



UNIDAD PROFESIONAL INTERDISCIPLINARIA EN
INGENIERÍA Y TECNOLOGÍAS
AVANZADAS – IPN



Materia

Sistemas distribuidos

Alumno

Suárez Ángeles Danna Paola
Martinez Aparicio Eduardo

Tema

Practica 1 SOCKET

Grupo

2TV7

Introducción

La comunicación entre procesos distribuidos es un pilar fundamental en el desarrollo de aplicaciones en red. Uno de los mecanismos más utilizados para este propósito son los **sockets**, que permiten establecer un canal de comunicación bidireccional entre programas ejecutados en diferentes computadoras. Un socket actúa como un punto final en la transmisión de datos, proporcionando una interfaz estandarizada para el envío y recepción de información a través de protocolos como TCP o UDP.

En esta práctica se abordará la implementación de sockets en un entorno remoto, es decir, entre dos equipos conectados a una misma red (WiFi o Ethernet) o incluso pertenecientes a redes distintas. Para el segundo caso, es necesario gestionar adecuadamente las direcciones IP públicas y los puertos de comunicación, o bien emplear soluciones de middleware como Hamachi, que facilita la interconexión de programas en redes diferentes al simular una red privada virtual (VPN).

El objetivo principal de la actividad es que el estudiante comprenda y experimente el proceso de creación de aplicaciones cliente-servidor basadas en sockets, desarrollando habilidades para la configuración de la comunicación remota y reforzando los conceptos de direccionamiento IP, protocolos de transporte y manejo de redes.

Preguntas anexadas al reporte

¿Qué necesitamos para que pueda comunicarse el programa Servidor (codificado en java) con un cliente (codificado en C) y viceversa?

Un protocolo de comunicación común: Ambos programas deben "hablar el mismo idioma"

- TCP/IP o UDP para la capa de transporte
- Un formato de datos acordado (texto plano, JSON, XML, binario, etc.)

Sockets de red: La interfaz de programación que permite la comunicación

- Ambos lenguajes tienen implementaciones de sockets
- Java: `java.net.Socket` y `java.net.ServerSocket`
- C: funciones de `<sys/socket.h>` en Unix/Linux o `<winsock2.h>` en Windows

Serialización de datos compatible: Convertir datos a un formato transmisible

- Usar formatos independientes del lenguaje (JSON, XML, Protocol Buffers)
- Evitar formatos específicos del lenguaje (Java Serialization, por ejemplo)

• ¿Cómo se llama esta característica/funcionalidad en un sistema distribuido?

Esta funcionalidad se llama Interoperabilidad (o *interoperability* en inglés).

Definición: Es la capacidad de diferentes sistemas, aplicaciones o componentes - potencialmente escritos en diferentes lenguajes y ejecutándose en diferentes plataformas - de trabajar juntos e intercambiar información de manera efectiva.

• ¿Qué es lo que permite que esta característica ocurra?

La interoperabilidad es posible gracias a varios elementos:

1. Estándares abiertos y protocolos:
 - TCP/IP, HTTP, WebSockets
 - Permiten que cualquier implementación compatible pueda comunicarse
2. Formatos de datos neutros al lenguaje:
 - JSON, XML, Protocol Buffers, MessagePack
 - Se pueden leer/escribir desde cualquier lenguaje
3. APIs de red del sistema operativo:
 - El SO proporciona sockets como abstracción
 - Cualquier lenguaje puede acceder a ellos
4. Capa de abstracción de red:
 - Separa la lógica de aplicación de los detalles de implementación
 - Permite que diferentes tecnologías se integren

Se realizó la compilación y ejecución de un servidor de eco en Java.

The image shows two side-by-side Windows PowerShell terminal windows. The left window displays the directory structure and the compilation of the Java server. The right window shows the execution of the server and the client's response.

```
-a---- 03/09/2025 03:49 p. m. 2132 SpotifyConsola.py
-a---- 03/09/2025 03:49 p. m. 1423 SpotifyGui.py
-a---- 03/09/2025 03:49 p. m. 709 usuarios.py

PS C:\Users\eduma\Downloads\practicass-Distribuidos-main\practicass-Distribuidos-main> cd practicauno
PS C:\Users\eduma\Downloads\practicass-Distribuidos-main\practicass-Distribuidos-main\practicauno> dir

Directorio: C:\Users\eduma\Downloads\practicass-Distribuidos-main\practicass-Distribuidos-main\practicauno

Mode                LastWriteTime         Length Name
----                -
-a---- 03/09/2025 03:49 p. m.         2657 ClienteTocToc.java
-a---- 03/09/2025 03:49 p. m.         1420 Cliente_de_Eco.java
-a---- 03/09/2025 03:49 p. m.         2305 ProtocoloTocToc.java
-a---- 03/09/2025 03:49 p. m.         2837 ServidorTocToc.java
-a---- 03/09/2025 03:49 p. m.          873 ServidorTocTocMultiple.java
-a---- 03/09/2025 03:49 p. m.         1784 ServidorTocTocMultipleHilos.java
-a---- 03/09/2025 03:49 p. m.         1452 Servidor_Eco.java

PS C:\Users\eduma\Downloads\practicass-Distribuidos-main\practicass-Distribuidos-main\practicauno> javac Servidor_Eco.java
PS C:\Users\eduma\Downloads\practicass-Distribuidos-main\practicass-Distribuidos-main\practicauno> java Servidor_Eco 5000
```

```
-a---- 03/09/2025 03:49 p. m. 1420 Cliente_de_Eco.java
-a---- 03/09/2025 03:49 p. m. 2305 ProtocoloTocToc.java
-a---- 03/09/2025 03:49 p. m. 2837 ServidorTocToc.java
-a---- 03/09/2025 03:49 p. m. 873 ServidorTocTocMultiple.java
-a---- 03/09/2025 03:49 p. m. 1784 ServidorTocTocMultipleHilos.java
-a---- 03/10/2025 08:49 p. m. 2316 Servidor_Eco.class
-a---- 03/09/2025 03:49 p. m. 1452 Servidor_Eco.java

PS C:\Users\eduma\Downloads\practicass-Distribuidos-main\practicass-Distribuidos-main\practicauno> java Cliente_de_Eco 127.0.0.1 5000
Error: Could not find or load main class Cliente_de_Eco
Caused by: java.lang.ClassNotFoundException: Cliente_de_Eco
PS C:\Users\eduma\Downloads\practicass-Distribuidos-main\practicass-Distribuidos-main\practicauno> java Cliente_de_Eco.java
Uso desde consola: java Cliente_de_Eco <nombre de host (computadora)> <numero de puerto>
PS C:\Users\eduma\Downloads\practicass-Distribuidos-main\practicass-Distribuidos-main\practicauno> java Cliente_de_Eco 127.0.0.1 5000
Uso desde consola: java Cliente_de_Eco <nombre de host (computadora)> <numero de puerto>
PS C:\Users\eduma\Downloads\practicass-Distribuidos-main\practicass-Distribuidos-main\practicauno> javac Cliente_de_Eco.java
PS C:\Users\eduma\Downloads\practicass-Distribuidos-main\practicass-Distribuidos-main\practicauno> java Cliente_de_Eco 127.0.0.1 5000
hola
dos
El eco del servidor dice: dos
```

Se realizó una prueba exitosa de comunicación cliente-servidor.

The image shows two side-by-side Windows PowerShell terminal windows. The left window displays the directory structure and the compilation of the Java server. The right window shows the execution of the server and the client's response, including a successful echo test.

```
-a---- 03/09/2025 03:49 p. m. 2132 SpotifyConsola.py
-a---- 03/09/2025 03:49 p. m. 1423 SpotifyGui.py
-a---- 03/09/2025 03:49 p. m. 709 usuarios.py

PS C:\Users\eduma\Downloads\practicass-Distribuidos-main\practicass-Distribuidos-main> cd practicauno
PS C:\Users\eduma\Downloads\practicass-Distribuidos-main\practicass-Distribuidos-main\practicauno> dir

Directorio: C:\Users\eduma\Downloads\practicass-Distribuidos-main\practicass-Distribuidos-main\practicauno

Mode                LastWriteTime         Length Name
----                -
-a---- 03/09/2025 03:49 p. m.         2657 ClienteTocToc.java
-a---- 03/09/2025 03:49 p. m.         1420 Cliente_de_Eco.java
-a---- 03/09/2025 03:49 p. m.         2305 ProtocoloTocToc.java
-a---- 03/09/2025 03:49 p. m.         2837 ServidorTocToc.java
-a---- 03/09/2025 03:49 p. m.          873 ServidorTocTocMultiple.java
-a---- 03/09/2025 03:49 p. m.         1784 ServidorTocTocMultipleHilos.java
-a---- 03/09/2025 03:49 p. m.         1452 Servidor_Eco.java

PS C:\Users\eduma\Downloads\practicass-Distribuidos-main\practicass-Distribuidos-main\practicauno> javac Servidor_Eco.java
PS C:\Users\eduma\Downloads\practicass-Distribuidos-main\practicass-Distribuidos-main\practicauno> java Servidor_Eco 5000
```

```
-a---- 03/09/2025 03:49 p. m. 1420 Cliente_de_Eco.java
-a---- 03/09/2025 03:49 p. m. 2305 ProtocoloTocToc.java
-a---- 03/09/2025 03:49 p. m. 2837 ServidorTocToc.java
-a---- 03/09/2025 03:49 p. m. 873 ServidorTocTocMultiple.java
-a---- 03/09/2025 03:49 p. m. 1784 ServidorTocTocMultipleHilos.java
-a---- 03/10/2025 08:49 p. m. 2316 Servidor_Eco.class
-a---- 03/09/2025 03:49 p. m. 1452 Servidor_Eco.java

PS C:\Users\eduma\Downloads\practicass-Distribuidos-main\practicass-Distribuidos-main\practicauno> java Cliente_de_Eco 127.0.0.1 5000
Error: Could not find or load main class Cliente_de_Eco
Caused by: java.lang.ClassNotFoundException: Cliente_de_Eco
PS C:\Users\eduma\Downloads\practicass-Distribuidos-main\practicass-Distribuidos-main\practicauno> java Cliente_de_Eco.java
Uso desde consola: java Cliente_de_Eco <nombre de host (computadora)> <numero de puerto>
PS C:\Users\eduma\Downloads\practicass-Distribuidos-main\practicass-Distribuidos-main\practicauno> java Cliente_de_Eco 127.0.0.1 5000
Uso desde consola: java Cliente_de_Eco <nombre de host (computadora)> <numero de puerto>
PS C:\Users\eduma\Downloads\practicass-Distribuidos-main\practicass-Distribuidos-main\practicauno> javac Cliente_de_Eco.java
PS C:\Users\eduma\Downloads\practicass-Distribuidos-main\practicass-Distribuidos-main\practicauno> java Cliente_de_Eco 127.0.0.1 5000
hola
dos
El eco del servidor dice: dos
practica realizada por lalo y dana
practica realizada por lalo y dana
El eco del servidor dice: practica realizada por lalo y dana
```

Se muestra la salida del cliente donde se visualizan las respuestas recibidas del servidor de eco.

```
El eco del servidor dice: dos
practica realizada por lalo y dana
practica realizada por lalo y dana
El eco del servidor dice: practica realizada por lalo y dana
```

II.---Com unicación con Sockets remota m ente

Para este ejercicio se requiere que la ejecución se realice en una red WiFi o Ethernet,

entre dos computadoras que pertenecen a la misma red o a diferentes redes, (si las

redes son distintas, entonces debes poder indicar/gestionar la dirección IP de t u c omputadora como IP pública o en su defecto, puedes instalar un software de

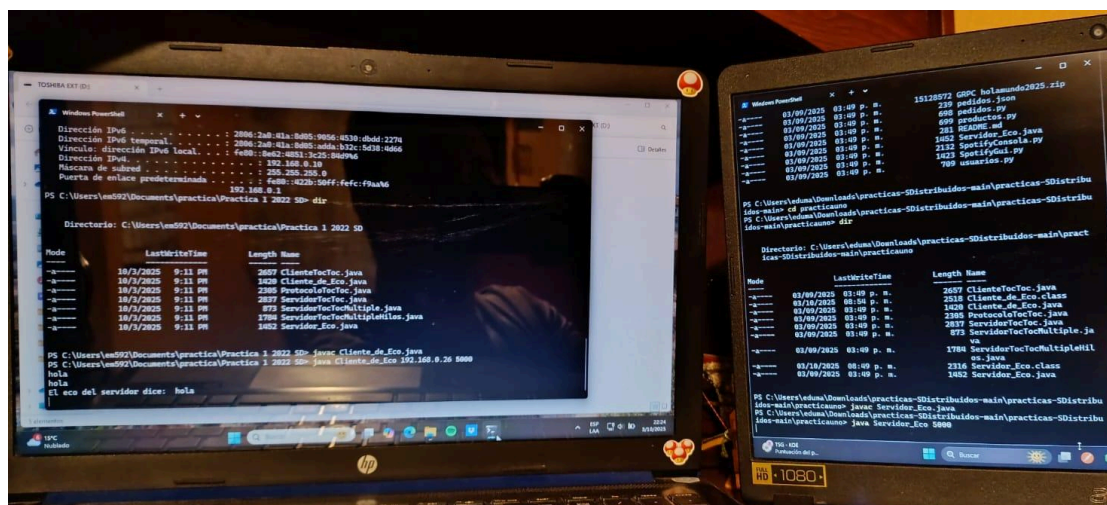
tipo MiddleWare para facilitar y lograr la comunicación entre tus programas (e.g. la

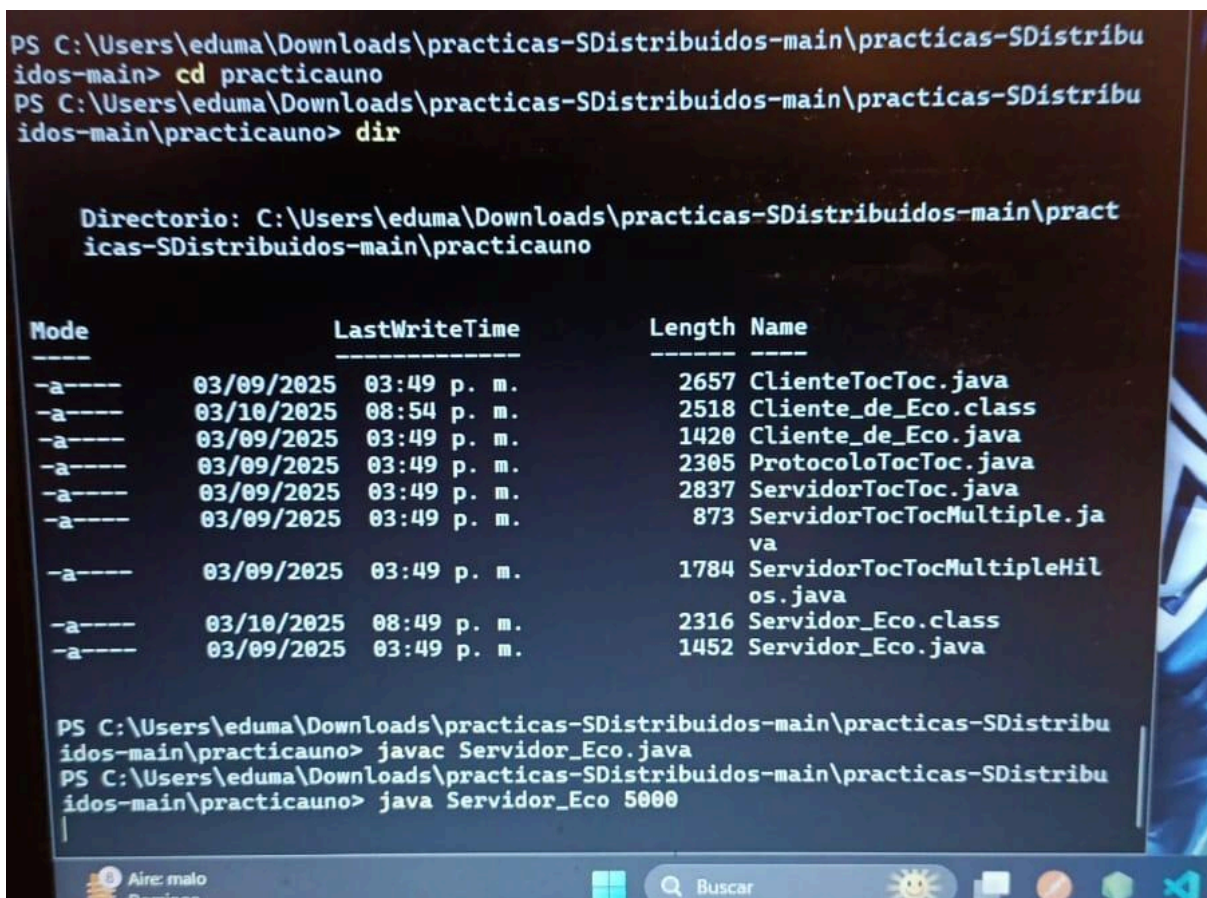
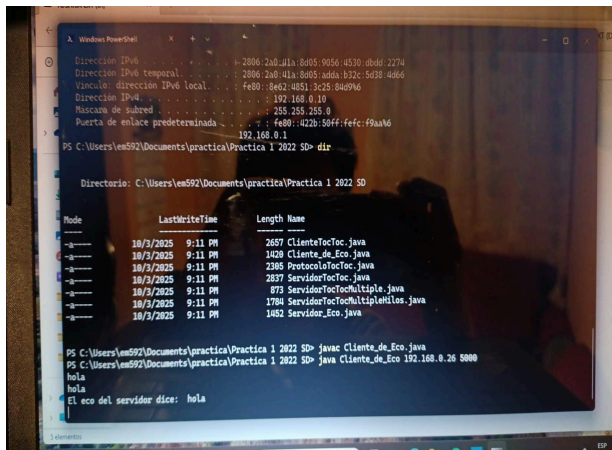
herramienta Hamachi permite comunicar programas que están en diferentes redes).

a) Ejecute el programa EchoServer en una computadora A.

a) Ejecute el programa Cliente_de_Eco en la computadora B ;

Debe lograrse el mismo resultado de comunicación que en paso I inciso a)





Ejercicio de tarea

1.- Codifique dos programas usando sockets, en el enfoque cliente-servidor, que permita el intercambio de mensajes de texto..

- o El programa Servidor (debe ser codificado en java)
- o Mientras que el cliente (debe ser codificado en C)

• Funcionamiento: Cuando se conecten entre si, el cliente enviará una cadena de texto cualquiera, por ejemplo Hola y el Servidor debe responder con algún otro mensaje, por ejemplo, Hola que tal.

```

PS C:\Users\eduma\Documents\Practicaone> dir

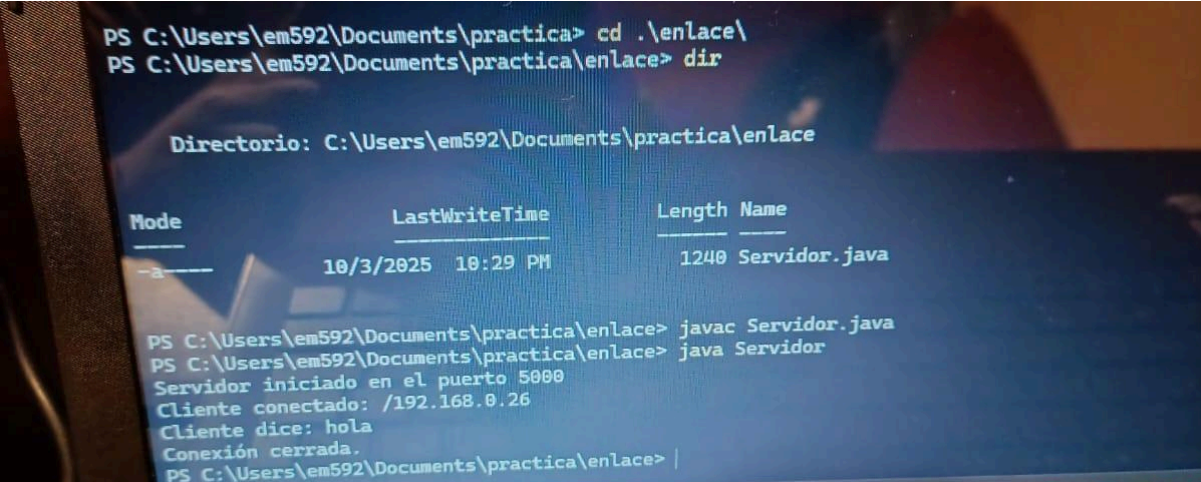
Directorio: C:\Users\eduma\Documents\Practicaone

Mode                LastWriteTime         Length Name
----                -
-a-----         03/10/2025  10:50 p. m.         1363 Cliente_Eco.java
-a-----         03/10/2025  10:29 p. m.         1240 Servidor.java

PS C:\Users\eduma\Documents\Practicaone> javac Cliente_Eco.java
PS C:\Users\eduma\Documents\Practicaone> java Cliente_Eco 192.168.0.10 5000
Conectado al servidor 192.168.0.10 en el puerto 5000
Escribe un mensaje: hola
Servidor responde: Hola que tal, recibí tu mensaje: hola
PS C:\Users\eduma\Documents\Practicaone>

```

cliente



```

PS C:\Users\em592\Documents\practica> cd .\enlace\
PS C:\Users\em592\Documents\practica\enlace> dir

Directorio: C:\Users\em592\Documents\practica\enlace

Mode                LastWriteTime         Length Name
----                -
-a-----         10/3/2025  10:29 PM         1240 Servidor.java

PS C:\Users\em592\Documents\practica\enlace> javac Servidor.java
PS C:\Users\em592\Documents\practica\enlace> java Servidor
Servidor iniciado en el puerto 5000
Cliente conectado: /192.168.0.26
Cliente dice: hola
Conexión cerrada.
PS C:\Users\em592\Documents\practica\enlace> |

```

servidor

2.- Codifique dos programas usando sockets, en el enfoque cliente-servidor, que permita que se envíen números enteros entre sí

El programa Servidor (debe ser codificado en C)

El programa cliente (debe ser codificado en Java)

- Funcionamiento: Cuando se conecten entre si, el cliente enviará un entero y el servidor lo incrementará en uno.

Ejemplo, el cliente envía un 5 y el servidor contesta con un 6, el programa terminará cuando el cliente escriba un cero.

La prueba y revisión de este ejercicio consiste en:

```
Directorio: C:\Users\em592\Documents\practica\enlace

Mode                LastWriteTime         Length Name
----                -
-a-                10/3/2025  10:59 PM           1420 clienteentero.java
-a-                10/3/2025  10:36 PM           2144 Servidor.class
-a-                10/3/2025  10:29 PM           1240 Servidor.java
-a-                10/3/2025  10:59 PM           1231 ServidorEntero.java

PS C:\Users\em592\Documents\practica\enlace> javac ServidorEntero.java
PS C:\Users\em592\Documents\practica\enlace> java ServidorEntero
Servidor escuchando en el puerto 5000
Cliente conectado: /192.168.0.26
Cliente envió: 5
Cliente envió: 6
Cliente envió: 8
```

servidor

```
PS C:\Users\eduma\Documents\Practicaone> javac ClienteEntero.java
PS C:\Users\eduma\Documents\Practicaone> java ClienteEntero 192.168.0.10 500
0
Conectado al servidor 192.168.0.10:5000
Escribe un número (0 para salir): 5
Servidor respondió: 6
Escribe un número (0 para salir): 6
Servidor respondió: 7
Escribe un número (0 para salir): 8
Servidor respondió: 9
Escribe un número (0 para salir):
```

cliente