Universidad Nacional Autónoma de México

Facultad de Ciencias

Computación Distribuida

Tarea 9

Johann Ramón Gordillo Guzmán
418046090

Tarea presentada como parte del curso de **Computación Distribuida** impartido por la profesora **M.C Karla Rocío Vargas Godoy**.

05 de Enero del 2021

Link al código fuente: https://github.com/JohannGordillo/

Actividades

1. ¿Cuál es el problema de implementar un detector de fallas en redes arbitrarias (no completas)?

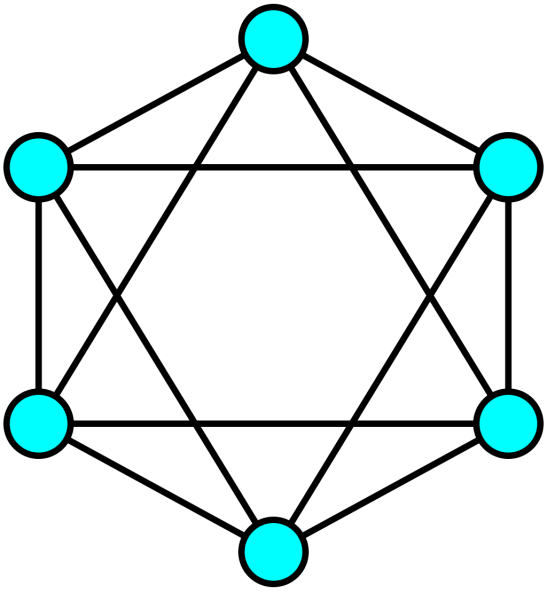
   Respuesta.

   El mayor problema que se tiene al implementar detectores de fallas en topologías arbitrarias es que se tiene que reenviar la información, por lo que se pueden formar ciclos.

   En una gráfica completa, un proceso tiene comunicación con todos los demás procesos de la gráfica, por lo que todos brindan información directa con respecto a su propio estado a los demás. Además, cuando un proceso falla, todos comienzan a sospechar pues no reciben información de éste proceso, y por ello es sencillo implementar detectores de fallos y demostrar sus propiedades.

   Por otro lado, en gráficas arbitrarias, es difícil saber si un proceso sigue con vida, pues no necesariamente un proceso está comunicado con todos los demás. Así, cuando un proceso manda señales de vida, únicamente lo hará a sus vecinos y éstos se encargarán de transmitirla a sus vecinos, y así sucesivamente. El problema es que puede llegar un punto en el que dos nodos vecinos tengan información distinta sobre el estado de un nodo en cuestión, digamos p_i. Para uno de los nodos p_i puede estar vivo y para el otro no, por lo que al intercambiar información acerca del estado de p_i entre ellos se generará un ciclo en el que no sabremos que información tomar como actual.

   Una manera en la que se podría pensar para resolver este problema es por medio de timestamps, pero estos crecen de manera no acotada, y en términos teóricos no es factible usarlos, por lo que la búsqueda de un mecanismo que permita el reenvío de información y que se sepa cuál es más reciente es otro problema. La solución, como vimos en clase, es suponer un sistema parcialmente síncrono implementado sobre una red arbitraria f-conexa con canales ADD.



2. ¿Por qué se pueden colapsar las 8 clases de detectores en solo 4?

Respuesta.

Sabemos que un detector de fallas D' es reducible a un detector de fallas D si existe un algoritmo distribuido T que pueda transformar a D en D', lo que se denota como $D \supseteq D'$. De esto, se tiene la siguiente jerarquía:

$$P \supseteq Q \qquad S \supseteq W \qquad \Diamond P \supseteq \Diamond Q \qquad \Diamond S \supseteq \Diamond W$$

Más aún, el proceso que tiene información sobre los fallos en el sistema distribuido puede avisar a los demás de su estado, por lo que podemos usar *integridad débil* para simular *integridad fuerte*. Por lo tanto:

$$P \simeq Q \qquad S \simeq W \qquad \Diamond P \simeq \Diamond Q \qquad \Diamond S \simeq \Diamond W$$

Donde $\simeq$ denota equivalencia. Es decir, se pueden colapsar las 8 clases de detectores de fallos en solo 4 de ellas.

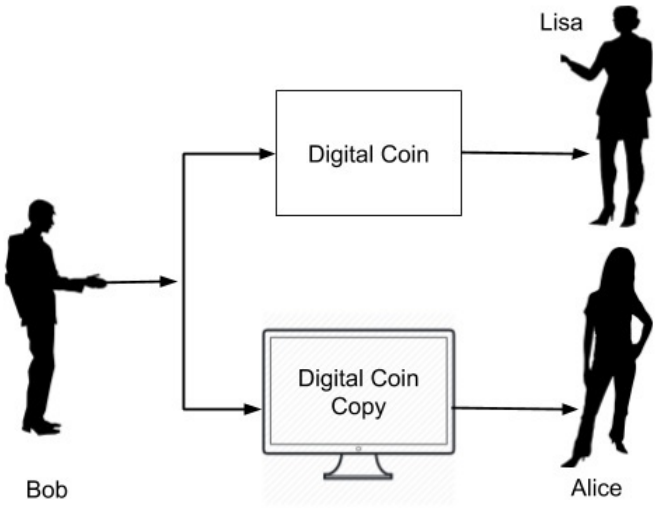
3. ¿Cuál es el problema del doble gasto?

Respuesta.

El problema del doble gasto es un defecto potencial del dinero digital, en el que una misma moneda (token) puede gastarse más de una vez.

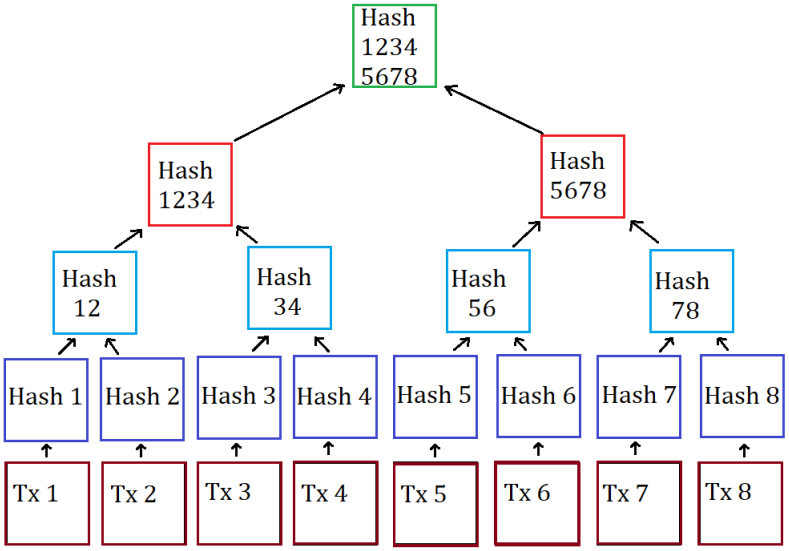
Cuando este problema está presente, un ciberdelincuente puede utilizar las mismas monedas para realizar distintos pagos.

Propuestas de dinero digital previas a Bitcoin y a Ethereum sufrían de este problema, sin embargo, Bitcoin lo solucionó utilizando una base de datos distribuida, donde se registran todas las transacciones realizadas en el sistema.



Recordemos que uno de los pasos para que una transacción sea completada en Bitcoin es la validación de las transacciones. Esta etapa es muy importante, pues la validación de transacciones permite evitar el problema del doble gasto, gracias a los árboles de Merkle que permiten ver de manera eficiente si una transacción ya se incluyó o no en la base de datos distribuida.

Los árboles de Merkle son árboles en los que cada nodo no hoja está etiquetado con el hash de la concatenación de los hashes de los nodos hijo. Se ven como sigue:

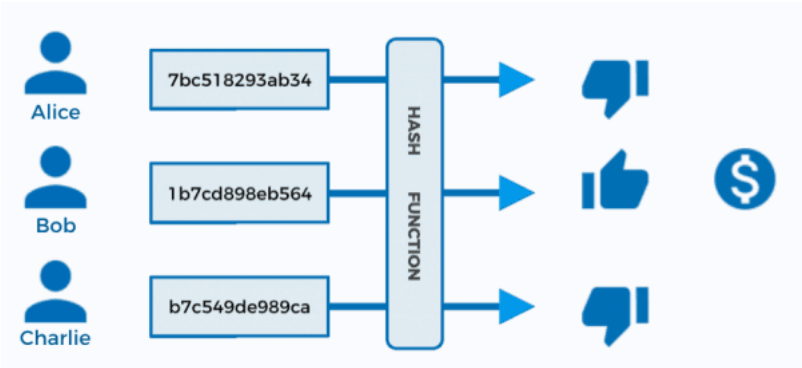


4. ¿Qué es la prueba de trabajo?

Respuesta.

La prueba de trabajo es el algoritmo de consenso (aunque no consenso como el que hemos revisado) que utilizan las cadenas de bloques para acordar que información es añadida a la cadena.

Es una competencia entre mineros que ayuda a mostrar que se han gastado los suficientes recursos computacionales para generar el bloque propuesto.



5. ¿Cómo se utiliza la prueba de trabajo para tratar de lograr consenso sobre qué bloque añadir a la cadena?

Respuesta.

La prueba de trabajo forma parte de la etapa conocida como minado de bloques y consiste en resolver un acertijo criptográfico. El minero que haya propuesto el bloque seleccionado como ganador es recompensado y su bloque es añadido a la blockchain.

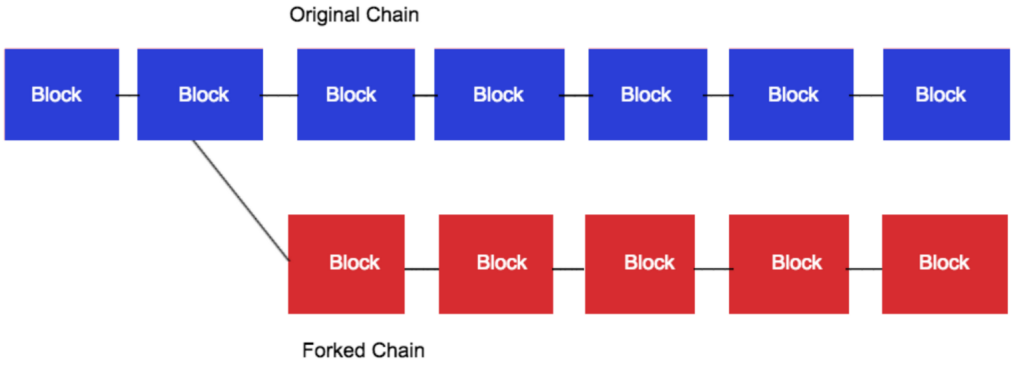
Primero los mineros calculan una dificultad objetivo d (en Bitcoin la dificultad objetivo es calculada cada 2016 bloques). Una vez hecho esto, cada minero genera un bloque B de transacciones que será propuesto. Ya que cada minero tiene su bloque propuesto, comienza la competencia: los mineros calcularán el hash $H(B)$ del bloque que propusieron y si no cumple con $H(B) > d$, modifican el *nonce* de su bloque y vuelven a calcular el hash; todo esto se repite hasta que un bloque es seleccionado como ganador.

Posteriormente, la propuesta ganadora se propaga para ser validada y finalmente se agrega a la blockchain, donde los bloques se enlazan por medio del hash del bloque previo.

6. ¿Qué es un fork? ¿Cómo se solucionan los forks?

Respuesta.

Los forks son un problema que se tiene en las cadenas de bloques cuando dos mineros resuelven la prueba de trabajo con distintos bloques al mismo tiempo. Consiste en la bifuración de la historia de la blockchain. Debido a los retardos en la propagación de un bloque los participantes pueden tener una historia distinta entre sí.



Para solucionar este problema hay que tomar la cadena sufijo más larga a partir de donde se generó el fork, y la única opción es esperar hasta que podamos tomar una decisión sobre que cadena tomar para finalmente desechar la otra.

7. ¿Cuándo se sabe que una transacción ya se hizo efectiva en una blockchain?

   Respuesta.

   Una vez que un bloque es seleccionado ganador de la prueba de trabajo, éste se propaga y cada minero verifica que el bloque propuesto satisface la dificultad objetivo. Si lo hace, añade el bloque a su blockchain.

   Cuando se confirma que se ha añadido un bloque con la transacción, hay que esperar que se añadan algunos bloques adicionales a la blockchain.

   Terminada esta etapa, decimos que la transacción se ha completado.

8. ¿Qué es lo que más te pareció importante o interesante del seminario de blockchains? ¿Por qué?

   Respuesta.

   Todo lo que se vio en el seminario me encantó. Desde los fundamentos de criptografía hasta el protocolo de Bitcoin y la breve introducción a Ethereum. Sin embargo, la parte que más me gustó fue donde revisamos el protocolo de Bitcoin, pues es una tecnología que me interesa mucho, pero no entendía nada de blockchain y no creí que el funcionamiento del sistema fuera tan complejo. Ahora entiendo por qué las personas que se dedican a minar criptomonedas usan equipo caro y especializado, pues para ser recompensados necesitan resolver la prueba de trabajo que es un acertijo criptográfico complicado y exige un alto consumo de recursos computacionales. Además, el seminario me ha ayudado a ver la importancia del cómputo distribuido en el mundo real.

Referencias

[1] Piña, M. (2020). *Introducción a Bitcoin y Ethereum*. Facultad de Ciencias, Universidad Nacional Autónoma de México.