



Programación y Redes

Programación en Red / Entornos Distribuidos

Curso 2022/2023
Universidad San Pablo-CEU
Escuela Politécnica Superior
Campus de Montepríncipe

1



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 comunicación entre procesos:**
 - **paso de mensajes**
 - › las primitivas de IPC más básicas
 - **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

25-ene.-23 Programación en Red / Entornos Distribuidos página 2

2




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
- » En la siguiente diapositiva se muestran las diferentes posibilidades de comunicación entre procesos en el sistema operativo Unix.
 - Las opciones basadas en **paso de mensajes** están **marcadas en negrita y en azul**.

25-ene.-23 Programación en Red / Entornos Distribuidos página 3

3



Comunicación entre procesos en Unix

- » Ficheros
- » Señales (interrupciones software)
 - La información que se envía es sólo un número
- » Tuberías
 - Sin nombre: **pipes**
 - Con nombre: **FIFOs**
- » Memoria compartida (System V/POSIX)
 - Espacio del sistema operativo. Requiere sincronización:
 - › Semáforos System V/POSIX
 - › Mutexes, variables de condición
 - Espacio compartido entre los procesos (*threads*)
- » Paso de mensajes
 - **Sockets BSD**
 - **Colas de mensajes System V/POSIX**

25-ene.-23 Programación en Red / Entornos Distribuidos página 4

4




Cuadro de tipos de IPC en Unix								
Tipo de IPC	Posix .1	XPG. 3	V7	SVR2	SVR 3.2	SVR4	4.3 BSD	4.3+ BSD
Pipes (half duplex)	✓	✓	✓	✓	✓	✓	✓	✓
FIFOs (named pipes)	✓	✓		✓	✓	✓		✓
STREAMS pipes (full duplex)					✓	✓	✓	✓
Named STREAMS pipes					✓	✓	✓	✓
Message queues		✓		✓	✓	✓		
Semaphores		✓		✓	✓	✓		
Shared memory		✓		✓	✓	✓		
Sockets						✓	✓	✓
STREAMS					✓	✓		

25-ene.-23

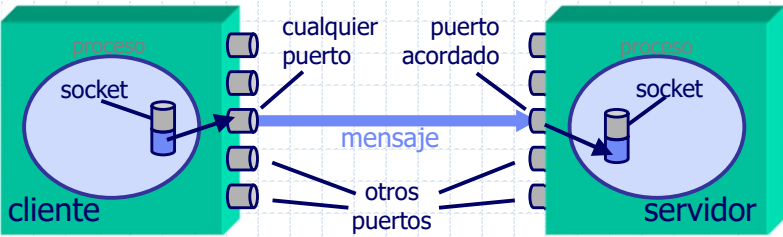
Programación en Red / Entornos Distribuidos

página 5

5



Sockets y puertos



Dirección IP = 138.37.94.148

Dirección IP = 138.37.88.149

» **socket** = conector

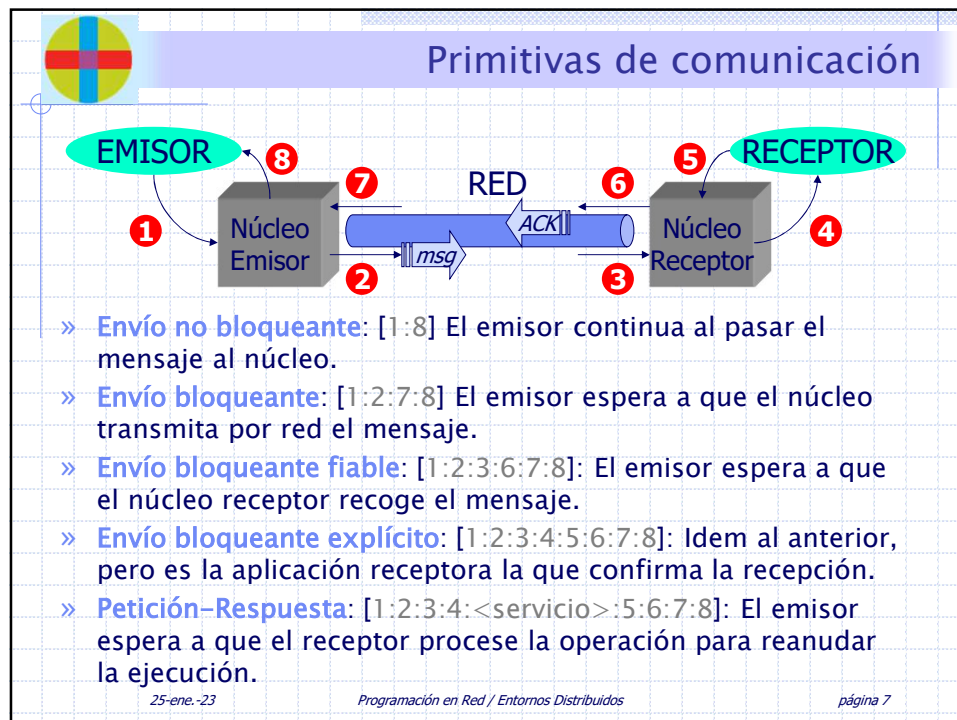
- ligado a una pareja (dirección IP+puerto) y a un protocolo (TCP, UDP, ...).
- > El socket es un elemento del proceso
- > El puerto es un elemento del sistema operativo
- Hay 2^{16} puertos posibles (algunos reservados)
- No se puede reabrir un puerto ya asignado a otro proceso

25-ene.-23

Programación en Red / Entornos Distribuidos

página 6

6



7



8

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)
```

```

graph LR
    C1((C1)) -- "send(S, msg1)" --> S((S))
    C2((C2)) -- "send(S, msg2)" --> S
    S -- "receive(&id_cliente, &msg)" --> R1[ ]
    S -- "receive(&id_cliente, &msg)" --> R2[ ]
  
```

25-ene.-23 Programación en Red / Entornos Distribuidos página 9

9

Comunicación indirecta

- » **Comunicación indirecta**:
 - Consiste en tratar las rutas de comunicación como objetos de primera clase.
- » Ejemplo: **buzones (mailboxes)**:



```
send(M, mensaje)
receive(M, mensaje)
```

```

graph LR
    C1[ ] -- "send(M, msg1)" --> M((buzón M))
    C2[ ] -- "send(M, msg2)" --> M
    C3[ ] -- "send(M, msg3)" --> M
    M -- "receive(M, &msg)" --> R1[ ]
    M -- "receive(M, &msg)" --> R2[ ]
    M -- "..." --> R3[ ]
  
```

25-ene.-23 Programación en Red / Entornos Distribuidos página 10

10



Mensajes de texto / binarios

» Mensajes de texto

- Estructura del mensaje... `"GET //www.ceu.es HTTP/1.1"`
 - › Cadenas de caracteres.
 - › Por ejemplo HTTP (como se ve en el cuadro de arriba).
- Envío del mensaje..... `send("GET //www.ceu.es HTTP/1.1");`
 - › El emisor debe hacer un análisis de la cadena de caracteres transmitida.

» 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);
```

25-ene.-23 Programación en Red / Entornos Distribuidos página 11

11



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:

1. **Tamaño** de los datos numéricos
 - › 16 bits vs 32 bits
2. **Ordenación** de bytes (*endianness*)
 - › *little-endian* vs *big-endian*
3. **Formatos** de texto
 - › ASCII vs Unicode

25-ene.-23 Programación en Red / Entornos Distribuidos página 12

12

Almacenamiento de datos multibyte

» **Endianness**

- El término inglés *endianness* ("extremidad") designa el formato en el que se almacenan los datos de más de un byte en un ordenador
- Tipos de *endianness*:
 - > *Little-endian*: Intel
 - > *Big-endian*: Motorola, PowerPC, SPARC, etc.
 - > *Bi-endian*: ARM, MIPS, DEC Alpha

Arquitectura
little-endian

Dato a enviar: 5

3	2	1	0
0	0	0	5

Valor: $0 \times 2^{24} + 0 \times 2^{16} + 0 \times 2^8 + 5$

Arquitectura
big-endian

0	1	2	3
0	0	0	5

Valor: $5 \times 2^{24} + 0 \times 2^{16} + 0 \times 2^8 + 0$

Dato recibido: 83.886.080

25-ene.-23
Introducción a la Ingeniería Informática
página 13

13

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 su transmisión

- Usando los estándares **físicos** adecuados
- Las tramas de subred van "saltando" desde el origen a su destino pasando por una secuencia de "routers"

25-ene.-23
Programación en Red / Entornos Distribuidos
página 14

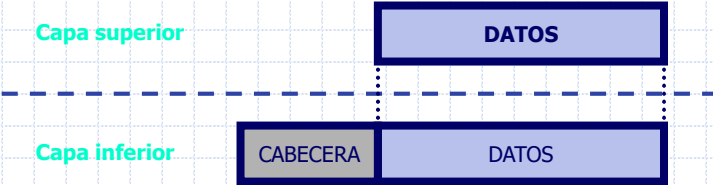
14



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 pueden ser **cabeceras** de capas aún más altas




25-ene.-23

Programación en Red / Entornos Distribuidos

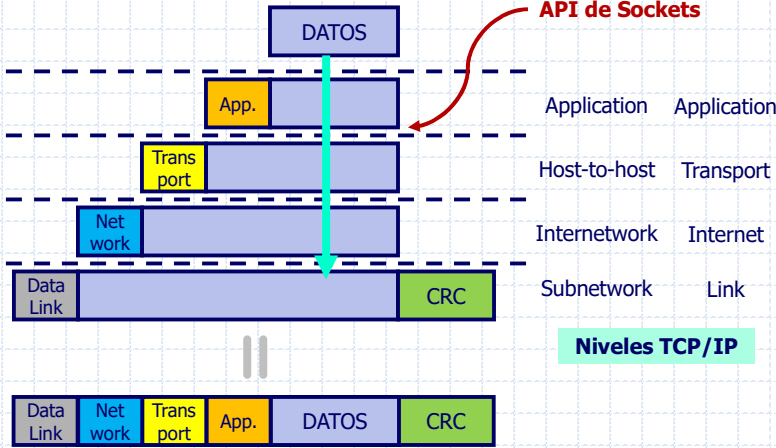
página 15

15



El modelo de capas de TCP/IP

» Más sencillo que el de OSI: sólo 4 capas




25-ene.-23

Programación en Red / Entornos Distribuidos

página 16

16



Componentes de TCP/IP

» Estos son algunos de los protocolos disponibles en un entorno TCP/IP

Niveles TCP/IP						
Telnet	SSH	SMTP	FTP	NFS	DNS	SNMP
TCP				UDP		
IP						
Ethernet, WiFi, Token Ring, RS232, FDDI, ISDN, HDLC, Frame Relay, ATM, Satellite, xDSL, G.hn, ...						

Application

Host-to-host

Internetwork

Subnetwork

Application

Transport

Internet


Link

25-ene.-23

Programación en Red / Entornos Distribuidos

página 17

17



Dir. IP reservadas y privadas (RFC 1918)


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	

25-ene.-23

Programación en Red / Entornos Distribuidos

página 18

18




Comandos de utilidad (UNIX / Windows)


- » 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

25-ene.-23 Programación en Red / Entornos Distribuidos página 19

19



¿Preguntas?



25-ene.-23 Programación en Red / Entornos Distribuidos página 20

20