# SSDD

## Práctica 2

**Pablo Orduna Lagarma y Daniel Naval Alcalá**

08 de noviembre de 2018

# INTRODUCCIÓN

A lo largo del desarrollo de esta práctica se ha implementado un chat distribuido peer to peer, de manera que no exista ningún proceso coordinador, sino que todos los procesos participantes tienen la misma responsabilidad en el sistema. En chat a implementar se intercambiarán mensajes entre sí con N nodos y el mensaje que envíe uno de los participantes tendrá que aparecer inmediatamente en las pantallas de todos los demás integrantes del chat.

# IMPLEMENTACIÓN

Para la implementación del sistema propuesto se han utilizado 5 procesos, los cuales se ejecutarán paralelamente en cada nodo para garantizar el correcto funcionamiento del chat distribuido. Con la finalidad de mantener el algoritmo lo más parecido posible a la implementación original en el lenguaje de programación ALGOL se han utilizado los mismos 3 procesos para la sincronización de la entrada a la sección crítica:

### distributed_critical_section

Es el proceso que pide el acceso a la sección crítica (request) y el realiza acciones en exclusión mutua una vez se encuentra dentro. Para conseguir adaptar el algoritmo para funcionar en un chat se ha añadido en dichas acciones un bucle que envía el mensaje escrito al resto de nodos. Además, este proceso es el que se dedica a obtener por entrada estándar el mensaje que se va a enviar, lo cual se hace con un IO.gets() al principio de cada iteración, de manera que el proceso se queda bloqueado y no pide acceso a la sección crítica hasta que se haya escrito el mensaje. Finalmente se responde (reply) a los nodos que se habían guardado en la lista de diferidos.

### receive_request_messages

Es el proceso dedicado a recibir las peticiones (request) de otros nodos, a los cuales responde (reply) o los añade a la lista de diferidos para responderlos más tarde dependiendo de si está en la sección crítica y de su número de secuencia en el caso de que así sea.

### receive_reply_messages

Este proceso se dedica a leer las respuestas (reply) de otros nodos y resta 1 por cada respuesta a la variable outstanding_reply. A estos 3 procesos presentes en el algoritmo original se les suman un proceso que actúa como mutex para y un proceso dedicado a la recepción de los mensajes del chat de otros nodos:

### mutex

Este proceso es el único que tiene acceso a las variables que tienen que ser accedidas en exclusión mutua por el resto de los procesos. Por tanto, se dedica a recibir mensajes de petición de acceso o modificación de dichas variables.

Los mensajes de petición están formados por: el pid del proceso que envía dicho mensaje (con la finalidad de responderle cuando haya terminado), un identificador de la acción que debe realizar el proceso mutex y, en ocasiones, ciertos valores necesarios para la finalización de la acción pedida.

Posteriormente se realizan las operaciones que se deben ejecutar en exclusión mutua y una vez terminadas, se manda un mensaje de confirmación con el identificador ":ok" al proceso que había realizado la petición y se vuelve a comenzar el bucle.

Dependiendo de la petición realizada, el mensaje de confirmación podrá contener a su vez ciertos valores que sean necesarios para el proceso que haya realizado dicha petición.

Cabe destacar que, con la finalidad de que el proceso distributed_critical_section no se quede realizando una espera activa mientras hasta que outstanding_reply sea 0, en el proceso mutex se manda un mensaje a dicho proceso con identificador ":ok_0" cuando la variable llegue a 0.

De esta manera el proceso distributed_critical_section se queda esperando en un receive y no se encuentra realizando una espera activa.

### show_messages

Como el propio nombre indica, este proceso se dedica a mostrar los mensajes de chat que han llegado, los cuales se diferencian por llevar el identificador ":mensaje".

Para que la interfaz sea lo más parecida posible a la de un chat se ha decidido mostrar a la izquierda de cada mensaje recibido la dirección del nodo que lo envía como se comentará a continuación en el apartado de validación.

Los procesos anteriores se lanzan desde una función llamada init, a la cual se le pasa la lista de los nodos y el ID del nodo en el que se va a ejecutar el programa.

Esta función registra los procesos antes de lanzarlos con la finalidad de facilitar la comunicación entre procesos.

# VALIDACIÓN

Para la verificación del correcto funcionamiento del sistema se han tenido en cuenta diferentes escenarios y posibilidades que garantizan que todos los aspectos del algoritmo de Ricart y Agrawala para exclusión mutua distribuida.

Una vez completado el programa se procede a probar la comunicación con dos nodos en la misma máquina para comprobar el correcto funcionamiento con la configuración más sencilla. Al obtener resultados positivos se prueba con más nodos por máquina para que el mensaje tenga que llegar a múltiples nodos y sigan apareciendo en el orden exacto. Finalmente se realiza una prueba vía internet en la que los nodos se encuentran en máquinas diferentes a una distancia considerable y atravesando el área local.

Para comprobar que el algoritmo funcione en los casos en el que un nodo quiere entrar en la sección crítica, pero debe quedarse esperando, se ha implementado un envío de mensaje con retado, que fuerza a esperar un tiempo determinado al proceso durante su estancia en la sección crítica; con esto se puede verificar que no hay problema alguno cuando un proceso tiene que esperar para posteriormente entrar en la sección crítica y luego enviar el mensaje.

Como método para facilitar la identificación del emisor del mensaje se incluye en cada línea recibida el PID completo de quién lo ha enviado compuesto por el nombre de nodo y su dirección seguido del mensaje.

# CONCLUSIÓN

El desarrollo de esta práctica ha resultado más ameno que la anterior al realizar un programa más dinámico que requiere la gestión de menos nodos simultáneamente y el desarrollo ha podido ser más mecánico al contar con un algoritmo de referencia funcional. Para toda la implementación de lo que es el chat se ha utilizado en todo momento como base los tres procesos imprescindibles del algoritmo de Ricart y Agrawala en el que se ha pensado como acoplar lo que es el sistema de un chat al sistema de exclusión mutua que ofrece el algoritmo, como la ubicación del envío y recepción de mensajes.

# BIBLIOGRAFÍA


- ✓ https://elixir-lang.org
- ✓ https://hexdocs.pm
- ✓ https://stackoverflow.com
- ✓ https://elixirforum.com
- ✓ https://thepugautomatic.com
- ✓ http://elixir-recipes.github.io
- ✓ https://www.oreilly.com