# SISTEMAS DISTRIBUIDOS

# Práctica 2

*Curso 2017-2018*

**Autores:**
Ignacio Palacios Gracia 739359
Rubén Rodríguez Esteban 737215

## 1. Resumen

Esta práctica tiene como objetivo implementar un chat distribuido peer to peer, de forma que no exista ningún proceso coordinador, sino que todos los procesos participantes tengan la misma responsabilidad en el sistema.

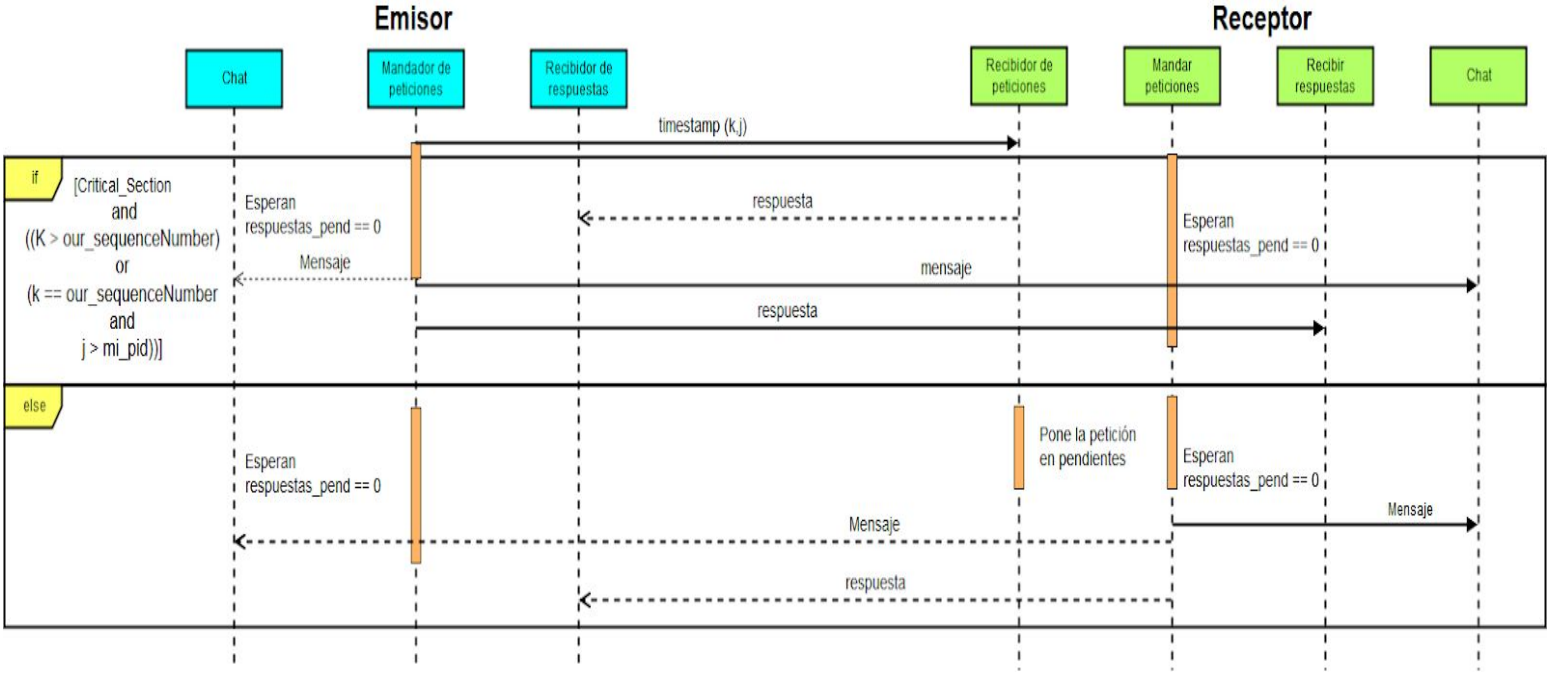
## 2. Introducción

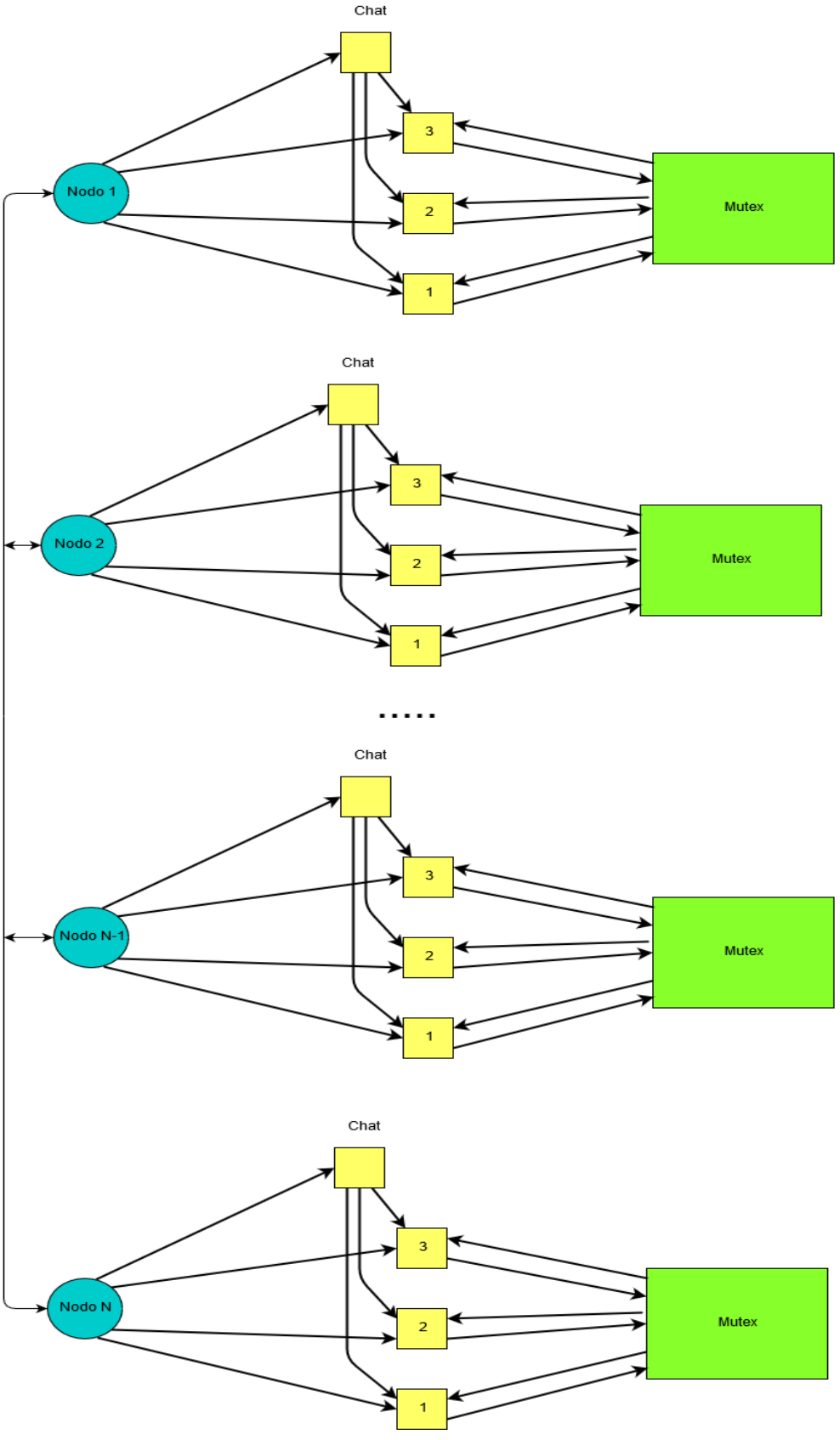
En esta práctica se va a implementar un chat distribuido. Una aplicación de chat distribuido permite a un grupo de usuarios enviarse mensajes de texto entre sí: lo que uno escribe, aparece inmediatamente en las pantallas de todos los participantes. Sin embargo, será requisito indispensable que todos los mensajes de texto se aparezcan en el mismo orden a todos los usuarios.

## 3. Solución planteada

Para llevar a cabo la implementación de este chat se ha utilizado el algoritmo de Ricart-Agrawala como solución. El chat está formado por todos aquellos nodos que participan en la configuración. Por simplicidad de diseño se ha optado por especificar los nodos de una forma estática y conocida a priori, dicho número de nodos se consigue a partir de una lista anteriormente introducida con todos los pid de los nodos a conectar, siendo estos compuestos por el nombre registrado y el nombre del nodo.

El algoritmo de Ricart-Agrawala ha sido implementado con las características vistas en clase y según la estructura original propuesta en ALGOL, de esta manera los procesos acceden a la sección crítica utilizando estampillas temporales o timestamps. Se estampilla cada mensaje con un valor que nos permite decidir si un cierto mensaje debe aparecer antes o después de otro. Además, se mandan mensajes de difusión múltiple, mandandolos a cada uno de los nodos antes de que otro proceso envíe el siguiente mensaje. A continuación se muestra un diagrama de como se ha implementado el sistema y qué procesos han intervenido.




Chat
3
2
1
Nodo 1
Mutex
Chat
3
2
1
Nodo 2
Mutex
.....
Chat
3
2
1
Nodo N-1
Mutex
Chat
3
2
1
Nodo N
Mutex

De acuerdo a como se muestra en el esquema, todos los nodos que intervienen constan de cuatro procesos, identificados con las cuatro cajitas. Cada una de estas cajitas está marcada con un identificador. Concretamente el proceso con identificador 1 es el proceso destinado a mandar peticiones a cada uno de los nodos menos a él mismo, esperar las respuestas, mandar el mensaje al chat y enviar las respuestas pendientes a los nodos que las habían pedido.. El proceso con identificador 2 es el encargado de recibir peticiones, comparar los identificadores y los números de secuencia de ambos nodos, y enviar una respuesta a dicho nodo o ponerlo en la lista de pendientes. El proceso con identificador 3 se encarga de recibir las respuestas a raíz de la petición enviada anteriormente a dicho nodo.

Por último, el proceso con identificador 4 es el chat. Este último se encarga de recibir todos aquellos mensajes procedentes de otros nodos, ya que es el único proceso que contiene el pid almacenado en la lista de nodos a conectar. Una vez recibidos, o bien los redirecciona al proceso correspondiente de ese mismo nodo, o se trata del mensaje a sacar por pantalla.

Adicionalmente, existe un proceso llamado mutex. Este proceso se encarga de almacenar todas las variables compartidas por ese nodo y de garantizar el acceso en exclusión mutua al resto de procesos. El acceso por parte de los demás procesos a este almacén se realiza mediante diversos send/receive, en los que se le indican al mutex realizar diversas acciones o devolverles un valor de los que está almacenando, garantizando así el acceso por parte de los tres procesos del nodo en exclusión mutua.

Cabe indicar que al iniciar todo el proceso de transmisión de mensajes, se le introduce por parámetros el identificador de nodo que usará éste, y que se utilizará especialmente en la generación de mensajes automáticos.

## 4. Validación

Para probar el correcto funcionamiento del sistema se ha probado principalmente alterando el número de participantes, llegando a variar este número de dos nodos al principio, hasta diez en total al final, aumentando cada prueba de dos en dos. También se ha realizado pruebas con los mensajes, como cambiar el tamaño desproporcionadamente a éstos, o mandando en el mensaje el identificador del nodo y el segundo exacto en el que se ha creado para poder comprobar que llegaban en un orden correcto y ordenado, y facilitar así la comprobación de que en todos los nodos se recibían todos los mensajes en el mismo orden.

## 5. Conclusión

A lo largo de esta práctica se ha aprendido a implementar, tanto en elixir como de modo conceptual, el algoritmo de Ricart-Agrawala y los relojes lógicos de Lamport. Para implementarlo en elixir se ha tomado como modelo el código en ALGOL propuesto por los profesores, utilizando para ello un mutex para compartir variables entre el mismo nodo mediante send y receive, y un proceso para redireccionar los mensajes de fuera a los procesos y sacar por pantalla los mensajes. Este programa se ha probado con diferente número de nodos, en diferentes ordenadores, y con diferentes mensajes.