experimentos realizados

misma. Luego los pools *de minería comunes* que tiene el mismo poder de cómputo que los anteriores, pero en todos los experimentos, la única red que van a utilizar para compartir sus bloques es la *P2P* original del protocolo. Y por último los *mineros comunes*, que son la mayor cantidad de clientes en proporción y tienen un *Hashing Power* casi despreciable individualmente, pero entre todos son relevantes y solo se comunican por la red original.

Tabla 4.6: Distribución de bloques dentro de la cadena principal para cada experimento realizado

Analizar la cantidad de bloques que terminaron dentro de la cadena principal, no aportó información contundente como podemos ver en el tabla 4.6. Vemos que los experimentos siguieron más o menos la distribución configurada de antemano. En términos generales, el 86 % de bloques fueron minados por los *pools* de minería con ese porcentaje de *Hashing Power* distribuido, y el restante 14 % por los mineros comunes.

Anteriormente, en la sección 4.2.2, pudimos ver cómo los tiempos de propagación mejoraban considerablemente con la red de retransmisión `FIBRE` activada. En el tabla 4.7 se puede apreciar con mayor detalle, cómo aquellos clientes que forman parte de la red ***`FIBRE`***, en todos los casos obtienen beneficios considerablemente mejores que el resto de los clientes. En todos los casos, vemos como los *boxplots* se encuentra con valores por debajo de los otros dos grupos de clientes. Lo cual muestra que siguen una distribución con tiempos de propagación menores que el resto.

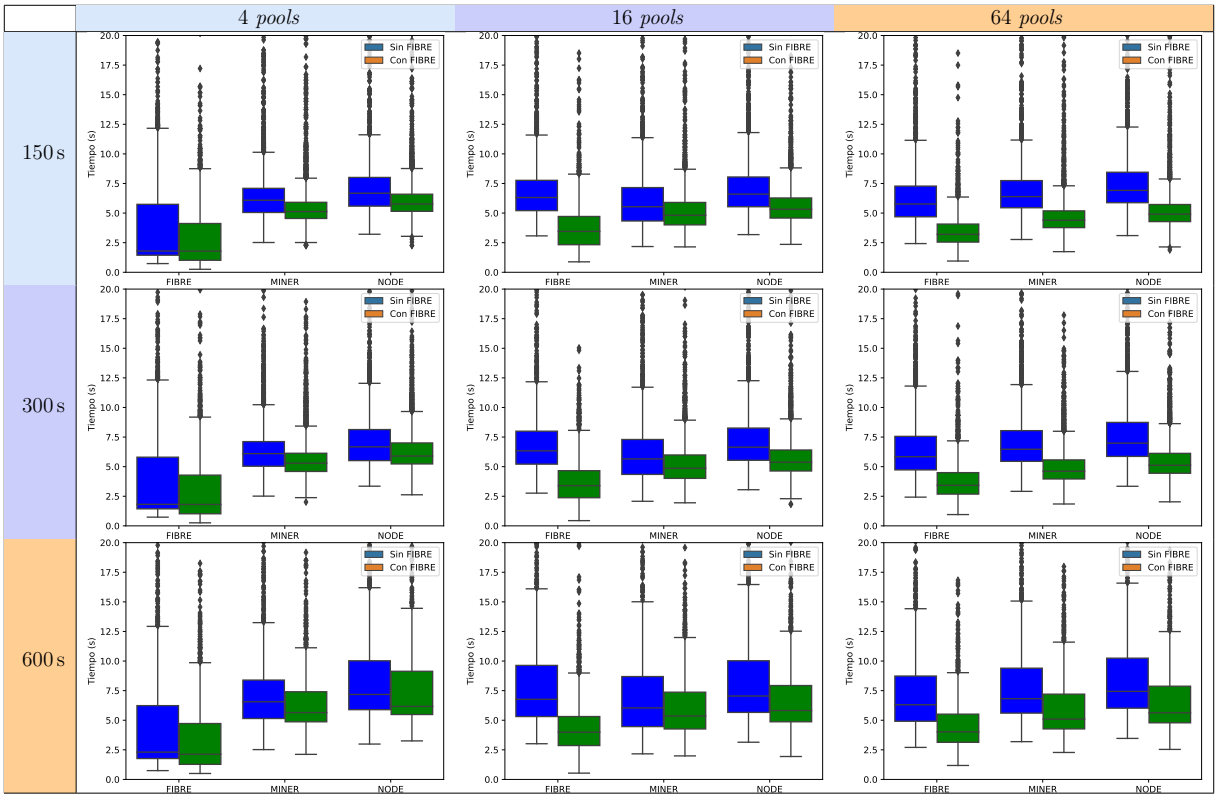


Tabla 4.7: Tiempo de descubrimiento promedio desagregado por tipo de cliente para cada experimento realizado

Hay un detalle en los experimentos con 4 *pools* de minería que nos llamó la atención.

Vemos como en todos los casos los datos de los *pools* de minería que están conectados a la red de retransmisión `FIBRE` tiene valores sensiblemente mejores que el resto, incluso con la red `FIBRE` desactivada. También, podemos ver en estos clientes que activar la red de retransmisión `FIBRE`, solo genera pequeñas diferencias en relación al resto de los clientes y experimentos. Atribuimos este fenómeno a la posición de estos mineros dentro de la topología de la red, y vimos que son clientes centrales en el sistema y por lo tanto, son clientes que están rápidamente al alcance de todo el resto, y el impacto de la red `FIBRE` en este caso no se puede apreciar con mucha claridad.

En cambio, vemos como cuando aumentamos la cantidad de *pools* de minería, todos se encuentran mejor distribuidos. Con la red de retransmisión `FIBRE` desactivada, la distribución de las muestras son mucho más parecidas Y podemos apreciar como en todos los escenarios los clientes conectados a la red de retransmisión `FIBRE` consiguen un desempeño mucho mejor al resto cuando dicha red está activada.

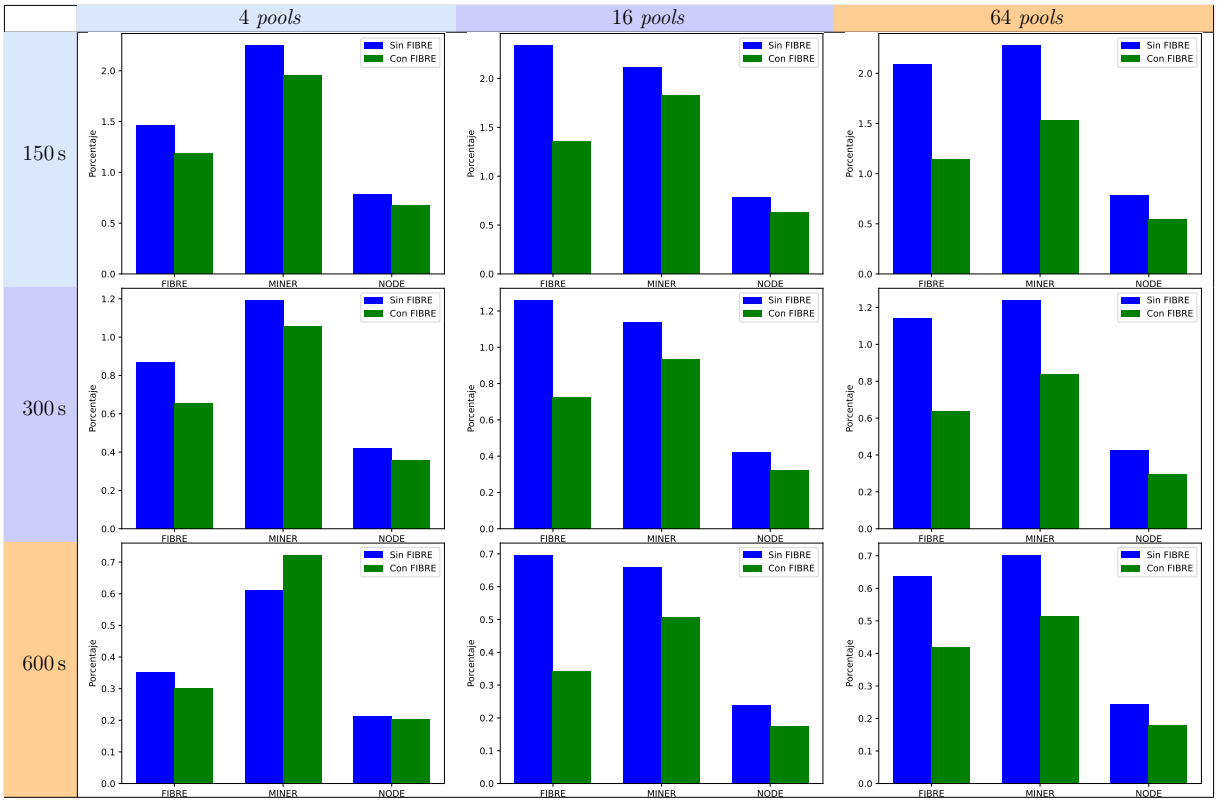


Tabla 4.8: Porcentaje del *Hashing Power* desperdiciado desagregado por tipo de cliente para cada experimento realizado

También habíamos visto como el *Hashing Power* desperdiciado se veía mejorado producto de la red de retransmisión `FIBRE`.

En el tabla 4.8 hay evidencias de cómo en todos los experimentos con más de cuatro *pools* de minería, aquellos conectados a la red de retransmisión `FIBRE` se ven considerablemente más beneficiados que sus contraparte.

Podemos ver en consonancia con los gráficos de los tiempos de propagación, cómo se ve una mejora global del sistema y una mejora sensiblemente mayor para aquellos *pools* de minería que tienen la posibilidad de recibir información por la red de retransmisión `FIBRE`.

Capítulo 5

Conclusiones y trabajo futuro

5.1 Conclusiones generales

En esta tesis queríamos analizar el impacto de las redes de retransmisión de bloques en un sistema de pagos electrónicos distribuido como lo es Bitcoin. Intentamos capturar distintos fenómenos que nos permitan cuantificar como afecta la presencia de las mismas al protocolo. Propusimos distintos enfoques para verificar si estas tecnologías en pos de mejorar algunos aspectos del sistema, atentan contra la naturaleza distribuida del protocolo.

Decidimos utilizar tecnologías de emulación para desarrollar distintos experimentos que nos permitan recolectar la información necesaria para dilucidar algunas dudas que nos planteamos a lo largo de esta experiencia. Para tal motivo, tuvimos que modificar el cliente de Bitcoin para recolectar información relevante a las métricas que decidimos monitorear y también conseguir que la utilización de recursos por parte de cada cliente sea la mínima indispensable sin alterar las características generales del protocolo. Tuvimos el desafío de pensar distintos modelos que nos permitan capturar en parte el funcionamiento del protocolo de un sistema distribuido con escala global con las limitaciones de recursos utilizados para la emulación. Las simplificaciones que se hicieron tanto en la distribución del *Hashing Power*, el modelo de red de retransmisión como en las latencias entre los clientes, fueron

necesarios para obtener un escenario en el cual se pudieron realizar comparaciones certeras.

Los experimentos de este trabajo consistieron en una red de Bitcoin de 200 clientes y una red de retransmisión la cual se iba a activar y desactivar para poder comparar entre dos instancias idénticas. Actualmente, el protocolo de Bitcoin establece un tiempo entre bloques esperado de 600 segundos, y usualmente cada bloque contiene en promedio 2000 transacciones. Hicimos un experimentos utilizando esta configuración, y también reduciendo el tiempo entre bloques esperado a la mitad y un cuarto. Por otro lado, modificamos la cantidad de clientes conectados a la red de retransmisión para verificar el impacto de la misma en función de su tamaño. Planteamos tres escenarios los cuales tenían 2 (1 %), 8 (4 %) y 32 (16 %) *pools* de mineros conectados a la red de *FIBRE* cuando correspondía.

Analizando los resultados de los experimentos pudimos destacar comportamientos tanto positivos como negativos del protocolo con el agregado de la red de retransmisión.

Comenzamos pensando que la cantidad de *forks* en la red era un muy buen indicador para catalogar el buen funcionamiento del sistema. Intuíamos que a menor cantidad de *forks*, mejor se comportaba el sistema en general. Esto fue relativamente cierto, dado que verificamos que el valor aislado de la cantidad de *forks* no representa en detalle la actividad del sistema y su eficiencia. Hubo experimentos en los cuales la cantidad de *forks* aumentaba, pero el poder de cómputo desperdiciado por el sistema era considerablemente menor. Pudimos concluir que un *fork* en la red es tan importante como su diseminación a lo largo de la misma. Al ser el minado de bloques un proceso estocástico, la aparición de *forks* dentro del sistema distribuido solo puede reducirse en función de los tiempos de propagación de la red.

Con respecto a los tiempos de propagación, verificamos la correlación que existe entre los tiempos de descubrimiento y aceptación de bloques. No pudimos percibir ningún fenómeno particular que haga interesante algún tipo de distinción entre estas dos medidas, por lo cual hicimos hincapié en el análisis del tiempo de descubrimiento promedio para todos los experimentos. Los resultados mostraron categóricamente que la red de retransmisión

consigue mejorar los tiempos de transmisión de los bloques dentro de la red, lo cual produjo grandes beneficios para el sistema.

Por lo tanto el análisis del *Hashing Power* desperdiciado puede explicarse casi exclusivamente por este fenómeno. Pudimos ver como esta métrica que tiene en consideración todo instante en cual el sistema utiliza recursos de sus clientes para lograr no solo validar transacciones dentro del mismo, si no converger a una decisión unánime y distribuida de cual es la cadena principal, como se vio beneficiado por la presencia de la red `FIBRE`. Llegamos a proponer que podrían conseguirse modificaciones del protocolo que mantengan los valores actuales de funcionamiento, mejorando algunos atributos que están en discusión por parte de la comunidad.

Por otra parte, cuando vimos los resultados desglosados en función de los distintos grupos que habíamos definido para los experimentos, pudimos detectar un claro beneficio para aquellos que participan dentro de la red de retransmisión. Esto demuestra cierta tendencia a la centralización del protocolo. Incluso asumiendo que todos los participantes son honestos y nadie tratara de explotar maliciosamente el protocolo, aquellos que no se conecten a las redes de transmisión siempre van a terminar indirectamente relegados.

Los desarrolladores de `FIBRE` alientan a que aquel que lo necesite pueda armar su propia red de retransmisión. Con una inversión relativamente pequeña, con empresas de `IaaS` alrededor del mundo pueden montar distintos clientes y generar una red de retransmisión que acerque puntos distantes del planeta.

Las redes de retransmisión no se ofrecen como una alternativa a la red original del protocolo. Todo lo contrario, no sería recomendable delegar esto a cualquier tercero que no fuera de confianza. Si alguien controla la forma en que se transmite la información del sistema, puede explotar esto de distintas formas. Podría decidir deliberadamente retrasar ciertos bloques que no le son beneficiosos por algún motivo con la idea de intentar imponer ciertos bloques.

La conclusión final es que la redes de retransmisión de bloques aportan grandes beneficios

al protocolo en general y a todos los integrantes del sistema. Los beneficios particulares para aquellos que se encuentran dentro del alcance de la red de retransmisión no producen ningún tipo de degradación del sistema. Pudimos ver en los experimentos que la cantidad de bloques generados por cada grupo de clientes, mantenía la distribución original del poder de cómputo. Lo que sí es bastante interesante, es entender que aquellos clientes que podían intercambiar información por medio de la red de retransmisión `FIBRE` conseguían reducir considerablemente el poder de cómputo que desperdiciaban. De esta manera, estos clientes son más eficientes y por lo tanto cualquier inversión que realicen va a ser más rentable que su contraparte.

5.2 Trabajo futuro

En esta sección se proponen algunas líneas de investigación que pueden servir como continuación a este trabajo. Las mismas están enfocadas a seguir desarrollando el modelo presentado y a analizar el funcionamiento de Bitcoin bajo escenarios que quedaron fuera del alcance de la tesis.

Mejoras en el modelo de Bitcoin Intentar modelar un sistema que se acerque mucho más a la realidad. Encontrar un modelo que represente mejor tanto la distribución del *hashing power* que existe en el sistema actualmente, como las latencias entre los clientes de todo el mundo. Siendo Bitcoin un sistema que funciona a escala global, crear un escenario en el cual la distribución de los clientes no sea simétrica y pueda representar mejor a la realidad con distintas latencias que puedan percibir distintos clientes en distintos países es un desafío muy interesante.

Mejoras en las redes de retransmisión El modelo propuesto en esta tesis fue una simplificación necesaria por la escala de los experimentos. Se podría experimentar con la saturación de la red o la perdida de paquetes `TCP` que hagan relevante la utilización de los protocolos de comunicación internos de la red `FIBRE` por medio de comunicaciones `UDP`.

Extendiendo la idea de realizar experimentos que emulen el sistema a escala global, se puede armar una red de retransmisión más interesante y experimentar con su topología y la ubicación de los nodos en un escenario que simule distintos países.

Tamaño del modelo Experimentos con una mayor cantidad de clientes permiten tener un modelo más cercano al real. En esta tesis, el tiempo nos jugó en contra para realizar algunos análisis producto de la simplificaciones que tuvimos que realizar. Aumentando considerablemente el tamaño del modelo, se pueden diseñar tanto distintos escenarios como medidas a considerar.

Alcance de la emulación En el recorrido de esta tesis, tuvimos un largo camino para entender si los resultados de los experimentos que estábamos realizando no estaban sesgado por algún factor inherente al desarrollo del mismo y no efectivamente lo que queríamos medir. Analizar los límites de la emulación, es interesante para poder estimar los recursos necesarios para poder escalar los experimentos. También tener mejores métricas de control, servirían para poder aumentar la confianza en los resultados obtenidos a lo largo de distintos experimentos.

Bibliografía

[AB02] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. 74:47–97, January 2002.

[Cor] Matt Corallo. Bip 152 - compact blocks. https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki. Accessed: 8 de agosto de 2022.

[CRS+] Andrew Clifford, Peter R. Rizun, Andrea Suisani, Andrew Stone, and Peter Tschipper. Towards massive on-chain scaling: Presenting our block propagation results with xthin. https://medium.com/@peter_r/towards-massive-with-xthin-da54e55dc0e4#.kk1znlg1v. 8 de agosto de 2022.

[DC19] Nicolás De Carli. Sobre los límites del tiempo entre bloques en bitcoin. Master's thesis, Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, 2019.

[DPSHJ14] Joan Antoni Donet, Cristina Pérez-Sola, and Jordi Herrera-Joancomartí. The bitcoin P2P network. In *Financial Cryptography and Data Security*, Lecture Notes in Computer Science, pages 87–102, Berlin, Heidelberg, 2014.

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

of the 2017 ACM on Management of Data, SIGMOD '17, pages 1085–1100, New York, NY, USA, 2017.

[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.

[GKL16] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol with chains of variable difficulty. Cryptology ePrint Archive, Report 2016/1048, 2016. https://eprint.iacr.org/2016/1048.

[GM18] Maximiliano Geier and Esteban Mocskos. Sherlockfog: Finding opportunities for MPI applications in fog and edge computing. In Esteban Mocskos and Sergio Nesmachnow, editors, *High Performance Computing*, pages 185–199, Cham, 2018. Springer International Publishing.

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

[MJ15] Andrew Miller and Rob Jansen. Shadow-bitcoin: Scalable simulation via direct execution of multi-threaded applications. In *of the 8th USENIX on Cyber Security Experimentation and Test*, CSET'15, pages 7–7, Berkeley, CA, USA, 2015. USENIX Association.

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

[OABS19] Kai Otsuki, Yusuke Aoki, Ryohei Banno, and Kazuyuki Shudo. Effects of a simple relay network on the bitcoin network. In *of the Asian Internet Engineering*, pages 41 – 46, 2019.

[Van16] Marco Vanotti. Un avance hacia entornos de gran escala para experimentos con criptomonedas. Master's thesis, Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, 2016.

[Vn17] Silvio Vileriño. Estudio de los límites de generación de bloques en blockchain. Master's thesis, Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, 2017.