Sistemas Distribuidos. Temario.

1. Introducción
2. Comunicación
3. Procesos
4. Nombrado
5. Sincronización
6. Consistencia y replicación
7. Tolerancia a fallos

Tema 1.- Introducción

1. Concepto de Sistema Distribuido
2. Objetivos de los Sistemas Distribuidos
3. Conceptos Hardware
4. Conceptos Software
5. El Modelo Cliente/Servidor

Bibliografía: Capítulo 1 de Tanenbaum

1.- Concepto de Sistema Distribuido

Un Sistema Distribuido es :

- Un conjunto de ordenadores independientes que ofrecen a sus usuarios la imagen de un sistema coherente único.

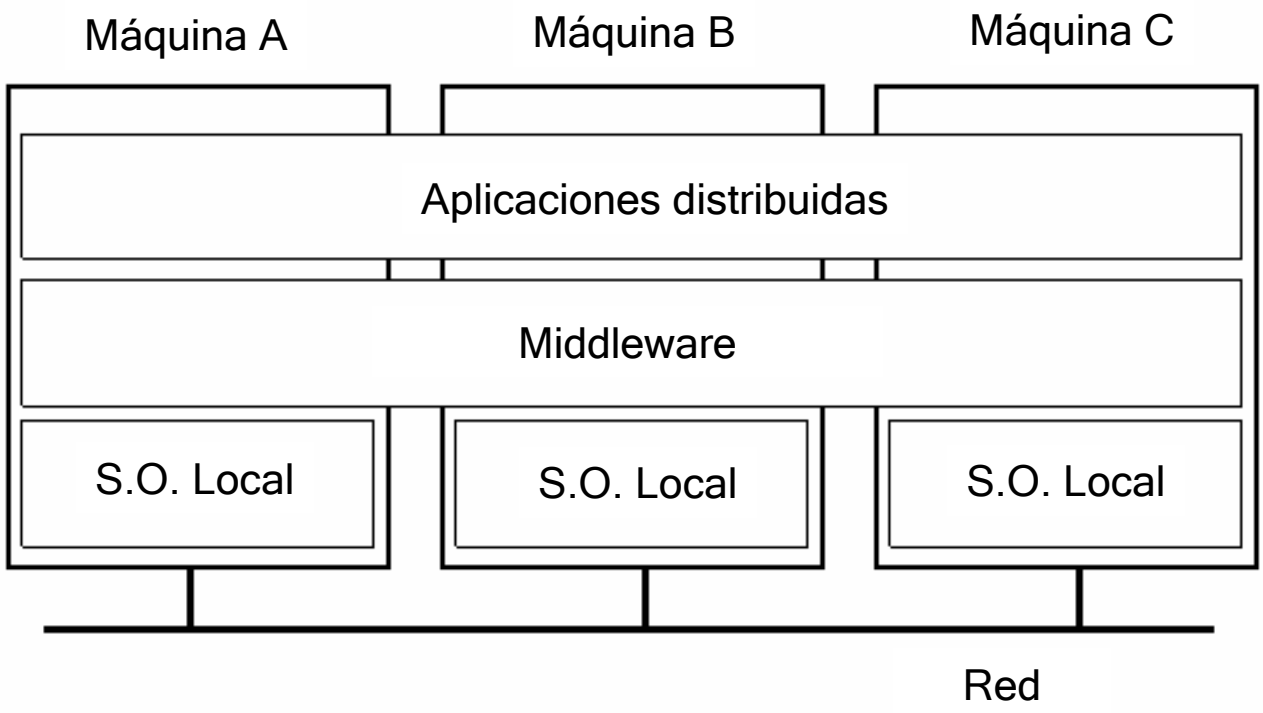
Dos aspectos:

- Hardware: máquinas autónomas
- Software: imagen de sistema único

1.1.- Características de los Sistemas Distribuidos (informal)

- Se ocultan a los usuarios las diferencias entre las máquinas y la complejidad de los mecanismos de comunicación entre máquinas.
- Los usuarios acceden a los sistemas distribuidos de forma homogénea sea cual sea el lugar desde el que lo hagan.
- Los sistemas distribuidos deben ser relativamente sencillos de ampliar, por lo que deben escalar bien.
- Los servicios que ofrecen los sistemas distribuidos deben estar siempre disponibles, aunque determinadas partes de los sistemas fallen o no estén disponibles por reparación, ampliación, sustitución, etc.
- Los sistemas distribuidos habitualmente se estructuran en niveles, siendo la estructuración como “middleware” una de las más frecuentes.

1.2.- Estructuración en ‘middleware’



Nótese que la capa “middleware” abarca varias máquinas.

1.3.- Ejemplos de sistemas distribuidos

- Ordenadores para la docencia en laboratorios
- El servicio de nombres de Internet
- Los servicios de mensajería de Internet
- Los servidores de Google
- WWW

Tema 1.- Introducción

1. Concepto de Sistema Distribuido
2. Objetivos de los Sistemas Distribuidos
3. Conceptos Hardware
4. Conceptos Software
5. El Modelo Cliente/Servidor

Bibliografía: Capítulo 1 de Tanenbaum

2.- Objetivos de los sistemas distribuidos

1. Facilitar el acceso de los usuarios a recursos remotos.

2. Transparencia de distribución
3. Concepción como sistemas abiertos
4. Escalabilidad

2.1.- Conectar usuarios a recursos

- El principal objetivo de todo SD es facilitar a los usuarios el acceso a recursos remotos.
- Recurso debe entenderse en sentido amplio.
- Ventajas:
  - Economía de recursos
  - Facilita la compartición de recursos entre usuarios.
- Desventajas:
  - Seguridad

2.2.- Transparencia de distribución

- Ocultar el hecho de que los procesos y los recursos están físicamente distribuidos sobre diferentes ordenadores. .
- Diferentes aspectos de transparencia.

2.2.1.- Tipos de transparencia

Transparencia	Descripción
Acceso	Oculta diferencias en la representación de los datos y en cómo se accede a los recursos
Ubicación	Oculta dónde se ubican los recursos
Migración	Oculta el hecho de que un recurso puede migrar de un lugar a otro
Reubicación	Oculta el hecho de que un recurso puede moverse de un lugar a otro mientras se utiliza
Replicación	Oculta el hecho de que un recurso puede tener más de una réplica, haciendo indistinguible a los usuarios la réplica que realmente utilizan
Concurrencia	Oculta el hecho de que un recurso pueda ser utilizado simultáneamente por más de un usuario.
Fallos	Oculta el fallo y la recuperación de los recursos
Persistencia	Oculta el hecho de que un recurso esté ubicado en memoria volátil o en memoria persistente.

Diferentes formas de transparencia en sistemas distribuidos

2.2.2.- Grados de transparencia

- Muchas veces la transparencia total es inabordable o poco conveniente:
  - Por ejemplo: dominios .es (conviene saber que se trata de páginas de España)
  - Transparencia ante fallos: teóricamente imposible en determinados sistemas.
    - Nodo que falla indistinguible respecto a nodo que va lento.
    - Imposible saber si un servidor completó una operación antes de fallar.
- Muchas veces hay que buscar punto de equilibrio entre eficiencia y transparencia, pues la transparencia suele ser cara:
  - Escribir en disco en cada operación sobre un servidor para enmascarar fallos
  - Mantener las caches de páginas Web siempre actualizadas con precisión

2.3.- Concepción como sistemas abiertos

- Decimos que un **sistema distribuido es abierto** si se ajusta a estándares que describen la sintaxis y la semántica de los servicios, estándares que deben potenciar la **flexibilidad.**
- Con la adecuación a estándares se logra:
  - **Interoperabilidad**: sistemas que se ajusten al mismo estándar pueden colaborar entre ellos.
  - **Portabilidad de las aplicaciones**: aplicaciones desarrolladas para una implementación concreta de un estándar deben funcionar para otras implementaciones del estándar.
- Ejemplos:
  - CORBA, IDL
  - HTML, XML
  - ODBC

2.3.- Concepción como sistemas abiertos

- La flexibilidad del sistema debe permitir:
  - Configurar el sistema combinando módulos distintos, de fabricantes posiblemente diferentes.
  - Añadir nuevos módulos
  - Reemplazar unos módulos por otros sin afectar al resto del sistema.
- La flexibilidad se logra:
  - Diseñando sistemas formados por pequeñas componentes claramente diferenciadas que interactúen entre ellas mediante interfaces claras.
  - Separando políticas de mecanismos.

2.3.1.- Separación de políticas y mecanismos

- Los sistemas deben construirse de tal forma que proporcionen una elevada riqueza de mecanismos.
- Las políticas deben ser configurables:
  - Permitiendo al usuario definir los parámetros
  - Permitiendo al usuario implementar nuevas políticas.
- Ejemplos:
  - Calidad del servicio del video/audio
  - Elección entre diferentes tipos de cifrado
  - Número de réplicas en sistemas replicados
  - Consistencia entre réplicas o entre caché y datos primarios.

2.4.- Escalabilidad

- Debe considerarse la escalabilidad en tres vertientes:
  - Escalabilidad de tamaño: Los sistemas deben permitir el crecimiento en procesos, nodos, usuarios, sin que se afecte al funcionamiento del sistema.
  - Escalabilidad de distancia: Los sistemas deben permitir que se reubiquen componentes de los sistemas en lugares físicamente distantes entre ellos.
  - Escalabilidad administrativa: Los sistemas deben ser fácilmente configurables incluso si abarcan organizaciones diferentes.
- La mayoría de veces sólo se menciona escalabilidad de tamaño, y muchas veces se recurre (equivocadamente) a soluciones temporales (aumentar la potencia de los servidores)

2.4.1.- Escalabilidad de tamaño

- Ejemplos donde se manifiesta necesidad de escalabilidad de tamaño, muchas veces difícil de solucionar

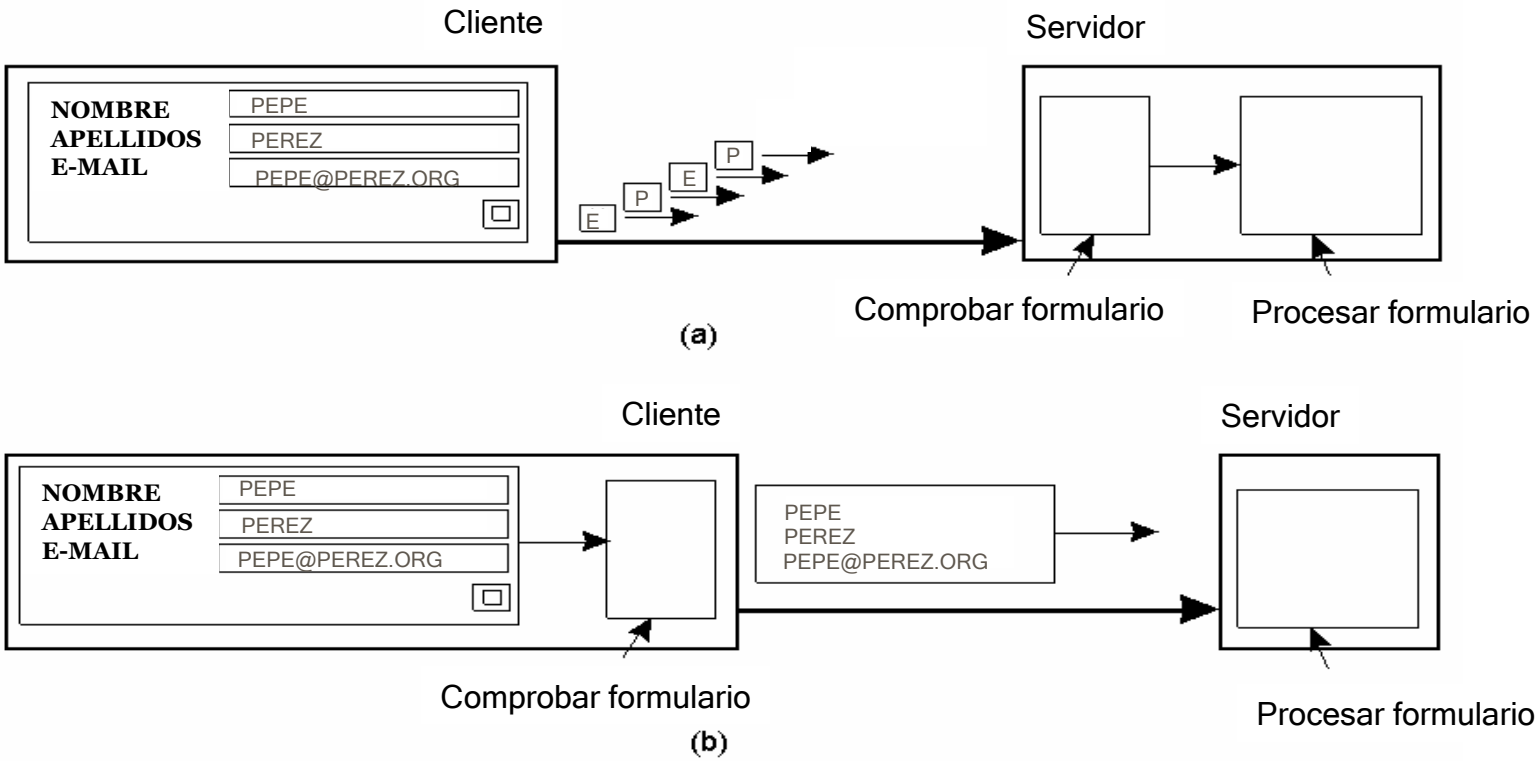
Situación	**Ejemplo**
Servicios centralizados	Servidores especializados para cálculos científicos
Datos centralizados	Registro civil, hacienda, etc.
Algoritmos centralizados	Encaminamiento basado en información completa

2.4.1.- Escalabilidad de tamaño

Técnicas

1. Distribución: dividir datos y/o computaciones entre diferentes ordenadores. Ejemplos:
   - Mover los cálculos a los clientes en los sistemas Web.
   - Particionar los dominios de DNS en diferentes zonas administrativas.
   - Crear algoritmos descentralizados (escalables)
2. Replicación: hacer copias de los datos en diferentes lugares. Ejemplos:
   - Sistemas de ficheros replicados. Fundamentalmente para tolerar fallos.
   - Bases de datos replicadas
   - Sistemas Web como espejos (mirroring)
3. Caching: proporcionar copias de los datos a los clientes
   - Caches de Web
   - Caches de ficheros

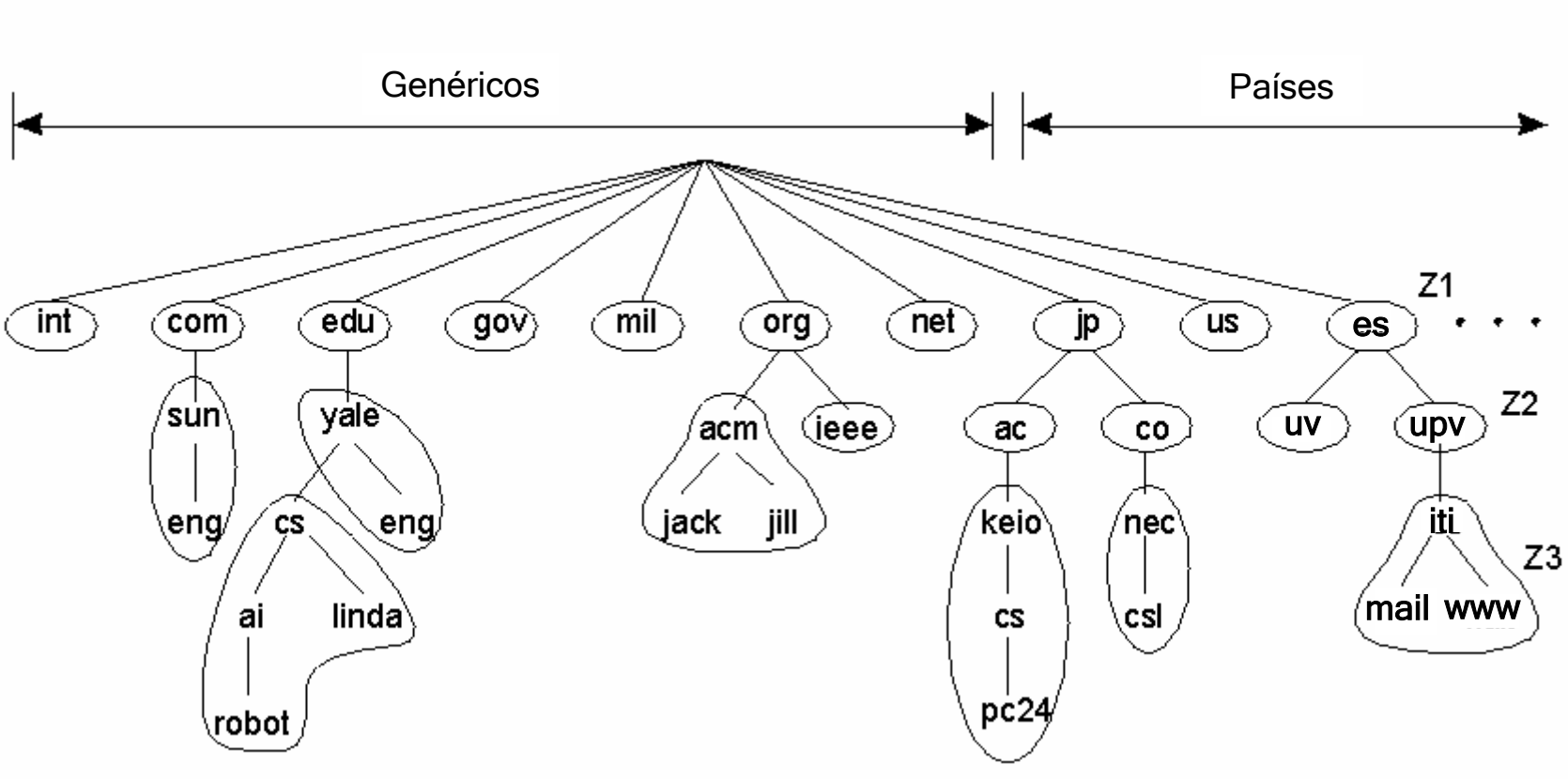
Escalabilidad por distribución



Ejemplo de **escalabilidad por distribución de los cálculos**: La diferencia entre permitir que los formularios sean comprobados por:

a) Un servidor
b) Un cliente

Escalabilidad por distribución



Ejemplo de **escalabilidad por distribución de los datos**: División de los DNS en zonas

Escalabilidad por distribución

Escalabilidad por distribución de los algoritmos. Creación de algoritmos descentralizados

- Conviene diseñar algoritmos que se ejecuten en varias máquinas en lugar de algoritmos que necesiten todos los datos para ejecutarse
- Se logra:
  - Distribuir la carga computacional y espacial entre diferentes máquinas
  - Distribuir la carga de las comunicaciones en la red.
- Características de los algoritmos descentralizados:
  - Ninguna máquina tiene toda la información completa que requiere el algoritmo
  - Las máquinas únicamente toman decisiones en base a su conocimiento local
  - El fallo de una máquina no impide que el algoritmo progrese
  - No se asume que exista un reloj físico global (para permitir escalabilidad de distancia)

Escalabilidad por replicación/caching

- Pese a que parece sencillo, presenta un problema importante.
- Tener múltiples copias de la misma entidad puede provocar que aparezcan inconsistencias: modificar sólo una copia hace que ésta sea diferente del resto.
- Hacer que todas las copias sean mutuamente consistentes requiere de protocolos de sincronización global ante cada modificación
- **La sincronización global no escala bien**
- Si pueden tolerarse inconsistencias, se pueden llegar a buenas soluciones, pero depende de cada aplicación en concreto.

2.4.2.- Escalabilidad de distancia

- Debe considerarse en el diseño de algoritmos descentralizados.
- Debe considerarse los efectos que supondrán a los algoritmos:
  - Los cambios en los tiempos de las comunicaciones debidos a cambios geográficos de los distintos ordenadores (WAN > LAN)
  - Las diferencias en cuanto a la fiabilidad de las comunicaciones (LAN > WAN)

2.4.3.- Escalabilidad administrativa

- Al instalar un sistema sobre diferentes organizaciones suelen aparecer dificultades relacionadas con la seguridad:
  - Diferentes organizaciones tienen implantadas diferentes políticas
  - Los administradores de los sistemas no suelen tener acceso a los sistemas de otras organizaciones.
  - Los usuarios suelen fiarse únicamente de sus propios administradores.
  - Deben implantarse mecanismos que protejan a cada organización del resto y además del propio sistema.

Tema 1.- Introducción

1. Concepto de Sistema Distribuido
2. Objetivos de los Sistemas Distribuidos
3. Conceptos Hardware
4. Conceptos Software
5. El Modelo Cliente/Servidor

Bibliografía: Capítulo 1 de Tanenbaum

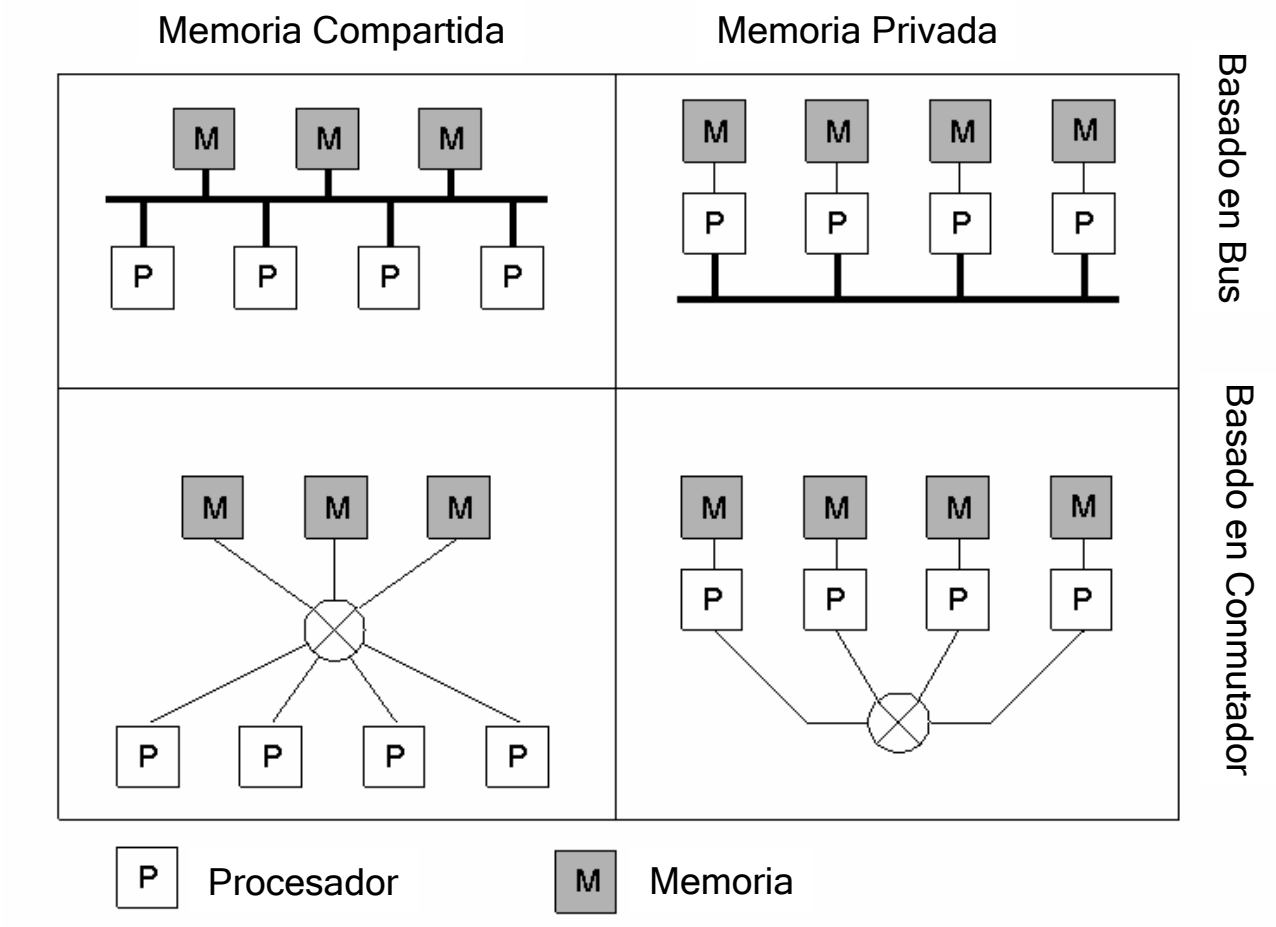
3.- Conceptos Hardware

1. Multiprocesadores
2. Multicomputadoras
3. Redes de ordenadores

3.1.- Multiprocesadores y Multicomputadoras

- Multiprocesadores: Cada CPU tiene acceso a toda la memoria
  - Diferentes tipos de configuraciones de la red (bus, conmutadores) para conectar CPUs con memorias
- Multicomputadoras: Cada CPU tiene acceso únicamente a su memoria privada.
  - Diferentes tipos de red para conectar las distintas CPUs entre ellas (bus, conmutadores)

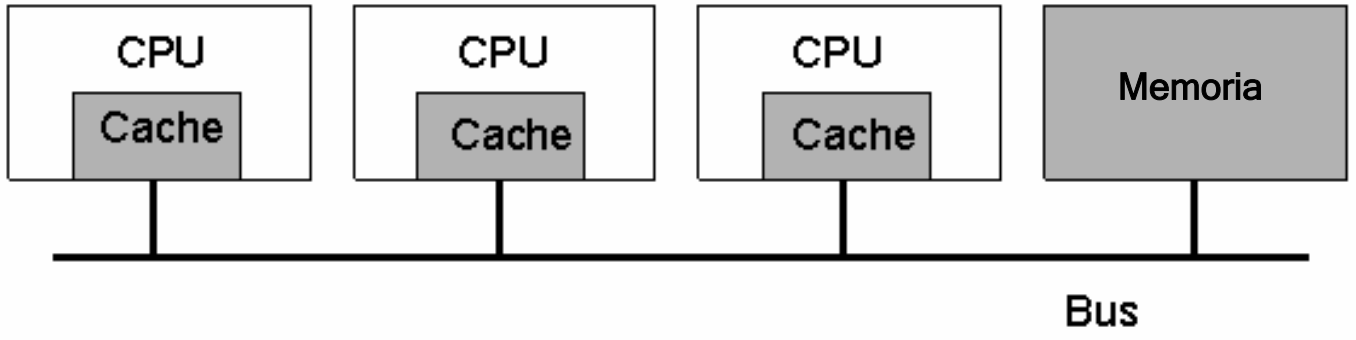
3.1.- Multiprocesadores y Multicomputadoras



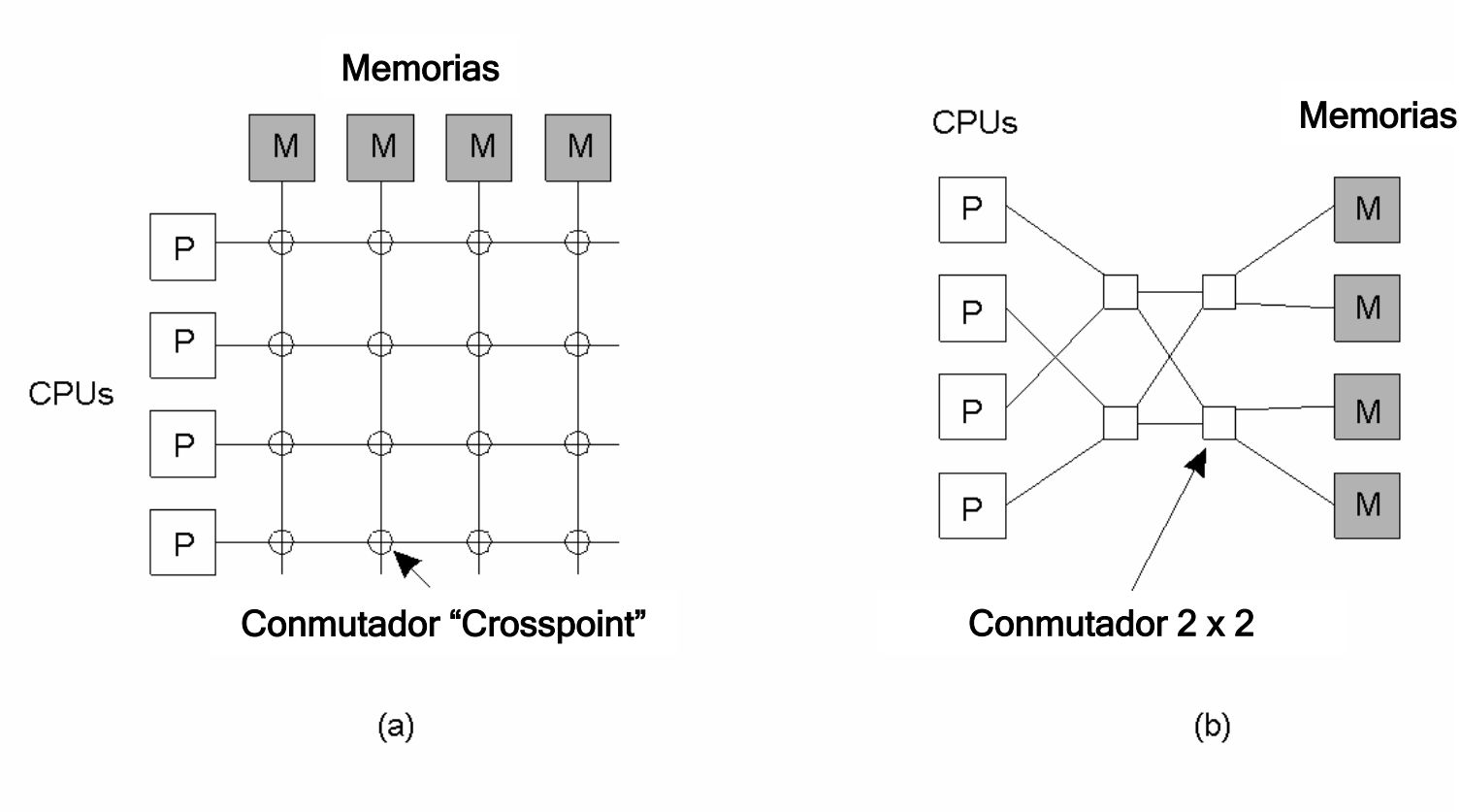
Configuraciones básicas de multiprocesadores y multicomputadoras

3.1.- Multiprocesadores y Multicomputadoras

- Un multiprocesador basado en bus.

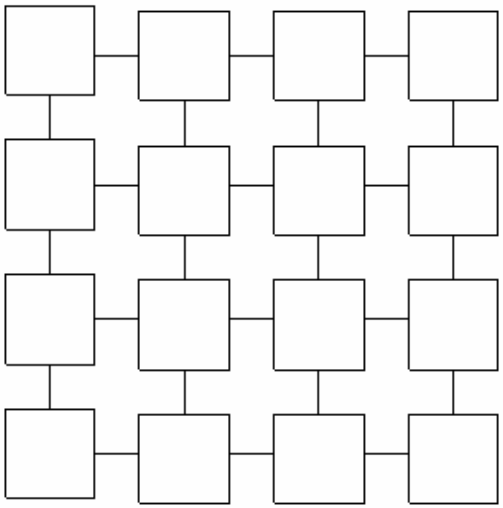


3.1.- Multiprocesadores y Multicomputadoras

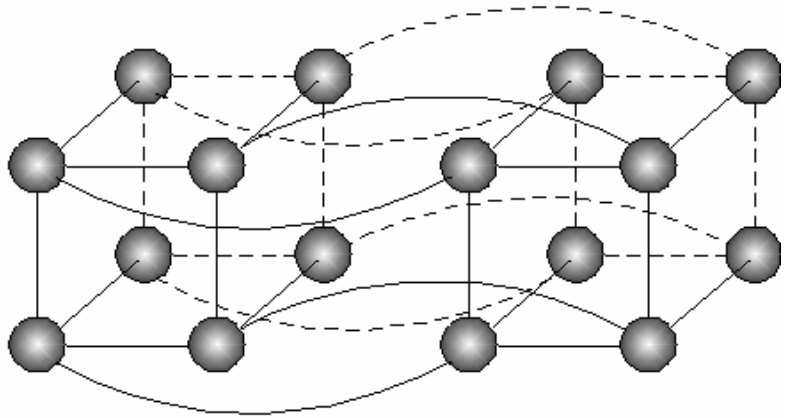


- Multiprocesadores con conmutadores

3.1.- Multiprocesadores y Multicomputadoras



(a)



(b)

a) Multicomputadora en Grid
b) Multicomputadora en Hipercubo

3.2.- Redes de ordenadores

- Altos grados de heterogeneidad en los nodos:
  - Computadoras de altas prestaciones para computación en paralelo (tanto multiprocesadores como multicomputadoras)
  - Ordenadores de gama alta (servidores)
  - Ordenadores convencionales de escritorio
  - Ordenadores portátiles
  - Estaciones de trabajo multimedia
- Altos grados de heterogeneidad en las redes:
  - Redes de área local de Gigabit
  - Conexiones inalámbricas
  - Conmutadores de alta velocidad para WANs
  - Conexiones de baja velocidad punto a punto

Idealmente un sistema distribuido ocultará todas estas diferencias

Tema 1.- Introducción

1. Concepto de Sistema Distribuido
2. Objetivos de los Sistemas Distribuidos
3. Conceptos Hardware
4. Conceptos Software
5. El Modelo Cliente/Servidor

Bibliografía: Capítulo 1 de Tanenbaum

4.- Conceptos Software

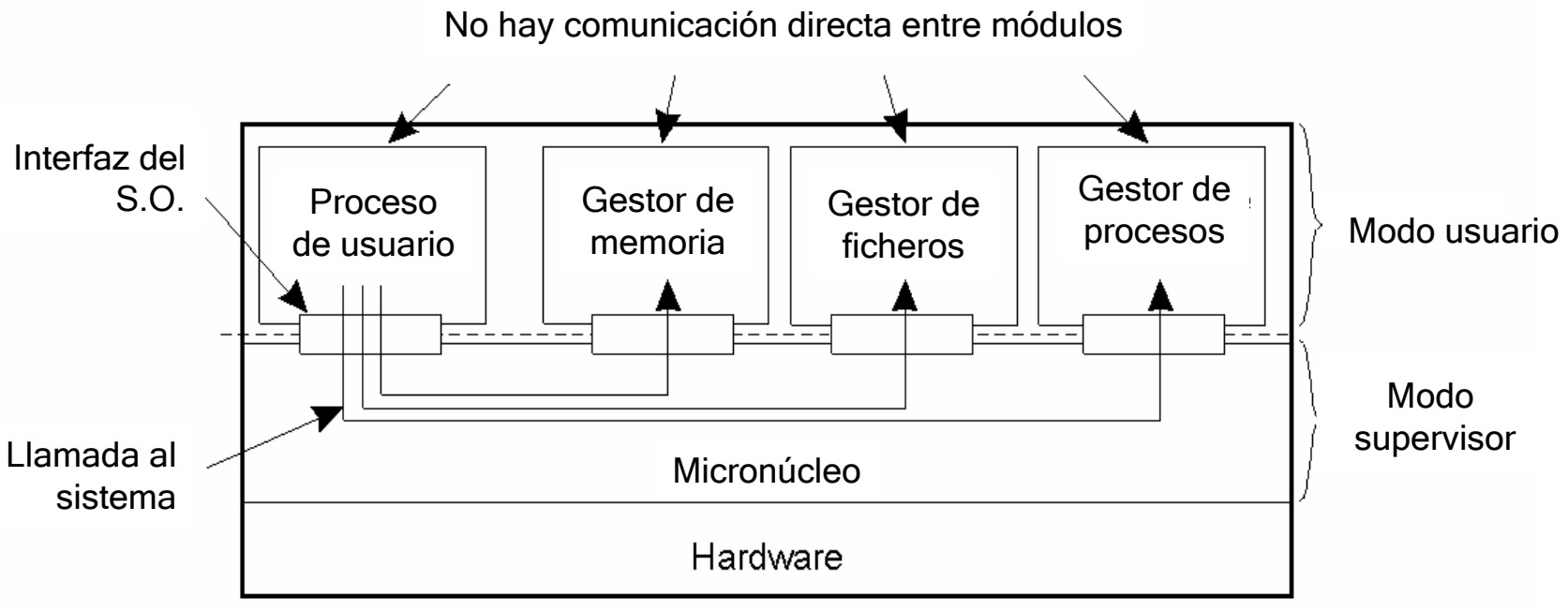
1. Sistemas operativos uniprocesador
2. Sistemas operativos multiprocesador
3. Sistemas operativos distribuidos
4. Sistemas operativos en red
5. Middleware

4.1.- Sistemas operativos uniprocesador

- **Objetivo**: Proporcionar a los usuarios y aplicaciones una forma sencilla y eficiente de compartir los recursos del ordenador (CPU, memoria, discos, dispositivos)
- Actualmente los sistemas operativos proporcionan una imagen de **máquina virtual** a las aplicaciones, de forma que éstas accedes a los recursos como si todos los recursos estuvieran dedicados a ellas.
- Un aspecto importante de la compartición de recursos es la seguridad: para ello el sistema ofrece servicios para acceder a los recursos y sólo se podrá acceder a los recursos a través de estos servicios.
- Para implementar esta compartición segura de recursos las CPUs cuentan al menos con dos modos de funcionamiento: modo supervisor y modo usuario.
- El sistema operativo se ejecuta en modo supervisor, teniendo entonces control sobre todos los recursos, y las aplicaciones se ejecutan en modo usuario.
- Tipos de sistemas operativos uniprocesador:
  - Monolíticos: todo el sistema se ejecuta en modo supervisor
  - Micronúcleo: sólo una pequeña parte del sistema se ejecuta en modo supervisor

4.1.- Sistemas operativos uniprocesador

- Organización de un sistema operativo como micronúcleo.



4.2.- Sistemas operativos multiprocesador

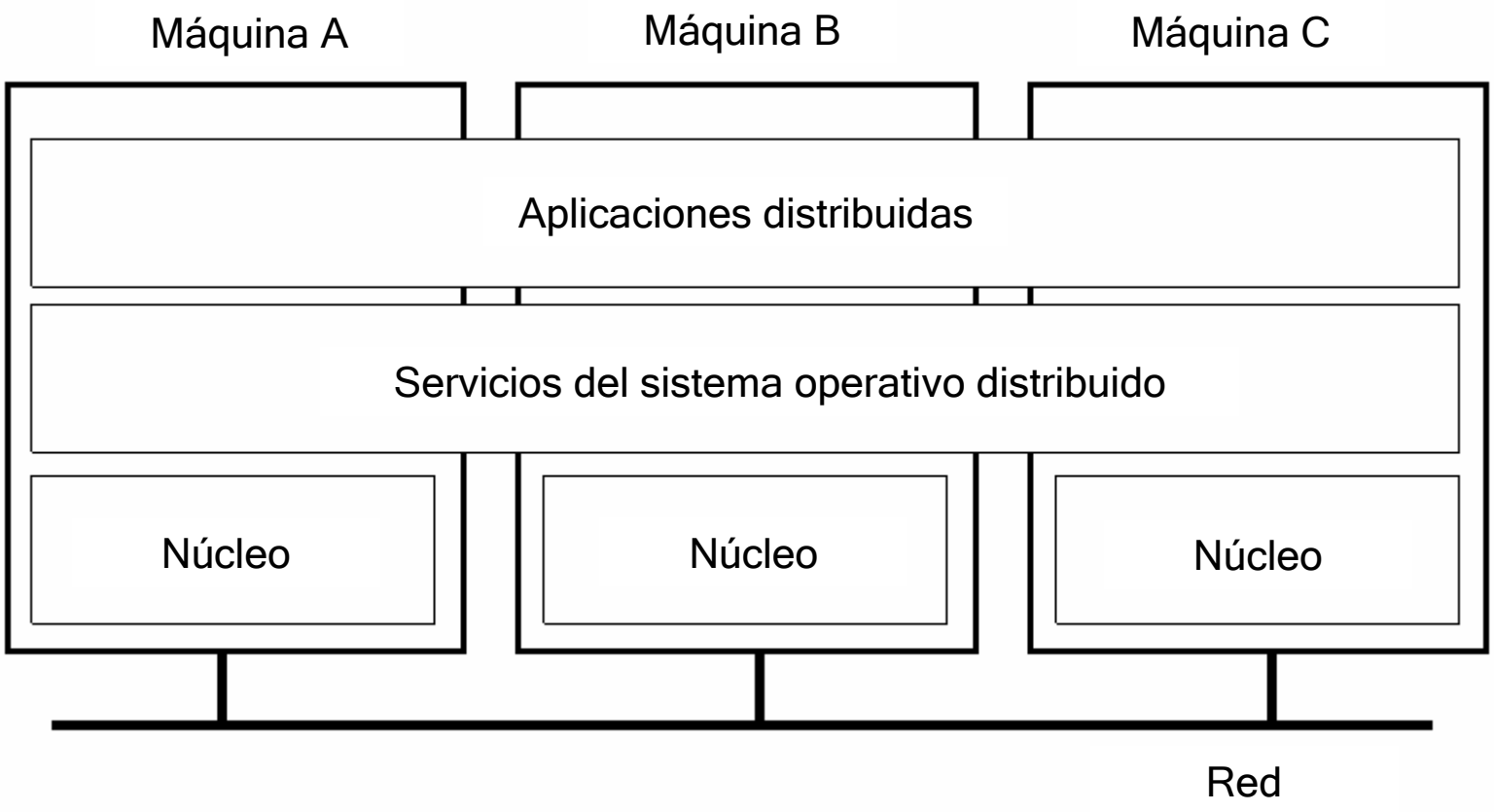
- Extensión de los sistemas operativos uniprocesador a varias CPUs.
- Extensión más sencilla en arquitecturas micronúcleo que en arquitecturas monolíticas.
- Los mismos objetivos que los sistemas operativos uniprocesador, donde adicionalmente se pretende:
  - Ocultar el hecho de disponer de varias CPUs
  - Mejorar el rendimiento global del sistema
- Problema: diferentes CPUs acceden concurrentemente a la memoria, que está compartida entre todas ellas:
- Para sincronizar el acceso concurrente a la memoria se suelen utilizar las mismas primitivas de sincronización que las utilizadas para sistemas multitarea:
  - Semáforos, cerrojos, monitores, etc.

4.3.- Sistemas operativos distribuidos

- Estructura totalmente diferente a la estructura de los sistemas operativos multiprocesador, pues no hay memoria compartida. Sólo es posible la comunicación entre ordenadores mediante **paso de mensajes.**
- Los sistemas operativos distribuidos se realizan sobre:
  - Multicomputadoras: el mismo sistema operativo sobre cada ordenador.
  - Redes de ordenadores: sistemas operativos diferentes en cada ordenador, ofreciendo todos ellos la misma interfaz.

4.3.- Sistemas operativos distribuidos

- Estructura general de un sistema operativo distribuido



4.3.- Sistemas operativos distribuidos

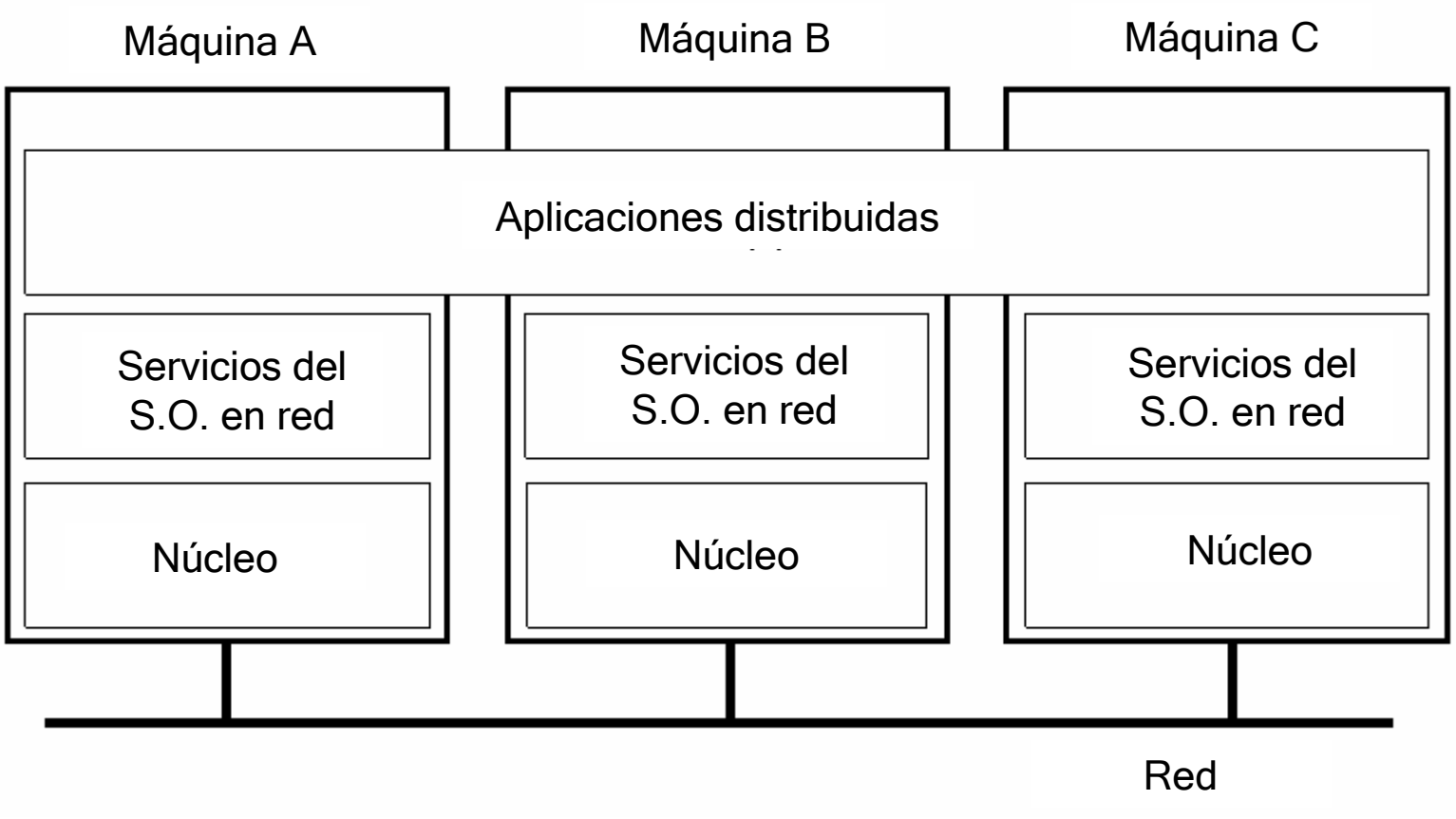
- Cada núcleo gestiona los recursos locales: CPU, discos, etc.
- La capa del sistema operativo distribuido, ofrece una imagen de máquina virtual a las aplicaciones, gestionando los recursos ofrecidos por cada núcleo.
- La capa de sistema operativo distribuido puede ofrecer a las aplicaciones:
  - Primitivas de paso de mensajes
  - Memoria compartida (construida por software sobre paso de mensajes)
- Complejidad:
  - En general no hay mecanismos sencillos de comunicación global (broadcast)
  - En general no hay mecanismos de sincronización global (no hay dispositivos físicamente compartidos: reloj, memoria, etc).
  - Es necesario el empleo de mecanismos distribuidos de gestión de recursos: complejos, aparecen interbloqueos, bajo rendimiento.

4.4.- Sistemas operativos en red

- Se construyen sobre redes de ordenadores heterogéneas **sin** la pretensión de ofrecer una **imagen única** de sistema operativo.
- Cada ordenador cuenta con un sistema operativo completo.
- Los diferentes ordenadores, se conectan en red y sobre la red se apoyan una serie de servicios.
- Claramente más sencillos que los sistemas operativos distribuidos, donde se penaliza la transparencia.
- Una de las pocas ventajas de estos sistemas es la facilidad para añadir o eliminar nodos sin afectar al resto.

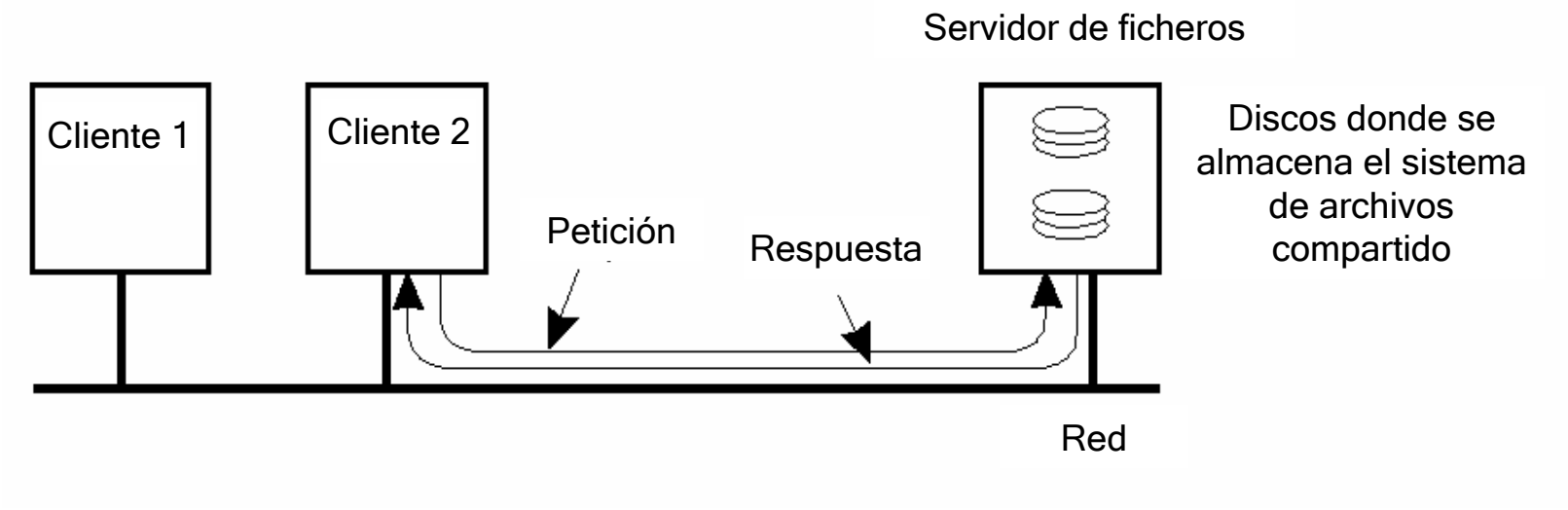
4.4.- Sistemas operativos en red

- Estructura general de un sistema operativo en red.



4.4.- Sistemas operativos en red

- Ejemplo: dos clientes y un servidor como sistema operativo en red

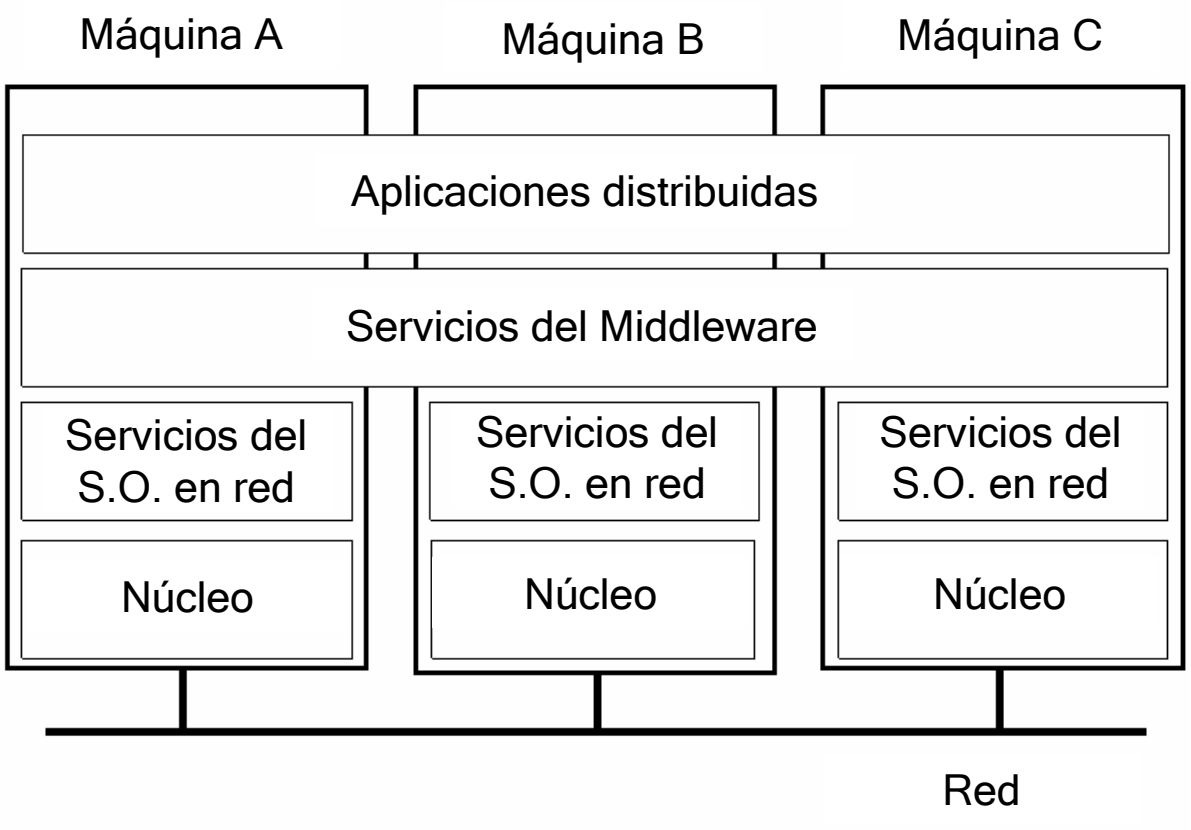


4.5.- Middleware

- Se pretende tener la escalabilidad y la flexibilidad de los sistemas operativos en red y la transparencia y facilidad de uso de los sistemas operativos distribuidos.
- Se construye una capa de software sobre un sistema operativo en red al que llamaremos middleware.
- **Objetivo**: ocultar la heterogeneidad de los sistemas operativos en red.
- Algunas características:
  - Cada S.O. no tiene porqué conocer sobre la existencia de los demás nodos.
  - El S.O. de cada nodo no tiene porqué ser el mismo.
  - Los servicios se distribuyen de forma transparente sobre los diferentes nodos.

4.5.- Middleware

- Estructura general de un sistema distribuido como middleware



4.5.- Middleware

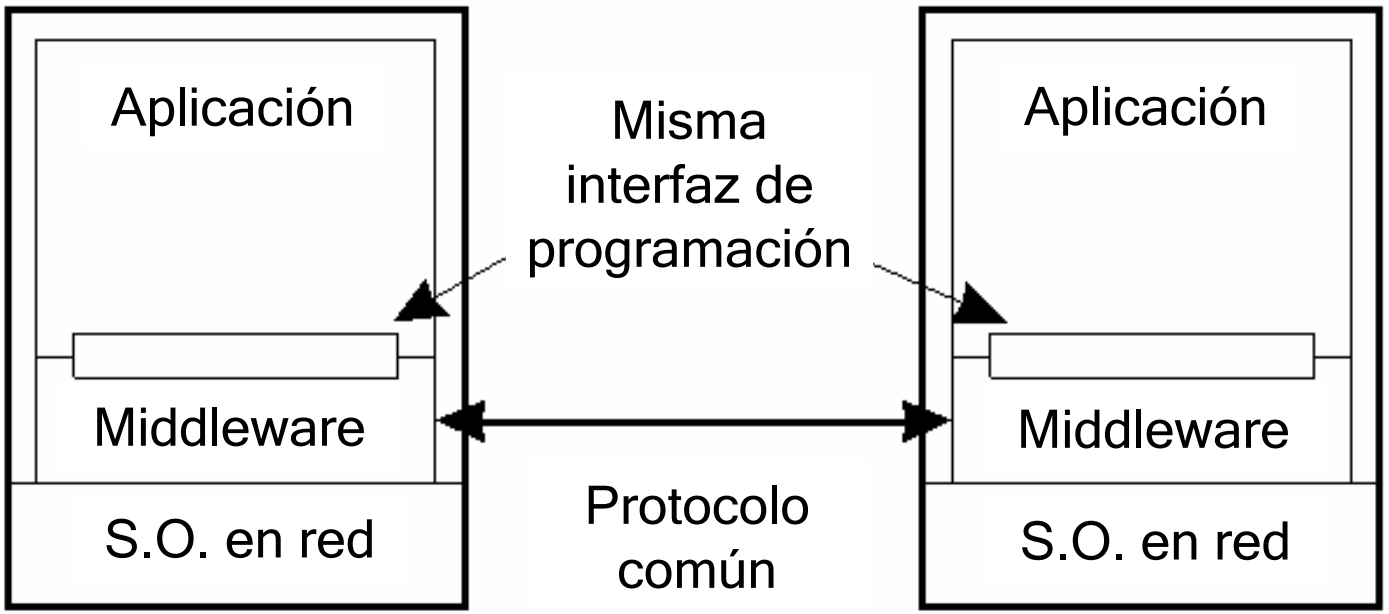
- Motivación: Integrar aplicaciones basadas en red es complejo.
  - Diferentes departamentos usan diferentes sistemas operativos en red.
  - Interoperabilidad sólo aparece a nivel de servicios muy básicos de los sistemas operativos en red.
  - Necesidad de servicios federados:
    - Combinar diferentes bases de datos, pero ofreciendo la imagen de una base de datos única.
    - Servicios de Internet comunes a toda una organización, apoyándose sobre sistemas de información existentes.
    - Permitir transacciones que abarquen diferentes bases de datos.
    - …
- Requerimiento: mantener los sistemas operativos actuales.

4.5.- Middleware

Servicios del middleware

- Servicios de comunicación: pasar de comunicación basada en sockets a:
  - RPC (Llamadas a procedimiento remoto)
  - ROMI (Invocaciones a objetos remotos)
  - Sistemas de notificación de eventos
  - Sistemas de 'streams' o colas avanzados.
- Servicios para sistemas información: servicios que permiten gestionar datos de manera distribuida:
  - Servicios de nombres distribuidos
  - Servicios de directorio y búsqueda distribuidos
  - Servicios de seguimiento de recursos móviles
  - Servicios de persistencia, caching y replicación de datos
- Servicios de control: servicios que permiten controlar a las aplicaciones cómo, cuándo y dónde acceden a los datos:
  - Servicios de transacciones distribuidas
  - Servicios de migración de código
- Servicios de seguridad:
  - Servicios de autenticación y autorización
  - Servicios de cifrado
  - Servicios de auditoría

4.5.- Middleware



- En un sistema distribuido abierto basado en middleware, los protocolos utilizados por cada nivel del middleware deben ser los mismos, de igual forma, el middleware debe ofrecer un conjunto de interfaces idénticas a las aplicaciones.

4.6.- Comparación entre sistemas

Sistema	Descripción	Objetivo Principal
SOD	Sistema Operativo fuertemente acoplado para multicomputadoras y redes de ordenadores homogéneos.	Ocultar y gestionar recursos hardware
SOR	Sistema operativo débilmente acoplado para redes de ordenadores heterogéneos (LAN y WAN)	Ofrecer servicios locales a clientes remotos
Middleware	Capa adicional encima de un SOR, implementando servicios de propósito general.	Proporcionar transparencia de distribución

- Una pequeña comparación entre:
  - SOD (Sistemas Operativos Distribuidos)
  - SOR (Sistemas Operativos en Red)
  - Middleware

4.6.- Comparación entre sistemas

Elemento	S.O. Distribuido		S.O. en red	S.O. basado en middleware
	Multiproc.	Multicomp.		
Grado de transparencia	Muy alto	Alto	Bajo	Alto
Mismo SO en cada nodo	Si	Si	No	No
Número de copias del SO	1	N	N	N
Mecanismo de comunicac.	Memoria compartida	Mensajes	Ficheros	Depende
Gestión de recursos	Global, central	Global, distribuido	Por nodo	Por nodo
Escalabilidad	No	Moderado	Si	Depende
Sistema Abierto/Cerrado	Cerrado	Cerrado	Abierto	Abierto

Tema 1.- Introducción

1. Concepto de Sistema Distribuido
2. Objetivos de los Sistemas Distribuidos
3. Conceptos Hardware
4. Conceptos Software
5. El Modelo Cliente/Servidor

Bibliografía: Capítulo 1 de Tanenbaum

5.- El modelo cliente/servidor

1. Modelo básico cliente/servidor
2. Descomposición de aplicaciones en capas
3. Arquitecturas cliente/servidor

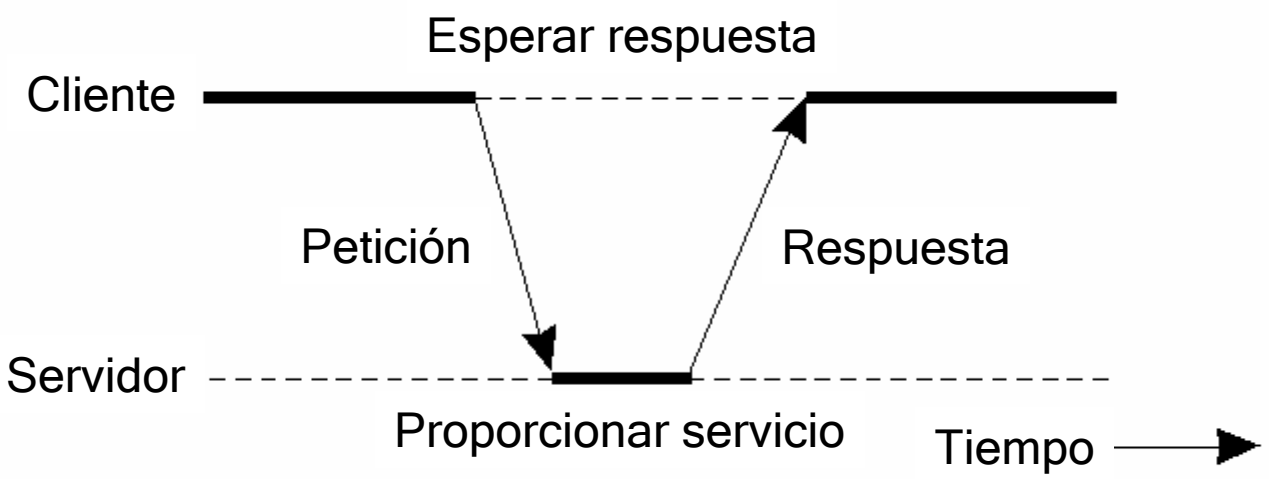
5.1.- Modelo básico cliente/servidor

Características:

- Hay procesos que ofrecen servicios
- Hay procesos que usan servicios
- Los clientes y los servidores pueden estar distribuidos en diferentes máquinas.
- El mecanismo general de comunicación consiste en:
  - Cliente envía petición y espera respuesta.
  - Servidor recibe petición, procesa y contesta con respuesta.

5.1.- Modelo básico cliente/servidor

- Interacción general entre cliente y servidor



5.1.- Modelo básico cliente/servidor

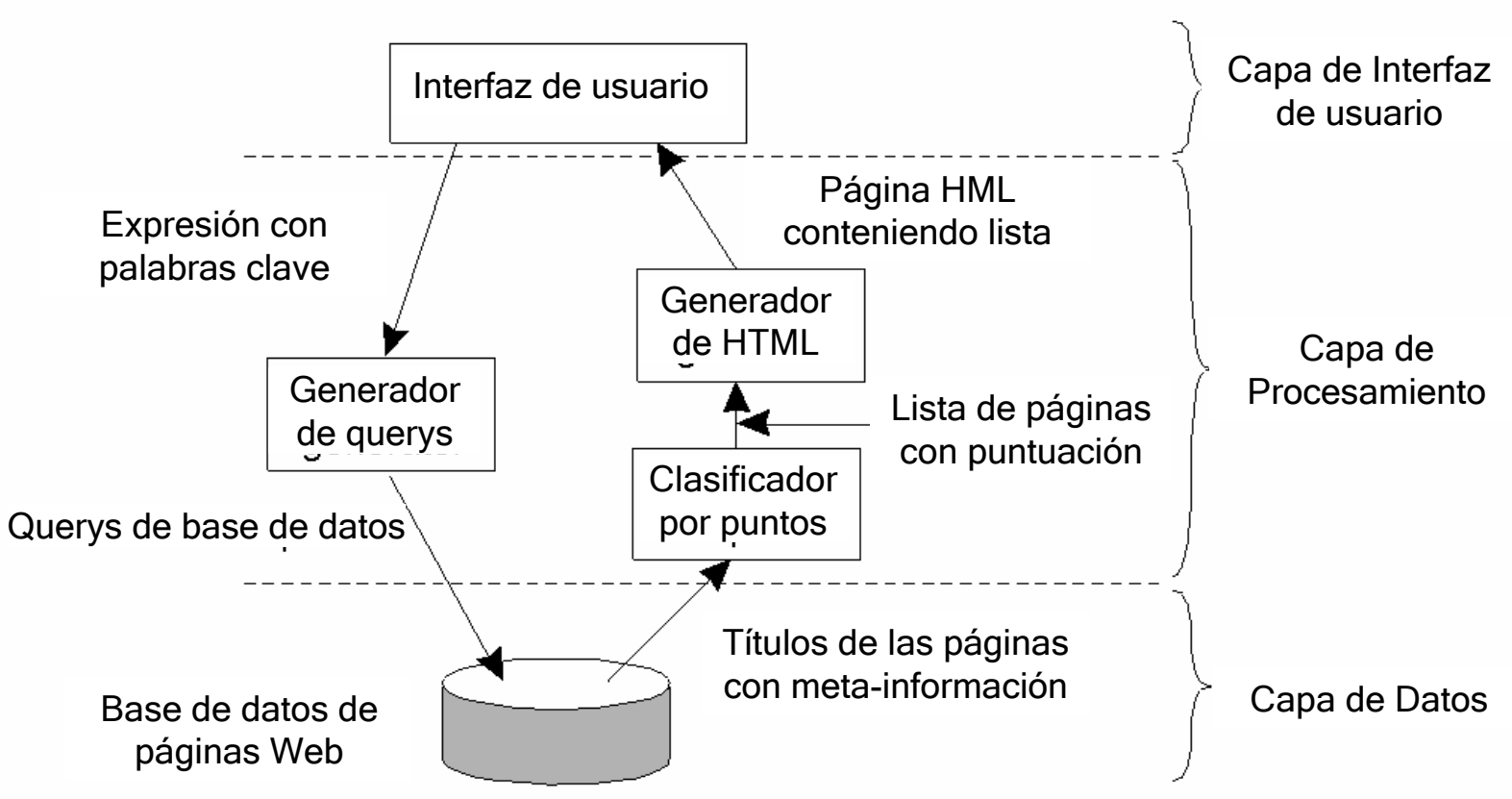
- Los servidores: en general proporcionan servicios relacionados con algún recurso compartido:
  - Servidores para sistemas de ficheros, bases de datos, etc.
  - Servidores para documentos compartidos, entrelazados entre ellos (Web, Lotus Notes, …)
  - Servidores de aplicaciones compartidas
  - Servidores de objetos compartidos
- Los clientes: permiten el acceso a los servicios remotos:
  - La interfaz de programación transforma peticiones locales al cliente en peticiones y espera de respuestas del servidor.
  - Dispositivos sencillos (lectores de códigos de barras, teléfonos móviles, etc)
  - Ordenadores que proporcionan interfaces de usuario específicas para cada servicio.
  - Ordenadores que proporcionan interfaces de usuario para combinar el acceso a diferentes servicios.

5.2.- Descomposición de aplicaciones en capas

- La descomposición tradicional en tres capas:
  - La capa de **interfaz** de usuario, contiene las componentes que forman las interfaces de usuario de las aplicaciones.
  - La capa de **procesamiento** (lógica de negocio), contiene las reglas de funcionamiento de la aplicación, sin integrar datos concretos.
  - La capa de **datos,** contiene los datos que los clientes manipulan mediante las demás componentes de la aplicación.

5.2.- Descomposición de aplicaciones en niveles

- Ejemplo: la organización general de un motor de búsqueda de Internet en 3 capas.

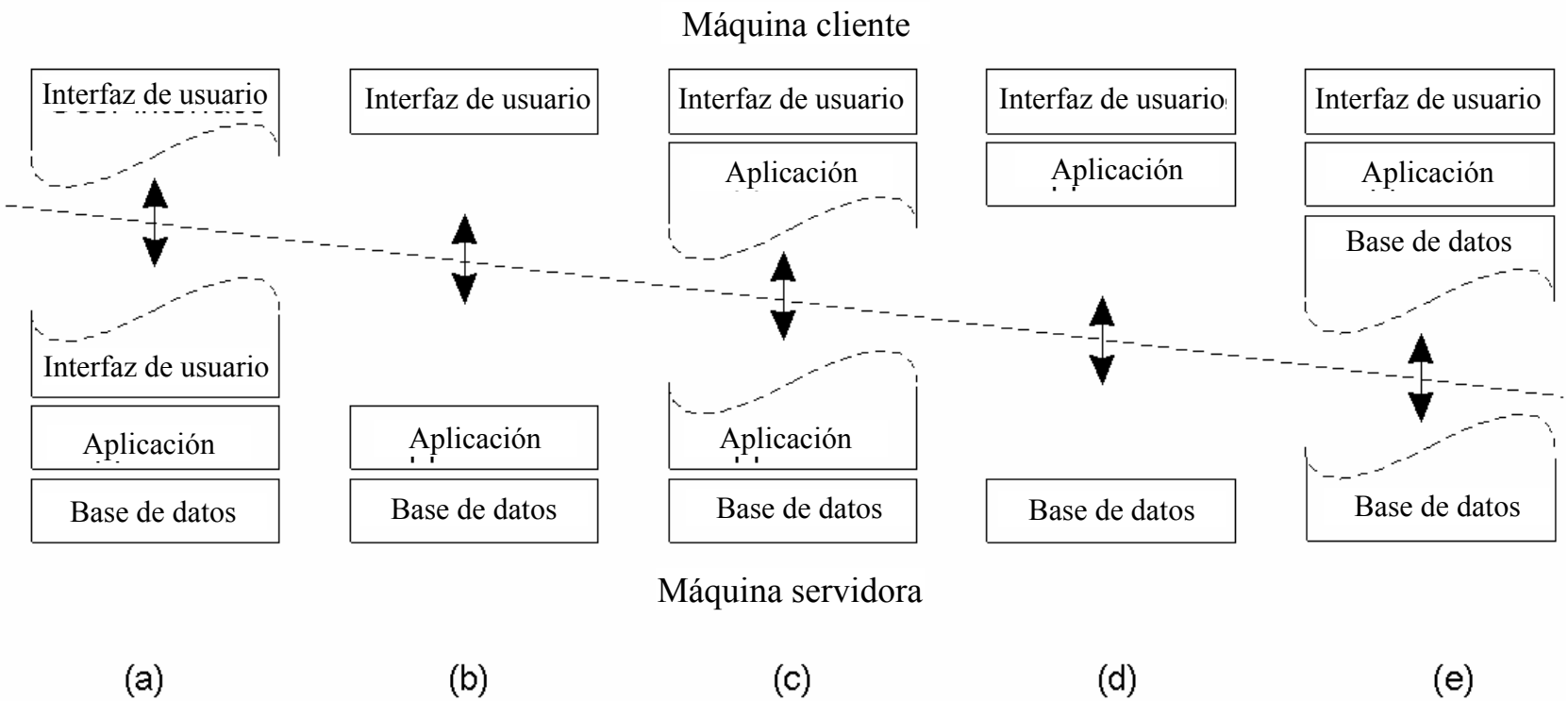


5.3.- Arquitecturas cliente/servidor

- Una sola capa: por ejemplo un terminal de líneas que interactúa con un ‘mainframe’.
- Dos capas: Cliente/Servidor.
- Tres o más capas: cada capa en una máquina diferente. Muchas veces se pueden redefinir como otras arquitecturas:
  - Sistemas de agentes cooperantes

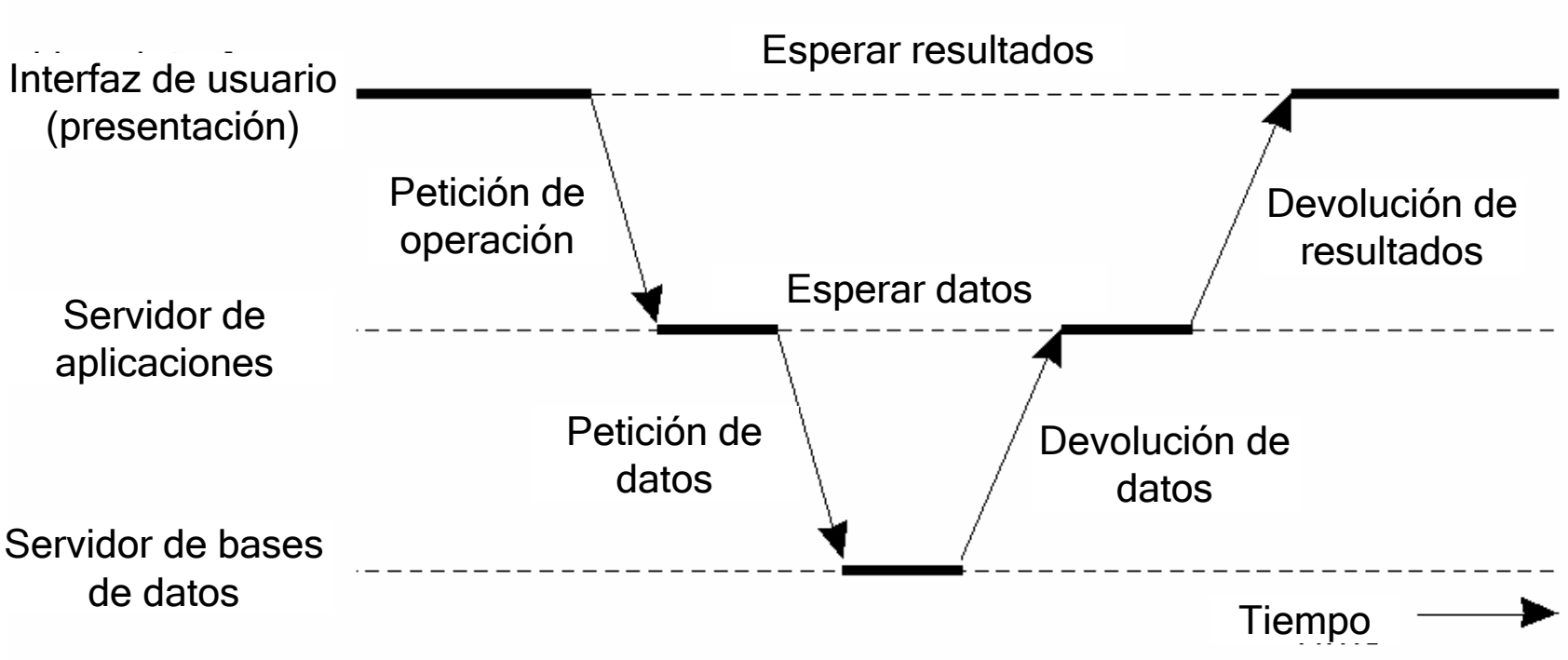
5.3.1.- Arquitectura en 2 capas

- Organizaciones alternativas de clientes y servidores (a) – (e).



5.3.2.- Arquitectura en 3 capas

- Obsérvese que el servidor actúa como cliente



5.3.3.- Arquitecturas alternativas

- La organización en capas se orienta a plantear una **distribución horizontal** de los servicios: cada unidad funcional es físicamente distribuida.
- Otra organización alternativa plantea la **distribución vertical**: cada cliente o cada servidor puede dividirse en partes lógicamente equivalentes, pero cada parte funcionando en un nodo distinto.
  - Si se dividen verticalmente los servidores: sistemas replicados
  - Si se dividen verticalmente los clientes: sistemas peer-to-peer

5.3.3.- Arquitecturas alternativas

- Ejemplo de distribución vertical de un servicio Web