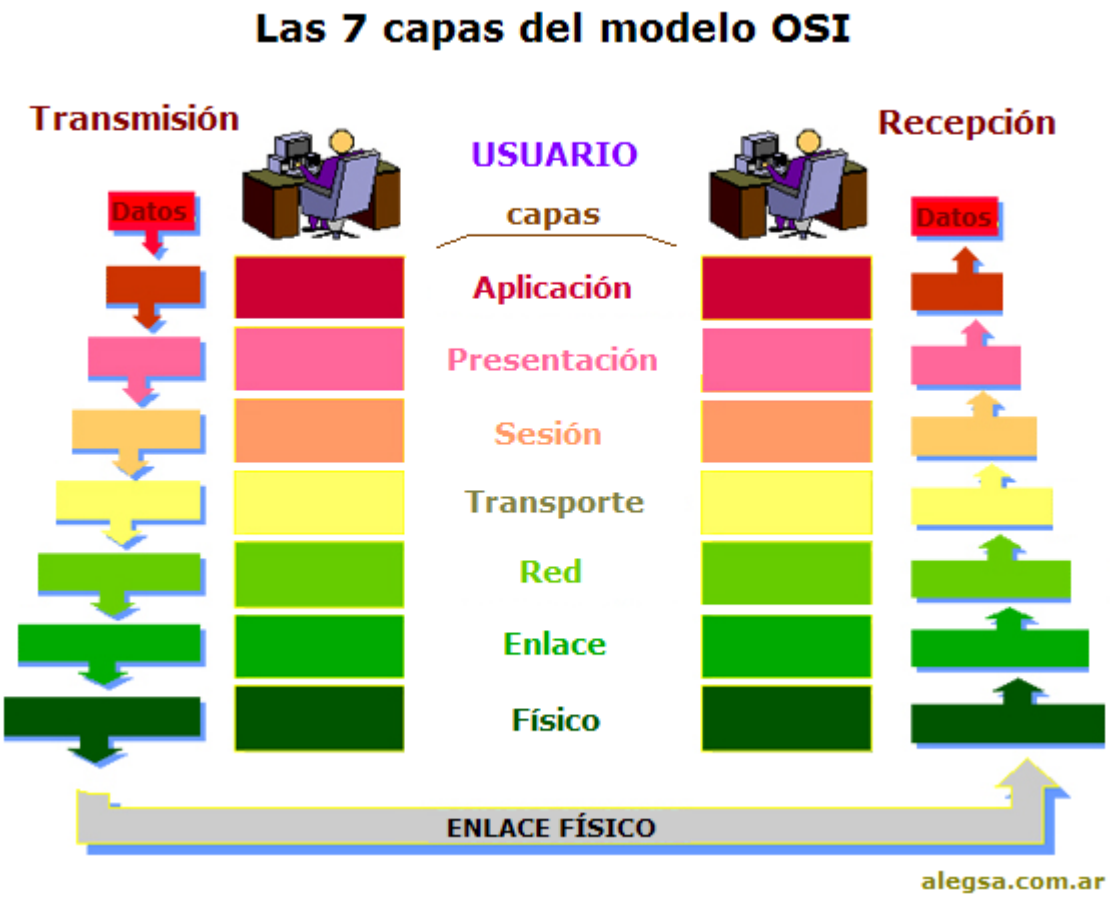
ARQUITECTURA CLIENTE SERVIDOR

Red de computadoras: Sistema o conjunto de computadoras (o nodos o hosts) interconectadas para compartir recursos e intercambiar información entre sí.

Topología de red: Es la forma en que está diseñada la red en el plano físico o lógico, puede ser de bus, estrella, anillo, árbol, malla, etc.

La ISO (International Standard Organization) define el modelo OSI (Open Systems Interconnection), una arquitectura de red basada en la descomposición de la red en 7 niveles de abstracción, cada uno de los cuales se apoya en los servicios que ofrece el nivel inmediato inferior y en el que cada nivel tiene una función bien definida.



El conjunto de reglas que rigen el diálogo entre un nivel de un sistema con su homólogo de otro sistema se conoce como **protocolo**.

En la capa 3 (Red) se solucionan los problemas de encaminamiento o enrutamiento de la información a través de las distintas subredes. Por lo tanto se necesita conocer la topología de la red y la interconexión de las redes.

En la capa 4 (Transporte) se establece la comunicación virtual entre aplicaciones siendo el modo conectado el servicio más importante. Mediante la multiplexación se podrá establecer varias conexiones a nivel de transporte mediante el uso de una sola conexión de red. Los protocolos TCP y UDP pertenecen a este nivel en la familia ARPA (INTERNET).

En las capas 5, 6 y 7 (Sesión-Presentación-Aplicación) se ofrecen servicios orientados al usuario y no sólo exclusivamente orientados a un nivel superior. El establecimiento de una sesión de comunicación, la gestión del diálogo y la sincronización son servicios que aportan estos niveles. También se ubican en este nivel las RPC (llamadas a procedimientos remotos), XDR y sistemas de ficheros tipo NFS.

Protocolo TCP/IP

El TCP es un protocolo orientado a conexión que transporta de forma segura grupos de octetos (segmentos) modo duplex (en los dos sentidos). Utiliza el mecanismo de puerto (al igual que el protocolo de transporte UDP, pero que actúa en modo datagrama no conectado). Este mecanismo se basa en la asignación para cada uno de los protocolos del nivel de transporte (TCP o UDP) de un conjunto de puertos de E/S identificados mediante un número entero. Así TCP y UDP multiplexarán las conexiones por medio de los números de los puertos. Existen una serie de puertos reservados a aplicaciones estándares Internet (ECHO - puerto 7, FTP - puerto 21, TELNET - puerto 23). El archivo /etc/services contiene la lista de los puertos estándar. En UNIX están reservados los números de puerto inferiores a 1024. El resto pueden ser utilizados, cualidad fundamental que aprovechan los programas definidos por el usuario para el establecimiento de comunicaciones entre hosts.

Con respecto a la arquitectura de Internet, la unión de las redes se basa en la existencia de gateways entre ellas. Pero lo más importante dentro de esta incompatibilidad de redes es otorgar a los hosts implicados una dirección lógica que se componga de:

- una dirección de red

- una dirección de la máquina dentro de la red

La dirección completa ocupa 32 bits, dándose en forma de 4 octetos:

n1.n2.n3.n4

Cada campo es un número decimal entre 0 y 255.

- La dirección nula se refiere a la red.

- Para direccionar un mensaje a todas las máquinas (broadcast), se usa una dirección dentro de la red en la cual todos sus bits son iguales a 1.

- El valor 127 en el primer campo se llama loopback y se refiere a una interfaz que permite al host enviarse paquetes a sí mismo.

Los *sockets* no son más que puntos o mecanismos de comunicación entre procesos que permiten que un proceso hable (envíe o reciba información) con otro proceso incluso estando estos procesos en distintas máquinas. Esta característica de interconectividad entre máquinas hace que el concepto de socket nos sirva de gran utilidad. Esta interfaz de comunicaciones es una de las distribuciones de Berkeley al sistema UNIX, implementándose las utilidades de interconectividad de este Sistema Operativo (*rlogin, telnet, ftp, ...*) usando sockets.

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) 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. Debido a esta característica, 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.

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**, el cual podrá "hablar" con el proceso servidor a través de la conexión con dicho **socket nombrado**.

El proceso crea un socket sin nombre cuyo valor de vuelta es un descriptor sobre el que se leerá o escribirá, permitiéndose 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. 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.

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. Para eliminar esta característica se cerrará el descriptor del socket servidor con nombre en cuanto realice una conexión con un proceso socket cliente.

-> 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.

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 como la que nos ocupa 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. Por debajo, 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.

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:

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

El cliente y el servidor deben estar en la misma máquina. Debe incluirse el archivo cabecera /usr/include/sys/un.h. La estructura de una dirección en este dominio es:

```
struct sockaddr__un {
    sa_family_t    sun__family;     /* en este caso AF_UNIX */
    char           sun__path[108];  /* ruta y nombre de archivo */
};
```

*** 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 */
    };
```