Universidad Nacional Autónoma de México

Facultad de Ciencias

Computación Distribuida

Tarea 2

Johann Ramón Gordillo Guzmán
418046090

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

13 de Octubre del 2020

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

Actividades

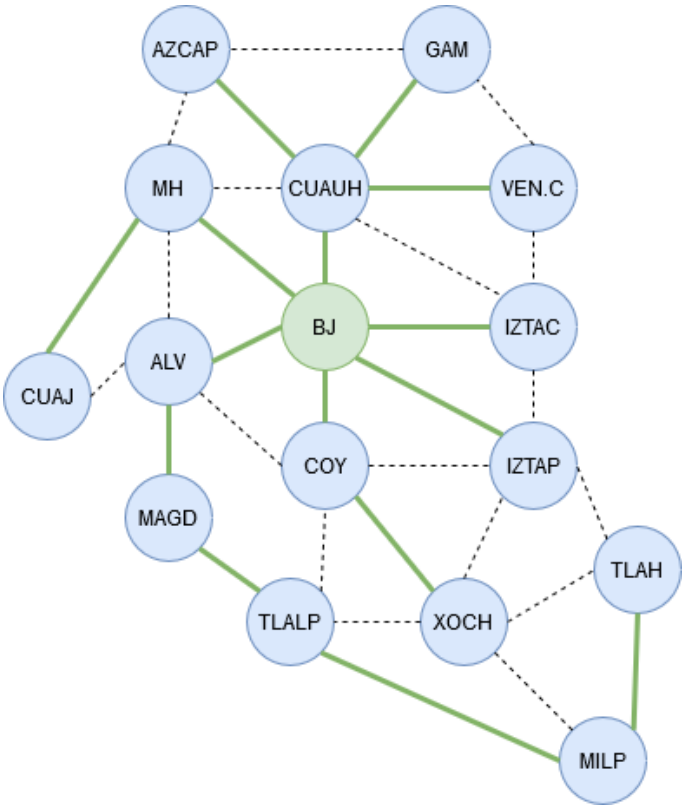
1. En la Ciudad de México varios puntos de control (nodos) fueron agregados. Cada uno de estos puntos tiene información de cuantas personas viven dentro de cierto díametro. Un punto (raíz) de la delegación Benito Juárez quiere recolectar la información de todos los puntos de control y determinar cuántas personas viven en la CDMX. Escribe un algoritmo distribuido para que cada nodo recolecte la información de sus hijos para envíarsela al padre sin repetirla y que el padre reporte el total de pobladores viviendo la CDMX.

   Respuesta.

   Para resolver este ejercicio me basaré en el Algoritmo de Broadcast y Convergecast sobre árboles generadores, visto en clase.

   Dado que la CDMX vista como una gráfica contiene ciclos, no sería eficiente usar únicamente Convergecast para obtener la población de cada delegación desde el nodo correspondiente a la delegación Benito Juárez. Entonces, lo que hay que hacer es construir el árbol generador de la gráfica correspondiente a la CDMX usando Broadcast y Convergecast y posteriormente hacer que cada delegación regrese a su padre su identificador y su número de habitantes, para que así esta información llegue a la raíz Benito Juárez y ésta pueda calcular el número total de habitantes de la CDMX como una suma de la población de cada delegación.

   En los mensajes GO() para hacer Broadcast no enviaremos información adicional, pero en los mensajes BACK() sí devolveremos el identificador de la delegación y el número de habitantes, excepto cuando el nodo ya tenga padre, en este caso devolveremos al conjunto vacío dentro del mensaje BACK().



Algoritmo 1 Algoritmo distribuido para encontrar la población de la CDMX

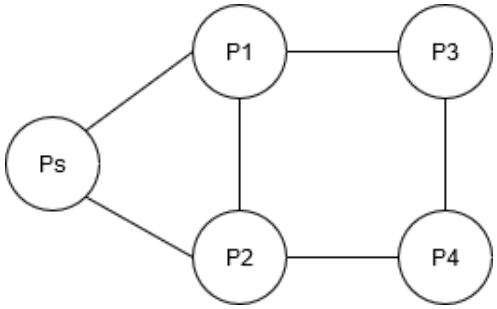
```
1: Initially do
2: begin:
3:     if p_bj = p_del then                          ▷ Si p_del es la delegación Benito Juárez
4:         parent_del ← del
5:         expected_msg_del ← |neighbors_del|
6:         for j ∈ neighbors_del do
7:             send GO() to p_j
8:         end for
9:     else
10:        parent_del ← ∅                            ▷ Si no, solo inicializa sus variables
11:    end if
12:    children_del ← ∅
13:    poblacion_del                                 ▷ Población de la delegación
14: end

15: when GO() is received from p_j do
16: begin:
17:    if parent_del = ∅ then
18:        parent_del ← j
19:        expected_msg_del ← |neighbors_del| − 1
20:        if expected_msg_del = 0 then
21:            send BACK((del, poblacion_del)) to p_j    ▷ Envíamos el id y su número de habitantes
22:        else
23:            for k ∈ neighbors_del\{j} do
24:                send GO() to p_k
25:            end for
26:        end if
27:    else
28:        send BACK(∅) to p_j
29:    end if
30: end

31: when BACK(val_set) is received from p_j do
32: begin:
33:    expected_msg_del ← expected_msg_del − 1
34:    if val_set ≠ ∅ then
35:        children_del ← children_del ∪ {j}
36:    end if
37:    if expected_msg_del = 0 then
38:        val_set_del ← ⋃_{x∈children_del} val_set_x ∪ {(del, poblacion_del)}
39:        if parent_del ≠ del then
40:            send BACK((del, poblacion_del)) to parent_del    ▷ Terminación local para p_del
41:        else
42:            total_habitantes ← 0
43:            for (del, poblacion) ∈ val_set_del do
44:                total_habitantes ← total_habitantes + poblacion
45:            end for
46:            Return total_habitantes                  ▷ Regresamos la población de la CDMX
47:        end if
48:    end if
49: end
```

2. Explica por qué el algoritmo de *convergecast* con *broadcast* puede producir más de un árbol.

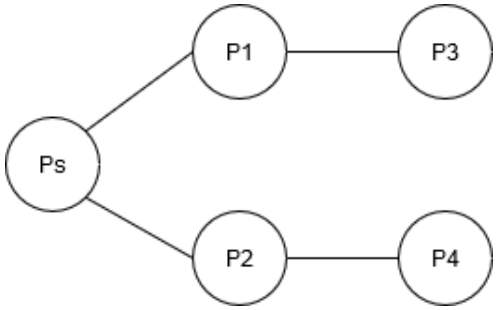
Respuesta.

El árbol generador construido con el algoritmo no es único, ya que en algunas gráficas podemos encontrar nodos de grado mayor o igual a dos a los que llegan mensajes al mismo tiempo, mismos que se irán guardando en el buffer, y dependiendo de cuál consideremos que llega primero, generaremos un árbol u otro. Es decir, la velocidad con la que son envíados los mensajes GO() a través de un canal determinará la estructura del árbol, y es por esto que la ejecución del algoritmo sobre la misma gráfica con el mismo proceso distinguido puede dar lugar a árboles distintos.

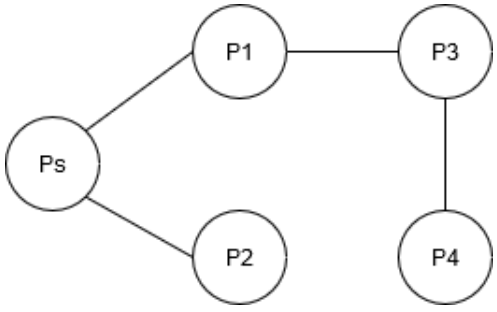
Veamos un ejemplo sobre la siguiente red:



El algoritmo produce el siguiente árbol generador:



Pero también el siguiente:



Dependiendo de si P4 recibe primero el mensaje de P3 o de P2. Es más sencillo verlo si agregamos un *peso* a las aristas, donde el mensaje llegará primero a través del camino cuya suma de pesos sea menor.

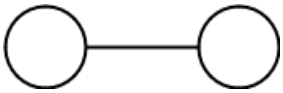
3. Considera el algoritmo de convergecast sobre el árbol generador ya construido. Demuestra que cualquier nodo a altura h (distancia más corta desde la hoja hacia el nodo) manda un mensaje en la ronda h a más tardar.

   Demostración.

   Procedemos por inducción sobre la altura h de un nodo.

   - **Caso Base.**

     Sea $v \in G$ tal que $h(v) = 1$. Es claro que el nodo manda un mensaje en la ronda 1, por lo que el caso base se cumple. Visualmente, la gráfica podría verse como sigue:

   - **Hipótesis de Inducción.**

     Supongamos que si $h(v) = k$, para un nodo v, el nodo manda un mensaje en la ronda k a más tardar.

   - **Paso Inductivo.**

     Sea $v \in G$ tal que $h(v) = k + 1$. Como por definición las hojas de un árbol tienen grado 1, su único vecino es su padre. Sea v_i la hoja que define la altura de v y sea v_j el padre de la misma.

     Como $h(v) = k + 1$, la distancia de v a v_j es k. Más aún, en el subárbol $G \backslash \{v_i\}$ se cumple que $h(v) = k$; por lo que, por hipótesis, v manda un mensaje en la ronda k a más tardar.

     Además, al ser v_j el padre de la hoja v_i en G es claro que $h(v_j) = 1$, por lo que mandará un mensaje en la ronda 1.

     De lo anterior se sigue que v manda un mensaje en la ronda $h(v) = k + 1$ a más tardar.

     Otra manera de verlo es por tiempos. El tiempo en que llega un mensaje de v a v_j es, por hipótesis, menor a k y el tiempo en que llega un mensaje de v_j a v_i es 1, por lo que claramente el tiempo en el que llegue un mensaje de v a v_i será menor o igual a $k + 1$.

   $\therefore$ Cualquier nodo a altura h manda un mensaje en la ronda h a más tardar. ■

Referencias

[1] Raynal, M. *Distributed Algorithms for Message-Passing Systems.* Springer, 2013.

[2] Aspnes, J. *Notes on Theory of Distributed Systems.* Yale University, 2017.