
UNIVERSIDADE DA CORUÑA
Departamento de Tecnoloxías da Información
e as Comunicacións

# LABORATORIO DE RC

## PRÁCTICA 1: IMPLEMENTACIÓN DE UN CLIENTE Y SERVIDOR DE ECO UDP

## PRÁCTICA 2: IMPLEMENTACIÓN DE UN CLIENTE Y SERVIDOR DE ECO TCP

# PRESENTACIÓN

El laboratorio de Redes constará de un conjunto de prácticas obligatorias sobre diversos protocolos del nivel de aplicación. Estas prácticas se realizarán en el laboratorio 0.3. Las prácticas se realizarán **individualmente** y la evaluación de las prácticas se realizará mediante la defensa de las prácticas 3 y 4 en el laboratorio. Las prácticas 1 y 2 se presentan como introducción al laboratorio de Redes, pero no se evalúan.

Cada práctica constará de un enunciado en donde se plantearán las tareas a realizar, y podrá incluir información complementaria para el desarrollo de la misma. Todas las prácticas deberán ser realizadas en **Java**.

A continuación se detalla el enunciado correspondiente a las prácticas 1 y 2.

# PRÁCTICA 1: IMPLEMENTACIÓN DE UN CLIENTE Y SERVIDOR DE ECO UDP

Esta práctica consiste en el desarrollo de un cliente y un servidor que implementen un servicio de eco sobre UDP. El servicio de eco consiste en que el cliente envía un mensaje (de texto) al servidor que este vuelve a enviar de vuelta al cliente.
Se recomienda iniciar la práctica con el desarrollo del cliente de eco y probar su funcionamiento utilizando el puerto 7 de los servidores del laboratorio.

Como ayuda para la realización de la práctica se presenta el pseudo-código del cliente y el servidor.

```
import java.net.*;

/** Ejemplo que implementa un cliente de eco usando UDP. */

public class ClienteUDP{

   public static void main (String argv[]){
       if (argv.length != 3){
           System.err.println("Formato: ClienteUDP <maquina> <puerto>
<mensaje>");
           System.exit(-1);
       }

       try{
            // Obtener la dirección IP del servidor
          // Creamos el socket no orientado a conexión (en cualquier
puerto)
          // Establecemos un timeout de 30 segs

          // Preparamos el datagrama que vamos a enviar y lo enviamos

          // Preparamos el datagrama de recepción

          // Cerramos el socket para liberar la conexión
       } catch(SocketTimeoutException e){
          // Han pasado 30 segundos sin recibir nada
       } catch (Exception e){
          System.err.println("Error: " + e.getMessage());
          e.printStackTrace();
       }
   }
}
```

```
import java.net.*;

/** Ejemplo que implementa un servidor de eco usando UDP. */

public class ServidorUDP{

   public static void main (String argv[]){
       if (argv.length != 1){
           System.err.println("Formato: ServidorUDP <puerto>");
           System.exit(-1);
       }

       try{
          // Creamos el socket del servidor

          // Bucle infinito
             // Preparamos un datagrama para recepción

             // Preparamos el datagrama que vamos a enviar y lo
enviamos
          // Cerramos el socket del servidor
       } catch (Exception e){
          System.err.println("Error: " + e.getMessage());
          e.printStackTrace();
       }
   }
}
```

# PRÁCTICA 2: IMPLEMENTACIÓN DE UN CLIENTE Y SERVIDOR DE ECO TCP

Esta práctica consiste en el desarrollo de un cliente y un servidor multithread que implementen un servicio de eco sobre TCP.
Al igual que en el caso anterior, se recomienda iniciar la práctica con el desarrollo del cliente de eco y probar su funcionamiento utilizando el puerto 7 de los servidores del laboratorio.

Como ayuda para la realización de la práctica se presenta el pseudo-código del cliente y el servidor.

```
import java.net.*;
import java.io.*;

/** Ejemplo que implementa un cliente de eco usando TCP. */

public class ClienteTCP{

   public static void main (String argv[]){
       if (argv.length != 3){
           System.err.println("Formato: ClienteTCP <maquina> <puerto>
<mensaje>");
           System.exit(-1);
       }

       try{
          // Obtener la dirección del servidor
          // Establecemos la conexión con el servidor al crear el
socket

          // Establecemos un timeout de 30 segs

          // Establecemos el canal de entrada
          // Establecemos el canal de salida

          // Enviamos el mensaje

          // Leemos la respuesta del servidor

          // Cerramos el socket para liberar la conexión
       } catch(SocketTimeoutException e){
          // Han pasado 30 segundos sin recibir nada
       } catch (Exception e){
          System.err.println("Error: " + e.getMessage());
          e.printStackTrace();
       }
   }
}
```

```
import java.net.*;
import java.io.*;

/** Ejemplo que implementa un servidor de eco usando TCP. */

public class ServidorTCP{

   public static void main (String argv[]){
       if (argv.length != 1){
           System.err.println("Formato: ServidorTCP <puerto>");
           System.exit(-1);
       }

       try{
          // Creamos el socket del servidor

          // Bucle infinito
             // Esperamos posibles conexiones

             // Lanzamos un nuevo thread para atender la nueva
conexión
          // Cerramos el socket
       } catch (Exception e){
          System.err.println("Error: " + e.getMessage());
          e.printStackTrace();
       }
   }
}
```

---

```
import java.net.*;
import java.io.*;

/** Thread que atiende una conexión de un servidor de eco. */

public class ThreadServidor extends Thread{

   // Constructor a partir de un socket

   public void run(){
      try{
          // Establecemos el canal de entrada
          // Establecemos el canal de salida

          // Recibimos el mensaje del cliente
          // Enviamos la respuesta

          // Cerramos el socket para liberar la conexión
      } catch (Exception e){
          System.err.println("Error: " + e.getMessage());
      }
   }
}
```