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

25 de septiembre de 2022

1. Investiga y explica brevemente el concepto de time-to-live (TTL) usado en redes de computadoras, y úsalo para modificar el algoritmo de flooding visto en clase, de modo que un líder comunique un mensaje m a los procesos a distancia a lo más d del líder (m y d son entradas del algoritmo); todos los procesos a distancia mayor no deberán recibir m. Da un breve argumento que demuestre que tu algoritmo es correcto, y también haz un análisis de tiempo y número de mensajes.

2. Considera un sistema distribuido representado como una gráfica de tipo anillo, cuyos canales son bidireccionales, con n = mk procesos, con $m > 1$ y k es impar. Los procesos en las posiciones $0,\ k,\ 2k, ..., (m-1)k$ son marcados inicialmente como líderes, mientras que procesos en otras posiciones son seguidores. Todos los procesos tienen un sentido de dirección y pueden distinguir su vecino izquierdo de su vecino derecho, pero ellos no tienen información alguna acerca de sus ids.

El algoritmo 1 está destinado a permitir que los líderes recluten seguidores. No es difícil ver que todo seguidor eventualmente se agrega a sí mismo a un árbol enraízado con padre en algún líder. Nos gustaría que todos esos árboles tuvieran aproximadamente el mismo número de nodos.

- ¿Cuál es el tamaño mínimo y máximo posible de un árbol?
- Dibuja el resultado de una ejecución para el algoritmo con $k = 5$ y $m = 4$.

Pseudocódigo 1: Algoritmo reclutamiento

```
1  Algoritmo reclutamiento(id, soyLider):
2
3  Inicialmente hacer:
4    if soyLider then
5      parent = id
6      send(<recluta>) a ambos vecinos
7    else
8      parent = ⊥
9
10 Al recibir <recluta> desde p hacer:
11   if parent = ⊥ then
12     parent = p
13     send(<recluta>) a mi vecino que no es p
```

▷ Empecemos dando respuesta a la pregunta ¿Cuál es el tamaño mínimo y máximo posible de un árbol?. Para contestar esta pregunta análicemos varias cosas:

1. Sabemos que la cantidad de procesos es múltiplo de la cantidad de líderes, pues si

$$n, m, k \in \mathbb{N} \Rightarrow \frac{n}{k} = m \in \mathbb{N}.$$

   Entonces, la cantidad de líderes (k) es proporcional a la cantidad de seguidores (m), hasta el momento no sabemos si a cada líder le corresponda la misma cantidad de seguidores basados en el algoritmo de `reclutamiento`.

2. Recordemos que k es impar y por tanto tiene la forma $k = 2r + 1$ ($r \in \mathbb{N} \cup \{0\}$). Ahora, nos podemos preguntar ¿cada cuántos procesos hay un líder contando desde el último anterior o próximo? la respuesta es cada k procesos y la diferencia entre estos es de $k - 1$ procesos, pues los líderes están ubicados en múltiplos de k, luego

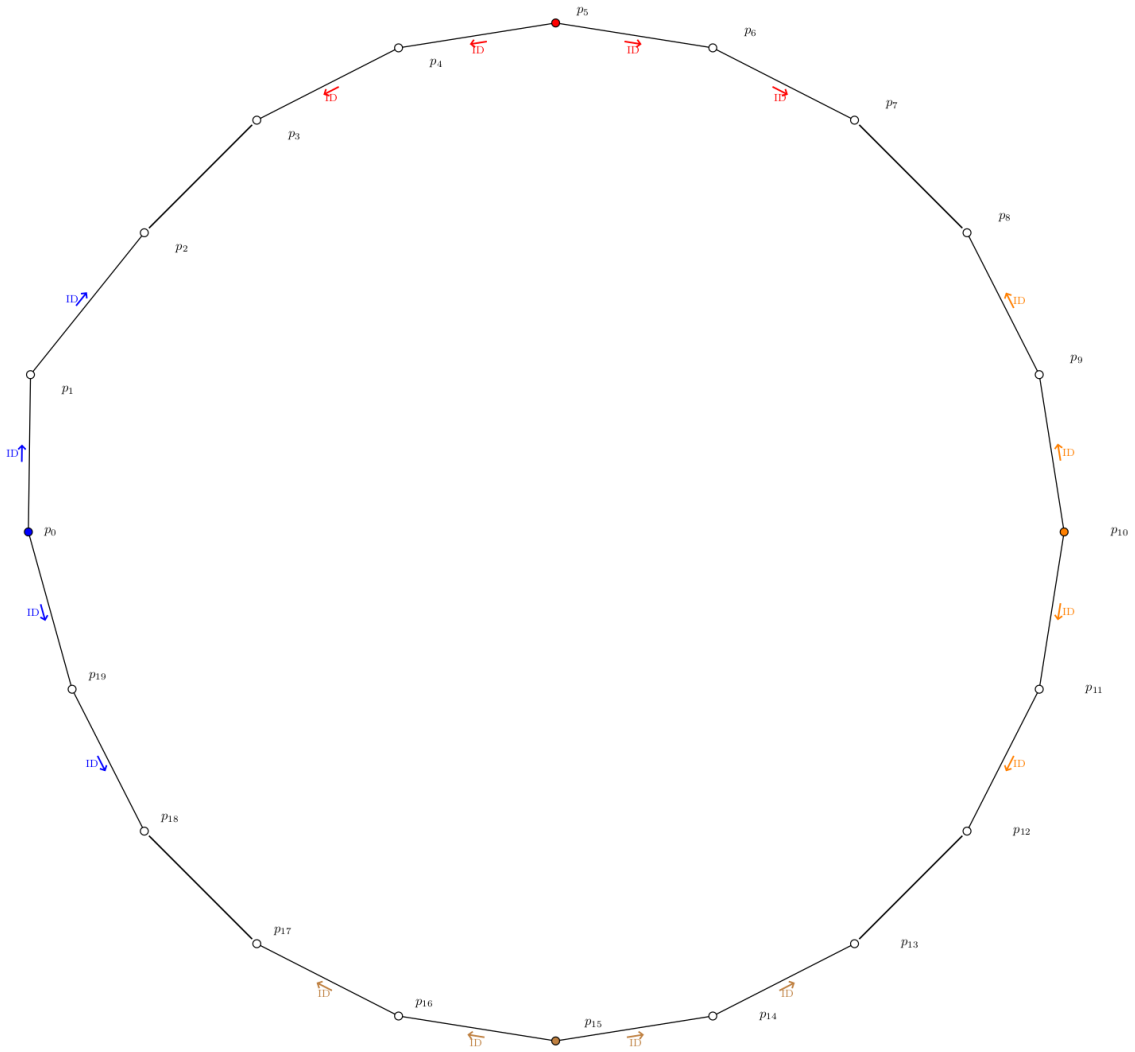
$$k - 1 = (2r + 1) - 1 = 2r.$$

   Inicialmente, los procesos líderes empiezan a reclutar (línea 6). Después de la primer ronda los procesos reclutados reclutan a sus vecinos que no son su padre (línea 13). Eventualmente todo los procesos seguidores son relcutados, como todo esto (en cada proceso reclutado lo que sigue es reclutar a su vecino) pasa al mismo tiempo (en cada ronda los procesos líderes tienen la misma cantidad de descendientes) podemos notar que la diferencia de procesos entre cualesquiera dos procesos líderes es dividida entre 2 para que cada líder termine por ser ancestro común de exactamente la mitad, adyacente al líder en cuestión, de esos procesos.

   Hasta este momento sabemos que un líder llegará a tener r descendientes por lado, en total cada líder tendrá $2r$ descendientes.

3. Como los árboles están enraizados por líderes y cada líder tiene $2r$ descendientes. Entonces, el total de nodos en los arboles generados será $2r + 1 = k$. Concluimos que el tamaño por árbol es de k sin importar como es m o k.

Por último, veamos como se ve la gráfica después de haber ejecutado el algoritmo de `reclutamiento`:



Los árboles en este caso se pueden formar desde cada nodo representado por algún color[I] y hasta donde se indica con las flechas del respectivo color. ◁

[I] estos son los nodos distinguidos como líderes.

3. ¿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.

4. Describe un algoritmo distribuido para construir un árbol generador sobre una gráfica arbitraria G, utilizando el algoritmo de elección de líder eligeLider para determinar la raíz del árbol y también el algoritmo BFS. Analiza la complejidad de tiempo y de mensajes.

Algorithm 1 arbolGenerador

1: profundidadArbol = profundidadT(soyRaiz), nivelT = 0, vecinos = 0
2: $PADRE, HIJOS, profundidad, soyRaiz = soyRaiz, string = ''$

5. El algoritmo puede mejorar su complejidad de tiempo si se ejecutan de forma paralela los dos algoritmos anteriores, es decir, si se ejecuta la elección de líder y la construcción del árbol BFS. Da un algoritmo distribuido que realice esto y muestra que es correcto. Adicionalmente, compara la complejidad de tiempo respecto al algoritmo anterior.

6. Prueba la siguiente afirmación:
El algoritmo BFS construye un árbol enraízado sobre un sistema distribuido con m aristas y diámetro D, con complejidad de mensajes $O(m)$ y complejidad de tiempo $O(D)$.