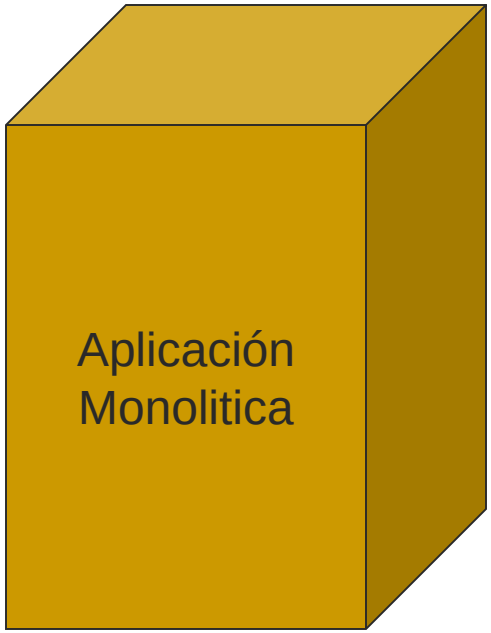
Desarrollo de software en sistemas distribuidos

Curso 2019

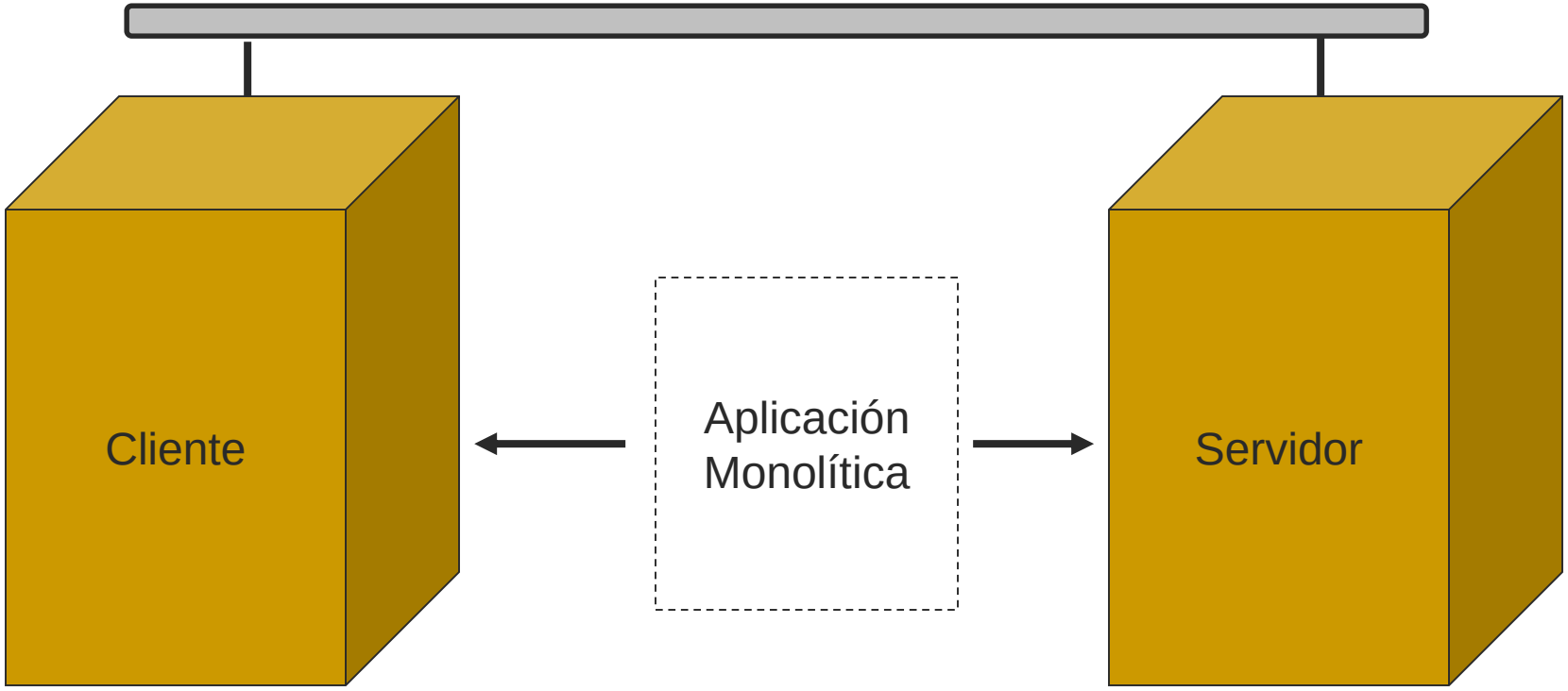
Evolución histórica de las aplicaciones

- Aplicación Monolítica
- Arquitectura Distribuida (2 capas y N capas)
- Arquitecturas Distribuidas con objetos (estándar CORBA)
- Integración de aplicaciones
- Arquitecturas Orientadas a Servicios

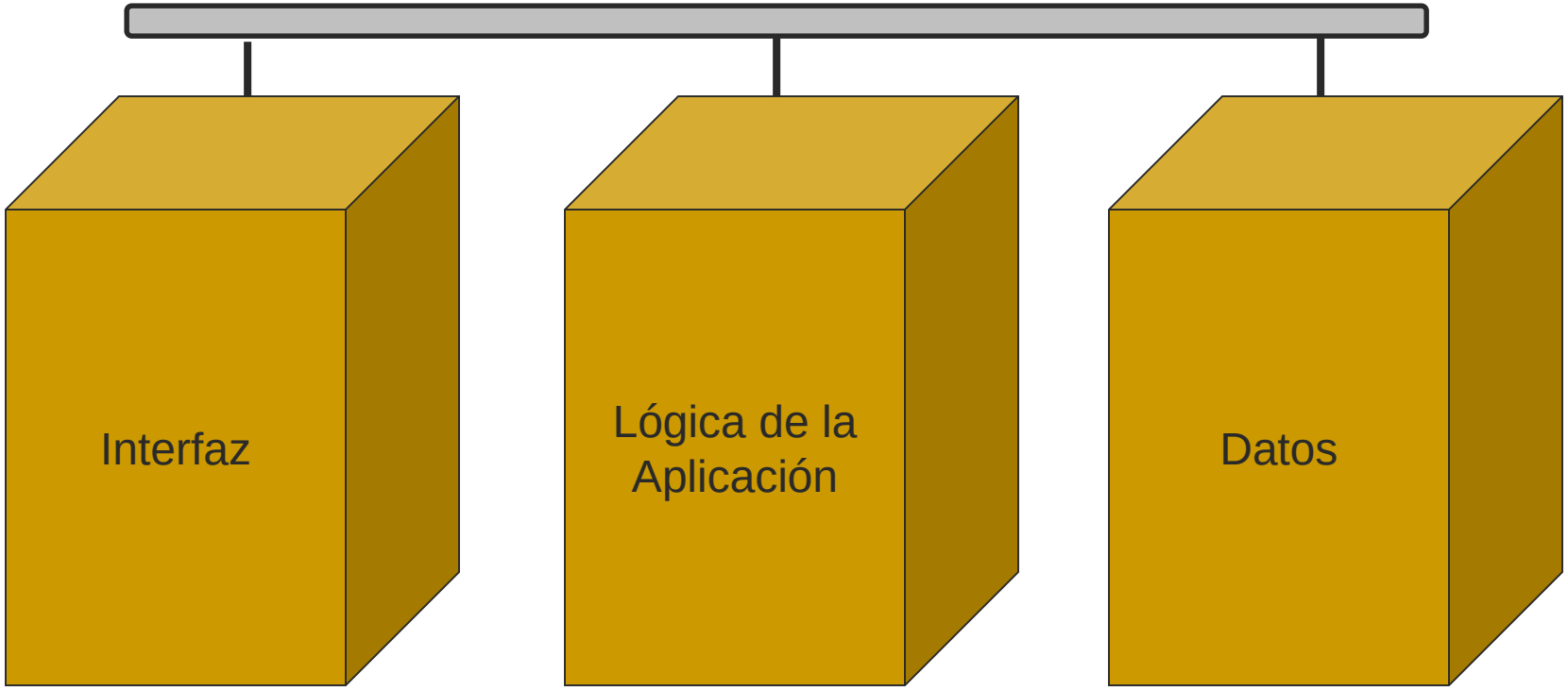
Aplicación Monolítica



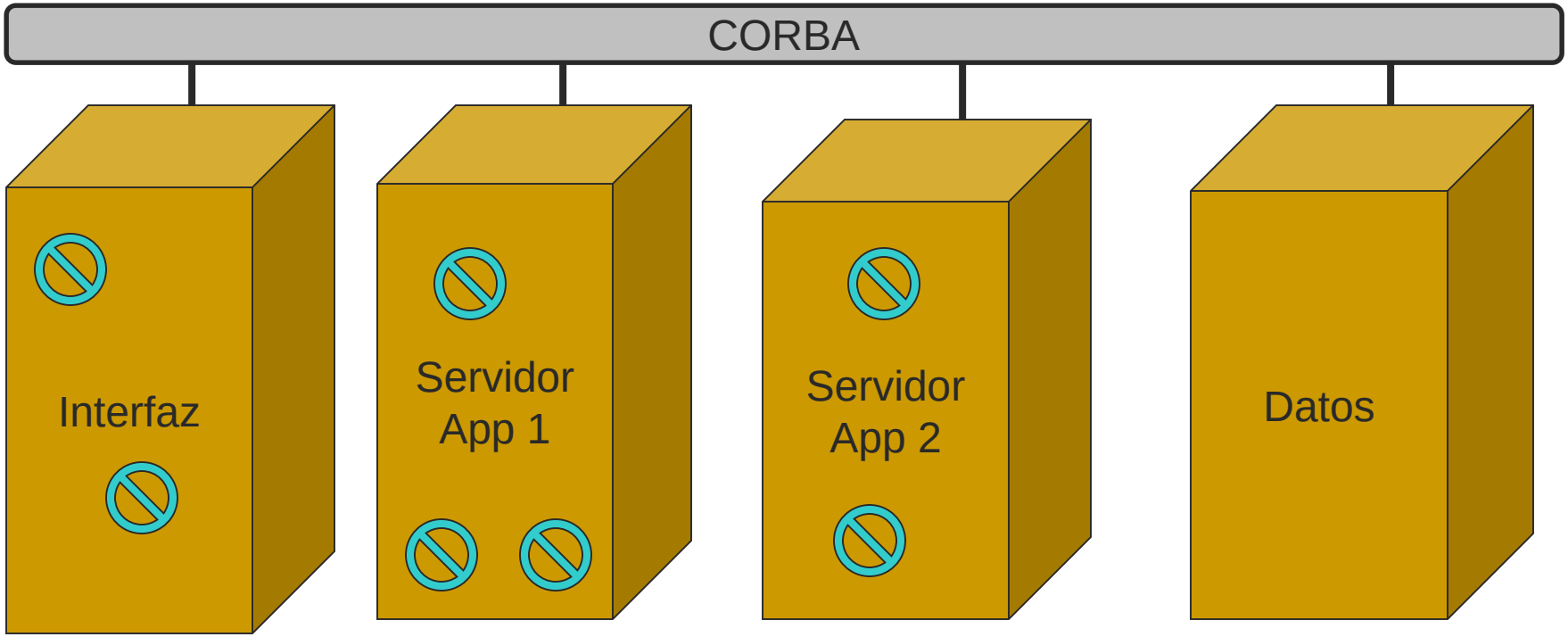
Arquitectura Distribuida Cliente/Servidor (2 capas)



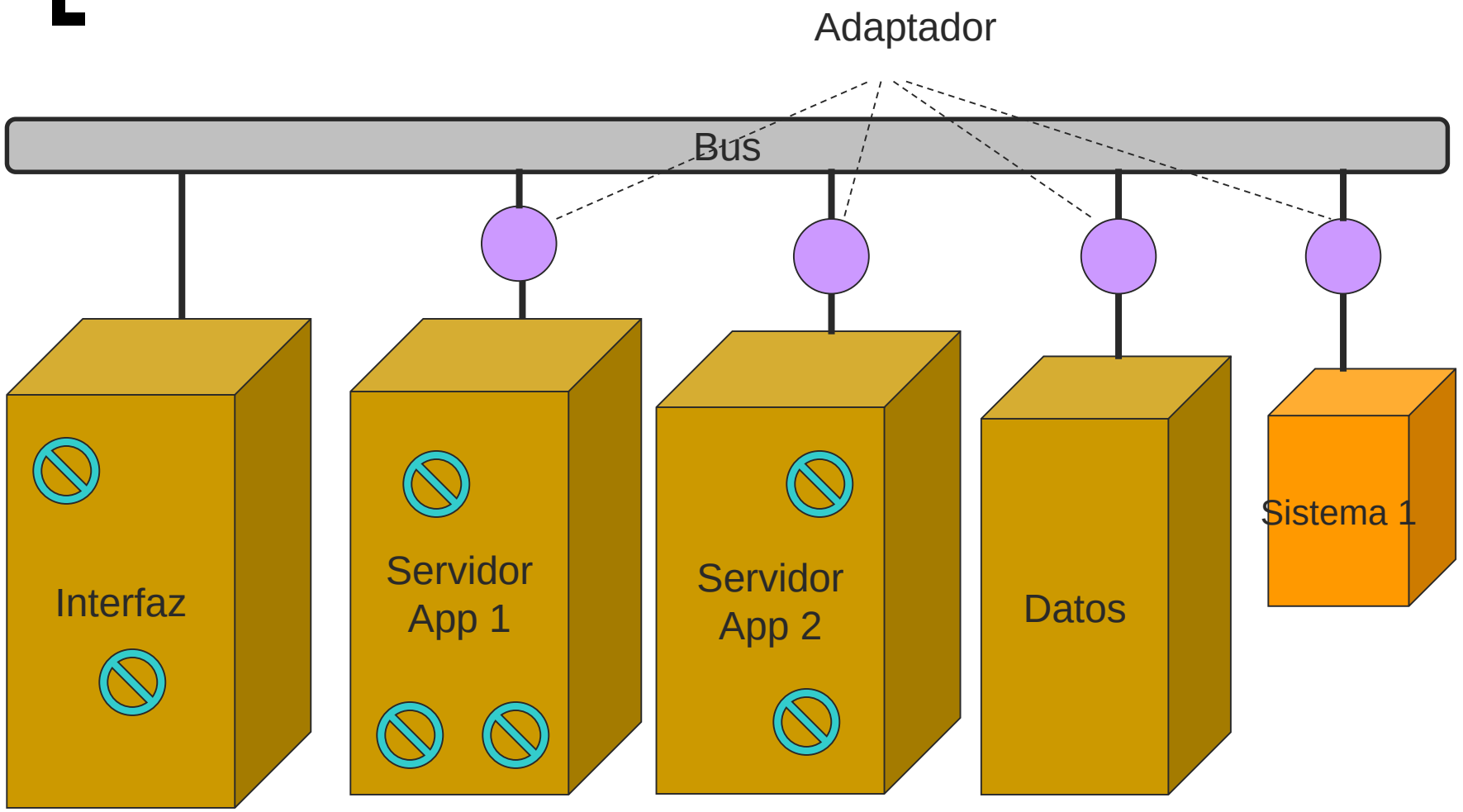
Arquitectura Distribuida (n capas)



Arquitectura Distribuida (con objetos y CORBA)



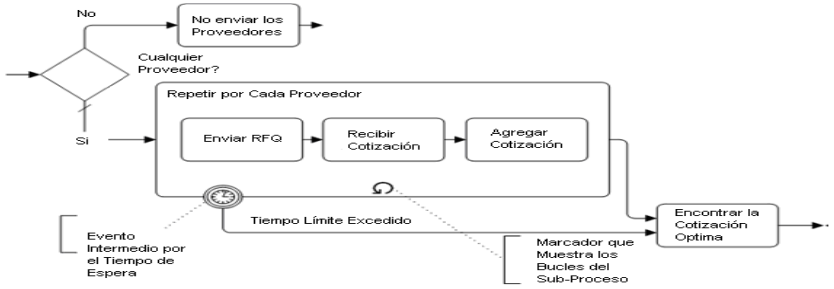
Integración de aplicaciones a través de un bus

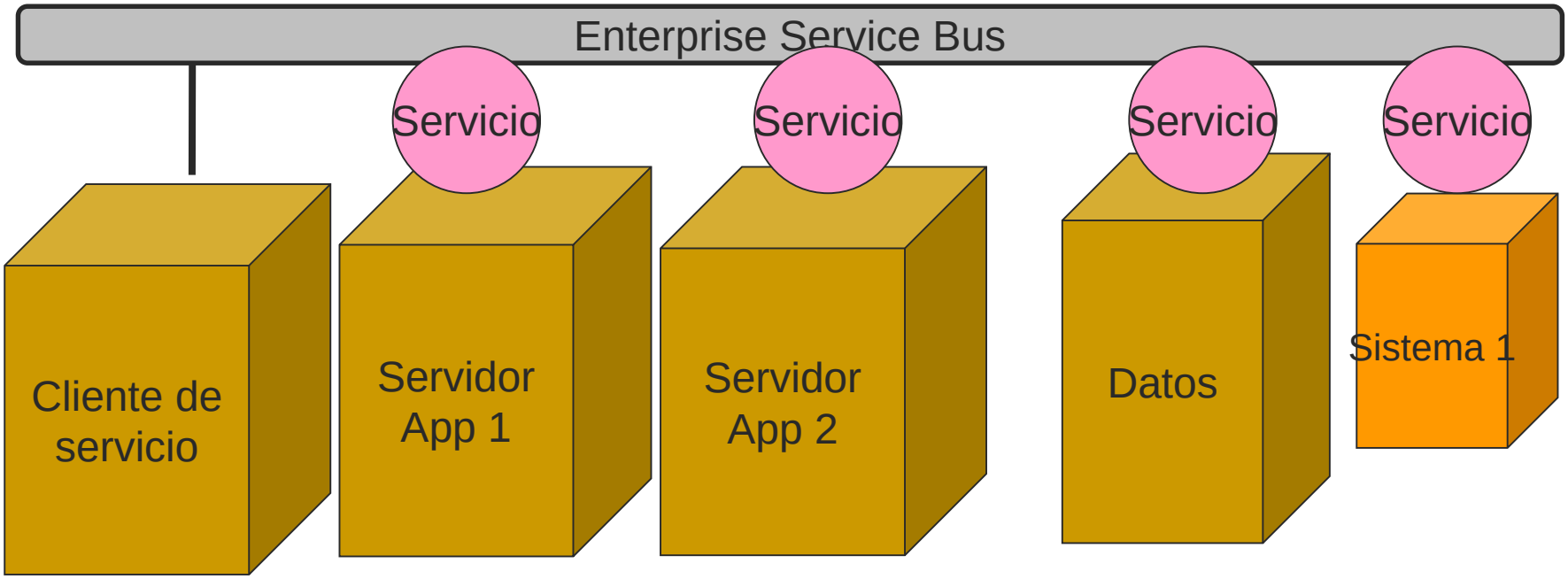


Inconvenientes de la integración punto a punto

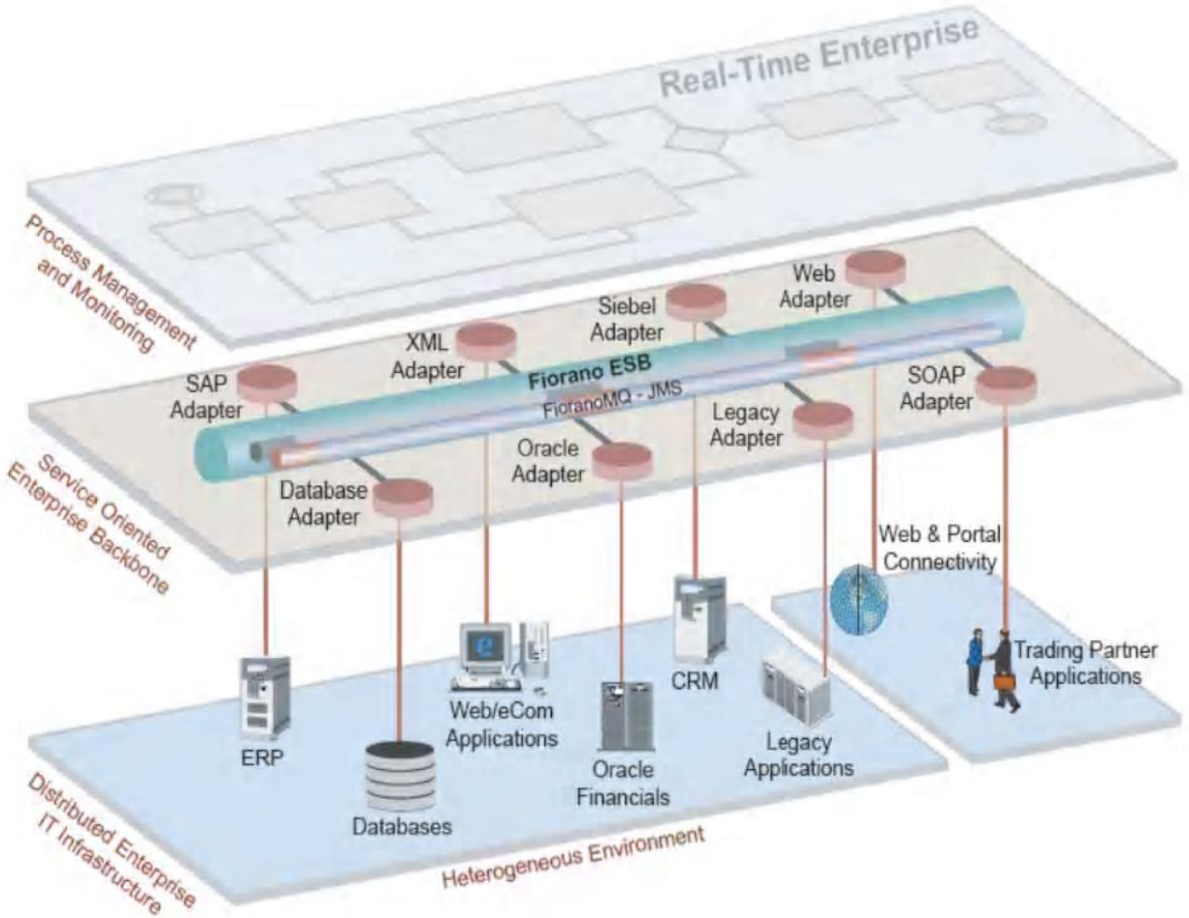
- Arquitectura accidental (sincrona, de grano fino y no escalable)
- Dificulta los cambios: nuevos canales, nuevos productos.
- Mantenimiento lento y caro
- Costoso de gestionar, monitorear y extender

Tendencia a modelar procesos de negocios





Esquema general de SOA



Definiciones de Sistemas Distribuidos

- Tanenbaum:

A distributed system is a collection of independent computers that appears to its users as a single coherent system

- Colouris:

A distributed system is one in which components located at networked computers communicate and coordinate their actions only by passing messages

Arquitectura de dos capas

- Surge en los 90 como solución para potenciar el poder de las pc’s y concentrar funcionalidad a nivel de “servidor”
- La arquitectura Cliente/Servidor constituyo uno de los primeros paradigmas “distribuidos”

Ventajas del Cliente/Servidor

- Aumento del poder de las PC reduciendo costos
- Permite que el procesamiento resida cerca de la fuente de datos reduciendo el trafico de red
- Facilita el uso de GUI
- Acepta el desafio de los sistemas abiertos
- Favorece el diseño modular

Desventajas del Cliente/Servidor

- Un desbalanceo de tareas entre cliente y servidor puede provocar cuellos de botella
- El desarrollo de aplicaciones es mas complejo creciendo también la complejidad de las herramientas de desarrollo y programación

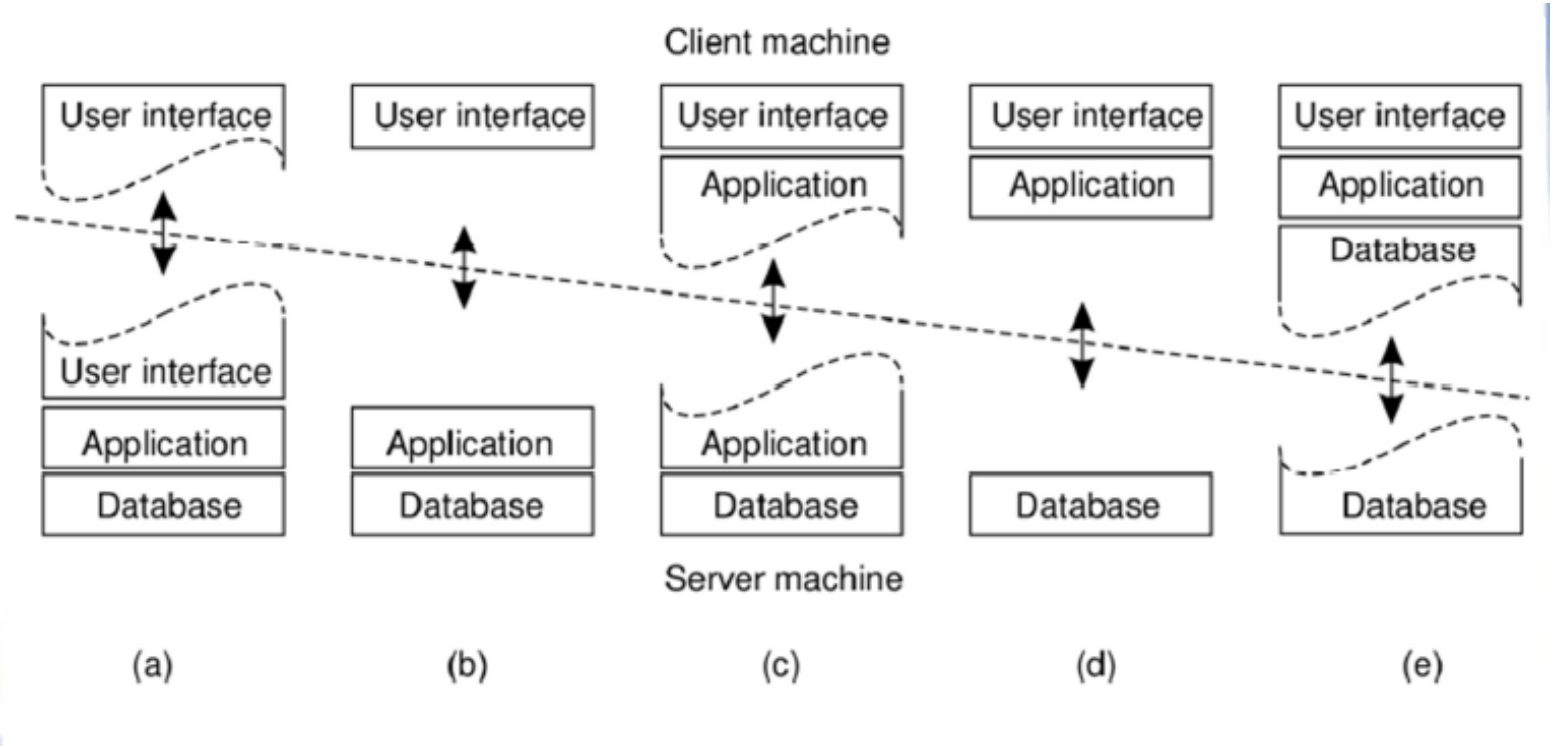
¿Qué es Cliente/Servidor?

- Servicio
- Recursos compartidos.
- Protocolos asimétricos
- Transparencia de la ubicación
- Intercambios basados en mensajes
- Encapsulamiento de servicios
- Escalabilidad
- Integridad

Variantes de Cliente/Servidor

- File Servers
- Database Servers
- Transaction Servers
- GroupWare Servers
- Object Servers

Aplicación en dos niveles



¿Que es MiddleWare?

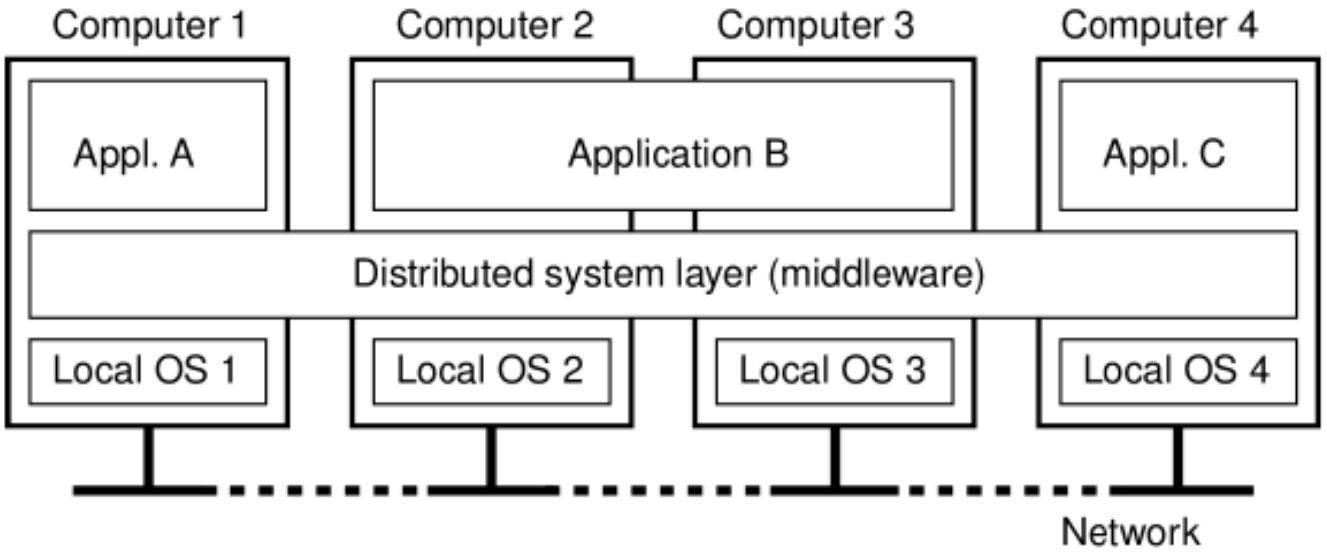
El MiddleWare comienza con la porción de cliente que es usada para invocar el servicio, cubre la transmisión sobre la red y retorna el resultado.

Existen dos grandes clases de middleware:

- General
- Especifico del servicio

Objetivo del Middleware

- Computadoras independientes
- Sistema único



Middleware para Comunicación

Componente Middleware

Corre en ambos lados de la aplicación, el lado del cliente y el del servidor. Dividimos esta componente en tres categorías:

- Protocolo de transporte
- Sistema operativo de red
- Especifico del servicio

El modelo OSI y el middleware de transporte

El modelo OSI constituye un estándar de construcción de software de redes que se fundamenta en la definición de niveles. La definición de software por niveles es una técnica que ayuda a dividir la complejidad de los programas en una jerarquía de servicios.

Tres conceptos claves

- En cada nivel, los pares cliente y servidor cooperan para proveer un servicio. El **protocolo** es el contrato entre los pares, especifica como se divide el trabajo, la semántica de intercambio de mensajes y las secuencias de saludos (“handshake”).
- Cada nivel construye los servicios que proveera al nivel superior. La **interface de servicios** especifica como un nivel obtiene el acceso a los servicios del nivel inferior.
- Los **servicios** son más abstractos a medida que se sube de nivel.

Niveles del modelo OSI

- Nivel 1 o nivel físico
- Nivel 2 o nivel de enlace
- Nivel 3 o nivel de red
- Nivel 4 o nivel de transporte
- Nivel 5 o nivel de sesión
- Nivel 6 o nivel de presentación
- Nivel 7 o nivel de aplicación

Middleware específico del servicio

El ambiente de programación RPC

El nacimiento del RPC corresponde al año 84 y especifica una manera diferente de trabajar sobre las redes que provoque el mismo efecto que se obtiene con la programación en un ambiente único con la idea de programa y subrutina. La idea nueva es obtener aplicaciones que funcionen exactamente igual en un ambiente centralizado que en un ambiente distribuido

Características del RPC

- Pasaje de parámetros
- Binding
- Tratamiento de excepciones
- Trasparencia de la representación de datos

Mensajes y Colas

- MOM (Message-Oriented Middleware) permite intercambio de mensajes de cualquier tipo entre cliente y servidor donde dichos mensajes son encolados. Las aplicaciones se comunican poniendo mensajes en una cola de mensajes.

MOM. Características

- Oculta totalmente los detalles físicos de la red y la comunicación se realiza sin saber si existe un enlace
- Los programas no hablan unos con otros en forma sincrónica.

MOM. Características (cont.)

- Muchos productos de MOM disponen de APIs que corren sobre múltiples sistemas operativos y plataformas y que proveen colas de mensajes persistentes y no persistentes.
- Muchos productos de manejo de mensajes proveen un mínimo de tolerancia a fallas proveyendo algún mecanismo de protección transaccional permitiendo que las colas participen de un protocolo de sincronización de two-face commit.

MOM vs. RPC

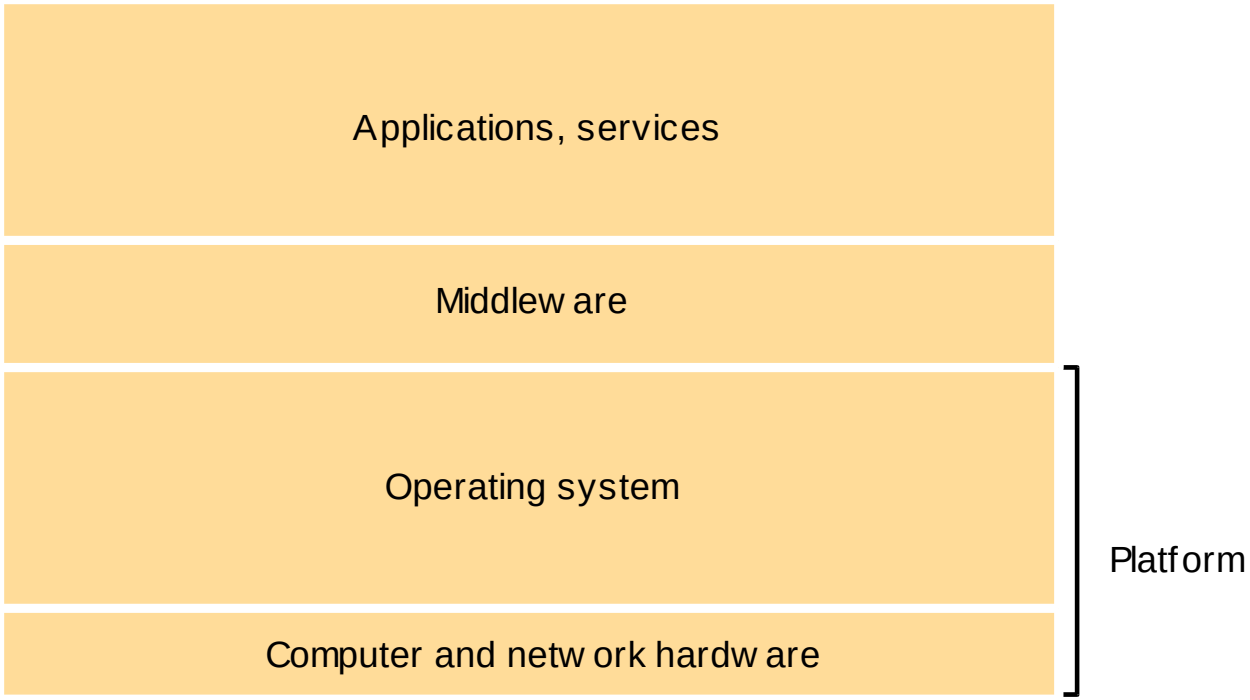
Característica	MOM	RPC
Metafora	Como oficina postal	Como telefono
Tiempo de relacion C/S	Asincronico. Clientes y servidores operan en distinto tiempo	Sincronico. Clientes y servidores corren a la vez y se esperan
Secuenciamiento C/S	Sin secuencia fija	Los servidores se lanzan antes que los clientes
Estilo	Cola	Call/Return
Datos persistentes	SI	NO
Compañero disponible	NO	SI
Balanceo de carga	Una cola simple FIFO o con prioridades	Requiere un monitor de TP adicional
Soporte transaccional	SI (algunos productos)	NO (requiere RPC transaccional)
Flitrado de mensajes	SI	NO
Performance	Baja	Alta
Procesamiento asincrónico	SI (se requiere colas y triggers)	Limitado (requiere hilos y trucos de programación)

Arquitectura distribuida

A grandes rasgos un arquitectura distribuida consiste de cuatro niveles:

- Nivel de aplicación
- Nivel de sistema operativo distribuido
- Nivel de manejo local
- Nivel del sistema de comunicación

Niveles de la Arquitectura



El protocolo transporte del middleware

<table>
<tr><td rowspan="2">APLICACIÓN</td><td rowspan="2">MOM</td><td colspan="2">RPC</td><td>Peer-to-Peer</td></tr>
<tr><td>Named Pipes</td><td>DCE</td><td></td></tr>
<tr><td>PRESENTACION</td><td></td><td colspan="2">NDR</td><td colspan="2">SXDR</td></tr>
<tr><td>SESION</td><td colspan="5">NETBIOS / SOCKETS / TLI</td></tr>
<tr><td>TRANSPORTE</td><td rowspan="3">RED</td><td colspan="2" rowspan="2">TCP/IP</td><td colspan="2" rowspan="2">IPX/SPX</td></tr>
<tr><td>RED</td></tr>
<tr><td>LLC</td><td colspan="4">802.2</td></tr>
<tr><td>MAC</td><td colspan="2">TOKEN RING/ETHERNET</td><td colspan="3">SDLC</td></tr>
<tr><td>FISICA</td><td>FIBRA OPTICA</td><td>COAXIL</td><td>TELEFONO</td><td></td></tr>
</table>

Servicios de seguridad distribuidos

El ambiente cliente/servidor introduce un nuevo tratamiento de la seguridad respecto del que se tenía en los sistemas tradicionales.

En un ambiente cliente/servidor no se puede asegurar que cualquier sistema operativo sobre la red proteja los recursos del servidor de acceso no autorizados y aunque se contara con clientes totalmente seguros, la red en sí es un medio altamente violable e inseguro.

Nivel C2 de seguridad

- **Autenticación:** (soy quien digo ser).
- **Autorización:** (sólo uso lo que me es permitido)
- **Auditoría:** (qué hice)

Lecturas recomendadas

- Libro de cátedra “Aplicaciones, servicios y procesos distribuidos. Una visión para la construcción de software”.
- “Distributed Systems Concepts and Designs” Fifth Edition 2012. George Colouris, Jean Dollimore, Tim Kindberg. Addison Wesley.
- “Distributed Systems Principles and Paradigms”. Andrew Tanenbaum, Maarten Van Steen. Pearson Edition. 2014.