

Introducción a la computación distribuida



Introducción a la computación distribuida por Francisco J. García Izquierdo. Universidad de la Rioja. Departamento de Matemáticas y Computación. se encuentra bajo una Licencia [Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 Unported](https://creativecommons.org/licenses/by-nc-sa/3.0/).



Departamento de
Matemáticas y
Computación

Grado en Ingeniería
Informática

Sistemas distribuidos

Objetivos

- Introducir el concepto de sistema distribuido y de computación distribuida
- Repasar las características de los sistemas distribuidos
- Repasar los retos tecnológicos que supone el empleo de sistemas distribuidos y su construcción, y analizar algunas de las soluciones
- Introducir la necesidad de repasar conceptos básicos de E/S, concurrencia, comunicaciones

Bibliografía

- ***Computación distribuida: conceptos y aplicaciones.*** M.L. Liu. Addison Wesley (2004). ISBN: 84-7829-066-4
 - Capítulo 1: Introducción a la computación distribuida
- ***Sistemas distribuidos: conceptos y diseño.*** G. Coulouris, J. Dollimore, T. Kindberg. Addison Wesley (2001). ISBN: 84-7829-049-4
 - Capítulo 1: Caracterización de los sistemas distribuidos
- ***Sistemas operativos distribuidos.*** A. Tanenbaum. Prentice Hall (1996). ISBN: 968-880-627-7

Agenda

- Definición
- Repaso de los tipos de computación
- Desafíos de los sistemas distribuidos
- Conceptos básicos en computación distribuida

Definición

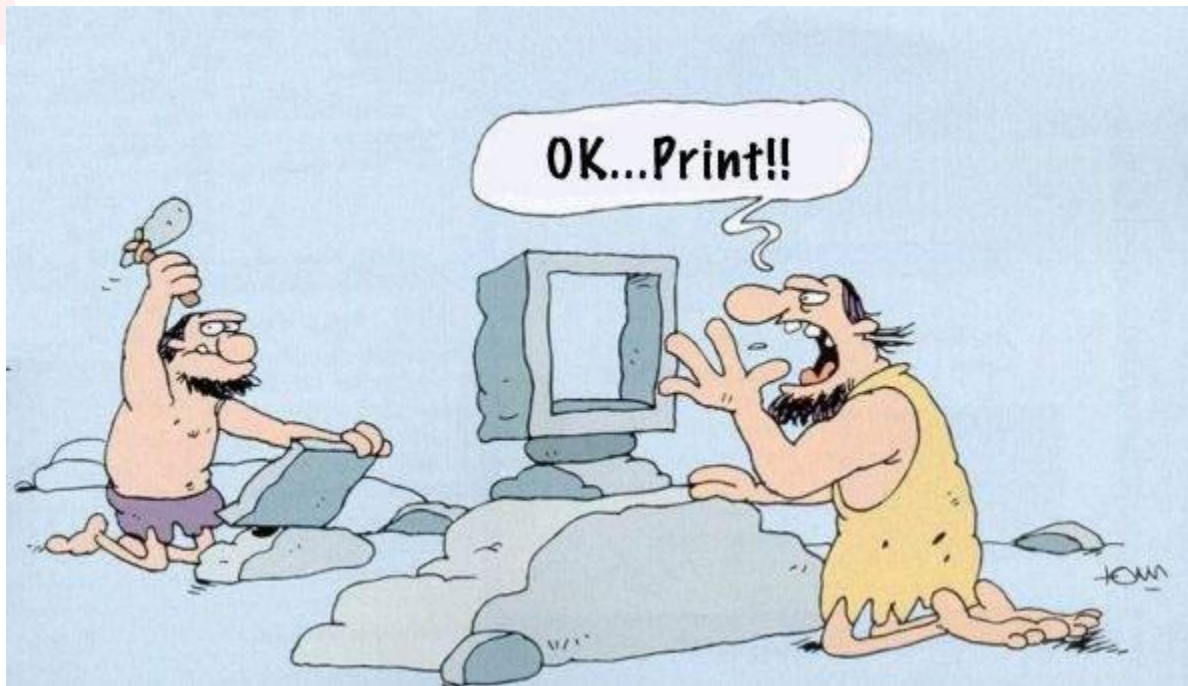
Un sistema distribuido es aquel en el que los componentes localizados en computadores, conectados en red, comunican y coordinan sus acciones mediante el paso de mensajes [Coulouris, 2001]

Conjunto de computadores independientes, interconectados a través de una red y que son capaces de colaborar para realizar una tarea [Liu, 2004]

Colección de computadores independientes que aparecen ante los usuarios como un único computador [Tanenbaum 1996]

- En cualquier caso
 - Varios computadores (nodos) conectados
 - Redes de cualquier tipo
 - Los nodos pueden estar en un mismo rack o en la otra punta del mundo
 - Apariencia de un único sistema

Un sistema distribuido ... de piedra



Consecuencias de la definición

- Lidiaremos con la concurrencia
 - Propia de una red en la que varios ordenadores funcionan a la vez
 - Permite sacar gran partido de los recursos del grupo
 - Desafío: coordinación, sincronización
- Aparecerán retos de coordinación:
 - Cuando los programas en los nodos necesitan colaborar coordinan sus acciones mediante intercambio de mensajes
 - Pueden darse situaciones de bloqueo, espera indefinida, diferencia de reloj, ...
 - Protocolos síncronos o asíncronos
- Cuidado con la seguridad
 - Para compartir y cooperar entre sí los nodos del S.D. deben estar abiertos al exterior. Esto implica riesgos para la disponibilidad
 - Es necesario gestionar la disponibilidad para evitar abusos, ataques, suplantaciones, ...

Consecuencias de la definición

- Fallos independientes

- Si todos los sistemas pueden fallar, los distribuidos pueden fallar más
 - Fallos de comunicación
 - Fallos en algún nodo
- Un problema común es que el resto de los nodos pueden no ser conscientes del fallo en la red o en otro nodo hasta un tiempo después de que éste ocurra
- Incluso pueden seguir trabajando de forma autónoma esperando el restablecimiento del servicio

“Un sistema distribuido es aquel en el que un fallo de un computador que ni siquiera sabes que existe puede dejar tu propio ordenador inutilizable” [Leslie Lamport]

Agenda

- Definición
- Repaso de los tipos de computación
- Desafíos de los sistemas distribuidos
- Conceptos básicos en computación distribuida

Tipos de computación

- Monolítica

- La que utiliza una única CPU para ejecutar uno o más programas por cada aplicación
- No conectado a red: sólo puede usar los recursos de los que dispone por sí
- Puede ser monousuario o multiusuario
- En multiusuario
 - Los recursos del computador se reparten en el tiempo (time-slicing)
 - Los usuarios (que puede estar dispersos geográficamente) se conectan a través de terminales
 - Ejemplos: IBM 360, Univac 1100 (Mainframes)
- Ejemplos de aplicaciones típicas: programas separados usados para una única función (contabilidad, nóminas,...)

Tipos de computación (II)

- Paralela
 - Utiliza más de un procesador a la vez para ejecutar un único programa
 - Puede darse en equipos con múltiples CPUs
- Distribuida
 - Usa múltiples computadores conectados a una red, cada uno de los cuales aporta su CPU y sus recursos
 - Los usuarios de cada equipo (multiusuario o monousuario) pueden acceder a los recursos de su propio computador y a los de los demás
- Cooperativa
 - Múltiples equipos, programas o recursos colaboran para realizar una tarea
 - Ejemplos:
 - P2P, GRID computing
 - Proyecto SETI

¿Por qué sistemas distribuidos?

- La motivación de la construcción de S.D. está relacionada con la [compartición de recursos](#).
- Un recurso es ... casi cualquier cosa (HW o SW): la CPU, un disco, contenido de cualquier tipo, un SGBD y sus esquemas, una impresora, un sensor, un algoritmo, ...
- Pero hay otras razones que apoyan su éxito
 - Económicas:
 - Precio de los PC y del acceso a red
 - Otros dispositivos: móviles, PDAs, sistemas embebidos, ...
 - Escalabilidad
 - En la computación monolítica los recursos estaban limitados por la capacidad del computador
 - Es posible añadir o captar más recursos si son necesarios
 - [Load Balancing](#)
 - Tolerancia a fallos
 - Es posible replicar recursos para presentar disponibilidad en presencia de fallos ([Mirroring](#))
 - ¡Y no digamos nada de Internet!

Ejemplos de sistemas distribuidos

- Internet

- Red enorme
- Programas ejecutándose en diferentes ordenadores
- Se comunican mediante paso de mensajes (ej HTTP)
- Empleando un medio común de comunicaciones
- El conjunto de servicios es abierto (se pueden añadir servidores)

- Intranet

- Porción de Internet con acceso restringido por una política de seguridad
- Múltiples configuraciones: desde una única LAN a varias LAN en distintos países (usando VPN o no)
- Se conecta a Internet mediante un router
- Se protege del exterior y limita los accesos desde el interior mediante un cortafuegos (firewall)
- Optimiza el acceso al exterior mediante caches (locales) y servidores proxy (compartidos)
- Hay casos en los que ni siquiera están conectadas a Internet (hospitales, redes militares, bancos, ...)

Agenda

- Definición
- Repaso de los tipos de computación
- Desafíos de los sistemas distribuidos
- Conceptos básicos en computación distribuida

Heterogeneidad

- Heterogeneidad: variedad y diferencia
- Se da en: redes, hardware de ordenadores, sistemas operativos, lenguajes de programación, implementaciones de diferentes desarrolladores
- La forma de tratar esta diversidad es mediante el uso de estándares
 - Internet usa un gran número de esos protocolos que permiten que nodos de distinto tipo (máquina, SO,...) se entiendan
 - HTTP, FTP, SMTP, POP, ...
- Otro tema difícil de tratar es la distinta representación de los datos en distintos entornos (máquinas, países):
 - Diferente codificación de los enteros
 - Diferentes juegos de caracteres
- Relacionado con los sistemas distribuidos orientados a objetos hay tecnologías que gestionan la heterogeneidad ofreciendo un interface común de acceso en forma de algún middleware

Middleware

- Capa software que proporciona una abstracción de programación y de la heterogeneidad subyacente (redes, S.O., lenguajes de programación)
- Proporciona un modelo computacional uniforme a disposición de los programadores de aplicaciones distribuidas:
 - Abstrae los protocolos y mecanismos de bajo nivel y
 - proporciona una serie de posibilidades de alto nivel a los desarrolladores
 - Ejemplo:
 - un middleware de comunicaciones que proporcione primitivas de comunicación y conexión independientemente del protocolo usado (TCP, UDP, Bluetooth, TIBCO rendezvous, ...)
- Ejemplos de middleware:
 - Invocación sobre objetos remotos: CORBA, Java RMI
 - Notificación de eventos remotos
 - Acceso remoto a BD (mediante SQL)
 - Acceso a sistemas mediante mensajes (síncronos o asíncronos), MOM
 - Servidores de aplicaciones, monitores transaccionales...

Extensibilidad

- Característica que determina si un sistema puede ser ampliado y/o reimplementado en distintos aspectos
 - Añadir nuevos recursos y servicios (ej: un nuevo disco)
 - Ampliar la capacidad de servicio de los ya existentes
 - Modificar los ya existentes por otros más capaces (ej: cambio de BD)
- Gracias al empleo de estándares estas tareas deberían poder realizarse por cualquier proveedor

Seguridad

- Los recursos de información que un sistema distribuido maneja pueden ser de alto valor
- La disponibilidad lleva asociados riesgos de seguridad
- La seguridad tiene varios componentes (aspectos):
 - **Confidencialidad**: protección contra el descubrimiento por parte de individuos no autorizados
 - Acceso no autorizado a sistemas
 - Captura de mensajes intercambiados con datos sensibles
 - **Integridad**: protección contra la alteración o corrupción (dentro del sistema o en el mensaje)
 - **Disponibilidad**: protección contra la interferencia con los procedimientos de acceso a los recursos (denegación de servicio, DNS cache poisoning)
 - **Autenticación**: asegurar que los interlocutores son realmente quienes pretenden ser
 - Otros: control de acceso, seguridad en el código móvil, no repudio, ...

Escalabilidad

- Un sistema es escalable si conserva su efectividad cuando ocurre un incremento significativo en el número de recursos y número de usuarios
- Para que un sistema sea escalable debe ser extensible
- El diseño de sistemas escalables presenta como retos:
 - Control de los recursos físicos
 - Al crecer la demanda debería ser posible, a coste razonable, extender los recursos físicos que prestan el servicio (ej: [Balanceo de carga / mirroring](#))
 - Control de las pérdidas de prestaciones: minimizar
 - Prevención del desbordamiento de recursos SW
 - Ej.: el desbordamiento del actual direccionamiento IP; efecto 2000
 - Difícil de prever con antelación; puede ser peor prevenir demasiado
 - Evitación de los cuellos de botella de prestaciones
 - Recursos que se acceden muy a menudo (Ej. Registro DNS)
 - Soluciones: réplicas, caches, descentralización (DNS)

Tipos de escalabilidad

● Vertical

- “Comprar” hardware más potente, reemplazando el actual.
- Está limitado por la tecnología disponible
- Es sencillo de poner en práctica
- El coste no suele escalar linealmente: un servidor el doble de rápido suele ser más del doble de caro

● Horizontal

- Comprar hardware adicional, que complementa al actual.
- Es más complejo de diseñar, construir y mantener
- Requiere planificación: el SW y HW debe permitir hacerlo

Tratamiento de fallos

Posible ampliación con
Técnicas de redundancia parcial, total, por votación
Bloques de recuperación
Recuperación hacia delante/atrás
Excepciones

- Disponibilidad: porción del tiempo en el que el sistema está disponible
- Los fallos en un S.D. suelen ser parciales y requieren un tratamiento de fallos
- En primer lugar es necesario detectar fallos
 - Algunos fallos pueden detectarse (ej con ayuda de checksums)
- Podemos considerar las siguientes técnicas
 - Enmascaramiento de fallos:
 - Tras detectarse, algunos fallos pueden ocultarse o atenuarse
 - Ojo que las contramedidas también puede fallar
 - Tolerancia a fallos:
 - Hacer un sistema 100% a prueba de fallos puede ser muy costoso (o imposible)
 - Hay ciertos fallos que pueden tolerarse
 - Recuperación frente a fallos: detectado el fallo se trata de dejar el sistema en un estado previo correcto (caso de las BD)
 - Redundancia:
 - Emplear varios recursos o servicios idénticos ([mirroring](#))
 - Reto: que el cambio entre réplicas sea rápido

Concurrencia

- Una característica básica de los sistemas distribuidos: compartición de recursos
- Hay sistemas que reciben miles de solicitudes por minuto (servidores Web)
- Los sistemas no pueden procesar estas solicitudes de forma secuencial (sería muy limitante)
- Existe la posibilidad de que un mismo recurso sea accedido por varios usuarios al mismo tiempo (ej, BD, sistema de subastas, aplicación de reserva de turno de prácticas, ...)
- Es necesario disponer de mecanismos que garanticen la integridad de los recursos ante accesos concurrentes (ej.: los problemas clásicos de acceso a BD)
 - Una solución podría pasar por disponer de mecanismos de bloqueo

Transparencia

- Es la ocultación al usuario y al programador de la separación de los componentes del sistema distribuido, de forma que lo perciba como un todo más que como una colección de componentes individuales (ej RMI).
- Tipos de transparencia
 - **Transparencia de acceso** que permite acceder a los recursos locales y remotos empleando operaciones idénticas.
 - **Transparencia de ubicación** que permite acceder a los recursos sin conocer su localización.
 - **Transparencia de concurrencia** que permite que varios procesos operen concurrentemente sobre recursos compartidos sin interferencia mutua.
 - **Transparencia de replicación** que permite utilizar múltiples ejemplares de cada recurso para aumentar la fiabilidad y las prestaciones sin que los usuarios y los programadores de aplicaciones necesiten su conocimiento.

Transparencia (II)

- **Transparencia frente a fallos** que permite ocultar los fallos, dejando que los usuarios y programas de aplicación completen sus tareas a pesar de fallos del hardware o de los componentes software.
- **Transparencia de movilidad** que permite la reubicación de recursos y clientes en un sistema sin afectar la operación de los usuarios y los programas.
- **Transparencia de prestaciones** (HW) que permite reconfigurar el sistema para mejorar las prestaciones según varía su carga.
- **Transparencia al escalado** (SW) que permite al sistema y a las aplicaciones expandirse en tamaño sin cambiar la estructura del sistema o los algoritmos de aplicación.
- Especialmente afectan a los sistemas distribuidos la transparencia de acceso y de ubicación (se le suele llamar transparencia de red

Transparencia (III)

- Ejemplos de transparencia
 - De acceso: sistema de ficheros, RMI
 - De ubicación: las URL
 - son nombres lógicos, no direcciones físicas (gracias al DNS)
 - No tiene transparencia de movilidad
 - Frente a fallos: el email, en presencia de fallos se reenvía hasta agotar un límite o lograr el envío; el protocolo TCP
 - De movilidad: cambio de célula de los teléfonos móviles

Agenda

- Definición
- Repaso de los tipos de computación
- Desafíos de los sistemas distribuidos
- Conceptos básicos en computación distribuida

¿Qué se necesita saber?

- Algunos de los conceptos clave que hemos de repasar para poder hacer programación distribuida son:
 - Programación en red
 - Lleva pareja un repaso profundo de los APIs de [entrada/salida](#) (I/O)
 - Programación concurrente
 - Serialización (de cara al estudio de RMI)
- Todo ello lo haremos usando Java como lenguaje de programación
- Los próximos temas los dedicaremos a cada uno de esos aspectos

Fin del tema 1

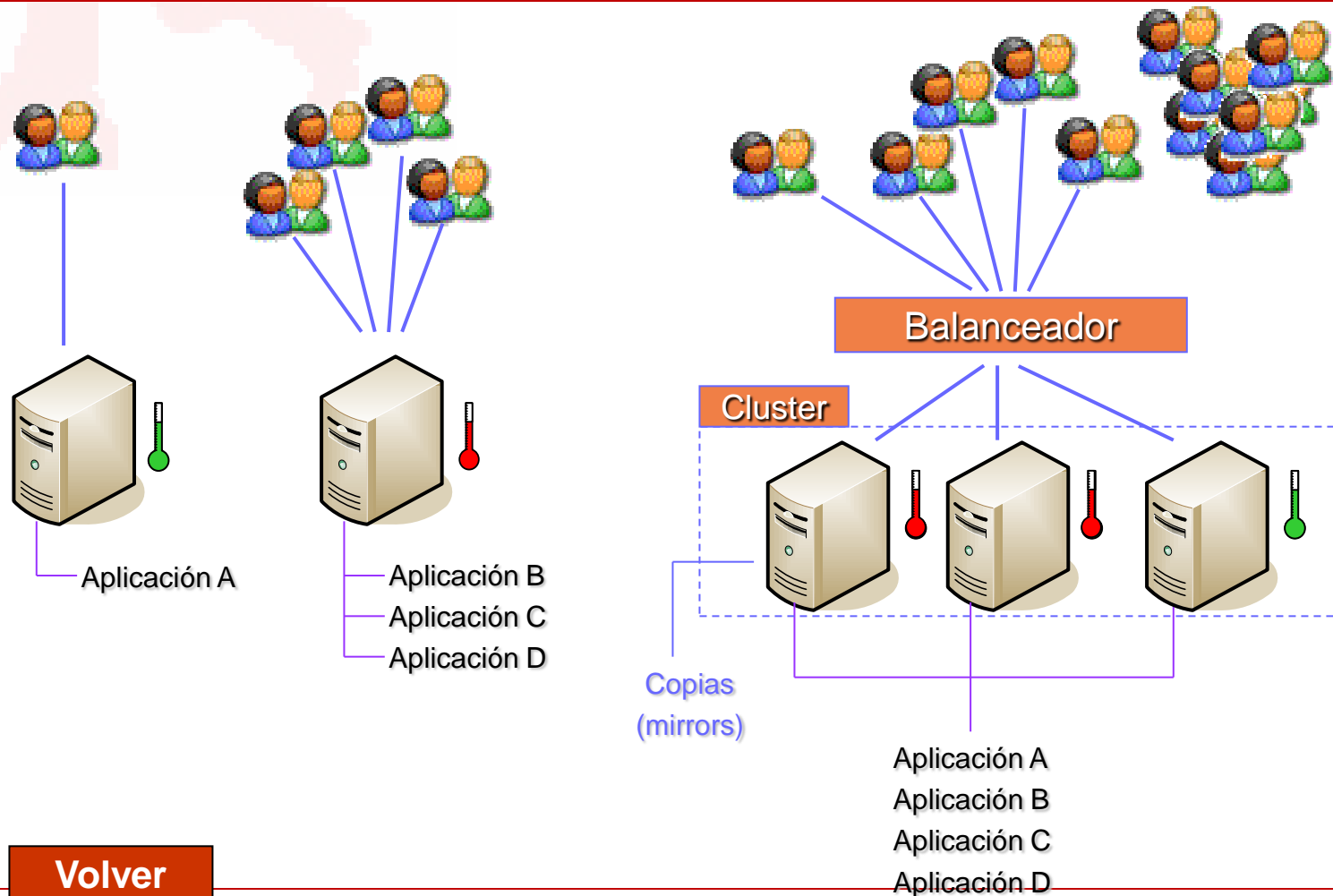


Departamento de
Matemáticas y
Computación

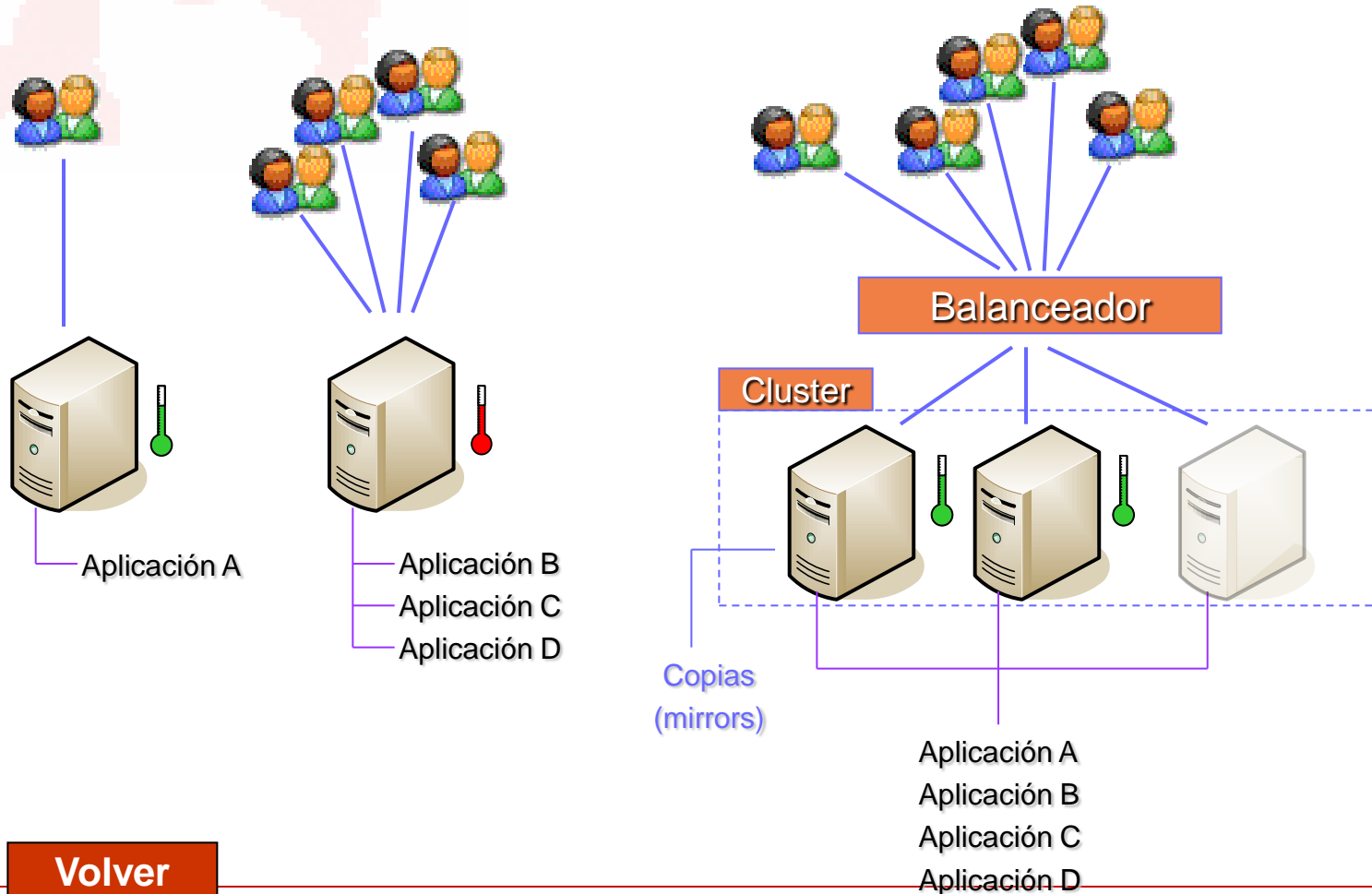
Grado en Ingeniería
Informática

Sistemas distribuidos

Balanceo de carga - mirroring

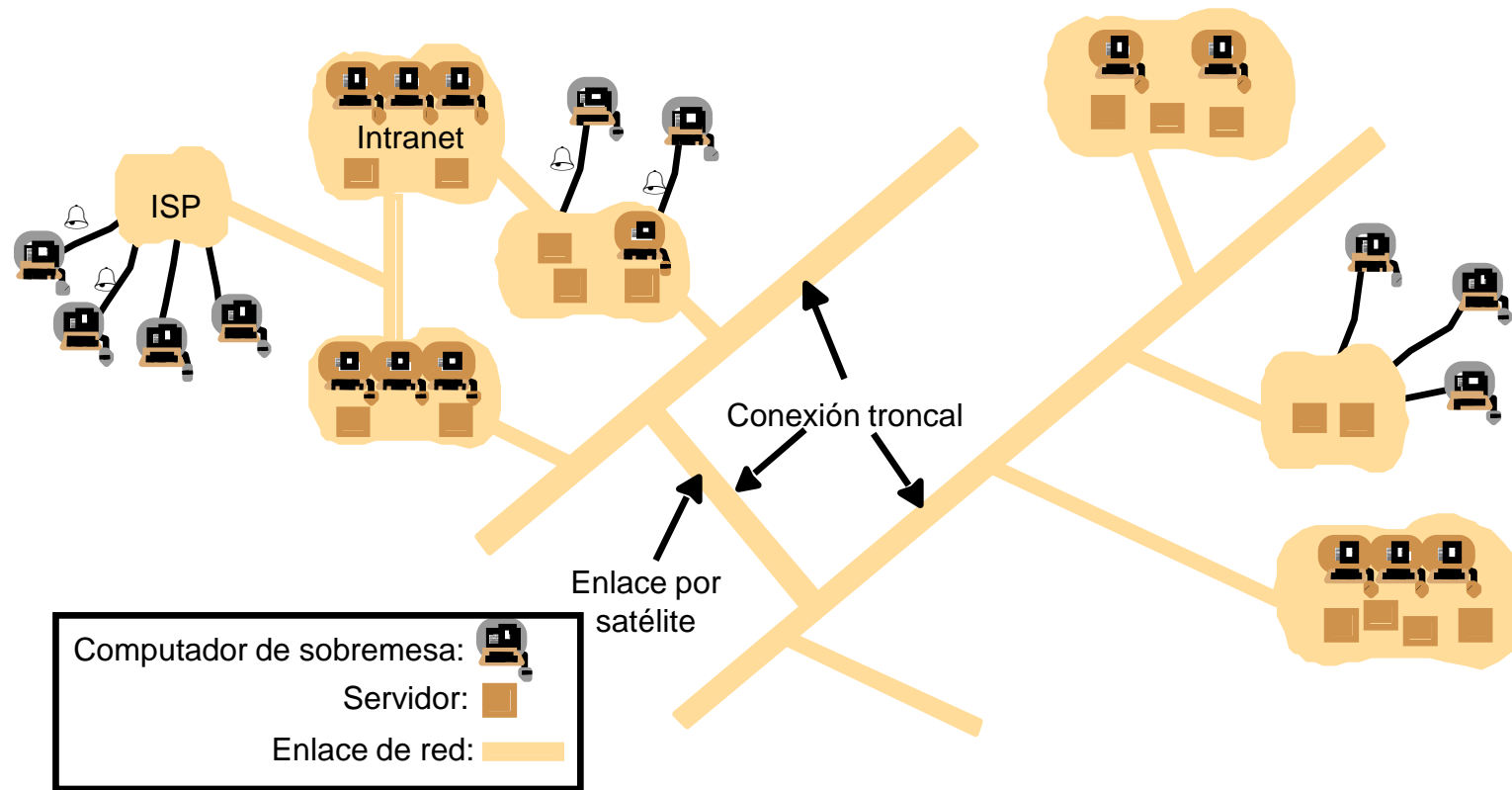


Tratamiento de fallos - mirroring



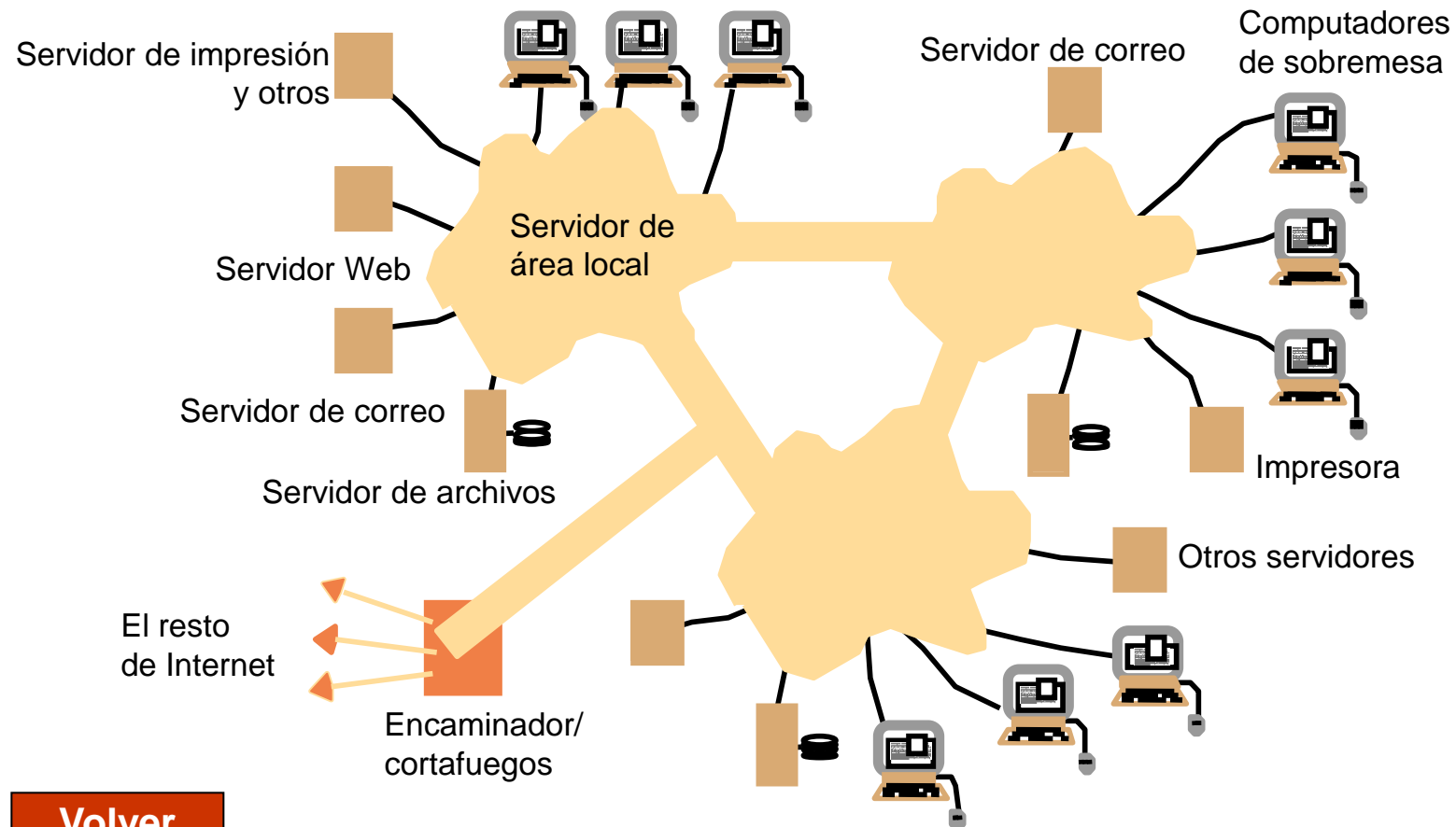


Internet



[Volver](#)

Intranet



Volver

Middleware

