Sockets

ARISO II - ETSETB

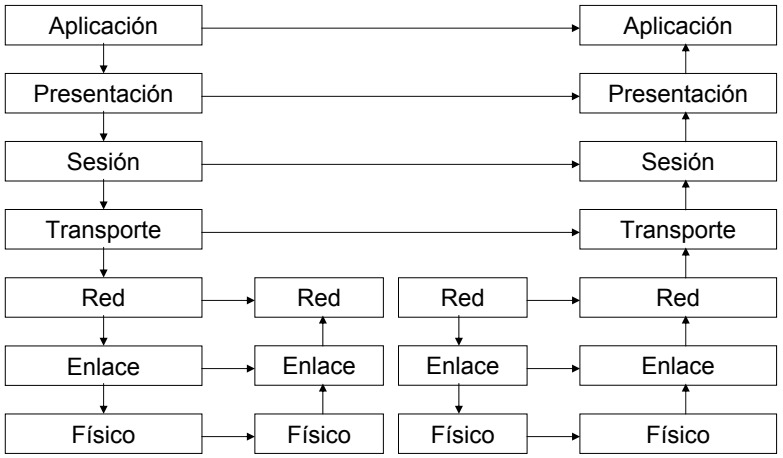
Modelo OSI

- Norma editada por ISO (International Organisation for Standarisation)
- Norma ISO-7494
- No garantiza la comunicación entre equipos pero pone las bases para una mejor estructuración de los protocolos
- La familia de protocolos que mejor se ajusta al modelo es TCP/IP
- El modelo que define la norma se estructura en siete niveles

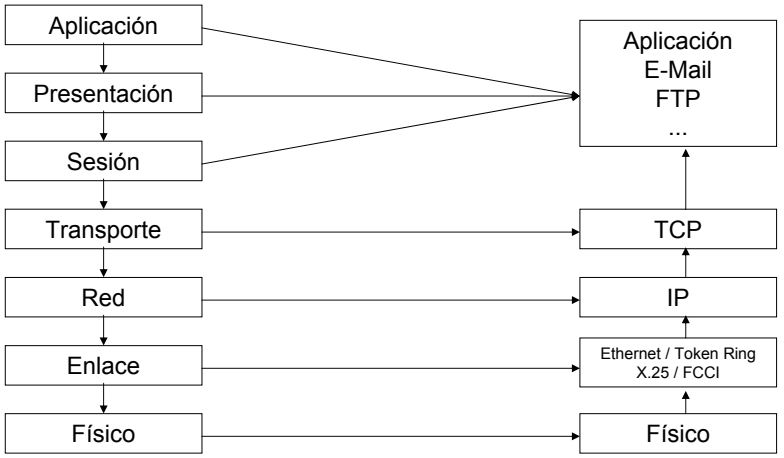
Modelo OSI

Nivel	Nombre	Función	Dispositivo y protocolo
1	Físico	Se ocupa de la transmisión del flujo de bits a través del medio	Cables, tarjetas, y repetidores (hub) RS-232, X.21, ...
2	Enlace	Divide el fujo de bits en unidades con formato (tramas) intercambiando estas unidades mediante el empleo de protocolos	Puentes (bridges). HDLC y LLC
3	Red	Establece las comunicaciones y determina el camino que tomarán los datos en la red	Encaminador (router). IP, IPX
4	Transporte	Asegura que el receptor reciba exactamente la misma información que envía el emisor	Pasarela (gateway). UDP, TCP, SPX
5	Sesión	Establece la comunicación entre las aplicaciones, las mantiene y las finaliza en el momento adecuado. Permite entrar en un sistema usando otro. Permite a un usuario mantener diferentes sesiones.	Pasarela
6	Presentación	Conversión entre diferentes representaciones de datos.	Pasarela. Compresión, encriptado, VT100
7	Aplicación	Servicios que ofrece el nivel de aplicación para poder realizar el trabajo que se le ha encomendado	X.400, ...

Modelo OSI



Modelo OSI / Internet



Modelo OSI / Internet

	TCP/IP	OSI
(NFS)	5. Aplicación	7. Aplicación
(XDR)		6. Representación
(RPC)		5. Sesión
(TCP/UDP)	4. Transporte	4. Transporte
(IP/ICMP)	3. Internet	3. Red
Trama Ethernet	2. Interfaz de red	2. Enlace de datos
Red Ethernet	1. Hardware	1. Físico

Internet

- Nivel Internet
  - Controla el direccionamiento de la información
  - IP: Se trata de un protocolo para el intercambio de datagramas en modo no conectado
  - No garantiza la llegada del mensaje (eso es función del nivel superior)
  - Se basa en tablas de direccionamiento (difundidos por los gateways)
- Nivel Transporte
  - Incluye el Protocolo de Control de Transporte (TCP) y el Protocolo de Datagrama de Usuario (UDP)
  - TCP es un protocolo orientado a conexión (transporte seguro de paquetes en modo duplex)
  - Utiliza el mecanismo PUERTO de E/S
  - Existen puertos reservados
    - ECHO = 7 / FTP = 21 / TELNET = 23 / ...
  - En Unix están reservados los números de puerto inferiores a 1024
- Nivel de Aplicación
  - Ejemplo: formato de trama telnet

Dir. Ethernet	IP	TCP	Telnetd
/etc/host	/etc/protocols	/etc/services	Inetd.conf

Arquitectura y direccionamiento

- Se otorga a los hosts implicados un dirección lógica que se compone de:
  - Dirección de red
  - Dirección de la máquina dentro de la red
- La dirección ocupa 32 bits, en 4 octetos
  - n1.n2.n3.n4
  - Cada campo es un valor entre 0 y 255
  - La dirección nula se refiere a la red
  - El valor 127 en el primer campo se llama *loopback*
- Redes
  - Tipo A: (gran tamaño): n1.0.0.0
  - Tipo B: (tamaño medio): n1.n2.0.0
  - Tipo C: (pequeñas): n1.n2.n3.0
- Para simplificar se usan direcciones simbólicas (nombres)

SOCKETS

Los *sockets* son mecanismos de comunicación entre procesos que permiten que un proceso hable (emita o reciba información) con otro proceso incluso estando en distintas máquinas.

Un **socket** es al sistema de comunicación entre ordenadores lo que un buzón o un teléfono es al sistema de comunicación entre personas: un punto de comunicación entre dos agentes (procesos o personas respectivamente) por el cual se puede emitir o recibir información.

La forma de referenciar un socket por los procesos implicados es mediante un **descriptor** del mismo tipo que el utilizado para referenciar ficheros.

Se podrá realizar redirecciones de los archivos de E/S estándar (descriptores 0,1 y 2) a los sockets y así combinar entre ellos aplicaciones de la red.

Todo nuevo proceso creado heredará, por tanto, los descriptores de sockets de su padre.

SOCKETS

La comunicación entre procesos a través de sockets se basa en la filosofía **CLIENTE-SERVIDOR**:

Un proceso en esta comunicación actuará de **proceso servidor** creando un socket cuyo nombre conocerá el **proceso cliente**, que podrá "hablar" con el proceso servidor a través de la conexión con dicho **socket**

El otro proceso actuará como **cliente** creando un socket sin nombre cuyo descriptor usará para leer o escribir

El enlace entre los dos sockets permite una comunicación **bidireccional**, característica propia de los sockets y que los diferencia de los **pipes**, o canales de comunicación unidireccional entre procesos de una misma máquina.

SOCKETS

El mecanismo de comunicación vía sockets tiene los siguientes pasos:

1º) El proceso servidor crea un socket con nombre y espera la conexión.

2º) El proceso cliente crea un socket sin nombre.

3º) El proceso cliente realiza una petición de conexión al socket servidor.

4º) El cliente realiza la conexión a través de su socket mientras el proceso servidor mantiene el socket servidor original con nombre.

SOCKETS

Es muy común en este tipo de comunicación lanzar un proceso hijo, una vez realizada la conexión, que se ocupe del intercambio de información con el proceso cliente mientras el proceso padre servidor sigue aceptando conexiones.

*Todo socket viene definido por dos características fundamentales:

- El **tipo** del socket, que indica la naturaleza del mismo, el tipo de comunicación que puede generarse entre los sockets.

- El **dominio** del socket especifica el conjunto de sockets que pueden establecer una comunicación con el mismo.

Tipos de SOCKETS

Define las **propiedades** de las comunicaciones en las que se ve envuelto un socket, esto es, el tipo de comunicación que se puede dar entre cliente y servidor.

Estas pueden ser:

- Fiabilidad de transmisión.
- Mantenimiento del orden de los datos.
- No duplicación de los datos.
- El "Modo Conectado" en la comunicación.
- Envío de mensajes urgentes

Tipos de SOCKETS

Los tipos disponibles son los siguientes:

* Tipo **SOCK_DGRAM**: sockets para comunicaciones en **modo no conectado**, con envío de **datagramas** de tamaño **limitado** (tipo telegrama). En dominios Internet el protocolo del nivel de transporte sobre el que se basa es el **UDP**.

* Tipo **SOCK_STREAM**: para comunicaciones **fiables en modo conectado**, de **dos vías** y con tamaño **variable** de los mensajes de datos. En dominios Internet subyace el protocolo **TCP**.

* Tipo **SOCK_RAW**: permite el acceso a protocolos de más bajo nivel como el **IP** (nivel de red)

* Tipo **SOCK_SEQPACKET**: tiene las características del SOCK_STREAM pero además el tamaño de los mensajes es fijo.

Dominio de un SOCKET

Indica el formato de las direcciones que podrán tomar los sockets y los protocolos que soportarán dichos sockets.

La estructura genérica es

```
struct sockaddr {
        u_short    sa_family;          /* familia */
        char       sa_data[14];        /* dirección */
    };
```

Pueden ser:

AF_UNIX

AF_INET

Dominio de un SOCKET

Dominio **AF_UNIX** (Address Family UNIX):

El cliente y el servidor deben estar en la misma máquina.

Debe incluirse el fichero cabecera **/usr/include/sys/un.h**.

La estructura de una dirección en este dominio es:

```
struct sockaddr_un {
   short       sun_family;          /* en este caso AF_UNIX */
   char        sun_data[108];       /* dirección */
   };
```

Dominio de un SOCKET

Dominio **AF_INET** (Address Family INET):

El cliente y el servidor pueden estar en cualquier máquina de la red Internet.

Deben incluirse los ficheros cabecera

/usr/include/netinet/in.h
/usr/include/arpa/inet.h
/usr/include/netdb.h.

La estructura de una dirección en este dominio es:

```
struct in__addr {
   u__long     s__addr;
   };

struct sockaddr__in {
   short       sin_family;              /* en este caso AF_INET */
   u__short    sin_port;                /* numero del puerto */
   struct      in__addr  sin__addr;     /* direcc Internet */
   char        sin_zero[8];             /* campo de 8 ceros */
   };
```

Filosofía Cliente-Servidor

El SERVIDOR

Vamos a explicar el proceso de comunicación servidor-cliente en **modo conectado**, modo utilizado por las aplicaciones estándar de Internet (telnet, ftp).

El servidor es el proceso que crea el socket no nombrado y acepta las conexiones a él.

El orden de las llamadas al sistema para la realización de esta función es:

1º) int **socket** (int *dominio*, int *tipo*, int *protocolo*)

2º) int **bind** (int *dfServer*, struct sockaddr* *direccServer*, int *longDirecc*)

3º) int **listen** (int *dfServer*, int *longCola*)

4º) int **accept** (int *dfServer*, struct sockaddr* *direccCliente*, int* *longDireccCli*)

Filosofía Cliente-Servidor

El SERVIDOR

1º) 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)

2º) int **bind** (int *dfServer*, struct sockaddr* *direccServer*, int *longDirecc*)

nombra un socket: asocia el socket no nombrado de descriptor *dfServer* con la dirección del socket almacenado en *direccServer*.
La dirección depende de si estamos en un dominio AF_UNIX o AF_INET.

3º) int **listen** (int *dfServer*, int *longCola*)

especifica el máximo número de peticiones de conexión pendientes.

4º) int **accept** (int *dfServer*, struct sockaddr* *direccCliente*, int* *longDireccCli*)

escucha al socket nombrado "servidor *dfServer"* y espera hasta que se reciba la petición de la conexión de un cliente. Al ocurrir esta incidencia, crea un socket sin nombre con las mismas características que el socket servidor original, lo conecta al socket cliente y devuelve un descriptor de fichero que puede ser utilizado para la comunicación con el cliente.

Filosofía Cliente-Servidor

El CLIENTE

Es el proceso encargado de crear un socket sin nombre y posteriormente enlazarlo con el socker servidor nombrado.

O sea, es el proceso que demanda una conexión al servidor.

La secuencia de llamadas al sistema es:

1º) int **socket** (int *dominio*, int *tipo*, int *protocolo*)

2º) int **connect** (int *dfCliente*, struct sockaddr* *direccServer*, int *longDirecc*)

Filosofía Cliente-Servidor

El CLIENTE

1º) 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)

2º) 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. El tipo de estructura dependerá del dominio en que nos encontremos.

SOCKETS

Comparación con PIPES

SOCKETS	PIPES
Referenciadop por descriptores	Referenciado por array de descriptores
Admite comunicación entre procesos de distintas máquinas	Solo admite comunicación entre procesos de una misma máquina
Comunicación bidireccional	Comunicación unidireccional
Filosofía cliente-servidor	Simple intercambio de información