Servicios en sistemas distribuidos

UNIVERSIDAD NEBRIJA

Índice

- Servicios de nombres
- Servicios web
- Servicios de archivos distribuidos
- Memoria distribuida

Servicios de nombres

- Nombre: es una cadena de bits o caracteres utilizados para hacer referencia a una entidad
  - Se utiliza para compartir recursos, para identificar entidades de manera única, para hacer referencia a ubicaciones, etc.
  - Cada entidad requiere un punto de acceso denominado dirección
    - La dirección es un tipo de nombre
    - Cada entidad puede tener varios puntos de acceso
      - Pero únicamente un nombre con independencia del número de puntos de acceso y direcciones
    - Los puntos de acceso pueden modificarse con el tiempo

Servicios de nombres

- Identificador: es un tipo de nombre que:
    - Hace referencia a una entidad como máximo
    - Cada entidad es referida por al menos un identificador
    - Siempre hace referencia a la misma entidad, nunca se reutiliza
- OJO: si se puede reasignar una dirección a una entidad diferente, no podemos utilizar una dirección como un identificador
- Para relacionar direcciones e identificadores la solución más simple es una tabla de pares (nombre, dirección)
    - En sistemas distribuidos que se expanden una tabla centralizada no funcionará

Servicios de nombres

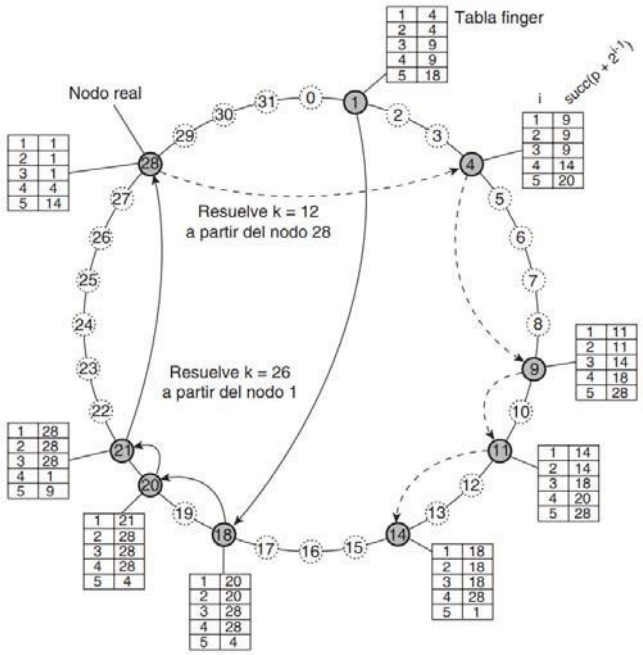
- Soluciones basadas en origen:
  - Ubicación de origen: mantiene el registro de la ubicación actual de una entidad
    - Se suele escoger como ubicación origen la ubicación en la que fue creada la entidad
    - El cliente debe hacer primero contacto con el origen, lo cual supone un problema de latencia y rendimiento para un sistema móvil

Servicios de nombres

- Soluciones basadas en tablas hash distribuidas:
  - El sistema de cuerdas representa a muchos sistemas DHT
  - El sistema de cuerdas utiliza un identificador de *m* bits para asignar los identificadores de manera aleatoria a los nodos, así como las llaves para especificar las entidades
  - En lugar de este método lineal basado en la búsqueda de la llave, cada nodo de cuerdas mantiene una tabla finger

Servicios de nombres

- Soluciones basadas en tablas hash distribuidas:

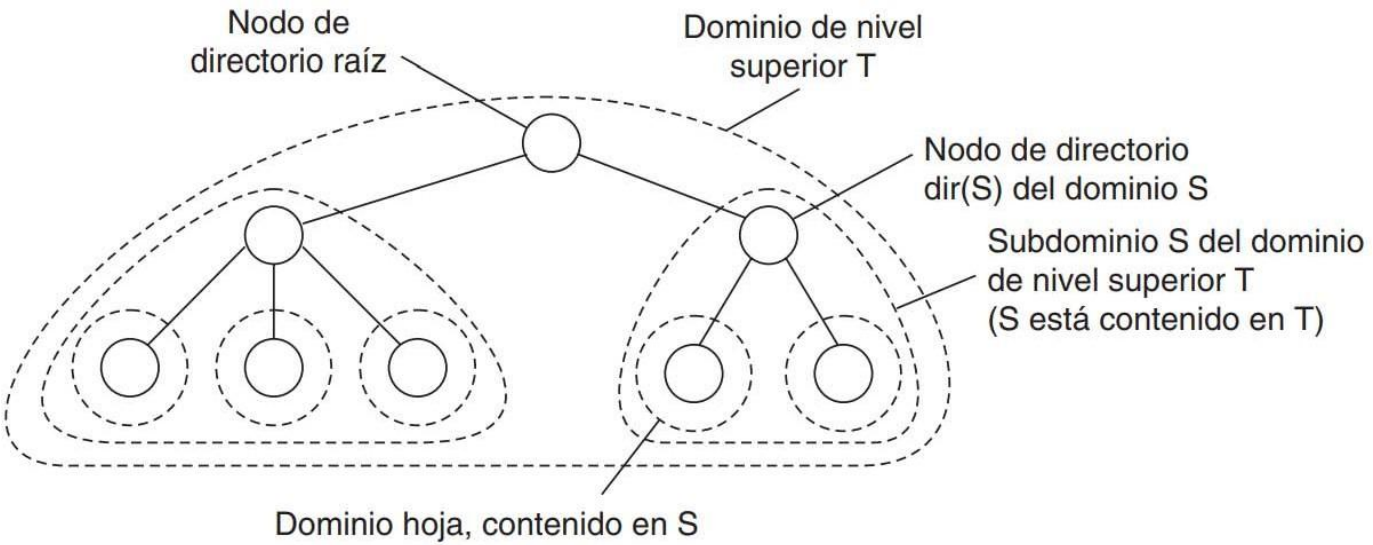


Servicios de nombres

- Soluciones basadas en tablas hash distribuidas:
  - Asignación de nodos basados en la topología: asignar identificadores de tal modo que dos nodos próximos tengan identificadores que se encuentren cercanos entre sí
    - PROBLEMA: mapear un anillo lógico en Internet no es trivial (sólo redes pequeñas)
  - Ruteo por proximidad: los nodos mantienen una lista de alternativas para reenviar una petición
    - Aumenta el almacenamiento requerido por las tablas finger pero robustece el sistema frente a errores

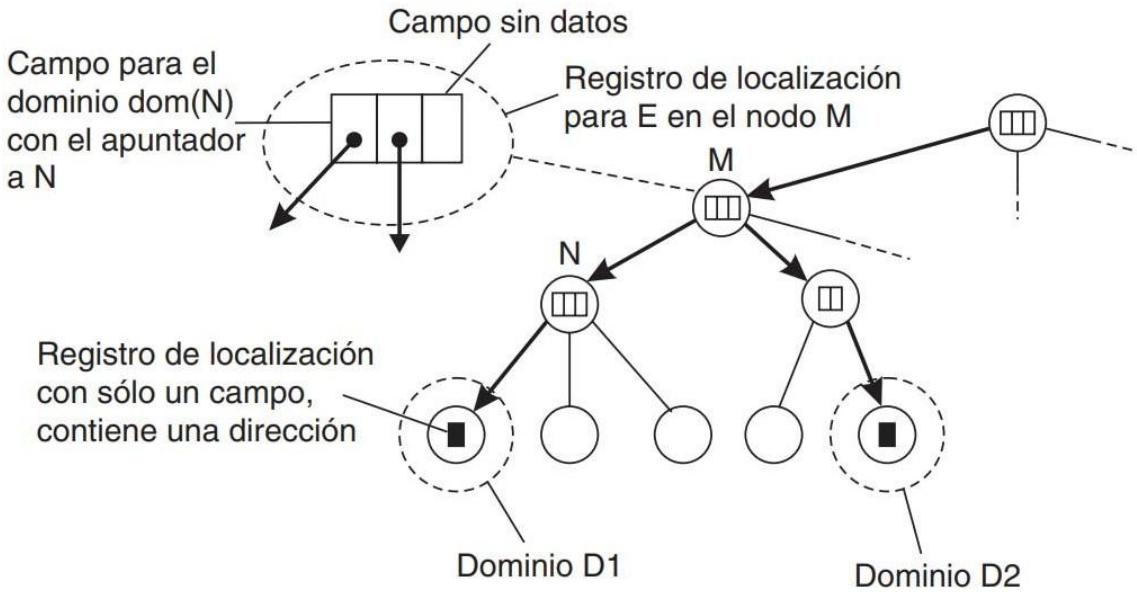
Servicios de nombres

- Dominios: arquitectura



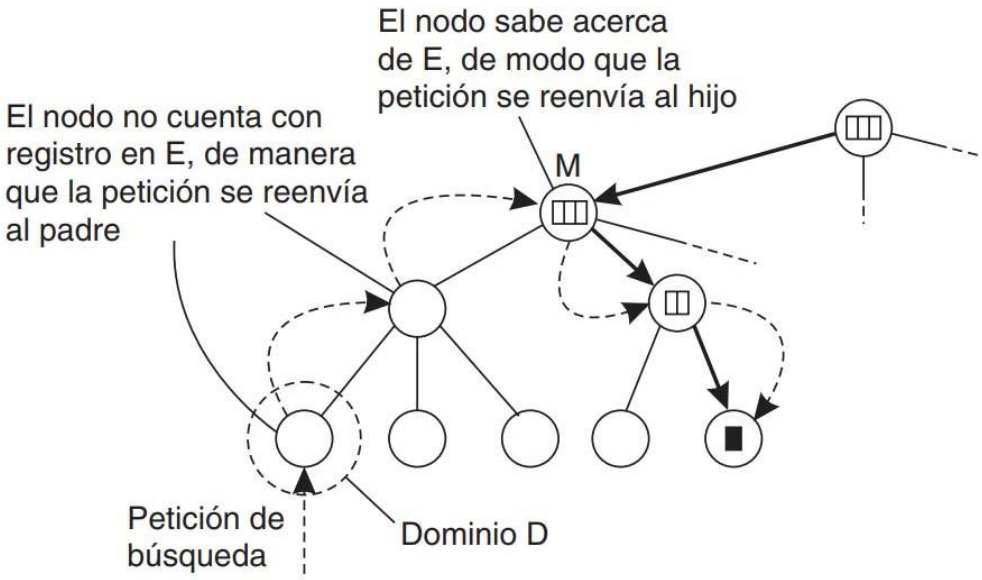
Servicios de nombres

- Dominios: estructuras de datos



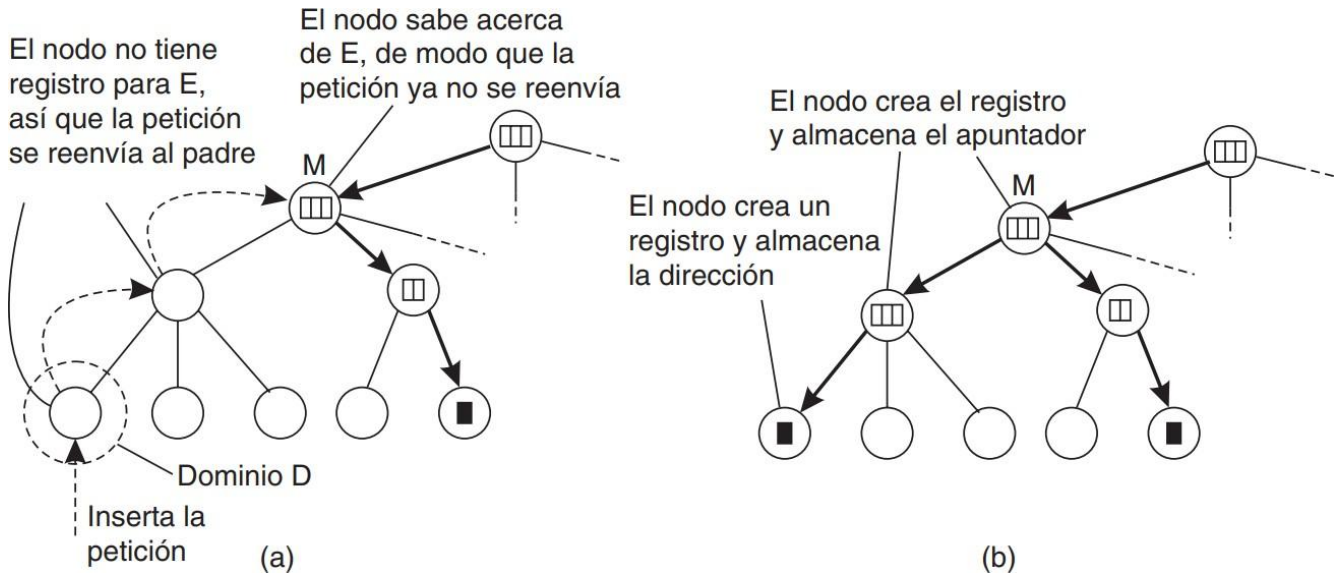
Servicios de nombres

- Dominios: búsquedas



Servicios de nombres

- Dominios: inserción de nodos



Servicios de nombres

- Resolución de nombre: proceso de búsqueda de un nombre
  - Comienza en el nodo *N* del grafo de nombres, donde el nombre *etiqueta_1* se busca en la tabla de directorio y el cual devuelve el identificador del nodo al que hace referencia, *etiqueta1*
  - La resolución se detiene en el último nodo referido como *etiqueta_n*, al devolver el contenido de dicho nodo
  - Una búsqueda de nombre devuelve el identificador de un nodo a partir del cual continúa el proceso de resolución de nombres

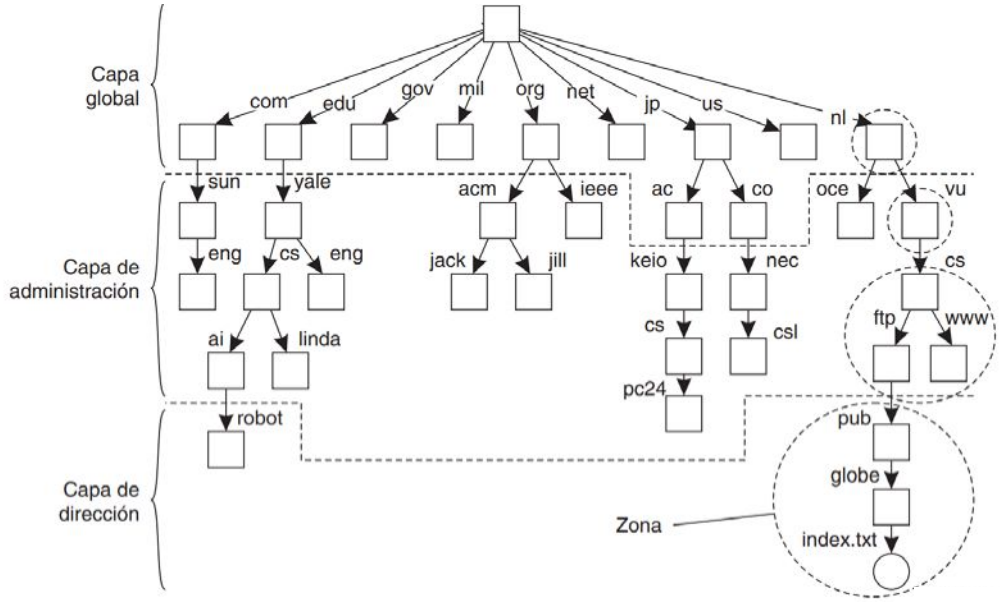
N:<etiqueta_1, *etiqueta_2*,…, *etiqueta_n>*

Servicios de nombres

- Espacio de nombre: es el núcleo de un servicio de nombres, que permite a los usuarios y procesos agregar, quitar, y buscar nombres
- Está constituido por tres capas lógicas:
  - Capa global: formada por los nodos de más alto nivel: nodo raíz y sus hijos
    - Se caracterizan por su estabilidad, las tablas de directorio rara vez se modifican
  - Capa de administración: formada por los nodos directorio que son administrados dentro de una única organización
    - Los cambios ocurren con más frecuencia que en los nodos de la capa global
  - Capa de dirección: consta de nodos que pudieran modificarse de manera regular
    - Redes de área local gestionadas por usuarios finales o administradores de sistemas

Servicios de nombres

- Espacio de nombre: Recuerda DNS



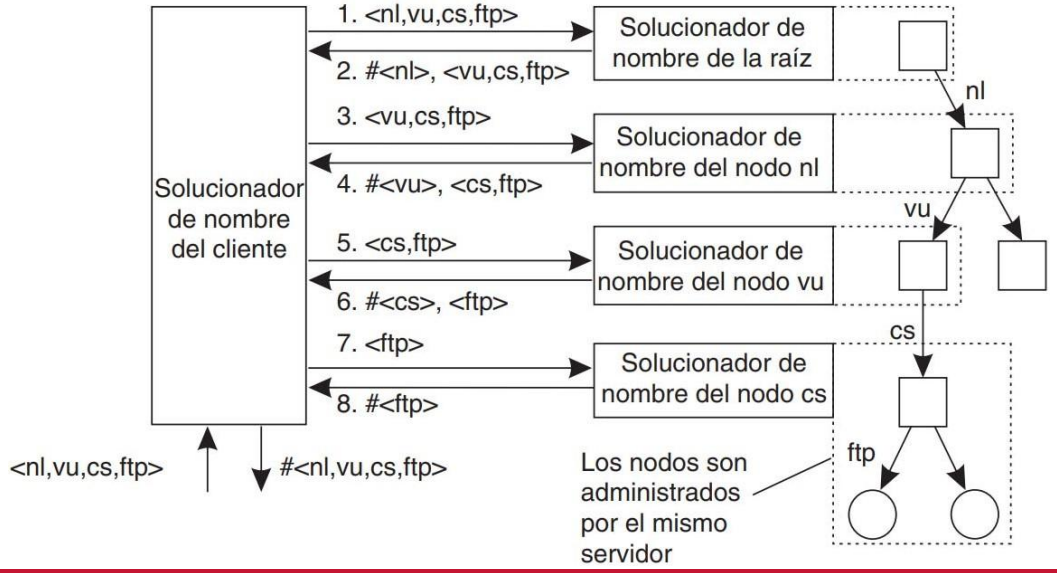
Servicios de nombres

- Espacio de nombre: Recuerda DNS

Elemento	Global	Administración	Dirección
Escala geográfica de una red	A nivel mundial	Organización	Departamento
Número total de nodos	Pocos	Muchos	Cuantiosos números
Respuesta a las búsquedas	Segundos	Milisegundos	Inmediata
Propagación de actualizaciones	Lenta	Inmediata	Inmediata
Número de réplicas	Muchas	Ninguna o pocas	Ninguna
¿Se aplica el cacheo del lado del cliente?	Sí	Sí	Ocasionalmente

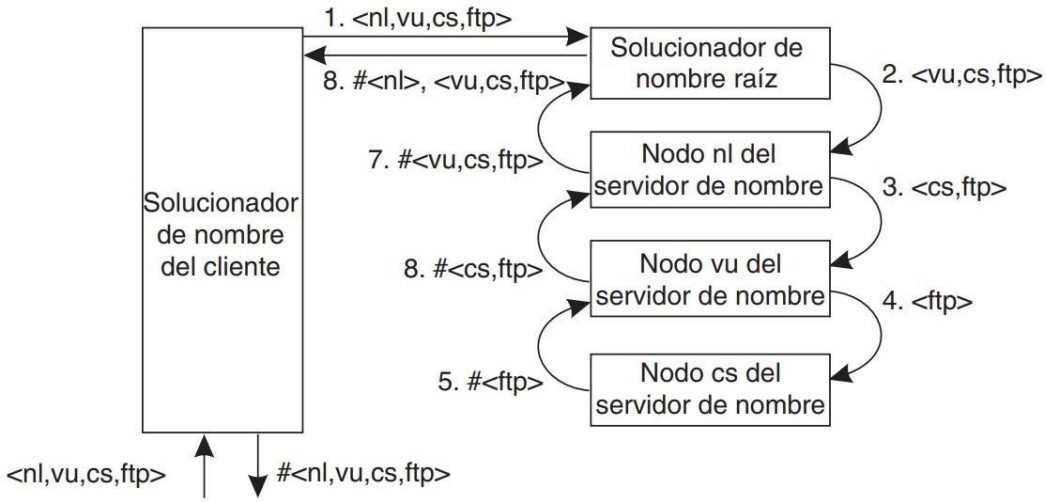
Servicios de nombres

- Espacio de nombre: Recuerda DNS
  - Resolución iterativa de nombres:



Servicios de nombres

- Espacio de nombre: Recuerda DNS
  - Resolución iterativa de nombres:



Ejercicio práctico

Buscar por comandos cual es la IP de la web www.mec.es

Encuentra la denominación del servidor DNS que contiene www.mec.es

Servicios web

- Uno de los problemas clásicos en los sistemas distribuidos es el del intercambio de información entre equipos
- Incluso con el esquema más sencillo (cliente-servidor) es necesario definir un protocolo, escoger un formato de datos y analizar su extensibilidad
- La opción clásica en el uso de sockets con un intercambio de mensajes serializados, no obstante tiene diversos inconvenientes: se debe diseñar el protocolo desde cero, no está estandarizado para por lo que es difícilmente escalable (no todos los equipos tienen estandarizados los protocolos)
- Como alternativa surgen los servicios web

Servicios web

- Los servicios web usan una serie de estándares para implementar un sistema de comunicación que pueda ser usado sin problema por multitud de aplicaciones hechas con tecnologías muy diferentes
  - Protocolo de intercambio de mensajes: Es el que regula cómo se construyen los mensajes de petición y de respuesta. Suele basarse en el estándar XML
  - Protocolo de transporte: Se usa principalmente HTTP, aunque es posible utilizar otros protocolos estándar tales como SMTP o FTP

- Estándares abiertos: existe la posibilidad de estandarizar más operaciones y publicarlas como nuevos servicios web

- Un servicio web consiste en un componente software que expone un conjunto de métodos que pueden ser utilizados desde otros programas

Servicios web

- Estructura de un servicio web: presenta dos componentes básicos
  - **Componente software**: código reutilizable que para implementar la **funcionalidad de los servicios**. La funcionalidad es **compartida por todos los módulos de la aplicación**
  - **Servidor SOAP**: es la **interfaz entre el cliente** y el c**omponente que implementa el servicio**. La comunicación entre cliente y servicio web se realiza **vía XML** utilizando el **protocolo de codificación SOAP** (Simple Object Access Protocol).
    - Se encarga de gestionar tanto las preguntas como las respuestas del protocolo

Servicios web

- **Roles de la arquitectura**
    - **Cliente del servicio**: usuario del mismo mediante mensajes XML a través de HTTP
    - **Proveedor de servicio**: la **plataforma que hospeda el servicio** y **da acceso al mismo**. El proveedor define una **descripción del servicio** con sus características y lo **publica para que sea accesible por los clientes**
    - **Registro de servicios**: contiene la **descripción de los servicios publicados** por los proveedores. Los clientes lo utilizan para **localizar un servicio web** y **obtener información** del mismo: quién es el proveedor, qué protocolos utiliza, etc.

Servicios web

- **Roles de la arquitectura**

  Para utilizar este registro se recurre a los siguientes protocolos:

  - **Protocolo de descubrimiento de servicios:** La idea es **registrar los servicios web en un sitio centralizado** (registro) de manera que **expongan su descripción y localización**. El registro está basado en **XML**

  - **Protocolo de descripción de servicios:** Es una tecnología que permite **describir la funcionalidad de un servicio web**. Esta basada en **XML** y nos dice que métodos se pueden **invocar en un servicio**, que **parámetros acepta**, que **valores devuelven**, etc...

Servicios web

- Basándose en XML **se definen una serie de estándares** para **la interacción entre los diferentes roles** de la arquitectura: **SOAP**, WSDL, UDDI
    - **SOAP** (Simple Object Access Protocol): Define el formato de los documentos XML que se intercambiarán entre las aplicaciones
    - **Partes del mensaje SOAP**
        - Envelope: elemento principal del documento, debe estar en todos los mensajes
        - Header: elemento hijo de Envelope que proporciona información sobre los nodos que atraviesa el mensaje
        - Body: parte principal del mensaje

Servicios web

- Basándose en XML **se definen una serie de estándares** para **la interacción entre los diferentes roles** de la arquitectura: **SOAP**, WSDL, UDDI
  - **SOAP en Invocación Remota a Métodos (RPC)**
  - Body del **mensaje de llamada** debe contener un elemento cuyo nombre coincida con el nombre del método de llamada

Método:
public float **GetPrecio**(String código)

Mensaje de llamada XML:

```
<SOAP-ENV: Body>
<GetPrecio xmlns="">
        <codigo>…</codigo>
</GetPrecio>
</SOAP-ENV: Body>
```

Servicios web

- Basándose en XML **se definen una serie de estándares** para **la interacción entre los diferentes roles** de la arquitectura: **SOAP**, WSDL, UDDI
  - **SOAP en Invocación Remota a Métodos (RPC)**
  - Body del **mensaje de respuesta** por convenio debe acabar con la palabra Response

Método:
public float **GetPrecio**(String código)

Mensaje de respuesta XML:

<SOAP-ENV: Body>
<**GetPrecioResponse** xmlns="">
<Precio>…</Precio>
</GetPrecio>
</SOAP-ENV: Body>

Servicios web

- Basándose en XML **se definen una serie de estándares** para **la interacción entre los diferentes roles** de la arquitectura: SOAP, **WSDL**, UDDI

  - **WSDL** (Web Service Definition Language): genera un documento XML que describe las características del servicio ofrecido. El documento debe contener:

    - Los **métodos proporcionados** por el servicio
    - Los **parámetros** de **entrada y salida**
    - Cómo **conectarse** al servicio

  - No es necesario generarlos manualmente, pueden generarse mediante diversas herramientas

Servicios web

- Basándose en XML **se definen una serie de estándares** para **la interacción entre los diferentes roles** de la arquitectura: SOAP, **WSDL**, UDDI

  - **Estructura del documento WSDL:**
    - Elemento raíz **definitions:** se declaran los namespaces utilizados por el documento: referencias al WSDL, al XML Schema, a SOAP y al propio documento
    - Elemento **types**: se definen los tipos de datos mediante la definición XML Schema
    - Elemento **message**: representan la información que va a ser intercambiada

Servicios web

- Basándose en XML **se definen una serie de estándares** para **la interacción entre los diferentes roles** de la arquitectura: SOAP, **WSDL**, UDDI

    - **Estructura del documento WSDL:**
        - Elemento **portType**: operaciones expuestas por el servicio web
        - Elemento **binding**: especifica el tipo de transporte utilizado, se vincula a un protocolo como SOAP, HTTP, etc.
        - Elemento **service**: especifica la dirección o punto de entrada del servicio web. Mediante port se especifica la dirección de cada elemento

Servicios web

- Basándose en XML **se definen una serie de estándares** para **la interacción entre los diferentes roles** de la arquitectura: SOAP, WSDL, **UDDI**
    - **UDDI:** define como publicar y localizar un servicio web. Por tanto, pueden acceder proveedores de servicios y clientes.
    - Recurre a **mensajes SOAP** para **publicar**, **editar** y **buscar** información

Servicios web

- **Arquitectura SOA (Service Oriented Applications):** sistemas con una arquitectura abierta y extensible, constituida por **servicios autónomos interoperables** entre ellos e implementados como servicios web

- La **funcionalidad global del sistema**, se consigue mediante una serie de **aplicaciones que consumen servicios**

- **Ventajas**
  - **Extensibilidad** : El sistema puede crecer más fácilmente en **funcionalidad**
    Abstracción y **encapsulamiento** de los datos
  - **Heterogeneidad**: Todos los servicios presentan un **modo de acceso**
  - **similar** y las **migraciones** pueden realizar se manera simultánea

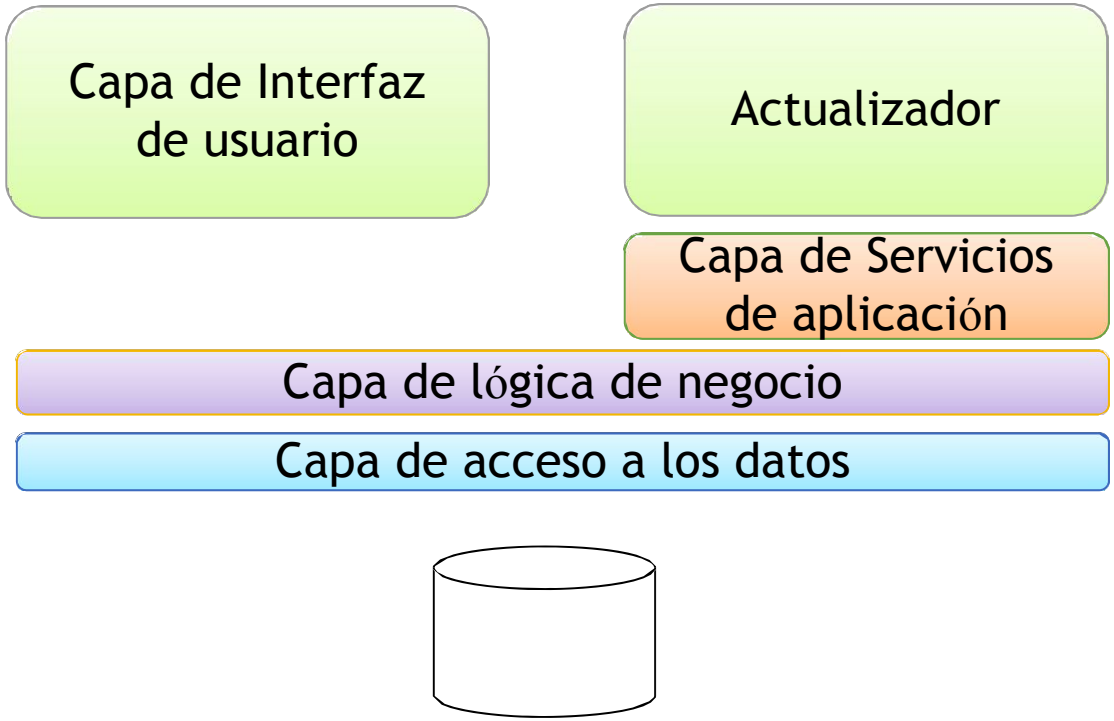
Servicios web

- **Arquitectura SOA (Service Oriented Applications):** sistemas con una arquitectura abierta y extensible, constituida por **servicios autónomos interoperables** entre ellos e implementados como servicios web

- **Ventajas**

  - **Facilita el desarrollo**: Simplifica la interacción entre los diferentes nodos a comunicar ya que no requiere conocer los protocolos ni implementar métodos para gestionar los ficheros XML
  - **Favorece la interacción de los clientes con nuestro sistema** a través de distintos tipos de aplicaciones (web, escritorio, móvil, etc.) Favorece la **seguridad exterior**, ya que las interfaces de uso del sistema están **reguladas** y **fijadas**
    Los servicios web de estas arquitecturas suelen implementar la capa de negocio de las arquitecturas multicapa

Servicios web



Servicios web

- **Arquitectura SOA (Service Oriented Applications):** sistemas con una arquitectura abierta y extensible, constituida por **servicios autónomos interoperables** entre ellos e implementados como servicios web

- **Principios**
  - **Las fronteras se definen de manera explícita**: el cliente y el servicio están desacoplados y sólo deben conocer los datos que intercambian recurriendo no a tipos específicos sino a esquemas de datos en XML
  - **Los servicios son autónomos**: no necesitan nada del cliente y gestionan su propia seguridad de manera independiente al cliente
  - **La política del servicio debe estar bien definida**: esta política indicará qué se puede hacer y cómo interaccionan los clientes

Servicios web

- **Arquitectura SOA (Service Oriented Applications):** sistemas con una arquitectura abierta y extensible, constituida por **servicios autónomos interoperables** entre ellos e implementados como servicios web

- **Principios**

    - **Los servicios son seguros a nivel de hilo:** no presentarán problemas de concurrencia
      Los servicios son fiables y robustos frente a fallos: la respuesta de los servicios siempre será determinista y se dispondrá de sistemas frente a recuperación del fallos

Servicios de archivos distribuidos

- Los **sistemas de archivos** son responsables de:
  - **Organizar** archivos
  - Almacenar archivos
  - **Recuperar** archivos
  - Nombrar archivos
  - **Compartir** archivos
  - Proteger archivos

- Los sistemas de archivos proporcionan una **interfaz de programación** que **abstrae** los procedimientos anteriores

```
File archivo = null;
FileReader fr = null;
BufferedReader br = null;
```

Creación

```
archivo = new File (this.filename);
fr = new FileReader (archivo);
br = new BufferedReader(fr);
```

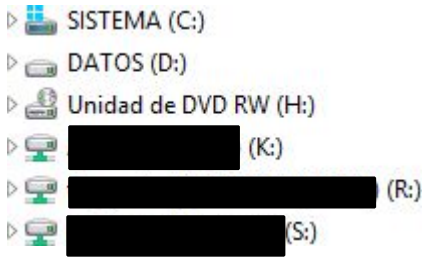
Apertura

```
String linea;
while((linea=br.readLine())!=null) {
   String[] tokens = linea.split(":");
}
```

Lectura

Servicios de archivos distribuidos

- Los **sistemas de archivos distribuidos** requieren una serie de **capas software** que se encargan de **servicios concretos** y que prestan **funcionalidad a capas superiores** para asegurar la **transparencia** del servicio
  - **Transparencia**: debe permitir la **escalabilidad** y la **flexibilidad** mediante
    - Transparencia de **acceso**: los clientes no deben conocer la distribución de los datos Su acceso debe **igual para un fichero local o remoto**



Servicios de archivos distribuidos

- Los **sistemas de archivos distribuidos** requieren una serie de **capas software** que se encargan de **servicios concretos** y que prestan **funcionalidad a capas superiores** para asegurar la **transparencia** del servicio
  - **Transparencia**: debe permitir la **escalabilidad** y la **flexibilidad** mediante
    - Transparencia de **ubicación**: los archivos pueden ser reubicados **sin cambiar sus nombres de ruta en los programas cliente**
    - Transparencia de **movilidad**: es posible mover un archivo como si de un **fichero local se tratara**

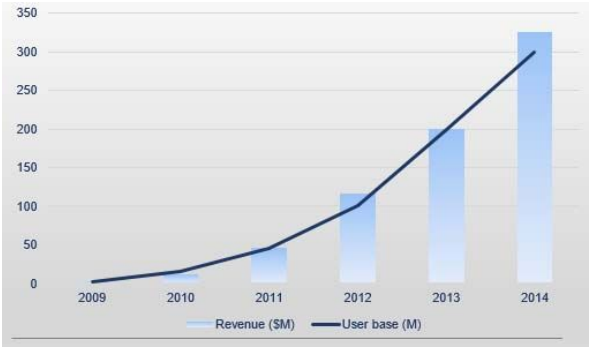
```
scp /path/to/file username@a:/path/to/destination
```

```
scp username@b:/path/to/file /path/to/destination
```

Servicios de archivos distribuidos

- Los **sistemas de archivos distribuidos** requieren una serie de **capas software** que se encargan de **servicios concretos** y que prestan **funcionalidad a capas superiores** para asegurar la **transparencia** del servicio
- **Transparencia**: debe permitir la **escalabilidad** y la **flexibilidad** mediante
  - Transparencia de **prestaciones**: el cliente debe seguir recibiendo un **servicio satisfactorio** dentro de una serie de condiciones preestablecidas
  - Transparencia de **escala**: el servicio debe funcionar correctamente **sin importar el número de archivos o clientes**

Dropbox is experiencing issues.



Servicios de archivos distribuidos

- **Actualización concurrente** de archivos: los cambios en un archivo por un cliente **no deben interferir** con la operación de otros clientes que acceden simultáneamente

- **Replicación de archivos**: realiza varias copias con el fin de permitir el **acceso a más clientes** y evitar posibles errores mediante la **redundancia de datos**

- **Heterogeneidad** del **hardware** y del **sistema operativo**

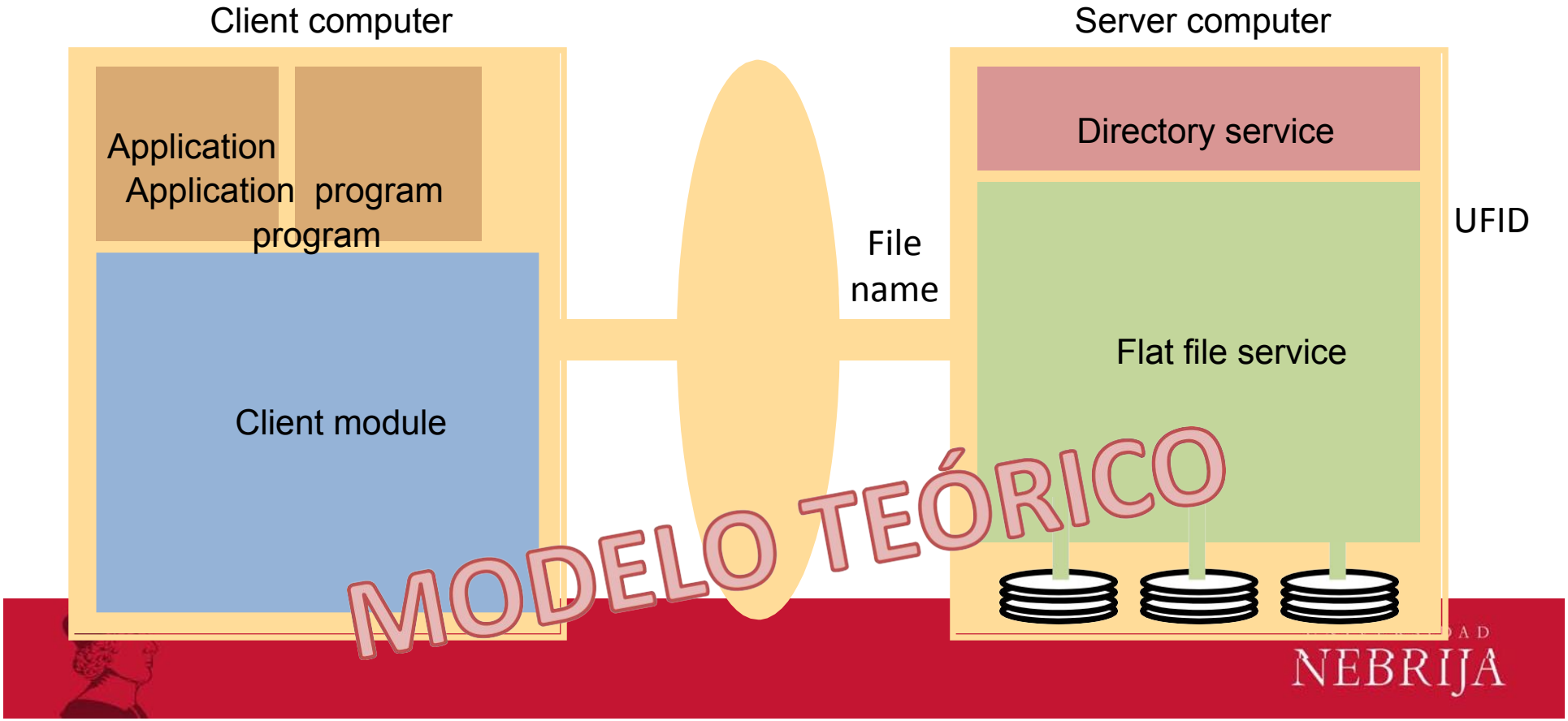
- **Consistencia**: asegurar que **todos** los clientes están **viendo la misma versión** del fichero aunque existan múltiples copias

- **Seguridad**: los archivos proporcionan mecanismos de control basados en **listas de acceso**

- **Tolerancia a fallos**: debe ser **más robusto que los sistemas locales**

Servicios de archivos distribuidos

•Arquitectura con tres módulos principales:



Servicios de archivos distribuidos

- Arquitectura con **tres módulos** principales:
  - **Servicio de archivos plano**: **operaciones en el contenido** de los archivos. Utilizan **identificadores únicos de archivos** (UFID)
  - **Servicio de directorio**: transformación de **nombres de texto** y **UFID**. **Gestiona directorios**
  - **Módulo cliente**: cubre con **una interfaz los dos anteriores**
    - Debe ser capaz de generar un sistema de archivos jerárquico basado en **árbol de directorios**
- **Control de acceso**: cada vez que se **solicita** una **transformación de nombre en UFID**, se realiza una **comprobación de acceso** cuya respuesta se **reenvía al cliente**. Se envía la identidad del usuario con cada solicitud de acceso
- **Agrupación de archivos**: grupo de archivos o colección que puede ser **recolocada en diferentes nodos** pero que **no puede modificar su identificador de grupo** (volumen) Esto permite **trasladar grandes volúmenes** de datos de **forma rápida**

Servicios de archivos distribuidos

NFS

- Se basa en la arquitectura de servicio de archivos. Es **independiente del sistema operativo y del hardware**
  - El servidor **NFS está instalado en el núcleo de cada nodo**
- Las solicitudes se traducen **en el módulo cliente a operaciones NFS** y **se transmiten al servidor**
  - El **cliente y el servidor** se comunican mediante **llamadas a procedimientos remotos** (RPC)
  - Posee una interfaz abierta, **cualquier proceso puede enviar una solicitud**

  De manera adicional se **comprobará la identidad del solicitante**
- Para que sea transparente el acceso a archivos locales y distribuidos se implementa un **módulo de sistema de archivos virtuales** que sirve para, mediante un nombre, **identificar** un **archivo de origen remoto o local**

  NO SOPORTA LA MIGRACIÓN MANUAL: Requiere de nueva configuración del sistema

Servicios de archivos distribuidos

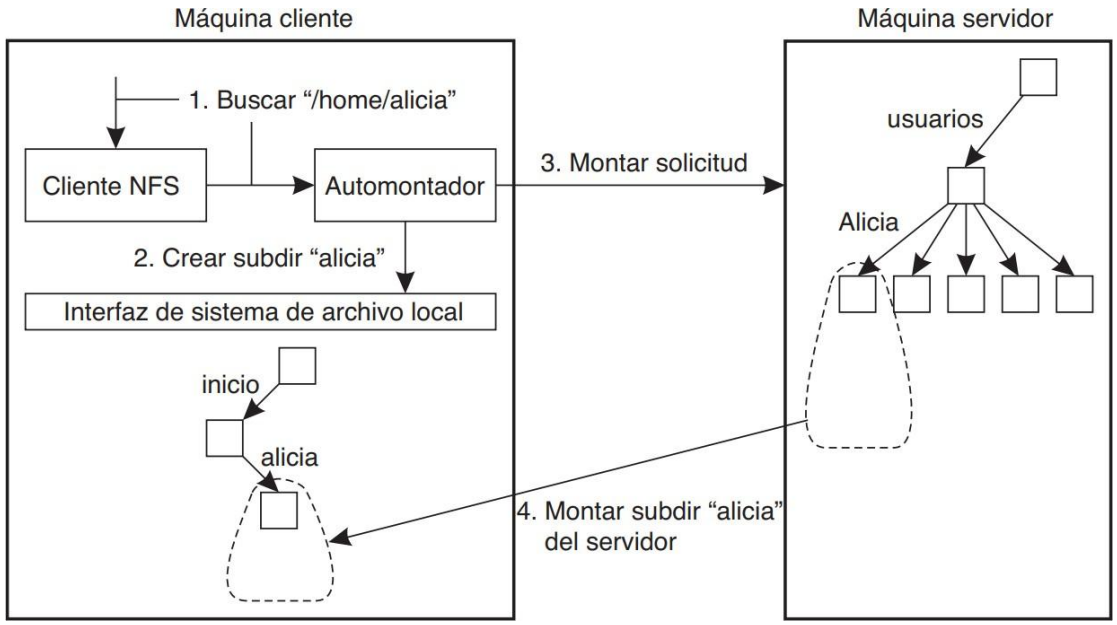
NFS

- **Control de acceso**: a diferencia de otros sistemas **no deja los archivos abiertos**, por tanto, **cada vez** que se desea acceder **se comprueban los permisos de usuario**
  - NFS dispone de un **servicio de montado manual** en el que:
    - Los clientes consultan al servidor y este les **devuelve la ruta de directorios** de los **ficheros remotos**
    - El **servidor** almacena todos las **rutas de archivos en un directorio conocido**
    - El **servicio** se interrumpe en los clientes **si cae el servidor central**
    - **Traducción del nombre de ruta**: se **gestiona en el cliente**
- **Automontador**: monta un volumen dinámicamente utilizando una **tabla de consulta**. Se utiliza para ficheros que se utilizan con mayor frecuencia.
    - **Envía la misma petición a varios servidores**
      - **Tolerante a fallos**: porque si el servidor contesta es que funciona
      - **Balancea la carga**: porque el servidor está disponible por tanto no está saturado

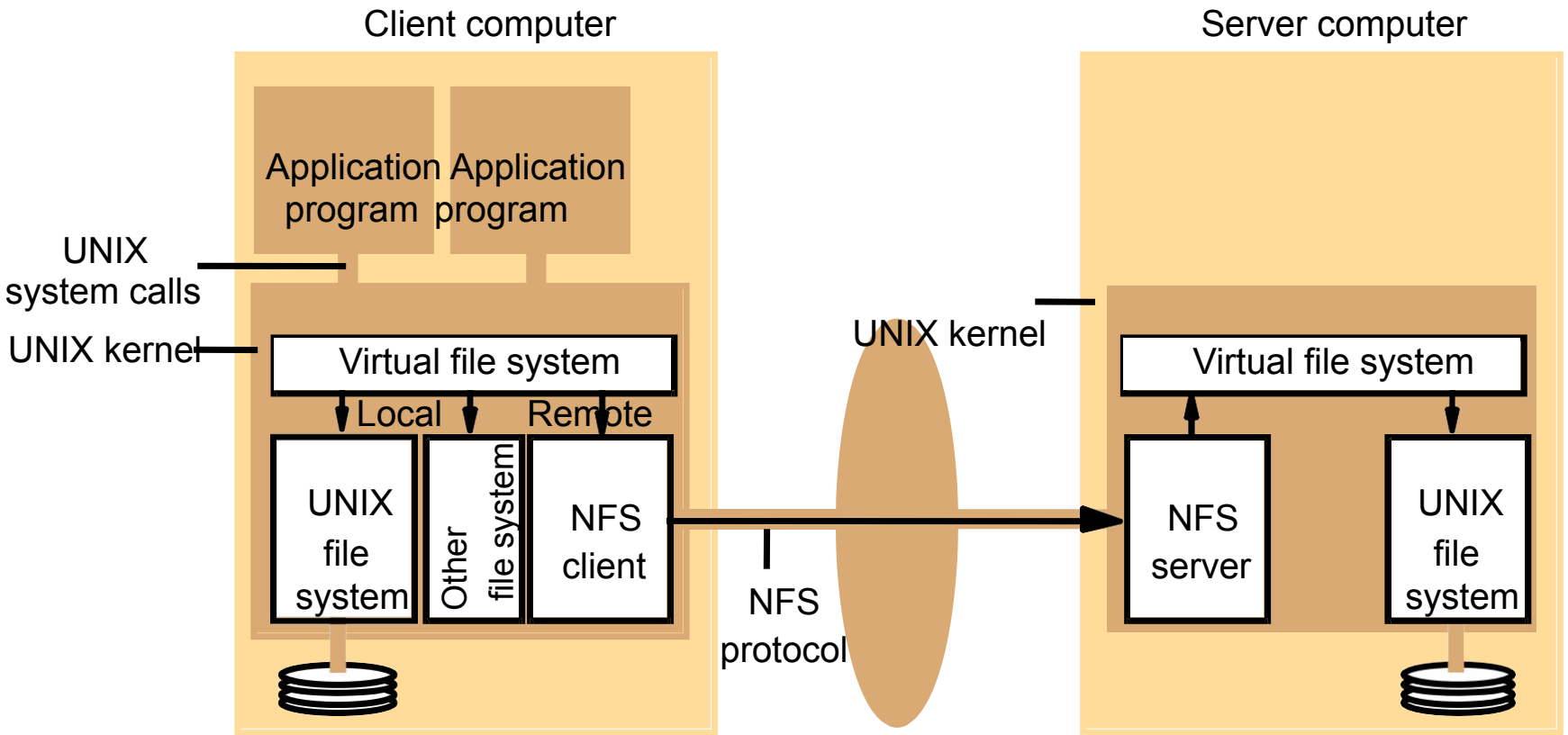
Servicios de archivos distribuidos

NFS

•**Automontador**: monta un volumen dinámicamente utilizando una **tabla de consulta**.



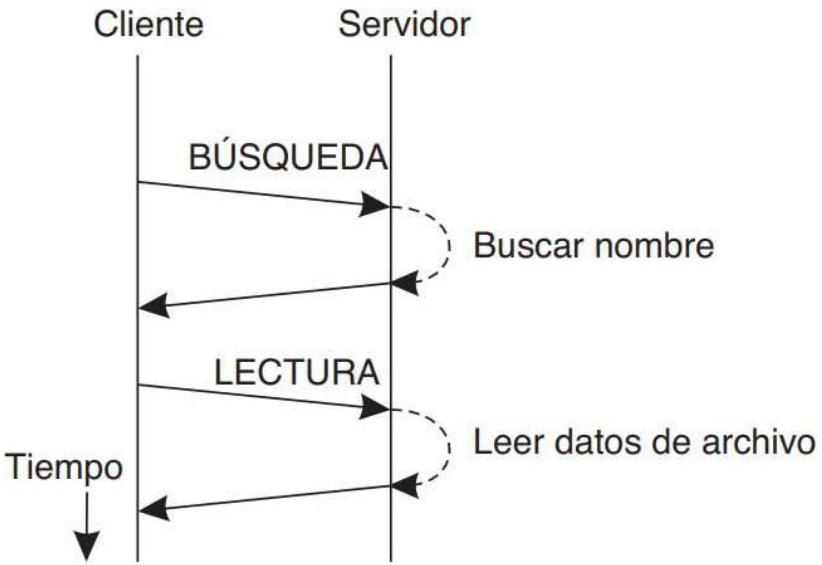
Servicios de archivos distribuidos

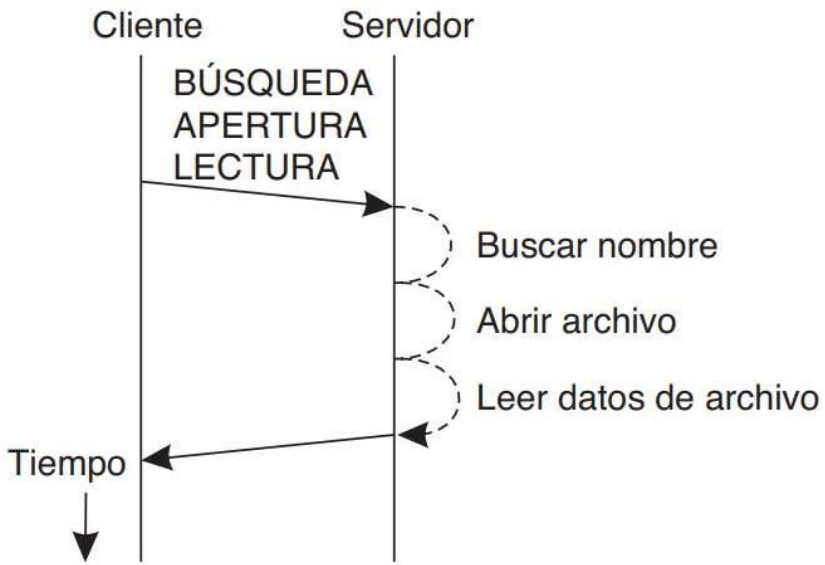


Servicios de archivos distribuidos

NFS

- Evolución del intercambio de mensajes basados en RPC con las versiones

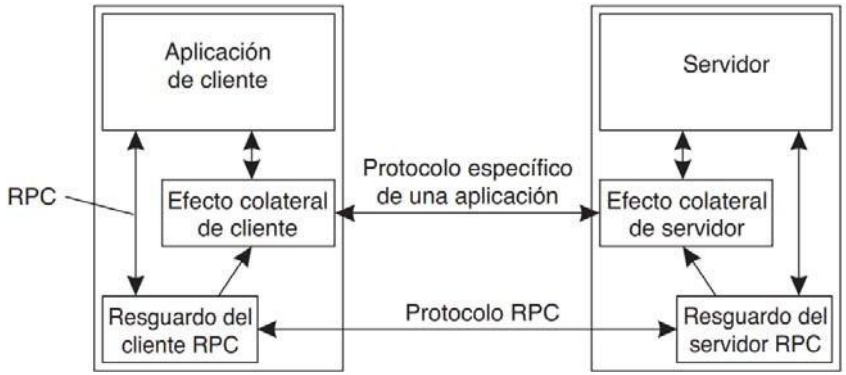




Servicios de archivos distribuidos

NFS

- RPC2 es un paquete que ofrece RPC fiables encima del protocolo UDP (no fiable)
  - El servidor devuelve regularmente mensajes al cliente para informarle que sigue trabajando en la solicitud
  - Efecto colateral: es un mecanismo mediante el cual el cliente y el servidor pueden comunicarse usando un protocolo específico de una aplicación
    - Ejemplo: Ajustar tiempo si la aplicación consiste en transmisión de vídeo



Servicios de archivos distribuidos

NFS

- Replicación del servidor:
  - Se aplica cuando la disponibilidad está en riesgo (pérdida de rendimiento)
  - No obstante, es preferible desplegar cachés en donde un archivo completo, o partes de él, se ponen localmente disponibles para un cliente (mejor rendimiento)
  - Es recomendable utilizar únicamente replicación para asegurar tolerancia a fallos
    - El sistema requiere un coordinador, que también puede fallar

Servicios de archivos distribuidos

NFS

- Replicación del servidor:
  - Proceso para evitar errores en el coordinador:
    1. Un cliente envía una solicitud a todo el grupo de servidores
    2. El coordinador transmite un número en secuencia en la fase de prepreparar
    3. Los nodos esclavos tienen que garantizar que el número en secuencia del coordinador sea aceptado por quórum (para asegurar que el coordinador funciona correctamente)
    4. Si se llega a un acuerdo, todos los procesos se pasan la información entre sí y ejecutan la operación, tras de lo cual el cliente finalmente puede ver el resultado

Servicios de archivos distribuidos

NFS

- Seguridad:
  - RPCSEC_GSS es una estructura general de seguridad que puede soportar un conjunto de mecanismos de seguridad para el establecimiento de canales seguros
  - Soporta integridad y confidencialidad de los mensajes además de las siguientes tecnologías:
    - El cliente transfiere su ID de usuario y de grupo al servidor, junto con una lista de grupos de los que afirma ser miembro (cliente fiable)
    - NFS seguro: un criptosistema de clave pública (sistema de autenticación más complejo
      - Kerberos
    - LIPKEY: sistema de clave pública que permite autenticar los clientes con una contraseña, en los servidores autenticados mediante una clave pública

Servicios de archivos distribuidos

NFS

- Seguridad:
  - Ataque de Sybil: un solo nodo es capaz de autoasignarse muchos identificadores (Bitcoin)
  - Ataque eclipse: un nodo malicioso controla a tantos nodos vecinos no defectuosos que llega a ser virtualmente imposible corregirlos (Ethereum)

  - Soluciones:
    - Restringir el número de conexiones entrantes para cada nodo
    - Evitar optimiza el rutado sólo por proximidad de red, ya que un atacante puede convencer con facilidad a un nodo para que apunte a nodos maliciosos
    - Remitir siempre los mensajes a lo largo de varias rutas desde nodos origen diferentes

Ejercicio práctico NFS

Servidor NFS

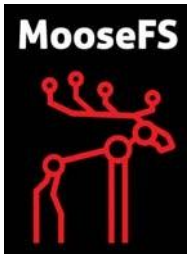
Cliente NFS

Servicios de archivos distribuidos

AFS

- El sistema de archivos en los servidores **se basa en NFS**
- AFS está diseñado para poder trabajar con un **gran número de clientes** utilizando la **caché**
  - Se recurre a **copias locales** de archivos y directorios completos
  - Cuando se **cierra un archivo remoto**, se **actualiza la versión del servidor** y se mantiene la **copia local** del cliente
- Posee una base de datos de todas las ubicaciones replicada para cada copia del volumen. Puede producir **inexactitudes** pero **aumenta el rendimiento**
- Trabaja utilizando **hilos** para que las solicitudes sean atendidas concurrentemente
- Cuando únicamente se realiza **lectura simplifica el procedimiento**
- Utiliza paquetes de **gran tamaño** para reducir el efecto de la **latencia**

Servicios de archivos distribuidos



AFS

- Algunas mejoras:
  - Redundant Arrays of Inexpensive Disk (RAID): datos segmentados en bloques de tamaño fijo a los que **se les aplica un FEC** para facilitar la **reconstrucción** en caso de fallo, aumentando las prestaciones frente a un sistema de disco dura local
  - Log-structured file storage (LFS): **acumular un conjunto de operaciones** de escritura y realizarlas simultáneamente para poder trabajar en las mismas regiones de disco o en el mismo nodo y de esta forma **optimizar espacio** y **velocidad de acceso**
- Otros sistemas de archivos distribuidos:
  - **Samba** : compartir archivos entre diferentes sistemas operativos
  - **BeeGFS**: sistema paralelo de clúster de archivos
  - **GlusterFS**: sistema de archivos distribuidos multiescalable
  - **MooseFS**: sistema de archivos distribuidos con tolerancia a errores

Servicios de archivos distribuidos

GFS

- Escenario:
  - Los archivos Google son de varios gigabytes y contienen muchos objetos más pequeños en su interior
  - Las actualizaciones de los archivos se realizan anexando datos en lugar de sobrescribir partes de un archivo
  - Los fallos en el servidor son una situación común

Servicios de archivos distribuidos

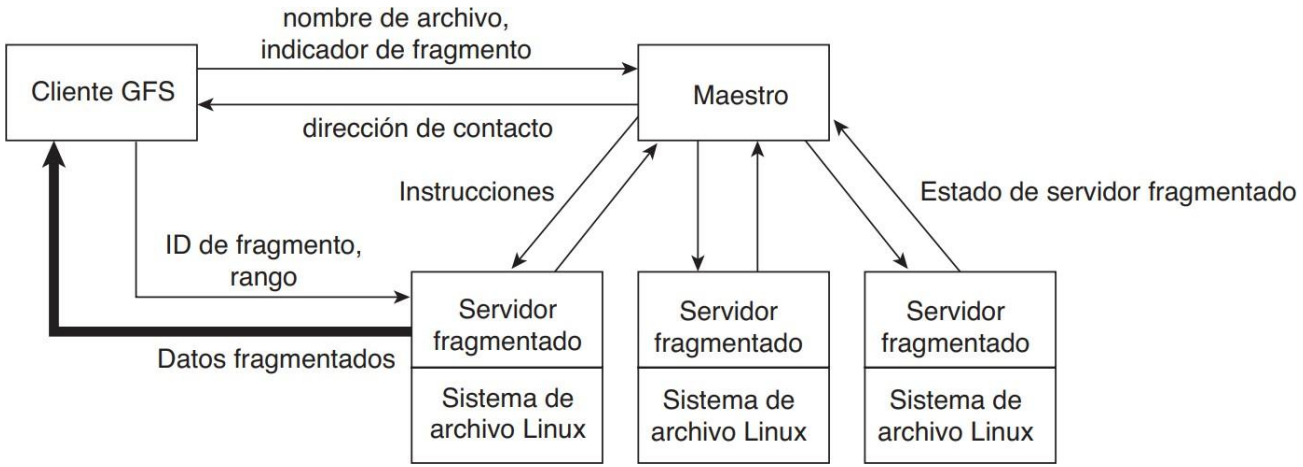
GFS

- Propuesta:
  - Cada grupo de GFS se compone de:
    - Un servidor maestro: controla cientos de servidores fragmentados
    - Múltiples servidores fragmentados: que realizan la mayor parte del procesado
  - Cada archivo GFS se divide en fragmentos de 64 Mbytes
  - Los fragmentos se distribuyen a través de los servidores llamados fragmentados
  - GFS maestro se contacta sólo para recabar información de metadatos
  - GFS cliente transfiere un nombre de archivo y un índice de fragmento al maestro, y espera una dirección de contacto para el fragmento
    - La dirección de contacto contiene toda la información necesaria para acceder al servidor fragmentado correcto y obtener el fragmento de archivo requerido

Servicios de archivos distribuidos

GFS

- Arquitectura:

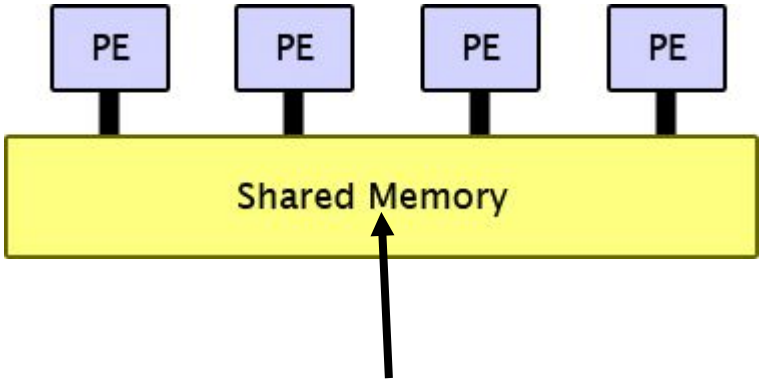


Memoria distribuida

- **Paso de mensajes** vs **Memoria compartida**:
  - **Paso de mensajes**: las variables se empaquetan en un proceso y el receptor las desempaqueta **protegiendo la información entre procesos. Requiere serializar**
    NOTA: Las secciones críticas (TEMA 6)
  - **Memoria compartida**: comparten **variables de manera directa** sin necesidad de empaquetar y desempaquetar los datos.
    IMPORTANTE: Se puede acceder a la información **en tiempo real**, siempre y cuando otro proceso no haya otro proceso manipulando los datos. En esos casos la información se vería alterada
    RECOMENDABLE para computación

Memoria distribuida

- **Memoria compartida**



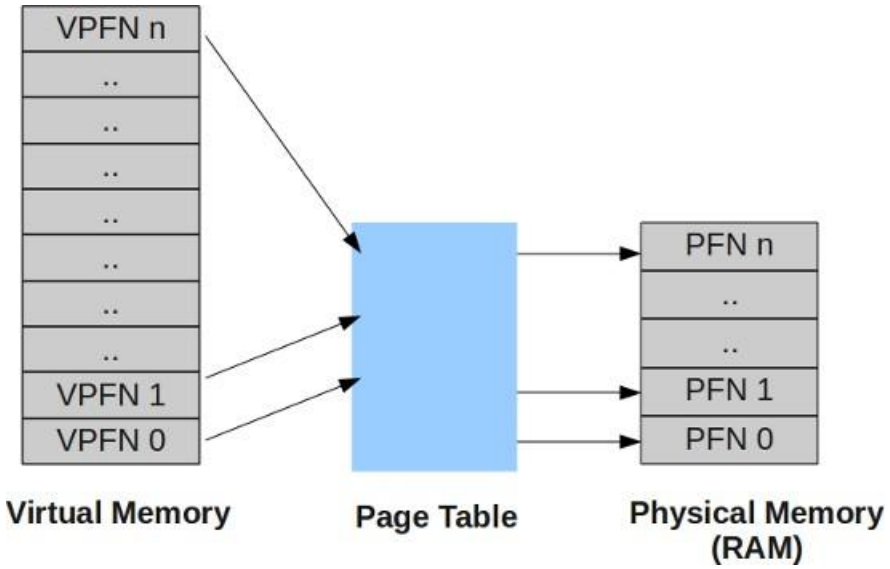
¿Cómo lo implementamos?

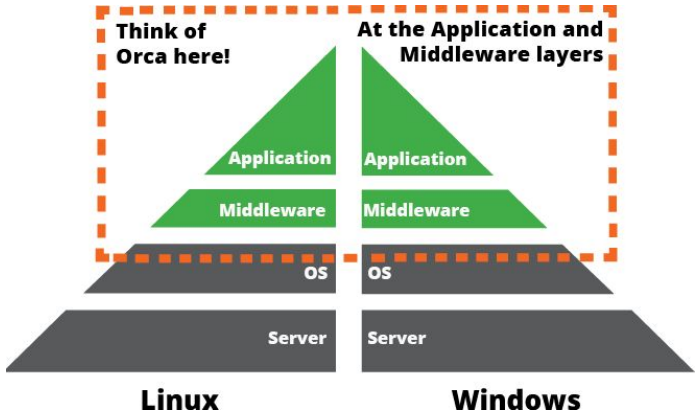
Memoria distribuida

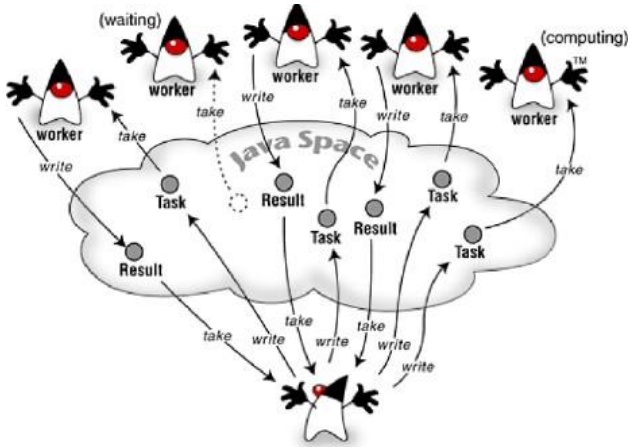
- **Implementación** de la memoria compartida:
  - **Hardware especializado**: desde el punto de vista es un único banco de memoria pero físicamente están separados
  - **Memoria virtual paginada**: memoria virtual que recurre al mismo rango de direcciones para cada proceso. Recurre a la **tabla de páginas** para relacionar las **direcciones físicas** con las **direcciones virtuales**
    - Permite reprogramación de las direcciones
    - Aparenta mayor capacidad desde el punto de vista del sistema
    - Sobrecarga el procesado del sistema
  - **Middleware**: lenguajes como Linda, Orca o JavaSpaces

Memoria distribuida

- **Implementación** de la memoria compartida:





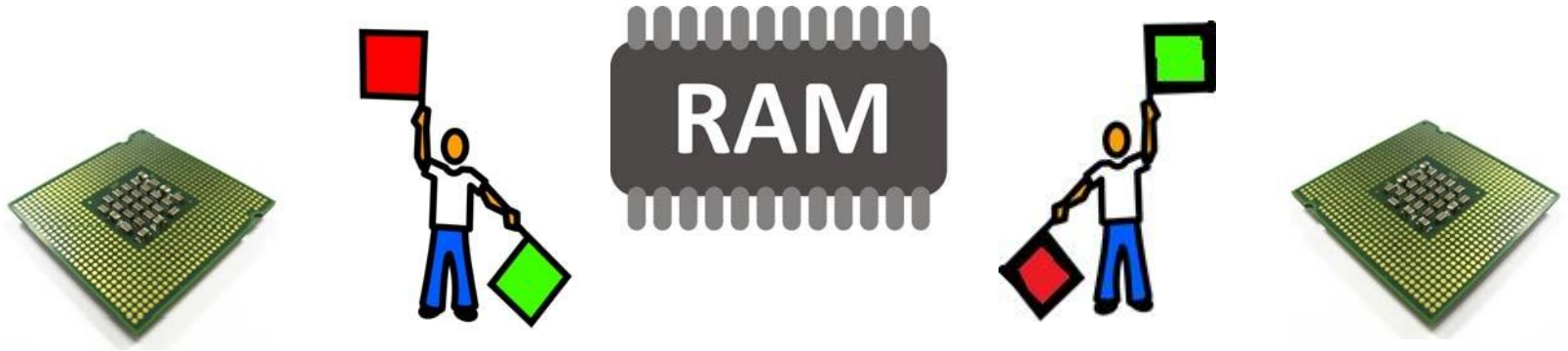


Memoria distribuida

- **Estructura** de los sistemas de memoria compartida:
  - **Orientada a byte**: Se puede almacenar **cualquier tipo de estructura de datos**
    - Se utiliza como la memoria convencional, **sólo se lee y se escribe**
  - **Orientada a objetos:** Recurre a una pila de objetos que se comparten aplicando **serialización**
    - Utilizado por el middleware Orca
  - **Orientada a datos inmutables:** Pueden leer y extraer información pero **no modificarla**
    - Proporcionan al programador una **colección de tuplas** llamada espacio de tuplas en la que pueden existir **cualquier combinación de tipos de datos**
      - JavaSpaces y Linda

Memoria distribuida

- **Sincronización** de los sistemas de memoria compartida:
  - Incluyen un sistema de **sincronización** distribuida con **semáforos** y **bloqueos** con el fin de que dos procesos **no alteren el valor** de una variable **de forma indeseada**



Memoria distribuida

- **Motivación para replicar información**:
  - Confiabilidad: mantener varias copias se hace posible proporcionar una mejor protección contra datos corruptos
  - Rendimiento: permite escalar en números y en área geográfica
- **Desventajas**:
  - Problemas de consistencia: Siempre que se modifica una copia, ésta se vuelve diferente al resto de copias
  - Para mantener copias actualizadas se requiere un mayor ancho de banda en la red

Memoria distribuida

- **Consistencia** de los sistemas de memoria compartida:
  - Un **gestor de réplica local** (que controla las copias en la caché) se implementa mediante el **middleware** y núcleo.
  - Pueden existir aplicaciones con un cierto **nivel de inconsistencia aceptable**: por ejemplo sistema que almacenan el valor de la carga de las CPU

Ejemplo de sistema inconsistente:

P1 ↓
```
br=b;
ar=a;
if (ar>br) print("OK");
```

```
a=a+1
b=b+1
```
↓ P2

- En un sistema distribuido los valores de a y b podrían actualizarse en otro orden diferente y hacer que la condición del proceso 1 no fuese siempre cierta

Memoria distribuida

- **Consistencia** de los sistemas de memoria compartida:
  - **Nivel de inconsistencia aceptable**:

    Consistencia continua: desviaciones como rangos pactados a nivel de aplicación

    - Desviación numérica absoluta
    - Desviación numérica relativa
    - Desviación de la ordenación en la actualización
    - Desviación vieja

  - Los desarrolladores deben especificar los requerimientos de consistencia necesarios para sus aplicaciones

Memoria distribuida

- **Consistencia** de los sistemas de memoria compartida:
  - **Modelos de consistencia:**
    - Consistencia momentánea: Sólo uno, o muy pocos procesos realizan operaciones de actualización (bases de datos principalmente de lectura)
      - No existen los conflictos de escritura-escritura
      - Sólo existen los conflictos de lectura-escritura
      - Ejemplo: sistema DNS
    - Consistencia centrada en el cliente: proporciona garantías para un solo cliente. No se proporciona garantía alguna con respecto a accesos concurrentes de diferentes clientes

Memoria distribuida

- **Consistencia** de los sistemas de memoria compartida:
  - **Modelos de consistencia:**
    - Consistencia centrada en el cliente: proporciona garantías para un solo cliente. No se proporciona garantía alguna con respecto a accesos concurrentes de diferentes clientes
      - Consistencia de lectura monotónica: si un proceso ha visto un valor de *x* el tiempo *t*, nunca verá una versión más vieja que *x* en un tiempo posterior. Ejemplo: información meteorológica
      - Consistencia de escritura monotónica: una operación de escritura hecha por un proceso sobre un elemento *x* se completa antes que cualquier otra operación sucesiva de escritura sobre *x* realizada por el mismo proceso. Si es necesario, la nueva escritura debe esperar a que terminen otras escrituras anteriores. Ejemplo: banca

Memoria distribuida

- **Consistencia** de los sistemas de memoria compartida:
  - **Modelos de consistencia:**
    - Consistencia centrada en el cliente: proporciona garantías para un solo cliente. No se proporciona garantía alguna con respecto a accesos concurrentes de diferentes clientes
      - Consistencia lea sus escrituras: una operación de escritura siempre se completa antes que una operación de lectura sucesiva del mismo proceso. Ejemplo: actualización de contraseñas
      - Consistencia las escrituras siguen a las lecturas: cualquier operación sucesiva de un proceso sobre un elemento de datos *x* se realizará sobre una copia de *x* que está actualizada con el valor más recientemente leído por ese proceso. Ejemplo: difusión de notificaciones (asegurar que se leen todas, incluidas las antiguas)

Memoria distribuida

- **Consistencia** de los sistemas de memoria compartida:
  - La secuencia de **operaciones entrelazada** cumple la especificación de una **única copia correcta** de los objetos
  - El **orden de las operaciones** en el entrelazado es consistente con los instantes reales de tiempo en los que las operaciones ocurrieron en la **ejecución actual**

  - Consistencia **secuencial**:
  - El **orden de las operaciones** es consistente con el **orden de ejecución** por cada **cliente** individual
  - Se requiere un **único servidor para gestionar todos los datos compartidos** y haciendo que los procesos envíen las solicitudes de lectura y escritura al servidor de forma global. **Servidor centralizado**

Memoria distribuida

- **Consistencia** de los sistemas de memoria compartida:
  - **Coherencia**: forma más **débil de consistencia**
  - Cada proceso llega a un **acuerdo sobre el orden de las operaciones** de escritura en la misma posición
  - Es como una consistencia secuencial **posición a posición**, por lo que el **acceso a dos páginas diferentes se realiza de forma independiente**

Memoria distribuida

•**Actualización** de los sistemas de memoria compartida:

Escritura actualizante: las actualizaciones de un proceso se realizan de **forma local** y se envían por **multidifusión a todos los gestores de réplica que poseen una copia** del dato. También pueden **escribir el mismo dato de forma simultánea** (compartición **múltiples lectores**/**múltiples escritores**)

Orca utiliza escritura actualizante mediante el protocolo de multidifusión

Escritura invalidante: **múltiples lectores**/ **único escritor**

Dos opciones: leída por todos los nodos o leída y escrita por un único nodo.

El proceso se bloquea y no vuelve a utilizarse hasta que la escritura termina

Sólo se **propaga cuando los datos son leídos**

Aunque es **más lento en las escrituras**, la **relación escrituras lecturas es baja** por tanto hace que siga siendo rentable

Memoria distribuida

- **Actualización** de los sistemas de memoria compartida:

  Granularidad:
  Todos los programas comparten la memoria entera, sin embargo **únicamente ciertas partes de los datos son en realidad compartidas** y únicamente durante ciertos periodos de tiempo
  Dividir en páginas más pequeñas no hace más eficiente el intercambio de la información, pero sí que **reduce la probabilidad** de que otro nodo requiera también la información contenida en la página

  Trashing: (sólo en escritura invalidante)
  Ocurre cuando un proceso está **escribiendo continuamente un dato que otro está leyendo continuamente**, esto hace que se invalide la información y que el sistema se vuelva ineficiente. En estos casos se recomienda **escritura actualizante**

Memoria distribuida

•**Actualización** de los sistemas de memoria compartida:

Operaciones:

1. Propagar la notificación de una actualización
   - Protocolo de invalidación: informa a otras copias que ha ocurrido una actualización, y que los datos que ellas contienen ya no son válidos
   - Replicación activa: cada réplica dispone de un proceso que mantiene actualizados sus datos asociados
2. Transferir datos de una copia a otra
3. Propagar la operación de actualización hacia otras copias
   - Pull: un servidor o un cliente solicita a otro servidor que le envíe cualquier actualización que tenga hasta el momento
   - Push: las actualizaciones se propagan hacia otras réplicas sin que éstas lo soliciten

Memoria distribuida

- **Protocolos:**
  - Protocolos de escritura replicados: cada réplica tiene un proceso asociado que realiza operaciones de actualización
    - Las operaciones deben realizarse en el mismo orden en cualquier parte
    - Secuenciador: le asigna un número de secuencia único (tiempo lógico, Lamport, TEMA 6) a cada operación y posteriormente remite la operación a todas las réplicas
  - Protocolos basados en quórum: los clientes deben adquirir el permiso de varios servidores antes de leer o escribir un elemento de datos replicado