Tema 2:

Sistema de archivos distribuidos NFS (Network File System)

NFS: Introducción

- Introducido por Sun Microsystems en 1985
- Desarrollado originalmente para UNIX.
- Se concibió como sistema abierto,
  - lo que le ha permitido ser adoptado por todas las familias UNIX y por otros sistemas operativos (VMS, Windows),
  - convirtiéndose en un estándar de facto en LANs.
- NFS ha evolucionado mucho, y la Versión 4 ya que incluye estado y la posibilidad de implementación en WAN.

NFS: Características generales

- Los servidores exportan directorios.
- Para hacer exportable un directorio se incluye el path en un determinado fichero de configuración.
- Los clientes montan los directorios exportados, y estos se ven en el cliente completamente integrados en el sistema de ficheros.
- El montaje se ejecuta en el booting del sistema operativo, o por demanda cuando se abre un fichero mediante un servicio adicional de NFS, el ***automounter***.
- Demonio ***nfsd*** atiende las operaciones sobre ficheros
- Demonio ***mountd*** atiende las operaciones de montado
- Servidores NFS son sin estado
  - evita el tener que tratar en el servidor los fallos de los clientes.
    - `Gracias a que la mayoría de las operaciones son idempotentes, la gestión de errores de comunicación en el cliente se simplifica.`

NFS: Características generales

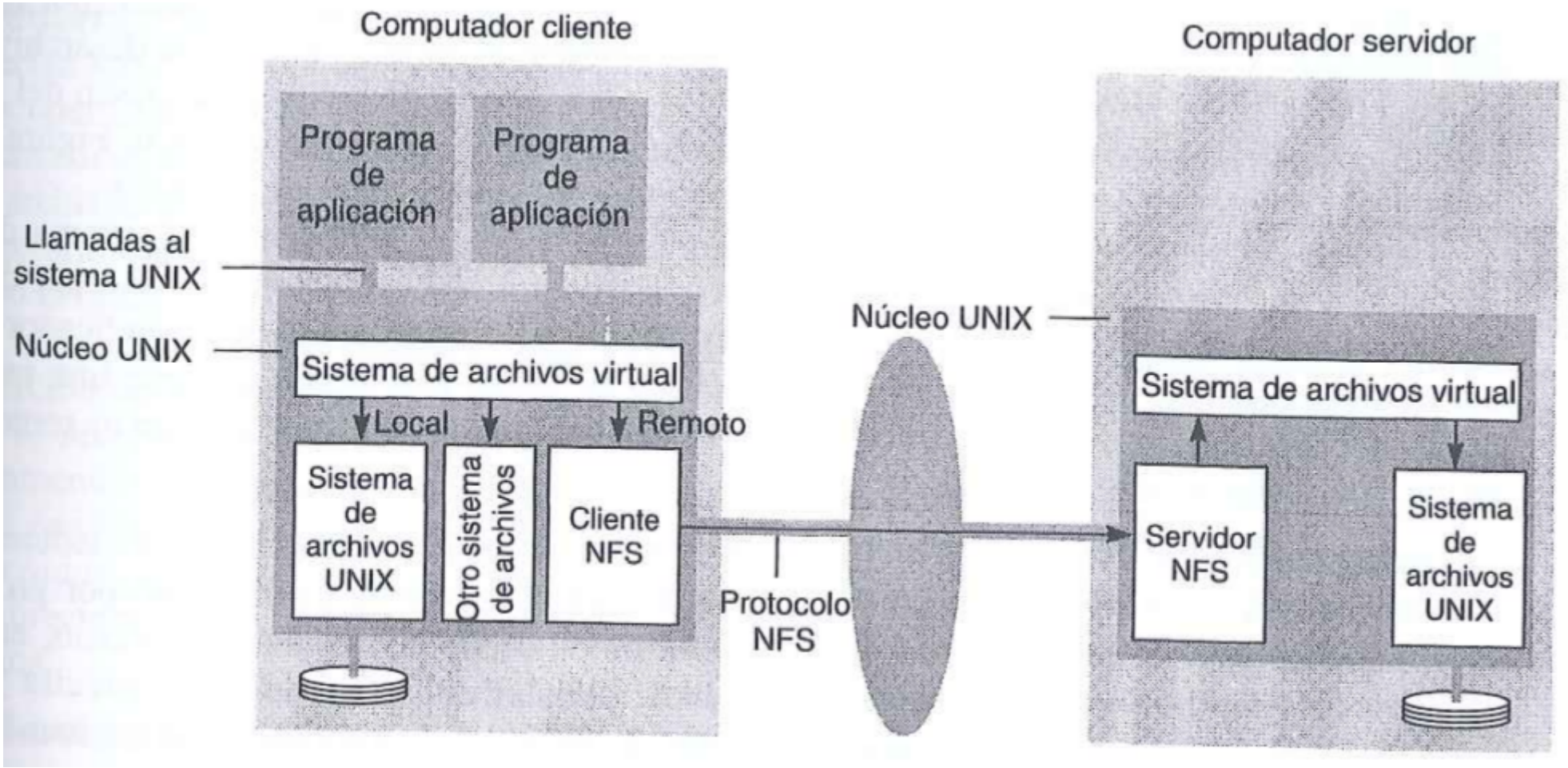
- La semántica de compartición intenta ser UNIX, aunque con alguna limitación, debido a
  - la gestión del caching y
  - a su condición de servidor sin estado.
- Ofrece el mismo modelo de protección de UNIX,
  - debido a la ausencia de estado en el servidor, los derechos de acceso se comprueban en cada operación de acceso al fichero en vez de sólo al abrir.
- Inicialmente NFS no adoptaba ningún mecanismo de autenticación.
  - La interfaz del cliente incluía en las RPCs el identificador de usuario UNIX, que se comprobaba en el servidor,
    - `lo que no impedía la posibilidad de suplantar la identidad de un usuario construyendo una RPC al margen de la ofrecida por la interfaz.`
- Actualmente suele combinarse con sistemas de autenticación como Kerberos.
  - Kerberos se basa en criptografía de clave simétrica y requiere un tercero de confianza.
    - `existen extensiones para utilizar criptografía de clave asimétrica.`
- NFS utiliza clásicamente el servicio NIS (Network Information Server) para centralizar la información sobre ubicación de los servidores.

NFS: Características generales

- El ***módulo servidor*** NFS reside en el núcleo que actúa de servidor
- El ***módulo cliente*** traduce las operaciones sobre ficheros remotos a operaciones del protocolo NFS que traslada al módulo servidor
- La comunicación se hace mediante RPC (Remote Procedure Call)
  - RPC se desarrollo para su uso con NFS
  - Puede usar UDP o TCP
- La interfaz es abierta
  - Cualquier proceso puede enviar solicitudes al servido NFS
  - Solo se ejecutarán la solicitudes con credenciales válidas

NFS: Sistema de archivos virtuales

- El sistema de archivos virtuales o Virtual File System (VFS) proporciona acceso trasparente
  - Los usuarios no distinguen entre ficheros locales y remotos



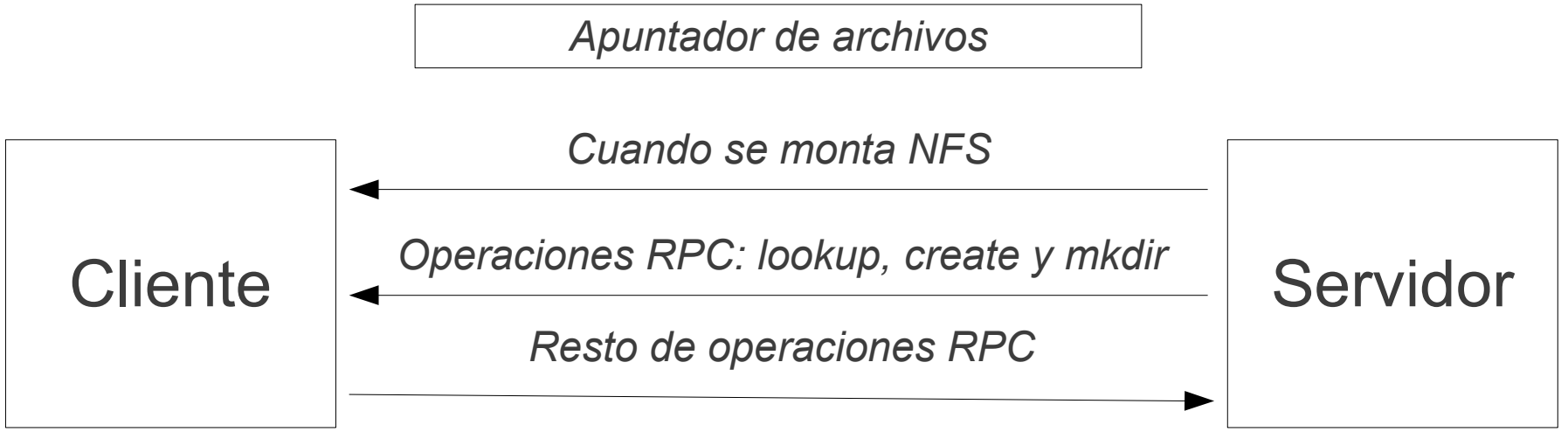
NFS: Sistema de archivos virtuales

- Los identificadores de ficheros utilizados en NFS se llaman ***apuntadores de ficheros (file handles).***
  - Para el cliente son opacos
  - Para el servidor contienen toda la información para distinguir un archivo individual

Apuntador de fichero = *identificador de* + *Número de i-node* + *Número de generación*
File handles *file system o volumen* (propio UNIX,) de i-nodos

  - *File system o volumen* corresponde al dispositivo de almacenamiento o partición y no al *sistema de ficheros* que es el componente software que proporciona el acceso a los ficheros
  - El identificador de volumen es un número único que se reserva para cada volumen cuando se crea y se almacena en UNIX en el superbloque del sistema de archivos
  - El *número de i-nodo* identifica a un fichero pero se reutiliza cuando se borra.
  - El *número de generación de un i-nodos* se incrementa cada vez que se reutiliza el *número de i-nodo*

NFS: Sistema de archivos virtuales



- El VFS mantiene
  - Una estructura VFS por cada sistema de archivos montado
    - Contiene un identificador para indicar si el archivo es local o remoto
  - Un *v-nodo por archivo abierto.*
    - *Si es local contiene un i-nodo*
    - *Si es remoto contiene un apuntador de fichero o file handles*

NFS: Integración del cliente

- El cliente NFS está integrado en el núcleo de UNIX
  - Los programas no necesitan ser modificados para usar NFS
  - Un único módulo cliente sirve todos los procesos con una caché compartida
  - La clave de encriptación para autenticar las ID de usuarios pasadas al servidor pueden retenerse en el núcleo
  - Comparte el mismo *buffer* de caché utilizado por el sistema de entrada/salida local

NFS: Control de acceso y autenticación

- El sistema UNIX es un sistema con estados
  - mantiene abierto los ficheros
  - identifica al usuario cuando los abre.
- El servidor NFS es sin estados
  - no mantiene los ficheros abiertos
  - identifica al usuario en cada operación
    - Ejemplo: los 16 bits del ID de usuario y grupo de UNIX
- No hay una llamada en el protocolo Sun RPC que haga la identificación sino que es parte del protocolo
- Seguridad
  - El cliente puede incluir el ID de cualquier usuario haciéndose pasar por él
  - Pare evitarlo suele combinarse con sistemas de autenticación como Kerberos.
    - Kerberos se basa en criptografía de clave simétrica y requiere un tercero de confianza.
      - existen extensiones para utilizar criptografía de clave asimétrica

NFS: Interfaz RPC del servidor NFS

lookup(aadir, nombre) → aa, atrib	Devuelve el apuntador del archivo y los atributos para el archivo *nombre* en el directorio *aadir*.
create(aadir, nombre, atrib) → aanuevo, atrib	Crea un nuevo archivo *nombre* en el directorio *aadir* con los atributos *atrib* y devuelve el nuevo apuntador de archivo y sus atributos.
remove(aadir, nombre) → estado	Elimina el archivo *nombre* del directorio *aadir*.
getattr(aa) → atrib	Devuelve los atributos del archivo *aa*. (Igual que la llamada UNIX *stat*.)
setattr(aa) → atrib	Establece los atributos (modo, ID usuario, ID grupo, tamaño, tiempo de acceso y tiempo de modificación de un archivo). Si se pone el tamaño a 0 se trunca el archivo.
read(aa, despl, conteo) → atrib, datos	Devuelve hasta *conteo* bytes de datos desde el archivo, comenzando en *despl*. También devuelve los atributos más recientes del archivo.
write(aa, despl, conteo, datos) → atrib	Escribe *conteo* bytes de datos sobre el archivo, comenzando en *despl*. También devuelve los atributos del archivo tras la operación de escritura.
rename(aadir, nombre, destaadir, destnombre) → estado	Cambia el nombre del archivo *nombre* del directorio *aadir* en *destnombre* del directorio *destaadir*.
link(nuevoaadir, nuevonombre, aadir, nombre) → estado	Crea una entrada *nuevonombre* en el directorio *nuevoaadir* que se refiere al archivo *nombre* en el directorio *aadir*.
symlink(nuevoaadir, nuevonombre, texto) → estado	Crea una entrada *nuevonombre* en el directorio *nuevoaadir*, con un *enlace simbólico* con el valor *texto*. El servidor no interpreta *texto* pero crea un archivo de enlace simbólico para alojarlo.
readlink(aa) → texto	Devuelve el *texto* asociado con el archivo de enlace simbólico identificado por *aa*.
mkdir(aadir, nombre, atrib) → nuevoaa, atrib	Crea un nuevo directorio *nombre* con los atributos *atrib* y devuelve el nuevo apuntador de archivo (*nuevoaa*) y sus atributos.
rmdir(aadir, nombre) → estado	Elimina el directorio vacío *nombre* del directorio padre *aadir*. Falla si el directorio no está vacío.
readdir(aadir, cookie, conteo) → entradas	Devuelve *conteo* bytes de entradas de directorio desde el directorio *aadir*. Cada entrada contiene un nombre de archivo, un apuntador a archivo, y un puntero opaco a la siguiente entrada del directorio, denominado *cookie* (galletita). *cookie* se emplea en las siguientes llamadas a *readdir* para comenzar la lectura desde la siguiente entrada. Si se pone el valor de *cookie* a 0, lee desde la primera entrada.
statfs(aa) → estadosf	Devuelve información sobre el sistema de archivos (tal como tamaño de bloque, número de bloques libres y demás) para el sistema de archivos que contiene el archivo *aa*.

NFS: Servicio de montado

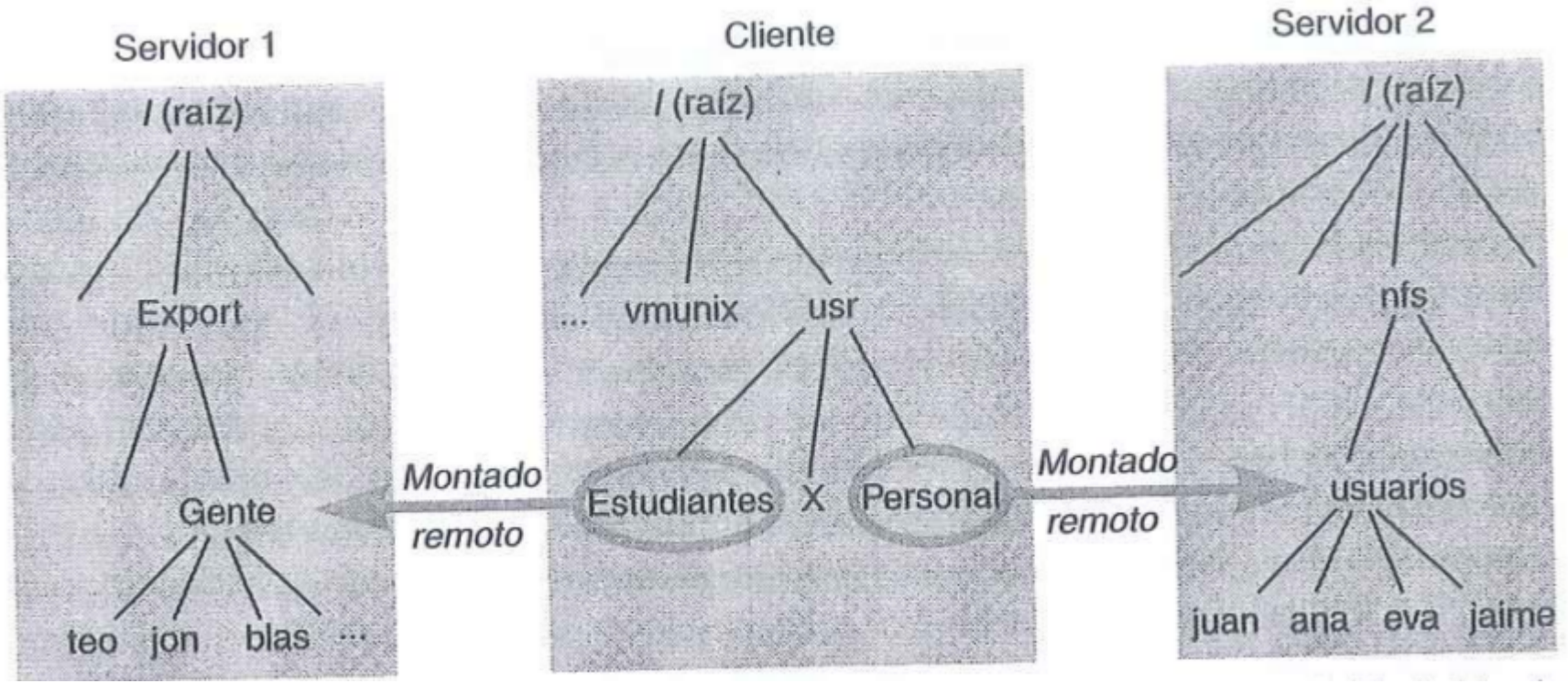
- El servicio de montado está separado y se ejecuta a nivel de usuario en cada computadora servidor
- `/etc/export` contiene los sistemas de fichero locales que se exportan y las máquinas autorizadas

```
/etc/export Servidor 1 (192.168.1.12)
/Export/Gente    192.168.1.0/24(rw,fsid=0,insecure,no_subtree_check,async)
```

```
/etc/export Servidor 2 (192.168.1.13)
/nfs/usuarios    192.168.1.10(rw,fsid=0,insecure,no_subtree_check,async)
```

- El montado en el cliente se se realiza sobre un fichero existente

```
Cliente (192.168.1.10)
$mount -t nfs -o vers=3 192.168.1.12:/Export/Gente /usr/Estudiantes
$mount -t nfs -o vers=4 192.168.1.13:/nfs/usuarios /usr/Personal
```



NFS: Servicio de montado

- Los volúmenes pueden montarse de dos formas
  - Montado rígido (hard-mounted)
    - Cuando un proceso accede a un archivo el proceso se suspende hasta que termina la operación.
      - Si falla el servidor los procesos se paran hasta que el servidor se recupera.
  - Montado flexible (soft-mounted)
    - Cuando falla el módulo cliente devuelve un fallo a los procesos y ellos gestionan el error, por ejemplo, haciendo otra petición.
      - Puede que el proceso no esté preparado para gestionar estos fallos
        - **Por eso se suele utilizar un montado rígido.**

NFS: Protocolo RPC de montado

```
program MOUNT_PROGRAM {
    version MOUNT_V3 {
        void MOUNTPROC3_NULL(void) = 0;
        mountres3 MOUNTPROC3_MNT(dirpath) = 1;
        mountlist MOUNTPROC3_DUMP(void) = 2;
        void MOUNTPROC3_UMNT(dirpath) = 3;
        void MOUNTPROC3_UMNTALL(void) = 4;
        exports MOUNTPROC3_EXPORT(void) = 5;
    } = 3;
} = 100005;
```

NFS: Protocolo RPC de NFS

```
program NFS_PROGRAM {
    version NFS_V3 {
        void NFSPROC3_NULL(void) = 0;
        GETATTR3res NFSPROC3_GETATTR(GETATTR3args) = 1;
        SETATTR3res NFSPROC3_SETATTR(SETATTR3args) = 2;
        LOOKUP3res NFSPROC3_LOOKUP(LOOKUP3args) = 3;
        ACCESS3res NFSPROC3_ACCESS(ACCESS3args) = 4;
        READLINK3res NFSPROC3_READLINK(READLINK3args) = 5;
        READ3res NFSPROC3_READ(READ3args) = 6;
        WRITE3res NFSPROC3_WRITE(WRITE3args) = 7;
        CREATE3res NFSPROC3_CREATE(CREATE3args) = 8;
        MKDIR3res NFSPROC3_MKDIR(MKDIR3args) = 9;
        SYMLINK3res NFSPROC3_SYMLINK(SYMLINK3args) = 10;
        MKNOD3res NFSPROC3_MKNOD(MKNOD3args) = 11;
        REMOVE3res NFSPROC3_REMOVE(REMOVE3args) = 12;
        RMDIR3res NFSPROC3_RMDIR(RMDIR3args) = 13;
        RENAME3res NFSPROC3_RENAME(RENAME3args) = 14;
        LINK3res NFSPROC3_LINK(LINK3args) = 15;
        READDIR3res NFSPROC3_READDIR(READDIR3args) = 16;
        READDIRPLUS3res FSPROC3_READDIRPLUS(READDIRPLUS3args) = 17;
        FSSTAT3res NFSPROC3_FSSTAT(FSSTAT3args) = 18;
        FSINFO3res NFSPROC3_FSINFO(FSINFO3args) = 19;
        PATHCONF3res NFSPROC3_PATHCONF(PATHCONF3args) = 20;
        COMMIT3res NFSPROC3_COMMIT(COMMIT3args) = 21;
    } = 3;
} = 100003;
```

NFS: Protocolo RPC de NFS

LOOKUP3res NFSPROC3_LOOKUP(LOOKUPc3args)

```
struct LOOKUP3args {
            diropargs3 what;
          };
struct diropargs3 {
            nfs_fh3 dir;
            filename3 name;
          };
union LOOKUP3res switch (nfsstat3 status) {
           case NFS3_OK:
                LOOKUP3resok resok;
           default:
                LOOKUP3resfail resfail;
          };
struct LOOKUP3resok {
           nfs_fh3 object;
           post_op_attr obj_attributes;
           post_op_attr dir_attributes;
          };
```

WRITE3res NFSPROC3_WRITE(WRITE3args)

```
struct WRITE3args {
          nfs_fh3 file;
          offset3 offset;
          count3 count;
          stable_how stable;
          opaque data<>;
        };
union WRITE3res switch (nfsstat3 status) {
          case NFS3_OK:
               WRITE3resok resok;
          default:
               WRITE3resfail resfail;
        };
struct WRITE3resok {
         wcc_data file_wcc;
         count3 count;
         stable_how committed;
         writeverf3 verf;
       };
```

READ3res NFSPROC3_READ(READ3args)

```
struct READ3args {
        nfs_fh3 file;
        offset3 offset;
        count3 count;
    };
union READ3res switch (nfsstat3 status) {
        case NFS3_OK:
             READ3resok resok;
        default:
             READ3resfail resfail;
    };
struct READ3resok {
        post_op_attr file_attributes;
        count3 count;
        bool eof;
        opaque data<>;
};
```

NFS: Traducción de un nombre de ruta

- Unix traduce una ruta de archivo a un i-node cuando se hace una llamada a *`open, create o stat.`*
- En NFS los nombres no pueden traducirse en el servidor ya que pueden cruzar un punto de montado en un cliente
- Los nombres de ruta se analizan y traducen en el cliente
  - Cuando una parte de un nombre se refiere a un directorio montado se hace un `lookup` al servidor remoto para que devuelva el ***apuntador de archivo***
  - Los resultados de sucesivos `lookup` se almacenan en la cache pudiendo hacer más eficiente este mecanismo aparentemente tan ineficiente

NFS: Automontador

- Se añadió a NFS para poder montar directorios remotos dinámicamente.
  - La implementación original se ejecuta como un proceso de usuario
  - Las versiones posteriores, llamado `autofs`, se implementa en el núcleo
    - `Solareis`
    - `Linux`
- Funcionamiento
  - Se mantiene una tabla de puntos de montaje con la referencia a uno o más servidores
  - Cuando el cliente solicita una volumen remoto hace una llamada lookup al demonio `automount` local
  - El automontador manda una solicitud de “prueba” a todos los servidores de la tabla
  - Monta el volumen del primer servidor en responder con un enlace simbólico
    - Si otro usuario lo solicita no hay que hacer otro montaje sino otro enlace simbólico
  - Cuando no hay referencias al enlace simbólico durante un tiempo este se desmonta.

NFS: Automontador

- Puede conseguirse una forma de replicación sencilla
  - Varios servidores contienen /usr/lib
  - El automontador monta el que primero responda que será el menos ocupado
  - Así se consigue un equilibrado de la carga en los servidores

NFS: Cache en el servidor

- Unix retiene en una *cache buffer* de memoria principal
  - Páginas de archivo
  - Directorios
  - Y atributos de archivos
- Todos los procesos de lectura/escrituran pasan por la cache
  - Lectura anticipada (read-ahead)
  - Escritura retardada (delayed-write)
    - Se escribe en disco la modificación cuando se solicita leer la página de información modificada
    - Para salvaguardar la consistencia, en UNIX cada 30 segundos se guardan la paginas actualizadas independientemente si se leen o no
- Los servidores NFS necesitan medidas extras para las escrituras

NFS: Cache en el servidor

- La operación `write` dos opciones
  - Escritura a través o write-throught de la cache.
    - `Los datos escritos se modifican en la cache y se escriben en disco antes de que envié la confirmación al cliente`
  - Los datos sólo se almacenan en la cache hasta que se hace un operación de *consumación* o `commit`
    - `Los clientes NFS estándar utilizan este modo de consumación.`

NFS: Cache en el cliente

- El módulo cliente emplea la cache para los resultados de las operaciones `read, write, getattr, lookup` y `readdir`
- Las escrituras de un cliente no producen la modificación de las cache de los otros clientes
- Los clientes sondean al servidor para comprobar la correspondencia con la cache del servidor
- Para validar los bloques en la cache cada elemento de datos o metadato de la cache se etiqueta con dos marcas
  - Tc → tiempo última validación de la caché
  - Tm → tiempo última modificación en la caché del servidor
- Una entrada en la caché es válida en el tiempo T cuanto

  $(T - Tc < t)$ v $(Tm_{cliente} = Tm_{servidor})$

  - Siendo t un intervalo de refresco → compromiso consistencia/eficiencia
    - `t<<< datos consistentes con una alta sobrecarga en la comunicación con el servidor para comprobar` $Tm_{servidor}$
  - En clientes Sun, t se asigna de forma adaptativa dependiendo de la frecuencia de escritura en un rango
    - `Ficheros → [3,30]segundos`
    - `Directorios → [30,60] seg. (riesgo escritura concurrente bajo)`

NFS: Cache en el cliente

- Un cliente no sabe si un archivo está siendo compartido o no
  - Obligatoriamente tiene que utilizar el procedimiento de validación
- Procedimiento de validación

```
Si T – Tc < t entonces
   la cache es válida
Sino
     llamar getattr al servidor para obtener Tm_servidor
     Si Tm_cliente = Tm_servidor entonces
        la caché es válida y se actualiza Tc
     Sino
         se invalida la cache
         se solicita los datos al servidor
     FinSi
FinSi
```

NFS: Cache en el cliente

- Medidas para reducir el trafico de llamadas `getattr` al servidor
  - El $Tm_{servidor}$ recibido se aplica para todas las entradas en la cache del fichero
  - El $Tm_{servidor}$ se manda en cada operación y se aplica para todas las entradas de la cache del fichero
  - El algoritmo adaptativo para fijar el intervalo de refresco
- Fuentes de retardo
  - Retardo de escritura antes de que los datos dejen la caché del cliente
  - Los 3 segundos del intervalo de refresco t
- Las aplicaciones UNIX no dependen de forma crítica de la sincronización de las actualizaciones

NFS: Cache en el cliente

- Escrituras
  - Cuando se modifica una página en la cache
    - Se marca como sucia y
    - Se planifica asíncronamente su volcado al servidor
      - Cuando se cierra el fichero o
      - Cuando VFS del cliente lanza un `sync` al servidor
- Lecturas adelantadas y escrituras retardada
  - Utilizando bio-demonios en cada cliente
    - Bio se refiere a Block input-output o entrada-salida de bloques
  - Funciones de los biodemonios
    - Después de cada lectura solicita la transferencia del siguiente bloque desde el servidor a la caché del cliente
    - Envía un bloque al servidor cuando se ha llenado
    - Los bloques de directorios se envían inmediatamente después de su modificación

NFS: Otras optimizaciones

- Los paquetes UDP utilizados en RPC son de 9 byte y UNIX utiliza 8 byte como tamaño de bloque disco
  - Una llamada RPC puede enviar un bloque de disco completo

- NFSv3 no hay tamaño máximo de los bloques de archivos que manejan las operaciones `write` y `read`
  - Cliente y servidor pueden negociar tamaños mayores de 8 byte

- Todas las operaciones tienen implícita una operación getattr para pedir el $Tm_{servidor}$
  - De esta forma los archivos más activos pueden consultar su estado cada 3 segundos sin mucha carga para el sistema

NFS: NFS seguro con Kerberos

- En NFS la identificación del usuario se incluye en cada solicitud como un identificador numérico no encriptado o encriptado en versiones posteriores
  - Supone confiar en todos los computadores clientes y su software
- Al no tener estados NFS, Kerberos necesitaría hacer una comprobación en cada operación
  - Demasiado costoso en tiempo
- Se opta por proporcionar al servidor de montado NFS todos los datos de autenticación de Kerberos para el usuario cuando se montan sus volúmenes raiz y home
  - Los datos de la autenticación incluyendo ID usuario y MAX del cliente se mantienen en el servidor junto a la información de montado de cada volumen almacenada en el servidor

NFS: Prestaciones

- En general, en una red de área local las prestaciones no se ven muy degradadas si el diseño es bueno
- Existe programas de pruebas conocidos como LADDIS cuyos resultados se pueden consultar en www.spec.org
  - Ejemplo 1.
    - 5.011 operaciones en el servidor por segundo,
    - 3,71 milisegundo de latencia promedio
    - 8ms de latencia máxima
    - Sistema: 1 x 450 Mhz Pentium II CPU, 34 discos en un controlador Ethernet de 1 Gbps y sistema operativo en tiempo real dedicado.
  - Ejemplo2.
    - 29.083 operaciones en el servidor por segundo
    - 4,25/7,8 latencia promedio/máxima
    - Sistema: 24 x 450Mhz IBM RS64-III CPU, 289 discos en cuatro controladores, 5 redes de 1Gbps, SO AIX UNIX
  - Más:

    http://www.spec.org/sfs2008/results/sfs2008nfs.html

NFS: Limitaciones

- Se pueden aumentar las prestaciones de un único servidor añadiendo procesadores, discos y controladores

- Otra opción, o cuando se alcanza el límite de la anterior, es la **replicación**, es decir, utilizar varios servidores NFS
  - NFS sólo permite replicar archivos de sólo lectura
  - NFS no soporta la replicación de archivos actualizables

- Si necesitamos replicar ficheros actualizables deberemos recurrir a otros sistemas de ficheros distribuidos.