Programación y Redes

Objetivos y mecanismos básicos

- Objetivos de una primitiva de comunicación
  - transferencia de datos
  - compartición de información
  - notificación de eventos
  - compartición y sincronización de recursos
  - control de procesos
- Primitivas de comunicaciones entre procesos
  - paso de mensajes
    - las primitivas del IPC más básicas
      - eso es lo que vamos a ver en la asignatura
  - llamadas a procedimientos remotos (RPC)
    - interacción entre procesos al nivel del lenguaje de programación
    - Comprobación de tipos
  - transacciones
    - soporte para operaciones (y su sincronización) entre objetos distribuidos
    - invocación de métodos remotos
    - ACID
      - Atomicity
        - se hace o no
      - Consistency
        - funciona o no
      - Isolation
        - separar
        - una transacción no afecta a otra
      - Durability
        - perdura en el tiempo
        - si se hace se guarda
    - BBDD
      - commit
      - rollback

Comunicación entre procesos (IPC)

- IPC = Inter-Process Communication
  - puede ser

        - local o remota
        - entre dos procesos cualquiera o hijos de un mismo padre
- Los procesos han de comunicarse y sincronizarse
    - la sincronización puede ser imprescindible
        - sincronizar es no hablar a la vez
            - tu hablas yo escucho
            - yo hablo tu escuchas

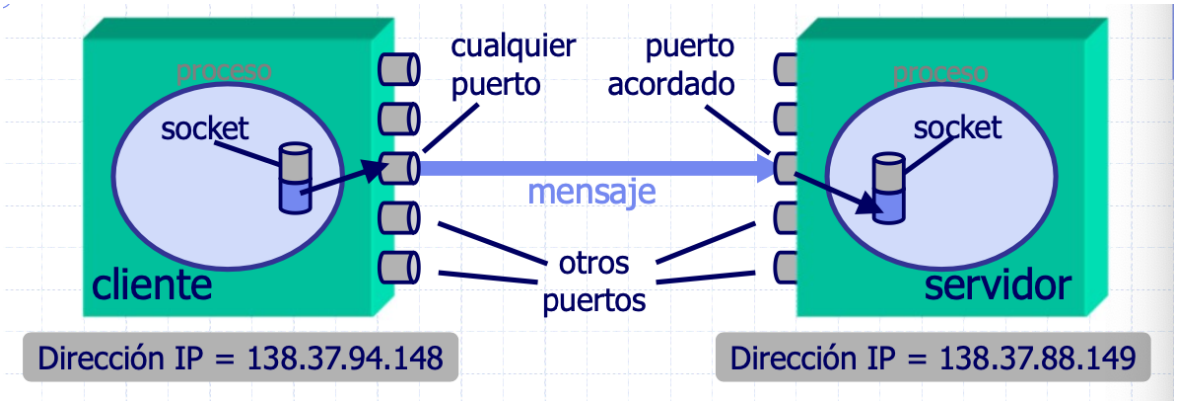
Comunicación entre procesos en Unix

- Ficheros
    - no sincroniza
- Señales (interrupciones software)
    - la información que se envía es sólo un número
        - no esta pensado para mandar información
- Tuberías
    - sin nombre: pipes
    - con nombre: FiFo
        - Implementan cola fifo
- Memoria compartida (system v / POSIX)
    - Espacio del sistema operativo
    - Requiere sincronización
        - Semáforos
            - system V / POSIX
        - Mutexes
            - variables de condición
    - Espacios compartido entre los procesos (threads)
- Paso de mensaje
    - Sockets BSD
        - funcionan en máquina remotas
    - Colas de mensajes System V / POSIX
        - colas de mensajes
        - semaforos
        - posix
        - Solo funcionan en misma máquina

Programa vs Proceso

- Proceso cuando lo ejecuto
- Programa el código escrito sin ejecutar

Sockets y puertos



- Dirección IP
  - interfaz de red

Hub

- Funciona al nivel más bajo
- Físico
- Nivel 1

Switch

- Nivel 2

Router

- Nivel 3

Nivel de red

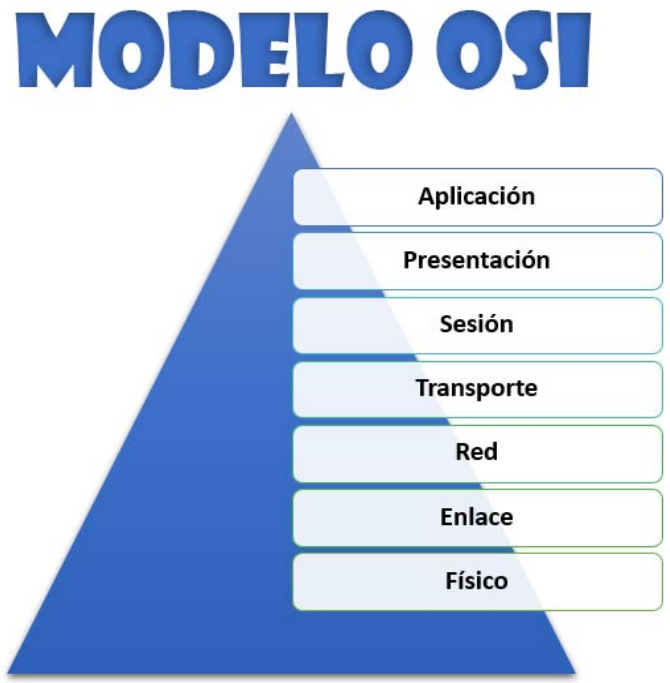
- Comunica interfaz con interfaz
- direcciones ip
  - nivel de red

Nivel de transporte

- Comunica proceso con proceso
- necesita mas que ip
- usa puertos

- Sockets = 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
    - El kernel unix tiene los niveles tcp las capas
        - kernel
            - darwin mac
            - mas usa bsd
            - bsd no es linux
            - linux esta basado en en svr pero tiene sockets
    - Socket es código
        - es virtual
        - usamos api
        - engancha un proceso con un puerto
        - socket es parre del proceso
        - socket esta ligado
            - a puerto
            - ip
            - protocolo
                - tcp o udp



Modelo OSI

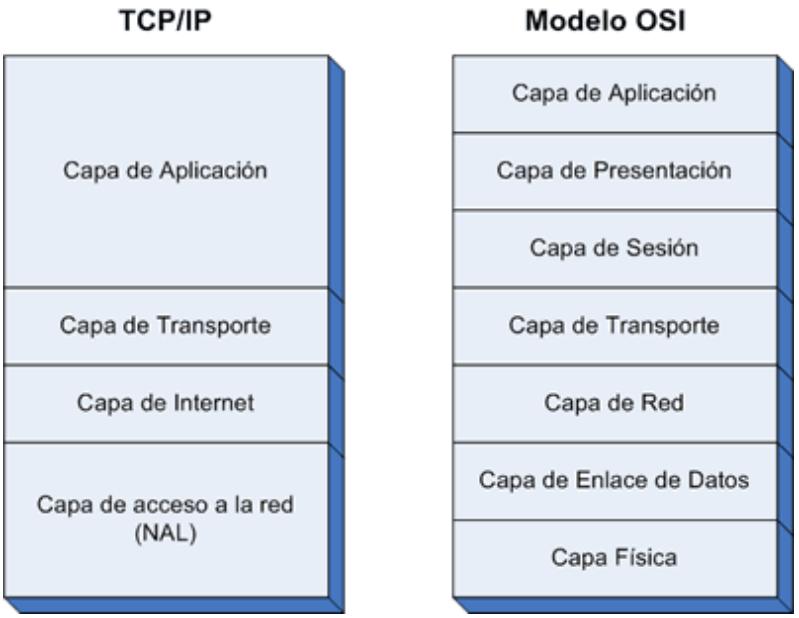
- Físico
    - cable
    - hablar entre dos personas

- Enlace
  - Transportar bits en el medio físico
  - enlace 1 y 1
- Red
  - Meter a más gente en la equación
  - múltiples enlaces de datos
  - mucha gente
  - identificar a la gente (ip)
  - store and forward
- Transporte
  - agentes finales
  - uno habla con uno dentro de una red de 80 personas
  - puertos
- Sesión
  - saber quien habla
  - yo hablo tu escuchas
  - tu hablas yo escucho
- Presentación
  - hablemos el mismo idioma
  - Mismo lenguaje
  - entendamos
- Aplicación
  - programa donde estamos nostros

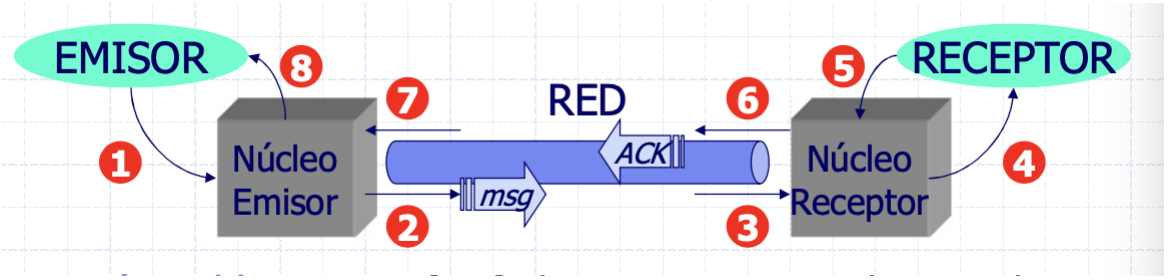
Red es commodity

- la usamos

Hacemos aplicaciones TCP/IP



Primitivas de comunicación



- Envío no bloqueante [1:8] El emisor continua al pasar el mensaje al núcleo
- Envío Bloqueante [1:2:7:8] El emisor espera a que el núcleo transmita por red el mensaje
- Envío bloqueante fiable [1:2:3:6:7:8] El emisor espera a que el núcleo receptor recoge el mensaje
- Envío bloqueante explícito [1:2:3:4:5:6:7:8] Idem al anterior, pero es la aplicación receptora la que confirma la recepción
- Petición-respuesta [1:2:3:4:<servicio>:5:6:7:8]: El emisor espera a que el receptor procese la operación para reanudar la ejecución
  - es la mejor hoy en día para aplicaciones de redes
- Sincrono
  - estar alineado
- asincrono
  - no esta alienado

JS asincrono

Direccionamiento

- Información válida para la identificación de elementos del sistema
  - Posibles receptores de un mensaje
- Mecanismos
  - Dirección dependiente de la ubicación (dirección física)
    - por ejemplo
      - dirección máquina + dirección puerto local
      - No proporciona transparencia
    - tu casa
    - la dirección cambia si te mueves
    - MAC
      - va asociada a tarjeta de red
  - Dirección independiente de la ubicación (dirección lógica)
    - facilita transparencia
    - la dirección no cambia si te mueves
    - Necesidad de proceso de localización
      - mediante broadcast
        - grito llega hasta donde llega la voz
        - no estan enrutados
      - Uso de un servidor de localización que mantiene relaciones entre direcciones lógicas y físicas
    - Uso de cache en clientes para evitar localización
  - ARP
    - Traducir de logica a física
- Cuando se enciende solo sabe la mac
  - grita y pregunta quien soy
  - el dhcp le da ip dns y el resto

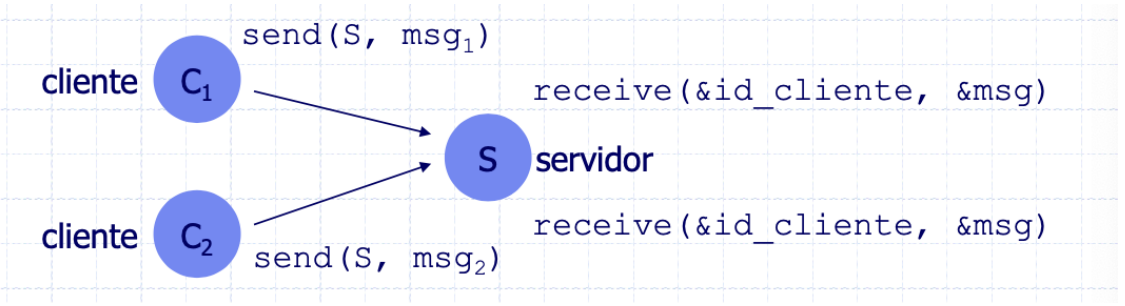
Desde que capa mandamos la ip es física o lógica

- No vemos las capas de abajo
- Depende de que capa
  - desde aplicación miramos para abajo tenemos transporte entonces para nosotros son físicas la ip + puerto
  - el localizados de físicas es el dns

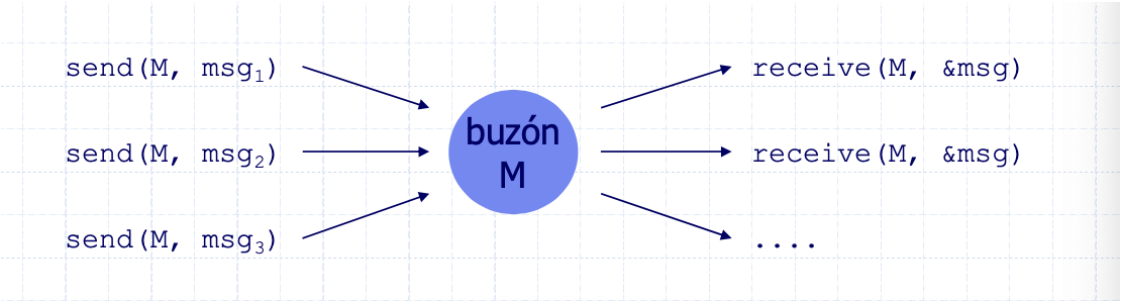
- lógico
  - direcciones de dominio
    - www.prueba.com

        - dns me dice donde esta

Comunicación directa

- En la comunicación directa los procesos deben nombrar explícitamente al otro
    - Direccionamiento simétrico
        - ```
          send(P, mensaje)
          receive(Q, mensaje)
          ```
    - Direccionamiento asimétrico
        - ```
          send(P, mensaje)
          receive(var, mensaje)
          ```
- 

Comunicación indirecta

- Consiste en tratar las rutas de comunicación como objetos de primera clase
- Ejemplo buzones (mailboxes)
    - ```
      send(M, mensaje)
      receive(M, mensaje)
      ```
- 

Mensajes de texto/binarios

- Estructura del mensaje
    - Cadenas de caracteres
      ```

    - Por ejemplo http
        - como se ve en el cuadro de arriba

```
"GET //www.ceu.es HTTP/1.1"
```

- Envío del mensaje
    - El emisor debe hacer un análisis de la cadena de caracteres transmitida

```
send("GET //www.ceu.es HTTP/1.1");
```

Mensajes Binarios

- Estructura del mensaje
    -
```
struct mensaje_st {
    unsigned int msg_tipo;
    unsigned int msg_seq_id;
    unsigned char msg_data[1024];
};
```
- Envío del mensaje
    -
```
struct mensaje_st confirm;
confirm.msg_tipo=MSG_ACK;
confirm.msg_seq_id=129;

send(confirm);
```

- Que textos son validos y cuales invalidos
- Mensajes de texto
    - Necesitamos definir una gramática
    - generamos un parser
    - ocupan más
    - proteger si el canal no es perfecto al contrario de binarios

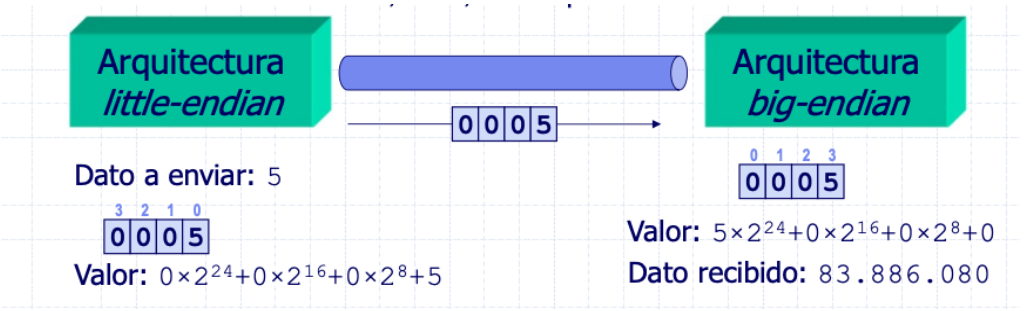
Formatos de representación

- Para la transmisión de formatos binarios tanto emisor y receptor deben coincidir en la interpretación de los bits transmitidos
- Problemática hay 3 cuestiones básicas que deben acordarse, y son
    - Tamaño
        - de los datos númericos
            - 16 vs 32 bits
    - Ordenación
        - de bytes (endianness)

        - little-endian vs big-endian
    - Formatos de texto
        - ASCII vs Unicode

Almacenamiento de datos multibyte
Endianness

- El término inglés endiannes (extremidad) desgina el formato en el que se alacenan los datos de más de un byte en un ordenador
- tipo de endianness
    - Little-endian: Intel
    - Big-endian: Motorola, PowerPC, SPAR,etc
    - Bi-endian: ARM, MIPS, DEC Alpha



Funcionan en big-endian

Capas de red

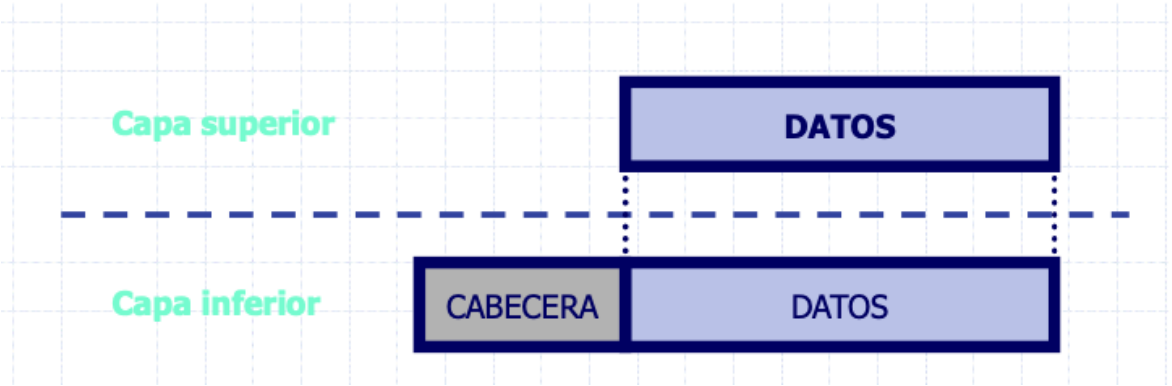
- Las aplicaciones se comunican entre ellas
    - llaman a las funciones de la capa de transporte
- La capa de transporte tiene que mover bits
    - llama a la capa de red
- La capa de red habla con el siguiente sistema
    - llama a la capa de subred
- La capa de subred organiza las tramas de datos para la transmisión
    - Usando los estándares físicos adecuados
    - Las tramas de subred van "saltando" desde el origen a su destino pasado por una secuencia de "routers"

Relaciones entre capas

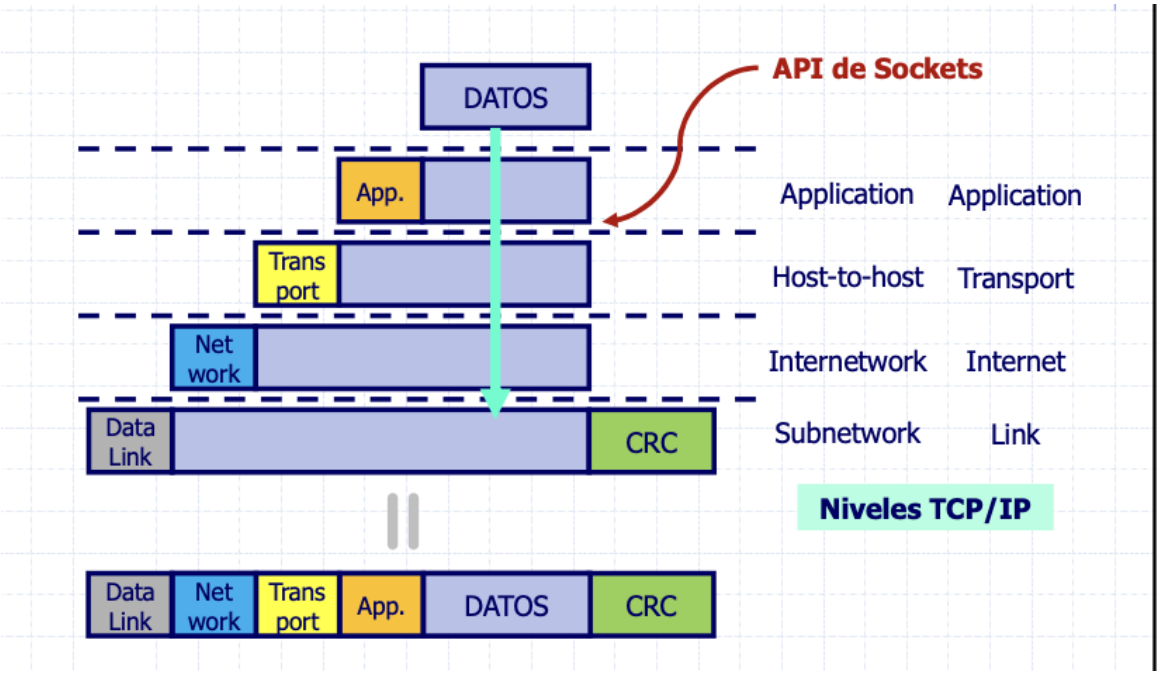
- Cada capa usa a la que tiene directamente debajo
    - la capa inferior añade cabeceras a los datos que recibe de la capa superior

- Parte de los datos de la capa superior puede ser cabeceras de capas aún más alta

En la cabecera de transporte va el puerto
TCP
IP



El modelo de capas de TCP/ IP



- Datagramas - transport
- Paquete - internet
- Tramas - link

Rayas son interfaces

- 1 - interfaz de usuario
- 2 - api de sockets

Distinguir procesos en una máquina el puerto

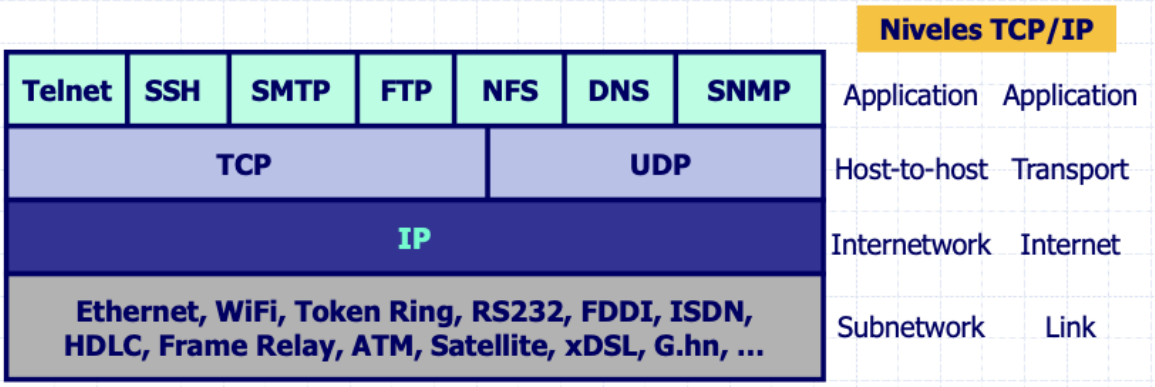
Componentes de TCP/IP
- estos son algunos de los protocolos disponibles en un entorno TCP TCO/IP

IP esta en el kernel
intefaz es el driver
ethernet es la tarjeta de red

Que se hace en ip
- Encaminamiento
- reenvío
- conmutación de paquetes
- garantía best effort
  - ninguna
- 4 problemas
  - perdidas
  - que se altere la información
  - que llegue desordenado
  - duplicación

UDP
- best effort
- pone y quita los puertos
- no orientado a conexión

TCP
- conmutación circuitos
- conexión virtual entre dos personas
- llegan de forma fiable
- orientado a conexión



Red o rango	Uso	
127.0.0.0	Reservado	fin clase A
128.0.0.0	Reservado	principio clase B
191.255.0.0	Reservado	fin clase B
192.0.0.0	Reservado	principio clase C
224.0.0.0	Reservado	principio clase D
240.0.0.0 – 255.255.255.254	Reservado	clase E
10.0.0.0	Privado	
172.16.0.0 – 172.31.0.0	Privado	
192.168.0.0 – 192.168.255.0	Privado	

- `ping`
  - Usa el protocolo ICMP para saber si hay conectividad IP con un remoto (envía `Echo Request`, espera `Echo Reply`)
- `traceroute / tracert`
  - Usa el campo TTL de los paquetes IP para trazar la ruta a un remoto
  - Puede no funcionar si los *routers* intermedios descartan paquetes
- `ifconfig / ipconfig`
  - Configura una interfaz de red (cableada / inalámbrica)
- `route`
  - Gestiona la tabla de encaminamiento de la máquina
- `arp`
  - Control del protocolo ARP (mapeo de direcciones físicas a IP)
- `telnet`
  - Conexión remota a un puerto TCP de una máquina