Práctica 2. Chat Distribuido

Jorge Aznar López

Jorge Fernández Muñoz

8 de noviembre de 2018

Introducción al algoritmo original en Algol

El trabajo propuesto en esta sesión de prácticas consiste en implementar un chat distribuido en Elixir basado en el algoritmo original de Ricart-Agrawala, el cual utiliza una variación de los relojes lógicos de Lamport (debido a las restricciones que tenían el tamaño de los enteros cuando se implementó en espacio de memoria) para determinar el orden de emisión de los mensajes, y así poder mostrarlo a todos los participantes correctamente.

Arquitectura del sistema

El sistema usado para validar la implementación del algoritmo se compone de tres nodos participantes, los cuales ejecutan cinco procesos aislados que se encargan de:

- Gestionar la exclusión mutua en los accesos a la base de datos
- Gestionar la consulta de datos de la base, así como los propios datos
- Recibir las peticiones de mandar un mensaje de otros participantes
- Enviar las peticiones de mandar un mensaje a otros participantes
- Recibir las respuestas a la solicitud de mandar un mensaje (ACK) y los mensajes a mostrar (y mostrarlos)

Algoritmo en Elixir

El algoritmo en Elixir es el encargado de crear y lanzar estos procesos en cada nodo participante al comienzo.

El proceso asignado a la gestión de la exclusión mutua se ha implementado simulando el paso de testigo, este recibe la petición del proceso solicitante y sí éste no se halla esperando el regreso del testigo procede a enviárselo al proceso solicitante para luego esperar su respuesta con el testigo de regreso.

La base de datos compartida entre el resto de los procesos contiene las variables del algoritmo original en Algol y se encarga de, según el tipo de petición y los datos solicitados, responder a las peticiones del resto de procesos.

El proceso receptor de peticiones se mantiene a la espera de recibir una solicitud de enviar un mensaje, una vez llegada actualiza la base de datos con la información recibida (número de secuencia más alto), solicita el testigo y decide quien debe enviar primero según el cálculo de prioridades del algoritmo original y procede a liberar el testigo para terminar enviando el ACK de confirmación si el emisor tiene la prioridad, o añadiéndolo a la cola de espera si es el propio nodo participante el que debe comunicarse primero por el chat.

Para emitir las peticiones se espera a que el nodo participante quiera emitir un mensaje y una vez quiere solicita el testigo, actualiza la base de datos (su número de secuencia), libera el testigo y procede a enviar la solicitud de comunicarse a el resto de los participantes y los ACK a los nodos con peticiones diferidas si ya ha enviado su mensaje.

El proceso ocupado de recibir los ACK se mantiene a la espera de recibir todos los ‘N-1’ mensajes de confirmación (en el caso práctico dos) y procede a notificar a la base de datos los ACK recibidos para, recibidos todos, enviar al resto de participantes su mensaje y mostrarlo por pantalla.

Variantes respecto al algoritmo original

El algoritmo original utiliza una base de datos compartida, que se ha implementado en un proceso aislado (el cual se invoca a sí mismo en cada actualización de la base) al cual envían peticiones el resto los procesos con acceso en exclusión mutua, el cuál se tuvo que implementar haciendo uso de un proceso adicional que gestiona el paso de testigo.

Asimismo, el algoritmo en Elixir utiliza sus propias primitivas (send y request) para que los procesos se comuniquen entre ellos y realicen las peticiones concretas necesarias para imitar el comportamiento del algoritmo original en Algol.

Validación empírica de algoritmo implementado

Para verificar el correcto funcionamiento del algoritmo se han utilizado tres nodos cliente en tres ordenadores distintos ocupados de correr los cinco procesos de cada nodo participante, estos nodos envían mensajes cada 1,5 segundos, automatizando así el envió de mensajes y verificando si el orden de prioridad establecido se mantenía a lo largo de la comunicación y si los participantes mostraban en orden correcto los mensajes recibidos. Se utilizaron concretamente tres nodos para asegurar que la probabilidad de mostrar los mensajes en desorden si el algoritmo fallaba era mayor, validando así que este funcionaba correctamente y en base a lo especificado en el enunciado.

Conclusiones

El algoritmo implementado en Elixir sigue la estructura propuesta en Algol con las varianzas ya explicadas anteriormente y se comporta de manera correcta, mostrando los mensajes siguiendo el orden de prioridad establecido por los relojes de Lamport y el propio algoritmo de Ricart-Agrawala.