

SOCKETS

El kernel se divide en dos subsistemas que son el de ficheros y el de procesos(es una abstracción de la máquina intermediada por el kernel(es el dueño de todos los recursos de un ordenador))(con el mod usuario y portegio(estee))se encarga de aislar las zonas de memoria de los otros procesos, **trap**,

Modo de trabajo de unix con los ficheros. OPEN, read, write , close,

Sockets: vamos a crear una abstracción que representa el extremo de un cable, es decir, un punto final de comunicación, los sockets pertenecen a los procesos, hicieron un diseño independiente de:

- el protocolo de comunicación, se usa para la torre de protocolos tcp/ip y otros protocolos como una torre osi, el bluetooth,
- son independientes del lenguaje de programación, es decir, que se pueden usar con cualquier lenguaje, es decir, puedes tener un proceso en Python y otro en ruby, es decir, no necesitas un socket Python para solo los de python, esto se logra mediante llamadas al sistema,
- independientes del sistema operativo, a pesar de estar hecho en el kernel de unix, se puede usar en cualquier so

Un socket te ofrece una interfaz a los servicios en nivel de transporte, (host to host en tcp/ip), un nivel de transporte mueve bits entre procesos, significa que tienes una interfaz en la que puedes mandar bits a otro proceso, tenemos tres prácticas con niveles de transportes distintos,.

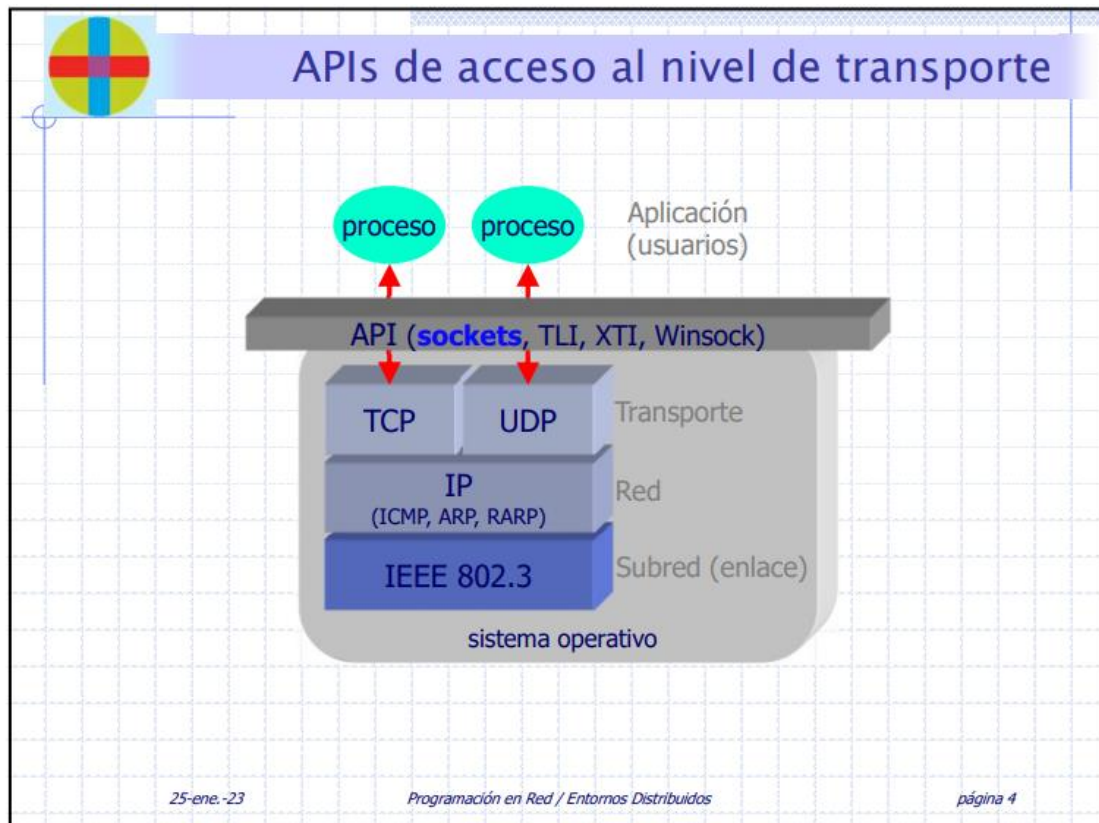
Los sockets usan direccionamiento **directo**, es decir direcciones en cada extremo de la comunicación, y **asimétrico**, es decir, tu sabes a quien le tienes que enviar el mensaje pero el destinatario no sabe quien se lo tiene que enviar,

- > Ejemplo: el socket TCP 172.25.4.4/1521 se refiere al puerto TCP 1521 en la dirección IP 172.25.4.4
- > La comunicación se produce siempre entre dos sockets.

La dirección incluye :

- » Un **socket** es un punto final de comunicación (socket → dirección IP + puerto + protocolo)

POSIX: interfaz portable(lo puedes utilizar en distintos sistemas unix) en el so unix,



Subred(enlace): ethernet(MAC, dentro de una LAN), host to host, punto a punto

Para poder colocar enlace deajo de ip: una interfaz que posee send packet y receive packet, un loopback(en cualquier maquina que tenga esta interfaz puedes probar la torre de protocolos),

Diferencia entre subred(homogeno) y red(heterogeneo)

Red: ip(encaminamiento y reenvio), le ofece a la capa de transporte llegar al destino mediante saltos, es decir, mueve bits entre maquinas(interfaces de red), maneja todo lo del nivel inferior,

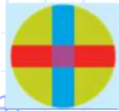
Transporte: udp(es un nivel de transporte(permite comunicación entre procesos), conmutacion de paquetes(lleva los bits dentro d ellos paquetes)), udp es un multiplexor, es decir, que los procesos son capaces de hablar, los puetos dsntingyena los varios procesos dentro de una maquina,

Tcp: protocolo conmutación de circuitos, es decir, establece un circuito virtual(un cable), tiene que resolver los porbemas de osi(son 3)

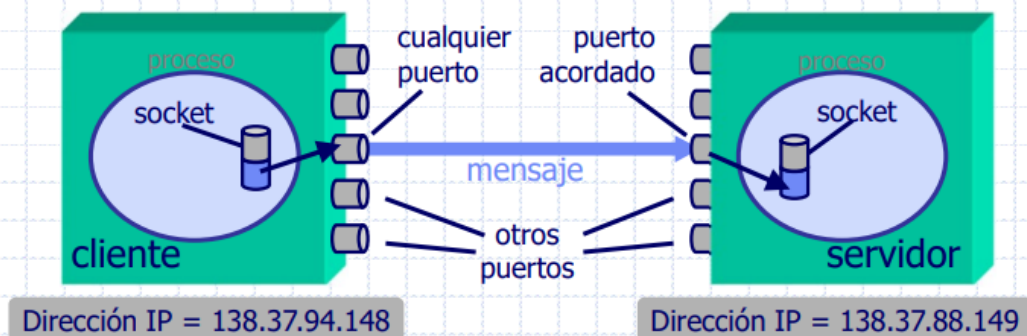
Cuando vas a trabajar tieens que elegir uno entre tcp y udp

Nivel 4(aplicacion): como pasa de un proceso al kenrel(con una llamada al sistema)

API de sockets(api de llamadas al sistema)



Sockets y puertos



» **socket** = conector

- ligado a una pareja (dirección IP+puerto) y a un protocolo (TCP, UDP, ...)
 - › El socket es un elemento del proceso
 - › El puerto es un elemento del sistema operativo
- Hay 2^{16} puertos posibles (algunos reservados)
- No se puede reabrir un puerto ya asignado a otro proceso

Quieres q un proceso hable con otro proceso de otra maquina, para ello utilizamos sockets, no puedes enganchar dos proceso al mismo puerto, solo uno por puerto, tcp deja el puerto reservdo drunte 90s cuando se ha muerto el porcsio asociado al puerto,



Modelo cliente-servidor con sockets

- » **Servidor:** proceso que se ejecuta en un nodo de la red y que proporciona un determinado recurso o servicio (está continuamente esperando peticiones)
 - Servidor **interactivo:**
 - › El mismo servidor recibe la petición y la atiende
 - › Si el servidor es lento en atender a los clientes, se pueden producir grandes esperas
 - Servidor **concurrente:**
 - › El servidor recibe las peticiones y genera por cada una un proceso que se encarga de atenderlas
 - › Sólo aplicable en sistemas multiproceso como UNIX
- » **Cliente:** proceso que se ejecuta en otro nodo (o el mismo) y realiza peticiones al servidor
 - Es el proceso que inicia y termina el diálogo

Los dos procesos que se comunican no son iguales, uno es cliente y otro es servidor, el cliente le pide cosas al servidor y el servidor lo responde, mientras espera, el proceso del servidor está durmiendo

El trabajo del servidor es dar servicio o escuchar peticiones,

Hay 2 tipos de servidores :

Intercativo vs concurrente: interactivo solo atiende a una petición, está pensado para peticiones de corta duración, concurrente, crea con un fork por cada petición que recibe,

Que hace un cliente y un servidor:



Implementación socket cliente-servidor

» Servidor:

1. Creación y enlace al puerto de servicio de un socket.
2. Desconexión del servidor de su terminal de lanzamiento o ejecución.
3. Bucle infinito en el cual el servidor:
 - a. Espera una petición.
 - b. La trata.
 - c. Calcula y formatea la respuesta.
 - d. Envía la respuesta.

» Cliente:

1. Creación del socket local.
2. Preparación de la dirección del servidor.
3. Envío del mensaje.
4. Espera del resultado.
5. Explotación del resultado.

Servidoro: renegar de su padre, x