A New Network File System is Born: SMB2

How does it stack up?
Is it worth implementing?

Steve French
Linux File Systems/Samba Design
IBM Linux Technology Center

http://svn.samba.org/samba/ftp/cifs-cvs/ols2007-smb2.pdf

Legal Statement

This work represents the views of the author and does not necessarily reflect the views of IBM Corporation.

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States and/or other countries: IBM (logo), A full list of U.S. trademarks owned by IBM may be found at http://www.ibm.com/legal/copytrade.shtml.

Linux is a registered trademark of Linus Torvalds.

Other company, product, and service names may be trademarks or service marks of others.

Outline

- What makes Network File Systems (and their protocols) different ...
- A short history
  - of SMB and the birth of SMB2
  - and of NFS
- SMB2 under the hood
- Comparison of SMB2, CIFS, SMB, NFS
- Problematic Linux operations
- Linux SMB2 implementation
- “Where do we go from here?”

Who Am I?

- Author and maintainer of Linux cifs network file system
- Veteran? Design/Developed various network file systems since 1989
- Member of the Samba team, coauthor of CIFS Technical Reference and former SNIA CIFS Working Group chair
- Architect for File Systems/NFS/Samba in IBM LTC

What is a File System?

- “a file system is a set of abstract data types that are implemented for the storage, hierarchical organization, manipulation, navigation, access, and retrieval of data” [http://en.wikipedia.org/wiki/Filesystem]
- A Linux kernel module used to access files and directories. A file system provides access to this data for applications and system programs through consistent, standard interfaces exported by the VFS
- This is much, much harder over a network ... which is why making **Network File Systems** is fun

What makes network file system developers lives miserable?

- Constraints from network fs protocol
- Bugs in various servers that must be worked around
- Races with other clients
- Recovery after failure
- Long, unpredictable network latency
- Hostile internet (security)
- More complex deadlocks and locking

Don't (always) blame the protocol ...

- Some problems are with the implementation (e.g. nfs.ko, cifs.ko) not with the protocol
- It takes a long time to get implementations right ... current Linux ones are still tiny (under 30KLOC)

Network File System Protocol Characteristics

- Network fs protocols differ from other application level protocols
  - Access via Files (and offsets within files) vs. Blocks
  - PDUs loosely match local fs (VFS) entry points
  - Support Hierarchical directory
  - Topologies/nets vary (server room, LANs, intranet or even Internet for some)
  - Application optimization possible
  - Transparency
  - Heterogeneity

Lots of Linux FS e.g.

58 Linux file systems (and 1 nfs server) in current kernel
not counting out of kernel fs: OpenAFS, GPFS, Lustre ...
nor the many fs and servers (Samba!) in user space...

FS Name	Type	Approx. size (1000 LOC)	
Proc	Spec. purp.	6	
Smbfs (obsol)	network	6	
ecryptfs	Spec. purp.	6	
AFS	Network	9	
Ext3	Local	12	
Ext4	Local	14	
GFS2	Cluster	19	(w/o dlm)
CIFS	Network	22	
NFS	Network	25	
OCFS2	Cluster	33	
XFS	Local	71	

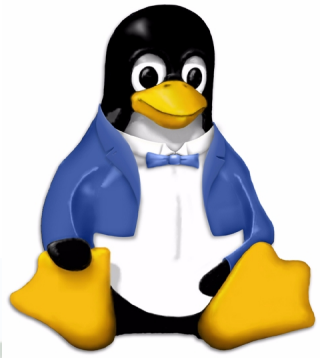
The Birth of Vista

- Release of Vista was in early 2007. Includes new default Network File System protocol: **SMB2**
- Prototype Implementations of SMB2 in Samba 4 by late 2006
- Wireshark support added even earlier

History of SMB/CIFS

Happy 23rd Birthday!

- Birth of SMB/CIFS: Dr. Barry Feigenbaum et al of IBM (published 1984 IBM PC Conf), continued by Intel, 3Com, Microsoft and others
- Became the default for DOS, Windows, OS/2, NT and various other OS.
- Evolved through various “dialects”

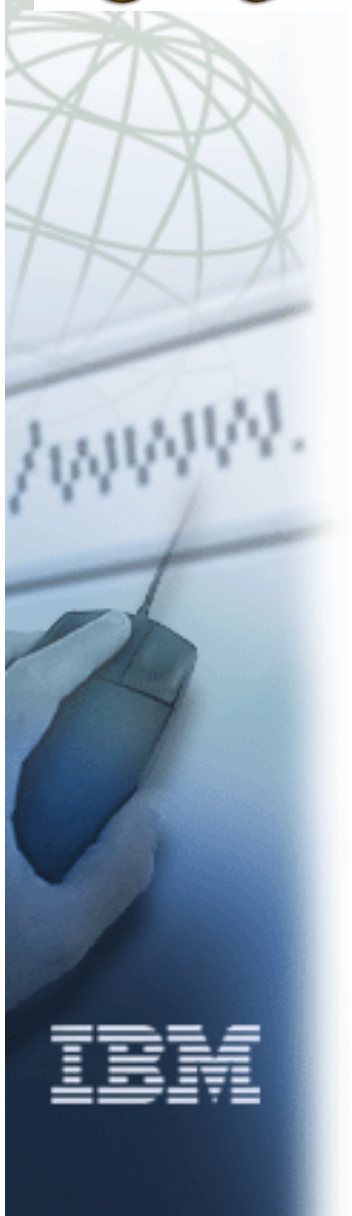


History (continued)

- 1992 X/Open CAE SMB Standard published (“LANMAN1.2” dialect)
- 1996 SMB renamed “CIFS” - Common Internet File System (“NT LM 0.12”)
- 2002 SNIA CIFS Technical Reference Published after 4 year effort (also includes Unix and Mac extensions)
- 2003-2007 Additional Extensions for Linux/Unix Documented and Implemented (on multiple clients and servers – not just Linux)

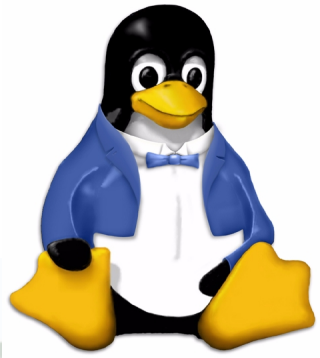
History of NFS

- NFS is born ... mid 1980's (Sun OS 2.0 1985)
- RFC 1094 (NFS v2) 1989
- RFC 1813 (NFSv3) 1995
- WebNFS 1996
- RFC 3530 (NFSv4) 2003
- NFS 4.1 documentation/prototyping (in progress) 2007-



And the alternatives?

- NFS v3 or v4
- AFS/DFS
- HTTP/WebDav
- Cluster Filesystem Protocols



Back where we started!

- Ancient NFS and SMB born mid-1980s and quickly became popular
- Lots of other network file systems died out in between
- HTTP/WebDAV too slow, and can't do POSIX
- No widely deployed cluster fs standard
- 2007: Back where we started with NFSv4 and SMB2 widely deployed and going to be dominant?

SMB2 Under the hood

- Not the same as CIFS but ... still reminiscent of SMB/CIFS
  - Same TCP port (445)
  - Small number of commands (all new) but similar underlying infolevels
  - Similar semantics

SMB2 vs. SMB/CIFS

- Header better aligned and expanded to 64 bytes (bigger uids, tids, pids)
- 0xFF “SMB” -> 0xFE “SMB”
- Very “open handle oriented” - all path based operations are gone (except OpenCreate)
- Redundant/Obsolete commands gone
- Bigger limits (e.g. File handle 64 bits)
- Better symlink support
- Improved DFS support
- “Durable File Handles”

SMB2 Create

```
SMB2 (Server Message Block Protocol version 2)
  SMB2 Header
      Server Component: SMB2
      Header Length: 64
      NT Status: STATUS_SUCCESS (0x00000000)
      Command: Create (5)
      unknown: 0000
    Flags: 0x00000000
        .... .... .... .... .... .... .... 0... = Signing: This pdu is
        .... .... .... .... .... .... .... ..0. = PID Valid: The pid fi
        .... .... .... .... .... .... .... ...0 = Response: This is a F
      unknown: 00000000
      Command Sequence Number: 4
      Process Id: 00000000 (not valid)
    Tree Id: 1  \\vista4\test
    User Id: 0x0000040000000001 Acct:tridge Domain:BLUDOM Host:BLU
      Signature: 00000000000000000000000000000000
      [Response in: 20]
  Create Request (0x05)
      Length: 56
      .... .... .... ...1 = Dynamic Part: True
    Create Flags: 0x0000
      Impersonation: Anonymous (0)
      unknown: 0000000000000000
      unknown: 0000000000000000
    Access Mask: 0x001f01ff
    File Attributes: 0x00000080
    Share Access: 0x00000007
      Disposition: Open If (if file exists open it, else create it) (3)
    Create Options: 0x00000002
    Filename: test9.dat
    ExtraInfo MxAc
```

SMB Create

```
SMB (Server Message Block Protocol)
  SMB Header
      Server Component: SMB
      [Response in: 27]
      SMB Command: NT Create AndX (0xa2)
      NT Status: STATUS_SUCCESS (0x00000000)
    Flags: 0x08
    Flags2: 0xc803
      Process ID High: 0
      Signature: 0000000000000000
      Reserved: 0000
    Tree ID: 2048
      Process ID: 26266
      User ID: 2048
      Multiplex ID: 8
  NT Create AndX Request (0xa2)
      Word Count (WCT): 24
      AndXCommand: No further commands (0xff)
      Reserved: 00
      AndXOffset: 0
      Reserved: 00
      File Name Len: 60
    Create Flags: 0x00000010
      Root FID: 0x00000000
    Access Mask: 0x0002019f
      Allocation Size: 0
    File Attributes: 0x00000080
    Share Access: 0x00000003
      Disposition: Create (if file exists fail, else create it) (2)
    Create Options: 0x00000040
      Impersonation: Impersonation (2)
    Security Flags: 0x03
      Byte Count (BCC): 63
      File Name: \rawopen\torture_ntcreatex.txt
```

A good new/old comparison from Tridge

octet 1	2	3	4	5	6	7	8
RFC 1001 msg type (session)	SMB length (some reserve top 7 bits)			0xFF	'S'	'M'	'B'
SMB Command	Status (error) code				SMB flags	SMB flags2	
Process ID (high order)	SMB Signature						
SMB signature (continued)	Reserved			Tree Identifier		Process Id (Low)	
SMB User Identifier	Word Count	(variable number of 16 bit parameters follow)			Byte Count (size of data area)		(data area follows)

Table 1: SMB Header Format (39 bytes + size of command specific wct area)

octet 1	2	3	4	5	6	7	8
RFC 1001 msg type (session)	SMB length			0xFE	'S'	'M'	'B'
SMB Header length (64)	reserved			Status (error) code			
SMB2 Command	Unknown			SMB2 Flags			
Reserved				Sequence number			
Sequence Number (continued)				Process Id			
Tree Identifier				SMB User Identifier			
SMB User Identifier				SMB Signature			
SMB Signature (continued)							
SMB Signature (continued)				SMB2 Parameter length (in bytes)		Variable length SMB Parm	Variable length SMB Data

Table 2: SMB2 Header Format (usually 68 bytes + size of command specific parameter area)

octet 1	2	3	4	5	6	7	8
SunRPC Fragment Header				XID			
Message Type (Request vs. Response)				SunRPC Version			
Program: NFS (100003)				Program Version (e.g. 3)			
NFS Command				Authentication Flavor (e.g. AUTH_UNIX)			
Credential Length				Credential Stamp			
Machine Name length				Machine name (variable size)			
Machine Name (continued, variable length)							
Unix UID				Unix GID			
Auxiliary GIDs (can be much larger)							
Verifier Flavor				Verifier Length			
NFS Command Parameters and/or Data follow							

Table 3: SunRPC/NFSv3 request header format (usually more than 72 bytes + size of nfs command)

19 known SMB2 PDUs (commands)

SMB2_NEGPROT 0x00

SMB2_SESSSETUP 0x01

SMB2_LOGOFF 0x02

SMB2_TCON 0x03

SMB2_TDIS 0x04

SMB2_CREATE 0x05

SMB2_CLOSE 0x06

SMB2_FLUSH 0x07

SMB2_READ 0x08

SMB2_WRITE 0x09

SMB2_LOCK 0x0a

SMB2_IOCTL 0x0b

SMB2_CANCEL 0x0c

SMB2_KEEPALIVE 0x0d

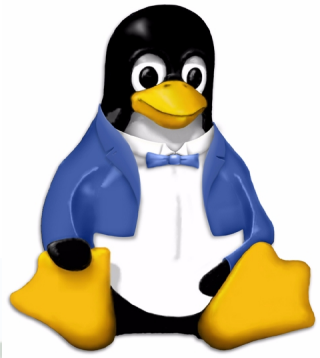
SMB2_FIND 0x0e

SMB2_NOTIFY 0x0f

SMB2_GETINFO 0x10

SMB2_SETINFO 0x11

SMB2_BREAK 0x12



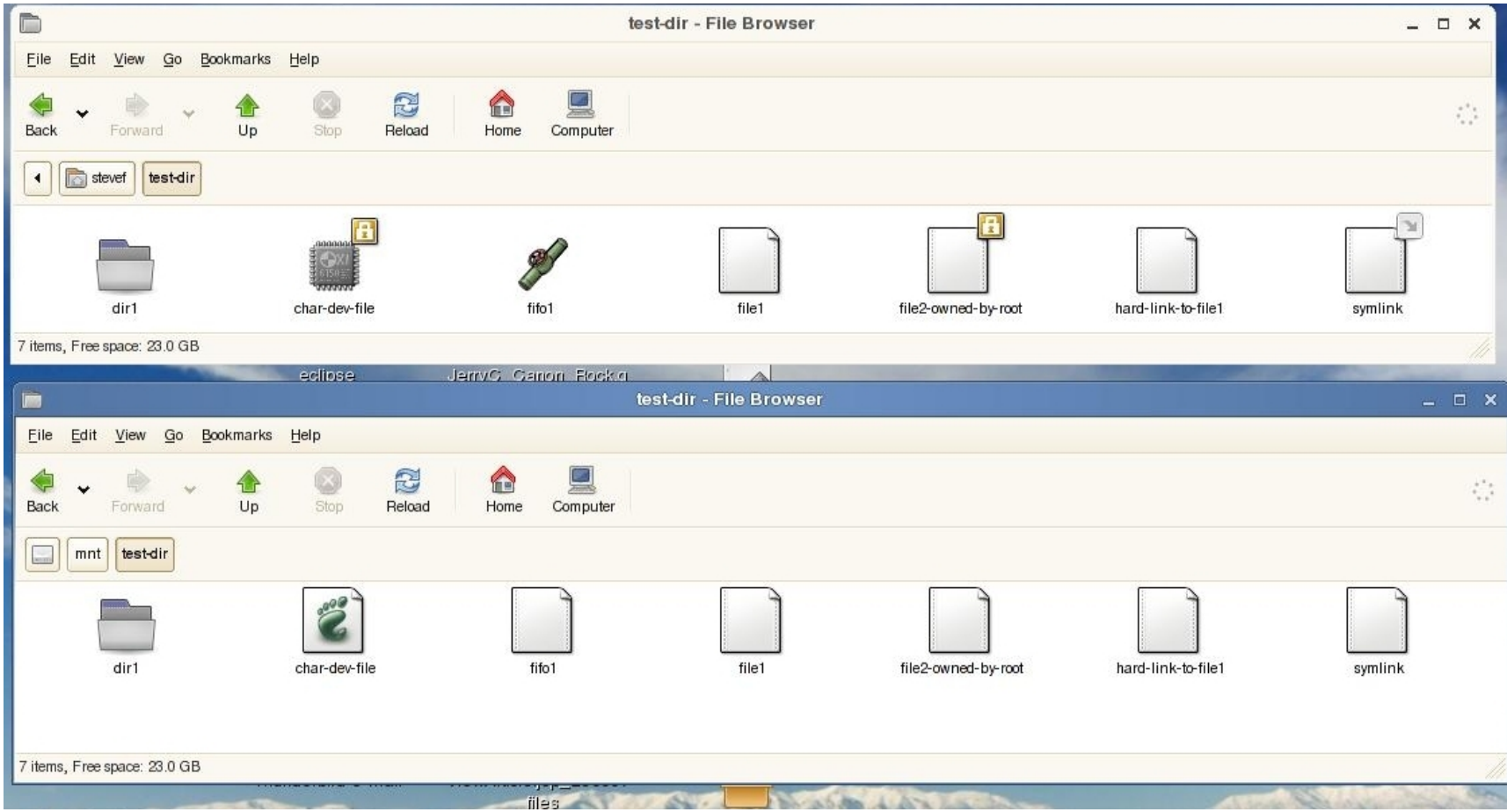
Other protocols

- SMB/CIFS has more than 80 distinct SMB commands (Linux CIFS client only needs to use 21). A few GetInfo/SetInfo calls, similar to SMB2, have multiple levels
- NFS version 2 had 17 commands (NFS version 3 added 8 more), but that does not count locking and mount which are outside protocol
- NFS version 4 has 37 commands (dropped some, added 25 more) but moved locking into core

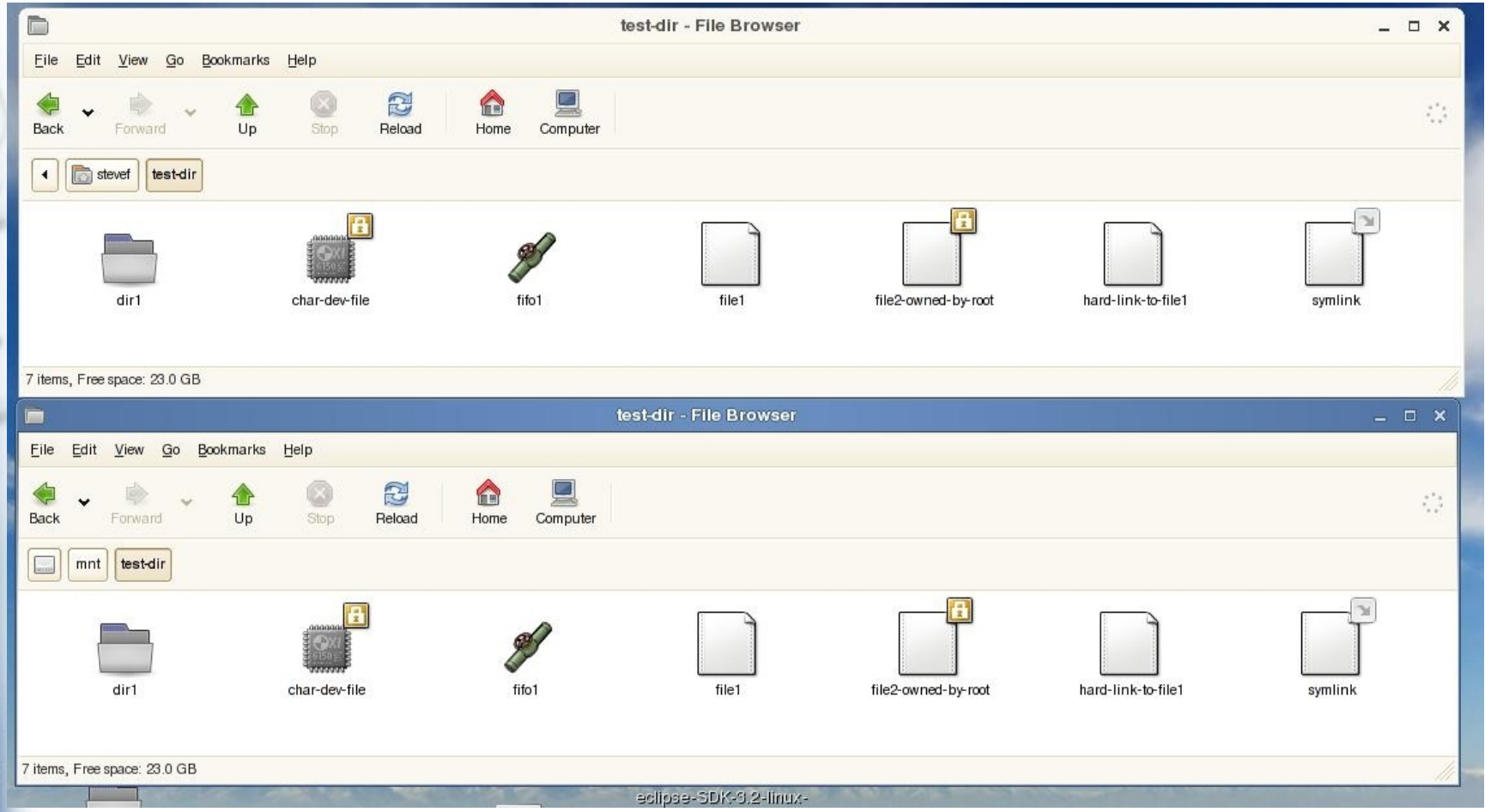
CIFS Linux (POSIX) Protocol Extensions

- The CIFS protocol without extensions requires awkward compensations to handle Linux
- Original CIFS Unix Extension (documented by HP for SNIA five years ago) helped:
  - Required only modest extensions to server
  - Solved key problems for POSIX clients including:
    - How to return: UID/GID, mode
    - How to handle symlinks
    - How to handle special files (devices/fifos)
  - But needed improvements

POSIX Conformance hard for original CIFS



CIFS with Protocol Extensions (CIFS Unix Extensions)



What about SFU approach?

- Lessons from SFU:
  - Map mode, group and user (SID) owner fields to ACLs
  - Do hardlinks via NT Rename
  - Get inode numbers
  - Remap illegal characters to Unicode reserved range
  - FIFOs and device files via OS/2 EAs on system files
- OK, **but not good enough** …
  - Some POSIX byte range lock tests fail
  - Semantics are awkward for symlinks, devices
  - UID mapping a mess
  - Performance slow
  - Operations less atomic and not robust enough
  - Rename/delete semantics hard to make reliable

CIFS Unix Extensions

- Problem ... a lot was missing:
  - Way to negotiate per mount capabilities
  - POSIX byte range locking
  - ACL alternative (such as POSIX ACLs)
  - A way to handle some key fields in statfs
  - Way to handle various newer vfs entry points
    - lsattr/chattr
    - Inotify
    - New xattr (EA) namespaces

Original Unix Extensions Missing POSIX ACLs and statfs info

```
smf-t41p:/home/stevef # getfacl /mnt/test-dir/file1
# file: mnt/test-dir/file1
# owner: root
# group: root
user::rwx
group::rw-
other::rwx

smf-t41p:/home/stevef # stat -f /mnt1
  File: "/mnt1"
    ID: 0        Namelen: 4096    Type: UNKNOWN (0xff534d42)
Block size: 1024       Fundamental block size: 1024
Blocks: Total: 521748     Free: 421028     Available: 421028
Inodes: Total: 0          Free: 0
```

With CIFS POSIX Extensions, ACLs and statfs better

```
smf-t41p:/home/stevef # getfacl /mnt/test-dir/file1
# file: mnt/test-dir/file1
# owner: stevef
# group: users
user::rw-
user:stevef:r--
group::r--
mask::r--
other::r--

smf-t41p:/home/stevef # stat -f /mnt1
  File: "/mnt1"
    ID: 0        Namelen: 4096    Type: UNKNOWN (0xff534d42)
Block size: 4096       Fundamental block size: 4096
Blocks: Total: 130437     Free: 111883     Available: 105257
Inodes: Total: 66400      Free: 66299
```

POSIX Locking

- Locking semantics differ between CIFS and POSIX at the application layer.
  - CIFS locking is mandatory, POSIX advisory.
  - CIFS locking stacks and is offset/length specific, POSIX locking merges and splits and the offset/lengths don't have to match.
  - CIFS locking is unsigned and absolute, POSIX locking is signed and relative.
  - POSIX close destroys all locks.

Protocol changes

- The mandatory/advisory difference in locking semantics has an unexpected effect.
- READX/WRITEX semantics must change when POSIX locks are negotiated.
  - Once POSIX locks are negotiated by the SETFSINFO call, the semantics of READ/WRITE CIFS calls change – they ignore existing read/write locks.
  - POSIX-extensions aware clients probably *want* these semantics.
    - It's a side effect, but a good one !

Problematic Operations

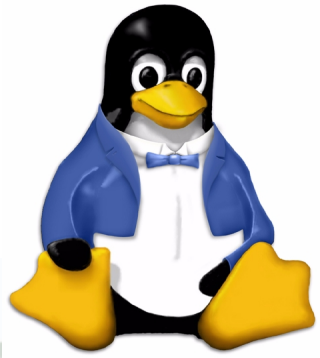
Olaf's “Why NFS Sucks” Talk at OLS 2006

The Nightmare Filesystem

- Some are hard to address (NFS over TCP still can run into retransmission checksum issues http://citeseer.ist.psu.edu/stone00when.html)
- Silly rename sideffects
- Byte Range Lock security
- Write semantics
- Lack of open operation lead to weak cache consistency model
- Most of these issues were addressed with NFSv4 as Mike Eisler pointed out

CIFS has problems too

- There is an equivalent of “commit” but it is not as commonly used (ie to force server to flush its server side caches and write to metal)
- No grace period for lock/open recovery after server is rebooted (clients can race to reestablish state)



Some questions even with NFSv4 ...

- Does extra layer between NFS and TCP (SunRPC), which is still required in v4, get in the way?
- Can RPSEC_GSS performance overhead be reduced enough?
- ACL mapping problems (NFSv4 ACLs are almost NTFS/CIFS ACLs but not quite). Management of ACLs from both sides (Windows or CIFS vs. NFSv4) could break. What about the ACL mask?
- UID -> username@domain mapping overhead
- “stable file handles” and even stable file system ids still a pain on many modern fs!

And more to analyze for NFSv4

- “Close to Open”and cache consistency
- utimes -> fsync (hurts performance)
- “COMMIT” and periodic write stalls
- What about “**Linux Affinity**?” How well does NFSv4 or CIFS map to the Linux VFS entries needed by applications (not just the minimal POSIX file calls)

•Linux has complex FS operations to implement

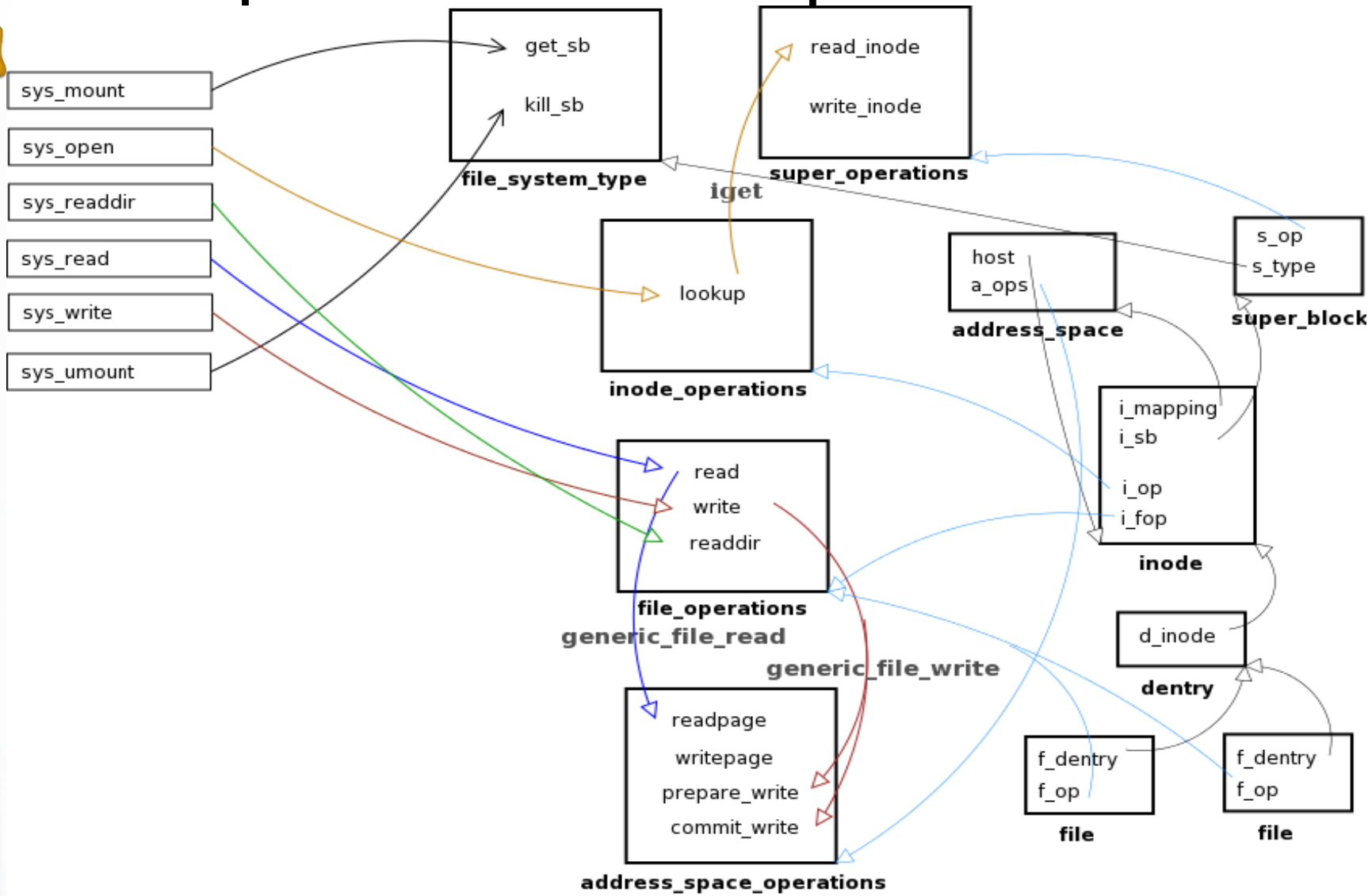


Fig: System call mapping and data structure relationships

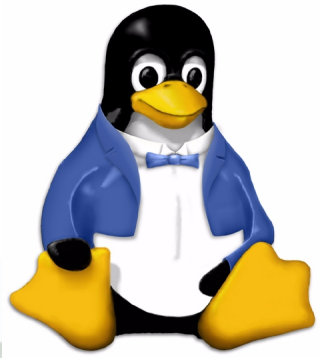
Source: http://www.geocities.com/ravikiran_uvs/articles/rkfs.html

All network fs can handle simple inode operations

- Linux inode operations
  - create
  - mkdir
  - unlink (delete)
  - rmdir
  - mknod
- Note vfs operations not all atomic (sometimes POSIX calls generate more than one vfs op although “lookup intents” help for nasty case of create) Some compensations are needed

Multipage operations and wsize/rsize

- For High Performance networks transfer size > 1MB may be optimal
- Linux has two interesting high performance page cache read/write op (nfs and cifs use both)
  - Readpages (10 filesystems use)
  - Writepages (a slightly different set of 10 filesystems use)
- Useful for coalescing reads/writes together to allow more efficiency
- Eventually RDMA-like features will be introduced into Linux network fs



readdir

- For network filesystems “ls” can cause “readdir” storms (hurting performance) by immediately following readdir with lots of expensive stat (and sometimes xattr/acl) calls (unless the stat results are requested together, or cached)
- Whether network equivalent of readdir can act as a “bulk stat” operation can affect performance

Byte range locks, leases/distributed caching

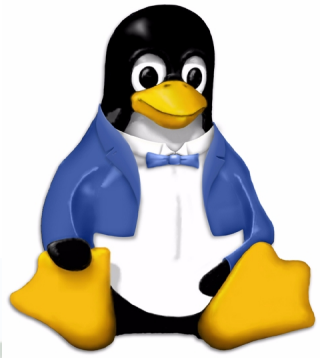
- Linux supports the standard POSIX byte range locking but also supports “leases”
- F_SETLEASE, F_GETLEASE, used by programs like Samba server, help allow servers to offer safe distributed caching (e.g. “Oplock” [cifs] and “delegations” [nfs4]) for network/cluster filesystems

Dir change tracking – inotify, d_notify

- There are two distinct mechanisms for change notification in Linux
  - Fcntl F_NOTIFY
  - And the newer, more general inotify

Xattrs

- Xattrs, similar in some ways to OS/2 “EAs” allow additional inode metadata to be stored
- This is particular helpful to Samba to store inode information that has no direct equivalent in POSIX, but a different category (namespace) also is helpful for storing security information (e.g. SELinux) and ACLs



POSIX ACLs, permissions

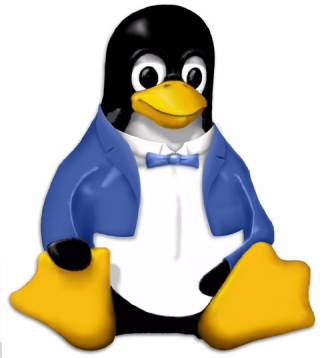
- Since Unix Mode bits are primitive, richer access control facility was implemented (based on an expired POSIX draft for ACLs).
- Now working w/CITI, Andreas et al to offer optional standard NFS4 ACLs (NFSv4 ACLs loosely based on CIFS)
- POSIX ACLs are handled via xattr interface, but can be stored differently internal to the filesystem. A few filesystems (including CIFS and NFS) can get/set them natively to Linux servers (but not for NFSv4)

Misc entry points: fcntl, ioctl

- Fcntl useful not just for get/setlease
- Ioctl includes two “semi-standard” calls which fs should consider implementing
  - getflags/setflags (chattr, lsattr on some other platforms)

Conclusion

- NFSv4 client in short term better performing in most (not all) workloads. Harder to configure for security though (AD is everywhere)
- With the newer Linux Extensions, CIFS to Samba is a great alternative
- CIFS (the implementation) missing some key features to catch up with competition
- CIFS will still be necessary for newer Windows until SMB2 support in kernel matures (we need to start now). To newer Windows servers use of SMB2 would be slightly better than CIFS
- We need to evaluate adding the Linux/Unix/POSIX extensions to SMB2 for Samba as we did with CIFS



What about an in kernel SMB2 or CIFS server?

- CIFS has too many protocol operations to do complete server implementation, but a limited implementation of just the minimum needed for clients using the POSIX CIFS extensions would be feasible
- SMB2 has fewer operations and may be feasible in kernel (with user space helpers for SID translation and Kerberos/SPNEGO session establishment etc.)

SMB2 Status

- Samba 4 server
- Samba 4 client and libraries
- Linux kernel client

4 obvious Linux SMB2 Implementation Alternatives

- Part of cifs.ko, add a few new C files (more complex to understand code, easy to start)
- Start from scratch, make smaller implementation (new smb2.ko)
- Borrow heavily from cifs vfs for new smb2.ko
- [or one could do nothing and hope Vista/Longhorn etc. go away]

- SMB2 support in kernel should be fairly easy to start, and fun ...
- Contact us on linux-cifs-client@lists.samba.org and samba-technical@lists.samba.org
- And linux-fsdevel@vger.kernel.org

Acknowledgements

Thanks to the Samba team, members of the SNIA CIFS technical work group, and others in analyzing and documenting the SMB/CIFS protocol and related protocols so well over the years. This is no easy task. In addition, thanks to the Wireshark team and Tridge for helping the world understand the SMB2 protocol better. Thanks to Jeremy Allison for helping me drive better Linux extensions for CIFS. Thanks to the Linux NFSv4 developers, the NFS RFC authors, and to Olaf Kirch, Mike Eisler for making less opaque the very complex NFSv4 protocol.

Thank you for your time!

For further reading

- This presentation and SMB2 and CIFS info:
  - http://svn.samba.org/samba/ftp/cifs-cvs/ols2007-smb2.pdf
  - Dr. A. Tridgell. Exploring the SMB2 Protocol. http://samba.org/~tridge/smb2.pdf
  - http://wiki.wireshark.org/SMB2
  - CIFS Unix Extensions Documentation http://wiki.samba.org/index.php/UNIX_Extensions
  - http://linux-cifs.samba.org
  - my paper in OLS proceedings has bigger bibliography
- NFS
  - “Why NFS sucks” by Olaf Kirch at OLS2006
  - And Mike Eisler's response ... http://nfsworld.blogspot.com/2006/10/review-of-why-nfs-sucks-paper-from.html
  - Checksum problem http://citeseer.ist.psu.edu/stone00when.html

Status

- Linux CIFS client
  - Version 1.49 (Linux 2.6.22) A year ago at this time ... cifs version 1.43
  - (1.43 included the much improved POSIX locking)
  - Version 1.32 included POSIX ACLs, statfs, lsattr
- Smbclient
  - Samba 3.0.25 includes client test code for POSIX locking, POSIX open/unlink/mkdir.
- HP/UX client also supports Unix Extensions
- Sun is developing a kernel CIFS client for Solaris
- Server
  - Samba 3.0.25 includes POSIX Locking (POSIX ACLs, QFSInfo, Unix Extensions implemented before) and POSIX open/unlink/mkdir.

A year in review ... (for the client)

- Growing fast (well over 100 changesets per year ...), one of the larger (22KLOC) kernel filesystems
- Write performance spectacularly better on 3 of 11 iozone cases
- POSIX locking, lock cancellation support (and much better POSIX byte range lock emulation to Windows)
- NTLMv2 (much more secure authentication, and new “sec=” mount options)
- Older server support (OS/2, Windows 9x)
- “deep tree” mounts
- New mkdir reduces 50% of network requests for this op
- Improved atime/mtime handling (and better performance)
- Improved POSIX semantics (lots of small fixes)
- Ipv6 support
- Can be used for home directory now ... everything should work!

Two recent examples

- mkdir improvement:
  - connectathon test1 (7 level deep dir creation, 1 file in each, 21845 directories)
  - 35% fewer frames sent, test completes 28% faster
- iozone write improvement
  - more than 10 times faster on 3 iozone write tests of 11
  - smf-t60p:/cifs-with-patch # time dd if=/dev/zero of=/cifs/.test bs=1024
  - count=250000
  - 256000000 bytes (256 MB) copied, 34.9132 s, 7.3 MB/s
  - (without patch) smf-t60p:/cifs # time dd if=/dev/zero of=/cifs/.test bs=1024
  - 256000000 bytes (256 MB) copied, 77.5971 s, 3.3 MB/s

Newest code

- POSIX OPEN/CREATE/MKDIR
- POSIX “who am I” (on this connection)
- POSIX stat/lookup
- Under development (3.0.27+ ?) -
  - CIFS transport encryption (GSSAPI encrypt at the CIFS packet level).
  - Based on authenticated user (vuid) – encryption context per user.
  - Allows mandatory encryption per share.

Roadmap

- Client
  - 2.6.22 includes new mkdir (new open/create and unlink in 2.6.23)
- Server
  - Samba 3.0.25 is complete (needs documentation). Encryption under development. Large (>128K) read support complete on server only.
  - Samba 4 Unix/POSIX Extensions started with new POSIX CIFS client backend
- In discussions with other client and server vendors about feature needs

Windows client/POSIX interaction

- POSIX clients read/write requests conflict with Windows locks, but not POSIX locks (Windows locks are mandatory for POSIX clients).
- Windows clients read/write requests conflict with both Windows and POSIX locks (both lock types are mandatory for Windows clients).
- Windows locks are set, unlocked and canceled via LOCKINGX (0x24) call.
- POSIX locks are set and unlocked via the Trans2 SETFILEINFO call, and canceled via the NTCANCEL call.

A few Extensions still needed

- inotify
- A few ioctls such as lsattr/chattr/chflags (currently implemented only in cifs client) e.g. To make a file immutable, or append-only, or to zero blocks on delete.

stevef@smf-t41p:~/test-dir> lsattr /boot/append-only-file
-----ad------ /boot/append-only-file
stevef@smf-t41p:~/test-dir> lsattr /mnt1/append-only-file
lsattr: Inappropriate ioctl for device While reading
flags on /mnt1/append-only-file

POSIX Errors

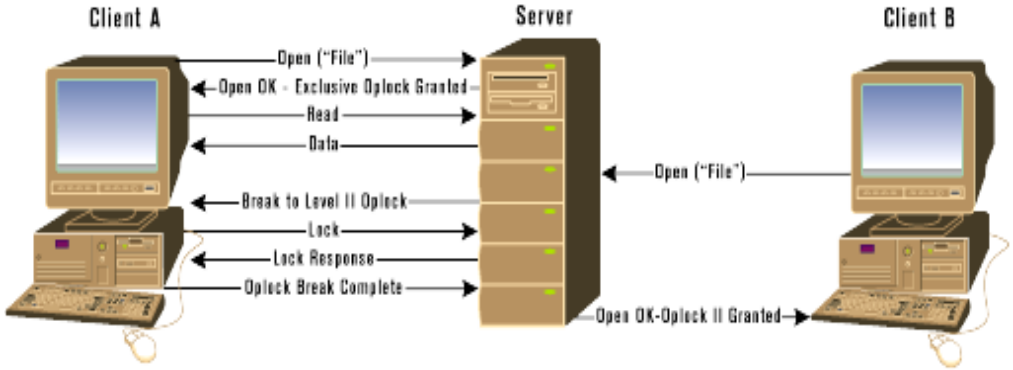
- NT Status codes (16 bit error nums) already has a reserved range
  - 0xF3000000 + POSIX errnum
  - POSIX errnum vary in theory, but not much in practice for common ones use
  - POSIX errnums fixed
  - New capability(will probably be)
    - #define CIFS_UNIX_POSIX_ERRORS 0x20
  - Do we need to define new errmapping SMB for client to resolve unknown POSIX errors backs to NT Status?

More general improvements still needed in our aging protocol

- These changes were not really Unix or Linux specific but POSIX apps may have stricter assumptions
- Full local/remote transparency desired
- Need near perfect POSIX semantics over cifs
- Newer requirements
  - Better caching of directory information
  - Improved DFS (distributed name space)
  - Better Performance
  - Better recovery after network failure
  - QoS

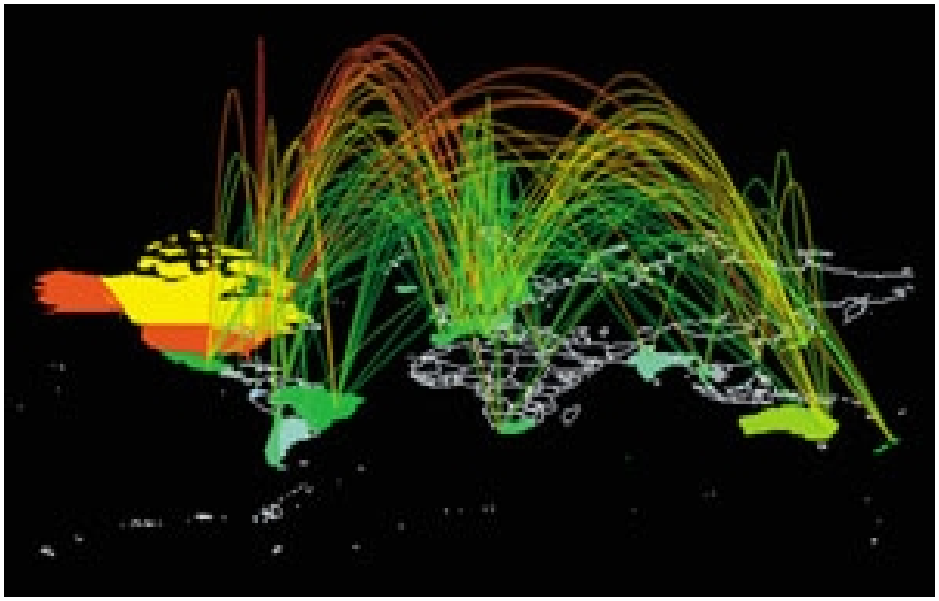
Caching improvements

- "Reacquire Oplock" concept
- FCNTLs already defined/reserved for this
  - #define FSCTL_REQUEST_OPLOCK_LEVEL_1 0x00090000
  - #define FSCTL_REQUEST_OPLOCK_LEVEL_2 0x00090004
  - #define FSCTL_REQUEST_BATCH_OPLOCK 0x00090008
  - #define FSCTL_REQUEST_FILTER_OPLOCK 0x0009008C
- Current work going on to test this



Source: http://www.microsoft.com/mind/1196/cifs.asp

DFS (Global Namespace) improvements



- DFS patch being reviewed
  - Part has already been merged in
- We need to improve ability to find nearest replica, and recover after failure
- And also to hint "least busy" server for load balancing

New Transports

- Ipv6 support here but ...
- reduce fragmentation / reassembly performance penalty (Ipv6 and others can leverage jumbo frames)
- To adapt to larger writes
- Other transports (Infiniband/RDMA)
  - Reduced latency
  - Improved performance
  - Quality of Service
-

Beating the competition - NFSv4

- Key differences
  - CIFS is richer protocol (huge variety of network filesystem functions available in popular servers)
  - CIFS supports Windows and POSIX model through different commands as necessary
  - CIFS can negotiate features with more flexibility: on a “tid” not just a session (or RPC pipe). This is helpful in tiered/gateway/clustered environments
  - CIFS does not have SunRPC baggage
  - And we have the Samba team ...
- And we are easier to configure than most cluster filesystems ...

Where to go from here?

- Discussions on samba-technical and linux-cifs-client mailing lists
- For Linux CIFS Extensions and CIFS: Wire layout is visible in fs/cifs/cifspdu.h
- For SMB2, see the Samba 4 source
- Working on updated draft reference document for these cifs protocol extensions
- See http://samba.org/samba/CIFS_POSIX_extensions.html