Implementación de un Servidor de Gestor de Vistas

Sergio Herrero Barco, NIA: 698521

Alex Oarga Hategan, NIA: 718123

Práctica 4 – Sistemas Distribuidos

Noviembre de 2017

Introducción

El servicio de gestor de Vistas consiste en un servidor tolerante a fallos que mantiene y gestiona una vista. Una vista es un reflejo del conjunto de miembros (primario y copia) que almacena el servicio. Cuando se incorpora un nuevo miembro o cae un miembro antiguo, la vista se actualiza con las incorporaciones/caídas, y envía a los demás miembros la nueva vista. También debe encargarse de controlar los miembros que siguen operativos.

Sección principal

Implementación del servidor

Para la implementación del servidor, se ha definido un tipo de dato struct, que va a representar una vista. Contiene un campo llamado *num_vista* que va a incluir en todo momento el número de vista, un campo llamado *primario* que va a contener la dirección del nodo primario de esa vista, y por ultimo un campo llamado *copia* que incluye el nodo copia.

El servidor debe almacenar una serie de datos en todo momento, es decir, tiene un estado. El estado se va a componer de una vista llamada *t_vista* que tendra la vista tentativa del servidor, una vista llamada *v_válida* que tendrá la vista valida y será la enviada a los clientes, y por ultimo una lista de cuantos latidos llevan fallidos el primario y la copia de la vista tentativa. Como en la vista no es necesario guardar aquellos nodos que van a estar en espera, se ha decidido meterlos en la lista de los latidos a partir de la segunda posición (primera y segunda corresponden al nodo primario y al nodo copia)

El servidor se debe de encargar fundamentalmente de 3 tareas. Debe encargarse de enviar la vista **valida** a aquellos clientes que se lo soliciten tras un mensaje. Debe gestionar las solicitudes de entrar al sistema de los nodos a través de los **latidos**. Y por último debe procesar la situación de los servidores que se comunican con él, es decir, comprobar quienes siguen vivos y quienes se han caído.

Para realizar estas funciones, el servidor se encuentra en todo momento esperando la recepción de un mensaje. Cuando un mensaje llega, cambia de estado, procesa el mensaje y finalmente vuelve al estado inicial. Se puede ver los estados posibles en la *figura 1*.

Los mensajes que puede recibir (Véase *Figura 2*) y las acciones que debe hacer se muestran a continuación:

- :latido, X, dirección: Al llegar un latido, debe analizar que numero de vista envía. Si el número de vista es 0, y coincide en que el servidor tiene como numero de vista tentativa un 0, significa que es un nuevo nodo que quiere incorporarse al sistema y entrará como **primario.** Si el número de vista es 0, pero el servidor no tiene como número de vista tentativa un 0, significa que o bien es un nuevo nodo que quiere incorporarse, o bien es un nodo caído que ha vuelto a conectar y quiere volver a entrar. En ambos casos, si la copia esta vacía entrara como copia, si no entrará como nodo en espera. Si el número de vista es -1, es un caso especial en el que el primer nodo que se incorpora al sistema contesta, y se le envía la vista tentativa actual. Por ultimo si llega un

numero de vista diferente de 0, se le enviara la vista tentativa actual. Al llegar cualquier latido, se va a poner los nodos fallidos del remitente a 0.

- :obtén_vista, dirección: Al llegar dicho átomo, se le enviara la vista actual, y un dato booleano que le dirá si la vista actual y la tentativa coinciden. La vista actual y la tentativa coinciden si y solo si el primario de la vista tentativa confirma dicha vista. Es el mensaje con el que se comunicaran los clientes con el servidor.
- :procesa_situacion_servidores: Es un mensaje que llega de manera periódica cada cierto tiempo (50 ms por ej.). Es el encargado de avisar al servidor que debe comprobar la situación de los servidores. Lo primero que va a hacer es sumar 1 a todos los latidos fallidos de los nodos que almacena. Seguidamente se comprobará si los nodos caídos han sido el primario y la copia, en caso afirmativo es un error crítico y el sistema se va a caer. Si no se ha dado el caso se comprobará si el número de latidos fallidos del primario es mayor que 4, (es el límite que se ha puesto). Si ha caído, se promocionará la copia a primario y un nodo espera a copia (si existe). Después se comprobará si el número de latidos fallidos de la copia es mayor que 4, si ha caído se promocionara un nodo espera a copia. Si no hubiera nodo en espera, la vista solo se compondría del nodo primario, por lo que no se asegura la **tolerancia a fallos**. Por último, se comprobará si hay algún nodo caído dentro de los nodo espera. En caso afirmativo, simplemente se le elimina de la lista de nodos espera.

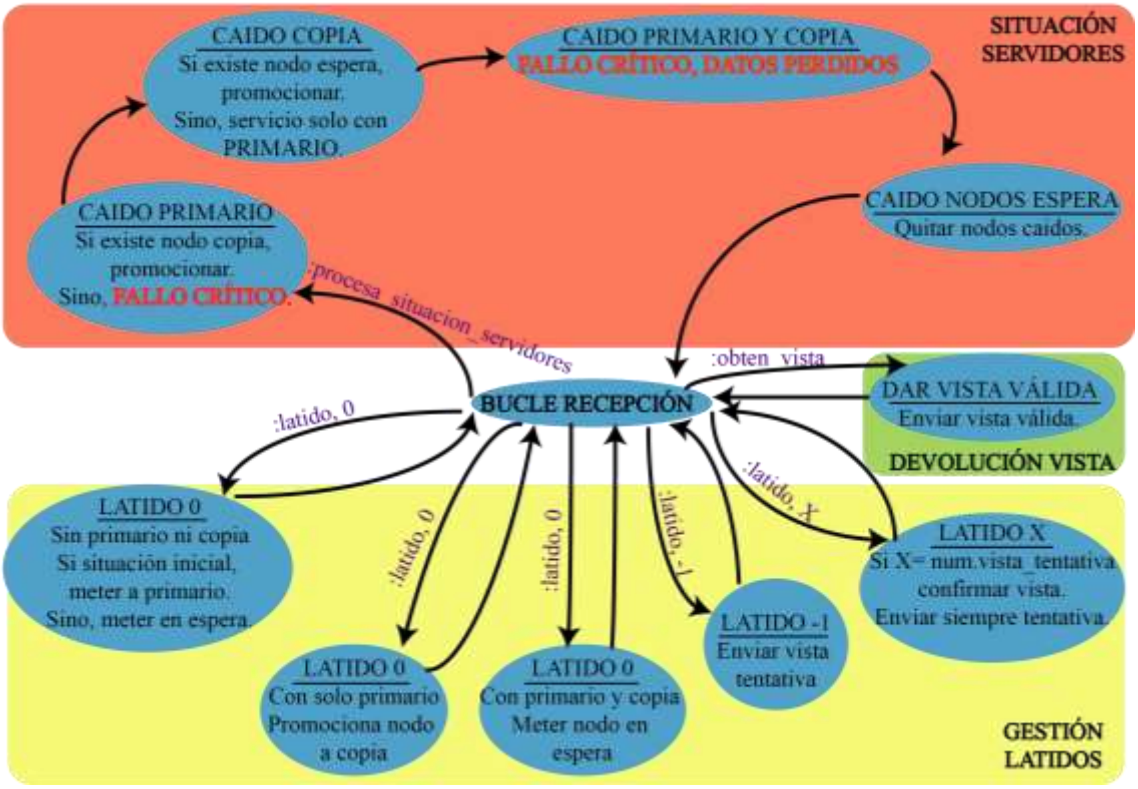

Figura 1: Diagrama de estados

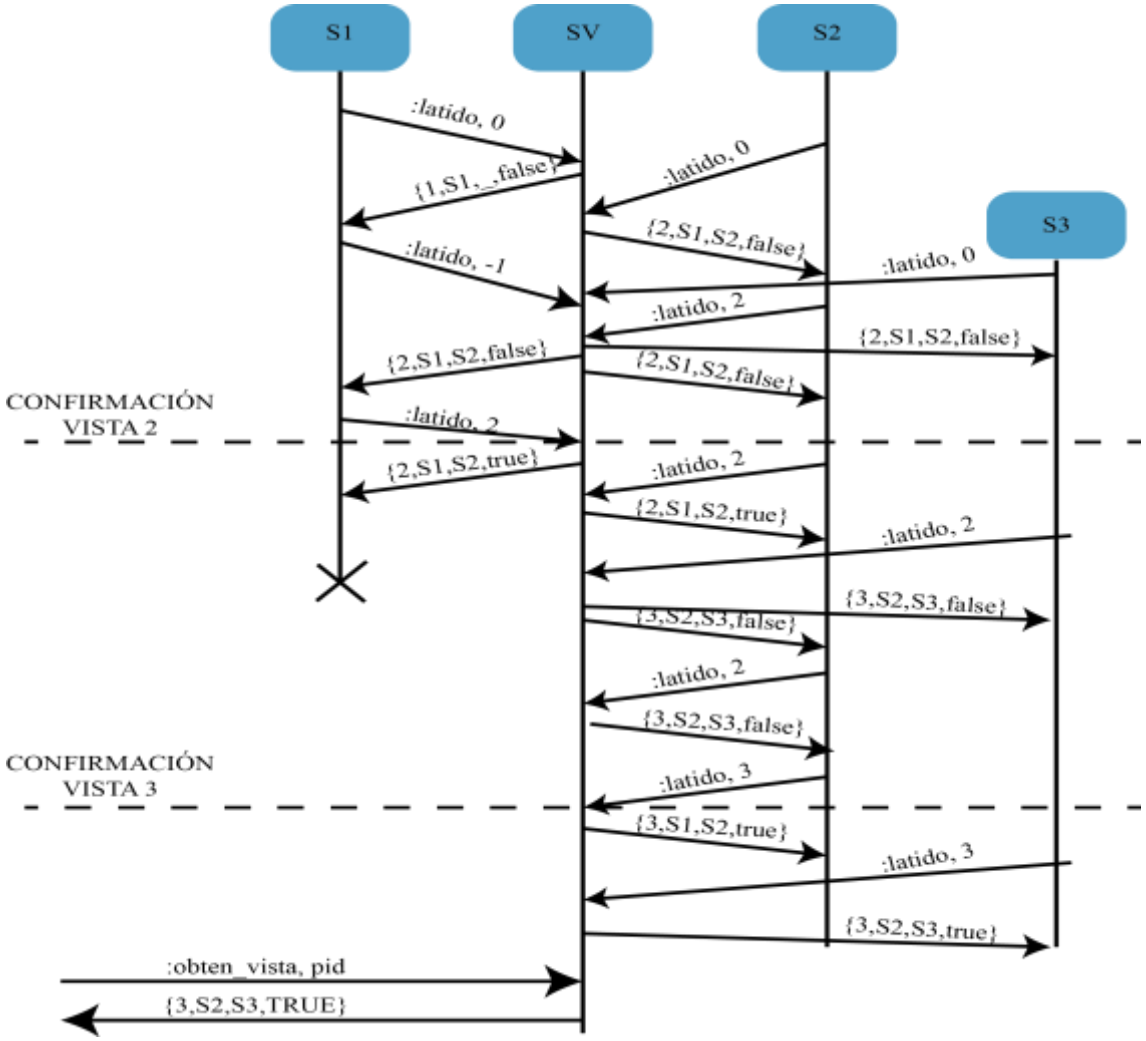

Figura 2: Diagrama de secuencia de mensajes (solo latidos y obtén_vista)

Validación

Para asegurarse de la correcta implementación, se ha usado el fichero de test de elixir. Al realizar las pruebas en una red Lan, se ha cambiado las direcciones dentro del fichero de las maquinas.

Para que funcione todo de manera correcta, se ha tenido que copiar la llave publica para poder acceder al ssh sin tener que poner la contraseña, y asegurarse que los ficheros están en el mismo *path* en las diferentes maquinas.

Dado que el fichero estaba incompleto, a continuación, se mostrará las acciones para superar el test completo:

- Test 7: Como el servidor c2 estaba caído, se envía un latido 0 para avisar de que vuelve a estar operativo. Después se obtiene la vista tentativa actual, y se comprueba si c2 no aparece como primario ni como copia. Si es así, **Test Superado**.
- Test 8: Como es contexto nuevo, c3 envía latido 0, y se convierte en primario, c1 envía latido 0 y se convierte en copia y finalmente c2 envía latido y se convierte en nodo en espera. Inmediatamente después se tira al primario, y se comprueba que, una vez caído, el nodo c1 no ha sido promocionado de copia a primario. Si es así, **Test Superado**
- Test 9: Como es contexto nuevo, c1 envía latido 0 y se convierte a primario, c2 envía latido 0 y se convierte en copia. Se confirma la vista valida, y a continuación se tiran los nodos primarios y copia. Después, c3 envía latido 0 como nuevo nodo. Si se comprueba que c3 no ha entrado como primario, **Test Superado.**

Tras todo esto, se ha ejecutado el test y se han obtenido 9 de 9 test superados.

Conclusión

Para la realización de un gestor de vistas totalmente tolerante a fallos, se debería replicar el servidor de gestor de vistas. Al no estar replicado, ante una posible caída del nodo que alberga el servidor, los datos se perderían y sería un **fallo crítico**.

Al estar los servidores replicados, se permite tener una tolerancia a fallos en cuanto a los datos que van a estar almacenando, en este caso una base de datos que se accede mediante clave.