Introducción al middleware

Introducción

Temario

- Introducción:
  - Definición y motivación
  - Escenarios y características
- Diferentes tipologías de Middleware:
  - Basic, Platform, Gateways.
- Platform Middleware actuales:
  - Integration Brokers.
  - Enterprise Service Bus.
  - Middleware para User Interaction

Introducción

- ¿Qué es el middleware?
  - Es el “pegamento” (glue) que ayuda a la conexión entre programas (o bases de datos).
  - Más formalmente:
    - Es el soft-sistema que permite las interacciones a nivel de aplicación entre programas en un ambiente distribuido.
    - Por soft-sistema (system software) se entiende el software posicionado entre una aplicación y un sistema de menor nivel (S.Op, DBMS, Servicio Red).
    - Un ambiente computacional se dice distribuido cuando sus programas o BDs están ubicados en dos o más computadores.

Introducción

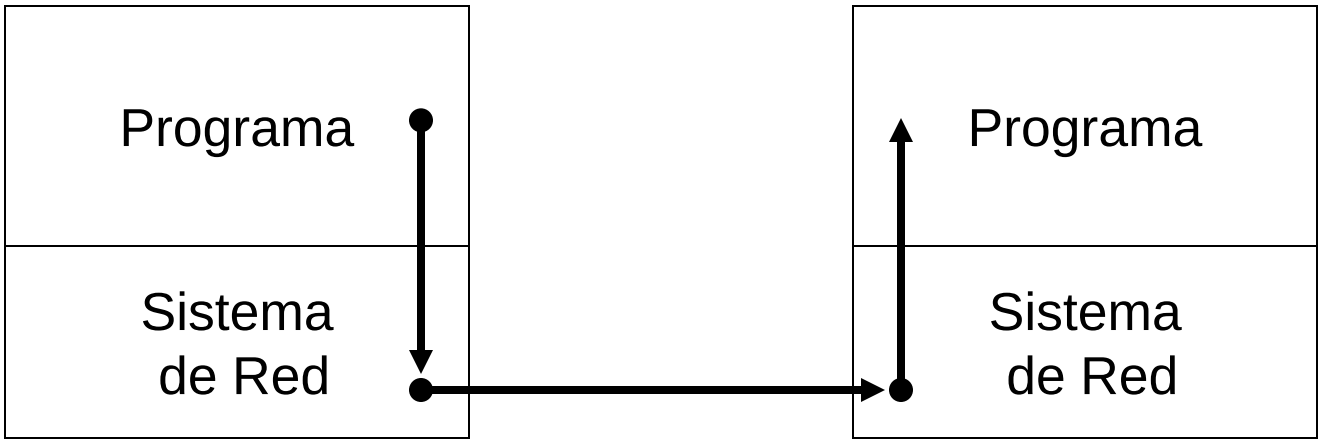
- ¿ Para qué usar middleware ?
  - Dadas dos aplicaciones que se quieren conectar, se usa para resolver la comunicación entre los procesos.
    - Si las aplicaciones se conectan directamente a soft de red, entonces no se necesita middleware.
    - Si no hay middleware se complica el desarrollo de aplicaciones:
      - Se debe programar módulos de bajo nivel.
      - Este desarrollo se repite para cada aplicación a conectar.
  - El soft de middleware permite realizar esta conexión a través de interfases de alto nivel, que permiten, por ej., ver un procedimiento remoto como si fuera local.

Introducción

- Escenarios de uso:
  - Cliente/Servidor en la misma máquina.
    - Se usa en sistemas de un computador, por ej. pequeñas oficinas, en casa, o en portables.
  - C/S a pequeña escala.
    - Aplicación clásica en una LAN con un único servidor.
    - Es la forma predominante de C/S.
  - C/S a gran escala.
    - Esquema multiservidor, que dan imagen de un único sistema.
  - C/S altamente distribuido.
    - Cada máquina es cliente y servidor, se basa en SOA.
    - Los servicios se utilizan con diferentes modalidades y acoplamiento.

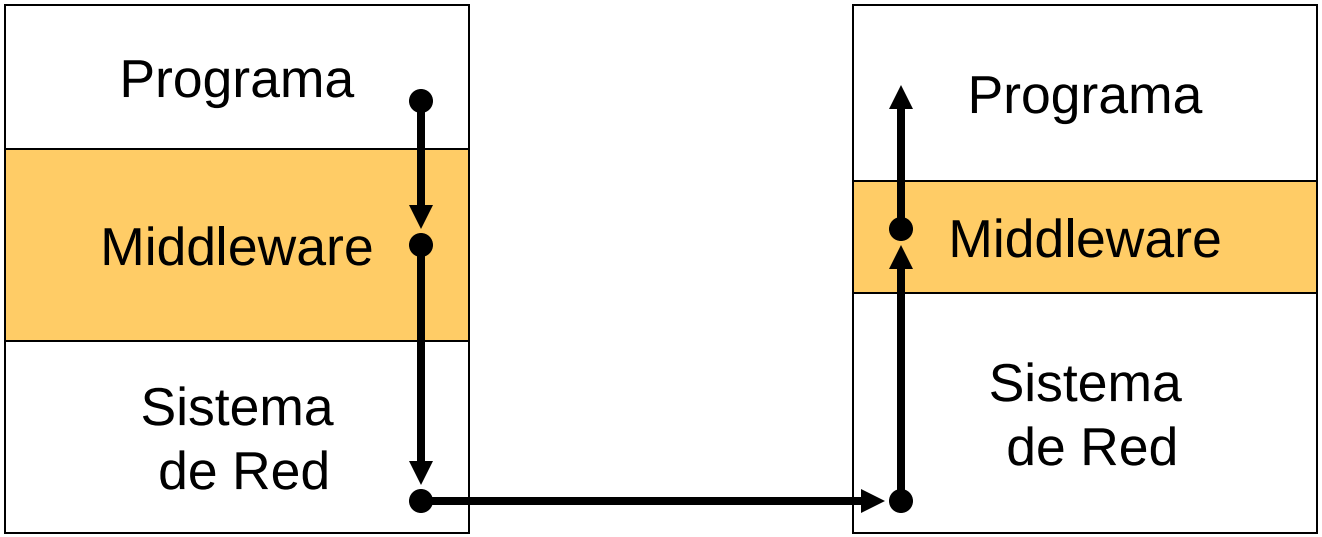
Introducción

- Esquema de conexión sin middleware.
  - Los programas deben resolver la conexión usando medios de bajo nivel, cercanos al Sistema de Red.



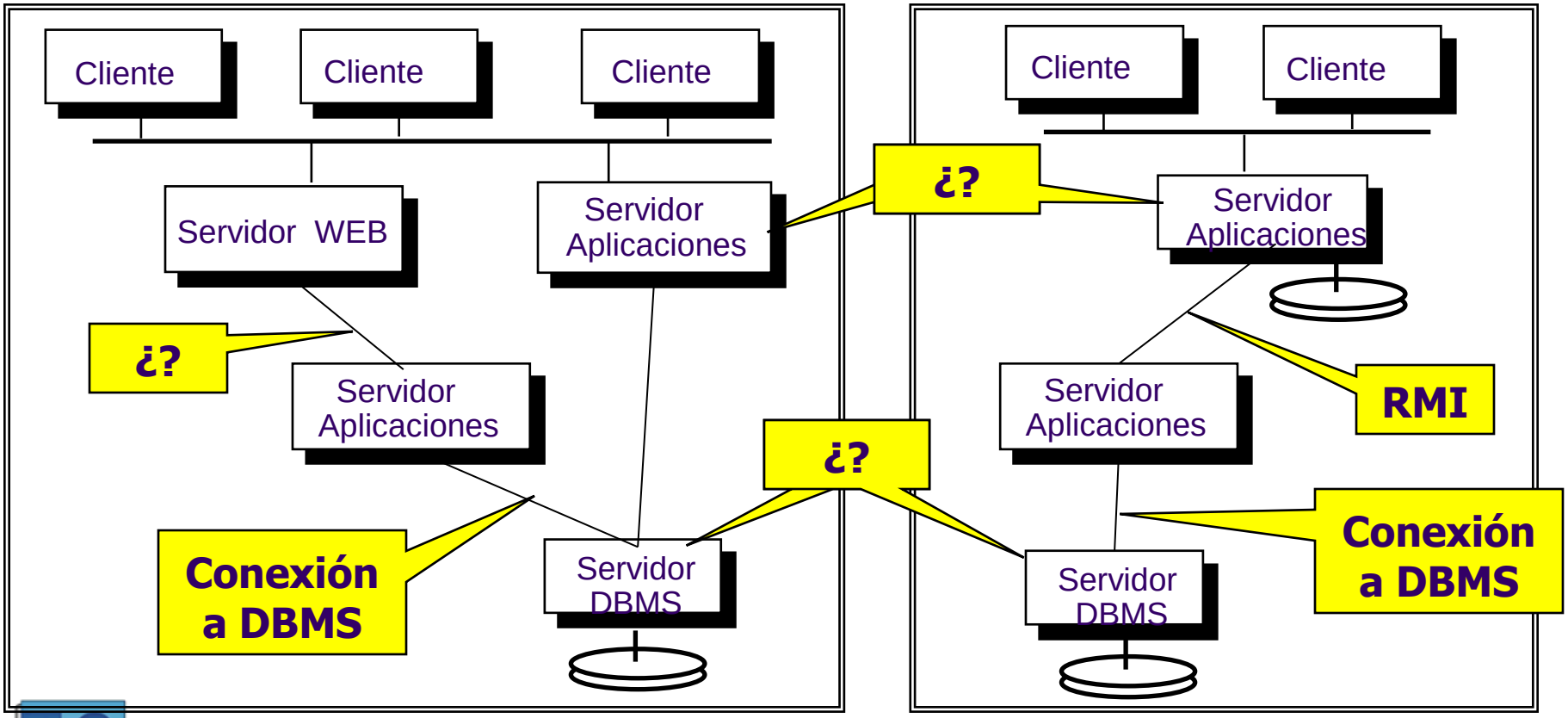
Introducción

- Esquema de conexión con middleware.
  - La capa de Middleware permite programar la comunicación mediante herramientas de alto nivel.
  - **Por ejemplo**: procedimientos, mensajes, acceso a objetos.



Introducción: Arquitectura (1)

- Aplicación en Arquitectura +3 niveles.



Introducción: Tipos Middl. (1)

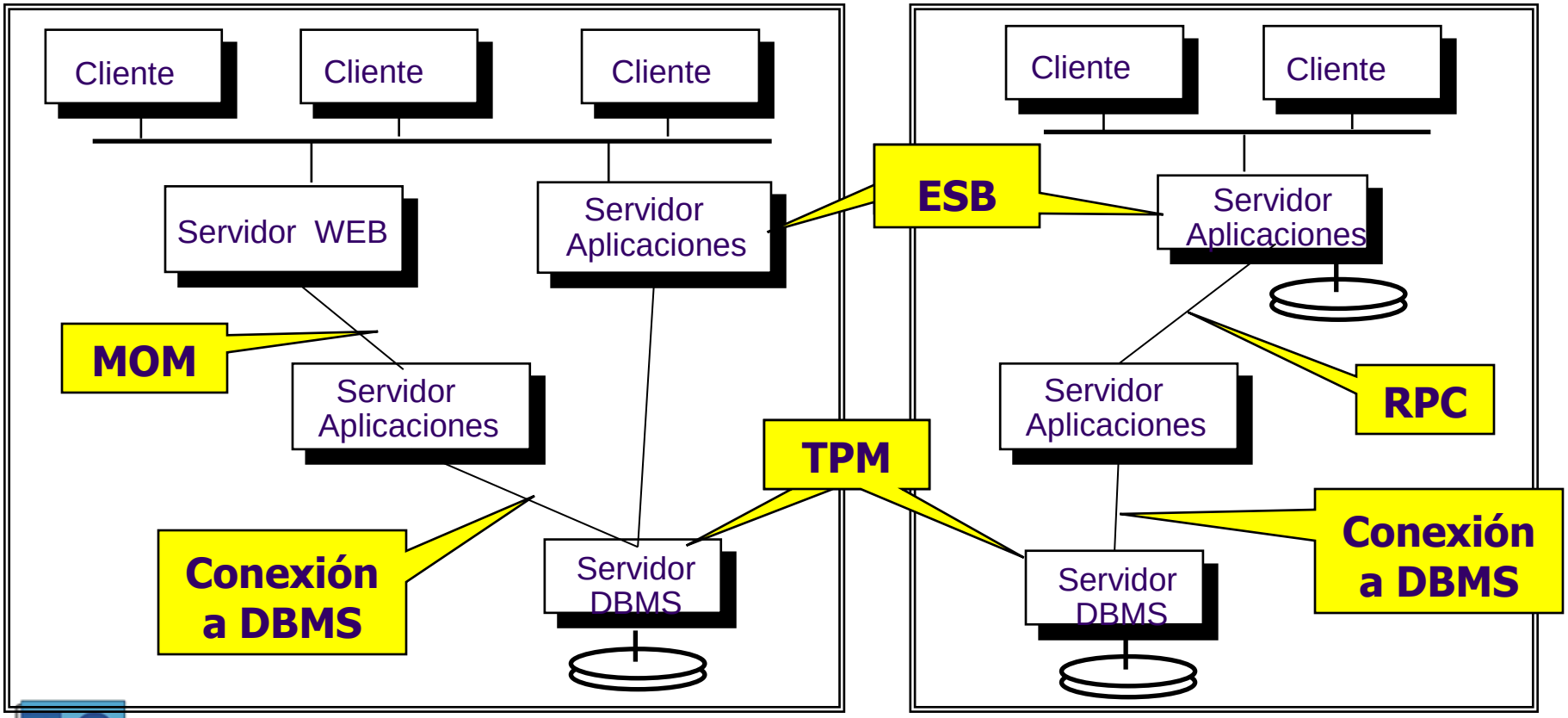
- Comunican 2 sistemas:
  - Drivers a DBMSs.
    - Acceso a DBMS desde un programa u otro DBMS.
  - Remote Procedure Call (RPC, RMI, Remoting).
    - Invocación a procedimientos remotos como si fueran locales al programa.
  - Web Services.
    - Invocación a procedimientos a través de HTTP.
- Comunican múltiples sistemas:
  - Message Oriented Middleware (MOM).
    - Envío de mensajes entre aplicaciones.
  - Object Request Brokers (ORB).
    - Invocación a procedimientos y propiedades de objetos.

Introducción: Tipos Middl. (2)

- Comunican múltiples sistemas:
  - Intregration brokers:
    - Comunican “n” aplicaciones en base a mensajes.
    - El “Integration broker” centraliza las comunicaciones:
      - Recibe mensajes de las aplicaciones.
      - Aplica reglas para determinar a qué aplicaciones deben enviarse.
  - Enterprise Service Bus:
    - Implementa mecanismos de comunicación:
      - Basado en invocaciones (de tipo RMI, Remoting, WS).
      - Basado en mensajes.
    - Son la evolución de los ORBs e Integration Brokers.

Introducción: Arquitectura (2)

- Aplicación en Arquitectura +3 niveles.



Introducción: Tipos Middl. (3)

- Portal servers.
  - Integrar aplicaciones (modularizadas) que se ejecutan en Portales.
    - Java: Servidores JSR 168 y JSR 286: portlets.
    - Microsoft: Sharepoint: webparts.
  - Son integrables a través del protocolo WSRP:
    - Web Service for Remote Portal.
- Mashup servers.
  - Integran aplicaciones heterogeneas.
  - Por ejemplo: Mapas, portlets/webparts, email, etc.

Introd.: características (1)

- Los middleware se caracterizan por implementar la interacción entre las aplicaciones de diferentes formas:
  - Interacción sincrónica:
    - Cuando una aplicación es “invocada” por otra, se ejecuta inmediatamente.
  - Interacción (sincrónica) bloqueante:
    - Cuando una aplicación invoca a otra, la primera queda esperando la respuesta de la segunda.
  - Interacción (sincrónica) no-bloqueante:
    - ¿ como sería ?

Introd.: características (2)

- Interacción asíncrona:
  - Una aplicación invoca otra pero no espera su ejecución inmediata.
  - Se implementa en base a mensajes.

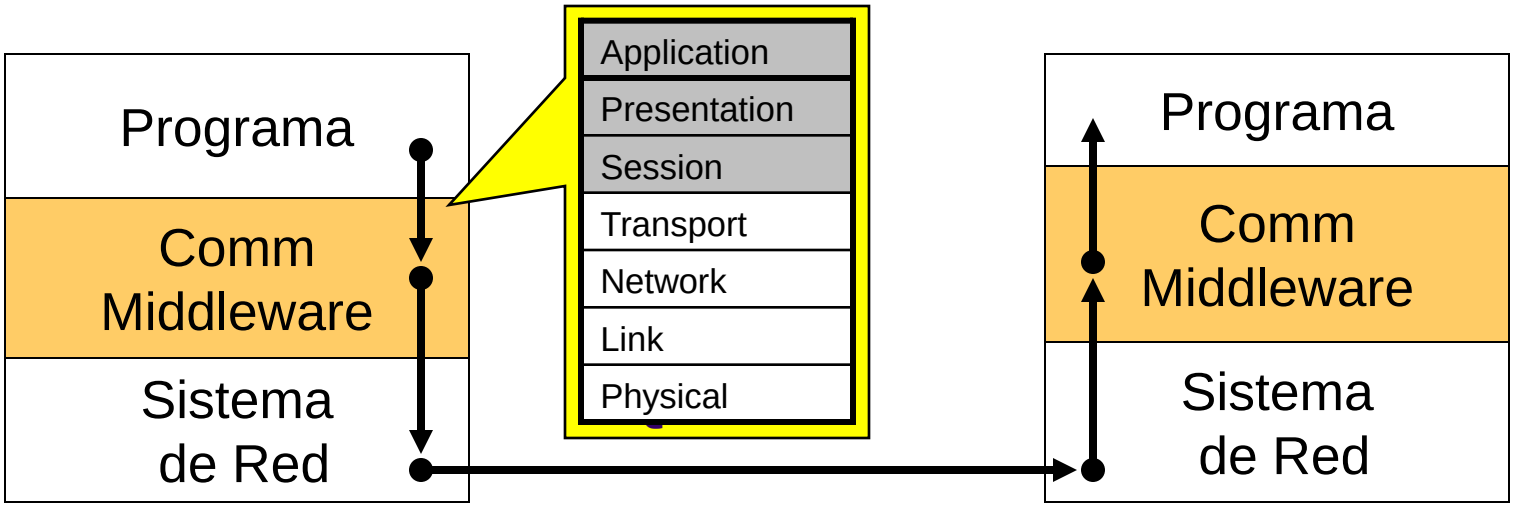
- Asegurando consistencia en los datos:
  - Transaccionalidad (2PC).
  - Consistencia en ambiente debilmente acoplado.
  - Mensajes persistentes.

Tipología de Middleware

- Basic:
  - RPC (RMI, etc.).
  - Message Oriented Middleware (colas de mensajes).
  - Data Middleware (drivers BD.).
- Platform:
  - ORB.
  - TPM.
  - Intregration Brokers.
  - Application Servers.
  - Enterprise Service Bus
- Gateways.

Basic Middleware

- Características:
  - Resuelven la comunicación entre 2 programas.
  - Cubre de las capas 5 a la 7 del stack OSI.
- Ejemplos:
  - RPC, MOM, Data Middleware.



RPC: Remote Procedure Call

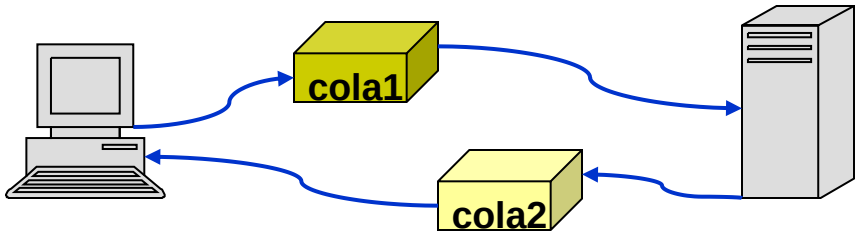
- Esconde la red, invocando procedimientos.
  - Cliente invoca a una función del servidor remoto y se bloquea hasta tener el resultado.
  - Se pasan parámetros de la forma normal.
- Componentes:
  - Aplicaciones: cliente y servidor se programan como locales.
  - Stub: Empaqueta, convierte...
    - Lenguaje IDL (Interface Definition Language).
    - Compilador IDL genera Stubs (C y S), que se linkeditan al prog.
  - Runtime:
    - En cliente invoca el RPC y se bloquea.
    - En servidor, recibe invocaciones (prioridades, seguridad...)

Asynchronous Middleware

- Permiten activar un proceso sin que el “invocador” quede bloqueado
  - Facilita la integración de aplicaciones en contextos de acoplamiento débil.
- Basado en envio de mensajes.
  - Por eso se les conoce como Message Oriented Middleware (MOM).

MOM: Message Oriented Middleware

- Comunicación usando colas de mensajes:
  - Aplicaciones sólo ponen y sacan mensajes de colas.
  - No se conectan. C y S pueden correr en diferentes tiempos.
  - No necesariamente se requiere respuesta.



- Consideraciones:
  - Se pueden implementar esquemas 1-N o N-1
    - Muchos clientes, varias instancias del servidor.
  - Colas pueden estar en disco o en memoria.
  - Pueden ser FIFO, por prioridades, balance de carga...

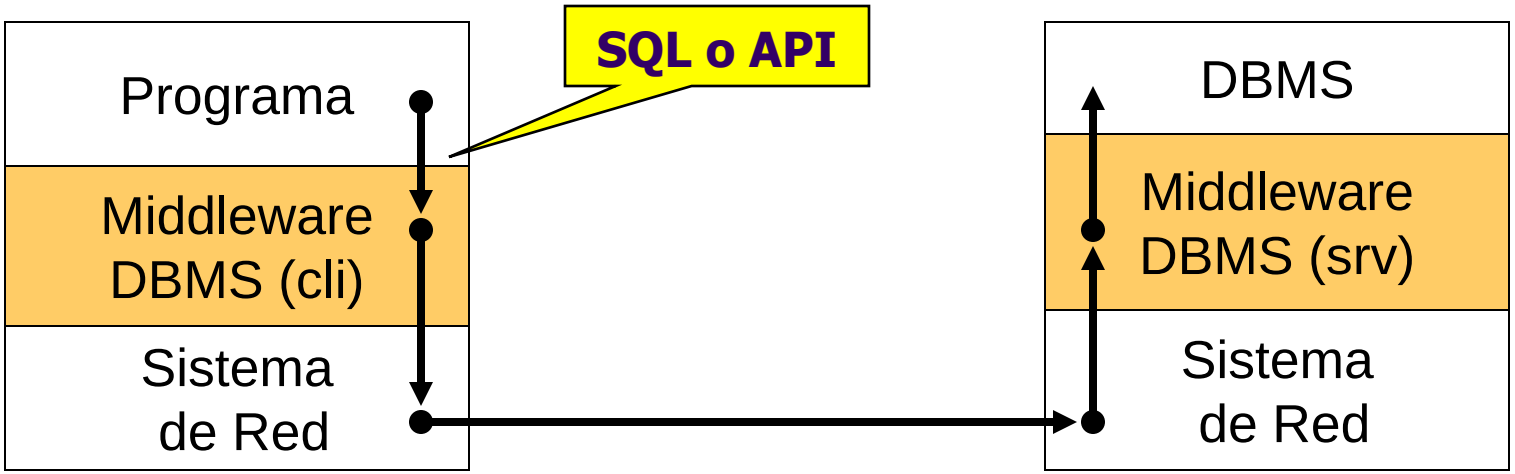
RPC vs. MOM

- RPC:
  - Síncrono: Se requiere una conexión. Cliente se bloquea.
  - Respuesta inmediata. Se asegura tiempo de respuesta.
  - Ideal para aplicaciones que sincronizar acciones.
  - Ejemplo: Aplicaciones interactivas, transacciones.

- MOM:
  - Asíncrono: Clente y servidor operan en diferentes tiempos.
  - Respuesta (eventualmente) lenta. No se asegura totalmente un tiempo de respuesta.
  - Ideal para informar, para aplicaciones poco conectadas.

Data Middleware

- Características:
  - Conectan programas con DBMS o DBMSs entre si a través de un API, con uso opcional de lenguaje de consulta.
  - Fuertemente asociados a tecnologías de DBMS.
  - Incluyen un componente cliente y otro servidor.
- Ejemplos:
  - ODBC, OLEDB, JDBC



SQL Middleware

- Objetivo ideal:
  - Diferentes DBMS, que dan la ilusión de ser un único sistema: sistema federado.
  - Diferentes clientes accediendo al sistema federado.

- Problema:
  - SQL no es tan estandar: SQL (‘86), SQL2 (‘92), SQL3 (‘99).
    - Cada vendedor tiene sus propias extensiones (dialectos).
  - Diferencias en:
    - APIs (Application Programming Interface).
    - Driver: Runtime que acepta llamadas, formatea mensajes (FAP: Format and Protocols) y maneja el intercambio.
    - Stacks. Sólo algunos usan transp standard: sockets, named pipes

Platform Middleware

- Características:
  - Permiten la comunicación entre programas a través de mecanismos de mayor nivel que los otros Basic Middleware.
  - Combinan técnicas de los Basic Middleware.

Platform Middleware

- Además proveen funciones tales como:
  - Gestión de memoria y procesos del S. Op.
  - Carga de programas, inicio y fin, pasaje de mensajes.
  - A veces balance de carga y gestión de transacciones.

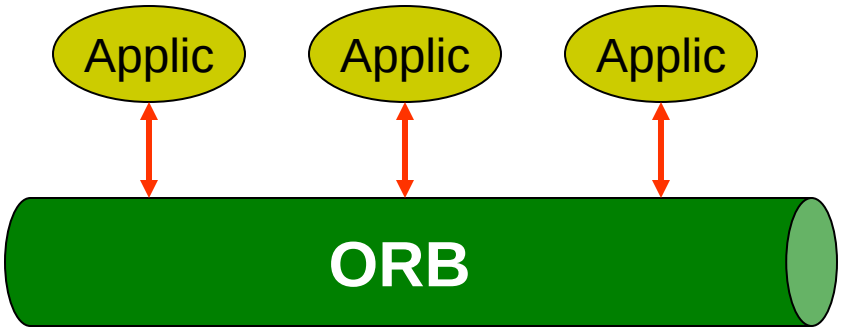
- Ejemplos:
  - Application Servers y ORBs (CORBA, JEE, .NET.)
  - TPM (Tuxedo, CICS, Encina)
  - Integration Brokers (IBM MQSeries, MS Biztalk).
  - Enterprise Service Bus.

ORBs: Object Request Broker

- Permiten:
  - Programar ensamblando componentes (building blocks).
    - Empaquetados como piezas de código indep y autocontenidas.
      - No está asociado a un programa, lenguaje o implementación.
    - Accedidos por invocaciones a métodos.
    - Interfase bien definida (IDL: Interface Definition Language).
  - Portabilidad e Interoperabilidad:
    - Transparentes al lenguaje, compilador, ubicación, s. operativo.
    - Se importan dentro de paletas o toolbars.
    - Puede ser invocado a través de espacios de direcciones, redes, lenguajes, sist operativos y herramientas.

ORB: Object Request Broker

- Es un bus de comunicación entre objetos:
  - Permite hacer/recibir requerimientos en forma transparente:
    - Objetos locales o remotos.
  - Funcionamiento:
    - Objeto cliente invoca un método en un objeto remoto.
    - ORB:
      - Localiza una instancia del objeto servidor.
      - Invoca el método.
      - Retorna el resultado al cliente.

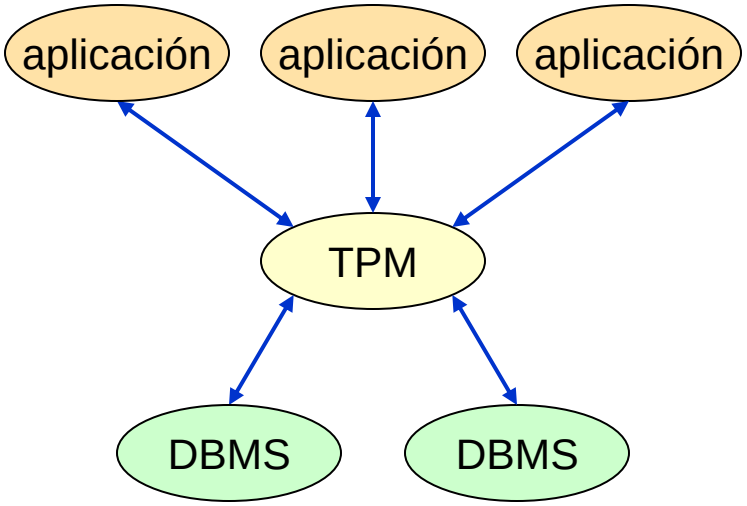


ORB: Object Request Broker

- Funcionalidades:
  - Control de transacciones y bloqueo:
    - Integridad “todo o nada”.
    - Locks para serializar acceso a recursos.
  - Persistencia.
  - Relacionamiento:
    - Relaciones dinámicas o permanentes con otros componentes.
  - Auto-testeo:
    - Correr programas de diagnóstico para determinar problemas.
  - Auto-instalación:
    - Instalarse y registrarse con S.O y/o registry.
    - Des-instalarse.

Monitores Transaccionales

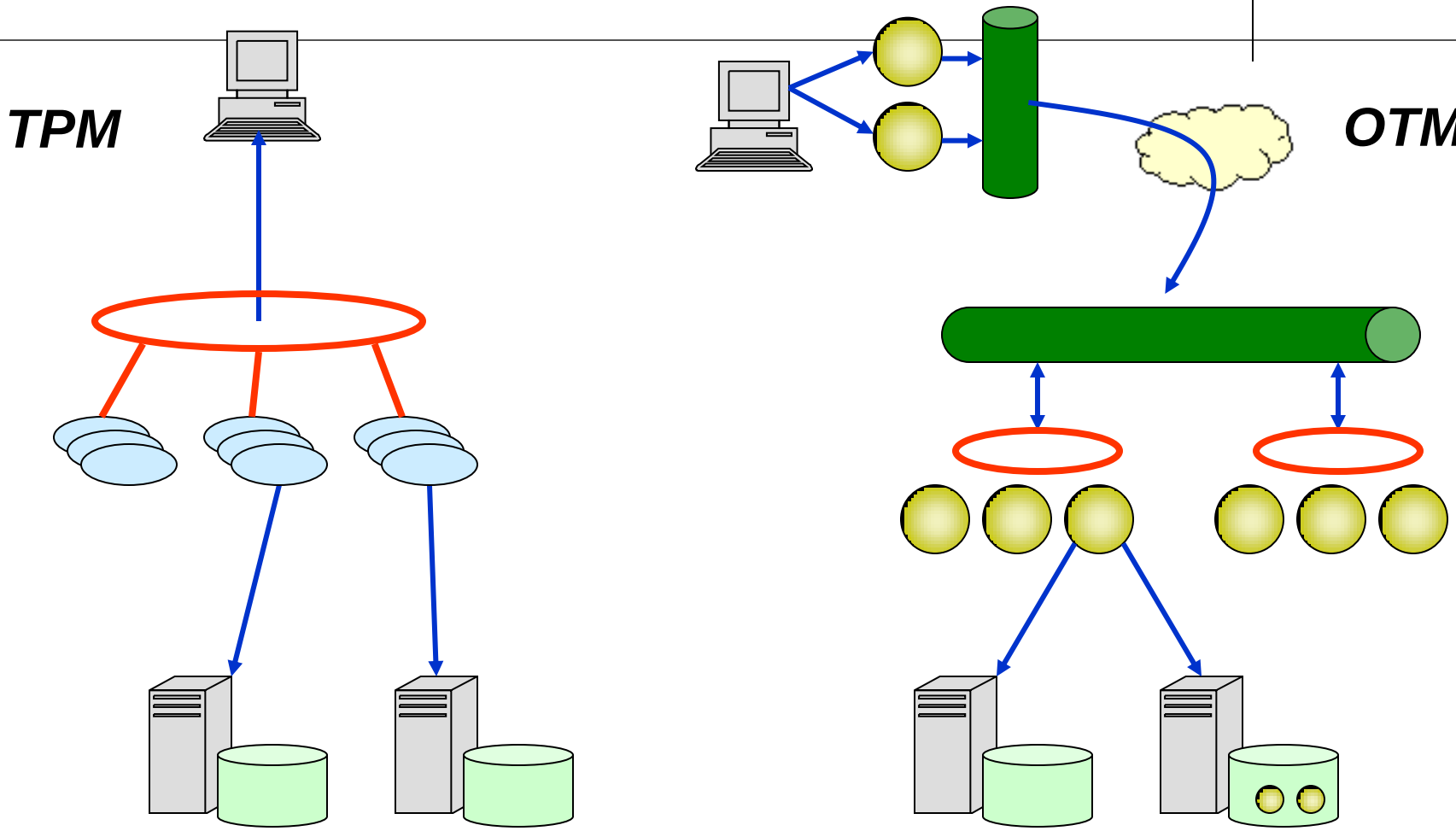
- Objetivo:
  - Es un sistema especializado en la creación, ejecución y manejo de aplicaciones de procesamiento de transacciones.
- Características:
  - Sistemas transaccionales tienen:
    - Muchas transacciones pequeñas.
    - Muchos usuarios concurrentes.
  - Coordinan las transacciones con:
    - Subsistemas ACID locales.
    - Manejadores de recursos.
      - DBMS, manejadores de colas, objetos persistentes, transporte de mensajes.



Monitores Transaccionales

- OTM: Object Transaction Monitors
  - Combinan ORBs con monitores de transacciones.
    - Maneja contenedores que corren los componentes que brindan los servicios.
  - Maneja objetos logrando: transaccionalidad, robustez, persistencia, seguridad, performance.
  - Carga un conjunto de objetos (pool), distribuye la carga, provee tolerancia a fallos, y coordina transacciones multi-componentes.

Monitores Transaccionales



Servidores de Aplicaciones

- Contexto:
  - Arquitecturas en múltiples capas:
    - Cliente: interface usuario.
    - Servidor Web:
      - Acceso HTTP, interface usuario.
    - Servidor de Aplicaciones:
      - Lógica del negocio.
      - Lógica de los datos.
      - Gestión de Transacciones.
      - Acceso a la BD.
      - Balance de carga en configuraciones paralelas.
    - Servidor de Base de Datos: almacenamiento.

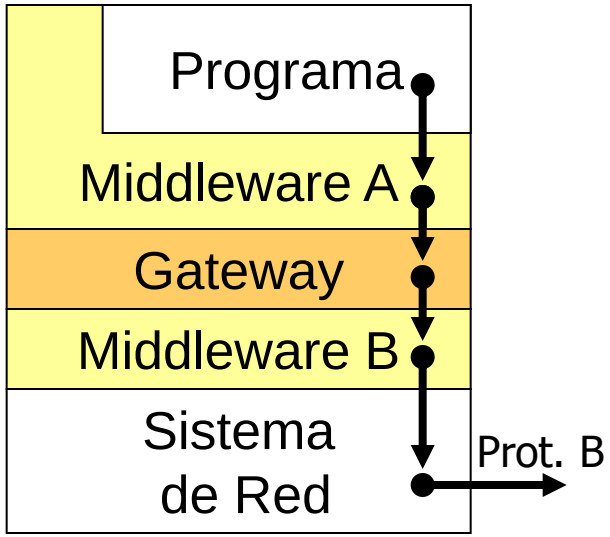
Servidores de Aplicaciones

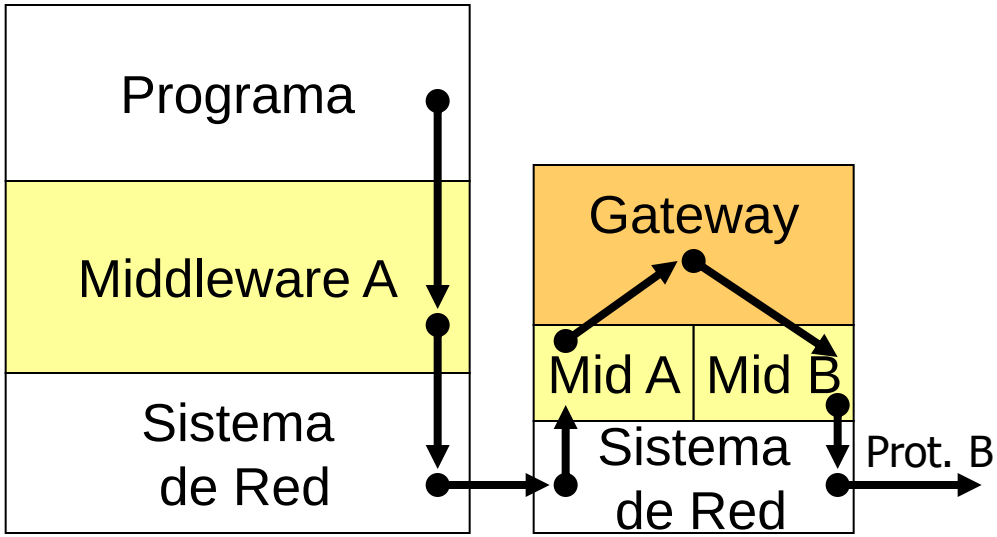
- Cubren:
  - Nivel Servidor de Aplicaciones.
  - Casi seguro: Gestión de Transacciones.
  - Web Server.

- Grandes familias:
  - JEE: Propuesta de Java.
  - COM/DCOM/COM+ .NET: Propuestas de MS

Gateways

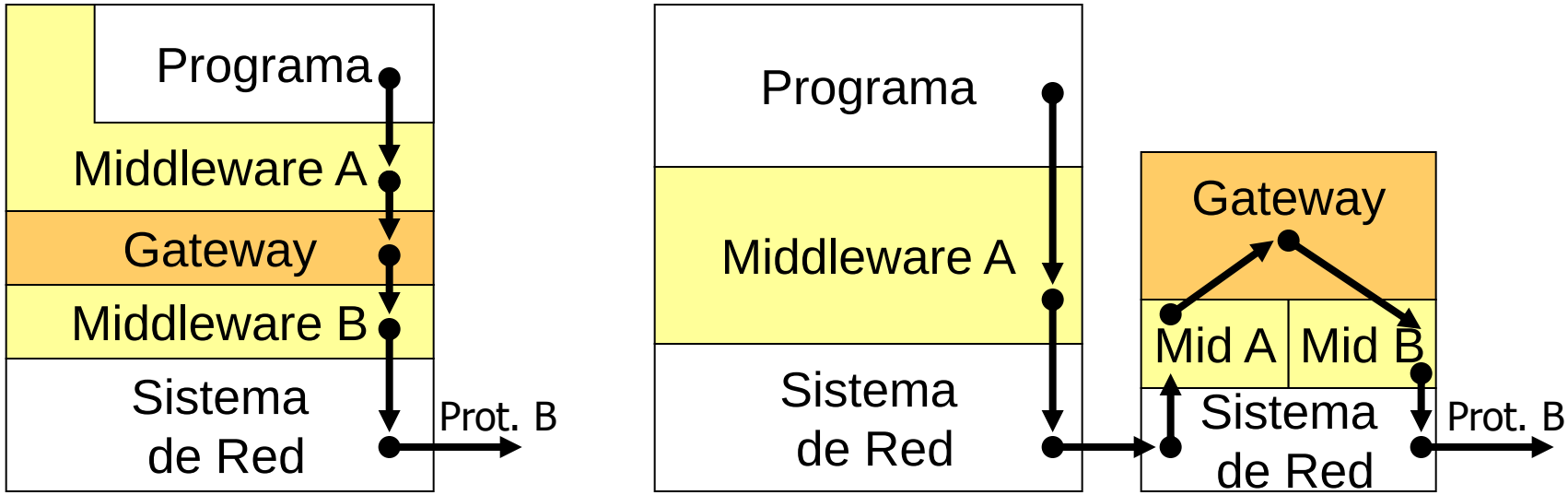
- Características:
  - Realizan la traducción entre 2 o más protocolos.
  - Existen gateways para:
    - DBMS, MOM.
    - Platform Midd: Corba←→COM, .NET←→JEE.





Gateways

- Características:
  - Realizan la traducción entre 2 o más protocolos.
  - Existen gateways para:
    - DBMS, MOM.
    - Platform Midd: Corba←→COM, .NET←→JEE.



Platform Middleware actual

- Middleware que permite integrar aplicaciones a escala empresarial.
- Provee al menos:
  - Capacidad para integrar aplicaciones:
    - Syncrónica y asincrónicamente
    - En equipos distribuidos.
  - Control de transacciones.
- Incluye:
  - Application Servers.
  - Integration Brokers.
  - Enterprise Service Bus.

Integration Broker

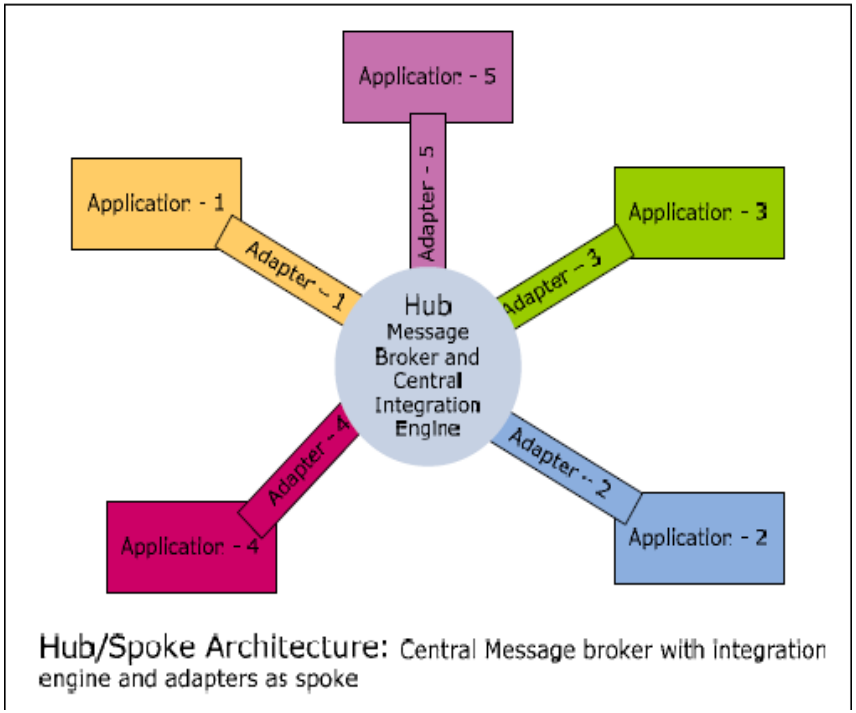
- Características:
  - Son intermediarios que facilitan la interacción entre programas.
    - Principalmente orientados a mensajes.
  - Proveen dos funciones de interés:
    - Transformation:
      - Transforma mensajes o contenidos de archivos.
      - Transforma modelos de datos de diferentes aplicaciones a un modelo común.
    - Flow automation (or flow control):
      - Son tratamientos inteligentes de flujos, por ejemplo: ruteo inteligente y/o basado en contenidos.

Integration Broker

- Características:
  - También pueden ofrecer:
    - Business process management (p.ej, workflow)
      - Interpretan reglas de negocio y responden a eventos de negocio y excepciones. Ayudan a automatizar tareas.
    - Message warehousing.
    - Administrative monitoring.
  - Algunos requieren un MOM en especial (ej. IBM MQSeries),otros tienen interfases a una gran variedad de productos.

Integration Brokers

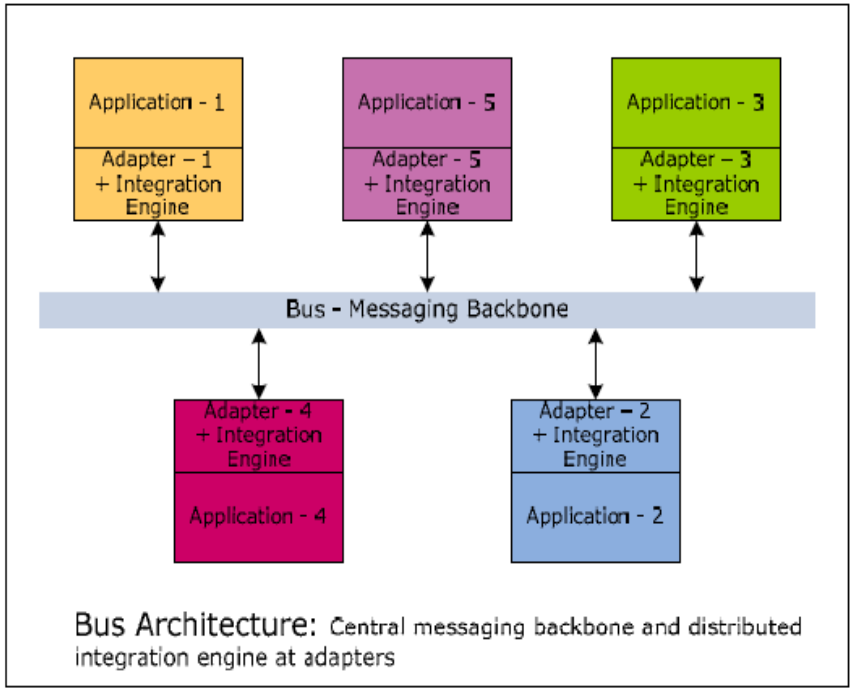
- Arquitec. “Hub & Spoke”:
  - Altamente centralizada.
    - Centrada en el HUB (message broker) es una pieza monolítica de software.
    - Realiza las operaciones de transformación y ruteo de mensejes.
    - Estas funcionalidades no puede ser conectadas a otro HUB.
    - Estos servicios son propietarios.
  - Los Spokes son aplicaciones a integrar.
  - La excesiva centralización complica la escalabilidad.



Hub/Spoke Architecture: Central Message broker with integration engine and adapters as spoke

ESB

- Arquitectura de BUS.
  - Incluye procesos que ejecutan las funcionalidades tales como ruteo, transformaciones, transacciones, etc.
  - Estos servicios son conectables a otros ESB.
  - Varios ESB pueden conectarse y hacer visibles los servicios publicados: favorece la escalabilidad.
  - Están basados en estándares y/o facilitan la interoperabilidad entre aplicaciones.
    - Por ejemplo: Web Services.



Bus Architecture: Central messaging backbone and distributed integration engine at adapters

ESB, SOA, Applic. Server y eventos

- ESB como Bus de Servicios:
  - Fuertemente asociados a la implementación de arquitecturas orientadas a servicios.
    - Las aplicaciones a integrar se “publican” como ofreciendo y/o consumiendo servicios.
- ESB interactuando con Applic. Servers
  - Se construyeron pensando en integrar aplicaciones ejecutándose en Applic. Servers.
- Las funcionalidades asíncronas:
  - Basadas en colas de mensajes.
  - Incluyen gestión de eventos para la suscripción (a mensajes publicados en colas o canales)

ESB vs. Integration Brokers

- Los ESB resultan más abiertos y escalables.
- También más interoperables con otros productos.
- Las ventajas se deben en gran parte:
  - La historia y evolución de los productos.
    - Los ESB surgieron posteriormente a los Integration Brokers, y corrigieron muchas de sus defectos.
  - Varios ESB se basan en Integration Brokers:
    - Biztalk Server de Microsoft.
    - Productos de IBM WebSphere.

Middleware para User interaction

- Las aplicaciones que interactuan con el usuario también deben integrarse:
  - Con los procesos de back-end, que implementan funcionalidades de negocio.
  - Con otros aplicaciones que implementan interacción con el usuario.
  - Con servicios utilitarios:
    - Por ejemplo: mapas, gestión de videos, etc.

- ¿ Como hacer posible esta integración ?

User interaction Middleware

- Modularización de aplicaciones.
  - Las aplicaciones que implementan la User Interaction deben implementarse basadas en componentes:
- Estandarización de protocolos.
  - Estas componentes deben poder interoperar entre si lo más posible.
- Orientación a servicios.
  - La interacción entre estos componentes y los otros externos debe seguir los mismos modelos que con los otros middleware.

User interaction Middleware

- Portales (Portal Servers).
  - Son servidores de:
    - “portlets” en plataforma Java.
    - “webparts en plataforma Microsoft (Sharepoint).
  - La interacción entre ellos es posible:
    - Dentro de la misma plataforma.
    - Utilizando WSRP (Web Service for Remote Portal) en plataformas diferentes.
  - Se integran con las Platform Middleware.

User Interaction Middleware

- Mashup.
  - Apuntan a facilitar el desarrollo de aplicaciones que combinan multiples funcionalidades con User Interaction.
  - Por ej:
    - Mapas + portlets + email + chat + videos.
  - Menos estandarizado que los Portal Server.
  - Compatibles con los Portal Servers.
  - Tratan de ser más livianos que los Portal Servers.

User Interaction Middleware

- Mashup.
  - Apuntan a facilitar el desarrollo de aplicaciones que combinan multiples funcionalidades con User Interaction.
  - Por ej:
    - Mapas + portlets + email + chat + videos.
  - Menos estandarizado que los Portal Server.
  - Compatibles con los Portal Servers.
  - Tratan de ser más livianos que los Portal Servers.