Aplicaciones Distribuidas

Introducción a los Sistemas Distribuidos

Edwin Salvador

7 de octubre de 2015

Sesión 1

Contenido I

Contenido II

Anuncios

- Cuenta de peerwise creada. Ingresar a `https://peerwise.cs.auckland.ac.nz`
- ID del curso: 12072
- Para la siguiente semana 5 preguntas hechas y 5 respondidas sobre el tema de hoy.
- Cuenta de Turnitin creada. Entregar deberes de consultas por este medio.

Contenido I

Contenido II

Contenido I

Contenido II

Tipos de Sistemas

- **Personales (PC)** Sistemas de coste reducido dedicados a un único usuario que poseen sus recursos locales. Hoy en día estos sistemas soportan multitarea y existen versiones móviles.
- **En Red** Un conjunto de computadores espacialmente separados e interconectados que intercambian mensajes basados en protocolos específicos. Las computadoras son identificadas mediante direcciones IP.
- **Distribuidos** Múltiples computadoras en la red trabajando juntas como un sistema único. La separación espacial y aspectos de comunicación están escondidos de los usuarios desaparece el concepto de local/remoto. Proporcionan de forma transparente la compartición de recursos.

Contenido I

Contenido II

Definición de un Sistema Distribuido

- Una colección de computadores independientes que ofrecen a sus usuarios una visión de sistema único [Tanenbaum].
- Un sistema distribuido es un sistema que puede fallar cuando una máquina que ni sabias que existía deja de funcionar.

Distribuido vs Red

- Ambos consisten en computadores interconectados
- En los sistemas distribuidos los computadores comparten un estado (estado global), esto los diferencia de los sistemas en red donde los computadores tienen estados independientes.
- Este estado único permite la visión de sistema único.
- Los SD ocultan la ubicación de los recursos.

¿Cómo se oculta la distribución de recursos?

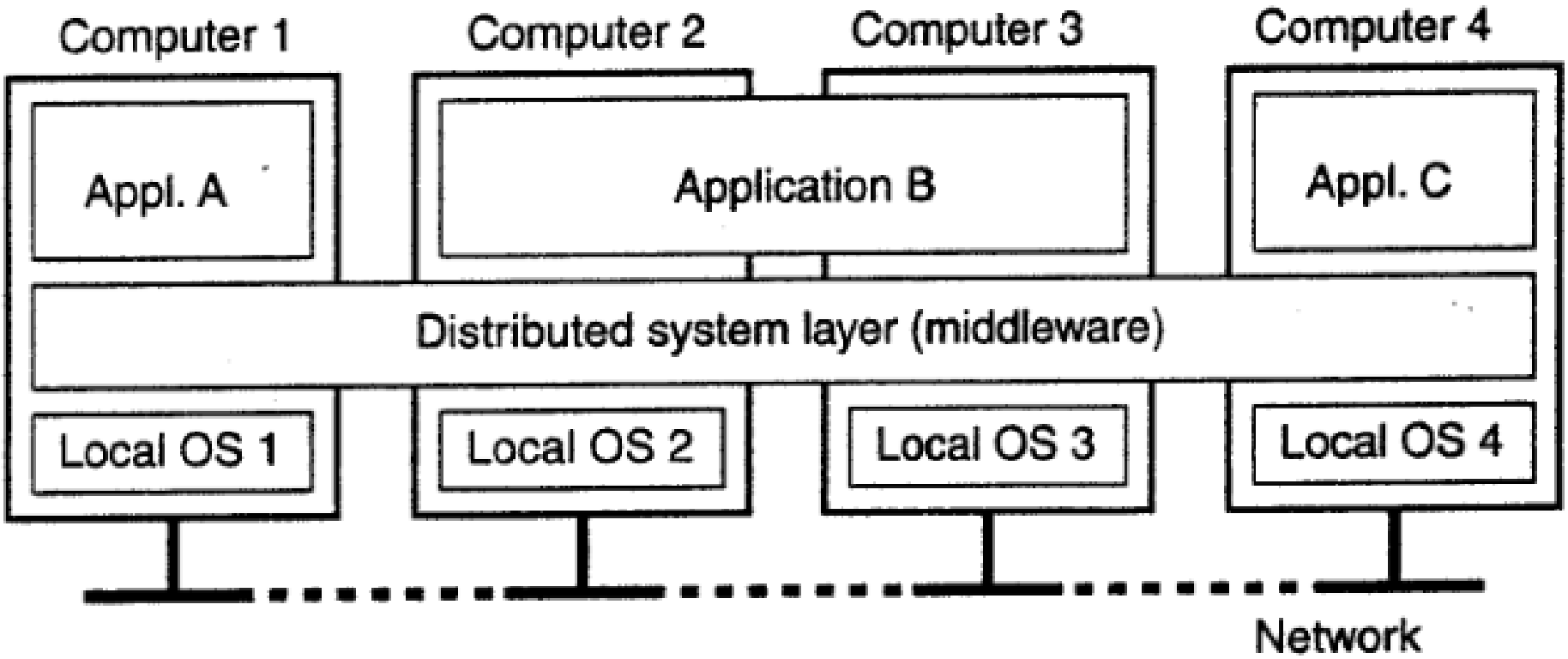


Figura: El middleware ofrece una misma interfaz a cada aplicación

Se oculta la heterogeniedad de componentes hardware y software.

Contenido I

Contenido II

Contenido I

Contenido II

- En un sistema distribuido las computadoras llevan a cabo sus tareas independientemente y se comunican con otras computadoras por medio del paso de mensajes cuando lo necesitan.
- Los servicios proveídos por un sistema distribuido serán accedidos por múltiples usuarios simultáneamente.
- El diseño de un sistema distribuido debe tomar en consideración e implementar técnicas apropiadas para manejar la concurrencia.

Contenido I

Contenido II

Inexistencia de reloj global

Dificultades de un Sistema Distribuido

- Cuando las aplicaciones deben cooperar entre ellas lo hacen mediante el paso de mensajes.
- Esta coordinación a veces depende del tiempo en el que ha realizado alguna acción un programa.
- Los relojes en las computadoras operan independientemente.
- Existen límites en la precisión con la que las computadoras en una red pueden sincronizar sus relojes.
- Por lo tanto, un sistema distribuido debe lidiar con el hecho de no tener un reloj global e implementar técnicas de sincronización cuando sea necesario.

Contenido I

Contenido II

Fallos independientes

Dificultades de un Sistema Distribuido

- Algunos componentes del sistema pueden fallar mientras otras continúan funcionando.
- Los fallos de las computadoras que conforman el sistema no son conocidos por otras computadoras inmediatamente.
- No es posible detectar cuando la red ha fallado o está excesivamente lenta.

Contenido I

Contenido II

Busqueda web

Ejemplos de Sistemas distribuidos

- 10 billones de busquedas/mes.

Busqueda web

Ejemplos de Sistemas distribuidos

- 10 billones de busquedas/mes.
- Indexar el contenido entero de la WWW (páginas, multimedia, libros)

Busqueda web

Ejemplos de Sistemas distribuidos

- 10 billones de busquedas/mes.
- Indexar el contenido entero de la WWW (páginas, multimedia, libros)
- 63 billones de páginas, 1 trillón de direcciones web.

Busqueda web

Ejemplos de Sistemas distribuidos

- 10 billones de busquedas/mes.
- Indexar el contenido entero de la WWW (páginas, multimedia, libros)
- 63 billones de páginas, 1 trillón de direcciones web.
- Google posee uno de los SD más grandes y complejos.

Busqueda web

Ejemplos de Sistemas distribuidos

- 10 billones de busquedas/mes.
- Indexar el contenido entero de la WWW (páginas, multimedia, libros)
- 63 billones de páginas, 1 trillón de direcciones web.
- Google posee uno de los SD más grandes y complejos.
  - Data centers alrededor del mundo.

Busqueda web

Ejemplos de Sistemas distribuidos

- 10 billones de busquedas/mes.
- Indexar el contenido entero de la WWW (páginas, multimedia, libros)
- 63 billones de páginas, 1 trillón de direcciones web.
- Google posee uno de los SD más grandes y complejos.
  - Data centers alrededor del mundo.
  - Un sistema de archivos distribuido que soporta archivos pesados optimizado para lecturas.

Busqueda web

Ejemplos de Sistemas distribuidos

- 10 billones de busquedas/mes.
- Indexar el contenido entero de la WWW (páginas, multimedia, libros)
- 63 billones de páginas, 1 trillón de direcciones web.
- Google posee uno de los SD más grandes y complejos.
  - Data centers alrededor del mundo.
  - Un sistema de archivos distribuido que soporta archivos pesados optimizado para lecturas.
  - Sistema distribuido de almacenamiento con acceso rápido a recursos.

Busqueda web

Ejemplos de Sistemas distribuidos

- 10 billones de busquedas/mes.
- Indexar el contenido entero de la WWW (páginas, multimedia, libros)
- 63 billones de páginas, 1 trillón de direcciones web.
- Google posee uno de los SD más grandes y complejos.
  - Data centers alrededor del mundo.
  - Un sistema de archivos distribuido que soporta archivos pesados optimizado para lecturas.
  - Sistema distribuido de almacenamiento con acceso rápido a recursos.
  - Un servicio de bloqueo distribuido.

Busqueda web

Ejemplos de Sistemas distribuidos

- 10 billones de busquedas/mes.
- Indexar el contenido entero de la WWW (páginas, multimedia, libros)
- 63 billones de páginas, 1 trillón de direcciones web.
- Google posee uno de los SD más grandes y complejos.
  - Data centers alrededor del mundo.
  - Un sistema de archivos distribuido que soporta archivos pesados optimizado para lecturas.
  - Sistema distribuido de almacenamiento con acceso rápido a recursos.
  - Un servicio de bloqueo distribuido.
  - Modelo de programación que soporta computación paralela y distribuida.

Juegos multiplayer masivos online

- Muchos jugadores (50 000) interactuan en Internet con un mundo virtual consistente.

Juegos multiplayer masivos online

- Muchos jugadores (50 000) interactuan en Internet con un mundo virtual consistente.
- Necesitan respuesta rápida, buena experiencia de usuarios.

Juegos multiplayer masivos online

- Muchos jugadores (50 000) interactuan en Internet con un mundo virtual consistente.
- Necesitan respuesta rápida, buena experiencia de usuarios.
- Eventos en tiempo real, vista consistente del mundo.

Juegos multiplayer masivos online

- Muchos jugadores (50 000) interactuan en Internet con un mundo virtual consistente.
- Necesitan respuesta rápida, buena experiencia de usuarios.
- Eventos en tiempo real, vista consistente del mundo.
- Modelo cliente-servidor. Un servidor central maneja el estado del mundo virtual y los clientes acceden a este.

Juegos multiplayer masivos online

- Muchos jugadores (50 000) interactuan en Internet con un mundo virtual consistente.
- Necesitan respuesta rápida, buena experiencia de usuarios.
- Eventos en tiempo real, vista consistente del mundo.
- Modelo cliente-servidor. Un servidor central maneja el estado del mundo virtual y los clientes acceden a este.
  - El servidor consiste en un cluster con cientos de nodos.

Juegos multiplayer masivos online

- Muchos jugadores (50 000) interactuan en Internet con un mundo virtual consistente.
- Necesitan respuesta rápida, buena experiencia de usuarios.
- Eventos en tiempo real, vista consistente del mundo.
- Modelo cliente-servidor. Un servidor central maneja el estado del mundo virtual y los clientes acceden a este.
  - El servidor consiste en un cluster con cientos de nodos.
  - La carga de trabajo se distribuye entre los nodos del servidor.

Juegos multiplayer masivos online

- Muchos jugadores (50 000) interactuan en Internet con un mundo virtual consistente.
- Necesitan respuesta rápida, buena experiencia de usuarios.
- Eventos en tiempo real, vista consistente del mundo.
- Modelo cliente-servidor. Un servidor central maneja el estado del mundo virtual y los clientes acceden a este.
  - El servidor consiste en un cluster con cientos de nodos.
  - La carga de trabajo se distribuye entre los nodos del servidor.
- Otros adoptan una arquitectura más distribuida, con varios servidores distribuidos geograficamente. Más escalable.

Juegos multiplayer masivos online

- Muchos jugadores (50 000) interactuan en Internet con un mundo virtual consistente.
- Necesitan respuesta rápida, buena experiencia de usuarios.
- Eventos en tiempo real, vista consistente del mundo.
- Modelo cliente-servidor. Un servidor central maneja el estado del mundo virtual y los clientes acceden a este.
  - El servidor consiste en un cluster con cientos de nodos.
  - La carga de trabajo se distribuye entre los nodos del servidor.
- Otros adoptan una arquitectura más distribuida, con varios servidores distribuidos geograficamente. Más escalable.
- Otra opción es la arquitectura peer-to-peer. Cada participante colabora con recursos (almacenamiento y procesamiento) para hacer posible el juego.

Mercados financieros

- La industria de las finanzas siempre ha estado al día con los SD.

Mercados financieros

- La industria de las finanzas siempre ha estado al día con los SD.
- Acceso en tiempo real, monitorea automático y aplicaciones de mercadeo.

Mercados financieros

- La industria de las finanzas siempre ha estado al día con los SD.
- Acceso en tiempo real, monitorea automático y aplicaciones de mercadeo.
- Mantener al día sobre eventos a todos los clientes que han mostrado interes en un item del mercado (caida del precio del petroleo o de accione de una empresa).

Mercados financieros

- La industria de las finanzas siempre ha estado al día con los SD.
- Acceso en tiempo real, monitorea automático y aplicaciones de mercadeo.
- Mantener al día sobre eventos a todos los clientes que han mostrado interes en un item del mercado (caida del precio del petroleo o de accione de una empresa).
- Mantienen una arquitectura basada en eventos.

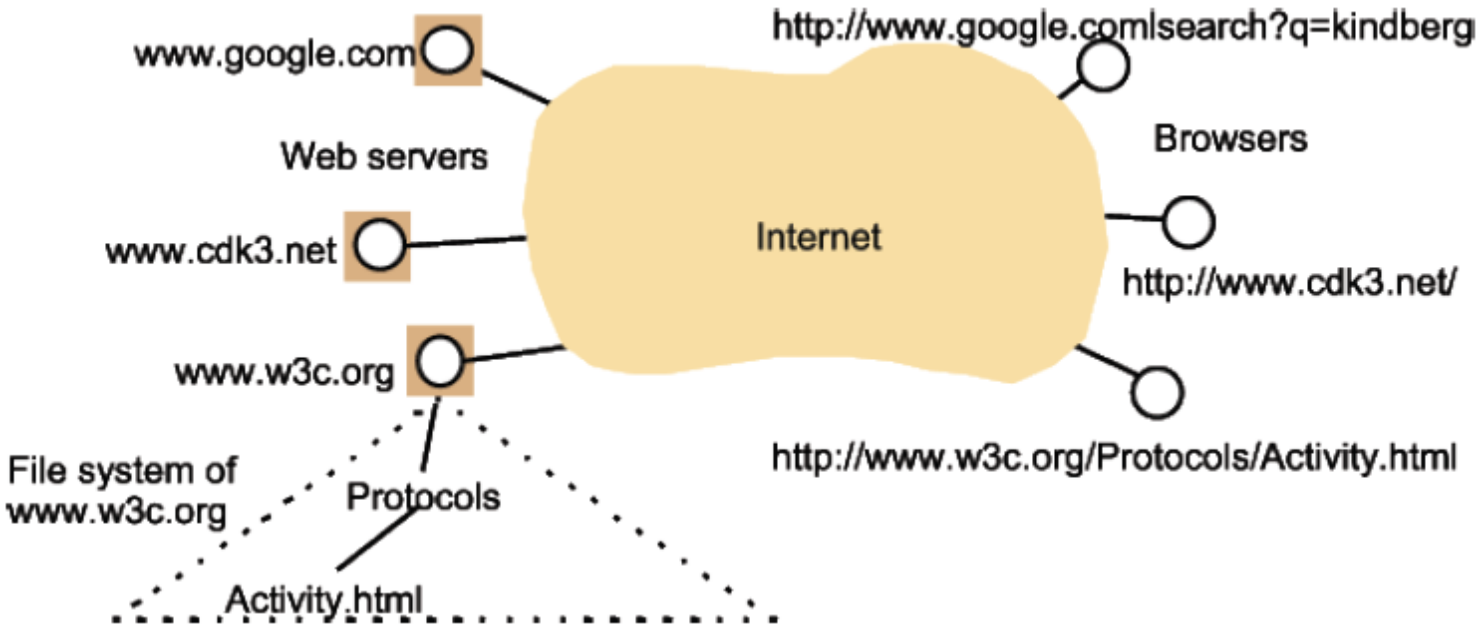


Figura: Servidores y clientes web

Dominios de aplicación y aplicaciones asociadas

- **Finanzas y comercio** Comercio electrónico (Amazon, eBay, PayPal), banca en linea
- **La sociedad de la información** Información web, motores de búsqueda, ebooks, Wikipedia; redes sociales
- **Industrias creativas y entretenimiento** Juegos en linea, música, cine (Youtube, Flickr, Netflix)
- **Salud** registro de pacientes en línea, monitoreo de pacientes.
- **Educación** e-learning, ambientes de educación virtual, educación a distancia.
- **Transporte y logística** GPS, servicios de mapas.
- **Ciencia** Colaboración de científicos en línea
- **Gestión ambiental** Tecnología de sensores para monitorear terremotos, tsunamis, etc.

Contenido I

Contenido II

Objetivos

- **Alto rendimiento** Procesamiento y memoria distribuida mediante clusters de computadores.
- **Tolerancia a fallos** Un sistema bancario necesita replicar la información sobre las cuentas de los clientes. Perder información es inaceptable. Muchas BDD proveen sistemas de replicación transparentes.
- **Alta disponibilidad** Ciertas aplicaciones necesitan que la info esté cerca del usuario y disminuir el tiempo de acceso. Técnicas de replicación tienen en cuenta la ubicación geográfica. Sistemas peer-to-peer son altamente escalables y y evitan cuellos de botella.
- **Movilidad y ubicuidad** Recursos inherentemente distribuidos, el usuario se mueve en un entorno con recursos.

Contenido I

Contenido II

Desafíos

- Los ejemplos de aplicaciones distribuidas presentadas anteriormente presentan algunos desafíos que deben ser tomados en cuenta al momento de diseñar una aplicación.
- Los principales desafíos son:
  - Heterogeneidad
  - Extensibilidad (apertura o flexibilidad)
  - Seguridad
  - Escalabilidad
  - Tratamiento de fallos
  - Concurrencia
  - Transparencia

Contenido I

Contenido II

Heterogeneidad

- Los sistemas distribuidos usan hardware y software de una vasta variedad de características (recursos heterogéneos). **Qué recursos podrían generar problemas a un SD?**

Heterogeneidad

- Los sistemas distribuidos usan hardware y software de una vasta variedad de características (recursos heterogéneos). **Qué recursos podrían generar problemas a un SD?**
  - Redes

Heterogeneidad

- Los sistemas distribuidos usan hardware y software de una vasta variedad de características (recursos heterogéneos). **Qué recursos podrían generar problemas a un SD?**
  - Redes
  - Hardware de computadoras

Heterogeneidad

- Los sistemas distribuidos usan hardware y software de una vasta variedad de características (recursos heterogéneos). **Qué recursos podrían generar problemas a un SD?**
  - Redes
  - Hardware de computadoras
  - Sistemas operativos

Heterogeneidad

- Los sistemas distribuidos usan hardware y software de una vasta variedad de características (recursos heterogéneos). **Qué recursos podrían generar problemas a un SD?**
  - Redes
  - Hardware de computadoras
  - Sistemas operativos
  - Lenguajes de programación

Heterogeneidad

- Los sistemas distribuidos usan hardware y software de una vasta variedad de características (recursos heterogéneos). **Qué recursos podrían generar problemas a un SD?**
  - Redes
  - Hardware de computadoras
  - Sistemas operativos
  - Lenguajes de programación
  - Implementaciones por distintos desarrolladores.

Heterogeneidad

- Los sistemas distribuidos usan hardware y software de una vasta variedad de características (recursos heterogéneos). **Qué recursos podrían generar problemas a un SD?**
  - Redes
  - Hardware de computadoras
  - Sistemas operativos
  - Lenguajes de programación
  - Implementaciones por distintos desarrolladores.
- Algunas medidas para menejar este problema de heterogeneidad son:

Heterogeneidad

- Los sistemas distribuidos usan hardware y software de una vasta variedad de características (recursos heterogéneos). **Qué recursos podrían generar problemas a un SD?**
  - Redes
  - Hardware de computadoras
  - Sistemas operativos
  - Lenguajes de programación
  - Implementaciones por distintos desarrolladores.
- Algunas medidas para menejar este problema de heterogeneidad son:
  - Utilizar protocolos estándares

Heterogeneidad

- Los sistemas distribuidos usan hardware y software de una vasta variedad de características (recursos heterogéneos). **Qué recursos podrían generar problemas a un SD?**
  - Redes
  - Hardware de computadoras
  - Sistemas operativos
  - Lenguajes de programación
  - Implementaciones por distintos desarrolladores.
- Algunas medidas para menejar este problema de heterogeneidad son:
  - Utilizar protocolos estándares
  - Utilizar formatos de mensajes y tipos de datos previamente acordados.

Heterogeneidad

- Los sistemas distribuidos usan hardware y software de una vasta variedad de características (recursos heterogéneos). **Qué recursos podrían generar problemas a un SD?**
  - Redes
  - Hardware de computadoras
  - Sistemas operativos
  - Lenguajes de programación
  - Implementaciones por distintos desarrolladores.
- Algunas medidas para menejar este problema de heterogeneidad son:
  - Utilizar protocolos estándares
  - Utilizar formatos de mensajes y tipos de datos previamente acordados.
  - Apegarse a una Interfaz de Programa de Aplicación (API) aceptado.

Heterogeneidad

- Los sistemas distribuidos usan hardware y software de una vasta variedad de características (recursos heterogéneos). **Qué recursos podrían generar problemas a un SD?**
  - Redes
  - Hardware de computadoras
  - Sistemas operativos
  - Lenguajes de programación
  - Implementaciones por distintos desarrolladores.
- Algunas medidas para menejar este problema de heterogeneidad son:
  - Utilizar protocolos estándares
  - Utilizar formatos de mensajes y tipos de datos previamente acordados.
  - Apegarse a una Interfaz de Programa de Aplicación (API) aceptado.
  - Utilizar un Middleware

Heterogeneidad

- Los sistemas distribuidos usan hardware y software de una vasta variedad de características (recursos heterogéneos). **Qué recursos podrían generar problemas a un SD?**
  - Redes
  - Hardware de computadoras
  - Sistemas operativos
  - Lenguajes de programación
  - Implementaciones por distintos desarrolladores.
- Algunas medidas para menejar este problema de heterogeneidad son:
  - Utilizar protocolos estándares
  - Utilizar formatos de mensajes y tipos de datos previamente acordados.
  - Apegarse a una Interfaz de Programa de Aplicación (API) aceptado.
  - Utilizar un Middleware
  - Utilizar código portable.

Middleware

- Un middleware oculta los problemas de heterogeneidad que sufren los sistemas distribuidos (SO y hardware)

Middleware

- Un middleware oculta los problemas de heterogeneidad que sufren los sistemas distribuidos (SO y hardware)
- En **servicios de comunicación** provee transparencia de acceso.

Middleware

- Un middleware oculta los problemas de heterogeneidad que sufren los sistemas distribuidos (SO y hardware)
- En **servicios de comunicación** provee transparencia de acceso.
- **Servicios de nombre** permite que recursos remotos sean buscados por nombre, similar a la búsqueda de directorios.

Middleware

- Un middleware oculta los problemas de heterogeneidad que sufren los sistemas distribuidos (SO y hardware)
- En **servicios de comunicación** provee transparencia de acceso.
- **Servicios de nombre** permite que recursos remotos sean buscados por nombre, similar a la búsqueda de directorios.
- **Almacenamiento** provee facilidades para persistencia de datos.

Middleware

- Un middleware oculta los problemas de heterogeneidad que sufren los sistemas distribuidos (SO y hardware)
- En **servicios de comunicación** provee transparencia de acceso.
- **Servicios de nombre** permite que recursos remotos sean buscados por nombre, similar a la búsqueda de directorios.
- **Almacenamiento** provee facilidades para persistencia de datos.
- **Transacciones distribuidas** habilidad de roll back. Si una operación falla todos los datos permanecen sin cambios.

Middleware

- Un middleware oculta los problemas de heterogeneidad que sufren los sistemas distribuidos (SO y hardware)
- En **servicios de comunicación** provee transparencia de acceso.
- **Servicios de nombre** permite que recursos remotos sean buscados por nombre, similar a la búsqueda de directorios.
- **Almacenamiento** provee facilidades para persistencia de datos.
- **Transacciones distribuidas** habilidad de roll back. Si una operación falla todos los datos permanecen sin cambios.
- **Seguridad** provee protección en contra de varios tipos de amenazas.

Middleware

- Un middleware oculta los problemas de heterogeneidad que sufren los sistemas distribuidos (SO y hardware)
- En **servicios de comunicación** provee transparencia de acceso.
- **Servicios de nombre** permite que recursos remotos sean buscados por nombre, similar a la búsqueda de directorios.
- **Almacenamiento** provee facilidades para persistencia de datos.
- **Transacciones distribuidas** habilidad de roll back. Si una operación falla todos los datos permanecen sin cambios.
- **Seguridad** provee protección en contra de varios tipos de amenazas.
- Ejemplos: CORBA, Java RMI, RPC.

Heterogeneidad y código descargable (móvil)

- Un código móvil o descargable puede ser enviado de una máquina a otra y ser corrido en la máquina de destino. (JS, applets)
- Un código que es compilado en un SO no necesariamente puede correr en otro SO.
- Una medida para resolver esto son las máquinas virtuales que permiten que el código sea ejecutable en cualquier máquina. Un compilador produce código que será interpretado por la máquina virtual.

Contenido I

Contenido II

Extensibilidad (apertura o flexibilidad)

- La extensibilidad se refiere a la habilidad de extender el sistema de diferentes maneras añadiendo nuevos recursos de hardware o software.

Extensibilidad (apertura o flexibilidad)

- La extensibilidad se refiere a la habilidad de extender el sistema de diferentes maneras añadiendo nuevos recursos de hardware o software.
- Algunas medidas para superar este reto son:

Extensibilidad (apertura o flexibilidad)

- La extensibilidad se refiere a la habilidad de extender el sistema de diferentes maneras añadiendo nuevos recursos de hardware o software.
- Algunas medidas para superar este reto son:
  - Publicación de la interfaces claves (estándares para el desarrollo del sistema)

Extensibilidad (apertura o flexibilidad)

- La extensibilidad se refiere a la habilidad de extender el sistema de diferentes maneras añadiendo nuevos recursos de hardware o software.
- Algunas medidas para superar este reto son:
  - Publicación de la interfaces claves (estándares para el desarrollo del sistema)
  - Permitir un mecanismo de comunicación uniforme para comentar sobre las interfaces publicadas (*Request For Comments (RFC)*)

Extensibilidad (apertura o flexibilidad)

- La extensibilidad se refiere a la habilidad de extender el sistema de diferentes maneras añadiendo nuevos recursos de hardware o software.
- Algunas medidas para superar este reto son:
  - Publicación de la interfaces claves (estándares para el desarrollo del sistema)
  - Permitir un mecanismo de comunicación uniforme para comentar sobre las interfaces publicadas (*Request For Comments (RFC)*)
  - Asegurarse de que todas las implementaciones se adhieran a los estándares publicados.

Contenido I

Contenido II

Seguridad

- Muchos sistemas distribuidos manejan información sensible (datos de pacientes, cuentas bancarias, etc)

Seguridad

- Muchos sistemas distribuidos manejan información sensible (datos de pacientes, cuentas bancarias, etc)
- Existen tres aspectos de seguridad que se deben tomar en cuenta:

Seguridad

- Muchos sistemas distribuidos manejan información sensible (datos de pacientes, cuentas bancarias, etc)
- Existen tres aspectos de seguridad que se deben tomar en cuenta:
  - **Confidencialidad** Protección contra *acceso de individuos no autorizados*.

Seguridad

- Muchos sistemas distribuidos manejan información sensible (datos de pacientes, cuentas bancarias, etc)
- Existen tres aspectos de seguridad que se deben tomar en cuenta:
  - **Confidencialidad** Protección contra *acceso de individuos no autorizados*.
  - **Integridad** protección contra la *alteración y corrupción de la información*

Seguridad

- Muchos sistemas distribuidos manejan información sensible (datos de pacientes, cuentas bancarias, etc)
- Existen tres aspectos de seguridad que se deben tomar en cuenta:
  - **Confidencialidad** Protección contra *acceso de individuos no autorizados*.
  - **Integridad** protección contra la *alteración y corrupción de la información*
  - **Disponibilidad** protección contra *interferencias en asuntos de acceso a los recursos* (ataques de denegación de acceso).

Seguridad

- Muchos sistemas distribuidos manejan información sensible (datos de pacientes, cuentas bancarias, etc)
- Existen tres aspectos de seguridad que se deben tomar en cuenta:
  - **Confidencialidad** Protección contra *acceso de individuos no autorizados*.
  - **Integridad** protección contra la *alteración y corrupción de la información*
  - **Disponibilidad** protección contra *interferencias en asuntos de acceso a los recursos* (ataques de denegación de acceso).
- Mecanismos de seguridad:

Seguridad

- Muchos sistemas distribuidos manejan información sensible (datos de pacientes, cuentas bancarias, etc)
- Existen tres aspectos de seguridad que se deben tomar en cuenta:
  - **Confidencialidad** Protección contra *acceso de individuos no autorizados*.
  - **Integridad** protección contra la *alteración y corrupción de la información*
  - **Disponibilidad** protección contra *interferencias en asuntos de acceso a los recursos* (ataques de denegación de acceso).
- Mecanismos de seguridad:
  - Encriptación (Blowfish, RSA)

Seguridad

- Muchos sistemas distribuidos manejan información sensible (datos de pacientes, cuentas bancarias, etc)
- Existen tres aspectos de seguridad que se deben tomar en cuenta:
  - **Confidencialidad** Protección contra *acceso de individuos no autorizados*.
  - **Integridad** protección contra la *alteración y corrupción de la información*
  - **Disponibilidad** protección contra *interferencias en asuntos de acceso a los recursos* (ataques de denegación de acceso).
- Mecanismos de seguridad:
  - Encriptación (Blowfish, RSA)
  - Autentificación (passwords, public key authentication)

Seguridad

- Muchos sistemas distribuidos manejan información sensible (datos de pacientes, cuentas bancarias, etc)
- Existen tres aspectos de seguridad que se deben tomar en cuenta:
  - **Confidencialidad** Protección contra *acceso de individuos no autorizados*.
  - **Integridad** protección contra la *alteración y corrupción de la información*
  - **Disponibilidad** protección contra *interferencias en asuntos de acceso a los recursos* (ataques de denegación de acceso).
- Mecanismos de seguridad:
  - Encriptación (Blowfish, RSA)
  - Autentificación (passwords, public key authentication)
  - Autorización (listas de control de acceso)

Seguridad

- Existen ciertos retos de seguridad que no han logrado ser solucionados hasta hoy:
  - Ataques de denegación de servicio (quién conoce de que se trata?)

Seguridad

- Existen ciertos retos de seguridad que no han logrado ser solucionados hasta hoy:
  - Ataques de denegación de servicio (quién conoce de que se trata?)
  - Seguridad contra el código descargable (archivos ejecutables adjuntos en correos electrónicos)

Contenido I

Contenido II

Escalabilidad

- Qué es un sistema escalable?

Escalabilidad

- Qué es un sistema escalable?
  Un sistema es considerado escalable si puede manejar el crecimiento del número de usuarios sin que esto afecte su rendimiento.

Escalabilidad

- Qué es un sistema escalable?
  Un sistema es considerado escalable si puede manejar el crecimiento del número de usuarios sin que esto afecte su rendimiento.
- El Internet es un gran ejemplo de un sistema distribuido con un incremento importante de usuarios y recursos.

Date	*Computers*	*Web servers*	*Percentage*
1993, July	1,776,000	130	0.008
1995, July	6,642,000	23,500	0.4
1997, July	19,540,000	1,203,096	6
1999, July	56,218,000	6,598,697	12
2001, July	125,888,197	31,299,592	25
2003, July	~200,000,000	42,298,371	21
2005, July	353,284,187	67,571,581	19

Retos sobre la escalabilidad

Retos sobre la escalabilidad:

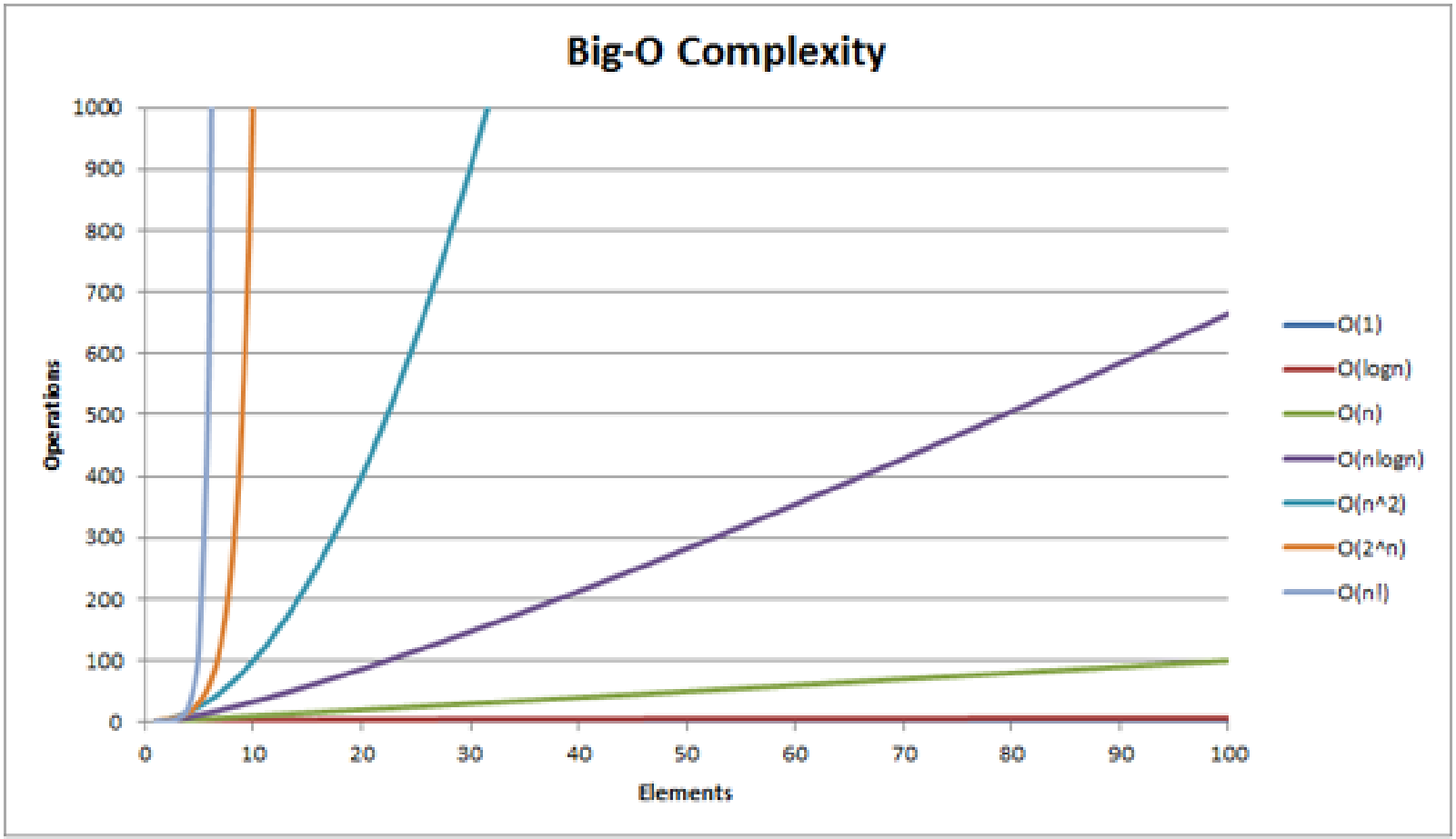
- **Costo de recursos físicos** Para que un sistema con n usuarios sea escalable, la cantidad de recursos requeridos para soportar el sistema debería ser $O(n)$. Si un servidor de archivos puede soportar 20 usuarios entonces dos servidores deberían soportar 40 usuarios.

Retos sobre la escalabilidad

Retos sobre la escalabilidad:

- **Costo de recursos físicos** Para que un sistema con n usuarios sea escalable, la cantidad de recursos requeridos para soportar el sistema debería ser $O(n)$. Si un servidor de archivos puede soportar 20 usuarios entonces dos servidores deberían soportar 40 usuarios.
- **Control de las pérdidas de prestaciones** La complejidad de los algoritmos utilizados para las búsquedas deberían ser escalables. Por ejemplo, un algoritmo de búsqueda para n entradas que requiere $O(log(n))$ pasos es escalable pero un que requiere n^2 no lo es.

Complejidad Big O



Retos sobre la escalabilidad

Retos sobre la escalabilidad:

- **Evitar que los recursos se agoten** Un sistema distribuido debe planificar el crecimiento a futuro y disponer de recursos suficientes para cubrir las demandas. Es muy difícil planificar la demanda de un servicio con años de anticipación. Un Ejemplo?

Retos sobre la escalabilidad

Retos sobre la escalabilidad:

- **Evitar que los recursos se agoten** Un sistema distribuido debe planificar el crecimiento a futuro y disponer de recursos suficientes para cubrir las demandas. Es muy difícil planificar la demanda de un servicio con años de anticipación. Un Ejemplo?
  las direcciones IP de Internet de 32-bit se están agotando y se requiere una nueva versión de 128-bits para cubrir la demanda actual.

Retos sobre la escalabilidad

Retos sobre la escalabilidad:

- **Evitar que los recursos se agoten** Un sistema distribuido debe planificar el crecimiento a futuro y disponer de recursos suficientes para cubrir las demandas. Es muy difícil planificar la demanda de un servicio con años de anticipación. Un Ejemplo?
  las direcciones IP de Internet de 32-bit se están agotando y se requiere una nueva versión de 128-bits para cubrir la demanda actual.
- **Evitar cuellos de botella en el rendimiento** se deben utilizar algoritmos descentralizados para evitar los cuellos de botella.

Contenido I

Contenido II

Tratamiento de fallos

- Debemos aceptar que los sistemas de computación pueden fallar.

Tratamiento de fallos

- Debemos aceptar que los sistemas de computación pueden fallar.
- En un sistema distribuido los fallos son parciales, es decir algún componente puede fallar mientras otros siguen funcionando.

Tratamiento de fallos

- Debemos aceptar que los sistemas de computación pueden fallar.
- En un sistema distribuido los fallos son parciales, es decir algún componente puede fallar mientras otros siguen funcionando.
- Algunas técnicas de tratamiento de fallos son:

Tratamiento de fallos

- Debemos aceptar que los sistemas de computación pueden fallar.
- En un sistema distribuido los fallos son parciales, es decir algún componente puede fallar mientras otros siguen funcionando.
- Algunas técnicas de tratamiento de fallos son:
  - **Detección** algunos tipos de fallos pueden ser detectados y corregidos. (utilizando checksums se puede detectar que los datos en un mensaje están corruptos.)

Tratamiento de fallos

- Debemos aceptar que los sistemas de computación pueden fallar.
- En un sistema distribuido los fallos son parciales, es decir algún componente puede fallar mientras otros siguen funcionando.
- Algunas técnicas de tratamiento de fallos son:
  - **Detección** algunos tipos de fallos pueden ser detectados y corregidos. (utilizando checksums se puede detectar que los datos en un mensaje están corruptos.)
  - **Enmascaramiento** Algunos fallos que han sido detectados pueden ser escondidos o reducir su impacto. (se puede retransmitir un mensaje que está corrupto)

Tratamiento de fallos

- Debemos aceptar que los sistemas de computación pueden fallar.
- En un sistema distribuido los fallos son parciales, es decir algún componente puede fallar mientras otros siguen funcionando.
- Algunas técnicas de tratamiento de fallos son:
  - **Detección** algunos tipos de fallos pueden ser detectados y corregidos. (utilizando checksums se puede detectar que los datos en un mensaje están corruptos.)
  - **Enmascaramiento** Algunos fallos que han sido detectados pueden ser escondidos o reducir su impacto. (se puede retransmitir un mensaje que está corrupto)
  - **Tolerancia** Al aceptar que un sistema puede fallar, los usuarios deben aprender a vivir con ellos (si un servidor web no está disponible, el explorador reporta este estado al usuario para que intente más tarde)

Tratamiento de fallos

- Debemos aceptar que los sistemas de computación pueden fallar.
- En un sistema distribuido los fallos son parciales, es decir algún componente puede fallar mientras otros siguen funcionando.
- Algunas técnicas de tratamiento de fallos son:
  - **Detección** algunos tipos de fallos pueden ser detectados y corregidos. (utilizando checksums se puede detectar que los datos en un mensaje están corruptos.)
  - **Enmascaramiento** Algunos fallos que han sido detectados pueden ser escondidos o reducir su impacto. (se puede retransmitir un mensaje que está corrupto)
  - **Tolerancia** Al aceptar que un sistema puede fallar, los usuarios deben aprender a vivir con ellos (si un servidor web no está disponible, el explorador reporta este estado al usuario para que intente más tarde)
  - **Recuperación** se puede diseñar un software para que se recupere de los fallos (implementando un mecanismo 'roll back')

Tratamiento de fallos

- Debemos aceptar que los sistemas de computación pueden fallar.
- En un sistema distribuido los fallos son parciales, es decir algún componente puede fallar mientras otros siguen funcionando.
- Algunas técnicas de tratamiento de fallos son:
  - **Detección** algunos tipos de fallos pueden ser detectados y corregidos. (utilizando checksums se puede detectar que los datos en un mensaje están corruptos.)
  - **Enmascaramiento** Algunos fallos que han sido detectados pueden ser escondidos o reducir su impacto. (se puede retransmitir un mensaje que está corrupto)
  - **Tolerancia** Al aceptar que un sistema puede fallar, los usuarios deben aprender a vivir con ellos (si un servidor web no está disponible, el explorador reporta este estado al usuario para que intente más tarde)
  - **Recuperación** se puede diseñar un software para que se recupere de los fallos (implementando un mecanismo 'roll back')
  - **Redundancia** se puede utilizar componentes redundantes (múltiples vías para routers, un archivo en varios dispositivos de almacenamiento, etc)

Contenido I

Contenido II

Concurrencia

- Múltiples usuarios pueden acceder al mismo recurso al mismo tiempo, en algunos casos para realizar actualizaciones.
- Una de las medidas para manejar la concurrencia es implementar un acceso secuencial, sin embargo esto puede hacer lento al sistema.
- Un mecanismo bien aceptado para manejar la concurrencia es el uso de semáforos de los SO.
- Pueden existir problemas de consistencia que deben ser manejadas correctamente por el programador.

Contenido I

Contenido II

Transparencia

- La transparencia se refiere a la ocultación al usuario y al programador de la separación de componentes de un sistema distribuido.
- Los tipos de transparencia definidos son:
  - **Transparencia de acceso** permite el acceso a recursos locales y remotos utilizando operaciones idénticas.

Transparencia

- La transparencia se refiere a la ocultación al usuario y al programador de la separación de componentes de un sistema distribuido.
- Los tipos de transparencia definidos son:
  - **Transparencia de acceso** permite el acceso a recursos locales y remotos utilizando operaciones idénticas.
  - **Transparencia de ubicación** Permite acceder a los recursos sin conocer la ubicación física ni la ubicación en la red.

Transparencia

- La transparencia se refiere a la ocultación al usuario y al programador de la separación de componentes de un sistema distribuido.
- Los tipos de transparencia definidos son:
  - **Transparencia de acceso** permite el acceso a recursos locales y remotos utilizando operaciones idénticas.
  - **Transparencia de ubicación** Permite acceder a los recursos sin conocer la ubicación física ni la ubicación en la red.
  - **Transparencia de concurrencia** El usuario no debería notar que existen más usuarios usando el mismo servicio al mismo tiempo.

Transparencia

- La transparencia se refiere a la ocultación al usuario y al programador de la separación de componentes de un sistema distribuido.
- Los tipos de transparencia definidos son:
  - **Transparencia de acceso** permite el acceso a recursos locales y remotos utilizando operaciones idénticas.
  - **Transparencia de ubicación** Permite acceder a los recursos sin conocer la ubicación física ni la ubicación en la red.
  - **Transparencia de concurrencia** El usuario no debería notar que existen más usuarios usando el mismo servicio al mismo tiempo.
  - **Transparencia de replicación** Permite el uso de varias instancias de los recursos o servicios para mejorar las prestaciones sin que los usuarios lo noten.

Transparencia

- **Transparencia frente a fallos** Algunos errores pueden ser detectados y corregidos sin la intervención del usuario.

Transparencia

- **Transparencia frente a fallos** Algunos errores pueden ser detectados y corregidos sin la intervención del usuario.
- **Transparencia de movilidad** Los recursos pueden ser movidos de un lugar a otro sin afectar el rendimiento o su forma de uso.

Transparencia

- **Transparencia frente a fallos** Algunos errores pueden ser detectados y corregidos sin la intervención del usuario.
- **Transparencia de movilidad** Los recursos pueden ser movidos de un lugar a otro sin afectar el rendimiento o su forma de uso.
- **Transparencia de rendimiento** Permite realizar reconfiguraciones al sistema cuando se necesario para mejorar el rendimiento.

Transparencia

- **Transparencia frente a fallos** Algunos errores pueden ser detectados y corregidos sin la intervención del usuario.
- **Transparencia de movilidad** Los recursos pueden ser movidos de un lugar a otro sin afectar el rendimiento o su forma de uso.
- **Transparencia de rendimiento** Permite realizar reconfiguraciones al sistema cuando se necesario para mejorar el rendimiento.
- **Transparencia de escalado** Los usuarios no deben notar cuando el número de usuarios o recursos en el sistema se ha incrementado.

Transparencia

- **Transparencia frente a fallos** Algunos errores pueden ser detectados y corregidos sin la intervención del usuario.
- **Transparencia de movilidad** Los recursos pueden ser movidos de un lugar a otro sin afectar el rendimiento o su forma de uso.
- **Transparencia de rendimiento** Permite realizar reconfiguraciones al sistema cuando se necesario para mejorar el rendimiento.
- **Transparencia de escalado** Los usuarios no deben notar cuando el número de usuarios o recursos en el sistema se ha incrementado.

Las más importantes son la **transparencia de acceso** y **transparencia de ubicación** ya que su presencia o ausencia afecta principalmente a la utilización de los recursos distribuidos.

Contenido I

Contenido II

Consulta

13 Oct. 11:59pm Via Turnitin

- Consulta: Técnicas para la sincronización de relojes de varias computadoras en un sistema distribuido.
- Consulta: Ataque de denegación de servicio y que medidas existen para prevenirlos.
- Dé 5 ejemplos de recursos de hardware y 5 recursos de software que pueden ser compartidos. De ejemplo de casos en los que se necesesitaría compartirlos en un sistema distribuido.
- Un programa servidor escrito en C++ provee la implementación de un objeto BLOB que se supone puede ser accedido por diferentes clientes que pueden estar escritos en diferentes lenguajes como Java. Los computadores cliente y servidor pueden tener diferente hardware pero todos están conectados al Internet. Describa los problemas ocasionados por cada uno de los 5 aspectos de heterogeneidad mencionados que deben ser resueltos para que un cliente pueda invocar un método en el objeto BLOB del servidor.