Universidad Nacional Autónoma de México

Facultad de Ciencias

Computación Distribuida

Tarea 3

Johann Ramón Gordillo Guzmán
418046090

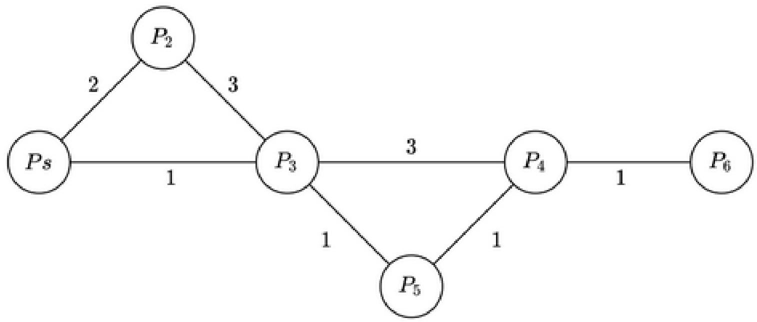
Tarea presentada como parte del curso de **Computación Distribuida** impartido por la profesora **M.C Karla Rocío Vargas Godoy**.

20 de Octubre del 2020

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

Actividades

1. Ejecuta el algoritmo BFS de la figura 1.11 [libro de M. Raynal] en la siguiente gráfica:



Respuesta.

En la **ronda 0**, P_s recibe el mensaje START() y se envía GO(-1) a si mismo. Hacemos:

$$\begin{aligned} parent_s &= P_s \\ children_s &= \varnothing \\ level_s &= -1 + 1 = 0 \\ expected\_msg_s &= 2 \end{aligned}$$

Posteriormente, P_s envía GO(0) a sus vecinos, P_2 y P_3.

En la **ronda 1**, P_3 recibe el mensaje GO(0) de P_s y el mensaje enviado a P_2 va a medio canal. P_3 actualiza sus valores:

$$\begin{aligned} parent_3 &= P_s \\ children_3 &= \varnothing \\ level_3 &= 0 + 1 = 1 \\ expected\_msg_3 &= 3 \end{aligned}$$

Posteriormente, P_3 envía GO(1) a P_2, P_4 y P_5.

En la **ronda 2**, P_2 recibe el mensaje GO(0) de P_s y p_5 recibe el mensaje de P_3. P_2 y P_5 actualizan sus valores:

$$\begin{aligned} parent_2 &= P_s \\ children_2 &= \varnothing \\ level_2 &= 0 + 1 = 1 \\ expected\_msg_2 &= 1 \end{aligned}$$

$$\begin{aligned} parent_5 &= P_3 \\ children_5 &= \varnothing \\ level_5 &= 1 + 1 = 2 \\ expected\_msg_5 &= 1 \end{aligned}$$

Posteriormente, P_5 envía GO(2) a P_4, y P_2 envía $GO(1)$ a P_3.

En la **ronda 3**, P_4 recibe el mensaje GO(2) de P_5. P_4 actualiza sus valores:

$$\begin{aligned} parent_4 &= P_5 \\ children_4 &= \varnothing \\ level_4 &= 2+1=3 \\ expected\_msg_4 &= 2 \end{aligned}$$

Posteriormente, P_4 envía GO(3) a P_6 y a P_3.

En la **ronda 4**, P_6 recibe el mensaje GO(3) de P_4 y P_6 actualiza sus valores:

$$\begin{aligned} parent_6 &= P_4 \\ children_6 &= \varnothing \\ level_6 &= 3+1=4 \\ expected\_msg_6 &= 0 \end{aligned}$$

Como $expected\_msg_6 = 0$, P_6 envía un mensaje BACK(YES, 4) a P_4.
Por otro lado, P_2 recibe el mensaje GO(1) de P_3 y como $parent_2$ no es vacío y no se cumple $level_2 > d+1 = 2$, entonces P_2 no actualiza sus valores y envía BACK(NO, 2) a P_3.
Mientras tanto, P_4 recibe el mensaje GO(1) de P_3 y como el padre de P_4 no es vacío, comprobamos que $level_4 = 3 > d+1 = 2$, como esto se cumple, tendremos que actualizar los valores de P_4:

$$\begin{aligned} parent_4 &= P_3 \\ children_4 &= \varnothing \\ level_4 &= 1+1=2 \\ expected\_msg_4 &= 2 \end{aligned}$$

Posteriormente, P_4 envía GO(2) a P_6 y a P_5.

En la **ronda 5**, P_3 recibe GO(1) de P_2 y como no se cumple $level_3 > 2$, entonces P_3 envía BACK(NO, 2) a P_2.
Al mismo tiempo P_5 recibe GO(2) de P_4, y como no se cumple que $level_5 > 3$, entonces P_5 envía BACK(NO, 3) a P_4.
También P_4 recibe BACK(YES, 4) de P_6, pero no hacemos nada ya que no se cumple que $level_4 = 4$.
Finalmente, P_6 recibe GO(2) de P_4, por lo que actualiza sus valores:

$$\begin{aligned} parent_6 &= P_4 \\ children_6 &= \varnothing \\ level_6 &= 2+1=3 \\ expected\_msg_6 &= 0 \end{aligned}$$

Posteriormente, P_6 envía un mensaje BACK(YES, 3) de vuelta a P_4.

En la **ronda 6**, P_4 recibe el mensaje BACK(YES, 3) de P_6 y como $level_4 + 1 = 3$, actualizamos los valores para P_4:

$$\begin{aligned} parent_4 &= P_3 \\ children_4 &= \{P_6\} \\ level_4 &= 2 \\ expected\_msg_4 &= 1 \end{aligned}$$

Al mismo tiempo, P_4 recibe el mensaje BACK(NO, 3) de P_5, y como $level_4 + 1 = 3$, actualizamos valores:

$$\begin{aligned} parent_4 &= P_3 \\ children_4 &= \{P_6\} \\ level_4 &= 2 \\ expected\_msg_4 &= 0 \end{aligned}$$

Ahora $expected\_msg_4 = 0$, por lo que enviamos el mensaje BACK(YES, 2) a su padre P_3.
Mientras tanto, P_3 recibe el mensaje GO(3) de P_4, pero no se cumple que $level_3 > d + 1$, por lo que P_3 envía un mensaje BACK(NO, 4) a P_4.

En la **ronda 7**, P_3 recibe el mensaje BACK(NO, 2) de P_2, y como $level_3 = 1$ y $d = 2$, se cumple que $level_3 + 1 = d$, por lo que actualizamos valores:

$$\begin{aligned} parent_3 &= P_s \\ children_3 &= \varnothing \\ level_3 &= 1 \\ expected\_msg_3 &= 2 \end{aligned}$$

Ahora estamos listos para pasar a la **ronda 8**.

En la **ronda 8**, P_2 recibe el mensaje BACK(NO, 2) de P_3, y como $level_2 = 1$ y $d = 2$, se cumple que $level_2 + 1 = d$, por lo que actualizamos valores:

$$\begin{aligned} parent_2 &= P_s \\ children_2 &= \varnothing \\ level_2 &= 1 \\ expected\_msg_2 &= 0 \end{aligned}$$

Como $expected\_msg_2 = 0$, P_2 envía un mensaje BACK(YES, 1) a su padre P_s.

En la **ronda 9**, P_4 recibe el mensaje BACK(NO, 4) de P_3, pero no hacemos nada ya que $d \neq level_4 + 1$.
Por otra parte, P_3 recibe el mensaje BACK(YES, 2) de P_4, y como $d = level_3 + 1$, actualizamos valores:

$$\begin{aligned} parent_3 &= P_s \\ children_3 &= \{P_4\} \\ level_3 &= 1 \\ expected\_msg_3 &= 1 \end{aligned}$$

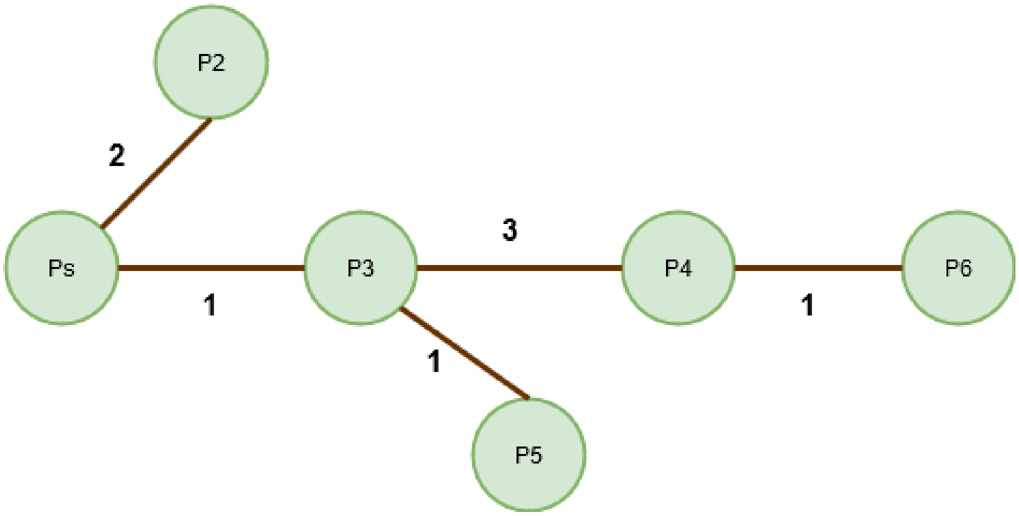
Ahora estamos listos para pasar a la siguiente ronda.

En la **ronda 10**, P_s recibe el mensaje BACK(YES, 1) de P_2, y como $d = 1$ y $level_s = 0$, se cumple que $level_s + 1 = d$, por lo que actualizamos valores:

$$\begin{aligned} parent_s &= P_s \\ children_s &= \{P_2\} \\ level_s &= 0 \\ expected\_msg_s &= 1 \end{aligned}$$

Sin embargo, notemos que P_3 nunca recibió el mensaje BACK() de P_5, por lo que $expected\_msg_3$ nunca llegará a ser 0 y no se podrá finalizar la construcción del árbol BFS. Se supone que P_5 recibiría BACK(NO, d) de P_4, lo que haría que $expected\_msg_5$ sea 0 y hará que P_5 envíe BACK(YES, d) a su padre P_3. Esto se debe a un pequeño error en el Algoritmo del libro de M. Raynal.

Suponiendo que el algoritmo finaliza correctamente, llegaremos a que el árbol resultante es:



2. ¿Se puede obtener más de un árbol BFS en la gráfica anterior?

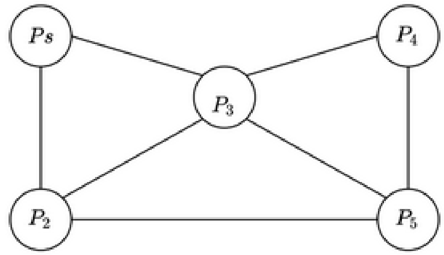
Respuesta.

No. Esto se debe a la distribución de los tiempos y las distancias en el sistema distribuido.

Es claro que P_2 y P_3 siempre serán hijos de P_s, y que P_6 siempre será hijo de P_4, pero notemos que gracias a la distribución de los tiempos P_5 no podrá ser hijo de P_4, y como la longitud de la trayectoria $P_S P_3 P_5 P_4$ siempre será mayor que la de la trayectoria $P_S P_3 P_4$, se cumple que P_4 siempre será hijo de P_3.

Al culminar la ejecución del algoritmo BFS sobre la gráfica anterior, siempre se eliminarán del árbol resultante las aristas $P_2 P_3$ y $P_5 P_4$.

3. Ejecuta el algoritmo DFS de la figura 1.17 [libro de M. Raynal] en la siguiente gráfica:



Respuesta.

En la **ronda 0**, P_s recibe el mensaje START() y actualiza sus valores:

$$parent_s = P_s$$
$$children_s = \{P_2\}$$

Posteriormente, envía un mensaje GO($\{P_s\}$) a uno de sus vecinos. Sin pérdida de generalidad, supongamos que a P_2.

En la **ronda 1**, P_2 recibe el mensaje GO($\{P_s\}$) de P_s.
Vemos que $neighbors_2 = \{P_s, P_3, P_5\} \not\subset visited = \{P_s\}$, por lo que P_2 envía el mensaje GO($\{P_s, P_2\}$) a uno de sus vecinos no visitados. Sin pérdida de generalidad, supongamos que envía el mensaje a P_3. Finalmente, actualizamos los valores de P_2:

$$parent_2 = P_s$$
$$children_2 = \{P_3\}$$

En la **ronda 2**, P_3 recibe el mensaje Go($\{P_s, P_2\}$) de P_2.
Como $neighbors_3 = \{P_s, P_2, P_4, P_5\} \not\subset visited = \{P_s, P_2\}$, entonces P_3 manda un mensaje GO($\{P_s, P_2, P_3\}$) a uno de sus vecinos, supongamos sin pérdida de generalidad que manda el mensaje a P_4. Posteriormente, actualizamos sus valores:

$$parent_3 = P_2$$
$$children_3 = \{P_4\}$$

En la **ronda 3**, P_4 recibe el mensaje Go($\{P_s, P_2, P_3\}$) de P_3.
Como $neighbors_4 = \{P_3, P_5\} \not\subset visited = \{P_s, P_2, P_3\}$, entonces P_4 manda un mensaje GO($\{P_s, P_2, P_3, P_4\}$) a su único vecino que no ha sido visitado: P_5. Posteriormente, actualizamos sus valores:

$$parent_4 = P_3$$
$$children_4 = \{P_5\}$$

En la **ronda 4**, P_5 recibe el mensaje GO($\{P_s, P_2, P_3, P_4\}$) de P_4.
Como $neighbors_5 = \{P_2, P_3, P_4\} \subset visited = \{P_s, P_2, P_3, P_4\}$, entonces P_5 envía un mensaje

BACK($\{P_s, P_2, P_3, P_4, P_5\}$) a P_4. Posteriormente, actualizamos sus valores:

$$parent_5 = P_4$$
$$children_5 = \varnothing$$

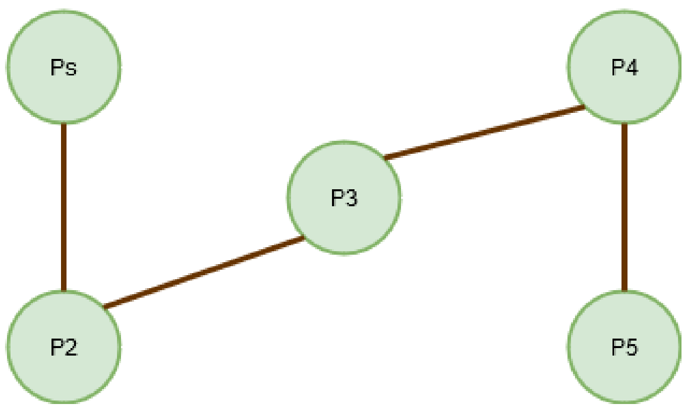
En la **ronda 5**, P_4 recibe el mensaje BACK($\{P_s, P_2, P_3, P_4, P_5\}$) de P_5.
Como $neighbors_4 = \{P_3, P_5\} \subset visited = \{P_s, P_2, P_3, P_4, P_5\}$, verificamos si $parent_4 = P_4$.
Como $parent_4 \neq P_4$, P_4 envía un mensaje BACK($\{P_s, P_2, P_3, P_4, P_5\}$) a su padre P_3.

En la **ronda 6**, P_3 recibe el mensaje BACK($\{P_s, P_2, P_3, P_4, P_5\}$) de P_4.
Como $neighbors_3 = \{P_s, P_2, P_4, P_5\} \subset visited = \{P_s, P_2, P_3, P_4, P_5\}$, verificamos si $parent_3 = P_3$.
Como $parent_3 \neq P_3$, P_3 envía un mensaje BACK($\{P_s, P_2, P_3, P_4, P_5\}$) a su padre P_2.

En la **ronda 7**, P_2 recibe el mensaje BACK($\{P_s, P_2, P_3, P_4, P_5\}$) de P_3.
Como $neighbors_2 = \{P_s, P_3, P_5\} \subset visited = \{P_s, P_2, P_3, P_4, P_5\}$, verificamos si $parent_2 = P_2$.
Como $parent_2 \neq P_2$, P_2 envía un mensaje BACK($\{P_s, P_2, P_3, P_4, P_5\}$) a su padre P_s.

En la **ronda 8**, P_s recibe el mensaje BACK($\{P_s, P_2, P_3, P_4, P_5\}$) de P_2.
Como $neighbors_s = \{P_2, P_3\} \subset visited = \{P_s, P_2, P_3, P_4, P_5\}$, verificamos si $parent_s = P_s$.
Como $parent_s \neq P_s$, terminamos la ejecución del algoritmo.

Finalmente, el árbol resultante es:



4. Considera el algoritmo BFS que no detecta terminación en un sistema síncrono. Sea D la distancia más grande de la raíz a cualquier otro proceso.
Demuestra que para cada $1 \leq t \leq D$, después de t rondas, cada vértice p_i a distancia t ya ha recibido un mensaje con $d = t - 1$ de algún vecino p_j y por lo tanto $distance_i = t$ y $parent_i = j$ tal que $distance_j = t - 1$ (**Hint:** en la ronda 0 la raíz comienza su ejecución y envía su mensaje a sus vecinos y estos lo reciben en la ronda 1).

```
initially do
    if pid = initiator then
        distance ← 0
        send distance to all neighbors
    else
        distance ← ∞
upon receiving d from p do
    if d + 1 < distance then
        distance ← d + 1
        parent ← p
        send distance to all neighbors
```

Demostración.

Procedemos por inducción sobre t.

- **Caso Base.**

  Probemos que se cumple para $t = 1$.

  En la **ronda 0**, la raíz comienza su ejecución, asignándose una distancia $d_{init} = 0$ y envía esta distancia a sus vecinos, que están a distancia 1. Los demás nodos inicializan su distancia a ∞.

  En la **ronda 1**, los nodos a distancia 1 de la raíz reciben el mensaje de la raíz conteniéndo su distancia $d_{init} = 0$, y como $0 + 1 = 1 < \infty$, éstos nodos actualizan su distancia a $d_i = 1 = t$. Además, su padre es $parent_i = init$ y el mensaje que reciben es $d_{init} = 0 = 1 - 1 = t - 1$

  $\therefore$ La afirmación se cumple para $t = 1$.

- **Hipótesis de Inducción.**

  Supongamos que la afirmación se cumple para $t \leq D - 1$.

- **Paso Inductivo.**

  Probemos que la afirmación se cumple para $t = D$.

Sea N un nodo a distancia D de la raíz y sea P un vecino de N a distancia $D-1$ de la raíz.

Luego, por Hipótesis de Inducción, después de $D-1$ rondas, el nodo P ya recibió un mensaje con $d = D-2$ de algún vecino p_j y por lo tanto $distance_P = D-1$ y $parent_P = j$ tal que $distance_j = D-2$.

De esta manera, como N es vecino de P, está a distancia 1, por lo que en la siguiente ronda, la **ronda D**, el nodo N recibirá un mensaje con $d = D-1$ de P y por lo tanto actualizará su distancia a $distance_N = D$ y $parent_N = P$ tal que $distance_P = D-1$.

$\therefore$ Se cumple la afirmación para $t = D$.

$\therefore$ Se cumple la afirmación. $\blacksquare$

Referencias

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

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