

# Tema 3: Java y los Threads

**ORACLE** (Sign In/Register for Account | Help) United States Communities I am a... I want to... Secure Search

Products and Services Solutions Downloads Store Support Training Partners About Oracle Technology Network

Oracle Technology Network > Java

### MOVING FORWARD

## JAVA

### FORWARD

**RELEASE**

#### JavaFX 2.0 Arrives and Will Be Open Sourced

Announced at JavaOne, JavaFX 2.0 has arrived with Java APIs for JavaFX, FXML, a new graphics pipeline for modern GPUs, and more than 50 easily customizable components. The source code will be released at OpenJDK as project OpenJFX.

Posted 10/3/11 // Tags: java, javafx, RIA // Headlines Archive

#### Software Downloads View All Downloads

Top Downloads	New Downloads
Java SE	Java SE 7 Update 1
Java EE and GlassFish	Released 10/18/11
Java FX	Java SE 6 Update 29
Java ME	Released 10/18/11
JDeveloper 11g and ADF	JavaFX 2.0.1
Enterprise Pack for Eclipse	Released 10/18/11
NetBeans IDE	NetBeans 7.1 Beta
Pre-Built VM for Java Developers	Released 10/4/11
Get Java	GlassFish Server 3.1.1
	Released 7/28/11
	Oracle JDeveloper 11g R2
	Released 6/6/11

#### Essential Links

- About Us/Become a Member
- Java APIs
- Technical Articles
- New to Java
- Java Certification & Training
- Java Bug Database
- "The Java Source" Blog
- Java on Twitter
- Java Developer Newsletter
- Java Magazine
- Delicious Feed
- java.net
- Java User Groups
- Java Community Process
- Facebook | Forums
- Events | Developer Days

#### Developer Spotlight

- Using MWait in Spin Loops
- New LWUIT Tutorial
- Oracle Enterprise Pack for Eclipse with Support for ADF, Coherence and Public Cloud
- How to Build a Successful Java User Groups (JUG)
- Why Developers Should Not Write Programs That Call 'sun' Packages
- Java Reloaded
- JDK 7 Adoption Guide
- JCP Executive Committee Elections
- The Importance of Twitter's Participation in OpenJDK
- Javatuples 1.2 Released
- JavaOne Content on Parleys.com
- Introducing the Oracle Java Cloud Service: For Standard Java EE Apps
- JavaFX 2.0 and Scala, Like Milk and Cookies
- Java 7 Summit at EclipseConEurope (Germany, 02-04 Nov.)

#### Blogs

- Latest OEPE (11.1.1.8) - Eclipse 3.7.1-based  
Posted: Nov 9
- Java Spotlight Episode 55: Georges Saab, Vice President of Development in the Java Platform Group  
Posted: Nov 9
- Changing a Node from its Popup Menu  
Posted: Nov 8
- Devbox for Java Developers  
Posted: Nov 8
- The Last Migration - GlassFish Wiki  
Posted: Nov 8
- Tip: "Unset as Main Project"  
Posted: Nov 8
- Put Siri-like tech on your Java ME mobile phone with Zvpr  
Posted: Nov 8
- GlassFish 3.1.2 themes and features  
Posted: Nov 7
- On the methodvalue attribute in javar.xml file  
Posted: Nov 6

#### Technologies

- Java SE
- Java SE Support
- Java SE Advanced & Suite
- Java Embedded
- Java EE
- Java ME
- JavaFX
- Java Card
- Java TV
- Java DB
- Developer Tools

#### Java Spotlight Podcast

- Java Spotlight Episode 54: Stuart Marks on the Coincification of JDK7  
Posted: Oct 2
- Java Spotlight Episode 53: Mark Reinhold, Chief Architect of the Java Platform Group  
Posted: Oct 25
- Java Spotlight Episode 52: Cameron Purdy, Vice President of Development at Oracle, on JavaEE  
Posted: Oct 18

#### JavaOne

NEW!

Java magazine Get it now for FREE!

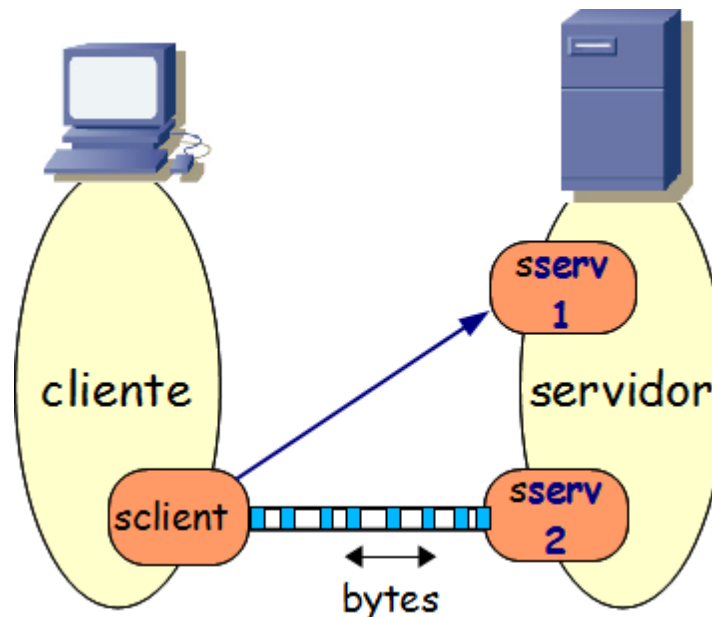
Subscribe Today



## Bibliografía:

- ❑ [Kurose10] Apartados 2.1, 2.7, 2.8
- ❑ <http://www.oracle.com/technetwork/java/index.html>
- ❑ <http://java.com/es/about/>
- ❑ <http://download.oracle.com/javase/6/docs/api/index.html>

- Cliente:
  - Cuando crea un socket (`sclient`) establece la conexión con el servidor
- Servidor:
  - Debe haber creado un socket (`sserv1`) donde espera a los clientes que conectan con él
  - Cuando un cliente se conecta con un servidor:
  - El servidor crea un nuevo socket (`sserv2`) para que el proceso servidor se comuniquen con el cliente
  - De esta forma es posible que un servidor se comuniquen con varios clientes simultáneamente



- Clase `ServerSocket`

- Constructores

- `ServerSocket(int puerto) throws IOException`
  - Abre un socket en el puerto indicado en modo de escucha
  - Si `port=0`, entonces se elige cualquier puerto libre
- `ServerSocket(int puerto, int backlog) throws IOException`
  - Abre un socket en el `puerto` indicado en modo de escucha
  - `backlog` indica la longitud máxima de la cola de conexiones en espera
  - Cuando llega una solicitud de conexión y la cola está llena, se rechaza la conexión



- Clase **ServerSocket**
- Algunos métodos importantes
  - **Socket accept() throws IOException**
    - Acepta una conexión de un cliente y devuelve un socket asociado a ella
    - El proceso se bloquea hasta que se realiza una conexión
    - El diálogo con el cliente se hace por el nuevo socket
    - El ServerSocket puede atender nuevas conexiones
  - **close() throws IOException**
    - Cierra el socket servidor



- Normalmente, un servidor debe estar preparado para atender muchos clientes
- Se puede hacer de dos maneras:
  - **Secuencial**: un cliente detrás de otro
  - **Concurrente**: varios clientes al mismo tiempo

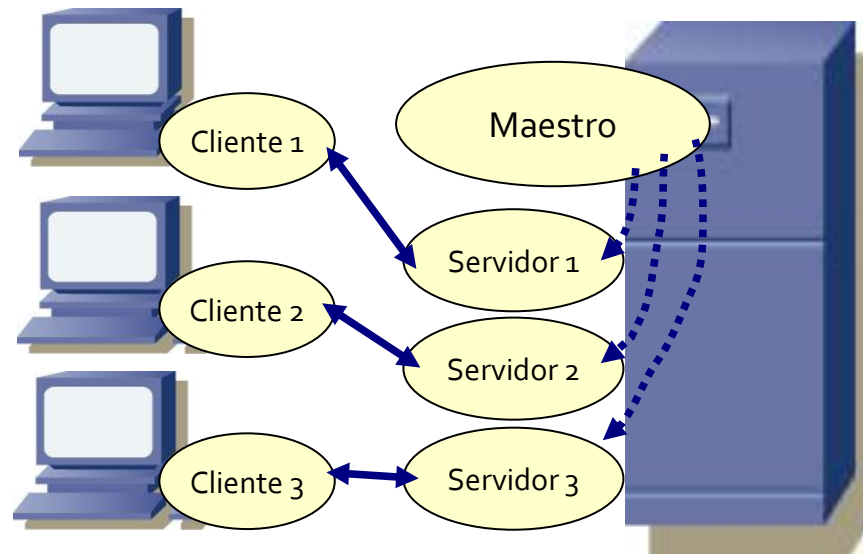


- En Java, la concurrencia la conseguimos usando hilos de ejecución
- Clase Thread
  - Se define una clase derivada de Thread
  - Código a ejecutar en cada hilo dentro del método run()
  - Se lanza el hilo con start()

```
class Hilos extends Thread {  
    int id;  
    public Hilos(int i)  
    {  
        id=i;  
    }  
  
    public void run() {  
        for(int i=0;i<100;i++) {  
            System.out.print(id);  
            try {sleep(100);}  
            catch(InterruptedException e) {}  
        }  
    }  
  
    public static void main(String args[])  
    {  
        for(int i=0;i<3;i++) {  
            Hilos h = new Hilos(i);  
            h.start();  
        }  
    }  
}
```

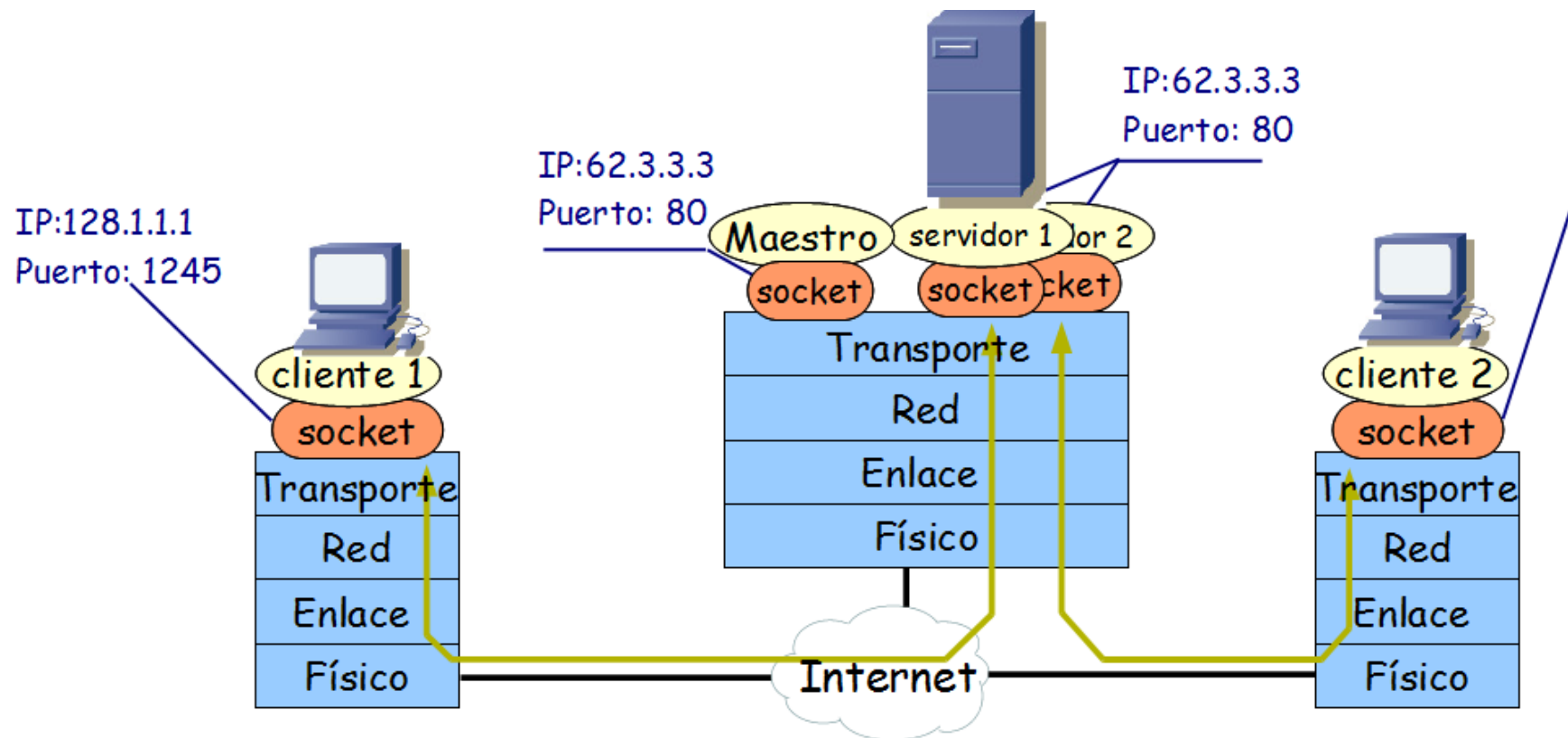


- Diversos hilos de ejecución:
  - En el hilo principal se ejecuta permanentemente el método `accept()`
    - Espera el establecimiento de nuevas conexiones
  - Para cada cliente que se conecta, se lanza un nuevo hilo de ejecución para gestionar esa conexión





- Ahora tenemos varios sockets asociados al mismo puerto
- Para identificar al socket destino hay que tener en cuenta la dirección (dir. IP + puerto) del socket origen





# Servidor concurrente TCP

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

```
class SCTCP extends Thread {
    Socket id;
    public SCTCP(Socket s) {id=s;}
    public void run() {
        try {
            PrintWriter salida=new PrintWriter(id.getOutputStream(),true);
            while(true){
                salida.println(System.currentTimeMillis());
                sleep(100);
            }
        } catch(Exception e) {}
    }
}
```

En el método run, ¡hay que capturar las excepciones!

```
public static void main(String args[]) throws IOException{
    ServerSocket ss=new ServerSocket(8888);
    while(true) {
        Socket s = ss.accept();
        SCTCP t = new SCTCP(s);
        t.start();
    }
}
}
```

