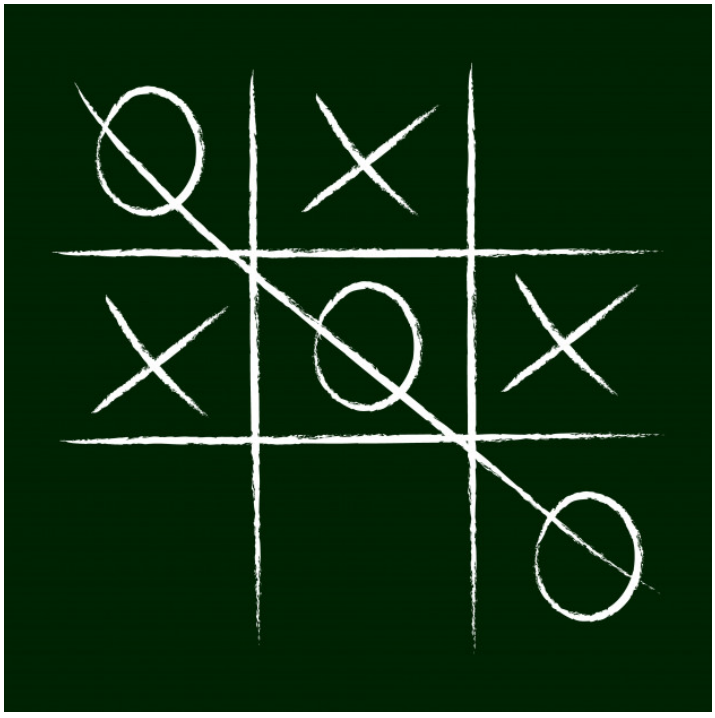
# Trabajo Final

Sistemas operativos I
LCC
UNR

Franco Sansone

Servidor de juegos distribuido



Se implementó un servidor de juegos TA-TE-TI distribuido, donde los servidores colaboran entre sí para llevar adelante las partidas y se comunican con los clientes mediante strings por sockets TCP.

El código fuente:
https://github.com/francosansone/GameServer

https://svn.dcc.fceia.unr.edu.ar/svn/lcc/R-322/Alumnos/2016/Sansone-Raimundo/tags/trabajo_final

* Algunos detalles de implementación:

Los servers que estén conrriendo ejecutan el mismo código. Dependiendo el puerto con el cual fueron inicializados, será el nombre del nodo. Los servers se conectan en forma transparente y se comunican usando la librería de erlang **net_kernel**. Para esto se siguió lo indicado en el capítulo 11 **Distributed Programming in Erlang** del libro **Erlang Programming** de Francesco Cesarini and Simon Thompson.
Cada server tiene un proceso *dispatcher* encargado de escuchar conexiones. Ante cada nueva conexión, lanza un nuevo proceso *psocket* encargado de escuchar las requests del nuevo cliente.
Cada server tiene un proceso *pbalance* y un proceso *pstat*. El segundo le envía en intervalos de 5 segundos, la carga del nodo. Al llegar una nueva petición de un cliente, pbalance chequeará cual es el nodo menos cargado y creada el proceso *pcommand* (encargado de las operaciones correspondientes) en el nodo menos cargado.

Para las conexiones TCP se siguió lo recomendado en el Capítulo 15 del **Erlang Programming in Erlang.** Para medir la carga de los nodos se usó la función erlang:statistics, presente en el libro **Joe Armstrong, Robert Virding, Claes Wikström y Mike Williams, Concurrent Programming in Erlang.** Los juegos se almacenan en un record llamado gameState, y los administra un proceso llamado game_list. Esto podría extenderse, guardando el estado de los juegos en forma persistente (por ej: un archivo de texto, una base de datos) y permitir recuperar el estado de los mismos ante la caída de alguno de los servers.

Las funciones que corren durante toda la vida del server fueron implementadas usando la técnica *Tail Recursive Function* (TLF) para evitar que el uso de memoria crea ante cada nueva llamada recursiva.

El código de los servers está configurada para correr dos servers, uno en el puerto 8000 y el otro en el puerto 8001. Podrían correrse más o menos servers modificando las funciones *node_names* y *my_node_name*.

Para correr los mismos debe garantizarse que esté corriendo *epmd.* Puede ejecutarse con el comando:

```
$ erl -pa server -eval "server:server(Port)"
```

En dos consolas corriendo ese comando con los puertos 8000 y 8001 harían funcionar el juego con la configuración actual.

Si se desea compilar, puede hacerse con el comando

```
$ erlc server.erl
```

O ejecutando `c(server)` desde la VM de Erlang.

Luego, para jugar, desde la VM de Erlang:

```
$ c(client).
$ client:init(Port)
```

Port debe ser el puerto de algunos de los servers disponibles.

Luego, para jugar los comandos son los siguientes:

- CON nombre
- LSG nombre
- NEW nombre
- ACC nombre game_id
- PLA nombre game_id jugada
- OBS nombre game_id
- LEA nombre game_id
- BYE nombre

Los comandos cumplen las mismas operaciones que las indicadas en la descripción del trabajo práctico.

Las actualizaciones de juego llegan con la forma
- UDP game_string

Pero esto es transparente para el usuario.

Un usuario puede jugar y observar diferentes juegos simultáneamente.

Finalmente, para realizar jugadas se debe tener en cuenta lo siguiente:

Durante las primeras 3 jugadas, el comando será de la forma:

```
PLA name game_id <Nro fila>,<Nro col>
```

Luego, para las jugadas siguientes, donde cambiaremos fichas de lugar:

```
PLA name game_id <Nro fila>,<Nro col>-><Nro fila>,<Nro col>
```

Por ejemplo, una secuencia podría ser:

```
4> client:init(8000).
Enter command: CON franco
sending CON franco
Enter command: User name updated franco
Enter command:NEW franco
sending NEW franco
OK NEW server1_0
Enter command:
********************
Room: server1_0
Player 1: franco
Player 2: lucas
Watchers: <>
Turn: franco
  |   |
---------------
  |   |
---------------
  |   |
********************
Enter command: PLA franco server1_0 2,2
sending PLA franco server1_0 2,2

********************
Room: server1_0
Player 1: franco
Player 2: lucas
Watchers: <>
Turn: lucas
  |   |
---------------
  | O |
---------------
  |   |
********************
```

En esa secuencia, se conectó un usuario con nombre franco y creó un juego. Un usuario lucas lo aceptó, y el usuario franco jugó la primer jugada.