SISTEMAS DISTRIBUIDOS

Práctica 4

Curso 2018-2019

Autores:
Ignacio Palacios Gracia 739359
Rubén Rodríguez Esteban 737215

En esta práctica se introduce el concepto de la tolerancia a fallos para nodos distribuidos con estado. La finalidad de esta práctica es la construcción de un servicio de almacenamiento clave/valor tolerante a fallos utilizando replicación Primario/Copia en memoria RAM.

1. Introducción

En la realización de esta práctica se ha procedido a diseñar un sistema de tolerancia a fallos basado en el servicio de vistas planteado en las clases de teoría de la asignatura a modo gestión de réplicas Primario/Copia. Para facilitar la tarea y lograr una mayor sencillez de diseño el gestor de vistas no estará replicado.

El sistema de gestión de vistas funciona de manera que cuando el primario falla promociona la copia enviada como primario. Si un nodo copia falla o es promocionado y hay nodos en situación de espera, es éste último quién promocionará como nodo copia.

Todas las vistas administradas por el gestor de vistas son identificadas por un número de vista, identificador (nombre completo) nodo primario, identificador (nombre completo) nodo copia, vista válida actual, y una lista de nodos en espera para cada vista.

Cada servidor clave/valor debe enviar un latido de manera periódica como máximo cada 50 ms para notificar que siguen vivos y que no han sufrido ningún problema. En el latido se incluye el identificador de la vista más reciente conocida por los servidores copia y primario. El servidor de vistas responde con la descripción de la vista más reciente del gestor. Si el gestor de vistas no recibe ningún latido de los servidores clave/valor durante un tiempo especificado en el programa el gestor de vistas los considera caídos. Cuando los servidores caídos vuelven a arrancar deben enviar uno o más latidos con argumento cero para informarle al gestor de vistas que de que han caído. A partir de aquí, se convertirá en un servidor en espera. Enviar latido(Node.self(),0) dos veces consecutivas se consideran como caídas consecutivas.

Se pide implementar el servicio de gestión de vistas presentado anteriormente de tal manera que el sistema sea capaz de recuperarse ante el fallo de una réplica. Cuando el primario o la copia fallen, será el gestor de vistas el encargado de reemplazar la configuración con otro nodo operativo.

2. Solución planteada

El diseño del fichero servidor_gv planteado para la solución contiene una definición de una estructura vista, que contiene el número de la vista, su nodo primario, su copia, el número de fallos del nodo primario y de la copia, una lista de nodos en espera, y la vista válida que se proveerá a los clientes.

La recepción de mensajes del servidor se encuentra en un bucle principal, en el cual en cada iteración se recibirá un mensaje y se actuará de la manera correspondiente. Pueden recibirse tres tipos de mensajes diferentes:

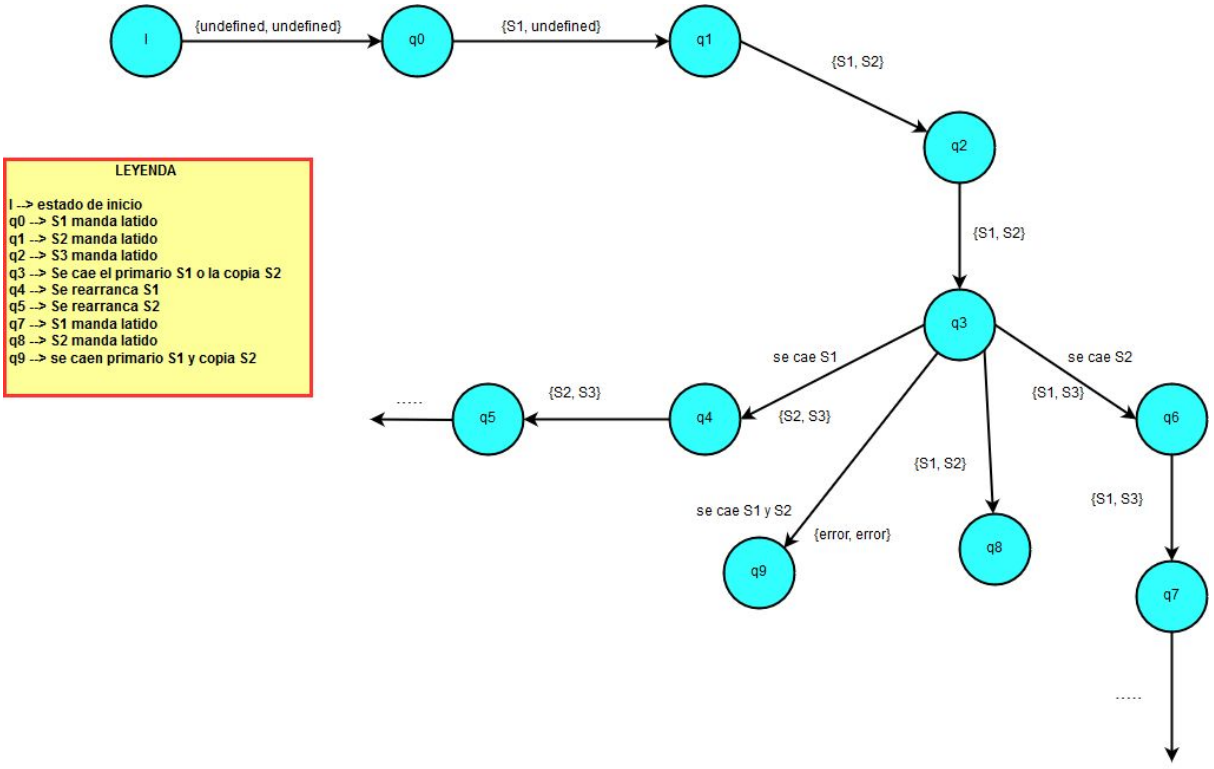
- Latido, el cual se recibe del cliente junto con su número de vista y su pid.
  Al recibir este mensaje, lo primero se comprueba si el cliente que ha mandado el latido es el primario o la copia, y cambia el número de fallos del primario o de la copia a 0. Tras realizar esta comprobación, se comprueba si existe ya un primario o copia en nuestra vista actual. Si el primario de la vista es undefined, se introduce el cliente que nos ha mandado el latido como primario, y si ya existe un primario pero no una copia, se introduce este como copia. Si ya existen ambos, se introduce este nodo en la lista de nodos pendientes.

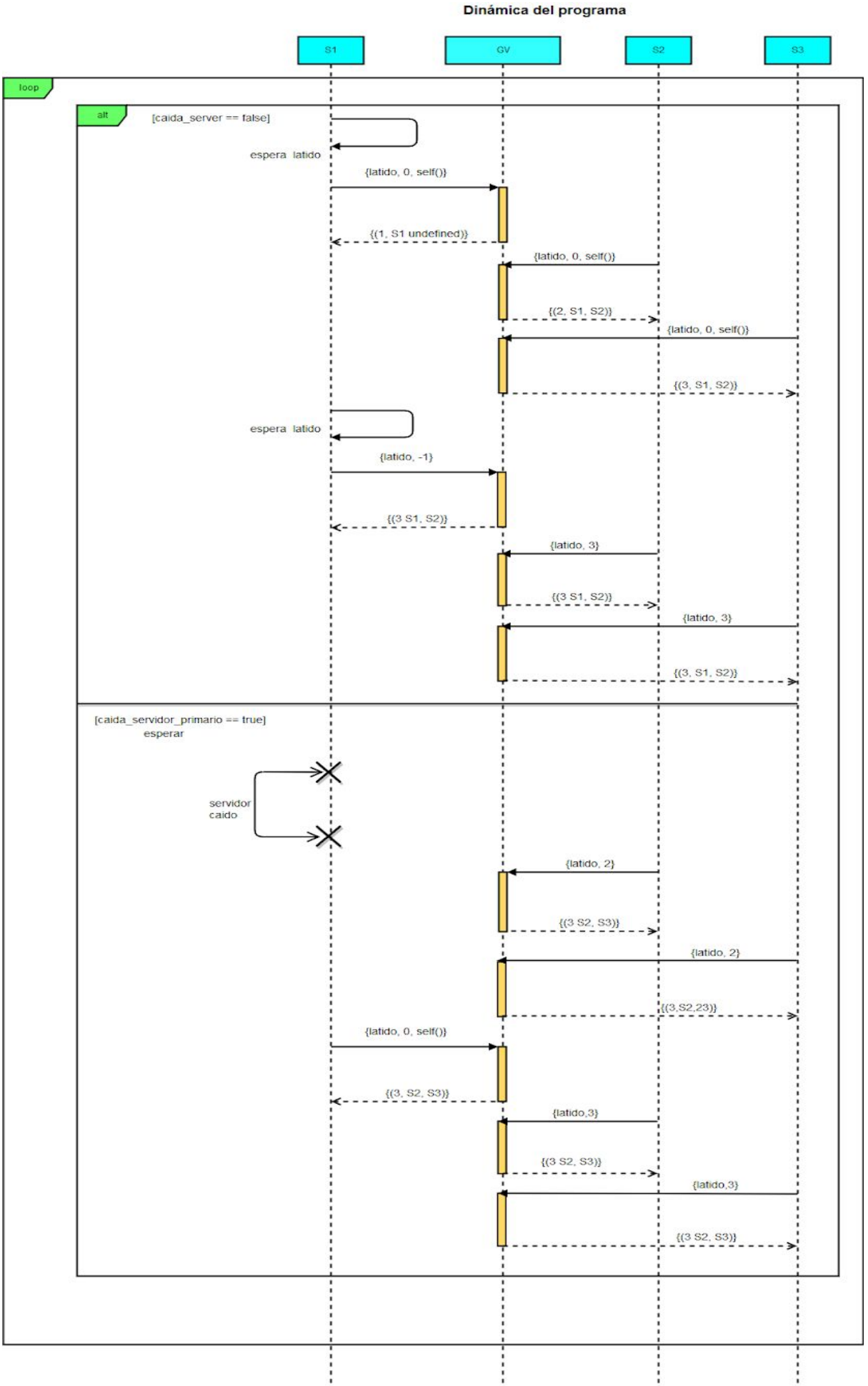
  A continuación, al ser el mensaje recibido un latido, se procede a cambiar la vista tentativa formada anteriormente por una vista válida si tanto el primario como la copia existen y no se han caído. Si el primario se ha caído sin haber confirmado la vista válida, se sustituirán el primario y la copia por una tupla {:error, :error}. Por último se manda al cliente una respuesta con la vista válida, o vista tentativa si se han caído el primero, la copia o ambos.

- Procesar situación de los servidores, el cual se recibe por un proceso creado anteriormente por el servidor y que lo manda una vez cada @intervalo_latido.
  Al entrar a la función que procesa este mensaje, se comprueba que la vista existe, y si existe, se añade un fallo al primario, y a la copia también si existe.
  Si se da el caso de que al comprobar que la copia existe, y la num_vista de la vista válida es diferentes a la vista actual, ponemos toda la vista como error, ya que el primario no ha confirmado la vista tentativa todavía, y podemos suponer que se ha caído.

A continuación se comprueba si el número de fallos del primario o de la copia son iguales al número de latidos fallidos máximos permitidos. Si los fallos del primario son iguales a los latidos fallidos máximos, se cambia el nodo primario por el de la copia en la vista, y se introduce uno de los nodos en espera en la copia. Si el que ha superado los fallos máximos permitidos ha sido la copia, se sustituye esta por uno de los nodos de la lista de espera. En estos dos casos, si no existe ningún nodo esperando en la lista, se sustituye el caído por el valor undefined. Si tanto el primario como la copia superan el número máximo de fallos, la vista actual se sustituye por una vista inicial con los valores predeterminados.

- Obtener vista, el cual se recibe del cliente.
  Tras recibir este mensaje, el servidor comprueba si existe una vista válida para mandarle al cliente. Si esta existe, le manda todos los valores de la vista válida actual junto con un valor true, y si no, le manda un valor undefined, o un error, junto con otro valor false.




Dinámica del programa
S1
GV
S2
S3
loop
alt
[caida_server == false]
espera latido
{latido, 0, self()}
{{1, S1 undefined}}
{latido, 0, self()}
{{2, S1, S2}}
{latido, 0, self()}
{{3, S1, S2}}
espera latido
{latido, -1}
{{3 S1, S2}}
{latido, 3}
{{3 S1, S2}}
{latido, 3}
{{3, S1, S2}}
[caida_servidor_primario == true]
esperar
servidor caido
{latido, 2}
{{3 S2, S3}}
{latido, 2}
{{3,S2,23}}
{latido, 0, self()}
{{3, S2, S3}}
{latido,3}
{{3 S2, S3}}
{latido,3}
{{3 S2, S3}}

3. Validación

Para poder llevar a cabo la evaluación y puesta a prueba de la práctica, el programa ha sido ejecutado en múltiples máquinas. Primero se probó lanzando el script de inicialización creando todos los nodos en la misma máquina. Una vez se comprobó que todos los test facilitados por el profesorado de la asignatura, y los test realizados por los alumnos de este grupo, funcionaban correctamente, se procedió a lanzar cada uno de los clientes y el servidor en máquinas diferentes, confirmando así que funcionaban correctamente tanto en local como de forma remota.

Los test realizados para poner a prueba el programa son los siguientes:

1. No debería haber primario antes de tiempo.
2. Hay un primer primario correcto.
3. Hay un nodo copia.
4. Copia toma relevo si primario falla.
5. Servidor rearrancado se convierte en copia. 6
6. Servidor en espera se convierte en copia si primario falla.
7. Primario rearrancado es tratado como caído y convertido en nodo en espera.
8. Servidor de vistas espera a que primario confirme vista pero este no lo hace.
9. Si anteriores servidores caen (Primario y copia), un nuevo servidor sin inicializar no puede convertirse en primario.

Se fueron probando de uno en uno, cambiando el código y el diseño del programa según superase los test o no. Una vez un test era superado, se pasaba al siguiente, así hasta que el código superase todos los test realizados. Una vez probados los test en una sola máquina, se probó el servidor en diferentes ordenadores personales, y por seguridad también se probó en el laboratorio de la universidad.

4. Conclusión

A lo largo de esta práctica se ha diseñado y puesto en funcionamiento un servidor de gestión de vistas, y se ha probado su correcto funcionamiento mediante una serie de tests proporcionados por el profesorado y creados por los alumnos. Para ello se ha definido una estructura de la vista compuesto por primario, copia, número de fallos del primario, número de fallos de la copia, una lista de los nodos en espera, y la vista válida actual.

El diseño del servidor de vistas se ha planteado con un bucle principal, en el que cada iteración recibirá un mensaje y actuará acorde al mensaje recibido. Puede recibir un mensaje latido de uno de los nodos, un obtener vista de uno de los nodos, y un mensaje para procesar la situación de los servidores, el cual es enviado por el mismo servidor. El código fue verificado por los test de verificación una vez terminado el diseño.