UNIVERSIDAD DE BUENOS AIRES

Facultad de Ciencias Exactas y Naturales

Departamento de Computación

Un estudio del rendimiento del minado Bitcoin en escenarios de Merged Mining

Tesis presentada para optar al título de Licenciado de la
Universidad de Buenos Aires en el Área Ciencias de la Computación

Martín Gonzalo Medina

Director: Dr. Esteban Mocskos

Diciembre de 2021
Buenos Aires, Argentina

Un estudio del rendimiento del minado Bitcoin en escenarios de Merged Mining

Resumen

La tecnología blockchain se caracteriza por permitir almacenar información de forma segura, verificable y no repudiable sin requerir de la presencia de terceras partes para generar confianza. Hoy en día, el máximo exponente de esta tecnología son las criptomonedas, que se utilizan principalmente para la transferencia de valor, siendo Bitcoin la criptomoneda más utilizada. Existe un gran mercado potencial para otros usos que podrían beneficiarse de Blockchain y, en particular, de Bitcoin dada su gran aceptación actual. Sin embargo, no es posible utilizar esta ventaja directamente debido a diversas limitaciones que presenta.

El mecanismo de *merged mining* permite utilizar este poder de cómputo utilizado en Bitcoin no solo para validar sus propios datos sino para cualquier otra criptomoneda o incluso de otro tipo de blockchains. Con este mecanismo sería posible validar datos de blockchains que no cuenten con la cantidad de actores suficientes, ni con el volumen de cómputo mínimo para funcionar de forma distribuida en manera segura. Es decir, permitiría darle solidez a otras blockchains a partir de Bitcoin.

Este trabajo presenta una metodología y resultados para el estudio del rendimiento del minado Bitcoin en escenarios de *merged mining*. Para esto se exhibe un conjunto de nuevas mediciones, desarrolladas especialmente para este estudio, y casos de prueba que representan diferentes escenarios.

Los resultados obtenidos permiten concluir que no se detecta impacto alguno del *merged mining* en la minería Bitcoin. Gracias a lo anterior, es posible avizorar la factibilidad técnica de una solución de *merged mining* basada en Bitcoin.

A study of Bitcoin mining performance in Merged Mining scenarios

Abstract

Blockchain technology stores information in a secure, auditable and non repudiable way without the need of involving trusted third parties. The greatest exponent for said technology is cryptocurrency. Cryptocurrency is used primarily to transfer value, with Bitcoin being the most used. There is a huge potencial market for other use cases that could benefit from Blockchain, particularly from Bitcoin given the fact that is widely adopted. However, it is not possible to directly use that advangtage because of some limitations that arise.

The *merged mining* mechanism brings the possibility to use the computing power of Bitcoin to not only validate it's own data but also to verify data from any other cryptocurrency or Blockchains. Said mechanism allows to validate data from other blockchains that do not have enough participants or computing power to work securely in a decentralized fashion. That is to say, other blokchains would be more robust thanks to Bitcoin.

This work presents a methodology and results for an study about Bitcoin mining performance in merged mining scenarios. For that purpose, a new set of metrics is presented and developed. Also, test cases are designed and built to represent the different scenarios.

Results indicate that there is no perceptible impact generated by merged mining in Bitcoin mining. Thanks to the previous conclusion, technical solutions based on merged mining with Bitcoin are feasable.

Dedicado a mis padres, a mi familia y a mis amigos.

Índice general

Capítulo 1

Introducción

La **tecnología blockchain** permite almacenar información de forma segura, verificable y no repudiable; incluso entre múltiples entidades de forma descentralizada. En este último aspecto es donde gana mayor importancia al contar con actores distribuidos que pueden producir información sin requerir de la presencia de terceras partes para generar confianza.

El mayor exponente de la tecnología blockchain son las criptomonedas, que demostraron mundialmente el empuje que puede generar esta tecnología. **Bitcoin**, como la primera criptomoneda de uso masivo, emplea una gran cantidad de poder de cómputo en su red que se utiliza únicamente como parte del mecanismo de funcionamiento de esta criptomoneda.

El mecanismo de **merged mining** plantea utilizar este poder de cómputo para validar datos no solo de Bitcoin, sino de cualquier otra criptomoneda o incluso de otro tipo de blockchains. Con este mecanismo, sería posible validar datos de blockchains que no cuenten con la cantidad de actores suficientes, ni con el volumen de cómputo mínimo que le asegure un funcionamiento seguro.

Como ejemplo, se presenta el caso de una empresa que almacena sus datos de trazabilidad de lotes en una blockchain con el objeto de poder compartirlos con sus distribuidores de forma segura y transparente. De esa manera, la blockchain de la empresa podría aprovechar el *merged mining* para asegurar su cadena con el nivel de confianza y seguridad que cuenta Bitcoin. Lo anterior le evitaría tener que contar con grandes equipos para minar, construir y mantener una blockchain del tipo de Bitcoin.

Este trabajo presenta una metodología para el estudio del rendimiento del *merged mining* en escenarios realistas. Para esto se exhiben un conjunto de nuevas mediciones, desarrolladas especialmente para este estudio, y casos de prueba que representan los diferentes escenarios.

A continuación se desarrollan los principales conceptos sobre los que se apoya el trabajo.

Bitcoin brinda un mecanismo de transferencia de dinero entre dos partes, llamados también **peers**, sin la necesidad de que exista un tercero de confianza involucrado. La estructura que se utiliza para almacenar toda la información de las transacciones es global; un libro contable público de transacciones, llamada **blockchain** [3]. Esta blockchain se encuentra replicada y, cada 10 min, un nuevo bloque de transacciones es agregado al libro contable. Todas las transacciones realizadas desde el inicio del sistema están públicamente disponibles para los clientes que forman parte del sistema.

El **minado** de Bitcoin es el proceso por el cual las transacciones, como parte de un bloque, son confirmadas y luego agregadas a la blockchain. Este proceso computacionalmente intensivo que involucra la inversión parcial de funciones de hash criptográfico, como `SHA256`, es conocido como **prueba de trabajo** o **proof of work (PoW)**. La obtención de una prueba de trabajo es aleatoria y su costo depende del poder de cómputo de la máquina donde se realiza esta tarea. Por ejemplo, resolver la prueba de trabajo con el procesador en una máquina de escritorio, genera trabajo a un ritmo varios órdenes menor que hacerlo uti-

lizando una placa de video. Este proceso ha hecho aparecer y evolucionar equipamiento especializado de minería para la resolución del algoritmo de prueba de trabajo [8]. Estos dispositivos son llamados **mineros** o **hardware de minería** y se encargan de computar una y otra vez la función criptográfica `SHA256`. Cada intento produce un hash, y la performance de la máquina es medida en términos de *terahashes* por segundo (TH/s), lo cual es denominado como **hashrate** o **hashing power**.

El poder de cómputo determina la cantidad de intentos que se pueden realizar para encontrar una prueba de trabajo y esto define el tiempo que demanda hallar dicha prueba. Sin embargo, la red de Bitcoin productiva tiene bloques cada 10 min en promedio, es decir, se espera que cada 10 min se encuentre una solución para la prueba de trabajo. Este tiempo se denomina **target time** y el sistema lo mantiene estable a pesar de los cambios en el poder de cómputo que posse gracias al **algoritmo de ajuste de dificultad**.

Al momento de crear una nueva prueba de trabajo, el algoritmo de ajuste tiene en cuenta el tiempo en que los últimos bloques fueron generados. Para el caso en que el tiempo fuera menor a 10 min, la nueva prueba de trabajo contará con una dificultad mayor a la anterior. El objetivo de dicho ajuste es que demande más trabajo encontrar una solución. En consecuencia también demandará más tiempo, restableciendo así el *target time* deseado. El mismo razonamiento vale pare el caso de tiempo mayor a 10 min, donde la dificultad de la nueva prueba sería menor.

La dificultad actual de la red de Bitcoin hace que la probabilidad de que un solo minero resuelva el algoritmo de prueba de trabajo sea muy baja. Es por esto que los mineros trabajan en conjunto para aumentar sus chances de poder validar un bloque. Se denomina **pool de minería** a la unión de los mineros. Debido a la naturaleza aleatoria de la prueba de trabajo, cualquiera de los participantes del pool de minería tiene chance de encontrar la solución. El premio será luego repartido con el resto de los participantes del pool de minería.

El **software de pool de minería** es el componente que intercede entre los mineros y la red de bitcoin a través del nodo de Bitcoin; envía el trabajo a realizar a los mineros y valida las soluciones candidatas que estos proveen.

Merged mining es el proceso en el cual los mineros generan trabajo para dos o más blockchains, sin necesidad de realizar esfuerzo extra. En este escenario hay una cadena primaria y una secundaria.

Una blockchain que existe en paralelo a una **cadena primaria**, es lo que se conoce como **cadena secundaria** o **sidechain**. Dichas cadenas se caracterizan por tener una relación a nivel de datos, que permite vincular la información presente en una con la información en la otra y viceversa. En este trabajo, la cadena primaria es Bitcoin mientras que la secundaria es RSK [7].

En la figura 1.1 se muestra un diagrama con la interconexión de los distintos componentes que participan del minado; en un escenario de *merged mining*, con Bitcoin como cadena primaria y RSK como cadena secundaria. El nodo de Bitcoin intercambia mensajes con la red de Bitcoin, provee trabajo y recibe soluciones del software de pool de minería. El nodo de RSK tiene responsabilidades similares a excepción de que intercambia mensajes con la red de RSK. El hardware de minería resuelve la prueba de trabajo y el software de pool de minería coordina el proceso.

Como condición requerida para *merged mining*, los mineros necesitan modificar el código de su software de pool de minería siendo los cambios más relevantes los transcriptos a continuación:

- Adición de interacción con el cliente RSK.
- Agregado de información de *merged mining* al flujo de minado Bitcoin.
- Modificación del ritmo de generación de trabajo.
- Inclusión de validación de solución RSK.
- Generación de bloque solución RSK.

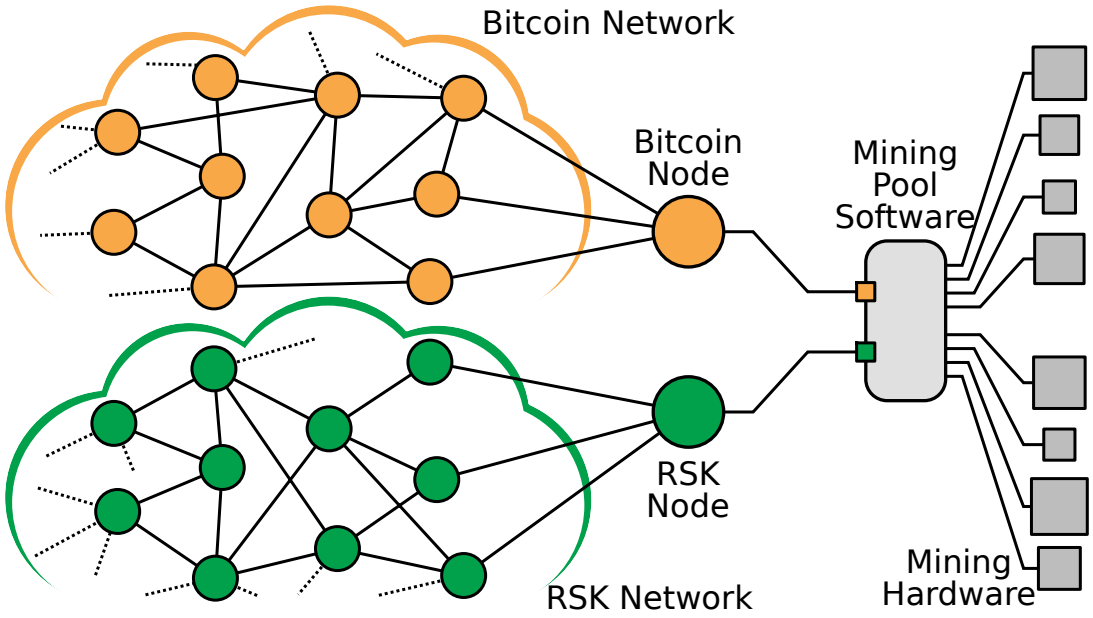


Figura 1.1: Topología de red para un escenario de *merged mining* con Bitcoin como cadena primaria y RSK como cadena secundaria. Además de los nodos de dichas cadenas se puede ver al software de pool de minería orquestando el proceso y al hardware de minería generando soluciones para la prueba de trabajo.

Merged mining puede beneficiar a todos los actores que participan del proceso. Bitcoin encuentra una forma segura de escalar a través de cadenas secundarias que realizan *merged mining*. Las cadenas secundarias se benefician de las bondades de Bitcoin, teniendo en este caso acceso a más hashing power para asegurar su funcionamiento. Los mineros pueden aumentar sus ingresos, haciendo el mismo trabajo y gastando los mismos recursos que si estuvieran minando solo la cadena principal.

El principal desafío del *merged mining* es permitir que las dos cadenas puedan ser minadas sin que existan efectos negativos sobre la cadena principal.

Para poder atacar dicho desafío, este trabajo presenta un estudio de la performance de *merged mining* al exponer un conjunto de nuevas mediciones. Estas mediciones se utilizan para medir la performance en sistemas con y sin *merged mining*, concentrándose en diferentes escenarios de exigencia de la vida real y otros aún peores.

A continuación se explicarán en detalle, los conceptos presentados anteriormente que requieren de un mayor grado de conocimiento para poder comprender el trabajo.

1.1 Criptomonedas, blockchain y minado

Este trabajo se concentra en la minería de criptomonedas y su rendimiento. Por lo anterior, resulta necesario poder comprender el proceso de minado Bitcoin con *merged mining* en detalle. Para dicho cometido hay que entender el funcionamiento de los componentes (figura 1.1) que hacen al mismo posible. En la presente sección se buscará explicar cómo funcionan tales componentes en detalle y al mismo tiempo introducir conceptos necesarios sobre criptomonedas, blockchain y minado.

En primer lugar, se describirá Bitcoin con el objeto de mostrar las estructuras esenciales de las criptomonedas; como son las transacciones, bloques y blockchain. También se mencionarán sus protocolos, como el consenso y el minado, que hacen posible al funcionamiento de dichas estructuras. Para concluir con Bitcoin, se exhibirán detalles sobre el funcionamiento de su infraestructura productiva así como otras nociones relevantes para entender este trabajo.

Luego se presentará la evolución de la minería Bitcoin, desde sus comienzos hasta la actualidad. Esto servirá para introducir múltiples conceptos relacionados a la misma. Entre ellos, se hará especial foco en los de hardware de minería, pool de minería y software de minería junto con una de sus implementaciones, `BTCPool`.

Después se expondrá *merged mining* a nivel conceptual y aplicado al caso de estudio de esta tesis, con Bitcoin y RSK como actores. Para poder comprender *merged mining* aplicado es que además se presentará RSK, haciendo énfasis en sus aportes como *sidechain* de Bitcoin y en el concepto de *contratos inteligentes* o *smart contracts*.

Finalmente, se mostrarán los lineamientos a seguir para poder implementar *merged mining* en un software de pool de minería y se exhibirá la implementación llevada a cabo en `BTCPool`.

1.1.1 Bitcoin

Es la primera criptomoneda creada en el año 2008 por una persona o grupo de personas bajo el pseudónimo de Satoshi Nakamoto [9]. En la actualidad existe una gran cantidad de criptomonedas siendo precisamente Bitcoin, la más importante de ellas.

El concepto de Bitcoin es amplio y puede ser abarcado desde distintas dimensiones.

En esta sección, se cubrirá su rol como método digital de transferencia de valor entre dos pares y su protocolo criptográfico; que, funcionando sobre una red descentralizada, hace posible dichas transferencias. Para poder entender cómo ocurren las transferencias, se introducirán además los detalles técnicos que hacen posible su funcionamiento.

Para que Bitcoin pueda funcionar como una criptomoneda, debe permitir que dos pares se transfieran valor entre sí de manera digital. En el contexto de Bitcoin, la unidad monetaria se la denomina **bitcoin** y es lo único que el protocolo que sostiene este sistema permite transferir.

Para que un par le pueda enviar bitcoins a otro, es necesario que el emisario firme criptográficamente un mensaje conocido como **transacción** en el que se incluye, entre otras cosas, el monto en bitcoin y el destino de dicho mensaje. El destino es un número que permite identificar, de manera única, al par que va a recibir el mensaje y, por ende, los bitcoin incluidos en la transacción. Una vez recibido el mensaje, el par receptor podrá hacer uso de sus bitcoins al enviarlos a otro par de la misma manera que ocurrió el intercambio recientemente explicado.

Este proceso se puede apreciar de manera esquemática en la figura 1.2. Allí, la transacción de la izquierda es primero verificada por su receptor (el dueño del par de claves pública-privadas 1). Luego, el mismo dueño decide transferir el valor recibido en la transacción de la izquierda al destinatario, identificado por su clave pública 2, y para esto firma la transacción del medio. Siguiendo los mismos pasos que utilizó el dueño de la clave pública 1, el dueño de la clave pública 2 valida y envía una nueva transacción (transacción de la derecha) al dueño de la clave pública 3.

Lo descripto hasta el momento no es más que un sistema de transferencia de valor, como existen ya muchos en la actualidad. Por ejemplo, una transferencia de dinero a través de la red bancaria, tiene un idéntico funcionamiento a nivel conceptual.

Lo que diferencia a Bitcoin de los métodos conocidos hasta el momento de su aparición, es que evita el **double spending** sin necesidad de recurrir a terceros de confianza. Tal como su nombre lo indica, el *double spending* es cuando el dinero puede ser gastado más de una vez gracias a que su condición digital facilita su falsificación o duplicación.

Retomando el ejemplo anterior y siguiendo la figura 1.2, el emisor podría utilizar la misma moneda para generar múltiples transacciones válidas y enviarlas a sus destinatarios. De ser aceptado, esto generaría un caso de *double spending* que, hasta la irrupción de Bitcoin, era impedido por un tercero de confianza, que evitaba la aparición de estos escenarios.

Sin un tercero de confianza, Bitcoin propone que todas las transacciones sean grabadas en un registro

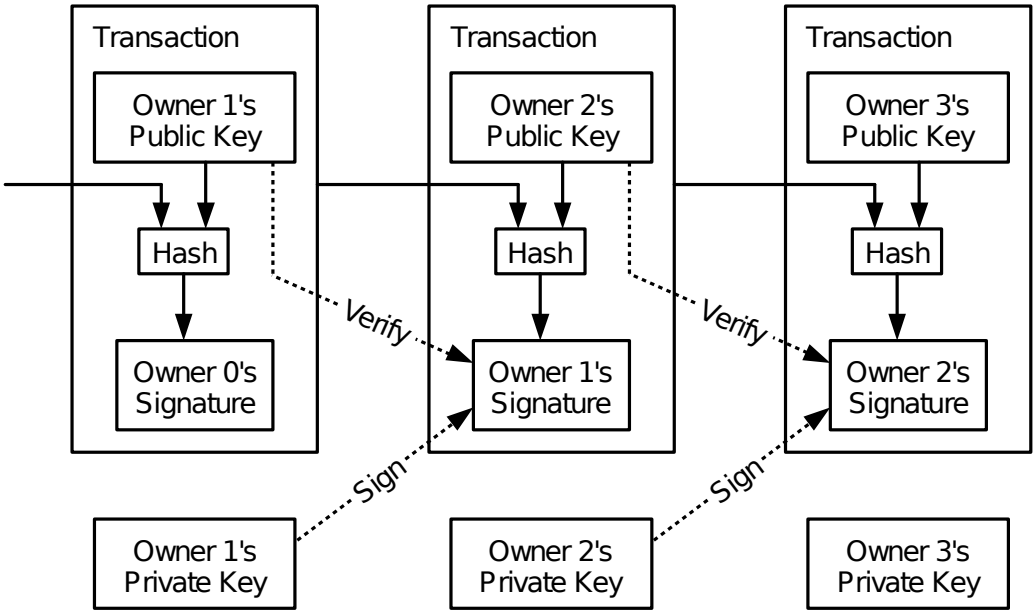


Figura 1.2: Esquema de transacciones digitales no supervisadas por un tercero de confianza. Cada transacción depende de la información de la anterior, que es luego validada y finalmente utilizada para generar una nueva. Fuente: [9]

público distribuido, la blockchain. En este esquema, cualquiera puede generar una transacción y la misma será válida siempre y cuando no exista otra igual en la blockchain. Esto evita el *double spending*.

Un ejemplo gráfico de dicha situación puede verse en la figura 1.3, donde se observan transacciones agrupadas en una estructura de datos llamada **bloque**. Cada bloque referencia al anterior y las transacciones ahí contenidas no pueden repetirse en bloques futuros.

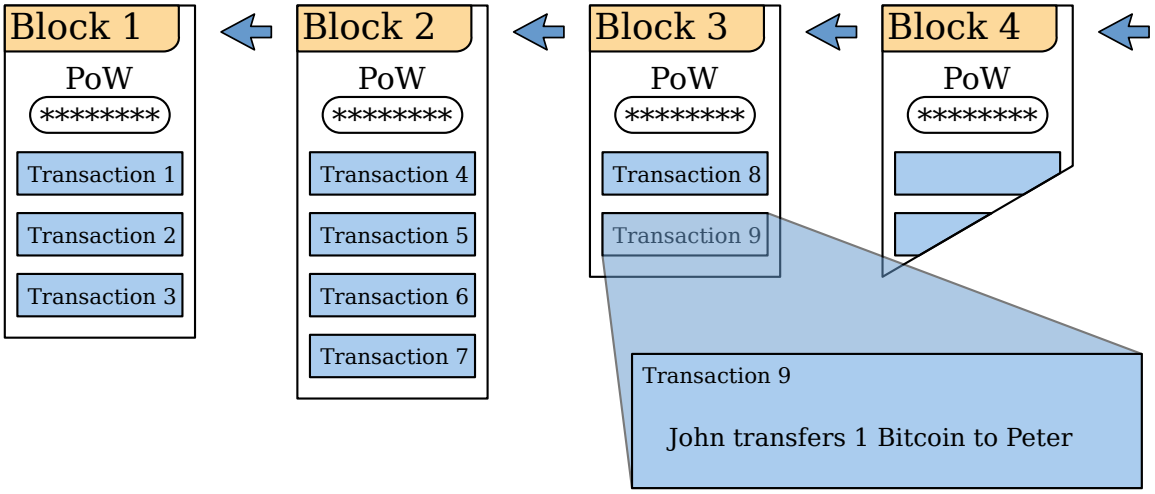


Figura 1.3: Esquema de envío de transacciones digitales utilizando blockchain.
En este caso el primer bloque es el 1, referenciado por el 2 y luego éste, es referenciado por el 3 y lo mismo ocurre con los siguientes. Las transacciones, como parte de cada bloque, no se repiten. Fuente: [11]

La transferencia de valor a través de Bitcoin, introducida previamente a nivel conceptual, requiere de distintos elementos para poder ser llevada a cabo en la práctica. A continuación se describen los mismos comenzando por las transacciones, siguiendo por su agrupamiento en una estructura de datos llamada

bloque y finalizando con su envío y registro, en una red descentralizada pública asegurada por el minado vía prueba de trabajo.

1.1.1.1 Transacciones

Tal como se ve en el trabajo original de Bitcoin [9], las transacciones son la estructura que contiene los datos que permiten la transferencia de valor entre dos pares en este sistema. Para que esto sea posible una transacción está formada por:

- **Inputs.** Es una lista y cada uno de sus elementos es llamado **input**. Un input contiene el hash de la transacción de la que fue *output*, el índice del *output* al que referencia, la clave pública del *output* y una firma digital utilizando la clave pública que se corresponde con el *output* referenciado.

- **Outputs.** Es una lista y cada uno de sus elementos es llamado **output**. Un *output* contiene un monto que representa la cantidad de bitcoins a enviar y un hash criptográfico de la clave pública del destinatario que se conoce como **dirección** o **address**.

El mecanismo de *inputs* y *outputs* hace que una transacción siempre referencie a la anterior y es poderoso ya que provee completa trazabilidad sobre el origen de los bitcoins recibidos. Tal como puede verse en la figura 1.4, se presenta una transacción con múltiples *inputs*. Se destacan los siguientes campos:

(A) Hash que identifica a la transacción.

(B) Lista de *inputs* de la transacción.

(C) Lista de *outputs* de la transacción.

(D) Hash de la transacción a la que hace referencia el *input*.

(E) Índice del *output* dentro de la lista de *outputs* de la transacción referenciada.

(F) Clave pública que se corresponde con el *output*.

(G) Firma digital con la clave privada correspondiente a la clave pública del *output*.

(H) Monto a transferir.

En particular, el hecho que un *input* referencie a un *output* anterior, es lo que brinda la posibilidad de validar su uso. Para verificar el mismo se debe calcular el hash de la clave pública y corroborar que coincide con el del el *output* usado. Luego solo resta validar la firma digital con dicha clave pública.

Los *outputs* pueden ser utilizados una sola vez lo cual lleva a que el monto completo de un *input* tenga que ser gastado en su totalidad, en la transacción que lo utiliza. Es por esto que existe la posibilidad de tener múltiples *outputs*. Por ejemplo, uno de ellos puede tener el monto y dirección del destinatario mientras que otro puede tener el cambio (cantidad que surge de la diferencia entre el valor del *output* a usar y el monto a enviar al destinatario) y la dirección de quien lo envía. Cualquier valor de *input* que no sea gastado es considerado como el costo a pagar para enviar dicha transacción y se denomina **transaction fee**.

El costo de envío de una transacción es variable y es responsabilidad del emisor establecerlo. Por lo general, una transacción con *fees* altas ocurre antes que otra con un valor mas bajo. Las razones que llevan a esto se tratarán en profundidad al momento de hablar de generación de bloques y minado.

En cuanto a la validez de una transacción, es necesario que se cumplan las siguientes condiciones:

- Sus *inputs* deben ser válidos de manera individual.

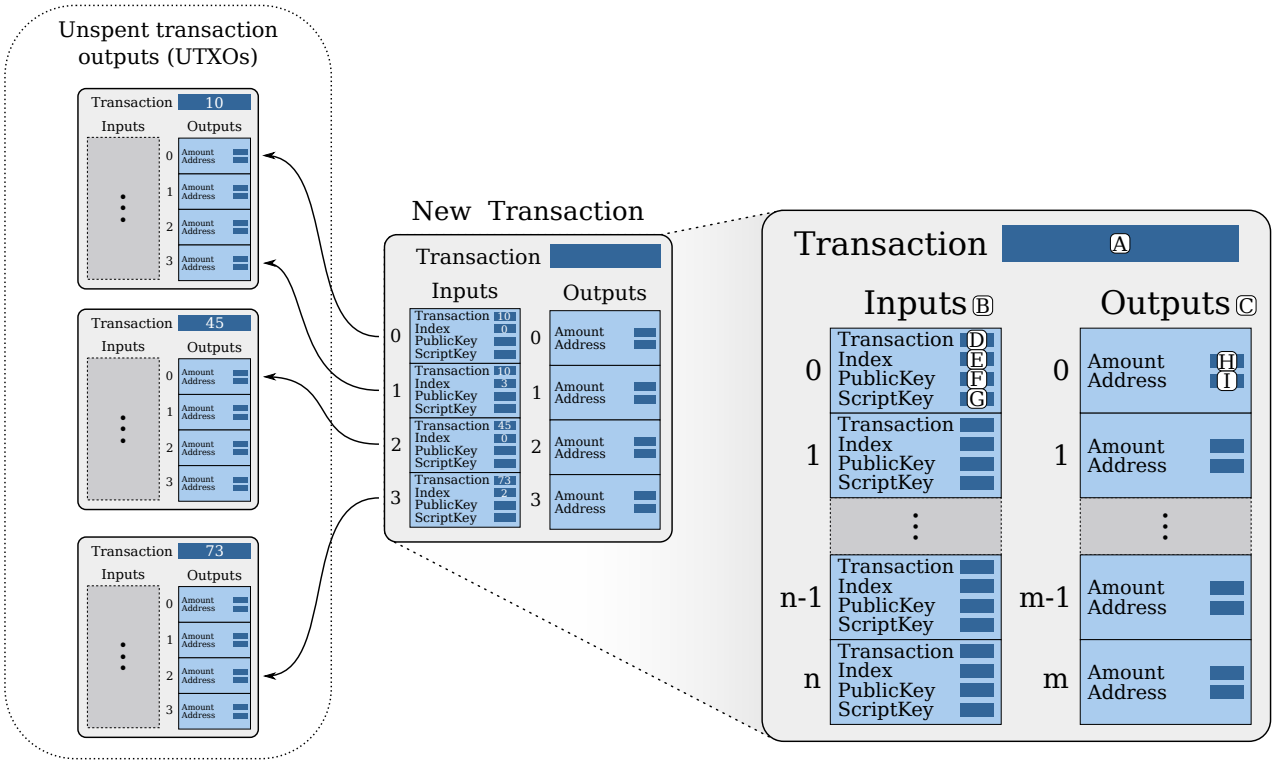


Figura 1.4: Transacción ejemplo de Bitcoin. La referencia con transacciones anteriores esta dada por sus *inputs*, mientras que transacciones futuras referenciarán a ésta a través de sus *outputs*. Fuente: [11]

- La suma de *outputs* referenciados por los *inputs* debe ser mayor o igual a la suma de los *outputs* de la transacción.

Las transacciones válidas generadas pero aún no incluidas en un bloque se denominan **transacciones pendientes** y se almacenan en una estructura interna del nodo de Bitcoin llamada **memory pool (mempool)**.

Es importante destacar que hasta el momento se hizo hincapié en transacciones que contienen una dirección y una firma digital utilizando la clave pública que se corresponde con el *output*. Éstas representan uno de los dos tipos de transacciones que Bitcoin permite realizar, siendo éste el más sencillo y llamado llama **Pay-to-PubkeyHash**.

El otro tipo, denominado **Pay-to-Script-Hash**, no tiene un formato predefinido para su contenido por lo que se puede incluir cualquier secuencia de datos dentro de lo que el lenguaje de *scripting* de Bitcoin permite. Este tipo de transacciones deposita mayor responsabilidad sobre el destinatario ya que deberá cumplir con los requisitos del *script* para poder gastar los bitcoins recibidos.

El lenguaje de *scripting* de Bitcoin es limitado en su expresividad; no tiene ciclos, lo cual hace que no sea Turing completo.

En esta sección se presentó en detalle la estructura de una transacción; cómo las mismas se referencian entre sí y los requisitos que deben cumplir para ser válidas. Cuando una transacción es válida, ya se encuentra lista para ser incluida en la blockchain. Esto ocurre cuando las transacciones, obtenidas del *mempool*, son agrupadas como parte de otra estructura de datos que las contiene y que se denomina bloque.

En la próxima sección se presentará en detalle a los bloques y su relevancia dentro de Bitcoin y como parte de la blockchain.

1.1.1.2 Bloques

Un **bloque** es una estructura de datos. La información presentada en esta sección sobre el mismo, así como detalles aún más técnicos, pueden ser encontrados en el libro *Mastering Bitcoin* [2].

La figura 1.5 presenta la estructura de un bloque, formada por dos partes:

- **Header.** El **header**, de solo 80 B, contiene metadatos que pueden ser interpretados en tres grandes grupos. Por un lado presenta una referencia al bloque anterior en forma de hash criptográfico conocida como **parent hash**. Por otro lado, presenta un conjunto de valores relacionados al minado siendo los mismos la **difficulty** o **dificultad**, el **nonce** y el **timestamp** o **tiempo del bloque**. Dichos valores serán explicados en detalle más adelante en el trabajo. Finalmente, el *header* contiene un resumen de las transacciones presentes en el bloque a través de un hash criptográfico denominado **merkle root**.

- **Body.** Posee una colección de todas las transacciones que fueron incluidas y es la parte que mayor cantidad de *bytes* ocupa del bloque.

  La primera transacción de la lista, denomina **coinbase**, se distingue de las demás al ser utilizada para cobrar las recompensas por generar el bloque.

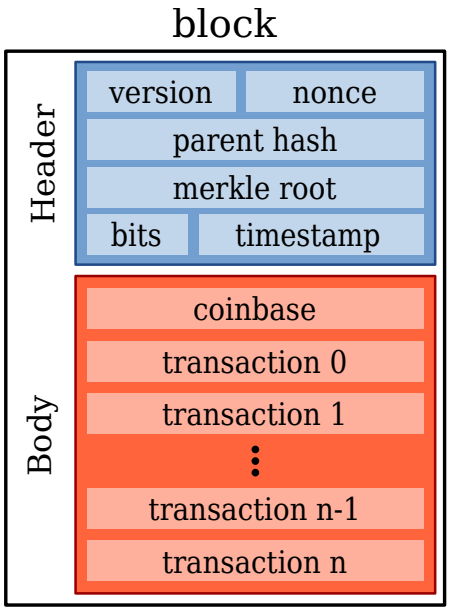


Figura 1.5: Bloque ejemplo de Bitcoin. Pueden observarse los elementos que lo componen, tanto para el *header* como el *body*. Fuente: [11]

El tamaño máximo de un bloque Bitcoin es de 1 MB. Sin embargo la cantidad de transacciones a incluir en el mismo puede variar por diferentes factores.

Uno de dichos factores es que las transacciones tienen un tamaño variable por lo que esto influye directamente en la cantidad de transacciones que ocuparan el espacio disponible en el bloque.

Otro de los factores es que no existe una única forma de incluir transacciones en un bloque por lo que pueden existir, por ejemplo, bloques sin transacciones así como bloques que estén completos.

La inclusión de transacciones es parte del proceso de minado de un bloque y éste será explicado más adelante.

Al igual que ocurría con las transacciones, un bloque debe cumplir con ciertas condiciones para que sea válido. Un bloque válido debe estar bien formado, es decir, sus campos deben tener los tipos y valores

correctos. Este trabajo no va a entrar en detalle en cómo se validan todos los campos del bloque pero sí hará foco en el caso del hash criptográfico *merkle root*.

El `merkle root` es un valor que resulta de computar un *merkle tree*. Un *merkle tree* es una estructura de datos de árbol binario que se caracteriza por tener hashes criptográficos como valores de sus hojas y para el cual cada nodo, que no es una hoja, calcula su valor como el hash criptográfico de sus hojas.

Para el caso de Bitcoin, al hablar de *merkle tree* se refiere a un árbol donde sus hojas están formadas por los hashes criptográficos de las transacciones del bloque. Computar dichos hashes a través de los nodos, da como resultado que la raíz del árbol sea un hash criptográfico que resume a todas las transacciones del bloque; este hash criptográfico es lo que se conoce como *merkle root*.

Un ejemplo de esto puede ser observado en la figura 1.6. En este caso, los hashes criptográficos de las transacciones desde la 0 a la 3 son ubicados en las hojas del árbol. Se computa el hash criptográfico para las transacciones 0 y 1 dando origen al `Hash01`. De igual manera se computa el hash criptográfico para las transacciones 2 y 3 dando lugar al `Hash23`. Finalmente se obtiene el *root hash* al computar el hash criptográfico entre `Hash01` y `Hash23` y se ubica el mismo, en el *header* del bloque Bitcoin.

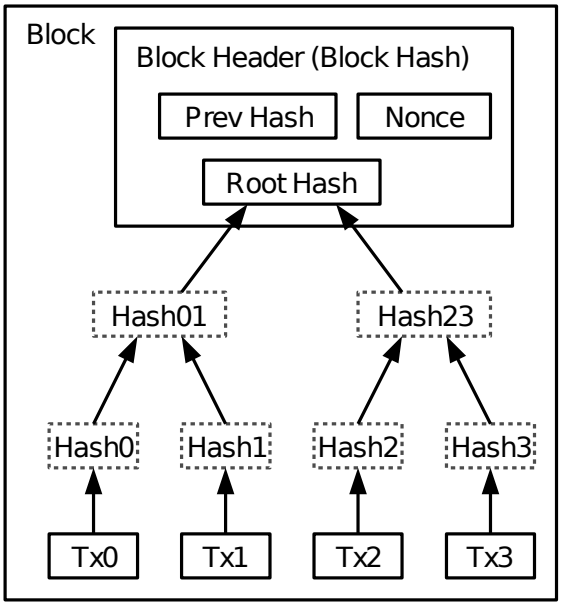


Figura 1.6: Construcción ejemplo de un *merkle tree*, también conocido como *root hash* para la red de Bitcoin. Los hashes criptográficos de las transacciones del bloque se ubican en las hojas. Luego se combinan de a pares hasta obtener el *root hash*, almacenado como campo del *header*. Fuente: [9]

Además de lo presentado hasta el momento, el bloque debe cumplir con la prueba de trabajo para ser válido. Para corroborar que la prueba de trabajo es correcta se realiza el hash criptográfico del encabezado del bloque y si el mismo es mayor que el valor del campo dificultad, entonces el mismo es válido.

Es importante destacar que hasta el momento se habló de dificultad del bloque por considerar más fácil la utilización de este concepto para explicar los demás conceptos necesarios. Sin embargo, el bloque tiene la dificultad en un formato conocido como **bits**, que es una forma compacta de representar el **target**. El *target* no es otra cosa que un valor inversamente proporcional a la dificultad.

En esta sección se introdujo el concepto de bloque cubriendo su estructura, la relación que un bloque tiene con los demás y las condiciones que se deben cumplir para que el mismo sea válido. También se introdujeron conceptos adicionales relevantes como el de *merkle tree* y otros relacionados a la minería, como dificultad y *target*.

En la sección siguiente se presentará a los bloques como parte de la blockchain y se explicará el mecanismo que permite que los mismos sean creados.

1.1.1.3 Blockchain

La **blockchain** es, cómo su nombre lo indica, una cadena de bloques que contienen transacciones.

El primer bloque de la blockchain se denomina génesis y tiene altura 0. El mismo fue incluido en el código fuente de Bitcoin y no minado dada su condición de no referenciar a ningún otro bloque, es decir, no tener bloque padre.

Tiene algunas otras curiosidades como un mensaje que indica la clara convicción de Satoshi Nakamoto de eliminar a los terceros de confianza de los sistemas de transferencia de valor.

El bloque de altura 1 se construye referenciando a su padre (el bloque 0) y lo mismo ocurre con todos los demás. De esta manera, a medida que pasa el tiempo, se van agregando más bloques que hace que crecer en tamaño a la blockchain.

A continuación y para finalizar con todo lo relevante a Bitcoin, se explicará el proceso de minado que permite agregar bloques a la blockchain, el consenso que en consecuencia debe establecerse y todo lo relevante a nivel infraestructura que hace posible el funcionamiento de Bitcoin en la vida real.

1.1.1.4 Minado

Es el proceso que permite que se agreguen bloques a la blockchain. En el caso de Bitcoin, los bloques se generan en promedio cada 10 min y este valor es conocido como **tiempo entre bloques** o **target time**.

La forma que tiene Bitcoin de lograr que este tiempo se respete es a través de la dificultad. Este es un valor que indica cuánto cómputo es necesario realizar para que un bloque sea aceptado.

Entonces, si un bloque fue minado en tiempo menor a 10 min, Bitcoin utilizando su algoritmo de ajuste de dificultad, modifica el valor de la misma para que el minado del próximo bloque requiera más poder de cómputo y por ende demore más tiempo. Es así que se tiende a restablecer el tiempo de 10 min de minado promedio. Lo mismo ocurre para un caso en que el tiempo de minado es mayor a 10 min solo que en esta ocasión se requiere una dificultad menor.

Lo recientemente explicado fue simplificado para poder exponer de manera más simple el concepto. Lo que ocurre en un entorno productivo, es que el ajuste de dificultad de Bitcoin ocurre cada 2015 bloques, representando esto un período de dos semanas aproximadamente.

El agregado de un bloque a la blockchain ocurre solo si el mismo tiene una prueba de trabajo o *proof of work (PoW)* [9] válida. La búsqueda de la prueba de trabajo consiste en encontrar el hash criptográfico del encabezado del bloque tal que sea mayor a la dificultad solicitada. La función de hash criptográfico utilizada por Bitcoin es doble `SHA256`.

Los denominados **mineros** son quienes realizan el cómputo de hashes criptográficos para validar bloques. Su trabajo consiste en ir colocando distintos valores aleatorios en el campo `nonce` del encabezado del bloque hasta encontrar uno que satisfaga la condición de la dificultad.

Por lo anteriormente mencionado, se desprende que el proceso de minado es aleatorio y las chances de encontrar una prueba de trabajo por parte de los mineros dependen directamente del poder de cómputo que los mismos tengan. Los equipos que se utilizan para minar son denominados **hardware de minería** y han ido evolucionando a lo largo del tiempo. Este tema será tratado en profundidad en la sección 1.1.2

Es importante destacar que el minado se caracteriza por requerir mucho trabajo para poder encontrar una solución de PoW pero por ser muy barato computacionalmente para verificar que una solución cumple con la prueba de trabajo. Además, dicha solución es solo útil para el bloque para el que fue obtenida ya que otros bloques tendrán otros valores en el *header*; por ejemplo la fecha, llevando esto a que el valor del hash

criptográfico se modifique y no sirva más. A la vez, el hash criptográfico del padre evita que se calcule la PoW de manera anticipada, es decir, cada minero podrá calcular la PoW para el bloque actual y contará con 10 min en promedio para lograrlo.

El minado de un bloque, es un proceso que permite la transferencia de valor entre los pares que lograron que sus transacciones fueran incluidas en el bloque. También cumple con una función importante para Bitcoin al ser el encargado de generar nuevos bitcoins. Está definido en el protocolo que cada bloque nuevo minado acuña bitcoins y que esos son destinados al minero que encontró la PoW.

La cantidad total de bitcoins a generar a través del minado es de 21 000 000 y este es el único mecanismo de creación de moneda que estableció Satoshi Nakamoto al diseñar el protocolo.

La distribución de dichos bitcoins generados ocurre con una función decreciente que comienza otorgando 50 bitcoins y luego cada 210 000 bloques divide, dicho valor a la mitad. En la actualidad la recompensa por minar un bloque es de 6,25 bitcoins [1]. El proceso de división de la recompensa se denomina **halving**.

De lo previamente expuesto se desprende que los mineros tienen un importante incentivo para realizar su trabajo, es decir los bitcoins *acuñados*, y éste no es el único ya que además se llevan las comisiones de las transacciones incluidas en el bloque. Para que esto sea posible, se utiliza un tipo especial de transacción que se ubica en el primer lugar del bloque. Dicha transacción, también conocida como **coinbase**, posee los *outputs* necesarios para que el minero luego de 100 bloques pueda gastar su recompensa.

La condición de espera de 100 bloques impuesta a los mineros se denomina **maturity** y surge del hecho de que la blockchain no es una cadena lineal sino que en realidad tiene forma de árbol y demora un tiempo hasta que un bloque tenga su lugar confirmado.

Hasta el momento, por razones de simplificar esta introducción, se explicó todo el proceso de minado de bloques suponiendo que un minero encuentra una solución y la misma es siempre aceptada haciendo que ese bloque quede en la blockchain. Sin embargo, en la realidad existen múltiples mineros buscando una solución de PoW para el mismo bloque al mismo tiempo y solo uno de ellos será el ganador, mientras que el resto de los bloques formarán cadenas paralelas.

Las reglas de cómo se elige a los bloques que terminan integrando la blockchain, son parte de lo que se conoce como **protocolo de consenso** [2] y serán explicados en la sección siguiente.

1.1.1.5 Consenso

Todo lo presentado hasta el momento funciona en la actualidad gracias a la existencia de código fuente que implementa el protocolo Bitcoin y que al ser compilado, da como resultado un programa ejecutable que se conoce como **nodo Bitcoin** o **cliente Bitcoin**. Cada uno de estos nodos conectados entre sí es lo que hace de Bitcoin una red descentralizada.

Dentro de esta red, y si bien todos los nodos son iniciados en base al mismo ejecutable, existen diferentes roles. Un nodo puede simplemente mantener una copia de la blockchain con el objeto de auditar la misma; otro nodo puede servir de punto de entrada para que un usuario envíe sus transacciones y otro puede servir para minar. Si bien estos roles son independientes también pueden ocurrir todos en un mismo cliente, en caso que quien lo esté administrando así lo desee.

Es así que la red de Bitcoin tiene múltiples nodos mineros que buscan minar bloques al mismo tiempo. Cuando uno de ellos encuentra una solución de PoW entonces propaga el bloque con la misma a toda la red y en ese momento es cuando, de ser ese bloque aceptado, se agrega el bloque nuevo a la blockchain.

Es aquí donde pueden darse distintos casos que se describen a continuación.

[1] `https://github.com/bitcoin/bitcoin/blob/b9ed2fd026926cafe38a1ee23d47eb891a94ddb1/src/validation.cpp#L1162`

Un caso cuando todos los nodos tienen la misma copia de la blockchain y se da que ninguno encontró un nuevo bloque al mismo tiempo que el minero, quién recién propagó su solución. En esta ocasión todos los nodos agregarán el nuevo bloque a su blockchain principal. La blockchain principal de un nodo es denominada **mainchain**.

Otro caso es cuando, debido a la naturaleza descentralizada de la red y a la competencia de minado existente entre los mineros, dos de ellos encuentran soluciones y las propagan en la red en momentos muy cercanos en el tiempo. Aquí es cuando los nodos de la red se van a encontrar con dos soluciones y deberán elegir una de ellas. El criterio que utiliza Bitcoin para determinar cuál nodo quedará en la blockchain, es el de cadena de mayor dificultad; esto quiere decir que el bloque de mayor dificultad será agregado a la *mainchain*. Si ambos tienen la misma dificultad el nodo agrega a la blockchain al bloque que recibió primero. El bloque que queda en la *mainchain* se denomina **best block**. A la vez, **stale block** es el bloque con PoW válida y misma altura que el *best block* pero que no forma parte de la *mainchain*.

También puede ocurrir que una parte de la red reciba un bloque nuevo A y otra reciba un bloque nuevo B, siendo ambos válidos. En este caso no habrá que desempatar y la red quedará particionada en dos. Una parte creerá que el *best block* es A mientras que la otra pensará que es B. A este escenario de red particionada se lo denomina **fork**. Si bien en este momento la parte de la red que tiene a A como *best block* no conoce la existencia de B, se considera que la cadena que tiene a B como *best block* es una *cadena paralela*.

Un ejemplo de algunos conceptos exhibidos en los casos previos puede verse esquematizado en la figura 1.7 y a través de lo siguiente:

(A) La blockchain es un árbol de bloques que cuenta con una cadena principal. En este caso la *mainchain* está representada en color azul. El bloque marcado con la letra A es el génesis.

Los demás bloques azules se han agregado posteriormente en el tiempo y en base al génesis. Todos ellos presentan una prueba de trabajo que fue encontrada antes, o con mayor dificultad, que sus competidores en gris.

(B) Los bloques que no pertenecen a la *mainchain*, en color gris, son *stale blocks* por presentar menor dificultad que los bloques azules de su misma altura.

(C) Un nuevo bloque naranja se agrega a continuación del que, hasta el momento era, el mejor bloque de la blockchain. En este caso el nuevo bloque pasa a formar parte de la *mainchain* como *best block*.

Las transacciones que contiene y agrega este bloque ocurren después de todas las transacciones que fueron incluidas en bloques anteriores.

Los casos citados anteriormente no son exhaustivos y representan solo algunos de todos los que pueden ocurrir en una red productiva como Bitcoin. Dichos casos fueron seleccionados por considerarse adecuados para introducir la serie de conceptos que se necesitaban presentar, dada su relevancia para este trabajo.

Por último hay que mencionar que los *forks* se solucionan a medida que los bloques se propagan por la red. Cuando un nodo cambia su *mainchain* se denomina **reorganización** y este proceso es algo que puede tomar algunos bloques en lograrse. En condiciones normales de funcionamiento de la red productiva de Bitcoin nunca hubo un *fork* de más de cuatro bloques.

Las reorganizaciones pueden producir que los bloques involucrados en las mismas no queden en la blockchain. Entonces, los usuarios que tienen transacciones en tales bloques están expuestos a la posibilidad de que estas no formen parte de la blockchain. Es por esto que en la red productiva de Bitcoin se suele esperar a que se minen seis bloques posteriores al bloque donde fue incluida la transacción, para considerarla confirmada.

Hasta el momento, se exhibió que el nodo Bitcoin tiene múltiples responsabilidades. Entre las más importantes se encuentran la de mantener una copia completa de la blockchain, de proveer una forma

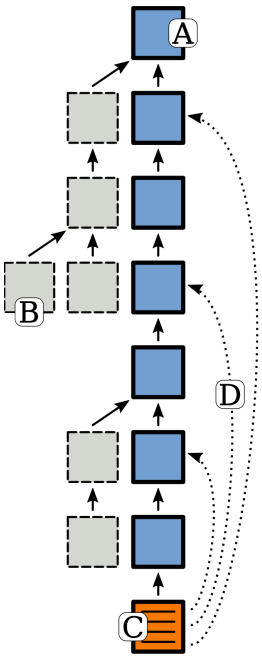


Figura 1.7: Esquema de una blockchain de ejemplo. En azul puede verse a la *mainchain*, con el bloque A como génesis. En gris se grafican los *stale blocks* mientras que en naranja se puede ver a un nuevo bloque que es agregado a la cadena principal.

para que los usuarios puedan realizar transacciones y que las mismas sean agregadas a un bloque a través del proceso de minado. Además, provee un protocolo de comunicación con los demás clientes de la red, permitiendo así la existencia de la red descentralizada Bitcoin, gracias a interfaces que el nodo de Bitcoin provee.

1.1.1.6 Infraestructura

La red productiva pública de Bitcoin es denominada **mainnet** y es la red en la que el nodo de Bitcoin es ejecutado por defecto.

Existen otras dos redes con las que se puede configurar el cliente. Una de ellas es la denominada **testnet** siendo la versión de *mainnet* pero destinada a pruebas. Tiene algunas diferencias para hacer que dichas pruebas sean más fáciles de llevar a cabo. Entre algunas de ellas se puede mencionar que la dificultad de minado se reinicia automáticamente si un bloque no fue encontrado en los últimos 20 min, que sus bitcoins no tienen ningún valor económico y que su blockchain es completamente independiente de la *mainnet*.

La otra red existente es la denomida **regtest**. En este caso es una red privada dado que no existen pares públicos a los que uno se puede conectar. Su objetivo es el de poder hacer pruebas, principalmente de desarrollo, en un entorno local controlado.

En lo presentando hasta el momento, quedó en evidencia que los nodos de la red de productiva de Bitcoin se comunican entre sí. Para poder hacerlo utilizan un protocolo binario llamado **wire protocol**.

El mismo consta de distintos mensajes que permiten iniciar una conexión con un par, verificar el estado del par, solicitar información como por ejemplo *headers* de bloques y propagar información como por ejemplo bloques nuevos, entre otros. Si bien dicho protocolo no será involucrado de manera directa en los estudios a presentar en este trabajo, es parte fundamental del funcionamiento del nodo y por eso es relevante conocer su existencia.

Mientras que el *wire protocol* se limita únicamente a la comunicación entre nodos, el nodo Bitcoin ofrece

una interfaz de tipo RPC para la comunicación con sus usuarios.

Dicha interfaz es la que permite que cualquiera con acceso de red al nodo Bitcoin, pueda interactuar con ella. La misma ofrece métodos para obtener y enviar todo tipo de datos sobre la red de Bitcoin como por ejemplo transacciones y bloques. Una posible categorización de dichos métodos muestra que existen métodos relacionados a la blockchain, de control, de red, de minería y de manejo de transacciones. Los métodos relevantes de la interfaz RPC para este trabajo pertenecen a la categoría minería y son los siguientes:

- **getBlockTemplate.** Provee una propuesta de bloque para utilizar en el proceso de minado. Dentro de los valores que devuelve al ser consultado, los más relevantes son una lista de transacciones, la dificultad con la que debe cumplir el bloque a ser minado y el hash criptográfico del *best block*.

- **submitBlock.** Acepta un bloque en formato binario, verifica su validez y, de ser válido, lo propaga a la red.

En un principio de esta introducción se mostró a los actores que llevan a cabo el minado. Sin embargo hasta el momento se trató al rol del minado en Bitcoin de manera conceptual y sin describir técnicamente a quienes están detrás de dicho rol.

La reciente presentación de los métodos del nodo Bitcoin relacionados al minado, permite que en la próxima sección se profundice en la explicación del proceso de minado y sus distintos actores.

1.1.2 Minería

En la sección 1.1.1.4 se introdujo el marco teórico necesario para entender el funcionamiento del minado como mecanismo de generación de bloques en Bitcoin.

En la presente sección se hará foco en cómo el minado ha funcionado en la red productiva de Bitcoin a lo largo de su historia.

En los comienzos, solo Satoshi Nakamoto minaba los bloques Bitcoin con su computadora personal. Este tipo de minado donde el encargado de generar los bloques es el CPU de la computadora se conoce como **CPU mining**.

El proceso de minado implica ir probando con distintos valores aleatorios en el campo `nonce` del *header* del bloque, hasta encontrar un hash criptográfico SHA256 del *header* del bloque cuyo valor sea mayor que la dificultad. La cantidad de hashes criptográficos que puede producir un CPU se mide en hashes por segundo. En la actualidad un CPU moderno produce en el orden de los *megahashes* por segundo (MH/s).

La naturaleza descentralizada diseñada por Nakamoto no tardó en surgir en Bitcoin ya que, a medida que el tiempo fue pasando, más personas se fueron sumando a minar. En consecuencia, las recompensas por generar bloques ya no fueron todas para Satoshi Nakamoto sino que se dividieron entre él y los nuevos participantes. Fue así cómo se originó la primera versión de minado descentralizado de la red de Bitcoin. Esto llevó también al comienzo de la adopción de Bitcoin ya que todo aquel que quisiera minar debía tener un nodo de la red propio.

En este punto de la historia, el cómputo de los hashes criptográficos para encontrar una PoW válida era realizado dentro del nodo de Bitcoin. Además, las recompensas por minar un bloque eran de 50 bitcoins con un valor del bitcoin de solo centavos de dólar estadounidense. Esto indica que quienes minaban en ese momento eran en su mayoría personas que querían probar la tecnología y no buscaban realizar un negocio con el minado.

A medida que el valor del bitcoin fue aumentando de los centavos de dólar estadounidense a los miles de dólares estadounidense el minado fue evolucionado. Fue así como frente a mayores incentivos económicos, surgió la búsqueda de mayor poder de cómputo, también llamado *hashing power*, para poder procesar más hashes criptográficos SHA256 y tener así una mayor chance de minar un bloque. Dicha demanda

de poder de cómputo fue satisfecha por placas de video (*Grapihcs Processing Unit* o *GPU*) dando origen a una nueva era de minado llamada **GPU mining**.

Las placas de video, con una potencia del orden de los *gigahashes* por segundo (GH/s), rápidamente dejaron obsoleto al *CPU mining* llevando a ese tipo de mineros a tener que esperar semanas para poder minar un bloque. Dicha situación solo empeoraría a lo largo del tiempo con la suma de más y más mineros que utilicen GPU. Sin embargo, con el objeto de mantener el minado con CPU vigente y la mayor cantidad de actores posibles en la red (mayor descentralización), es que surge el concepto de **minado cooperativo** o **pool mining**. Este concepto es claramente opuesto al minado que se venía desarrollando hasta el momento llamado **solo mining**. En *solo mining* cada minero era responsable de buscar una solución para la PoW.

El minado cooperativo implicó un cambio de paradigma para el minado ya que técnicamente surgió un nuevo actor en el proceso, el **software de pool de minería**. Hasta el momento el minado era llevado a cabo dentro del nodo de Bitcoin o, en el caso del minado con GPU, con programas de software especializados que interactuaban con el nodo de Bitcoin. El minado cooperativo consiste en unir el poder de cómputo de muchos mineros para buscar todos juntos soluciones de PoW. El responsable de orquestar dicha unión es el software de pool de minería, que se ubica entre el nodo de bitcoin y quienes saben resolver la prueba de trabajo.

Con la aparación del software de pool de minería se dejó de minar con CPU dentro del nodo Bitcoin para pasar a hacerlo con un programa específico llamado **CPU miner**.

La arquitectura de minado, tal como se puede ver en la figura 1.1, termina contando con los siguientes componentes:

- **Nodo Bitcoin.** Provee trabajo a través del método `getBlockTemplate` y recibe bloques nuevos por medio del método `submitBlock`.

- **Software de pool de minería.** Responsable de solicitar el trabajo al nodo de Bitcoin, construir bloques y enviarlos a los mineros. Al recibir una solución de un minero debe enviar la misma al nodo Bitcoin. En caso que la solución sea aceptada por la red, las ganancias se reparten entre todos los mineros de manera proporcional al trabajo aportado.

  El administrador del software de pool de minería se queda con un pequeño porcentaje de dichas ganancias por el servicio que provee.

- **Mineros.** Máquinas responsables de calcular la prueba de trabajo, también denominados **hardware de minería**.

El minado siguió evolucionando y al igual que el *CPU mining* fue sustituido por el *GPU mining*; luego el *GPU mining* fue sustituido por el **ASIC mining**.

ASIC (*Application-Specific Integrated Circuit*) es una plataforma de cómputo especialmente diseñada para minar. Esto quiere decir que un minero `ASIC` no es una pieza de hardware multipropósito, como el `CPU` o una placa de video, sino que solo sabe hacer la función de calcular hashes criptográficos `SHA256`. El poder de cómputo de un ASIC minero es del orden de los *terahashes* por segundo (TH/s) y es la tecnología que se utiliza en la red productiva Bitcoin en la actualidad.

Hoy en día el minado se realiza con *pools* de minería que son empresas establecidas que se dedican a ese negocio mientras que el cálculo de hashes criptográficos se realiza con mineros de tipo `ASIC`.

Es importante destacar que los *pools* de minería de la actualidad son pocos actores (entre 10 y 20) que agrupan todo el poder de cómputo utilizado por la red Bitcoin, siendo esto algo que atenta contra la descentralización y uno de los puntos de mayor crítica sobre el funcionamiento de la red.

En esta sección se presentó un recorrido por la evolución de la minería a lo largo de la historia de Bitcoin. En dicho recorrido se fueron introduciendo diferentes conceptos como los tipos de minado (*CPU mining*, *GPU mining* y *ASIC mining*), el minado cooperativo y la arquitectura que se utiliza para minar en la actualidad.

Es fundamental para este trabajo no solo entender el concepto de minería cooperativa sino también sumergirse en los detalles técnicos de funcionamiento del software de *pool* de minería. Lo anterior se desarrollará con profundidad en la próxima sección.

1.1.2.1 Pool de minería

En el presente existen diferentes compañías que se dedican a ofrecer el servicio de *pool* de minería Bitcoin. Cada una de ellas presenta a nivel comercial, distintos incentivos para atraer a mineros a su plataforma. Una mayor cantidad de mineros, implica que el *pool* de minería tendrá un mayor poder de cómputo. Gracias a esto podrá minar una mayor cantidad de bloques y así obtener mayores ganancias.

La competencia en el mercado de los *pools* de minería es grande, lo cual genera movimientos de poder de cómputo de un *pool* a otro con frecuencia. Tales movimientos son posibles dado que, de cara a los mineros, los *pools* comparten la misma interfaz provocando que el cambio sea tan simple como modificar una dirección IP.

Si bien los *pools* exponen una interfaz común de cara a los mineros, la historia es diferente puertas adentro. Cada compañía de *pool* de minería posee una infraestructura propia y única para funcionar. Este trabajo se concentrará en el componente principal de dicha infraestructura, el **software de pool de minería**.

Esta tesis busca estudiar el rendimiento del proceso de minado. Por lo vertido anteriormente, se presentará en detalle el rol del software de pool de minería, que permite el minado de bloques Bitcoin utilizando el poder de cómputo de múltiples mineros. Es importante destacar que el software de *pool* de minería cumple además otros papeles, tales como llevar registro del trabajo que cada uno de los mineros realizó y pagarles periódicamente por el mismo.

El flujo de minado que realiza el software de *pool* de minería así como su interacción con los demás actores involucrados en el minado se detallan seguidamente:

1. **Obtención del trabajo.** El software de *pool* de minería necesita contar con información para poder construir un bloque. Dicho bloque, llamado **bloque candidato**, se caracteriza por no tener un valor de prueba de trabajo, es decir, el campo `nonce` aún no fue completado.

   Para lograr armar el bloque candidato, el software de *pool* interactúa con el nodo Bitcoin a través del método `getBlockTemplate`. Como resultado de la interacción, se obtienen:

   - Una lista con las transacciones válidas pero aún no incluidas en ningún bloque, llamadas **transacciones pendientes**.
   - La dificultad que el bloque candidato deberá satisfacer para ser aceptado por la red de Bitcoin.
   - Otros valores que completarán el bloque candidato.

2. **Construcción del bloque candidato.** Utilizando las transacciones pendientes, se selecciona la mayor cantidad posible, tal de maximizar las fees a ganar en caso de que el bloque sea aceptado.

   Luego, las transacciones seleccionadas, son incluidas en el *body* del bloque. Como parte del proceso anterior se construye también la *coinbase*, incluyendo en la misma, la dirección de pago del *pool*.

   Teniendo el *body* del bloque listo se calcula el `merkle root` y junto con los demás valores del *header* del bloque, se tiene el bloque candidato.

3. **Interacción con mineros.** El proceso de conexión e intercambio de mensajes con los mineros se rige por un protocolo llamado **Stratum**.

   A los fines de la explicación actual, es suficiente con saber que el *pool* cuenta con un canal de comunicación con los mineros. A través de dicho canal, envía el trabajo a los mineros para que busquen una prueba de trabajo válida. Los mineros reciben los campos del encabezado del bloque necesarios para calcular la prueba de trabajo y poder compararla con la dificultad, que también es enviada por *pool*.

   Es importante destacar que para este caso, la dificultad utilizada no es la dificultad de Bitcoin, sino una que se llama **dificultad de pool** o *pool difficulty*. Dicha dificultad es varios órdenes inferior a la de la red de Bitcoin. Su objeto es lograr que los mineros envíen trabajo de manera periódica al *pool*. Algo que no ocurriría si se usara la dificultad de la red de Bitcoin, dado que ésta es muy elevada para que un minero pueda encontrar varias soluciones nuevas por minuto.

   Gracias a la naturaleza aleatoria del proceso de búsqueda de prueba de trabajo, ocurre que una solución que satisfaga la *pool difficulty*, conocida como **share**, puede también satisfacer la dificultad de la red de Bitcoin. El *pool* es el encargado entonces, de comparar las *shares* recibidas de sus mineros con la dificultad de la red de Bitcoin, para determinar si la cumple.

4. **Propagación de bloque candidato.** Un nuevo bloque válido de Bitcoin se encuentra cuando un minero envía al *pool* una *share*. Dicha *share* debe cumplir con la dificultad de la red de Bitcoin, o sea que es mayor a ella.

   En consecuencia, el *pool* ensambla el bloque y lo envía al nodo de Bitcoin al cual se encuentra conectado. Para lo anterior se utiliza en método `submitBlock` dela interfaz RPC del nodo de Bitcoin.

   Es aquí cuando el trabajo del *pool* está terminado y queda en manos de la red de Bitcoin, aceptar o rechazar el bloque enviado.

Existen diversas implementaciones técnicas para un software de *pool* de minería. Las mismas se diferencian principalmente por factores como el lenguaje de programación en el que fueron hechas, la arquitectura que presentan, la cantidad de clientes que soportan y el desempeño que ofrecen.

La afirmación precedente, se basa en implementaciones de software de *pool* de minería de código fuente abierto, es decir, aquellas a las que se puede acceder libremente. Este trabajo no tuvo acceso a implementaciones propietarias o de código cerrado, por lo cual las mismas no serán tenidas en cuenta.

Dentro del universo de implementaciones de código abierto, la más utilizada es **BTCPool** y por esa razón fue elegida para este trabajo. `BTCPool` es un software de *pool* de minería multi-blockchain que, según cómo sea configurado, puede funcionar tanto para Bitcoin como para otras criptomonedas. Se caracteriza por su arquitectura basada en microservicios, que se comunican a través de colas de mensajes.

A continuación se exhibirán detalles sobre el diseño, la implementación y el funcionamiento del software de *pool* de minería `BTCPool`. La mayor parte de la información a presentar sobre `BTCPool` fue adquirida por haber trabajado en el mismo de manera previa a esta tesis; también se obtuvieron datos de su documentación [2].

Los conceptos de `BTCPool` y software de *pool* minería se utilizarán indistintamente de ahora en más ya que este trabajo, solo se concentrará en dicho software.

1.1.2.2 BTCPool

El *pool* está formado por microservicios implementados en el lenguaje de programación `C++`. A la vez, la comunicación entre ellos se realiza usando `Apache Kakfa` como *framework* de colas de mensajes.

[2] `https://github.com/btccom/btcpool-ABANDONED#readme`

Sumado a lo anterior, el *pool* utiliza distintas bibliotecas para poder realizar su trabajo. Entre las más relevantes se encuentran aquellas que le permiten interactuar con las colas de mensajes, proveer una interfaz de comunicación de red a los mineros, dejar registro en archivos de *log* de los eventos ocurridos y utilizar las estructuras de datos de bitcoin.

Por último, hay que mencionar que se vale de una base de datos `MySQL` para guardar estadísticas de minado, tanto particulares de los mineros como generales de funcionamiento del *pool*.

Seguidamente, se listarán los microservicios que emplea el *pool* para hacer posible la minería Bitcoin. Para cada uno de ellos se incluirán los detalles de implementación y funcionamiento.

- ***Get block template maker (gbtmaker).*** Responsable de interactuar con el nodo Bitcoin a través del método `getBlockTemplate`.

  Dicha interacción ocurre utilizando el mecanismo de *polling* y la frecuencia del mismo, es configurable por medio de un archivo de configuración. Además, en dicho archivo se puede configurar todo lo necesario para conectarse al nodo Bitcoin.

  Al obtener la respuesta del llamado a `getBlockTemplate`, los datos de la misma son interpretados y volcados en estructuras de datos internas, para ser luego serializados en formato `JSON` e incluidos en una cola de mensajes llamada `RawGbt`.

- ***Job maker (jobmaker).*** Encargado de gestionar la información de minado que obtiene de consumir la cola de mensajes `RawGbt`. Para esto es utiliza una estructura de datos llamada **trabajo** o **job**.

  Cada trabajo, tiene asociados los datos obtenidos del nodo de Bitcoin para poder construir un bloque candidato y la dirección de pago de recompensa del *pool* (obtenida de archivo de configuración). Una vez que el trabajo es creado, se serializa en formato `JSON` y se incluye en una cola de mensajes llamada `BtcJob`.

  El rol del `jobmaker`, es el de manejar dichos trabajos y generarlos frente a alguna de las siguientes condiciones:

  - Cuando el *pool* recién inicia y no existe ningún trabajo previo.
  - Cada un intervalo de tiempo fijo, configurable por medio de un archivo de configuración. Dicha actualización periódica es útil, porque permite generar un trabajo nuevo y tener en cuenta la presencia de transacciones nuevas. Estas transacciones son aquellas que el nodo de Bitcoin, no tenía disponibles en el momento en que se construyó el trabajo anterior, puesto que las recibió tiempo después.

    El agregado de transacciones nuevas puede impactar en las ganancias del pool si las mismas ofrecen mayores comisiones que las existentes en el trabajo viejo.
  - Frente a la aparición de un bloque Bitcoin distinto al del trabajo más reciente. Este caso indica que ya sea el *pool* u otro actor de la red Bitcoin, minó un bloque y el mismo fue aceptado.

- ***Stratum server (sserver).*** Este microservicio es el responsable de comunicarse con los mineros. Para dicho cometido cuenta con:

  - La información necesaria para enviarle a los mineros; obtenida de escuchar la cola de mensajes `BtcJob`, procesar el `JSON` recibido y conservar el último trabajo leído en una estructura de datos interna en memoria.
  - Un servidor escuchando constantemente, en un puerto configurable, por conexiones entrantes. El servidor utiliza el protocolo de comunicación *Stratum* y es capaz de lidiar con múltiples mineros al mismo tiempo.

Una vez que un minero desea comenzar a trabajar en el *pool*, el mismo se conecta y recibe a cambio el trabajo a realizar. El *pool* envía los campos del encabezado del bloque Bitcoin para poder calcular el hash criptográfico y la dificultad del *pool*. La dificultad del *pool* es calculada por este microservicio, en base a un tiempo promedio de share que se quiera recibir, este tiempo es configurable por archivo de configuración.

Luego, el minero comienza su trabajo y envía *shares* al *pool* en el tiempo requerido.

El *pool* recibirá un conjunto de *shares*, de ellas habrá una gran mayoría que no será solución válida para el bloque de Bitcoin, ya que cumplirá con la *pool difficulty* y no con la dificultad de la red de Bitcoin. En este caso, el *pool* solo registrará el trabajo del minero dentro de sus estadísticas de minado, para luego poder pagarle su recompensa.

Dependiendo del *hashing power* del *pool*, cada cierto tiempo existirá una *share* que será mayor que la dificultad de Bitcoin, convirtiéndose ésta en solución para el bloque actual. Es aquí cuando el *pool* tomará la información de la *share*, junto a la del *header* del bloque, la serializará en formato `JSON` y la enviará a la cola de mensajes llamada `BtcSolvedShare`.

- ***Block maker (blkmaker).*** A cargo de enviar bloques válidos al nodo de Bitcoin, a través del método `submitBlock` de dicho nodo.

  Para construir el bloque válido, este microservicio lee mensajes de las colas de mensajes `RawGbt`, `BtcSolvedShare` y `BtcJob`.

  Con toda la información recabada, cruza datos para corroborar que efectivamente tiene que armar un bloque válido y, en caso afirmativo, armar el bloque y enviarlo a la red de Bitcoin.

En la figura 1.8, se puede ver en azul a cada uno de los microservicios presentados anteriormente, junto con sus interacciones con otros componentes. Además se muestran otros microservicios que `BTCPool` ofrece. Estos últimos no son necesarios para el funcionamiento básico de minado del software, por lo cual no serán explicados.

Cada uno de los microservicios, tiene flechas que apuntan al servicio de colas y muestran el nombre de la cola de mensajes que utilizan. Por ejemplo, el *jobmaker* escucha de la cola de mensajes `RawGbt` y escribe en la cola `stratumJob` que para este trabajo fue renombrada a `BtcJob`.

Por último, se puede observar la interacción de los microservicios con agentes externos. Para el caso del *gbtmaker* y del *blkmaker*, ambos consumen vía `RPC` al nodo de Bitcoin, representado en blanco. El *sserver* interactúa con los mineros, graficados con una nube de color amarillo.

Es importante destacar que todos los componentes cuentan con lógica de manejo de errores y de registro de eventos.

La configuración tanto de las colas de mensajes como de cada uno de los microservicios entre sí, es independiente. Es más, el software de *pool* de minería no provee un método de instalación a través de ejecutables pre compilados, por lo que la compilación queda del lado del usuario.

En la próxima sección, se contarán detalles del hardware de minería a utilizar en este trabajo y de su interacción con el `BTCPool`.

1.1.2.3 Hardware de minería o minero

En la sección 1.1.1.4, se presentó la evolución de los equipos de minado y su rol en la historia de la minería Bitcoin. Para entender este trabajo es necesario entrar en detalle en el minado que se realiza con el procesador de una computadora, llamado `CPU mining`.

[3] `https://github.com/btccom/btcpool-ABANDONED#readme`

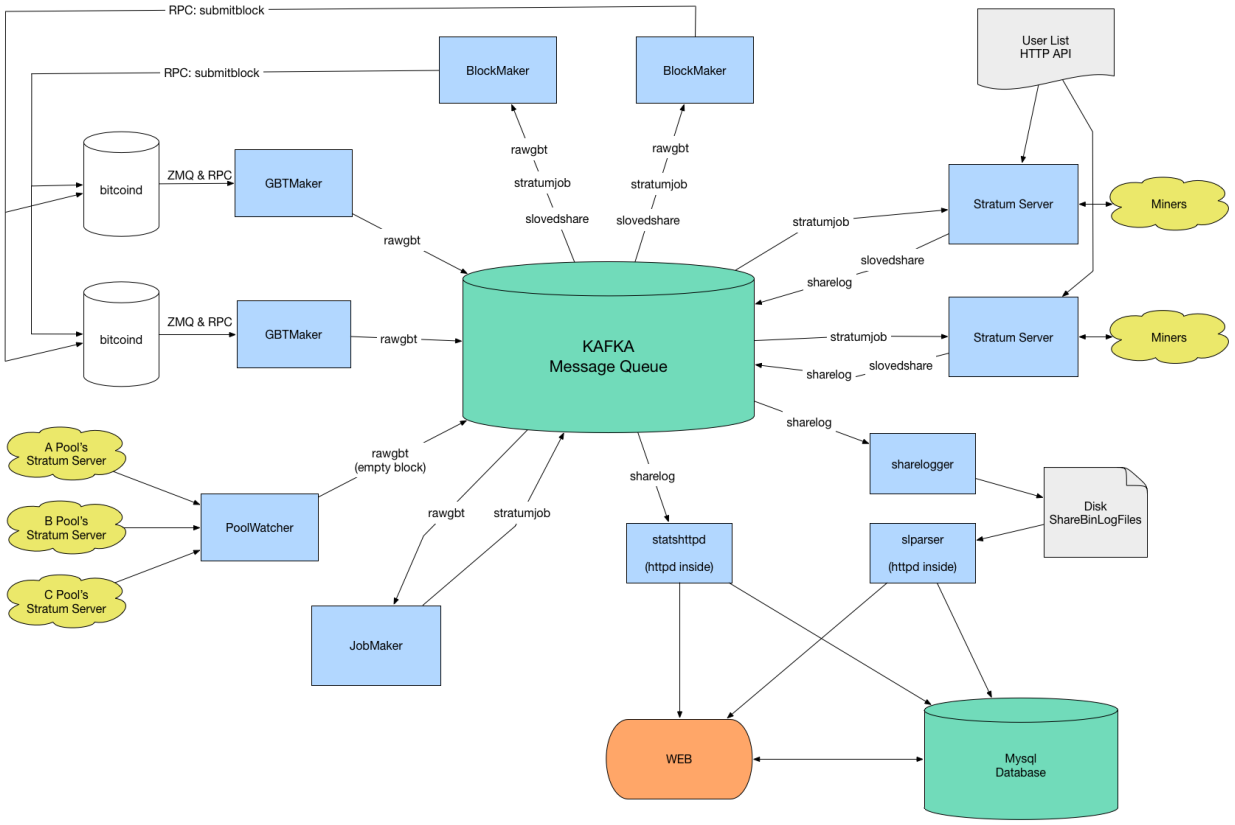


Figura 1.8: Arquitectura del software de *pool* de minería `BTCPool`. Fuente: Documentación `BTCPool` [3]

En los comienzos de Bitcoin, el componente de minado por CPU que se utilizaba para minar era parte del código fuente del nodo Bitcoin. Por lo anterior, esta acción resultaba tan sencilla, como configurar el nodo Bitcoin para que lo hiciera.

Con el avance en el poder de cómputo y el surgimiento de los *pools* de minería, se dejó de usar ese componente. A modo de reemplazo para quienes deseaban seguir minando con CPU, surgieron programas específicos para dicha tarea.

El primero y uno de los más populares, fue el llamado **CPU miner** que está implementado en el lenguaje de programación C y su función es la de resolver la prueba de trabajo de Bitcoin. También soporta el minado de otra criptomoneda, siempre y cuando utilice el mismo algoritmo de hash criptográfico que Bitcoin.

La puesta en funcionamiento del `CPU miner` es simple, solo requiere de configurar la dirección IP del *pool* de minería y la cantidad de *threads* del CPU que se quieren destinar para el cómputo de hashes. Para el caso de `BTCPool`, el minero se conecta al microservicio `sserver`. Una vez que la conexión es establecida, se utiliza el protocolo *Stratum* tanto para la recepción de trabajo como para el envió de *shares*. A partir de ahí, el minero computa hashes para cada uno de los trabajos provistos por el software de *pool* de minería.

Hasta el momento, se introdujeron los conceptos relevantes de criptomonedas y blockchain a través de la presentación del caso Bitcoin. Luego, se puso foco en conceptos relacionados a minería ya que es uno de los temas centrales de esta tesis.

A continuación, se avanzará con más conceptos de minería, en este caso con *merged mining* y todos las nociones de relevancia que lo rodean.

1.2 Merged Mining

Especificado inicialmente en el año 2015 [4], es el proceso por el cual los mineros pueden generar pruebas de trabajo para dos o más blockchains, sin esfuerzo de minado adicional. En este proceso, existen dos cadenas donde una es llamada **cadena primaria** y la otra es denominada **cadena secundaria** o **cadena auxiliar**.

La adopción del *merged mining* ocurre cuando los mineros, que ya se encontraban minando la cadena primaria, suman su poder de cómputo para minar cadenas secundarias vía *merged mining*. Las principal razón que lleva a los mineros a realizar *merged mining*, son los incentivos económicos; éstos son las recompensas que pueden ganar por su trabajo realizado para las cadenas auxiliares. Todo lo anterior, sin incurrir en gastos extra a los de su operación de minado habitual para la cadena primaria.

El *pool* de minería se ve beneficiado de igual forma que los mineros, ya que puede ahora ganar una comisión del ingreso que reciben los mineros por su trabajo en la cadena secundaria. Distinto es el caso a nivel técnico, el *pool* de minería debe modificar su software de *pool* de minería para poder realizar *merged mining*.

Para las cadenas involucradas, el beneficio es que pueden establecer entre ellas una relación a nivel de datos. Lo anterior es posible gracias a que el *merged mining* fija una referencia entre ambas cadenas. Además, el *merged mining* es ideal para las cadenas secundarias ya que les presenta la oportunidad de obtener *hashing power* de forma rápida. Esto, en comparación con el tiempo que le llevaría a una blockchain nueva generar una comunidad de minado que la provea de la seguridad suficiente.

Hasta el momento se introdujo el concepto de *merged mining* de manera conceptual. Se desconoce, el rendimiento del mismo en la práctica, es decir, si su funcionamiento tiene algún impacto por sobre el desempeño de la cadena primaria.

Este trabajo se concentrará en el estudio del rendimiento del *merged mining* con Bitcoin como cadena primaria y RSK como cadena secundaria. De ahora en adelante, y a menos que se explicite lo contrario, siempre que se haga referencia a *merged mining* será para las cadenas de Bitcoin y RSK.

Esta introducción brindó la información necesaria para entender Bitcoin y su minado en la sección 1.1.1 y en la sección 1.1.2, respectivamente.

A continuación, se exhibirán detalles del funcionamiento de RSK, necesarios para poder entender el proceso de *merged mining* entre Bitcoin y RSK.

El realizar *merged mining* partiendo de una infraestructura de minado Bitcoin, solo requiere cambios en el software de pool de minería. Es por lo anterior, que para finalizar esta sección se presentarán las modificaciones para que `BTCPool`, a nivel de implementación, permita realizar *merged mining* a quienes lo utilicen.

1.2.1 RSK

En esta sección se presentará la plataforma RSK [7]. Para poder lograr esto, se utilizará a Bitcoin como base para los conceptos de blockchain y sobre eso se exhibirán las bondades de RSK.

Bitcoin es una red que permite realizar transferencias de valor entre dos pares. Dichas transferencias, están condicionadas en su complejidad por el lenguaje de *scripting* de Bitcoin. Un lenguaje que cuenta con una limitada expresividad, ya que por ejemplo, no tiene ciclos. Esto provoca que no sea Turing completo. Lo más complejo que se puede presentar en una transacción Bitcoin, es un *output* que requiera la verificación de múltiples firmas digitales.

Como respuesta a la necesidad de agregar mayor lógica a las transacciones, es que surgen las plataformas de *contratos inteligentes* o *smart contracts*.

[4] `https://en.bitcoin.it/wiki/Merged_mining_specification`

Los contratos inteligentes utilizan un lenguaje Turing completo, que hace posible la ejecución de código sin las restricciones de Bitcoin. La eliminación de las restricciones y la posibilidad que ofrecen los *smart contracts* de comunicarse entre sí, abre la puerta a un sinfín de nuevos casos de uso sobre blockchain.

A modo de ejemplo, existen conjuntos de *smart contracts* que permiten llevar la transferencia de valor introducida por Bitcoin a un nuevo nivel. Esto se logra al incorporar conceptos de las finanzas tradicionales, como los instrumentos financieros, el *trading* de dichos instrumentos, la implementación de préstamos entre pares, entre otros.

RSK es una de varias plataformas de *smart contracts* que existen en la actualidad. Es muy similar a Bitcoin, en cuanto a que maneja estructuras de datos de transacciones, bloques y cuenta con su propia blockchain. A diferencia de Bitcoin que utiliza direcciones, RSK maneja el concepto de cuentas que van modificando su estado cada vez que forman parte de una transacción incluida en un bloque.

Las transacciones de RSK son más complejas que las de Bitcoin, puesto que no solo permiten la transferencia de la moneda nativa de la plataforma, sino que también facilitan la inclusión de código compilado para la ejecución de *smart contracts*.

Existe un costo asociado a la ejecución de cada transacción, llamado **Gas**; está determinado por las instrucciones que ejecuta y los *bytes* que transfiere o ejecuta. A la vez, el límite para dicha ejecución, está dado por la cantidad de dinero disponible en las cuentas, en particular en la cuenta que generó la transacción. Es responsabilidad de quien envía la transacción establecer el valor de **GasLimit**, que indica cuanto puede se puede gastar como máximo durante la ejecución. Además, el **GasPrice** determina cuánto se gastará por cada unidad de `Gas` siendo éste, fijado en **RBTC**, la moneda nativa de la red de RSK.

Los valores de `Gas`, `GasLimit` y `GasPrice`, son parte del encabezado del bloque de RSK. El bloque de RSK, al igual que el bloque de Bitcoin, se divide en *header* y *body*.

El RBTC tiene paridad de uno a uno con el bitcoin lo cual quiere decir que 1 bitcoin equivale a 1 RBTC.

La paridad implica que RSK no acuña nuevas monedas y esto es fundamental, porque determina que los incentivos para minar bloques provengan de las comisiones de las transacciones. La manera que ofrece la red de obtener RBTC, es enviando bitcoin a través de un *bridge* entre las dos redes. La conversión en sentido inverso, también es válida. Este mecanismo no es relevante para esta tesis por lo que no se presentará en detalle.

RSK y Bitcoin también coinciden en que ambos utilizan el mismo algoritmo de PoW. En consecuencia, RSK puede ser la cadena secundaria de un proceso de *merged mining* donde Bitcoin sea la cadena primaria.

Estos últimos dos puntos, misma moneda y mismo algoritmo de PoW, muestran que las dos cadenas tienen bastante en común. RSK podría ser la blockchain que provea a Bitcoin de una capa de lógica que siempre se quiso incluir pero que a la vez fue no adoptada, debido a que implica cambios muy grandes en el protocolo.

En el aspecto técnico, Bitcoin y RSK también se parecen ya que RSK tiene su lógica implementada en un cliente o nodo que cada usuario puede descargar y ejecutar para formar parte de la red descentralizada. RSK posee los mismos tipos de redes que Bitcoin, es decir, *mainnet*, *testnet* y *regtest*. El código fuente del nodo de RSK, está programado en `Java` y ofrece un registro de eventos configurables para poder acceder a detalles de su funcionamiento.

Hasta el momento se presentó una visión general y suficientemente detallada, para entender los conceptos necesarios sobre la red de RSK. Resta introducir un último concepto y dada la complejidad y extensión del mismo se exhibirá en la próxima sección.

1.2.1.1 Minería

En cuanto a minería y consenso, RSK no es tan sencillo como Bitcoin.

En Bitcoin, el único bloque que da recompensa a los mineros es aquel que es aceptado en la *mainchain*, mientras que en RSK se premia a más bloques y de manera diferente.

Los bloques de RSK tienen un tiempo de madurez y, a lo largo de ese tiempo, se van acumulando sus comisiones o *fees*, en lo que se conoce como **reward balance**. Cada bloque tiene una transacción que llama a un *smart contract*, el **reward manager smart contract (REMASC)**, que es el encargado de calcular los valores para el pago de recompensas y de transferir las mismas a quien le corresponde. Cuando un bloque aceptado en la *mainchain*, llamado **best block**, cumple con la madurez establecida por la red entonces recibe el 10 $\%$ del *reward balance*.

RSK, además, premia a los bloques que no quedan en *mainchain* pero que son válidos. Un bloque válido pero no aceptado para la *mainchain* se denomina **sibling**, en el contexto de la altura de la blockchain para la que fue minado.

Cada bloque de RSK tiene como parte de su estructura, un lugar para agregar a estos bloques. Como un bloque válido no se puede modificar, pues dejaría de servir su PoW, lo que se hace es conservar esos bloques y permitir su agregado en bloques posteriores. La adición de bloques válidos de alturas anteriores a la del bloque que está siendo minado, se hace bajo el nombre de **uncles**.

En la figura 1.9, se puede ver un ejemplo donde se grafican varios de los conceptos introducidos previamente. Para su correcta lectura, hay que mencionar que N representa la altura a la que corresponde el bloque. En dicha figura se observa que:

- Los bloques A, B y C comparten el mismo padre P.
- B es el *best block* de altura N, por esto son A y C son *siblings*. Al ser minado como *best block* de altura N+1, D incluye a C como *uncle*. Al ser minado como *best block* de altura N+2, E incluye a C como *uncle*.

  Con A y C agregados como *uncles*, es correcto afirmar que para la altura N los bloques minados fueron A, B y C.
- A los bloques que incluyen *uncles* se los suele llamar *publishers*, en este caso son D y E.
- El bloque F es un nuevo bloque agregado a la *mainchain*, su padre es E, el *best block* al momento en que se buscaba una prueba de trabajo válida para F.

En caso de existir, los *uncles* reciben recompensa igual que los *best blocks*. La forma en que se distribuye es dividiendo el *reward balance* correspondiente a la altura a pagar en partes iguales según la cantidad de bloques válidos. A la vez, el bloque que decide incluir *uncles*, es también recompensado.

Es importante destacar, que los *uncles* pueden ser agregados como máximo diez bloques después de la altura para la que hubieran sido *best blocks*. También existe un máximo de diez *uncles* por altura.

Por último, los mineros reciben solo el 80 $\%$ del *reward balance*. El 20 $\%$ restante es distribuido entre otros actores de la red de RSK.

El *target time* en la red productiva de RSK es de 30 s. Dicho tiempo promedio de bloques se mantiene gracias al ajuste de la dificultad. El algoritmo de ajuste de dificultad de RSK lo hace bloque a bloque a diferencia de Bitcoin, que utiliza una ventana de 2015 bloques.

El nodo de RSK ofrece una interfaz RPC para la interacción de todo lo referido al minado.
La misma expone los siguientes métodos:

- **mnr_getWork.** Provee la información requerida para poder minar un nuevo bloque.

  Entre los campos más relevantes se encuentran el hash criptográfico del bloque candidato de RSK, la dificultad y un *flag* llamado **notify**, que es verdadero solo cuando este trabajo es distinto al último entregado.

[5] `https://developers.rsk.co/rsk/architecture/mining/remasc/`

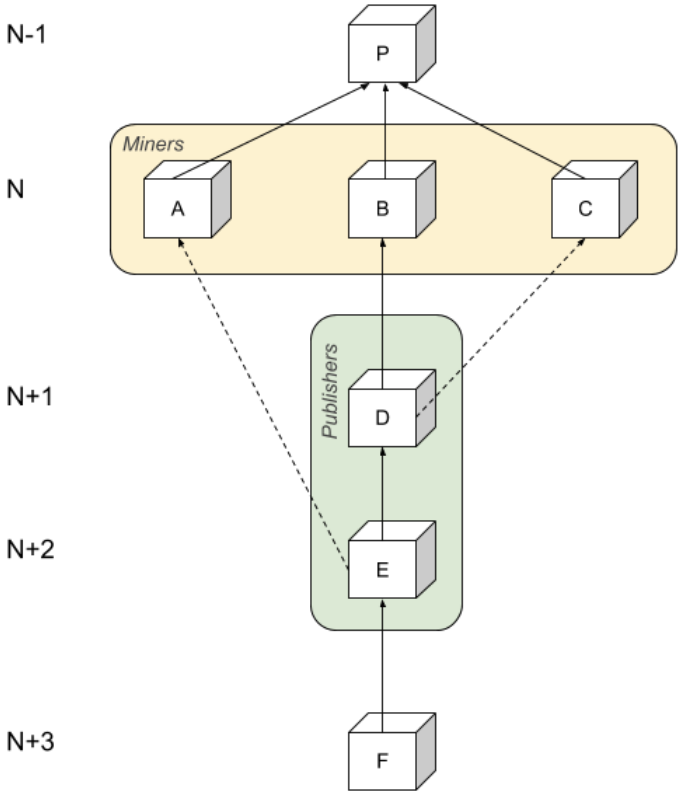


Figura 1.9: Ejemplo de minado de bloques en RSK. Fuente: Documentación RSK [5]

- **mnr_submitBitcoinBlockPartialMerkle.** Uno de los múltiples métodos que el nodo ofrece para recibir bloques con prueba de trabajo válida.

  El mismo recibe los siguientes parámetros:

  1. **blockHashHex.** Hash criptográfico del bloque Bitcoin usado en *merged mining*.
  2. **blockHeaderHex.**. *Header* del bloque Bitcoin usado en *merged mining*.
  3. **coinbaseHex.** Transacción *coinbase* del bloque Bitcoin usado en *merged mining*.
  4. **merkleHashesHex.** Hashes criptográficos del *merkle tree* del bloque Bitcoin, usado para *merged mining*.
  5. **blockTxnCountHex** Cantidad de *merkle* hashes enviados en el parámetro anterior.

  Una vez procesada la información, la respuesta de este método contiene:

  1. **blockImportedResult.** El resultado de agregar el bloque a la blockchain.

     El mismo puede ser *IMPORTED_BEST* que indica que el bloque fue aceptado como *best block*, *IMPORTED_NOT_BEST* que indica que el bloque fue aceptado como *uncle* y otros resultados que indican que el bloque no fue aceptado.
  2. **blockHash.**. Hash criptográfico del bloque al que se corresponde la respuesta.
  3. **blockIncludedHeight.** Altura en la que el bloque fue agregado a la blockchain, en caso de haber sido aceptado.

Cabe destacar que RSK construye el bloque candidato de manera interna en el nodo. Es por lo anterior, que en la respuesta del método `mnr_getWork` se incluye un hash criptográfico y no una lista de transacciones como lo hace Bitcoin, en la respuesta del método `getBlockTemplate`.

Esta sección introdujo todo lo referido a RSK.

Finalmente, se presentará una especificación para hacer *merged mining* con Bitcoin como cadena primaria y RSK como cadena secundaria. Como parte de la misma sección se presentarán los cambios que se hicieron a `BTCPool`, para poder hacer *merged mining* respetando la especificación.

1.2.2 Implementación para Bitcoin y RSK

La red de RSK no permite el minado de bloques de la misma manera que ocurre en Bitcoin. Para poder minar un bloque en RSK se debe hacer vía *merged mining*.

Dicho proceso implica utilizar la prueba de trabajo obtenida para un bloque Bitcoin para validar un bloque de RSK de manera directa, esto gracias a que ambos comparten el mismo algoritmo de prueba de trabajo. La condición que debe cumplir dicha prueba es ser mayor a la dificultad de la red de RSK.

La forma que propone *merged mining* para satisfacer la dificultad de RSK es incluir una referencia al bloque de RSK en el bloque Bitcoin.

En particular, la referencia será el hash criptográfico del bloque de RSK e irá insertada como *output* de la transacción *coinbase*, tal como puede verse en la figura 1.10.

Además, se pueden ver en detalles los campos del *header* y del *body* de los bloques de Bitcoin y RSK. En el caso de RSK, se muestran en verde los datos de Bitcoin que se almacenan en cada bloque. Dichos datos son utilizados para validar la prueba de trabajo obtenida por *merged mining*.

Se eligió la *coinbase*, ya que es la que menos cambios implica para el proceso de minado que el minero ya realiza. Además, es la única transacción sobre la que el minero tiene control total, ya que es la que debe armar para poder cobrar las recompensas por el minado de un bloque Bitcoin.

El agregado de la información de RSK se hará como el último *output* de la *coinbase*. Dicha información contará con el siguiente formato:

```
OP_RETURN + Length + RSKBLOCK: + RskBlockInfo
```

El signficado de cada uno de los valores es:

- **OP_RETURN.** Es una instrucción del lenguaje de scripting de Bitcoin, indicando que un *output* es invalido. Es correcto marcarlo de esta manera ya que el *output* de *merged mining* no tiene uso en la red de Bitcoin.
- **Length.** Representa el largo de la información a incluir después de la instrucción `OP_RETURN`
- **RSKBLOCK:** Identificador que se utiliza para indicar que a continuación se incluirá información de RSK.
- **RskBlockInfo.** Hash criptográfico que identifica al bloque con el que se quiere hacer *merged mining*.

El nodo RSK, es quien luego debe interpretar la *coinbase* del bloque Bitcoin, para validar su bloque. Es por lo anterior, que RSK establece ciertas restricciones para que dicha transacción sea válida. Entre las más relevantes se encuentran, que luego de `RskBlockInfo` no pueden incluirse más de 128 B y que dentro de esos *bytes*, no puede aparecer el identificador `RSKBLOCK:`.

Las dificultades de Bitcoin y RSK no son iguales y la relación entre ellas depende de dos factores.

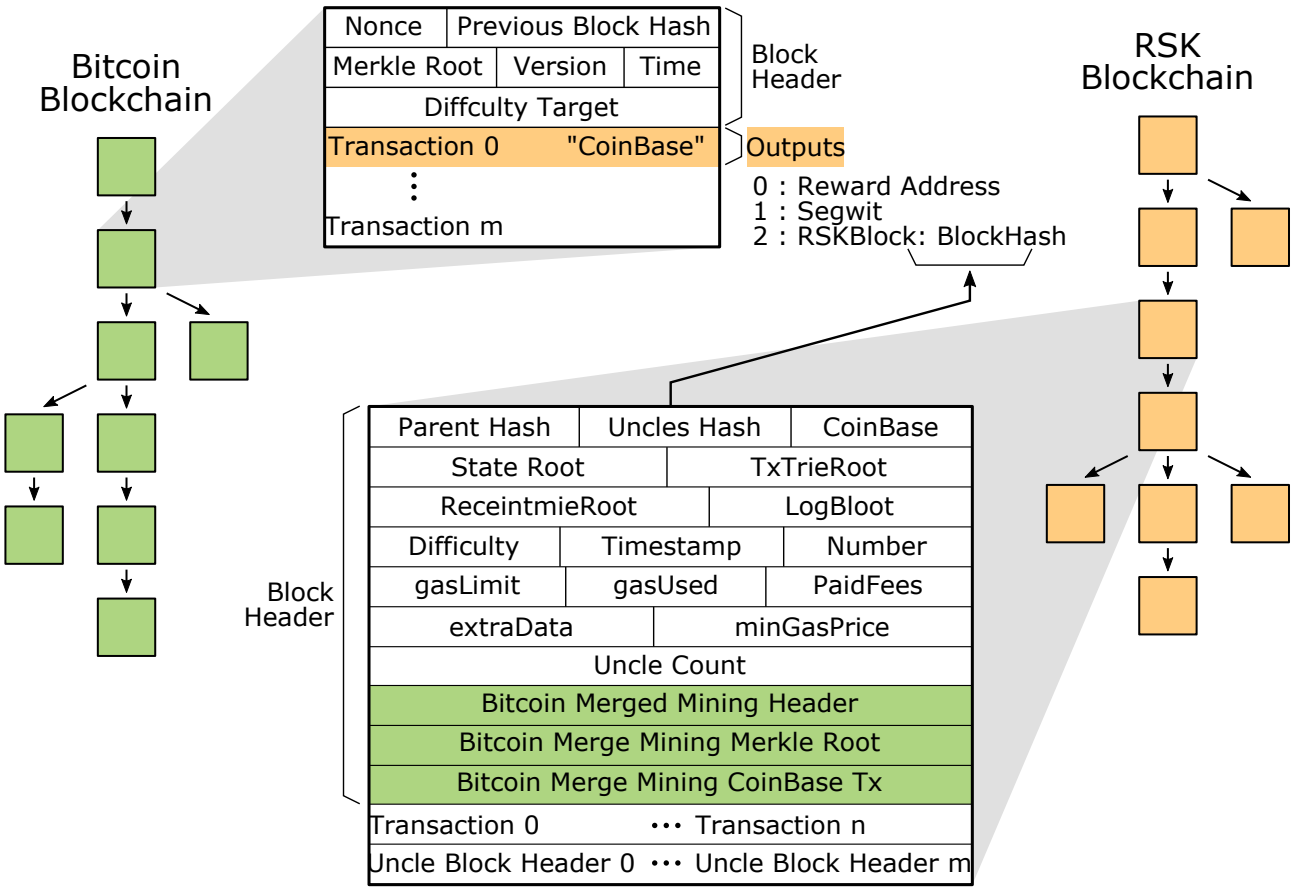


Figura 1.10: Relación de bloques para *merged mining* entre Bitcoin y RSK. En el esquema se muestra cómo se coordina el minado. Gracias a esto, al buscar soluciones para un bloque Bitcoin, también se buscan para RSK.

Por un lado, la dificultad se ajusta en base al *hashing power* que los mineros aportan. Esto con el objetivo de mantener el tiempo entre bloques promedio.

Entonces, el *hashing power* de cada red determinará el valor de dificultad de cada una de ellas. Dado que los bloques para RSK solo pueden ser minados vía *merged mining* con Bitcoin, es correcto afirmar que el *hashing power* de RSK, será menor o igual al de Bitcoin. Por ende, la dificultad será también menor o igual.

Por otro lado, RSK tiene un tiempo entre bloques de 30 s mientras que Bitcoin tiene un tiempo entre bloques de 10 min. Cuanto mayor frecuencia de minado se necesite, menor debe ser el valor de la dificultad. En consecuencia, la dificultad para minar de RSK, será menor que la de Bitcoin.

Hasta el momento se determinó la relación entre las dificultades de Bitcoin y RSK. Se puede afirmar que la dificultad de Bitcoin, será siempre mayor que la de RSK y a la vez, ambas serán varios ordenes mayores que la dificultad del *pool*.

Conocer la relación entre las dificultades de las redes, permite concluir que siempre que se encuentre una solución para un bloque Bitcoin, la misma será también una solución para el bloque de RSK asociado.

Sin embargo, el hecho de que todas las soluciones de Bitcoin se puedan utilizar para RSK, no es suficiente para satisfacer el tiempo entre bloques de RSK. Es decir, soluciones cada 10 min en promedio (*target time Bitcoin*), no permiten minar bloques cada 30 s (*target time* RSK).

Por consiguiente, para poder satisfacer el *target time* de RSK, hay que usar *shares* que para el caso de Bitcoin son descartadas por no cumplir con la dificultad.

Esto modifica el escenario de evaluación de validez de un *share*, por parte del software de *pool* de mine-

ría. Los casos que el mismo debe considerar para *merged mining* son los siguientes:

- **El hash propuesto por el minero cumple con la dificultad de Bitcoin.** Este es el caso que el *pool* ya evaluaba en el minado Bitcoin sin *merged mining*. La solución será válida para ambas redes, Bitcoin y RSK.

- **El minero envía un hash que cumple con la dificultad de RSK pero no con la de Bitcoin.** Es aquí, cuando debe enviarse dicha solución a la red de RSK y descartarla para la red de Bitcoin.

- **El software de *pool* de minería recibe un hash que solo cumple con la dificultad del *pool*.** En esta situación, dicha solución no será válida para Bitcoin ni para RSK y solo servirá para ser contabilizada como trabajo aportado por el minero.

Tener en cuenta nuevos casos al momento de recibir *shares*, es solo una parte de lo necesario para poder hacer *merged mining*. La otra parte, implica que el *pool* actualice el trabajo de sus mineros no solo frente a nuevo trabajo de Bitcoin sino también frente a nuevo trabajo de RSK.

Para dicho cometido es que *merged mining* introduce la propuesta de **notify policies**. Una *notify policy*, establece bajo qué condiciones se debe actualizar el trabajo de los mineros. Los casos que tiene en cuenta son:

- **El *pool* no modifica su comportamiento de actualización de trabajo hacia los mineros.** Es decir, que sigue actualizando el trabajo tal como lo hacía para Bitcoin.

  Esto implica que el minado de RSK no cumplirá con el tiempo entre bloques deseado, ya que su actualización dependerá del minado de un nuevo bloque Bitcoin o de la actualización periódica del trabajo Bitcoin realizada por el software de minería.

- **Los mineros reciben una actualización del software de *pool* de minería cuando hay un nuevo trabajo de RSK. Pueden terminar su trabajo actual antes de comenzar con el nuevo.**

  Dependiendo del tipo de minero, puede demorar más o menos tiempo en finalizar su trabajo y esto puede impactar en el ritmo de minado. En consecuencia, puede que en ciertos casos se demore más tiempo en minar un bloque. Aún así, en este escenario se espera que RSK pueda cumplir con bloques para el *target time* solicitado.

- **Los mineros reciben una actualización del software de pool de minería cuando hay un nuevo trabajo de RSK. Deben comenzar con el trabajo nuevo de inmediato.**

  En este escenario, se sabe que RSK podrá minar bloques cumpliendo con su *target time*.

Esta sección presentó todos los cambios que se deben hacer al proceso de minado Bitcoin, para poder hacer *merged mining* con RSK. A continuación, se explicará cómo dichos cambios fueron aplicados en `BTCPool` para que *merged mining* sea posible en ambientes productivos.

1.2.2.1 BTCPool

Este software de *pool* de minería soporta *merged mining* con diferentes blockchains, entre ellas se encuentra RSK.

En la presente sección se presentan las funcionalidades extra y las modificaciones que el *pool* tiene, para poder realizar *merged mining* entre Bitcoin y RSK.

Para mayor claridad, las mismas se presentan diferenciadas por microservicio que las implementa:

- ***Get work maker (gwmaker).***

  Responsable de interactuar con el nodo RSK a través del método `mnr_getWork`.
  Dicha interacción ocurre utilizando el mecanismo de *polling* y la frecuencia del mismo es configurable vía archivo de configuración. Además, en el archivo de configuración se puede configurar todo lo necesario para conectarse al nodo RSK.

  Al obtener la respuesta del llamado a `getWork`, los datos de la misma son interpretados y volcados en estructuras de datos internas, para ser luego serializados en formato `JSON` e incluidos en una cola de mensajes llamada `RawGw`.

  Este componente es nuevo y fue especialmente creado para poder hacer *merged mining* con RSK.

- ***Job maker (jobmaker).*** Se le agrega la responsabilidad de gestionar la información de minado de RSK, que obtiene de consumir la cola de mensajes `RawGw`.

  A cada trabajo se le adicionan los datos obtenidos del nodo de RSK. Eso posibilita el agregado de los mismos al bloque candidato Bitcoin siendo este un bloque de *merged mining* con RSK.

  A las condiciones de generación de nuevos trabajos que el `jobmaker` tenía, se le añade una nueva para contemplar la aparición de nuevo trabajo de RSK. Para poder juzgar si generar un nuevo trabajo de RSK, utilizará la información de minado de RSK y la *notify policy* que tenga configurada. La *notify policy*, está implementada con números que representan los distintos casos existentes. El número deseado de política se lee del archivo de configuración de este microservicio.

- ***Stratum server (sserver).*** Al momento de construir el bloque candidato, debe agregar la información de *merged mining* de RSK a la *coinbase*. Dicha información es obtenida del trabajo proveniente de la cola de mensajes `BtcJob`.

  Al recibir una *share*, se debe validar la misma contra la dificultad de minado de RSK, para poder determinar si es solución. La dificultad de minado de RSK es obtenida del trabajo proveniente de la cola de mensajes `BtcJob`.

  En caso que la *share* cumpla con la dificultad del RSK, el *pool* tomará la información de la *share*, junto a la del encabezado del bloque Bitcoin, la serializará en formato `JSON` y la enviará a la cola de mensajes, llamada `RskSolvedShare`.

- ***Block maker (blkmaker).*** Suma la responsabilidad de enviar bloques válidos al nodo de RSK, a través del método `mnr_submitBitcoinBlockPartialMerkle` de dicho nodo.

  Para construir el bloque válido, este microservicio lee mensajes de las colas de mensajes `RawGw`, `RskSolvedShare`, `BtcJob` y `RawGbt`. Con toda la información recabada, cruza datos para corroborar que efectivamente tiene que armar un bloque válido y, en caso afirmativo, arma el bloque y lo envía a la red de RSK.

Es importante destacar que el rol prioritario del *pool* es minar Bitcoin, es por eso que todo lo que se agrega para RSK, tiene un orden de prioridad menor. Por ejemplo, si se encuentra una solución para Bitcoin y RSK, primero se enviará la misma a la red de Bitcoin y luego se hará lo propio con la red de RSK.

1.3 Antecedentes

En la práctica, existen criptomonedas que ya han utilizado *merged mining* en los últimos años.

`Namecoin` fue la primera y lo hizo con Bitcoin como cadena principal. Otras fueron, `Litecoin` y `Dogecoin`, aunque estas usan un algoritmo de prueba de trabajo diferente al utilizado por Bitcoin. Los ejemplos anteriores indican que *merged mining* es un tema instalado y una metodología utilizada en la práctica por la comunidad de la criptomonedas.

En cuanto a la teoría y, al momento de la escritura de este trabajo, no existen publicaciones sobre la eficiencia del *merged mining*. Sin embargo, sí existen trabajos científicos que tratan el tema desde otros ángulos:

- Judmayer et al. [5] se concentra en los problemas de seguridad que el *merged mining* puede traer a las criptomonedas que lo utilicen.
- Kalodner et al. [6] estudia `Namecoin`, que es la moneda más antigua en utilizar el proceso de *merged mining*.
- Ali et al. [1] propone una nueva blockchain y presenta su experiencia desde el punto de vista del funcionamiento para un sistema de gran escala. Ellos presentan algunas características interesantes sobre *merged mining* realizado en `Namecoin`.
- Stifter et al. [10] presenta una metodología para poder extraer información de *forks* en la cadena principal de blockchains que hacen *merged mining*. Esto lo logra a partir de los datos de dicha cadena, que se encuentran almacenados en los bloques minados en las cadenas secundarias.
- Zhipeng et al. [12] propone utilizar *merged mining* para escalar Bitcoin a través de la utilización de una cadena secundaria para minar un tipo particular de transacciones denominadas *pequeñas y rápidas*.

Capítulo 2

Métodos

En este capítulo se explicará en detalle la metodología utilizada en la realización de los experimentos de este trabajo.

Primero se mostrará, en la sección 2.1, el entorno de pruebas a usar; incluyendo una descripción del mismo y las decisiones que llevaron a su elección.

Luego, en la sección 2.2 se detallarán las modificaciones a nivel técnico que se realizaron a los componentes de software. Dichos componentes son los que forman parte del entorno de pruebas y los cambios serán cubiertos componente a componente.

Además, se incluirá en la sección 2.3, toda la información relacionada a la generación de datos necesarios, para los distintos escenarios en los que se ejecutarán los experimentos.

A continuación, se mostrará un detalle de las configuraciones necesarias para lograr el funcionamiento de los componentes de software. Esto como parte de la sección 2.4.

Contando con el entorno de prueba, las versiones finales de los componentes de software a utilizar y las configuraciones de los mismos, se procede a ejecutar un experimento de prueba. Dicha experiencia se detalla en la sección 2.5. Como parte del mismo apartado, se exhibirán las medidas de control que se implementaron para determinar que tanto el software, como los experimentos, se ejecuten de forma satisfactoria.

Finalmente, la sección 2.6 exhibirá los programas necesarios para automatizar la ejecución y repetición de las pruebas.

2.1 Entorno de pruebas

Los experimentos a realizar, deben ser ejecutados en un entorno que sea lo más similar posible a aquél usado en la realidad, para el minado de Bitcoin y el *merged mining* con RSK.

Además, el entorno debe proveer cierto nivel de control sobre la cantidad de transacciones, tanto de Bitcoin como de RSK. También debe brindar control sobre el *target time* de Bitcoin, el *target time* de RSK y el utilizado para regular la frecuencia de las *shares*, en el software de *pool* de minería.

Debido a la condición de control, el entorno de pruebas será una red que no tenga conexión con las redes productivas existentes de Bitcoin y de RSK, *mainnet* o *testnet*. Es decir, esta red de pruebas estará aislada.

Son varias las razones que llevaron a esta decisión; una de ellas es que no se puede modificar el *target time* de Bitcoin y participar de una red con *target time* diferente, como por ejemplo *mainnet* ya que esto rompería el consenso. Romper el consenso quiere decir que los bloques de una red, no serán válidos en la otra que tenga reglas diferentes y viceversa.

Otra de las razones, es que en una red productiva no hay control sobre la generación de transacciones; es decir que existen otros actores que generan transacciones, como parte de su operatoria diaria. En el caso de *mainnet* de Bitcoin, existe además el hecho de que las transacciones tienen un costo, a diferencia de *testnet*, donde también lo tienen, pero los bitcoins allí utilizados se pueden conseguir gratuitamente.

Ambas razones previamente expuestas, son también válidas para la red de RSK.

La última de las razones relevantes que se expondrán está relacionada con el minado. Incluso pudiendo controlar el *target time* y la cantidad de transacciones para ambas redes, no se cuenta con poder de minado suficiente para generar bloques que cumplan con la dificultad de las mismas. Esto se torna en un limitante, dado que para los experimentos es necesario tener control sobre los bloques minados.

La red de pruebas será entonces, una red especialmente montada para los experimentos a realizar en el marco de este trabajo. La misma proveerá las condiciones necesarias para tener un entorno de minado, que sea igual al productivo en las redes Bitcoin y RSK, pero sin contar con conexión las mismas.

Para poder lograr esto, se cuenta con una `LAN` que tiene conectados vía `Ethernet` los siguientes equipos:

- **Máquina A**.
  - Procesador `AMD Opteron(TM) 6320`, frecuencia de reloj 1,40 GHz
  - Memoria 512 GB DDR3 1600 MHz
  - `Ubuntu 20.04.2 LTS`
- **Máquina B**.
  - Procesador `Intel(R) Core(TM) i7-3770`, frecuencia de reloj 3,40 GHz
  - Memoria 16 GB DDR3 1033 MHz
  - `Ubuntu 20.04.2 LTS`

En las máquinas de la `LAN`, se ejecutarán todos los componentes de software involucrados en el proceso de minado de los experimentos de este trabajo. Para esto, se utilizará el código fuente de las versiones productivas de dichos componentes. Sin embargo y para poder correr las pruebas, los mismos tendrán algunas modificaciones, que se presentarán en la sección 2.2.

Dichas modificaciones no afectan los requisitos mínimos de hardware, que cada uno de ellos pide, para poder funcionar con normalidad. Las máquinas de la red de pruebas cumplen con creces con dichos requisitos, incluso para el caso en que varios componentes fueran ejecutados en una misma máquina.

En un ambiente productivo no se suelen usar máquinas con tanta capacidad como la máquina A. Por lo anterior, se ejecuta el software de *pool* de minería junto con el nodo de Bitcoin en una máquina. En caso de haber *merged mining*, el nodo de RSK suele estar en una máquina separada pero siempre formando parte de la misma red local.

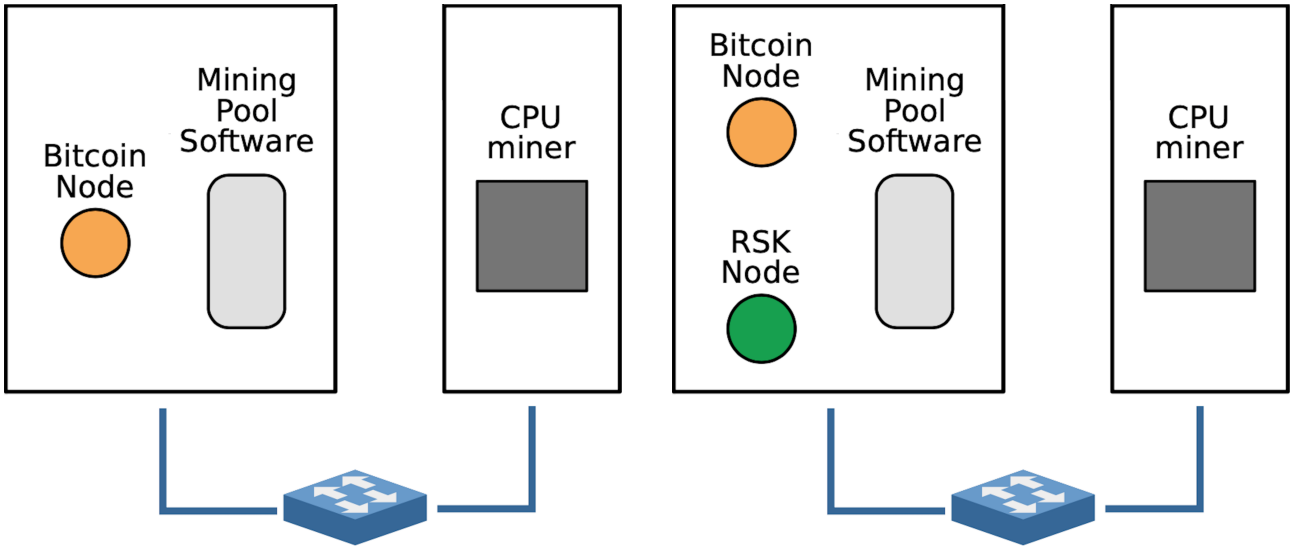
En la red de pruebas se ejecutarán los nodos de Bitcoin y de RSK, en la máquina A, junto con el software de *pool* de minería. La máquina B será utilizada únicamente para minar.

La red de pruebas descripta hasta el momento es la que permitirá ejecutar todos los experimentos. Para que esto ocurra, se usarán dos distribuciones principales de los componentes de software.

Una de ellas será la **red de minado Bitcoin** que tendrá un par de nodos Bitcoin, un software de *pool* de minería y un minero.

La otra red será la **red de minado Merged Mining**, que tendrá los mismos componentes que la red de minado Bitcoin, con el agregado de un nodo de RSK.

Ambas redes puede verse en la figura 2.1.



(a) Red de minado Bitcoin.
A la izquierda se representa a la máquina A, donde se ejecutará el nodo de Bitcoin junto con el software de *pool* de minería.
A la derecha figura la máquina B, que será utilizada únicamente para minar.

(b) Red de minado *merged mining*.
A la izquierda se representa a la máquina A, donde se ejecutará el nodo de Bitcoin y el nodo de RSK junto con el software de *pool* de minería.
A la derecha figura la máquina B, que será utilizada únicamente para minar.

Figura 2.1: Distribución de los componentes necesarios para los experimentos en cada una de las redes experimentales, utilizadas en este trabajo.

El hecho de que las redes tengan la distribución de componentes mencionada antes, no implica que sean los mismos programas ejecutables, o las mismas configuraciones de dichos ejecutables, los que se utilicen en cada experimento. Por ejemplo, se utilizarán distintas versiones del nodo Bitcoin según la cantidad de transacciones que se quieran incluir en el bloque. El último ejemplo y demás versiones de software usadas para cada uno de los componentes, serán introducidos más adelante en este capítulo.

Es importante destacar que tanto las máquinas como la red donde están conectadas, fueron de uso exclusivo para los experimentos de este trabajo; es decir, no se ejecutaron otros programas o incluso experimentos de este trabajo al mismo tiempo.

A continuación, se presentarán las razones que llevaron a la elección de las implementaciones de los componentes a utilizar en los experimentos. Además, en la tabla 2.1, se muestran las versiones de los mismos.

En el caso del nodo de Bitcoin, se decidió utilizar la implementación conocida como `Bitcoin Core`. La misma es la más popular a nivel adopción y es la evolución del código inicial creado por Satoshi Nakamoto.

Para el nodo de RSK, se optó por la implementación llamada `RSKj` y fue elegida por ser la única disponible.

En cuanto al minero, la primera decisión tomada, fue la del tipo de minado a realizar. Al no contar con un `ASIC` ni con una `GPU`, es que se decidió de usar `CPU`.

Para eso, se escogió el programa `CPU miner` por ser la implementación de referencia surgida al momento de comenzar a minar con `CPU`, en un componente separado al nodo Bitcoin. Otra de las razones que llevaron a su elección, es que posee un código de fácil lectura, lo cual hace más sencillo la aplicación de cambios.

La decisión del software de *pool* de minería requirió de un estudio, ya que tal como se mencionó en la introducción, existen múltiples soluciones de software de *pool* de minería; cada una de ellas con diferentes decisiones de diseño, tanto a nivel software como de negocio.

Dicho estudio se concentró en el uso de los distintas aplicaciones de *pool* de minería en la red productiva de Bitcoin.

Para conocer la adopción de cada software de *pool*, se asoció a aquellos con mayor *hashing power*, al software que emplean. Luego, se procedió a elegir el más utilizado.

A raíz de este estudio, se decidió usar el *pool* llamado `BTCPool`, que tanto al momento de los experimentos como de la escritura de este documento, es el que más *hashing power* posee.

Componente	Versión
`Bitcoin Core`	`0.16.0`[1]
`RSKj`	`1.0.0`[2]
`CPU miner`	`2.5.0`[3]
`BTCPool`	`Latest`[4]

Tabla 2.1: Versiones de los componentes a utilizar en los experimentos de este trabajo.

Conviene acotar, que siempre se buscó utilizar las últimas versiones de cada uno de los componentes, disponibles al momento de realizar las pruebas. Sin embargo, para el caso de `Bitcoin Core` esto no fue posible. La razón que provocó esto fue que el software `BTC Pool` requiere refenciar al código fuente de Bitcoin, al momento de compilación. Las últimas versiones de Bitcoin no eran compatibles, por lo cual se tuvo que usar la primera que si lo fuera, la versión `0.16.0`.

Hasta el momento se presentó el entorno de pruebas a utilizar en los experimentos de este trabajo, junto con las dos configuraciones de redes elegidas.

Se incluyó tanto una descripción de los equipos que forman parte de ellas, como así también una distribución de los componentes de software en las mismas. Para cada componente, se mostró la versión de código a utilizar. Partiendo de la versión productiva de cada componente, se requirieron de cambios en su código fuente, para poder cumplir con que las condiciones de la red experimental sean iguales a las de una productiva.

En la siguiente sección, se presentan los componentes juntos con dichas modificaciones.

2.2 Modificación de componentes de software

2.2.1 Cliente Bitcoin

El cliente `Bitcoin Core` será utilizado como implementación del protocolo Bitcoin. Con el fin de poder realizar los distintos experimentos, se realizaron cambios de protocolo a la red *regtest*. Vale recordar que esta red tiene parámetros predeterminados distintos a la *mainnet* y se usa principalmente para hacer pruebas y experimentos.

Como en los experimentos se necesita emular a la red real, en la sección 2.2.1.1 se mencionan todos los cambios realizados, para que la red *regtest* del cliente utilizado en las pruebas, emule a la red *mainnet*.

[1] https://bitcoin.org/en/release/v0.16.0

[2] https://github.com/rsksmart/rskj/releases/tag/WASABI-1.0.0

[3] https://github.com/pooler/cpuminer/releases/tag/v2.5.0

[4] https://github.com/btccom/btcpool-ABANDONED/tree/0127a750b3b26a515e66261cbd9ad7f95c54de01

Por último, en la sección 2.2.1.2 se detallan los cambios al procesamiento de transacciones que se realizaron al programa. Además, se indican las motivaciones que llevaron a hacer dichas modificaciones.

2.2.1.1 Cambios en la red Regtest

En los experimentos se utilizará siempre la red *regtest*, que al ser privada y local, permite tener el ambiente controlado para los mismos.

Todos los cambios que se muestran seguidamente, fueron aplicados para ambos nodos Bitcoin de las redes de prueba. Es decir, todos los experimentos necesitaron de dos instancias del nodo Bitcoin funcionando y ambas fueron siempre generadas en base al mismo código fuente modificado.

Uno de los beneficios del uso de *regtest*, es que no hubo que hacer cambios en la aceptación de bloques como parte de la red. Esto debido a que en *regtest*, todos los bloques enviados, son aceptados por el cliente. Sin embargo, esto tendrá implicancias al momento de lograr que se cumpla el *target time* necesario para los experimentos. Dicho cometido, será logrado a través de cambios en el software de *pool* de minería y los mismos serán presentados en la sección 2.2.4.

Donde sí hubo que hacer cambios, es en el parámetro de consenso referido al *halving*. Para este parámetro no se uso el valor de *mainnet*, sino que se utilizó el valor 160 000. La razón detrás de ese número, se basa en que es lo suficientemente grande para poder minar bloques y generar una buena cantidad de bitcoins.

Luego, esos bitcoins obtenidos del minado, serán utilizados para generar las transacciones necesarias para llenar los bloques en los experimentos. En la sección 2.3.1, se mostrará el proceso para generar las transacciones.

Hay que remarcar que, como los nodos Bitcoin de la red son siempre ejecutados en base al mismo binario, no hay problemas de consenso y ambos aceptan o rechazan las mismas transacciones, como así también los mismos bloques.

Por último, 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. Estas validaciones fueron deshabilitadas, ya que no se efectúan cuando el cliente utiliza *mainnet*.

Habiendo realizado los cambios mencionados, se puede utilizar *regtest* para realizar experimentos, y obtener resultados, que serán también válidos para la red productiva.

2.2.1.2 Procesamiento de transacciones

Antes de entrar en detalle en los cambios, vale la pena recordar el concepto de *memory pool (mempool)*, la estructura interna que almacena todas las transacciones pendientes del nodo Bitcoin. De ahí se selecciona un subconjunto de transacciones cada vez que se solicita un nuevo trabajo de minado.

Para los experimentos a realizar, surge la necesidad de contar con distintos niveles de llenado de bloques con transacciones. Por citar algunos ejemplos, bloques llenos por completo o bloques llenos por la mitad.

A la vez, los experimentos realizarán el minado de 144 bloques; es la cantidad de bloques Bitcoin promedio, que se minan durante un día en la red *mainnet* y tiempo que durarán como máximo los experimentos.

Además, se usará un cliente que no recibirá transacciones mientras mina, siendo esto razón suficiente para tener que generar las mismas, antes de comenzar el experimento. Esto, con el fin de poder estudiar el

rendimiento de minado en aislamiento con respecto a otras funciones del cliente, como pueden ser procesar nuevas transacciones o bloques.

Por las razones expuestas anteriormente, el objetivo es llenar el *mempool* con las transacciones necesarias, para completar los 144 bloques. Hay que destacar que dicho número de transacciones que deberá manejar el nodo, es varias veces mayor a lo que manejó alguna vez en la red productiva.

Con la finalidad de cumplir con los requisitos anteriores, se adaptó el nodo de Bitcoin, para que permitiera la generación eficiente y el almacenamiento de una gran cantidad de transacciones. A continuación se presentan los cambios.

El nodo de Bitcoin, posee una limitación sobre la cantidad de transacciones que puede tener sin confirmar en el *mempool*, es decir, transacciones que dependen de otras que también se encuentran en el *mempool*. Remover tal validación, fue uno de los cambios realizados, para poder generar la cantidad de transacciones necesarias para las pruebas.

Otro de los cambios efectuados, fue desactivar ciertas validaciones realizadas por el nodo, al momento de recibir una nueva transacción e incluirla en el *mempool*. La generación de una gran cantidad de transacciones, es un proceso costoso que será explicado en detalle en la sección 2.3.1 y desactivar las validaciones contribuyó a reducir el costo. Desde el punto de vista de su correctitud, no hubo impacto alguno. De manera que todas las transacciones fueron generadas, para que cumplieran con las validaciones desactivadas.

Finalmente, se modificó el código fuente, para generar distintas versiones del nodo Bitcoin que devolvieran una cantidad fija de transacciones del *mempool*. Lo mencionado anteriormente, permitirá la generación de bloques llenos con distintas cantidades de transacciones, según lo requiera el experimento.

En esta sección se presentaron cambios para el nodo Bitcoin. Los referidos a parámetros de *Regtest* detallados en la sección 2.2.1.1, serán cambios comunes a todas las versiones del nodo que se utilizarán en los experimentos. Las modificaciones de validaciones mencionadas en la sección 2.2.1.2, también serán parte de todas las versiones.

Sin embargo, como los experimentos necesitarán de distintas cantidades de transacciones para llenar sus bloques, esto dará lugar a diferentes versiones del nodo Bitcoin. Retomando el ejemplo del principio de esta sección, existirá una versión del nodo de Bitcoin, que devuelva transacciones que permitan llenar bloques por la mitad, mientras que habrá otra, que devuelva transacciones para llenar bloques por completo.

Luego, cada experimento usará la versión del nodo Bitcoin que se adapte a sus requerimientos.

2.2.2 Cliente RSK

La red que se utilizará para los experimentos es *regtest*; es de carácter local, privada e ideal para pruebas.

Uno de sus beneficios es que la dificultad de minado es suficientemente baja, tal que permite ser minada con CPU; de hecho acepta bloques con cualquier valor de dificultad y no ajusta la misma en ningún momento.

Al igual que ocurrió con Bitcoin, el ajuste del ritmo de minado para RSK, será logrado a través de modificaciones en el software de *pool* de minería, las mismas se exhibirán en la sección 2.2.4.

El resto de los valores de *regtest*, son los mismos que *mainnet* o no requieren ser modificados, dado que no impactan en los experimentos a realizar.

Con respecto a las transacciones necesarias para realizar los experimentos, los mismos buscarán un caso de exigencia máxima de la red, por lo que se utilizarán bloques siempre completos.

Utilizar bloques completos implica la necesidad de contar con una gran cantidad de transacciones para llenar todos aquellos bloques que serán minados durante cada experimento.

Las transacciones serán generadas de antemano debido a que en los experimentos se desea tener al nodo solo realizando tareas de minado y no consumiendo recursos en otros trabajos, como recibir y procesar transacciones.

Para poder tener transacciones generadas de antemano, se modificó el código fuente del nodo de RSK.

Por un lado, el nodo RSK tiene un límite de tiempo en el que una transacción puede ser candidata a ser incluida en un bloque. Mientras la transacción esta pendiente de inclusión, la misma se encuentra en el *transaction pool*, una estructura que cumple en RSK la misma función que cumple el *mempool* en Bitcoin.

El límite de tiempo de permanencia de una transacción en el *transaction pool* fue removido, para permitir generar de antemano las transacciones.

Por otro lado, el *transaction pool* solo vive en memoria, por lo cual todas las transacciones generadas se pierden al terminar con la ejecución del nodo de RSK. Esto es algo que atenta contra la posibilidad de reproducir los experimentos y obligaría a crear las transacciones, previo a cada experimento. La creación de un gran número de transacciones es costosa.

La solución para evitar la situación anterior, fue modificar el código fuente. El cambio implicó el agregado de lógica para se guarden las transacciones del *transaction pool* al finalizar la ejecución del nodo. Con posterioridad y al encenderlo, también se hicieron modificaciones para leer las transacciones guardadas del *transaction pool*, para poder tenerlas disponibles para su uso.

De la mano del manejo de una gran cantidad de transacciones, surgió otro problema. Esto ocurrió porque, el nodo de RSK recorre linealmente y en varias oportunidades las transacciones del *transaction pool*, para poder armar un bloque candidato. A los fines de los experimentos que se quieren realizar, lo único que interesa es que los bloques estén llenos. Es por esto que se desactivó la lógica de selección de transacciones, que implicaba la recorrida lineal de las mismas, reemplazándola por una simple selección de las primeras transacciones del *transaction pool*, para así completar el bloque candidato.

Habiendo realizado los cambios y las consideraciones mencionados, el nodo de RSK *regtest* que contiene los mismos, es el que se utilizará para realizar experimentos.

2.2.3 Hardware de minería

En los experimentos de este trabajo, se optó por utilizar minado por CPU, a diferencia del minado por hardware especializado, que se emplea para minar la red *mainnet* de Bitcoin.

El minado por CPU se ejecuta en una computadora multi-propósito y la potencia de minado depende directamente de la capacidad de procesamiento de la máquina.

Con el fin de poder realizar los distintos experimentos, el software `CPU miner` fue usado para realizar minado por CPU y su implementación fue modificada.

Al código fuente del programa se le agregaron *logs* para:

- ***Hashing power.*** Cada 10 s se exhibe y registra el mismo en un archivo.

  El objeto de este cambio, es poder realizar una prueba de *benchmark* para determinar el hashing power de las máquinas, donde se correrán los experimentos.

  Además, contar con un registro del *hashing power* permitirá utilizarlo para controlar el correcto funcionamiento de los experimentos. Más detalles al respecto se incluirán en futuras secciones de este capítulo.

- ***Shares.*** Cada vez que una *share* es encontrada por el minero, la misma se registra en un archivo.

Por último, se realizaron cambios para siempre utilizar todos los cores del CPU al momento de minar y para que el proceso de minado se ejecutara con máxima prioridad, frente a otros procesos.

Los cambios mencionados fueron aplicados y dieron como resultado una versión nueva del software `CPU miner`, la misma es la que se empleará en los experimentos.

2.2.4 Software de pool de minería

La pieza de software que sirve de nexo entre todos los componentes presentados hasta el momento, es el *pool* de minería. `BTCPool` será el utilizado para todas las pruebas y para eso se llevaron a cabo distintas modificaciones.

Los cambios necesarios para que el *pool* realizara *merged mining*, con Bitcoin como cadena primaria y RSK como cadena secundaria, se implementaron previo a este trabajo. Los mismos fueron contribuidos a la versión de código abierto del software y luego mantenidos por la comunidad.

Como parte de tales modificaciones, se incluyó un modo de desarrollo que permite establecer, a través de archivo de configuración, la *pool difficulty* a utilizar.

Dicho modo también incluye cambios a nivel código fuente, para que el valor de *pool difficulty* sea leído y luego aplicado en las distintas secciones de código, para logar el comportamiento deseado.

La posibilidad de establecer la *pool difficulty* es relevante, dado que es el valor que utiliza el *pool*, para cumplir con el *target time* promedio deseado. A la vez, el valor de *pool difficulty* depende del poder de cómputo disponible.

Por ejemplo, la *pool difficulty* para obtener *shares* cada 10 s, será una, si se mina con CPU y otra mayor si se mina con ASIC, debido al mayor poder de cómputo de un ASIC.

Gracias al modo de desarrollo, se puede configurar el software de *pool* para tener la frecuencia de *shares* deseada.

Este cambio es especialmente útil para poder realizar minado con CPU, ya que el software de *pool*, no considera valores de *pool difficulty* tan bajos como para minar con el poder de cómputo de tal tipo de máquina.

Esto se debe a que hoy en día, no se hace minado por CPU en entornos productivos. El mismo fue reemplazado por minado con ASIC y el software de minería contempla eso.

Hay que mencionar también, que en el modo de desarrollo se desactivó el ajuste de dificultad para la *pool difficulty*. El impacto resultante es que si se agrega o quita poder de cómputo, el *pool* no hará nada para mantener el *target time*, a menos que se modifique el valor en el archivo de configuración. De todas maneras, esto no representa una limitación para los experimentos a realizar, puesto que los mismos se harán con un *hashing power* estable, a lo largo de toda su duración.

Es importante destacar que tener control sobre la frecuencia de las *shares*, no provee de control sobre la frecuencia de minado de Bitcoin o de RSK. En un ambiente productivo, dichas frecuencias son reguladas a través de la dificultad y su realización, es responsabilidad de los nodos.

Tal como se explicó en las secciones anteriores, los nodos de Bitcoin y de RSK en *regtest* tienen una dificultad tal que aceptan bloques de cualquier dificultad. Es por esta razón, que se decidió realizar el control de la frecuencia de minado de Bitcoin y de RSK, desde el *pool* de minería.

Para lograr dicho objetivo, en el marco de este trabajo, se agregaron dos configuraciones extras en el archivo de configuración del componente `sserver`; una para la dificultad deseada para Bitcoin y otra para la dificultad deseada para RSK. Ambas son solo válidas, cuando el *pool* es ejecutado en modo de desarrollo.

Es por esto que ahora, cada vez que el *pool* de minería deba decidir si una *share* cumple con la prueba de trabajo, lo que hará es compararla con los valores de configuración y no con los provistos por los nodos

de Bitcoin y RSK.

En particular, los cambios de modo de desarrollo presentados hasta el momento no implican trabajo extra, durante el proceso de minado para el software de *pool*. Esto se debe a que la lectura de los valores de configuración, se hace al momento de iniciar el *pool* y en esa etapa el mismo no esta disponible para minar. Luego, las comparaciones necesarias para determinar si se cumple con la prueba de trabajo, eran algo que el *pool* ya realizaba como parte del proceso de minado. La única diferencia es que ahora lo hará, utilizando un valor obtenido desde otro lugar, el archivo de configuración.

Además de los cambios mencionados hasta el momento, se agregaron algunos *logs* adicionales para usar en conjunción con los ya presentes, en el software de *pool* versión productiva. El objetivo de esto, fue el de poder monitorear mejor algunos eventos del *pool*.

Para todos los experimentos de este trabajo, se utilizará la versión del *pool* que cuenta con modo de desarrollo y *logs* adicionales. Es relevante aclarar que, al hablar del software de *pool*, se hace referencia al conjunto de todos los microservicios que lo componen.

2.3 Generación de datos

En esta sección se explicará cómo se generaron los datos necesarios para correr los experimentos. El enfoque para mostrar esto será por dato generado; los componentes de software se vincularán luego, a los conjuntos de datos que los involucren.

2.3.1 Transacciones

Tal como se mencionó en secciones anteriores, existe la necesidad de generar transacciones de antemano, tanto para la red de Bitcoin como para la red de RSK.

En el caso de Bitcoin, se necesita tener el *mempool* con suficientes transacciones para completar 144 bloques.

Tal como se presentó en la introducción de esta tesis, el llenado de un bloque no está limitado por el número de transacciones, sino por el tamaño que las mismas ocupan.

Existen muchas maneras de generar transacciones en la red de Bitcoin. Para poder hacerlo, se buscó un balance entre aquellas que permitieran llenar los bloques de forma rápida y que a la vez representen transacciones reales, que ocurren en la red productiva.

A la vez, si bien se desactivaron la mayor cantidad de validaciones posibles en el nodo Bitcoin, no fue viable evitar que cada vez que se recibe una transacción, el procesamiento ocurra de manera completamente serial. Es decir, no es posible comenzar a procesar una nueva, hasta no haber terminado con la anterior. Esta validación requiere tiempo lineal, con respecto a la cantidad de transacciones, consumiendo la mayor cantidad del tiempo del proceso de generación de una transacción nueva.

Por lo expuesto anteriormente, se encontró que utilizar transacciones con diez *outputs*, permite cumplir con tener transacciones reales y a la vez ocupar espacio suficiente, para llenar el bloque con un número razonable de las mismas. Gracias a esto, cada bloque lleno tendrá aproximadamente 2200 transacciones.

Además, para evitar consumir tiempo en la validación lineal, se encontró que lo ideal es que las transacciones no se relacionen entre ellas. Esto se logra utilizando siempre direcciones nuevas en los *outputs* de las transacciones.

Para poder generar el total de 320 000 transacciones (2200 transacciones para 144 bloques y algunas extra) en el *mempool*, se realizó un programa que funcionó de la siguiente forma:

1. **Mina** 160 000 **bloques.** Es aquí donde es importante haber modificado el parámetro de *halving*. De no haberlo hecho, la última tanda de bloques no generaría ningún tipo de recompensa.

   Esta modificación no tiene ningún impacto sobre el minado.

2. **Genera direcciones destino.** La cantidad necesaria es de 1 440 000.

3. **Arma la transacción a enviar:**

   - Elige un *output* no usado, que tenga bitcoins recientemente generados como parte de minar uno de los 160 000 bloques.
   - Utiliza el *output* como *input* de la nueva transacción.
   - Selecciona nueve direcciones no usadas y junto con la dirección de donde se esta enviando el dinero, obtiene las diez direcciones necesarias.
   - Genera los diez *outputs* con cada una de las direcciones y un monto de bitcoins.

4. **Envía la transacción.** Realizado a través del nodo Bitcoin al que está el programa se encuentra conectado.

5. **Repite los pasos** 3 **y** 4**.** Para generar la totalidad de las transacciones.

Finalmente, la ejecución de varias instancias del mismo programa en paralelo, cada una conectada a distintos nodos Bitcoin pertenecientes a la misma red, es lo que permitió obtener las distintas bases de transacciones a utilizar en los experimentos.

En el caso de RSK, se necesita contar con suficientes transacciones en el *transaction pool*, para tener bloques llenos durante 144 bloques Bitcoin, cantidad de bloques que durarán de los experimentos.

Se deben llenar 2880 bloques, dado que el *target time* es de 30 s para RSK. El llenado de un bloque en RSK no depende directamente del tamaño de las transacciones, sino que depende del Gas que las mismas consuman. El límite de Gas de un bloque es de 6 800 000.

En cuanto al tipo de transacciones, se busca que la red de RSK tenga un tráfico similar al que tiene una red de contratos inteligentes, en la vida real. Para esto se caracterizaron los tipos de transacciones que pueden presentarse y los mismos son: transferencia de moneda nativa, creación de contrato inteligente e interacción con un contrato inteligente.

Luego, se estudió la distribución de transacciones en una red de contratos inteligentes análoga a RSK, pero con alto nivel de tráfico, como es Ethereum. Dicho estudio consistió en tomar todas las transacciones de un día aleatorio de la red de Ethereum y contar la cantidad de apariciones de cada uno de los tipos.

Hasta el momento, se conoce la cantidad de Gas que tiene que tener un bloque para ser llenado y el total de bloques a llenar con transacciones. Además, se sabe la distribución de los tipos de transacciones, que el mismo deber tener.

Utilizando esta información, se desarrolló un programa para generar las transacciones necesarias en el *transaction pool*. La lógica detrás del mismo fue:

1. **Despliega un contrato inteligente.** Condición necesaria dado que hay que generar transacciones que interactúen con un contrato inteligente.

   Para esto se programó un contrato de los más populares en la red de Ethereum, que permite generar una moneda y realizar transferencias entre cuentas de la misma.

2. **Obtiene direcciones destino.** El nodo de RSK en *regtest* tiene algunas direcciones ya generadas, se utilizarán las mismas.

   A diferencia de Bitcoin, la dependencia entre las direcciones usadas en las transacciones, no genera problemas de rendimiento.

3. **Arma las transacciones de moneda nativa a enviar:**

   - Elige un dirección de destino y monto.
   - Genera la transacción, utilizando como cuenta de envío una que ya tiene RBTC, cortesía de utilizar *regtest*.
   - Repite los pasos previos, hasta tener la cantidad necesaria a incluir en el bloque.

4. **Arma las transacciones de interacción con el contrato inteligente:**

   - Elige un dirección de destino y monto.
   - Genera la transacción, utilizando como cuenta de envío una que ya tiene RBTC, cortesía de utilizar *regtest*. La cuenta de destino en este caso es la del contrato inteligente.
   - Repite los pasos previos, hasta tener la cantidad necesaria a incluir en el bloque.

5. **Envía las transacciones.** Válido para las generadas en los pasos 2 y 3. Realizado a través del nodo RSK, al que el programa se encuentra conectado.

6. **Repite los pasos** 2**,** 3 **y** 4**.** Esto permite generar la totalidad de las transacciones.

2.3.2 Hashing power

En los experimentos a realizar se necesita poder contar con distintos valores de *target time*, principalmente para la red de Bitcoin y, en menor medida, para la red de RSK. La forma de regular la cantidad de trabajo generado por el minero es a través de la dificultad.

Es entonces la dificultad, el valor a establecer para poder cumplir con un *target time* deseado. La forma que ofrece `BTCPool` para fijar dicha dificultad, es por medio de establecer los valores de dificultad para Bitcoin y para RSK en su archivo de configuración.

Para poder conocer el valor de dificultad a utilizar, es necesario saber el *hashing power* con el que se cuenta. Esto debido a que únicamente con *hashing power* y dificultad, se puede conocer la cantidad de trabajo generado. Lo anterior se refiere a cuantas soluciones de prueba de trabajo se obtendrán.

En consecuencia, conociendo el *hashing power* y la cantidad de trabajo que se desea generar, representada por el *target time*, se puede conocer la dificultad que satisface la ecuación.

En el caso actual, se conocen los *target time* y se quiere averiguar la dificultad. Se desconoce el *hashing power*, pero es posible averiguarlo.

A continuación se presenta un experimento llamado *benchmark*, que se realizó en la máquina B para poder conocer su *hashing power*.

El `CPU miner` posee un modo para minar sin necesidad de estar conectado a un software de *pool* de minería. El objetivo de dicho modo es poder determinar la cantidad de hashes por segundo que puede producir. Para esto, genera trabajo de forma interna y comienza a buscar soluciones.

En el experimento de *benchmark* se utilizó el `CPU miner`, en el modo descripto anteriormente y por una duración de tres días. Utilizando el *log* agregado para conocer el *hashing power* (sección 2.2.3), se fue registrando el mismo cada 10 s. Finalizado el experimento, se procesaron dichos valores y se obtuvo que

el *hashing power* para la máquina B es de 24,205 MH/s. La desviación estándar de dicho experimento fue menor al 1 %.

Conociendo el *hashing power* y los *target time* deseados, se realizó un programa que, tomando ambos valores, calcula la dificultad.

Finalmente, ejecutando ese programa por cada valor de *target time*, se obtuvieron los valores de dificultad a utilizar en los experimentos.

2.4 Configuraciones

En esta sección se presentarán las configuraciones realizadas a cada uno de los componentes, para poder formar la red de pruebas y utilizarla en los experimentos.

Tal como se presentó en la sección 2.1, existen dos redes de prueba, la red de minado Bitcoin y la red de minado *merged mining*. La red de *merged mining* es más completa, al tener el nodo de RSK junto con los mismos componentes que la de Bitcoin; por eso será la utilizada para presentar las configuraciones.

Comenzando con el nodo de Bitcoin, se necesitan dos instancias, como mínimo, para que funcione el sistema correctamente. Cada una de las instancias debe estar conectada a la otra, formando así una red. Para que esto se posible, se debe configurar el par dirección `IP` - puerto, en la que cada nodo estará escuchando por conexiones entrantes. Un par es necesario para el *wire protocol*, mientras que otro par lo es para la interfaz `RPC`. Luego, cada uno de los nodos debe incluir en su configuración, el par referido al *wire protocol* para así poder intercambiar mensajes. Lo anteriormente presentado se configura en un archivo de configuración, el cual se referencia en el comando utilizado para iniciar la instancia.

Desde el punto de vista de dependencias para su funcionamiento, el nodo Bitcoin necesita de pocas bibliotecas de código. Todas deben ser instaladas para poder lograr su correcto funcionamiento.

Siguiendo con el nodo de RSK, el mismo solo requiere de la instalación de `Java` versión 8, para funcionar. El resto de las bibliotecas de código que necesita para funcionar, están incluidas en el binario que se usa para ejecutarlo.

En cuanto a las configuraciones de archivo, se utilizaron las que vienen por defecto para la red *regtest*. Incluso se respetó el mismo puerto donde expone su interfaz `RPC`. A diferencia de Bitcoin, solo una instancia del nodo de RSK, es necesaria para su correcto funcionamiento de minado.

Existen otras funcionalidades que provee el nodo y que no fueron configuradas. Es debido a que a los fines de este trabajo, solo se necesita que funcione el minado y no otras cosas, como por ejemplo el *bridge* con Bitcoin.

Por último, se modificó el archivo de configuración específico de control de los eventos que registra el nodo. Gracias a esto, se va a poder contar con el registro de todos los eventos necesarios, para controlar el correcto funcionamiento del componente.

Continuando con el `CPU miner`, se configuró para utilizar todos los *cores* disponibles en la máquina B. Misma configuración, había sido utilizada al momento de realizar el experimento de *benchmark*.

A su vez la máquina B, fue configurada de tal manera que no tuviera `Hyperthreading` ni `Turbo Boost` activados. También el procesador fue configurado, para funcionar con `governance` en `Performance`. Tales configuraciones aseguran que el procesador, esté en funcionamiento pleno, siempre que el `CPU miner` lo necesite.

Finalmente, en el archivo de configuración del minero, se configuró la dirección `IP` y el puerto del software del *pool* de minería, para que puedan comunicarse.

Por último, el componente `BTCPool` fue el que más trabajo de configuración requirió.

El mismo cuenta con varios requerimientos para poder funcionar. Entre ellos se encuentran las instalaciones de:

- MySQL, junto con la generación de todas las tablas necesarias.
- Apache Kafka, junto con la creación de todas las colas de mensajes que utilizan los microservicios del *pool*, para poder comunicarse.
- Una API de clientes autorizados para minar. En este caso se programó una que siempre retornara el mismo cliente, la instancia del CPU miner empleada para minar.
- Múltiples bibliotecas de código.

Desde el aspecto de configuración, cada microservicio cuenta con su archivo propio por lo cual se debió configurar uno a uno. A continuación se mencionan los valores que fueron cambiados, respecto de los que vienen por defecto en dichos archivos de configuración:

- ***Get work maker (gwmaker).*** Se configuró la dirección IP y el puerto de la interfaz de RPC del nodo de RSK. El tiempo de polling fue establecido en 2000 ms.
- ***Get block template maker (gbtmaker).*** Se configuró la dirección IP y el puerto de la interfaz de RPC del nodo de Bitcoin. El tiempo de polling fue establecido en 5 s.
- ***Job maker (jobmaker).*** Es el que cuenta con la configuración de *notify policy*. A la misma se le asignó un valor de 2, es decir, los mineros reciben una actualización de trabajo cuando hay uno nuevo RSK. Implica además, que deben abandonar el trabajo que estaban realizando, para comenzar con el nuevo de inmediato.
- ***Stratum server (sserver).*** La interfaz que se expone para conexión de los mineros es responsabilidad de este componente. Se configuró la dirección IP y el puerto tal que coincida con los establecidos en el archivo de configuración del CPU miner.

  Además, es en este componente donde se activó el modo de desarrollo y donde, según la necesidad de cada experimento, se configurarán las diferentes dificultades.

  Si bien no tiene impacto alguno en lo que se busca estudiar en los experimentos, se tuvo que configurar la conexión a la base de datos MySQL, para el registro de las *shares* generadas por los mineros.
- ***Block maker (blkmaker).*** Se configuró la dirección IP y el puerto de la interfaz de RPC del nodo de Bitcoin, para que pudiera enviar bloques cada vez que se encuentre una prueba de trabajo.

  La misma configuración, pero con la IP y puertos correspondientes, se llevó a cabo para el nodo de RSK.

  Finalmente, se tuvo que configurar la conexión a la base de datos MySQL, para el registro de las soluciones a los bloques candidatos encontradas por cada minero. Esto último no presenta relevancia para los experimentos a realizar pero es necesario para el correcto funcionamiento de este componente.

A la vez, hay que notar que todas las versiones de los componentes utilizadas, fueron compiladas especialmente para los experimentos.

Las configuraciones explicadas hasta el momento, permiten tener a todos los componentes de la red de minado *merged mining* comunicados entre sí y listos para ser ejecutados.

Para poder iniciar la red, solo hay que ejecutar cada uno de los componentes. Se deben ejecutar primero los nodos de Bitcoin y RSK. Ambos demoran unos minutos en estar listos, debido a que deben cargar una gran cantidad de transacciones en sus *pools* de transacciones pendientes. Una vez terminada dicha carga, se puede iniciar el componente `BTCPool` y finalmente conectar el minero al mismo, para comenzar a minar.

2.5 Supervisión de funcionamiento

Hasta el momento se presentó todo lo necesario para poder tener las redes de pruebas funcionando. Antes de poder utilizarlas en los experimentos se necesita validar, que además de funcionar, lo hagan correctamente.

Para esto se comprobó que el minado, tanto de Bitcoin como de RSK, generara la cantidad de bloques esperada, de acuerdo al *target time* configurado.

A continuación se presentará un detalle de los puntos verificados para una ejecución de 144 bloques de minado Bitcoin, con *merged mining*:

- **Generación de *shares*.** Este proceso se inicia cuando el `CPU miner`, luego de haber recibido el trabajo del `sserver`, busca soluciones para la prueba de trabajo. El mismo se repite tantas veces como trabajos existan, a lo largo de todo el experimento.

  Una vez que una solución es encontrada, el `CPU miner` registra el evento y la enía al `sserver`. A su vez, el `sserver` registra la recepción de *shares* junto con detalles de su procesamiento.

  La validación realizada, constó en verificar que por cada *share* enviada existiera una recibida.

- **Generación de bloques Bitcoin.** Cada vez que `BTCPool` recibe una *share*, si la misma cumple con la dificultad de Bitcoin, se construye un bloque y se envía al nodo de Bitcoin.

  Dicho envío queda registrado en los *logs* del microservicio `blkmaker` y al recibir un bloque, el nodo Bitcoin registra ese evento en su *log*.

  Al finalizar el experimento, se analizan ambos archivos de *log* para evaluar que los eventos coincidan.

  Además, se consulta al nodo de Bitcoin, a través de su interfaz `RPC`, la cantidad de bloques actuales. Se valida que esta cantidad sea igual a la obtenida en los *logs*. También se verifica que la cantidad de bloques minados sea un número cercano a los 144 bloques esperados y que la diferencia entre ellos, concuerde con el *target time* deseado.

- **Generación de bloques RSK.** Este caso es muy similar al de Bitcoin explicado en el punto anterior.

  Se realizan las mismas verificaciones, solo que utilizando el log del nodo RSK para verificar los bloques minados y la interfaz `RPC` del nodo RSK, para averiguar la cantidad de bloques minados.

- ***Hashing power*.** Es importante que el mismo se mantenga constante a lo largo de todo el experimento.

  Para verificar lo anterior, se realizó lo mismo que en el experimento de *benchmark* mencionado en la sección 2.3.2. Es decir, promediar los valores de *hashing power* informados por el `CPU miner`. Con el objeto de completar la verificación, además se compara el resultado promedio con el valor de *hashing power* esperado para la máquina B.

En la práctica, lograr que el entorno de pruebas funcione correctamente, llevó varios intentos, pero eventualmente se logró que todas las validaciones pasen de manera satisfactoria.

Es importante destacar, que además de las validaciones anteriores que apuntaban a la corrección del proceso de minado, se llevaron a cabo verificaciones a nivel ejecución de los componentes. Esto quiere decir, que periódicamente se comprobaba que cada uno de los componentes, estuviera funcionando. Lo anterior, también vale para las dependencias de los componentes, como puede ser el caso de `Apache Kafka`.

En este apartado, se presentaron las verificaciones realizadas al momento de ejecutar el entorno de prueba, para confirmar su correcto funcionamiento.

En el final de este capítulo, se mostrará cómo dichas validaciones fueron utilizadas, como parte del control de cada uno de los experimentos de este trabajo.

2.6 Ejecución automatizada

Hasta el momento, cada vez que se necesitaba llevar a cabo un experimento, se debía realizar la configuración y ejecución de cada uno de los componentes del entorno de pruebas, de manera individual.

Dada la gran cantidad de experimentos a correr, se automatizó dicho proceso. Para esto se creó una versión base de cada uno de los componentes, es decir, un conjunto de ejecutable, archivos de configuración y datos necesarios. Partiendo de eso, se escribió un programa para que cada componente, cumpliera las siguientes tareas:

1. Genera un nuevo directorio con la versión base del componente.
2. Reemplaza en el archivo de configuración los valores para el experimento que se desea realizar.
3. Copia al directorio los datos necesarios. Por ejemplo, la base de datos de transacciones, para el caso que el componente sea el nodo Bitcoin.
4. Inicializa la ejecución del componente para el tiempo de duración del experimento.
5. Registra cualquier tipo de errores en su ejecución en un archivo.
6. Repite los pasos del 1 al 4, tantas veces como ejecuciones del experimento se deseen realizar.

Lo mostrado anteriormente, es una descripción general del programa que permite ejecutar los componentes. Para la ejecución de cada uno en particular, solo se necesita instanciar al programa con los valores de configuración y datos correspondientes al componente de turno.

Finalmente, llevar a cabo el experimento, implica iniciar tantos programas como componentes el mismo requiera.

Hay que enfatizar que adicionalmente a la ejecución automatizada descripta, para cada experimento se realizarán las verificaciones expuestas en la sección 2.5.

En este capítulo se presentó la información necesaria para poder llevar a cabo los experimentos.

En el capítulo siguiente, se introducen los experimentos junto con un análisis de sus resultados.

Capítulo 3

Resultados

En el presente capítulo se exhibirán los experimentos y los resultados obtenidos luego de realizarlos en los entornos de prueba introducidos en la sección de metodología de este trabajo. Acompañará a los resultados, un detallado análisis de los mismos, sumado a información pertinente, sobre la forma en que estos fueron obtenidos y procesados.

Primero, en las sección 3.1 se presentarán los resultados de generación o minado de bloques Bitcoin. Esto será para escenarios, donde se busca evaluar el rendimiento del minado Bitcoin con *merged mining*, al variar la cantidad de transacciones incluidas en los bloques Bitcoin.

En la sección 3.2 se continuará con el estudio del rendimiento del minado Bitcoin con *merged mining*, utilizando resultados de minado de bloques Bitcoin, pero en este caso los escenarios a estudiar se caracterizan por variar el *target time* de Bitcoin.

Cubiertos los casos de estudio de rendimiento del minado Bitcoin con *merged mining*, al variar valores de funcionamiento de la red Bitcoin, se procederá a exhibir resultados para el estudio de dicho rendimiento, en un contexto de modificación de valores de la red de RSK. Para lograr lo expuesto previamente, la sección 3.3 presentará los resultados de generación de bloques Bitcoin vinculados a variar el *target time* de RSK, para valores fijos de *target time* de Bitcoin.

Finalmente, la sección 3.4 no se concentrará en el análisis del rendimiento del minado Bitcoin bajo *merged mining* sino que hará foco en exhibir los resultados para el rendimiento del minado de RSK. Lo anterior se hará valiéndose de los bloques minados para RSK, en condiciones tanto productivas como más exigentes, que las de un entorno productivo.

3.1 Rendimiento de la generación de bloques Bitcoin variando la cantidad de transacciones

Los resultados que se expondrán en esta sección representan distintos escenarios de minado con y sin *merged mining*, donde varía la cantidad de transacciones en los bloques de Bitcoin.

En dichos escenarios, los bloques Bitcoin se utilizaron sin transacciones o vacíos, llenos con 25 % de transacciones, llenos con 50 % de transacciones y llenos con 100 % de transacciones o completos. Para el caso de RSK, se utiliza el peor caso. En consecuencia, los bloques de dicha red estarán llenos para todos los escenarios.

El *target time* utilizado, fue de 6 min entre bloques Bitcoin y de 30 s entre bloques RSK, para todos los experimentos. Los bloques totales a minar por experimento fueron 144; este número representa la cantidad de bloques que se espera minar en promedio, en un día en la red mainnet de Bitcoin. Dichas configuracio-

nes de *target time* y cantidad total de bloques, ambas para Bitcoin, hicieron que la duración total de cada experimento sea de 864 min.

Cada experimento, se repitió diez veces para proveer de solidez estadística a los resultados. Como consecuencia, se tienen diez repeticiones para cada bloque por tipo de minado (con o sin *merged mining*) y para cada escenario de llenado de bloques.

3.1.1 Análisis inicial

En esta sección se analizarán los resultados para los diferentes escenarios de llenado de bloques con transacciones Bitcoin. Como los casos de llenado produjeron resultados muy similares, se procederá a estudiar a continuación los más relevantes, es decir, bloques vacíos y bloques llenos. Para los casos de bloques llenos al 25 % y bloques llenos al 50 %, se exhiben los resultados en el anexo A.1.

Para comenzar, se muestra un análisis de la diferencia en segundos del minado de cada bloque, con respecto al *target time* esperado para el caso de bloques vacíos con y sin *merged mining*.

Es importante destacar que al contar con diez repeticiones del experimento para el escenario sin *merged mining* y diez repeticiones para el escenario con *merged mining* se realizó el promedio de los resultados de ellas por escenario. Es decir, para cada bloque de cada repetición y de cada escenario, se promediaron los diez tiempos de minado. Así se obtuvo un único tiempo denominado resultado o tiempo promedio.

Para obtener los resultados, se utilizó el archivo de *log* del componente `blkmaker` del software de *pool*. Dicho archivo, registra los eventos más importantes llevados a cabo por el software, desde el momento en que recibe una solución candidata, hasta que construye y envía el nuevo bloque al nodo de Bitcoin. También lo envía al nodo de RSK, en caso de estar configurado para *merged mining*.

El evento que registra el envío de un nuevo bloque al nodo de Bitcoin, fue el utilizado para obtener los tiempos de minado de los bloques de cada repetición de los experimentos. Cabe mencionar que el número total de eventos de generación de un nuevo bloque registrados por el software de *pool*, coincide con los bloques minados reportados por el nodo Bitcoin.

Los resultados fueron procesados en primera instancia por un programa que interpreta el *log* del software de *pool*, extrae la información sobre los bloques minados y la graba en un único archivo que agrupa las diez repeticiones de cada escenario para cada experimento. Un segundo programa se encargó de leer cada archivo con los datos agrupados, para luego realizar la diferencia con el *target time*, los promedios necesarios y finalmente graficar los resultados.

Iniciando el análisis con la comparación de las figuras 3.1(a) y 3.1(b) para bloques vacíos Bitcoin, se observa que los tiempos promedio de minado para cada bloque no son distinguibles.

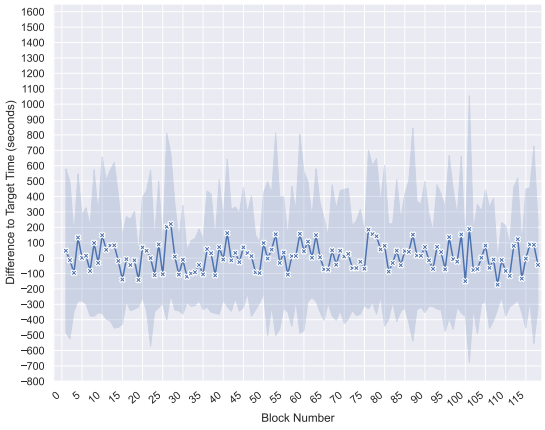
Las figuras presentan algunos picos grandes que corresponden a los bloques minados con mayor diferencia respecto del *target time* esperado. Sin embargo, en este análisis inicial no se ve que alguna figura presente más picos que la otra. Para el resto de los tiempos de minado de los bloques, los picos son más chicos y pueden ser considerados como el comportamiento esperado de la función de minado.

Vale recordar, que si bien el minado tiene un *target time* de 6 min, el proceso de generación de hashes para obtener una solución válida para cada bloque, es aleatorio. Esto provoca que ese *target time* de 6 min no sea exactamente alcanzado en cada bloque, sino que ocurra en promedio.

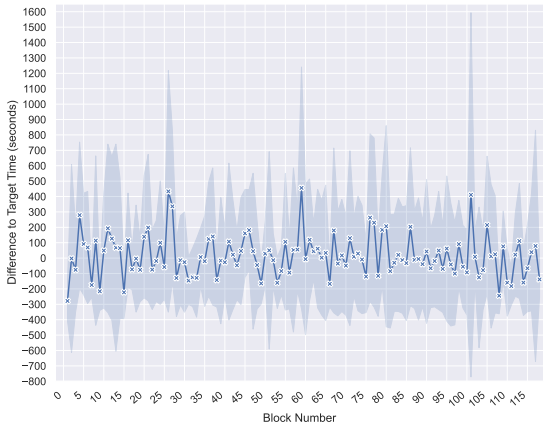
Luego del estudio anterior, es correcto afirmar que no se distingue diferencia entre los escenarios para los resultados promedio estudiados y que no se aprecia impacto del *merged mining*, en el tiempo de minado de bloques Bitcoin.

Las figuras 3.2(a) y 3.2(b) para bloques llenos, muestran que al igual que para bloques vacíos, los tiempos promedio de minado para cada bloque, no son distinguibles.

Un análisis de los picos sobre el promedio de tiempo de cada bloque, permite arribar a conclusiones similares al caso de bloques vacíos. En este caso podemos agregar, que no es perceptible una diferencia

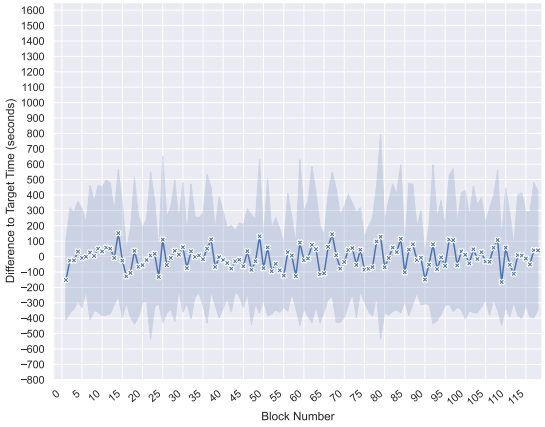


(a) Sin *merged mining*

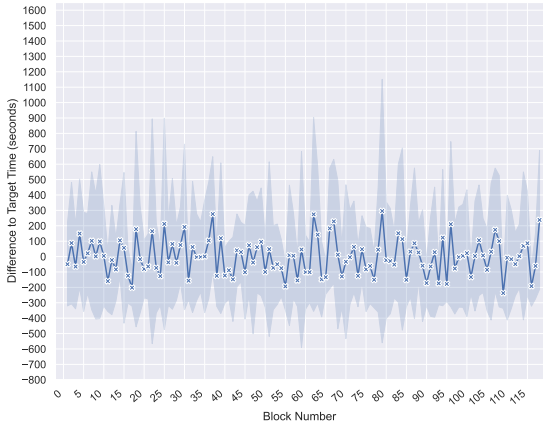


(b) Con *merged mining*

Figura 3.1: Escenario de minado de bloques Bitcoin vacíos.
Análisis de la diferencia entre el tiempo promedio de minado por bloque y su *target time*. Se incluye el desvío estándar alrededor del promedio. Cuando no hay diferencia en los tiempos, indica que el bloque fue minado en el *target time* esperado (6 $\mathrm{mín}$). Una diferencia negativa, indica que el bloque fue minado con anterioridad al *target time* y una positiva indica que el bloque fue minado con posterioridad.



(a) Sin *merged mining*



(b) Con *merged mining*

Figura 3.2: Escenario de minado de bloques Bitcoin completos.
Análisis de la diferencia entre el tiempo promedio de minado por bloque y su *target time*. Para tener un mejor entendimiento de la dispersión de los datos se graficó el desvío estándar alrededor del promedio. Cuando no hay diferencia en los tiempos, indica que el bloque fue minado en el *target time* esperado (6 $\mathrm{mín}$). Una diferencia negativa, indica que el bloque fue minado con anterioridad al *target time* y una positiva indica que el bloque fue minado con posterioridad al *target time*

en el desvío estándar para ambas muestras graficadas, por lo que la comparación de las mismas es válida. El análisis para el desvío estándar, puede hacerse extensivo al caso de bloques vacíos donde tampoco se percibe discrepancia.

Para el caso de bloques llenos, se puede concluir que no hay impacto visible del *merged mining* en el tiempo de minado de bloques Bitcoin.

Finalmente, una comparación de las figuras 3.1(a), 3.1(b), 3.2(a) y 3.2(b) en su conjunto, y utilizando los mismos criterios de análisis que al compararlas de a pares, indica que no se observa entre ellas ninguna anomalía o indicio que permitan distinguir los tiempos promedio de minado de los bloques exhibidos y tampoco su desvío estándar.

Basado en lo expuesto previamente, el análisis inicial permite concluir que el minado Bitcon no se ve impactado por el *merged mining* y que esta conclusión es válida, independientemente de la cantidad de transacciones Bitcoin que tengan los bloques.

3.1.2 Análisis estadístico

En esta sección se utilizarán métodos estadísticos para realizar un estudio más profundo de los resultados obtenidos en el análisis inicial 3.1.1 sobre el rendimiento de la generación de bloques Bitcoin.

Primero se presentará el test estadístico a utilizar y luego se analizarán los escenarios donde los bloques Bitcoin se encuentren vacíos, llenos con 25 % de transacciones, llenos con 50 % de transacciones y completos con un 100 % de transacciones. Dichos escenarios de llenado de bloques para minado con y sin *merged mining*, son los mismos que se estudiaron en la sección de análisis inicial, por lo que aquí se utilizarán los resultados de aquellos experimentos.

Para el análisis de los resultados se empleará la estadística inferencial. La misma se refiere a una parte de la estadística que comprende los métodos y procedimientos que, por medio de la inducción, determinan propiedades de un conjunto de datos.

La estadística inferencial presenta dos ramas de procedimientos estadísticos y de decisión, la estadística paramétrica y la no paramétrica [4]. Ambas permiten realizar distintos tipos de análisis, entre ellos se encuentran pruebas para comparar dos conjuntos de datos relacionados, pruebas para comparar dos conjuntos de datos no relacionados, pruebas para comparar tres o más conjuntos de datos relacionados, pruebas para comparar tres o más conjuntos de datos relacionados no relacionados, pruebas para comparar conjuntos de datos categorizados, entre otros. También existen casos en los que no existe un equivalente de prueba entre ambos tipos de estadística. Dichos casos no serán introducidos ya que no serán necesarios para este trabajo.

La estadística paramétrica es ampliamente utilizada en el ámbito científico y se vale de asunciones sobre características del conjunto de datos a analizar, conocidas como parámetros. En particular, las asunciones sobre los parámetros de la estadística paramétrica requieren que:

- los datos provengan de observaciones independientes,
- los datos tengan un tamaño suficientemente grande, la mayoría de los investigadores asume grande como mayor a 30,
- los datos pertenezcan a un intervalo,
- se conozca el tipo de distribución que tienen los datos,
- la distribución de los datos se aproxime a una distribución normal.

Para el caso de los escenarios a estudiar en esta tesis, se desconoce totalmente qué distribución tienen los datos. Es por esto que una de las condiciones que debe cumplir el conjunto de datos, para poder ser sometidos a pruebas de estadística paramétrica, ya no se cumple.

Para este tipo de casos existen algunas alternativas tal que se pueda utilizar la estadística paramétrica. Por ejemplo, en casos donde el número de datos no es suficiente, se pueden sumar mediciones para

alcanzar el mínimo de datos deseado. Otro caso es aplicar transformaciones a los datos tal que terminen teniendo una distribución normal. Desgraciadamente no siempre es posible realizar estos cambios.

En particular, para el conjunto de datos de esta tesis, no es la mejor opción transformar los datos para que cumplan con las condiciones de la estadística paramétrica. Esto debido a que es, como mínimo, un proceso complejo y que se presta a generar alteraciones, tal como datos distorsionados.

Como se exhibipreviamente, la estadística paramétrica requiere condiciones sobre el conjunto de datos, en cambio, la familia de pruebas que ofrece la estadística no paramétrica no posee dichos requerimientos.

Es por lo anterior, que se utilizará la estadística no paramétrica por sobre la paramétrica. La estadística no paramétrica suele emplearse para conocer la distribución de un conjunto de datos por comparación. Para lograr dicho cometido, se compara el conjunto de datos del que se quiere conocer la distribución con otro que contiene una distribución conocida. De lo anterior se desprende que las pruebas disponibles en la estadística no paramétrica pueden utilizarse para determinar si dos conjuntos de datos tienen la misma distribución.

En el caso de los datos a analizar en este trabajo, no es necesario conocer la distribución de los mismos. Entonces, la estadística no paramétrica será utilizada para determinar si los datos son distinguibles, a través de la comparación de su distribución.

Tal como se listó anteriormente, existen múltiples procedimientos estadísticos no paramétricos utilizados en la actualidad. El procedimiento elegido para el análisis del conjunto de datos de esta tesis es la prueba de Kolgomorov-Smirnov [4], debido a que permite comparar dos conjuntos de datos que no se relacionan entre sí, es decir, que fueron obtenidos de manera independiente uno del otro. Además, la prueba se caracteriza por ser sensible a diferencias, tanto locales como de forma en la distribución empírica, que calcula en base a los conjuntos de datos recibidos para comparar.

La prueba de Kolgomorov-Smirnov se basa en el cálculo de la frecuencia acumulada para cada uno de los datos de los conjuntos de entrada.

La frecuencia acumulada es útil para determinar cuantos valores de un conjunto de datos están por encima o por debajo de cierto umbral. Se calcula tomando la frecuencia de aparición de un valor y sumándole la de todos sus predecesores. Con el cálculo de la misma, para cada uno de los conjuntos de datos, se obtienen dos muestras de frecuencias acumuladas que luego se pueden comparar punto a punto. Utilizando dicha comparación, se puede estudiar la divergencia entre cada uno de los puntos de ambas frecuencias acumuladas.

Finalmente, un análisis estadístico sobre la divergencia máxima para los pares de puntos de las distribuciones de frecuencias acumuladas determinará si las muestras son estadísticamente similares o diferentes.

Gracias a que se adapta a las características de los conjuntos de datos a estudiar y a su sensibilidad, es que se empleará Kolgomorov-Smirnov como procedimiento estadístico para el análisis de los resultados.

Los conjuntos de datos a analizar en este caso son los escenarios de llenado de bloques de a pares, según tipo de minado.

Lo previamente expuesto implica que se analizarán dos pares para el caso de bloques Bitcoin vacíos. Uno de ellos es el que contiene a los resultados del experimento de minado sin *merged mining* y el otro es el que contiene los resultados del experimento de minado con *merged mining*.

De igual manera, se estudiarán los resultados obtenidos para los casos de llenado de bloques restantes.

Esto dará como resultado cuatro pares de comparaciones. Los valores de los datos a analizar en cada par de comparaciones, dependerán del escenario que se esté estudiando. Sin embargo, el tipo de dato a analizar es análogo entre todos los escenarios, siendo el mismo la diferencia entre el tiempo de minado de cada bloque Bitcoin y el *target time*.

Antes de comenzar el análisis con el método estadístico de Kolgomorov-Smirnov, se muestra que los datos a analizar cumplen con las condiciones requeridas por el método:

- **Los dos conjuntos de datos a comparar son independientes.**

  Los datos son el resultado de ejecuciones, en momentos distintos del tiempo de los experimentos.

  Además, cada experimento comienza con un estado limpio de los componentes de software, es decir, se ignora todo tipo de datos proveniente de una corrida anterior y se comienza con datos nuevos.

  Por ejemplo, la base de datos del nodo Bitcoin, comienza con una cantidad de bloques, que al finalizar el experimento es mayor. Para el próximo experimento se descarta la anterior y se comienza con una base nueva con la cantidad de bloques requerida para la corrida que se desea hacer.

- **Las observaciones que se utilizan para computar la prueba, son obtenidas de manera aleatoria de una población continua y sus valores son independientes mutuamente.**

  Cada uno de los datos de los resultados a analizar representa el tiempo de minado de un bloque Bitcoin o algún dato que se deduce de ese valor, tal como puede ser, la diferencia con respecto al *target time* esperado. Por definición del proceso de minado Bitcoin, el tiempo de minado de cada bloque es independiente del tiempo de minado de su bloque anterior (o incluso posterior), por lo que se cumple que los valores a analizar son mutuamente independientes.

  Es más, la población de bloques minados es una población continua, ya que los posibles tiempos o valores para los que puede ser minado un bloque, son infinitos.

  El conjunto de valores a analizar es obtenido aleatoriamente, ya que se utilizan los bloques minados en los 864 $\mathrm{mín}$ de duración del experimento y sin manipular de ninguna forma la elección de los mismos.

La validación de condiciones presentada, tiene vigencia para cada par de datos a analizar en esta sección.

Existen cuatro pares de datos a examinar según los escenarios planteados. Cada elemento del par se diferencia en el tipo de minado (con o sin *merged mining*), mientras que los pares entre sí, se diferencian por la cantidad de transacciones con que fueron llenados los bloques Bitcoin (vacío, 25 $\%$ completo, 50 $\%$ completo, lleno)

Los elementos a presentar a continuación para la prueba de Kolgomorov-Smirnov son comunes a cada par de casos, por lo que se exhiben de forma general:

- **Hipótesis nula.** No hay diferencia entre el minado con y sin *merged mining*.

- **Hipótesis alternativa.** Existe diferencia entre el minado con y sin *merged mining* pero no se puede indicar en favor de cual.

- **Nivel de significación asociado a la hipótesis nula.** Determinado por el valor crítico, también llamado α. En este análisis se utilizará $\alpha = 0{,}01$, por ser uno de los valores que son empleados habitualmente para este tipo de pruebas. Además, es uno de los valores más estrictos en cuanto a confianza sobre el resultado, indicando que existen un 99 $\%$ de chances de que el mismo sea real y no debido al azar.

- **Resultado Estadístico.** Valor utilizado por la prueba para sintetizar los datos y luego poder determinar la aceptación o el rechazo de la hipótesis nula.

- **p-valor.** Valor que se deriva del resultado estadístico y se utiliza directamente para comparar con α. Si α es menor al p-valor, entonces se acepta la hipótesis nula. Caso contrario se debe rechazar la hipótesis nula.

Para poder ejecutar la prueba de Kolgomorov-Smirnov, se creó un programa que leyera de los archivos con información agrupada por escenario y experimento. Dichos archivos son los mismos que se utilizaron en el análisis inicial.

Culminada la lectura, el programa se encarga de computar la diferencia con respecto al *target time* y utilizar el resultado de ese cómputo, como dato de entrada para la prueba estadística.

La biblioteca de código utilizada para computar la prueba estadística fue `SciPy` versión 1.6.3 para `Python` en su versión 3.9.0. En particular, se utilizó el método `ks_2samp` que toma dos parámetros de entrada, siendo ellos las dos muestras sobre las que se quiere aplicar el test estadístico.

`SciPy`, por defecto, imprime por pantalla el resultado estadístico y el p-valor. A dicha impresión, se le agregó información extra para poder obtener todos los datos relacionados al test al mismo tiempo . Tales valores fueron el α, el tamaño de la muestra y la comparación entra α y p-valor para determinar si aceptar o rechazar la hipótesis nula.

Los valores obtenidos para cada caso de este escenario pueden verse volcados en la tabla 3.1.

Llenado de bloques	**Tamaño resultados con *merged mining***	**Tamaño resultados sin *merged mining***	**Resultado estadístico**	α	**p-valor**
0 %	118	118	0,11	0,01	0,37
25 %	118	118	0,12	0,01	0,29
50 %	118	118	0,12	0,01	0,29
100 %	118	118	0,16	0,01	0,09

Tabla 3.1: Resultados de aplicar la prueba estadística de Kolgomorov-Smirnov, a los pares de resultados con y sin *merged mining* para distintos escenarios de llenado de bloques Bitcoin.

Puede observarse que el p-valor es mayor al valor de α para todos los casos de llenado de bloques. Esto indica que la hipótesis nula debe ser aceptada y por ende que no existe diferencia, entre los escenarios de minado con y sin *merged mining*.

En esta sección, empleando el análisis estadístico, se estudió en detalle la relación entre las distribuciones de minado de bloques Bitcoin con y sin *merged mining*, para los distintos escenarios de llenado con transacciones de bloques Bitcoin.

Luego del análisis, es válido concluir que el *merged mining* no genera impacto en el minado Bitcoin, al variar la cantidad de transacciones que el bloque Bitcoin posee.

Gracias al resultado de este experimento, se puede dejar fija la cantidad de transacciones del bloque Bitcoin para futuros experimentos. Esto permitirá expandir el análisis, modificando otras variables relevantes para el minado y evaluando si se detecta algún el impacto del *merged mining* en el minado Bitcoin.

3.2 Rendimiento de la generación de bloques Bitcoin variando el *target time* de Bitcoin

En esta sección se presentan y analizan resultados de distintos escenarios de minado con y sin *merged mining*, donde varía el *target time*, también llamado tiempo entre bloques, para los bloques Bitcoin. Los tiempos entre bloques que se verán en los resultados, son de 1 $\mathrm{mín}$, 6 $\mathrm{mín}$ y 10 $\mathrm{mín}$.

El tiempo entre bloques de 6 $\mathrm{mín}$ es el mismo que se utilizó en el experimento 3.1, donde variaba la cantidad de transacciones en el bloque Bitcoin. Dicho tiempo fue elegido, dado que llevar a cabo todos los experimentos con el tiempo que utiliza Bitcoin en su red productiva (10 $\mathrm{mín}$), hubiera demorado demasiado por la cantidad de pruebas a realizar. Sin embargo, con el objetivo de hacer extensivos los resultados obtenidos para un *target time* de Bitcoin de 6 $\mathrm{mín}$, se incluyen como parte de estas pruebas, los resultados para un *target time* de 10 $\mathrm{mín}$.

El tiempo de 1 $\mathrm{mín}$ entre bloques responde a lo que consideramos como peor caso, es decir, un escenario con generación de bloques diez veces más rápida que la red productiva de Bitcoin. Con esto se busca exponer al proceso de minado y a los componentes de software involucrados en el mismo, a una mayor exigencia que la que demandan los escenarios reales productivos.

A continuación se detallan los valores para otros parámetros de los experimentos de esta sección:

- ***Target time* para bloques RSK.** El valor utilizado fue de 30 s, es decir, el mismo valor que en la red productiva de RSK.
- **Cantidad de transacciones en bloques.** Los bloques de Bitcoin estuvieron siempre llenos de transacciones al igual que los de RSK, ya que esto representa el peor caso de exigencia de procesamiento para ambos nodos.
- **Cantidad de bloques a minar en Bitcoin.** Se eligieron 144 bloques para cada escenario de los experimentos, ya que este valor representa la cantidad de bloques totales a minar en la red mainnet de Bitcoin.

Gracias a la cantidad de bloques de Bitcoin determinada para cada escenario y a los tiempos entre bloques de 1 $\mathrm{mín}$, 6 $\mathrm{mín}$ y 10 $\mathrm{mín}$, es que las pruebas tuvieron duraciones de 144 $\mathrm{mín}$, 864 $\mathrm{mín}$ y 1440 $\mathrm{mín}$ respectivamente.

Finalmente y con el objeto de obtener una estadística confiable, las pruebas para cada escenario de *target time* de este experimento, se repitieron diez veces para cada bloque por tipo de minado (con o sin *merged mining*).

3.2.1 Análisis inicial

En este fragmento de análisis, se procederá a comparar los resultados que representan el minado de bloques de Bitcoin para los escenarios con y sin *merged mining*. Esto se realizará utilizando un *target time* de Bitcoin de 1 $\mathrm{mín}$, 6 $\mathrm{mín}$ y 10 $\mathrm{mín}$.

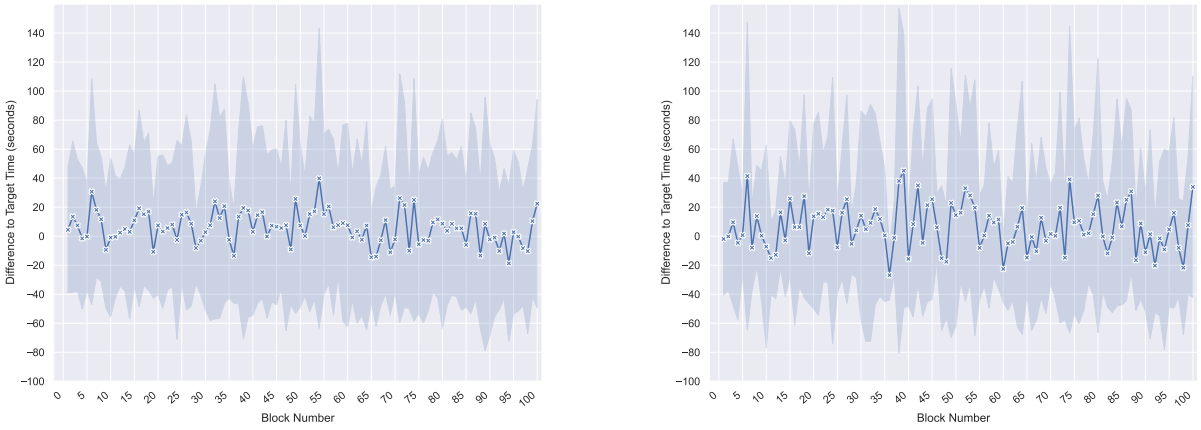
Para poder generar dichas figuras, se utilizaron los mismos programas de procesamiento de *logs* del nodo Bitcoin y de producción de gráficos que en la sección de análisis inicial 3.1.1.

A continuación se presentará un análisis para las figuras de a pares y agrupadas por *target time*.

Para el caso de *target time* de Bitcoin de 1 $\mathrm{mín}$, se puede observar en las figuras 3.3(a) y 3.3(b) que no hay diferencias significativas en el promedio y en el desvío estándar. Esto indica, a simple vista, que no es posible distinguir una muestra de minado de bloques Bitcoin obtenida minando solo Bitcoin, de una que mina Bitcoin con *merged mining*.

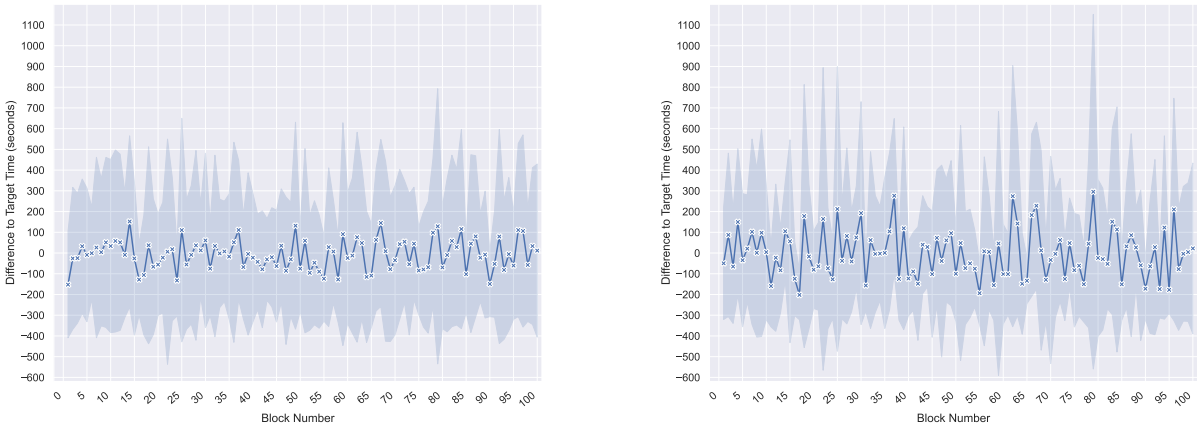
Al observar el caso de *target time* de Bitcoin de 6 $\mathrm{mín}$, a través de las figuras 3.4(a) y 3.4(b), se aprecia que los puntos graficados tienen una distribución similar. Esta característica hace que no se pueda diferenciar un gráfico generado a partir de bloques Bitcoin obtenidos con *merged mining*, de uno construído a partir de bloques generados solo con minado Bitcoin.

Finalmente, para el caso de *target time* Bitcoin de 10 $\mathrm{mín}$, la diferencia entre el tiempo de minado de cada bloque con su *target time*, no presenta outliers que indiquen que existe alguna anomalía, para algunos



(a) Sin *merged mining* (b) Con *merged mining*

Figura 3.3: Escenario de minado de bloques Bitcoin con *target time* de 1 $\mathrm{mín}$.
Análisis de la diferencia entre el tiempo promedio de minado por bloque y su *target time*. Para tener un mejor entendimiento de la dispersión de los datos se graficó el desvío estándar alrededor del promedio. Cuando no hay diferencia en los tiempos, indica que el bloque fue minado en el *target time* esperado (1 $\mathrm{mín}$). Una diferencia negativa, indica que el bloque fue minado con anterioridad al *target time* y una positiva, indica que el bloque fue minado con posterioridad al *target time*.

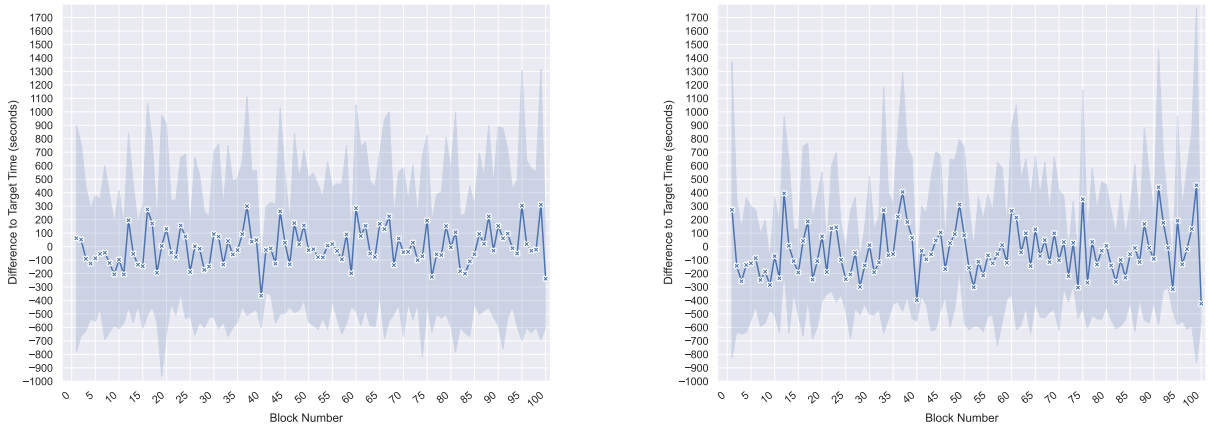


(a) Sin *merged mining* (b) Con *merged mining*

Figura 3.4: Escenario de minado de bloques Bitcoin con *target time* de 6 $\mathrm{mín}$.
Análisis de la diferencia entre el tiempo promedio de minado por bloque y su *target time*. Para tener un mejor entendimiento de la dispersión de los datos, se graficó el desvío estándar alrededor del promedio. Cuando no hay diferencia en los tiempos, indica que el bloque fue minado en el *target time* esperado (6 $\mathrm{mín}$). Una diferencia negativa, indica que el bloque fue minado con anterioridad al *target time* y una positiva, indica que el bloque fue minado con posterioridad al *target time*.

de los escenarios de minado. El análisis previo surge de observar las figuras 3.5(a) y 3.5(b) y permite concluir que son indistinguibles.

Hasta aquí se estudiaron las distribuciones generadas por minado con y sin *merged mining* para los *target time* de Bitcoin de 1 $\mathrm{mín}$, 6 $\mathrm{mín}$ y 10 $\mathrm{mín}$. Para cada análisis particular, se exhibieron distintas razones



(a) Sin *merged mining*

(b) Con *merged mining*

Figura 3.5: Escenario de minado de bloques Bitcoin con *target time* de 10 $\mathrm{mín}$.
Análisis de la diferencia entre el tiempo promedio de minado por bloque y su *target time*. Para tener un mejor entendimiento de la dispersión de los datos se graficó el desvío estándar alrededor del promedio. Cuando no hay diferencia en los tiempos, indica que el bloque fue minado en el *target time* esperado (10 $\mathrm{mín}$). Una diferencia negativa, indica que el bloque fue minado con anterioridad al *target time* y una positiva, indica que el bloque fue minado con posterioridad al *target time*.

por las que los pares de gráficos eran indistinguibles.

Es correcto afirmar que las razones mostradas para un par de gráficos, son también válidas para los demás pares. Basta con tomar las justificaciones expuestas para un par de gráficos y observar cualquier otro par, que se diferencia en el *target time* de Bitcoin, para que quede en evidencia lo afirmado.

La indistinguibilidad de los gráficos permite concluir, que por el momento no se percibe un impacto del minado con *merged mining*, sobre el rendimiento del minado Bitcoin.

Al comparar las figuras en conjunto, es decir, una figura cualquiera de un escenario de minado Bitcoin con *merged mining* y una de un escenario de solo minado Bitcoin, se observa que se mantiene la misma indistinguibilidad que en las comparaciones agrupadas por igual *target time*.

Es por lo anterior, que a pesar de que varíe el *target time*, no es posible diferenciar gráficos generados a partir de bloques minados con Bitcoin de otros generados en base a bloques minados con Bitcoin realizando *merged mining*.

El resultado previo es importante, porque confirma además que las conclusiones exhibidas para los experimentos de llenado de bloques Bitcoin, con distinta cantidad de transacciones y *target time* de 6 $\mathrm{mín}$ (sección 3.1), pueden ser extendidas al caso de *target time* Bitcoin de 10 $\mathrm{mín}$ (escenario productivo).

En este análisis inicial, se compararon las figuras que representan la diferencia de tiempo entre el minado de cada bloque y su *target time*, para distintos valores de *target time* Bitcoin y en escenarios de minado Bitcoin con y sin *merged mining*.

Luego de comparar las figuras en detalle, se concluye que no se detecta impacto del *merged mining* en el rendimiento del minado de Bitcoin al variar el *target time* de Bitcoin. Queda para la siguiente sección complementar este análisis con uno estadístico para ver si las conclusiones alcanzadas, mantienen su validez.

3.2.2 Análisis estadístico

El objetivo de este análisis será decidir si los resultados de los experimentos son distinguibles estadísticamente. Esto determinará el rendimiento del minado Bitcoin al realizarse en conjunto con *merged mining* con la red de RSK. Para lograr dicho cometido, se utilizará el método estadístico de Kolgomorov-Smirnov, presentado en la sección de análisis estadístico 3.1.2.

En esta ocasión, los datos a analizar son los resultados obtenidos de los escanarios con y sin *merged mining* para los valores de *target time* de $1\ \mathrm{mín}$, $6\ \mathrm{mín}$ y $10\ \mathrm{mín}$. En particular, para cada uno de los valores de *target time*, se analizará la diferencia en segundos del tiempo de minado de cada bloque Bitcoin respecto del *target time* esperado. Esta comparación será realizada para los bloques generados por minado con y sin *merged mining*.

La única diferencia respecto del análisis estadístico realizado en la sección 3.1.2 es que en este caso hay distintos valores de *target time*. El hecho de variar el *target time*, no modifica el tipo de datos a analizar (diferencia en segundos del tiempo de minado de cada bloque Bitcoin respecto del *target time*) y solo cambia la frecuencia de los mismos, además de la duración de la muestra.

Los últimos dos factores no impactan en los requisitos que pide el método de Kolgomorov-Smirnov sobre los datos de entrada, por lo que las condiciones expuestas en la sección 3.1.2 se mantienen vigentes para este análisis.

Para poder llevar a cabo la prueba estadística, se emplearon los mismos programas que en la sección de análisis estadístico 3.1.2.

Esto quiere decir que en primera instancia, un programa leyó de *log* del software de *pool* los datos y los volcó en un archivo común por escenario. Luego, ese archivo fue leído por otro programa, que se encargó de procesar los datos y manipularlos para finalmente utilizarlos de entrada de la prueba de Kolgomorov-Smirnov.

Los resultados finales fueron impresos por pantalla, para cada escenario a evaluar de este experimento.

En la tabla 3.2 se exhiben los resultados de la prueba de Kolgomorov-Smirnov sobre los distintos escenarios ya presentados. allí se puede observar que el valor de α es menor que el del p-valor para todos los escenarios, siendo esta razón suficiente para aceptar la hipótesis nula.

Por lo expuesto, es correcto concluir que para todos los escenarios, no se distingue impacto del *merged mining* en el rendimiento del minado Bitcoin. El resultado más interesante es que incluso para un escenario de exigencia alta tal como lo fue el de *target time* de $1\ \mathrm{mín}$ tampoco se observó impacto alguno.

Target time **(minutos)**	**Tamaño resultados con *merged mining***	**Tamaño resultados sin *merged mining***	**Resultado estadístico**	α	**p-valor**
1	100	100	0,12	0,01	0,46
6	100	100	0,14	0,01	0,28
10	100	100	0,21	0,01	0,02

Tabla 3.2: Resultados de aplicar la prueba estadística de Kolgomorov-Smirnov a los pares de resultados con y sin *merged mining*, para distintos escenarios de *target time* de Bitcoin.

Hasta el momento se presentó un estudio de los resultados agrupados de a pares por *target time* y utilizando un método estadístico.

Para finalizar el análisis, se estudiará la relación del tiempo promedio de minado de los bloques entre

los distintos escenarios. Los resultados de dicho estudio, serán utilizados para compararlos con los obtenidos en el análisis inicial y en la prueba de Kolgomorov-Smirnov.

La comparación de los resultados obtenidos con y sin *merged mining*, agrupados por *target time*, muestra que no existe diferencia significativa entre ambos tipos de minado. Esto puede ser visualizado en el análisis que se presenta a continuación, basado en los datos exhibidos en la tabla 3.3.

Para un tiempo entre bloques de $1\,\mathrm{mín}$, se observa una diferencia de $9{,}92\,\%$ para el caso de minado Bitcoin sin *merged mining*; mientras que para el caso de minado con *merged mining*, la misma es de $11{,}03\,\%$. La diferencia entre los casos es de $1{,}11\,\%$.

En cuanto a tiempo entre bloques de $6\,\mathrm{mín}$, la diferencia se ubica en un $1{,}34\,\%$. Esto debido a que el minado Bitcoin sin *merged mining* arrojó una diferencia de $-1{,}58\,\%$, mientras que con *merged mining* fue del $-0{,}24\,\%$.

Con un tiempo entre bloques de $10\,\mathrm{mín}$, se observa que la diferencia para el caso de minado Bitcoin sin *merged mining*, es del $-0{,}03\,\%$; mientras que para el caso de minado Bitcoin con *merged mining* es del $-5{,}63\,\%$. Entre ambos escenarios la diferencia es del $5{,}60\,\%$.

Target time **(minutos)**	**Resultados sin *merged mining***			**Resultados con *merged mining***		
	Promedio (segundos)	**Desvío Estándar**	**Diferencia con *target time***	**Promedio (segundos)**	**Desvío Estándar**	**Diferencia con *target time***
1	5,95	11,04	9,92 %	6,62	15,71	11,03 %
6	−5,70	71,19	−1,58 %	−0,87	114,84	−0,24 %
10	−0,19	136,51	−0,03 %	−33,83	185,70	−5,63 %

Tabla 3.3: Resultados de calcular estadísticas en base a los datos obtenidos de los experimentos con y sin *merged mining*, para distintos escenarios de *target time* de Bitcoin.
El promedio fue calculado utilizando todos los bloques como dato de entrada y sobre la diferencia, entre el tiempo de minado de cada uno y el *target time*.
La diferencia con el *target time*, fue obtenida tomando el valor promedio y calculando cuántos segundos más (o menos) que el tiempo entre bloques, tomó minar los mismos. Está expresada en forma porcentual con respecto a su *target time* correspondiente, para poder comparar estos valores, entre los distintos escenarios de *target time* de Bitcoin.

La diferencia con respecto al *target time* para el minado con y sin *merged mining* es mínima, al ubicarse siempre entre el $1\,\%$ a $2\,\%$, para los casos de *target time* Bitcoin de $1\,\mathrm{mín}$ y $6\,\mathrm{mín}$. En cambio, para el caso de *target time* Bitcoin de $10\,\mathrm{mín}$, se observa una diferencia de alrededor de un $5\,\%$.

Dado que el minado es un proceso aleatorio, se pueden atribuir las diferencias mostradas anteriormente, a la aleatoriedad inherente al mismo.

Los resultados recientemente expuestos, confirman que no hay impacto del *merged mining* en el rendimiento de la generación de bloques Bitcoin y se alinean con las conclusiones ya obtenidas en el análisis inicial y la prueba de Kolgomorov-Smirnov.

Para finalizar, se introduce un análisis adicional de la diferencia de tiempo de minado de los bloques entre escenarios de distinto valor de *target time* de Bitcoin.

En este caso, se puede ver que el minado sin *merged mining*, tiene una diferencia del $9{,}92\,\%$, de $-1{,}58\,\%$ y de $-0{,}03\,\%$ para los casos de $1\,\mathrm{mín}$, $6\,\mathrm{mín}$ y $10\,\mathrm{mín}$ de tiempo entre bloques respectivamente. A su vez, en el caso de de minado con *merged mining*, las diferencias son de $11{,}03\,\%$, $-0{,}24\,\%$ y $-5{,}63\,\%$ para los casos de $1\,\mathrm{mín}$, $6\,\mathrm{mín}$ y $10\,\mathrm{mín}$ de *target time*.

Se puede observar que para el caso de $1\,\mathrm{mín}$, la diferencia se ubica alrededor de un $10\,\%$; mientras que en los casos de $6\,\mathrm{mín}$ y $10\,\mathrm{mín}$, la misma no supera un $2\,\%$. La disparidad entre $10\,\%$ y $2\,\%$, es relevante y

nos indica que el minado tiende a generar bloques más alejados del *target time*, a medida que el mismo es más chico.

Se desconoce la razón de dicha disparidad, pero una posible hipótesis es que un *target time* de 1 $\mathrm{mín}$ resulte demasiado exigente para el proceso de minado (concebido para un tiempo entre bloques de 10 $\mathrm{mín}$). Este tiempo puede producir un impacto en alguno de los componentes de software involucrados, desencadenando que los bloques se generen más lejos del tiempo esperado.

Si bien hay una diferencia al comparar resultados para distintos valores de *target time*, ocurre que esto se manifiesta para los casos de minado Bitcoin con y sin *merged mining*. Es por lo anterior que este último análisis no afecta las conclusiones ya presentadas, es decir, no hay impacto probado del *merged mining* en el rendimiento de la generación de bloques Bitcoin.

En esta sección se varió el *target time* para bloques Bitcoin, con el objetivo de evaluar, si con distintos valores de éste se detectaba un impacto del *merged mining* en el rendimiento del minado Bitcoin. Se arribó a la conclusión de que el rendimiento del minado Bitcoin no se ve impactado por el *merged mining* en ninguno de los escenarios planteados, incluso en el más exigente donde el tiempo entre bloques fue de 1 $\mathrm{mín}$.

En el análisis presentado hasta el momento, se realizaron experimentos modificando la cantidad de transacciones del bloque Bitcoin en la sección 3.1 y con distintos valores de *target time* para la generación de bloques Bitcoin en la sección 3.2.

Dichos experimentos permitieron evaluar el impacto del *merged mining*, en el rendimiento del minado Bitcoin para escenarios de peor caso, incluso llegando a niveles de exigencia a los que la red productiva de Bitcoin no ha sido expuesta. Con esto se cubren todos los casos de máxima exigencia que se pueden generar, al variar parámetros de la red de Bitcoin.

En la próxima sección se llevará la red de RSK a un nivel de exigencia mayor al que alguna vez se la expuso en producción, con el fin de evaluar si esto genera algún impacto en el minado de bloques Bitcoin.

3.3 Rendimiento de la generación de bloques Bitcoin variando el *target time* RSK

Los resultados que se analizan a continuación, corresponden a un escenario de minado con *merged mining*, para un *target time* en la red de Bitcoin de 1 $\mathrm{mín}$ y distintos valores de *target time* para la red de RSK.

En los experimentos de las secciones 3.1 y 3.2 se modificaron distintas condiciones del minado Bitcoin, para llevar el mismo al límite. De esa manera, se evaluó si el *merged mining* generaba un impacto en su performance. Se concluyó que no había impacto, por lo que en esta sección se buscará exponer el proceso de minado y el sistema que lo implementa a un nivel de exigencia más alto.

Para llevar el minado a dicho nivel, y habiendo agotado las modificaciones a realizar en la red de Bitcoin, lo que se modificó en este caso fue el *target time* para la red de RSK. En los experimentos anteriores se utilizó el valor de la red productiva de RSK (*target time* de 30 s), por lo que para este experimento se redujo el tiempo entre bloques llevando el mismo a un valor de 5 s.

A su vez, se varió el tiempo de *polling* utilizado por el software de *pool*, para obtener el trabajo del nodo de RSK. Este cambio llevó el tiempo a 500 ms, un valor más chico en comparación con los 2000 ms usados en los experimentos anteriores y en redes productivas.

La razón para reducir el tiempo de *polling*, es por un lado, la de aumentar el nivel de exigencia del software de *pool* y del nodo RSK. Por el otro, al aumentar el tiempo entre bloques se debe mantener el correcto funcionamiento del minado RSK y para eso es fundamental que el software de *pool*, cuente siempre con el

último trabajo para construir sus bloques.

De esa manera, al minar bloques con una frecuencia mayor, hay más chances de que el trabajo con el que cuenta el software de *pool* quede desactualizado. Con el objeto de reducir dichas chances, se aumentó la frecuencia de refresco del trabajo.

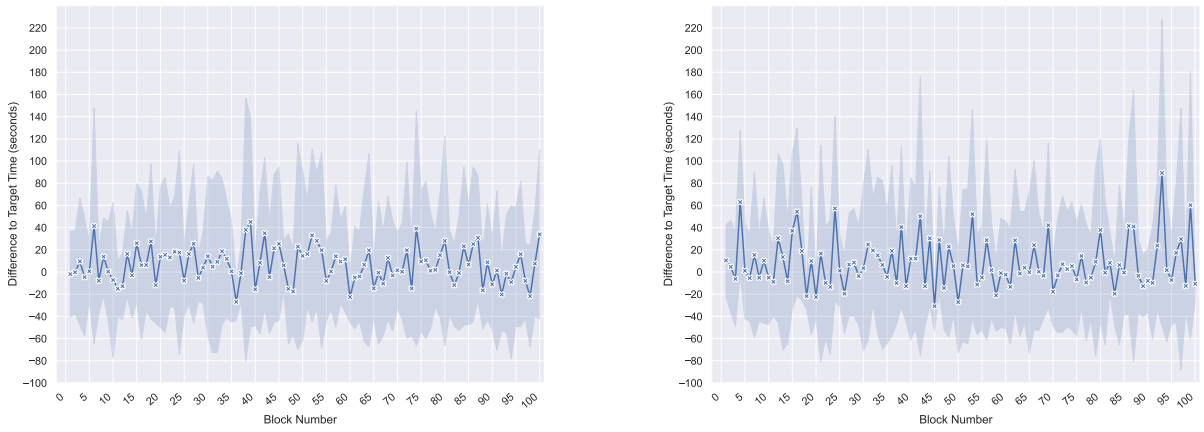
Por último, se modificó la frecuencia de generación de soluciones candidatas por parte del software de *pool* a 1,6 s para poder mantener la relación con respecto al *target time* de RSK.

Las primeras dos modificaciones previamente explicadas, permiten aumentar la exigencia de la parte correspondiente a RSK del proceso de minado con *merged mining*. Este nuevo nivel de exigencia, es varias veces mayor a lo utilizado hasta el momento, en las pruebas y a lo visto alguna vez en redes productivas.

Si combinamos lo anterior con un *target time* de Bitcoin de 1 $\mathrm{mín}$, se obtiene el peor caso que se presentará en esta sección y que será comparado con el peor caso de la sección anterior (*target time* de 1 $\mathrm{mín}$ para Bitcoin y de 30 s para RSK), ambos minando con *merged mining*.

Para comparar ambos casos se llevaron a cabo dos experimentos, cada uno de ellos con una duración de 144 $\mathrm{mín}$. Dicha duración fue determinada por la cantidad de bloques Bitcoin que se minan en un día en un ambiente productivo (144), suponiendo 1 $\mathrm{mín}$ de *target time*. Los dos experimentos fueron repetidos diez veces para cada bloque Bitcoin y para cada escenario de *target time* de RSK; contando así con la robustez estadística necesaria para realizar un correcto análisis.

Para el procesamiento de los resultados que se observan en las figuras 3.6(a) y 3.6(b), se utilizaron los mismos programas de las secciones anteriores con los ajustes pertinentes para los escenarios actuales de *target time*.



(a) RSK *target time* de 30 s

(b) RSK *target time* de 5 s

Figura 3.6: Escenario de minado de bloques Bitcoin con *target time* Bitcoin de 1 $\mathrm{mín}$ y variando el *target time* para RSK.
Análisis de la diferencia entre el tiempo promedio de minado por bloque y su *target time*. Para tener un mejor entendimiento de la dispersión de los datos, se graficó el desvío estándar alrededor del promedio. Cuando no hay diferencia en los tiempos, indica que el bloque fue minado en el *target time* esperado (1 $\mathrm{mín}$). Una diferencia negativa, indica que el bloque fue minado con anterioridad al *target time* y una positiva, indica que el bloque fue minado con posterioridad al *target time*

Tal como se puede observar en las figuras 3.6, la generación de bloques Bitcoin para el escenario de *target time* de RSK de 30 s, no es distinguible de aquellos bloques generados en el escenario de *target time* de RSK de 5 s. La observación anterior, tiene sustento en que ambas figuras presentan una distribución simi-

lar para la diferencia de los bloques minados con su *target time*, un desvío estándar semejante y ninguna anomalía que permita diferenciarlas.

En el caso de la prueba estadística de Kolgomorov-Smirnov, los resultados a analizar son la diferencia entre el tiempo de minado de cada bloque Bitcoin y su *target time* esperado. Estos resultados, tienen el mismo formato que los utilizados para correr la prueba estadística en secciones anteriores por lo que se mantienen vigentes las mismas asunciones.

Vale la pena recordar que la hipótesis nula para Kolgomorov-Smirnov, es que no hay diferencia entre el minado con y sin *merged mining*. La tabla 3.4 muestra que el valor de α es menor al del p-valor, por lo que es correcto aceptar la hipótesis nula.

Bitcoin *target time* (minutos)	**Tamaño resultados RSK *target time* 5 s**	**Tamaño resultados RSK *target time* 30 s**	**Resultado estadístico**	α	**p-valor**
1	100	100	0,11	0,01	0,58

Tabla 3.4: Resultado de aplicar la prueba estadística de Kolgomorov-Smirnov a los pares de resultados con *target time* de 5 s y 30 s, para el escenario de 1 $\mathrm{mín}$ de *target time* de Bitcoin.

Tanto la comparación de los gráficos de generación de bloques como el análisis estadístico, muestran que no es posible distinguir un resultado de generación de bloques Bitcoin para el escenario con *target time* RSK de 30 s, de aquél obtenido para el escenario con *target time* RSK de 5 s.

En esta sección, se presentaron y analizaron los resultados para el caso de exigencia máxima del proceso de minado Bitcoin, con *merged mining* en la red de RSK. En dicho caso, Bitcoin se configuró para generar bloques cada 1 $\mathrm{mín}$ y RSK cada 5 s; se utilizó una configuración de *polling* del trabajo de RSK de 500 ms y una generación de soluciones candidatas por parte del software de *pool* cada 1,6 s.

Es correcto concluir entonces, que no es posible identificar impacto del *merged mining* en el desempeño del minado Bitcoin para los dos escenarios de *merged mining* estudiados.

Los escenarios recientemente comparados fueron de minado Bitcoin con *merged mining*, utilizando ambos *target time* Bitcoin de 1 $\mathrm{mín}$ y cada uno de ellos, *target time* RSK de 30 s y de 5 s.

El caso de *target time* de Bitcoin de 1 $\mathrm{mín}$ y *target time* RSK de 30 s, ya había sido presentado en la sección 3.2. Ahí se lo estudió en conjunto con otros dos escenarios, minado Bitcoin con y sin *merged mining*, para los que el *target time* de Bitcoin fue de 6 $\mathrm{mín}$ y 10 $\mathrm{mín}$ con un *target time* RSK de 30 s.

Con los resultados de la presente sección, se encontró que el escenario de mayor exigencia donde el *target time* RSK es de 5 s no pudo ser distinguido del caso de *target time* de RSK de 30 s, ambos con *target time* Bitcoin de 1 $\mathrm{mín}$. Es por los anteriormente mencionado, que no se pueden distinguir los resultados de minado de bloques Bitcoin para los escenarios de distinto *target time* RSK.

De tal modo, se puede afirmar además que tampoco se puede distinguir el caso de máxima exigencia de cualquiera de aquellos, con un *target time* Bitcoin de 6 $\mathrm{mín}$ o de 10 $\mathrm{mín}$.

Lo previo, permite hacer extensiva la comparación actual con los resultados de la sección 3.2. Dicho esto, es válido afirmar que no se pueden distinguir los resultados obtenidos de un escenario sin *merged mining*. donde los bloques Bitcoin son generados con *target time* de 1 $\mathrm{mín}$, de otro escenario de igual *target time* Bitcoin, con *merged mining* y *target time* RSK de 5 s.

En los experimentos mostrados hasta el momento se modificaron los valores más significativos de funcionamiento de minado para las redes de Bitcoin y de RSK, llevando esto a generar diversos escenarios de exigencia del proceso minado. Para los distintos escenarios con y sin *merged mining*, productivos y no productivos pero más exigentes, se encontró que los resultados de minado de bloques Bitcoin no son distinguibles.

Por las razón anteriormente expuesta, es correcto concluir que el *merged mining* no afecta el rendimiento del minado Bitcoin.

3.4 Rendimiento de la generación de bloques RSK variando *target time* Bitcoin y *target time* RSK

El análisis realizado hasta el momento, estuvo enfocado en la generación de bloques Bitcoin y el impacto que el *merged mining* pudiera tener en el rendimiento de dicho minado.

En cada una de las secciones previas, se fue monitoreando el correcto funcionamiento del minado de RSK sin encontrar anomalías. Sin embargo, en algunos escenarios se identificó un cambio en la proporción de *best blocks* y *uncles* minados en RSK. Si bien ya se demostró que realizar *merged mining* no afecta de ninguna manera al rendimiento del minado Bitcoin, foco principal de análisis de este trabajo, es interesante ver de qué forma se ve afectado el minado RSK, en los distintos escenarios donde la proporción de *best blocks* y *uncles* no fue la esperada.

En esta sección se presenta un análisis detallado de la generación de bloques RSK para diferentes escenarios.

Inicialmente se estudiarán los casos para *target time* de RSK de 30 s (valor utilizado en entornos de producción) y distintos valores de *target time* Bitcoin. Luego, se tomará un *target time* fijo de Bitcoin de 1 $\mathrm{mín}$ y se presentarán resultados para distintos escenarios de *target time* de RSK.

Antes de analizar los pormenores de los resultados de los experimentos, se detallarán las condiciones de los mismos y esto se hará de manera unificada, ya que los casos a estudiar compartieron dichas condiciones.

El *target time* de Bitcoin y de RSK se fue configurando de igual manera que en las secciones anteriores, para generar los escenarios deseados. Otros parámetros tales como el *target time* del *pool* de minería y el *polling time* al nodo de RSK, fueron configurados en sus respectivos archivos de configuración dentro del software de *pool*.

Los bloques tanto para Bitcoin como para RSK estuvieron siempre llenos de transacciones y la duración de cada experimento fue el equivalente en minutos, al minado de 144 bloques Bitcoin (cantidad promedio esperada de bloques minados en un día en el ambiente productivo Bitcoin).

Cada experimento fue repetido diez veces para obtener resultados más confiables desde el punto de vista estadístico. En consecuencia, para cada bloque de RSK se tienen diez repeticiones, siendo esto valedero para cada uno de los escenarios estudiados.

Para la obtención de los resultados se utilizó el archivo de *log* del nodo de RSK donde, como parte de la información registrada, se encuentra aquella sobre los bloques minados. Esta incluye varios datos relacionados al proceso de minado, entre los cuales figuran el tiempo de minado del bloque y el resultado de procesar dicho bloque.

El resultado del procesamiento de un bloque en RSK puede arrojar distintos valores, los únicos observados en los experimentos de esta sección, son los de *best block* y *uncle* por lo que no se hará hincapié en los demás. El análisis de dichos resultados, se realizó con una serie de programas de procesamiento.

El primero de ellos se encargó de interpretar el *log* del nodo RSK, extraer la información sobre los bloques y volcar la misma en un archivo común que tuviera las diez ejecuciones de cada experimento. Luego, otro programa leyó la información almacenada en el archivo común del experimento, para calcular las estadísticas necesarias.

Es importante destacar que para dichas estadísticas, se contó la cantidad de *best blocks* y de *uncles* presentes en cada una de las diez ejecuciones, para luego realizar un promedio de ellas. Dicho promedio es el

valor que se utilizará para para luego obtener la proporción de *best blocks* y *uncles* necesaria para el análisis. Si bien el total de bloques minados varió entre cada ejecución, la variación fue lo suficientemente pequeña para que no hiciera falta ignorar bloques en pos de tener resultados de igual tamaño.

Para comenzar con el análisis se estudiarán los resultados para el experimento donde el *target time* de Bitcoin toma los valores de 1 $\mathrm{mín}$, 6 $\mathrm{mín}$ y 10 $\mathrm{mín}$. Dichos resultados se exhiben en la tabla 3.5 y muestran que un funcionamiento normal del minado de RSK, arroja una proporción del 58 $\%$ de *best blocks* y del 42 $\%$ de *uncles*.

A la conclusión anterior se llega luego de observar que esos porcentajes, se presentan para el caso de *target time* Bitcoin de 10 $\mathrm{mín}$ (escenario productivo) y para el de *target time* Bitcoin de 6 $\mathrm{mín}$. Es llamativo cómo esa proporción se modifica en favor de más *best blocks* al reducir el *target time* Bitcoin.

Como consecuencia de este último comportamiento, se investigó el flujo de minado en detalle, partiendo desde el momento en que el software de *pool* obtiene los trabajos del nodo de Bitcoin y de RSK. Se continuó con la interpretación de los mismos y con el proceso de generación del nuevo trabajo, hasta finalizar con la construcción y envío del mensaje con la actualización al hardware de minería.

El software de *pool* tiene en su documentación, registro de que es posible configurarlo para enviar una actualización del trabajo al hardware de minería cada vez que se recibe un nuevo trabajo de *merged mining*, en este caso provisto por el nodo de RSK. Tanto en los experimentos de esta sección como en los de las anteriores, se utilizó una configuración de software de *pool* que enviara actualización al hardware de minería, cada vez que se recibe un nuevo trabajo de RSK. Sin embargo, al investigar el flujo de minado, se descubrió que dicha configuración genera el resultado esperado en todos los componentes del software de *pool* salvo por el `StratumServer`. La función de dicho componente es la de interactuar directo con el hardware de minería y tomar la decisión de enviar actualizaciones con nuevos trabajos.

Se encontró en el código fuente que, al momento de tomar la decisión de enviar un nuevo trabajo por *merged mining*, además considera si la última altura de minado de bloque Bitcoin cambió. Esto es contradictorio con los demás componentes, donde según la configuración de *notify policy*, se puede ignorar dicha validación para lograr el envío de actualizaciones por *merged mining*, independientemente de lo que ocurra con la altura de los bloques Bitcoin.

Por lo tanto, al realizar esta validación adicional queda condicionada toda actualización por *merged mining*. La condición es acerca debe haberse producido un cambio de altura, es decir, que debe haber un nuevo un nuevo bloque Bitcoin para poder hacer también verdadera cualquier condición sobre *merged mining*.

El análisis previo implica que el software de *pool* de minería, no envía actualización al hardware de minería siempre que hay un nuevo trabajo de *merged mining*; por lo que existen períodos de tiempo para los que el hardware computa soluciones para trabajos desactualizados. Por ende, dichos trabajos desactualizados son aceptados por el nodo de RSK en forma de *uncles* y no de *best blocks*, siendo ésta la razón por la que se observa una proporción alta de *uncles* entre los bloques minados.

Gracias a lo estudiado hasta el momento, se puede entender la existencia de *uncles* en los escenarios de prueba estudiados.

Dado que la cantidad de actualizaciones de trabajo RSK es dependiente de Bitcoin, tiene sentido que a menor *target time* de Bitcoin, se generen menos *uncles* de RSK. En ese caso, RSK recibiría actualizaciones de trabajo con mayor frecuencia. Los porcentajes de *best blocks* y *uncles* expuestos en la tabla 3.5 confirman dicha dependencia ya que para el *target time* de Bitcoin de 1 $\mathrm{mín}$, hay 32,85 $\%$ de *uncles*, mientras que para el *target time* de Bitcoin de 6 $\mathrm{mín}$ y 10 $\mathrm{mín}$, se presentan alrededor de un 41 $\%$ de *uncles*.

En este trabajo cubrimos distintos escenarios, siempre utilizando versiones productivas de los distintos software empleados en el proceso de minado Bitcoin con *merged mining* para la red de RSK.

Queda pendiente para futuros trabajos, la realización de una prueba con una versión del software de *pool*, donde la condición de validación de altura de bloque Bitcoin no esté presente al momento de evaluar

Bitcoin *target time* (minutos)	RSK *target time* (segundos)	*Best blocks*	*Uncles*
1	30	67,15 %	32,85 %
6	30	58,35 %	41,65 %
10	30	57,76 %	42,24 %

Tabla 3.5: Resultado de contar la cantidad de bloques minados de RSK para los escenarios de *target time* Bitcoin de 1 mín, 6 mín y 10 mín, con un *target time* RSK de 30 s.
Los porcentajes de *best blocks* y de *uncles* fueron calculados sobre el promedio de la cantidad de apariciones de cada bloque minado, en las diez ejecuciones de cada escenario. El desvío estándar para dicho promedio se ubicó por debajo del 1 %.

el envío de actualizaciones por parte del software de *pool* al hardware de minería. El objetivo de la prueba mencionada anteriormente sería el de ver si se reduce la cantidad de *uncles*.

Hasta el momento se estudió el impacto de distintos valores de *target time* Bitcoin para un mismo *target time* de RSK (30 s).

A continuación, se examinará el caso donde el *target time* de Bitcoin se encuentre fijo en 1 mín y se varíe el *target time* de RSK, con el objetivo de evaluar la evolución de la proporción de *best blocks* y *uncles*.

La elección de 1 mín de *target time* de Bitcoin tiene que ver con que éste es el caso de mayor exigencia de minado Bitcoin que estudiamos en este trabajo. Los valores de *target time* de RSK a utilizar fueron los de 5 s, 30 s y 60 s. En este caso se utiliza el valor de 30 s como referencia al ser el de entornos productivos y el utilizado a lo largo de todo este trabajo.

Se busca estudiar un escenario donde la exigencia hacia el minado RSK sea menor a la productiva y por eso se presentarán resultados donde el *target time* sea de de 60 s. A la vez, se examinarán escenarios de exigencia máxima del minado RSK donde el *target time* sea de 5 s. La variación del *target time* de RSK, implica que otras variables del proceso de minado que se relacionan con el mismo deben ser modificadas. Las mismas son:

- ***Target time* del software de *pool*.** Tiene una relación tres veces menor al *target time* de RSK según configuraciones utilizadas en ambientes productivos. Dicha relación se respetó para todos los escenarios de este experimento.

- **get work *polling time*.**. El valor sugerido por la red de RSK para entornos productivos, es de 2000 ms. Tal valor se utilizó en todos los escenarios salvo para uno de los dos cuya configuración de *target time* de RSK fue de 5 s. En ese caso se redujo a 500 ms con el objeto de tener lo más pronto posible el último trabajo de la red de RSK.

  Un posible caso donde se evidencia el beneficio para el minado de *best blocks* es el siguiente: el trabajo de RSK se consultó inmediatamente antes de que se minara un *best block* y, gracias a tener un *polling time* de 500 ms, el software de *pool* se enteró 1500 ms antes del nuevo trabajo, si se compara esto con un caso de *polling time* de 2000 ms. El caso anterior es extremo, pero deja en evidencia que un menor valor de *polling time* permite tener mayor disponibilidad del último trabajo de RSK.

En la tabla 3.6 se pueden observar las distintas configuraciones previamente mencionadas para cada uno de los escenarios de este experimento, como así también los resultados de *best blocks* y *uncles* obtenidos.

Al observar los resultados, se puede ver que el porcentaje de *best blocks* aumenta a medida que el *target time* de RSK lo hace. Esto tiene su explicación en que las actualizaciones de trabajo de RSK desde el software

Bitcoin *target time* (minutos)	**RSK *target time* (segundos)**	**Pool *target time* (segundos)**	**get work *polling time* (milisegundos)**	***Best blocks***	***Uncles***
1	5	1,66	500	46,44 %	53,56 %
1	5	1,66	2000	44,40 %	55,60 %
1	30	10	2000	67,15 %	32,85 %
1	60	20	2000	89,80 %	10,20 %

Tabla 3.6: Resultado de contar la cantidad de bloques minados de RSK para los escenarios de *target time* Bitcoin 1 míns y de *target time* RSK de 5 s, 30 s y 60 s.
Los porcentajes de *best blocks* y de *uncles* fueron calculados sobre el promedio de la cantidad de apariciones de cada bloque minado en las 10 ejecuciones de cada escenario. El desvío estándar para dicho promedio se ubicó por debajo del 1 %.

de *pool* al hardware de minería están, como se mostró anteriormente, condicionadas por la presencia de nuevos bloques o trabajos de Bitcoin.

Por lo tanto, al acercarse el valor del *target time* de RSK al valor de *target time* de Bitcoin, es cuando se obtiene la mejor tasa de actualización de trabajo de RSK. Es decir, el software de *pool* notifica con mayor frecuencia del último trabajo de RSK al hardware de minería, por lo que éste, puede computar soluciones para trabajos actualizados por un mayor tiempo. Esas soluciones terminan siendo *best blocks*.

Lo expuesto previamente se observa en la tabla 3.6 donde la mayor cantidad de *best blocks*, arroja un valor del 89,80 % para el caso en el cual el *target time* de RSK es igual al de Bitcoin. Dicho valor es superior al 67,15 % obtenido para el caso de *target time* de RSK de 30 s y notoriamente superior al obtenido para el caso de target time de RSK de 5 s (55,60 %).

Extendiendo el análisis para los casos de igual *target time* de RSK pero diferente *polling time*, podemos ver que a menor *polling time*, se obtiene un mayor porcentaje de *best blocks*. Se ubica de esta manera la diferencia en 2,04 %.

Este resultado, consecuencia de la reducción del *target time*, se condice con lo esperado y demuestra que este cambio genera un impacto positivo, ya que aumenta la cantidad de *best blocks*. Sin embargo, dicho impacto no es tan notorio como sí ocurre para el caso de aumento del *target time* de RSK. Es posible que ese impacto tome relevancia en un escenario, en el que no exista la condición de validación de cambio de altura de Bitcoin, al momento de decidir el envío de una actualización generada por *merged mining* y hacia el hardware de minería.

Queda pendiente para futuros estudios, la comparación de la cantidad de *best blocks* minados para distintos valores de *polling time*, cuando las actualizaciones de trabajo RSK no están condicionadas por el cambio de altura de bloques Bitcoin.

En esta sección se presentaron y analizaron los resultados para estudiar la proporción de *best blocks*, como así también de *uncles* en los bloques minados de la red de RSK.

En un primer experimento, se varió el *target time* de Bitcoin dejando fijo el *target time* de RSK en su valor productivo de 30 segundos. Se descubrió que para los casos de *target time* Bitcoin de 6 mín y 10 mín, hay una cantidad de *best blocks* de alrededor de un 58 %. Cómo dicho valor fue obtenido con configuraciones de producción, es correcto afirmar que el mismo es el valor esperado para un escenario productivo.

Además, se encontró que para el caso de 1 mín de *target time* Bitcoin, la cantidad de *besst blocks* aumentaba siendo esto contrario a lo esperado. La investigación del flujo de minado en detalle, llevó a encontrar que esto se debe, a que existe un componente de software que tiene una condición que impide que se actualice el trabajo de RSK, al hardware de minería en todas las oportunidades que debería ocurrir.

Un segundo experimento se concentró en variar el *target time* de RSK, dejando el *target time* de Bitcoin

fijo en un valor de peor caso de 1 $\mathrm{mín}$. La conclusión de dicho experimento fue la esperada ya que a mayor *target time* de RSK o mayor refresco del trabajo de RSK a través del *get work polling*, se obtuvieron mayor cantidad de *best blocks*.

Los resultados expuestos en este trabajo, demuestran que para escenarios donde se experimentó con la modificación de valores del minado Bitcoin, cantidad de transacciones de sus bloques y *target time*, el *merged mining* no genera impacto alguno en el minado Bitcoin.

Además, se experimentó con distintos valores de *target time* de RSK y de configuración del software de *pool* pudiendo caracterizar cuál es la proporción productiva de *best blocks* y de *uncles*.

Finalmente se pudo conocer que tanto el *target time* como el *get work polling time*, son los parámetros que determinan la proporción de *best blocks* y de *uncles*.

Capítulo 4

Conclusiones y Trabajo Futuro

La red Bitcoin utiliza un gran poder de cómputo para poder funcionar, en particular para el minado de sus bloques. Dicho poder de cómputo solo cumplía la función de brindar seguridad a la red. Esto cambió con la aparición de *merged mining*, expandiendo la utilidad del poder de cómputo.

Merged mining beneficia a todos los actores del ecosistema de minado Bitcoin. Juega un rol fundamental en el crecimiento de cadenas secundarias de Bitcoin, permitiendo así su escalabilidad. Estas cadenas facilitan la expansión de los distintos casos de uso de la tecnología blockchain.

Muchos de estos casos mejoran la forma en que se utiliza la información, permitiendo implementar soluciones a problemas que antes no tenían o incluso permitiendo solucionar problemas existentes con enfoques más seguros.

La pérdida de rendimiento del minado Bitcoin es la principal amenaza técnica para la adopción de *merged mining*.

En este trabajo, se introdujo una metodología de estudio del rendimiento del minado Bitcoin en distintos escenarios de *merged mining*, realistas y de peor caso.

RSK es la cadena que se eligió como secundaria para los escenarios de esta tesis.

Para estudiar el rendimiento, se diseñó y construyó un entorno de pruebas. Utilizando dicho entorno, se llevaron a cabo experimentos donde se comparó con distintos métodos los resultados de bloques minados con y sin *merged mining*.

En un primer escenario de experimentación, se modificó la cantidad de transacciones incluidas en los bloques Bitcoin; los casos cubiertos fueron bloques vacíos, bloques con un 25 % de transacciones, bloques con un 50 % de transacciones y bloques completos.

Luego, se consideraron escenarios con distintos valores de *target time* de Bitcoin, abarcando 10 mín, 6 mín y 1 mín.

Hasta el momento, todos los experimentos fueron realizados con un *target time* de RSK de 30 s.

Por último, escenarios con diferentes *target time* de RSK, fueron estudiados.

Como parte de los experimentos, se descubrió además que el minado de RSK presenta una proporción de *best blocks* y de *uncles* que no es la esperada. Este hecho no tiene impacto alguno en el rendimiento del minado Bitcoin pero si puede tener un impacto en el minado de RSK.

Los resultados obtenidos demuestran que no se pudo detectar una pérdida de rendimiento en el minado Bitcoin, al hacer *merged mining* con RSK como cadena secundaria.

A continuación se describirá cómo utilizar lo presentado en esta tesis para otros casos, relacionados al rendimiento de Bitcoin y al *merged mining*.

En este trabajo se decidió utilizar a RSK como cadena secundaria. Sin embargo, podría haber sido cualquier otra cadena la elegida.

Es por lo anterior que tanto la metodología, como los escenarios y las herramientas pueden ser utilizados para evaluar el rendimiento de Bitcoin haciendo *merged mining* con una blockchain distinta de RSK.

El trabajo requerido para esto, sería únicamente el de adaptar el software de *pool* de minería, para que haga *merged mining* con la blockchain deseada.

Este trabajo presentó herramientas para ejecutar experimentos de escenarios donde se varía el *target time* de RSK.

Dichos experimentos no dieron los resutlados esperados. Es por esto que se puede valer de las herramientas para repetir los mismos.

Múltiples razónes pueden llevar a dicha repetición. Por citar una de ellas, se podría continuar realizando cambios en el software de *pool* de minería, para evaluar si la proporción de *best blocks* y *uncles* se ve alterada.

Conociendo ahora que el *merged mining* no produce ningún impacto perceptible en el minado Bitcoin, se elimina una amenaza para la adopción de este tipo de minado.

Dejando de lado estudios de rendimiento como el de este trabajo, se puede seguir la línea de investigación de *merged mining*, concentrándose en propuestas que extiendan sus casos de uso. Una de ellas es la que permita utilizarlo, para asegurar blockchains que usen mecanismos de validación de bloques que no sean prueba de trabajo.

Por ejemplo, se podría resumir en un *merkle root* la información de un rango de bloques provenientes de la cadena que no utiliza prueba de trabajo. Luego, ese *merkle root*, incluirlo vía *merged mining* en Bitcoin. En consecuencia, el rango de bloques tendría no solo la validación de su cadena sino también la de Bitcoin. Este es un caso claro de aumento de la seguridad de la cadena que no emplea prueba de trabajo.

Bibliografía

[1] Muneeb Ali, Jude Nelson, Ryan Shea, and Michael J. Freedman. Blockstack: A global naming and storage system secured by blockchains. In *Proceedings of the 2016 USENIX Conference on Usenix Annual Technical Conference (ATC '16)*, pages 181–194, Berkeley, CA, USA, June 2016. USENIX Association.

[2] Andreas M. Antonopoulos. *Mastering Bitcoin: Programming the Open Blockchain*. O'Reilly Media, 2nd edition, June 2017.

[3] Michael Bedford Taylor. Bitcoin and the age of bespoke silicon. In *2013 International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, pages 1–10, 2013.

[4] Gregory W. Corder and Dale I. Foreman. *Nonparametric Statistics: A Step-by-Step Approach*. Wiley, 2 edition, May 2014.

[5] Aljosha Judmayer, Alexei Zamyatin, Nicholas Stifter, Artemios G. Voyiatzis, and Edgar Weippl. Merged mining: Curse or cure? In Joaquin Garcia-Alfaro, Guillermo Navarro-Arribas, Hannes Hartenstein, and Jordi Herrera-Joancomartí, editors, *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, volume 10436 of *Lecture Notes in Computer Science*, pages 316–333, Cham, 2017. Springer International Publishing.

[6] Harry Kalodner, Miles Carlsten, Paul Ellenbogen, Joseph Bonneau, and Arvind Narayanan. An empirical study of namecoin and lessons for decentralized namespace design. In *Proceedings of the 14th Workshop on the Economics of Information Security, WEIS '15*, pages 1–23, June 2015.

[7] Sergio Lerner. RSK bitcoin powered smart contracts. Technical report, RSK, 2019.

[8] I. Magaki, M. Khazraee, L. V. Gutierrez, and M. B. Taylor. Asic clouds: Specializing the datacenter. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 178–190, June 2016.

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

[10] Nicholas Stifter, Philipp Schindler, Aljosha Judmayer, Alexei Zamyatin, Andreas Kern, and Edgar Weippl. Echoes of the past: Recovering blockchain metrics from merged mining. In Ian Goldberg and Tyler Moore, editors, *Financial Cryptography and Data Security (FC 2019)*, volume 11598 of *Lecture Notes in Computer Science*, pages 527–549, Cham, September 2019. Springer International Publishing.

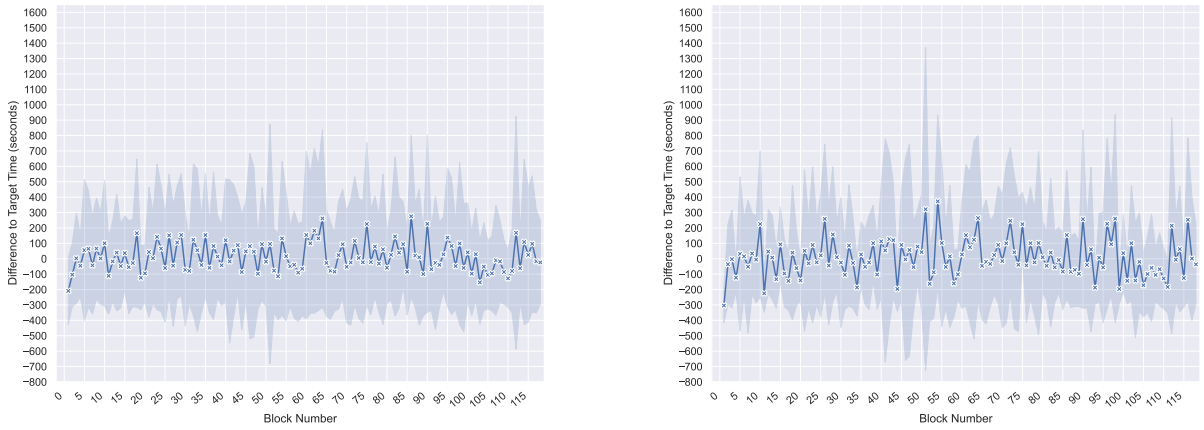
[11] Marco Vanotti. Un avance hacia entornos de gran escala para experimentos con criptomonedas. Master's thesis, Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, 2016.

[12] Xue Zhipeng, Xuliang Duan, and Zheng Wen. Research of merged mining technology in bitcoin blockchain scaling. In Hong Lin, editor, *Proceedings of 2018 the 8th International Workshop on Computer Science and Engineering (WCSE 2018)*, pages 411–416, June 2018.

Apéndice A

Resultados adicionales

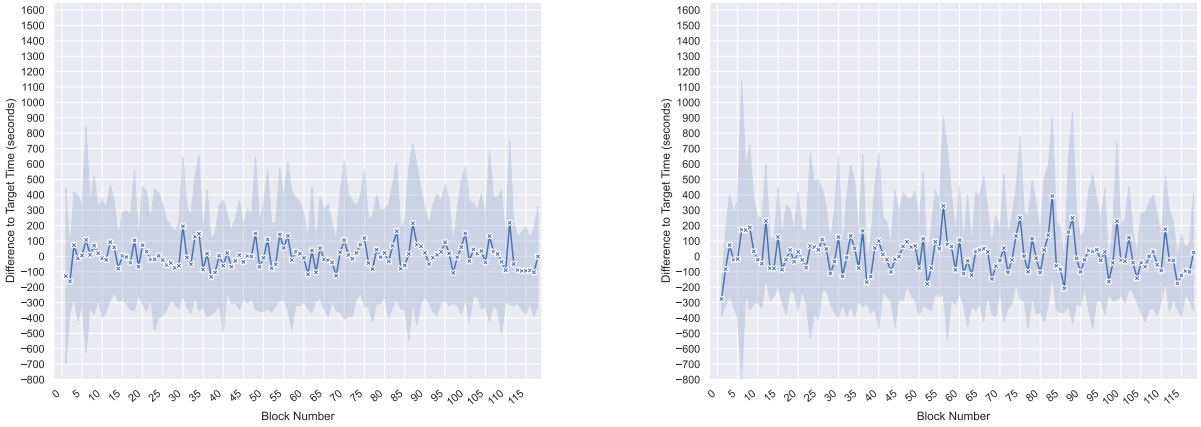
A.1 Rendimiento de la generación de bloques Bitcoin variando la cantidad de transacciones Bitcoin



(a) Sin *merged mining*

(b) Con *merged mining*

Figura A.1: Escenario de minado de bloques Bitcoin llenos con un 25 % de transacciones.
Análisis de la diferencia entre el tiempo promedio de minado por bloque y su *target time*. Se incluye el desvío estándar alrededor del promedio. Los 0 s de diferencia indican que el bloque fue minado en el *target time* esperado (6 min). Una diferencia negativa, indica que el bloque fue minado con anterioridad al *target time* y una positiva, indica que el bloque fue minado con posterioridad.



(a) Sin *merged mining*

(b) Con *merged mining*

Figura A.2: Escenario de minado de bloques Bitcoin llenos con un 50 % de transacciones. Análisis de la diferencia entre el tiempo promedio de minado por bloque y su *target time*. Se incluye el desvío estándar alrededor del promedio. Los 0 s de diferencia indican que el bloque fue minado en el *target time* esperado (6 min). Una diferencia negativa, indica que el bloque fue minado con anterioridad al *target time* y una positiva, indica que el bloque fue minado con posterioridad.