

## Servicios definidos en la capa de transporte.

Hay 2 tipos de servicio, orientado y no orientado a conexión.

El orientado a conexión consta de 3 partes:

- Establecimiento
- Transferencia de datos
- Liberación

En el servicio no orientado a la conexión se tratan los paquetes de manera individual.

## Pila de protocolos TCP/IP

Capa de red Física. especifica las características del hardware que se usará para la red.

Capa de vínculo de datos. identifica el tipo de protocolo de red del paquete, en este caso TCP/IP, proporciona también control de errores y estructuras.

Capa de Internet. acepta y transfiere paquetes para la red, incluye el protocolo IP, el protocolo de solución de direcciones (ARP) y el protocolo de mensajes de control de Internet (ICMP).

Capa de transporte. garantiza que los paquetes lleguen en secuencia y sin errores.

Capa de aplicación. define las aplicaciones de red y los servicios de Internet estándar que puede utilizar un usuario.

## Modelo Cliente-Servidor

el cliente hace una petición o solicita un servicio de red y el servidor responde a las peticiones de los clientes, proporcionando el servicio requerido

### Sockets TCP

Server

Socket()

bind()

listen()

accept()

\*call block

espera conexión

establece conexión

connect()

\*call block

recv()

\*call block

proceso petición

send()

respuesta

send()

recv()

\*call block

Client

Socket()

connect()

send()

recv()



**Socket()** crea un socket, sus parametros son:  
Dominio, tipo y protocolo.  
Devuelve un entero.

$\geq 0$  Si se crea correctamente  
 $< 0$  si se produce un error en la creacion

**bind()** asocia un socket a un puerto. Uso:

`int bind (int sockfd, struct sockaddr *addr, int addrlen);`

sockfd: Socket creado con anterioridad

addr: Puntero a la estructura sockaddr\_in

addrlen: Tamaño de la estructura apuntada por el puntero addr

devuelve 0 si funciona o  $< 0$  si hay un error

**listen()** Escucha de un puerto TCP, sus parametros:  
- descriptor del socket a poner a la escucha  
- Tamaño de cola de aceptacion de peticiones

Devuelve 0 si funciona o  $< 0$  si hay un error

**accept()** acepta las peticiones de conexion

Parametros:

- Socket asociado a la direccion y puerto puesto a la escucha

- estructura que contendra la direccion y puerto del cliente del que se acepta la conexion

- Puntero al tamaño de la estructura

Devuelve  $\geq 0$  Id del socket de donde se acepto la conexion  
Al aceptar la conexion se crea un nuevo socket que es el que atiende la conexion

• El socket original que escuchaba la direccion y puerto no se altera despues de alterar la conexion

$< 0$  error en la aceptacion

connect() solicita conexión, sus parámetros son:

- Descriptor del socket que se usa para la conexión
- Estructura con dirección y puerto al que desea conectarse
- Longitud de la estructura anterior

devuelve 0 si es correcto o  $\neq 0$  si hay un error

recv() recepción de datos en UDP, parámetros:

- Descriptor del socket
- buffer donde almacena datos a leer
- Tamaño máximo de los datos a leer
- Opciones de envío, en general 0 (ninguna opción)

devuelve  $< 0$  si hay error o  $\geq 0$  con el número de datos  
leídos o escritos

sendto() Envío de datos en UDP, parámetros:

- Descriptor del socket
- buffer con los datos a enviar
- Tamaño de los datos a enviar
- Opciones de envío, en general 0 (ninguna opción)
- puntero a la estructura de datos que contiene la dirección y puerto donde enviar los datos
- Longitud de la estructura anterior

devuelve  $< 0$  si hay error

$\geq 0$  si, número de bytes escritos.

Mora Guzman Jose Antonio

TAREA 3.