DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico Nº 2

Algoritmos en sistemas distribuidos

Sistemas Operativos
Primer cuatrimestre 2018

Integrante	LU	Correo electrónico
Travi Fermin	234/13	fermintravi@gmail.com
Ingani Bruno	50/13	chino117@hotmail.com
Lucero Emiliano	108/12	eslucero2010@gmail.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

- **Aclaraciones sobre la implementación**

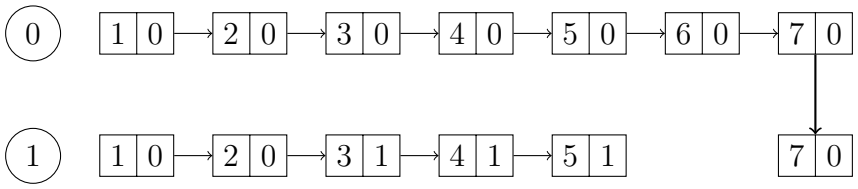
  - Los nodos no finalizan su ejecución. El thread minero de cada nodo termina una vez alcanzada una cadena de longitud `MAX_BLOCKS`, pero el thread que se encarga de recibir y responder mensajes sigue ejecutándose.
  - Para hacer que cada nodo envíe el mensaje con el tag `TAG_NEW_BLOCK` en orden distinto a los demás, se utiliza un arreglo denominado `nodos_mezclados`. El mismo contiene los *ranks* de los demás nodos; aquellos cuyo *rank* sea mayor al del nodo se encuentran primeros (de menor a mayor) y luego siguen aquellos que sean menores al del nodo (de menor a mayor). No se incluye el *rank* del nodo, y se inicializa al comenzar la ejecución.

1. Soluciones

1. **¿Puede este protocolo producir dos o más blockchains que nunca converjan?**

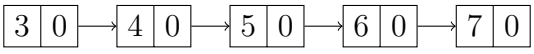
Sí, puede ocurrir, dado que el protocolo depende fuertemente de la cantidad máxima de bloques que pueden separar a un par de *blockchains* para decidir si se debe migrar hacia la otra cadena o no.

Considerar el siguiente caso, donde la ejecución consta de dos nodos y la constante `VALIDATION_BLOCKS` vale 5. El nodo circular indica el *rank* del nodo, mientras que la lista enlazada representa la cadena de cada uno. Cada nodo de dicha lista contiene el índice (elemento izquierdo) y el *owner* (elemento derecho) del bloque minado.



Como se puede observar, ambos nodos comparten el primer par de bloques minados. Sin embargo, a partir del tercero, cada uno minó su propio bloque y descartó el bloque del otro. En determinado momento, el nodo 0 minó el bloque con índice 7 y se lo envió al nodo 1.

Al recibir dicho bloque, el nodo 1 verifica las condiciones impuestas por el protocolo, y concluye que corresponde migrar de cadena ya que recibió un bloque más adelantado. Prosigue a pedirle al nodo 0 la lista con los últimos `VALIDATION_BLOCKS` bloques. Recibe la siguiente lista:

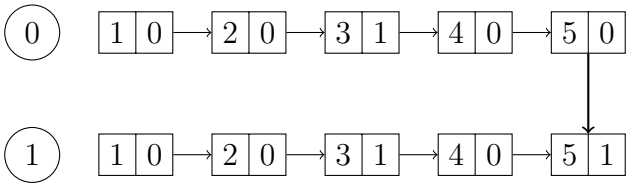


Luego, al recibirla, el nodo 1 prosigue a hacer las verificaciones correspondientes. Como es condición necesaria que al menos un bloque de la lista recibida ya debe pertenecer a la cadena, el mismo la descarta y no realiza la migración.

Esto presenta un problema para la convergencia de las *blockchains*, debido a que bloques subsiguientes minados por el nodo 0 serán rechazados por el nodo 1 y viceversa. Por lo tanto, ambos nodos continuarán con cadenas diferentes.

Es fácil ver que este problema se puede generalizar a tres o más nodos. Basta con que cada uno mine bloques lo suficientemente rápido como para rechazar aquellos producidos por los demás nodos. En cuanto la diferencia entre cada par de *blockchains* sea mayor a `VALIDATION_BLOCKS`, por más que un nodo le pida lista a otro, va a proseguir a rechazarla.

La divergencia también puede ocurrir si se finaliza la ejecución de los nodos, lo cual sucede cuando alguno de ellos alcanzó una lista de tamaño `MAX_BLOCKS`. Considerar el siguiente caso, donde dicha constante vale 5.



En este caso, ambos nodos alcanzaron el final de la lista. El nodo 0 minó el último bloque y se lo envía al nodo 1. Sin embargo, antes de que el nodo 1 reciba dicho bloque, el mismo minó por su cuenta el bloque con índice 5. Cuando recibe el bloque enviado por el nodo 0, lo rechaza. Luego, las *blockchains* divergen.

2. **¿Cómo afecta la demora o la pérdida en la entrega de paquetes al protocolo?**

   Los paquetes que se envían en el protocolo se dividen en 3 tipos:

   - **TAG_NEW_BLOCK**: Su demora o pérdida puede ocasionar que las cadenas diverjan. Supongamos que tenemos 2 nodos A y B que hasta el bloque i comparten la misma cadena, que B mina `VALIDATION_BLOCKS` bloques más mientras que A no mina ninguno y que ninguno de los mensajes enviados por B a A sobre los últimos bloques minados llegó. Si B mina uno más y se lo envía con éxito, debido a que las cadenas de A y B difieran en más de `VALIDATION_BLOCKS` bloques, A rechazaría esa cadena, ocasionando divergencia entre sus cadenas.
   - **TAG_CHAIN_HASH**: La demora o pérdida del envío de este tipo de mensaje no afecta al nodo que lo realiza, debido a que continúa su ejecución y espera un tipo de mensaje `TAG_CHAIN_RESPONSE`.
     El nodo receptor utiliza el método `MPI_Probe` sobre cualquier tipo de mensaje antes de recibir uno. Por lo tanto, no se quedará bloqueado esperando un mensaje que nunca llegó y continuará su ejecución en cuanto reciba alguno. Si se demora la recepción, puede ocurrir que el nodo que lo reciba contenga una cadena diferente a la cual tenía cuando el otro nodo hizo el pedido. Sin embargo, no es un problema ya que el nodo utiliza la estructura `node_blocks` para buscar los nodos a enviar, por lo que responderá con una cadena obsoleta pero no inválida.
   - **TAG_CHAIN_RESPONSE**: Debido a que la recepción de mensajes de este tipo lo realizamos de forma bloqueante en `verificar_y_migrar_cadena`, la ejecución quedará detenida esperando el paquete. Si este se perdiera, el thread encargado de correr `nodo` quedaría trabado en ese paso y no volvería al loop del método `nodo`, impidiendo la recepción de otros mensajes (incluyendo nuevos bloques minados). Además, como el método `verificar_y_migrar_cadena` se ejecuta sobre una sección crítica, el otro thread del nodo (el encargado de minar nuevos bloques) tampoco ejecutaría. Es decir, el nodo entero dejaría de correr.

3. **¿Cómo afecta el aumento o la disminución de la dificultad del Proof-of-Work a los conflictos entre nodos y a la convergencia?**
   Experimente variando la constante DEFAULT_DIFFICULTY.

   La dificultad determina la cantidad de tiempo que le llevará a cada nodo minar un nuevo bloque. A mayor dificultad, menor cantidad de bloques minados por unidad de tiempo. Por lo tanto, si se posee una dificultad baja (por ejemplo, si su valor es 5), aumentarán los conflictos entre nodos, ya que cada uno producirá a mayor velocidad y enviarán el mensaje `TAG_NEW_BLOCK` más frecuentemente.

   Debido a esto, la probabilidad de que surja un caso como el explicado en el punto 1 aumenta considerablemente. Es decir, la convergencia de las distintas *blockchains* se ve prácticamente imposibilitada.

   Al aumentar la dificultad, el tiempo requerido para minar es mayor. Luego, el intercambio de mensajes `TAG_NEW_BLOCK` entre los nodos se hace de manera más espaciada, disminuyendo los eventuales conflictos que puedan surgir. Si `DEFAULT_DIFFICULTY` es lo suficientemente grande (por ejemplo, si

su valor es 15 o 20) como para que no ocurra que un nodo pueda tener una cadena que contenga más de `VALIDATION_BLOCKS` bloques diferentes que los demás, las cadenas van a converger.

4. **Investigue y explique cómo maneja Bitcoin los cambios en el poder de cómputo para que la dificultad no queda atrasada.**

   El proof-of-work es esencialmente una CPU con un voto. La decisión de la mayoría está representada por la cadena más larga, que tiene el mayor proof-of-work invertido en ella. Si la mayoría de la potencia de la CPU está controlada por nodos honestos, la cadena honesta crecerá más rápido y superará a las cadenas competidoras. Para modificar un bloque pasado, un atacante tendría que volver a hacer la proof-of-work del bloque y todos los bloques posteriores y luego ponerse al día y superar el trabajo de los nodos honestos. Mostraremos más adelante que la probabilidad de que un atacante más lento se ponga al día disminuye exponencialmente a medida que se agregan los bloques subsiguientes.

   Para compensar el aumento de la velocidad del hardware y el interés variable en ejecutar nodos a lo largo del tiempo, la dificultad del proof-of-work está determinada por una media móvil que apunta a un número promedio de bloques por hora. Si se generan demasiado rápido, la dificultad aumenta.

   Supongamos que un jugador con crédito ilimitado comienza con un déficit y juega potencialmente un número infinito de pruebas para tratar de alcanzar el umbral de rentabilidad. Podemos calcular la probabilidad de que llegue al punto de equilibrio, o que un atacante alguna vez alcance la cadena honesta, de la siguiente manera:

   $$\begin{aligned} p &= \text{probabilidad de que un nodo honesto encuentre el siguiente bloque} \\ q &= \text{probabilidad de que el atacante encuentre el siguiente bloque} \\ q_z &= \text{probabilidad que el atacante alguna vez alcanzará desde z bloques detrás} \end{aligned}$$

   $$q_z = \begin{cases} 1 & \text{si } p \leq q \\ (q/p)^z & \text{si } p > q \end{cases}$$

   Dada nuestra suposición de que $p > q$, la probabilidad cae exponencialmente a medida que aumenta el número de bloques que el atacante tiene que alcanzar. Con las probabilidades en su contra, si no hace una buena estocada desde el principio, sus posibilidades se vuelven infinitamente pequeñas a medida que se retrasa.