NFS Version 4 Open Source Project

William A.(Andy) Adamson
Center for Information Technology Integration
University of Michigan

NFS Version 4 Open Source Project

- Sponsored by Sun Microsystems
- Part of CITI's Linux Scalability project
- IETF reference implementation
- 212 page spec
- Linux and OpenBSD
- Inter-operates with Solaris, Java, Network Appliance, Hummingbird, EMC implementation
- April 1, 2001 - Linux 2.4 release

NFS Version 4: What's New?

- Lots of state
- Compound RPC
- Extensile security added to RPC layer
- Delegation for files - client cache consistency
- Lease based non-blocking byte range locks
- Win32 share locks
- Mountd - gone.
- Lockd, statd - gone

NFS Version 4 State

- Compound RPC - server state
- Win32 share locks - server and client state
- Delegation - server and client state
- Byte-range locks - server and client state

Per Thread Global State

- Compound operations often use result of previous operation as arguments.
- NFS file handle is the coin of the realm
- *Current file handle* <=> Current working directory
- Some operations (RENAME) need two file handles - *Save file handle*.

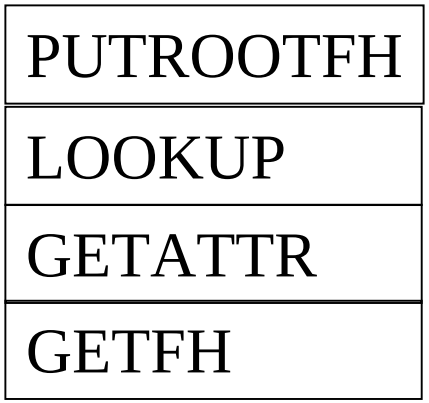
Compound RPC

- 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: kmalloc buffer for args and recv

Compound RPC Call Interface

- Goal is to XDR args directly into RPC buffer and to allow a variable length receive buffer
- Encode and decode routines not called from RPC layer
- Considering requesting buffer from RPC layer to remove one copy
- Decode handlers provide ideal place to handle common errors
- Use same calling interface for Linux and OpenBSD

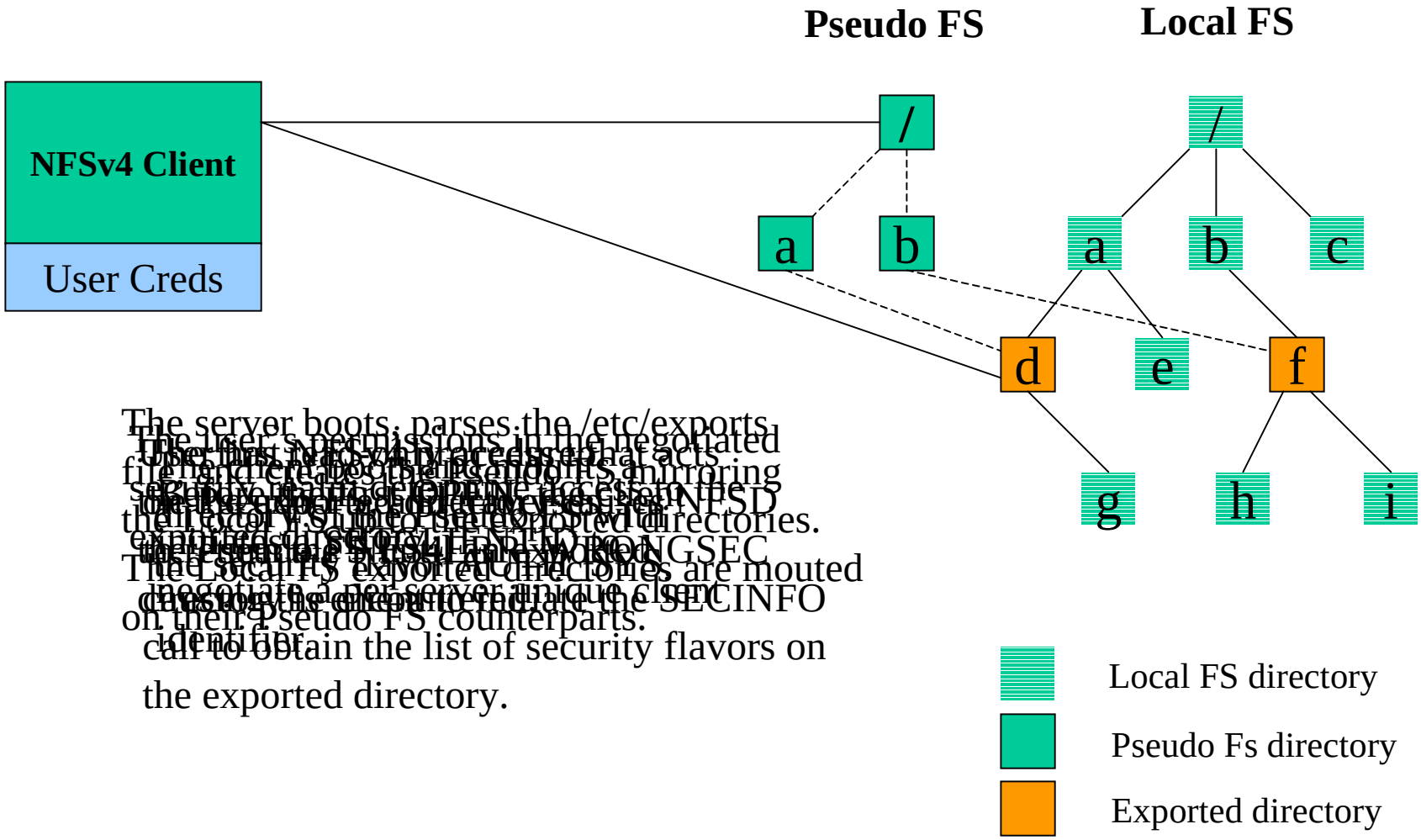
Mount Compound RPC



NFS v4 Mount

- Server *PseudoFS* joins exported sub trees with a read only virtual file system
- Any client can mount into the *PseudoFS*
- Users browse the *PseudoFS* (via LOOKUP)
- Access into exported sub trees based on user's credentials and permissions
- Client /etc/fstab doesn't change with servers export list
- Server /etc/exports doesn't need to maintain an IP based access list

Mount and the Pseudo File System

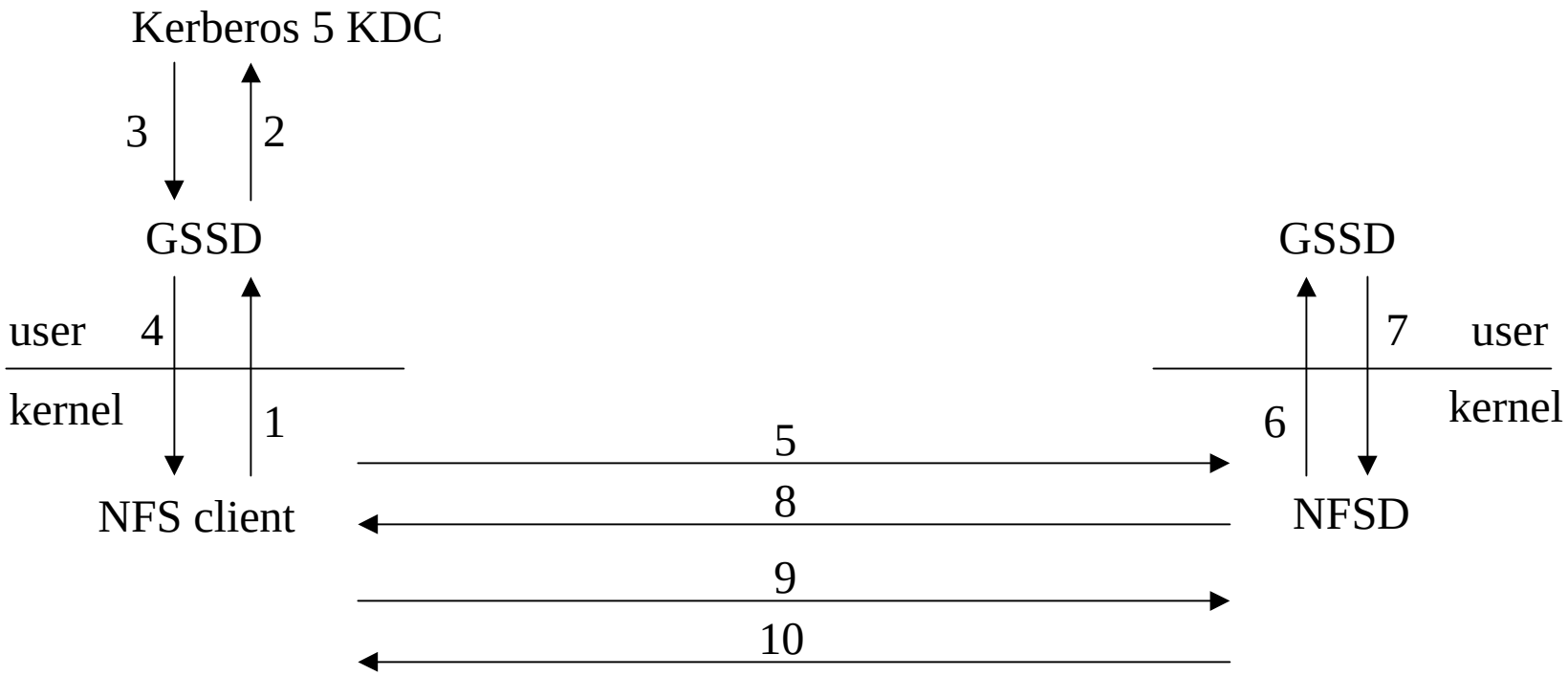


The server boots, parses the /etc/exports, and creates the Pseudo FS.

The LocalFS exported directories are mounted on their Pseudo FS counterparts.

call to obtain the list of security flavors on the exported directory.

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

- User-level
  - Complete (mostly) Kerberos 5 implementation
  - mutual authentication, integrity, privacy
  - inter-operates with Solaris
- Kernel
  - Kerberos 5
  - mutual authentication, no encryption
  - user-level daemon, GSSD
  - integrated with file system access
  - inter-operates with NetApp, Solaris

LIPKEY & SPKM3

- Adding LIPKEY to our user level RPCSEC_GSS
- Sits directly on top of SPKM3
- Enabled mechanism glue code in MIT kerberos 5
  - can switch on mechanism and hit SPKM3 calls
- Valicert asn1parser produces DER encode and decode routines that call SSLeay functions
  - doesn't handle all the necessary semantics (ANY, SEQUENCE OF SEQUENCE to name a few)
  - will have to hand code
- First pass SPKM3 Diffie-Hellman init_sec_context and accept_sec_context being coded

State: Server Locking

- Need to associate a file, lock, lockowner, & lease
- Per lockowner lock sequence number
- Server doesn’t own local file system structures
- Hash tables for clients, files, lockowners, locks
- Stateid: handle to server lock state
- Per client state: lock lease

State: Client Locking

- Client owns local file system structures, use private data areas
- Hash table for lockowners
- Delegation means that the client needs to hold the same locking state as the server

Byte-Range Locking

- Lease based locks. No byte range callback mechanism
  - – Server defines a lease for all per client lock state
  - – Server can reclaim all client state if lease not renewed
- OPEN sets lock state which includes a lockowner (clientid, pid)
- Server returns lock stateid
- Stateid mutating operations are ordered
  - –OPEN, OPEN_CONFIRM, CLOSE, LOCK, LOCKU, OPEN_DOWNGRADE

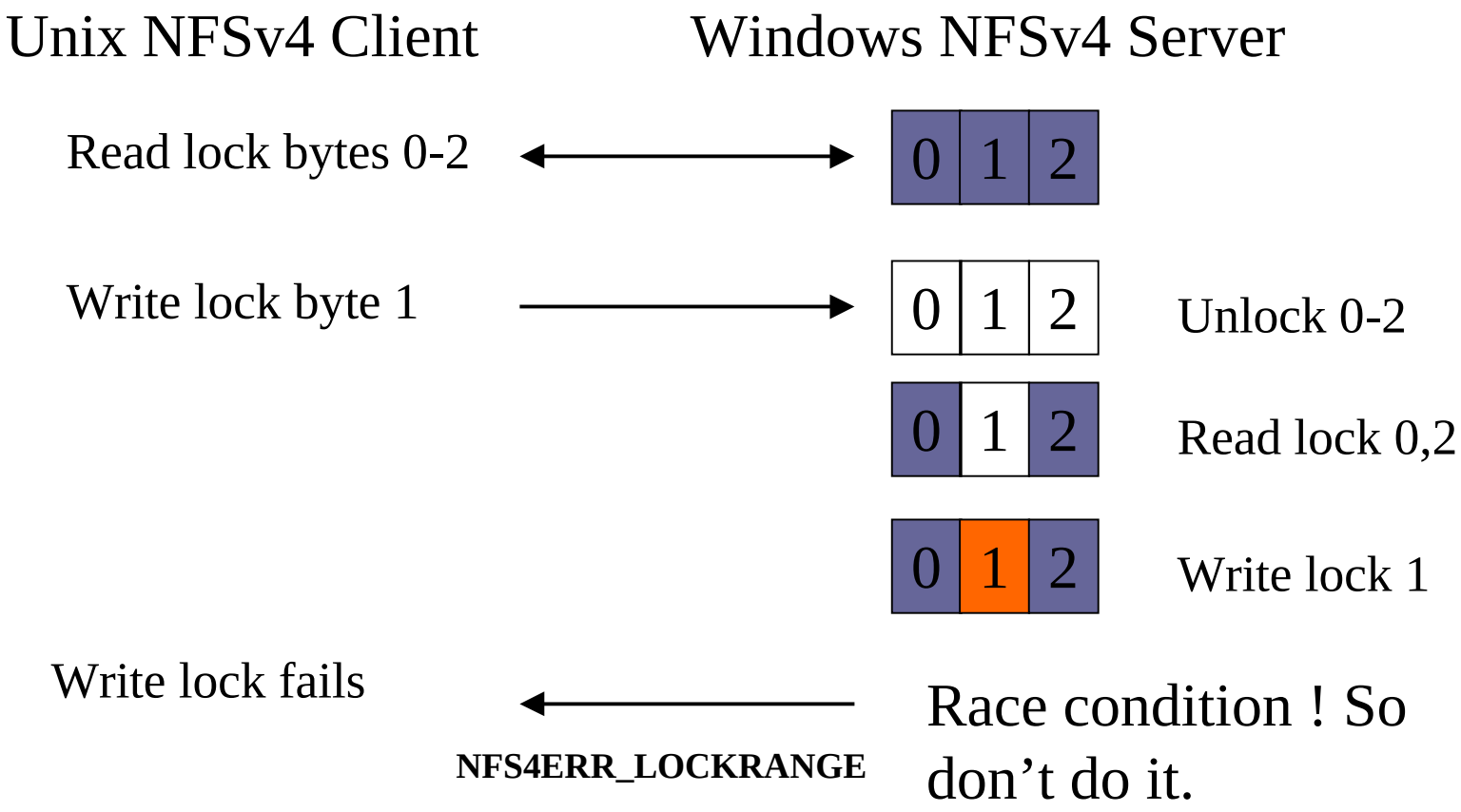
Byte-Range Locking

- NFSv4 tries to join POSIX and Win32 lock semantics
- Client: lockowner can obtain a byte range lock and then:
- Upgrade the initial lock (read lock -> write lock)
- Request a change to a sub-range of the initial lock

Byte-Range Locking

- Sub-range problem: POSIX splits and coalesces locks, Win32 doesn’t.
- NFSv4 allows for both behaviors, returning NFS4ERR_LOCKRANGE to signal non-support of sub-range semantics
- Useful for a client to be able to determine what type of byte-range locking support exists on a server

Sub-Range Problem



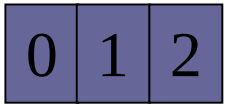
Sub-Range Problem

Unix NFSv4 Client Windows NFSv4 Server

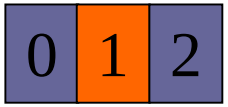
Open for write →

← Write delegation

Read lock bytes 0-2



Write lock byte 1



Delegation

- Goal is to reduce traffic
- Server decides to hand out delegation at OPEN
- If client accepts, client provides callback
- Many read delegations, or one write delegation
- When client delegates a cached file it handles:
  - all locking, share and byte range
  - future OPENS
- Client can’t reclaim a delegation without a new OPEN
- No delegation for directories

State: Server Delegation

- Associates delegation with a file
- Delegation state in linked list off file state
- Stateid: separate from the lock stateid
- Client call back path

Linux VFS Change

- Shared problem: OPEN with O_EXCL described by Peter Braam
- NFSV4 implements WIN32 share locks which require an atomic OPEN with CREATE
- Linux 2.2.x and Linux 2.4 VFS is problematic
- To CREATE and OPEN a file, three inode operations are called in sequence
- LOOKUP resolves the last name component
- CREATE is called to create an inode
- OPEN is called to open the file

XOPEN

- Inherent race condition means no atomicity
- We partially solved this problem
- We added a new inode operation which performs the OPEN system call in one step.
- int xopen(struct file *filep, struct inode *dir_i, struct dentry *dentry, int mode)
- if the xopen() inode operation is NULL, the current two step code is used
- NFSv4 OPEN subsumes LOOKUP, CREATE,OPEN,ACCESS

Namespace Issues

- Local file system uses uid/gid
- Protocol specifies <username>@<realm>
- No auth type associated with name in ACL
- UNIX username
- Kerberos 5 realm
- PKI realm - X500 or DN naming
- GSSD resolves <username>@<realm> to local file system representation - currently /etc/passwd

Open Issues

- Local file system choices
  - Currently ext2
  - ACL implementation will determine FS for Linux 2.4
  - Ext3, XFS both support local ACL
  - Linux developing ACL interface
- Kernel additions and changes
  - Crypto
  - Atomic OPEN

What’s Next

- April 1, 2001 - full Linux 2.4 implementation, without ACL’s
- July 1, 2001 - ACL’s added
- Network Appliance sponsored NFSv3/v4 Linux performance project

Questions?

http://www.citi.umich.edu/projects/nfsv4

http://www.nfsv4.org