



IN3501 - Tecnologías de Información y Comunicaciones
para la Gestión

CLIENTE SERVIDOR DE MULTIPLES CAPAS

PROFESORES

Evelyn Andaur
Juan D. Velásquez
Gastón L'Huillier
Víctor Rebolledo Lorca

Temario

- Introducción.
- Redes, Internet y Web.
- **Cliente Servidor de Múltiples Capas.**
- La capa de datos.
- La capa de negocios.
- La capa de presentación.
- Administración de proyectos informáticos.



Capitulo III



CLIENTE SERVIDOR DE MÚLTIPLES CAPAS

Temario

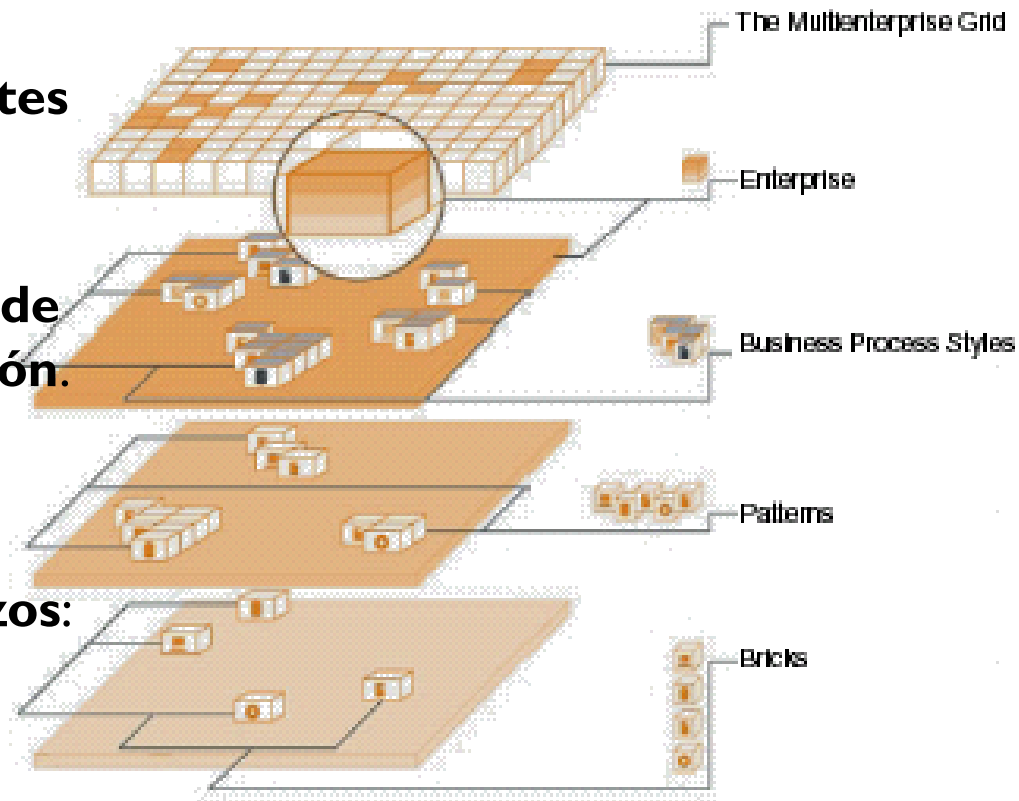
- Arquitectura de Sistemas
- Esquema de mainframe
- Cliente servidor de dos capas
- Cliente servidor multicapa y middleware
- La solución para sitios web muy transaccionales.
- Consideraciones de diseño
- Aplicaciones Web
- Arquitecturas Modernas

¿Qué es una Arquitectura?

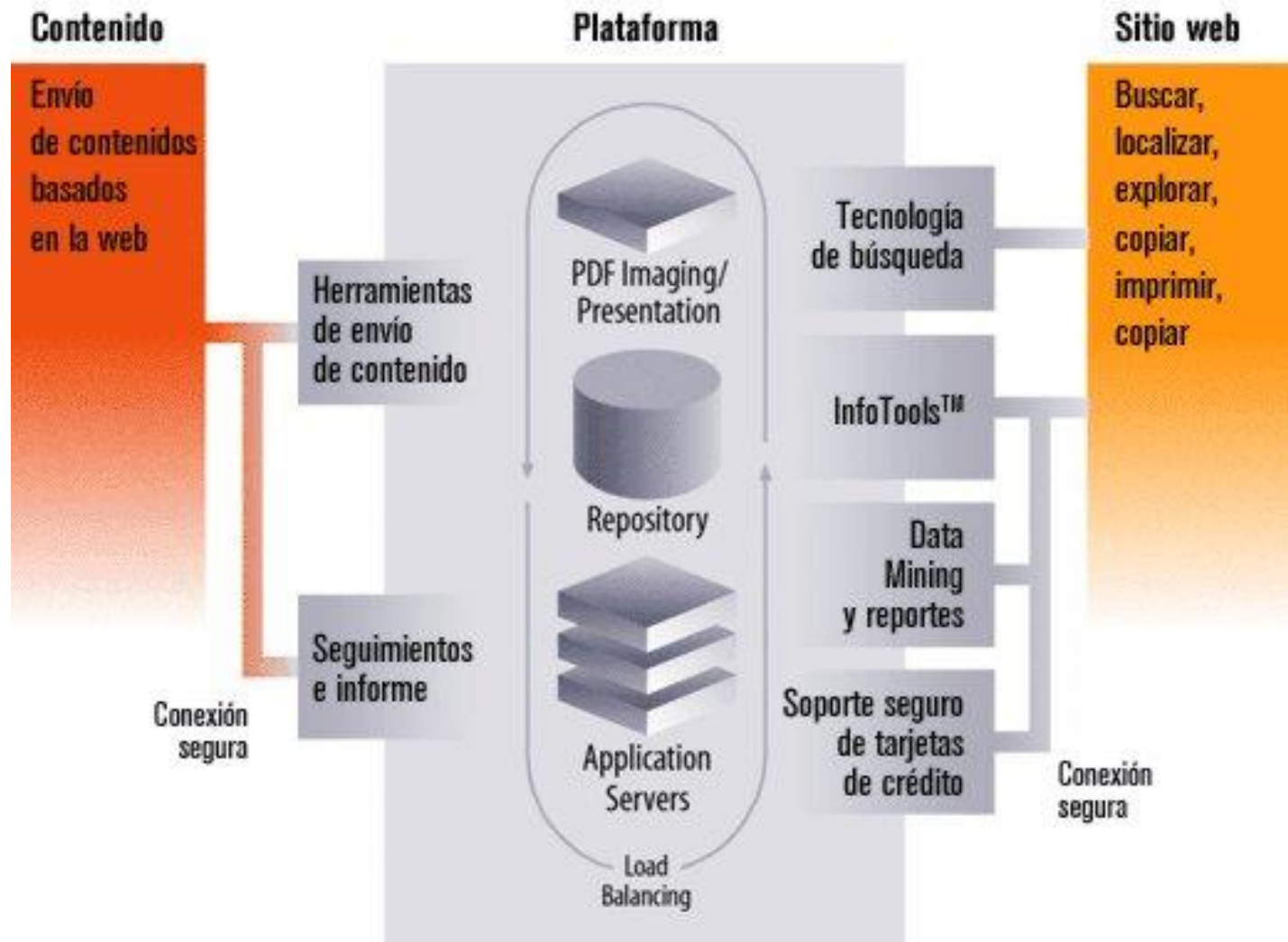


¿Qué es una Arquitectura?

- Modelo que contiene un conjunto de **Componentes** y sus **relaciones**.
- Define el **marco** para el **diseño de y desarrollo de Sistemas de Información**.
- Plano Regulador.
- Guías de **Diseño**.
- Permite **enfocar esfuerzos**:
 - *Abstracción.*



¿Qué es una Arquitectura?



Beneficios de contar con una Arquitectura Tecnológica



- Documenta los **objetivos** de su **organización**, los **procesos de negocio** y **identifica recursos asociados**.
- **Identifica recursos de IT con objetivos organizacionales.**
- Es crucial en la **interoperabilidad** de los sistemas.
- **Evitar duplicidad** de funciones.
- Permite el desarrollo de **servicios comunes**.

Arquitectura de Sistemas:

“Procesamiento y Red”

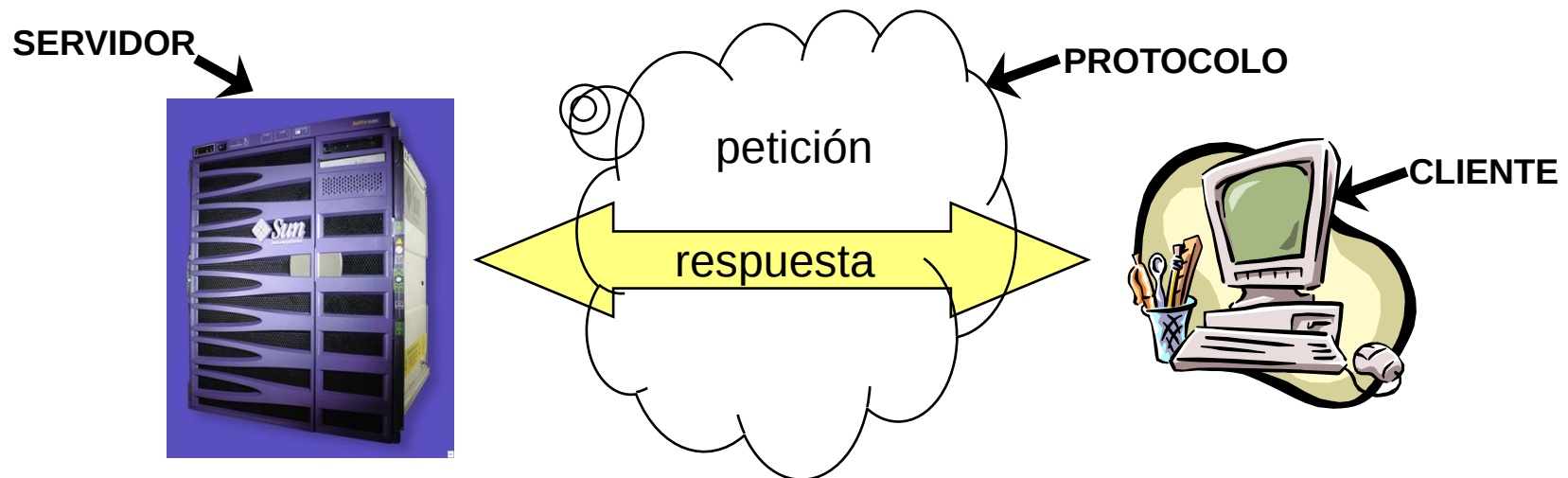
- El diseño de la arquitectura de un sistema consiste en **planes para el hardware, software, comunicación y seguridad** para apoyar la implantación de una aplicación o sistema.
- Es de particular importancia definir, donde ocurren los **procesos** y cual es la forma de **distribuir** los datos entre los **centros de procesamiento**.
- El procesamiento puede efectuarse en:
 - En un “**Cluster Linux**”
 - En un “**servidor**”
 - En un **computador personal**
 - En una **Palm con red inalámbrica**, otros ...

Definiciones

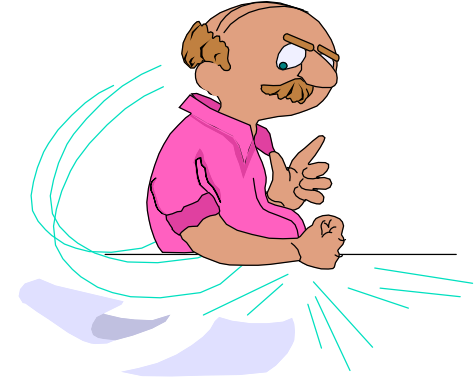
- El “**modelo de red**” muestra las principales *componentes* del sistema, donde están ubicadas y cómo se conectan unas con otras.
- La **especificación de hardware y software** describen estas componentes en detalle y apoya la compra y adquisición de productos para la implementación.

Unidad de procesamiento: ¿Que es un Servidor?

- Es un **programa** cuyo input son peticiones recibidas por la red, y cuyo output es enviado por la red al **cliente** que envió la petición.
- Usualmente se le *asocia a la máquina física*, pero *una máquina por si sola no hace nada...*
- Las peticiones/respuestas están formalizadas por el llamado **Protocolo de transmisión**. El lenguaje de la transmisión y los pasos que se deben cumplir para enviar/recibir los datos.



Exclamación Típica: Se cayó el servidor!



- Un servidor no es una aplicación simple.
- En su estructura interna es muy similar a un **Sistema Operativo**.
- Ya conocen cuánto tiempo ha costado que los SO se estabilicen.
- Entre los mas estables:
 - el servidor web APACHE (<http://www.apache.org>)
 - la base de datos ORACLE (<http://www.oracle.com>)
- **Razones Típicas:**
 - Muchos clientes conectados.
 - Falta de memoria.
 - Errores del programa.
 - Problemas de concurrencia.



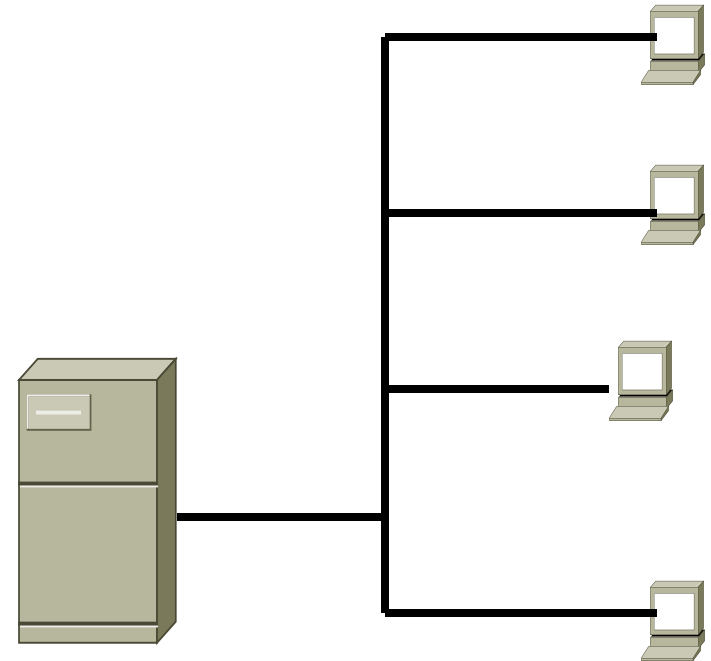
Arquitecturas Clásicas

- Si consideramos separar la lógica del procesamiento de los datos en **componentes** según:
 - *Almacenamiento de Datos* (archivos y prog que accede)
 - *Lógica de Acceso a los Datos* (extraer los datos)
 - *Lógica de la Aplicación* (negocio)
 - *Lógica de Presentación* (interfaces)
- Podemos tener una **variedad de arquitecturas** según como **distribuyamos los procesos** anteriores en **CAPAS**.



Esquema de mainframe

- Se trata de un **computador** que posee toda la *lógica del negocio* y los *datos centralizados*.
- Se accede a través de un **terminal “tonto”** (no procesa)
- Este esquema **quedó obsoleto**, pero hoy en día **ha vuelto a la vida** con la introducción de los **Network Computer**



Una Capa:

Modelo Mainframe

- **Una capa es la encapsulación de lógica y de procesamiento**
 - **Centraliza** el procesamiento en un servidor llamado **Host**, con el que varios usuarios interactuaban a través de dispositivos como *terminales, lectoras de tarjetas e impresoras*.
 - En el Host **se almacenan los datos y se realiza todo el procesamiento:**
 - Recibir el requerimiento del usuario
 - Actualizar y rescatar información de las bases de datos,
 - Realizar el procesamiento necesario
 - Entregar una respuesta al usuario y realizar la gestión de los recursos compartidos.

Una Capa:

Modelo Mainframe

- **Ventajas:**

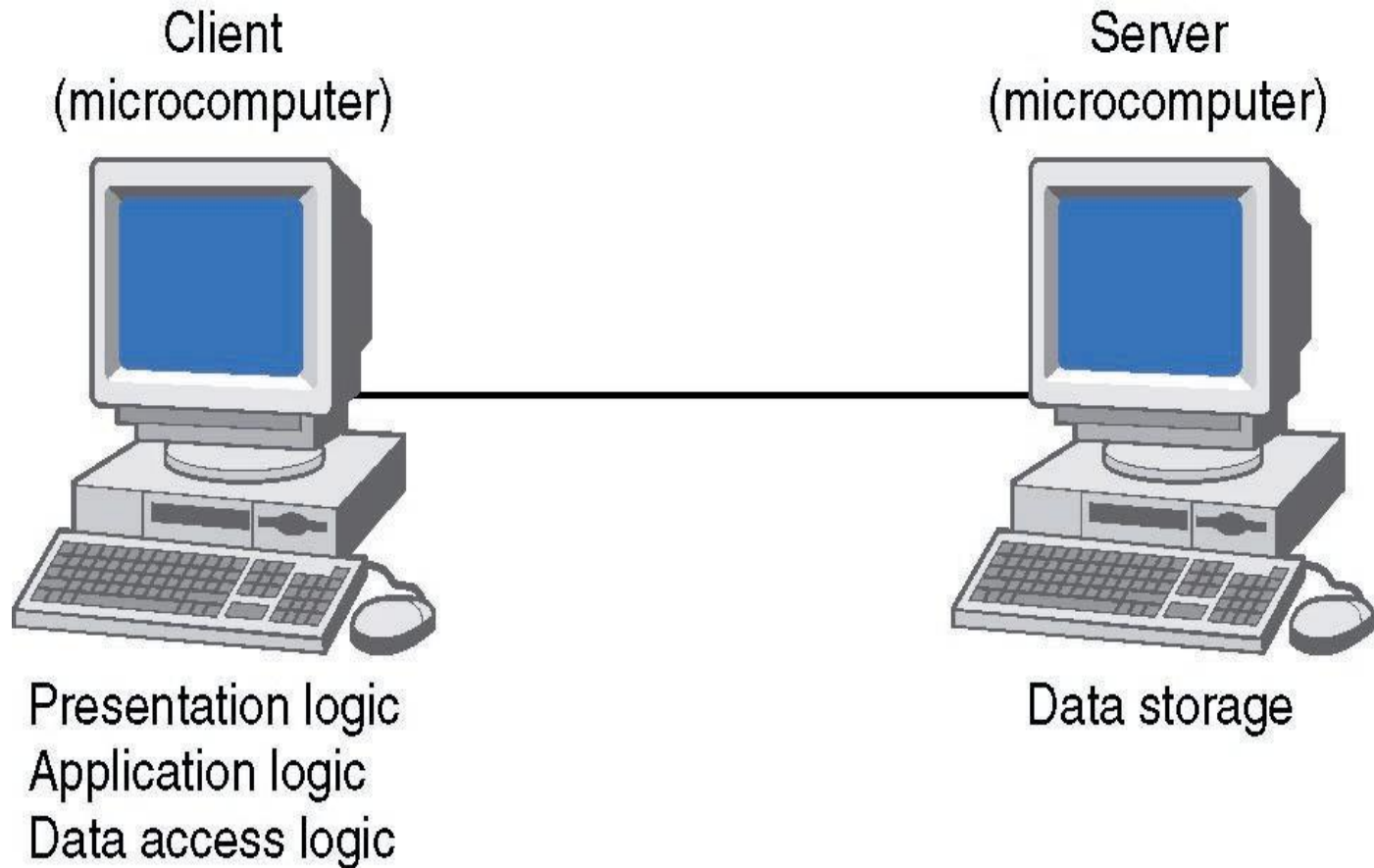
- Punto de **control centralizado**.
- **Actualizar** cambios en la lógica del negocio es **fácil** (es 1 sólo lugar).

- **Desventajas:**

- **Sobrecarga** del servidor (más CPU por tareas)
- **Upgrade** del servidor es **caro** (tiempo de no-uso)
- Se requiere **más disco duro** (Programas + Data)
- **Más Riesgoso** ante una Falla (todos los huevos en una misma canasta).

Dos Capas:

Arquitectura basada en el cliente.



Dos Capas:

Arquitectura basada en el cliente.

- La **capa de datos** se mantiene en el servidor.
- La **capa de negocios** se comparte entre cliente y servidor.
- La **capa de presentación** está totalmente en el cliente.
- Típicamente:
 - Muchos pc's con **un programa** (Visual Basic) que accede a un servidor de archivos común en la empresa.

Dos Capas:

Arquitectura basada en el cliente.

Ventajas:

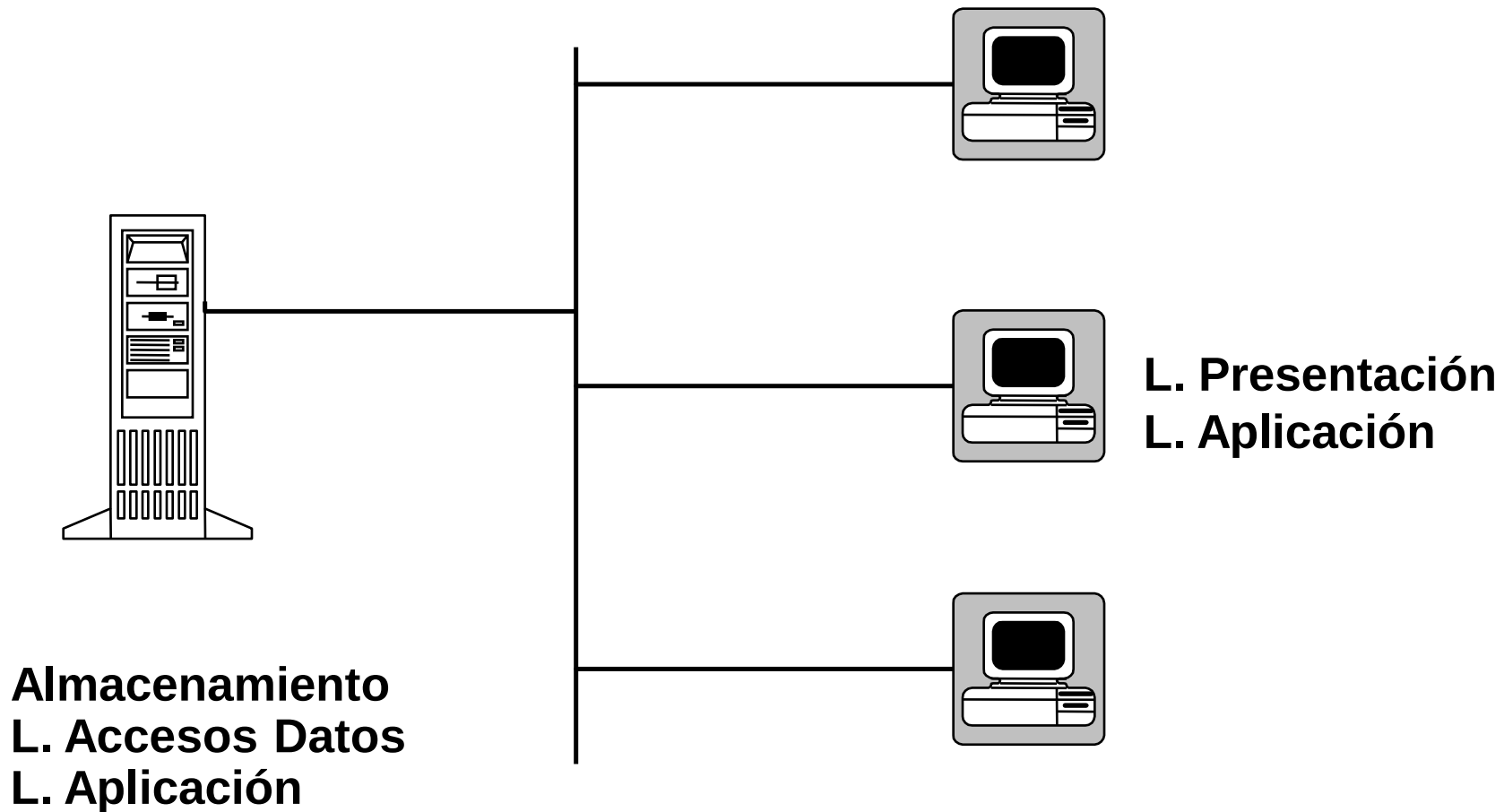
- Los **clientes** son **más baratos**
- Software
 - + **barato** de desarrollar que Mainframe
 - (db-server simplifica).
 - + **fácil** de usar.

Desventajas:

- **Sobrecarga de la red** por flujos de datos.
- **Costosos de mantener** en el tiempo si son *muchos clientes* (**¿el www podría funcionar así?**).
- **Acceso a la BD** también **se puede sobrecargar**.

Dos Capas:

Arquitectura Cliente Servidor (Tradicional).



Dos capas:

Cliente servidor tradicional

- Se distribuye la **carga de procesamiento** entre la máquina cliente y el servidor.
- En la máquina cliente se ejecuta una aplicación que se encarga de **validar y procesar el requerimiento** del usuario antes de ser enviado al servidor.
- El **servidor procesa** dicho requerimiento, realiza las actualizaciones necesarias y devuelve los resultados al cliente.
- Los **resultados son recibidos por la aplicación cliente** y presentados al usuario.

Dos capas:

Desventajas del Cliente/Servidor (I)

- **Distribución y Actualización de Aplicaciones**
 - Se deben *instalar porciones importantes de código en el Cliente*
 - Problemas con el Sistema Operativo
 - Manejo de Versiones (upgrade)
 - Las **actualizaciones** generalmente requieren desinstalar y reinstalar el programa

Dos capas:

Desventajas del Cliente/Servidor (2)

- **Capacitación de Usuarios**
 - *Distintas interfaces para las aplicaciones*
 - incluso para versiones del mismo programa (dependiente del proveedor)
 - *Dificulta estructurar las aplicaciones en una aplicación central*

Dos capas:

Desventajas del Cliente/Servidor (3)

- Evolución de las Aplicaciones
 - **Cambios** en las aplicaciones se *deben replicar en cada cliente*
 - Esto incluye **errores con la evolución normal de los programas**

Dos capas:

Desventajas del Cliente/Servidor (4)

- **Distribución de Carga del Sistema**
 - **No existe posibilidad de diferenciar los clientes** según *capacidad de proceso o ancho de banda*
 - **Programación en el Cliente incompatible** completamente con *la del Servidor*
 - Lenguajes
 - Sistemas operativos
 - Etc.

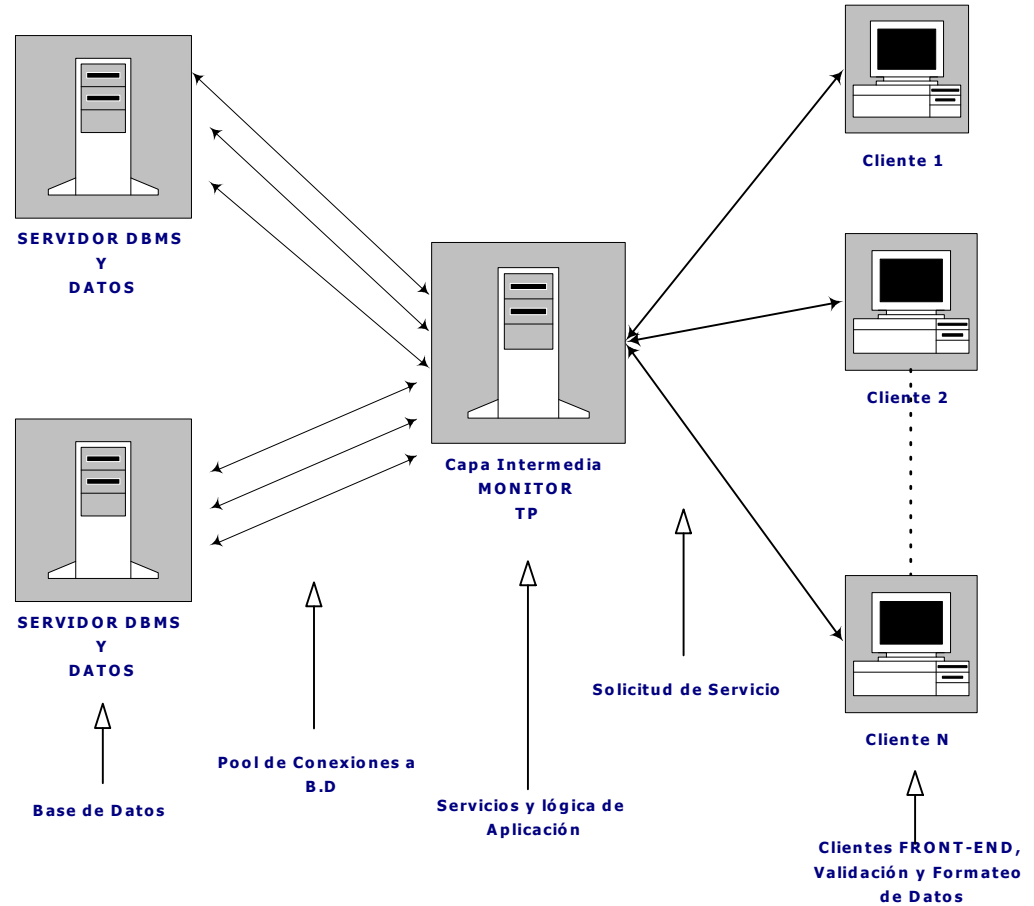
Dos capas:

Desventajas del Cliente/Servidor (5)

- **Disponibilidad de las Aplicaciones**
 - Se deben **instalar las aplicaciones en los clientes** para *acceder a los servicios*
 - *Usuarios remotos*
 - *Usuarios temporales*
 - **No se puede definir** con facilidad *niveles de acceso a las aplicaciones*

Arquitectura de Tres Capas

- Pretende dar solución a los problemas enunciados anteriormente
- ***Divide la aplicación en 3 tipos de servicios***
 - Visualización
 - Lógica
 - Datos



Capa de Presentación

- Involucra toda la **interacción con el usuario**.
Especifica operaciones como *repintar una ventana, capturar el clic del Mouse y realizar validaciones mínimas*.
- El modelo no permite la interacción del usuario con otras capas.
- *Esta capa no conoce sobre la tecnología de almacenamiento de datos y tiene un conjunto definido de interfaces que la habilitan para comunicarse con la capa “Lógica del Negocio” y generar transacciones de negocio.*

Capa de Aplicación (Negocio)

- Es la **capa media** entre el usuario y el almacenamiento físico de datos, donde se efectúa la mayoría del procesamiento.
- Esta capa no conoce los detalles específicos de la capa servicio de datos ni el tipo de “Presentación”.
- **Sólo puede procesar datos, no almacenarlos ni presentarlos.**
- Es en esta capa donde se resuelve toda la **lógica del negocio asociada a las transacciones** (reglas que definen las actividades que son factibles de realizar sobre los datos), y se realiza una conexión eficiente a la Base de Datos.

= *Encapsulación*

Capa de Datos

- Esta capa se encarga de cualquier **persistencia física** requerida por los datos de la aplicación. Aquí van los *mecanismos de servicios de datos específicos*.
- Aquí se definen los **motores de bases de datos** o **manejadores de archivo** que se usarán.

Ventajas

- Pueden convivir **distintos tipos de aplicaciones** en la capa cliente:
 - Aplicación Java, aplicación CORBA y clientes delgados HTML.
- **Aporta flexibilidad en la capa base de datos** porque permite el acceso a variados servicios de datos:
 - Distintos motores de bases de datos, con distintas tecnologías (*relacionales u orientados a objetos*), en distintos servidores.
- Además de la posibilidad de **utilizar al mismo tiempo más de un tipo de motor de base de datos**.

Ventajas (2)

- Utilizando la **programación multitareas** es posible *aumentar la cantidad de conexiones a los servidores de bases de datos.*
- Es posible **compartir y reutilizar el código de los servicios de la capa intermedia.**
- Por ejemplo, *un mismo servidor intermedio puede atender a diversas aplicaciones cliente.*
- Se pueden destinar los servidores de más memoria y capacidad de cálculo para el manejo de la capa de la aplicación.
- Y destinar los que poseen mayor capacidad de manipulación de discos para la administración de la base de datos.

Ventajas (3)

- La **carga de trabajo puede ser distribuida** en *varias aplicaciones o servidores de aplicación*.
- Es posible diseñar las aplicaciones trabajando **inicialmente con un único servidor**, para luego *desdoblarla en sus capas intermedias* cuando el crecimiento de los requerimientos del procesamiento así lo exijan.
- Ante un incremento en las exigencias del sistema, es posible *incorporar en la capa intermedia, máquinas de similares o diferentes características, logrando mayor eficiencia en la atención de requerimientos, gracias al* **procesamiento paralelo**.

Desventajas

- La arquitectura necesaria, **requiere de más equipamiento computacional.**
- También de **software específico.**
- Desarrollar un proyecto en tres capas **necesita de especialistas** que por lo general son **costosos.**
- El modelamiento de las tres capas **requiere de análisis mucho más profundos** que en el dos capas.

Tres capas

- **Lógica**

- **Visualización:**

- presentación de información y recopilación de datos

- **Lógica:**

- todos los algoritmos y procesos que componen el sistema

- **Datos:**

- la información que maneja el sistema

Tres capas (2)

- Físicamente se puede estructurar esta arquitectura según el **tipo de aplicación** y las **necesidades del sistema**
- Los parámetros principales para el análisis
 - *Capacidad de Proceso*
 - *Ancho de Banda*

Tres capas (3)

- **Capacidad de Proceso**

- Si no se tiene una estimación clara de la capacidad de proceso de los clientes, se **asume la mínima capacidad**
 - **Ejemplo:** Internet
- Si se tiene una noción más precisa de la capacidad de los clientes, se considera incorporar un **mayor nivel de lógica**
 - **Ejemplo:** Intranet

Tres capas (4)

- **Ancho de Banda**

- Si se tiene un ancho de **banda reducido** se trata de transmitir la *menor cantidad de información*.
 - Se privilegia el HTML
- Si se tiene un **ancho de banda más controlado**, se considera el *envío de cantidad de información mayores*, incluyendo *aplicaciones al cliente*

Tres capas (5)

- Se consideran 2 escenarios típicos
 - **Internet:**
 - Baja capacidad de proceso
 - Ancho de banda reducido
 - **Intranet / Extranet**
 - Capacidad de proceso más controlada
 - Mejor ancho de banda

La solución para sitios web muy transaccionales.

- **Asumamos que la Internet es lenta.**
- Si agregamos *lentitud en la respuesta* a la solicitud del usuario, el resultado es que, después de un tiempo, *nadie visite nuestro web site*.
- En **comercio electrónico**, lo anterior es **crítico**.
- *El problema no se resuelve aumentando la capacidad del servidor de datos.*
- Actualmente, la mayoría de los servidores de datos son **sistemas administradores de bases de datos relacionales** (Oracle, Informix, Sybase)

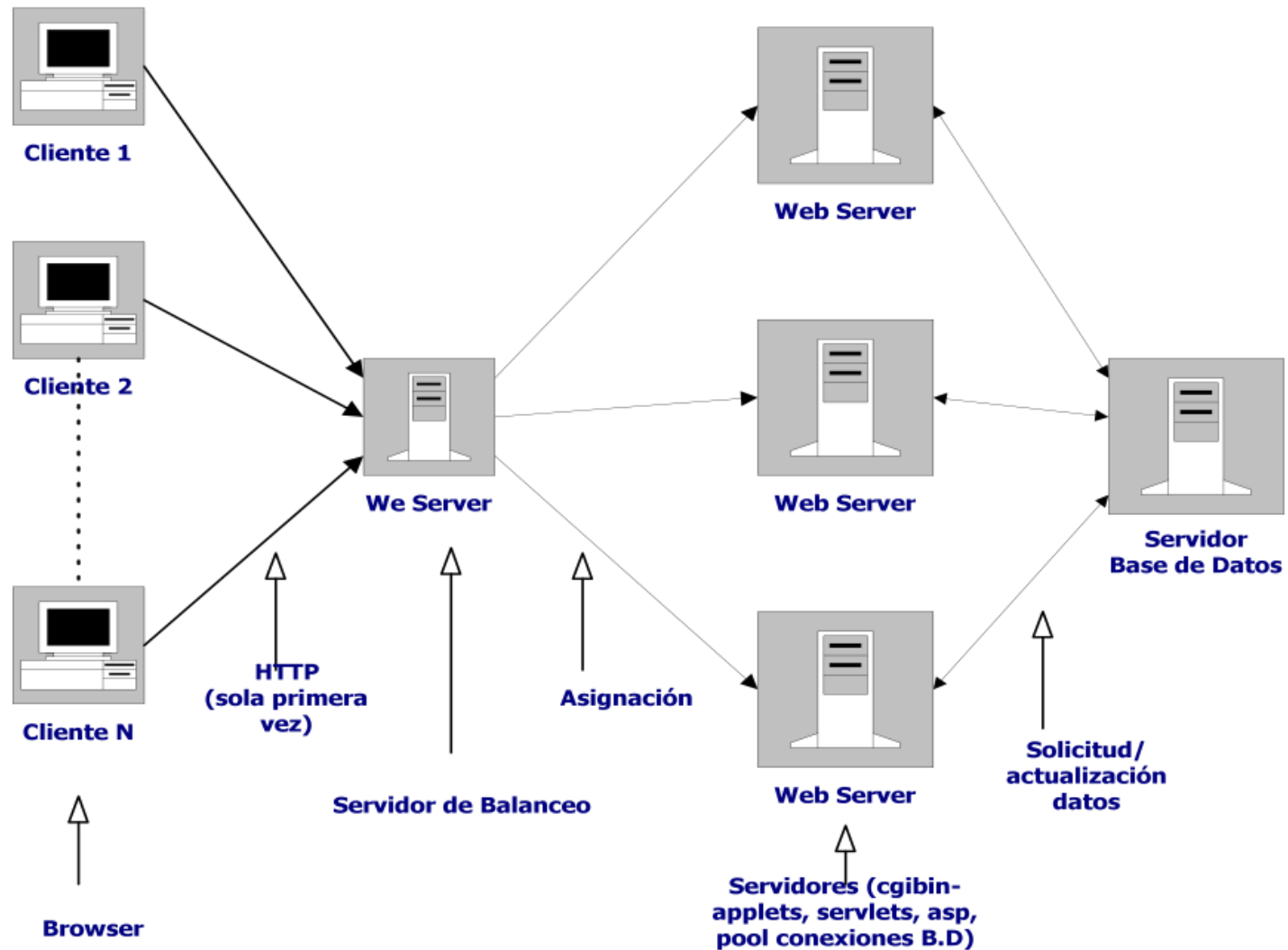
La solución para (2)

- La operación tradicional de un **Motor de datos** en C/S, indica que *por cada conexión a la base, se crea un proceso que atiende la solicitud.*
- ***La creación del proceso tiene su costo en tiempo y CPU.***
- *Si hay muchas conexiones, no hay Motor que lo soporte.*
- Un **requerimiento web**, tiene la característica de ser **muy rápido y específico**.
- Una conexión tradicional a BD, levanta *un proceso de comunicación y luego lo baja*, producto del tipo de transacción. Lo anterior es **muy costoso**.

La solución para (3)

- La solución sería **mantener un grupo de procesos** que *atienden a los usuarios web y que mantienen la comunicación constante con el motor de datos.*
- Lo anterior define una **arquitectura de tres capas.**
- Pero
 - *¿quién administra los procesos de la capa media?*
- Surge la idea de una **solución transaccional para la capa media**, la que depende de cuántas solicitudes recibirá el sitio

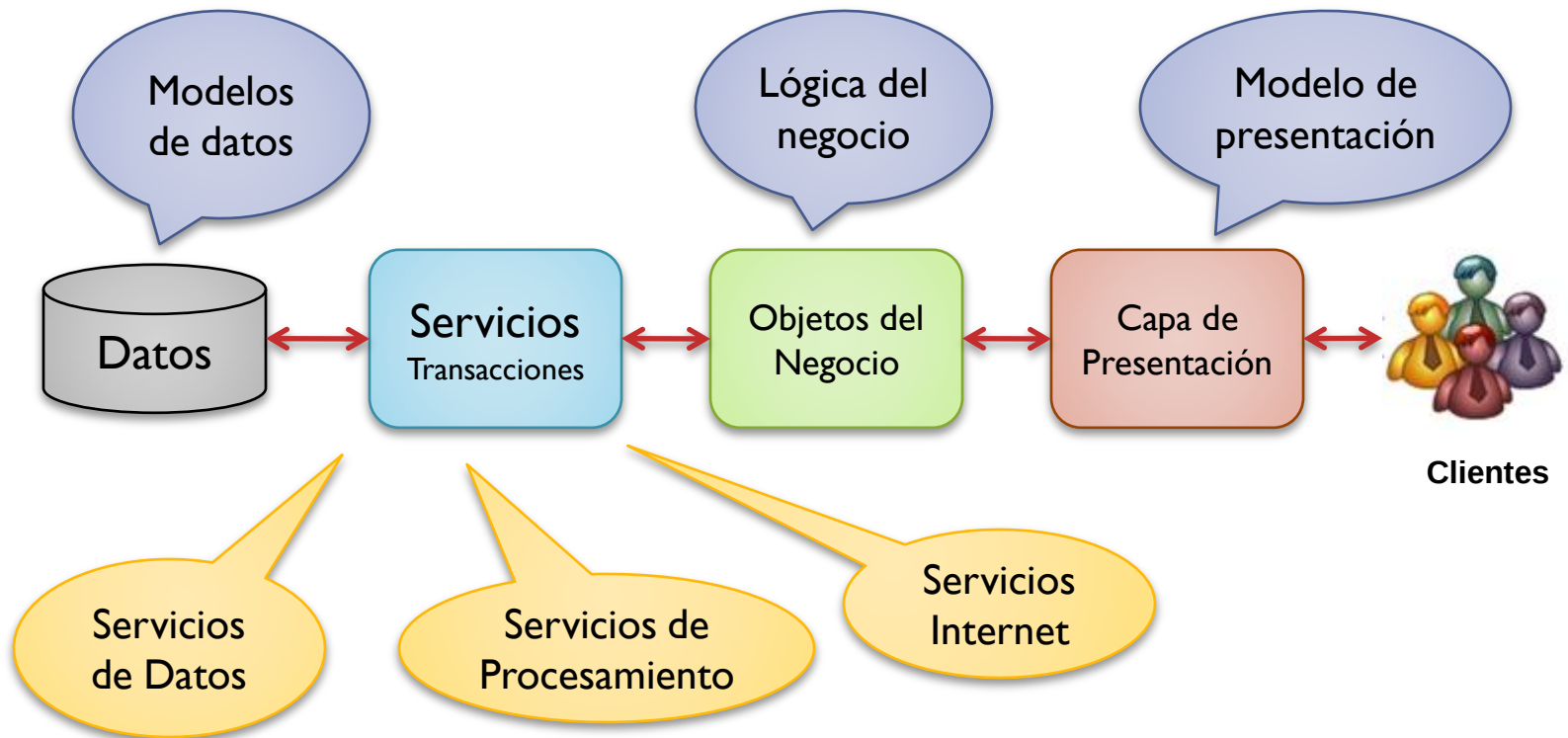
La solución para (4)



Consideraciones de diseño

- Los **sitios web muy transaccionales**, deben considerar **capas medias complejas**, incluso **servidores de balanceo**
- En general, *oriente el desarrollo a tres capas aunque físicamente sólo sean dos, en un futuro no lejano, puede necesitar la tercera capa.*
- Pese a toda la publicidad sobre **motores de bases de datos**, está claro que su **mejor trabajo es administrar datos y no la Web.**
- Oriente la capa media al uso de **monitores transaccionales**

Una capa lógico no es una capa física



Múltiples capas:

Dividir para reinar

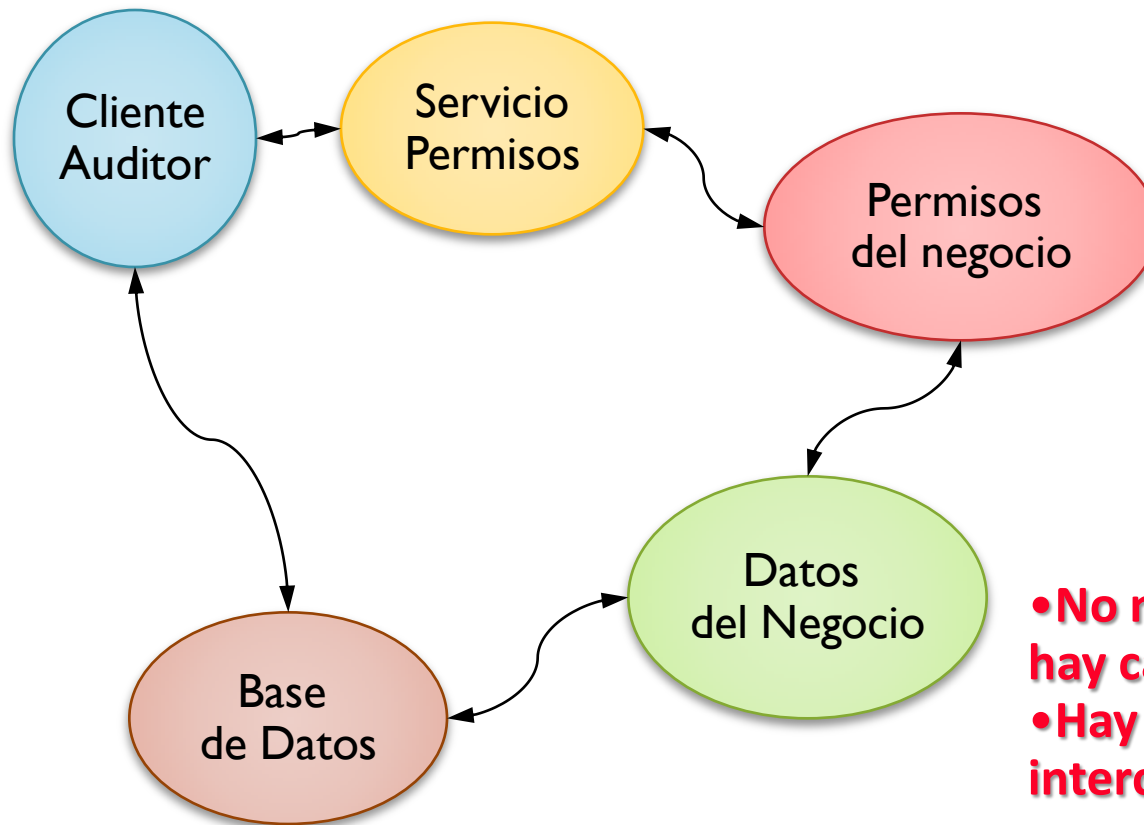
- Extendiendo la lógica en que se basa esta arquitectura, es posible dotar al modelo de tantas capas como sean convenientes de acuerdo a las soluciones requeridas, dando origen a **aplicaciones multi-capas**.
- La esencia del modelo de desarrollo de múltiples-capas, es dividir un sistema en **partes lógicas bien definidas**, que pueden ser diseñadas, construidas separadamente y procesar flexiblemente a través de **múltiples máquinas**.

Múltiples capas (2):

Cada capa es un objeto

- La idea es separar las diferentes partes del sistema, minimizar la interdependencia entre ellas (**MODULARIDAD**) y diseñarlas con un alto grado de **COHESION** en su funcionalidad interna.
- ***Mínimo acoplamiento y máxima cohesión.***
- Una capa particular no puede ver más que la interfaz pública expuesta de su capa adyacente. De este modo, **los cambios en una capa tienen un impacto mínimo sobre las otras**, facilitando la expansión y mejoras en el tiempo.

Múltiples Capas (3) : Sistema de “Objetos” en red

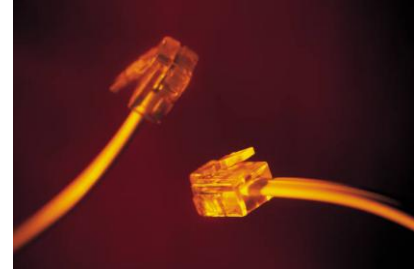


- **No necesariamente hay capas**
- **Hay necesidad de interconexión**

Nuevas Arquitecturas

- *Middleware*
 - muchas-muchas 'capas'
- *Peer to peer*
 - Gnutella, Napster, Bittorrent)
- *Seti@Home*
- *Mobile Agents*
 - programas que emigran
- *Grid Computing* (la matrix)

Middleware



Buzzword:

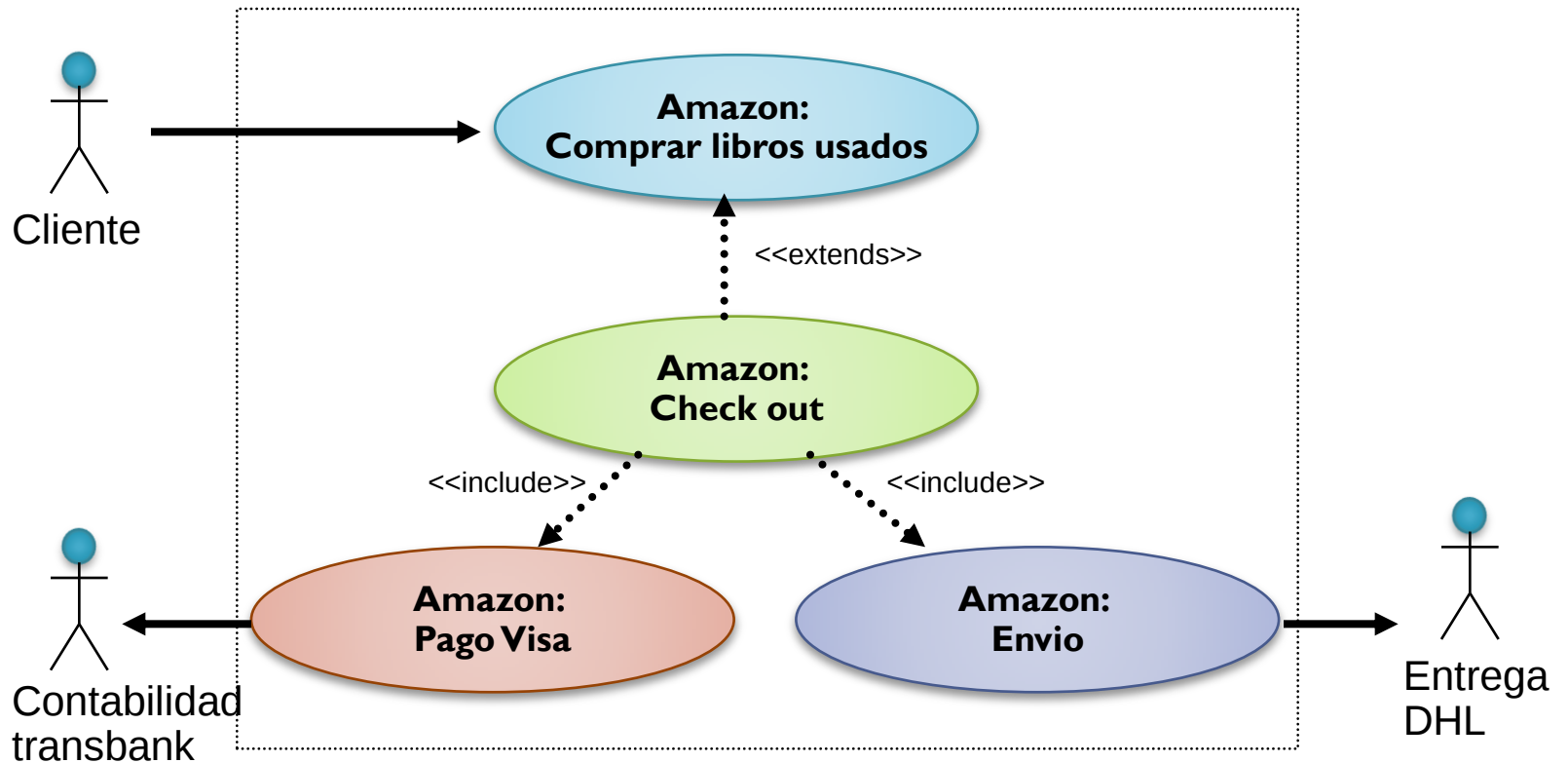
“Es el software que conecta aplicaciones”

- **Cual es la motivación?**

- E-business tienen *nuevas necesidades* que no están soportadas explícitamente en las arquitecturas clásicas.
 - Requieren *interconectarse* para tener en línea informaciones como:
 - rastrear productos (Amazon/DHL), disponibilidad de productos de terceros, estatus de cuentas bancarias.
 - Requieren *de efectuar transacciones con terceros*
 - Servicios de Webpay, Visa, Amazon.

- **Abarca una gran variedad de tecnologías específicas.**

Middleware para Amazon



¿Conexión con DHL y TRANSBANK?

Tipos de Middleware

- **Orientados a mensajes**
 - Componentes que *implementan un protocolo de comunicación con diferentes servicios en Internet.*
- **Orientados a RPC** (Remote Procedure Call):
 - Implementan la *posibilidad de ejecutar procedimientos en servicios remotos*
 - Ejemplo: Abortar transacción
- **Orientados al acceso de datos**
 - Típicamente *parecidos a ODBC.*
- **Orientados a las transacciones distribuidas**
 - Implementan un protocolo para *ejecutar operaciones distribuidas en la red*
 - Inicio, Fin y Rollback





ARQUITECTURAS MODERNAS

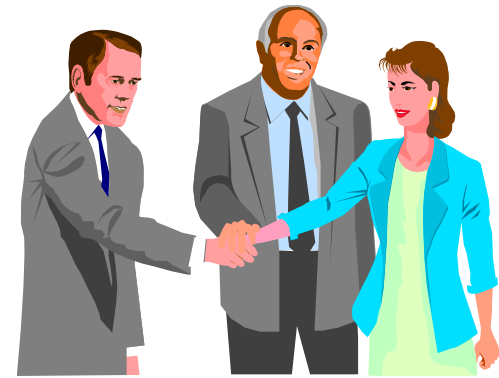
Paradigma peer to peer (p2p)



Bittorrent

- *Intercambia archivos entre personas conectadas a la red.*
- **Cada nodo actúa como cliente y servidor a la vez.**
 - Son los llamados **servents** (server+clients).
- **Existen servidores que almacenan a los usuarios conectados.**
 - Estos entregan *direcciones de quienes se encuentran online*
- Como cliente:
 - envía *queries* (consultas) de búsquedas y solicita archivos.
- Como servidor:
 - *responde peticiones.*

Paradigma peer to peer (p2p)



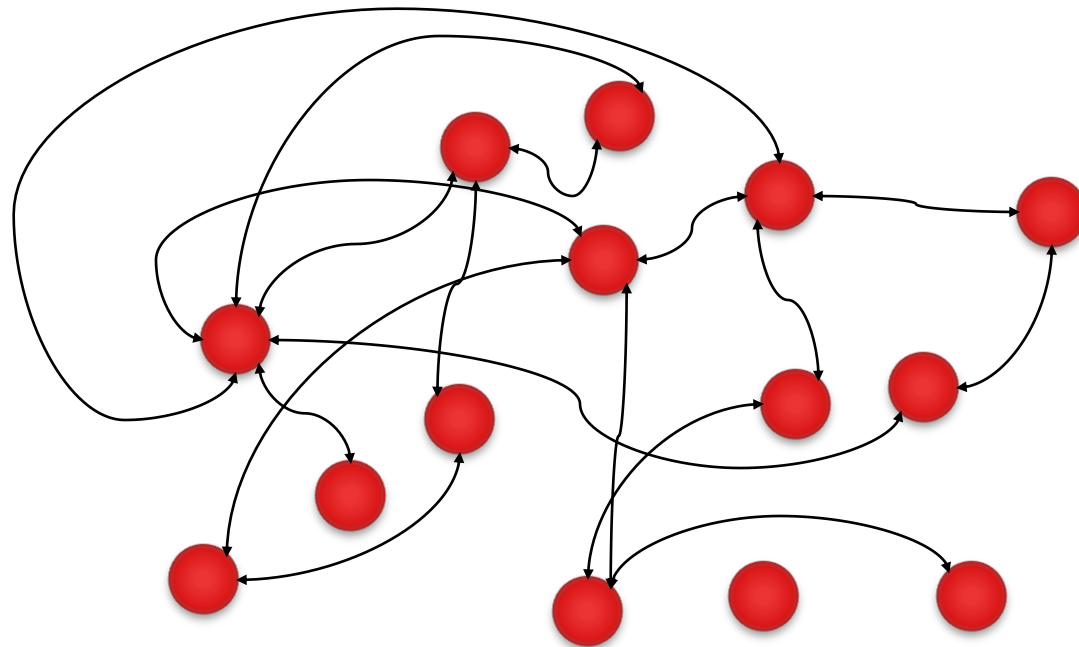
Protocolo Gnutella

- Mensajes de pertenencia al grupo:
 - Se efectúan **broadcast** para *detectar nodos vecinos y mejorar continuamente la conectividad.*
- **Cada nodo puede transferir una petición a todos sus vecinos o responderla.**
- **Búsquedas:**
 - *Se envían búsquedas que son propagadas a través de los nodos para la búsqueda de archivos (numero máximo de up).*
- Transferencia FTP
 - el protocolo de transmisión FTP.

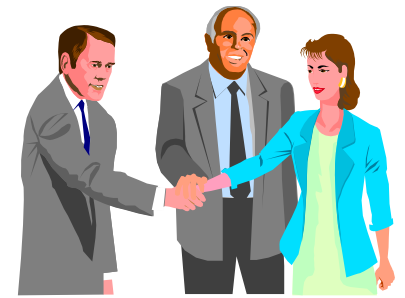
Paradigma peer to peer (p2p)

Resultados: en promedio la red se auto ajusta transparentando la comunicación.

Ya que se buscan **el camino mas rápido** para una transferencia.



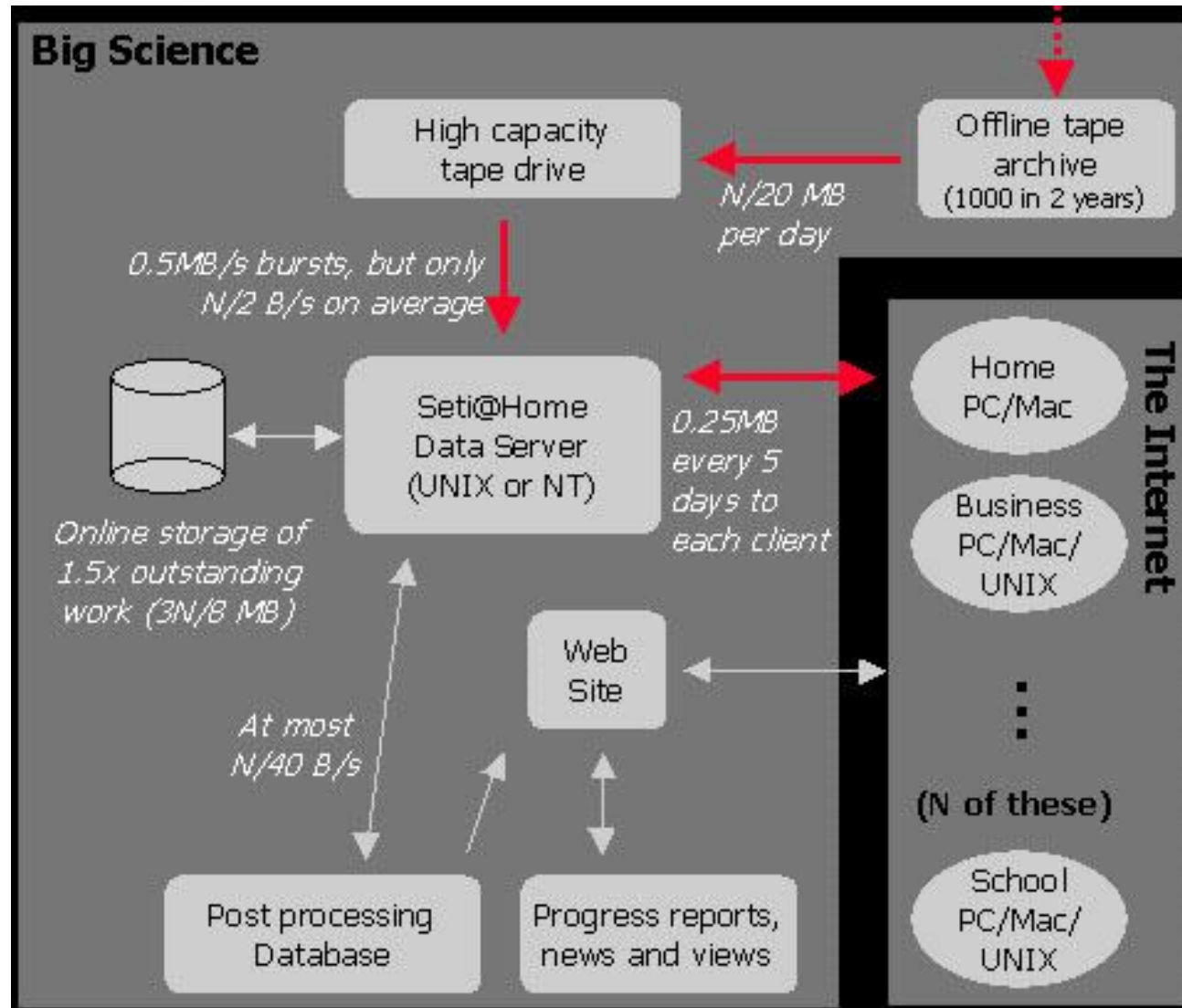
Paradigma peer to peer (p2p)



Importancia ...

- **Problemas con las arquitecturas centralizadas:**
 - *Interconectividad* inter-empresa.
- P2P entrega un **canal de comunicación colaborativo** que puede ser usado para *integrar sistemas*.
- Imagine las siguientes consecuencias:
 - P2P que use análogamente otro protocolo como Oracle, RPC, u otro. **¿En que se convierte p2p?**
 - P2P que realice búsquedas de servicios, **¿A que corresponde?**

Seti@Home



Seti@Home

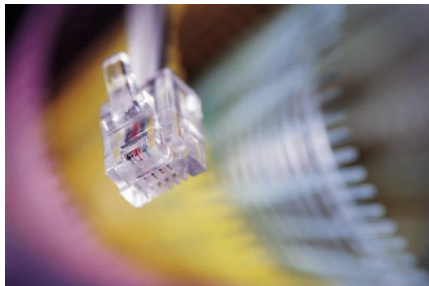


- Se requiere procesar para cada punto del cielo las señales recibidas por el radio telescopio Arecibo Puerto Rico.
- Se distribuye el procesamiento en cada PC cliente-seti.
- Se procesa y envía el resultado a un **repositorio centralizado**.
- *Cantidades astronómicas de datos.*
- Cada PC procesa una *porción de datos* cuyo *resultado se envía al computador central*.

Agentes Móviles

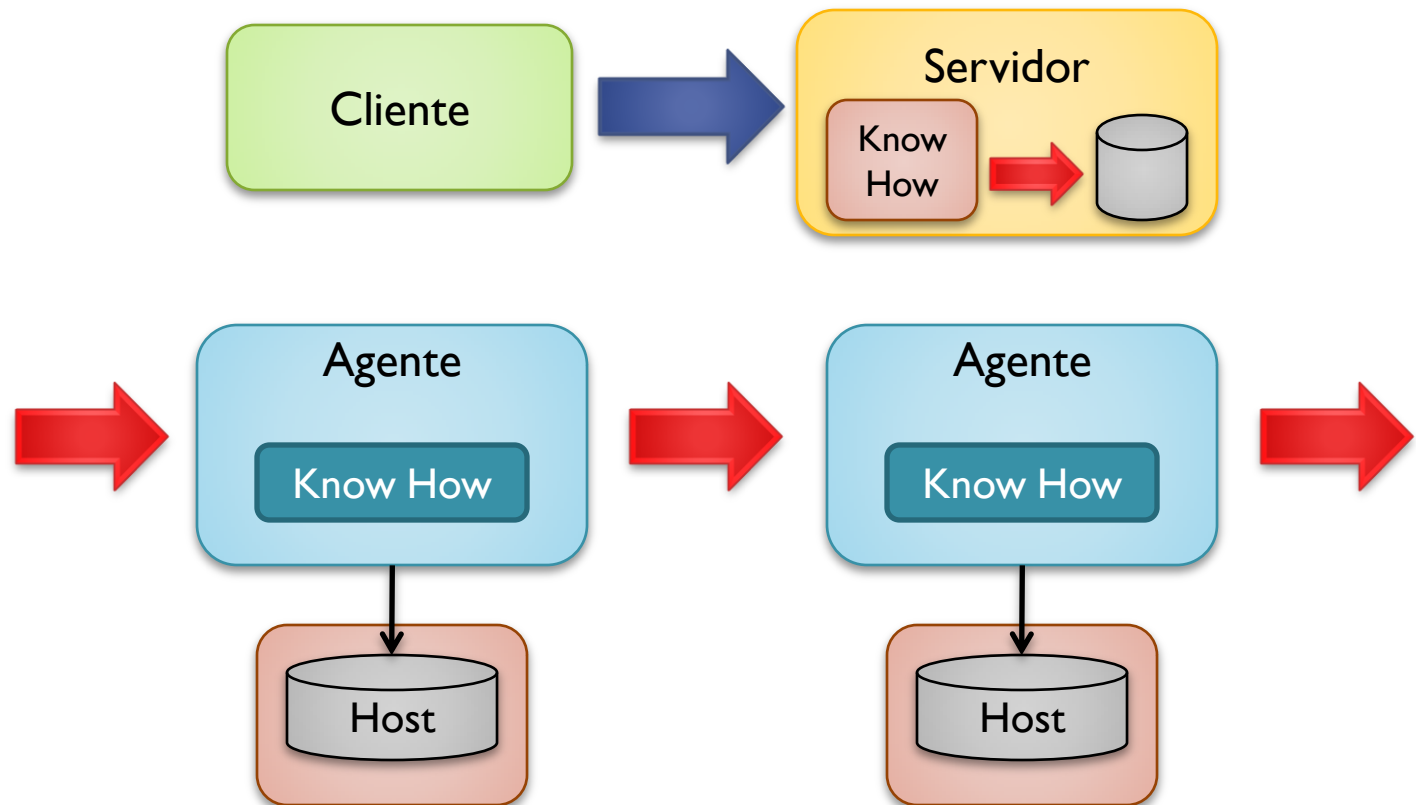


- Java Aglets.
- Perspectiva del **usuario final**:
 - un software que presta algún tipo de *asistencia* y que *actúa de forma autónoma*.
- Perspectiva del sistema:
 - objeto de software reactivo, autónomo, de ejecución continua, que se propaga por la red.
- Es un “**Viajero autónomo por la red**”



Agentes Móviles

- Paradigma de *agentes móviles vs cliente servidor*



Agentes Móviles

- Paradigma de *agentes móviles vs cliente servidor*

	Cliente Servidor	Agente Móvil
Característica	Escalabilidad de los servidores	Cada nodo es un servidor
	Calidad de conexión de la red	AM se conecta directamente.
	Protocolo estricto	AM se conecta directamente.