SSDD

- práctica 4 -

Por:

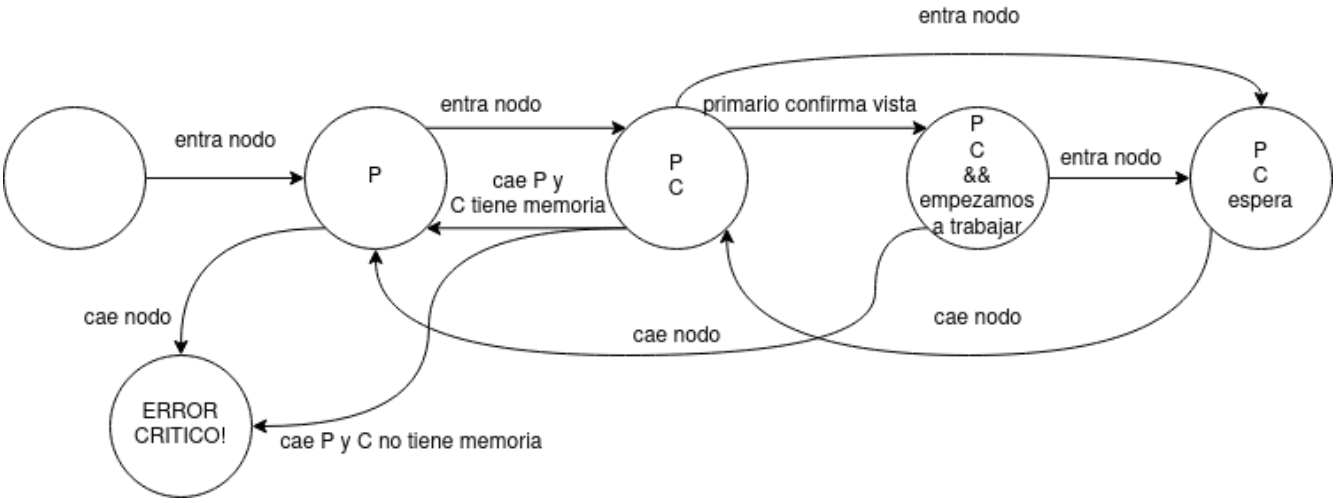
Diego Caballé Casanova (738712)

1. Enunciado

En esta práctica vamos a implementar un servicio de GV, encargado de tolerar fallos, mantener la consistencia de los datos y establecer la comunicación entre los clientes y las máquinas con los datos. Además, desarrollaremos una serie de pruebas para comprobar el correcto funcionamiento de nuestra implementación.

2. Implementación

En esta práctica nos centraremos principalmente en el comportamiento del GV. Este comportamiento se puede describir con el siguiente diagrama de estados:



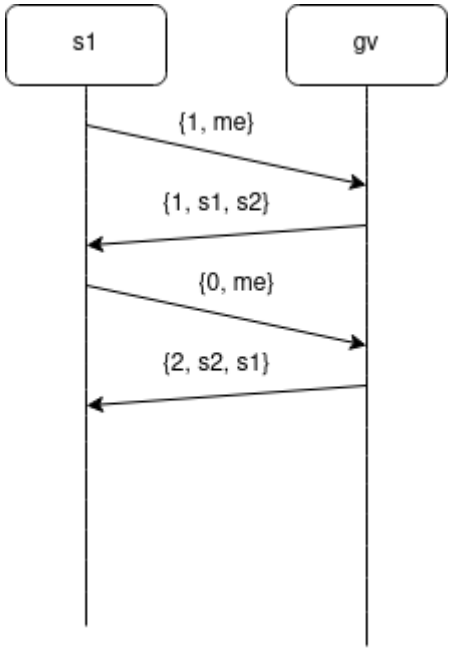
Este diagrama de estados se encuentra implementado en servidor_vistas.go, para conseguir implementarlo, hemos creado el tipo de dato ServVistas con los siguientes campos:

```
type ServVistas struct {
        msgsys.MsgSys
        doneTicker chan struct{}
        actual gvcomun.Vista
        tentativa  gvcomun.Vista
        espera [] msgsys.HostPuerto
        hanLatido map[msgsys.HostPuerto] int //-1 ha latido, 0 no ha latido, > 0 no ha latido durante x intentos (hasta gvcomun.LATIDOSFALLIDOS)
        hemosEmpezadoATrabajar bool
}
```

·Actual es la vista una vez está confirmada. Es la vista a la que tienen acceso los clientes del servicio de almacenamiento.
·Tentativa es la vista que se va alterando mientras hay caidas o que aún no ha sido confirmada por el actual primario. Es la vista que a la que tienen acceso los clientes del servicio de gestión de vista, de esta manera saben que rol tiene que nodo y pueden asumir mejor su rol o conocer un cambio de rol.
·Espera, es una lista en donde el GV apunta los nodos que tiene en espera que usará para sustituir a un nodo copia.
·HanLatido, una lista por nodos en la que se apunta cuales han latido, y en caso negativo, cuantos intervalos de latidos han fallado cada uno de los nodos.
·HemosEmpezadoATrabajar, es un booleano que nos dice si el nodo primario y copia ya han empezado a almacenar memoria, de esta manera sabremos si ante varias caídas estamos o no ante un error crítico.

Para transicionar entre un estado y otro, y actualizar el contenido de estas variables para cambiar de estado existen 2 tipos de mensaje importante, los latidos y los ticks.

Los ticks son eventos internos que ocurren cada intervalo de latido, con este tick podemos monitorizar los distintos nodos que tenemos apuntados y saber si están vivos y muertos, y en el caso de estar muertos, actualizar conforme el diagrama de estados.
Por otro lado, los latidos son mensajes que nos llegan de los clientes del servicio de gestión de vistas, con este mensaje recibiremos la información de: un nodo se ha conectado al sistema, un nodo ha perdido la memoria y el primario nos ha validado la vista.
El caso de perdida de memoria es especial, pues en vez de detectar que el nodo se ha caído, el nodo sólo nos avisará con un mensaje del tipo {0, me}, de forma que, mientras tratamos el latido, tendremos que actualizar la vista para quitarle a ese nodo su rol (si lo tuviera) y ponerlo en la lista de nodos espera (si no lo estaba ya). Esto se verá representado por el siguiente diagrama de secuencia:



3. Pruebas

Hemos tenido que implementar las pruebas de la 4 a la 9. Lo que hacemos en ellas es:

4. Partiendo de la vista anterior con ID: 2 P: NODOCLIENTE1 C: NODOCLIENTE2, no enviamos latidos de NODOCLIENTE1 durante 200ms, de esta manera, el GV se da cuenta de su caida. Ahora, sólo nos queda comprobar que la vista tiene la forma Primario: *NODOCLIENTE2*, Copia: *HOSTINDEFINIDO*, NumVista: 3.

5. Partiendo de la vista anterior, comprobamos que ante un latido0 de NODOCLIENTE1 (reconexión) este provoca que la vista se actualice a la siguiente forma: Primario: *NODOCLIENTE2*, Copia: *NODOCLIENTE1*, NumVista: 4.

6. Partiendo de la vista anterior, enviamos latido0 de NODOCLIENTE3, para colocarlo en nodos en espera, y no enviamos latidos desde NODOCLIENTE2 durante 200ms. Finalmente, comprobamos que la vista tiene la forma Primario: *NODOCLIENTE1*, Copia: *NODOCLIENTE3*, NumVista: 5.

7. Partiendo de la vista anterior, lanzamos latido0 de NODOCLIENTE2 y comprobamos que la vista no ha cambiado por la reconexión del anterior primario.

8. Partiendo de la vista anterior, comprobaremos que por mucho que los clientes envíen latidos, si no confirman la vista, la vista que estabamos gestionando (la tentativa) es distinta de la válida. Para ello, enviaremos unos cuantos latidos que no confirman la vista tentativa y después preguntaremos por la vista válida. Como en ninguno de nuestros tests hemos validado ninguna vista, vamos a comprobar que la vista es del tipo Primario: msgsys.*HOSTINDEFINIDO*, Copia: msgsys.*HOSTINDEFINIDO*, NumVista: 0

9. En esta última prueba comprobamos el error crítico, para ello, validaremos la última vista que teniamos, y dejaremos morir todos los nodos, esto provocará que la vista se actualice hasta que genere el error crítico, este error cierra el proceso del GV con exit 1, así que comprobamos que no nos podemos conectar al GV y terminamos el test.

El test 9 nos ha obligado a alterar la parte de stopDistributedProcesses, en donde no dejamos que se envíe el MsgFin al GV para evitar un error de connect.