Computación Distribuida
Tarea 11

González Montiel Luis Fernando

25 de noviembre de 2019

1. Sincronizadores

   ¿Qué pasa si usamos un sincronizador sobre un sistema síncrono?
   Respuesta: Un sincronizador junto con sistema síncrono puede simular un sistema asíncrono.

   Explica cómo funciona la noción de proceso seguro y qué problema resuelve.
   Respuesta: Usando el reconocimiento de mensajes, es sencillo para un proceso pi saber cuándo es seguro con respecto al pulso r. Esto ocurre cuando ha recibido el reconocimiento asociado con el mensaje enviado durante el pulso r. Un problema crucial de los sincronizadores es que dentro de la ronda r, un proceso no puede saber cuál de sus vecinos le mandó el mensaje, ésto se soluciona agregando al sincronizador la noción de proceso seguro.

2. Auto-estabilización.

   ¿Se sabe cuándo converge un algoritmo auto-estabilizado?
   Respuesta: Como el valor máximo de φ es $O(n^2)$, implica que el algoritmo converge en $O(n^2)$ pasos.

   ¿Cuál es el problema del ghost root para obtener un Spanning Tree?
   Respuesta: Por alguna razón no llegó a algún nodo x, y este conserva los valores de la ejecución anterior. Entonces este nodo x tiene un ID raíz menor al de cualquiera de sus vecinos, por lo que ellos tomarían el rx de x como suyo, y estimando su distancia a la distancia inalcanzable de esa raíz fantasma (ghost root), esto se repetiría con el resto de los nodos volviendo todo rv = rx, pero esta ejecución nunca converguería a una configuración estable ya que dicha raíz no existe y la distancia a ella solo aumentaría en cada ronda. Para solucionar este problema necesitamos una cota D que nos aproxime la distancia máxima que puede tener un camino mínimo entre cualesquiera dos nodos.

3. Árboles generadores de peso mínimo.

   ¿Qué es un fragmento?
   Respuesta: Un fragmento es una parte de la gráfica que a su vez, es parte del árbol generador de peso mínimo, un fragmento puede ser desde un nodo o varios que sean conexos.

4. Teorema CAP

   ¿Qué significa cada una de las siglas de CAP?
   Respuesta: Consistency, Availability, Partition Tolerance.

   ¿Qué dice el teorema CAP?
   Respuesta: Enuncia que es imposible para un sistema de cómputo distribuido garantizar simultaneidad.

5. Elección de líder en anillos.

   ¿En qué consiste cada una de las 4 fases del algoritmo sobre anillos bidireccionales?
   RESPUESTA:
   Fase 1: Cada proceso envía un mensaje ELECTION(idi,r,d) a sus dos vecinos de ambos lados, donde IDiessu identificador, r = 0 (este es el número de ronda asociado con el mensaje) y d = 1 qué es el proceso que ya ha sido visitado por este mensaje.
   Fase 2: 1.-Cuando los procesos reciben el mensaje ELECTION por la izquierda con etiqueta (resp., righti) 2.1.- Si el id recibido es mayor a tuyo y si la d es menor a 2 a la r entonces envía un ELECTION con el Id del recibido, la ronda actual y d +1 a la derecha con la etiqueta (resp., lef ti) 2.2.- Si el id recibido es mayor o igual al tuyo y la d es mayor o igual a 2 a la r entonces envía un REPLY con el id del mensaje recibido y la ronda a la izquierda (resp., righti) 2.3.- Si el id del recibido es menor al tuyo entonces skip 2.4.- Si el id del recibido es igual al tuyo entonces envías un ELECTED (id) on lefti, electedi ¡- true
   Fase 3: 1.- Cuando se recibe un REPLY(id ,r) con la tupla un id y la ronda, si se recibe por la izquierda con la etiqueta (resp., righti) 2.- si el id recibido es diferente al tuyo entonces envía un REPLY(id, r) a la derecha (resp., lef ti) pero si el REPLY se recibió por la derecha (resp, lef ti) entonces envía un ELECTION con el id del recibido y la ronda recibida + 1 y la d = 1 a ambos lados.
   Fase 4: 1.- Cuando se recibe un ELECTED(id) solo el id por la derecha 2.- el líder ← id; done ← true 3.-si el id recibido es diferente al tuyo entonces el elegido es falso si no se envía un ELECTED con el id recibido a la izquierda.

   Los algoritmos vistos en clase, ¿Solo funcionan para anillos?
   RESPUESTA: Este algoritmo corre sobre una estructura llamada anillo, los algoritmos que vimos se pueden implementar en anillos unidireccionales y bidireccionales, además que dependiendo la implementación el tiempo de ejecución de esta mejora o empeora según sea el caso.

6. Elección del coordinador probabilística.

   ¿Es necesario saber cuántos turistas participan?
   RESPUESTA: No es necesario, no se pide que sea un número específico.

   ¿Cuál es la principal diferencia entre el algoritmo aleatorio y el determinístico?
   RESPUESTA: En el primer caso, cuando dotamos de un contador y un conjunto de instrucciones, a la vez que un algoritmo que nos de valores aleatorios, puede ser mucho más complicado porque dependemos principalmente de la aleatoriedad, pero justamente esta misma nos permite asegurar que todos lleguen al mismo lugar. Para el segundo al dar valores únicos, minimizamos la cantidad de instrucciones necesarias para terminar. Asegurando que el tiempo de terminación será igual, siempre y cuando tengamos la misma cantidad de procesos; haciendo que el tiempo de terminación dependa del conjunto de procesos.

   ¿Cuál es la topología de la gráfica que se utiliza en este algoritmo?
   RESPUESTA: Es un modelo que se tienen dos servidores y procesos que se comunican al mismo tiempo.

7. Topología en sistemas distribuidos.

   Menciona cómo se resuelve el problema de “The muddy children” si k = 1, con k el número de niños sucios.
   RESPUESTA: Tenemos solo 1 niño ”sucio” entonces cuando el maestro dice que hay 1 niño ”sucio” éste podrá ver que los demás están limpios por tanto el es el sucio y a la hora el se expone como niño con un grano en la frente.

8. Coloración de vértices en log * (n)

   Explicar bien qué es log * n
   RESPUESTA: Es el número de veces que podemos iterar log(n) hasta que llegue a 1 o menos. log*n:= 0 si n

menor o igual a 1. log*n:= 1+log*(log n) si n mayor a 1.

¿Por qué el algoritmo 6-color tiene un tiempo de complejidad log*n ?
Respuesta: Definamos la función: f(n)= 1 cuando n=2 y 1+log(f(n)) en otro caso n es el número de bits, y por hipótesis tenemos que n de tamaño log n en la primera ronda por definición de log*n tenemos que : f(n)-1= log*n
Por lo tanto: el algoritmo 3 termina en log*n

9. Un detector de fallas eventualmente perfecto para redes de topología arbitraria conectadas por canales ADD usando valores Time-To-Live

¿Cuál es la finalidad de los TTL?
Respuesta: Los TTL se usan comúnmente para limitar la vida útil de paquetes en una red o el número de saltos que un paquete puede tomar. Usamos esta idea en la implementación P, para resolver el desafío de enviar mensajes de tamaño pequeño.

¿Por qué es necesario extender la definición de integridad y precisión en redes particionables?
Respuesta: Los TTL se usan comúnmente para limitar la vida útil de paquetes en una red o el número de saltos que un paquete puede tomar. Usamos esta idea en la implementación P, para resolver el desafío de enviar mensajes de tamaño pequeño. dos procesos correctos p y q conectados por un camino correcto en la gráfica final, eventualmente conoce p y q eventualmente no sospecha de p.

10. Variantes del Consenso.

¿Cuál es el principio básico sobre el cual se basa la implementación de acuerdos simultáneos de decisión temprana?
Respuesta: Cuando se busca un acuerdo simultáneo, los procesos deben acordar cuantos accidentes tienen que ocurrir para poder decidir antes de la última ronda (ronda r+1). Cuando Y procesos se bloquean simultáneamente durante la ronda r, en el sentido de que todos los procesos que terminan esta ronda detectan estos bloques, la simultaneidad de estos bloqueos permite guardar (Y-1) rondas, es decir, los procesos pueden decidir con seguridad durante la ronda t+1-(Y-1).

Una ronda libre de fallas es necesariamente una ronda limpia? ¿Sí? ¿No? ¿Por qué?
Respuesta: Una ronda limpia no es necesariamente una ronda libre de fallas o una ronda atómica. Es posible que un proceso p se bloquea en una ronda limpia r pero ningún proceso activo al final de r notó su bloqueo (p se bloqueó después de su fase de envío y antes del final de la ronda r, o más generalmente se bloqueó durante r después del envío su mensaje a los procesos que terminan la ronda r).

11. Leader crash recovery.

¿Qué condición es necesaria para que el algoritmo elija un líder?
Respuesta: Revisar si el candidato a líder se comunica de manera correcta con una mayoría de procesos. En este caso, selecciona el proceso con menor número de penalties. Finalmente si un proceso q se considera a sí mismo como un líder, entonces cada proceso p toma a q como líder, en otro caso no se selecciona a un líder.

¿Qué ocurre cuando un proceso es líder y falla después de haber sido elegido?
Respuesta: Sin embargo, puede suceder que q haya fallado realmente, o haya sufrido una omisión de mensaje con respecto a p. Por tanto, la recepción de mensajes de p con q es establecida como FALSE, y también el valor de si p se considera a sí mismo como líder leaderCandidatesp[p]. Entonces, p llama el método UpdateLeader().

12. Consenso fallas bizantinas.

¿Cuál es el la mejor cota que podemos dar para t para resolver el consenso en sistemas síncronos con fallas bizantinas?
RESPUESTA: El algoritmo presentado en este artículo resuelve el consenso en el modelo BSMPn.t con una cota óptima de $t < \frac{n}{3}$ fallas. Se conoce como recolección de información exponencial, o EIG (exponential information gathering), ya que el tamaño de sus mensajes crece exponencialmente.

¿Qué cota para t tiene el algoritmo que resuelve éste problema con mensajes de tamaño constante?
RESPUESTA: El algoritmo resuelve el consenso en el modelo BSMPn.t y usa con mensajes de tamaño constante, pero requiere una cota superior muy restringida sobre el número de fallas, $t < \frac{n}{4}$.