

The future of CIFS

Jelmer Vernooij jelmer@samba.org



Samba Team

22 April 2004 — Initial document

28 May 2004 — Spelling updates and small corrections

2 June 2004 — Couple of fixes, reported by Jean-Baptiste Marchand

Abstract

“SMB” (also known as “CIFS”) is a file-sharing protocol that has been used since the mid-eighties. Most people know SMB as the protocol behind the “Network Neighbourhood” and remote printing in Windows.

This paper describes the history of SMB, gives some general information about the protocol itself, and finally, it discusses the future of CIFS, technologies that will replace it and their impact on the Unix environment.

Several parts of the protocol are not discussed in this paper, such as mailslots, browsing and dfs, to prevent it from getting to complex.

The slides used when this paper was presented are available at <http://jelmer.vernastok.nl/slides/nluug-vj04.pdf>.

This paper was awarded “Best Paper” at the NLUUG 2004 Spring Conference.

Contents

1	A little bit of history	2
1.1	Invention by IBM and Microsoft	2
1.2	Samba’s start	3
2	The protocol itself	3
2.1	Network analysing	4
2.2	The NFS/CIFS marketing war	4
2.3	Standards?	4
2.4	Packet format	5
2.5	RPC calls	6
3	Active directory	6
4	Samba 4	7
4.1	Subsystem dependencies	7
4.2	Better architecture for new technology	7
4.3	Release time-frame	8
5	Longhorn and Blackcomb	8
5.1	Backwards compatibility	8

IPX	IP	SNA	DECNet		
NetBIOS				TCP/IP	NetBEUI
SMB					
SMB Pipes					Mailslots
RPC				RAP	Browsing
...	DCOM	NT Printing	Registry	9x Printing	

Figure 1: The SMB protocol layers

5.2 Unix compatibility	8
6 The death of CIFS?	9

1 A little bit of history

1.1 Invention by IBM and Microsoft

SMB is not very old, but it has a long history of modifications and extensions. The original protocol was meant to run over “NetBIOS”, which was the name of the DOS interface to a very simple LAN system developed by IBM. NetBIOS was developed because SNA, IBM’s other main protocol at the time, was much too advanced for use in DOS.

The NetBIOS API in these days (early eighties) was nothing more than the interface to a very simple link-layer protocol over which several protocols, including SMB, were used. It could do reads and writes to services on remote hosts, which were identified by case-insensitive names, and discover all available hosts and services.

Dr. Barry Feigenbau, an IBM employee, invented the core of the original SMB protocol, which he initially named after himself: “BAF”. He later changed the name to be “SMB” (for “Server Message Block”). Every packet in the protocol starts with a byte *0xFF* and these three letters.

IBM, Microsoft, 3Com and Intel made up the rest of the initial protocol together. The commands the protocol supported at this stage were basically a mirror of the DOS File IO API calls, which meant the protocol wasn’t very efficient. The protocol also lacked authentication support. Everybody on the network could do reads and writes, which meant this protocol wasn’t very suitable for large enterprises.

NetBIOS is an API that has had various implementations; there is NetBIOS over TCP/IP (NBT), NetBIOS over IPX, NetBIOS over SNA and even NetBIOS over DECNet. Mostly used these days is NetBIOS over TCP (NBT).

This is also where things are starting to get hairy. Since NetBIOS identifies hosts by their name, NetBIOS clients had to start doing IP broadcasts to figure out the IP of the host they had to connect to. Several schemes were introduced to do name lookups crossing subnet boundaries, using name servers, etc. We're basically emulating a NetBIOS LAN in order to be able to run SMB.

Doing NetBIOS over IP is not very sane, however, the NBT implementation itself in Windows isn't very nice either. It has horrible limits, special exceptions, several broken schemes for looking up names (including two kinds of name servers). NetBIOS and NetBIOS over TCP/IP are described in RFC1001 and RFC1002.

After the "CORE" dialect, IBM and Microsoft implemented a new dialect known as "LANMAN". This dialect was used by Windows for Workgroups, OS/2 and Windows 9x which all know it under a different name. A 'virtual' file system was also added, which was used for doing remote function calls (RAP, for "Remote Administration Protocol").

For Windows NT, yet another dialect was added, named 'NT'. The NT dialect had its own set of file I/O functions (similar to the NT File I/O API) and it had support for yet another way of doing remote function calls: DCE/RPC. RPC's are used for DCOM and several of the subsystems in NT that can be accessed remotely (registry, printing, user management, logging on, etc).

In Windows NT 5 (marketing name: Windows 2000), NetBIOS-less SMB was introduced. This means SMB is used directly over TCP port 445 instead of via NetBIOS over TCP/IP. DNS is used for looking up machine names.

1.2 Samba's start

Andrew "tridge" Tridgell wrote the initial version of Samba somewhere at the end of '91 when he was a PhD student at the ANU in Canberra. He figured out the protocol by writing a program that would "sniff" what was sent on the network. Out of what he figured out, he wrote a program for Ultrix, running on an old DECserver.

The initial version of Samba had the inspiring name 'server'. Server 0.1 was released in the beginning of '92.

After a threat from a company named "Syntax", which claimed that they had the sole right to use the name "smb server", Andrew egrepped for words containing the letters S, M and B in */usr/share/dicts* and he immediately liked the word "Samba", which turned up as a result.

Samba has evolved a lot since '93, though the infrastructure has mostly remained the same. If you look at the source code of smbclient and smbd from Samba 3.0, you will see that it still contains an awful lot of code from the original server program!

At the moment, the Samba team is a loose-knit group of about 30 people from around the world, with 10 to 15 doing development on an active basis.

"Server-1.0" actually still works with modern NT workstations. Even though so much things have changed in the protocol, newer clients still have support for very old dialects. Backwards compatibility at the price of a bloated client!

2 The protocol itself

As can be seen in figure 1, the protocol has quite some layers. The reason for all these layers is a historical one: it's easier to extend some existing stuff than to build it up from scratch. Luckily,

Windows 2000 added support for running SMB over plain TCP/IP, using DNS for name lookups.

As was explained before, the protocol has several dialects. New Windows servers still support all of the old dialects. Implementing a 'full' SMB server that can handle all types of clients is very hard because of the code size.

2.1 Network analysing

Samba has been developed by “network analysis” of the SMB protocol. Some of the developers have done an outstanding job by figuring out what the data types and function calls on the wire are. Some of the other parts of the protocol that caused us headaches were the various crypto parts. Various developers did a very nice job by dissecting the sign and encrypting parts of the protocol.

Note that network analysis is not the same thing as reverse engineering.

A nice metaphor for understanding the difference between the two is learning a foreign language (say, French). French can be learned by sitting in a French café and trying to understand what people are saying to each other. Or it can be learned by taking a scalpel and using it on the waiter...

The difference between reverse engineering and network analysis is very much like that. For Samba development, we don't do any reverse engineering (i.e. decompilation), just network analysis ('sniffing') using ethereal and tcpdump.

2.2 The NFS/CIFS marketing war

During the internet hype in the nineties, Sun and Microsoft got in a fight about which remote file system API was going to make it. Sun was promoting NFS, Microsoft was promoting SMB.

In order to get SMB supported by other vendors, Microsoft did a couple of things:

- They renamed SMB to CIFS. CIFS stands for “Common Internet File System”. This name change was just for marketing purposes, there is no real technical difference between the two
- Several proposed standards were published explaining the syntax of the protocol. They have all expired by now.

Samba even got donations from Microsoft during this period, including funding for trips to conferences and MSDN donations. Microsoft developers were encouraged to work with Samba developers to get a working implementation of a SMB server and client on Unix.

Microsoft won the war. CIFS became the standard (for LANs, at least). After this, they lost interest in having other vendors support CIFS. Rather, they tried to get everybody to use their products.

2.3 Standards?

SMB has become ugly over the years. Since Microsoft only tests against their own products, several scenarios aren't tested. Certain functions aren't used by Microsoft or in different order, so unless 3rd party implementations behave just like Microsoft does, they can expect their clients and servers to break against their implementation.

Figure 2: An SMB packet, as dissected by ethereal

```
⊞ Frame 1 (156 bytes on wire, 156 bytes captured)
⊞ Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:00:00:00:00:00
⊞ Internet Protocol, Src Addr: 127.0.0.1 (127.0.0.1), Dst Addr: 127.0.0.1 (127.0.0.1)
⊞ Transmission Control Protocol, Src Port: 32776 (32776), Dst Port: microsoft-ds (445), Seq: 0, Ack: 0, Len: 90
⊞ NetBIOS Session Service
⊞ SMB (Server Message Block Protocol)
  ⊞ SMB Header
    Server Component: SMB
    Response in: 2
    SMB Command: Trans2 (0x32)
    NT Status: STATUS_SUCCESS (0x00000000)
    ⊞ Flags: 0x08
    ⊞ Flags2: 0xc001
    Process ID High: 0
    Signature: 0000000000000000
    Reserved: 0000
    Tree ID: 1
    Process ID: 2791
    User ID: 100
    Multiplex ID: 8
  ⊞ Trans2 Request (0x32)
```

I.e. in some of the the RAP calls, you can specify you want 30 characters of a share name, but you're going to get back only 13 of them anyway. And Microsoft clients will not like it if they get 30 characters.

Since most of the clients out in the wild are from Microsoft at the moment, Microsoft gets to decide what the "standard" is in CIFS. There are several extensions in SMB by non-Microsoft parties (such as the HP Unix extensions), but they are hardly useful unless Microsoft will put support for them in Windows . If Windows implements a certain call with a bug, other implementations have to implement the same bug, because they might break windows support (which means breaking support with 95% of all the clients out there).

In other words: There is no standard but what Windows does.

2.4 Packet format

The protocol works with clients and servers. A client does a request and the server replies. The server is usually sharing data/printers/etc.

Every connection starts with a negotiation between the server and the client, in which they tell each other which dialects they support and later tell each other which dialect they are going to use.

Then there's the authentication negotiation. Because of concerns with security, the security format has been upgraded several times. Older clients use either plain text passwords, simple hashed passwords (LM hashes) or somewhat harder hashed passwords (NT hashes). Windows 2000 and later can also use a combination of several authentication layers, such as Kerberos, SASL, NTLMSSP, GSS-API and SPNEGO. We have seen Kerberos inside GSS-API inside SASL inside SPNEGO on the wire.

The header of each SMB packet contains information about the command that is being executed, the kind of errors that are returned (DOS, Windows, NT), whether or not Unicode is used, and whether signing or encryption is used.

Currently, over a 100 regular SMB calls are known to exist. There are over 120 RPC calls in the basic pipes.

Writing a CIFS client implementation is a lot easier than writing a server implementation, because the client can choose which few commands of the whole lot that are available it uses. A server has to implement all calls, because there's no way to know what calls a client is going to use.

CIFS has a very neat system called 'oplocks', which stands for "Opportunistic Locks". This allows clients to lock a part of a file and getting the permission to be the only one using that range, so that it can cache the contents of the file when doing reads and writes.

There are three kinds of oplocks: read, write and batch. Batch oplocks are basically read oplocks which stay open a few seconds after they are closed and they exist because for some reason, DOS closes batch files for every line in them it executes...

2.5 RPC calls

RPC calls are used mostly for non-file I/O things in later version of Windows.

At the moment, the most important known RPC "pipes" are:

- atsvc: Run a specified command at a specified time, much like the Unix "at" command
- lsarpc: Security information
- browser: Modern version of browsing
- eventlog: System logging
- epmapper: RPC endpoint mapping
- samr: User management
- w32time: Time server
- netdfs: Distributed File System support
- srvsvc: Server administration
- wkssvc: Workstation administration
- winreg: Registry
- spoolss: Printing
- netlogon: Logging on, joining domains, ...

Samba-3 currently has most of these implemented. For Samba-4, we're working on full coverage.

3 Active directory

One of the most recent changes in the Windows networking world is the introduction of Active Directory. This technology was first supported by Windows 2000 and is also used in later versions of the OS.

Active Directory is more than just CIFS. It is a collection of protocols with certain proprietary extensions. CIFS is among these protocols, for file sharing and RPC's, but it also includes DNS, LDAP, Kerberos, DHCP and something new called 'CLDAP'.

Problem with using Active Directory in a Unix world is the fact that Microsoft has added extensions to these protocols, extensions they rely on. This makes it quite hard to support Active Directory from a Samba point of view — we need either our own, or patched versions of DHCP, Kerberos,

LDAP, and bind. And all the daemons for these protocols need to communicate with each other as they are closely tied to each other in AD.

CLDAP is something new. It's a bit like LDAP over UDP, with dynamic content. CLDAP is even somewhat like RPC, and replaces it in some of the newer calls.

At the moment, Samba-3 has support for being a member of an Active Directory domain.

4 Samba 4

The Samba team is currently working on the next major version of Samba, version 4.0. Version 4 will be a restructured version of the code base, with a large number of subsystems rewritten.

While Samba was initially built as a file server, several other functions were later added on top of it that, such as domain control and print serving. This means the current infrastructure is somewhat flawed for what we're trying to do.

4.1 Subsystem dependencies

Other problems with the pre-Samba4 architecture is the fact that the subsystems weren't written to be independent of each other. At first this didn't matter, as Samba's only executable was `smbd`, but these days, there are about a dozen utilities in Samba, that use subsystems that all depend on each other. For example, "pdbedit", the user account management program, depends on the printing libraries! There were no clear interfaces between subsystems and there was no clear distinction between internal and external functions in subsystems.

Samba-4 tries to make code more modular — this means it will be easier for outsiders to use and understand subsystems, and that it will be easier to change interfaces when we have to. It should also severely reduce the sizes of the binaries.

4.2 Better architecture for new technology

In Samba-3, all RPC marshalling and UN-marshalling code was written manually. In the Microsoft world, all this code is generated from IDL code, using the `midl` compiler. Samba-4 now has "pidl", which it uses to generate code from IDL as well. This should severely reduce the amount of effort required to implement a DCE/RPC call.

Active Directory DC support was one of the goals set for Samba 4. Functionality like an RPC endpoint mapper, which was hard to implement in Samba-3, is just a few days work in Samba-4, thanks to the infrastructure. Several other things, such as CLDAP¹, have already been implemented in Samba-4 as well.

Samba-4 will feature full NT semantics. Instead of implementing just some of the calls we need, we are implementing all calls and separating the protocol code and the mapping code.

This means that, with a modified kernel, people will be able to store NT semantics without any need for mapping.

¹I'm actually not sure what the status of CLDAP is, except that it's supposed to be in `nmbd` in Samba 4 soon

4.3 Release time-frame

At the moment, Samba 4 already works! It still lacks some important features, such as multiple user support (everything is done as root), but it is quickly being expanded. Samba 4 also implements some features which Samba-3 lacks, such as CLDAP support and support for various obscure RPC calls. It will certainly take at least 2 years before a stable version will be released.

During the 2004 edition of the SambaXP conference, it was decided that some of the rewritten subsystems in Samba 4, especially the RPC and core SMB subsystems, will be merged back into Samba 3. Samba 3.1, and later 3.2 ² will be the first releases that contain updated subsystems.

5 Longhorn and Blackcomb

For a couple of years now, work has been going on on “Longhorn”, which is going to be the next Windows version after XP. This new OS will contain quite some new features. One of the original new features, ‘WinFS’ has been partially postponed to “Blackcomb”, the successor to Longhorn. Blackcombs release is currently planned somewhere at the end of the current decade.

WinFS is supposed to replace NTFS ³. This file system is no longer hierarchical, but relational. It’s more like your hard disk is one big SQL database with arbitrary data types.

This will allow users to search for “the powerpoint presentation I made last week”, or “pictures with cats on them’“. “Folders” will still be supported, but they will rather be data types that can contain other data types then parts of the file system.

Since this file system is so fundamentally different from FAT and NTFS, applications will need to be rewritten to take advantage of the new FS.

Another major feature of WinFS is synchronisation support between various hosts.

RPC functionality, currently implemented in RPC over SMB, will be replaced by ‘Indigo’, which is a new RPC subsystem, written from scratch.

5.1 Backwards compatibility

Despite the fact that WinFS introduces very radical changes, it will still be backwards compatible with older local and remote file API’s. Win32 applications will still be able to run on Longhorn and Blackcomb.

But, even though older applications will be able to run on Longhorn and Blackcomb and access SMB, that doesn’t mean newer applications will be able to work with SMB or work as well as they would with WinFS.

5.2 Unix compatibility

Losing CIFS doesn’t necessarily have to mean losing the ability to share files and printers between Windows and Unix. The Mono folks are working on an open source implementation of the various .NET parts, including indigo. All the necessary information has been published by Microsoft as open standards.

²Samba uses the same versioning system as the Linux kernel

³I haven’t been able to figure out yet whether WinFS is going to use NTFS itself (hiding NTFS from the user), or whether NTFS is going away completely

Question is, however, whether Microsoft will keep being open and friendly with Mono.

6 The death of CIFS?

So, now that Microsoft appears to be phasing out SMB in their newer products, will that mean that SMB will die?

Old Windows clients won't go away very easily. There are still quite some businesses running NT4, even though it is almost 8 years old. Microsoft also will have to maintain some kind of backwards compatibility within their new products, so users can more easily upgrade.

Blackcomb might not be picked up by the masses. Perhaps we are all using Mac OS X or Linux by the time Blackcomb will be released ?

SMB might have a future beyond Windows. Steve French has been working on a new Linux kernel module that supports CIFS. This kernel module also supports the HP Unix extensions (just like Samba does), so it can be used for Unix ↔ Unix file system sharing without losing semantics.

Some of the NAS vendors have been considering building their own SMB client for Windows in case Microsoft's goes away. While they might be able to do this successfully using Microsoft's publicly available interfaces, this is probably not going to be as good as WinFS, as Microsoft will be making small changes in the interface with each version and service pack of Windows, thus breaking 3rd party file-systems.

References

- [1] Ethereal. <http://www.ethereal.com/>.
- [2] Linux cifsfs. <http://www.samba.org/samba/linux-cifs-client.html>.
- [3] Microsoft cifs mailing list archives. <http://discuss.microsoft.com/SCRIPTS/WA-MSD.EXE?A1=ind0404c&L=cifs>.
- [4] The mono project. <http://www.go-mono.com/>.
- [5] .net show about winfs. <http://msdn.microsoft.com/theshow/Episode041>.
- [6] Samba. <http://www.samba.org/>.
- [7] Christopher R. Hertel. *Implementing CIFS*. Prentice Hall PH, 2003. <http://www.ubiqx.org/cifs>.
- [8] Dan Shearer. *The History of SMB*. 1996. <http://samba.org/cifs/docs/smb-history.html>.
- [9] John H. Terpstra and Jelmer R. Vernooij. *The Official Samba-3 HOWTO Collection*. Prentice Hall PH, 2004. <http://www.samba.org/samba/docs>.