Comunicación entre procesos

El desarrollo de aplicaciones concurrentes involucra el uso de mecanismos de comunicación. Estos mecanismos, conocidos genéricamente como **mecanismos de comunicación entre procesos** (IPC) permiten que los distintos procesos que conforman una aplicación "hablen entre sí".

Los procesos en los que se descompone una aplicación pueden ejecutarse en un mismo ordenador (siempre que tengamos un sistema operativo multitarea) o en máquinas diferentes. En este último caso, la comunicación entre procesos involucra el uso de **redes de ordenadores**.

Mecanismos de comunicación entre procesos [IPC: InterProcess Communication]

Cuando en un sistema tenemos distintos procesos, necesitamos disponer de mecanismos que hagan posible la comunicación entre ellos. Se pueden utilizar distintos mecanismos de comunicación entre procesos:

- *Pipes anónimos* [*Anonymous pipes*]: Permiten redireccionar la entrada o salida estándar de un proceso (utilizando | en la línea de comandos, por ejemplo).

- *Sockets*: Usan la familia de protocolos TCP/IP (la que se utiliza en Internet). Su diseño original proviene del BSD UNIX [Berkeley Software Distribution].

- *Estándares de paso de mensajes* como MPI (Message Passing Interface, muy utilizado en clusters y supercomputadores) o PVM (Parallel Virtual Machina, otro estándar utilizado en multiprocesadores y multicomputadores).

- *Llamadas a procedimientos remotos* (RPC: Remote Procedure Call): Permiten realizar la comunicación entre procesos como si se tratase de simples llamadas a funciones.
  - En Java, se utiliza un mecanismo conocido como *RMI* [*Remote Method Invocation*].
  - En la plataforma .NET, se utiliza *.NET Remoting* (un mecanismo similar a RMI)
  - El RPC de Windows cumple con el estándar OSF DCE [Open Software Foundation Distributed Computing Environment], lo que permite la comunicación entre procesos que se ejecuten en sistemas operativos diferentes a Windows.

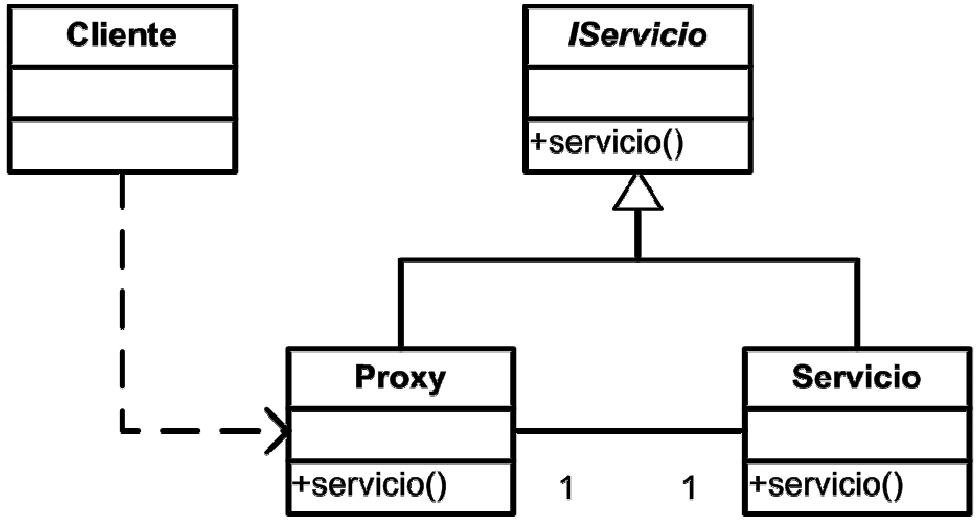
- *Middleware*: Software que se utiliza para conectar los componentes de un sistema distribuido.
  - **CORBA** [Common Object Request Broker Architecture], estándar del OMG [Object Management Group].
  - **Servicios web** [Web services], promovidos por el W3C (el consorcio que propone los estándares usados en la web).
  - **J2EE** [Java 2 Enterprise Edition] estandariza los servicios que ha de ofrecer un servidor de aplicaciones Java. © Sun Microsystems
  - **COM / DCOM** [Component Object Model / Distributed COM] establece un estándar binario mediante el cual se puede acceder a los servicios ofrecidos por un componente. © Microsoft Corporation

Como se puede ver, disponemos de una amplia variedad de mecanismos de comunicación entre procesos (y eso sin contar los múltiples mecanismos específicos que ofrece cada sistema operativo).

En el caso particular de **Windows**: el *portapapeles*, *DDE* [*Dynamic Data Exchange*], *OLE* [*Object Linking and Embedding*], *ActiveX*, *Mailslots* (en Win32 y OS/2), el mensaje *WM_COPYDATA*, *ficheros mapeados en memoria*, *pipes con nombre* [*Named pipes*], semáforos, eventos, “mutex” y otras primitivas de sincronización…

Independientemente del mecanismo de comunicación entre procesos que decidamos emplear, nuestra aplicación debería acceder a los recursos externos de la misma forma que accede a recursos locales.

Para encapsular el acceso a recursos externos se suelen emplear proxies o gateways (para que en nuestra aplicación se pueda cambiar el mecanismo de comunicación entre procesos con el menor esfuerzo posible):



Siempre debemos tener en cuenta que…

- Los mecanismos de comunicación no siempre son fiables (algunos paquetes se pierden)
- La comunicación entre procesos consume tiempo (la latencia no es cero)
- La capacidad del canal de comunicación no es infinita (el ancho de banda es un recurso muy valioso)
- Las comunicaciones no siempre se realizan a través de medios seguros.

TERMINOLOGÍA: Los procesos de una aplicación distribuida suelen clasificarse como clientes o servidores, si bien pueden desempeñar ambos roles en distintos momentos. El **cliente** es el que solicita algún servicio proporcionado por otro proceso. El **servidor** es el que atiende las peticiones de los clientes.