A New Network File System is Born: Comparison of SMB2, CIFS and NFS

Steven M. French
IBM
Samba Team
sfrench@us.ibm.com

Abstract

In early 2007, SMB2 became the first widely deployed network file system protocol since NFS version 4. This presentation will compare it with its predecessors (CIFS and SMB) as well as with common alternatives. The strengths and weaknesses of SMB/CIFS (the most widely deployed network file system protocol) and NFS versions 3 and 4 (the next most popular protocols) and SMB2 will be described.

Now that the CIFS POSIX Protocol extensions are implemented in Linux kernel, Samba, and multiple operating systems, whether SMB2 would be better for Linux in particular than these CIFS POSIX Protocol extensions can be analyzed. In addition, some of the alternatives such as HTTP, WebDav and cluster file systems will be reviewed. Implementations of SMB2 so far include not just Vista and Longhorn but also Samba client libraries and decoding support already is available in Wireshark. Linux implementation progress and alternatives for SMB2 client and server will also be described, and recommendations made for future work in this area.

1 Introduction

The SMB2 protocol, introduced in Microsoft Vista this year, is the default network file system on most new PCs, and differs from its predecessors in interesting ways.

Although a few experimental network file system protocols were developed earlier, the first that were widely deployed date from the mid-1980s: SMB (by IBM, Microsoft and others), AT&T's RFS protocol, AFS from Carnegie-Mellon University, NFS version 2 (from Sun)[1] and Novell's NCP. The rapid increase in numbers of personal computers and engineering workstations quickly made network file systems an important mechanism for sharing programs and data. More than twenty years later, the successors to the ancient NFS and SMB protocols are still the default network file systems on almost all operating systems.

Even if HTTP were considered a network file system protocol, it is relatively recent, dating from the early 1990s, and its first RFC [RFC 1945] was dated May 1996. HTTP would clearly be a poor protocol for a general purpose network file system on most operating systems including Linux. Since HTTP lacked sufficient support for "distributed authoring" without locking operations, with little file metadata and lacking directory operations, "HTTP Extensions for Distributed Authoring – WEBDAV" (RFC 2518) was released in February 1999, but WEBDAV did not displace CIFS or NFS, and few operating systems have a usable in-kernel implementation of WEBDAV.

So after more than twenty years, despite the invention of some important cluster file systems and the explosion of interest in web servers, we are almost back where we started, comparing NFS [3] Version 4 with the current CIFS extensions and with a new SMB—the SMB2 protocol. File systems still matter; network file systems are still critical in many small and large enterprises. File systems represent about 10% (almost 500KLOC) of the 2.6.21 Linux Kernel source code, and are among the most actively maintained and optimized components. The nfs[1] and cifs modules are among the larger in-kernel file systems. Network file systems matter—the protocols that they depend on are more secure, more full featured and much more complex than their ancestors, and some of the better NAS[2] implementations can perform as well as SAN and cluster file systems for key workloads.

2 Network File System Characteristics

Network protocols can be considered to be layered. Network file system protocols are the top layer, far removed from the physical devices such as Ethernet adapters that send bits over the wire. In the Open System Interconnection (OSI) model, network file system protocols would be considered as layer 6 and 7 ("Presentation" and "Application") protocols. Network file system protocols rely on lower level transport protocols (e.g., TCP) for reliable delivery of the network file systems protocol data units (PDUs), or include intermediate layers (as NFS has done with SunRPC) to ensure reliable delivery.

Network file system protocols share some fundamental characteristics that distinguish them from other "application level" protocols. Network file system clients and servers (and the closely related Network Attached Storage, NAS, servers) differ in key ways from cluster file systems and web browsers/servers.

- Files vs. Blocks or Objects: This distinction is easy to overlook when comparing network file system protocols with network block devices, cluster file systems and SANs. Network file systems read and write files not blocks of storage on a device. A file is more abstract—a container for a sequential series of bytes. A file is seekable. A file conventionally contains useful metadata such as ACLs or other security information, timestamps and size. Network file systems request data by file handle or filename or identifier, while cluster file systems operate on raw blocks of data. Network file system protocols are therefore more abstract, less sensitive to disk format, and can more easily leverage file ownership and security information.
- Network file system protocol operations match local file system entry points: Network file system protocol operations closely mirror the function layering of the file system layer (VFS) of the operating system on the client. Network file system

[1] lowercase "nfs" and "cifs" are used to refer to the implementation of the NFS and CIFS protocol (e.g., for Linux the `nfs.ko` and `cifs.ko` kernel modules), while uppercase "NFS" and "CIFS" refer to the network protocol

[2] Network Attached Storage (NAS) servers are closely related to network file servers.

operations on the wire often match one to one with the abstract VFS operations (read, write, open, close, create, rename, delete) required by the operating system. The OS/2 heritage of early SMB/CIFS implementations and the Solaris heritage of NFS are visible in a few network file system requests.

- Directory Hierarchy: Most network file systems assume a hierarchical namespace for file and directory objects and the directories that contain them.

- Server room vs. intranet vs. Internet: Modern network file system protocols have security and performance features that make them usable outside of the server room (while many cluster file systems are awkward to deploy securely across multiple sites), but HTTP and primitive FTP are still the most commonly choices for file transfers over the Internet. Extensions to NFS version 4 and CIFS (DFS) allow construction of a global hierarchical namespace facilitating transparent failover and easier configuration.

- Application optimization: Because the pattern of network file system protocol requests often more closely matches the requests made by the application than would be the case for a SAN, and since the security and process context of most application requests can be easily determined, network file system servers and NAS servers can do interesting optimizations.

- Transparency: Network file systems attempt to provide local remote transparency so that local applications detect little or no difference between running over a network file system and a local file system.

- Heterogeneity: Network file system clients and servers are often implemented on quite different operating systems; clients access files without regard to their on-disk format. In most large enterprises, client machines running quite different operating systems access the same data on the same server at the same time. The CIFS (or NFS) network file system client that came by default with their operating system neither knows nor cares about the operating system of the server. Samba server has been ported to dozens of operating systems, yet the server operating system is mostly transparent to SMB/CIFS clients. Network file systems are everywhere, yet are not always seen when running in multi-tier storage environments. They often provide consistent file access under large web servers or database servers or media servers. A network file system server such as Samba can easily export data on other network file systems, on removable media (CD or DVD), or on a local file system (ext3, XFS, JFS)—and with far more flexibility than is possible with most cluster file systems.

Network file systems differ in fundamental ways from web clients/servers and cluster file systems.

2.1 History of SMB Protocol

Invented by Dr. Barry Feigenbaum of IBM's Boca Raton laboratory during the early development of personal computer operating system software, who briefly named it after his initials ("BAF") before changing the protocol name to "Server Message Block" or *SMB*. IBM published the initial SMB Specification book at the 1984 IBM PC Conference. A few years later a companion document, a detailed LAN Technical Reference for the NetBIOS protocol (which was used to transport SMB frames), was published. An alternative transport mechanism using TCP/IP rather than the Netbeui frames protocol was documented in RFCs 1001 and 1002 in 1987.

Microsoft, with assistance from Intel and 3Com early on, periodically released documents describing new "dialects" of the SMB protocol. The LANMAN1.0 SMB dialect became the default SMB dialect used by OS/2, and at least two other dialects were added for subsequent OS/2 versions.

In 1992 X/Open CAE Specification C209 better documented this increasingly important standard. The SMB protocol was the default network file system for DOS and Windows, but also for OS/2. IBM added Kerberos and Directory integration to the SMB protocol in its DCE DSS project in the early 1990s. A few years later Microsoft also added Kerberos security to their SMB security negotiation for their Windows 2000 products. Microsoft's Kerberos authentication encapsulated service tickets using SPNEGO in a new SMB SessionSetup variant, rather than using the original SecPkgX mechanism used by earlier SMB implementations (which had been documented by X/Open). The SMB protocol increasingly was used for purposes other than file serving, including remote server administration, network printing, networking messaging, locating network resources and security management. For these purposes support for various network interprocess communication mechanisms was added to the SMB protocol, including Mailslots, Named Pipes, and the "LANMAN RPC." Eventually more complex IPC mechanisms were built allowing encapsulating DCE/RPC traffic over SMB (even supporting complex object models such as DCOM).

In the mid 1990's the SMBFS file system for Linux was developed. Leach and Naik authored various CIFS IETF Drafts in 1997, but soon CIFS Documentation activity moved to SNIA. Soon thereafter CIFS implementations were completed for various operating systems including OS/400 and HP/UX. The CIFS VFS for Linux was included in the Linux 2.6 kernel. After nearly four years, the SNIA CIFS Technical Reference[4] was released in 2002 and included not just Microsoft extensions to CIFS, but also CIFS Unix and Mac Extensions.

In 2003 an additional set of CIFS Unix Extensions was proposed, and Linux and Samba prototype implementations were begun. By 2005 Linux client and Samba server had added support for POSIX ACLs,[3] support for POSIX[4] path names, a request to return all information needed by statfs, and support for very large read requests and very large write responses.

In April 2006 support for POSIX (rather than Windows-like) byte range lock semantics were added to the Samba server and Linux cifs client (Linux Kernel 2.6.17). Additional CIFS extensions were proposed to allow file I/O to be better POSIX compliant. In late 2006 and early 2007, joint work among four companies and the Samba team to define additional POSIX extensions to the CIFS protocol led to creation of a CIFS Unix Extensions wiki, as well as implementations of these new extensions[8] in the Linux CIFS client and Samba server (Mac client and others in progress). The CIFS protocol continues to evolve, with security and clustering extensions among the suggestions for the

[3] "POSIX ACLs" are not part of the official POSIX API. POSIX 1003.1e draft 17 was abandoned before standardization

[4] In this paper, "POSIX" refers narrowly to the file API semantics that a POSIX-compliant operating system needs to implement. When the file system uses the CIFS network file system protocol, providing POSIX-like file API behavior to applications requires extensions to the CIFS network protocol. The CIFS "POSIX" Protocol Extensions are not part of the POSIX standard, rather a set of extensions to the network file system protocol to make it easier for network file system implementations to provide POSIX-like file API semantics.

next round of extensions. As the technical documentation of these extensions improves, more formal documentation is being considered.

2.2 History of NFS Protocol

NFS version 1 was not widely distributed, but NFS version 2 became popular in the 1980s, and was documented in RFC 1094 in 1989. Approximately 10 years after NFS version 2, NFS version 3 was developed. It was documented[2] by Sun in RFC 1813 in 1995. Eight years later RFC 3530 defined NFS version 4 (obsoleting the earlier RFC 3010, and completing a nearly five year standardization process). An extension to NFS version 3, "WebNFS," documented by Sun in 1996, attempted to show the performance advantages of a network file system for Internet file traffic in some workloads (over HTTP). The discussion of WebNFS increased the pressure on other network file systems to perform better over the Internet, and may have been a factor in the renaming of the SMB protocol—from "Server Message Block" to "Common Internet File System." Related to the work on NFS version 4 was an improvement to the SunRPC layer that NFS uses to transport its PDUs. The improved RPCSECGSS allowed support for Kerberos for authentication (as does CIFS), and allows negotiation of security features including whether to sign (for data integrity) or seal (for data privacy) all NFS traffic from a particular client to a particular server. The NFS working group is developing additional extensions to NFS (NFS version 4.1, pNFS, NFS over RDMA, and improvements to NFS's support for a global namespace).

The following shows new protocol operations introduced by NFS protocol versions 3 and 4:

NFS VERSION 2 Operations:

- GETATTR 1
- SETATTR 2
- ROOT 3
- LOOKUP 4
- READLINK 5
- WRITE 8
- CREATE 9
- REMOVE 10
- RENAME 11
- LINK 12
- SYMLINK 13
- MKDIR 14
- RMDIR 15
- READDIR 16
- STATFS 17

New NFS VERSION 3 Operations:

- ACCESS 4
- READ 6
- MKNOD 11
- READDIRPLUS 17
- FSSTAT 18
- FSINFO 19
- PATHCONF 20
- COMMIT 21

New NFS Version 4 Operations

- CLOSE 4

- DELEGPURGE 7
- DELEGRETURN 8
- GETFH 10
- LOCK 12
- LOCKT 13
- LOCKU 14
- LOOKUPP 16
- NVERIFY 17
- OPEN 18
- OPENATTR 19
- OPEN-CONFIRM 20
- OPEN-DOWNGRADE 21
- PUTFH 22
- PUTPUBFH 23
- PUTROOTFH 24
- RENEW 30
- RESTOREFH 31
- SAVEFH 32
- SECINFO 33
- SETATTR 34
- SETCLIENTID 35
- SETCLIENTID-CONFIRM 36
- VERIFY 37
- RELEASE-LOCKOWNER 39

3 Current Network File System Alternatives

Today there are a variety of network file systems included in the Linux kernel, which support various protocols including: NFS, SMB/CIFS, NCP, AFS, and Plan9. In addition there are two cluster file systems now in the mainline Linux kernel, OCFS2 and GFS2, and a few popular kernel cluster file systems for Linux that are not in mainline (including Lustre and IBM's GPFS). The cifs and nfs file system clients for Linux are surprisingly similar in size (between 20 and 30 thousand lines of code) and change rate. The most common SMB/CIFS server for Linux is Samba, which is significantly larger than the Linux NFS server in size and scope. The most common Linux NFS server is of course nfsd, implemented substantially in kernel.

Windows Vista also includes support for various network file system protocols including SMB/CIFS, SMB2 and NFS.

4 SMB2 Under the Hood

The SMB2 protocol differs[7] from the SMB and CIFS protocols in the following ways:

- The SMB header is expanded to 64 bytes, and better aligned. This allows for increased limits on the number of active connections (uid and tids) as well as the number of process ids (pids).
- The SMB header signature string is no longer 0xFF followed by "SMB" but rather 0xFE and then "SMB." In the early 1990s, LANtastic did a similar change in signature string (in that case from "SMB" to "SNB") to distinguish their requests from SMB requests.

- Most operations are handle based, leaving Create (Open/Create/OpenDirectory) as the only path based operation.
- Many redundant and/or obsolete commands have been eliminated.
- The file handle has been increased to 64 bits.
- Better support for symlinks has been added. Windows Services for Unix did not have native support for symlinks, but emulated them.
- Various improvements to DFS and other miscellaneous areas of the protocol that will become usable when new servers are available.
- "Durable file handles"[10] allowing easier reconnection after temporary network failure.
- Larger maximum operation sizes, and improved compound operation ("AndX") support also have been claimed but not proved.

Currently 19 SMB2 commands are known:

- 0x00 NegotiateProtocol
- 0x01 SessionSetupAndX
- 0x02 SessionLogoff
- 0x03 TreeConnect
- 0x04 TreeDisconnect
- 0x05 Create
- 0x06 Close
- 0x07 Flush
- 0x08 Read
- 0x09 Write
- 0x0A Lock
- 0x0B Ioctl
- 0x0C Cancel
- 0x0D KeepAlive
- 0x0E Find
- 0x0F Notify
- 0x10 GetInfo
- 0x11 SetInfo
- 0x12 Break

Many of the infolevels used by the GetInfo/SetInfo commands will be familiar to those who have worked with CIFS.

5 POSIX Conformance

5.1 NFS

NFS version 3 defined 21 network file system operations (four more than NFS version 2) roughly corresponding to common VFS (Virtual File System) entry points that Unix-like operating systems require. NFS versions 2 and 3 were intended to be idempotent (stateless), and thus had difficulty preserving POSIX semantics. With the addition of a stateful lock daemon, an NFS version 3 client could achieve better application compatibility, but still can behave differently[6] than local file systems in at least four areas:

- Rename of an open file. For example, the "silly rename" approach often used by nfs clients for renaming open files could cause `rm -rf` to fail.

- Deleting an existing file or directory can appear to fail (as if the file were not present) if the request is retransmitted.
- Byte range lock security (Since these services are distinct from the nfs server, both lockd and statd have had problems in this area).
- write semantics (when caching was done on the client).

NFS also required additional protocol extensions to be able to support POSIX ACLs, and also lacked support for xattrs (OS/2 EAs), creation time (birth time), nanosecond timestamps, and certain file flags (immutable, append-only etc.). Confusingly, the NFS protocol lacked a file open and close operation until NFS version 4, and thus could only implement a weak cache consistency model.

5.2 NFSv4

NFS version 4, borrowing ideas from other protocols including CIFS, added support for an open and close operation, became stateful, added support for a rich ACL model similar to NTFS/CIFS ACLs, and added support for safe caching and a wide variety of extended attributes (additional file metadata). It is possible for an NFS version 4 implementation to achieve better application compatibility than before without necessarily sacrificing performance.

5.3 CIFS

The CIFS protocol can be used by a POSIX compliant operating system for most operations, but compensations are needed in order to properly handle POSIX locks, special files, and in order to approximate reasonable values for the mode and owner fields. There are other problematic operations that, although not strictly speaking POSIX issues, are important for a network file system in order to achieve true local remote transparency. They include symlink, statfs, POSIX ACL operations, xattrs, directory change notification (including inotify) and some commonly used ioctls (for example those used for the lsattr and chattr utilities). Without protocol extensions, the CIFS protocol can adequately be used for most important operations but differences are visible as seen in figure 1.

5.4 CIFS with Unix Protocol Extensions

As can be seen in figure 2, with the CIFS Unix Extensions it is possible to more accurately emulate local semantics for complex applications such as a Linux desktop.

The Unix Extensions to the CIFS Protocol have been improved in stages. An initial set, which included various new infolevels to TRANSACT2 commands in the range from 0x200 to 0x2FF (inclusive), was available when CAP_UNIX was included among the capabilities returned by the SMB negotiate protocol response.

Additional POSIX extensions are negotiated via a get and set capabilities request on the tree connection via a Unix QueryFSInfo and SetFSInfo level. Following is a list of the capabilties that may be negotiated currently:

- CIFS_UNIX_FCNTL_LOCKS_CAP
- CIFS_UNIX_POSIX_ACLS_CAP
- CIFS_UNIX_XATTR_CAP
- CIFS_UNIX_EXATTR_CAP

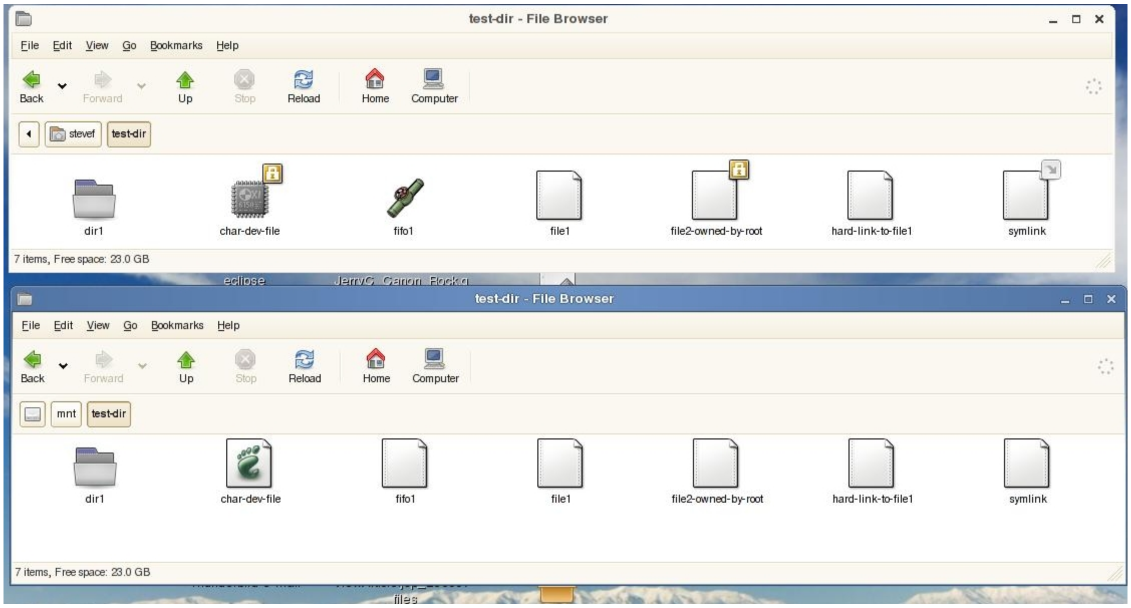

Figure 1: Without extensions to CIFS, local (upper window) vs. remote (below) transparency problems are easily visible

- CIFS_UNIX_POSIX_PATHNAMES_CAP (all except slash supported in pathnames)
- CIFS_UNIX_POSIX_PATH_OPS_CAP

A range of information levels above 0x200 has been reserved by Microsoft and the SNIA CIFS Working Group for Unix Extensions. These include Query/SetFileInformation and Query/SetPathInformation levels:

- QUERY_FILE_UNIX_BASIC 0x200 Part of the initial Unix Extensions
- QUERY_FILE_UNIX_LINK 0x201 Part of the initial Unix Extensions
- QUERY_POSIX_ACL 0x204 Requires CIFS_UNIX_POSIX_ACL_CAP
- QUERY_XATTR 0x205 Requires CIFS_UNIX_XATTR_CAP
- QUERY_ATTR_FLAGS 0x206 Requires CIFS_UNIX_EXTATTR_CAP
- QUERY_POSIX_PERMISSION 0x207
- QUERY_POSIX_LOCK 0x208 Requires CIFS_UNIX_FCNTL_CAP
- SMB_POSIX_PATH_OPEN 0x209 Requires CIFS_UNIX_POSIX_PATH_OPS_CAP
- SMB_POSIX_PATH_UNLINK 0x20a Requires CIFS_UNIX_POSIX_PATH_OPS_CAP
- SMB_QUERY_FILE_UNIX_INFO2 0x20b Requires CIFS_UNIX_EXTATTR_CAP

Currently the CIFS Unix Extensions also include the following Query/SetFileSystemInformation levels that allow retrieving information about a particular mounted export ("tree connection"), and negotiating optional capabilities. Note that unlike NFS and SMB/CIFS, the CIFS Unix Extensions allow different capabilities to be

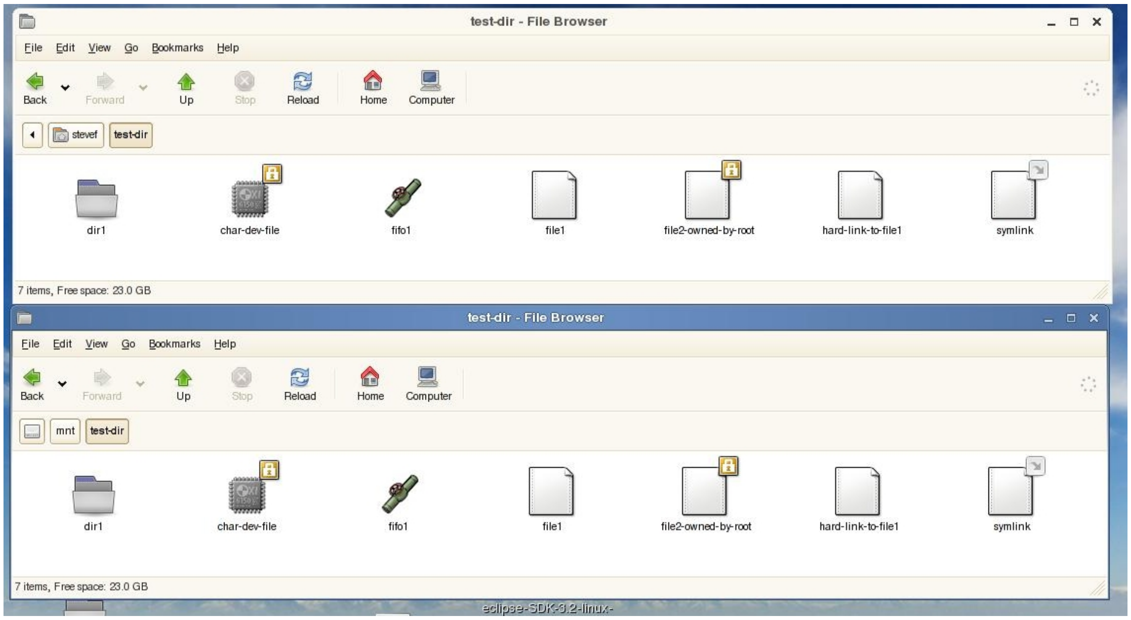

Figure 2: Better local (upper window) vs. remote (below) transparency with CIFS Unix extensions

negotiated in a more granular fashion, by "tree connection" rather than by server session.

If a server is exporting resources located on two very different file systems, this can be helpful.

- SMB_QUERY_CIFS_UNIX_INFO 0x200 (Part of the original Unix Extensions)
- SMB_QUERY_POSIX_FS_INFO 0x201
- SMB_QUERY_POSIX_WHO_AM_I 0x202

These Unix Extensions allow a CIFS client to set and return fields such as uid, gid and mode, which otherwise have to be approximated based on CIFS ACLs, and also to drastically reduce the number of network roundtrips, the number of operations required for common path based operations. For example, with the older CIFS Unix Extensions, a file create operation takes many network operations: QueryPathInfo, NTCreateX, SetPathInfo, QueryPathInfo in order to implement local Unix create semantics correctly. File creation can be done in one network roundtip using the new SMB_POSIX_PATH_OPEN, which reduces latency and allows the server to better optimize. The improved atomicity of mkdir and create makes error handling easier (e.g., in case a server failed after a create operation, but before the SetPathInfo).

5.5 SMB2

The SMB2 protocol does improve upon its predecessors in including symlink support, but retrieving mode and Unix uid and gid from NTFS/CIFS ACLs is awkward, and SMB2 appears only slightly improved in this area, and substantially worse than the CIFS Unix Extensions for this purpose.

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)

6 Performance

CIFS has often been described as a "chatty" protocol, implying that it is inherently slower than NFS, but this is misleading. Most chattiness in observed behavior of CIFS is the result of differences between the operating system implementations being compared (e.g., Windows vs. Linux). Another factor that leads to the accusation of the CIFS protocol being "chatty" (wasteful of network bandwidth) is due to periodic broadcast frames that contain server announcements (mostly in support of the Windows Network Neighborhood). These are not a required part of CIFS, but are commonly enabled on Windows servers so that clients and/or "Browse Masters" contain current lists of the active servers in a resource domain.

There are differences between these protocols that could significantly affect performance though, and these include: compound operations, maximum read and write sizes, maximum number of concurrent operations, endian transformations, packet size, field alignment, difficult to handle operations, incomplete operations that require expensive compensations.

To contrast features that would affect performance it is helpful to look at some examples.

6.1 Opening an existing file

The SMB2 implementation needs a surprising eight requests to handle this simple operation.

6.2 Creating a new file

The SMB2 protocol appears to match perfectly the requirements of the Windows client here. Attempting a simple operation like:

```
echo new file data > newfile
```

results in the minimum number of requests that would reasonably be expected (opencreate, write, close). Three requests and three responses (823 bytes total).

6.3 Mount (NET USE)

Once again the SMB2 protocol appears to match well the requirement of the client with only 11 requests (four are caused by the Windows desktop trying to open `Desktop.ini` and `AutoRun.inf`).

7 Linux Implementation

Much of the progress on SMB2 has been due to excellent work by the Samba 4 team, led by Dr. Andrew Tridgell. Over the past year and a half, they have implemented a comprehensive client library for SMB2, implemented a test suite (not as comprehensive yet), implemented DCE/RPC over SMB2 (for remote administration), implemented a SMB2 server (not complete), and in cooperation with Ronnie Sahlberg, implemented a wireshark (ethereal) protocol analyzer.

8 Future Work and Conclusions

Although great progress has been made on a prototype user space client in Samba 4, an implementation of SMB2 in kernel on Linux also needs to be completed, and we have begun a prototype. The SMB2 protocol represents a modest improvement over the older SMB/CIFS protocol, and should be slightly better despite the slightly larger frame size caused by the larger header. With fewer commands to optimize and better aligned fields, performance may be slightly improved as server developers better tune their SMB2 implementations.

Despite the addition of support for symlinks, the SMB2 protocol lacks sufficient support for features needed by Unix and Linux clients. Adding Unix extensions to SMB2, similar to what has been done with CIFS, would be possible and could reuse some of the existing Unix specific infolevels.

With current Linux kernels, NFS version 4 and CIFS (cifs client/Samba server) are good choices for network file systems for Linux to Linux. NFS performance for large file copy workloads is better, and NFS offers some security options that the Linux cifs client does not, but in heterogeneous environments that include Windows clients and servers, Samba is often much easier to configure.

9 Acknowledgements

The author would like to express his appreciation 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, and of course thanks to the Linux NFSv4 developers and the NFS RFC authors, for implementing and documenting such a complex protocol. Thanks to Dr. Mark French for pointing out some of the many grammar errors that slipped through.

10 Legal Statement

References

[1] Sun Microsystems Inc. RFC 1094: NFS: Network File System Protocol Specification. March 1989. See `http://www.ietf.org/rfc/rfc1094.txt`

[2] Callaghan et al. RFC 1813: NFS Version 3 Protocol Specification. June 1995. See `http://www.ietf.org/rfc/rfc1813.txt`

[3] S. Shepler, et al. RFC 3530: Network File System (NFS) version 4 Protocol. April 2003. See `http://www.ietf.org/rfc/rfc3530.txt`

[4] J. Norton, et al. SNIA CIFS Technical Reference. March 2002. See `http://www.snia.org/tech_activities/CIFS/CIFS-TR-1p00_FINAL.pdf`

[5] C. Hertel. Implementing CIFS. 2004. See `http://ubiqx.org/cifs/`

[6] O. Kirch. Why NFS Sucks. *Proceedings of the 2006 Ottawa Linux Symposium*, Ottawa, Canada, July 2006.

[7] Dr. A. Tridgell. Exploring the SMB2 Protocol. SNIA Storage Developer Conference. September 2006. http://samba.org/~tridge/smb2.pdf

[8] CIFS Unix Extensions. http://wiki.samba.org/index.php/UNIX_Extensions

[9] Linux CIFS Client and Documentation. http://linux-cifs.samba.org

[10] What's new in SMB in Windows Vista http://blogs.msdn.com/chkdsk/archive/2006/03/10/548787.aspx