

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.
- Utilización de sockets para la transmisión y recepción de 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 de red
 - IP (IPv4, IPv6)
 - TCP, UDP
 - RPC, SSL
 - SMTP, FTP, SSH, HTTP, Telnet, LDAP
 - ...

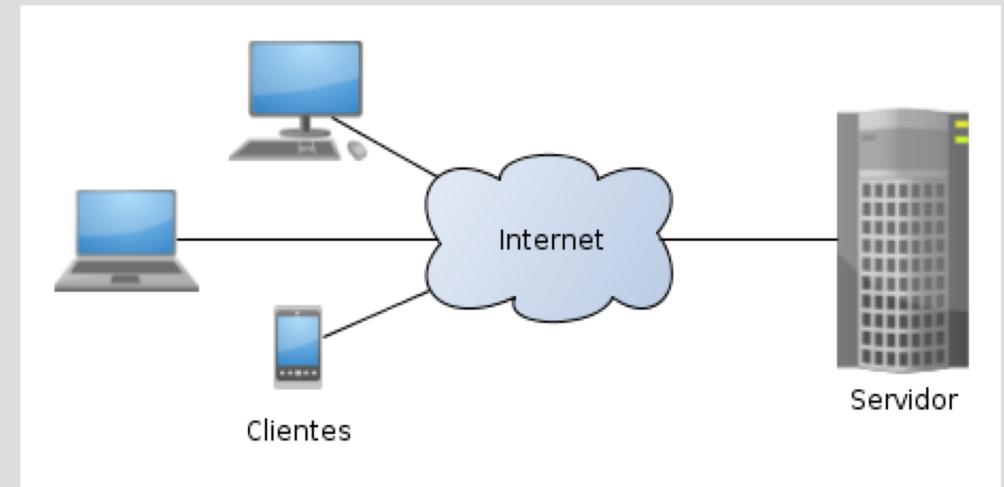


Comunicación entre aplicaciones

- Software síncrono
 - 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

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.
- Utilización de sockets para la transmisión y recepción de 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.

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

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

Sockets. Creación de sockets

- 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. Creación de sockets

- 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();
}
```