RAFT - 1ªPARTE

SISTEMAS DISTRIBUIDOS - PRÁCTICA 3

Lizer Bernad Ferrando - 779035
Lucía Morales Rosa – 81690

---

1 INTRODUCCIÓN

En esta práctica se ha llevado a cabo la implementación de un servicio clave-valor empleando una replicación distribuida basada en Raft. Se ha implementado una versión de Raft que elige correctamente a un líder sobreponiéndose a algunos fallos básicos y que trata de forma adecuada la adición de entradas nuevas en las réplicas sin fallos.

Además, para la conexión entre máquinas se emplean las llamadas RPC "*AppendEntries*" y "*RequestVote*".

2 DISEÑO DEL ALGORITMO

Para realizar esta práctica se ha hecho uso del documento *In Search of an Understandable Consensus Algorithm (Extended Version)* de Ongaro y Ousterhout (2014), desarrollado en la Universidad de Stanford y proporcionado con el guion de la práctica. A partir de este documento, se ha extraído el diseño para la máquina de estados de los nodos mostrado en la Ilustración 1.

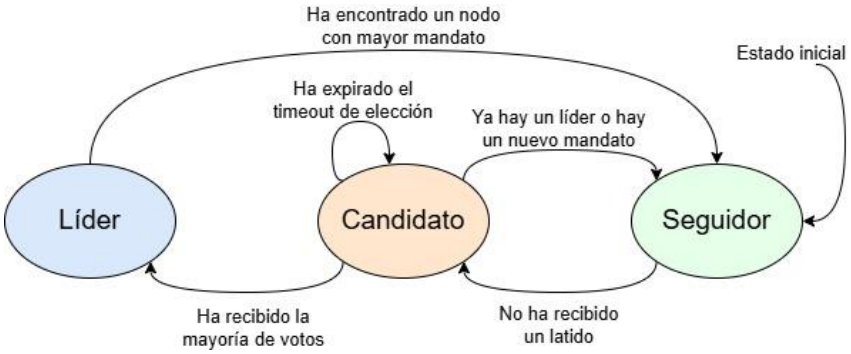

Ilustración 1: Máquina de estados de los nodos en Raft

En Raft, cada nodo puede tener un único estado de entre los tres estados posibles: Líder, Candidato o Seguidor. Inicialmente todos los nodos comenzarán siendo seguidores y establecerán un *timeout* en el que deberán recibir un latido del líder. Como esto no ocurrirá puesto que no hay líder, el seguidor al que primero le venza el *timeout* pasará a ser candidato y comenzará una elección.

Cuando un nodo es "Candidato" establecerá un *timeout* aleatorio para la elección, de forma que, si se vence dicho *timeout*, se reinicie la elección. A continuación, comenzará la elección votándose a sí mismo para líder y enviando peticiones de voto al resto de nodos que la aceptarán o no dependiendo de algunos parámetros que se indican en la Sección 3.2. Al obtener la respuesta de un nodo, el candidato comprobará si debe pasar a seguidor porque el mandato del nodo que ha respondido es mayor o si tiene una mayoría de votos y puede convertirse en el líder.

Lizer Bernad Ferrando - **779035**
Lucía Morales Rosa - **816906**

Si el nodo candidato consigue convertirse en "Líder", establecerá un *timeout* de 50ms para enviar latidos periódicos a todos los nodos, que se caracterizan por tener el parámetro "EntradaLog" vacío. En respuesta a ese latido, los seguidores y candidatos confirmarían su posición de seguidor y responderían al líder con su mandato. De esta forma, el líder puede comprobar si su mandato es mayor o igual al del resto de nodos o si está desactualizado y debe dejar de ser el líder.

Además, al nodo líder le van a llegar peticiones de los clientes. En este caso, el líder en lugar de mandar un latido con el parámetro "EntradaLog" vacío, rellenará ese campo con el mandato y la operación especificada por el cliente. Los seguidores, al recibir este mensaje a través de la llamada RPC "*AppendEntries*", manejarán la operación uniéndola en su Log y respondiendo de forma exitosa a la llamada. Cuando el líder reciba una respuesta exitosa de más de la mitad de los nodos, tomará la entrada como comprometida y responderá al cliente.

Como en esta versión básica de Raft las entradas comprometidas no se aplicarán a la máquina de estados, el trabajo de los nodos respecto a las entradas comprometidas terminará en este punto, mostrando un mensaje por pantalla que indique que se ha comprometido en el líder.

El comportamiento del sistema descrito se puede observar de forma gráfica en la Ilustración 2.

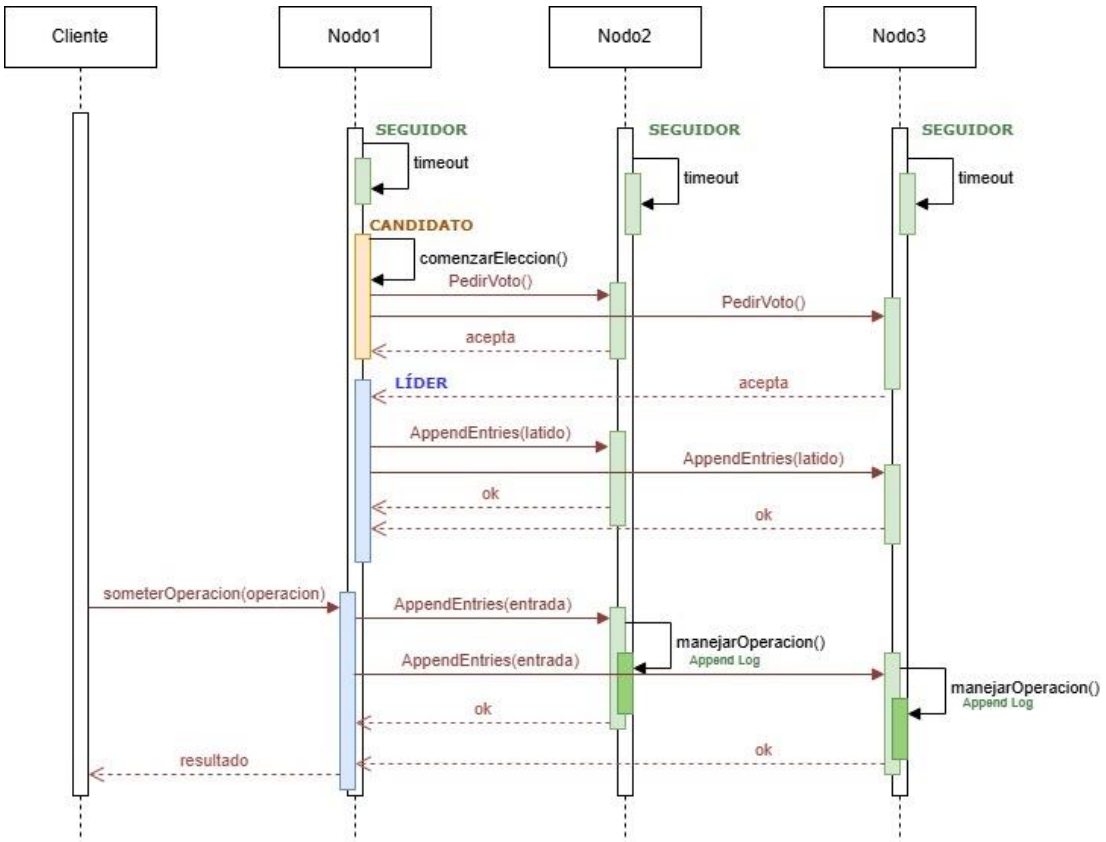

Ilustración 2: Diagrama de secuencia de Raft con 3 nodos y un cliente

Nota: Para facilitar la comprensión del diagrama, se han representado las llamadas RPC entre los procesos con flechas rojas para diferenciarlas de las llamadas a función representadas con líneas negras.

Lizer Bernad Ferrando - **779035**
Lucía Morales Rosa - **816906**

3 ASPECTOS RELEVANTES EN LA IMPLEMENTACIÓN

3.1 Estructuras de datos

3.1.1 NodoRaft

La estructura "NodoRaft" es responsable de la representación de un único nodo o replica de Raft. Esta estructura almacena todos los campos necesarios para el correcto funcionamiento del nodo.

Para empezar, "NodoRaft" incluye un *mutex* para proteger el acceso compartido a determinadas variables y evitar así las condiciones de carrera. También incluye un array con las direcciones y puertos de las demás replicas del sistema y dos campos enteros que indican el identificador de proceso de este nodo y del proceso que cree que es el líder. Además, contiene un campo "Logger" para emplear en el proceso de depuración y seguir un registro de las trazas.

Por otro lado, se han definido tres canales de tipo booleano para permitir el envío de eventos. Estos canales son: "Latido", que permite notificar que se ha recibido un latido del líder; "SoySeguidor", que permite indicar al nodo que su estado ha sido modificado a seguidor y debe cambiar su comportamiento y "SoyLider" que indica al nodo que su estado ha pasado a ser el de líder.

También cuenta con otros campos como "*CurrentTerm*" que almacena el último mandato que ha visto el nodo, un campo "*VotedFor*" que guarda el identificador de proceso del candidato por el que ha votado el nodo y un *array* de "EntradaLog" que representa el log del nodo que debería ser replicado en algún momento en la máquina de estados.

Finalmente, cuenta con el campo "Estado" que puede tomar los valores LIDER, SEGUIDOR o CANDIDATO, y el campo "NumVotos" que lleva el recuento de los votos obtenidos cuando el nodo es un candidato.

3.1.2 ArgsPeticionVoto

Esta estructura es empleada cuando un candidato desea obtener votos de los otros nodos. La estructura de datos "ArgsPeticionVoto" cuenta con los campos "Term" e "IdCandidato" que el candidato inicializa con su mandato actual y su número de proceso. Con estos datos, el nodo al que se le solicita el voto puede validar la legitimidad del candidato tal y como se explica en la Sección 3.2.

3.1.3 RespuestaPeticionVoto

La estructura "RespuestaPeticionVoto" tiene los campos "Term", donde el nodo al que se le solicita el voto establece su número de mandato y el campo "VotoDado". El campo "VotoDado" es un booleano que indica si ha dado el voto al candidato que lo solicitaba o no.

3.1.4 ArgAppendEntries

Esta estructura es enviada por el líder para indicar los latidos a los demás nodos y enviarles las entradas con las operaciones solicitadas por los clientes. "*ArgAppendEntries*" cuenta con cuatro campos para almacenar el mandato en el que se realiza la llamada RPC, el identificador del proceso líder, la entrada de log vacía en caso de ser un latido y con contenido si se envía una entrada y el número de la última entrada comprometida por el líder.

Lizer Bernad Ferrando - **779035**
Lucía Morales Rosa - **816906**

3.1.5 Results

La estructura "Results" es empleada por los nodos seguidores que han recibido un latido para responder al líder con su mandato, de forma que este pueda valorar si debe o no seguir siendo líder y con un campo booleano para indicar si la entrada ha sido unida exitosamente al log del seguidor.

3.2 PedirVoto

El método "PedirVoto" recibe como parámetros una petición de voto de tipo "ArgsPeticionVoto" y una respuesta "RespuestaPeticionVoto" por referencia.

En la petición, el candidato ha especificado su mandato y su identificador de proceso, con los cuales el nodo puede determinar si el candidato debería convertirse en líder.

Para ello, el nodo compara su mandato con el del candidato y si el del candidato es menor, el nodo rechazará la petición de voto automáticamente. En caso de ser iguales, se evaluará también que el nodo no haya votado por otro candidato anteriormente y si no es el caso, dará su voto al candidato que ha realizado la solicitud. Finalmente, si el mandato del candidato es mayor estricto que el del nodo, este le votará modificando entonces también su mandato para estar actualizado y pasando a ser seguidor en caso de que su estado fuera otro.

Para responder al candidato, el nodo empleará la estructura "RespuestaPeticionNodo" donde indicará su mandato, de forma que si es mayor el candidato pueda actuar en consecuencia dejando de serlo y con el campo "VotoDado" para indicar si ha sido votado o no por el nodo.

3.3 AppendEntries

El método "AppendEntries" recibe por referencia un argumento de tipo "ArgAppendEntries" y otro de tipo "Results".

Este método comprueba si la llamada RPC la ha producido un latido o la necesidad de evaluar una entrada dependiendo del contenido del campo "Entries" de la estructura "ArgAppendEntries". Si el contenido del campo es vacío significa que es una llamada de latido, por lo que el nodo que la recibe debe comparar su mandato con el recibido por el líder para reafirmar su estado de seguidor.

Para ello, si el mandato del líder es mayor al del nodo, el nodo actualiza los campos "idLider", y "CurrentTerm". Además, reafirma su estado de seguidor y notifica que ha recibido un latido a través del campo "nr.Latido" de la estructura "NodoRaft" para resetear el temporizador.

En caso de que los mandatos sean iguales, el nodo receptor reconocerá al otro como líder. Sin embargo, si el mandato del receptor es mayor, no reconocerá al líder y responderá con su mandato para que el líder sepa que hay otro nodo más actualizado que él y pueda dejar de serlo.

Por otro lado, si el líder no está enviando un latido y quiere que el nodo añada una entrada a su log, mandará la entrada que desea unir en el campo "Entries" y el nodo receptor unirá el log que tuviera en ese momento con la nueva entrada, devolviendo además un éxito al líder a través del campo "Success" de la estructura "Results".

Lizer Bernad Ferrando - **779035**
Lucía Morales Rosa - **816906**

4 Tests de prueba

Se han diseñado cuatro tests de prueba para verificar el correcto comportamiento del algoritmo de Raft. Para ejecutar los tests se lanzan tres nodos replica en las direcciones definidas previamente en el fichero "internal/testintegracionraft1/testintegracionraft1_test.go" y tras realizar las comprobaciones se detienen los nodos.

El primer test "soloArranqueYparada" obtiene comprueba el estado de cada nodo haciendo uso de la llamada "obtenerEstadoRemoto" y verifica que los valores son los esperados antes de inicializar el protocolo de Raft en las réplicas. Para que esto se satisfaga, el mandato obtenido debería ser 0, no debería ser líder y el identificador del proceso líder debería ser -1.

La segunda prueba "ElegirPrimerLider" comprueba que hay un único líder, haciendo uso de la función "pruebaUnLider".

El test "FalloAnteriorElegirNuevoLider" desconecta al nodo líder tras verificar que dicho nodo existe y se espera hasta que los nodos restantes convoquen una elección y vuelva a surgir un líder.

Finalmente, en la prueba "tresOperacionesComprometidasEstable" se comprueba que se puedan comprometer 3 operaciones en tres entradas teniendo un líder estable y sin fallos. Para ello, se verifica que existe un líder y se someten tres operaciones cualesquiera haciendo uso de la llamada RPC "SometerOperacionRaft".