Tema 1
Introducción a los Sistemas Distribuidos

Sistemas Distribuidos

Grado en Ingeniería Informática

Universidad Carlos III de Madrid

Contenido

- Evolución hacia los sistemas distribuidos
- Memoria compartida vs memoria distribuida
- Concepto de sistema distribuido
- Repaso básico de redes de computadores
- Principales aspectos de diseño de un sistema distribuido
- Paradigmas de comunicación en sistemas distribuidos
- Arquitecturas de comunicación

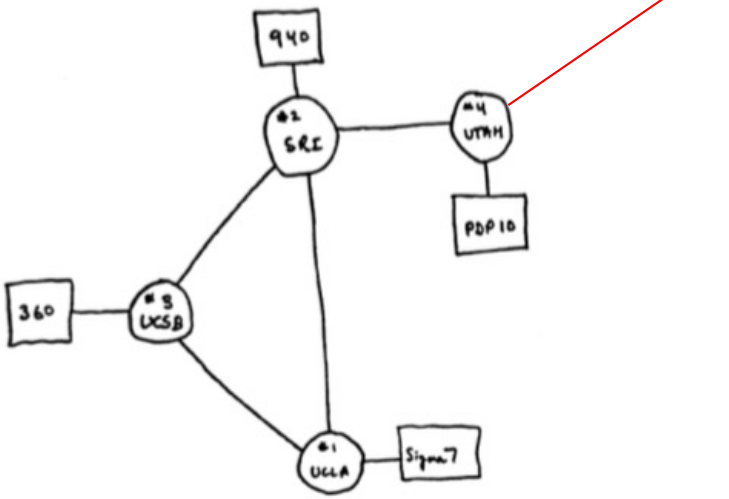
Evolución de la Informática

- En los años 60-70:
  - Mainframes centrales
    - Sistemas de tiempo compartido
    - Recursos centralizados
    - Terminales simples
  - Interfaces de usuario poco amigables
  - Aparecen las primeras redes

Fuente: Mainframe Computer by naotakem, CC BY-NC-SA 2.0

Aparición de TCP/IP

- Origen en 1969 con la red ARPANET



Fuente: ARPANET first router
by Steve Jurvetson from Menlo Park, USA , CC BY-NC-SA 2.0

- 1974: Se diseña TCP/IP (Vinton G. Cerf, Robert E. Kahn)

Evolución de la Informática

- En los años **80**:
  - PCs y estaciones de trabajo
    - Estandarización
    - Crecimiento en el nº de computadores
  - Predominio de aplicaciones ejecutadas localmente
  - Interfaces amigables
  - Redes de área local (LAN)
    - Acceso a recursos compartidos (impresoras, sistemas de ficheros)
  - Aparecen los primeros sistemas operativos distribuidos
    - Mach, Sprite, Chorus, ...

Fuente: IBM PC Computer
by Accretion Disc, CC BY-NC-SA 2.0

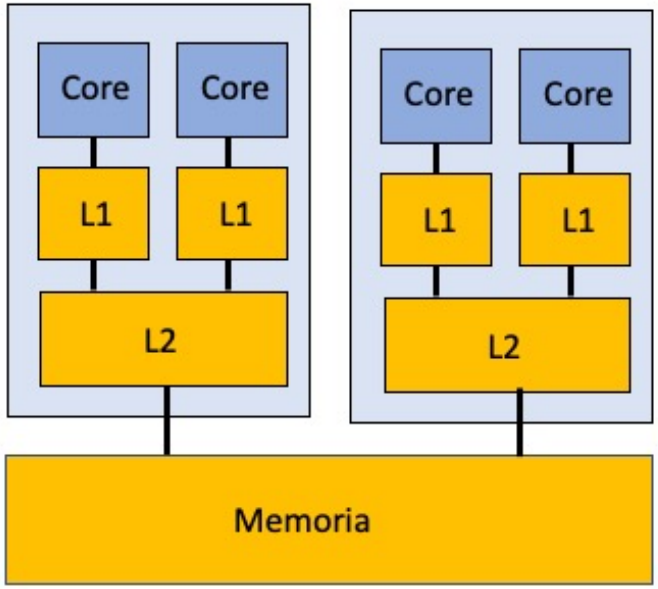
Evolución de la informática

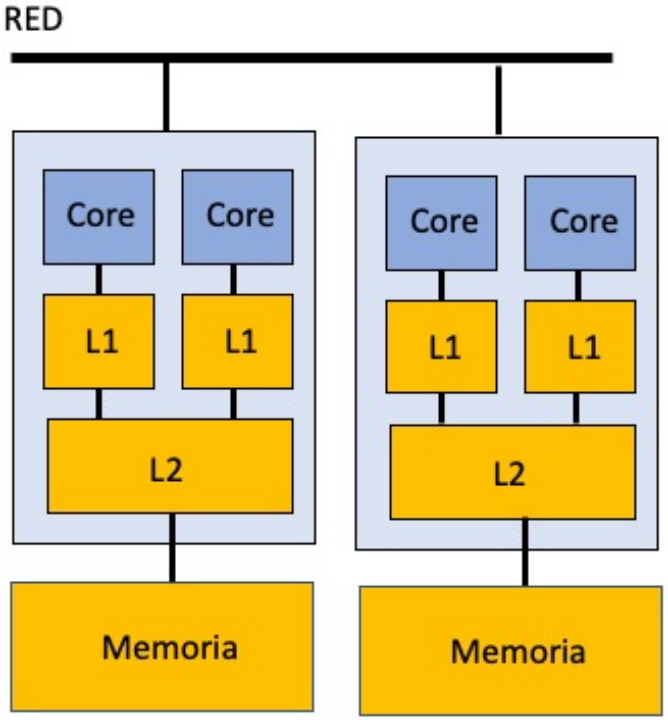
- En los años **90**:
  - Despegue de las aplicaciones cliente/servidor
  - Más descentralización
    - Aplicaciones ejecutadas localmente y en red
  - Aparece la web en 1991 (Tim Berners-Lee en el CERN)
  - Enorme difusión de Internet gracias a la web
  - Nuevas necesidades y aplicaciones basadas en web
    - Comercio electrónico, Multimedia
    - Sistemas de control
    - Aplicaciones médicas
    - Supercomputación en Internet

Evolución de la informática

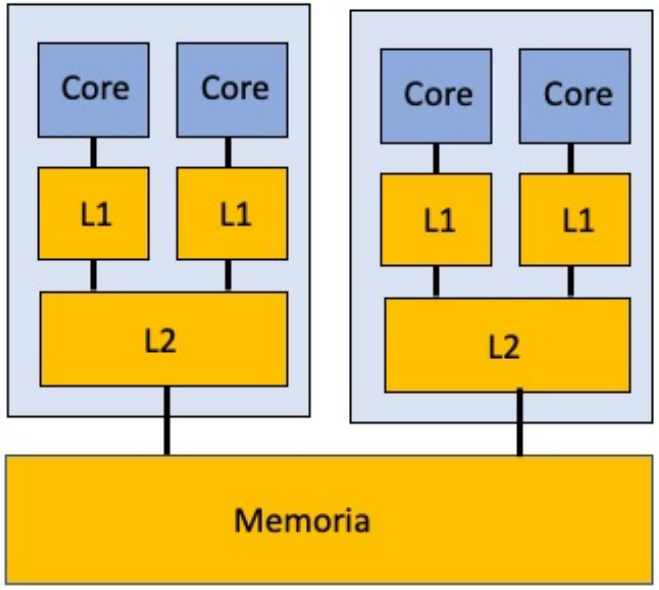
- En los años **2000:**
  - Nuevos paradigmas de computación distribuida
    - Cluster computing, Grid computing, cloud computing
    - Peer-to-Peer
    - Computación ubicua, móvil
    - Computación voluntaria
  - Dispositivos móviles
  - Aplicaciones para Internet basadas en Web
  - Tendencias:
    - Todas las aplicaciones y datos en red
      - Cloud Computing
      - Computación móvil
    - Todos los dispositivos conectados
      - Internet of Things

Memoria compartida vs memoria distribuida

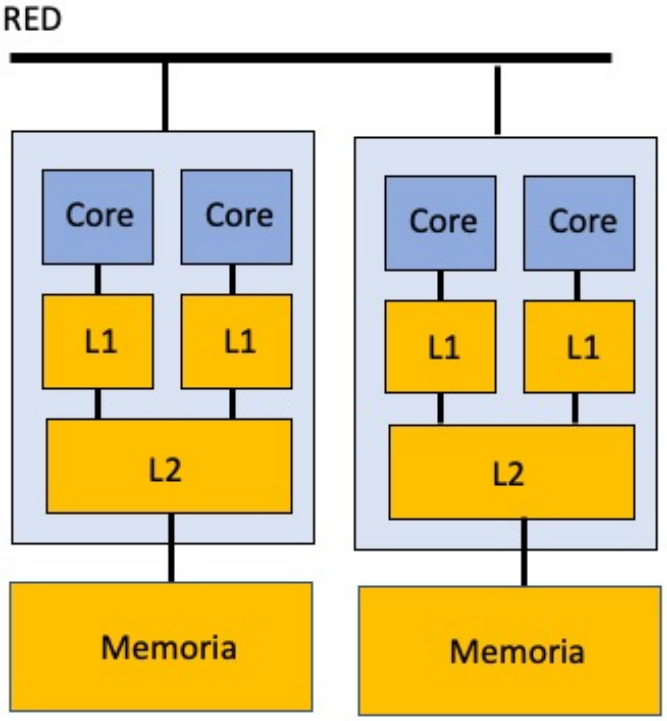




Memoria compartida vs memoria distribuida

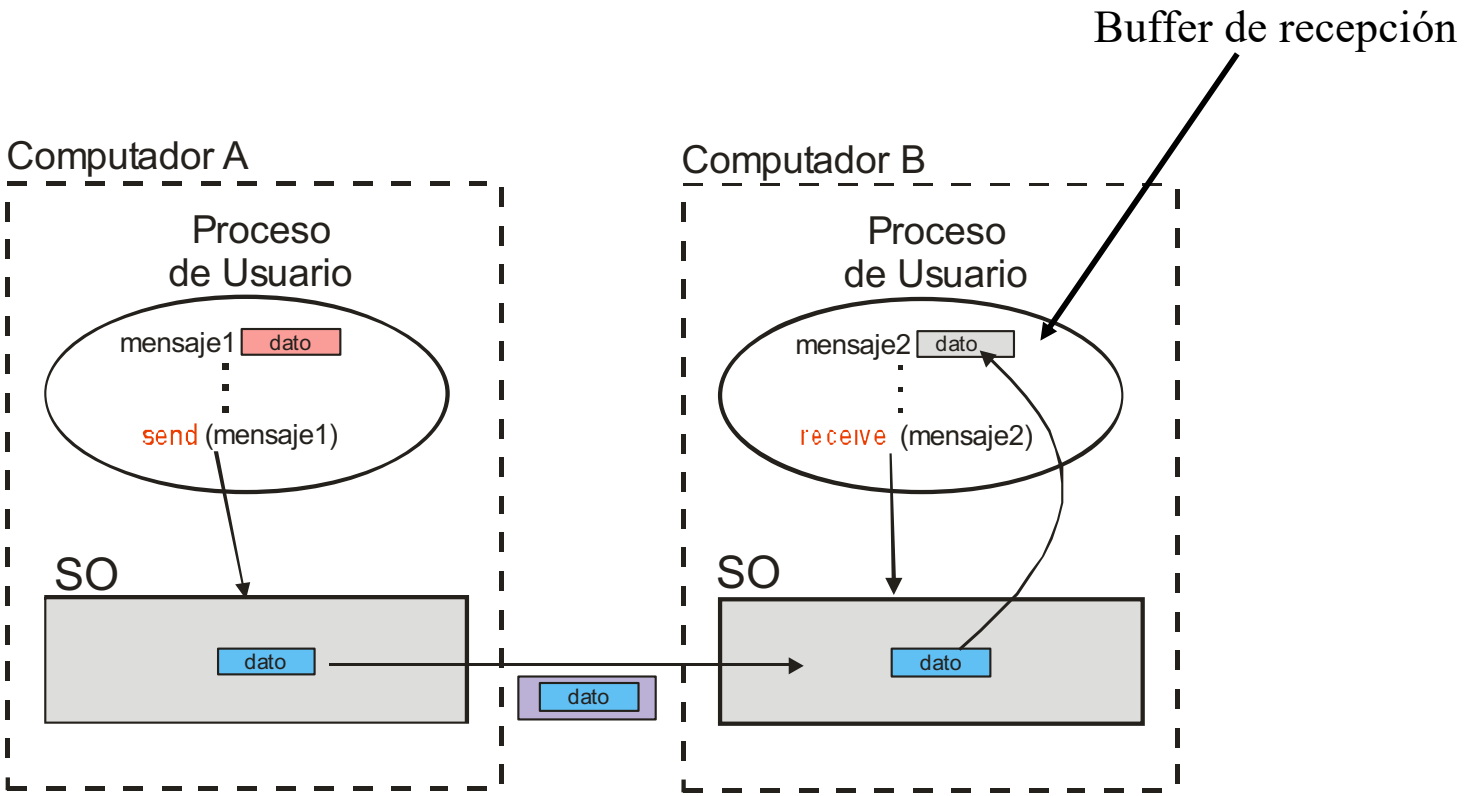


- Comunicación mediante memoria
- Sincronización mediante Locks



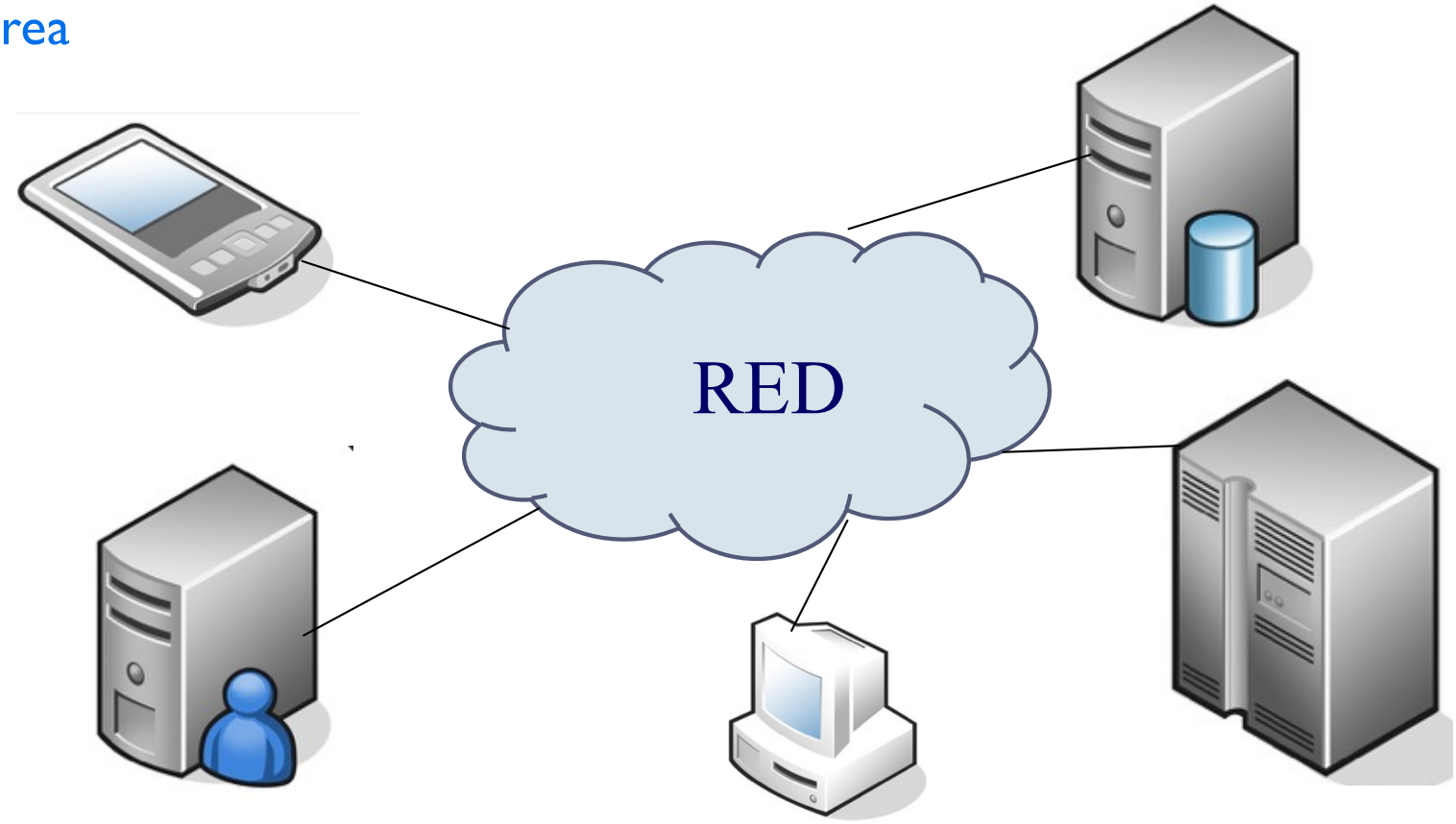
- Comunicación mediante paso de mensajes
- Sincronización mediante paso de mensajes

Paso de mensajes



Sistema Distribuido (definición)

- Sistema formado por recursos hardware y software físicamente distribuidos e interconectados a través de una red, que comunican mediante **paso de mensajes** y cooperan para realizar una determinada tarea

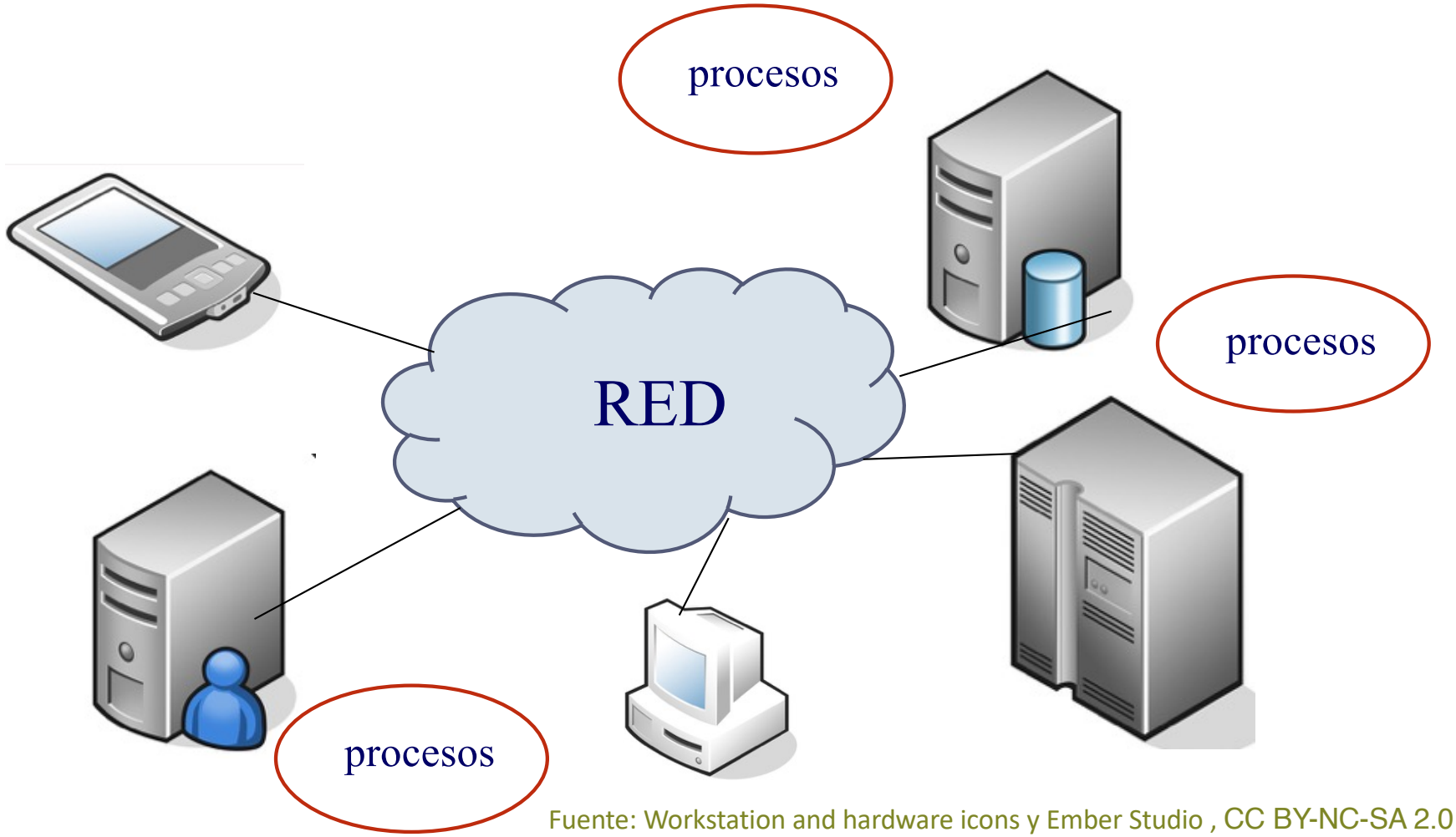


Conceptos previos

- Un **programa** es un conjunto de instrucciones
- Un **proceso** es un programa en ejecución
- Una **red de computadores** es un conjunto de computadores conectados por una red de interconexión
- **Sistema distribuido** Un conjunto de computadores (sin memoria ni reloj común) conectados por una red
  - **Aplicaciones distribuidas:** Conjunto de procesos que ejecutan en uno o más computadores que colaboran y comunican intercambiando *mensajes*.
- Un **protocolo** es un conjunto de reglas e instrucciones que gobiernan la comunicación en un sistema distribuido, es decir, el intercambio de mensajes

Computación distribuida

- Computación que se realiza en un sistema distribuido



Fuente: Workstation and hardware icons y Ember Studio , CC BY-NC-SA 2.0

Características principales

- Múltiples componentes autónomos
- Concurrencia. Los programas ejecutan de forma concurrente
- Única forma de comunicación y sincronización: paso de mensajes
- Ausencia de un reloj global, que permita coordinar las acciones de los programas que ejecutan en el sistema
- Fallos independientes
- SW más complejo

Ejemplo: Web

- Programas: navegador (cliente), servidor web (servidor)
- Concurrencia: servidor debe ser concurrente
- Protocolo de aplicación: HTTP
- Sistema de nombrado: URL (*uniform resource locator*) (www.uc3m.es)

Ejemplos de aplicaciones y sistemas distribuidos

- Correo electrónico (IMAP, POP)
- Transferencia de ficheros (FTP)
- *World Wide Web* (WWW)
- Sistemas de control de tráfico aéreo
- Aplicaciones bancarias
- Comercio electrónico
- Aplicaciones multimedia (videoconferencias, vídeo bajo demanda , *etc.*)
  - El ancho de banda en estas aplicaciones es un orden de magnitud mayor que en otras
  - Requieren calidad de servicio (QoS)
- Aplicaciones médicas (transferencia de imágenes)
- Aplicaciones móviles (comunicación inalámbrica)
- Aplicaciones con redes de sensores
- Internet of Things

Sistemas distribuidos y paralelos

- **Sistemas distribuidos**
  - **Objetivo**: compartir recursos y colaborar
- **Sistemas paralelos**
  - **Objetivo**:
    - Alto rendimiento (High Performance Computing)
    - Alta productividad (High Throughput Computing)
  - Máquinas paralelas (arquitecturas dedicadas)
    - Multiprocesadores
    - Multicomputadores
  - Redes de estaciones de trabajo trabajando como un multicomputador (*cluster*)
  - *Grid Computing*
  - *Clusters virtuales en cloud computing*
  - *Supercomputadores*

Ventajas que pueden ofrecer los SSDD

- Compartir recursos (HW, SW, datos)
  - Acceso a recursos remotos.
    - Modelo cliente-servidor
    - Modelo basado en objetos
- Ofrecen una buena relación coste/rendimiento
- Capacidad de crecimiento (escalabilidad)
- Tolerancia a fallos, disponibilidad
  - Replicación
- Concurrencia: servicio a múltiples usuarios simultáneamente
- Velocidad: capacidad global de procesamiento disponible para:
  - Ejecución paralela de una aplicación

Desventajas de los sistemas distribuidos

- Interconexión
  - La red es un elemento crítico
  - Fiabilidad
  - Saturación
- Seguridad y confidencialidad
- Software más complejo

Repaso de conceptos básicos de redes

- **Subsistema de comunicación:** conjunto de componentes HW y SW que proporcionan servicios de comunicación en un sistema distribuido
- **Red de computadores:** conjunto de computadores conectados por medio de una red
- Un **host** es un computador u otros dispositivo que usan la red para propósitos de comunicación.
- **Protocolo:** conjunto de reglas e instrucciones que gobiernan el intercambio de paquetes y mensajes
- **Mensaje:** objeto lógico que se intercambian entre dos o más procesos
  - Su tamaño puede ser bastante grande
  - Un mensaje se descompone en paquetes
- **Paquete:** tipo de mensaje que se intercambia entre dos dispositivos de comunicación
  - Tamaño limitado por el hardware

Propiedades de un subsistema de comunicación

- Tasa de transferencia:
  - Velocidad de transferencia de los datos
  - Medida en bits por segundo
- Latencia:
  - Retardo, tiempo que hay entre el envío de un mensaje y éste comienza a llegar al destino
  - Puede medirse como el tiempo requerido para enviar un mensaje vacío
- Tiempo de transferencia:
  - latencia + (tamaño datos /tasa de trasferencia)
- Paquetes/segundo (rendimiento o *throughput*):
  - Número de paquetes enviados/recibidos en un segundo
- Capacidad de crecimiento (escalabilidad):
  - Medida de: efectividad vs. nº nodos/nº recursos
- Calidad de servicio (QoS):
  - Importante en aplicaciones multimedia y de tiempo real
- Fiabilidad del subsistema
  - Mecanismos de detección de errores
- Seguridad (protección de los paquetes):
  - Confidencialidad: proteger la identidad de los emisores

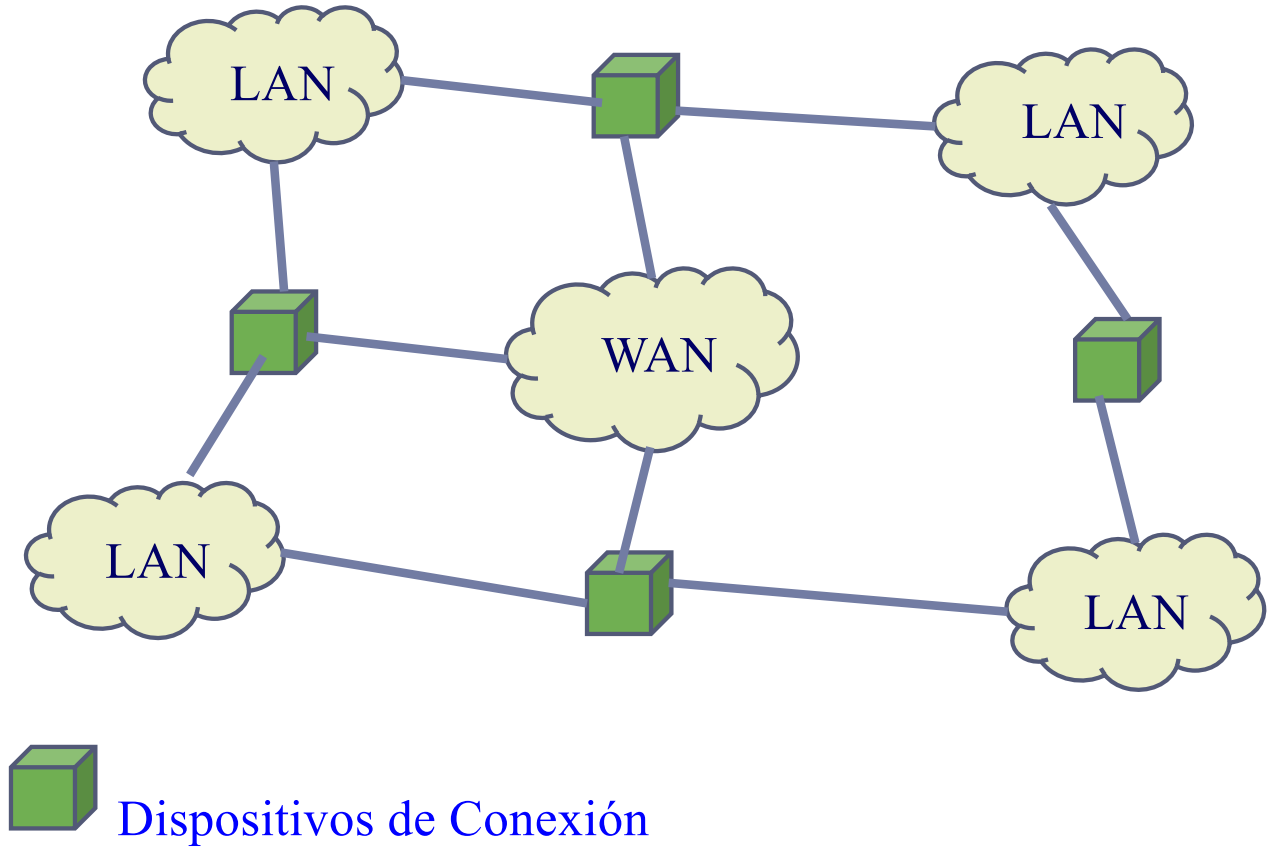
Tipos de redes

- Redes de área local (LAN, *Local Area Network*)
  - Redes que enlazan sistemas cercanos
  - Posibilidad de difusión de mensajes (*broadcast*)
  - Personal Area Networks (PANs) son un tipo de LAN (bluetooth)

- Redes de área extensa (WAN, *Wide Area Network*)
  - Menor ancho de banda
  - Mayor latencia
  - Redes telefónicas, redes públicas de datos, fibra óptica RDSI, B-RDSI, ATM
  - Redes de área local inalámbricas (WLAN, *Wireless LAN*)
    - Eliminan la necesidad de infraestructura cableada para conectar dispositivos
    - Ejemplo: IEEE 802.11 (WiFi)
- Diferencias entre LAN y WAN cada vez menores

Interconexión de redes

- Una interred o internet es un conjunto de redes interconectadas



Protocolos y arquitecturas

- Un **protocolo** es un conjunto de reglas y formatos que permiten la comunicación entre procesos
- La definición de un protocolo tiene dos partes:
  - Especificación de la secuencia de mensajes que deben intercambiarse
  - Especificación del formato de mensajes

- Pila de protocolos: organización en capas del HW y SW que implementa un protocolo
  - Ejemplo: TCP/IP

Funciones de una pila de protocolos (I)

- Segmentación y ensamblado de los mensajes
  - Segmentación: fragmentación de los mensajes en paquetes más pequeños de acuerdo al MTU
  - Ensamblado: composición del mensaje original a partir de los fragmentos del mensaje en la máquina destino
- Encapsulado: incorporación de información de control a los datos
  - Dirección del emisor y receptor
  - Código de detección de errores

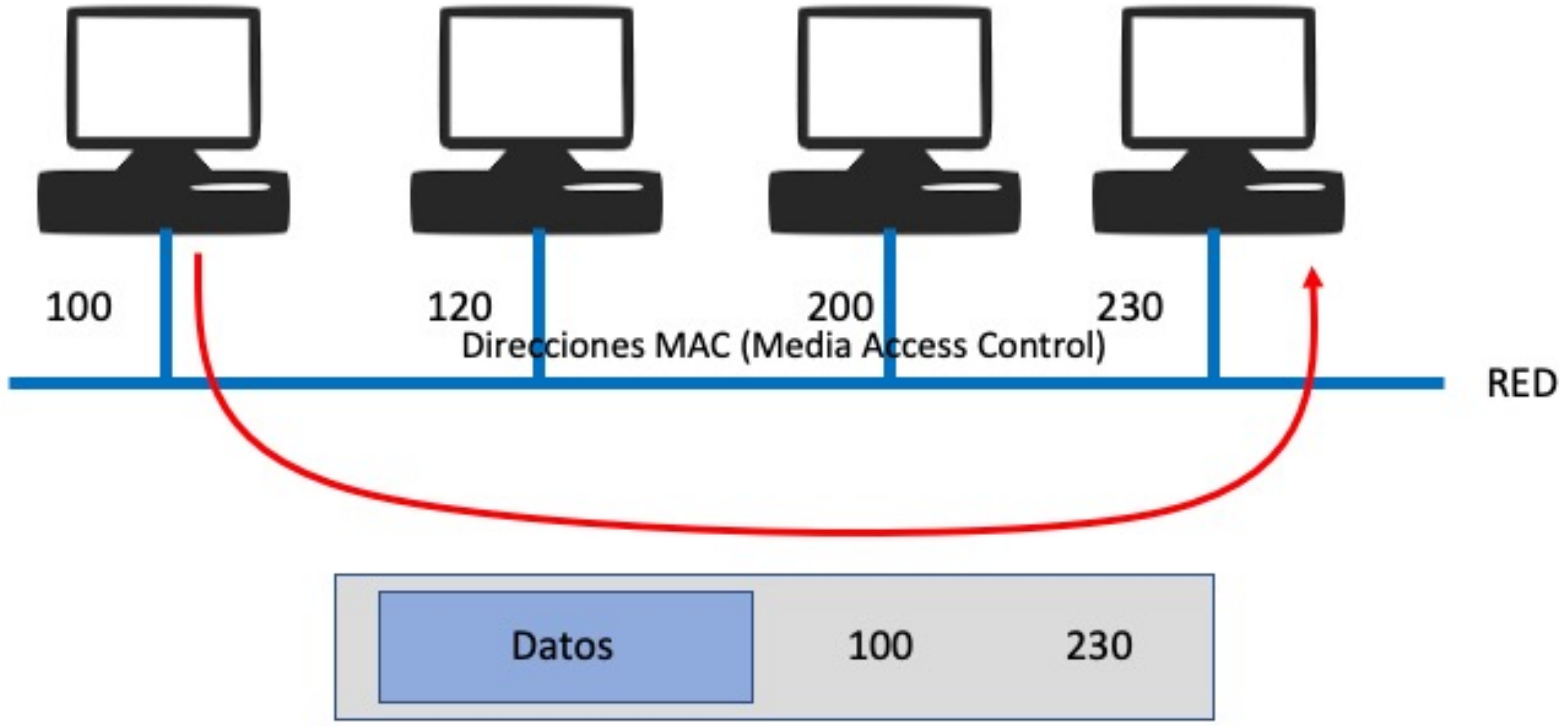
Funciones de una pila de protocolos (II)

- Control de conexión
  - Protocolos orientados a conexión
  - Protocolos no orientados a conexión
    - No se asegura el orden secuencial de los datos transmitidos
- Entrega ordenada en protocolos orientados a conexión
  - Números de secuencia
- Control de congestión o flujo: función realizada en el receptor para limitar la cantidad o tasa de datos que envía el emisor
  - Tamaño de ventana indica la cantidad de datos que el emisor puede enviar antes del siguiente ACK

Funciones de una pila de protocolos (III)

- Control de errores: se basan en el uso de una secuencia de comprobación y reenvío
- Direccionamiento: entrega de los mensajes al receptor
  - Cada host se identifica mediante una dirección de red y un número de puerto
- Multiplexación: necesario para un uso más eficiente de los servicios
- Servicios de transmisión:
  - Prioridad
  - Calidad de servicio
  - Seguridad

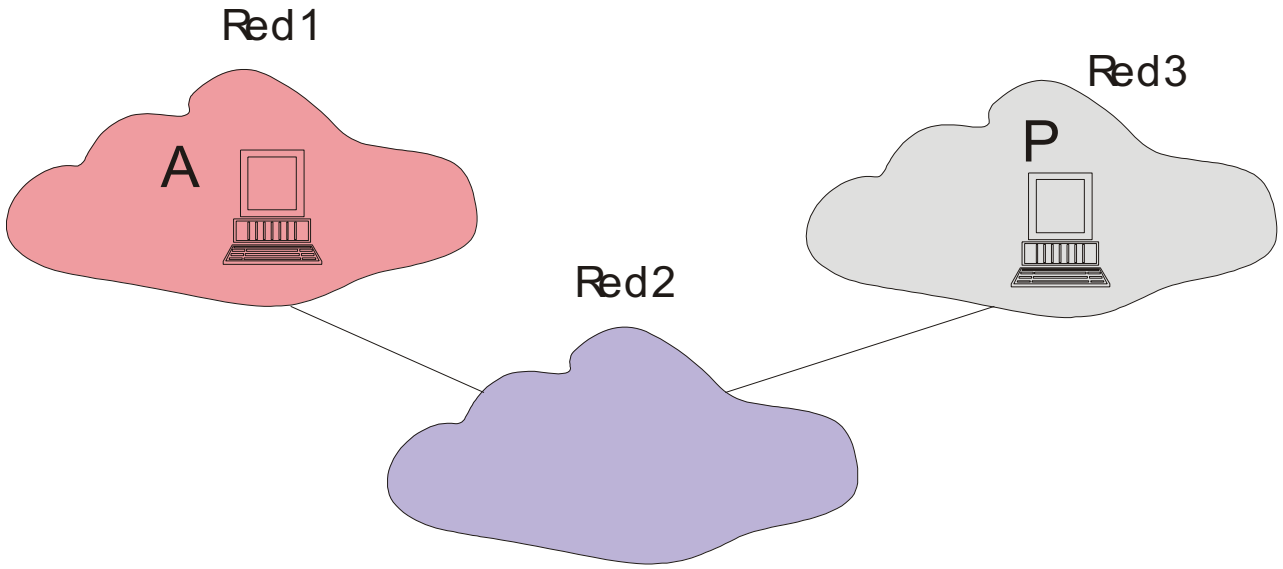
Nivel de enlace de datos



Nivel de red

- **Encaminamiento** de mensajes entre el nodo origen y el destino cuando entre ellos hay una distancia de n saltos (n>1)

- Ejemplo: envío de datos de un nodo con dirección de red A al nodo con dirección de red P



Nivel de transporte

- Entrega extremo a extremo (proceso a proceso)
- Entrega fiable
  - Control de errores
  - Control de secuencia
  - Control de pérdida
  - Control de duplicación
- Multiplexación
  - Hacia arriba (nivel de sesión)
  - Hacia abajo (nivel de red)
- Comunicación
  - Orientado a conexión
  - No orientado a conexión

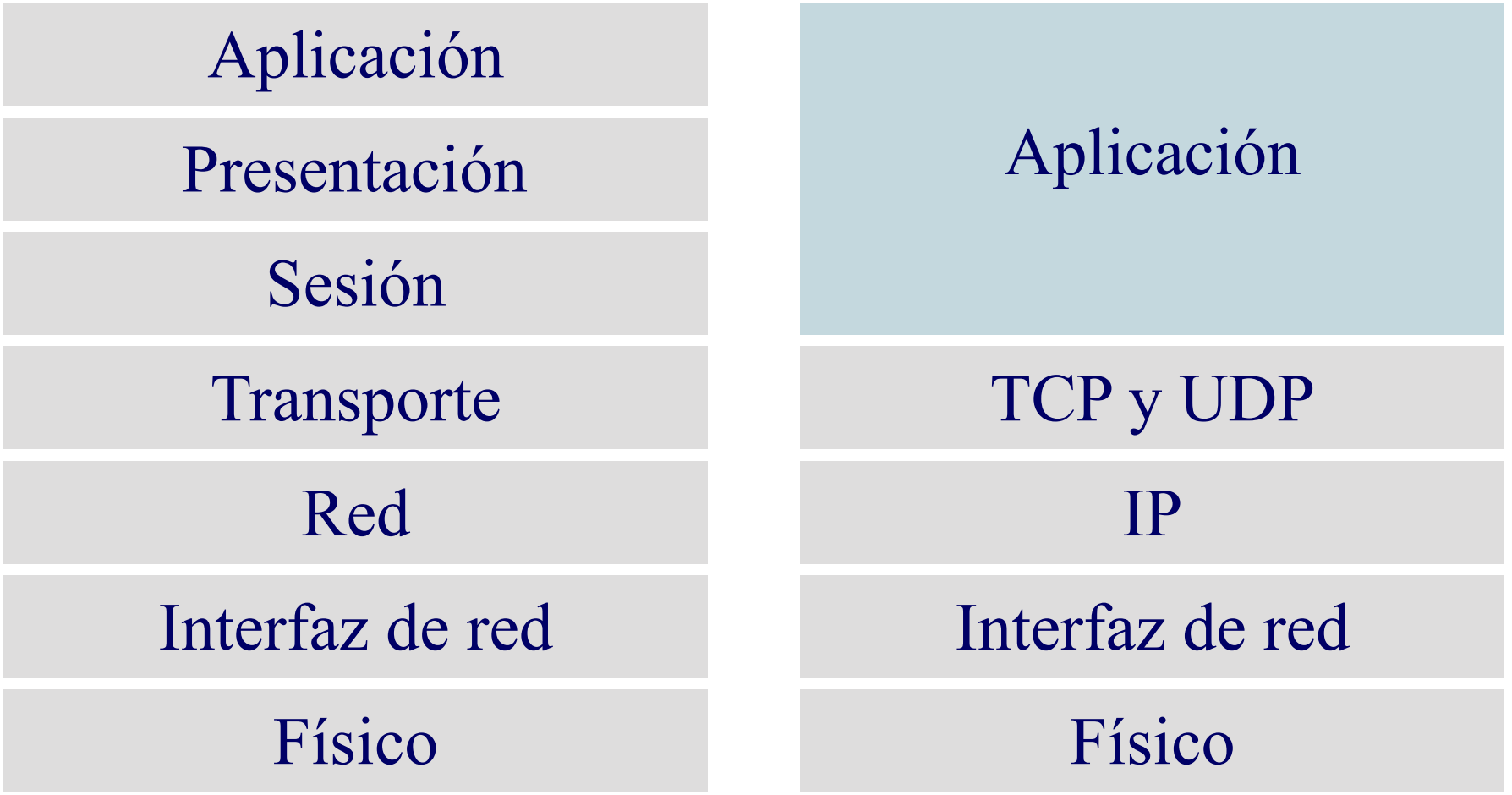
Protocolos TCP/IP

- Resultado de dos décadas de investigación y desarrollo llevados a cabo en la red ARPANET (financiada por DARPA) en los años **70**

- Familia de protocolos utilizados en Internet

- En los años **90** se ha establecido como la arquitectura comercial dominante:
  - Se especificaron y utilizaron antes que OSI
  - Independiente de la tecnología de red utilizada
  - Internet está construida sobre un conjunto de protocolos TCP/IP
  - Espectacular desarrollo de *World Wide Web*

TCP/IP y modelo OSI



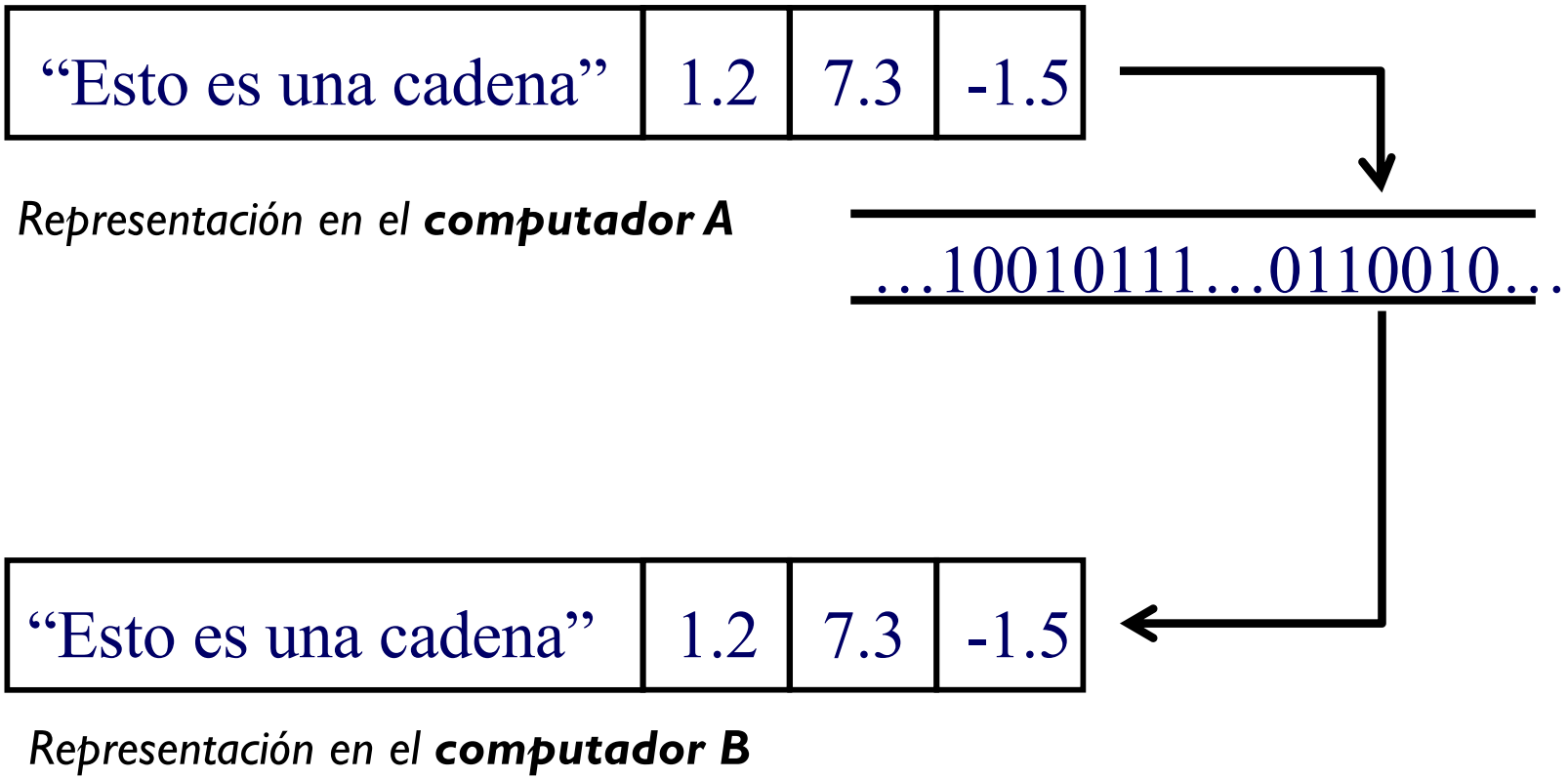
Modelo de referencia OSI

Pila de protocolos TCP/IP

Niveles de sesión, presentación y aplicación

- Nivel de sesión: encargado de proporcionar los mecanismos para controlar el diálogo de las aplicaciones de los sistemas finales
- Nivel de presentación: encargado de la representación de los datos que intercambian dos máquinas origen-destino (posiblemente heterogéneas) usando una representación de red independiente
  - Caracteres (ASCII, Unicode, EBCDIC)
  - Números (Little-endian, Big-endian)
  - Algunas de sus funciones:
    - Serialización y conversión de los datos
    - Cifrado de los datos
    - Compresión de los datos
- Nivel de aplicación: protocolos de alto nivel diseñados para satisfacer los requisitos de las aplicaciones
  - Definen la interfaz de acceso a un servicio
  - Ejemplos: DNS, FTP, HTTP, SSH, TELNET, SMTP, etc.

Serialización y representación de datos



Protocolo Internet (nivel IP)

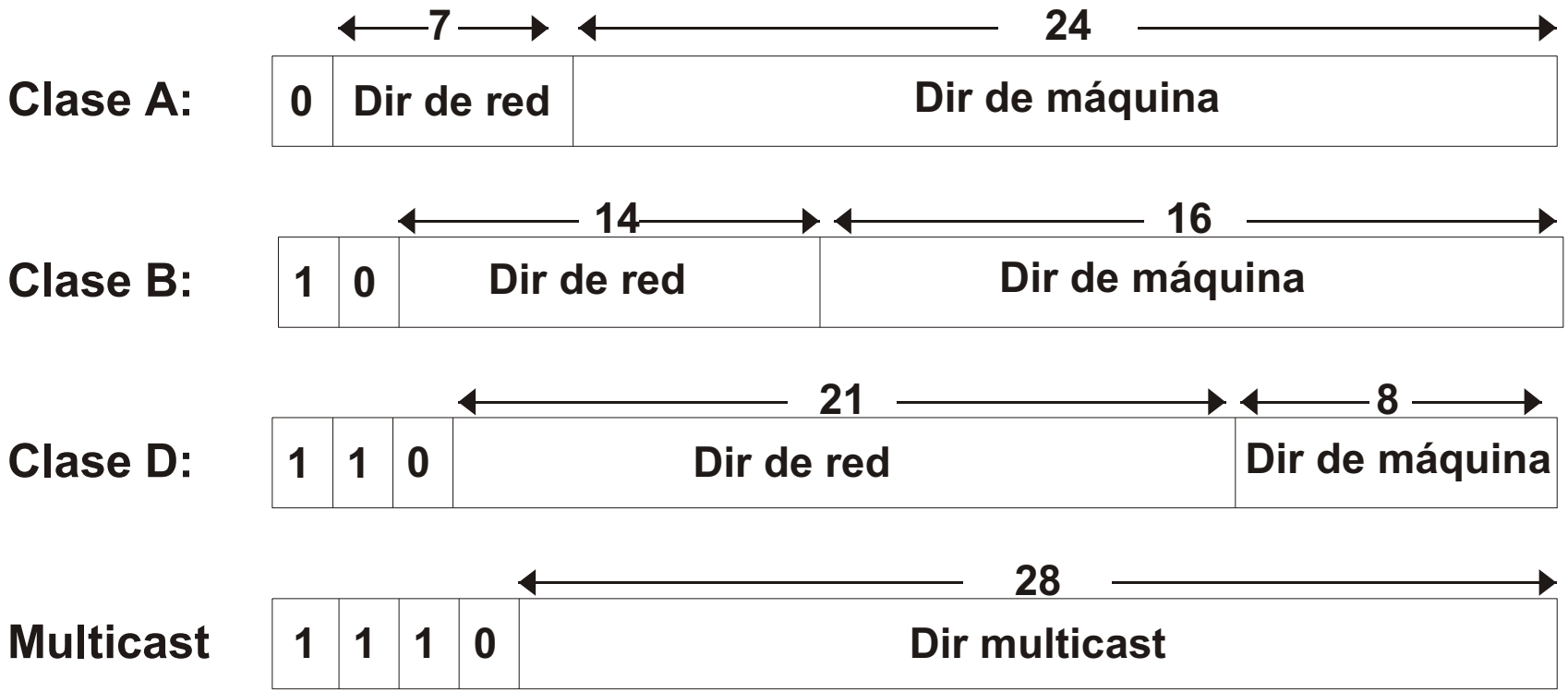
- Responsable de la transmisión de paquetes (datagramas IP) de tamaño variable (máximo 64 KB) con cabecera de 20 bytes (dirección IP origen y destino)
  - Si length(datagrama) > MTU de la red subyacente ➔ Fragmentar
  - Ejemplo: MTU de Ethernet es 1500 bytes
- Se corresponde con el nivel de red del modelo OSI
- Protocolo no orientado a conexión
- El encaminamiento se realiza mediante tablas de encaminamiento
  - Estáticas (las más usadas) o dinámicas
  - Si se utiliza encaminamiento dinámico un datagrama podría viajar de forma indefinida
    - Solución: contador de saltos (*Time To Live*)

Protocolo Internet (nivel IP)

- Semántica *best-effort*
  - Transmisión no fiable
  - No hay garantía de entrega
  - Mecanismo de comprobación de errores basado en *checksum* de la cabecera
  - Los paquetes se pueden descartar por: Expiración del tiempo de vida (TTL), congestión, error en la suma de comprobación (*checksum*)

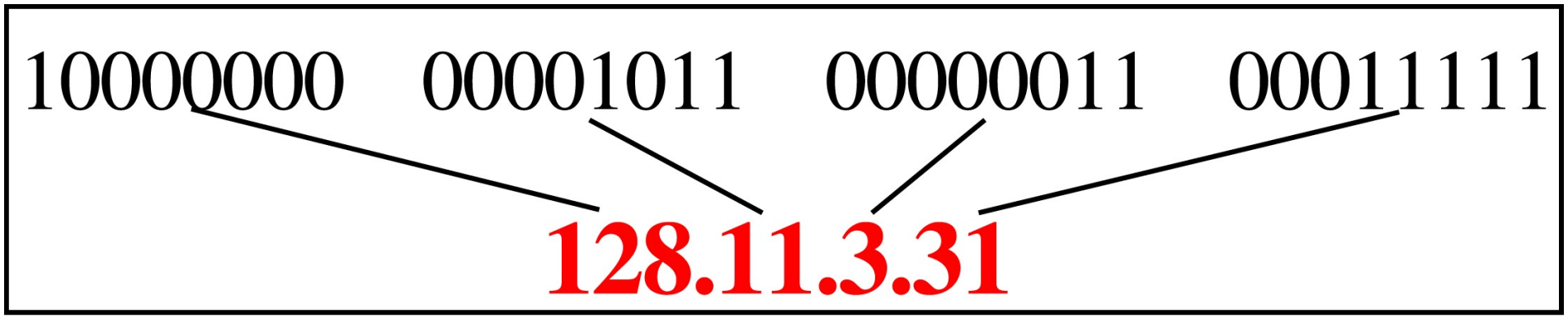
Direcciones IPv4

- Cada computador tiene una dirección IP (32 bits) que le identifica de forma única
  - 2^{32} hosts direccionables

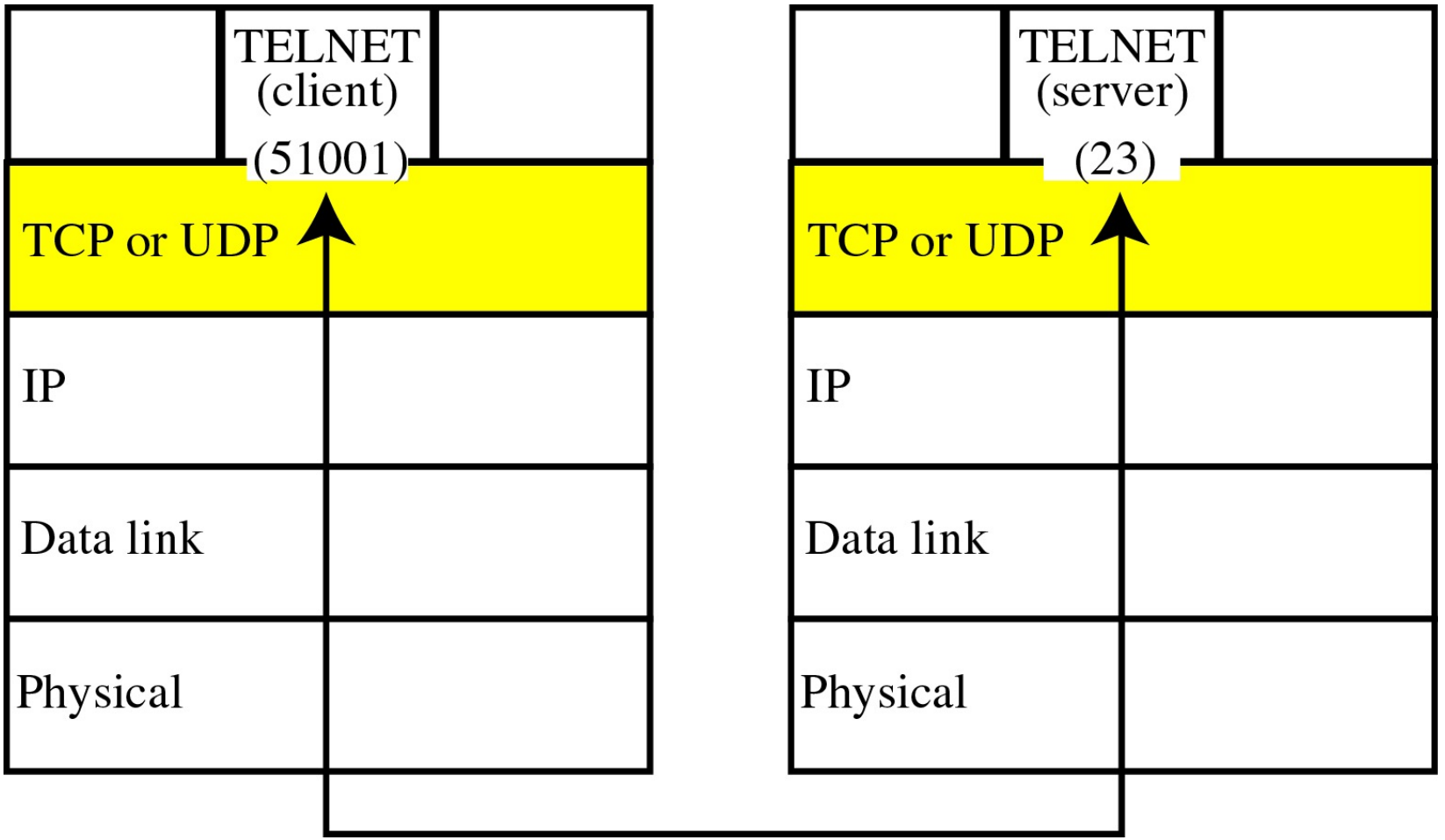


Dirección IP en notación decimal

- Una dirección IPv4 se expresa como una secuencia de 4 bytes separados por puntos.
  - El valor que puede tomar cada octeto es 0..255 (2^8-1)



Nivel de transporte



Direcciones del nivel de transporte

- Las direcciones IP identifican de forma única una máquina
- Un puerto es un número de 16 bits que identifica de forma única a un proceso en una máquina
- Uso de los puertos:
  - Permitir a los protocolos de transporte comunicación proceso a proceso
  - Una aplicación que quiere recibir un mensaje debe abrir un puerto
  - Un aplicación que quiere enviar un mensaje a otra debe conocer la dirección IP de la máquina donde ejecuta y el puerto asociado
- Los servicios de Internet *well-known* usan puertos predefinidos (0..1023) que no pueden ser usados por ningún otro proceso
  - La autoridad central encargada de registrar puertos a servicios es la IANA
  - Los puertos por encima de 1023 están disponibles para uso privado
  - **Ejemplos:** *ftp* utiliza el puerto 21, *telnet* el 23, *http* el 80

Protocolos de transporte

- **Protocolo TCP (Transport Control Protocol):**
  - Orientado a conexión
  - Garantiza que los datos se entregan en el mismo orden en el que se envían
  - Las conexiones TCP se ven como un flujo de bytes
    - El flujo de bytes se fragmenta en segmentos de datos y se transfieren como paquetes IP
  - La transmisión se considera "*fiable*".
    - Números de secuencia para ordenar los segmentos recibidos en el destino
    - Control de flujo
    - Retransmisiones (ACKs)
    - Buffering
    - *Checksum* sobre los datos del segmento
  - Cuando los mensajes son muy pequeños, TCP los retrasa hasta conseguir uno más grande
    - Esta opción debe desactivarse si es necesario
  - Escrituras concurrentes sobre una misma conexión TCP pueden provocar que los datos se entremezclen

Protocolo de transporte

- **Protocolo UDP (User Datagram Protocol):**
  - Protocolo de datagramas no orientado a conexión
  - Protocolo no fiable
    - Los paquetes se pueden perder, duplicar, recibir en orden distinto al enviado
  - Tamaño máximo del mensaje: 64 KB
    - Un datagrama UDP se encapsula dentro de un paquete IP

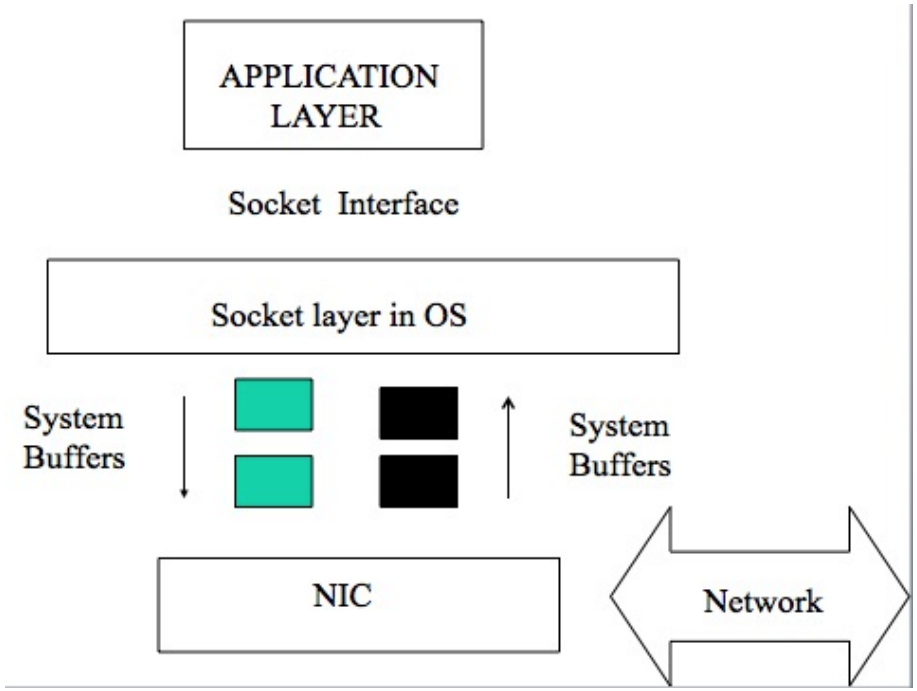
Comparaciónón entre TCP y UDP

Carácterística	TCP	UDP
Unidad de datos	Flujo de datos	datagrama
Conexión	Orientado a conexión	Sin conexión previa
Fiabiliadad	Alta. Confirmación de datos y retransmisión de datos perdidos	No
Control de flujo	Si	No
Control de congestión	Si	No
Números de secuencia	Si	No
Ordenación de mensajes y duplicación	Si	No
Manejo de mensajes perdidos	Se retransmiten	Mo
Buffering	Datos transmitidos como un flujo	Dtagramas individuales
Sobrecarga y efciciencia	Alta	Baja

Papel del sistema operativo

- Los sistemas operativos proporcionan implementaciones de los protocolos de comunicaciones

- El SW de comunicación de un sistema operativo se organiza como un conjunto de componentes con tareas concretas
  - Subsistema de almacenamiento: *buffers* donde almacenar los paquetes que llegan y se envían (limitado)



Problemas de fiabilidad

- Pérdida de datagramas entre dos procesos que ejecutan en la misma máquina. Intercambio de 300000 datagramas.

Problemas de fiabilidad

- Pérdida de datagramas entre dos procesos que ejecutan en la misma máquina. Intercambio de 300000 datagramas.

Mensaje (bytes)	% datagramas perdidos	longitud de la ráfaga máxima
64	0.17	177
128	0.09	274
256	0.034	102
512	0.12	373
1024	0.32	959
2048	0.16	482
4096	0.18	534
8192	0.41	64

Problemas de fiabilidad

- Pérdida de datagramas entre dos procesos que ejecutan en dos máquinas. Intercambio de 300000 datagramas.

Problemas de fiabilidad

- Pérdida de datagramas entre dos procesos que ejecutan en dos máquinas. Intercambio de 300000 datagramas.

Mensaje (bytes)	% datagramas perdidos	longitud de la ráfaga máxima
64	0	
128	0	
256	0.08	17
512	0	
1024	0.004	14
2048	0	
4096	0.0036	5
8192	0.0073	21

Problemas de fiabilidad

- Pérdida de datagramas entre dos procesos que ejecutan en una instancia de máquina virtual de Amazon.

Mensaje (bytes)	% datagramas perdidos	longitud de la ráfaga máxima
64	1,95	1951
128	2,87	1972
256	3,63	2030
512	2,29	2072
1024	4,22	2095
2048	2,06	2060
4096	2	2049
8192	1,79	1712

Problemas de fiabilidad

- Pérdida de datagramas entre dos procesos que ejecutan en dos instancias de máquinas virtuales en Amazon.

Mensaje (bytes)	% datagramas perdidos	longitud de la ráfaga máxima
64	3,45	175
128	0,25	119
256	1,11	209
512	2,83	307
1024	1,97	341
2048	3,56	263
4096	*7,73*	99
8192	5,35	60

Funciones del SO en las comunicaciones

- El SW de comunicación de un sistema operativo se organiza como un conjunto de componentes con tareas concretas
  - Subsistema de almacenamiento: *buffers* donde almacenar los paquetes que llegan y se envían (limitado)
- En implementaciones UNIX típicas
  - TCP reserva para cada puerto (socket) un buffer de 8 KB y UDP 2 buffers de 8KB. El tamaño se puede incrementar hasta 64 KB.
  - Los mensajes a enviar se copian a estos buffers
  - El contenido de estos buffers se fragmenta y se copian a nuevos bloques de memoria a utilizar por IP
  - IP envía finalmente los paquetes por la interfaz de red correspondiente

Sobrecarga introducida por el SO

- Procesamiento de TCP/IP
- Cambios de contexto
- Copias de datos en buffers intermedios

Sistemas Distribuidos
Problemas de diseño

- Heterogeneidad de los componentes
- Nombrado
- Comunicación y sincronización
- Rendimiento
- Concurrencia
- Capacidad de crecimiento, escalabilidad
- Estructura de software
- Tolerancia a fallos
- Calidad de servicio (QoS)
- Transparencia

Heterogeneidad

- Heterogeneidad de los componentes:
  - Es la variedad y diferencia de los siguientes componentes:
    - Redes
    - HW de computadores
    - Sistemas operativos
    - Lenguajes de programación
    - Aplicaciones

¿Cómo resolver la heterogeneidad?

- Empleo de sistemas abiertos y estándares (es la característica del sistema que determina si el sistema puede ser extendido y re-implementado)
  - Especificaciones e interfaces de acceso públicas (ej. RFCs)
  - Mecanismos de comunicación uniformes
  - Se pueden construir sobre SW y HW heterogéneo

- Ejemplos de sistemas abiertos:
  - TCP/IP
  - NFS
  - CORBA (www.omg.org)
  - HTTP
  - XML

Nombrado

- Los usuarios designan a los objetos mediante un nombre (ej. www.uc3m.es)
- Los programas designan a los objetos mediante un identificador (ej. 163.117.131.31)
- Resolver un nombre implica obtener el identificador a partir del nombre
- **Objetivo importante**: los nombres deben ser independientes de su localización
- Consideraciones de diseño a tener en cuenta:
  - El espacio de nombres (tamaño, estructura, jerarquía, ...)
  - El servicio de nombres que realiza la resolución (ej. DNS)
  - Distribución y replicación del servicio de nombres

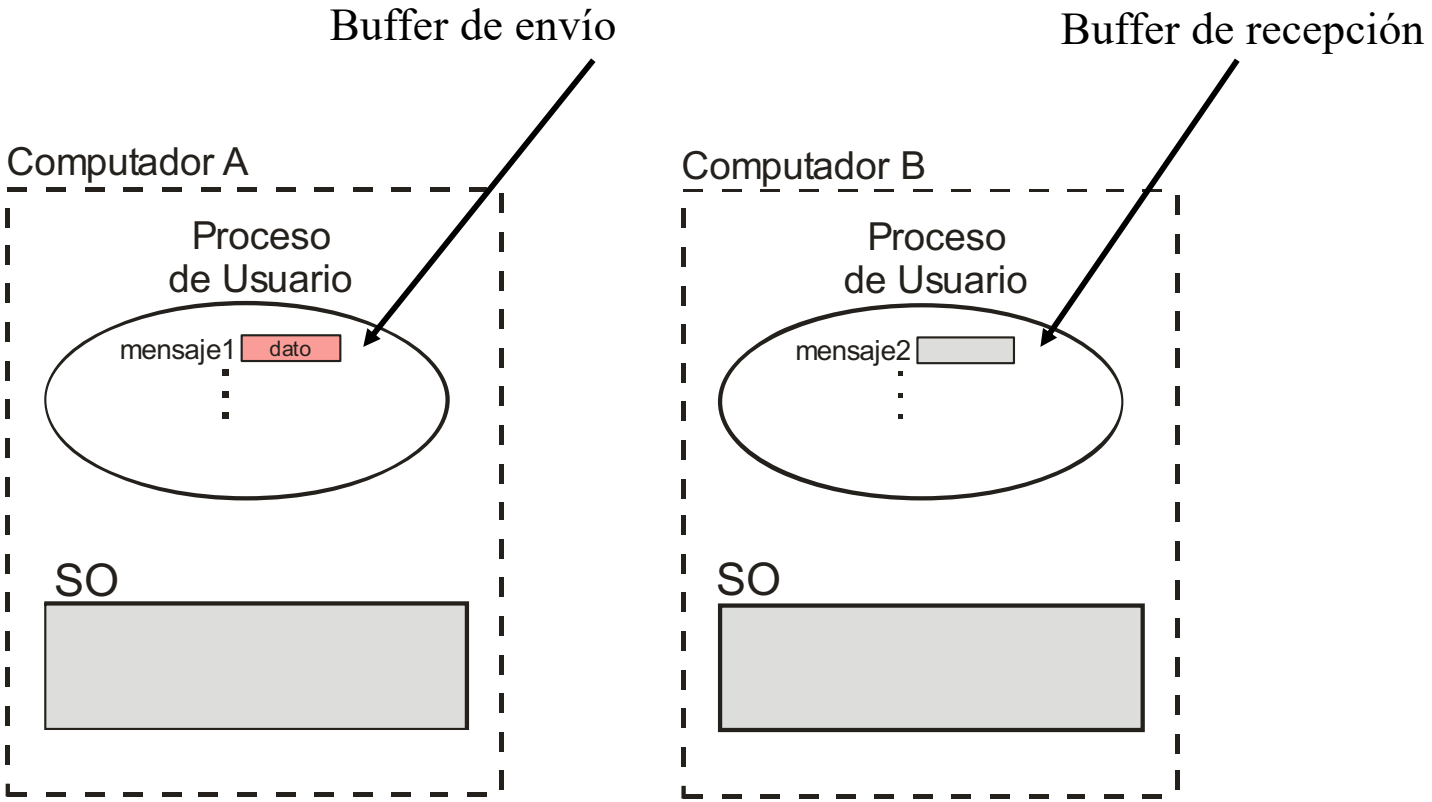
Comunicación y sincronización (C y S)

- Forma básica de C y S: paso de mensajes
  - Mecanismos síncronos
  - Mecanismos asíncronos

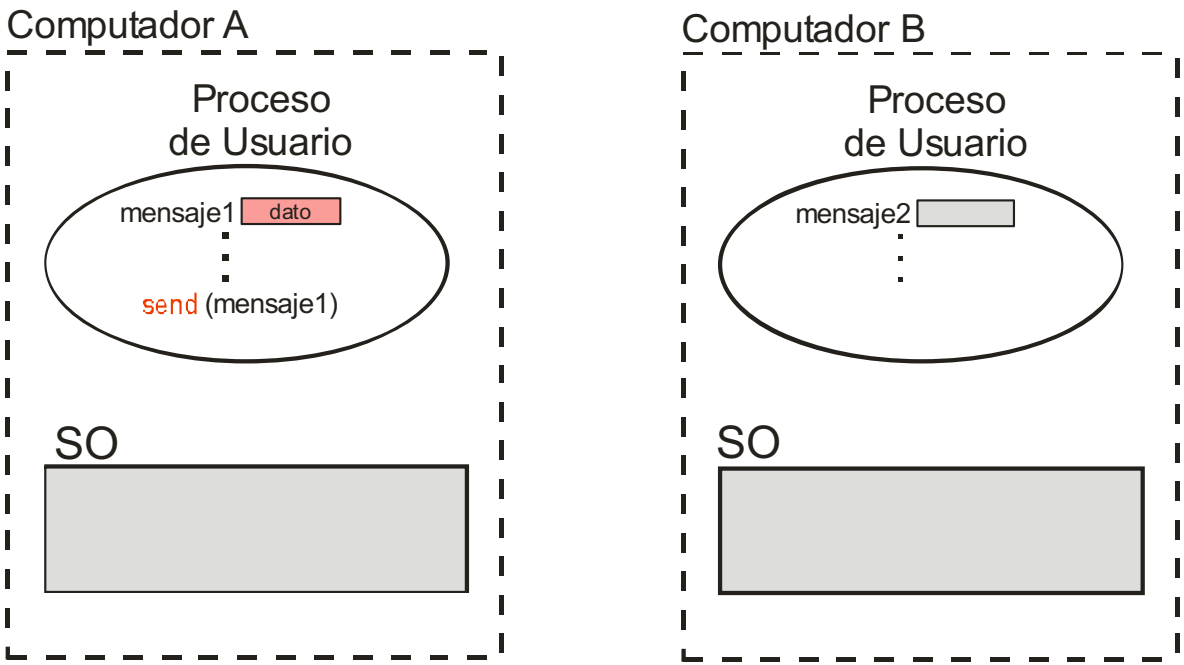
- Comunicación entre procesos:
  - Las entidades que se comunican en distintas máquinas son procesos
  - Primitivas básicas de comunicación:
    - send
    - receive
  - Llamadas a procedimientos remotos
  - Invocación de objetos remotos

- Comunicación en grupos
  - *Multicast*, *broadcast*
  - Útil para el trabajo en grupo, localizar el objeto, tolerancia a fallos, mejorar el rendimiento (replicación), asegurar la consistencia

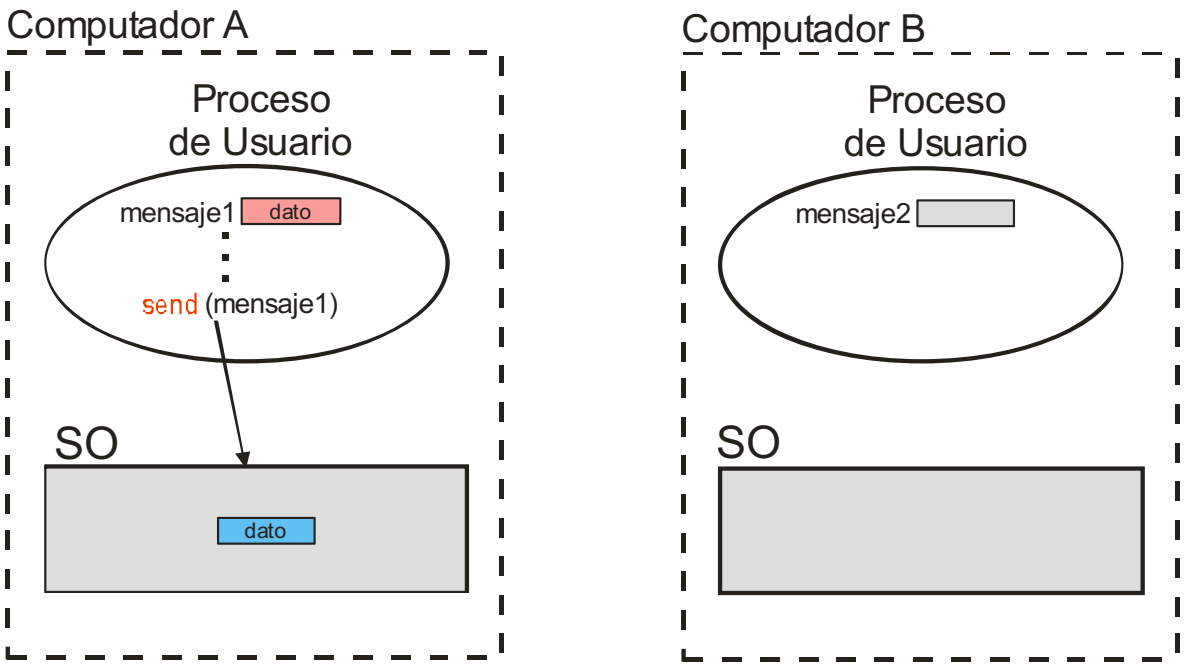
Paso de mensajes



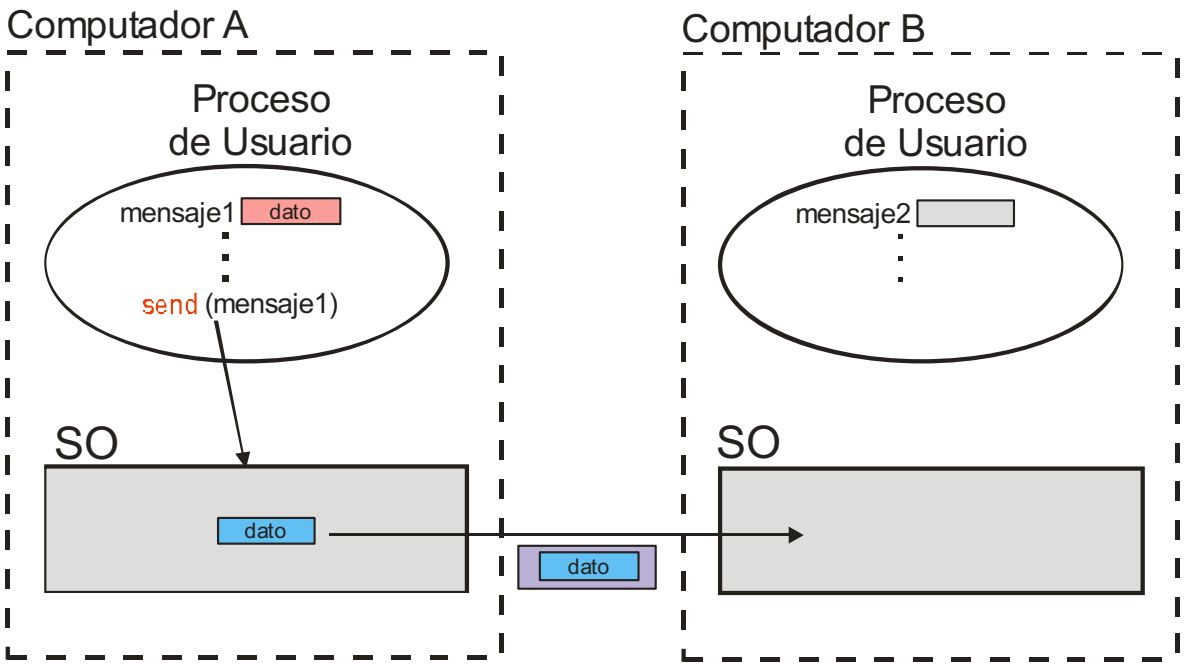
Paso de mensajes



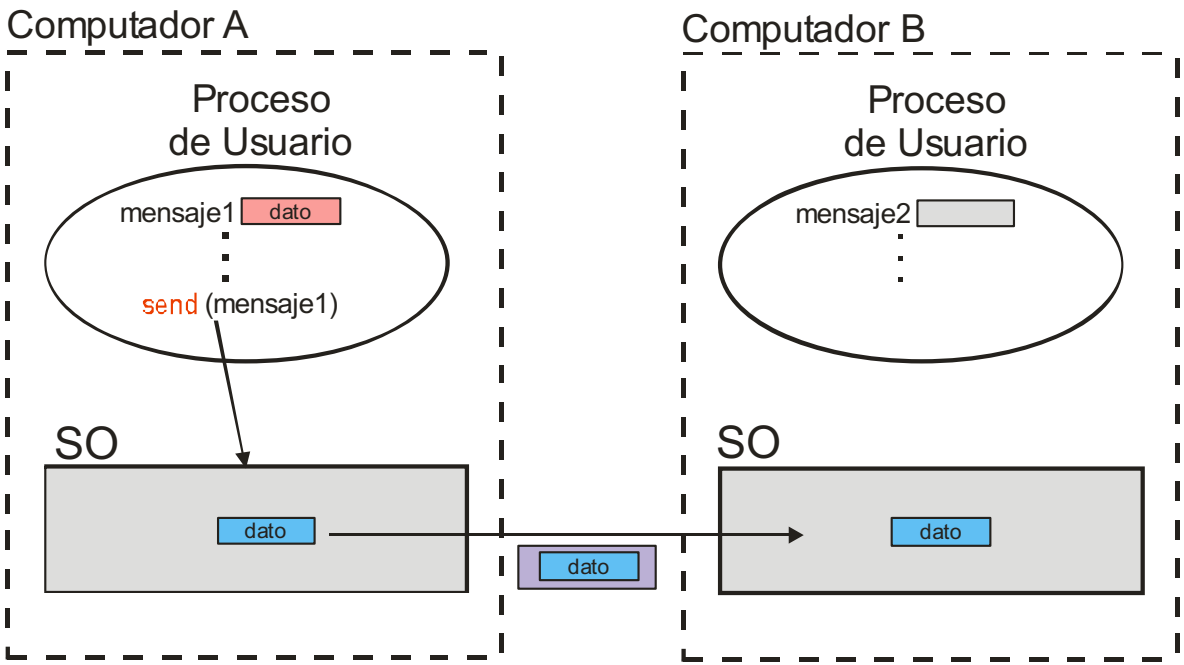
Paso de mensajes



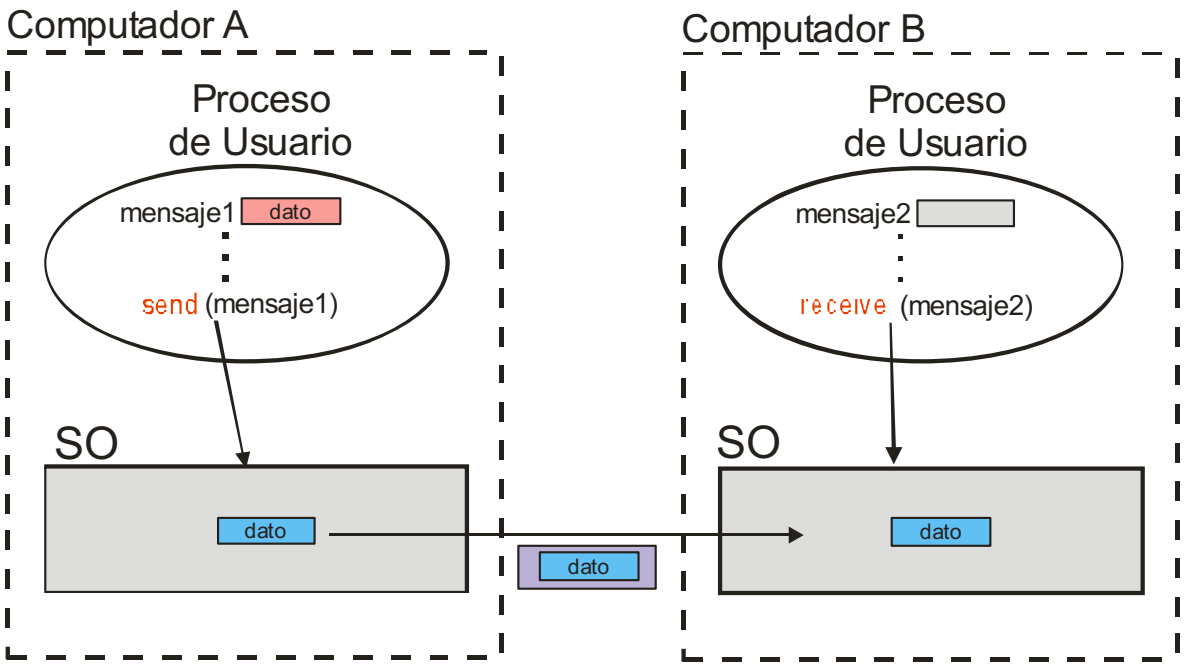
Paso de mensajes



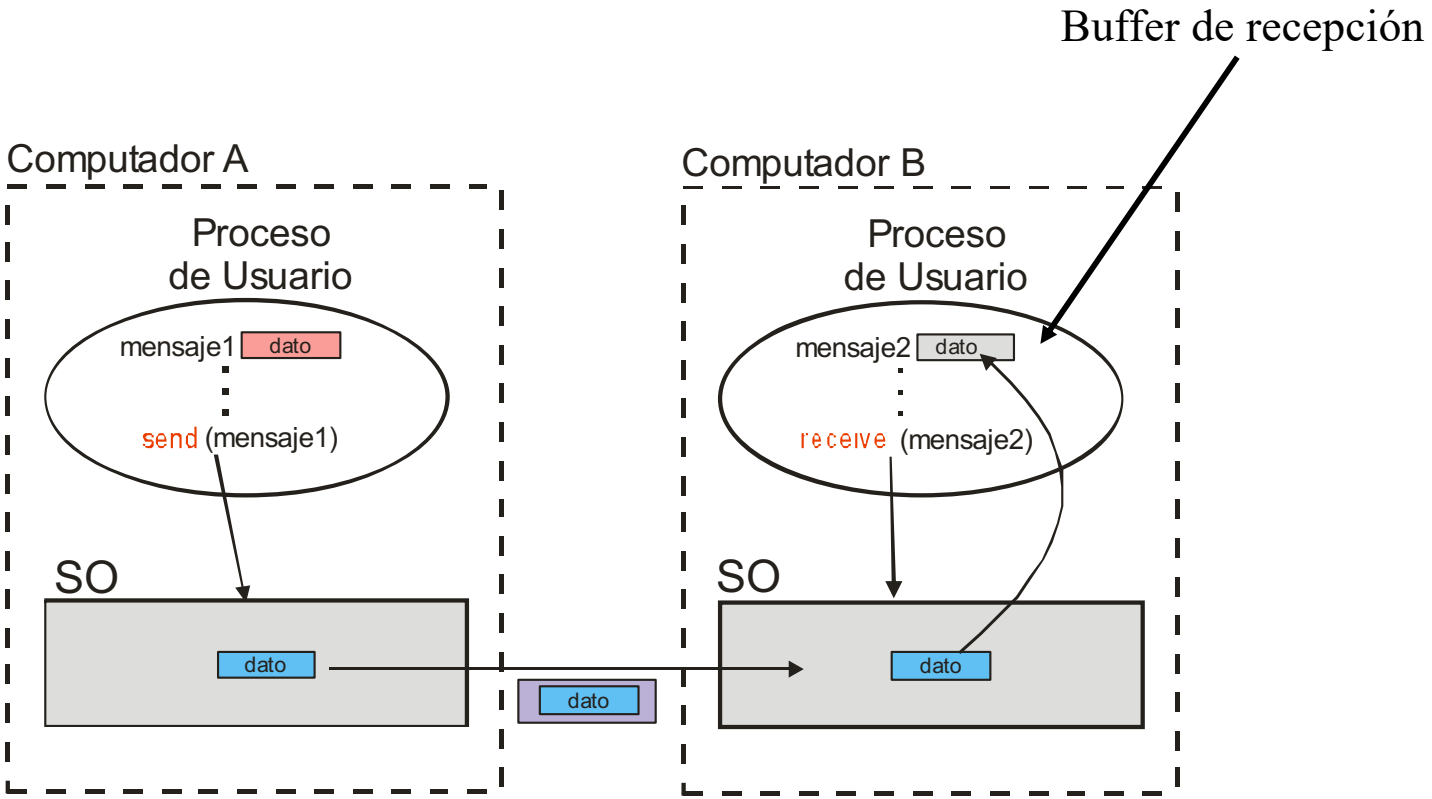
Paso de mensajes



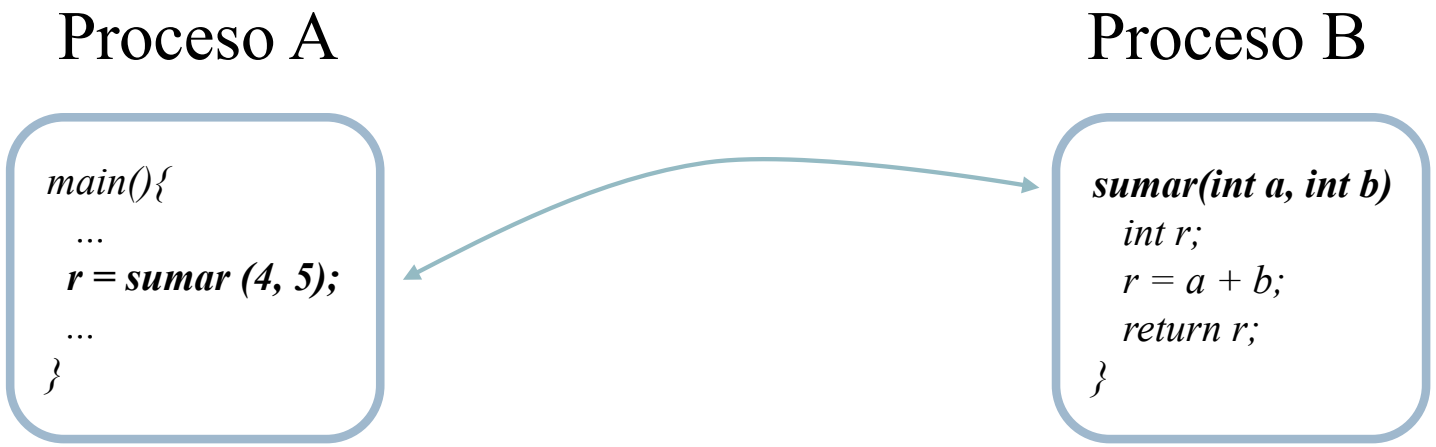
Paso de mensajes



Paso de mensajes



Llamadas a procedimientos/invocación de métodos



Rendimiento

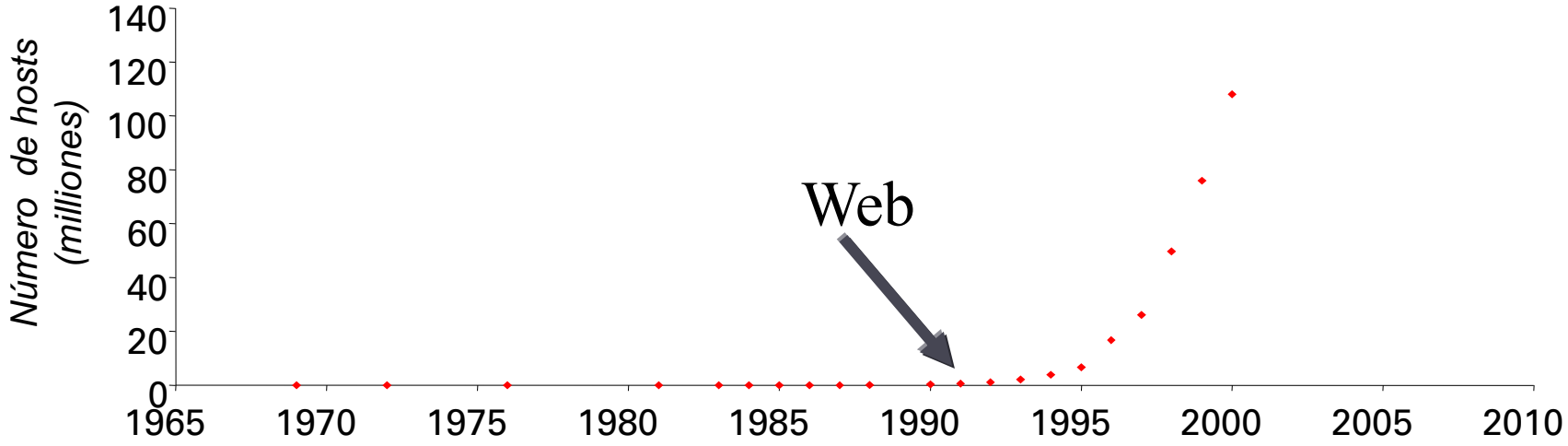
- Para un servicio multiusuario:
  - Objetivo: rendimiento no peor que un sistema centralizado
- Para la ejecución de una aplicación paralela:
  - Objetivo: rendimiento proporcional a los procesadores empleados
- Factores que afectan:
  - Infraestructura HW (red, nodos, etc)
  - Empleo de cahé: intentar que muchos accesos se hagan localmente
  - Empleo de replicación: reaparto de carga entre diferentes componentes

Control de concurrencia

- Un sistema distribuido es inherentemente concurrente
- Concurrencia: varias actividades se ejecutan simultáneamente
  - Permite reducir el tiempo de ejecución de un problema
- La concurrencia puede introducir condiciones de carrera (los resultados dependen de la secuencia de eventos)
- Los recursos compartidos deben protegerse (cerrojos, semáforos, etc.) y asegurar:
  - Exclusión mutua
  - Progreso
  - Espera acotada

Capacidad de crecimiento

- Un sistema posee capacidad de crecimiento o escalabilidad si conserva su efectividad cuando se incrementa significativamente el número de recursos o usuarios.
- Ejemplo: crecimiento de Internet

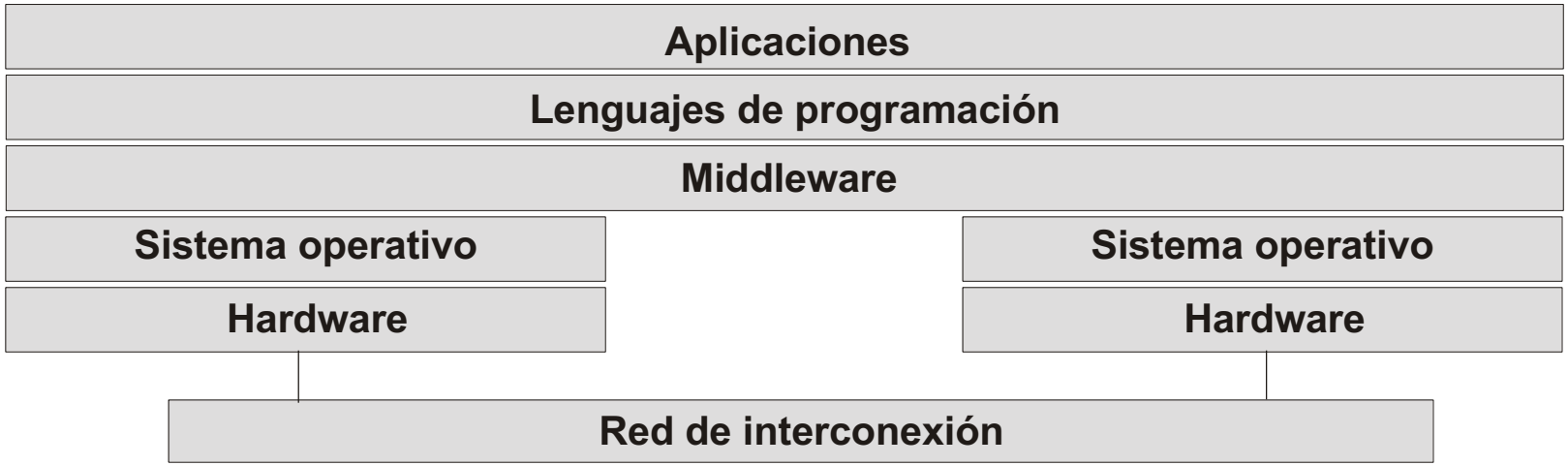


Aspectos a considerar en la escalabilidad

- Para que un sistema con n usuarios sea escalable, la cantidad de recursos necesarios para soportarlo debería ser proporcional a n *ó* O(n)
- Técnicas de escalabilidad:
  - Replicación
    - Datos: múltiples copias del mismo dato
    - Procesos: múltiples ejecuciones de la misma computación
  - Caching
    - Copias locales de datos recientemente o más utilizados
  - Distribución
    - Distribución de carga en múltiples nodos
      - DNS, DHT
  - La replicación y las técnicas de caching pueden introducir problemas de consistencia y coherencia

Estructura del software: Middleware

- Software que ofrece servicios que permiten enmascarar la heterogeneidad de las redes subyacentes, HW, SO y Lenguajes de programación
- Servicios y protocolos estandarizados: sistemas abiertos
- Ofrecen servicios no incluidos en el SO (construcción de aplicaciones distribuidas, servicios de ficheros distribuidos, servicios de nombres, ...)
- Facilitan el desarrollo de aplicaciones distribuidas
- Independientes del HW y del SO subyacente



Tolerancia a fallos y fiabilidad

- Fiabilidad es la probabilidad de que un sistema funcione o desarrolle cierta función, bajo condiciones fijadas y durante un período de tiempo
- Un sistema distribuido es inherentemente más propenso a errores
- Un sistema es tolerante a fallos si el sistema cumple sus especificaciones a pesar de la presencia de fallos
- Se debe asegurar:
  - Disponibilidad: los recursos son disponibles a pesar de que haya fallos.
  - Atomicidad: la consistencia de los recursos se debe asegurar a pesar de fallos

Tolerancia a fallos

- Leslie Lamport: A distributed system is **one in which the failure of a computer you didn't even know existed can render your own computer unusable**

Tipos de fallos

- Fallos hardware
  - Por ejemplo un disco duro tiene un tiempo medio de fallos de 10 a 50 años (Brian Beach: "Hard Drive Reliability Update – Sep 2014," backblaze.com, September 23, 2014.)
    - En un cluster con 10.000 discos, habría un fallo en un disco cada día
  - Fallos software: fallos en las especificaciones, el diseño o la programación
  - Fallos humanos, por ejemplo errores en configuración

Consistencia

- El problema de la consistencia (coherencia) surge cuando varios procesos acceden y actualizan datos de forma concurrente
  - Coherencia de las actualizaciones
  - Coherencia de la replicación
  - Coherencia de caches
  - Coherencia ante fallos
  - Relojes consistentes

Seguridad

- Los recursos de información disponibles en los SSDD pueden tener un valor importante para los usuarios (ej. información bancaria)
- La seguridad tiene tres componentes:
  1. Confidencialidad:
     - protección contra el descubrimiento de datos por individuos no autorizados
  2. Integridad
     - protección contra la alteración o corrupción de los datos
  3. Disponibilidad:
     - protección contra la interferencia en los procedimientos de acceso a los recursos
- Otros problemas de seguridad
  - Ataques de denegación de servicio
  - Seguridad del código móvil

Calidad de servicio (QoS)

- Es la habilidad de satisfacer los requerimientos de tiempo cuando se transmiten y procesan flujos de datos multimedia en tiempo real
- Rendimiento de un sistema:
  - Tiempo de respuesta adecuado
    - Latencias
  - Tasa de transferencia de datos
    - Velocidad en la cual los datos pueden ser transferidos entre dos computadoras de la red, usualmente medido en bits por segundo (bps)
  - El rendimiento viene determinado por:
    - La red de comunicación
    - Los servicios de comunicación empleados
    - El sistema operativo
    - El soporte para la programación de sistemas distribuidos

Objetivo de un sistema distribuido: Transparencia

- **Ocultación** al usuario de los componentes que conforman un sistema distribuido:
  - Heterogeneidad: acceso a recursos heterogéneos de forma idéntica
  - Acceso: acceso a recursos locales y remotos de forma idéntica
  - Ubicación: Acceso a recursos sin conocimiento de su posición física en la red
  - Concurrencia: ejecución concurrente de procesos sin interferencias
  - Replicación: uso de múltiples réplicas de recursos sin conocimiento de su existencia
  - Fallos: permitir la ejecución aun en presencia de fallos
  - Movilidad: permitir el movimiento de los recursos y clientes sin afectar a su funcionamiento
  - Crecimiento: permitir la capacidad de crecimiento sin afectar al sistema y aplicaciones
  - Rendimiento: rendimiento similar en recursos locales y remotos
  - Implementación: ocultar detalles de implementación de los componentes

Falacias de los sistemas distribuidos

«The Eight Fallacies of Distributed Computing - Tech Talk»

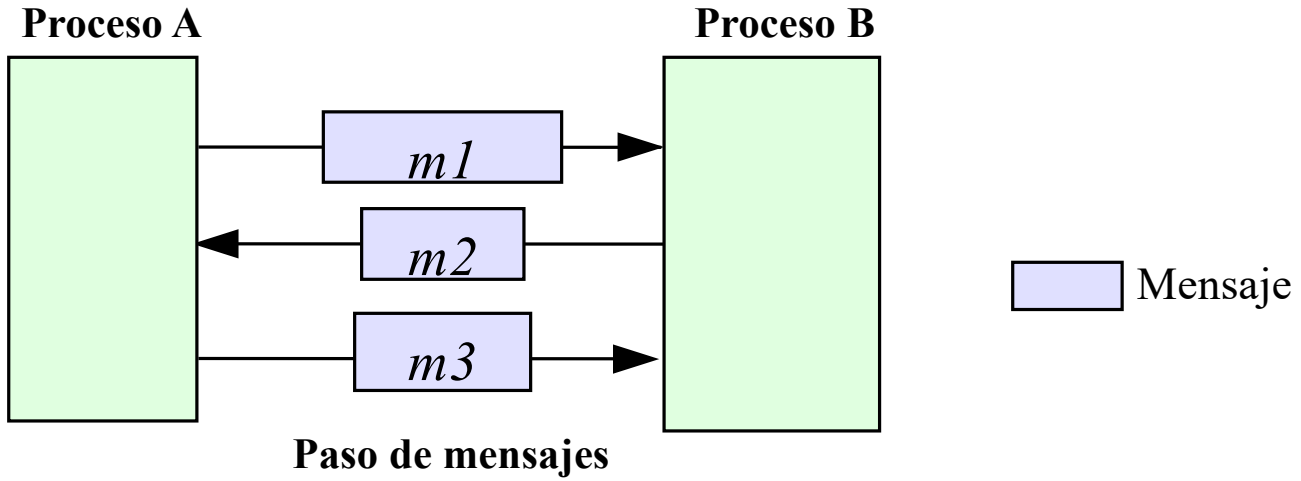
- La red es fiable
- La latencia es cero
- El ancho de banda es infinito
- La red es segura
- La topología no cambia
- Hay un administrador
- El coste de transporte es cero
- La red es homogénea

Paradigmas de comunicación

- Paso de mensajes (sockets)
- Colas de mensajes
- Llamadas a procedimientos remotos (RPC)
- Invocación de métodos remotos (RMI)
- Objetos distribuidos (CORBA)
- Comunicación de grupos
- Servicios web

Paso de mensajes

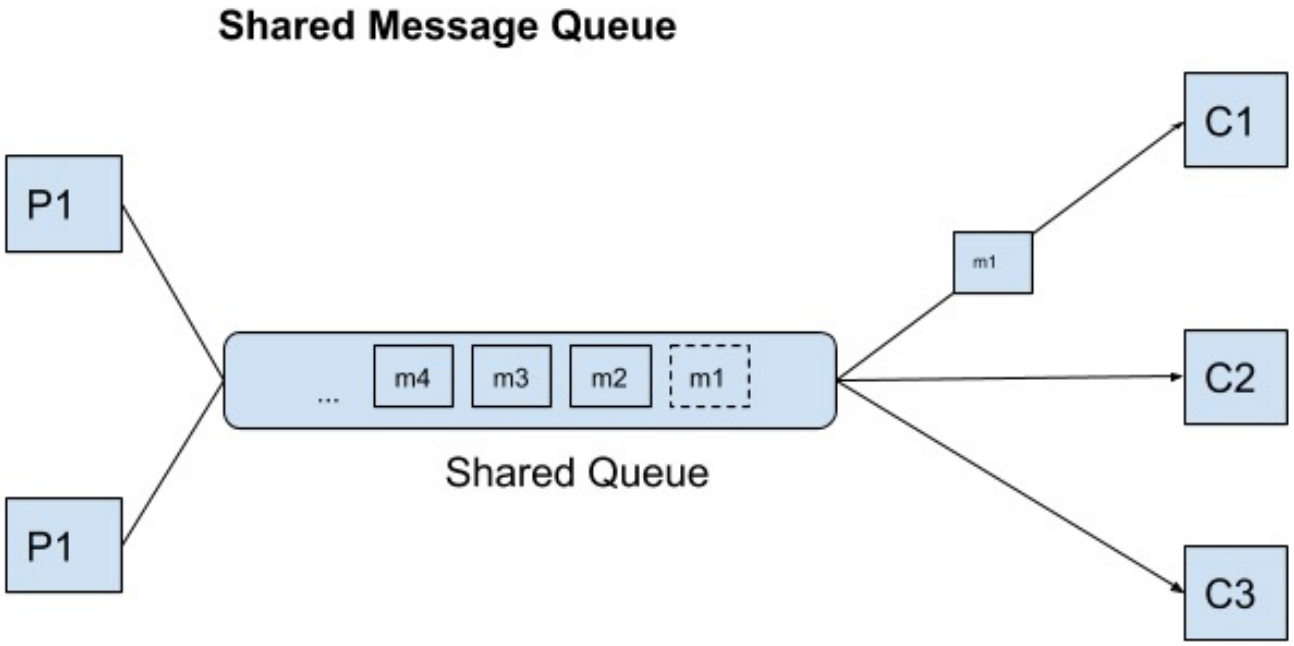
- Paradigma **fundamental** para aplicaciones distribuidas
- Un proceso emisor envía un mensaje de solicitud
- El mensaje llega al proceso receptor, el cual procesa la solicitud y devuelve un mensaje en respuesta
- Esta respuesta puede originar posteriores solicitudes por parte del proceso emisor



Paso de mensajes

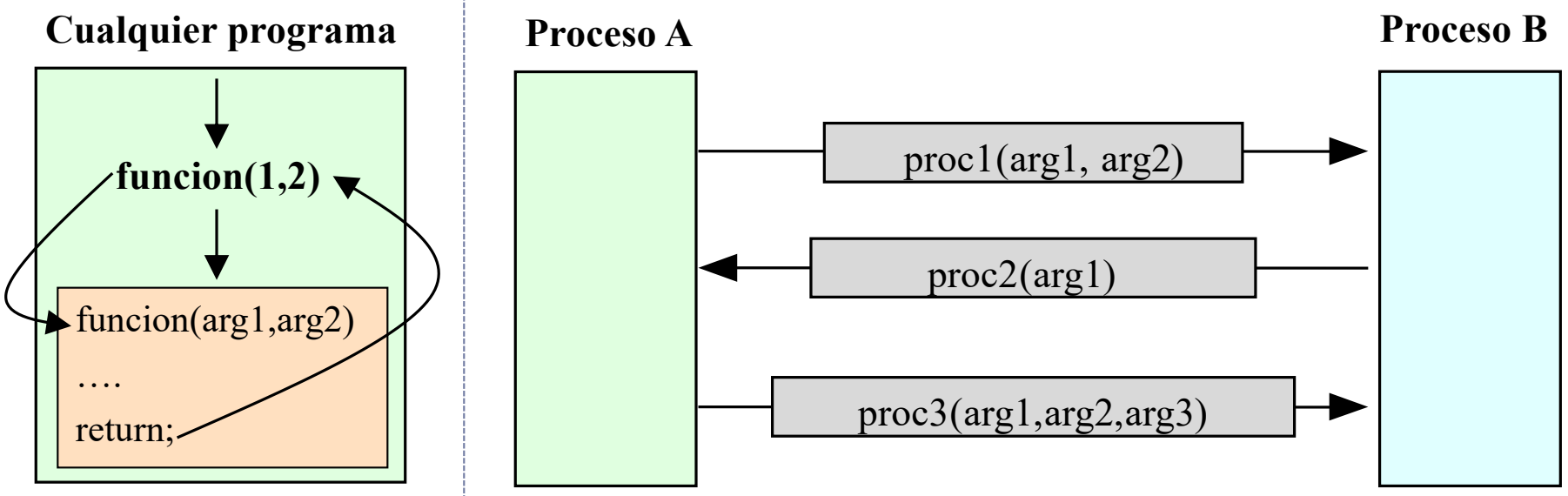
- Operaciones básicas:
  - Enviar (*send*)
  - Recibir (*receive*)
- Modelos de comunicación:
  - Orientadas a conexión
    - Operaciones para conectar y desconectar
  - No orientadas a conexión
- Ejemplo: **sockets**

Colas de mensajes



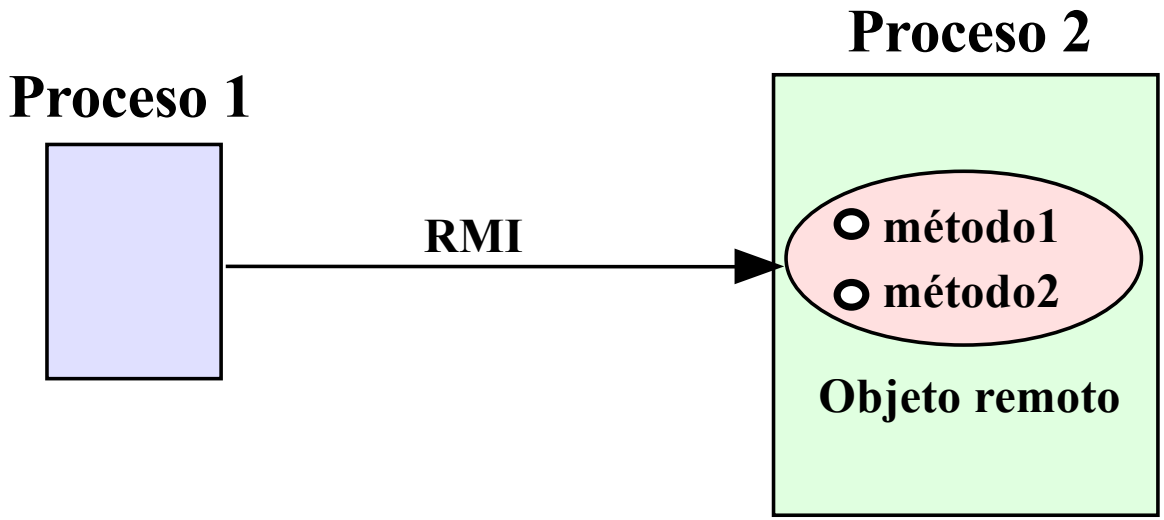
Llamadas a procedimientos remotos

- Idea: hacer que el software distribuido se programe igual que una aplicación no distribuida
- Conceptualmente igual que la invocación de un procedimiento local o método local



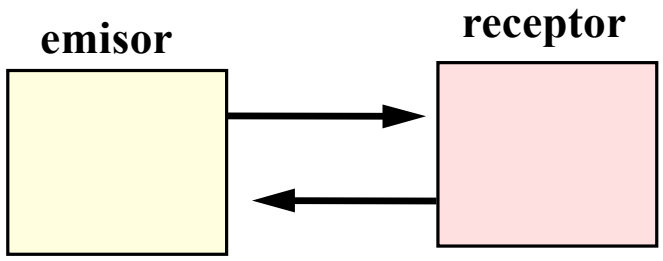
Invocación de métodos remotos

- Modelo equivalente a las llamadas a procedimientos remotos
- Proceso invoca un método local de otro proceso
- **Ejemplos**: CORBA, RMI de Java, Microsoft COM, DCOM, Java Beans, .NET Remoting

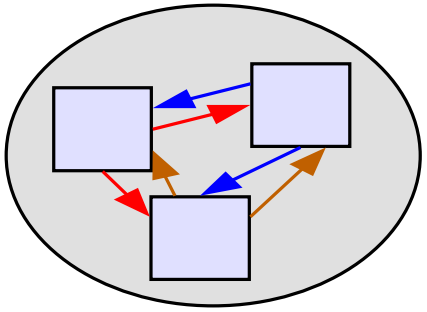


Comunicación de grupos

- Utiliza mensajes *multicast*
- Útil para:
  - Ofrecer tolerancia a fallos basado en servicios replicados
  - Localizar objetos en sistemas distribuidos
  - Mejor rendimiento mediante datos replicados
  - Actualizaciones múltiples
  - Operaciones colectivas en cálculo paralelo



IPC uno-a-uno



IPC grupo o multidifusión

Servicios web

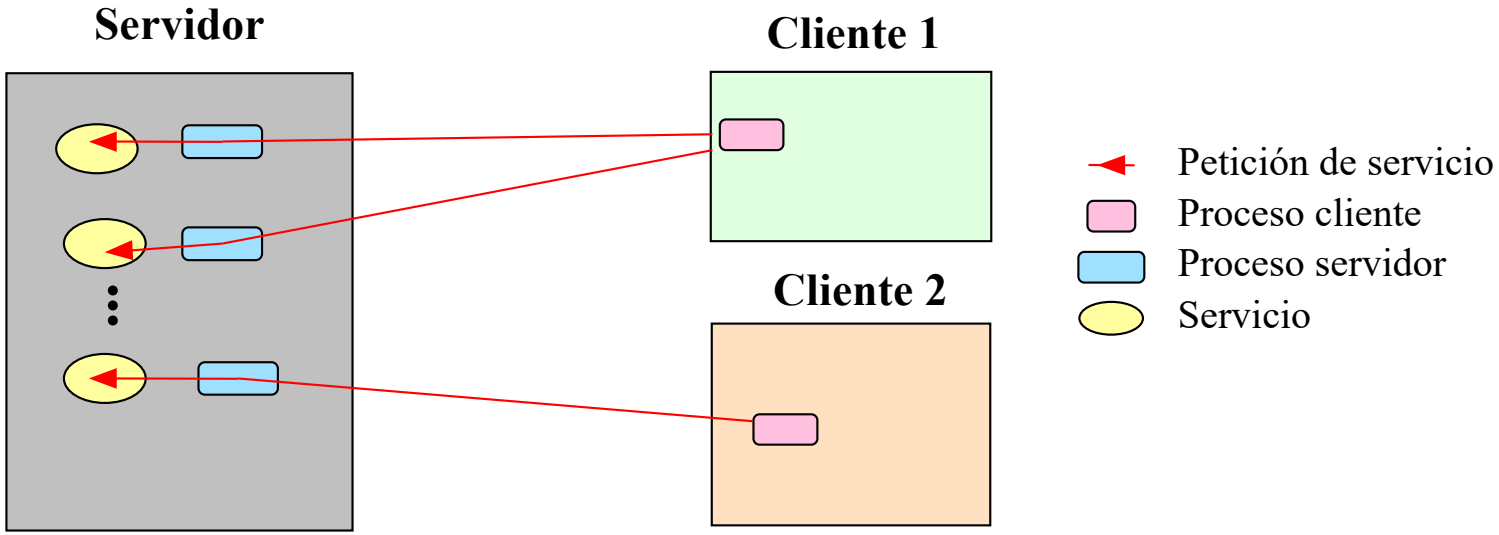
- Modelo de comunicación entre aplicaciones que utiliza como protocolo de transporte HTTP
  - El servicio se solicita utilizando HTTP
- Tipos:
  - SOAP (Simple Object Access Protocol): utiliza mensajes codificados en XML para solicitar los servicios y devolver las respuestas
  - REST (Representational state transfer): servicios solicitados a través de los métodos HTTP mediante parámetros codificados en las URL y datos transmitidos en JSON o XML

Arquitecturas de comunicación

- Cliente/Servidor
- Editor/Subscriptor
- Peer-to-Peer
- Arquitecturas para computación distribuida (ejemplo: maestro/trabajador)

Cliente-Servidor

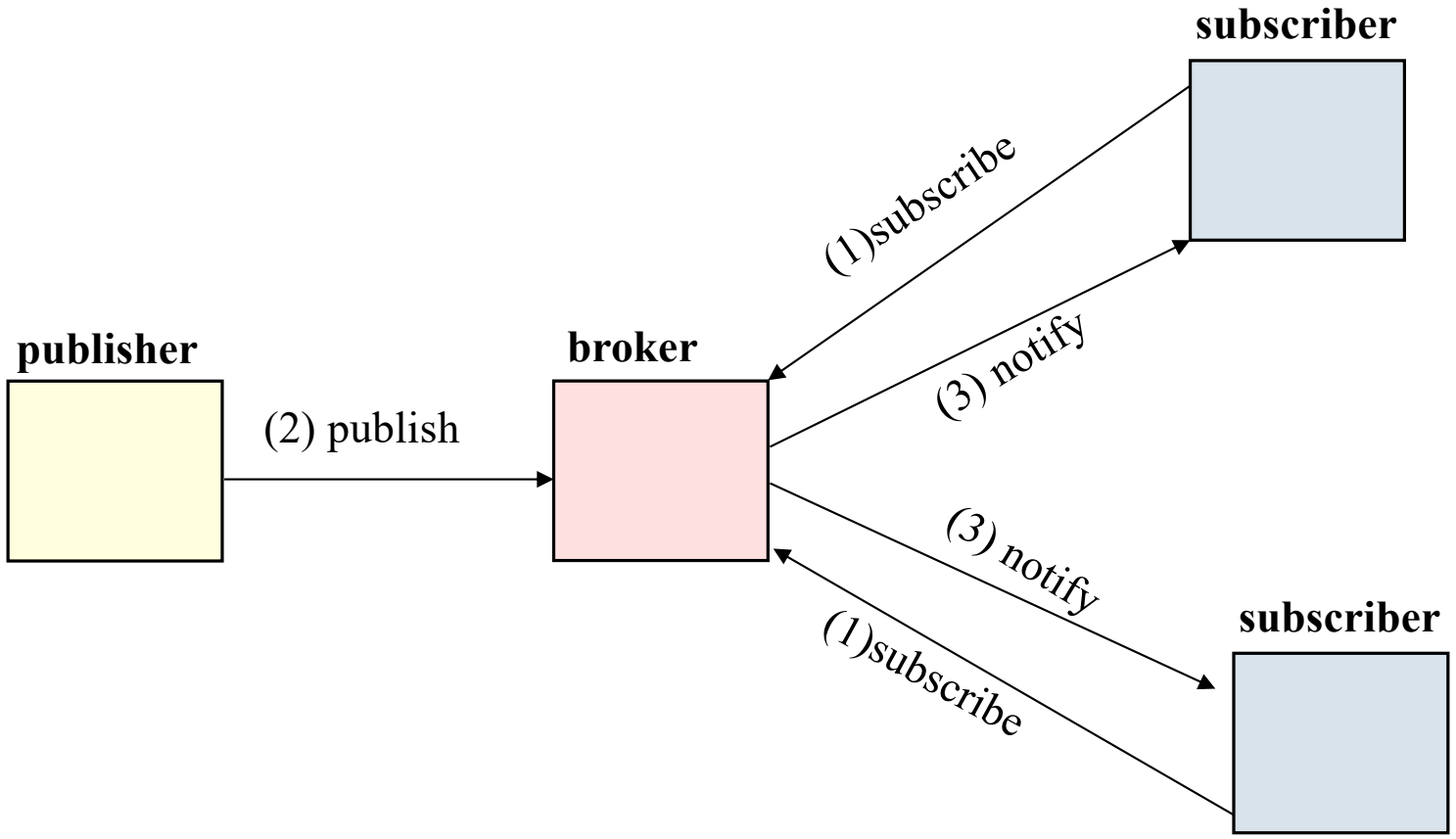
- Asigna roles diferentes a los procesos que comunican: cliente y servidor
- Servidor:
  - Ofrece un servicio
  - Elemento pasivo: espera la llegada de peticiones
- Cliente:
  - Solicita el servicio
  - Elemento activo: invoca peticiones



Cliente-Servidor

- Abstracción eficiente para facilitar los servicios de red
- La asignación de roles asimétricos simplifica la sincronización
- Implementación mediante:
  - Sockets
  - Llamada a procedimientos remotos (RPC)
  - Invocación de métodos remotos (RMI, CORBA, …).
  - Servcios Web
- Paradigma principalmente adecuado para servicios centralizados
- **Ejemplos**: servicios de Internet (HTTP, FTP, DNS, …)

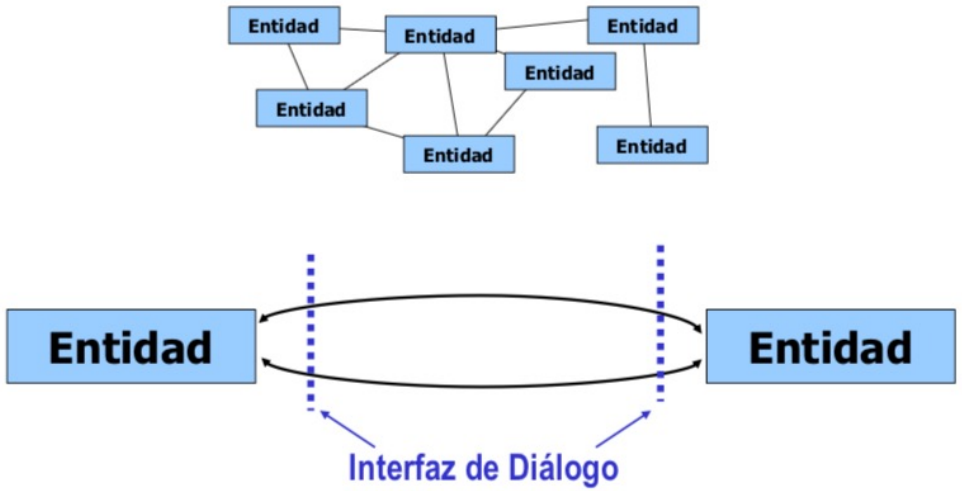
Modelo publicación-sucripción



- Ejemplo MQTT (MQTT: MQ Telemtry Transport) Protocolo ligero diseñado para la comunicación entre sensores y dispositivos en entornos de Internet of Things

Modelo peer-to-peer (P2P)

- Todos los nodos tienen el mismo rol y funcionalidad (cliente/servidor)
  - Se reducen los cuellos de botella y los puntos críticos
  - Se aprovechan los recursos de todas las máquinas
  - Modelos empleados en redes de distribución de contenidos

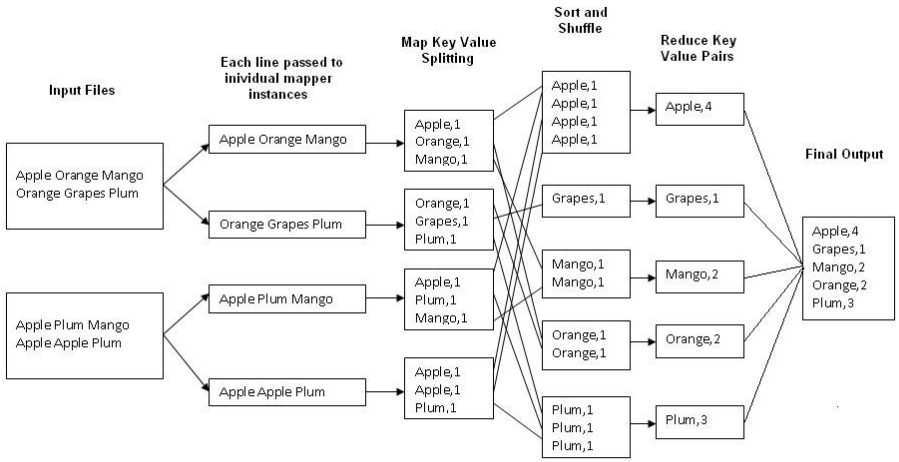


Arquitecturas para computación distribuida

- Modelo maestro-trabjador
  - Se reparten los trabajos entre los nodos trabajadores
  - Tolerancia a fallos:
    - Caída de un trabajador: se reasignan sus trabajos
    - Caída del maestro: se necesita replicación
- Ejemplo: MapReduce de Hadoop
  - Map: El trabajador procesa su parte de datos y genera (clave, valor)
  - P. ej: Extrae de logs web → (página, usuario que la accede)
  - Reduce: Procesa los valores asociados a una determinada clave
    - P. Ej: calcula el número de accesos únicos a cada página → (página, nº de accesos)

Modelo Map Reduce
Ejemplo: Word Count

```
void map(file, text) {
        foreach word in text.split() {
                output(word, 1); }
}
void reduce(word, list(count)) {
        output(word, sum(count));
}
```



Fuente: WordCountFlow.JPG, Wikimedia Commons, CC BY-NC-SA 2.0