



Programación y Redes

Programación en Red / Entornos Distribuidos

Curso 2024/2025
Universidad San Pablo-CEU
Escuela Politécnica Superior
Campus de Montepríncipe

1

En lo que se suele fijar la gente (como los cv)

va a decir algo si el otro ordenador se me te una hostia

Permite enviar los bits entre procesos no necesariamente locales

Los procesos tienen su propia memoria (no se puede meter en la memoria de otro proceso)



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** ACID
 - › soporte para operaciones (y su sincronización) entre objetos distribuidos
 - › invocación de métodos remotos

Como caracterizamos los servicios de transporte

mueve bits entre procesos

tenemos que estar aquí a la vez, que sea sincrónico. (asíncrono como correos, sincrónico como llamada)

Llamadas a procedimientos/funciones de manera remota.

Atomicity: divisible, o se hace o no se hace
Consistency: al ejecutar se haga o no, deja el sistema en estado correcto
Isolation: mis transacciones no afectan a otras
Durability: el efecto de la transacción no se altera por apagados

18-ene.-25

Programación en Red / Entornos Distribuidos

página 2

2

Tendrás que tener un sistema de primitiva que no dependa de cosas locales

Si no hay sistema de sincronizacion la comunicacion es un caos. No espera respuesta y sigue hablando.

Hace falta ambas comunicacion y sincronizacion

Inter Process Communication (and Synchronization)



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**.

18-ene.-25

Programación en Red / Entornos Distribuidos

página 3


3

Escribiendo en un fichero, despues el otro lo lee. Pero FALTA DE SINCRONIZACION si hay dos activos Pero mucho ancho de banda (el disco) aunque local.

Tiene sincronizacion, lo hace el kernel de unix, pero ancho de banda de un bit, para mandar mucha info es basicamente imposible.

Solo local

Hacer que un cacho de memoria que comparten ambos procesos, pero para ello tiene que tener sync (con sem/mutex), pero no funciona en remoto (mem local)



Comunicación entre procesos en Unix

» Ficheros no hechos para comunicar entre dos procesos

» Señales (interrupciones software)

- La información que se envía es sólo un número

» Tuberías

- Sin nombre: **pipes** primeras que tuvieron comm y sync a la vez
- 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 Berkley integraron TCP/IP en Unix

- **Sockets BSD** comm se metio en ficheros
- **Colas de mensajes System V/POSIX** Unix tiene subsistema: procesos ficheros

18-ene.-25

Programación en Red / Entornos Distribuidos

página 4

4

mega pipe (una direccion)

van juntos siempre

Cuadro de tipos de IPC en Unix								
Tipo de IPC	Estandares				(no va a mas reciente)			
	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					✓	✓		

18-ene.-25

Programación en Red / Entornos Distribuidos

página 5

Cuando IBM saca el SystemV Release 4 (SVR4) e integran los sockets, ganan la 'guerra'

5

Los portatiles tienen dos interfaces de red, WIFI y eth. Pero son dos vias terminales, no coge los paquetes de una interfaz y los pasa por la otra.

Tienen una interfaz de red para esta asignatura

La interfaz de red la tendran que compartir los procesos.

Identifico diferentes procesos por su puerto

Lo que garantiza IP, es transmitir los bits en paquetes (lo trocea) pero no lo garantiza/best effort

TCP te da 'un cable' virtual, pero colo lo hace por encima de IP que no da garantias, pues difícil

Lo que hace un router es mover paquetes entre sus interfaces

En verdad pasa por toda la pila de protocolos, pero eso no es relevante para esta asignatura

Sistemas finales

Sistema finales

cliente

servidor

socket

socket

cualquier puerto

puerto acordado

mensaje

otros puertos

Dirección IP = 138.37.94.148

interfaz de red

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 2¹⁶ puertos posibles (algunos reservados)

• No se puede reabrir un puerto ya asignado a otro proceso

18-ene.-25

Programación en Red / Entornos Distribuidos

página 6

El socket forma parte del proceso en si. Pertenece al proceso. Pero el puerto pertenece al OS, habra que pedirle al OS que nos deje un puerto.

Direccion del remoto: IP + puerto

No han aumentado los puertos, porque sobran para el numero de procesos.

6


UDP basicamente solo annade los puertos al paquete/multiplexacion (IP no se encarga porque solo es a nivel de red)

Transporte: IP+Puerto



Pero todo depende de la perspectiva, tambien se puede considerar logica la IP siendo su fisica la MAC (con ARP)

Un servidor no hace nada si no se comunica con el



Comunicación directa

» En la **comunicación directa** los procesos deben nombrar explícitamente al otro.

» Direccionamiento **simétrico**:

```
send(P, mensaje)
receive(Q, mensaje)
```

» Direccionamiento **asimétrico**:

el que usa TCP/IP

```
send(P, mensaje)
receive(var, mensaje)
```



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


En el que nombramos explícitamente el origen y el destino.

Destino necesita saber antes la IP del origen

El destino no necesita saber la del primero, porque la sabras cuando recibas el primer mensaje

Un rol activo/pasivo

9



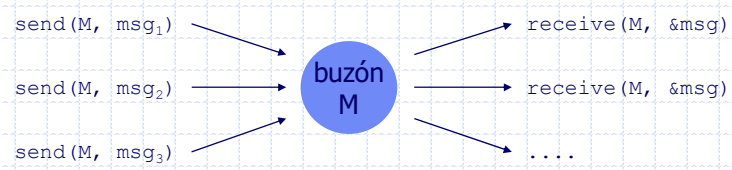
Comunicación indirecta

» **Comunicación indirecta**:

- Consiste en tratar las rutas de comunicación como objetos de primera clase.

» Ejemplo: **buzones (mailboxes)**:

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



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


En vez de identificar el origen y destino. La direccion la tiene un objeto intermediario

No hay manera de identificar los origenes y los destinos

Se usan muchas menos direcciones, lo que la gente tiene que saber es la direccion del buzón

10

Con los de texto, hay que hacer un parseo para dividir y entender los mensajes, con los binarios no. Pero es mas rapido y mas facil.



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];
};
registro (c)
- Envío del mensaje.....

struct mensaje_st confirm;
confirm.msg_tipo=MSG_ACK;
confirm.msg_seq_id=129;

send(confirm);


18-ene.-25

Programación en Red / Entornos Distribuidos

página 11

Pero como depende de las maquinas varia el endianness puede dar problemas/lios. Ya teniendo parseador, mejor usar mensajes de texto generalmente.

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

Para mensajes binarios

Para mensajes de texto


18-ene.-25

Programación en Red / Entornos Distribuidos

página 12

Usar utf-8 para codificar los textos, al menos de momento

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*

Arquitectura *big-endian*

TCP/IP usa big-endian

Dato a enviar: 5

0005

0005

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

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

Dato recibido: 83.886.080


18-ene.-25

Introducción a la Ingeniería Informática

página 13

Cuando intercambiamos numeros tenemos que tener en cuenta la endianness

13



Las reales 4 capas de TCP/IP

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

host to host

procesos (solo bits)

- Llama a la capa de red

» La **capa de red** habla con el siguiente sistema

internetwork

sistemas

- 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"

18-ene.-25

Programación en Red / Entornos Distribuidos

página 14

lo que vaya por encima lo resuelve el, como el servicio web HTTP

En TCP/IP lo hace el de presentacion


falta sincronizacion (sesion en OSI)

falta de que se habla (presentacion en OSI)

Lo que se habla en web es HTML

14

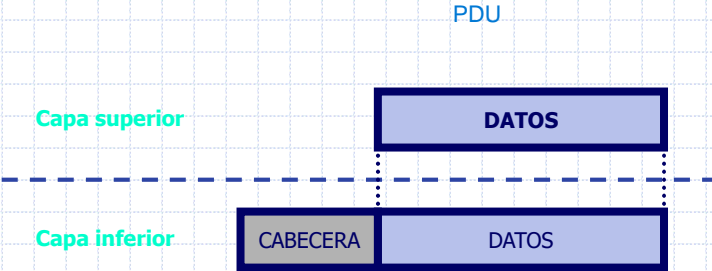
Se divide en capas para dividir un problema en diferentes partes, haciendolo mas mas facil de resolver



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



18-ene.-25 Programación en Red / Entornos Distribuidos página 15

Cada vez que se pasan a la siguiente capa, la anterior le mete una cabecera

15


Al final lo que va por el cable es mas grande que lo que se enviaba originalmente

Membranas que separan las capas, se cruzan con una API
organizada por la red

Ethernet, transmite tramas o frames
1516 bits

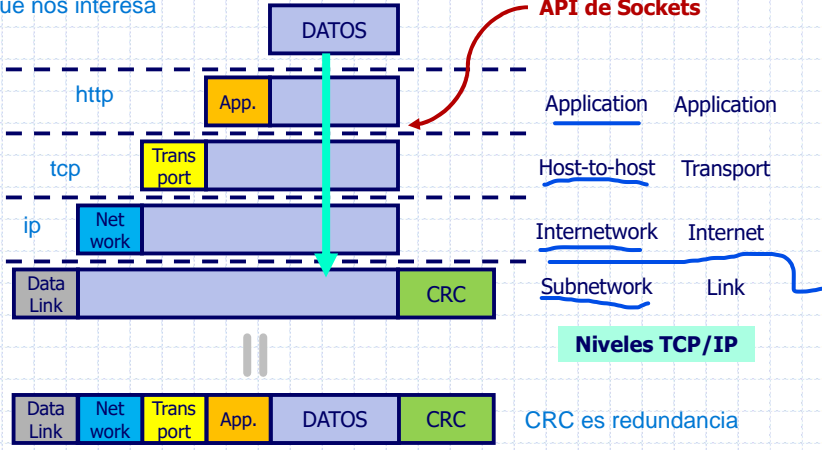
Sistema de red de area local, enchufado en un cable, modo bus.
CSMA-CD para detectar colisiones

Aplicacion: resuelve sesion, presentacion y aplicacion de OSI.



El modelo de capas de TCP/IP

» Más sencillo que el de OSI: sólo 4 capas



18-ene.-25 Programación en Red / Entornos Distribuidos página 16

En OSI habrian muchas mas cabeceras

Aunque se envíe por un gig, por la sobrecarga de protocolos, va a ir a menos de un giga

2 APIs, enviar y recibir paquete

Al final lo que acaba llendo por el cable son los datos mas muchas cabeceras

16 Transporte: transporta bits de proceso a proceso. El como depende del protocolo.

En el nivel de Subnetwork/enlace, solo se hablar con quien usa la misma tecnología que tu

Nivel de IP comunica entre redes homogeneas. Conecta las redes pequenas de Ethernet entre si, pero a nivel de maquina, y TCP a nivel de proceso

Telnet es basicamente ssh sin ser secure, solo sh

SMTP el correo

FTP para ficheros

NFS directorios

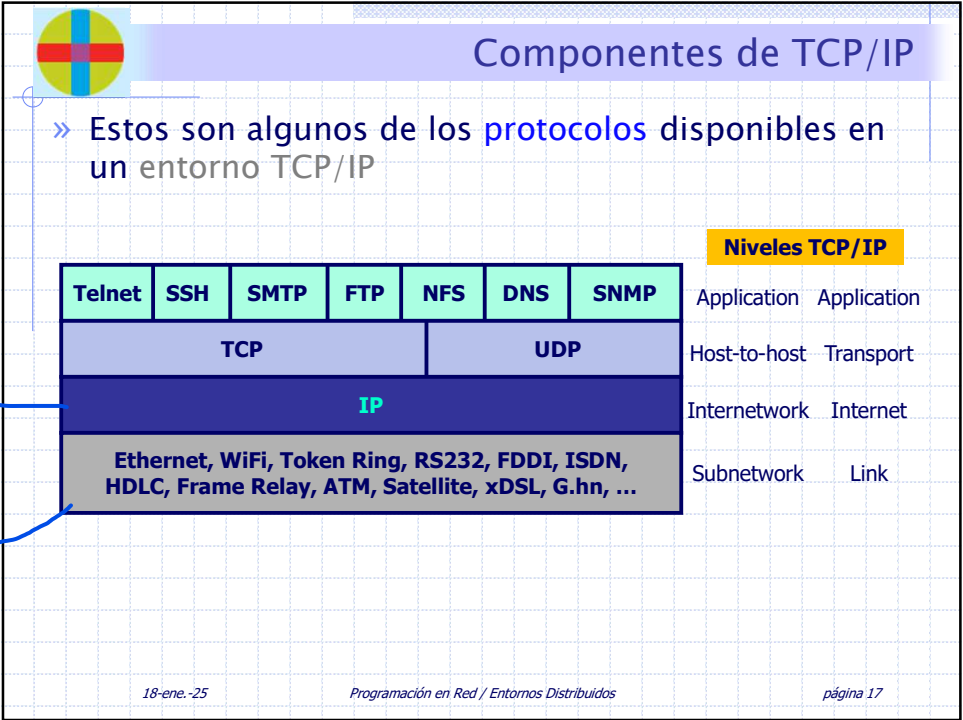
Un protocolo para unirlos a todos

Tecnologías de acceso, tienen que hablar los mismos

El arte de montar un wifi es bajar la potencia de transmision, de los puntos de acceso, funciona mejor, menos colisiones

Esta en orden de con que protocolos funcionan. SSH usa TCP, DNS usa UDP, ...

G.hn y TLC lo mandan por la red electrica



17


Realmente no hay diferencias entre IPs publicar y privadas, viene mas de un concenso de reserva de IPs, que se tiran el paquete si detecta que es de una publica.

Pero TCP/IP le da igual.

Por esto no se puede usar IPs privadas para salir de la red privada.

Cuando vieron que no daban las IPs publicas, usaron NAT para pasarlas de privada a publica. Se usa la dir publica del NAT basicamente

Si no se pone la mascara, TCP/IP usa valores por defecto. Las clases.



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	

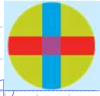
18-ene.-25

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

Usa el TTL para sacar las IP, con paquetes con TTL que van llenando a mas. Primero uno de TTL=1, despues 2,... y va mediendo la ruta, y sacando IPs

Las MAC porque es a nivel de ethernet. ARP es quien hace el mapeo.

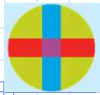
18-ene.-25

Programación en Red / Entornos Distribuidos

página 19

19

¿Preguntas?



18-ene.-25

Programación en Red / Entornos Distribuidos

página 20

20