Comunicación entre Procesos y Sockets

Temas de la clase de hoy

- *Proceso*
- *Sockets*
- *Dominios, protocolos y tipos vinculados a los sockets*
- *Introducción a Stream y Datagram*
- *El modelo cliente-servidor*
- *Funciones del cliente*
- *Funciones del servidor*
- *Orientación a conexión versus No orientación a conexión*
- *Ejemplos y tiempo de preguntas...*

Comunicación entre Procesos y Sockets

Refrescando un concepto... **Proceso**

El concepto más importante en cualquier sistema operativo es el de Proceso, que es una abstracción de un programa en ejecución (1)

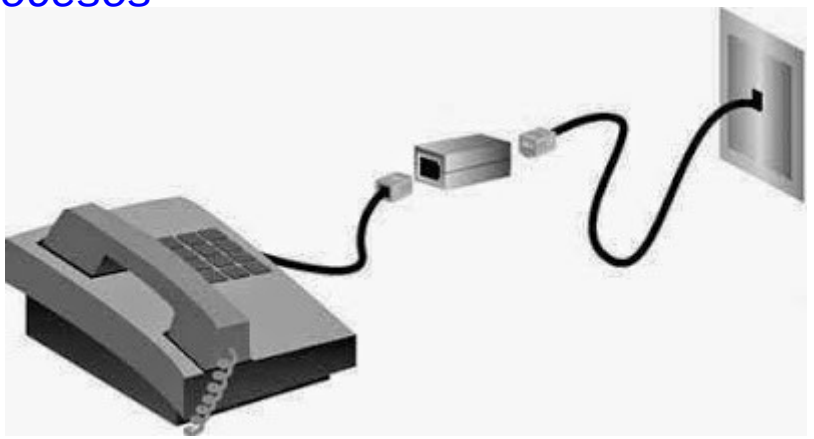
Adicionalmente, podemos decir que un Proceso es una unidad de actividad que se caracteriza por la ejecución de una secuencia de instrucciones, un estado actual, y un conjunto de recursos del sistema asociados (2)

(1) Tanenbaum, Andrew S. (2009). Sistemas operativos modernos (3 edición). Prentice Hall. p.83
(2) Stallings 5º edición p.109

Comunicación entre Procesos y Sockets

Presentando un concepto... **Socket** …¿ pero que es eso ? Definición

Los Sockets son una de las herramientas que ofrecen los Sistemas Operativos para la comunicación entre diferentes procesos



Los sockets son análogos a las bandejas de correo y los conectores de pared para los teléfonos, ya que actúan como interfaz entre los usuarios y la red, de igual forma que las bandejas de correo actúan como interfaz entre las personas y el sistema postal, y los conectores de pared de teléfono les permiten enchufar teléfonos y conectarse al sistema telefónico (1)

(1) Tanenbaum, Andrew S. (2009). Sistemas operativos modernos (3 edición). Prentice Hall. p.773

Comunicación entre Procesos y Sockets

¿ Y para que se utilizan los Sockets ?

Entre otras cosas, para...

- Traspasar información
- Ser una de las herramientas que ofrecen los Sistemas Operativos para la comunicación entre diferentes procesos
- Posibilitar la comunicación aún cuando ambos procesos estén corriendo en distintos sistemas unidos mediante una red

Adicionalmente, la *Interfaz de Programación de Aplicaciones (API)* de los **Sockets** son la base de cualquier aplicación que funcione en red, puesto que ofrece una librería de funciones básicas que el programador puede usar para desarrollar aplicaciones en red.

Desde el punto de vista de programación, un **socket** no es más que un "fichero" que se abre de una manera especial.

(1) Tanenbaum, Andrew S. (2009). Sistemas operativos modernos (3 edición). Prentice Hall. p.773

Comunicación entre Procesos y Sockets

Presentando un concepto... **Socket** …¿ pero que es eso ? Propiedades

Propiedades de los Sockets:

- Fiabilidad de la Transmisión. No se pierden los datos transmitidos
- Conservación del Orden de los Datos. Los datos llegan en el orden del Orden de los Datos.
- No Duplicación de los Datos. El Dato sólo llega una vez.
- La conexión está establecida antes de iniciar la comunicación.
- Conservación de los límites de los mensajes. Los límites de mensajes emitidos pueden encontrarse o conocerse en el destino.

Comunicación entre Procesos y Sockets

Presentando un concepto... **Socket** …¿ pero que es eso ? Atributos

Los 3 atributos de los Sockets:

- **Dominio.** Especifica el medio de comunicación de la red que el Socket utilizará.
- **Protocolo.** Especifica que protocolo se usará.
- **Tipo.** Los protocolos de internet proveen dos niveles de servicios: flujos y datagramas.

Comunicación entre Procesos y Sockets

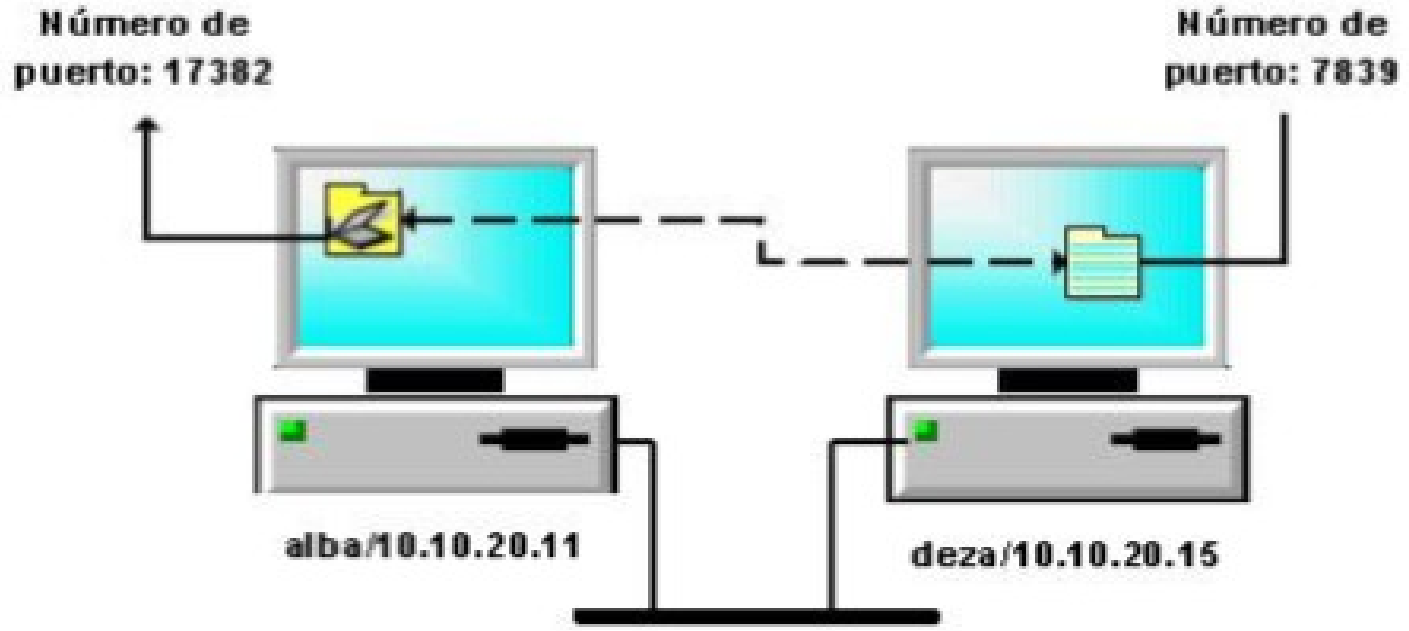
Un caso práctico de aplicación: **Sockets en una red IP**

- Los Sockets para TCP/IP permiten la comunicación de dos procesos que estén conectados a través de una red TCP/IP
- Cada máquina está identificada por medio de su dirección IP que debe ser única. Sin embargo, en cada máquina pueden estar ejecutándose múltiples procesos simultáneamente
- Cada uno de estos procesos se asocia con un número de puerto, para poder así diferenciar los distintos paquetes que reciba la máquina (proceso de multiplexación).

Comunicación entre Procesos y Sockets

Un caso práctico de aplicación: **Sockets en una red IP**

- Entonces... un Socket se identifica unívocamente por la dupla dirección IP + número de puerto.

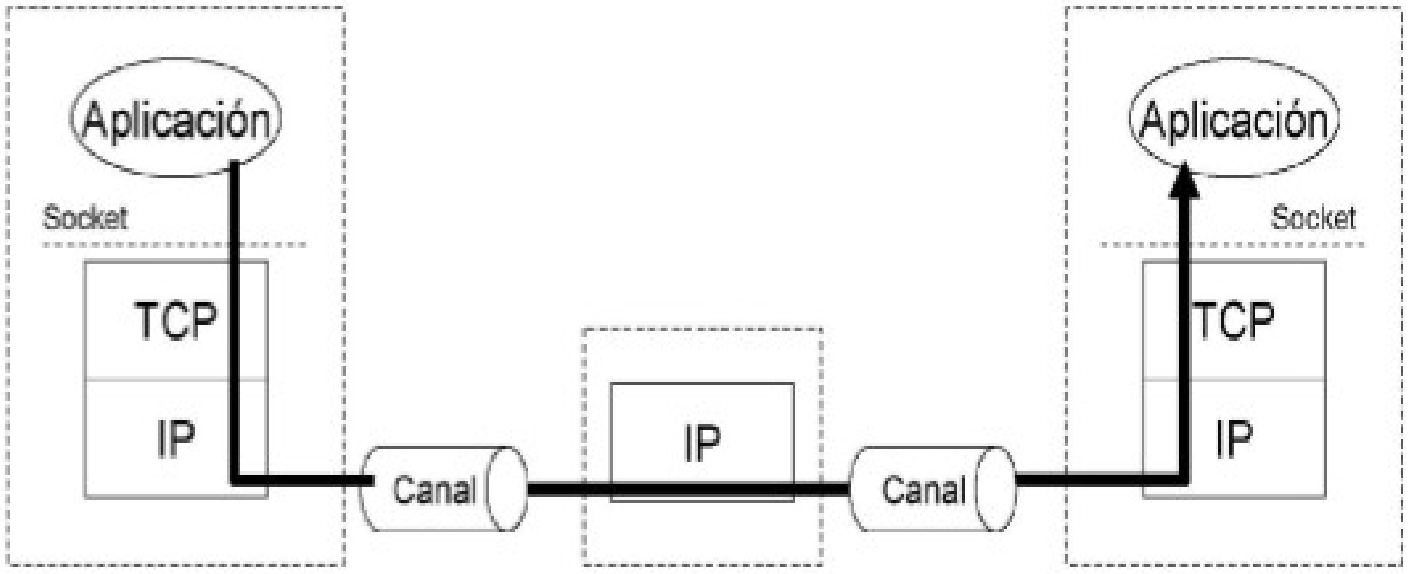


- Una comunicación entre dos procesos se identifica mediante la asociación de los Sockets que estos emplean para enviar y recibir información hacia y desde la red: Identificador de Socket origen + identificador de Socket destino

Comunicación entre Procesos y Sockets

Un caso práctico de aplicación: **Sockets en una red IP**

- Finalmente.... un Socket es una **abstracción a través de la cual una aplicación puede enviar y recibir información.**

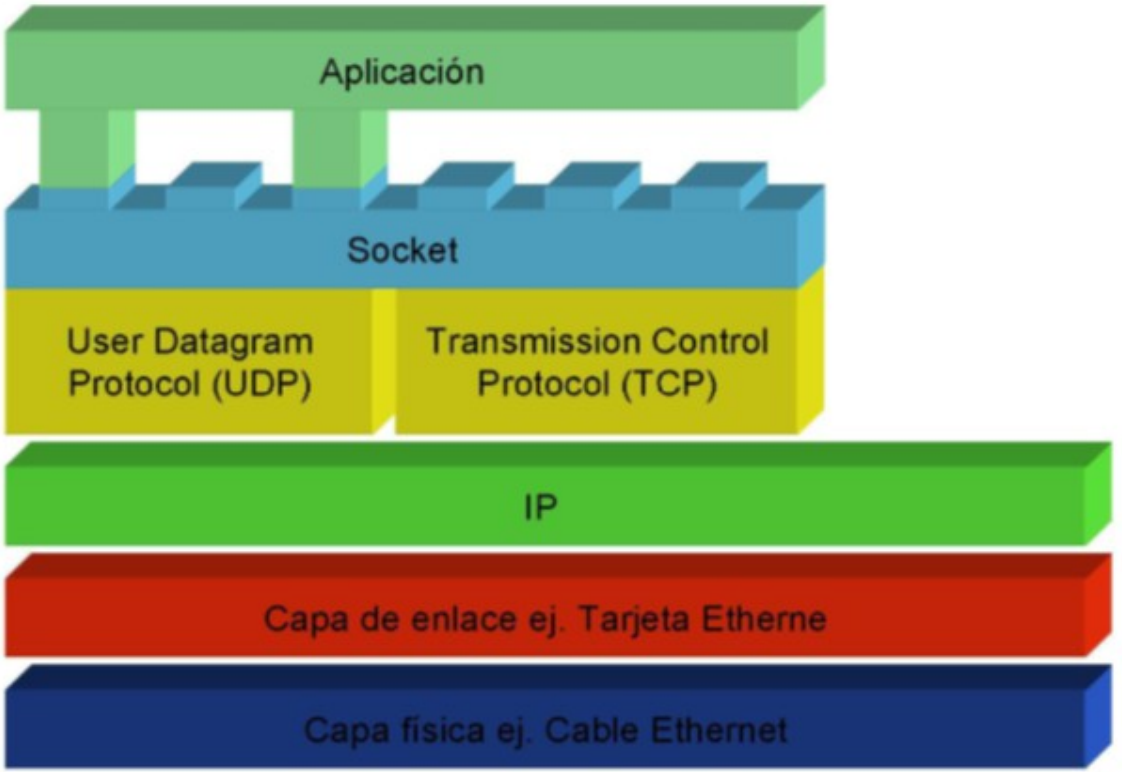


- La información que una aplicación envía por su Socket origen puede ser recibida por otra aplicación en el Socket destino y viceversa.

Comunicación entre Procesos y Sockets

Un caso práctico de aplicación: **Sockets en una red IP**

Existen diferentes tipos de Sockets dependiendo de la pila de protocolos sobre la que se cree dicho Socket. Como ejemplo nos centraremos en la pila de protocolos TCP/IP.

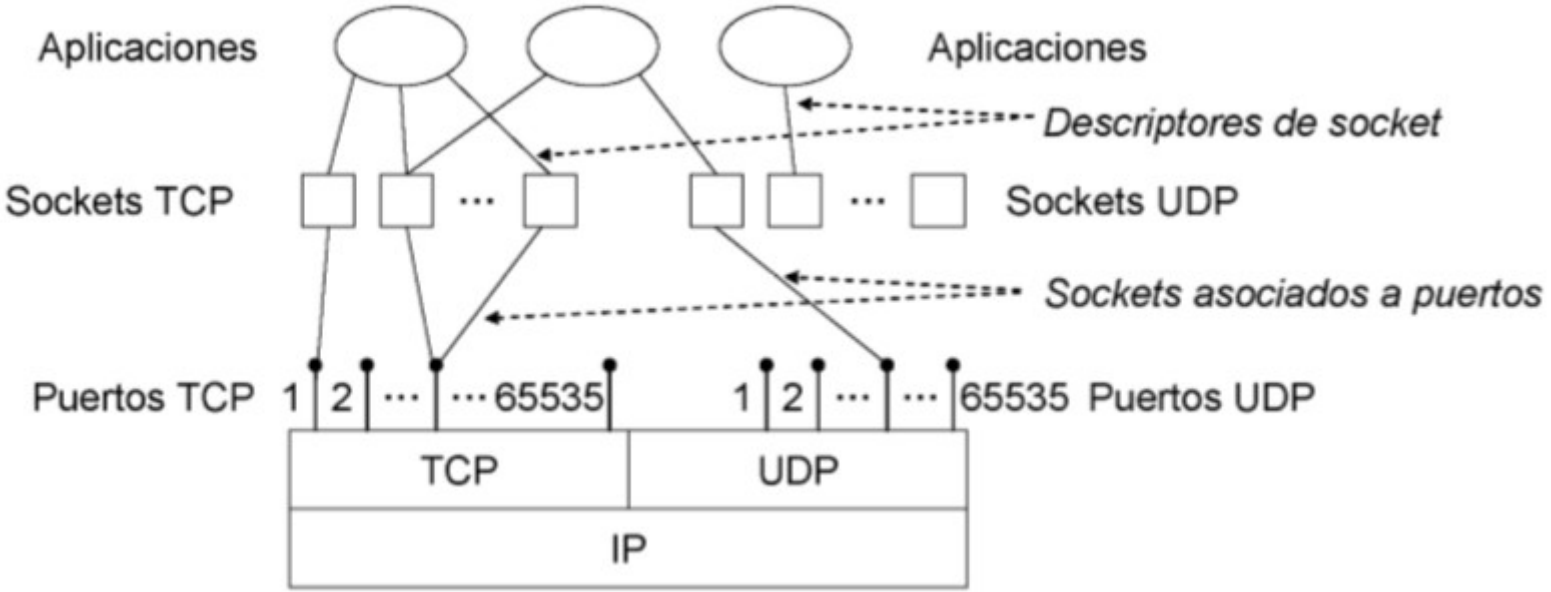


Interfaz de sockets

Comunicación entre Procesos y Sockets

Un caso práctico de aplicación: **Sockets en una red IP**

Diagrama genérico de relación lógica entre sockets, protocolos, puertos y la aplicación



- Un programa puede usar más de un Socket al mismo tiempo.
- Diferentes programas pueden usar el mismo Socket al mismo tiempo aunque esto es menos común.
- Cada Socket tiene asociado un puerto TCP o UDP, según sea el caso. Cuando se recibe un paquete dirigido a dicho puerto, este paquete se pasa a la aplicación correspondiente.

Comunicación entre Procesos y Sockets

Un caso práctico de aplicación: **Sockets en una red IP**

¿pero... como encontrar cuales son los procesos que se van a comunicar?

Concepto: Dominios de comunicación

El dominio de comunicación es una definición específica que permite identificar el lugar donde se encuentran los procesos que se van comunicar.

AF_UNIX: Sockets internos de UNIX

AF_INET: Protocolos de internet internet ARPA

AF_ISO: Protocolos estándar ISO.

AF_NS: Protocolos de redes Xerox

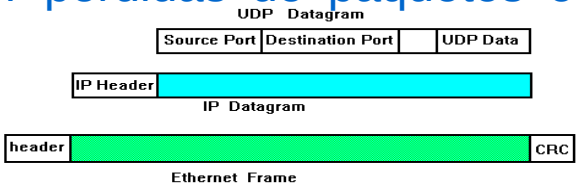
Comunicación entre Procesos y Sockets

Un caso práctico de aplicación: **Sockets en una red IP**

TIPOS de Sockets

Stream. Hace uso del protocolo **TCP** (protocolo de la capa de transporte) que provee un flujo de datos bidireccional, orientado a conexión, secuenciado, sin duplicación de paquetes y libre de errores.

Datagram. Hacen uso del protocolo **UDP** (protocolo de la capa de transporte), el cual provee un flujo de datos bidireccional, no orientado a conexión, en el cual los paquetes pueden llegar fuera de secuencia, puede haber pérdidas de paquetes o pueden llegar con errores.



Raw. Permiten un acceso a más bajo nivel, pudiendo acceder directamente al protocolo IP del nivel de Red. Su uso está mucho más limitado ya que está pensado principalmente para desarrollar nuevos protocolos de comunicación, o para obviar los protocolos del nivel de transporte.

5r5894yhd950f33333e1af8c6bc2fh98y48cc5 huffingtonpost.com 2533655546 [12/Sep/2012:10:59:59 -0400] "ts=1347461999317&h=www.huffingtonpost.com&v=14&t=Lindsay%Lohan%3A%20EWG's%20Goes%20Of%Braless%20and%20Flashes%20Major%20Sideboob%20PHOTOS%20Home%20-%20The%20Huffington%20Post&r=http%3A%2F%2Fwww.google.com%2Fm%2Fsearch%3Ftbm%3Dnws%26hl%3Den%26source%3Dmog%26gl%3Dus%26tab%3Dwn%26q%3Dhalt%2520of%2520none%2520PHOTOS%2520fort%2520sideboob%26sa%3DN&l=3327&k=1&ms=10664&ah=http%3A%2F%2Fsc.blah.htg%2Fboobs%2F2012%2Flindsay_lohan%2Flohan_sideboob_braless.pdf&at=Lohan%20braless%20of%20new%20list&cx=266&cy=6&id=me-

Comunicación entre Procesos y Sockets

Un caso práctico de aplicación: **Sockets en una red IP**

TIPOS de Sockets

Primero hay que establecer correctamente la conexión.

Los Orientados a conexión...

Se usa el protocolo **TCP** del protocolo **TCP/IP**, para gestionar la conexión.

Comunicaciones fiables

Se garantiza que todos los datos van a llegar de un programa al otro correctamente.

Circuito virtual

Se utiliza cuando la información a transmitir es importante, no se puede perder ningún dato.

Los NO orientados a conexión...

El programa de aplicación da la fiabilidad

Garantiza que los datos que lleguen son correctos, pero no garantiza que lleguen todos.

No es necesario que los programas se conecten

Es el llamado protocolo **UDP**

Cualquiera de ellos puede transmitir datos en cualquier momento, independientemente de que el otro programa esté "escuchando" o no.

Se utiliza cuando es muy importante que el programa no se quede bloqueado.

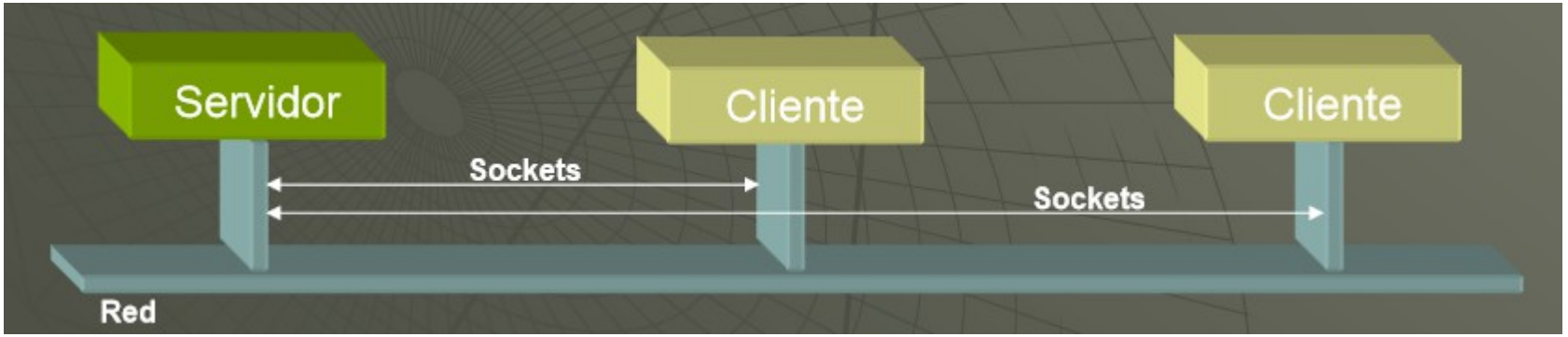
No importa que se pierdan datos.

Comunicación entre Procesos y Sockets

Concepto: Arquitectura Cliente-Servidor

Cliente: es el programa que solicita la conexión para pedir datos

Servidor: es el programa que permanece pasivo a la espera de que alguien solicite conexión con él.

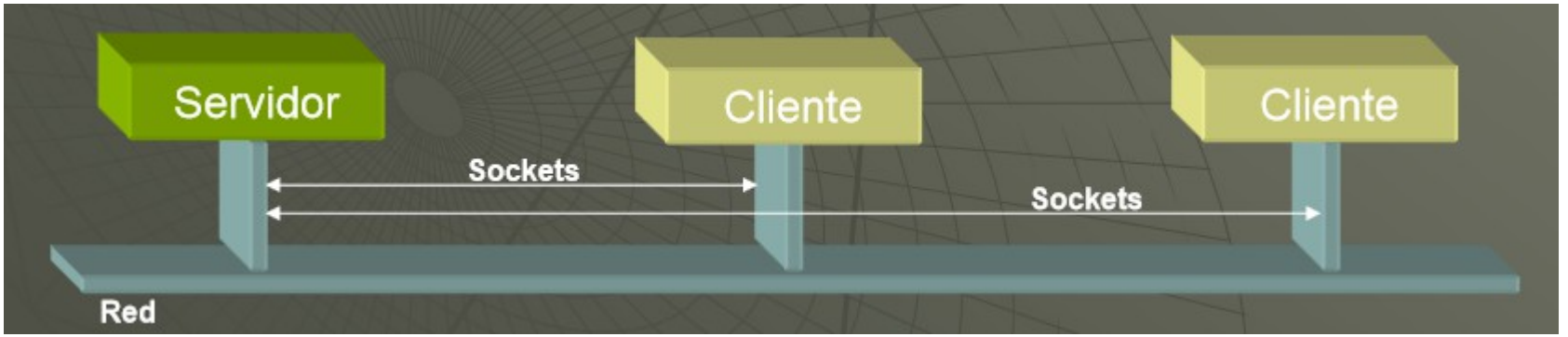


Servidor: Está ejecutándose y esperando a que otro quiera conectarse a él.
Nunca da "el primer paso" en la conexión. Es el que "sirve" información al que se la pida.

Cliente: Es el programa que da “primer paso” en la en la conexión. En el momento de ejecutarlo o cuando lo necesite, intenta conectarse al servidor. Es el que solicita información al servidor.

Comunicación entre Procesos y Sockets

Concepto: Arquitectura Cliente-Servidor

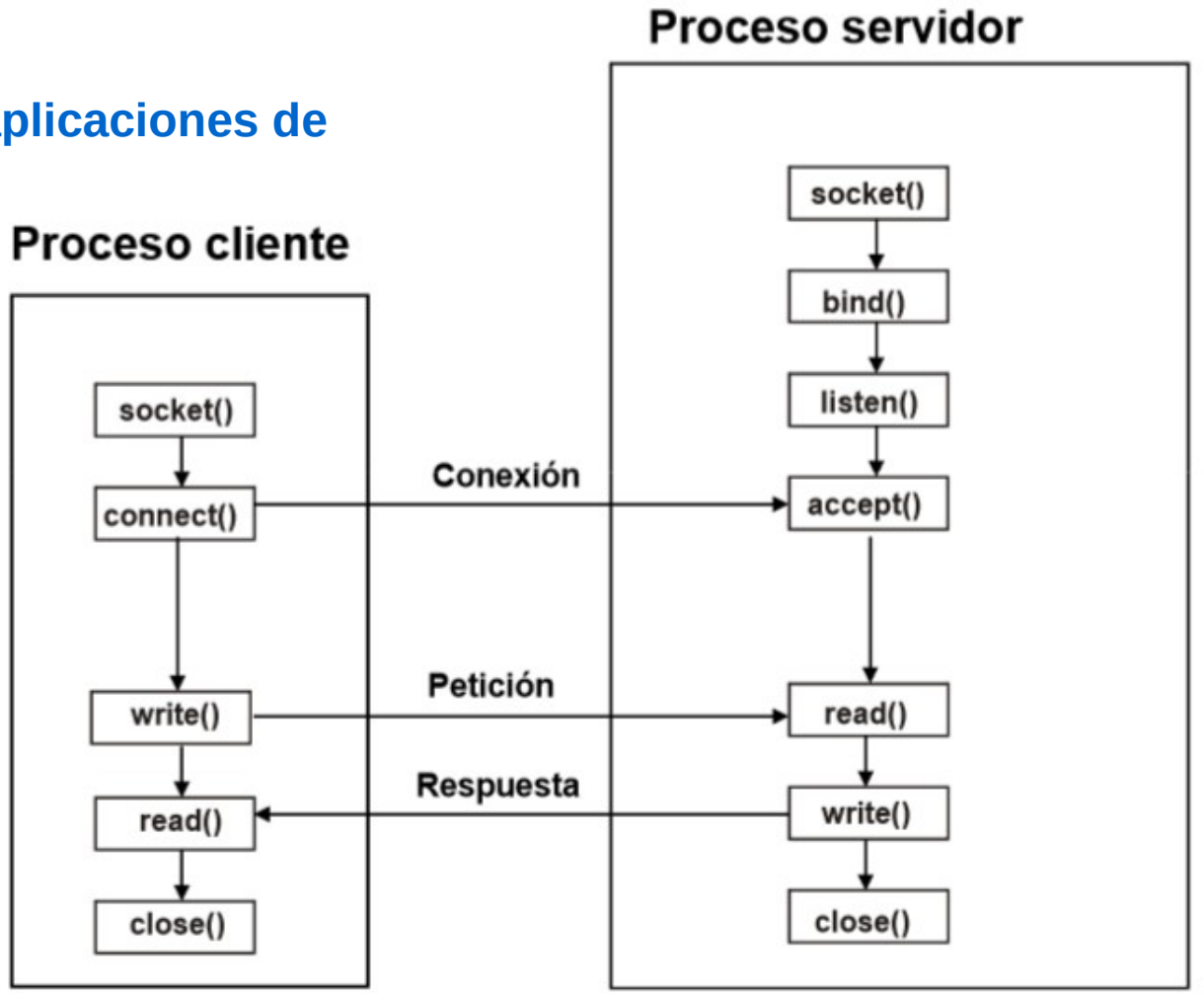


Para poder realizar la conexión entre ambos programas (cliente y servidor) es necesario conocer la dirección IP del servidor y el servicio que queremos utilizar.

Comunicación entre Procesos y Sockets

Concepto: Arquitectura Cliente-Servidor

Modelo de desarrollo de aplicaciones de red Cliente-Servidor



Comunicación entre Procesos y Sockets

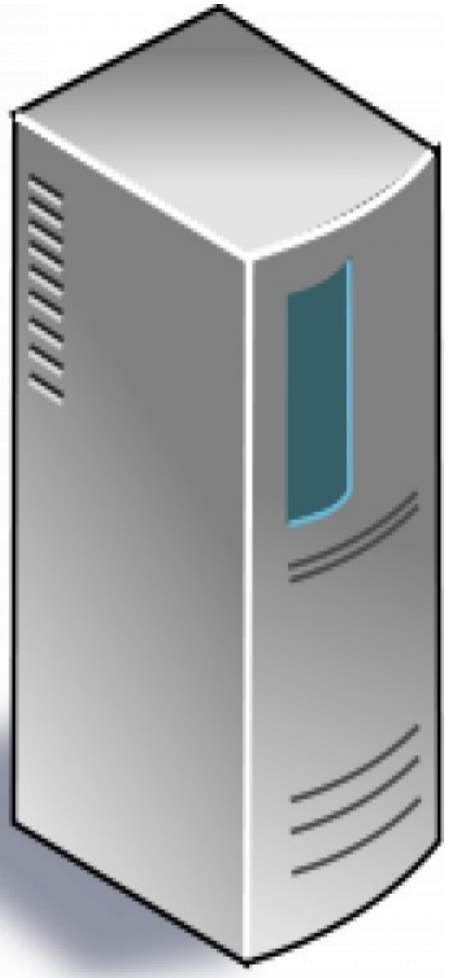
Concepto: Arquitectura Cliente-Servidor

Funciones del Servidor

Escribir y recibir datos del cliente, por medio de las funciones **write()** y **read()**, que son exactamente las mismas que usamos para escribir o leer de un fichero.

- Obviamente, tanto cliente como servidor deben saber qué datos esperan recibir, qué datos deben enviar y en qué formato.

Cierre de la comunicación y del socket, por medio de la función **close()**, que es la misma que sirve para cerrar un fichero.



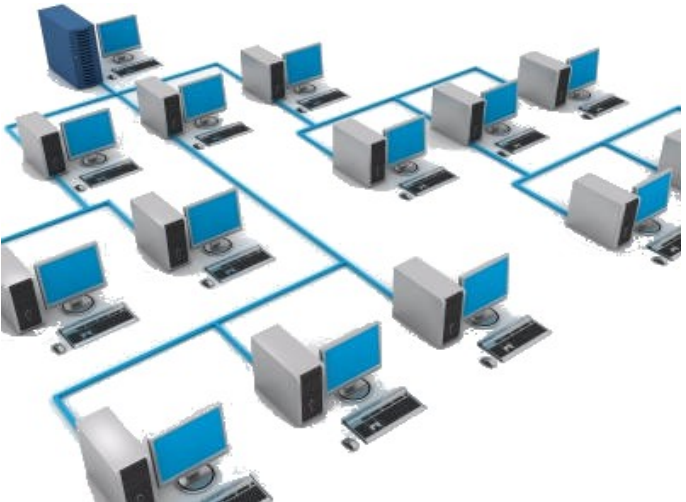
Comunicación entre Procesos y Sockets

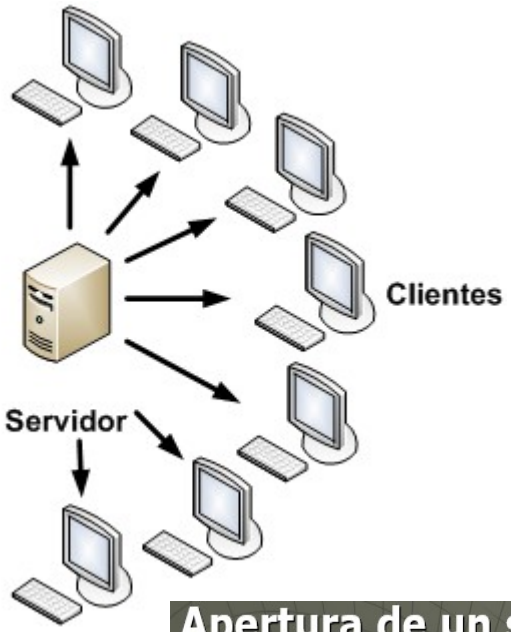
Concepto: Arquitectura Cliente-Servidor

Funciones del Cliente

int socket (int *dominio*, int *tipo*, int *protocolo*)
- crea un socket sin nombre de un dominio, tipo y protocolo específico
- *dominio* : AF_INET, AF_UNIX
- *tipo* : SOCK__DGRAM, SOCK__STREAM
- *protocolo* : 0 (protocolo por defecto)

int connect (int *dfCliente*, struct sockaddr* *direccServer*, int *longDirecc*)
- intenta conectar con un socket servidor cuya dirección se encuentra incluida en la estructura apuntada por *direccServer*.
- El descriptor *dfCliente* se utilizará para comunicar con el socket servidor.





Apertura de un socket, como el servidor, por medio de la función **socket()**

Solicitar conexión con el servidor por medio de la función **connect()**.
- Dicha función quedará bloqueada hasta que el servidor acepte nuestra conexión o bien si no hay servidor en el sitio indicado, saldrá dando un error.

Escribir y recibir datos del servidor por medio de las funciones **write()** y **read()**.

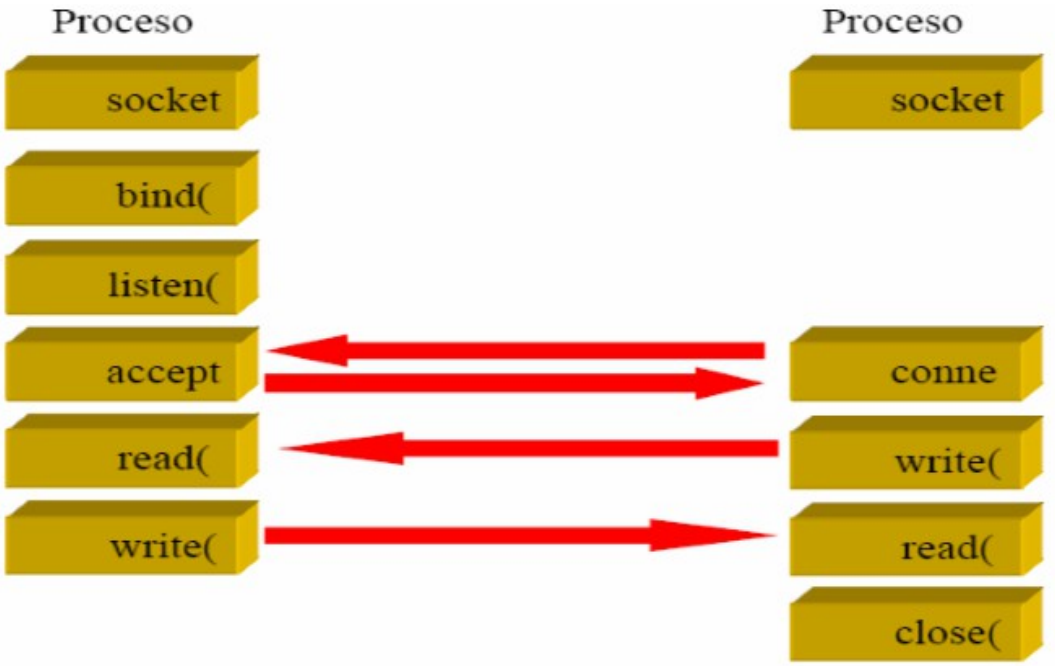
Cerrar la comunicación por medio de **close()**

Comunicación entre Procesos y Sockets

Concepto: Arquitectura Cliente-Servidor

Ejemplos de conexión: Sockets orientados a conexión (TCP)

- Orientados a “ristras” o Stream
- Permite conexión por circuito virtual
- Full duplex

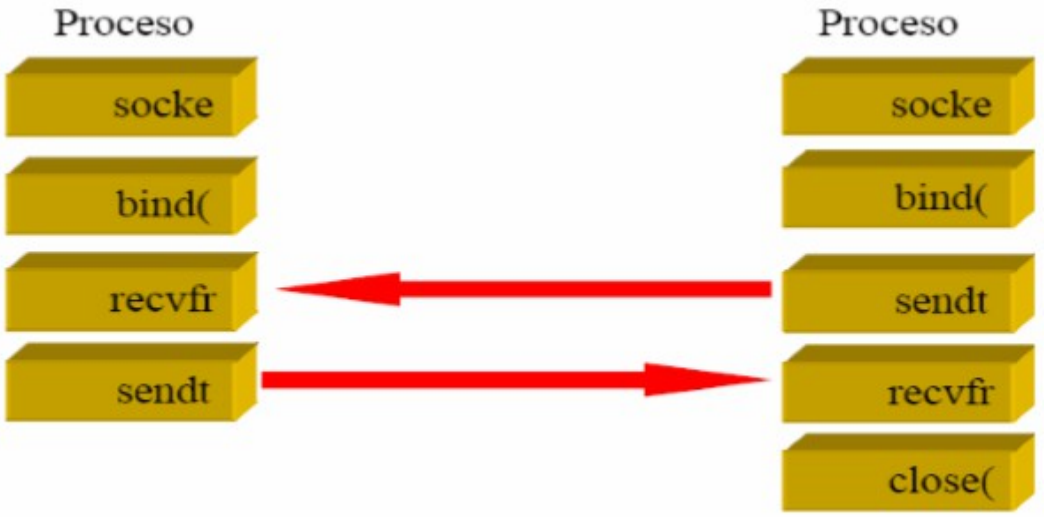


Comunicación entre Procesos y Sockets

Concepto: Arquitectura Cliente-Servidor

Ejemplos de conexión: Sockets No Orientados a conexión (UDP)

- Se conservan los límites de los mensajes
- En el dominio internet, el protocolo subyacente es el UDP
- Transmisión a nivel de paquetes que pueden tomar rutas distintas
- No se garantiza una recepción secuencial



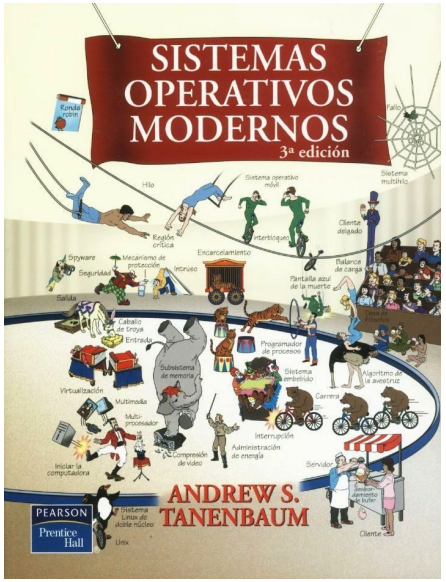
Comunicación entre Procesos y Sockets

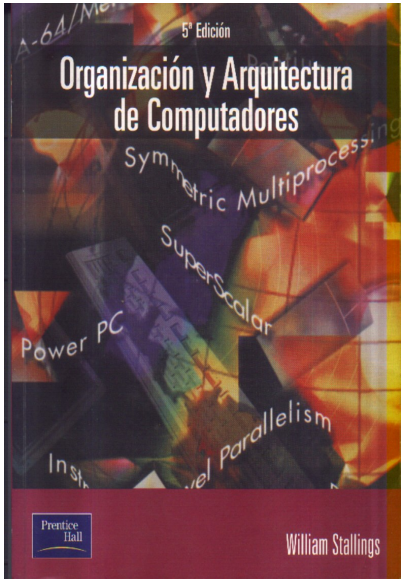
Material de consulta

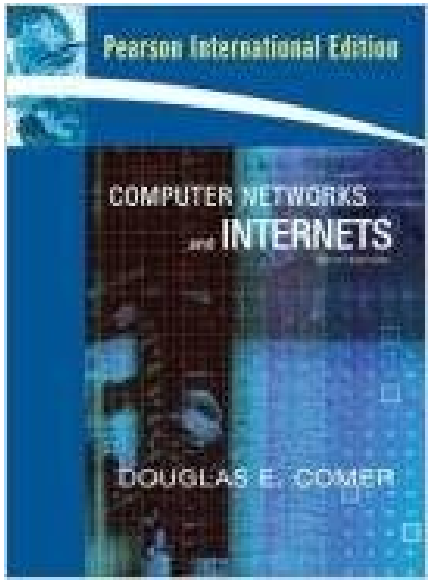
Sistemas Operativos Modernos. A. S. Tanenbaum. Prentice – Hall. 2005.

Sistemas Operativos. William Stallings. Prentice y Hall. 5º Ed. 2007

Computer Networks and Internets (5th Edition) Douglas E. Comer. Pearson IE







Comunicación entre Procesos y Sockets

Ejemplo de aplicación de un caso práctico

¿Preguntas?