

# Comunicaciones en Red

Programación de servicios y procesos

<https://github.com/GMQPSP>

gmp.psp2019@gmail.com

# Comunicaciones en Red

- **Protocolos de comunicaciones**
- Comunicación entre aplicaciones
- Roles cliente – servidor
- Puertos de comunicaciones.
- Sockets. Creación de sockets.
- Enlazado y establecimiento de conexiones.
- Sockets para transmitir y recibir información.
- Programación de aplicaciones cliente y servidor
- Utilización de hilos en aplicaciones en red.
- Depuración. Monitorización de tiempos de respuesta

# Protocolos de comunicaciones

- Sistema de reglas que permiten que dos o más entidades de un sistema de comunicación se comuniquen entre ellas para transmitir información
- Se trata de las reglas o el estándar que define la sintaxis, semántica y sincronización de la comunicación, así como también los posibles métodos de recuperación de errores
- Los protocolos pueden ser implementados por hardware, por software, o por una combinación de ambos

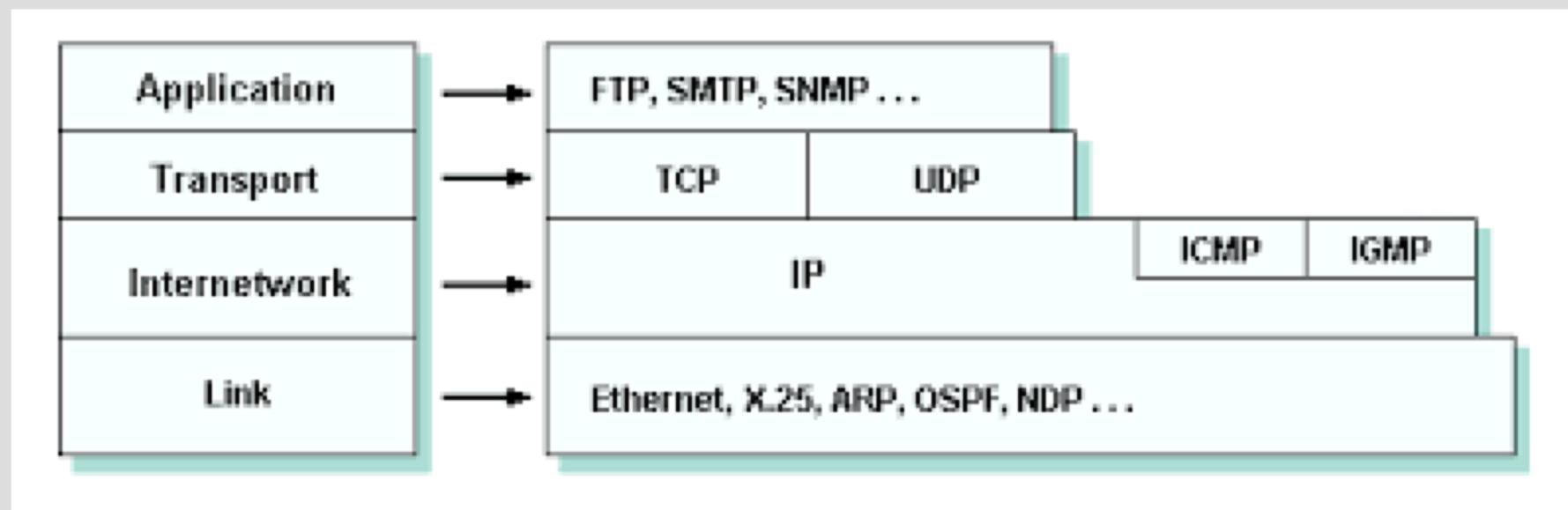
# Protocolos de comunicaciones

- Ejemplos de protocolos más conocidos
  - IP (IPv4, IPv6)
  - TCP, UDP
  - RPC, SSL
  - SMTP, FTP, SSH, HTTP, Telnet, LDAP
  - ...



# Protocolos de comunicaciones

- Pila de protocolos
  - Simplificando modelo OSI:



# Protocolos de comunicaciones

- Protocolos usados en capa de transporte:
  - TCP (Transmission Control Protocol)
    - Protocolo fundamental de internet
    - Transmisión controlada de datos
    - Puerto
  - UDP (User Datagram Protocol)
    - Intercambio de datagramas (datos)
    - Sin establecimiento de conexión
    - Servicios como DNS o DHCP

# Comunicaciones en Red

- Protocolos de comunicaciones
- **Comunicación entre aplicaciones**
- Roles cliente – servidor
- Puertos de comunicaciones.
- Sockets. Creación de sockets.
- Enlazado y establecimiento de conexiones.
- Sockets para transmitir y recibir información.
- Programación de aplicaciones cliente y servidor
- Utilización de hilos en aplicaciones en red.
- Depuración. Monitorización de tiempos de respuesta

# Comunicación entre aplicaciones

- Normalmente, sistemas distribuidos
- Se necesitan dos roles diferentes
  - Sender/Publisher/Server
    - Envía datos
    - Publica mensajes
  - Receiver/Consumer/Client
    - Realiza peticiones
    - Recibe mensajes
- Una arquitectura clásica y habitual es la del *cliente-servidor*

# Comunicación entre aplicaciones

- Por tipo de petición-respuesta
  - Sistemas síncronos
    - Los participantes reciben y envían información (mensajes) en “tiempo real”.
  - Sistemas asíncronos
    - Se envía información y la respuesta llega cuando el otro sistema está disponible
  - Sistemas híbridos:
    - Sistemas que soportan los dos tipos de comunicación

# Comunicación entre aplicaciones

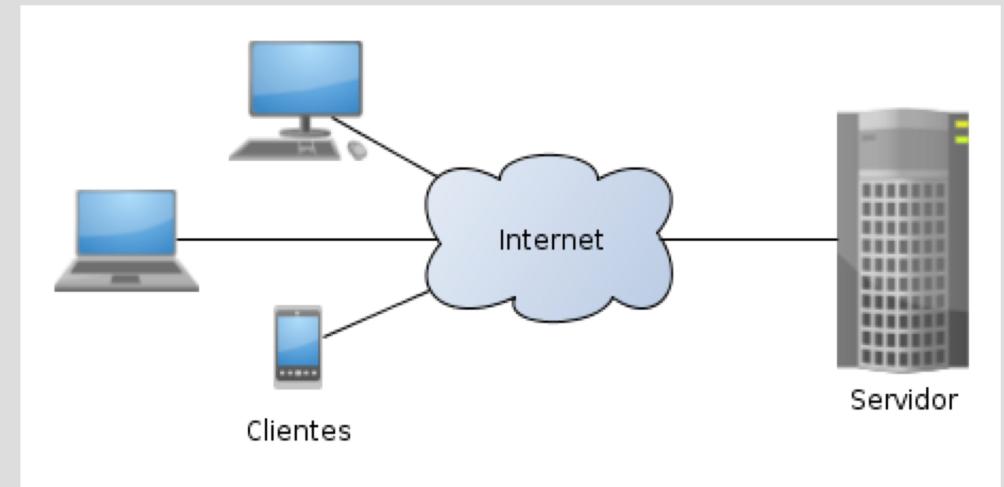
- Por envío de información
  - Request HTTP
    - Utilizado por navegadores en Internet
    - Ejemplo: formulario
  - Colas de mensajes
    - Se necesita un broker que gestione las colas (RabbitMQ, Kafka, ActiveMQ, ...)
    - Protocolo propio de mensajes
  - Ficheros
    - Se envía la información a través de ficheros que las aplicaciones leen y transforman sus datos
    - Procesos batch

# Comunicaciones en Red

- Protocolos de comunicaciones
- Comunicación entre aplicaciones
- **Roles cliente – servidor**
- Puertos de comunicaciones.
- Sockets. Creación de sockets.
- Enlazado y establecimiento de conexiones.
- Sockets para transmitir y recibir información.
- Programación de aplicaciones cliente y servidor
- Utilización de hilos en aplicaciones en red.
- Depuración. Monitorización de tiempos de respuesta

# Cliente - servidor

- Modelo de diseño de software en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados **servidores**, y los demandantes, llamados **clientes**
  - Los clientes realizan peticiones a los servidores
  - Ejemplos:
    - Correo electrónico
    - La Web



# Cliente - servidor

- **Cliente:** remitente de la solicitud
  - Inicia solicitudes o peticiones, tiene un papel activo
  - Espera y recibe las respuestas del servidor
  - Puede conectarse a varios servidores a la vez
  - Normalmente, interactúa con los usuarios a través de una interfaz gráfica

# Cliente - servidor

- **Servidor:** receptor de las solicitudes
  - Espera que le lleguen las solicitudes o peticiones, tiene un papel pasivo (esclavo)
  - Espera recepción de una solicitud, la procesan y luego envían la respuesta al cliente
  - Por lo general, acepta las conexiones de un gran número de clientes

# Cliente - servidor

- **Características**

- Pueden actuar como una sola entidad o como entidades separadas, realizando actividades o tareas independientes.
- En plataformas separadas o en la misma plataforma.
- Cada plataforma puede ser escalable independientemente.
- El acceso a los recursos de la red no muestra la complejidad de los diferentes tipos de formatos de datos y de los protocolos.

# Comunicaciones en Red

- Protocolos de comunicaciones
- Comunicación entre aplicaciones
- Roles cliente – servidor
- **Puertos de comunicaciones.**
- Sockets. Creación de sockets.
- Enlazado y establecimiento de conexiones.
- Sockets para transmitir y recibir información.
- Programación de aplicaciones cliente y servidor
- Utilización de hilos en aplicaciones en red.
- Depuración. Monitorización de tiempos de respuesta

# Puertos de comunicaciones

- En el ámbito de Internet, un puerto es el valor que se usa para distinguir entre las múltiples aplicaciones que se pueden conectar al mismo host, o puesto de trabajo.
- La IANA (Internet Assigned Numbers Authority) determina las asignaciones de todos los puertos comprendidos entre los valores [0, 1023]
- Los puertos numerados en el intervalo [1024, 65535] se pueden registrar con el consenso de la IANA, vendedores de software y otras organizaciones. Por ejemplo, el puerto 1352 se asigna a Lotus Notes.

# Puertos de comunicaciones

- En una misma máquina van a correr diferentes servicios en sendos puertos
  - En el puerto:
    - 80: HTTP
    - 443: HTTPS
    - 21: FTP
    - 22: SSH

# Comunicaciones en Red

- Protocolos de comunicaciones
- Comunicación entre aplicaciones
- Roles cliente – servidor
- Puertos de comunicaciones.
- **Sockets. Creación de sockets.**
- Enlazado y establecimiento de conexiones.
- Sockets para transmitir y recibir información.
- Programación de aplicaciones cliente y servidor
- Utilización de hilos en aplicaciones en red.
- Depuración. Monitorización de tiempos de respuesta

# Sockets. Creación de sockets

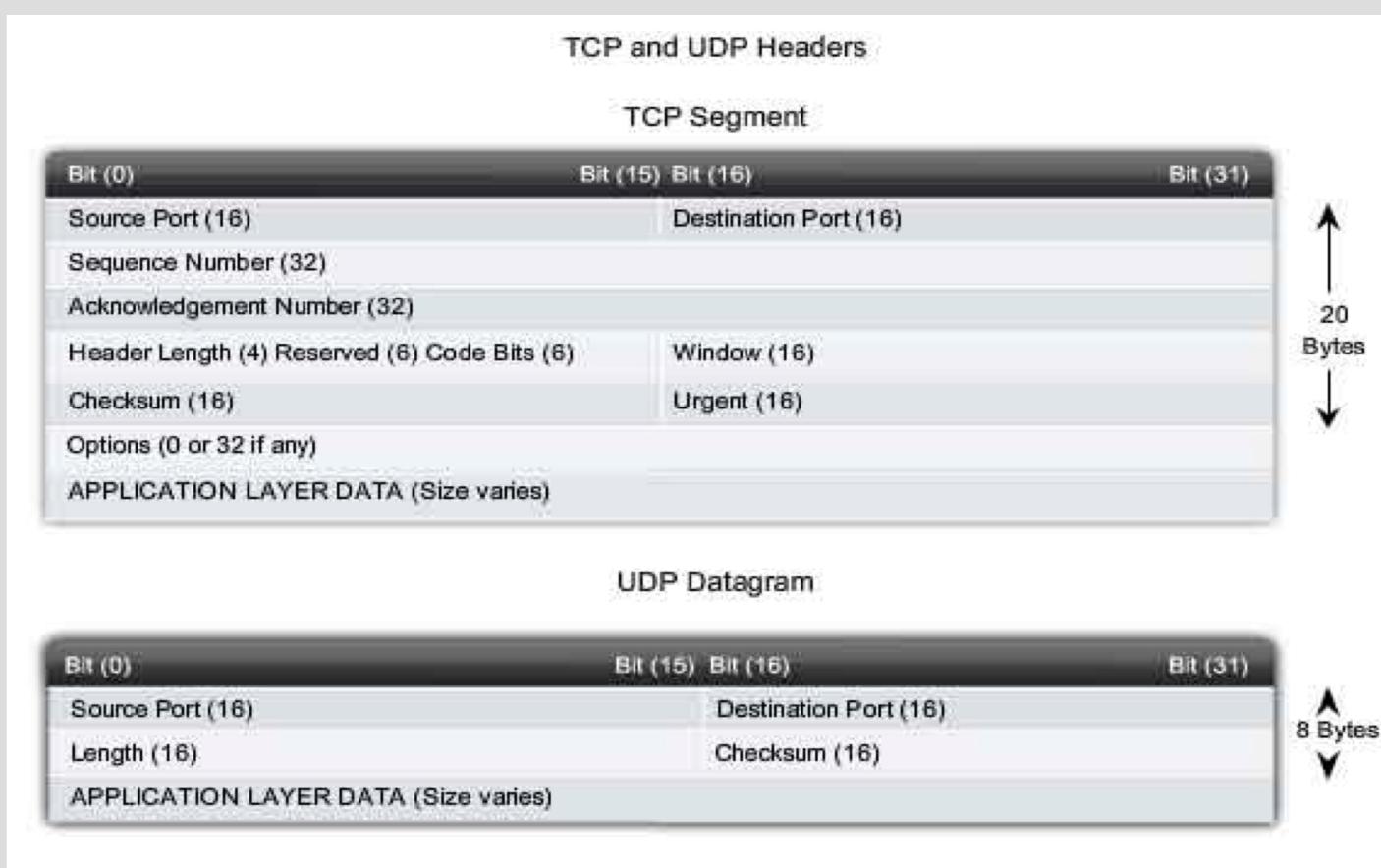
- **Socket:** concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada.
- **Un socket queda definido por un par de direcciones IP local y remota, un protocolo de transporte y un par de números de puerto local y remoto.**
- Para que dos programas puedan comunicarse entre sí es necesario que se cumplan ciertos requisitos:
  - Que un programa sea capaz de localizar al otro.
  - Que ambos programas sean capaces de intercambiarse cualquier secuencia de octetos, es decir, datos relevantes a su finalidad.

# Sockets. Creación de sockets

- Para ello son necesarios los dos recursos que originan el concepto de socket:
  - Un par de direcciones del protocolo de red (dirección IP, si se utiliza el protocolo TCP/IP), que identifican la computadora de origen y la remota.
  - Un par de números de puerto, que identifican a un programa dentro de cada computadora.
- Los sockets permiten implementar una arquitectura cliente-servidor.
- Proceso o hilo existente en la máquina cliente y en la máquina servidor que permiten leer y escribir información. Esta información será la transmitida por las diferentes capas de red.

# Sockets. Creación de sockets

- TCP vs UDP :



# Sockets. Creación de sockets

- Java TCP :
  - `java.net.Socket`
  - `java.net.ServerSocket`
  - `java.net.SocketImpl`
    - En las nuevas versiones de Java, se considera código *legacy*.
    - JDK Enhanced Proposal: [JEP 353](#)
  - `java.net.URL`
    - Internamente utiliza sockets

# Sockets. Creación de sockets

- Java UDP
  - DatagramSocket
  - DatagramSocketImpl
  - DatagramPacket

# Ejercicio

- Utilizando el API de la clase URL
  - Haz una llamada a una web
  - Imprime en pantalla el contenido que retorne la página web
  - Imprime en pantalla la información que nos aporta la clase `java.net.URL`
  - Web de ejemplo: *<http://www.google.com>*



# Comunicaciones en Red

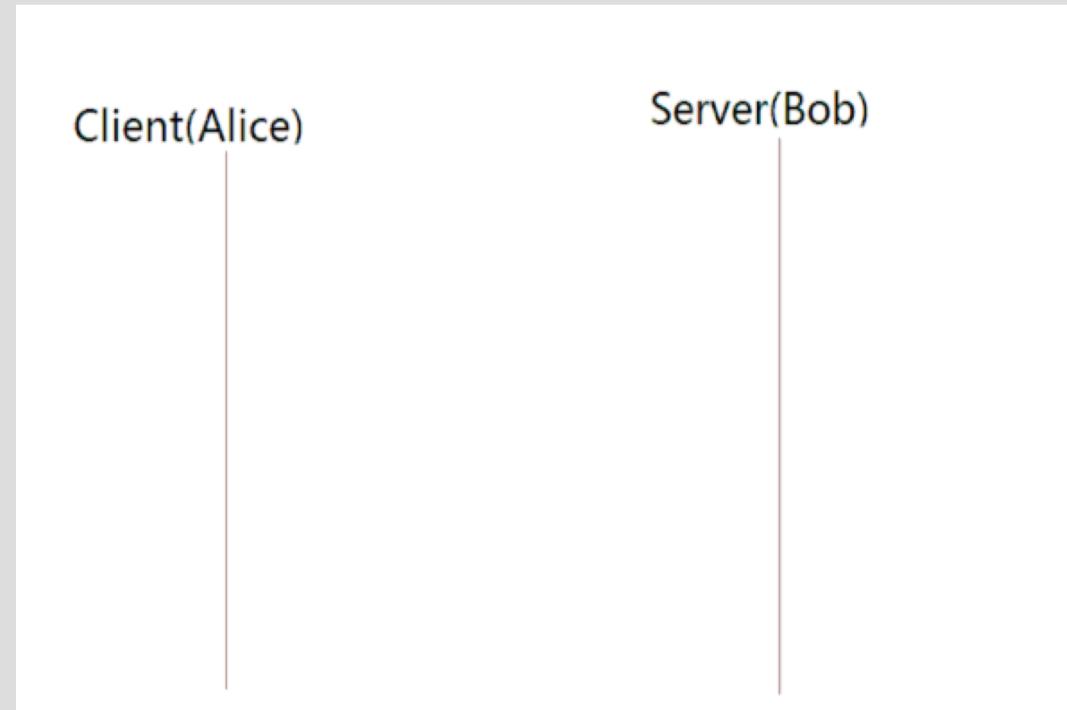
- Protocolos de comunicaciones
- Comunicación entre aplicaciones
- Roles cliente – servidor
- Puertos de comunicaciones.
- Sockets. Creación de sockets.
- **Enlazado y establecimiento de conexiones.**
- Sockets para transmitir y recibir información.
- Programación de aplicaciones cliente y servidor
- Utilización de hilos en aplicaciones en red.
- Depuración. Monitorización de tiempos de respuesta

# Enlazado y establecimiento de conexiones

- Sobre TCP:
  - Son orientados a la conexión.
    - Que primero deben establecer una conexión
  - Se garantiza la transmisión de todos los octetos sin errores ni omisiones.
  - Se garantiza que todo octeto llegará a su destino en el mismo orden en que se ha transmitido.
- Sobre UDP
  - No orientado a conexión, sin garantía de entrega
  - No se garantiza que lleguen todos los mensajes en el mismo orden en el que se mandaron.

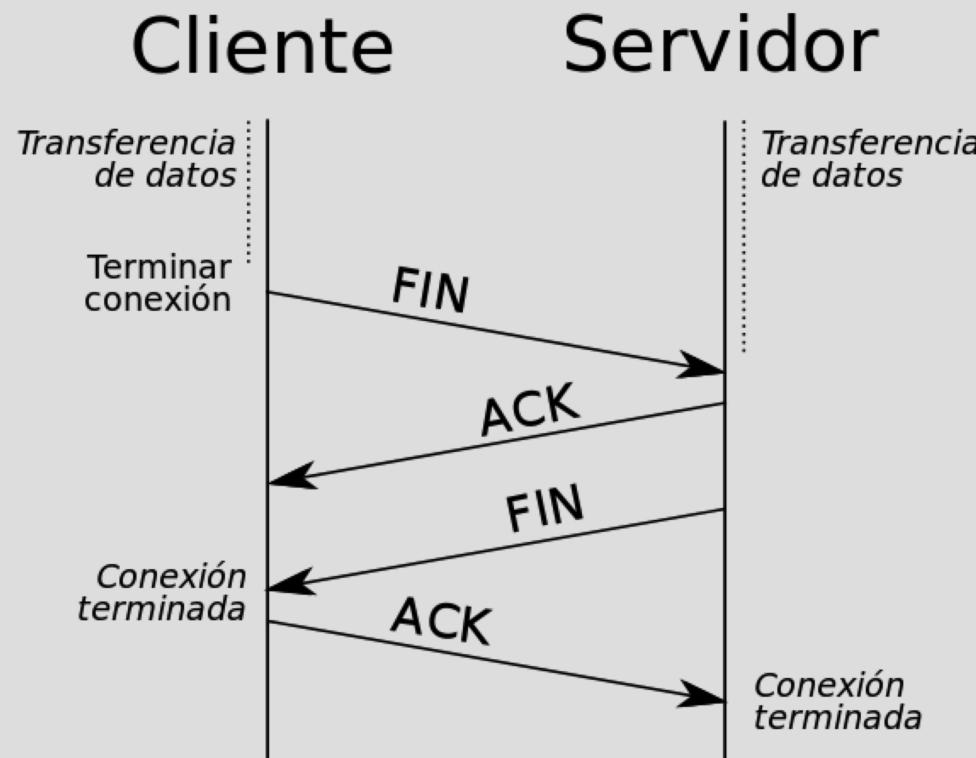
# Enlazado y establecimiento de conexiones

- Establecimiento de conexión:



# Enlazado y establecimiento de conexiones

- Finalización de la conexión



# Enlazado y establecimiento de conexiones

- Javadoc Socket: connect

## connect

```
public void connect(SocketAddress endpoint)
                     throws IOException
```

Connects this socket to the server.

### Parameters:

endpoint - the SocketAddress

### Throws:

IOException - if an error occurs during the connection

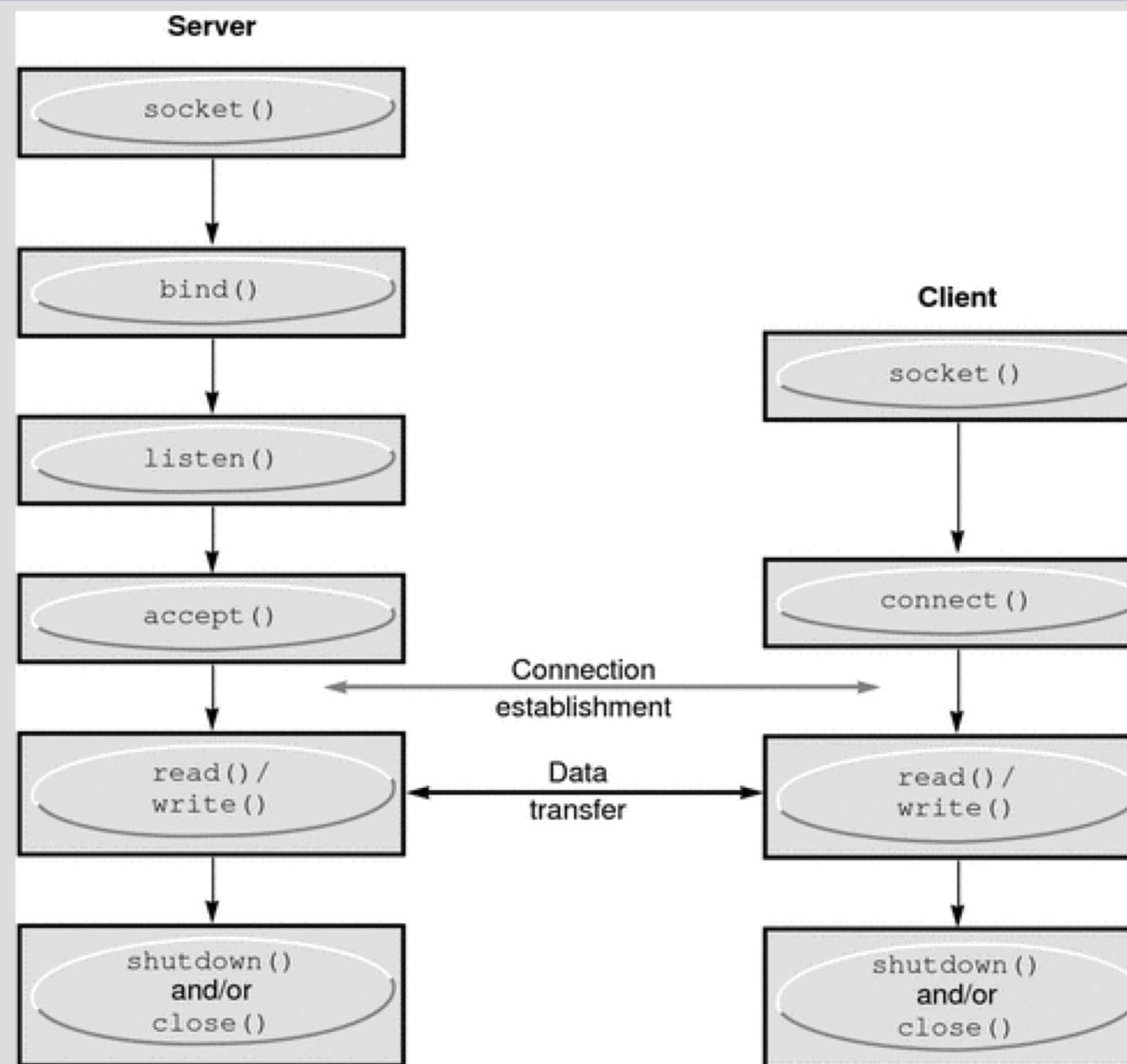
IllegalBlockingModeException - if this socket has an associated channel, and the channel is in non-blocking mode

IllegalArgumentException - if endpoint is null or is a SocketAddress subclass not supported by this socket

# Comunicaciones en Red

- Protocolos de comunicaciones
- Comunicación entre aplicaciones
- Roles cliente – servidor
- Puertos de comunicaciones.
- Sockets. Creación de sockets.
- Enlazado y establecimiento de conexiones.
- **Sockets para transmitir y recibir información.**
- Programación de aplicaciones cliente y servidor
- Utilización de hilos en aplicaciones en red.
- Depuración. Monitorización de tiempos de respuesta

# Sockets para transmitir y enviar información



# Comunicaciones en Red

- Protocolos de comunicaciones
- Comunicación entre aplicaciones
- Roles cliente – servidor
- Puertos de comunicaciones.
- Sockets. Creación de sockets.
- Enlazado y establecimiento de conexiones.
- Sockets para transmitir y recibir información.
- **Programación de aplicaciones cliente y servidor**
- Utilización de hilos en aplicaciones en red.
- Depuración. Monitorización de tiempos de respuesta

# Sockets. Cliente-Servidor

- Socket servidor
  - Arranca en un puerto definido
  - Envía mensajes por el outputStream

```
try {  
    ServerSocket serverSocket = new ServerSocket(port);  
    Socket socket = serverSocket.accept();  
    OutputStream outPutStream = socket.getOutputStream();  
    DataOutputStream dataOutputStream = new DataOutputStream(outPutStream);  
    dataOutputStream.writeUTF( str: "La fecha actual es :" + Instant.now().toString());  
} catch (IOException e) {  
    System.out.println("ERROR "+e);  
}
```

# Sockets. Cliente-Servidor

- Socket cliente: conectarse al Server y leer su mensaje

```
Socket client = new Socket();
SocketAddress address = new InetSocketAddress(hostname, port);

try {
    client.connect(address);
    InputStream is = client.getInputStream();

    //Flujos que manejan caracteres
    InputStreamReader isr = new InputStreamReader(is);
    int c;
    while((c = isr.read()) != -1){
        System.out.print((char)c);
    }

    isr.close();
    is.close();
} catch (IOException e) {
    System.out.println("Error "+e);
} finally {
    client.close();
}
```

# Sockets. Cliente-Servidor

- Crea dos sockets que
  - Uno haga de servidor y
  - El otro haga de cliente
  - El servidor envía 10 mensajes y el cliente los escribe por pantalla
  - Ahora hazlo sobre el protocolo UDP



# Sockets. Cliente-Servidor

- Partiendo del ejercicio anterior
  - El envío de mensajes deja de ser unidireccional, ahora el cliente va a responder al servidor
  - Una vez que el cliente reciba un mensaje con la fecha actual, él va enviarle otro mensaje al servidor dándole las gracias

“¡Gracias, Server!”



# Comunicaciones en Red

- Protocolos de comunicaciones
- Comunicación entre aplicaciones
- Roles cliente – servidor
- Puertos de comunicaciones.
- Sockets. Creación de sockets.
- Enlazado y establecimiento de conexiones.
- Sockets para transmitir y recibir información.
- Programación de aplicaciones cliente y servidor
- **Utilización de hilos en aplicaciones en red.**
- Depuración. Monitorización de tiempos de respuesta

# Multihilo en aplicaciones en red

- Un socket que haga de servidor necesitará soportar recibir varias llamadas concurrentes: multihilo
  - Cada cliente un hilo
- Establecer política al alcanzar el número máximo de hilos:
  - Rechazar nuevas conexiones
  - Dejarlas a la espera
  - Expulsar una de las conexiones existentes

# Multihilo en aplicaciones en red

- Partiendo del ejemplo del socket que retornaba la hora actual:
  - Haz las modificaciones oportunas para ser capaz de recibir hasta 10 clientes concurrentes
  - ¿Qué ocurre si hay 11 clientes haciendo llamadas concurrentes?



# Comunicaciones en Red

- Protocolos de comunicaciones
- Comunicación entre aplicaciones
- Roles cliente – servidor
- Puertos de comunicaciones.
- Sockets. Creación de sockets.
- Enlazado y establecimiento de conexiones.
- Sockets para transmitir y recibir información.
- Programación de aplicaciones cliente y servidor
- Utilización de hilos en aplicaciones en red.
- **Depuración. Monitorización de tiempos de respuesta**

# Monitorear tiempos de respuesta

- Controlar el rendimiento de nuestras aplicaciones
- Aún más necesario en sistemas distribuidos
- No debe influir en la propia aplicación
- Control de
  - Tiempos de respuesta
  - Uso de recursos: memoria, CPU
  - Latencias
- Objetivo: encontrar debilidades del software
  - Cuellos de botella → puntos de mejora

# Monitorear tiempos de respuesta

- Java Mission Control
  - Recoge datos de las optimizaciones dinámicas propias de la JVM
  - Elimina el “efecto observador”
  - Interfaz gráfica: JMC Client (`JAVA_HOME\bin\jmc`)
  - Ejecución por línea de comandos:  
`jmc`
    - Opción para monitorear en remoto
- Java Flight Recorder
  - Integrado en la JMC

# Monitorear tiempos de respuesta

- Java Mission Control
  - JVM Browser
    - Aplicaciones corriendo en cada JVM
  - JMX Console
    - Datos en tiempo real
    - Se pueden crear alertas (aviso si memoria > 80%)
  - Java Flight Recorder
    - Recopila la información de una aplicación en ejecución para un análisis posterior

JVM Browser Event Types

[1.8.0\_131] The JVM Running Mission Control  
MBean Server  
Flight Recorder

[1.8.0\_131] org.jetbrains.jps.cmdline.Launcher /  
MBean Server  
Flight Recorder

[1.8.0\_131] org.jetbrains.kotlin.daemon.KotlinCor  
MBean Server  
Flight Recorder

[1.8.0\_152-release] -Xms128m (364)  
MBean Server  
Flight Recorder

[1.8.0\_152-release] org.jetbrains.idea.maven.ser  
MBean Server  
Flight Recorder

[1.8.0\_152-release] org.jetbrains.idea.maven.server.RemoteMavenServer (772)

## Overview

Historical Data Settings

Dashboard

Used Java Heap Memory: Now: 23.3 MiB Max: 42.8 MiB

JVM CPU Usage: Now: 0.15 % Max: 18.5 %

Live Set + Fragmentation: No value yet

Processor

Processor Usage: JVM CPU Usage (blue line) and Machine CPU Usage (dark blue line). The chart shows usage fluctuating between 10% and 40% over the period from 7:04:08 PM to 7:05:00.

Memory

Memory Usage: Committed Java Heap (red line), Maximum Java Heap (blue line), Total Physical Memory (green line), Used Java Heap Memory (dark red line), and Used Physical Memory (orange line). The Used Java Heap Memory shows a significant drop from approximately 35 MiB to 16 MiB around 7:04:10 PM.

Overview MBean Browser Triggers System Memory Threads Diagnostic Commands