Sistemas Distribuidos

Grado en Ingeniería Informática

Práctica (parte 1)

Diseño e implementación de un sistema peer-to-peer

Félix García Carballeira
Francisco Javier García Blas

3 de marzo de 2020

Índice

1. Objetivo

El objetivo global de toda la práctica es desarrollar un sistema peer-to-peer de distribución de ficheros entre clientes. La práctica completa se hará de forma incremental a través de tres partes.

El objetivo de esta primera parte de la práctica es que el alumno llegue a conocer los principales conceptos relacionados con la comunicación de procesos usando sockets TCP. Para ello se propone desarrollar una aplicación de distribución de ficheros entre clientes, como se describe en las siguientes secciones.

2. Descripción de la funcionalidad de la primera parte

Se trata de desarrollar un servicio peer-to-peer como el que se muestra en la Figura 1. La aplicación a desarrollar se compone de dos programas independientes:

- *Cliente*. Se trata de un programa que ejecutan los clientes del sistema. Este programa permite publicar contenido y proporcionar ficheros a otros usuarios.
- *Servidor*. Es el programa servidor que coordina el funcionamiento de la aplicación.

Los usuarios puede registrarse, darse de baja, conectarse y desconectarse del servicio (estos serán mensajes enviados desde los clientes al servidor). Cada usuario conectado podrá publicar nombres de ficheros en el servidor. También podrá eliminar estos nombres del servidor. El servidor mantendrá para cada cliente del sistema (conectado o no) la lista con los nombres de fichero que ha hecho públicos. Un usuario conectado podrá pedirle al servidor la lista de usuarios del sistema y podrá pedirle la lista de ficheros publicados por un determinado usuario. Cuando un usuario desee descargarse un fichero almacenado en otro cliente, directamente se lo pedirá a ese cliente. Es decir, el servidor no almacena los ficheros que comparten los usuarios. Los ficheros los tiene cada usuario en su máquina y los puede enviar a otros clientes que se lo soliciten.

3. Primera parte

El alumno deberá diseñar, codificar y probar, utilizando el lenguaje C y sobre un sistema operativo UNIX/Linux, un servidor que gestione la funcionalidad del sistema y, por otro lado, deberá diseñar, codificar y probar, utilizando el lenguaje Java y sobre un sistema operativo UNIX/Linux, el código de los clientes.

A continuación se detallan las características del sistema. En esta parte de la memoria se va a describir el protocolo a seguir entre el servidor y el cliente. Este protocolo permitirá a cualquier cliente que lo siga comunicarse con el servidor implementado. Esto hace que, diferentes alumnos puedan probar sus clientes con los servidores desarrollados por otros.

Para el almacenamiento de los usuarios y de los nombres de ficheros publicados por cada usuario, en esta primera parte se podrán utilizar listas en memoria como las usadas en el primer ejercicio evaluable. También podrá emplear almacenamiento en disco. Tenga en cuenta que, posteriormente, esta información habrá de almacenarse de forma persistente utilizando servicios que emplean RPC (llamadas a procedimientos remotos). Cada grupo de prácticas podrá elegir el diseño que desee para almacenar esta información.

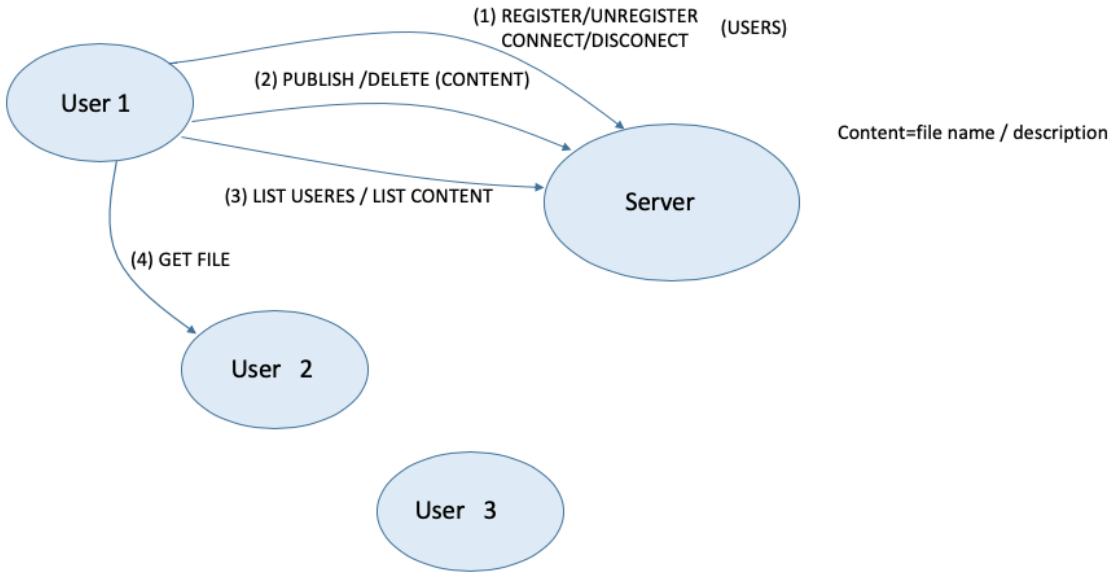

Figura 1: Modelo de aplicación

3.1. Desarrollo del servicio

El objetivo es diseñar y desarrollar los dos siguientes programas:

- Un **servidor concurrente multihilo** (escrito en C) que proporciona la funcionalidad del servidor.
- Un **cliente multihilo** (escrito en Java) que ofrece la funcionalidad que ejecutan los usuarios del sistema peer-to-peer.

4. Desarrollo y protocolo del cliente

El cliente ejecutará un intérprete de mandatos para comunicarse con el servidor siguiendo el protocolo que se describe en las siguientes secciones.

4.1. Uso del cliente

Este intérprete se proporciona como material de apoyo. Descomprima el archivo `ssdd_pract_material`

```
$tar xvf ssdd_pract_material.tar
```

Dentro de este directorio se encuentra el archivo *cliente.java* y todo el código necesario para su compilación. Para compilar este archivo y obtener su clase ejecutable ejecute:

```
$ javac   cliente.java
```

A continuación, puede ejecutar el programa cliente de la siguiente manera:

```
$ java -cp  .  client  -s <server> -p <port>
```

Donde **server** y **port** representan la dirección IP y el puerto donde ejecuta el servidor. El nombre **server** puede ser tanto el nombre (dominio-punto) como la dirección IP (decimal-punto) del servidor. El intérprete de mandatos mostrará:

```
c>
```

y estará a la espera de que el usuario introduzca mandatos. Para finalizar este intérprete, el cliente debe escribir **QUIT**.

```
c> QUIT
```

A lo largo de la ejecución del cliente pueden obtenerse errores debidos a la red (servidor caído, fallo en la red, etc), todos estos errores hay que tenerlos en cuenta. Por tanto, se recomienda hacer un correcto tratamiento de los errores devueltos por las llamadas al sistema y de la biblioteca de sockets utilizada.

El cliente proporcionado incluye todo el código para leer los mandatos introducidos por el usuario y que se describen a continuación.

Si un usuario está conectado y se introduce QUIT, el cliente deberá desconectarse previamente del sistema.

4.2. Registro de un cliente en el sistema

Cuando un cliente quiera registrarse como usuario del servicio deberá ejecutar el siguiente mandato en la consola del cliente:

```
c> REGISTER <userName>
```

Cuando un cliente quiere registrarse en el sistema realizará las siguientes operaciones:

1. Se conecta al servidor, de acuerdo a la IP y puerto pasado en la línea de mandatos al programa cliente.
2. Se envía la cadena 'REGISTER" indicando la operación. La cadena de caracteres finaliza con el código ASCII `'\0'`.
3. Se envía una cadena de caracteres con el nombre del usuario que se quiere registrar y que identifica al usuario.
4. Recibe del servidor un byte que codifica el resultado de la operación: 0 en caso de éxito, 1 si el usuario ya está registrado previamente, 2 en cualquier otro caso.
5. Cierra la conexión.

Recuerde que todas las cadenas de caracteres que se envíen finalizan con el código ASCII `'\0'`. El tamaño máximo de un nombre de usuario es de 256 bytes.

Este servicio una vez ejecutado en el servidor, puede devolver tres resultados (0,1 o 2). Si la operación se realiza correctamente, el cliente recibirá por parte del servidor un mensaje con código 0 y mostrará por consola el siguiente mensaje:

```
c> REGISTER OK
```

Si el usuario ya está registrado se mostrará:

```
c> USERNAME IN USE
```

En este caso el servidor no realizará ningún registro.

En caso de que no se pueda realizar la operación de registro, bien porque el servidor este caído o bien porque se devuelva el código 2 por cualquier otro error, se mostrará en la consola el siguiente mensaje:

```
c> REGISTER FAIL
```

4.3. Darse de baja en el sistema

Cuando un cliente quiere darse de baja del servicio deberá ejecutar el siguiente mandato en la consola:

```
c> UNREGISTER <userName>
```

Para ello realizará las siguientes operaciones:

1. Se conecta al servidor, de acuerdo a la IP y puerto pasado en la línea de mandatos al programa.
2. Se envía la cadena "UNREGISTER" indicando la operación.
3. Se envía a continuación una cadena con el nombre del usuario que se quiere dar de baja.
4. Recibe del servidor un byte que codifica el resultado de la operación: 0 en caso de éxito, 1 si el usuario no existe, 2 en cualquier otro caso.
5. Cierra la conexión.

Este servicio una vez ejecutado en el servidor, puede devolver tres resultados (0, 1, 2). Si la operación de baja se realiza correctamente y el usuario es borrado en el servidor, el servidor devolverá un 0 y el cliente mostrará por la consola:

```
c> UNREGISTER OK
```

Si el usuario que se quiere dar de baja del servicio no existe, se mostrará el siguiente mensaje por la consola del cliente:

```
c> USER DOES NOT EXIST
```

En caso de que no se pueda realizar esta operación, bien porque el servidor este caído y no se pueda establecer la conexión con él o bien porque devuelva el código 2, se mostrará en la consola el siguiente mensaje:

```
c> UNREGISTER FAIL
```

4.4. Conectarse al sistema

Una vez que un cliente está registrado en el sistema, este puede conectarse y desconectarse del servicio tantas veces como desea. Para conectarse debe enviar (utilizando el protocolo que se describe más adelante) al servidor su dirección IP y puerto. La estructura de un proceso cliente conectado al servicio se muestra en la figura 2.

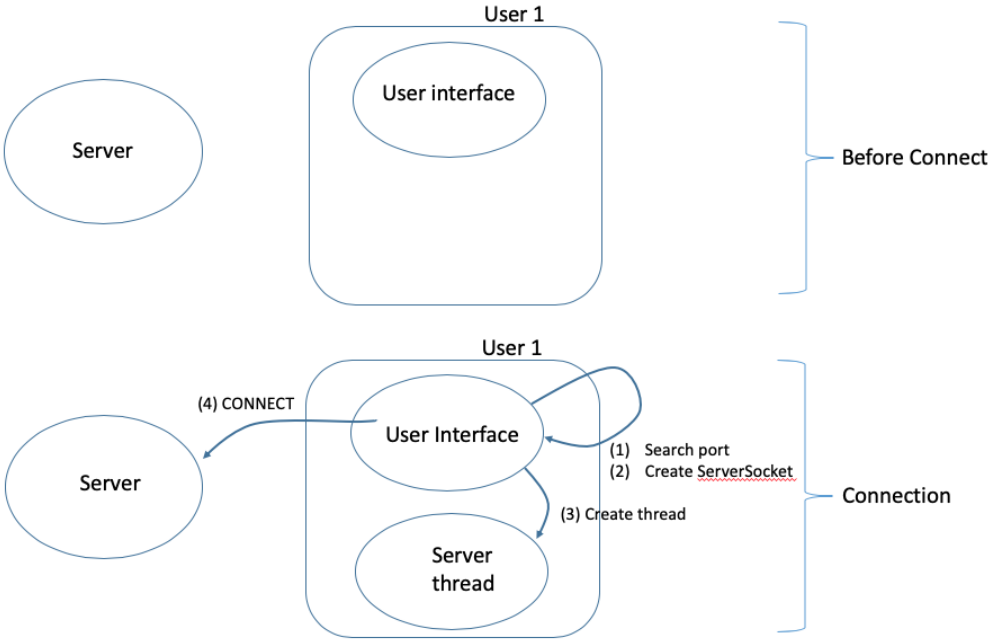

Figura 2: Estructura de un proceso cliente conectado al servicio

Para ello el cliente introducirá por consola:

```
c> CONNECT <userName>
```

Internamente el cliente buscará un puerto válido libre (1). Una vez obtenido el puerto, y antes de enviar el mensaje al servidor, el cliente debe crear un ServerSocket (2) y un hilo (e) que será el encargado de escuchar (en la IP y puerto seleccionado) y atender las peticiones de descarga de ficheros de otros usuarios. A continuación el cliente enviará (4) la solicitud de conexión con la siguiente información:

1. Se conecta al servidor, de acuerdo a la IP y puerto pasado en la línea de mandatos al programa.
2. Se envía la cadena 'CONNECT" indicando la operación.
3. Se envía una cadena con el nombre del usuario.
4. Se envía una cadena de caracteres que codifica el número de puerto de escucha del cliente. Así, para el puerto 456, esta cadena será "456".
5. Recibe del servidor un byte que codifica el resultado de la operación: 0 en caso de éxito, 1 si el usuario no existe, 2 si el usuario ya está conectado y 3 en cualquier otro caso.

6. Cierra la conexión.

Una vez establecida la conexión en el sistema, el servidor devolverá un byte que codificará el resultado de la operación: 0 en caso de éxito, 1 si el usuario no existe, 2 si el usuario ya está conectado y 3 en cualquier otro caso.

Si todo ha ido bien, se mostrará por consola:

```
c> CONNECT  OK
```

En caso de código 1 (usuario no está registrado en el sistema), el cliente mostrará el siguiente error por consola:

```
c> CONNECT FAIL, USER DOES NOT EXIST
```

En caso de que el cliente ya estuviera conectado en el sistema (código 2), el cliente mostrará por la consola:

```
c> USER ALREADY  CONNECTED
```

En caso de que no se pueda realizar la operación de conexión, bien porque el servidor este caído, se produzca un error en las comunicaciones o se devuelva el código 3, se mostrará en la consola el siguiente mensaje:

```
c> CONNECT FAIL
```

4.4.1. Publicación de contenidos

Cada cliente conectado podrá publicar contenidos en el servidor. Esta publicación consistirá en enviar un nombre de fichero y una breve descripción asociada al fichero. Solo se publica el nombre del fichero, no su contenido. El fichero siempre se almacena en el cliente y la transferencia de ficheros se realiza entre clientes. Para publicar un contenido el cliente introducirá por consola:

```
c> PUBLISH  <file name>  <description>
```

Donde `<file name>` representa una cadena con el nombre del fichero y `<description>` la descripción asociada. El cliente realizará con el servidor las siguientes operaciones:

1. Se conecta al servidor, de acuerdo a la IP y puerto pasado en la línea de mandatos al programa.
2. Se envía la cadena "PUBLISH" indicando la operación.
3. Se envía una cadena con el nombre del fichero (esta cadena no podrá contener espacios en blanco). El tamaño máximo del nombre del fichero será de 256 bytes.
4. Se envía una cadena de caracteres con la descripción del contenido. Esta cadena podrá contener espacios en blanco y su tamaño máximo será de 256 bytes.
5. Recibe del servidor un byte que codifica el resultado de la operación: 0 en caso de éxito, 1 si el usuario no existe, 2 si el usuario no está conectado, 3 el fichero ya está publicado, y 4 en cualquier otro caso.
6. Cierra la conexión.

Una vez enviado un mensaje para publicación, se mostrará por consola:
Si todo ha ido bien y el código devuelto por el servidor es 0:

```
c> PUBLISH  OK
```

En caso de código 1 (usuario no está registrado en el sistema), el cliente mostrará el siguiente error por consola:

```
c> PUBLISH FAIL, USER DOES NOT EXIST
```

En caso de que el cliente NO estuviera conectado en el sistema (código 2), el cliente mostrará por la consola:

```
c> PUBLISH FAIL, USER  NOT CONNECTED
```

En caso de que el nombre de fichero ya haya sido publicado por el cliente, el cliente mostrará por la consola:

```
c> PUBLISH FAIL, CONTENT ALREDAY PUBLISHED
```

En caso de que no se pueda realizar la operación de conexión, bien porque el servidor este caído, se produzca un error en las comunicaciones o se devuelva el código 4, se mostrará en la consola el siguiente mensaje:

```
c> PUBLISH FAIL
```

4.4.2. Borrado de contenidos

Cada cliente conectado podrá eliminar contenidos publicados en el servidor. El cliente enviará el nombre de fichero a eliminar del sistema. Para borrar un contenido el cliente introducirá por consola:

```
c> DELETE  <file name>
```

A continuación realizará las siguientes operaciones:

1. Se conecta al servidor, de acuerdo a la IP y puerto pasado en la línea de mandatos al programa.
2. Se envía la cadena "DETELE" indicando la operación.
3. Se envía una cadena con el nombre del usuario que realiza la operación de borrado.
4. Se envía una cadena con el nombre del fichero (esta cadena no podrá contener espacios en blanco). El tamaño máximo del nombre del fichero será de 256 bytes.
5. Recibe del servidor un byte que codifica el resultado de la operación: 0 en caso de éxito, 1 si el usuario no existe, 2 si el usuario no está conectado, 3 el fichero no ha sido publicado previamente, y 4 en cualquier otro caso.
6. Cierra la conexión.

Una vez enviado un mensaje de eliminación, si todo ha ido bien, se mostrará por consola:

```
c> DELETE  OK
```

En caso de código 1 (usuario no está registrado en el sistema), el cliente mostrará el siguiente error por consola:

```
c> DELETE FAIL, USER DOES NOT EXIST
```

En caso de que el cliente NO estuviera conectado en el sistema (código 2), el cliente mostrará por la consola:

```
c> DELETE FAIL, USER  NOT CONNECTED
```

En caso de que el nombre de fichero NO haya sido publicado previamente por el cliente, el cliente mostrará por la consola:

```
c> DELETE FAIL, CONTENT NOT PUBLISHED
```

En caso de que no se pueda realizar la operación de conexión, bien porque el servidor este caído, se produzca un error en las comunicaciones o se devuelva el código 4, se mostrará en la consola el siguiente mensaje:

```
c> DELETE FAIL
```

4.4.3. Listado de usuarios

Un cliente conectado en el sistema podrá conocer todos los usuarios conectados en el sistema. El cliente enviará una solicitud para conocer los nombres de los usuarios conectados. El servidor devolverá la lista de usuarios conectados con su IP y puerto correspondiente. Solo se enviará información de los clientes que estén conectados en el sistema. Para listar los usuarios el cliente introducirá por consola:

```
c> LIST_USERS
```

A continuación el cliente realizará las siguientes operaciones:

1. Se conecta al servidor, de acuerdo a la IP y puerto pasado en la línea de mandatos al programa.
2. Se envía la cadena "LIST_USERS" indicando la operación.
3. Se envía una cadena con el nombre del usuario que realiza la operación.
4. Recibe del servidor un byte que codifica el resultado de la operación: 0 en caso de éxito, 1 si el usuario no existe, 2 el usuario no está conectado y 3 en cualquier otro caso.
5. En caso de que el código sea 0 el servidor enviará la siguiente información de vuelta al cliente:
   a) Una cadena que codifica el número de usuarios cuya información se va a enviar. Si se recibe la cadena "5", el servidor a continuación enviará la información asociada a 5 clientes. Por cada cliente enviará 3 cadenas de caracteres codificando el nombre del usuario, la dirección IP y el puerto.

6. Cierra la conexión.

Si todo ha ido bien (código 0), se mostrará por consola:

```
c> LIST_USERS  OK
        USER1  IP1  PORT1
        USER2  IP2  PORT2
        ....
```

indicando la lista de usuarios conectados y sus direcciones.

En caso de código 1 (usuario no está registrado en el sistema), el cliente mostrará el siguiente error por consola:

```
c> LIST_USERS FAIL, USER DOES NOT EXIST
```

En caso de que el cliente NO estuviera conectado en el sistema (código 2), el cliente mostrará por la consola:

```
c> LIST_USERS FAIL, USER  NOT CONNECTED
```

En caso de que no se pueda realizar la operación, bien porque el servidor este caído, se produzca un error en las comunicaciones o se devuelva el código 3, se mostrará en la consola el siguiente mensaje:

```
c> LIST_USERS FAIL
```

4.4.4. Listado de contenido

Un cliente conectado en el sistema podrá conocer el contenido publicado por otro usuario. El cliente enviará una solicitud al servidor para conocer los ficheros publicados por un determinado usuario. El servidor devolverá la lista ficheros publicados por ese usuario. Para listar el contenido publicado por el usuario con nombre `user_name` el cliente introducirá por consola:

```
c> LIST_CONTENT <user_name>
```

A continuación el cliente realizará las siguientes operaciones:

1. Se conecta al servidor, de acuerdo a la IP y puerto pasado en la línea de mandatos al programa.
2. Se envía la cadena "LIST_CONTENT" indicando la operación.
3. Se envía una cadena con el nombre del usuario que realiza la operación.
4. Se envía una cadena con el nombre del usuario cuyo contenido se quiere conocer (`user_name`).
5. Recibe del servidor un byte que codifica el resultado de la operación: 0 en caso de éxito, 1 si el usuario que realiza la operación no existe, 2 el usuario que realiza la operación no está conectado, 3 el usuario cuyo contenido se quiere conocer no existe y 4 en cualquier otro caso.
6. En caso de que el código sea 0 el servidor enviará la siguiente información de vuelta al cliente:

a) Una cadena que codifica el número de nombre de ficheros publicados. Si se recibe la cadena "7", el servidor a continuación enviará la información asociada a 7 nombres de ficheros. Por cada fichero el servidor enviará una cadena de caracteres con el nombre del fichero (el tamaño máximo del fichero es de 256 bytes).

7. Cierra la conexión.

Si todo ha ido bien (código 0), se mostrará por consola:

```
c> LIST_CONTENT  OK
        file_name_1
        file_name_2
        file_name_3
        ....
```

indicando la lista de usuarios conectados y sus direcciones.

En caso de código 1 (usuario no está registrado en el sistema), el cliente mostrará el siguiente error por consola:

```
c> LIST_CONTENT FAIL, USER DOES NOT EXIST
```

En caso de que el cliente NO estuviera conectado en el sistema (código 2), el cliente mostrará por la consola:

```
c> LIST_CONTENT FAIL, USER  NOT CONNECTED
```

En caso de código 3 (usuario remoto no está registrado en el sistema), el cliente mostrará el siguiente error por consola:

```
c> LIST_CONTENT FAIL, REMOTE USER DOES NOT EXIST
```

En caso de que no se pueda realizar la operación, bien porque el servidor este caído, se produzca un error en las comunicaciones o se devuelva el código 4, se mostrará en la consola el siguiente mensaje:

```
c> LIST_CONTENT FAIL
```

4.5. Desconectarse del sistema

Un cliente puede desconectarse del sistema introduciendo por consola:

```
c> DISCONNECT <userName>
```

Internamente el cliente debe parar la ejecución del hilo creado en la operación CONNECT, cerrar su puerto de escucha y destruir el hilo. El cliente realizará las siguientes operaciones:

1. Se conecta al servidor, de acuerdo a la IP y puerto pasado en la línea de mandatos al programa.

2. Se envía la cadena "DISCONNECT" indicando la operación.

3. Se envía una cadena con el nombre del usuario que se desea desconectar.

4. Recibe del servidor un byte que codifica el resultado de la operación: 0 en caso de éxito, 1 si el usuario no existe, 2 si el usuario no está conectado y 3 en cualquier otro caso.

5. Cierra la conexión.

Si todo ha ido correctamente, el servidor devolverá un 0 y el cliente mostrará el siguiente mensaje por consola:

```
c> DISCONNECT OK
```

Si el usuario no existe, se mostrará el siguiente mensaje por consola:

```
c> DISCONNECT FAIL / USER DOES NOT EXIST
```

Si el usuario existe pero no se conectó previamente, se mostrará el siguiente mensaje por consola:

```
c> DISCONNECT FAIL / USER NOT CONNECTED
```

En caso de que no se pueda realizar la operación con el servidor, porque éste esté caído, hay un error en las comunicaciones o el servidor devuelve un 3, se mostrará por la consola el siguiente mensaje:

```
c> DISCONNECT FAIL
```

En caso de que se produzca un error en la desconexión, el cliente de igual forma parará la ejecución del hilo creado en la operación CONNECT, actuando a todos los efectos como si se hubiera realizado la desconexión.

Esta operación de desconexión deberá realzarse siempre que el usuario introduzca por consola el comando QUIT.

4.6. Transferencia de un archivo

Una vez que el cliente esté registrado y conectado al sistema podrá enviar mensajes a otros usuarios registrados para descargar el contenido de un fichero. Para ello deberá ejecutar el siguiente mandato por la consola:

```
c> GET_FILE <userName> <remote_file_name> <local_file_name>
```

Para ello el cliente realizará las siguientes operaciones:

1. Se conecta al cliente, de acuerdo a la IP y puerto de escucha de dicho cliente (cuyo nombre `userName` se indica en la operación y que podrá conocer mediante la operación de listado de usuarios..

2. Se envía la cadena "GET_FILE" indicando la operación.

3. Se envía una cadena con el nombre del fichero que se desea descargar.

4. Recibe del otro cliente un byte que codifica el resultado de la operación: 0 indicando que el fichero se va a transferir, 2 si el fichero no existe en el cliente remoto y 2 en cualquier otro caso. En caso de código 0 el cliente remoto transferirá el contenido del archivo cerrando la conexión cuando se haya completado la transferencia. El cliente local a medida que reciba el contenido del fichero lo irá almacenando en el fichero local que se ha pasado en la consola.

5. Cierra la conexión.

En caso de éxito (código 0) y una vez el fichero se ha transferido correctamente se mostrará por consola el siguiente mensaje:

```
c>  GET_FILE OK
```

En caso de que el fichero no exista (código 1) y mostrará por consola:

```
c> GET_FILE FAIL / FILE NOT EXIST
```

En caso de que se produzca un error (cliente caído, no se puede realizar la conexión con el cliente, error de comunicaciones, o cualquier otro tipo de error, o se devuelva un error de tipo 2) se mostrará por consola:

```
c> GET_FILE FAIL
```

En caso de que la transferencia no se haya podido completar, el fichero local será borrado.

5. Desarrollo del servidor

El servidor se desarrollará para poder atender todas las peticiones de los clientes de acuerdo al protocolo descrito en las secciones anteriores. El servidor debe ser capaz de gestionar varias conexiones simultáneamente (debe ser concurrente) mediante el uso de múltiples hilos (multithread). El servidor utilizará sockets TCP orientados a conexión.

5.1. Uso del servidor

Se ejecutará de la siguiente manera:

```
$ ./server -p <port>
```

Al iniciar el servidor se mostrará el siguiente mensaje:

```
s> init server <local IP>:<port>
```

Antes de recibir comandos por parte de los clientes mostrará:

```
s>
```

Por cada petición realizada por un cliente mostrará por pantalla la siguiente información:

```
s> OPERATION FROM USER
```

Donde USER es el nombre del usuario que realiza la operación. El programa terminará al recibir una señal SIGINT (Ctrl+c).

6. Normas generales

Es importante analizar el código de apoyo proporcionado con la práctica ya que será el punto de partida para la realización de la misma. Se va a proporcionar el esqueleto del programa servidor y cliente. Hay que utilizar estos programas como punto de partida. Todo el tratamiento de los argumentos y del intérprete de mandatos está ya implementado.

1. Las prácticas que no compilen o no se ajusten a la funcionalidad y requisitos planteados, obtendrán una calificación de 0.
2. Las prácticas que tengan warnings o no estén comentadas serán penalizadas. Un programa no comentado, obtendrá una calificación de 0.
3. La entrega de la práctica se realizará a través de los entregadores habilitados. No se permite la entrega a través de correo electrónico.
4. Se prestará especial atención a detectar funcionalidades copiadas entre dos prácticas. En caso de detectar copia, se aplicará la normativa de la Universidad y los alumnos de los grupos involucrados en la copia tendrán un 0 en la nota final de la asignatura.
5. La práctica debe funcionar en los ordenadores de las aulas informáticas (4.0.F16, 4.0.F18, 2.2.C05 y 2.2.C06) o en guernika.
6. El sistema debe funcionar con todos los componentes del sistema ejecutando en máquinas distintas.
7. Debe seguirse el protocolo descrito.
8. La memoria debe tener una longitud máxima de 10 páginas.

7. Documentación a entregar y plazo de entrega

La práctica consta de tres partes y se va a ir realizando de forma incremental. Solo se realizará una única entrega. La fecha de entrega de la práctica será el **3 de mayo**. En la siguiente parte del enunciado ciado se indicará la documentación a entregar.