Network File Sysytem

Redes y Servicios de Comunicaciones

Andrés Marín López

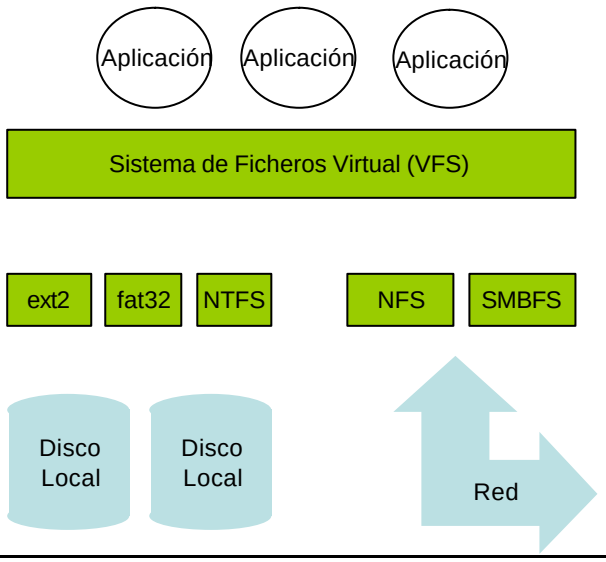
INDICE

- Introducción
- XDR
- RPC
- Modelo de sistema de ficheros de NFS
- Procedimientos NFS
- MOUNT
- Lock Manager y Status Manager

Introduccion

- FTP: transferencia de ficheros remotos
- NFS: acceso a ficheros remotos
  - solo transfiere lo que se necesita
  - ficheros remotos parecen locales
  - ficheros consistentes
  - clientes sin disco
  - cerrojos
- Se monta sobre RPCs y utiliza protocolos auxiliares: mount, nlm, nsm.

Sistemas de ficheros



Retos

- Diversidad de arquitecturas de máquinas, sistemas operativos
  - XDR representación común
- Modelo cliente/servidor, diversidad de parámetros, funciones y resultados
  - RPC abstracción común transporte
- Modelo de ficheros y operaciones
  - NFS
- Protocolo inicial común:
  - MOUNT
- Mecanismos de cerrojos
  - NLM, NSM

XDR

XDR

- External Data Representation (RFC 1832)
- Independizar representacion de datos (HW, SO, ...)
  - Tipos primitivos: Integer, Boolean, String, Opaque data, etc.
  - Tipos estructurados: Array, Union, List
  - Sintaxis para describir secuencias y tipos de datos codificados en XDR

Tipos de datos

int: enteros 32 bits comp. a 2, big-endian
unsigned int.
enum: subconjunto del tipo entero.
bool: caso particular de enumeración.
hyper int y *unsigned hyper int*: de 64 bits.
float: 32 bits, estándar IEEE coma flotante.
double: 64 bits, IEEE para doble precisión.
opaque: secuencias de bytes que se transmiten sin interpretación.
strings: cadena de caracteres en ASCII.
arrays de tamaño fijo y variable.
Otros: *struct, union*, v*oid*, *cont*, *typedef*

XDR: ejemplo

```
const MAXL= 32;
enum clasef{
   TEXT= 0;
   DATA= 1;
   EXEC= 2;
}
union tipof switch(clasef c){
case TEXT:   void;
case DATA:  string creator<MAXL>;
case EXEC:string interprete<MAXL>;
}
```

XDR: ejemplo

```
struct fichinfo{
   string nombre<MAXL>;
   tipof type;
   string owner<MAXL>;
   opaque data<MAXL>;
}
```

RPC

RPC

- Remote Procedure Call (RFC 1831, 1832)
- Independencia de transporte (sin ocultarlo)
- Semantica (at-least-once, at-most-once, exactly-once)
  => ***Idempotencia: f(f(...(f(x))...))=f(x)***
- Sirve para un gran numero de programas diferenciados mediante un mecanismo de versiones y números

RPC: Mensaje de llamada

- XID: identificador único (llamada y respuesta)
- CALL=0
- RPC version=2
- Program Number
- Version Number
- Procedure Number
- Credential
- Verifier
- Parameters

RPC: Mensaje de llamada

- XID se debe reutilizar para retransmisiones de llamadas
- El servidor también lo puede almacenar en cache para evitar repetir el procedimiento (idempotencia)
- La credencial autentica el cliente frente al servidor; la verificación valida la credencial
- Tipos de autenticación: AUTH_NONE, AUTH_SYS, AUTH_SHORT, AUTH_DH, AUTH_KERB4, AUTH_RSA, RPCSEC_GSS

RPC: Mensaje de Respuesta

- XID: identifica transacción
- REPLY=1
- REPLY Status=(1-msg denegado, 0-aceptado)
- Verifier
- ACCEPTED StatusParameters=

  0-éxito, 1-prog no disponible, 2-numero no reconocido, 3- procedimiento no disponible, 4- error argumentos, 5- error servidor

RPC: Mensaje de Respuesta

- Un mensaje puede ser denegado:
  - por utilizar una version no valida de RPC
  - o bien por credencial/verificador inválido (fallo de autorización)
- Despues del ACCEPTED Status aparecen los datos de la respuesta en XDR (si los hay).

Portmap

- Protocolo privilegiado de RPC
- Sirve para registrar al resto de los programas que van a utilizar RPCs
- Soporta procedimientos: registro y anulacion de programas, obtencion del puerto de un programa, volcado de tablas, invocacion indirecta

Protocolo PORTMAP

- PMAPPROC_NULL
- PMAPPROC_UN/SET (des/registro número programa, puerto y protocolo)
- PMAPPROC_GETPORT
- PMAPPROC_DUMP (volcado tablas)
- PMAPPROC_CALLIT (llamada a procedimiento de otro programa a través del portmapper)

NFS

NFS: objetos

- `fsid`: identificador único del sistema de ficheros remoto
- `fileid`: identificador único del fichero o directorio remoto
- ficheros: secuencias de bytes no interpretadas
- directorios: listas de nombres de ficheros
- enlaces simbolicos: referencias que interpreta el cliente como ficheros o directorios
- ficheros especiales

Manejadores de ficheros

- Referencia a un fichero o directorio independiente del nombre (32B v2, 64 v3)
- El cliente no lo interpreta (opacidad)
- Persistentes: válidos durante toda la vida del fichero
- Independientes de la ubicación: permite operaciones de renombrado y desligado (unlink)

Protocolo NFS

- Sin estado: el servidor no necesita mantener el estado del cliente (las implementaciones pueden hacerlo)
- Recuperación más sencilla de errores
- Operaciones idempotentes: retransmisión de peticiones con el mismo resultado (no todas los son: REMOVE)
- Alta disponibilidad (duplicidad de datos y servidores)

Tipos de datos de NFS

```
enum stat{
  NFS_OK=0,              NFSERR_FBIG=27,
  NFSERR_PERM=1,         NFSERR_NOSPC=28,
  NFSERR_NOENT=2,        NFSERR_ROFS=30,
  NFSERR_IO=5,           NFSERR_NAMETOOLONG=63,
  NFSERR_NXIO=6,         NFSERR_NOTEMPTY=66,
  NFSERR_ACCESS=13,      NFSERR_DQUOT=69,
  NFSERR_EXIST=17,       NFSERR_STALE=70,
  NFSERR_NODEV=19,       NFSERR_WFLUSH=99
  NFSERR_NOTDIR=20,    }
  NFSERR_ISDIR=21,
```

Tipos de datos de NFS

```
const MAXDATA=8192;/*max read/write*/
const MAXPATHLEN=1024;
const MAXNAMELEN=255;
const COOKIESIZE=4;/*READIR cookie size*/
const FHSIZE=32;
typedef opaque fhandle[FHSIZE];
struct timeval{
  uint32 seconds;
  uint32 useconds;
}
```

Tipos de datos de NFS

```
struct attr{
  ftype type;      /*tipo de fichero*/
  uint32 mode;     /*permisos*/
  uint32 nlink;    /*num de hard links*/
  uint32 uid;      /*id usuario*/
  uint32 gid;      /*id grupo*/
  uint32 size;     /*bytes fichero*/
  uint32 blocksize;/*bytes de bloque*/
  uint32 rdev;     /*num dispositivo*/
  uint32 blocks;   /*num bloques*/
  uint32 fsid;     /*id sistema ficheros*/
  uint32 fileid;   /*id fichero en FS*/
  timeval atime;   /*tiempo de acceso*/
  timeval mtime;   /*ultima modificacion*/
  timeval ctime;   /*modif. atributos*/
}
```

Tipos de datos de NFS

```
enum ftype{
  NFNON=0,/*non file*/
  NFREG=1,/*regular file*/
  NFDIR=2,/*directory*/
  NFBLK=3,/*bock device*/
  NFCHR=4,/*char device*/
  NFLNK=5,/*symbolic link*/
  NFSOCK=6,/*socket*/
  NFFIFO=7/*named pipe*/
}
```

Procedimientos de NFS

- **NULL (pruebas y medidas de tiempos)**
- **GETATTR/SETATTR (toma/fija atributos de fichero)**
- **LOOKUP (busqueda del nombre: secuencial)**
- **LINK/SYMLINK/READLINK (crea un enlace/simbolico/lee de un enlace)**
- **CREATE/READ/WRITE/REMOVE/RENAME (crea/lee/escribe/borra/renombra un fichero)**
- **MKDIR/RMDIR/READDIR (crea/borra/lee un directorio)**
- **STATFS (toma los atributos de un sistema de ficheros)**

MOUNT

NFS Mount

- Procedimiento inicial para obtener el manejador inicial del sistema de ficheros
- Fuera del kernel, distintos protocolos
- Opciones de montaje:
  - ro, rw, etc.
  - Insecure
  - no_root_squash
  - dominios, etc.

Protocolo MOUNT

- **NULL (pruebas y medidas de tiempos)**
- **MNT (añade entrada mount)**
- **DUMP (vuelca entradas de mount)**
- **UMNT (quita entrada mount)**
- **UMNTALL (quita todas las entradas)**
- **EXPORT (devuelve lista exportaciones)**
- **EXPORTALL (idem)**
- **PATHCONF (informacion del sistema de ficheros para POSIX pathconf)**

Tabla MOUNT

- Tabla mantenida por el servidor de clientes y sistemas de ficheros montados
- Permite montar subdirectorios de los directorios exportados

```
mount \
   server:/usr/home/amarin /home/amarin
```

Limitaciones

- No hay soporte a manejadores de ficheros, si un sistema de ficheros no da manejadores de ficheros, no puede ser exportado.

Ej: /proc

- No permite reexportación en general (los manejadores podrían confundirse)
- No hay exportaciones jerárquicas

Cerrojos

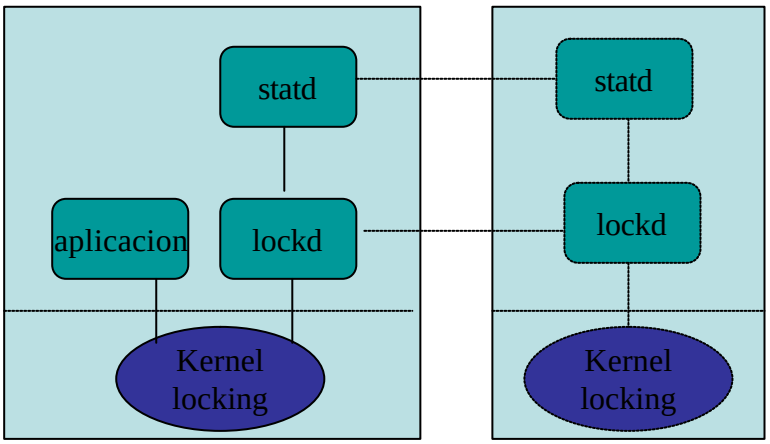
Lock Manager y Status Manager

- Mantenimiento de cerrojos sobre ficheros
- Problemas
  - pérdida del estado del servidor (el cliente puede pensar que sigue manteniendo sus cerrojos que se han perdido)
  - pérdida del estado del cliente (el servidor puede mantener cerrojos que un cliente ha perdido y bloquear al cliente y al resto)

Tipos de cerrojos

- Mandatory/advisory: impiden o no el acceso de otros clientes a los datos (DOS: mandatory)
- exclusivos/no exclusivos (shared): impiden o no que se añadan cerrojos en zonas superpuestas. Tambien se conocen como write/read locks.

Cerrojos: Arquitectura



Lockd y statd

- lockd: implementacion del protocolo Network Lock Manager
- statd: implementacion del protocolo Status Monitor
- la aplicación solicita al kernel un cerrojo, el cual se la pide a lockd que se la pasa al lockd del servidor quien se la solicita a su kernel.

Protocolo NLM

NULL (pruebas)

TEST (probar si tenemos un cerrojo disponible)

LOCK (solicitar un cerrojo, puede obtener el cerrojo o ser bloqueada hasta que se pueda devolver)

CANCEL (anular una peticion bloqueada de un cerrojo)

UNLOCK (solicita devolver un cerrojo)

GRANTED (servidor devuelve un cerrojo)

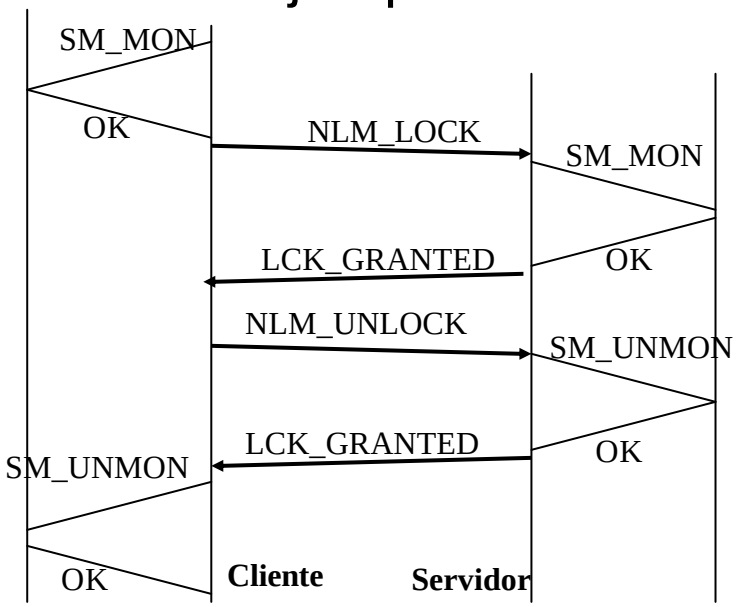
SHARE (solicita crear una reserva compartida)

UNSHARE

NM_LOCK (solicita cerrojo no monitorizado)

NLM_FREE_ALL (solicita devolver todos sus cerrojos)

Protocolo NSM

- **NULL (pruebas)**
- **STAT (solicita saber si una maquina esta siendo monitorizada)**
- **MON (solicita monitorizar una maquina)**
- **UNMON**
- **UNMON_ALL**
- **SIMU_CRASH (si el lockd del cliente se cae informa de que ha perdido toda la informacion de status)**
- **NOTIFY (notifica a todas las maquinas que estaba monitorizando de una caida)**

Ejemplo