UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

Tarea 2

INTEGRANTES

Torres Valencia Kevin Jair - 318331818
Aguilera Moreno Adrián - 421005200
Natalia Abigail Pérez Romero - 31814426

PROFESOR

Miguel Ángel Piña Avelino

AYUDANTE

Pablo Gerardo González López

ASIGNATURA

Computación Distribuida

15 de septiembre de 2022

1. Considera el algoritmo de Flooding visto en clase. Demuestra el siguiente corolario:
Todo proceso p_i que ejecuta el algoritmo de `Flooding`, recibe M en tiempo a lo más el diámetro $diam(G)$ de la gráfica del sistema distribuido.
Dem. Procedamos por inducción en el número de procesos. Supongamos que nuestros procesos viven en la gráfica G conexa.

- Observemos que pasa cuando solo hay un proceso en G, entonces este es el lider y además tiene el mensaje `M` desde antes de la primer ronda[I] (desde la ronda 0).
- Supongamos que la gráfica G, con k procesos, cumple con poder inundarse[II] en tiempo a lo más `diam(G)`.
- Veamos que pasa cuando tenemos G' conexa donde

$$V_{G'} = V_G + p_i \quad \text{y} \quad E_{G'} = E_G + \text{ al menos una arista que conecte a } p_i \text{ con } G.$$

  Existen dos casos posibles,

  1. `diam(G') = diam(G)`. Para este caso tomemos al primer vecino p_j de p_i tal que $d(p_j, v)$ sea mínima para todo $v \in V$ (en contraste con el resto de vecinos de p_i, que no necesariamente existen). Como (por el supuesto anterior) podemos inundar G en tiempo a lo más `diam(G) - 1` [Si suponemos que esto es cierto para `diam(G)` resulta que el diámetro aumenta, esto va en contra de la consideración de conservar el diámetro de G para G'].

     Así, supongamos sin pérdida de generalidad que, p_j recibe el mensaje `M` en tiempo `diam(G) - 1` (desde algún proceso distinguido como líder, según el algoritmo `flooding`). Como p_i y p_j son vecinos, distribuir `M` toma tiempo 1. Entonces la gráfica se inunda, para cada proceso, en tiempo a lo más `(diam(G) - 1) + (1) = diam(G)`.
  2. `diam(G') = diam(G) + 1`. Sabemos que p_i tiene algún vecino p_j (por conexidad de G'). Supongamos, sin pérdida de generalidad, que p_j recibe el mensaje `M` en tiempo `diam(G)`. Así, transmitir `M` de p_j a p_i toma tiempo 1, luego transmitir `M` desde un proceso distinguido como líder hacia p_i toma tiempo `diam(G) + 1`[III].

■

[I] Sabemos que el `diam(G) = 0`.
[II] Por simplicidad, llamaremos inundar a distribuir el mensaje `M` por cada proceso en G por medio del algoritmo `flooding`.
[III] El tiempo de transmitirlo hasta p_j más el tiempo de transmitirlo de p_j a p_i.

2. Considera el algoritmo de BroadcastTree visto en clase. ¿Cuál sería el peor caso en complejidad de tiempo para el algoritmo BroadcastTree? Explica detalladamente.

3. Considera el algoritmo de BroadConvergecast visto en clase.

1. Prueba el siguiente lema: «Todo proceso p_i a profundidad D, recibe el mensaje $< START >$ en tiempo D».

   Demostración por inducción sobre el sobre el camino formado por los *parents*

   Caso base: Sea un árbol distribuido con la raíz sin hijos, por lo tanto la profundidad es 0. Además el mensaje lo recibe en tiempo 0. Por lo tanto se cumple.

   Hipótesis inductiva: Sea un proceso u en un árbol distribuido a profundidad D, el mensaje $< START >$ llegara en tiempo D

   Paso inductivo: Sea v un proceso en el árbol distribuido tal que $u = v.parent$, es decir, u es padre de v entonces la $profundidad(v) = profundidad(u) + 1$ por hipótesis inducctiva $profundidad(v) = D + 1$, además el mensaje tomara $D + 1$ tiempo en llegar a v por que tarda D tiempo para llegar a u y v es hijo de u.

   □

2. Prueba el siguiente lema: «Todo proceso p_i a profundidad D envía su mensaje a la raíz en el tiempo $D + 2 * altura(p)$».

   Demostración por inducción sobre el sobre el camino formado por los *parents*

   Caso base: Sea un árbol distribuido con la raíz u sin hijos, por lo tanto $profundidad(u) = 0 = altura(u)$. El mensaje lo recibe en tiempo 0. Por lo tanto se cumple $D + 2 * altura(u) = 0 + 2 + (0) = 0$

   Hipótesis inductiva: Sea un proceso v en un árbol distribuido a profundidad D envía su mensaje a la raíz en el tiempo $D + 2 * altura(v)$

   Paso inductivo: Sea u un proceso en el árbol distribuido tal que $u = v.parent$, es decir, u es padre de v. Por hipótesis inductiva el proceso v, a profundidad D, envía su mensaje en $D + 2 * altura(v)$. Ya que u es padre de v su profundidad es $D - 1$ y su altura es $altura(v) - 1$ por tanto envia su mensaje en $(D - 1) + 2 * (altura(v) - 1)$

   □

4. ¿Se basan los algoritmos de BroadcastTree y ConvergeCast en el conocimiento acerca del número de nodos en el sistema? ¿Por qué?

Examinar el algoritmo BroadcastTree

El algoritmo requiere de información acerca de los nodos, necesita conocer el número de hijos de un nodo para enviar el mensaje a estos, en la linea 5 y 8 realiza esta instrucción. Además requiere conocer a su PADRE para recibir un mensaje de el en la linea 7.

Algorithm 1 BroadcastTree(ID,soyRaiz,M)

```
1: PADRE, HIJOS
2: Ejecutar inicialmente
3:
4: if soyRaiz then
5:     send(< M >) a todos en HIJOS
6: end if
7: Al recibir < M > de Padre
8: send(< M >) a todos en HIJOS
```

Examinar el algoritmo ConvergeCast

El algoritmo requiere de información acerca de los nodos, necesita conocer el número de hijos de un nodo para saber si el nodo es una hoja y debe enviar el mensaje a su padre. En la linea 6 pregunta por la cantidad de hijos del nodo, luego en la 11 requiere esta información de nuevo. Además requiere conocer a su PADRE pare enviarle el mensaje en las lineas 7 y 12.

Algorithm 2 ConvergeCast(ID,soyRaiz)

```
1: PADRE ← 0
2: HIJOS ← 0
3: noRecibidos ← 0
4: Ejecutar inicialmente
5:
6: if |HIJOS| == 0 then                                   ▷ es una hoja
7:     send(< ok >) a PADRE
8: end if
9: Al recibir < ok > de algún puerto en HIJOS
10: noRecibidos + +
11: if noRecibidos == |HIJOS| then
12:     send(< ok >) a PADRE
13: end if
```

Sin embargo, ninguno de estos algoritmos requiere conocer el total de nodos en la red ya que avanza por la red mediante relaciones de padre-hijo.

5. Generaliza el algoritmo de Broadconvergecast para:

1. Construya un árbol generador, es decir, inicialmente cada proceso tendrá sus variables $PADRE = \perp$ e $HIJOS = \emptyset$ y conforme el algoritmo vaya avanzando en el número de rondas, esas variables se vayan actualizando. El proceso raíz (distinguido) debe conocer el momento en que se terminó de construir este árbol generador.

2. Suponga que cada proceso tiene una entrada distinta para reportar algo (pueden ser información de sensores, lecturas, etc.) A partir del algoritmo anterior, indica las modificaciones que se tendrían que hacer en el algoritmo, para que se recolecte la información de estos procesos y la raíz tenga todas la entradas. Analiza la complejidad en bits, es decir, el total de bits que son enviados sobre los canales de comunicación (hint: Cada mensaje de información puede tomar k bits).

6. Da un algoritmo distribuido para contar el número de procesos en cada capa de un árbol enraizado T de forma separada. Al final la raíz reportará el número de procesos por capa. Analiza la complejidad de tiempo y la complejidad de mensajes de tu algoritmo.