NFSv4 Open Source Project Update

Andy Adamson

CITI, University of Michigan

Ann Arbor

A little bit of history

- NFSv4 Open Source Reference Implementation Project
- Sponsored by Sun Microsystems
- IETF reference implementation
- 212 page spec
- Linux and OpenBSD

Topics

- Brief overview of implemented features
  - Changes to Linux kernel
  - POSIX vs NFSv4
  - State Management
- Administration

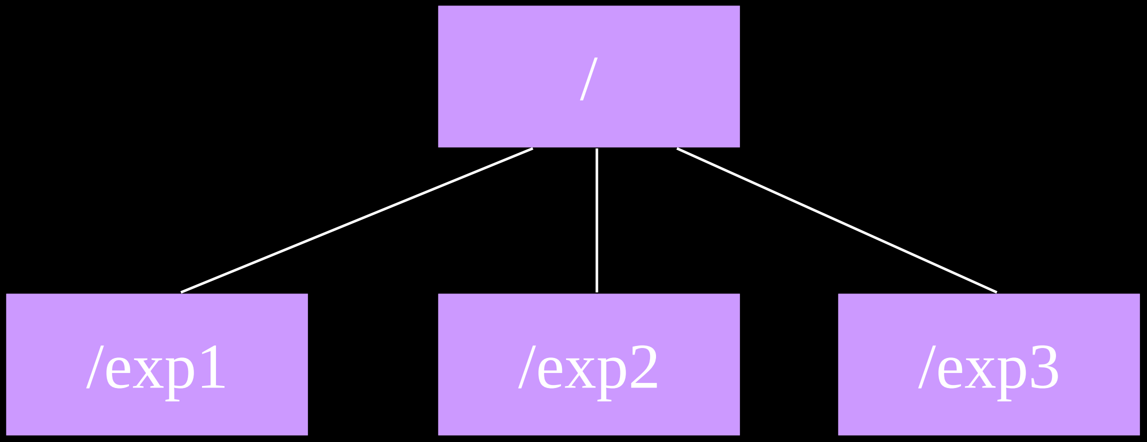
Implemented Features

- Pseudo File system
- Compound RPC
- Byte-Range Locking
- Access Control Lists
- RPCSEC_GSS
- Delegation

Pseudo Filesystem

- NFSv4 RFC does not specify the relationship between the names in the pseudo fs and the names of the actual exported directories on the server
- In our implementations, the namespaces are completely independent

Pseudo Filesystem



The pseudofs is just a device to allow clients to browse the exports without need for an auxiliary protocol

Compound RPC

- Designed to reduce traffic
- Complex calling interface, complex to parse
- Partial results used
- RPC/XDR layering
  - RPC layer does not interpret compound operations
  - Additional replay cache for lock mutating ops
  - Have to decode to decide which replay cache to use
- Variable length: malloc buffer for args and recv

Mount Compound RPC

PUTROOTFH
LOOKUP
GETATTR
GETFH

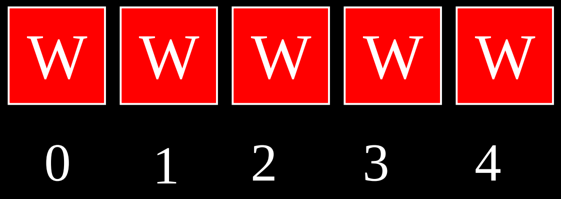
Start with the pseudofs root, lookup mount point Path name, and return attributes and file handle.

Compound RPC

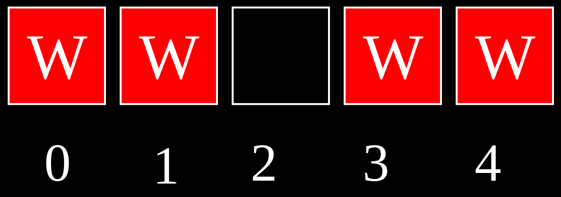
- Latest Linux implementation uses hand coded XDR encode and decode compound RPC operations with buffer overflow checks and short call stack.
- Decode handlers provide ideal place to process common errors
  - Wrong RPC security flavor
  - Lease expiration

Locking: POSIX vs. Windows

POSIX locking has the curious property that locks can be split

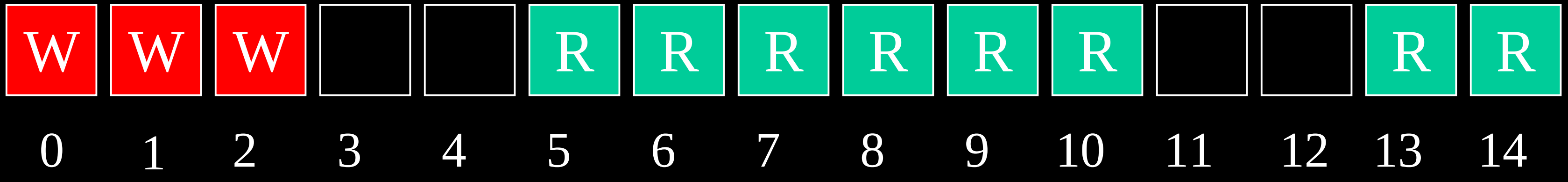


Fcntl(F_SETLK, F_UNLCK, 2, 1)

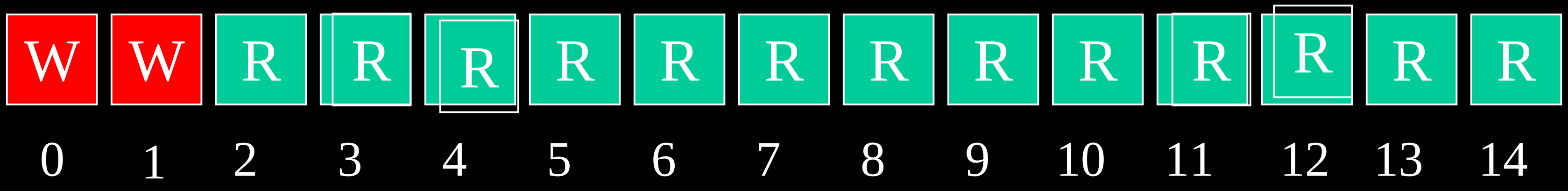


Locking: POSIX vs. Windows

or coalesced



fcntl(F_SETLK, F_RDLCK, 2, 13)



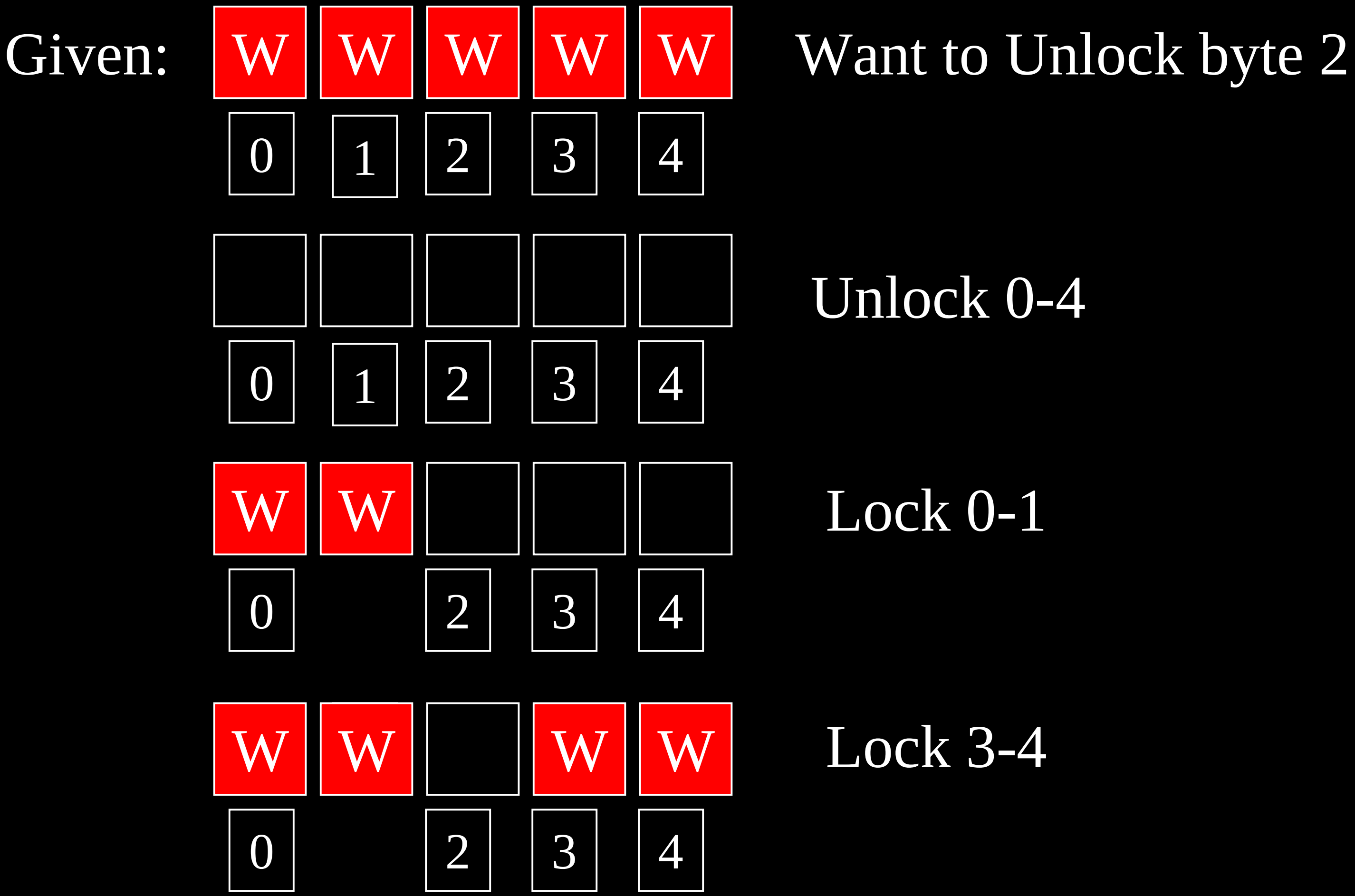
Locking: POSIX vs. Windows

- In fact, any byte range is legal in a POSIX locking request!
- For the sake of discussion, call any locking request which splits or coalesces a byte range "exotic".

Locking: POSIX vs Windows

- Problem: Windows servers cannot support exotic locking requests!
- According to NFSv4 RFC, Servers should support exotic locking requests if possible, but are not required to.
- clients wanting to emit an exotic requests is responsible for simulating it by a sequence of non-exotic requests.

The "Lost Lock" Race Condition



Linux Implementation of Locking

Two code paths, as mount option

- RFC compliant, simulate exotic requests with a sequence of non-exotic ones
- Send exotic requests unmodified to the server, this is the default behavior

Against a non-Windows server use 2

Against a Windows server ????

Access Control Lists

- We're implementing on top of Andreas Gruebacher's POSIX ACL patch (http://acl.bestbits.at).
- ACL code within CONFIG_POSIX_ACL
- NFSv4 defines ACL's which are much richer and more highly granulated than POSIX ACL's.

Access Control Lists

- Implemented a subset of the NFSv4 ACL spec which is functionally equivalent to POSIX ACL's
- In a purely Linux environment, can use POSIX ACL's over NFSv4
- In an environment containing other NFSv4 implementations, ACL's may not work

Access Control Lists

- Complete implementation of NFSv4 ACL spec deferred to a future project
- Outstanding issue with NFSv4 ACL's: support for POSIX ACL_MASK.

RPCSEC_GSS

- RFC 2203 defines the addition of the GSSAPI to the ONC RPC.
- An application which uses the GSSAPI can "plug in" any security service which implements the API
- RPCSEC_GSS not widely implemented

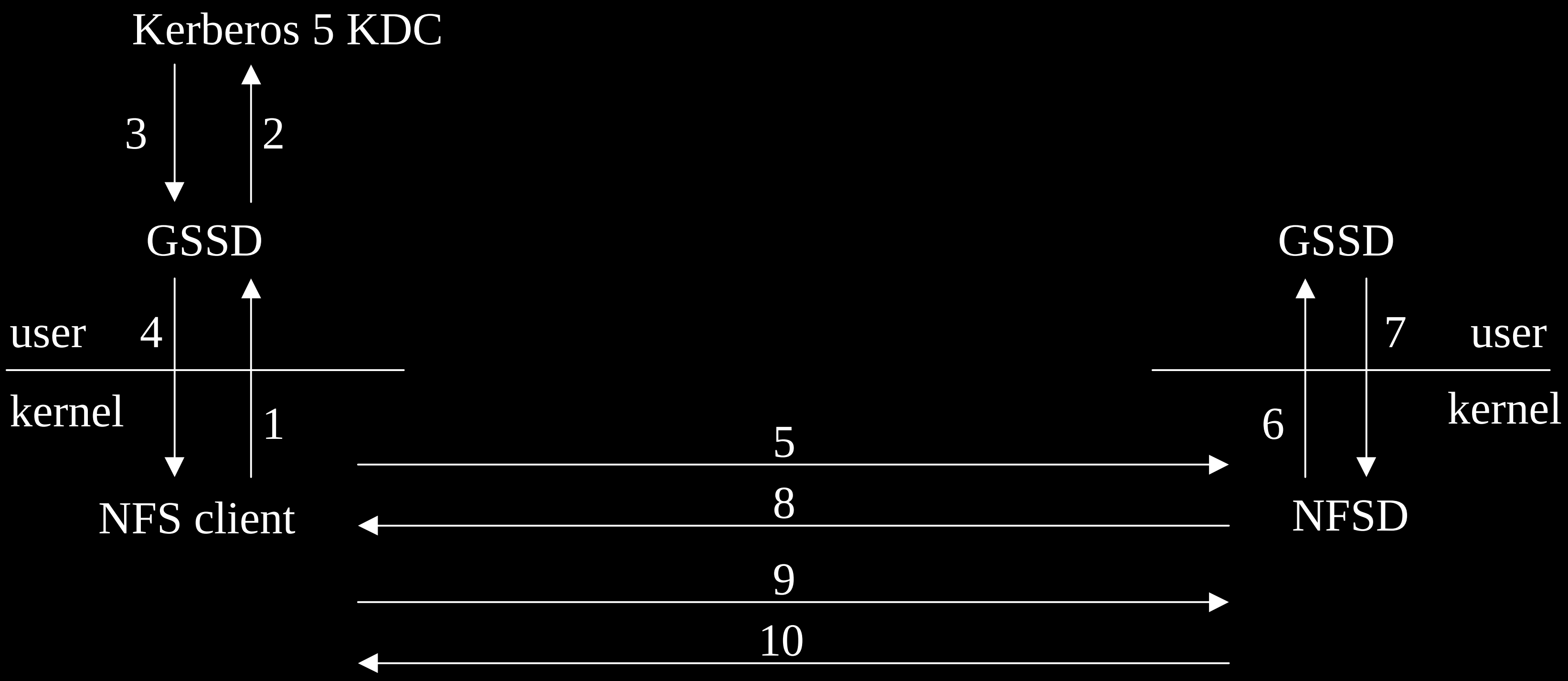
GSSAPI

- MIT Kerberos 5 currently implements the GSSAPI.
- We're implementing LIPKEY, a GSSAPI-based service that requires SPKM3.
- We’re implementing SPKM3, (Simple Public Key Mechanism)
- The combination of LIPKEY and SPKM3 provides a security service similar to TLS

RPCSEC_GSS

- Defines an on-the-wire scheme for embedding GSSAPI tokens into RPC requests.
- Initial exchange of context establishment tokens takes place over a sequence of overloaded RPC NULL procedures.

Kerberos 5 Security Initialization



2,3	Kerberos 5 TCP/IP
1,4,6,7	GSSD RPC interface
5,8	NFSV4 overloaded NULL procedure
9,10	NFSV4 COMPOUND procedure

RPCSEC_GSS

- RPC header, with an auth flavor of AUTH_GSS.
- Per message verifier hashed with a negotiated algorithm
- Import GSS context from GSSD after initialization
- Linux kernel crypto patch: currently cryptoapi-2.4.10.diff

SPKM3

- Server has public keypair, which is not required on client
- Similar to "one-sided" security services such as SSL, TLS, and as such, suffers from the same man-in-the middle attack
- Client public kepairs are allowed – we plan to use client machine keypairs

SPKM3

- Diffie-Hellman keyexchange in combination with server (and possibly client) PKI certificates establish a secure channel
- If User PKI credentials have not been used, User is still anonymous to the server

LIPKEY

- The user is prompted for a username and password which is sent to the server encrypted with SPKM3 established session key.
- LIPKEY does not specify how the server validates username and password.
- Linux: GSSD will attempt an ordinary unix login.

RPCSEC_GSS Implementation

- RPCSEC_GSS (userspace & in kernel)
- GSSD (userspace)
- SPKM3 (userspace)
- LIPKEY (userspace)
- Performance-critical GSSAPI calls for Kerberos 5 and LIPKEY (kernel)

File Delegation

- Server issues delegations to clients
- A read delegation on a file is a guarantee that no other clients are writing to the file.
- A write delegation on a file is a guarantee that no other clients are accessing the file.

File Delegation

- Eases revalidation requirements.
- Not necessary for correctness.
- Designed to reduce RPC requests to the server
- Expect performance enhancement
- Non-delegated files fall back to NFSv3 semantics

File Delegation

- Server may recall a delegation at any time when another client OPENs a file.
- Might not have two way reachability (server must probe callback path) before it knows it's safe to issue delegations.
- Client cannot regain a recalled delegation without another OPEN

File Delegation

- Delegation state management implemented
- Exploring two choices for client cache
  - Virtual memory system: let the pager store the files on disc
  - Local filesystem

Administrative Issues

- New server export model
- Namespace management
- Server state management

Export Management

- In NFSv3, clients must rely on an auxiliary protocol, the MOUNT protocol, to:
  - request a list of the server's exports
  - obtain the root file handle of a given export
- Changes to the server export list requires changes to the client mount request

Export Management

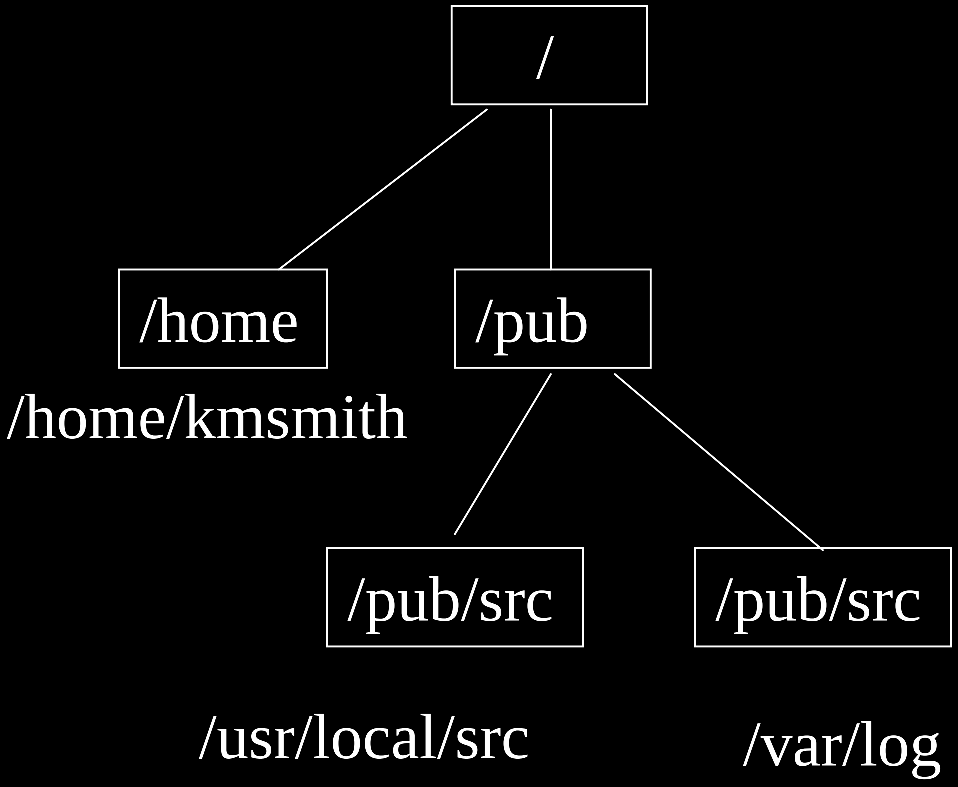
- NFSv4 pseudo fs allows the client to mount the root of the server, and browse to discover the offered exports
- Access into the pseudo fs is read-only and requires no credentials
- Access into an export is based on the users credentials.
- No more client list in /etc/exports

/etc/exports

A sample /etc/exports with the pseudo fs followed by the directories to be exported

```
/home       /home/kmsmith
/pub/src    /usr/local/src
/pub/log    /var/log
```

/etc/exports and the Pseudo FS



Namespace Management

- NFSv4 user and group names are of the form name@realm
- Translation between the names and UID/GID used by the local file system is needed
- Currently, /etc/passwd and /etc/group is used

State Management

- Sometimes client state on the server will need to be reaped by administrators
- We’re developing a TCL/TK based tool to manage server side state

Availability

- New and Improved Client and Server based on the Linux 2.4.4 kernel first week of November, 2001.
- OpenBSD port will be concentrated on in following months. We expect a release by years end.

Any questions?

`http://www.citi.umich.edu/`