

# Trabajo Práctico Final 2019

Alvaro Barroso , Ignacio Seret , Joaquín Manuel

## Procesos Principales

### Responsables de la Información:

Estos procesos se encuentran en cada uno de los nodos, pero no tienen la misma información, ya que cada proceso tiene la información Local del nodo, es decir toda la información correspondiente a los usuarios que tienen al proceso **pSocket** corriendo en el nodo. Por lo tanto la información está distribuida. En caso de ser necesaria la información Global estos procesos se encargan de pedir la información Local de cada uno de los demás nodos.

- **usersManager**: Es el proceso encargado del manejo de los usuarios, tiene una lista de los nombres de los usuarios Locales del nodo donde se encuentra con el PID de sus respectivos Repliers. Este proceso es el que agrega un nuevo usuario o lo elimina.

Un ejemplo de la lista de usuarios que maneja es:

[{"Juan", < 0.107.0 >}, {"Pepe", < 0.108.0 >}, {"Maria", < 0.117.0 >}]

- **gamesManager**: Es el proceso encargado del manejo de las partidas, tiene una lista las partidas en curso de los usuarios Locales del nodo donde se encuentra con sus respectivos PIDs. Este proceso es el que crea nuevas partidas, elimina partidas, y sirve de intercomunicador entre el pComando y el proceso de la partida. La lista que maneja es muy similar a la anterior a diferencia de que los nombres son de las partidas (las partidas tienen el nombre de su creador) y los PIDs son de los procesos encargados de las partidas.

### Responsables del Balanceo de Cargas:

- **pStat**: Envía cada 1 segundo la carga del nodo en el que se encuentra a los procesos pBalance de todos los nodos que componen el servido.
- **pBalance**: Su función es mediante los datos que tiene indicar cual es el nodo con menor carga del server.

## Responsables de la Comunicación con el Usuario:

- **dispatcher:** Está escuchando hasta que alguien se quiere conectar, en ese caso acepta la solicitud y crea un proceso pSocket pasando el socket de comunicación.
- **pSocket:** Es el encargado de recibir todos los comandos enviados por el usuario y crear el proceso pComand en el nodo que tenga menos carga para que realice las operaciones necesarias para responder al comando del usuario.
- **pReplier:** Es el proceso que mediante el protocolo TCP IP se encarga de enviarle mensajes al usuario, es el proceso que maneja la comunicación de parte del servidor hacia el usuario.

## Características

En este trabajo se supuso lo siguiente:

- Los clientes se conectan aleatoriamente a los nodos que componen el servidor, es decir es igual de probable que se elija cualquier nodo del servidor para conectarse. Lo que evitará que muy pocos nodos tengan una gran cantidad de procesos **pSocket** corriendo.
- Los usuarios pueden participar de muchas partidas pero solo pueden crear una.
- Los usuarios no pueden aceptar partidas que ellos mismos crearon, es decir no pueden jugar contra ellos mismos.
- Cuando un jugador se va de la partida que creó la partida se elimina.
- Cuando un jugador se va de una partida en la que participa, la partida se reinicia y queda esperando a un jugador.
- Empieza la partida el jugador invitado.
- Una vez que se termina la partida ésta se reinicia.

## Comandos

- **CON nombre** Crea usuario e inicia sesión.  
Ejemplo: CON Juan
- **LSG** Muestra las partidas.
- **NEW** Crea un juego que llevará su nombre.

- **ACC nombre\_partida** Acepta una partida.  
Ejemplo: ACC Juan
- **OBS nombre\_partida** Empieza a observar una partida.
- **LEA nombre\_partida** Deja de observar una partida.
- **PLA nombre\_partida leftPlay** Sale de una partida.  
Ejemplo: PLA Juan leftPlay
- **PLA nombre\_partida numero\_fila numero\_columna** Indica que quiere poner su ficha en la posición (fila,columna) donde  $fila, columna \leq 3$ .  
Ejemplo: PLA Juan 1 1
- **BYE** Se desconecta del server, cerrando la partida de la que sea dueño, y saliendo de las que participa.

## Iniciando Múltiples Nodos:

El script `create_nodes.sh` el cual se ejecuta de la siguiente manera:

```
./create_nodes.sh cantidad_de_nodos puerto_inicial
```

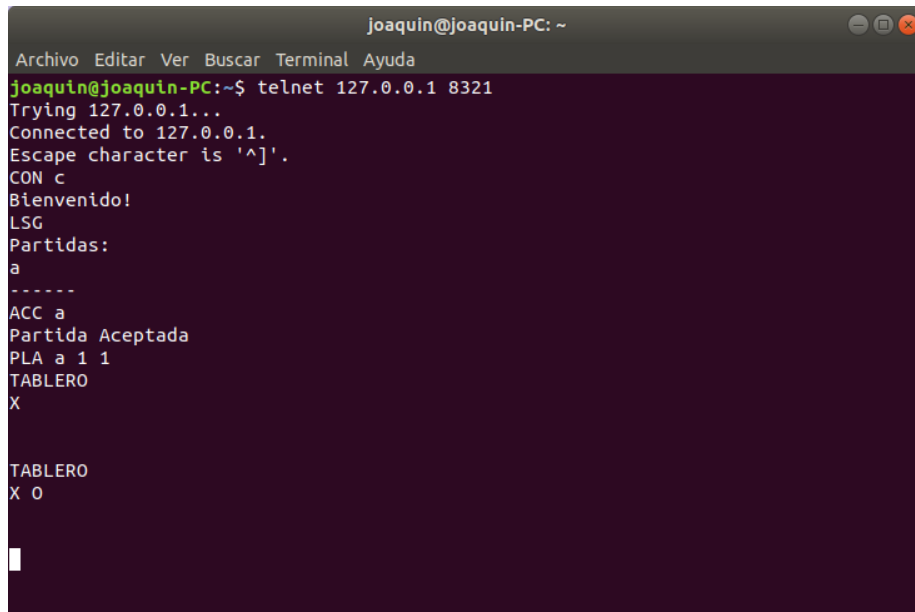
Permite crear un servidor con N nodos, los cuales estarán escuchando desde el puerto P hasta el puerto P + N - 1 (donde P es el puerto que se pasa como parametro). Antes de que los clientes se conecten a los nodos se debe esperar a que terminen de conectarse entre ellos.

**Nota:** Es necesario compilar el archivo "server.erl" (c(server).) antes de ejecutar el script.

## Cliente:

Se puede simular un cliente simplemente intercambiando mensajes luego de conectarse a algún nodo mediante:

```
telnet 127.0.0.1 puerto_de_algun_nodo
```



```
joaquin@joaquin-PC: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
joaquin@joaquin-PC:~$ telnet 127.0.0.1 8321  
Trying 127.0.0.1...  
Connected to 127.0.0.1.  
Escape character is '^]'.  
CON c  
Bienvenido!  
LSG  
Partidas:  
a  
-----  
ACC a  
Partida Aceptada  
PLA a 1 1  
TABLERO  
X  
  
TABLERO  
X 0  
  
|
```

Tambien se puede correr un cliente "Interactivo" el cual se encuentra en *client.py* y se ejecuta:

```
python3 client.py puerto tam_ventana &
```

Sino se le pone el tamaño, tomara el tamaño por defecto (500).

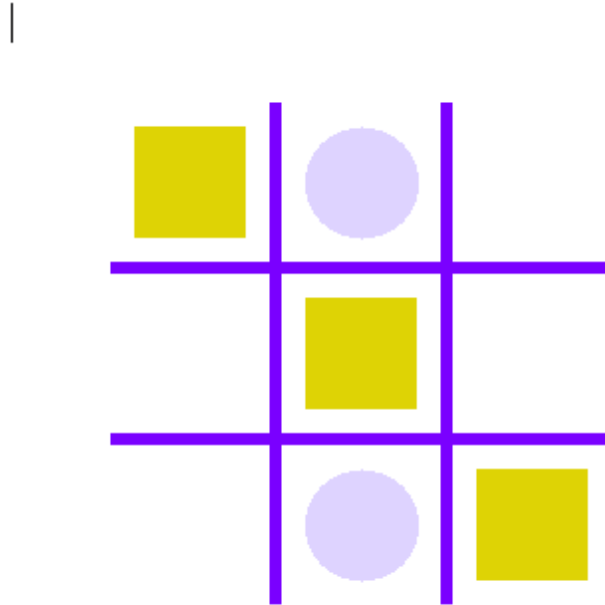
**Nota:** Es necesario instalar "pygames" para python3.

### Como usarlo:

Este cliente "Interactivo" permite al igual que telnet enviarle mensajes mediante TCP, pero aqui los comandos deben terminar con un punto ".".

Ejemplo: **"CON Pepe."**

También tiene un tablero en el que se puede observar la partida que se esté jugando, y en el caso que se cree una partida o se acepte una partida se puede hacer click sobre el tablero para jugar:



Cliente Interactivo

Es conveniente ejecutar el creador de nodos en una terminal separada a la terminal en la que se crean los clientes, para poder observar los mensajes que los nodos van emitiendo.

En el cliente interactivo si se quiere cambiar de partida en la se que está jugando se deberá realizar la primer jugada por comando.