DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 2

Bitcoins - Sistemas distribuidos

22 de noviembre de 2019

Sistemas Operativos
2do Cuatrimestre de 2019

Integrante	LU	Correo electrónico
Giudice, Carlos Rafael	694/15	carlosr.giudice@gmail.com
Olmedo, Dante	195/13	dante.10bit@gmail.com
Rosende, Federico Daniel	222/16	federico-rosende@hotmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Palabras Clave

Bitcoins, Sistemas distribuidos,MPI

Índice

1. Introducción

1.1. Cadena de bloques(blockchain)

El problema a resolver es el desarrollo de una blockchain, la cual es una lista enlazada de bloques distribuida.

La utilidad del blockchain es su inmutabilidad, un bloque tiene metadatos sobre el bloque anterior a el en la cadena, por lo cual si uno quisiera modificar el bloque tendría que modificar también todos los que tiene delante, para que la información sea consistente. Por otra parte la cadena esta distribuida entre varios nodos/participantes, lo cual permite que por consenso pueda determinarse si una cadena es legitima o no en base a la información que los demás participantes de la blockchain guarden acerca de la misma. Esto dificulta la posibilidad de falsificar la información de la cadena cuantos mas participantes puedan realizar consenso sobre la misma, volviendo a blockchain una buena forma de crear una base de datos publica no relacional fiable.

1.2. Bitcoins

Una base de datos publica como la descripta en el item anterior puede tener varios usos, sin embargo para el scope de este trabajo practico nos vamos a enfocar en la aplicación de blockchain como solución para la creación de la moneda virtual Bitcoin.

El Bitcoin es una moneda virtual que busca evitar la utilización de third-parties para la realización de comercio en Internet. Una motivación por la cual esto seria de interés, es la incapacidad de generar transacciones completamente irreversibles, pues el third-party(un banco por ejemplo) no puede evitar tener que mediar una disputa del otro lado, eso hace que la transacción sea mas cara de lo que originalmente, pues la mediación tiene un costo, y por ello la cantidad mínima de una transacción para que sea viable aumenta, generando una limitación.

2. Análisis

2.1. Convergencia de blockchains

Desde un punto de vista teórico, podemos definir una situación hipotética en la cual no ocurra la convergencia. Pensemos en una red de n nodos. Cada uno de estos nodos intentará minar bloques y comunicará al resto en la eventualidad de haberlo logrado. Ahora bien, la aceptación de el bloque minado depende del estado de los nodos que reciben la notificación. Estos solo aceptarán al nuevo bloque si este posee un índice que es mayor al último bloque de las cadenas de ellos pero no lo excede en más de una unidad. Entonces si imaginamos una situación en la cual todos los nodos minan exitosamente bloques exactamente al mismo tiempo, ninguno de los mensajes va a ser aceptado, porque en ningún caso se cumple que una cadena está mas adelantada que otra. Analicemos ahora este fenómeno con una perspectiva más práctica. Si modelamos al éxito del minado como una variable geométrica con probabilidad p, donde p será mas chica cuanto mayor sea la dificultad del minado, vemos que la probabilidad de que todos los nodos resuelvan la ecuación en un momento dado es p^n. Esto representa en si a otra variable geométrica, donde la probabilidad de que en sucesivas iteraciones coincidan los eventos de minado exitoso decrece exponencialmente con el número de iteraciones. De esta manera podemos convencernos de que, en la práctica, utilizar una dificultad no trivial nos garantiza convergencia de las cadenas.

2.2. Pérdida de paquetes y demoras

La demora en los tiempos de envío de paquetes impacta negativamente en los tiempos de ejecución del thread que se encuentra esperando mensajes. Esto se debe a que la función utilizada para recibirlos *RECV* es de carácter bloqueante y, por lo tanto, permanece ahí hasta que suceda algo en el medio que se está escuchando. Así, si se tienen tiempos demasiado grandes de espera, el thread encargado de esta tarea, dedica grandes cantidades de su ejecución a esperar. Además, si la demora fuera lo suficientemente grande como para permitir que se agreguen bloques a la cadena por el thread que se encuentra minando, podría provocar una migración evitable. Es decir, se podría caer en una situación con dos cadenas de longitudes que distan en más de un bloque debido a la tardanza del mensaje. En otras circunstancias, podría evitarse esta situación y simplemente caer en algún otro caso de validación de bloque.

Ahora bien, el peor caso sería que un mensaje se pierda. Esto puede causar una situación similar a la de la migración innecesaria (mencionada en el párrafo anterior). Como idea general, si un mensaje se perdiera, el nodo al que se le enviaba nunca se enterará de su existencia y aquí se tienen dos casos:

- En un primer lugar, está la situación en la que el thread que iba a recibir el mensaje estaba simplemente en el *RECV* del ciclo que se encarga de determinar lo que se hace con el bloque recibido. Aquí la consecuencia es que ese nodo no considerará al bloque enviado y esto puede derivar en migraciones futuras.
- Por otro lado, se tiene el caso en el que el thread se encontraba migrando la cadena y esperaba una subcadena del otro nodo para acoplarla a la suya. En este caso, la consecuencia es mucho más severa puesto que este thread se quedará esperando allí por siempre.

A modo de solución se podría demandar confirmación del receptor. De no recibir la misma, se reenvía el mensaje. Algo similar a un commit de dos fases. Por otro lado, en el caso de la velocidad de tranferencia de mensajes, la solución que surge naturalmente es mejorar el medio de transferencia de los mensajes.

2.3. Conflictos entre nodos respecto a proof of work

El aumento de la dificultad del proof of work provocaría que el tiempo que requiera minar un bloque sea mayor, esto disminuiría la cantidad de migraciones por unidad de tiempo. Esto se debe a que es más fácil que los nodos mantengan sus cadenas en longitudes parejas. Como consecuencia hay un menor solapamiento de los envíos de mensajes. En consecuencia, el medio de transferencia de los mismos está menos congestionado y podría impactar positivamente en la velocidad de transferencia de los mismos.

Un aspecto positivo del aumento de la dificultad es que se evita desperdiciar tiempo en migraciones y problemas de cadenas adelantadas.

Respecto a la convergencia, así como se aclaro en la primer sección, es altamente improbable que se de la situación de no convergencia, esto se mantiene al variar el proof of work, lo que si podría influenciar el aumento de dificultad es a una convergencia mas rápida, pues al haber menos conflictos y migraciones la consistencia entre cadenas de distintos nodos se mantiene más fácilmente.