PRÁCTICA 2

Problema de los Lectores y Escritores Distribuido

PRÁCTICA 2 1

Introducción 2

1. Algoritmo de Ricart-Agrawala 2

2. Problema Lectores y escritores 3

3. Validación experimental 5

4. Conclusiones 6

Aarón Ibáñez Espés 779088
Ángel Espinosa Gonzalo 775750

Introducción

El siguiente documento consiste en una memoria técnica de la práctica 2 de la asignatura de sistemas distribuidos. En ella se resuelve el problema de los lectores y escritores distribuido utilizando el algoritmo de Ricart-Agrawala generalizado y se proporciona una validación experimental de la implementación utilizando la librería GoVector en la que se comprueba su correcto funcionamiento.

1. Algoritmo de Ricart-Agrawala

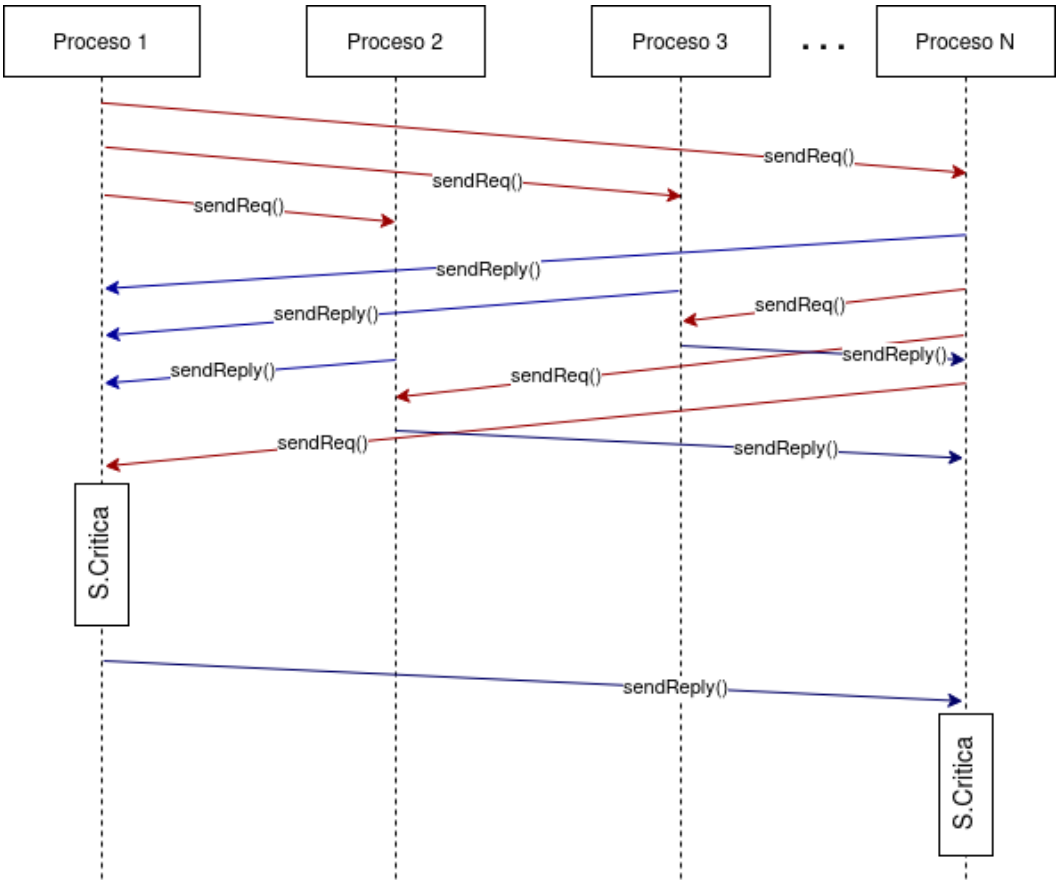

Imagen 1. Algoritmo de Ricart-Agrawala (Rojo: REQUEST, Azul: REPLY)

El algoritmo de Ricart-Agrawala es un mecanismo de control de acceso a la sección crítica para sistemas distribuidos en el cual N procesos distribuidos que no comparten memoria, se comunican mediante el paso de mensajes asíncronos.

El comportamiento general del algoritmo de Ricart-Agrawala es:

- Si el proceso i quiere acceder a la sección crítica y recibe una petición con un timestamp mayor al suyo, este la difiere y continua su ejecución hacia la sección crítica.
- En caso de recibir una petición para acceder a la sección crítica y no querer acceder a ella o que dicho mensaje tenga un timestamp menor al suyo, el proceso responde

con un mensaje de permiso para que el emisor de la petición pueda acceder a la sección crítica.

En la imagen 1 se muestra un ejemplo donde el proceso 1 quiere acceder a la sección crítica, para ello ejecuta el preprotocol, donde envía un mensaje de solicitud (*sendReq()*), al resto de procesos de la red y espera su respuesta de permiso. Una vez ha recibido el permiso de todos los procesos para acceder a la sección crítica. Éste la ejecuta y al salir, en el postprotocol, responde a las peticiones que le habían llegado anteriormente con el permiso, dando lugar así a que el proceso N entre en la sección crítica.

2. Problema Lectores y escritores

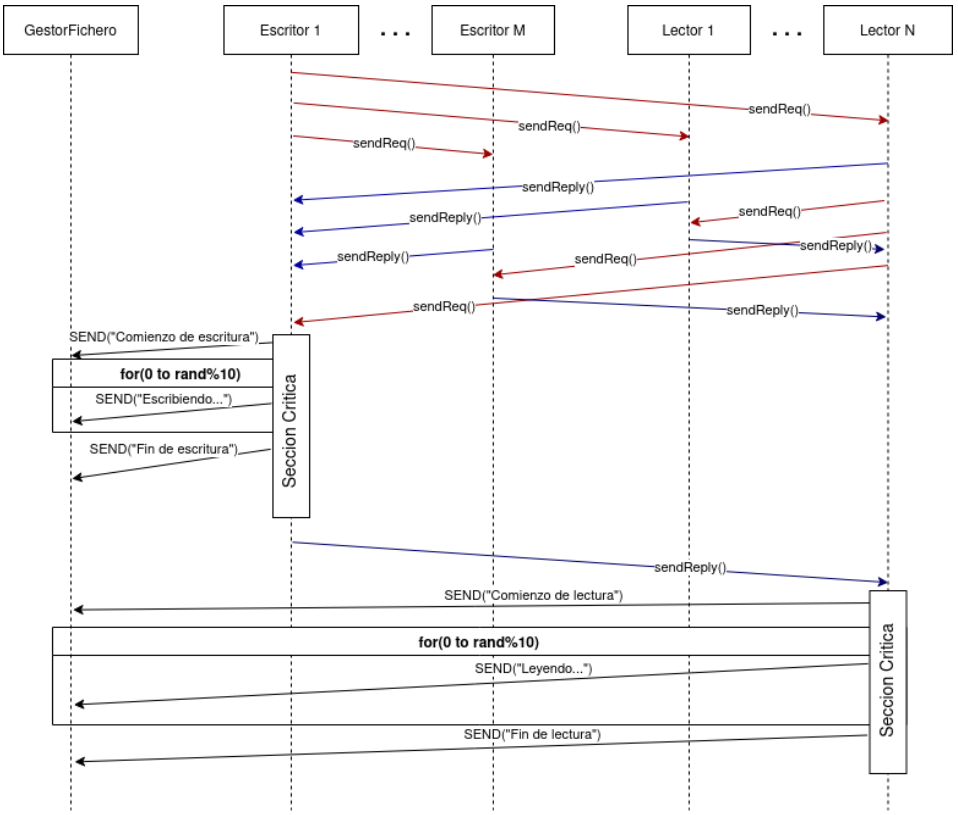

Imagen 2. Arquitectura de lectores y escritores distribuidos

La arquitectura lector-escritor cuenta con N procesos lectores, M procesos escritores y una base de datos compartida a la cual los procesos quieren acceder en lectura o escritura.

Además, se tiene un proceso llamado gestorFichero, encargado de gestionar el acceso a la base de datos, teniendo en cuenta las siguientes reglas de exclusión:

1. Solamente podrá haber un proceso escribiendo en la base al mismo tiempo.
2. Varios procesos lectores pueden estar leyendo de la base en el mismo momento.
3. No puede haber un proceso escritor y un proceso lector accediendo a la base de datos concurrentemente.

Para garantizar que estas reglas se cumplen se ha definido una matriz de exclusión con los permisos entre las operaciones.

En la imagen 2 se muestra un ejemplo de su ejecución donde el proceso Escritor 1 quiere escribir en la base de datos, para ello envía los mensajes de solicitud de permiso al resto de los procesos de la red, y una vez recibido el permiso de todos ellos accede a la sección crítica, es decir, a escribir en la base de datos. Mientras el Escritor 1 se encuentra en la sección crítica, el Lector N envía su solicitud para acceder a la base. Todos los procesos responden con el permiso excepto el Escritor 1, que se encuentra en la sección crítica, por ello difiere la petición. Una vez el Escritor 1 ha acabado la escritura, responde a las peticiones que habían sido diferidas, para continuar con la ejecución y garantizar así el acceso en exclusión mutua a la sección crítica.

En cuanto a la carga de red generada por la comunicación de los procesos, el algoritmo de ricart-Agrawala, es más eficiente que la implementación de los relojes lógicos de Lamport, siendo el total de mensajes enviados de 2*(N-1), (N es el número total de procesos de la red). Este es el número de mensajes que se envían en el caso peor de que un proceso quiera acceder a la sección crítica, por lo que pese a que el número de mensajes sea inferior al de otras implementaciones, en el caso de construir una red con muchos nodos podría tener problemas de escalabilidad si la calidad de la red no fuese buena. Además de los mensajes de comunicación entre procesos, se envían (2 + rand()%10) mensajes al proceso gestorFichero, cada vez que uno de los procesos ejecuta la sección crítica.

3. Validación experimental

Para la validación experimental se ha utilizado la librería de GoVector. Esta permite generar mensajes de log por cada uno de los usuarios del sistema distribuido, es decir, por cada uno de los lectores y escritores.

Una vez generados un fichero de log para cada uno de los usuarios del sistema, se unifican en un fichero nombrado *Shivz.log* y se crea el siguiente diagrama de paso de mensajes.

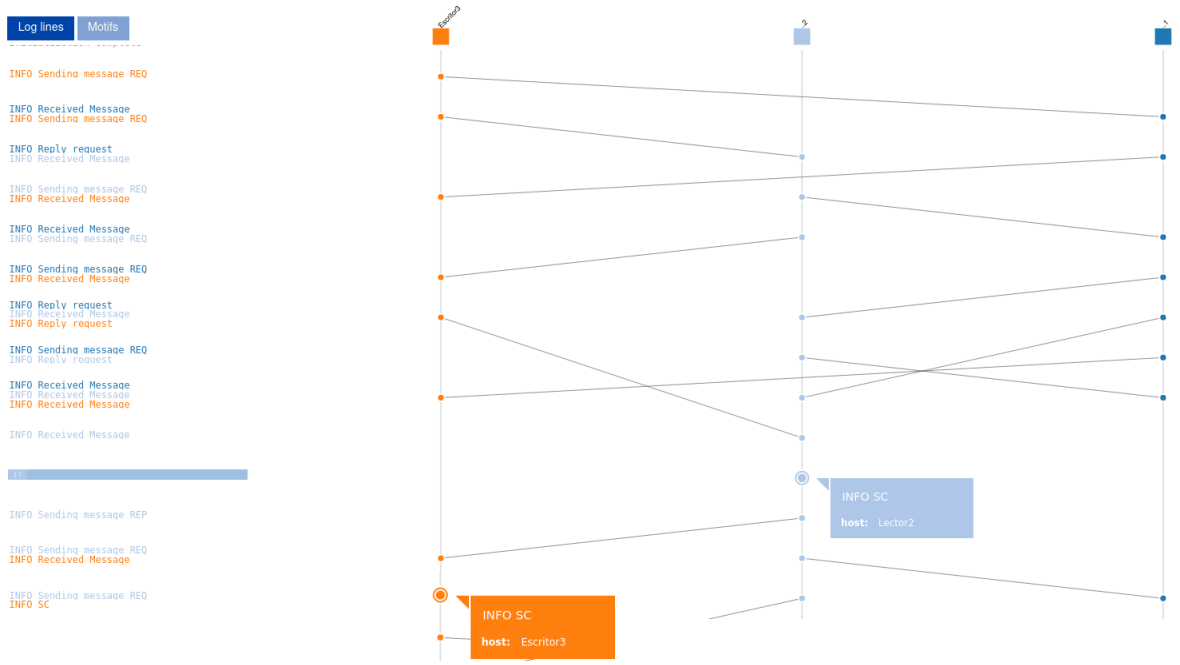

Imagen 3. Diagrama de mensajes generado por Shviz

En la imagen 3 se puede ver la traza de mensajes generada por el sistema con dos procesos lectores y un proceso escritor. Se puede ver como el Escritor 3 manda los dos mensajes de request, sin embargo, el proceso Lector 2, también manda los mensajes de request y es el primero en recibir el permiso por los otros dos procesos de la red, por lo que entra en la sección crítica. Una vez ha terminado de ejecutar la sección crítica, envía el mensaje de permiso al proceso Escritor 3 y este ya tiene permiso para ejecutar la sección crítica.

Esta traza de mensajes permite comprobar el correcto funcionamiento del sistema, debido a que se ha comprobado el acceso a la sección crítica en exclusión mutua, lo que garantiza que el fichero, en caso de los lectores y escritores distribuido, no tendrá inconsistencias.

Para la utilización de esta librería, se han modificado los siguientes ficheros :

- ra .go→ raGoVec.go
- escritor.go → escritorGoVec.go
- lector.go → lectorGoVec.go

Además, para comprobar que se ha implementado correctamente el algoritmo y que la matriz de exclusión de operaciones funciona bien, se analiza el fichero *sharedDatabase.txt*.

```
PROCESO: 3: Escribiendo...
PROCESO: 3: Fin de escritura
PROCESO: 1: Comienzo de lectura
PROCESO: 1: Leyendo...
PROCESO: 1: Leyendo...
PROCESO: 1: Leyendo...
PROCESO: 2: Comienzo de lectura
PROCESO: 2: Fin de lectura
PROCESO: 1: Leyendo...
PROCESO: 1: Leyendo...
PROCESO: 1: Leyendo...
PROCESO: 1: Fin de lectura
PROCESO: 2: Comienzo de lectura
PROCESO: 2: Leyendo...
PROCESO: 2: Leyendo...
PROCESO: 2: Leyendo...
PROCESO: 2: Fin de lectura
PROCESO: 3: Comienzo de escritura
PROCESO: 3: Escribiendo...
PROCESO: 3: Fin de escritura
PROCESO: 2: Comienzo de lectura
PROCESO: 2: Leyendo...
PROCESO: 2: Leyendo...
PROCESO: 2: Leyendo...
```

Imagen 4. Fragmento del fichero sharedDatabase.txt

En la imagen 4 se puede observar cómo se produce un acceso concurrente al fichero por los procesos lectores 1 y 2, sin embargo, el proceso escritor 3 espera hasta que ha acabado la lectura para comenzar a escribir.

4. Conclusiones

Finalmente, gracias a la realización de la práctica se ha podido comprobar el potencial del algoritmo de Ricart-Agrawala para gestionar el acceso a la sección crítica en un sistema distribuido.

Sin embargo, pese a su mayor eficiencia en el envío de mensajes y de no requerir de una estructura auxiliar como una heap para controlar el acceso a la sección crítica, este depende en gran medida de una red de comunicación libre de errores.