

# Tema 1: Introducción a los Sistemas Operativos Distribuidos

## 1.1. Objetivos

### 1.1.1. Ventajas de los Sistemas Distribuidos.

- ✍ SO Distribuidos versus Centralizados.
- ✍ SO Distribuidos versus PC.

### 1.1.2. Desventajas de los Sistemas Distribuidos.

## 1.2. Conceptos de Hardware.

### 1.2.1. Multiprocesadores con base en buses.

### 1.2.2. Multiprocesadores con conmutador.

### 1.2.3. Multicomputadoras con base en buses.

### 1.2.4. Multicomputadoras con conmutador.

## 1.3. Conceptos Software.

### 1.3.1. Sistemas Operativos de Redes y NFS.

### 1.3.2. Sistemas realmente Distribuidos.

### 1.3.3. Sistemas de multiprocesador con tiempo compartido.

## 1.4. Aspectos de Diseño.

### 1.4.1. Transparencia.

### 1.4.2. Flexibilidad.

### 1.4.3. Confiabilidad.

### 1.4.4. Desempeño.

### 1.4.5. Escalabilidad.

## Bibliografía:

- ✍ Andrew S. Tanenbaum: "Sistemas Operativos Distribuidos"; tema 1, Prentice-Hall, 1996.
- ✍ Coulouris, George: "Sistemas Distribuidos: conceptos y diseño"; tema 1, Addison-Wesley, 2001.


# Tema I.- Introducción a los Sistemas Operativos Distribuidos.

## mitad 80's:

- ✍ 1º) Desarrollo de los microprocesadores (16, 32, 64 bits) con una capacidad de cómputo equivalente a un mainframe.
- ✍ 2º) Redes de área local (LAN)
- ✍ ? Como resultado: Sistemas de cómputo compuestos por gran número de CPU, conectados mediante una red de alta velocidad: Sistemas Distribuidos.
- ✍ **Definición de Sistema Distribuido:** es aquel en el que los componentes hardware o software, localizados en computadores unidos mediante una red, comunican y coordinan sus acciones sólo mediante paso de mensajes.
- ✍ **Consecuencias:**
  - ✍ **Concurrencia:** ejecución de programas concurrentes que pueden compartir recursos.
  - ✍ **Inexistencia de un reloj global:** no hay una única noción global del tiempo correcto.
  - ✍ **Fallos independientes:** fallos en la red, parada de un computador, terminación inesperada de un programa. Cada componente puede fallar con independencia que los demás continúen su ejecución.


# Tema I.- Introducción a los Sistemas Operativos Distribuidos.

## Motivación para construir y utilizar sistemas distribuidos

 **Compartir recursos: componentes hardware (discos, impresoras), software (ficheros, bases de datos, objetos).**

### **Ejemplos:**

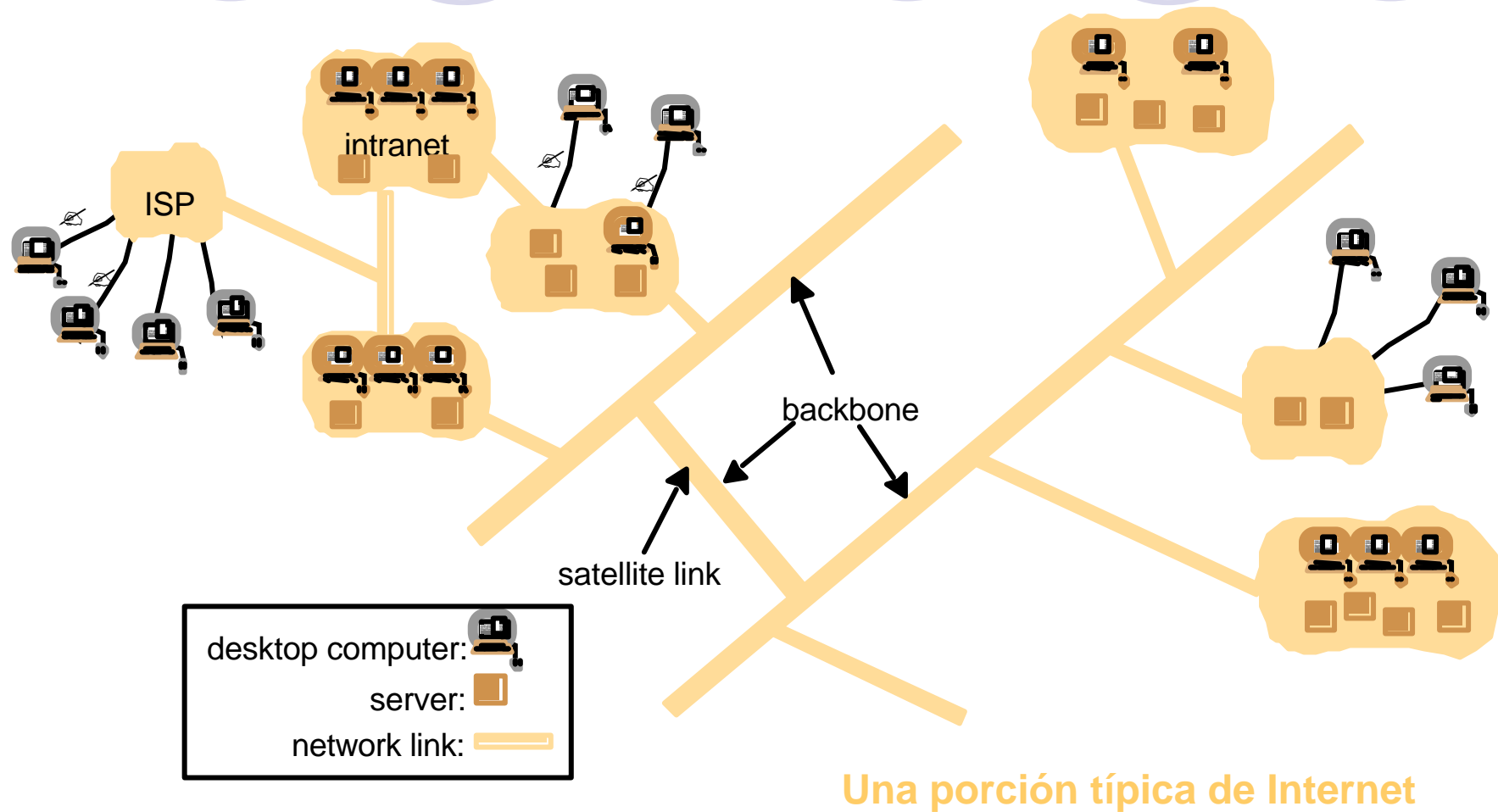
 **Internet. Ofrece servicios como WWW, correo electrónico y transferencia de ficheros. Se necesita un proveedor, una empresa que te conecte.**

 **Intranet. Una porción de Internet compuesta por varias LANs enlazadas por conexiones troncales. Se conecta a Internet por medio de un router. Se protegen con cortafuegos para que no entren o salgan mensajes no autorizados.**

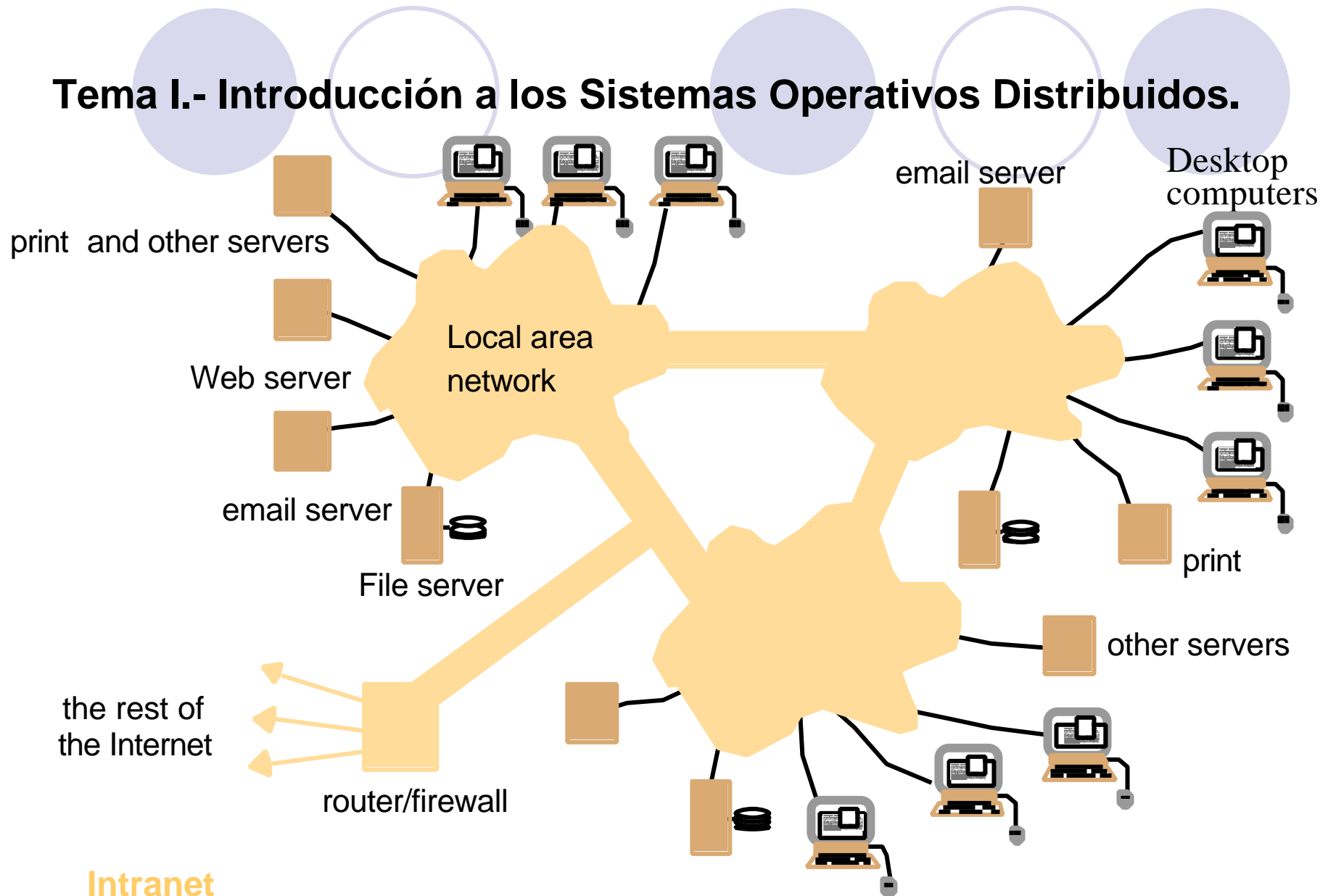
# Tema I.- Introducción a los Sistemas Operativos Distribuidos.

- ✍ **World Wide Web es un sistema abierto para publicar y acceder a recursos y servicios a través de Internet.**
- ✍ **Desarrollado en 1989 por CERN (Berners-Lee) para el intercambio de documentos entre físicos. La Web tiene una estructura hipermedia ideada por Bush en 1945.**
- ✍ **Se basa en tres componentes tecnológicos básicos:**
  - ✍ **HTML (Hypertext Markup Language)** un lenguaje de marcas para especificar el contenido y diseño de las páginas y establecer enlaces a otros documentos. El navegador formatea lo que recibe del servidor, interpreta HTML
  - ✍ **URL (Uniform Resource Locator)** un localizador de recursos en la Web.
    - ✍ *esquema:localización-específica-del-esquema*, esquema = tipo de URL (mailto, ftp, http, telnet ...)
    - ✍ Web es abierto. Se pueden inventar nuevos tipos de recursos con su propio direccionamiento.
    - ✍ *http://nombredelservidor [:puerto] [/nombredelcaminodelservidor] [?argumentos]*  
*http://www.google.com/search? q=kindberg*
  - ✍ **Arquitectura cliente-servidor, utiliza el protocolo HTTP (Hypertext Transfer Protocol)**
    - ✍ Interacciones petición-respuesta.
    - ✍ Tipo de contenido. Dependiendo será interpretado por el navegador o por una aplicación externa.
    - ✍ Un recurso por solicitud (si una página tiene 9 imágenes, 10 peticiones).
    - ✍ Control de acceso simple. Si queremos restringir, poner contraseña.
- ✍ **Problemas:** Mantenimiento y pérdida en el hiperespacio.

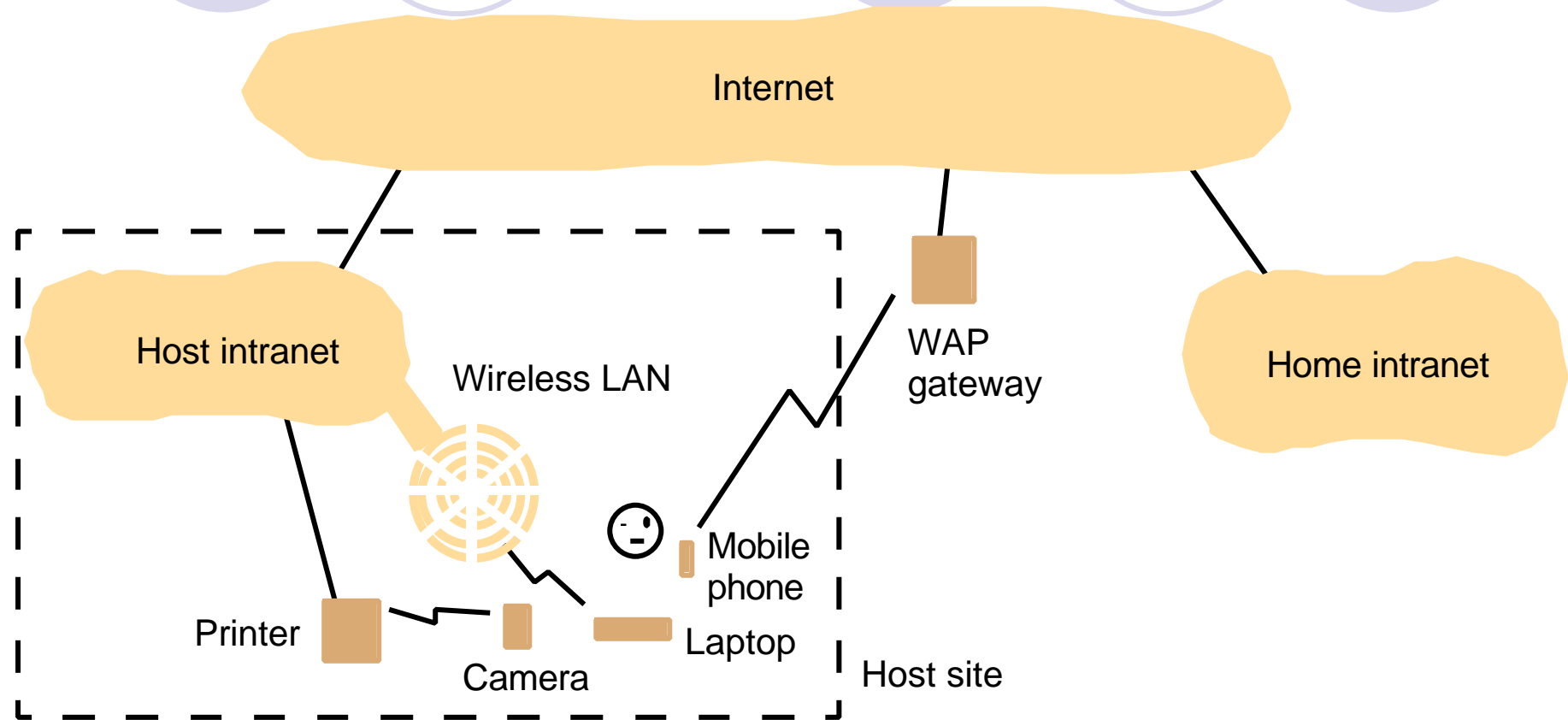
# Tema I.- Introducción a los Sistemas Operativos Distribuidos.



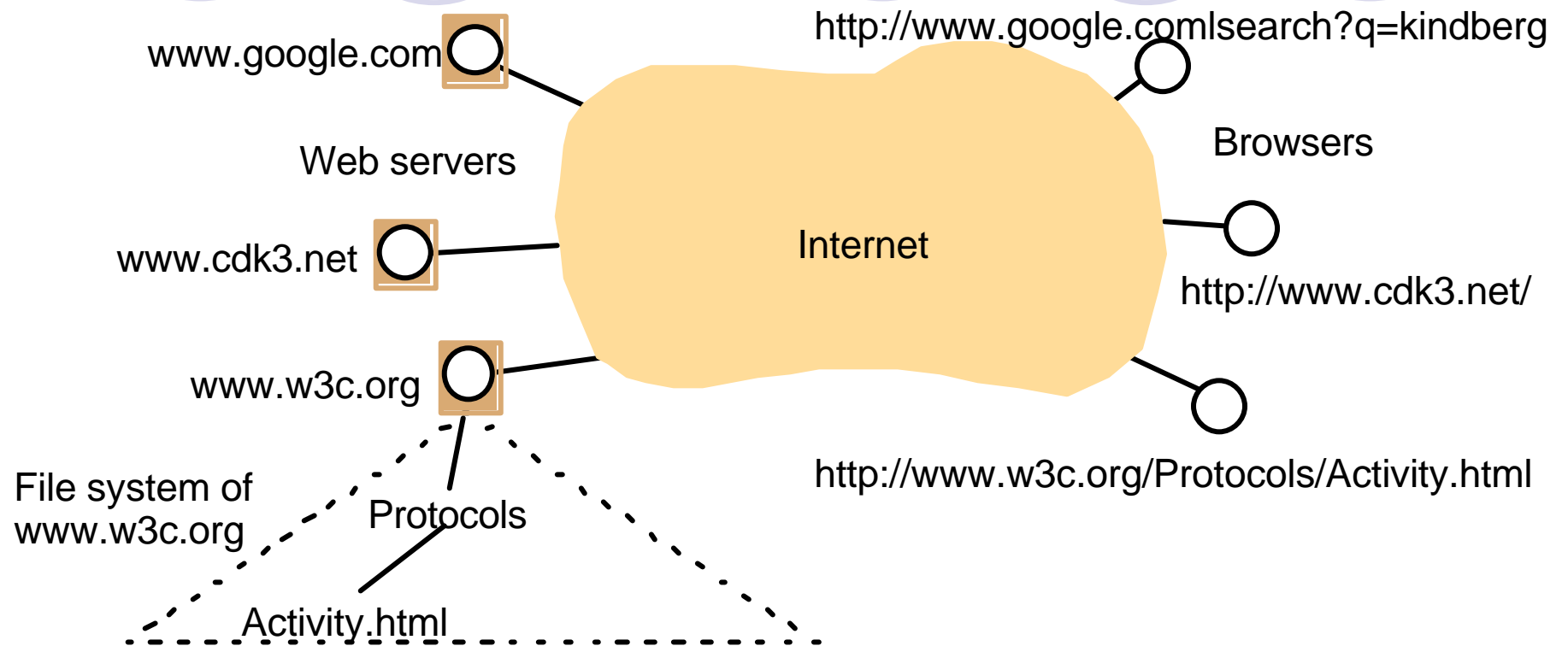
# Tema I.- Introducción a los Sistemas Operativos Distribuidos.



# Tema I.- Introducción a los Sistemas Operativos Distribuidos.



# Tema I.- Introducción a los Sistemas Operativos Distribuidos.



## Servidores y navegadores Web



## Tema I.- Introducción a los Sistemas Operativos Distribuidos.

<i>Date</i>	<i>Computers</i>	<i>Web servers</i>
1979, Dec.	188	0
1989, July	130,000	0
1999, July	56,218,000	5,560,866

### Computadores en la Web

<i>Date</i>	<i>Computers</i>	<i>Web servers</i>	<i>Percentage</i>
1993, July	1,776,000	130	0.008
1995, July	6,642,000	23,500	0.4
1997, July	19,540,000	1,203,096	6
1999, July	56,218,000	6,598,697	12

### Computadores vs. Servidores Web en Internet

# Tema I.- Introducción a los Sistemas Operativos Distribuidos.

## 1.1. Objetivos.

- ✍ ¿Tiene sentido disponer de sistemas operativos distribuidos??

### 1.1.1. Ventajas de los Sistemas Operativos Distribuidos.

#### SO Distribuidos versus Centralizados.

- ✍ Economía: los microprocesadores ofrecen una mejor proporción precio/rendimiento que los mainframes
- ✍ Velocidad: Un sistema distribuido puede tener un mayor poder de cómputo que un mainframe
  - ej) 1000 chips cada uno con  $v = 20$  MIPS dan un rendimiento de 20000 MIPS.
  - una sola CPU necesita una ejecutar una instrucc. cada  $0.05 \text{ nseg} = 20.000 \text{ Mhz}$
  - Einstein: tope velc. de la luz ? en  $0.05 \text{ nseg}$  se cubren  $1.5 \text{ cm}$  ? una computadora debe estar contenida en un cubo de  $1.5 \text{ cm}$ . Esto es imposible, el calor fundiría el sistema
- ✍ Distribución inherente: sistema de automatización de una fábrica, un banco
- ✍ Confiabilidad: Distribuimos la carga entre muchas máquinas, el fallo de una de ellas no supone el fracaso total del sistema
- ✍ Crecimiento por incrementos: Desarrollo gradual conforme surjan las necesidades

#### SO Distribuidos versus PC.

- ✍ ¿ Por qué NO trabajar de forma independiente? ? NO compartir datos!!!
- ✍ Datos compartidos: Permiten a los usuarios acceder a una B.D. común
- ✍ Dispositivos compartidos: Permiten compartir periféricos caros, impresora laser, de color, equipos de fotocomposición, cd-rom
- ✍ Comunicación: correo electrónico, envío de ficheros
- ✍ Flexibilidad: Difunde la carga de trabajo entre las máquinas disponibles en la forma más eficaz en cuanto a los costos

# Tema I.- Introducción a los Sistemas Operativos Distribuidos.

## 1.1.2. Desventajas de los Sistemas Distribuidos.

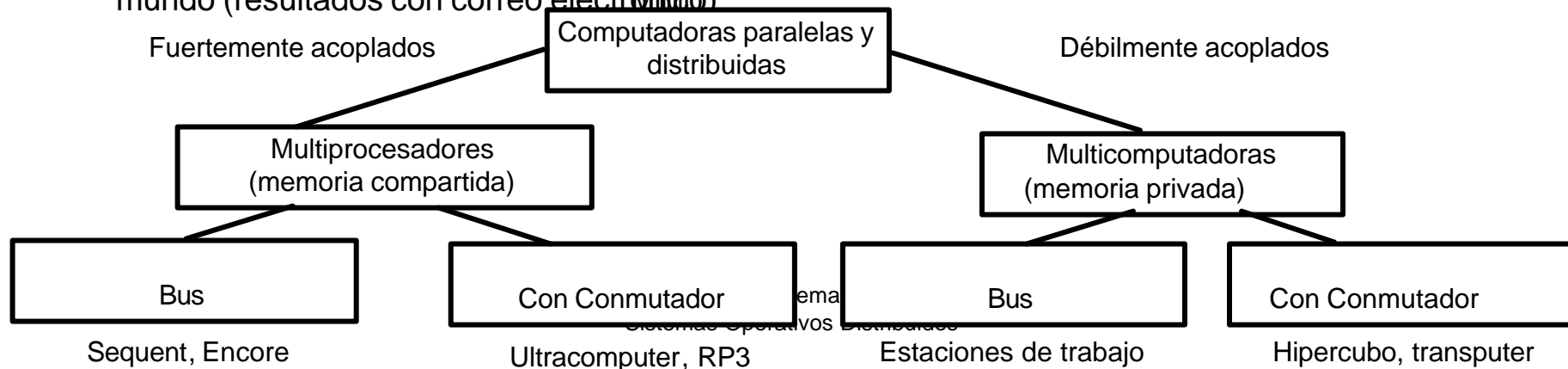
- ✍ Software!!!: No experiencia en diseño, implantación y uso de software distribuido.
  - ¿Qué tipo de sistemas operativos, lenguajes de programación y aplicaciones son adecuadas en sistemas distribuidos?
  - ¿Qué deben saber los usuarios de la distribución?
  - ¿Qué % debe hacer el sistema y qué % los usuarios?
- ✍ Redes: Necesitamos software especial de red que sea eficiente. El sistema llega a depender de la red.
- ✍ Seguridad: La información se hace más vulnerable en un sistema en red.

## 1.2. Conceptos de Hardware.

- ✍ Sistemas Distribuidos constan de varias CPU's. ? Organizaciones ? según la forma de interconexión y comunicación
- ✍ Flynn (1972): clasificación de las computadoras según el nº de flujos de instrucciones y el nº de flujos de datos
  - ✍ SISD (Single Instruction Single Data): Computadoras de un sólo CPU.
  - ✍ SIMD (Single Instruction Multiple Data): Una unidad de instrucción y que instruye a varias unidades de datos que trabajan en paralelo.
  - ✍ MISD (Multiple Instruction Single Data): Ninguna computadora se ajusta a este modelo.
  - ✍ MIMD (Multiple Instruction Multiple Data): **Un grupo de computadores independientes, cada una con su propio CP, programa y datos. Todos los sistemas Distribuidos.**
    - **multiprocesadores**: memoria compartida
    - **multicomputadoras**: cada máquina su propia memoria.

# Tema I.- Introducción a los Sistemas Operativos Distribuidos.

- ✍ Arquitectura de la red de conexión: bus y con conmutador
  - ✍ bus: un solo medio (cable) conecta todas las máquinas
  - ✍ conmutador: cables individuales de una máquina a otra, utilizan patrones diferentes de cableado. Los mensajes viajan por los cables y se toma en cada nodo una decisión explícita de conmutación
- ✍ Fuertemente/Débilmente acoplados
  - ✍ fuertemente acoplados: el retraso que experimenta el envío de un mensaje de una computadora a otra es corto y tasa de transmisión ( $n^{\circ}$  de bits por seg transferidos) es alta. Dos chips de CPU en la misma tarjeta de circuito impreso.
  - ✍ débilmente acoplados: dos computadoras conectadas por un modem.
- ✍ Los sistemas fuertemente acoplados = sistemas paralelos (un solo problema)
- ✍ Los sistemas débilmente acoplados = sistemas distribuidos (varios problemas no relacionados entre sí).
- ✍ !!! contraejemplo) factorizar un  $n^{\circ}$  de 100dígitos con varios cientos de computadoras en todo el mundo (resultados con correo electrónico)



# Tema I.- Introducción a los Sistemas Operativos Distribuidos.

## 1.2.1. Multiprocesadores con base en buses.

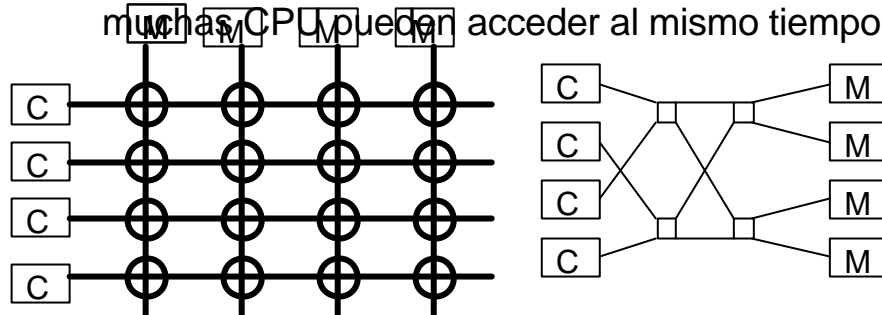
- ✍ Constan: varias CPU's conectadas a un bus común, junto con un módulo de memoria.
- ✍ Todas las CPU's operan sobre el mismo módulo de memoria ? memoria es **coherente**
- ✍ Sobrecarga del bus con muchas CPU's!!!

✍ Solución: Memoria caché entre CPU y el bus ✍ Problema: Memoria incoherente ✍ Solución: Obligar escritura en M.P. cuando se escriba en caché.

Es posible colocar de 32 a 64 CPUs en el mismo bus.

## 1.2.2. Multiprocesadores con conmutador.

- ✍ Multiprocesadores con más de 64 procesadores necesitan otro método de conexión a memoria.
- ✍ Dividir la memoria en módulos y conectarlos a las CPUs con un conmutador de cruceta. Ahora muchas CPU pueden acceder al mismo tiempo a la memoria (módulos distintos)



✍ Con  $n$  CPUs y  $n$  memorias, necesitamos  $n^2$  conmutadores. Prohibitivo!!!

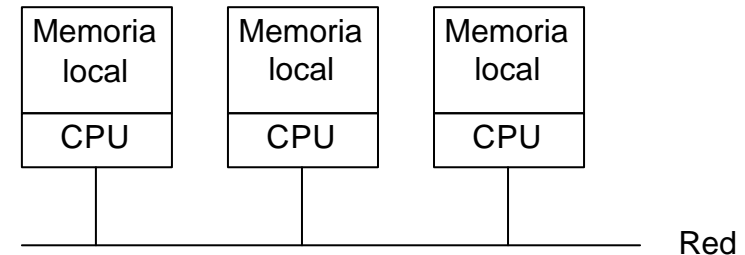
✍ Solución: La red omega. Con  $n$  CPUs y  $n$  memorias, la red omega necesita  $n$  etapas de conmutación, cada una con  $\log_2 n$  conmutadores, para un total de  $n \log_2 n$  conmutadores. Sigue siendo Prohibitivo!!!

Además problema: Retraso!!!

# Tema I.- Introducción a los Sistemas Operativos Distribuidos.

- ✍ Solución: Sistemas jerárquicos. Rápido acceso a su propia memoria y un acceso más lento a las demás. Mejor promedio de acceso pero....
- ✍ Problema: colocación de programas y datos para que los accesos sean locales. Necesita de complejos algoritmos.
- ✍ Conclusión: la construcción de un multiprocesador grande, fuertemente acoplado y con memoria compartida es difícil y cara.

Estaciones de trabajo

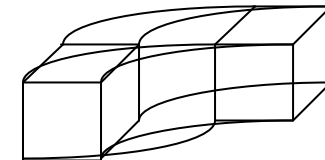
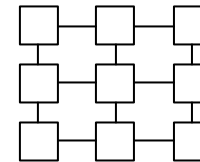


## 1.2.3. Multicomputadoras con base en buses.

- ✍ Fácil de construir.
- ✍ Único problema: comunicación de las distintas máquinas. Necesitan de algún esquema de interconexión pero volumen de tráfico entre máquinas es de un orden mucho menor que entre CPU-memoria.
- ✍ Desde el punto de vista topológico es igual al multiprocesador basado en bus. Pero el medio de conexión puede ser una LAN de mucha menor velocidad (10-100 Mb/seg LAN, en comparación 300Mb/seg de bus plano)

## 1.2.4. Multicomputadoras con conmutador.

- ✍ Dos topologías comunes de conexión: retícula y un hipercubo



# Tema I.- Introducción a los Sistemas Operativos Distribuidos.

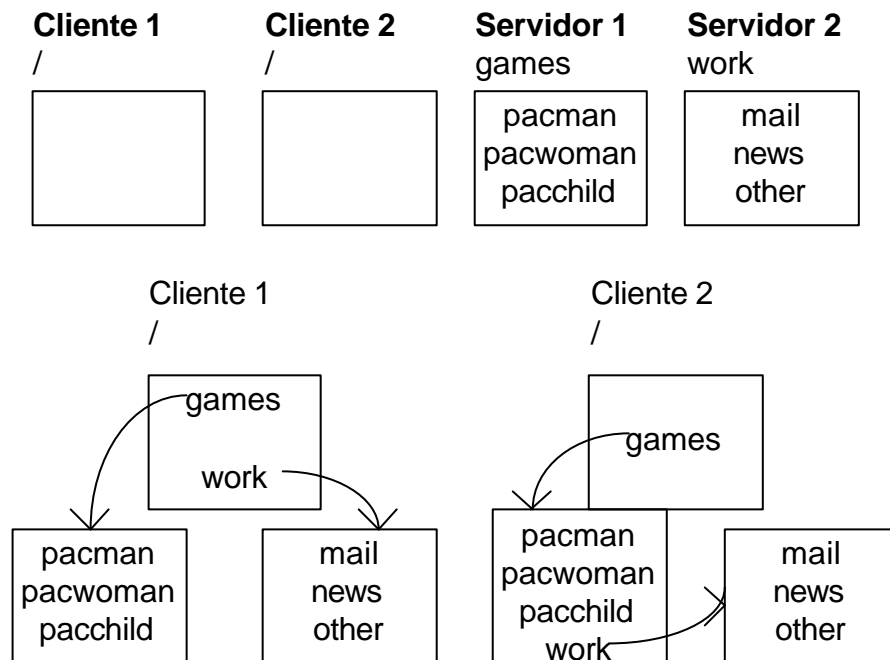
## 1.3. Conceptos Software.

- ✍ ¿Qué tipo de software (SO) se relaciona con cada tipo de hardware?
  - ✍ Distinguimos 2 tipos de SO: los débilmente acoplados y los fuertemente acoplados (relacionado con el software débil/fuertemente acoplado).
- ✍ Software débilmente acoplado: máquinas y usuarios de un sistema distribuido son independientes en lo fundamental, pero interactúan en cierto grado.
  - ✍ ejemplo) un grupo de PCs con CPU, memoria y disco y su propio SO, pero compartiendo ciertos recursos como impresora láser y BD en una LAN.
- ✍ !!! 4 tipos de hardware distribuido y 2 tipos de software: 8 combinaciones hardware/software????
- ✍ En realidad, tan sólo 4, para el usuario la tecnología de la interconexión no es visible. Un multiprocesador es un multiprocesador,

## 1.3.1. Sistemas Operativos de Redes y NFS.

- ✍ Software débilmente acoplado: red de estaciones trabajo conectadas con LAN
  - ✍ Todos tienen su SO, ejecución local de órdenes.
  - ✍ Eventualmente un usuario se puede conectar de manera remota con otra estación de trabajo con *rlogin machine*. Ahora funciona como una terminal remota. Las órdenes son enviadas a la máquina remota.
- ✍ Conexión primitiva!!. Mejor disponer de un sistema de archivos global compartido, accesible por todas las estaciones de trabajo.
  - ✍ Una o varias máquinas, **servidores de ficheros**, aceptan solicitudes de los programas de usuarios que se ejecutan en otras máquinas, **clientes**.
- ✍ Las estaciones de trabajo pueden

# Tema I.- Introducción a los Sistemas Operativos Distribuidos.



✍ SO debe controlar las estaciones de trabajo, a los servidores de archivo y debe encargarse de la comunicación entre ellos

✍ Si los clientes y los servidores ejecutan ? SO ? deben coincidir en el formato y significado de los mensajes que pueden intercambiar.

✍ A estos SO se les conoce como **Sistemas Operativos de red**

✍ Uno de los más conocidos es Network File System (NFS). Fue desarrollado para estaciones Unix. Ahora soporta otras plataformas. NFS soporta sistemas heterogéneos (Unix, DOS).

✍ ejemplo) clientes MS-DOS (Intel 386) y servidores UNIX (Motorola 68030 ó SPARC).

✍ Tres aspectos importantes de NFS: la arquitectura, el protocolo y la implantación.



# Tema I.- Introducción a los Sistemas Operativos Distribuidos.

## La arquitectura de NFS.

Objetivo: permitir que un grupo de clientes y servidores compartan un sistema de ficheros común.

- ✍ Una máquina puede ser a la vez cliente y servidor al mismo tiempo.
- ✍ Cada servidor NFS exporta uno o varios de sus directorios para el acceso por parte de clientes remotos (árboles completos de directorios). La lista de directorios exportados se conserva en */etc/exports* (son exportados al arrancar el servidor).
- ✍ Los clientes pueden tener acceso a los directorios exportados mediante el montaje. Cuando un cliente monta un directorio, se convierte en parte de la jerarquía.
  - ✍ Hay estaciones sin disco ? sist. archivos está soportado en su totalidad por el servidor remoto.
- ✍ No hay que hacer nada para compartir los archivos. Los programas en la máquina cliente se ejecutan de igual manera ya sea local o remoto.

## Protocolos de NFS.

Objetivo de NFS: soportar un sistema heterogéneo, en donde clientes y servidores pueden ejecutar ?  
SO en hardware diverso ? la interfaz entre clientes y servidores debe estar bien definida.

- ✍ NFS lo logra mediante dos protocolos cliente-servidor.

# Tema I.- Introducción a los Sistemas Operativos Distribuidos.

## Protocolo 1.- Maneja el montaje.

- ✍ Un cliente envía el nombre de una ruta de acceso a un servidor y solicita el permiso para montar ese directorio en alguna parte de su jerarquía de directorios.
- ✍ Si el nombre de ruta es válido y el directorio especificado ha sido exportado, el servidor devuelve un *file handle* al cliente.
  - ✍ *file handle* contiene campos que identifican el tipo de sistema de ficheros, el disco, el número de i-nodo del directorio y la inf. relativa de seguridad.
  - ✍ las peticiones de lectura/escritura posteriores para el directorio montado utilizan el *file handle*.
- ✍ Muchos clientes están configurados para que monten ciertos directorios remotos sin intervención manual (/etc/rc, guión shell con los comandos de montaje remoto, se ejecuta durante el arranque).
- ✍ Automontaje: un conj. de directorios remotos queda asociado con un directorio local pero no son montados durante el arranque del cliente sino cuando se intente abrir un fichero remoto.
- ✍ Ventajas sobre el montaje estático:
  - ✍ Si uno de los servidores NFS de /etc/rc no funciona es difícil recuperar al cliente.
  - ✍ Utiliza un conj. de servidores en

## Protocolo 2.- Acceso a directorios y ficheros.

- ✍ Los clientes pueden enviar mensajes a los servidores para el manejo de directorios y lectura/escritura ficheros. También tienen acceso a sus atributos (modo, tamaño, fecha).
- ✍ NFS soporta la mayoría de las llamadas al sistema UNIX salvo: OPEN y CLOSE.
- ✍ Un cliente envía al servidor un mensaje con el nombre del archivo y una solicitud para revisarlo y devolver un *file handle* (LOOKUP).
- ✍ LOOKUP no copia inf. en las tablas internas del sistema como OPEN ? el servidor no tiene que recordar las conexiones abiertas, si falla no se pierde inf. sobre los ficheros abiertos. A estos servidores se les llama **sin estado**.
- ✍ Una llamada READ contiene *file handle*, el offset y el número de bytes deseados
- ✍ RFS (Remote File System), necesita que los ficheros sean abiertos!!!
- ✍ NFS complica la semántica de un fichero UNIX. En un servidor sin estado, los bloqueos no pueden asociarse con los ficheros abiertos ? NFS necesita de un mecanismo adicional independiente para controlar el bloqueo.
- ✍ NFS utiliza el esquema de protección UNIX (rwxrwxrwx) y solicita la identificación del usuario.

# Tema I.- Introducción a los Sistemas Operativos Distribuidos.

## Implantación de NFS.

✍ La implantación del código del cliente y servidor es independiente de los protocolos NFS.

### ejemplo SUN)

- ✍ Capa superior: llamadas al sistema. Después de analizar la llamada (OPEN, READ, CLOSE) y verificar los parámetros, llama a la segunda capa.
- ✍ Segunda capa: sistema virtual de ficheros (VFS). Mantiene una tabla, con una entrada por cada fichero abierto, un *nodo-v* (nodo-i virtual).
- ✍ Los *nodos-v* indican si el fichero es local o remoto. Para los remotos, se dispone de información para acceder a ellos.

uso de nodos-v a través de las llamadas MOUNT, OPEN y READ.

- 1) **Montar sistema remoto de ficheros**, administrador sistema ejecuta *mount* con la información de directorio remoto, el directorio local y otros datos adicionales
- 2) *mount* analiza nombre directorio remoto y descubre el nombre de la máquina donde se encuentra.
- 3) Si el directorio existe y está disponible el directorio remoto, el servidor devuelve *file handle*, *mount* lo transfiere al núcleo.
- 4) El núcleo construye un *nodo-v* para el directorio remoto y solicita el código del cliente NFS para crear un *nodo-r* (nodo-i remoto) en sus tablas internas. El *nodo-v* apunta al *nodo-r* (del código de NFS) si es remoto o al *nodo-i* del SO si es local.

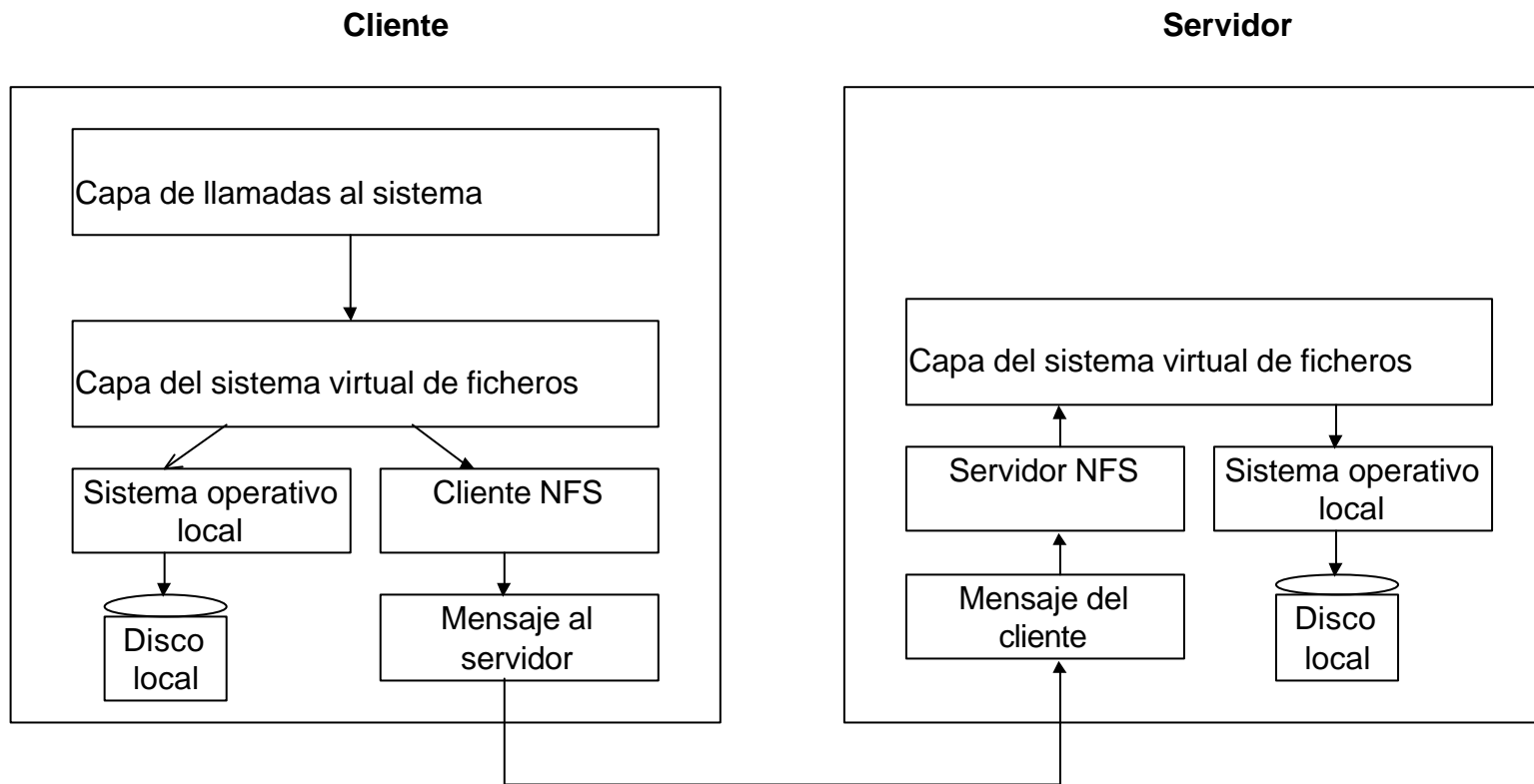
# Tema I.- Introducción a los Sistemas Operativos Distribuidos.

- 1) **Abrir un fichero remoto**, en el análisis de la ruta de acceso se alcanza el directorio en donde se monta el sistema de ficheros remoto.
  - 2) En el nodo-v del directorio encuentra el apuntador al nodo-r.
  - 3) Se pide a NFS-cliente que abra el fichero.
  - 4) NFS-cliente busca el resto de la ruta en el servidor y devuelve *file handle*. Crea en sus tablas un *nodo-r* para el fichero remoto y regresa a la capa VFS, que coloca en sus tablas un nodo-v para el fichero que apunta al *nodo-r*.
- ✍ Todo fichero o directorio abierto tiene un nodo-v que apunta a un nodo-r o a un nodo-i.
  - ✍ Por razones de eficiencia, las transferencias cliente-servidor se realizan en bloques grandes (8k), VFS realiza lecturas adelantadas (*read ahead*).
  - ✍ Evitar tráfico de red mediante *catching*, mantener i-nodos y datos del i-nodo en caché del cliente. Provoca problemas de **coherencia!!!**

## Soluciones parciales:

- ✍ asociar un temporizador a cada bloque de caché, cuando expira la entrada se descarta (3seg para datos y 30seg para directorios).
- ✍ al abrir un fichero con caché, se envía un mensaje al servidor para revisar la hora de la última modificación. Si no coincide se descarta y se utiliza la nueva copia.
- ✍ el temporizador expira cada 30 seg. y todos los bloques modificados en el caché se envían al servidor.
- ✍ NFS no implanta de manera adecuada la semántica de UNIX, la ejecución de un conjunto de programas en cooperación puede producir diversos resultados según la sincronización.
- ✍ NFS resuelve sólo el compartir un sistema de ficheros, no aborda la ejecución de un proceso.

# Tema I.- Introducción a los Sistemas Operativos Distribuidos.



# Tema I.- Introducción a los Sistemas Operativos Distribuidos.

## 1.3.2. Sistemas realmente Distribuidos.

- ✍ NFS es un ejemplo de software débilmente acoplado en hardware débilmente acoplado. El sistema consta de computadoras independientes que pueden compartir sistema de ficheros, entre cliente-servidor se establece un tráfico que obedece a los protocolos NFS.
- ✍ Sería mejor disponer de software fuertemente acoplado en hardware débilmente acoplado (multicomputadoras). Que toda la red de computadoras funcionara como un solo sistema de tiempo compartido.
- ✍ Definición sistema distribuido: *es aquel que se ejecuta en una colección de máquinas sin memoria compartida, pero que aparece ante sus usuarios como una sola computadora*.
- ✍ Algunos autores resumen esta idea como:
  - imagen de un único sistema,
  - uniprosesador virtual.

- ✍ Idea esencial: los usuarios no deben ser conscientes de la existencia de varias CPUs en el sistema.
- ✍ Ningún sistema cumple en su totalidad este requisito, pero hay aproximaciones.

### Características de un sistema distribuido.

- ✍ Mecanismo de comunicación global entre los procesos (local o remota)
- ✍ Esquema global de protección
- ✍ Administración de procesos uniforme. La forma en que se crean, destruyen, inician y detienen los procesos no debe variar entre máquina
- ✍ ? En resumen: No sólo debe existir un único conjunto de llamadas al sistema disponible en todas las máquinas, sino que estas llamadas deben ser diseñadas para que tengan sentido en un ambiente distribuido.
- ✍ Al tener una misma interfaz de llamadas al sistema ? tenemos núcleos idénticos en todas las CPUs del sistema. Esto facilita la coordinación.  
ejemplo) al iniciar un proceso, todos los núcleos deben cooperar en la búsqueda del mejor lugar donde ejecutarlo.
- ✍ Sin embargo, cada núcleo debe tener control sobre sus recursos locales.  
ejemplo) cada núcleo maneja su propia memoria.

# Tema I.- Introducción a los Sistemas Operativos Distribuidos.

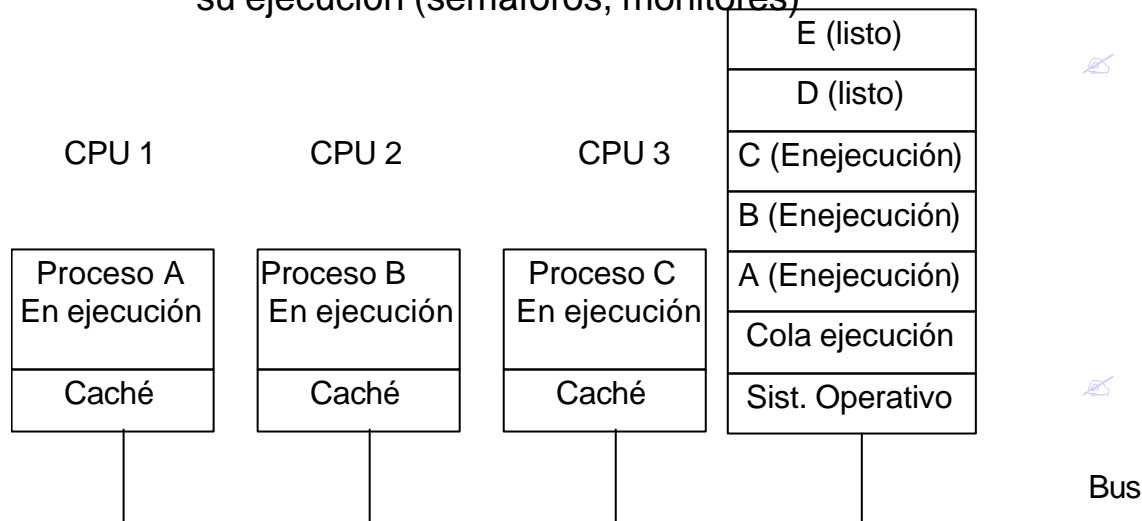
## 1.3.3. Sistemas de multiprocesador con tiempo compartido.

Software fuertemente acoplado en hardware fuertemente acoplado

Solo existe una cola de ejecución alojada en la memoria compartida.

ejemplo) Supongamos que B se bloquea en espera de E/S. El SO debe suspenderlo y buscar otro proceso.

El planificador debe ser una sección crítica para evitar que dos CPU elijan el mismo proceso para su ejecución (semáforos, monitores)



En principio es igual que un proceso se ejecute en la misma CPU o en diversas CPUs.

Pero!! si existe ocultamiento mediante caché ? será preferible elegir la misma CPU suponiendo que ningún otro proceso se ejecutó antes ? permanece su espacio de direcciones en caché.

Mejora: Si un proceso se bloquea por tiempo breve será preferible realizar una espera ocupada que un bloqueo. Se puede conservar la CPU por unos miliseg., por si la E/S terminase pronto.

Este tipo de multiprocesador difiere de los sistemas en red y de los distribuidos en el sistema de archivos. Cada vez que una CPU ejecute una sección crítica o tenga acceso a una tabla central se debe bloquear el acceso al resto.

Se puede intentar asemejar estos sistemas más a los de una CPU haciendo que solo una maquina ejecute el SO, pero de esta forma se convertirá en un cuello de botella.

# Tema I.- Introducción a los Sistemas Operativos Distribuidos.

Elemento	SO de red	SO distribuido	SO multiprocesador
¿Se ve como un uniprocador virtual?	No	Si	Si
¿Todas tienen que ejecutar el mismo SO?	No	Si	Si
¿Cuántas copias del SO existen?	N	N	1
¿Cómo se logra la comunicación?	Archivos compartidos	Mensajes	Memoria compartida
¿Se requiere un acuerdo en los protocolos de la red?	Si	Si	No
¿Existe una única cola de ejecución?	No	No	Si
¿Existe una semántica bien definida para los ficheros compartidos?	Por lo general no	Si	Si



# Tema I.- Introducción a los Sistemas Operativos Distribuidos.

## 1.4. Aspectos de Diseño.

### 1.4.1. Transparencia.

- ✍ Aspecto más importante: conseguir la imagen de un único sistema. Que los usuarios vean un sistema de tiempo compartido de un solo procesador.
- ✍ Transparencia en dos niveles: usuario y programador:  
Más fácil ocultar la distribución al usuario que se dedica a ejecutar órdenes y programas.
- ✍ Concepto escurridizo!!  
ejemplo) Un sistema en donde el acceso a los ficheros remotos se realice mediante el establecimiento explícito de una conexión en red con un servidor remoto no es transparente.
- ✍ Podemos aplicar el concepto de transparencia ? aspectos del sistema distribuido:

- ✍ Transparencia en la localización: los usuarios no conocen la verdadera localización de los recursos hardware/software (CPU, impresoras, ficheros y bases de datos). Ej URLs
  - ✍ Transparencia de Acceso: operaciones idénticas
- ✍ Transparencia de migración: significa que los recursos pueden moverse de una posición a otra sin tener que cambiar sus nombres.
  - ✍ URLs no son transparentes
- ✍ Transparencia de réplica: SO es libre de fabricar copias adicionales de los ficheros y otros recursos sin que lo noten los usuarios.
- ✍ Transparencia frente a fallos: ocultarlos.
  - ✍ Correo electrónico
- ✍ Transparencia en la concurrencia: los usuarios no deben notar la existencia de otros usuarios. Deben resolverse los problemas derivados del acceso a un mismo recurso.
- ✍ Transparencia en el paralelismo: el compilador, el sistema de tiempo de ejecución y el SO deben conjuntamente aprovechar el paralelismo potencial sin que el programador lo sepa. Esto es lo más difícil de conseguir, ahora, los programadores deben programarlo de forma explícita para utilizar varias CPUs.

# Tema I.- Introducción a los Sistemas Operativos Distribuidos.

## 1.4.2. Flexibilidad o Extensibilidad

Dos escuelas con respecto a la estructura de sistemas distribuidos:

- ✍ Cada máquina debe ejecutar un núcleo tradicional que proporcione la mayoría de los servicios (núcleo monolítico).
- ✍ El núcleo debe proporcionar lo menos posible y el grueso de los servicios del SO se obtienen de los servidores al nivel de usuario (micronúcleo).
- ✍ 1.- núcleo monolítico: SO centralizado básico actual, aumentado con las capacidades de red y la integración de servicios remotos. La mayoría de las máquinas tienen disco y administra sus propios sistemas locales de ficheros. Muchos de los sistemas distribuidos actuales utilizan este método.
- ✍ 2.- micronúcleo: es más flexible, porque casi no hace nada. Proporciona sólo cuatro servicios mínimos:
  - ✍ un mecanismo de comunicación entre procesos.
  - ✍ cierta administración de la memoria.
  - ✍ cantidad limitada de planificación y administración de procesos de bajo nivel.
  - ✍ E/S de bajo nivel.

- ✍ No proporciona el sistema de ficheros y directorios, la administración completa de procesos y gran parte de manejo de las llamadas al sistema.
- ✍ Objetivo del micronúcleo: sea reducido. El resto de los servicios se implementan a nivel de usuario. El usuario envía un mensaje al servidor apropiado, éste realiza el trabajo y devuelve el resultado.
- ✍ Sistema muy modular: existe un interfaz bien definida con cada servicio y cada servicio es igual de accesible para todos los clientes independientemente de su posición.
- ✍ Fácil de implantar, instalar y depurar nuevos servicios. La adición o modificación de un servicio no requiere revisar todo el sistema.
- ✍ Los usuarios pueden reescribir sus propios servicios.
- ✍ Ventaja del sistema monolítico: el rendimiento. No obstante hay ejemplos de sistema distribuidos en micronúcleo (amoeba) que muestran gran eficiencia.
- ✍ Se cree que los sistemas micronúcleo irán evolucionando y haciendo sombra a los monolíticos.

# Tema I.- Introducción a los Sistemas Operativos Distribuidos.

## 1.4.3. Confiabilidad

- ✍ Si una de la máquinas falla el sistema siga funcionando aunque con menor rendimiento.
- ✍ Teoría??. En la práctica debe asegurarse el funcionamiento de ciertos servidores para que todo funcione.
- ✍ Lamport definió un sistema distribuido como:  
*aquel del cual no puedo obtener un trabajo debido a que cierta máquina de la cual nunca he oído se ha descompuesto.*

### Aspectos de la confiabilidad:

- ✍ disponibilidad: fracción de tiempo que el sistema se puede utilizar. Se puede mejorar con la redundancia (duplicar piezas clave hardware o software).
- ✍ consistencia: la redundancia aumenta el peligro de la inconsistencia de los datos!!!
- ✍ seguridad: en un sistema distribuido, cuando llega un mensaje a un servidor, éste no tiene una forma sencilla para determinar de quién proviene. El emisor puede mentir en su identificación.

- ✍ tolerancia a fallos: se deben ocultar los fallos a los usuarios. No deben notar la pérdida de servidores, solo notar una cierta degradación en el rendimiento.

# Tema I.- Introducción a los Sistemas Operativos Distribuidos.

## 1.4.4. Desempeño.

- ✍ No nos sirve de nada un sistema transparente, flexible y confiable si es lento!!!
  - ✍ Difícil de lograr!!!
  - ✍ Métricas de desempeño: tiempo de respuesta, rendimiento (trabajos por hora), uso del sistema y cantidad consumida por la red.
  - ✍ **Problema**: la comunicación, factor esencial en un sistema distribuido.
- ejemplo) el envío de un mensaje y obtención de respuesta en una LAN es 1 mseg
- ✍ Optimizar el desempeño supone minimizar el número de mensajes.
  - ✍ Pero la mejor manera de mejorar el desempeño es tener muchas actividades en ejecución paralela ? envío de muchos mensajes.
  - ✍ Evaluar el **tamaño de grano** de todos los cálculos???:

- ✍ el costo de la comunicación no debe exceder al cálculo.
- ✍ los trabajos que implican gran número de pequeños cálculos, que interactúan en gran medida con otros tendrán problemas en los sistemas distribuidos (paralelismo de grano fino).
- ✍ los trabajos que implican grandes cálculos y bajas tasas de interacción (paralelismo de grano grueso) se ajustan mejor a un sistema distribuido.
- ✍ la tolerancia a fallos tiene como contrapartida un precio. Si se duplica la gestión de una solicitud para asegurar su ejecución se aumentará el tráfico en la red y bajará el desempeño.

# Tema I.- Introducción a los Sistemas Operativos Distribuidos.

## 1.4.5. Escalabilidad.

- ✍ En general, los sistemas distribuidos constan de unos cuantos cientos de CPUs.
- ✍ ¿Es posible que las soluciones actuales funcionen para mayor número de máquinas?

Posibles cuellos de botella en enormes sistemas distribuidos:

Conceptos	Ejemplos
Componentes centralizados	un solo servidor de correo para todos los usuarios
Tablas centralizadas	un único directorio telefónico en línea
Algoritmos centralizados	Realización de un ruteo con base en la información completa

- ✍ Hay que evitar que un algoritmo opere mediante recolección de inf. en todos los lugares, para enviarlos a un sola máquina para su procesamiento y después distribuir los resultados. Se deben utilizar algoritmos descentralizados:
  - ✍ Ninguna máquina tiene la inf. completa acerca del estado del sistema.
  - ✍ Las máquinas toman decisiones sólo en base a la inf. disponible de manera local.
  - ✍ El fallo de un máquina no arruina el algoritmo.
  - ✍ No existe un hipótesis implícita de la existencia de un reloj global.
- ✍ Cualquier algoritmo que se base en un reloj fracasará. Es imposible sincronizar de manera precisa todos los relojes.
- ✍ Mientras mayor sea el sistema mayor será la incertidumbre.