Sistemas Distribuidos

Grado en Ingeniería Informática

Sistemas de archivos distribuidos

Departamento de Ingeniería Informática
Universidad de Cádiz

Curso 2014 – 2015

Indice

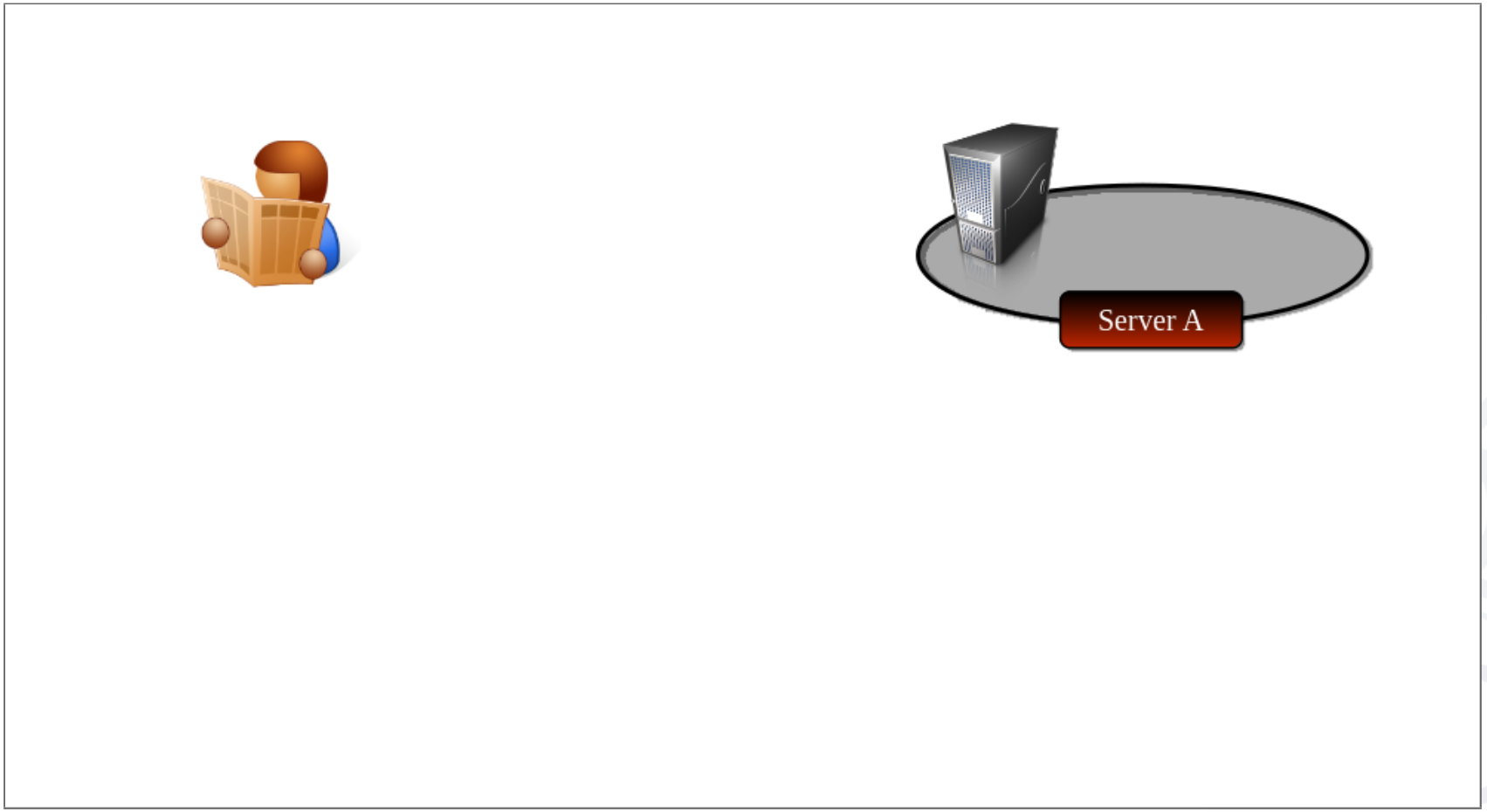
Sección 1 | Introducción

Introducción

- **Objetivo** Compartir información almacenada en localizaciones distribuidas
- **Enfoques de almacenamiento**
  - **No persistente:**
    - Almacenamiento en memoria compartida
    - Almacenamiento de objetos: CORBA, EJB, etc.
  - **Persistente:**
    - Datos no estructurados (Ficheros) (NFS, xFS..gNutella, Peer-to-Peer)
    - Datos Estructurados: BBDD distribuidas/federadas/paralelas, Datawarehousing

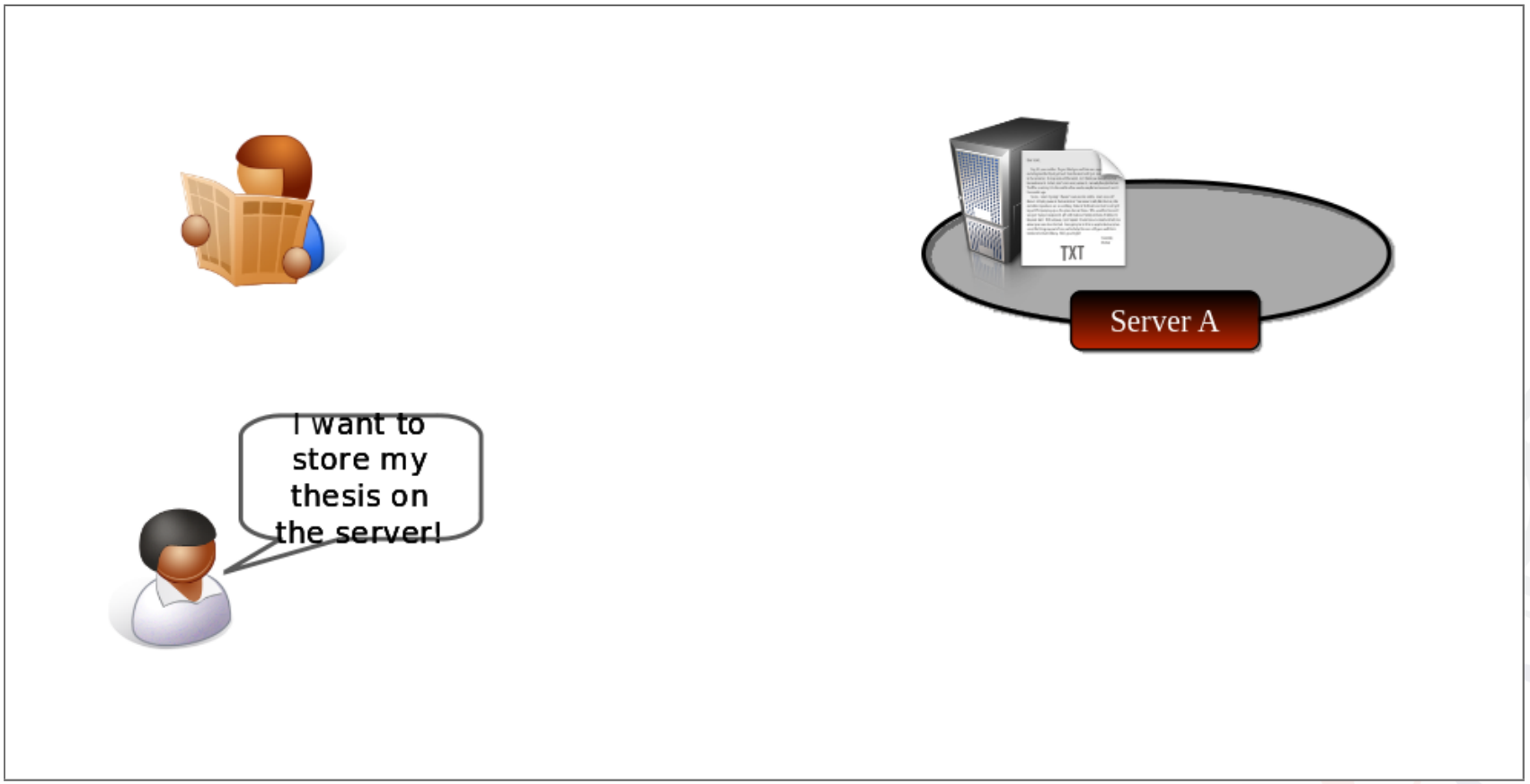
Introducción

Un caso de SD



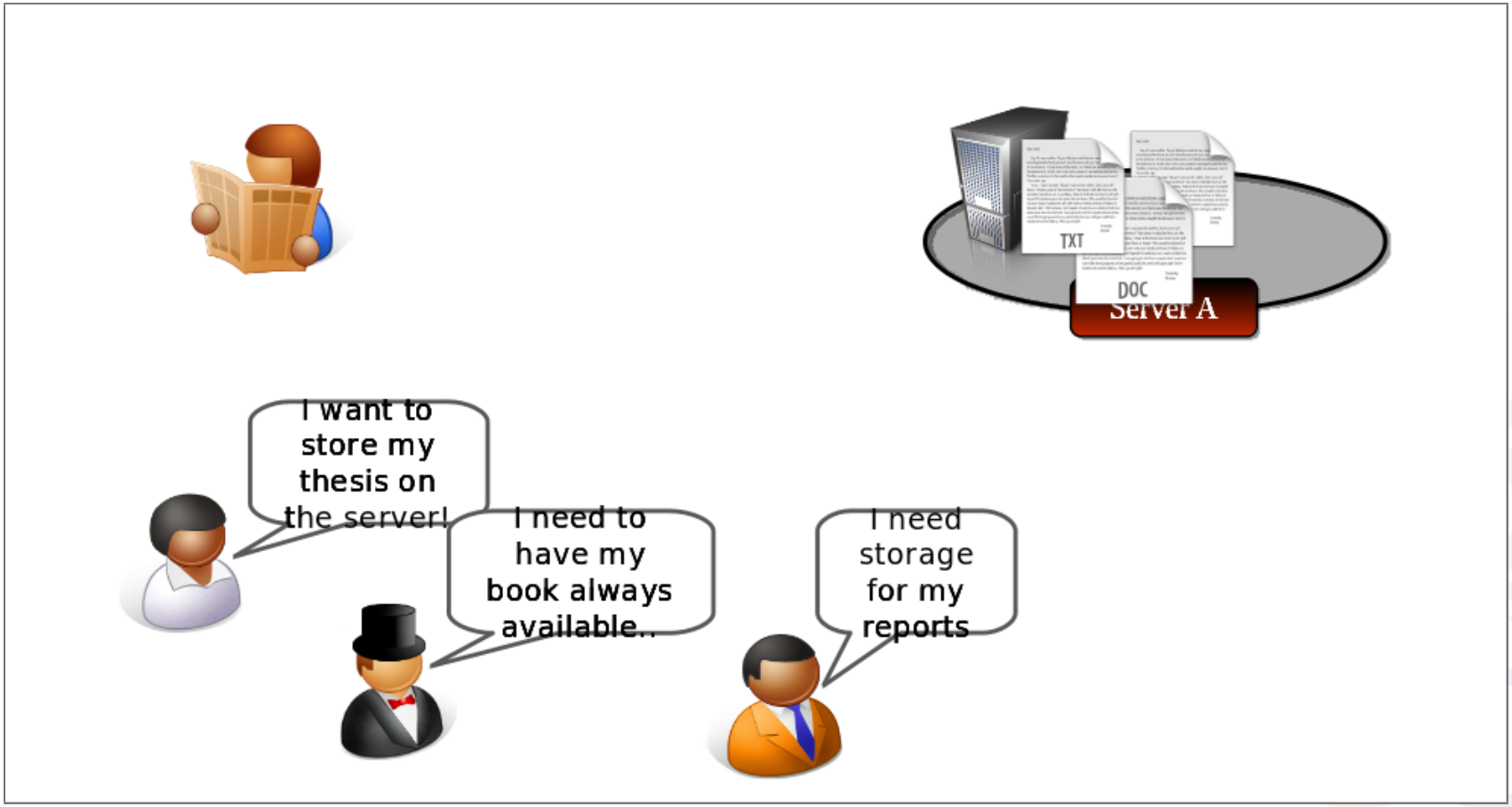
Introducción

Un caso de SD



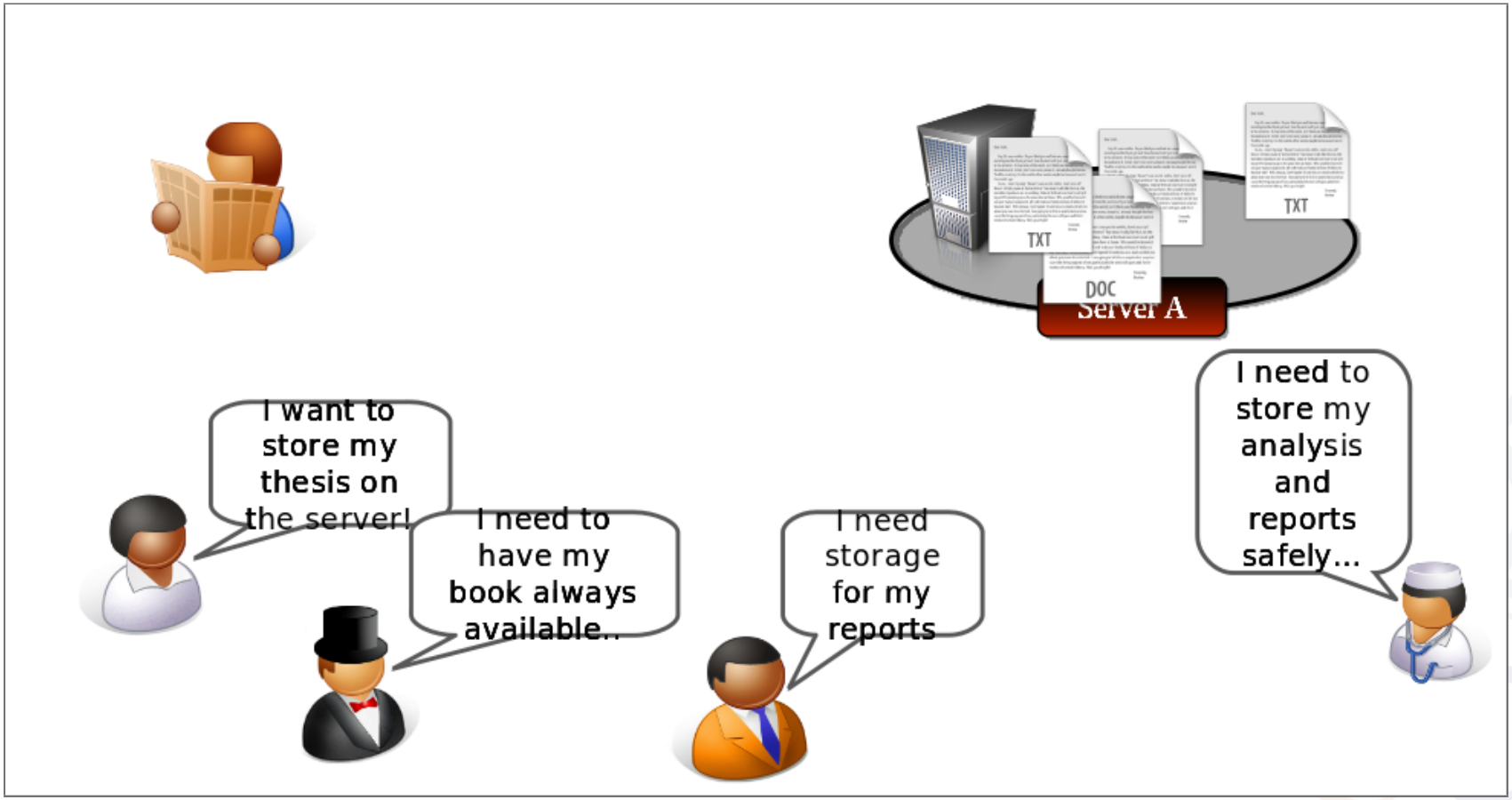
Introducción

Un caso de SD



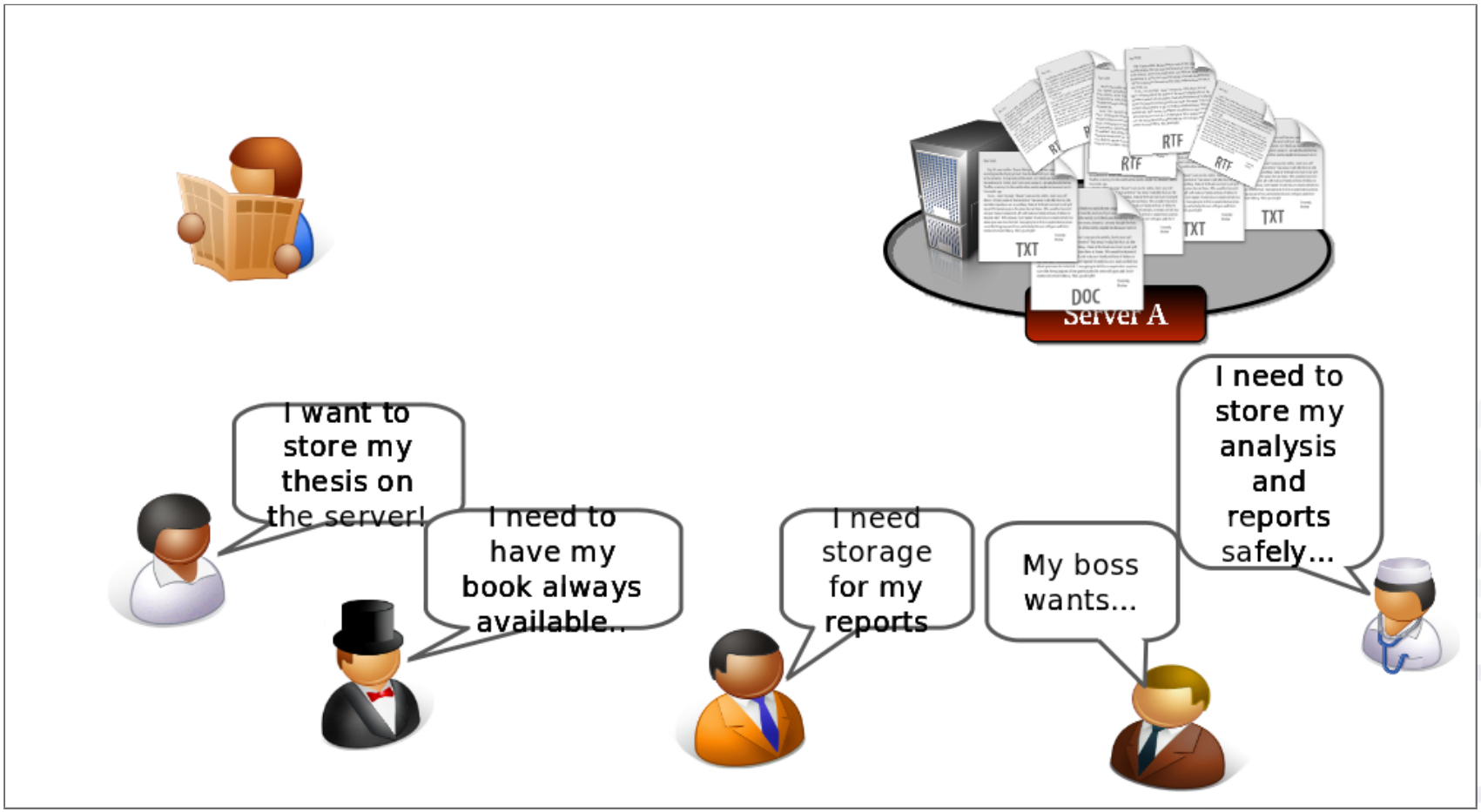
Introducción

Un caso de SD



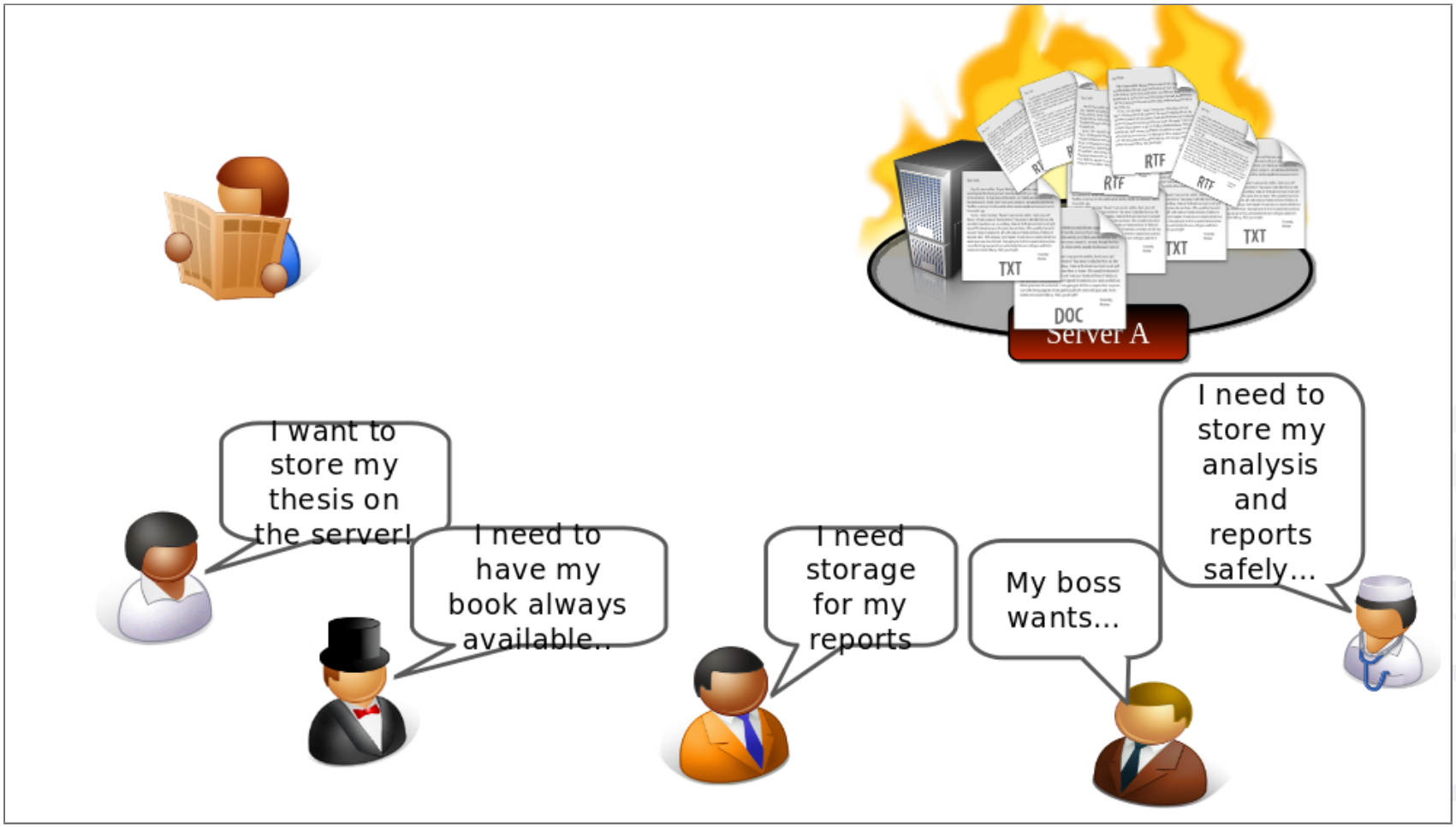
Introducción

Un caso de SD



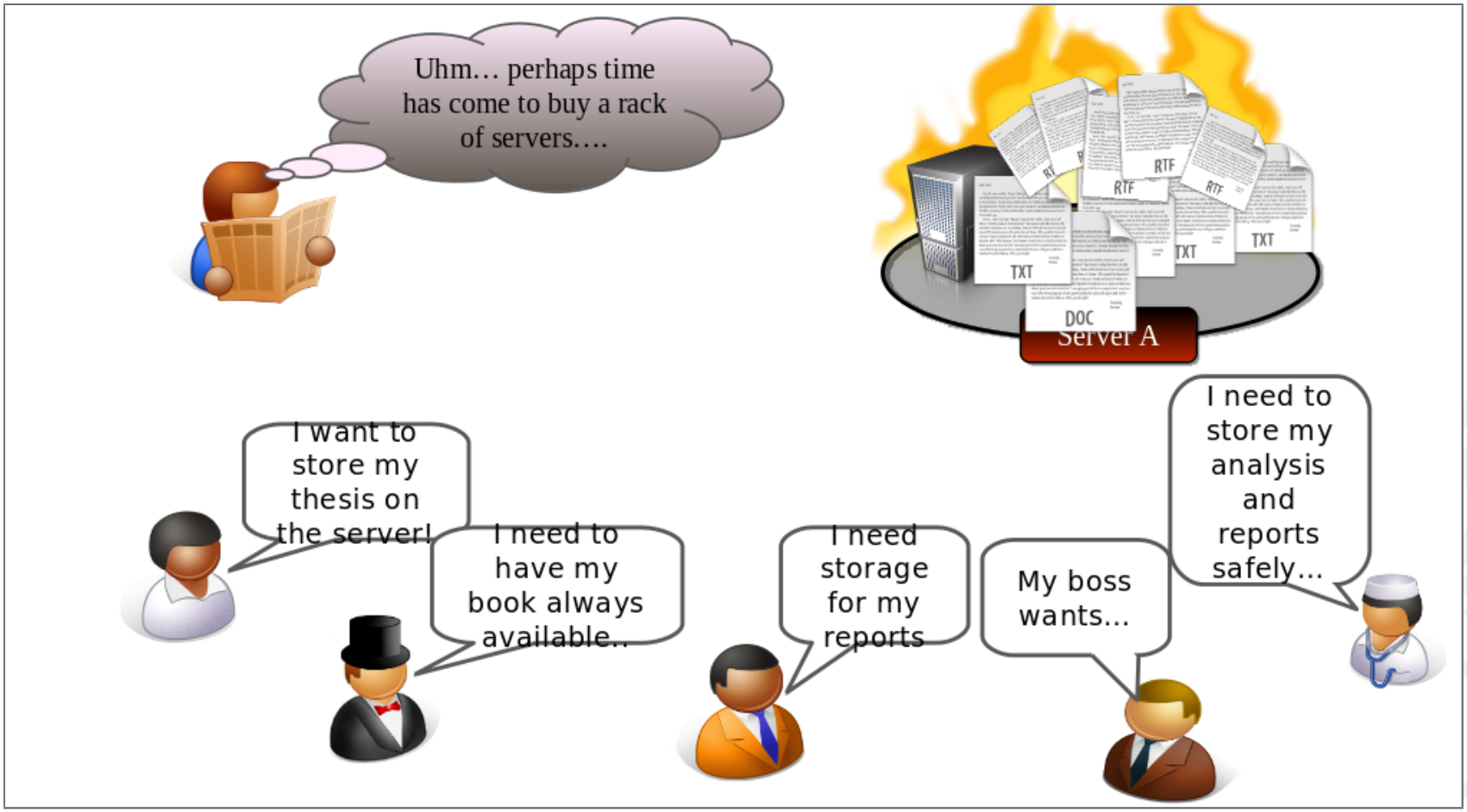
Introducción

Un caso de SD



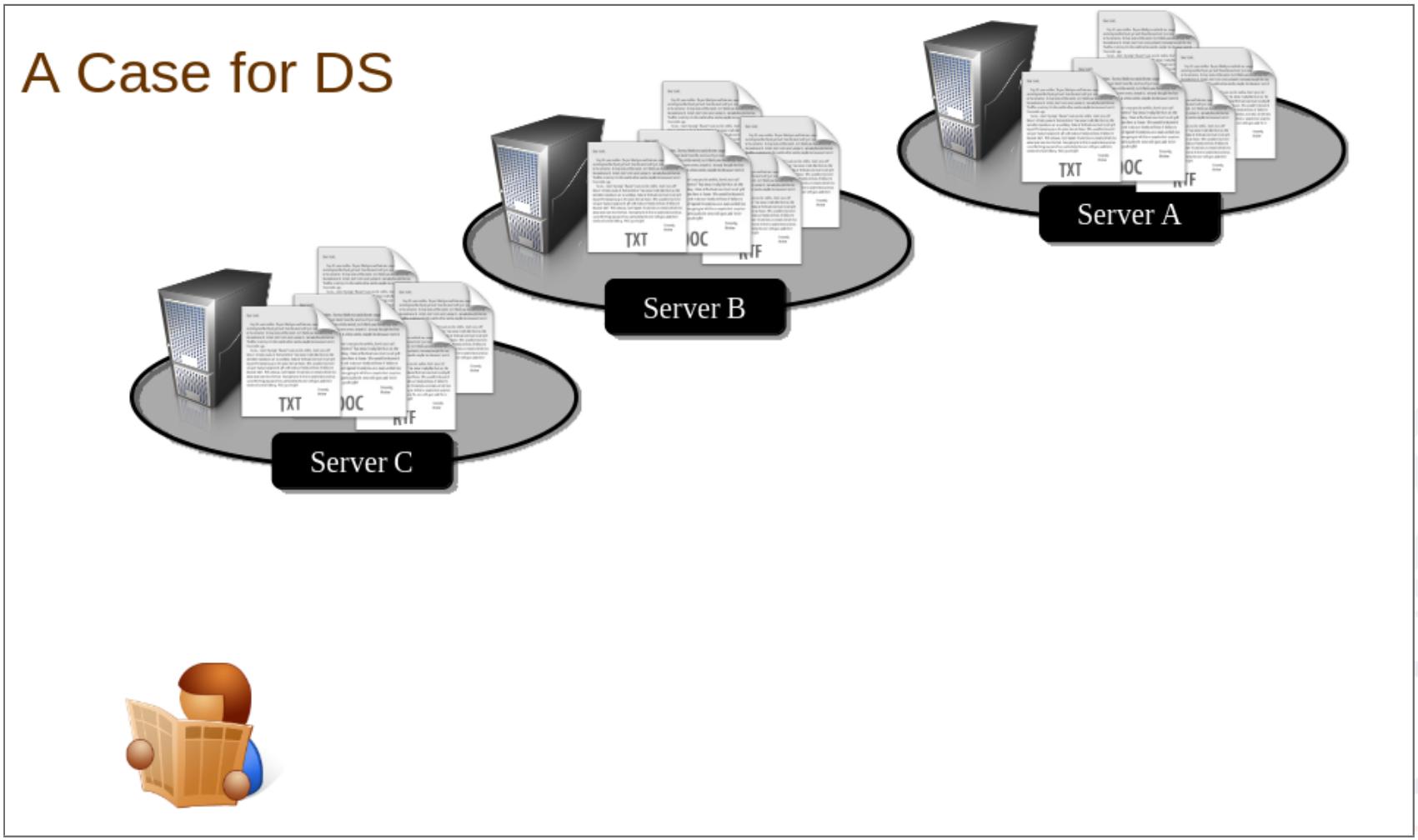
Introducción

Un caso de SD



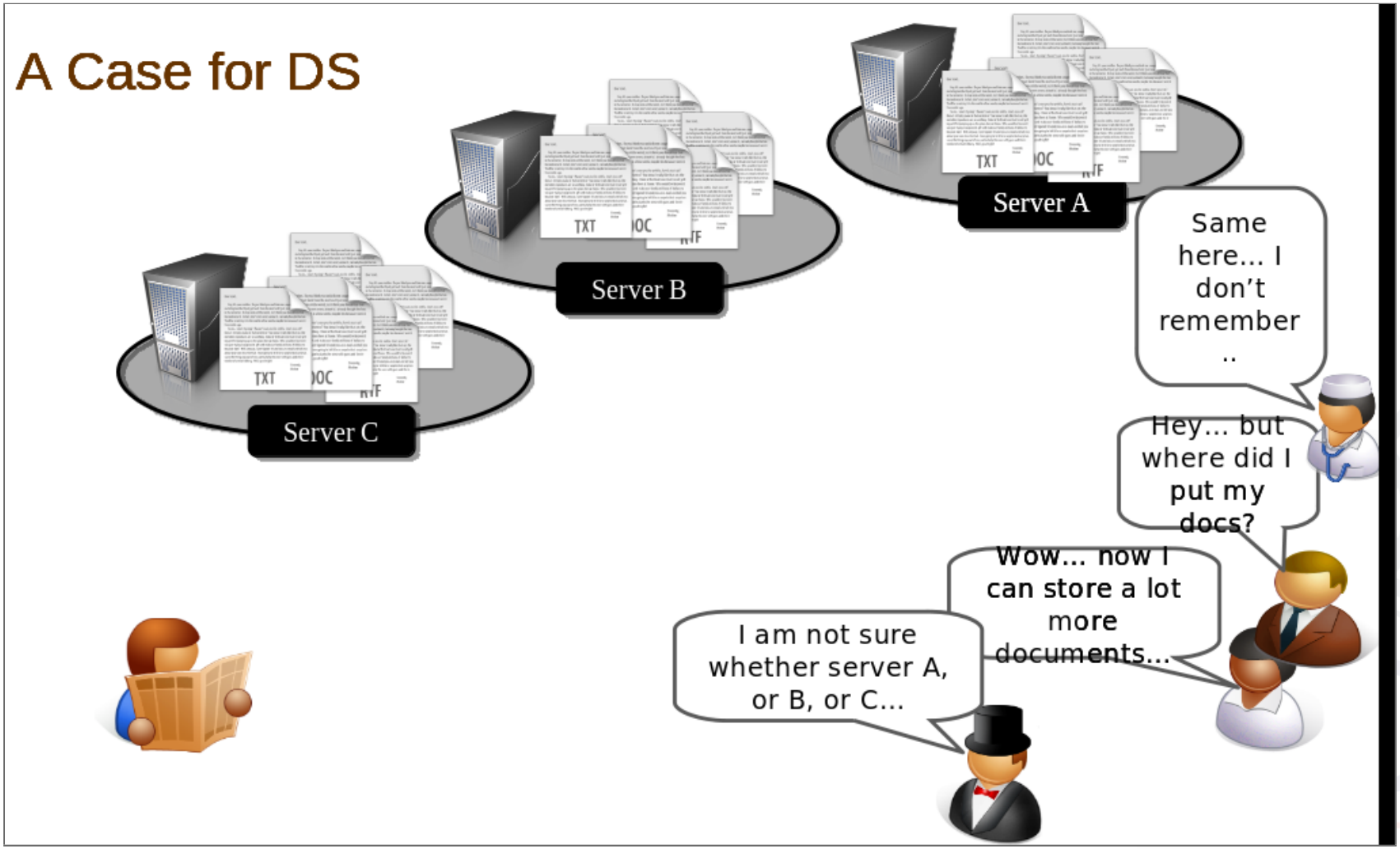
Introducción

Un caso de SD



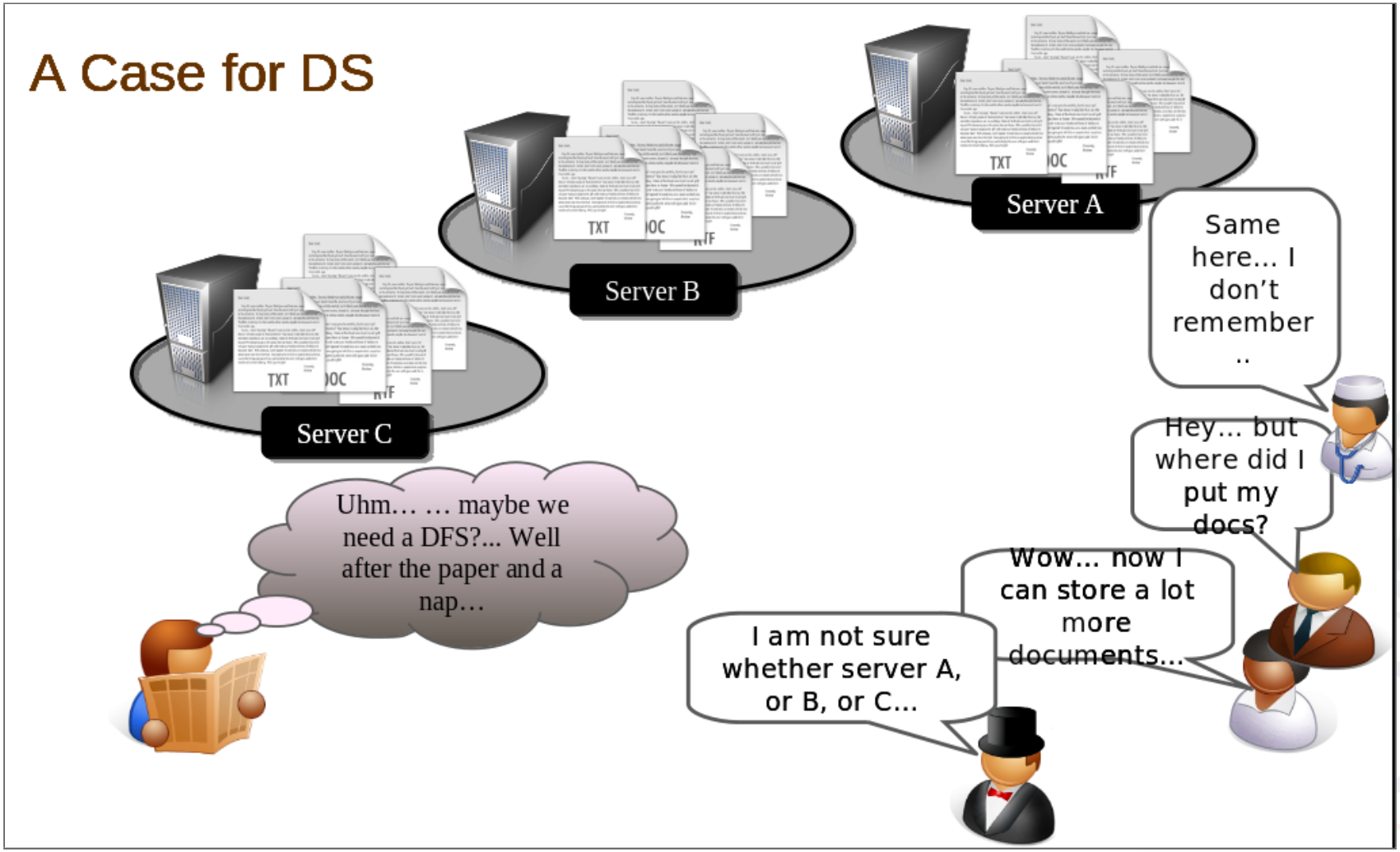
Introducción

Un caso de SD



Introducción

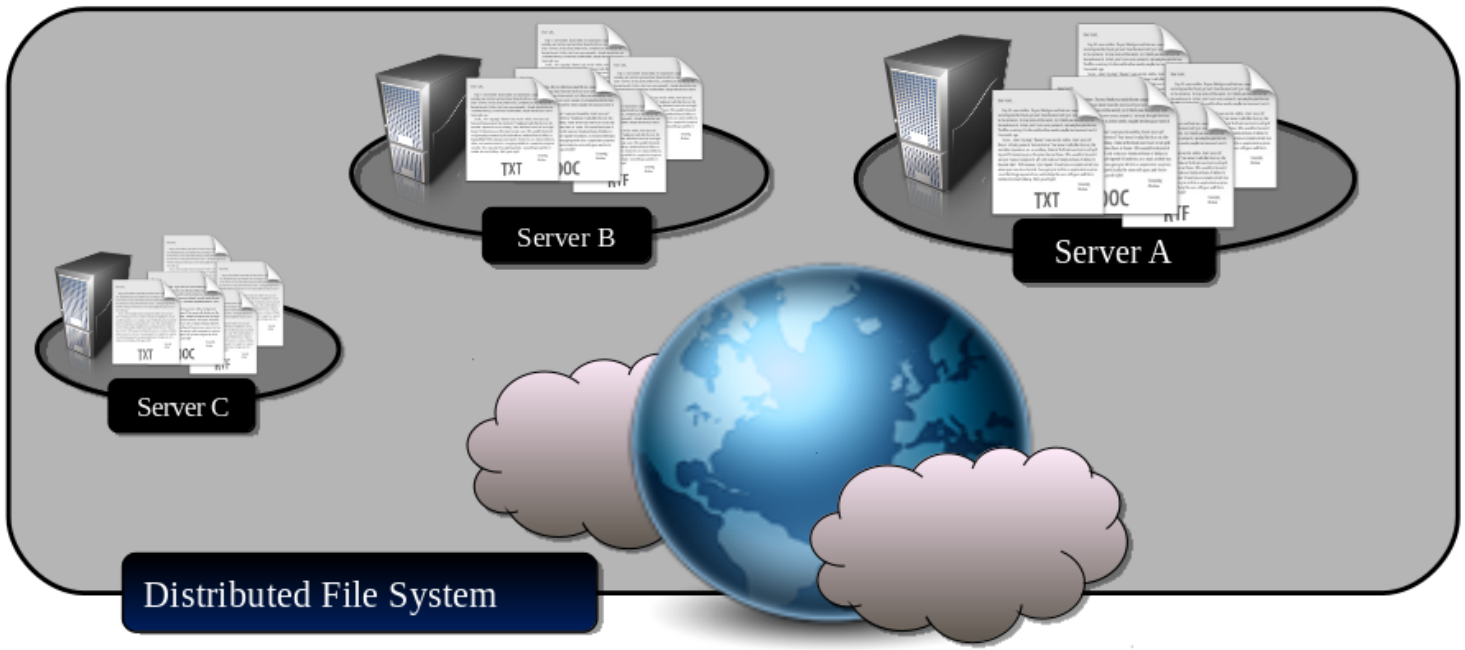
Un caso de SD



Introducción

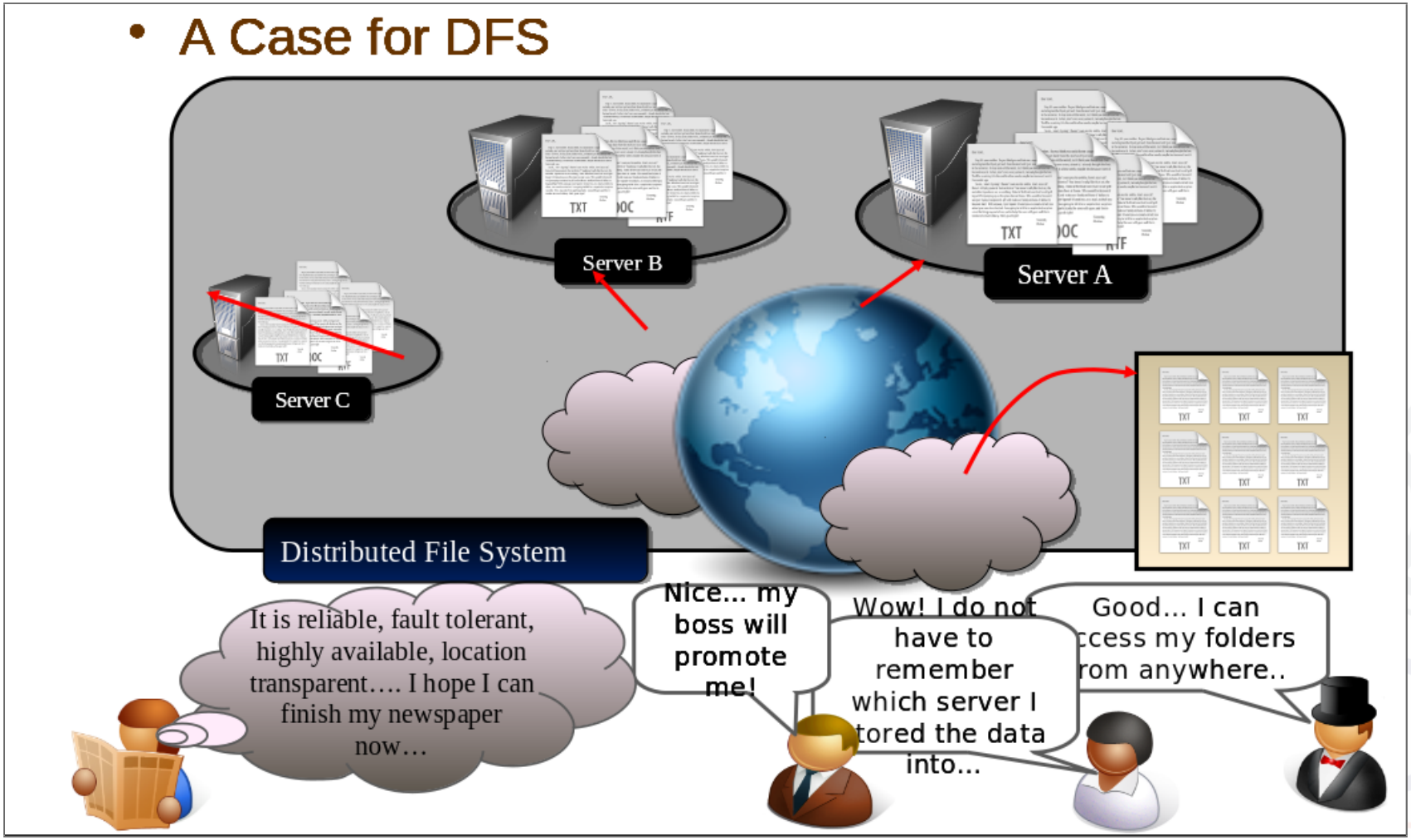
Un caso de SD

- A Case for DFS



Introducción

Un caso de SD



Introducción

Conceptos

- Caracterización
  - **Sistema o servicio de ficheros:** software cuyo propósito es emular la funcionalidad de un sistema de ficheros no distribuido
  - **Objetivo:** permitir a programas clientes acceder a los ficheros como si estuvieran en su nodo local
- Conceptos implicados
  - **Ficheros** elementos almacenados formados por “datos” (secuencias de bits) y “atributos” (longitud, tipo, timestamp, propietario, ACL, etc.)
  - **Directorios** ficheros especiales que contienen nombres (id) de ficheros dependientes de él
  - **Metadatos:** información usada para la gestión del sistema de ficheros
  - **Operaciones:** crear, abrir, cerrar, leer escribir, posicionar, eliminar, renombrar, asignar atributos.

Introducción

Requisitos

- Transparencia: equilibrio entre flexibilidad y escalabilidad de la transparencia frente a complejidad y rendimiento del SW
  - **De acceso:** los clientes no deben ser conscientes de la distribución de los ficheros → conjunto único de operaciones
  - **De localización:** los clientes deben ver un único espacio de nombres de ficheros uniforme → transferencia de ficheros sin que cambie el "pathname"
  - **De movilidad:** información en los clientes acerca de los ficheros no debe variar cuando se migra un fichero
  - **De rendimiento:** los clientes deben seguir con su rendimiento aun cuando la carga del servicio de ficheros distribuidos aumenta
  - **De escalabilidad:** el sistema de ficheros debe ser capaz de escalar sin que mermen sus capacidades

Introducción

Requisitos

- Concurrencia: Los cambios hechos a un fichero por parte de un cliente no deben afectar al acceso a ese fichero por parte de otros clientes

- Replicación: un mismo nombre de fichero puede referirse a varios ficheros replicados.
  - **Ventaja:** reparto de carga y tolerancia a fallos
  - **Desventaja:** actualizaciones y mantenimiento de copias
- Heterogeneidad
  - El servicio puede ser accedido por clientes que utilizan plataformas o sistemas operativos diferentes
  - El diseño debe ser compatible con los sistemas de ficheros de los diferentes sistemas operativos
  - Se necesita saber la especificación de API para acceder a los servicios

Introducción

Requisitos

- Tolerancia a fallos: recuperación ante fallos del cliente o del servidor
- Consistencia: debido a la existencia de replicación, todas las copias de un fichero deben ser iguales → propagación de cambios
- Seguridad: mecanismos de control de acceso a los ficheros
- Eficiencia: un sistema de ficheros distribuidos debe ofrecer, al menos, la misma eficiencia que uno local

Evolución

1974-1995

En la primera generación de los sistemas distribuidos, los sistemas de ficheros (como NFS), eran solo sistemas de almacenamiento en red.

1995-hoy

Con los avances de los sistemas de objetos distribuidos (como **CORBA, Java**) y la web han creado sistemas más complejos

Actualmente

Los sistemas a gran escala y con almacenamiento escalable

- Google File System
- Amazon S3
- Cloud Storage (e.g., Dropbox)

Introducción

Sistemas de almacenamiento y sus propiedades

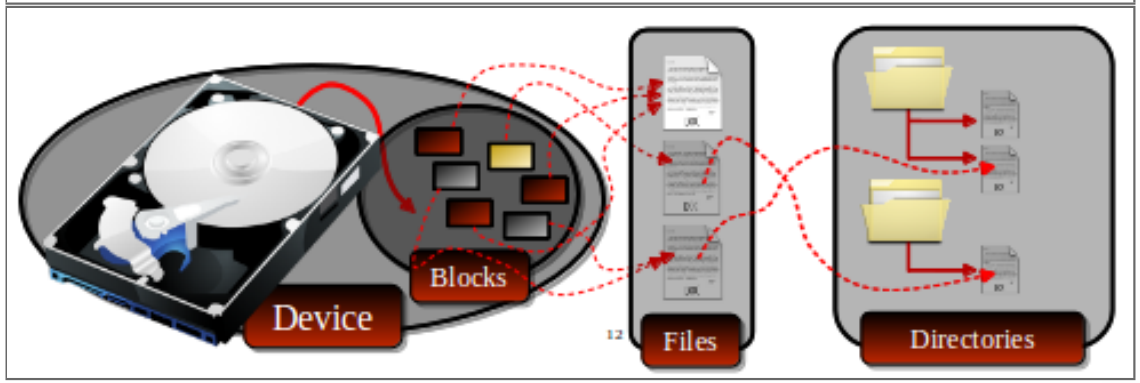
	Sharing	*Persistence*	*Distributed cache/replicas*	*Consistency maintenance*	*Example*
Main memory	✗	✗	✗	1	RAM
File system	✗	✓	✗	1	UNIX file system
Distributed file system	✓	✓	✓	✓	Sun NFS
Web	✓	✓	✓	✗	Web server
Distributed shared memory	✓	✗	✓	✓	Ivy (DSM, Ch. 6)
Remote objects (RMI/ORB)	✓	✗	✗	1	CORBA
Persistent object store	✓	✓	✗	1	CORBA Persistent State Service
Peer-to-peer storage system	✓	✓	✓	2	OceanStore (Ch. 10)

Types of consistency:
1: strict one-copy ✓: slightly weaker guarantees 2: considerably weaker guarantees

Sistema de ficheros

Módulos típicos para un sistema no Distribuido

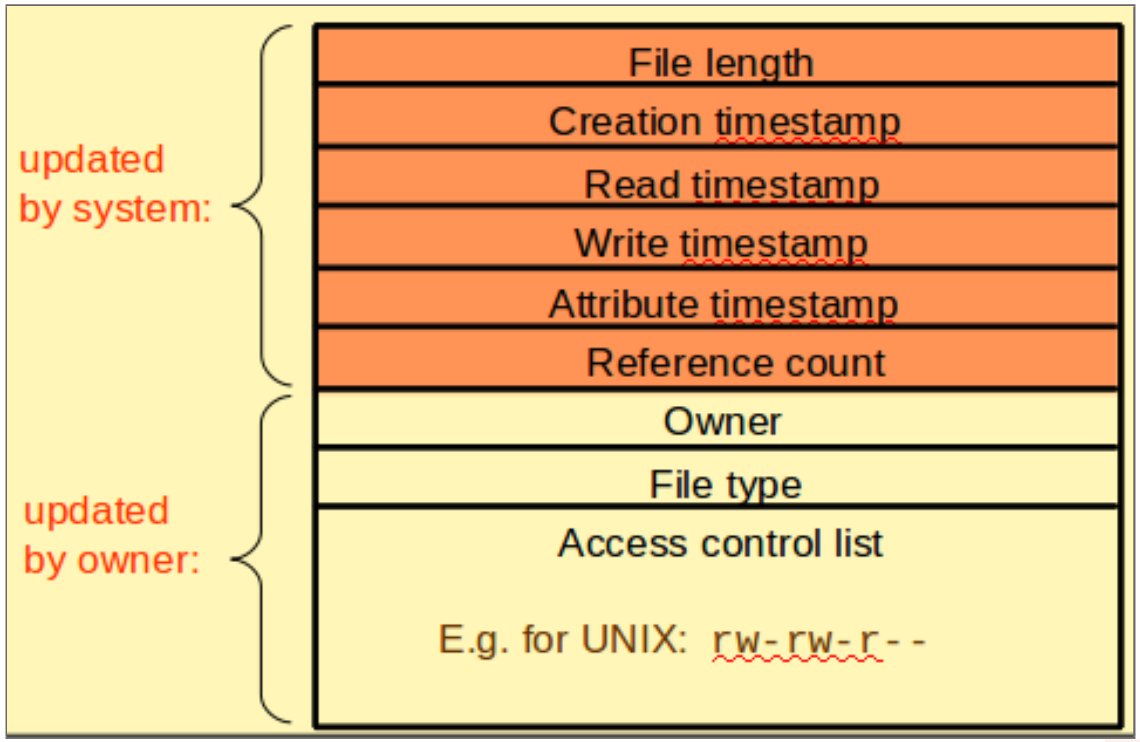
Directory module:	relates file names to file IDs
File module:	relates file IDs to particular files
Access control module:	checks permission for operation requested
File access module:	reads or writes file data or attributes
Block module:	accesses and allocates disk blocks
Device module:	performs disk I/O and buffering



Sistema de ficheros

Estructura típica del registro de atributos

Los sistemas de archivos son responsables de la organización, almacenamiento, recuperación, nominación, compartición y protección de los archivos. Los archivos contiene datos y atributos



Ejemplos de sistemas de ficheros distribuidos

- Arquitectura de servicio de ficheros:
  - Modelo arquitectónico genérico sobre el que se asientan NFS o AFS
  - Basado en la división de responsabilidades entre el cliente y el servidor
- NFS: Network File System
  - Idea de Sun Microsystems (1985)
  - Protocolo NFS: RFC 1813 (1995)
  - Relación simétrica entre clientes y servidores (multiplataforma)
- AFS: Andrew File System
  - Entorno de computación distribuido de la CMU (1986)
  - Basado en la transferencia de grandes bloques de ficheros y el uso de cachés

Sección 2 | Arquitectura de Servicio de Ficheros

Arq. de Servicio de Ficheros

Componentes

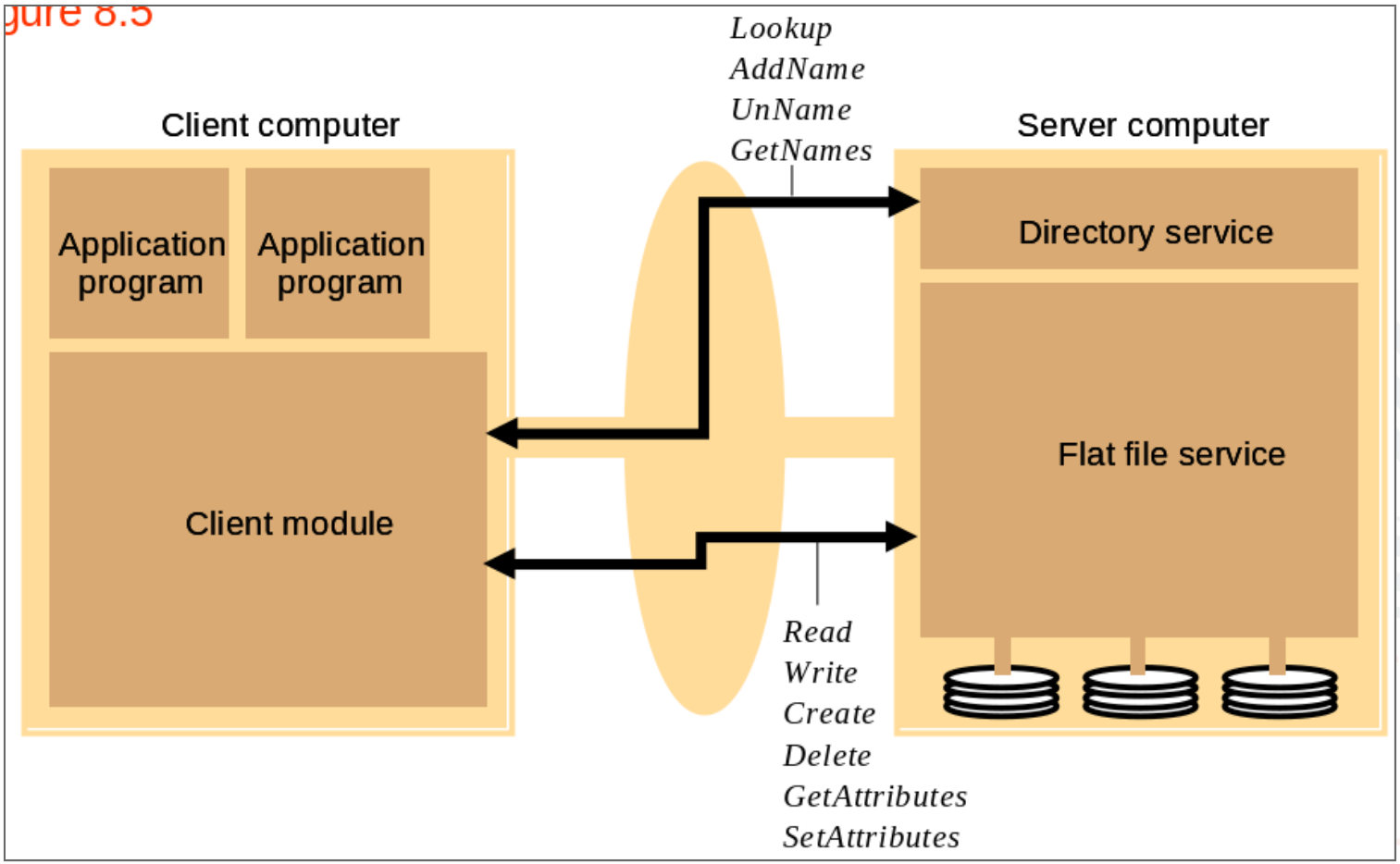
El alcance de la extensibilidad y configurabilidad se mejora si el servicio se estructura en tres componentes encargados de lo concerniente al acceso a ficheros

- Servicios de Ficheros Planos
  - Implementa las operaciones en los contenidos de los ficheros para su acceso
  - Crear **UFIDs** (Unique File identifiers) para representar cada fichero físico
- Servicios de directorio
  - Mapea nombres(texto) a UFIDs
  - Gestiona ficheros y directorios
- Módulo cliente
  - Integra y extiende las operaciones del servicio de ficheros planos y del servicio de directorio como un simple API.
  - Crea un interfaz común de acceso para aplicaciones cliente..
  - Mantiene información sobre las localizaciones en la red.

Sistema de ficheros

Estructura típica del registro de atributos

Los módulos y sus relaciones.



Sistema de ficheros

Interfaces/operaciones del Servicio

- Interfaz del servicio de ficheros planos
  - Conjunto de operaciones usadas por un cliente
  - Basado en RPC (usado por el cliente) en las que se indica un UFID (global del sistema de ficheros distribuido pero resuelto por un servidor concreto)
- Interfaz del servicio de directorio
  - Propósito: ofrecer una forma de trasladar nombres de ficheros planos a UFIDs
  - Ofrece también operaciones sobre directorios.

Sistema de ficheros

Interfaces/operaciones del Servicio

Flat file service

Read(FileId, i, n) -> Data

(i: position of first byte)

Write(FileId, i, Data)

(i: position of first byte)

Create() -> FileId

Delete(FileId)

GetAttributes(FileId) -> Attr

SetAttributes(FileId, Attr)

Directory service

Lookup(Dir, Name) -> FileId

AddName(Dir, Name, FileId)

UnName(Dir, Name)

GetNames(Dir, Pattern) -> NameSeq

Sistema de ficheros

Interfaces/operaciones del Servicio

- Control de acceso
  - OPCIÓN 1: Se realiza cuando se realiza la conversión de un nombre de archivo plano a UFID, los resultados se codifican en forma de habilitación que se devuelve al cliente para su envió con las solicitudes posteriores
  - OPCIÓN 2: Se envía la identidad del usuario al servidor para su comprobación antes de cada operación de fichero.
- Sistema de archivos jerárquico
- Agrupación de archivos

  Conjunto de ficheros almacenados en un servidor que puede ser referenciado como un todo. Debe ser único a lo largo de un sistema distribuido.

Sistema de ficheros

¿Suficiente?

¿Asunto zanjado? No todo está resuelto:

- Resolución de nombre de fichero:
  Cliente y varios servidores involucrados. ¿Cómo se reparten trabajo?
- Acceso a los datos:
  - ¿Se transfiere sólo lo pedido? ¿más cantidad? ¿todo el fichero?
  - Uso de cache en el cliente. Coherencia entre múltiples caches.
- otros
  - Migración
  - Replicación

Sección 3 | Network File System

Sun Network File System

Introducción

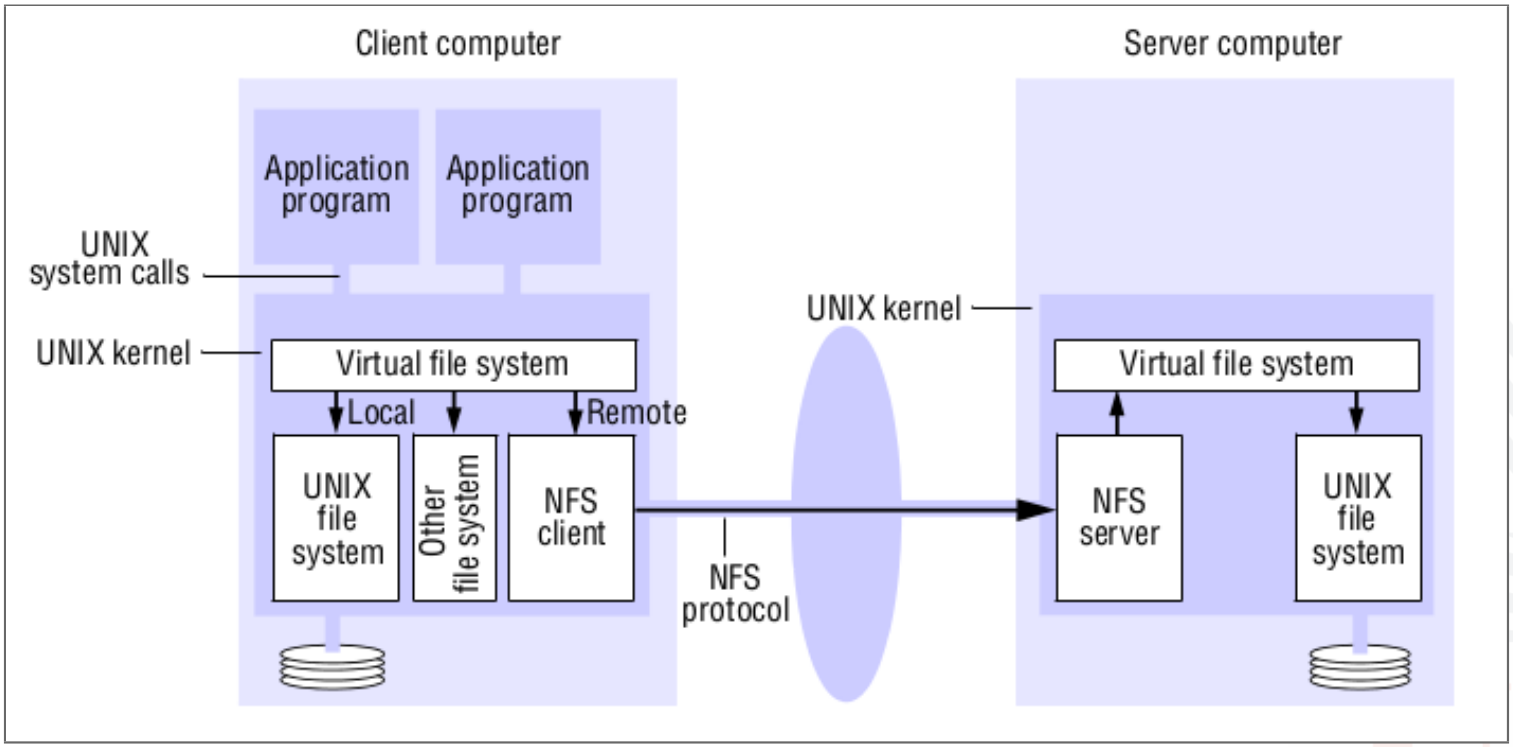
- Desarrollado por Sun Microsystems in 1985
- Muy popular, abierto y ampliamente usado
- Estandarizado: Versión 3 IETF (RFC 1813), Versión 4 RFC-3010

Características

- Soporta mucho de los requerimientos de diseño mencionados
  - Transparencia, Heterogeneidad, Eficiencia, Tolerancia a fallos
- Comportamiento limitado respecto a
  - Concurrencia, Replicación, Consistencia, Seguridad
- El protocolo no ofrece mecanismos de control de concurrencia (Protocolo independiente de NFS: **Network Lock Manager**)

Sun Network File System

Arquitectura NFS



Sun Network File System

Operaciones NFS

El identificador de fichero usado en NFS se llama ***file handles***

- *read(fh, offset, count) -> attr, data*
- *write(fh, offset, count, data) -> attr*
- *create(dirfh, name, attr) -> newfh, attr*
- *remove(dirfh, name) status*
- *getattr(fh) -> attr*
- *setattr(fh, attr) -> attr*
- *lookup(dirfh, name) -> fh, attr*
- *rename(dirfh, name, todirfh, toname)*
- *link(newdirfh, newname, dirfh, name)*
- *readdir(dirfh, cookie, count) -> entries*
- *symlink(newdirfh, newname, string) -> status*
- *readlink(fh) -> string*
- *mkdir(dirfh, name, attr) -> newfh, attr*
- *rmdir(dirfh, name) -> status*
- *statfs(fh) -> fsstats*

fh = file handle:

Filesystem identifier	i-node number	i-node generation

Sun Network File System

Conceptos Básicos

- **Protocolo NFS:** conjunto de llamadas a procedimiento remoto que ofrecen las capacidades necesarias para que los clientes puedan realizar operaciones sobre un almacenamiento de ficheros remoto.
- Protocolo de nivel de aplicación en la pila OSI
- Clientes y servidores NFS:
  - Un servidor y varios clientes
  - Se comunican por RPCs siguiendo el protocolo NFS
  - Comunicación con operaciones síncronas
  - Con TCP o UDP por debajo
- Únicas **restricciones de uso**: que la petición esté bien formulada y que el usuario tenga las credenciales adecuada

Sun Network File System

Componentes

- Sistema de ficheros virtuales (Virtual File System)
  - Módulo que implementa la transparencia de acceso
  - Conoce si debe transmitir una petición de acceso a fichero al sistema local de ficheros o al cliente NFS para acceso remoto (remote filesystem)
- Módulo Cliente NFS (Client NFS)
  - Ofrece una interfaz adecuada para ser utilizada por aplicaciones cliente
  - En UNIX suele estar integrado en el kernel
    - Acceso a ficheros con llamadas al sistema (sin recompilar) no como librería
    - Un único módulo atiende a varias aplicaciones cliente
  - Se comunica y coordina con el sistema de ficheros virtuales

Sun Network File System

Componentes

- Control de acceso y autenticación
  - El servidor NFS no guarda el estado de apertura de un fichero (STATELESS) (Si en la versión 4) → chequeo de permisos ante cada petición cliente
  - Toda consulta por parte del cliente debe ir acompañada del userID y el groupID, insertado estos por el RPC.
  - Problema: cualquiera podría enviar una petición RPC con la ID de otro usuario encapsulada en la petición
  - Solución parcial: Kerberos se ha integrado con Sun NFS para proveerlo de una seguridad más fuerte y comprensiva.

Sun Network File System

Servicio de montado

- Establece una conexión lógica entre el servidor y el cliente
- Cada máquina incluye una “lista de exportación” qué “árboles” exporta y quién puede montarlos
- Petición de montaje incluye máquina y directorio remotos
  - Se convierte la petición en RPC al servidor de montaje remoto
  - Si permiso en lista, devuelve un identificador “opaco” (handle) Cliente no conoce su estructura interna
- La operación de montaje sólo afecta al cliente no al servidor se permiten montajes NFS anidados, pero no “transitivos”
- Aspectos proporcionados por algunas implementaciones:
  - montajes hard o soft: en montaje, si servidor no responde espera ilimitada HARD o plazo máximo de espera SOFT
  - automontaje: no solicita montaje hasta acceso a ficheros

Sun Network File System

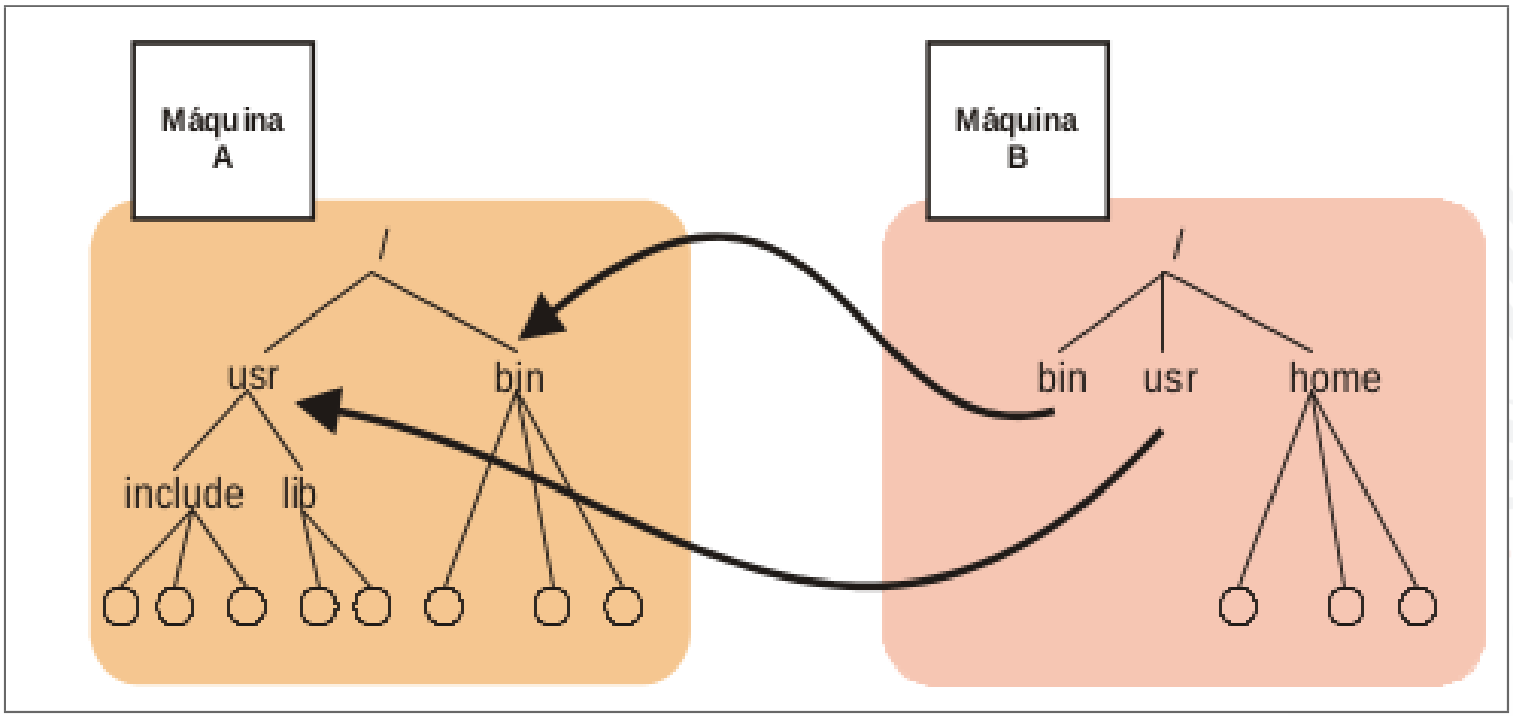
Servicio de montado

- Operación de montado: mount(remotehost, remotedirectory, localdirectory)
- El servidor mantiene una tabla de los clientes que han montado el sistema de ficheros en ese servidor.
- Cada cliente mantiene una tabla de los sistemas de ficheros montados mediante (IP address, port number, file handle)

Sun Network File System

Servicio de montado

- La máquina A exporta /usr y /bin
- Para montar en la máquina B (mount maquina-A:/usr /usr)



Sun Network File System

Automontado

- Añadido en la implementación UNIX para poder montar un directorio remoto dinámicamente cuando un punto de montado, vacio, es referenciado por un cliente.
  - El automontador mantiene una tabla de puntos de montaje y múltiples servidores candidatos para ello
  - Este manda un mensaje a cada servidor candidato y se monta el que primero responda.
- El automontador se encarga de mantener un tabla pequeña de montaje.
- Provee de una manera simple de replicación para sistemas de solo-lectura

Sun Network File System

Seguridad

- El protocolo Keberos es demasiado costoso para aplicarlo en cada acceso a fichero
- Se usa por lo tanto en el servicio de montado
  - Para autenticar la identidad del usuario
  - El userId y el groupId son almacenados en el servidor con la IP del cliente
- Para cada consulta de fichero el userId y el groupID enviados debe coincidir con el almacenado en el servidor, así como la dirección IP
- Este enfoque tiene varios problemas
  - Múltiples usuarios compartiendo la misma máquina cliente
  - Cada vez que un usuario entra en el sistema todos los sistemas remotos deben ser montados.

Sun Network File System

Cache

- Cache en el Servidor
  - Similar a un sistema de ficheros locales UNIX
  - Ofrece dos estratégias de actualizaciones.
    - write-through: Los datos recibidos de los clientes en las operaciones de escritura son almacenados en la memoria caché en el servidor y se escriben en el disco antes de que se envíe una respuesta al cliente. El cliente puede estar seguro de que los datos son almacenados persistentemente
    - delayed commit: Los datos son almacenados en la cache y no se graban en disco hasta que se recibe un **commit()**. Opción por defecto en NFS v3.

Sun Network File System

Cache

- Cache en el cliente
  - La cache en el servidor no reduce el tráfico RPC, por lo que es necesario una optimización
  - Los clientes son responsables de sondear la servidor para comprobar la actualidad de los datos en la caché..
- Para validar los bloques de caché antes de ser utilizados se emplea un método basad en marcas de tiempo.

$$(T - T_c < t) V (T_m cliente = T_m servidor)$$

  - T_c Tiempo en el que la entrada en la caché fue validada últimamente
  - T_m Tiempo en el que bloque fue modificado por última vez en el servidor
  - T Tiempo actual
  - t intervalo de refresco (Importante un valor acertado para la consistencia y eficacia)

Sun Network File System

Otras optimizaciones

- Sun RPC corre sobre UDP por defecto pero puede usar TCP
- Sun RPC usa BSD Fast File System con bloque de 8 kbytes
- Lectura y escritura pueden ser de cualquier tamaño mientras antes se haya negociado entre cliente y servidor
- El intervalo de tiempo de refresco puede ser adaptado individualmente por fichero, para de esta manera reducir el numero de llamadas de actualización
- Los atributos de información de fichero incluido Tm, es devuelto en todas las consultas de ficheros.

Sun Network File System

Resumen

- Un excelente ejemplo de sistema distribuido sencillo, robusto y alto rendimiento.
- **Acceso:** Excelente. El API se esta intergrado en el infefaz de llamadas locales y remotas de UNIX.
- **Localización** No se garantiza pero normalmente funciona. El nombre de sistemas de ficheros es controlado por los clientes durante las operaciones de montado, pero la transparencia se puede asegurar mediante una configuración apropiada del sistema
- **Concurrencia** Limitada, pero adecuada para la mayoría de los propositos. Cuando ficheros de lectura-escritura son compartidos de manera concurrente entre cliente la consistencia no es perfecta.
- **Reclicación** Limitada a los ficheros de solo lectura, para los de escritura, el SUN Network Informaticon Service (NIS) corre sobre NFS y es usado para sistema esencial de réplica

Sun Network File System

Resumen

- **Tolerancia a fallos** Limitado pero eficaz. El servicio es suspendido si un servidor falla. La recuperación gracia al diseño **stateless**
- **Movilidad** Apenas conseguido. La reubicación entre servidores se puede realizar pero la actualización de las tablas de montado remoto en cada cliente deber hacerse de manera individual.
- **Rendimiento** Bueno. Los servidores multiprocesador permiten un alto rendimiento.
- **Escalabilidad**Buena. Se puede aumentar las prestaciones de un servidor mediante la adición de procesadores. Los grupos de ficheros pueden ser subdivididos y realojados en otros servidores. CUANDO NECESITEMOS CASOS EXTREMOS SERÁ MEJOR COMO AFS, QUE REDUCE EL TRÁFICO DEL PROTOCOLO HACIENDO CACHÉ DE LOS ARCHIVOS COMPLETOS.

Sección 4 | Otros

Google File System (GFS)

Desarrollo

Google File System (GFS)

- Características
- Arquitectura
- Funciones del servidor Master y Chunk
- Como se resuelve una consulta de un cliente
- GFS API
- Replicaciones
- Consistencia

Hadoop Distributed File System (HDFS)

Desarrollo

Hadoop Distributed File System (HDFS)

- Características
- Arquitectura
- Funciones de los servidores
- Como se resuelve una consulta de un cliente
- HDFS API
- Replicaciones
- Consistencia