Internet Systems Programming

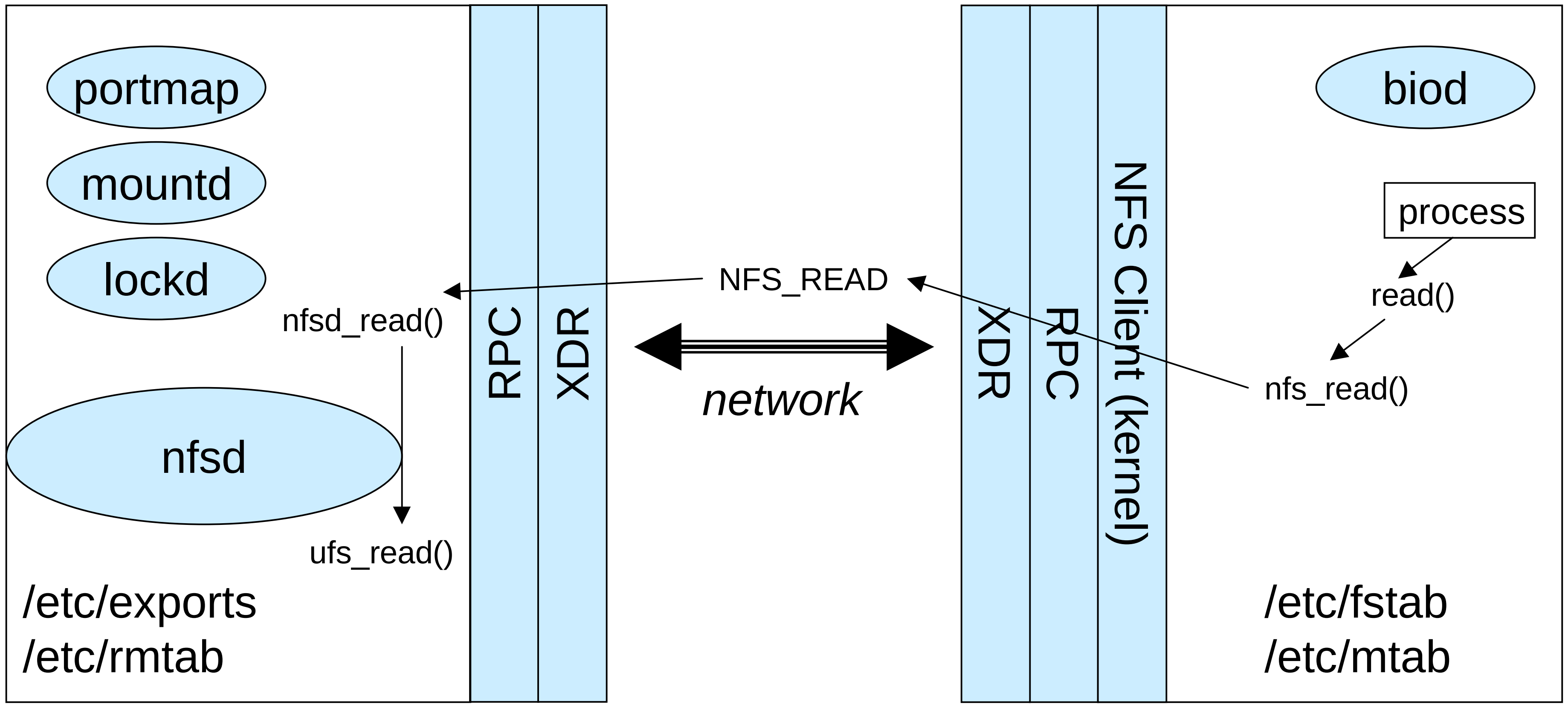
NFS: Protocols, Programming, and Implementation

Erez Zadok

`ezk@cs.columbia.edu`

October 25, 1999

The BIG Picture



NFS Server **NFS Client**

NFS Overview

- using RPC: Remote Procedure Calls
  - which use XDR: eXternal Data Representation
- stateless server
  - crash recovery
- client side caching (data and attributes)
  - request retransmission
- file handles: 32 bytes opaque to client
  - server encodes: fsid, inum, igen, possibly more

XDR: eXternal Data Representation

- de/serializes data into network-order bytes

```
bool_t xdr_long(XDR *xdrs, long *lp);
```

- repeated calls encode/decode more "XDR" bytes

```
struct foo {
   int i;
   char *buf;
};
bool_t xdr_foo(XDR *xdrs, struct foo *foop) {
   if (!xdr_int(xdrs, &foop->i))
       return FALSE;
   if (!xdr_wrapstring(xdrs, &foop->buf))
       return FALSE;
   return TRUE;
}
```

RPC: Remote Procedure Call

- server does:

```
registerrpc(prognum,versum,procnum,s_inproc,in,s_outproc,out);
svc_run()
```

- client issues:

```
callrpc(char *host, rpcprog_t prognum, rpcvers_t versnum,
    rpcproc_t procnum, xdrproc_t inproc, char *in,
    xdrproc_t outproc, char *out);
```

- which contacts server's portmapper, then RPC server w/ *procnum*.

when client request comes

  - find procnum
  - call s_inproc to decode client args
  - call s_outproc to encode output to client
  - return => client returns (or times out)

- rpcgen produces headers and .c stubs from .x files

Additional NFS Components

- on server:
  - mountd:
    - listen for mount requests
    - authenticate requests
    - return root fhandles
- on client:
  - biod: dirty page clustering, simulate async writes
- on both:
  - lockd: coordinates local/remote record locks
    - flock() uses lockd; lockf() only local locks; fcntl() can use both
  - statd: synchronizes lock information
    - client reboot: tell server to release locks
    - server reboot: tell all clients to reclaim locks
  - portmapper: the mother of all RPC servers

Example: mounting a remote server

- get fhandle (via MOUNTPROC_MNT rpc to mountd)
- fill in struct nfs_args
  - `struct nfs_args na`
- call mount(2) syscall
  - `mount(“/mnt”, flags, “nfs”, &na, sizeof(na))`

Contents of struct nfs_args

```
NA->addr {sockaddr_in} (len=16) =
   "02000801803b14640000000000000000"
NA->addr.sin_family = "2"
NA->addr.sin_port = "264"
NA->addr.sin_addr = "803b1464"
NA->hostname = "opus"
NA->namlen = 255
NA->filehandle =
   "008000f400000002000a000000000026e
   065b6c000a000000000026e065b6c"
```

```
NA->version = 3
NA->flags = 0x0
NA->rsize = 4096
NA->wsize = 4096
NA->bsize = 0
NA->timeo = 7
NA->retrans = 3
NA->acregmin = 3
NA->acregmax = 60
NA->acdirmin = 30
NA->acdirmax = 60
```

NFS V.2

- Built on top of UDP
- 17 calls

NFS_NULL	**0**	**NFS_CREATE**	**9**
NFS_GETATTR	**1**	**NFS_REMOVE**	**10**
NFS_SETATTR	**2**	**NFS_RENAME**	**11**
NFS_ROOT	3	**NFS_LINK**	**12**
NFS_LOOKUP	**4**	**NFS_SYMLINK**	**13**
NFS_READLINK	**5**	**NFS_MKDIR**	**14**
NFS_READ	**6**	**NFS_RMDIR**	**15**
NFS_WRITECACHE	7	**NFS_READDIR**	**16**
NFS_WRITE	**8**	**NFS_STATFS**	**17**

(why no lseek?)

Ex: NFS_READ Call

```
struct readargs {
       fhandle file;
       unsigned offset;
       unsigned count;
       unsigned totalcount;
};

union readres switch (stat status) {
       case NFS_OK:
            fattr attributes;
            nfsdata data;
       default:
            void;
};
```

NFS V.3

- TCP and UDP
- 64 byte file handles
- files > 2GB
- ACLs supported
- Kerberos authentication type
- All ops return old/new attributes
  - saves on most popular call, getattr (update client caches faster)

NFS V.3 Protocol

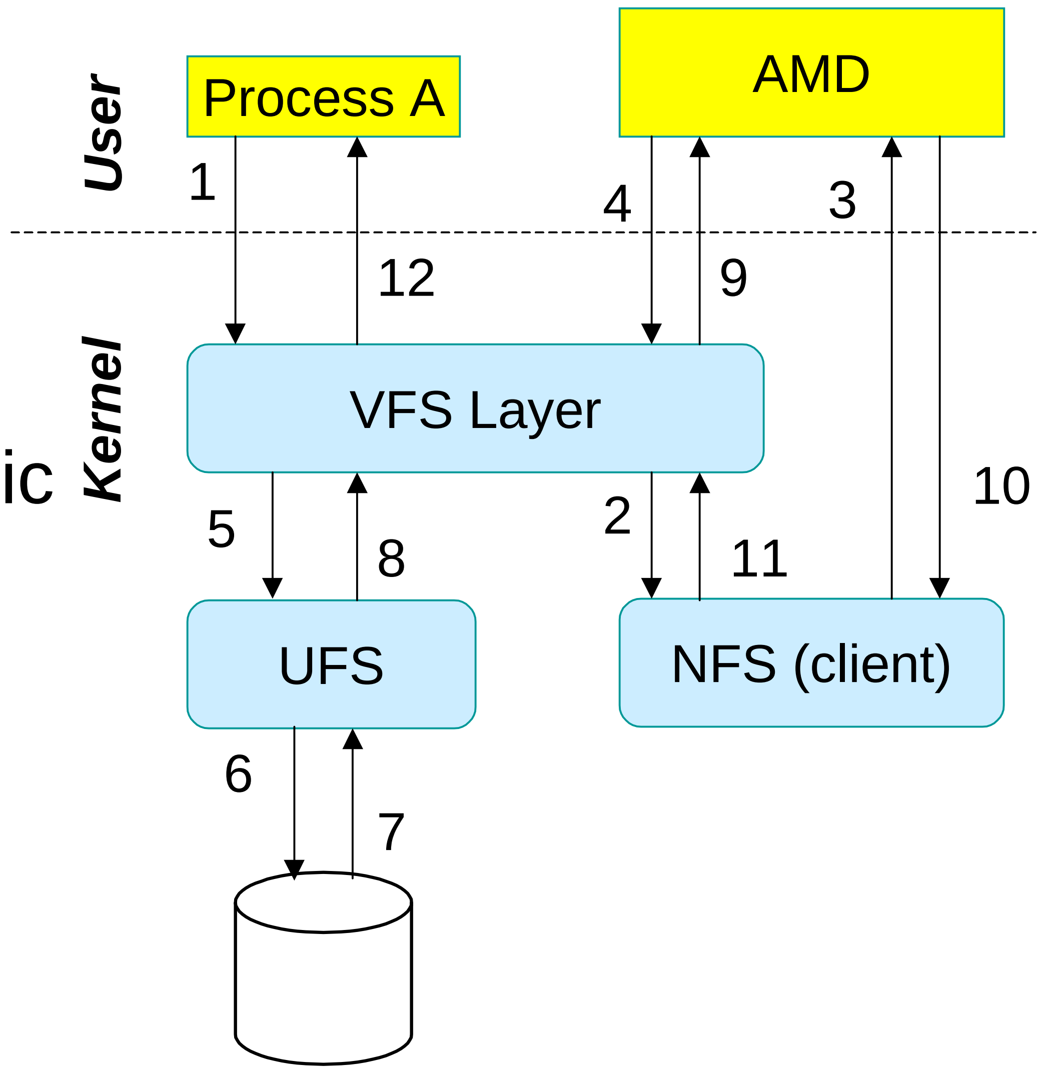
- Removed: ROOT and WRITECACHE
- Added:
  - **`READDIRPLUS:    17`**
    - also returns file handles
    - saves on NFS_LOOKUPs
  - **`FSSTAT:         18`**
  - **`FSINFO:         19`**
  - **`PATHCONF:       20`**
  - **`COMMIT:         21`**
    - **`Saves cached data to disk`**

NFS V.4

- IETF design, not Sun
- Integrated file locking and mount protocol
- Stronger security w/ negotiation
  - Public file handles
  - Works with firewalls & proxies
- Compound operations
- Internationalization
- Better suited for Internet (i.e., WAN)
- Migration and replication
- Extensible protocol

User Level NFS-Based File Servers

- Context switches
- extra communication
- Amd dead/hung?
- CFS: cryptographic file server



Resources

- RFC 1094/1813
  - Usenix papers [Sandberg 84] and [Pawlowski 94]
- NFS V.2/3/4 specs and drafts
  - ftp://ftp.cs.columbia.edu/archives/doc/rfc
  - ftp://ftp.cs.columbia.edu/archives/doc/internet-drafts
- sources to CFS
  - http://www.cryptography.org/
- Amd
  - http://www.cs.columbia.edu/~ezk/am-utils/
- Email: ezk@cs.columbia.edu