Trabajo Práctico Final

Sistemas Operativos I

Dzikiewicz, Luis Enrique
Legajo: D-3850/4
luisdzi.87@gmail.com

Ernandorena, Iván
Legajo: E-1115/1
ivan.ernandorena@gmail.com

Güella, Julio
Legajo: G-5061/1
julioguella@hotmail.com

Fecha de entrega: -2017

Docentes

Guido Macchi
Guillermo Grinblat
José Luis Díaz

Diseño del trabajo práctico

El trabajo práctico consta de dos archivos que permiten su funcionamiento:

- `server.erl`: En él se encuentra todo lo relacionado a los servidores. Se tratan las conexiones de los usuarios, creacion de juegos, jugadas, y demás comandos.
- `game.erl`: Se encarga de la interfaz y de la parte visual relacionada a el juego de TA-TE-TI, por ejemplo, se encuentran las funciones encargadas de imprimir la ayuda, el tablero, chequear cuando un jugador gana, etc.

Conceptos de diseño

- Al iniciar un servidor, se crea un nuevo proceso donde se llevara una lista con los nombres de los clientes de todo el sistema distribuido llamado `list_of_client`, que se registra globalmente con el nombre de `clients_pid` para poder acceder a este proceso desde cualquier nodo. Se realiza lo mismo, para un proceso registrado globalmente como `games_pid` para llevar los nombres de las partidas en curso. Además tanto los clientes como las partidas en curso han sido registradas globalmente. Tambien, se da inicio y registro local a los procesos de `pbalance` y `pstat`. Luego de iniciar estos procesos claves para el almacenamiento de informacion y funcionamiento del sistema se pasa a la funcion `dispatcher` con el Socket donde el nodo recibe conexiones entrantes.
- Con respecto al balanceo, la función `pstat` se encarga de enviarle a la funcion `pbalance` de todos los nodos el estado de carga de él mismo. Luego, `pbalance` recibe el estado de carga de todos los nodos y se queda con el nodo de menor carga, asi cuando un proceso desea saber que nodo es el de menor carga, `pbalance` le envia este nodo.
- Un juego consta de sus jugadores, observadores, el tablero de juego, y de quien es el turno actual. Los jugadores dentro de un juego estan almacenados de la forma {`N,Jugador`}, donde `N` es un 1 o un 2 correspondiente al turno y `Jugador` es el nombre registrado globalmente (un atomo). Por diseño, el usuario que crea el juego con el comando `NEW` va a ser el jugador numero 1 y esté obtendra el primer turno.
- Cuando se realiza un cambio en un juego, el servidor envia automaticamente a cada jugador y observador de ese juego el nuevo tablero indicando quien fue el jugador que realizó esa jugada. Se obvio el comando `UPD` por esto, ya que esta alternativa parece mas clara y eficiente para los usuarios.

Inicializar el sistema

- Antes de inicializar la consola de Erlang, ejecutamos el comando `$epmd -daemon` por si el comando `$erl` arranca automaticamente el epmd (Erlang Port Mapper Daemonepmd). Luego abrimos la terminal de Erlang con `$erl` y compilamos el servidor con `1>c(server).` El servidor se arranca con la funcion `start\2` del modulo `server`. Está funcion toma como parametros un nombre para el nodo y un puerto.
- Si se quiere conectar otro servidor que este en otra PC al sistema distribuido se debe: Conectar al server TCP mediante `gen_tcp:connect(Address,Port,[Options])`, donde `Address` es la direccion IP donde se encuentra el otro servidor, `Port` es el puerto con el que se inicio el servidor y `[Options]` puede ser la lista vacia. Si se requiere algun tipo especial de conexion ver las opciones aqui: `http://erlang.org/doc/man/gen_tcp.html#connect-3`.

  Luego, se deben sincronizar ambos nodos (al sincrnoziar un nodo con otro que ya esta sincronizado con otros, automaticamente se sincroniza con todos los demas) con `net_kernel:connect_node(Node)`, donde `Node` es el nombre que se le dio al nodo. Si se quiere saber el nombre de un nodo se puede utilizar la funcion `node()`. Además, luego de la sincronizacion, se puede saber con que nodos estoy conectado con `nodes()`.
- Si tenemos dos servidores en la misma PC, basta conectar ambos servidores con la funcion `net_kernel:connect_node(Node)`.