Práctica 4.
Primario - Copia

Andrés Gavín Murillo 716358

Borja Aguado Díez 741440

5 de diciembre de 2018

Introducción

Se ha desarrollado un sistema con tolerancia a fallos con estado, concretamente, mediante el sistema de replicación Primario/Copia. Se plantea una solución basada en el servicio de vistas visto en clase. Este no está replicado, por lo que no se plantea una solución completa de tolerancia a fallos.

Sección Principal

La práctica ha sido desarrollada en Elixir. Se ha implementado el sistema siguiendo el algoritmo de tolerancia a fallos con estado enseñado en clase [1]. Si el primario cae por cualquier motivo, la copia disponible en ese momento toma su lugar y el sistema continúa normalmente. En este contexto, una "vista" representa un estado del sistema, es decir, quién es el nodo primario, quién es el nodo secundario y cuál es el número de vista.

A continuación se explica el funcionamiento del sistema desarrollado.

Arquitectura del sistema

El gestor de vistas gestiona una secuencia de vistas numeradas, y cada vista se representa como una tupla de 3 valores: {nº de vista, identificador (nombre completo) nodo primario, identificador (nombre completo) nodos copia}. En Elixir, la vista inicial del sistema es la siguiente:

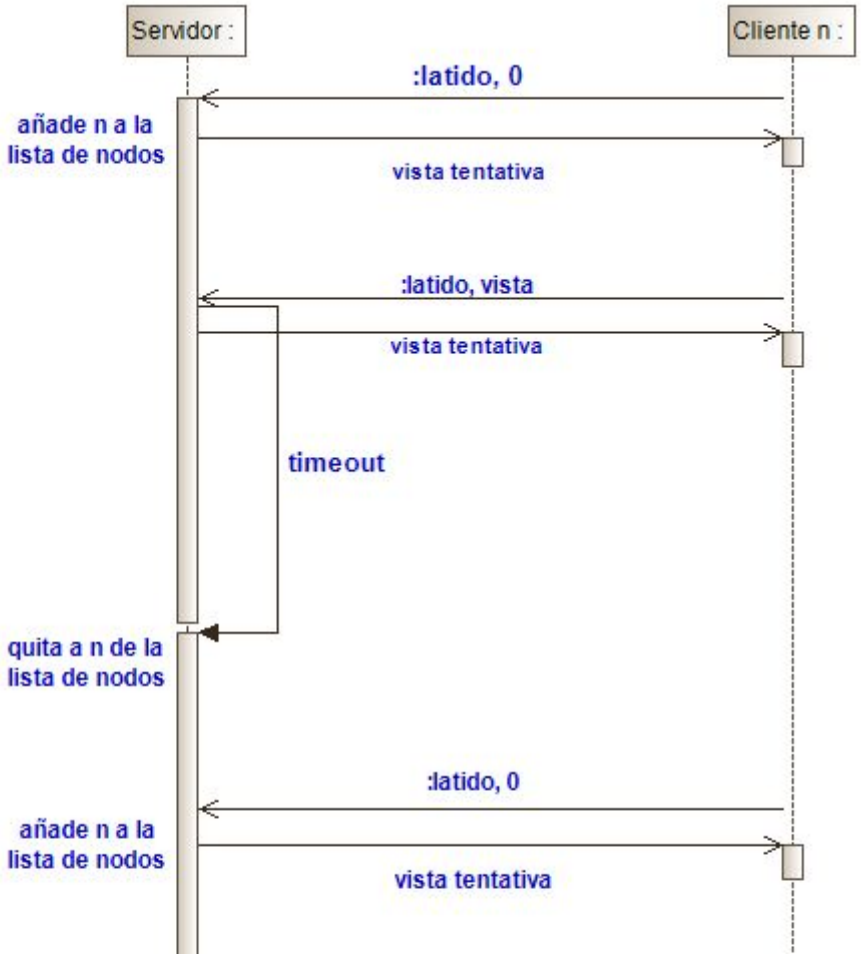
```
{num_vista: 0, primario: :undefined, copia: :undefined}
```

El primario, salvo al inicio del sistema, es el primario o una copia del estado anterior, con el objetivo de garantizar la preservación del estado. Cada uno de los servidores envía constantemente un mensaje al servidor de vistas, antes de que finalice un intervalo de @intervalo_latido (por defecto 50 ms), para indicar que el nodo sigue operativo. Este mensaje se conoce como "latido".

Dicho latido incluye el número de vista más reciente conocido. Si el nodo no envía un latido durante un número @latidos_fallidos de @intervalo_latido, se considerará que ha caído. Tanto al iniciar el sistema como cuando un nodo se recupera de una caída, envía al gestor de vistas un latido con argumento 0.

A continuación se explica dicho proceso con un diagrama de secuencia.

[1] Diapositiva 11 del pdf "L08 – Tolerancia a fallos con estado"



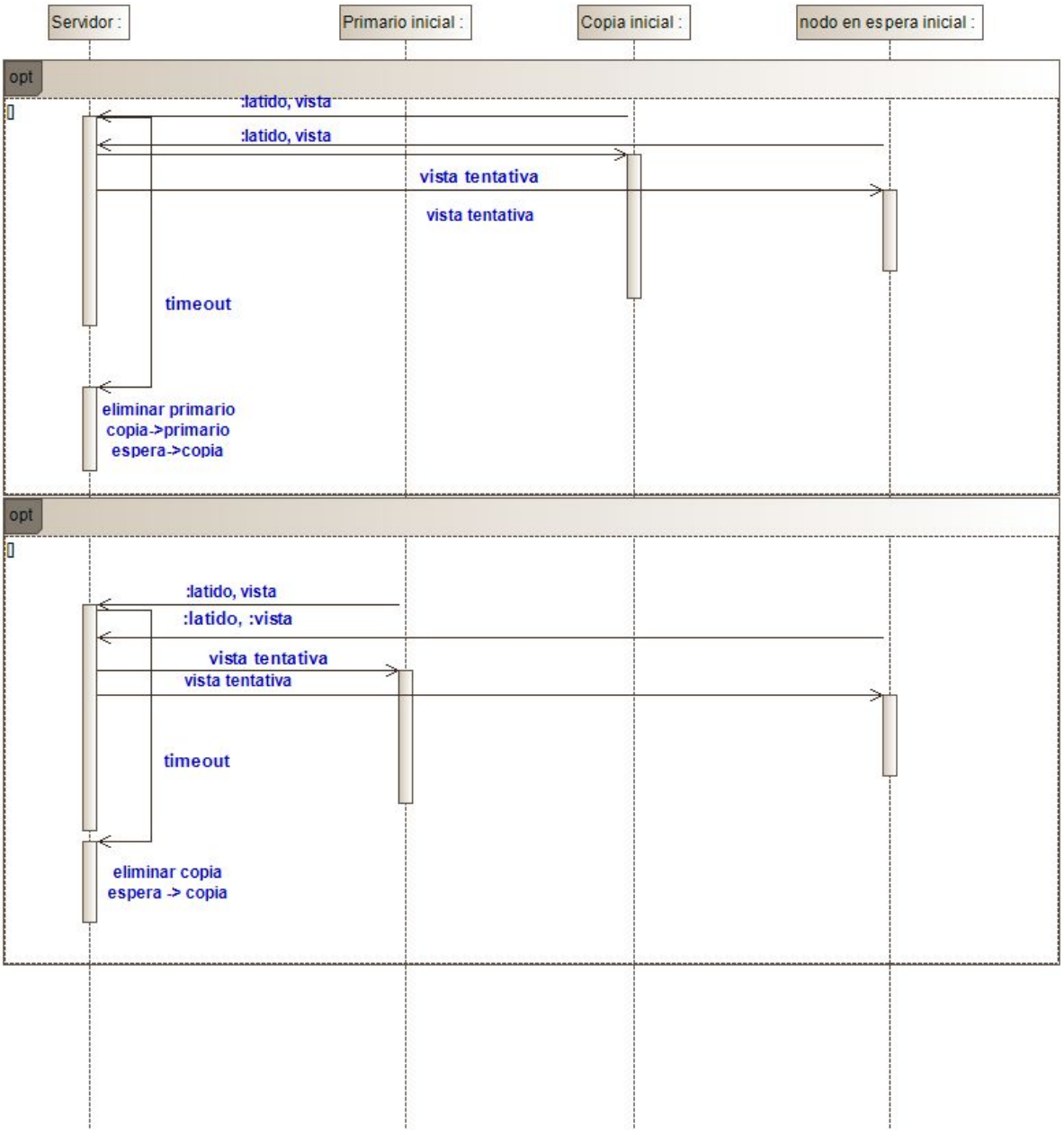
Cada @intervalo_latido se procesa la situación del servidor.

Si existe al menos un nodo (el primario), se actualizan los latidos de todos ellos. Después, se eliminan los nodos en espera que han caído y se comprueba si el primario o la copia han caído. Si ambos han caído, se ha perdido la consistencia del sistema. Se alerta de ese error y el sistema ha dejado de funcionar correctamente.

En el caso de que el primario haya caído, se promociona el nodo copia como primario, y si existen nodos en espera, el primer nodo de la lista se promociona como nodo copia.

Si ha sido la copia la que ha caído, se promociona el primer nodo que se encuentre en espera (si existe) como copia.

El siguiente diagrama de secuencia ilustra este comportamiento.


Servidor :
Primario inicial :
Copia inicial :
nodo en espera inicial :
opt
:latido, vista
:latido, vista
vista tentativa
vista tentativa
timeout
eliminar primario
copia->primario
espera->copia
opt
:latido, vista
:latido, :vista
vista tentativa
vista tentativa
timeout
eliminar copia
espera -> copia

Validación

Para la validación del sistema, se han realizado 9 tests, con resultados satisfactorios, explicados a continuación:

1. Se comprueba que en la ejecución inicial del sistema no hay primario.
2. Se añade un nodo primario y se comprueba que la vista es la correcta.
3. Partiendo del punto anterior, se añade un nodo copia y se comprueba que la vista es la correcta.
4. Partiendo del punto anterior, falla el primario y se comprueba que copia es promocionada como primario.
5. Partiendo del punto anterior, rearranca el anterior primario, y se comprueba que es convertido en copia.
6. Partiendo del punto anterior, se añade un servidor en espera y, posteriormente, falla el primario. Se comprueba que el servidor en espera se ha convertido en copia.
7. Partiendo del punto anterior, rearranca el anterior primario y se comprueba que es convertido en nodo en espera.
8. Se añaden un primario, una copia y un nodo en espera, pero, el primario falla y no confirma la vista. Se comprueba que la copia no es promocionada como primario porque la vista no estaba confirmada.
9. Se añaden un servidor primario y una copia (confirmados), pero fallan. Se añade un nuevo servidor, y se comprueba que no se ha convertido en primario.

La traza de ejecución al lanzar el sistema es la siguiente:

Tests 1-7:

```
Test: No deberia haber primario ...
 ... Superado
Test: Primer primario ...
. ... Superado
Test: Primer nodo copia ...
. ... Superado
Test: copia toma relevo si primario falla ...
.AVISO: No hay copia.
 ... Superado
Test: Servidor rearrancado se conviert en copia ...
. ... Superado
Test: Servidor en espera se convierte en copia ...
.nil
 ... Superado
Test: Primario rearrancado tratado como caido y es convertido
        en nodo en espera ...
.nil
 ... Superado
```

Tests 8:

Test: Servidor de vistas espera a que primario confirme vista
pero este no lo hace ...
... Superado

Tests 9:

Test: Servidores caen y un nuevo servidor no puede convertirse
en primario ...
ERROR: Se han perdido el primario y la copia.
... Superado

Conclusiones

Los algoritmos de tratamiento de errores como el utilizado en esta práctica son imprescindibles para el correcto funcionamiento de un sistema distribuido en un sistema real.

La buena implementación de algoritmos como el de tolerancia a fallos con estado garantizan que el sistema es capaz de soportar caídas de ciertos de sus nodos sin afectar a su ejecución.