Computación Distribuida
Tarea 8

González Montiel Luis Fernando

22 de octubre de 2019

1. Recuerda la reducción dada en clase para transformar cualquier detector de fallas que cumpla integridad débil en uno que cumple integridad fuerte.
   Algorítmo1

```
output p = null
while true do
p queries its local failure detector module D p
suspects p = D p
send(p, suspects p ) to all
end while
upon receiving (q, suspects q ) from neighbor q
begin:
output p = (output p u suspects q ) - {q}
end
```

   Demostrar que el algoritmo de transformación es correcto, es decir, demostrar que el detector de fallas simulado satisface integridad fuerte y que mantiene las propiedades de precisión (perpetua o eventual).

   Solución:
   Si algún proceso p falla, hay un correcto proceso q que eventualmente sospecha de p (integridad débil). Cuando q envía su lista de sospechosos a cada proceso, p se marca como sospechoso y cada proceso agrega p a su lista de sospechosos. Como p falló, ahí será el momento en el que cada proceso deja de recibir mensajes de p, por lo que no eliminan p de su lista de sospechosos nuevamente.

   Para probar que se preserva la precisión perpetua, ya que ningún proceso sospecha p antes del tiempo, ningún proceso envía un mensaje con p en la lista de sospechosos antes de que p se crashe (si lo hace). Para una precisión eventual, si p es correcto y todos los procesos correctos no sospechan p, y hay algún proceso que sospecha p, entonces eventualmente falla. En otro caso, si p es correcto, eventualmente cada proceso correcto q recibe nuevamente un mensaje de p, luego q elimina p de la salida q. Dado que no hay procesos correctos sospecha p, ningún proceso envía p a la lista de sospechosos. Entonces, q no agrega p a la salida q nuevamente.

2. Recuerda la implementación de $\diamond$P vista en clase (Algoritmo 1).
   Suponemos que el sistema es parcialmente síncrono, es decir, eventualmente se estabilizan las velocidades relativas de los procesos y las velocidades de transmisión de los canales; y que todos los procesos pueden comunicarse entre ellos. Demuestra que la implementación vista cumple con integridad fuerte.

   Solución:
   Bajo la premisa que siempre hay un proceso correcto que manda la información correcta y aunque fallen n procesos, siempre habrá uno que siga mandando la información correcta por los Lemas ya vistos. Entonces poco a poco se irá creando un conjunto de procesos correctos que tengan información correcta.

3. Si tuviéramos un sistema con topología arbitraria, ¿el algoritmo del inciso pasado funcionaría? Argumenta tu respuesta.

```
    Constants
neighbors
T
n
Variables
clock()
lastHB = [0, . . . . , 0]
process
timeout = [T, . . . . , T ]
suspect = [false, . . . . , false]

every T units of time
for each q \in neighbors do
   send(hHB, pi)
end for

upon the receiving of hHB, qi from a neighbor q
begin:
   if (suspect[q] = true) then
      timeout[q] = timeout[q] + 1
      suspect[q] = f alse
   end if
   lastHB[q] = clock()
end

when timeout[q] == clock() - lastHB[q] . when the timer for a message of
q \in \prod expires, p starts suspecting q
begin:
   suspect[q] = true
end
```

Solución:
Suponiendo que se habla de cualquier gráfica, es una disconexa no se podría porque no llegan los mensajes.