PROGRAMACION CONCURRENTE (Curso 2013/2014)

PRÁCTICA 6
FECHA REALIZACIÓN: semanas 12 y 19 de mayo

Objetivos de la práctica:

1. Programación distribuida y concurrente
2. Socket y ServerSocket en Java

Implementa una aplicación capaz de gestionar y llevar a cabo el intercambio de ficheros entre máquinas remotas. Se trata de un híbrido entre arquitectura cliente-servidor y peer-to-peer, es decir, el intercambio de ficheros se realiza directamente entre los propios clientes, y el servidor únicamente actúa proporcionando información acerca de qué ficheros hay disponibles en el sistema y quiénes son los propietarios.

Cliente Funcionamiento básico del programa cliente:

- Al iniciar la aplicación se pregunta al usuario por su nombre de usuario y contraseña.
- Una vez iniciada la sesión, el cliente puede realizar diversas consultas como por ejemplo: búsqueda de ficheros que coincidan con una cadena determinada.
- Una vez el usuario elija el fichero a descargar, comenzará el proceso de descarga (en realidad se descarga directamente de la máquina del usuario propietario) de tal forma que el programa cliente siga su curso natural, y en particular permitiendo que se realicen otras consultas e incluso otras descargas mientras continua la descarga del primer fichero (concurrencia).
- Al margen de la voluntad del usuario, el programa cliente puede actuar como emisor de cualquiera de sus archivos compartidos, como propietario de un fichero que otro cliente solicite. Esta acción será llevada a cabo en un segundo plano permitiendo al usuario continuar con el uso normal de la aplicación.
- Al terminar la aplicación se deberá comunicar el fin de sesión al servidor, y permitir así que éste actualice apropiadamente su base de datos.

Servidor Acciones básicas de la aplicación servidor:

- Al iniciarse, leerá de un fichero “users.txt” la información de los usuarios registrados en el sistema, sus contraseñas y todos aquellos datos relativos a éstos que se consideren oportunos.
- El servidor atiende de forma concurrente todas las peticiones que realizan todos los clientes conectados al sistema, en particular:
  - Solicitud de búsqueda: El servidor realiza una búsqueda en su base de datos y devuelve los resultados obtenidos.
  - Solicitud de descarga de un fichero: El servidor se comunica con los dos clientes en cuestión, gestionando el inicio de la comunicación p2p entre ellos (de acuerdo al protocolo que definiremos). Una vez los clientes establecen conexión el servidor se desentiende de la comunicación p2p.
  - Fin de sesión: Se actualiza apropiadamente la bases de datos (en realidad son estructuras de datos).

Detalles de implementación Se deberán implementar las siguientes clases:

- *Clase Cliente*: Clase principal de la aplicación cliente. Tendrá al menos los siguientes atributos: nombre de usuario, dirección ip de la máquina. Por comodidad, conviene tener también como atributos a los objetos que proporcionan la comunicación con el servidor (socket y flujos). Es responsable de llevar a cabo todas las comunicaciones con el servidor, y cuando sea necesario ejecutar el envío o recepción de ficheros. Además ofrece el soporte para la interacción con el usuario del sistema.

- *Clase OyenteServidor:* Implementa el interfaz "Runnable" o hereda de la clase "Thread", y será usada para llevar a cabo una escucha continua en el canal de comunicación con el servidor, en un hilo diferente, proporcionándose así concurrencia.

- *Clase Usuario*: Guarda información para un usuario registrado en el sistema. Tendrá al menos los siguientes atributos: identificador de usuario, dirección ip y lista de ficheros compartidos. El servidor tendrá una lista (tabla) con todos los usuarios registrados en el sistema (instancias de la clase Usuario).

- *Clase Mensaje (abstracta)*: Sirve como raíz de la jerarquía de mensajes que deberemos diseñar. Tiene como atributos al tipo, origen y destino del mensaje en cuestión; y declara al menos los siguientes métodos:

  ```
  public int getTipo();

  public String getOrigen();

  public String getDestino();
  ```

- *Clase Servidor*: Clase principal de la aplicación servidor. Tendrá como atributo principal a una lista de usuarios (instancias de la clase Usuario). El servidor espera la llegada de peticiones de inicio de sesión, y asocia un hilo de ejecución con cada usuario.

- *Clase OyenteCliente*: Implementa el interfaz "Runnable" o hereda de la clase "Thread", y es usada para proporcionar concurrencia respecto a las sesiones de casa usuario con el servidor. El método "run()" se limita a hacer lecturas del flujo de entrada correspondiente, realizar las acciones oportunas, y devolver los resultados en forma de mensajes que serán enviados al usuario o usuarios involucrados.

Será necesario implementar además varias clases adicionales. En particular: clases para cada tipo de mensaje, clases para soporte de concurrencia en el cliente, y clases para la interfaz con el usuario (GUI o de texto).