Práctica 2.1: Aplicación *Echo* usando UDP

Descripción:

En esta práctica el/la estudiante tendrá que implementar el cliente y el servidor de una aplicación *Echo* utilizando el lenguaje de programación Python. Una aplicación *Echo* es una aplicación muy sencilla: El servidor tendrá que enviar de vuelta toda información que el cliente le envíe, es decir, el servidor "hace eco" de lo que el cliente envía, de ahí el nombre de la aplicación. En este caso, como protocolo de nivel de transporte usaremos el protocolo UDP.

Material:

Para hacer esta práctica se dispone de los siguientes recursos:

1. Cliente ejecutable de un ejemplo de implementación de la aplicación *Echo* (*echo_cli_udp.pyc*) que podremos ejecutar en la máquina local. Además, se estará ejecutando un servidor *Echo* en el puerto 50007 de la máquina dif-linuxserver.ehu.es (158.227.106.30). De esta manera, ejecutando el cliente compilado contra el servidor remoto, nos podremos hacer una idea de cómo debería funcionar la aplicación a desarrollar.
2. Ficheros con el código fuente del cliente y servidor parcialmente desarrollados (*echo_cli_udp.py* y *echo_ser_udp.py*, respectivamente). Éstos son los ficheros que habrá que modificar para desarrollar la práctica.

Protocolo de aplicación

Como hemos dicho esta aplicación es especialmente sencilla, más todavía en el caso de su implementación sobre UDP. Debido a la naturaleza de la propia aplicación, ni siquiera es necesario definir comandos, ni respuestas. El procedimiento general de la aplicación se basa en el intercambio de un sólo mensaje cada vez: El cliente prepara un mensaje, lo envía en un único segmento UDP y el servidor le devuelve los mismos datos recibidos en otro segmento UDP.

Servidor

El servidor de esta aplicación no tiene nada de particular. Simplemente ha de guardar en un buffer los datos que recibe por un socket UDP y, tal y como están, tendrá que devolvérselos al emisor por el mismo socket. A pesar de que está establecido que la aplicación *Echo* usa el puerto 7, en esta práctica utilizaremos el puerto 50007 para que no sea necesario tener permisos de administración.

Cliente

El cliente de la aplicación irá leyendo líneas de texto desde la entrada estándar y se las irá enviando al servidor utilizando la codificación UTF-8 (el nombre o la dirección IP del servidor será pasada como argumento al programa por el usuario). Por cada línea de texto que el cliente envíe, se tendrá que quedar a la espera del "eco" enviado por el servidor y mostrará por pantalla lo recibido. Puesto que se usa el protocolo de transporte UDP, todos los caracteres enviados en un mensaje serán recibidos en un solo paquete (es decir, con una sola lectura).

El cliente dará por terminada la comunicación con el servidor, cuando lea por la entrada estándar un mensaje vacío (pulsando solo la tecla ENTER) o cuando ocurra algún otro error.

Dinámica de trabajo

1. Probar el cliente compilado proporcionado contra el servidor remoto.
2. Implementar el cliente y probarlo contra el servidor remoto.
3. Implementar el servidor y probarlo con el cliente compilado proporcionado y el cliente implementado por el/la estudiante.

Fichero *“echo_cli_udp.py”*:

```
#!/usr/bin/env python3

import socket, sys

PORT = 50007

# Comprueba que se ha pasado un argumento.
if len( sys.argv ) != 2:
      print( "Uso: {} <servidor>".format( sys.argv[0] ) )
      exit( 1 )

"""A COMPLETAR POR EL/LA ESTUDIANTE:
Crear el socket.
"""

print( "Introduce el mensaje que quieres enviar (mensaje vacío para
terminar):" )
while True:
      mensaje = input()
      if not mensaje:
            break
      """A COMPLETAR POR EL/LA ESTUDIANTE:
      Enviar mensaje y recibir 'eco'.
      Mostrar en pantalla lo recibido.
      """
"""A COMPLETAR POR EL/LA ESTUDIANTE:
Cerrar socket.
"""
```

Fichero *“echo_ser_udp.py”*:

```
#!/usr/bin/env python3

import socket

PORT = 50007

"""A COMPLETAR POR EL/LA ESTUDIANTE:
Crear un socket y asignarle su direccion.
"""

while True:
      """A COMPLETAR POR EL/LA ESTUDIANTE:
      Recibir un mensaje y responder con el mismo.
      """
"""A COMPLETAR POR EL/LA ESTUDIANTE:
Cerrar socket.
"""
```

Posibles mejoras

1. Modifica el cliente para que muestre en pantalla la longitud de los mensajes enviados y recibidos. Muestra la longitud tanto en caracteres como en bytes.
2. Modifica el servidor para que muestre en pantalla la dirección IP y puerto del emisor de los mensajes recibidos.
3. Modifica el cliente para que muestre en pantalla su dirección IP y puerto (utiliza la función *getsockname* del módulo *socket*).
4. Modifica la llamada a la función *bind* para que el sistema asigne un puerto cualquiera al servidor. Para que puedas comunicarte con él a través del cliente primero tendrás que averiguar qué puerto se le ha asignado.
5. Modifica el cliente para que pregunte al usuario el nombre del servidor con el que se quiere comunicar en caso de que no haya sido introducido como argumento.
6. Prueba tu cliente con un servidor que se esté ejecutando en otro ordenador.
7. Establece un tamaño máximo de mensaje que el cliente podrá enviar (usa un valor pequeño) y si el usuario introduce un mensaje más largo envíalo troceado en varios paquetes.