Tema 2: EL MODELO CLIENTE/SERVIDOR

E. U. Informática en Segovia
Departamento de Informática
Universidad de Valladolid

Sistemas Distribuidos

Definición de sistemas cliente/servidor (1)

- En la arquitectura cliente/servidor:
  - Los **clientes** (o programas que representan entidades que necesitan servicios) y los **servidores** (o programas que proporcionan servicios) son objetos separados desde un punto de vista lógico y que se comunican a través de una red de comunicaciones para realizar una o varias tareas de forma conjunta
  - Un **cliente** hace una petición de un servicio y recibe la respuesta a dicha petición; un **servidor** recibe y procesa la petición, y devuelve la respuesta solicitada
- Características de la arquitectura cliente/servidor
  - **Protocolos asimétricos**: hay una relación muchos a uno entre los clientes y un servidor. Los Clientes siempre inician un diálogo mediante la solicitud de un servicio. Los Servidores esperan pasivamente por las solicitudes de los clientes.
  - **Encapsulación de servicios**: El servidor es un especialista, cuando se le entrega un mensaje solicitando un servicio, él determina cómo conseguir hacer el trabajo. Los servidores se pueden actualizar sin afectar a los clientes en tanto que la interfaz pública de mensajes que se utilice por ambos lados, permanezca sin cambiar
  - **Integridad**: el código y los datos de un servidor se mantienen centralizados, lo que origina que el mantenimiento sea más barato y la protección de la integridad de datos compartidos. Al mismo tiempo, los clientes mantienen su independencia y "personalidad"

Definición de sistemas cliente/servidor (2)

- Características de la arquitectura cliente/servidor (continuación)
  - **Transparencia de localización**: el servidor es un proceso que puede residir en la misma máquina que el cliente o otra una máquina diferente de la red. El software cliente/servidor (*midleware*) habitualmente oculta la localización de un servidor a los clientes mediante la redirección de servicios. Un programa puede actuar tanto como cliente, como servidor o como cliente y servidor simultáneamente
  - **Intercambios basados en mensajes**: Los clientes y servidores son procesos débilmente acoplados que pueden intercambiar solicitudes de servicios y respuestas utilizando mensajes
  - **Modularidad**, diseño extensible: el diseño modular de una aplicación cliente/servidor permite que la aplicación sea tolerante a fallos
    - En sistemas tolerantes a fallos, los fallos pueden ocurrir sin causar la caída de la aplicación completa
    - En una aplicación cliente/servidor tolerante a fallos, uno o más servidores pueden fallar sin parar el sistema total mientras que los servicios proporcionados por los servidores caídos estén disponibles en otros servidores activos
    - Otra ventaja de la modularidad es que una aplicación cliente/servidor puede responder automáticamente al incremento o decremento de la carga del sistema mediante la incorporación o eliminación de uno o más servicios o servidores

Definición de sistemas cliente/servidor (3)

- Características de la arquitectura cliente/servidor (continuación)
  - **Independencia de la plataforma**: el software cliente/servidor "ideal" es independiente del hardware o sistemas operativos, permitiendo al programador mezclar plataformas de clientes y servidores
    - El entorno de explotación de clientes y servidores puede ser sobre diferentes plataformas, con el fin de optimizar el tipo de trabajo que cada uno desempeña
  - **Código reutilizable**: La implementación de un servicio puede utilizarse en varios servidores
  - **Escalabilidad**: Los sistemas cliente/servidor pueden ser escalados horizontal o verticalmente
    - El escalado horizontal significa añadir o eliminar estaciones clientes con un ligero impacto en el rendimiento
    - El escalado vertical significa la migración a una máquina servidora más grande y rápida o la incorporación de nuevas máquinas servidoras
  - **Separación de la funcionalidad del cliente/servidor**: El modelo cliente/servidor es una relación entre procesos que se ejecutan en la misma o en máquinas separadas. Un proceso servidor es un proveedor de servicios. Un cliente es un consumidor de servicios. El modelo cliente servidor proporciona una clara separación de funciones
  - **Recursos compartidos**: un servidor puede proporcionar servicios a muchos clientes al mismo tiempo, y regular el acceso de éstos a un conjunto de recursos compartidos

Tecnología cliente/servidor(1)

- Servidores de ficheros: los clientes hacen solicitudes de ficheros al servidor: forma de compartir ficheros en una red (repositorios de documentos, imágenes, programas, etc.)
- Servidores de bases de datos: aplicaciones del cliente mandan solicitudes SQL al servidor. El servidor devuelve el resultado de la consulta.
- Servidores de transacciones: el cliente invoca procedimientos remotos o transacciones (conjunto de instrucciones SQL) sobre la base de datos. Los datos intercambiados son:
  - Cliente -> servidor: solicitud
  - Servidor -> cliente: mensaje de resultado
- Servidores groupware: intercambio de información semiestructurada: texto, imágenes, u otros (Lotus Notes o Microsoft Exchange). Cada vez más se usa e-mail

Tecnología cliente/servidor(2)

- Servidores de aplicaciones de objetos:
  Aplicación cliente/servidor: conjunto de objetos de comunicación. Los objetos del cliente usan un Object Request Broker (ORB). El cliente invoca un método remoto, el ORB localiza una instancia de la clase del objeto en el servidor, invoca el método y devuelve el resultado al objeto del cliente.
  CORBA (Common Object Request Broker Arquitecture)

- Servidores de aplicaciones web:
  World Wide Web: arquitectura cliente/servidor (los clientes solicitan documentos a los servidores). La solicitud es por nombre y el protocolo es HTTP
  Hay objetos web y toda clase de aplicaciones nuevas

Evolución sistemas Cliente/Servidor (1)

- El término Cliente/Servidor se ha asociado tradicionalmente con un PC de escritorio conectado a través de una red a algún tipo de servidor de base de datos
  - De hecho, el término Cliente/Servidor se refiere formalmente a un modelo lógico que proporciona una división de tareas dentro de las capas (o niveles) "cliente" y "servidor"
- Arquitecturas (C/S) monolíticas (una capa)
  - La industria de la Tecnología de la Información ha puesto en práctica una forma muy sencilla de computación cliente/servidor desde la aparición inicial de los mainframe. En esa configuración un host mainframe y un terminal "tonto" directamente conectado con el mainframe puede verse como un modelo C/S de una capa
- Arquitecturas C/S de dos capas
  - El cliente se comunica directamente con un servidor de bases de datos
  - La aplicación o lógica de negocio bien reside en el cliente, o en el servidor de base de datos en la forma de procedimientos almacenados
  - Un primer modelo C/S de dos capas comenzó a emerger con las aplicaciones desarrolladas para redes LAN a finales de los 80 y principios de los 90
    - Estas aplicaciones se basaban en técnicas sencillas de compartición de archivos, implementadas mediante lenguajes del tipo Xbase (Xbase se refiere genéricamente a los lenguajes derivados a partir del lenguaje de dBase: dBase, FoxPro, Clipper, Paradox, etc.)

Evolución sistemas Cliente/Servidor (2)

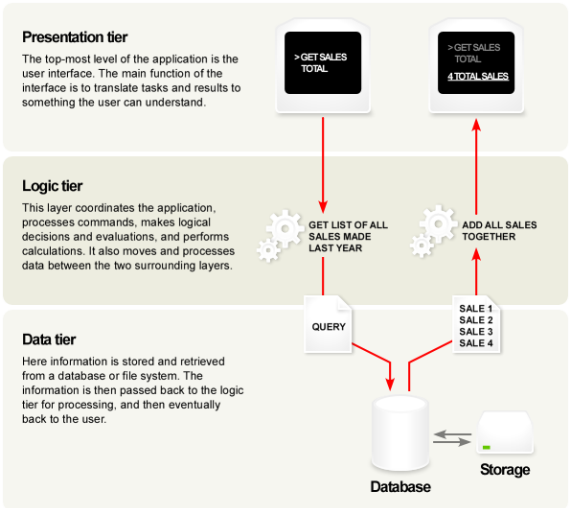
- Arquitecturas C/S de dos capas (continuación)
  - **Cliente grueso**
    - Inicialmente, en el modelo de dos capas intervienen equipos que no tienen la característica de mainframe (un servidor de archivos en red) y un cliente "grueso" inteligente, donde se hace la mayor parte del procesamiento
      - Esta configuración no es fácilmente escalable en sistemas de gran, e incluso medio, tamaño (50 o más clientes conectados)
    - Entonces el Interfaz Gráfico de Usuario (GUI, *Graphical User Interface*) emerge como el entorno dominante para las aplicaciones de escritorio y con él, emerge un nuevo enfoque en el planteamiento inicial de la arquitectura de dos capas
      - El servidor de ficheros en red de propósito general se reemplaza por un servidor de bases de datos especializado
      - Esto modelo origina la aparición de nuevas herramientas de desarrollo: PowerBuilder, VisualBasic y Delphi, por citar algunas
    - La mayor parte del procesamiento tiene lugar aún en los clientes "gruesos", pero ahora la información se hace llegar al cliente utilizando un Lenguaje Estructurado de Consulta (SQL, *Structured Query Language*) para realizar peticiones al servidor de base de datos, que simplemente informa del resultado de las consultas
    - Cuanto más complicada la aplicación, más "grueso" pasa a ser el cliente y más potente debe ser el hardware que debe soportarlo
      - El coste de adecuar la tecnología del cliente pasa a ser prohibitivo y puede fustrar la abordabilidad de las aplicaciones
    - Además, la carga de la red utilizando este tipo de clientes es muy grande, de modo que el ancho efectivo de la red (y por lo tanto del número de usuarios que pueden utilizarla) se reduce

Evolución sistemas Cliente/Servidor (3)

- Arquitecturas C/S de dos capas (continuación)
  - **Servidor grueso**
    - Una configuración alternativa Cliente "fino" <--> Servidor "grueso" es otra aproximación utilizada en la arquitectura de dos capas
      - En este caso el cliente invoca procedimientos almacenados en el servidor de base de datos
      - El modelo del Servidor "grueso" tiene un mejor rendimiento "grueso" porque aunque la carga de red es todavía pesada, es más ligera que en la aproximación del Cliente "grueso"
    - El inconveniente de esta aproximación es que el uso de procedimientos almacenados hace depender el desarrollo excesivamente del software del vendedor
    - Otro inconveniente se deriva del hecho de que los procedimientos están almacenados conjuntamente con los datos y cada base de datos que contiene el procedimiento debe modificarse cuando cambia la lógica de la aplicación
      - En grandes bases de datos distribuidas esto puede conducir a una administración dificultosa
  - En ambos caso, se utiliza un protocolo de transporte de bases de datos (como SQL-net) para llevar las transacciones de un extremo a otro, que generalmente resulta ser un proceso "pesado"
  - No importa que modelo particular se utilice, los sistemas de dos capas no se ajustan bien cuando se manejan aproximadamente 100 usuarios

Evolución sistemas Cliente/Servidor (4)

- Arquitecturas C/S de tres capas
  - Una generación más novel de la arquitectura C/S añade una capa intermedia (*middle tier*)
  - En la **arquitectura de tres capas** (en general, en la arquitectura multicapa) el cliente implementa la lógica de presentación (cliente "fino"), el servidor(es) de aplicación implementan la lógica de negocio y los datos residen en uno (o varios) servidor(es) de bases de datos
  - Una arquitectura multicapa se define por tanto por las siguientes tres capas de componentes:



  - Un componente *front-end* que es el responsable de proporcionar la lógica de presentación
  - Un componente *back-end* que proporciona acceso a servicios dedicados, tales como un servidor de bases de datos
  - Un componente que hace las funciones de capa intermediaria (*middle-tier*) que permite a los usuarios compartir y controlar la lógica de negocio mediante su aislamiento de la aplicación real

Evolución sistemas Cliente/Servidor (5)

- Arquitecturas C/S de tres capas (continuación)
  - Una arquitectura multicapa aumenta la arquitectura C/S tradicional mediante la introducción de una o más componentes intermedios
    - El sistema cliente interactúa con la capa intermedia vía un protocolo estándar como HTTP o RPC
    - La capa intermedia interactúa con el servidor de datos (*back-end*) mediante protocolos de bases de datos estándar tales como SQL, ODBC y JDBC
  - Esta capa intermedia contiene la mayor parte de la lógica de la aplicación, traduciendo las llamadas del cliente en consultas (u otras acciones) a la base de datos y traduciendo los datos provenientes de la base de datos en datos del cliente para devolvérselos
  - Este emplazamiento de la lógica de negocio sobre el servidor de aplicaciones proporciona escalabilidad y aislamiento de la lógica de negocio con el fin de manejar rápidamente los cambios necesarios de ésta
    - Además, este hecho permite ampliar las opciones en lo que se refiere a la elección de un software propietario de bases de datos
  - La arquitectura de 3 capas se puede extender a n capas cuando la capa intermedia soporta conexiones a diferentes tipos de servicios (no sólo servicios de almacenamiento de datos), integrándolos y acoplándolos al cliente y entre ellos

Evolución sistemas Cliente/Servidor (6)

- Arquitecturas C/S de tres capas (continuación)
  - Otras ventajas de la arquitectura C/S multicapa son:
    - Cambios en la interfaz de usuario o en la lógica de la aplicación son muy independientes entre sí, permitiendo a la aplicación evolucionar fácilmente para satisfacer los nuevos requisitos
    - Los cuellos de botella de la red de comunicaciones se minimizan porque la capa de aplicación no transmite datos extras al cliente, sólo lo que necesite para llevar a cabo la tarea
    - Cuando se requieren cambios en la lógica de negocio, sólo debe actualizarse el servidor. En la arquitectura de dos capas, cada cliente debe ser modificado cuando cambia la lógica
    - El cliente está aislado de la base de datos y las operaciones de red. El cliente puede acceder fácil y rápidamente sin saber dónde están los datos o cuántos servidores se están utilizando
    - Las conexiones de bases de datos se pueden agrupar y, por tanto, compartidas por varios usuarios, lo que reduce considerablemente el coste asociado a las licencias por usuario
    - La organización es independiente de la base de datos, porque la capa de datos se escribe utilizando SQL estándar que es independiente de la plataforma
    - La lógica de la aplicación se puede utilizar un lenguaje estándar como Java, C o COBOL

Configuraciones cliente/servidor típicas

Tres bloques básicos en una arquitectura cliente/servidor:
cliente, servidor, middleware

Ejemplos de configuraciones:

- Arquitecturas cliente/servidor en la misma máquina. Ej: sistema de gestión de una consulta de un médico con un solo ordenador => alta escalabilidad
- Arquitecturas cliente/servidor con servidor único. Ej: sistemas basados en LAN con un servidor y varios clientes (terminales)
- Arquitecturas cliente/servidor con varios servidores.
  - Varios servidores con funciones distintas
  - Duplicación de servidores para robustez frente a fallos o para aumento de rendimiento más fácil y flexible
- Arquitecturas cliente/servidor en la cual cada máquina que es un cliente y un servidor completo

Elementos de arquitecturas cliente/servidor

- Tres bloques básicos en una arquitectura cliente/servidor:
  - Cliente: incluye sistema operativo (OS) sobre con interfaz gráfico de usuario (GUI) o interfaz orientado a objetos de usuario (OOUI)
  - Servidor: ejecuta software especializado
  - Middleware: software distribuido para interacciones entre cliente y servidor
    Desde la API del cliente usada para invocar el servicio, la transmisión de la solicitud y la respuesta hasta el sistema que informa al servidor
    No incluye el software que proporciona el servicio ni el interfaz de usuario en el cliente. Parte en el cliente y parte en el servidor
    El middleware incluye:
    - Protocolos de transporte, como TCP/IP, IPX...
    - NOS's (Sistemas operativos de red), como RPC, Samba...
    - Middleware específico para el servicio como HTTP, ORB...
    - Responsable del buen funcionamiento, especialmente en N niveles

Características del servidor: funciones del servidor

- Esperar peticiones de clientes (mensajes). A veces sesión por cliente y otras conjunto dinámico de sesiones
- Atender solicitudes simultáneas => concurrencia. Sin riesgo para la integridad de los recursos compartidos
- Prioridades en la atención de las solicitudes
- Capacidad de lanzar tareas en segundo plano no relacionadas con el servicio Ejemplo: un servidor de ftp aprovecha las horas de la noche para actualizar un mirror
- Robustez: crítica en servidores
- Escalabilidad y extensibilidad

Características del servidor: requisitos del SO del servidor (1)

Distinguimos en un SO servicios básicos ("de serie") y servicios extendidos:

SERVICIOS BÁSICOS:

- Alto nivel de concurrencia (tanto de tareas como en cada tarea)
- *Task preemption*: fin de las tareas "voluntario" => peligro. Mejor slots de tamaño fijo
- Prioridades
- Mecanismos de concurrencia (semáforos, monitores)
- Mecanismos de comunicación entre procesos. Redireccionamiento transparente
- Threads
- Sistema de ficheros multiusuario de altas prestaciones: muchos ficheros abiertos simultáneamente y protección de integridad
- Sistema eficaz de gestión de memoria: manipulación de objetos y programas grandes. Sistema de intercambio ágil con el disco
- Extensibilidad sin recompilar o (idealmente) rearrancar

Características del servidor: requisitos del SO del servidor (2)

SERVICIOS EXTENDIDOS:

- Soporte para distintos protocolos de comunicación => servicio a clientes distintos
- Extensiones para acceso transparente a recursos compartidos (ficheros, impresoras...)
- Recursos de manipulación de BLOBs (Binary Large OBjects): imágenes, video, gráficos...
- Sistema de directorio global o páginas amarillas (localización de recursos por su nombre)
- Servicios de autenticación (un cliente es quien dice ser) y autorización (un cliente puede hacer lo que está haciendo)
- Gestión del sistema: configuración, monitorización, generación de alertas, distribución y manipulación de paquetes de software para los clientes, identificación de virus o intrusos...
- Sincronización temporal entre clientes y servidor
- Servicios de bases de datos y de transacciones
- Servicios de internet: HTTP, SSL, firewalls, DNS...
- Servicios orientados a objetos

Características del servidor: evolución de los servidores (1)

Segmento muy heterogéneo:

- desde servidores simples de impresora
- hasta servidores de clusters para procesamiento masivo

El segmento más importante (comercialmente): servidores de aplicaciones (web, bases de datos, objetos, groupware...)

Por segmentos:

- Segmento bajo y medio: NetWare, Microsoft y Unixes (Solaris, Linux, FreeBSD)
- Segmento alto: casi exclusivamente Unix

Características del servidor: evolución de los servidores (2)

CARACTERÍSTICAS:

- NetWare:
  - Buen servidor de ficheros; mal servidor de aplicaciones
  - Soporta clientes de Windows, Mac y Linux
  - Incorpora LDAP, una máquina virtual Java, CORBA, etc.
- Microsoft:
  - Servidor de aplicaciones, de ficheros e impresora y de bases de datos
  - Bien con clientes Windows e incorpora herramientas de Microsoft
  - Problemas de Microsoft y no es fácilmente escalable
  - Mal para multiprocesador
- Unix:
  - A bajo nivel, linux+apache predomina en Internet
  - A nivel medio, Unix y Linux
  - A alto nivel Unix

Características del cliente

Dividimos los clientes en tres tipos:

- Sin GUI (Interfaz gráfico de usuario):
  - Lectores de códigos de barras, demonios...
- Con GUI:
  - Sustituyeron a los terminales sin gráficos
  - Normalmente usan el modelo objeto/acción: selección de objetos y acciones para realizar sobre éstos.
  - Normalmente los diálogos de naturaleza secuencial
  - Ejemplos: SOs antiguos o las páginas web con formularios
- Con OOUI (Interfaz de usuario orientada a objetos):
  - El usuario manipula de forma objetos en pantalla (*drag-and-drop*)
- Diferencias entre GUI y OOUI:
  - Los OOUI en realidad extensiones del interfaz del sistema operativo => no es fácil decir donde acaba la aplicación y empieza el SO
  - GUI: icono=aplicación
  - OOUI: icono=objeto
- Ejemplos:
  - GUI: Windows 3.X, Motif y páginas web sencillas
  - OOUI: MacOs, Windows 32 bits, Gnome, KDE y páginas web que utilizan Java 2 JavaBeans.

Características del cliente: requisitos del SO del cliente

- Todos necesitan mecanismo para implementar el mecanismo de solicitud/respuesta (evidente)
- Todos necesitan algún tipo de transferencia de ficheros (intercambio de imágenes, texto...)
- Facilidades multitarea (prioridades, preasignación temporal de tareas, comunicación entre procesos, threads)
  - Imprescindible en clientes sin GUI con multitarea en el servicio y clientes con OOUI
  - Para GUI simples viene bien
  - Para clientes sin GUI y sin multitarea no es necesario
- Portabilidad de código: máquina virtual Java en los clientes
- Robustez: el servidor no controla los clientes => evitar que un proceso de cliente dé problemas en el servicio

Características del cliente: evolución de los clientes

Sector en cambio vertiginoso

Tendencias en la evolución de los clientes:

- SO de los clientes cada vez menos monopolizado: hace años sólo Windows 3.X y el DOS. Ahora, Windows distintos, MacOS X, Linux, PalmOs...
  - Cliente universal: navegador de internet
  - Diversificación de PCs: PC's supergruesos y PC's superfinos
  - Cada vez más clientes incrustados en dispositivos portátiles

SOs más habituales en los clientes:

- Windows:
  - Ventajas: interfaz muy conocido; soporta distintos protocolos como TCP/IP, NPX/SPX, PPP...
  - Desventajas: coste, seguridad y acaparador de recursos
- Mac OS X:

  Presencia en Internet muy superior a su presencia en ordenadores personales
  - Ventajas: entorno gráfico magnífico; basado en FreeBSD
  - Desventajas: coste, software, y mal soporte a hardware de otros fabricantes
- Linux:
  - Ventajas: fiabilidad, prestaciones, gratuidad, soporte sobre distintos sistemas
  - Desventajas: instalación, Office, un buen emulador de windows gratuito.

El middleware: objetivos del middleware

Función del middleware: que todo funcione con trasparencia
El sistema da la impresión de ser único
Internet = sistema único de millones de usuarios

Tipos de transparencia:

- De localización: innecesario saber la localización de un recurso \\Máquina\directorio\fichero viola la transparencia de localización
- De nombres: mismo espacio de nombres en toda la red
- De acceso: sistema de acceso único
- De replicación: trabajar con recursos duplicados como únicos. Ej: mantener las copias y sincronizar una base de datos replicada
- De acceso local o remoto: acceder a distancia igual que localmente
- Temporal: mantenimiento de relojes de todo el sistema
- De fallos: el NOS debe controlar reintentos y recuperaciones
- De administración: interfaz de administración única y consistente

El middleware: funcionalidades del middleware

El middleware debe ofrecer las siguientes funcionalidades:

- Sistema de ficheros distribuido
- Servicio de directorio global
- Servicio de tiempo distribuido
- Mecanismos de seguridad
- Sistemas de comunicaciones a través de la red:
  - Un sistema de comunicaciones punto a punto
  - Un sistema de invocación remota de procedimientos
  - Un sistema de mensajería de red