Grado en Ingeniería Informática

Departamento de Ingeniería Informática

Sistemas Distribuidos

Práctica 2: Programación con Sockets

Autores:
Borja Romero Fernández
Carmen del Mar Ruiz de Celis

Fecha:
2 de mayo de 2020

Índice

Índice de figuras

1. Programa 1: Probar sockets

En este primer programa creamos dos archivos, uno que define el comportamiento del cliente y otro que define el comportamiento del servidor en la conexión.

Desde el lado del cliente simplemente creamos el socket para la conexión y enlazamos con el puerto 1200. Después, en el caso del socket TCP, tanto cliente como servidor reciben y envían un mensaje, mientras que en el socket UDP el cliente manda un mensaje y es el servidor el que lo recibe. Después de este intercambio la conexión se cierra entre ambos.

Mientras tanto, los servidores TCP y UDP se encargan de realizar las operativas necesarias para que ambas permitan el funcionamiento correcto del paso de mensajes. Para que esto sea posible, necesitamos al inicio enlazar el socket que creamos con el puerto 1200. A parte, en TCP debemos poner el servidor a escuchar en el socket y, cuando este encuentre una nueva conexión, la acepte.

¿Que diferencias hay entre TCP y UDP?

Como ya hemos explicado, la diferencia principal entre TCP y UDP es que en TCP ponemos el servidor a escuchar en el socket para que pueda aceptar las peticiones de conexión, mientras que en UDP no.

En TCP existe un protocolo de confirmación que reconoce (*ACK, acknowledge*) al cliente y realiza un *handshake*, o "apretón de manos" en español.

¿Que pasa si el cliente se inicia antes que el servidor en TCP y UDP?

Debido a esto último que hemos explicado, en TCP cuando el cliente se lanza antes que el servidor, al no poder realizarse la conexión, se muestra un mensaje de error. En el caso de UDP, al no haber un protocolo de confirmación, la ejecución del cliente simplemente termina sin ningún tipo de error.

2. Programa 2: Chat Simple

En este ejercicio hemos creado tres archivos: un servidor, que funcionará como mensajero, y dos clientes que serán aquellos que manden y reciban los mensajes. Vamos a ver las diferencias entre los protocolos TCP y UDP:

2.1. Protocolo TCP

Haciendo uso de este protocolo, el servidor necesitará contar con dos sockets, uno por cada cliente que lancemos. El primer socket estará enlazado al puerto 1201 y el segundo al puerto 1202, para que no haya conflicto en las conexiones. Después ponemos a escuchar al servidor en ambos sockets y hacemos que acepte las conexiones cuando lleguen. A partir de este punto el servidor entra en un bucle *while* que solo termina cuando uno de los dos clientes se desconecte del chat. A lo que se dedica el servidor en este bucle es a redireccionar los mensajes entre el cliente 1 y el cliente 2. En el chat, el cliente 1 será el que hable primero y el cliente 2 responderá una vez que esto ocurra, convirtiéndolo en un chat síncrono. En caso de que uno de los dos clientes escriba "desconectado" en su chat, el servidor enviará un mensaje al otro cliente, avisando de que la otra persona se ha desconectado, y cerrará conexión con ambos clientes.

Los clientes, en este caso, realizan la conexión a sus puertos respectivos (el cliente 1 al puerto 1201 y el cliente 2 al 1202) haciendo uso de un socket. En cuanto se realiza la conexión entre cliente y servidor, también entran en un bucle while que no parará hasta que no escriban "desconectado" por teclado. En el bucle ambos clientes escriben y reciben los mensajes del otro (el cliente 1 envía el mensaje primero y luego lo recibe un mensaje, y en el cliente 2 pasa lo mismo pero del revés). Cuando uno de los dos escribe "desconectado" las conexiones se cierran desde el servidor, recibiendo un mensaje el último cliente que quede en el servidor.

2.2. Protocolo UDP

La diferencia entre el funcionamiento del sistema en TCP y UDP es que en UDP el servidor no espera a la conexión. A parte de esto, no existe mucha diferencia entre TCP y UDP.

En UDP también existe un bucle *while* en el servidor que termina cuando uno de los dos clientes escribe "desconectado" por el chat. Mientras que esto no pase el servidor redirecciona síncronamente los mensajes entre clientes.

Lo mismo pasa con los clientes, que tienen el mismo comportamiento que en TCP, enviando y recibiendo mensajes hasta que las conexiones se cierren.

¿Qué función de Python hay para leer por teclado?

La función que hemos utilizado para leer por teclado, en ambos casos, es la función *input()*, que lee el teclado hasta que encuentre un salto de línea. Esta función guarda el resultado como una cadena (*string*), que podemos enviar fácilmente entre clientes y servidor, codificándola a *UTF-8* y decodificándola allá donde queramos mostrarla.

3. Programa 3: FTP simple

Por último, en el programa 3 creamos un FTP haciendo uso de TCP. Para ello creamos un archivo para el cliente y otro para el servidor. A parte hemos creado tres archivos con extensión *.txt* para probar las descargas (*hola.txt*, *prueba1.txt* y *prueba2.txt*).

En el servidor creamos un socket enlazado con el puerto 1210 y esperamos a aceptar la conexión poniéndolo a escuchar. También guardamos en una variable el listado de ficheros que existen en el directorio del mismo archivo. Esto es posible gracias a la biblioteca *os* y la función *listdir()*. Lo siguiente que hace el servidor es mandar un mensaje informativo al cliente con las opciones que tiene, y espera a la respuesta del cliente.

Si el cliente escribe por teclado "ls", el servidor devolverá una cadena al cliente con el listado de ficheros que guardamos en la variable que creamos al principio. Tras esto manda un mensaje que pide el nombre del archivo que el cliente desea, y finalmente espera la respuesta del cliente y comprueba si el fichero existe o no. En caso de que exista, el servidor abre el archivo y copia en una cadena el contenido del fichero pedido. Esta cadena es pasada al cliente para crear un nuevo fichero. Si no existe se muestra un mensaje de error. En caso de que el cliente no introduzca "ls", el servidor realiza sólo el último paso y comprueba si el fichero existe o no.

El cliente entonces se encarga de seleccionar el fichero que desea o la opción "ls". Una vez que realiza la petición espera o a la lista de ficheros disponibles o al propio fichero pedido. Si el servidor recibe un fichero válido que se encuentre en el servidor, este le pasa una cadena con el contenido del archivo requerido. Con esta cadena, el cliente crea un nuevo archivo con el mismo nombre y extensión que el archivo que pidió y vuelca los datos de esta misma en el nuevo archivo creado, obteniendo así un archivo exactamente igual al que se encuentra en el servidor. Este archivo se guardará en la carpeta "recibido", encontrada en el directorio anterior al actual. Una vez realizado el intercambio, la conexión es cerrada por ambas partes.