Sistemas Distribuídos

Práctica 2: Problema de Santa Claus

Descripción del problema.

El problema de Santa Claus es un problema de sincronización similar a los que hemos estudiado en clase. A diferencia de otros problemas que pueden continuar su ejecución indefinidamente, el problema de Santa Claus tiene un final claramente descrito: la entrega de los regalos. Para llegar a ese punto deberán regresar todos los renos que estaban ausentes (9 en nuestro caso). Mientras tanto, los elfos (o duendes) irán construyendo los regalos que habrán de ser entregados, y Santa estará durmiendo o cuando sea necesario, ayudando a los elfos.

Necesariamente debemos utilizar hilos concurrentes y mecanismos de sincronización entre gorrutinas. Podemos diferenciar 3 funciones distintas para gorrutinas:

- **Santa**: duerme hasta que es despertado por un elfo, en cuyo caso, ayuda con los regalos, o por el último reno, que será el momento de repartir los regalos y finalizar la ejecución del programa.

- **Elfo**: construye su regalo (la gorrutina duerme durante un número variable de segundos). Estadísticamente uno de cada tres elfos necesitará ayuda para poder terminar su regalo. Además, únicamente despiertan a Santa si se juntan tres elfos con problemas, y entonces santa ayudará a los tres elfos en conjunto y se volverá a dormir.

- **Reno**: llega tras un tiempo aleatorio de unos segundos. Para garantizar la restricción del enunciado del mínimo de segundos entre las llegadas de dos renos consecutivos he optado por lanzar las gorrutinas reno tras intervalos de 5 segundos.

Para lanzar las gorrutinas reno y elfo, por lo tanto, utilizo otras dos funciones que se lanzan concurrentemente desde main.

~

Estrategias y mecanismos de sincronización.

Requisitos de tiempos: Para modelar los tiempos que deben emplear las gorrutinas en las diferentes acciones he definido una serie de variables globales que utilizo en los sleep.

Sincronización: En Go podemos utilizar *buffered channels* como semáforos o *locks* para sincronizar las gorrutinas y gestionar el acceso a variables compartidas. Yo he optado por utilizar varios mutex y variables compartidas.

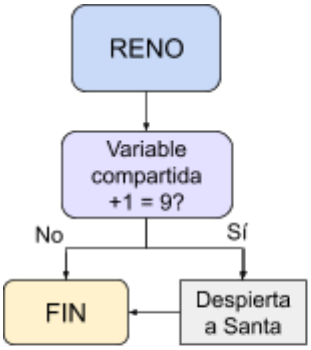
Santa se duerme bloqueándose en un lock (intentando coger un lock que no está disponible). Para despertarlo, habrá que hacer unlock (liberarlo) desde otra gorrutina.
También utilizo dos mutex para controlar el acceso a las variables compartidas que cuentan el número de renos que ya han llegado y el número de elfos que están en problemas en un momento dado.
Por último, utilizo un mutex para bloquear al elfo que llama a Santa mientras Santa ayuda (espera en un *sleep*) a los tres elfos, ya que no debe permitirse el acceso a otros elfos a la variable compartida del número de elfos en problemas hasta que este último elfo la haya reseteado a 0 debidamente.

Prioridad de Santa: En el enunciado se describe la restricción de que si llegan todos los renos y Santa está ocupado ayudando a un grupo de elfos, terminará de ayudarlos e inmediatamente saldrá a repartir los regalos con los renos, independientemente de si hay más grupos de 3 elfos que necesiten ayuda. Esto sencillamente se cumple si al despertarse Santa comprueba primero si están todos los renos listos antes de ayudar a los elfos.

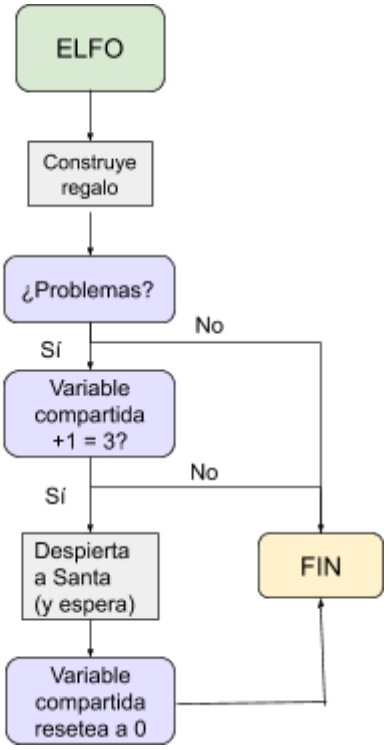
Esperar a gorrutinas: El hilo de ejecución principal (main) debe esperar a que todas las gorrutinas terminen su ejecución. Para ello he utilizado una variable global *wait group* del paquete sync de Go. Nada más empezar la ejecución, se indica el número de gorrutinas por las que se debe esperar (números de elfos + números de renos + Santa), las cuales notificarán su finalización llamando al método Done() de la variable. Una vez todas hayan finalizado, el main podrá terminar también su ejecución.

Seguimiento: Para poder hacer un seguimiento de lo que está ocurriendo en cada momento en tiempo de ejecución es indispensable imprimir mensajes por pantalla que describan la situación de cada gorrutina.

Originar problemas: Para decidir si un elfo necesita ayuda o no recurro a la función Intn() de math/rand. Recibe como parámetro un número natural y devuelve un número entero en el intervalo [0, n). En nuestro caso los posible valores devueltos son 0, 1 y 2, y decido que cuando sea 0 el elfo necesitará ayuda.


RENO
Variable compartida +1 = 9?
No
Sí
FIN
Despierta a Santa


SANTA
Duerme
Reno
Elfo
FIN
Ayuda a los elfos


ELFO
Construye regalo
¿Problemas?
No
Sí
Variable compartida +1 = 3?
No
Sí
Despierta a Santa (y espera)
FIN
Variable compartida resetea a 0