**Chat distribuido peer-to-peer**

**Sergio Herrero Barco 698521**

**Alex Oarga Hategan 718123**

**1 de noviembre de 2017**

# Introducción

Un chat distribuido peer-to-peer consiste en un grupo de usuarios que se comunican entre ellos sin ningún proceso coordinador. Para ello cada usuario va a enviar su mensaje a todos los demás. Un requisito de todo chat distribuido es que los mensajes lleguen en el orden correcto. Para eso el envío de mensaje se va a realizar en exclusión mutua, usando el algoritmo de *Ricart-Agrawala*. Así un usuario que quiera enviar un mensaje, solicita entrar en la Sección Critica, y cuando haya recibido las confirmaciones de todos los demás, podrá enviar su mensaje.

# Sección Principal

Se ha creado un fichero en lenguaje *Elixir* en el que esta implementado el chat distribuido.

## Arquitectura del Sistema

El sistema se compone de una serie de nodos que van a actuar siguiendo una Arquitectura Peer-to-Peer. Dicha arquitectura consiste en que cada nodo de la red actúa simultáneamente como cliente y como servidor, es decir, en este caso cada nodo se encarga de enviar y recibir sus mensajes.

1. Un proceso llamado *pantalla* que se va a encargar solamente de recibir los mensajes que han enviado por el chat, y mostrarlos por pantalla.
2. Un proceso llamado *gestorVariables* que se va a encargar de almacenar, entregar y/o modificar las variables del sistema, sea en Exclusion Mutua o no.
3. Un proceso llamado *receptorRequest* que se va a encargar de gestionar los request que llegan y la lista de los diferidos.
4. Un proceso llamado *mutex\_exclusion* que se va a encargar de leer por pantalla el mensaje que se quiere enviar, pedir el acceso a la sección critica, enviar el mensaje a todos los usuarios, y la liberación de la Seccion critica.

## Implementacion de Ricart-Agrawala

En el algoritmo original de *Ricart-Agrawala* se usa el paradigma de memoria compartida y de semáforos. En *Elixir* se usa el paradigma del paso de mensajes, por lo que se ha tenido que hacer una ligera modificación del algoritmo general. Para ello, se ha creado un proceso llamado *GestorVariables* que va a almacenar todas las variables compartidas y gestionara que solo haya un proceso modificándolas (patrón Mutex). Como no siempre que un proceso pide variables lo quiere hacer en exclusión mutua(véase el *Highest\_Sequence\_Number*), esta función también va a poder modificar y entregar variables.

Toda la serie de acciones que se deben de hacer para solicitar la entrada a la Sección Critica, llamadas *Pre-Protocol* se han encapsulado en una función llamada con el mismo nombre, y que gestionara el envío de la petición a los demás usuarios, y la recepción de las confirmaciones.

Todas las acciones de liberación de la Sección Critica, llamadas *Post-Protocol se han encapsulado en una función llamada con el mismo nombre, y que gestionara el envío de las recepciones a todos aquellos usuarios que van por detrás del usuario.*

## Corrección del sistema

Inicialmente se ha probado el sistema con 3 usuarios, llamados nodo0, nodo1 y nodo2.