Tarea 3

Ian Israel García Vázquez Armando Ramírez González
Ricardo Montiel Manriquez Christopher Alejandro Escamilla Soto
Jonás García Chavelas

24 de octubre de 2021

1. ¿El árbol generador de la figura 1 puede obtenerse en alguna ejecución del algoritmo BFS visto en clase? De ser el caso, describe la ejecución, y de no serlo, explica por qué no se puede. Haz lo mismo con el algoritmo DFS

BFS

a) **Paso 1**

$$Padre = A$$

$$\mathsf{send}(BFS, A)$$

b) **Paso 2**
Reciben B, C $< BFS, A >$

$$B.padre = A$$

$$\mathsf{send}(< parent >) \rightarrow A$$

$$\mathsf{send}(< BFS, B >) \rightarrow \text{Todos los vecinos}$$

$$C.padre = A$$

$$\mathsf{send}(< parent >) \rightarrow A$$

$$\mathsf{send}(< BFS, C >) \rightarrow \text{Todos los vecinos}$$

c) **Paso 3**
A recibe $< parent >$ de B y C

$$Hijos = B, C$$

Aclaración Importante: *Supondremos que los mensajes de C a D llegan ligeramente antes que los de B a D y los de B a E.*

d) **Paso 4**
D recibe $< BFS, C >$

$$D.padre = C$$

$$\mathsf{send}(< parent >) \rightarrow C$$

$$\mathsf{send}(< BFS >, C)$$

D recibe $< BFS, B >$

$$D\ \mathsf{send}(< already >) \rightarrow B$$

e) **Paso 5**
E recibe $< BFS, B >$

$$E.padre = B$$

$$\mathsf{send}(< parent >) \rightarrow B$$

$$\mathsf{send}(< BFS, e >)$$

E recibe $< BFS, C >$

$$E\ \mathsf{send}(< already >) \rightarrow C$$

f) **Paso 6**

$$Otros = E$$

g) **Paso 7**
B recibe $< already >$ de D

$$Otros = \{E\} \cup \{D\} = \{E, D\}$$

Como $Hijos \cup Otros ==$ Todos los vecinos $/\{p_j\}$; terminamos.

Termina el algoritmo y ha generado un árbol como el de la imagen, es importante notar que es gracias a la suposición de la nota aclaratoria previa, en otro caso D, E tendrían a los padres contrarios.

2. Describe el algoritmo para el problema de los k caminos disjuntos descrito en clase, argumenta que es correcto y haz un an álisis de tiempo y número de mensajes. Recuerda que en este problema tenemos $2k \leq n$ procesos distinguidos en un árbol con raíz, y el objetivo es emparejar a los procesos distinguidos de forma tal que los caminos en el árbol que conectan a las parejas son disjuntos en aristas (ver figura 2). Al final del algoritmo, todo proceso distinguido debe tener el ID de su pareja.

   Lo que tenemos que hacer s empezar por sus hojas, verificamos si el nodo esta marcado, en caso de estarlo lo agregamos a una pila y ahora revisamos al padre, si el padre tambien esta marcado lo agregamos a la pila y regresamos estos dos elementos y vaciamos la pila, en caso de que los vertices no esten marcados seguimos revisando a sus padres hasta encontrar alguno que este marcado.

   Aplicando el algoritmo de:

   Tiempo = 2 * Profundidad(T) Mensajes = 2*(—V— - 1)

3. Describe un algoritmo distribuido basado en DFS que cuente el número de procesos en un sistema distribuido cuya gráfica G es arbitraria. Al terminar de contar, debe informar a todos los procesos el resultado del conteo. Muestra que es correcto.

Algoritmo DFS(ID, soyLider): // ID $\in$ N

```
Padre = ⊥, Hijos = ∅, SinExplorar = todos los vecinos, contador,existeGra = false

   Si no he recibido algun mensaje:
       if soyLider and Padre = ⊥ then:
           existeGraf = True;
           if(existeGraf) then: contador = 1 end if
           Padre = ID
           explore(contador)

    Al recibir <M,acc> desde el vecino pj:
       if Padre = ⊥ then:
```

```
            Contador = acc++;
            Padre = j
            elimina pj de SinExplorar
            explore()
        else:
            send(<already>) a pj
            elimina pj de SinExplorar

    Al recibir <already> desde el vecino pj:
        explore()

    Al recibir <parent> desde el vecino pj:
        Hijos = Hijos U {pj}
        explore()
    procedure explore():
        if SinExplorar ≠ ∅ then:
            elegir pk en SinExplorar
            eliminar pk de SinExplorar
            send(<M,acc>) a pk
        else:
            if Padre ≠ ID then send(<parent>) a Padre
                terminate


```

Algoritmo Flood(ID, Lider, M):

```
    mensaje
    flag = false

    Ejecutar inicialmente:
        Si ID == Lider entonces: // Soy el lider
                flag = true
            mensaje = M
            send(<M>) por todos los puertos


    Al recibir algun mensaje <M> por algun puerto:
        Si not flag entonces:
                flag = true
            mensaje = M
            send(<M>) por todos los puertos

```

Ejecutamos primero al algoritmo modificado DFS para posteriormente ejecutar un algoritmo de flooding al cual se le pasara como parameto M el valor del contador obtenido del ultimo nodo en terminar el algoritmo anterior (DFS) para enviar un mensaje a todos los nodos de una grafica, donde cada uno de ellos recibira el mensaje M, que contiene el resultado del acumulador y el cual contiene la variable mensaje donde podremos consultar el resultado final.

EL algoritmo es correcto por que lo unico que agregamos al algoritmo DFS son dos variables, existeGraf que nos dira si existe por lo menos un proceso y este iniciara a la variable contador en 1, que es el primer proceso.

La ultima cosa que modificamos al algoritmo es que al enviar el mensaje $< M >$ enviamos tambien el resultado del acumulador, es decir el contador del ultimo nodo, como esta variable solo se incrementa cuando se cumple la condicion de que el padre sea nulo, evita que cometamos un error al contar, pues al igual que a cada nodo solo se le define un padre, en cada nodo solo incrementamos una vez al acumulador.

4. Describe un algoritmo distribuido basado en DFS que, en una gráfica G arbitraria con n vértices anónimos, asigne etiquetas únicas en el rango $[1 \dots n]$ a los vértices de G. Muestra que es correcto.

   Damos el siguiente algoritmo DFS modificado para etiquetar vértices anónimos de una gráfica arbitraria:

```
etiquetaDFS(soyLider):
  padre = null, Hijos = null, sinExplorar = puertos del vertice
  Si no he recibido algun mensaje:
    if soyLider and padre == null entonces:
       padre = 1
       ID = 1
       explore(ID)

  Al recibir <id,m> desde un puerto j:
    if padre == null entonces
      ID = m
      padre = id
      elimina a j de sinExplorar
      explore(m)
    else
      send(<already,m-1>) desde el pueto j
      elimina a j de sinExplorar

  Al recibir <already, m> desde el puerto j:
    explore(m)

  Al recibir <parent,n,ID> desde el puerto j:
    Hijos = Hijos U {ID}
    explore(n)


//Funcion explore
explore(n)
  if sinExplorar != 0 entonces:
    elegir un puerto i en sinExplorar
    eliminar al puerto i de sinExplorar
    send(<ID,n+1>) al puerto i
  else
    if padre != n entonces
      send(<parent, n, ID>) a padre
      terminacion

```

   Damos la siguiente afirmación del algoritmo:

   Afirmación: el algoritmo etiquetaDFS construye un árbol DFS con raíz en soyLider y todos los vértices están etiquetados únicamente desde 1 hasta n.

   Demostramos la afirmación por Inducción sobre el tiempo del algoritmo. Entonces, sea G una gráfica arbitraria conexa.

   Caso Base $t = 1$

   En el tiempo uno todos los vértices de la gráfica G tienen a su padre marcado como nulo y el mensaje está saliendo del líder que ha sido etiquetado con el número 1.

   Hipótesis de Inducción: Supongamos que el algoritmo se cumple para el tiempo $t - 1 \geq 1$.

Paso inductivo: En el tiempo t, tenemos que $t-1$ mensajes son recibidos por lo que el vértice que recibe el mensaje $t-1$ esta a distancia t del líder.

Sabemos que los vértices que se encuentran a distancia t han recibido el mensaje $< id, m >$ por hipótesis de inducción sabemos que los nodos se encuentran etiquetados desde 1 hasta $m-1$. Entonces, en el tiempo t el vértice que recibe el mensaje $< id, m >$ tiene su padre igual a $null$ por lo que su padre será establecido como id y el vértice será etiquetado con el valor de m, porteriormente será enviado el mensaje DFS y seguirán etiquetandose los vértices que aún no se hayan etiquetado.

Por lo tanto en el tiempo t el algoritmo ha etiquetado de manera única $[1 \dots m]$ nodos y tendremos un árbol DFS cuya raíz será el vértice soyLider.

Por lo tanto el algoritmo $etiquetaDFS$ cumple la afirmación y por lo tanto es correcto.

5. Modifica el algoritmo DFS para que corra en tiempo a lo más $2|V|$ y mande no más de $2|E|$ mensajes (con las aristas bidireccionales). Hint: Cuando un nodo recibe el mensaje $\langle M \rangle$ por primera vez, el notifica a todos sus vecinos pero envía su mensaje con su ID a sólo uno de ellos.

   Dado el algoritmo anterior podemos alterarlo de la siguiente manera:

```
optimoDFS(G):
padre = null, lider = -1, Hijos = null, sinExplorar = puertos del vertice v
Si no he recibido algun mensaje:
    if padre = null
        lider = id
        padre = v
        explore()

Al recibir <lider, id> desde un vertice w:
    if lider < id entonces
        lider = id
        padre = w
        Hijos = null
        sinExplorar = todo puerto de v excepto por w
        explore()
    else if lider = id entonces
        send(<already, lider>) a w

Al recibir <already, id> desde w:
    if id = lider entonces
        explore()

Al recibir <padre, id> desde w:
    if id = lider entonces
        Hijos = Hijos U {w}
        explore()

//Funcion explore
explore():
    if sinExplorar = null entonces
        elegir un puerto i en sinExplorar
        eliminar al puerto i de sinExplorar
        send(<lider, lider>) al vertice i
    else
        if padre != i entonces
            send(<padre, lider>) a padre
        else
            terminacion como raiz
```

Sea el vértice v_n el aquel con la ID maximal n; como en el peor de los casos los vértices se "agregan" al árbol uno a uno tenemos que la complejidad en tiempo es $O(n)$ que por definición de n es menor a $2|V|$. Luego, el algoritmo `etiquetaDFS` envía a lo más dos mensajes a sus puertos adyacentes y recibe a lo más dos, por lo que a lo más se envían $4n$ mensajes, pero en el algoritmo `optimoDFS` tenemos que en el peor de los casos, cada vértice intenta "armar" un árbol DFS, entonces la cantidad de mensajes de `optimoDFS` es a lo más m veces la de `etiquetaDFS`, es decir, nm, finalmente podemos afirmar que $2|E| \geq nm$ recordando que en un árbol el número de vértices y aristas es el mismo.

6. *a*) Describe un algoritmo distribuido para construir un árbol generador sobre una gráfica arbitraria G, usando el algoritmo de elección de lider simple, para poder determinar la raíz del árbol y también el algoritmo de BFS. Analiza la complejidad de tiempo y de mensajes.

   Simplemente ejecutamos al algoritmo eligeLider(ID, total)

```
Algoritmo eligeLider(ID, total): //  total = |V|

   Lider = ID, ronda = 0
       Ejecutar en todo momento t ≥ 0:
       send(<LIDER>) a todos los vecinos

       Al recibir mensaje de todos los vecinos en tiempo t ≥ 1:
           Mensajes = {<l1>, <l2>, ..., <ld>} // d grado
       Lider = max(Mensajes)
       ronda = ronda + 1
       Si ronda == total entonces:
       terminar


```

   Al terminar solo hara falta ejecutar BFS con la variable Lider resultado de el eligeLider:

```
Algoritmo BFS(ID, soyLider): // ID ∈ N
  Padre = ⊥, Hijos = ∅, Otros = ∅

   Si no he recibido algun mensaje:
     If soyLider and Padre = ⊥ then:
       send(<BFS, ID>) a todos mis vecinos
       Padre = ID

   Al recibir <BFS, j> desde el vecino pj:
     if Padre = ⊥ then:
       Padre = j
       send(<parent>) a pj
       send(<BFS, ID>) a todos los vecinos excepto pj
    else:
      send(<already>) a pj

  Al recibir <parent> desde el vecino pj:
    Hijos = Hijos U {pj}
    if Hijos U Otros tienen a todos los vecinos
    excepto Padre then:
      Terminar

  Al recibir <already> desde el vecino pj:
    Otros = Otros U {pj}
    if Hijos U Otros tienen a todos los vecinos
    excepto Padre then:
      Terminar


```

Como vimos en clase, el algoritmo eligeLider trabaja con una complegidad en Tiempo = d (distacia maxima) mientras que el algoritmo BFS tarda lo mismo, por lo tanto si ejecutamos los dos algoritmos, uno despues del otro la compejidad de tiempo total sera de dos veces d; Complejidad Tiempo = 2d La compejidad de Mensajes en el algoritmo eligeLider es de $2d\|E\|$ donde $\|E\|$ = numero de Aristas y ademas sabemos que la complejidad de mensajes en el algoritmo BFS es —E— y como estamos ejecutando un algoritmo despues de otro, el resultado de sumar las dos complejidades es : Complejidad de Mensajes = $2d2\|E\|$

b) El algoritmo puede ser mejorado en la complejidad de tiempo si se ejecutan de forma paralela los dos algoritmos anteriores, es decir, si se ejecuta la elección de lider y la construcción del árbol BFS. Da un algoritmo que haga esto y muestra que es correcto. Adicionalmente compara el tiempo respecto al algoritmo anterior

```
BFSImproved(ID):
padre=null, lider=-1, Hijos=null, Otros=null, flag=False
Si no he recibido algun mensaje:
    if padre=null and flag = False then:
        padre=id
        flag=True
        send(<leader, id>) a todos los vecinos


Al recibir <leader,id> desde el vecino pid:
    if lider < id then:
        lider=id
        padre=pid
        send(<parent>) a  pid
        send(<leader, id>) a todos lo vecinos excepto pid
    else if lider=id then:
        send(<already, lider>) a pid

Al recibir <parent> desde el vecino pid

    Hijos = Hijos U {pid}
    if Hijos U Otros tiene a todos los vecinos excepto Padre then:
        Terminar

Al recibir <already> desde el vecino pid:

    Otros = Otros U {pid}
    if Hijos U Otros tienen a todos los vecinos excepto Padre then:
        Terminar
```

Observemos que como primer paso, se selecciona un nodo al azar, este mismo es nombrado padre y este código no volverá a ser ejecutado gracias a la bandera $flag$, al recibir el mensaje observemos que el primer valor de variable $lider = 1$, el cual será intercambiado por el valor del ID, cualquiera que este sea ya que $ID \in \mathbb{N}$, consideremos que este mensaje es enviado a todos los vecinos del nodo seleccionado, y todos los nodos que han recibido este mensaje confirmarán ser hijos del nodo a través del mensaje $< Parent >$; algo que puede quedar señalado es el hecho de que si se trata de una gráfica conexa K-completa, finalizará casi inmediatamente tras este paso, pues todos serán marcados como hijos y habrá finalizado, por otro lado se continuará ejecutando el algoritmo hasta que todos estén marcados como Hijos u Otros, dejando también un nodo maximal llamado líder que finalmente será el padre, observemos que la complejidad $M = 2|E|$ y la complejidad en tiempo será $T <= |2V|$ gracias a la lista de Hijos y Otros que terminan con el proceso en el momento que se han abarcado todos los vértices y han recibido mensaje de su padre. Consideremos que el algoritmo anterior nos da una complejidad de mensajes $M = 2d2|E|$, tal que d es el diametro,

y una complejidad en tiempo $T = 2d$, y nuestro algoritmo, logra una complejidad $M = 2|E|$ y $T = d$, una mejora sustancial en terminos de tiempo, gracias a la ejecución simultanéa de líder y BFS.

7. Reportes:

Reporte - Ian Israel García Vázquez

La charla que seleccioné a estudiar fue *Byzantine-tolerant Distributed Grow-only Sets: Specification and Applications* impartida por el Dr. Antonio Fernández Anta; él menciona que tras diversas pláticas con investigadores y expertos en el área de blockchains, encontraba díficil comprender las verdaderas funciones del blockchain y aquello que le era intrínseco, de alguna forma veía la ambiguedad en su definición, ¿Qué es? Fue la premisa principal para comenzar a interesarse en él, a pesar de ello no se lograba diferenciar si este era un mecanismo o un servicio, como si fuera impresindible entre la criptomoneda y el blockchain, es decir, eran iguales o solo componentes de un ambiente; yo aún me hago la misma pregunta, y es que apesar de ser un sistema distribuido, no se han logrado garantizar o crear uniformidad entre los distintos blockchains y sus componentes como sistemas distribuidos.

En 2018, como primer trabajo, definen la abstracción básica de un blockchain, como un objeto sobre el cual se opera llamado *ledger*, y este mismo almacena una secuencia de eventos (transacciones, bloques, operaciones), tiene en su forma más sencilla solo añadir registros al final y por otro lado permite ver su contenido.De la misma forma se encuentra un servidor que centraliza todas las operaciones sobre el registro, finalmente esto nos lleva a tener a un conjunto de servidores que interactúan entre si, para permitir transacciones entre usuarios, de tal forma que se mantenga el orden en cada registro, donde cada cliente debe obtener información coherente y consistente sobre las transacciones.

Actualmente el mundo cuenta con varias y diferentes tecnologías blockchain, y está tecnología tiene múltiples cadenas creadas, un ejemplo muy ilustrativo es la transferencia e intercambio de bienes de manera atómica, es decir, si alguno de los dos clientes anula su aportación ó simplememente no la hace, la transacción debe ser cancelada; parte de la solución implica, colocar el bien intercambiable en un punto neutro entre ambos, si el cliente A recibe el pago correspondiente, el bien es entregado, en caso contrario, le es devuelto, claro utiliza un tiempo límite de espera para ello, aunque si alguno de ellos o ambos fallan, podría ser una situación irreparable; sin embargo al colocarle una DLO auxiliar, permite guardar parte del proceso en ella y está mantiene una toleracia a fallos que permitirá llevar a un término seguro la transacción.Asimismo más adelante se habla de un conjunto de registros, que permite realizarlo sin consenso, facilitando las transacciones de alguna forma, sin embargo es advertido por el Dr. Fernández, la implementación puede ser sumamamente compleja.

Finalemente, me ha parecido bastante fascinante la serie de resultados presentados, tal que estos mismos están comenzando a formalizar los Blockchains y mostrando las diversas posibilidades de manejo de transacciones, así como las soluciones y posibles implementaciones para los DLO y su aportación en seguridad y potencia para aquellas transacciones en blockchain y tal vez par alguna otra área y aunado a ello la gran importancia del computo distribuido en la actualidad , y como esto mismo actúa como una herramienta potente para las áreas aún dispersas como Blockchain.

Reporte - Armando Ramírez González

La plática que resultó de mi interés fue aquella dada por Baltazar Rodríguez Tellez, Client Technical en IBM, llamdas *Blockchain aplicaciones empresariales: Más allá del crypto*. La plática inició con una introducción del surgimiento de la contabilidad, en especial, se nos presentó a *Luca Pacciolli* como el inventor de la balanza contable así como la idea de partida doble, conceptos que han sido muy importantes en el ámbito de la contabilidad. A su vez se nos mencionó como estos inventos llegaron a revolucionar la forma en como se realizaban las operaciones contables y se han mantenido prácticamente iguales desde su invención, alrededor del año 1940, en palabras de Baltazar podemos ver estas creaciones como *Ïnvenciones bonitas, tan bonitas que nadie las ha modificado"*.

Posteriormente, se nos habla de lo qué suele ocurrir con la contabilidad en las organizaciones, llamada fricción, aspecto que a una organización suele costarle demasiado dinero resolver por lo que estas buscan la mejor manera de evitar que esta fricción aparezca en el ámbito contable y un ejemplo que se nos dió de esto situación fue por un lado con el Registro Público de la Propiedad, organismo en el que se hizo necesario tener muchos candados en sus documentos para poder garantizar la propiedad de alguien sobre una casa, terreno, entre otros tipos de propiedad, todo con el objetivo de reducir la fricción y garantizar la seguridad en las operaciones realizadas. Por otro lado se nos expuso el ejemplo del Servicio de Administraciób Tributaria, lugar en donde Baltazar trabajó durante un período y que gracias a ello y después de realizar muchas investigaciones y trabajar de la mano con muchos proveedores que facturaban para el SAT, logró la implementación de lo que hoy es conocida como la e-firma portable; elemento basado en ideas de blockchain y criptografía que le permitió a las personas poder facturar de forma más fácil, reduciendo gastos y en especial, reduciendo la fricción en las organizaciones.

Pero blockchain se presenta como una buena balanza distribuida, ya que todos los nodos presentes en el sistema válidan la información y observan todo dentro de la balanza, entonces ningún participante tiene más poder que el de a lado. De igual manera blockchain crea una confianza distribuida gracias a la desconfianza de nodos, porque sí todos desconfían todos revisan; tenemos la implicación desconfianza distribuida entonces confianza sitémica. Todo esto sumado a que un sitema distribuido atiende aquello temas que generan mayor fricción, por lo que al disminuir la fricción también disminuyen costos y erradican el fraude en las empresas.

Entonces la tecnología blockchain nos ofrece muchas ventajas al momento de usarse, ya que también nos brinda una herramienta eficaz, conocida como los contratos inteligentes que son creadas bajo reglas de negocio acordadas, verificadas y compartidas; lo que nos permite automatizar procesos y nos ofrece la ventaja de reducir errores, reducir costos, incrementar la incertidumbre y reducir tiempos en la detección de problemas.

A lo largo de la conferencias se nos presentan diversas aplicaciones de blockchain y algunos de ellos son que proporcionan trazabilidad, lo que nos permite determinar el origen así como las ubicaciones en las que han estado alimentos, minerales u otros productos dentro de la cadena de producción cuyo proceso quiere conocerse por ejemplo, para la detección de alimentos contaminados. En al ámbito de los fármacos, para poder eliminar la piratería en los prodcutos; en el ámbito bancario donde tener un sistema de identidad basado en blochchain le permitiría a los bancos reducir costos en la acción de tener que almacenar los datos de cada unos de sus clientes por cada banco en el que un cliente quiere abrir una cuenta; y por último en el ámbito de los seguros, donde gracias a blockchain se podría erradicar los fraudes a aseguradoras registrando y validando las facturas y pólizas en un sistema distrbuido como blockchain.

La conferencia me resultó interesante ya que a lo largo de mi vida el concepto de blockchain lo relacionaba inmediatamente con bitcoin y todo el mundo de las ciptomonedas, algo que no es totalmente correcto. La tecnología blockchain tiene muchas más aplicaciones en diversas organizaciones y puede ayudar a automaitzar y resolver diversos problemas, aunque el blockchain no debe ser tomada como

una tecnología única y universal, ya que esta no sirve para todo y existen muchos tipos de blockchain diseñados para resolver ciertos problemas.

Reporte - Jonás García Chavelas
El tema que más me llamó la atención fue *How to make blockchains more secure and scalable* del ponente Vincent Gramoli, quien dio inicio al seminario y principió con una breve introducción al concepto del *blockchain* y los persistentes problemas que éste plantea para la computación distribuída. En partícular, expusó que el omnipresente problema bizantino del concenso tiene como consecuencia que varias máquinas en una red pueden llegar a diferentes puntos de vista acerca del estado de la misma, en la práctica esto puede ser grave pues si tenemos una transacción ya realizada, podemos usar las mismas unidades monetarias para otra transacción, prácticamente creado dinero, como una especie de error de duplicación. Esto fue interesante para mí, porque por lo abstracto del problema del concenso bizantino, nunca había pensado en sus ramificaciones más concretas.

El Dr. Gramoli continua al explicar que Gavin Wood presentó una especie de solución al problema acuñada *"smart contract"*[6] (que de acuerdo con la ponencia de Maurice Herlihy: *"it's a marketing term: it's not smart and it's not a legal contract"*). Este sistema va un poco más allá de las transacciones directas, lo que hace es adjunta un *script* a la red, el cual tiene una función determinista que podrán ejecutar todos los miembros de una red cuando algún cliente realice una transacción, esto para mantener la consistencia de la red. En este punto, es cuando la exposición de Vincent se empieza a transformar en una suerte de relato, que fue una de las cosas que me gustaron ésta. Un relato que se enfoca en mostrar distintos tipos de ataques que son posibles dependiendo de la variante de protocolo de consenso que Ethereum use.

Gramoli cuenta que buscaba entender más el sistema que Ethereum usa para mantener el consenso en sus redes, ¿cómo pueden los miembros de la red acordar en el mismo estado para algún bloque de la *blockchain*? Lo anterior llevo descubrir una vulnerabilidad[5] porque al revisar el código observó que estaba implementado de manera que escencialmente para algún índice es posible tener más de un bloque, generando una bifurcación que puede ser mantenida por mucho tiempo. Una implementación de este tipo de ataques fue llevada a cabo en 2016[4] contra un sistema de prueba de trabajo, sistema contra el cual también se demostro un ataque de intermediario[2] que Vincent nota, puede ser una amenaza partícular para redes privadas. Además de los sistemas de prueba de trabajo, también los hay por prueba de autoridad contra los que uno de los estudiantes de Gramoli desarrollo el novedoso ataque de los clones[3].

Lo que todos estos ataques tienen en común es el uso de retrasos para generar una inconsistencia en la red, por lo que crear redes tolerantes a esto es deseable y lógico, y volviendo al título de la conferencia, idealmente quisieramos una solución que pueda ser ampliable a redes de gran tamaño. También es importante no asumir una sincronía total del sistema distribuído pues es lo que lleva a la proliferación de éste tipo de fallos. Una manera en la que podemos intentar solucionar el problema es muy familiar, es el uso de un nodo líder que administra las acciones de los demás para llegar al consenso, pero esto tiene muy mala escalabilidad, lo cual es bastante intuitivo si se toma en cuenta lo centralizado que es. En cambio, en una situación sin líderes podemos mantener el tiempo de transmisión constante incluso si varia mucho el tamaño del sistema porque la carga se reparte entre los nodos del sistema, y finalmente esto nos lleva a un algoritmo que presenta Gramoli con estas características[1].

En conclusión, la exposición me pareció muy interesante por como da a conocer estos conceptos muy partículares acerca de lo que muchos asumen es un sistema muy seguro, poniendo en evidencia que esto no es necesariamente cierto, y quizás sugiriendo que siempre se pueden imaginar soluciones más óptimas si se es un poco curioso sobre las cosas.

Reporte - Ricardo Montiel Manriquez

La platica que fue de mi interés fue dada por el Dr. Antonio Fernández Anta donde el expone sobre Byzatine-tolerant Distributed Grow only Sets: Specification and Applications. El nos habla sobre lo que es el Blockchain y del problema que con el que se encontró en el 2017 (en un año sabático que decidió tomar) fue que hablando con investigadores no lograba entender el servicio y que suposiciones hacia del sistema distribuido en el que se apoyaba, así como también de las confusiones que había entre la moneda, blockchange y el bitcoin.

Ante esta problemática en el 2018 realizo un trabajo que trataba la abstracción básica de un blockchange, donde lo definen como un objeto sobre el que van a operar al que laman ledger el cual almacena una secuencia de eventos y en su forma mas sencilla solo dos operaciones: Append(r) que es utilizada para añadir algún contenido y Get() para devolver una copia del contenido añadido. Despues nos presenta un servidor centralizado donde se recopila todas las operación Append(r) y Get() que se lleguen a realizar, todo esto nos lleva a tener un conjunto de servidores que interactúan entre si utilizando las mismas operaciones con el fin de permitir transacciones entre usuarios manteniendo un orden y sin importar del lugar donde se este realizando la operación y también todos los clientes obtengan información coherente y consiente sobre sus transacciones.

Nos comenta el Dr. que el futuro nos depara muchos blockhanges en el mundo con una gran variedad de tecnologías (bitcoin, cosmos, etc) y ademas de una misma tecnología hay múltiples cadenas creadas, ya solo la gente elige el código con el que le interese trabajar. Nos presenta Atomic Swap que es una operación que involucra múltiples cadenas tenemos dos clientes, uno de ellos tiene un bien (carro), y el otro cliente tiene otro bien (criptomoneda) y lo que se busca es que el cliente A le transfiera el coche al Cliente B y el Cliente B le transfiera la criptomoneda al Cliente B. Si alguno de los clientes anula la operación la transacción simplemente debe de ser cancelada, pero también hay una posibilidad de fallo lo cual seria una situación irreparable, para esto se tiene una DLO auxiliar que permite guardar parte del proceso de la operación en ella, mantiene una tolerancia a los fallos la cual permitirá llevar la transacción de forma segura.

También se habla de un conjunto de registros donde se permite realizarlo sin consenso, así facilitando las transacciones, pero su implementación puede ser sumamente compleja.

Para concluir, la platica es sumamente interesante porque te expone el problema inicial (Blockchange) todo lo que hay detrás de el como sus problemáticas y como darles soluciones, empezando por explicar que es un blockchange y al mismo tiempo mostrándonos el manejo de transacciones, su funcionamiento básico y los problemas que pueden surgir de estas también y derivado de esto nos da a conocer el manejo a fallos utilizando DLO's dándonos a conocer su importancia gracias a las aportaciones en seguridad que nos brinda, te das cuenta de la inmensidad de los blockchange y como el computo distribuido influye en el para su buen funcionamiento.

Reporte - Christopher Alejandro Escamilla Soto

La platica que escogi fue la que impartio el Dr. Francisco Rodriguez Henriquez exponiendo Viaje al centro de la criptomoneda. Empieza definiendo el dinero y sus tres funcionalidades: Medio de cambio, Unidad de medida y Deposito de Valor, asi como importancia porque es simbolo de soberania, nos permite un control en la economia e influye en deciciones politicas.

Entocnes el bitcoin como sustituto del dinero viendolo desde el punto de vista economico es de las monedas mas lentas en el mundo por su tasa de transacciones por segundo en promedio de 5, debido a

sus grndes fluctuaciones lo limitan a ser utilizado como unidad de medida.

Nos habla de los blockchain y que estas son el centro de las criptomonedas, estas se encuentran almacenadas de forma ordenada, cada cliente puede almacenar de forma independiente una copia de blockchaing. Hace mencion a los mineros que son los encargados de la organizacion de los bloques generados posteriormente del bloque genesis.

Hay datos curiosos como el que existe un Bitcoin halving que conciste en que cada vez que se registran 210,000 bloques se reduce el ritmo de creacion de monedas y tambien solo se generan 21 millones de Bitcoins, se tiene estimado que se dejen de producir para el año de 2140.

Se hace la pregunta de si el Bitcoin es antiecologico debido a la complejidad de hallar el nonce correcto que es de $1/2^{45}$ y esto tiene un consumo en energia de 110 TeraWatt por hora al año siendo un consumo parecido al de Suecia o Malasia siendo asi la empresa mas dilapadora de energia de todos los tiempo, siendo esto falso.

Hay personajes con tanta influencia en el mercado que incluso se han creado bots para que cada vez que Elon Musk haga un tweet sobre criptomonedas estos bots las compren, y se predice tambien que el bitcoin rompera la barrera de 100k dolares en este año.

Para concluir con este reporte, la bitcoin en pocas palabras llego para quedarse, es un tema muy controversial para los gobiernos e incluso para la gente misma pero un tema muy amplio y de gran interes.

Referencias

[1] Tyler Crain, Christopher Natoli y Vincent Gramoli. «Red Belly: a secure, fair and scalable open blockchain». En: *Proceedings of the 42nd IEEE Symposium on Security and Privacy (S&P'21)*. 2021.

[2] Parinya Ekparinya, Vincent Gramoli y Guillaume Jourjon. «Impact of man-in-the-middle attacks on ethereum». En: *2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS)*. IEEE. 2018, págs. 11-20.

[3] Parinya Ekparinya, Vincent Gramoli y Guillaume Jourjon. «The attack of the clones against proof-of-authority». En: *arXiv preprint arXiv:1902.10244* (2019).

[4] Christopher Natoli y Vincent Gramoli. «The balance attack against proof-of-work blockchains: The R3 testbed as an example». En: *arXiv preprint arXiv:1612.09426* (2016).

[5] Christopher Natoli y Vincent Gramoli. «The blockchain anomaly». En: *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*. IEEE. 2016, págs. 310-317.

[6] Gavin Wood y col. «Ethereum: A secure decentralised generalised transaction ledger». En: *Ethereum project yellow paper* 151.2014 (2014), págs. 1-32.